Paintings Without Painters: Artist Fingerprinting and Style Transfer

Morgan Hobson, Ziyan Jiang, Tan Nguyen

***Abstract*—In recent years, people explore different applications of deep learning, trying to prove that this technology can match human's abilities in areas that only humans can do, one of which is arts. In this paper, we want to explore this potential of deep learning understanding arts by testing whether it can differentiate paintings coming from the same artists or not using Siamese architecture with different models. Afterwards, we want to use deep learning for style transfer: apply a new art style for an image from another style, using image-to-image neural transfer and CycleGAN.**

I. Introduction

For our project, we choose to tackle the problem presented in the Painter by Numbers Kaggle competition[1]. The competition wants to design a model to recognize whether two paintings are by the same artist or not. Part of what makes this problem unique is that the model is not meant to recognize specific artists, but rather the features of the paintings and make the decision. Therefore, even if the model encounters artists it was not trained on, it still need to perform well. We chose to develop and compare the results of models developed around three different model discussed in class: ResNet18, AlexNet, and VGG16[3] while applying on an adjusted Siamese architecture.

Initially, Siamese architecture was our only goal; however, after receiving feedback from our initial proposal, we followed up our work with style transfer, translating paintings from one style to another. In the dataset provided by the competition, in addition to assigning each painting to an artist, images were also labeled as belonging to specific styles (ie. Art Nouveau, Romanticism). We designed and tested models using two different techniques to accomplish this: image-to-image style Transfer (one to one) and CycleGAN (collection to collection).

II. Dataset and Preprocessing

The Painter by Numbers competition provided labelled training and testing datasets for developing our models. The training dataset contained 77,104 images while the testing dataset contained 23,812 images. To regularize model input, we scaled each image down so that its shorter dimension was 256 pixels and then cropped the ends of the longer dimension to give each painting a resolution of 256×256 without stretching the image. Cropping the images removed some of their visual information, but in general the middle square of a painting is representative of its features.

To make our results repeatable and cut down on processing time, the classification models were trained and tested on pre-generated pairs of images with an equal distribution of matching and differing assignments. Our final models were trained on a set of 120,000 pairs, validated on a set of 3,000 pairs, and tested on a set of 30,000 pairs. When running analysis on these two datasets, we find that our training dataset is balanced: 60,165 pairs from the same artists (50.13%). Similarly, our testing dataset is also balanced: 14,925 pairs from the same artists (49.75%). Therefore, we can guarantee that our model can be trained properly.

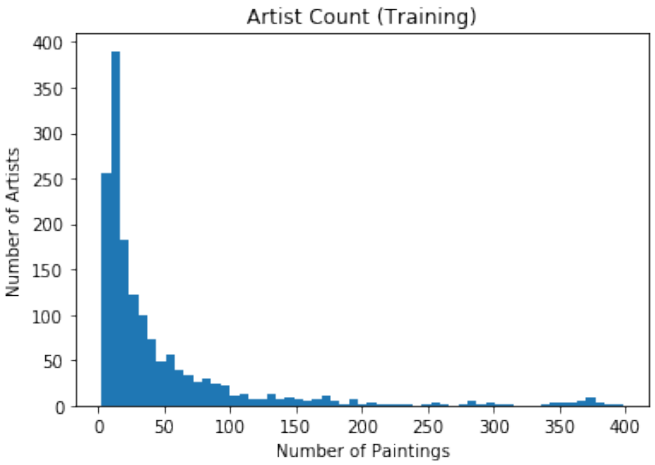

Fig. 1: Histogram showing the number of paintings featured from each artist in the training set. Mean: 48.7 Median: 21

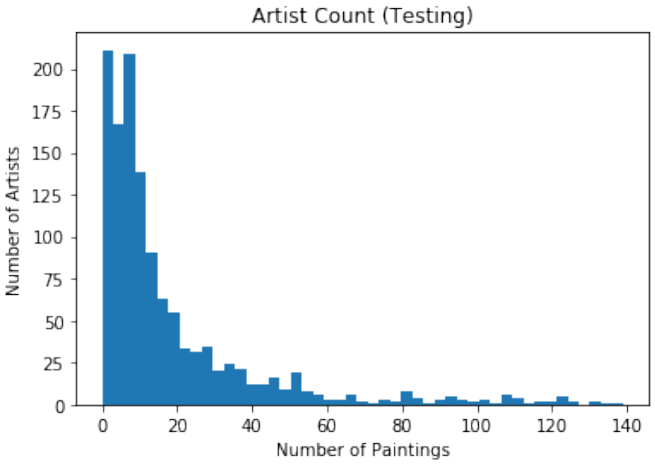

Fig. 2: Histogram showing the number of paintings featured from each artist in the training set. Mean: 18.0 Median: 9

However, the dataset do not have a uniform distribution of

paintings per artist. As shown in figures 1 and 2, the number of paintings attributed to an artist in each dataset varied widely. This could have contributed to the overfitting that occurred in each model, as files used in pairs were randomly sampled from each set according to the label being generated, and thus artists with more paintings in the sets occurred more often in training and testing.

In order to emphasize the fact that the models were not intended to identify specific artists, the datasets provided were selected from artists such that there were many artists exclusive to training or testing. 14.1% of paintings used in the training dataset were by artists exclusive to training, while 3.3% of paintings used in the testing set were by artists exclusive to testing.

III. Artist Fingerprinting

Although the problem posed here seems like a simple classification problem, in reality, we cannot apply traditional architecture because we do not know the number of classes for labels nor we want to. We need to feed in two inputs to the model and classify whether the inputs are similar or not. Therefore, we choose Siamese architecture [4].

For Siamese architecture, we want to feed two inputs to the same model, producing two outputs. Then, we want to minimize Contrastive Loss [2] between these two outputs to see if their outputs are relatively close to each other.

$$L(W,Y,X_1,X_2) = (1-Y)\frac{1}{2}(D_W)^2 + (Y)\frac{1}{2}\{max(0,D_W)\}^2 \quad (1)$$

Here, D_W is the euclidean distance between the outputs of the sister Siamese networks. However, Contrastive Loss requires us to provide 2 new hyperparameters: $margin$ which defines the accepted range of results from an output, and $threshold$, which defines the accepted distance for similarity. These two hyperparameters differ for different datasets, and because of the complexity of our models, it is almost impossible to tune these two correctly. In fact, when using contrastive loss, we constantly get supoptimal accuracy below 50%. Therefore, we move on to a different approach.

Instead of using traditional Siamese architecture, we concatenate the two outputs and feed them into a fully connected net with a Sigmoid function as activation function at the final layer. This result proves to be more efficient and accurate, as our model performs relatively well.

For our model, we use three models: AlexNet, VGG16, and ResNet18. Instead of training the model from scratch, we use Pytorch pretrained model. For these models, we use the same architecture: loss function, number of epochs (10 epochs), same optimizer, etc. The table below shows the results of our Siamese architecture running with on Google Cloud using Ubuntu 16.04 LTS with NVDIA Tesla K80 GPU Accelerator.

Model	Training Acc (120k)	Testing Acc (30k)
VGG16	0.95	0.7265
AlexNet	0.97	0.6956
ResNet18	0.83	0.7363

Model	Training Loss (120k)	Testing Loss (30k)
VGG16	0.1337	0.0484
AlexNet	0.0737	0.0641
ResNet18	0.3749	0.0056

Model	Running Time (per epoch)
VGG16	119 minutes
AlexNet	33 minutes
ResNet18	19 minutes

From the table result above, we can see that VGG16 and Resnet18 outperformed AlexNet, though just by values, all three models overfitted, especially VGG16 and AlexNet where the discrepancy between training accuracy and testing accuracy is quite large. We also then want to consider the loss function and accuracy over time.

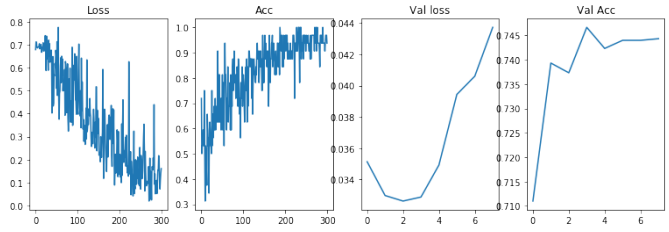

Fig. 3: Loss function and accuracy for VGG16

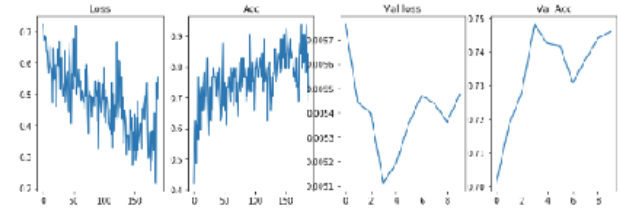

Fig. 4: Loss function and accuracy for Resnet18

One of the reasons for this overfitting is the dataset where some artists have way more paintings than others. Another reason is because we do not provide the model with enough data points. With such complex model, it requires way more than 120,000 pairs to reach good accuracy. However, we did not enough computing power to run these models in a reasonable time. Looking at ResNet18, we can see that the training accuracy is still quite low, meaning the model has the potential to perform even better with a larger dataset. Among the three models, ResNet18 performs the best and with the lowest running time as well.

When running these architectures, we specifically use pretrained model. In the beginning, we start with training these models from scratch; however, it takes us way too long per epoch while achieving very similar results, sometimes even worse. Using transfer learning from a pretrained model helps us save time and achieve decent results.

IV. Style Transfer

As mentioned above, our dataset has label of ”style”, so we want to try to transfer between different art styles. In this part, we will use two methods to fulfill the neural transfer work.

The first algorithm is developed by Leon A. Gatys et al. which implement a image-image neural style transfer [5]; The second algorithm is proposed by Jun-Yan Zhu et al. which implement a collection-collection transform based on CycleGAN [6].

A. *Neural Style Transfer Using Convolutional Neural Network*

The algorithm is the first style transfer method based on convolutional neural network. The algorithm produces a new image which combines the content of one image and style of another. This is considered an optimization problem as it tries to balance between content and style by considering both content loss and style loss. The convolutional neural network takes an input, and transform it to minimize the loss function based on the content image, style image and CNN architecture. For content-loss, distance between feature maps of input image and content image is calculated in different neural network layers. For style-loss, Gram matrix is used to calculate the feature correlations of multiple layers, so it is possible to capture texture information of the style image.

We use one image from Art Nouveau as content image and one image from Romanticism as style image.

Fig. 5: The bottom images have different ratio between matching the content and matching the style. A higher emphasis on the content in the bottom left and on the style in the bottom right.

B. *CycleGAN*

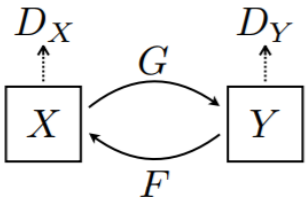

Fig. 6: CycleGAN

CycleGAN was introduced to solve the unpaired image-to-image translation problem. Unlike neural style transfer, we feed the network with two collections of images of different styles rather than two images. The framework is indicated as above.

CycleGAN uses two generators and two discriminators. The generator G converts images from the X domain to the Y domain. The other generator F converts images from Y to X. Each generator has a corresponding discriminator, which attempts to tell apart its generated images from real ones.

The CycleGAN objective function contains an adversarial loss and a cycle consistency loss. The adversarial loss is applied as a binary cross-entropy loss. The adversarial loss is:

$$L_{adv}(G, D_X, D_Y) = E[\log D_Y(y)] + E[\log(1 - D_Y(G(x)))] \tag{2}$$

When you convert an image to the other domain and back again by successively pass through both generators, you should get back something similar to the original image. That's the cycle consistency loss which is defined as a L_1 loss.

$$Ł_{cyc}(G, F, X, Y) = \frac{1}{m}\sum_{i=1}^{m} ||F(G(x_i) - y_i)||_1 + ||G(F(y_i) - x_i)||_1 \tag{3}$$

There are different networks used for the generator and the discriminator. In our experiment, we adopt Resnet-Generator. The generator includes three sections: an encoder, a transformer, and a decoder. The transformer has 9 residual blocks since our image has a resolution of 256 * 256. For discriminator architecture,we use 70 * 70 PatchGAN. Rather than deciding if the whole image is real or not, PatchGAN looks at patches of image and output a possibility matrix. We will average the possibility matrix to get this possibility.

Fig. 7: Transfer from Romanticism to Art Nouveau (Modern)

Fig. 8: Transfer from Art Nouveau (Modern) to Romanticism

Looking at the transformation of the paintings over multiple epochs, we can see how CycleGAN learns from the style, rather than just simple features like colors. Looking at epoch 10, we can see that the color schemes of both styles shifted to that of the other. However, the more epochs, the more

the color shifted back to the original color and the texture and style changes become more prominent. In particular, the shift from Romanticism to Art Nouveau (Modern), according to art history, increases contrast and emphasizes bodies as is common in lithography. The shift from Art Nouveau (Modern) to Romanticism makes colors splotches and muted and emphasizes shading where Art Nouveau (Modern) would typically hold firmer color blocking. Though it is hard to see that the style is fully converted, we see that CycleGAN makes a good attempt at converting between these styles.

C. Comparison Between Neural Style Transfer and CycleGAN

Here we will compare the result of different style transform methods using Art Nouveau and Romanticism painters.

For the neural style transfer, we input one Romanticism image as content image and two Art Nouveau images as style image separately. Also, to compare the transfer result against entire style collection, we compute the average Gram Matrix across the target domain and use this matrix to transfer the average style with the algorithm.

For the CycleGAN, we choose 1,000 images from both collections and do the training. For testing we then run the models on different Romanticism images as the input so that we could get the new images with Art Nouveau style.

For testing purposes, we use the same input for both methods to provide the outputs.

Fig. 9: We compare CycleGAN with neural style transfer(Gatys et al.).Left to right: input image, results from neural style transfer using two different images as style images, results from neural style transfer using all the images from the target domain, and CycleGAN.

We could find that the result from CycleGAN is more natural-looking and closer to the target style. And the generated image from neural style will be affected largely by the specific style image. Working with the average Gram Matrix gives us a better result.

V. Discussion

A. Artist Fingerprinting

From our models, we can see that most architectures overfit after four or five epochs, showing that we do not have enough data points in the dataset, leading the network to overfit. It is possible to increase the size of the dataset, so the model can generalize better, thus achieving a better testing accuracy. It is possible, in our belief, that given more time, our Resnet architecture can reach up to 90% accuracy.

Secondly, we also want to test with different models as well, using more complicated models like InceptionV2, different VGG and Resnet models, etc. In particular, we also tested out VGG19 and Resnet152. For the same dataset, VGG19 takes around 130 mins/epoch and Resnet152 takes around 120 mins/epoch, but both models respectively provide similar results compared to their counterparts. However, these complex models may benefit with a much larger datasets. Moving forward, adjusting hyperparameter as well as different models can benefit the overall accuracy as well.

B. Style Transfer

Given two approaches to style transfer, such as using convolutional neural network and CycleGAN, we can see that CycleGAN provides better, more generalized, and more natural results compared to neural transfer. Since the style is learnt from multiple images, it helps with generalization.

From our model of CycleGAN, there are still opportunities to improve. It is recommended in the original implementation to run more epochs, up to 200 epochs. As demonstrated above, paintings after 30 epochs show a more apparent style transfer and it may not be the best yet. Therefore, more epochs can definitely help.

Simultaneously, CycleGAN can be improved by changing its architecture. Another option for CycleGAN can be a different loss function. Many suggested that CycleGAN performs better when using Mean Square Error Loss Function instead of Binary Cross Entropy Loss Function because MSE Loss is more stable during training and generates higher quality results. Also, instead of using ResNet, many implement U-Net as well, thus can be an option for our model.

C. Things we learnt

- Implement from scratch different models and architecture and see many exciting results.
- Utilize different resources to maximize running time for various tasks of the project.
- Different applications and different extensions (CycleGAN) of common models

D. Advice to next year's DL students and instructors

For students:

- Choose an interesting final project, so you will enjoy and learn a lot from it.
- Start homework, projects early. Code takes time to run, so don't wait until last minute.
- Have a good background in mathematics and some basic understanding of machine learning, so the experience in class will be easier.

For instructors:

- Dive deeper into advanced topics and applications of deep learning
- More organized in homework and be more responsive
- Have more application-based homework to prepare for final project.

REFERENCES

[1] Painter by Numbers Kaggle competition https://www.kaggle.com/c/painter-by-numbers

[2] Hadsell, Raia, Sumit Chopra, and Yann LeCun. "Dimensionality reduction by learning an invariant mapping." null. IEEE, 2006.

[3] Canziani, Alfredo, Adam Paszke, and Eugenio Culurciello. "An analysis of deep neural network models for practical applications." arXiv preprint arXiv:1605.07678 (2016).

[4] Koch, Gregory, Richard Zemel, and Ruslan Salakhutdinov. "Siamese neural networks for one-shot image recognition." ICML Deep Learning Workshop. Vol. 2. 2015.

[5] Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "Image style transfer using convolutional neural networks." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.

[6] Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." arXiv preprint (2017). https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix

[7] CycleGAN Reference https://github.com/aitorzip/PyTorch-CycleGAN

[8] Style Transfer Reference https://pytorch.org/tutorials/advanced/neural_style_tutorial.html