MAIS 202 Project Deliverable 2
Landscape Classifier

Benjamin Jiang

1. Problem Statement
   My project is a classification problem between six different landscapes. With the input of a picture representing a landscape, the program could identify which landscape it identified it to be.

2. Data Preprocessing
   The dataset I am working with is the one found on Kaggle, for the intel image classification problem found at https://www.kaggle.com/puneet6060/intel-image-classification. The dataset consisted of about 12 000 training images, 3 000 test images and 7000 prediction images. However, only the training and test images were separated with respect to their classification (glacier, street, forest, mountain, buildings and sea) in subfolders with the appropriate label. Since the prediction set was not separated, I combined both the training and test images, to then split them into a 80-20 split (the model I use performs training and validation with the input data in a 70-30 split resulting in a 56-24-20 split with the testing data). Not much pre-processing was needed other than resizing the images and placing them into a matrix, with their matching labels in the same order in another matrix (labels were given a value from 0-5).

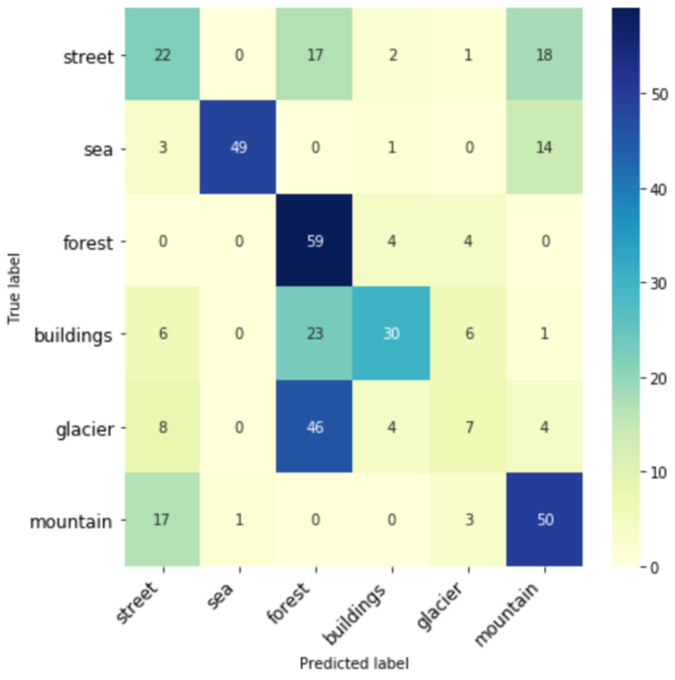
3. Machine learning model
   I proposed an approach utilizing the YOLO algorithm used for object recognition, but I decided to change the model since my project did not require it. My project is a rather classical classification problem and the use of the keras sequential model is more appropriate.

   When I attempted training my model, I ran into time issues. Training the model on Jupyter notebook took an obscene amount of time. Instead, I trained my model on terminal which proved to be faster, but I haven't undergone training with the full dataset yet. I trained a model with 2000 training data images (about 20%). This resulting in underfitting due to lack of data. Furthermore, the model I trained only had 5 epochs, which also played part in the low accuracy. I tested the 'test-model' using part of the test set (400 images) and the evaluation function from keras methods.

4. Preliminary results

   The model I trained had a test accuracy of 54%. It also had a logistic loss of 1,14. These results are clearly indicative of underfitting. Here is a confusion matrix of the predictions compared to the actual labels:

```
0.5425
```



   With these preliminary results, I remain optimistic to the feasibility of the project, since I think that more data and more epochs will result in a model with better results and good accuracy.

5. Next steps

   The main next task will be to fully train my model, making sure not to overfit in a non-obscene amount of time with more data to then look at the results and evaluate what I should change in my model. Once the model is trained, implementing it into a web-app will be the next challenge. I am aware that the compatibility of libraries can pose significant problems for the implementation of a model into a web-app.