CS230

Style Verification - Differentiating Artists Based on Their Style

Josef Malmström
josefmal@stanford.edu
Stanford University

Albin Andersson Jagesten
alban567@stanford.edu
Stanford University

***Abstract*—Since famous paintings can be incredibly valuable, there is a demand for reliable forgery detection techniques. In the past decade, attempts have been made at developing such techniques using concepts from the field of deep learning. In this paper, we introduce a novel approach to using deep neural networks for classification of pairs of paintings as either being painted by the same artist, or painted by different artists. We are able to show that by combining the concepts of neural style transfer and face verification, we can produce a model which is able to differentiate between the styles of a limited number of artists with very high accuracy. The derived model in its current state does not seem to generalize to a larger number of artist, but further work is needed to fully evaluate the approach.**

I. Introduction

Since before ancient Rome there has been skilled forgers copying the works of famous artist. As of today, forgery is still a prevalent problem in the art community. A single Picasso piece could typically be valued at about 100 million dollars [1]. Thus finding a reliable way of detecting a forge could hold massive economic value. Modern painting authentication techniques primarily consist of historical, stylistic and forensic analysis performed by art forgery experts [2]. Since these techniques are generally expensive and time consuming in nature, there is an interest in finding more efficient methods of authentication. In the past decade deep neural networks have proven to be very effective at solving problems in the domain of computer vision (e.g. object detection and face recognition). Given their success in other areas, there has also been attempts to design learning algorithms to be used for painting authentication.

This was also the inspiration for the competition "Painter by Numbers" hosted on the site Kaggle in 2016 [1]. The competition challenge consisted of designing an algorithm which, given two images of paintings, can tell whether the paintings were made by the same artist or not.

Past approaches to this problem from a deep learning point of view have mainly consisted of training a classifier that, for each painting, classifies which artist the painting was painted by. This approach has been proven to generalize well to paintings that the classifier was not trained on but that were painted by artists the the classifier is familiar with. However, a significant limitation of this approach is that it will not be able to generalize to paintings by artists who's artwork the classifier has not been trained on.

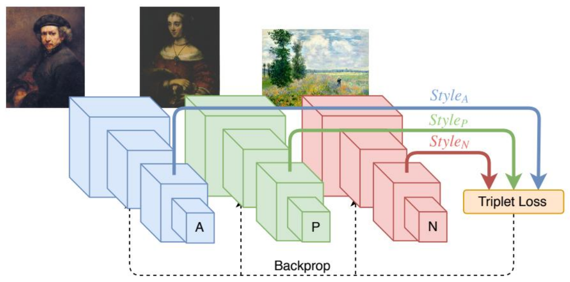


Fig. 1. A triplet of images (anchor, positive and negative) is fed through a siamese CNN, and an encoding of the style of each image is extracted. The network is then trained on the triplet loss of the image styles.

In an attempt to design an algorithm which could potentially be able to generalize beyond the artists in the training set, we have implemented a methodology for this problem that combines the concepts of two well known computer vision applications: neural style transfer and face verification. The general structure of the architecture used in this approach is shown in Figure 1. The method can be outlined as follows: From a training set of paintings we form triplets, each consisting of a base painting (anchor), another painting by the same artist (positive) and a third painting by a different artist (negative). When training, the paintings of a triplet is fed through a set of pretrained siamese CNNs and an encoding of the style of each painting is extracted (much like in neural style transfer). We then train the network further, on the triplet loss formed in terms of the style encodings of the anchor, positive and negative paintings. To evaluate if two paintings were made by the same artist, the paintings are fed through the same, trained network and the distance between their style encodings are compared.

In this paper we will go through the details of this approach and evaluate its performance on the dataset from the 2016 Kaggle competition.

II. Related work

The problem of using learning algorithms to identify the artists of paintings has been explored by many others, [3], [4] and [5], to name a few. One of the first attempts at recognizing artists from their artwork by using CNNs was presented by [5]. They were able to show that training a standard CNN-classifier on down-sized images of paintings could be an effective method for accurately identifying the artist of a painting. Additionally, by using occlusion sensitivity visualizations, they were able to conclude that the classifier did in fact assign more weight to regions of the images that were characteristic to its artist.

In a CNN, the input image needs to be of a fixed size. For this reason, as well as for the sake of computational efficiency, the image is often down-sampled into a smaller size which means the fine details in the image are lost. An interesting approach by [3] was to decompose the image into different layers of sub-images. With these layers combined they were able to capture different levels of information in the image. In theory, this could be an effective approach since the style of an artist is also captured in the finer details (e.g. brush strokes, choice of paint, etc.).

Most of the previous work approaches the problem of identifying the artists of paintings as a classification problem, attempting for each painting to answer the question "Which artist made this painting?". The Kaggle competition mentioned in the previous section studies a slightly different problem, where for each pair of paintings you are to attempt to answer the question "Were these two paintings made by the same artist?". The winner of the competition, Nejc Ilenic, was however able to show that training a classifier, letting it classify each of the images and comparing the results can be a successful approach [6]. More specifically Ilenic's approach can be described as follows: A CNN was trained as a multi-class classifier to identify the artist of a single painting. In order to compare two paintings, both were input into the trained CNN separately and the prediction was computed as the dot product between the two output vectors. A disadvantage is that this type of approach will be unable to generalize to artists it has not been trained on, which significantly limits its usefulness in practice. This begs the question whether it is possible to design an algorithm which is able to generalize beyond the artists it has seen in training.

Our suggested design for such an algorithm is based in great part on the concepts of neural style transfer and face verification, which are not covered in detail in this paper. A TensorFlow implementation of neural style transfer by Anish Athalye was used as code base for the project [7]. Our solution employs a triplet loss function, frequently used in face verification as described in [8]. Our method of evaluation when comparing two images also has significant similarities to evaluation schemes used in face verification.

III. Dataset

For this project we used the dataset supplied in "Painter by Numbers" on Kaggle [1]. The dataset consists of about 100 000 images of paintings by 2 319 different artists. Most of the images are from WikiArt.org, but the competition creator also included the work of some obscure, less well known artists as well as that of an infamous forger. The number of paintings per artist varies significantly. The most well represented artist has over 400 paintings in the data set while for the least represented artist, there are only a couple of paintings. Figure 2 shows a few examples of images in the dataset.

Fig. 2. A few random images from the dataset.

A. Preprocessing data

The resolution and size of the images varies greatly. To have a uniform resolution through out the dataset the images were rescaled and cropped into a much smaller scale of size 3x224x224. This smaller scale also reduces the computational complexity of training on the images. Lanczos resampling was used to downscale the images. Since the new resolution is square the images were cropped to preserve the original proportions. To help increase the training speed and accuracy the intensities of the images were centered at 0 by subtracting a mean pixel value which was included as a parameter in the pretrained VGG19 weights that were used for our model [9]; see Section IV-B for a detailed description of the model.

B. Subsection datasets and data split

In order to evaluate the performance of our derived architectures on a more manageable amount of data, three smaller subsections of the data were extracted to be used for development. The three datasets created each consisted of:

1) All paintings in the dataset by the 3 artist with the largest number of paintings (951 paintings in total).
2) The paintings of 5 randomly chosen artists from the dataset that each have approximately 100 paintings (559 paintings in total).
3) All paintings in the dataset by the 10 artists with the largest number of paintings (3 823 paintings in total).

Any dataset used was split into a training set and a test set with the ratio 0.95 to 0.05.

C. Generating triplets

As mentioned in the introduction, the network was trained using triplet learning. To be able to use triplet learning the training data needs to be grouped into triplets. This was handled in the preprocessing step. A triplet consist of an anchor A, a positive P (by the same artist as A) and a negative N (by a different artist). Each image from the dataset was used as an anchor at least once. The number of times each image is used as an anchor can be specified as a hyperparameter when preprocessing. For each triplet, the positive image is randomly chosen from one of the images painted by same artist in the dataset and the negative as a painting by a random different artist.

IV. Method

The basic structure of the architecture used when training is outlined in Figure 3. The main steps of the approach can be described as follows: Triplets are formed from the dataset as described in Section III-C. After preprocessing the triplets, the images are input into a set of siamese networks. The siamese networks share weights, and each have the architecture and pretrained weights of VGG19; see Section IV-B for details. Then, just like in neural style transfer, the output volumes of a few selected intermediate layers in the networks are extracted. The extracted volumes are used to compute Gram matrices to get an encoding of the style of each of the three images in the triplet. These encodings are then passed into a triplet loss function, and the cost is computed as the average loss in the training batch. By backpropagating, the shared weights of the siamese networks are trained further. An initial approach for predicting whether two paintings were made by the same artist was as follows: the two paintings were input into the CNN - if the difference between style encodings is was than some threshold, we predicted 0 (not the same artist), otherwise we predicted 1 (the same artist). This method for evaluating was later revised into a new approach, described in detail in Section IV-C

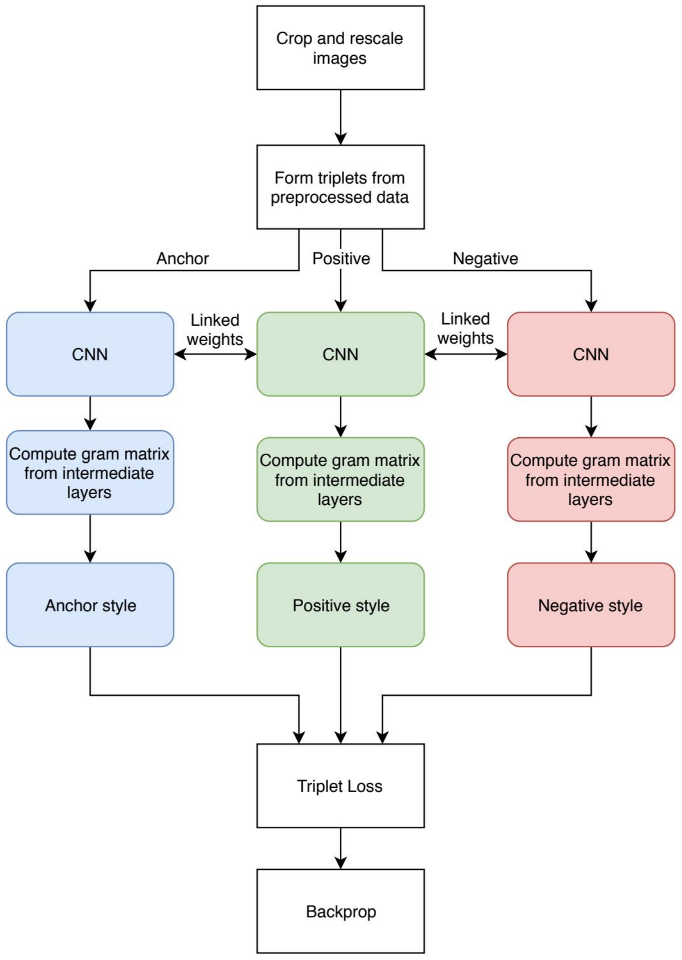
Below we provide a more detailed description of each of the steps in our method and the architectures used.

A. Baseline model

The winning architecture in "Painter by Numbers" on Kaggle, was used as a baseline model [6]. The architecture supplied by Ilenic on GitHub was trained and tested on each of the datasets described in Section III-B. Each model was trained for 200 epochs with default settings. Performance on the test set was evaluated with the AUC metric. Performance on the training set was supplied as F1-score by default; see Section V for baseline performance.

B. CNN architecture and loss function

The main architecture used for this project is a siamese build of the VGG19 convolutional neural network, as implemented and described by [8]. The pretrained weights we used are available from MatConvNet (a Matlab toolbox which has implementations of several well-known networks) [9]. For the purpose of this project, only the layers up to and including the layer named "relu5_1" in MatConvNet are included in our siamese build. This is due to the fact that this is the last layer we are interested in extracting style encodings from, and thus the remaining layers are not needed.



Crop and rescale images
Form triplets from preprocessed data
Anchor
Positive
Negative
CNN
Linked weights
CNN
Linked weights
CNN
Compute gram matrix from intermediate layers
Compute gram matrix from intermediate layers
Compute gram matrix from intermediate layers
Anchor style
Positive style
Negative style
Triplet Loss
Backprop

Fig. 3. Overview of training.

The choice of which layers to extract style encodings from was based on a popular implementation of neural style transfer in TensorFlow by Anish Atalye [10]. Our assumption is that layers that work well for style transfer should also provide a representation of the image style that could be of value when attempting to identify the artist of the image. Thus, we have used the same layers as Atalye (or some subsection of them) to extract our style encodings.

For any layer l in which we want to extract style, the style encoding is computed as the Gram matrix $G^{[l]}$ of the activation $A^{[l]}$

$$G^{[l]} = (A^{[l]})^T A^{[l]}. \tag{1}$$

The triplet loss function of style encodings for the anchor A, positive P and negative N is then defined as

$$\mathscr{L} = \sum_l ||G_A^{[l]} - G_P^{[l]}||^2 - ||G_A^{[l]} - G_N^{[l]}||^2 \tag{2}$$

where $G_A^{[l]}$, $G_P^{[l]}$ and $G_N^{[l]}$ are the Gram matrices in layer l of A, P and N respectively. We want to minimize the norm distance between encodings of paintings by the same artist (A and P) and maximize the norm distance between encodings of paintings by different artists (A and N). In accordance with the typical approach to triplet learning, a threshold α is added, and we take the maximum of the distance plus the threshold and 0 to be the loss in each layer

$$\mathscr{L} = \sum_l max(||G_A^{[l]} - G_P^{[l]}||^2 - ||G_A^{[l]} - G_N^{[l]}||^2 + \alpha, 0). \quad (3)$$

This is because we do not necessarily want to find a style encoding that results in the largest possible difference between the A/P distance and the A/N distance. Rather we want to find an encoding which results in a sufficiently large difference between these distances, where the threshold α determines what distance is deemed sufficient.

If we have m triplets of images in the batch, the cost is defined as the average of the loss for all triplets in the batch

$$J = \frac{1}{m} \sum_{i=1}^{m} \mathscr{L}(A^{(i)}, P^{(i)}, N^{(i)}). \quad (4)$$

Adam optimizer with default parameters was used for training on the triplet cost. A learning rate of 1×10^{-4} and a mini-batch size of 32 proved most effective; see Section IV-D for details on hyperparameter tuning.

C. Evaluating

The planned approach for evaluating on a pair of images was initially to compute the l2-distance between the style encodings of the paintings, and compare it to the average distance between anchors and positives in the training set, and anchors and negatives in the training set respectively. If the distance was closer to the average A/P-distance we would predict 1 (same artist) for the image pair. Conversely, if the distance was closer to the average A/N-distance we would predict 0 (not the same artist). As this approach quickly proved to be ineffective on some of the smaller datasets used, another method for evaluating was synthesized.

The updated approach can be summarized as follows: Upon completing training, an average style encoding for each artist in the training set is computed and saved with the network's weights. When evaluating on a pair of images, the distance to each of the average style encodings is computed for both images in the pair. Each of the two images is classified as the artist which had the closest average style encoding. If the two images were classified as the same artist. we predict 1 (same artist), otherwise we predict 0 (not the same artist).

Note that this revised approach presents limitations in generalizing to artists that were not seen during training. However, if the pair of images to be evaluated consists of two paintings by the same artist, it may be likely that both images are still classified to the same (incorrect) artist. Conversely, if the pair of images are by two different artists, it may be likely that they would be classified as two different (incorrect) artists. In this sense, the process of computing the average style for each known artist might serve as a discretization of the space of style encodings. Evaluating thus becomes a process of computing which style "bucket" each painting belongs to, and checking whether they end up in the same bucket.

D. Hyperparameter tuning

The hyperparameters that were assumed to be most vital in this architecture were the learning rate, mini-batch size, the triplet threshold α, the number of layers to extract style from and which of the pretrained layers to freeze during training. Out of this set of vital hyperparameters, the most critical hyperparameters proved to be the learning rate and triplet threshold α.

For the learning rate, values between 1×10^{-3} and 1×10^{-9} were tested and a value around 1×10^{-4} gave best performance. For the triplet threshold α, values between 0 and 1×10^{10} were tested and a value around 1×10^{9} gave good results.

We also experimented with a number of different permutations of which layers the style was extracted from. The choice of style layers, among the five that were used by [10], did not seem to result in any significant difference in performance. The layers we ended up extracting style from was: "relu4_1" and "relu5_1"; see [9] for a detailed description of the layers of VGG19.

The mini-batch sizes 1, 8 and 32 were tested. The combination of a mini-batch size of 32 and a learning rate of 1×10^{-4} resulted in the quickest convergence. Models trained on the small sub-datasets described in Section III-B appeared to converge after $5 - 15$ epochs, depending on the complexity of the dataset.

The best performance was achieved when there were no frozen layers.

E. Visualizations using PCA and t-SNE

PCA was used to visualize the style encodings learned by the model. For each subset of the dataset (3 artists, 5 artists, 10 artists) we used the scikitlearn implementation of PCA (with n_components=2) on the style encodings of the training set and the test set, and plotted the results in one figure respectively. In cases where the PCA-figure was deemed too messy or irregular to interpret, t-SNE, another popular dimensionality reduction technique by Laurens van der Maaten, was applied to the data [11]. In the cases where t-SNE was used, we first applied PCA with n_components=50, and then t-SNE with n_components=2 and plotted the result in a figure. In an attempt to study the generalizability of our approach, the PCA visualization was also applied on the style encodings of paintings from a few random artists not seen during training, and the results were plotted in a figure.

V. Results

The performance results when training and evaluating on the different datasets, for the baseline as well as our model, can be found in Table I. Visualizations of the results through PCA and t-SNE are presented in Figure 4. The visualizations using

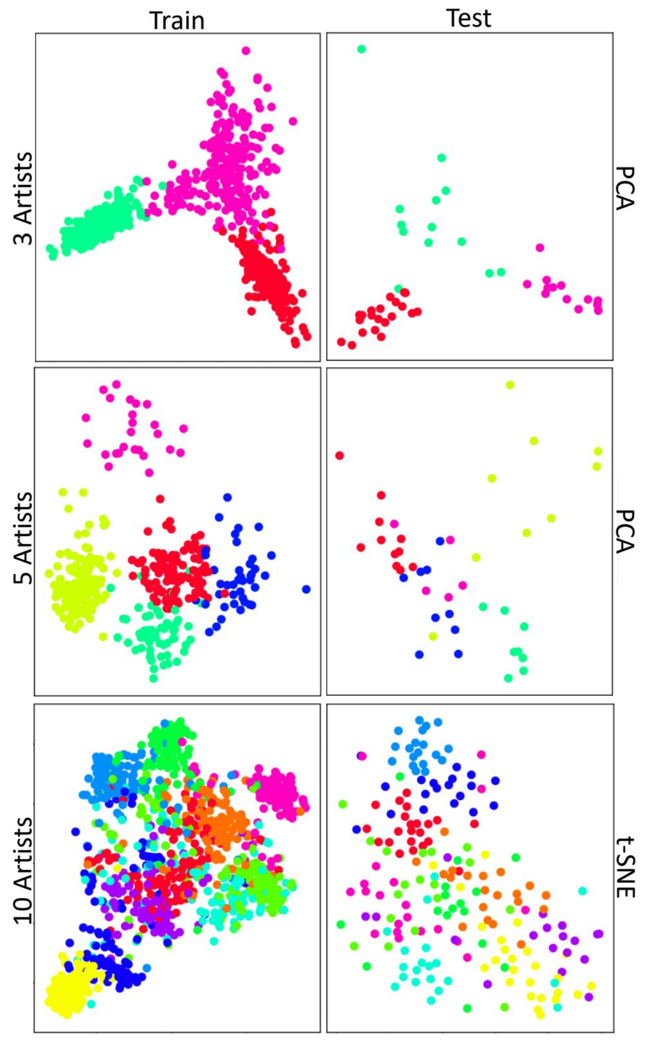


Fig. 4. PCA and t-SNE for the sub-datasets of 3 artists, 5 artists and 10 artists.

	Baseline		Our model	
# artists	train (F1)	test (AUC)	train (AUC)	test (AUC)
3	0.984	0.967	0.980	0.963
5	0.972	0.812	0.974	0.833
10	0.857	0.837	0.700	0.633

TABLE I
PERFORMANCE ON THE DIFFERENT DATASETS.

PCA and t-SNE were also applied to a few random artists not seen during training. Since the results in this experiment were very poor, and lacked any distinct pattern the figure was not deemed necessary to present here.

VI. DISCUSSION

In Table I we can see that for a limited number of artists our approach performs just as well as the baseline. However, as the number of artists increases the performance seems to worsen. A reason for this could be the way the artist of a painting is determined given its style encoding. As the number of artists increase, so does the number of average style encodings we need to compare to the style encoding of the painting. Therefore the distance between each average style encoding is more likely to be smaller. Finding a more effective way of determining which cluster a particular painting belongs to might be a way to improve performance.

As the number of artists increases, and the space of style encodings becomes more cramped, we believe it may also become more vital to limit the triplets used for training to ones that are non-trivial to the model. During training, the cost seemed to quickly converge to a lower value from which it was very difficult for the model to improve any further. This could be due to a lack of difficult triplets (i.e. triplets where, before any training, the style encodings of A and N are similar, and/or style encodings of A and P are dissimilar).

Since the PCA visualizations have very distinct and tight clusters for the different artists when the number of artists is small, it is reasonable to think that the network has in fact learned an underlying style encoding for the images to some extent. However, as mentioned in Section IV-E we tried using paintings from artists unknown to the model as input to the visualization in order to see whether the network could find style patterns for each artist and with that distinguish their paintings from one another. The outcome of this experiment, as stated in Section V, did not show signs of such generalization. However these artists were chosen at random and therefore their styles may have been similar to some extent. An interesting experiment would be to instead hand-pick a set of artists with very distinct and different styles and see if there is some clearer clustering. On the other and, it might also be unreasonable to think that the network could learn a generalizable style encoding from such a limited number of artists. Finding a sustainable way of training on an increased number of artists is thus perhaps the most critical aspect moving forward.

VII. CONCLUSION AND FUTURE WORK

We were able to show that training a set of siamese CNNs on the triplet loss of style encodings can be an effective way of telling the work of different artists apart when the number of artists is limited. As the number of artists increased, the performance of our model worsened compared to the baseline model (a traditional artist classifier).

The current approach to evaluating on a pair of images poses significant limitations, as described in Section VI. Therefore, there is potential for improvement if a new method for evaluating could be synthesized. Since the triplets used for training were generated randomly, it is also likely that there is a lot of redundancy in the training set in the form of trivial triplets. A future improvement would therefore be to find a algorithmic method for generating triplets that are predominantly non-trivial.

VIII. ACKNOWLEDGEMENT

We would like to thank Daniel Kunin for helping us develop this idea and for his valuable guidance throughout the project.

IX. Contributions

Both project members shared the workload of the project in equal parts. Albin worked a bit more on preprocessing and code structuring, while Josef worked a bit more on the documentation and visualizations. The central aspects of the project (i.e. building, training and testing our model and the baseline) were shared in equal parts.

X. Code

All code used for this project can be found in our GitHub repository "artist-differentiator", available at github.com/josefmal/artist-differentiator. Then main programming framework we used was TensorFlow [12]. Other libraries used included scikit-learn, numpy, pillow, scipy, h5py, tqdm.

References

[1] Kiri Nichol, "Painter by Numbers — Kaggle," 2016. [Online]. Available: https://www.kaggle.com/c/painter-by-numbers

[2] G. Bonner and J. V. Noble, "Forgery — art — Britannica.com," 2016. [Online]. Available: https://www.britannica.com/art/forgery-art

[3] K. A. Jangtjik, M.-C. Yeh, and K.-L. Hua, "Artist-based Classification via Deep Learning with Multi-scale Weighted Pooling," in *Proceedings of the 2016 ACM on Multimedia Conference - MM '16*. New York, New York, USA: ACM Press, 2016, pp. 635–639. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2964284.2967299

[4] E. Levy, O. E. David, and N. S. Netanyahu, "Genetic algorithms and deep learning for automatic painter classification," in *Proceedings of the 2014 conference on Genetic and evolutionary computation - GECCO '14*. New York, New York, USA: ACM Press, 2014, pp. 1143–1150. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2576768.2598287

[5] N. van Noord, E. Hendriks, and E. Postma, "Toward Discovery of the Artist's Style: Learning to recognize artists by their artworks," *IEEE Signal Processing Magazine*, vol. 32, no. 4, pp. 46–54, 7 2015. [Online]. Available: http://ieeexplore.ieee.org/document/7123719/

[6] Nejc Ilenic, "GitHub - inejc/painters: Winning solution for the Painter by Numbers competition on Kaggle," 2018. [Online]. Available: https://github.com/inejc/painters

[7] A. Athalye, "GitHub - neural-style," 2015. [Online]. Available: https://github.com/anishathalye/neural-style

[8] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering," 3 2015. [Online]. Available: http://arxiv.org/abs/1503.03832http://dx.doi.org/10.1109/CVPR.2015.7298682

[9] VLFeat.org, "Pretrained CNNs - MatConvNet," 2017. [Online]. Available: http://www.vlfeat.org/matconvnet/pretrained/

[10] A. Athalye, "An AI That Can Mimic Any Artist," 2015. [Online]. Available: https://www.anishathalye.com/2015/12/19/an-ai-that-can-mimic-any-artist/

[11] Laurens van der Maaten, "t-SNE." [Online]. Available: https://lvdmaaten.github.io/tsne/

[12] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citrov, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, R. Jozefowicz, Y. Jia, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, M. Schuster, R. Monga, S. Moore, D. Murray, C. Olah, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "{TensorFlow}: Large-Scale Machine Learning on Heterogeneous Systems," 2015. [Online]. Available: https://www.tensorflow.org/