Lambton College | In Toronto

AML-1214(Term Project)

Art Author detection:

Identifying the artist from the art.

Group Name: Python Noobs

Group Members:

First name	Last Name	Student number
Aishlee	Aishlee	C0913045
Bimal	Kumar Shrestha	C0919385
Danilo	Diaz	C0889539
Ernie	Sumoso	C0881591
Shivam	Kumar Patel	C0907758

Table of Contents

Table of Figures

Abstract

"Art Author Detection: Identifying the Artist from the Art" is a project designed to address the problem of artist identification, seeking to enhance accessibility to art knowledge among its audience. To solve this problem, we have built a web application that utilizes visual recognition technology to identify artists from their art (primarily paintings).

The flow behind the application starts when a user uploads an image (in **JPG, PNG, or JPEG** format) containing any famous painting from the most reputable artists (listed in the following sections) and selects a model for the prediction. Then, the application's output will be two elements: the name of the predicted artist and a score, which indicates the probability or confidence of the prediction. To achieve this, we programmed an application using **Streamlit** and **FastAPI**, both are web development frameworks in Python that allow us to set the frontend and backend of the application.

The front-end view contains some text instructions and functionality for the users. Then, it presents two sections, one to upload an image and another to select a model. When both options (image and model) are chosen, the user can send his request. Finally, after some short processing time (1-5 seconds), the model makes a prediction. The front end shows the uploaded image (in its original size), and the results section displays the artist's name and score (confidence) of the prediction. We used pre-trained machine learning models using Keras and Tensorflow libraries to achieve image detection. We instantiated the **ResNet50** model, as well as a **VGG19** model. Both are convolutional neural networks (CNN) trained with more than one million images from ImageNet and used to classify images into a thousand categories. We also used JupyterLab notebooks and Kaggle computing resources to train both models using our art datasets, including **8446 art pieces from 50 artists**. Finally, we integrated this model within our web app.

Introduction

Art is one of the highest forms of expression of human emotions. It has been practiced since the early civilization of human beings. In this technological era, even computers can create art using complicated AI and millions of pictorial data. The problem we are trying to solve is the challenge of artist identification, seeking to enhance accessibility to art knowledge by accurately associating artists' names with their corresponding masterpieces. As is evident, not all individuals have access to art education, and even if they do, not everyone is appropriately dedicated to learning authors from the most famous paintings. Often, people will contemplate and appreciate art pieces that are presented in front of them. To help people learn artists' names and diffuse art education, we have designed **"Art Author Detection,"** a cutting-edge application for art enthusiasts, utilizing visual recognition technology to identify artists from their paintings.

To achieve this project, we employed a machine-learning model for image detection. By seamlessly integrating this model into a user-friendly web app interface, our users can effortlessly upload an image and get the corresponding prediction (along with the confidence score) of the artist's name linked to the artwork. Our project's **target audience** is art lovers, artists, art-curious people, and even users too lazy to learn about artwork authors. This app will provide them with a straightforward but dynamic platform for discovering the artist's name according to their unique creations.

Our approach involves constructing a **deep-learning model** to identify the authors' names. Recognizing the depth of visual information, we build our model using **Convolutional Neural Networks** (CNNs). This advanced neural network architecture is designed for image-related tasks, allowing it to recognize and interpret features such as the colours employed and the patterns in each painting. Using CNNs equips our system with the ability to capture patterns and spatial dependencies within images, making it adept at **unravelling the artist names that define their masterpieces**.

Another critical step of our project is building the web application (both front-end and back-end) needed for user interaction. We chose to use **Streamlit** and **FastAPI** stack, as they involve a fast way to build a web application using Python**,** especially for short-scoped projects that need quick responses. This software development project was implemented using a Waterfall approach as its life cycle. This means that we designed our requirements first, then we designed the system, we proceeded to the development stage, and finally, we tested our application. The maintenance step was not included as we had to build this project with a predefined deadline. Later, we plan to develop more features and improve the model’s performance.

Methods

Dataset description

We have utilized the “Best Artworks of All Time” dataset from Kaggle source for this project. This dataset contains 8446 paintings by 50 different artists. All pictures are stored as JPG files. The fifty unique artists belong to various art paradigms with multiple techniques and centuries. We have our image files organized in two ways; first, we have them organized by artist name, where each folder groups paintings from one unique author. On the other hand, we also have the images resized (224 x 224 pixels) and saved in the same folder. The artists included in the dataset are:

Amedeo Modigliani	Hieronymus Bosch	Alfred Sisley	Edgar Degas	Henri Rousseau
Vasiliy Kandinskiy	Kazimir Malevich	Pieter Bruegel	Rembrandt	Georges Seurat
Diego Rivera	Mikhail Vrubel	Marc Chagall	Titian	Paul Klee
Claude Monet	Pablo Picasso	Giotto di Bondone	Henri de Toulouse-Lautrec	Piet Mondrian
Rene Magritte	Peter Paul Rubens	Sandro Botticelli	Gustave Courbet	Joan Miro
Salvador Dali	Pierre-Auguste Renoir	Caravaggio	Camille Pissarro	Andy Warhol
Edouard Manet	Francisco Goya	Leonardo da Vinci	William Turner	Paul Gauguin
Andrei Rublev	Frida Kahlo	Diego Velazquez	Edvard Munch	Raphael
Vincent van Gogh	El Greco	Henri Matisse	Paul Cezanne	Michelangelo
Gustav Klimt	Albrecht Dürer	Jan van Eyck	Eugene Delacroix	Jackson Pollock

Figure 1. Some Famous Paintings and Artists from the dataset

Machine Learning method

For our machine learning model, we have utilized CNN (Convolutional Neural Network), which is like traditional ANN (Artificial Neural Network) in the sense that both use neurons to self-optimize their weights during training. Both of their neurons receive some input, which is operated using some scalar product and then some nonlinear function like relu, softmax, etc. to output the numbers for the other layer. CNN is primarily used in the field of pattern recognition in images. The traditional ANN cannot handle the computational complexity needed to handle image data, while CNN has a better architecture for dealing with that data type. This architecture can be distributed into four key areas:

Input Layer: The input layer will hold the image's pixel values.

Convolutional Layer: The output of neurons connected to local portions of the input will be determined by the convolutional layer by calculating the scalar product between their weights and the region related to the input volume. The rectified linear unit (abbreviated ReLu) attempts to apply an 'elementwise' activation function, such as sigmoid, to the previous layer's activation output.

Pooling Layer: This layer will downsize the sample along the spatial dimensions of the provided input, lowering the number of parameters inside that activation even further.

The fully connected layers: This will then perform the same functions as typical ANNs, attempting to generate class scores from the activations for classification. To boost performance, it is also proposed that ReLu be utilized between these layers (Keiron O'Shea, 2015).

Of the two models we used, **ResNet50** got higher accuracy at **52.30%** at validation, which took close to **6 hours to train.**

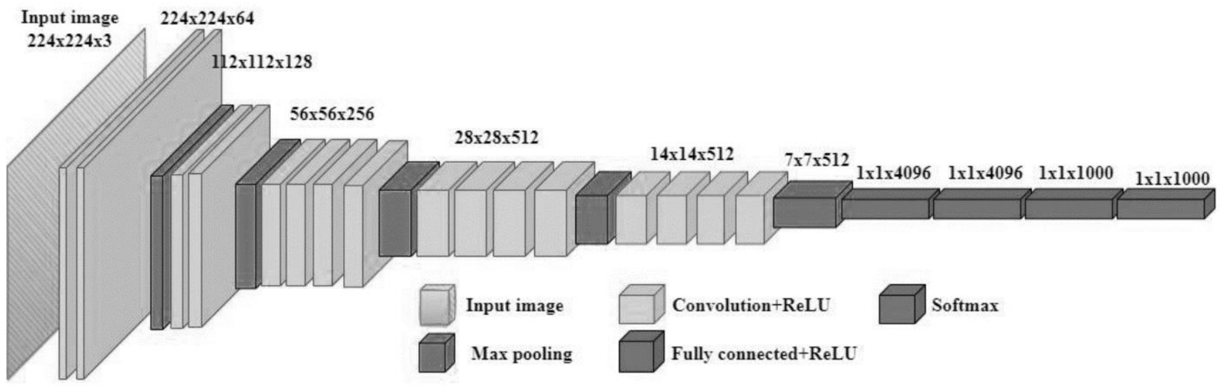


Figure 2. VGG19 Architecture

VGG19

- Nineteen layers-deep networks.
- Pretrained on more than a million image data.
- The network can learn feature-rich representations from a wide range of images.
- Image input size is 224x224 pixels.

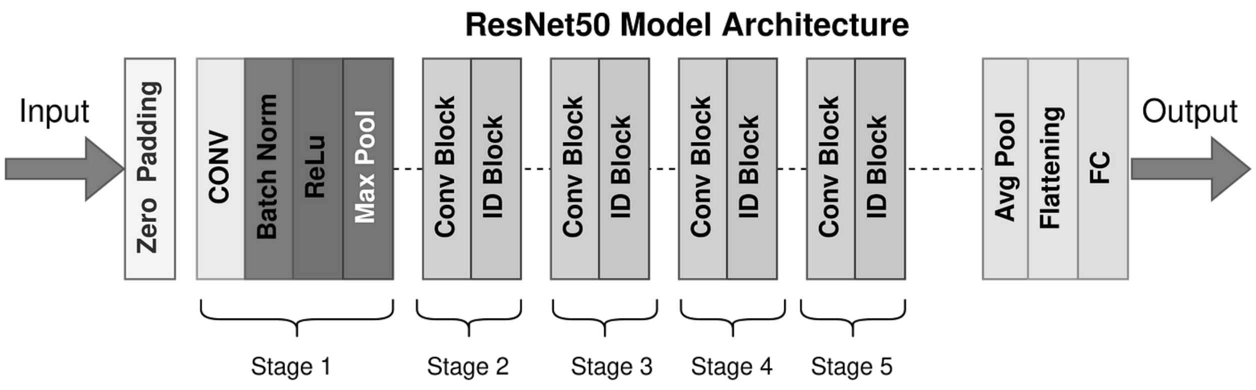


Figure 3. ResNet50 Architecture

ResNet50

- Fifty layers-deep networks.
- Pretrained on more than a million image data.
- Can classify images into 1000 object categories.
- Image input size is 224x224 pixels x3 colors (RGB).

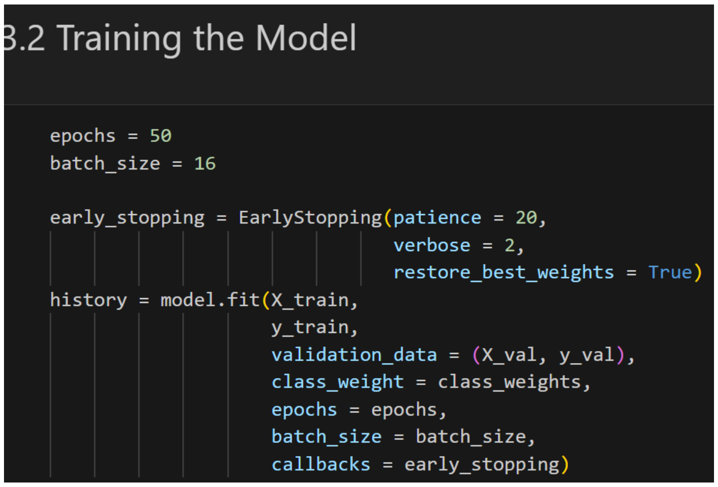
Software development method

The project employed a Waterfall SDLC, a sequential software development process in which the process falls gradually downward through a sequence of steps conducted to build the product. Four stages out of five were followed for this project, leaving out the maintenance stage. In the requirement analysis phase, we defined a description of the features and functions of the project. Then, for the design stage, the planning and problem-solving for the project was done. This helped fix the overall idea for the algorithms, GUI, and software architecture design. The development phase starts after realizing all the project needs and design specifications. This is the stage at which the actual code was written and compiled into a working model, and the web app and notes were created. In other words, it was the process of transforming the complete requirements and blueprints into a production environment. Then, the final stage for this project is the testing phase, known as verification and validation, which is a procedure that ensures a software solution fits the original requirements and specifications and serves its intended function. This stage and the development stage are iterative, as many issues might be found in the testing phase (Bassil, 2012).

Deployment

The project was started on Python, and different modules like NumPy, pandas, matplotlib, TensorFlow, OpenCV, etc., were used to explore, manipulate and visualize data. The obtained dataset had lots of images and artists to explore. So, an exploration of the artist's names and the paintings was done. **One of the challenges** was the **imbalance** in the class distribution as famous artists had numerous paintings, whereas lesser-known artists had fewer paintings. This might lead to bias towards the majority of classes in the training period, so to counter this, class weight was calculated prior to the training and assigned to the classes during model training. This helps the model to pay more attention to the minority classes during the training process.

Once everything was ready and set up, we built two CNN Models, ResNet50 and VGG19, which were loaded. Creating an image processing model from scratch can be complex and computationally expensive. On top of that, it needs a considerable amount of data to train, which is not feasible for the time and resources we have available. We used a process called transfer learning where we used the pre-trained models ResNet50 and VGG19 and customized the model per our needs. The idea is to transfer the knowledge learned by the pre-trained model on a large dataset to a different but related task. One of the main benefits of using a pre-trained model as our base is that we can speed up and simplify the training process. Instead of starting from nothing, you can use the weights and features learned by the pre-trained model as a starting point and fine-tune them to your specific task. After the model was customized with our training set, the training was done, which took almost 6 hours for both the models. Early stopping was used with favourable patience to use the time more efficiently. This helped significantly reduce the time as iterating the training process numerous times would be time- and resource-consuming. As stated earlier, using class weight also helped improve the model by decreasing the bias.



3.2 Training the Model

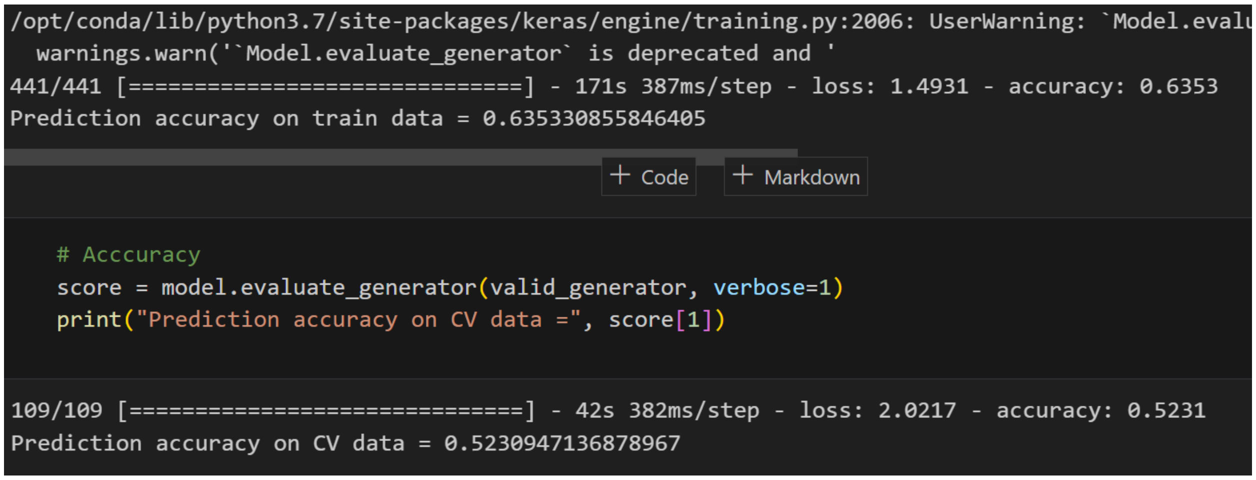
```
epochs = 50
batch_size = 16

early_stopping = EarlyStopping(patience = 20,
                               verbose = 2,
                               restore_best_weights = True)
history = model.fit(X_train,
                    y_train,
                    validation_data = (X_val, y_val),
                    class_weight = class_weights,
                    epochs = epochs,
                    batch_size = batch_size,
                    callbacks = early_stopping)
```

Figure 4. Training process for VGG19 model

Once the training process was completed, the accuracy and other scores were evaluated. Since the model performed reasonably well, it was extracted for the demo in the web application. The exact process was also revised for the other model, ResNet50. Once both models were ready, the web app development was initiated. For that, Streamlit was used for the front-end development and FastAPI for the back-end development. Both are Python frameworks that are used for short-scoped yet powerful projects. Streamlit is an open-source Python Library that allows users to create fascinating and robust web applications for machine learning and data science. Similarly, FastAPI is a modern web framework for building Restful APIs with Python 3.8+. This framework was chosen for its simplicity, speed, and robustness. The web application had a UI for uploading the painting and choosing two models. Once the user presses the make prediction button, the input is passed to the backend, where the prediction is done using the extracted models. The model performed well, considering the time it was given for training image data.

Results

The project started with research on the problem statement and discussion about the web application's methodology and features, which went through numerous trials and errors. Fixing problems and bugs along the journey, the project helped provide insights into the working mechanisms of neural networks, particularly on how convolutional neural networks work. The team was also introduced to frameworks like Streamlit and FastAPI, which assisted in personal growth and skill acquisition.



```
/opt/conda/lib/python3.7/site-packages/keras/engine/training.py:2006: UserWarning: `Model.evalu
  warnings.warn('`Model.evaluate_generator` is deprecated and '
441/441 [==============================] - 171s 387ms/step - loss: 1.4931 - accuracy: 0.6353
Prediction accuracy on train data = 0.635330855846405
```

```
# Acccuracy
score = model.evaluate_generator(valid_generator, verbose=1)
print("Prediction accuracy on CV data =", score[1])
```

```
109/109 [==============================] - 42s 382ms/step - loss: 2.0217 - accuracy: 0.5231
Prediction accuracy on CV data = 0.5230947136878967
```

Figure 5. Model Evaluation

The models and the web app created performed well even though the validation accuracy score was not too high, probably due to class imbalance and the presence of numerous classes and further work is required. Apart from the project's objective to create a model that predicted paintings, there was also the learning aspect. Besides that, the models were overfitting a bit as the scores on the test dataset were lower. Parameter tuning was not possible for this large dataset as tuning parameters need the model to go through multiple iterations of training. There were time and computational parameters that obstructed the path to this measure. The class imbalance also led to some bias in the prediction. Even though the concept of class weights was used during the model training, the model still faced some issues while predicting the arts of lesser-known artists.

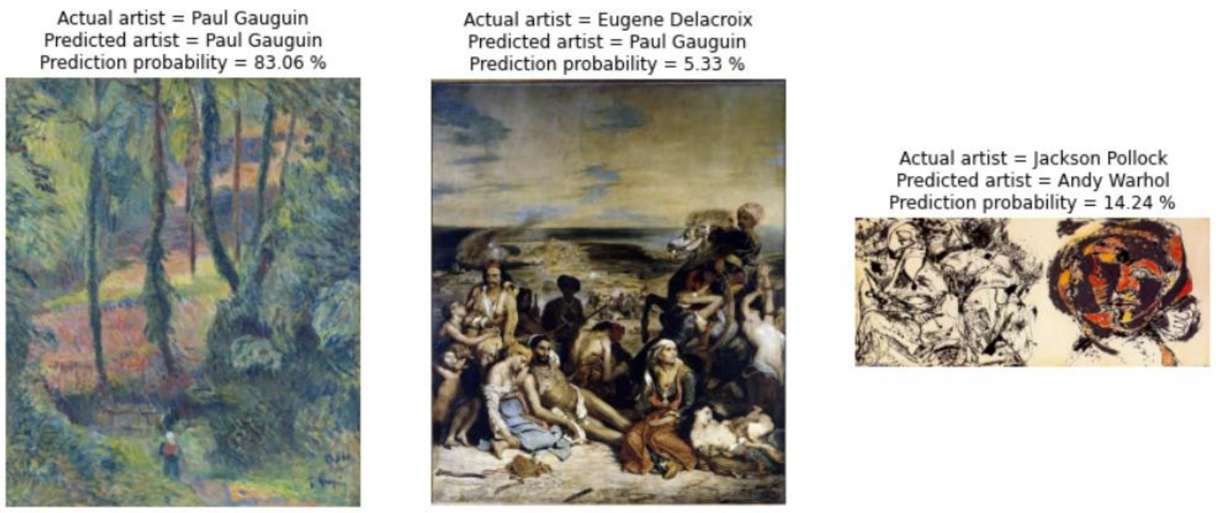
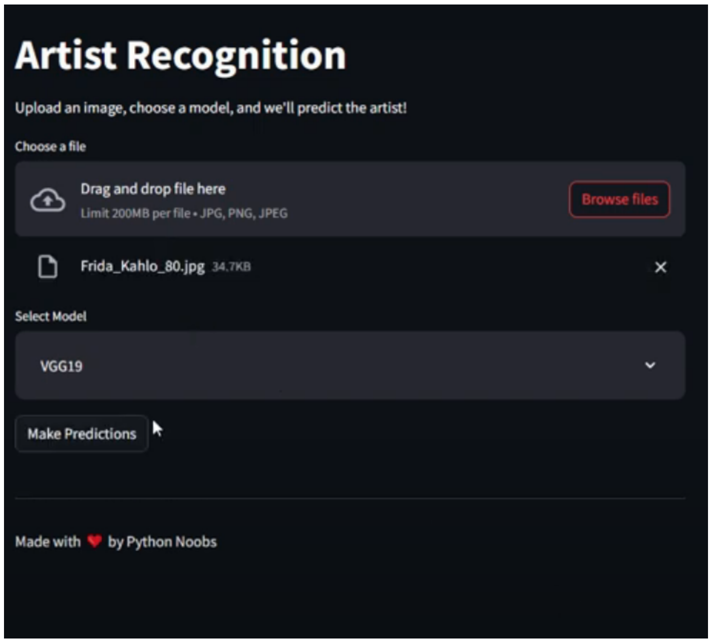

Actual artist = Paul Gauguin
Predicted artist = Paul Gauguin
Prediction probability = 83.06 %
Actual artist = Eugene Delacroix
Predicted artist = Paul Gauguin
Prediction probability = 5.33 %
Actual artist = Jackson Pollock
Predicted artist = Andy Warhol
Prediction probability = 14.24 %

Figure 6. Prediction on the paintings

On the contrary, the implementation of the web app, both back and frontend, was smooth. All the functionalities were functioning correctly without any hiccups. The steps for the web app are demonstrated below.

STEP 1: The user uploads an image to the web application front-end.


Artist Recognition

Upload an image, choose a model, and we'll predict the artist!

Choose a file

Drag and drop file here

Limit 200MB per file • JPG, PNG, JPEG

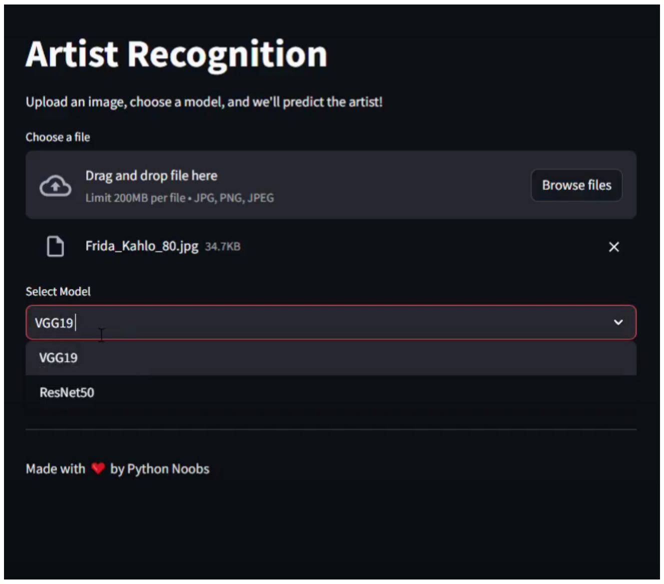
Frida_Kahlo_80.jpg 34.7KB

Select Model

VGG19

Figure 7. User Uploads Picture

STEP 2: The user chooses one of two given models (ResNet50 and VGG19) for prediction.


Artist Recognition

Upload an image, choose a model, and we'll predict the artist!

Choose a file

Drag and drop file here
Limit 200MB per file • JPG, PNG, JPEG

Browse files

Frida_Kahlo_80.jpg 34.7KB

Select Model

VGG19

VGG19
ResNet50

Made with ❤ by Python Noobs

Figure 8. The user selects the model.

STEP 3: The user clicks on 'Make Prediction', and the picture is sent to the backend for preprocessing and prediction.

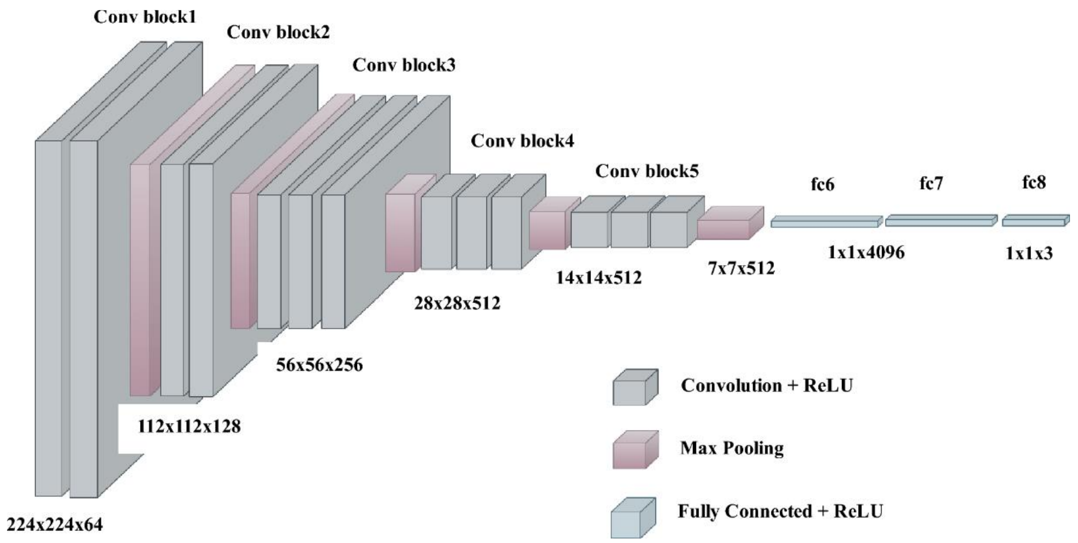
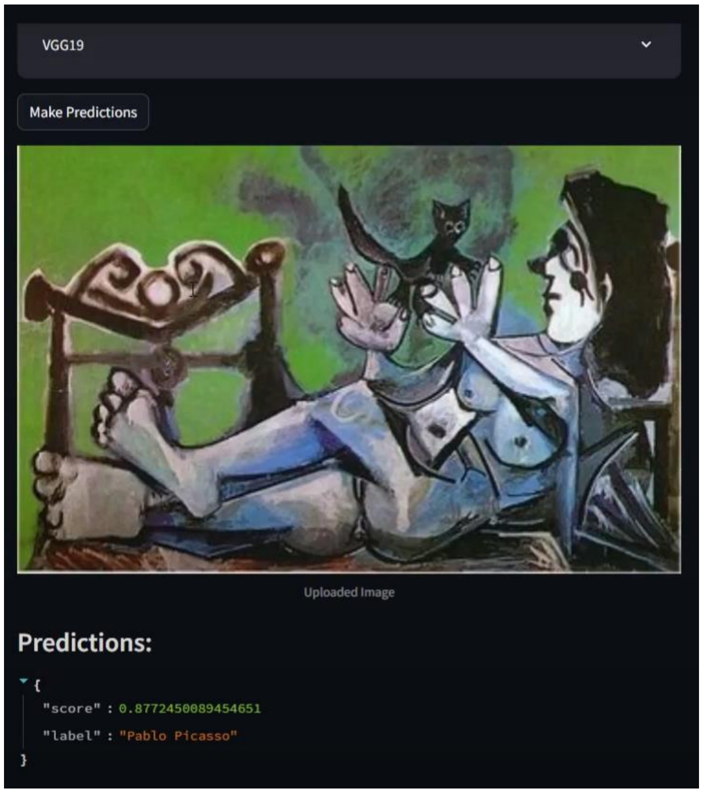


Figure 9. Processing in the model

STEP 4: The user gets the prediction with the probability of the prediction



Predictions:

```
{
  "score" : 0.8772450089454651
  "label" : "Pablo Picasso"
}
```

Figure 10. User gets

Conclusions and Future Work

The main problem this project aimed to solve was the challenge of artist identification from a painting and how it can help users get involved in art and associate the paintings with their creators. This can significantly help newcomers who want to start painting. As most artists have their unique style, if any user likes a painting style, they can use this app to find the artist, get inspiration, and learn from them. This can also be used in the field of education for teaching purposes as well.

There is always room for improvement. For this project as well, multiple areas were lacking. Multiple techniques used in the project did not hold up as anticipated, and they can be improved through further work. Some notable issues and how they can be improvised are mentioned below.

Overfitting: Multiple efforts can be taken to tackle this issue. In the field of CNN, the most effective techniques are the use of concepts like regularization techniques, including steps like L1 and L2 regularization, dropout, Hyperparameter tuning, And ensemble methods like bagging and boosting. These techniques can help improve the performance of the model massively.

Class Imbalance: Even though class weight assignment was used to decrease the bias in the training process. There was still bias observed in the prediction process. For artists of the minority class, their paintings were labelled as the ones from the majority class. To help overcome this, stacking and voting methods can be used. Instead of choosing between two models, one more option for more accurate prediction can be added as a union prediction where the mentioned techniques can be used. Techniques like oversampling or undersampling can also be used to lower the bias.

Web Application: As a part of improvements, while predicting the artists, different models can also be created to predict the art style. The user can then gain information about the artist and the theme of the art he prefers. This can benefit the end user.

References

Bassil, Y. (2012). A Simulation Model for the Waterfall. *International Journal of Engineering & Technology (iJET)*.

Kaiming He, X. Z. (2015, Dec 10). *Deep Residual Learning for Image Recognition.* Retrieved from arxiv: https://arxiv.org/abs/1512.03385

Karen Simonyan, A. Z. (2014, Sep 14). *Very Deep Convolutional Networks for Large-Scale Image Recognition.* Retrieved from arxiv: https://arxiv.org/abs/1409.1556

Keiron O'Shea, R. N. (2015, Dec 2). *Neural and Evolutionary Computing.* Retrieved from arxiv: https://arxiv.org/pdf/1511.08458.pdf

Kaggle - Best Artworks of All Time. Retrieved from:

https://www.kaggle.com/datasets/ikarus777/best-artworks-of-all-time/data

Github - Artist-Recognition-Noobs. Retrieved from:

https://github.com/NILodio/Artist-Recognition-Noobs/tree/develop

Python Official Documentation. Retrieved from:

https://www.python.org/

FastAPI Official Documentation. Retrieved from:

https://fastapi.tiangolo.com/

Streamlit Official Documentation. Retrieved from:

https://docs.streamlit.io/

Deploying Machine Learning models with Streamlit, FastAPI and Docker Retrieved from:

https://rihab-feki.medium.com/deploying-machine-learning-models-with-streamlit-fastapi-and-docker-bb16bbf8eb91