Art Classifier

Malak Matar (2079203), Narges Najiantabriz (2053031)

Abstract

Our project is designed to categorize art into five classes: drawings, engraving, iconography, painting, sculpture, using deep learning. If categorized as a painting, the model will predict the painting's style. This report provides the details of the approach that we took. We begin by discussing the significance of automated art classification and describe the dataset used, including the prepossessing steps necessary to prepare the images for training. The architecture of our CNN model, implemented in Keras[4], and VGG16[5] is outlined, followed by a discussion of the training process, and the use of evaluation metrics to assess model performance. At the end, we discuss the challenges encountered during the project and propose potential improvements.
Key Terms: Art, Classification, Keras, TensorFlow - VGG16

1 Introduction

The classification of artworks into distinct categories is a complicated task due to the various similarities that makes it hard for a machine learning model to classify. Since doing manual classification is a lengthy and impractical process, in this project we aim to use the potential of deep learning techniques, specifically leveraging the Keras library, to automatically classify artworks into five categories: drawings, engraving, iconography, painting, and sculpture. Since Keras is a high-level neural network API which runs on top of TensorFlow, it was very suitable for the purpose of our project. Keras was employed to design, train and evaluate a CNN to classify artworks. The dataset used is from Kaggle which contained around 10,000 photos in total from all the categories. In addition, we also trained VGG16 model over 3 different paint styles.

We compared our results with existing approaches on Kaggle [1] and found that our model outperforms the best existing solution by 10%. Similarly, attempts to classify painting styles [2] achieved less than 50% accuracy, reinforcing the need for our improved approach.

2 Methodology

2.1 The Dataset

Since we were basically creating two models, we had to work with two datasets. Both datasets were taken from Kaggle. The dataset for art types was balanced, we only needed to separate the training and validation data. However, the dataset for painting movement style was biased towards some specific styles and we had to balance them. The dataset[3] consisted of 13 styles with a size of 30GB. So, to manage computational efficiency, we narrowed our focus to three styles: Japanese Art, Primitive, and Neoclassicism.

2.2 Data Pre-processing

The pre-processing steps involved defining the labels and image sizes for both models and then loading the image data from specified directories. Images were read from their respective folders, resized to a consistent size, and assigned class labels based on their directory structure. This ensures that all images fed into the models are of uniform dimensions, facilitating effective training. By resizing images to 64x64 pixels we decrease the total number of pixels per image,

which lowers the computational load without significant loss of visual information. This resizing step also standardizes the input size across all images, ensuring consistency and enabling the neural network models to process and learn from the data more efficiently.The data was also normalized by scaling pixel values to a range of 0 to 1, improving the models' convergence during training. Furthermore, we applied data augmentation techniques such as rotation, shifting, zooming, and flipping to artificially expand the training dataset, enhancing the models' ability to generalize to new, unseen data. With this comprehensive pre-processing pipeline we ensured that the input data is in optimal shape for training robust and accurate classification models.

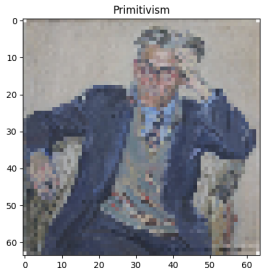

Figure 1: example of pre-processed art style

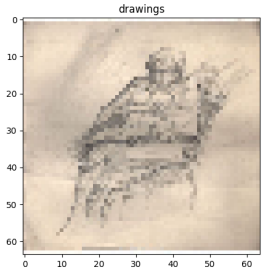

Figure 2: example of pre-processed art type

2.3 The Model Overview

Both classifiers were built using the framework of Keras to achieve successful classification of artworks. For the art style classification, VGG16 CNN was used. The VGG16 architecture consists of the following convolutional layers:

- Layer 1: 64 filters of size 3x3, followed by another convolutional layer with 64 filters of size 3x3.
- Layer 2: 128 filters of size 3x3, followed by another convolutional layer with 128 filters of size 3x3.
- Layer 3: 256 filters of size 3x3, followed by another two convolutional layers with 256 filters of size 3x3.
- Layer 4: 512 filters of size 3x3, followed by another two convolutional layers with 512 filters of size 3x3.
- Layer 5: 512 filters of size 3x3, followed by another two convolutional layers with 512 filters of size 3x3.

After each block of convolutional layers, a max-pooling layer is applied to reduce the spatial dimensions (width and height) of the feature maps while retaining the important features. This helps in reducing the computational load and controlling overfitting. We excluded the top fully connected layers of the VGG16 model to add custom fully connected layers to tailor the network for our specific task as follows:

1. Flatten Layer: This layer converts the 3D output from the last convolutional layer of the VGG16 model into a 1D vector.
2. Fully Connected Layer 1: This dense layer has 1024 neurons and uses the ReLU activation function to introduce non-linearity.
3. Dropout Layer 1: This dropout layer has a dropout rate of 50%, which helps prevent overfitting by randomly setting half of the input units to zero at each update during training.
4. Fully Connected Layer 2: Another dense layer with 1024 neurons, also using the ReLU activation function.
5. Dropout Layer 2: A second dropout layer with a 50% dropout rate.
6. Output Layer: The final dense layer has neurons equal to the number of classes (in this case, 3 for 'Japanese Art', 'Neoclassicism', 'Primitivism') and uses the softmax activation function to produce probability distributions over the classes.

Layer (type)	Output Shape	Param #
conv2d_15 (Conv2D)	(None, 64, 64, 32)	896
max_pooling2d_15 (MaxPooling2D)	(None, 32, 32, 32)	0
conv2d_16 (Conv2D)	(None, 32, 32, 32)	9,248
max_pooling2d_16 (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_17 (Conv2D)	(None, 16, 16, 64)	18,496
max_pooling2d_17 (MaxPooling2D)	(None, 8, 8, 64)	0
dropout_6 (Dropout)	(None, 8, 8, 64)	0
flatten_5 (Flatten)	(None, 4096)	0
dense_12 (Dense)	(None, 128)	524,416
dense_13 (Dense)	(None, 5)	645

Total params: 553,701 (2.11 MB)

Trainable params: 553,701 (2.11 MB)

Non-trainable params: 0 (0.00 B)

Figure 3: The art type model summary

The architecture of the CNN model used for art type classification is summarized in Figure 3. The model consists of three convolutional layers, each followed by a max-pooling layer. The first convolutional layer has 32 filters of size 3x3, followed by a max-pooling layer that reduces the spatial dimensions by half. This pattern is repeated with the second convolutional layer, which also has 32 filters, and the third convolutional layer, which has 64 filters. After the convolutional and pooling layers, a dropout layer with a rate of 50% is applied to prevent overfitting. The output is then flattened into a 1D vector of size 4096. This is followed by a fully connected (dense) layer with 128 neurons using the ReLU activation function. The final output layer is a dense layer with 5 neurons, corresponding to the 5 classes of art types, using the softmax activation function. The model has a total of 553,701 trainable parameters. This architecture effectively captures the features necessary for distinguishing between different art types while maintaining computational efficiency.

2.4 Model training

Our model training utilizes several effective strategies for improving performance and preventing overfitting. The dataset is partitioned into 70% for training and 30% for validation. This partitioning helps evaluate the model's performance on unseen data during training.

During training, data augmentation was employed, which dynamically augments the training data, enhancing the model's ability to generalize. The model was compiled using the Adam optimizer with a learning rate of 0.0001, appropriate for fine-tuning model weights. The loss function used was Sparse Categorical Crossentropy, suitable for multi-class classification tasks.

To optimize training, we utilized three key callbacks: EarlyStopping, ReduceLROnPlateau, and ModelCheckpoint. EarlyStopping halts training when validation loss stops improving for 10 consecutive epochs, thereby preventing overfitting. ReduceLROnPlateau reduces the learning rate by a factor of 0.2 if validation loss does not improve after 5 epochs, aiding convergence. ModelCheckpoint saves the best model based on validation accuracy, ensuring that the most performant model is retained. Over the course of 200 epochs, these strategies collectively contributed to a robust and well-optimized training regimen, fostering a model that balances accuracy and generalization on the provided dataset split.

3 Results

The accuracy of the first CNN we created for art type classification was 70%, we tried to further improve it so we added more augmentation to the validation data and callbacks in training process to enable automatic change of learning rate, saving the best model reached during training, and early stopping in case of increase in validation data loss(to avoid over fitting). By these improvements we got to an accuracy of 85%. For the painting style classification, our first attempt was using the ResNet50 CNN and it gave disappointing results. Although it got to a high accuracy but the accuracy and loss graphs suffered from a lot of fluctuation.

Figure 4: accuracy graph of art type classifier after improvements

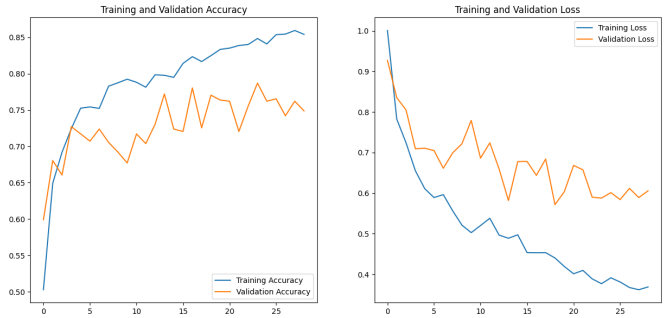

Figure 5: accuracy graph of art style classifier using ResNet50

So, after further research, VGG16 seemed like a better fit and it indeed gave outstanding results. The training accuracy increases steadily and reaches around 85% towards the end of the training. The validation accuracy also improves, but at a slower rate, reaching around 80%.

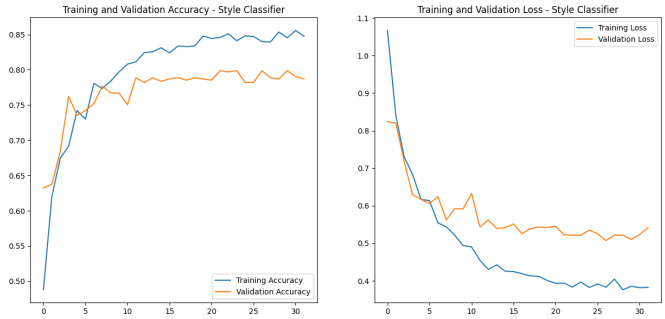

Figure 6: accuracy graph of art style classifier using VGG16

Saving the best models in the process of training on both the art types and art styles, we merged them so that if the input is classified as painting, it further predicts it's style as well.

Testing our two models on different images was satisfactory. Here are some test outputs that we got and you can see that images are labeled correctly:

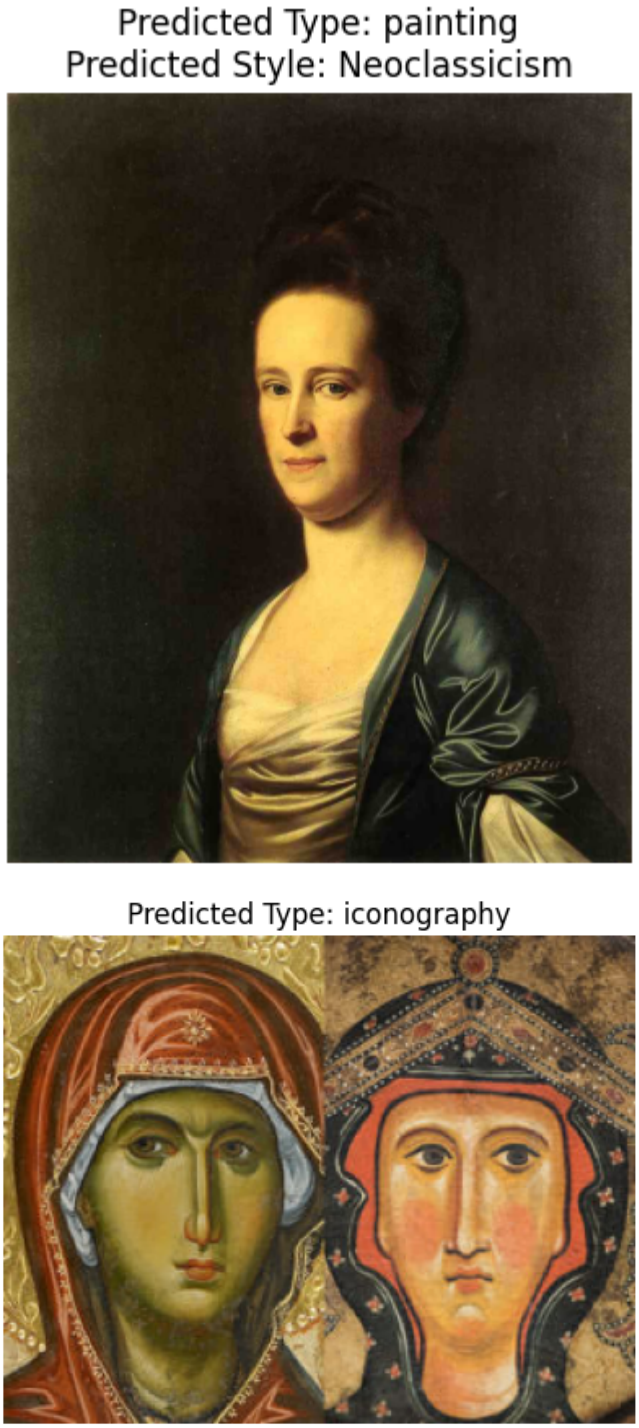

Figure 7: some examples of test outputs

Figure 8: some examples of test outputs

4 Conclusion

In this project, we successfully developed an automated art classifier capable of categorizing artworks into five distinct classes: drawings, engraving, iconography, painting, and sculpture. Additionally, for artworks classified as paintings,

our model further predicted the painting's style among Japanese Art, Primitive, and Neoclassicism.

We leveraged deep learning techniques and utilized the Keras library, along with the VGG16 architecture, to build our models. The data preprocessing steps, including resizing, normalization, and data augmentation, played a crucial role in improving the models' performance. Through iterative improvements and careful tuning of hyperparameters, we achieved an accuracy of 85% for art type classification and 80% for painting style classification.

Our results demonstrate the potential of deep learning in the field of art classification, providing an efficient and scalable solution for categorizing large collections of artworks. However, the project also highlighted several challenges, including the need for balanced datasets and the importance of selecting appropriate model architectures.

Overall, this project contributes to the field of automated art classification, providing a foundation for further research and development in this area.

5 Future Developments

Future work could involve:

1. Expanding the model to include more art styles
2. Recognize the styles of the other pieces not only paintings
3. Integrate a reverse image search feature to find out more information about the piece like the artist
4. Exploring more advanced deep learning architectures to enhance classification accuracy

References

[1] The Downhill. *Art Images: Drawings, Painting, Sculpture, Engraving*. 2021. URL: https://www.kaggle.com/datasets/thedownhill/art-images-drawings-painting-sculpture-engraving.

[2] Gabriel. *Recognising Art Style by Using Neural Network*. 2023. URL: https://github.com/Gabrielprogramist/Recognising-Art-Style-by-Using-Neural-Network/tree/main?tab=readme-ov-file.

[3] Sivarazadi. *WikiArt - Art Movements/Styles*. 2021. URL: https://www.kaggle.com/datasets/sivarazadi/wikiart-art-movementsstyles.

[4] Keras Team. *Keras API Reference*. 2015. URL: https://keras.io/api/.

[5] Keras Team. "Keras Applications - VGG16". In: (2023). URL: https://keras.io/api/applications/vgg/.