Defining Painters based on Artwork

Samarth Kamat & Tyler
Gavaletz

ABSTRACT

This project consists of using Residual Neural Networks to create a machine learning program that can identify the artists of given paintings.

Key Words

Residual Neural Networks, linear layers, artwork, Keras, ResNet

1. INTRODUCTION

After spending an entire summer in Europe visiting numerous museums and art exhibits, as well as seeing hundreds of art pieces, we have decided that it is only fitting that our final project should be relating to art pieces. This project could be of use with large collections of unidentified paintings. Such a collection exists in the MoMA database, and they are also working on a project with the goal of quickly identifying all their pieces, with equal or greater success than human curators.[4]

This tends to be considered a difficult and complex problem for a variety of reasons, mainly that artists tend to have a variety of styles and will change over time. Another factor is that many artists from a similar period will have styles or nuances that are repeated from artist to artist.

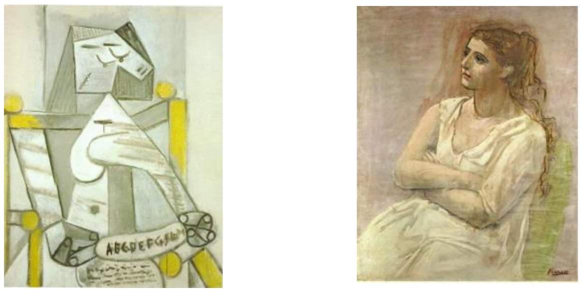


Figure 1 - two paintings by Picasso showing the vast differences in style over his career

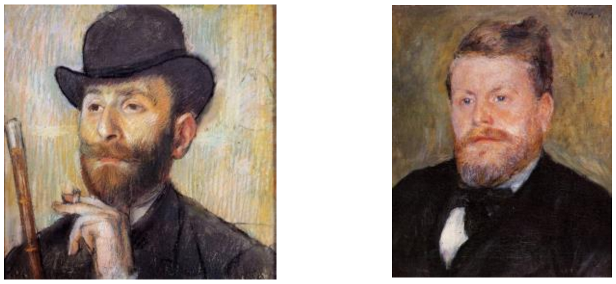


Figure 2 - paintings by Degas and Renoir, both impressionists from the 1840s who have similar painting features

Most of the past projects that have tackled this tend to focus on features specific to artwork, such as brushstrokes, more than general image features. This project also depends on an assumption that every artist has their own unique style of art that can be identified by certain patterns in the work. While this might not be a hundred percent true, the fact is that many artists do possess a unique enough style for a computer to be able to identify.

In the past, projects relating to art pieces have received some attention. In 2010, Alexander Blessing and Kai Wen from Stanford wrote a paper on their attempt of identifying artists based on art pieces and obtained fairly decent results with only focusing on seven artists.[1] In 2008, a group of researchers created a paper specifically focused on creating statistical models based on features that stand out in paintings.[2] While the contents of this paper might be slightly beyond the scope of this project, it is possible to use select pieces of information, and could be useful if this project got any attention in the future. In another Stanford study, Viswanathan utilized convolutional neural networks in tandem with ResNet systems on a similar dataset to identify artists.[3]

2. DATA

2.1 What is the Data

The dataset we are using for this project is called "*Best Artworks of All Time*" and is a dataset retrieved from Kaggle.[12] This set contains a collection of the top 50 most influential artists of all time with their art pieces. The artwork is collected from the artists' Wikipedia pages, and not all the artwork is digitally scanned pieces, which could result in a possible source of error for this project.

The contents of the data file are one csv file containing all the artists and some information about them, such as years of birth and death, number of paintings etc. Alongside the csv file is two folders of images. One folder is the raw image, one is a downsized version for quicker and easier image detection and data manipulation. It is a difference of the larger image folder being 1.61 GB, while the downsized version is 747 MB. We are using the resized images, because of the ease that comes with smaller images.

2.2 Data Manipulation

To start our data manipulation, we first need to figure out our painting limit that we want to work with. We only want the artists with over two hundred images to allow ample training for the machine. To find these artists, we took the information of all artists with over two hundred images and found the top eleven artists who were over the threshold. The artists we are focusing on here are Van Gogh, Degas, Picasso, Renoir, Durer, Gauguin, Goya, Rembrandt, Sisley, Titian, and Chagall. Next, we collected the images that belong to each of these artists and put them in folders with the name of the artist on it. This was difficult as there were over eight thousand images in the dataset, so we set up a bash script to do this for us.

After all the wanted images were separated into their folders, we had to get the images into our neural network to train it. For this, we used the Keras deep learning API of Tensorflow.[6] Learning how to use Keras modules and data loading was the most time-consuming part of this project. First, we attempted to use the Image Data Generation tool. This would allow us to pull images from directories then mutate it with certain transformations that can help the model find patterns easier. This is also supposed to create a model that is more robust against altered versions of original images.[5] We have not been able to spot a significant benefit with using this method. This could possibly be because of incorrect implementation, but it is something that we want to try and figure out in the future. Because it hasn't had a significant impact on our model, we have not been using it for every implementation of the neural network that has been tried. We also tried to use the base Tensorflow method of importing data, but that did not provide the results that we were looking for. Finally, we settled on using image_from_directory method in Keras to get the data that we wanted imported into our neural network.

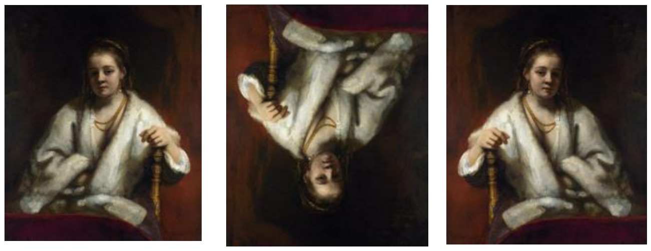


Figure 3 - Original Rembrandt(left) next to two possible mutations the Keras Image Data Generator can perform (flip horizontally and vertically)

Once we have the images imported into the program, we then resize all the images to 244 x 244. We chose this size because that is the dimensions that ResNet runs on, and we did not want to resize it before running it through ResNet. Then, we randomly separate the data into training and validation data, with a validation split of .2. Before the training begins, we give class weights to each artist in our dataset depending on the number of paintings they have in the set. This prevents unfair amounts of predictions for artists that have significantly more paintings in the set. For example, Van Gogh has 877 paintings, while the second most in the set, Degas, has 702 paintings, and the least number of paintings is Chagall with 239 paintings.

3. METHODOLOGY

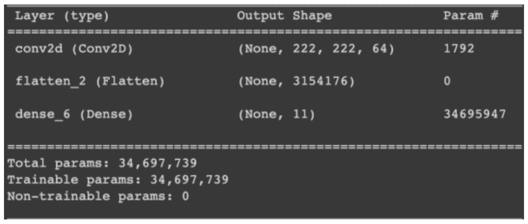
We have tried a few different implementations, namely: two convolutional layers with a linear layer, a single convolutional layer with a linear layer, ResNet50 with a linear layer, ResNet50 with training generator, and ResNet50 with three linear layers.

For our optimizer in this project, we chose to use the Adam optimizer because it is what we have used in past projects. We chose to use this stochastic gradient descent optimizer because we are used to it, and it is time and computationally efficient.[11] For our loss function we were originally using the binary cross entropy loss function that comes in the Keras module, but we were getting bad results until we realized that we needed to use the categorical cross entropy loss function because we are classifying our data into multiple categories, not just two.[6]

3.1 What Doesn't Work

Our first attempt at creating a neural network for this project was a two-level convolutional neural network followed by a linear layer. The training time complexity of this model was far too long, and we learned that having two layers for convolution was too much.

Our next neural network had the architecture of a single convolutional layer followed by a linear layer. This model trained well but suffered from overfitting. In this situation we chose to add dropout layers to reduce the amount of overfitting happening, but the dropout layers only slightly improved the accuracy. These dropout layers were later removed from our code to emphasize the model overfitting. From this model we learned that we need to find a different solution because the combination of linear and convolutional layers will result in overfitting.



```
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 222, 222, 64)      1792

 flatten_2 (Flatten)         (None, 3154176)           0

 dense_6 (Dense)             (None, 11)                34695947

=================================================================
Total params: 34,697,739
Trainable params: 34,697,739
Non-trainable params: 0
```

Figure 4 - System Architecture for the first working neural network that resulted in overfitting.

3.2 Residual Neural Network (ResNet)

The alternate solution that we have turned to for creating our neural networks layers is the Residual Neural Network, or ResNet. ResNet is a neural network that was created in 2015 with the goal of helping to identify images and to help other neural networks find more complex patterns that computers have had trouble spotting before this.

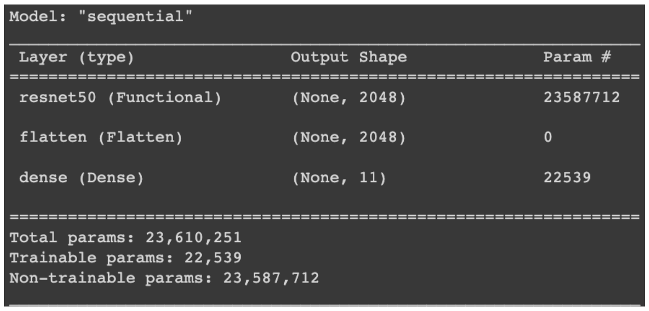
The original way that programmers would further develop deep neural networks is just by adding more layers to existing neural networks, which worked, only up to a certain extent. For some image detection training models adding more layers would just create overfitting, the same situation we are running into now.

The creators of ResNet found the solution to this issue of overfitting with extra layers by implementing skip connections in the ResNet. These skip connections take the weights found in the stacked convolutional layers and factors in information from the original input. This architecture significantly enhances neural network performances. Especially on neural networks that have higher numbers of layers, specifically with eliminating the issue of the vanishing gradient as there is another path for information to flow through that affects the weights.[7] The number of layers in the ResNet matters as well, ResNet with 18 layers compared to a normal neural network with 18 layers has similar error percentages, while ResNet-36 and a plain neural network with 36 layers will have greatly different error percentages.

A Stanford study [3] uses ResNet-18 in their model for identifying artists, but the error percentages for ResNet-18 aren't as impressive as the differences in error percentages for ResNet-50. Therefore, we chose to use ResNet-50 to try and make our model as accurate as possible. Because of the large number of layers in this ResNet, it takes a little longer to train. Therefore, to reduce on the time it takes to train, we chose to freeze some of the top-level layers that do the more basic image detection. This allows the model to go through the deeper layers of ResNet to get the more specific complex image detection that will affect the weights for our paintings specifically.

The first time we used ResNet-50, we didn't use the Keras image generator and saw immediate improvements in our accuracy when using the pretrained weights the fully connected ResNet provides. After we noticed this improvement, we attempted to use ResNet with the Keras Image Data Generator, which took much longer to train over, and the accuracy dropped by half. With more time we could explore where this drop comes from, but for now, we are not using the Image Data Generator.

After determining that using ResNet is the way to go, we created a neural network that uses ResNet followed by a linear layer. The addition of the linear layer helped to provide some additional information used in our classification, but the output of ResNet has 2048 perceptrons, then we downsize that to a linear layer that outputs eleven perceptrons. This improved our accuracy, but we hypothesize that narrowing down that much to the last perceptron is ineffective, so we tried to add more linear layers for more effective training.

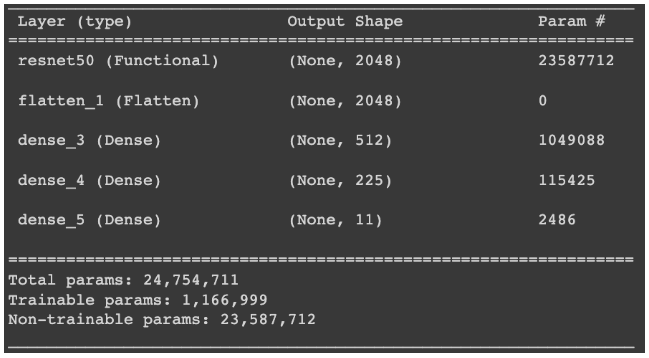


```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 2048)              23587712

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 11)                22539

=================================================================
Total params: 23,610,251
Trainable params: 22,539
Non-trainable params: 23,587,712
_________________________________________________________________
```

Figure 5 - System architecture for the first ResNet neural network

3.3 What Works

Our final solution was to start with the ResNet, followed by three layers of linear perceptrons, without the Keras Image Data Generator. This keeps the effective ResNet layer that caused a large increase in accuracy and improves upon it by having three linear layers to do additional painting specific classification. We chose this architecture because we found it has the best balance between running time and effective classification.



```
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 2048)              23587712

 flatten_1 (Flatten)         (None, 2048)              0

 dense_3 (Dense)             (None, 512)               1049088

 dense_4 (Dense)             (None, 225)               115425

 dense_5 (Dense)             (None, 11)                2486

=================================================================
Total params: 24,754,711
Trainable params: 1,166,999
Non-trainable params: 23,587,712
_________________________________________________________________
```

Figure 6 - System Architecture for the final ResNet with three linear layers

4. FINDINGS

The first neural network we created had two convolutional layers followed by a linear layer. This network took too long to train as it took about 37 minutes to train one epoch, and after a single epoch it only got to 7% accuracy on the training data. This network was scrapped quickly, as this would have taken far too long to train effectively.

The first neural network we were able to run to completion was the model that suffered from overfitting. This model ran significantly faster as each epoch took about 13 minutes, and we ran 4 epochs during training. This model had a 97% accuracy on the training data, but only a 30% accuracy on validation data. When we added the dropout layers to attempt to fix this issue, we only improved to 35% accuracy, still too low to be considered a success.

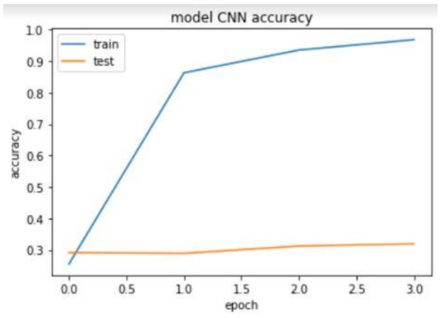


Figure 7 - Graph of accuracy over time for the convolutional neural network that suffered from overfitting, training data accuracy increases, testing data does not

When we added the first residual neural network with 50 layers and the linear layer for output, we had an accuracy of 93% on training data and 77% on validation data. This model took about 10 minutes for each epoch, and we ran 10 epochs during trainings. This was already a large improvement in accuracy of 42 percent just by switching to the ResNet architecture, and even a decrease in training time. But we can still have a little more improvement.

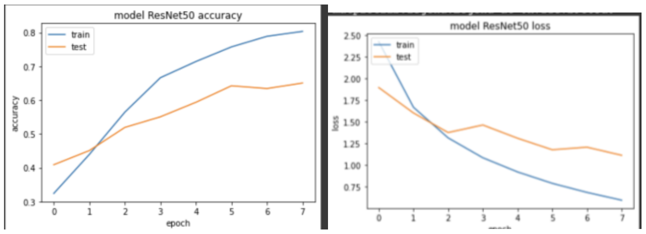


Figure 8 - accuracy of the model on training data versus testing data (left) loss function of model on training data versus testing data(right)

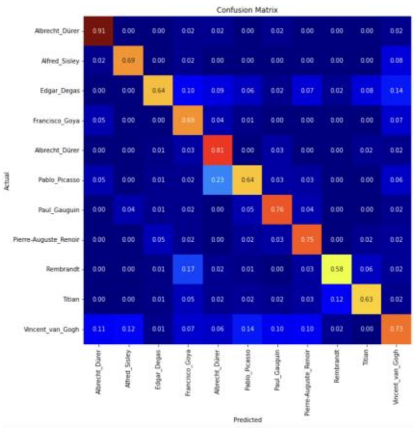


Figure 9 - Confusion matrix showing the accuracy of the model on each painter in the dataset.

The ResNet that ran with the Keras Image Data Generator ran for about the same time as the ResNet, but with significantly different accuracy readings. This ResNet ran about 35% accuracy on the training data and around 40% on the validation data. We are not exactly sure how the validation accuracy is higher than the training data. This difference in accuracy is why we decided not to run with the Image Data Generator during this project.

Lastly, our best working model with the ResNet followed by three linear layers took about ten minutes per epoch, and we ran ten epochs. The accuracy for this model ran about 97% accuracy on the training data with 82% accuracy on validation data. This model did best at classifying Durer and Degas, and classified Titan the worst.

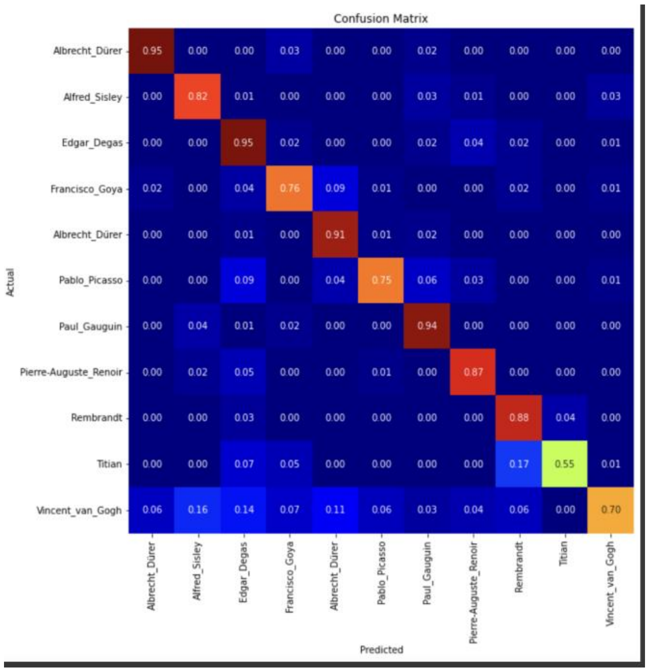
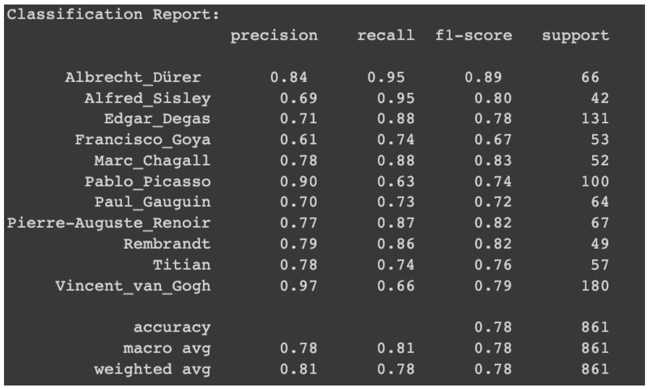


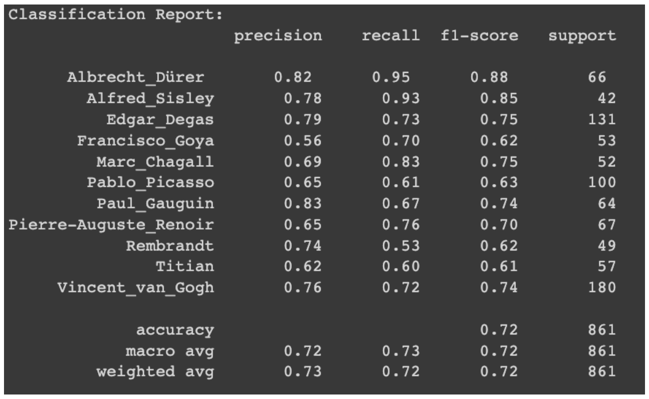
Figure 10 - Confusion matrix for our final neural network with ResNet and Linear Layers

The two models that ended up working the best were the ones that included ResNet and trained on the normal training data rather than the Image Data Generator. Our best model had an average f1-score that was better: 78% versus 72%. However, a curious feature occurred where an artist like Alfred Sisley ended up having a higher f1-score in the model that had a lower average. This artist also had the lowest number of painting when testing. Though we are not sure why this occurred, it could be worth investigating why the overall lower model had a better score for this artist.



```
Classification Report:
                       precision    recall  f1-score   support

      Albrecht_Dürer       0.84      0.95      0.89        66
       Alfred_Sisley       0.69      0.95      0.80        42
         Edgar_Degas       0.71      0.88      0.78       131
      Francisco_Goya       0.61      0.74      0.67        53
        Marc_Chagall       0.78      0.88      0.83        52
       Pablo_Picasso       0.90      0.63      0.74       100
        Paul_Gauguin       0.70      0.73      0.72        64
Pierre-Auguste_Renoir       0.77      0.87      0.82        67
           Rembrandt       0.79      0.86      0.82        49
              Titian       0.78      0.74      0.76        57
    Vincent_van_Gogh       0.97      0.66      0.79       180

            accuracy                           0.78       861
           macro avg       0.78      0.81      0.78       861
        weighted avg       0.81      0.78      0.78       861
```

Figure 11 - Summary of results for the final neural network with ResNet-50 and 3 linear layers



```
Classification Report:
                       precision    recall  f1-score   support

      Albrecht_Dürer       0.82      0.95      0.88        66
       Alfred_Sisley       0.78      0.93      0.85        42
         Edgar_Degas       0.79      0.73      0.75       131
      Francisco_Goya       0.56      0.70      0.62        53
        Marc_Chagall       0.69      0.83      0.75        52
       Pablo_Picasso       0.65      0.61      0.63       100
        Paul_Gauguin       0.83      0.67      0.74        64
Pierre-Auguste_Renoir       0.65      0.76      0.70        67
           Rembrandt       0.74      0.53      0.62        49
              Titian       0.62      0.60      0.61        57
    Vincent_van_Gogh       0.76      0.72      0.74       180

            accuracy                           0.72       861
           macro avg       0.72      0.73      0.72       861
        weighted avg       0.73      0.72      0.72       861
```

Figure 12 - Summary of results for neural network wtih just ResNet-50 and a linear layer

Through this project, we have learned that using pretrained models are best way to get accurate training the fastest. Training from scratch will almost always take longer and produce lesser results. Therefore, if pretrained models and data exist, almost always try to use them.

5. FUTURE WORK

While we do have a good base for the model that we have created here, there is much more that could be done to improve it. If we had additional time to work on this project, one of the first things that we would want to improve is to adapt our model to look for more features specifically related to paintings. There is a good article by IEEE Signal Processing Magazine that gives a lot of detail into how to create neural networks and statistical models to identify paintings based on certain features specific in painting like brushstrokes, textures, wavelet templets and more. While this paper focuses on Van Gogh, we could implement the information provided in a way to allow us to widen the scope to be able to train our model for all the artists we are looking at.[8] With a deep dive into this paper we could vastly improve our model, and possibly find additional features to be able to extract from paintings to ascertain the creator. With additional time we could also explore the methods referenced in the Stanford papers mentioned in the introduction. The paper by Blessing and Wen mentions using classification algorithms such as Naïve Bayes, and SVM through the LIBLINEAR library.[1] If we could find a way to integrate these algorithms and methods into our neural network, there's a possibility that our accuracy could be improved.

With these modifications to our neural network, this effective machine learning program could be used in a variety of ways. Currently, there are efforts to use neural networks such as this to be able to identify vast collections of digitally scanned pieces of artwork that galleries such as the MoMA or the Met possess.[4] It takes human art historians large amounts of time to identify the artists of each painting in the collection. On top of this, research in this field is being developed to help art critics and historians be able to identify forgeries of artwork done collaboratively. While in the past, identifying forgeries with artificial intelligence has been very tricky, new studies and methodologies recently discovered can help to identify forgeries when AI is given smaller sample sizes using image entropy. Image entropy is how much diversity and "randomness" exists in the image.[9] This means a monotone painting will have a much lower entropy than a colorful one. Implementing this novel ideology into our neural network can allow our program to be able to do both image-artist identification as well as be able to detect forgeries. If we could effectively implement these capabilities into our program, this could become highly sought-after technology in the art world.

The usefulness for a program like this in the far future has uses, again, at places like the MoMA. In addition to wanting a neural network to be able to identify paintings and forgeries, they want a neural network that can do the same things, but with sculptures and other 3D artwork.[4] Therefore, the research and work going into this project is not in vain.

6. CONCLUSION

We successfully demonstrated that this machine learning project can be used to identify these elven artists. We achieved an overall accuracy of 82% while incorporating Residual Neural Networks in tandem with Linear Layers. Through this project we learned a lot about Keras, ResNet, data manipulation, data loading, and we were able to apply techniques that we learned in class. We were able to compare models we created while also understanding how setting up a model differently affects the overall performance.

7. REFERENCES

[1] Blessing, A., & Wen, K. (2010). Using machine learning for identification of art paintings. *Technical report.*

[2] Johnson, C. R., Hendriks, E., Berezhnoy, I. J., Brevdo, E., Hughes, S. M., Daubechies, I., ... & Wang, J. Z. (2008). Image processing for artist identification. IEEE Signal Processing Magazine, 25(4), 37-48.

[3] Viswanathan, N. (2015). Artist Identification with Convolutional Neural Networks. *Technical report.*

[4] MoMA. (n.d.). Identifying art through machine learning: Moma. The Museum of Modern Art. Retrieved July 20, 2022, from https://www.moma.org/calendar/exhibitions/history/identifying-art

[5] Bhandari, A. (2020, August 16). *Image augmentation keras: Keras imagedatagenerator*. Keras Image Augmentation. Retrieved July 24, 2022, from https://www.analyticsvidhya.com/blog/2020/08/image-augmentation-on-the-fly-using-keras-imagedatagenerator/#:~:text=Keras%20ImageDataGenerator%20is%20a%20gem,up%20on%20the%20overhead%20memory

[6] Chollet, F., & others. (2015). Keras.

[7] Dwivedi, P. (2019, March 27). *Understanding and coding a ResNet in Keras*. Medium. Retrieved July 25, 2022, from https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33#:~:text=The%20ResNet%2D50%20model%20consists,over%2023%20million%20trainable%20parameters.

[8] Jr, Johnson, & Hendriks, E. & Berezhnoy, Igor & Brevdo, Eugene & Hughes, Shannon & Daubechies, Ingrid & Li, Jia & Postma, Eric & Wang, James. (2008). Image processing for artist identification. Signal Processing Magazine, IEEE. 25. 37 - 48. 10.1109/MSP.2008.923513.

[9] Anderson, M. (2021, June 24). *Amateurs' al tells real Rembrandts from fakes*. IEEE Spectrum. Retrieved July 25, 2022, from https://spectrum.ieee.org/the-rembrandt-school-of-ai-an-algorithm-that-detects-art-forgery

[10] Wood, T. (2019, May 17). *F-score explained*. DeepAI. Retrieved July 27, 2022, from https://deepai.org/machine-learning-glossary-and-terms/f-score

[11] Brownlee, J. (2021, January 12). *Gentle introduction to the adam optimization algorithm for deep learning*. Machine Learning Mastery. Retrieved July 26, 2022, from https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

[12] ICARO. (2019, March 2). *Best artworks of All time*. Kaggle Dataset. Retrieved July 27, 2022, from https://www.kaggle.com/datasets/ikarus777/best-artworks-of-all-time