Proceedings of the Seminar

Machine Learning in Computer Vision and Natural Language Processing

University of Colorado, Colorado Springs

August 9, 2019

Editors: Jugal K. Kalita, Jonathan Ventura and Terrance Boult

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 2019 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 Dr. Guy Hagen, who was a faculty advisor for several of the REU students. We also thank Alessandra Langfels for working out all the financial and administrative details. We also thank our graduate and undergraduate students, in particular, Thomas Conley, Ahmed Bensaoud, Brandon Collins and Zanyar Zohourianshahzadi, for helping the students with ideas as well as systems and programming issues. Our gratitude to Ginger Boult for being the "REU Mom" and having the welfare of the REU interns at her heart all through the summer. Justin Johnson, Xian Tan and his team also deserve our sincere gratitude for making sure that the computing systems performed reliably during the summer.

Sincerely,

Jugal Kalita
jkalita@uccs.edu
Professor

Jonathan Ventura
jventu09@calpoly.edu
Assistant Professor

Terrance Boult
tboult@uccs.edu
El Pomar Professor of Security and Innovation

August 9, 2019

Table of Contents

NSF REU Seminar on Machine Learning
Department of Computer Science
University of Colorado, Colorado Springs
Osborne A343 UCCSTeach Room
August 9, 2019: Friday

10*:30-10:40 AM:* Introduction by Dr. Jugal Kalita, followed by Welcome Remarks by Dr. Donald Rabern, Dean of the College of Engineering, University of Colorado, Colorado Springs

10:40-*12:15 AM* Session Chair*:* Dr. Philip Brown, Assistant Professor in Computer Science, University of Colorado, Colorado Springs

- 10:40-11:05 Justin Leo, University of Colorado, Colorado Springs, CO: *Moving Towards Open Set Incremental Learning: Readily Discovering New Authors*
- 11:05-11:30 Brendan Bena, Drury University, Springfield, MO: *Injection of Creativity and Emotion-Elicitation in Poetry Generation*
- 11:30-11:55 Andrew Conley, Rensselaer Polytechnic Institute, Troy, NY: *Enhancing Language Models with Knowledge Graph Embeddings*

12:00-1:15 PM: Lunch with Drs. Thomas Christensen, Provost and Executive Vice Chancellor for Academic Affairs and Professor of Physics, and Jessi Smith, Associate Vice Chancellor for Research and Research Integrity Officer, both of the University of Colorado, Colorado Springs

1:10-1:15 PM: Welcome Back Remarks by Dr. Jessi Smith

1:15-2:55 PM Session Chair: Dr. Yanyan Zhuang, Assistant Professor of Computer Science, University of Colorado, Colorado Springs, CO

- 1:15-1:40 Joshua Frederick, California Polytechnic State University, San Luis Obispo, CA: *PixelMRF: A Deep Markov Random Field for Image Generation*
- 1:40-2:05 Sonia Rao, University of Georgia, Athens, GA: *Self-supervised Deep Learning for Fluorescence Microscopy Denoising*
- 2:05-2:30 Clare Minnerath, Providence College, Providence, RI: *Self-Supervised Learning for Single-Molecule Localization Microscopy Denoising*
- 2:30-2:55 Jonathan Schwan, University of Colorado, Colorado Springs, CO: *I-MOVE: Independent Moving Objects for Velocity Estimation*

2:55-3:10 PM: Break

3:10-4:25 PM Session Chair: Julian Medina, B.S. in Computer Science and former REU student, University of Colorado, Colorado Springs, CO

- 3:10-3:35 Sven Marnauzs, Boise State University, Boise, ID: *A Domain Independent Social Media Depression Detection Model*
- 3:35-4:00 Kaden Griffith, University of Colorado, Colorado Springs, CO: *Solving Arithmetic Word Problems Automatically Using Transformer and Unambiguous Representations*
- 4:00-4:25 Tiffany Chien, University of California, Berkeley, CA: *Adversarial Analysis of Natural Language Understanding Systems*

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

Our Session Chairs and Guests

Dr. Philip Brown is an Assistant Professor in the department of Computer Science at the University of Colorado, Colorado Springs. His areas of research are cyber-social systems, strategic aspects of security, and robust network games.

Dr. Thomas M. Christensen is Provost and Executive Vice Chancellor for Academic Affairs at the University of Colorado at Colorado Springs. He has served the campus as a faculty member, department chair, associate dean and dean. Dr. Christensen has received both the College and campus Outstanding Teaching Awards and the Chancellor's Award to recognize his service and teaching.

Julian Medina is a recent BS graduate of the University of Colorado, Colorado Springs in Computer Science. He participated in the REU program at UCCS in 2018. Julian published one paper based on his REU research at the IEEE International Conference on Machine Learning and Applications in 2018.

Dr. Donald Rabern is the new Dean of the College of Engineering and Applied Science at the University of Colorado, Colorado Springs. He was a professor of engineering at Fort Lewis College in Durango, Colorado, dean of engineering and professor of aerospace engineering at Embry-Riddle Aeronautical University in Prescott, Arizona, department chair and professor of civil engineering and engineering mechanics at Montana State University, before coming to UCCS. He also worked at the Los Alamos National Laboratory for more than 15 years on two occasions.

Dr. Jessi Smith is a Professor of Psychology and Associate Vice Chancellor for Research at the University of Colorado, Colorado Springs. Her areas of research are motivation, stereotype, gender, and research diversity.

Dr. Yanyan Zhuang is an Assistant Profesor in the Department of Computer Science at the University of Colorado, Colorado Springs. Her areas of research are cybersecurity, privacy, computer networking and software engineering. Yanyan's work on software code comprehension received an ACM SIGSOFT Distinguished Paper Award in 2018.

NSF REU Proposal Presentation Meeting

Department of Computer Science
University of Colorado, Colorado Springs
Engineering Building, Room 103
June 12, 2019: Wednesday

1:30-1:35 PM: Welcome Remarks by Dr. Thottam Kalkur, Professor of Electrical Engineering and Associate Dean, College of Engineering and Applied Science

1:40-2:40 PM
Session Chair: Dr. Terrance Boult, Department of Computer Science, University of Colorado, Colorado Springs, CO

Jonathan Schwan, University of Colorado, Colorado Springs, CO: Estimating Motion Parameters from Unconstrained Video

Sonia Rao, University of Georgia, Athens, GA: Self-supervised Deep Learning for Fluorescence Microscopy Denoising

Clare Minnerath, Providence College, Providence, RI: Self-supervised Learning for Single-molecule Localization Microscopy Denoising

Joshua Frederick, California Polytechnic State University, San Luis Obispo, CA: PixelMRF: A Deep Markov Random Field for Image Generation

2:55-3:55 PM
Session Chair: Steve Cruz, Department of Computer Science, University of Colorado, Colorado Springs, CO

Tiffany Chien, University of California, Berkeley, CA: Adversarial Analysis of Natural Language Understanding Systems

Kaden Griffith, University of Colorado, Colorado Springs, CO: Using Unambiguous Intermediate Representation to Solve Arithmetic Word Problems

Sven Marnauzs, Boise State University, Boise, ID: A Domain Independent Social Media Depression Detection Model

4:05-4:50 PM
Session Chair: Akshay Dhamija, University of Colorado, Colorado Springs, CO

Andrew Conley, Rensselaer Polytechnic Institute, Troy, NY: Using Graph Embedding for Natural Language Inference

Brendan Bena, Drury University, Springfield, MO: Emotion Eliciting Poetry Generation

Justin Leo, University of Colorado, Colorado Springs, CO: Moving Towards an Incremental Learning Model

Our Session Chairs

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.

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 are in Computer Vision and Machine Learning, specific areas include Open-Set Recognition, Face Recognition, Image Forensics, and Incremental Learning. He has been an author on 2 published papers within the last year and has 1 under review. Something interesting - he has taught at UCCS and enjoys snowboarding.

Akshay Dhamija is a PhD student at the University of Colorado, Colorado Springs. His research focuses on Deep Learning for Computer Vision. Akshay is keenly interested in applications of computer vision algorithms to real world scenarios.

NSF REU Midsummer Meeting

Department of Computer Science
University of Colorado, Colorado Springs
Engineering Building, Room 103
July 12, 2019: Wednesday

1:30-1:35 PM: Welcome Remarks by Dr. Xiaobo Zhou, Interim Dean of the College of Engineering and Applied Science, University of Colorado, Colorado Springs.

1:40-2:40 PM
Session Chair: Thomas Conley, Department of Computer Science, University of Colorado, Colorado Springs, CO

Brendan Bena, Drury University, Springfield, MO: Emotion Eliciting Poetry Generation

Kaden Griffith, University of Colorado, Colorado Springs, CO: Using Unambiguous Intermediate Representation to Solve Arithmetic Word Problems

Andrew Conley, Rensselaer Polytechnic Institute, Troy, NY: Using Graph Embedding for Natural Language Inference

Tiffany Chien, University of California, Berkeley, CA: Adversarial Analysis of Natural Language Understanding Systems

2:55-3:40 PM
Session Chair: Dr. Yanyan Zhuang, Department of Computer Science, University of Colorado, Colorado Springs, CO

Sven Marnauzs, Boise State University, Boise, ID: A Domain Independent Social Media Depression Detection Model

Justin Leo, University of Colorado, Colorado Springs, CO: Moving Towards an Incremental Learning Model

Joshua Frederick, California Polytechnic State University, San Luis Obispo, CA: PixelMRF: A Deep Markov Random Field for Image Generation

3:55-4:40 PM
Session Chair: Joseph Worsham, University of Colorado, Colorado Springs, CO

Jonathan Schwan, University of Colorado, Colorado Springs, CO: Estimating Motion Parameters from RGB-D Video

Sonia Rao, University of Georgia, Athens, GA: Self-supervised Deep Learning for Fluorescence Microscopy Denoising

Clare Minnerath, Providence College, Providence, RI: Self-supervised Learning for Single-molecule Localization Microscopy Denoising

Our Session Chairs

Thomas Conley has been a computer programmer for more than 30 years and has worked in many domains including computational linguistics and bioinformatics. He is currently the Information Security Officer at UCCS. He has been an instructor at UCCS and has recently entered the PhD program in Computer Science where he will concentrate on computational linguistics. Tom co-authored a paper titled "Improving Computer Generated Dialog with Auxiliary Loss Functions and Custom Evaluation Metrics" with an REU student at the International Conference on Machine Learning in 2018.

Dr. Yanyan Zhuang is an Assistant Profesor in the Department of Computer Science at the University of Colorado, Colorado Springs. Her areas of research are cybersecurity, privacy, computer networking and software engineering. Yanyan's work on software code comprehension received an ACM SIGSOFT Distinguished Paper Award in 2018.

Joseph Worsham received his BS and MS in Computer Science from the University of Colorado, Colorado Spring, and currently is a PhD student in Computer Science. He is a full-time employee at Lockheed Martin. Joe presented a paper titled "Genre Identification and the Compositional Effect of Genre in Literature" at the prestigious COLING conference held in Santa Fe, New Mexico, in August 2018.

Moving Towards Open Set Incremental Learning: Readily Discovering New Authors

Justin Leo
Department of Computer Science
University of Colorado
jleo@uccs.edu

Jugal Kalita
Department of Computer Science
University of Colorado
jkalita@uccs.edu

Abstract

The classification of textual data often yields important information. Most classifiers work in a closed world setting where the classifier is trained on a known corpus, and then it is tested on unseen examples that belong to one of the classes seen during training. Despite the usefulness of this design, often there is a need to classify unseen examples that *do not belong* to any of the classes on which the classifier was trained. This paper describes the open set scenario where unseen examples from previously unseen classes are handled while testing. We examine a process of enhanced open set classification with a deep neural network that discovers new classes by clustering the examples identified as belonging to unknown classes, followed by a process of retraining the classifier with newly recognized classes Through this process we move to an incremental learning model where we continuously find and learn from novel classes of data that have been identified automatically. We also develop a new metric that measures multiple attributes of clustering open set data. Multiple experiments across two author attribution data sets show we are able to create an incremental model that produces excellent results.

Introduction

Formal as well as informal textual data are over-abundant in this Internet-connected era of democratized publishing and writing. These textual information sources are in multiple forms such as news articles, electronic books and social media posts. The use of text classification allows us to determine important information about the texts that can often be used to connect to the respective authors, naturally leading to the concept of Authorship Attribution. Authorship Attribution is seen as the process of accurately finding the author of a piece of text based on its stylistic characteristics (Rocha et al. 2016). Authorship Attribution is useful in scenarios such as identification of the author of malicious texts or the analysis of historical works with unknown authors.

Typically, text classification has a few well-established stages. The words in the text corpus are transformed using an embedding algorithm, and a classifier is trained with documents labeled with associated classes. In Authorship Attribution, the text samples tend to be books such as novels, transcribed speeches, or Internet-mediated social media posts, where each sample is labeled with the corresponding author. The trained text classifier is given testing data that is usually unseen text samples from the same set of trained authors. This process describes a closed set approach because the tested samples are associated with the same trained classes. A problem with this process of classification arises if the testing data includes samples from unfamiliar authors. In these cases, the classifier typically and erroneously associates the piece of text with a wrong author—an author on which it was trained. To remedy this problem, a new approach called open set classification has been proposed. Open set classification enables the classifier to discriminate among the known classes, but additionally and importantly, to identify if some test example is not associated with any of the classes on which it was trained (Scheirer et al. 2012).

There has been some recent work on open set classification using convolution neural networks (CNN) and recurrent neural networks (RNN). Prior work on open set classification has often been in areas such as computer vision (Bendale and Boult 2015), speech processing (Dahl et al. 2011), and natural language processing (Higashinaka et al. 2014). In this paper, we utilize open set recognition to identify the presence of test examples from novel classes, and incorporate these new classes to those already known to create an incremental class-learning model.

The rest of the paper is organized as follows. After describing related work in the the next section, we present our approach to identifying new classes and instantiating them. Then, we discuss our evaluation metrics for assessing incremental learning, followed by experimental results using authorship attribution datasets and analysis. We conclude by reiterating our accomplishments and thoughts on future work.

Related Work

We discuss related work in terms of four topics: deep networks for open set classification, metrics for open set classification, open set text classification, and recent proposals to use loss functions for open set classification in the context of computer vision.

Open Set Deep Networks

Using deep neural networks for open set classification often requires a change in the network model. Modern neural net-

works have multiple layers connected in various ways, depending on the classifier architecture being used. Most models eventually include a softmax layer that classifies the data to the known classes, with an associated confidence level or probability for each class. A test example is considered to belong to the class which has the highest probability among all the classes. To adapt this model to the open set scenario, the softmax layer was replaced by a unique layer named the OpenMax layer (Bendale and Boult 2016). This layer estimates the probability of an input being from one of the known classes as well as an "unknown" class, which lumps together all classes unseen during training. Thus, the network is able to recognize examples belonging to unknown classes, enhancing the ability of the closed set classifier it starts with.

Metric for Evaluating Open Set Classification

The process of open set class recognition leads to new challenges during the evaluation process. There are multiple sources of error that could be present including: misclassification of known or unknown classes and determination of novel classes. Bendale and Boult (2015) proposed a metric to evaluate how individual examples are classified. Although they proposed it for use in computer vision, we think it is applicable in author attribution as well.

Deep Open Set Text Classification

Prakhya, Venkataram, and Kalita (2017) modify the single OpenMax layer proposed by (Bendale and Boult 2016) to replace the softmax layer in a multi-layer convolution neural networks with an ensemble of several outlier detectors to obtain high accuracy scores for open set textual classification. The ensemble of classifiers uses a voting model between three different approaches: Mahalanobis Weibull, Local Outlier Factor (Kriegel et al. 2009), and Isolation Forest (Liu, Ting, and Zhou 2008). The average voting method produced results that are more accurate in detecting outliers, making detection of unknown classes better.

Loss Functions for Open Set Classification

A problem that often occurs in open set classification is the classifier labeling known class data as unknown. This problem typically occurs if there are some similar features in the examples of the pre-trained classes and unknown classes encountered during testing. In the context of computer vision, Dhamija, Günther, and Boult (2018) introduce what is called the Entropic Open-Set loss function that increases the entropy of the softmax scores for background training samples and improves the handling of background and unknown inputs. They introduce another loss function called the Objectosphere loss, which further increases softmax entropy and performance by reducing the vector magnitudes of examples of unknown classes in comparison with those from the known classes, lowering the erroneous classification of known class data as unknown. Since this approach squishes the magnitudes of all examples that belong to all unknown classes, it makes later separation of individual unknown classes difficult.

Approach

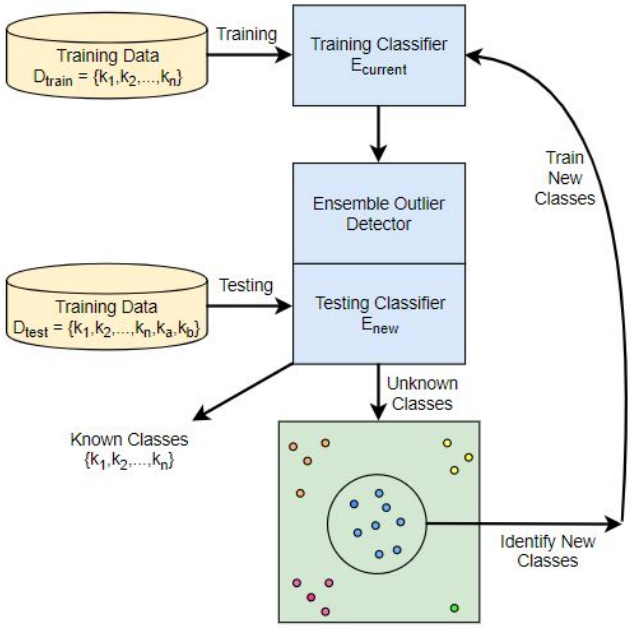


Figure 1: Protocol for Open Set Classification and Incremental Class Learning

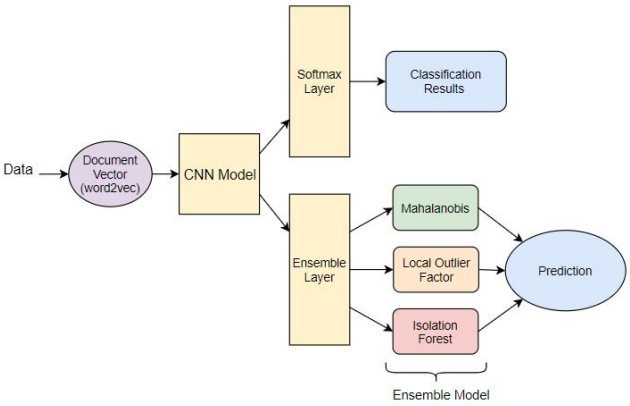


Figure 2: Ensemble Model and Testing Classifier Diagram. This diagram more clearly describes the 'Ensemble Outlier Detector' component from Figure 1.

This paper explores open set classification and the process of moving towards incremental learning of new classes. The objective is to create a classifier framework that can incrementally learn and expand its knowledge base as additional data is presented as shown in Figure 1. The approach is also outlined in Algorithm 1.

In prior work on open set classification, authors have focused on recognizing test samples as belonging to classes unknown during training. To the best of our knowledge, we

are the first ones to instantiate new classes iteratively, extending prior work to real incremental class learning. We first summarize our approach to provide a easily comprehensible sketch, before moving on to details. We seed our classifier framework by training it with examples from a small number of selected classes. We then expose the trained classifier to a mix of examples from the already-known classes as well unknown classes, during testing. At a certain point, we stop the testing of the current-classifier and cluster all examples recognized as belonging to unknown classes. Clustering allows for the grouping of similar data and visually represents the differences between unique clusters. Our hypothesis is that, if the clustering is good, one or more of the clusters of unknown examples can be thought of as new classes the current-classifier has not seen and these clusters are instantiated as new classes, by making up new unique labels for them. At this point, the current-classifier is updated by retraining it with all examples of the old known classes as well the newly instantiated classes. This process—of training, accumulating of outliers, clustering, and instantiating selected new classes out of the clusters—is repeated repeated a number of times, as long as the error of the entire learning process remains acceptable.

In particular, the classifier is a multi-layer CNN structure for training purposes. During testing, the softmax layer at the very end replaced by an outlier ensemble, following the work of (Prakhya, Venkataram, and Kalita 2017). The outlier detector ensemble consists of a Mahalanobis model, Local Outlier Factor model, and an Isolation Forest model, like (Prakhya, Venkataram, and Kalita 2017). The classifier model, as used in training is shown in Figure 2. Initially the model is created by training a classifier $E_{current}$ with a given k_{seed} number of classes found in the entire training data set $\mathbf{D}$. Then we create a derived dataset $D^{test}_{current}$ for testing the model by mixing examples of $k_{unknown}$ unknown classes with the previously trained k_{seed} classes. We always add k_{new} classes to the number of known classes. Thus, at the end of the ith iteration of class-learning, the classifier knows $k_{seed} + (i-1)k_{new}$ classes. We instantiate "new" classes by choosing dominant clusters, and then retrain the model with these new classes. The classes are then removed from the set of all classes and new ones are selected for the incremental addition.

We experiment with multiple clustering techniques including K-Means (Hartigan and Wong 1979), Birch (Zhang, Ramakrishnan, and Livny 1996), DBScan (Ester et al. 1996), and Spectral (Stella and Shi 2003), to determine the most suitable one for author attribution. We also experiment with various values of the parameters: k_{seed}, $k_{unknown}$ and δ.

Evaluation Methods

Since we use clustering as well as classification in our protocol for incremental classification, we need to evaluate both. Below, we first outline how clusters obtained from examples classified as unknown are evaluated, and then we describe how the incremental classifier is evaluated.

Evaluation of Clustering

There are a variety of clustering algorithms, and we need to choose one that works well in the domain of author attribution. The test samples that are deemed to be outliers are clustered, with the hypothesis that some of these clusters correspond to actual classes in the original dataset. We use the Davies-Bouldin Index as shown in Equation (1) to evaluate clustering (Davies and Bouldin 1979).

$$DB = \frac{1}{n}\sum_{i=1}^{n} max_{j\neq i}\left(\frac{\sigma_{\mathrm{i}} + \sigma_{\mathrm{j}}}{d(c_{\mathrm{i}}, c_{\mathrm{j}})}\right) \tag{1}$$

In this formula, n is the number of clusters produced, σ_i is the average distance between the points in cluster i and its centroid, $d(c_i, c_j)$ is the Euclidean distance between the centroids of clusters indexed i and j. Typically lower Davies-Bouldin Index scores indicate better clustering. Another clustering evaluation metric that we use is the V-Measure as shown in Equation (2), which has been widely used in clustering in natural language processing tasks when ground truth is known, i.e., we know samples and the classes they belong to. This metric computes the harmonic mean between homogeneity and completeness (Rosenberg and Hirschberg 2007). Homogeneity measures how close the clustering is such that each cluster contains samples from one class only. Completeness measures how close the clustering is such that samples of a given class are assigned to the same cluster. Typically scores close to 1 indicate better clustering. Here β is a parameter used to weigh between the two components—a higher value of β weighs completeness more heavily over homogeneity, and vice versa.

$$V = \frac{(1+\beta) * homogeneity * completeness}{\beta * homogeneity + completeness} \tag{2}$$

Evaluation of Open Set Misclassification Error

Assuming there are n known classes, multi-class classification using a classifier $E_n()$, trained on n classes, can be evaluated using the misclassification error:

$$\epsilon_n = \frac{1}{N}\sum_{i=1}^{N}\left[E_n(\mathbf{x}^{(i)}) \neq y^{(i)}\right] \tag{3}$$

where N is the total number of samples in the dataset. When we test the same classifier $E_n()$ in the context of open set classification, we need to keep track of errors due that occur between known and unknown classes. When we test this classifier on N samples from n known classes and N' samples from u unknown classes, we test a total of $N + N'$ samples over $n + u$ classes. The open set classification error ϵ_{OS} for classifier E_n is given as (Bendale and Boult 2015):

$$\epsilon_{OS} = \epsilon_n + \frac{1}{N'}\sum_{j=N+1}^{N'}\left[E_n(\mathbf{x}^{(i)}) \neq unknown\right] \tag{4}$$

Evaluation of Incremental Class Learning Accuracy

For our research we are using clustering in order to obtain new classes after we perform open set recognition. This way

Input: Training Set $\mathbf{D} = \left\langle \mathbf{x}^{(i)}, y^{(i)} \right\rangle, i = 1 \cdots N$, samples from all known classes
Output: An incrementally trained classifier E on examples from a number of classes in $\mathbf{D}$

$\mathcal{C}_{all} \leftarrow \left\{C_1, \cdots C_n\right\}$, set of all known classes
$\mathcal{C}^{train}_{current} \leftarrow$ (randomly) pick k_{seed} classes from $\mathcal{C}_{all}$
$\mathbf{D}^{train}_{current} \leftarrow \left\{\left\langle \mathbf{x}^{(i)}, y^{(i)} \right\rangle \mid y(i) \in \mathcal{C}^{train}_{current}\right\}$, samples from classes in $\mathcal{C}^{train}_{current}$
repeat
  $\mathcal{C}^{unknown}_{current} \leftarrow$ (randomly) pick $k_{unknown}$ classes from $\mathcal{C}_{all} - \mathcal{C}^{train}_{current}$
  $\mathbf{D}^{test}_{current} \rightarrow \mathbf{D}^{train}_{current} \bigcup \left\{\left\langle \mathbf{x}^{(i)}, y^{(i)} \right\rangle \mid y^{(i)} \in \mathcal{C}^{unknown}_{current}\right\}$
  $E_{current} \leftarrow$ (CNN) classifier trained on $\mathbf{D}^{train}_{currrent}$
  $\mathbf{O} \leftarrow$ outlier samples detected by ensemble outlier detector when tested on $\mathbf{D}^{test}_{current}$
  $\mathcal{L} \leftarrow$ set of clusters produced from $\mathbf{O}$ using a selected clustering algorithm
  $\mathcal{L}_{dominant} \rightarrow$ pick k_{new} dominant clusters from $\mathcal{L}$, call these clusters new classes by making up new labels for them
  $\mathcal{C}^{train}_{current} \leftarrow \mathcal{C}^{train}_{current} \bigcup \mathcal{L}_{dominant}$, increase the number of "known" classes
  $\mathbf{D}^{train}_{current} \leftarrow \mathbf{D}^{train}_{current} \bigcup \left\{\left\langle x, y \right\rangle \in L_j \mid L_j \in \mathcal{L}_{dominant}\right\}$
until *too low accuracy* **or** n *times*;
$E \rightarrow E_{current}$
return E

Algorithm 1: Algorithm for Incremental Class-Learning

the new data identified for the novel classes can be used to incrementally train the model. For the evaluation of these clusters we present a new metric *ICA* (Incremental Class Accuracy) which takes into account the specific data from an identified cluster and averages calculations of homogeneity, completeness, and unknown identification accuracy of the cluster. We define homogeneity as the ratio of the number of data samples of the predominant class *c* in the cluster *k* ($n_{c|k}$) and the total number of values in the cluster (N_k). We define completeness as the ratio of the number of data samples of the predominant class *c* in the cluster *k* ($n_{c|k}$) and the total number of tested samples of the same class N_c. We define unknown identification accuracy as ratio of the number of unknown *u* data samples in the cluster *k* ($n_{u|k}$) and the total number on values in the cluster N_k. The equation used for ICA assumes only one cluster is being evaluated, but the equation can be adapted for multiple clustering by finding multiple ICA scores for each cluster and averaging.

$$Homogeneity = \frac{max(n_{\mathrm{c|k}})}{N_{\mathrm{k}}} \tag{5}$$

$$Completeness = \frac{max(n_{\mathrm{c|k}})}{N_{\mathrm{c}}} \tag{6}$$

$$\textit{Unknown Identification Accuracy} = \frac{(n_{\mathrm{u|k}})}{N_{\mathrm{k}}} \tag{7}$$

$$ICA = \left(\frac{max(n_{\mathrm{c|k}})}{N_{\mathrm{k}}} + \frac{max(n_{\mathrm{c|k}})}{N_{\mathrm{c}}} + \frac{(n_{\mathrm{u|k}})}{N_{\mathrm{k}}}\right) * \frac{1}{3} \tag{8}$$

Other metrics that will be used to determine the performance of the model will be accuracy and F-score, these figures inherently show the accuracy of the classifier as well as novel data detection.

Experiments and Results

In this section we discuss the data sets used, the experiments performed, and the results with analysis.

Datasets

Since our objective is on open set author attribution, we use two datasets each of which contains 50 authors.

- **Victorian Era Literature Data Set** (Gungor 2018): This dataset is a collection of writing excerpts from 50 Victorian authors chosen from the GDELT database. The text has been pre-processed to remove specific words that identify the individual piece of text or author (names, author made words, etc.). Each author has hundreds of unique text pieces with 1000 words each.
- **CCAT-50** (Houvardas and Stamatatos 2006): This data set is a collection of 50 authors each with 50 unique text pieces divided for both training and testing. These texts are collections of corporate and industrial company news stories. This data is a subset of Reuters Corpus Volume 1.

Preliminary Clustering Results

After experimental comparison of the different clustering techniques, we decided to use Spectral Clustering (Stella and Shi 2003) as this typically produces the highest accuracy results as seen in Figure 3 and Figure 4, the clustering evaluation scores are also used for comparison. We use the pre-trained model *word2vec* (Mikolov et al. 2013) to obtain the word embeddings to pass into the multi-layer CNN structure.

Incremental Classification Results

For the first experiment our objective was to see if we could use our method to improve our classification accuracy and to also decide which clustering algorithm would work best. We run both data sets individually with five known training classes and then with ten known training classes, then we introduce three unknown classes during the testing phase for each of the tests. Our results include the comparison with accuracy and F1-Score as found on Table 1; a significant

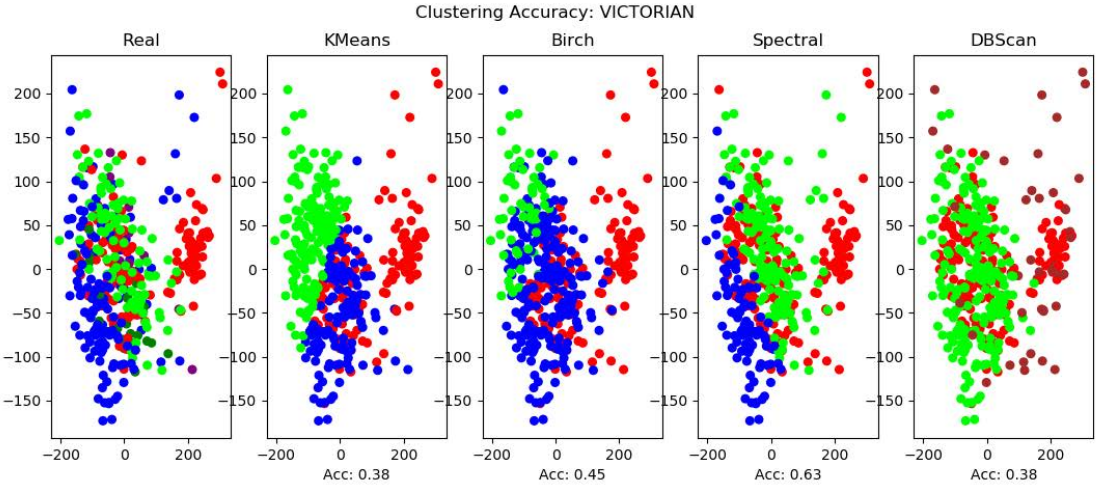


Figure 3: Clustering Plots for Victorian Literature Data with Accuracy Score, 5 Trained Classes and 8 Tested Classes

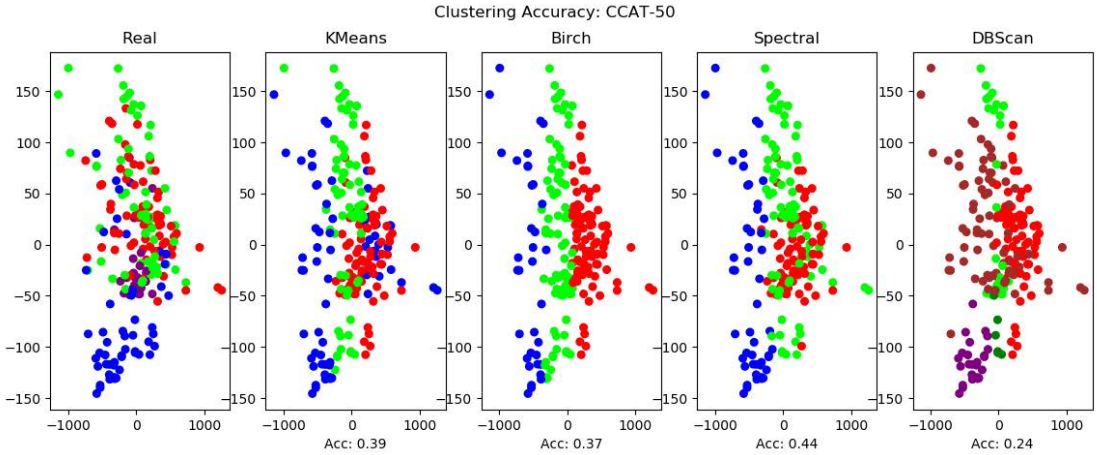


Figure 4: Clustering Plots for CCAT-50 Data with Accuracy Score, 5 Trained Classes and 8 Tested Classes

increase of these values is observed after the classifier is re-trained with the identified novel classes. Our clustering evaluation metrics are found on Table 2. V-Measure scores prove to be more useful because the Davies-Bouldin scores do not always indicate the highest accuracy of clustering, this is because the best formed clusters does not necessarily mean higher accuracy. Even though our chosen data sets have not been used for open set classification in prior research we can compare our open set classification scores with the state of the art closed set classification scores. As far as we know, the best classification F1-Score from prior work for the Victorian Literature data set using only few classes is 0.808 (Gungor 2018) and our model has a similar score. Also as far as we know, the best classification accuracy score for the CCAT-50 data set for using only few classes is 86.5% and we are obtaining similar results. Our clustering models seem to have the most error for both data sets (especially the CCAT-50 data), thus presumably better clustering models or would produce greater results.

Dataset	Pre-Trained		Post-OpenSet	
	Acc	F1	Acc	F1
Victorian 5class	56.29%	0.592	85.43%	0.855
CCAT-50 5class	54.75%	0.565	83.00%	0.825
Victorian 10class	61.29%	0.644	71.38%	0.706
CCAT-50 10class	62.50%	0.727	86.77%	0.866

Table 1: Pre-Trained Class Scores and Post-Open Set Classification Scores, Either 5 or 10 initial trained classes and 3 unknown added during testing

For the second experiment we initially train with a fixed amount of classes k_{seed} and then incrementally add a $k_{unknown}$ amount of classes for testing. We repeat this process to demonstrate the model incrementally learns as the learning and open set classification cycle is repeated. We run this test by adding classes for multiple iterations and record the change in the F1-Score for the overall classification and generation of new classes; we attempt to run each test until

	Davies-Bouldin				V-Measure			
Data Set	Vic-5	CCAT-5	Vic-10	CCAT-10	Vic-5	CCAT-5	Vic-10	CCAT-10
K-Means	2.739	2.045	1.989	0.876	0.078	0.039	0.147	0.082
Birch	2.670	2.237	4.193	3.654	0.165	0.075	0.147	0.083
Spectral	4.457	2.550	4.841	0.807	0.319	0.242	0.328	0.258
DBScan	4.031	2.432	4.783	4.349	0.065	0.101	0.158	0.149

Table 2: Davies Bouldin Index and V-Measure Score for Clustering methods evaluated, Either 5 or 10 trained classes and 3 unknown added during testing.

the results drop significantly or until we have reached a max value of classes. From Figure 5 we notice the results of the incremental cycle and we notice that we achieve better results when fewer classes are added at a time. We run tests for adding 1, 2, and 3 classes at a time. We also keep track of the open set error shown in Equation 4; this metric shows error of unknown data identification but not novel class generation. The problem we notice with the experiment is that error will propagate through the process so as error accumulates the results deter. We also notice based on the results from both data sets, adding one class incrementally each iteration has better results because this limits the clustering error. We also notice that the Victorian Literature does worse than the CCAT-50 data and we think this is because of the text samples; the Victorian text includes words with slurs and accent mark symbols and *word2vec* is not pre-trained with these new features. The CCAT-50 data tends to have very distinct authors and the pieces of text tend to also tend to be more unique. Overall based on the results, we notice that most of this error can be attributed to the clustering process.

Form the previous experiments we realized the clustering process tends to have the most variance, this is evident from the low clustering accuracy due to the lack of fully distinct clusters. Thus, there needs to be a way to evaluate the clustering. Using our Incremental Class Accuracy (ICA) metric shown from Equation 5 we will be able to evaluate the clustering in regards to homogeneity, completeness, and unknown identification accuracy. From the previous experiment we also notice that adding one class at a time incrementally tends to produce the best results, so we calculate the ICA score when one class is added and instantiated. The results for both data sets is shown in Table 3. From these results we notice having a fewer amount of initial trained k_{seed} classes produces better results and this is expected as the $k_{unknown}$ classes are more easily identified.

Initial Training	Victorian	CCAT-50
5 Classes	0.687	0.875
10 Classes	0.593	0.754
15 Classes	0.529	0.764
20 Classes	0.387	0.681

Table 3: ICA Scores for 1 added class/cluster evaluation. Scores based on Equation 5.

Conclusion

This research works with open set classification regarding NLP text analysis in the area of Authorship Attribution. The model created will be to determine the originating author for a piece of text based on textual characteristics. We also move towards a novel incremental learning approach where unknown authors are identified and then the data is labeled so the classifier expands on its knowledge. Through this process we expand upon the state of the art implementation by creating a full cycle model by training on given data and then expanding the trained knowledge based on new data found for future testing.

Text based Authorship Attribution can be applied to research involving security and linguistic analysis. Some similar developing work using similar research methods involving image recognition (Rebuffi et al. 2017), this can be applied to facial recognition tasks and video surveillance applications. This model can also be further improved by developing a more precise way of distinguishing different pieces of text. Another method for future research is using backpropagation. Once novel classes are identified, the model should be then able to modify the already trained classifier with the $D^{train}_{current}$ data. Then the model can be tested with the $D^{test}_{current}$ to determine if the model can recognize previously unknown classes. Backpropagation of a neural network requires a fully inter connected set of layers that allow the processing of data through either side of the model (Hecht-Nielsen 1992). This process would save the step of fully retraining the classifier model. A similar approach to this can also be to add new "neurons" to a deep neural network to allow for an extension of a trained model (Draelos et al. 2017). With these new future improvements our model can be further improved and potentially obtain better results.

Acknowledgement

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

References

Bendale, A., and Boult, T. 2015. Towards open world recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1893–1902.

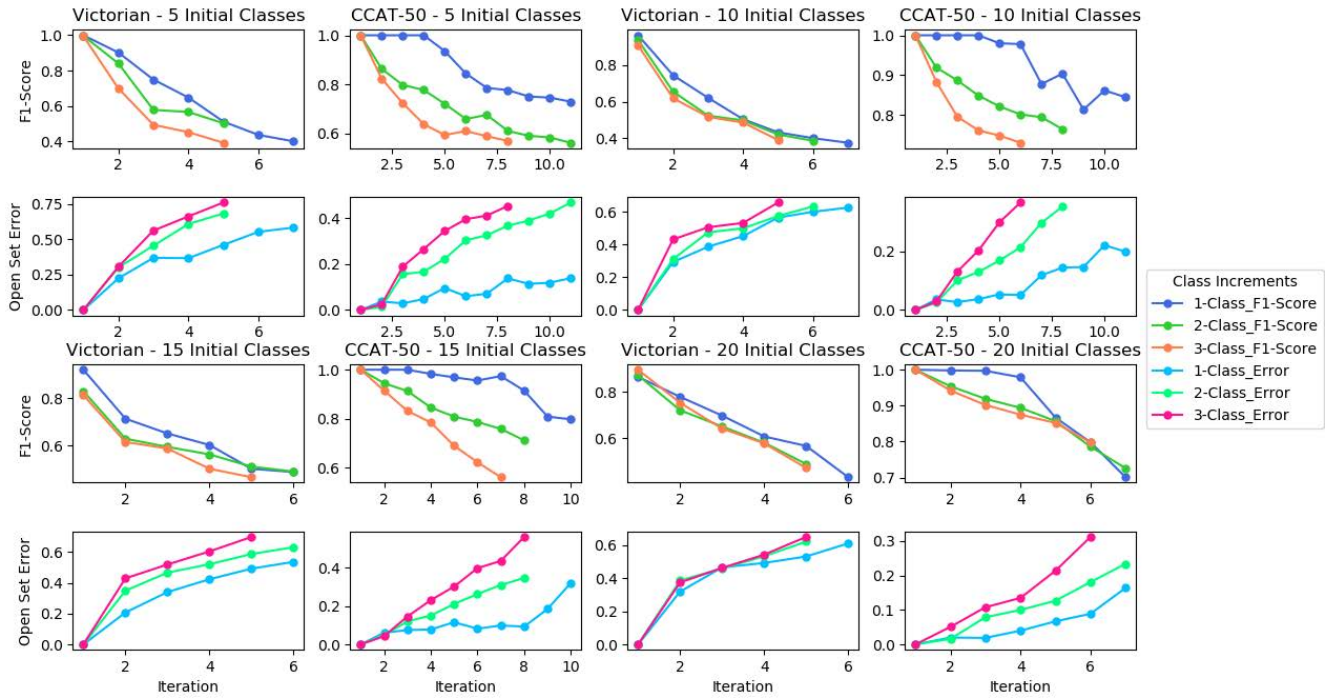


Figure 5: Incremental Learning Plots. Initially trained with 5, 10, 15, and 20 initial classes then tested by incrementally adding 1, 2, and 3 Classes. These plots show the final F1-Scores and Open Set Error from Equation 4.

Bendale, A., and Boult, T. E. 2016. Towards open set deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1563–1572.

Dahl, G. E.; Yu, D.; Deng, L.; and Acero, A. 2011. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on audio, speech, and language processing* 20(1):30–42.

Davies, D. L., and Bouldin, D. W. 1979. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence* (2):224–227.

Dhamija, A. R.; Günther, M.; and Boult, T. 2018. Reducing network agnostophobia. In *Advances in Neural Information Processing Systems*, 9157–9168.

Draelos, T. J.; Miner, N. E.; Lamb, C. C.; Cox, J. A.; Vineyard, C. M.; Carlson, K. D.; Severa, W. M.; James, C. D.; and Aimone, J. B. 2017. Neurogenesis deep learning: Extending deep networks to accommodate new classes. In *2017 International Joint Conference on Neural Networks (IJCNN)*, 526–533. IEEE.

Ester, M.; Kriegel, H.-P.; Sander, J.; Xu, X.; et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, 226–231.

Gungor, A. 2018. *Benchmarking authorship attribution techniques using over a thousand books by fifty Victorian era novelists*. Ph.D. Dissertation.

Hartigan, J. A., and Wong, M. A. 1979. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28(1):100–108.

Hecht-Nielsen, R. 1992. Theory of the backpropagation neural network. In *Neural networks for perception*. Elsevier. 65–93.

Higashinaka, R.; Imamura, K.; Meguro, T.; Miyazaki, C.; Kobayashi, N.; Sugiyama, H.; Hirano, T.; Makino, T.; and Matsuo, Y. 2014. Towards an open-domain conversational system fully based on natural language processing. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, 928–939.

Houvardas, J., and Stamatatos, E. 2006. N-gram feature selection for authorship identification. In *International conference on artificial intelligence: Methodology, systems, and applications*, 77–86. Springer.

Kriegel, H.-P.; Kröger, P.; Schubert, E.; and Zimek, A. 2009. Loop: local outlier probabilities. In *Proceedings of the 18th ACM conference on Information and knowledge management*, 1649–1652. ACM.

Liu, F. T.; Ting, K. M.; and Zhou, Z.-H. 2008. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, 413–422. IEEE.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Prakhya, S.; Venkataram, V.; and Kalita, J. 2017. Open set text classification using convolutional neural networks. In *International Conference on Natural Language Processing, 2017*.

Rebuffi, S.-A.; Kolesnikov, A.; Sperl, G.; and Lampert, C. H. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2001–2010.

Rocha, A.; Scheirer, W. J.; Forstall, C. W.; Cavalcante, T.; Theophilo, A.; Shen, B.; Carvalho, A. R.; and Stamatatos, E. 2016. Authorship attribution for social media forensics. *IEEE Transactions on Information Forensics and Security* 12(1):5–33.

Rosenberg, A., and Hirschberg, J. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, 410–420.

Scheirer, W. J.; de Rezende Rocha, A.; Sapkota, A.; and Boult, T. E. 2012. Toward open set recognition. *IEEE transactions on pattern analysis and machine intelligence* 35(7):1757–1772.

Stella, X. Y., and Shi, J. 2003. Multiclass spectral clustering. In *null*, 313. IEEE.

Zhang, T.; Ramakrishnan, R.; and Livny, M. 1996. Birch: an efficient data clustering method for very large databases. In *ACM Sigmod Record*, volume 25, 103–114. ACM.

Injection of Creativity and Emotion-Elicitation in Poetry Generation

Brendan Bena
Drury University
900 N. Benton Ave.
Springfield, Missouri 65109

Jugal Kalita
UC-Colorado Springs
1420 Austin Bluffs Pkwy.
Colorado Springs, Colorado 80918

Abstract

Poetry Generation, in the context of Natural Language Generation (NLG), involves teaching systems to automatically generate text that resembles poetic work. A system learns to recreate poetry through training on a corpus of poems and modeling the particular style of language. In this paper, we propose taking an approach of fine-tuning GPT-2, a pre-trained language model, to our downstream task of poetry generation. Specifically, we attempt to create emotion-eliciting poetry and dream poetry. Our first goal is to elicit emotions within the reader through the automatically generated text, so we believe a crowdsourced human-evaluation is the proper form of metric. Our model for the emotions of *sadness* and *joy* produced poems that correctly elicited emotions 87.5 and 85 percent of the time, respectively. Our second goal is to apply transfer learning to inject creativity and produce dreamlike poetry. Poems from this model are shown to capture elements of dream poetry with scores of no less than 3.2 on the Likert scale. For further quantitative evaluation, we make use of the Coh-Metrix tool, outlining certain metrics we use to gauge the quality of text generated.

Introduction

Many natural language processing tasks require the generation of human-like language. Some tasks, such as image and video captioning and automatic weather and sports reporting, convert non-textual data to text. Some others, such as summarization and machine translation, convert one text to another. There are additional tasks that aim to produce text, given a topic or a few keywords. These tasks include story generation, joke generation, and poetry generation, among others.

Poetry generation produces creative content, and delivers the content in an aesthetically pleasing manner, usually following a specific structure. Thus, in addition to generating text as if in a story, the lines produced usually have a certain length, quite frequently there is a rhyming scheme as well as rhythm, and organization into structures such as couplets, quatrains, quintets, and stanzas. Among other things, creativity comes from unusual usage of words through effects such as alliteration, assonance, and elision; use of metaphors, symbolism, and other linguistic devices; licensing of underlying imagery with expressed feelings, sentiments and emotions.

Work in natural language generation can be traced to pioneering rule-based simulations of chatbots such as the "psychotherapist" Eliza (Weizenbaum and others 1966) and paranoid schizophrenia-suffering PARRY (Colby 1981). Surveys such as (Hovy 1990; Reiter and Dale 2000; Gatt and Krahmer 2018; Santhanam and Shaikh 2019) have described the progress in natural language generation over 50 years. Of late, the use of deep learning has produced enviable progress in natural language generation, especially in topics such as machine translation (Bahdanau, Cho, and Bengio 2014; Wu et al. 2016), image captioning (Mao et al. 2014) and dialogue generation (Li et al. 2016).

This paper discusses an attempt to generate natural-sounding poems that are creative and can potentially evoke a response from the readers or hearers in terms of emotions and feelings they generate. We choose dreams as our form of creative expression due to its long standing history in poetry. Dream poetry is dated back to medieval times where famous 14th century authors, like Chaucer, experiment using dreams as the structure for an image or picture they wish to paint with a poem (Spearing 1976). A dream poem is said to be characterized by the 'I' of the poem and its substances of a dream or a vision included (Lynch 1998). To the best of our knowledge, prior work on poetry generation, whether using deep learning or not, has not explored the incorporation of emotion-eliciting phraseology or elements of creativity like dream poetry as we do in this paper.

Our research provides the following contributions:

- The use of GPT-2 for poetry generation
- Leveraging a word-level emotion lexicon to categorize emotion-based text
- Exploration of injecting creativity in poetry through the use of dream data

This paper is organized in the following way. Section 2 presents related work. Next, section 3 discusses our approach to creative text generation including pre-processing steps and architecture used. Section 4 talks about our experiments and results. Finally, section 5 gives an evaluation of our research.

Related Work

Early methods for poetry generation made use of template oriented and rule-based techniques. These approaches often

required a large amount of feature picking and knowledge of syntactic and semantic rules in a language (Oliveira 2009; Oliveira 2012). Other methods treated poetry generation as special cases of machine translation or summarization tasks (Yan et al. 2013; He, Zhou, and Jiang 2012). Such methods did not have the ability to learn any aspects of the language in which the poems were written, and thus we feel that they were incapable of any injection of creativity in the generation process. Forcing a model to adhere to specific rules or templates, or summarizing or translating a given text to generate new poetry was unlikely to lead to artistically expressive quality we seek to create.

More recently, deep learning methods have become prevalent in natural language generation, including poetry generation. Zhang and Lapata (2014) used Convolutional (CNN) and Recurrent Neural Networks (RNN) to generate Chinese Poetry. RNNs allow for short-term memory of the language to be maintained by inputting the generated output of a network cell back into itself, essentially building context.

Ghazvininejad et al. (2017) used Long Short-Term Memory (LSTM) units, which are advanced gated versions of RNNs, to the task of poetry generation. Wei, Zhou, and Cai (2018) attempted to address the style issue by training the networks using particular poets and controlling for style in Chinese poetry. They found that with enough training data, adequate results could be achieved. The structure problem was addressed by (Hopkins and Kiela 2017). They generated rhythmic poetry by focusing on training the network on only a single type of poetry to ensure produced poems adhered to a single rhythmic structure. It was found in human evaluations that while the poems produced were rated to be of lower quality than human produced poems, they were indistinguishable from human produced poems. Lau et al. (2018) took the LSTM approach one step further with the *Deepspeare* model by employing an attention mechanism to model interactions among generated words. They also use three neural networks, one for rhythm, one for rhyming and another for word choice in their quest to generate Shakespeare-like sonnets.

Vaswani et al. (2017) developed a deep neural architecture called the Transformer that did away with any sort of need for recurrence. The Transformer also employed an elaborate attention mechanism that has been shown to be useful in natural language tasks. Radford et al. (2019) used this architecture in their Generative Pretrained Transformer 2 (GPT-2) model. GPT-2 is capable of many downstream tasks like text generation, but to our knowledge research has not been published using the GPT-2 model specifically for poetry generation.

On a slightly different but related note, natural language generation influenced by multi-modal input was attempted by (Vechtomova et al. 2018) to generate song lyrics in the style of specific artists by fusing outputs coming from lyrical inputs processed by an RNN and audio clips processed by a CNN. Text generation has also been influenced, in a cross domain manner, through images. The works of (Liu et al. 2018) have shown that coupled visual-poetic embeddings can be used to pick out poetic clues in images, which in turn can be used to inspire the generated text. Though influenced natural language generation in and of itself is not a novel idea, we feel our attempt to style text with the intent of eliciting particular emotions provides a creative way to explore this subtask.

Approach

Our work involves a preliminary step of scoring a corpus of downloaded poems for emotion to produce subsets of poems that express one of eight different identified emotions. This step is followed by the actual generation of poems by finetuning the pre-trained GPT-2 natural language model. Emotion poem generation involves training eight separate models, one on each type of emotion poem, to learn how to model a poetic style of language. These poems are evaluated using automated techniques as well as humans for the emotions they express or elicit in a reader. Finally, we describe how we attempt to introduce aspects of what is deemed as creativity in poetry into poems that are composed automatically. To do so, we gather dream data and apply transfer learning by finetuning on dreams, then again on poetry. A high-level overview of the emotion elicitation portion of our project is shown in Figure 1.

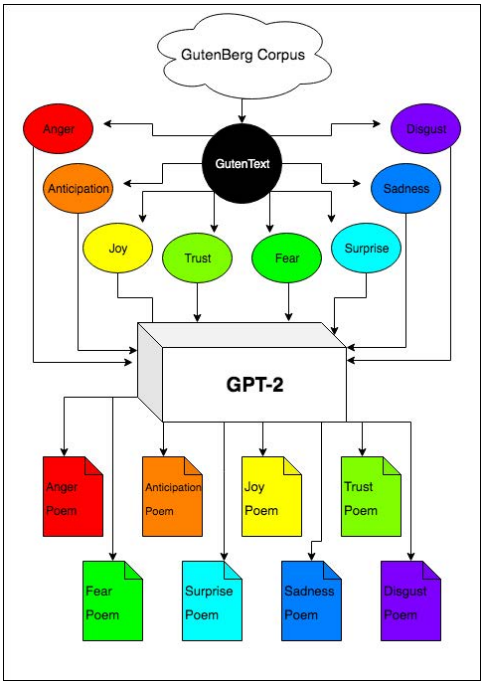


Figure 1: A high-level overview of our project implementation for emotion eliciting poetry

Poem Emotion Scoring

To decipher text depending on the emotions they elicit, we make use of the EmoLex dictionary (Mohammad and Turney 2013). EmoLex is a word-level emotion lexicon that associates English words with the 8 different emotion categories we wish to explore. Each poem (or book of poems) in our dataset is given a score that is the total of the associated

emotion scores in EmoLex for each word. The maximum emotion word score is taken and the poem is labeled under that emotion category. This classification method allows us to train multiple models on our split dataset.

Currently, the emotions of *joy, anticipation, trust, anger*, and *sadness* represent a large portion of our data while the emotions of *surprise, disgust,* and *fear* are severely underrepresented. Table 1 shows key differences in models including the number of tokens in the text and the final average loss during training.

Data	Model Size	# of Tokens	Final Loss
anger	345M	1,292,457	0.27
anticipation	345M	2,314,637	1.30
joy	345M	11,668,792	3.19
sadness	345M	2,090,915	1.03
trust	345M	16,667,178	3.39

Table 1: Comparison of 5 emotion models trained.

GPT Architecture

To create a model for poetic language, we propose finetuning OpenAI's GPT-2 architecture. GPT-2 is a Transformer-based model that was trained simply to predict the next word in a 40GB text corpus (Radford et al. 2019). This 40GB dataset, *WebText*, was scraped from the internet with certain heuristics that aimed to gather only quality text (i.e. only outbound Reddit links from posts with a karma rating of 3 stars or better). By training on such a largely encompassing corpus of text, the architecture has proven to model the English language well and has obtained state-of-the-art results on downstream text-based tasks such as machine translation, question answering, and summarization. We leverage GPT-2's pre-trained knowledge of language for our downstream task of peotry generation.

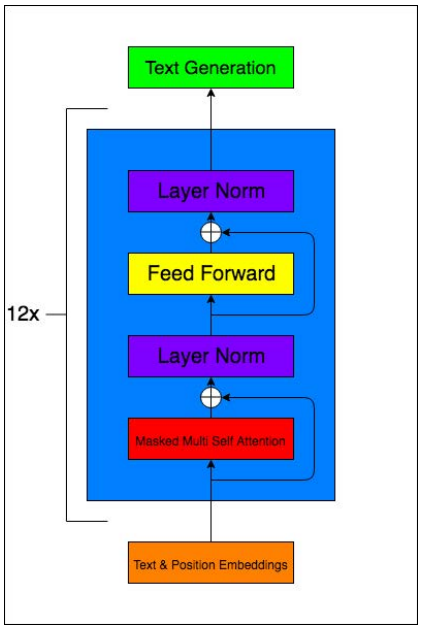


Figure 2: GPT Architecture. Adapted from (Radford et al. 2018; Radford et al. 2019)

GPT-2 (Radford et al. 2019) is the successor of OpenAI's first Transformer-based architecture, GPT (Radford et al. 2018), with a few changes to the structure. The medium version of GPT-2 we use contains 345M parameters and is a 24 layer, decoder-only Transformer architecture. GPT-2 moves layer normalization to the input of each sub-block, adds another layer normalization after the final self-attention block and increases context size from 512 to 1024 tokens. This architecture allows for long term dependencies to be captured better in language modeling. GPT-2's attention mechanism is referred to as a masked multi self-attention head. This technique allows for a relationship to be modeled for all words in an input sequence. Words that have multiple meanings can then be represented based on the context they appear in. Higher attention scores from surrounding words relate to a larger contribution to the representation of a word. GPT-2 makes use of byte-pair encoding (BPE) like its predecessor GPT but on UTF-8 byte sequences (Sennrich, Haddow, and Birch 2015). GPT-2's encoding is somewhere in between character level and word level. The model also prevents different versions of common words from being duplicated (i.e. *fate!*, *fate?*, and *fate* would not be joined). This technique improves the quality of the final byte segmentation. GPT-2's encoding rids the need for pre-processing or tokenization of data and is able to assign a probability to any Unicode string.

The task-agnostic nature of GPT-2 allows us to employ what we claim to be a semi-supervised fine-tuning approach to our downstream task of poetry generation. Though the GPT-2 model learns in an unsupervised manner, our poetry data is split into categories, so that we can train already pre-trained GPT-2 on a sub-corpus of poems that demonstrate a certain emotion or dream-like text without explicitly being told to do so.

Text Generation and Sampling

As stated by Radford (2019), the core approach of GPT-2 is language modeling. A language model can be thought of as a probability distribution over a sequence of words in the form:

$$p(w_1, ..., w_n) \tag{1}$$

Likewise, natural language tends to have a sequential order so it can be modeled as the probability of word given preceding words in the form (Bengio et al. 2003):

$$p(w_n|w_1, ..., w_{n-1}) \tag{2}$$

We make use of the probabilistic style of language modeling by sampling from the distribution in a semi-random fashion. Just as the GPT-2 paper does for its text generation, we make use of Top K sampling, limiting the possible guesses of words to 40. In addition to Top K, we make use of a temperature constant of 0.75 which controls randomness in the distribution. A temperature closer to 0 correlates to less randomness and a temperature of 1 relates to more randomness. Finally, at the end of the generation process, we

employ a simple text cleaning algorithm that allows poems to end more naturally and not trail off as they do sometimes.

Experiments and Results

Datasets and Resources

In order to classify emotion-eliciting poems or books, we use the NRC Word-Emotion Association Lexicon (EmoLex) resource. EmoLex was created by the National Research Council of Canada and includes 14,182 English words that are associated with different emotions and positive or negative sentiment (Mohammad and Turney 2013). Words in EmoLex have been manually annotated via crowd-sourcing and emotions fall into one or more categories of eight basic emotions: *joy, trust, fear, surprise, sadness, anticipation, anger, and disgust* (Plutchik 2014). This resource provides us with a way to fabricate a ground truth in the types of emotion-infused texts we wish to use for training data.

Figure 3: American pyschologist Robert Plutchik's Wheel of Emotions

To handle the training and generation portions of the project, we draw data from the Project Gutenberg website. Project Gutenberg is a massive online database containing over 59,000 eBooks. We limit this corpus to a smaller subcorpus using an adaptation of the GutenTag tool (Brooke, Hammond, and Hirst 2015). This tool allows us to place constraints on the amount of literature we choose to use in our work. Our final dataset includes approximately three million lines of poetic text from the Gutenberg database and is further divided by poem/book into our eight emotion categories.

We attempt to create dream poetry by making use of the *DreamBank* dataset. The *DreamBank* was created by Schneider & Domhoff at UC-Santa Cruz. The dataset contains a collection of over 20,000 dreams from users age 7 to 74. We scraped this dataset from the website assuring that dreams collected were recorded only in English. The *DreamBank* allows us to attempt transfer learning by finetuning on the dream dataset first, then further finetuning on our poetry dataset.

Amidst the chaos throng'd, with angry voices each
His rival's mockery; loud their scorn was fill'd;
So fierce their rage, and in their eager power
Met on the walls of Troy, were fill'd with dismay.

Figure 4: A hand-picked poem from the anger model

Heard I a song of joy,
A song of happy sound,
Fills all the air I breathe,
To him I sing, to him
I sing the happy song.
All night long on the steep green grass
I ride and sing

Figure 5: A hand-picked poem from the joy model

1 2 3 4

Initially, we have retrained 6 GPT-2 based models. Default training parameters were used each of the 5 different emotion datasets and our dream dataset. All were trained for 12,000 steps (other than our dream model that was trained for 12k steps on both dreams then poetry) with a learning rate of 0.0001. When generating text, we do not input context and allow the model to write the poem entirely through the sampling of conditional probability from the language it has modeled.

Figures 4 through 8 give examples of 5 poems that we have hand-picked to illustrate the quality of poems generated. A cursory glance at the poems shows that the quality of the text in terms of lexical choice, grammatically, and semantic cohesion is high. We discuss how we quantitatively assess the poems below.

[1] https://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm
[2] https://www.gutenberg.org/
[3] https://www.dreambank.net/
[4] https://github.com/nshepperd/gpt-2

We have reached the peak of the highest mountain in the world
The mountain of dreams.
This is the view
Across the valley,
One hour's journey back,
We crossed it on the way between
A band of beautiful young women.
There was

Figure 6: A hand-picked poem from the anticipation model

A long trail of falling mist
Had made its way here, and now
Aerily it seemed, as if to drown
The discordant thunder clang.
It seemed to drown the music of the rain;
In this lost place of sorrow
Far off

Figure 7: A hand-picked poem from the sadness model

The other, who with one accord
Wrote my essay, in that he was dear
And good, and knew well, how we ought to treat
A man of such renown, and such love?
He's a good honest man, no doubt

Figure 8: A hand-picked poem from the trust model

A thousand stars at once,
An hundred thousand stars!
The sun was low,
And the stars were bright,
My heart would do the same.
A thousand stars at once,
A hundred thousand stars!
The night had begun,
And the stars were all the same.
When I came back from the dead,
I saw the stars

Figure 9: A hand-picked poem from the dream model

For she was mine.
I was the only one
She had,
And a thousand other friends,
And a hundred more
She held me dear.
Her eyes were clear, her cheeks were bright,
Her heart was like a rose,
Her mouth was full of music,
Her lips were white
As snow,
And the music she sang

Figure 10: A hand-picked poem from the dream model

Emotion	Anger	Antic.	Joy	Sadness	Trust
%	65	40	85	87.5	32.5

Table 2: Average percentage of correctly elicited emotion across four poems in each category

Poem	1	2	3	4
Qual 1	**5**	4.9	4.8	4.5
Qual 2	3.5	**4.1**	3.2	3.3
Qual 3	3.9	**4.2**	3.7	3.7

Table 3: Average Likert score of users for each poem

Evaluation

In the first crowd-sourced analysis of our emotion-eliciting poetry we presented 4 poems, of the five data-represented emotion categories, to ten human reviewers of undergraduate level educational backgrounds. These reviewers were asked to rate each poem based on the emotions elicited within them after reading. Table 2 illustrates the results from our evaluation. When taking the average percentage of correctly emotion-eliciting poems, the models of joy, sadness, and anger produced the most promising results while the trust and anticipation models were less than satisfactory.

To preserve consistency in our experiments, we evaluate our dream model poetry in a similar manner to the emotion poems. 4 poems from the model are presented to the same ten judges and they were asked to assess the poems based on qualities of dream poetry. A dream poem is said to have these qualities:

- The poem is generally a first-person expression
- The poem main substance is dream or vision like
- The poem recounts or foretells an experience or event

Analysis of results show that poems are able to capture the first person perspective well, achieving between 4.5 and 5 average Likert scores. The poems also appear to retell a story or an event often, scoring between 3.7 and 4.2 average Likert scores. The nature of poetry and dream recounts that make up our data is often narrative so this result stands to reason. However, Quality 2 scores of the poem substance containing a dream or vision are questionable. We suspect the Quality 2 score is lower due to the ambiguity in ascertaining dream text from regular text. Table 3 highlights our results for the dream model.

Currently, there exists no widely available standard for evaluating poetry generation. Scores like BLEU, ROUGE, METEOR, etc. are more suited for Machine Translation (MT) tasks (Zhang et al. 2019). For example, they compare how similar sentence P is to translated-sentence $\hat{P}$. Instead, we outline some metrics from the Coh-Metrix web tool that helps us further quantitatively evaluate the quality of text generated. With the goal in mind of eliciting emotions, we claim that subjective analysis of generated poetry will be superior to any available objective metrics.

Model	RDFRE	RDFKGL	WRDIMGc	WRDCNCc	LDTTRa	PCREFp	PCSYNp	PCNARp
anger	93.073	2.011	445.914	407.159	0.527	0.680	80.780	53.190
anticipation	100	0.832	440.931	403.104	0.404	7.780	83.650	81.860
joy	100	0.394	446.231	403.072	0.389	11.900	91.310	78.520
sadness	98.200	1.180	444.963	403.252	0.444	1.880	88.690	72.910
trust	100	0.156	434.664	412.717	0.334	18.140	84.610	91.310
dream	100	0	427.363	377.476	0.238	99.900	65.170	70.880

Table 4: Average Coh-Metrix evaluations across 25 randomly selected poems from each model.

Coh-Metrix

To provide a quantitative calculation of the caliber of text our models produce, we outline relevant metrics from the University of Memphis Coh-Metrix tool (Graesser et al. 2004). Coh-Metrix is a text evaluation software kit and from it, we have chosen 8 forms of assessment. The first two, Flesch-Kincaid Grade Level (RDFKGL) and Flesch Reading Ease (RDFRE), are two standard measures that deal with text readability and ease (Klare 1974). The RDFKGL scores a text from grade level 0 to 18, while the RDFRE score is a 0-100 index with 100 being an easily readable text. We aim to produce text that is readable by all, so a low RDFKGL score and high RDFRE score would be ideal. The next metrics we use evaluate at the word level. The word the imageability (WRDIMGc) and word concreteness (WRDCNCc) scores measure content words on their ability to create an image in the reader's mind and their ability to appeal to a reader's senses, respectively (Coltheart 1981). We aim for our art to create a connection between the reader and poem, so we believe imageability and concreteness of content words are two good measures with this in mind. We also make use of three text easibility principal component scores in narrativity (PCNARp), referential cohesion (PCREFp), and syntactic simplicity (PCSYNp) (Graesser et al. 2004). All text easibility PC scores are percentile scales, and thus we aim for higher numbers for these scores. Finally, we make use of the Lexical Diversity Type:Token Ratio score (LDTTRa) for all words. LDTTRa measures the ratio of *type* (unique) words to all *tokens* in the text. Because our text is relatively short, we aim for a middle ground in the LDTTRa ratio, meaning there is uniqueness in the word choice of the text, but cohesion is still upheld.

Inspection of our Coh-Metrix results show that randomly selected poems from all models fall at or below the 2nd-grade reading level in RDFKGL scores and are greater than 93 on the RDFRE scale. This suggests generated poems are easily readable by the majority of viewers. Looking at the WRDIMGc and WRDCNCc, we see our poems, except for the dream model concreteness, fall in the 400s. Words with higher imageability and concreteness fall around the low 600s while words that are lower fall around the upper 200s on this scale. These scores reveal that our models are creating text that is both concrete in word choice and paint a picture. Our dream model scoring lower in the concreteness is reasonable as the word choice of dreams tends to be more abstract. Lastly, percentile scores of PCSYNp and PCNARp show that the majority of models are producing poems that are both syntactically simplistic and narrative. Most PCREFp scores are on the lower end of the scale, but because the poems are not necessarily related and were all input at once, we suspect that is the reason these scores are lower. Table 4 highlights these scores for each poetry model.

Conclusion & Future Work

In this paper we attempted to influence natural language generation in the form of poetry generation through the use of classified emotion poems and dream text. To do so, we first leveraged a word-level emotion lexicon to construct a meaning for emotion-eliciting text and used that text to train separate language models. Next, we gathered data of dream records and employed transfer learning in attempts to create dream-like poetry. This paper seeks to create art in the form of auto-generated poetry while opening the door to more projects involving emotion-eliciting text-based tasks and influenced creative neural generation.

Future research in this project will involve gathering data for the underrepresented emotion categories, allowing us to have a language model for each emotion. We will also consider external crowd-sourced evaluation methods like Amazon Turk for a more expansive judgment of our results. In addition, we are interested in the exploration of using other word-level or segment-level emotion lexicons to influence our text generation. Finally, we wish to seek out additional forms of replicating creativity that artists incorporate in their work.

Acknowledgement

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

References

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

[Bengio et al. 2003] Bengio, Y.; Ducharme, R.; Vincent, P.; and Jauvin, C. 2003. A neural probabilistic language model. *Journal of machine learning research* 3(Feb):1137–1155.

[Brooke, Hammond, and Hirst 2015] Brooke, J.; Hammond, A.; and Hirst, G. 2015. Gutentag: an nlp-driven tool for digital humanities research in the project gutenberg corpus. In *Proceedings of the Fourth Workshop on Computational Linguistics for Literature*, 42–47.

[Colby 1981] Colby, K. M. 1981. Modeling a paranoid mind. *Behavioral and Brain Sciences* 4(4):515–534.

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

[Gatt and Krahmer 2018] Gatt, A., and Krahmer, E. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research* 61:65–170.

[Ghazvininejad et al. 2017] Ghazvininejad, M.; Shi, X.; Priyadarshi, J.; and Knight, K. 2017. Hafez: an interactive poetry generation system. In *Proceedings of ACL 2017, System Demonstrations*, 43–48. Vancouver, Canada: Association for Computational Linguistics.

[Graesser et al. 2004] 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* 36:193–202.

[He, Zhou, and Jiang 2012] He, J.; Zhou, M.; and Jiang, L. 2012. Generating chinese classical poems with statistical machine translation models. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

[Hopkins and Kiela 2017] Hopkins, J., and Kiela, D. 2017. Automatically generating rhythmic verse with neural networks. In *Proceedings of the 55th Annual Meeting of the ACL (Volume 1: Long Papers)*, 168–178.

[Hovy 1990] Hovy, E. H. 1990. Pragmatics and natural language generation. *Artificial Intelligence* 43(2):153–197.

[Klare 1974] Klare, G. R. 1974. Assessing readability. *Reading research quarterly* 62–102.

[Lau et al. 2018] Lau, J. H.; Cohn, T.; Baldwin, T.; Brooke, J.; and Hammond, A. 2018. Deep-speare: A joint neural model of poetic language, meter and rhyme. *CoRR* abs/1807.03491.

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

[Liu et al. 2018] Liu, B.; Fu, J.; Kato, M. P.; and Yoshikawa, M. 2018. Beyond narrative description: Generating poetry from images by multi-adversarial training. In *Proceedings of the 26th ACM International Conference on Multimedia*, MM '18, 783–791. New York, NY, USA: ACM.

[Lynch 1998] Lynch, K. L. 1998. *Medieval Dream-Poetry*. Cambridge University Press.

[Mao et al. 2014] Mao, J.; Xu, W.; Yang, Y.; Wang, J.; Huang, Z.; and Yuille, A. 2014. Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv preprint arXiv:1412.6632*.

[Mohammad and Turney 2013] Mohammad, S. M., and Turney, P. D. 2013. Crowdsourcing a word-emotion association lexicon. 29(3):436–465.

[Oliveira 2009] Oliveira, H. 2009. Automatic generation of poetry: an overview. *Universidade de Coimbra*.

[Oliveira 2012] Oliveira, H. G. 2012. Poetryme: a versatile platform for poetry generation. *Computational Creativity, Concept Invention, and General Intelligence* 1:21.

[Plutchik 2014] Plutchik, R. 2014. *Emotions*. Psychology Press.

[Radford et al. 2018] Radford, A.; Narasimhan, K.; Salimans, T.; and Sutskever, I. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

[Radford et al. 2019] Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language models are unsupervised multitask learners. *OpenAI Blog* 1(8).

[Reiter and Dale 2000] Reiter, E., and Dale, R. 2000. *Building natural language generation systems*. Cambridge university press.

[Santhanam and Shaikh 2019] Santhanam, S., and Shaikh, S. 2019. A survey of natural language generation techniques with a focus on dialogue systems-past, present and future directions. *arXiv preprint arXiv:1906.00500*.

[Sennrich, Haddow, and Birch 2015] Sennrich, R.; Haddow, B.; and Birch, A. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

[Spearing 1976] Spearing, A. C. 1976. *The High Medieval Dream*. Stanford University Press.

[Vaswani et al. 2017] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is all you need. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc. 5998–6008.

[Vechtomova et al. 2018] Vechtomova, O.; Bahuleyan, H.; Ghabussi, A.; and John, V. 2018. Generating lyrics with variational autoencoder and multi-modal artist embeddings. *CoRR* abs/1812.08318.

[Wei, Zhou, and Cai 2018] Wei, J.; Zhou, Q.; and Cai, Y. 2018. Poet-based poetry generation: Controlling personal style with recurrent neural networks. In *2018 International Conference on Computing, Networking and Communications (ICNC)*, 156–160. IEEE.

[Weizenbaum and others 1966] Weizenbaum, J., et al. 1966. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM* 9(1):36–45.

[Wu et al. 2016] 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*.

[Yan et al. 2013] Yan, R.; Jiang, H.; Lapata, M.; Lin, S.-D.; Lv, X.; and Li, X. 2013. I, poet: automatic chinese poetry composition through a generative summarization framework under constrained optimization. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

[Zhang and Lapata 2014] Zhang, X., and Lapata, M. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 670–680.

[Zhang et al. 2019] Zhang, T.; Kishore, V.; Wu, F.; Weinberger, K. Q.; and Artzi, Y. 2019. Bertscore: Evaluating text generation with BERT. *CoRR* abs/1904.09675.

Enhancing Language Models with Knowledge Graph Embeddings

Andrew Conley
Rensselaer Polytechnic Institute (RPI)
110 Eighth Street
Troy, NY USA 12180
Email: conlea@rpi.edu

Jugal Kalita
University of Colorado Colorado Springs
1420 Austin Bluffs Pkwy
Colorado Springs, Colorado 80918
Email: jkalita@uccs.edu

Abstract

Most NLP tasks use word embeddings to improve performance. Breakthroughs like ELMo (Peters et al. 2018) and BERT (Devlin et al. 2018) have shown that state of the art results can be achieved in many NLP tasks through good language models, even without a task-specific architecture. Word vectors have been a simple, popular, and effective language model for years. Methods for generating these word vectors typically use unsupervised learning based on the context in which each word is used within the greater corpus. We propose new method of generating these word vectors. We use knowledge embeddings extracted from knowledge bases like Freebase and WordNet (Bordes et al. 2013) as a starting point, and introduce syntactic information captured from existing language models. By incorporating knowledge directly into the word embedding we aim to improve the task of natural language inference, similar to those achieved by applying knowledge bases to machine reading (Yang and Mitchell 2019). The new embeddings are judged primarily on their perfermance on the natural language inference model HBMP (Talman, Yli-Jyrä, and Tiedemann 2019). This performance is compared to that of Glove (Pennington, Socher, and Manning 2014), with the same architecture. No improvement was found to accuracy, however further steps to achieve the desired improvement are well defined.

Introduction

Many NLP tasks see improved performance through the use of a pre-trained language model. Recently, BERT has been used to achieve state of the art results on many NLP tasks, even without the use of extensive task specific architectures (Devlin et al. 2018). Common methods of generating word embeddings like Word2vec (Mikolov et al. 2013), GloVe (Pennington, Socher, and Manning 2014), and Fasttext (Bojanowski et al. 2017), assign a vector representation to each word in a particular vocabulary. Metrics like the Euclidean distance or cosine similarity between vectors are thought of as a pseudo measurement of similarity between words in the language model. Something to note about these processes is that they are built solely from the context in which a word is used within its corpus, and do not incorporate outside knowledge. This leads to results that contradict this notion of similarity. For example, using Word2vec yields a very small distance between the vector for "$Hello''$ and the vector for "$Dolly''$ despite the meanings of the two words being unrelated. We propose using altered knowledge graph embeddings as a language model for natural language inference in order to capture semantic information instead of just syntactic information.

Knowledge graphs are large graphs of triples made up of two entities (nodes) and a relation (edge) connecting them. These graphs are often highly curated and contain specific information about how various entities are related. When dealing with knowledge graphs, we often talk about the task of link prediction. Link prediction takes an existing knowledge graph and tries to predict relations between entities not present in the graph. Graph embeddings, a method of modelling entities and relations, as well as scoring triples, are used to achieve good results in link prediction(Trouillon et al. 2016)(Ding et al. 2018)(Bordes et al. 2013).

TransE (Bordes et al. 2013) is a model for graph embedding that represents entities as low dimensional vectors and relations as translations (low dimensional vector similar to those of entities) on those vectors. For head entity h, tail entity t, and relation (link) l, training minimizes the euclidean distance between $h + l$ and t, while maximizing the distance between $h + l$ and other entities.

The TransE model is particularly suited for creating a general language model since it models entities as low dimensional vectors, the same form used by methods like Word2vec, GloVe, and Fasttext. This allows TransE embeddings to replace these common language models without needing to alter the architecture used for downstream tasks. TransE also includes a constraint on entity embeddings, restricting their L2 norm to be equal to 1 This was done to prevent artificial reduction of the loss function by increasing distance between entities that do not have a relation by arbitrarily increasing the size of entity embeddings. These restrictions keep the embeddings in a smaller space, which keeps related entities closer together, much like Word2vec and similar language models.

Related Work

Generating good embeddings for rare words is a difficult task and big problem. Since generating word embeddings is often an unsupervised task, a large amount of data is required to create good embeddings. Rare words, by definition, do not appear often, and this lack of data often results in unreliable word vectors that do not accurately represent the

similarity of a rare word to other words in the vocabulary. Herbelot and Baroni (2017) describe an effective method to train these rare word vectors by altering word2vec. This method learns the majority of the vocabulary in accordance with the methods of word2vec, then holds the learned word vectors constant while learning the rare words with a higher rate of learning and a larger window size to take in as much context as possible. This paper also utilizes a smaller, curated subset of its corpus to train the rare words. This subset is made up of sentences from encyclopedic data, with context believed to be highly informative to the meaning of the rare word.

A La Carte Embedding (Khodak et al. 2018) builds off of this paper applying similar techniques to phrases. Notably, this word achieves low computational cost embedding chimeras (a combination of words to simulate a new rare word), and excellent results. Chimera representations for idioms have been shown to be very similar to their meaning, for example, the representation for beef up was (by cosine similarity) most similar to need and improve, and the representation for cutting edge was most similar to innovative and technology.

Another approach for improving rare word embeddings is the Fasttext language model (Bojanowski et al. 2017). Fasttext is a character based language model where words vectors are a sum of n-gram vectors. One major advantage of this is that out of vocabulary words, a huge problem for dictionary based models, can have approximate meanings guessed from the vector representation of prefixes, suffixes, and root words. This model creates word vectors that are very related to words that share a root, which both strongly indicates similarity linguistically and allows detection of similarity across languages that share roots (i.e. romance languages sharing Latin roots). This model has been combined with a Gaussian Distribution model (Athiwaratkun and Wilson 2018) which uses probability densities to be able to model multiple meanings of a single word. The result of the combination, Probabilistic Fasttext (Athiwaratkun, Wilson, and Anandkumar 2018) is able to accurately predict the meaning of rare, misspelt, and unseen words. This model seperates the representation of a word into various mixture component, each representing a different sense or meaning of the word. In doing so, this model has achieved state of the art results on tasks that require the use and differentiation of different meanings of a word.

There are currently many approaches to generating graph embeddings for the task of link prediction. One notable technique applies constraints to limit irrelevant or unwanted information taken from the knowledge graph when generating embeddings (Ding et al. 2018). These constraints came in two forms: a non-negativity constraint and approximate entailment constraints. The non-negativity constraint does not allow negative relations to be considered (i.e. a cat is not a car, or a frog is not an instrument). The effect of this constraint was a more sparse and interpretable set of entity vectors. The approximate entailment constraint allowed detection of entailment between relations. Using the ComplEx (Trouillon et al. 2016) model, Ding et al. were able to formulate an equation for detecting entailment with probability λ. This constraint alters the imaginary component of each vector to encode this entailment information without greatly affecting metrics of entity similarity in the vector space (distance and cosine similarity).

Embedding Model	Accuracy
GloVe	85.89%
TransE ConceptNet	71.35%
Word2vec	71.35%
TransE ConceptNet+Word2vec	71.35%
TransH ConceptNet+Word2vec	71.35%
DistMult ConceptNet+Word2vec	71.35%
Randomly Initialized	71.35%

Table 1: Results of each language model on snli dataset using HBMP model

Results

We propose a new method of altering knowledge graph embeddings to act as a language model for natural language inference. We approach this by replacing GloVe word vectors with standard knowledge embeddings of the knowledge graph Concept Net, and replacing those vectors with knowledge embeddings generated from altered knowledge graph. The quality of created language models is evaluated by performance on the natural language inference model described in (Talman, Yli-Jyrä, and Tiedemann 2019).

Initially, standard TransE embeddings were trained on Concept Net triples and applied to the HBMP model. Using OpenKE packages (Han et al. 2018), entities and relations are indexed and triples are split into a text file for training and a text file for testing, both of the form, entityid entityid relationid. TransE was chosen for this initial task because it models entities and relations from the knowledge graph as low dimensional vectors, similar to the GloVe embeddings to be replaced.

In an attempt to improve these embeddings, a new, modified set of triples was created in an attempt to include syntactic information from the successful language model word2vec. New triples were generated and added to Concept Net to incorporate the similarity found in an existing word2vec model. These triples were created by linking each word in the model's vocabulary (entities) with the word closest to it by cosine similarity, via a new context relation. In practice, this yields connections such as "$unamused''$ is related by context to "$bemused''$ or "$pastured''$ is related by context to "$Simmental_cattle''$. The word2vec model from which these connections were generated was downloaded pre-trained on the google news groups corpus.

As an extension to the previous improvement, we attempted to expand this to a general n most similar words by cosine similarity. These words would be connected by relations, Ċontext0, Context1, ..., Contextn̈respectively. Unfortunately, this proved too time consuming for the scope of this project, as these new sets of triples would have taken weeks to create.

We then moved away from TransE graph embeddings to TransH and DistMult. Both of these new embedding mod-

els have proven more effective than TransE at the task of link prediction, which would suggest it might generate a better general language model. TransH models entities in the same way as TransE and GloVe, allowing for simple substitution of the newly created vectors for the default GloVe vectors. TransH differs from TransE in that it models relations as a hyperplane, instead of just a translation. This allows for more intricate combinations of relations to be modeled and tested successfully in the task of link prediction. DistMult models relations as a bilinear operator between entities. This model is also a multiplicative graph embedding model, meaning in the task of link prediction, the scoring and prediction of triples is based on a multiplication of vectors instead of addition in the case of TransE and TransH. Multiplicative models tend to perform better than additive ones at link prediction, but are less similiar to conventional language models like GloVe.

The vectors we have been dealing with so far have all used 300 dimensions. However, DistMult embeds its entities with 100 dimensional vectors by default. Scaling the size of these vectors up to 300 during training would be possible, but not feasible given time constraints. In order to allow these embeddings to match the HBMP architecture, they were padded with values of 0 to reach 300 dimensions.

Table 1 shows the accuracy achieved on the NLI task for each set of embeddings described above. A set of randomly initialized vectors was also tested to use a baseline of a language model expected to cause no improvement. Every set of embeddings except the original GloVe embeddings achieved an accuracy of 71.35, while the GloVe embeddings achieved 85.89. The interpretation of these results will be discussed in the next section.

Discussion

Every novel language model tested produced the same result of 71.35. This accuracy is notably worse than the 85.89 observed using GloVe. We believe this occurred either due to little overlap of key vocabulary between the knowledge graph and NLI corpus, or because of unexpected behavior of the HBMP model when changing embeddings.

Assuming the HBMP model performed as expected, the poor results were likely due to the novel language models lacking reliable representations of common words from the NLI corpus. The knowledge graph Concept Net is particularly useful when it has been applied previously because it captures semantic information. General language models merely capture syntactic information, the general context each word is used in, but lacks any complex or deeper understanding. Concept Net is able to capture complex ideas used in speach. This is apparent by the presence of idioms as entities in its graph. For example, the phrase "$few_cards_shy_of_full_deck$" is present in Concept Net. The meaning of this phrase is not the same as the literal interpretation of those words in sequence. Therefore, the ability to leverage this complex concept should prove invaluable to natural language inference, a task that inherently requires some kind of understanding. The problem is that in Concept Net this idiom is represented as written above, several words connected by underscores and stored as a single entity. This is not the same format that would emerge in the NLI task. If this phrase was present at all, it would be a series of words without underscores, which the model would not equate to the representation of the idiom found in Concept Net. When we consider how common phrases or expressions are in Concept Net, and that many of connections present for individual words in Concept Net connect those words to phrases, it seems likely that Concept Net fails to create a densely connected network of words found in our downstream task. This interpretation is also supported by the performance of the randomly initialized vectors. Randomly initialized vectors should contain no useful information whatsoever, and therefore not enhance the performance of the NLI task. Since the same results are seen using our Concept Net models and random vectors, our vectors provide no improvement to the model.

If these results were caused primarily by an issue of vocabulary, we would expect including information from Word2vec to improve results. Word2vec trained on the google news corpus should share a large portion of vocabulary with the GloVe embeddings as well as the NLI task. However we observe the same poor results on each model that includes Word2vec based triples. In practice, merely 1 context relation did not add any significant information. When looking through the added triples by hand, it becomes clear that very often the nearest vector is too similar to the original vector to add any meaningful information. We see many relations connecting words like peice to piece (a misspelling) or connecting mage to mages (plural). If these words were present in Concept Net to begin with, it is likely that these relations are already represented. Also, it becomes evident that a single relation for each word cannot create good representations. If a word from Word2vec is not present in Concept Net, it is not meaningfully added to the vocabulary. Although it technically becomes included in the vocabulary, it will only be connected to its closest word, which is also likely not in the vocabulary. This yields a very sparse set of connections, which will net create useful vectors. However, even if the word is present in Concept Net, we are still only adding a single relation. It is possible that many relations (although expensive with regard to time) may improve these results, but that is not certain. Even with hundreds of relations, it is likely that the words from Word2vec will become significantly connected with other words from Word2vec and entirely unconnected to entities from Concept Net. In that case, Concept Net would not add anything to the language model, and performance would strive to match plain Word2vec, not surpass it.

All issues discussed above would also apply DistMult and TransH. If the issues already presented were not the cause of these poor results, it is possible that existing graph embedding methods are inherently ill-suited for acting as a general language model. Typical language models use unsupervised learning to group words used in similar contexts. This keeps similar words very close together in the vector space. Graph embeddings, on the other hand, require more specific information than just whether or not two words are related. Because a graph embedding needs to know the way in which

two entities are related, it often does not group similar vectors very close together. Instead these embeddings rely on using specific directions or other similar methods to distinguish between relations. Ambiguity is unwelcome in graph embeddings, as it is as important to be confident two entities do not share a relation as it is to know the entities do share a relation. Because of this, a greater distance between entities can often prove benefitial. The TransE paper specifical mentioned needing to add a constraint to prevent entities from drifting too far apart, which was found to happen often to arbitrarily reduce the loss function during training.

Alternatively, these poor results may be due to unexpected behavior involving changing the language model, or a failure to import the new vectors. If this is the case, many of the previously discussed issues are still relevant.

Concept Net includes many phrases and idioms. Assuming those phrases are connected to many other entities that can illustrate their meaning, those representations contain valuable information. However, that information cannot be accessed in its current state. Merely introducing connections between words in Word2vec will not cause the representation of a Concept Net phrase to be used in a NLI task if the phrase does not appear in the same form (underscores).

We also would not know if the graph embedding models are suitable to be used as a general language model. As stated earlier, graph embeddings often have more euclidean distance between vectors. This makes the vectors themselves more varied between related words. Since the layers of a neural network use matrix multiplication to generate outputs, intuition would suggest vectors that are made up of very similary values would produce very similar results with a neural NLI model. It is desireable to have similar words behave in similar ways, so loosing that quality may hurt performance when switching to a graph embedding.

Future Work

Given adequate time, we would like to explore including more relations from Word2vec as triples. We would also like to develop a method of connecting Word2vec entities to existing Concept Net entities. Recent work (Yang and Mitchell 2019) has shown good results applying attention to knowledge graph embeddings. Specifically their method of choosing embeddings to attend to, searching the graph for substring matches, is a promising system for choosing entities to relate to our vocabulary.

Once serious improvement is seen, we will experiment with restricting the information used from our knowledge graph. Work by (Ding et al. 2018) has shown that simple constraints, such as not including negative relations (is not a, does not have, etc.), can greatly improve performance in link prediction. One would expect a negative relation contradict the advantages of a language model, keeping similar words near each other in vector space, by reducing the distance between words that are known to be related by their dissimilarity, for instance, an Antonym relation.

We would still like to experiment with othergraph embedding models. Although the models we tried in this paper did not show any change, different embedding models can represent entities and especially relations in a myriad of ways, some of which should vastly outperform the others. In particular, a character based embedding may help to address the issue of formatting phrases, by being less particular about the presence or absence of underscores.

It would also be worthwile to look into a method of preprocessing the NLI dataset to isolate phrases and format them ahead of time to match their appearance in Concept Net. Concept Net is a large graph containing a large amount of semantic world knowledge. If we could specifically tailor our task to match Concept Net, much more of that information could likely be used.

We would also like to look into ways of incorporating information from knowledge graphs into language models beyond introducing new triples and generating a graph embedding. The original intent was to build off the work of nonce2vec (Herbelot and Baroni 2017) and from an existing set of graph embeddings, additionally selectively train the remainder of a desired vocabulary using word2vec. This approach was hidered by the requirement of a word2vec model to base training off of, instead of just vectors, where knowledge embedding vectors would have been easily substitutable. Should a solution to this issue be found, a definition dataset, required for nonce2vec, has already been prepared from a general corpus. There has also been discussion of simply concatonating the representation for each word created by glove and by a knowledge embedding into a single vector to be fed into the downstream model. This method should also capture semantic information from the knowledge graph, but would require modification of the downstream architecture and involve higher dimensional vectors.

Acknowledgement

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

References

Athiwaratkun, B., and Wilson, A. G. 2018. Hierarchical density order embeddings. *arXiv preprint arXiv:1804.09843*.

Athiwaratkun, B.; Wilson, A. G.; and Anandkumar, A. 2018. Probabilistic fasttext for multi-sense word embeddings. *arXiv preprint arXiv:1806.02901*.

Bojanowski, P.; Grave, E.; Joulin, A.; and Mikolov, T. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5:135–146.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, 2787–2795.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ding, B.; Wang, Q.; Wang, B.; and Guo, L. 2018. Improving knowledge graph embedding using simple constraints. *arXiv preprint arXiv:1805.02408*.

Han, X.; Cao, S.; Lv, X.; Lin, Y.; Liu, Z.; Sun, M.; and Li, J. 2018. Openke: An open toolkit for knowledge embedding. In *Proceedings of EMNLP*, 139–144.

Herbelot, A., and Baroni, M. 2017. High-risk learning: acquiring new word vectors from tiny data. *arXiv preprint arXiv:1707.06556*.

Khodak, M.; Saunshi, N.; Liang, Y.; Ma, T.; Stewart, B.; and Arora, S. 2018. A la carte embedding: Cheap but effective induction of semantic feature vectors. *arXiv preprint arXiv:1805.05388*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Talman, A.; Yli-Jyrä, A.; and Tiedemann, J. 2019. Sentence embeddings in nli with iterative refinement encoders. *JNLE*.

Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, É.; and Bouchard, G. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, 2071–2080.

Yang, B., and Mitchell, T. 2019. Leveraging knowledge bases in lstms for improving machine reading. *arXiv preprint arXiv:1902.09091*.

PixelMRF
A Deep Markov Random Field for Image Generation

Joshua Frederick
Departments of Computer Science & Mathematics
California Polytechnic State University
San Luis Obispo, California, USA
jmfreder@calpoly.edu

Jonathan Ventura
Department of Computer Science
California Polytechnic State University
San Luis Obispo, California, USA
jventu09@calpoly.edu

Abstract

In this work we introduce a new graphical model for generative image modeling. The popular PixelCNN has utilized deep graphical models to represent images as directed, acyclic Markov chains. The limitations enforced by the directed, acyclic nature of PixelCNN have restricted the model to conceptualize an image as a sequence of directional dependencies, where each pixel is conditionally dependent on the pixels before it in row-major order. We instead propose PixelMRF, a deep graphical model that fully captures the conditional distributions for each pixel of an image. PixelMRF achieves this goal by representing the pixels in an image as a undirected Markov random field, and so allows the dependencies that PixelCNN lacks.

Introduction

The need for powerful generative modeling is clear from consistent growth in area. One such highly used model is PixelCNN [van den Oord, Kalchbrenner, and Kavukcuoglu, 2016]. The autoregressive network, PixelCNN is a convolutional neural network (CNN) that represents the joint distribution of an image X as the product of conditional distributions

$$p(X) = \prod_{i=1}^{n^2} p(x_i | x_1, \ldots, x_{i-1})$$

where x_i is the ith pixel in row major order and X is a $n \times n$ image (see Figure 1 for a visual representation).

The network utilizes a series of masked convolutional layers with residual connections to guarantee that each pixel is only dependent on the prior pixels. The mask is implemented by taking the Hadamard product between each convolution kernel and a matrix that is 1 in each position prior to the central position and 0 otherwise, an example is given in Figure 2 [van den Oord, Kalchbrenner, and Kavukcuoglu, 2016].

Networks with the structure of purposely obscuring information from the receptive field of each feature are commonly denoted as *blind-spot networks*. To our knowledge, this terminology was constructed in [Krull, Buchholz, and Jug, 2018], and expanded in [Laine, Lehtinen, and Aila, 2019]. Similarly to [Laine, Lehtinen, and Aila, 2019], we adopt this terminology but note that the term "blind-spot" has a contradictory meaning in PixelCNN literature, where it denotes an unintentional obscuring of the receptive field. For the sake of concreteness of the blind-spot concept, note that the blind spot for a pixel x_i in PixelCNN is all posterior pixels of x_i and x_i itself. We question whether this blind-spot formulation given by PixelCNN is limiting its effectiveness, and so through this work suggest a reconsideration that we believe will achieve better performance.

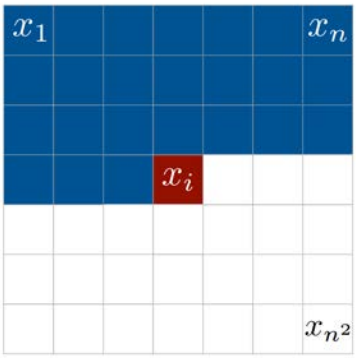


Figure 1: A visual representation of the dependencies of pixel x_i [Harshit Sharma, 2017]

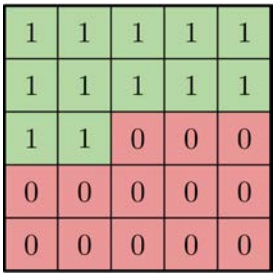


Figure 2: An example of a mask used in PixelCNN [van den Oord et al., 2016b]

Previous Works

Before continuing, we consider some prior improvements made to PixelCNN. Soon after its inception in 2016, two improvements to PixelCNN were suggested. An extension of the team that developed the original model improved their orignal architecture to the Gated PixelCNN model [van den Oord et al., 2016b] by adding gated activation such as in a LSTM. Then a series of improvements was suggested in [Salimans et al., 2017], ultimately leading to the current PixelCNN++.

One important attribute of PixelCNN is that it formally optimizes the log-likelihood of the training data unlike other forms of generative modeling such as GANs. Prior to PixelCNN, other generative models sought to generate images by optimizing the log-likelihood [Theis and Bethge, 2015; van den Oord and Schrauwen, 2014; Dinh, Krueger, and Bengio, 2015].

After PixelCNN in 2018, a Berkeley team used self-attention together with the traditional convolutional networks of PixelCNN to achieve state-of-the-art results on a number of standard data sets [Chen et al., 2018]. Most recently an OpenAI team took the "attention is all you need" philosophy of [Vaswani et al., 2017] and used sparse transformers to achieve the current state-of-the-art performance on several data sets [Child et al., 2019]. On the topic of Markov random fields in image generation, work in [Wu, Lin, and Tang, 2016] explored the utilization of Markov random fields for local image modeling and super-resolution.

Motivation

The chain rule for probability motivates and validates PixelCNN's representation of the joint distribution of an image. However, the representation leads to certain restriction. For an individual pixel x_i, PixelCNN restricts x_i to be viewed as conditionally dependent on only the pixels prior to itself, and so fails to represent the underlying conditional distribution of each x_i. Indeed, to fully express the dependencies of x_i we would desire to model each pixel distribution as

$$p(x_i|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{n^2}) := p(x_i|x_{-i})$$

and so model the joint distribution of the image as

$$p(X) = \frac{1}{Z} \cdot \tilde{p}(X) = \frac{1}{Z} \prod_{i=1}^{n^2} p(x_i|x_{-i})$$

where $\tilde{p}(x)$ is the pseudo-likelihood and Z is a normalizing constant.

Implementation

The representation of $p(X)$ presented previously is that of a Markov random field (MRF). Indeed, this realization provides the motivation for the name PixelMRF. We follow the following explicit implementation. Jointly, use two PixelCNNs to represent the conditional distribution of x_i, one model for the pixels prior to x_i in X and another reversed PixelCNN to represent the dependencies of x_i on later pixels in X. This joint model is visually represented in Figure 3. We additionally notice that this architecture is a blind-spot

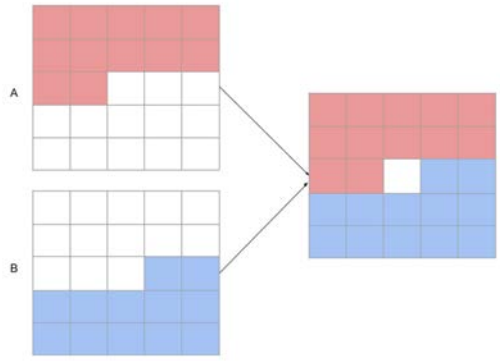


Figure 3: A visual representation of the receptive field of two PixelCNNs and the resulting receptive field of PixelMRF, where two PixelCNNs are working in unison.

network where the receptive field does not contain the center pixel. As such, we consider that for the sake of further exploration we could also consider the architecture presented in [Laine, Lehtinen, and Aila, 2019]. One issue with this representation is that computing the normalizing constant Z is intractable. As noted in [Goodfellow, Bengio, and Courville, 2016], one could represent the distribution using the pseudo-likelihood accepting that

$$p(X) \approx \prod_{i=1}^{n^2} p(x_i|x_{-i})$$

essentially ignoring the normalizing constant. Instead, we use the Markov chain Monte Carlo (MCMC) methods of contrastive divergence to confront the intractability of Z.

Contrastive Divergence

To motivate the process of contrastive divergence, notice that

$$\begin{aligned} \log p(X) &= \log\left(\frac{1}{Z} \cdot \tilde{p}(X)\right) \\ &= \log \tilde{p}(X) - \log Z \\ &= \log \tilde{p}(X) - \log \sum_{X'} \tilde{p}(X'). \end{aligned} \tag{0}$$

Thus to maximize the log-probability of our training data, we can maximize the pseudo-log-likelihood the training data while minimizing the weight of the total distribution

$$\sum_{X'} \tilde{p}(X').$$

Notice that this negative term is the only difference from simply minimizing the pseudo-log-likelihood. The process to approximate the objective given in equation (0) optimization is contrastive divergence, which we now state in general terms:

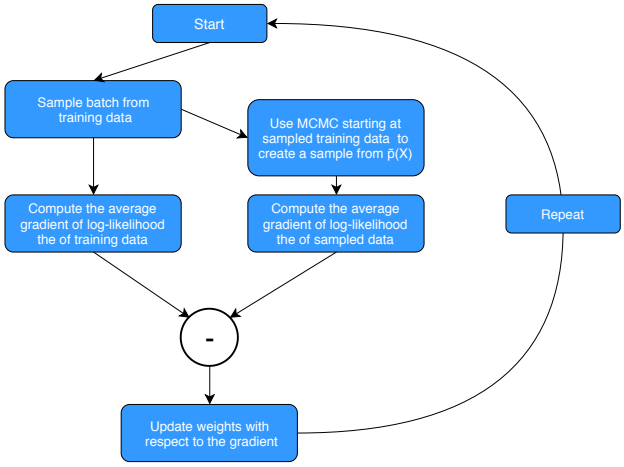


Figure 4: A visual representation of the contrastive divergence algorithm

Algorithm 1 Contrastive Divergence

1: **for** $k \leftarrow 1$ **to** maximum of iterations **do**
2: Sample batch of "real" data X from training data
3: Sample batch of "fake" data X' from model
4: Compute mean gradient ∇ for log-likelihood of X
5: Compute mean gradient ∇' for log-likelihood of X'
6: Update weight with respect to $\nabla - \nabla'$
7: **end for**

In Figure 4, we give a visual representation of the contrastive divergence algorithm that we believe better displays the process. After attempting to train PixelMRF by optimizing the object given in equation (0), we relaxed the objective to

$$\log \tilde{p}(X) - \alpha \log \sum_{X'} \tilde{p}(X') \tag{1}$$

where

$$0 \leq \alpha \leq 1.$$

In the context of equation (1), the variable α can be seen as a term to compromise between optimizing the psuedo-loglikelihood and the contrastive divergence objective. To see why this change was made, note in figures 12 and 13 the instability and subsequent inability to train PixelMRF using the objective given by equation (0).

Hamiltonian Monte Carlo

For the "fake" sampling portion of contrastive divergence (see line 3), we use the Hamiltonian dynamic based sampling method discussed in [Neal, 2010]. Hamiltonian Monte Carlo sampling utilizes the gradient of the log-likelihood to take informed steps, and so allows a shorter number of steps for a fully converged sample. Additionally, we note that for further consideration, future work could consider other state-of-the-art MCMC methods such as the non-trivial improvements to Hamiltonian MC: DeepHMC [Levy, Sohl-dickstein, and Hoffman, 2018], NUTS (No U-Turn Sampler)

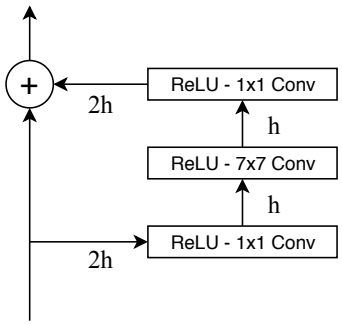


Figure 5: A layer in a PixelCNN. A total of 16 such layers were used in each PixelCNN.

[Homan and Gelman, 2014], or the replica exchange (parallel tempering). In a later section we discuss why exploring different sampling methods may be needed, but the need can be seen in the poor quality of the samples in Figure 11.

Model Configuration

Our model architecture closely follows the model implementation given in [van den Oord, Kalchbrenner, and Kavukcuoglu, 2016]. Of course our implementation doubles that of PixelCNN, as PixelMRF contains two PixelCNNs working in unison. For each PixelCNN, all but the first layer and the final layer take the form given in Figure 5, where $h = 32$ and $2h = 64$ are the number the filters and the convolution is appropriately masked. The first layer of the PixelCNNs is a simple masked convolutional layer that additionally masks the center pixel to obscure it from the receptive field. Due to the everywhere-differentiable regularity requirements of HMC, unlike PixelCNN and later models, we enforce a Gaussian prior on PixelMRF. As such, the last layers are convolutional layers to correctly shape the output to produce a mean and standard deviation for each dimension in the input image.

For the configuration of Hamiltonian Monte Carlo we used 2 leapfrog integrator steps and an adaptive step size with an initial value of 0.1. Additionally due to computational limitations, we had only 10 burnin steps, and so to handle a zero HMC acceptance rate, noise sampled from $\mathcal{N}(0, 0.05)$ was added to the "fake" data. Moreover, we used a non-standard training scheme. We start by training each PixelCNN in the traditional form as in [van den Oord, Kalchbrenner, and Kavukcuoglu, 2016] for 400 epochs. We follow this by optimizing the pseudo-loglikelihood for 100 epochs and then the full contrastive divergence loss with $\alpha = 10^{-6}$ for another 100 epochs. We additionally tested the PixelSNAIL architecture of [Chen et al., 2018]; however, due to our computational limit this did not lead to a noticeable improvement, and so was not included in the final results.

Evaluation

Finally, on the discussion of evaluating PixelMRF, we evaluated PixelMRF on the Frey data set. To properly evalu-

Figure 6: Sampling from PixelCNN using ancestral sampling.

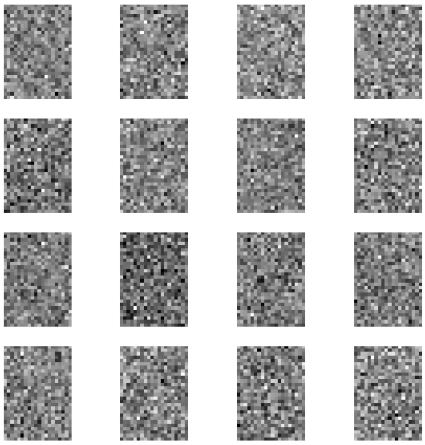


Figure 7: Sampling from PixelCNN using HMC sampling.

ate the performance of or MCMC method, we use the following process: train a PixelCNN in the traditional fashion and evaluate the sampling performance of the the MCMC method for our selection of hyperprameters. This allows us to decouple the evaluation of our contrastive divergence parameters and PixelMRF's architecture from the evaluation of MCMC methods. As mentioned previously, one consequent obstacle we had to overcome is that all of PixelCNN, GatedPixelCNN, and PixelCNN++ model some portion of $p(X)$ as a discrete distribution, and most advanced MCMC methods, such as Hamiltonian MC and NUTS, have everywhere-differentiable regularity requirements. We handled this obstacle by instead enforcing a Gaussian prior on both PixelCNN and PixelMRF when testing MCMC methods.

As both PixelCNN and PixelMRF are attempting to represent the joint distribution $p(X)$, it would be desirable to directly compare their performance. A hurdle to this comparison is realization that due to the intractability of computing Z, computing the exact negative log-likelihood for PixelMRF is impossible. At the current time, we instead use qualitative, visual evaluation of the generated samples. In future work we hope to overcome this limitation, and so we consider the use of annealed importance sampling (AIS) to estimate Z [Neal, 2001], and so compare approximate negative log-likelihood as in [van den Oord, Kalchbrenner, and Kavukcuoglu, 2016].

We now move on to concrete examples of performance on the Frey data set. In Figure 6 and Figure 7, we provide a comparison of images generated by PixelCNN. We have sharp but sometimes distorted images produced by standard ancestral sampling and complete noise produced Hamiltonian Monte Carlo sampling. This absolute failure of Hamiltonian Monte Carlo sampling is not ideal, but the success of the ancestral sampling does confirm that a Gaussian prior can adequately perform on the Frey data set.

With that knowledge we can assume the noise is a result of the Hamiltonian Monte Carlo sampling. In figures 8 and 9, we provide a display of the effect of five steps of HMC

Figure 8: Five steps of the PixelCNN based Hamiltonian Monte Carlo initialized to training examples.

Figure 9: Five steps of the PixelMRF based Hamiltonian Monte Carlo initialized to training examples.

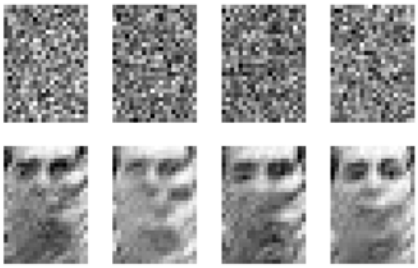


Figure 10: Samples from PixelMRF using the Gibbs sampling procedure. Top: the starting noise. Bottom: the produced sample

sampling initialized at training examples for both PixelCNN and PixelMRF. We see that the HMC process appears to be adding noise to the examples and is failing to step to reasonable, consistent examples. In Figure 11 we see images generated by a trained PixelMRF and Hamiltonian Monte Carlo. Similar to the previous PixelCNN example, these images have no clear relation to the training data and appear to be noise, suggesting that Hamiltonian Monte Carlo is to blame for the noisy generation by PixelMRF. In contrast, in Figure 10 we have samples from PixelMRF produced by the Gibbs sampling procedure. These images are lower in quality than the PixelCNN samples, but have clear learned structure.

Issues and Improvements

As we discussed previously and can see in Figure 11, PixelMRF with Hamiltonian Monte Carlo sampling currently has extremely poor performance. We have reasonable explanations as to why this may be the case. To begin, in

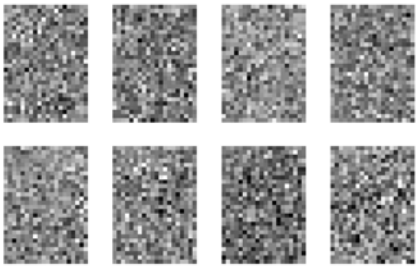


Figure 11: Samples from PixelMRF using the HMC sampling procedure. Top: the starting noise. Bottom: the produced sample

[Neal, 2010], Neal states that the performance of Hamiltonian Monte Carlo decreases heavily in a high dimensional setting. Even in 28x20 grayscale images such as in the Frey data set our image space still has 560 dimensions. This realization could explain a portion of our poor Hamiltonian Monte Carlo performance. This is further supported by the success of Gibbs sampling in Figure 10. This comparison and the eventual zero acceptance rate of HMC suggests that one possible solution would be utilize Gibbs sampling in place of HMC sampling in the training procedure. The main issue with using Gibbs instead HMC is time; Gibbs sampling is sequential and so cannot parallelized like HMC sampling, and so becomes computational unfeasible with large images.

Additionally, the log-likelihood estimate utilized by Hamiltonian Monte Carlo is given by the model. Thus, as the performance of the model on the training data increases, the log-probability distribution becomes flatter at training examples and the gradients of log-probability goes to zero. Consequently, the acceptance rate of Hamiltonian Monte Carlo at the training data goes to zero, and so the difference in gradients given in line 6 goes to zero. Once this occurs, the PixelCNN models underlying PixelMRF fail to learn. In the case of our training, this zeroing of the acceptance rate occurs only a few batches into training. Additionally, the acceptance rate naturally decreases exponentially in the number of dimension, and as discussed previously, image data is necessarily high dimensional [Neal, 2001]. This fact means that even without a trained model we would expect some acceptance rate issues with HMC on our data.

Finally, another issue is the instability of contrastive divergence. Recalling the objective given in (0), we see that contrastive divergence training is inherently adversarial. In cases where the PixelMRF successfully trains, we have noisy loss functions such as in Figure 12. However, more frequently the model chooses to minimize the log-probability of the entire distribution including the training data, instead of correctly maximizing the log-probability of the training data while minimizing the mass of the total distribution. We can see an example of this failed training in Figure 13.

In terms of solutions for these issues, we believe that the

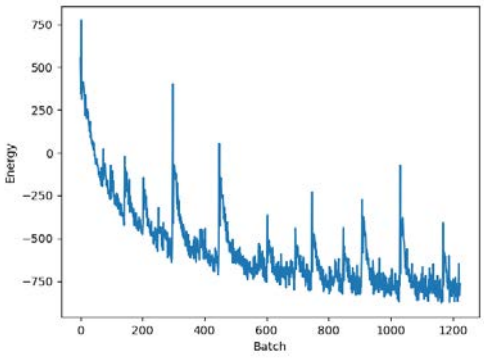


Figure 12: A plot of the loss over time in the early stages of a successful PixelMRF training.

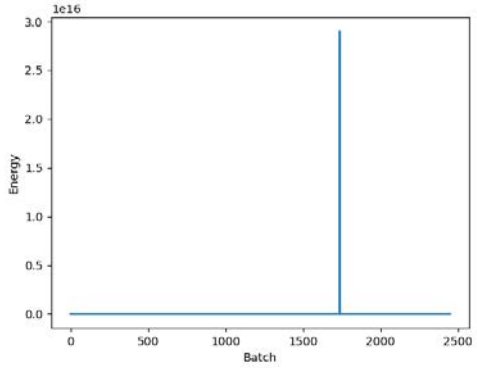


Figure 13: A plot of the loss over time in an unsuccessful PixelMRF training.

MCMC methods discussed previously could solve these issues. DeepHMC parameters Hamiltonian Monte Carlo in the form of a deep network that learns the appropriate parameters to optimize proposals and the acceptance rate for a given data set. Replica exchange Monte Carlo adds a temperature parameter to Hamiltonian Monte Carlo which adds a probability for Hamiltonian Monte Carlo to accept a worse state. This could help solve the issue of the model failing to learn after a small amount of time. Finally, NUTS would provide a fully adaptive sampler no u-turns sampler that would possibly solve all but the problems associated with the dimensionality of our data.

Conclusion

The autoregressive model PixelCNN has been used in many various applications [Kolesnikov and Lampert, 2016][Fauw, Dieleman, and Simonyan, 2019]. Additionally, variants of PixelCNN have been developed for domains such as audio [van den Oord et al., 2016a] and text [Dauphin et al., 2016]. However, PixelCNN has limitation that stem from its acyclic conceptualization of pixel dependencies. Through PixelMRF we represent pixel dependencies completely and confront the resulting normalizing constant through contrastive divergence. At the current time, the performance of PixelMRF is poor. However by exploring the samplers discussed in the document, we believe this novel representation can be made to improve the performance of the already powerful model, PixelCNN.

References

[Chen et al., 2018] Chen, X.; Mishra, N.; Rohaninejad, M.; and Abbeel, P. 2018. Pixelsnail: An improved autoregressive generative model.

[Child et al., 2019] Child, R.; Gray, S.; Radford, A.; and Sutskever, I. 2019. Generating long sequences with sparse transformers. *CoRR* abs/1904.10509.

[Dauphin et al., 2016] Dauphin, Y. N.; Fan, A.; Auli, M.; and Grangier, D. 2016. Language modeling with gated convolutional networks. *CoRR* abs/1612.08083.

[Dinh, Krueger, and Bengio, 2015] Dinh, L.; Krueger, D.; and Bengio, Y. 2015. NICE: non-linear independent components estimation. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*.

[Fauw, Dieleman, and Simonyan, 2019] Fauw, J. D.; Dieleman, S.; and Simonyan, K. 2019. Hierarchical autoregressive image models with auxiliary decoders. *CoRR* abs/1903.04933.

[Goodfellow, Bengio, and Courville, 2016] Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[Harshit Sharma, 2017] Harshit Sharma, S. M. 2017. Auto-regressive generative models (pixelrnn, pixelcnn++). https://towardsdatascience.com/ auto-regressive-generative-models-pixelrnn-pixelcnn-32d192911173. Accessed: 2019-06-07.

[Homan and Gelman, 2014] Homan, M. D., and Gelman, A. 2014. The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. *J. Mach. Learn. Res.* 15(1):1593–1623.

[Kolesnikov and Lampert, 2016] Kolesnikov, A., and Lampert, C. H. 2016. Deep probabilistic modeling of natural images using a pyramid decomposition. *CoRR* abs/1612.08185.

[Krull, Buchholz, and Jug, 2018] Krull, A.; Buchholz, T.; and Jug, F. 2018. Noise2void - learning denoising from single noisy images. *CoRR* abs/1811.10980.

[Laine, Lehtinen, and Aila, 2019] Laine, S.; Lehtinen, J.; and Aila, T. 2019. Self-supervised deep image denoising. *CoRR* abs/1901.10277.

[Levy, Sohl-dickstein, and Hoffman, 2018] Levy, D.; Sohl-dickstein, J.; and Hoffman, M. 2018. Generalizing hamiltonian monte carlo with neural networks.

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

[Neal, 2010] Neal, R. M. 2010. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo* 54:113–162.

[Salimans et al., 2017] Salimans, T.; Karpathy, A.; Chen, X.; and Kingma, D. P. 2017. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *CoRR* abs/1701.05517.

[Theis and Bethge, 2015] Theis, L., and Bethge, M. 2015. Generative image modeling using spatial lstms. 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. 1927–1935.

[van den Oord and Schrauwen, 2014] van den Oord, A., and Schrauwen, B. 2014. Factoring variations in natural images with deep gaussian mixture models. In Ghahramani, Z.; Welling, M.; Cortes, C.; Lawrence, N. D.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc. 3518–3526.

[van den Oord et al., 2016a] van den Oord, A.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A. W.; and Kavukcuoglu, K. 2016a. Wavenet: A generative model for raw audio. *CoRR* abs/1609.03499.

[van den Oord et al., 2016b] van den Oord, A.; Kalchbrenner, N.; Vinyals, O.; Espeholt, L.; Graves, A.; and Kavukcuoglu, K. 2016b. Conditional image generation with pixelcnn decoders. *CoRR* abs/1606.05328.

[van den Oord, Kalchbrenner, and Kavukcuoglu, 2016] van den Oord, A.; Kalchbrenner, N.; and Kavukcuoglu, K. 2016. Pixel recurrent neural networks. *CoRR* abs/1601.06759.

[Vaswani et al., 2017] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is all you need. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc. 5998–6008.

[Wu, Lin, and Tang, 2016] Wu, Z.; Lin, D.; and Tang, X. 2016. Deep markov random field for image modeling. *CoRR* abs/1609.02036.

Self-Supervised Deep Learning for Fluorescence Microscopy Denoising

Sonia Rao
University of Georgia
Athens, GA, USA
sonia.rao25@uga.edu

Jonathan Ventura
California Polytechnic State University
San Luis Obispo, CA, USA
jventu09@calpoly.edu

Guy Hagen
University of Colorado
Colorado Springs, CO, USA
ghagen@uccs.edu

Abstract

This work modifies existing self-supervised denoising architectures for Poisson noise typical within low-intensity Fluorescence Microscopy (FM) imagery. Biologists are often required to use low-exposure imaging technology, such as Fluorescence Microscopy, to preserve the integrity of sensitive samples. While these advancements have allowed for broader biological analysis, the collected data is usually corrupted by noise that obscures valuable biological insight. Furthermore, ground truth images of live cells are nearly impossible to gather without causing permanent damage to the observed specimen. FM data is unique in that the signal captured is both inherently weak and dominated by Poisson noise rather than Gaussian noise characteristic of traditional optical microscopy. Although many deep learning denoising algorithms exist, namely self-supervised methods in which elusive ground-truth images are not needed, no models directly address the Poisson component abundant in FM data. This work utilizes a novel non-parametric probabilistic prior and posterior approximation to restore FM imagery specifically dominated by Poisson noise.

Introduction

Fluorescence Microscopy (FM) has allowed biologists to observe imperative cellular processes that are otherwise obscured from traditional optical microscopes. An examined specimen must be stained with a fluorescent label, if not inherently capable of expressing a fluorescent protein, and is then excited with low-exposure or extremely brief light sources to observe the specimen while also preserving its biological integrity. The low-intensity imaging process, captured by Confocal, Two-Photon, or Wide-Field lenses, results in heavily noisy images as shown in Figure 1. Because the amount of photons entering the imaging space is discrete, FM imagery contains predominantly Poisson noise in contrast to conventional Gaussian noise found in optical imaging (Zhang et al. 2018). The phenomenon of Poisson noise coupled with low-intensity light limitations presents a unique challenge for the necessary task of FM image denoising.

Deep learning algorithms have traditionally had success restoring corrupted images; however, most networks require both noisy data and their corresponding ground-truth images to implicitly learn a mapping process (Remez et al. 2017). This is typically impossible as sensitive specimens cannot be safely photographed for adequate time or exposure-level necessary to capture a usable ground-truth signal. Recent work has addressed this limitation by introducing 'self-supervised' models in which only noisy data and relevant latent visual context are used together during training (Doersch and Zisserman 2017). Contemporary self-supervised denoising algorithms have utilized independent noise realizations, pixel receptive fields as supervisory context, or probabilistic estimation to restore corrupted images. While these existing self-supervised models have performed exceptionally well on traditional optical imagery, most models fail to generalize within Poisson dominated noise (Laine et al. 2019). We expand recent algorithms to specifically address the Poisson component present in FM imagery; by implementing a novel loss function to estimate a probabilistic prior distribution of clean pixels, we can restore FM imagery given only noisy training data.

Related Work

Prior to self-supervised methods, several deep network architectures have been proposed to learn clean signals from noisy data on a variety of corrupted data. U-Nets, a deep learning architecture derived specifically for biomedical use (Ronneberger, Fischer, and Brox 2015), and Residual Neural Networks (Heinrich, Stille, and Buzug 2018) have achieved great success on denoising tasks where ground truth data is readily available for training. In the case of sensitive low-lighting imagery, the acquisition of ground truth data is avoided by either self-supervised and unsupervised methods.

Contemporary self-supervised denoising models have made significant advancements towards quality of fully supervised deep learning models. Work by Lehtinen et al. 2018, dubbed Noise2Noise (N2N), introduced robust self-supervision by using pairs of independently realized corrupted images for training. The underlying principle is that two individual noisy images, whose re-

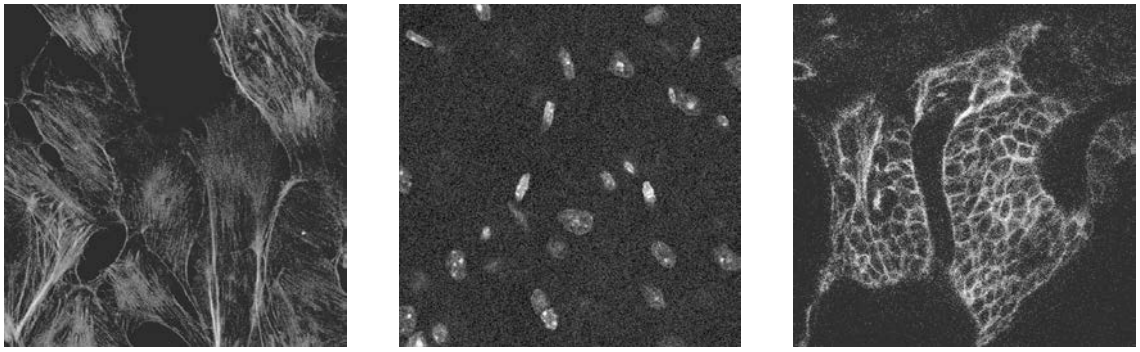


Figure 1: Noisy imagery of Bovine Pulmonary Artery Endothelial proteins, Mice cells, and Zebrafish tissue captured with a Fluorescence Microscope (Zhang et al. 2018).

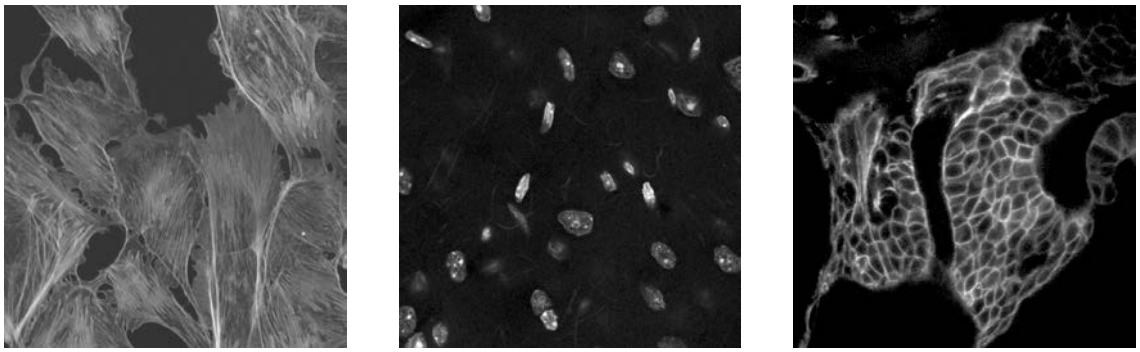


Figure 2: Averaged clean imagery of Bovine Pulmonary Artery Endothelial proteins, Mice cells, and Zebrafish tissue (Zhang et al. 2018).

spective noise distributions are independent, averaged pixel-wise is likely to render a clean image; results rivaled those trained on clean ground-truth signals. However, acquiring two renditions of the same Frame of View (FOV) would require that the observed specimen is captured twice in the same instant; this is biologically infeasible for most sensitive specimens that are inspected with FM.

Noise2Void (N2V) resolves N2N's limitation by using only one noisy image for training in conjunction with a novel blind-spot network (Krull, Buchholz, and Jug 2018). Convolutional Neural Networks (CNN) employ a receptive field typically used to influence predictions of its center pixel. By obscuring the center pixel from its receptive field, the network avoids directly learning a potentially noisy pixel identity. The center pixel instead mimics a randomly chosen pixel from its receptive field. While N2V compares closely to N2N regardless of the loss of information during training, N2V does not exploit possible information from the center pixel identity. This constraint is examined by Laine et al. in their contemporary blind-spot architecture.

Laine et al. 2019 expand on N2V's blind-spot concept while further improving training efficacy and restoration accuracy. This state-of-the-art self-supervision constructs a blind spot network by creating four cropped rotations of the same noisy image, thereby restricting the receptive field without limiting the loss function's scope. This work further estimates uncorrupted signal values through Maximum a Posteriori (MAP) estimation; MAP estimation combines probabilistic network outputs with the observed pixel values to yield more informed pixel predictions without relying heavily on a potentially noisy observed pixel value. Although this work is an exceptional estimation of Gaussian corruption, the model worsens when faced with low-intensity FM imagery.

Probabilistic Noise 2 Void (PN2V) specifically addresses the application of biomedical low-intensity imagery denoising (Krull, Vicar, and Jug 2019). PN2V's deep neural network outputs 800 possible signal prior distributions by using a histogram-based noise distribution. While this method is robust to varying noise-types and intensities, the noise distribution is constructed using pairs of clean and noisy signals such that emulated or collected ground truth imagery is necessary for both training and final restoration; we do not consider this work to be self-supervised and instead target N2V's metrics as a baseline. During our experimentation process, we recreate PN2V's results and use the described histogram-based noise model to evaluate our network on the same data.

Using elements from several of these works, we attempt to create a model that will discriminate signal from noise using only one corrupted image.

Implementation Theory

This work utilizes a contemporary blind-spot architecture, designed specifically for Poisson-dominated imagery, in conjunction with Maximum a Posteriori (MAP) estimation and Expected Value of the Posterior (EVP) to reconstruct multiple classes of FM images. Since all imagery within our dataset is comprised of 8-bits, we treat each possible pixel identity as a discrete value ranging from 0 to 256. Given observed pixel z_i sampled from a Poisson distribution and hypothetical ground truth pixel x_i sampled from X, our underlying goal is to identify the clean pixel value using information from Z.

$$\hat{x_i} = \arg\max_x P(X_i = x_i | Z_i = z_i), \tag{1}$$

However, we are unable to evaluate (1) directly; by providing the noisy pixel z_i, a network would learn a direct mapping of the noisy pixel identity to the estimated true signal value rather than reasonable estimates of clean pixel values (Krull, Buchholz, and Jug 2018). Instead, we must marginalize clean pixel probabilities, $P(X_i = x_i)$, from the joint distribution of noisy and clean pixels. $P(X_i = x_i)$ acts as a prior belief about the spread of unattainable ground truth pixel values; using knowledge of the Poisson probability mass function $P(Z_i = z_i | X_i = x_i)$ we create a network to output clean priors without directly learning pixel identities. We are then able to compute a probability distribution around the observed pixels (2).

$$P(Z_i = z_i) = \sum_x P(Z_i = z_i | X_i = x_i) P(X_i = x_i) \tag{2}$$

We begin our model by constructing a blind-spot network, a modified Convolutional Neural Network in which the center pixel is rotationally obscured from its receptive field (Laine et al. 2019) 3. With a noisy image as input, we modify the network's loss function such that computed priors model pixel probabilities at every possible value of Poisson component λ, the discrete rate at which photons are absorbed by the imaging sensor (3).

$$P(Z_i = z_i | X_i = x_i) = \frac{x^z \exp(-x)}{z!} \tag{3}$$

The model's softmax output is tuned by minimizing the averaged negative log likelihood of $P(Z_i = z_i)$ combined with an entropy regularization measure of prior belief uncertainty (4). The entropy term is multiplied by an arbitrary constant.

$$\mathcal{L}_i = \frac{1}{N} \sum_N - \log P(Z_i = z_i) + S_x \tag{4}$$

With both the ground truth prior and joint probability density function marginalized, we apply Bayes' transformation on (1) to make our objective function determinable.

$$\hat{x_i} = \arg\max_x \frac{P(Z_i = z_i | X_i = x_i) P(X_i = x_i)}{P(Z_i = z_i)} \tag{5}$$

To avoid information loss, we further expand on this estimation to include information about the observed center pixel z_i. Maximum a Posteriori estimation provides the Posteriori approximation with the highest likelihood (6).

$$\hat{x_i} = \arg\max_x P(Z_i = z_i | X_i = x_i) P(X_i = x_i). \tag{6}$$

Expected Value of the Posterior estimation computes the expected value over each λ value of the normalized posterior distribution (7).

$$\hat{x_i} = E[P(Z_i = z_i | X_i = x_i) P(X_i = x_i)]. \tag{7}$$

Data and Evaluation

A large Fluorescence Microscopy Denoising Dataset (FMDD) was generated by (Zhang et al. 2018) specifically for the task of Gaussian-Poisson denoising. FMDD consists of various specimens at the cellular level captured by Confocal and Two-Photon microscopes. For each specimen and imaging modality pair, FMDD contains 20 Frames of View (FOV) each with 50 captures. While the content of each FOV differs, the captures of each FOV are only different in their individual realizations of Poisson noise. Zhang et al. simulate ground truth by averaging all 50 captures per FOV; by assuming noise independence for each capture, the average yields an estimated clean image **??**.

In addition to averaged ground truth, Zhang et al. provide five degrees of averaged images to estimate varied levels of noise typically present in biological imagery. These categories are referred to as Noise Regimes (NR) ranging from 1 to 5 where NR1 is an raw noisy capture and NR5 is an image averaged over 16 captures within each FOV. All five noise regimes are included in model evaluation for both Confocal Mice and Confocal Zebrafish datasets.

While the averaged realizations are potentially useful, it is not realistic to get multiple captures of each FOV and essentially impossible to obtain enough captures to emulate an averaged ground truth. Instead of incorporating any averaged images into training, we only utilize the lesser noise images to evaluate our pipeline.

Our evaluation mirrors that of Probabilistic Noise2Void (PN2V) (Krull, Vicar, and Jug 2019); PN2V utilizes the same dataset which allows for efficient comparison. We train our model on captures from the first 18 FOV's, validate on the 20th FOV, and use the 19th FOV for testing evaluation. The metric

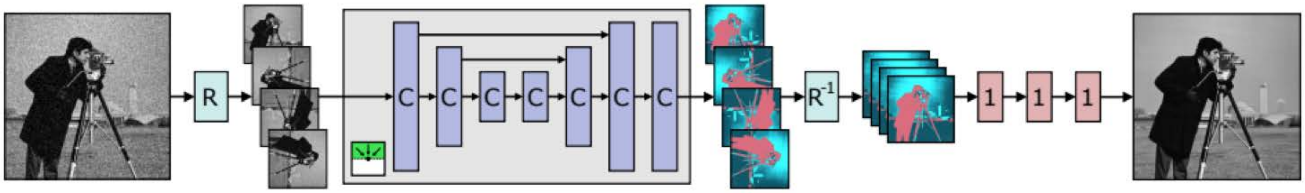


Figure 3: Rotational Blind-spot Network Architecture (Laine et al. 2019).

	NR1	NR2	NR3	NR4	NR5	Mean
Baseline	29.38	32.44	35.59	38.90	42.64	35.79
Poi2Void argmax	33.897	34.419	34.538	34.727	34.744	34.465
Poi2Void MAP	30.989	33.364	35.412	37.199	38.440	35.081
Poi2Void EVP	31.539	34.297	37.132	40.229	43.323	37.304
PN2V	38.24	39.72	41.34	43.02	45.11	41.49
N2V	37.56	38.78	39.94	41.01	41.9	39.85
N2N	38.19	39.77	41.28	42.83	44.56	41.33
U-Net	38.38	39.90	41.37	43.06	45.16	41.58

Table 1: Initial Confocal Mice PSNR Comparisons

used for testing is Peak Signal to Noise Ratio (PSNR), a widely used measure of signal power to corruption power on an image; similar to PN2V, we obtain the PSNR values for restored images of every noise regime.

Experimentation

We train two models separately using TensorFlow on 14,400 128x128 crops of Confocal Mice samples and 14,400 128x128 crops of Confocal Zebrafish samples within the FM dataset provided by Zhang et al. Image crops were necessary due to computational restrictions; each crop was chosen from a randomly selected region of a randomly selected image for each batch. We tuned network hyperparameters until converging on a 5.0e-6 learning rate, 100 training epochs, and a batch of one 128x128 crop again due to computational limitations. Before training, we normalized each training, validation, and testing image such that the 256 discrete pixel values were transformed into decimals between 0 and 1. The original blindspot architecture is then changed to outputs a final softmax layer to predict those pixel values as a prior for uncorrupted signal distributions. To avoid getting NaN loss during training, specifically caused by taking an improper log when calculating negative log likelihood, we clip the logits to be between 1.0e-5 and 80.

After training models on each dataset, we restored and concatenated cropped corrupted images in three ways: the maximum value of the network's softmax output at each pixel to act as a baseline (argmax), the Maximum a Posteriori evaluation (MAP), and the Expected Value of the posterior distribution (EVP). Our entire model is dubbed Poisson2Void (Poi2Void) for brevity in comparisons. Using PSNR as the sole metric, Tables 1 and 2 show the results of our model on each NR.

To improve our results, we add an entropy regularization term to our loss function. Entropy, the measure of uncertainty for categorical distributions, is used in this case to limit the model from choosing dominating values too early. By minimizing both negative log likelihood and entropy, we encourage our model to be as certain as possible about its output distribution. After experimentation, we determine that the optimal weights for the entropy calculation is 1.0e-2 on both datasets.

To further increase our PSNR values, instead of concatenating each restored crop into a 4x4 grid to yield the final image, we stride the crops over the image such that the window moves every 64 pixels instead of 128. This yields a 7x7 grid of restored crops that had to be carefully concatenated; each overlapping region was averaged to yield a more robust estimate for each testing image. Table 3 shows the non-trivial improvements in PSNR after strided restoration on our Confocal Mice dataset. Figures 4 and 5 show denoised images for all restoration methods.

Poi2Void does not yet surpass other contemporary semi-supervised and fully-supervised methods. Although the theory of our model is similar to that of other successful semi-supervised methods, such as PN2V (Krull, Vicar, and Jug 2019), we ran into several limitations during our experimentation that could have contributed to our modest results. Where other models trained models on entire images or larger crops, we were constrained by our GPU VRAM capacity. Additionally, we were forced to use a batch size of one small crop rather than 50-80 such as in recent work.

It is notable that although for raw imagery, argmax compares or surpasses our posterior manipulations, but as the noise regime increases, argmax falls behind the other methods. We use this as evidence that posterior

	NR1	NR2	NR3	NR4	NR5	Mean
Baseline	22.81	25.89	29.05	32.39	36.21	29.27
Poi2Void argmax	23.114	23.329	23.498	23.564	23.591	23.419
Poi2Void MAP	23.098	25.977	28.845	31.594	34.155	28.734
Poi2Void EVP	23.251	26.226	29.330	32.528	36.078	29.483
PN2V	32.45	33.96	35.48	37.07	39.08	35.61
N2V	32.10	33.34	34.43	35.39	36.21	34.30
N2N	32.93	34.37	35.71	37.06	38.65	35.74
U-Net	32.93	34.35	35.67	37.11	39.09	35.83

Table 2: Initial Confocal Zebrafish PSNR Comparisons

		NR1	NR2	NR3	NR4	NR5
	Baseline	22.81	25.89	29.05	32.39	36.21
Strided	Poi2Void argmax	34.485104	33.702407	33.82227	33.96317	33.98511
	Poi2Void MAP	31.08027	33.27537	35.12076	36.65470	37.72091
	Poi2Void EVP	34.72008	35.434919	37.27999	40.54513	43.752390
Grid	Poi2Void argmax	33.15988	33.52715	33.63908	33.76065	33.77202
	Poi2Void MAP	31.05832	33.23513	35.06682	36.57020	37.60538
	Poi2Void EVP	31.69174	34.43680	37.28053	40.43303	43.64095

Table 3: Strided Prediction PSNR vs Grid Prediction on Confocal Mice

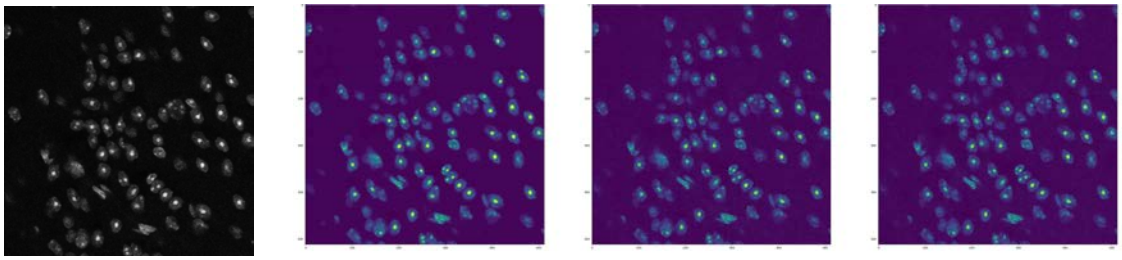


Figure 4: Poi2Void results on raw NR1 Confocal Mice data (left) using argmax, MAP, and EVP.

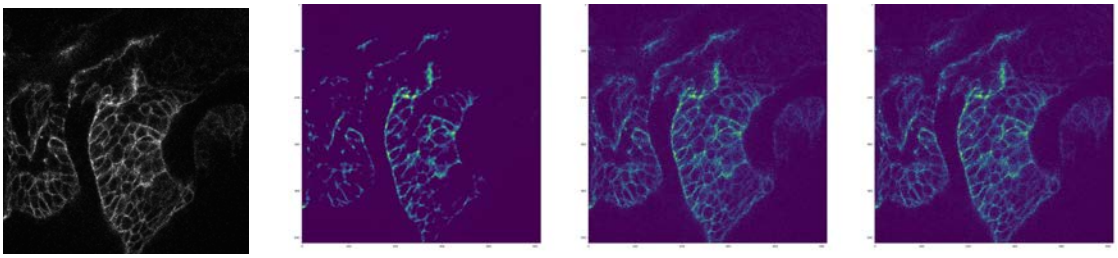


Figure 5: Poi2Void results on raw NR1 Confocal Fish data (left) using argmax, MAP, and EVP.

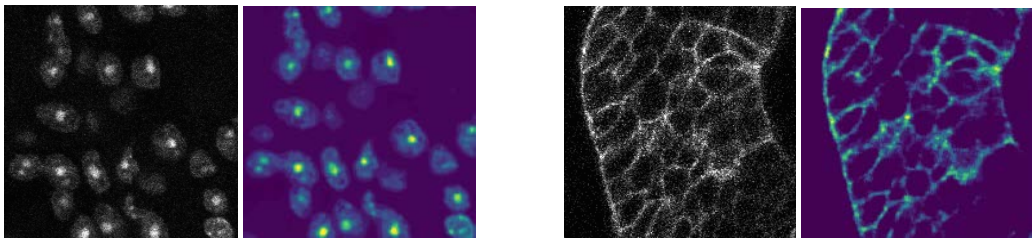


Figure 6: Zoomed in Comparison of Noisy to Restored on Confocal Mice (right) and Confocal Zebrafish (left)

manipulation is superior to only modelling clean pixel value probabilities; it is practically advantageous for our model to be able to generalize on varied levels of noise seen in the field.

Conclusion and Further Work

This work seeks to expand on existing literature by addressing Poisson-dominated noise within Fluorescence Microscopy imaging via a self-supervised denoising pipeline. Our approach's novelty lies in the self-supervision aspect of this pipeline, in contrast with contemporary model PN2V (Krull, Vicar, and Jug 2019), and its direct application to low-intensity biomedical imagery. Our implementation utilizes non-parametric prior probabilities of clean pixels, evaluated through a modified blind-spot network, with Maximum a Posteriori and Expected Value of the Posterior estimation to include possible ignored clean signals.

Although this model did not succeed in progressing past state-of-the-art self-supervised methods specifically for FMDD, we treat this experimentation as grounds for future work. We intend on replicating much of PN2V's network composition, via code provided by Krull et al.; PyTorch enables us to utilize a larger virtual batch size. If training becomes less computationally expensive, we can reasonably expect our loss to plateau at a lower value than at present. Additionally, we have made several hypotheses to modify the initial noise distribution used to calculate training loss and the posterior distribution. While our noise distribution follows the Poisson probability mass function, we clip the distribution to model λ from 1 to 257, and then normalize back into a usable probability distribution. It is possible that this process leaves out valuable information about the true noise. Lastly, we have recently acquired FM video data containing multiple noisy frames of a sample unharmed by typical phototoxicity. We expect that incorporating frames with very dependent signal values and independent noise realizations could greatly help our model.

Biologists have been using deep learning tools for imperative scientific work, and will continue to do so at increasing lengths as computational biology expands. We hope to continue developing and refining self-supervised denoising methods for Fluorescence Microscopy imaging.

Acknowledgement

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

References

Doersch, C., and Zisserman, A. 2017. Multi-task Self-Supervised Visual Learning. *arXiv:1708.07860 [cs]*. arXiv: 1708.07860.

Heinrich, M. P.; Stille, M.; and Buzug, T. M. 2018. Residual U-Net Convolutional Neural Network Architecture for Low-Dose CT Denoising. *Current Directions in Biomedical Engineering* 4(1):297–300.

Krull, A.; Buchholz, T.-O.; and Jug, F. 2018. Noise2void - Learning Denoising from Single Noisy Images. *arXiv:1811.10980 [cs]*. arXiv: 1811.10980.

Krull, A.; Vicar, T.; and Jug, F. 2019. Probabilistic Noise2void: Unsupervised Content-Aware Denoising. *arXiv:1906.00651 [cs, eess]*. arXiv: 1906.00651.

Laine, S.; Karras, T.; Lehtinen, J.; and Aila, T. 2019. High-Quality Self-Supervised Deep Image Denoising. *arXiv:1901.10277 [cs, stat]*. arXiv: 1901.10277.

Lehtinen, J.; Munkberg, J.; Hasselgren, J.; Laine, S.; Karras, T.; Aittala, M.; and Aila, T. 2018. Noise2noise: Learning Image Restoration without Clean Data. *arXiv:1803.04189 [cs, stat]*. arXiv: 1803.04189.

Remez, T.; Litany, O.; Giryes, R.; and Bronstein, A. M. 2017. Deep Convolutional Denoising of Low-Light Images. *arXiv:1701.01687 [cs]*. arXiv: 1701.01687.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv:1505.04597 [cs]*. arXiv: 1505.04597.

Zhang, Y.; Zhu, Y.; Nichols, E.; Wang, Q.; Zhang, S.; Smith, C.; and Howard, S. 2018. A Poisson-Gaussian Denoising Dataset with Real Fluorescence Microscopy Images. *arXiv:1812.10366 [cs, eess, stat]*. arXiv: 1812.10366.

Self-Supervised Learning for Single-Molecule Localization Microscopy Denoising

Clare Minnerath
Providence College
Providence, RI, USA
cminnera@friars.providence.edu

Jonathan Ventura
California Polytechnic State University
San Luis Obispo, CA, USA
jventu09@calpoly.edu

Guy Hagen
U. of Colorado Colorado Springs
Colorado Springs, CO, USA
ghagen@uccs.edu

Abstract

We evaluate the ability of self-supervised deep learning for Poisson denoising of Single-Molecule Localization Microscopy (SMLM) in addition to the impact denoising can have on the ability to locate molecules within the Single-Molecule Localization Microscopy images. SMLM images are predominantly corrupted with Poisson noise. There are currently existing methods for producing super-resolved SMLM images. However, there is a need for more accurate SMLM images in order for scientists to gain a better understanding of the functions of live cells at the nanoscale. By denoising SMLM images prior to the images undergoing the current state-of-the-art super-resolution techniques, we create a less corrupted version of SMLM images. As a result, the exact locations of the molecules in the images can be determined with more accuracy and precision. We denoise SMLM images utilizing only the original noisy images as training data with a Self-Supervised Deep Learning model. By modifying the previous Self-Supervised techniques that have been successful in denoising images with Gaussian noise, we remove Poisson noise from SMLM images.

Introduction

Since a majority of biological processes occur at the nanoscale, the ability to study cell and molecular behavior at this scale is critical for scientists. Specifically, the knowledge of nanoscale functions can help medical researchers design tools, treatments, and therapies that are more precise and personalized than conventional ones (Nano.gov).

Due to the effects of diffraction, classic optical microscopes are only able to resolve structures larger than 200nm (Allen et al. 2016). Single Molecule Localization Microscopy is a key technology for creating nanoscale images with optical microscopes. Using both super-resolution techniques and statistically locating individual molecule's blinks to sub-pixel resolution, SMLM is able to produce a new image which compiles the individually detected molecules together. The more exact SMLM can be in detecting and locating individual molecule blinks is a determining factor in how concise the location of the molecules can be determined. So, there is still room for improvement in locating individual molecules with more accuracy than is currently possible from noisy SMLM images due to the fact that the noise within an image makes pinpointing each molecule's true signal difficult.

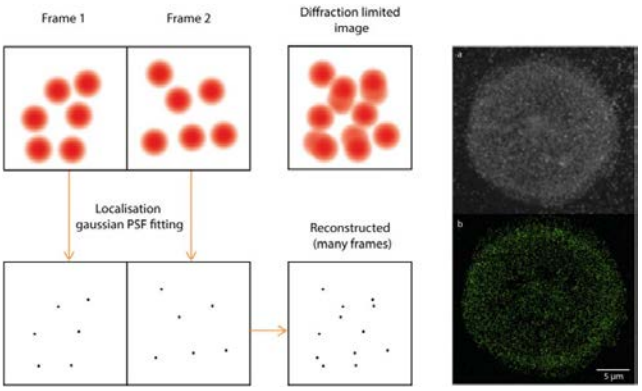


Figure 1: Process of SMLM localization and reconstruction to surpass the defraction barrier (Shannon et al. 2015).

SMLM images are taken as light is absorbed by the molecules, and, as a result, the molecules randomly emit light of a larger wavelength in small blinks which are captured by the camera. Because the blinks only emit a small number of photons, the resulting images are very noisy. Due to the dynamic blinking and low light levels in SMLM images, only a single noisy instance of a signal is available for a given instance of an image. Without a clean version of an image's signal, the use of a traditional deep learning approach that trains by mapping noisy images to the corresponding clean ones is not possible.

A current state-of-the-art super-resolution technique which utilizes deep networks is DeepSTORM, a CNN that trains on simulated and experimental data (Nehme et al. 2018). However, this method, although it has strong super-resolution performance, does not use any localization techniques which are vital in SMLM. Therefore, this method is not a practical solution for locating molecules on the nanoscale.

Traditionally, the best performance in deep learning approaches for image restoration and the denoising of corrupted images has been done using a supervised deep network that utilizes pairs of corresponding corrupt and clean images. In particular, there are content-aware image restora-

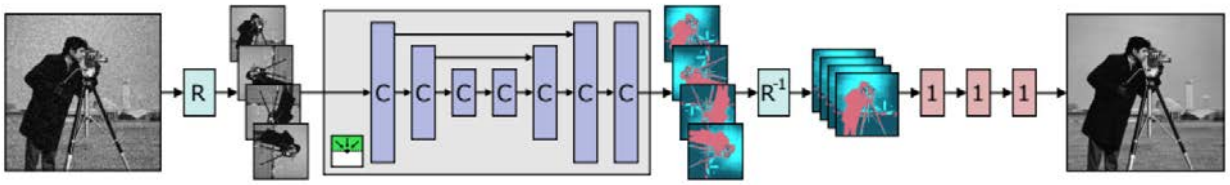


Figure 2: Proposed blind-spot network architecture for denoising SMLM images (Laine et al. 2019)

tion (CARE) networks that have proven useful for denoising fluorescence microscopy data given noisy and clean images (Weigert et al. 2018). Techniques such as these are rendered useless when considering SMLM data which has no way of obtaining clean images.

Related Work

New findings have shown that despite the limitations of solely individual noisy SMLM images, denoising to the performance level of supervised deep networks could be possible. The work of Lehtinen et al. (2018) shows that it is no longer necessary to provide a clean instance of an image. In NOISE2NOISE (N2N), they show by training using pairs of two corresponding noisy images that share the same signal, it is possible to achieve equal quality, if not higher quality (when training with finite data), performance for denoising. This method still requires at least 2 noisy realizations of an image's signal.

Building off the ideas from N2N, NOISE2VOID (N2V) shows that one can build a model for denoising using only individual noisy images for training data (Self-Supervised) (Krull, Buchholz, and Jug 2018). N2V introduces the use of a blind-spot network which masks a pixel's data from the network's receptive field and allows the data surrounding the said blind-spot to effectively predict the the pixel's signal. The N2V model is limited because it assumes every pixel's signal is dependent of the signals surrounding it.

To address the limitation of N2V, Laine et al. (2019) designed a different blind-spot architecture that uses 4 different receptive fields to allow every pixel in the image to contribute to the loss function while maintaining the blind-spot. This differs from N2V which selected a smaller receptive field from within the image. In order to predict the blind-spot pixel's signal, they apply a maximum a posteriori estimation (MAP) denoising procedure during testing that allows for the prediction of the clean signal to take into account the masked pixel. This method can perform on par with the supervised traditional deep denoising models under the assumption of Gaussian noise in the data.

Laine et al. does include a model for Poisson noise. But, in a manner impractical to SMLM images, their model uses a Gaussian approximation to represent the Poisson noise distribution. This method is impractical because the signal from SMLM is low, which causes a Gaussian approximation does not accurately represent the Poisson noise present in SMLM images.

More continued work since N2V also attempts to predict the blind-spot pixel without the assumption of a Gaussian noise model (Krull, Vicar, and Jub 2019). The PROBABILISTICNOISE2VOID (PN2V) model trains a probability distribution for 800 possible output values for a given pixel. PN2V is able to out perform N2V using the same U-Net architecture, but does not perform to the level of traditional supervised techniques. PN2V is working with discrete values for its pixel distribution which makes it a good fit to combat Poisson noise efficiently. But, since SMLM images are 16-bit, each pixel has 2^{16} possible values. Since SMLM images have such a large possible range in pixel values and the PN2V method shows success using a distribution of 800 possible output values, PN2V is not the optimally suited for denoising SMLM.

Research Questions

1. How well can self-supervised learning denoise SMLM images?

   Hypothesis: By modifying the work which successfully denoises images with Gaussian noise, it will be possible to denoise SMLM images to the levels of other self-supervised denoising models.

2. Does denoising SMLM images prior to localization techniques and super-resolution improve the ability to locate molecule positions compared to just using current localization and super-resolution techniques on noisy images?

   Hypothesis: Given SMLM currently depends on the noisy signals from the blinking molecules, by denoising the images prior to detecting and locating the molecules, our localizing abilities will be more precise. This is because the signal from the blinking molecules will be more representative of the molecules exact position in an image that is not corrupted with noise.

Method

By utilizing the blind-spot architecture designed by Laine et al. (Figure 2), which rotationally obscures the middle pixel, while also implementing an adjusted method for training the model and predicting the blind-spot signal's output, we are able to remove Poisson noise from SMLM images. In the process described by Laine et al. (2019), Poisson noise is approximated by considering a Gaussian distribution with $\sigma^2 = \lambda$. This estimate will only provide a strong approximation when the standard deviation within the Poisson noise is small and the image's signal is large.

To better train our network and more accurately estimate a masked pixel's signal while considering the Poisson noise present in SMLM, we created a process that can model the Poisson noise while producing non-discrete clean signal values. Using the Poisson probability mass function (Equation 1), it is possible to represent a distribution for the Poisson noise present in SMLM images.

$$P(Z = z) = \frac{\lambda^z \exp(-\lambda)}{z!} \tag{1}$$

We minimized the likelihood of the noisy pixel values over all possible clean values (Equation 2) to train our self-supervised denoising model.

$$P(Z_i = z_i) = \int_0^\infty P(Z_i = z_i | X_i = x) P(X_i = x) dx \tag{2}$$

Since we only have the noisy image signals, Z_i, our loss function must be representative of these known values. To do this we introduced the conjugate prior for the Poisson Distribution, the Gamma distribution. By representing the distribution of possible clean image signals, X_i, by the Gamma distribution, it is possible to marginalize out the clean realization of our image which is unknown in SMLM (Equation 3).

$$\begin{aligned} P(Z_i = z_i) &= \int_0^\infty \frac{\beta^\alpha}{\Gamma(\alpha)} \cdot \frac{x^{z_i+\alpha-1} e^{(-x-\beta x)}}{z_i!} \\ &= \frac{\beta^\alpha}{(1+\beta)^{\alpha+z_i}} \cdot \frac{\Gamma(\alpha+z_i)}{\Gamma(\alpha)} \cdot \frac{1}{z_i!} \end{aligned} \tag{3}$$

Using this marginalized equation, the negative log likelihood loss function can be calculated without using an approximation (Equation 4).

$$\begin{aligned} \mathcal{L}_i &= -\log\left(\frac{\beta^\alpha}{(1+\beta)^{\alpha+z_i}} \cdot \frac{\Gamma(\alpha+z_i)}{\Gamma(\alpha)} \cdot \frac{1}{z_i!}\right) \\ &= -\alpha\log(\beta) + (\alpha+z_i)\log(1+\beta) \\ &\quad - \log(\Gamma(\alpha+z_i)) + \log(\Gamma(\alpha)) + \log(\Gamma(z_i+1)) \end{aligned} \tag{4}$$

In order for the obscured pixel value to contribute to the clean pixel estimation, the best results were obtained using the Gamma posterior mean (Equation 5). This preformed better than the a the Gamma MAP estimation which tended to weigh the noisy pixel value too heavily.

$$\text{Gamma Posterior Mean: } x_i = \frac{\alpha + z_i}{\beta + 1} \tag{5}$$

For each YFP dataset we trained a separate model. Because of differences in noise levels and molecule shapes, the best results for denoising were produced by training and validating on 1000 images cropped to 512x512 pixels from a single dataset. Best results were obtained when the model is trained on images with a average to slightly higher signal. So images with little to no signal comparatively to the rest of the dataset were not included in the training batch. After training the networks using single 900 image batches with 100 validation images, the networks could then successfully denoise the entirety of the respective datasets. Training the model for each individual dataset is feasible because it takes between 12-18 epochs to train at approximately 6 minutes per epoch.

The visibly denoised images are then processed using the ImageJ Plugin, ThunderSTORM (Ovesn et al. 2014; Schneider, Rasband, and Eliceiri 2012). We then compared the results from the processed denoised images to the processed noisy images. We used this comparison as well as the results from testing on simulated datasets to make sure our model is removing the noise while retaining the true signal from SMLM images.

Evaluation

Data

To train and test our model we used two single molecule microscopy datasets of yellow fluorescent protein (YFP)-tagged growth factor receptors taken from human Epithelial carcinoma A431 cells expressing mCitrine-ErbB3 (Luke et al. 2018). Since this dataset uses YFP-tagged receptors rather than traditional dye, the molecules emit less photons than most SMLM images. As a result, the YFP datasets have low signal to noise ratio making them good candidates for denoising.

We also evaluated our model using simulated datasets. Simulated data will be useful for testing the accuracy of the localizing from our denoised data. Testing this allows us to gain some verification that the model does not alter the true signal or add signal from non-existing molecules because ground truth molecule locations are known for this data set. Using the ThunderSTORM Performance Testing tool we created 7 simulated data sets. One of the created data sets used the default simulation setting and was tested on all models. For the other 6 datasets, we created two groups each with 3 simulated datasets. Each group of data imitates one of two real YFP data sets at 3 different levels of noise resulting 6 more generated datasets.

Metrics

To evaluate how well our model is able to denoise the SMLM images we looked at the signal to background ratio (SBR). In order to find the SBR, there needed to be a way of distinguishing the background from the foreground. By implementing an automatic iterative threshold selection commonly used for gray-scale image thresholding, it was possible to determine a signal threshold to separate the signal from the background (Svoboda, Kybic, and Hlavac 2007). We hoped this metric would give a marker of how well our model is denoising given we have no clean image comparison to compare our results to. However, even when the results qualitatively appeared to improve and the localization uncertainty lowered, the SBR of a resulting image was not

necessarily greater than its noisy pair. Using the signal to background ratio is also not an ideal way to generally assess denoising because our model could successfully denoise the black background without having the same effect across the entirety of the image. For these reasons, we solely look at the qualitative results from the denoising model prior to utilizing our super-resolution and localization techniques. Then, if there is improvement in the ability to locate molecules in comparison to the localization ability on the images without denoising, it can be assumed that the denoising our model achieved occurred across the entirety of the image.

Once we have qualitatively determined our model's ability to denoise the SMLM images, we then evaluated the effect the denoising model had on the ability to locate individual molecules within the images. For this we used ThunderSTORM. ThunderSTORM is a SMLM image processing, analyzing, and visualizing tool (Ovesn et al. 2014). ThunderSTORM can be implemented as a plugin for the image processing application ImageJ (Schneider, Rasband, and Eliceiri 2012). It is particularly helpful in assessing a threshold for detection of molecules in raw and filtered images. So, in order to evaluate a potential improvement in the denoised images threshold for detection, we will first run the raw/noisy YFP data through ThunderSTORM and record the results. Then, we will denoise the YFP data with our Self-supervised denoising model. Finally, we will run the denoised data through ThunderSTORM once more and compare the threshold for detection results between the raw and denoised data. Along with this quantitative data, ThunderSTORM has a visualization tool that creates a new image of the cell by compiling the detected molecules locations together. So, a final visual result can also be compared between the noisy and denoised datasets.

Results

After training and testing using the self-supervised denoising model on yellow fluorescent protein (YFP)-tagged data, there have been improvements in both the reduction of noise in the image (evaluated qualitatively) and the ability to locate individual molecules within the SMLM images.

Despite the Signal to Background Ratio metric not being a quality measure of the noise in the images, it is not difficult to qualitatively evaluate the denonising process. In Figures 3 and 4, images (a) and (b) from YFP-dataset 3 as well as images (e) and (f) from YFP-dataset 4 demonstrate the models ability to pick out the signal amongst noise. Images (b) and (f) do not appear to have as much signal as their noisy counterparts images (a) and (e). But, the difference is due to the noise being amplified in areas of signal in images (a) and (e), not a loss of molecule signal. The ability to get rid of the excess noise around the signal allows for individual molecules to be pinpointed. In an area where multiple molecules are close together, their respective noise can blur together making it unfeasible to locate each molecule individually with

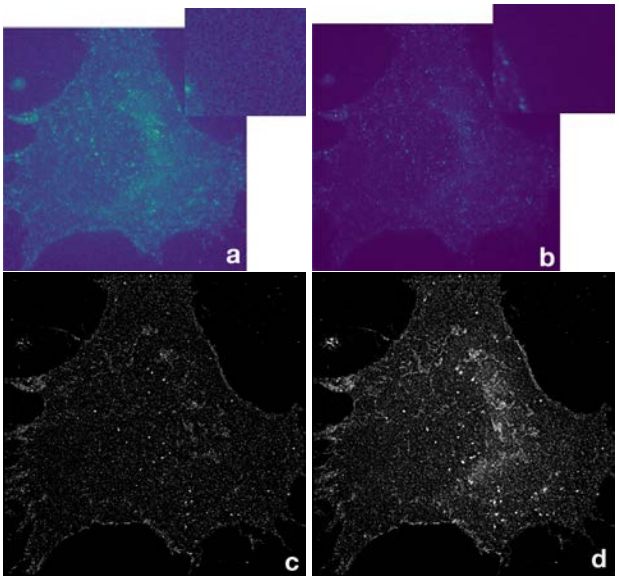


Figure 3: YFP-Dataset 3 denoising visual results: (a) Single noisy YFP image with an additional zoom of the top right corner. (b) Single corresponding denoised YFP image with an additional zoom of the top right corner. (c) Image compiled from molecule localization from all YFP-dataset 3 noisy images. (d) Image compiled from molecule localization from all denoised YFP-dataset 3 images.

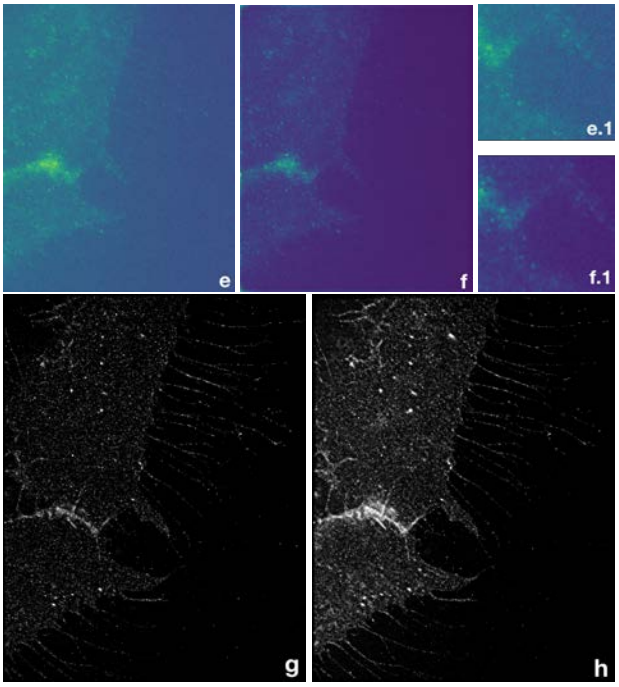


Figure 4: YFP-Dataset 4 denoising visual results: (e) Single noisy YFP image. (e.1) An additional zoom from image (e). (f) Single corresponding denoised YFP image. (f.1) An additional zoom from image (f). (g) Image compiled from molecule localization from all YFP-dataset 4 noisy images. (d) Image compiled from molecule localization from all denoised YFP-dataset 4 images.

Data	Molecules Located	Mean Sigma [nm]	Standard Dev. Sigma [nm]	Mean Uncertainty [nm]	Standard Dev. Uncertainty [nm]
Noisy YFP-Dataset 3	364,509	104.8981	62.35194	17.43365	5.770702
Denoised YFP-Dataset 3	**882,188**	111.466	60.64917	**13.46961**	6.124521
Noisy YFP-Dataset 4	153,061	109.2096288	88.81489875	17.08962255	6.198097691
Denoised YFP-Dataset 4	**974,163**	138.215115	101.79013	**16.9050282**	5.87430829

Table 1: Molecule localization data obtained using ThunderSTORM with default settings + EM Gain = 150.0 for YFP-Dataset 3 and EM Gain 100.0 for YFP-Dataset 4. Post-processing with ThunderSTORM: Duplicate removal and drift correction.

Data	True Positives	False Positives	False Negatives	Jaccard Index	Precision	Recall	F1
Noisy Simulated	150,630	3,790	16,874	0.879	0.975	0.899	0.936
YFP-3 Model	156,795	3,899	10,709	**0.915**	0.976	0.936	0.955
YFP-4 Model	156,136	4,178	11,341	**0.91**	0.974	0.932	0.953

Table 2: Performance evaluation results from the ThunderSTORM simulated dataset using the default settings. The data from each row is the result of a comparison between the dataset's localizations and the ground truth molecule locations.

any certainty. To locate each molecule in an SMLM image, ThunderSTORM fits a Gaussian distribution to the signals from the image. Using the data from the Gaussians at each point, it can be determined whether a molecule is present, where it is present, and a location uncertainty value. We know more molecules can be located in the denoised images because of the compiled images produced as well as the localization data found in Table 1.

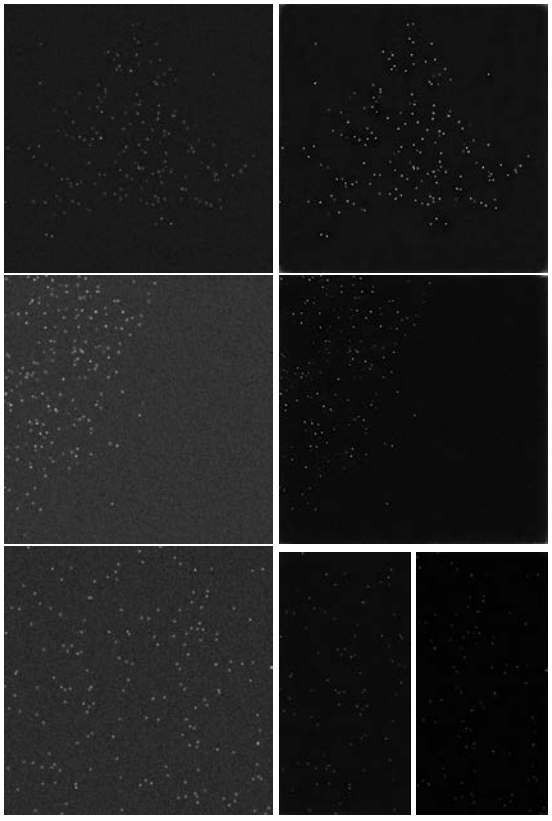


Figure 5: Single noisy ThunderSTORM simulated images on the left with the corresponding denoised images on the right. From top to bottom: generated data simulating YFP-dataset 3 (Mean Noise Level = 150.0 photons), generated data simulating YFP-dataset 4 (Mean Noise Level = 150.0 photons), generated data using default settings.

Table 1 shows the results of the analysis of the original noisy image sets in comparison with analysis of the denoised image sets. Sigma is the standard deviation of the gaussian fit to a molecules signal and the uncertainty measure is calculated using a combination of this value, the intensity of the signal, and other parameters from the gaussian distribution. Using ThunderSTORM it was possible to locate more than double the amount of molecules with the denoised data compared to the original corrupted data. Both models were able to make these large jumps in localizaion of molecules while still maintaining lower mean uncertainties.

Applying our denoising networks to simulated data also produced improvements in visual denoising and molecule locating. Using the ground truth molecule locations it is possible to determine how many molecules are being accurately located as well has the number of false positive and false negative molecule localizations. For our experiments, a localization of a molecule is considered correct if it is within 50nm of the ground truth location (default setting). The results from our simulated data give reassurance in our denoised molecule localization numbers from the real data.

Utilizing the ThunderSTORM simulated data default noise and signal settings, we created a 1,000 image dataset. Table 2 and the bottom images in Figure 5 display the results of the models which were trained on real data being used to denoise the ThunderSTORM default simulated dataset. The left of the bottom right two images is the output of the model trained on YFP-dataset 3 and the right of the two is the output of the model trained on YFP-dataset 4. Both models produced images which were able to detect more true molecule locations than the original noisy image set. However, on this specific dataset both model's output images had more false positive localizations than from the noisy data. We speculate that a model trained and tested on data with different levels

Noise Level	Data	True Positives	False Positives	False Negatives	Jaccard Index	Precision	Recall	F1
50.0	Noisy	167,163	7,075	14,690	0.885	0.959	0.919	0.939
	Denoised	171,418	2,847	10,435	**0.928**	0.984	0.943	0.963
100.0	Noisy	166,354	7,768	15,253	0.878	0.955	0.916	0.935
	Denoised	169,642	3,979	11,965	**0.914**	0.977	0.934	0.955
150.0	Noisy	165,023	8,466	16,258	0.870	0.951	0.910	0.930
	Denoised	167,947	5,890	13,334	**0.897**	0.966	0.926	0.946

Noise Level	Data	True Positives	False Positives	False Negatives	Jaccard Index	Precision	Recall	F1
50.0	Noisy	167,814	11,855	24,052	0.824	0.934	0.875	0.903
	Denoised	169,974	6,407	21,892	**0.857**	0.964	0.886	0.923
100.0	Noisy	165,988	12,903	25,867	0.811	0.928	0.865	0.895
	Denoised	167,174	8,514	24,681	**0.834**	0.952	0.871	0.910
150.0	Noisy	163,058	13,732	28,328	0.795	0.922	0.852	0.886
	Denoised	163,228	12,032	28,158	**0.802**	0.931	0.853	0.890

Table 3: Performance evaluation results from ThunderSTORM generated data imitating YFP-dataset 3 (above) and YFP-dataset 4 (below) at three different mean background noise levels (photons). The data from each row is the result of a comparison between the dataset's localizations and the ground truth molecule locations.

of signal and noise leads to lower quality denoising performance and therefore lower localization abilities. Nonetheless, the overall result is that all the similarity indexes are improved by the model's denoised images except the precision of localization in the images from the YFP-4 model.

To better represent the real data that our models were trained on, we generated simulated data with a comparable signal intensity range as well as comparable full width at half maximum range for the molecule signal spread to the real YFP datasets. Using a mask (thresholded image) from the original YFP-dataset, the ThunderSTORM data generator spreads the signal intensities according to the mask's values. We then replicated the simulation at 3 different noise levels for both of the datasets since the level of noise throughout the real YFP images frames is not uniform. Each dataset consisted of 1,000 images and was tested on the network trained on the data it is simulating. After processing the noisy data and model's output, the produced molecule localizations are compared with the ground truth locations. As can be seen in Table 3, at each level of noise both of the model's are able to pinpoint more molecules while maintain less false positives and false negatives than the noisy data. As a result, the similarity indexes are higher at every point for the denoised data in comparison with the original simulated noisy data. An example of the qualitative denoising results from the model on the simulated data can be seen in Figure 5. So, our models, when trained at certain signal intensities/noise levels, are able to produce denoised images which accurately locate higher numbers of molecules in a given dataset.

Conclusion

We have developed and evaluated a self-supervised deep learning model for denoising Single Molecule Localization Microscopy images which are corrupted with Poisson noise. To do this, we built off recent research in self-supervised denoising models and fit the techniques to denoise images taken in low light conditions (images containing large amounts of Poisson noise). As a result of the denoising, there has been no correlation with the Signal to Background metric. However, utilizing the ImajeJ plugin, ThunderSTORM, we have been able to see an improvement in detecting and locating individual molecules within SMLM images. Reproducing positive results on real data as well simulated data has shown that our models are able denoise SMLM images without distorting the true image signal. By locating more molecules and continuing farther past diffraction barrier, we hope to create better SMLM images which will allow biologists to advance the studies of cell and molecular behavior at the nanoscale.

Acknowledgement

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

References

Allen, J.; Silfies, J.; Schwartz, S.; and Davidson, M. 2016. Single-molecule super-resolution imaging. *Nikons MicroscopyU*.

Krull, A.; Buchholz, T.-O.; and Jug, F. 2018. Noise2void-learning denoising from single noisy images. *arXiv preprint arXiv:1811.10980*.

Krull, A.; Vicar, T.; and Jub, F. 2019. Probabilistic noise2void: Unsupervised content-aware denoising. *arXiv preprint arXiv:1906.00651*.

Laine, S.; Karras, T.; Lehtinen, J.; and Aila, T. 2019. High-quality self-supervised deep image denoising. *arXiv preprint arXiv:1901.10277*.

Lehtinen, J.; Munkberg, J.; Hasselgren, J.; Laine, S.; Karras, T.; Aittala, M.; and Aila, T. 2018. Noise2noise: Learning image restoration without clean data. *arXiv preprint arXiv:1803.04189*.

Luke, T.; Pospil, J.; Fliegel, K.; Lasser, T.; and Hagen, G. 2018. Supporting data for quantitative super-resolution single molecule microscopy dataset of yfp-tagged growth factor receptors. *GigaScience Database*.

What's so special about the nanoscale? *Official website of the United States National Nanotechnology Initiative*.

Nehme, E.; Weiss, L. E.; Michaeli, T.; and Shechtman, Y. 2018. Deep-storm: super-resolution single-molecule microscopy by deep learning. *Optica* 5(4):458–464.

Ovesn, M.; Kek, P.; Borkovec, J.; vindrych, Z.; and Hagen, G. M. 2014. Thunderstorm: a comprehensive imagej plug-in for palm and storm data analysis and super-resolution imaging. *Bioinformatics* 30(16):23892390.

Schneider, C. A.; Rasband, W. S.; and Eliceiri, K. W. 2012. Nih image to imagej: 25 years of image analysis. *Nature Methods* 9:671675.

Shannon, M.; Burn, G.; Cope, A.; Cornish, G.; and M Owen, D. 2015. Protein clustering and spatial organization in t-cells. *Biochemical Society transactions* 43:315–21.

Svoboda, T.; Kybic, J.; and Hlavac, V. 2007. *Image Processing, Analysis & and Machine Vision - A MATLAB Companion*. Thomson Learning, 1st edition.

Weigert, M.; Schmidt, U.; Boothe, T.; Mller, A.; Dibrov, A.; Jain, A.; Wilhelm, B.; Schmidt, D.; Broaddus, C.; Culley, S.; Rocha-Martins, M.; Segovia-Miranda, F.; Norden, C.; Henriques, R.; Zerial, M.; Solimena, M.; Rink, J.; Tomancak, P.; Royer, L.; Jug, F.; and Myers, E. W. 2018. Content-aware image restoration: pushing the limits of fluorescence microscopy. *Nature Methods*.

I-MOVE: Independent Moving Objects for Velocity Estimation

Jonathan Schwan **Akshay Raj Dhamija** **Terrance E. Boult**
University of Colorado, Colorado Springs
{jschwan2 | adhamija | tboult} @ vast.uccs.edu

Abstract

This undergraduate research was conducted because although there are multiple static RGB-D, stereo, and ego-motion datasets, none specifically address the problem of motion parameters and object velocity estimation. We introduce I-MOVE, the first publicly available RGB-D/stereo dataset for estimating velocities of independently moving objects. The dataset features various outdoor and indoor scenes of single and multiple moving objects. Compared to other datasets, a unique property of I-MOVE is that the 3D position and 3D velocity for each object is supplied for a variety of different settings / environments and objects / motions. The dataset includes training and test sequences captured from four different RGB-D camera views and three 4K-stereo setups. The data are also time synchronized with dual Doppler radars to provide velocity ground truth. Multiple scenes are designed for high-quality ground truth computations with increasing levels of complexity. The I-MOVE dataset also includes complex scenes from moving pedestrians to multiple flying drones captured with the seven stereo cameras. **We look forward to the constructive feedback on the idea of the dataset and its collection process.**

Introduction

"What we see depends mainly on what we look for." John Lubbock, *The Beauties of Nature and the Wonders of the World We Live in*, 1892.

The above quote has stood the test of time in the field of computer vision. In the realm of still imagery, several problems have and continue to be addressed, such as image classification and object detection. Similarly, with the rise in popularity of videos, problems such as tracking (Atev et al. 2005; Sadeghian, Alahi, and Savarese 2017; Jiang et al. 2018; Kim, Li, and Rehg 2018), localization and mapping (Zhang et al. 2018; Brosh et al. 2019), action recognition, as well as sentiment analysis have been identified (Chang et al. 2019; Piergiovanni and Ryoo 2019). In this undergraduate work, we present the relatively unexplored task of motion parameter estimation. Even though motion parameters are useful for a large variety of applications, estimating them from videos has not been studied extensively. Motion parameters are a necessary component in numerous applications such as robotic navigation (Chuang et al. 2018; noa) and collision detection (Atev et al. 2005; Gandhi and Trivedi 2006; Heyman 2019). Because having related data to this problem has become so necessary, many synthetic environments have been created (Fei et al. 2019; Zamora et al. 2016; Fan et al. 2018). These synthetic environments have greatly helped people approach the problem, but in the unconstrained real world environments these tasks are much more complicated than the research environments. In problems such as collision detection it is not only necessary to take into account your directional velocity but that of other objects as well. In order to accomplish this it is required that you have the three dimensional directional velocity of each of the objects. Similarly, in robotics if one wishes to enable a robot to interact with an independently moving object (ex flying ball or frisbee or independent drone), the motion parameters of these objects need to be accurately estimated in order to understand the trajectory. Another important application area for motion parameter estimation is sports. In numerous sports performance analysis of athletes, relies on velocity and acceleration information. Most obviously, sports where speed is the main component (running, biking, swimming, etc.), but also for sports such as weightlifting where the athletes are looking for their lift force and acceleration in order to calculate the best feasible posture or lift technique. Similarly, motion parameters may also be useful for training purposes in various sports such as skiing, snowboarding or skateboarding.

While most of the constrained application areas could either document motion parameter information using specialized sensors, such as IMUs (Inertial Measurement Unit) or from egocentric videos, these are not viable for unconstrained scenarios because both of these approaches need to have access to the object in motion. With a task such as estimating the instantaneous velocity of a flying ball neither IMUs nor egocentric videos may be used (Chatzitofis, Zarpalas, and Daras 2018; Chatzitofis et al. 2013; Einfalt, Zecha, and Lienhart 2018; Lee and Kitani 2016). Since views from cameras can be easily accessible for such problems they become the logical choice to create a more useful and robust method for motion parameter estimation.

Since, to the best of our knowledge none of the current datasets provide velocity estimation information or other vital motion parameter ground truth for such complicated tasks, we introduce a new dataset (I-MOVE). Our dataset focuses on aiding motion parameter estimation for outdoor and

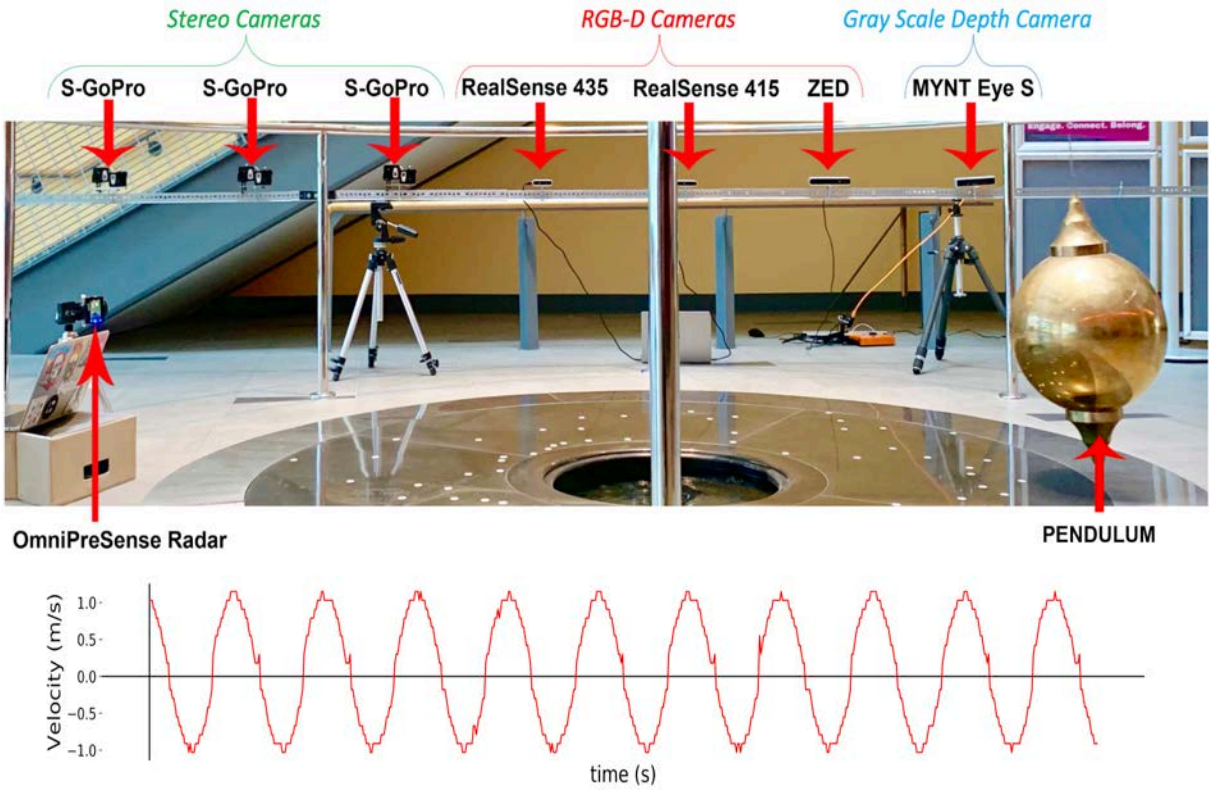


Figure 1: SETUP FOR DATASET COLLECTION *The above picture describes the setup used for the data collection process for a moving pendulum, which is just one of the various moving objects in our dataset. Our dataset aims to allow vision-based estimation the velocity of moving objects from any of the three types of sensors. The first set are three pairs of high-resolution stereo cameras, 4K hardware synchronized Go-pros in custom mounts, with 2.92mm, 4.3 and 5.2mm lenses respectively. The second set are Intel RealSense RGB-D cameras (415 and 435), with active illumination. The third sensor is a passive stereo system (ZED). The final video is a grayscale active illumination depth sensor (MYNT Eye S). The same setup is used for data collection on all our moving objects. In order to acquire the ground truth velocity we use one or two Doppler radars with varying positions relative to the moving object(s). The doppler radars provide the instantaneous velocity of the pendulum as depicted in the bottom plot of the figure. Since the pendulum is moving towards and away from the radar it provides a sinusoidal instantaneous velocity. For further details on the setup please refer Section .*

indoor objects of various kinds. In order to enable research in incremental steps for such a hard problem our dataset includes three types of vision sensors, providing RGB, Stereo and RGB-D data. The ground-truth velocities are obtained via Doppler radars. The data is also collected in sufficient amounts from varied sources of cameras to enable training of supervised deep learning methods. The objects also vary throughout the scenes, for example some scenes are based upon a person's movement while others are designed for easier predictable movements, e.g. a rolling ball or pendulum as can be seen in Figure 1. Unlike any of the existing datasets, we also provide ground truth measurements of 3D velocity parameters for each moving object in each scene using Doppler radars. Depending on the object's path / movement we will use either one or two doppler radars (for more unpredictable movements where triangulation is necessary to estimate the velocity). In order to validate the accuracy of the Doppler radars we use them in simple scenes, where the velocity information can be easily verified by using the laws of physics. Such scenes include rolling objects down inclined planes, motion of a pendulum, and falling/flying objects.

Related Datasets

Vision based velocity estimation has been studied for decades (Hinedi 1988; Nakazawa, Ishihara, and Inooka 2003; Mahapatra and Mehrotra 2000). In recent years, with application of computer vision algorithms to the domains of robotics (Xia et al. 2018; Hua et al. 2018) and autonomous driving (cars (Fangjun Jiang and Zhiqiang Gao 2000; Kampelmhler, Mller, and Feichtenhofer 2018) and drones (Chuang et al. 2018)), the number of works attempting to estimate motion parameters has grown dramatically (Coskun et al. 2017; Prez-Ortiz et al. 2003). As a result the need for these datasets has also grown greatly (Zhu et al. 2018; Sturm et al. 2012; Kim et al. 2018; Chen, Jafari, and Kehtarnavaz 2015; Sigal, Balan, and Black 2010). Many of these works differ in the primary purpose of the dataset and the objects for which motion parameters are estimated. In this section we will first recognize the datasets that either are aimed for motion parameter estimation or a related task. These datasets generally either use RGB data or data acquired from non vision systems. Then we discuss the most similar datasets within the RGB-D realm.

	HumanEva (Sigal, Balan, and Black 2010)	DIML (Kim et al. 2018)	Evaluation of RGB-D SLAM Systems (Sturm et al. 2012)	UTD-MHAD (Chen, Jafari, and Kehtar-navaz 2015)	Human Activity Recognition (Anguita et al. 2013)	I-MOVE (Ours)
Velocity Ground Truth						✓
Moving Object(s)	✓				✓	✓
Any Specific Motion Parameter Ground Truth	✓					✓
Images / Video Provided	✓	✓	✓	✓		✓
Indoor Images	✓	✓	✓	✓		✓
Outdoor Images						✓
Multiple Cameras Used	✓					✓
Multiple Perspectives of Same Object	✓					✓
RGB-D		✓	✓			✓
Passive STEREO						✓

Table 1: COMPARISON OF THE CURRENTLY AVAILABLE DATASETS *Above we summarize some of the currently available datasets. It should be observed that none of the available datasets provide a velocity ground truth which is a big contribution of our I-MOVE dataset. The closest dataset to ours is the HumanEva dataset which unlike ours only contains humans.*

RGB or Motion Parameter Only Datasets

The Human Activity Recognition dataset (Anguita et al. 2013) provides potentially useful data to address the motion parameter estimation problem. The dataset contains smartphone accelerometer information collected by numerous people performing various tasks such as sitting, walking, and going up stairs. However, this dataset contains no images / video and was created with the intention of creating a model that could predict activity solely from the accelerometer information. Since the aim of the problem presented in this paper is to estimate the motion parameters of an object from a video, this dataset cannot be utilized for its addressal.

Another human activity recognition based dataset is the UTD-MHAD (Chen, Jafari, and Kehtarnavaz 2015) which contains both IMU and video information. This dataset contains 27 actions performed by 8 subjects (4 females and 4 males). Each subject repeated each action 4 times. The actions, such as knock on door, sit to stand, and stand to sit, are fairly limiting in movement, and hence do not make them as desirable to estimate motion parameters.

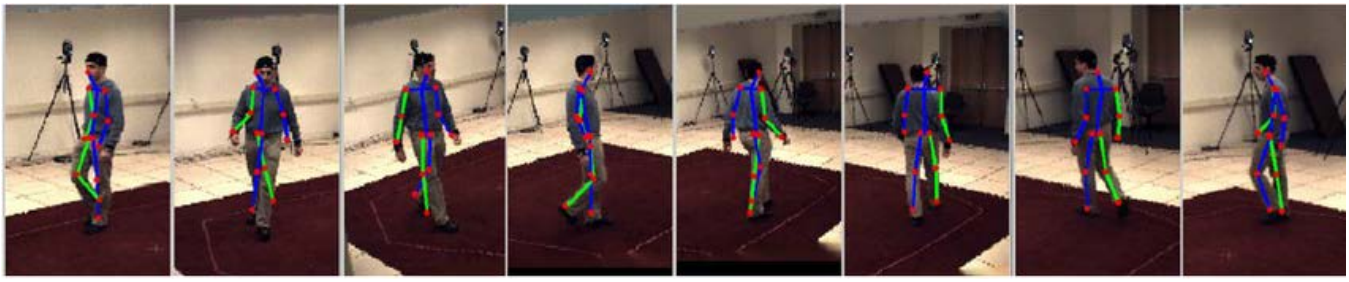
Another interesting dataset is the HumanEva dataset (Sigal, Balan, and Black 2010), which is a synchronized video and motion capture dataset. It consists of 4 subjects performing a set of six predefined actions three times (twice with video and motion capture, and once with motion capture alone). This dataset was intended to be used to improve existing three dimensional pose estimation, but it may also be used for motion parameter estimation.

RGB-D Datasets

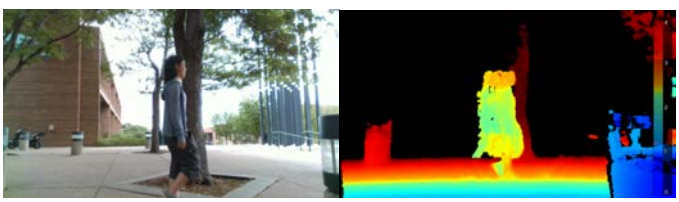
While there are a variety of RGB-D datasets, to the best of our knowledge there is no RGB-D dataset that contains velocity ground truth to accurately evaluate an algorithms performance. The most similar dataset is the one proposed in the paper, A Benchmark for the Evaluation of RGB-D SLAM Systems (Sturm et al. 2012). The dataset contains the color and depth images from a Microsoft Kinect sensor along with the ground-truth trajectory of the sensor. The ground-truth trajectory was obtained from a high-accuracy motion-capture system with eight high-speed tracking cameras plus the accelerometer data from the Kinect. However, since the kinect has limited performance in outdoor environments, the dataset was restricted to indoor use only. Moreover, the dataset only contained a single type of object and hence even if someone would attempt to create a system for motion parameter estimation on this dataset it may not generalize well on other objects. The DIML RGB-D Dataset (Kim et al. 2018) also contains data collected with a Kinect, however this database does include indoor and outdoor video in addition to object segmentation making it more plausible to conduct tests for motion parameter estimation purposes. But this dataset too does not contain any velocity ground truth. As well as since the dataset only contains single camera views any system created to estimate motion parameters on this dataset may not translate well to data from a different camera source.

The I-MOVE Dataset

Though our dataset collection process is ongoing, we have completed our initial round of dataset collection, which included deciding upon the sensors and the conditions for data collection as well as recording the initial physics based setups to mathematically verify the quality and plausibility of our velocity estimation. Through this paper we attempt to get a feedback / input from the computer vision community on any shortcoming or missing aspects in our initial setup.



(a) Samples from HumanEva Dataset



(b) Ours I-MOVE (Intel RealSense 435)

(c) Ours I-MOVE (ZED)

Figure 2: HUMANEVA DATASET VERSUS OUR I-MOVE DATASET *While the HumanEva dataset uses humans as the moving subjects whose motion is recorded with motion capture, the dataset is primarily aimed for key point tracking (Fig 2(a)). These keypoints may be used for motion parameter estimation but the limited motion of the subjects does not make it interesting enough for motion estimation task. Moreover, the dataset is restricted to indoors not providing enough lighting variations for a good relevance to the real world scenarios. To overcome the short coming of the HumanEva dataset we propose I-MOVE where the subjects are captured in an unconstrained environment with a variety of movements. As visible from the Fig 2(b) and 2(c). The variations in the type of camera also provide considerable variations to the same scenes, these variations may also prove useful for training a deep network based approach.*

Compared to the reviewed datasets, I-MOVE is unique in the following ways:

(a) To the best of our knowledge, this dataset is the first to focus on and provide object motion parameters.

(b) I-MOVE also contains a variety of objects in both indoor and outdoor scenes.

(c) Each scene is captured with a variety of cameras from different viewing angles. This variety of data provides the necessary means for developing more robustness approaches.

(d) We also provide the ground truth velocity for each moving object in the scene using doppler radars.

(e) The performance of the doppler radars is validated in controlled experiments where the results can be verified using basic laws of physics.

The velocity ground truth is also thoroughly proofed and tested to ensure accuracy with physics based examples and settings to allow completely mathematical based calculations to be done by hand and compared against.

Setup

The apparatus used for data collection was meticulously crafted to ensure the most reliable results in various locations / scenes. The seven cameras and two radars were mounted on a 14 gauge angle bar as can be seen in Figure 1, this allows us to adjust the various heights of the apparatus (cameras) as needed to give us more accuracy and reliability when collecting data on an uneven surface.

The cameras and radars were mounted identically at each location such that each camera's individual abilities can also be evaluated (range, quality, accuracy, etc.) in different settings. The cameras used were three GoPro Hero 3 stereo rigs (six GoPros in total because there are two GoPros per stereo setup), along with two Intel RealSense cameras (a 415 and 435), a ZED camera, and lastly a MYNT Eye S. The radars used were OmniPreSense doppler radars, which provide velocity of objects within their 78° wide beams.

Due to the wide variety in object size, scene layouts, and environments where data was collected, the apparatus was created to accommodate these differences. The cameras all have a different field of view and so the order and spacing of them was vital to collect the data as best possible. Two of the GoPro setups have modified lenses giving them a field of view of 54.1° and 64.7°, respectively. The standard / unmodified GoPro stereo set up has a horizontal field of view of 122.6°. The Intel RealSense 435 has a field of view of 85° horizontally, while the RealSense 415 version has 63°, and finally, the ZED is capable of 90° viewing horizontally. We also had additional depth information recorded with a MYNT Eye S (that has a 122° field of view horizontally) to allow for better camera to camera comparisons; however, the MYNT we used is not RGB / only monochromatic.

Given our purposeful placement based on the field of view of the cameras, see Fig. 3, we were able to obtain a two foot spacing between each camera allowing for fairly significant

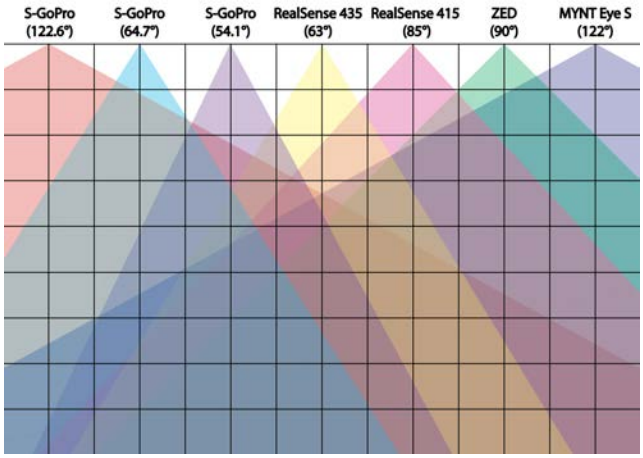


Figure 3: SKEMATIC FOR HORIZONTAL FIELD OF VIEW OF EACH CAMERA *Each box represents a 1 foot x 1 foot square and as can be seen in the figure the cameras are placed two feet apart horizontally. The figure also shows that object has to be slightly less than five feet away from the center camera (Intel RealSense 435) in order to be in the field of view of all the cameras.*

difference in camera perspective while still having all cameras capturing the object and it's most valuable movements. The set up was arranged from left to right (facing the lens) as follows: Stereo GoPro (standard), Stereo GoPro (modified), Stereo GoPro (modified), Intel RealSense 415, Intel RealSense 435, ZED, MYNT Eye S (Monochromatic). This can be seen in Fig 1. This rig allows us to have a fairly portable and consistent system which is essential given the numerous locations / environments where the data was collected.

Calibration and Synchronization of Sensors

Intrinsic and extrinsic distortion parameters of the sensors were found using the checkered board approach commonly done with OpenCV (Romero-Ramirez, Muoz-Salinas, and Medina-Carnicer 2018; Datta, Kim, and Kanade 2009). Due to the variation in lens and distortion for every single camera used in our rig, it is necessary that the OpenCV calibration approach is done separately for each camera at each location or every time there is a significant change in lighting or background. For some of the cameras such as the ZED and Intel RealSense, calibration options were available within the SDKs, so these were used when possible. In addition to calibration, synchronization was also essential due to the fact that the same motion ground truth was used for different cameras and perspective. For this reason it is also crucial to ensure the radars providing the ground (that is not obtainable via physics based setups / equations) are time synced with the cameras so that the ground truth can be accurately applied to the appropriate frame from each stereo camera setup. In order to do this we had the same data collection device (an HP XPS 13) used to collect the information from both OmniPreSense radars also to collect the Inter RealSense 435 data. Both radars are set to return their timestamp information in addition to the speed data and these timestamps are synchronize with the Intel RealSense 435's data. This synchronization allows us to use a flash event, where we utilize a camera flash that lasts 4 milliseconds allowing the moment to be visually captured by all the stereo cameras and using the frame(s) with flash to appropriately sync the velocity information to each frame.

Data Collection

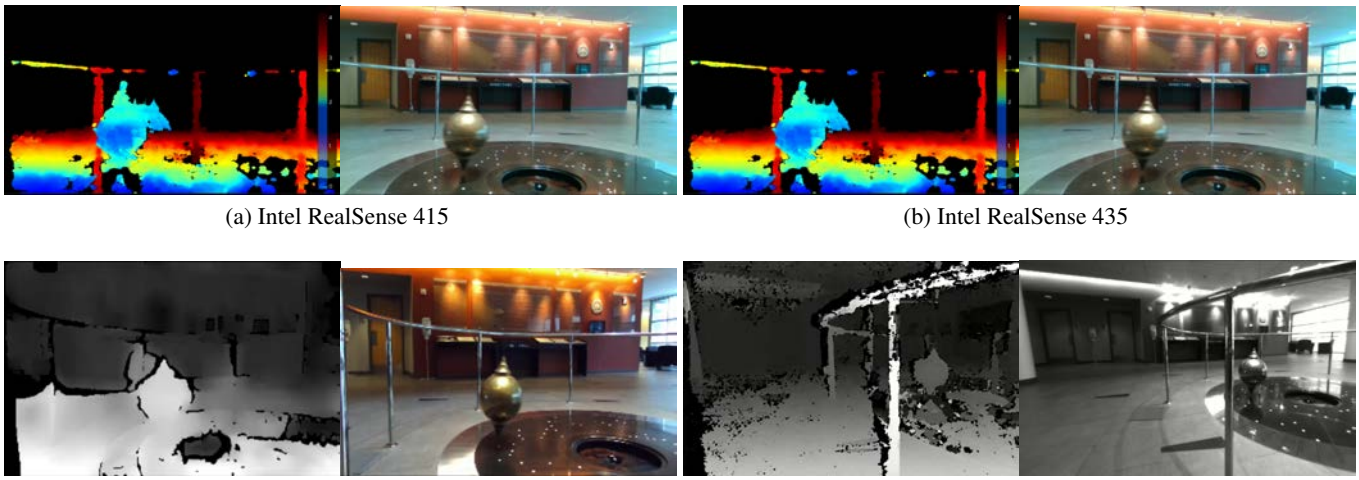
Our data collection process was intended to include a significant variety in objects, object motion, object velocity, scenes / environments and lighting, while still ensuring maximum accuracy in the ground truth for segmentation and velocity of the object. The dataset was also designed to allow the same ground truth to be applicable to multiple views. The six cameras used ensure a fairly significant variety in camera perspective / viewing location in addition to the difference in field of view. The variety in cameras also allows the dataset user to compare the performance between them if they wish.

Currently, the dataset features ten different objects: person, car, dog, skateboarder, skateboard, biker, ball, drone, pedestrian and RC Car. These objects differ widely not only in shape and size but also motion paths making them suitable to train and test upon. We have numerous scenes that involve the different objects in their respective environments with their frequent movements and motion. However, in an attempt to ensure that a model may accurately learn the motion parameters regardless of the object itself the I-MOVE dataset also contains objects traveling on the same path with an easily estimated velocity using physics. For example a variety of objects have been dropped, rolled down a ramp with a constant rolling base object, or swung on a pendulum. I-MOVE's variety in objects, scenery, and motion parameters/movement paths is designed to help create the most robust velocity estimation models yet.

Calculation and Verification of Velocity

The main purpose of this dataset is to provide the necessary data for better prediction of motion parameters, in particular the velocity of an object. Choosing the point of the object in which to document the velocity of is itself is often non-trivial. For example, when a person is walking they have multiple components (legs, arms, torso) moving at different velocities. In order to make our dataset as useful as possible for multiple applications and objects we have attempted to find the ground truth velocity for the center point of each object being tracked. Because of this, it is especially vital that the ground truth is as accurate as possible. To ensure the velocity of each object is correct we have additional purely physics based scenes put in place, some of these include dropping an object, rolling a ball (or rollable object) down a ramp, and swinging a pendulum with the object attached at the end.

With these known physics based environments it becomes possible to use physics equations to find the instantaneous velocities for each set up, and to use this information to help perfect the radar setup helping ensure that the triangulation of the radars used for other setups is accurate. This fine tuning of the radars allows us to provide more accurate velocity ground truths for the non-easily physics calculated



(a) Intel RealSense 415

(b) Intel RealSense 435

(c) ZED

(d) MYNT

Figure 4: SAMPLE IMAGES FROM OUR DEPTH CAMERAS *Though all the above images provide us depth information from the scene, each of them provides a slightly different view point. These viewpoint variations can provide more data while training deep learning approaches. Also some very different camera models may be held back in order to test trained approaches. For example, while 4(a) Intel RealSense 415 and 4(b)Intel RealSense 435 could be used for training, 4(c) ZED and 4(d) Mynt Eye S could be used to test the generalization of the approach.*

set ups. The equations used for each of the setups required the more complex instantaneous velocity calculations to be used as opposed to the more common final velocity equations. This is because we wanted to obtain velocity for each frame/image within the videos collected.

The velocity data for the object drop was computed using the commonly known equation:

$$V = \frac{1}{2}gt^2 \tag{1}$$

For this equation g (gravity) is 9.8 meters per second squared and t is the time since object was released. Because all cameras are returning time stamp information the velocity is easily calculated by pinpointing the timestamp of the moment the object was released and using the time difference between that frame(s) and the future frames in which the ball is falling as t.

In order to accurately calculate the rolling ball / object's instantaneous velocity a more complex approach had to be used. The ramp itself has friction with the ball or rolling object so in order to make the ramp have as little friction a metal surface was placed over the wooden ramp supports. This reduction in friction allows us to use more common and less complicated physics equations. The final velocity (velocity when the rolling object reaches the end of the ramp) is calculated using the following equation:

$$V_{final} = \sqrt{\frac{10}{7}gh} \tag{2}$$

Where g is once again gravity and h is the height of the ramp.

Now that the final velocity is obtained, and given that we know the initial velocity is zero we can find the average acceleration using the equation,

$$a = \frac{v_{final}}{\Delta t} \tag{3}$$

by dividing the final velocity by the change in time (time it takes to reach the bottom of ramp) we can then use this average acceleration to find the velocity at any point between the object starting down the ramp and reaching the ground. To do this we use the following equation:

$$V_t = at \tag{4}$$

This equation multiplies the acceleration down the ramp by the time (t) since the release of the object / when the object started rolling, allowing us to calculate the instantaneous velocity of the object.

In order to calculate instantaneous velocity for a pendulum a series of calculations were also required. We know the length of the string (L) used for the pendulum and the gravitational acceleration (g) so we were able to find the period (time to complete a swing) by using the equation:

$$P = 2\pi\sqrt{\frac{L}{g}} \tag{5}$$

When the period (P) is found we can then use the information we have to find the instantaneous angle of the pendulum also known at θ_t.

$$\theta_t = \theta_{highest} cos(\frac{2\pi}{P}t) \tag{6}$$

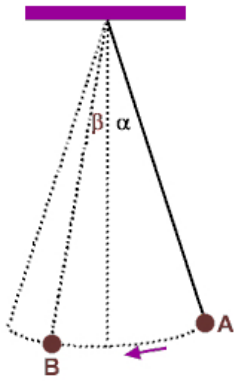


Figure 5: PHYSICS FOR ESTIMATING INSTANTANEOUS VELOCITY OF A PENDULUM *This sketch shows how the equation to find V_B is used.*

The equation uses $\theta_{highest}$ which is the highest point of the swing (or the initial drop angle). P, the period of the swing, and t, the time since $\theta_{highest}$ in which you are trying to find the angle for. Given this angle information we can then find the instantaneous velocity of the pendulum with the equation:

$$V_B = \sqrt{2(gLcos\beta - gLcos\alpha)} \quad (7)$$

V_B is the velocity at the point we are attempting to find. The equation uses g, gravitational acceleration, L, the length of the pendulum string, α, the angle from vertical in which the pendulum was released, and β the angle from vertical the pendulum is currently at in comparison to α. To help clarify this a visual aid accompanying this set up can be seen in Figure 5.

Now that we are able to solve for the instantaneous velocity of the pendulum we apply this to each time-step in the recorded pendulum data. The accuracy of this velocity data is also significantly improved by the fact that we applied this to pendulum drops of 20°or less making it a simple small amplitude pendulum problem and improving the data generated via the prior equations. These physics based setups / environments with known equations used to calculate velocities are also accompanied by two OmniPreSense radars allow us to provide the most accurate velocities we can.

Conclusion

"What we see depends mainly on what we look for", and in this I-MOVE dataset we have chosen to look for that which is crucial to many applications. We presented a novel dataset intended to help researchers progress and refine their approaches to produce more robust motion parameter estimation, specifically the velocity of the object being tracked. We identify several drawbacks, and limitations with the existing datasets in addition to explaining the differences between our dataset and ground truths. We also explain how none of the preexisting datasets contain the necessary information to adequately approach the problem of single moving object velocity estimation. We detail our meticulously crafted setup and explain how ground truth estimation from a dedicated motion parameter sensor like the Doppler radar can be verified using controlled environments and basic laws of physics. To the best of our knowledge this is the first dataset that is directly target to the problem of motion parameter estimation on independently moving objects in a complicated environment.

In future works we plan to extend the dataset to contain more diverse environments, classes of objects, as well as complicated motion paths, such a stunts by a gymnast and skateboarder or a flying Frisbee etc. With this additional information we hope to be able to make more robust models that may be applied to a wider range of applications. As an extension, the community may explore more motion parameters such as angular velocity or rotation in degrees, but it is much harder to obtain ground truths for such setups. Another extension may also be to estimate motion parameters for each limb of a subject rather than the complete body. Though such an addition will be useful for applications oriented towards sports it may not be easy to collect.

Acknowledgement

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

References

Anguita, D.; Ghio, A.; Oneto, L.; Parra, X.; and Reyes-Ortiz, J. L. 2013. A Public Domain Dataset for Human Activity Recognition using Smartphones. In *ESANN*.

Atev, S.; Arumugam, H.; Masoud, O.; Janardan, R.; and Papanikolopoulos, N. P. 2005. A vision-based approach to collision prediction at traffic intersections. *IEEE Transactions on Intelligent Transportation Systems* 6(4):416–423.

Brosh, E.; Friedmann, M.; Kadar, I.; Yitzhak Lavy, L.; Levi, E.; Rippa, S.; Lempert, Y.; Fernandez-Ruiz, B.; Herzig, R.; and Darrell, T. 2019. Accurate Visual Localization for Automotive Applications. 0–0.

Chang, C.-Y.; Huang, D.-A.; Sui, Y.; Fei-Fei, L.; and Niebles, J. C. 2019. D3tw: Discriminative Differentiable Dynamic Time Warping for Weakly Supervised Action Alignment and Segmentation. 3546–3555.

Chatzitofis, A.; Vretos, N.; Zarpalas, D.; and Daras, P. 2013. Three-dimensional monitoring of weightlifting for computer assisted training. In *Proceedings of the virtual reality international conference: Laval virtual*, 3. ACM.

Chatzitofis, A.; Zarpalas, D.; and Daras, P. 2018. A computerized system for real-time exercise performance monitoring and e-coaching using motion capture data. In *Precision Medicine Powered by pHealth and Connected Health*. Springer. 243–247.

Chen, C.; Jafari, R.; and Kehtarnavaz, N. 2015. UTD-MHAD: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor. In *2015 IEEE International Conference on Image Processing (ICIP)*, 168–172.

Chuang, H.-M.; Wojtara, T.; Bergstrm, N.; and Namiki, A. 2018. Velocity Estimation for UAVs by Using High-Speed Vision. *Journal of Robotics and Mechatronics* 30(3):363–372.

Coskun, H.; Achilles, F.; DiPietro, R.; Navab, N.; and Tombari, F. 2017. Long Short-Term Memory Kalman Filters: Recurrent Neural Estimators for Pose Regularization. 5524–5532.

Datta, A.; Kim, J.-S.; and Kanade, T. 2009. Accurate camera calibration using iterative refinement of control points. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, 1201–1208. IEEE.

Einfalt, M.; Zecha, D.; and Lienhart, R. 2018. Activity-conditioned continuous human pose estimation for performance analysis of athletes using the example of swimming. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 446–455. IEEE.

Fan, L.; Zhu, Y.; Zhu, J.; Liu, Z.; Zeng, O.; Gupta, A.; Creus-Costa, J.; Savarese, S.; and Fei-Fei, L. 2018. SURREAL: Open-Source Reinforcement Learning Framework and Robot Manipulation Benchmark. In *Conference on Robot Learning*, 767–782.

Fangjun Jiang, and Zhiqiang Gao. 2000. An adaptive nonlinear filter approach to the vehicle velocity estimation for ABS. In *Proceedings of the 2000. IEEE International Conference on Control Applications. Conference Proceedings (Cat. No.00CH37162)*, 490–495.

Fei, F.; Tu, Z.; Yang, Y.; Zhang, J.; and Deng, X. 2019. Flappy Hummingbird: An Open Source Dynamic Simulation of Flapping Wing Robots and Animals. *arXiv:1902.09628 [cs]*. arXiv: 1902.09628.

Gandhi, T., and Trivedi, M. M. 2006. Pedestrian collision avoidance systems: a survey of computer vision based recent studies. In *2006 IEEE Intelligent Transportation Systems Conference*, 976–981.

Heyman, J. 2019. TracTrac: A fast multi-object tracking algorithm for motion estimation. *Computers & Geosciences* 128:11–18.

Hinedi, S. 1988. An extended kalman filter based automatic frequency control loop. *Telecommunications and Data Acquisition Progress Report* 95:219–228.

Hua, M.; Manerikar, N.; Hamel, T.; and Samson, C. 2018. Attitude, Linear Velocity and Depth Estimation of a Camera Observing a Planar Target Using Continuous Homography and Inertial Data. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 1429–1435.

Jiang, M.-x.; Deng, C.; Pan, Z.-g.; Wang, L.-f.; and Sun, X. 2018. Multiobject Tracking in Videos Based on LSTM and Deep Reinforcement Learning.

Kampelmhler, M.; Mller, M. G.; and Feichtenhofer, C. 2018. Camera-based vehicle velocity estimation from monocular video. *arXiv:1802.07094 [cs]*. arXiv: 1802.07094.

Kim, Y.; Jung, H.; Min, D.; and Sohn, K. 2018. Deep Monocular Depth Estimation via Integration of Global and Local Predictions. *IEEE Transactions on Image Processing* 27(8):4131–4144.

Kim, C.; Li, F.; and Rehg, J. M. 2018. Multi-object Tracking with Neural Gating Using Bilinear LSTM. In Ferrari, V.; Hebert, M.; Sminchisescu, C.; and Weiss, Y., eds., *Computer Vision ECCV 2018*, volume 11212. Cham: Springer International Publishing. 208–224.

Lee, N., and Kitani, K. M. 2016. Predicting wide receiver trajectories in american football. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 1–9. IEEE.

Mahapatra, P. R., and Mehrotra, K. 2000. Mixed coordinate tracking of generalized maneuvering targets using acceleration and jerk models. *IEEE Transactions on Aerospace and Electronic Systems* 36(3):992–1000.

Nakazawa, S.-i.; Ishihara, T.; and Inooka, H. 2003. Real-time algorithms for estimating jerk signals from noisy acceleration data. *International Journal of Applied Electromagnetics and Mechanics* 18(1-3):149–163.

Vision-only egomotion estimation in 6dof using a sky compass | Robotica | Cambridge Core.

Piergiovanni, A. J., and Ryoo, M. S. 2019. Representation Flow for Action Recognition. 9945–9953.

Prez-Ortiz, J. A.; Gers, F. A.; Eck, D.; and Schmidhuber, J. 2003. Kalman filters improve LSTM network performance in problems unsolvable by traditional recurrent nets. *Neural Networks* 16(2):241–250.

Romero-Ramirez, F. J.; Muoz-Salinas, R.; and Medina-Carnicer, R. 2018. Speeded up detection of squared fiducial markers. *Image and Vision Computing* 76:38–47.

Sadeghian, A.; Alahi, A.; and Savarese, S. 2017. Tracking the Untrackable: Learning to Track Multiple Cues with Long-Term Dependencies. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 300–311. Venice: IEEE.

Sigal, L.; Balan, A. O.; and Black, M. J. 2010. HumanEva: Synchronized Video and Motion Capture Dataset and Baseline Algorithm for Evaluation of Articulated Human Motion. *International Journal of Computer Vision* 87(1-2):4–27.

Sturm, J.; Engelhard, N.; Endres, F.; Burgard, W.; and Cremers, D. 2012. A benchmark for the evaluation of RGB-D SLAM systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 573–580. Vilamoura-Algarve, Portugal: IEEE.

Xia, X.; Xiong, L.; Liu, W.; and Yu, Z. 2018. Automated Vehicle Attitude and Lateral Velocity Estimation Using a 6-D IMU Aided by Vehicle Dynamics. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, 1563–1569.

Zamora, I.; Lopez, N. G.; Vilches, V. M.; and Cordero, A. H. 2016. Extending the OpenAI Gym for robotics: a toolkit for reinforcement learning using ROS and Gazebo. *arXiv:1608.05742 [cs]*. arXiv: 1608.05742.

Zhang, X.; Wei, Y.; Feng, J.; Yang, Y.; and Huang, T. S. 2018. Adversarial Complementary Learning for Weakly Supervised Object Localization. 1325–1334.

Zhu, A. Z.; Thakur, D.; zaslan, T.; Pfrommer, B.; Kumar, V.; and Daniilidis, K. 2018. The Multivehicle Stereo Event Camera Dataset: An Event Camera Dataset for 3d Perception. *IEEE Robotics and Automation Letters* 3(3):2032–2039.

A Domain Independent Social Media Depression Detection Model

Sven Marnauzs
Department of Mathematics
Boise State University
svenmarnauzs@u.boisestate.edu

Jugal Kalita
Department of Computer Science
UC Colorado Springs
jkalita@uccs.edu

Abstract

Due to negative social stigmas surrounding mental health disorders, a large portion of the affected population are reluctant to seek help. In consequence, many of those in need remain undiagnosed and uneducated about their condition. Language contains information on the author's mental state and demographic, and has been leveraged by NLP models that learn to predict each one independently or jointly. With the advent of social media, we have access to an unprecedented amount of natural language data. However, these models require large labeled data sets that can be very expensive and time consuming to create. In addition, these data sets are typically derived from a single domain such as Twitter, Facebook, Reddit, Instagram, etc. As such, they are unable to generalize well to other mediums outside of their domain. Focusing on the diagnosis of depression disorder, the present contribution aims to create a domain independent depression detection model that uses a novel combination of deep learning, NLP, and machine learning techniques while using datasets much smaller than those found in the literature.

Introduction

The World Health Organization (WHO) reports that mental health disorders such as depression and anxiety are among the largest contributors to global disability. These types of disorders are detrimental to every aspect of an affected individuals health. Depression alone is estimated to affect more than 300 million people worldwide. Yet, a large portion of this population remains undiagnosed and reluctant to seek help due to the negative social stigma surrounding such actions or time and financial limitations.

There has been a large push to use recent advancements in Natural Language Processing (NLP) that can leverage social media data as a depression detection system (De Choudhury et al. 2013), (Coppersmith, Dredze, and Harman 2014), (Coppersmith et al. 2015). There has also been work on using multitask learning to predict mental disorders (depression, PTSD, bipolar, etc.) and gender simultaneously in an attempt to be able to use smaller datasets (Hovy, Mitchell, and Benton 2017). More recently, there has been work to negate the affects of data imbalance in datasets created for the depression detection task (Cong et al. 2018), (Gerych, Agu, and Rundensteiner 2019).

However, previous studies have either only used data from one type of domain or have data from multiple domains but no way of fine tuning that data for new data sets. To the best of our knowledge, there is not any work being done to create a general depression detection system that can be fine tuned to data outside of its domain. For example, a depression detection classifier based on Facebook data may not classify well on data taken from Twitter. In general, end-to-end deep learning approaches are not feasible on small datasets.

In the present contribution, we propose a general depression detection model. This model will take in labeled natural language data provided by the user, feed it to various depression classifiers from different domains, and then use predictions from those classifiers as features in a machine learning model. Our hypothesis is that the machine learning model will be able to determine which classifiers are able to best diagnose depression in the new domain.

Related Work

Predicting Depression via Social Media

Using the CES-D (Center for Epidemiologic Studies Depression Scale) questionnaire, De Choudhury et al. (2013) determined the depression level of 1,583 crowdsourced subjects. At the end of the session, they asked each subject if they would like to share their Twitter username. From those that agreed, they created a database of 544 twitter user names and their corresponding tweets, and then labeled each user as depressed or not depressed based on their answers to the CES-D questionnaire. The final database contained 171 users classified as depressed and 305 as not depressed. For those classified as depressed, they only kept tweets from one year prior to their onset or diagnosis date in an attempt to create a model that can predict future episodes of depression. They found that their SVM based model with 188 features was able to classify depressed users with about 70 percent accuracy. While their results were promising, their collection and feature extraction methods were very time consuming and expensive.

Multitask Learning for Mental Health Conditions with Limited Data

It has been shown that there can be a considerable performance jump when transitioning from Single Task Learning (STL) to MTL (Caruana 1993). Hovy, Mitchell, and Benton (2017) were the first to use deep-learning in the task of mental disorder detection. In addition, they used a combination of automatic data collection from twitter and hand-annotation to create their labeled database. While their methods may not be as concrete as De Choudhury's, they were certainly less time consuming. They developed neural MTL models trained on twitter data for 8 mental condition prediction tasks and 2 auxiliary prediction tasks (neurotypicality, and gender). They found that their most complex MTL models (ones that simultaneously trained the most tasks) performed significantly better than independent STL models when they had smaller amounts of data on certain conditions such as bipolar disorder and PTSD. They theorize that when they force the model to predict conditions which have large amounts of data available, they significantly improve the prediction accuracy of other similar conditions with small amounts of available data.

X-A-BiLSTM: a Deep Learning Approach for Depression Detection in Imbalanced Data

Deep learning approaches to the social media depression detection task are hindered by imbalanced datasets. In an attempt to lift this weight, Cong et al. (2018) use a novel combination of traditional machine learning techniques and deep-learning. To validate their proposed model, they used the RSDD dataset (Yates, Cohan, and Goharian 2017). Language data of each author in the training set was initially feed into an XGBoost model that significantly reduced the degree of imbalance in the dataset by weeding out negative samples. The positive predictions were then feed into an attention Bi-LSTM deep-learning model to output the final classes of positive and negative samples. Their proposed model surpassed several other state-of-the-art deep-learning models in the social media depression classification task.

Data

Data was collected from two social media domains. The first is the RSDD dataset from Yates et al. (2017), which is composed of public Reddit posts. The second is the CLPsych 2015 shared task Twitter dataset from Coppersmith et al. (2015). Both datasets were created via a combination of automatic retrieval and hand annotated labeling similar to the procedures of Coppersmith et al. (2014). Social media users choose to publicly post statuses of their mental health for a variety of reasons such as looking for support and sympathy from their social network. Another common reason to publicly report their diagnosis is to educate others about their condition. Nevertheless, publicly available posts on being clinically diagnosed with depression provides us with an opportunity to explore the language differences between depressed and non-depressed users. As both the RSDD dataset and the CLPsych 2015 shared task dataset use these publicly available posts of diagnosis, their ground truth values share a similar degree of reliability. This allows us to conduct a exploration of the differences and similarities in how users communicate on their respective platforms. Our intuition tells us that there should be a difference in the way that depressed and non-depressed users use language. For example, a good generalized model trained on domain A should be able to classify users on domain B with a precision close to that of a good model being tested and trained on solely on B. The acquired datasets allow us to explore the correctness of this intuition.

We also make use of age and gender labeled datasets provided by various 2017 PAN shared tasks (Potthast et al. 2017). These datasets are of the tweets from hundreds of Twitter users, where each user is given a label for gender, and then their respective age group (18-24, 25-34, 35-49, 50-XX). Since depression is slightly correlated with age and gender, our intuition tells us that incorporating this information into our model should increase its predictive accuracy.

Data preprocessing

In an attempt to remove the effect of domain specific text patterns on our model, we take all data through various preprocessing techniques. However, we still seek to capture as much unique and quirky language from the users as possible, so we make our best effort strike a balance between preprocessing and leaving the data as is. We also must take into account that our primary model is an n-gram character language model (CLM). Therefore, by reducing the character vocabulary of the text, we substantially reduce the complexity of our model.

RSDD Reddit dataset As Reddit posts have essentially no character limit, they are typically filled with special formatting such as newlines, tabs, links, etc. The first step was to remove these unnecessary gaps between sections of text. This first step greatly reduced the complexity of our higher order n-gram CLMs. Next, we substituted references to links, usernames, and subreddits with special characters or just simply removed them. It is possible that references to subreddits could potentially help a model determine the classification for a user. For example, depressed users may mention r/depression more often than other users. However, to keep our model as generalized as possible, we chose to remove such references as they are not present in other social media domains. Another decision we made was to change all text to lowercase for the sake of simplicity.

CLPsych 2015 and PAN 2017 Twitter dataset Even though Twitter has its own unique style when compared to Reddit, most of the preprocessing steps were identical. We remove links, usernames, newlines, tabs, and unnecessary whitespace. Unique to twitter is the retweet type of post. We removed all retweets as these tweets are not written by the user. We chose to keep in standard punctuation, however these may need to be removed so that we can reduce the complexity of our CLMs. At the order four CLM, we already had 10^6 four-grams, and we were not able to upscale

to a higher order due to a memory error. Due to complications, the PAN dataset was not filtered or processed in any way.

Model Architecture

Baseline Model Social media is rampant with internet-slang, misspelled words, niche text patterns, and obscure references. As a simple solution to capture all of these nuances, we employ n-gram character-level language models with k-smoothing to score an aggregate of x tweets at a time, where the scores indicate whether a user is depressed or not. Our approach follows closely to the MIQ team approach in CLPsych 2015 shared task competition; see Coppersmith et al. (2015) for more details. Through this approach we examine how likely a sequence of characters is to be generated by a depressed or non-depressed user. We begin by building an n-gram character-level model for each condition based on the training subset of our data. For each user in the test set, we score an aggregate of x tweets based on its character-level n-grams. Let T_x be the aggregate of x tweets for a given user, D the CLM for depression, and D' the CLM for control. Our scoring function f is thus

$$f(T_x) = \frac{\sum_{T_x} \log p(c_D) - \log p(c_{D'})}{|T_x|}$$

where $p(c_D)$ is the probability of an n-gram character sequence appearing in model D, and $p(c_{D'})$ is the probability of an n-gram character sequence appearing in model D'. To compute the final score for each individual user, we average the scores in a sliding window of five x tweet aggregates at a time. Once a score is obtained for each window, the median of the set of window scores is used as the final score for the user. Our model was essentially identical between training and testing on the RSDD and CLPsych datasets. However, for the RSDD data, we did not use a sliding window. The predictive accuracy of our CLMs will serve as our cross-domain baseline.

Improved model To improve upon the performance of our baseline cross-domain model, we propose a model that uses a combination of conventional machine learning, deep learning, and multi-task learning to create an automatic domain independent social media depression detector. We use a deep-learning transformer model from Google's Tensor2Tensor library to pre-train age, gender, and depression classifiers (Vaswani et al. 2018). To test our cross-domain model, we use the RSDD dataset to train our depression classifier, and then test the model on our Twitter data. We use the depression, gender, and age labels as semi-supervised features in a Random Forest Classifier (RFC) model, where the labels are produces by feeding in the Twitter data to each pre-trained classifier. We show a generalized diagram of our model in Figure (1), where $C : [C_1, C_2, ..., C_n]$ is the set of classifiers, n is the number of classifiers, a is the number of authors in the local database, $O : [C_{1,i}, C_{2,i}, ..., C_{n,i}]$ is a $n \times a$ dimensional matrix that holds the outputs of each classifier for author i where $i \, \epsilon [1, 2, ..., a]$, and y is an array that contains a label for each author (depressed or not depressed).

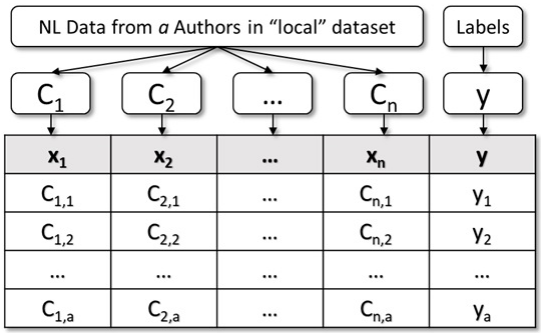


Figure 1: Semi-supervised ML Model Flowchart

As can be seen in the above diagram, natural language data from n Twitter authors is processed by the pre-trained classifiers in C, and the outputs O will be used as values for each author's feature vector. This dataset of feature vectors and corresponding y labels will then be split into training and testing datasets used to train the RFC model.

Experiments

RSDD For the RSDD data, we experimented with one, two, and three gram CLMs. We also experimented with many different combinations of window and aggregate post size. We found that we obtained the best results when compiling 20 ($x = 20$) posts and omitting the window approach entirely. Filtered and unfiltered data were also trained and tested on. Filtered data out-preformed unfiltered data in every case.
We also trained a transformer model on the RSDD data, but we did not conduct an in depth evaluation of this model. This model was only used for generating semi-supervised depression labels of Twitter users in our CLPsych 2015 dataset.

CLPsych 2015 shared task Due to a smaller dataset size, we were able to expand to a four order CLM in addition to the lower orders. We also experimented with different combinations of window and aggregate post size, and found that aggregating 20 posts together with a window size of 5 gave us the best results. As with the RSDD experimentation, filtered data out-preformed unfiltered in every case.
We train a transformer model on the CLPsych Twitter data to compare its predictive accuracy against our cross-domain model and our CLM model. The transformer model gave a binary label to each post of a user. For the final score of each user, we used the median of an array of 20 post sliding window scores, where each window was scored as the ratio of depressed posts to non-depressed posts. This experiment was done to verify that the results of the transformer model trained on a single domain were as good as or better than our baseline model results.

Cross domain testing baseline The main objective of the current contribution is to approve upon a baseline test on how well a model trained on one social media domain will fair when given data from a different domain. We experimented with using the one, two, and three RSDD CLMs

to score Twitter users from the CLPsych 2015 shared task dataset. As this kind of cross domain testing has not been explored before, we use the result as a baseline test, and seek to improve upon it.

Improved cross domain model To improve upon the baseline, we use a RFC with features extracted from feeding Twitter data into our pre-trained transformer model classifiers. Each pre-trained classifier outputs a label for each post of a user. We then use the sliding window approach (size 20) to produce the final labels for gender, age, and depression for each user in the Twitter dataset. This dataset was split into training and testing, where the training set was used to train the RFC model, and the testing set was used for validation and baseline comparison tests.

Results

Baseline model results

Figure (1) shows the best results of training and testing a CLM on the RSDD dataset. We found that taking the median of 20 post aggregate scores that were scored by a second order CLM trained and tested on filtered text gave us the best results. The AUC of our model is 0.79.

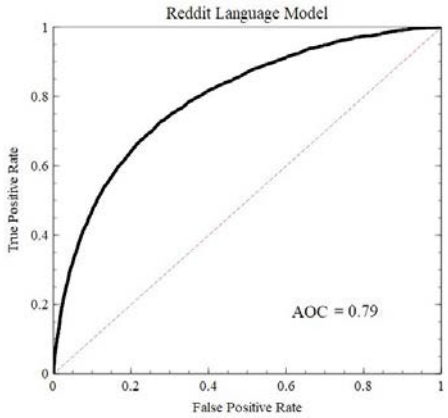


Figure 2: The ROC curve for a second order CLM trained and test on filtered Reddit data.

Figure (2) shows the best results of training and testing a CLM on the CLPsych dataset. We found that taking the median of window scores of size five gave us the best results, where each window took the mean of 20 post aggregate scores that were scored by a second order CLM trained and tested on filtered text. We found the average precision was 0.65, and the AOC was 0.80 for our model.

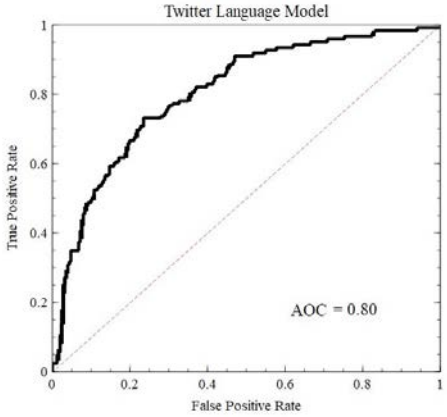


Figure 3: The ROC curve for a order four CLM trained and tested on filtered Twitter data.

Figure (3) shows how well our order three Reddit CLM preformed on classifying twitter users. The parameters of both models were identical to the models which gave us the best results in their respective domains. The AUC and average precision of the Reddit based CLM was 0.67 and 0.55 respectively. For comparison, the AUC and average precision of the Twitter based CLM was 0.78 and 0.62 respectively.

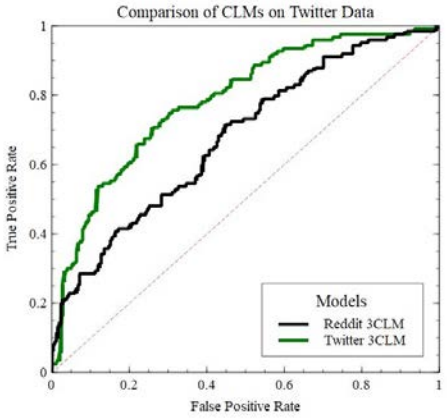


Figure 4: A comparison of ROC curves for order three CLMs trained on filtered Twitter data and Reddit data, and then tested only on Twitter data.

Figure (4) shows a comparison between a $n = 3$ CLM trained and tested on Twitter data, and a $n = 3$ CLM trained on Reddit but tested on Twitter. For the Twitter CLM, we obtained an AUC score of 0.78 with and average precision of 0.63. The Reddit CLM gave us an AUC score of 0.63 with an average precision of 0.55. We were unable to upscale to a higher-order CLM for the RSDD data due to various complications. As such, we thought it fair to compare the 3-gram RSDD CLM to the 3-gram CLPsych CLM instead of the better performing 4-gram CLM.

Transformer/RFC model results

We explore the results of our "improved" models against the baseline models. We used the best results from each model type instead

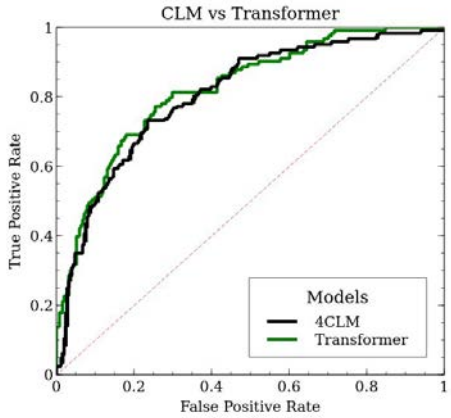


Figure 5: A comparison between the transformer model and $n = 4$ CLM trained and tested on the CLPsych 2015 Twitter data.

Figure (5) shows the results of the $n = 4$ CLM and our transformer model trained and tested on Twitter data. This single domain experiment was conducted to verify that our transformer model could perform as well as our baseline model when trained on a single domain. The 4-gram CLM had an AUC of 0.80 with an average precision of 0.65. Our transformer model had an AUC of 0.83 with an average precision of 0.72

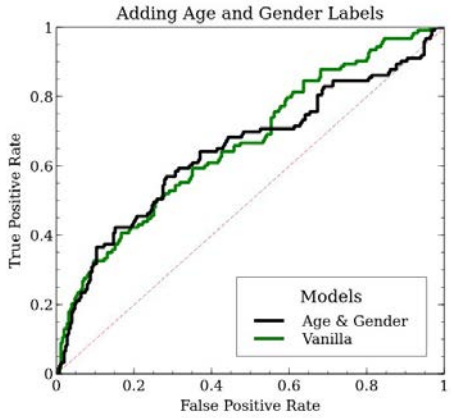


Figure 6: A comparison between our RFC model with semi-supervised labels and a "vanilla" (only depression labels) transformer model when tested on the CLPsych data. Both models were never trained on any of the CLPsych data.

We explore the effect of adding age and gender information to our model in Figure (6). Both models had an AUC of 0.65. Our vanilla model gave us an average precision of 0.53, while our RFC model had an average precision of 0.52.

Discussion

The performance of our CLM models on Reddit and Twitter data were surprisingly good (Figure (1) and (2)). We expect that as we are able to increase the order of our CLMs we will achieve better performance. As can be seen in Figure (3), our Reddit base CLM model does not preform as well as the Twitter based one. To have a fair comparison between models, both models used third order CLMs. We attribute the poor performance to the large differences in how Twitter and Reddit operate as social media sites. One possible explanation is that CLMs are not be able to pick up the true sentiment of users posts. Another could be that we simply did not take advantage of higher order CLMs. We expect that as we are able to increase the order of both CLMs, the gap between their performance will decrease. However, we do not know if this gap will ever converge using just CLMs. Nevertheless, we will use the CLM models as our baseline performance.

The performance of Google's transformer model trained and tested on the CLPsych data set was good, but not as good as we had hoped it would be, as it only slightly outperformed our simple 4-gram CLM. However, we did not attempt to use an aggregate training or scoring method. These methods significantly improved the performance of our CLMs. Therefore, we expect that we would achieve a similar performance boost for the transformer model as well.

Both the "vanilla" transformer model and our RFC model (with age and gender labels) produced very similar results. This came to our surprise, as we intuitively theorized that incorporating this extra information into a model would increase the cross-domain generalization, thus increasing the accuracy once shown data from another domain. There are multiple explanation for why our results were not satisfactory. First, we did not pre-process the age and gender data due to time constraints and complications. We also did not do a through evaluation of how well the age and gender models were performing on ground truth age and gender labeled Twitter data. Going through these tasks and evaluations would likely improve the accuracy of our semi-supervised labels for our RFC model. There is also a possibility that the RFC is not taking advantage of the gender and age labels due to over-weighing the depression labels.

Future Work

Model exploration and fine-tuning, multi-task learning, and extra features

First, our RFC may not be taking advantage of additional information. We should explore other models that will better use this extra data. We will also attempt to fine-tune parameters of the RFC to see if it can make better use of all its features. There is also a good chance that we can make our transformer perform better across all tasks by fine-tuning parameters and learning how to train it better, as these are not trivial tasks. For example, we will train a transformer model on an aggregates of posts instead of individual ones. However, we are experiencing memory error when attempting to do so. We will likely find a way to overcome this obstacle in the near future, and when we do, we expect a large across-the-board performance increase. Text CNNs for post classification are also a topic of interest (Kim 2014). We have already begun initial experimentation using this approach.

The possibility of using multi-task learning for classifying

depressed users is very intriguing. We have proposed using a text CNN network to pre-train on the gender and age data, and then use those trained models to calculate vectors for each sample in the CLPsych dataset. We would then train another text CNN model on the CLPsych depression data while incorporating the pre-calculated vectors. We have created the groundwork for this approach, but we have yet to fully implement it. We theorize that making our text CNN model train on several tasks at once will increase the performance across all tasks. It could also improve the cross-domain compatibility of the model (i.e. perform good when given data from other domains).
Finally, the addition of features in our RFC could help improve the cross-domain performance of the model. Other features considered are personality traits, the general emotion of a users posts, and additional depression classifiers. These addional depression classifiers could each have differnet scoring methods and models. For example, we could use a combination of multi-task and single-task models using different parameters for training and scoring (i.e. different window sizes).

Obtaining relevant datasets

For our model to work as planned, we must seek out additional datasets other than RSDD and CLPsych 2015. We have already obtained several Twitter datasets related to age and gender tasks that were created for a shared task competitions at the PAN workshops held annually at CLEF. We have also obtained the DAIC-WOZ dataset which contains transcribed interviews of subjects and their ground truth scores of depression based on PHQ-8 questionnaires. Aside from these datasets, there is possibility that we will want to incorporate a dataset from a third social media domain such as Facebook. This would allow us to perform three-fold cross validation of our cross-domain model. A three-fold cross validation scheme would greatly increase the validity of our model because using just two social media domains would likely lead to over fitting our model.

Conclusion

Much work has been done to improve social media depression detection models. However, these models are trained and tested on a single social media domain. As such, they are likely unable to generalize well. We have begun to confirm our intuition by conducting a baseline test using n-gram CLMs. We found that the baseline models performed poorly when trained on Reddit data but subjected to Twitter data for testing. We sought to improve upon this baseline by using a RFC with semi-supervised features extracted by feeding Twitter data into transformer models pre-trained on depression, gender, and age labeled datasets. While our initial results are not satisfactory, there are many untraveled avenues to explore. We cannot yet conclude that our proposed "improved" model will not perform better than those in the literature when subjected to data from external domains.

References

Caruana, R. 1993. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on Machine Learning*, 41–48. Morgan Kaufmann.

Cong, Q.; Feng, Z.; Li, F.; Xiang, Y.; Rao, G.; and Tao, C. 2018. Xa-bilstm: a deep learning approach for depression detection in imbalanced data. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 1624–1627. IEEE.

Coppersmith, G.; Dredze, M.; Harman, C.; Hollingshead, K.; and Mitchell, M. 2015. CLPsych 2015 Shared Task: Depression and PTSD on Twitter. In *Proceedings of the 2nd Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality*, 31–39. Denver, Colorado: Association for Computational Linguistics.

Coppersmith, G.; Dredze, M.; and Harman, C. 2014. Quantifying mental health signals in twitter. In *Proceedings of the workshop on computational linguistics and clinical psychology: From linguistic signal to clinical reality*, 51–60.

De Choudhury, M.; Gamon, M.; Counts, S.; and Horvitz, E. 2013. Predicting depression via social media. In *Seventh international AAAI conference on weblogs and social media*.

Gerych, W.; Agu, E.; and Rundensteiner, E. 2019. Classifying depression in imbalanced datasets using an autoencoder-based anomaly detection approach. In *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, 124–127.

Hovy, D.; Mitchell, M.; and Benton, A. 2017. Multitask Learning for Mental Health Conditions with Limited Social Media Data. In *EACL*.

Kim, Y. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Potthast, M.; Rangel, F.; Tschuggnall, M.; Stamatatos, E.; Rosso, P.; and Stein, B. 2017. Overview of panâĂŹ17. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, 275–290. Springer.

Vaswani, A.; Bengio, S.; Brevdo, E.; Chollet, F.; Gomez, A. N.; Gouws, S.; Jones, L.; Kaiser, L.; Kalchbrenner, N.; Parmar, N.; Sepassi, R.; Shazeer, N.; and Uszkoreit, J. 2018. Tensor2tensor for neural machine translation. *CoRR* abs/1803.07416.

Yates, A.; Cohan, A.; and Goharian, N. 2017. Depression and self-harm risk assessment in online forums. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2958–2968. Association for Computational Linguistics.

Solving Arithmetic Word Problems Automatically Using Transformer and Unambiguous Representations

Kaden Griffith and Jugal Kalita
University of Colorado Colorado Springs
kadengriffith@gmail.com and jkalita@uccs.edu

Abstract

Constructing accurate and automatic solvers of math word problems has proven to be quite challenging. Attempts using machine learning have so far been trained on corpora specific to math word problems to produce arithmetic expressions in infix notation before answer computation. Neural networks have struggled to generalize, even when trained on large and diverse datasets. This paper outlines the use of Transformer networks trained to translate simple math word questions to equivalent mathematical expressions in infix, prefix, and postfix notations. We use a pretraining approach to translation, followed by training on corpora specific to word problems, and compare results produced by a large number of neural configurations. We find that the use of the prefix notation produces the best results, surpassing the state of the art.

Introduction

Solving a math word problem (MWP) starts with one or more sentences describing a problem to be understood. These sentences are processed to produce an arithmetic expression, which is evaluated to provide an answer. Recent neural approaches to solving arithmetic word problems have used various flavors of recurrent neural networks (RNN) as well as reinforcement learning. Such methods have had difficulty achieving a high level of generalization. Often, they can extract the relevant numbers, but misplace them in the generated expressions. At other times, they have replaced the numbers in the problem statement with arbitrary other numbers when formulating corresponding math expressions. The infix notation used requires pairs of parenthesis to be balanced and also placed correctly, bracketing the right numbers. There have been problems with both of these requirements as well.

Figure 1: Misinterpretations of a MWP.

Question:
At the fair Adam bought 13 tickets. After riding the ferris wheel he had 4 tickets left. If each ticket cost 9 dollars, how much money did Adam spend riding the ferris wheel?
Some possible expressions that can be produced:
(13 - 4) * 9, 9 * 13 - 4, 5 * 13 - 4, 13 - 4 * 9,
13 - (4 * 9), (9 * 13 - 4)

Figure 1 gives examples of some infix representations that a machine learning word problem solver can produce from a simple word problem. Of the expressions shown, only the first one is correct. We start with a hypothesis that the sole use of the infix expression representation as a precursor to solving such problems attributes to some of these problems in automatic solvers. In modern application of math, we are used to using infix notation, but the formulation of expressions does not make this requirement.

We have also noticed that the actual numbers used in MWPs vary widely from problem to problem. Real numbers can take any conceivable value, making it almost impossible for a neural network to learn good representations for them. Thus, we hypothesize that replacing the numbers in the problem statement with generic tags like $\langle \mathrm{n1} \rangle$, $\langle \mathrm{n2} \rangle$, and $\langle \mathrm{n3} \rangle$ and saving their values as a pre-processing step, will not take away from the generality of the solution, but will suppress the problem of fertility in number generation, which leads to the introduction of numbers not present in the question sentences, in the relevant solution.

Another hypothesis that forms the basis of our approach is that a neural network which has been pre-trained on general language knowledge will be better able to "understand" the semantics of the problem to produce the correct arithmetic expressions with the tags we use.

We use the Transformer model [Vaswani et al., 2017] to solve arithmetic word problems as a particular case of machine translation from text to the language of mathematical expressions. Transformers in various configurations have become a staple of NLP in the past two years. Past neural approaches did not treat this problem as pure translation like we do but augmented the neural architectures with various external modules such as parse trees or used deep reinforcement learning. In this paper, we demonstrate that Transformers can be used to solve MWPs successfully. We show that our results outperform state-of-the-art results by [Wang et al., 2018, Hosseini et al., 2014, Kushman et al., 2014, Roy, Vieira, and Roth, 2015, Robaidek, Koncel-Kedziorski, and Hajishirzi, 2018], which we use for comparison.

We organized our paper as follows. The second section presents related work. Then we discuss our approach. We follow by an analysis of experimental results and how they compare to recent networks. We also discuss our successes and shortcomings in this section. Finally, we share our con-

cluding thoughts and end with our direction to future work.

Related Work

Many past strategies to solve math word problems have utilized rules and templates to match sentences to arithmetic expressions. Some such approaches seemed to solve problems impressively within a narrow domain, but performed poorly when out of domain, lacking generality that is necessary to solve common questions [Bobrow, 1964, Bakman, 2007, Liguda and Pfeiffer, 2012, Shi et al., 2015]. Kushman et al. (2014) used feature extraction and template-based categorization by representing equations as expression tree forests and finding a near match. Such methods required human intervention in the form of feature engineering and development of templates and rules, which is not desirable for expandability and adaptability. Hosseini et al. (2014) performed statistical similarity analysis to obtain acceptable results, but did not perform well with texts that were dissimilar to training examples.

Existing approaches have used various forms of auxiliary information. For example, Hosseini et al. (2014) used verb categorization to identify important mathematical cues and contexts. Mitra and Baral (2016) used predefined formulas to assist in matching. Koncel-Kedziorski et al. (2015) parsed the input sentences, enumerated all parses, and learned to match, requiring expensive computations. Roy and Roth (2017) performed searches for semantic trees over significant computational spaces.

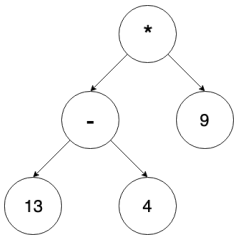
Some recent approaches have transitioned to using artificial neural networks. Semantic parsing of MWPs is a recently developed strategy which takes advantage of RNN architectures to parse math word problems directly into equations or expressions in a uniquely developed math-specific language [Shi et al., 2015, Sun et al., 2019]. Systems using RNNs have shown promising results, but they have had difficulties correctly learning balanced parenthesis, and also, sometimes incorrectly choose numbers when generating equations. Most recently, Sun et al. (2019) used a Bi-Directional LSTM architecture for math word problems. Huang et al. (2018) used a deep reinforcement learning model to achieve character placement in both seen and novel equation templates. Wang et al. (2018) also used deep reinforcement learning to solve MWPs more comprehensively than RNNs.

Approach

We view MWP solving as a sequence-to-sequence translation problem. Systems using RNNs have excelled in sequence-to-sequence problems such as text translation and question answering. The recent introduction of attention mechanisms has improved the performance of RNN models. In particular, Vaswani et al. (2017) introduced the Transformer network, which uses stacks of attention layers instead of recurrence. The use of Transformers in various configurations has shown vastly improved results, achieving state-of-the-art performance in many natural language processing tasks.

Figure 2: Representations of a MWP.

Question:
At the fair Adam bought 13 tickets. After riding the ferris wheel he had 4 tickets left. If each ticket cost 9 dollars, how much money did Adam spend riding the ferris wheel?
Expression Tree:



Output:
Prefix: - 13 4 * 9
Postfix: 13 4 - 9 *
Infix: (13 - 4) * 9
Computed Answer:
81 dollars

We use several configurations of Transformer networks to learn the prefix, postfix, and infix notations of each MWP equation independently. Prefix and postfix representations of equations do not contain parentheses, which has been a source of unnecessary confusion in some approaches. Figure 2 shows the three notations for a problem. We expect that the use of suitable training data, in particular, if the learned target sequences are less likely to mislead or throw off a neural network, may help the learning of the model to be more robust.

We train on standard datasets, which are readily available and commonly used for our task. Our method considers the translation of English text to simple algebraic expressions, although the general approach can be used for other tasks as well. After performing experiments by training directly on math word problem corpora, we perform a different set of experiments by pretraining our network on general language corpora. The success of pre-trained models such as ELMo [Peters et al., 2018], GPT-2 [Budzianowski and Vulić, 2019], and BERT [Devlin et al., 2018] for all manners of natural language tasks, may encourage better learning of our system. However, in our case, the output is not natural language, but algebraic expressions, and that is why we were apprehensive at first. Our success in such heterogeneous translations opens up other possible usages of such pre-trained networks.

Data

Four individual datasets are combined to create a super-collection of MWPs that we call MWP-Data. The datasets contain addition, subtraction, multiplication, and division word problems. Questions in MWP-Data pair with equivalent math equations.

1. **AI2** [Hosseini et al., 2014]. AI2 is a collection of 395 problems. Each problem contains multiple numeric values, where some may not be relevant for answering the question. These questions are addition and subtraction problems.
2. **CC** [Roy and Roth, 2016]. The Common Core dataset contains 2-step algebra word questions. The Cognitive Computation Group gathered these questions. We use a total of 600 unique MWPs from this set in training. This included all of the set's examples.
3. **IL** [Roy, Vieira, and Roth, 2015]. The Illinois dataset contains 1-step algebra word questions. The Cognitive Computation Group also compiled these questions. All of 562 MWPs were used in training from this set.
4. **MAWPS** [Koncel-Kedziorski et al., 2016]. MAWPS is a relatively large collection of MWPs which are primarily from other MWP datasets. We use 2,373 of 3,915 MWPs from this set. The problems not used from this set were more complex problems requiring systems of equations to derive a final solution.

We take a randomly sampled 95% of MWP-Data for training. Networks test on the 5% of withheld examples from each MWP dataset. Training and testing are repeated three times, and reported results are an average between the three outcomes.

Representation Conversion

We take a simple approach when converting infix expressions found in MWP-Data to the other two representations. We strip out the equals sign and variable representing the solved value in each equation. Then two stacks are filled; one with operators found in the equation and the other with the operands. From these stacks, we form a binary tree structure. A given expression tree resembles the one depicted in Figure 2. Traversing an expression tree in pre-order results in a prefix conversion. Post-order traversal gives us a postfix expression. We pre-process the original infix expression as well, removing the variable, and equals sign. Three versions of our training and testing data are used to account for each conversion. By training on different representations, we expect our test results to change.

Pretraining

We pre-train half of our networks to endow them with a good foundational knowledge of English. Pretraining models on significant-sized language corpora has been a common approach recently. We explore the pretraining approach using a general English corpus because our MWPs resemble the English language. For this task, we use the IMDb Movie Reviews dataset [Maas et al., 2011]. This set contains 314,041 unique sentences. Since movie reviewers wrote this data, it is a reference to natural language not necessarily related to arithmetic. Although we pre-train, we do so using a relatively small and specialized corpus that is readily available, to investigate the efficacy of pretraining for math word problems. Training on a much bigger and general corpus like Wikipedia may make the language model stronger, but we leave this for future work.

We compare pre-trained models to non-pre-trained models to observe performance differences. Our pre-trained models are trained in an unsupervised fashion to improve the encodings of our fine-tuned solvers.

Method: Training and Testing

The input sequence is a natural language specification of an arithmetic word problem. The MWP questions and equations have been encoded using the subword text encoder provided by the TensorFlow library. The ideal output will be an expression in prefix, infix, or postfix notation, which then can be manipulated further and solved to obtain a final answer.

Many of the examples in MWP-Data contain unique numbers. Rare terms are adverse for generalization since many of these numbers occur only once in the dataset. The networks may have difficulty dealing with the presence of unique values. As a precaution to this issue, our networks do not consider any relevant numbers during training. Before the networks attempt any translation, we pre-process each question and expression in MWP-Data by a number mapping algorithm. This algorithm replaces each numeric value with a corresponding identifier (e.g., $\langle n1 \rangle$, $\langle n2 \rangle$, etc.), and remembers the necessary mapping. We find that this approach significantly improves how networks interpret each question. When translating, the numbers in the original question are tagged and cached. From the encoded English and tags, a predicted sequence resembling an expression presents itself as output. Since each network's learned output resembles something like $\langle n1 \rangle + \langle n2 \rangle * \langle n3 \rangle$, we use the cached tag mapping to replace the tags with the corresponding numbers and return a final mathematical expression.

Three representation models are trained and tested separately: Prefix-Transformer, Postfix-Transformer, and Infix-Transformer. For each experiment, we use representation specific Transformer architectures. Each model uses the Adam optimizer with $beta_1 = 0.95$ and $beta_2 = 0.99$ with a standard epsilon of $1e^{-9}$. The learning rate is reduced automatically in each training session as the loss decreases. Throughout the training, each model respects a 10% dropout rate. We employ a batch size of 128 for all training. The networks are trained on a machine using 1 Nvidia 1080 Ti graphics processing unit (GPU).

We compare medium-sized, small, and minimal networks to show if network size can be reduced to increase training and testing efficiency while retaining high accuracy. Networks over six layers have shown to be non-effective for this task. We later discuss the issues that arose when attempting such configurations. We tried many configurations of our network models, but report results with only three configurations of Transformers.

- **Transformer Type 1:** This network is a small to medium sized network consisting of 4 Transformer layers. Each layer utilizes 8 attention heads with a depth of 512 and a feed-forward depth of 1024.
- **Transformer Type 2:** The second model is small in size,

using 2 Transformer layers. The layers utilize 8 attention heads with a depth of 256 and a feed-forward depth of 1024.

- **Transformer Type 3:** The third type of model is minimal, using only 1 Transformer layer. This network utilizes 8 attention heads with a depth of 256 and a feed-forward depth of 512.

Objective Function We calculate the loss in training according to the mean categorical cross-entropy formula. Evaluation between the possible translation classes (all vocabulary subword tokens) and the produced class (predicted token) is the metric of performance here. During each evaluation, target terms are masked, predicted, and then compared to the masked (known) value. This process is applied to every determined subword in a translation and then the model loss is adjusted according to the mean of the translation accuracy.

$$loss = \sum_{i=1}^{I} \frac{1}{J} \sum_{j=1}^{J} \left(-\sum_{k=1}^{K} target_{j,k} * log\big(p(j \in k)\big) \right) \quad (1)$$

where $K = |Translation\ Classes|$, $J = |Translation|$, and I is the number of examples.

Experiment 1: Representation Many of the problems encountered by prior approaches seem to be encouraged by the use of infix notation. In this experiment, we compare translation BLEU-2 scores. Traditionally, a BLEU score is a metric of translation quality. Our presented BLEU scores represent an average of scores a given model received over each of the target test sets. We use a standard bi-gram weight to show how accurate translations are within a window of two adjacent terms. After testing translations, we calculate an average BLEU-2 per test set, which is related to the success over that data. The scores for each dataset are then averaged for the presented value.

$$model_{avg} = \frac{1}{N} \sum_{n=1}^{N} BLEUavg_n \quad (2)$$

where N is the number of test datasets, which is 4.

Experiment 2: State-of-the-art This experiment compares our networks to recent previous work. We count a given test score by a simple "correct versus incorrect" method. The answer to an expression directly ties to all of the translation terms being correct, which is why we do not consider partial precision. We compare average accuracies over 3 test trials on different randomly sampled test sets from each MWP dataset. This calculation more accurately depicts the generalization of our networks.

Effect of Pretraining We also explore the effect of language pretraining. Half of the models are pre-trained on unlabelled English data. This training occurs over 30 iterations to install an advanced level of language understanding before training on MWP-Data. The same Transformer architectures are also trained solely on MWP-Data. Each model is trained on MWP-Data for 300 iterations before testing.

$$model_{avg} = \frac{1}{R} \sum_{r=1}^{R} \left(\frac{1}{N} \sum_{n=1}^{N} \frac{C \in n}{P \in n} \right) \quad (3)$$

where R is the number of test repetitions, which is 3; N is the number of test datasets, which is 4; P is the number of MWPs, and C is the number of correct equation translations.

Results

We now present the results of our various experiments. We compare the three representations of target equations and three architectures of the Transformer model in each test. Not all past work has used the data we use here, but we attempt to communicate our capabilities compared to past neural approaches to the best of our ability.

Table 1: BLEU-2 comparison for Experiment 1.

(Type) Model	Average
Pre-trained	
(1) Prefix-Transformer	94.03
(1) Postfix-Transformer	92.71
(1) Infix-Transformer	93.59
(2) Prefix-Transformer	93.51
(2) Postfix-Transformer	92.92
(2) Infix-Transformer	93.34
(3) Prefix-Transformer	93.39
(3) Postfix-Transformer	92.84
(3) Infix-Transformer	93.14
Non-pre-trained	
(1) Prefix-Transformer	94.95
(1) Postfix-Transformer	87.55
(1) Infix-Transformer	93.56
(2) Prefix-Transformer	**95.57**
(2) Postfix-Transformer	94.01
(2) Infix-Transformer	93.39
(3) Prefix-Transformer	95.13
(3) Postfix-Transformer	94.03
(3) Infix-Transformer	93.36

Results of Experiment 1 are given in Table 1. By using BLEU scores, we assess the translation capability of each network. This test displays how networks understand different math representations to a character summary level. We compare by average BLEU-2 accuracy among our tests in the *Average* column of Table 1 to communicate these translation differences.

From our results, prefix representation of our target language performs better than the generally used infix notation. The best performing network was a non-pre-trained type 2 prefix Transformer arrangement. Infix notation was better understood than postfix in the smaller models.

The reasoning behind why similar representations provide different results is somewhat unclear. One hypothesis to explain this is related to the way attention in the systems is

accumulated. A network is more likely to produce a successful translation if the operators, determined by language, are close to one another. Numbers and operators appear partially grouped in both prefix and postfix representations, which could be the cause of observable enhanced learning.

Table 2 provides detailed results of Experiment 2. Results by [Wang et al., 2018, Hosseini et al., 2014, Roy, Vieira, and Roth, 2015, Robaidek, Koncel-Kedziorski, and Hajishirzi, 2018] are sparse but indicate the scale of success in past approaches. Prefix, postfix, and infix representations in Table 2 shows that network capabilities are changed by how teachable the target data is.

While our networks fell short of Wang et al., 2018 AI2 testing accuracy, we present state-of-the-art results for the remaining three datasets. The type 2 prefix Transformer received the highest testing average of 86.73% accurate. Comparatively, the Transformer network performs well and requires little computation time.

Our attempt of language pre-training fell short of expectations. It is odd to see that more examples of English does not improve the understanding of MWPs. Use of advanced embedding techniques tried by Robaidek, Koncel-Kedziorski, and Hajishirzi, 2018 also seem to limit MWP coherence. In future attempts, using a more general corpora of language could help grow semantic ability.

Analysis

All of the network configurations used were very successful for our task. To display the capability of our most successful model (type 2 prefix Transformer), we present some outputs of the network in Figure 3.

Even when incorrect, the syntax of math expressions was strongly held. For the majority of questions, our translators were able to determine operators based solely on the context of language.

Larger networks in tandem with pre-training have been shown, in many cases, to relate semantics of natural language better. Our pre-training was unsuccessful in improving accuracy, even when applied to networks larger than those reported. We may need to use more general language, or pre-train on very math specific texts to be successful. Our results support our thesis of infix limitation.

Error Analysis Our system, while performing above standard, could still apply some improvements. One issue originates from the algorithmic pre-processing of our questions and expressions. In Figure 4 we show an example of one such issue. The excerpt comes from a type 3 non-pre-trained Transformer test. The example shows that identifier $\langle n1 \rangle$ was seemingly overlooked after translation. The issue is attributed to the identifier algorithm only considering numbers in the problem. Observe in the question that the word "eight" is the number we expect to relate to $\langle n2 \rangle$. Our identifying algorithm could be improved by considering such number words and performing conversion to a numerical value. If our algorithm performed as expected, the identifier $\langle n1 \rangle$ would be related with 4 (the first occurring number in the question) and $\langle n2 \rangle$ with 8 (the converted number word appearing second in the question).

Figure 3: Successful prefix translations.

AI2
A spaceship traveled 0.5 light-year from earth to planet x and 0.1 light-year from planet x to planet y. Then it traveled 0.1 light-year from planet y back to Earth. How many light-years did the spaceship travel in all?
Translation Produced:
+ + 0.5 0.1 0.1

CC
There were 16 friends playing a video game online when 7 players quit. If each player left had 8 lives, how many lives did they have total?
Translation Produced:
* 8 - 16 7

IL
Lisa flew 256 miles at 32 miles per hour. How long did Lisa fly?
Translation Produced:
/ 256 32

MAWPS
Debby's class is going on a field trip to the zoo. If each van can hold 4 people and there are 2 students and 6 adults going, how many vans will they need?
Translation Produced:
/ + 2 6 4

Figure 4: Number replacement errors.

Question (MAWPS)
Melanie is selling 4 gumballs for eight cents each. How much money can Melanie get from selling the gumballs?
Correct Translation (Infix)
4 * 8
Hypothesized Translation
$4 + < n1 >$

The overall translation was incorrect whether or not our algorithm was successful, but it is essential to analyze problems like these that result in future improvements. Had all questions been tagged correctly, our performance would have likely improved.

One approach not tried here is the use of alphabetical assignment for numerics. Instead of $\langle n1 \rangle$, $\langle n2 \rangle$, we could easily use $\langle a \rangle$, $\langle b \rangle$; which, reduces the chance of prediction error by 25%. In addition, better and more accurate replacement techniques exist which could further improve our work, but for our initiative, we wish to focus less on algorithm improvement and more on neural performance. In an ideal application of a MWP solver, no pre-processing would be necessary, but our approach significantly changed the focus of the system from rare occuring numbers to important language cues found in problem text.

Table 2: Test results for Experiment 2 (* denotes averages on present values only).

(Type) Model	AI2	CC	IL	MAWPS	Average
Hosseini et al., 2014	77.7	–	–	–	*77.7
Kushman et al., 2014	64.0	73.7	2.3	–	*46.7
Roy, Vieira, and Roth, 2015	–	–	52.7	–	*52.7
Robaidek et al. 2018	–	–	–	62.8	*62.8
Wang et al., 2018 (MathDQN)	**78.5**	75.5	73.3	–	*75.4
Pre-trained					
(1) Prefix-Transformer	70.2	91.1	95.2	82.4	84.7
(1) Postfix-Transformer	66.7	90.0	92.9	82.7	83.1
(1) Infix-Transformer	70.2	93.3	**96.4**	82.4	85.6
(2) Prefix-Transformer	66.7	91.1	**96.4**	82.1	84.1
(2) Postfix-Transformer	68.4	93.3	94.1	82.4	84.6
(2) Infix-Transformer	70.2	94.4	94.1	84.4	85.8
(3) Prefix-Transformer	68.4	91.1	95.2	82.4	84.3
(3) Postfix-Transformer	66.7	92.2	94.1	82.1	83.8
(3) Infix-Transformer	68.4	93.3	95.2	84.1	85.2
Non-pre-trained					
(1) Prefix-Transformer	71.9	94.4	95.2	83.4	86.3
(1) Postfix-Transformer	63.2	81.1	92.9	75.7	78.2
(1) Infix-Transformer	68.4	**97.8**	92.9	83.4	85.6
(2) Prefix-Transformer	73.7	94.4	94.1	**84.7**	**86.7**
(2) Postfix-Transformer	68.4	94.4	94.1	83.1	85.0
(2) Infix-Transformer	68.4	95.6	94.1	83.1	85.3
(3) Prefix-Transformer	73.7	93.3	95.2	84.1	86.6
(3) Postfix-Transformer	68.4	94.4	94.1	82.4	84.8
(3) Infix-Transformer	66.7	95.6	94.1	81.7	84.5

Figure 5: Failure of large Transformer.

Question (IL)
There are 4 cards. 3 cards more are added. How many are there total?
Correct Translation (Infix)
4 + 3
Hypothesized Translation
⟨⟨⟨⟨⟨ ... ⟨⟨⟨⟨⟨

Others have had great success with large Transformer architectures, especially when tasked with translation. We observe problems that result in an unusable arrangement when attempting networks of large sizes. Figure 5 demonstrates issues of attempted large networks. This occurance was common among pre-trained and non-pre-trained models of various dimensions above 6 Transformer layers.

Conclusions and Future Work

We show that alternative math representations provide more stability in automatic solvers. Use of Transformer networks improves automatic math word problem solving. We make improvements through very accessible means with thoughtful data and training of general natural language. Automatically solving math word problems will undoubtedly be very useful for entities such as question answering services. Transformers with unambiguous intermediate representations produce state-of-the-art arithmetic word problem translations.

Datasets such as Dolphin18k Huang et al., 2016, consisting of web-answered questions from Yahoo! Answers, require more variety of questions to be understood by the system. We hope to expand to step-based calculations, more complex concepts such as probabilities or calculus, and push the limits of machine understanding of classical mathematics.

Extensive pre-training over a large corpora of language has extended the capabilities of many neural approaches. Networks like BERT Devlin et al., 2018, trained extensively on data from Wikipedia, perform relatively better in many contexts. Alternatively, creating intentional bias toward words which determine a specific operation (i.e., +, *, -, /) could increase the connection between natural language and mathematics.

With a hope to further advance this area of research and heighten interests, all of the code and data used is available on GitHub.

Acknowledgement

The National Science Foundation supports the work reported in this paper under Grant No. 1659788. Any opinions, findings any conclusions or recommendations expressed in this work are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Bakman, Y. 2007. Robust understanding of word problems with extraneous information. *arXiv preprint math/0701393*.

Bobrow, D. G. 1964. *Natural language input for a computer problem solving system*. Ph.D. Dissertation, Massachusetts Institute Of Technology.

Budzianowski, P., and Vulić, I. 2019. Hello, it's gpt-2–how can i help you? towards the use of pretrained language models for task-oriented dialogue systems. *arXiv preprint arXiv:1907.05774*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Hosseini, M. J.; Hajishirzi, H.; Etzioni, O.; and Kushman, N. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 523–533.

Huang, D.; Shi, S.; Lin, C.-Y.; Yin, J.; and Ma, W.-Y. 2016. How well do computers solve math word problems? large-scale dataset construction and evaluation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, 887–896.

Huang, D.; Liu, J.; Lin, C.-Y.; and Yin, J. 2018. Neural math word problem solver with reinforcement learning. In *Proceedings of the 27th International Conference on Computational Linguistics*, 213–223. Santa Fe, New Mexico, USA: Association for Computational Linguistics.

Koncel-Kedziorski, R.; Hajishirzi, H.; Sabharwal, A.; Etzioni, O.; and Ang, S. D. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics* 3:585–597.

Koncel-Kedziorski, R.; Roy, S.; Amini, A.; Kushman, N.; and Hajishirzi, H. 2016. Mawps: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1152–1157.

Kushman, N.; Artzi, Y.; Zettlemoyer, L.; and Barzilay, R. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, 271–281.

Liguda, C., and Pfeiffer, T. 2012. Modeling math word problems with augmented semantic networks. In *International Conference on Application of Natural Language to Information Systems*, 247–252. Springer.

Maas, A. L.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; and Potts, C. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 142–150. Portland, Oregon, USA: Association for Computational Linguistics.

Mitra, A., and Baral, C. 2016. Learning to use formulas to solve simple arithmetic problems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, 2144–2153.

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Robaidek, B.; Koncel-Kedziorski, R.; and Hajishirzi, H. 2018. Data-driven methods for solving algebra word problems. *arXiv preprint arXiv:1804.10718*.

Roy, S., and Roth, D. 2016. Solving general arithmetic word problems. *arXiv preprint arXiv:1608.01413*.

Roy, S., and Roth, D. 2017. Unit dependency graph and its application to arithmetic word problem solving. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Roy, S.; Vieira, T.; and Roth, D. 2015. Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics* 3:1–13.

Shi, S.; Wang, Y.; Lin, C.-Y.; Liu, X.; and Rui, Y. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1132–1142.

Sun, R.; Zhao, Y.; Zhang, Q.; Ding, K.; Wang, S.; and Wei, C. 2019. A neural semantic parser for math problems incorporating multi-sentence information. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)* 18(4):37.

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*, 5998–6008.

Wang, L.; Zhang, D.; Gao, L.; Song, J.; Guo, L.; and Shen, H. T. 2018. Mathdqn: Solving arithmetic word problems via deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Adversarial Analysis of Natural Language Inference Systems

Tiffany Chien
University of California, Berkeley

Jugal Kalita
University of Colorado, Colorado Springs

Abstract

The release of large natural language inference (NLI) datasets like SNLI and MNLI have led to rapid development and improvement of completely neural systems for the task. Most recently, heavily pre-trained, Transformer-based models like BERT and MT-DNN have reached near-human performance on these datasets. However, these standard datasets have been shown to contain many annotation artifacts: features that correlate strongly with the correct label in training (and testing), but are clearly non-generalizable (e.g. sentence length, word overlap). This allows models to shortcut understanding using simple fallible heuristics, and still perform well on the test set. So it is no surprise that many adversarial (challenge) datasets have been created that cause models trained on standard datasets to fail dramatically. Although extra training on this data generally improves model performance on just that type of data, transferring that learning to unseen examples is still partial at best. This work evaluates the failures of state-of-the-art models on existing adversarial datasets that test different linguistic phenomena, and find that even though the models perform similarly on MNLI, they differ greatly in their robustness to these attacks. In particular, we find syntax-related attacks to be particularly effective across all models, so we provide a fine-grained analysis and comparison of model performance on those examples. We draw conclusions about the value of model size and multi-task learning (beyond comparing their standard test set performance), and provide suggestions for more effective training data.

Introduction

In recent years, deep learning models have achieved and continued to improve on state-of-the-art results on many NLP tasks. However, models that perform extremely well on standard datasets have been shown to be rather brittle and easily tricked. In particular, the idea of *adversarial* examples or attacks was brought over from computer vision, and various methods of slightly perturbing inputs have been developed that cause models to fail catastrophically (McCoy, Pavlick, and Linzen 2019; Glockner, Shwartz, and Goldberg 2018; Naik et al. 2018).

Adversarial attacks need to be studied from a security perspective for the deployment of real-world systems, but they are also a powerful lens into *interpretability* of black-box deep learning systems. By examining the failures of state-of-the-art models, we can learn a lot about what they are really learning, which may give us insights into improving their robustness and general performance.

One philosophical generalization about the cause of failure for all current NLP systems is a lack of deep, 'real' understanding of language. We will focus on the task of natural language inference (NLI), which is a basic natural language understanding task thought to be a key stepping stone to higher-level understanding tasks like question answering and summarization. The setup of the NLI task is to determine whether a *hypothesis* is true given a *premise*, answering *entailment*, *contradiction*, or *neutral*.

The current top-performing systems for NLI rely on pre-training on generic tasks, followed by fine-tuning on a labeled task-specific dataset. This is in contrast to older (before late 2018) models, which were primarily task-specific architectures trained primarily on task-specific labeled datasets. In addition, the Transformer architecture (Vaswani et al. 2017) now outperforms the previously dominating recurrent architectures (LSTM and variants). We want to analyze what kinds of adversarial attacks are still potent on highly-acclaimed recent NLP models like BERT (Devlin et al. 2018) and MT-DNN (Liu et al. 2019).

Our contributions are as follows:

- We test models on a variety of existing adversarial datasets, with a high level of granularity to different linguistic phenomena. Results indicate that the pre-trained models are remarkably good at lexical meaning, while struggling most with logic and syntactic phenomena.
- We focus in on the syntax-focused dataset created by McCoy, Pavlick, and Linzen. We look closely at the 30 subcases, and analyze the effects of model size (base vs. large size) and multi-task learning (MT-DNN vs. BERT). We also examine what subcases all models fail at.
- We experiment with fine-tuning the models with (flattened) dependency parses as input (with no adjustments to architecture or data pre-processing). We find that this does improves performance on some, but not all, subcases that rely on the hierarchical structure of sentences.
- Lastly, we investigate MNLI's biases by analyzing performance after different amounts of fine-tuning (more and more overfitting) on MNLI.

Related Work

This work joins a growing movement in NLP to go beyond improving test set metrics to more deeply analyze model learning and performance (Belinkov and Glass 2019). This genre of work believes in the value of interpretability, both to build safer practical systems, and just to find fruitful directions for improving raw model performance.

Liu, Schwartz, and Smith (2019) use a metaphor of inoculation to disentangle the blame for adversarial vulnerability between training data and model architecture. They expose a small part of the challenge dataset to the model during training, and re-test its evaluation performance on the original test set and the challenge dataset.

1. If the model still fails the challenge dataset, the weakness probably lies in its design/architecture or training process.

2. If the model can now succeed at the challenge dataset (without sacrificing performance on the original dataset), then the original dataset is at fault.

3. If the model does better on the challenge dataset but worse on the original dataset, the challenge dataset is somehow not representative of the phenomenon it was trying to test, for example having annotation artifacts or being very skewed to a particular label.

Unfortunately, even if adversarial training does improve model performance on that particular dataset, it is fundamentally impossible to devise and train on all possible linguistic phenomena. The transferability of adversarial robustness to new kinds of examples has been tested by some of the creators of adversarial datasets, by withholding some example generation methods while training on others. Nie, Wang, and Bansal (2018) find that knowledge of each of their rule-based templates was almost completely non-transferable to others. In fact, training on some specific templates caused overfitting and hurt overall robustness. McCoy, Pavlick, and Linzen (2019) find more mixed results, with some cases of successful transfer.

Many standard datasets for different tasks have been shown to have blatant annotation artifacts, allowing models to learn features that are strong in the training (and testing) data, but that have nothing to do with actually performing the task. Gururangan et al. (2018) find many of these artifacts in standard NLI datasets (SNLI and MNLI). For example, *neutral* hypotheses tend to be longer in length, because an easy way to generate a hypothesis that isn't necessarily entailed by the premise is to add extra details. Meanwhile, strong negation words like *nobody, no, never* are strong indicators of *contradiction*. With these artifacts in mind, they split the data into "hard" and "easy" versions, and model performance decreased by about 15% on the hard test set. These findings suggest that it is not the models' faults for failing on adversarial examples, given that there exist easier ways to get high accuracy than truly understanding anything. But it also means that current evaluation metrics greatly overestimate models' abilities and understanding.

Models

The two new models that we study gain most of their power from pre-training on a generic language task with a huge unlabeled dataset. They achieve state-of-the-art performance on a variety of language understanding tasks.

1. **BERT** (Devlin et al. 2018) pre-trains on a bidirectional word-masking language modelling task, in addition to sentence pair prediction, i.e. whether the second sentence is likely to directly follow the first.

2. **MT-DNN** (Liu et al. 2019) builds on BERT by performing multi-task learning on the nine GLUE (General Language Understanding Evaluation) benchmark tasks (Wang et al. 2018), after BERT's pre-training.

BERT is based on the Transformer architecture (Vaswani et al. 2017), a non-recurrent, purely attention-based architecture. BERT has a base version (12 Transformer layers), and a large version (24 layers). We trained base and large versions of both BERT and MT-DNN. These models are fine-tuned on MNLI starting from publicly available pre-trained checkpoints.

We compare with an older recurrent model, ESIM (Enhanced Sequential Inference Model) (Chen et al. 2016). It is NLI-task-specific and only trained on MNLI, with no huge pre-training. It uses a bidirectional LSTM to encode the premise and hypothesis sentences, and uses attention across those representations.

We also considered another model, Syntactic TreeLSTM (**S-TLSTM**), which is identical to ESIM except it uses a TreeLSTM that takes a dependency parse as input (Chen et al. 2016). This model may provide a useful comparison to BERT because its explicit use of the hierarchical structure of language is the exact opposite model design direction from extensive unsupervised pre-training. However, various studies suggest that the BERT architecture does in fact learn hierarchical structure: Goldberg (2019) found that BERT performed remarkably well when fine-tuned for external syntactic classification tasks, and Jawahar, Sagot, and Seddah (2019) showed that different layers of BERT learned structural representations of language at different abstraction levels. McCoy, Pavlick, and Linzen (2019) test a different tree-based model (SPINN (Bowman et al. 2016)) on their adversarial dataset, and find that it outperforms ESIM, but not BERT. Considering all this, and the fact that there is currently no tree-based model that comes close to outperforming BERT and variants on standard datasets, we decided not to test S-TLSTM, despite its philosophical appeal.

Overall Results and Analysis

First, for reference, we provide the accuracies on the matched MNLI dev set for the models we trained (and tested) in Table 1. BERT-large results do not quite match published results, but we had limited hardware and did not carefully tune hyperparameters. The BERT-based models all perform comparably, and even ESIM does respectably.

Let us now analyze the performance of the selected models on the adversarial datasets (also called challenge sets, stress tests). We discuss the first two briefly and then focus

Model	Accuracy (%)
ESIM	76.80
BERT base	84.17
BERT large	**85.84**
MT-DNN base	84.20
MT-DNN large	**86.69**

Table 1: Overall MNLI Results

on the last one (McCoy, Pavlick, and Linzen 2019) because it is the most interesting in terms of actually distinguishing the strengths of the better-performing models.

Glockner, Shwartz, and Goldberg (2018) This dataset is created by modifying SNLI examples with single word replacements of different lexical relations, based on WordNet. It tests lexical inferences and relatively simple world knowledge. They test a model called KIM (Knowledge-based Inference Model) (Chen et al. 2016), which builds on ESIM to explicitly incorporate knowledge from WordNet in a variety of ways, including in architecture additions. However, the BERT-based models still significantly outperform KIM. This could be due to model architecture, but is most likely a result of their extensive pretraining on a huge diverse corpus. There is not a big difference between model sizes, or between MT-DNN and BERT. This suggests that lexical semantics is more basic and low-level, so learning it does not need so many layers of abstraction, or multi-task learning (see Table 2).

Model	Accuracy (%)
ESIM*	65.6
KIM*	83.5
BERT base	92.2
BERT large	**94.2**
MT-DNN base	92.9
MT-DNN large	**94.8**

Table 2: Single Word Replacement Attacks from (Glockner, Shwartz, and Goldberg 2018). ESIM and KIM results from original paper.

Naik et al. (2018) This dataset is composed of a variety of tests motivated by a manual examination and categorization of 100 mistakes made by the best performing model at the time (Nie and Bansal 2017). The categories are antonyms, word overlap (append "and true is true"), negation words (append "and false is not true"), length mismatch (append "and true is true" 5 times), and spelling errors. Antonyms and Spelling are "competence" tests, while the rest are "distraction" tests. The examples are generated by modifying examples from MNLI. We report accuracy averaged over all categories in Table .

$\mathrm{BERT}_{\mathrm{large}}$ and $\mathrm{MT\text{-}DNN}_{\mathrm{large}}$ do best. Overall model performance trends the same as performance on MNLI, but

Model	Accuracy (%)
ESIM	68.39
BERT base	74.30
BERT large	**77.21**
MT-DNN base	73.73
MT-DNN large	**77.14**

differences are not huge. Furthermore, when we examined performance on specific categories, all models had about the same pattern of relative performance on different categories of tests, i.e. they have the same relative successes and failures. This consistency and generally similar performance indicates in this case that the dataset is not well-targeted enough for really interesting insight. In addition, compared to McCoy, Pavlick, and Linzen (2019) (below), the way that examples are generated is more artificial, and maybe less meaningful. Of course, a robust NLI system still should not be defeated by this kind of attack, i.e. be able to determine irrelevant information, including tautologies, and this test shows that even the best models do not have this capability mastered properly.

McCoy, Pavlick, and Linzen (2019) They hypothesize that models utilize shallow, fallible syntactic heuristics to achieve accuracy on MNLI, instead of "real" understanding. The dataset consists of examples generated from manually created templates that break these heuristics. They have three categories of heuristics (each is a special case of the one before).

1. Lexical overlap: Model is likely to answer *entailment* if the premise and hypothesis share a lot of words.
   Would trick bag-of-words (no word order) models.
2. Subsequence: The hypothesis is a contiguous string of words from the premise.
   The ball by the bed rolled. $\nrightarrow$ *The bed rolled.*
   Could confuse sequence models too.
3. Constituent: The hypothesis is a syntactic constituent in the premise.
   If the boys slept, they would not eat. $\nrightarrow$ *The boys slept.*
   Could confuse models that know about syntax.

All three heuristics involve the model thinking the answer is *entailment* when it is not, i.e. the *non-entailment* examples are the ones that contradict the heuristic. So the extreme imbalance in model performance between entailment and non-entailment examples is strong evidence that the models do indeed rely on the hypothesized heuristics (Table 3 vs. 4).

Entailment	word overlap	subseq	constituent
ESIM	96.52	98.46	94.48
$\mathrm{BERT}_{\mathrm{base}}$	97.20	99.52	99.04
$\mathrm{BERT}_{\mathrm{large}}$	**90.48**	99.48	96.70
$\mathrm{MT\text{-}DNN}_{\mathrm{base}}$	97.22	99.98	99.22
$\mathrm{MT\text{-}DNN}_{\mathrm{large}}$	96.06	99.54	99.14

Table 3: Accuracy on examples labeled 'entailment'

Non-entailment	word overlap	subseq	constituent
ESIM	1.56	4.88	3.32
$\text{BERT}_{\text{base}}$	54.68	9.46	4.88
$\text{BERT}_{\text{large}}$	**83.44**	**31.38**	**44.72**
$\text{MT-DNN}_{\text{base}}$	72.96	5.66	16.50
$\text{MT-DNN}_{\text{large}}$	**88.08**	**31.24**	22.88

Table 4: Accuracy on examples labeled 'non-entailment'

All the BERT-based models do significantly better than the LSTM-based ESIM in most categories, as we see in Table 4. But $\text{BERT}_{\text{large}}$ and $\text{MT-DNN}_{\text{large}}$ do vastly better than all others, a difference that was not nearly as apparent in any of the other datasets we tested. In combination with the granularity in the manually created templates, these huge differences in performance indicate that this dataset more directly probes and reveals the strengths and weaknesses of different models.

The success of $\text{BERT}_{\text{large}}$ and $\text{MT-DNN}_{\text{large}}$ suggests that structural/syntactic information is learned more deeply by a larger model with more layers and parameters to work with (in contrast to lexical semantics (Glockner, Shwartz, and Goldberg, above)). $\text{BERT}_{\text{large}}$ also has lower accuracy on the *entailment* examples, also indicating that it is less prone to blindly following the heuristics.

$\text{MT-DNN}_{\text{base}}$ (which is built on $\text{BERT}_{\text{base}}$ and is therefore of comparable size) does significantly better than $\text{BERT}_{\text{base}}$ in some categories, indicating the value of multi-task learning (specifically on language understanding tasks).

Fine-grained Model Comparison

Comparison of $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$

$\text{BERT}_{\text{large}}$ performs better than or equal to $\text{BERT}_{\text{base}}$ (at worst -1%) on all fifteen *non-entailment* subcases. Some templates saw particularly large improvement, such as modifying clauses:

- Relative clauses that modify nouns (+42.4%)
  The artists that supported the senators shouted. $\nrightarrow$ *The senators shouted.*
- Prepositional phrase modifiers (+38%)
  The managers next to the professors performed. $\nrightarrow$ *The professors performed.*

Understanding modifying clauses requires understanding the mechanics of compositional semantics (probably utilizing some kind of hierarchical syntax), which is a basic but crucial step in language understanding. So BERT-large's performance over BERT-base on these examples is evidence of significantly deeper understanding.

Another area of improvement is the lexical meanings of special subclasses of verbs and adverbs.

- Non-truth verbs with clause complements (+60.4%)
  The tourists said that the lawyer saw the secretary. $\nrightarrow$ *The lawyer saw the secretary.*
  This template uses a variety of verbs, all of which suggest but do not entail their complements.
- Modal adverbs (+26.7%)
  Maybe the scientist admired the lawyers. $\nrightarrow$ *The scientist admired the lawyers.*

Similarly, passive voice is a special *syntactic* phenomenon that BERT-large improves on, but still has trouble with.

- Passive voice (3.6% $\rightarrow$ 29.8%)
  The managers were advised by the athlete. $\nrightarrow$ *The managers advised the athlete.*

$\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$ were trained (pre-training and fine-tuning) on the same data, so the difference in the richness of their learning must reside only in the doubled number of layers in $\text{BERT}_{\text{large}}$. These performance improvements are evidence that more layers is necessary space for learning all the different special cases of language.

There are also some partially learned special cases, such as the meaning of "if" and related (logical implication).

- 76.6% $\rightarrow$ 98.7%: *Unless the professor danced, the student waited.* $\nrightarrow$ *The professor danced.*
- both 0%: *Unless the bankers called the professor, the lawyers shouted.* $\nrightarrow$ *The lawyers shouted.*

Meanwhile, all models fail to understand the logical meaning of disjunction (0-2%).

- *The actor helped the lawyers, or the managers stopped the author.* $\nrightarrow$ *The actor helped the lawyers.*

Logic is a very important component of inference as an understanding task, but understandably difficult for statistical models to learn properly, because it is in some sense not probabilistic, in addition to being dependent on exact meanings of single function words. Many traditional inference systems relied primarily on formal logic machinery, and finding a way to incorporate that into new models seems like a promising direction. Designing and training neural networks that parse and understand formal, symbolic logic is a pretty well-studied problem (Evans et al. 2018), and it is certainly known theoretically that general neural networks can represent arbitrary nonlinear logical relations. The difficulty is getting natural language models to actually care enough about logic during training to use it correctly for a specific task. Many different approaches have been explored recently, including but not limited to modifying the loss function to encourage logical consistency (Minervini and Riedel 2018), rule distillation in a teacher-student network (Hu et al. 2016), and indirect supervision using probabilitic logic (Wang and Poon 2018). To our knowledge, these have not yet been incorporated into state-of-the-art models, but they show promising results on the baseline models tested, especially in lower-resource scenarios.

All of these special cases are almost certainly encountered in BERT's huge pre-training corpus, but that unsupervised stage does not necessarily teach the model how to use that information towards performing inference. This is why larger and larger pre-training may not be the most effective or at least efficient way to achieve language understanding.

Some of the subsequence templates are still a struggle for all models, including large BERT and MT-DNN ($<$10%):

- *The manager knew the athlete mentioned the actor* $\nrightarrow$ *The manager knew the athlete.*

Heuristic	Syntactic subcategory	MT-DNN large	BERT large	MT-DNN base	BERT base	ESIM	BERT large UP	MT-DNN base PO
Lexical Overlap	subject/object_swap	**0.999**	**0.994**	0.935	0.729	0	0.988	0.936
	preposition	0.934	**0.979**	0.794	0.745	0.004	0.960	0.889
	relative_clause	0.912	**0.928**	0.699	0.504	0.069	**0.930**	0.837
	passive	**0.625**	0.298	0.432	0.036	0	0.214	0.505
	conjunction	0.934	**0.973**	0.788	0.720	0.005	0.943	0.711
Subseq	NP/S	0.042	0.003	0	0.016	0.058	0.004	0.003
	PP_on_subject	0.668	0.673	0.168	0.293	0.001	**0.786**	0.533
	relative_clause_on_subject	0.749	0.698	0.082	0.133	0.087	**0.863**	0.347
	past_participle	0.006	0.049	0.013	0.018	0.050	0.032	0.008
	NP/Z	0.097	0.146	0.020	0.013	0.047	**0.217**	0.172
Constituent	embedded_under_if	0.703	**0.987**	0.369	0.767	0.137	0.907	0.387
	after_if_clause	0.001	0	0	0	0	0	0.010
	embedded_under_verb	0.342	**0.903**	0.252	0.299	0	0.546	0.146
	disjunction	0.005	0	0.001	0.001	0.029	0.008	0.002
	adverb	0.093	**0.346**	0.203	0.079	0	0.083	0.036

Table 5: Results for *non-entailment* subcases. Each row corresponds to a syntactic phenomenon. BERT large UP: trained on unparsed then parsed; MT DNN-base PO: trained on parsed only

- *When the students fought the secretary ran. ↛ The students fought the secretary.*

These templates are in the spirit of *garden path sentences*, where local syntactic ambiguity causes a sequential reading of a sentence to lead to an incorrect interpretation. This kind of sentence has been studied extensively in cognitive science, specifically language processing, as human readers are first misled and then must backtrack to reanalyze the composition of the sentence to understand it properly (Ferreira and Henderson 1991; Osterhout, Holcomb, and Swinney 1994). Goldberg (2019) shows that BERT performs well on complex subject-verb agreement tasks, even without any fine-tuning, indicating that the pre-trained model already has the ability to correctly parse this kind of sentence. So the model somehow knows about syntax but does not know how to use it towards the task of inference, a teaching failure that can only be blamed on the inference-task-specific fine-tuning. MNLI probably has a low occurrence of complex syntax, but perhaps more importantly, the complete syntactic information is rarely necessary to perform the task. Nevertheless, an ability to utilize challenging syntax is an important generalizable skill, because it indicates deep, principled understanding of language.

Comparison of BERT and MT-DNN

Even though MT-DNN$_{large}$ performs better on MNLI than BERT$_{\text{large}}$, BERT beats MT-DNN on more subcases in this dataset. In particular, MT-DNN$_{large}$ struggles much more with subcases that test special lexical meanings that prevent entailment (number is difference between MT-DNN$_{large}$ and BERT$_{\text{large}}$):

1. conditionals: if, unless, whether or not (28.4%)
2. 'belief' verbs: believed, thought, hoped (56.1%)
3. uncertainty adverbs: hopefully, maybe, probably (25.3%)

The only subcase that MT-DNN$_{large}$ is significantly better at is the passive voice (+32.7%).

MT-DNN is trained starting with a pre-trained BERT and then fine-tuning on the 9 language understanding tasks in the GLUE benchmark (before fine-tuning again on MNLI). So if MT-DNN performs worse than a BERT model of the same size, this fine-tuning caused it to *forget* some knowledge that it had before. This would happen if the datasets being fine-tuned on do not explicitly test that knowledge, teaching the model to care less about the information from these words. Considering that most of the GLUE tasks are not straight NLI tasks, it is somewhat unsurprising that the model forgot how these words affect entailment.

Parses as Input

Considering that syntactic phenomena are one of the models' weaknesses, we conduct an experiment of simply passing the flattened binary parses as the input "sentences". We use the automatically generated parses that come with MNLI and the adversarial dataset. We test on the dataset from McCoy, Pavlick, and Linzen (2019).
We try two fine-tuning regimens:

1. Fine tune on original (unparsed) MNLI, then fine-tune again on the same data, parsed (labeled UP in Table 5).
2. Only fine-tune on parsed MNLI (no other inference-specific fine-tuning) (labeled PO in Table 5).

We find that it is rather difficult to get the different models to train well. Some had loss that never converged, some got near 0% on all *non-entailment* subcases. The only reasonable parsed models are BERT$_{\text{large}}$ under the first regimen (UP), and MT-DNN$_{\text{base}}$ under the second (PO). It is likely that these difficulties could be overcome with some systematic hyperparameter tuning, but we see substantial consistency (in model performance on the adversarial dataset) between the two successes, so do not think it would be very in-

Type	Sentence 1	Sentence 2
NP/S	The manager knew the tourists supported the author.	The manager knew the tourists.
NP/Z	Since the judge stopped the author contacted the managers.	The judge stopped the author.
past_participle	The scientist presented in the school stopped the artists.	The scientist presented in the school.
after_if_clause	Unless the scientists introduced the presidents, the athletes recommended the senator.	The athletes recommended the senator.

Table 6: Non-entailed cases where $\mathrm{BERT}_{\mathrm{large}}$ does very poorly: Sentence 1 does not entail Sentence 2.

sightful to test more. But the fact that the models responded so differently to fine-tuning suggests that the models have significantly different 'knowledge states' in terms of what they learned about how to solve tasks, i.e. they ended up in different local optima after pre-training. This idea deserves more analysis, because the whole point of huge pre-training is to learn maximally transferable and general representations of language. Thus, how to guide models towards these ideal local optima (and away from overfitting) is a very important and difficult question.

The fact that any model is able to learn what to do with parses is already surprising, given that none of their pre-training is parsed. Evaluating on the parses of MNLI (matched dev set), $\mathrm{BERT}_{\mathrm{large}}$ achieves 82% accuracy (compare to 86% unparsed), and $\mathrm{MT\text{-}DNN}_{\mathrm{base}}$ gets 84% (equal to unparsed).

These are the six subcases that saw a 10% or greater change in accuracy between parsed and unparsed inputs. Numbers are percent change from unparsed to parsed ($\mathrm{BERT}_{\mathrm{large}}$, $\mathrm{MT\text{-}DNN}_{\mathrm{base}}$).
Parsing does better on:

- Modifiers on subject
  The managers next to the professors performed. $\nrightarrow$ *The professors performed.* (+11.3, +36.5)
  The artists that supported the senators shouted. $\nrightarrow$ *The senators shouted.* (+16.5, +26.5)
- NP/Z (+7.1, +15.2)
  Since the athlete hid the secretaries introduced the president. $\nrightarrow$ *The athlete hid the secretaries.*
  The parsed models still only achieve 21.7% and 17.2% accuracy, but this is still some improvement.
- Conjunction (+22.2, +1.8 (unparsed $\mathrm{MT\text{-}DNN}_{\mathrm{base}}$ already gets 90.8))
  The tourists and senators admired the athletes $\rightarrow$ *The tourists admired the athletes.*
  This is an *entailment* template, so $\mathrm{BERT}_{\mathrm{large}}$'s lower accuracy actually indicates less heuristic reliance, and parsed improvement from 64.4 $\rightarrow$ 86.6 really indicates better understanding (while $\mathrm{MT\text{-}DNN}_{\mathrm{base}}$'s performance could just be using the heuristic).

Parsing does worse on:

- Embedded clause under non-truth verb (-35.7, -10.6)
  The lawyers believed that the tourists shouted. $\nrightarrow$ *The tourists shouted.*
- Adverbs indicating uncertainty (-26.3, -16.7)
  Hopefully the presidents introduced the doctors $\nrightarrow$ *The presidents introduced the doctors.*

Of this small set of significant changes, it can be said that the parsed inputs helped the model with syntactic, hierarchical examples, and hurt it on specific lexical semantics. This is a surprisingly intuitive result: the model shifted its focus more to syntax!

However, these are the only subcases that changed significantly, out of 30, suggesting either that the parses don't encode that much useful information, or (more likely) that the fine-tuning didn't teach the model how to use the extra information. For example, maybe $\mathrm{BERT}_{\mathrm{large}}$ (trained on unparsed then the exact same data parsed) just learned to ignore parentheses.

Furthermore, the subcases which had score close to 0 for the unparsed model basically did not see any improvement. These obstinate cases are given in Table 6. Most of these cases are tests of syntactic phenomena, so parsed data certainly contains useful information, but again, the fine-tuning is somehow not enough to teach the model how to use it.

We do not think that parsing is necessarily a preprocessing step that should be incorporated into future models/systems, because it takes extra computational and annotated data resources. But this experiment does show that without induced biases, BERT's massive, generic pre-training does not capture some basic rule-like principles.

Overfitting to MNLI

Models learn and use fallible heuristics only because it works on their training datasets; in other words, they are overfit to their training data, MNLI. We analyze this process by evaluating the model after different amounts of fine-tuning on MNLI. We perform this experiment on $\mathrm{MT\text{-}DNN}_{large}$, the best performer on MNLI, and gauge overfitting by evaluating on the adversarial dataset from McCoy, Pavlick, and Linzen (non-entailment subcases).

Epoch #	1	2	3
matched MNLI dev set	85.66	86.69	86.59
McCoy *non-entailment*	44.09	47.40	42.49

Table 7: Accuracy (%) for $\mathrm{MT\text{-}DNN}_{large}$ fine-tuned on MNLI for varying numbers of epochs, and then evaluated on the dataset from McCoy, Pavlick, and Linzen.

The model trains very quickly, reaching 1% away from max dev accuracy after only one epoch of fine-tuning, and decreasing slightly on dev accuracy by the third epoch. This is a claimed benefit of multi-task learning: the model is more flexible to learning different tasks quickly.

From epoch 2 to 3, MNLI dev performance decreases by only 0.1%, but according to performance on the adversarial dataset, the model is relying significantly more on heuristics, revealing a more overfit state. Looking at specific subcases, the epoch-3 model differs by more than 10% in 6 subcases, split very similarly to what happened with parsed inputs:

- Improves at lexical semantics: 'belief' verbs (believed, thought) (+11.8%) and uncertainty adverbs (hopefully, maybe) (+24.3%)
- Gets worse at structural/syntactic phenomena: passive voice (-24.4%), conjunction (-12.4%), and subject modifiers (PP (-15.6%), relative clauses (-19.1%))

Interestingly, the subcases that more MNLI fine-tuning helps are exactly the same as the ones that $\mathrm{BERT}_{\mathrm{large}}$ beats $\mathrm{MT\text{-}DNN}_{large}$ on. This strongly suggests that the purpose of these words is emphasized in MNLI: MT-DNN forgets about it while fine-tuning on other GLUE tasks, and more fine-tuning on MNLI makes it re-learn it.

On the other hand, the subcases that more fine-tuning hurts are all structural/syntax-focused, indicating that MNLI is biased against actually utilizing complex syntactic phenomena in a way that affects entailment (supporting the *syntactic* heuristic hypothesis of McCoy, Pavlick, and Linzen).

Creating feasibly-sized training datasets with "no biases" is impossible. Here we find some subtle examples in MNLI, emphasizing the sensitivity of these models to pick up on any useful signal. NLI is a very broad task, making it hard to define what a natural or representative input distribution would be, so dataset should depend on desired abilities and applications.

Conclusion

In this work, we use adversarial and challenge datasets to probe and analyze the failures of current state-of-the-art natural language inference models, comparing BERT and MT-DNN models of different sizes. Evaluating on these datasets distinguishes the actual understanding capabilities of the different models better than simply their performance on MNLI (the large dataset they were trained on). Our analysis is very fine-grained, targeting many specific linguistic phenomena. We find various improvements from larger model size and multi-task learning. We find that the most difficult examples for the best models are logic or syntax-based, including propositional logic and garden-path sentences. We experiment with passing parses as input to the out-of-the-box pre-trained models, and find that it does provide some improvement in examples that require understanding syntax, demonstrating the value of syntactic induced biases. We analyze what overfitting to MNLI looks like, and reveal some biases/artifacts in the dataset.

Some may argue that testing NLI systems on artificially challenging datasets is unfair and not useful, because it is not representative of their performance on naturalistic, real-world data. But even if the data humans naturally produce is not so difficult (because humans also are lazy and use heuristics), the difference is that we always *can* parse sentences correctly, utilizing rules and principles. And we intuitively know that ability is crucial to robust, trustworthy, and *real* language understanding.

Acknowledgement

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

References

Belinkov, Y., and Glass, J. 2019. Analysis Methods in Neural Language Processing: A Survey. *Transactions of the Association for Computational Linguistics* 7:49–72.

Bowman, S. R.; Gauthier, J.; Rastogi, A.; Gupta, R.; Manning, C. D.; and Potts, C. 2016. A Fast Unified Model for Parsing and Sentence Understanding. *arXiv:1603.06021 [cs]*. arXiv: 1603.06021.

Chen, Q.; Zhu, X.; Ling, Z.; Wei, S.; Jiang, H.; and Inkpen, D. 2016. Enhanced LSTM for Natural Language Inference. *arXiv:1609.06038 [cs]*. arXiv: 1609.06038.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*. arXiv: 1810.04805.

Evans, R.; Saxton, D.; Amos, D.; Kohli, P.; and Grefenstette, E. 2018. Can Neural Networks Understand Logical Entailment? *arXiv:1802.08535 [cs]*. arXiv: 1802.08535.

Ferreira, F., and Henderson, J. M. 1991. Recovery from misanalyses of garden-path sentences. *Journal of Memory and Language* 30(6):725–745.

Glockner, M.; Shwartz, V.; and Goldberg, Y. 2018. Breaking NLI Systems with Sentences that Require Simple Lexical Inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 650–655. Melbourne, Australia: Association for Computational Linguistics.

Goldberg, Y. 2019. Assessing BERT's Syntactic Abilities. *arXiv:1901.05287 [cs]*. arXiv: 1901.05287.

Gururangan, S.; Swayamdipta, S.; Levy, O.; Schwartz, R.; Bowman, S. R.; and Smith, N. A. 2018. Annotation Artifacts in Natural Language Inference Data. *arXiv:1803.02324 [cs]*. arXiv: 1803.02324.

Hu, Z.; Ma, X.; Liu, Z.; Hovy, E.; and Xing, E. 2016. Harnessing Deep Neural Networks with Logic Rules. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2410–2420. Berlin, Germany: Association for Computational Linguistics.

Jawahar, G.; Sagot, B.; and Seddah, D. 2019. What Does BERT Learn about the Structure of Language? In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, 3651–3657. Florence, Italy: Association for Computational Linguistics.

Liu, X.; He, P.; Chen, W.; and Gao, J. 2019. Multi-Task Deep Neural Networks for Natural Language Understanding. *arXiv:1901.11504 [cs]*. arXiv: 1901.11504.

Liu, N. F.; Schwartz, R.; and Smith, N. A. 2019. Inoculation by Fine-Tuning: A Method for Analyzing Challenge Datasets. *arXiv:1904.02668 [cs]*. arXiv: 1904.02668.

McCoy, T.; Pavlick, E.; and Linzen, T. 2019. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, 3428–3448. Florence, Italy: Association for Computational Linguistics.

Minervini, P., and Riedel, S. 2018. Adversarially Regularising Neural NLI Models to Integrate Logical Background Knowledge. *arXiv:1808.08609 [cs, stat]*. arXiv: 1808.08609.

Naik, A.; Ravichander, A.; Sadeh, N.; Rose, C.; and Neubig, G. 2018. Stress Test Evaluation for Natural Language Inference. *arXiv:1806.00692 [cs]*. arXiv: 1806.00692.

Nie, Y., and Bansal, M. 2017. Shortcut-Stacked Sentence Encoders for Multi-Domain Inference. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP*, 41–45. Copenhagen, Denmark: Association for Computational Linguistics.

Nie, Y.; Wang, Y.; and Bansal, M. 2018. Analyzing Compositionality-Sensitivity of NLI Models. *arXiv:1811.07033 [cs]*. arXiv: 1811.07033.

Osterhout, L.; Holcomb, P. J.; and Swinney, D. A. 1994. Brain potentials elicited by garden-path sentences: Evidence of the application of verb information during parsing. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 20(4):786–803.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention Is All You Need. *arXiv:1706.03762 [cs]*. arXiv: 1706.03762.

Wang, H., and Poon, H. 2018. Deep Probabilistic Logic: A Unifying Framework for Indirect Supervision. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 1891–1902. Brussels, Belgium: Association for Computational Linguistics.

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. *arXiv:1804.07461 [cs]*. arXiv: 1804.07461.

Author Index