Sentiment Manifolds

John Cavalieri
jcavalieri@cs.uml.edu

Santiago Paredes
sparedes@cs.uml.edu

Abstract

In this paper, we propose sentiment manifolds: opinionated information forming locally resemblant Euclidean spaces in its embedding. This paper describes our experience in using a siamese architecture with convolutional neural networks (CNN) for training a similarity metric from labeled data. We present an application of dimensionality reduction given neighborhood relationships between samples in the input space. All this culminates in an approach to solving the task of sentiment analysis in movie reviews.

1. Introduction

The modern film industry teems with box office hits and, unfortunately, total cinematic failures as well. To create a successful film, it's essential that producers and directors understand their target audiences values, interests and personalities. Conveniently, movie reviews can provide a rich understanding of public receptivity to a particular movie. With ever increasing demands to understand target bases, the machine learning task known as sentiment analysis has ushered in fresh ways to analyze large quantities of data in film criticism.

For the purpose of detecting patterns and drawing accurate conclusions about a movie and its audience, we present a new way to analyze a dataset of movie reviews and determine their associated binary sentiment polarity labels, i.e. positive or negative.

2. Related Work

In the past decade, several works have pioneered the use of CNN's for sentiment classification, a most notable work being [4], authored by Yoon Kim. Kim designed a CNN for general sentence classification that used pre-trained word vectors[1] as universal feature extractors that were robust across numerous datasets. This yielded formidable results against deep learning models that utilize more sophisticated layering techniques, such as Dynamic k-Max Pooling (from [3]) and Sentiment Treebanks (from [7]) yet demanded less complexity in the internal design of the CNN. Interestingly enough, Kim also proposed a more focused approach to training his word vectors by tailoring them to specific classification tasks before they were used for training. Kim reported that using these specialized pre-trained word vectors created substantial performance gains in classification accuracy.

Recently still, landmark research surrounding the field of image classification has shown that the use of discriminatively learned similarity metrics in CNN's can create robustness to abstruse geometric distortions, a particularly relevant work being [1]. While the task of classifying movie reviews presents its own set of challenges distinct from image classification (challenges including sarcasm, misspellings, misuse of words, etc.), movie review classification does have a quasi-conceptual equivalence to the geometric distortions we see in the image classification problem. We hypothesize that these distortions include style, diction, tone, and voice – basic elements of writing. Therefore, learning a similarity metric discriminatively as described in [1] can be used to compare or match movie reviews robust to differences in these basic elements of writing.

In this work, we have opted to utilize the tailored pre-trained word vectors alongside a discriminatively learned similarity metric for the task of sentiment analysis.

3. Approach

Our approach utilizes a siamese architecture with CNN's (inspired by [1]) and the k-nearest neighbor algorithm to produce a learned globally coherent non-linear function that maps labeled movie review data to a manifold that exists in a lower dimension than our input space.

3.1. Troubling Properties of Neural Networks

In [8], Christian Szegedy studied state-of-the-art neural networks that generalize well on object recognition tasks – the neural network of focus being a large, deep CNN used to classify 1.2 million high-resolution images from the ImageNet LSVRC-2010 contest. This giant CNN was dubbed "AlexNet."

In Szegedy's research, it was discovered that the stabil-

[1]These word vectors, described in [6], were trained on a 100 billion word corpus from Google News.

ity of a neural network's prediction error could be arbitrarily increased by applying small, imperceptible perturbations to input instances that were initially correctly classified with high probability prior to perturbing – so called adversarial examples. These perturbations were essentially non-random transformations to the input. They were found by optimizing the network's input $X^{(i)} \quad i = 1, ..., m \quad$ to maximize prediction error. This can be achieved with the following algorithm:

1. Train the network in the usual way and hold those weights fixed for the remaining steps

2. Negate the loss: $L_{neg} = -1 \times L(W, X^{(i)}, y^{(i)})$

3. Run gradient descent on the inputs $X^{(i)}$ with update step $X^{(i)} = X^{(i)} - \alpha \times \frac{\partial L_{neg}}{\partial X^{(i)}}$

It is extremely counterintuitive that even a network like AlexNet cannot handle these small perturbations. The following properties exemplify why this is so troubling:

- AlexNet is known to generalize well
- The pixel distance between $X^{(i)}$ and $X^{(i)}_{perturb}$ is such that $||X^{(i)} - X^{(i)}_{perturb}||_2 < \epsilon$
- The class of $X^{(i)}$ is $y^{(i)}$ and imperceptible perturbations do not change class label; yet

  $P(y^{(i)}|X^{(i)}, W) \approx 1$

  $P(y^{(i)}|X^{(i)}_{perturb}, W) \approx 0$

If AlexNet is capable of generalizing to input instances unseen during training and correctly predicting their class labels, then why is it failing to correctly predict the class of $X^{(i)}_{perturb}$? We contend that the answer to this question lies in the Szegedy's research that this is in fact a characteristic of neural networks. To be exact, the issue exists in the nature of how the network learns a linear decision boundary in the space generated by the output of the last layer. The error is computed based on this linear decision boundary, and that error is back propagated to W in the network during training. We hypothesize that forcing the network to choose W such that the final learned feature representation in the last layer is linearly separable may be too demanding of a requirement. If the network is struggling to meet this requirement during the learning process, it may choose weights that create blind spots (further discussed in Section 3.2) around the decision boundary where the network will misclassify instances like $X^{(i)}_{perturb}$ despite that fact that $X^{(i)}$ is classified correctly.

Figure 1. These are examples of imperceptible perturbations (images from [8]). The original images and their perturbed counterparts are barely different.

3.2. Blind Spots & Manifold Hypothesis

In [8], Szegedy found that the adversarial examples are, in themselves, robust. Adversarial examples – which fall into their own respective blind spots – generated from one particular network trained on one particular dataset statistically fall into blind spots of completely different networks trained on entirely different datasets, implying this phenomena of adversarial examples is also some intrinsic property of neural networks. We have hypothesized that because the perturbed instances' class are unchanged, they are in the same neighborhood of other instances of the same class on the manifold on which they exist. This perspective relies on the manifold hypothesis which states that natural data tend to lie on a manifold with less dimensions than the input space. With these assumptions, blind spots can be mitigated by employing a type of k-Nearest Neighbors (k-NN) on the output of a siamese network (discussed in Section 3.3) with a learned distance metric. Under the above hypothesis, this method is robust to adversarial examples and blind spots because the perturbed instance will be mapped to a position on the manifold where it will be correctly classified based on its distance to other instances of the same class via k-NN, eliminating the use of a linear decision boundary.

3.3. Siamese Architecture Learns Distance Metric

Having developed a better understanding of the problem and a way to conceptualize it through the manifold hypothesis, we now take a look at the architecture of our model, as illustrated by Figure 2.

A basic scan of the architecture should engender the ultimate intuition that a comparison will take place near the final output of the model (see Figure 2). While the motivation for this will be addressed shortly, it is helpful to remember we're proceeding towards this final comparison as we step through each component of the architecture.

Attention should immediately be drawn to the input of the architecture, specifically, there are *two* inputs, X_1 and X_2, being fed into the model. There are two generic cases to consider concerning potential input combinations between X_1 and X_2:

1. They are sampled from the same class, forming a "genuine" or "similar" pair i.e. two positive reviews,

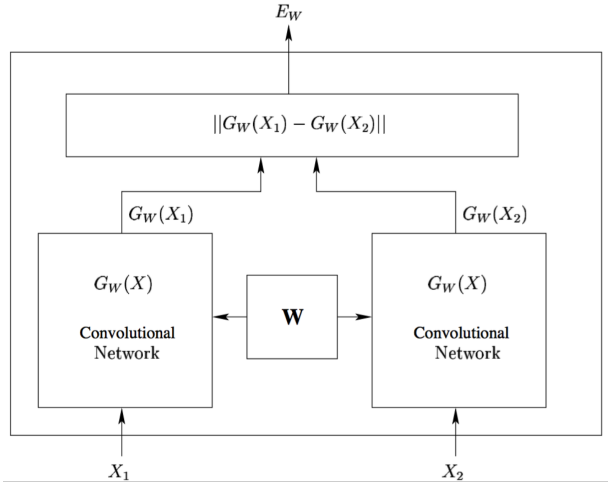

Figure 2. Architectural illustration taken from [1]

$X_1 = pos$ and $X_2 = pos$, or two negative reviews, $X_1 = neg$ and $X_2 = neg$.

2. They are sampled from different classes, forming an "impostor" or "dissimilar" pair i.e. a positive review and a negative review, $X_1 = pos$ and $X_2 = neg$ or vice versa.

The inputs are fed into seemingly separate CNN's with two distinct characteristics:

1. They are structural twins (they have the same exact layers and layer configurations), hence they are both defined by the family of functions $G_W(X)$.

2. They share a weight matrix, W, such that, during training and optimization, when one CNN tweaks an arbitrary weight, the other CNN is affected by the same adjustment. This is necessary because both branches of the siamese network are fed movie reviews and any feature learned by one branch should be a feature learned by the other branch – weight sharing achieves this.

The result is an *identical* arrangement of *identical* CNN's in the architecture, producing two high level representations, $G_W(X_1)$ and $G_W(X_2)$, of the original input pair. Recall that these new representations of the input will be compared in the final step of our architecture. It would be appropriate then to discuss what this comparison, or similarity metric, will achieve.

Let Y be a binary label of an input pair, where $Y = 0$ for a genuine input pair and $Y = 1$ for an impostor input pair. The final step of the architecture defines the following "energy" function:

$$E_W = ||G_W(X_1) - G_W(X_2)||$$

.

This (fundamentally Euclidean distance) function measures the "compatibility" of an input pair based on the similarity of its high level output from the CNN region. Therefore, as $G_W(X_1)$ and $G_W(X_2)$ are more similar, their associated energy approaches zero. Conversely, as $G_W(X_1)$ and $G_W(X_2)$ are dissimilar, their energy increases. Hence, the label for a genuine pair is $Y = 0$ (they belong perfectly to the same class and should have an energy of zero) and $Y = 1$ for an impostor pair (they belong entirely to different classes and should have heightened energies).

As we intentionally feed genuine and impostor pairs into the architecture during training, we're laboring towards solving an intrinsically symmetric problem – training two branches to learn features simultaneously – and therefore, the use of siamese networks to create an invariant similarity metric is justified.

3.4. Contrastive Loss

Ultimately, we'll need to define a loss function that, when minimized, promotes low energy output for genuine pairs and high energy output for impostor pairs. [1] suggests an elegant solution:

$$\begin{aligned} L(W) &= \sum_{i=1}^{P} L(W, (Y, X_1, X_2)^i) \\ L(W, (Y, X_1, X_2)^i) &= (1 - Y) L_G(E_W(X_1, X_2)^i) \\ &\quad + Y L_I(E_W(X_1, X_2)^i) \end{aligned} \tag{1}$$

The loss function is composed of two partial loss functions: L_G and L_I. Intuitively, defining the partial loss functions in the following way produces ideal behavior in optimization:

- L_G - a monotonically *increasing* function
- L_I - a monotonically *decreasing* function

In training, minimizing $L(W)$ drives E_W down for $Y = 0$ and drives E_W up for $Y = 1$. The net effect on the manifold is that similar reviews are driven tightly together and dissimilar reviews are driven far apart. This pushing and pulling of the manifold is known as contrastive loss.

3.5. CNN Structure

Figure 3 illustrates $G_W(X)$ as it is defined in our current CNN model.

$G_W(X)$ begins with an efficient embedding layer which maps our vocab indices into the dimension of a dense input embedding. Then we proceed to stack dropout, 1-D convolutional, and 1-D max pooling layers in the network, using the ReLU activation function throughout. In the final

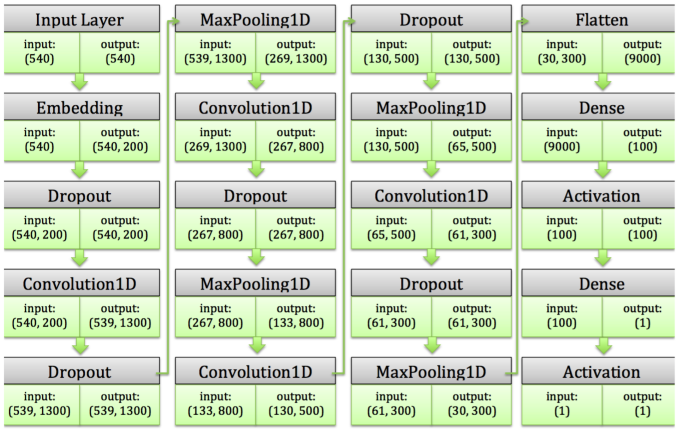

Figure 3. Structure of $G_W(X)$

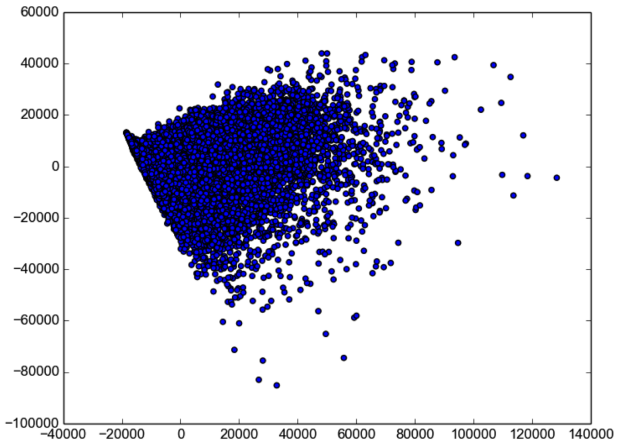

Figure 4. PCA of IMDB movie reviews with top 2 principal components

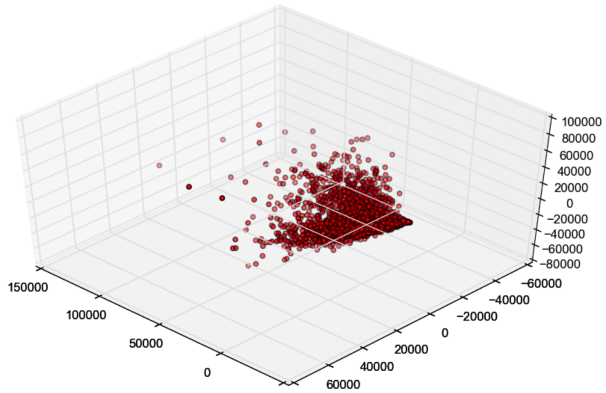

Figure 5. PCA of IMDB movie reviews with top 3 principal components

layers of the network, we flatten the output so that we can project the output onto a single-unit output layer, ultimately squashing it with a Sigmoid activation layer.

4. Dataset

We used a dataset adapted from the [5] dataset consisting of 12,000 highly polarized movie reviews for training and 11,000 for testing.[2] This dataset was intended for binary sentiment classification.

We performed several analyses on the data using Python's Natural Language Toolkit (NLTK) and Sklearn. We discovered that positive reviews contained more words on average than negative reviews (see Table 1).

Average negative review length	1,310
Average positive review length	1,360
Max positive review length	13,600
Max negative review length	8,729

Table 1. Notable quantitative characteristics of data (measured in words)

Reviews were most distinctly characterized by two sets of polar words: positive and negative. Positive words included words like *funny, family, better, wonderful, performance, plot, excellent, beautiful, director, amazing*, while negative words included *bad, worst, plot, didnt, terrible, poor, wrong*. Using dimensionality reduction to help visualize the dataset, it became clear, as was suspected, that the dataset in its original space had no clear structure that could benefit sentiment analysis, as can be seen in Figure 4. Another view can be seen in Figure 5 (a 3D version of 4).

To adapt movie reviews into encodings capable of being fed into our network, we converted each review to a matrix of one-hot vectors, so that the first row of the review matrix corresponds to the first word of the review. We had decided to truncate all reviews to 532 words based on statistical analysis–any review that contained more words was padded with zeros. Our vocabulary length was the total number of unique words seen during training in addition to some punctuation marks and a special "UNK" token for out-of-vocabulary words.

[2] The subset of [5] can be found here: `https://goo.gl/1of8KR`

5. Experiments and Results

We tested seven different architectures with slight differences on the IMDB movie review dataset, the results of the best of which are reported in Table 2. In each architectural iteration, we discovered certain properties related to the reaction of neural networks to the IMDB dataset. Our choice to use CNN's was partly motivated by recent research showing state-of-the-art results on sentiment classification tasks in addition to the notion that we could capture n-grams with the filter size of each CNN layer. The first layer filter size is 2, capturing bigrams from the raw data. The next filter size is 3, capturing trigrams from the sequence of learned

bigrams. Finally, our last CNN layer filter size is 4, capturing 4-grams from the sequence of learned trigrams from the previous layer; all strides are equal to one.

We also found the standard guard against over-fitting, $l2$ regularization, proved to negatively affect performance for this task. We found that the random dropout technique yielded substantially better results. We also discovered that progressively increasing the probability of dropout from the first hidden layer to the last was effective. This seems logical given that a high dropout on the first hidden layer would be randomly removing information from the raw data that could be useful for classification further down the pipeline.

Despite the many examples of CNN's terminating with a few fully connected layers, we found that anything more than a single fully connected layer at the end of our CNN reduced accuracy. It may be the case that further testing would have revealed a different result but we had neither the time nor the computing power to confirm this.

Accuracy	89.6%
Precision	88.9%
Recall	90.0%
F1	89.6%
Kaggle Score	87.7%

Table 2. Results

After testing, there was significant evidence from data visualizations, both in the embedded vocab (see Figure 6)[3] and in the reviews mapped onto the manifold (see Figures 7, 8, and 9)[4], to strongly suggest that positive and negative reviews were tightly separated on the manifold as a result being processed by $G_W(X)$. This not only affirms the sentiment manifold hypothesis, but it provides for us an opportunity to locally approximate and compute a non-parametric method for carrying out the task of movie review classification. The ultimate significance of this is discussed in Section 6.

6. Future Work

6.1. k-Nearest Neighbors Algorithm Actualizes the Globally Coherent Non-Linear Mapping Function

In order to create an even better mapping that is invariant to syntactic translations of the input, [2] suggests learning a mapping with temporal neighborhoods. In testing, the siamese architecture could actually utilize a non-parametric method to create a more expressive and robust decision boundary. In fact, this is the precise direction in which we wanted to proceed from the onset, however, more time was

[3] The points on this graph are the output of the first hidden layer.

[4] The points on these graphs are the output of the last hidden layer for each review, in other words, the points are the output of $G_W(X)$ for each review.

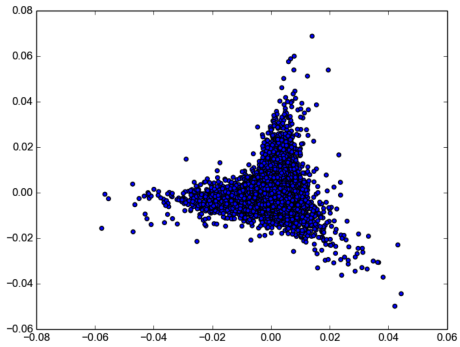

Figure 6. Vocab embedded in $R^2 00$

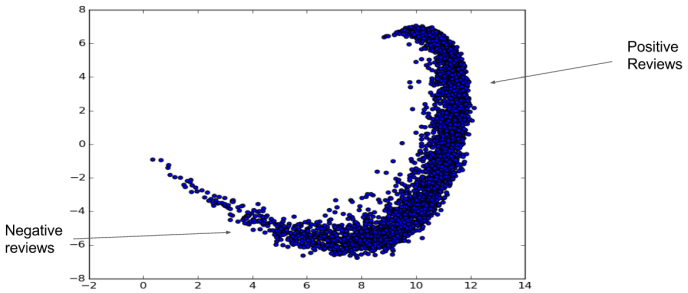

Figure 7. Reviews mapped onto manifold by $G_W(X)$

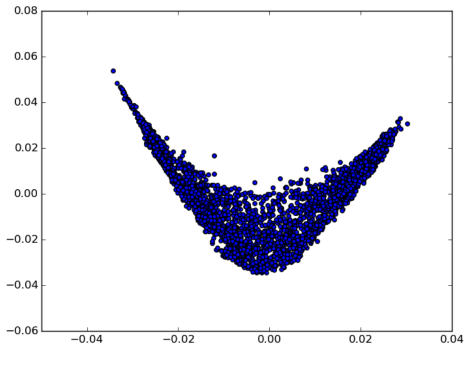

Figure 8. Different view of Figure 7

needed to bring this concept to fruition, and so at this point in time it remains in development.

Nevertheless, our CNN could accomplish the mapping function described above simply by removing the last softmax layer and cutting the CNN short at the final dense layer with 100 nodes. This neural pruning is illustrated in Figure 10.

The result would be a feature space where k closest training examples were used as input for the final task of classification. Of course, this exactly describes the k-NN algorithm discussed in Section 3.2, in which a movie review can be classified by a majority vote of its neighbors, ultimately

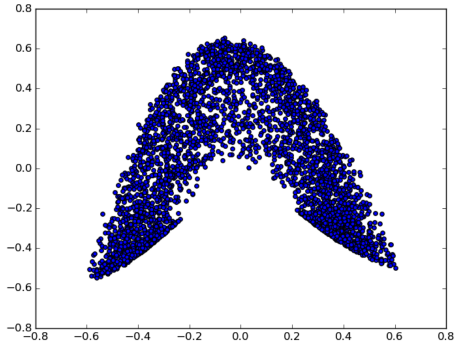

Figure 9. Another view of Figure 7

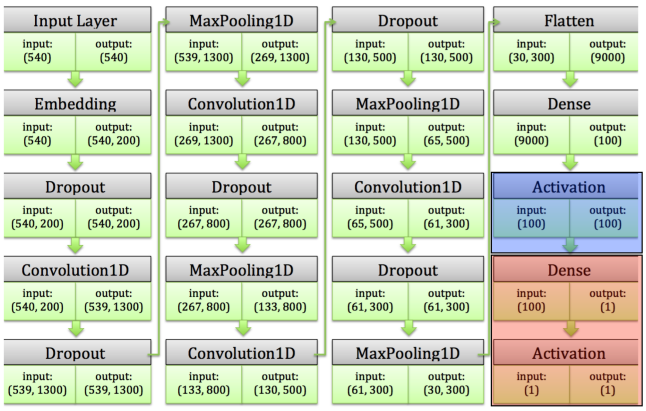

Figure 10. The globally coherent non-linear mapping function is actualized when the softmax and final projection layer are removed (demonstrated in red) from $G_W(X)$. The new $G_W(X)$ ends with a "vanilla" activation layer (demonstrated in blue) producing a final dense layer of 100 nodes.

receiving an assignment to the class most common among its k nearest neighbors. In theory, this would yield a notable improvement in the efficacy of the movie review sentiment classifier.

It should be noted that because this is a binary classification problem, k should be an odd number to eliminate the issue of tied votes.

7. Conclusion

We demonstrated an effective method of using CNN's on sentence classification tasks, in our case sentiment classification. We designed our network to progressively capture n-grams of higher order using the filter size of each layer–starting with bigrams on the raw movie reviews in one-hot matrix format then building trigrams and 4-grams from learned representations of movie reviews from previous layers.

As suggested by our own experiments and research, an architecture motivated by the manifold viewpoint could be useful in accurately classifying data with neural networks.

8. Acknowledgments

We would like to thank Professor Kate Saenko for her direction on developing this research project.

For this project, John was the project lead, and was therefore responsible for providing vision as he guided research and development throughout the project. Santiago responded accordingly, following John's lead and also ensuring that source code, presentations, and papers were documented professionally and accurately.

Peer programming was utilized during the bulk of development stages.

References

[1] S. Chopra, R. Hadsell, and Y. Lecun. Learning a similarity metric discriminatively, with application to face verification. In *In Proc. of Computer Vision and Pattern Recognition Conference*, pages 539–546. IEEE Press, 2005.

[2] R. Hadsell, S. Chopra, and Y. Lecun. Dimensionality reduction by learning an invariant mapping. In *In Proc. Computer Vision and Pattern Recognition Conference (CVPR06*. IEEE Press, 2006.

[3] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. *CoRR*, abs/1404.2188, 2014.

[4] Y. Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

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

[6] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013.

[7] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Stroudsburg, PA, October 2013. Association for Computational Linguistics.

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