Classification of Paintings

Project: Deep Learning TensorFlow (2021.01, opencampus)
Nils Berns, John Kimani

Objectives

Goal

Implementation of a deep learning convolution neural network to classify painting of various influential painters.

We do this by achieving the following objectives:

1. Aggregate data from different sources to achieve objectives (2 – 4)
2. Match the painting by artist.
3. Differentiate between real and fraudulent paintings.
4. Differentiate between painting-stylised photos with actual painting.

Data

Data sets


[1]


[1]

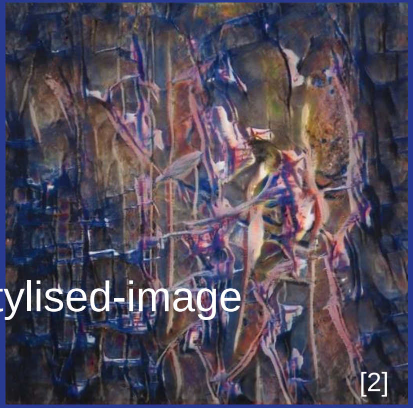

[1]

„Best Artworks of All Time“ [1]
- https://www.kaggle.com/ikarus777/best-artworks-of-all-time
- Collection of paintings of the 50 most influential artists of all time.
- 8446 images.

„https://thisartworkdoesnotexist.com/“ [2]
- With every request (1 per sec) an artificial painting is generated
- Style is quite modern/contemporary.
- Downloaded 1000 images.


[3]

„bored humans AI paintings“ [3]
- https://boredhumans.b-cdn.net/art/
- Database of artificial paintings generated with styleGAN2
- Downloaded 1000 images of ~5000 with.


[2]

„Paired landscape and Monet-Stylized image“ [4]
- https://www.kaggle.com/shcsteven/paired-landscape-and-monetstylised-image
- Used only the stylised images.
- 1030 images.


[1]

Challenges

with “Best Artworks of All Time“

- Most artists have less than 100 paintings others more than 500.
- Artists from the same genre share more features.
- Artist’s style changes over time.
- Trade-off between computational costs and the image size required for capturing all style features.

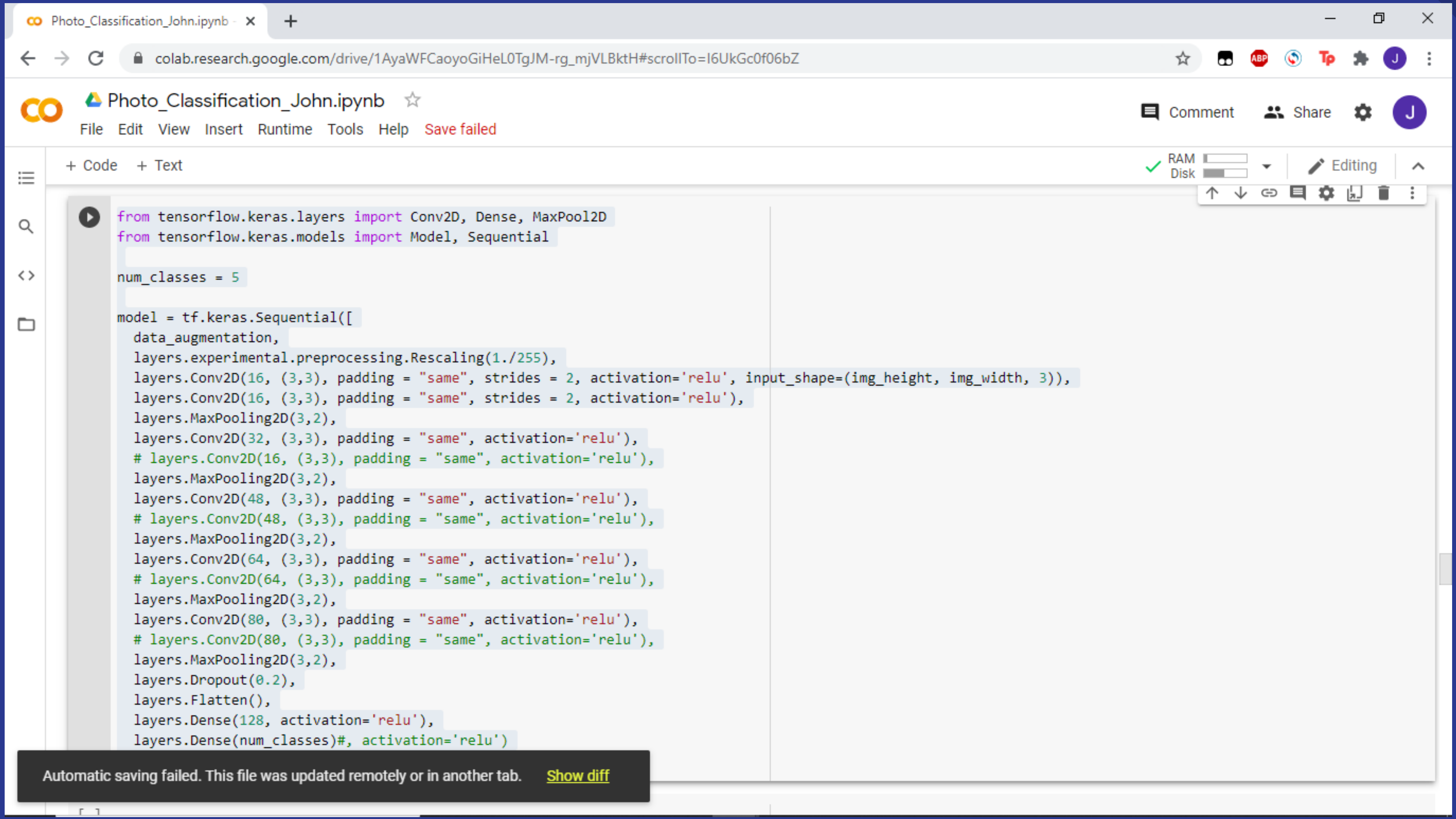
Implementation

- Choices:
- Minimum number of paintings an artist has to have.
- Inclusion of an additional class with fake/artificial paintings (max. resolution 512x512).

- Downloading and organising the data takes a while during first run.
- 80% are assigned to the training set, 10% to development and test set each.

Technical Environment

Implementation



Colab for coding

- TensorFlow 2.4.0
- Shairing colab via google Drive in Colab

Model

Data Augmentation

```
data_augmentation = tf.keras.Sequential(
  [
    layers.experimental.preprocessing.RandomFlip("horizontal", input_shape=(img_height, img_width, 3)),
    layers.experimental.preprocessing.RandomRotation(0.1),
    layers.experimental.preprocessing.RandomZoom(0.1),
  ]
)
```

Model Architecture

```
from tensorflow.keras.layers import Conv2D, Dense, MaxPool2D
from tensorflow.keras.models import Model, Sequential
num_classes = 5
model = tf.keras.Sequential([
  data_augmentation,
  layers.experimental.preprocessing.Rescaling(1./255),
  layers.Conv2D(16, (3,3), padding = "same", strides = 2, activation='relu', input_shape=(img_height, img_width, 3)),
  layers.Conv2D(16, (3,3), padding = "same", strides = 2, activation='relu'),
  layers.MaxPooling2D(3,2),
  layers.Conv2D(32, (3,3), padding = "same", activation='relu'),
  layers.MaxPooling2D(3,2),
  layers.Conv2D(48, (3,3), padding = "same", activation='relu'),
  layers.MaxPooling2D(3,2),
  layers.Conv2D(64, (3,3), padding = "same", activation='relu'),
  layers.MaxPooling2D(3,2),
  layers.Conv2D(80, (3,3), padding = "same", activation='relu'),
  layers.MaxPooling2D(3,2),
  layers.Dropout(0.2),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes
])
```

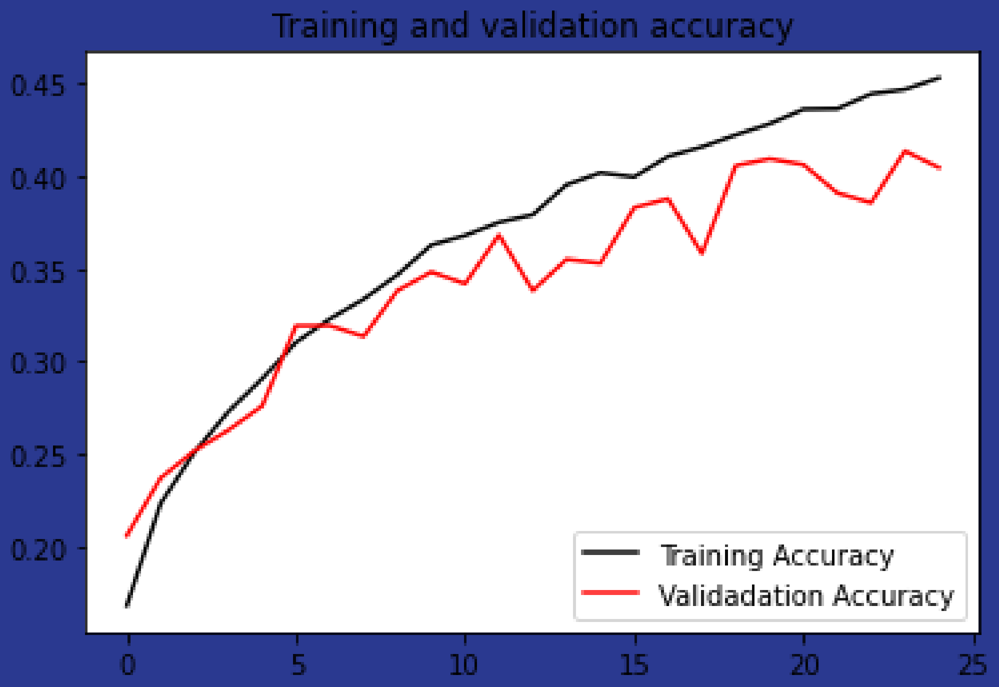
Optimiser

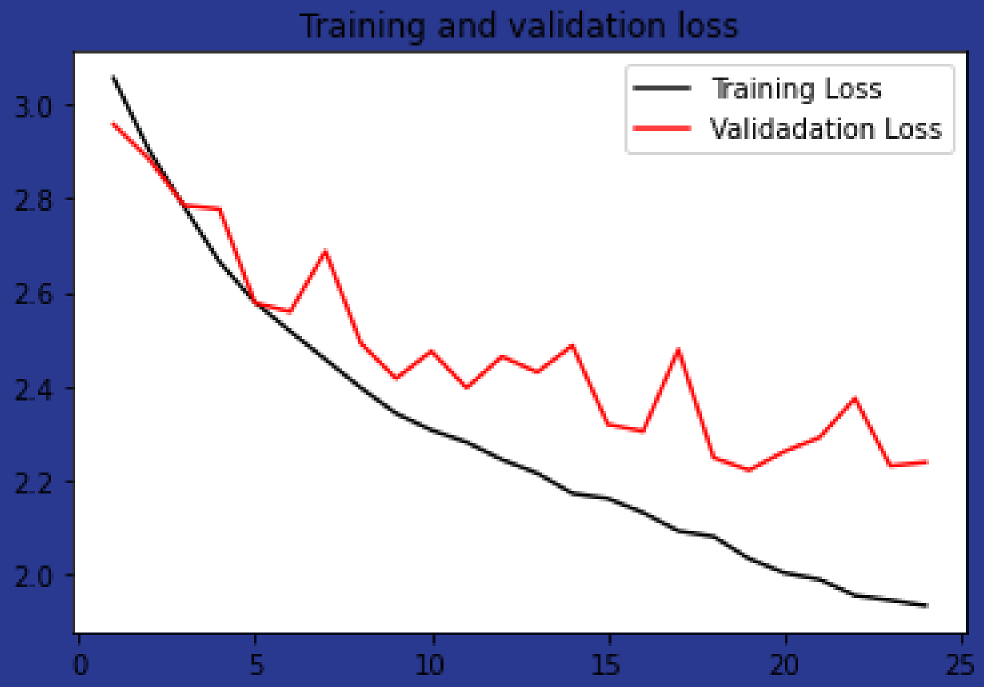
```
model.compile(
  optimizer='adam',
  loss=tf.losses.SparseCategoricalCrossentropy(from_logits=True),
  metrics=['accuracy'])
```

Performance and Result

50 artists and Monet Stylised image

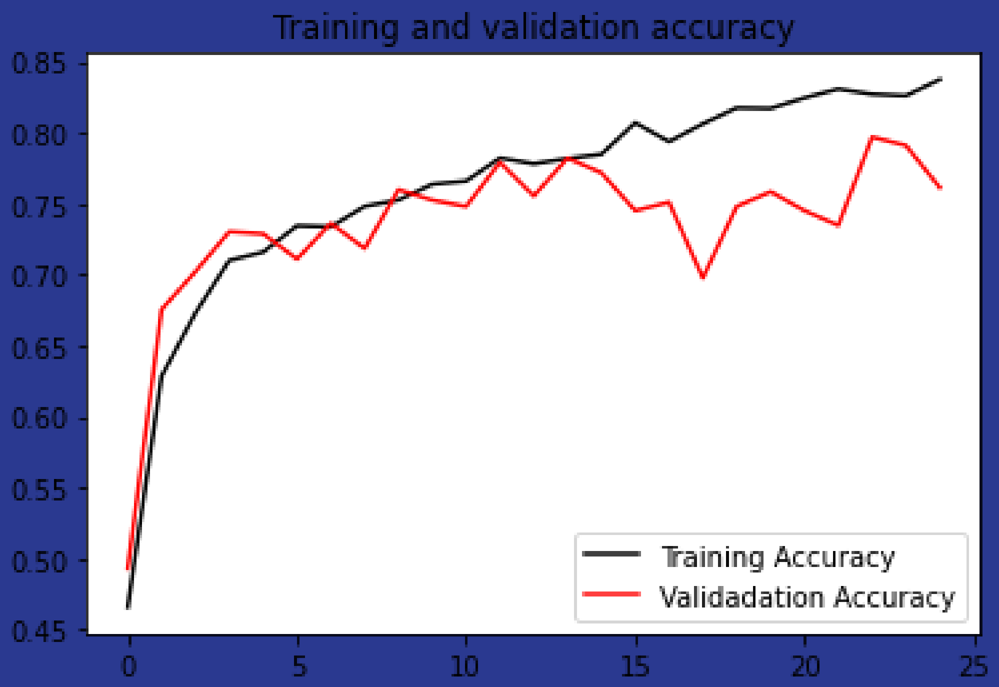
Model with data augmentation and 1 drop-out layer

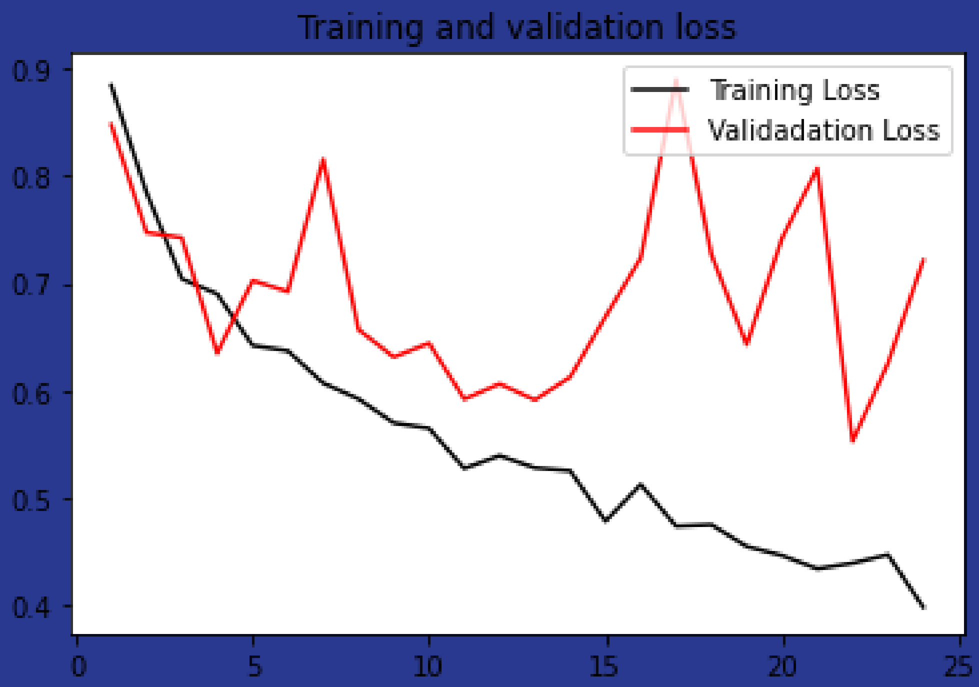




4 artists and Monet Stylised image

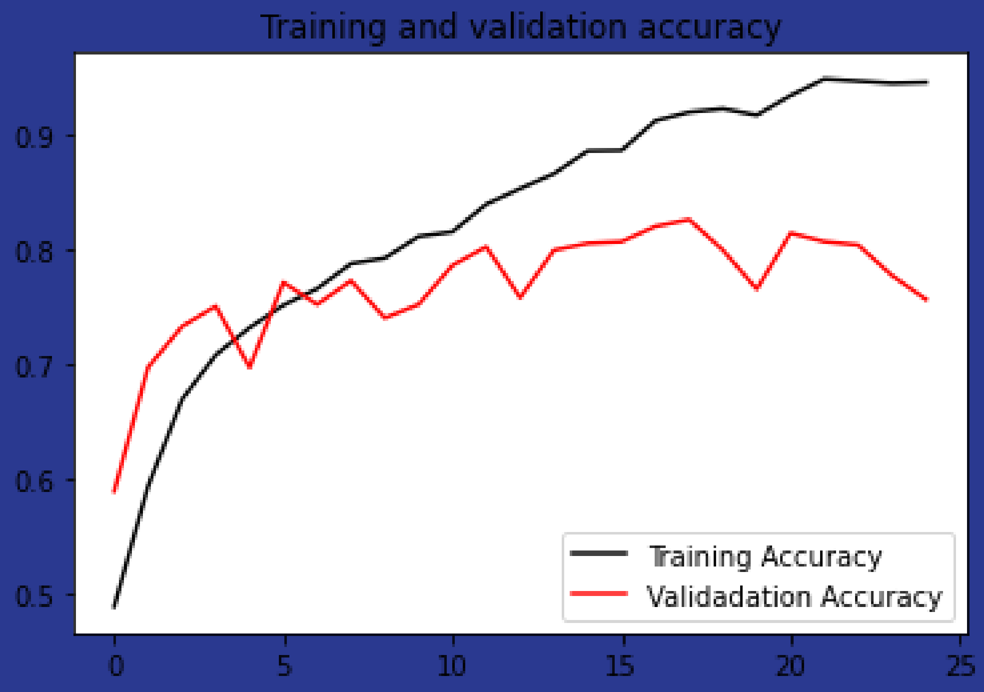
Model with data augmentation and drop-out layer

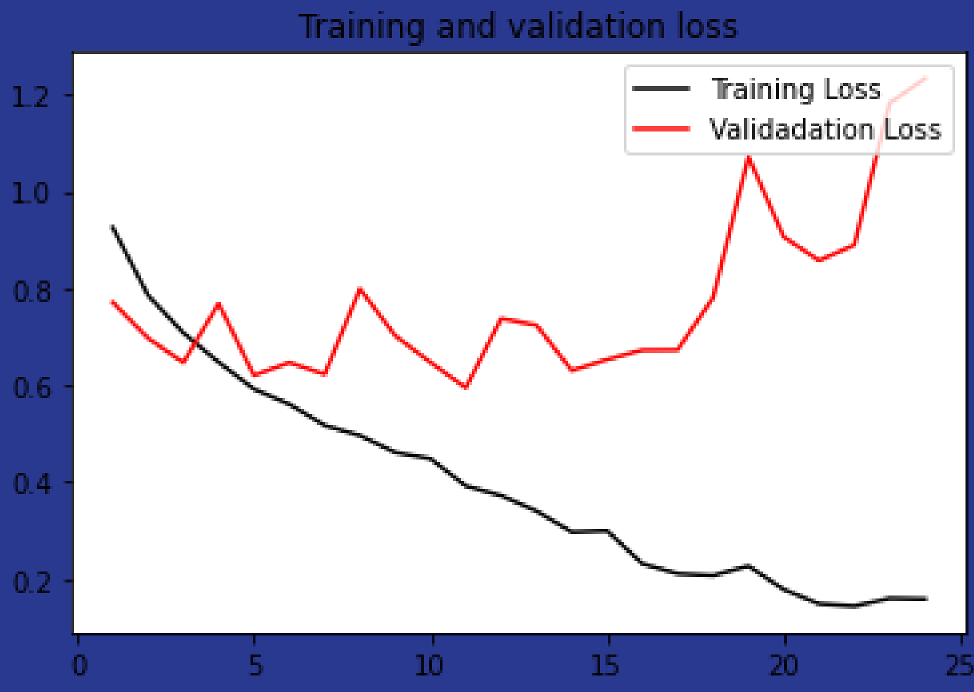




4 most populated „artists“ and Monet Stylised image

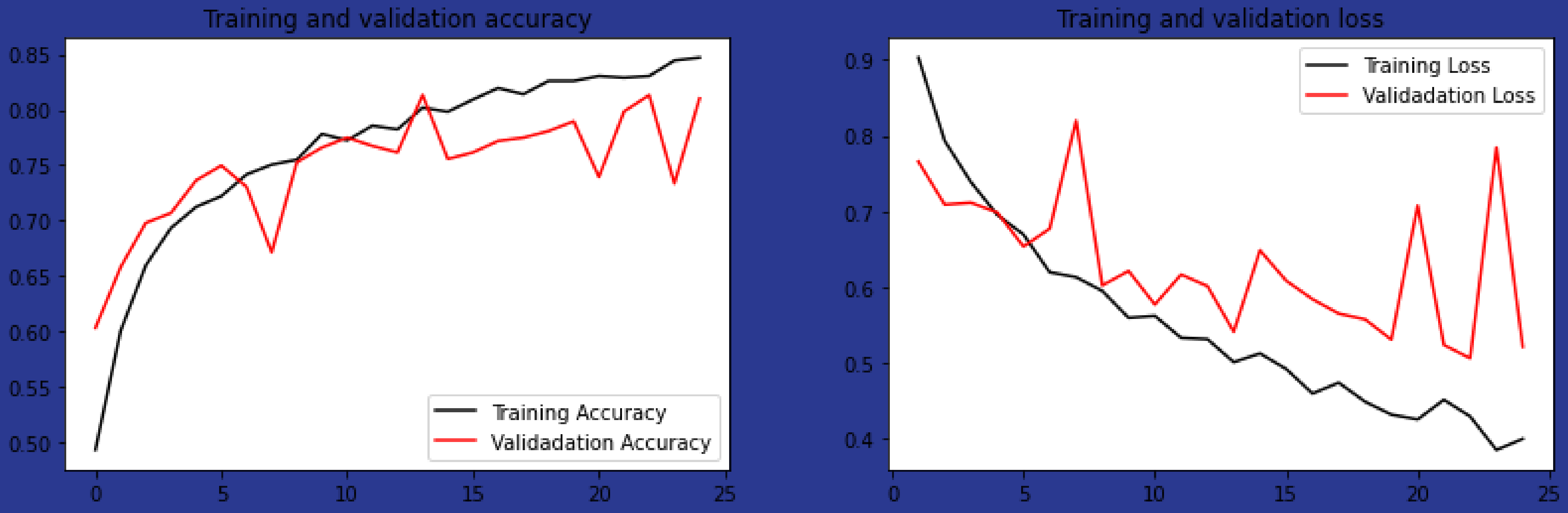
Model with no data augmentation and drop out layer





4 most populated „artists“ and Monet Stylised image

Model with data augmentation and no Drop out layer



Comparison

Number of artists	Max. training accuracy	Max. validation accuracy
50 + Monet stylised images	~ 50%	~ 50%
11	~ 75%	~ 70%
4+ Monet stylised images	~ 95%	~ 80%

Best setting

- 5 most populated „artists“
- Augmented Data
- No drop-out layer

Conclusions

Conclusions

- Most artists in the data-set have too few paintings
  –> fewer artists with more paintings increase the accuracy
  –> data augmentation increased the performance
- The model architecture does not have to be complex
- Overfitting can occur
- Adding the fake class increases the accuracy by ~5%
- Size of the data-set becomes an issue in Colab, when the resolution is too high (even with 5 classes).
- Random search for best hyperparameters was terminated by Colab due to the duration (10 sets, not in parallel, image size 128x128).

Outlook

- Evaluate more performance metrics
- Plot the outputs of the convolutional layers
- Use training and test set only
- Use less popular painters with more paintings
- Perhaps avoid painters with too similar style
- Prefer painters with a consistent style
- Search for a better peer model architecture
- Use transfer learning with a pre-trained model
- Parallelise the hyperparameter search
- Increase the image resolution to capture more features