Detection and Classification of Artistic Styles in Photographed Artwork using Deep Learning

Nanna Katrin Hannesdottir, Cole Hunter, Kevin Vogt-Lowell[1]

Boston University
Department of Electrical and Computer Engineering
8 Saint Mary's Street
Boston, MA 02215
www.bu.edu/ece

May 6, 2022

Technical Report No. ECE-2022-3

[1] All authors equally contributed to this work.

Summary

This is the Final Report for our project on art style classification in ENG EC 520: Digital Image Processing and Communication. We will detail our comparison of two Deep Neural Networks that are able to take in photos of artwork, and then output labels of the different styles present in the image. Included in this report is a brief literature review, a description of the problem, our proposed solutions, how we implemented them, and our experimental results.

Contents

1. **Introduction** **1**
2. **Literature Review** **1**
3. **Problem Statement** **2**
4. **Implementation** **5**
5. **Experimental Results** **8**
6. **Conclusion** **13**
7. **References** **14**

List of Figures

Fig. 1 Isolated scaling vs. compound scaling

Fig. 2 Definition of scaling constants and coefficient

Fig. 3 Feature extraction using standard convolution

Fig. 4 Depth wise convolution splits the feature extraction into two stages

Fig. 5 Visual Transformer Architecture

Fig. 6 Examples of Images with respective labels from the WikiArt dataset

Fig. 7 Importing a pre-trained EfficientNetV2 model

Fig. 8 Softmax Activation

Fig. 9 Results of ViT hyperparameter tuning resulted in an optimal learning rate of 0.06 and an optimal batch size of 128.

Fig. 10 Kullback-Leibler Divergence, and the Joint Probability Distribution of Points x_i and x_j in Original Dimension(p) and corresponding points y_i and y_j in Reduced Dimension (q)

Fig. 11 t-SNE Scatter Plots of ViT Testing Data After PCA modification

Fig. 12 t-SNE Scatter Plots of EfficinetNetV2 Testing Data After PCA modification

Fig. 13 Confusion Matrices for ViT (left) and EfficientNetV2 (right)

List of Tables

Table 1 Validation Accuracy Results

Table 2 Testing Accuracy Results on Non-Augmented Test Set

Table 3 Testing Accuracy Results on Augmented Test Set

1 Introduction

The artistic style of a painting is a label rich with information, describing characteristic visual attributes like texture, color, and object interaction in a piece, as well as information regarding historical context. A stylistic label is composed of many artistic nuances and their relations, which poses a complex image processing and classification problem with very interesting applications. For example, what if there existed a tool that allowed individuals to curate, understand, and verbalize their unique personal styles and preferences purely using visual information? The inspiration for our project was an idea for a software application in which users can take photos of art they find and like, and then immediately receive information detailing the artistic styles/themes present in the picture via deep learning, making personal style curation easier and more accessible to individuals regardless of background. With that end goal in mind, we implemented, fine-tuned, and compared two deep neural networks for art style classification: the EfficientNetV2, based on convolutional neural networks, and the Visual Transformer, based on successful transformer architectures from natural language processing.

2 Literature Review

Prior to the introduction of convolutional neural networks (CNN), one of the most important tasks in designing successful image classifiers was defining algorithms capable of extracting relevant features from an image. However, successful algorithmic feature extraction often depended on outside domain knowledge from subject-matter experts, making this method difficult to implement efficiently. One of the primary advantages of CNNs is their ability to effectively perform automatic feature extraction. Instead of using rigid predefined filters, CNNs model filter coefficients as weights within the network, constantly updating them throughout the learning process in order to minimize the loss.

For the specific problem of art style recognition in images, CNN-based solutions have been widely studied. In ”Recognizing Art Style Automatically with deep learning” by Lecoutre et al. (2017), the research team pretrained ResNet and AlexNet, two popular CNN models at the time, on object recognition using ImageNet and showed that, through fine-tuning, the networks could obtain reasonably high performance for artistic style detection. Furthermore, Lecoutre showed that the proportion of layers that were fine-tuned in their models had a significant effect on model performance: training about 20% of the layers in either model appeared to maximize successful style classification, with deeper fine-tuning deleteriously affecting shared high-level information and shallower fine-tuning failing to specialize the model enough for artistic style recognition. Since the publication of this paper, newer and more advanced models have been continuously developed and deployed, but not on the task of artistic style classification.

3 Problem Statement

The problem of art style classification involves the design of a model or some sort of mechanism that can distinguish the style of an artwork instantly from the image features themselves without additional context. The ‘style’ of an artwork here means the characteristic visual elements, techniques, methods, and themes that encapsulate the way the artwork looks to an observer. These styles are often connected with historical periods or cultural movements, some examples being Renaissance, Impressionism and Abstract Art. Artistic style manifests itself in many different aspects of an image and spotting it is not an easy task even for humans, unless they are specialized art curators. Unlike other image classification tasks such as object recognition that can rely on more clearly identifiable features, artistic style has no definitive identifiers. The task then becomes to find a solution that can capture multiple relationships within the same image and distinguish between fine-grained differences, for what really is the difference between Early-Renaissance and High-Renaissance? We believe that deep neural networks are well-suited to this task, as they have been shown to be able to learn complex relationships from data efficiently and with good results. As pointed out in our reference paper [1], this effectiveness does come with a slight tradeoff regarding transparency, as it is hard to pinpoint exactly how the models relate features to each other. An alternative approach, like basing classification off pre-computed features of some sort, might give better clarity as to what is going on ‘underneath the surface’. Still, we believe that the empirical power of deep learning outweighs these limitations.

For our solution we created two modern implementations of the art style classification neural network: one transformer-based implementation and one convolution neural network implementation. Then, by comparing the results of each implementation, we can evaluate whether one approach more effectively determines styles from images than the other. Furthermore, we will inspect if these newer methods yield better results than our reference paper. Both network types will be pre-trained on image classification to maximize performance gains from transfer learning and to preserve high-level representations of image content, which can also contribute to evaluations of a style label.

3.1 CNN Solution: EfficientNetV2

Since the Lecoutre paper in 2017 on art style classification using ResNet and AlexNet, significant development has occurred in the field of CNNs. In their 2019 paper “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”, authors Tan and Le introduced a new type of CNN. EfficientNet has shown promise in object detection and immediately achieved state of the art accuracy on common datasets such as ImageNet, CIFAR and Flowers. The aim of the architecture is two-fold: to increase efficiency within the convolutional layers, as working with image data can be very slow in deep neural networks, and to improve performance by introducing a new way of ‘scaling up’ the network.

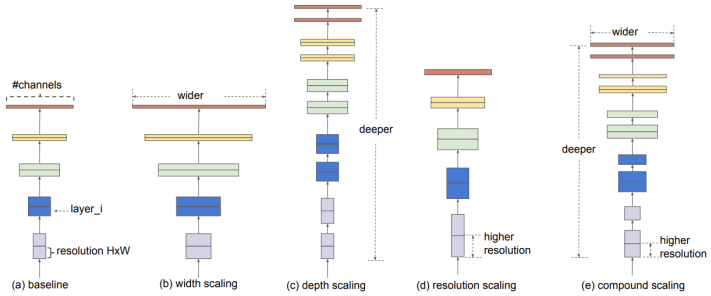

Figure 1. Isolated scaling vs. compound scaling [7]

Commonly, upscaling the dimensions of a CNN, meaning the depth (number of layers), width (number of channels per layer) and image resolution (image size), is believed to improve performance. Instead of simply scaling one 'dimension' at a time, for example by increasing the number of layers similarly to that done in ResNets, the EfficientNet is designed using a compound scaling algorithm where all 3 dimensions of the network are scaled jointly according to a fixed ratio. A scaling coefficient ϕ is defined to represent the amount of extra resources available and scaling constants α, β, γ to describe the optimal distribution between dimensions (Fig. 2). First ϕ is fixed to 1 and α, β, γ are found using a grid search. Then, α, β, γ are fixed and the network is scaled up by increasing the value of ϕ. The end result is an architecture which, according to the paper, optimally combines the size of the dimensions.

$$\begin{aligned} \text{depth: } & d = \alpha^\phi \\ \text{width: } & w = \beta^\phi \\ \text{resolution: } & r = \gamma^\phi \\ \text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ & \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned}$$

Figure 2. Definition of scaling constants and coefficient [7]

The layers themselves in EfficientNet are made up of so-called MBConvBlocks that use depthwise convolution instead of standard convolution. Instead of applying one kernel at a time jointly over all dimensions of a m-D input such as a RGB image (Fig. 3), depthwise convolution applies a

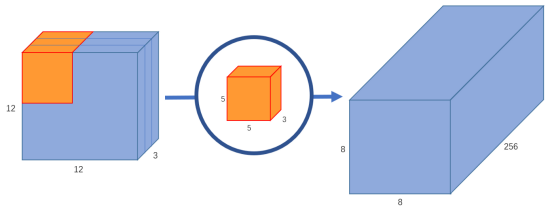

Figure 3. Feature extraction using standard convolution [9]

separate kernel to each channel in the input (Fig. 4) and then combines the channel feature maps with a pointwise convolution. On creating multiple feature maps between layers this reduces the computations needed greatly.

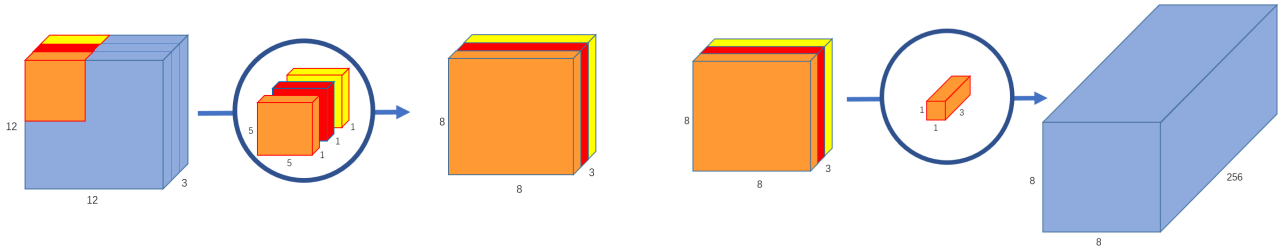

Figure 4. Depth wise convolution splits the feature extraction into two stages [9]

On a higher level, the MBConv blocks are ‘inverted bottleneck’ blocks: a sequence of depthwise convolutional layers where the layers at the beginning and end have fewer channels than the layers in the middle (narrow-wide-narrow). In the inverted structure combined with depthwise convoltuion reduces parameter count and increases efficiency (hence, Efficient-Net).

In 2021, Tan and Le released a new and improved version of EfficientNet, EfficientNetV2 with slight improvements to their architecture. They decrease the input image sizes used in training, replace some of the depthwise convolution with standard convolution on because “Depthwise convolutions have fewer parameters and FLOPs than regular convolutions, but they often cannot fully utilize modern accelerators,” [4] and add some restrictions to the dimension scaling to make it less uniform. In our solution, we use this newest EfficientNet version and apply it to the domain of art style classification with transfer learning.

3.2 Transformer Solution: Vision Transformers

Transformers, originally introduced in 2017 by Vaswani et al. in “Attention is All You Need”, have experienced tremendous success within natural language processing tasks and have quickly become the de-facto architecture for the domain. However, despite their success, the application of transformer-based architectures to computer vision tasks remained extremely limited, until recently. In 2021, Dosovitskiy et al. shook the world of computer vision after publishing “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale,” a paper detailing a groundbreaking new transformer-based model for image recognition tasks: the Visual Transformer (ViT). In their publication, Dosovitskiy’s team showed that their Vision Transformer was capable of not only matching, but outperforming, the majority of state-of-the-art CNN models available at the time.

From a high level, the ViT functions very similarly to the transformer-based models from the world of NLP: a sequence of embedded data containing positional and classification

information is passed through a series of encoders, some form of attention is calculated between the sequence elements, and the results of these calculations are passed to a fully-connected head for classification. However, the major difference is that the image data passed to the model has to be creatively transformed in order for the format to be compatible with the transformers and for attention calculations to be computationally feasible. Dosovitskiy et al. addressed this issue by transforming individual images into a sequence of 16x16 or 14x14 image patches. Each patch within the sequence is linearly-projected/flattened into a vector and receives a positional embedding to inform the transformer of its location in the true image relative to the other patches for parameter determination. The sequence is fed into the transformer with a learnable class embedding, and the transformer then functions exactly as in typical NLP tasks, calculating global attention between the vectorized image patches and using the output to classify the input according to the information provided in the class embedding.

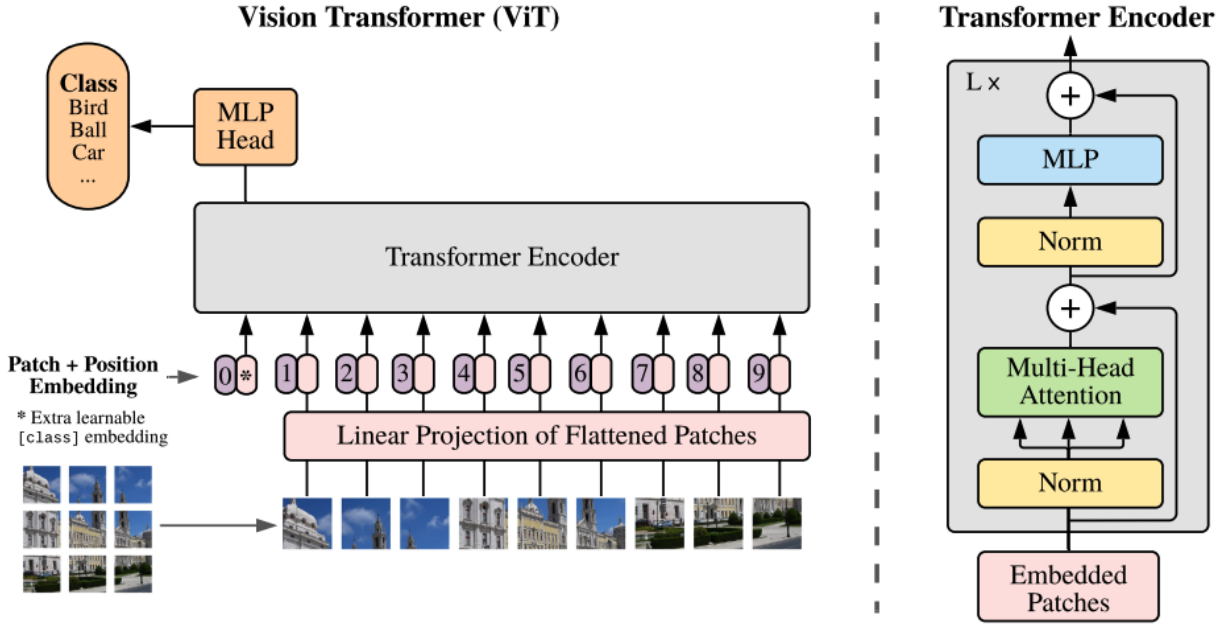

Figure 5. Visual Transformer Architecture

In our project, we wanted to take the capabilities of the ViT one step further and explore its performance on the task of artistic style recognition after fine-tuning. Could the transformer's new method of processing images reveal a latent ability to outperform CNNs in the realm of image style analysis?

4 Implementation

Our proposed solutions rely on importing models pre-trained for image classification and fine-tuning them for style classification. In this section, we will go over how we implemented

each model. To allow for a fair comparison of results, we ensured that both models were pre-trained on ImageNet21k, a large computer vision dataset of pictures and associated classes.

4.1 WikiArt Data

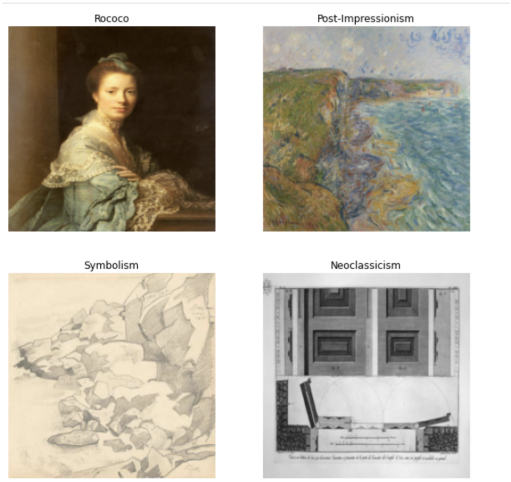

Figure 6. Examples of Images with respective labels from the WikiArt dataset

In order for our results to be comparable to the results produced in our primary reference paper by Lecoutre et. al. [1], we wanted to make sure that we found and used the exact same WikiArt dataset, which was originally gathered by Tan et. al. [2] in 2016. We managed to locate the original GitHub repository created by Lecoutre et. al. for the paper and the associated dataset used for the study, previously split into separate training, validation, and testing groups. The data is open-sourced and available for download. In its entirety, the data consists of 82,133 different images across 25 different classes representing art style. Curiously, this image count is 21 images short of the image count reported in our reference paper, a difference that may be a result of files having been removed from the dataset some time after publication of the paper. However, such a small difference is unlikely to have a significant impact on our results.

We also decided to add augmented images to supplement the original dataset, eventually ending up with 245,450 individual images, split evenly across each of the 25 art styles. This decision was based on two factors. First, we wanted to balance the amount of images present in each class. The original dataset was heavily skewed, with significantly more paintings coming from impressionism and realism relative to every other art style. Second, given our application-focused objective, we needed to make the model more resilient to the types of imperfections that might be seen from user-submitted images. Such imperfections include, but are not limited to, pixelation due to compression, images out of focus, motion blur, poor cropping, rotation, changes in contrast, and noise related to imperfect sensors. A random selection of these augmentations, as well as the standard horizontal and vertical flips could be applied to selected images.

Initially, we created a new dataset which included all images from the original WikiArt, plus an added number of augmented images so that each class had exactly the same number. The issue with this approach was that it left the impressionist paintings entirely unmodified. After performing initial tests, it was clear that the model was learning to classify every un-augmented image as impressionist. To alleviate this tendency, we added in a variable amount of augmentation to the impressionist images.

4.2 Vision Transformers Implementation

For our implementation of the Vision Transformer, we decided to conduct initial experimentation using the two most performant models from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale": the ViT-L/16 and ViT-H/14. PyTorch versions of these models pre-trained on ImageNet21k are available for import via the PyTorch Image Models (timm) deep-learning library. The library also provides some functions for appropriately applying basic image transformations needed prior to ingestion by the ViTs.

The primary difference between ViT-L/16 and ViT-H/14 is size of the transformer component: ViT-L/16 ("ViT Large") consists of 24 layers with 307M parameters and uses 16x16 image patches, whereas ViT-H/14 ("ViT Huge") consists of 32 layers with 632M parameters and uses 14x14 image patches. Our implementation began with quick performance comparisons of the two models using a small subset of the WikiArt dataset, and we found that the ViT-H to be more performant than the ViT-L/16. Dosovitskiy et al. [5] observed similar results in their experiments and also noticed that the true performance advantage of the ViT-H is often revealed with massive datasets. For these reasons, we chose to import the ViT-H model as our pretrained ViT representative. All ViT functionality and testing is contained within a Jupyter notebook.

4.3 EfficientV2 Implementation

The EfficientV2 reference paper links to an open-source Github repository with available versions of their EfficientNetV2 model pre-trained on ImageNet21k. Following the repository's instructions, the model can be imported and loaded into a python-based script. We decided to use Python Jupyter Notebooks on the BU SCC. The model is set up to be used with the Tensorflow Machine Learning framework, which we used for our EfficientNet experiments. As a consequence we worked with built-in methods and classes from Tensorflow to construct our model and as building blocks for our training and testing process.

Numerous versions (or 'checkpoints') of the pretrained weights are available from different stages of the EfficienNetV2 development pipeline. We chose 'efficientnetv2-l', the largest and most robust checkpoint. On importing the EfficientNet model, a model instance can be created with only a few lines of code (Fig n).

```
import effnetv2_model

base_model = effnetv2_model.get_model('efficientnetv2-b0', include_top=False)
```

Figure 7. Importing a pre-trained EfficientNetV2 model

The ‘include_top = False argument’ gets rid of the prediction layer from pre-training in order to be replaced with a custom one. For classifying art styles, we made EfficientNet the first sub-module in a tensorflow.keras.sequential model and directed its output to a linear layer with output size 25 to match the classes of the WikiArt data. A softmax activation function was then applied to the output to get the probabilities of each class, with the highest probability corresponding to the predicted label.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Figure 8. Softmax activation

4.4 Code Resources

For convenience, we have summarized the main code resources here.

- EfficientV2 model: https://github.com/google/automl/tree/master/efficientnetv2
- Tensorflow:
  - https://www.tensorflow.org/api_docs/python/tf/keras/Model
  - https://www.tensorflow.org/api_docs/python/tf/keras/Sequential
- PyTorch Image Models (timm): https://github.com/rwightman/pytorch-image-models
- vit-explain: https://github.com/jacobgil/vit-explain

5 Experimental Results

From a high level, we organized our experimental approach for evaluating both models into three phases. Our first phase was considered our benchmarking phase, in which we trained and tested both models exclusively on non-augmented data. The purpose of this phase was purely to establish baseline performances for both models. The primary focus of the second phase was to try to isolate and study the effects of data augmentation on our models’ abilities to learn artistic styles. For this phase, we trained both models on an augmented training dataset, but still conducted testing on the non-augmented test datasets. For our final phase, we wanted to test model performance on test sets containing a better representation of images that we might expect users to upload in a real-world environment. To this end, we concluded with training and testing both models on augmented training and test datasets.

A key part of our experimentation was conducting validation testing to allow us to determine the conditions and parameters that would produce the most performant models for testing. Validation testing for both models started with hyperparameter tuning, which consisted of an exhaustive grid search over a defined set of learning rates and batch sizes. To minimize the

time required to perform the grid search, we tested each parameter pairing by fine-tuning only the head layers of each model and using the resulting models to generate the respective trial accuracies. An example of the values tested for the ViT and the results of each trial can be seen in Figure 9.

Learning Rate \ Batch Size	32	64	128	256	512
0.003	20.02	18.88	17.01	16.63	16.32
0.01	20.57	20.30	19.63	17.28	15.98
0.03	19.37	22.77	23.31	20.36	16.54
0.06	18.75	24.19	24.45	23.54	19.08

Figure 9. Results of ViT hyperparameter tuning resulted in an optimal learning rate of 0.06 and an optimal batch size of 128.

After hyperparameter tuning, mechanisms for tracking validation loss after each training epoch were implemented for the purposes of retaining the most performant models produced during isolated training loops and triggering early stopping when necessary. After each epoch, model performance on the validation dataset was evaluated and compared to the lowest value recorded up to that point in the training. If the new validation loss was lower than the previous lowest validation loss, the new version of the models was saved and retained as the best model until either a better version appeared in a later epoch or the training ended. For early stopping, we monitored the loss after each epoch and ended training if the loss increased for three epochs in a row. This optimization saved us training time and allowed us to find our best validated model more quickly by stopping the training process once the model began to overfit our training data.

The final portion of our validation testing focused on determining the optimal number of layers to fine-tune versus freeze in the models to maximize performance, as suggested by Lecoutre et al. To do so, we varied the amount of layers that were frozen (parameters retained their pretrained weights) versus fine-tuned (parameters were trained using the WikiArt data) in our imported model, trained the given model, and evaluated its top-1, top-3, and top-5 performance on the validation dataset. Top-k accuracy is a metric where a prediction is marked correct if any of the top-k probabilities in the model output belong to the correct label. Top-1 is thus ‘traditional’ accuracy, but top-3 and top-5 give additional insights into how well the model has learned. From these experiments, we found that the ViT achieved the best performance with 19/32 layers fine-tuned, whereas the performance of EfficientNetV2 peaked with 36/80 layers fine-tuned. With the ViT, we were unable to determine an exact peak at which the layer freezing maximized performance, as the accuracy continuously increased with increasing amounts of unfrozen layers, but computational and memory issues due to the number of unfrozen parameters made further experimentation untenable. Table 1 shows the results obtained by these models on

the validation set during the layer freezing tests, alongside the results from our reference paper on the same validation set.

Validation Accuracy Results				
Model	Layers Unfrozen	Top-1 Accuracy	Top-3 Accuracy	Top-5 Accuracy
ResNet50 (Paper)	20/100	61.1%	86.3%	93.6%
EfficinetNetV2	**36/80**	**66.94%**	**89.57%**	**95.31%**
ViT	19/32	58.82%	86.27%	94.05%

Table 1. Validation Accuracy Results

Having determined the optimal conditions for the ViT and EfficientNetV2, we finally tested the models on the non-augmented test set to compare the results with those achieved by Lecoutre et. al. in our reference paper. Each of these models was trained on an augmented dataset and tested on the same test dataset, allowing for an appropriate comparison of results. The results can be seen in Table 2 and show that EfficientNetV2 successfully surpassed those obtained in our reference paper, while the ViT came close but was ultimately unable to achieve better.

Testing Accuracy Results: Non-Augmented Test Set				
Model	Layers Unfrozen	Top-1 Accuracy	Top-3 Accuracy	Top-5 Accuracy
ResNet50 (Paper)	20/100	62.8%	86.0%	93.3%
EfficinetNetV2	**36/80**	**66.7%**	**89.15%**	**95.13%**
ViT	19/32	58.48%	86.34%	93.44%

Table 2. Testing Accuracy Results on Non-Augmented Test Set

To conclude our three phases, we tested our models on the augmented test set to gain a more realistic understanding of how the models might be expected to perform on non-ideal, user-uploaded photos. The results from these tests are summarized in Table 3.

Testing Accuracy Results: Augmented Test Set				
Model	Layers Unfrozen	Top-1 Accuracy	Top-3 Accuracy	Top-5 Accuracy
EfficinetNetV2	**36/80**	**62.92%**	**86.54%**	**93.33%**
ViT	19/32	54.97%	83.00%	90.4%

Table 3. Testing Accuracy Results on Augmented Test Set

To visualize the similarities that might be present between certain art styles, we employed principal component analysis (PCA) followed by t-Distributed Stochastic Neighbor Embedding (t-SNE). The goal was to try to reduce the extremely high-dimensional image data into a lower-dimensional representation. t-SNE accomplishes this by minimizing the Kullback-Leibler divergence between the points in the original space, and those present in the reduced space. In the high dimensional space, this requires calculating the "similarity" between all points, which is equivalent to the conditional probability of point x_j being a "neighbor" of x_i, if neighbors are selected in proportion to the probability density under a Gaussian distribution centered at x_i. In this way, nearby points to x_i have a conditional probability of being neighbors that is quite high. After calculating each of these conditional probabilities, the data points are then randomly spread throughout the reduced space (in our case, a 2-dimensional space). Finally, using stochastic gradient descent, each point in the reduced space has its location updated based on minimizing the difference between probability distributions in the high-dimensional space and the updated 2-dimensional space. For computational efficiency, and to aid in visualization, the distributions in the 2D space are modeled using the t-distribution with a single degree of freedom. Since the t-distribution is long tailed, this method prevents the points in the reduced space from stacking on top of each other [8].

$$\mathrm{KL}\left(P \parallel Q\right) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / 2\sigma_i^2)} \quad , \quad p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}$$

Fig. 10 Kullback-Leibler Divergence, and the Joint Probability Distribution of Points x_i and x_j in Original Dimension(p) and corresponding points y_i and y_j in Reduced Dimension (q)

As the training of our models progressed, we extracted the vector embeddings from the final layer of both the ViT and EfficientNetV2 (which themselves are lower-dimensional representations of the images), performed PCA to reduce the dimensionality to a suitable size for t-SNE, and plotted the results for the top 4 classes by recall. Moving from left to right in figure 11, we can see the results for the ViT model with 1, 8, and 19 trainable layers, respectively. This

provided visual confirmation that the ViT model was learning to differentiate between various art styles.

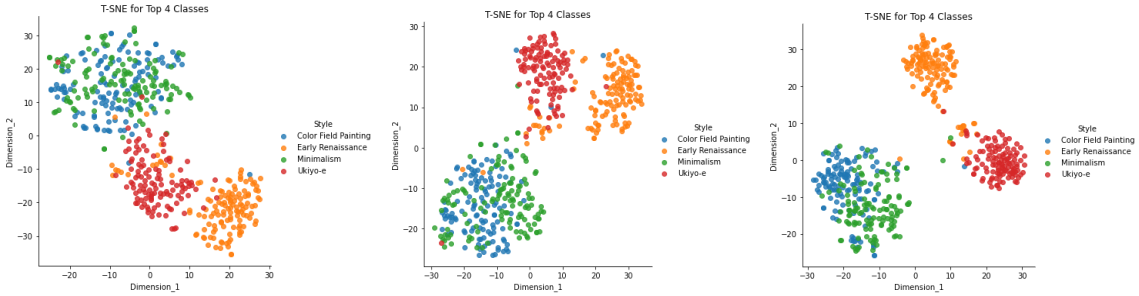

Fig. 11 t-SNE Scatter Plots of ViT Testing Data After PCA modification

The visual contrast between initial trained models and the final trained model was even more stark when looking at the plots produced using the final layer of the EfficientNetV2. While starting with a near identical plot to the ViT for a single retrained layer, our most performant model, which had 36 retrained layers, showed clear delineation between classes, even when they were quite similar (Early Renaissance vs Northern Renaissance). Based purely on these visuals, one would expect that the EfficientNetV2 was the most performant model, and our results confirmed that hypothesis.

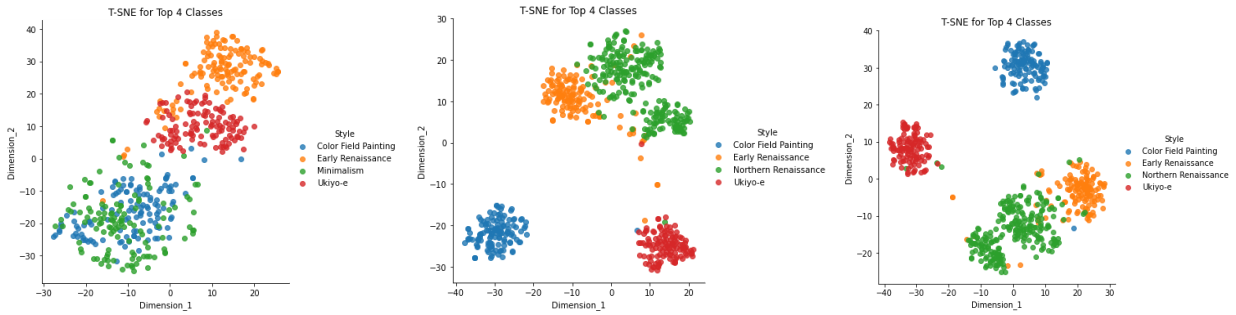

Fig. 12 t-SNE Scatter Plots of EfficinetNetV2 Testing Data After PCA modification

The confusion matrices for both models also do a good job of summarizing the general trends we observed. First, the majority of misclassifications for both models came when trying to predict art styles which are similar to impressionism, particularly post-impressionism and realism. Unsurprisingly, both models were nearly perfect in predicting ukiyo-e art. Though subjective, when compared to the other 24 art styles we examined, ukiyo-e is clearly the most unique.

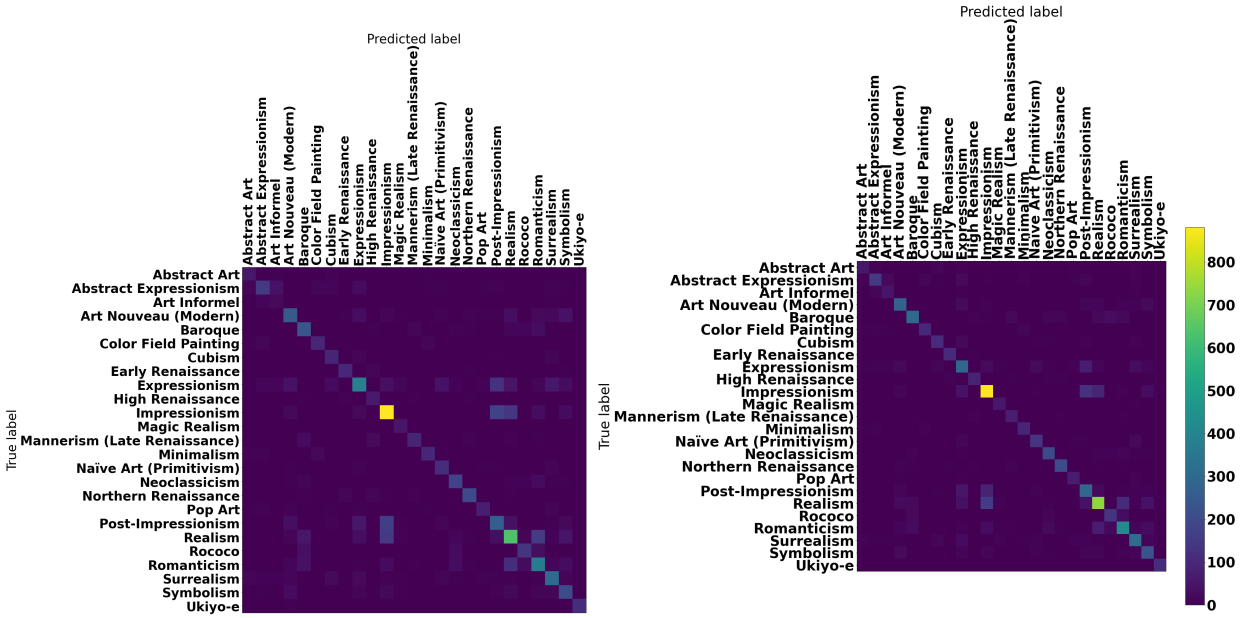

Fig. 13 Confusion Matrices for ViT (left) and EfficientNetV2 (right)

Conclusion

Overall, both models were able to yield impressive results compared to the art classification capabilities of the average person, but in the end EfficientNetV2 proved to be the all-around better model for art style classification when compared to the Visual Transformer. Not only did the EfficientNetV2 score better top-k accuracies across all tests, but it was also a much quicker model to train (the ViT with 19/37 layers frozen took about twice as long as the slowest EfficientNet model) and was more portable than the ViT-H.

We've also seen that the task of art style classification remains a challenging one, particularly when more labels are considered. While the performance of either model was far from inadequate, there exists definitive room for improvement in classification accuracy, especially when considering the use of such models in a user-facing application. However, the improvement on the results obtained by Lecoutre et al in 2017 sheds light on the promising progress made towards the enhancement of CNNs over time, making the idea of an even more effective artistic style classifier in the near future a very realistic one.

Regarding further improvements, it would have been interesting to explore how the models may have performed on more modern artistic styles, as the WikiArt dataset primarily contained what most would consider more "classical" styles. Yet, modern styles of painting and newer art forms such as installation art and light art have become increasingly popular nowadays, so any user-facing model to be used in artistic style curation should be able to account for such artwork. Additionally, both models would benefit from further work on improving classification accuracies when the photograph of the artwork contains minor obstructions or noise surrounding the piece. In testing, we noticed that the ViT in particular would incorrectly classify any photographs in which people crowded the area near the artwork as surrealism, with a

high level of confidence. Improvements to this type of behavior would be pivotal for successful deployment in an application, as user photographs most often contain noise around the point of interest.

References

[1] A. Lecoutre, B. Negrevergne, and F. Yger, “Recognizing Art Style Automatically with deep learning”, *Proceedings of Machine Learning Research, PMLR*, 77: pp.327 - 342, ffhal-02004781f, 2017.

[2] W. Tan, C. Chan, H. Aguirre, and K. Tanaka, “Ceci n’est pas une pipe, A deep convolutional network for fine-art paintings classification”, *International Conference on Image Processing (ICIP)*, pages 3703–3707, 2016.

[3] K. Salman, et al. “Transformers in Vision: A Survey.” ACM Computing Surveys, Jan. 2022, p. 3505244. arXiv.org, https://doi.org/10.1145/3505244.

[4] M. Tan and Q. Le “EfficientNetV2: Smaller Models and Faster Training”, Computer Vision and Pattern Recognition, Apr. 2021, arXiv:2104.00298, 2021.

[5] A. Dosovitskiy, et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, Computer Vision and Pattern Recognition, arXiv:2010.11929v2, 2021.

[6] Vaswani, Ashish, et al. “Attention Is All You Need.” ArXiv:1706.03762 [Cs], Dec. 2017. arXiv.org, http://arxiv.org/abs/1706.03762.

[7] M. Tan and Q. Le “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”, Computer Vision and Pattern Recognition, Sep. 2020, arXiv:1905.11946, 2020.

[8] van der Maaten, Laurens. “*Visualizing Data Using T-Sne*” , The Journal of Machine Learning Research, pages 2580 - 2605, 2008

[9] Wang, Chi-Feng “A Basic Introduction to Separable Convolutions”, Towards Data Science, 2018. Available at: https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728