

Capstone Project 2 Milestone Report

1. Updated Problem Statement:

Image classification is one of the most popular areas in the last decade. There are many developing areas that need AI implementations in order to recognize the environmental images. For instance; unmanned vehicles need maximum possible accuracy to render their environmental obstacles. In my project I want to classify nature pictures with high accuracy and find ways to improve the accuracy. That will contribute future models as well as training of future data scientists. The objective of this project is as defined below.

- According to given dataset, to build neural network model to predict randomly chosen nature pictures under six classes.
- To find possible ways to improve the accuracy of model.
- To give a general picture of accuracy for the test set.

2. Obtaining Data:

The context of the data includes image data of Natural Scenes around the world. Photographs were collected from Jan Bottinger who is a German photographer and traveller. He publishes his photographs publicly on Unsplash web portal. The Train, Test and Prediction data is separated in each zip files. There are around 14k images in Train, 3k in Test and 7k in Prediction.

This data was initially published on <https://datahack.analyticsvidhya.com> by Intel to host an Image Classification Challenge. This data contains around 25k images of size 150x150 distributed under 6 categories. {'buildings' -> 0, 'forest' -> 1, 'glacier' -> 2, 'mountain' -> 3, 'sea' -> 4, 'street' -> 5}

The data downloaded from web portal under the folder classification for train and test set. Both train and test folders have subfolders that previously classified in six categories. There is a third folder for prediction which has no classification in it.

In order to import data “os” module was imported, class names were defined, a new function was created. This steps are reflected below:

```
1 import numpy as np
2 import os
```

Categories

```
In [46]: 1 class_names = ['mountain', 'street', 'glacier', 'buildings', 'sea', 'forest']
2 class_names_label = {'mountain': 0,
3                       'street' : 1,
4                       'glacier' : 2,
5                       'buildings' : 3,
6                       'sea' : 4,
7                       'forest' : 5
8                       }
9 nb_classes = 6
```

Data loading

```
In [3]: 1 def load_data():
2     """
3     Load the data:
4     - images to train the network.
5     - images to evaluate how accurately the network learned to classify images.
6     """
7
8     datasets = ['seg_train/seg_train', 'seg_test/seg_test']
9     size = (150,150)
10    output = []
11    for dataset in datasets:
12        directory = dataset
13        images = []
14        labels = []
15        for folder in os.listdir(directory):
16            curr_label = class_names_label[folder]
17            for file in os.listdir(directory + "/" + folder):
18                img_path = directory + "/" + folder + "/" + file
19                curr_img = cv2.imread(img_path)
20                curr_img = cv2.resize(curr_img, size)
21                images.append(curr_img)
22                labels.append(curr_label)
23        images, labels = shuffle(images, labels)
24        images = np.array(images, dtype = 'float32')
25        labels = np.array(labels, dtype = 'int32')
26
27        output.append((images, labels))
28
29    return output
```

```
In [4]: 1 (train_images, train_labels), (test_images, test_labels) = load_data()
```

3. Data Cleaning & Data Wrangling:

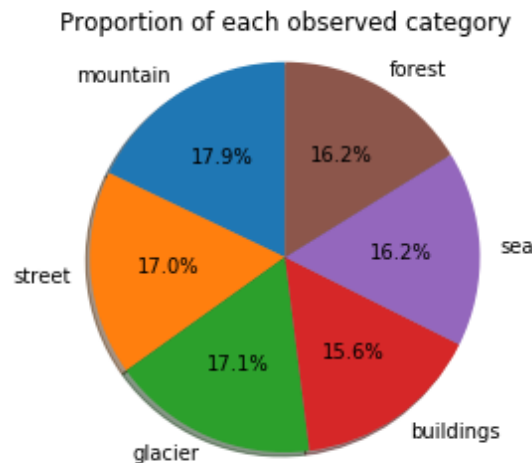
The data was pretty clean and ready to use. After importing the data, the information for the data was analyzed as it is seen below:

```
In [5]: 1 print ("Number of training examples: " + str(train_labels.shape[0]))
2 print ("Number of testing examples: " + str(test_labels.shape[0]))
3 print ("Each image is of size: " + str(train_images.shape[1:]))
```

Number of training examples: 14034
Number of testing examples: 3000
Each image is of size: (150, 150, 3)

Distribution of Data

```
In [20]: 1 sizes = np.bincount(train_labels)
2 explode = (0, 0, 0, 0, 0, 0)
3 plt.pie(sizes, explode=explode, labels=class_names,
4 autopct='%1.1f%%', shadow=True, startangle=90)
5 plt.axis('equal')
6 plt.title('Proportion of each observed category')
7
8 plt.show()
```



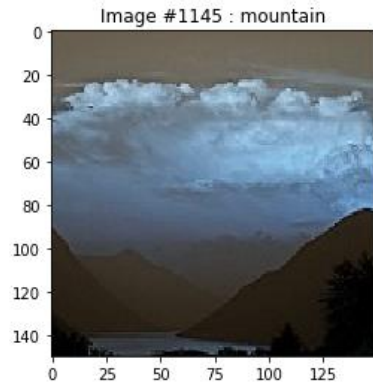
4. Initial Findings:

- After preparing the dataset the images are scaled as below:

```
In [21]: 1 train_images = train_images / 255.0
2 test_images = test_images / 255.0
```

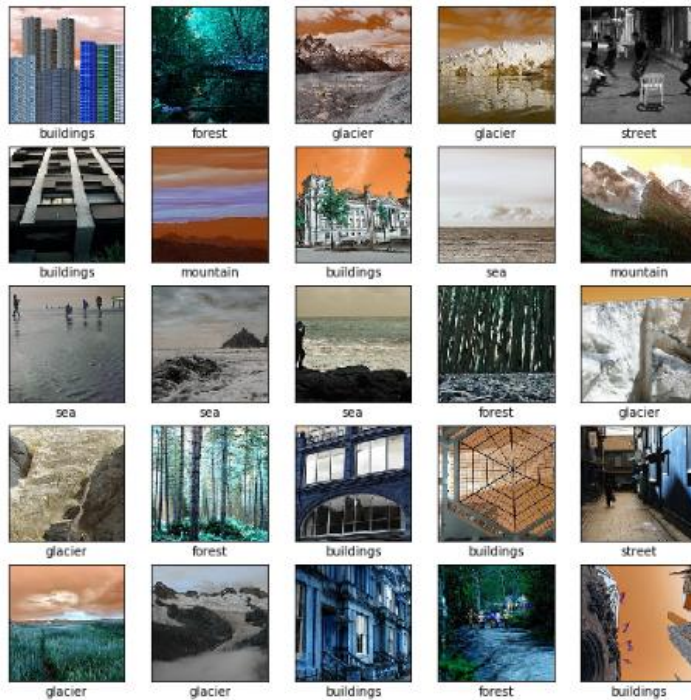
- Then to display a random image in the training set, the code below conducted. The labels were checked by this for both one single picture and multiple pictures.

```
In [22]: 1 index = np.random.randint(train_images.shape[0])
2         plt.figure()
3         plt.imshow(train_images[index])
4         plt.grid(False)
5         plt.title('Image #{} : {}'.format(index) + class_names[train_labels[index]])
6         plt.show()
```



```
In [23]: 1 fig = plt.figure(figsize=(10,10))
2         fig.suptitle("Some examples of images of the dataset", fontsize=16)
3         for i in range(25):
4             plt.subplot(5,5,i+1)
5             plt.xticks([])
6             plt.yticks([])
7             plt.grid(False)
8             plt.imshow(train_images[i], cmap=plt.cm.binary)
9             plt.xlabel(class_names[train_labels[i]])
10        plt.show()
```

Some examples of images of the dataset



After this stage we will start building the deep learning model by the help of tensorflow Keras.

The further stage for this study will be CNN model on train set images which transformed into one dimension array to predict labels of pictures with a high accuracy.