The Power of Transfer Learning in Artist Identification

Xingyu Zhou*
Department of ECE
The Ohio State University
zhou.2055@osu.edu

Abstract

Artist identification of fine art paintings is a challenging problem primarily handled by art historians with extensive training and expertise. Many previous works have explored this problem by explicitly defining classification features. Instead, we apply transfer learning in this context so as to identify the artist of a painting as accurately and precisely as possible. We train a variety of models ranging from a simple CNN designed from scratch to a VGG16 network with transfer learning. We show that the best model with transfer learning is able to achieve a test accuracy of 98.3%, which is significantly higher than the baseline CNN. Additionally, we perform multiple experiments to explore and understand the learned representation of our networks. Our results demonstrate that transfer learning is a powerful tool for artist identification and that when asked to identify artists, they are able to learn a representation of painting style.

1 Introduction

Artist identification is the task of identifying the artist of a painting given no other information about it. This is an important requirement for cataloguing art, especially as more and more art is increasingly digitized. One of the most comprehensive datasets, Wikiart, has around $150,000$ artworks by $2,500$ artists [2]. Artsy has a growing collection with the aim of making all art easily available and accessible online [1]. As these collections grow, it becomes increasingly important to be able to efficiently label and identify newly digitized art pieces. Thus, a reliable way to identify artists is not only useful for labeling art pieces but also for identifying forgeries, another important art historical problem.

Artist identification was previously often performed by art experts, who have the necessary backgrounds and experience. However, as the number of digitized art works increases everyday, it becomes imperative for human experts to do all these category tasks. Thus, it requires a more efficient way to identify the paints of different artists. The covolutional neural networks (CNN) with transfer learning is the rescue. In this paper, we successfully apply this method and our main contributions of can be summarized as follow.

- We first deal with the identification between Claude Monet and Vincent Van Gogh. Then, we try to solve a harder problem to distinguish between Claude Monet and Alfred Sisley, since they are exactly in the same period, have the same styles and often paint the same landscape together.
- For both of the problems, we first train a simple CNN as the baseline. Then, we apply two different ways of transfer learning, i.e., *feature extraction* and *fine-tuning*. By using feature extraction, we adopt the VGG16 pre-trained on ImageNet to extract the useful features and

*Homepage: xingyuzhou.org

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

(a) (b) (c)

Figure 1: Sample images in our data set: (a) is from Claude Monet, (b) is from Vincent Van Gogh, (c) is from Alfred Sisley

.

then use these generated features to train a classifier. By using fine-tuning, we collect a dense layer onto the base of VGG16 and train the whole network with certain layers frozen.

- For the first problem, our best model can achieve a test accuracy of 98.3% while the baseline CNN can only achieve 83.3%. For the harder problem, the accuracy of our best model is 86.6% while the baseline CNN is only 71.1%.

2 Data set

2.1 Overview

In order to train our model, we first need to collect the art works for each artist. In particular, we collect 300 images for each artist from Wikiart [2]. This is relatively small data set; however, we are going to show that even with this small data set, we are able to achieve a very high accuracy with transfer learning, which effectively demonstrates the power of transfer learning. We split the data set into training, validation and test sets using 80-10-10 split per artist. As a result, the training set of each artist consists of 240 images. The validation and test sets each has 30 images for each artist. Some sample images of our data set are shown in Fig. 1

2.2 Preprocessing and data augmentation

In order to fit our images to the pre-trained VGG16 model, we need first resize each image into 224×224, which is the standard input size of VGG16. Since our data set is relatively small, we adopt data augmentation to combat overfiting. In particular, we randomly horizontally flip the input image with a 50% probability. This randomness adds variety to the training data and helps avoid overfitting. However, we do not adopt other types data augmentation. The reason is that some of them, such as rotation and whiting, might result in the loss of original style information.

3 Method

We develop and train three different architectures (models) to identify artists. Every network we use takes as input a $224 \times 224 \times 3$ RGB image and outputs the scores for each of the two artists present in our dataset at each time.

For all of our networks, we use a softmax classifier with cross-entropy loss,

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right), \tag{1}$$

where L_i is the loss for example i in the training mini-batch, f is the score for a particular class calculated by the network, j is one of the possible classes, and y_i is the correct class for example i. This loss function ensures that our network is constantly trying to maximize the score of the correct artists of its training examples relative to the other during training.

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 222, 222, 32)      896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 111, 111, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 109, 109, 64)      18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 54, 54, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 52, 52, 128)       73856
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 26, 26, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 24, 24, 128)       147584
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 12, 12, 128)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 18432)             0
_________________________________________________________________
dropout_1 (Dropout)          (None, 18432)             0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               9437696
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 513
=================================================================
Total params: 9,679,041
Trainable params: 9,679,041
Non-trainable params: 0
_________________________________________________________________
```

Figure 2: Architecture for our baseline CNN

.

3.1 Baseline CNN

We train a simple CNN from scratch for artist identification. Figure 2 shows the architecture of this network. As the name implies, this network serves as a baseline for comparison with the transfer learning approach. Every layer in the network downsamples the image in order to reduce computational complexity, but the downside of this approach is that it might not allow sufficient exploration of lower-level features as the image details are quickly aggregated.

3.2 Transfer learning

There are two ways to use pre-trained model to do transfer learning: *feature extraction* and *fine-tuning*. We will implement both of them.

3.2.1 Feature extraction

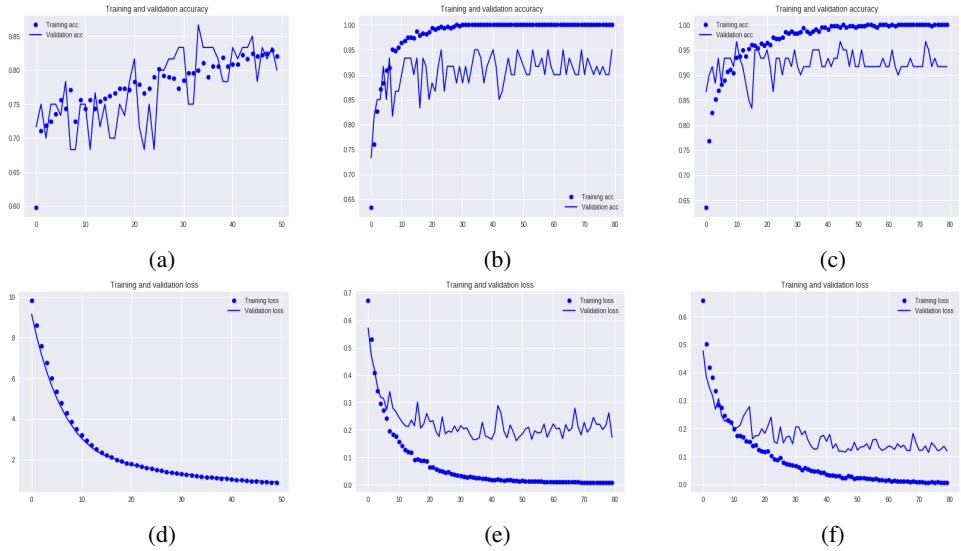
Feature extraction consists of using the representations learned by a previous network to extract interesting features from new samples. These features are then run through a new classifier, which is trained from scratch. In the case of convnets, feature extraction consists of taking the convolutional base of a previously trained network, running the new data through it, and training a new classifier on top of the output.

In our case, we first use the base of VGG16 to extract the features for our training, validation and test images. Then, we use these generated features to train a simple two-class classifier. One drawback of this approach is that it does not allow us to use data augmentation.

3.2.2 Fine-tuning

Another approach is to extend the convolutional base of VGG16 by adding dense layers on top, and running the whole thing end to end on the input data. This will allow us to use data augmentation, because every input image goes through the convolutional base every time it's seen by the model. But for the same reason, this technique is far more expensive than the first.

In our case, we first frozen the all the convolutional base of VGG16, unfrozen the last dense layer and train the whole network. Then, we further unfrozen the last two layers to improve the result.



(a) (b) (c)

(d) (e) (f)

Figure 3: Accuracy and loss with the epoch for our three models: (a)(d) are Baseline CNN; (b)(e) are Feature extraction; (c)(f) are Fine-tuning

.

4 Experiments

4.1 Setup

All of our models and experiments are implemented in Keras with tensorflow backend. We follow the guideline in [3] to set up the transfer learning on VGG16. All experiments were performed on Google Colaboratory, which is free and more importantly it supports K80 GPU, which is quite essential to our experiment.

4.2 Implementation details

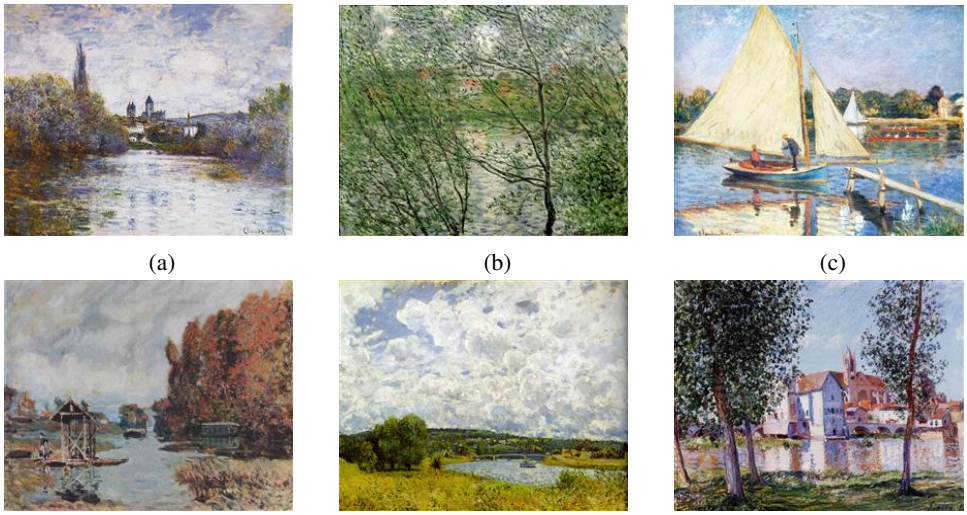
The optimizer used in this work is RMSprop with learning rate $2e^{-5}$. We also tried other optimizers, but we found that RMSprop is very easy to train and has the best result. The mini-batch size is 10 with training time of 80 epochs. The L_2 regularization is $1e^{-5}$. To handle overfitting, we introduce dropout with probability 0.5. It is worth pointing out that transfer learning with VGG16 is very easy to train compared to other pre-trained model in our case. We tried ResNet and Xception, but the results are not as good as that of VGG16.

5 Results

5.1 Quantitative analysis

In the first experiment, we focus on the identification between Monet and Van Gogh. The baseline CNN trained from scratch is able to achieve a test accuracy of 83.3%, which is acceptable. By using the feature extraction approach, we are able to achieve a test accuracy of 94.6%. Further, by using the fine-tuning approach, we are able to achieve a test accuracy of 98.3%, which means that our best model only misclassifies *one* of all 60 test images. This result sufficiently demonstrates the power of transfer learning in artist identification.

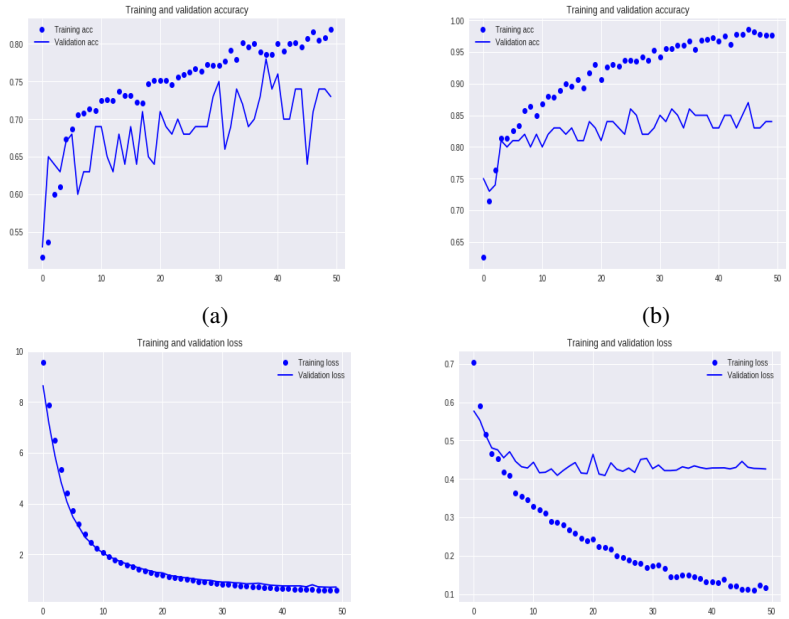
Figure 3 shows how the training and validation accuracies and loss of our networks changed during training. We can see that in the baseline CNN, training and validation accuracies track very closely together, indicating little overfitting. This could mean that performance of it would improve with more training epochs and more training data as we generally expect to see a small but noticeable gap between training and validation/test accuracy in a fully-trained network. The transfer learning does



(a) (b) (c)

(d) (e) (f)

Figure 4: Similarity between Monet's and Sisley's works: (a)(b)(c) are Monet's works; (d)(e)(f) are Sisley's works.



(a) (b)

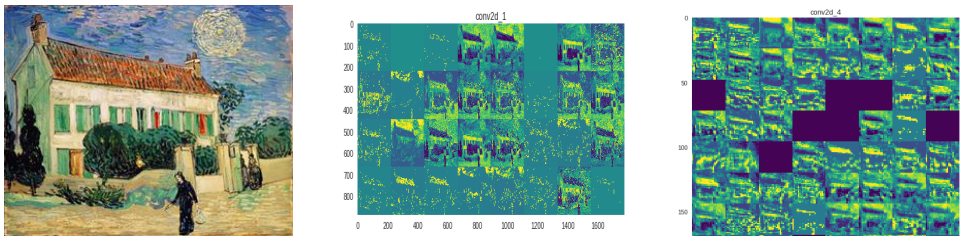
(c) (d)

Figure 5: Accuracy and loss with training epoch: (a)(c) are Baseline CNN; (b)(d) are transfer learning with fine-tuning

.

have a gap between training and test accuracy, indicating that adding more training data and further training might not add as much value.

In our second experiment, we focus on the identification between Monet and Sisley. This is really a harder problem, because their works enjoy a lot of similarity, as shown in Figure 4. Despite this difficulty, we show that with transfer learning, our best model can achieve a test accuracy of 86.6%, while the baseline CNN can only achieve an accuracy of 71.1%.



(a) (b) (c)

Figure 6: Activations visualizations: (a) is the initial image, (b) is the output of the first conv layer, (c) is the output of the fourth conv layer

.

(a) (b)

Figure 7: Predict regular images transformed by Van Gogh's style

.

5.2 Qualitative analysis

First, we try to understand the activations in our models. This is useful for understanding how successive convnet layers transform their input, and for getting a first idea of the meaning of individual convnet filters. Figure 6 illustrates the initial image and the activations of the first conv layer and the fourth conv layer, respectively. From it, we can summarize our key observations.

- The first layer acts as a collection of various edge detectors. At that stage, the activations retain almost all of the information present in the initial picture.
- As it goes higher, the activations become increasingly abstract and less visually interpretable. Higher presentations carry increasingly less information about the visual contents of the image, and increasingly more information related to the class of the image.
- The sparsity of the activations increases with the depth of the layer: in the first layer, all filters are activated by the input image; but in the following layers, more and more filters are blank. This means the pattern encoded by the filter isn't found in the input image.

Second, we would like to understand whether or not our model has really learned the style of each artist. To this end, we take style-transferred versions of regular images and run them through our network to obtain a predicted artist. Figure 7 shows two regular images transformed to have the style of Van Gogh. The model trained with transfer learning is able to identify both of them as Van Gogh's works, indicating that the network is able to recognize Van Gogh's style independent of the specific content of the painting.

6 Conclusion

In this paper, we apply transfer learning to artist identification problem. We trained a simple CNN from scratch as the baseline. Then, we adopt two different approaches of transfer learning. The first one is called feature extraction. In particular, we use pre-trained VGG16 without the final layer to extract useful features, and then use these generated features to train a simple classifier. The other one

is to directly add a dense layer on top of the VGG16 base, and then frozen certain layers to train the whole network. Our best model is able to enjoy a test accuracy of 98.3% in distinguishing between Monet and Van Gogh, and a accuracy of 86.6% in distinguishing between Monet and Sisley due to the similarity between Monet's and Sisley's works.

Future works will be explored to improve the accuracy in distinguishing between Monet and Sisley in the hope that we can at least achieve 90% accuracy. Possible methods that can be used include model ensemble, batch normalization, dynamical rate adjustment and other pre-trained models.

References

[1] Artsy. `https://www.artsy.net/about/the-art-genome-project`.

[2] Wikiart. `https://www.wikiart.org/`.

[3] Francois Chollet. *Deep learning with Python*. Manning Publications Co., 2017.