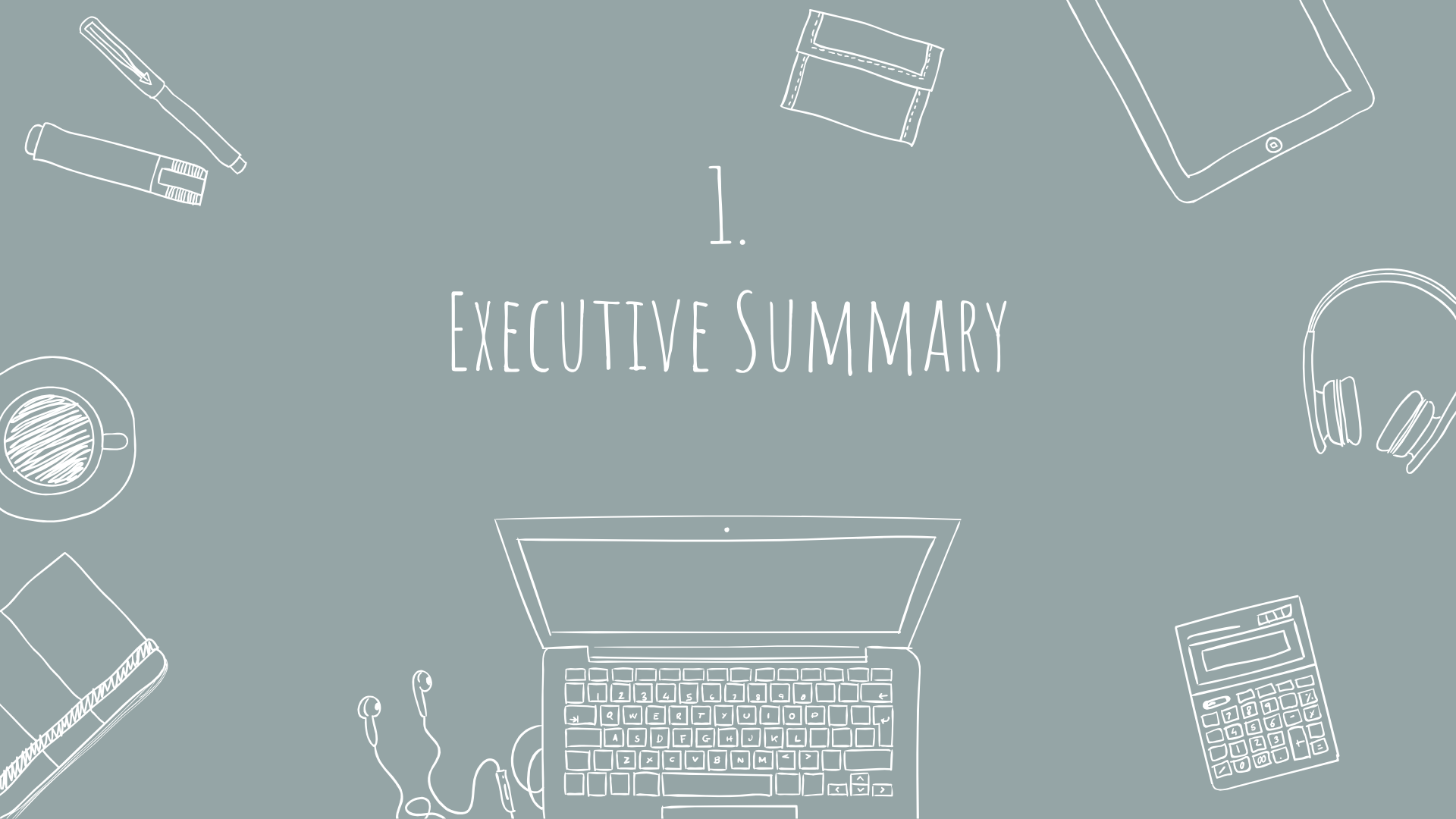
Machine Learning: Identify Artist

Team 5

Table of contents

1.

Executive Summary



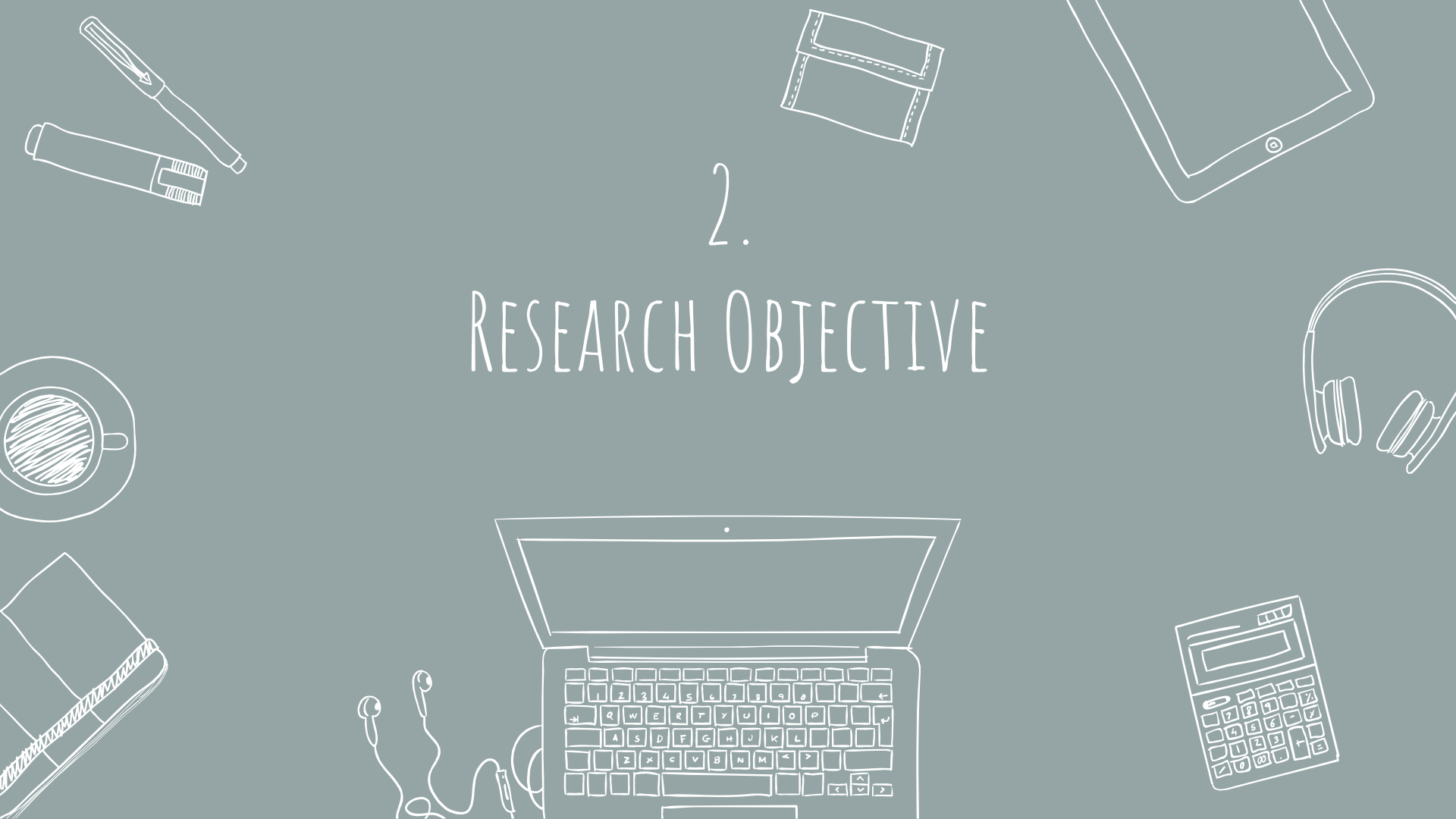
Executive Summary

Nowadays, due to the continuous exploration of previous artworks, galleries and art companies need to face huge artwork identification problems. We want to implement some machine learning algorithms to improve their working efficiency in this process.

In order to achieve this goal, we will implement some deep learning algorithms to identify artists. We would also explore using neural style transfer to create artistic images in the style of classic paintings.

2.

Research Objective

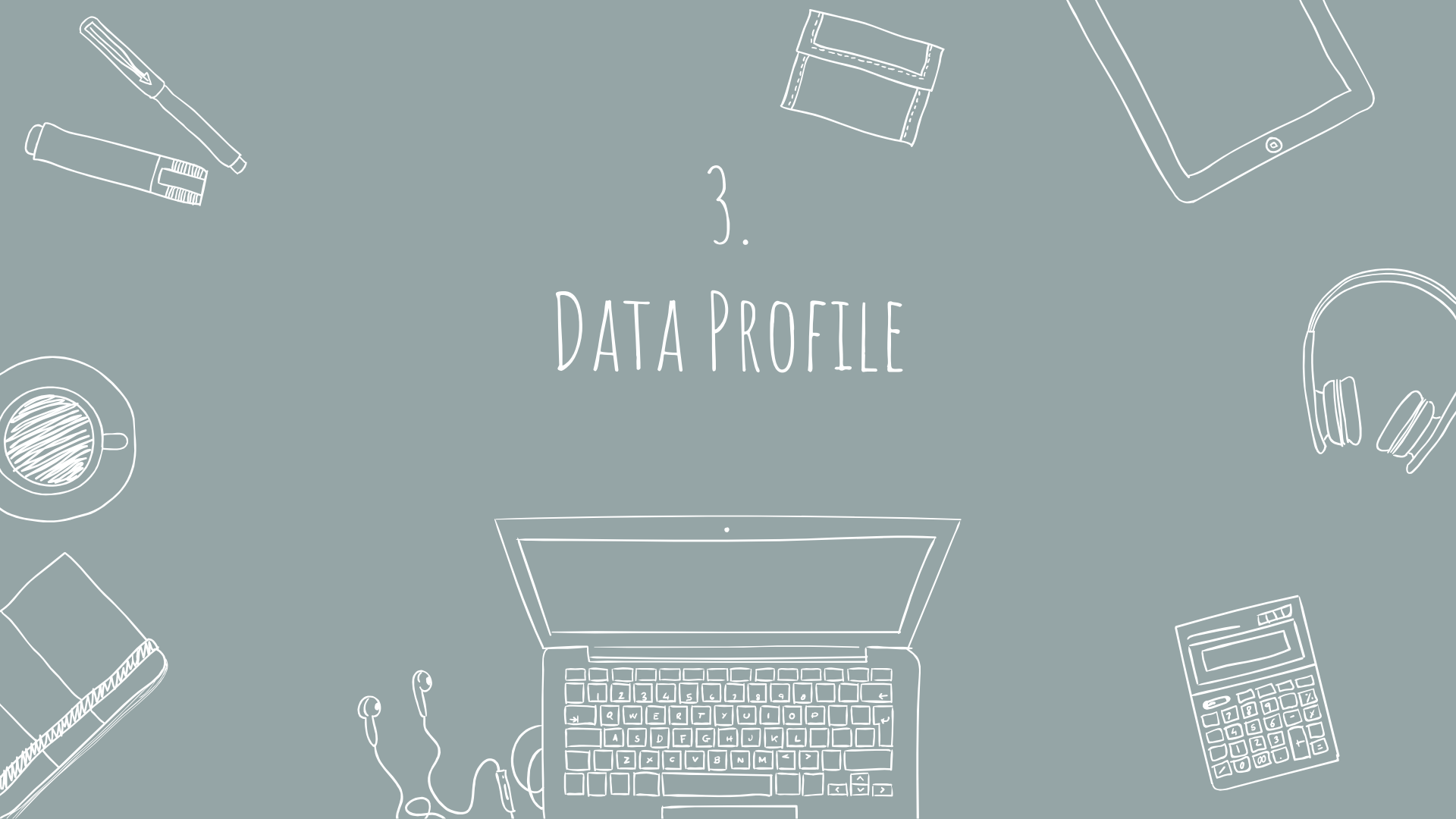


Research Objective

- Our goal is to identify the genre and artist's name based on the paintings with the neural network model.
- Recognize the artists looking the colors used and the geometric patterns inside the pictures.
- Identify the common characteristics of artists' styles in a specific time period.

3.

Data Profile



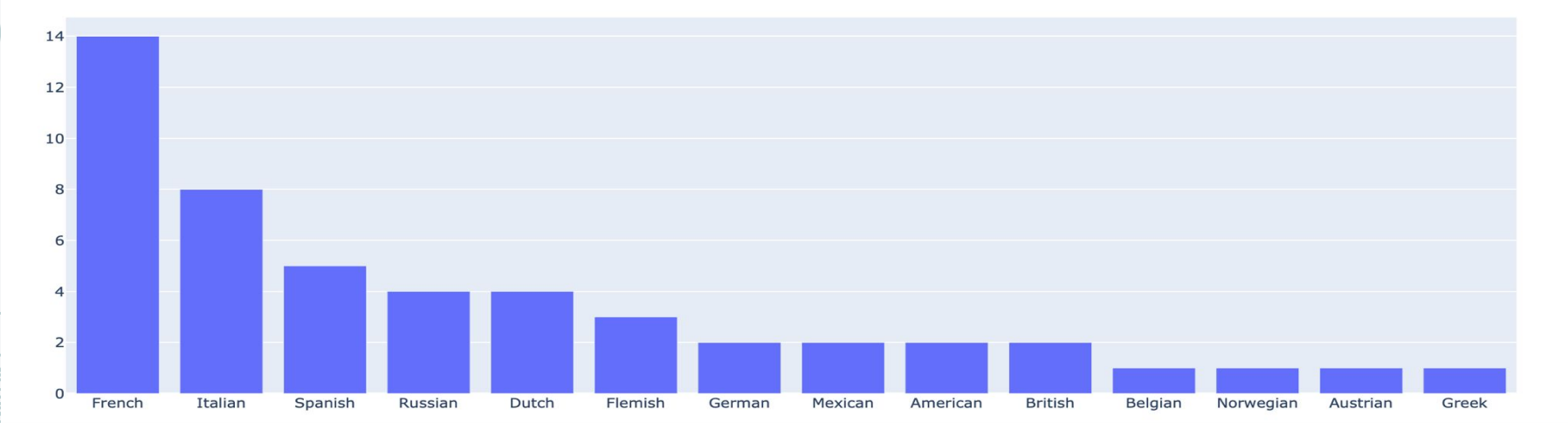
Exploratory Data Analysis (EDA)

```
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   id            50 non-null     int64
 1   name          50 non-null     object
 2   years         50 non-null     object
 3   genre         50 non-null     object
 4   nationality   50 non-null     object
 5   bio           50 non-null     object
 6   wikipedia     50 non-null     object
 7   paintings     50 non-null     int64
dtypes: int64(2), object(6)
```

This dataset contains artworks from 50 most influenced artists of all time and basic information.

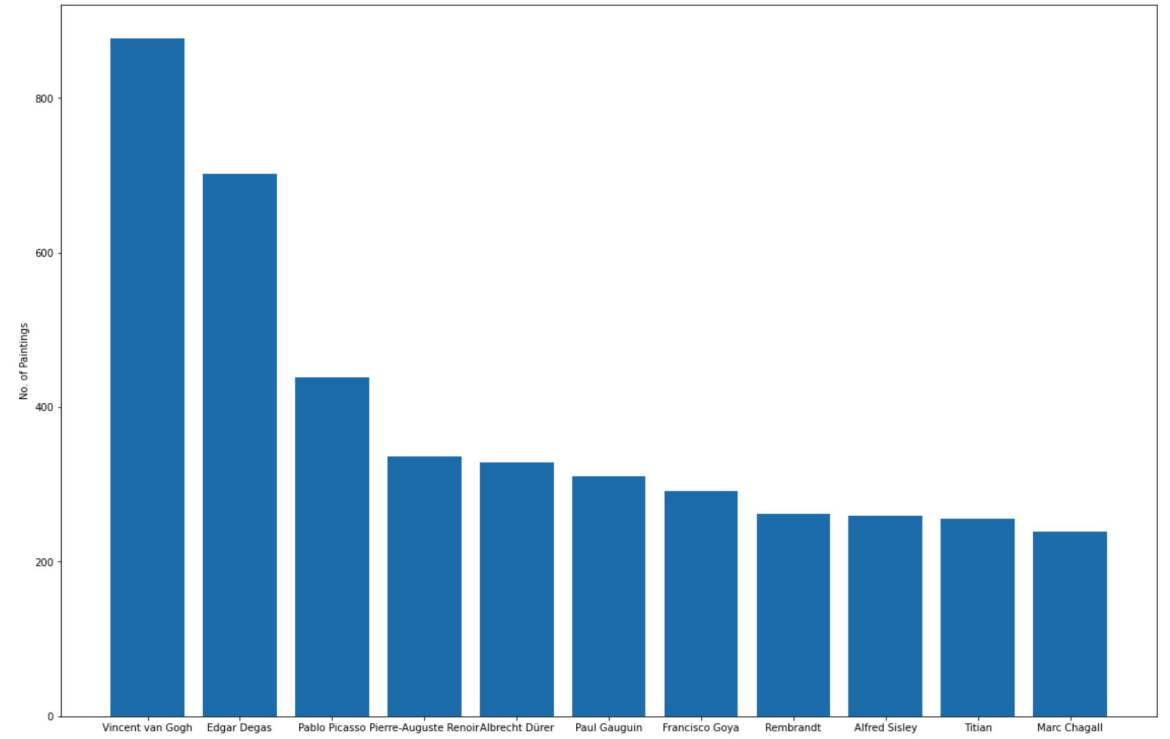
We have 8 variables in total, in which 2 of them are integers and 6 of them are objects

Exploratory Data Analysis (EDA)



According to the distribution of number of Nationality by artists, we can see that French has highest number of artists and Greek has lowest.

Exploratory Data Analysis (EDA)



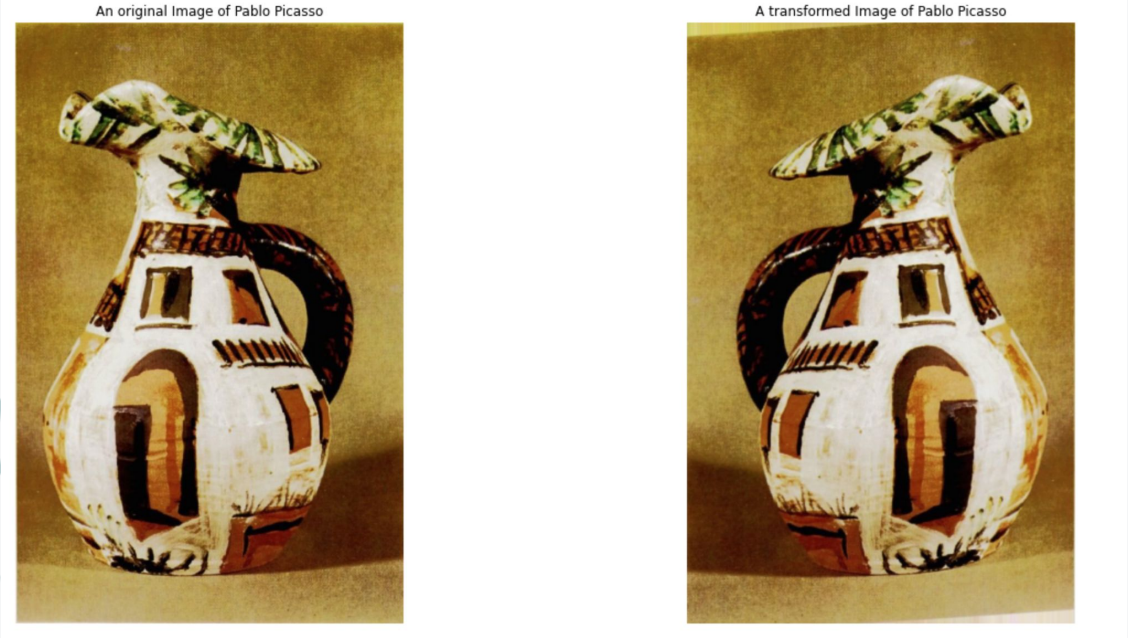
- There are paintings of 50 artists in the dataset. We would select artists with more than 400 paintings for better training. 3 artists are selected.
- The dataset is unbalanced.
- Van Gogh has 877 paintings which is the highest.

Data Profile - Visualizations of Art Work

4.

Data Preparation

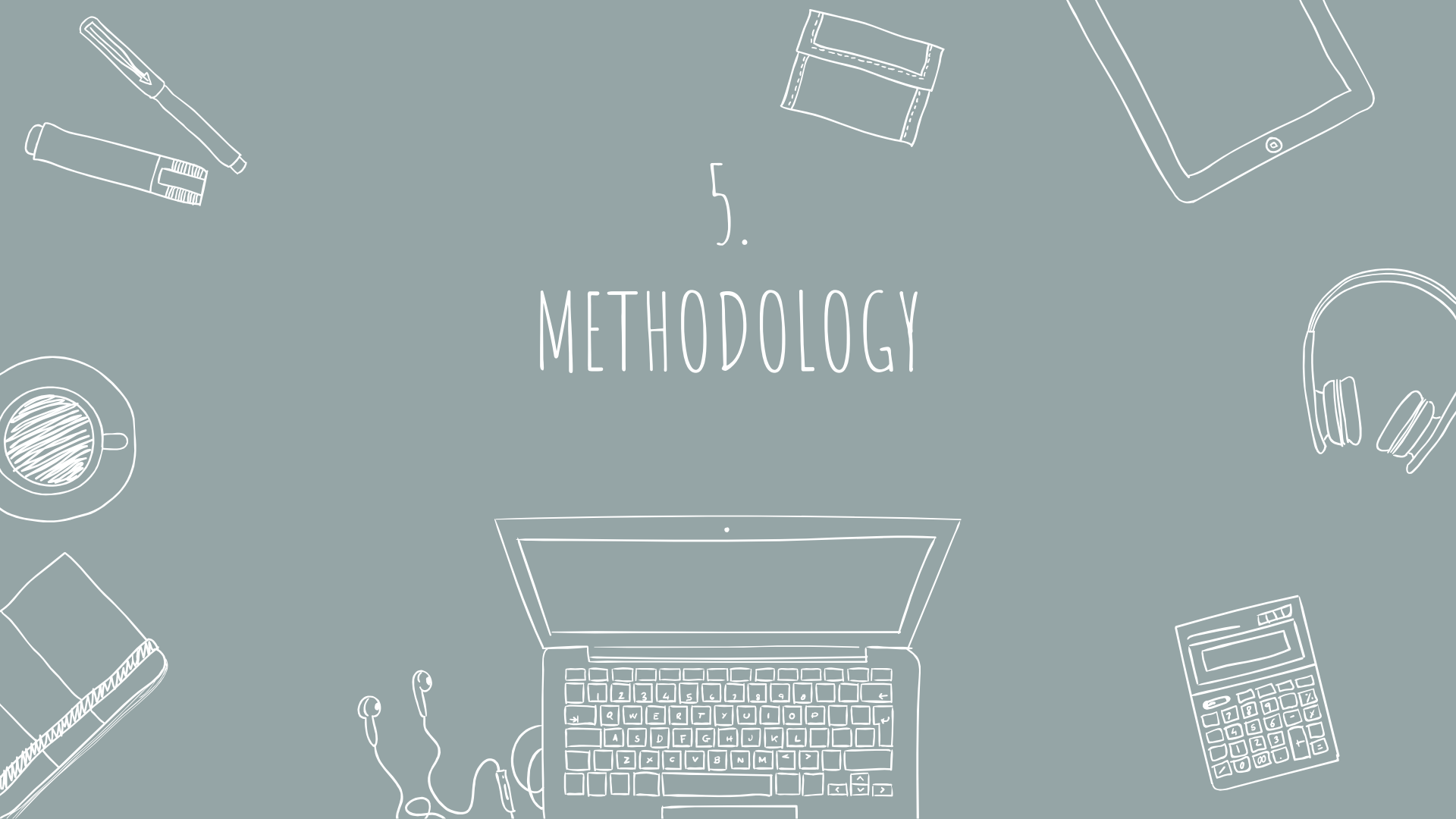
Data Augmentation



Data augmentation is useful to improve the performance and outcomes of machine learning models by forming new and different examples to train datasets. If the dataset in a machine learning model is rich and sufficient, the model performs better and more accurate.

5.

METHODOLOGY



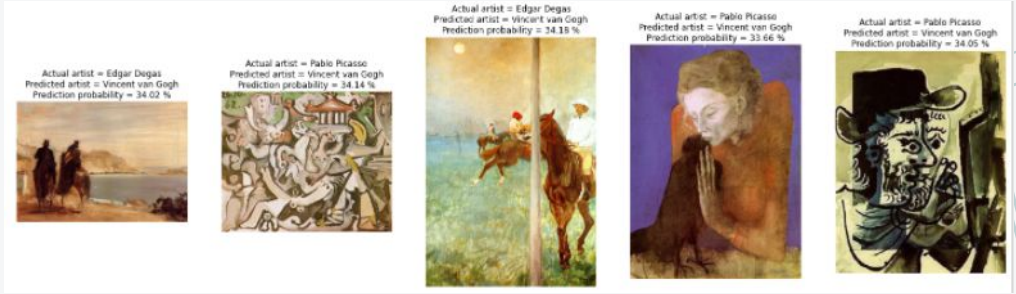
Methodology - Model 1: CNN MODEL

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_28 (Conv2D)           (None, 224, 224, 32)      11648
_________________________________________________________________
batch_normalization_5 (Batch (None, 224, 224, 32)      128
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 112, 112, 32)      0
_________________________________________________________________
conv2d_29 (Conv2D)           (None, 112, 112, 64)      100416
_________________________________________________________________
batch_normalization_6 (Batch (None, 112, 112, 64)      256
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 56, 56, 64)        0
_________________________________________________________________
conv2d_30 (Conv2D)           (None, 56, 56, 128)       204928
_________________________________________________________________
batch_normalization_7 (Batch (None, 56, 56, 128)       512
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 28, 28, 128)       0
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 28, 28, 256)       295168
_________________________________________________________________
batch_normalization_8 (Batch (None, 28, 28, 256)       1024
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 14, 14, 256)       0
_________________________________________________________________
dropout_2 (Dropout)          (None, 14, 14, 256)       0
_________________________________________________________________
flatten_3 (Flatten)          (None, 50176)             0
_________________________________________________________________
dense_9 (Dense)              (None, 256)               12845312
_________________________________________________________________
dense_10 (Dense)             (None, 64)                16448
_________________________________________________________________
dense_11 (Dense)             (None, 3)                 195
=================================================================
Total params: 13,476,035
Trainable params: 13,475,075
Non-trainable params: 960
```

- **CNN model:**
  - 4 Convolutional Layers
    - Kernel Size: 11*11, 7*7, 5*5, 3*3
  - 4 Pool layers
- **Prediction accuracy :**
  - Training data: 0.3478
  - Validation data: 0.3482

Methodology - Model 2: CNN MODEL

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_32 (Conv2D)           (None, 224, 224, 32)      896
_________________________________________________________________
conv2d_33 (Conv2D)           (None, 224, 224, 32)      9248
_________________________________________________________________
conv2d_34 (Conv2D)           (None, 224, 224, 32)      9248
_________________________________________________________________
conv2d_35 (Conv2D)           (None, 224, 224, 32)      9248
_________________________________________________________________
conv2d_36 (Conv2D)           (None, 224, 224, 32)      9248
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 112, 112, 32)      0
_________________________________________________________________
conv2d_37 (Conv2D)           (None, 112, 112, 64)      18496
_________________________________________________________________
conv2d_38 (Conv2D)           (None, 112, 112, 64)      36928
_________________________________________________________________
conv2d_39 (Conv2D)           (None, 112, 112, 64)      36928
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 56, 56, 64)        0
_________________________________________________________________
conv2d_40 (Conv2D)           (None, 56, 56, 128)       73856
_________________________________________________________________
conv2d_41 (Conv2D)           (None, 56, 56, 128)       147584
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 28, 28, 128)       0
_________________________________________________________________
conv2d_42 (Conv2D)           (None, 28, 28, 256)       295168
_________________________________________________________________
batch_normalization_9 (Batch (None, 28, 28, 256)       1024
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 14, 14, 256)       0
_________________________________________________________________
dropout_3 (Dropout)          (None, 14, 14, 256)       0
_________________________________________________________________
flatten_4 (Flatten)          (None, 50176)             0
_________________________________________________________________
dense_12 (Dense)             (None, 256)               12845312
_________________________________________________________________
dense_13 (Dense)             (None, 64)                16448
_________________________________________________________________
dense_14 (Dense)             (None, 3)                 195
=================================================================
Total params: 13,509,827
Trainable params: 13,509,315
Non-trainable params: 512
_________________________________________________________________
```



- **CNN model:**
  - 11 Convolutional Layers
    - Kernel Size: 3*3
  - 4 Pool layers
- **Prediction accuracy :**
  - Training data: 0.4337
  - Validation data: 0.4278

Methodology - Model 3: VGG

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_43 (Conv2D)           (None, 224, 224, 64)      1792
_________________________________________________________________
conv2d_44 (Conv2D)           (None, 224, 224, 64)      36928
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 112, 112, 64)      0
_________________________________________________________________
conv2d_45 (Conv2D)           (None, 112, 112, 128)     73856
_________________________________________________________________
conv2d_46 (Conv2D)           (None, 112, 112, 128)     147584
_________________________________________________________________
max_pooling2d_22 (MaxPooling (None, 56, 56, 128)       0
_________________________________________________________________
conv2d_47 (Conv2D)           (None, 56, 56, 256)       295168
_________________________________________________________________
conv2d_48 (Conv2D)           (None, 56, 56, 256)       590080
_________________________________________________________________
conv2d_49 (Conv2D)           (None, 56, 56, 256)       590080
_________________________________________________________________
max_pooling2d_23 (MaxPooling (None, 28, 28, 256)       0
_________________________________________________________________
conv2d_50 (Conv2D)           (None, 28, 28, 512)       1180160
_________________________________________________________________
conv2d_51 (Conv2D)           (None, 28, 28, 512)       2359808
_________________________________________________________________
conv2d_52 (Conv2D)           (None, 28, 28, 512)       2359808
_________________________________________________________________
max_pooling2d_24 (MaxPooling (None, 14, 14, 512)       0
_________________________________________________________________
conv2d_53 (Conv2D)           (None, 14, 14, 512)       2359808
_________________________________________________________________
conv2d_54 (Conv2D)           (None, 14, 14, 512)       2359808
_________________________________________________________________
conv2d_55 (Conv2D)           (None, 14, 14, 512)       2359808
_________________________________________________________________
max_pooling2d_25 (MaxPooling (None, 7, 7, 512)         0
_________________________________________________________________
flatten_5 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_15 (Dense)             (None, 4096)              102764544
_________________________________________________________________
dense_16 (Dense)             (None, 4096)              16781312
_________________________________________________________________
dense_17 (Dense)             (None, 3)                 12291
=================================================================
Total params: 134,272,835
Trainable params: 134,272,835
Non-trainable params: 0
_________________________________________________________________
```

- **VGG-16 Structure:**
  - 13 Convolutional Layers + 5 Pool layers
- **Prediction accuracy :**
  - Training data: 0.4344
  - Validation data: 0.4353

Methodology - Model 3: VGG Pretrained

```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, None, None, 512)   14714688
_________________________________________________________________
global_average_pooling2d_3 ( (None, 512)               0
_________________________________________________________________
dense_32 (Dense)             (None, 512)               262656
_________________________________________________________________
dense_33 (Dense)             (None, 3)                 1539
=================================================================
Total params: 14,978,883
Trainable params: 14,978,883
Non-trainable params: 0
_________________________________________________________________
```

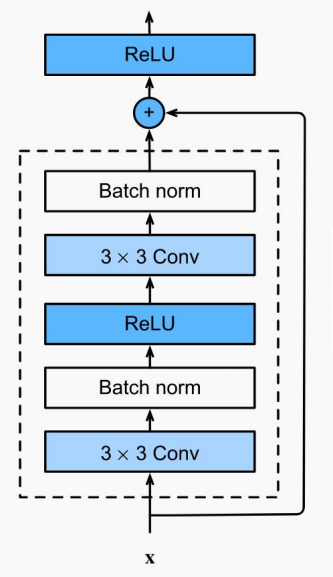
- None Convolutional Layers in the pre-trained VGG-16 model is trainable
  - Training data: 0.7123
  - Validation data: 0.6915
- The last 3 Convolutional Layers are trainable
  - Training data: 0.7228
  - Validation data: 0.6816

Methodology - Model 4: ResNet50

```
# Add layers at the end
X = base_model.output
X = Flatten()(X)

X = Dense(512, kernel_initializer='he_uniform')(X)
X = Dropout(0.5)(X)
X = BatchNormalization()(X)
X = Activation('relu')(X)

X = Dense(16, kernel_initializer='he_uniform')(X)
X = Dropout(0.5)(X)
X = BatchNormalization()(X)
X = Activation('relu')(X)

output = Dense(n_classes, activation='softmax')(X)

model4 = Model(inputs=base_model.input, outputs=output)
```

- **ResNet50**:
  - Weights: imagenet
- **Prediction accuracy** :
  - Training data: 0.999
  - Validation data: 0.923

Methodology - Model 4: ResNet50



A Resnet Block

x
identity

Traditional CNNs

- There is a maximum threshold for depth

Resnet

- Add the output from the previous layer to the layer ahead
- Train much deeper networks without hurting the model

6.

Lessons learned

Lessons

Layer (type)	Output Shape	Param #
conv2d_72 (Conv2D)	(None, 222, 222, 32)	896
batch_normalization_14 (Batc	(None, 222, 222, 32)	128
max_pooling2d_31 (MaxPooling	(None, 111, 111, 32)	0
conv2d_73 (Conv2D)	(None, 109, 109, 32)	9248
batch_normalization_15 (Batc	(None, 109, 109, 32)	128
max_pooling2d_32 (MaxPooling	(None, 54, 54, 32)	0
flatten_7 (Flatten)	(None, 93312)	0
dense_21 (Dense)	(None, 128)	11944064
dropout_6 (Dropout)	(None, 128)	0
dense_22 (Dense)	(None, 1)	129

Total params: 11,954,593
Trainable params: 11,954,465
Non-trainable params: 128

Simple CNN
Training Accuracy: 0.665
Validation Accuracy: 0.646

- ResNet50 performs the best among 4 models
- With regard to the identification of the artist, the model performs better with shallow layers. We think the main reason is that for each class, it would be better to have more than thousands of paintings per class to train deeper layers. Although we set the threshold to 400, the number is still not large enough.

7.

Recommendations

Recommendations

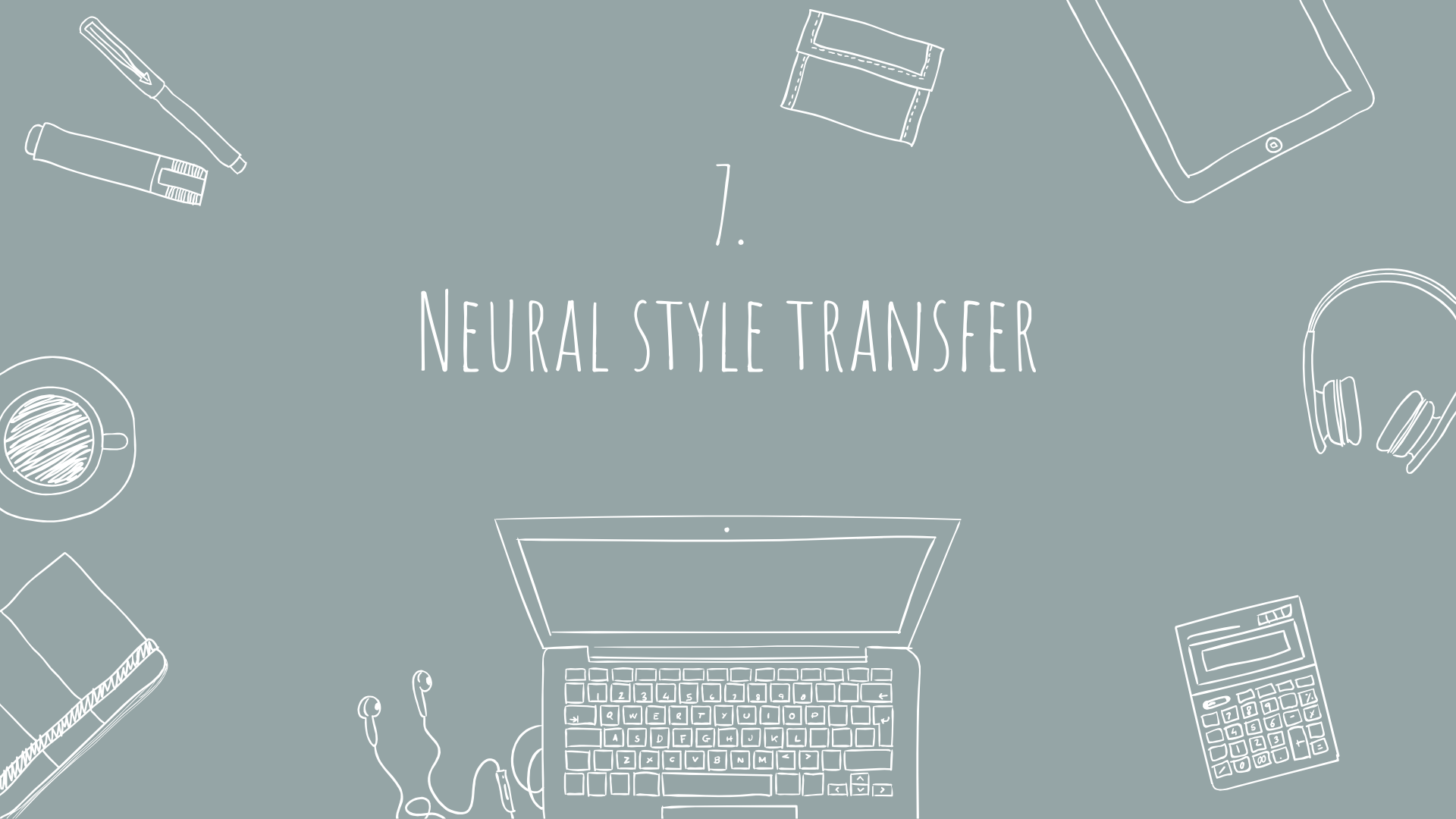
Further Improvement

- We envision this project being helpful for identifying artists with data from not only in Europe, but all over the world. In order to do that, we will include more artists and their artworks to the dataset.

- Pre-trained model can perform better due to the limitation number of artworks for each artist. We are more likely to introduce art industry to use pre-trained ResNet model.

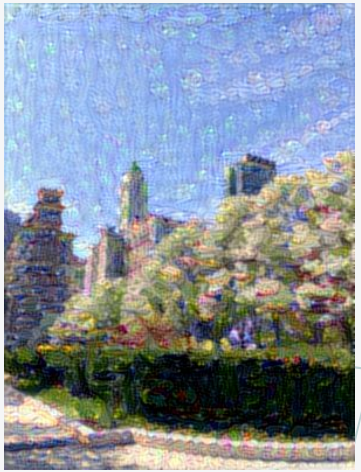
7.

Neural style transfer



What is neural style transfer?

Claude Monet

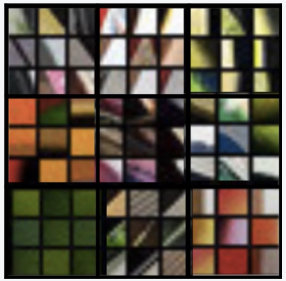


Base Image (Content)

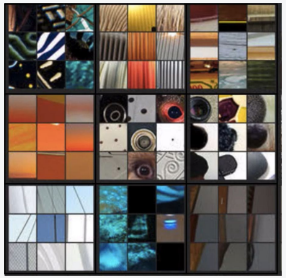
Style Image

Generated Image

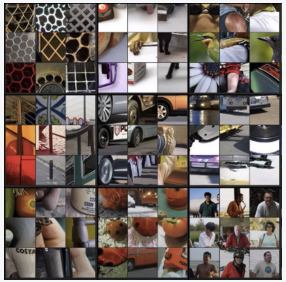
Visualise Deep Layers



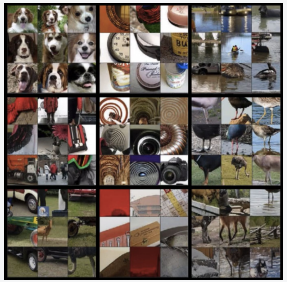
Layer 1



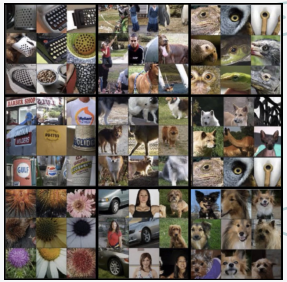
Layer 2



Layer 3

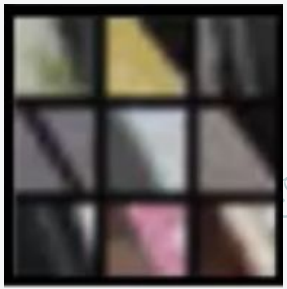


Layer 4



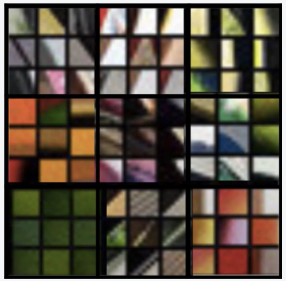
Layer 5

- Pick a filter in one layer (3 * 3)
- Nine image patches that maximize the filter's activation.

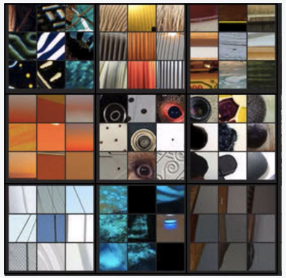


Layer 1

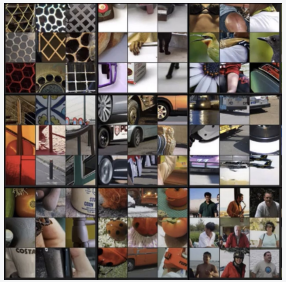
Visualise Deep Layers



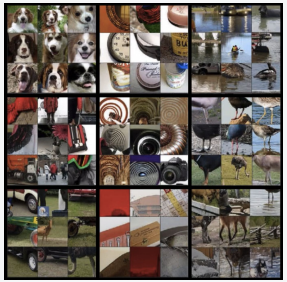
Layer 1



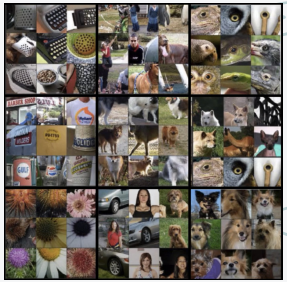
Layer 2



Layer 3



Layer 4

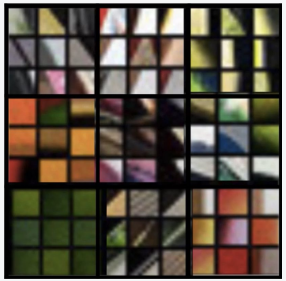


Layer 5

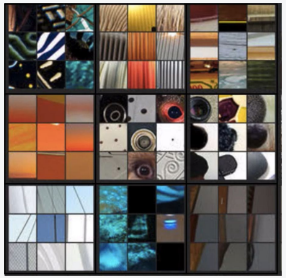
- Pick a filter in one layer (3 * 3)
- Nine image patches that maximize the filter's activation.
- Earlier layers learn more fundamental features such as lines and shapes
- Latter layers learn more complex features (content).

Layer 5

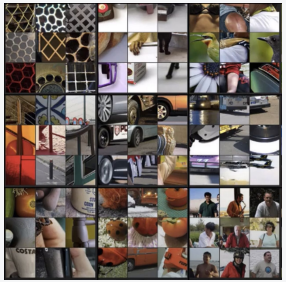
Implement Neural Style Transfer



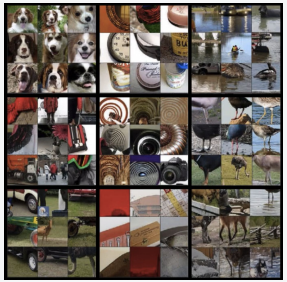
Layer 1



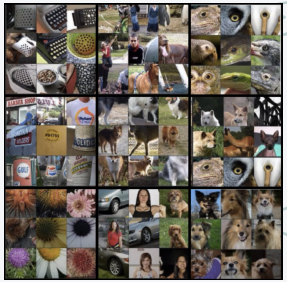
Layer 2



Layer 3



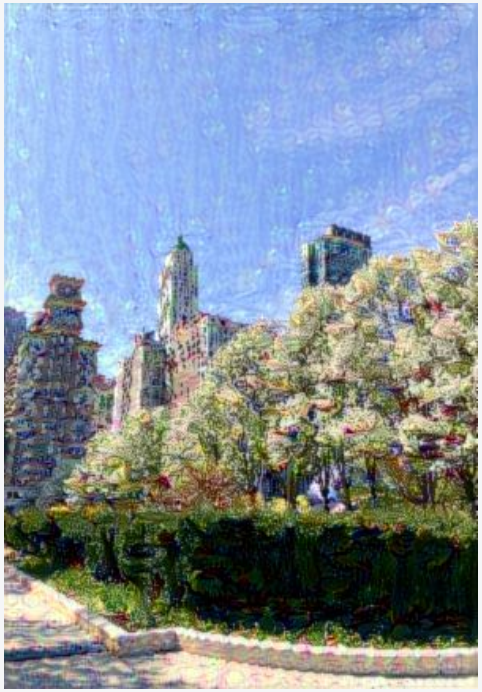
Layer 4



Layer 5

- Use pre-trained neural networks (VGG19)
- Backpropagate
- So our output image will have:
  - Content close to the base image
  - Style close to style image

Original Image

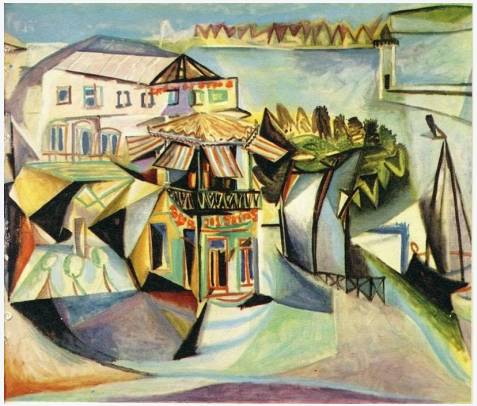


100 Iterations

1000 Iterations

Style Image

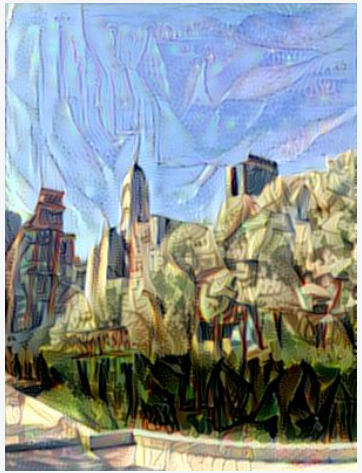
Vincent van Gogh



Pablo Picasso

Salvador Dali

Generated Image



Thanks!

Any questions?