Proceedings of the Seminar

Machine Learning in Computer Vision and Natural Language Processing

University of Colorado, Colorado Springs

August 4, 2017

Editor: Jugal K. Kalita

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 2017 at the University of Colorado, Colorado Springs. Within a very short period of time, the students were able to choose cutting-edge projects involving machine learning in the areas of computer vision and natural language processing, write proposals, design interesting algorithms and approaches, develop code, and write papers describing their work. We hope that the students will continue working on these projects and submit papers to conferences and journals within the next few months. We also hope that it is the beginning of a fruitful career in research and innovation for all our participants.

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

Sincerely,

Jugal Kalita
jkalita@uccs.edu
Professor
August 4, 2017

Table of Contents

NSF REU Seminar on Machine Learning

Department of Computer Science
University of Colorado, Colorado Springs
Engineering 105
August 4, 2017: Friday

10:30-10:35 AM: Welcome Remarks by *Dr. Kelli Klebe, Associate Vice Chancellor for Research and Faculty Development, Dean of the Graduate School and Professor of Psychology, University of Colorado, Colorado Springs, CO*

10:35-11:50 AM *Session Chair: Lisa Jesse, Co-founder, Intelligent Software Solutions, Inc., Colorado Springs, CO*

10:35-11:00 *Sridhama Prakhya,* BML Munjal University, Gurgaon, India: Open-Set Deep Learning for Text Classification

11:00-11:25 *Derek S. Prijatelj*, Duquesne University, Pittsburgh, PA: Textual Semantic Analysis using Differential Neural Computers

11:25-11:50 *Christopher Towne*, New College of Florida, Sarasota, FL: Computing Semantic Roles using ANNs with External Memory

11:50-12:45 PM: Lunch

12:45-2:25 PM *Session Chair: Dr. Ethan Rudd, University of Colorado, Colorado Springs, CO*

12:45-1:10 *Adly Templeton*, Williams College, Williamstown, MA: Extractive Summarization using Vector Semantics

1:10-1:35 *Ryan Gribenow*, University of Colorado, Colorado Springs, CO: Open Set Image Forgery Classification and Localization

1:35-2:00 *Seyed Masoumzadeh*, University of Colorado, Colorado Springs, CO: Learning to Detect and Classify Forgeries of Digital Images in Open-Set Recognition

2:00-2:25 *Harriet Small,* Brown University, Providence, RI: Handling Unbalanced Data in Deep Image Segmentation

2:25-2:35 PM: Break
2:35-3:50 PM *Session Chair: Dr. Robert Carlson, Professor, Mathematics, University of Colorado, Colorado Springs, CO*

2:35-3:00 *Kyle Yee,* Swarthmore College, Swarthmore, PA: Super-Resolving Fluorescent Proteins Using Convolutional Neural Networks

3:00-3:25 *Adia Meyers*, Clayton State University, Morrow, GA: Determining Gaussian Edge Potentials with Deep Encoders

3:25-3:50 *Diptodip Deb*, Georgia Institute of Technology, Atlanta, GA: Learning Perspective-free Counting using Dilated Convolutions

3:50 PM: Closing Remarks

Our Session Chairs and Guests

Dr. Robert Carlson is professor and chair of the Mathematics Department at the University of Colorado at Colorado Springs. Early in his career he worked on a variety of computer vision problems, both in the aerospace industry and at UCCS.

Lisa Jesse is a co-founder of Intelligent Software Solutions, an international software analytics company headquartered in Colorado Springs. She earned both a B.S. and M.S.in Computer Science at UCCS and has over 25 years of experience in applied research in Artificial Intelligence, data analysis and visualization.

Dr. Kelli Klebe is a quantitative psychologist who has been at UCCS for 27 years. As a faculty member she taught courses in statistics and research methods and her research areas are on the effectiveness of social interventions and exploring the best statistical methods for analyzing change. She is currently the dean of the graduate school and the associate vice chancellor for research.

Dr. Ethan Rudd graduated with a PhD in Computer Science from the University of Colorado, Colorado Springs in May 2017, working under Dr. Terry Boult on various machine learning topics. He is now a senior level data scientist at Sophos.

NSF REU Proposal Presentation Meeting

Department of Computer Science
University of Colorado, Colorado Springs
Engineering Building, Room 109
June 9, 2017: Friday

2:00-2:05 PM: Welcome Remarks by Christopher Nelson, Assistant Dean, College of Engineering and Applied Science

2:05-3:05 PM
Session Chair: Abigail Graese, Department of Computer Science, University of Colorado, Colorado Springs

Adly Templeton, Williams College, Williamstown, MA: <u>Extractive Summarization using Vector Semantics</u>

Christopher Towne, New College of Florida, Sarasota, FL: <u>Semantic Role Labeling with a Differentiable Neural Computer</u>

Derek S. Prijatelj, Duquesne University, Pittsbugh, PA: <u>Performance of Differential Computers in Semantics</u>

Sridhama Prakhya, BML Munjal University, Gurgaon, India: <u>Open Set Deep Learning for Text Classification</u>

3:15-4:00 PM
Session Chair: Steve Cruz, Department of Computer Science, University of Colorado, Colorado Springs

Ryan Gribenow, University of Colorado, Colorado Springs, CO: <u>Open Set Image Forgery Classification and Localization</u>

Seyed Masoumzadeh, University of Colorado, Colorado Springs, CO: <u>Learning to Detect and Classify Forgeries of Digital Images in Open-Set Recognition</u>

Diptodip Deb, Georgia Institute of Technology, Atlanta, GA: <u>Learning Perspective-free Counting</u>

4:15-5:00 PM
Session Chair: Dr. Jonathan Ventura, University of Colorado, Colorado Springs

Adia Meyers, Clayton State University, Morrow, GA: <u>Determining Gaussian Edge Potentials with Deep Encoders</u>

Harriet Small, Brown University, Providence, RI: <u>Handling Unbalanced Data in Deep Image Segmentation</u>

Kyle Yee, Swarthmore College, Swarthmore, PA: <u>Super-Resolving Fluorescent Proteins Using Convolutional Neural Networks</u>

Our Session Chairs

Abigail Grasse a rising senior at UCCS and she has been doing research focused on deep neural networks and computer vision in the VAST lab under Dr. Terrance Boult for about a year. She published her first paper titled, "Assessing Threat of Adversarial Examples on Deep Neural Networks" at the IEEE International Conference on Machine Learning Applications (ICMLA), Anaheim, California, and December 2016.

Steve Cruz graduated in May with a Bachelor of Innovation degree in Computer Security. He has published 2 papers this year, "Open Set Intrusion Recognition for Fine-Grained Attack Categorization" at the IEEE International Symposium on Technologies for Homeland Security (Waltham, MA) and "Towards Open-Set Face Recognition" at the Biometric Workshop at the Conference on Vision and Pattern Recognition (Honolulu, HI). Steve is now a graduate student with Dr. Terrance Boult, pursing a PhD in Engineering with Concentration in Security.

Dr. Jonathan Ventura is an assistant professor in the Department of Computer Science at the University of Colorado, Colorado Springs. His areas of expertise are computer vision, geometric problems such as 3D modeling and camera localization, medical image analysis, and mobile augmented reality. Dr. Ventura has a PhD from the University of California at Santa Barbara, and has published 30 papers. As an undergraduate, he was in an REU program himself at the UCSB.

NSF REU Midsummer Meeting

Department of Computer Science
University of Colorado, Colorado Springs
Engineering Building, Room 105
July 7, 2017: Friday

2:00-2:05 PM: Welcome Remarks by Dr. Xiaobo Zhou, Interim Dean, College of Engineering and Applied Science

2:05-2:45 PM
Session Chair: Dr. Manuel Gunther, Department of Computer Science, University of Colorado, Colorado Springs

Kyle Yee, Swarthmore College, Swarthmore, PA: Super-Resolving Fluorescent Proteins Using Convolutional Neural Networks

Harriet Small, Brown University, Providence, RI: Handling Unbalanced Data in Deep Image Segmentation

Adia Meyers, Clayton State University, Morrow, GA: Determining Gaussian Edge Potentials with Deep Encoders

3:00-3:40 PM
Session Chair: Dr. Abdullah Sheneamer, Department of Computer Science, University of Colorado, Colorado Springs

Diptodip Deb, Georgia Institute of Technology, Atlanta, GA: Learning Perspective-free Counting using Dilated Convolutions

Ryan Gribenow, University of Colorado, Colorado Springs, CO: Open Set Image Forgery Classification and Localization

Sridhama Prakhya, BML Munjal University, Gurgaon, India: Open-Set Deep Learning for Text Classification

4:00-4:45 PM
Session Chair: Dr. Terrance Boult, University of Colorado, Colorado Springs

Adly Templeton, Williams College, Williamstown, MA: Extractive Summarization using Vector Semantics

Christopher Towne, New College of Florida, Sarasota, FL: Computing Semantic Roles using ANNs with External Memory

Derek S. Prijatelj, Duquesne University, Pittsburgh, PA: Textual Semantic Analysis using Differential Neural Computers

Monday, July 10, 9 AM

Seyed Masoumzadeh, University of Colorado, Colorado Springs, CO: Learning to Detect and Classify Forgeries of Digital Images in Open-Set Recognition

Our Session Chairs

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

Dr. Abdullah Sheneamer received his PhD from the University of Colorado at Colorado Springs in 2017, working in the LINC Lab under the supervision of Dr. Jugal Kalita. His research interests include data mining, machine learning, and software engineering. In particular, he is interested in applying machine learning to detection of cloned and obfuscated code as well as detection of malware and plagiarism. Dr. Sheneamer teaches Computer Science at Jazan University, Saudi Arabia.

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 surveillance, biometrics, sensor networks, and distributed steganalaysis and general projects in computer vision. He also works with The Colorado Institute for Technology Transfer and Implementation (CITTI) through which he works with many local companies.

Open-Set Deep Learning for Text Classification

Sridhama Prakhya
Email: sridhama@sridhama.com

Vinodini Venkataram
Email: vvenkata@uccs.edu

Jugal Kalita
Email: jkalita@uccs.edu

***Abstract*—Most research in text classification has been done under a *closed world* assumption. That is, the classifier is tested with unseen examples of the same classes that it was trained with. However, in most real world scenarios, we come across novel data that do not belong to any of the known classes, and hence should not ideally be categorized correctly by the classifier. The goal of *open world* classifiers is to anticipate and be ready to handle test examples of classes unseen during training. The classifier can simply declare that a test example belongs to an unknown class, or alternatively, incorporate it into its knowledge as an example of a new class it has learned. Although substantial research has been done in open world image classifiers, its applications in text classification is yet to be explored thoroughly.**

Keywords*—*open-set classification, text classification, convolutional neural networks, deep learning, outlier ensembles, isolation forest, weibull distribution

I. Introduction

With increasing amounts of textual data from various online sources like social networks, text classifiers are essential for the analysis and organization of data. Text classification usually consists of a corpus being assigned one or more classes according to its content. Some popular text classification applications include: spam filtering, sentiment analysis, movie genre classification and document tagging. Traditional text classifiers assume a *closed world* approach. The classifier is expected to be tested with the same classes that it was initially trained with. Such classifiers fail to identify and mitigate when examples of new classes are presented during testing. In real world scenarios, classifiers must be able to recognize unknown classes and accordingly adapt their learning model. This is known as the *open world* approach. A popular example of an open world text classification scenario is authorship attribution. An open world text classifier must recognize the author of a document and subsequently label it appropriately. The classifier must also recognize whether the writing style matches a known author, or is something unknown.

In this paper, we elaborate the methodology that we followed in developing our CNN-based open-set text classifier.

II. Related Work

A majority of existing open-set learning techniques deal with image classification rather than text classification.

The basis of most open-set classifiers is the Nearest Class Mean Classifier (NCM) [16]. This classifier represents classes by the mean feature vector of its elements. An unseen example is assigned a class with the closest mean. This is calculated by taking the distance *(Euclidean)* between the test vector and the computed class mean feature vectors.

Mensink et al. [4] proposed the nearest class mean metric learning (NCMML) approach extending the NCM technique by replacing the Euclidean distance with a learned low-rank Mahalanobis distance. This showed better results than the former as the algorithm was able to learn features inherent in the training data. The Nearest Non-Outlier (NNO) algorithm [3] adapts NCM for open world recognition based on a metric known as *open space risk*. This concept, introduced by Scheirer et al [11], minimizes an error function combining empirical risk over training data with the risk model for the open space. The NNO algorithm proved to perform better at image classification than the NCMML technique.

Regarding closed-set text classifiers, Fei and Liu [1] piloted an approach that they call *CBS learning*. Doan and Kalita [2] built upon the NCM, designing a set of closest neighbors of centroid class rather than the class mean for each class member.

Most ANN-based closed-world text classifiers use recurrent neural network based architectures e.g., Long short-term memory (LSTM) models. The state-of-the-art classifier uses a convolutional neural network model that is 29 layers deep [7]. Conneau et al. were able to show that the performance of their model increased with depth of the network.

III. Method

A. Datasets

For an efficacious open-world evaluation, we must choose a dataset with a large number of classes. This allows us to hide classes during training. These hidden classes can later be used during testing to gauge the open-world accuracy. We plan on using two freely available data sets:

- 20 Newsgroups [14] - Consists of 18828 documents partitioned (nearly) evenly across 20 mutually exclusive classes.
- Amazon Product Reviews [13] - Consists of 50 classes of products or domains, each with 1000 review documents.

B. Evaluation Procedure

Traditional evaluation (closed-set) is when the classifier is assessed with data similar to what was learned during training. The number of classes presented during testing is equal to that the model was trained on. In open-set evaluation, the classifier has incomplete knowledge during the training phase. Unknown classes can be submitted to the classifier during the testing phase. During the training phase, we will train the classifiers on a limited number of classes. While testing, we then present the model with the classes that were not learned during training.

We evaluate the performance of the classifier based on how well it identifies these new classes. "Openness", proposed by Scheirer et al. [9] [11], is a measure to estimate the open-world range of a classifier. This measure is only concerned with the number of classes used rather than the open space itself.

Accuracy, precision, recall, and F-score are used to measure the closed-set performance of our model. These metrics are expanded to the open-set scenario by grouping all unknown classes into the same set. A True Positive is when a known class is correctly classified and a True Negative is when an unknown class is correctly predicted as unknown. False Positives (an unknown class predicted as known) and False Negatives (a known class predicted as unknown) are the two types of incorrect class assignment. We use the F-score as a primary metric compared to accuracy, as it takes the incorrectly classified examples (FP and FN) into consideration.

$$openness = 1 - \sqrt{(2 \times C_T/(C_R + C_E))},$$

where C_R = number of classes to be recognized,

C_T = number of classes used in training, and

C_E = number of classes used during evaluation (testing)

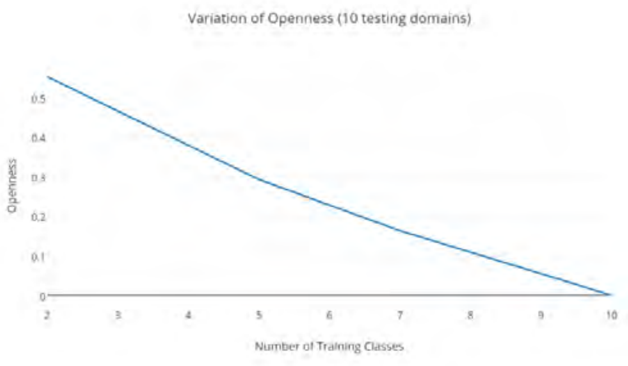


Fig. 1: Variation of openness with number of training classes

IV. Experiments

We initially experimented with distances from mean document vectors to see if they followed a Weibull distribution. We calculated document vectors by taking the mean of all word embeddings in each document. The cosine similarity between each training example and its respective mean document vector was calculated. All vectors were normalized (using Euclidean norm) to improve computation time, as vector magnitude does not affect the angle between two vectors (vector similarity). Table I shows the 5 closest cosine similarities (averaged) between 20 examples from the "comp.graphics" class to other mean document vectors. According to the data, examples from the "comp.graphics" class are more similar to "comp.windows.x", rather than the class itself. Due to the similarities being too close (sometimes overlapping), we concluded that calculating cosine similarity at the document level was not suitable for open-set classification.

We decided to follow a CNN-based approach due to their ability of extracting useful features. For all experiments, the

TABLE I: Cosine similarities between examples of "comp.graphics" to other mean document vectors

Class	Cosine similarity
comp.windows.x	0.23269
comp.graphics	0.24248
comp.os.ms-windows.misc	0.24905
comp.sys.ibm.pc.hardware	0.25001
comp.sys.mac.hardware	0.28630

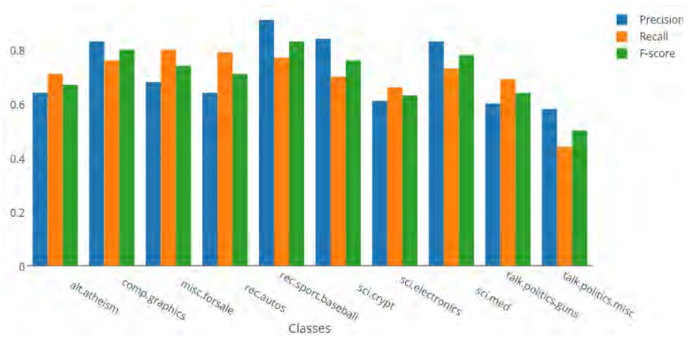


Fig. 2: l_2 constraint = 0.0
Model Accuracy: 0.710309

CNN-static architecture proposed by Kim [12] was used. We used pre-trained *word2vec* [10] vectors for our word embeddings. These embeddings are kept static while other parameters of the model are learned. According to the experiments of Zhang and Wallace [17], imposing an l_2 norm constraint on the weight vectors generally does not improve performance drastically. Figures 2, 3, 4 show the accuracies achieved on the 20 Newsgroups dataset while varying the l_2 norm constraint. The configuration details of the CNN used in all our experiments are shown in Table II. Figure 5 shows the CNN architecture we followed. In our case, we used a single static channel instead of multiple channels.

A. Ensemble Approach

In our open-set classifier, we use an ensemble of different approaches to determine whether an example is known or not. This ensemble includes probabilistic and high dimensional outlier detectors.

1) Isolation Forest: An Isolation forest is a combination of a set of isolation trees. Isolation trees consist of data being recursively partitioned at random partition points with randomly chosen features. Doing so isolates instances into nodes containing one instance. The heights of branches containing outliers are comparatively less than other data points. The

TABLE II: CNN baseline configuration

Description	**Values**
input word vectors	Google word2vec (300 dimensional)
filter sizes	(3,4,5)
feature maps	100
activation function	ReLU
pooling	1-max pooling
dropout rate	0.5
l_2 norm constraint	0.0

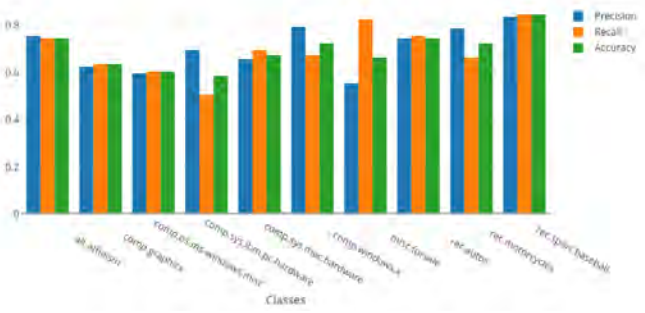


Fig. 3: l_2 constraint = 2.0
Model Accuracy: 0.688253

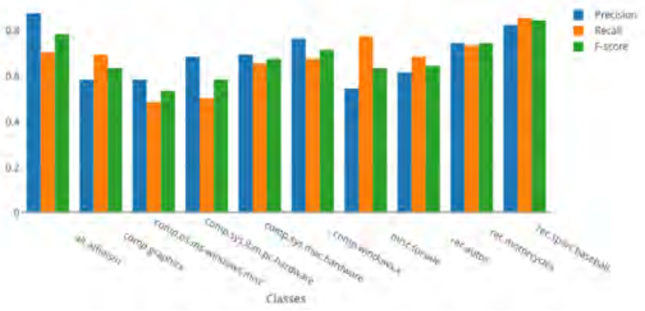


Fig. 4: l_2 constraint = 3.0
Model Accuracy: 0.672197

height of the branch is used as the outlier score. The scores obtained from the isolation forest are min-max normalized. Scores are calculated for every trained class. Examples with scores below a predefined threshold are labelled as unknown. In case of multiple scores above the threshold, the example is assigned to the class with the highest score.

2) Probabilistic Approach: In closed-set classification, the Softmax layer essentially chooses the output class with the highest probability with respect to all output labels. This idea was extended to open-set image classification by Bendale and Boult [8]. They proposed the OpenMax, which is a new model layer that estimates the probability of an input belonging to an unknown class. OpenMax is based on the concept of Meta-Recognition [15]. For all positive examples of every trained class, we collect the scores in the penultimate layer of our neural network. We call these scores activation vectors (AV). We deviate from the original OpenMax by finding the k medoids of every trained class. For every class, the distances between the class activation vectors and the respective k class medoids are calculated. For every activation vector, we take the average of the k calculated distances. As the number of classes in our dataset is far less than those used in image classification, the k medoids of a class are used represent a class more accurately than a single mean activation vector.

In our outlier ensemble, we have used two distance metrics - Mahalanobis distance and Euclidean-cosine (Eucos) distance [8].

Ideally, we want a distance metric that can tell how much an example deviated from the class mean. The Mahalanobis distance does this by giving us a multi-dimensional generalization about the number of standard deviations a point is from the distribution's mean. The closer an example is to the distribution mean, the lesser the Mahalanobis distance. The Mahalanobis distance between point x and point y is given by:

$$d(\vec{x},\vec{y}) = \sqrt{(\vec{x}-\vec{y})'C^{-1}(\vec{x}-\vec{y})} \tag{1}$$

Here, *C* is the covariance matrix that is prior calculated among the feature variables.

The Euclidean-cosine distance is a weighted combination of Euclidean and cosine distances. While using this metric, we do not normalize the activation vectors. Doing so decreases the vector magnitude, thereby affecting the overall distance.

The distances obtained are used to generate a Weibull model for every training class. We use the libMR [15] FitHigh method to fit these distances to a Weibull model that returns a probability of inclusion of the respective class. Figure 6 shows the probabilities of inclusion obtained from the Weibull distribution for a training class from the 20 Newsgroups dataset. As an example deviates more from the mean (k-medoids), the probability of inclusion decreases.

The sum of all inclusion probabilities is taken as the total closed-set probability. Open-set probability is computed by subtracting the total closed-set probability from 1. We then compare the maximum closed-set probability and total open-set probability. If the total open-set probability is greater than the former, we label the example as unknown, otherwise, the example is assigned the class with the highest closed-set probability. Parameters like threshold and distribution tail-size can be be adjusted to decrease the open-space risk.

$$\text{open set probability} = 1 - \text{total closed set probability} \tag{2}$$

We used a voting scheme to combine the three approaches (Mahalanobis Weibull, Eucos Weibull and Isolation Forest). It has been observed that Mahalanobis and Eucos perform nearly the same. Predictions from the Isolation Forest are usually used as a tie-breaker in case of differing predictions. When all 3 predictions differ, we give the Eucos Weibull the highest priority.

V. Results and Discussion

Open-set performance largely depends on the "unknown" classes used during evaluation. This is true especially when classes are not completely exclusive. The activation vectors of similar classes usually overlap in their vector space. Similar to [1], [2], we conduct our experiments by introducing "unseen" classes during testing. In reality, as the train-test partition can be random, we arbitrarily specify the number of testing domains. For every domain, we report our results using 5 random train-test partitions for each dataset. Both datasets are

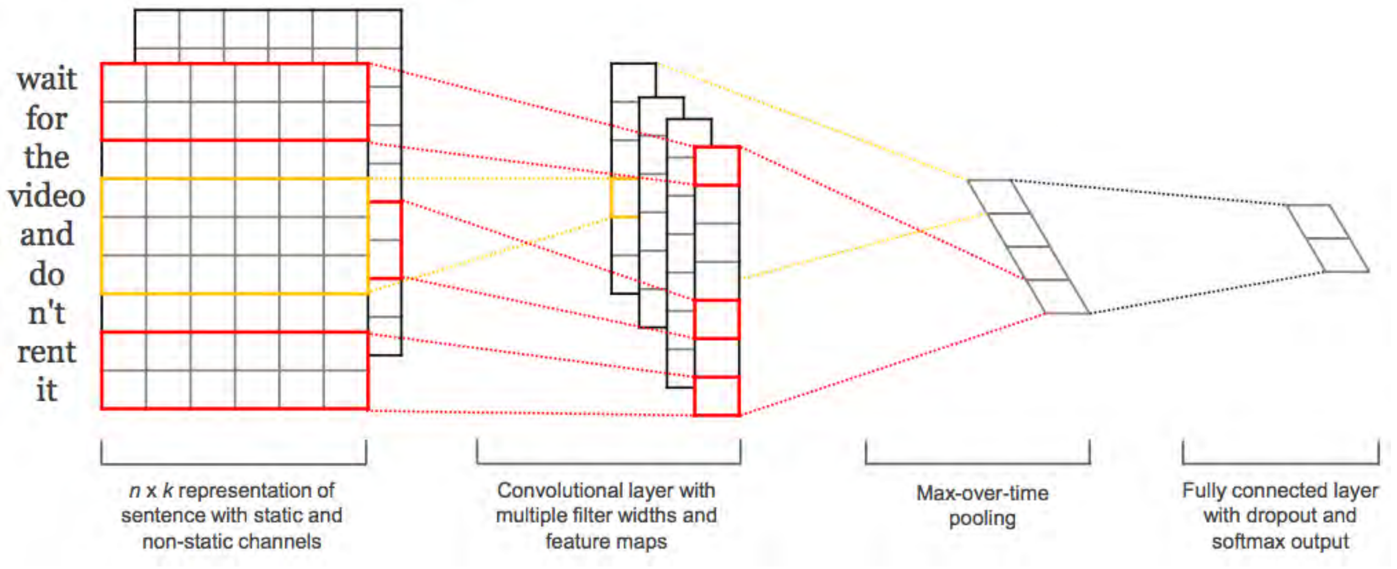

wait
for
the
video
and
do
n't
rent
it
n x *k* representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Fig. 5: Model architecture with two channels for an example sentence (image taken from [12] without permission)

TABLE III: Experiments on Amazon Product Reviews dataset and 20 Newsgroups dataset (10, 20 domains)

Amazon Product Reviews	**10 Domains**			
	25%	**50%**	**75%**	**100%**
our model	**0.797**	**0.753**	0.727	0.821
NCC*	0.61	0.714	**0.781**	0.854
cbsSVM*	0.45	0.715	0.775	**0.873**
1-vs-rest-SVM*	0.219	0.658	0.715	0.817
ExploratoryEM*	0.386	0.647	0.704	0.854
1-vs-set-linear*	0.592	0.698	0.7	0.697
wsvm-linear*	0.603	0.694	0.698	0.702
wsvm-rbf*	0.246	0.587	0.701	0.792
P_i-osvm-linear*	0.207	0.59	0.662	0.731
P_i-osvm-rbf*	0.061	0.142	0.137	0.148
P_i-svm-linear*	0.6	0.695	0.701	0.705
P_i-svm-rbf*	0.245	0.59	0.718	0.774

Amazon Product Reviews	**20 Domains**			
	25%	**50%**	**75%**	**100%**
our model	**0.648**	0.603	0.663	-
NCC*	0.606	0.657	**0.702**	**0.78**
cbsSVM*	0.566	**0.695**	0.695	0.760
1-vs-rest-SVM*	0.466	0.610	0.616	0.688
ExploratoryEM*	0.571	0.561	0.573	0.691
1-vs-set-linear*	0.506	0.560	0.589	0.620
wsvm-linear*	0.553	0.618	0.625	0.641
wsvm-rbf*	0.397	0.502	0.574	0.701
P_i-osvm-linear*	0.453	0.531	0.589	0.629
P_i-osvm-rbf*	0.143	0.079	0.058	0.050
P_i-svm-linear*	0.547	0.620	0.628	0.644
P_i-svm-rbf*	0.396	0.546	0.675	0.714

20 Newsgroups	**10 Domains**			
	25%	**50%**	**75%**	**100%**
our model	**.719**	.747	.738	.864
NCC*	652	**.781**	**.818**	**.878**
cbsSVM*	0.417	0.769	0.796	0.855
1-vs-rest-SVM*	0.246	0.722	0.784	0.828
ExploratoryEM*	0.648	0.706	0.733	0.852
1-vs-set-linear*	0.678	0.671	0.659	0.567
wsvm-linear*	0.666	0.666	0.665	0.679
wsvm-rbf*	0.320	0.523	0.675	0.766
P_i-osvm-linear*	0.300	0.571	0.668	0.770
P_i-osvm-rbf*	0.059	0.074	0.032	0.026
P_i-svm-linear*	0.666	0.667	0.667	0.680
P_i-svm-rbf*	0.320	0.540	0.705	0.749

20 Newsgroups	**20 Domains**			
	25%	**50%**	**75%**	**100%**
our model	**0.668**	0.686	0.685	-
NCC*	0.635	**0.723**	**0.735**	**0.884**
cbsSVM*	0.593	0.701	0.720	0.852
1-vs-rest-SVM*	0.552	0.683	0.682	0.807
ExploratoryEM*	0.555	0.633	0.713	0.864
1-vs-set-linear*	0.497	0.557	0.550	0.577
wsvm-linear*	0.563	0.597	0.602	0.677
wsvm-rbf*	0.365	0.469	0.607	0.773
P_i-osvm-linear*	0.438	0.534	0.640	0.757
P_i-osvm-rbf*	0.143	0.029	0.022	0.009
P_i-svm-linear*	0.563	0.599	0.603	0.678
P_i-svm-rbf*	0.370	0.494	0.680	0.767

evaluated on the same number of test classes (10, 20). We also evaluate our model on smaller domains, shown in Table IV. The number of testing classes used during training is varied in quarter-step increments (25%, 50%, 75% and 100%). We take the floor value in case of fractional percentages. Using 100% of the testing classes during training corresponds to closed-set classification.

Results for the 20 Newsgroups and Amazon Product Reviews dataset are shown in Table III. We report only the F-scores due to space constraints. Our model performs better than cbsSVM and NCC classifiers in smaller domains. Figure 7 shows the activation vectors obtained from models trained on 2 classes plotted in 2-dimensional space. The plots show distinct clusters of the class activation vectors. Due to such distinct clusters, we believe our model performs better than other SVM based approaches in smaller domains.

Unlike cbsSVM, our model is an incremental model i.e. we do not have to retrain the model when new unknown classes are introduced. Such models are more viable in real world scenarios.

VI. Future Work

We are currently working on adapting our open-set classification techniques to multi-layered CNNs. This involves changing the longitudinal kernal (height x 300) to a lateral

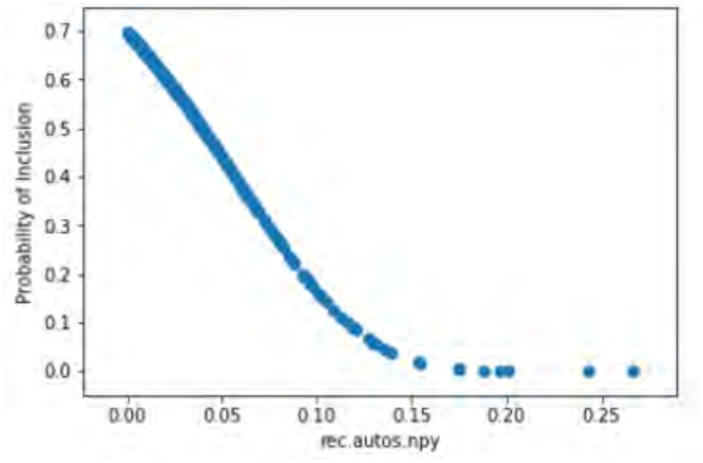


Fig. 6: Weibull distribution for the *rec.autos* class

TABLE IV: Results of Amazon Product Reviews Dataset in smaller domains (3, 4, 5)

Classes Trained on	Classes Tested On		
	3	4	5
2	0.802	0.824	0.808
3	-	0.725	.763
4	-	-	0.797

kernal (height x 1). This allows us to extract activation vectors from the antepenultimate layer which may represent the input data more accurately.

VII. Conclusion

Our incremental open-set approach handles text documents of unseen classes in smaller domains more consistently than existing text classification models, namely *CBS learning* [1] and *nearest centroid class classification* [2]. This research can prove beneficial when classifying novel data, applications of which can be used to tackle tough text classification problems like authorship attribution and sentiment analysis.

VIII. Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant Nos. IIS-1359275 and IIS-1659788. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] Fei, G. and Liu, B., 2016. Breaking the Closed World Assumption in Text Classification. In HLT-NAACL (pp. 506-514).

[2] Doan, T. and Kalita, J., 2017, January. Overcoming the challenge for text classification in the open world. In Computing and Communication Workshop and Conference (CCWC), 2017 IEEE 7th Annual (pp. 1-7). IEEE.

[3] Bendale, A. and Boult, T., 2015. Towards open world recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1893-1902).

[4] Mensink, T., Verbeek, J., Perronnin, F. and Csurka, G., 2013. Distance-based image classification: Generalizing to new classes at near-zero cost. IEEE transactions on pattern analysis and machine intelligence, 35(11), pp.2624-2637.

[5] Jnior, P.R.M., de Souza, R.M., Werneck, R.D.O., Stein, B.V., Pazinato, D.V., de Almeida, W.R., Penatti, O.A., Torres, R.D.S. and Rocha, A., 2017. Nearest neighbors distance ratio open-set classifier. Machine Learning, 106(3), pp.359-386.

[6] Gogoi, P., Bhattacharyya, D.K., Borah, B. and Kalita, J.K., 2011. A survey of outlier detection methods in network anomaly identification. The Computer Journal, 54(4), pp.570-588.

[7] Conneau, A., Schwenk, H., Barrault, L. and Lecun, Y., 2016. Very deep convolutional networks for text classification. arXiv preprint arXiv:1606.01781.

[8] Bendale, Abhijit, and Terrance E. Boult. ”Towards open set deep networks.” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.

[9] Scheirer, W.J., de Rezende Rocha, A., Sapkota, A. and Boult, T.E., 2013. Toward open set recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(7), pp.1757-1772.

[10] 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 (pp. 3111-3119).

[11] Scheirer, Walter J., Lalit P. Jain, and Terrance E. Boult. ”Probability models for open set recognition.” IEEE transactions on pattern analysis and machine intelligence 36.11 (2014): 2317-2324.

[12] Kim, Yoon. ”Convolutional neural networks for sentence classification.” arXiv preprint arXiv:1408.5882 (2014).

[13] Jindal, Nitin, and Bing Liu. ”Opinion spam and analysis.” Proceedings of the 2008 International Conference on Web Search and Data Mining. ACM, 2008.

[14] Rennie, J. 20-newsgroup dataset. 2008

[15] Scheirer, W.J., Rocha, A., Micheals, R.J. and Boult, T.E., 2011. Meta-recognition: The theory and practice of recognition score analysis. IEEE transactions on pattern analysis and machine intelligence, 33(8), pp.1689-1695.

[16] Rocchio, J.J., 1971. Relevance feedback in information retrieval. The Smart retrieval system-experiments in automatic document processing.

[17] Zhang, Y. and Wallace, B., 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. arXiv preprint arXiv:1510.03820.

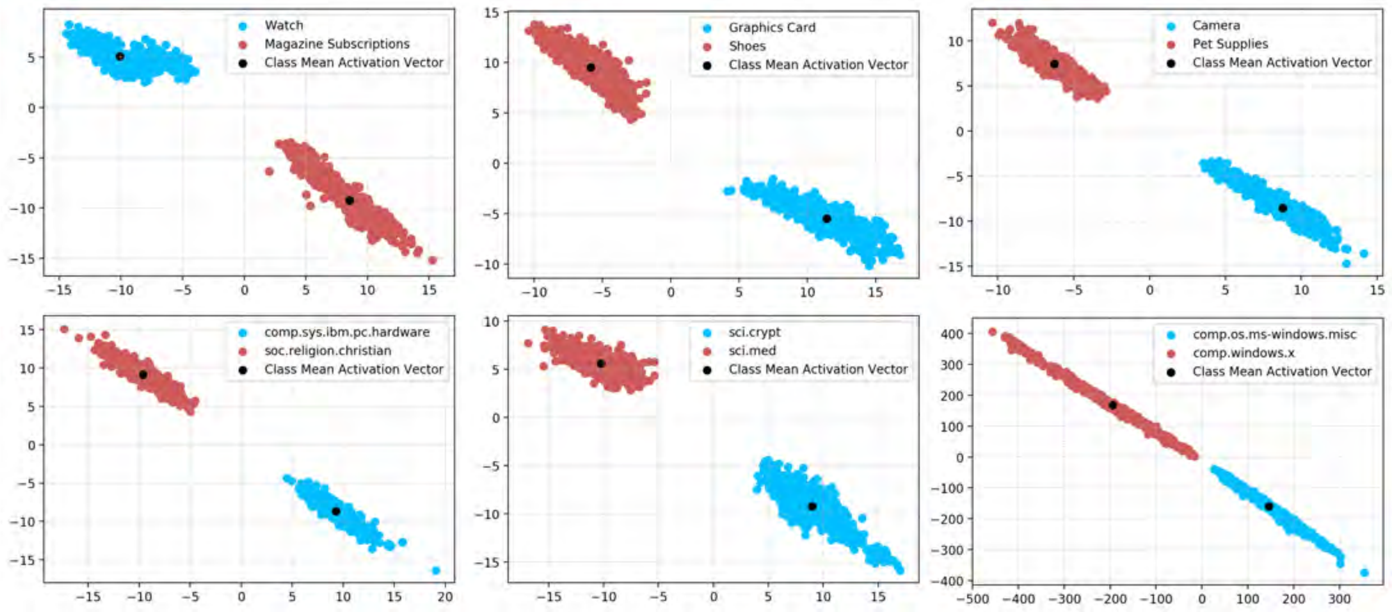


Fig. 7: Activation vectors obtained from models trained on 2 randomized classes.

Survey of Simple Neural Networks in Semantic Textual Similarity Analysis

Derek S. Prijatelj, Jonathan Ventura, and Jugal Kalita

Abstract—**Learning the semantic resemblance between natural language sentences for computers is a difficult task due to the inherent complexity of natural language. To combat this complexity in learning natural language semantics, a rise in the research of complex architectures for machine learning has become prevalent in the field. It is necessary to establish a lower bound of performance that must be met in order for new complex architectures to be not only novel, but also worth while in terms of implementation. This paper focuses on the specific natural language processing task of determining semantic textual similarity (STS). To construct the lower bound that all complex models must surpass to be deemed practical, this research investigated the performance of relatively simpler models in the STS matching task using SemEval's 2017 STS competition dataset.**

Index Terms—**semantic matching, semantic textual similarity, compositional distributional semantics**

I. Introduction

SEMANTICS in natural languages have eluded humans for centuries. Even today, the true meaning of a word can neither be quantified nor computed, but methods have arisen in defining the difference in the meaning between different words and phrases. Distributional semantics plays a key role in this definition of a meaning, where although the actual meaning is unknown, the words or phrases that share the same meaning may be approximated. This is known as the *distributional hypothesis*, where words that share the same context will tend to share similar meaning [1]. While this hypothesis holds true for words, applications of traditional distributional semantics ignore the context of word phrases and longer text fragments [2], [3]. With the introduction of compositional distributional semantics, different methods have been created to represent the meaning of word phrases, and perform better than traditional distributional semantics where context is necessary [3], [4]. There are different methods of representing the semantics of words and phrases, including word embedding and sentence or document embedding. This research concerned itself specifically with the semantic representation of sentences, and compared the different representations in the task of semantic textual similarity matching.

Semantic textual similarity matching is the task of determining the resemblance of the meanings between two sentences. The dataset used for this task is SemEvals' 2017 Semantic Textual Similarity corpus[1][2]. The task specifically is to output a continuous value on the scale from [0, 5] that represents the degree of semantic similarity between two given English sentences, where 0 is no similarity and 5 is complete similarity. In terms of machine learning, this is a regression problem. The 2017 STS corpus contains 1186 English sentence pairs with a corresponding rating and 249 pairs as the test set. The test set has been labeled with the average of multiple human expert ratings that SemEval calls the "golden standard". The distribution of ratings is stated to be as uniform throughout as they could make it, as well as have the same ratios as the training set's ratings.

The models that are examined in this research are simple neural network architectures compared to some of the more complicated models that are popular in recent natural language processing research [5]–[12]. Examining the simple neural network architectures better defines the perspective on creating new architectures for practical applications. If a simple architecture can perform equivalently or better than a newer model, then the new model is simply a new way to accomplish a task using a worse method. To properly establish the threshold that new models must surpass for practical performance in the STS task, simple models such as the perceptron to simple LSTMs and bidirectional LSTMs are evaluated on the STS task. The major components in these models are the pre-trained word vectors, the sentence embeddings, and the comparator of the two sentence embeddings that performs the regression.

II. Related Work

Both semantic representation and related natural language processing tasks have become more popular due to the introduction of distributional semantics. In the recent past, there have been many improvements, enough to make a comparison of the simplest and latest methods necessary to comprehend the current standard.

A. Semantic Representation

Mikolov invigorated the interest in distributional semantics with his team's creation of Word2Vec, a means of representing the co-occurrences of words in written text as vectors in a vector space [2]. This method is fairly successful, but by its very nature does not consider the context of larger phrases; this is where compositional distributional semantics was introduced. Stanford developed another method of computing

D. S. Prijatelj is with the Department of Mathematics and Computer Science, Duquesne University, Pittsburgh, PA, 15282 USA e-mail: prijateljd@duq.edu

This research was supported by the National Science Foundation under Grants No. 1359275 and No. 1659788 at the Department of Computer Science, University of Colorado at Colorado Springs, Colorado Springs, CO, 80918 USA

Manuscript received August 3, 2017

[1]http://alt.qcri.org/semeval2017/task1/index.php?id=data-and-tools
[2]http://ixa2.si.ehu.eus/stswiki/index.php/Main_Page

the distributional semantics of text and this method is known as GloVe. GloVe is similar to Word2Vec in that it computes the co-occurrence frequency of words and creates a vector of a specified dimension to represent a word, but the methods they use are somewhat different [13]. Either may be used for natural language processing tasks depending on preference or performance of the pre-trained word embeddings. In this research, 300 dimensional GloVe word vectors will be used as the initial state of the word vectors.

There are various methods that exist to embed a sentence represented by a list of word vectors. Some of these methods involve the use of neural networks, including, but not limited to, LSTMs and their variations [5], [10], [14], [15]. There have also existed some algorithms to compute sentence representation as well, either similar to methods applied for word embeddings or as extensions of the word embedding methods [3], [4]. Arora et. al 2016 proposed a ”simple but tough-to-beat baseline for sentence embeddings” called the SIF embedding method [16]. SIF involves taking the average of all the word vectors in a sentence and removing the first principal component. Arora et. al have reported it to be a satisfactory baseline and it will be compared to simple neural networks for embedding to determine which is a better baseline for sentence embeddings.

B. Semantic Matching

Different simple semantic matching processes will be examined as the comparator component of the models. These methods will be compared to modified versions of one of the recent developed neural net approaches to semantic matching, the Matrix Vector-LSTM (MV-LSTM) from [5]. The modified versions in this research will replace the similarity tensor with euclidean distance and cosine similarity to establish an understanding of the simplified models' performance. The models used will keep the use of bidirectional LSTMs for learning the sentence embeddings from the paired list of word vectors, and will keep the multi-layered perceptron at the end of the comparator component.

There exist other recent architectures for semantic matching of sentences. One of these newer architectures is the Deep-Fusion LSTM (DF-LSTM) by Penfang et. al 2016 [9]. The DF-LSTM builds upon two LSTM stacks of equal length for reading the two sentences by connecting the individual LSTM's together with a connection between the paired LSTM units' inputs. It's performance rivals that of the MV-LSTM, but is a more complicated version than the LSTM models examined in this paper. There are more complex architectures for semantic matching or similar tasks as well, which supports the need for an established lower bound of practical performance derived from the simpler models [6]–[8].

III. Examined Models

The simple models examined all share the same architecture. Pre-trained word embeddings, a sentence embedding component, and a comparator component. The sentence embedding component takes the list of word vectors that represents a

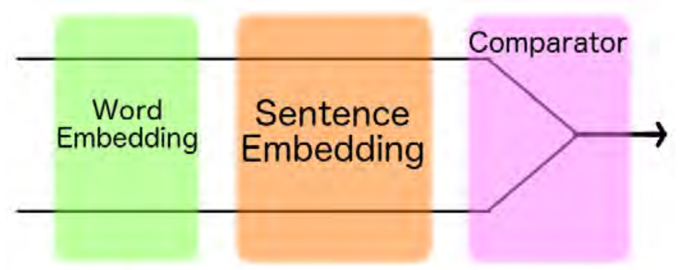


Fig. 1. The overall architecture of the simple models for the STS task. Two string sentences are are the inputs and one float in the range [0, 5] is the output.

sentence and combines them into a single vector that represents the meaning of the original sentence. The comparator component is the part of the model that evaluates the similarity between the two sentence vectors and performs regression to output the sentence pair's similarity score on the continuous inclusive scale from 0 to 5. For all components and individual neural network units of the model, ELU activations are used. The initial weights of each unit are randomly initialized using the He normal distribution [17]. For all models, RMSprop is used as the optimizer with a learning rate of 1e-4. Mean squared error is the loss function for all models as well. The metrics that are calculated are mean squared error, and the Pearson correlation coefficient (PCC), or Pearson R. The SemEval STS competition uses the PCC as the primary metric of a model's performance.

A. Pre-Trained Word Vectors

The models start with the input of two sentences represented by strings. The sentences are embedded into word vectors based on the provided pre-trained word embeddings, which in this case is a set of GloVe word vectors. This specific set of word vectors have 300 dimensions and were pre-trained on 840 billion tokens taken from Common Crawl[3]. Different pre-trained word vectors may be used in-place of this specific pre-trained set. After being embedded into the pre-trained word vectors, the data is randomly shuffled and then sent to the sentence embedding component.

B. Sentence Embedding

The model component responsible for taking a list of word vectors that represent a sentence and embedding them into a single vector to represent the entire sentence. The sentence vector should compress the size of the data that represents the sentence, yet still maintain the important information of the semantics of the sentence.

1) Smooth Inverse Frequency (SIF): Arora et. al 2017 proposed their method of sentence embedding called smooth inverse frequency (SIF) as a simple baseline for all sentence representations to surpass [16]. Their method involves taking the mean of all word vectors in a list and removing the first principal component. They found that this simple method of

[3]https://nlp.stanford.edu/projects/glove/

sentence embedding creates satisfactory results. SIF will serve as the simplest method of sentence representation tested in all models in this research.

2) LSTM: Sentences are sequences of words where order matters and each word may affect any other's meaning despite their location in the sentence. Given that sentences are sequences, it is only natural to use the version of the recurrent neural network known as the LSTM. The version of the LSTM used throughout model is based on the original from Horchreiter et. al 1997 [18]. This sentence embedding component consists of a single LSTM per sentence with a number of hidden units in parallel equal to that of the word embedding's number of dimensions.

3) Stacked LSTMs: The stacked LSTMs' construction is the same as the the single LSTM, except that instead of one LSTM per sentence there are two stacks of LSTMs of equal length. All hyper-parameters are the same otherwise. Various sized stacks of LSTMs are experimented with, including 2, 3, 4, 5, and 10. Multiple LSTMs should be able to capture the kernels of meaning in a sentence. As stated by Palangi et. al 2016, the higher the number of LSTMs in the stack, the better the predicted performance of the sentence embedding [14].

C. Comparator

The comparator examines the two sentence embeddings and performs regression on them to find a continuous value on the inclusive range from 0 to 5. This continuous value indicates the level of similarity between the two sentences, where 0 is no semantic similarity and 5 is complete semantic similarity.

1) Perceptron: The simplest of all the comparators, the perceptron with ELU as its activation is used as the regression operation. The weights are initialized at random using the He normal distribution. The outputs from the sentence embeddings are concatenated and sent to a fully connected dense layer, which then connects to a single output node.

2) LSTM: In order to learn the relationship between the words in the two sentences, a LSTM takes the concatenated sequence output from the two LSTM sentence embedding components. This single LSTM performs the regression on the two embeddings and learns how the two embeddings relate to one another.

3) Stacked LSTMs: Applying the reasoning behind deep LSTM stacks as proposed by Palangi et. al 2016, a stack of LSTMs is used as the comparator of LSTM sentence embeddings. The process is the same as the single LSTM comparator, but instead with a stack of LSTMs. Varying sizes of stacks are used, but match the size of the LSTM stacks in the sentence embedding component.

D. Simplified MV-LSTM: L2-LSTM

Unlike the other simple models that come in parts, this model comes together as a whole. A simplified version of the MV-LSTM from Wan et. al 2016 will also be tested among the simple models [5]. This model matches the MV-LSTM exactly except for the similarity tensor which is replaced with a euclidean distance calculation to compare the similarity to the two sentence embeddings. Bidirectional LSTMs are used for the sentence embeddings and the euclidean distance is followed by a multilayered perceptron with 3 layers that cuts their density in half from the previous layer. The first layer has 5 nodes. This simplified version of the MV-LSTM will be referred to as the L2-LSTM.

IV. Current Implementation

To analyze the task of semantic textual similarity,

The training data is then given to the neural networks with mean squared error as their loss, due to this being a regression problem. After training, the model is evaluated using the remaining fold as the validation set.

V. Evaluation Process

The SemEval STS dataset is already provided with training and testing data. Each dataset either has or can be made to have proper training, testing, and validation proportions. Also, Each dataset can easily be evaluated to determine the statistics of the models' performances. Most, if not all, already have their own rules for determining the performance of the model. Each model will be evaluated on each task and will be compared to other models tested in this research as well as those tested on the same datasets in the same standardized fashion. These external models have their information listed on the websites hosting the datasets. The overall best model in each task will be indicated, as well as those with in the subcategories of supervised, semi-supervised, and unsupervised.

VI. Results

The results indicate that models with a better capacity for memory storage are better suited for solving the STS task optimally. The simplified MV-LSTMs also perform approximately the same as a perceptron, and thus should be discarded from use in practical application for the STS task. However, these are only simplified versions of the MV-LSTM. The actual MV-LSTM could perform better than these less complicated versions.

A. LSTM

The Single LSTM embeddings for both the perceptron comparator and the single LSTM comparator performed worse than any of the models that included a stack of LSTMs. This indicates that the memory of a single LSTM compared to that of a stack of LSTM is unable to learn the semantic kernels of a sentence. This encourages the use of models with increased memory due to their ability to learn important semantic features of a sentence.

B. Stacked LSTMs

The stacked LSTMs performed the best overall with the paired stack of 10 LSTMs for embedding and a perceptron comparator as the best of all LSTM stack embedding and perceptron comparator models. The stack of 2 LSTMs with a stack of 2 LSTMs as the comparator performed the best out of all of the models with a .05 lead over the second place model, the stack of 10 LSTMs and perceptron model. The success

Simple Models' Mean Performances across 10 K-Fold Cross Validation		
Model Name	Pearson R	In-Sample Pearson R
2 LSTM Stack and 2 LSTM Stack Comparator	0.86075441	0.99629578
10 LSTM Stack and Perceptron	0.78242820	0.90816820
2 LSTM Stack and Perceptron	0.75952832	0.87565377
3 LSTM Stack and Perceptron	0.72353597	0.82746079
4 LSTM Stack and Perceptron	0.71500020	0.82689888
5 LSTM Stack and Perceptron	0.45382610	0.64646486
1 LSTM and Perceptron	0.43011681	0.54451700
1 LSTM and 1 LSTM	0.41634211	0.99024633
L2-LSTM 50 epochs	0.2740426	0.3536559
L2-LSTM 100 epochs	0.2183386	0.3065541
SIF and Perceptron	0.2211686836	0.879533587

Fig. 2. The mean Pearson R out-of-sample and in-sample from k-fold cross validation where k = 10. The LSTM and LSTM Stack embeddings were all computed with 50 epochs. The SIF embedding and perceptron comparator were calculated with 100 epochs. The Model Names are ordered by embedding component and comparator, except for the L2-LSTM model which is combined embedding and comparator.

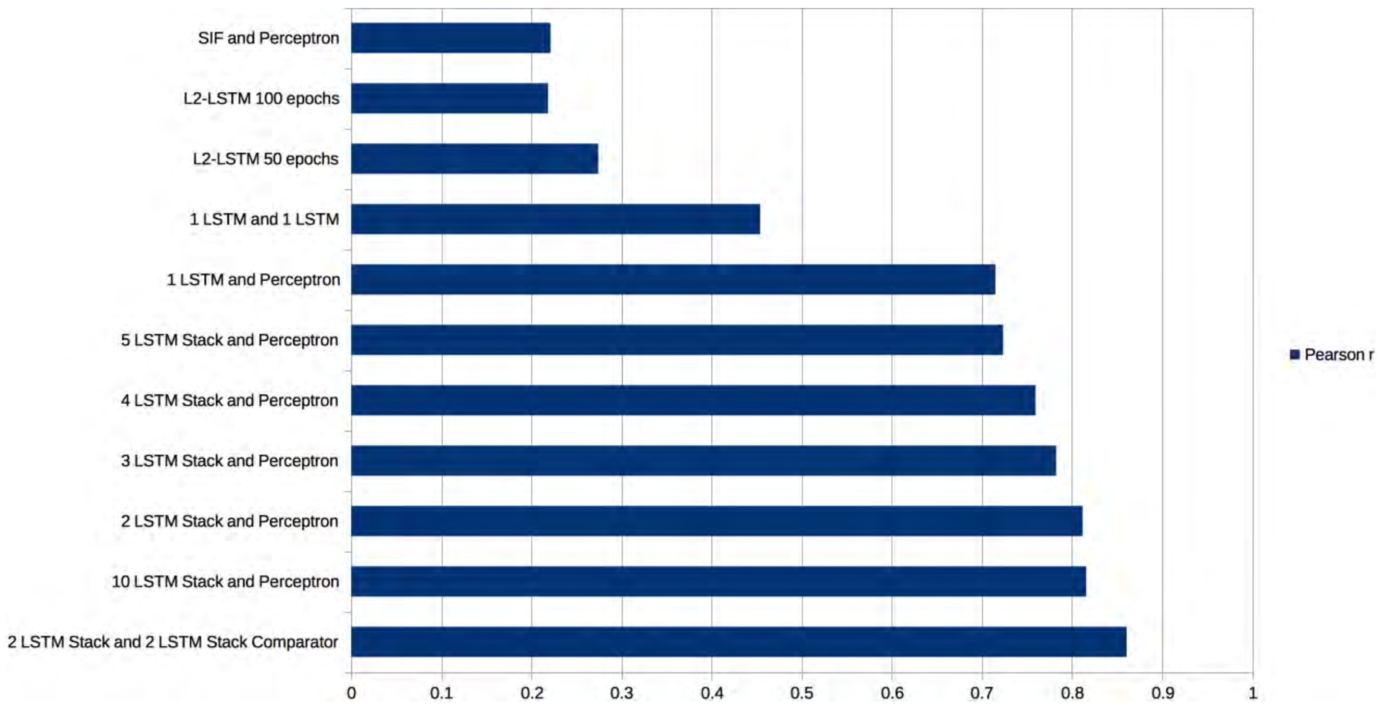


Fig. 3. The mean Pearson R across all test and validation sets in k-fold cross validation where k = 10.

of the LSTM stacks indicates that these models were able to learn kernels of meaning in the sentences and compare them correctly to one another. The quality performance from these models raise the standards for newer, more complex models for the STS task.

C. Simplified MV-LSTM: L2-LSTM

The L2-LSTM performed worse than any of the other models, except for the perceptron when compared to the MV-LSTM with 50 epochs. This indicates that either the bidirectional LSTMs are not suitable for learning the semantics between the two sentences, or the similarity comparison with the euclidean distance is not as effective as the power of the learning the sequences with LSTMs. Given its performance roughly matches that of a perceptron, the L2-LSTM is a model not to be used given its similar performance to, but greater complexity than the perceptron.

VII. Further Research

The performance of the simplified MV-LSTMs bring into question the adequacy of the original MV-LSTM for the STS task. The next step is to evaluate the performance of the MV-LSTM in the STS task and compare it to that of the LSTM stacks. The results indicated that models with a higher capacity for memory were better suited to learn the semantic representation of the sentence and appropriately compare them. These results encourage further research in memory augmented neural networks for use in learning the semantics of natural languages. Exploring the implementation of more complicated memory augmented neural networks, such as the DNC model created by Graves et. al 2016, is the next step in pursuing better performance in sentence embedding and semantic textual similarity matching [19].

VIII. Conclusion

The performances of various simple neural network models have been examined on the task of semantic textual similarity matching using SemEval's provided dataset. The model to perform the best with a Pearson correlation of 0.8608, based on the mean k-fold cross validation, is the model where a stack of 2 LSTMs embedded the sentences and were then compared with another stack of 2 LSTMs after concatenating the two sentence embedding stacks' sequences output. This supports the findings that natural language tasks are sequence problems where the elements in the sequence have interconnected relatedness, in which neural networks with memory are better at learning. The large number of LSTMs in the stack also suggests that there exist major groups of meaning in a sentence that can be learned to know the unique meaning of that sentence. This supports the findings with the MV-LSTM. The evaluation of these simple models for semantic textual similarity serves as the lower bound to compare all other models that possess increased complexity in their design. All future researchers should ensure that their new model architectures surpass these lower bounds.

References

[1] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *ArXiv preprint arXiv:1301.3781*, 2013.

[3] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 1188–1196.

[4] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, "Skip-thought vectors," in *Advances in neural information processing systems*, 2015, pp. 3294–3302.

[5] S. Wan, Y. Lan, J. Guo, J. Xu, L. Pang, and X. Cheng, "A deep architecture for semantic matching with multiple positional sentence representations," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[6] Z. Wu, H. Zhu, G. Li, Z. Cui, H. Huang, J. Li, E. Chen, and G. Xu, "An efficient wikipedia semantic matching approach to text document classification," *Information Sciences*, vol. 393, pp. 15–28, 2017.

[7] J. Fu, X. Qiu, and X. Huang, "Convolutional deep neural networks for document-based question answering," in *International Conference on Computer Processing of Oriental Languages*, Springer, 2016, pp. 790–797.

[8] J. Guo, Y. Fan, Q. Ai, and W. B. Croft, "Semantic matching by non-linear word transportation for information retrieval," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, ACM, 2016, pp. 701–710.

[9] P. Liu, X. Qiu, J. Chen, and X. Huang, "Deep fusion lstms for text semantic matching," in *Proceedings of Annual Meeting of the Association for Computational Linguistics*, 2016.

[10] P. Liu, X. Qiu, and X. Huang, "Dynamic compositional neural networks over tree structure," *ArXiv preprint arXiv:1705.04153*, 2017.

[11] ——, "Adversarial multi-task learning for text classification," *ArXiv preprint arXiv:1704.05742*, 2017.

[12] ——, "Recurrent neural network for text classification with multi-task learning," *ArXiv preprint arXiv:1605.05101*, 2016.

[13] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162.

[14] H. Palangi, L. Deng, Y. Shen, J. Gao, X. He, J. Chen, X. Song, and R. Ward, "Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 24, no. 4, pp. 694–707, 2016.

[15] J. Chen, K. Chen, X. Qiu, Q. Zhang, X. Huang, and Z. Zhang, "Learning word embeddings from intrinsic and extrinsic views," *ArXiv preprint arXiv:1608.05852*, 2016.

[16] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," 2016.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *CoRR*, vol. abs/1502.01852, 2015. [Online]. Available: http://arxiv.org/abs/1502.01852.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735. eprint: http://dx.doi.org/10.1162/neco.1997.9.8.1735. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[19] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, *et al.*, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, 2016.

Computing Semantic Roles Using ANN's with External Memory

Christopher Towne
New College of Florida
Email: Christopher.Towne14@ncf.edu

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

Abstract—**Building on recent improvements in neural based Semantic Role Labeling (SRL) and on some recent improvements in neural architectures, in this paper we demonstrate a new memory network based model for SRL. Unlike previous neural architectures for the task of SRL, our model's architecture has the addition of long term memory capabilities and recall. In order to do so we had to remove the bidirectional capabilities of our controller and weight the cost function of the model in order to make it small and sensitive enough to be trained at all. However despite the working model, due to some of the weaknesses of the kind of memory network that we used, the Differntiable Neural Computer (DNC), we are currently unable to demonstrate fully trained results.**

Keywords*—*Semantic Role Labeling, Frame Semantic Parsing, FrameNet, Differntiable Neural Computers

I. Introduction

Semantic Role Labeling (SRL) is a natural language processing task that involves the figuring out the different semantic roles that different words play in a sentence. In all frameworks of SRL, those roles are defined in reference to something, either some concept or some verb. In the form of SRL that this paper is involved in, frame semantic parsing, the roles that words play are tied to the different semantic frames that lie within a given sentence. With the same word being able to belong to multiple frames without any problem. Taken from the FrameNet project [1] , each semantic frame represents a kind of event, relation, or entity and their constituent parts.

As an example, if one takes the standard sentence: "Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo" and if one were to attempt to parse it in the task of frame semantic parsing, the task would be to note that the lexical units, the words that invoke the frame itself, in this case "Manipulate_into_doing", are the 5th and 6th buffalo(s), that role of Manipulator falls on the 2nd buffalo in the frame invoked by the 5th buffalo and falls on the 4th buffalo in the frame invoked by the 6th buffalo, and that the role of Victim falls upon the 2nd buffalo in the frame invoked by the 5th buffalo and falls upon the 8th buffalo in the frame invoked by the 6th buffalo.

Our approach to this task is a neural based approach similar to those done by Collobert et al. [2], Zhou and Xu [3], Swayamdipta et al. [4], and He et al. [5] . However the difference in our approach is the addition of a new recurrent neural based technique that we have added to our model. As neural techniques have gotten better and added into new SRL models, the capabilities of those models have also increased with them. As such into our model we utilized the recently publish technology of Differntiable Neural Computers (DNC) [6] in order to amplify the capabilities of our model.

Unfortunately, like all recently created technologies, while the DNC system has fixed a number of the kinks that its predecessor, the Neural Turing Machine [7], had, it still has a number of issue yet to be worked out. Namely problems with efficiency of memory access because of the DNC architecture's use of soft attention. There are multiple different lines of SRL setups that have been worked on throughout out the years. The framework that surrounds neural SRl attempts on datasets of the PropBank annotations, have developed a standard setup, however the framework for neural architectures that work on the FrameNet dataset are much more recent. As such we have chosen to follow the lead of Swayamdipta et al. [4] and focus our work on tackling the problem of argument identification in FrameNet. Our decision was mostly a practical one as a result of the aforementioned inherent problems that DNC setups have with efficiency and more namely with the lengths of time spent in training.

A. Related work

The first work and the pioneering work on the automation of SRL was done by Gildea and Jurafsky [8]. For a more recent overview of the task, Màrquez et al. [9] have a detailed introduction.

The first work on the subject of neural SRL was done by Collobert et al. [2]. In their paper they built a few end-to-end neural networks which all undertook a few separate natural language processing tasks, including the PropBank version of SRL, with the same instances of time delayed convolutional neural networks; yet, while the system was competitive within the three other natural language processing tasks, their system had some difficulty handling automated SRL.

That difficulty with SRL was later addressed by Zhou and Xu [3] who made improvements over the Collobert et al.'s convolutional approach by utilized the memory capacities of recurrent neural networks with a deep bidirectional LSTM network in order to have the neural network learn and remember the long term dependency's through out the sentences. The addition of short term memory allowed for a neural approach that was capable of rivaling the other, feature engineered, methods in SRL.

Recently He et al. [5] modified and improved Zhou and Xu's architecture by adding to it highway connections, recurrent dropout, decoding with BIO constraints, and ensemble methods among other things. They improved upon Zhou and Xu's result, and creating the current state of the art system in the PropBank framework.

Also recently Swayamdipta et al. [4] applied segmental RNNs with a softmax-margin cost function to frame semantic parsing and also achieved state of the art performance in FrameNet's subsection of SRL.

II. Task Overview - Frame Semantic Parsing

FrameNet's style of annotating semantic roles is based around the concept of semantic frames. More specifically FrameNet maintains a list of semantic frames, also referred to as just frames, and for each of those frames FrameNet also maintains a list of lexical units, words that can invoke those frames, and frame elements, the roles or arguments that make up the context of each frame.

Within the annotations of FrameNet, annotated sentences are marked with targets, the instances of lexical units that are invoking frames in those instances, and the target's invoked frame. And for each of those target and frame pairs, phrases or clusters of words that serve as frame elements are mark with the identity of the roles they play.

The task in frame semantic parsing is the task of identifying the location and identities of frame elements that belong to each frame in a sentence, however the details of the task varies. Depending, some attempts of the task take a more in depth route and not only identify the frame elements, but also identify the frames and targets themselves. That would have been the preferred route of this work, however due do the limits of speed of training a DNC model we did not take that route and instead chose to focus on frame element, argument, prediction.

III. Model Summary

The model that we demonstrate in this paper is a instance of a Differntiable Neural Computers (DNC) [6]. DNCs are a recent kind of neural network addon that were built upon the advancements of recurrent neural networks, except that they take memory a bit further and add to a given network the additional capabilities of long term memory storage. A DNC is not a neural network architecture on its own, but is composed of an external memory matrix, which is gated by a number of accessing functions, and coupled with a normal neural network that controls access to the external memory matrix, all of which is differntiable.

A. Controller Network

The controller of a DNC, or the neural network that is coupled with the memory matrix, can be any neural network architecture as the only modification that the DNC does to its controller is to change the size of the input into its controller. At the beginning of each timestep, before the controller is run, the previous output of the DNC's memory matrix is append to the input. And after that run of that timestep a portion of the controller's output is split off into different vectors that are passed through the gates of the memory accessing function in order to dictate the internal on going of the otherwise external memory.

[*] Taken without permission from Deepmind's website (deepmind.com)

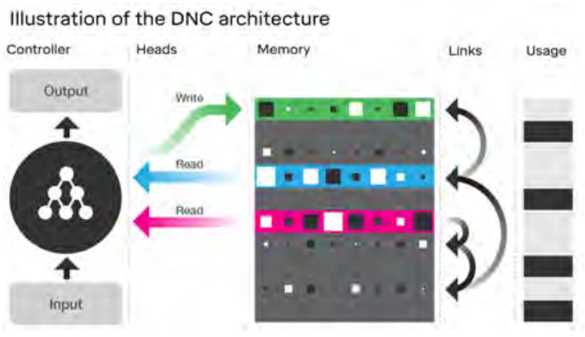


Fig. 1: An example of the framework of the DNC architecture [*].

The model that we used to control the memory matrix is LSTM stack of 8, where each of those 8 were created with a hidden size of 300. Around each of the LSTM cells in that stack highway connections [10], a form of gating that allows input to either go through the neural network layer or pass around the layer unchanged, are wrapped around them. In addition, around the 6 LSTM cells in the middle of the stack, dropout [11] around their output has been implemented.

B. Input and Output Setup

In our setup there are three main feature in the input, the untouched words, the lemmas of those words, and the frame and target identification. Each of those feature is placed into an embedding and trained. The initial embedding for words and lemma are taken from pretrained GloVe [12] embeddings on 840 billion tokens. Any word or lemma that was not found in GloVe was given an randomly initialized vector for as unk symbol. The embeddings for frames were randomly initialized. In addition to those core input features we also gave the DNC parts of speech, dependency, and syntactic parsings as input features as well. Those feature were not embedded.

Most other neural models for SRL utilize bidirectional recurrent neural networks. However because of the addition of the memory matrix and the soft attention used to access it, DNCs do not scale well and creating a bidirectional network with DNC's was infeasible with our computational limitations. As a work around to that issue we decided to experiment with instead giving the network its input twice: once to analysis the input, and once again to have the network give its predictions. As such all of the input sequence is doubled except for a binary sequence used to indicate to the model the valid and unmasked output locations in the sequence, which is only on during the second iteration of the input sequence.

C. Cost Function

The output of our model was a sequence of vectors, where each element in the vector represented a frame element id. However at most or really all timesteps in the sequence most of the frame element output markers would be marked off. As such every every output node tended toward zero using an unweighted cost function. We used two additional weightings to reorient the loss.

The first weight was a basic weighting to balance out the classes (in this case the frame element identities), and the second was a simple penalty on the loss of the output nodes in the timesteps in the sequence where a false negative had occurred. The weight we settled on for that penalty was a times 13. Some alternate weight schemes had been try, namely lowering the cost for all nodes where the label was null. However that proved to be too finicky to find a hyperparameter weight value and the scheme was switched to penalizing false negatives.

IV. Differentiable Neural Computers

Differentiable Neural Computers are in the most recent iteration of the attempt to augment the already impressive short term memory capabilities of recurrent neural network with a form of long term memory storage. Like other kinds of memory networks, DNCs have a long term component; in their case, their external memory matrix. Unlike the memory in something like an LSTM, where memory is split up and distributed throughout the network, the memory in a DNC's memory matrices is external and explicitly accessed.

To access its memory matrix, the DNC utilizes soft attention. The DNC forms read, write, and erase vectors for both keys and strength. One of the interesting properties of the DNC is it weighs its reading of the matrix by both an explicit vector and by a content weighed vector. For a memory matrix $M_t \in \mathbb{R}^{NxW}$, the read vectors, $[r_t^1, .., r_t^R]$ where R is the number of read head, are defined as $r_t^i = M_t^\intercal w_t^{r,i}$, where $w_t^{r,i}$ is the weighting of that row in the matrix for that head. Defining the read weights, w^r, themselves is a bit more complicated, but in short it is the sum of temporal weighting (the time of writing) and the content bases weight, which itself is a cosine distance based metric multiplied by the explicitly passed in weight for the strength of reading at that row location.

Writing in a DNC system has two parts, erasing and writing. More specifically it is defined as $M_{t+1} = M_t \circ (E - w_t^w e_t^\intercal) + w_t^w v_t$. Where $\circ$ denotes an element wise multiplication, E is a matrix of ones, w^w is the write weight, $e \in [0,1]^W$ is the erase vector, and lastly v is the write vector, the actual value to write. Once again the complicated part is w^w, the writing weight. In short, there is a gate that gates the decision to write, which gates another gate that decides whether to use a dynamic allocation based writing system or a content similarity based writing system.

In short the DNC does three main things: it takes read arguments and returns sums of the rows of the memory matrix, weighted by both explicitly passed weights and content similarity based weights. It takes a collection of write arguments and can write to dynamically selected locations or to content similar locations. And it is differentiable. Once it finishes all those operations, the read vectors are passed through a weight matrix and added to the controller output. In addition the read vector are append to the input of the next time step.

V. Experiments

A. Dataset

In our experiments we used data taken from the 1.5 version of the FrameNet database. More specifically we used the same training, development, and testing sets as Das and Smith [13]. In addition the non-core features and the lemmas mention in the Input and Output Setup subsection were all taken from prior processing done by Das and Smith on the splits. For some of the data multiple frames with the same target and frame identities were given, with differing only on the frame elements addressed. Those frames were merged into one frame and treated as such in training and testing. In addition one frame was thrown out of the training set, the frame 'Test35', as it could not be located within FrameNet itself and did not appear in either the development or testing sets.

Furthermore, because FrameNet is setup so that each of its frames treat their frame elements as separate roles, it is difficult to compress the number of possible frame elements as they are unorganized. As such we compressed them by name; so two frame elements both called Time in two different frames would be assumed to represent the same semantic role within their frames an were given one identity within the model.

B. Experimental Setup

As noted above, for practical reasons with the DNC we have chosen to focus on frame element/argument identification within this paper. A model for identifying frame identifications was created, where the prediction of that model could be piped into the model for frame element identification; however since training that model on top of the argument identification model was infeasible and because of the complication it caused with optimizing hyperparameters it was shelved. In addition the because of the DNC's slowness in gradient decent we were unable to have enough time to run more than about 14 epochs.

C. Evaluation

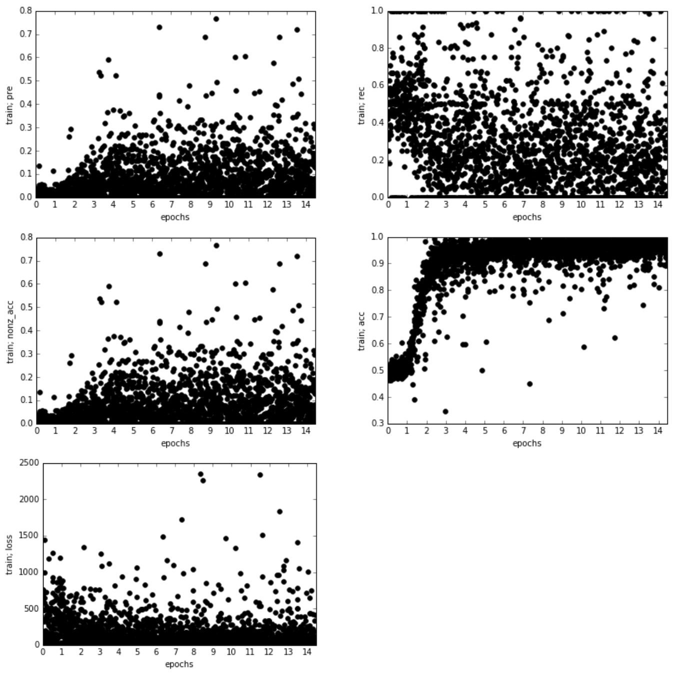
To evaluate, we used a modified version of the standard evaluation script from SemEval '07, where point score for frame evaluations was set to zero. However we are not an expert in Perl so that may have broken it, but it likely does as it is supposed to. That Perl script takes in XML documents to compare, as such we we developed a pipeline to convert the output of the neural network into XML. To do so we masked out any frame elements that could not occur with the frame that the model was given. We then rounded the output of the neural network, grouped the output by continuous streams of the same predicted frame element, and converted those streams into two start and end tags. The SemEval '07 script gives three metrics: recall, precision, and the F score.

D. Results

We will use one other papers in frame semantic parsing as a base line to compare our work; the work of Swayamdipta et al. [4]. While there is other work in argument identification in SRL, there are few other papers the look at argument identification on its own with the FrameNet database. Most other argument identification systems have in the past focused on the PropBank styled datasets.

	P	R	F
Swayamdipta et al.	71.7	66.3	68.9
DNC Model	0.00034	0.00297	0.00062

In addition, below we provide graphs of our DNC's performance on the training set in different metrics:



As a note, the precision and recall scores in the graphs above are not measure in the same way that the precision and recall scores are generated by the evaluation script. The script generates its scores by comparing the nodes of trees generated from the XML, while the scores in the graphs are generated by directly comparing the predicted sequence to the gold standard label. Furthermore, because the script is tree based it is possible that the script looks for more precision in the predicted frame elements that our training metrics, like exact matching of frame element locations or similar.

E. Analysis

Unfortunately, the main conclusion that we are able to draw from this work is that the addition of the DNC's memory matrix and operations makes a model really slow. It took two and a half days to train the model over 14 epoch. To reach a decent amount of epoch, say 500, it would take 3 months. As a result of of low amount of training, there are few solid conclusions that we are capable of drawing. We had intended to run some comparisons of the metrics in terms of the commonous of frame elements, the distance between targets and frame elements, and the length of the sentence; however, with the low accuracy in the XML document, they all drop to zero, rendering it pointless.

VI. Future Work

In this paper we have described a working model that comes with long term memory capabilities, however, unfortunately, describing is the main thing we have done. Upgrading our DNC model into a fully functional SRL model would likely take a good bit more work. There is not much of a way to know with the number of epochs trained over how our setup is. As such there are a few directions that we can take our work.

The most straight forward path would to be to just run it for a few months and get the proper result, however doing so might be ridiculous depending on resources. Another path would be to shrink the model in either the controller size or in the size of the memory matrix and to streamline the training process, however if we did such we would never learn the true capabilities of the larger DNC models. One last option would be to, potentially, drop the soft attention of the DNC model and attempt to train them using reinforcement learning methods [14] or evolutionary methods [15].

VII. Acknowledgments

This work was supported by the REU program at the University of Colorado, Colorado Springs, which is supported by the NSF grants #1659788 and #1359275 and by professors Kalita, Ventura, and Boult.

References

[1] C. F. Baker, C. J. Fillmore, and J. B. Lowe, "The berkeley framenet project," in *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, Association for Computational Linguistics, 1998, pp. 86–90.

[2] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[3] J. Zhou and W. Xu, "End-to-end learning of semantic role labeling using recurrent neural networks.," in *Association for Computational Linguistics*, (Beijing, China), ACL, Jul. 2015, pp. 1127–1137.

[4] S. Swayamdipta, S. Thomson, C. Dyer, and N. A. Smith, "Frame-semantic parsing with softmax-margin segmental rnns and a syntactic scaffold," *arXiv preprint arXiv:1706.09528*, 2017.

[5] L. He, K. Lee, M. Lewis, and L. Zettlemoyer, "Deep semantic role labeling: What works and what's next," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2017.

[6] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, *et al.*, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, 2016.

[7] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[8] D. Gildea and D. Jurafsky, "Automatic labeling of semantic roles," *Computational linguistics*, vol. 28, no. 3, pp. 245–288, 2002.

[9] L. Màrquez, X. Carreras, K. C. Litkowski, and S. Stevenson, "Semantic role labeling: An introduction to the special issue," *Computational linguistics*, vol. 34, no. 2, pp. 145–159, 2008.

[10] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015.

[11] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Advances in neural information processing systems*, 2016, pp. 1019–1027.

[12] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162.

[13] D. Das and N. A. Smith, “Graph-based lexicon expansion with sparsity-inducing penalties,” in *Proceedings of the 2012 conference of the North American chapter of the Association for Computational Linguistics: human language technologies*, Association for Computational Linguistics, 2012, pp. 677–687.

[14] W. Zaremba and I. Sutskever, “Reinforcement learning neural turing machines,” *arXiv preprint arXiv:1505.00521*, vol. 419, 2015.

[15] R. B. Greve, E. J. Jacobsen, and S. Risi, “Evolving neural turing machines for reward-based learning,” in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, ACM, 2016, pp. 117–124.

Vector Properties of Good Summaries

Adly Templeton
Williams College
at7@williams.edu

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

***Abstract*—Vector semantics is used in many areas of Natural Language Processing. We explore the application of vector semantics to the problem of automatic summarization. We demonstrate several properties of vector semantics useful for this purpose. In particular, we show that cosine similarity between sentence vectors and document vectors is strongly correlated with sentence importance and that vector semantics can identify and correct gaps between the sentences chosen so far and the document. In addition, we identify specific dimensions which are linked to effective summaries. To our knowledge, this is the first time specific dimensions of sentence embeddings have been connected to sentence properties. We also compare the features of different methods of sentence embeddings. Many of these insights have applications in uses of sentence embeddings far beyond summarization.**

***Index Terms*—Vector Semantics, Automatic Summarization, Extractive Summarization, Sentence Embeddings**

I. Introduction

The large volume of news articles published every day motivates effective forms of automatic summarization. The most basic and most well-studied form of automatic summarization is sentence extraction, in which entire unmodified sentences are selected from the original document. These selected sentences are concatenated to form a short summary, which ideally contains the most important information from the original document while avoiding redundancy.

While a variety of successful models have been proposed for sentence extraction, these models often rely on various metrics, such as term frequency, which are based on the occurrence of individual words. For instance, a state-of-the-art graph-based summarization model by Parveen et al. uses the words shared between sentences for all three of their major metrics (importance, redundancy, and coherence) [1]. However metrics based on the occurrence of specific words, and not the meaning of these words, may perform sub-optimally in certain situations (such as dealing with synonyms).

Meanwhile, *vector semantics* have been growing in popularity for many other natural language processing applications. Vector semantics attempt to represent words as vectors in a high-dimensional space, where vectors which are close to each other have similar meanings. Various models of vector semantics have been proposed, such as LSA [2], word2vec [3], and GLOVE [4], and these models have proved to be successful in other natural language processing applications. While these models work well for individual words, producing equivalent vectors for sentences or documents has proven to be more difficult.

This material is based upon work supported by the National Science Foundation under Grant No. 1659788 and 1359275

II. Problem Description

For our purposes, extractive summarization is essentially reducible to *sentence selection*. That is, we want to select a subset of sentences from the set of all sentences in the original document, which maximizes the quality of the summary while remaining under some word limit. Note that this problem is more complex than text classification: The quality of each individual sentence depends of the other sentences in the summary. In particular, a good summary should contain minimal redundancy. This trade off between sentence salience and redundancy is the basis for many summarization algorithms, including some in our work.

Any practical summarization system would likely include other steps after sentence selection, such as *sentence reordering* or *text-simplification*. However, effective algorithms for these tasks already exist, and we are primarily concerned with sentence selection in this paper.

We are primarily concerned with multi-document summarization algorithms, which summarize a *cluster* of related documents. However, all our algorithms disregard the document-by-document information and treat the document cluster.

III. Related Work

Particular attention should be given to the method of combining word embeddings into sentence embeddings, a very difficult problem. However, news summarization may not require some of the nuances in meaning to be represented, as we are primarily concerned with the topic, not the meaning, of a sentence. In particular, news articles will rarely contain sentences expressing contradictory views on the same subject. For instance, our algorithm will rarely need to differentiate between the sentence embeddings for sentences such as “John loves Mary” and “Mary loves John”, which have divergent meanings but the same words. This is in sharp contrast to typical testing cases for sentence embeddings, such as the detection of paraphrased sentences. Then, summarization gives us an opportunity to compare the effectiveness of different sentence embeddings in practice.

In recent years, a number of techniques for sentence embeddings have emerged. One promising method is *paragraph vectors* (Also known as Doc2Vec), described by Le and Mikolov [5]. The model behind paragraph vectors resembles that behind word2vec, except that a classifier uses an additional ‘paragraph vector’ to predict words in a *Skip-Gram* model. When trained, the paragraph vectors capture the meanings of their paragraphs.

Another model, *skip-thoughts*, attempts to extend the word2vec model in a different way [6]. The center of the skip-thought model is an encoder-decoder neural network which, given a sentence, attempts to predict the surrounding sentences. The result, *skip-thought vectors*, achieve good performance on a wide variety of natural language tasks.

Li et al. used a LSTM-based auto encoder, where the encoding in the middle of the autoencoder is taken as a vector representation of a sentence. In addition, a hierarchical version of the autoencoder can also be used to form vectors for whole paragraphs or documents. [7]

Despite the complexity of these neural network approaches, simpler approaches based on linear combinations of the word vectors have managed to achieve state-of-the-art results for non-domain-specific tasks [8]. Arora et al. [9] offer one particularly promising such approach. First, this model finds the weighted average of the word vectors (less frequent words weighted higher). This weighting is achieved through the smooth inverse function $w = \frac{a}{a+f}$, where w is the weight of a word's vector, f is the estimated frequency of that word, and a is a hyperparameter ($a \approx 0.0001$). Note that this weighting is roughly analogous to weighting by tfidf. The second step is to subtract the projection along a specific vector, c_0. c_0 is the "common discourse vector", correlated with words such as "the" or "and" which appear consistently in all English contexts. c_0 is found by taking the first principal component of a representative sample of text vectors. This simple method was found to achieve equal or greater performance in some tasks than more complicated supervised learning methods.

A. Extractive Summarization

State-of-the-art extractive summarization techniques have been achieved with a wide variety of methods. Here, we provide a short overview of some recent successful techniques techniques.

Cao et al. [10] used a recursive neural network, which operates on a parsing tree of a sentence, to rank sentences for summarization.

Cheng and Lapata [11] successfully used a neural-network-based sentence extractor. This extractor was a recurrent neural network with a neural attention mechanism. The network considered the document encodings and the previously selected sentences, as well as the current sentence in making its decisions.

Parveen et al. [1] used a graph-based approach, modeling a document as "a bipartite graph consisting of sentence and entity nodes", and used various rankings on the graph to select sentences according to certain criteria (importance, redundancy, and coherence).

Ren et al. [12] achieved state-of-the-art results through a regression-based approach. A variety of engineered features are used as inputs into a regression model, which estimates the redundancy-aware importance of a sentence given the sentences already selected. The single highest rated sentence is selected, and the process is repeated.

B. Previous Embedding-based Approaches

To our knowledge, no one has explored the use of modern sentence embedding methods, such as Paragraph Vectors or Skip-Thought vectors, in summarization. However, some work has been done on summarization using word2vec representations.

Gong and Liu [13] presented a version of text summarization based on vector semantics. However, instead of using the embeddings of a word as determined by a larger corpus, they attempted to calculate the embedding of a word based off of analysis only on the document in question. In addition, due to the age of their work, they used LSA instead of techniques such as word2vec.

Kageback et al. [14] used cosine similarity between the sum of word2vec embeddings, as well as a recursive auto-encoder, to modify another standard summarization algorithm (sub-modular optimization)

Ren et al. [12] used "Average Word Embedding" as one of many independent variables in a regression model, but it is unclear how much that particular variable contributed to the success of the model.

Cheng and Lapata [11] used word embeddings as the input to a neural network as part of summarization, but they did not directly compare embeddings. They used a single-layer convolutional neural network to encode sentences, and a LSTM neural network to encode documents.

Nayeem and Chali [15] recently achieved state-of-the-art results using a modified version of LexRank, using a combination of cosine similarity of weighted averages of vectors and named entity overlap.

IV. Methods

To explore potential sentence embeddings, we implement the four sentence embeddings above as *vector functions*, which convert sentences or documents to vectors.

A. Vector Functions

- *SIF Average*: The most basic sentence embedding is simply the weighted average of word vectors from Arora et al. [9], without the common component removal. We use the Brown corpus [16] for word frequency information.
- *Arora*: This method is simply the method described in Arora et al. It is equivalent to the one above, except with common component removal added. We use the Brown corpus both to compute the common component vector, and for word frequency information.
- *Paragraph Vectors*: The paragraph vector approach described above. We used the 300-dimensional DBOW model pretrained by Lau et al. [17] on the wikipedia corpus.
- *Skip-Thought Vectors* The skip-thought vector approach described above. We used the 4800-dimensional combined-skip model [6], [18].

All sentence embeddings are normalized to produce unit vectors

B. Potential Selector Functions

There are many potential ways to use vector semantics. To explore the design space, we consider combinations of *vector functions* and *selector functions*, functions which, given vector representations for sentences, extracts a summary. We present a large variety of example selector functions in order to allow us to explore interaction effects between selector functions and vector functions.

- *Near*: The most basic selector function, *Near*, selects the sentences whose sentence vectors have the highest cosine similarity with the document vector
- *Near Nonredundant*: An attempt at balancing redundancy with salience, Near Nonredundant down-weights the cosine similarity scores by their average cosine similarity the sentences selected so far. Because this redundancy measure is strongly (quadratically) correlated with cosine similarity to the document, we fit a regression for redundancy for each vector function, and use the residual on this regression for the final algorithm.
- *LexRank*: Our next selector is based off of the classical LexRank algorithm [19]. The center of this algorithm is a weighted graph where nodes represent sentences, and weights are the similarities between sentences, as determined by cosine similarity between tfidf vectors. The PageRank algorithm is then used on this graph to identify the most salient sentences for extraction. We use a modified version of this algorithm, where the weights of edges are determined by the cosine similarity between sentence embeddings.
- *Cluster*: The basis of this selector is a simple clustering algorithm in the vector space of sentence embeddings. We use an Agglomerative Clustering algorithm (using cosine similarity as its distance metric) to find clusters in the set of sentence embeddings. We then find the sentence closest to the average of each cluster and add it to the summary. To ensure we find summaries which meet the word-length requirement, we increase the number of clusters we search for until we have selected sentences totaling 100 words.
- *Greedy*: The greedy selector, at each step, selects the sentence such that the cosine similarity of the new summary (including previously selected sentences) is maximized. This is subtly different than the Near selector for average-based vector functions, but significantly different for Paragraph Vectors.
- *Brute Force*: Another attempt at optimizing the cosine similarity between the summary and the document, this selector creates a pool of the 20 sentences with the highest cosine similarity. From this pool, every combination of sentences (with an appropriate word count) is tried, and the combination with the highest cosine similarity is selected as the summary.
- *Max Similarity*: A proof-of-concept selector which computes results for both the Greedy and Brute Force selectors and then selects the result with the highest cosine similarity to the document vector.
- *Near-then-Redundancy*: Similar to the Brute Force selector, this selector creates the same pool of sentences described above, except with a size of 15. From this pool, this algorithm optimizes via brute force to minimize redundancy (defined as the average cosine similarity between pairs of sentences). Note that the size of the sentence pool, which is essentially a computational shortcut in the Brute Force selector, is now a performance-critical hyper-parameter.
- *PCA*: This selector performs Principal Component Analysis (PCA) on the set of sentence vectors in a document. Then, the algorithm selects the one sentence closest to the first component, one sentence closest to the second component, and so on, until the length capacity is met.
- Random: This selector simply selects sentences at random, until the word limit is reached. This provides a lower-bound on the performance of an effective algorithm, and is used for baseline comparisons

V. Performance of Selector Functions

A. Experimental Evaluation

Because evaluation of summarization is fundamentally a subjective task, human evaluations are ideal. However, human evaluations are often expensive and time-consuming to obtain. Luckily, some metrics of automatically evaluating summaries, by comparison to a human-written summary, have been developed. Traditionally, various forms of the ROUGE metric, which compare shared n-grams, have been used. [20]. ROUGE has been shown to correlate strongly with human judgments [21], and is our primary metric for evaluating summaries[1] We report ROUGE-1 and ROUGE-2 statistics, which correspond to unigrams and bigrams, respectively.

We split the document clusters in the DUC 2004 dataset into a testing set and a validation set of approximately equal sizes. The pre-defined training set of the DUC 2001 dataset was used as a training set for some of the graphs and data analysis presented here.

B. Results

We present results for Multi-Document Summarization on the DUC 2004 dataset (Table I). A few notes on the results:

- Our results fall far below the state of the art, likely due to the simplicity of our selector functions as compared to the state-of-the-art methods. In addition, almost all of our methods are unsupervised, unlike most of the state-of-the-art methods.
- The best performing selector, Greedy, is both very simple and based on fundamental principles of vector semantics.
- Paragraph Vectors work much worse with the Clustering and Greedy algorithms, and work much better with Near and SVMs.
- Many combinations of selector function and vector function do not work above the level of random chance.
- In general, despite their sophistication, Paragraph Vectors and Skip-Thought vectors perform worse than much more basic approaches.

[1]For direct comparison with Hong et al., we trucate summaries to 100 words and use the following parameters, for direct comparison with Hong et al. [22]: -n 4 -m -a -l 100 -x -c 95 -r 1000 -f A -p 0.5 -t 0.

	SIF Average	Arora	Paragraph Vectors	Skipthought
LexRank	32.6 (6.8)	32.6 (6.8)	32.6 (6.8)	**32.6 (6.8)**
Near Nonredundant	33.6 (6.1)	**34.5** (6.3)	32.6 (5.5)	32.1 (4.9)
Brute Force	32.0 (5.7)	32.2 (6.3)	**33.0 (6.6)**	31.4 (4.5)
Near-then-Redundancy	33.2 (6.2)	34.2 **(6.9)**	31.5 (5.4)	33.1(5.3)
PCA	32.9 (5.6)	33.5 (5.6)	32.0 (5.5)	NA
Max Similarity	32.0 (5.7)	32.2 (6.3)	**33.0 (6.6)**	NA
Greedy	**35.1 (7.0)**	33.1 (6.0)	NA	NA
Near	32.5 (5.4)	32.2 (5.5)	33.1 (6.1)	NA
Cluster	NA	NA	NA	32.1 (4.6)
Random	30.1 (4.1)			
State-of-the-art (Ren et al.) [12]	40.4 (11.7)			

TABLE I: ROUGE-1 Results on the DUC 2004 dataset. ROUGE-2 results in parentheses. All combinations which do not perform significantly better than random chance ($p < .05$, using a paired t-test) are replaced with 'NA' for clarity. SIF Average with either Max Similarity or Brute Force were included, despite having p=.051. In addition, one combination (Max Similarity with Skipthought Vectors) are not computed, but are not expected to perform better than chance. Selector Functions are roughly organized according to the vector functions with which they are effective. For Skipthought vectors, *docvec-avg* is used (Section VI-E)

	SIF Average	Arora	Paragraph Vectors	Skipthought
Near Nonredundant	-1.98	-1.72	-0.734	+3.81
Brute Force	-0.323	-0.256	-1.24	+3.42
Near-then-Redundancy	+0.739	-0.584	-0.205	+3.46
Max Similarity	-0.323	-0.256	-1.24	NA
Greedy	+0.254	-0.813	-3.11	NA
Near	-0.868	-0.0652	-5.53	+1.76
Total Average	-.417	-.614	-2.01	+2.74

TABLE II: A comparison of document vector methods. Numbers represent the difference in ROUGE-1 scores between document vector methods. Positive numbers represent a gain when using *docvec-avg*. Selectors which do not use the document vector have been omitted.

VI. Discussion

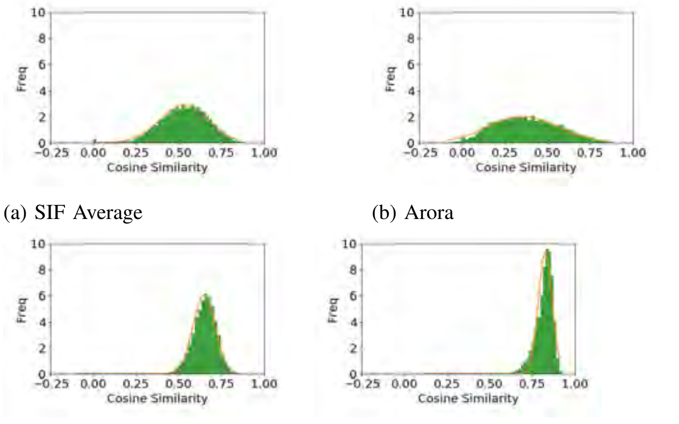
Despite the poor performance of our models compared to the baselines, analyses of the underlying data provide many useful insights into the behavior of vector semantics in real-world tasks.

A. Distributions of Cosine Scores

The cosine scores between all sentence vectors and the corresponding document vectors follow a normal distribution for all vector functions (Fig. 1), but this effect is most pronounced for paragraph vectors ($r^2 = .996$). In addition, the sentence embeddings for paragraph vectors and skip-thought vectors are far closer to the document embedding than would be expected from a random distribution, with mean cosine similarities of .65 and .84, respectively (Unsurprisingly, this also holds for Average and Arora, though the similarity is notably lower (.52 for Average, .35 for Arora).

B. Correlation of Cosine Scores with Good Summaries

By identifying the sentences present in an optimal summarization, we show that optimal sentences have higher cosine scores, and that this effect is increased after adjusting cosine scores for word length (Fig. 2). However, there is a lot of overlap, implying that, although this method has some power to discern good summaries from bad summaries, the power of this method alone is not high enough to product good summaries.



(a) SIF Average (b) Arora

(c) Paragraph Vectors (d) Skipthought

Fig. 1: Distribution of cosine similarity scores between each sentence vector and their corresponding document vector, for all four vector functions.

C. Regression on Vector Dimensions

We calculated the isolated ROUGE score of each individual sentence in the training set, and the sentence embeddings for these sentences on all four vector functions. To partially eliminate the effects of the sentence's context, we subtract the corresponding document vector from all sentence vectors before regression.

Due to the large number of predictor variables, we use a Bonferroni correction, considering values significant only if they have p-values of $\frac{\alpha}{n}$, which, for $\alpha = .05$, corresponds

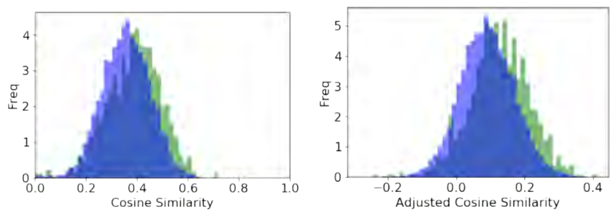
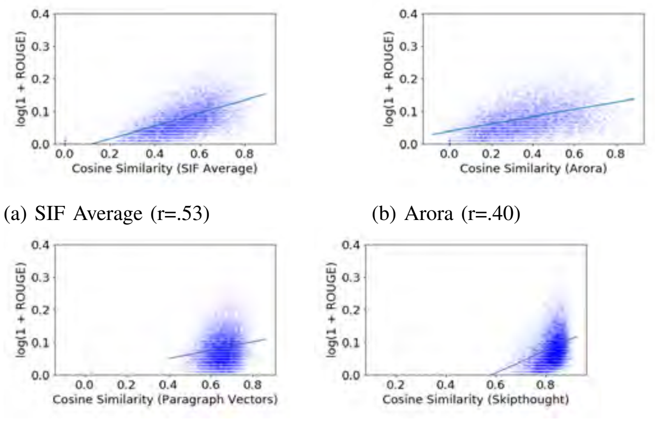


Fig. 2: Cosine Similarity to the Document Vector for non-optimal (blue) and optimal (green) sentences. Figure on the right shows Cosine Similarity adjusted for sentence word count.



(a) SIF Average (r=.53) (b) Arora (r=.40)

(c) Paragraph Vectors (r=.17) (d) Skipthought (r=.33)

Fig. 3: Correlation of ROUGE scores and Cosine Similarity scores per sentence. ROUGE scores transformed with $r' = log(1 + r)$, to account for zero values. Some high-leverage points have been excluded for Paragraph Vectors and Skipthought.

approximately to $p < .00001$ for the skip-thought vectors, and $p < .00016$ for all other vectors.

Three dimensions are significant at this level for *SIF Average* vectors. No dimensions are significant at this level for *Arora* vectors, though the three values significant for *SIF Average* achieve p values of .0021, .0485 and .0006. 29 dimensions are significant for *Paragraph Vectors*. 5 dimensions are significant, at the much higher threshold, for *Skip-Thought Vectors*. It appears that these specific dimensions correspond to aspects of a sentence that make it somehow more suited for a summary. Despite the theoretical implications of this result, the regression models do not have enough predictive power to create good summaries by themselves.

D. The Performance of the Greedy Algorithm

The Greedy algorithm is the most successful algorithm we present here. As its name implies, the Greedy algorithm appears to be simply an attempt at maximizing the following objective function:

$$f_{cos}(summary) = vector(summary) \cdot vector(document) \quad (1)$$

Of course, this objective function can only be an approximation to the informally-defined criteria for good summaries.

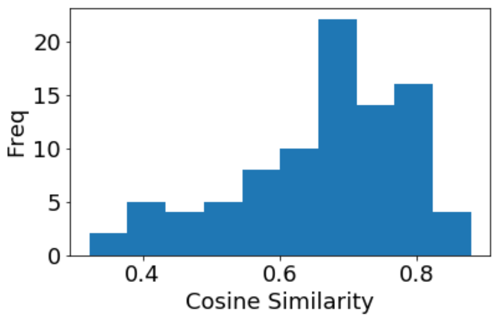


Fig. 4: A histogram of the cosine similarities of sentences selected by the Greedy algorithm.

Even so, Table I suggests that the performance of the greedy algorithm is not based on the accuracy of the corresponding objective function. In particular, consider the two other strategies which try to maximize the same objective function: Brute force, and Maximum Similarity (which simply selects Greedy or Brute Force based on which one creates a summary with a higher cosine similarity). Brute Force consistently and significantly creates summaries with higher cosine similarity to the document, outperforming the Greedy selector on its objective function. By construction, the Max Similarity algorithm outperforms in cosine similarity to an even greater degree. But both of these algorithms perform much worse than the Greedy algorithm.

Deeper analysis into the decisions of the Greedy algorithm reveals some reasons for this discrepancy. It appears that the good performance of the Greedy algorithm results not from the associated objective function, but by the way in which it maximizes this objective function. In particular, the Greedy algorithm selects sentences with low cosine similarity scores in a vacuum, but which increase the cosine similarity of the overall sentence (Fig. 4).

To understand why this is true, it we consider the step-by-step behavior of the Greedy algorithm. The first choice of the greedy algorithm is simple: it chooses the sentence with maximum cosine similarity to the document vector:

$$\bar{s}_1 = \operatorname*{argmax}_{\bar{s} \in S} \bar{s} \cdot \bar{d}$$

(Recall that all vectors have unit-length, so cosine similarity is equivalent to the dot product).

To select the second vector, the greedy algorithm is maximizing the following equation:

$$\bar{s}_2 = \operatorname*{argmax}_{\bar{s} \in S'} \left(\frac{\bar{s} + \bar{s}_1}{\|\bar{s} + \bar{s}_1\|}\right) \cdot \bar{d} \quad (2)$$

$$= \operatorname*{argmax}_{\bar{s} \in S'} \frac{\bar{d} \cdot \bar{s}_1 + \bar{d} \cdot \bar{s}}{\sqrt{1 + \bar{s}_1 \cdot \bar{s}}} \quad (3)$$

Where S' is the set of remaining sentences [2]

[2]Note that the results reported above do not represent the Greedy algorithm averaging together the vectors, though the difference is minimal for SIF Average and Arora vectors(see Section VI-E for more information)

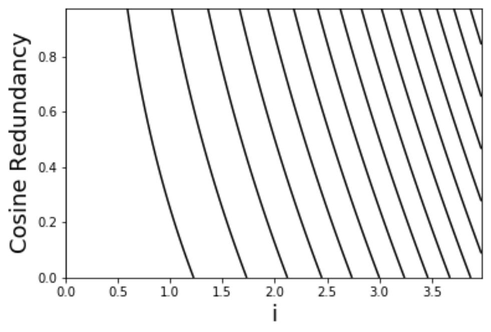


Fig. 5: A contour plot of the denominator of Equation 5

Note that this equation consists of three parts: $\bar{d} \cdot \bar{s}_1$ (a constant wrt. $\bar{s}$), $\bar{d} \cdot \bar{s}$, which is simply the salience of a sentence measured by cosine similarity, and the denominator, which is essentially a measure of redundancy. Not only does this simple metric lead to a 'natural' penalty for redundancy, it performs better than our handcrafted redundancy penalties. The way this algorithm scales when picking the i^{th} sentence is particularity noteworthy:

$$\bar{s}_{i+1} = \underset{\bar{s} \in S'}{\operatorname{argmax}}\left(\frac{\frac{1}{i+1}\bar{s} + \frac{i}{i+1}\bar{s}_p}{\|\frac{1}{i+1}\bar{s} + \frac{i}{i+1}\bar{s}_p\|}\right) \cdot \bar{d} \tag{4}$$

$$= \underset{\bar{s} \in S'}{\operatorname{argmax}} \frac{i(\bar{d} \cdot \bar{s}_p) + \bar{d} \cdot \bar{s}}{\sqrt{i^2 + 1 + 2i\bar{s}_p \cdot \bar{s}}} \tag{5}$$

$$\text{Where } \bar{s}_p = \frac{\sum_{j=0}^{i} \bar{s}_j}{\|\sum_{j=0}^{i} \bar{s}_j\|}$$

As shown in Figure 5, the behavior of this function changes as i increases. In particular, the function becomes more sensitive to redundancy, and less sensitive to salience, as the algorithm selects more sentences. In other words, the algorithm will first try to select important sentences, and then select sentences to fill in the gaps. This result, and the success of the resulting algorithm, has implications for balancing salience and redundancy in future summarization algorithms.

E. Document Vector Computation

In general, there are two ways to compute a document vector. The most obvious is to pass the entire text of the document into the vector function. This has two theoretical problems. The first is that the 'documents' in our algorithms are really clusters of documents, and are therefore non-coherent. The second is that Skip-thought vectors are not designed to handle text longer than a sentence. However, an alternative document vector, *docvec-avg*, is formed by taking the mean of the (normalized) sentence vectors. This corresponds to treating the document as a collection of sentences, instead of a collection of words. We present a comparison of the two methods in Table II

As expected, Skipthought vectors, which are not designed for text larger than a sentence, perform significantly better with the *docvec-avg* strategy. More notable is the poor performance of the *docvec-avg* strategy with Paragraph Vectors. The size of the performance gap here implies that Paragraph Vectors can combine information from multiple sentences in a manner more sophisticated than simple averaging.

More interesting is the performance for SIF Average and Arora vectors. For these vector functions, which are based on taking the average of words, *docvec-avg* very closely resembles the simple strategy. And yet there is a small but significant performance gap. The difference between the two document vectors is the weighting. *Docvec-avg*, which normalizes vectors before adding them together, removes some weighting information present in the simple strategy. In particular, the simple strategy assigns more weight to sentences with a lot of highly-weighted words. Presumably, *docvec-avg*, by ignoring this weighting, leaves out useful information. This hypothesis is supported by the greater performance gap for Arora vectors, which effectively downweights certain common words and therefore could be expected to carry more information in word weightings. Similar, but much smaller, gaps exist when computing the vectors for summaries at each step in the greedy algorithm.

F. Properties of Different Sentence Embeddings

Based on the interactions between selector functions and vector functions, as well as a variety of other pieces of data in the proceeding sections, we present a broad comparison of the properties of different sentence embedding schemes.

1) SIF Average/Arora Vectors: Three selector functions perform better with both SIF Average and Arora vectors: Near Nonredundant, Greedy, and PCA. These functions seem to be united in their comparisons between the vectors of sentence embeddings (implicitly, in the case of the greedy algorithm). These selector functions correspond to the most basic test for sentence embeddings: Judging the similarity of two sentences.

The exact difference the common component removal makes is less clear. Arora vectors hold a slight performance edge for all selectors except for Near and Greedy (the Greedy algorithm loses a full 2 points).

2) Paragraph Vectors: Two selector functions perform better with Paragraph Vectors: Near and Brute Force. Both of these are very similar: They require comparing sentence vectors to the document vector. The poor performance on algorithms such as Near Nonredundant suggests that Paragraph Vectors are especially poor at comparing sentence vectors to each other. These results suggest that Paragraph Vectors are especially good at computing document vectors, a hypothesis also implied by the results of Section VI-E.

The other distinguishing property of Paragraph Vectors is their very high correlation when regressing on the individual features.

3) Skipthought Vectors: It is hard to disentangle the properties of Skipthought vectors from the high dimensionality of the pretrained vectors we used. In general, Skipthought vectors performed poorly. They only performed better than other vector functions with one selector, Clustering, although their performance with this selector was significant.

In general, these results suggest that different sentence embedding methods are suited for different tasks, often drastically so, and the choice should be made carefully.

References

[1] D. Parveen and M. Strube, "Integrating importance, non-redundancy and coherence in graph-based extractive summarization." in *IJCAI*, 2015, pp. 1298–1304.

[2] T. K. Landauer and S. T. Dumais, "A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge." *Psychological Review*, vol. 104, no. 2, p. 211, 1997.

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[4] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation." in *EMNLP*, vol. 14, 2014, pp. 1532–1543.

[5] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 1188–1196.

[6] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, "Skip-thought vectors," 2015, pp. 3294–3302.

[7] J. Li, M.-T. Luong, and D. Jurafsky, "A hierarchical neural autoencoder for paragraphs and documents," *arXiv preprint arXiv:1506.01057*, 2015.

[8] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu, "Towards universal paraphrastic sentence embeddings," *arXiv preprint arXiv:1511.08198*, 2015.

[9] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," *ICLR*, 2016.

[10] Z. Cao, F. Wei, L. Dong, S. Li, and M. Zhou, "Ranking with recursive neural networks and its application to multi-document summarization." in *AAAI*, 2015, pp. 2153–2159.

[11] J. Cheng and M. Lapata, "Neural summarization by extracting sentences and words," *arXiv preprint arXiv:1603.07252*, 2016.

[12] P. Ren, F. Wei, Z. Chen, J. Ma, and M. Zhou, "A redundancy-aware sentence regression framework for extractive summarization."

[13] Y. Gong and X. Liu, "Generic text summarization using relevance measure and latent semantic analysis," in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2001, pp. 19–25.

[14] M. Kågebäck, O. Mogren, N. Tahmasebi, and D. Dubhashi, "Extractive summarization using continuous vector space models," in *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)@ EACL*, 2014, pp. 31–39.

[15] M. T. Nayeem and Y. Chali, "Extract with order for coherent multi-document summarization," *arXiv preprint arXiv:1706.06542*, 2017.

[16] W. N. Francis and H. Kucera, "The Brown Corpus: A Standard Corpus of Present-Day Edited American English," 1979, brown University Liguistics Department.

[17] J. H. Lau and T. Baldwin, "An empirical evaluation of doc2vec with practical insights into document embedding generation," *arXiv preprint arXiv:1607.05368*, 2016.

[18] R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, A. Belopolsky *et al.*, "Theano: A python framework for fast computation of mathematical expressions," *arXiv preprint*, 2016.

[19] G. Erkan and D. R. Radev, "Lexrank: Graph-based lexical centrality as salience in text summarization," *Journal of Artificial Intelligence Research*, vol. 22, pp. 457–479, 2004.

[20] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out: Proceedings of the ACL-04 workshop*, vol. 8. Barcelona, Spain, 2004.

[21] P. A. Rankel, J. M. Conroy, H. T. Dang, and A. Nenkova, "A decade of automatic content evaluation of news summaries: Reassessing the state of the art." in *ACL (2)*, 2013, pp. 131–136.

[22] K. Hong, J. M. Conroy, B. Favre, A. Kulesza, H. Lin, and A. Nenkova, "A repository of state of the art and competitive baseline summaries for generic news summarization." 2014, pp. 1608–1616.

Image Splicing Detection

Ryan Griebenow
University of Colorado, Colorado Springs
Colorado Springs, CO 80915
Email: rgrieben@uccs.edu

***Abstract*—Thus far, most research in Image Forgery Detection has concentrated on the detection and localization of specific methods of forgery using methods like patch-matching, anomaly detection, and examining residual-based local descriptors. Recent research has shown that sufficiently trained Convolutional Neural Networks can learn functions similar to those of networks trained on handcrafted features. This research focuses on combining this new knowledge with various preprocessing methods to demonstrate a proof-of-concept model.**

***Keywords*—Image Forensics, Splicing, Machine Learning, Convolutional Neural Networks, Autoencoders**

I. Introduction

Increasingly, members of society rely on digital imagery to make decisions and form opinions in the digital world. Images are used everywhere: in courtrooms as evidence of a crime, by car insurance agencies to evaluate damage after an accident, in magazines to sell products or brands. Naturally, as reliance on digital imagery grows, so, too, does the use of photo editing software such as Adobe Photoshop or the GNU Image Manipulation Program (GIMP). Using such software, users are capable of drastically changing the content of an image. Manipulations range from removing red-eye in a family portrait to completely removing people or objects. Object removal is typically done by copying some content that already exists in the image over the pixels that contain the object. The process is called Copy-Move. It is also possible to add in content from one image (or several) to another in a process called Splicing.

With free access to tools like GIMP and an internet full of free resources, the use and abuse of photo editing software has exploded. Humorously doctored images are spread across image boards and email inboxes while politically charged forgeries get blasted as news article headlines. With images being used to make decisions with heavy consequences, there exists a clear need for reliable forgery detection methods.

Within the realm of digital image forgery detection there exist many methods [1–4] for detection and localization. Some focus on emphasizing unique noise patterns within images to create a “fingerprint” of the camera that captured the image [5] while other methods attempt to identify copy-move forgeries using block-matching/PatchMatch [6]. These solutions utilize supervised machine learning on existing, labeled datasets to train a machine so that it may classify images as Pristine or Forgery. Localization of the manipulations takes place after an image is determined to be forged. With Copy-Move forgeries, patches of pixel that are copied can be found within the image and highlighted. Splicing localization can be done by detecting and highlighting a break in a boundary between the host image’s content and the foreign spliced content.

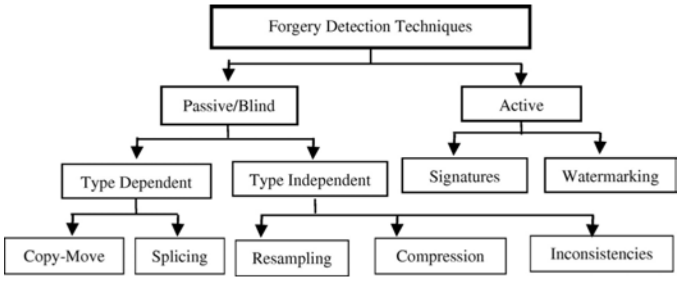


Fig. 1. Classification of image forgery detection techniques. From [4]

Typically, splicing detection involves the use of handcrafted filters and features within a neural network or other machine learning system. These features are difficult to produce with much research being done to develop newer, state-of-the-art features. It has been shown in [7] that residual-based local descriptor features can be replaced by CNNs with equal or better performance. This removes the complexity and difficulty of detecting forgeries based on handcrafted features.

Recent research by Cozzolino et al. has shown that an Autoencoder network is capable of localizing forgeries within a forged image by classifying portions of the image as pristine or forged and selectively training the network only on the correctly determined pristine portions [8]. Thus, over iterations, the network learns to reliably reproduce the pristine data while giving rise to large errors when reproducing forged data. This network was trained using a selection of handcrafted features.

It is the goal of this research to further investigate the use of a CNN in classifying spliced images by training a deep residual learning network similar to ResNET [9] but much smaller in depth. Deeper networks with many layers require proportionally large amounts of data to train on. While many image forensics datasets exist, large set are often produced automatically as in [10]. This research into splicing detection is motivated by a need for a forgery detector capable of working on realistic, high quality forgeries. Datasets containing high quality, realistic forgeries are harder to create and, in the cases of the DSO-1 and DSI-1 datasets in [11, 12], contain a scarce few examples, 200 and 50 respectively. By dividing images into patches, the number of available training samples increases significantly. These datasets are much smaller than other well known sets such as the IFS-TC dataset in [13] with 1818 images or the ImageNet database of [14] with 14,197,122 images.

Attaching an Autoencoder network to a Residual CNN will allow the Autoencoder to make use of the CNNs extracted features. Allowing the backpropagation from the Autoencoder training through to the CNN may further increase the effectiveness of the network in classifying digital images.

II. Existing Methods

Much research has been conducted in the Image Forgery field. Because of this, there exist many methods for detecting and localizing multiple different forms of image manipulation. Some, [2, 5, 6, 15] focus on splicing and copy-move detection. Other methods are more general. Fig 1. shows a brief overview of detection methods.

A. Detectable Manipulations

What follows is a general list of detectable manipulations. Many more methods exist for many more manipulations than are listed here.

1) Splicing: Cozzolino et al. have heavily investigated splicing detection and Copy Move forgery. Using an Autoencoder, Cozzolino and his team were able to localize anomalies within an image to detect the presence and location of splicing [8]. This work builds upon their research in [1]. Additional splicing research exists in [15]. Typically, splicing is found by identifying broken patterns within an image. These patterns are usually emphasized using complex, handcrafted data transformations on images prior to processing them in a neural network. By highlighting these patterns and features, networks can focus on the features that matter most when training and optimizing without having learn what to ignore.

2) Copy-Move: As mentioned, Cozzolino et al have researched Copy-Move detection and localization [2, 6]. The primary method involves analyzing small patches of an input image and scanning for duplications that are present elsewhere in the image–though perhaps rotated or scaled. Copy-Move forgeries differ significantly from splicing forgeries in that copy-move manipulations use image data from the same host image whereas splicing forgeries paste in data from other donor images. [15] also presents some research on copy-move detection.

3) Inpainting: Inpainting refers to the process of filling in portions of an image whose content has been erased. Unlike Copy-Move forgeries, which are large patches of data that are duplicated elsewhere, inpainting takes small samples from multiple locations within the same host image to fill in deleted data. These patches are much smaller than typical copy-move forgery patches and can easily fool PatchMatch algorithms. However, Liu et al. have demonstrated Discrete Cosine Transforms and ensemble learning techniques that can classify JPEG inpainting [16].

4) Seam Carving and Scaling: Seam Carving is a content-aware image resizing algorithm that involves identifying the least import "seams" of an image that can be removed for scaling purposes. Seam Carving can be used to maintain ratios and proportions within the image that are otherwise lost with basic stretching or shrinking along an image's axis. It is also possible to completely remove objects from images using seam carving techniques. Fillion, et al. have used an SVM classifier to detect the presence of content adaptive scaling in an image [17] while others use Discrete Cosine Transforms to aide in classifying input, [18].

5) Blur: Due to the nature of camera lens photography, blurring on objects out of focus is a common phenomenon. Clever forgery artists can attempt to take advantage of this by manually blurring selected sections or objects in an image. Wang, et al. have shown in [19] that it is possible to detect manually blurred edges within an image using an SVM classifier.

B. Multiple Method Manipulations

It is important to state that manipulation detection methods rely heavily on the type of manipulation present in the image. For instance, a splicing detection algorithm or network might have trouble finding Copy-Move forgeries or vice versa. For example, look at the images present in Fig. 2, where a patch-match algorithm successfully found duplicated tile images but failed to detect the obvious splicing present in the image. This image is an example forgery collected from an online Photoshop community as part of an effort to build a new dataset containing the output of multiple forgery methods and artists. This endeavour is discussed further in section III.

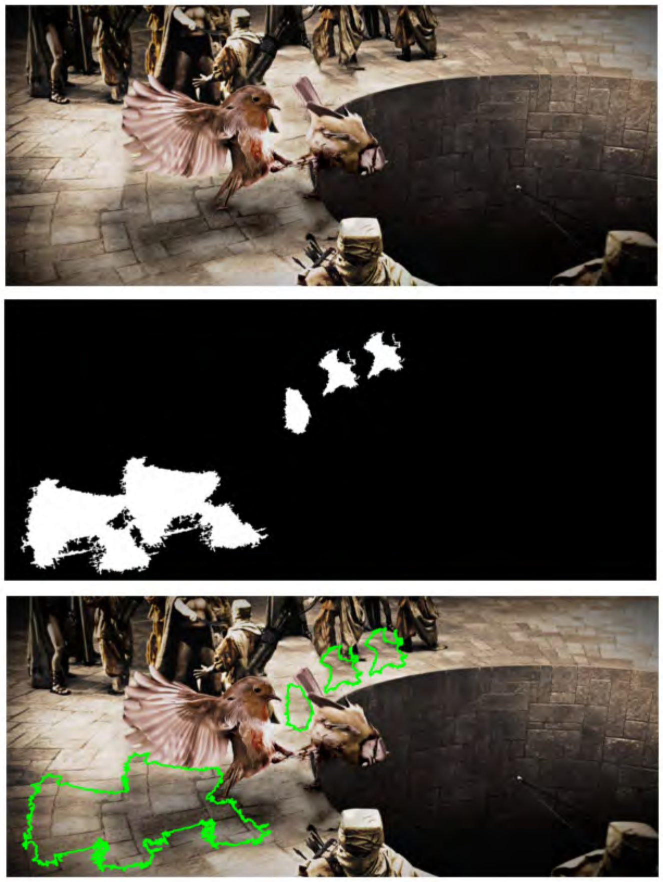


Fig. 2. An example of a patch-match algorithm used in [10] failing to detect and localize splicing within an image. Top: The input image. Middle: Generated ground truth. Bottom: Output image

C. Localization

Localization of manipulated data also depends on the type of detection and the method of forgery. For instance, Copy-Move forgeries can be detected using PatchMatch. As stated

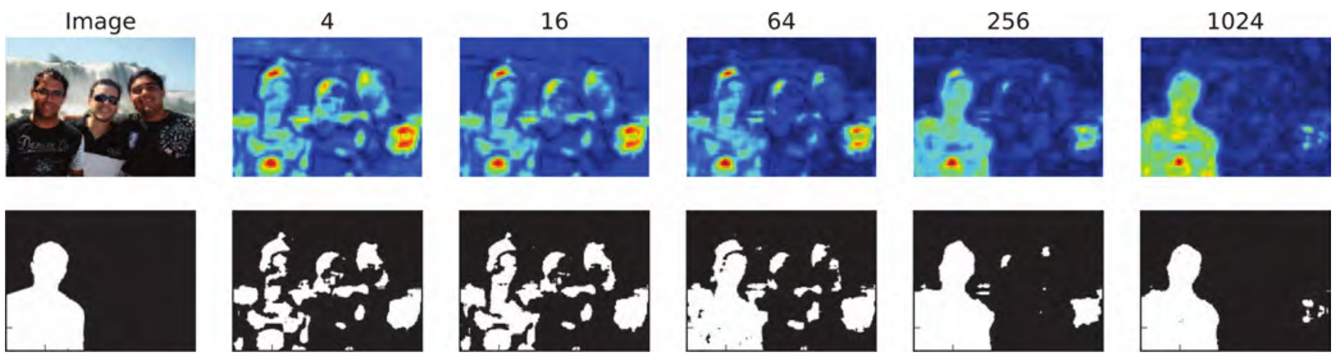


Fig. 3. An example of the Autoencoder Anomaly Detection method over several epochs. The top layer show the "heat map" of image reproduction errors. The localization of anomalies becomes more reliable as discriminative learning takes place.

previously, PatchMatch algorithms can locate patches of the data that have been copied (and perhaps rotated, or scaled) to different sections of the image and then highlight the duplicate patches in the image [6].

Splicing localization is a harder task. One method is by finding broken patterns, like broken noise residuals as in [5]. Methods like this typical involve applying some form of transformation or alteration to the data before sending it to a network in order to emphasize important features. In many cases, very specific data transformations are the key to unlocking higher accuracy in detection and localization.

Another recently proposed method [8] shows that an Autoencoder network can localize forgeries without prior training on labeled data. The Autoencoder network instead learns how to differentiate pristine pixels from forged or manipulated pixels from the image itself. By discriminatively training the Autoencoder on pristine data, it learns to reliably reproduce clean portions of the image while forged areas give rise to high errors. The pixels that give rise to high errors are considered forged. Over iteration, the Autoencoder reliably learns which pixels are original and which are likely to be forged. Fig. 3 shows an example of this process. It is important to note that the Autoencoder localization assumes forgery detection has already taken place and that the image is determined to be a forgery. The Autoencoder network itself is not capable of making such a classification.

D. Convolutional Neural Networks

Convolutional Neural Networks use convolutional layers to reduce an image down to a smaller number of features. A convolutional layer scans the entirety of and input image using a sliding window that compares small windows of the image with several different filter patterns. The values returned correlate with the likeness of the image window data to the filter patterns. CNNs usually make use of several convolutional layers where initial layers reduce input images to more general features while deeper layers learn more complex, descriptive features. We note that in [7], it has been shown that Convolutional Neural Networks can learn functions similar to many handcrafted features used in previous research experiments in computer vision. This replaces the need for newer, more targeted data transformations by relying on a neural network to learn its own functions based on the data provided.

E. ResNET

Google Inc's ResNET architecture is a deep convolutional network that incorporates residual learning. By providing a prior layers input to a later layer, without applying any transformations, latter layers can learn residual features. He et al. in [9] claim that such networks are easier to optimize and benefit from added depth.

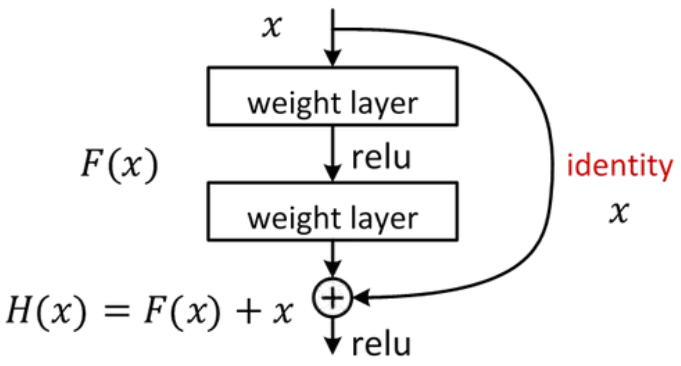


Fig. 4. An example residual learning layer from [9]. The input to the F(x) + x layer includes the previous layer's input x. The F(x) + x layer then learns residual features

III. Insolubles

Over the 10 week term of the REU program, many solutions were proposed for problems centered around image forgery detection and localization. As time was invested, many of these proposed solutions began to show signs of future complications or otherwise proved themselves to be improper solutions under the small time constraint. This section contains a brief summary for each attempted research focus and some analysis including some insight into the decisions to change focus. Each topic also offers a discussion of what future research in that topic might benefit from using or avoiding.

A. Realistic Dataset

Initially, we had focused on collecting a new dataset for training and testing. It was our opinion that existing image forgery datasets such as the IEEE IFS-TC Image Forensics Challenge dataset used in [20] and Image Manipulation Dataset [10] did not contain a broad enough range of manipulations that accurately represent what can be encountered in real

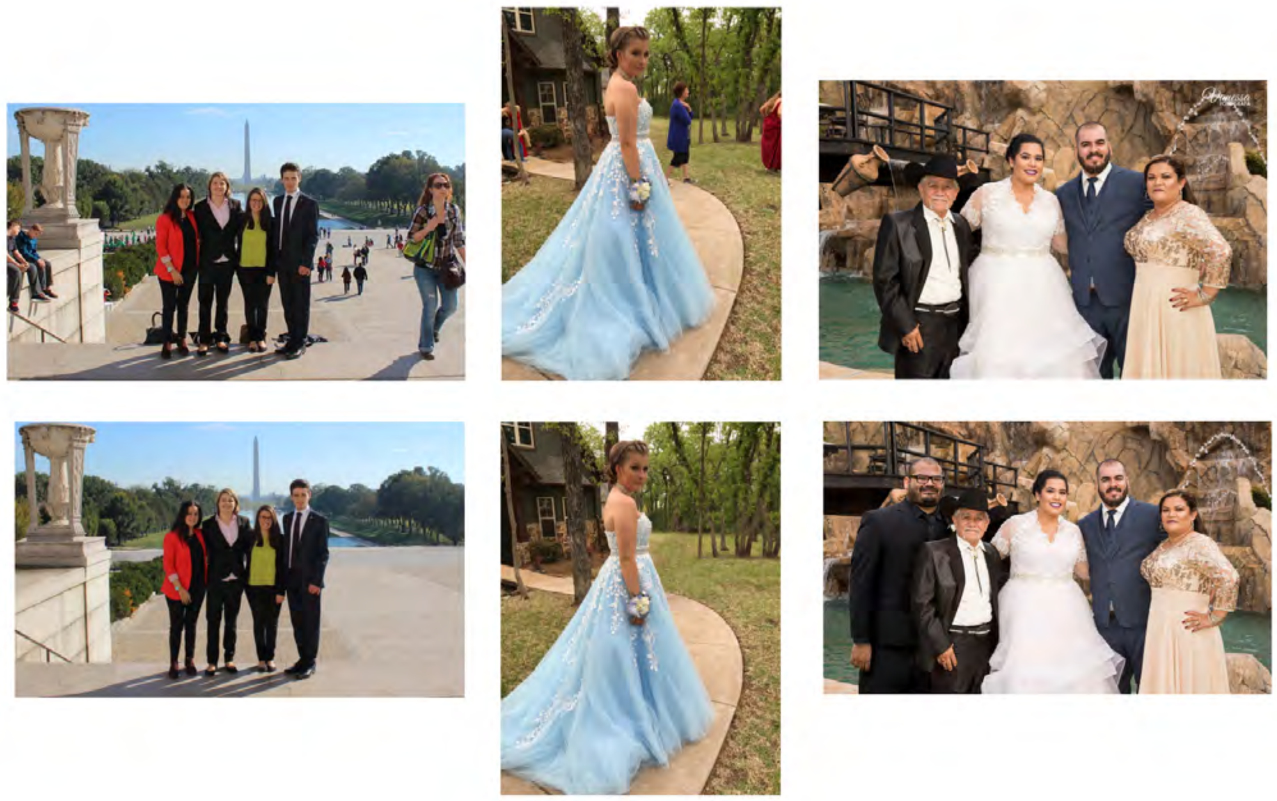


Fig. 5. Pictured are the three best submissions as determined by the author. The forgeries in the first two images proved difficult to detect using the CMFD-Framework while the last image (an example of splicing) appeared to be the most deceptive in our visual analysis. Top: Original. Bottom: Forgery

forgery cases. For example, the dataset used in [10] was automatically generated by software to create a large set of copy-move forgeries. We reasoned that because it is computer generated, it cannot possibly contain examples of all Copy-Move forgery methods. Likewise, the IEEE IFS-TC splicing dataset in [20] is comprised of images constrained to a single technique (Alpha matting) with various degrees of photorealism. Our research focus is on the detection of many different forms of splicing. Using a CNN on a dataset containing only examples of a single splicing method will result in a CNN that learns features related exclusively to that particular splicing method. It was clear that a better, more realistic dataset was needed for the CNN to learn a general set of features rather than features that describe a single forgery method.

In order to obtain a more diverse dataset, we asked several online image forgery communities to submit forgeries together with their pristine hosts. We added a financial incentive for those who where able to create a forgery that fooled current detection methods—like the CMFD-Framework software provided by the authors of [10]. A deadline for submission was set so that participants had two weeks (specifically including two weekends) from initial posting to create their forgeries. Unfortunately, by the submission deadline, we had only received a small handful of forgeries. In total, 24 forgeries were submitted by 8 unique participants over the two week period. Using the CMFD-Framework, we were partially able to detect many of the forgeries but not all. This is in part due to a lack of familiarity with the software.

Regardless, the number of submitted forgeries is not anywhere close to the number needed for our experiments, especially not for deep learning which would require a dataset several orders of magnitude larger. The submissions will instead be used in experiments on the proposed solution to gauge both the quality of forgery (difficulty in detection) and the accuracy of the trained model. More effort could be spent attempting to better incentivize the members of a wider array of forgery communities, but we deemed this an inefficient use of time and resources. In addition, the thought of writing a 6-10 page paper on the creation of a new dataset was troubling enough to stop any further work in this area.

With a longer time frame for submissions, it is likely that we would have received more submissions for review. To create an adequately sized dataset would take much longer and would require a restructuring of incentives used. By rewarding the first twenty submissions that meet the requirements, we found ourselves incentivising the speed of forgery production rather than the quality. Instead, comparing the prediction confidence between submissions or the accuracy of localization using a network might have served as a better metric of ”quality forgery” and properly incentivised participants to create quality forgeries rather than fight against the clock. Future dataset creation attempts might also benefit from involving a wider and more diverse set of online communities. A wider variation in applicants likely also leads to more variance in data and forgery techniques which can lead to interesting insights.

B. A Quicker Dataset

During the request period, we also attempted to scrape data from an online Photoshop community. The goal was to have a reasonably sized dataset made up of a large variety of forgery techniques and forgery artists. The subreddit, r/PhotoshopBattles, (a subreddit is a user created and moderated community within the larger Reddit website that is usually focussed on a topic, theme, or idea) hosts threads where the thread "topic" is a host image and all replies in the thread are doctored versions of the host image.

The image scraper successfully scrapes and organizes images by thread such that the pristine host image is associated with all forgeries and can be used to create a ground truth label. But, the scraped forgeries are not high quality nor are they necessarily representative of the types of manipulations malicious forgers might use. They consist mostly of hastily pasted splicings of the host image into humorous settings. This endeavour was abandoned shortly after the realization that collected images did not provide any benefits to a neural network over those in other currently existing networks.

It is likely that other online communities would serve as better targets of a web scraping script. A community focussed less on humor and more on the creation of high quality images would yield more appropriate data for use in forgery detection research. Finding such a community (or several) is simply a matter of exploration, discovery, and time.

C. Finetuning Pretrained Model

Many popular and effective neural network architectures are available for use online in a variety of machine learning frameworks. These models often come pretrained and ready to work, having been trained on high end graphics cards and large datasets for weeks. It was thought that a pretrained ResNET model could be fine-tuned on the submitted forgeries or on other well known datasets such as the DSO-1 and DSI-1 datasets in [11, 12].

Fine-tuning, also known as Transfer Learning, is the process of retraining a network on new data. The pretrained network weights can be used as either a fixed feature extractor or as the initialization of a new training session. A fixed feature extractor is in essence a trained CNN with the final layer, the one that produces a classification, removed or replaced. Thus, the trained network produces a high dimension vector that can then be fed to a new linear classifier. Using the pretrained network as an initializer for a new network then allowing backpropagation to continue from the classification layer all the way (or part of the way) through the CNN.

Fine-tuning saves an immense amount of time. Modern Convolutional networks can take weeks across multiple high end GPUs to fully train on a sufficiently large enough dataset. Often, fully training a new network is unnecessary if pretrained checkpoints are available for the same architecture trained on a related dataset.

After a lengthy amount of time, the team was able to successfully fine-tune a small number of models (ResNET, Google's Inception, and an InceptionResNET) on a few different datasets. It was hoped that fine-tuning the final layer of these networks might yield some meaningful results as well as offer a comparison between network architectures and between datasets. The highest accuracy reported was by an Inception ResNET V2 architecture trained on the IFS-TC dataset that achieved 74 percent accuracy. Unfortunately, the same model fine-tuned on the scraped forgeries (which were previously determined to be of no use) yielded similar results of 71 percent accuracy. If the network was having the same difficulty in detecting forgeries within two datasets of wildly contrasting forgery skill levels, the network certainly wasn't learning much that was helpful or meaningful during its fine-tuning. This discovery prompted us to reevaluate and pivot towards fully training a smaller network.

One possible reason for the poor performance of the fine-tuned networks is that the original networks were trained for 1000-way image classification. The weights and biases learned by such networks are very capable of determining the general features of, say, a dog versus a cat after sufficient training on an appropriately labeled dataset. They are not, however, trained to detect small perturbations in pixel data patterns which would require training on forged and pristine data. It is likely that the pretrained feature weights did not aide the network much during fine-tuning and did not serve as ideal initialization weights for transfer learning.

Future research in this area using transfer learning might benefit from using features extracted in earlier layers of the network. These features are more general and are less specific to the task being trained for. In the future, models pretrained for the purposes of image forensics might be available for use publicly like those available for classification.

IV. Proposed Solution

Our current research focus proposes combining the findings of [7]—that properly trained CNNs can replace the need for complicated, handcrafted features—with the localization powers of an Autoencoder as seen in [8] We hypothesize that the combined network would be capable of state-of-the-art performance after training on properly labeled data. By back-propagating the errors in the final reproduction step of the Autoencoder network all the way through the ResNET, we suspected that the proposed network will be able to learn more descriptive features than those learned in existing network architectures.

A. Meaningful Features with Patches

In [7], the authors show that a class of residual-based local descriptors can be thought of as a constrained CNN. The authors were able to train and fine-tune their network on a smaller dataset and achieve high accuracy in testing. We plan to extend this insight by feeding the network a set of image patches rather than the whole image. Patches can be compared with a identical cropping of their ground truth images to determine if the patch hosts any manipulated pixels. If any pixels within the ground truth mask are manipulated, the whole patch is considered forged.

With smaller inputs, the network has less of a need to convolve the data down to less dimensionality. This preserves more of the original structure and content of the data from the patch which we predict will aide the network in learning more meaningful features.

B. Fed to an Autoencoder

As stated, an Autoencoder architecture will be connected to the end of a CNN architecture. This way, the Autoencoder will be fed the convolved features extracted by the CNN portion of the network. The Autoencoder will encode and compress the features down into a bottleneck of neural nodes and then attempt to rebuild and decode the compressed features back into the full patches that were fed as input into the CNN. The deepest layer of the Autoencoder, once trained, can be fed to a classification layer to predict whether an input patch is forged or not. The loss function will be the squared error of the predictions. If the network predicts that the patch is a forgery, we plan to experiment with the decoding layers to localize the manipulations within the patch in a fashion similar to [8]. We predict that the features extracted by the CNN portion of the network will be more meaningful to the Autoencoder and allow for faster, easier optimization on the patches than the handcrafted features used previously. It is also suspected that the features extracted by a CNN will decrease the number of iterations needed by an Autoencoder to localize splicing forgeries.

V. Conclusion

This paper has presented a number of research focuses and an analysis of why these focuses were not producing or were not likely to produce quality results. It was shown how existing datasets do not contain within them enough examples of common forgery techniques and consequently how difficult and time consuming it is to crowd source the creation of a new dataset. It has also been shown that quickly scraped forgeries from the web also lack high forgery quality, though it is possible to create a large dataset rather quickly and easily. An overview of fine tuning has been provided along with a discussion on why the fine-tuning attempted in this research was not likely to be beneficial as our research focus, image forgery detection, is not aligned with the pre-existing networks trained for 1000-way classification. In each of these topics, suggestions were given for future research conducted in the field based on what we learned and our analysis.

Building off of the accumulated knowledge, a solution was proposed that aims to combine the feature extraction capabilities of deep learning, Convolutional Neural Networks with the localization powers of an Autoencoder. The proposed network is likely to learn features general to many types of splicing forgery when trained on an appropriately various dataset.

The final goal of this research is to demonstrate the use of a CNN and Autoencoder in image forgery detection. By using residual learning, the proposed network will hopefully be capable of detecting more meaningful and descriptive features than the handcrafted features used currently. A network of smaller than usual depth will preserve more content from the original input and hopefully allow for more meaningful

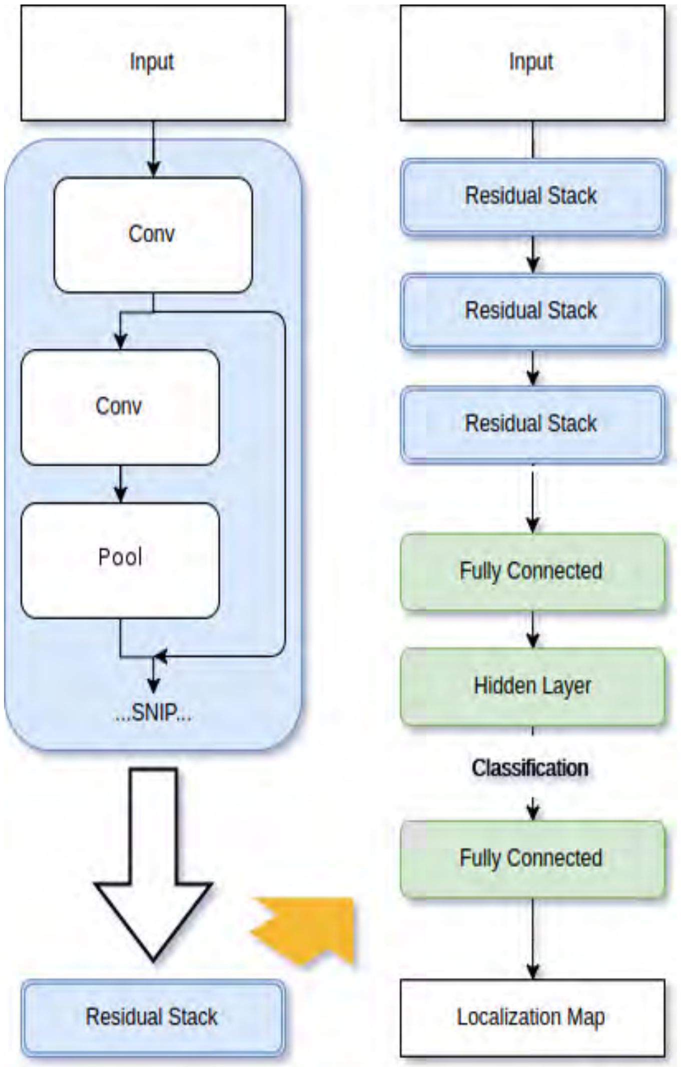


Fig. 6. A visual depiction of the proposed network. On the left, a section of a residual network is condensed into a “Residual Stack”. On the right, an example combination of Residual Stacks feeding into an Autoencoder network (in green) after classification. The Autoencoder network then produces a localization map.

associations between datapoint inside both the CNN and the Autoencoder. A smaller sized network also benefits from a reduction in the size of dataset required to train and reach optimization. The proposed residual CNN will be combined with an Autoencoder for the detection and localization of image forgery. By combining multiple architectures, it is hoped that the proposed network will be capable of state-of-the-art performance.

VI. Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant No. 1359275 and 1659788.

References

[1] D. Cozzolino, G. Poggi, and L. Verdoliva, "Splicebuster: a new blind image splicing detector," in *2015 IEEE International Workshop on Information Forensics and Security (WIFS)*, Nov. 2015, pp. 1–6. DOI: 10.1109/WIFS.2015.7368565.

[2] D. Cozzolino, D. Gragnaniello, and L. Verdoliva, "Image forgery detection through residual-based local descriptors and block-matching," in *2014 IEEE International Conference on Image Processing (ICIP)*, Oct. 2014, pp. 5297–5301. DOI: 10.1109/ICIP.2014.7026072.

[3] M. A. Qureshi and M. Deriche, "A bibliography of pixel-based blind image forgery detection techniques," *Signal Processing: Image Communication*, vol. 39, Part A, pp. 46–74, 2015, ISSN: 0923-5965. DOI: https://doi.org/10.1016/j.image.2015.08.008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0923596515001393.

[4] K. Asghar, Z. Habib, and M. Hussain, "Copy-move and splicing image forgery detection and localization techniques: a review," *Australian Journal of Forensic Sciences*, vol. 49, no. 3, pp. 281–307, 2017. DOI: 10.1080/00450618.2016.1153711. eprint: http://dx.doi.org/10.1080/00450618.2016.1153711. [Online]. Available: http://dx.doi.org/10.1080/00450618.2016.1153711.

[5] M. Chen, J. Fridrich, M. Goljan, and J. Lukas, "Determining image origin and integrity using sensor noise," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 1, pp. 74–90, Mar. 2008, ISSN: 1556-6013. DOI: 10.1109/TIFS.2007.916285.

[6] D. Cozzolino, D. Gragnaniello, and L. Verdoliva, "Image forgery detection based on the fusion of machine learning and block-matching methods," *CoRR*, vol. abs/1311.6934, 2013. [Online]. Available: http://arxiv.org/abs/1311.6934.

[7] D. Cozzolino, G. Poggi, and L. Verdoliva, *Recasting residual-based local descriptors as convolutional neural networks: an application to image forgery detection*, 2017. eprint: arXiv:1703.04615.

[8] D. Cozzolino and L. Verdoliva, "Single-image splicing localization through autoencoder-based anomaly detection," in *2016 IEEE International Workshop on Information Forensics and Security (WIFS)*, Dec. 2016, pp. 1–6. DOI: 10.1109/WIFS.2016.7823921.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[10] V. Christlein, C. Riess, J. Jordan, C. Riess, and E. Angelopoulou, "An evaluation of popular copy-move forgery detection approaches," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 6, pp. 1841–1854, Dec. 2012, ISSN: 1556-6013. DOI: 10.1109/TIFS.2012.2218597.

[11] T. J. d. Carvalho, C. Riess, E. Angelopoulou, H. Pedrini, and A. d. R. Rocha, "Exposing digital image forgeries by illumination color classification," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 7, pp. 1182–1194, Jul. 2013, ISSN: 1556-6013. DOI: 10.1109/TIFS.2013.2265677.

[12] T. Carvalho, F. A. Faria, H. Pedrini, R. da S. Torres, and A. Rocha, "Illuminant-based transformed spaces for image forensics," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 4, pp. 720–733, Apr. 2016, ISSN: 1556-6013. DOI: 10.1109/TIFS.2015.2506548.

[13] L. Verdoliva, D. Cozzolino, and G. Poggi, "A feature-based approach for image tampering detection and localization," in *2014 IEEE International Workshop on Information Forensics and Security (WIFS)*, Dec. 2014, pp. 149–154. DOI: 10.1109/WIFS.2014.7084319.

[14] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015. DOI: 10.1007/s11263-015-0816-y.

[15] Y. Rao and J. Ni, "A deep learning approach to detection of splicing and copy-move forgeries in images," in *2016 IEEE International Workshop on Information Forensics and Security (WIFS)*, Dec. 2016, pp. 1–6. DOI: 10.1109/WIFS.2016.7823911.

[16] Q. Liu, A. H. Sung, B. Zhou, and M. Qiao, "Exposing inpainting forgery in jpeg images under recompression attacks," in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Dec. 2016, pp. 164–169. DOI: 10.1109/ICMLA.2016.0035.

[17] C. Fillion and G. Sharma, "Detecting content adaptive scaling of images for forensic applications," in *IS&T/SPIE Electronic Imaging*, International Society for Optics and Photonics, 2010, 75410Z–75410Z.

[18] A. Sarkar, L. Nataraj, and B. S. Manjunath, "Detection of seam carving and localization of seam insertions in digital images," in *Proceedings of the 11th ACM Workshop on Multimedia and Security*, ser. MM&#38;Sec '09, Princeton, New Jersey, USA: ACM, 2009, pp. 107–116, ISBN: 978-1-60558-492-8. DOI: 10.1145/1597817.1597837. [Online]. Available: http://doi.acm.org/10.1145/1597817.1597837.

[19] J. Wang, G. Liu, B. Xu, H. Li, Y. Dai, and Z. Wang, "Image forgery forensics based on manual blurred edge detection," in *2010 International Conference on Multimedia Information Networking and Security*, Nov. 2010, pp. 907–911. DOI: 10.1109/MINES.2010.193.

[20] D. Cozzolino, D. Gragnaniello, and L. Verdoliva, "Image forgery localization through the fusion of camera-based, feature-based and pixel-based techniques," in *2014 IEEE International Conference on Image Processing (ICIP)*, Oct. 2014, pp. 5302–5306. DOI: 10.1109/ICIP.2014.7026073.

Learning to Detect and Classify Forgeries of Digital Images in Open-Set Recognition

Sina Masoumzadeh
Computer Science
University of Colorado Colorado Springs
Email: smasoumz@uccs.edu

***Abstract*—In this era that software's are getting more advanced that are being used to edit images. Applications Like Adobe Photoshop have a lot of benefits for modifying personal photographs, the downside of this kind of applications performed for forgery.There are copy/move and splicing that is used most often to make forged images.This research we want to try to merge two of the specific techniques that are for detecting copy/move and splicing and trying to apply that to machine learning in an image to make the probability higher and to make sure the detection has a higher reliability. In addition to that, we also want to add and classifications, benign in this research.**

***Keywords*—Digital Images, Forgery, Fusion, Machine Learning, Benign, Forensic**

I. Introduction

Images can have a high impact in different categories of daily needs like in social media, military, crime, historical events, machine learning. The originality of images is necessary trying to use them as examples that do define above. Images can be a critical evidence to identifying the truth of a claim or even a crime.

In this digital era, it is harder to detect forgery because tracking them might not be comfortable with naked eyes or the fingerprint left behind it is impossible was traced with all these techniques.There are also some techniques to identify what camera was used to detect if the pattern of the image matches the pattern of that picture.[1]

In this era that we are living, there are a lot of advanced applications that can modify images with different kind of methods, for example, an advanced cloning or splicing that can make it impossible to detect with naked eyes.There are some ways to help us identify if based on that analysis, for example, Discrete Cosine Transform(DCT), Error Level Analysis(ELA), Gio Metric distortion, Photo Response Non-uniformity(PRNU).[2] In most of the research that has issued, they have used fusion to add some of these techniques trying to merge their result to have a better probability to find better ways to add them to the learnings. One of the classifications were not mentioned often in previous studies, benign in images. forensic.[3]

II. Problem Definition

The previous works have their errors or vulnerabilities. When we look at the algorithms, they are not as reliable as the ones that have few techniques fused will make the detection more accurate than the individual aspect of them. In the past, there is not much of research in classifying benign, a class that has changes in the image like resizing or changing its contrast. Moreover, these pictures are still in its original condition; unspoiled.[5] The absence of this classification can make our techniques to consider them as a class of forgery that will cause it not to have the original condition as evidence. The definition of the original condition that we are using as benign

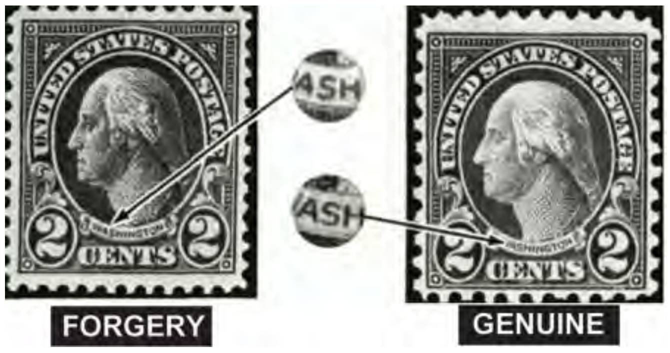


Fig. 1. It is an image of a stamp that the forged one is on the left, and the original image is on the right.It is not easy to detect with naked eyes.

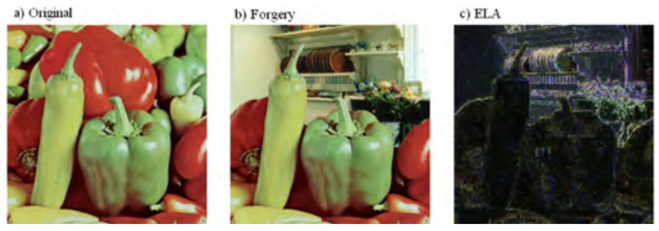


Fig. 2. This figure contains three images the original, forged,forged in ELA .Like you can see the forged area in the ELA is easy to detect for machines.[4]

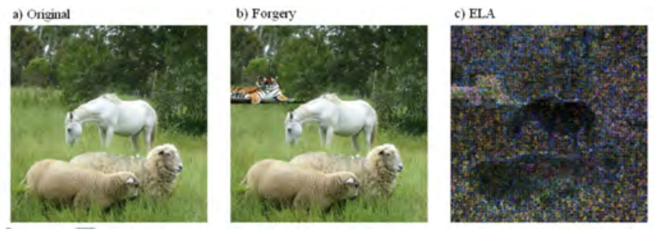
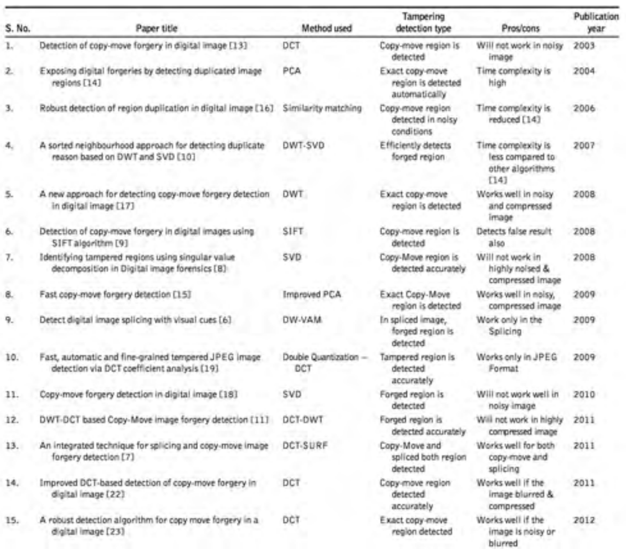


Fig. 3. This figure contains three images the original, forged, forged in ELA.It is hard for the machines to detect the forged parts because of the high-level spatial characteristics of the image such as vegetation increase error in the entire image.[4]



S. No.	Paper title	Method used	Tampering detection type	Pros/cons	Publication year
1.	Detection of copy-move forgery in digital image [13]	DCT	Copy-move region is detected	Will not work in noisy image	2003
2.	Exposing digital forgeries by detecting duplicated image regions [14]	PCA	Exact copy-move region is detected automatically	Time complexity is high	2004
3.	Robust detection of region duplication in digital image [16]	Similarity matching	Copy-move region detected in noisy conditions	Time complexity is reduced [14]	2006
4.	A sorted neighbourhood approach for detecting duplicate reason based on DWT and SVD [10]	DWT-SVD	Efficiently detects forged region	Time complexity is less compared to other algorithms [14]	2007
5.	A new approach for detecting copy-move forgery detection in digital image [17]	DWT	Exact copy-move region is detected	Works well in noisy and compressed image	2008
6.	Detection of copy-move forgery in digital images using SIFT algorithm [9]	SIFT	Copy-move region is detected	Detects false result also	2008
7.	Identifying tampered regions using singular value decomposition in Digital image forensics [8]	SVD	Copy-Move region is detected accurately	Will not work in highly noised & compressed image	2008
8.	Fast copy-move forgery detection [15]	Improved PCA	Exact Copy-Move region is detected	Works well in noisy, compressed image	2009
9.	Detect digital image splicing with visual cues [6]	DW-VAM	In spliced image, forged region is detected	Work only in the Splicing	2009
10.	Fast, automatic and fine-grained tempered JPEG image detection via DCT coefficient analysis [19]	Double Quantization – DCT	Tampered region is detected accurately	Works only in JPEG Format	2009
11.	Copy-move forgery detection in digital image [18]	SVD	Forged region is detected	Will not work well in noisy image	2010
12.	DWT-DCT based Copy-Move image forgery detection [11]	DCT-DWT	Forged region is detected accurately	Will not work in highly compressed image	2011
13.	An integrated technique for splicing and copy-move image forgery detection [7]	DCT-SURF	Copy-Move and spliced both region detected	Works well for both copy-move and splicing	2011
14.	Improved DCT-based detection of copy-move forgery in digital image [22]	DCT	Copy-move region detected accurately	Works well if the image blurred & compressed	2011
15.	A robust detection algorithm for copy move forgery in a digital image [23]	DCT	Exact copy-move region detected	Works well if the image is noisy or blurred	2012

Fig. 4. This is a diagram of the techniques that were used to detect a different kind of digital image forgery in past and present.
[2]

means an digital image that can define as an image that still holds its content.

If it is forged or not, show the results as unknown rather a weak decision, giving an answer that has a low reliability but the machine will give a precise answer about its detection can bring a lot of confusion and miss leading in the future.[6] An open set is critical in other fields like medicine that needs it gives a very accurate output or if the detection does not reliably provide an output of unknown. Machines should learn the scope of what they learned, so that helps them to answer ”I do not know” if they have not seen an example related to the Supervised Learning. Learning.[7]

III. Related Work

There is a research about using ELA features and try to detect if this features can utilize for image forgery detection.Their work had the result that based on the difference of the error level related to the original pixel.[8] The errors were related to compression loss. If the difference is small that means the chance of forgery was low.[9] The difference was significant, the chance that that image will be considered as forged.There were few Voluntaries in this research paper. There were not any numeric values mentioned in the article for comparison.[4] Some works tried to use linear BitMap image format(LBM) for encoding the micro-edge pattern and DCT which is used to encode the frequency content.[10] Applying and them to detect copy/move and splicing together to have a more useful technique beside implementing it with Support Vector Machine(SVM).[11]

IV. Proposed Solution

In this Research, we want to add a class called benign based on ELA features and try to use it in digital image forgery detection. ELA is Error Level Analysis is the analysis of compression artifacts in digital data with lossy compression such as JPEG. Fuse the discovered algorithm for benign class

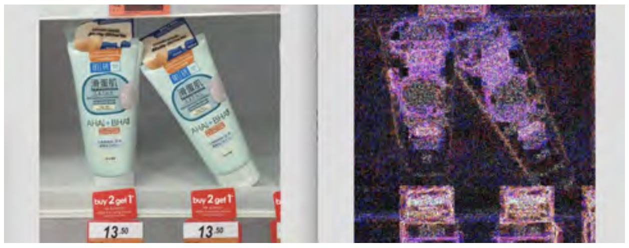


Fig. 5. There are two images the one on the left is the original image and the one on the left it is the transformation of original to ELA.

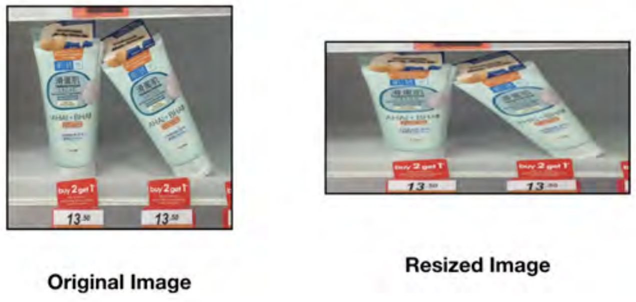


Fig. 6. The left image is the original image; the right image has been the modified in the size of the image that will be one of our sub-classes of benign class.

with the previous forgery detection for splicing, to get a forgery detection technique for the three class.1)Original 2)Forged 3)Benign.

A. Benign

Based on the definition of benign a gentle disposition, The term benign is mostly relevant in medical fields, like detecting if a tumor is cancerous or not. [12] There is not much of a definition in Image forgery field.When an image possesses some changes will not include as forged because it contains its values as evidence. We can use some tools to modify the size of the image or even modify the contrast of it, and it still keeps its values as an original image. We want to add this class that how to use learning methods so that the machine can detect the difference between a forged and a benign Image.

B. Unknown

Images with classifications that were not in the scope of the learning are going to identify as unknown classification.So this is going to be an algorithm of open-set that does not have boundaries after it is out of the scope of the defined classes. [13]

C. Technique

First of all, we want to add the benign class to the previous techniques. We want to use some other researched technique using the fusion method so we can get a more accurate result and use each technique based on their unique modification.[14][15] For the open-set algorithm, we will use the suggested version of Learning Support Vector Machine(SVM) that is Weibull-calibrated Support Vector

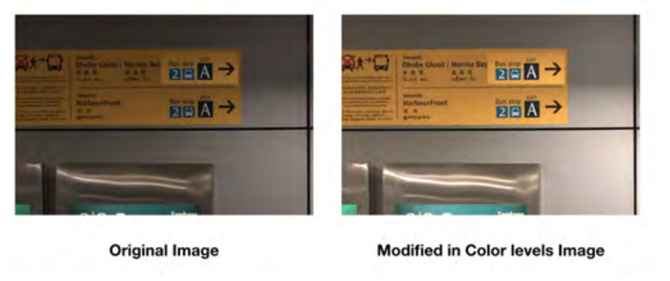


Fig. 7. The left image is the original image; the right image has been the modified in the level color of the image that will be one of our sub-classes of benign class.

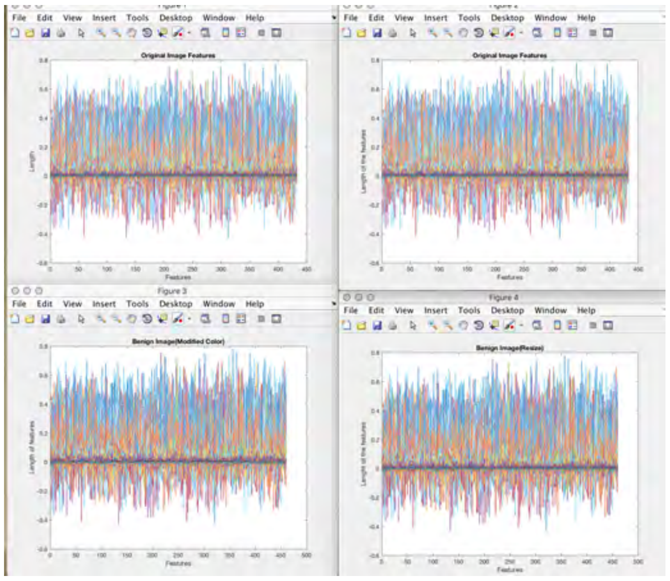


Fig. 8. There are four plots from top left; the original image top right is the forged image and left down is the benign modified color image and the last image right down is the benign re-sized image. These plots show the number of features and the length of them based on the scale between -1 and 1 ([-1,1]).

Machine(W-SVM). W-SVM combines the useful properties of statistical extreme value theory for score calibration with one-class and binary support vector machines.[16]

V. Results

We have added a benign class with two sub classes 1) Re-size 2) Color level adjustment to a previous data set that is 50 images, that include original and forged images. The most major algorithms are for detecting copy/move and splicing in digital images.They only look at pieces of the image and try to find an algorithm that can detect those changes in the image that mostly don't cover the whole image. The reason why we cannot use those algorithms is that we need an algorithm that looks at the entire image and analyzes it instead of parts of interest. So we were seeking to find some algorithm that can distinguish the class original and benign.The short period for this research paper left a lot of tests that were not finished to be part of the future work. [17]

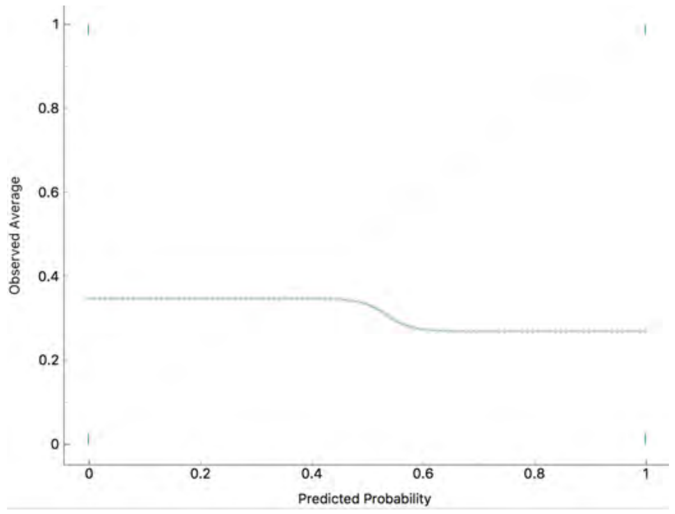


Fig. 9. This is a graph of the data set of original images that (x,y), the x is the observed average, the y predicted probability based on the features.

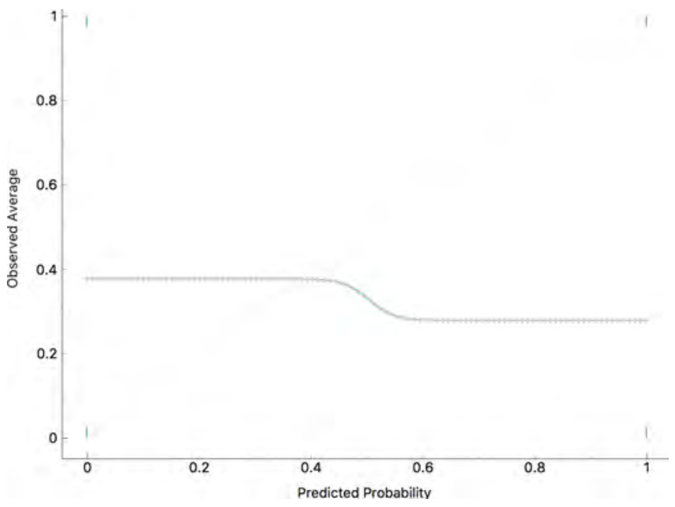


Fig. 10. This is a graph of the data set of forged images that (x,y), the x is the observed average, the y predicted probability based on the features.

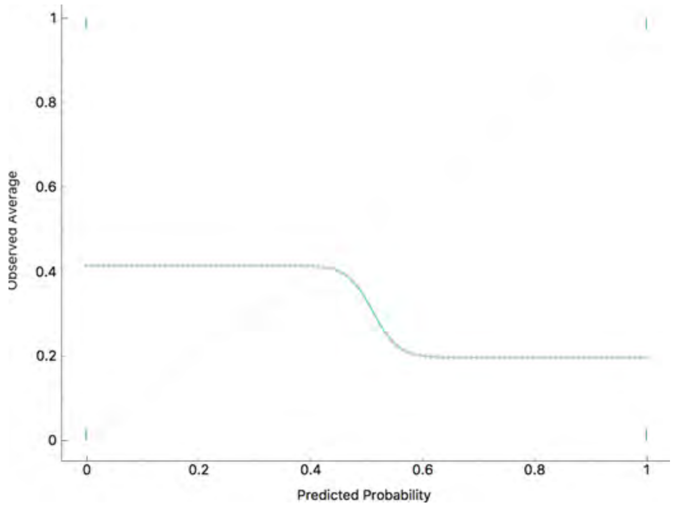


Fig. 11. This is a graph of the data set of benign images that (x,y), the x is the observed average, the y predicted probability based on the features.

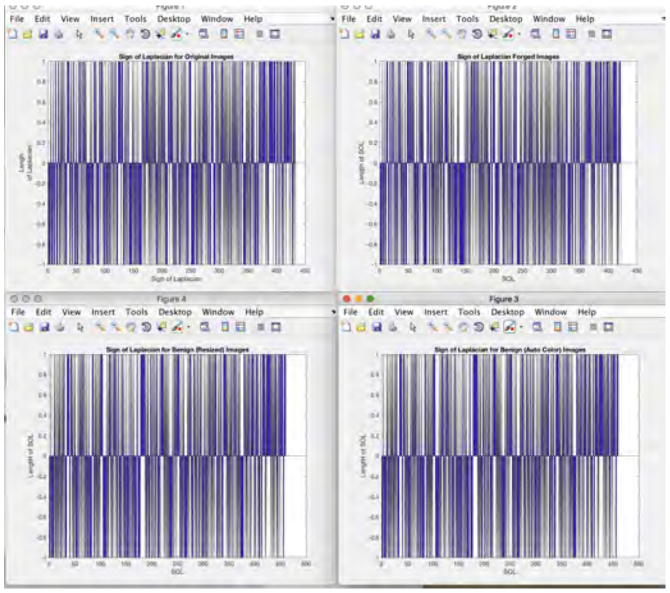

Sign of Laplacian for Original Images
Sign of Laplacian Forged Images
Sign of Laplacian for Benign (Resized) Images
Sign of Laplacian for Benign (Auto Color) Images
Length of SOL
SOL

Fig. 12. Four plots are based on Sign of Laplace-Gaussian from top left; the original image top right is the forged image and left down is the benign re-sized, and the last image right down is the benign modified color image. scale between -1 and 1 ([-1,1]).

After extracting three classes feature individually then tried Laplace-Gaussian Operator(LOG) that is edge detector for edge detection for image forgery and tried to make some comparison based on the LOG of these three classes.[18] Trying a different kind of image feature extraction to compare, if those features can be used for defining our classes for machine learning digital image forgery detection. After training the machine and getting models and testing them using k-fold cross validation the results were not reliable based on the variations. The most recent ones are in figures 14,15,16. In figure 16 we can see that there are three classes and the predictions for detecting the benign are 28.9 percent also showing benign as forged 40 percent and benign as original 31.1 percent. It gives us the understanding that based the extracted features and ten fold cross validation the benign is mostly comprising detected as forged image. The short period a lot of testings was not done, and that will be part of the future work.

VI. Future Work

For the next period of this research, we are going to experiment more with these features and try to find a technique in classifying the benign class. The next step is to add open-set algorithm by using W-SVM. Try to fuse two technique to get more accurate result in forgery detection. Finally, apply all these steps for our supervised machine learning.

VII. Conclusion

A. Benign Detection

If adding the benign class in image forgery detection that has not discussed in previously researched scope will give us results with high accuracy sometimes can save time or even help to identify what has changed but still good as its original version.

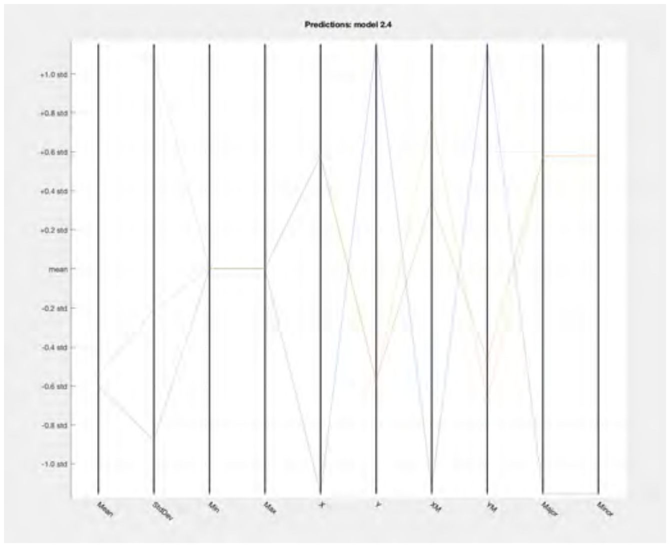


Fig. 13. This is a graph of the three class based on different features.The light orange line is the original class, the other orange line is forged class, and the blue line is benign class.

B. Unknown

Also, predict the results as unknown because it was not in the scale of its supervised learning.In that case, we expect the machine to print ”I do not know” for the image forgery detection.

C. Fusion Result

We want to see if the result of the two previous techniques Error Level Analysis(ELA) and (Singular Value Decomposition)SVD can give more accurate results in detecting the four classes, Original, Forged, Benign, Unknown.

VIII. Acknowledgement

A. Founder of this program

I want to thank the National Science Foundation for funding Research Experience for Undergraduate(REU) program with award number:1359275 and 1658788.Thanks to Program Manager William Bainbridge and Wendy Nilsen.

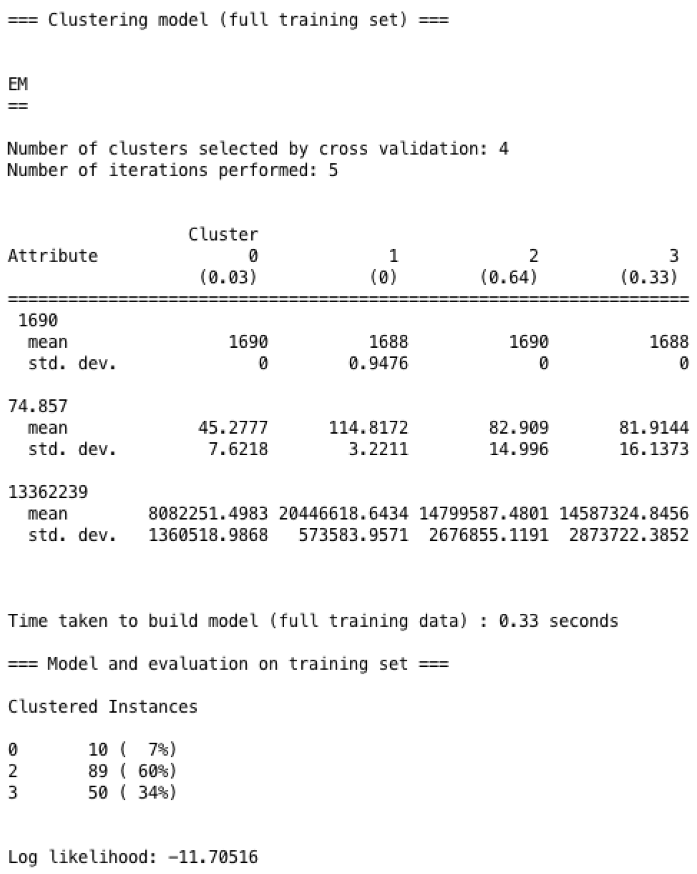
B. Team of REU

Thanks to Dr.Jugal Kalita as the Principal Investigator. Thanks to my mentor in this research Dr.Terry Boult.I have learned a lot from the REU team specially Dr.Terry Boult.

References

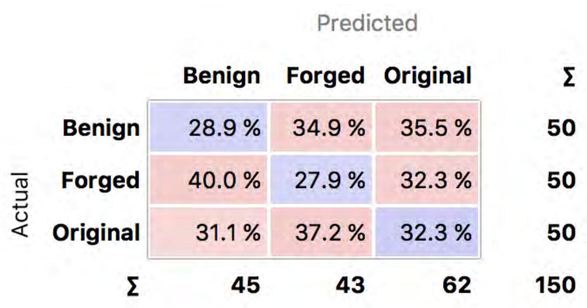
[1] M. Chen, J. Fridrich, M. Goljan, and J. Lukás, “Determining image origin and integrity using sensor noise,” *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 1, pp. 74–90, 2008.

[2] Y. Zhang, L. L. Win, J. Goh, and V. L. Thing, “Image region forgery detection: A deep learning approach,” in *Proceedings of the Singapore Cyber-Security Conference (SG-CRC) 2016: Cyber-Security by Design*, IOS Press, vol. 14, 2016, p. 1.



```
=== Clustering model (full training set) ===


EM
==

Number of clusters selected by cross validation: 4
Number of iterations performed: 5


              Cluster
Attribute           0           1           2           3
               (0.03)         (0)      (0.64)      (0.33)
=============================================================
 1690
  mean           1690        1688        1690        1688
  std. dev.         0      0.9476           0           0

74.857
  mean        45.2777    114.8172      82.909     81.9144
  std. dev.    7.6218      3.2211      14.996     16.1373

13362239
  mean   8082251.4983 20446618.6434 14799587.4801 14587324.8456
  std. dev. 1360518.9868  573583.9571 2676855.1191 2873722.3852



Time taken to build model (full training data) : 0.33 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       10 (  7%)
2       89 ( 60%)
3       50 ( 34%)


Log likelihood: -11.70516
```

Fig. 14. This is results from data with four cross validation with the result of three clusters, five iterations with a different prediction for each cluster.



Actual \ Predicted	Benign	Forged	Original	Σ
Benign	28.9 %	34.9 %	35.5 %	50
Forged	40.0 %	27.9 %	32.3 %	50
Original	31.1 %	37.2 %	32.3 %	50
Σ	45	43	62	150

Fig. 15. This is a diagram that how did the trained model do based on the extracted feature and labeling the data with three classes. 1) Original, 2) Forged, 3) Benign.

[3] T. Huang and H. Koller, "Coding of multilevel graphics," *IEEE Transactions on Communications*, vol. 23, no. 6, pp. 598–606, Jun. 1975, ISSN: 0090-6778. DOI: 10.1109/TCOM.1975.1092857.

[4] D. C. Jeronymo, Y. C. C. Borges, and L. dos Santos Coelho, "Image forgery detection by semi-automatic wavelet soft-thresholding with error level analysis," *Expert Systems with Applications*, 2017.

[5] H. Li, W. Luo, X. Qiu, and J. Huang, "Image forgery localization via integrating tampering possibility maps," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 5, pp. 1240–1252, 2017.

[6] M. C. Stamm and K. R. Liu, "Forensic detection of image manipulation using statistical intrinsic fingerprints," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 3, pp. 492–506, 2010.

[7] A. Kashyap, R. S. Parmar, M. Agrawal, and H. Gupta, "An evaluation of digital image forgery detection approaches," *arXiv preprint arXiv:1703.09968*, 2017.

[8] V. Schetinger, M. Iuliani, A. Piva, and M. M. Oliveira, "Digital image forensics vs. image composition: An indirect arms race," *arXiv preprint arXiv:1601.03239*, 2016.

[9] T. S. Gunawan, S. A. M. Hanafiah, M. Kartiwi, N. Ismail, N. F. Za'bah, and A. N. Nordin, "Development of photo forensics algorithm by detecting photoshop manipulation using error level analysis," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 7, no. 1, 2017.

[10] K. Asghar, Z. Habib, and M. Hussain, "Copy-move and splicing image forgery detection and localization techniques: A review," *Australian Journal of Forensic Sciences*, vol. 49, no. 3, pp. 281–307, 2017.

[11] A. Alahmadi, M. Hussain, H. Aboalsamh, G. Muhammad, G. Bebis, and H. Mathkour, "Passive detection of image forgery using dct and local binary pattern," *Signal, Image and Video Processing*, vol. 11, no. 1, pp. 81–88, 2017.

[12] N. A. Ofei-Tenkorang, R. V. Kanj, and L. L. Breech, "Benign tumors masquerading as malignant: A case of sclerosing stromal tumor of the ovary in an adolescent," *Journal of Pediatric and Adolescent Gynecology*, vol. 30, no. 2, p. 316, 2017.

[13] W. J. Scheirer, L. P. Jain, and T. E. Boult, "Probability models for open set recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, vol. 36, 11 Nov. 2014.

[14] W. J. Scheirer, L. P. Jain, and T. E. Boult, "Probability models for open set recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2317–2324, Nov. 2014, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2014.2321392.

[15] B. Bigdeli and P. Pahlavani, "Quad-polarized synthetic aperture radar and multispectral data classification using classification and regression tree and support vector machine–based data fusion system," *Journal of Applied Remote Sensing*, vol. 11, no. 1, pp. 016 007–016 007, 2017.

[16] N. Bonneel, B. Kovacs, S. Paris, and K. Bala, "Intrinsic decompositions for image editing," in *Computer Graphics Forum*, Wiley Online Library, vol. 36, 2017, pp. 593–609.

[17] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[18] Z. Zhang, Z. Yu, and B. Su, "Detection of composite forged image," in *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, vol. 11, Oct. 2010, pp. V11-572-V11-576. DOI: 10.1109/ICCASM.2010.5623140.

Handling Unbalanced Data in Deep Image Segmentation

Harriet Small
Brown University
harriet_small@brown.edu

Jonathan Ventura
University of Colorado, Colorado Springs
jventura@uccs.edu

***Abstract*— We approach the problem of training Convolutional Neural Networks (CNNs) for image segmentation tasks that involve unbalanced data—meaning that some of those classes we seek to identify and label occur with significantly less frequency than other classes represented in the dataset. We investigate alternative sampling strategies intended to increase the accuracy of the learned model itself and neutralize misclassifications arising from the unbalanced nature of the training data, and examine their efficacy in comparison to random sampling.**

Keywords: class imbalance, image segmentation, convolutional neural networks, machine learning

I. INTRODUCTION

One pervasive challenge in the field of deep image segmentation is the unbalanced distribution of classes in much training data [7], [8]. If pixels corresponding to a particular "majority" label are far more numerous than pixels of one or more "minority" class, the rarity of the "minority" class in the training data inhibits accurate learning and labeling, as the learned model will tend to classify most pixels as members of the "majority" class. As class imbalance in a data set increases, the performance of a neural net trained on that data has been shown to decrease dramatically [6].

The segmentation of MRI images is one notable application of deep learning in which such a class imbalance exists; in a typical MRI image of a brain tumor, the volume of healthy brain tissue is significantly greater than the volume of cancerous tissue [4]. For the purposes of this paper, we trained our neural networks on the BraTS Challenges 2013 dataset, which is comprised of MRI images of brain tumors.

Using these MRI images, we explore techniques for surmounting learning obstacles introduced by unbalanced training data. In particular, our focus is on modifying the procedure by which we sample batches to train a Convolutional Neural Network (CNN) intended to classify unbalanced data. We compare the performance of random sampling with two alternatives: a sampling protocol that generates batches containing each class in equal proportion, and a second protocol which re-introduces incorrectly classified (and borderline correctly-classified) samples from prior epochs into the batches for the current epoch. We evaluate the efficacy of these three methods by examining their effect on the correct labeling of small tumor substructures.

II. PRIOR RESEARCH

A variety of attempts to rectify the class imbalance problem have been made. In a survey study, López et al. identified

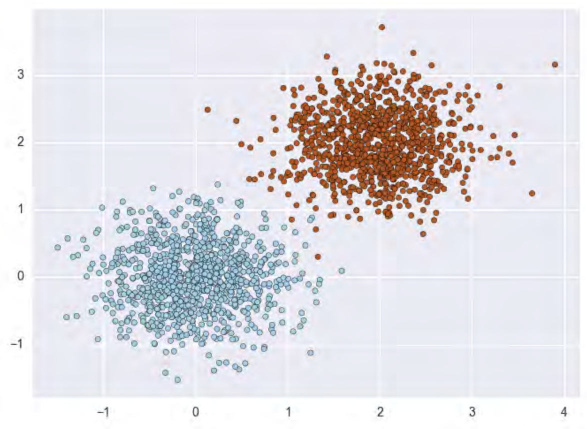


Fig. 1. A typical example of a balanced data distribution. Note that there are approximately the same number of examples of the red and blue classes. Compare with unbalanced distribution in next image. Source: https://svds.com/learning-imbalanced-classes/

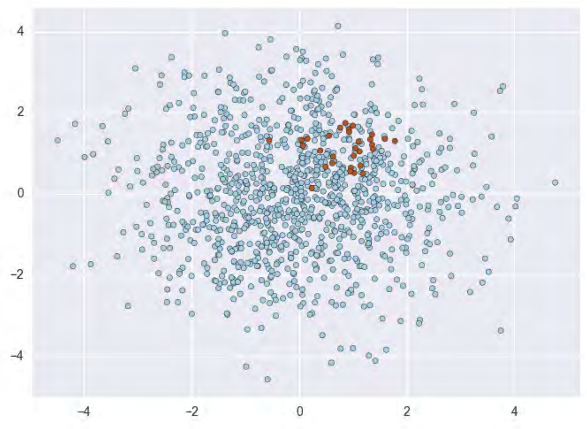


Fig. 2. An unbalanced data distribution; note that the vast majority of samples are of the blue class, and that there are comparatively few red examples. Source: https://svds.com/learning-imbalanced-classes/

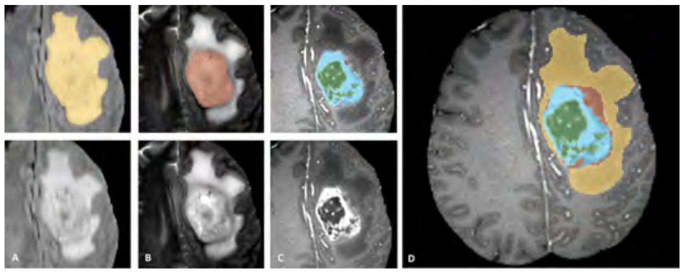


Fig. 3. A brain tumor image from BraTS annotated with four types of diseased tissue. Note that the number of pixels representing tumor tissue is much smaller than the total number of pixels [3].

three notable types of solutions: modification of data prior to learning via oversampling or undersampling, modification of the learning process itself, and application of cost-sensitive learning techniques, which weigh the relative "costs" of misclassification for each class against each other [8].

One notable approach—which falls into the first of the above categories—is the Synthetic Minority Over-Sampling Technique, or SMOTE. This technique involves generating synthetic samples of the minority class to train on, thus reducing the class imbalance by artificially inflating the size of the minority class itself [9]. This strategy, when tested alongside undersampling of the majority class, was shown to improve the performance of the trained model. Our methodology also focuses on modifying the class distribution in the dataset, although we will use only data from the original set rather than replicating additional minority samples.

A considerable amount of prior research has focused on the application of cost-sensitive learning techniques to the class imbalance problem. Often, the real-world misclassification cost of a minority sample is greater than the misclassification cost of a majority sample; when identifying a rare disease, for example, a false positive has the potential to be less damaging to the patient than a false negative. Researchers have incorporated this concern into learning algorithms by modifying the loss function at the center of the learning process to overvalue classification mistakes on the minority class, thus emphasizing the correct classification of minority samples at the expense of identifying the majority class. This type of cost-sensitive technique, while not a part of our approach, is certainly relevant in the area of tumor segmentation; the misclassification of cancerous tissue as healthy is far more costly to a patient than the misclassification of healthy tissue as cancerous [8].

One sampling-based attempt to counteract the negative effects of an imbalanced dataset was presented by Felzenszwalb et al. in their work on object detection [1]. Their dataset consisted of images with large amounts of negative space with interspersed objects; the relative rarity of the objects themselves demanded a nuanced approach. Their technique, "hard negative mining", involved identifying those examples of the majority class—the background—which the current classifier was not correctly labeling, and reintroducing those examples for further training. Focusing on "harder" negative examples in this manner allowed them to increase the fraction of minority samples used during the training process without sacrificing performance on the majority class.

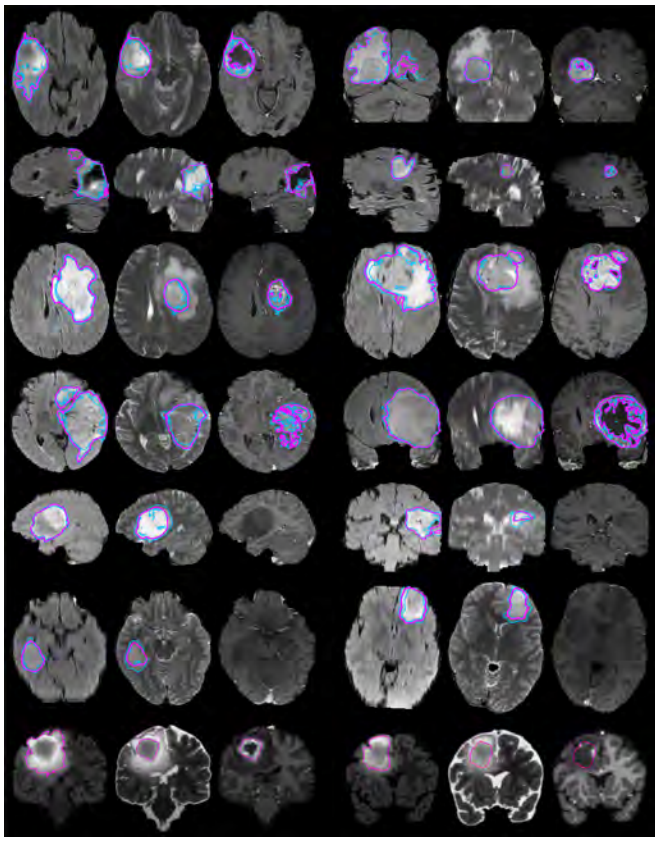


Fig. 4. A small subset of the MRI images from the BraTS 2013 Challenge Dataset, with expert annotations of the four tumor substructures shown in purple and blue. Image from [3].

Initially, our intent was to bring "hard negative mining" to bear on our own classifier, but experimentation revealed that increasing the fraction of minority class samples in training batches did not reduce accuracy on the majority class, and thus this corrective process was not required. However, we did attempt a modified version of the strategy (targeting minority classes rather than simply the majority class), an approach which is described in more detail in a later section of this paper.

III. EXPERIMENTAL DATA

For the purposes of our experimentation, we use the non-synthetic data from the 2013 edition of the Multimodal Brain Tumor Image Segmentation (BraTS) Challenge. This data set is a series of 65 MRI images of brain tumors which must be segmented into healthy tissue and four differing types of cancerous tissue herein referred to as tumor substructures. The substructures labeled in the training data are as follows:

- Edema
- Non-Enhancing (Solid) Core
- Necrotic (Fluid-Filled) Core
- Non-Enhancing Core

The challenge dataset is a typical example of an unbalanced class problem; the images in it contain a disproportionate number of healthy tissue examples and only small areas of cancerous tissue. See Fig. 4 for the subset of the challenge images themselves. A more detailed description of the dataset can be found in Havaei et al. [4].

IV. Methodology

In order to determine the optimal sampling method for this unbalanced dataset, we trained three separate convolutional neural networks with consistent architecture. The first was trained using random sampling, and the second and third using our modified sampling techniques. We then compared their performance on the identification and segmentation of minority classes to evaluate the sampling methods themselves.

During each training epoch, we refined the parameters of our CNNs using a series of batches of the training data, each consisting of a number of square sections from training images, herein referred to as patches. This type of mini-batch sampling—taking a few pixels (or, in our case, patches) from each of a diverse set of training images—has been demonstrated effective in pixel-labeling problems such as edge-detection and image segmentation [5]. This strategy reduces the computational load of processing each batch by taking advantage of the dependencies between neighboring pixels; adjacent pixels tend to have similar surroundings and therefore including many neighboring pixels in a batch is redundant [5]. Each of the networks discussed below was trained for exactly 50 epochs, each consisting of 1000 individual batches of training data, each containing 120 image patches that were 64 pixels square.

V. Evaluation of Results

Metrics for evaluating the correctness of models trained and tested on unbalanced data using can yield misleading results [7], [8]. A model which is well-attuned to the features of a majority class but has poor performance when labeling a minority class, for example, might have high overall accuracy as the test set contains mostly pixels of the majority class. However, such a model cannot be considered successful.

Several alternative metrics sensitive to data imbalance have been proposed and used to evaluate models of unbalanced data. For the purposes of this project, we adopt the measuring scheme used for the BRATS challenge [3], which uses the Dice score to quantify the overlap between the ground-truth area of a particular tissue type and the area labeled as that type by our classifier. For some tissue class c, let P be a binary map of every pixel in the image to 1 if it is a member of class c, 0 otherwise. Furthermore, let T be the ground-truth binary mapping. Let P_1 and T_1 be the sets of all pixels mapped to 1 by P and T respectively. The Dice score is calculated thus:

$$Dice(P,T) = \frac{|P_1 \cup T_1|}{(|P_1| + |T_1|)/2}$$

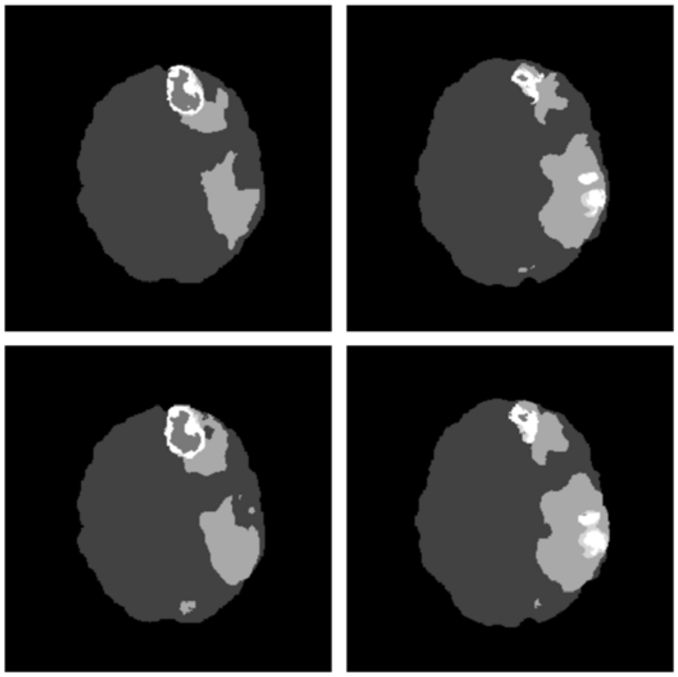


Fig. 5. The top two images are the ground truth labelings for two brain scans, the second row contains the segmentations done by the balanced-sampling model.

That is, the score for a particular class c is the size of the overlap between the predicted region and its true counterpart, normalized by the averaged size of the two regions.

The BRATS benchmark adapts the binary Dice score to the multi-class segmentation problem by choosing a subset of the set of minority classes and treating all tissue types in that subset as a single class. The three subsets under consideration are the entire tumor (containing all four cancerous tissue types), the tumor excluding edema, and the enhancing core region, which consists of a single tissue class [3].

VI. Experiments in Balanced Sampling

Initially, we investigated the impact of forcing each batch of training data to contain the same number of examples of each class represented in the dataset. Fig. 2 contains segmentations performed by this model alongside the ground truth labeling and segmentations produced by our baseline random-sampling model.

Note that the random model's segmentations tend to underestimate the amount of cancerous tissue contained in the scan, especially tissue of the rarest substructure types (those with the lightest pigmentation in the segmentations). That is, this model displays a typical failing of a classifier trained on unbalanced data: a tendency to overclassify the majority class at the expense of one or more minority classes. The balanced model does not display this tendency; instead, segmentations produced through balanced sampling tend to overestimate the area covered by the rare cancerous tissue types.

The overclassification vs. underclassification problem described above is evident in quantitative as well as qualitative

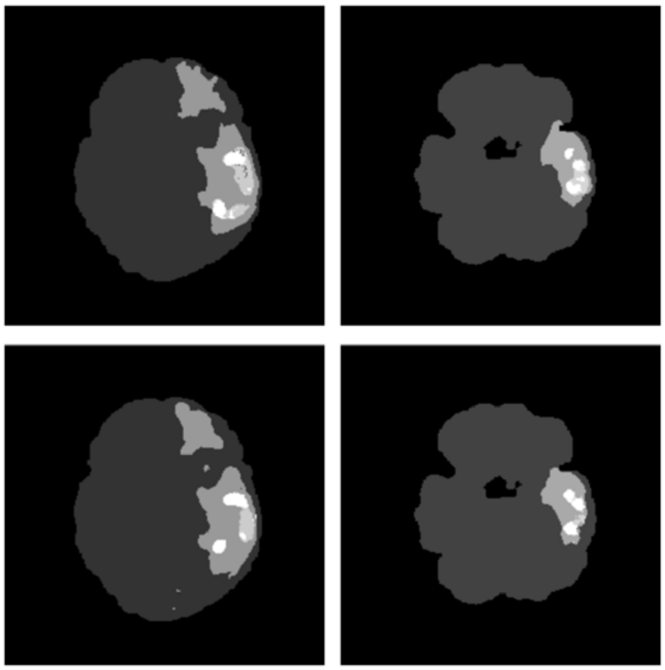


Fig. 6. The top two images are the ground truth labelings for two brain scans, the second row contains the segmentations done by the random sampling model.

evaluation of the generated segmentations. Below are the Dice Scores of the models trained using random and balanced sampling techniques.

	Whole Tumor	Core	Active Region
Random	0.863	0.776	0.786
Balanced	0.833	0.845	0.862

Note that although the average dice score of the random model over the entire area of the tumor is greater than the same score for the balanced model, the core and active-region scores for the balanced model are significantly higher. That is, those two scores most heavily dependent on the rarest tumor substructures are higher for the balanced model. However, this increased facility in identifying rare classes is coupled with a loss of facility in identifying more common classes. In an attempt to preserve the advantages of the balanced-sampling model and further improve its score on the full tumor region, we investigated the strategy of hard example mining.

VII. Experiments in Hard Example Mining

In an attempt to improve upon the balanced sampling approach, we incorporated the concept of hard example mining. This technique was introduced by Felzenszwalb et al., who combat class imbalance in object detection with a technique they termed "hard negative mining" [1]. This technique involves constructing each training batch such that it contains a disproportionately large number of minority-class samples alongside a small subset of the majority-class samples that are deemed "hard" for the current classifier. A sample is considered "hard" with respect to a classifier if it was misclassified or only correctly classified by a small margin by that classifier in the previous epoch. The intent is to reduce the proportion of majority class examples without sacrificing the classifier's ability to identify the majority class by emphasizing those examples it fails to classify correctly.

However, our focus was on identifying and reintroducing hard samples of all classes rather than just hard majority samples. Adopting a balanced sampling approach (and thus necessarily reducing the frequency of the majority class) did not substantially reduce our classifier's ability to identify the majority class. The Dice score for healthy tissue in the model trained on balanced batches of patches was greater than 0.99 and only marginally less than that of the model trained on randomly selected batches. That is, capping the fraction of majority class samples per batch at just 20% of the batch size didn't negatively impact the model's ability to recognize that class. Therefore, Felzenszwalb et al.'s "hard negative" approach was not appropriate for our problem, because performance on the majority class did not need improvement. Instead, we brought their technique to bear on each of the minority classes—our rare tumor substructures—in hopes that it would further increase our classifier's ability to identify those classes.

To evaluate the efficacy of this hard example strategy, we trained a third CNN. Between each training epoch, we ran the current classifier on a randomly selected subset of each class and stored the indices of all samples in this subset which were misclassified, or correctly classified by a small margin (with less than 75% confidence). When generating batches for the following epoch, we drew first from these "hard" examples to fill out each batch. We continued to balance the proportions of each class in each batch, gathering a number of samples from each class equal to 20% of the overall batch size.

Ultimately, this approach proved less effective than simply using balanced sampling methods alone. The table below displays the comparative Dice scores for the three sampling strategies.

	Whole Tumor	Core	Active Region
Random	0.863	0.776	0.786
Balanced	0.833	0.845	0.862
Hard Example	0.810	0.801	0.835

Note that hard example mining fails to perform as well as balanced sampling at segmentation all of the three tumor categories. The segmentations themselves (images below) display the problem visually. Evidently, the model trained primarily on hard data has the tendency to overdraw the tumor region. Although the locality of the tumor itself is correct, this classifier exaggerates its size and fails to recognize its boundaries. Perhaps reintroducing hard examples to the learning process overemphasizes the those characteristics which result in a class being confused for another class,

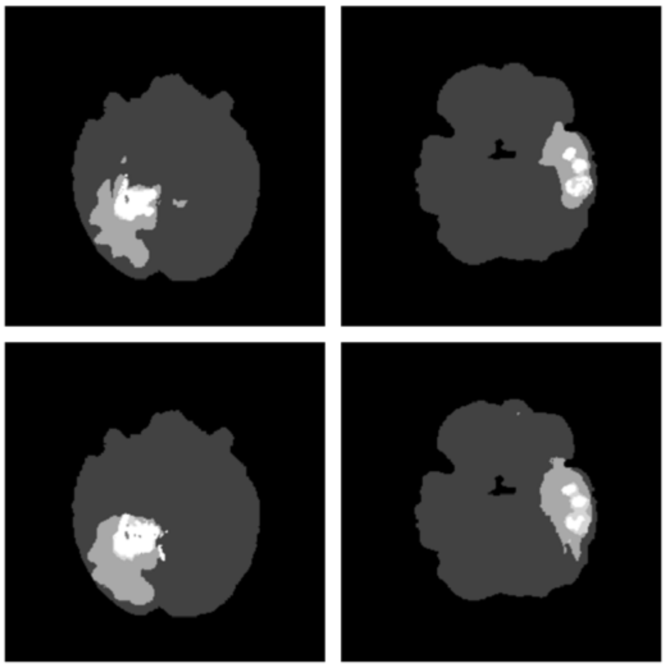


Fig. 7. The two bottom images are segmentations produced by the classifier trained on hard examples. Their corresponding ground truths are above.

ultimately blurring the boundary between the five classes in the dataset.

Although our experimentation with hard example mining failed to improve our classifier, the technique may merit further research, as discussed in the final section of this paper.

VIII. Areas of Further Exploration

Given the semi-promising results yielded by our hard example mining, I believe further exploration of this technique is certainly worthwhile. Further experiments could investigate the potential of hard example mining with differing ratios of majority and minority classes in training batches; for example, we could attempt to mimic the distribution of the minority classes in relation to each other in each training batch, while continuing to artificially skew the number of minority class samples in relation to the majority class, and combine this approach with hard example mining on those minority classes. It is also worth investigating whether this sampling technique is more effective in training on other datasets displaying the imbalance problem. If so, what characteristics of a dataset predict how useful the technique will prove in training a classifier for that data?

There is also further experimentation to be done in finding the ideal confidence threshold for identifying a "hard" example. It is possible that this sampling strategy proves more effective if only those samples which are actually misclassified are labeled hard, or, alternatively, when every sample not assigned to the correct class with 99% confidence is considered "hard."

A second promising further area of research is in mitigating the multi-class imbalance problem by training two separate classifiers, one for the majority class and another for the minority classes. The former would be trained to segment images into instances of the majority class and instances of any other class. The latter would be trained only on examples of the rarer classes and could then be used to further segment the non-majority pixels identified by the former into those rarer classes. In the context of the BraTS dataset, the first classifier would make a binary determination of whether each pixel represented healthy or diseased tissue, and the second classifier would identify the tumor substructures contained in the diseased patches identified by the first. Ideally, the second classifier would not be subject to the problem of class imbalance, as the diseased subset of the data has a much more balanced class distribution than the dataset as a whole.

Although much scholarship on the class imbalance problem exists already, the multitude of applications for powerful machine learning-based classifiers that can be trained on unbalanced data render further research and exploration in the area absolutely indispensable.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. 1659788 and Grant No. 1359275.

References

[1] Felzenszwalb, Pedro, David McAllester, and Deva Ramanan. "A discriminatively trained, multiscale, deformable part model." In Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, pp. 1-8. IEEE, 2008.

[2] Zhou, Zhi-Hua, and Xu-Ying Liu. "Training cost-sensitive neural networks with methods addressing the class imbalance problem." IEEE Transactions on Knowledge and Data Engineering 18, no. 1 (2006): 63-77.

[3] Menze, Bjoern H., Andras Jakab, Stefan Bauer, Jayashree Kalpathy-Cramer, Keyvan Farahani, Justin Kirby, Yuliya Burren et al. "The multimodal brain tumor image segmentation benchmark (BRATS)." IEEE transactions on medical imaging 34, no. 10 (2015): 1993-2024.

[4] Havaei, Mohammad, Axel Davy, David Warde-Farley, Antoine Biard, Aaron Courville, Yoshua Bengio, Chris Pal, Pierre-Marc Jodoin, and Hugo Larochelle. "Brain tumor segmentation with deep neural networks." Medical image analysis 35 (2017): 18-31.

[5] Bansal, Aayush, Xinlei Chen, Bryan Russell, Abhinav Gupta, and Deva Ramanan. "Pixelnet: Towards a general pixel-level architecture." arXiv preprint arXiv:1609.06694 (2016).

[6] Mazurowski, Maciej A., Piotr A. Habas, Jacek M. Zurada, Joseph Y. Lo, Jay A. Baker, and Georgia D. Tourassi. "Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance." Neural networks 21, no. 2 (2008): 427-436.

[7] He, Haibo, and Edwardo A. Garcia. "Learning from imbalanced data." IEEE Transactions on knowledge and data engineering 21, no. 9 (2009): 1263-1284.

[8] López, Victoria, Alberto Fernández, Salvador Garca, Vasile Palade, and Francisco Herrera. "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics." Information Sciences 250 (2013): 113-141.

[9] Chawla, Nitesh V., Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. "SMOTE: synthetic minority over-sampling technique." Journal of artificial intelligence research 16 (2002): 321-357.

Localizing Fluorescent Proteins Using Super-Resolution Neural Networks

Kyle Yee
Swarthmore College
kyee1@swarthmore.edu

Guy Hagen
University of Colorado
Colorado Springs
ghagen@uccs.edu

Jonathan Ventura
University of Colorado
Colorado Springs
jventura@uccs.edu

***Abstract*—Recent advances in microscopy and data analysis have allowed for the resolution of photoactivatable fluorescent protein (PA-FP) samples past the theoretical diffraction limit of 200 nm. To do this, several techniques have been developed which perform well on low density protein samples, but which have more difficulty resolving high density PA-FP images. This project seeks to super-resolve PA-FP samples using Convolutional Neural Networks (CNNs) by combining super-resolution and localization techniques. This implementation achieves good results on existing contest datasets and acts as a generalizable model to other protein samples. Notably, the neural network significantly outperforms other easily-accessible algorithms. Results from these experiments suggest that convolutional neural networks are a very promising method for single-molecule localization in a wide array of situations, with evaluation times much faster than existing methods.**

***Keywords*—Computational and artificial intelligence, Neural networks, Image processing, Machine vision, Object recognition**

I. Introduction

In conventional optics, the resolution achievable by an optical instrument is limited by the effects of light diffraction at small scales. In microscopy, this limit occurs around 200 nm, where objects or features smaller than this scale are not resolvable by the instrument alone. However, Betzig et al. [1] introduced a new technique for super-resolving images of photoactivatable fluorescent proteins (PA-FPs). These proteins are on the order of 10 nm and can be made to fluoresce at random intervals through exposure to laser light. The density of proteins flashing at one time can be varied by changing the frequency of laser pulses. By recording a signal from active PA-FPs, techniques such as PALM [1] and STORM [2] fit these signals with a single point-spread function (PSF). Through this process, PALM and STORM achieve good results in locating the position of a given PA-FP up to 20 nm, a factor of ten past the diffraction limit.

Despite this success of PALM and STORM, these methods have difficulty resolving PA-FP high-density fluorescence signals where diffraction patters occur simultaneously in close proximity. Instead, these techniques are limited to "sparse fields" where the majority of the sample is inactive at a given time [1]. However, when limited to sparse fields, sampling times must be long (on the order of 10 seconds) in order to capture an occurrence of each individual protein fluorescence, and such a limitation poses challenges to imaging living samples. Currently, there is an active area of research on developing techniques which can handle these high-density situations in the hope of decreasing the imaging time required for individual PA-FP samples. With this in mind, this research project seeks to use CNNs to resolve microscopic images through a machine learning approach, which has not yet been thoroughly explored.

II. Related Work

There are many algorithms which seek to accomplish the task of high-density localization microscopy. While none of the current leading algorithms use machine learning techniques, there are quite a few which improve on the original PALM and STORM methods. Some notable techniques are listed below.

Holden et al. [4] introduce DAOSTORM, an improvement on single PSF methods which is capable of fitting multiple PSFs to locations of high-density PA-FP signals. By doing this, DAOSTORM processes high-density signals with better precision than STORM and PALM methods. DAOSTORM achieves some of the best results to date on localization tasks involving high-density data.

Zhu et al. [5] build Compressed Sensing STORM (CSSTORM), which achieves higher density results than DAOSTORM and improves sampling time to 3 seconds. CSSTORM models PSFs as linear transformations on protein position data, and divides the output space into grid locations as small as one-eighth pixel size.

In the same year, Mukamel et al. [6] create deconvolutional STORM (deconSTORM). This technique also treats PSFs as reversible transformations on an original image, referred to as convolutions (distinct from convolutional neural networks). However, Mukamel et al. introduce non-linearity to this transformation to achieve good high-density results. By performing deconvolutions, deconSTORM is able to reconstruct super-resolution images.

In 2014, Ovensky et al. introduce ThunderSTORM [7], an ImageJ plugin which acts as a culmination of several different methods used to super-resolve proteins. ThunderSTORM has high performance in a variety of localization tasks, and is extremely accessible. Along with performing analysis, ThunderSTORM includes a realistic simulator of PA-FP data as well as an evaluator to analyze the performance of other methods.

Most recently, Min et al. [8] design the FAst Localization algorithm based on a CONtinuous-space formulation (FALCON). As opposed to previous methods, FALCON achieves continuous output space by fitting PA-FP signals using Taylor

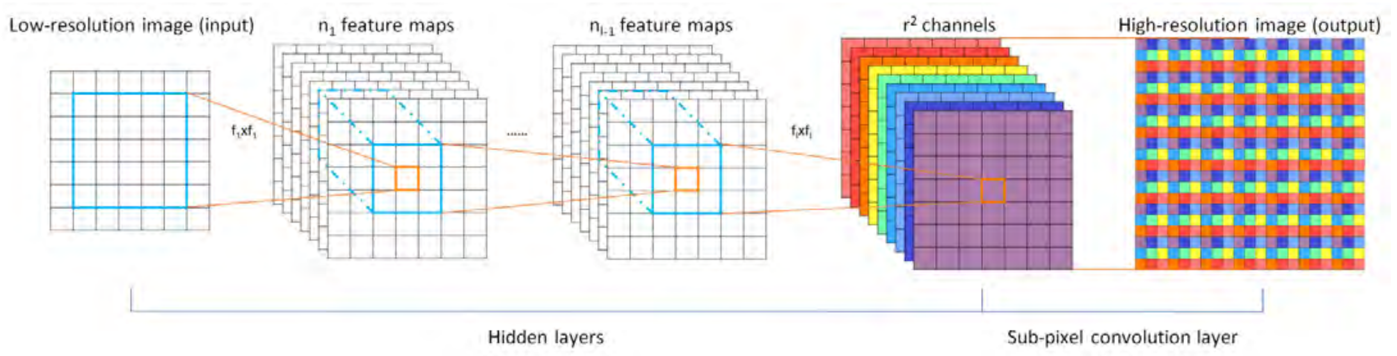


Fig. 1: Efficient implementation of the subpixel convolution layer (image taken without permission from Shi et al. [3]). This diagram shows two initial convolution layers followed by the subpixel convolution layer, which outputs a single-channel image by rearranging the channels in the subpixel layer to increase resolution.

approximations of PSFs. Thus, as compared to other methods, FALCON is able to achieve higher-precision localization without significantly increasing computation complexity. Compared to CSSTORM, FALCON also reduces sample time to 2.5 second temporal resolution. Along with DAOSTORM, FALCON performs very well in high-density localization tasks.

III. Methods

A. Super-Resolution

Recently, CNNs have achieved state-of-the-art results for super-resolution tasks [3], [9], [10]. In particular, Shi et al. [3] introduce an efficient method for learning super-resolution by upscaling in the network directly, rather than relying on other external upscaling methods. This particular architecture is based on the idea of subpixel convolutions, where a normal convolutional layer is used with fractional stride size in order to upscale the resulting output. However, as noted by Shi et al., subpixel convolution exponentially increases training time. Thus, they introduce a novel, efficient method for computing an output equivalent to that of subpixel convolutions. This is achieved by convolving a normal filter with $r{*}r{*}c$ channels and a stride of 1×1, where r is the upscaling factor. The resulting image has size $w \times h \times r * r * c$, and is then rearranged into an image of size $w * r \times h * r \times c$. This operation is represented by Figure 1.

Our network architecture implements this technique in order to map low-resolution microscope data to a high-resolution label space. By using this convolutional layer, we can train an end-to-end super-resolution network for localization with arbitrarily-large output resolution, thus increasing our localization accuracy.

B. Distance Transform Regression

The task of localization is closely related to the well-studied problem of counting. The current best methods for counting are based on the work of Lempitsky and Zisserman. [11], where regression models are trained to map objects to density maps. These density maps represent objects as Gaussian distributions, and are designed in such a way such that the discrete integral of the map is equal to the count of objects in the given image. While this technique has achieved state-of-the-art results for the task of counting on various datasets [12], it does not provide a good method for accurately localizing objects in these images.

However, density-map regression can be slightly altered in order to better-localize proteins. Rather than learning regression model from objects to Gaussians, we attempt to map a pixel to its distance from the nearest protein location, as proposed by Kainz et al. [13]. Each pixel in a label sample is assigned a value d based on its Euclidean distance to the nearest localization. Then, the following operation is performed on the label data:

$$f(d) = \begin{cases} e^{\alpha(1-\frac{d}{d_{max}})} - 1 & 0 \leq d < d_{max} \\ 0 & d \geq d_{max} \end{cases} \tag{1}$$

Through this operation, called the *distance transform*, each protein is assigned an exact peak with the value e^{α}. After training our model to regress protein images to this function, we can find local maxima of the network output, thresholded at some lower-bound. These maxima correspond to protein locations and are accurate up to the resolution of the output image. Furthermore, these locations can be refined slightly by fitting a quadratic to a small neighborhood about any local maxima, thus producing a continuous output space.

IV. Experiments

A. Datasets

Test data for this experiment comes from the Single-Molecule Localization Microscopy (SMLM) Symposium challenges from 2013 and 2016. These challenge datasets provide simulated PA-FP microscope samples at varying densities as well as ground-truth locations for these simulated proteins. These simulations are designed to model proteins in various tubulin structures.

The SMLM challenges provide leaderboards showing results from a number of existing techniques (including those mentioned above) on contest datasets. While the ground truth localizations for these datasets are not provided, our results may be sent in for evaluation, allowing us to make general comparisons between our method and various state-of-the-art algorithms.

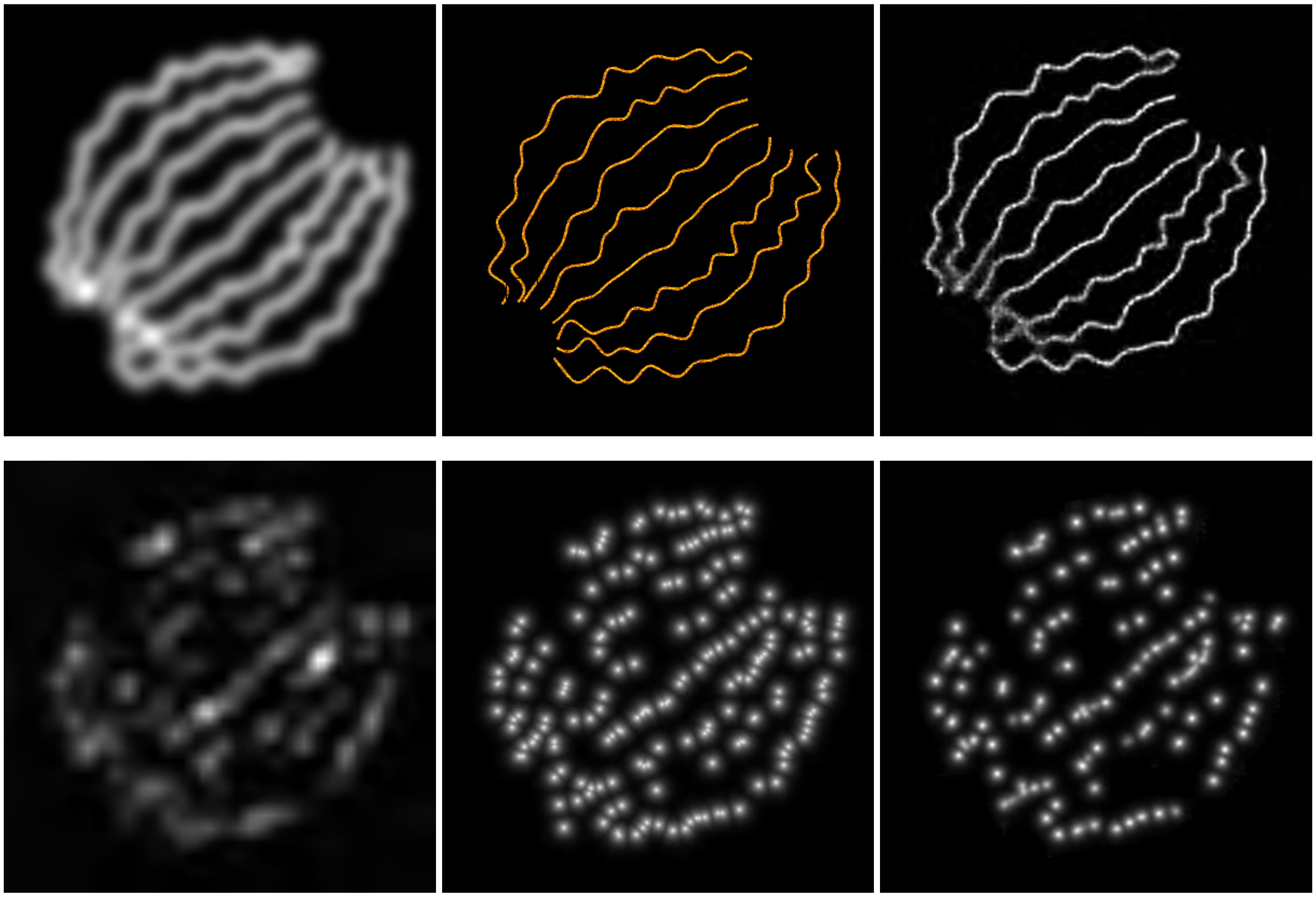


Fig. 2: Results from the 2013 SMLM dataset using the distance transform. The top row shows the 2013 dataset averaged over all frames (left), the ground-truth protein positions (middle), and a histogram of our network protein location predictions (right), where a brighter value signifies more-frequent occurrences of protein locations. The bottom row contains a sample testing image (left), its label data using the distance transform (middle), and our network prediction (right)

1) 2013: The 2013 challenge is focused on high-density localization. From this challenge, we use the “Bundled Tubes High Density” dataset, which models 8 tubulins with a diameter of 30 nm, in a 100 nm thick microscope slide. This dataset has 81049 fluorophores contained in 168 frames, making it the highest density dataset used in our experiments.

2) 2016: The primary focus of the 2016 challenge is on 3D localization. However, the challenge contains multiple 2D training datasets. In order to distinguish these datasets from those in the 2013 challenge, these 2D localization datasets simulate a thicker microscope slide of 1500 nm, meaning that approximately half of the proteins appear out of focus. From the 2016 challenge, we use the MT0.N1.HD and MT0.N2.HD datasets, both of which represent high-density samples. Proteins in these datasets occur at a slightly lower density than the 2013 samples, modeling three microtubules over 2500 frames and a total of 11172 flashes. These datasets simulate identical protein flashes and locations within each frame, differing only in signal-to-noise (SNR) ratios. The MT0.N1.HD dataset has a high peak SNR average of 22.597, while the MT0.N2.HD has a lower peak SNR average of 19.425. We will refer to these datasets as high-SNR and low-SNR 2016 datasets respectively.

3) Evaluation: Training data is generated using ThunderSTORM’s simulator, which creates artificial microscopic images together with ground-truth locations. This simulator incorporates a number of parameters which specify the background noise, density of proteins, and camera detector settings. In order to evaluate the neural network performance, we use ThunderSTORM’s built-in evaluation program. This program computes statistics such as the precision, recall, Jaccard Index, F1-Measure, and root-mean-squared (RMS) error of a result compared to ground-truth data. In this evaluation, one specifies the maximum distance allowed for a localization to be considered correct, which we refer to as the tolerance. Our neural network predictions are generated using the methods described in the following subsection.

B. 2013 Experiments

For the 2013 dataset, we use an architecture of 9 convolutional layers with 3×3 filter-size and 32 feature-detectors, 1 subpixel layer with an upscaling factor $r = 7$ and 32 feature

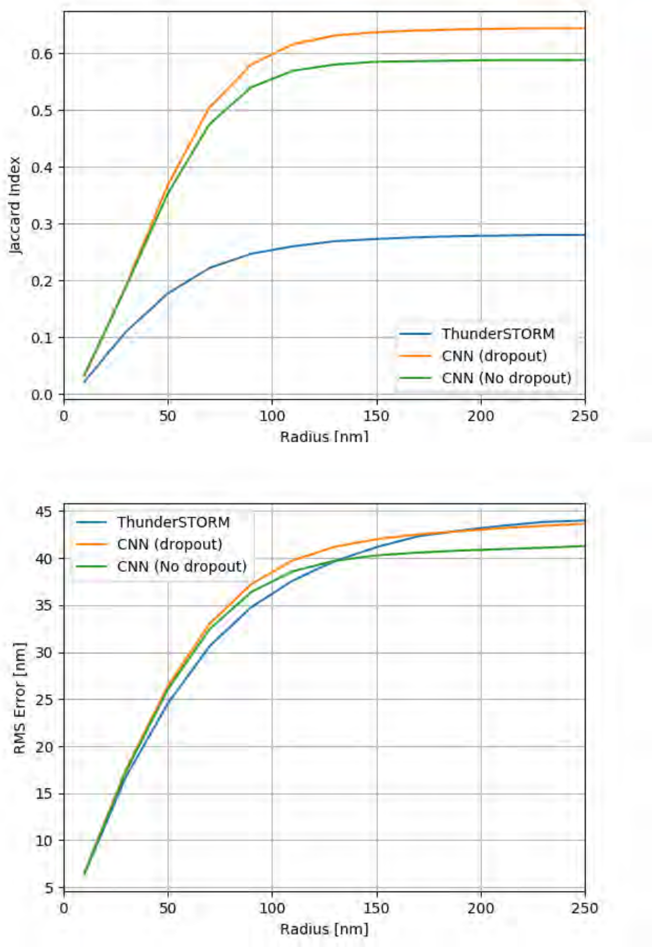


Fig. 3: Jaccard Index (top) and RMSE (bottom) for our 2013 neural network predictions at a range of tolerances. These plots show our neural network results with and without dropout layers, as well as the results from ThunderSTORM's analysis

detectors, and a 1×1 filter size flattening convolutional layer which outputs gray scale image. We train this network on 1000 64×64 images with a pixel-size of 100×100 nm generated with ThunderSTORM. These images are designed to have density and background noise similar to that found in the 2013 SMLM HD dataset, but with completely random distribution rather than tubulin structure. For the distance transformation, we set $\alpha = 7$ and $d = 35$ pixels. We experiment with this configuration using a network with no dropout as well as one with a dropout rate of $.5\%$. Visual results of the resulting regression map are shown in Figure 2.

When evaluating these results, we choose a threshold of 300 for local-maxima. After evaluating the Jaccard Index and RMS error of our results at various tolerances between 10 and 250 nm, we also analyze the 2013 dataset using ThunderSTORM's built-in protein localization software. A comparison between these methods is shown in Figure 3, which includes both dropout and non-dropout network architectures.

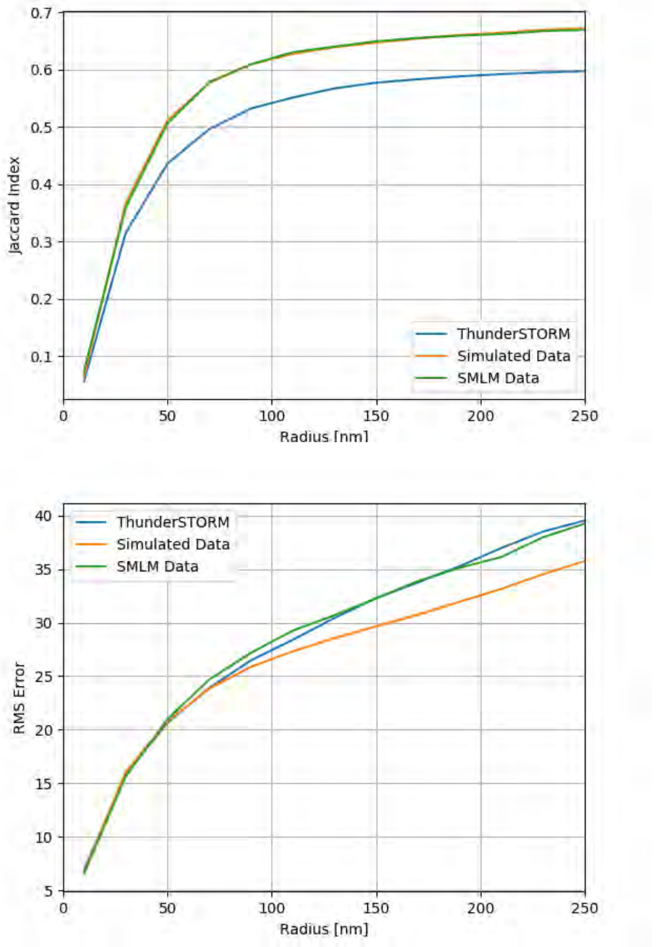


Fig. 4: Jaccard Index (top) and RMSE (bottom) on the 2016 high-SNR SMLM dataset. Here, we show results when training the CNN on data simulated from ThunderSTORM and on SMLM contest data. We also show ThunderSTORM results.

C. 2016 Experiments

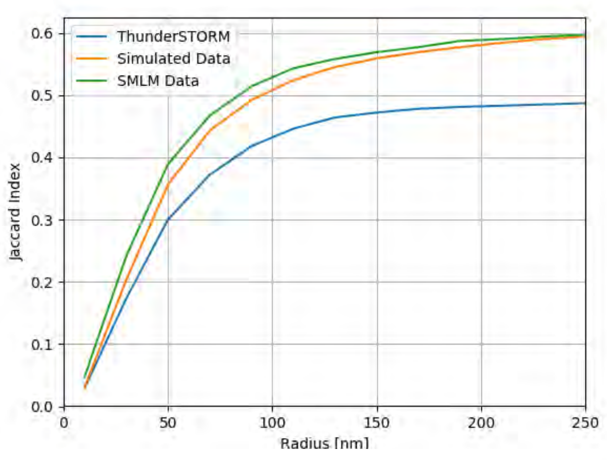
Due to the thicker nature of the 2016 SMLM datasets, not all protein flashes are in focus in the SMLM training datasets. Because of this, our training method for the 2016 data differ slightly from those in the 2013 experiments. For both low- and high-SNR datasets, we train on 4000 simulated ThunderSTORM images. These simulated images are made using gradient density masks, which increase protein density from top to bottom of the image, as well as gradient noise masks, which increase the noise signal in the simulations from the left to the right. During preprocessing, we flip and rotate these images in cycles of 8 to achieve all possible orientations of these density and noise masks. In both datasets, we use an a model with 11 convolutional layers (32 feature detectors), 1 subpixel layer with a factor $r = 7$ and 10 feature detectors, and a final 1×1 convolutional layer. In both of these models, we use a value of $\alpha = 7$ and $d = 42$.

We compare these results to the ThunderSTORM evaluation of these datasets. We also train our model on low-SNR and high-SNR training sets directly in order to compare against the performance using simulated training data. Before training on these datasets directly, we set aside the first 500 frames of both datasets for validation. The results from all of these experiments are shown in Figures 4 and 5, all tested on the 500 validation frames.

D. Results

In many of these experiments, we see that our super-resolving CNN outperforms ThunderSTORM in both Jaccard Index and RMS error. The exception to this is in the 2016 low-SNR RMS error, where ThunderSTORM obtains a slightly lower value. Despite this, our implementation improves significantly on ThunderSTORM's results in terms of Jaccard Index (see Tables I and II)

From these experiments, we see that our network learns a generalizable model for protein localization. After training on stochastically-placed protein signals and testing on data with a tubulin structure, our network achieves good results in localizing high-density proteins. Furthermore, in the 2016 experiments, there is no significant difference between CNN performance when trained on ThunderSTORM-simulated data as opposed to contest data directly. Thus, our experiments



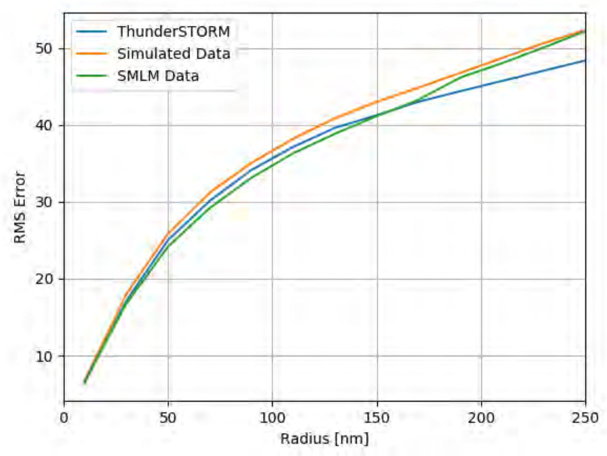


Fig. 5: Jaccard Index (top) and RMSE (bottom) on the 2016 low-SNR SMLM dataset. As in Figure 4, we show ThunderSTORM analysis as well as results when training the network on ThunderSTORM simulated data and on SMLM data.

	ThunderSTORM	CNN (dropout)	CNN (no dropout)
Jaccard Index	.280	**.644**	.588
RMSE [nm]	44.0	43.7	**41.3**

TABLE I: 2013 results

	ThunderSTORM	Simulated Data	SMLM Data
Jaccard Index	.597/.487	.672/.594	**.669/.597**
RMSE [nm]	39.5/**48.4**	**35.7**/52.4	39.2/52.2

TABLE II: 2016 results (high-SNR/low-SNR)

suggest that this network can be applied to a wide array of microscope data, by simply changing the parameters of the ThunderSTORM simulations used for training.

Finally, note that evaluation times are significantly reduced in the neural network method when compared to methods such as ThunderSTORM. After training, our network evaluates the 500-frame 2016 validation sets in approximately 90 seconds, where the majority of this time is spent loading the images and writing out the results. This time itself could be improved significantly by using a solid-state drive

V. Conclusion

Our network achieves very promising results on high-density datasets. With further refinement, we expect our results to be competitive with other top methods aimed at localizing high-density proteins. Although CNNs have not been thoroughly explored in this context, initial results from this project indicate their applicability to the field of nanometer-scale microscopy. By implementing techniques such as subpixel super-resolution and distance transform regression, we have shown that neural networks are a fast and accurate method for imaging living samples beyond the diffraction limit of 200 nanometers.

VI. Future Work

In the upcoming months, we propose further modifications to our model. Currently, our architecture only uses one scaling layer in order to upscale the image. We propose to experiment with multiple scaling layers with smaller scale factors, thus introducing non-linearities between phases of scaling. One example architecture consists of three scaling layers with a scale-factor $r = 2$, thus resulting in a total upscaling of 8. Furthermore, we have up until now chosen to focus on 2D localization tasks. However, the 2016 SMLM Challenge primarily focuses on 3D localization, and thus has several 3D datasets available. In the coming months, we propose to extend our model to 3D localization tasks. Finally, we plan to submit our results to the SMLM 2016 Challenge in order to more-directly compare our method against current state-of-the-art algorithms.

VII. Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 1359275 and 1659788. Furthermore, we acknowledge Diptotip Deb and Sridhama Prakhya for their helpful conversations and insights during the research process.

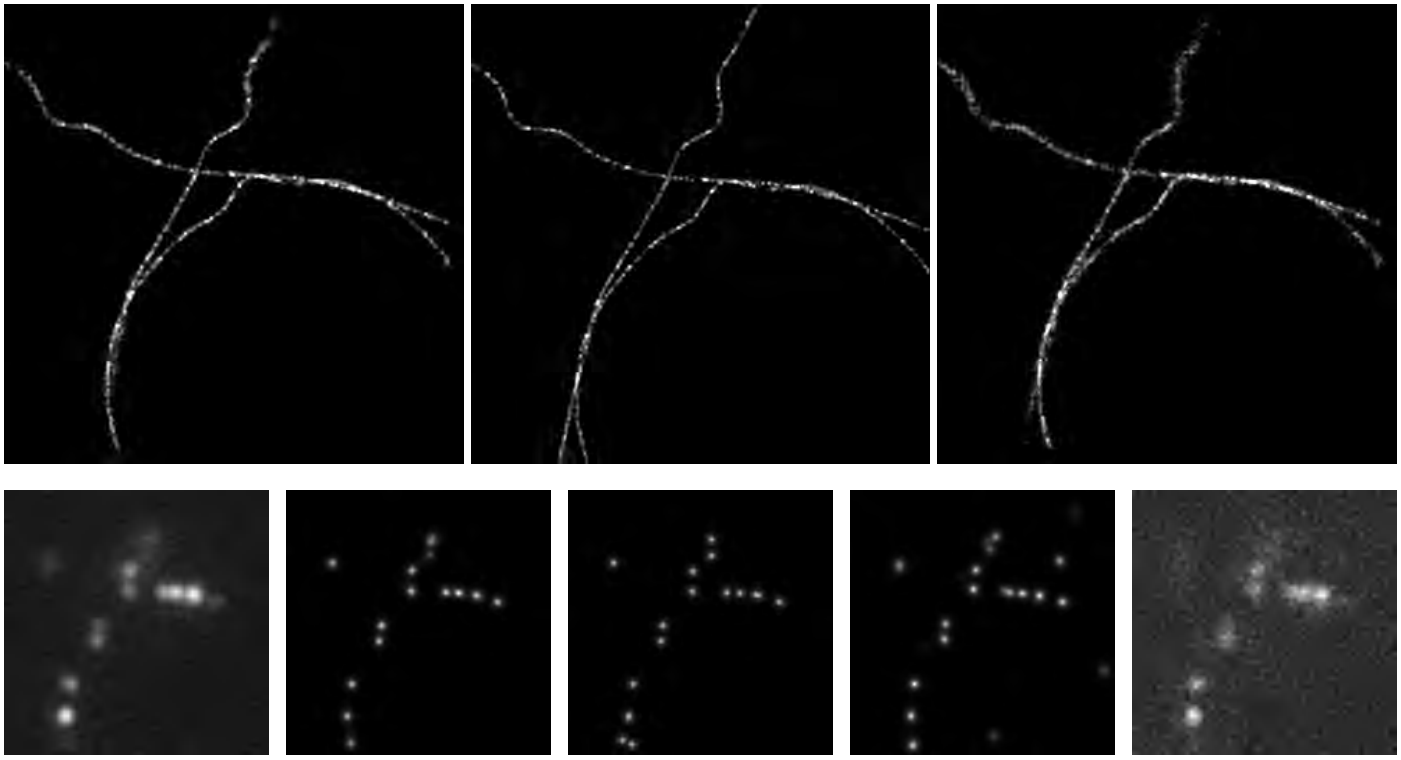


Fig. 6: Results from the 2016 low- and high-SNR datasets. The top row shows average histograms from the network predictions, where the center shows the ground truth, the left shows high-SNR results and the right shows low-SNR results. The bottom row displays a high-SNR testing image (left), the high-SNR network prediction (left-middle), the ground-truth label (middle), the low-SNR network prediction (right-middle), and the low-SNR testing image (right)

References

[1] E. Betzig, G. H. Patterson, R. Sougrat, O. W. Lindwasser, S. Olenych, J. S. Bonifacino, M. W. Davidson, J. Lippincott-Schwartz, and H. F. Hess, "Imaging Intracellular Fluorescent Proteins at Nanometer Resolution," *Science*, vol. 313, no. 5793, 2006. [Online]. Available: http://science.sciencemag.org/content/313/5793/1642

[2] M. J. Rust, M. Bates, and X. Zhuang, "Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (storm)," *Nat Meth*, vol. 3, no. 10, pp. 793–796, 10 2006. [Online]. Available: http://dx.doi.org/10.1038/nmeth929

[3] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1874–1883, 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7780576/

[4] S. J. Holden, S. Uphoff, and A. N. Kapanidis, "DAOSTORM: an algorithm for high- density super-resolution microscopy," *Nature Methods*, vol. 8, no. 4, pp. 279–280, 2011. [Online]. Available: http://www.nature.com/doifinder/10.1038/nmeth0411-279

[5] L. Zhu, W. Zhang, D. Elnatan, and B. Huang, "Faster STORM using compressed sensing," *Nature Methods*, vol. 9, no. 7, pp. 721–723, 2012. [Online]. Available: http://www.nature.com/doifinder/10.1038/nmeth.1978

[6] E. A. Mukamel, H. Babcock, and X. Zhuang, "Statistical deconvolution for superresolution fluorescence microscopy," *Biophysical Journal*, vol. 102, no. 10, pp. 2391–2400, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.bpj.2012.03.070

[7] M. Ovesný, P. Křížek, J. Borkovec, Z. Švindrych, and G. M. Hagen, "ThunderSTORM: A comprehensive ImageJ plug-in for PALM and STORM data analysis and super-resolution imaging," *Bioinformatics*, vol. 30, no. 16, pp. 2389–2390, 2014.

[8] J. Min, C. Vonesch, H. Kirshner, L. Carlini, N. Olivier, S. Holden, S. Manley, J. C. Ye, and M. Unser, "FALCON: fast and unbiased reconstruction of high-density super-resolution microscopy data," *Scientific Reports*, vol. 4, no. 1, p. 4577, 2015. [Online]. Available: http://www.nature.com/articles/srep04577

[9] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *European Conference on Computer Vision*, vol. 8689, 2014, pp. 184–199. [Online]. Available: http://link.springer.com/10.1007/978-3-319-10593-2_13

[10] J. Kim, J. K. Lee, and K. M. Lee, "Accurate Image Super-Resolution Using Very Deep Convolutional Networks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1646–1654.

[11] V. Lempitsky and A. Zisserman, "Learning To Count Objects in Images," *Advances in Neural Information Processing Systems*, pp. 1324–1332, 2010. [Online]. Available: http://papers.nips.cc/paper/4043-learning-to-count-objects-in-images.pdf

[12] D. Onoro-Rubio and R. J. Lopez-Sastre, "Towards Perspective-Free Object Counting with Deep Learning," in *ECCV 2016: 14th European Conference, Amsterdam, The Netherlands*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Springer International Publishing, 2016, pp. 615–629. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46478-7_38

[13] P. Kainz, M. Urschler, S. Schulter, P. Wohlhart, and V. Lepetit, "You Should Use Regression to Detect Cells," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9351, pp. 276–283, 2015.

Segmenting Images with a Deep Auto-encoder and K-Means Clustering

Adia Meyers
Clayton State University
Email: ameyers3@student.clayton.edu

***Abstract*—The purpose of this research is to improve some of the current methods of image segmentation, thereby allowing for more accurate results when categorizing images. By specifically venturing into image segmentation and neural networks, we hope to find a collaborative and beneficial correlation between convolutional neural networks and categorizing images. This research combines various segmentation methods and evaluation methods in hopes of creating a robust algorithm. By performing a particular technique of image segmentation, the desired product is to be capable of classifying local, global, and multi class images. In turn, this will constitute for an improved and more accurate way of segmenting images.**

Keywords*—*Image segmentation, convolutional neural network

I. Introduction

Image segmentation is the process of isolating an image into a number of sections which are known as segments [5,6,7,8] so that the image can be classified. Image segmentation plays a large role in: classifying terrains in satellite images, medical image analysis, character recognition, and more. There are a variety of image segmentation techniques, such as thresholding, clustering, edge detection, implementing a watershed transform, using artificial intelligence for segmentation, and many more methods. Thresholding segments images by looking at the intensity values of pixels [8]. Clustering groups images with similar characteristics [4,14], and edge detection detects discontinues between objects in an image [12], therefore finding the boundaries of each object in an image. Watershed algorithms transform a grayscale image by acting as a topographic map, with the brightness of each point representing its height [2]. Artificial intelligence based classification has recently begun to play a larger role in image segmentation in the form of neural networks, in particular convolutional neural networks [5,6,10].

A neural network is an artificial intelligence system modeled after our very own human brains and is made of layers of nodes. Every nodes incoming connection has a weight associated with it. This weight is multiplied by the input. Neural networks consist of at least three layers. A input layer, hidden layer and an output layer. Neural networks learn by first evaluating training images. Then test images, usually of the same object type as the training images are evaluated. Convolutional neural networks are very similar to ordinary neural networks, meaning that they are made up of neurons that have learn-able weights and biases [1,10,16]. The main differences between the two being that convolutional neural networks do not make use of every feature in an input image and they are capable of conductiong dimensionality reduction. Convolutional neural network architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture [12]. The Convolutional neural network processes the input and output data. Any data that is given after the dataset is classified appropriately. What makes neural networks so unique, is that they do not require pre-made categories in order to classify images.

II. Related Work

Segmenting a multi-class image as well as a large scale image can be some of the most difficult aspects of image classification. In 1987, auto-encoders were first proposed as a means of aiding in image segmentation [1]. An auto-encoder is a unsupervised convolutional neural network that applies back-propagation, meaning that the output is the same as the input [10]. An auto-encoder is made of two parts, an encoder and a decoder. The encoder portion breaks down the input into a vector, and the decoder builds the vector back up into the original input image. In a paper entitled Image Restoration Using Convolutional Auto-encoders with Symmetric Skip Connections, Mao et la. use a deep auto-encoder for image restoration. Auto-encoders pose as useful attributes for image segmentation due to their ability to perform dimensionality reduction and extract important features of an image [3,9,15].

Although auto-encoders aide in image segmentation, they do not actually classify an image. For this research project, a clustering algorithm will be implemented in order to segment the images. K-means clustering is an unsupervised clustering algorithm. The k means algorithm takes in the input as well as a required parameter (k) which will determine the number of clusters. The desired result is for the points in a similar cluster to have a minimized distance and for the distance between clusters to be maximized. Duan et al. implement a simple k means algorithm for the purpose of classifying fish images. It divides data into a predetermined classes on the basis of minimizing the error function.

III. Problem Definition

When implementing certain image segmentation methods such as using a watershed transform or thresholding algorithm, there always the constant needed requirement of pre-made categories. Using a convolutional neural network eliminates this prerequisite. Despite the fact that there are implementations of convolutional neural networks and image segmentation, there is still much room for improvement. For example, Meyer et la. establish a convolutional radial basis function solver paired with k means clustering to perform image segmentation [11]. Their experiment yielded accurate results, however, they still had an accuracy loss of about twenty percent on average. By replacing the radial basis function solver with a different convolutional neural network, we may be able to generate even more accurate results. This will also grant us the opportunity to create another method of image segmentation that could potentially outperform other segmentation methods as well.

IV. PROPOSED RESOLUTION

Our proposed resolution is to implement a model with a high accuracy for image segmentation. We will combine the methods of a conolutional neural network and a k-means clustering algorithm. The type of convolutional neural network to be implemented will be an auto-encoder. Refer to figure 1 for a representation of our network.

A. The Auto-encoder: The auto-encoder we developed mainly served as a means of extracting important features and reducing dimensions of images. Our auto-encoder is made of convolutions and deconvolutions. The encoder portion of the auto-encoder takes in a patch from the image as its input and break the input down into a vector known as the embedding of the auto-encoder. Then the rELU activation function is applied to the embedding and the decoder builds the vector back up to the original image. Lastly the L2 loss function will be calculated for the sole purpose of determining how well the output image matches the input image. This will measure the robustness of the auto-encoder.

B. T-SNE The vector from the embedding of the auto-encoder is extracted and the t-Distributed Stochastic Neighbor Embedding is applied to that vector in order to transform the vector points into two dimensional points. Ergo making it easier to plot the points for our k means resulting cluster graph.

C. The K-means clustering algorithm: In the second process, the two-dimensional vector points from the T-SNE replace the color values in the cluster vector. This will allow us to use distance instead of a color based vector as our cluster input vector. The K means algorithm has one parameter, which is the desired number of clusters (k).

D. Lastly, we plan to use the recall @ K method to determine the accuracy of our network. Recall @ K relies on the number of tests made [11]. R represents the total tests done and N represents the amount of test which were correct. N is divided by R and that determines the accuracy. The larger N is, constitutes for a better accuracy representation.

V. EXPERIMENT RESULTS

The first step in our experiment was to train the auto-encoder. There were 2 data sets which were run on the auto-encoder, mnist and cifar. There are 60,000 images in each of these data sets. 50,000 were used for training and 10,000 were used for testing. The cifar data set consists of random colored objects such as animals and cars. The mnist data set consisted of black and white images of handwritten numbers. The cifar images were 32x32 pixels and the mnist data set were images of 28x28 pixels. Each data set was trained for 6,000 epochs and a batch size of 1 due to their small sizes. The output results for the mnist data set is shown in figure 2 and the results for the cifar data set is shown in figure 3.

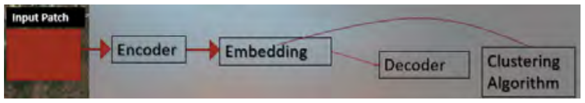


Fig. 1. A representation of our future image segmentation model.

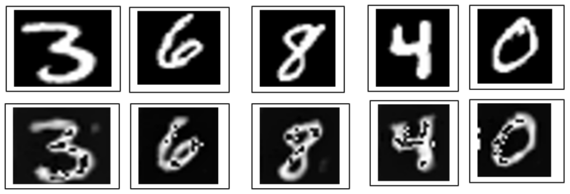


Fig. 2. Images from mnist datanset after being processed through our auto-encoder.

During the development of the k means algorithm, all of the figures 4 - 9 were developed from the mnist data set points. Figure 4 shows our graph results after applying the t-SNE to our data points without performing any k means on our data. As of right now, we are attempting to fix our t-SNE code, due to the fact that there should be much more data points in figure 4. Ergo indicating that our code needs improvement. Before resulting to the recall @K technique, we tried another evaluating technique that looked at the predicted labels vs actual labels on a graph. However we unable to get that functioning and decided to move on to a different method. Figure 6 was the first real plot that showed any sort of clustering, we initialized our k clusters to three for this trial. Despite the small milestone figure 6 signified, it still did not display all the points as we wanted. Figure 7 was an unsuccessful plot, which did not plot the clusters individually. Figures 8 and 9 are from our most recent results and we initialized our number of cluster classes to ten for both of these experiments. Figure 8 is graph of our most recent k means code without applying t-SNE to our data points before clustering them. Figure 9 is a graph of our data points after having the t-SNE applied to them and running our k means code. Despite being unable to get the recall @K to work, we were able to see that the points in figure 8 were closer together and therefore more clustered.

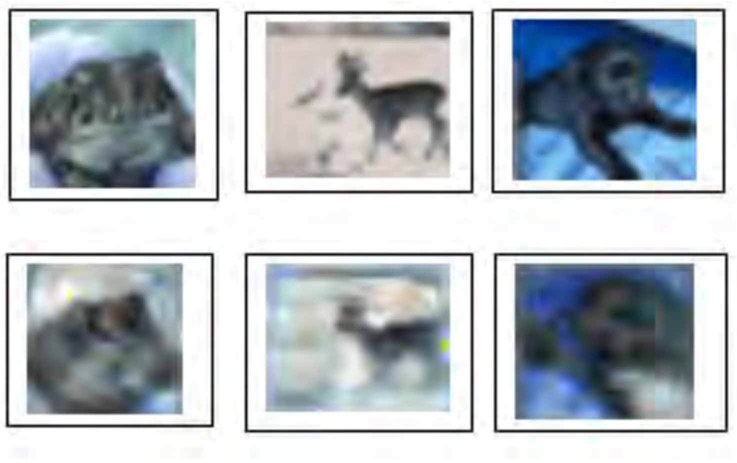


Fig. 3. Images from mnist dataset after being processed through our auto-encoder.

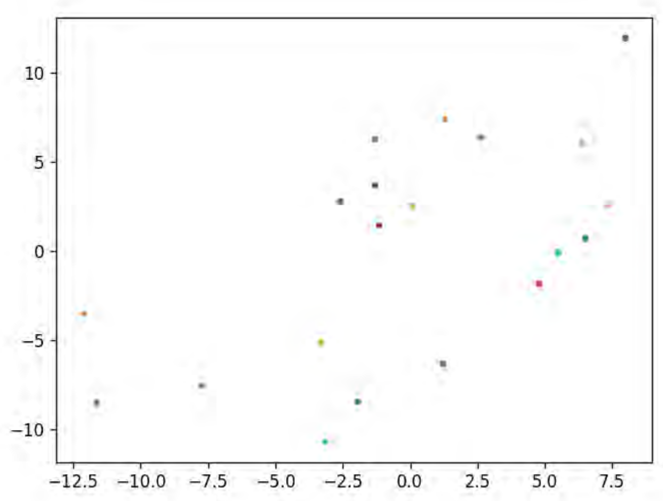


Fig. 4. Plot of points after applying t-SNE.

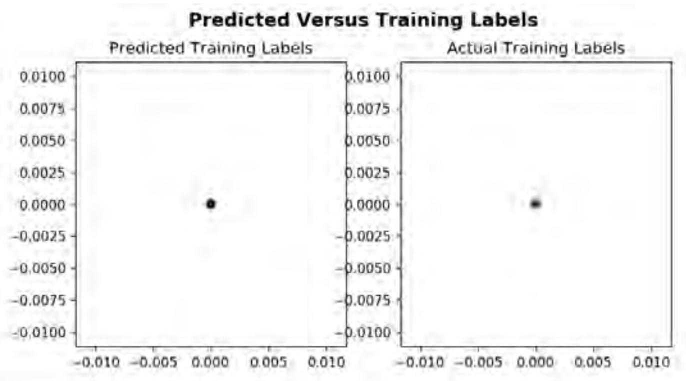


Fig. 5. A plot that was supposed to show our predicted vs actual training labels.

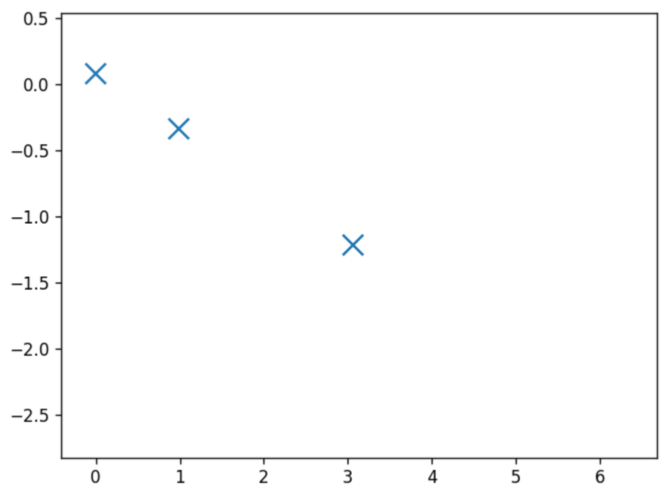


Fig. 6. Plot of the one of our trial k means algorithms using mnist data set.

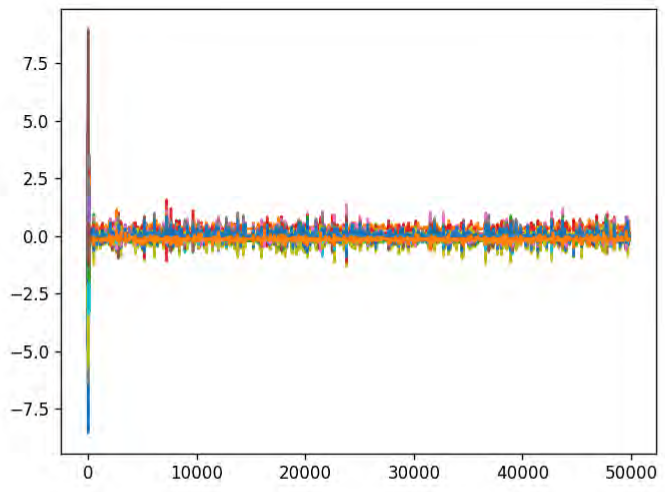


Fig. 7. Plot of the one of our trial k means algorithms using mnist data set.

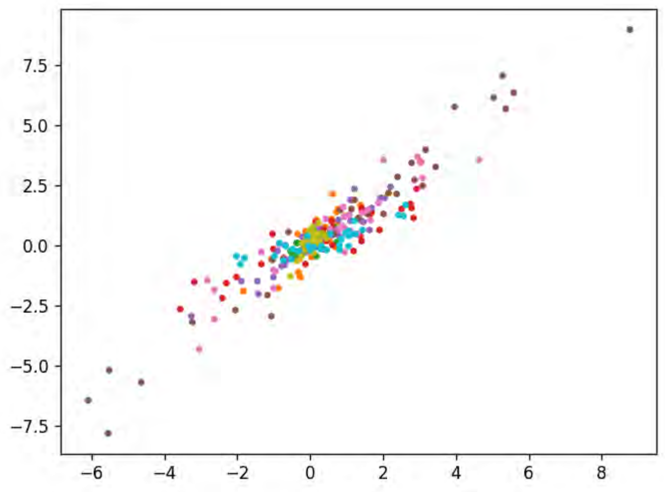


Fig. 8. K means cluster of our data before applying the t-SNE component to our data points

VI. Future Work

There are several objectives which need to be completed for this research project. The matter of highest concern is to make a more robust k means algorithm. As of right now our clusters are all quite close together with quite a few outliers, which means that our results need to be improved. In order to get more uniform clusters, our accuracy measurement code needs to be properly implemented. We hope to have our recall @ K evaluator working soon, however, as of right now, our code does not properly measure accuracy. In addition, the radial basis function solver needs to created so that we are actually able to compare its performance against our very own auto-encoder based model. There are three main data sets which we will be working with. Currently, we have used the mnist data

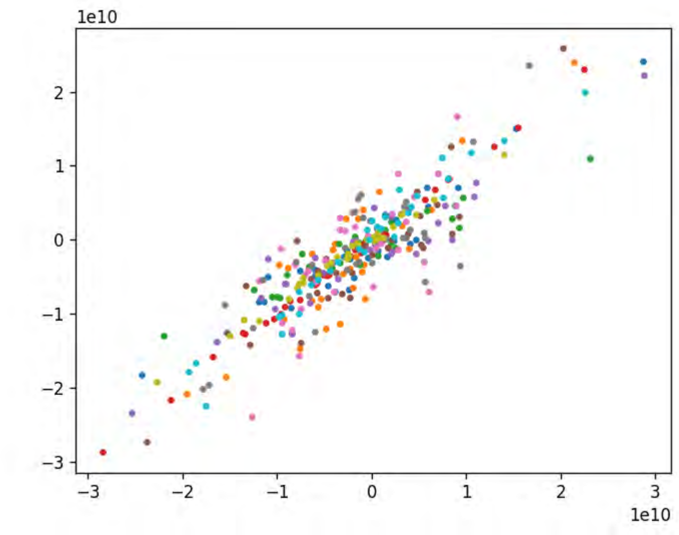


Fig. 9. K means cluster of our data before applying the t-SNE component to our data points

set of handwritten digits on our auto-encoder model. Once our model is fully up and running with the mnist data set, then we will run the cifar data set and the mass buildings data set on our model and the radial basis function solver. Lastly, we will explore the possibility of increasing our auto-encoder channels. This in turn may allow for better quality of our training images, making it easier to cluster and classify testing images.

VII. Conclusion

The objective of this task is to establish a mechanism that will provide for optimal image segmentation. By breaking down images into layers using an auto-encoder and then applying a k-means clustering algorithm to the dimensionally reduced images, we aspire to strengthen the link between neural networks and image segmentation. If this experiment is successful, the outcome will produce well defined clusters and highly accurate classification of images. In turn, this would become another way to improve current knowledge and procedures for future and present problems dealing with image segmentation.

Acknowledgment

This research was primarily funded and supported by the Research Experience for Undergraduates program at the University of Colorado Colorado Springs Machine Learning in Natural Language Processing and Computer Vision department under grant No. 1659788 from the NSF and grant No. 1359275 from the NSF.

References

[1] Bengio, Yoshua. Learning Deep Architectures for AI. Now Publishers Inc, 2009.

[2] Benson, C. C., V. L. Lajish, and Kumar Rajamani. ”Brain tumor extraction from MRI brain images using marker based watershed algorithm.” In Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on, pp. 318-323. IEEE, 2015.

[3] Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. ”Image style transfer using convolutional neural networks.” In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[4] Gong, Maoguo, Linzhi Su, Meng Jia, and Weisheng Chen. ”Fuzzy clustering with a modified MRF energy function for change detection in synthetic aperture radar images.” IEEE Transactions on Fuzzy Systems 22, no. 1 (2014):

[5] Kapoor, Dimple, and R. Kashyap. ”Segmentation of Brain Tumor from MRI Using Skull Stripping and Neural Network.” (2016).

[6] Kim, Jiwon, Jung Kwon Lee, and Kyoung Mu Lee. ”Accurate image super-resolution using very deep convolutional networks.” In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[7] Koltun, Philipp Krhenbhl Vladlen. ”Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials.” (2011).

[8] Kumar, S., et al.: Skull stripping and automatic segmentation of brain MRI using seed growth and threshold techniques pp. 422426 (2007)

[9] Le, Quoc V. ”A Tutorial on Deep Learning Part 2: Autoencoders, Convolutional Neural Networks and Recurrent Neural Networks.” Google Brain (2015).

[10] Mao, Xiao-Jiao, Chunhua Shen, and Yu-Bin Yang. ”Image restoration using convolutional auto-encoders with symmetric skip connections.” arXiv preprint arXiv:1606.08921 (2016).

[11] Meyer, Benjamin J., Ben Harwood, and Tom Drummond. ”Nearest Neighbour Radial Basis Function Solvers for Deep Neural Networks.” arXiv preprint arXiv:1705.09780 (2017).

[12] Perona, P., Malik, J.: Scale-space and edge detection using anisotropic diffusion. Pattern Analysis and Machine Intelligence, IEEE Transactions on 12(7), 629639 (1990)

[13] Radhakrishna, Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. ”Slic superpixels.” Technical Report 149300, EPFL (2010).

[14] Ray, Siddheswar, and Rose H. Turi. ”Determination of number of clusters in k-means clustering and application in colour image segmentation.” In Proceedings of the 4th international conference on advances in pattern recognition and digital techniques, pp. 137-143. 1999.

[15] Schmidhuber, Jrgen. ”Deep learning in neural networks: An overview.” Neural networks 61 (2015)

[16] Stutz, David. ”Understanding convolutional neural networks.” In Seminar Report, Fakultt fr Mathematik, Informatik und Naturwissenschaften Lehr-und Forschungsgebiet Informatik VIII Computer Vision. 2014.

Learning perspective-free counting via dilated convolutions

Diptodip Deb
College of Computing
Georgia Institute of Technology
Atlanta, Georgia, USA
Email: diptodipdeb@gatech.edu

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

***Abstract*—We propose the use of dilated convolutions as a simpler approach to the perspective-free counting problem. Counting is a common problem in computer vision (e.g. cells in a microscope image or pedestrians in a crowd). Modern approaches to the counting problem involve the production of a density map via regression whose integral is equal to the number of objects in the image. This method of counting can also be used to locate objects in the image if the regressor used has enough accuracy and precision. However, objects in the image can occur at different scales (e.g. due to perspective effects) which can make it difficult for a neural network to learn the proper density map. A recent result for multiscale counting involves the use of a complicated pyramid of image patches. However, dilated convolutions have been shown to allow for the incorporation of multiscale information without such a complicated design in segmentation problems. We see that our dilated convolutional regression network obtains results comparable to and occasionally superior to the current state of the art.**

***Keywords*—Computational and artificial intelligence, neural networks, machine vision.**

I. Introduction

Many computer vision problems involve dense prediction [1]. Generally, these problems can involve discrete or continuous labeling of images. Counting objects in an image is a specific sub-problem of this sort. Specifically, the problem involves enumerating the number of objects in a given still image or video frame [2]. We have seen that good performance on counting tasks can be achieved without learning to detect and localize dense objects in images through the regression of a density map [2].

This means we can focus on the specific case in which each object has been labeled with a dot (one dot per object). In this case, the supervised learning agent simply learns a regression function to produce the density map such that the integral of the density map is equal to the number of objects in the image [2].

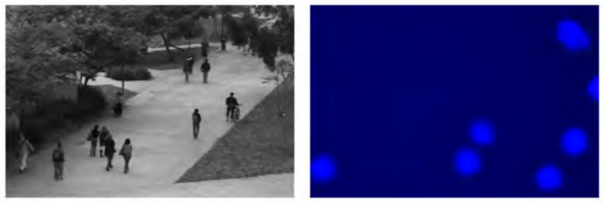


Fig. 1. UCSD pedestrian traffic data (left) and simulated microscopy image of biological cells (right). These are example images that can be used in counting tasks.

Indeed, we see that in many natural counting problems (such as those involving cells in a microscopy image, pedestrians in a crowd, or a traffic jam), individual detectors are not reliable [3]. This is due to a variety of challenges including overlap of objects, perspective shifts causing variance in shapes and sizes of objects, etc. [3].

A recent result in counting achieves state-of-the-art performance on counting objects that might be transformed by such perspective shifts [3]. This approach involves the regression of a density map as in [2], however their approach involves a complex convolutional neural network architecture that samples re-sized patches of different scales in order to incorporate multiscale information [3].

Indeed, convolutional neural networks have proven to provide state-of-the-art performance on a variety of dense prediction computer vision tasks [4], [5]. However, it is not clear whether the approach of sampling patches of varying scale is necessary for incorporating multiscale information [1].

Dilated convolutions, a simple modification to the structure of traditional, straightforward convolutional neural network designs, have proven to provide competitive performance in the dense, multiscale segmentation problem in which objects of different sizes in an image must be segmented [1]. We propose the use of such a network for the problem of regression of a density map for the purposes of counting rather than classification and segmentation. Such a network would have a much simpler architecture than that of [3], bypassing the need to sample multiscale patches of an image.

II. Related Work

Counting using a supervised regressor to formulate a density map was first shown by [2]. In this paper, Lempitsky et al. show that the minimal annotation of a single dot blurred by a Gaussian kernel produces a sufficient density map to train a network to count. All of the counting methods that we examine as well as the method we use in our paper follow this method of producing a density map via regression. This is particularly advantageous because a sufficiently accurate regressor can also locate the objects in the image via this method. However,

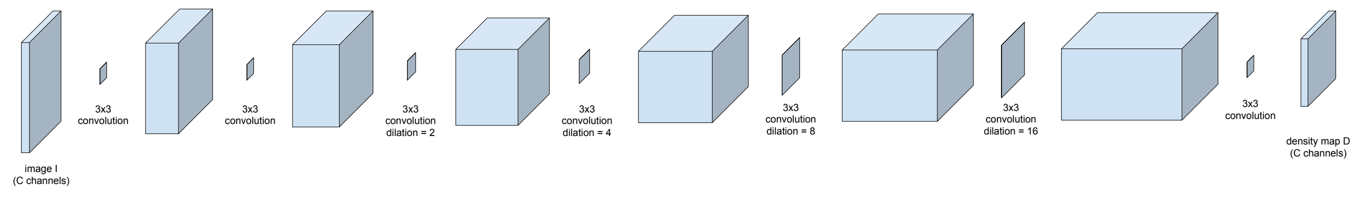


Fig. 2. Visualization of the dilated convolution regression network architecture. All layers are activated using the ReLU function.

the Lempitsky paper ignores the issue of perspective scaling and other scaling issues. The work of [6] introduces CNNs (convolutional neural networks) for the purposes of crowd counting, but performs regression on similarly scaled image patches.

These issues are addressed by the work of [3]. Rubio et al. show that a fully convolutional neural network can be used to produce a supervised regressor that produces density maps as in [2]. They further demonstrate a method dubbed HydraCNN which essentially combines multiple convolutional networks that take in differently scaled image patches in order to incorporate multiscale, global information from the image. The premise of this method is that a single regressor will fail to accurately represent the difference in values of the features of an image caused by perspective shifts (scaling effects) [3].

However, the architectures of both [3] and [6] are complicated due to requiring multiple image patches and, as discussed in [1], the experiments of [7], [8] and [9]–[11] leave it unclear as to whether rescaling patches of the image is truly necessary in order to solve dense prediction problems via convolutional neural networks. In [7], [8], upsampling is used to recover scale information from downsampled layers, which puts into question the necessity of downsampling scaled layers in the first place. Further, it is also unclear in [9]–[11] as to whether separate inputs of rescaled patches of the image are necessary. The work of [1] proposes the use of dilated convolutions as a simpler alternative that does not require sampling of rescaled image patches to provide global, scale-aware information to the network.

It should be noted that other methods of counting exist, including training a network to recognize deep object features via only providing the counts of the objects of interest in an image [12] and using CNNs (convolutional neural networks) along with boosting in order to improve the results of regression for production of density maps [13]. In the same spirit, [14] combines deep and shallow convolutions within the same network, providing accurate counting of dense objects (e.g. the UCF50 crowd dataset).

In this paper, however, we aim to apply the dilated convolution method of [1], which has shown to be able to incorporate multiscale information without using multiple inputs or a complicated network architecture, to the counting problem.

III. Method

A. Dilated Convolutions

We propose the use of dilated convolutions as an attractive alternative to the more complicated architecture of the HydraCNN [3]. We largely keep the architecture shown in [1], making a simple modification in order to produce a regression network rather than a segmentation (classification) network which is described below. Dilated convolutions, as discussed in [1], allow for the exponential increase of the receptive field with a linear increase in the number of parameters with respect to each hidden layer.

In a traditional 2D convolution, we define a real valued function $F : \mathbb{Z}^2 \to \mathbb{R}$, an input $\Omega_r = [-r, r]^2 \in \mathbb{Z}^2$, and a filter function $k : \Omega_r \to \mathbb{R}$. In this case, a convolution operation as defined in [1] is given by

$$(F * k)(\mathbf{p}) = \sum_{\mathbf{s}+\mathbf{t}=\mathbf{p}} F(\mathbf{s})k(\mathbf{t}).$$

A dilated convolution is essentially a generalization of the traditional 2D convolution that allows the operation to skip some inputs. This enables an increase in the size of the filter (i.e. the size of the receptive field) without losing resolution. Formally, we define from [1] the dilated convolution as

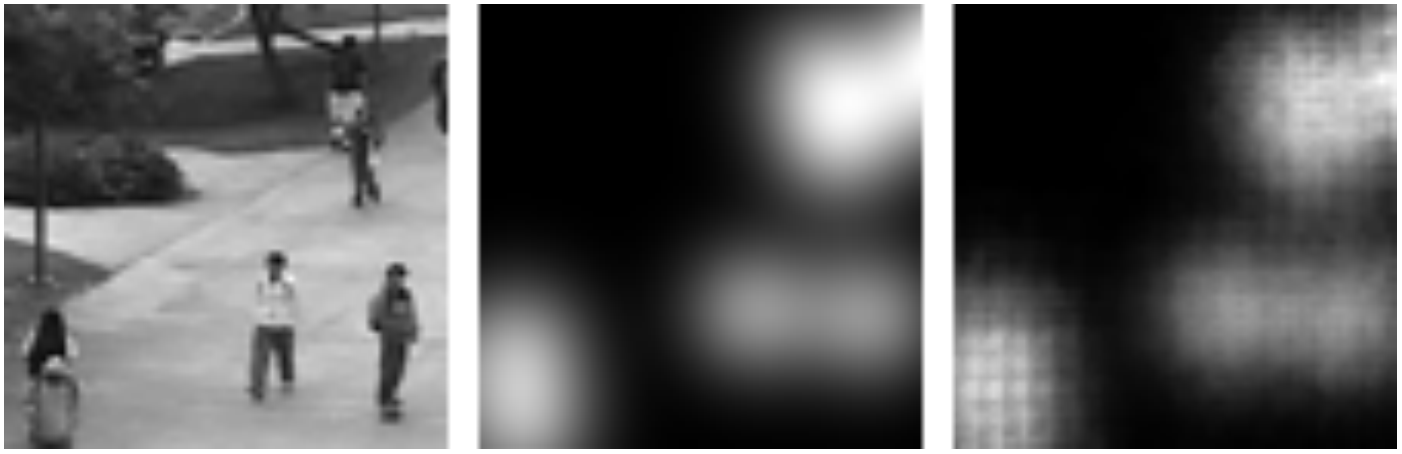
$$(F *_l k)(\mathbf{p}) = \sum_{\mathbf{s}+l\mathbf{t}=\mathbf{p}} F(\mathbf{s})k(\mathbf{t})$$

where l is the index of the current layer of the convolution.

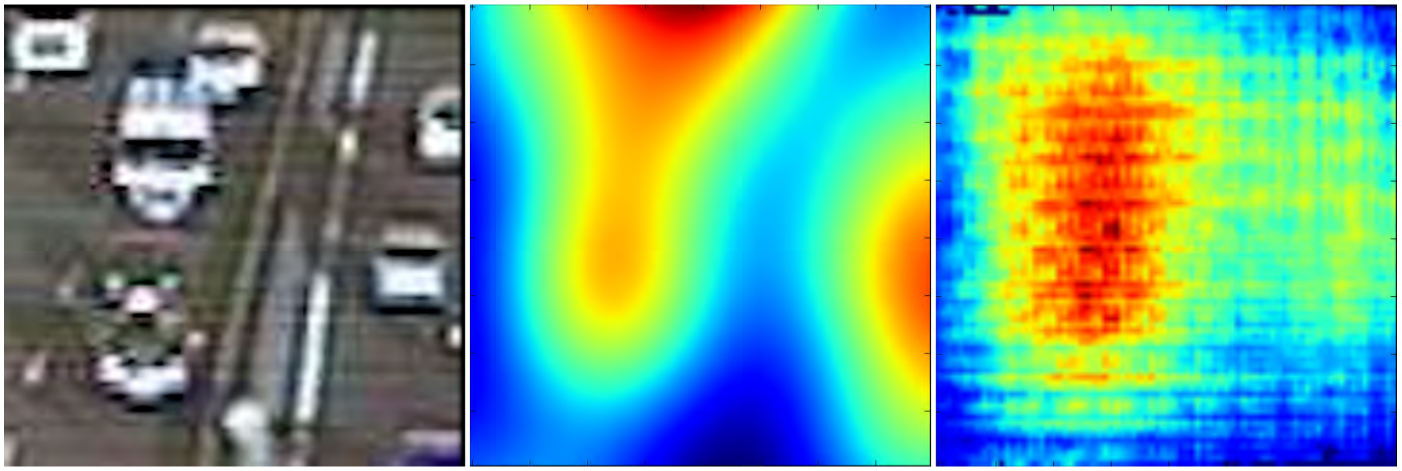
We mostly keep the architecture of the network in [1] in terms of the dilated convolution filters. Because we have a regression problem as opposed to a segmentation problem, we do not implement the front end to extract features, leaving this to the dilated convolutions themselves. Furthermore, we use a ReLU activation for all the layers in order to facilitate regression of floating point 32-bit pixel-values (which have a range of 0 to 1). We refer to this network as the dilated convolutional regression network, henceforth shortened as the DCR network.

B. Experiments

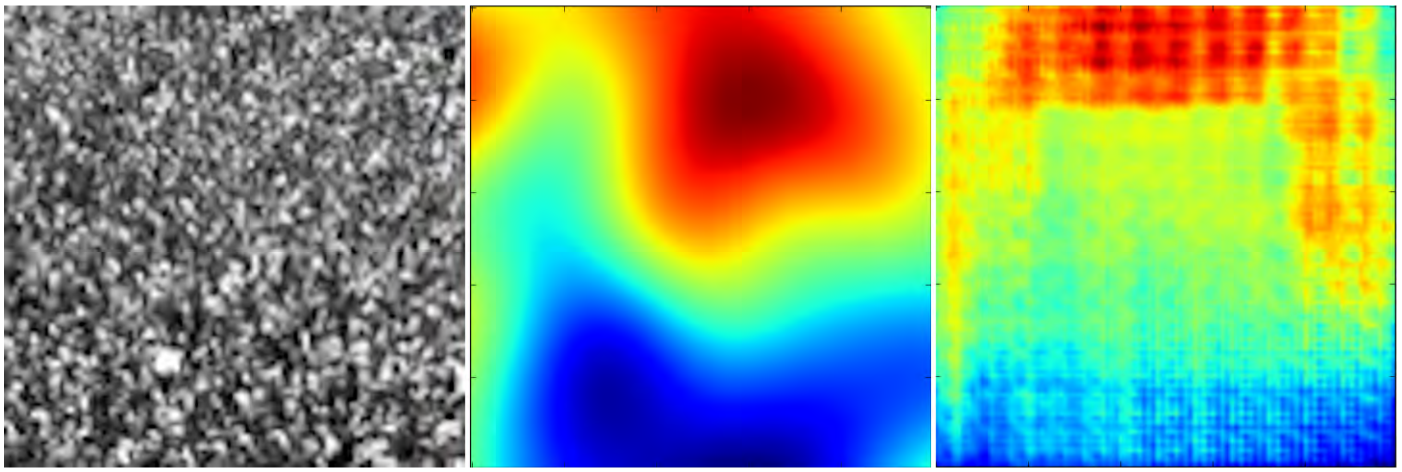
We evaluated the performance of dilated convolutions against the HydraCNN on a variety of common counting datasets: UCF50 crowd data, UCSD crowd data, and TRANCOS traffic data [3] and [15]. For each of these data sets, we used labels given by the corresponding density map for each image. An example of this is shown in Figure III-A. Currently, we have performed experiments on the four different splits of the UCSD data as defined in [3] and the split of the UCSD data as defined in [15] (which we call the Shanghai split). We also evaluated the performance of our network on the TRANCOS traffic dataset [16]. We have also experimented with testing on small patches from the UCF data (discussed in section IV-C).



(a) UCSD dilation network sample.



(b) TRANCOS dilation network sample.



(c) UCF dilation network sample.

Fig. 3. Left: input counting image. Middle: Generated ground truth density map. Right: Dilated convolutional network prediction of density map on test image. The network never saw these images during training. All images and density maps were one channel only (i.e. grayscale), but may be colored here for clarity. Note the gridding pattern in (a). This is mitigated somewhat in (b), though the actual values are overall worse in (b) than in (a).

We have so far observed that dilated convolutions produce density maps (and therefore counts) that are on par with or better than those of HydraCNN [3]. We measure density map regression loss via L1 loss. We compare accuracy of the counts via mean absolute error for the crowd datasets and the GAME metric in the TRANCOS dataset as explained in Section IV-A3. Beyond the comparison to HydraCNN, we will also compare to other recent convolutional counting methods, especially those of [15], [12], [13], and [14]. Further experiments in counting could involve testing the effect of increasing dilation size and deepening the network (which effectively increases the size of the receptive field) on the size of images the network is able to take in.

IV. Results

We perform experiments on various data sets. For all datasets, we use patched input images and ground truth density maps produced by summing a Gaussian of a fixed size (σ) for each object. This size varies from dataset to dataset, but remains constant within a dataset. We do not take any perspective information into account for training our network. All experiments were performed using Keras with the Adam optimizer at its default learning rate [17].

A. Datasets

1) UCSD: The UCSD crowd counting dataset consists of frames of video of a sidewalk. There are relatively few people in view at any given time (approximately 25 on average). Furthermore, because the dataset comes from a video, there are many nearly identical images in the dataset. For this dataset, there have been two different ways to split the data into train and test sets. Therefore, we report results using both methods of splitting the data. The first method consists of four different splits: maximal, downscale, upscale, and minimal. Minimal is particularly challenging as the train set contains only 10 images. Moreover, upscale appears to be the easiest for the majority of methods [3]. The second method of splitting this data is much simpler, leaving 1200 images in the testing set and 800 images in the training set [15].

2) UCF: UCF is a particularly challenging dataset. The difficulty is due not only to the very low number of images in the dataset, but also to the fact that the images are all of varying scenes. Furthermore, perspective effects are particularly noticeable for particular images in this dataset. The average image has on the order of 1000 people in a crowd in this dataset.

3) TRANCOS: TRANCOS is a traffic counting dataset that comes with its own metric [16]. This metric is known as $GAME$, which stands for Grid Average Mean absolute Error. $GAME$ splits a given density map into 4^L grids, or subarrays, and obtains a mean absolute error within each grid separately. The value of L is a parameter chosen by the user. These individual errors are summed to obtain the final error for a particular image. The intuition behind this metric is that it is desirable to penalize a density map whose overall count might match the ground truth, but whose shape does not match the ground truth [16]. More formally, we define

$$GAME(L) = \frac{1}{N} \cdot \sum_{n=1}^{N} \left(\sum_{l=1}^{4^L} |e_n^l - t_n^l| \right)$$

where N refers to the number of images, L is the level parameter for $GAME$, e_n^l is the predicted or estimated count in region l of image n and t_n^l is the ground truth count in region l of image n [16].

B. UCSD Crowd Counting

For this dataset, each object is annotated with a Gaussian of size $\sigma = 8$. The ground truth map is produced by summing these. There are two different ways to split the dataset. We have experimented on the split that gave [3] the best results as well as the split used in [15].

We note that training this method using the symmetric dilation method as introduced in [18] results in density maps that are better in terms of the shape produced, but worse in terms of the actual count values. This can be seen in Figure IV-B. At best, symmetric dilations are approximately equivalent to the standard dilated regression network and at worst symmetric dilations are unable to learn anything at all on the same training set that was used to train the standard dilated regression network. Hence, we have proceeded with the standard dilation regression network.

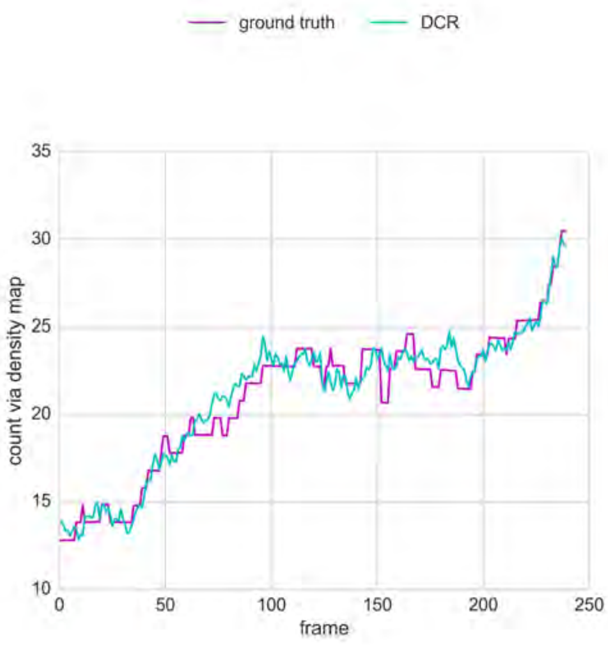
1) Upscale Split: We see that the “upscale” split as defined in [3] gives us very good results on counting for this dataset. For this experiment, we sampled 1600 random patches of size 119×79 pixels (width and height respectively) for the training set and split the test set images into 119×79 quadrants that could be easily reconstructed by simply piecing them together without overlap. Results appeared consistent over multiple trainings.

2) Shanghai Split: We see that the “Shanghai” split as defined in [15] gives us somewhat worse results for counting on this dataset. For this experiment, we again sampled 1600 random patches of size 119×79 pixels (width and height respectively) for the training set and split the test set images into 119×79 quadrants that could be easily reconstructed by simply piecing them together without overlap. Results appeared consistent over multiple trainings. While the performance of the network was not as good, i.e. the network does not achieve state of the art performance, we see that the results are comparable to the state of the art and the previous state of the art. This is compelling because the purpose of the dilated regression network is to show that perspective-free counting can be learned without creating image pyramids or combining multiple CNNs learning features at different scales.

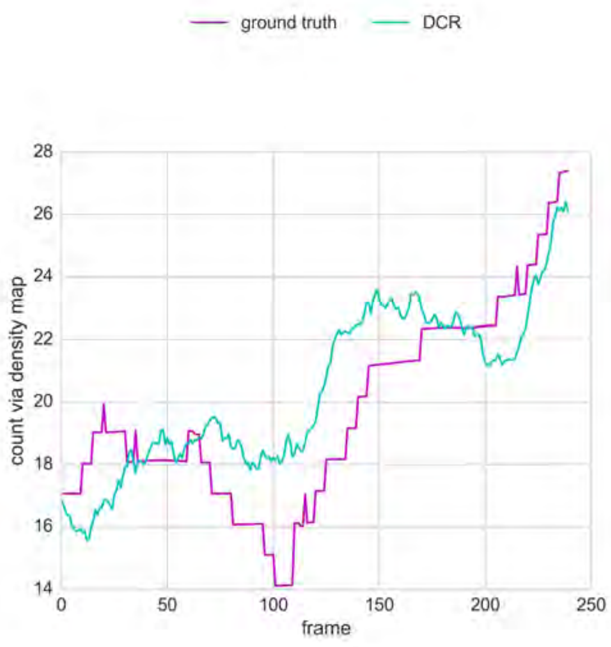
C. UCF Crowd Counting

For this dataset, we initially did not fully test the images. Instead, for this dataset we also test on random image patches of the same size as the training patches. We take 1600 random patches of size 100×100 for training. We do the same for testing. Ground truth density maps are produced by annotating each object with a Gaussian of $\sigma = 15$. We see that because the UCF dataset has over 1000 people on average in each image, the shapes output by the network in the density map are not as well defined or separated as in the UCSD dataset. This can be seen in Figure 3c. While the average error when testing on patches seems to be quite low as indicated by Table II, when we test on 5 cross validation folds of the data as defined in [3], we find that the error increases to an average of approximately 1000, which is far higher than the state of the art. The low average error on the patches is misleading because the average error of the patches is summed over the number of patches for each image. However, modifications to the DCR network could yield significantly better results for these dense images.

Fig. 4. Left: input counting image. Middle: Generated ground truth density map. Right: Dilated convolutional network prediction of density map on test image. Note the grid pattern in Figure 3a. The symmetric dilation network does somewhat mitigate this grid pattern as shown here, but it ultimately performs worse for obtaining an actual count.



(a) UCSD upscale split.



(b) UCSD Shanghai split.

Fig. 5. Both plots show a comparison of the predicted and ground truth counts as time (the current frame) progresses. We see that while the DCR network does not do as well on the Shanghai split as on the upscale split and is not state of the art, the predictions still follow the true counts reasonably.

For example, taking an average of overlapping patches of the image during testing, i.e. densely scanning using a stride of 64 pixels, yielded a mean absolute error on the first fold of approximately 640.

D. TRANCOS Traffic Counting

Our network performs very well on the TRANCOS dataset. We see that although the shapes in the density map no longer match as closely to the ground truth density maps, the counts are significantly more accurate than other methods. For training this dataset, we take 80×80 patches which we can stitch back together into the full-sized 640×480 images. As seen in Table III, we achieve state of the art results as measured by the $GAME$ metric [16]. We trained the DCR network with density maps produced with a Gaussian of $\sigma = 15$ as specified in [3].

V. Conclusion

A. Summary

We have proposed the use of dilated convolutions as an alternative to the complicated HydraCNN [3] or Multicolumn CNN [15] for the vision task of counting objects in images. While we largely keep the structure of the dilated convolutions the same as in [1], we use ReLU activations for the purposes

Method	maximal	downscale	upscale	minimal	Shanghai
DCR (without perspective information)	1.63	1.43	**0.70**	2.72	1.64
[3] (with perspective information)	1.65	1.79	1.11	1.50	-
[3] (without perspective information)	2.22	1.93	1.37	2.38	-
[2]	1.70	1.28	1.59	2.02	-
[19]	1.70	2.16	1.61	2.20	-
[20]	1.43	1.30	1.59	1.62	-
[21]	**1.24**	1.31	1.69	**1.49**	-
[6]	1.70	**1.26**	1.59	1.52	1.60
[15]	-	-	-	-	**1.07**
[15]	-	-	-	-	2.16
[15]	-	-	-	-	2.25
[15]	-	-	-	-	2.24
[15]	-	-	-	-	2.07

TABLE I. MEAN ABSOLUTE ERROR OF VARIOUS METHODS ON UCSD CROWDS

Network	Mean Absolute Error
DCR (on 100×100 patches)	9.94
Multicolumn CNN **(on whole images)** [15]	377.4
HydraCNN **(on whole images)** [3]	**337.4**

TABLE II. COMPARING PERFORMANCE ON THE FIRST FOLD OF UCF DATASET

Method	GAME(L = 0)	GAME(L = 1)	GAME(L = 2)	GAME(L = 3)
DCR	**9.05**	15.53	17.25	**17.90**
[3]	10.99	**13.75**	**16.69**	19.32
[2] + SIFT from [16]	13.76	16.72	20.72	24.36
[19] + RGB Norm + Filters from [16]	17.68	19.97	23.54	25.84
HOG-2 from [16]	13.29	18.05	23.65	28.41

TABLE III. MEAN ABSOLUTE ERROR OF VARIOUS METHODS ON TRANCOS TRAFFIC

of regression. We have performed experiments on two different splits of the UCSD crowd counting dataset, the UCF crowd counting dataset, as well as the TRANCOS dataset. We obtain comparable or better results in two of three of these datasets as compared to [3]. In fact, the DCR network, which never uses perspective information in our experiments, occasionally outperformed HydraCNN with perspective information. These results show that the DCR network performs surprisingly well and is also robust to scale effects (the sizes of the cells in the images were varied randomly). Further, the DCR network shows promising results not only on the relatively low density UCSD dataset, but also on the higher density TRANCOS dataset. However, it performs rather poorly on the extremely dense and varied UCF dataset.

B. Future Work

We would like to compare this procedure on other large crowd datasets, specifically those of [6] and [22] for the World-Expo crowd dataset as well as [15] for the Shanghaitech crowd dataset. Further, we would like to attempt other techniques for training the UCF dataset to possibly improve results on highly dense images.

In addition to an analysis of performance on counting, we also plan to examine the ability of these different approaches to locate the objects in the image. As mentioned previously, if the regressor is accurate and precise enough, the resulting density map can be used to locate the objects in the image (and we expect this to outperform more traditional feature/localization-based methods for dense images where features may be difficult to extract). We expect that in order to do this, we will have to regress each object to a single point rather than a region specified by a Gaussian. Perhaps this might be accomplished by thresholding the activations of the final layer. Moreover, we might examine the effect of increasing the depth of the DCR network along with the size of its dilations on the resulting regressed density maps.

Because the results of the dilated convolution network are promising, we might consider extending this work to the case where perspective effects occur in such a way that objects "behind" the current plane of focus should have the same density shape, but a lower density value. This situation arises when looking at smFISH confocal microscopy, which results in a 3D stack of images representing slices of a cell going down in vertical space. If the regressor is able to accurately label the densities in these images, we could potentially use the dilated convolution regression network to feed features to a recurrent neural network and obtain accurate counts of these densely packed 3D single molecules.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. 1359275 and 1659788. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. Furthermore, we acknowledge Kyle Yee and Sridhama Prakhya for their helpful conversations and insights during the research process.

REFERENCES

[1] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[2] V. Lempitsky and A. Zisserman, "Learning to count objects in images," in *Advances in Neural Information Processing Systems*, 2010, pp. 1324–1332.

[3] D. Onoro-Rubio and R. J. López-Sastre, "Towards perspective-free object counting with deep learning," in *European Conference on Computer Vision*. Springer, 2016, pp. 615–629.

[4] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[5] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[6] C. Zhang, H. Li, X. Wang, and X. Yang, "Cross-scene crowd counting via deep convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 833–841.

[7] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1520–1528.

[8] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2758–2766.

[9] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1915–1929, 2013.

[10] G. Lin, C. Shen, A. van den Hengel, and I. Reid, "Efficient piecewise training of deep structured models for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3194–3203.

[11] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille, "Attention to scale: Scale-aware semantic image segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3640–3649.

[12] S. Seguí, O. Pujol, and J. Vitria, "Learning to count with deep object features," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015, pp. 90–96.

[13] E. Walach and L. Wolf, "Learning to count with cnn boosting," in *European Conference on Computer Vision*. Springer, 2016, pp. 660–676.

[14] L. Boominathan, S. S. Kruthiventi, and R. V. Babu, "Crowdnet: A deep convolutional network for dense crowd counting," in *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 2016, pp. 640–644.

[15] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma, "Single-image crowd counting via multi-column convolutional neural network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 589–597.

[16] R. L.-S. S. M. B. Ricardo Guerrero-Gmez-Olmedo, Beatriz Torre-Jimnez and D. Ooro-Rubio, "Extremely overlapping vehicle counting," in *Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, 2015.

[17] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[18] F. Yu, V. Koltun, and T. Funkhouser, "Dilated residual networks," *arXiv preprint arXiv:1705.09914*, 2017.

[19] L. Fiaschi, U. Köthe, R. Nair, and F. A. Hamprecht, "Learning to count with regression forest and structured labels," in *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE, 2012, pp. 2685–2688.

[20] V.-Q. Pham, T. Kozakaya, O. Yamaguchi, and R. Okada, "Count forest: Co-voting uncertain number of targets using random forest for crowd density estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3253–3261.

[21] C. Arteta, V. Lempitsky, J. A. Noble, and A. Zisserman, "Interactive object counting," in *European Conference on Computer Vision*. Springer, 2014, pp. 504–518.

[22] C. Zhang, K. Kang, H. Li, X. Wang, R. Xie, and X. Yang, "Data-driven crowd understanding: a baseline for a large-scale crowd dataset," *IEEE Transactions on Multimedia*, vol. 18, no. 6, pp. 1048–1061, 2016.

Author Index

Keyword Index