# [Team 29] ProjF4 [Classifying Artwork by Artists using Neural Networks]

Radek Pudelko (rmpudelk@ncsu.edu), Melvin Orichi Socana (msorichi@ncsu.edu)

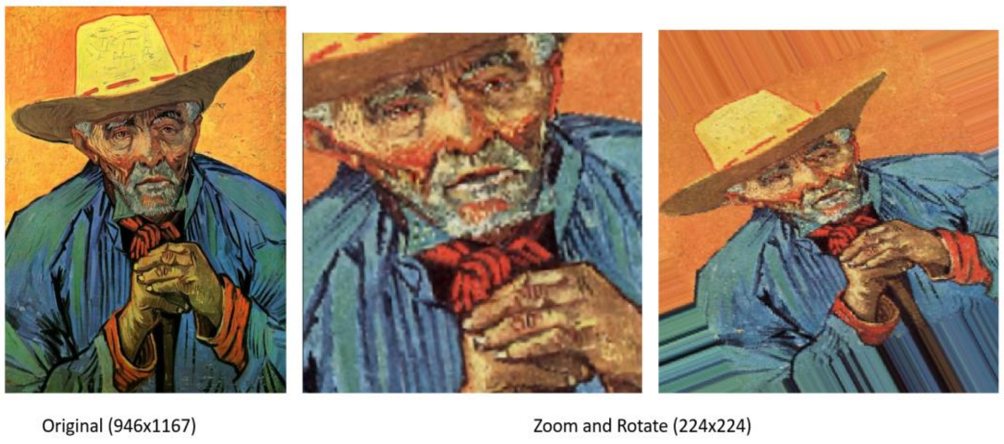


Figure 1 Data Augmentations Sample (Vincent Van Gogh)

## INTRODUCTION

The goal of our project was to build a robust image classifier using neural networks with the goal of classifying artwork by the artist who made it. The motivation behind this was that we wanted to apply what we had learned in class to and to see what was possible for what we thought would be a difficult dataset. It can be very difficult to identify the artist who created a painting that was hidden from the world for decades. Identification of art can be a feat too difficult for humans; however, with the algorithms behind neural networks a deep learning model may have the ability to pick up on the subtle features and intricacies of art in order to identify the artist. There is large variability in artwork, even for those within the same artistic movement and for the same artist. Not only do the colors and geometric patterns vary from piece to piece but also the size of artwork and orientation. Because of the sizes and varying aspect ratios, there will be some degree of distortion present after augmenting the data for training, which may limit the quality of the features the models can learn. Additional challenges include the computational cost as these images are large with an average size of 510x563. This impacts how many images we can use for training as well as the size of batches which imparts the amount of time it takes to train a model and therefore run experiments to improve upon the baseline.

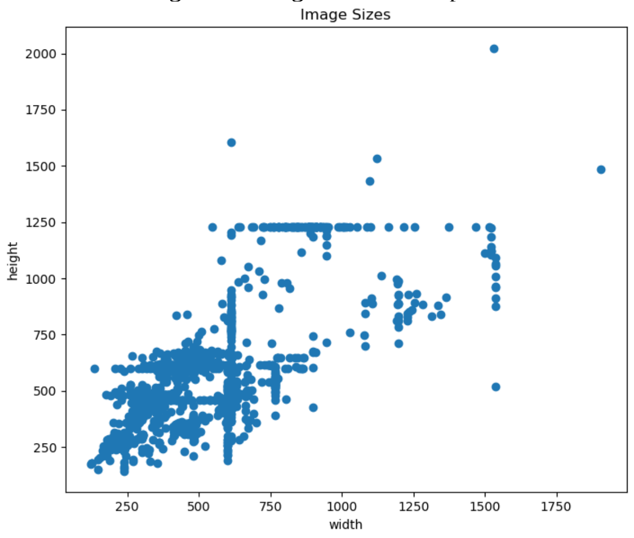
Our team made use of the TensorFlow library in Python for this project [1].

## DATA

We used a Kaggle dataset called “Best Artwork of All Time” [2]. This dataset consists of 8683 artworks from 50 different artists saved as .png files. We extracted 2000 works corresponding to the top 10 artists with the most artwork for our model. This was done because initial attempts to train and experiment with the whole dataset were computationally expensive. With this selection and a batch size of 32, it takes roughly 1 min 30 s to train and validate a single epoch on Google Collab.

Most images were below the size of 600x600 as seen in figure 2, and the aspect ratios were ~.77 and ~1.3 for portrait and landscape images with most images being portrait.

Figure 2. Image Size Scatterplot



We also made use of data augmentation to assist in training. All images' RGB values were rescaled so that their values ranged from 0-1 and the images were resized to 224x224. This size was selected as it was the default size for the ResNet50 model we would use for transfer learning.

Data was split 80/20 train/validation. With resizing, there is some added distortion due to the change in the aspect ratio and loss in information due to image shrinking.

Images from the train dataset were loaded through a generator (ImageDataGenerator) which modified images rotating within a range of 30 degrees from the original as well as providing .5 zoom (.5-1.5 of the original zoom). These augmentations are intuitive for this dataset as viewers may move closer/further to the art to get a better view as well as look from different angles. Additionally, paintings that were flipped horizontally and vertically were also used to augment training data. These modifications varied each epoch. Figure 1 shows an original source image and 2 augmented versions. The loss of pixel information is appart in the first augmentation where resize and zoom reveal a more grainy image, while in the second image additional information is added in order to account for missing information when zooming out outside the main image.

The validation images were not modified except to rescale and resize the images. This was done so that performance could be compared across different models.

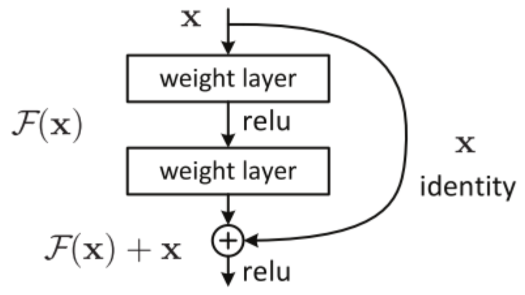
## METHODOLOGY

In order to build a robust model, we made use of transfer learning with the ResNet50 model, which was trained on the 1000 class ImageNet dataset [3]. ResNet50 is a residual neural network designed to avoid common problems associated with deep neural networks such as the degradation of training accuracy.

ResNet50 was chosen for transfer learning because of its feature extraction capabilities. We removed the output layer from ResNet50 and replaced it with a fully connected (Dense) (FC) layer with 10 nodes for our 10 artists with a softmax activation. Between ResNet50 and our output layer we included layers to meet the fixed size requirement for fc layers. We followed 2 approaches. In the first model, we used an average global pooling layer (GlobalAveragePooling2D), which acts as a replacement for fully-connected layers when using data in variable dimensions. In the second model, we used a spatial pyramid pooling (SpatialPyramidPooling2D) and a flatten (Flatten) layer. We refer to these models as GAP and SPP, respectively. The pyramid layer was chosen because it would allow the model to function with non fixed size images and therefore more interesting architectures. The pyramid layer works by using pooling windows to convert a variable size input to a fixed size output. A number of pooling windows can be defined, which will then pool a proportionally to the input. For instance, a [1,1] pool is effectively the same as global max pooling, [2,2] will look at 4 windows of the input [3,3] will look at 9, etc. [4].

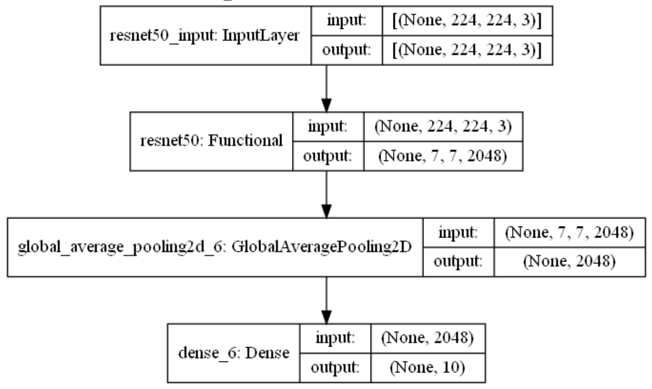
Figure 4 shows the GAP model and 5 shows the SPP model. The SPP model uses 3 pools of sizes [[1,1], [2,2],[3,3]], which results in an output size of 1 + 4 + 9 = 14. These two models are our baselines. GAP slightly outperforms SPP with the F1 score metric; because of this, we consider GAP to be our main baseline model and will primarily show figures for that model.

Figure 3: Residual Network Building Block



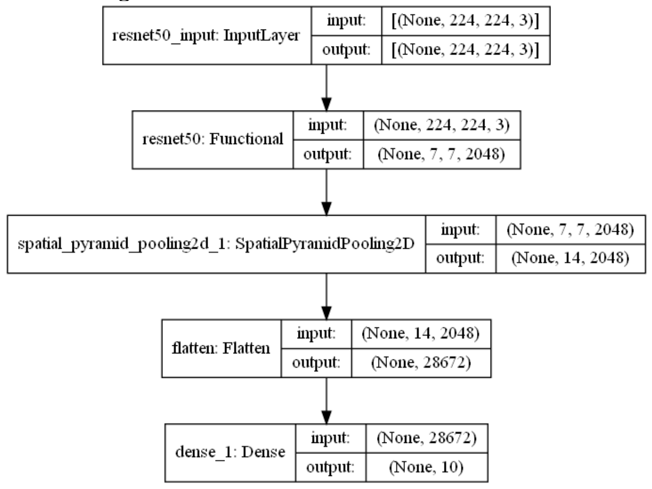
x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 4. GAP Baseline



resnet50_input: InputLayer | input: [(None, 224, 224, 3)] | output: [(None, 224, 224, 3)]
resnet50: Functional | input: (None, 224, 224, 3) | output: (None, 7, 7, 2048)
global_average_pooling2d_6: GlobalAveragePooling2D | input: (None, 7, 7, 2048) | output: (None, 2048)
dense_6: Dense | input: (None, 2048) | output: (None, 10)

Figure 5. SPP Baseline Model Architecture



resnet50_input: InputLayer | input: [(None, 224, 224, 3)] | output: [(None, 224, 224, 3)]
resnet50: Functional | input: (None, 224, 224, 3) | output: (None, 7, 7, 2048)
spatial_pyramid_pooling2d_1: SpatialPyramidPooling2D | input: (None, 7, 7, 2048) | output: (None, 14, 2048)
flatten: Flatten | input: (None, 14, 2048) | output: (None, 28672)
dense_1: Dense | input: (None, 28672) | output: (None, 10)

These models were trained along with early stopping (EarlyStopping) and learning rate reduction (ReduceLROnPlateau) callbacks. These callbacks monitored the f1 score and had a patience value of 4 and 10 respectively. The reduced learning rate callback multiplied the learning rate by a factor of .1 if the F1 score failed to improve within 4 epochs. This reduced learning rate allowed for fine-tuning of the model and prevented divergence from the maximum possible F1 score. Additionally, the Adam optimizer was used with an initial learning rate of .0001 and weights were initialized to the ImageNet weights.

We used the f1 score as our primary metric as there is significant imbalance in the total dataset. In order to improve upon our baselines, we defined a block consisting of a fc, dropout (Dropout) and batch normalization (BatchNormalization) layers. Dropout and batch normalization were added to help generalize our models, as the models overfit to the train data.

We placed two of these blocks right before the output fc layer. Dropout was set to .5 and we defined a search space for the 2 fc layers as 128:64:512 (128, 192, …, 512) and 32:32:256, respectively. This corresponds to 42 combinations to try per baseline model. With these additional layers, the goal was to be able to achieve better generalization and use of the extracted features from the ResNet50 model.

These models could be trained on Google Collab, but were trained locally using a NVIDIA GeForce RTX 3070 gpu, which improved training time to ~30s per epoch. The model with the full dataset was trained on Google Collab with a Tesla P100 GPU. All combinations were tried to find the best model.

After the first round of training was finished on the model, the ResNet50 portion of the baseline was frozen to incorporate transfer learning [5]. Since the initial baseline was trained on data augmented with rotation and zooms, the subsequent model would be trained on horizontal and vertical flips of the data. Freezing the entire ResNet layer allowed for additional training without destroying any of the information obtained from previous rounds. An additional 40 epochs of training were completed on the Dense and Batch Normalization layers. Lastly, after completion of the round of transfer learning training, the ResNet layer was unfrozen and trained with an incredibly low learning rate for fine-tuning of the model. This would allow for small, meaningful improvements on the final model by adapting pre-trained features to the newly flipped data.

## EVALUATION

Table 1 shows the top 3 models per baseline and along with evaluation metrics. The table is organized with the GAP models first ordered as baseline followed by the top 3 models. Dense 1 and 2 refer to the number of neurons in the added blocks in each model. Top 1 and 2 refer to the accuracy score.

| Model | Dense 1 | Dense 2 | Top 1 | Top 2 | F1 |
|---|---|---|---|---|---|
| GAP | Base | Base | 0.863 | 0.943 | 0.855 |
| 1 | 384 | 128 | 0.868 | 0.94 | 0.877 |
| 2 | 448 | 192 | 0.863 | 0.933 | 0.870 |
| 3 | 448 | 160 | 0.873 | 0.938 | 0.869 |
| SPP | Base | Base | 0.832 | 0.935 | 0.844 |
| 1 | 320 | 64 | 0.865 | 0.93 | 0.864 |
| 2 | 192 | 224 | 0.840 | 0.923 | 0.859 |
| 3 | 128 | 160 | 0.837 | 0.925 | 0.858 |

Table 1. Top Models

Our primary evaluation metric is the f1 score to which all models were trained to improve on. Improvements of about 2% were seen for both the GAP and SPP models in the f1 score. No significant improvements were seen in the top 1 and 2 accuracy scores. The overall best model was the GAP model with 384 and 128 neurons, with a f1 score of .877.

Figure 6&7 GAP Baseline F1 Score

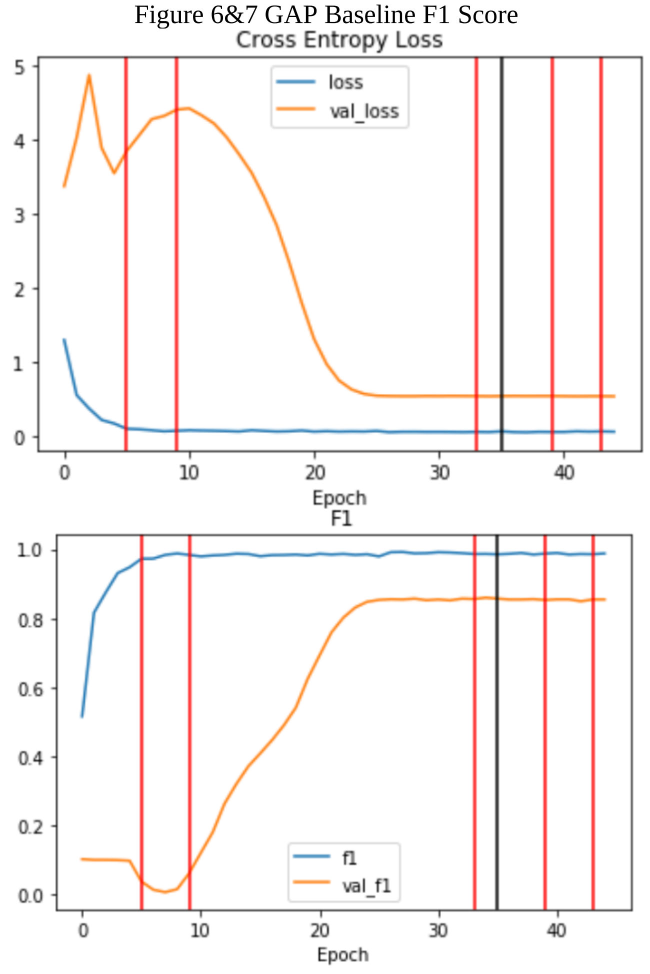
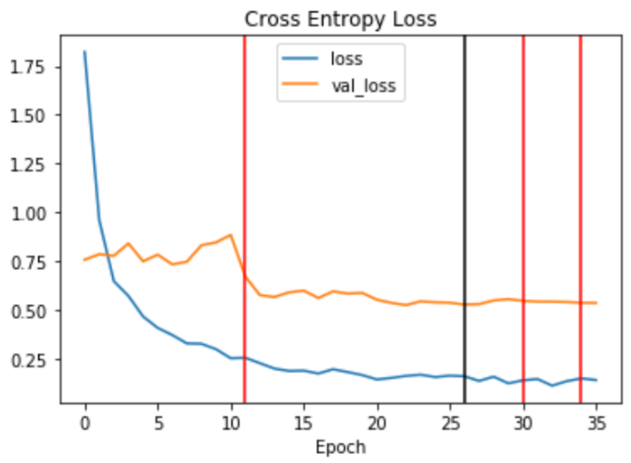
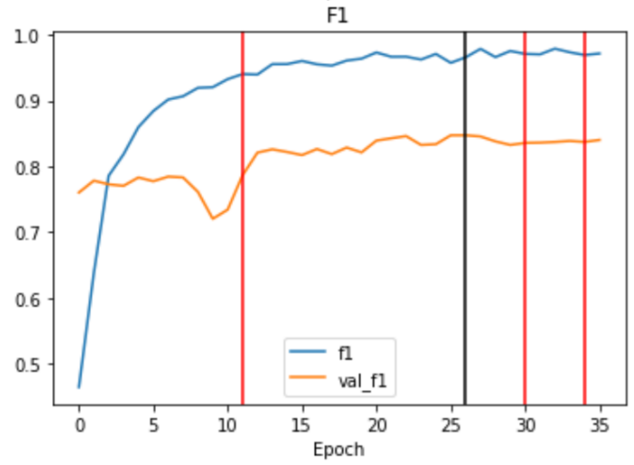


Figure 8&9 GAP (384x128) Loss and F1 Score





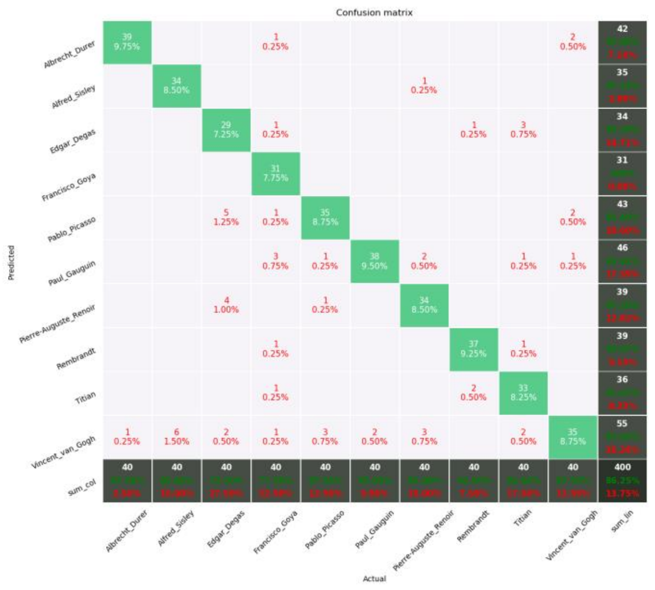
Figures 6 and 7 plot the loss and F1 score v. epochs for the GAP baseline model while 8 and 9 plot for the top GAP model. The red vertical lines indicate when the learning rate was reduced while the black line indicates the best epoch. The models with the additional dense layers were loaded with the baseline weights in order to reduce training time.

Interestingly, the greatest reduction in validation loss in the baseline model came with a reduction of the learning rate to 1E-6, further reductions only slightly improved performance. In the case of the improved model, improvement was only seen upto the 1E-5 learning rate.

An additional model was built based on the SPP baseline without the use of dense layers, but with training using images of multiple sizes. Research indicates that training with multiple sized images results in higher scores [4]. We trained using images of size 224, 260 and 296 (square), but did not see an improvement in any metric. An attempt was made to run 3 predictions for each image under this model, one for each size so that a final prediction was the one with the most votes from the 3 image sizes. This only resulted in lower scores and so is not shown here.

The confusion matrix for the best GAP model is shown below in figure 10.

Figure 10. GAP Baseline Confusion Matrix

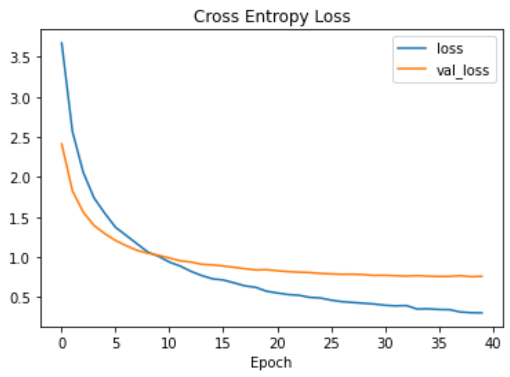


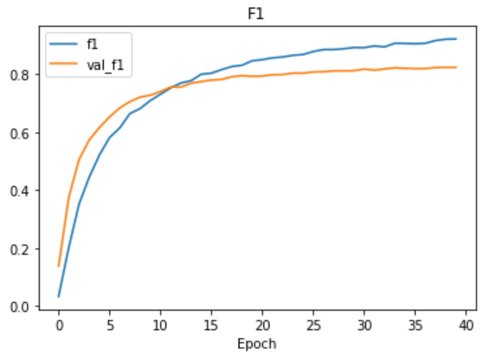
The greatest confusion was for Edgar Degas who was mostly commonly confused for Pablo Picasso and Pierre-Auguste Renoir. Edgar and Pierre are both French Impressionist Artists, which could explain the confusion as the styles are similar, but Pablo is a Spanish Cubist. Looking closer at the prediction errors, the confused works are similar in the colors used and subjects painted. This may explain the confusion between these artists.

Because the improvements are only a few percentage points, the confusion matrix and loss/F1 plots appear very similar to each other across all models.

A final full-model was trained and validated 50 artists' paintings with an 80/20 training and validation split using the GAP model and transfer learning. The model successfully avoided overfitting and was able to achieve a final F1 score of 0.82399 compared to the previous score of 0.78 achieved from the base model trained on rotated data.

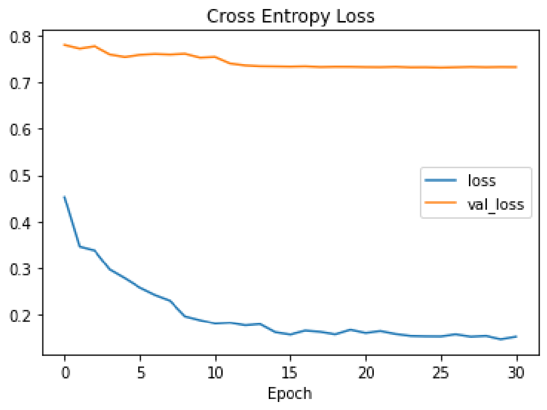
Figures 11&12 Loss & F1 Score of Full Dataset

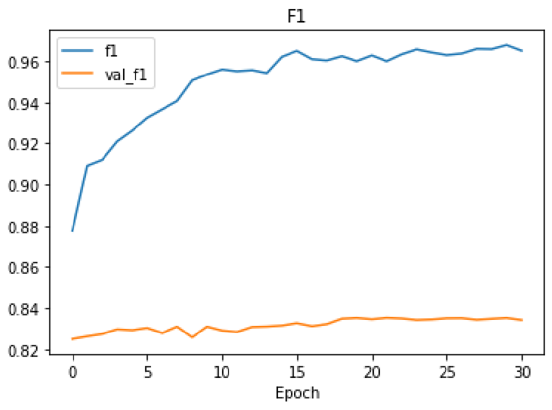




Lastly, the ResNet layers were unfrozen and fine-tuned with a learning rate of 1E-7 for 30 epochs which resulted in a 1.133% improvement with a final F1 score of 0.83532.

Figures 13&14 Full Unfrozen Model Loss & F1 Scores





## REFERENCES

[1] Abadi, M., "TensorFlow: A system for large-scale machine learning", 2016.

[2] "Best Artworks of all Time", Kaggle Dataset: www.kaggle.com/ikarus777/best-artworks-of-all-time

[3] He, K., Zhang, X., Ren, S., and Sun, J., "Deep Residual Learning for Image Recognition", 2015.

[4] He, K., Zhang, X., Ren, S., and Sun, J., "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition", 2015.

[5] Zhuang, F., "A Comprehensive Survey on Transfer Learning", 2019.