

Determining Style of Paintings using Deep Learning and Convolutional Neural Networks

Sean Chang
schang18@stanford.edu

Jeffrey Chen
jchen623@stanford.edu

Patrick Mogan
pjmogan@stanford.edu

March 19, 2019



Fig. 1) *The Starry Night* by Vincent van Gogh [1]

1 Motivation

The study of artwork has figured prominently in the deep learning community, with applications ranging from neural style transfer to digital restoration of damaged paintings. Equally fascinating is analyzing its subjectivity. For example, different art historians have classified Vincent van Gogh’s 1889 masterpiece *The Starry Night* (Figure 1) as either an Expressionist or Post-Impressionist work, illustrating that interpretation of artwork and determination of artistic style are topics of debate even among experts in the field. We seek to explore the degree to which computers can learn the subjectivity of these classifications. The input to our algorithm is an image of a painting, which is then passed into different CNN models such as a shallow CNN and ResNet-50s. The output is a predicted style classification.

2 Related Works

We took inspiration from a similar task outlined in *Using CNN to Classify and Understand Artists from the Rijksmuseum*, by Tara Balakrishnan, Sarah Rosston, and Emily Tang [2]. In this study, the aforementioned authors sought to predict the creator of a painting from the image of that painting, and generated class activation maps for prediction analysis. We thought that predicting a subjective rather than objective label would be an interesting challenge that could provide insights into the subtleties of painting styles.

In addition, we were inspired to use a ResNet-50 by a paper written by Adrian Lecoutre, Benjamin Nègrevergne, and Florian Yger titled *Recognizing Art Style Automatically in painting with deep learning* [3]. We also used this paper as a reference to compare model results.

3 Dataset

We used the Kaggle dataset “Painters By Numbers”, which consists of roughly 103,000 unique images of paintings primarily from WikiArt.org, and their corresponding titles, styles, genres, artists, and dates [4]. Specifically, there are 125 styles accounted for throughout the entire dataset. The dimensions of images in the original dataset varied, thus we resized them to $256 \times 256 \times 3$ pixels. With our pre-processed dataset, we partition the dataset into an 80/10/10 train, dev, and test splits based on recom-

mended splits for datasets of similar sizes. Throughout model evaluation, we encountered bias towards particular styles of paintings due to high prevalence of these styles over others. As a result, we decreased the number of styles we classified to the 10 most represented styles in the dataset. Lastly, we performed data augmentation on underrepresented styles in order to increase the amount of training data and reduce potential overfitting.



Fig. 2) *Boulevard Montmartre: Afternoon, Sunshine* by Camille Pissarro [5]

4 Methods

4.1 Baseline Model

For our baseline, we began with looking at simple CNN architectures as CNNs are often applied to visual applications of deep learning. We began with a shallow three-layer CNN in Tensorflow, which was originally written for binary classification, and modified it to classify art styles [6]. We chose this architecture to quickly test a basic CNN and get familiar with processing our input data and interpreting the results of a CNN. The baseline CNN consists of the architecture represented in the table below (Figure 3).

$256 \times 256 \times 3$	Input
$256 \times 256 \times 16$	CONV 5×5
$128 \times 128 \times 16$	POOL 5×5
$128 \times 128 \times 32$	CONV 4×4
$64 \times 64 \times 32$	POOL 4×4
$64 \times 64 \times 64$	CONV 3×3
$32 \times 32 \times 64$	POOL 3×3
$104,448 \times 1$	Flattening
1024×1	Fully Connected
125×1	Fully Connected
125×1	Softmax

Fig. 3) Architecture for three-layer baseline CNN [6]

The general approach of this CNN was to reduce the image scale by a factor of two for each layer, and ideally extract the most important features as the model progressed. For the loss function, we used the cross-entropy loss function for multiclass classification to predict style:

$$L = - \sum_{c=1}^M y_{o,c} \log(p_{o,c})$$

Due to its shallow nature, we expect that low-level features were not recognized by the model as well as we would have liked, inspiring our decision to move forward with much stronger architectures.

4.2 Advanced Models

We next decided to move forward with a stronger architecture. In addition to increasing the number of layers, we decided to use a residual network, an architecture known for its success with image recognition using CNNs [7]. We picked the ResNet-50 over the ResNet-18 because at worst the ResNet-50 would train slower and provide a model with an effectiveness equal to or greater than that of the ResNet-18. Our model was created using PyTorch libraries, most importantly torchvision. For the loss function, we again used the cross-entropy loss function due to the nature of our multi-class classification task, and used an adaptive moment estimation (Adam) optimizer largely because it has shown to perform effectively on visual tasks employing CNNs [8]. The architecture is laid out in the table below.

$7 \times 7, 64, \text{stride } 2$	
$3 \times 3 \text{ max pool, stride } 2$	
$3 \times$	$\left\{ \begin{array}{l} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{array} \right\}$
$4 \times$	$\left\{ \begin{array}{l} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{array} \right\}$
$6 \times$	$\left\{ \begin{array}{l} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{array} \right\}$
$3 \times$	$\left\{ \begin{array}{l} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{array} \right\}$
Average pool	
1000-dimensional FC	
Softmax	

Fig. 4) Architecture for ResNet-50 [7]

The ResNet-50 model we used was pre-trained on ImageNet. As our application of classifying style of painting was similar to classifying ImageNet images, we first tested transfer learning on the ResNet-50, freezing all layers of the ResNet-50 model except the last fully connected layer, and then retrained on our training data. We continued testing this approach with additional data augmentation in order to better understand how manipulating our training data would impact the performance of our model.

In addition to transfer learning, we were interested in testing other hyperparameters applicable to the ResNet-50. As classifying style may require slightly different feature detection than image classification, we began to test different numbers of unfrozen residual layers in the ResNet-50, therefore allowing the model to retrain additional pre-trained layers. We experimented with unfreezing the last, last two, and last three residual blocks. We also added dropout to the fully connected layer at the end of the ResNet-50. The latter was a reaction to our model's tendency to overfit on our training data. Below are the train and validation accuracies for each model over 50 epochs, which guided our decision of which model to proceed with.

$$\text{Accuracy} = \frac{\# \text{ correct predictions}}{\# \text{ total predictions}}$$

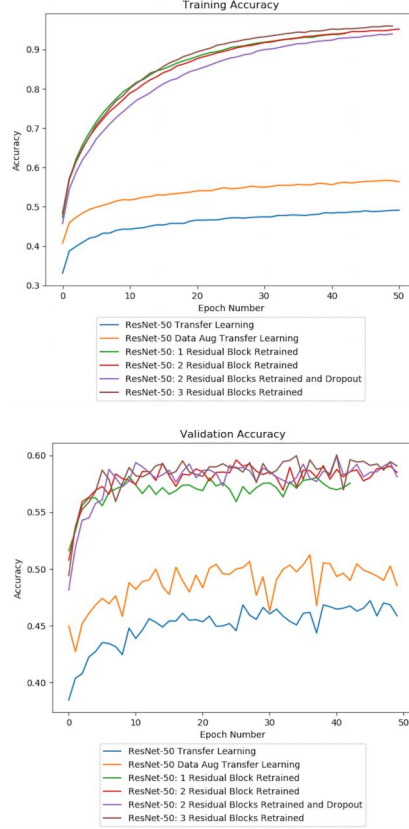


Fig. 5) Accuracy for train and validation sets for each ResNet-50 model tested

The code for training and evaluating our modified ResNet-50 model can be found at our GitHub repository listed at [9].

5 Results and Analysis

We chose to analyze precision, recall, and F1 score as metrics for our results. These metrics are standard amongst deep learning applications, and give us insight into where the model is potentially failing. Our results for all the models we tested are presented below (Figure 6).

Model	Precision	Recall	F1 Score
Baseline*	0.075	0.024	0.023
Resnet-50: Transfer Learning*	0.475	0.472	0.466
Resnet-50: Transfer Learning	0.516	0.514	0.509
Resnet-50: 1 Residual Block Retrained	0.592	0.588	0.587
Resnet-50: 2 Residual Block Retrained	0.587	0.581	0.582
Resnet-50: 3 Residual Block Retrained	0.608	0.592	0.593
Resnet-50: 2 Residual Blocks Retrained and Dropout	0.595	0.59	0.587

*Tested prior to data augmentation

Fig. 6) Validation results

Based on our results from Figure 6, we can obviously see that the baseline model is not performing well, and that a 3-layer CNN is not nearly strong enough for our application. This first iteration signaled to us that we should consider using the more complex architectures described above.

We see that our model improves once we apply data augmentation (Figure 6). Without data augmentation we observed that our model performed more poorly on images that belong to categories like Roccoco, Art Nouveau, and Symbolism because these styles were underrepresented in our original dataset and therefore our original training set.

After data augmentation, we observed a subtle improvement in our model's performance on validation images based on the confusion matrices below. Therefore, we continued forward with the augmented dataset (Figures 7 and 8).



Fig. 7) Validation results confusion matrix with unaugmented training data



Fig. 8) Validation results confusion matrix with augmented training data

Our best-performing model on the validation set was the ResNet-50 with three residual blocks retrained on augmented data. The final results for this model on the test set are displayed in the figure below.

Final ResNet-50 model test results	
Precision:	0.604
Recall:	0.593
F1 Score:	0.595

Fig. 9) Results for ResNet-50 with three residual blocks retrained

To better understand how we could potentially improve our model in the future, we created activation maps to visualize exactly how our model is interpreting misclassified images [10]. In these activation maps, the brighter pixels indicate greater relevance to the classification.

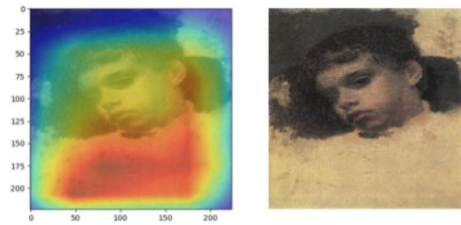


Fig. 10) *Portrait of Koyla (Nikolay) Simonovich* by Valentin Serov [11]

We observed that often our model would perform poorly on black and white images. The heatmap above for example, shows how our model prioritizes empty whitespace while predicting, when we should be using features extracted from the painting such as the face instead. This suggests we should train on more black and white images in order to detect more significant features rather than just the colors present in such photos.

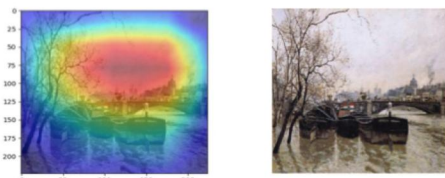


Fig. 11) *Flooding by the Seine* by Fritz Thawo [12]

Our current model also sometimes fails to recognize images that are in the foreground and instead prioritizes objects and empty spaces in the background. This suggests that we could potentially improve our model by adding more images to our training set that have subtle foregrounds and plain backgrounds.

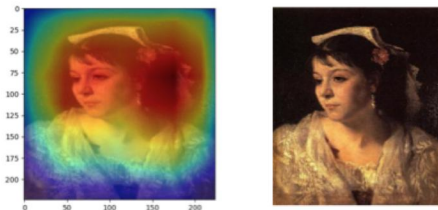


Fig. 12) *Head of an Italian Woman* by John Sargent [13]

However, our model currently performs very well on colored images where there is one distinct object or figure in the foreground. The activation map in Figure 12, for example, shows how our model appropriately uses facial features to classify *Head Of An Italian Woman* as Realism.

Ultimately, these class activation maps suggest appropriate next steps would be to add more black and white, nondescript images of paintings to our

dataset to improve our model's ability to classify paintings with a limited color palette and a complex foreground.

6 Conclusion and Future Work

Our final model did not perform as well as a panel of art experts would, but it performed comparably to models built for similar tasks [3]. Most mistakes involved classifying a painting's style as one that is represented more frequently in the original dataset, suggesting the need to train on more paintings from underrepresented classes.

In the future, we'd like to reduce overfitting, a problem experienced by most models we tried. We think that using early stopping could be effective, because the degree of overfitting seemed to increase in later epochs of training. We'd also like to use weight decay and add more unique training examples of underrepresented classes.

7 Contributions

Each team member played a vital role in completing this project. Jeffrey was critical in maintaining our GitHub repository and training our models on our AWS account. Additionally, Jeffrey wrote scripts to create class activation maps in order to assist with error analysis. Patrick contributed heavily by creating scripts to preprocess our dataset as well as augment the train data in order to improve training. In addition, Patrick was also involved with model result interpretation by creating F1 score, precision, and accuracy scripts. Sean was heavily involved with setting up the ResNet-50 scripts as well as testing hyperparameters on the ResNet-50 model such as number of frozen layers and dropout rate. Sean aided in error analysis by creating the confusion matrix script for visual interpretation. All three worked together to understand existing code bases, debug code, evaluate model performance, and present our results. Special thanks to our Teaching Assistant Sarah Najmark who guided us with project direction, model architecture, and use of class activation maps.

8 References

- [1] V. van Gogh, The Starry Night. San Francisco: Museum of Modern Art, 1889.
- [2] T. Balakrishnan, S. Rosston, and E. Tang, “Using CNN to Classify and Understand Artists from the Rijksmuseum.” 2017 [Online]. Available: Stanford University. Accessed March 16, 2019.
- [3] A. Lecoutre, B. Nevgrevergne, and F. Yger, “Recognizing Art Style Automatically in painting with deep learning.” JMLR: Workshop and Conference Proceedings. 2017. [Online]. Available: Univeriste Paris Dauphine. Accessed March 16, 2019.
- [4] Kaggle. *Painter by Numbers*. 2017. Web. 16 January 2019.
<https://www.kaggle.com/c/painter-by-numbers>.
- [5] C. Pissaro, Afternoon, Sunshine. St. Petersburg, Russia: State Hermitage Museum 1897
- [6] perseus784. BvS. 2018. Web. 12 February 2019.
<https://github.com/perseus784/BvS>
- [7] K. He, X. Zhang, S. Ren, and J. Sun. “Deep Residual Learning for Image Recognition.” *Computer Vision and Pattern Recognition*. Dec, 2015. [Online]. Available: Arvix, arxiv.org. Accessed March 16, 2019.
- [8] D. Kingma, J. Lei Ba, “Adam: A Method for Stochastic Optimization.” *Computer Vision and Pattern Recognition*. 2014. [Online]. Available: Arvix, arxiv.org. Accessed March 16, 2019.
- [9] jdchen623. CS_230_Final_Project. 2019. Web. 19 March 2019.
https://github.com/jdchen623/CS230_Final_Project
- [10] I. Pointer, “Class Activation Mapping In PyTorch,” Snappish Thoughts. [Online]. Available: <http://snappishproductions.com/blog/2018/01/03/class-activation-mapping-in-pytorch.html>. Accessed March 15, 2019.
- [11] V. Serov, Portrait of Koyla (Nikolay Simonovich). 1880.
- [12] F. Thalow, Flooding by the Seine. 1893
- [13] J. Sargent, Head of an Italian Woman. 1878.