Natural scene image classification

SARWAR MIR MUHAMMAD SULEMAN

AD20216802

NETWORK CONVERGENCE LAB

Contents

Introduction

This is image data of Natural Scenes around the world

This data contains around 25k images of size 150x150, distributed under 6 categories.

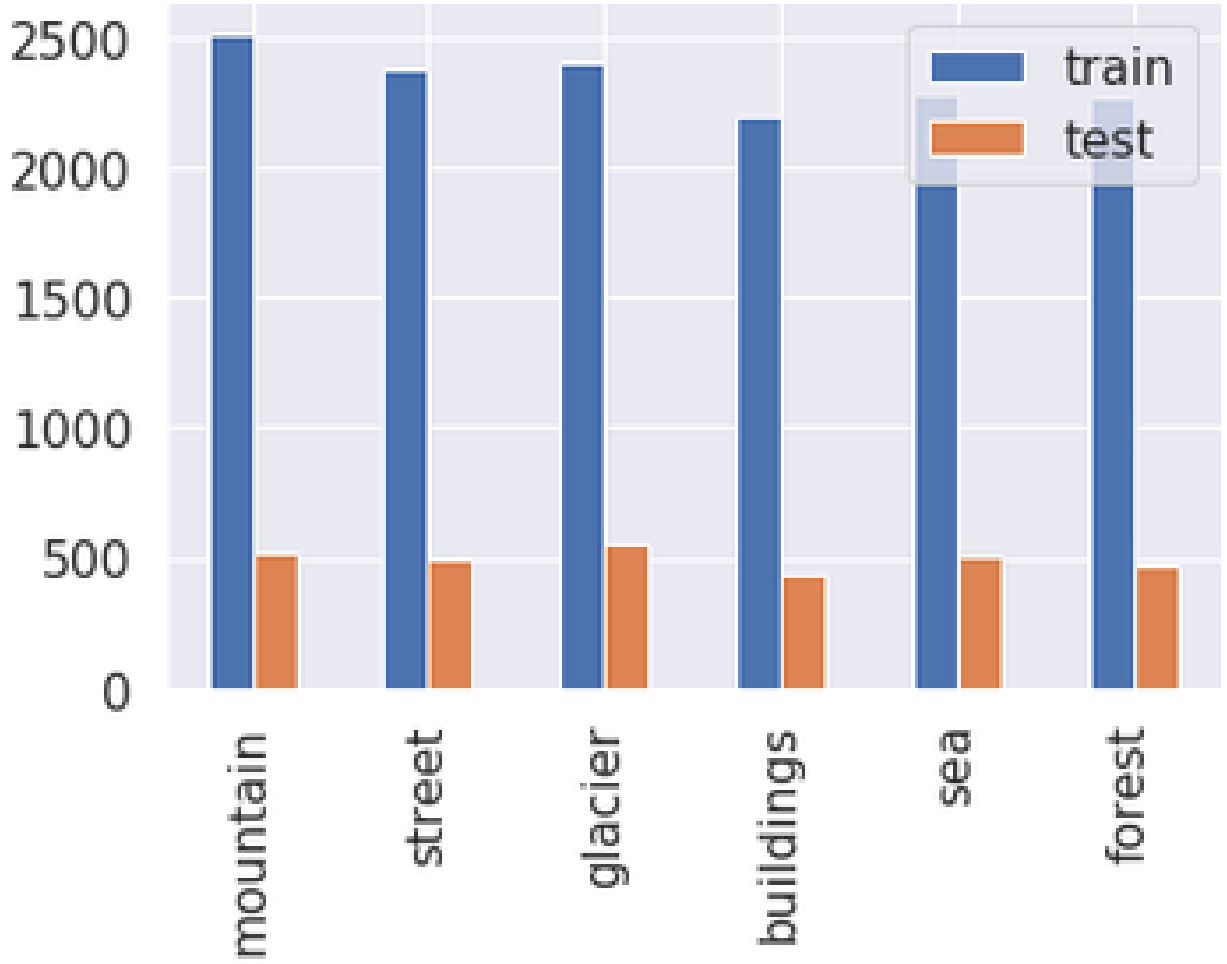
- 'buildings' -> 0, 'forest' -> 1, 'glacier' -> 2, 'mountain' -> 3, 'sea' -> 4, 'street' -> 5
- 14k images in Train, 3k in Test and 7k in Prediction

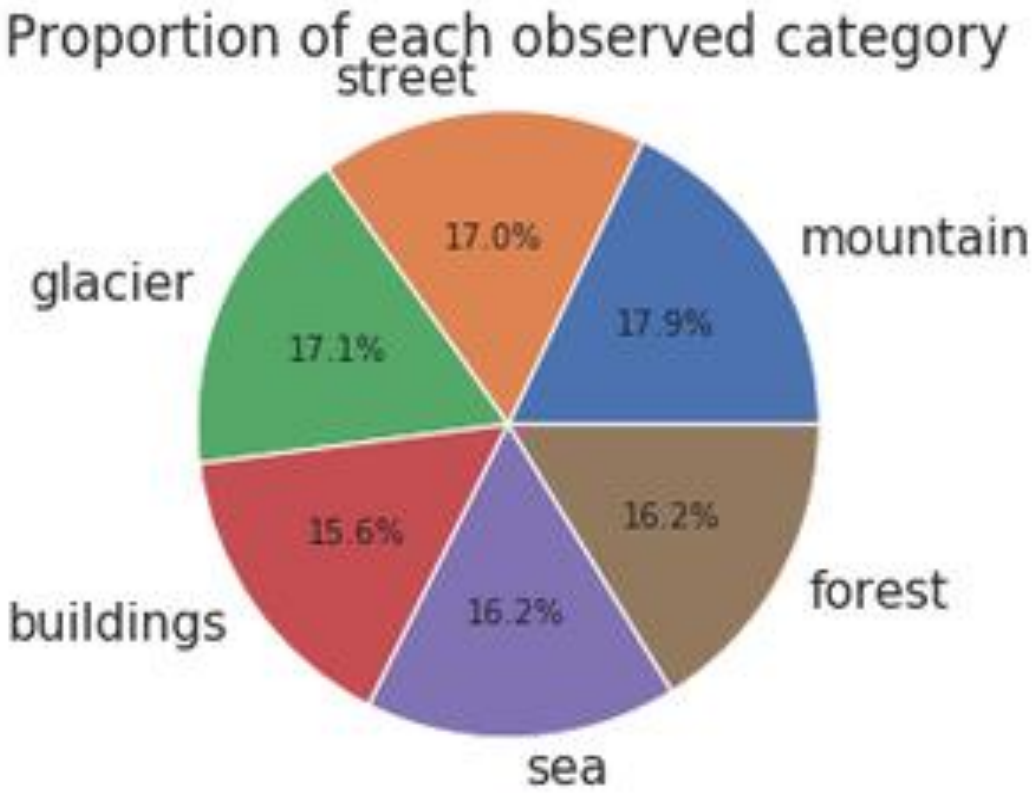
Exploratory Data Analysis (EDA)

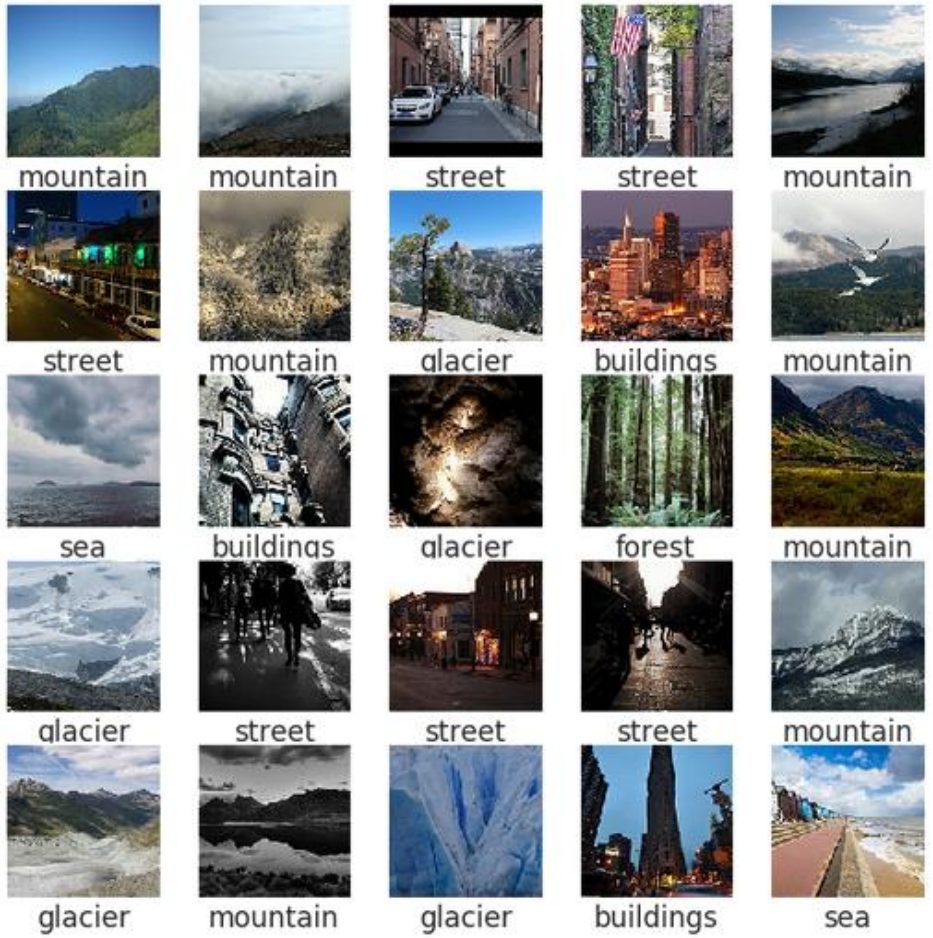
```
import numpy as np
import os
from sklearn.metrics import confusion_matrix
import seaborn as sn; sn.set(font_scale=1.4)
from sklearn.utils import shuffle
import matplotlib.pyplot as plt
import cv2
import tensorflow as tf
from tqdm import tqdm
```

```
class_names = ['mountain', 'street', 'glacier', 'buildings', 'sea', 'forest']
class_names_label = {class_name:i for i, class_name in enumerate(class_names)}

nb_classes = len(class_names)

IMAGE_SIZE = (150, 150)
```

```
datasets = ['../input/seg_train/seg_train', '../input/seg_test/seg_test']
output = []

# Iterate through training and test sets
for dataset in datasets:

    images = []
    labels = []

    print("Loading {}".format(dataset))

    # Iterate through each folder corresponding to a category
    for folder in os.listdir(dataset):
        label = class_names_label[folder]

        # Iterate through each image in our folder
        for file in tqdm(os.listdir(os.path.join(dataset, folder))):

            # Get the path name of the image
            img_path = os.path.join(os.path.join(dataset, folder), file)

            # Open and resize the img
            image = cv2.imread(img_path)
            image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
            image = cv2.resize(image, IMAGE_SIZE)

            # Append the image and its corresponding label to the output
            images.append(image)
            labels.append(label)

    images = np.array(images, dtype = 'float32')
    labels = np.array(labels, dtype = 'int32')

    output.append((images, labels))
```


2500
2000
1500
1000
500
0
train
test
mountain
street
glacier
buildings
sea
forest


Proportion of each observed category
street
mountain
glacier
buildings
forest
sea
17.0%
17.9%
17.1%
15.6%
16.2%
16.2%


mountain
mountain
street
street
mountain
street
mountain
glacier
buildings
mountain
sea
buildings
glacier
forest
mountain
glacier
street
street
street
mountain
glacier
mountain
glacier
buildings
sea

Model configuration

```
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation = 'relu', input_shape = (150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(32, (3, 3), activation = 'relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation=tf.nn.relu),
    tf.keras.layers.Dense(6, activation=tf.nn.softmax)
])
```

Model compilation and training

```
model.compile(optimizer = 'adam', loss = 'sparse_categorical_crossentropy', metrics=['accuracy
'])
```

```
history = model.fit(train_images, train_labels, batch_size=128, epochs=20, validation_split =
0.2)
```

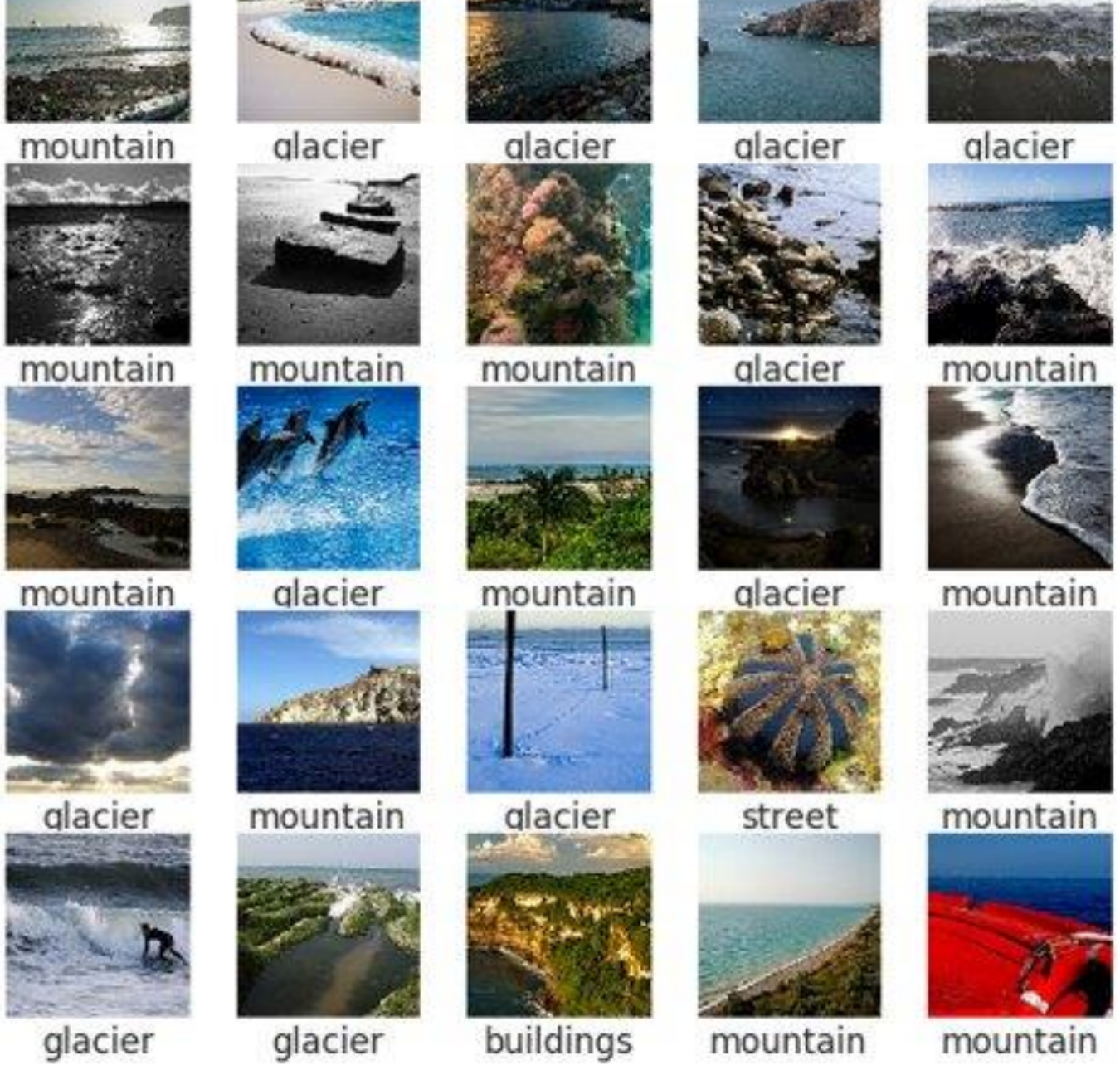
Model evaluation- 78%

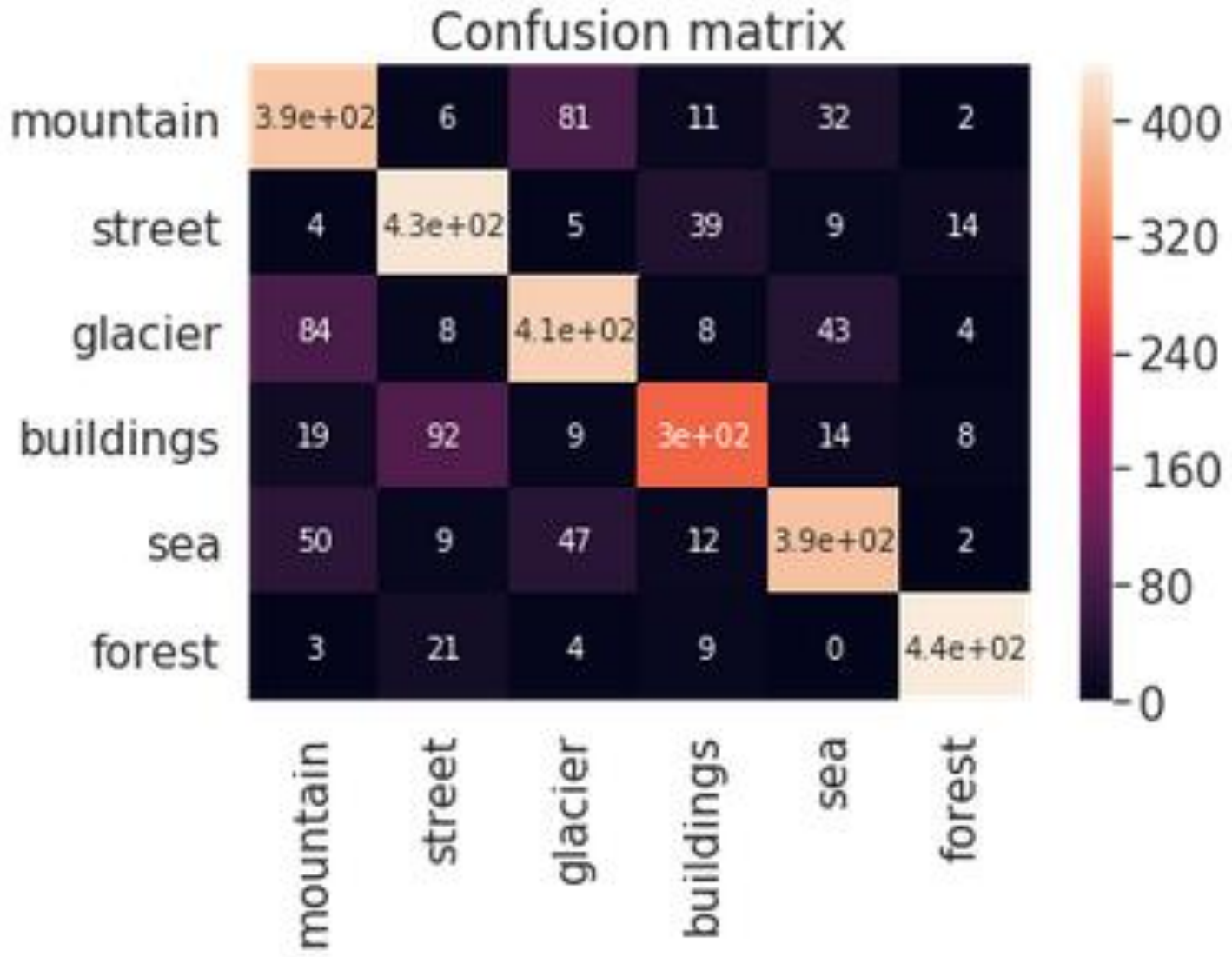
```
test_loss = model.evaluate(test_images, test_labels)
```

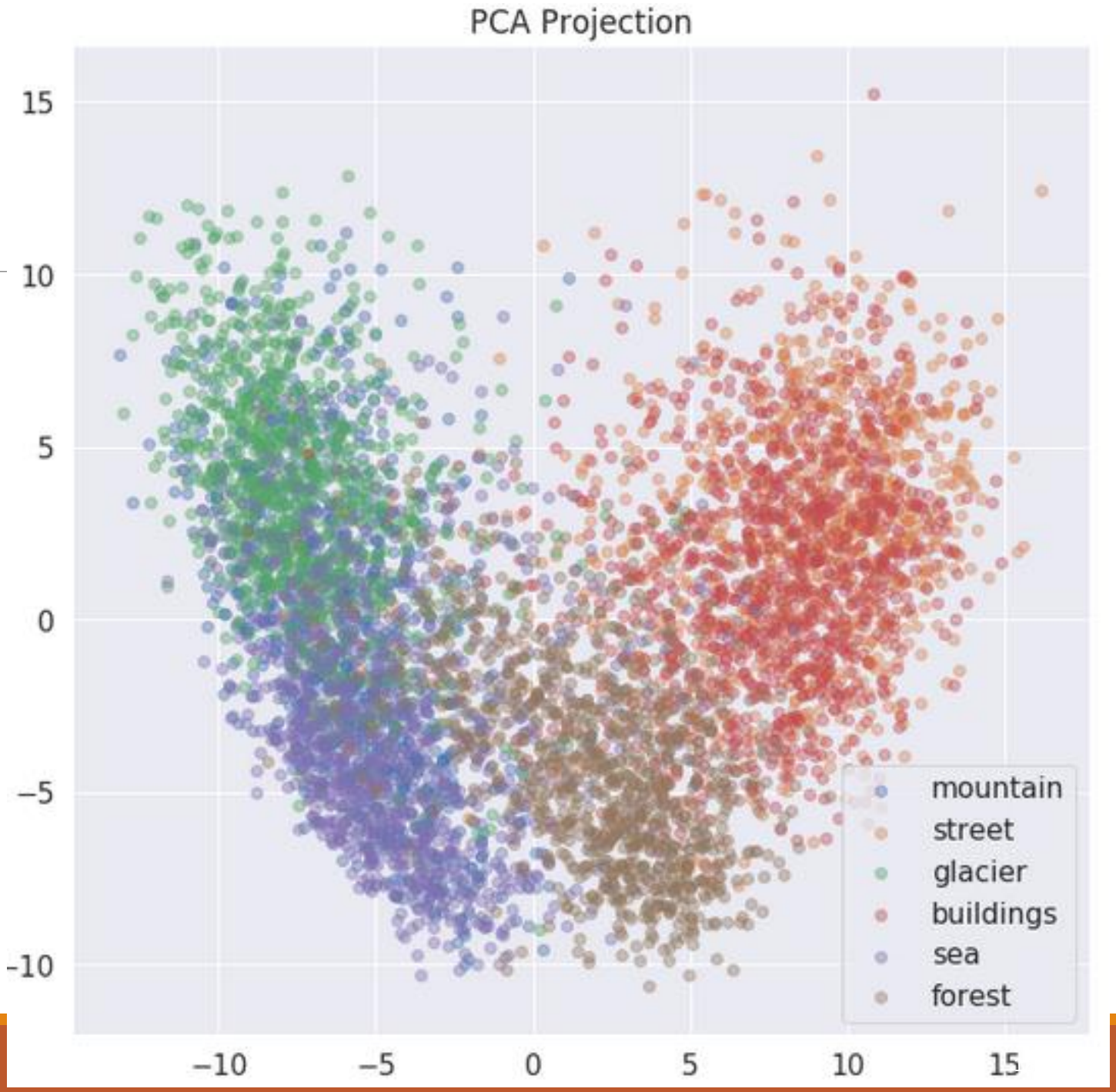
```
3000/3000 [==============================] - 1s 264us/sample - loss: 1.0085 - acc: 0.7837
```

```
predictions = model.predict(test_images)     # Vector of probabilities
pred_labels = np.argmax(predictions, axis = 1) # We take the highest probability

display_random_image(class_names, test_images, pred_labels)
```

Image #1888 : glacier


mountain
glacier
glacier
glacier
glacier
mountain
mountain
mountain
glacier
mountain
mountain
glacier
mountain
glacier
mountain
glacier
mountain
glacier
mountain
glacier
street
mountain
glacier
glacier
buildings
mountain
mountain


Confusion matrix
mountain
street
glacier
buildings
sea
forest
3.9e+02
6
81
11
32
2
4
4.3e+02
5
39
9
14
84
8
4.1e+02
8
43
4
19
92
9
3e+02
14
8
50
9
47
12
3.9e+02
2
3
21
4
9
0
4.4e+02
400
320
240
160
80
0


PCA Projection
15
10
5
0
−5
−10
−10
−5
0
5
10
15
mountain
street
glacier
buildings
sea
forest

VGG Model

imports

```
from keras.applications.vgg16 import VGG16
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
```

```
model2 = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape = (x, y, z)),
    tf.keras.layers.Dense(50, activation=tf.nn.relu),
    tf.keras.layers.Dense(6, activation=tf.nn.softmax)
])

model2.compile(optimizer = 'adam', loss = 'sparse_categorical_crossentropy', metrics=['accurac
y'])

history2 = model2.fit(train_features, train_labels, batch_size=128, epochs=15, validation_spli
t = 0.2)
```

VGG evaluation – 87%

```
test_loss = model2.evaluate(test_features, test_labels)
```

```
3000/3000 [==============================] - 0s 90us/sample - loss: 0.4345 - acc: 0.8730
```

```
from keras.layers import Input, Dense, Conv2D, Activation , MaxPooling2D, Flatten

model2 = VGG16(weights='imagenet', include_top=False)

input_shape = model2.layers[-4].get_input_shape_at(0) # get the input shape of desired layer
layer_input = Input(shape = (9, 9, 512)) # a new input tensor to be able to feed the desired la
yer
# https://stackoverflow.com/questions/52800025/keras-give-input-to-intermediate-layer-and-get-fi
nal-output

x = layer_input
for layer in model2.layers[-4::1]:
    x = layer(x)

x = Conv2D(64, (3, 3), activation='relu')(x)
x = MaxPooling2D(pool_size=(2, 2))(x)
x = Flatten()(x)
x = Dense(100,activation='relu')(x)
x = Dense(6,activation='softmax')(x)

# create the model
new_model = Model(layer_input, x)
```

Model compilation and training

```
new_model.compile(optimizer = 'adam', loss = 'sparse_categorical_crossentropy',
metrics=['accuracy'])
```

```
history = new_model.fit(train_features, train_labels, batch_size=128, epochs=10,
validation_split = 0.2)
```

```
from sklearn.metrics import accuracy_score

predictions = new_model.predict(test_features)
pred_labels = np.argmax(predictions, axis = 1)
print("Accuracy : {}".format(accuracy_score(test_labels, pred_labels)))
```

```
Accuracy : 0.8933333333333333
```

Model	CNN	VGG	VGG(fine-tune)
Accouracy	78%	87%	89%

conclusion

Firstly, the dataset is analyzed

A basic CNN is trained and tested but the model has an error while predicting mountain and glacier and the accuracy is 78%

PCA is used to understand the behavior of data

Then, VGG is trained to resolve the aforementioned errors and accuracy is improved to 87%

For further enhancement in accuracy, the model is fine-tuned

The comparison results shows that with fine-tuned VGG the accuracy is improved up to 89%

Thank You