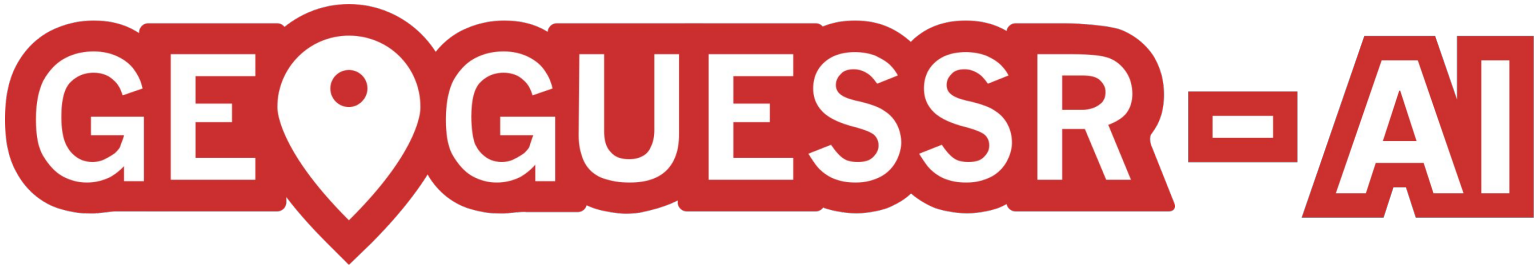


www.geoguessr.com

Philipp Ollmann, Lukas Rockenschaub, Sebastian Weidlinger, Michael Zauner

GeoGuessr

- Popular Online Game
- Guess location of “random” Street View image
- Really fun but also quite difficult

Initial Idea for Project:
Guess location based on GeoGuessr Image

Try to base recognition on human player approach:

- Position of sun
- Road markings
- Position of Street View car (left- or right-hand traffic)
- Vegetation

Decided against, this idea and approaches because it is really difficult and extensive pre-processing required

Our more realistic Idea:
Distinguish between Europe and Asia

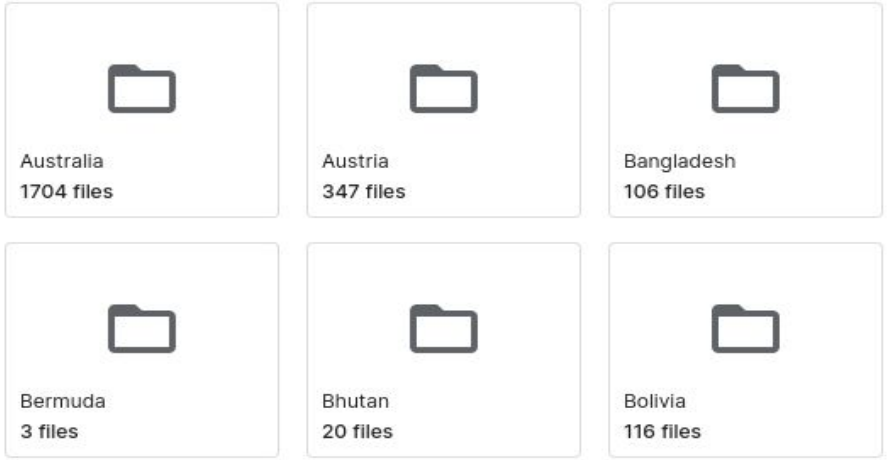
Goal

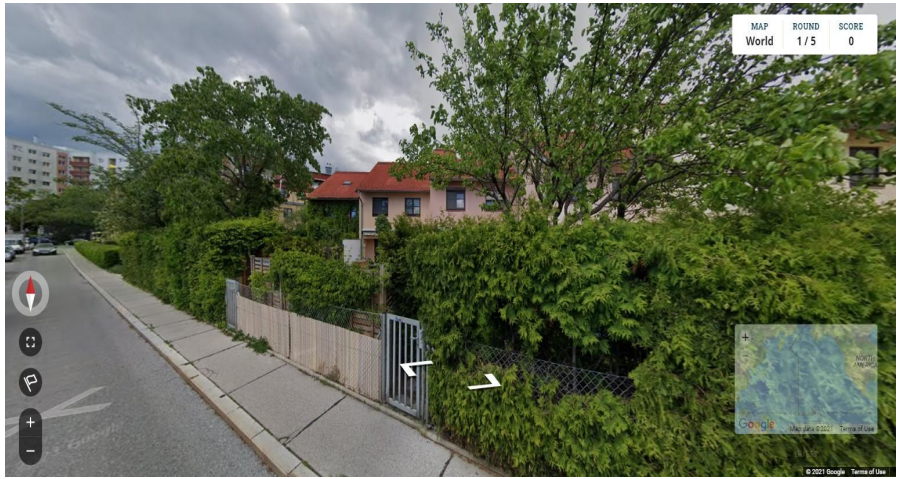
- Successfully distinguish between Asia and Europe countries
- Better accuracy than dummy classifier (0.5)
- Evaluate different model architectures and model setups

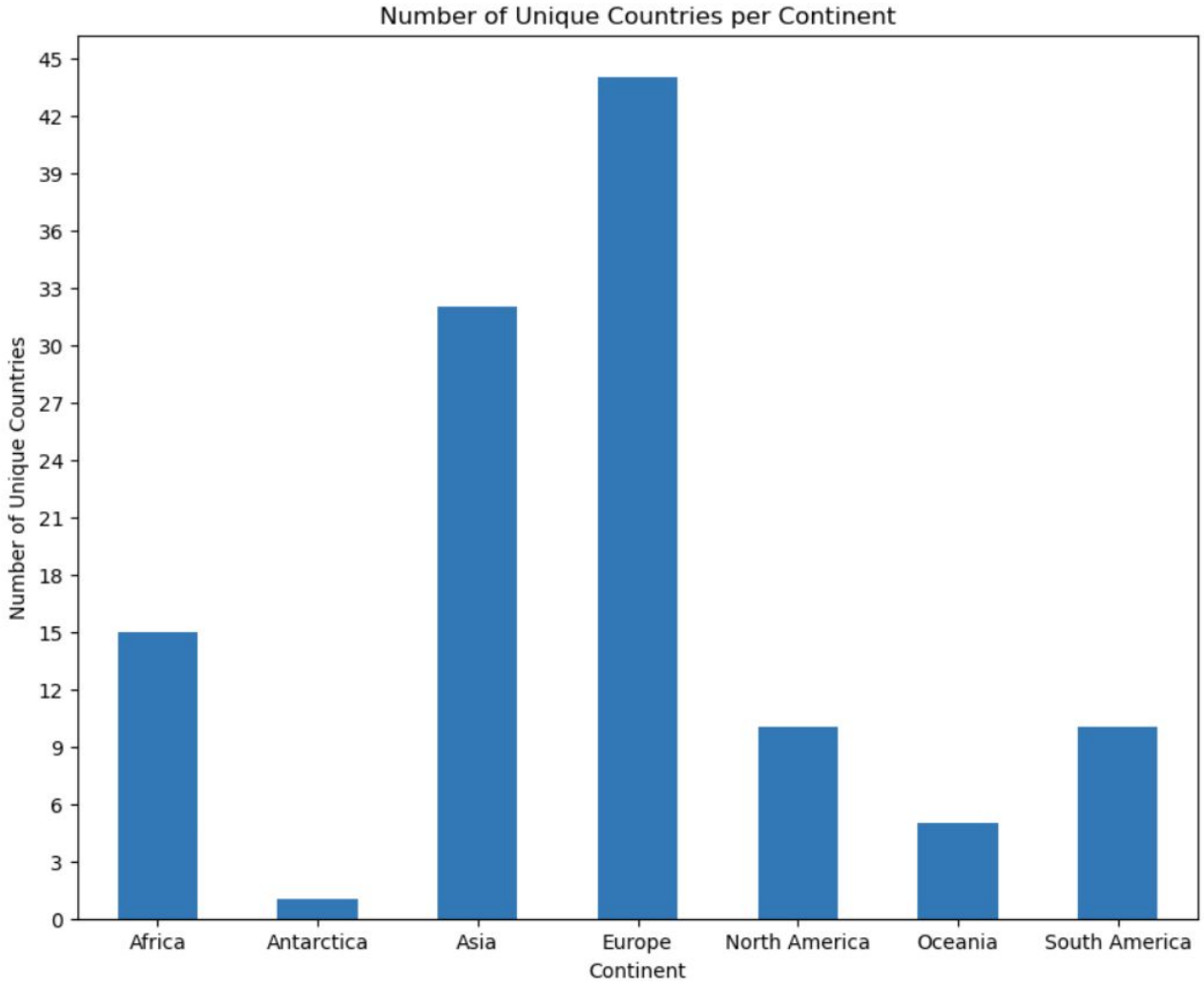
Dataset and Pre-Processing

Dataset

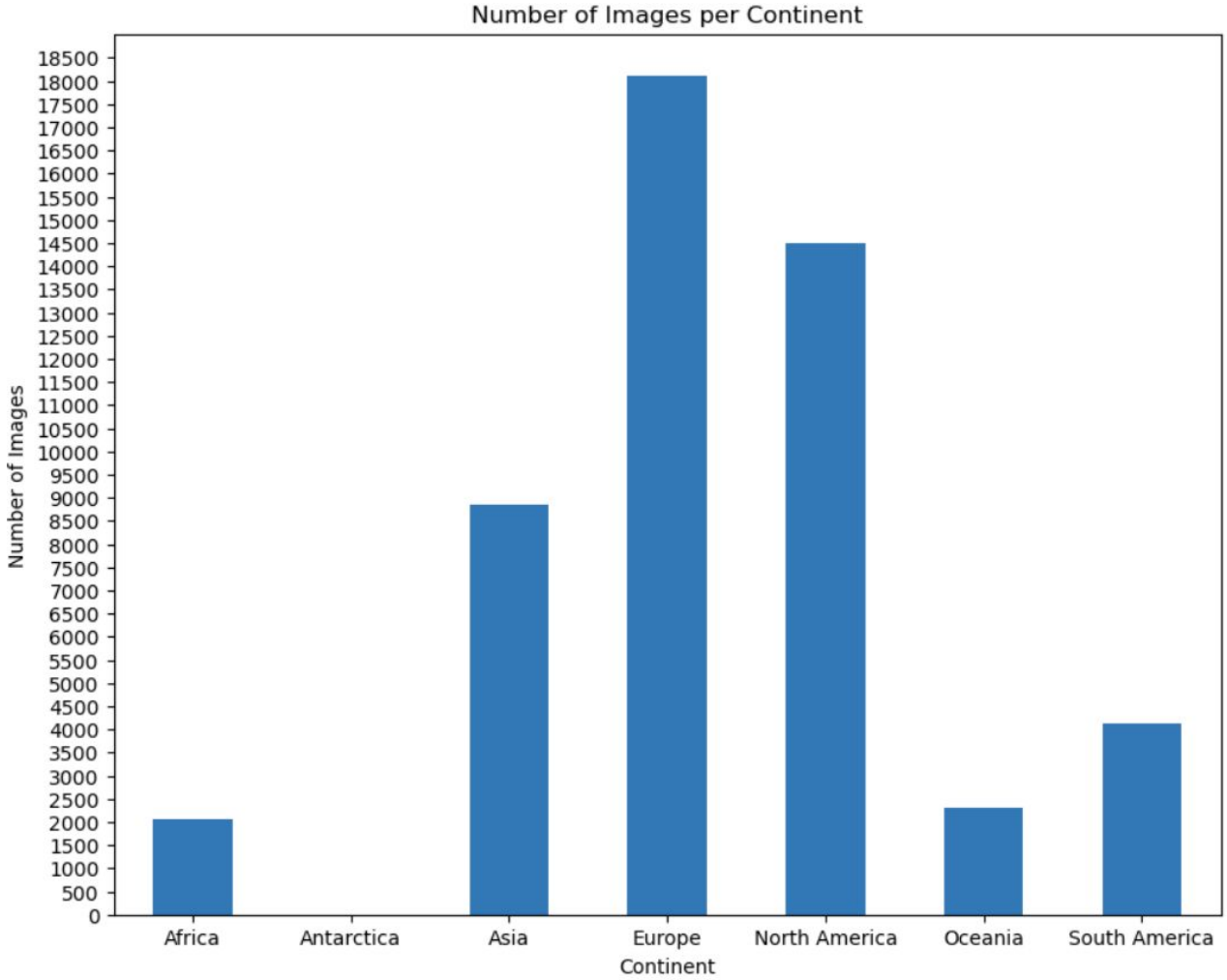
- 50k Images
  [GeoLocation - Geoguessr Images (50K) (kaggle.com)]
- Split into individual countries
- Data scraped from GeoGuessr
- Strong class imbalance, since amount of Street View images is highly dependent on country






Number of Unique Countries per Continent
Number of Unique Countries
0
3
6
9
12
15
18
21
24
27
30
33
36
39
42
45
Africa
Antarctica
Asia
Europe
North America
Oceania
South America
Continent

Max. amount we used for training



Data Preparation

1. Group countries into continents
2. Resize images to 224x224
3. (Non) Crop
4. Split into three sets
    a. 70% Train
    b. 20% Validation
    c. 10% Test

Sample non cropped

Sample center cropped

Simple. Flexible. Powerful.

Model Architectures

A Note on Preprocessing

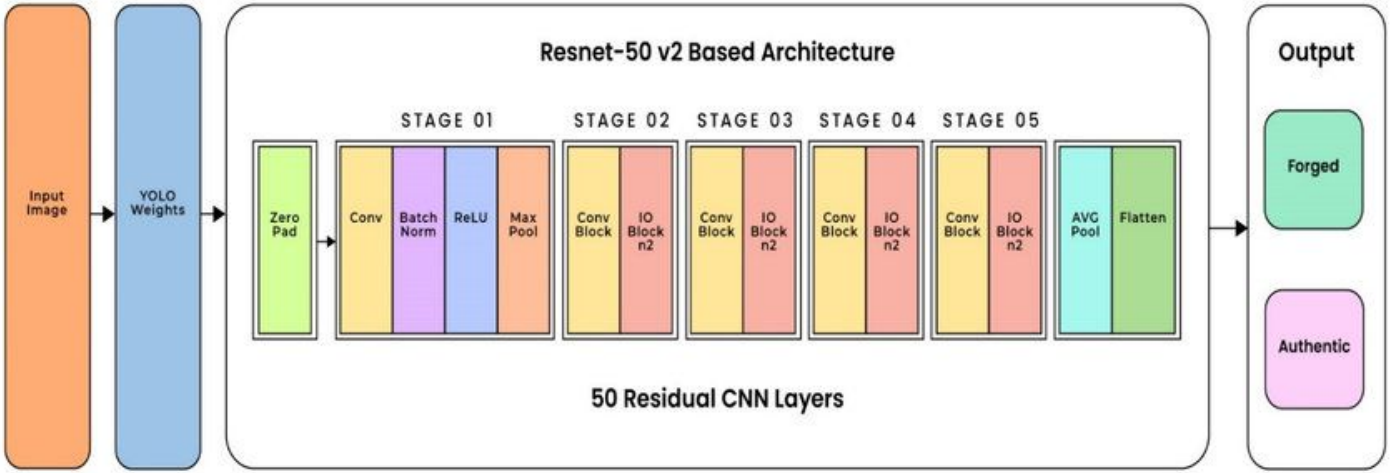
- Each model architecture required its own preprocessing
  - Keras provides a function, mostly scaling between [0,1] or [-1,1]
  - sometimes includes data centering

- Data augmentation could be performed in augmentation layers
  - Horizontal flips, stretching, rotating etc.

Note: each Keras Application expects a specific kind of input preprocessing. For ResNet, call `keras.applications.resnet_v2.preprocess_input` on your inputs before passing them to the model. `resnet_v2.preprocess_input` will scale input pixels between -1 and 1.

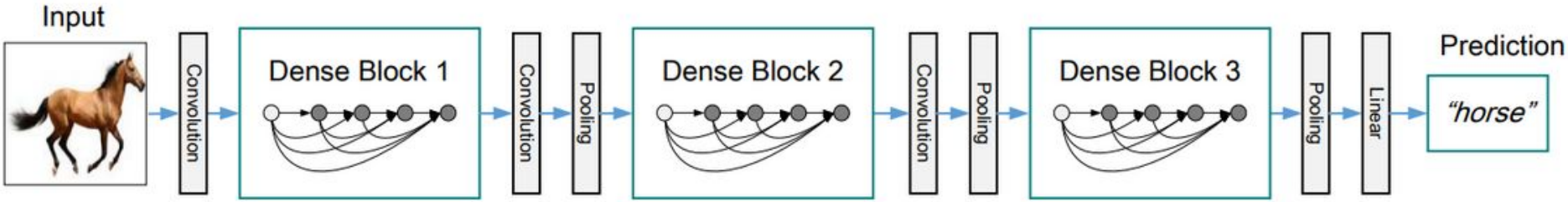
ResNet50(v2)

- Residual Network with 50 layers
- CNN Architecture
- reduce vanishing gradient problem in deep neural networks by using residual connections or skip connections



DenseNet121 - Densely Connected Convolutional Network

- Connects each layer to every other layer in a feed-forward fashion
- Each layer uses feature-maps of all preceding layers as inputs:
  - consequently its own feature-maps are used as inputs into all subsequent layers.
- **Advantages:**
  - strong feature propagation and feature reuse:
    - reduces the total number of parameters

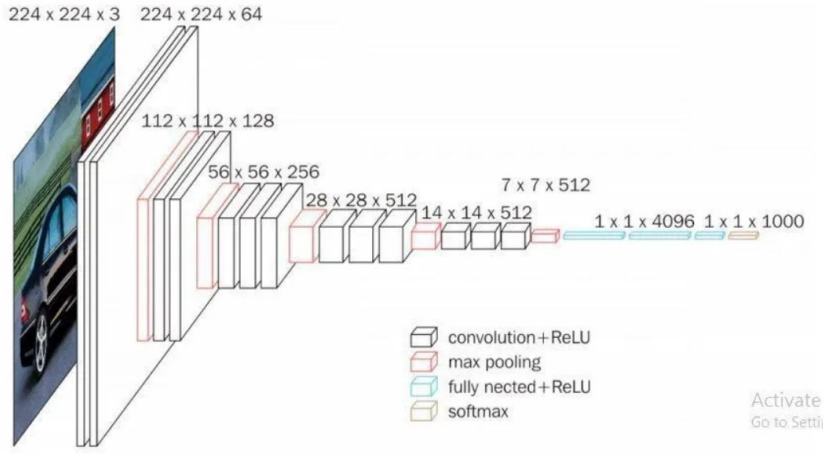


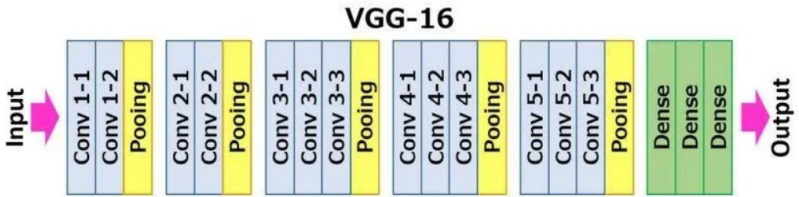
VGG16 - Very Deep Convolutional Network

- By the Visual Geometry Group, Oxford
- 16 layers that have trainable weights,

  21 total
- 13 convolutional layers
- small 3x3 convolution kernel

  to avoid too many parameters





Training

Training

- Use 8000 images per continent
- Utilized EarlyStopper and Checkpoints
- Lots of trial and error in terms of applying learning rates and regularization

Training duration

CPU based training (DenseNet121) time roughly 300 seconds per epoch

Tried to use GPU for learning on two local machines -> could not get it to work

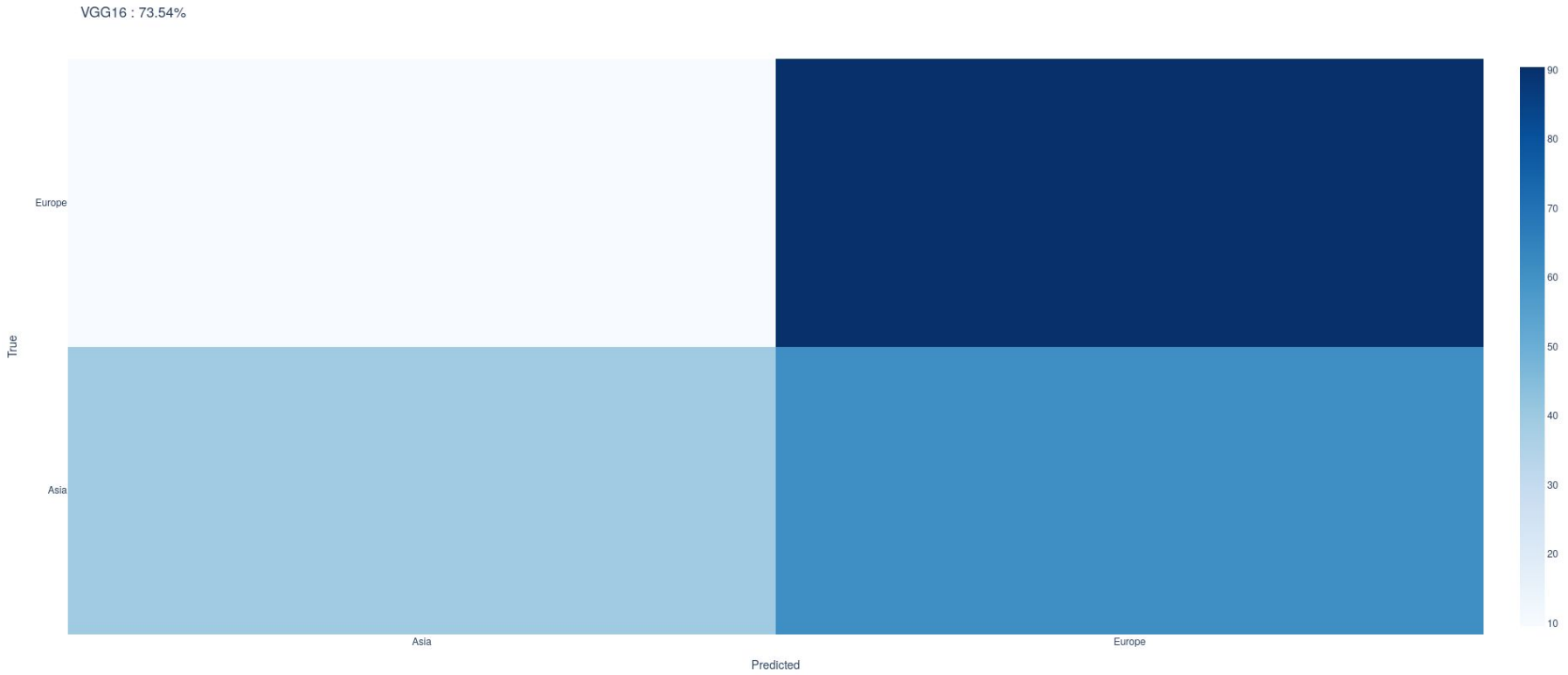
Bought Google Colab Pro in hope to improve training time and make Collaboration easier

Evaluation

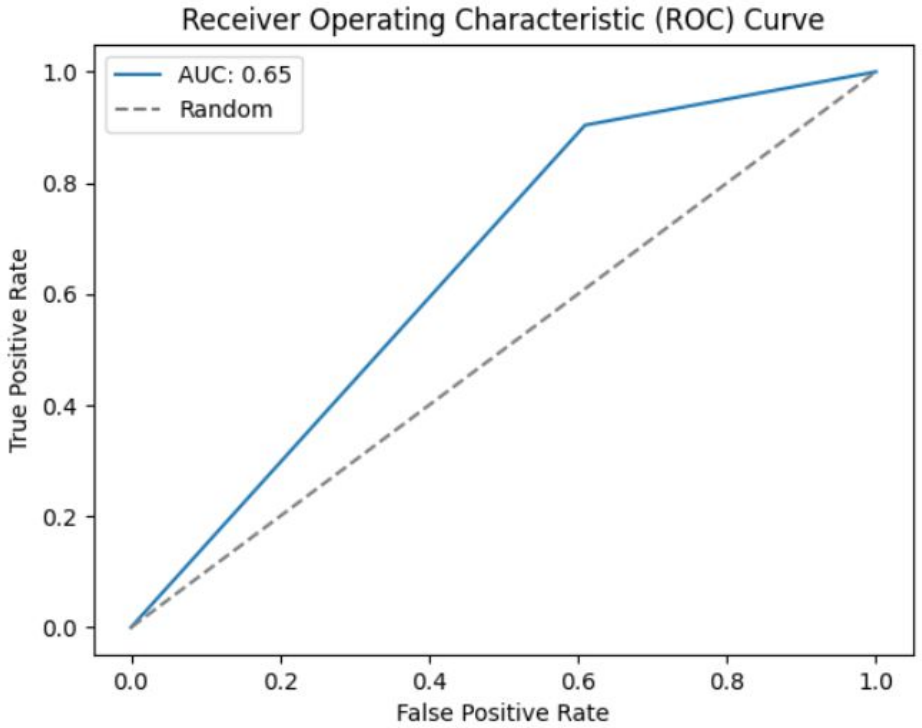
Dense121 - center cropped images


```
Epoch 11/30
350/350 [==============================] - ETA: 0s - loss: 0.4756 - accuracy: 0.7760
Epoch 00011: val_loss improved from 0.52002 to 0.48650, saving model to models/dense2_model_best_adam.hdf5
350/350 [==============================] - 25s 73ms/step - loss: 0.4756 - accuracy: 0.7760 - val_loss: 0.4865 - val_accuracy: 0.7716
```

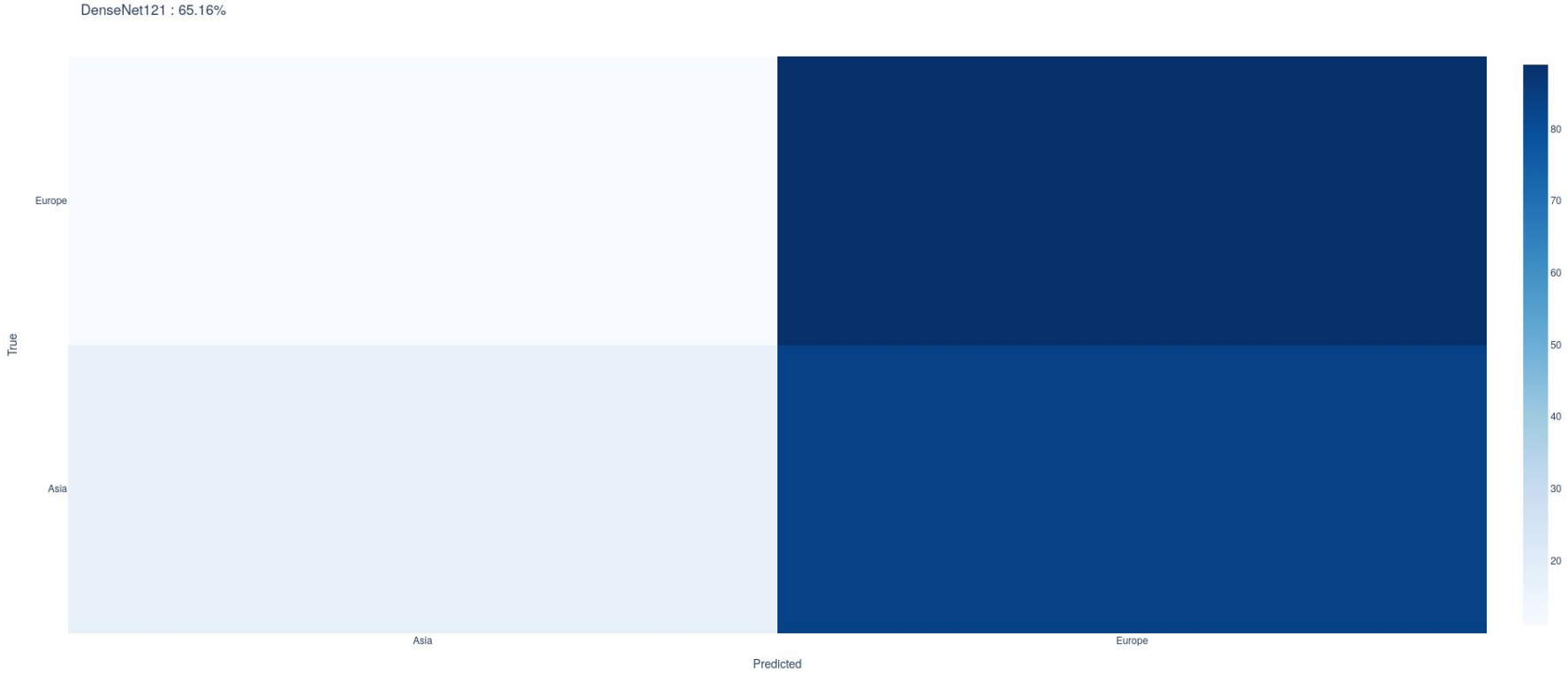
VGG - center cropped images



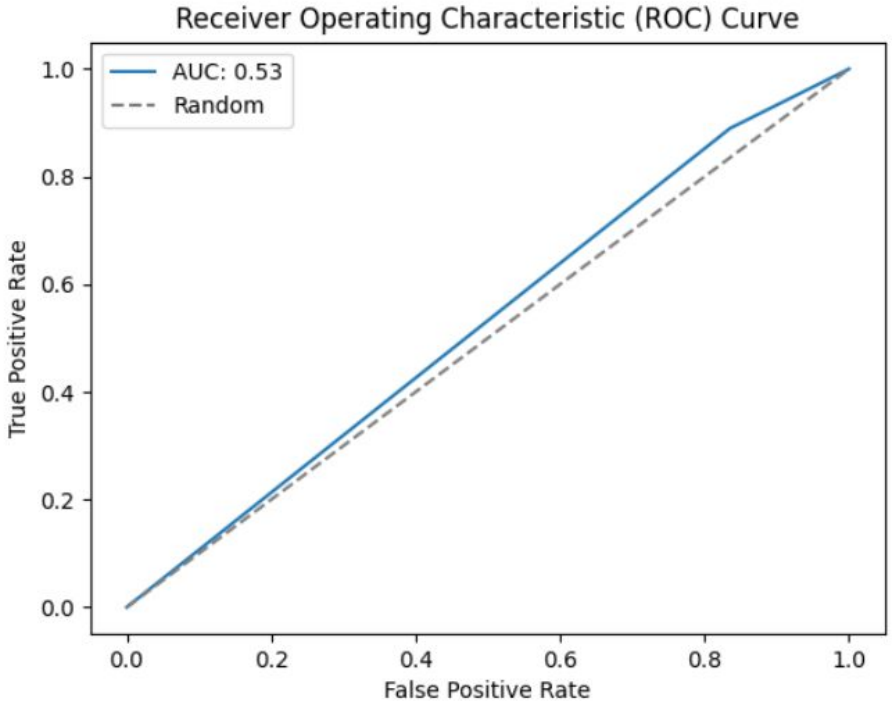
VGG - center cropped images



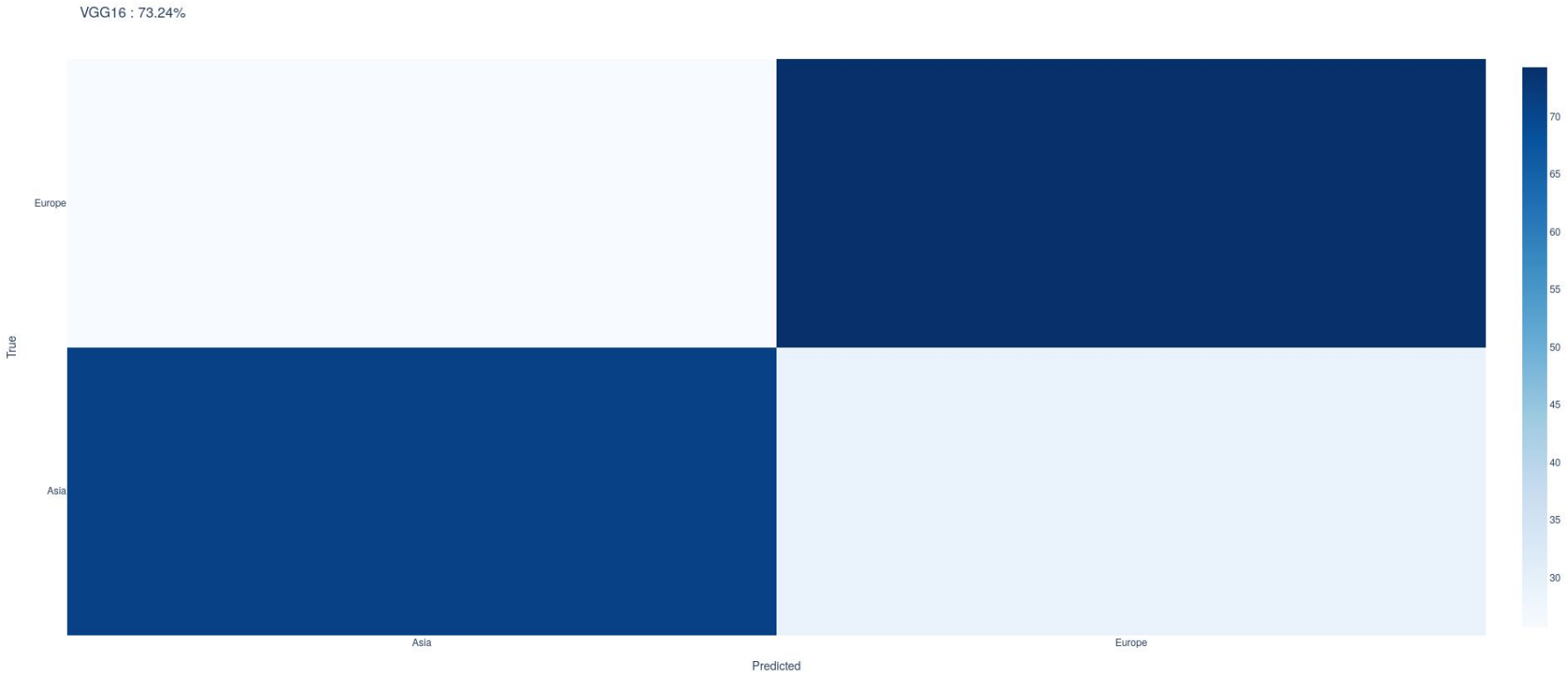
Dense - non cropped images



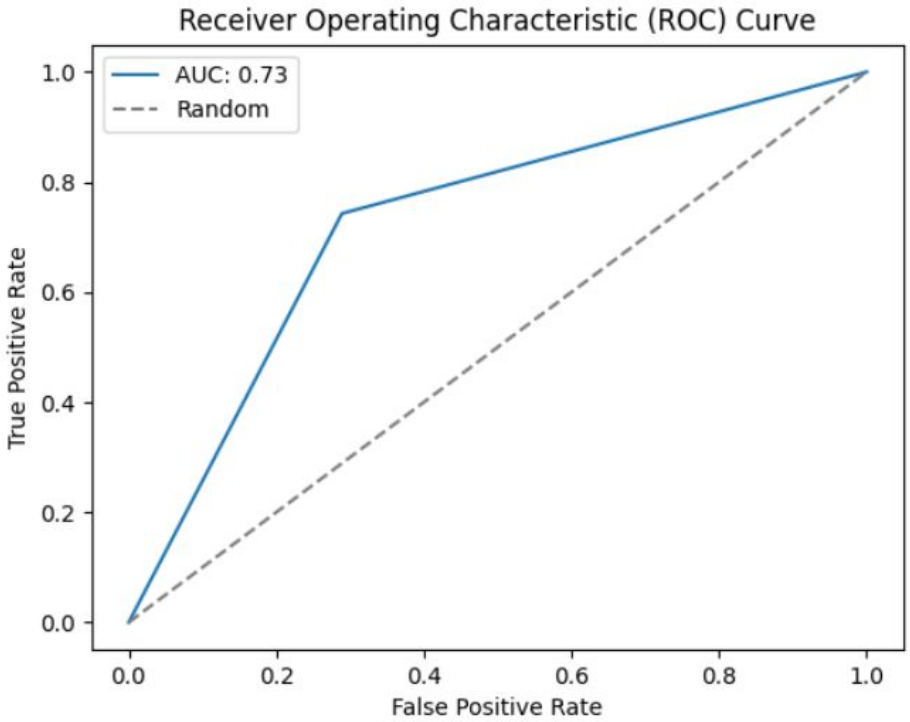
Dense - non cropped images



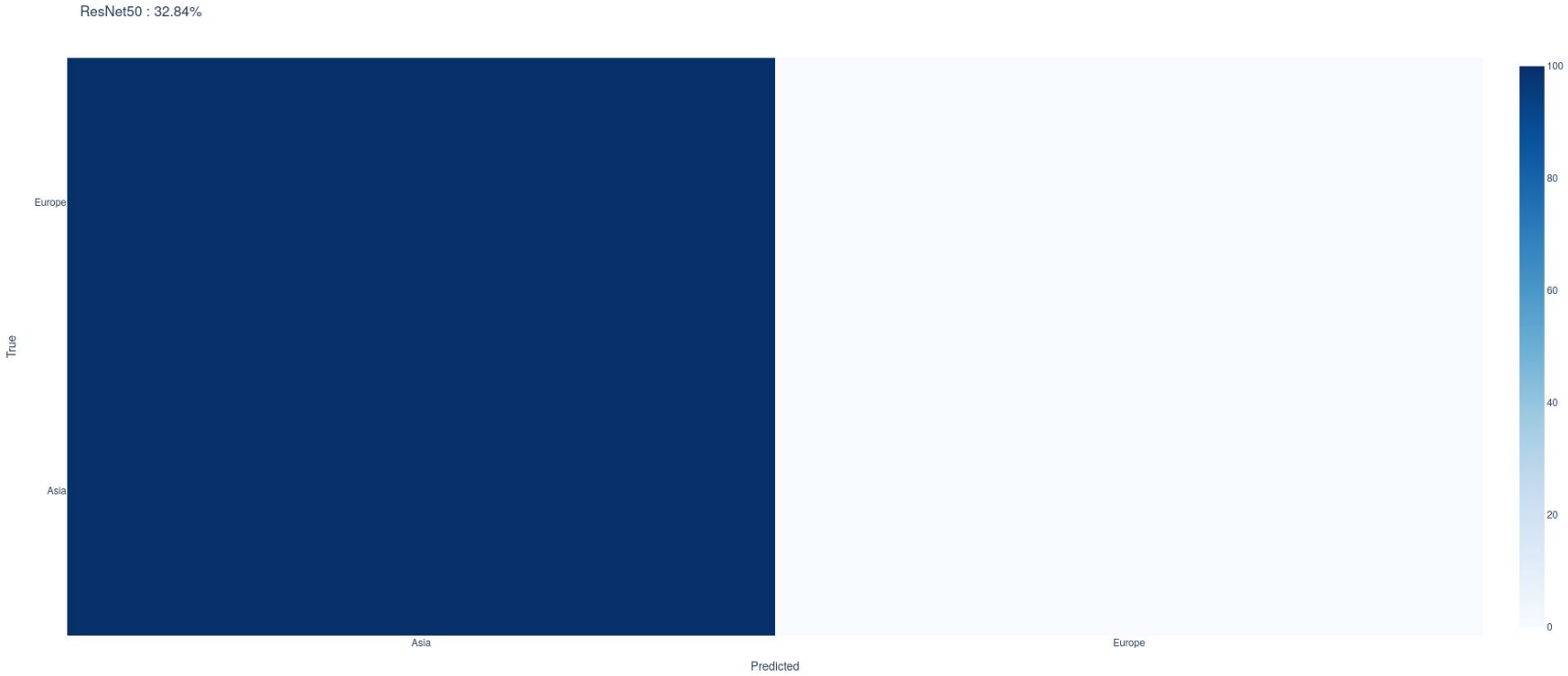
VGG - non cropped images



VGG - non cropped images



ResNet


ResNet50 : 32.84%
Europe
Asia
True
Asia
Europe
Predicted
100
80
60
40
20
0

Lessons learned

Lessons Learned

- Transfer learning is its own science
- We were lacking both compute and more images
- Overfitting is tough to avoid with complex network architectures

What we would have done differently?

- Accumulated more data in the beginning
- Capture panorama (3 images) to better determine one position
- More research about transfer learning and preventing overfitting
- Limit the scope to a more specific part of the image

What could be added to the approach?

Filtering:

Filtering with Places365 dataset (Scene Recognition)

Filtering with YOLO for cropping to areas of interest

- ex. trees, foliage, houses

Further data augmentation