On the Use of K-Competitive Networks for Writing Style Transfer

Cameron Ibrahim
Cornell University
cai29@cornell.edu

Yuji Akimoto
Cornell University
ya242@cornell.edu

Ryan Butler
Cornell University
rjb392@cornell.edu

Luca Leeser
Cornell University
ll698@cornell.edu

ABSTRACT

Writing style is a key component of the quality of any online review - a detailed and informative review is only useful insofar as it is able to retain a reader's attention. In this paper, we explore the use of k-competitive layers in convolutional neural networks to classify the author of a given piece of text. We then apply the techniques used in neural style transfer for images to transfer the writing style of one author onto the content written by another. Given sufficient examples of well-written reviews, our techniques can be used to improve the quality of reviews on online platforms such as Yelp or Amazon.

1. INTRODUCTION

Neural style transfer [5] has proven very successful at transferring the artistic style of an image onto the content of another. Due to the layered structure of convolutional neural networks, it was observed that later layers in the network produce feature representations of the content of an image, while the correlations between these features give a good representation of artistic style. More specifically, given a content image and style image, the algorithm treats the input to the network as a variable and simultaneously minimizes the ℓ_2 norm between the input and content image's feature representations, and the input and style image's style representations.

Figure 1: Cornell's Gates Hall stylized in the form of Vincent Van Gogh's *Starry Night*.

Modifications that optimize computational time [8], preserve the texture of the style image [6], and preserve facial structures in the content image [10], among others, have further advanced the art of neural style transfer for images. However, applying similar techniques to text has proven difficult for a number of reasons. Firstly, textual data lives in an extremely sparse space, and therefore it is difficult to iteratively produce better stylized text while avoiding phrases that make no sense. Secondly, style and content are much more closely linked in text than they are in images - the same writer may write two passages of text in a completely different tone, yet maintain the same underlying style in a manner that is very subtle.

In this paper, we explore the use of k-competitive convolutional layers, a generalization of k-competitive layers designed for autoencoders in [2], which in turn are based on k-sparse layers as defined in [16]. We hypothesize that this technique will allow us to mitigate the high-dimensional sparsity of textual data and reduce the problem of writing style transfer to the well-established field of image style transfer. We integrate k-competitive layers into preprocessing and training of a word-level convolutional neural network [11], and perform a qualitative analysis of the quality of content and style representations generated by these networks. A key benefit of our method is that it does not require a training set of translations between writing styles, which are very difficult to obtain.

2. RELATED WORK

Early methods at quantitatively determining the writing style of a piece of text focused on statistical analyses of the rates of passive and active sentences, the ratio of pronouns to nouns to adjectives, and the likelihood of adjacent words given their distribution in the English language. However, these methods rely on unreliable tokenizations and are difficult to adapt to style transfer. Xu et. al. [21] provided some of the first algorithmic style transfer methods between Shakespearean and modern English, but relied on a phrase table and did not take advantage of modern machine learning methods.

Recent methods based on neural networks fall into two broad categories: those that treat the problem as a monolinguistic machine translation problem, and those that attempt to generate style and content representations of text. [1] and [7] fall under this first category, and generate stylized text that perform well according to BLEU score [18]. However, these translation-based methods require a large dataset of tupled phrases that are identical in content and differ only in style. Use of passages in the Bible in [1] and SparkNotes' Shakespearean translations in [7] present interesting but impractical test cases, while generating rap lyric translations

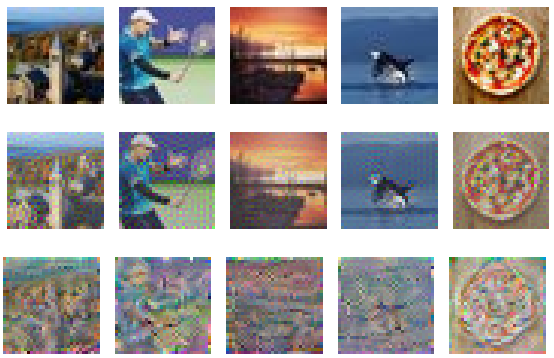
with Google Translate via French [7] leaves much to be desired in terms of accuracy.

[3] and [9] both consider a similar method to ours in terms of generating style and content representations of text, and [9] defines three evaluation criteria: soundness (preservation of content), coherence (grammatical correctness & common sense), and effectiveness (matches intended style). However, these lack sufficient experiments with which to evaluate the quality of their results. [4] provides a quantitative assessment of their content and style representations, in addition to a more complex alternative architecture. All of the aforementioned works use a recurrent neural network encoder-decoder structure (otherwise referred to as a "seq2seq" model), with either a long short-term memory (LSTM) or gated recurrent unit (GRU) used as the underlying structure. These methods seek to encode the content of the input text into a low dimensional space, and then use the decoder to generate text in a specified style. Our key contribution is in the use of convolutional neural networks, which have proven very successful at tasks such as sentiment analysis [11].

3. PRELIMINARY ANALYSIS

The majority of implementations of style transfer for images use the VGG-19 [20] architecture to generate content and style representations of an image. This network was trained to classify images into one of the one thousand classes of the ImageNet challenge dataset [19] and therefore is able to generate informative feature representations of a very large variety of objects. Given the resources required to train a similarly robust classifier for text, we first explore the tradeoffs of using an underlying network that is trained to discriminate between a much smaller number of classes by training a VGG network with the CIFAR-10 dataset [13] (here, we use pre-trained weights provided by [14]). Although this modified network is far less generalizable in terms of classification, we found that the content representations provided by later layers of the network were still informative enough to reconstruct the original image, even for images containing objects not included in the ten classes of CIFAR-10. By reconstruction, we refer to treating the input to the network as a variable, and minimizing ℓ_2 loss between the input image and original image's feature representations in a certain layer.

Figure 2: Several 32x32 images (top) reconstructed in the conv2_2 layer (middle) and conv4_2 layer (bottom) of VGG-19 trained on CIFAR-10 data; maximum 10000 iterations.



In the reconstruction phase, we use average-pooling in place of max-pooling, as suggested in [5]. Although none of the images in Figure 2 resemble any of the ten classes present in CIFAR-10 data, the network still produces highly informative feature representations through convolution. This is a key observation in making the transition to text data, as writing style is subjective and therefore it is difficult to collect data from a large number of distinct classes that span the entire space of writing styles.

However, we also observe that this reduction in generalizability of the underlying network architecture is not without tradeoffs. When using conv4_2 as the layer at which to match the feature representations of two images (the layer used in Figure 1), the reconstructed images are mostly unrecognizable, and it takes a much larger number of iterations to produce something even remotely close to the original. However, we believe that there is enough signal even within the unrecognizable images, and with further hyperparameter optimization we will be able to mitigate this loss in reconstruction accuracy. This justifies the use of networks for author classification that have been trained on a limited number of classes as a starting point.

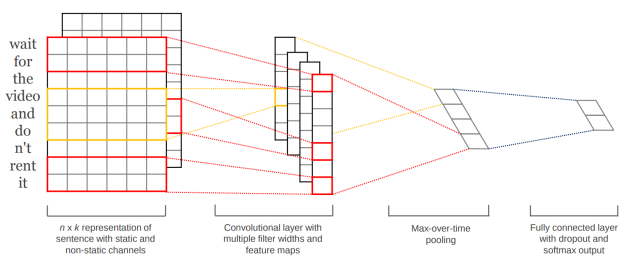
4. MODEL

4.1 Convolutional Neural Networks for Text Classification

Although recurrent neural networks have proven adept at language recognition tasks, evaluating the overall content and style of written text is a task that is less dependent on sequential information (outside the scope of a receptive field), and a hidden state is likely too small to encode enough information for our purposes. We believe that a a convolutional neural network allows us to take advantage of the success of style transfer for images, in contrast to aforementioned approaches that treat style transfer as a machine translation problem.

As the baseline for our comparisons we use a word-level convolutional neural network as designed by Yoon Kim in [11]. A 1-d convolutional layer varies from those commonly found in architectures for image classification in that the receptive field of a filter is one-dimensional, and subsequent pooling layers operate along the time axis. The network takes as input a sentence, with each word encoded as a vector. As a "shallow" architecture, the network performs layer of convolution with filter lengths of 3, 4, 5 with 100 feature maps each, before a max-over-time pooling operation and a softmax layer.

Figure 3: Structure of the baseline convolutional neural network model.

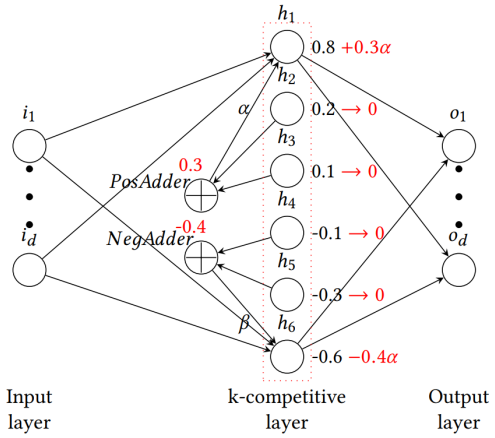


4.2 K-Competitive Convolutional Layers

It has been observed that encouraging sparsity when learning representations acts as a capable reguarlizer that improves performance on classification tasks. In [16], Makhzani and Frey designed the k-sparse autoencoder and demonstrated its effectiveness as a preprocessing step on image data. In the feedforward phase, sparsity is used as the sole non-linearity: after computing the activations $z = W^T x + b$, only the largest k entries of z are kept and the rest are set to 0.

Since textual data often has high-dimensionality, sparsity, and power-law distributions [2], autoencoders for text have often learned trivial features such as proper nouns specific to a certain passage. Chen and Zaki made a modification to k-sparse layers by introducing competition in the place of truncation: after computing the activations $z = tanh(W^T x + b)$, only the k entries with the largest absolute value are kept non-zero. These k most "competitive" neurons are then augmented (with an amplification constant) with the **energy** of the "losers" that were made inactive, where the energy of a subset of neurons H is defined as $\sum_{h_i \in H} |z_i|$.

Figure 4: A fully connected k-competitive layer.



Here, we propose a generalization of k-competitive layers to convolutional layers. Let z be the activation volume produced by a convolutional layer, where $z_{x,y,:}$ denotes the feature vector at spatial location (x, y). Then for each spatial location (x, y) we select the k most active features by absolute value. These k most active features may all have been originally positive or negative, which contrasts with the implementation in [2] which requires that half of the k "winners" to have been positive and half negative. Let I_{xy} denote the indices of the k "winner" filters at spatial location (x, y), and define the energy of a subset of filters H at (x, y) by $E_{H_{xy}} = \sum_{h_i \in H} |z_{xyi}|$. Then the output of a k-competitive layer is given by:

$$kcomp(z_{xyi}) = \begin{cases} z_{xyi} + \alpha \; sign(z_{xyi}) E_{I_{xy}} & \forall i \in I_{xy} \\ 0 & \forall i \notin I_{xy} \end{cases}$$

where the amplification constant, α is a hyperparameter. This is equivalent to conducting the k-competitive algorithm for a fully connected network across the feature vector at a specific spatial location.

5. EXPERIMENTS

5.1 Datasets

We use the most prolific reviewers on Yelp[1] to generate a large amounts of text for various styles of writing. Although the reviews for a frequent user of Yelp may span a number of years, we make the assumption that a user's writing style remains constant over that period of time. This may not strictly be the case as language, particularly of the colloquial variety that is often found online, evolves over time and certain words serve as clear indicators of certain time periods. However, dealing with a time-varying vocabulary set is an extension that we have not yet considered. After identifying the most frequent reviewers, we tokenize all of their reviews using NLTK [15] to generate a corpus. To create a baseline task, we also create a corpus of Shakespearean works from Project Gutenberg[2].

5.2 Comparison of Architectures

We explore the use of k-competitive layers in two settings: in an autoencoder to generate word embeddings, and as a non-linearity after convolutional layers.

5.2.1 K-Competitive Autoencoder for Embeddings

In natural language processing tasks it is common encode textual data into a sparse space before generating embeddings that project these word or sentence-level vectors into a lower dimensional space. The k-competitive autoencoder for text (KATE) proposed in [2] generates word embeddings that compare favorably to the more commonly used word2vec [17], and we explore its use as a preprocessing step for an convolutional author classifier network. The two variants that we explore are in the initial sparse representation of training data: we compare between the use of a sentence-level log-normalized word count vectors similar to [2] and word-level one-hot encoded vectors.

Since the autoencoder is used to generate vector embeddings of words, it can be argued that using sentence-level embeddings as training data creates a model that is asked to perform different tasks at training and test time. At the same time, these log-normalized word count vectors contain entries between 0 and 1, and can therefore be interpreted as probabilities of potentially overlapping classes. Compared to the one-hot encoding approach that treats words as belonging in mutually exclusive classes, this approach more accurately reflects the reality that words often have overlapping meanings. The amount of training data produced from a single passage of text depends on the relative magnitudes of the number of unique words and number of unique sentences, but given an even ratio the word count approach likely produces more useful data - a single word is much more likely to be meaningless noise than a sentence.

Consistent with the probabilistic interpretation mentioned above, we swap the sigmoid activation function used in the word count approach for a softmax function when using one-hot encoding. We explore the tradeoffs between these two approaches, and quantitatively assess their performance by comparing reconstruction accuracy of the autoencoder, that is, the ℓ_2 loss between the input vector and the decoded vector of its low dimensional representation.

[1] `https://www.yelp.com/dataset/challenge`
[2] `https://www.gutenberg.org/`

5.2.2 *Interpretation of Weight Matrix for Nearest Neighbors Lookup*

For both the bag of words and one-hot versions of KATE, it is useful to perform nearest neighbors lookups on the embeddings and understand how they relate to the weights W for the first fully connected layer in the encoder portion of the autoencoder. Using the one-hot encoding h of any word in the vocabulary and performing Wh yields s, the scores that result from the matrix multiplication. Because h is a vector of all zeros except the index that corresponds to the chosen word, this is equivalent to indexing W along its columns by the index of the word in the vocabulary. This means that each column of W can be directly interpreted as the raw embeddings for each word in the vocabulary before passing them through the sigmoid function. We will refer to s as the "raw embeddings" (in contrast to the "embeddings" which are equivalent to the raw embeddings after being passed through the sigmoid function). To perform the nearest neighbors lookup on any particular word, simply generate its corresponding raw embedding by getting the scores that are generated by the encoder (before the sigmoid function) from the one-hot verson of the word. Then normalize that vector, as well as each column of W. Then, perform $W^T h$ to compute the dot product of h with each column of W. This will result in a vector of similarities that compare s, the raw embedding, with every other word's raw embedding. Selecting the n largest values will then yield the n nearest neighbors, using cosine similarity as the metric for comparison.

5.2.3 *K-Competitive Layer as Activation Function*

Both convolutional and fully connected k-competitive layers are non-linear functions and are therefore appropriate as activation functions. Since they set the activations of the least active neurons to 0, these act similarly to a rectified linear unit with an adaptive threshold. As with ReLU, this property enforces sparsity among the coefficients and therefore acts both as a regularizer and non-linearity. Meanwhile, the allocation of energy from "loser" to "winner" nodes prevents these neurons from becoming entirely dead and preserves gradient flow in backpropagation. Since the baseline convolutional architecture is shallow and only has one convolutional layer (albeit many filters), we add one k-competitive layer following the convolutional layer, with a ReLU layer following it to preserve the non-linearity at inference time (the k-competitive layer is defined as the identity function at test time in [2]).

We compare the performance of the following model configurations on two author classification tasks: the baseline convolutional network with one-hot encoded vectors as input; the baseline convolutional neural network with KATE embedded vectors as input; a convolutional neural network with a k-competitive layer taking one-hot encoded vectors as input. The classification tasks are: the baseline task of differentiating Shakespearean text from the most prolific Yelp reviewer; differentiating between the two most prolific Yelp reviewers.

5.3 Hyperparameters and Training

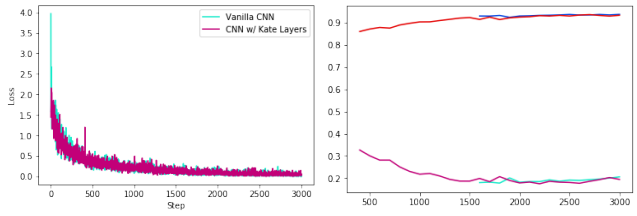
All models were implemented in TensorFlow and trained on a TITAN Xp GPU. Our implementation of the convolutional neural network from [11] is credited to Denny Britz[3]. Both variations of the k-competitive autoencoder were trained with a bottleneck layer of size 128, $k = 32$, amplification constant $\alpha = 6.26$ [2], learning rate 0.001, 100 epochs, and a batch size of 128 using the Adam optimizer [12]. All variations of the convolutional network were trained with dropout rate 0.5, ℓ_2 regularization parameter $\lambda = 0$, learning rate 0.001, 200 epochs, and a batch size of 64 using the Adam optimizer. Variations with the k-competitive layer used $k = 32, 64$ and $\alpha = 6.26$.

6. RESULTS

6.1 K-Competitive Layer Activation

The results of the comparison between the general CNN for text classification and the CNN with K-Competitive activation introduced for its filters are largely inconclusive. Both models converged rapidly towards an accuracy rating nearing or at 100% for the vast majority of samples, achieving a loss regularly below 0.01 within 3000 steps of the 80000 step training process. By plotting the loss of two models over time during training, we see a nearly identical trend in both models, while what differences do exist may be attributed to the random sampling that occurs during the training process of each model. Similarly, during the evaluation that would occur every 100 epochs, both models had nearly identical trends in both accuracy and loss.

Figure 5: Training (left) and validation (right) results for the basic CNN for text recognition (blue) and the CNN with added K-Competitive Layers (red).



These findings would suggest that the K-competitive activated layers had almost no impact on the performance of the model. The plotted results show neither a benefit, nor a detriment to the performance of the model. This would suggest that K-competitive layers have no effect whatsoever on the model, it's ability to deal with sparsity, or the meaningfulness of the features it has learned. This may suggest that the base model was in fact too effective, so that its performance made unnecessary any benefits or detriments that the k-competitive activations could possibly have had. However, although the k-competitive layers seem to have negligible impact on the accuracy of the CNN, it does not mean that they wouldn't help the quality of any text reconstruction. Further experimentation is required to determine what, if any, impact the k-competitive layer has on the quality of the reconstructions for text.

7. ACKNOWLEDGMENTS

[3]`https://github.com/dennybritz`

The authors would like to thank Professor Thorsten Joachims at the department of Computer Science at Cornell University for his guidance, and their fellow members of the Cornell Data Science NLP Research Team and Deep Learning Team for their support.

8. REFERENCES

[1] K. Carlson, A. Riddell, and D. N. Rockmore. Zero-shot style transfer in text using recurrent neural networks. *CoRR*, abs/1711.04731, 2017.

[2] Y. Chen and M. J. Zaki. Kate: K-competitive autoencoder for text. *KDD*, 2017.

[3] T. Edirisooriya and M. Tenney. Applying artistic style transfer to natural language.

[4] Z. Fu, X. Tan, N. Peng, D. Zhao, and R. Yan. Style transfer in text: Exploration and evaluation. 2017.

[5] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.

[6] L. A. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis and the controlled generation of natural stimuli using convolutional neural networks. *CoRR*, abs/1505.07376, 2015.

[7] S. W. Jang, J. Min, and M. Kwon. Writing style conversion using neural machine translation. 2016.

[8] J. Johnson, A. Alahi, and F. Li. Perceptual losses for real-time style transfer and super-resolution. *CoRR*, abs/1603.08155, 2016.

[9] J. Kabbara and J. C. K. Cheung. Stylistic transfer in natural language generation systems using recurrent neural networks. 2016.

[10] P. Kaur, H. Zhang, and K. J. Dana. Photo-realistic facial texture transfer. *CoRR*, abs/1706.04306, 2017.

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

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[13] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[14] S. Liu and W. Deng. Very deep convolutional neural network based image classification using small training sample size. 2015.

[15] E. Loper and S. Bird. Nltk: The natural language toolkit. In *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics*, 2002.

[16] A. Makhzani and B. J. Frey. k-sparse autoencoders. *CoRR*, abs/1312.5663, 2013.

[17] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[18] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. 2002.

[19] 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)*, 115(3):211–252, 2015.

[20] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[21] W. Xu, A. Ritter, W. B. Dolan3, R. Grishman, and C. Cherry. Paraphrasing for style. 2012.

APPENDIX

All code associated with this paper can be found here:
`https://github.com/CornellDataScience/Yelp-FA17/dl_style_transfe`