Cross-Depiction Transfer Learning for Art Classification

Debnil Sur
Stanford University
353 Serra Mall, Stanford, CA 94305
debnil@stanford.edu

Ellen Blaine
Stanford University
353 Serra Mall, Stanford, CA 94305
eblaine@stanford.edu

Abstract

Automated art analysis is useful as a tool for both research and educational app development. We perform artist classification on 13,574 works from 25 artists in the Rijksmuseum collection using various applications of transfer learning. The most successful model (based on ResNet18) achieved 82.5% accuracy on a held-out test set, which is a 7% improvement relative to the previous best single model approach's performance on this dataset [23]. We illustrate by example that the errors this pipeline makes may help identify new relationships between artists or schools of art. Finally, we also propose a new application of style transfer to classification. Although our experiments with this pipeline did not perform as well as those that used ResNet18, we argue that the insights this method can provide into an artists' work merit further investigation of this technique. All code has been published to a public Git repository.[1]

1. Introduction

Many art museums have digitized their collections, which allows developers to use computer vision to create more engaging experiences for museum-goers. The most obvious example is a mobile application that uses a phone camera to identify an object on display. We believe that this data can also be used to reveal trends in artwork of great relevance to art historians. In the process, computer scientists can better understand intricacies and extensions of modern paradigms, especially in computer vision. One such question at the intersection of art history and computer vision is the identification of a painting's artist. The Rijksmuseum in Amsterdam released a dataset of its collection in 2014, challenging researchers to automatically classify works in the dataset by artist, type, material, and year [15]. In this paper, we describe approaches for tackling the artist classification challenge using transfer learning, including a pre-trained ResNet architecture and a new application of style transfer to artwork classification [9, 5]. Given a photographic reproduction from one of the 25 most prolific artists in the dataset, we automatically predict the artist who created the work.

[1] All code can be found at https://github.com/eblaine/cs231a-project

2. Related Work

A mixture of conventional and deep learning approaches has been used to tackle painting classification. Earlier work uses handcrafted features to identify a painting's artist. Specific mathematical methods, like sparse coding and fractal geometry, have been applied to certain artists' works to differentiate them from those of imitators [10, 21]. Collaboration with art historians improved feature engineering, which led to promising results in attributing artworks by van Gogh and his contemporaries [11]. For example, [15] uses normalized Fisher vectors with 1-vs-Rest linear SVM classifiers to correctly classify 74.8% of artists in a held-out test set. Feature learning learns features directly from the data; while more data-intensive, biologically-inspired approaches in this vein, like convolutional neural networks, have outperformed all existing learning algorithms on many challenging image classification tasks [14]. The best reported use of a single CNN for artist classification achieved 78.3% accuracy using an AlexNet-inspired architecture [24]. The best reported ensemble model ensembles several multi-scale models in an effort to combine scale-invariant and scale-variant representations. This achieves 82.1% accuracy and is the best reported result on the artist classification task.

The success of the last class of methods especially depends on the use of transfer learning. This is the process of training a model to perform some task in a given domain, starting from a model trained for another domain and/or task. This general approach has yielded state-of-the-art results in a wide diversity of tasks, such as Web document classification [4], indoor WiFi localization [16], and sentiment classification [1]. In computer vision, this paradigm has been used in situations that require a high-capacity model but have only a small quantity of annotated detection data. Specifically, convolutional architectures trained for image classification see millions of images, and their features have been remarkably generalizable to other tasks. Most notably, [6] reduced error on object detection on the PASCAL-VOC dataset by almost 30% through using a pre-trained ImageNet. These nets can also be used as the initialization of architectures for wholly different transfer tasks. [25] use a pre-trained ImageNet as an initial architecture to transfer night-time satellite images for mapping poverty. Given this, pre-trained ImageNet architectures seem like sufficient and suitable candidates for painting classification.

In particular, this particular experiment serves as a testbed for two fascinating challenges in transfer learning: scene composition and cross-depiction transfer. First, ImageNet is trained on hundreds of thousands of pictures of twenty different objects. The features extracted in a pre-trained ImageNet architecture are thus quite expressive of the style and content of these chosen objects. On the other hand, paintings have scenes. It will be interesting to see how features of individual objects generalize to large, complex scenes composed of many objects, many of which are not in the original set of objects. Second, ImageNet is trained on photographic images of these objects. The art we are looking at spans the 15^{th} to 20^{th} centuries, and as a result, they range in material from oil to charcoal to porcelain. Consequently, the styles conveyed in the pre-trained features may be manifested very differently in convolution than those of a neural net trained on our dataset. Studying the effects of transfer learning across depiction materials serves as another important contribution of our project.

Although transfer learning has been hardly used for painting classification, the techniques underlying it have been extensively used in recent relevant work on style transfer. [5] use CNNs to separate and recombine the content and style of arbitrary images. These findings have been effectively used in generative approaches to changing images' visual content [12, 22]. Other work has used explicit, general representations of the convolutional layers to apply a variety of styles, rather than just that of a single painting [3] However, to the best of our knowledge, the core principles of style transfer have not been used for a logical classification task. Thus, we seek to combine the principles behind style transfer and transfer learning to more effectively identify paintings' artists and approach genealogical questions about art.

3. Dataset and Features

3.1. The Rijksmuseum Dataset

The Rijksmuseum Challenge dataset contains 222,945 photographic reproductions of works from 20,116 artists. Most of the artists in the dataset produced only a few of the items in the collection, so a classifier would have difficulty

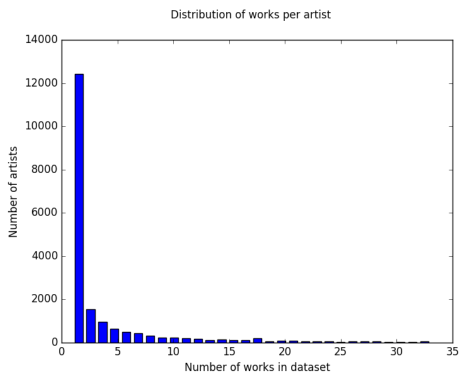


Figure 1. Histogram of how many artists produced various numbers of works. Notice that most artists produced very few (< 5) works.

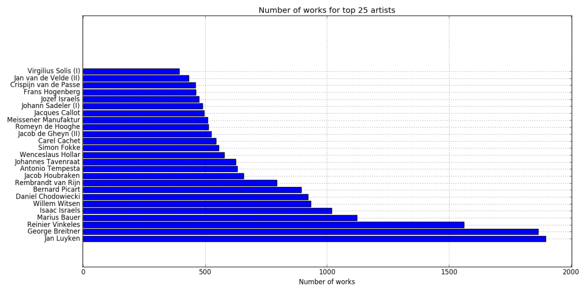


Figure 2. Counts of pieces for each of the top 25 artists.

learning meaningful representations of those artists' works. Figure 1 displays the distribution of numbers of works produced by artists in the dataset.

Past attempts have trimmed this dataset based on a threshold of works produced per artist (e.g. van Noord et al. use 64 [23]). However, we chose to use pieces from the 25 most prolific artists, for 13,574 total data points. We made this choice because the dataset has more than doubled and the number of artists has more than tripled since that previous attempt.

Within this set of 25 artists, there is still a fair amount of variability in number of works produced (see Figure 2 for a visual). To divide these remaining works into train, validation, and test subsets, we employed the following method for class balancing from van Noord et al. [23]: We assigned 70% of each artist's works to the train set, 10% to the validation set, and 20% to the test set.

To work nicely with pretrained networks, all images were resized and center-cropped to 224 x 224 pixels.

3.2. Features

For the classical baseline and transfer learning with classical techniques pipeline, we extract histogram of oriented gradients (HOG) features. This feature extraction algorithm bins local gradient orientations in an image, which might implicitly capture some sense of an artist's style for classification. We made this choice over other methods that might also capture style because (1) HOG features were a popular choice of descriptor for object detection before the networks we used for transfer learning were introduced and (2) we found experimentally that some other common features (SIFT, SURF) cannot be extracted from paintings reliably.

4. Methods

The best Rijksmuseum Challenge submission to date uses transfer learning, so we attempted to replicate and extend those results [23]. Additionally, we implemented a baseline model for comparison using classical computer vision techniques and a new architecture that uses style transfer for classification.

4.1. Logistic Regression

The classical model we implemented uses a logistic regression-based classifier, which uses cross-entropy loss [17]:

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

where f_j corresponds to the predicted probability that data point i is in class j, and y_i is the true class for data point i.

4.2. Convolutional Neural Networks

The next suite of methods utilize convolutional neural networks, which include convolutional operations over the input and are designed specifically for vision tasks. These convolutional filters encode translation invariance, which helps discover useful features in images [2]. A CNN is a general function approximator consisting of a set of convolutional and fully connected layers, such that the output of one layer is the input to the next. The first layers of the network typically learn lower-level features like edges and corners, while further layers learn high-level features like textures and objects. A convolutional layer maps a tensor $x \in \mathbb{R}^{h\times w\times d}$ to $g_i \in \mathbb{R}^{\hat{h}\times\hat{w}\times\hat{d}}$, such that $g_i = p_i(f_i(W_i * x + b_i))$. For the i-th convolutional layer, $W_i \in \mathbb{R}^{l\times l\times\hat{d}}$ is a tensor of $\hat{d}$ convolutional filter weights of size $l\times l$, $(*)$ is the 2-dimensional convolution operator over the last two dimensions of the inputs, b_i is a linear bias, f_i is an elementwise nonlinearity, and p_i is a pooling function [25]. The output dimensions $\hat{h}$ and $\hat{w}$ depend on the stride and padding of the layer, which control how convolutional filters slide over the input.

In addition to convolutional layers, CNN models also have fully connected layers in the final layers of the network. These map an unrolled version of the input $\hat{x} \in \mathbb{R}^{hwd}$, a one-dimensional vector of the elements of a tensor $x \in \mathbb{R}^{h\times w\times d}$ to an output $g_i \in \mathbb{R}^k$ such that $g_i = f + i(W_i\hat{x} + b_i)$. Here, $W_i \in \mathbb{R}^{k\times hwd}$ is a weight matrix, b_i is a bias term, and f_i is a (typically rectified linear unit) nonlinearity function. These layers encode the input examples as feature vectors, to then be used as inputs to a final classifier. These summarize the input into a vector for classification. The model is trained end-to-end via minibatch gradient descent and backpropagation.

4.3. Transfer Learning

In transfer learning, a neural architecture trained for one task is retrained for another. Typically, a new classifier layer is trained first, to take full advantage of the successful features of the prior model. The convolutional features are then re-trained with a lower learning rate to adjust them to the task at hand. Given the aforementioned results of transfer learning in other computer vision tasks, we expect great success.

We formalize transfer learning as in [16]. A domain $D = \{X, P(X)\}$ consists of a feature space X and marginal probability distribution $P(X)$. Given a domain, a task $T = \{Y, f\}$ consists of a label space Y and a predictive function f which models $P(y|x)$ for $y \in Y$ and $x \in X$. We define a source domain and learning task D_S and T_S and target domain and learning task D_T and T_T. Transfer learning aims to better learn the target learning task's predictive function f_T using the knowledge from D_S and T_S, where $D_S \neq D_T$ and $T_S \neq T_T$.

Here, the source domain is ImageNet, an object classification image dataset of over 14 million images with 1000 class labels that have led to significant breakthroughs in many vision tasks [18]. We utilize a deep residual network, or ResNet, a family of deep architectures showing compelling accuracy and nice convergence [9]. The relative recency of this work has made it underexplored in transfer learning approaches, as other slightly older, more lightweight architectures tend to be used more frequently. Thus, in addition to our specific task, this also serves as a case study in the utility of ResNet architecture in transfer learning.

The target domain is the Rijksmuseum Challenge dataset, using the preprocessing described before. The target task is artist classification for each painting. We will follow the two cycles of training (classifier, then convolutional layers). In these training cycles, we will use a cross-entropy loss of the same form as logistic regression.

4.4. Transfer Learning with Classical Features

This pipeline is similar to the previous, but its fully connected layer has a small change. We concatenate the flattened output from the network's convolutional layers with a point in HOG space corresponding to the original image (see Dataset and Features section). The resulting vector is then used as input to the linear fully connected layer. This step significantly increases the size of the linear layer's feature space.

The baseline (HOG and logistic regression) method demonstrated that HOG-based models are prone to overfitting for this task, so we also introduced regularization in this pipeline. We apply dropout to the HOG feature vector, with the probability that a given parameter is dropped set to $p = 0.5$ [20]. The classification vector produced by the linear layer is a probability distribution over the number of artists. For each minibatch, the cross-entropy loss is computed using the correct class labels and the output vector of the linear classifier layer.

4.5. Style Transfer

Recent advances in style transfer have successfully separated a painting's style from its content. This has then been used to apply one painting's style to another image. We seek to take these principles from generative modeling to classification. Consider a convolutional architecture with L layers for feature extraction. Of these, let there be L_S style layers and L_C content layers. We can write the dimensions of a single feature map corresponding to a style layer as $N \times C_j \times H_j \times W_j$, where $j \in \{1, 2, ..., L_S\}$ is the index of the style layer, N is batch size, and C_j, H_j, W_j are the layer's number of channels, height, and width respectively. We can flatten this feature map into a $N \times C_j H_j W_j$ matrix to more compactly represent each individual example's feature map.

We wish to use a single style or content layer, or some combination of layers of the same type, to generate predictions. To do this, we can create a set of embeddings for each batch of examples and then run it through a linear classifier. For example, for the single style layer described above, we can create a $N \times C_j H_j W_j$ embeddings matrix, where each row represents the flattened style map for a single painting. This concept can be easily generalized to multiple layers by concatenating them together to form a larger embedding. For example, flattening and concatenating all style layers gives us a $N \times \sum_{j=1}^{L_S} C_j H_j W_j$ embeddings matrix. We can then run a series of fully-connected layers (with regularization methods like dropout and ReLU) to generate a classification vector with the desired output dimension. The same principle can be applied for content.

It is of note that [5] approach the style layer considerably differently than we do. They attempt to minimize the difference in style between one image and another, transformed in style in the pattern of the first. They represent style through computing the spatial correlation of values within an image, which is mathematically done via the Gram matrix of a single layer. The loss is then computed via the L2 norm of the sum of the difference between the various layers' Gram matrices. The Gram matrix is used as a proxy for covariance and is the best way to compactly represent this principle when computing a loss. Since the embeddings will be piped through a fully-connected layer before classification, the training process will backpropagate through this layer and, through retraining weights, better capture relevant relationships between regions of the image. Moreover, using the Gram matrix in designing an embedding would square the number of terms in an already large embedding matrix, which is simply untenable from a memory perspective. Consequently, for the purposes of linear classification, using a concatenation of flattened style layers seemed sufficient.

5. Experiment/Results/Discussion

5.1. Experimental Setting

As outlined in our methods, we applied a variety of classical and modern approaches in image classification. In all of these tasks, we processed images in batches of 50. We experimented with several batch sizes and found that this was the largest size that had reasonable speed without excessive memory usage. We also utilized the Adam optimizer [13]. This has been shown to be faster and reach better minima than most other built-in PyTorch optimizers. It was also used in most recent transfer learning papers for vision, though no authors provided a justification. Thus, we decided to use it as well.

5.2. Logistic Regression

In this experiment, we ran a logistic regression classifier trained with stochastic gradient descent using HOG features as input. This model overfits quickly, despite regularization via L2 penalization [7]. With a learning rate of $\alpha = 10^{-4}$, the model achieved 71.4% and 31.8% accuracy on the train and test sets, respectively, after 6 epochs. Decreasing the learning rate did not appreciably improve performance. This significant gap between train and test accuracy indicates that HOG features alone are prone to overfitting. One explanation for this is that a HOG feature vector encodes both an artist's style and a painting's composition, so the model does not generalize well to paintings by the same artist with very different compositions (e.g. consider a landscape and a portrait painted by the same individual).

5.3. Transfer Learning

This model uses the PyTorch implementation of ResNet18, pre-trained on ImageNet. A new classifier layer is used with the number of classes equal to the number of artists. Dropout with probability 0.5 is applied before classification; this substantially reduced overfitting. We first train this classifier layer using an Adam optimizer with a learning rate of 1×10^{-3} for 5 epochs. We initially utilized

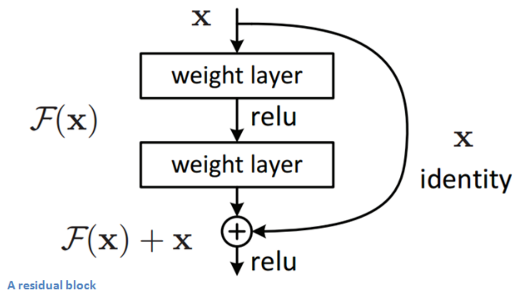


Figure 3. A sample residual block [9].

this learning rate then randomly sampled others; however, it yielded better results than any tried during the search. We then retrain the full architecture using a learning rate of $4.5e - 4$, which was found to yield the best results through random sampling. This had a recorded test accuracy of 82.5%, over 7% better than the prior best single model approach. The train accuracy of 99.6% and broadly convergent validation loss seemed to indicate that we would not receive substantially better results with more training, so we moved on to other experiments. Running this architecture multiple times and combining the resultant models through ensembling would likely substantially boost the results.

Before settling on the ResNet18, we also attempted the PyTorch implementations of AlexNet, VGG-16, and ResNet50. All were pre-trained for the ImageNet challenge and run for 5 epochs of classifier training and 5 of convolutional training. The difference between these results has implications for the use of ResNet in future transfer learning-based experiments. Our experiments directly compare the most common CNN-based architectures and help test their performance across depiction materials and in scenes.

The AlexNet architecture performed substantially worse, even with significant hyperparameter search. The highest recorded test accuracy was 32.3%. Given the success of pre-trained AlexNets in many other transfer learning experiments, this may be due to poor hyperparameter search ranges. It is more likely, though, that the ResNet18 uses better, more generalizable convolutional features, as its error rate on the ImageNet is half that of AlexNet's. This improvement can be directly attributed to the use of residual blocks. The input x typically goes through conv-relu-conv layers to compute some $F(x)$. Let the value propagated to the next layer be $H(x)$. In a traditional CNN, $H(x) = F(x)$. This traditional representation keeps no information about the original x. On the other hand, in a residual block architecture, $H(x) = F(x) + x$. Each layer thus computes a change to the original input to get an altered representation. In terms of training, it is easier to optimize the residual mapping than the original mapping, since the addition operations distribute gradients to all layers. This set-up makes the ResNet particularly better in our transfer setting, when the target domain images significantly differ in both content and style from those in the style domain. Even after some retraining, the representation $F(x)$ produced by the convolutional weights is likely to significantly differ from the optimal convolution of the source image. As a result, passing the raw input through the layers, as in residual modules, will pass more representative features to the final image layers. This helps explain the significantly better performance of the pre-trained ResNet in a cross-depiction transfer learning setting.

Second, the VGG-16 architecture seemed promising, given that other literature has demonstrated its success in transfer learning [8]. However, a single epoch of VGG was five times slower than one of ResNet, so fully training the model was untenable given the timeline of the project. This stems from the much higher memory and computation requirements of VGG [19]. Given our own limitations on GPU use, we felt that it was a better use of our time to focus on hyperparameter optimization and other models than spending hours tuning VGG.

Third and finally, ResNet50 was much more difficult to re-train than the ResNet18. It had a peak train/test accuracy of 81.3% and 67.4%. Training a linear classifier used a learning rate of 1×10^{-3}, and training the full architecture used a learning rate of 3×10^{-3}. The higher learning rate for the convolutional layers reflects the difficulty of re-training deeper networks in a transfer setting. While deeper networks yield much better results on specific tasks, this seems to decrease the generalizability of their features. Typically, higher layers capture more abstract features of the input, while lower layers are more tailored to the specific objects being captured. Here, despite the success of residual modules in propagating the raw input, the extracted features will be less generalizable as more layers are used. Our direct comparison of the ResNet18 and ResNet50 experimentally confirms this through a case study.

The ResNet18 confusion matrix is displayed in Figure 4. The bright diagonal indicates performance was generally good. Bright spots outside the diagonal indicate incorrect predictions. Some errors are a function of similarity between works of the same type created at around the same time. For example, Virgilius Solis I was frequently misclassified as Crispijn van de Passe, which makes sense because both men were engravers who depicted scenes from mythology and fantasy in the late Sixteenth Century. We also see some confusion between artists belonging to related schools, such as Antonio Tempesta the Baroque engraver begin mistaken for Johann Sadeler (I) the Mannerist (Mannerism is considered a precursor to the Baroque style).

The two most-confused artists are more unexpected, however. Daniel Chodowiecki, a Polish-then-German printmaker whose work is best classified as Classicist, is fre-

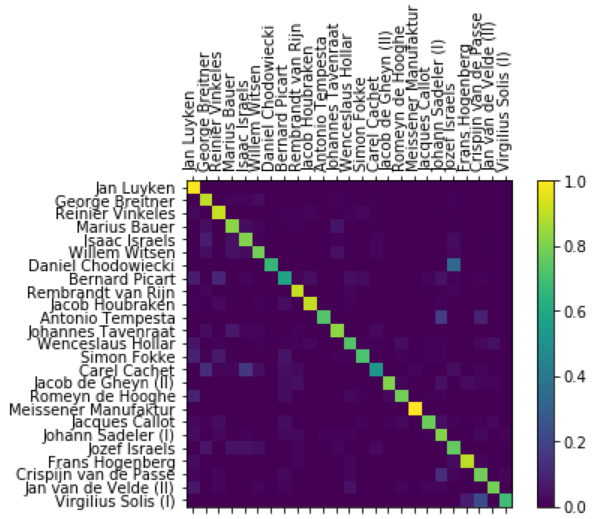


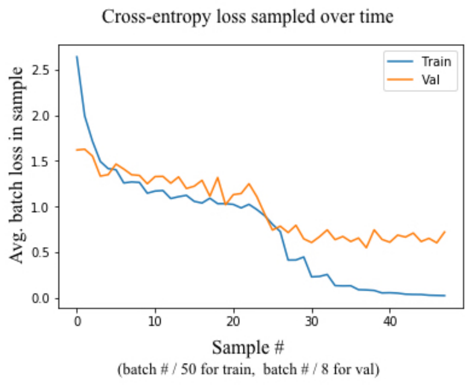
Figure 4. Confusion matrix for ResNet18 architecture. Vertical axis is true labels, horizontal is predicted.

quently predicted to be that of Jozef Israels, a social realist from over a century later. Social Realism actively rejected Romanticism, which borrowed from Classicism, so one would not expect these two artists to produce similar works. Moreover, these artists worked with different primary media, depicted individuals of different social class, and displayed different themes. However, closer inspection of Chodowiecki's paintings reveals that both artists paint figures with intense *chiaroscuro* and use similar scene compositions. This indicates that our model's errors could encourage art historians to consider relationships between artists that are not usually discussed together.

5.4. Transfer Learning with Classical Features

This model employs a pretrained ResNet18 model and concatenates a HOG feature vector to its flattened convolutional layer output before its fully-connected layer. Figure 5 shows the model's training and validation loss over time compared to that of the previous model, which performed better. This pipeline trained its fully-connected layer for 8 epochs with a learning rate of 10^{-3}, and then all layers were trained with a smaller learning rate of $4x10^{-5}$. We kept the learning rates the same as the previous experiment for two reasons: (1) we wanted an apples-to-apples comparison of the performance for each model and (2) training this model is slow because HOG features cannot be batch-extracted easily, and there is a large enough performance deficit between this model and the previous that hyperparameter tuning did not seem worth the extra time.

After all 16 epochs, the model classified training data with 92.5% accuracy and test data with 59.5% accuracy. Several factors might contribute to this performance reduction. In this setting, the linear layer must reduce the vector space dimensionality from over one million to 25 in a single step. Adding intermediate layers may have helped performance, but was not possible due to memory constraints. Additionally, HOG features are not well suited to scene description, and may add a significant amount of noise.



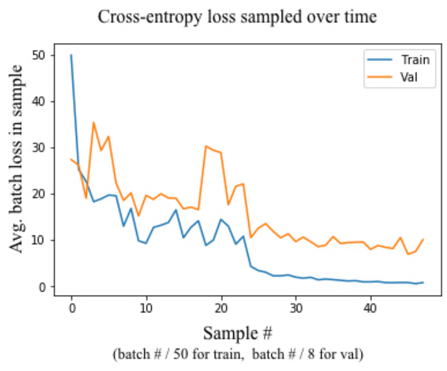
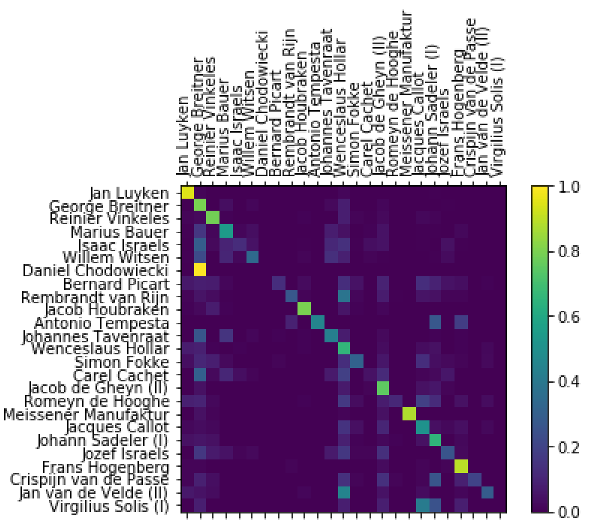


Figure 5. Cross-entropy loss (training and validation) over several epochs. *Top:* ResNet18 architecture. *Bottom:* ResNet18 with HOG features appended to fully-connected layer input.

5.5. Style Transfer

This model uses a pre-trained SqueezeNet to extract style and content embeddings for each batch of paintings. While prior experiments used the ResNet, we used a SqueezeNet for this task due to its faster training and familiarity with the relevant layers corresponding to style and content. Moreover, creating a fully-connected layer from a large, flattened feature map to a much smaller classification vector can require a vast number of parameters. We hoped that due to SqueezeNet's significantly smaller number of parameters, this process would take less memory. Unfortunately, due to RAM constraints, utilizing all content and style layers was not possible. Instead, we only used one content layer and one style layer in this classification. This was not an issue with content, as only one convolutional layer has been de-



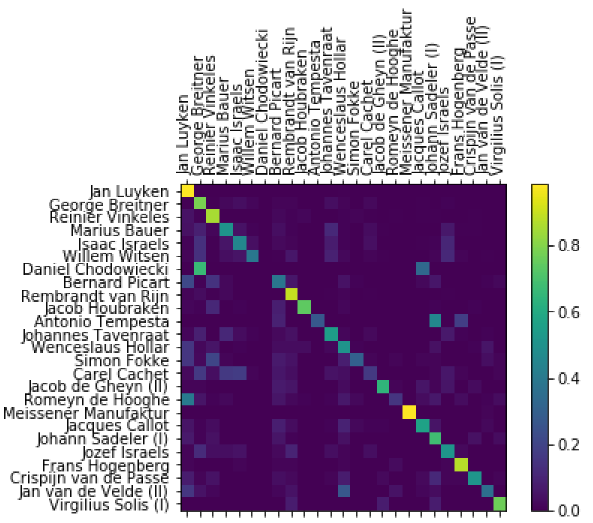


Figure 6. Confusion matrices for classifiers based on content and style embeddings. Vertical axes are true labels, horizontal are predicted. *Top:* Content layer embeddings. *Bottom:* Style layer embbedings.

termined to hold relevant features. However, four layers' activations are correlated with style. We chose one of the middle layers, as we thought that higher layers would be too abstract but lower ones may be too specific to the original data. We then created embeddings following the procedure outlined before.

In training this model, we employed a cyclical training technique. We first froze the gradients for the SqueezeNet and passed a batch of images through. For each batch, we flattened the relevant content activations. We then trained a linear layer for artist classification on top of these flattened activations. We conducted five training epochs for this linear layer, using a learning rate of 1×10^{-3} and cross-entropy loss. The true classes of each painting were used for ground truth. Then, we retrained the SqueezeNet using a new classifier layer with output size equal to the number of artists. We used a learning rate of 5×10^{-5} with the same loss and ground truth. We conducted this cyclical process for 5 iterations. Since our goal here was not to train the best possible model, but rather to understand and analyze sources of confusion, we did not focus on hyperparameter optimization. We repeated this same process for the style activations. Ultimately, we found a train/test accuracy of 77.8% and 48.9% for content and 97.4% and 63.5% for style. The significantly better performance of style is not surprising; it makes far more sense that style is more associated with a certain artist than content would be. The significantly better performance of the train set indicates overfitting, despite the addition of dropout. While we'd have liked to add extra linear layers, with ReLU and dropout, memory constraints made a better linear classifier unworkable.

The relevant confusion matrices are displayed in Figure 6. Overall, performance is worse than the standard transfer learning with ResNet18. Additionally, while the ResNet18 errors revealed meaningful insights about artist relationships, the errors we observed mostly reveal limitations of the architecture and dataset. For example, we only use one convolutional layer in each case due to memory constraints, even though there are 4 layers that correspond to style. Hence, the feature space is not as expressive as it could be.

That said, we hypothesized that the content embeddings would confuse works in the same type from approximately the same time, and this may be true to some extent. For example, many engravers and printmakers from the Seventeenth Century are classified as Wenceslaus Hollar, who was a prolific artist matching that description. However, the confusion matrix also highlights class imbalance as an issue with this model. The vertical column of artists mistaken for George Breitner demonstrate this problem especially well. Breitner was a social realist who painted ordinary people in Amsterdam with completely flat color. The model incorrectly classifies many artists as Breitner, and some of those artists have no obvious connection to Breitner (e.g. Chodowiecki, a Classicist engraver and painter that used high contrast to depict wealthy Romans and mythological characters).

Style embeddings exhibit the class imbalance problem to a lesser extent, but also reveal some of the more meaningful cross-school relationships as observed in the ResNet18 architecture. For example, Mannerists and Baroque artists are confused again. With more time, we would have investigated whether balancing classes as a preprocessing step would generate more meaningful errors.

6. Conclusion/Future Work

We broadly investigated transfer learning as a means of classifying works in the Rijksmuseum collection by their artists. The best model, which began with a pretrained ResNet18 model, classified works from 25 artists with an accuracy of 82.5% on a held-out test set. This is a 7% improvement relative to the previous best single model architecture's performance on this dataset. This performance could likely be improved with intelligent model ensembling and more extensive hyperparameter tuning. In particular, our experimentation with different architectures for transfer learning serves as a testbed for future cross-depictional study. Due to the use of residual modules and reasonable depth, the ResNet18 seems best for tasks in which the target material and content differ significantly from the source. Additionally, classification based on style transfer may have some promise, but more careful engineering of model architecture, data preprocessing, and tuning is needed.

With so many museums looking to use computer vision to create mobile apps for enhanced museum visits, automated art classification is a worthwhile task in itself. However, we argue that the pipeline described here could be useful to art historians as well. From our small set of 25 artists, we identify a possible novel relationship between two artists that are not typically discussed together (Daniel Chodowiecki and Jozef Israels). With a larger classification space, others could use this pipeline as inspiration for new connections to investigate more thoroughly.

References

[1] J. Blitzer, M. Dredze, F. Pereira, et al. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, volume 7, pages 440–447, 2007.

[2] J. Bouvrie. Notes on convolutional neural networks. 2006.

[3] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua. Stylebank: An explicit representation for neural image style transfer. *arXiv preprint arXiv:1703.09210*, 2017.

[4] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu. Boosting for transfer learning. In *Proceedings of the 24th international conference on Machine learning*, pages 193–200. ACM, 2007.

[5] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2414–2423, 2016.

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[7] C. Gu and Y.-J. Kim. Penalized likelihood regression: general formulation and efficient approximation. *Canadian Journal of Statistics*, 30(4):619–628, 2002.

[8] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, pages 1135–1143, 2015.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[10] J. M. Hughes, D. J. Graham, and D. N. Rockmore. Quantification of artistic style through sparse coding analysis in the drawings of pieter bruegel the elder. *Proceedings of the National Academy of Sciences*, 107(4):1279–1283, 2010.

[11] C. R. Johnson, E. Hendriks, I. J. Berezhnoy, E. Brevdo, S. M. Hughes, I. Daubechies, J. Li, E. Postma, and J. Z. Wang. Image processing for artist identification. *IEEE Signal Processing Magazine*, 25(4), 2008.

[12] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.

[13] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[15] T. Mensink and J. Van Gemert. The rijksmuseum challenge: Museum-centered visual recognition. In *Proceedings of International Conference on Multimedia Retrieval*, page 451. ACM, 2014.

[16] S. J. Pan, V. W. Zheng, Q. Yang, and D. H. Hu. Transfer learning for wifi-based indoor localization. In *Association for the advancement of artificial intelligence (AAAI) workshop*, page 6, 2008.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[18] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[19] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[21] R. P. Taylor, R. Guzman, T. Martin, G. Hall, A. Micolich, D. Jonas, B. Scannell, M. Fairbanks, and C. Marlow. Authenticating pollock paintings using fractal geometry. *Pattern Recognition Letters*, 28(6):695–702, 2007.

[22] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *Int. Conf. on Machine Learning (ICML)*, 2016.

[23] 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*, 32(4):46–54, 2015.

[24] 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*, 32(4):46–54, 2015.

[25] M. Xie, N. Jean, M. Burke, D. Lobell, and S. Ermon. Transfer learning from deep features for remote sensing and poverty mapping. *arXiv preprint arXiv:1510.00098*, 2015.