Proceedings of the Seminar

Machine Learning in Computer Vision and Natural Language Processing

University of Colorado, Colorado Springs

August 3, 2018

Editor: Jugal K. Kalita and Jonathan Ventura

Funded by

National Science Foundation

Preface

It is with great pleasure that we present to you the papers describing the research performed by the NSF-funded Research Experience for Undergraduates (REU) students, who spent 10 weeks during the summer of 2018 at the University of Colorado, Colorado Springs. Within a very short period of time, the students were able to choose cutting-edge projects involving machine learning in the areas of computer vision and natural language processing, write proposals, design interesting algorithms and approaches, develop code, and write papers describing their work. We hope that the students will continue working on these projects and submit papers to conferences and journals within the next few months. We also hope that it is the beginning of a fruitful career in research and innovation for all our participants.

We thank the National Science Foundation for funding our REU site. We also thank the University of Colorado, Colorado Springs, for providing an intellectually stimulating environment for research. In particular, we thank Drs. Terrance Boult and Guy Hagen, who were faculty advisors for the REU students. We also thank Alessandra Langfels and Salma Kazemi for working out all the financial and administrative details. We also thank our graduate and undergraduate students, in particular, Vinodini Venkataram and Tom Conley, for helping the students with ideas as well as systems and programming issues. Xian Tan and his team also deserve our sincere gratitude for making sure that the computing systems performed reliably during the summer. Our thanks also go to Dr. Robert Carlson of Mathematics for being a constant well-wisher and for stimulating discussions.

Sincerely,

Jugal Kalita
jkalita@uccs.edu
Professor

Jonathan Ventura
jventura@uccs.edu
Assistant Professor

August 3, 2018

Table of Contents

NSF REU Proposal Presentation Meeting

Department of Computer Science
University of Colorado, Colorado Springs
Engineering Building, Room 105
June 7, 2018: Friday

1:30-1:35 PM: Welcome Remarks by xx, College of Engineering and Applied Science

1:40-2:40 PM
Session Chair: Dr. Jonathan Ventura, Department of Computer Science, University of Colorado, Colorado Springs

Julian Medina, University of Colorado, Colorado Springs, CO: Varied and Parallel Attention Mechanics in Neural Machine Translation

Jacob Krantz, Gonzaga University, Spokane, WA: Abstractive Summarization Using Attentive Neural Techniques

Mehdi Drissi, Harvey Mudd College, Claremont, CA: Hierarchical Text Generation using GANs

Jack St. Clair, Haverford College, Haverford, PA: Reinforcement Learning and Attention Models for Multi Response Dialogue Generation

2:55-3:55 PM
Session Chair: Dr. Manuel Gunther, Department of Computer Science, University of Colorado, Colorado Springs

Krishan Rajaratnam, The University of Chicago, Chicago, IL: Defense Against Adversarial Attacks on Automatic Speech Recognition

Kieran Parikh, Middlebury College, Middlebury, VT: Distributed Representations of Authorship Style

Chance Hamilton, Florida Gulf Coast University, Fort Myers, FL: Video Frame Interpolation via Pixel Polynomial Modeling

Kayleigh Migdol, Humboldt State University, Arcata, CA and Carnegie Mellon University, Pittsburg, PA: PixelMRF: Deep Markov Random Field for Image Modeling

4:05-4:50 PM
Session Chair: Dr. Terrance Boult, University of Colorado, Colorado Springs

Alisha Sharma, University of Maryland University College, Adelphi, MD: Estimating Depth in Cylindrical Panoramic Images

Tram-Anh Nguyen, George Mason University, Fairfax, VA: Deep Learning for Denoising of Fluorescence Microscopy Images

Gia Zhuang, University of Colorado, Colorado Springs, CO: Mood Prediction from Engagement

Our Session Chairs

Dr. Jonathan Ventura is an assistant professor in the Department of Computer Science at the University of Colorado, Colorado Springs. His areas of expertise are computer vision, geometric problems such as 3D modeling and camera localization, medical image analysis, and mobile augmented reality. Dr. Ventura has a PhD from the University of California at Santa Barbara, and has published 35 papers. As an undergraduate, he was in an REU program himself at the UCSB.

Dr. Manuel Gunther received his PhD in Computer Science from the Ruhr-University Bochum, Germany in 2011, following which, he spent four years as a post-doc in Switzerland at the Idiap Research Institute. In 2015, he joined the VAST Lab at UCCS as a research associate under the supervision of Dr. Terrance Boult. His research interests include automatic face recognition, and other face processing tasks such as face detection or facial attribute prediction, as well as open source software development.

Dr. Terrance Boult is an El Pomar Endowed Chair of Communication and Computation in the Department of Computer Science at the University of Colorado, Colorado Springs. He runs the Vision and Security Technology Lab (VAST Lab), focused on projects in Security including machine learning, surveillance, biometrics, sensor networks, and distributed steganalaysis and general projects in computer vision. He also works with The El Pomar Institute for Innovation and Commercialization through which he works with many local companies.

.

NSF REU Seminar on Machine Learning
Department of Computer Science
University of Colorado, Colorado Springs
Osborne A206
August 3, 2018: Friday

10*:30-10:40 AM: Introduction by Dr. Jonathan Ventura, followed by* Welcome Remarks by *Dr. Thomas Christensen, Provost and Executive Vice Chancellor for Academic Affairs and Professor of Physics, University of Colorado, Colorado Springs, CO*

10:40-*12:15 AM Session Chair: Thomas Conley, Information Security Officer and Ph.D. student in Computer Science, Colorado Springs, CO*

- 10:40-11:05 Krishan Rajaratnam, The University of Chicago, Chicago, IL*:* Speech Coding and Audio Preprocessing for Mitigating and Detecting Audio Adversarial Examples on Automatic Speech Recognition
- 11:05-11:30 Kieran Parikh, Middlebury College, Middlebury, VT*:* Towards a Universal Document Encoder for Authorship Attribution
- 11:30-11:55 Julian Medina, University of Colorado, Colorado Springs, CO*:* Parallel Attention Mechanisms in Neural Machine Translation
- 11:55-12:20 Jacob Krantz, Gonzaga University, Spokane, WA*:* Abstractive Summarization Using Attentive Neural Techniques

12:20-1:15 PM: Lunch

1:15-2:55 PM *Session Chair: Dr. Janet Burge, Associate Professor of Computer Science, Colorado College, Colorado Springs, CO*

- 1:15-1:40 Mehdi Drissi, Harvey Mudd College, Claremont, CA: Hierarchical Text Generation using an Outline
- 1:40-2:05 Jack St. Clair, Haverford College, Haverford, PA: Impact of Auxiliary Loss Functions on Dialogue Generation using Mutual Information
- 2:05-2:30 Gia Zhuang, University of Colorado, Colorado Springs, CO: Engagement Based Mood Prediction
- 2:30-2:55 Kayleigh Migdol, Humboldt State University, Arcata, CA and Carnegie Mellon University, Pittsburgh, PA*:* PixelMRF: A Convolutional Markov Random Field for Image Generation

2:55-3:10 PM: Break

3:10-4:25 PM *Session Chair: Vinodini Venkataram, M.S. Student in Computer Science, University of Colorado, Colorado Springs, CO*

- 3:10-3:35 Chance Hamilton, Florida Gulf Coast University, Fort Myers FL: Video Frame Interpolation via Pixel Polynomial Modeling
- 3:35-4:00 Tram-Anh Nguyen, George Mason University, Fairfax, VA*:* Deep Learning for Signal to Noise Enhancement of Fluorescence Microscopy Images
- 4:00-4:25 Alisha Sharma, University of Maryland University College, Adelphi, MD: Estimating Depth in Cylindrical Panoramic Images

4:30 PM: Closing Remarks by *Dr. Terrance Boult*

Our Session Chairs and Guests

Dr. Thomas M. Christensen is Provost and Executive Vice Chancellor for Academic Affairs at the University of Colorado at Colorado Springs. Tom Christensen joined the faculty of the University of Colorado at Colorado Springs Department of Physics and Energy Science in 1989. He has served the campus as a faculty member, department chair, associate dean and dean. Dr. Christensen has received both the College (1993) and campus (1996) Outstanding Teaching Awards and the Chancellor's Award (2003) to recognize his service and teaching. Dr. Christensen's research in experimental surface physics has led to 21 published papers in international science journals and over 90 presentations at scientific meetings. He has been the principal investigator on over $0.5 million in research grants and contracts for work in surface physics and in science education. In his spare time, Dr. Christensen plays string bass with the Pikes Peak Philharmonic orchestra and bass guitar with the Physics Classic Rock and Roll Orchestra.

Dr. Janet Burge is an Associate Professor of Computer Science at Colorado College. Her research area is in Design Rationale, methods for capturing and using the reasons behind decisions made when designing software or any other artifact. She is interested in this area because successful software systems often outlast the tenure of their developers, which means critical knowledge can be lost forever if there is no way to retrieve and use it. Dr. Burge was awarded the NSF CAREER Award in 2009. She had taught at Wesleyan University and Miami University in Ohio, before moving to Colorado College in 2017.

Thomas Conley has been a computer programmer for more than 25 years and has worked in many domains. For the last 15 years he has specialized in Information Security at Ohio University and UCCS. He has been an instructor in CS here UCCS and has recently entered the PhD program in Computer Science where he will concentrate on computational linguistics.

Vinodini Venkataram is an M.S. student at the University of Colorado, Colorado Springs. currently working on her M.S. thesis. She presented a paper titled "Open set Text Classification using Convolutional Neural Networks" at ICON (International Conference on Natural Language Processing) held in Dec 2017 (India).

NSF REU Midsummer Meeting

Department of Computer Science
University of Colorado, Colorado Springs
UCCSTeach Room, A343, Osborne Building
July 6, 2018: Friday

1:30-1:35 PM: Welcome Remarks

1:40-2:40 PM
Session Chair: Steve Cruz, Department of Computer Science, University of Colorado, Colorado Springs

Gia Zhuang, University of Colorado, Colorado Springs, CO: Engagement Based Mood Prediction

Tram-Anh Nguyen, George Mason University, Fairfax, VA: Deep Learning for Signal to Noise Enhancement of Fluorescence Microscopy Images

Alisha Sharma, University of Maryland University College, Adelphi, MD: Estimating Depth in Cylindrical Panoramic Images

Kayleigh Migdol, Humboldt State University, Arcata, CA and Carnegie Mellon University, Pittsburgh , PA: PixelMRF: Modeling Markov Random Fields with Convolutional Neural Networks

2:55-3:40 PM
Session Chair: Marc Moreno Lopez, Department of Computer Science, University of Colorado, Colorado Springs

Jack St. Clair, Haverford College, Haverford, PA: Dialogue Generation with Reinforcement Learning and Attention

Kieran Parikh, Middlebury College, Middlebury, VT: Distributed Representations of Authorship Style

Krishan Rajaratnam, The University of Chicago, Chicago, IL: Audio Adversarial Examples and Voice over IP Compression: Mitigation through Speech Coding

3:55-4:55 PM
Session Chair: Joseph Worsham, Department of Computer Science, University of Colorado, Colorado Springs

Julian Medina, University of Colorado, Colorado Springs, CO: Parallel Attention Mechanisms in Neural Machine Translation

Jacob Krantz, Gonzaga University, Spokane, WA: Abstractive Summarization Using Attentive Neural Techniques

Mehdi Drissi, Harvey Mudd College, Claremont, CA: Hierarchical Text Generation based on an Outline

Chance Hamilton, Florida Gulf Coast University, Fort Myers FL: Video Frame Interpolation via Pixel PolyNet

Our Session Chairs

Steve Cruz is a doctoral student at the University of Colorado, Colorado Springs. Steve received his Bachelor of Innovation degree in Computer Security from the University of Colorado, Colorado Springs in 2017. His research interests sare in Computer Vision and Machine Learning. Specific areas include Open-Set Recognition, Face Recognition, and Incremental Learning. He has published 3 papers in the past two years.

Marc Moreno Lopez is a Ph.D. student at the University of Colorado, Colorado Springs. Marc received his BS in Computer Engineering from Universitat Politecnica de Catalunya in 2015, and his MS in Computer Science from the University of Colorado, Colorado Springs in 2017. Marc has published one paper in the last year, "Dilated convolutions for brain tumor segmentation in MRI scans" at the International MICCAI Brainlesion Workshop (Quebec City, Canada)

Joseph Worsham is working on his MS thesis at the University of Colorado, Colorado Springs. He will present a paper titled “Genre Identification and the Compositional Effect of Genre in Literature” at the prestigious COLING Conference to be held in Santa Fe, New Mexico, in August 2018. He received his BS in Computer Science from the University of Colorado, Colorado Springs in 2014 and is a full-time employee in Lockheed Martin, doing research in machine learning.

.

Speech Coding and Audio Preprocessing for Mitigating and Detecting Audio Adversarial Examples on Automatic Speech Recognition

Krishan Rajaratnam
The College
The University of Chicago
Chicago, Illinois, USA
Email: krajaratnam@uchicago.edu

Basemah Alshemali
Department of Computer Science
University of Colorado
Colorado Springs, Colorado, USA
Email: balshema@uccs.edu
Department of Computer Science
Taibah University
Al-Medina, KSA

Kunal Shah
Department of Biology
University of Florida
Gainesville, Florida, USA
Email: kshah1997@ufl.edu

Jugal Kalita
Department of Computer Science
University of Colorado
Colorado Springs, Colorado, USA
Email: jkalita@uccs.edu

Abstract

An adversarial attack is an exploitative process in which minute changes are made to a natural input, causing that input to be misclassified by a neural model. Due to recent trends in speech processing, this has become a noticeable issue in speech recognition models. In late 2017, an attack was shown to be quite effective against the Speech Commands classification model. Limited-vocabulary classifiers, such as the Speech Commands model, are used quite frequently for managing automated attendants in traditional telephony and voice over IP (VoIP) contexts. As such, this research examines the effectiveness of VoIP speech coding in mitigating audio adversarial attacks when compared to more primitive forms of audio preprocessing and shows that an ensemble defense in tandem with speech coding is more robust than other forms of preprocessing defenses in mitigating adversarial examples. This research also proposes a new metric for evaluating preprocessing defenses against adversarial attacks. Additionally, this research explores using speech coding and various other forms of preprocessing for detecting adversarial examples.

***Index Terms*—adversarial attack, speech recognition, deep learning, audio compression, speech coding**

Introduction

The growing use of deep learning models necessitates that those models be accurate, robust, and secure. However, these models are not without exploitable flaws. Initially applied to computer vision systems (Szegedy et al. 2014), the generation of adversarial examples is a process in which seemingly imperceptible changes are made to an image, with the purpose of inducing a deep learning based classifier to misclassify the image. The effectiveness of such attacks is quite high, often resulting in misclassification rates of above 90% in image classifiers (Goodfellow, Shlens, and Szegedy 2015). Due to the exploitative nature of these attacks, it can be difficult to defend against adversarial examples while maintaining general accuracy.

The generation of adversarial examples is not just limited to image recognition. Although speech recognition traditionally relied heavily on signal processing and hidden Markov models, the gradual growth of computer hardware capabilities and available data has enabled end-to-end neural models to become more popular and even state of the art. As such, speech recognizers that rely heavily on deep learning models are susceptible to adversarial attacks. Recent work has been done on the generation of targeted adversarial examples against a convolutional neural network trained on the widely used Speech Commands dataset (Alzantot, Balaji, and Srivastava 2017) and against Mozilla's implementation of the DeepSpeech end-to-end model (Carlini and Wagner 2018), in both cases generating highly potent and effective adversarial examples that were able to achieve up to a 100% misclassification rate. Due to this trend, the reliability of deep learning models for automatic speech recognition is compromised; there is an urgent need for adequate defense against adversarial examples.

Related Work

The attack against Speech Commands described by Alzantot et al. (Alzantot, Balaji, and Srivastava 2017) is particularly relevant within the realm of telephony, as it could be adapted to fool limited-vocabulary speech classifiers used for automated attendants. This attack produces adversarial examples using a gradient-free genetic algorithm, allowing the attack to penetrate the non-differentiable layers of preprocessing typically used in automatic speech recognition.

Methods of defense against adversarial examples can be divided into two categories: mitigation (i.e. retrieving the original label of an adversarial example) and detection (i.e. declaring a given example as adversarial or benign). This section will discuss audio preprocessing mitigation methods and draw attention to a preprocessing-based ensemble detection method used for detecting adversarial examples in the image space.

Audio Preprocessing Defenses

Recent work within computer vision classifiers has shown that some preprocessing, such as JPEG and JPEG2000 image compression (Aydemir, Temizel, and Temizel 2018), cropping and resizing (Graese, Rozsa, and Boult 2016), and pixel deflection (Prakash et al. 2018) have a certain degree of success in defending against adversarial attacks. In a similar vain, preprocessing defenses have also been used for defending against adversarial attacks on speech recognition. Work has shown that using local smoothing, downsampling, and quantization can be somewhat effective in

disrupting adversarial examples produced by the attack of Alzantot et al. (Yang et al. 2018). While quantizing with $q = 256$, Yang et al. were able to achieve their best result of correctly retrieving the original label of 63.8% of the adversarial examples, with a low cost to general model accuracy. As quantization causes the amplitudes of sampled data to be rounded to the closest integer multiple of q, adversarial perturbations with small amplitudes can be disrupted.

Work has also been done in employing audio compression, Hertz shifting, noise reduction, and a low-pass filter (Lemmond and Fitzgibbons 2018) to defend against Carlini and Wagner's attack (Carlini and Wagner 2018) on the DeepSpeech model. The results of Lemmond and Fitzgibbons show that the most promising preprocessing defense tested was the low-pass filter, which achieved a 90.11% success rate in defeating Carlini and Wagner's adversarial examples while maintaining a relatively high general accuracy of 90.91%. This high rate of success may be attributed to the fact that audio samples from human speech are found within relatively lower frequencies, allowing for many of the high-frequency adversarial perturbations to be removed while largely preserving the quality of human speech.

Speech Coding

Although the results of Lemmond and Fitzgibbons seem to suggest that audio compression is outclassed by methods such as low-pass filtering for mitigating adversarial examples, only the Advanced Audio Coding (AAC) and MP3 audio coding standards were discussed. While these compression standards are quite popular and are used in a variety of commercial situations, they are not necessarily optimal for destroying adversarial examples on speech recognition. For teleconferencing and VoIP purposes, speech codecs such as Speex (Valin 2006) and Opus (Valin, Vos, and Terriberry 2012) are more commonly used, as they are able to preserve human speech even through very lossy compression and low bitrates.

In 2002, Valin (Valin 2006) began the Speex project with the goal of providing "a free codec for free speech." Over time, Speex began to grow in popularity, being adopted for many practical VoIP applications, such as TeamSpeak[1] and Twinkle[2]. The codec is based off of the Code Excited Linear Prediction (CELP) algorithm (Schroeder and Atal 1985), which (in a simplified sense) models the vocal tract using a linear prediction model while minimizing the difference with the uncompressed source within a "perceptually weighted domain." In particular, this minimization is accomplished by applying the following weighting filter to the input:

$$W(z) = \frac{A(z/\gamma_1)}{A(z/\gamma_2)} \tag{1}$$

where A is a linear prediction filter with γ_1 and γ_2 controlling the shape of the filter. This filter allows for different levels of noise at various frequencies, and seems to be quite useful for destroying adversarial perturbations while preserving human speech. Speex also includes many additional features, such as voice activity detection, denoising, and support of various bandwidths. As this compression seems to exhibit many similar properties to audio preprocessing methods that have shown to be moderately successful in mitigating perturbations, it seems much more suited to the task of defending against adversarial attacks than MP3 or AAC compression.

In 2012, the Opus codec, which is currently used by the widely-used proprietary VoIP application Discord[3], was released as a successor to the Speex codec (Valin, Vos, and Terriberry 2012). It combines the CELT algorithm with SILK, a linear predictive coding algorithm developed by Skype Technologies in 2009[4]. As this is widely considered an improvement to Speex for speech coding, the performance of Opus compression against adversarial attacks is also worth testing.

Ensemble Detection

Preprocessing defenses against adversarial examples can only be effective and practical if they are able to mitigate adversarial examples without greatly compromising general model accuracy. A viable form of preprocessing would disrupt the predictions of adversarial examples more than it would disrupt the predictions of benign examples. In particular, there should ideally be a small difference between the output vectors produced by passing the raw input and preprocessed input through a neural network when the input is benign, but that same difference should be much larger if the input is adversarial. This core idea can be used to apply preprocessing methods to detect adversarial examples, rather than simply mitigating or neutralizing perturbations.

Within the field of computer vision, ensembles of preprocessing methods have been used for detecting adversarial examples (Xu, Evans, and Qi 2018). Xu et al. proposed the feature squeezing method for detecting adversarial examples. This method combines smaller "squeezing" methods into an ensemble, and calculates an L_1 score from of the maximum L_1 distance between any pair of output probability vectors produced by passing the raw and squeezed inputs through a deep neural network (DNN). Using feature squeezing, Xu et al. were able to consistently detect over 80% of adversarial examples produced from a variety of attacks.

Methods and Evaluation

The aim of this research can be divided into two parts: using Speex and Opus compression as isolated forms of preprocessing for mitigating adversarial examples, and integrating voice compression with an ensemble defense. The adversarial examples are produced using the gradient-free attack of Alzantot et al., against the same pre-trained

[1] https://teamspeak.com/en/features/overview
[2] https://www.linuxlinks.com/Twinkle/
[3] http://discordapp.com/features
[4] http://www.h-online.com/open/news/item/Skype-publishes-SILK-audio-codec-source-code-955264.html

Speech Commands model (Alzantot, Balaji, and Srivastava 2017).

Speech Commands Dataset and Model

The Speech Commands dataset was first released in 2017 and contains 105,829 labeled utterances of 32 words from 2,618 speakers (Warden 2018). The Speech Commands model is a light-weight model based on a keyword spotting convolutional neural network (CNN) (Sainath and Parada 2015) that achieves a 90% classification accuracy on this dataset. For the purposes of this research, a subset of only 30,799 labeled utterances of 10 words are used, for consistency with previous work regarding the adversarial examples of Alzantot, et al. From this subset, 20 adversarial examples are generated for each nontrivial source-target word pair, for a total of 1800 examples. Each example is generated by implementing the attack with a maximum of 500 iterations.

Voice Compression as a Preprocessing Mitigation Defense

The performance of Opus and Speex compression in destroying adversarial examples are compared with the following forms of preprocessing:

- MP3 Compression,
- AAC Compression,
- Band-pass Filtering, and
- Audio Panning and Lengthening.

While the MP3 and AAC compressions correspond directly to preprocessing defenses in related work described earlier, two of the above preprocessing defenses have not yet been directly tested against audio adversarial examples. The band-pass filter combines the low-pass filter of Lemmond and Fitzgibbons with a high-pass filter, with the purpose of eliminating more adversarial perturbations outside of the frequency range for natural human speech. Audio panning is a form of preprocessing typically used in audio mixing that distributes a signal across stereophonic channels, distorting the channel volumes to mimic the perception of audio coming from an off-centered position. The audio panning and lengthening defense lengthens the audio by 1% after panning to increase the spatial distortion of adversarial perturbations in the signal.

Ensemble Mitigation Defense

Despite the apparent success of isolated preprocessing against certain adversarial examples, it has been shown that attacks aware of the preprocessing defense can optimize examples robust to this(Carlini and Wagner 2018). As such, the use of speech coding alone for mitigating adversarial examples would render the model more insecure and vulnerable to smarter attacks. Therefore, an ensemble of preprocessing methods deployed in tandem with speech coding may be able to provide a more secure defense.

The proposed ensemble involves computing three distinct probability vectors produced by passing the following signals through the pre-trained Speech Commands model after speech coding:

- The decoded signal without additional preprocessing,
- The decoded signal passed through a band-pass filter, and
- The decoded signal panned and lengthened by 1%.

These methods of preprocessing were chosen due to their fundamental differences in how they distort the signal; this may allow for more robust mitigation of adversarial perturbations. The three resultant probability vectors are then added together, with the maximum class being returned as the prediction.

Isolated Preprocessing for Detection

A simple method for using preprocessing to detect adversarial examples is by checking to see if the prediction produced by the model changes if the input is preprocessed; if the model's prediction of the raw input does not match the prediction of the preprocessed input, it is declared adversarial.

The following preprocessing methods are used in isolation for detecting adversarial examples:

- MP3 Compression,
- AAC Compression,
- Speex Compression,
- Opus Compression,
- Band-pass Filtering, and
- Audio Panning and Lengthening.

These are the same methods of preprocessing that are tested as isolated preprocessing mitigation defenses, as described earlier.

Ensemble Detection Methods

Similarly to the motivation behind incorporating speech coding into a larger preprocessing ensemble mitigation defense, it may be more secure to add some extra complexity to the detection methods discussed earlier. The aforementioned isolated preprocessing detection methods can be combined into larger and more secure ensemble detection methods. This research explores and compares various configurations for combining the isolated preprocessing detection methods into an ensemble.

Majority Voting Ensemble: The simplest method of combining the preprocessing methods together would be by assigning each preprocessing method a vote, and declaring an audio signal as adversarial if a majority of the ensemble declares the signal adversarial. As there are six preprocessing methods that are combined into an ensemble, ties with this discrete voting scheme are possible. To err on the side of security, this procedure will declare a signal as adversarial in the event of a tie.

Learned Threshold Voting Ensemble: The majority voting ensemble declares an audio signal as adversarial if there are at least three votes in favor of it being adversarial. This threshold for deciding how many votes are needed to declare an audio signal as adversarial is arbitrary, and can adapt to different circumstances. A low threshold would result in a high recall in detecting adversarial examples, but would sacrifice precision. A high threshold would result in

a lower recall in detecting adversarial examples, but would yield a higher precision. This ensemble method experiments with using various voting thresholds for detecting adversarial examples on a labeled training set, and chooses the threshold that results in the best precision and recall. To balance both precision and recall, F_1 scores are used for selecting the best threshold, although in practice, one could adjust the F-measure to reflect one's attitude on the relative importances of precision and recall.

L_1 scoring: The previously discussed ensemble voting methods are relatively simple, as they simply examine the model's discrete prediction of the raw and preprocessed inputs for each preprocessing method. Additionally, the voting methods above are indiscriminate and treat each member of the ensemble equally. A more nuanced approach for measuring the differences in predictions between raw and preprocessed inputs is by L_1 scoring the different output logit vectors, similar to how Xu et al. integrated the multiple squeezing methods in their feature squeezing defense. In this method, an ideal threshold L_1 score is learned from training data by finding the threshold of maximum information gain, and test examples that surpass this threshold are declared adversarial. This method uses the maximum L_1 distance to calculate the score, implicitly assigning more importance to preprocessing methods that produce output vectors that are highly different than the output vectors produced by predicting raw signal. As such, this method would theoretically be more sensitive in detecting adversarial examples, but it may also be quite aggressive in declaring signals as adversarial at the risk of falsely declaring benign examples as adversarial.

Tree-based Classification Algorithms: The above ensemble methods discard information of the class-specific variation in the output vector for each preprocessing method, relative to the raw input. In order to preserve this information, a multidimensional vector can be used, with each dimension accounting for the output vector variation for that class. For the tree-based detection methods discussed in this research, a multidimensional vector composed of the summed absolute class-specific differences between the raw input's resultant probability vector and the preprocessed input's resultant probability vector over each method of preprocessing. In particular, the ith dimension of this summed absolute difference (SAD) vector S is calculated as follows:

$$S_i = \sum_{p \in P} |r_i - p_i| \tag{2}$$

where P corresponds to the set of output probability vectors yielded by the methods of preprocessing in the ensemble, and r corresponds to the output probability vector produced by passing the raw signal through the Speech Commands model without any preprocessing.

This vector will preserve information about class-specific variation between the predictions, and will reduce the number of features of the vector inputted to the tree-based classifier down to 12 (which is the same as the number of classes). Considering the relatively small training dataset size (which is discussed in Section 3.4), having less features for tree-based classification may improve performance. However, the 84dimensional vector formed by simply concatenating each output probability vector together would preserve the most amount of information. As such, the use of this concatenated probability (CP) vector for tree-based classification is also tested, even if the dataset isn't large enough for the classification algorithms to effectively handle that large of a vector.

Decision tree-based classification algorithms are well-suited for classifying vectors of features into discrete classes. In this research, three tree-based classification algorithms are employed for using vectors of summed absolute differences for detecting adversarial examples: random forest classification, adaptive boosting, and extreme gradient boosting. Random forest classification functions by constructing many decision trees in an attempt to stave off the possibility of overfitting. Adaptive boosting and extreme gradient boosting are gradient boosting algorithms which function by building an ensemble of weak learners in a stagewise fashion. Each of these tree-based algorithms are used twice in this research: once for using SAD vectors for classification and once for using CP vectors for classification. These tree-based algorithms have had quite high success in applied problems, are possibly well-suited for detecting adversarial examples.

Evaluation

All of the aforementioned preprocessing mitigation defenses are evaluated by their robustness r against the attack and their effect on the general model accuracy a_g. Within the context of this paper, the measurements are defined as such:

$$\begin{aligned} r &= c_{adv} \ / \ n_{adv} \\ a_g &= c_{ben} \ / \ n_{ben} \end{aligned} \tag{3}$$

where c_{adv} represents the number of adversarial examples correctly labeled by the classifier after preprocessing, n_{adv} represents the total number of adversarial examples, c_{ben} represents the number of benign samples correctly labeled by the classifier after preprocessing, and n_{ben} represents the total number of benign samples.

Lemmond and Fitzgibbons noted a tradeoff between general model accuracy and robustness in their comparison of various preprocessing defenses against Carlini and Wagner's attack on DeepSpeech. An inverse correlation between these two quantities would be troublesome; given the widespread use of automatic speech recognition, there is an urgent need for models to be both accurate for overall usability and secure against potentially malicious attacks. Therefore, in order to honestly compare the performances of preprocessing defenses, a metric that acknowledges both model accuracy and robustness must be used.

The F-measure (F_β) was first proposed in 1979 to balance the need for both precision and recall measurements in evaluating binary classifiers (Chinchor 1992). Taking inspiration from this method of combining two measurements to form a single metric, this research proposes the

R-measure (R_β) for responsibly evaluating preprocessing defenses against adversarial examples:

$$R_\beta = (1+\beta^2)\frac{a_g r}{\beta^2 a_g + r}. \quad (4)$$

where β denotes the relative importance assigned to robustness over general model accuracy. While security against attacks is crucial, it should not come at great expense to model usability. As such, $\beta = 1$ is used for calculating R-measures to reflect this attitude of assigning equal importance to both r and a_g.

The detection methods are evaluated by calculating their precision and recall values in detecting adversarial examples. Although it is important to have a high recall in detecting adversarial examples for the sake of security, a low precision in detection would cause the model to decline in usability. This research takes the stance of both security and general model accuracy being equally important. To reflect this attitude, F_1 scores are used to combine the precision and recall measurements with equal consideration.

Results

Mitigation Methods

The robustness, general accuracy, and R_1 scores are evaluated for each of the discussed defenses and are summarized in Table I. From the results, one can see that all of the methods of preprocessing are able to noticeably mitigate adversarial examples produced by the attack. These results are consistent with the findings of Lemmond and Fitzgibbons that MP3 and AAC compression do not perform as well as filtering against adversarial examples. However, we see that Opus and Speex voice compression perform much better than traditional audio compression. Speex compression, in particular, yields noticeably higher robustness and a better R_1 score than the other forms of individual preprocessing, including the quantization method described by Yang, et al. It is also worth noting that all of the defenses only resulted in a slight decline in general model accuracy, going against the notion of a major tradeoff between general model accuracy and robustness conjectured by Lemmond and Fitzgibbons. A decline in general model accuracy would perhaps be more noticeable if these preprocessing defenses were applied on a continuous speech recognition model.

TABLE I
PERFORMANCE OF PREPROCESSING DEFENSES

Preprocessing Defense	Robustness	General Model Accuracy	R_1 Score
No Defense	7.1%	**90.3%**	0.132
MP3 Compression	53.5%	89.4%	0.669
AAC Compression	59.6%	89.6%	0.716
Quantization-256[a]	63.8%	89.0%	0.743
Band-Pass Filtering	68.7%	89.7%	0.778
Audio Panning[b]	69.1%	89.7%	0.781
Opus Compression	65.0%	90.1%	0.755
Speex Compression	76.8%	89.8%	0.828
Ensemble Defense	**77.4%**	89.7%	**0.831**

[a]Taken from Yang et al.
[b]Includes a lengthening of 1% after the panning.

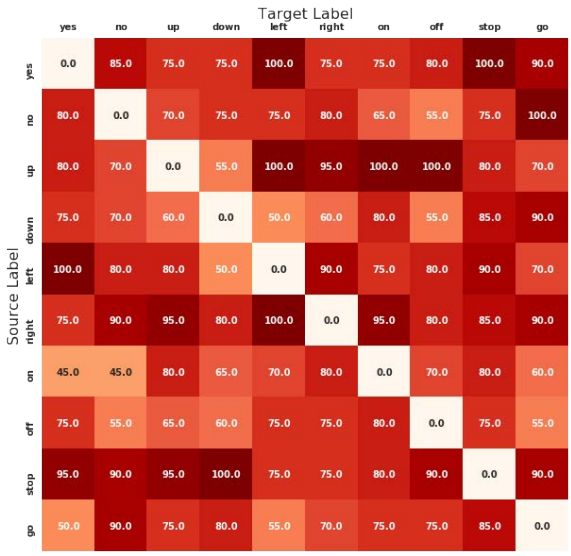


Fig. 1. A heat map depicting the robustness measurements (in percentages) of the ensemble defense to specific targeted attacks. The diagonal of zeroes correspond to trivial source-target pairs for which no adversarial examples were generated.

The ensemble defense achieved better robustness and brought down the targeted attack success rate to a mere 2.9%. The full defense ultimately incorporated just Speex as the speech codec, as islolated Speex compression generally outperformed isolated Opus compression for defending against adversarial examples and achieved a higher R_1 score; additionally, there were no specific targeted examples where Opus compression outperformed Speex compression. As Speex compression was able to preserve human voice quite well, the general model accuracy was not noticeably compromised when using the ensemble defense. The robustness of the full mitigation defense is detailed in Fig. 1.

Detection Methods

The results of the isolated preprocessing detection methods described in summarized in Table II. Measurements indicate that all of the methods are capable of detecting adversarial examples produced by the attack with varying rates of success. These results are also consistent with the findings of Lemmond and Fitzgibbons in that MP3 compression performs adequately at best when compared with the other methods. AAC and Opus compression perform notably better, but are not able to achieve as high of a recall as Speex compression (which also yields the highest F_1 score).

Although the use of bandpass filtering for detecting adversarial examples is extremely precise, it yields a remarkably low recall, which suggests it is a bit too passive with its declaration of adversariality. As many of these preprocessing methods distort audio signals in fundamentally different ways, the overall high precision (and lower recall) measurements of each of the individual preprocessing suggest that some of the ensemble methods may be more effective in detecting adversarial examples.

TABLE II
PERFORMANCE OF ISOLATED PREPROCESSING DETECTION METHODS

Preprocessing Defense	Precision	Recall	F_1 Score
MP3 Compression	93.7%	70.7%	0.806
AAC Compression	95.0%	81.2%	0.876
Band-Pass Filtering	**97.3%**	40.6%	0.573
Audio Panning[a]	95.8%	82.4%	0.886
Opus Compression	94.5%	81.8%	0.877
Speex Compression	93.7%	**88.5%**	**0.910**

[a]Includes a lengthening of 1% after the panning.

TABLE III
PERFORMANCE OF ENSEMBLE DETECTION METHODS

Ensemble Detection Method	Precision	Recall	F_1 Score
Majority Voting	**96.1%**	88.1%	0.919
Learned Threshold Voting	93.5%	91.2%	**0.924**
L_1 Scoring	76.9%	92.4%	0.840
Random Forest Classification (SAD Vector)	79.3%	87.0%	0.830
Random Forest Classification (CP Vector)	86.7%	**94.4%**	0.904
Adaptive Boosting (SAD Vector)	83.5%	81.8%	0.827
Adaptive Boosting (CP Vector)	86.7%	93.0%	0.898
Extreme Gradient Boosting (SAD Vector)	83.0%	84.2%	0.836
Extreme Gradient Boosting (CP Vector)	88.3%	**94.4%**	0.913

[a]Includes a lengthening of 1% after the panning.

The results of the ensemble detection methods are summarized in Table III. The voting methods performed quite well and achieved the two highest F_1 scores of all the methods discussed in this paper. This may be attributed to the high precisions and low recalls of the individual preprocessing methods described in Table II; the relatively strict voting threshold of votes needed for an adversarial declaration capitalizes on the high precision of each of the methods and is able to increase recall. The majority voting method especially benefited from the high precisions of its constituents and yielded an extremely high precision of 96.1%. The Learned Threshold Voting method was able to learn a lower voting threshold of only two votes needed for an adversarial declaration. As such, this method was able to yield a notably higher recall than what was achieved through majority voting, but at a noticeable cost to precision. As the Learned Threshold Voting method still retained a fairly high precision, it achieved the overall highest F_1 score of any of the other preprocessing methods.

The L_1 Scoring method was able to achieve higher recall than either of the two voting methods, perhaps due to its aggressive nature. However, this was achieved at the cost of precision, which evidently lowered the F_1 score.

Although tree-based classification algorithms can be quite powerful in a variety of situations, the tree-based methods were not able to perform as well as the voting methods in detecting adversarial examples using SAD vectors. This may be because the SAD vectors fed into the tree algorithms discarded important voter-specific information. In particular, the vector of summed absolute differences effectively anonymizes the voters in the ensemble; it inherently considers each member of the ensemble equally.

This discarded information proved to be quite crucial for effectively detecting adversarial examples, as the tree-based classification methods performed significantly better with CP vectors (which are highly conservative). In particular, the extreme gradient boosting and adaptive boosting classification algorithms were able to yield the highest recall values for detecting adversarial examples out of all of the detection methods discussed in this research. Considering that the tree-based classification methods performed significantly better with the voter-specific information available in the CP vector, it is worth noting that the Learned Threshold Voting method, which yielded a higher F_1 score than any tree-based classification method, does not use voter-specific information; each vote carries equal weight towards breaking the learned threshold. As such, it may be possible that the tree-based classification methods outperform the Learned Threshold Voting method on larger datasets, as it could be that this training dataset was not sufficiently large enough for learning how to optimally use an 84dimensional vector for classification. However, given the heavy reliance of training data that the tree-based classification methods exhibit, they are likely not as well-suited for flexibly handling different types of attacks as the voting methods.

Future Work

Although the results suggest that a ensemble defenses incorporating speech coding are effective in both mitigating and detecting adversarial examples produced by the unmodified algorithm of Alzantot et al., it does not necessarily show that this defense is secure against more complex attacks. Although an ensemble defense may provide marginal security over isolated preprocessing, recent work has shown adaptive attacks on image classifiers are able to bypass ensembles of weak defenses (He et al. 2017); this work could be applied to attack speech recognition models. Future work can be done to adapt speech coding into a stronger defense that can withstand these types of

adaptive adversarial examples, or at least cause the attacks to become more perceptible.

Additionally, this paper only discusses two speech codecs, both of which are popular among VoIP applications. Speech codecs that are used more commonly in cellular communication applications are generally better equipped to handle noise; as such, the use of those codecs for mitigating adversarial perturbations is a realm for future work.

Furthermore, this paper merely focuses on mitigating and destroying adversarial examples through preprocessing, rather than detecting adversarial examples. As such, another area for future work is incorporating the current defense with common adversarial detection techniques found in image processing.

Future Direction of This Research

Although there are many avenues for continue work, the future direction of this research will likely be towards re-implementing the attack of Alzantot, et al. to be aware of the preprocessing defense, and evaluating that new attack on the defense. While this new attack should be able to penetrate much of the defense, ideally these aware attacks would contain more perceptible perturbations. The next phase of research will focus on analyzing the perceptibility of the perturbations within examples, and see if it is true that the preprocessing-aware attacks produce "noisier" examples to penetrate the defense.

Conclusion

This paper showed that speech coding commonly used in VoIP applications is currently fairly effective in mitigating the single-word targeted adversarial attacks of Alzantot, et al. This paper also proposed a more secure ensemble defense in tandem with speech coding, and compared this defense with isolated preprocessing defenses, using a newly defined metric to balance robustness and general model accuracy. While these defenses would not be extremely secure against more adaptive attacks, this research aimed ultimately to further discussion of defenses against adversarial examples within the audio domain: a field in desperate need of more literature.

Acknowledgements

I would like to thank Terrance Boult for general advice and interesting ideas regarding the research, and the UCCS LINC and VAST labs for providing a supportive community for conducting this work. This work is supported by the National Science Foundation under Grant No. 1659788. Any opinions, findings, and conclusions or recommendations expressed in the material are those of the author(s) and do not necessarily represent the views of the National Science Foundation.

References

Alzantot, M.; Balaji, B.; and Srivastava, M. 2017. Did you hear that? adversarial examples against automatic speech recognition. In *31st Conference on Neural Information Processing Systems (NIPS)*.

Aydemir, A. E.; Temizel, A.; and Temizel, T. T. 2018. The effects of JPEG and JPEG2000 compression on attacks using adversarial examples. *arXiv preprint* (1803.10418).

Carlini, N., and Wagner, D. 2018. Audio adversarial examples: Targeted attacks on speech-to-text. In *1st IEEE Workshop on Deep Learning and Security*.

Chinchor, N. 1992. MUC-4 evaluation metrics. In *4th Conference on Message Understanding*.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Graese, A.; Rozsa, A.; and Boult, T. E. 2016. Assessing threat of adversarial examples on deep neural networks. In *15th IEEE International Conference on Machine Learning and Applications (ICMLA)*.

He, W.; Wei, J.; Chen, X.; Carlini, N.; and Song, D. 2017. Adversarial example defense: Ensembles of weak defenses are not strong. In *11th USENIX Workshop on Offensive Technologies, WOOT 2017*.

Lemmond, D., and Fitzgibbons, R. 2018. Adversarial examples in audio. CS4860 Final Project Report, University of Colorado, Colorado Springs Spring 2018.

Prakash, A.; Moran, N.; Garber, S.; DiLillo, A.; and Storer, J. 2018. Deflecting adversarial attacks with pixel deflection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Sainath, T. N., and Parada, C. 2015. Convolutional neural networks for small-footprint keyword spotting. In *INTERSPEECH*.

Schroeder, M. R., and Atal, B. S. 1985. Code-excited linear prediction (CELP): High-quality speech at very low bit rates. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 937–940.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I. J.; and Fergus, R. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations*.

Valin, J.-M.; Vos, K.; and Terriberry, T. B. 2012. Definition of the Opus audio codec. RFC 6716.

Valin, J.-M. 2006. Speex: A free codec for free speech. In *Proceedings of linux.conf.au*.

Warden, P. 2018. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint* (1804.03209).

Xu, W.; Evans, D.; and Qi, Y. 2018. Feature squeezing: Detecting adversarial examples in deep neural networks. In *2018 Network and Distributed System Security Symposium (NDSS18)*.

Yang, Z.; Li, B.; Chen, P.-Y.; and Song, D. 2018. Towards mitigating audio adversarial perturbations.

Towards a Universal Document Encoder for Authorship Attribution

Kieran Sagar Parikh
Department of Computer Science
Middlebury College
Middlebury, Vermont, USA
kparikh@middlebury.edu

Vinodini Venkataram and **Jugal Kalita**
Department of Computer Science
University of Colorado
Colorado Springs, Colorado, USA
vvenkata@uccs.edu, jkalita@uccs.edu

Abstract

Distributed embeddings of words and sentences have been successful in representing the semantic and syntactic properties of text. In this paper, we examine the creation of distributed document embeddings based on textual style. We present a supervised training architecture to train an encoder to produce these embeddings. We explore the viability of these embeddings for the authorship attribution task, and assess the effects of encoder architecture, text processing, and classifier architecture on authorship attribution performance. While our model does not meet or exceed state of the art results on the same datasets, we are able to confirm our hypothesis that our supervised encoder training method produces an encoder which embeds texts based on that which unites an authors own work and distinguishes it from others, collectively speaking.

Introduction

Writing style represents the distinguishing characteristics of a given author's writing, and characterizing writing style is a central topic in literary and forensic scholarship. Writing style plays a crucial role in authorship analysis tasks.

In this paper, we present the development of a text encoder which learns to produce document vectors reflecting the author-specific (stylistic) qualities of texts. We follow by exploring the viability of such an encoder for authorship attribution. Authorship attribution is the process of inferring characteristics of multiples authors' writing styles from examples of their work, and subsequently using these inferred characteristics to classify unseen texts by author (Juola, 2008).

Distributed representations or so-called embeddings are widely used to represent the semantic properties of words, sentences, and snippets of text (Mikolov et al., 2013; Kiros et al., 2015; Le and Mikolov, 2014). Embeddings have also been shown to be useful in a limited way in capturing the stylistic qualities of an author for use in authorship attribution (Koppel, Schler, and Argamon, 2011; Gomez-Adorno et al., 2018). Most such embeddings have been computed using unsupervised approaches, based on the context in which a word or sentence appears. However, Conneau et al. (2017) have recently taken a supervised approach to teach an encoder to create "universal" sentence embeddings.

Our research focuses on evaluating the performance of an encoder-classifier model for authorship attribution. We take a supervised learning approach to create an encoder to effectively map texts to embeddings based on (the stylistic) characteristics of their writing. We use these embeddings with various classification architectures to perform experiments in authorship attribution. We believe that our research could be extended to create a universal encoder to create (style) embeddings for any texts for later use in classification based on the encoded (stylistic) properties.

An encoder-classifier model has several advantages over an end to end model for author attribution. First, embeddings produced by such an encoder may be useful in a variety of authorship analysis tasks, such as classifying texts based by age or gender, or clustering authors based on stylistic similarities. Second, an encoder that maps texts with different styles to distant vectors in a vector space could also help in open-set author attribution to identify writings of unknown authors. An open-set classifier could use a threshold maximum distance in the vector space to detect writing samples of unknown authors. Third, by breaking up the model into two distinct pieces (encoder and classifier) end-user flexibility is increased, enabling, for instance, encoding to happen on a mobile device and classification to happen at a later point, perhaps on a centralized server. Furthermore, such an architecture has the added benefit that different classification algorithms can be used on the same embedding without having to re-encode the relevant text.

Finally, in the universal case, the encoder needs to be trained only once to develop a general ability to encode texts, but then could be used to solve different authorship attribution problems by training only a classifier on the relevant dataset after it has been encoded. This could significantly decrease the time and resources required to solve an authorship attribution problem.

Related Work

Koppel, Schler, and Argamon (2011) represented each text as a vector based on the frequencies of 250,000 unique, but overlapping, space free character 4-grams. A new text is assigned to the author whose texts are closest to it in terms of cosine similarity in the vector space. The authors addressed the open-set version of the problem, where an anonymous text that is not similar enough to any known texts is not assigned to any candidate author.

More recently, the use of word embeddings computed using Word2Vec (Mikolov et al., 2013) or GloVe (Pennington, Socher, and Manning, 2014) in an unsupervised manner have become commonplace. Word2Vec was further extended with the introduction of paragraph vectors, which are learned embeddings of variable-length texts (Le and Mikolov, 2014). Paragraph vectors are learned at the same time as the word embeddings to encode the meanings of all words in the paragraph, thereby preserving an understanding of the ordering of words within a text.

Kiros et al. (2015) presented an unsupervised approach to train an encoder to produce generic distributed sentence encodings. Adapting the skip-gram model of Mikolov et al. (2013) to sentences, they used a new objective function to encode a sentence based on others around it. The chose to use gated recurrent units (GRUs) to both encode a sentence and decode the previous and next sentence from such an embedding.

Conneau et al. (2017) created an encoder to produce universal sentence embeddings using a supervised approach. They used a dataset with 570 thousand sentence pairs labelled as having one of three semantic relationships: entailment, contradiction or neutral. The training model created separate embeddings for the two sentences in a pair. Three matching methods were applied to these two embeddings and the result was then fed into a 3-class classifier. The quality of the learned embeddings were evaluated by their performance in transfer tasks, such as sentiment classification, semantic relatedness, paraphrase detection and semantic textual similarity. They found that a BiLSTM network with max pooling produced the best embeddings, outperforming SkipThought vectors and requiring less training time.

Sari, Vlachos, and Stevenson (2017) adapted the FASText method to perform text classification (Joulin et al., 2017) for authorship attribution. They learned custom word and character n-gram embeddings at classifier training time. The embedding for a text was found by averaging the embeddings of all its n-grams, and it was fed to a linear classifier. They achieved state-of-the-art classification accuracy on benchmark datasets.

Gomez-Adorno et al. (2018) use paragraph vectors for authorship attribution. They learn paragraph embeddings considering character, word, and POS n-grams, inspired by the knowledge that character n-grams have had considerable success in authorship attribution. The embeddings are used with a logistic regression classifier to perform cross-topic authorship attribution. They find that the best model is embeddings created from the concatenation of vectors trained on POS 1,2,3-grams and Word 1,2,3-grams.

Architecture and Experimental Protocol

There are two main steps in the way we go about authorship attribution. First, we train a Siamese twin network by providing it with pairs of documents as input. Once the encoder has been trained using this architecture, we encode all our documents using the learned encoder weights. In the second step, we use a decoupled machine learning technique to classify the the documents based on their embeddings.

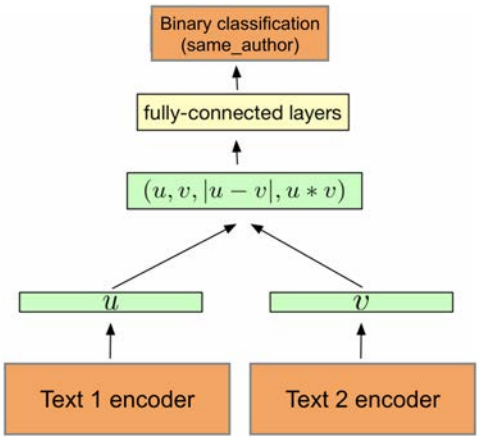


Figure 1: Siamese encoder training architecture. Adapted from Conneau et al. (2017).

Siamese Twin Encoder

We adapt the supervised learning approach for sentence embeddings used by Conneau et al. (2017) to develop an encoder to create embeddings based on authorship style. We modify their training model to take in two variable length snippets of text with a binary label indicating whether the two snippets were written by the same author.

The Siamese encoder training architecture we use is conceptually illustrated in Figure 1. Siamese networks, where two identical sub-networks are used to train on a pair of inputs at the same time have been used successfully for signature verification (Baldi and Chauvin, 1993; Bromley et al., 1994), face verification (Hu, Lu, and Tan, 2014) and other image recognition tasks (Koch, Zemel, and Salakhutdinov, 2015) that require learning suitable distance metrics, adapted to a dataset or domain. However, in practice, we have only one encoder. During training, the encoder is first used to obtain an embedding u of the first text of the pair. The same encoder is next used to obtain the embedding v of the second text of the pair. Once both embeddings are in place, three methods are applied to them: concatenation, element-wise product, and absolute element-wise difference, to produce a vector that combines the embedded information for the two documents. This resulting vector is passed to a dense neural network, consisting of two 64-node hidden layers and a softmax layer, which performs binary classification. Backpropagation on the weights of the encoder is performed based on the errors in pairwise classification, i.e., if the pair was judged correctly to be authored by the same or different authors, as in the input.

We hypothesize that this approach will force the encoder to learn to make embeddings based on that which unites an author's own work and distinguishes it from others', collectively speaking, ultimately resulting in embeddings of documents that are grouped by author within the vector space.

Generating Training Pairs for Encoder

The datasets used in this research contain individual texts labelled by author. It would be computationally expensive

to attempt to train on the complete set of all possible pairs given its large size[1], and would also likely result in significant overfitting as the number of training pairs would be much larger than the number of unique training texts. Therefore, pairs of texts are created at the start of each training epoch. In the first training epoch, 100 differing author pairs and 100 same author pairs are created for each author in the dataset. Differing author pairs are created by holding constant, the author of the first text and randomly choosing a second author for each pair. Same author pairs are created by holding constant the author of both pieces of text in a pair. For every pair, texts are chosen randomly from among the relevant author's works. In subsequent epochs, pairs are created both randomly and adaptively based on the author pairs on which the model performed poorly in the previous epoch. Ten differing author pairs and fifty same author pairs are created for each author in the dataset. A misclassification count from the previous epoch organized by author pair is used to create ten training pairs for each misclassified pair, with a maximum of hundred training pairs for any unique author pair. Pairs created based on previous misclassifications are created using the same authors as the misclassified pair, with texts chosen randomly from among the relevant authors' works. Pair generation is done in this way to avoid overfitting to specific pairs in the set of all possible pairs, and to adaptively focus training on difficult author pairs.

Encoder Models Used

We investigated variation of both the fundamental architecture of the encoder and the way in which an input text is prepared for the encoder. We investigated four encoder architectures: a bidirectional LSTM (BiLSTM) with max pooling, a biLSTM with mean pooling, a Hierarchical Convolution Neural Network (ConvNet) with max pooling, and a ConvNet with mean pooling.

In terms of preparation, three main encoder models were used. The first model used was a word based model which fed texts encoded using word embeddings into the LSTM to create 256 dimensional embeddings. The second was a character based 3-gram model which encoded texts using character 3-gram embeddings and used the LSTM to create 256 dimensional embeddings. The third model was a combination of the first two, using two encoders in parallel, one operating on words and the other on character 3-grams, to produce 512 dimensional embeddings.

Furthermore, Sari, Vlachos, and Stevenson (2017) found word and character n-gram embeddings learned directly on the dataset to be effective in authorship attribution, so our models were tested using both pretrained GloVe word embeddings (Pennington, Socher, and Manning, 2014) and pretrained character 3-gram embeddings from Hashimoto et al. (2016) as well as custom word and character embeddings. Custom word and character 3-gram embeddings were produced using the skip-gram model (Mikolov et al., 2013) on both the train set exclusively and the combined train and test set of the CCAT-50 dataset.

[1]For instance, over 3 million unique pairs could be created from the 2500 training set examples in the CCAT-50 dataset.

Metric Learning

Since our model aims to produce document embeddings such that documents are grouped by author within the vector space, the effect of using metric learning techniques to group classes together and separate them from other classes was investigated. We used Large Margin Nearest Neighbor (LMNN) (Weinberger and Saul, 2009) metric learning, which is a metric learning algorithm designed specifically to improve *k*NN performance. LMNN learns a Mahalanobis distance metric in a supervised way where, for a given data point, it is rewarded if the point's k nearest neighbors (measured using the learned metric) are of the same class as the data point, and is penalized if any of its k neighbors are of a different class.

Classification Methods

As mentioned earlier, the Siamese twin network has a single encoder in practice, although it is illustrated as if there are two parallel encoders. The purpose of training the Siamese network is to train this encoder to produce document embeddings that can discriminate among authors. Thus, after training, the encoder is used to produce embeddings of all texts in both the training and test sets. These embeddings are then fed to various decoupled classifiers for the authorship attribution task. All classifiers are trained on the embeddings of the training set and tested on the embeddings of the test set. The first classification algorithm uses SVMs with RBF kernels in a "one-against-one" (Knerr, Personnaz, and Dreyfus, 1990) approach to perform multi-class classification. The second algorithm is *k*-nearest neighbors (*k*NN) with a k value of 5. The SVM and *k*NN classifications are performed using Scikit-learn (Pedregosa et al., 2011). The third classifier is a cohort algorithm which takes advantage of the binary classifier learned during encoder training. Each unknown text is compared to 30 training set texts from each author using the binary classifier, which predicts whether or not two texts were written by the same author. When the binary classifier predicts that two texts were written by the same author, it is counted as a vote for author of the text to which the unknown text is being compared. Votes are tallied for each author and the text is attributed to the author with the most votes.

Various methods of ensemblizing classifiers were also investigated. Four basic ensemble architectures were tested.

Discrete Plurality Voting The first is a discrete plurality voting ensemble. Four classifiers are used in the ensemble: *k*NN, SVMs with RBF kernels in a "one-against-one" (Knerr, Personnaz, and Dreyfus, 1990) approach, the cohort algorithm described earlier, and a dense neural network with two 256 node layers, one 64 node layer, and a softmax layer, trained for 500 epochs. Each classifier votes for one class and all votes are weighted equally. Ties are broken by choosing *k*NN.

Meta-Classifier The second ensemble uses an meta-classifier, which seeks to predict which classifier(s) (*k*NN, SVM, etc.) will correctly predict the author of an example given its embedding. The same four classifiers from the discrete plurality voting ensemble are used. Predictions for the

Table 1: Multiclass classification accuracy (%) by encoder architecture, classifier type, and dataset.

	CCAT-50			CCAT-10			IMDB62		
Model	SVM	*k*NN	Cohort	SVM	*k*NN	Cohort	SVM	*k*NN	Cohort
Word	62.9	**63.8**	61.36	75.4	**76.6**	68.2	**89.9**	88.1	78.1
Char 3-grams	48.2	49.3	40.4	66.8	67.2	66.0	47.7	35.1	28.7
Combined	59.6	61.8	55.7	72.6	73.0	67.0	88.5	84.7	71.6

training set from each classifier are used to label each embedding in the training set with the classifier(s) that correctly predicted it. The meta-classifier is trained using the training set embeddings and these labels, and is subsequently used on the test set embeddings to predict which classifiers will correctly predict this example. For each Various different meta-classifiers were tested, including SVMs, decision trees, naive Bayes, logistic regression, and XGboost (Chen and Guestrin, 2016).

XGBoost Logits The third ensemble architecture uses XGboost (Chen and Guestrin, 2016) to predict an example's class given the raw logits from each classifier. The same four classifiers from the discrete plurality voting ensemble and meta-classifier ensemble are used. XGBoost is trained using the raw logits and class label for each example in the training set. It is used with the raw logits from the test set examples to predict their classes.

Soft Voting The fourth ensemble architecture uses soft voting, which predicts an example's class using the max of the sums of the probabilities for each class from each classifier. Four classifiers are used in the ensemble: *k*NN, SVMs with RBF kernels in a "one-versus-rest" approach, Random Forest, and Logistic Regression.

Experiments, Results, & Analysis

This section describes the datasets used and the experiments performed along with the results obtained.

Datasets Used

Three datasets were used to evaluate the viability of this approach for authorship attribution.

CCAT-50 (Houvardas and Stamatatos, 2006) is a set of 5000 news stories, comprised of 50 training texts and 50 test texts for each of 50 authors. All texts are corporate news stories to reduce the effect of genre on classification. This dataset is a subset of the Reuters Corpus Volume 1.

CCAT-10 (Stamatatos, 2008) is a subset of the CCAT-50 dataset with fewer authors. It contains 50 training texts and 50 test texts for 10 authors. Like CCAT-50, all texts are corporate news stories to reduce the effect of genre on classification.

IMDb62 (Seroussi, Zukerman, and Bohnert, 2011) is a set of 62,000 movie reviews from the Internet Movie Database (IMDb). It contains 1,000 texts from each of 62 authors. This dataset was split into a train and test set by setting aside 10% of each author's texts for the test set and using the remaining 90% as the training set.

Table 2: Multiclass classification accuracy (%) for word-based model using *k*NN classifier

	CCAT-50	CCAT-10
kNN	**63.8**	**76.6**
kNN w/ LMNN	62.8	75.6

Preliminary Results

To obtain preliminary results, we trained biLSTM with max pooling encoders on word, character 3-gram, and combined models for all three datasets. We used the trained encoders to produce embeddings for each dataset which were subsequently fed to SVM, *k*NN, and Cohort classifiers. Results are presented in Table 1. Our word-based model achieved higher classification accuracy than the other two models on all datasets.

We also observed that *k*NN and SVM classifiers had less than 2% difference in classification accuracy on all three datasets using the word model, and on CCAT-50 and CCAT-10 using the character 3-gram and combined models. Nevertheless, all models and classifiers fail to meet state of the art results on these datasets. A comparison of this research against the state of the art is provided in Table 3.

As a result, further experimentation was undertaken to both attempt to improve classification accuracy to state of the art levels and assess the advantages of this approach. In the subsections that follow, the effects of metric learning, variation of encoder architecture, the use of custom word and n-gram embeddings, and the use of ensemble-ized classifiers are presented. Finally, a visual representation of the documents in the CCAT-10 dataset is presented and analyzed.

Metric Learning

Given the success of *k*NN classifiers, the effect of applying the Large Margin Nearest Neighbor (LMNN) (Weinberger and Saul, 2009) metric learning algorithm to the embeddings was investigated with a k value of 5. Results from these experiments are presented in Table 2. It was found that the use of LMNN caused a small decrease in classification accuracy on both the CCAT-50 and and CCAT-10 datasets.

Custom Embeddings

Custom word and character 3-gram embeddings were produced using the skip-gram model (Mikolov et al., 2013) on both the train set and the combined train and test set of the CCAT-50 dataset. These embeddings were then used to train biLSTM encoders and perform classification. Results from these experiments are presented in Table 4.

Table 3: Comparison against other reported results.

Model	CCAT-50	CCAT-10	IMDb62
Continuous n-gram words (Sari, Vlachos, and Stevenson, 2017)	70.16	77.80	87.87
Continuous n-gram char (Sari, Vlachos, and Stevenson, 2017)	72.60	74.80	**94.80**
SVM with affix & punctuation 3-grams (Sapkota et al., 2015)	69.30	78.80	-
D2V words (Posadas-Duran et al., 2017)	71.84	80.80	-
D2V words + 2 + 3-grams (Posadas-Duran et al., 2017)	**75.24**	82.80	-
D2V words + 2 + 3 + 4 + 5-grams (Posadas-Duran et al., 2017)	74.84	84.60	-
SVM with bag of local histograms (Escalante, Solorio, and Montes-y Gmez, 2011)	-	**86.40**	-
Token SVM (Seroussi, Zukerman, and Bohnert, 2013)	-	-	92.52
Words - Discrete Plurality Voting Ensemble	64.80	76.20	89.56
Words - Soft Logit Voting Ensemble	65.64	77.00	91.69

Table 4: Multiclass classification accuracy (%) by embedding type.

	CCAT-50		
Model	SVM	*k*NN	Cohort
Words (pre-trained)	62.9	**63.8**	61.4
Words (train)	47.8	49.6	39.5
Words (train+test)	52.6	52.1	44.7
Char 3-grams (pre-trained)	48.2	49.3	40.4
Char 3-grams (train)	49.5	50.6	51.8
Char 3-grams (train+test)	50.3	50.4	52.6

Table 5: Multiclass classification accuracy (%) by encoder type.

	CCAT-50		
Encoder Architecture	SVM	*k*NN	Cohort
biLSTM w/ max pooling	62.9	**63.8**	61.4
biLSTM w/ mean pooling	58.4	61.8	61.6
ConvNet w/ max pooling	52.2	52.0	50.6
ConvNet w/ mean pooling	35.3	56.8	53.8

Interestingly, it was found that custom embeddings increased the accuracy of the character based model but decreased the accuracy of the word based model. This indicates that the semantic nature of pretrained GloVe embeddings may be important in the production of meaningful text embeddings, since the custom word embeddings probably captured less semantic value than the GloVe embeddings given the muhc larger training set used to create the GloVe embeddings. This is further evidenced by the fact that the char n-grams, which do not have any semantic value, performed better with custom embeddings. If this is the case, this would also explain the word model's better performance on IMDb62 compared with CCAT-10 and CCAT-50, as previous users of these datasets have noted the importance of text topic for discrimination of authors in IMDb62 (Sari, Vlachos, and Stevenson, 2017; Seroussi, Zukerman, and Bohnert, 2013), while it is less influential in CCAT-10 and CCAT-50 as these datasets have already been controlled for topic.

Unsurprisingly, accuracy with both words and characters was slightly higher using custom embeddings trained on the combined train and test set when compared with custom embeddings trained only on the train set. This gain in accuracy was larger for the word model, likely because there are more words in the test set that are not in the train set than there are character 3-grams that are only in the test set.

Encoder Type

We varied the type of encoder that was used to observe its effect on classification accuracy. Results from these experiment are presented in Table 5. We found that no new encoder architecture could outperform our previous best classification accuracy of 63.8%, obtained using a biLSTM with max pooling and a *k*NN classifier.

Ensemble Classifier

Analysis of the test examples misclassified by the various classifiers revealed that the set of correctly classified examples from each classifer do not completely overlap. For the standard word-based biLSTM with max pooling with the CCAT-50 dataset, it was found that 582 examples—or 23.3% of the dataset—were misclassified by all classifiers. This shows that ensemble-izing these classifiers could result in a higher classification accuracy than the current best of 63.8% using only kNN.

The results of these experiments are presented in Table 6,

Table 6: Multiclass classification accuracy (%) for word-based model using ensemble-ized classifiers

Ensemble Architecture		CCAT-50
Discrete Plurality Voting		64.8
Meta-Classifier	SVM	63.9
Meta-Classifier	Decision Tree	64.7
Meta-Classifier	Naive Bayes	64.1
Meta-Classifier	Logistic Regression	64.6
Meta-Classifier	XGBoost	64.4
XGBoost Logits	max depth = 6	60.8
XGBoost Logits	max depth = 20	61.84
XGBoost Logits	max depth = 50	61.28
XGBoost Logits	max depth = 100	61.2
XGBoost Logits	max depth = 256	61.04
Soft Logit Voting Ensemble		**65.6**

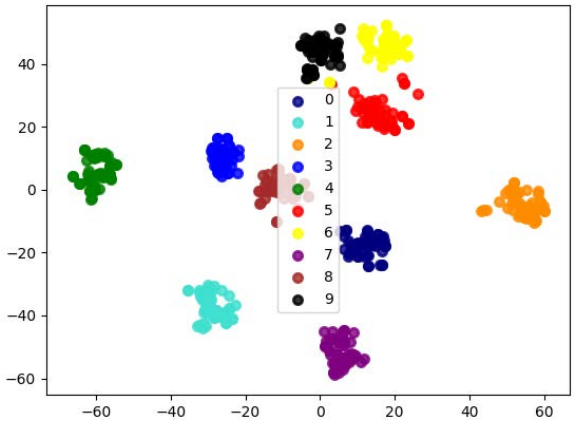


Figure 2: CCAT-10 Train Set

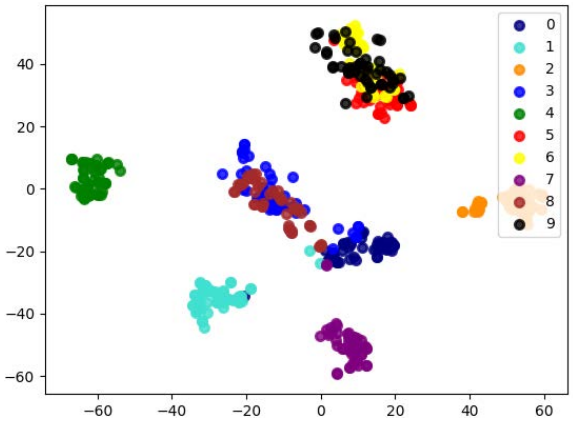


Figure 3: CCAT-10 Test Set

and a comparison against state of the art results is presented in Table 3. It was ultimately found that the soft voting ensemble achieved the highest classification accuracy, but still failed to meet state of the art results on all three datasets.

Visual Analysis of CCAT-10 Documents

The document embeddings produced by our word-based biLSTM with max pooling encoder for the CCAT-10 dataset were used to produce visual representations of the distribution of the encoded texts in the vector space. The vectors were reduced from 256 to 2 dimensions using tSNE (Maaten and Hinton, 2008). The training set is represented in Figure 2, while the test set is represented in Figure 3.

Comparative analysis of the two figures can help to explain some of our results. Examples from the train set appear to be clustered by class, with visible margins between class clusters, few outliers, and little class overlap. Test set examples, however, appear less tightly clustered and have significant class overlap for some classes. While each author class has its own unique cluster in the train set, only four authors (1, 2, 4, and 7) have their own unique cluster in the test set. Two large clusters contain examples from the remaining classes: one with authors 5, 6, and 9, and another with authors 0, 3, and 8.

The tighter clustering in the train set when compared with the test set explains the performance of *k*NN on this dataset. During *k*NN, while the train examples to which a test example is compared may be clustered closely together by class, the test examples tend to be less tightly clustered and have more variance in position. Some may, therefore, be closer to a different class's cluster in the train set, resulting in test example misclassification.

This also explains the poor performance of metric learning techniques: examples in the training set were already clustered by class with margins between clusters, so it is unlikely that any metric could be learned from this train data that would bring closer the test examples which are far from their classes' clusters in the train set.

Despite the tighter clustering of train set examples when compared with test set examples, the encoder still manages to embed unseen examples to the same general area as the train examples of the same class, even if it does not manage to embed them precisely enough to achieve better authorship attribution performance. For instance, the large overlapping cluster in the test set containing authors 5, 6, and 9 occupies roughly the same space as the three individual clusters for authors 5, 6, and 9, in the train set. A similar phenomenon can also be observed with the other overlapping cluster in the test set. This indicates that the encoder is learning to embed documents based on the stylistic characteristics that distinguish the authors in a way that does generalize to unseen texts, if only to a limited extent. This confirms our hypothesis that this training model would force the encoder to learn to make embeddings based on that which unites an author's own work and distinguishes it from others', collectively speaking.

Nevertheless, this analysis reveals that the encoder could generalize better to unseen texts. To improve performance of this model for authorship attribution, the encoder needs to be modified so that the distribution of train text embeddings in the vector space is more similar to the distribution of test texts in the vector space. This could be achieved by producing embeddings of unseen texts that are more tightly clustered together by author or by producing embeddings of train texts that are less tightly clustered. Furthermore, given the tight clusters, it is possible the encoder may be overfitting to the training texts, enabling it to cluster these texts together more tightly at the expense of the quality of the embeddings of unseen texts, thereby explaining the general difference in the distribution of train and test embeddings.

Conclusion

In this research, we presented the development of a text encoder which learns to produce document vectors reflecting the author-specific (stylistic) qualities of texts. We subsequently assessed the viability of such an encoder for authorship attribution and found that the use of a bidirectional LSTM encoder with a soft voting ensemble classifier achieves a classification accuracy that surpasses all our other encoder-classifier approaches, but still fails to meet state of the art results on the same datasets.

While we failed to achieve state of the art performance,

analysis of the embeddings enabled us to confirm our hypothesis that our architecture for encoder training would produce an encoder which embeds texts based on that which unites an authors own work and distinguishes it from others, collectively speaking.

As a result, in future work, we would like to continue to improve this encoder so that it may achieve state of the art authorship attribution results. First, we would like to decrease the effect of encoder overfitting to the train set texts. This will be accomplished by modifying the training pair generation to produce significantly fewer training pairs per epoch. Furthermore, we would like to test our model on a dataset that has many training examples per author, as the CCAT-50 and CCAT-10 have only 50 examples per author, which also could have contributed to overfitting.

Second, we would like to implement and test an encoder architecture based on Transformer (Vaswani et al., 2017), to see if attention based encoder models can achieve competitive performance when dealing with textual style.

Finally, working towards our goal of creating a universal encoder, we would like to investigate the effect of training the encoder on a very large and diverse dataset, of which any potential authorship attribution dataset would be only a small subset.

Acknowledgements

We are thankful to the National Science Foundation for support through grant 1659788.

References

Baldi, P., and Chauvin, Y. 1993. Neural networks for fingerprint recognition. *Neural Computation* 5(3):402–418.

Bromley, J.; Guyon, I.; LeCun, Y.; Säckinger, E.; and Shah, R. 1994. Signature verification using a "siamese" time delay neural network. In *Advances in neural information processing systems*, 737–744.

Chen, T., and Guestrin, C. 2016. Xgboost: A scalable tree boosting system. *CoRR* abs/1603.02754.

Conneau, A.; Kiela, D.; Schwenk, H.; Barrault, L.; and Bordes, A. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 670–680. Copenhagen, Denmark: Association for Computational Linguistics. arXiv: 1705.02364.

Escalante, H. J.; Solorio, T.; and Montes-y Gmez, M. 2011. Local Histograms of Character N-grams for Authorship Attribution. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, 288–298. Stroudsburg, PA, USA: Association for Computational Linguistics.

Gomez-Adorno, H.; Posadas-Duran, J.-P.; Sidorov, G.; and Pinto, D. 2018. Document embeddings learned on various types of n-grams for cross-topic authorship attribution. *Computing*.

Hashimoto, K.; Xiong, C.; Tsuruoka, Y.; and Socher, R. 2016. A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks. *arXiv:1611.01587 [cs]*. arXiv: 1611.01587.

Houvardas, J., and Stamatatos, E. 2006. N-Gram Feature Selection for Authorship Identification. In *Artificial Intelligence: Methodology, Systems, and Applications*, Lecture Notes in Computer Science, 77–86. Springer, Berlin, Heidelberg.

Hu, J.; Lu, J.; and Tan, Y.-P. 2014. Discriminative deep metric learning for face verification in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1875–1882.

Joulin, A.; Grave, E.; Bojanowski, P.; and Mikolov, T. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 427–431. Valencia, Spain: Association for Computational Linguistics.

Juola, P. 2008. *Authorship Attribution*, volume 1 of *Foundations and Trends in Information Retrieval*. Now Publishing.

Kiros, R.; Zhu, Y.; Salakhutdinov, R. R.; Zemel, R.; Urtasun, R.; Torralba, A.; and Fidler, S. 2015. Skip-Thought Vectors. In Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc. 3294–3302.

Knerr, S.; Personnaz, L.; and Dreyfus, G. 1990. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In *Neurocomputing*, NATO ASI Series. Springer, Berlin, Heidelberg. 41–50.

Koch, G.; Zemel, R.; and Salakhutdinov, R. 2015. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2.

Koppel, M.; Schler, J.; and Argamon, S. 2011. Authorship attribution in the wild. *Language Resources and Evaluation* 45(1):83–94.

Le, Q., and Mikolov, T. 2014. Distributed Representations of Sentences and Documents. *Proceedings of the 31st International Conference on Machine Learning* 9.

Maaten, L. v. d., and Hinton, G. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9(Nov):2579–2605.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed Representations of Words and Phrases and their Compositionality. *Proceedings of the 26th International Conference on Neural Information Processing Systems* 2:3111–3119.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, . 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12(Oct):2825–2830.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global Vectors for Word Representation. 1532–1543. Association for Computational Linguistics.

Posadas-Duran, J.-P.; Gmez-Adorno, H.; Sidorov, G.; Batyrshin, I.; Pinto, D.; and Chanona-Hernndez, L. 2017. Application of the distributed document representation in the authorship attribution task for small corpora. *Soft Computing* 21(3):627–639.

Sapkota, U.; Bethard, S.; Montes, M.; and Solorio, T. 2015. Not All Character N-grams Are Created Equal: A Study in Authorship Attribution. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 93–102. Denver, Colorado: Association for Computational Linguistics.

Sari, Y.; Vlachos, A.; and Stevenson, M. 2017. Continuous N-gram Representations for Authorship Attribution. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 267–273. Valencia, Spain: Association for Computational Linguistics.

Seroussi, Y.; Zukerman, I.; and Bohnert, F. 2011. Authorship Attribution with Latent Dirichlet Allocation. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, CoNLL '11, 181–189. Stroudsburg, PA, USA: Association for Computational Linguistics.

Seroussi, Y.; Zukerman, I.; and Bohnert, F. 2013. Authorship Attribution with Topic Models. *Computational Linguistics* 40(2):269–310.

Stamatatos, E. 2008. Author identification: Using text sampling to handle the class imbalance problem. *Information Processing & Management* 44(2):790–799.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, .; and Polosukhin, I. 2017. Attention is All you Need. In *Proceedings of the 31st Conference on Neural Information Processing Systems*, 11.

Weinberger, K. Q., and Saul, L. K. 2009. Distance Metric Learning for Large Margin Nearest Neighbor Classication. 38.

Parallel Attention Mechanisms in Neural Machine Translation

Julian Medina and **Jugal Kalita**
University of Colorado
Colorado Springs, Colorado, USA
jmedina5@uccs.edu

Abstract

Recent papers in neural machine translation have proposed the strict use of attention mechanisms over previous standards such as recurrent and convolutional neural networks (RNNs and CNNs). We propose that by running traditionally stacked encoding branches from encoder-decoder attention-focused architectures in parallel, that even more sequential operations can be removed from the model and thereby decrease training time. In particular, we modify the recently published attention-based architecture called Transformer by Google, by replacing sequential attention modules with parallel ones, reducing the amount of training time and substantially improving BLEU scores at the same time. Experiments over the English to German and English to French translation tasks show that our model establishes a new state of the art.

Introduction

Historically, statistical machine translation involved extensive work in the alignment of words and phrases developed by linguistic experts working with computer scientists (Jurafsky 2000). Deep Learning surpasses these historically used methods and has primarily replaced these with the recent use of neural machine translation (NMT). The predominant design of the state of the art is the encoder-decoder model. The encoder takes sequential text, turning it into an internal representation. The decoder then takes this internal representation and generates a subsequent output. Since their emergence, attention mechanisms (Bahdanau, Cho, and Bengio 2014) have been at the forefront of machine translation.

Attention mechanisms help the neural system focus on parts of the input, and possibly the output as it learns to translate. This concentration facilitates the capturing of dependencies between parts of the input and the output. After training the network, the attention mechanism enables the system to perform translations that can handle issues such as the movement of words and phrases, and fertility. However, even with these attention mechanisms, traditional NMT models have their drawbacks, which include long training time and high computational requirements.

Recent papers (Vaswani et al. 2017; Ahmed, Keskar, and Socher 2017) in neural machine translation have proposed the strict use of attention mechanisms in such networks as the Transformer over previous approaches such as recurrent neural networks (RNNs) (Elman 1990) and convolutional neural networks (CNNs) (LeCun et al. 1998). In other words, these approaches dispense with recurrences and convolutions entirely. In practice, attention mechanisms have mostly been used with recurrent architectures, and removing the recurrent nature of the architecture makes the training more efficient by the removal of necessary sequential steps.

This paper contributes by continuing to pursue the removal of sequential operations within encoder-decoder models. These operations are removed through the parallelization of previously stacked encoder layers. This new parallelized model can obtain a new state of the art in machine translation after being trained on one NVIDIA GTX 1070 for as little as three hours.

The paper includes the following: a discussion of related work in the field of machine translation including encoder-decoder models and attention mechanisms; an explanation of the proposed novel architecture and its motivations; and an examination of used methodology and evaluation including data sets, hardware, hyper-parameters, and metrics. This paper concludes with results and possible avenues for future research.

Related Work

There has been a plethora of work in the past several years on end-to-end neural translation. ByteNet (Kalchbrenner et al. 2016) uses CNNs with dilated convolutions for both encoding and decoding. Zhou et al. (2016) use stacked interleaved bi-directional LSTM layers (up to 16 layers) with skipped connections; ensembling gives the best results. Google's earlier and path-breaking end-to-end translation approach (Wu et al. 2016) uses 16 LSTM layers with attention; once again, ensembling produces the best results. Facebook's end-to-end translation approach (Gehring et al. 2017) depends entirely on CNNs with attention mechanism.

Our work reported in this paper is based on another translation work by Google. Google's Vaswani et al. (2017) proposed the reduction of sequential steps seen in CNNs and RNNs. The sole use of attention mechanisms and feed-forward networks within the common encoder-decoder sequential model replaces the necessity of deep convolutions for distant dependent relationships, and the memory and computation intensive operations required within recurrent networks. Original training and testing were over both the

WMT 2014 English-French (EN-FE) and English-German (EN-DE) data sets, while this paper only uses the WMT 2014 EN-DE set and the IWSLT 2014 EN-DE and EN-FR data sets. This model will continue to be discussed later in the paper.

Works in the field of NMT recommend a particular focus on the encoder. Analysis by Domhan (2018) poses two questions: what type of attention is needed, and where. In this analysis, self-attention had a higher correspondence with accuracy when placed in the encoder section of the architecture than the decoder, even claiming that the decoder, when replaced with a CNN or RNN, retained the same accuracy with little to no loss in robustness. Imamura, Fujita, and Sumita's (2018) study shows that the current paradigm of using high-volume sets of parallel corpora are sufficient for decoders but are unreliable for the encoder. These conclusions encourage further research in the manipulation of position and design of the encoder and these attention mechanisms within them.

Architecture

The Transformer architectures proposed by Vaswani et al. (2017), seen in Figure 1, inspires this paper's work. We have made modifications to this architecture, to make it more efficient. However, our modifications can be applied to any encoder-decoder based model and is architecture-agnostic. These alterations follow from the proceeding two hypotheses.

1. Reduction in the number of required sequential operations throughout the encoder section is likely to reduce training time without reducing performance.
2. Replacing the subsequent encoder attention stack is expected to result in discarding of inter-dependencies, and possibly incorrect, assumptions of encoder attention mechanisms and layers, improving performance.

For simplification, but without loss of generalization, this paper discusses the use and modification of such Transformer based-models. The original Transformer model is composed of stacked self-attention layers. These self-attention mechanisms compare and relate multiple positions of one sequence in order to find a representation of itself. In Figure 1, we see such attention layers, one working on the input embedding, another on the output embedding, and the third on the both the input and the output embeddings. Each of these layers contains two main sub-layers including multi-head self attention, which feeds a simple feed-forward network, and a final layer of normalization. Around each of the main sub-layers, a skip or residual connection (He et al. 2016) is also used. This same structure is used in the decoder with an attention mask to avoid attending to subsequent positions.

The attention mechanism used by Vaswani et al. (2017) can be thought of as a function that maps a query and set of key-value pairs to an output. The query, keys, values and output are all vectors. The output is obtained as a weighted sum of the values. The weight given to a value is learned by the system by considering how compatible the query is to the corresponding key. The particular form of attention used is

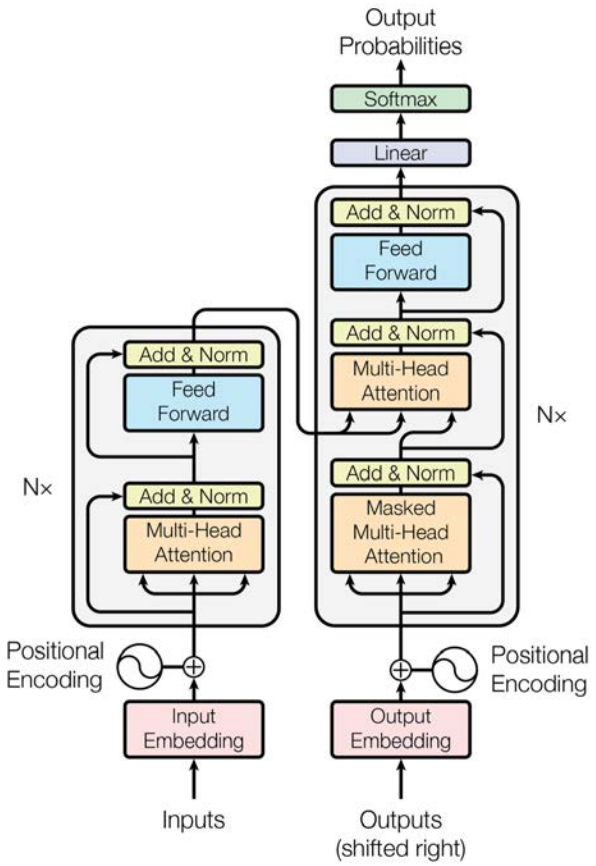


Figure 1: Transformer model as proposed by Vaswani et. al (2017).

called *scaled dot-product attention*. This is due to the mechanism being homologous to a scaled version of the multiplicative attention proposed by Luong, Pham, and Manning (2015). Several attention layers used in parallel constitute what is called *multi-head attention*.

A brief description the proposed modifications of this architecture is discussed below.

Parallel Encoding Branches

A motivation for creating the Transformer model was the sluggish training and generation time of other common sequence-to-sequence models such as RNNs and CNNs (Vaswani et al. 2017). This was done by simplifying and limiting sequential operations and computational requirements while also increasing the model's ability to exploit current hardware architecture. This paper proposes that removal of the previously stacked branches of the encoder (there is a stack of N encoder and other blocks on the left side of Figure 1), parallelizing these separate encoder 'trees', and incorporating their learned results for the decoder, which will further eliminate sequential steps and accelerate learning within current sequence-to-sequence models. The architectures discussed are modeled in Figure 2.

Alterations to this parallel transformer model were made

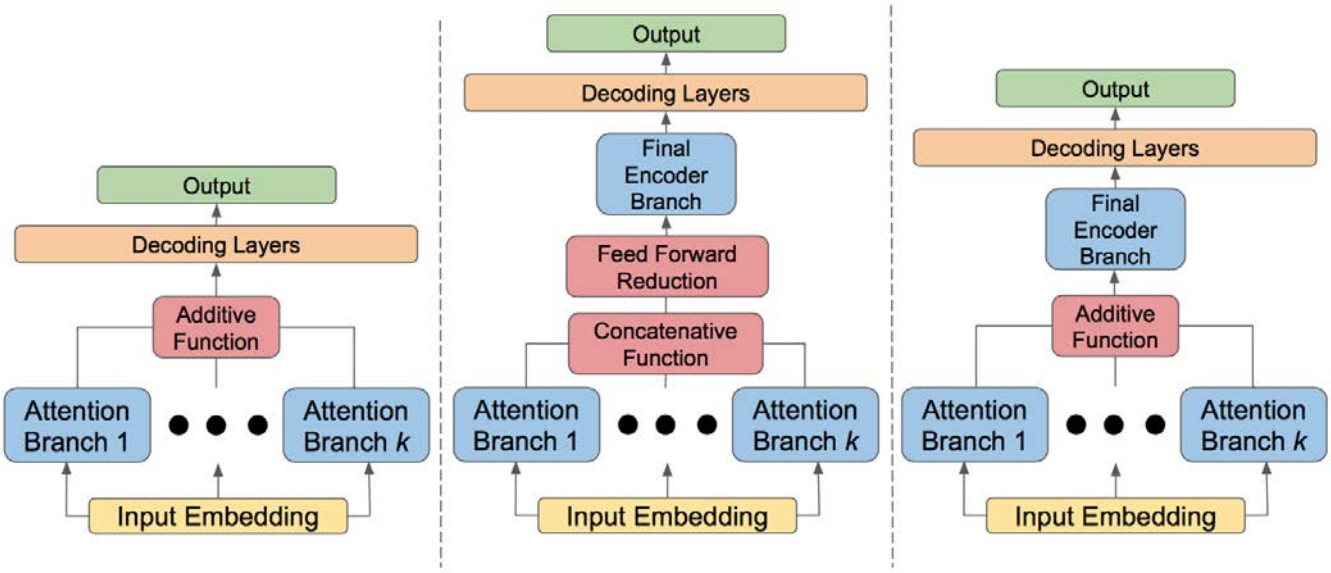


Figure 2: From left to right we present three models. 1) APA: Parallel encoded Transformer where homologous stacks of encoding trees through random initialization add their learned attention. 2) ACPA: Attended Parallel encoding where the branches concatenate learned results, a feed-forward network reduces dimensionality, and a final encoder branch encodes the results. 3) AAPA: Attended Parallel Encoding Branches where a final encoding attention branch attends the added learned results.

and the following models were trained, tested, and are discussed in this paper:

- Additive Parallel Attention (APA),
- Attended Concatenated Parallel Attention (ACPA), and
- Attended Additive Parallel Attention (AAPA).

Model Variations

Additive Parallel Attention (APA): We replace the entire stack of (multi-head attention, add and normalize, feed forward, add and normalize) repeated N times on the original Transformer architecture on the left column, on the input side. We instead have several such attention sub-networks in parallel. The output layers of these networks contain attention embeddings for the input. The values at the output layers among the stacks are added. This model is seen to the left in Fig. 2.

Attended Concatenated Parallel Attention (ACPA): This approach is similar to APA and AAPA, but the values at the output layers of the attention sub-networks are concatenated instead of being added. This model is seen in the middle of Fig. 2.

Attended Additive Parallel Attention (AAPA): This model is built similarly to the APA model. However, it removes one of the parallel stacks and uses it as a final sequential attention mechanism over the additive results. This model is seen to the right in Fig. 2.

When incorporating the results of the parallel encoding branches, two models of thought are pursued: additive and concatenation. The APA and AAPA models directly add the results of all encoding branches, whereas the ACPA models concatenate all encoding results and use a simple nonlinear layer to learn a dimension-reduction among all attention branches. The attended parts of both the ACPA and AAPA models incorporate a final attention layer over all encoding branches before they are sent to the decoding layers.

Experiments and Evaluation

All proposed architectures including the base Transformer model (Vaswani et al. 2017) are trained over the International Workshop on Spoken Language Translation (IWSLT) 2016 corpus and tested similarly over the IWSLT 2014 test corpus (Mauro, Christian, and Marcello 2012). The training corpus includes over 200,000 parallel sentence pairs, and 4 million tokens for each language. The testing set contains 1,250 sentences, and 20-30 thousand tokens for French and German. This paper also performed experiments over the larger WMT data set including 4.5 and 36 million training sentence pairs for the EN-DE and EN-FR tasks respectively. The testing set for these experiments was the standard Newstest 2014 test set including around 3000 sentence pairs for each language task. These statistics can be noted in Table 1. The sentence pairs range in length from one to sixty tokens to get a full measure of the tested models and robustness to both short and long input.

Across all models, a greedy-decoding function for both training and testing time, the Kullback-Leibler divergence loss function, the Adam optimizer (Kingma and Ba 2014), and the number of training epochs (10) were kept constant. The training and testing were done using the NMT task of English to German (EN-DE) and IWSLT English to French and English to German translation and each network was

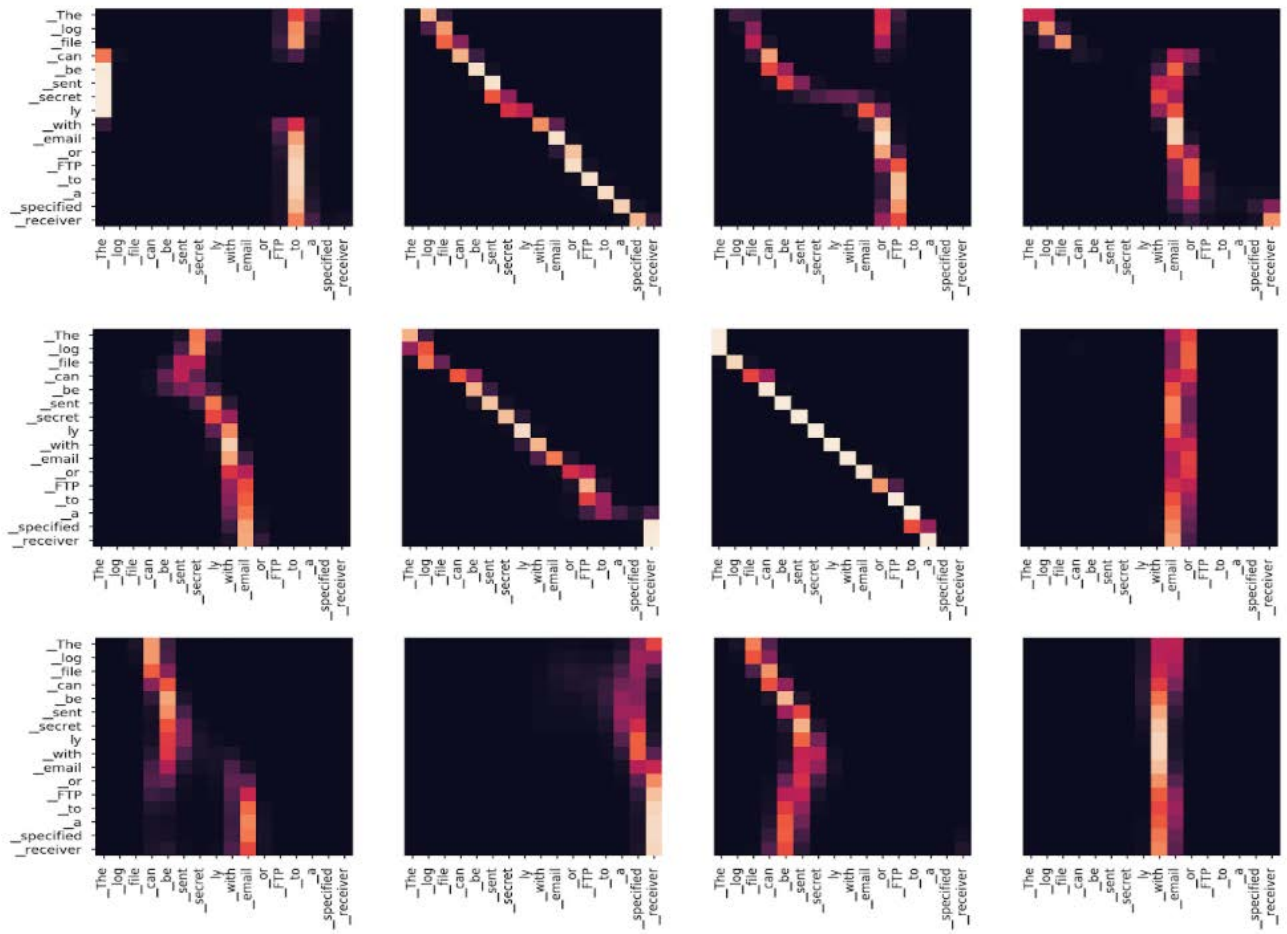


Figure 3: Visualization of multi-head attention weights in encoder branches 0, 2, and 4. Although each receives the same input embedding, through random initialization, each learns different focuses.

trained using one graphics processing unit (GPU). The utilized machine GPU configuration was one NVIDIA GTX 1070.

For the assessment of each model and translation task this paper uses the bilingual evaluation understudy (BLEU) metric (Papineni et al. 2002). This is a modified precision calculation using n-grams such as unigram, grouped unigrams, and bigrams. The BLEU metric claims to have a high correlation to translation quality judgments made by humans. BLEU computes scores for individual sentences by comparing them with good quality reference translations. The individual scores are averaged over the the entire corpus, without taking intelligibility or grammatical correctness into account.

Results

Attention Visualization

One concern during early hypothesis testing was that if each attention branch looks at the same input, that each one would learn to focus on the same properties of the original embedding. However, through visualization of each attention layer, it is obvious that regardless of the same input, the encoder branches through random initialization learn different focuses as seen in Figure 3. The final branch for the attended models however would learn very light to no attention weights as seen in Figure 4. This is one area of research this group wishes to pursue in the future.

Machine Translation

Experimentation shows in Table 2 that the AAPA model consistently performed on average nearly ten points higher in the BLEU metric on the English to German translation task on the IWSLT 2014 test set. It also performed very well on the English to French translation task. On the much larger WMT English-German test set, all our models achieve better results then Vaswani et al. (2017). Our model with five parallel encoding branches has a BLEU score of 62.69 compared to 60.95 and 61.00 for the two Transformers shown in Table 3. Our approach also takes considerably less time than the large Transformer model with a stack of eight encoder attention heads, although it is a little slower than the smaller

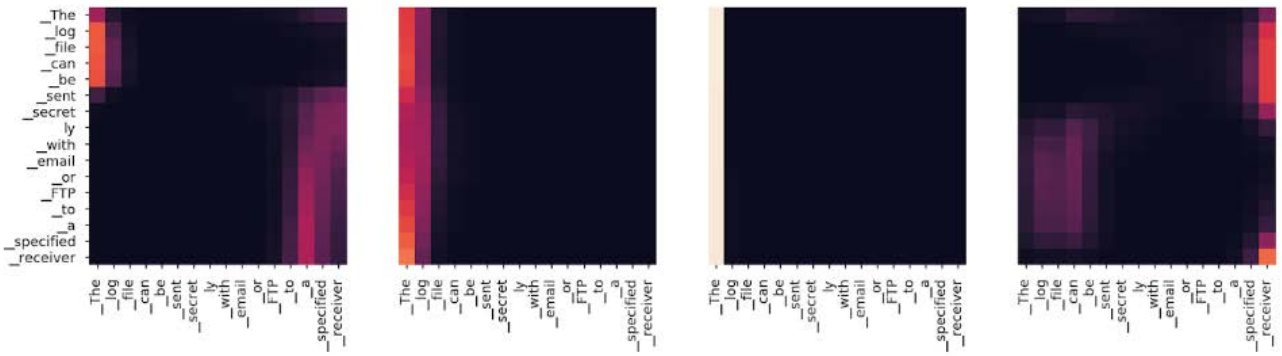


Figure 4: Visualization of the weights for the final encoder that attends over all other encoding branches. This encoder's weights are relatively light, abstract, and have less obvious patterns when compared to the individual encoding branches.

Data Set	No. Training Sentence Pairs		No. Testing Sentence Pairs	
	EN-DE	EN-FR	EN-DE	EN-FR
IWSLT (Mauro, Christian, and Marcello 2012)	197K	220K	628	622
WMT	4.5M	36M	3000	3000

Table 1: Data sets comparisons of training and testing sentence pairs for the translation tasks of English-German and English-French. The English-French statistics were included for the WMT data set although it is not directly used in the paper, as it will be included in future work.

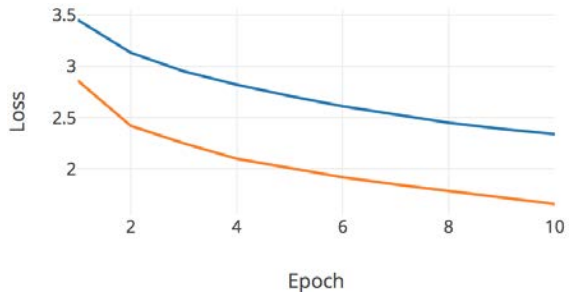


Figure 5: This plot shows validation loss for both the Transformer model (blue) and our modified model (orange) over the IWSLT EN-DE task. The parallel encoder shows a consistently lower starting and end-training loss.

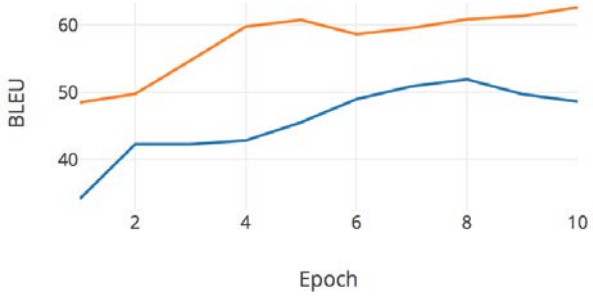


Figure 6: This plot shows validation BLEU metric score for both the Transformer model (blue) and our modified model (orange) over the IWSLT EN-DE task. The parallel encoder shows a consistently higher BLEU score and shows linear increase while the Transformer shows some plateauing in later epochs.

Transformer model reported by Vaswani et al. (2017). In terms of the BLEU metric, we establish state-of-the-art performance for both EN-DE and EN-FR translation considering both IWSLT 2014 and WMT data sets. Since our results came up very good, surpassing state of the art, we ran our experiments multiple times to ensure the results are correct. During the Transformer and attended parallel model's training lifetime, it can be seen that loss was consistently lower for our modified parallel model with five parallel stacks as seen in Figure 5. In this task, loss doesn't always correspond to a higher metric, in this case our model also shows a continuous higher score in the BLEU metric over the validation set while the Transformer shows signs of plateauing early on Figure 6.

However, our parallelized model did have a slightly higher training time over a single GPU. One final experiment conducted to improve this drawback, also seen in the same table, is the reduction of number of parallel branches in the encoder. By reducing the number incrementally, our BLEU score stays equivalent to higher perplexity layers, but linearly reduces the run-time.

Conclusions

In step with the goals of the original Transformer, this work continued to pursue the removal of sequential operations within attention-based translation models. Further work in this model requires increasing diversification of the encoder attention branches and thereby insuring varied focuses and

Model	BLEU		Single GPU Run-Time (s)	
	EN-DE	EN-FR	EN-DE	EN-FR
Transformer as proposed by Vaswani et al.(Vaswani et al. 2017)	47.57 ± 4.97	56.15 ± 0.42	8052.19	9480.70
Attended Additive Parallel Attention 5 Parallel Branches (AAPA)	$\mathbf{57.05} \pm 0.45$	$\mathbf{63.26} \pm 0.43$	8158.26	9596.08
Attended Additive Parallel Attention 4 Branches	56.22 ± 0.63	62.68 ± 0.25	7805.73	9114.84
Attended Additive Parallel Attention 3 Branches	56.68 ± 0.47	62.75 ± 0.35	7412.81	8686.92
Attended Additive Parallel Attention 2 Branches	55.94 ± 0.01	61.24 ± 0.53	**6998.18**	**8228.14**
Attended Concatenated Parallel Attention (ACPA)	48.67 ± 4.47	62.31 ± 0.21	8186.77	9710.70

Table 2: Model comparison for test results over the IWSLT 2014 test set. The BLEU score is given as an average of the final epoch over multiple runs where also a standard deviation (SD) is also given. By reducing the number of parallel branches in the encoder, the model can maintain a high accuracy and reduce run-time.

Model	BLEU	Single GPU Run-Time (s)
Transformer Large	60.95	168,806.61
Attended Additive Parallel Attention Large 7 Parallel Branches (AAPA)	61.98	173,163.03
Transformer	61.00	138,032.33
Attended Additive Parallel Attention 5 Parallel Branches	62.69	141,041.74
Attended Additive Parallel Attention 4 Branches	**62.77**	133,374.33
Attended Additive Parallel Attention 3 Branches	62.07	123,929.10
Attended Additive Parallel Attention 2 Branches	62.59	**116,450.75**
Attended Concatenated Parallel Attention (ACPA)	60.32	142,363.06

Table 3: Model comparison for test results over the larger NMT English-German test set.

a more robust encoder. This new parallelized Transformer model reaches a new state-of-the-art in machine translation and provides multiple new directions for future research.

Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant No. 1659788. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Ahmed, K.; Keskar, N. S.; and Socher, R. 2017. Weighted transformer network for machine translation. *CoRR* abs/1711.02132.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473.

Domhan, T. 2018. How much attention do you need? a granular analysis of neural machine translation architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, 1799–1808.

Elman, J. L. 1990. Finding structure in time. *Cognitive science* 14(2):179–211.

Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; and Dauphin, Y. N. 2017. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, 1243–1252.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Imamura, K.; Fujita, A.; and Sumita, E. 2018. Enhancement of encoder and attention using target monolingual corpora in neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, 55–63.

Jurafsky, D. 2000. *Speech & Language Processing*. Pearson Education.

Kalchbrenner, N.; Espeholt, L.; Simonyan, K.; Oord, A. v. d.; Graves, A.; and Kavukcuoglu, K. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Luong, T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1412–1421.

Mauro, C.; Christian, G.; and Marcello, F. 2012. Wit3: Web inventory of transcribed and translated talks. In *Conference of European Association for Machine Translation*, 261–268.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, 311–318. Association for Computational Linguistics.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, 6000–6010.

Wu, Y.; Schuster, M.; Chen, Z.; Le, Q. V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Zhou, J.; Cao, Y.; Wang, X.; Li, P.; and Xu, W. 2016. Deep recurrent models with fast-forward connections for neural machine translation. *Transactions of the Association of Computational Linguistics* 4(1):371–383.

Abstractive Summarization Using Attentive Neural Techniques

Jacob Krantz
Gonzaga University
Spokane, WA
jkrantz@zagmail.gonzaga.edu

Jugal Kalita
University of Colorado, Colorado Springs
Colorado Springs, CO
jkalita@uccs.edu

Abstract

In a world of proliferating data, the ability to rapidly summarize text is growing in importance. Automatic summarization of text can be thought of as a sequence to sequence problem. Another area of natural language processing that solves a sequence to sequence problem is machine translation, which is rapidly evolving due to the development of attention-based encoder-decoder networks. This work applies these modern techniques to abstractive summarization. We perform analysis on various attention mechanisms for summarization with the goal of developing an approach and architecture aimed at improving the state of the art. In particular, we modify and optimize a translation model with self-attention for generating abstractive sentence summaries. The effectiveness of this base model along with attention variants is compared and analyzed in the context of standardized evaluation sets and test metrics. However, we show that these metrics are limited in their ability to effectively score abstractive summaries, and propose a new approach based on the intuition that an abstractive model requires an abstractive evaluation.

Introduction

The goal of summarization is to take a textual document and distill it into a more concise form while preserving the most important information and meaning. To this end, two approaches have historically been taken; extractive and abstractive. Extractive summarization selects the most important words of a given document and combines and rearranges them to form a final summarization (Nallapati, Zhai, and Zhou 2017). This approach is restricted to using words directly from the source document and so is unable to paraphrase. Abstractive algorithms generate a summary from an attempt to understand a document's meaning, allowing for paraphrasing much like a human may do. Abstractive approaches are more difficult to develop than extractive ones because an intermediate representation of knowledge is required. As such, dominant techniques of summarization have been extractive in nature, with wide-ranging solutions utilizing statistical, topic-based, graph-based, and machine learning approaches (Gambhir and Gupta 2017). With the potential for generating more coherent and insightful summaries, abstractive approaches are gaining in popularity fueled by novel deep learning techniques (See, Liu, and Manning 2017). The abstractive summarization pipeline includes converting words to their respective embeddings, computing a document representation, and generating output words. Neural networks have recently been shown to perform well for every step (Dong 2018).

In deep learning models, attention allows a decoder to focus on different segments of an input while stepping through output regions. In the related sequence to sequence task of machine translation, attention was introduced to the existing encoder-decoder model (Bahdanau, Cho, and Bengio 2014). This resulted in large improvements over past systems due to the ability to consider a larger window of context during the output generation. Progressing this further, Vaswani et al. (2017) showed that multi-headed self-attention can replace recurrence and convolutions entirely. As the areas of machine translation and abstractive summarization are related both structurally and semantically, the developments in machine translation may inform the direction of research in abstractive summarization. In this paper, we apply these advancements and develop them further in pursuit of sentence summarization. In any attempt at summarization, the resulting text must be much more condensed than the original. In this task, all generated summaries are constrained to a fixed maximum length so that tested models must learn how to decide what information should be reproduced.

Related Work

Successful sentence summarization approaches have classically used statistical methods. TOPIARY (Zajic, Dorr, and Schwartz 2004) detected salient topics that guided sentence compression while using linguistic transformations. MOSES, a statistical machine translation system, also performed well when directly used for summarization (Koehn et al. 2007). Attention mechanisms have been shown to improve the results of abstractive summarization. Rush, Chopra, and Weston (2015) improved over classic statistical results by using a neural language model with a minimal contextual attention encoder. After the primary model training, an extractive tuning step was performed on an adjacent dataset. A related extension of this used a convolutional attentive encoder and experimented with replacing the decoder language model with RNN variants. LSTM cells and RNN-Elman both showed improved ROUGE scores (Chopra, Auli, and Rush 2016). An attentive encoder-decoder was

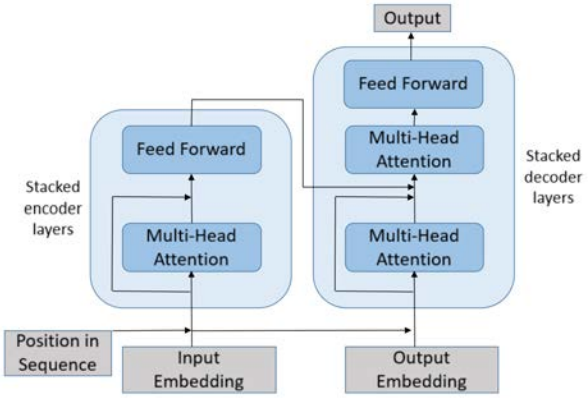


Figure 1: Transformer-based network architecture. The multi-headed attention mechanisms contain various recall options similar to and that expand upon Vaswani et al. (2017).

also employed by Zeng et al. (2016) with one RNN architecture to re-weight another to improve context across the input sequence. Their decoder used attention with a copy mechanism that differentiated between out of vocabulary words based on their usage in the input. Nallapati et al. (2016) continued progress on encoder-decoder architectures by employing a bidirectional GRU-RNN encoder with a unidirectional GRU-RNN decoder. Imposing dynamic vocabulary restrictions also improved results while reducing the dimensionality of the softmax output layer. Pointer-Generator networks encode with a bidirectional LSTM and decode with attention restriction. A coverage vector that limits the attention of words previously attended over is maintained (See, Liu, and Manning 2017).

Recently, summarization has made progress at the paragraph level due to reinforcement learning. A recurrent abstractive summarization model used teacher forcing and a similarity metric that compared the generated summary with the target summary (Paulus, Xiong, and Socher 2017). The architecture contained a bi-directional LSTM with intra-attention. Actor-critic reinforcement learning was used by Li, Bing, and Lam (2018) to produce the highest scores for sentence summarization. One important consideration when optimizing purely on the test metric is that while overall recall is improved, higher ROUGE scores do not necessarily correlate with the readability of summaries.

Models

Encoder-decoder architectures provide an adaptable structure for the development of systems that solve sequence to sequence problems. The encoder maps the input sequence to a latent vector representation. The decoder takes this representation, called the context vector, and generates the output sequence. The models and their variants that follow are structured as such. We select a base architecture that provides a strong foundation on which to analyze the effect of self-attention variants.

The Transformer

The Transformer architecture as proposed by Vaswani et al. (2017) is notable for performing state of the art Machine Translation, and is more efficient to train than past systems by orders of magnitude. This is made possible by replacing sequence aligned recurrence with self-attention. The sequence order is preserved in the self-attention modules by including positional embeddings. Instead of incremental values, the positional embeddings are determined by position on a sinusoidal time series curve. Further, masking of the decoder self-attention is performed, making the output of the next token dependent on that which has already been generated. Multi-headed self-attention is used in both the encoder and decoder. These mechanisms map a query vector to a key-value vector pair which results in an output vector. Tying together the encoder and decoder is a third multi-headed attention mechanism. The query comes from the self-attentional output of the decoder, and the keys and values from the self-attentional output of the encoder. In the work done by Vaswani et al. (2017), all attention heads used scaled dot-product attention, which is computationally efficient as multiple query, key, and value vectors can be implemented as a combined matrices. Scaled dot-product attention also defines the structure for the self-attention mechanisms we present below.

$$attention = softmax(\frac{QK^T}{\sqrt{d}})V \quad (1)$$

Many other attention mechanisms exist beyond the base dot-product attention. We analyze the performance of these mechanisms in the context of abstractive summarization. Changing the way the query, key, and value vectors interact allows an attention mechanism to learn different relationships between sequence elements.

Relative dot-product attention uses scaled dot product attention, but instead of using absolute positional encodings, uses a relative positional encoding. These relative encodings learn to relate the elements of the query to both the elements of the keys and values (Gehring et al. 2017). The encodings can be distance-limited to a context window in the vector sequences.

Local attention divides the key-value vectors into localized blocks (Liu et al. 2018). Each query is strided over a corresponding block with a given filter size. Blocks can contain positions both prior to and following a given position, thereby not masking any element based on absolute position. Self-attention is performed over each block in isolation.

Local masked attention adds a mask to the blocks of local attention. Blocks in a future sequential position are masked from the query but all elements within a block remain visible to a given query position. Intuitively, masking future positions forces a mechanism to attend to current and past positions, which may be an important restriction of the attention distribution.

Local block masked attention masks both previous blocks and future blocks for a query position. Further, future positions within individual blocks are masked.

Dilated attention also divides the key-value vectors into blocks, but introduces a gap in between each block. Each

Target	*Endeavour astronauts join two segments of International Space Station.*
Gen1	Endeavour astronauts join two **sections** of International Space Station.
Gen2	Endeavour astronauts **remove** two segments of International Space Station.
Gen3	Endeavour astronauts join two segments of International Space Station.

Sentence	ROUGE-1	ROUGE-2	ROUGE-l	Cos-Sim	WMD	VERT
Gen1	88.89	75.00	88.89	0.979	0.418	94.77
Gen2	88.89	75.00	88.89	0.924	0.512	91.08
Gen3	100.00	100.00	100.00	1.000	0.000	100.00

Table 1: Highlighted differences between ROUGE and VERT scoring. Notice that an incorrect word replacement (*Gen2*) scores the same as a reasonable word replacement (*Gen1*) in ROUGE. VERT discounts the score of *Gen2* accordingly. *Gen3* is included to show the perfect scores for an identical summary.

query position is limited to a context window of a specified number of blocks both preceding and following the memory position.

Dilated masked attention performs the same operations as dilated attention and masks future memory positions within each block.

Evaluation

The standard test metric for automatic summary generation is ROUGE, or Recall-Oriented Understudy for Gisting Evaluation (Lin 2004). Before the ROUGE metrics were introduced, human judges were used for summary evaluation. Human judges provide an ideal evaluation, but are impractical for regular use. ROUGE allows for easy comparison of generated summaries to target summaries, where target summaries are human-generated. Limited-length recall is commonly reported using ROUGE-1, ROUGE-2, and ROUGE-L. ROUGE-1 and ROUGE-2 compare unigram and bigram overlap, respectively. This generalizes to ROUGE-N for n-gram overlap. ROUGE-L determines the longest common subsequence (LCS). Evaluation quality of summarization models can be directly compared to previous work because the same metrics were reported for past models by Rush, Chopra, and Weston (2015), Zeng et al. (2016), Nallapati et al. (2016), Li, Bing, and Lam (2018), and others. These metrics allow for reasonably accurate comparison of summary generation models, but inherent problems exist. One critical limitation is that ROUGE does not consider reasonable paraphrasing or synonymous concepts. Since ROUGE works at the word level, meaning can only be captured and compared in a binary manner; either a word appears in the generated summary or it does not.

ROUGE 2.0 was proposed to alleviate this problem as well as remove the expectation that generated summaries need to be identical to the target summary (Ganesan 2015). As pointed out by Rush, Chopra, and Weston (2015), even the best human evaluator scored just 31.7 ROUGE-1 on the DUC2004 dataset. This illustrates the idea that two summaries do not need to be the same in order for both to be of high quality. Thus, a more appropriate approach to summary comparison may be to evaluate the semantic similarity between the generated and target summaries instead of using isolated word counts. ROUGE 2.0 captures semantic similarity using a synonym dictionary while still evaluating n-grams and LCS. While this addresses the word-level shortcoming of the original ROUGE metrics, similarity is still fixed to a discrete list of acceptable alternatives, which does not fully capture phrase substitution. A further improvement could be to evaluate the semantic similarity between two entities on a continuous scale.

VERT Metric

To improve the quality of summary evaluation, we introduce the VERT metric[1], an evaluation tool that scores the quality of a generated hypothesis summary as compared to a reference target summary. VERT stands for Versatile Evaluation of Reduced Texts. VERT compares summaries on their underlying semantics rather than word count ratios. To calculate a VERT score for a summary pair, a similarity sub-score and dissimilarity sub-score are calculated and functionally combined. Naturally, a higher similarity score and a lower dissimilarity score leads to a higher, better VERT score. The similarity sub-score considers the semantics of each summary taken at the document level. A sentence embedding vector is synthesized for both generated and target summaries, and the cosine similarity between these two vectors provides the similarity score. The sentence embeddings are generated using *InferSent*, an open-source neural encoder trained on natural language inference tasks (Conneau et al. 2017). *InferSent* was chosen because it has been shown to generalize well for use in various problems requiring sentence representations. The dissimilarity sub-score operates at the individual word level rather than at the sentence level. An aggregate Euclidean distance is calculated between the words of the generated summary and the words of the target summary. This is done using word mover's distance (WMD), a measure of how far document A must travel to match document B within the word vector space (Kusner et al. 2015). Stop words are discarded prior to the distance calculation as their effect on the distance between documents is negligible.

Sub-Score Motivations

A consideration would be to use just one of the two subscores as they are independent calculations. However, both the *InferSent* cosine similarity and WMD are made more robust by the presence of the other score. WMD is unaffected by word ordering, whereas the encoder of *InferSent*

1. Our VERT implementation is made publicly available at: `https://github.com/jacobkrantz/VertMetric`

Metric	Pearson	P-Value
ROUGE-1	0.3039	0.0319
ROUGE-2	0.2577	0.0708
ROUGE-L	0.3071	0.0300
VERT	0.3681	0.0085

Table 3: Pearson correlation coefficient between automatic metrics and human evaluation of responsiveness.

WMD	Summary Count
$0 \rightarrow 1$	74
$1 \rightarrow 2$	860
$2 \rightarrow 3$	2858
$3 \rightarrow 4$	2150
$4 \rightarrow 5$	58
$5+$	0

Table 2: WMD among human summaries on DUC2004. For each article, every human summary was held out as the target to compare the other human summaries to resulting in 6000 comparisons.

maintains sequential input. To illustrate, suppose the target sentence is *"go right and then left"* and the generated sentence switches the order, stating *"go left and then right."* WMD gives this a perfect distance of 0.0 but the *InferSent* similarity more accurately discounts the score by 4.3%. On the other hand, when longer summaries are compared, *InferSent* embeddings begin to lose the effect of individual words because the word embeddings are replaced with a singular embedding. This is less of a problem for WMD. Finally, the similarity sub-score uses GloVe embeddings[2] pretrained on Common Crawl while the dissimilarity sub-score uses Word2Vec[3] trained on the Google News dataset. Using different word embeddings provides resistance to potential learned representation biases.

Formula Specification

The similarity sub-score is defined as $sim(s_1, s_2) = cos(encode(s_1), encode(s_2))$ and the dissimilarity sub-score is defined as $dis(s_1, s_2) = min(wmd(s_1, s_2), \alpha)$. The maximum dissimilarity value α is the default distance when all of the generated words are out of vocabulary. Without this default, summaries with no words to compare would have an infinite distance and too strongly influence VERT score averages. Resulting sub-score values range as such: $0.0 \leq sim(s_1, s_2) \in \mathbb{R} \leq 1.0$, and $0.0 \leq dis(s_1, s_2) \in \mathbb{R} \leq \alpha$. We seek to combine these scores such that the final VERT score can be treated as a percentage: $0.0 \leq VERT(s_1, s_2) \in \mathbb{R} \leq 1.0$. Further, $sim(s_1, s_2)$ and $dis(s_1, s_2)$ should be given equal weight in the final VERT score. To satisfy both criteria, we present the VERT equation:

$$VERT(s_1, s_2) = \frac{1}{2}(1 + (sim(s_1, s_2) - \frac{1}{\alpha} dis(s_1, s_2))) \quad (2)$$

where $\alpha = 5.0$. The dissimilarity is normalized by α and the outer linearity, as multiplied by $\frac{1}{2}$, shifts the range from $[-1.0, 1.0]$ to $[0.0, 1.0]$. For the choice of α, we observe an empirical distance ceiling of 5.0 in Table 2. Incorporating this ceiling gives both sub-scores equal precedence while removing the necessity of a nonlinearity, such as normalization by the hyperbolic tangent.

Hyperparameters and Baseline

The similarity sub-score uses a pre-trained *InferSent* encoder for reproducibility, and thus needs no hyperparameter adjustments. The dissimilarity requires just the hyperparameter α to specify the maximum threshold of WMD and can stay at the default value of 5.0. With the same value used to normalize the dissimilarity, VERT is straightforward to use with just this single hyperparameter. To provide a scoring reference, we test each human summary of DUC2004 on VERT using the same holdout process as done in Table 2. The average similarity sub-score is 0.74875, the average dissimilarity sub-score is 2.71700, and combined the average VERT score is 0.60268.

Comparison to Human Evaluation

To evaluate the effectiveness of VERT, we calculate the correlation between VERT scores and scores given by human judges. Using the relative dot product attention model, 50 summaries are generated on the DUC2004 dataset and evaluated with the VERT metric by averaging the VERT scores between the four target summaries. We then conduct an experiment in which two human evaluators score the 50 generated summaries based on the DUC 2006 Responsiveness Assessment[4]. The primary consideration of responsiveness is the amount of information in the summary that relates to the original sentence. The evaluators score the level or responsiveness on a 5-point Likert scale, with 5 being the best possible. Table 3 shows that VERT correlates with human judgment of responsiveness stronger than all three standard ROUGE metrics.

Experiments

Experiment Setup

The environment and evaluation of all models strictly follow the precedent set by Rush, Chopra, and Weston (2015). For both training and testing, we extract sentence-summary pairs from news articles. The first sentence of each article is treated as the sentence to be summarized, while the headline of the article acts as the target summary.

Datasets

The training data comes from the Gigaword dataset, which is a collection of about 4 million news articles (Graff et al. 2003). It is necessary to discard certain article-headline pairs as some news articles open with a sentence that poorly relates to the headline, such as a question. Preprocessing tasks includes filtering, PTB tokenization, lower-casing, replacing digit characters with #, and replacing low-frequency words with *UNK*. Evaluation for hyperparameter tuning is

2. https://nlp.stanford.edu/projects/glove/
3. https://code.google.com/archive/p/word2vec/
4. https://duc.nist.gov/duc2007/responsiveness.assessment.instructions

Mechanism	RG-1	RG-2	RG-L	VERT-S	VERT-D	VERT
s-dot-prod	25.72	8.51	23.08	0.73523	2.76307	59.13
rel-s-dot-prod	**27.05**	**9.54**	**24.44**	**0.73876**	**2.73907**	**59.55**
local	1.93	0.00	1.93	0.02084	5.00000	1.04
local-mask	25.72	8.54	23.30	0.73361	2.77857	58.89
local-blk-mask	14.13	2.75	12.63	0.67226	3.18881	51.73
dilated	0.01	0.00	0.01	0.09509	3.66543	18.10
dilated-mask	19.06	5.23	17.45	0.68682	3.04922	53.85

Table 4: Comparison of attention mechanisms using DUC2004. RG represents ROUGE-Recall, VERT-S is the *InferSent* cosine similarity sub-score, and VERT-D is the average word mover's distance.

Dataset	# Articles	Sent Len	Sum Len
Gigaword	3803957	31.4	8.300
DUC2003	624	32.7	11.242
DUC2004	500	31.3	11.710

Table 5: Comparison of general dataset details. Sentence and summary lengths are reported as the average word count. Gigaword has noticeably shorter target summaries than either DUC dataset. To counteract the models generating too short of summaries, we augment the beam search decoding probabilities to encourage longer summaries.

performed on the DUC2003 dataset[5]. Testing is done on the DUC2004 dataset[6] where the summaries are capped at a length of 75 bytes. For both DUC2003 and DUC2004, each article has four target summaries to be compared against. For processing Gigaword, we used the same data provided by Rush, Chopra, and Weston (2015), but both DUC datasets had to be preprocessed according to the tasks specified. Certain sentence-summary pairs within DUC 2004 poorly relate to each other due to the fact that the human-generated summaries used the context of the entire DUC article to decide on an adequate summary. Since this shortcoming is present across all models attempting sentence summarization on DUC, we made no effort to remove these difficult pairings from the test set.

Base Implementation

For the hyperparemeter specification, every model used 8 attention heads and dense feed forward layers had dimensions of 2048. Cross entropy was used for the loss function, and optimization was performed with the Adam optimizer using a variable learning rate to encourage final convergence. Training required approximately 25 epochs. A promising feature of using an attention-based architecture is that the models used here are capable of being trained in approximately 4 hours on a single GPU, whereas recent state of the art recurrent summarization models have been mentioned to take 4 days (Rush, Chopra, and Weston 2015). We implemented these models using the Tensor2Tensor[7] library backed by TensorFlow. A strong local minimum exists when training, which closely relates to extracting the first n words of the input text up to 75 bytes. Such a trivial approach produces relatively high ROUGE scores simply due to the natural similarity between target summaries and input sentences. Visualization of the attention heads showed that each head attended directly across to the corresponding word in the input sequence during decoding. Diversity of attention can be encouraged by varying the learning rate and modifying the attention mechanism itself. For the decoding step, beam search is used with a beam size of 8. This results in ROUGE scores that are higher than a more simple greedy inference. Decoding to a fixed length of 75 bytes does not align easily with word-level decoding, so for the implementation we approximate the cutoff by limiting the summary sequence to 14 words.

Results

Attention Comparisons

For each of the attention mechanisms described above, we performed a full scale analysis of their performance by training each model on the Gigaword dataset and evaluating on DUC2004. For each experiment, the foundational architecture was held constant. We modified both the encoder self-attention and decoder self-attention to perform as specified by the given attention mechanism. In Table 4, the model that used scaled dot product attention acted as the baseline (s-dot-prod). The highest performing mechanism was relative scaled dot product attention, showing that relative positional encodings can be more insightful than absolute encodings. This demonstrates that token generation may rely more heavily on the relationships between surrounding words than relationships at a global sequential level. Local masked attention attained identical ROUGE-1 scores to scaled dot-product attention with marginally higher ROUGE-2 and ROUGE-L scores. However, scaled dot-product attention scored noticeably higher with VERT, primarily due to the similarity sub-score. This suggests the scaled dot-product model is better than the local-mask model when considering the summary semantics across the full length of sequences. Both local and dilated attention mechanisms performed poorly, repeating the same words regardless of input sentence; both masked counterparts did not have this problem.

An interesting observation during the training process of the attention models was the high dependence on batch size. Models would not converge when batch sizes were at or below 2000 tokens per batch. The batch size used to train the above models was 8192 tokens. Some attention models, dilated attention and dilated-mask attention, had higher mem-

5. https://duc.nist.gov/duc2003/tasks.html
6. https://duc.nist.gov/duc2004/
7. https://github.com/tensorflow/tensor2tensor

Model	RG-1	RG-2	RG-L	VERT
TOPIARY (Zajic, Dorr, and Schwartz 2004)	25.12	6.46	20.12	-
ABS (Rush, Chopra, and Weston 2015)	26.55	7.06	22.05	58.49
RAS-LSTM (Chopra, Auli, and Rush 2016)	27.41	7.69	23.06	-
MOSES+ (Koehn et al. 2007)	26.50	8.13	22.85	-
RAS-Elman (Chopra, Auli, and Rush 2016)	28.97	8.26	24.06	-
ABS+ (Rush, Chopra, and Weston 2015)	28.18	8.49	23.81	59.05
RA-C-LSTM (Zeng et al. 2016)	29.89	9.37	25.93	-
words-lvt5k-1sen (Nallapati et al. 2016)	28.61	9.42	25.24	-
S-ATT-REL (ours)	27.05	9.54	24.44	59.55
AC-ABS (Li, Bing, and Lam 2018)	32.03	10.99	27.86	-

Table 6: ROUGE-recall scores of compared models on DUC2004. Sorted by ROUGE-2 score. VERT scores for ABS and ABS+ were calculated using generated summaries provided by Rush, Chopra, and Weston (2015). Other authors were contacted for summaries from their models but did not respond.

S(1): exxon corp. and mobil corp. have held discussions about combining their business operations , a person involved in the talks said wednesday .
Target: exxon corp. and mobil corp. may combine business operations
S-ATT-REL: exxon and mobil discuss merger

S(2): prime minister rafik hariri , the business tycoon who launched lebanon 's multibillion dollar reconstruction from the devastation of civil war , said monday he was bowing out as premier following a dispute with the new president .
Target: prime minister hariri , claiming constitution violation , bows out
S-ATT-REL: lebanese prime minister resigns after dispute with new president

S(3): organizers of december 's asian games have dismissed press reports that a sports complex would not be completed on time , saying preparations are well in hand , a local newspaper said friday .
Target: bangkok says sports complex will be completed in time for asian games
S-ATT-REL: asian games organizers say sports complex will not be completed on time

S(4): a struggle for control of the house is under way , with rep. robert livingston conducting a telephone campaign that could lead to him running against newt gingrich as speaker .
Target: election of gingrich as house speaker in doubt as small group opposes him
S-ATT-REL: house speaker 's phone campaign could lead to gingrich

S(5): premier romano prodi battled tuesday for any votes freed up from a split in a far-left party , but said he will resign if he loses a confidence vote expected later this week .
Target: italian premier to resign if he loses pending confidence vote
S-ATT-REL: italy 's prodi says he will resign if he loses confidence vote

Figure 2: Examples of generated summaries by the relative dot-product self-attention model.

ory requirements and had to be trained at lower batch sizes. This may have negatively effected their results.

Model Comparisons

We compare our best model with past work by comparing published ROUGE scores. Slight variances may be present in the reported metrics due to potential differences in data preprocessing routines. In Table 6, we compare our best model with that of published results. The relative dot-product self-attention model (S-ATT-REL) beats all ROUGE scores of ABS, but has a lower ROUGE-1 when ABS is tuned with an extractive routine on DUC2003 (ABS+). S-ATT-REL is comparable to but lower than most models when it comes to ROUGE-1 scores. However, over the longer subsequence comparisons of ROUGE-2 and ROUGE-L, S-ATT-REL performs very well. This can be attributed to the ability of self-attention mechanisms to retain a strong memory over past elements of both the input and decoded sequences. Only the actor-critic method (AC-ABS) beats S-ATT-REL in all ROUGE categories.

Qualitative Discussion

The summaries generated by our best model are strongly abstractive, illustrated by Example *S(1)* in Figure 2. Example *S(2)* showcases the ability to utilize long range recall. From the appositive phrase, the model determined that Hariri was the prime minister of Lebanon and adjusted the morphology of the country for succinctness. The model also determined Hariri was resigning based on the words "bowing out". Occasionally, attention heads are misdirected and attend to words or phrases that do not contain the primary meaning. This occurred in Example *S3* with was incorrectly modified by the inclusion of "not". The generated summaries exhibit information beyond what was directly in the input sentence; Example *S5* correctly identifies Premier Romano as Italian which greatly improves the informedness of the summary. A primary strength of the self-attentive model is incorporating abstract information from all segments of the input sentence. This is suggested in the long subsequence ROUGE scores above, and seen clearly in qualitative analysis.

An assessment of linguistic quality[8] was performed alongside the DUC Responsiveness Assessment. This followed the same procedure detailed in Section . Questions pertained to grammaticality, non-redundancy, referential clarity, and structure and coherence. Grammaticality scored 4.48, non-redundancy scored 4.95, referential clarity scored 4.7, and structure and coherence scored 4.53. All scores averaged between "Good" and "Very Good". Non-redundancy is nearly perfect, likely because the summaries are too short for redundancy to likely be of issue. The referential clarity scored high as well, which can be associated with the performance of the self-attention over the the words already decoded.

Conclusion

The effect of modern attention mechanisms as applied to sentence summarization has been tested and analyzed. We have shown that a self-attentional encoder-decoder can perform the sentence summarization task without the use of recurrence or convolutions, which are the primary mechanisms in state of the art summarization approaches today. An inherent limitation of these systems is the computational cost of training associated with recurrence. The models presented can be trained on the full Gigaword dataset in just 4 hours on a single GPU. Our relative dot-product self-attention model generated the highest quality summaries among tested models and displayed the ability of abstracting and reducing complex dependencies. We also have shown that n-gram evaluation using ROUGE metrics falls short in judging the quality of abstractive summaries. The VERT metric has been proposed as an alternative to evaluate future automatic summarization based on the premise that an abstractive summary should be judged in an abstractive manner. For future directions of research, reinforcement learning could be applied to the core self-attention model. Also the models presented should be tested with longer summaries as they displayed strong recall over long subsequences.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1659788. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Chopra, S.; Auli, M.; and Rush, A. M. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *NACL-HLT*, 93–98.

Conneau, A.; Kiela, D.; Schwenk, H.; Barrault, L.; and Bordes, A. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Dong, Y. 2018. A survey on neural network-based summarization methods. *arXiv preprint arXiv:1804.04589*.

Gambhir, M., and Gupta, V. 2017. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review* 47(1):1–66.

Ganesan, K. 2015. Rouge 2.0: Updated and improved measures for evaluation of summarization tasks. *arXiv preprint arXiv:1803.01937*.

Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; and Dauphin, Y. N. 2017. Convolutional sequence to sequence learning. In *ICML*, 1243–1252.

Graff, D.; Kong, J.; Chen, K.; and Maeda, K. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia* 4:1.

Koehn, P.; Hoang, H.; Birch, A.; Callison-Burch, C.; Federico, M.; Bertoldi, N.; Cowan, B.; Shen, W.; Moran, C.; Zens, R.; et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Annual meeting of the ACL on interactive poster and demonstration sessions*, 177–180.

Kusner, M.; Sun, Y.; Kolkin, N.; and Weinberger, K. 2015. From word embeddings to document distances. In *International Conference on Machine Learning*, 957–966.

Li, P.; Bing, L.; and Lam, W. 2018. Actor-critic based training framework for abstractive summarization. *arXiv preprint arXiv:1803.11070*.

Lin, C.-Y. 2004. Rouge: A package for automatic evaluation of summaries. *Proceedings of the ACL Workshop: Text Summarization Branches Out*.

Liu, P. J.; Saleh, M.; Pot, E.; Goodrich, B.; Sepassi, R.; Kaiser, L.; and Shazeer, N. 2018. Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*.

Nallapati, R.; Zhou, B.; dos Santos, C. N.; Gülçehre, Ç.; and Xiang, B. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Conference on Computational Natural Language Learning*, 280–290.

Nallapati, R.; Zhai, F.; and Zhou, B. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*, 3075–3081.

Paulus, R.; Xiong, C.; and Socher, R. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Rush, A. M.; Chopra, S.; and Weston, J. 2015. A neural attention model for abstractive sentence summarization. In *EMLP*, 379–389.

See, A.; Liu, P. J.; and Manning, C. D. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 1073–1083.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *NIPS*, 6000–6010.

Zajic, D.; Dorr, B.; and Schwartz, R. 2004. Bbn/umd at duc-2004: Topiary. In *Proceedings of the HLT-NAACL 2004 Document Understanding Workshop, Boston*, 112–119.

Zeng, W.; Luo, W.; Fidler, S.; and Urtasun, R. 2016. Efficient summarization with read-again and copy mechanism. *arXiv preprint arXiv:1611.03382*.

8. `https://duc.nist.gov/duc2007/quality-questions.txt`

Hierarchical Text Generation using an Outline

Mehdi Drissi[1] **Jugal Kalita**[2]
[1]Department of Computer Science, Harvey Mudd College, Claremont, USA
[2]Department of Computer Science, University of Colorado Colorado Springs, Colorado Springs, USA

Abstract

Many problems in natural language processing require generating text. This includes problems like language translation, dialogue generation, and speech recognition. For all of these problems, text generation becomes more difficult as the text becomes longer. One difficulty with current language models is they often struggle to keep track of coherence for long pieces of text. Here, we attempt to have the model construct and use an outline of the text it generates to keep it focused. We find that the usage of an outline improves perplexity. For initial experiments, we do not find that using the outline improves human evaluation over a simpler baseline. Similarly, hierarchical generation is not found to improve in human evaluation.

Introduction

Recurrent neural networks have been successfully used for a variety of tasks in dealing with natural language. Successes include language translation (Sutskever, Vinyals, and Le 2014), speech recognition (Graves 2012), and text to speech (Kalchbrenner et al. 2018). They all learn to model the conditional probability of a sequence. They are usually approached by sequence to sequence models. These models have the advantage that their negative log likelihood is differentiable allowing them to be directly trained through gradient descent.

A similar task is language modeling. Here the goal is to determine the probability of a sequence of words. Being able to model text is important for natural language understanding. These models can be used for detecting possible errors in sentences, determining possible ways to extend a piece of writing, and generating text. Language modeling is also commonly done with recurrent neural networks by directly optimizing the negative log likelihood.

One shared difficulty in all of these problems is although in theory a recurrent model can preserve information for arbitrarily long sequences, in practice recurrent models tend to struggle to keep track of context as the sequence length becomes high. Models for tasks like language translation tend to avoid translating entire paragraphs and instead focus on only generating a sentence at a time. Similarly, when you generate multiple sentences of text from language models, they tend to be locally coherent, but not globally coherent.

The difficulty of generating large samples also arises in generating images. In the realm of images though, a different technique is commonly used for generation. Here, generative adversarial networks (Goodfellow et al. 2014) have been successfully used to generate images directly. Similar to text, initially these models only worked well for generating small images. Generating large images (like 1024 by 1024) was difficult for these models. In some recent work, the issue of large images was dealt with by generating images in a hierarchical manner. Instead of directly learning to generate the desired image, lower resolution images were first generated and then improved upon (Zhang et al. 2016). Initially, this was done by generating one lower resolution image and then directly the final image. More recently, this has been extended to starting off with generating a small image and iteratively increasing its resolution by double until reaching the desired size (Karras et al. 2017).

Inspired by the idea of generating images in a hierarchical manner, here we will explore generating text in a hierarchical manner. Similar to (Zhang et al. 2016), we will approach this by generating in two steps. One difficulty with hierarchical text generation, is meaningfully down sampling text is more difficult than down sampling images. The textual equivalent of decreasing resolution is summarization, which is a difficult problem in itself. To side step this issue we will use a simple extractive summarization approach to get an outline. Using an extractive summarization approach to acquire information to build upon has been done previously in generating Wikipedia articles (Liu et al. 2018). The difference is there they used the summarization to extract relevant information from references to generate the article, while here we are applying the summarization to the target text to acquire an outline of the text. The hierarchical aspect will be also having a separate model component to directly generate the outline. That way when we want to generate a complete document, we first generate the outline, and then condition upon the outline to generate the entire document.

Our main contribution will be to explore generating text in a hierarchical manner by separating text generation into two phases. The first phase generates an outline of the text, while the second phase uses the outline to generate the complete document.

Background

The general framework of sequence to sequence models is to have an encoder and decoder model (Sutskever, Vinyals, and Le 2014). For our purposes, we will be using convolutional sequence to sequence models (Gehring et al.). The encoder model's purpose is to take in an input sequence and construct representations of each of the token in the sequence. The decoder's hidden state is initialized based upon the encoder's final hidden state. At each step the decoder predicts the next token until it predicts an end of sequence token.

As it is difficult to encode an entire sequence into one vector, generally the decoder is allowed to look back at the representations of the tokens the encoder created through an attention mechanism (Bahdanau, Cho, and Bengio 2014). In an attention mechanism the decoder's hidden state is scored against the encodings of all the tokens in the input sentence. Those scores are converted to probabilities and then each of the encodings is weighted by its probability to be focused upon to determine the context vector. This context vector is then fed in to the decoder to aid it in keeping track of information from the entire sentence.

One weakness of conventional attention is it only allows focusing on words in the source sequence and not in the target sequence. Self-attention (Vaswani et al. 2017) is a modification that attends on both words in the source and prior words in the target. This is especially important for models that generate long sequences to be able to keep track of what has been made. A second weakness of conventional attention is it only operates at the word level. This ignores that for the outlines we condition upon they are built from sentences. An extension of attention to account for both word level and sentence level information is hierarchical attention (Ling and Rush 2017).

For summarization, approaches fall into two primary categories. Extractive summarization focuses on choosing the main sentences/words from the document and then having the summary consist of directly copying those words. Abstractive summarization focuses on having a model generate the words directly for the summary. As our goal is to simply have an outline of the text extractive summarization is sufficient. It would be interesting future work to see how different methods of generating an outline affect document generation.

The summarization algorithm that will be used is called SumBasic (Nenkova and Vanderwende 2005). This algorithm is based upon choosing sentences with words that are frequent in the document and after choosing a sentence, down-weighting those words to avoid choosing sentences that are too similar.

Methods

The main idea is to generate text in a hierarchical manner by generating the topic sentences of the document first and then generating the entire paragraph by conditioning on the topic sentence.

Summarization

As the primary focus of this work is not on new methods of summarization, prior text summarization methods will be used. SumBasic (Nenkova and Vanderwende 2005) is one frequency based method for determining the topic sentence. It tries to find sentences whose words are common in the document. One way to avoid overweighting common words like 'the' is to penalize words that are common across many articles. TFIDF (Term Frequency Inverse Document Frequency) does this by multiplying by the negative log probability of a word appearing in a document. SumBasic is often tweaked to use TFIDF instead of directly using word frequencies (Allahyari et al. 2017).

As the outline is intended to contain information about each section of the text instead of directly applying to SumBasic to the full document, it will be applied at the meta-paragraph level. Here, meta-paragraph does not refer to the actual paragraphs in the text because their lengths are very inconsistent. For many of the documents in the training data, the paragraphs only contain one or two sentences. Extracting a topic sentence from each of these paragraphs as an outline would be problematic as we would effectively be letting the outline contain too much of the document. To avoid this issue, the actual paragraphs will be aggregated together using the rule that any paragraph under a threshold number of sentences k will be combined with the next paragraph to form a meta-paragraph. As a side effect every meta-paragraph except for possibly the last one will end up having at least k sentences.

Lastly, some preprocessing is done before directly applying SumBasic. Stop words are removed using a list of nltk's English stop words, numbers are removed, punctuation is removed, and words are stemmed using Porter stemming (Porter 1980).

Model

The model will be an extension of the model used in Neural Story Generation (Fan, Lewis, and Dauphin 2018). In that work, a sequence to sequence model was used to convert that prompt into a complete document. Our main extension is dividing the sequence to sequence model into two components to assist in the coherence for the generated text. The overview of the two components is there will be one for generating topic sentences and one for expanding topic sentences into complete paragraphs.

For generating the document from the prompt we will first generate the outline associated to the document. This component will be trained using almost the same type of architecture as the sequence to sequence model used by (Fan, Lewis, and Dauphin 2018). The one difference is we did not use the cold fusion mechanism, mainly to lower needed training time.

The second component will use the outline to generate the entire document. It will also be based upon the sequence to sequence model used by (Fan, Lewis, and Dauphin 2018) and similarly will not include cold fusion. It will be extended in one way. The self attention heads in the decoder will remain the same, but the attention over the encoded outline

will be replaced by a hierarchical attention. The sentence vectors will be obtained by summing the encoded word vectors for each word in the corresponding sentence. The sentence vectors then serve as the key and value for the attention, while the query is the decoder's activation at the layer the attention is used. Similar to how the prior decoder attention was gated, the hierarchical attention is also gated and uses the same gating. The gating used is a sequence of layers consisting of a linear layer, GLU (gated linear unit) (Dauphin et al. 2016), linear layer, GLU, and a final linear layer. Both model components will be trained separately by directly optimizing the negative log likelihood.

Lastly, strictly speaking the probability of an entire document can only now be obtained by marginalizing over all possible outlines. A lower bound for the probability can be obtained by instead generating the most likely outline for a given prompt and then finding the probability of the document conditioned on that outline. This lower bound however is intractable to compute as it involves generating many outlines and due to the long length of the sequences and the relatively slow generation time, even only using 10 outlines for the approximation it would take approximately 100 days just to evaluate on the validation/test set on one 1080 Ti. It also turns out to be intractable as the memory needed to do a beam search with such long sequences is too high and it runs out of memory if you increase the beam size beyond 2.

As the two components are trained separately, this similarly leads to the overall training loss not corresponding to the actual story negative log likelihood. As using that loss properly would require being able to efficiently compute the probability of a document, this is intractable for the reasons given in the prior paragraph.

A second discrepancy that arises in the model is that due to training the components separately, the second component is only trained on good outlines. It's never trained to deal with poorly generated outlines, so if the first component generates a bad outline, the second component is unlikely to be able to recover from the mistake. This is unlike the training for StackGAN (Zhang et al. 2016) where the second part of the model was trained using the output of the first part of the model. The primary difficulty from using this training for the current model is the slow generation time that would end up increase the training time by a factor of roughly 30 (the value partly depends on the dataset used).

Evaluation Approach

Datasets

The main dataset that will be used is the Wikitext-103 dataset described in (Merity et al. 2016). It consists of a large collection of preprocessed wikipedia articles. As this dataset does not come with prompts, the prompt used will simply be the first sentence of the article with the goal being generating the entire article. This dataset also had preprocessing done primarily to eliminate tables (which were difficult to distinguish from paragraphs), to canonicalize numbers, and to lowercase everything.

A second relevant dataset for story generation was a collected from a subreddit called WritingPrompts. Here, topics were created with an story prompt and users responded with stories (Fan, Lewis, and Dauphin 2018). This dataset is available on github[1], and mostly the same text preprocessing will be used. The main difference is that quotation marks and numbers were canonicalized, markup was removed, and the text was lowercased. To still be able to compare against them we will re-train the models they used. Due to the length of training time on this dataset experiments on this dataset will primarily be done in future work.

Possible Evaluation Metrics

The two automated evaluation metrics used for previously for the story dataset are perplexity and prompt relevance. Both are problematic for the hierarchical model as they involve computing the probability of a story. Due to the previously discussed computational intractability of computing story probabilities neither can be used for the hierarchical model. For the non-hierarchical models and the components of the hierarchical model we can still measure the perplexity to see how well the model captures the text distribution.

Recently, (Semeniuta, Severyn, and Gelly 2018) explored various evaluation metrics for language models. Two automated metrics they explored are the Frechet Infersent Distance (FID) and Reverse Language Model Score. The FID metric is based on encoding all of the generated and real documents as vectors and then comparing the distributions of these vectors by approximating them as Gaussian. The FID metric is problematic as it requires a model that can encode the meaning of a document well. In their work, they only focused on sentence level generation and were able to use a model that could create sentence vectors. A second issue with the FID metric is that they did not find it sensitive to word order which indicates it not correlating well with human perception of quality. The Reverse Language Model score is based on the idea that if the generated texts are similar enough to the real texts, than a language model trained on the generated texts should then have a low perplexity on the real texts. The main issue is it involves generating many texts. Due to the slow generation time of our current model, this metric is expensive to compute.

One, method we can use on any model and have initial results for is human evaluation. The way human evaluation was done was that each model being analyzed had about 30 stories generated. Then, a group of native English speakers were found and each had to evaluate ten stories on both global coherence and overall quality on a 5 point Likert Scale.

Models

The models that will be compared are a model from source $\Rightarrow$ outline, outline $\Rightarrow$ story, outline $\Rightarrow$ story + hierarchical attention (h.a.), source $\Rightarrow$ story, hierarchical source $\Rightarrow$ story + h.a..

[1] https://github.com/pytorch/fairseq/tree/master/examples/stories

Table 1: Model Perplexity

Model	Validation Perplexity
source ⇒ outline	45.63
outline ⇒ story	21.08
outline ⇒ story + h.a.	20.49
source ⇒ story	30.96

Evaluation Results

Perplexity Evaluation

There are a couple interesting things from the model perplexity experiments. The biggest one is that the perplexity of source ⇒ story is lower than the perplexity of source ⇒ outline indicating it is easier on average word wise to generate the full article than just the outline. One possible explanation for this is that in generating the outline there are much more abrupt shifts in topic when compared to generating the article and as each word is conditional on the prior words, the topic flowing more smoothly may make it easier to guess the next word.

The other interesting result is that hierarchical attention led to a small improvement in perplexity. In the prior work by (Ling and Rush 2017) involving hierarchical attention, it was mainly motivated by trying to have a more computationally efficient attention mechanism over long sequence and was not found to be helpful otherwise. The improvement however is small enough that it would be necessary to do multiple experiments to tell if it was an improvement due to noise or a genuine improvement.

Human Evaluation

The results can be found in table 2 on the next page. Currently, the number of stories evaluated total is only 70. For the three models, only one pair is significantly different in global coherence. That pair is hierarchical model vs source ⇒ story where the latter is better ($p = 0.005$). The hierarchical model is also significantly different in quality from both of the other two models (for both $p = 0.049$). The worst model is still within a standard deviation evaluation wise of the best model in both metrics indicating how noisy the quality of the stories are and the reviews.

If the mean and standard deviations where to continue to look the same as the sample size grows that would lead to a weird result of the source to story model performing better in human evaluation on global coherence than the outline to story model. Considering the outline is intended to have more information the model should be able to extract more content. It would also be weird given that the outline to story model heavily outperformed the source to story model when it comes to perplexity and would mean perplexity and human evaluation do not correlate well for article generation. Lastly, overall quality and global coherence correlate strongly ($r = 0.74, p < 0.001$).

Failure Cases

Here, we will examine two of the worst generated stories. Worst is defined as being rated as a 1 in both global coherence and overall quality. The first is from the outline ⇒ story and the second was from the hierarchical model with attention. As a note for reasons of space stories were truncated to two paragraphs while outlines were truncated to five lines. unk was used for unknown tokens, num was used for numbers, and newline was used to keep track of paragraph breaks. The text each model was conditioned on will also be shown.

Outline 1: after thurstan 's death , osbert was opposed to one of the candidates for the archbishopric , william fitzherbert , and worked to secure fitzherbert 's deposition and replacement by henry murdac . newline this document lists him as " osbert archdeacon " , which means that he probably held the archdeaconry of richmond . newline originally , osbert had supported elias unk , murdac 's choice for abbot , but then changed his stance and helped with the deposition . newline unk obtained a hearing on the charges at a royal council presided over by king stephen of england at michaelmas in num , but stephen 's subsequent death prevented a resolution . newline he also acted as a steward for hugh de tilly .

Story 1: leofric or unk (unk or unk ; c . num num july num) was the son of king philip iii of scotland . in num , he was crowned king , but after num , a short time spent on england , and his death in num , he was deposed by king stephen i of scotland . the marriage was short - lived ; however , the death is unknown . newline leofric was born about num : num to unk de unk , one of the brothers of the great unk . he took up a good education and was taught by william of unk , who had studied there until num . he also studied a variety of clerical and mathematics , particularly as a tutor , and was taught to study at the royal royal library . in december num , he became archbishop in england . he wrote his last letter to unk , a latin text , and was a member of a group of royal royal clerks . newline

Prompt 2: the portage to san cristobal of a¡dot¿h¡dot¿ is a num literary and philosophical novella by george steiner , in which jewish nazi hunters find a fictional adolf hitler (a¡dot¿h¡dot¿) alive in the amazon jungle thirty years after the end of world war ii .

Outline 2: he began work as a unk at the age of num . newline during world war ii , unk joined the unk unk (royal society) in the united states where he made his world debut in the world . newline unk 's first name was unk unk . newline he was promoted to unk 's unk unk - unk in num . newline unk became the new man 's second - oldest student , unk unk - unk , as a student from unk - unk . newline unk was one of the founding members of the unk unk - unk , the first to be called unk . newline

Story 2: unk unk (unk , " young woman 's sister , the unk , " to the unk ") is unk (unk , " young woman 's sister , the unk) , " girl 's son , and the unk " . the name unk is from the greek unk or " woman 's mother unk " , referring to her marriage to unk . after the marriage , her son , ila , is the only person to be killed in the world . newline during world war ii , il began work as a unk at the age of num . she was the first female woman to be a unk (unk) . after being married , ila became the new man 's second - oldest student ; however , her marriage to unk was interrupted in num . newline

Table 2: Initial Human Evaluation Results

Model	μ-Global Coherence	σ-Global Coherence	μ-Quality	σ-Quality
source ⇒ story	3.36	1.00	2.91	1.07
hierarchical source ⇒ story + h.a.	2.54	0.96	2.26	0.89
outline ⇒ story + h.a.	3.14	1.31	2.90	1.22

The first example does not actually look particularly bad, so that points to unreliable human evaluation scores. Especially not when a score of 2 would still be considered poor and 1 meant very poor. For future human evaluation, less people will be used to try and have the scores be more consistent. It did interestingly not seem to be very related to the outline. While king stephen and scotland are present, a lot of the rest of the details are not very related. Admittingly, that may partly be due to uncommon names likely having been fed to the model as unknown so the many early names in the outline were likely not really read by the model.

The second example looks to be a case where the first component of the hierarchical model did poorly leading to the model struggling to generate a story from that outline. It is not the only example where too many unknown word tokens end up leading to poor stories and while repetition occurs more broadly, unknown tokens have the worst tendency to lead the story to become too repetitive. This issue may also be helped by training the second component of the hierarchical model on not just good outlines, but also sampled outlines so it can learn to not be too reliant on the outline.

More broadly, the main type of error found in generated stories is too much redundancy. Often individual sentences would not be too bad, but a very similar sentence would appear a bit later in the paragraph. Story quality also generally becomes worse near the end of the story.

Best Cases

Here are two of the best stories based on achieving a 5 in global coherence and a 4 in overall quality. The first is from the source ⇒ story model and the second is from the outline ⇒ story model. The text each model was conditioned in is also shown.

Prompt 3: the zrinski battalion (croatian : unk zrinski) was a special forces unit of the croatian national guard (zbor narodne garde zng) and later of the croatian army (hrvatska vojska hv) established in kumrovec on num may num , during the croatian war of independence .

Story 3: the croatian battalion (croatian : unk unk) was a special forces unit of the croatian national guard (zbor narodne garde zng) and later of the croatian army (hrvatska vojska hv) established in unk on num may num , during the croatian war of independence . the battalion was formed during the croatian campaign of the croatian jna alliance and the croatian police , and was responsible for the yugoslav military 's first military operation , which was to be located in the unk , a small village . its main role in operation came from the croatian military 's , as well as other croatian forces . newline in the early days of the war in croatia , the battalion deployed a number of special forces , which were the most heavily defended the yugoslav army . the unit comprised three battalions of four companies . a number of the battalion 's personnel were deployed for the first time during the croatian campaign . during the war , a number of jna operations and training depots had been deployed to croatia in the nums . newline after being handed over on num may num by the croatian national guard , the battalion was reorganised into the croatian national guard (zbor narodne garde zng) and was reorganised as the military 's military 's (unk) battalion in the unk (sao krajina) . the battalion 's operations and operations were transferred to the newly created unk . the battalion 's activities were mainly limited by the majority of the battalion 's personnel , including num officers who participated in the num yugoslav campaign of the croatian war of independence . in june num , the regiment was transferred from its subordinate units , with the remaining members being the croatian - croatian police (unk republike) . the battalion 's activities in croatia were largely subordinated to the government of croatia and a small number of other battalions from other countries . newline

Outline 4: he was previously the head coach at butler university in indianapolis . newline on april num , num , he became the head coach after todd unk left to coach the iowa hawkeyes . newline in july num , he signed a six - year , num million dollar contract to become the head coach of the boston celtics in the nba . newline bradley kent stevens grew up in unk , indiana , where he developed his love for basketball . newline he wore no. num in high school in honor of indiana pacers guard reggie miller . newline he earned three letters in basketball , three in track , and one in baseball during his days at unk . newline the more success you had at lilly , the harder it would be to leave . " newline

Story 4: walter george stevens (may num , num october num , num) was an american major league baseball (mlb) outfielder who played for the boston red sox (num num , num num) and the yankees (num num) . stevens was an outfielder , placekicker and punter , and had played with the boston indians and boston red sox from num to num . he was previously the head coach at butler university in indianapolis . he played num seasons in major league baseball (mlb) for the boston red sox , and in num played on the chicago red sox from num to num . newline stevens played num major league baseball (mlb) all - star team in num , and led the league to five consecutive national champions with a num num conference record , the longest streak in career history . he was also the only player to reach the major division i 's num playoffs . the num and num seasons have taken place in the central division (num num) . newline at the time of the num mlb draft , he was known for the right to handle a hit . he was one to only two players . in num , the red sox traded him in minor league organization with the red sox . the red sox promoted him with the numth overall pick in the num mlb draft , while the red sox traded him to the cleveland indians . newline

In both cases we see that the text conditioned upon was very informative. Doing a better job of preprocessing would likely help in some cases as some of the other prompts end up being truncated too heavily (mainly due to difficulty determining where a sentence ends). The main topic is preserved although a lot of the details can change. For the second story, originally the player was a basketball player, but morphed into a baseball player.

Conclusion

Thus far, we have tried generating text in a hierarchical manner for generating wikipedia articles. Most automated metrics for measuring overall story generation quality are inapplicable due to being computationally intractable. When it comes to human evaluation metrics, the collection of that data is still ongoing, but currently it appears that the hierarchical model is statistically significantly worse than the prompt $\Rightarrow$

For future work, the most important thing is to simplify the model to reduce the heavy computational time associated with it, as this would open up a lot more options that are currently blocked by it would take weeks or months to do. In particular, the current hierarchical model is only trained on true outlines which makes it likely to do poorly if the outline it initially generates is flawed. If the model was faster at generating text, it would also become feasible to evaluate the model using reverse language model score.

Two other possible directions is to make better usage of the outlines. Specifically, for ideal outlines we know that the sentences present in the outline should also be present in the target text. Allowing the model to copy an entire sentence would be beneficial and make it more likely it fully uses the outline. This does come with the downside that a poor sentence in a generated outline being copied would be problematic. That could hopefully be dealt with by training on a mixture of generated and real outlines. A second issue is currently the model is not forced to attend to different parts of the outline. Considering that different parts of the story should correspond to different parts of the outline it should be pushed to examine the entire outline. This could be achieved by adding a term to the loss that pushes it to attend to each sentence at some point. This type of loss has been used successfully before to promote diversity of attention weights in (Kiddon, Zettlemoyer, and Choi 2016).

Acknowledgements

I would like to thank the UCCS NLP lab for many helpful conversations. This material is based upon work supported by the National Science Foundation under Grant No. 1659788. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Allahyari, M.; Pouriyeh, S.; Assefi, M.; Safaei, S.; Trippe, E. D.; Gutierrez, J. B.; and Kochut, K. 2017. Text Summarization Techniques: A Brief Survey. *International Journal of Advanced Computer Science and Applications* (1).

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations* abs/1409.0473.

Dauphin, Y. N.; Fan, A.; Auli, M.; and Grangier, D. 2016. Language Modeling with Gated Convolutional Networks. *International Conference on Machine Learning*.

Fan, A.; Lewis, M.; and Dauphin, Y. 2018. Hierarchical Neural Story Generation. *ArXiv e-prints*.

Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; and Dauphin, Y. N. *International Conference on Machine Learning*.

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative Adversarial Networks. 1–9.

Graves, A. 2012. Sequence Transduction with Recurrent Neural Networks. *International Conference of Machine Learning*.

Kalchbrenner, N.; Elsen, E.; Simonyan, K.; Noury, S.; Casagrande, N.; Lockhart, E.; Stimberg, F.; van den Oord, A.; Dieleman, S.; and Kavukcuoglu, K. 2018. Efficient Neural Audio Synthesis.

Karras, T.; Aila, T.; Laine, S.; and Lehtinen, J. 2017. Progressive Growing of GANs for Improved Quality, Stability, and Variation. *International Conference on Learning Representations* 1–25.

Kiddon, C.; Zettlemoyer, L.; and Choi, Y. 2016. Globally Coherent Text Generation with Neural Checklist Models. {*Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*} 329–339.

Ling, J., and Rush, A. M. 2017. Coarse-to-Fine Attention Models for Document Summarization. *Proceedings of the Workshop on New Frontiers in Summarization* (2015):33–42.

Liu, P. J.; Saleh, M.; Pot, E.; Goodrich, B.; Sepassi, R.; Kaiser, L.; and Shazeer, N. 2018. Generating Wikipedia by Summarizing Long Sequences. 1–18.

Merity, S.; Xiong, C.; Bradbury, J.; and Socher, R. 2016. Pointer Sentinel Mixture Models.

Nenkova, A., and Vanderwende, L. 2005. The impact of frequency on summarization. *Microsoft Research Redmond Washington Tech Rep MSRTR2005101*.

Porter, M. F. 1980. An algorithm for suffix stripping. *Program* 14(3):130–137.

Semeniuta, S.; Severyn, A.; and Gelly, S. 2018. On Accurate Evaluation of GANs for Language Generation.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems (NIPS)* 3104–3112.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention Is All You Need. (Nips).

Zhang, H.; Xu, T.; Li, H.; Zhang, S.; Wang, X.; Huang, X.; and Metaxas, D. 2016. StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks. *International Conference on Computer Vision*.

Impact of Auxiliary Loss Functions on Dialogue Generation Using Mutual Information

Jack St. Clair
Haverford College
370 Lancaster Ave
Haverford, Pennsylvania 19041
Email: jrstclair@haverford.edu

Thomas Conley and **Jugal Kalita**
University of Colorado Colorado Springs
1420 Austin Bluffs Pkwy
Colorado Springs, Colorado 80918
Email: tconley@uccs.edu, jkalita@uccs.edu

Abstract

Dialogue generation involves teaching a program to generate natural conversation. Assuming there are two participants, it requires developing a program that can converse with a human being or another program, and do so coherently and fluently. This paper presents the development of a dialog generating program, popularly called a chatbot, that learns from a corpus of conversations, using a basic sequence to sequence (Seq2Seq) model with a variety of auxiliary loss functions. Auxiliary loss functions are similar to loss functions used during training, but are instead used during generation and do not have to be differentiable. The auxiliary loss functions developed for this chatbot are variants of mutual information between the utterances of one speaker and those of the other, because the objective is to couple these utterances tightly. We demonstrate that using different forms of mutual information leads to developments of chatbots of varying quality. The research shows that when these different chatbots chat with themselves, it is not a sufficient replacement for a human.

Keywords: dialogue generation, Seq2Seq model, maximum mutual information

Introduction

To be able to participate in a dialog, or simply chat with others is a natural human ability. Certain people are good chatters and other people are drawn to them. In other words, such individuals are able to steer the conversation onto topics that are of interest to other participants, follow a topic of conversation for an extended duration, and add details as necessary. There are many books in the market that provide guidelines for interesting and engaging conversation, e.g., (Fine 2005) and (Wadsworth 2017). We focus here on what is called small talk in general parlance. Webster's Dictionary defines small talk as "light or casual conversation: chitchat".[1] The Urban Dictionary defines small talk as "useless and unnecessary conversation attempted to fill the silence in an awkward situation"[2].

Computer scientists have tried to build chatbots for a long time, starting from the initial attempt at building an artificial psycho-therapist called Eliza (Weizenbaum 1966). Because of the nature of psychotherapy, even with its limited abilities, Eliza was able to impress the populace at large, in addition to the research community. Eliza worked simply by pattern matching, and produced inane responses when pattern matching failed to produce a meaningful response. The frame-based architecture for conversation making, introduced by (Bobrow et al. 1977) in the GUS system, ensconced itself as the predominant approach to building dialog agents for several decades. Apple's SIRI and other digital assistants were built using this architecture (Bellegarda 2013; 2014; Jurafsky and Martin 2018). Such speech-based conversational agents used Partially Observable Markov Decision Process (Sondik 1971) in the context of the frame-based architecture, maintaining a system of beliefs and updating them using Bayesian inference. They also used reinforcement learning (Sutton and Barto 1998).

Recently, researchers had started building chatbots by training machine learning programs on transcripts of conversations. Ritter, Cherry, and Dolan (2011) presented a data-driven approach to generating responses to Twitter status posts, using statistical machine translation, treating a status post as a question and the response as its "translation". Of late, researchers have built chatbots using Artificial Neural Networks or Deep Learning. Such research usually uses Seq2Seq models (Cho et al. 2014; Sutskever, Vinyals, and Le 2014). Seq2Seq models have been used by many recent chatbots (Vinyals and Le 2015; Li et al. 2016b; 2016a; Shao et al. 2017; Wu, Martinez, and Klyen 2018). Although the Seq2Seq framework has shown good results in dialogue generation, we believe that the evaluation of the dialogues can be better measured. Most approaches are composed of two Recurrent Neural Networks (RNN) or Long Short-Term Memory (LSTM) units, with the first encoding from words into vectors. The second then decodes these vectors back into words to create the output. The chatbot can thus create a response relevant to the input.

The research presented in this paper aims to examine the role that the use of various auxiliary loss functions plays in the quality of dialog generated when trained on several conversational corpora. Our contribution lies in detailed analysis of the dialogs at various levels of granularity, using a number or metrics. We believe that this is the first time such detailed analysis of automatically generated dialogs has been carried out. We use a simple RNN model for training the conversational agents in small talk since our focus

[1] https://www.merriam-webster.com
[2] https://www.urbandictionary.com/

is more on the auxiliary loss functions. We believe that these loss functions are likely to behave in similar ways with other agent architectures as well.

Problem Statement

Consider a dialog with two participatnts Q and A. Q initiates the conversation with a question, statement or comment q_i, and A follows with a response or a follow-up statement or comment a_i. Thus, a conversation is a sequence of textual elements

$$<< q_1, a_1 >, < q_2, a_2 >, \cdots, < q_i, a_i > \cdots, < q_k, a_k >> . \quad (1)$$

In this paper, we discuss the development of a conversational agent that can be either Q or A or both. We develop this agent by training a machine learning model (Seq2Seq) on a corpus of dialog.

In other words, we give a training sequence T of conversations, ideally between two agents, to a Seq2Seq learner that learns the association between q_i and a_i. This is done by optimizing an Artificial Neural Network (ANN) model, so that given an unseen q, the model can generate an appropriate a based on the learned associations. Once it has been trained, it presumably becomes a "competent" small-talk chatter. During testing, the learned chatbot is given a previously unseen sentence q and it responds with a sentence a. The conversation may continue for a while, and ends when a conversation end indicator is produced.

Conversational ability acquired through training using a Seq2Seq neural model depends on the "loss function" used during training. But, it is also possible to manipulate a trained network's outputs during usage or testing to produce a variety of outputs as seen in this paper. When producing output, the final sentences are generated by searching through a set of candidates, and some candidates may turn out to be more appropriate based on additional processing.

Related Work

Using Seq2Seq models for dialogue generation has become commonplace in recent years. Ritter, Cherry, and Dolan (2011) were the first to use a model used for Statistical Machine Translation (SMT) to generate responses to queries by training on a corpus of query-response pairs. Sordoni et al. (2015) improved Ritter et al.'s work by re-scoring the output of the SMT-based response generation system with a Seq2Seq model that took context into account.

Vinyals and Le (2015) used an RNN-based Seq2Seq model using the cross-entropy auxiliary loss function and a greedy search at the output end. Wen et al. (2015) used LSTMs for joint planning of sentences and surface realization by adding an extra cell to the standard LSTM architecture (Hochreiter and Schmidhuber 1997), and using the cross-entropy loss. They produced sentence variations by sampling from sentence candidates. Li et al. (2016a) used Maximum Mutual Information (MMI) as the objective function to produce diverse, interesting and appropriate responses. This objective function was not used in the training of the network, but to find the best among candidates produced by the model at the output end during generation of responses. Our paper is substantially inspired by this work.

Li et al. (2016b) applied deep reinforcement learning using policy gradient methods to punish sequences that displayed certain unwanted properties of conversation: lack of informativity, incoherence and responding inanely. Lack of informativity was measured in terms of high semantic similarity between consecutive turns of the same agent. Semantic coherence was measured in terms of mutual information, and low values were used to penalize ungrammatical or incoherent responses. The approach also gave negative rewards for inane responses that belong to one of a number of inane responses such as "I don't know".

Su et al. (2018) use a hierarchical multi-layered decoding network to generate complex sentences. The layers are GRU-based (Cho et al. 2014), and each layer generates words associated with a specific Part-Of-Speech (POS) set. In particular, the first layer of the decoder generates nouns and pronouns; the second layer generates verbs, the third layer adjectives and adverbs; and the fourth layer, words belonging to other POSes. They also use a technique called teacher forcing (Williams and Zipser 1989) to train RNNs using the output from the prior step as an input.

In spite of the complex approaches that are being proposed to generate text in the context of question-answering, dialog generation or otherwise, the evaluation of the dialogs have been primarily being in terms of the BLEU (Bilingual Evaluation Understudy) score (Papineni et al. 2002), a metric that was designed for evaluation of SMT. BLEU scores are highly correlated with human judgments of quality for SMT at the corpus level. BLEU computes scores for individual translated sentences by comparing overlaps in terms of n-grams with a set of good quality reference translations. Those scores are then averaged over the whole corpus to reach an estimate of the translation's overall quality. It does not take into account intelligibility or grammatical correctness, and is not a good measure of translations of individual sentences. BLEU score shines as a metric for SMT, however we believe that using BLEU scores alone for evaluating dialogs is limiting. Li et al. (2016b) used two additional computable metrics: the length of the dialog generated, and diversity by calculating the the number of distinct unigrams and bigrams. These two are good additions to the BLEU metric, but we believe that it can be further expanded. Coh-Metrix (Graesser et al. 2004) is a Web-based tool that analyzes texts on over 200 measures of cohesion, language, and readability. We use Coh-Metrix in the evaluation of dialogs in this paper to provide a rich understanding of their quality.

Loss Function

Our training model employs a softmax cross entropy calculation on the logits as provided by TensorFlow. We experimented with hinge and sigmoid cross-entropy functions as an alternative loss measurement during training. Results with these other training functions were inconclusive and since we had no logical reasoning for trying alternate losses without a ground truth for comparison, we leave this area of research for future work.

Instead, we concentrate on the auxiliary loss function needed during sentence generation. These functions operate on partially generated sequenced of states in a beam search. The measure of loss when evaluating these solution states used to find consensus among a number of choices equal to the beam width. We tested extensively using a beam width of 2 using four auxiliary loss functions

We begin with a test using NET loss; by using no loss function at all we predict subsequent characters using only the probabilities predicted by the network.

Another function which uses MMI measurements as loss is shown in equation 2 where S represents the current set of states during sentence generation in the beam search; T represents the set of possible next states. This function is modeled after work conducted by (Li et al. 2016a) and is shown in Equation 2.

$$\hat{T}_{MMI} = \arg\max_{T} \{ \log p(T|S) - \lambda \log p(T) \} \quad (2)$$

We further develop this MMI approach by including Entropy normalization. This approach is inspired by (Estévez et al. 2009) who used Normalized MMI for feature selection. We calculate entropy from predicted network probabilities as shown in equations 3 and 4.

$$H_S = \sum_{t=0}^{|S|} -P(S_t) \times log(P(S_t)) \quad (3)$$

$$H_T = \sum_{t=0}^{|T|} -P(T_t) \times log((T_t) \quad (4)$$

The minimum of these values is used to normalize our MMI value as in Equation 5.

$$\hat{T}_{NORM} = \frac{\hat{T}_{MMI}}{min(H_S, H_T)} \quad (5)$$

Finally we experiment with MMI Entropy normalization where entropy is not calculated but measured directly from the training corpus in terms of character frequencies. Optimizing based on this function should affect the uniqueness of generated sentences.

Architecture

The core of our model is a stack of dense layers comprised of gated recurrent unit (GRUs) cells. We performed tests on a configuration with 3 layers, each divided into 3 blocks, where each block contained 2048 GRUs. This architecture is based on a prior implementation available at on-line[3].

The GRU stack is initialized with the previous state (s_{t-1}) and the current character encoding (x_t) at each time step t in the character sequence. The GRU output (Y_t) and the weights from the final stack layer (W_t) are combined with a bias (b) to produce logits at time t. We define logits as the raw output of the GRU stack which can be normalized and passed to a softmax function to produce probabilities. In this scheme, we update the logits by applying weights and biases from the last GRU layer as shown in Equation 6. The logits are then passed to a loss function for back propagation within the GRU stack. We do not limit or pad the length of the input sequence but perform back propagation through time (BBTT), relying on TensorFlow's default truncated back-propagation capabilities.

$$Logits = (Output \times Weights) + Biases \quad (6)$$

Note that, output sequences $(y_0, ..., y_t)$ are not generated during the training phase where only the logits are used for back-propagation. It is after training, during testing or dialog generation, that the logits are converted to probability using softmax. Finally, probabilities are converted to character sequences using a beam search.

Our beam search employs custom loss functions based on Maximum Mutual Information (MMI) as described in (Li et al. 2016b). We extend this concept to include entropy-normalized MMI, which has been used for feature selection by (Trinh et al. 2018), and is used in this research to select the optimal path in our beam search.

Figure 1 illustrates a single time-step t in sequence processing by our recurrent neural network.

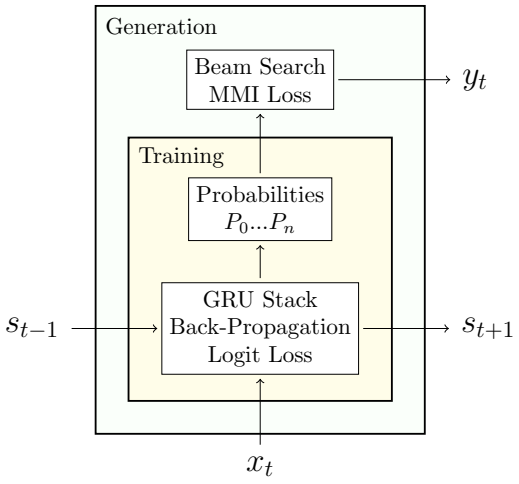


Figure 1: Custom Loss Model

The model accepts a (one-hot) binary vector X and a previous state vector, S, as inputs and produces a state vector, S and a predicted probability distribution vector P_t, for the (one-hot) binary vector Y_t.

Evaluation Metrics

The responses of the chatbot are impacted by the auxiliary loss function used. That is why we vary the loss function to examine how the choices of auxiliary loss functions and datasets change the nature and quality of the generated conversation.

Evaluating the responses automatically is difficult because as of yet, there is no good and agreed-upon way to evaluate a responses created by a chatbot (Liu et al. 2016).

[3]https://github.com/pender/chatbot-rnn

Word-overlap metrics such as BLEU (Papineni et al. 2002), METEOR (Banerjee and Lavie 2005) and ROUGE (Lin 2004) have been commonly used in the past, but we believe that high word overlap between a question and a response does not always make for good conversation, although some overlap shows continuity of topic and thus, coherence. The metrics used here are from a rich discourse evaluation suite called Coh-Metrix (Graesser et al. 2004), as mentioned earlier. Coh-Metrix is an online tool with over 100 different metrics that determine semantic and syntactic features of a given text. These metrics help determine how different loss functions and datasets differ in the output that the model generates. Whether the learning regimen imposed by the loss function and the processing of the output candidates using the MMI criterion help generate syntactically, semantically and discourse-wise effective conversations will be measured by a choice of Coh-Metrix metrics. We use the following metrics in this paper, although all of Coh-Metrix metrics for the conversations are available in Supplementary Material.

- *Mean Words per Sentence:* This metric calculates the number of words in each sentence and then gives the mean of their lengths.
- *Narrativity:* This is a complex metric that measures the narrative or story telling qualities of a text. A narrative text has characters, places, events and chronology of events in it[4]. Narrativity is higher on texts with reoccurring people, places and things. Novels and dramas are examples of narrative text, whereas informational texts are not unless they are written deliberately in a narrative manner. However, every text has some elements of narrativity in it , and Coh-Metrix combines 17 simple metrics and computes a single narrativity number (Graesser, McNamara, and Kulikowich 2011). Making a text narrative makes it easier to follow. Good conversation usually follows a narrative genre. Its value is in the range 0-100.
- *Syntactic Simplicity:* Texts with fewer words and simple sentence structures will receive high scores. Sentences with a lot of words and complex syntax will receive low scores. The value is between 0 and 100.
- *Referential Cohesion:* Texts with words that continue to be mentioned throughout the text receive high referential cohesion scores. This is a simple measure of cohesion that measures the overlap of nouns, pronouns, content words, etc., in adjacent sentences (question-answer pairs, in our case) as well as in the entire text. The value ranges from 0-100. Higher cohesion means the discourse is easier to follow.
- *Sentence Semantic Similarity:* This metric is calculated by latent semantic analysis and does this for all sentences. It looks at the meaning of each sentence and sees if there are any similar themes within adjacent sentences. The value is between 0 and 1. Latent Semantic Analysis (Deerwester et al. 1990; Landauer et al. 2013) can be used to measure semantic overlap among sentences and paragraphs. LSA creates word co-occurrence matrix for words in the document or a smaller unit of the document, performs matrix decomposition and size reduction to obtain representational vectors for individual words. Coh-Metrix computes 8 LSA metrics: LSA cosines between adjacent sentences (in our case, question-answer pairs), sentences in a paragraph, sentences in adjacent paragraphs, their means and standard deviations, etc. We present only one of these in our results.
- *Lexical Diversity:* Lexical diversity calculates the type-token ratio for all words in a text. Type-token ratio (TTR) is defined as "the number of unique words divided by the number of tokens of these words" (Templin 1957). Each unique word is called a type, and each occurrence a token. When TTR is around 1, each word occurs only once, making comprehension difficult since each word needs to be integrated with the conversation. When TTR is lower, words are repeated in the conversation, and it is easier to process. TTR is computed for content words only.
- *Connective Word Occurrence:* This calculates the diversity of words throughout the text, higher lexical diversity means a higher score.
- *Modifiers per Noun Phrase:* This metric calculates the number of modifiers in all noun phrases in the text and then takes the mean of those values.
- *Sentence Syntax Similarity:* This metric takes the syntax trees of all sentences in the text and compares them, calculating the number of similar nodes between the trees.
- *Content Word Frequency:* This metric calculates the average occurrence of content words. Content words include nouns that refer to objects of conversation, lexical or non-auxiliary verbs that describe what can be done with or to these objects, adjectives and adverbs describing qualities of the objects. Although there are only about 150 non-content of function words, they are used heavily in conversation or text. The high presence of content words in a conversation is likely to indicate that the conversation regards something substantial, rather that something meaningless.
- *Word Familiarity:* A piece of text is scored based on the average familiarity of all the words in it. The metric uses familiarity scores assigned to 3488 words in a database (Coltheart 1981) of words that were rated on a 7-point scale by adult raters, 1 being given to words previously unseen, and 7 to words that are seen almost daily. The ratings are multiplied by 100. Sentences with more familiar words can be processed and understood quickly. In a small-talk situation, high familiarity is important, but not so in formal or academic exchanges.
- *Reading Ease:* This metric provides the Flesch reading score for the entire text where a higher score means that the text is easier to read. The formula used is given below (Flesch 1948).

$$r = 206.835 - (1.015 * \overline{sl}) - (84.6 * \overline{spw})$$

where r is *Reading Ease*, $\overline{sl}$ is the *average sentence length*, and $\overline{spw}$ is the *average number of syllables per word*. A Flesch score of 90-100 signifies that the text is

[4]http://wikis.sub.uni-hamburg.de/lhn/index.php/Narrativity

at the 5th grade level, easily understood by a typical 11-year old. A score of 0-30 indicates readability at college graduate level, signifying high difficulty.

Experiments and Results

Our model is trained on data from multiple locations: over 2 GBs of conversations in the comments of Reddit posts, and data from proceedings in the Supreme Court of the United States (SCOTUS). The model is also trained on two other datasets; the Cornell movie corpus(Danescu-Niculescu-Mizil and Lee 2011), a corpus of the scripts from over 600 movies and also on Shakespeare's Romeo and Juliet[5]. These datasets are used for training and then by running the trained model, one can converse with the chatbot.

Metrics Results

Multiple tests were run using Coh-Metrix and the Reddit trained neural network as well as the four distinct auxiliary loss functions NET, MMI, NORM and ENT described in this research. All generated conversations consist of 15 question and answer pairs generated by two different chatbots. From this data, some interesting trends can be observed.

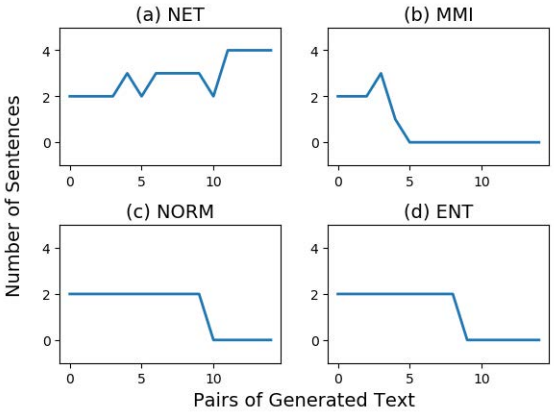


Figure 2: Number of sentences as a measure of sophistication for 4 auxiliary loss functions.

	NET	MMI	NORM	ENT
Mean Words per Sentence	10.070	3.200	1.550	51.389
Narrativity	99.910	98.170	57.140	78.810
Syntactic Simplicity	58.320	41.680	99.930	0.160
Referential Cohesion	90.820	64.800	100	100
Sentence Semantic Similarity	0.363	0.359	0.167	0.624
Lexical Diversity	0.366	0.594	0.333	0.096
Connective Word Occurrence	48.499	0	0	57.297
Modifiers per Noun Phrase	0.408	0.231	0	0.908
Sentence Syntax Similarity	0.114	0.158	0.593	0.040
Content Word Frequency	2.813	4.580	2.358	2.835
Word Familiarity	589.115	572	591.5	583.183
Reading Ease	90.526	100	98.835	63.476

Table 1: Coh-Metrix values for different generated texts

[5]https://github.com/ravexina/shakespeare-plays-dataset-scraper

Figure 3: Coh-Metrix results, the blue dotted line is Narrativity, the orange dashed line is Referential Cohesion and the green dotted and dashed line is Reading Ease.

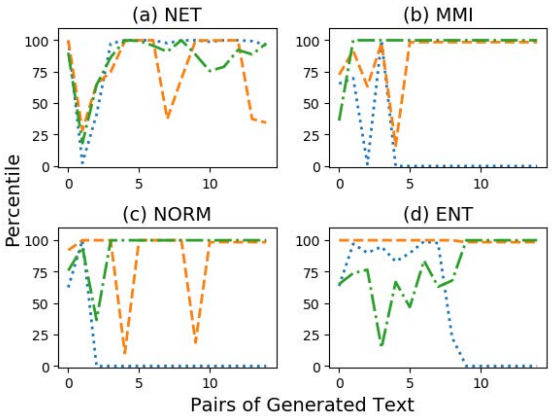


Figure 4: Coh-Metrix results, the blue dotted line is Narrativity, the orange dashed line is Referential Cohesion and the green dotted and dashed line is Reading Ease.

Parts (a) and (b) in Figure 2 show conversations between two chatbots where one is generated using MMI and the other is generated using NET. These figures may suggest that the information the MMI is based on, namely the semantics of the sentence could be harder to decipher using the MMI model instead of the simpler way that the NET model looks at the semantics. Due to the plethora of things that the MMI is calculating, the MMI predictions for characters may be more directed towards simplicity due to the overflow of information the generator is being sent. The simpler NET model may not be able to get as much information but because it gets less information it is able to create a constant dialogue between two chatbots that are simpler in their responses.

In contrast, looking at parts (c) and (d) in Figure 2, the number of sentences in the ENT and NORM generated dialogues are almost identical. The more complex NORM calculation is actually able to barely hold the conversation for longer that the simpler ENT model. NORM is almost the

exact same model as the MMI model, but it includes a normalization of the probabilities that the MMI model does not. It would then make sense that this normalization makes the complex probabilities from the MMI generator into simpler probabilities. The MMI probabilities that are too complex for the chatbots to hold a longer conversation, are made simpler by the normalization. This makes the NORM generator understand more about the conversation and relevant information than the ENT and NET models, but it seems that it is still able to generate at similar rates to them.

However, the graphs alone do not tell the whole story. As can be seen in Table 1, there is a mean of 1.550 words per sentence when generating using the NORM loss function. That means that although sentences are being generated, these generated sentences are without much substance. The fact that the NORM model also has a 0.593 sentence syntax similarity, which is 0.435 more than the next model (MMI with a 0.158 sentence syntax similarity) shows the monotony of the NORM generated dialog. The same conclusion can be taken from the 0.167 value for NORM sentence semantic similarity, which is the lowest of all the models. This is interesting because MMI models were found to improve the output of chatbots in (Li et al. 2016b).

After more tests generating sets of 15 question-response pairs, the reasons for the failure of the NORM generator trained on Reddit data are revealed. On all other datasets, the two chatbots chatting are able to create 15 pairs of dialogue. The most probable reason for this is the huge disparity in the size of the datasets, the data other than Reddit were trained on 10,000 lines of dialogue (other than Romeo and Juliet which is only 840 lines). As can be seen in Figure 3, the Reddit dataset stands out as having low narrativity, but high reading ease. This happens because when it does not generate sentences, there is no story, yet the lack of sentences is easy to read. Figure 4 shows this same fact, only the NET model shows high narrativity and reading ease, it was also the only model to complete the dialogue. It now seems obvious, that the less data that is present and the fewer hoops to jump through in generation, the fewer probabilities the generator has to consider. This could show that more data may only be better with chatbots conversing with humans, because when chatbots are chatting with each other, the dialogue needs to create long enough sentences to keep the flow of information. Thus the less information passed to each chatbot, and the more information they have in common, the easier it will be for them to converse. With human input however, more data is required because the human can ask anything and will expect a coherent response from the chatbot. This is only possible if the chatbot has seen data similar to the input, those chances are of course higher with more data.

Figure 5 shows the results of comparing similarity between network generated responses to randomly selected questions in the corpus. We measure similarity between question and answer with four commonly used metrics. Greater sentence similarity is expected from an intelligent system when compared with a random response from the training corpus. This is shown to be true as generated answers show greater similarity than random answers in three of the four selected metrics.

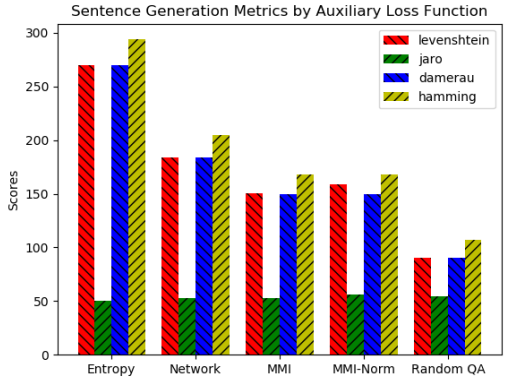


Figure 5: Comparison of auxiliary loss functions by 4 string distance measurements including comparison with randomly selected questions and answers.

Generation Results

Table 2 illustrates a number of things about generated dialogue. First of all, we suppose that the capital letters at the beginning of the sequence are the chatbot's best attempt at writing the name of a supreme court justice as each sentence in the training corpus is labeled thus. The example in this example is most likely "JUSTICE SCALIA:" as represented in the 100 corpus records. While the poorly trained model did not generate the actual name of a justice we admire the attempt considering it is based on only 100 lines of data.

This dialogue in Table 2 is pretty good, but this trained model does not create very coherent dialogue between participants and it also often reaches the character limit for a response which makes the dialogue end abruptly. The chatbot trained on 10,000 lines of the Cornell movie corpus does not give very relevant answers and some of the answers themselves are hard to decipher:

> "You know you've got to go to ask you telling you, I'm afraid of yourself."

Although this is not a grammatical sentence we observe that it does seem to flow somewhat naturally. This begs the question, what is the best kind of dialogue to generate? Is meaning the most important part, or should it flow well?

Conclusion and Future Work

This paper took a Seq2Seq model chatbot and trained it on multiple datasets and used different auxiliary loss functions to see what would generate the best dialogue between two chatbots. Unexpectedly, the MMI and NORM auxiliary loss functions preformed the poorest when trained on the largest dataset. This suggests that MMI based functions get confused with too much data, slowly degrading the dialogue. This concludes that trying to create a chatbot by testing it in a conversation with itself does not reveal whether or not a chatbot is inteligent.

Trained on 100 sentences from SCOTUS
”JMTM SLIE :unt the the aoe the co satiii as io aoe toos con is tuue te the cat ot tian toite on te tcte an ao the ean to ar anert ao the the so tnnt cot aorey sae thot ao the to cait ar te teat oa that ens ae ao thos wo toe ton fe the contits te see tun tee aoe tont cots sh the ian se tee an thar tonttis the uant aae tin an tha oita cntte io on tae se the oan toe tot ton as por aor iie cot ao thl bid the san thee the oo to the tce fortte ior an aar sos aont te talt oh uortee innt aoit cors ite o”
Trained on 10,0000 sentences from SCOTUS
”MR. COLE: I think we’re talking about the district court to review it does, Your Honor. I believe that’s correct, Justice Ginsburg. It’s – it’s in navigation. If you have the distinction between aliens who we collect taxes. They’re – they’re contested, would be able to read the restatement of the landowners – or – or that decision. In that instance, I think that was referred to the issue before this Court that have standing alone and then have set forth in these kinds of prosecutions, when i”

Table 2: Samples of generated text using the NET model showing improved coherence and cohesion with a larger training corpus (10,000) as compared with a very small corpus (100).

Dialogue generation still has a long way to go in creating chatbots that can fluently converse in the same sophisticated natural language of humans. In future work, some easy ways to create better dialogue would be to add a reinforcement learning and attention model to the Seq2Seq model from this research. These models would help the chatbot create relevant responses. Other future additions that could help create a more comprehensive chatbot include some larger and more sophisticated datasets which would allow for more testing on the model and may create responses even more like that of humans than the implementation in this research. Furthermore, advancements in neural networks may bring about faster and more complex neural networks that could also help create even more human like responses. There is also potential for great strides in the ability of evaluation models. With better evaluation models, the dialogue generation model can learn more from itself. Additionally, as mentioned in (Wu, Martinez, and Klyen 2018), there is potential to differentiate the responses of the multiple participants and potentially allow for more than two contributors to the conversation. As of now the best metric available is human evaluation, this was not used in this paper, but it would shed more light on the competency of chatbot dialogue and would be a very useful metric in further work. All of these possible future topics could bring human level computer dialogue generation closer to reality.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 1659788. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Banerjee, S., and Lavie, A. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, 65–72.

Bellegarda, J. R. 2013. Natural language technology in mobile devices: Two grounding frameworks. In *Mobile Speech and Advanced Natural Language Solutions*. Springer. 185–196.

Bellegarda, J. R. 2014. Spoken language understanding for natural interaction: The siri experience. In *Natural Interaction with Robots, Knowbots and Smartphones*. Springer. 3–14.

Bobrow, D. G.; Kaplan, R. M.; Kay, M.; Norman, D. A.; Thompson, H.; and Winograd, T. 1977. Gus, a frame-driven dialog system. *Artificial intelligence* 8(2):155–173.

Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734.

Coltheart, M. 1981. The mrc psycholinguistic database. *The Quarterly Journal of Experimental Psychology* 33(4):497–505.

Danescu-Niculescu-Mizil, C., and Lee, L. 2011. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*.

Deerwester, S.; Dumais, S. T.; Furnas, G. W.; Landauer, T. K.; and Harshman, R. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science* 41(6):391–407.

Estévez, P. A.; Tesmer, M.; Perez, C. A.; and Zurada, J. M. 2009. Normalized mutual information feature selection. *IEEE Transactions on Neural Networks* 20(2):189–201.

Fine, D. 2005. *The Fine Art of Small Talk: How To Start a Conversation, Keep It Going, Build Networking Skills – and Leave a Positive Impression!* Hyperion Books, New York, NY.

Flesch, R. 1948. A new readability yardstick. *Journal of applied psychology* 32(3):221.

Graesser, A. C.; McNamara, D. S.; Louwerse, M. M.; and Cai, Z. 2004. Coh-metrix: Analysis of text on cohesion and language. *Behavior research methods, instruments, & computers* 36(2):193–202.

Graesser, A. C.; McNamara, D. S.; and Kulikowich, J. M. 2011. Coh-metrix: Providing multilevel analyses of text characteristics. *Educational researcher* 40(5):223–234.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Jurafsky, D., and Martin, J. 2018. *Speech & Language Processing (Third edition draft, available at https://web.stanford.edu/ jurafsky/slp3*. Pearson.

Landauer, T. K.; McNamara, D. S.; Dennis, S.; and Kintsch, W. 2013. *Handbook of Latent Semantic Analysis*. Psychology Press.

Li, J.; Galley, M.; Brockett, C.; Gao, J.; and Dolan, B. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 110–119.

Li, J.; Monroe, W.; Ritter, A.; Galley, M.; Gao, J.; and Jurafsky, D. 2016b. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*.

Lin, C.-Y. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Liu, C.-W.; Lowe, R.; Serban, I.; Noseworthy, M.; Charlin, L.; and Pineau, J. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2122–2132.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, 311–318. Association for Computational Linguistics.

Ritter, A.; Cherry, C.; and Dolan, W. B. 2011. Data-driven response generation in social media. In *Proceedings of the conference on empirical methods in natural language processing*, 583–593. Association for Computational Linguistics.

Shao, Y.; Gouws, S.; Britz, D.; Goldie, A.; Strope, B.; and Kurzweil, R. 2017. Generating high-quality and informative conversation responses with sequence-to-sequence models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2210–2219.

Sondik, E. J. 1971. The optimal control of partially observable markov decision processes. *PhD the sis, Stanford University*.

Sordoni, A.; Galley, M.; Auli, M.; Brockett, C.; Ji, Y.; Mitchell, M.; Nie, J.-Y.; Gao, J.; and Dolan, B. 2015. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714*.

Su, S.-Y.; Lo, K.-L.; Yeh, Y. T.; and Chen, Y.-N. 2018. Natural language generation by hierarchical decoding with linguistic patterns. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, 61–66.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, 3104–3112.

Sutton, R. S., and Barto, A. G. 1998. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.

Templin, M. C. 1957. Certain language skills in children; their development and interrelationships.

Trinh, T. H.; Dai, A. M.; Luong, T.; and Le, Q. V. 2018. Learning longer-term dependencies in rnns with auxiliary losses. *CoRR* abs/1803.00144.

Vinyals, O., and Le, Q. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Wadsworth, M. 2017. *How to Make Small Talk: Conversation Starters, Exercises, and Scenarios*. Adams Media, Avon, MA.

Weizenbaum, J. 1966. Elizaa computer program for the study of natural language communication between man and machine. *Communications of the ACM* 9(1):36–45.

Wen, T.-H.; Gasic, M.; Mrksic, N.; Su, P.-H.; Vandyke, D.; and Young, S. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.

Williams, R. J., and Zipser, D. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation* 1(2):270–280.

Wu, X.; Martinez, A.; and Klyen, M. 2018. Dialog generation using multi-turn reasoning neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, 2049–2059.

Engagement Based Mood Prediction

Gia Zhuang
University of Colorado, Colorado Springs
Colorado Springs, Colorado, USA
Email: gzhuang@uccs.edu

Terrance E. Boult
University of Colorado, Colorado Springs
Colorado Springs, Colorado, USA
Email: tboult@vast.uccs.edu

Abstract

To interpret the engagement of an individual using various features such as facial affects or mood has been an important research problem in many fields such as academia and human-robot interactions. While much research has been focused on predicting engagement from a wide variety of features, the use of engagement in predicting a feature such as mood is a relatively unexplored area. In this work, we attempt to verify whether any correlation may exist between mood sub-scale scores and self-reported engagement. We also attempt to verify correlations for engagement with the total mood disturbance scores of the mood-states from our dataset and the change in mood in our dataset from before and after a web session. We use Support Vector Machine (SVM) regression in order to analyze to what degree, if any, these mood scores are related to engagement. Furthermore, we model mood as a sequence learning problem using facial Action Units (AUs) and engagement as input into a Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN) in order to analyze to what degree engagement may aid in predicting mood. We evaluate our work on a rich dataset with many features comprising of 110 subjects who participated in a trauma recovery web-intervention. Our experiments using SVM regression indicate the correlation between mood and engagement is not significant. Furthermore, our results from testing with AUs and engagement as input to our LSTM were inconclusive due to not attaining results as good as or better than the work this builds on: Dhamija, S., and Boult, T. E. 2017a. Automated mood aware engagement prediction. In 2017 Seventh International Conference on, Affective Computing and Intelligent Interaction (ACII), 18. IEEE.

Introduction

Mood significantly influences our daily lives, affecting multiple aspects of our cognition including behavior, perception, and communication. As user-focused services continually grow in demand, so too does the importance of enhancing affective computing for use in various applications. Mood prediction has particular importance in fields such as clinical psychology and trauma recovery, in which predicting the mood-states of an individual will relieve the need to take extensive psychological assessments for a population which may already be burdened. In addition to the psychological benefits, the ability to interpret mood-states can also result in improvements to many of our current technologies which emphasize such implementations as the personalization of interactive features to the user. Being able to predict moods can lead to such implementations in the future as being able to correspond to the mood of the user in the field of advertising (Lee and Hsieh 2009) and enhance other such interactive technology to the user. Predicting moods can also be used to develop on the field of human-robot interaction (Salam et al. 2017) to eventually create robots which are able to adapt themselves to the mood of the people they interact with.

Like mood, research into determining and classifying engagement based on a person's affects has also been well established and is an area of interest for many fields such as academics (Fredricks, Blumenfeld, and Paris 2004; Whitehill et al. 2014), mental health (Dhamija and Boult 2017b), and human-technology interactions (O'Brien and Toms 2008). Various research has been conducted into classifying the engagement level of an individual based on multiple factors such as context and facial affects determined through computer vision (Whitehill et al. 2014; Dhamija and Boult 2017b; Grafsgaard et al. 2013). O'Brien and Toms (2008) has defined engagement to be determined based on the experience of the interaction between the user and the attributes of the system that he or she is involved with. Based on this interaction, engagement can be observed to a relatively high accuracy through facial data (Grafsgaard et al. 2013) and can be augmented through the addition of mood data (Dhamija and Boult 2017a).

Despite these continual advancements in determining engagement, determining the mood of an individual from his or her engagement has yet to be explored in-depth. This work explores this novel concept of predicting mood from engagement and verifies whether any such correlation between engagement and mood exists. We will be aiming to answer these questions:

- If a correlation between engagement and mood exists, to what degree is engagement able to determine the mood-state of an individual?
- Would engagement better predict the change in mood from pre-session to post-session?
- Will adding in other mood-states of different categories from the mood-state we are trying to output aid in pre-

dicting mood from engagement?

- Additionally, if no correlation exists, then would engagement aid in the prediction of mood using AUs?

Background Information

Moods and Mood Assessments: Wilhelm and Schoebi (2007) have defined moods as affective states that influence the cognition, experience, and behavior of an individual. This is differentiated from emotions which are more short-term reaction to stimuli or events. The mood of an individual can typically be determined through the use of one or more assessment tests. Each assessment has its own advantages and may interpret a person's mood in different ways depending on the type of test taken. Wilhelm and Schoebi (2007) describes some assessments including the UWIST Mood Adjective Checklist, the Multidimensional mood questionnaire, the Profile of Mood States (POMS), and the Positive and Negative Affect Schedule (PANAS) when detailing the types which have been used before. A shortened version of the POMS, referred to as the POMS-Short Form (POMS-SF), has also been introduced by Shacham (1983) and works just as well as the POMS for determining the mood state of an individual. As POMS-SF was the assessment used to gather the data with which we will be working with, its results will be the one used with our mood prediction model.

Forecasting Engagement: Prior research has worked on interpreting and forecasting the level of engagement based on facial affects and the context of the user. Grafsgaard et al. (2013) identified the engagement towards and effectiveness of tutoring by identifying facial expressions. Whitehill et al. (2014) also conducted research in academic engagement by defining the engagement level of a group of participants with binary classification and determining the academic effect of high versus low engagement. They utilized three different machine classifiers to determine the engagement of each of the participants: GentleBoost with Box Filter features, SVM with Gabor features, and Multinomial logistic regression from the Computer Expression Recognition Toolbox. Based on their results, they were able to determine the engagement to a relatively high accuracy. Monkaresi et al. (2017) worked on the automatic detection of engagement based upon several components such as the facial expressions and heart rate of the participants. From their research, the Kinetic Face Tracker produced the highest correlation data to the actual engagement in a task when compared against their other independent tests such as monitoring the heart rate. This was accentuated by the participants' task of writing which would mostly obscure the facial features when they were looking down while writing and had limited facial expressiveness.

Mood Prediction in a Different Context: Research has also been conducted on predicting and classifying the mood of an individual. In particular, researchers have proposed various types of emotion and mood aware models for classifying or predicting the mood or emotion of a user based on different methods including recognizing mood from keyboard and mouse interactions (Khan, Brinkman, and Hierons 2013), recognizing mood from smartphone usage (LiKamWa et al. 2013), gathering mood data by extracting daily human behavior patterns and analyzing them (Ma et al. 2012), and determining mood from a segment of a music track (Lu, Liu, and Zhang 2006). Likewise, in the domain of emotion-aware models, similar research has been done on recognizing emotions from speech (Koolagudi and Rao 2012) and recognizing emotions from a combination of cues from audio and visual data (Sebe et al. 2006). Applications of such models span a wide variety of fields including online learning (Mao and Li 2009), expressing affective states through an interface (Lisetti and Nasoz 2002), and e-commerce recommender systems (Shi and Marini 2016). Dhamija and Boult (2017a) attempted to automate the process of detecting moods by using LSTM to assess using automated mood from video segments to aid in predicting engagement. Based on their results, the mood prediction from their visual estimates performed as good as or better than the self-reported total mood disturbance of the participants. LSTM has also been used in other applications such as speech recognition (Graves, Mohamed, and Hinton 2013) and classifying high-resolution images (Krizhevsky, Sutskever, and Hinton 2012).

Engagement Arousal Self-Efficacy (EASE) Dataset

We used the EASE dataset which consisted of a wide variety of data including the self-reported engagement and the POMS-SF mood assessment of a user during a trauma recovery web session on the recovery website http://ease.vast.uccs.edu/. This is the same dataset used by Dhamija and Boult (2017a) and further described by them in their paper. For this work, we will simply provide a brief summary of the dataset and what parts we used from it.

In this dataset, each session consisted of two modules, with the first two sessions being controlled sessions restricted to the Relaxation (RX) and Trigger (TR) modules, and the last session allowing the user to choose whichever of the four remaining modules he or she wants to take. Due to lost data from system errors or data corruption, some participants only have either their first or second sessions recorded, but not both. The participants of this dataset were recruited from various clinical centers mainly focused on health and trauma. This dataset consisted of 110 participants of which, 88 were female, 17 were male, and 5 did not specify. 80% were below the age of 46. To reduce the burden on the participant's who took the mood assessment, the original POMS-SF questionnaire was reduced from 37 questions down to 24.

The POMS-SF mood assessment was taken before and after each session in which their engagement was also recorded through self-reports. Both the mood assessments and the self-reported engagement were rated on a five-point scale. The engagement was reported about 3 times over the course of a session. From the assessments, the scores for each mood were split into positive and negative sentiments, of which only vigor represented the positive sentiment while all the other mood sub-categories were negative sentiments. A Total Mood Disturbance (TMD) score was then calculated

as the difference of the sum of negative sentiments n(x) and positive sentiments p(x):

$$\text{TMD} = \sum_{x \in \text{negative sentiments}} n(x) - \sum_{x \in \text{positive sentiments}} p(x). \quad (1)$$

Due to the some missing data in our dataset and a mismatch of sessions and modules recorded for our mood-state scores and our self-reported engagement, we cut down the number of participants to 69. From these participants, their scores from each of their sessions and modules were used as input features for our prediction model. We used 3 different features from this dataset to predict mood and compare against our results for ground truth. In these features, our input data included the associated action units (AU), which are the calculated facial features from the participants' videos, and their self-reported engagement. Their mood-state scores from their pre-session and post-session POMS-SF assessments were used as ground truth. A delta mood was also calculated from these scores as the change in mood score from pre-session to post-session mood assessments. Not all the mood-states could be scored the same however, with each having a different amount of questions, e.g., tension: 5 questions, depression: 6 questions, anger: 5 questions, fatigue: 2 questions, confusion: 2 questions and vigor: 4 questions. Similarly with the self-reported engagement, while usually each module and session would have 3 self-reported engagement scores, some of the participants either had more than or less than 3 reported scores. Of these scores, we would get rid of the extraneous engagement scores from our data and only use the first 3 scores if more than 3 self-reported engagement scored were available, or we would cut out the participants who only had 1 or 2 self-reported engagement scores.

Experimental Results

We started experimenting with a SVM regression model for each category of mood, regressing the pre-session and post-session mood scores individually rather than appending one to another, to try to find some pattern in our data. Due to engagement being recorded multiple times over a session and module, to make it a simple input for our initial tests, we took the average of the engagement scores and used that as our input. Additionally, we also did a regression for change in scores from pre-session to post-session of each of the moods. We then compared these delta mood score regressions to the results of the regression for the pre-session and post-session moods to analyze which one would have a higher correlation. Furthermore, we expanded on these models to regress along multiple input features in addition to engagement to lower the rate of error in our data. For our tests, we modeled a regression along moods of different categories in addition to engagement to verify whether we could lower the error from our results or not. Furthermore, we also set up a basic LSTM to predict mood based on facial AUs and engagement to determine whether or not we could lower the loss of our model by adding in engagement on top of the AUs to predict mood. For determining the accuracy of our model, we compare our results to the true values from our data and compute a rate of error for our dataset. For computing our error, we calculated the mean squared error (MSE) of our predicted values from the ground truth of the actual values in our dataset. In addition, we used a 2-sided paired t-test to calculate the statistical significance of our data.

SVR Engagement Based Mood Prediction

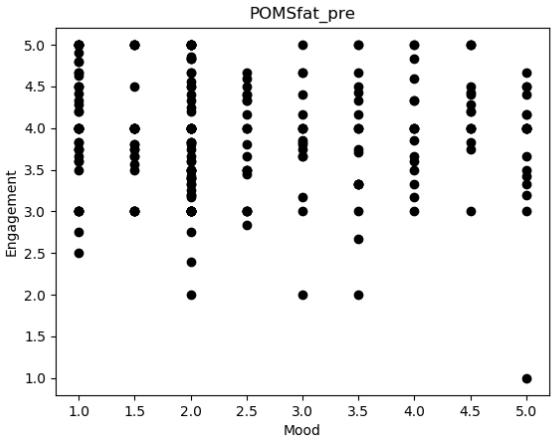


Figure 1: A scatter plot of the test data from the fatigue mood score calculated in the pre-session mood assessment set against engagement. No pattern can seen in this data due to the wide distribution of points across the entire plot. Additionally, from having only been scored by 2 questions, the scatter plot for fatigue shows a noticeable gap from the mood score points only being plotted out every 0.5 points. While not as randomly scattered as fatigue, all the other mood state categories still lack any real pattern in their data distribution,

SVR Regression Results In the process of training our model, we optimized the parameters of our SVM with GridSearchCV. Because of the limited data we have available for training and testing, we used the cross-validation to ensure our model was optimized to predict in a more generalizable nature so as to avoid overfitting. From our initial regression tests on the dataset, we have found little to no correlation for any of our regressed models. This is due to the highly scattered nature of our data when setting mood against engagement, showing only as loosely clustered points over a section of the graph. This caused the MSE to result in a fairly high rate of error for all of our regression models which were used to predict the individual mood-state scores and the delta mood-state scores. Fatigue in particular (as shown in Figure 1) showed a high variation in levels of engagement for each mood score making calculating a regression line much harder. This is likely also because of the different amount of questions per mood-state which resulted in the scoring for each also being different.

SVR Regression Results against Simpler Models Although there was no pattern to be found in the data, the possibility of other, simpler models having better results was

MSE			
Mood	SVR	Ridge	Elastic Net
TMD:	1.57	1.02	0.97
Vigor:	1.06	1.06	1.09
Anger:	0.81	0.81	0.81
Tension:	0.95	1.01	0.99
Fatigue:	1.01	1.2	1.19
Confusion:	1.05	1.11	1.05
Depression:	1.21	1.02	0.94
All Moods Average MSE:	1.09	1.03	1.00

Table 1: Simpler regression models occasionally may perform better compared to more complex regression models such as SVR. We compare SVR to two simpler models, Elastic Net regression and Ridge regression, to evaluate the performance between models in predicting mood from engagement.

also a possibility, so we also tested on Ridge regression and Elastic Net regression to verify if maybe a simpler model would train better in comparison to SVR. As shown by Table 1, the results of simpler regression models were not much better either. While using Elastic Net regression, the line of regression was simply a straight line through the middle of the data which demonstrates the lack of pattern in the overall data. As can be shown from Table 1, all of the moods had a high MSE, with the simpler regression models neither performing significantly better than or worse than the SVR. The 2-sided paired t-test also showed a lack of any significance in the performance of each of the models as demonstrated by the p values being generally in a range from 0.2 to 0.8. Since no real pattern existed inside the data, none of the regressions performed much better in comparison to one another, showing in how the average rate of error was about equal for all of the regression models. When calculating the delta mood-state scores as the change in mood from before a session to after a session, a slightly higher correlation could be seen with the engagement during said session, although the lack of any pattern still applies to the delta moods as it did to the pre-session and post-session moods.

Multiple Regression Results By adding in one more input feature into the SVM regression, we were able to reduce the mean squared error of our data overall. From our initial regression models in which we only regressed a single mood against the average engagement over a web session, we found our MSE to be generally high in the range of 0.9 to 1.8, illustrating the lack of suitability for prediction in those models. By adding in additional features such as moods of different categories however, the MSE of some of the mood prediction models decreased down to around the range of 0.4 to 0.6 by adding in moods of other categories as input features.

Due to some mood scores lacking any significant change pre-session to post-session, using the pre-mood of the same category as additional input to predict the post-mood would not have resulted in much of a problem. Therefore, for the input, we compared each and every one of the mood-states from separate categories in order to see if any mood-states from separate categories could lower the rate of error in predicting any single mood. Generally, the error did appear to lessen with an additional mood input; however, some moods were obviously very poor indicators of each other and actually made the MSE higher, even if just by a small margin. On the other hand, a not statistically significant lowered MSE could also be found by moods which seemed like they would affect each other to a larger degree. Anger and tension appeared to have a higher correlation compared to the other moods which anger was regressed with alongside engagement. In the pre-session, POMS anger regressed against POMS tension and engagement resulted in a MSE of 0.44, although it was not statistically significant ($p = 0.36$). Similarly during the post-session, POMS anger regressed against POMS tension and engagement resulted in a MSE of 0.40, although this was also not statistically significant ($p = 0.278$). On the other hand, post-session POMS anger regressed against pre-session POMS tension and engagement resulted in a MSE of 1.00, with these results actually statistically significant ($p = 0.026$). The possibility exists that some of these correlations, or lack of correlations, occur because of the biases contained in the data (referring to the different number of questions for scoring per mood).

For the most part however, although these results were not statistically significant, moods from the same assessment (either the pre-session mood assessment or the post-session mood assessment) had an apparently lower MSE as compared to when predicting the mood from the other assessment. As moods from assessments taken at separate times would have less relevance to each other as compared to moods taken from the same assessment, theoretically, these results are to be expected. A few exceptions did exist however, although the difference was not too big. E.g., post-session POMS depression regressed against post-session POMS confusion and engagement resulted in a MSE of 0.73, although these results were not statistically significant ($p = 0.665$). Then post-session POMS depression regressed against pre-session POMS confusion and engagement resulted in a MSE of 0.69, although these results were also not statistically significant ($p = 0.495$).

Interestingly enough, although the correlation between any one mood and the rest were generally not good across every one of the mood-state categories (with no improvement or barely any when predicting with an additional mood minus the exception of a few as mentioned before), using the TMD based on the scores of a participant's mood assessment resulted in a lowered rate of error across every single mood category. Except for vigor, every other mood category regressed with TMD as an additional input had a relatively lowered error rate, dropping to a MSE as low as in the range of 0.3 to 0.4 for the moods from the associated assessment (Although the results from the paired t-test indicated that there was no statistical significance for these values with p usually being in a range from 0.1 to 0.8). As the TMD was calculated off of all the separate mood categories with only vigor as a positive sentiment being subtracted from the rest, these results appear to theoretically have some significance.

Based on these results, although most of them were not statistically significant, a good possibility exists that predicting a mood based on the moods of other categories (or at least based on the TMD score) in addition to engagement, depending on the mood, could have a decently high correlation with each other.

LSTM Mood Prediction

While attempting to predict mood from engagement and with other categories of mood was interesting, more evidence exists to support the prediction of mood with facial AUs (Dhamija and Boult 2017a). In order to predict mood with AUs, we will need to model mood as a sequence learning problem. LSTMs are able to better handle longer sequences with long-term dependencies of AUs as opposed to SVMs. For this reason, a sequence learning model such as LSTM is better for adding in the sequences of associated AUs from the videos of each session and module in our dataset. So in this section of our work, we turn to building a LSTM which will be used to predict mood with facial AUs in addition to engagement. We utilize the automated mood prediction method from facial AUs done by Dhamija and Boult (2017a) as a baseline for building our LSTM model to use in our training. Their implementation was based on Tensorflow, which was in turn based on Zaremba, Sutskever, and Vinyals (2014) and Graves (2013). For our work, we use Keras as a base to build our mood prediction LSTM model. For the training and testing of our model, we used the precomputed AUs provided by Dhamija. These AUs were calculated using the work on OpenFace proposed by Baltrušaitis, Robinson, and Morency(2016). OpenFace is an open-source tool for detecting features such as gaze, head pose, and AUs. Our work uses the same AUs as Dhamija and Boult (2017a), e.g. Inner Brow Raiser, Outer Brow Raiser, Brow Lowerer (intensity), Upper Lid Raiser, Cheek Raiser, Nose Wrinkler, Upper Lip Raiser, Lip Corner Puller (intensity), Dimpler, Lip Corner Depressor (intensity), Chin Raiser, Lip Stretched, Lips Part, Jaw Drop, Brow Lowerer (presence), Lip Corner Puller (presence), Lip Corner Depressor (presence), Lip Tightner, Lip Suck, and Blink. In total, 20 AUs were tracked across the videos in our dataset. The total number of frames the AUs were tracked over was 6126581 from all of the videos of the sessions and modules we used from our data. We used as input segments of 30 seconds from each of our videos with the AUs having been calculated at 30 frames per second for a total of 900 frames per session and module.

Predicting Mood using AUs and Engagement Our results for predicting mood using AUs was inconclusive as when predicting with both individual AUs and the entire set of AUs, neither gave usable results. Due to our model mainly predicting around the mean of our dataset, our results produced a MSE of over 1, with some results also exceeding a MSE of 2. These high rates of error can be attributed to our model not training well however. Better training will be necessary to actually predict mood based on AUs as was similarly done by Dhamija and Boult (2017a). After conducting proper optimization of our model and cross-validating our data to avoid overfitting, the rate of error should become smaller and the model should fit better during training. In addition, as we had overall poor results, we could not determine whether adding in engagement as an input on top of the facial AUs actually aided in predicting mood or not. Although our work has not produced substantial results for predicting mood from engagement with facial AUs, the results of automated mood predictors done by Dhamija and Boult (2017a) does show a strong indication of being able to predict moods based on AUs at the very minimum. As our results could not match up to theirs, more research will need to be done in order to verify whether predicting mood from AUs with engagement will actually produce better or worse results.

Future Work

Future work in engagement based mood prediction may focus on whether using more time segments or less as input may enhance the prediction capabilities of the LSTM. In our work, we focused on predicting mood using 30 second time segments from videos to predict mood with the aid of engagement. As our work did not produce useful results from using facial AUs as input to predict mood, first optimizing our LSTM model would be necessary to perform as good as or better than Dhamija and Boult (2017a). After finally having a working model, shortening the time segments down to 15 seconds may still be enough to provide results or it is also possible extending the time segment to 45 seconds may give better results in predicting mood. In addition, using the trigger and relaxation modules to predict moods in context would also be a good test to identify how much context may affect mood scores or change in mood scores. Similarly, detecting which AUs correlate to which mood-state scores the highest may also be a good test to identify whether certain AU features, such as inner brow raiser or outer brow raiser, may affect certain mood-states more than others. In addition, the statistical significance of most of our results were not significant enough to decide whether our data was good or not. As a consequence, we could only, at most, form conjectures based on the results of our data. In future work, further testing of the significance of our results can be conducted using bonferroni corrections.

Conclusion

Most prior work has focused on identifying and classifying engagement using computer vision, occasionally with the use of the acquired mood data of the individual to augment the results. Mood on the other hand has been predicted using visual, audio, or other such data, but not engagement. We work on a novel problem which has yet to be explored by anyone to the best of our knowledge by attempting to predict mood from engagement. The ability to predict moods from engagement has many possible applications in a wide variety of fields such as robot-human interactions, clinical psychology and advertising. In correspondence with this work, trauma recovery could also be improved to minimize the need of taking mood assessments by just predicting their mood from some other data instead. Based on our current

results, regression for a mood with just engagement as an input does not produce any patterns which can be used for predicting mood. On the other hand, adding in more input features, depending on what input feature is used, can either positively influence, or negatively influence the rate of error. From attempting to regress mood along moods of different categories, while some moods showed a decrease in the MSE, others actually increased it. Only the calculated TMD score of the mood set actually decreased the error for all of the moods, although some moods decreased from the TMD input more than others. Due to a flaw in our model, our work using a LSTM for predicting mood from facial AUs was unable to produce results comparable to Dhamija and Boult (2017a). As we could not obtain results as good as or better than them, we were unable to verify at the current time whether or not adding in engagement would aid the model during training or not.

Acknowledgments

We would first like to acknowledge UCCS and the entire VAST lab for their support and all the useful conversations we have had on this research. Additionally, we would like to acknowledge Svati Dhamija for providing us with her data and continual support in conducting this research. This material is based upon work supported by the National Science Foundation under Grant No. 1659788. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Baltrušaitis, T.; Robinson, P.; and Morency, L.-P. 2016. Openface: an open source facial behavior analysis toolkit. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, 1–10. IEEE.

Dhamija, S., and Boult, T. E. 2017a. Automated mood-aware engagement prediction. In *2017 Seventh International Conference on, Affective Computing and Intelligent Interaction (ACII)*, 1–8. IEEE.

Dhamija, S., and Boult, T. E. 2017b. Exploring contextual engagement for trauma recovery. In *2017 IEEE Conference on, Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2267–2277. IEEE.

Fredricks, J. A.; Blumenfeld, P. C.; and Paris, A. H. 2004. School engagement: Potential of the concept, state of the evidence. *Review of Educational Research* 74(1):59–109.

Grafsgaard, J.; Wiggins, J. B.; Boyer, K. E.; Wiebe, E. N.; and Lester, J. 2013. Automatically recognizing facial expression: Predicting engagement and frustration. In *Educational Data Mining 2013*, 43–50.

Graves, A.; Mohamed, A.-r.; and Hinton, G. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on, Acoustics, Speech and Signal Processing (ICASSP)*, 6645–6649. IEEE.

Graves, A. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Khan, I. A.; Brinkman, W.-P.; and Hierons, R. 2013. Towards estimating computer users mood from interaction behaviour with keyboard and mouse. *Frontiers of Computer Science* 7(6):943–954.

Koolagudi, S. G., and Rao, K. S. 2012. Emotion recognition from speech: a review. *International journal of speech technology* 15(2):99–117.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 1097–1105.

Lee, C.-C., and Hsieh, M.-C. 2009. The influence of mobile self-efficacy on attitude towards mobile advertising. In *New Trends in Information and Service Science, 2009. NISS'09. International Conference on*, 1231–1236. IEEE.

LiKamWa, R.; Liu, Y.; Lane, N. D.; and Zhong, L. 2013. Moodscope: Building a mood sensor from smartphone usage patterns. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, 389–402. ACM.

Lisetti, C. L., and Nasoz, F. 2002. Maui: A multimodal affective user interface. In *Proceedings of the Tenth ACM International Conference on Multimedia*, MULTIMEDIA '02, 161–170. New York, NY, USA: ACM.

Lu, L.; Liu, D.; and Zhang, H.-J. 2006. Automatic mood detection and tracking of music audio signals. *IEEE Transactions on audio, speech, and language processing* 14(1):5–18.

Ma, Y.; Xu, B.; Bai, Y.; Sun, G.; and Zhu, R. 2012. Daily mood assessment based on mobile phone sensing. In *2012 Ninth International Conference on, Wearable and implantable Body Sensor Networks (BSN)*, 142–147. IEEE.

Mao, X., and Li, Z. 2009. Implementing emotion-based user-aware e-learning. In *CHI'09 Extended Abstracts on Human Factors in Computing Systems*, 3787–3792. ACM.

Monkaresi, H.; Bosch, N.; Calvo, R. A.; and D'Mello, S. K. 2017. Automated detection of engagement using video-based estimation of facial expressions and heart rate. *IEEE Transactions on Affective Computing* 8(1):15–28.

O'Brien, H. L., and Toms, E. G. 2008. What is user engagement? a conceptual framework for defining user engagement with technology. *Journal of the Association for Information Science and Technology* 59(6):938–955.

Salam, H.; Celiktutan, O.; Hupont, I.; Gunes, H.; and Chetouani, M. 2017. Fully automatic analysis of engagement and its relationship to personality in human-robot interactions. *IEEE Access* 5:705–721.

Sebe, N.; Cohen, I.; Gevers, T.; and Huang, T. S. 2006. Emotion recognition based on joint visual and audio cues. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 1, 1136–1139. IEEE.

Shacham, S. 1983. A shortened version of the profile of mood states. *Journal of Personality Assessment* 47(3):305–306.

Shi, F., and Marini, J.-L. 2016. Can e-commerce recommender systems be more popular with online shoppers if they are mood-aware? In *WEBIST (2)*, 173–180.

Whitehill, J.; Serpell, Z.; Lin, Y.-C.; Foster, A.; and Movellan, J. R. 2014. The faces of engagement: Automatic recognition of student engagement from facial expressions. *IEEE Transactions on Affective Computing* 5(1):86–98.

Wilhelm, P., and Schoebi, D. 2007. Assessing mood in daily life: Structural validity, sensitivity to change, and reliability of a short-scale to measure three basic dimensions of mood. *European Journal of Psychological Assessment* 23(4):258.

Zaremba, W.; Sutskever, I.; and Vinyals, O. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

PixelMRF: A Convolutional Markov Random Field for Image Generation

Kayleigh Migdol
Department of Statistics and Data Science
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
Email: kmigdol@andrew.cmu.edu

Jonathan Ventura
Department of Computer Science
University of Colorado, Colorado Springs
Colorado Springs, Colorado, USA
Email: jventura@uccs.edu

Abstract

Generative models have been rising in popularity within the past several years. They have been used for a wide range of applications, especially in art (Gatys, Ecker, and Bethge 2015; Elgammal et al. 2017). Many of these models, however, are directed. This creates restrains on how the model can be used along with the potential accuracy. One of these models is the PixelCNN. While it and its successors are state-of-the-art, they fail to look at the whole picture. Meanwhile, many of the commonly used undirected models, such as Restricted Boltzmann Machines, are limited in their flexibility due to the need for discretization. We propose a undirected model, the PixelMRF, in response to these concerns.

Introduction

Over the past several years, generative models have become more and more popular. With the introduction of structures such as GANs, VAEs and the PixelCNN, there have been more and more options for the approach. That being said, many of the models being used are directed. In the PixelCNN, for example, each pixel is dependent only on those before it. However, this potentially ignores the entire image! While this is an elegant solution and makes it relatively easy to sample, it is creating restraints on its potential accuracy. In addition, it forces known pixels to be before those being sampled. This keeps the models from being used in applications that require more flexibility such as inpainting.

Meanwhile, past attempts at undirected models have been limited in scope. The deep markov random field, wa recurrent neural network, is too focused on local structure, needing additional help to create coherent samples (Wu, Lin, and Tang 2016). Restricted Boltzmann Machines, on the other hand, force discretization of the input and output. Because of this, it is less robust than those that use a continuous distribution. These issues lead to our proposal of the PixelMRF, a convolutional markov random field.
Our contributions are as follows:

1. We propose our new model, the PixelMRF, which is comprised of two PixelCNNs working in parallel: one looking at prior pixels and one looking at succeeding ones.
2. We discuss some of the roadblocks that come up when working with a markov random field. These include training, comparison against other networks, and sample generation. We discuss our solutions to these issues as well.
   (a) We show the accuracy of annealed importance sampling, a technique to estimate the partition function, by testing it on a toy network.
3. We demonstrate our network's ability to produce global structure through generating samples from the MNIST and Frey datasets.

PixelCNN

The auto regressive network, PixelCNN, is a convolutional neural network (CNN) based off of the idea of representing an image x's probability as

$$p(x) = \prod_{i=1}^{n^2} p(x_i|x_1...x_{i-1}) \tag{1}$$

where the image is $n \times n$. Because of this view, each pixel is only dependent on the ones before it which prevents the entire context from being considered. Given an image, the network outputs a discrete probability distribution for the red, green, and blue values of a specific pixel. The network itself is a series of convolutional layers with residual connections that include a mask to block the current pixel from seeing the following pixels (Oord, Kalchbrenner, and Kavukcuoglu 2016).

Improvements Previously Made

Since the PixelCNN was released, there has been two major improvements. The first one being the Gated PixelCNN, which improved the architecture within the layers (van den Oord et al. 2016). The second is PixelCNN++, which made various smaller improvements to the Gated PixelCNN (Salimans et al. 2016). One of the most notable improvements is the use of using a logistic distribution vs the softmax distribution used in the original PixelCNNs. We will be basing the PixelMRF off of the Gated PixelCNN except we are using the logistic distribution of the PixelCNN++. We are using a modified version of this for grey scale.

Drawbacks

While the PixelCNN and its successors are very powerful tools, they has one major drawback. Because in the probability model each pixel is only dependent on those before it,

the pixels after it are ignored. This prohibits it from taking into consideration the full context of the image. In addition, known pixels must be before the pixels whose probability distribution will be calculated. This restricts how the network can be trained and utilized. One example of this is in inpainting. The PixelCNN is unable to properly inpaint as it only takes the prior pixels into account.

PixelMRF

We wish to solve this issue and take the entire picture into account. That is, we wish to have the unnormalized probability represented as:

$$\tilde{p}(x) = \prod_{i=1}^{n^2} p(x_i|x_{-i}) \tag{2}$$

where x_{-i} is the set of all pixels in an image x except x_i.

Implementation

We call our network PixelMRF due to the fact that the probabilistic model is a markov random field. We are using two variations of the gated PixelCNN simultaneously with the first one being identical to the original, taking only the previous pixels into account. The second one is the opposite, taking only the following pixels into account. This is done by flipping the input, first left-right then up-down, feeding it into the network, then flipping it back. We then combine them together at the end through concatination and then feeding them into a layer of 1x1 convolutions. This is shown in Figure 1. We have also attempted using the PixelCNN++ instead of the gated PixelCNN. However, the downsampling included in the PixelCNN++ led to incoherent sample generation.

Drawbacks

The Partition Function The issue with our model, however, is that, in order to calculate the normalized probability

$$p(x) = \frac{1}{Z}\tilde{p}(x), \tag{3}$$

where $\tilde{p}(x)$ is the unnormalized probability, we need Z, the partition function. This is near impossible for us to do due to the computational resources required. For us, this creates a lot of challenges. First, we are unable to train using the log likelihood. As our true goal is to optimize this value, it would be best if we could train using the function. Another issue with this is that log likelihood is the standard metric of generative models. We are unable to do a proper comparison of the PixelMRF against other models without it. As the models output the probability density that is then sampled using some technique, it would be improper of us to compare samples as this is partially determined on the technique used for sampling. Thus, we need to compare log likelihood directly.

Sampling Another issue we have with using a markov random field is that it makes sampling quite complicated. With the PixelCNN, sampling is quite simple as it is a directed model and so it is possible to use ancestral sampling. On the other hand, with the PixelMRF, because all pixels are dependent on all the other pixels both before and after, there is a need to cycle through.

Proposed Solutions

Psuedolikelihood

Optimally, we would use the log-likelihood for training:

$$\sum_{i=1}^{n^2} \log \tilde{p}(x_i|x_{-i}) - logZ \tag{4}$$

However, Z, the partition function, is something that we are unable to calculate. Therefore, we must find an alternate solution. Thus, for our training, we are using the pseudolikelihood (Besag 1975):

$$\sum_{i=1}^{n^2} \log \tilde{p}(x_i|x_{-i}) \tag{5}$$

The idea behind using pseudolikelihood is that the likelihood distribution is still the same despite the partition being ignored when it comes to the locations of optima. As the pseudolikelihood and log likelihood have about the same optima, while this won't be as accurate as using the log likelihood, it serves as a good approximation. In this case, you are still able to find good weights for the log-likelihood. An alternative to this is using contrastive divergence (Hinton 2002). This is something to look into in the future.

Annealed Importance Sampling

Algorithm 1 AIS on Toy Model

for $k = 1...K$ **do**:
  Sample $x_{\eta_1}^{(k)} \sim p_0(x)$
  Sample $x_{\eta_2}^{(k)} \sim T_{\eta_1}(x_{\eta_2}^{(k)}|x_{\eta_1}^{(k)})$
  ...
  Sample $x_{\eta_n}^{(k)} \sim T_{\eta_{n-1}}(x_{\eta_n}^{(k)}|x_{\eta_{n-1}}^{(k)})$
end for
$w^{(k)} = \frac{\tilde{p}_{\eta_1}(x_{\nu_1}^{(k)})}{\tilde{p}_0(x_{\nu_1}^{(k)})} \frac{\tilde{p}_{\eta_2}(x_{\nu_2}^{(k)})}{\tilde{p}_{\eta_1}(x_{\nu_2}^{(k)})} ... \frac{\tilde{p}_1(x_{\nu_n}^{(k)})}{\tilde{p}_{\eta_{n-1}}(x_{\nu_n}^{(k)})}$
$\frac{Z_1}{Z_0} \approx \frac{1}{K} \sum_{k=1}^{K} w^{(k)}$

For our testing, we need to be able to calculate an approximate of the log likelihood so that we are able to compare against other models. For this, we need an approximation of Z. In order to do that, we can use a technique called Annealed Importance Sampling, or AIS (Neal 2001). The goal of the algorithm is to calculate Z_1/Z_0. If Z_0 comes from a simple enough distribution that you are able to calculate Z_0 by hand, you are able to find Z_1. The idea is to progress through a series of intermediate distributions that are closely related in order to approximate the ratio. The algorithm is described in Algorithm 1.

For our starting distribution, we are having the network with the weights (but not biases) set to zero. Because of this, the output is completely independent of the input, meaning

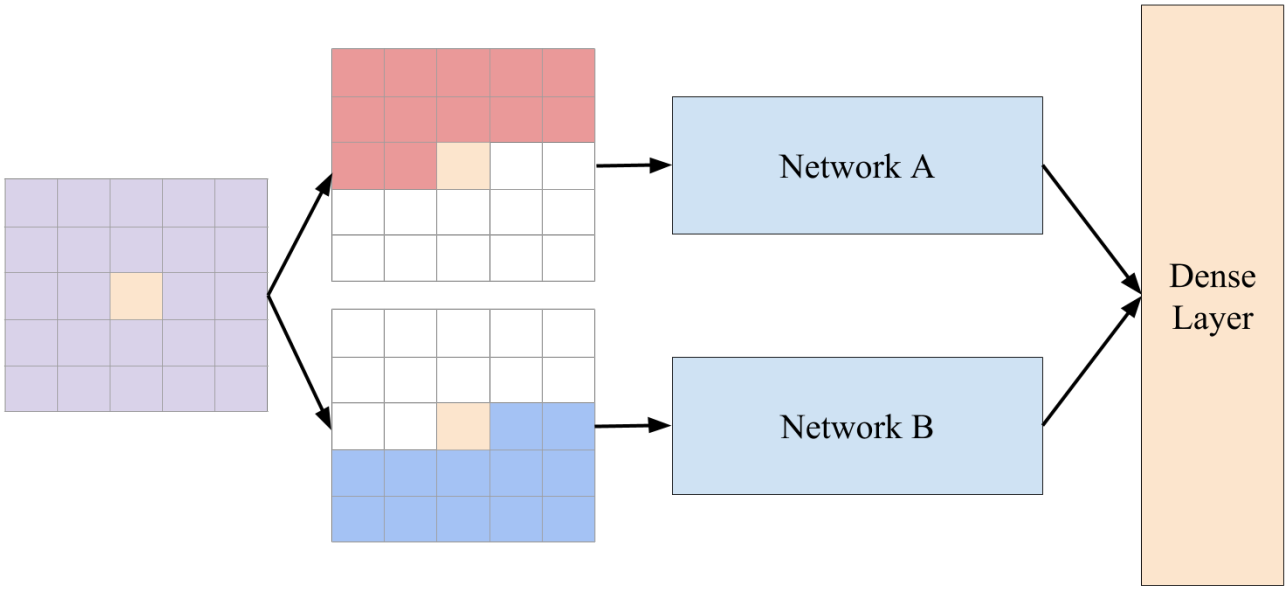


Figure 1: In this case, the current pixel is the one in the center. Network A considers only pixels before and Network B only considers pixels after. Combined, they take all of the pixels (besides the current one) into consideration

that there are no relationships between the pixels. Thus it becomes a Bayesian network (vs a markov random field). As it is a directed network, the partition, Z_0, is one.

The goal of the intermediate distributions, $\tilde{p}_{\eta_1}$ through $\tilde{p}_{\eta_n}$, is to serve as a bridge between p_0 and p_1. For these, we are using an arithmetic mean:

$$p_{\eta_n} = \eta_j * p_1 + (1 - \eta_j) * p_0. \quad (6)$$

For our transitions functions, we are using gibbs sampling with the probability distribution of T_{η_j} being p_{η_j}. Due to time restraints, we were unable to use the AIS on the PixelMRF. However, we were able to test it on a small, toy network as described below.

Gibbs Sampling

In order to sample from the PixelMRF, we have used gibbs sampling. In this, we go through and sample each pixel individually, updating the image and thus the input as we go. This repeats several times.

Experiments

As we are unable to calculate the log-likelihood without the AIS, we are unable to do a comparison of it against other models. Still, we are able to show the accuracy of our version of AIS on a toy model. We are also able to look at the model's ability to generate samples.

Annealed Importance Sampling

While we did not have time to run the AIS on the PixelMRF, we were able to test it on a smaller toy model. In this case, we initialized a random single-layer feed forward network with binary inputs and a softmax distribution for the output. While the model is much more simple than the PixelMRF, it is still representing a markov random field. By using a small number of binary units for the input, we are able to calculate the true partition by hand. Through this, we are able to show that our intermediate distributions and transitions are valid and accurate. In this, we used an increasing number of hidden nodes (25, 50, 500). In all cases, we had 500 runs, 15,000 steps, and five binary inputs. Our results are in Table 1.

Sample Generation

We were able to generate samples using gibbs sampling, starting from random noise.

Frey For the Frey dataset, our PixelCNNs had 16 filters and two Resnet layers. Our samples are in Figure 2. As the Frey dataset doesn't have much variation, it was quite easy for the network to learn the dataset and produce good samples.

MNIST When generating samples from the MNIST dataset, the PixelCNNs had 32 filters along with five Resnet layers. Our results are in Figure 3. The greater diversity along with the introduction of class labels led to the samples being not quite as nice as Frey's. However, the network is still able to establish a global structure when generating the samples.

Discussion

While our results are not quite state-of-the-art, our primary contribution comes from the development of this new model, along with the ability to sample from it. Although there have

Number of Hidden Units	True Z	Estimated Z	True Average NLL	Est. Average NLL	Percent Error
25	-0.2017	-0.2520	10.7860	10.7357	0.4663 %
50	-0.4385	-0.7674	6.6094	6.2805	4.931 %
500	0.2786	0.2484	29.7789	29.7487	0.1014 %

Table 1: Results of AIS test on the toy network. It is a one layer feed forward network with five binary inputs and a softmax output. NLL stands for negative log likelihood.

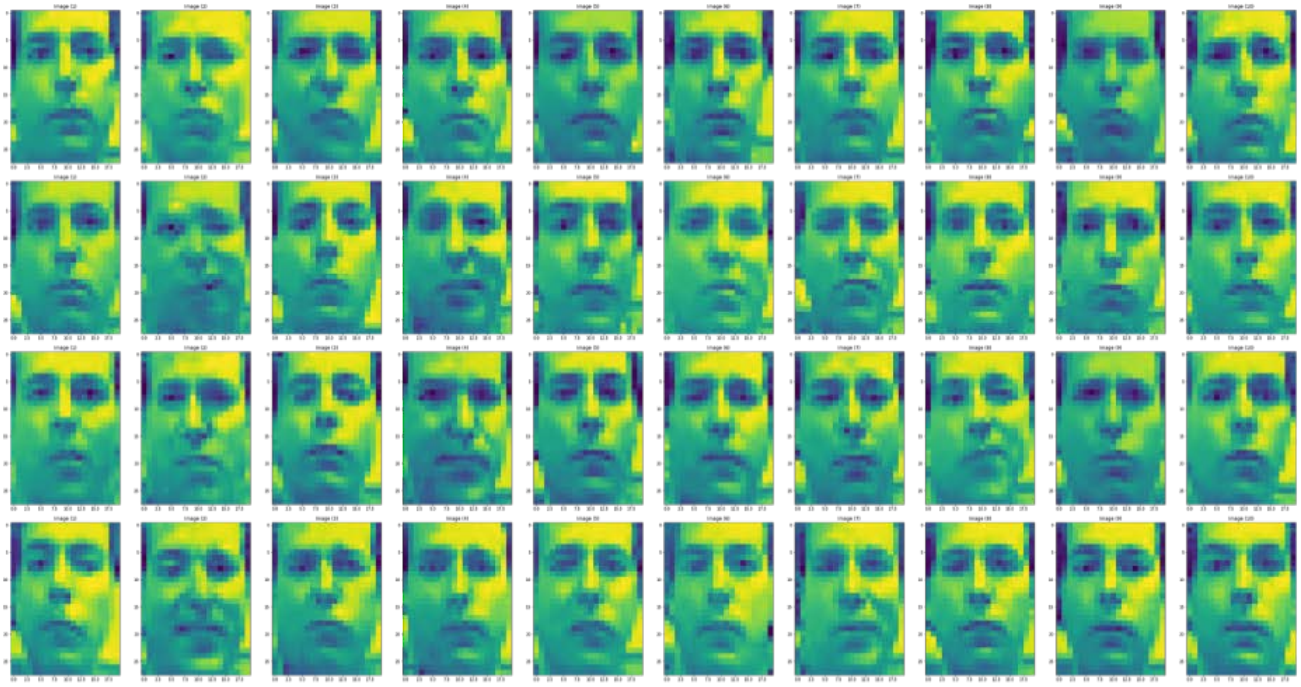


Figure 2: Samples generated by the PixelMRF after being trained on the Frey dataset

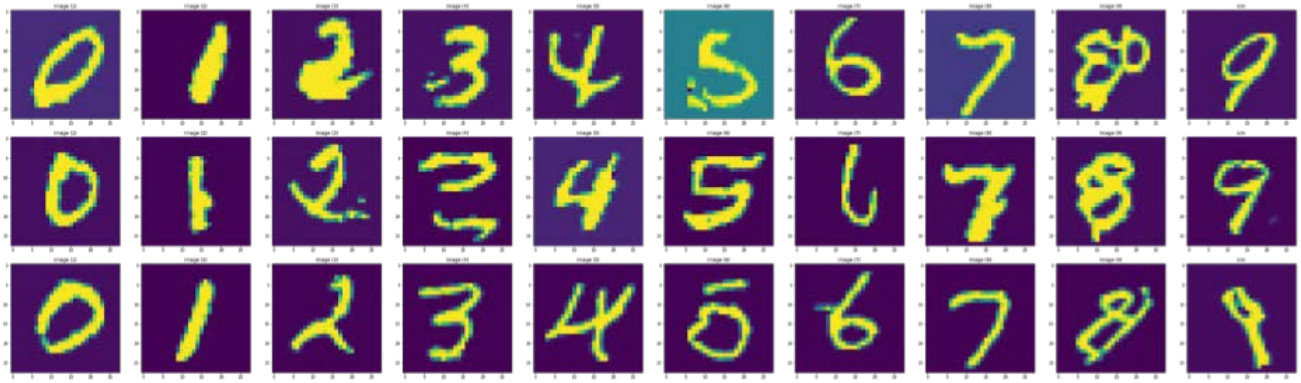


Figure 3: Samples generated by the PixelMRF after being trained on the MNIST dataset

been variations of a deep MRF in the past (Wu, Lin, and Tang 2016), they had to combine their model with others in order to successfully generate samples. With markov random fields, it can be difficult to determine global structure, as was the case in Wu, Lin, and Tang. In our case, however, we are able to determine it due to the use of convolutions in the network. This is what allows us to use the PixelMRF to generate samples without assistance.

It could be argued to use a Restricted Boltzmann Machine instead, as it is a simpler model that is achieves the same goal. The main difference between the two is the use of the energy function in the RBM. We use a logistic distribution instead. As the logistic distribution is continuous with respect to pixel values, this allows us more flexibly and robustness in our predictions. However, as the two have similar challenges, such as when it comes to calculating likelihood, one possible research path is seeing how exactly they compare.

Future Work

In the future, we plan to look into using contrastive divergance for training opposed to pseudolikelihood. By taking the gradient of log Z into account, the network will be able to train faster and be more accurate. We also plan to do a comparison of the PixelMRF to Restricted Boltzmann Machines as discussed before. Through the comparison, we hope to gain new insight into the PixelMRF along with its pitfalls. Something else to look into is variations of PixelMRF's architecture. One of these posible compositions is having four PixelCNNs, one for each corner.

AIS Testing and Use

We plan to test our version of AIS on the PixelCNN to confirm that it does indeed work on larger networks. Once we are able to do this verification, we will use the annealed importance sampling to estimate the partition function of the PixelMRF. Then, we will be able to compare the estimated log likelihood of the PixelMRF to the (estimated or bounded) log likelihood of other generative models.

Applications of PixelMRF

There are many datasets we hope to apply the PixelMRF to in the future. These include CIFAR-10, ImageNet, and LFW. From looking at these datasets which are larger in both number of images along with image size, we hope to see whether the network will be able to determine global structure for them. Another application we are interested in is using the PixelMRF for inpainting. As one of the benefits that the model has over the PixelCNN is it's flexibility, which allows it to inpaint, it will be interesting to see how it turns out.

We also hope to compare the PixelMRF against other MRF-based networks such as the recurrent neural network based one in (Wu, Lin, and Tang 2016). One quantitative method for doing this is using the network for image super resolution, and then using the resulting PSNR for comparison. Qualitatively, we could train the model on the Brodatz and/or VisTex datasets and use it for texture generation. Another qualitative approach is using a version of the PixelMRF modified to match the network in (Wu, Lin, and Tang 2016) for natural image generation.

One additional application that the PixelMRF could be for image segmentation. For this, it can be compared against (Rother, Kolmogorov, and Blake 2004) or (Chen et al. 2018).

Conclusion

As an undirected model, the PixelMRF includes more flexibility and power than its directed counterparts such as the PixelCNN. However, this does come at a price. The structure of the model leads to several challenges, including with training, comparison against other networks, and sample generation. That being said, there are many solutions to all of these problems as described earlier. The PixelMRF has huge potential in terms of its future applications. It could potentially be used for image restoration, inpainting, and so much more.

Acknowledgements

The authors would like to thank Chance Hamilton, Lee Sharma, and TA Nguyen for their discussions throughout this project. This material is based upon work supported by the National Science Foundation under Grant No. 1659788. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Besag, J. 1975. Statistical analysis of non-lattice data. *The statistician* 179–195.

Chen, L.-C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; and Yuille, A. L. 2018. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence* 40(4):834–848.

Elgammal, A.; Liu, B.; Elhoseiny, M.; and Mazzone, M. 2017. Can: Creative adversarial networks, generating" art" by learning about styles and deviating from style norms. *arXiv preprint arXiv:1706.07068*.

Gatys, L. A.; Ecker, A. S.; and Bethge, M. 2015. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*.

Hinton, G. E. 2002. Training products of experts by minimizing contrastive divergence. *Neural computation* 14(8):1771–1800.

Neal, R. M. 2001. Annealed importance sampling. *Statistics and computing* 11(2):125–139.

Oord, A. V.; Kalchbrenner, N.; and Kavukcuoglu, K. 2016. Pixel Recurrent Neural Networks. In *International Conference on Machine Learning*, 1747–1756.

Rother, C.; Kolmogorov, V.; and Blake, A. 2004. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM transactions on graphics (TOG)*, volume 23, 309–314. ACM.

Salimans, T.; Karpathy, A.; Chen, X.; and Kingma, D. P. 2016. PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications.

van den Oord, A.; Kalchbrenner, N.; Espeholt, L.; kavukcuoglu, k.; Vinyals, O.; and Graves, A. 2016. Conditional Image Generation with PixelCNN Decoders. In Lee, D. D.; Sugiyama, M.; Luxburg, U. V.; Guyon, I.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc. 4790–4798.

Wu, Z.; Lin, D.; and Tang, X. 2016. Deep markov random field for image modeling. In *European Conference on Computer Vision*, 295–312. Springer.

Video Frame Interpolation via Pixel Polynomial Modeling

Chance Hamilton
Departments of Mathematics and Computer Science
Florida Gulf Coast University
Fort Myers, Florida, USA
Email: cjhamilton4176@eagle.fgcu.edu

Jonathan Ventura
Department of Computer Science
University of Colorado, Colorado Springs
Colorado Springs, Colorado, USA
Email: jventura@uccs.edu

Abstract

Video frame interpolation (VFI) is the practice of generating intermediate video frames given two consecutive frames. We propose a novel VFI algorithm that uses a polynomial to model the change in pixel values for each pixel with respect to time. We have use a neural network to predict the coefficients of each of the polynomials and use the nice mathematical properties of polynomials to generate a parametric generative model. We have created a novel parametric generative model that can quickly synthesize multiple video frames between two consecutive frames.

Introduction

Video Frame Interpolation (VFI), as seen in Fig. 1, is a well-studied problem in image and video processing. VFI algorithms have numerous applications, such as temporal upsampling for generating slow motion video (Jiang et al. 2017), virtual view synthesis (Martinian et al. 2006), and most importantly frame rate conversion (e.g., converting between broadcast standards) (Niklaus, Mai, and Liu 2017a). The demand for high-quality video streaming has skyrocketed due to the popularity of streaming application such as Youtube, Netflix, and Hulu, and as a result, the demand for quality and efficient video frame rate conversion algorithms has increased exponentially. Generally speaking, the development of efficient frame rate-up algorithms is challenging due to the fact that in this process the algorithm must generate intermediate frames that are visually appealing while still maintaining benchmark performance. One avenue of research for scaling up temporal resolution is VFI, and in recent years VFI algorithms have set state-of-the-art standards (Meyer et al. 2015; Niklaus, Mai, and Liu 2017a).

Related Work

Block Based Modeling and Prediction

Traditionally, researchers utilized a block-based segmentation approach to analyze and synthesize the intermediate frames block-by-block using various machine learning techniques (Jeon et al. 2003; Schutten and De Haan 1998; Ishwar and Moulin 2000). However, this method often introduce holes in the segmentation as well as overlapping blocks that may cause the interpolated frames to be blurry and inconsistent with the frame sequences. These problems have been addressed in order to improve interpolated frame quality; however, overlapping and hole processing methods introduce complexity to the algorithm as well as performing poorly on videos with non static viewpoints (e.g. videos with camera movement or panning) (Schutten and De Haan 1998; Jeon et al. 2003; Ishwar and Moulin 2000). The question then arises, "How do we overcome the pitfalls that are inherent in block based interpolation?"

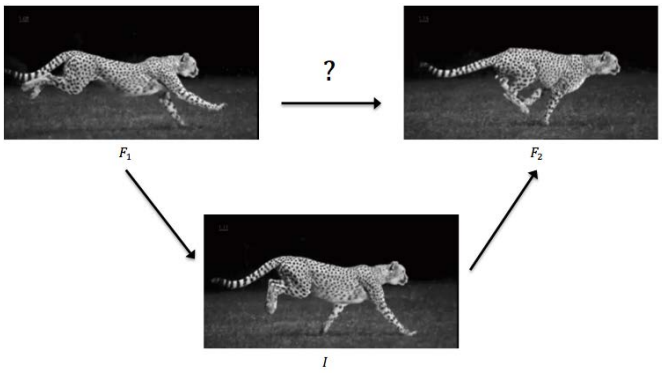


Figure 1: Example of a VFI problem. Given frames F_1 and F_2 the goal is to interpolate frame I

CNNs to Synthesize Individual Pixels

More recently, the use of Convolution Neural Networks (CNNs) to synthesize pixels of intermediate video frames have set state-of-the-art standards (Niklaus, Mai, and Liu 2017a; Ren et al. 2017; Meyer et al. 2015; Niklaus, Mai, and Liu 2017b; Karani et al. 2018; Wadhwa et al. 2013; Ascenso, Brites, and Pereira 2005; Fuchs et al. 2010; Baker et al. 2011). A pixel based approach circumvents the pitfalls of the block based approach because interpolated frames are synthesized on a pixel bases and thus has no overlapping or holes in the segmentation. One of the most promising pixel based approach to VFI was done by Niklaus et al. in 2017, where they developed a deep convolutional neural network to estimate a proper convolutional kernel to synthesize each output pixel in the interpolated video frames. When compared quantitatively to more than one hundred

methods reported in the Middlebury benchmark, their algorithm was found to be the best on two of the four real-world datasets and placing 2nd and 3rd on the remaining two sets; however, they found that their method preformed poorly on synthetic datasets, because their network was only trained on real-world scenes (Baker et al. 2011; Niklaus, Mai, and Liu 2017a). Qualitatively speaking, the phase based method developed by Myers et al. in 2015 and the convolutional kernel method mentioned above were both found to handle challenging scenarios, such as blurry regions and abrupt brightness changes, better than the optical flow estimations that were reported in the Middlebury benchmark (Niklaus, Mai, and Liu 2017a; Meyer et al. 2015; Baker et al. 2011). However, these models both fail to generate multiple frames.

We have developed a similar pixel based approach to this problem; more specifically, we are using a CNN to learn a parametric generative model that interpolates any number of immediate frames. Our parametric generative model is utilize polynomials to model the change in pixel values on each channel with respect to time. Our solution is novel since no one, has used polynomials to model pixel values over time in order to generate intermediate frames. Our goal is to create a model that will maintain benchmark performance when doubling (or even tripling) the temporal resolution of any given video.

Research Questions and Hypothesis

Our research is primarily focused on answering the following questions:

Q1 How does our method compare to state-of-the-art methods, like those reported in the Middlebury benchmark when generating a single frame between two input frames?

Q2 How well does our model produce multiple frames and quantitatively how do they compare to ground truth?

Q3 What type of video content can our method handle the best/worst?

From the questions above we are able to form the following hypothesis:

H1 Given any type of video content, our method will produce interpolated frames that are both quantitatively and qualitatively comparable to state-of-the-art methods.

H2 Given any video frame sequence, our method will be able to triple or even quadruple the temporal resolution.

H3 Our model will be able to handle challenging scenarios such as non-static backgrounds and frame sequences with large movements.

Implementation Overview

We have built a CNN model that takes the frames F_0 and F_k as input and returns a polynomial for each pixel on each channel that calculates the pixel value at time t. For example, the polynomial from Eq. 1 calculates the pixel value $x(t)$ at time t.

$$x(t) = c_n t^n + c_{n-1} t^{n-1} + \ldots + c_0 \quad (1)$$

Our model predicts the coefficients $c_0, \ldots, c_n$ when given the two frames F_0 and F_k. We will train our network on full frame sequences (i.e., $F_0, \ldots, F_k$). The coefficients are used to calculate the absolute difference between actual pixel values and the predicted pixel values, as well as the absolute difference between the approximate derivative of ground truth and the derivative of the polynomials. We explain this in further detail in the section entitled Loss Function.

The Models

Foundational Model: DispNet

DispNet is modeled after an encoder-decoder design which utilizes a process known as skip layer connections (Zhou et al. 2017). This process uses layers on the encoder side and concatenates them with layers on the decoder side so that the model can regain some of the information that the layers may have lost during the encoding process. More specifically, by concatenating the layers from the encoder side to the layers of the decoder side the model should produce clear and crisp output.

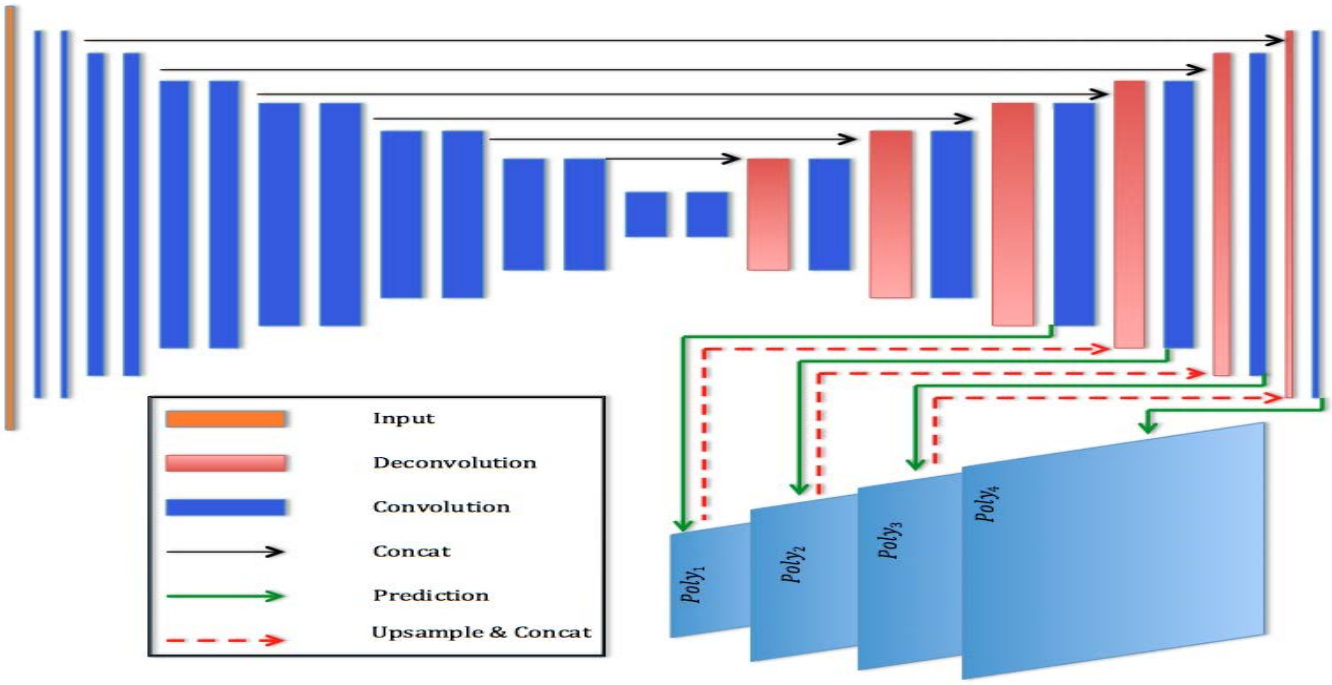
DispNet has multi-scaled side predictions that are used to compare the model's output and ground truths at varying scales. By comparing the multi-scaled side predictions and the corresponding scaled ground truths we can make our loss function more robust which should help combat overfitting. The encoder-decoder architect of DispNet is ideal for dealing with video frame sequences that contain large movements since the model scales down the input by a factor of sixteen. This process reduces the larger movement to smaller movements and makes them more readily detected by the kernels of each of the convolution layers. For these reasons we have chosen to adopt the dispNet architecture as the foundation for our VFI model. We will refer to our model, described herein, as PolyNet on account that our model will learn the coefficients for a video frame's pixel polynomials.

PolyNet's Architecture

PolyNet, as seen in Fig. 2, is a thirteen layer CNN (not including the input layer). The first 4 conv layers have a kernel size of 7, 7, 5, and 5 respectively with the remaining layers each having a kernel size of 3.

Polynet's Output

The $Poly_i$s (seen in fig. 2) can be thought of as a two dimensional matrix whose dimensions match the scaled spatial dimensions of the input frames. Each element of this two dimensional matrix correspond to a pixel. For each pixel element in the two dimensional matrix, the model learns three coefficient vectors (one for each channel). More specifically, each of the coefficient vectors consists of $n + 1$ elements where n is the degree of the polynomial used to model that pixel's specific RGB value with respect to time. This is visualized in Fig. 3.



.

Figure 2: (PolyNet) Note that each increase/reduction in block size indicates a spatial dimension change of factor 2. PolyNet produces four predictions (three side predictions at smaller spatial scales and one final prediction that matches the spatial dimensions of the input frames) which are labeled as the $Poly_i$s.

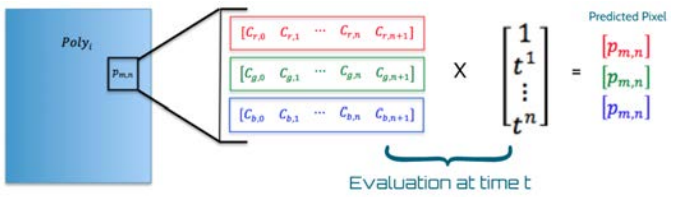


Figure 3: Here we see a break down of the predictions of polyNet. Note that $p_{m,n}$ corresponds to the pixel located at m, n. For each $p_{m,n}$ there are three coefficient vectors (one for each channel). These vectors are used to form the pixel polynomials that are evaluated at time t to predict the pixel values for all three channels.

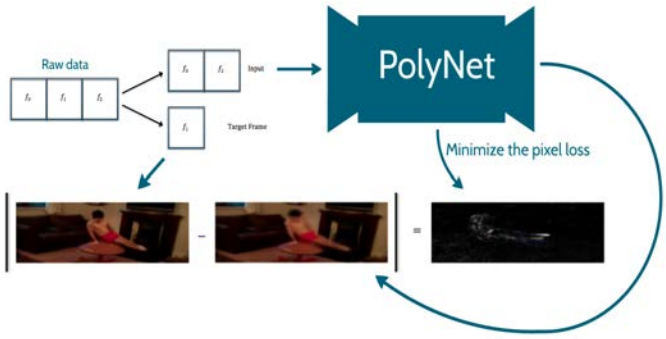


Figure 4: The frame on the far left is the ground truth and the frame in the middle is the model's prediction. On the far right, we see the absolute difference between the two frames.

Loss Function Our model is trained using a custom loss function that consist of three separate terms $\mathcal{L}_p$, $\mathcal{L}_g$, and $\mathcal{L}_d$. Our loss functions are primarily geared towards minimizing the absolute pixel loss $\mathcal{L}_p$ (see Eq. 2) between all of the ground truth frames and the corresponding predictions (this can be seen in Fig. 4).

$$\mathcal{L}_p = \sum_{t=0}^{k} |f_t - p(t)| \tag{2}$$

Note that in Eq. 2, the f_t denote actual pixel values and the $p(t)$ denote polynomial outputs at time t.

Early testing on the validation sets showed us that the model learns how to produce the coefficients that accurately model the colors of the pixel; however, the structure of moving objects were often blurry and contained ghosting and artifact effects present in the predictions (refer to Fig. 5).

In order to deal with this blurring issue we needed a way to train the model on the structural information in the frame sequences. We were able to accomplish this in two different ways: derivatives with respect to the x and y dimensions, also known as image gradients, and derivatives with respect to time. Image gradients are calculated using the image gradient method from the TensorFlow image library. Once we obtain the image gradients from the input frames and the predicted frames, we take the absolute difference between

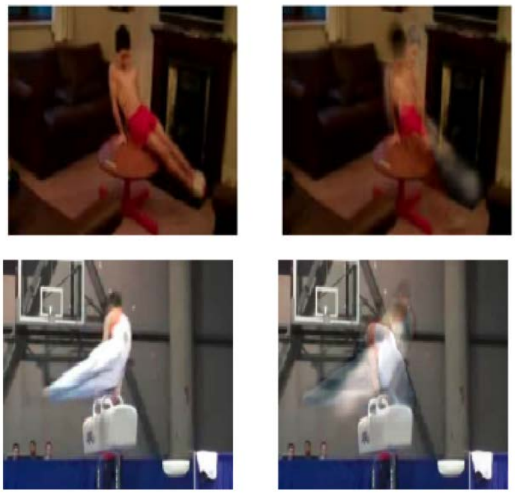


Figure 5: The frames on the left are the ground truth and the frames on the right our the model's prediction. We see on the top right prediction the image has a ghosting effect and the bottom right prediction has artifact effects.

the two and add this term to our loss function (refer to Eq. 3). By adding this term our model should learn how to detect the edges of objects and thus learn how to maintain the edges of the objects as we interpolate frames.

$$\mathcal{L}_g = \sum_{i=0}^{k} |\frac{d}{dx} f_i - \frac{d}{dx} p_i| + |\frac{d}{dy} f_i - \frac{d}{dy} p_i| \tag{3}$$

Note that in Eq. 3 the f_i denotes the input frame at time i, and p_i denotes the predicted frame at time i. Here we are summing over $0, \ldots, k$ since we assuming there are k frames in the sequence.

When tested on the validation sets, the addition of the $\mathcal{L}_g$ term to the loss function improved the performance of the model; however, it was still not on par with what the state of the art was reporting for their validation metrics. In order to close the gap between our model and the state of the art, we added a term that represented the absolute difference between approximate derivative of input frames with respect to time and the derivatives of the pixel polynomials with respect to time. To approximate the derivative of an input frame, say f_i, with respect to time we calculate the difference between the frame after (f_{i+1}) and before (f_{i-1}), and then divide by the change in time. Since the approximation of the derivative requires there to be a frame before and after the frame in consideration, we can only calculate this term for the frames in the middle of the first and last frames of the input sequence. This derivative can be expressed mathematically as Eq. 4.

$$f'_k = \frac{f_{k+1} + f_{k-1}}{2} \tag{4}$$

Note that in Eq. 4 the f'_k denotes the derivative of the frame f_k at time k. Also note that we divide by two since we are always approximating the derivative by comparing the average rate of change of the frames before and after the frame we are interested in, and gives us a time change of two.

Once we calculate f'_k, we need to compare this derivative to the derivative of the pixel polynomials evaluated at the same time k. We calculate the derivative of the pixel polynomials very simulare to how we calculated the predicted frames; this can be seen in Fig 6.

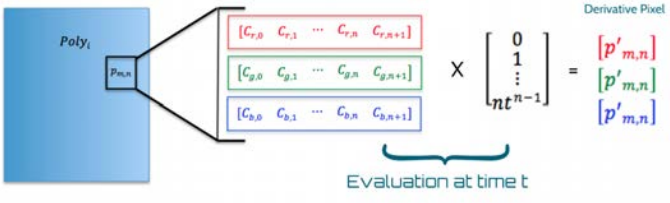


Figure 6: Here we see how we use the predictions of polyNet to calculate the derivative with respect to time. Just like in Fig. 3, we use matrix multiplication to differentiate with respect to time t for all three color channels.

Finally, we can add the absolute difference between the approximation and the predicted derivatives to the loss function. Mathematically this is represented as Eq. 5.

$$\mathcal{L}_d = \sum_{i=1}^{k-1} |f'_i - p'_i| \tag{5}$$

Note that in Eq. 5, the f'_i denotes the approximate derivative for the input frame f_i and p'_i is the derivative of the predicted frame p_i evaluated at time i.

In the Experimentation section, we will break down how each of these terms are combined in the loss function and also report how these combinations preformed on our validation sets.

Training Procedure and Training Data We are currently training PolyNet on frame sequences consisting of three frames. PolyNet takes the first and last frame of the sequence as input and learns the coefficients needed to predict all three of the frames. As the model is training, the predicted frames are compared to the ground truth and the $\mathcal{L}_p$, $\mathcal{L}_g$, and $\mathcal{L}_d$ are calculated using the methods described above. Again, the primary goal of the model is to produce the coefficients that minimize the pixel loss as much a possible for all scaled predictions. Fig. 7 is a schematic of the training procedure that is currently implemented by PolyNet.

For training, we have adopted the UCF101 dataset developed by Soomro, Zamir, Shah in 2012 (Soomro, Zamir, and Shah 2012). This data set is composed of 101 action classes, over 13k clips, and 27 hours of video data. The database consists of realistic user uploaded videos containing camera motion and cluttered background. We converted the video into frame sequences of length three. After converting the videos into frame sequences we randomly set aside $\sim 10\%$ for validation and trained on the rest. The frames are 416 x 128, and are used in their entirety for training. Since our model

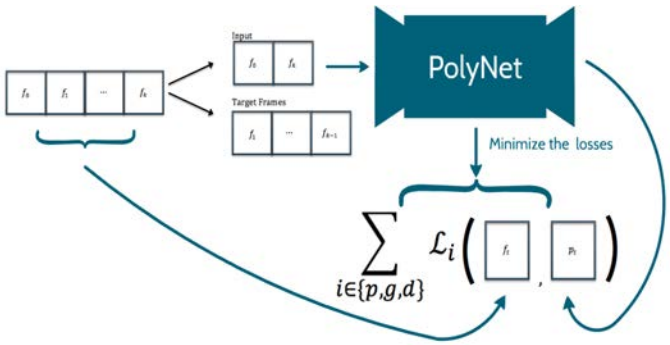


Figure 7: Our model takes in the first and last frame of a sequence and fits the pixel polynomials to model all the frames. The model is subjected to minimize the loss terms $\mathcal{L}_p$, $\mathcal{L}_g$, and $\mathcal{L}_d$.

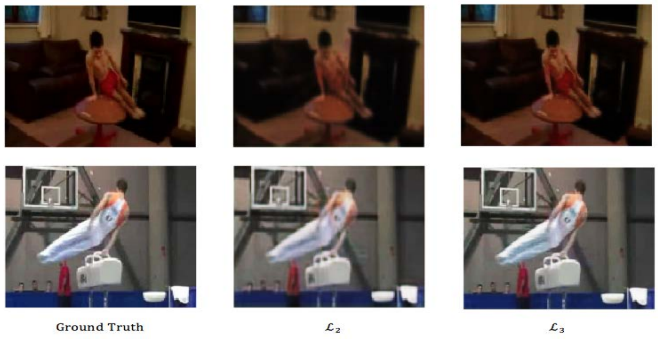


Figure 8: This showcases the differences in the interpolated frames when the model was trained subjected to $\mathcal{L}_2$ and $\mathcal{L}_3$.

learns the information from the frames before and after the target frame, we do not have implement boundary handling procedures and our model can readily predict the pixels on the edges of the frames without the need for padding.

Experimentation

For experimentation we developed the following three loss functions:

$$\mathcal{L}_1 = \mathcal{L}_p \tag{6}$$

$$\mathcal{L}_2 = \mathcal{L}_p + \mathcal{L}_g \tag{7}$$

$$\mathcal{L}_3 = \mathcal{L}_p + \mathcal{L}_g + \mathcal{L}_d \tag{8}$$

Where $\mathcal{L}_p, \mathcal{L}_g$, and $\mathcal{L}_d$ are those loss terms that we defined in the Loss Function section. We trained the our model subject to these different loss functions with third degree pixel polynomial. When we ran our model on our validation set, we were able to obtain the results reported in the table below.

Loss Function	MAE	SSIM	PSNR	RMSE
$\mathcal{L}_1$	0.02	0.91	28.26	0.04
$\mathcal{L}_2$	0.02	0.92	29.69	0.04
$\mathcal{L}_3$	0.01	0.97	35.10	.01

Table 2: Extensive quantitative evaluation of the validation set

In the Fig. 8, we see difference in interpolated frames when the model was trained on $\mathcal{L}_2$ and $\mathcal{L}_3$ respectively.

Benchmark Evaluation

In order to compare our model with those of state of the art, we tested our model on the Middlebury testing set and reported the average interpolation error, which is the metric used in the Middlebury benchmark (Baker et al. 2011). These results can be found in Table 1 located in Appendix A. We can see from the results that qualitatively our model does not surpass those reported from Niklaus et al. nor Myers et al. (Niklaus, Mai, and Liu 2017a). In fact, at first glance some may say our model failed miserably. However, we encourage the reader to remember one important fact, the quality of the training data greatly influences the quality of the of the trained model. Niklaus et al. was using videos that are rated as high quality (Niklaus, Mai, and Liu 2017a). We on the other hand are using videos of poor quality, to which we contribute our poor performance on the Middlebury Benchmark. However, we have shown through cross validation that our model can learn to handle very challenging video quality and still produce visually coherent frames. Our model may prove to have applications in the video surveillance industry, which is notorious for having poor video quality.

Project Novelty

Despite our model poor performance on the Middlebury Benchmark, our model has the novel ability to generate multiple frames between the input. In most cases, the state of the art fails to produce more than a single frame between input frames. Since our model uses parameterized polynomials to model the pixels in a frame we can generate multiple frames in between the input frames by evaluating the pixel polynomials at different time t. Fig. 9, located in Appendix A, showcases a few examples of the series of frames we generated from a given input sequence. One last aspect of our project that sets it apart from other VFI models is the training time. Our model was trained using a single GeForce GTX 1080ti GPU and was able to train on the entirety of our dataset in 5 hours and 15 minuets. This is faster than the convolutional kernel method developed by Niklaus et al. which was reported to takes 20 hours to train using four Nvidia Titan X (Pascal) GPUs.

Current Limitations and Future Works

As you can see in Fig. 8 and 9, our model does a decent job at interpolating frames. However, some of the interpolated frames blur the object in motion and have artifacts and ghosting effects. Although these are common short comings in VFI algorithms, we still wish to address these issues in our future work. One possible solution is to implement a supervisor to our model architecture. An optical flow supervisor would be ideal for learning the flow from frame to frame, which would help the model produce polynomials that main-

Loss Function	Mequon	Schefflera	Urban	Backyard	Basketball	Dumptruck	Evergreen	AVERAGE
Our $\mathcal{L}_2$	16.6	16.8	12.7	18.6	14.9	22.93	16.7	17.0
Our $\mathcal{L}_3$	10.7	11.7	8.91	12.4	9.58	11.85	11.0	10.9
Niklaus $\mathcal{L}_1$	**2.52**	3.56	4.17	10.2	5.47	6.88	**6.63**	**5.61**
Niklaus $\mathcal{L}_F$	2.60	3.87	4.38	**10.1**	5.98	6.85	6.90	5.81
MDF-Flow2	2.89	**3.47**	3.66	10.2	6.13	7.36	7.75	5.82
DeepFlow2	2.99	3.88	**3.62**	11.0	5.83	7.60	7.82	6.02
AdaConv	3.57	4.34	5.00	10.2	**5.33**	**7.30**	6.94	6.20

Table 1: Results from the Middelbury benchmark evaluation. The table reports the Average Interpolation Error, also known as root-mean-error.

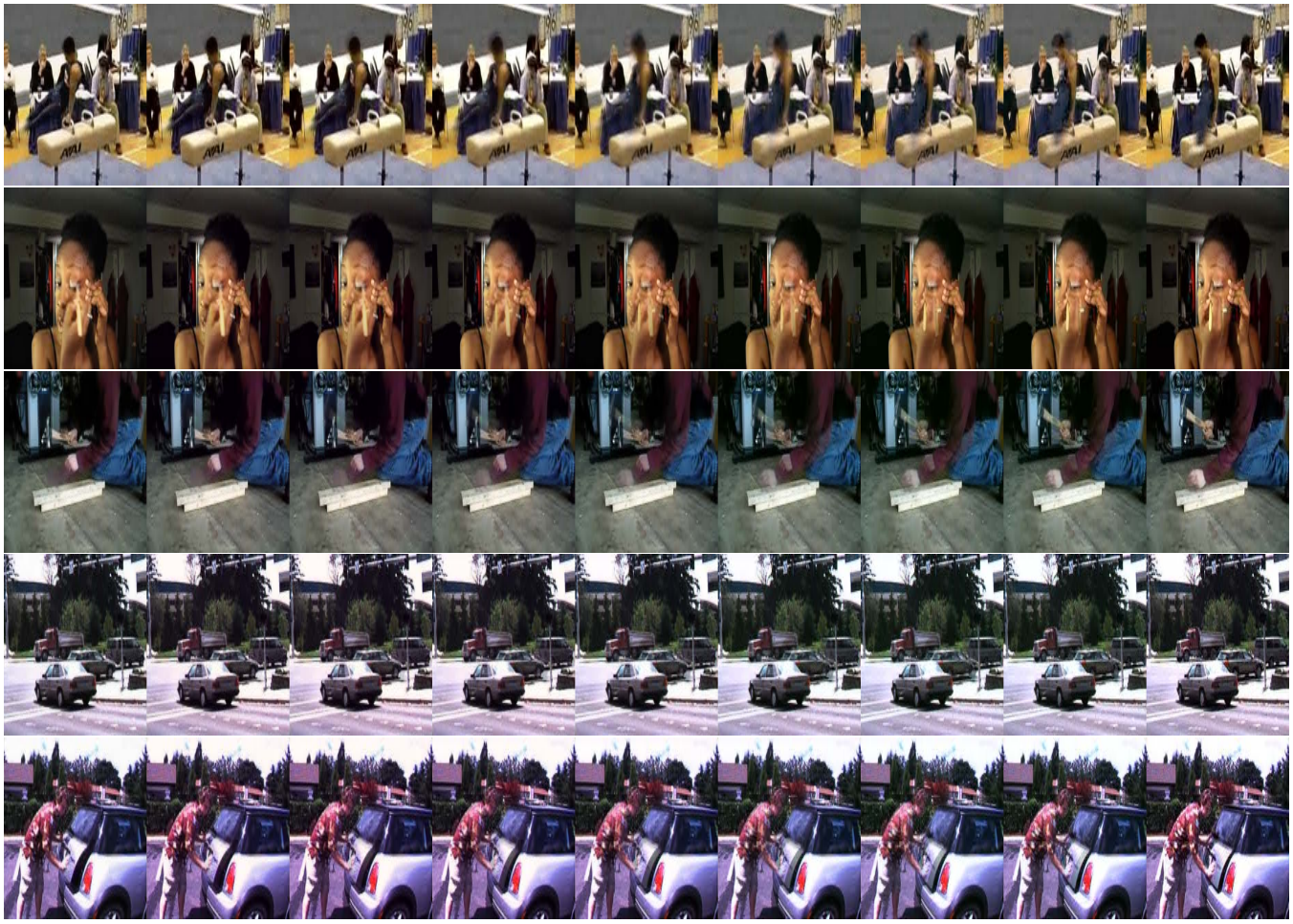


Figure 9: Our model was able to to produce seven intermediate frames between the input frames. The top three frame sequences were generated from our validation set and the bottom 2 are from the Middlebury test set.

tain the flow in the frames, de-blurring the objects in motion. Another option would be to feed the output of our model into an image reconstruction and image synthesis model such as the one developed by Chuan Li and Michael Wand in 2016 where they used a dCNN that utilized a Markov random field to synthesize 2D images (Li and Wand 2016). This model could be used to reconstruct the parts of the frame that seem blurry using the information learned from the input frames.

We also would like to explore how training on different data sets might effect the outcome of the experiments. The UCF101 data set is a nice well rounded video data set. However, the videos are not what we would call high quality in terms of today's standards. The YouTube 8M data set, for example, might prove to be a great data set to train on since the data set contains many videos with 4k resolution. This may result in our model producing more clear and crisp interpolated frames and possibly put us at state of the art or at the very least greatly improve our performance on the Mid-

dlebury evaluation.

One last avenue that we are interested in exploring is changing the training procedure slightly so that it resemble a residual learning algorithm. In order to do this, we will eliminate the constant coefficient in the pixel polynomials and replace these with the pixel values of the first frame in the sequence. In doing this we hope to ease the training of PolyNet and shift the importance of learning coefficients that more accurately models the intermediate frames rather than the input frames.

Conclusion

We have proposed a novel method to synthesize the pixel of interpolated frames by training a CNN to predict polynomials that model the pixel values on each channel with respect to time. We have adapted our model from the DispNet architecture that was implemented in 2017 by Zhou et al. for an unsupervised model to predict depth and pose from video frames (Zhou et al. 2017). Our model was trained using the UCF101 dataset and was shown to be able to train faster than state of the art and was able to produce multiple interpolated frames, unlike the state of the art models. Although our model preformed poorly on the Middlebury Benchmark, we have shown that our model can learn to handle challenging video quality and still produce interpolated frames that are visually coherent. In short our model can be summed up by the following phrase: "Quantity over Quality"! Yet, there is still hope to change "Quantity over Quality" to "Quantity and Quality". We discussed future directions to address the short comings of our model and hope to implement these ideas so that we can run the experiments to see how we compare to state of the art. Overall we have shown that polynomials are decent interpolaters and given the right training data may prove to be great interpolaters.

Acknowledgement

We would like to thank Dr. Terry Boult for his helpful insights to the wonders of polynomials. We would also like to thank the Computer Science department at the University of Colorado at Colorado Springs for allowing access to the GPU servers. This material is based upon work supported by the National Science Foundation under Grant No. 1659788. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Ascenso, J.; Brites, C.; and Pereira, F. 2005. Motion compensated refinement for low complexity pixel based distributed video coding. In *IEEE Conference on Advanced Video and Signal Based Surveillance.*, 593–598. IEEE.

Baker, S.; Scharstein, D.; Lewis, J.; Roth, S.; Black, M. J.; and Szeliski, R. 2011. A database and evaluation methodology for optical flow. *International Journal of Computer Vision* 92(1):1–31.

Fuchs, M.; Chen, T.; Wang, O.; Raskar, R.; Seidel, H.-P.; and Lensch, H. P. 2010. Real-time temporal shaping of high-speed video streams. *Computers & Graphics* 34(5):575–584.

Ishwar, P., and Moulin, P. 2000. On spatial adaptation of motion-field smoothness in video coding. *IEEE Transactions on Circuits and Systems for Video Technology* 10(6):980–989.

Jeon, B.-W.; Lee, G.-I.; Lee, S.-H.; and Park, R.-H. 2003. Coarse-to-fine frame interpolation for frame rate up-conversion using pyramid structure. *IEEE Transactions on Consumer Electronics* 49(3):499–508.

Jiang, H.; Sun, D.; Jampani, V.; Yang, M.-H.; Learned-Miller, E.; and Kautz, J. 2017. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. *arXiv preprint arXiv:1712.00080*.

Karani, N.; Tanner, C.; Kozerke, S.; and Konukoglu, E. 2018. Reducing navigators in free-breathing abdominal mri via temporal interpolation using convolutional neural networks. *IEEE Transactions on Medical Imaging* 1.

Li, C., and Wand, M. 2016. Combining markov random fields and convolutional neural networks for image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2479–2486.

Martinian, E.; Behrens, A.; Xin, J.; and Vetro, A. 2006. View synthesis for multiview video compression. In *Picture Coding Symposium*, volume 37, 38–39.

Meyer, S.; Wang, O.; Zimmer, H.; Grosse, M.; and Sorkine-Hornung, A. 2015. Phase-based frame interpolation for video. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1410–1418. IEEE.

Niklaus, S.; Mai, L.; and Liu, F. 2017a. Video frame interpolation via adaptive convolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 670–679.

Niklaus, S.; Mai, L.; and Liu, F. 2017b. Video frame interpolation via adaptive separable convolution. In *Proceedings of the IEEE International Conference on Computer Vision*, 261–270.

Ren, Z.; Yan, J.; Ni, B.; Liu, B.; Yang, X.; and Zha, H. 2017. Unsupervised deep learning for optical flow estimation. In *Association for the Advancement of Artificial Intelligence*, 1495–1501.

Schutten, R., and De Haan, G. 1998. Real-time 2-3 pull-down elimination applying motion estimation/compensation in a programmable device. *IEEE Transactions on Consumer Electronics* 44(3):930–938.

Soomro, K.; Zamir, A. R.; and Shah, M. 2012. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*.

Wadhwa, N.; Rubinstein, M.; Durand, F.; and Freeman, W. T. 2013. Phase-based video motion processing. *ACM Transactions on Graphics (TOG)* 32(4):80.

Zhou, T.; Brown, M.; Snavely, N.; and Lowe, D. G. 2017. Unsupervised learning of depth and ego-motion from video. In *CComputer Vision and Pattern Recognition*, volume 2, 7.

Deep Learning for Denoising of Fluorescence Microscopy Images

Tram-Anh Nguyen
George Mason University
Fairfax, VA
tnguy150@gmu.edu

Guy Hagen and **Jonathan Ventura**
University of Colorado Colorado Springs
Colorado Springs, CO
ghagen@uccs.edu, jventura@uccs.edu

Abstract

Fluorescence microscopy images are often taken at low light and short exposure times to preserve the integrity of cell samples. However, imaging under these conditions leads to severely degraded images with low signal to noise ratios. To computationally restore these images, we introduce novel loss functions to denoise microscopy images. These loss functions will be folded into the CARE algorithm. The results produced by this modification will be evaluated against traditional TV filtering and NL means techniques. The modified model will also be compared against its CARE predecessor using standard image quality metrics.

Introduction

Fluorescence microscopy is vital for understanding processes and structures at the cellular level. Because imaging at the cellular level under strong lighting conditions or long exposure times may damage the cell sample through phototoxicity, fluorescence microscopy images need to be restored. A safe way to image a cell is to use low light conditions and/or low exposure times, which unfortunately lowers the signal to noise ratio (Xing et al. 2017).

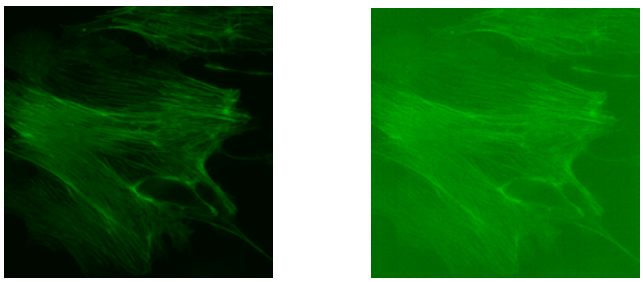


Figure 1: Fluorescence microscopy images taken of actin under high light (left) and low light conditions (right)

Noise in an image depends on a combination of factors, including exposure time and physical experimental conditions. In fluorescence microscopy, noise is typically described by a Poison-Gaussian model. There has been extensive work done in image restoration through filtering noise from microscopy images. These filtering techniques have limitations, which call for a more generalized solution.

Deep learning methods have been successful in restoring corrupted images, as well as in other image processing tasks such as classification, segmentation, and object detection. A widely used deep learning architecture in image processing is a convolutional neural network (CNN). A CNN consists of an input layer, hidden layers, and an output layer. Another popular network in image processing is an autoencoder. Typically, autoencoders are used for denoising and reducing the number of dimensions of input data (Xing et al. 2017).

Related Work

Deep learning approaches to image deblurring may involve blind and non-blind image deconvolution. There are a wealth of studies devoted to the non-blind image deconvolution approach, but these networks are limited, as they rely on information about the non-blurry image beforehand. By contrast, blind deblurring models are more flexible since information about the non-blurry image is not required for the network to deblur an input image.

In 2014, Xu et al. introduced a natural image deconvolution that is data-driven and does not rely on traditional assumptions. For example, generative models tend to assume that noise in an image is identically and independently distributed, even if this assumption is not necessarily true. Instead, this CNN was trained on images that were not deblurred ahead of time and the network learned the deconvolution operation without requiring information about the original image. The main contribution of Xu et al. was developing this deep convolutional neural network (DCNN) that consisted of two sub models—one for deconvolution and the other for denoising. The models perform inverse filtering using large 1D kernels and the former sub model is pre-trained to mimic Weiner deconvolution (Xu et al. 2014).

A major drawback of the previously discussed blind DCNN was that it failed if the original image was not blurry. To address this shortcoming, Conti et.al introduced a convolutional neural network that consisted of a regularization term in the cost function. This improved model was able to denoise a blurry image and maintain the quality of images that are not noisy. The major modifications made by Conti et al. were that they used single 2D convolutional layers rather than 1D kernels for deblurring. The regularized

cost function was built using the results of a classification network trained to distinguish blurry and non-blurry images that had roughly 80 percent accuracy when evaluated(Conti, Minucci, and Derakhshan 2017).

More recently, Weigert et al. published a series of image restoration methods that succeeded in restoring seven images of various organisms (i.e. planaria flatworm, fruit fly wings). The major contributions of this work include: generating training data without requiring manual labeling, replicating live imaging for organisms in which live imaging had once been near-impossible, and restoring microscopy images even when lighting conditions are reduced by 60-fold. Weigert et al. demonstrated high quality results with restorations of images containing *Tribolium castaneum* (red flour beetle) and *Schmidtea mediterranea* (a flatworm commonly known as planaria). This Content-Aware Image Restoration (CARE) network is based on the U-Net network, which consists of an encoder-decoder architecture. The only difference between the CARE network and the U-net algorithm is that the former outputs a per-pixel Laplace distribution whereas the latter outputs one value per pixel. Although these restoration methods are promising, each pair of image content and corruption requires a unique data set. Each model must be retrained for an image of that particular content and corruption to be successfully restored (Weigert et al. 2017).

Modifying network architecture is a popular strategy to achieve better performance. Another widely used approach to this goal is data augmentation. Data augmentation is widely used for image classification(Paulin et al. 2014), as applying random transformations to training data effectively provides more data for a network to learn from. Not surprisingly, training with large data sets produces high-quality image restorations (Burger, Schuler, and Harmeling 2012). Typical transformations for data augmentation include translation, rotation, and scaling. Augmenting data is a manual process in which these image transformations are specified by humans. To automate this process, Jain et al. developed an unsupervised learning procedure that generated training samples using different noise models (Jain and Seung 2009). Likewise, Hauberg et al. developed a learned augmentation scheme that outperforms manual augmentation of MNIST data when used as training for a multilayer perceptron and a CNN (Hauberg et al. 2016).

Lastly, altering the loss function is a viable strategy, though this approach tends to be overlooked (Zhao et al. 2017). Typically, the mean absolute error and mean squared error loss functions are employed in image processing networks (Zhao et al. 2017; Burger, Schuler, and Harmeling 2012; Agostinelli, Anderson, and Lee 2013; Chen et al. 2018). In 2017, Zhao et al. introduced a new loss function for image restoration that combined the multi-scale SSIM (MS-SSIM) metric with L_1 loss. Without changing network architecture, Zhao et al. demonstrated that by using this mixed loss function, their fully convolutional neural network outperformed state-of-the-art networks on tasks such as joint denoising and demosaicking (Zhao et al. 2017). Drawing inspiration from this approach, we will replace the existing Laplace loss function of the CARE network with a novel loss function that enhances edge restoration in fluorescence microscopy images.

Research Questions and Hypothesis

The questions we will address in this study are:

1. How well does the CARE network perform on our microscopy data set?
2. How can we produce image restorations that are sharper than those produced by CARE?
3. How does the network perform when it is trained on one kind of sample and tested on a different kind of sample?
4. Can we reliably restore live cell images?

We hypothesize that our improvement on the method of Weigert et al. will restore microscopy images that are less blurry and more detailed than the restorations of the original CARE model. Thus, our model will be a more faithful solution compared to the denoising approach of Weigert et al.

Proposed Implementation

We modified the loss function of the CARE network in hopes of producing restorations that are more faithful to ground truth images. Ultimately, the objective of this study is to restore microscopy images free of artifacts and without loss of fine details.

Experimental Setup

The data set we used to train the standard CARE model is High Low, which consists of over 400 fluorescent images of actin and mitochondria, in addition to 170 images of dendra. All images in the High Low data set were taken using an Olympus IX83 with 60X/1.3NA objective lens. The Andor Zyla CMOS camera was used to image cell organelles.

For all of our experiments, we first observed the behavior of the network and assessed the quality of the network without loss function modifications. These results were then compared to results produced by training the CARE network using our FFT and bandpass cost functions. These experiments include restorations of actin imaged at 1 millisecond and 10 milliseconds, restorations of mitochondria imaged at 1 millisecond, model mismatch experiments, and restorations of dendra imaged at 10 milliseconds. We use the term model mismatch to indicate experiments in which images of one type of cell content are used for training while images of another type of cell content are used for testing. For example, we used images of mitochondria as training data for the CARE algorithm, and subsequently tested the model using noisy images of actin. Our most recent experiments involved restorations of dendra imaged at 10 milliseconds using dendra imaged at 10 and 400 milliseconds as the training set. The results of these experiments were evaluated using peak signal to noise (PSNR) and structural similarity (SSIM) image quality metrics.

To conduct our experiments, we used the default configurations of the standard CARE model. The training batch size was 16 images, the number of training epochs was 100, the initial learning rate was 0.0004, and the iterations per epoch (training steps) was 400. The training images were

2048 pixels wide, and 2048 pixels high, with 1 grayscale channel. In sampling the training images, 800 patches per image of size 64 pixels by 64 pixels were used to train the CARE model. In all experiments, images were split according to the ratio 4:1 for training and validation respectively. Nine or ten images were used for testing in all experiments. Table 1 provides an overview of the experiments we performed along with their abbreviations.

table

Method

The denoising method of Weigert et al. is successful when restoring images with up to 60-fold reduction in light exposure. Beyond that range, however, we found the CARE algorithm is not able to restore images with fine details. In

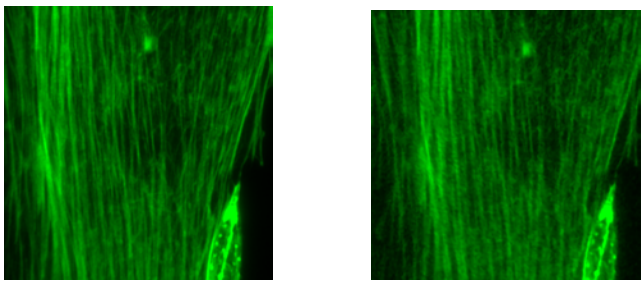


Figure 2: Ground truth image of actin (left) and CARE restoration of actin using Laplace loss (right)

the CARE network, the popular stochastic gradient descent Adam optimizer is used to minimize a Laplace loss function.

$$L_{laplace}(\theta) = \frac{1}{T}\frac{1}{N}\sum_{t=1}^{T}\sum_{i=1}^{N}\frac{|\ y_i^t - \mu_\theta(x^t)_i|}{\sigma_\theta(x^t)_i} + \log\sigma_\theta(x^t)_i$$

where T indicates the number of training images, N indicates the number of pixels per image, y^t corresponds to the ground truth pixel value, and x^t corresponds to the input pixel. μ and σ correspond to the mean and variance of the predicted pixel distribution.

Loss function modification

The restored images produced by this algorithm suffered from blurriness and lack of fine details. In response to this issue, we tested various loss functions tailored to preserve edges. Despite the strong performance of the combined MS-SSIM and L_1 loss function introduced by Zhao et al. (2017), we found a similar combined MS-SSIM and L_1 loss function yielded poor results when applied to the CARE network. Incorporating the SSIM metric into the loss function resulted in slightly better performance according to the SSIM metric, which was expected. In addition to SSIM-based loss functions, edge detection techniques were applied to the loss function. We obtained poor results by using the Sobel operator in the loss function. The loss functions that we introduce in this study are the FFT and bandpass loss functions. So far, these two loss functions have produced the most faithful results for denoising images in our High Low data set compared to other loss functions we designed.

The FFT loss function is similar to mean absolute error with one key difference. Instead of taking differences between corresponding pixel values, the FFT loss function considers differences between per-pixel frequencies represented in the 2D Fourier transforms of the restored image and its corresponding ground-truth image.

$$L_{FFT} = \frac{1}{N \times M}\sum_{j=1}^{M}\sum_{i=1}^{N} |\ f_{i,j}^r - fi, j^t|$$

where f represents the frequency at the (i,j) pixel in the Fourier transforms of the restored and ground truth images (represented by r and t, respectively). M and N represent the M x N pixels in an image. To preserve edges, the bandpass loss function was designed to emphasize high frequencies. This loss function (L_b) consists of taking a difference of Gaussians, with $\sigma_1 = 0.5$ and $\sigma_2 = 5$. These sigma values were chosen arbitrarily, and may be adjusted through trial and error. In the equation below, G^t_{i,j,σ_1} denotes Gaussian blur applied to the ground truth image with standard deviation σ_1 and G^t_{i,j,σ_2} denotes Gaussian blur applied to the ground truth image with standard deviation σ_2. Likewise, G^r_{i,j,σ_1} denotes Gaussian blur applied to the restored image with standard deviation σ_1 and G^r_{i,j,σ_2} denotes Gaussian blur applied to the restored image with standard deviation σ_2

$$L_b = \frac{1}{N \times M}\sum_{j=1}^{M}\sum_{i=1}^{N} |\ (G^t_{i,j,\sigma_1} - G^t_{i,j,\sigma_2}) - (G^r_{i,j,\sigma_1} - G^r_{i,j,\sigma_2})|$$

Results

Restoration of actin

To restore actin imaged using 1 millisecond of exposure time, the CARE algorithm was trained using sixty pairs of actin images taken using 1 millisecond and 100 milliseconds of exposure time. After the model was trained, nine testing images of actin taken at 1 millisecond of exposure time were restored using the CARE prediction function. The following table displays peak signal to noise ratios and structural similarity measurements of the initial input images and the corresponding restored images. In the following tables, the PSNR and SSIM values of the input image are displayed in the Input column , with the rest of the column headings indicating the loss function used (Laplace, FFT, and bandpass).

Actin	Input	Laplace	FFT	BP
0	26.690	**35.869**	35.676	35.304
1	28.322	35.256	33.500	**35.559**
2	29.074	**39.915**	39.747	39.833
3	33.395	42.284	42.140	**43.218**
4	31.766	39.827	**40.741**	40.657
5	32.937	39.892	**40.534**	40.406
6	31.452	38.235	36.589	**38.297**
7	27.677	**36.458**	33.306	35.642
8	32.679	**41.278**	39.607	40.980
Mean	30.444	38.779	37.982	**38.877**

Table 2: PSNR values produced by loss functions (AA)

Encoding	Training Set (Exposure Time)		Input	Ground Truth (Exposure Time)
	Low SNR	High SNR		
MM	Mitochondria (1 ms)	Mitochondria (100 ms)	Mitochondria (1 ms)	Mitochondria (100 ms)
AA	Actin (1 ms)	Actin (100 ms)	Actin (1 ms)	Actin (100 ms)
MA	Mitochondria (1 ms)	Mitochondria (100 ms)	Actin (1 ms)	Actin (100 ms)
AM	Actin (1 ms)	Actin (100 ms)	Mitochondria (1 ms)	Mitochondria (100 ms)
DDS	Dendra (10 ms)	Dendra (400 ms)	Dendra (10 ms)	Dendra (400 ms)
AAS	Actin (10 ms)	Actin (400 ms)	Actin (10 ms)	Actin (400 ms)
AAE	Actin (1 ms)	Actin (100 ms)	Actin (1 ms)	Actin (100 ms)

Table 1: Summary of experiments

Actin	Input	Laplace	FFT	BP
0	0.447	0.895	0.907	**0.914**
1	0.838	0.951	0.947	**0.954**
2	0.707	0.967	**0.970**	0.966
3	0.829	0.976	**0.981**	0.979
4	0.831	0.976	**0.978**	0.977
5	0.870	0.977	**0.980**	0.979
6	0.895	0.972	0.968	**0.977**
7	0.846	**0.956**	0.930	0.953
8	0.889	**0.980**	0.979	0.977
Mean	0.795	0.961	0.960	**0.964**

Table 3: SSIM values produced by loss functions (AA)

Mitochondria	Input	Laplace	FFT	BP
0	0.805	0.983	0.985	**0.985**
1	0.907	0.981	**0.985**	0.977
2	0.797	0.981	0.983	**0.984**
3	0.605	0.957	**0.965**	0.961
4	0.886	0.988	**0.989**	0.988
5	0.844	0.977	0.977	**0.978**
6	0.830	**0.985**	0.983	0.985
7	0.895	0.984	**0.986**	0.983
8	0.844	**0.987**	0.984	0.986
Mean	0.824	0.980	**0.982**	0.981

Table 5: SSIM values produced by loss functions (MM)

Restoration of mitochondria

An identical experiment was performed to restore mitochondria images taken with 1 millisecond of exposure time. Sixty pairs of mitochondria were used for training and validation while ten pairs of mitochondria images were used for testing. The following tables display results of this experiment.

Mitochondria	Input	Laplace	FFT	BP
0	32.073	**41.059**	40.644	40.213
1	31.624	34.233	34.474	**36.011**
2	32.167	39.965	**40.171**	39.937
3	27.757	35.261	**36.709**	35.068
4	34.548	41.057	**41.583**	40.292
5	31.943	35.940	**36.585**	35.206
6	32.995	40.704	**41.148**	39.873
7	34.169	39.387	**40.401**	39.163
8	33.385	40.702	**40.773**	39.253
Mean	32.296	38.701	39.165	38.335

Table 4: PSNR values produced by loss functions (MM)

Model mismatch to restore actin

Actin imaged at 1 millisecond was restored using the standard CARE model trained with mitochondria images. As CARE is a content-aware network, the PSNR and SSIM values produced by the algorithm were less faithful to ground truth images compared to previous experiments (MM and AA). The following tables display results of this model mismatch experiment.

Actin	Input	Laplace	FFT	BP
0	26.690	**36.666**	36.060	35.363
1	28.322	30.095	30.557	**30.661**
2	29.074	37.591	**38.431**	36.413
3	33.395	38.911	**40.440**	37.258
4	31.766	36.436	**38.631**	36.627
5	32.937	36.125	**39.027**	35.660
6	31.452	33.647	**34.482**	33.029
7	27.677	28.937	**29.406**	27.911
8	32.679	35.851	**36.631**	35.440
Mean	30.444	34.918	**35.963**	34.262

Table 6: PSNR values produced by loss functions (MA)

Actin	Input	Laplace	FFT	BP
0	0.448	0.936	0.930	**0.937**
1	0.838	0.937	**0.954**	0.941
2	0.707	0.957	**0.963**	0.958
3	0.829	0.967	**0.974**	0.962
4	0.831	0.958	**0.972**	0.962
5	0.870	0.959	**0.975**	0.956
6	0.895	0.964	**0.974**	0.962
7	0.846	0.919	**0.939**	0.895
8	0.889	0.971	**0.976**	0.967
Mean	0.795	0.952	**0.962**	0.949

Table 7: SSIM values produced by loss functions (MA)

Model mismatch to restore mitochondria

Mitochondria imaged at 1 millisecond were restored using the standard CARE model trained with actin images. The following tables display results of the second model mismatch experiment.

Mitochondria	Input	Laplace	FFT	BP
0	32.073	39.803	**40.825**	40.779
1	31.624	**35.938**	34.370	35.862
2	32.167	39.416	**39.808**	39.666
3	27.757	34.052	35.616	**35.641**
4	34.548	40.982	40.920	**41.938**
5	31.943	36.055	36.485	**36.923**
6	32.995	38.463	**40.084**	39.956
7	34.169	40.389	40.016	**41.037**
8	33.385	39.696	40.299	**40.401**
Mean	32.396	38.311	38.714	**39.134**

Table 8: PSNR values produced by loss functions (AM)

Mitochondria	Input	Laplace	FFT	BP
0	0.805	0.977	0.982	**0.983**
1	0.907	0.978	**0.984**	0.977
2	0.797	0.977	**0.981**	0.980
3	0.605	0.942	0.953	**0.955**
4	0.886	0.985	**0.987**	0.987
5	0.844	0.974	0.977	**0.978**
6	0.830	0.971	**0.982**	0.980
7	0.895	0.983	**0.985**	0.984
8	0.844	0.974	**0.983**	0.982
Mean	0.824	0.973	**0.979**	0.979

Table 9: SSIM values produced by loss functions (AM)

Restoration of dendra

Dendra imaged at 10 milliseconds were restored using the standard CARE model trained with dendra images. The dendra samples were imaged at 200 timesteps, with each timestep lasting 400 milliseconds. The following table displays average PSNR and SSIM results of denoising using the Laplace, FFT, and bandpass loss functions.

	Input	Laplace	FFT	BP
PSNR	26.639	32.691	28.534	32.758
SSIM	0.570	0.897	0.846	0.909

Restoration of actin sequence

Actin imaged at 10 milliseconds were restored using the standard CARE model trained with actin images. The actin samples were imaged at 200 timesteps, with each timestep lasting 400 milliseconds. Due to the large number of samples restored (200 images), the following table displays average PSNR and SSIM results of denoising using the Laplace, FFT, and bandpass loss functions.

	Input	Laplace	FFT	BP
PSNR	25.672	25.917	26.023	26.002
SSIM	0.837	0.872	0.874	0.874

Table 10: Average PSNR and SSIM values produced by loss functions (AAS)

Restoration of extremely noisy actin

In our AAE experiment, actin imaged at 1 millisecond were restored using the standard CARE model trained with actin images. Microscope settings were altered prior to imaging these actin samples to induce significant noise in these images (taken with 1 millisecond of exposure). The accompanying table that displays these results will be included in a future revision.

Discussion

The results of our study were analyzed using paired sample t-tests with an alpha significance value of 0.05. We demonstrate statistically significant results with respect to SSIM measurements, particularly in model mismatch experiments. The sequence of actin restorations (AAS) also demonstrated statistically significant results. The table below summarizes t-test results (p-values) for each experiment. The PSNR and SSIM values obtained using the FFT and bandpass loss functions were each evaluated against PSNR and SSIM values obtained by the original Laplace loss function.

Experiment	FFT/Laplace		BP/Laplace	
	PSNR	SSIM	PSNR	SSIM
MM	0.03	0.150	0.264	0.426
AA	0.110	0.201	0.640	0.710
MA	0.016	0.007	0.311	0.311
AM	0.260	0.0008	0.002	0.008
DDS	—	—	—	—
AAS	1.509e-232	5.811e-205	4.055e-232	5.458e-199
AAE	—	—	—	—

The following table summarizes the average change in PSNR and SSIM values organized by experiment description and loss function.

Metric	Loss	MM	AA	MA	AM	DD
$\Delta PSNR$	Laplace	6.405	8.336	4.474	6.015	DD
	FFT	6.870	7.539	5.519	6.418	–
	BP	6.039	8.434	3.819	6.838	–
$\Delta SSIM$	Laplace	0.157	0.166	0.157	0.150	DD
	FFT	0.158	0.165	0.167	0.156	–
	BP	0.157	0.169	0.154	0.155	–

Table 11: Average Increase in PSNR and SSIM

Further Research

To further evaluate our deep learning approach, we will employ traditional filtering techniques (total variation and non-local means) and compare our results with these conventional practices using standard metrics (PSNR and SSIM). The total variation minimization (TV) technique restores images by minimizing a cost function. The TV method smooths excess details while maintaining sharp edges. The non-local means (NL means) filtering technique finds the average of all pixels in an image, and makes each pixel a linear combination of patches. Similar patches are weighted more heavily than dissimilar patches (Buades, Coll, and Morel 2005).

In the near future, we will train the CARE network while varying the patch size of the training images to 32 x 32 and 128 x 128. By introducing different patch sizes for training, the CARE network performance may improve. We would also like to implement combined loss functions (FFT loss combined with bandpass loss). Perhaps an adaptive, GAN-based loss function may better outperform the current state-of-the-art. To test the limits of our computational restoration method, we will conduct future studies to determine what is the lowest amount of light that can be used when imaging a sample such that the image can successfully be restored by our implementation?

Conclusion

The purpose and major contribution of this research is to modify and improve existing restoration methods for fluorescence microscopy imaging. Compared to the Laplace loss function, the results of this study indicate that there were statistically significant improvements in image denoising using FFT loss and bandpass loss to train the CARE network. Our model mismatch and actin sequence restoration experiments yielded the most prominent statistically significant results, which confirms that the CARE model generalizes poorly when it is content-unaware. By developing ways to denoise fluorescence microscopy images faithfully, significantly less time and resources will be required to image 2D structures.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 1659788. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Agostinelli, F.; Anderson, M. R.; and Lee, H. 2013. Adaptive multi-column deep neural networks with application to robust image denoising. In *Advances in Neural Information Processing Systems*, 1493–1501.

Buades, A.; Coll, B.; and Morel, J.-M. 2005. A non-local algorithm for image denoising. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005.*, volume 2, 60–65. IEEE.

Burger, H. C.; Schuler, C. J.; and Harmeling, S. 2012. Image denoising: Can plain neural networks compete with bm3d? In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 2392–2399. IEEE.

Chen, C.; Chen, Q.; Xu, J.; and Koltun, V. 2018. Learning to see in the dark. *arXiv preprint arXiv:1805.01934*.

Conti, F. L.; Minucci, G.; and Derakhshan, N. 2017. A regularized deep learning approach for image de-blurring.

Hauberg, S.; Freifeld, O.; Larsen, A. B. L.; Fisher, J.; and Hansen, L. 2016. Dreaming more data: Class-dependent distributions over diffeomorphisms for learned data augmentation. In *Artificial Intelligence and Statistics*, 342–350.

Jain, V., and Seung, S. 2009. Natural image denoising with convolutional networks. In *Advances in Neural Information Processing Systems*, 769–776.

Paulin, M.; Revaud, J.; Harchaoui, Z.; Perronnin, F.; and Schmid, C. 2014. Transformation pursuit for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, 3646–3653. IEEE.

Weigert, M.; Schmidt, U.; Boothe, T.; Andreas, M.; Dibrov, A.; Jain, A.; Wilhelm, B.; Schmidt, D.; Broaddus, C.; Culley, S.; et al. 2017. Content-aware image restoration: Pushing the limits of fluorescence microscopy. *bioRxiv* 236463.

Xing, F.; Xie, Y.; Su, H.; Liu, F.; and Yang, L. 2017. Deep learning in microscopy image analysis: A survey. *IEEE Transactions on Neural Networks and Learning Systems* 119.

Xu, L.; Ren, J. S.; Liu, C.; and Jia, J. 2014. Deep convolutional neural network for image deconvolution. In *Advances in Neural Information Processing Systems*, 1790–1798.

Zhao, H.; Gallo, O.; Frosio, I.; and Kautz, J. 2017. Loss functions for image restoration with neural networks. *IEEE Transactions on Computational Imaging* 3(1):47–57.

Estimating Depth in Cylindrical Panoramas

Lee Sharma
University of Maryland, University College
Adelphi, MD
lee@leesharma.com

Jonathan Ventura
University of Colorado, Colorado Springs
Colorado Springs, CO
jventura@uccs.edu

Abstract

Predicting 3D scene structure from a single image is a significant obstacle in fields such as autonomous robotic navigation, entertainment, and 3D modelling. In the past several years, researchers have made promising strides in predicting 3D scenes from flat perspective images; however, little work has been done towards applying this to panoramic imagery. In this project, we develop a model for estimating depth and ego-motion from single cylindrical panoramic images.

Introduction

Understanding the structure of a 3D scene is an important problem in many fields, from autonomous vehicle navigation to 3D filming. Unfortunately, predicting 3D structure from a single image is extremely challenging. The number of confounding factors (e.g. varied texture, lighting, occlusions, and object movement) make it an ill-posed problem: a single image could represent many possible 3D scenes.

Early attempts at estimating scene structure from motion (also known as SfM) focused on directly analyzing factors such as the geometry and flow of the image (Bergen et al. 1992; Mur-Artal, Montiel, and Tardos 2015; Saxena, Sun, and Ng 2009). However, these models were often fragile in the face of occlusions, object motion, and other inconsistent (but real-world) conditions. In the past several years, many exciting advances have been made in estimating scene structure and ego-motion—motion of the observer—using deep neural networks.

Early research relied on labelled data for training (Eigen, Puhrsch, and Fergus 2014; Liu et al. 2016). Unfortunately, labelled 3D footage is expensive to create, limiting the quantity and diversity of available training data. This limitation has triggered a promising new area of research: unsupervised SfM models, which figure out scene attributes such as depth and ego-motion without requiring labelled data. In the past few years, several unsupervised models have been proposed with comparable performance to the supervised state-of-the-art (Godard, Mac Aodha, and Brostow 2017; Zhou et al. 2017; Vijayanarasimhan et al. 2017; Mahjourian, Wicke, and Angelova 2018; Wang et al. 2018), lowering the cost and expanding the diversity of potential training datasets.

While much progress has been made for ”standard” pinhole perspective images, little work has been done for other projection models. There are many compelling applications for computer vision with non-pinhole projection images, such as robotic vision with omnidirectional cameras. Spherical panoramas are particularly interesting, but the projection model introduces distortions that prevent a naive approach (Cohen et al. 2018). Cylindrical panoramas offer many of the same benefits as spherical, but the projection model is much more straightforward. However, to our knowledge, deep networks for depth prediction have not yet been applied to cylindrical panoramic imagery.

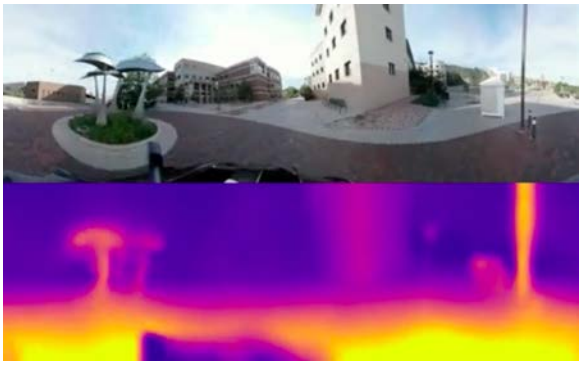


Figure 1: An example of extracting 3D depth information from a 2D image.

In this paper, we tackle that gap by proposing an unsupervised convolutional model that estimates depth and ego-motion from cylindrical panoramas.

This research was motivated by the following two questions:

1. How can existing convolutional neural networks (CNNs) designed for pinhole images be adapted to take cylindrical input?
2. Can cylindrical input improve the performance in unsupervised structure-from-motion models?

Major Contributions

This work has two major contributions:

1. We present CylindricalSfMLearner, an unsupervised model for estimating structure from motion given cylindrical panoramic input, and

2. We present a library and techniques for extending other CNNs to take cylindrical panoramic input.

We hope that our work here will provide a solid base for future research into CNNs and panoramic input.

Related Work and Background

Supervised Monocular Depth Prediction

Early research focused on detecting structure from *stereo*—or multi-source—imagery. Stereo SfM is much more constrained than detecting structure from *monocular*—single-source—input, but the stereo input requirements limits the model's flexibility. Eigen, Puhrsch, and Fergus (2014) proposed a different approach using deep neural networks. They presented a supervised model for estimating depth maps from monocular input images. Their model was composed of two stacks, one for coarse estimation and one for fine estimation, and joined the two predictions (Eigen, Puhrsch, and Fergus 2014).

Supervised to Unsupervised Models

While this progress in supervised models was exciting, collecting labelled footage is very expensive, increasing training cost and limiting the size and diversity of datasets. This limitation triggered a number of researcher to turn towards unsupervised models. Godard, Mac Aodha, and Brostow (2017), taking inspiration from previous stereo techniques, proposed a model that was trained on unlabelled stereo footage. Their trained model outperformed the previous supervised state-of-the-art on urban scenes and reasonably well on unrelated datasets (Godard, Mac Aodha, and Brostow 2017).

Zhou et al. (2017) removed the constraint of stereo training footage. They proposed an unsupervised model composed of jointly-trained depth and pose CNNs using a loss function tied to novel view synthesis. They found that their unsupervised model performed comparably to supervised models on the known datasets and reasonably well when tested against a completely unknown data set. Unfortunately, while the model could be trained on monocular footage, it assumes a given camera calibration, which prevents random footage from the web from being used as training data (Zhou et al. 2017).

Further Enhancements

In a concurrent study, Vijayanarasimhan et al. (2017) addressed this shortcoming by explicitly modelling scene geometry. Inspired by geometrically-constrained Simultaneous Localization and Mapping (SLAM) models and Godard, Mac Aodha, and Brostow's work on left-right consistency, they proposed a model capable of detecting both ego-motion and object motion—as well as depth and object segmentation—from uncalibrated monocular images (Vijayanarasimhan et al. 2017). Building upon Zhou's and Vijayanarasimhan's models, Mahjourian, Wicke, and Angelova (2018) proposed a completely unsupervised model with explicit geometric scene modelling the following year. Their model introduced a new 3D loss function and added a new principled mask for handling unexplainable input (Mahjourian, Wicke, and Angelova 2018).

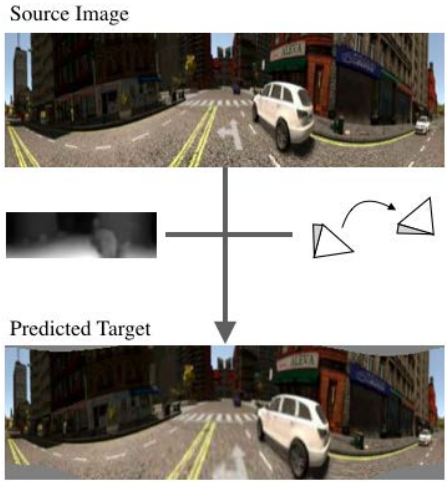


Figure 2: SfMLearner uses view synthesis as a supervisor: the source image, depth, and pose transformation are used to synthesize a target view, and the loss depends on the disparity. As the synthesized view improves, the depth and pose predictions improve.

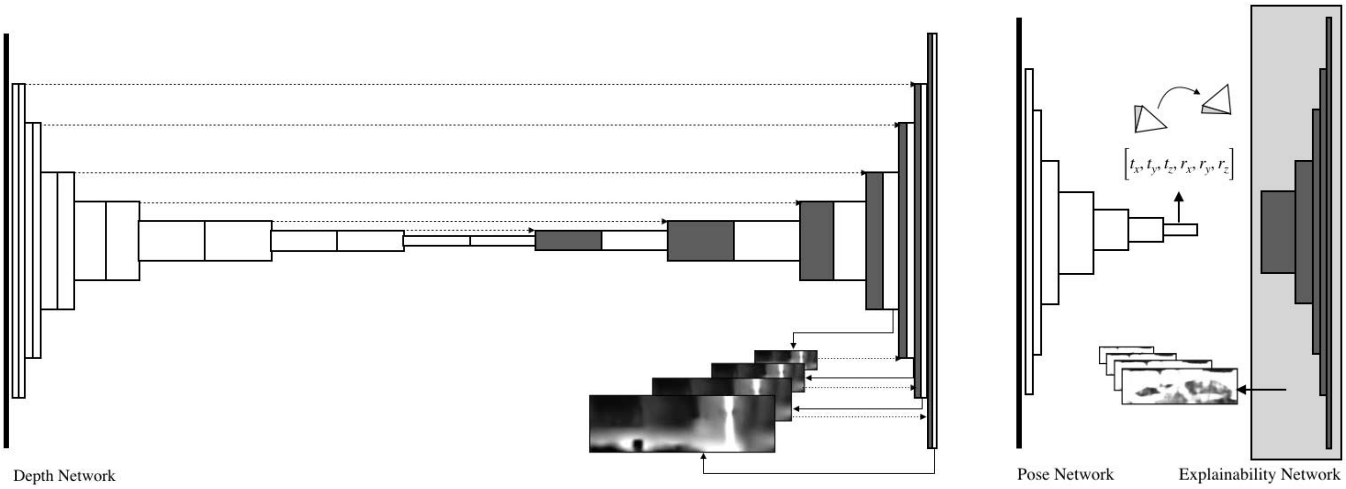
Methods

In this project, we present an unsupervised convolutional model that estimates (a) the depth map from a single cylindrical panoramic image and (b) ego-motion from image sequences.

Model Architecture

Our architecture is based off of SfMLearner (Zhou et al., 2017), an unsupervised model designed to predict depth and ego-motion in monocular pinhole images. The architecture, illustrated in Figure 3, is a convolutional network consisting of two jointly-trained stacks: (a) a depth network to estimate the depth map, (b) a pose network/explainability mask to estimate the change in pose in image sequences and handle unexplainable input. The depth network follows the DispNet (Mayer et al. 2016) skip-layer architecture, with seven contracting layers and seven expanding layers, outputting a multi-scale depth prediction. The pose network (PoseExpNet) consists of five contracting convolutional layers and three pose layers, outputting the predicted translation and rotation between source and target views. The explainability network consists of a final five upconvolution layers and returns a multi-scale explainability mask, which masks "unexplainable" motion.

SfMLearner works using *view synthesis*—the prediction of a target frame given source frames—as an internal supervisor. At each step, the joint DispNet and PoseExpNet stacks predict the depth of a target frame and the pose difference between the target and source frames. The depth and pose are then used for synthesizing a target view: as the predicted view improves the depth and pose predictions also improve



(a) The depth network consists of a seven layer encoder followed by a seven layer decoder with a skip-layer architecture. The network returns the multi-scale disparity predictions, which can then be converted to depth predictions.

(b) The pose network consists of a five-layer encoder followed by two pose layers and returns the pose as a six element vector. The explainability network adds a five-layer decoder and returns the multi-scale explainability mask.

Figure 3: The SfMLearner model consists of two jointly-trained CNN stacks. The left diagram shows the depth CNN, and the right diagram shows the pose/explainability CNN.

(Zhou et al. 2017). This process is illustrated in Figure 2.

For this model, we use a three-part objective function. The main component is the *photometric loss* ($\mathcal{L}_{\text{pixel}}$), which minimizes the difference between synthesized views and the target view, which, in turn, improves the depth and pose predictions. This is regularized by the *smooth loss* ($\mathcal{L}_{\text{smooth}}$), which minimizes the second derivatives with respect to the depth and the *explainability loss* ($\mathcal{L}_{\text{exp}}$), which makes the model more resilient to anomalous input (e.g. moving objects). If λ_s and λ_e represent the smooth and explainability weights, the total loss can be written as follows:

$$\mathcal{L} = \sum_{\text{scales}} \left(\sum_{\text{sources}} \mathcal{L}_{\text{pixel}} + \lambda_s \mathcal{L}_{\text{smooth}} + \sum_{\text{sources}} \lambda_e \mathcal{L}_{\text{exp}} \right) \quad (1)$$

If $\mathcal{I}$ is an RGB image, $\mathcal{D}$ is the depth prediction, and e is the explainability mask, the three loss components can be written as follows:

$$\mathcal{L}_{\text{pixel}} = \frac{\sum e \left| \mathcal{I}_{\text{proj}} - \mathcal{I}_{\text{target}} \right|}{\left\| \mathcal{I}_{\text{target}} \right\|} \quad (2)$$

$$\mathcal{L}_{\text{smooth}} = \left| \frac{\delta^2 \mathcal{D}}{\delta x^2} \right| + \left| \frac{\delta^2 \mathcal{D}}{\delta x \delta y} \right| + \left| \frac{\delta^2 \mathcal{D}}{\delta y \delta x} \right| + \left| \frac{\delta^2 \mathcal{D}}{\delta y^2} \right| \quad (3)$$

$$\mathcal{L}_{\text{exp}} = \sum \text{softmax}\left(\mathcal{E}\right) \quad (4)$$

Two major modifications were required to allow for cylindrical input: (a) the intrinsics and view synthesis functions were modified to account for cylindrical projection, and (b) the convolutional layers, resampling functions, and loss were modified to preserve horizontal wrapping.

Camera Projection and Cylindrical Panoramas Our view synthesis function works by (a) projecting a source image onto the 3D sensor coordinate system, (b) inverse warping the points from the source pose to target pose, and (c) projecting the warped points back onto a 2D image plane. To adapt this process to work with cylindrical input, we modified the projection functions between the pixel and camera coordinate systems as well as the expected camera intrinsics.

Most structure-from-motion systems expect *pinhole projection* images as input. Pinhole projection images project a 3D scene from the world coordinate system onto a flat image plane; this process can be described by the focal length f, principle point c, the image plan height H, and the image plane width W, as shown in Figure 4.

In contrast, *cylindrical projection* projects the 3D world onto a curved cylindrical surface, as seen in Figure 5. The goal of this process is to take a 3D point in the world coordinate system and project it onto a rectangular cylindrical panorama. This requires projecting the 3D point onto the cylindrical image surface and converting the image surface into a Cartesian coordinate system.

The transformation between the sensor and pixel coordinate systems can be described by the following equations: let $P = (x_s, y_s, z_s)$ represent a point in the 3D sensor coordinate system $X_s Y_s Z_s$. We can find point $Q = (r, \theta, h)$, the point where P is projected onto a unit cylinder around the origin, with the following formula:

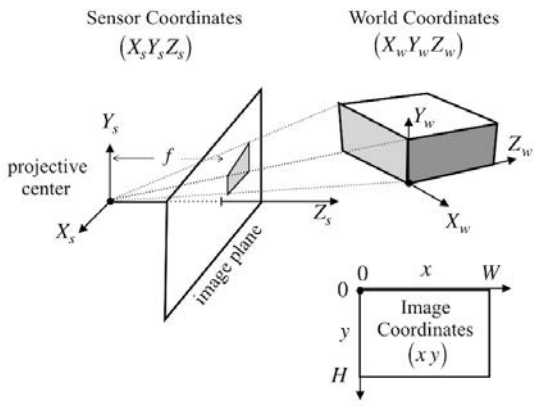


Figure 4: Pinhole projection model. The 3D world (on the world coordinate system) is projected on a flat image plane; the image plane is a focal length f away from the projective center along the Z_s axis (on the sensor coordinate system). The result is an $H \times W$ rectangular image.

$$\begin{bmatrix} r \\ \theta \\ h \end{bmatrix} = \begin{bmatrix} 1 \\ \arctan\left(\frac{x_s}{z_s}\right) \\ \frac{y_s}{\sqrt{x_s^2 + z_s^2}} \end{bmatrix} \tag{5}$$

This can be represented by the 2D point $q = (\theta, h)$.

To "unroll" this cylinder into a rectangular image with the Cartesian system xy, we must use some camera intrinsics: scaling factor f; output width W; output height H; principle point (c_x, c_y). The following equations will convert $q = (\theta, h)$ into $p = (x, y)$:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{\theta}{2\pi} W + c_x \\ fH + c_y \end{bmatrix} \tag{6}$$

The final panoramic image has a fixed height H and a width of W (representing the full 360° view), as seen in Figure 5.

The reverse projection ($P = (x_s, y_s, z_s)$ given $q = (\theta, h)$ and some scaling factor d) can be described as follows:

$$\begin{bmatrix} x_s \\ y_s \\ z_s \end{bmatrix} = \begin{bmatrix} d \sin\theta \\ dh \\ d\cos\theta \end{bmatrix} \tag{7}$$

The cylindrical intrinsics $\mathcal{K}$ can be represented by the following matrix:

$$\mathcal{K} = \begin{bmatrix} f_\theta & 0. & c_\theta \\ 0. & f_h & c_h \\ 0. & 0. & 1. \end{bmatrix} \tag{8}$$

Let θ represent the field of view, x represent the horizontal pixel position, h represent the height coverage, and y represent the vertical pixel position. Then we can find the intrinsics values with the following formulas:

$$f_\theta = \frac{\theta_0 x_1 - x_0 \theta_1}{\theta_0 - \theta_1} \tag{9}$$

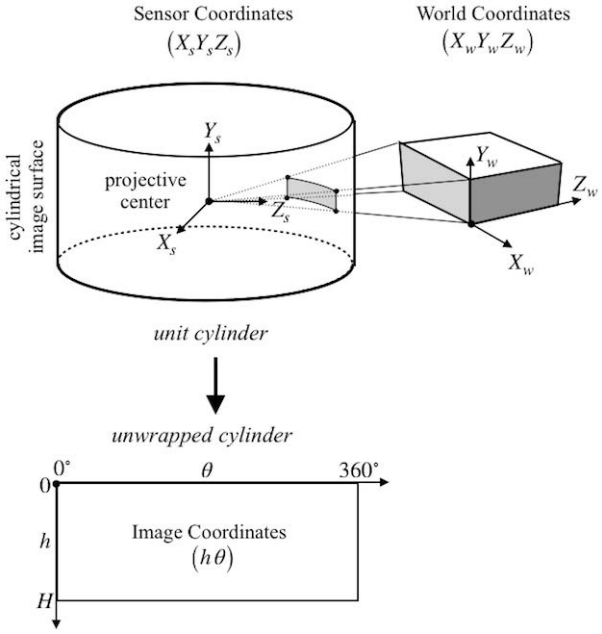


Figure 5: Cylindrical projection model. In contrast with pinhole projection, this projects an image onto a curved cylindrical surface. The final result is a rectangular image with height H and a width representing the full 360°.

$$c_\theta = \frac{x_0 - x_1}{\theta_0 - \theta_1} \tag{10}$$

$$f_h = \frac{h_0 y_1 - y_0 h_1}{h_0 - h_1} \tag{11}$$

$$c_h = \frac{y_0 - y_1}{h_0 - h_1} \tag{12}$$

Horizontal Wrapping In order to extend the model for cylindrical panoramic images, we needed to modify the convolutional layers, smooth loss function, and 2D projection to account for horizontal wrapping.

In a cylindrical image, the left and right side of the input image are adjacent. For a convolutional layer to work with cylindrical input, it must preserve this horizontal wrapping property. This can be done by padding the right side of the tensor with columns from the left and vice-versa. An example of a standard convolutional operation vs. a cylindrical convolutional operation can be seen in Figure 6.

We accomplished this by writing a library extending `tensorflow.contrib.slim`—a tensorflow library providing convenience functions for common operations—for cylindrical input. Our library is intended as a drop-in replacement for `slim`, so our functions follow the same interface as their corresponding `slim` functions. We also added simple wrapping functions to the smooth gradient loss, ensuring the depth is smooth between the two sides of the unwrapped depth image, and the bilinear sampler, ensuring that the synthesized views take the full 360° into account.

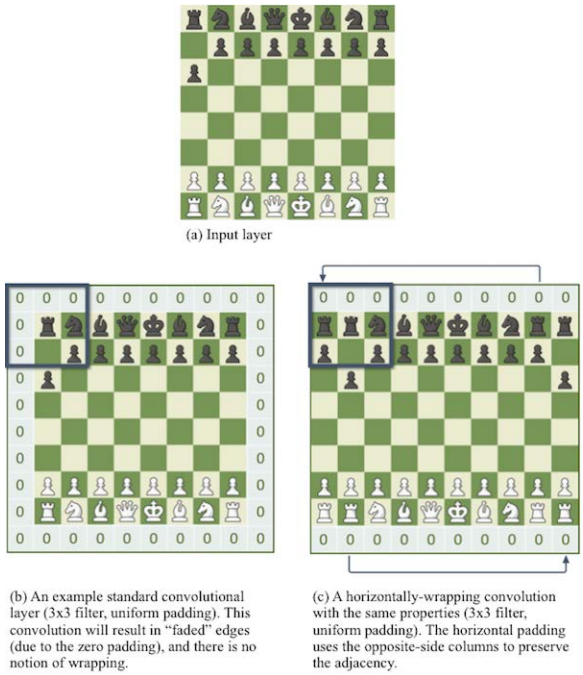


Figure 6: An example of a standard convolutional layer vs. horizontal wrapping convolutional layer. This figure shows an 8×8 matrix with a 3×3 kernel. To represent horizontal wrapping, the leftmost column is appended to the right side of the matrix during the convolution and vice-versa. Adapted from chess.com.

Experiments

Datasets

SYNTHIA-Seq We use the SYNTHIA-Seq dataset as our primary experimental dataset. SYNTHIA-Seq is a large synthetic dataset of driving scenes in a virtual city designed to be comparable to KITTI or CityScapes—the major benchmarks for structure-from-motion problems (Ros et al. 2016). SYNTHIA-Seqs contains both groundtruth depth information and sufficient sensor information to construct 360° cylindrical panoramas, making it a good fit for this research. Some sample frames can be seen in Figure 7.

To prepare the dataset, we first stitched the pinhole images and depth groundtruths from each camera into 360° panoramas and calculated the modified camera intrinsics. The images were finally formatted into three-image sequences using the same technique as Zhou et al. (2017). For these experiments, we used a subset of SEQS-02 (summer city driving scenes) consisting of 2,042 frames with a 90/10 training-testing split.

Headcam One of the strongest motivations for unsupervised structure-from-motion research is the availability and diversity of amateur footage online. In order to evaluate our model's performance on these informal datasets, we created our own panoramic dataset consisting of several hours of footage around Colorado Springs. Footage was collected with a Samsung Gear 360 (2016) camera mounted on a standard bicycle helmet. Some sample frames from this dataset can be seen in Figure 8.

We prepared this dataset by first stitched the footage into a spherical panorama using Samsung software, then splitting the footage into frames at 10 fps, and finally unwarping the images into cylindrical panoramas.

Finally, frames were concatenated into 3-frame sequences using a similar technique as Zhou et al. (2017). For the experiments in this paper, we used a subset of this dataset from the University of Colorado: Colorado Springs campus consisting of 5,202 frame sequences (90/10 training-testing split).

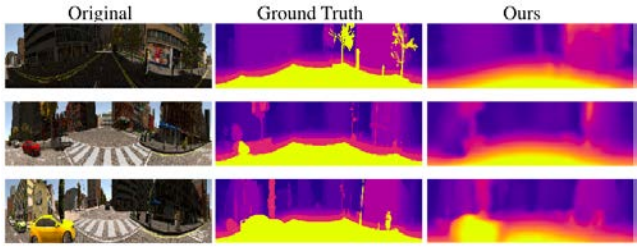


Figure 7: A sample of depth predictions and groundtruth depth for SYNTHIA-Seq (testing split).

Evaluation of Cylindrical Depth Prediction

We trained and evaluated CylindricalSfMLearner's performance on the SYNTHIA-Seq dataset to determine the basic effectiveness of our model; a sample of our predicted depth compared with the groundtruth can be found in Figure 7. To the best of our knowledge, there are no other attempts at depth prediction for cylindrical panoramic imagery, so we have elected to report on the same metrics common in pinhole depth prediction (Zhou et al. 2017; Garg et al. 2016; Mahjourian, Wicke, and Angelova 2018; Eigen, Puhrsch, and Fergus 2014). Table 1 shows our model's disparity and accuracy results; we also include recent results for pinhole depth prediction models on the KITTI dataset for context, reprinted from Mahjourian, Wicke, and Angelova (2018).

While our model is able to successfully handle scenery such as buildings, trees, and the overall scene depth, it often made mistakes on empty sections of road and moving vehicles. While we were not able to address these problems in the scope of this project, we suspect that this issue could be mitigated by altering smoothing and explainability weights, and we hope to resolve this in future work.

Evaluation on the Headcam Dataset

In order to evaluate CylindricalSfMLearner's performance on informal datasets, we trained our model on Headcam data and generated depth predictions for our test split. A visual comparison of the input RGB image and the predicted depth can be seen in Figure 8. The visible helmet and the rider's shadow cause some visible distortion, but objects such as buildings, trees, lightpoles, and statues are recognizable in the depth predictions.

				Disparity (lower is better)				Accuracy (higher is better)		
Method	Supervision	Dataset	Cap	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Ours (full 360°)	-	**S**	**80m**	**0.425**	**3.115**	**5.754**	**0.467**	**0.540**	**0.808**	**0.9052**
Eigen et al. Coarse	Depth	K	80m	0.214	1.605	6.563	0.292	0.673	0.884	0.957
Eigen et al. Fine	Depth	K	80m	0.203	1.548	6.307	0.282	0.702	0.890	0.958
Liu et al.	Depth	K	80m	0.201	1.584	6.471	0.273	0.68	0.898	0.967
Zhou et al.	-	K	80m	0.208	1.768	6.856	0.283	0.678	0.885	0.957
Mahjourian et al.	-	K	80m	0.163	1.240	6.220	0.250	0.762	0.916	0.968

Table 1: Our depth prediction performance on SYNTHIA-Seq (bolded) shown alongside the state-of-the-art results on KITTI using pinhole projection. In the dataset column, S means SYNTHIA-Seq (Ros et al. 2016) and K means KITTI (Geiger, Lenz, and Urtasun 2012). Results reprinted from Mahjourian, Wicke, and Angelova; Zhou et al.; Liu et al.; Eigen, Puhrsch, and Fergus (2018; 2017; 2016; 2014)

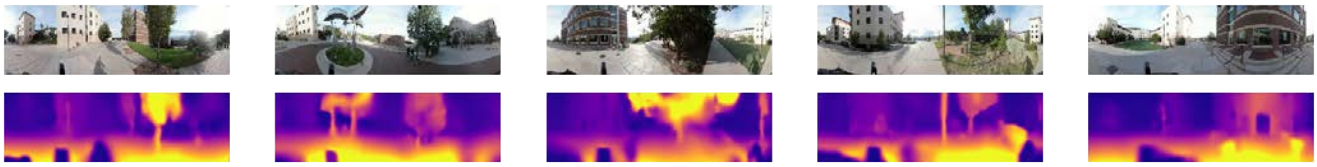


Figure 8: A sample of depth predictions for our self-collected headcam dataset (testing split).

Disparity (lower is better)

Method	Abs Rel	Sq Rel	RMSE	RMSE log
Wrapping	**0.425**	3.115	**5.754**	**0.467**
No wrapping	0.430	**2.988**	5.960	0.476

Accuracy (higher is better)

Method	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Wrapping	**0.540**	**0.809**	**0.905**
No wrapping	0.517	0.793	0.898

Table 2: Comparison between the model trained with and without horizontal wrapping. The models were trained for 50,000 steps with the following settings: learning rate $\alpha = 0.0001$, batch size $= 8$, smooth weight $\lambda_s = 0.4$, explainability weight $\lambda_e = 0.1$, and sequence length $= 3$. The model with wrapping performs consistently better than the model without.

Evaluation of Horizontal Wrapping

Finally, in order to demonstrate the significance of our modifications, we evaluated two models with and without horizontal wrapping. Each model was trained for 50,000 steps on our training subset of the SYNTHIA-Seqs dataset; the results can be seen in Table 2 and Figure 9. The model with horizontal wrapping outperformed the model without on nearly every metric.

Conclusion

In this paper, we have demonstrated that an existing unsupervised structure-from-motion architecture can be adapted for cylindrical panoramic imagery by altering the projection model and implementing cylindrical wrapping.

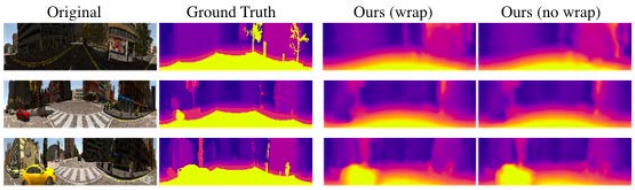


Figure 9: Comparison of depth predictions with and without wrapping (testing split).

Challenges and Limitations

The novelty of this project is also its biggest obstacle: there is very little structure from motion research using cylindrical input. Unlike with pinhole projection input, there is no standard structure-from-motion dataset like KITTI (Geiger, Lenz, and Urtasun 2012) or CityScapes (Cordts et al. 2016). As a result, it has been challenging to find datasets with all of the required features: 360° panoramic frame sequences, depth ground truth, relevance to the expected application domains, and an acceptable quality and measurement accuracy.

A related challenge is contextualizing our results. Since (to our knowledge) no other research has attempted to estimate general structure from motion in cylindrical imagery, it is difficult to benchmark against previous research.

Further Research

While we have answered our first research question, our second—"Can cylindrical input improve the performance in unsupervised structure-from-motion models?"—remains unanswered. Our current model serves as a proof-of-concept: the results demonstrate that CNNs can be modified for cylindrical input; however, there are numerous pinhole projection SfM models that report better performance.

With that in mind, we have several goals for future work: (a) we would like to tune our model and experiments to better evaluate the strengths/weaknesses of cylindrical imagery for depth estimation, and (b) we would like to to fine-tune our hyperparameters and implement some new state-of-the-art techniques. Specifically, we would like to explore ways in which traditional SLAM and SfM methods could be used to improve performance, such as the geometric loss and refined smooth loss implemented by Mahjourian, Wicke, and Angelova; Yin and Shi (2018; 2018) and the direct pose methods implemented by Wang et al. (2018). Aside from our base performance, benchmarking was a major challenge in this project. We hope to improve our experiments to better evaluate the strengths and limitations of cylindrical input.

It would also be interesting to experiment with changing the model architecture, such as the ResNet architecture proposed by Yin and Shi (2018) or the multi-scale attention back-end proposed by Xu et al. (2018); a different architecture may show better performance with the cylindrical projection model. Finally, we hope to apply our techniques for converting from perspective to cylindrical panoramic input to another convolutional model (e.g. object detection or object motion).

Beyond our immediate goals, there are many interesting applications for cylindrical SfM models. One exciting application could be to generate stereo binocular images from panoramic input. There are thousands of hours of monocular panoramic footage available online from panoramic phone cameras, VR video cameras, and other similar devices; stereo footage generation would allow people to experience this in 3D though VR headsets, which could be useful in videoconferencing, entertainment, and training. Another possibility is adapting the model to handle spherical panoramic images. This is a more challenging problem (Cohen et al. 2018), but it has major implications for problems such as self-driving cars.

Acknowledgements

We'd like to thank Chance Hamilton, Gia Zhuang, Kayleigh Migdol, and T.A. Nguyen for their feedback and helpful conversations throughout this project. This material is based upon work supported by the National Science Foundation under Grant No. 1659788. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Bergen, J. R.; Anandan, P.; Hanna, K. J.; and Hingorani, R. 1992. Hierarchical model-based motion estimation. In *European Conference on Computer Vision*, 237–252. Springer.

Cohen, T. S.; Geiger, M.; Koehler, J.; and Welling, M. 2018. Spherical CNNs. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*.

Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; and Schiele, B. 2016. The cityscapes dataset for semantic urban scene understanding. In *IEEE CVPR*, 3213–3223.

Eigen, D.; Puhrsch, C.; and Fergus, R. 2014. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, 2366–2374.

Garg, R.; BG, V. K.; Carneiro, G.; and Reid, I. 2016. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*, 740–756. Springer.

Geiger, A.; Lenz, P.; and Urtasun, R. 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. In *IEEE CVPR*, 3354–3361.

Godard, C.; Mac Aodha, O.; and Brostow, G. J. 2017. Unsupervised monocular depth estimation with left-right consistency. In *IEEE CVPR*, volume 2, 7.

Liu, F.; Shen, C.; Lin, G.; and Reid, I. 2016. Learning depth from single monocular images using deep convolutional neural fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38(10):2024–2039.

Mahjourian, R.; Wicke, M.; and Angelova, A. 2018. Unsupervised Learning of Depth and Ego-Motion from Monocular Video Using 3d Geometric Constraints. In *IEEE CVPR*.

Mayer, N.; Ilg, E.; Hausser, P.; Fischer, P.; Cremers, D.; Dosovitskiy, A.; and Brox, T. 2016. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE CVPR*, 4040–4048.

Mur-Artal, R.; Montiel, J. M. M.; and Tardos, J. D. 2015. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics* 31(5):1147–1163.

Ros, G.; Sellart, L.; Materzynska, J.; Vazquez, D.; and Lopez, A. M. 2016. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *The IEEE CVPR*.

Saxena, A.; Sun, M.; and Ng, A. Y. 2009. Make3d: Learning 3d scene structure from a single still image. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31(5):824–840.

Vijayanarasimhan, S.; Ricco, S.; Schmid, C.; Sukthankar, R.; and Fragkiadaki, K. 2017. Sfm-net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*.

Wang, C.; Buenaposada, J. M.; Zhu, R.; and Lucey, S. 2018. Learning Depth from Monocular Videos using Direct Methods. In *IEEE CVPR*.

Xu, D.; Wang, W.; Tang, H.; Liu, H.; Sebe, N.; and Ricci, E. 2018. Structured attention guided convolutional neural fields for monocular depth estimation. In *IEEE CVPR*, 3917–3925.

Yin, Z., and Shi, J. 2018. GeoNet: Unsupervised learning of dense depth, optical flow and camera pose. In *IEEE CVPR*, volume 2.

Zhou, T.; Brown, M.; Snavely, N.; and Lowe, D. G. 2017. Unsupervised learning of depth and ego-motion from video. In *IEEE CVPR*, volume 2, 7.

Author Index