Outdoor Scene Classification based on CNN

Project guide

Shantala giraddi

Presented by

Sindhu Hachadad
Shrinivas Miskin
Shriya Bannikop
Sushmita Talawar

Problem Statement

Scene Image Classification: Identify which kind of natural scenes can the image be categorised into.

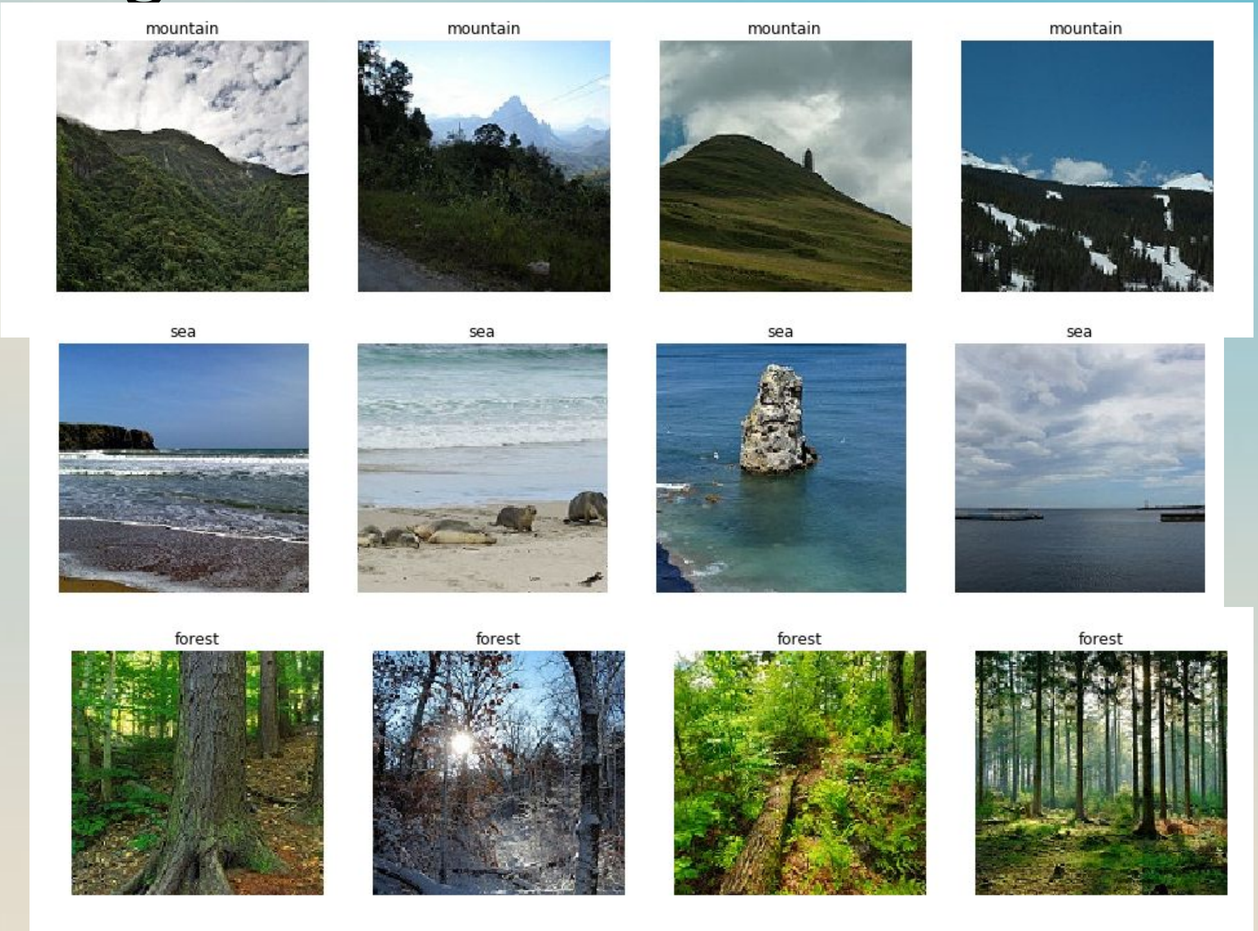
Motivation

- Classification is very important as we use it in daily life. It makes things easier to find and recognise. Differentiation of objects is what allows us to classify them into groups.
- Image Classification is one of the core problems in Computer Vision that, despite its simplicity, has a large variety of practical applications.
- Moreover, as we can see, many other seemingly distinct Computer Vision tasks (such as object detection, segmentation) can be reduced to image classification.

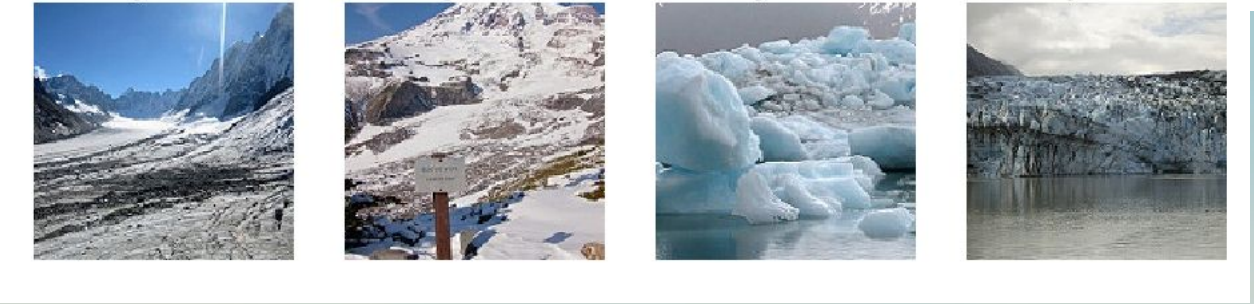
Objectives

- Identify the image.
- Classify into respective category.

Image dataset




buildings
buildings
buildings
buildings


glaciers
glaciers
glaciers
glaciers


street
street
street
street

Dataset:

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 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.

Preprocessing:

Resize all the input images to 150x150

Requirements

A. Functional Requirements

❖ User level

- User shall be able to view images.
- User shall be able to see the category under which it belongs to.

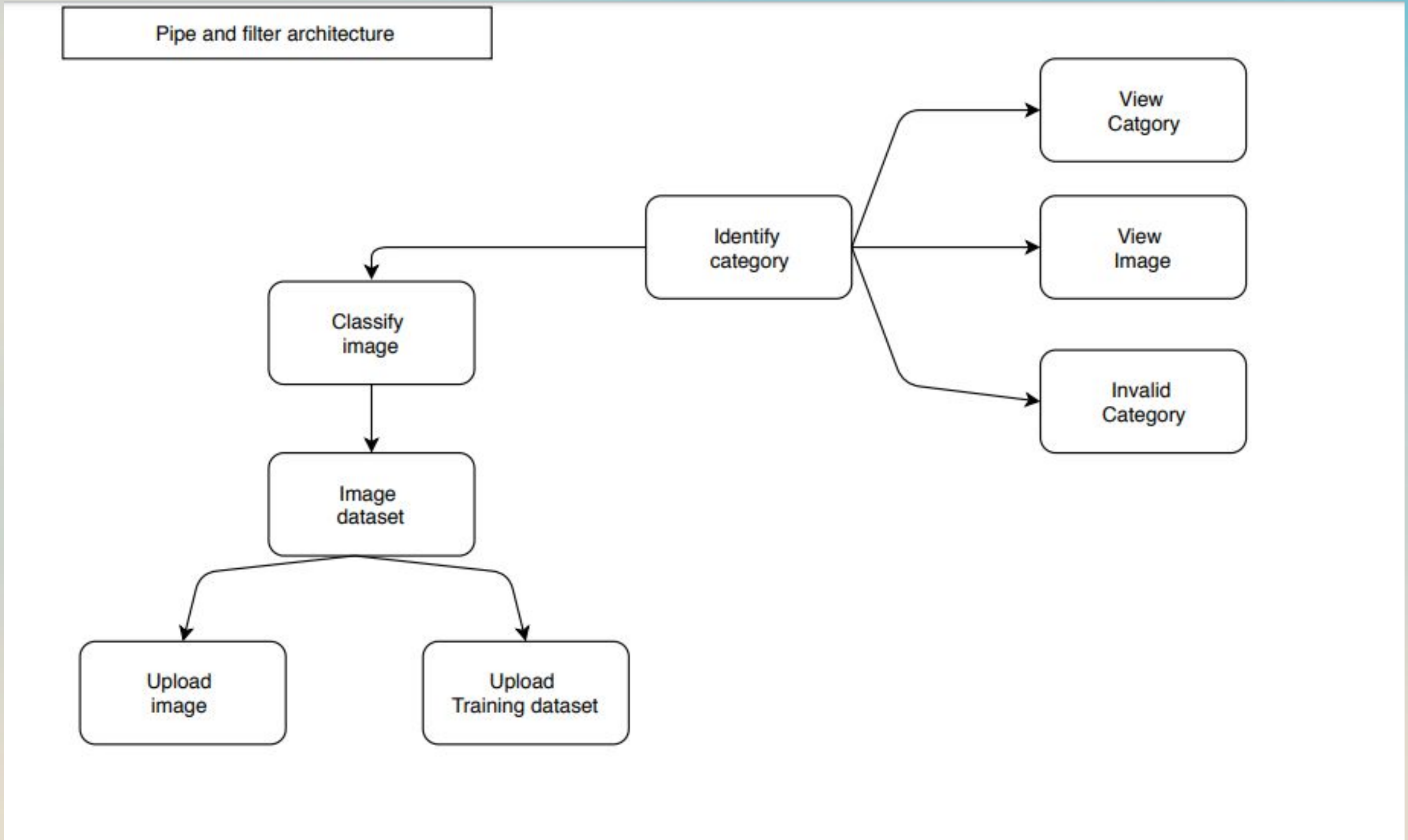
❖ System level

- System shall be able to store data.
- System shall be able to extract features.
- System shall be able to resize the images.
- System shall be able to categorise the images.

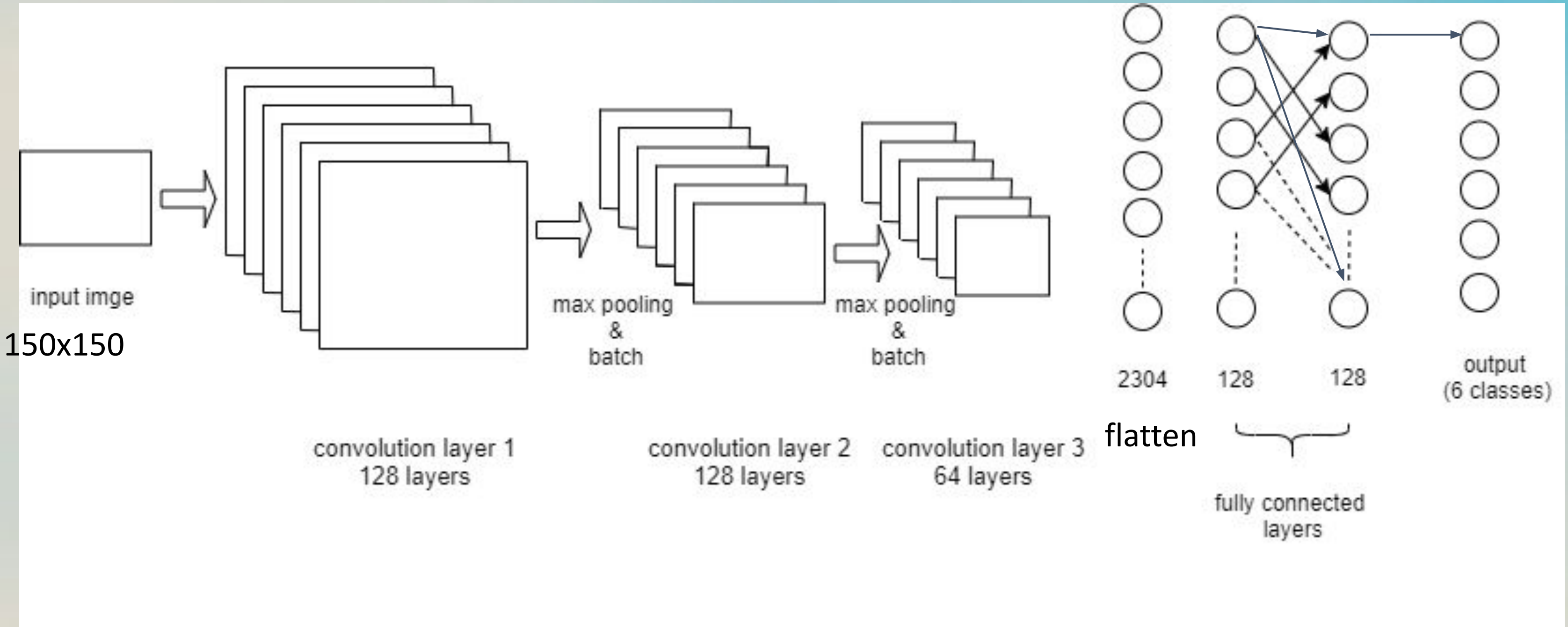
Non-Functional Requirements

- Ease of use
- The system should be able to expand for further storing.
- The system should be compatible with any browser on any environment.
- The system should be able to perform a failure-free operation for a specified period of time in a specified environment.

Architecture



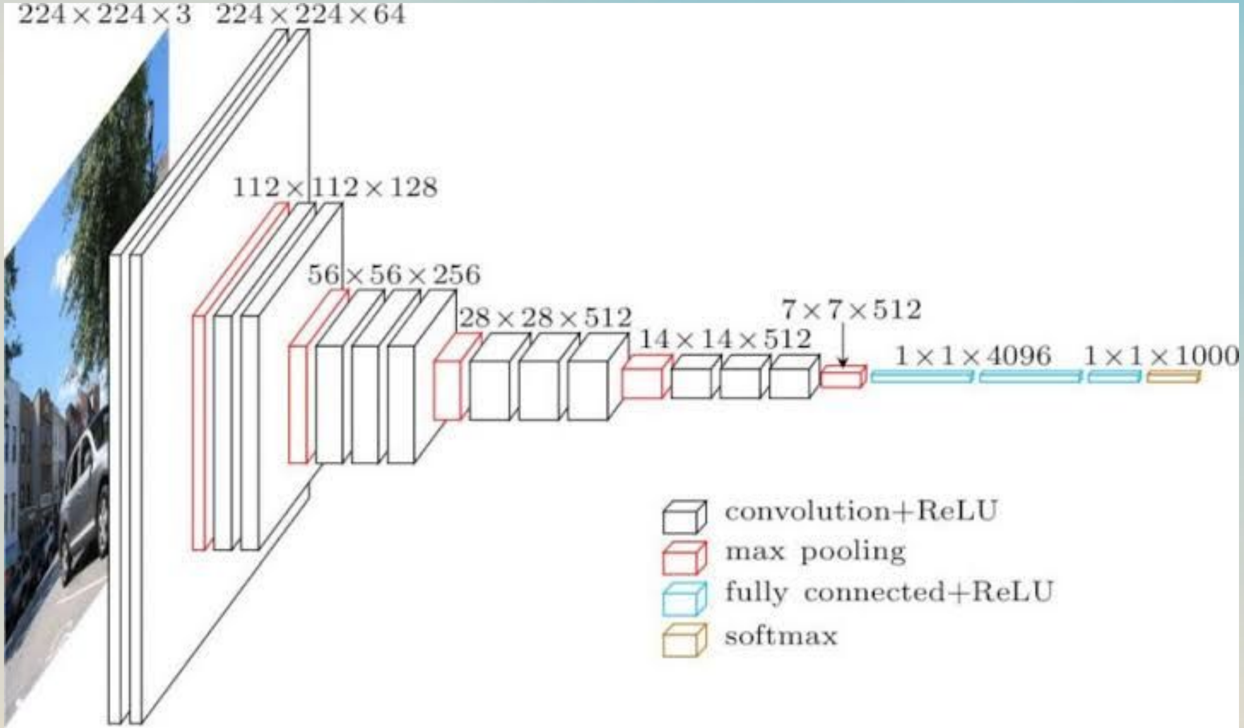
Convolutional Neural Network



Output visualisations

conv2d_1 (Conv2D)	(None, 148, 148, 128)	3584
max_pooling2d_1 (MaxPooling2	(None, 74, 74, 128)	0
batch_normalization_1 (Batch	(None, 74, 74, 128)	512
conv2d_2 (Conv2D)	(None, 72, 72, 128)	147584
max_pooling2d_2 (MaxPooling2	(None, 36, 36, 128)	0
batch_normalization_2 (Batch	(None, 36, 36, 128)	512
conv2d_3 (Conv2D)	(None, 34, 34, 64)	73792
max_pooling2d_3 (MaxPooling2	(None, 6, 6, 64)	0
flatten_1 (Flatten)	(None, 2304)	0
dense_1 (Dense)	(None, 128)	295040

VGG16

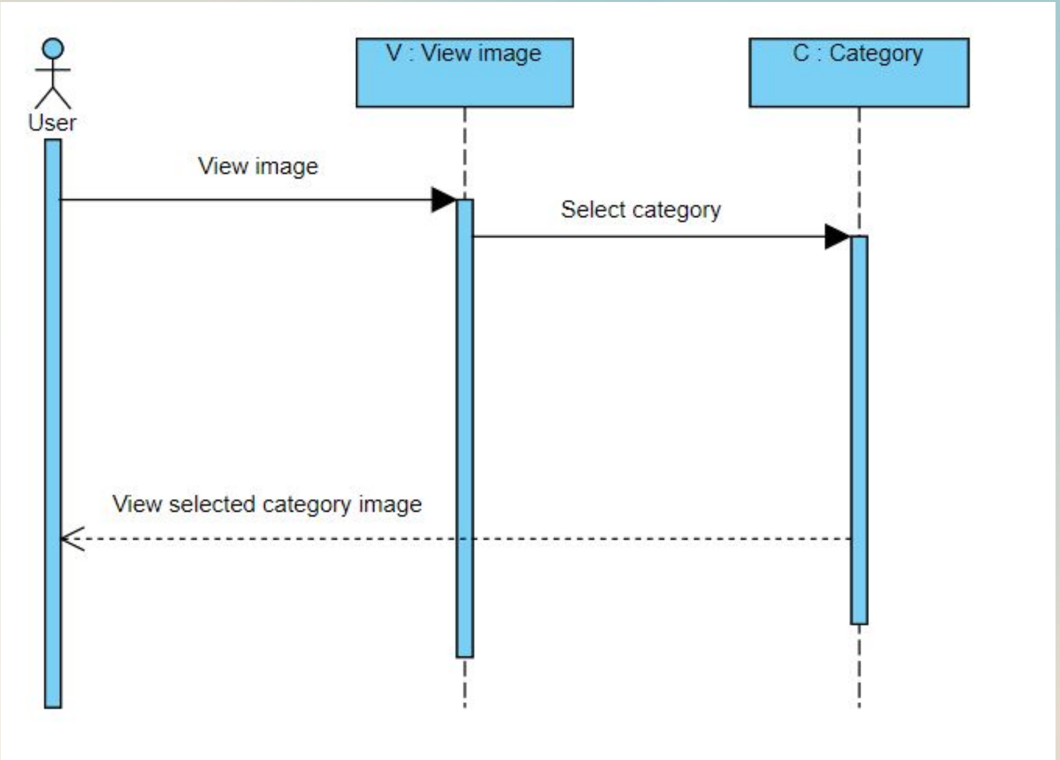


224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU
softmax

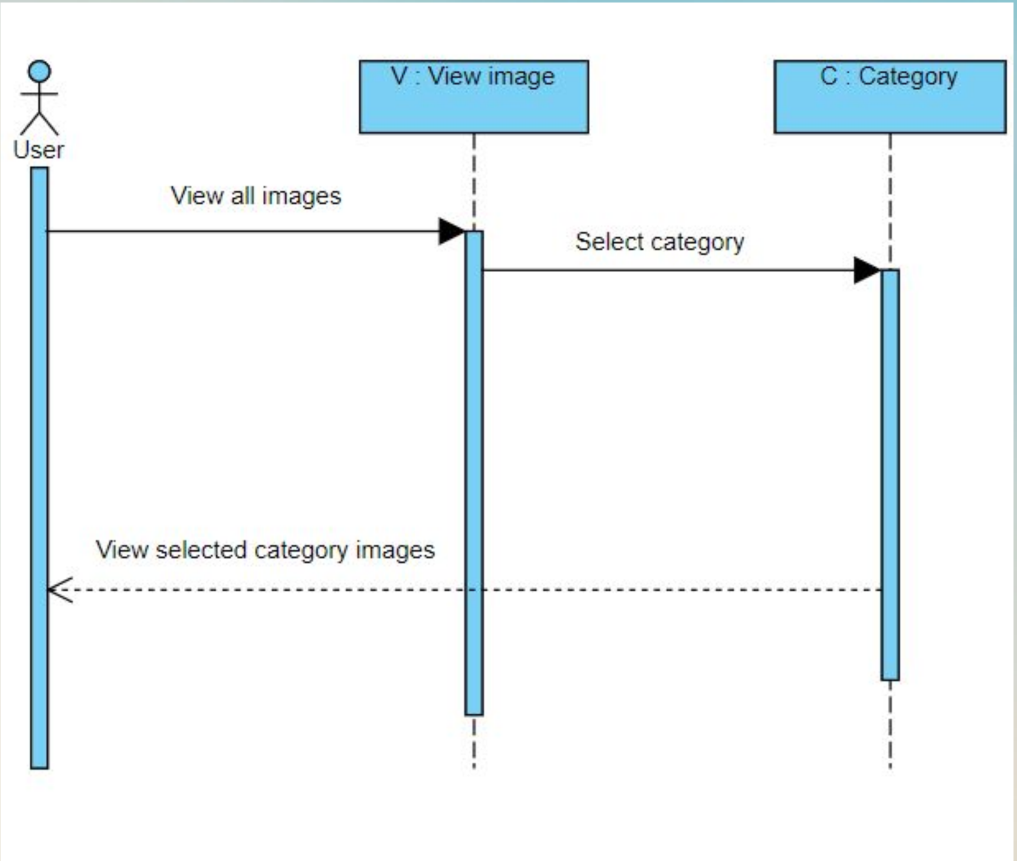
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 4, 4, 512)         14714688
_________________________________________________________________
flatten_1 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_1 (Dense)              (None, 180)               1474740
_________________________________________________________________
dense_2 (Dense)              (None, 100)               18100
_________________________________________________________________
dense_3 (Dense)              (None, 50)                5050
_________________________________________________________________
dropout_1 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_4 (Dense)              (None, 6)                 306
=================================================================
Total params: 16,212,884
Trainable params: 16,212,884
Non-trainable params: 0
_________________________________________________________________
```

Sequence Diagram

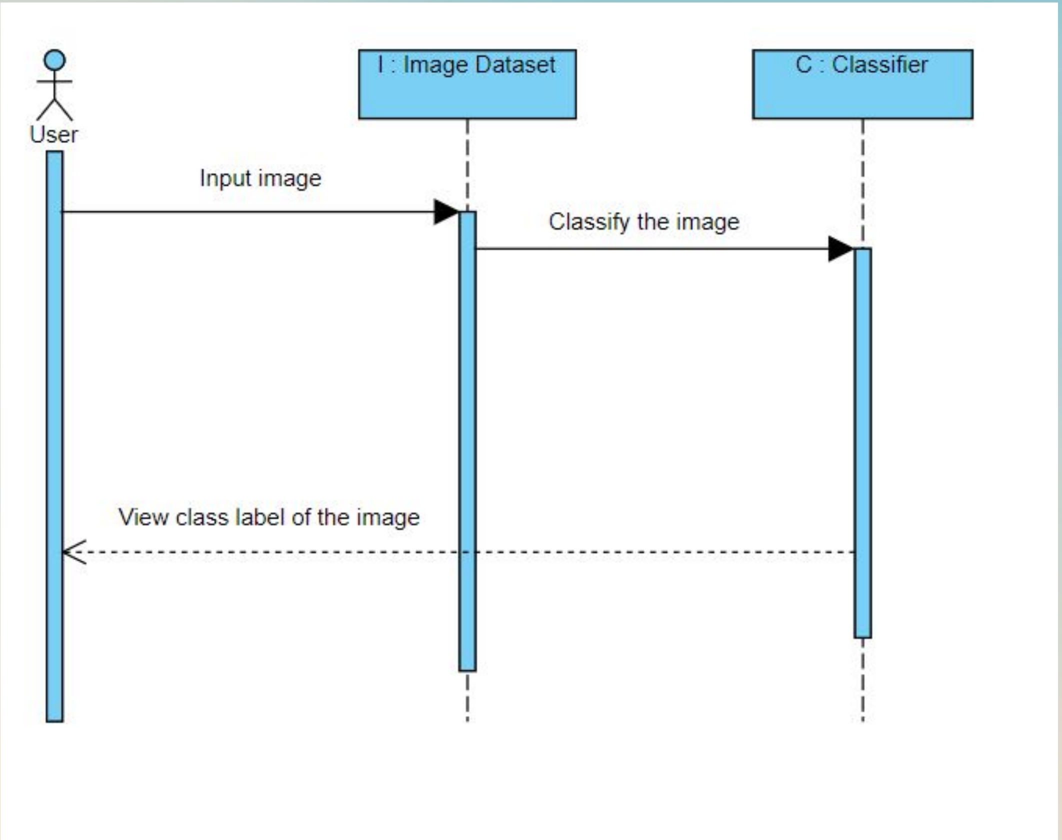
- Sequence diagram for view image



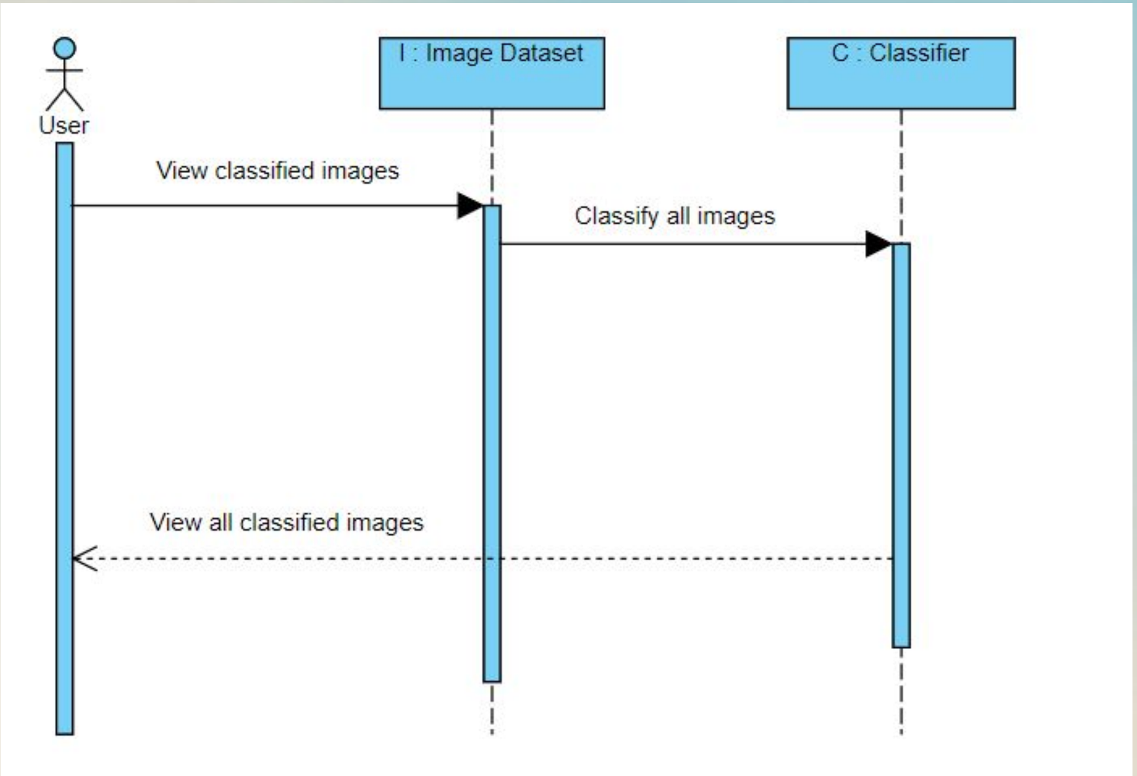
- ➢ Sequence diagram to view all images.



- Sequence diagram to categorize the image.



- Sequence diagram for classify all images.



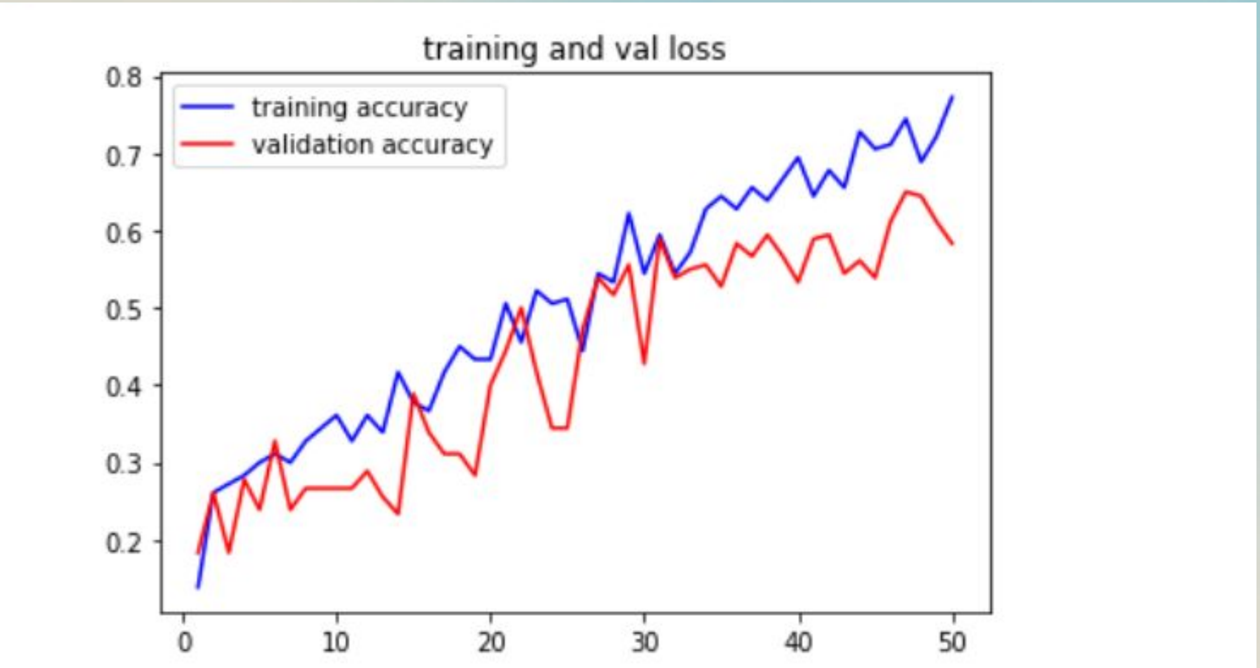
Status of implementation

- CNN using 180 sample dataset

```
Epoch 11/20
18/18 [==============================] - 5s 259ms/step - loss: 1.1254 - accuracy: 0.5722 - val_loss: 1.5289 - val_accuracy: 0.5
444
Epoch 12/20
18/18 [==============================] - 5s 265ms/step - loss: 1.0056 - accuracy: 0.6167 - val_loss: 0.6549 - val_accuracy: 0.6
111
Epoch 13/20
18/18 [==============================] - 5s 257ms/step - loss: 1.0669 - accuracy: 0.5944 - val_loss: 1.0264 - val_accuracy: 0.5
833
Epoch 14/20
18/18 [==============================] - 4s 246ms/step - loss: 0.9645 - accuracy: 0.6278 - val_loss: 1.6021 - val_accuracy: 0.5
611
Epoch 15/20
18/18 [==============================] - 4s 245ms/step - loss: 0.9067 - accuracy: 0.6889 - val_loss: 1.1841 - val_accuracy: 0.6
278
Epoch 16/20
18/18 [==============================] - 4s 243ms/step - loss: 1.0602 - accuracy: 0.6278 - val_loss: 1.5115 - val_accuracy: 0.5
833
Epoch 17/20
18/18 [==============================] - 4s 239ms/step - loss: 0.8839 - accuracy: 0.6778 - val_loss: 1.5035 - val_accuracy: 0.5
889
Epoch 18/20
18/18 [==============================] - 4s 246ms/step - loss: 0.8747 - accuracy: 0.6333 - val_loss: 3.5019 - val_accuracy: 0.5
833
Epoch 19/20
18/18 [==============================] - 4s 245ms/step - loss: 0.9635 - accuracy: 0.6556 - val_loss: 1.1756 - val_accuracy: 0.5
056
Epoch 20/20
18/18 [==============================] - 4s 240ms/step - loss: 0.8739 - accuracy: 0.6500 - val_loss: 0.8842 - val_accuracy: 0.5
111
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 148, 148, 32)      896
_________________________________________________________________
activation_6 (Activation)    (None, 148, 148, 32)      0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 72, 72, 32)        9248
_________________________________________________________________
activation_7 (Activation)    (None, 72, 72, 32)        0
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 36, 36, 32)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 34, 34, 64)        18496
_________________________________________________________________
activation_8 (Activation)    (None, 34, 34, 64)        0
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 17, 17, 64)        0
_________________________________________________________________
flatten_2 (Flatten)          (None, 18496)             0
_________________________________________________________________
dense_3 (Dense)              (None, 64)                1183808
_________________________________________________________________
activation_9 (Activation)    (None, 64)                0
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_4 (Dense)              (None, 6)                 390
_________________________________________________________________
activation_10 (Activation)   (None, 6)                 0
=================================================================
Total params: 1,212,838
Trainable params: 1,212,838
Non-trainable params: 0
```

```
0.5611
Epoch 45/50
18/18 [==============================] - 5s 269ms/step - loss: 0.7973 - accuracy: 0.7056 - val_loss: 0.9434 - val_accuracy:
0.5389
Epoch 46/50
18/18 [==============================] - 5s 271ms/step - loss: 0.7712 - accuracy: 0.7111 - val_loss: 1.5107 - val_accuracy:
0.6111
Epoch 47/50
18/18 [==============================] - 5s 287ms/step - loss: 0.7185 - accuracy: 0.7444 - val_loss: 2.0767 - val_accuracy:
0.6500
Epoch 48/50
18/18 [==============================] - 5s 286ms/step - loss: 0.9956 - accuracy: 0.6889 - val_loss: 1.2893 - val_accuracy:
0.6444
Epoch 49/50
18/18 [==============================] - 5s 299ms/step - loss: 0.8007 - accuracy: 0.7222 - val_loss: 1.3065 - val_accuracy:
0.6111
Epoch 50/50
18/18 [==============================] - 5s 300ms/step - loss: 0.6717 - accuracy: 0.7722 - val_loss: 2.8436 - val_accuracy:
0.5833
```

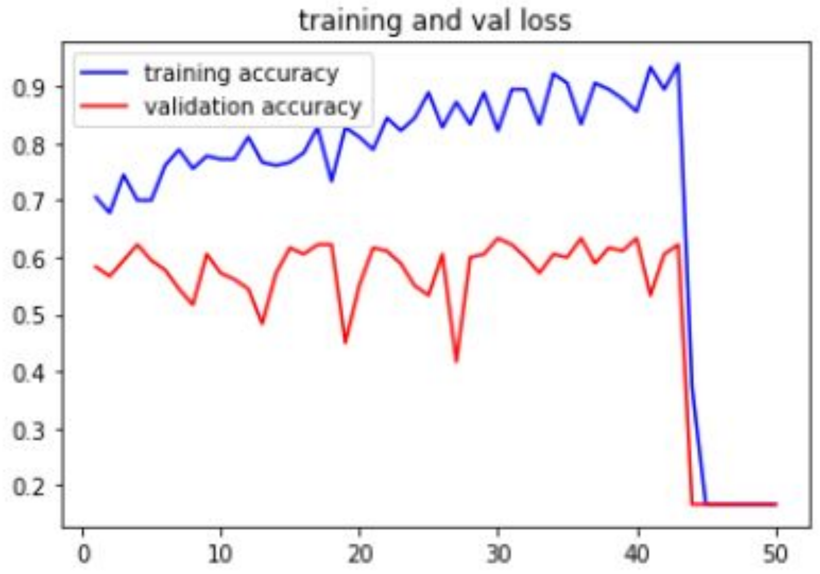


training and val loss
training accuracy
validation accuracy
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
10
20
30
40
50

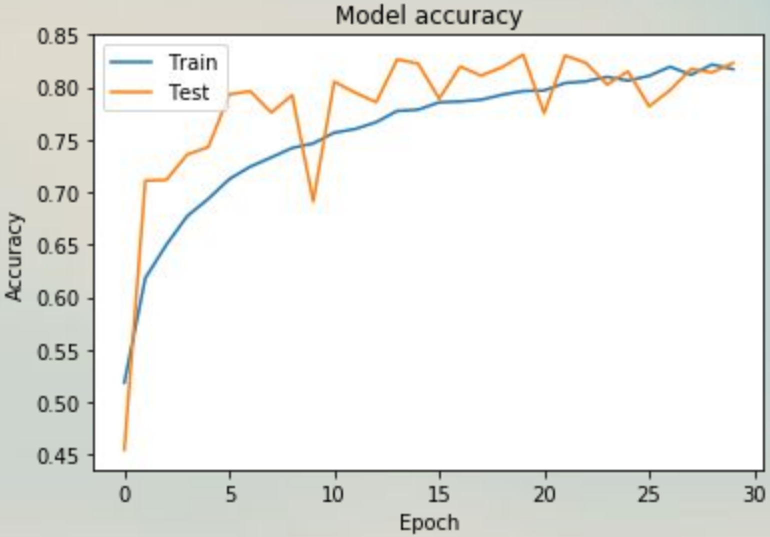
```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_18 (Conv2D)           (None, 148, 148, 32)      896
_________________________________________________________________
activation_28 (Activation)   (None, 148, 148, 32)      0
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 72, 72, 32)        9248
_________________________________________________________________
activation_29 (Activation)   (None, 72, 72, 32)        0
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 36, 36, 32)        0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 34, 34, 64)        18496
_________________________________________________________________
activation_30 (Activation)   (None, 34, 34, 64)        0
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 17, 17, 64)        0
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 15, 15, 128)       73856
_________________________________________________________________
activation_31 (Activation)   (None, 15, 15, 128)       0
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 7, 7, 128)         0
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 5, 5, 256)         295168
_________________________________________________________________
activation_32 (Activation)   (None, 5, 5, 256)         0
_________________________________________________________________
max_pooling2d_22 (MaxPooling (None, 2, 2, 256)         0
_________________________________________________________________
flatten_6 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_11 (Dense)             (None, 64)                65600
_________________________________________________________________
activation_33 (Activation)   (None, 64)                0
_________________________________________________________________
dropout_6 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_12 (Dense)             (None, 6)                 390
_________________________________________________________________
activation_34 (Activation)   (None, 6)                 0
=================================================================
Total params: 463,654
Trainable params: 463,654
Non-trainable params: 0
```

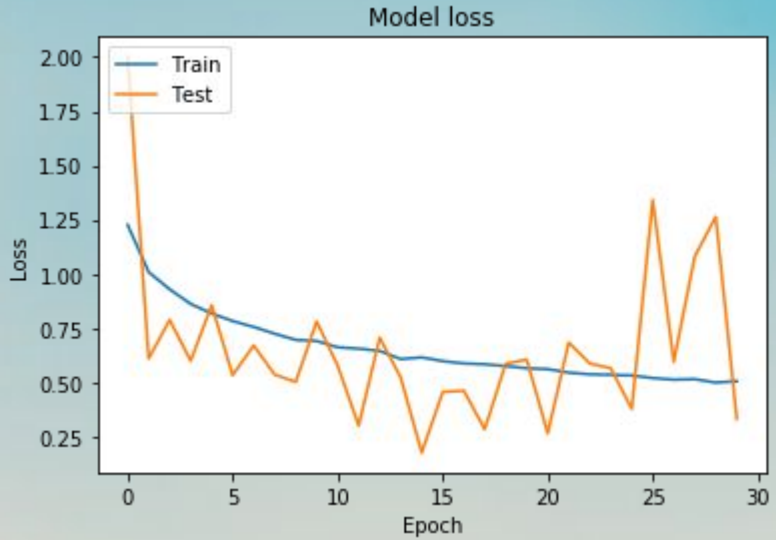
Overfitting

```
Out[18]: <function matplotlib.pyplot.show(*args, **kw)>
```



Results
CNN



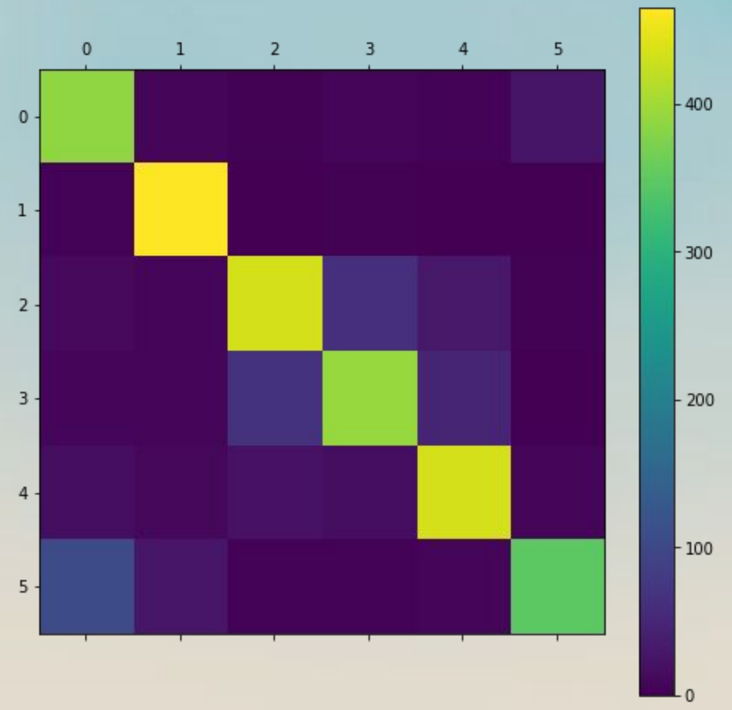


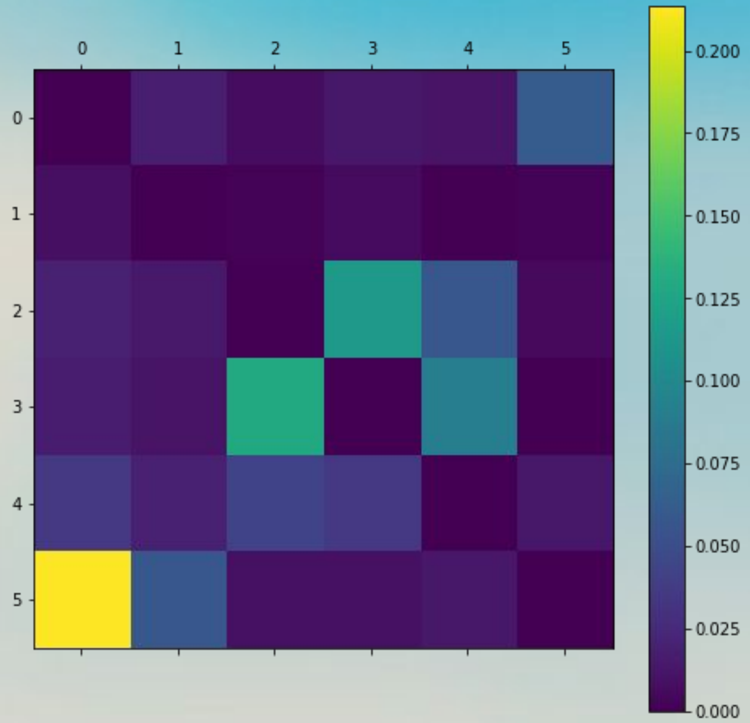
The model does not seem to overfit, it has a better accuracy on the validation set

```
437/437 [==============================] - 789s 2s/step - loss: 0.5440 - accuracy: 0.8040 - val_loss: 0.6835 - val_accuracy: 0.
8302
Epoch 23/30
437/437 [==============================] - 790s 2s/step - loss: 0.5364 - accuracy: 0.8056 - val_loss: 0.5863 - val_accuracy: 0.
8231
Epoch 24/30
437/437 [==============================] - 786s 2s/step - loss: 0.5344 - accuracy: 0.8098 - val_loss: 0.5648 - val_accuracy: 0.
8026
Epoch 25/30
437/437 [==============================] - 794s 2s/step - loss: 0.5318 - accuracy: 0.8063 - val_loss: 0.3771 - val_accuracy: 0.
8150
Epoch 26/30
437/437 [==============================] - 789s 2s/step - loss: 0.5188 - accuracy: 0.8108 - val_loss: 1.3408 - val_accuracy: 0.
7817
Epoch 27/30
437/437 [==============================] - 785s 2s/step - loss: 0.5123 - accuracy: 0.8195 - val_loss: 0.5945 - val_accuracy: 0.
7972
Epoch 28/30
437/437 [==============================] - 789s 2s/step - loss: 0.5140 - accuracy: 0.8118 - val_loss: 1.0801 - val_accuracy: 0.
8174
Epoch 29/30
437/437 [==============================] - 782s 2s/step - loss: 0.4984 - accuracy: 0.8216 - val_loss: 1.2630 - val_accuracy: 0.
8140
Epoch 30/30
437/437 [==============================] - 784s 2s/step - loss: 0.5051 - accuracy: 0.8172 - val_loss: 0.3312 - val_accuracy: 0.
8231
```

The confusion matrix will show us the most frequent mistakes made by this classifier

	B	F	G	M	S	St
B	[[388	8	3	6	5	27]
F	[4	465	1	3	0	1]
G	[11	8	435	64	32	3]
M	[9	6	68	394	48	0]
S	[18	10	22	18	435	7]
St	[107	29	5	5	7	348]]





This matrix shows us that the most common confusions are between **streets and buildings**, there are also confusions between **glaciers and mountains**.

Images of streets classified as buildings

Let's look at some badly classified images

```
#Images of buildings classified as streets
errors(y_pred,0,5,10)
```

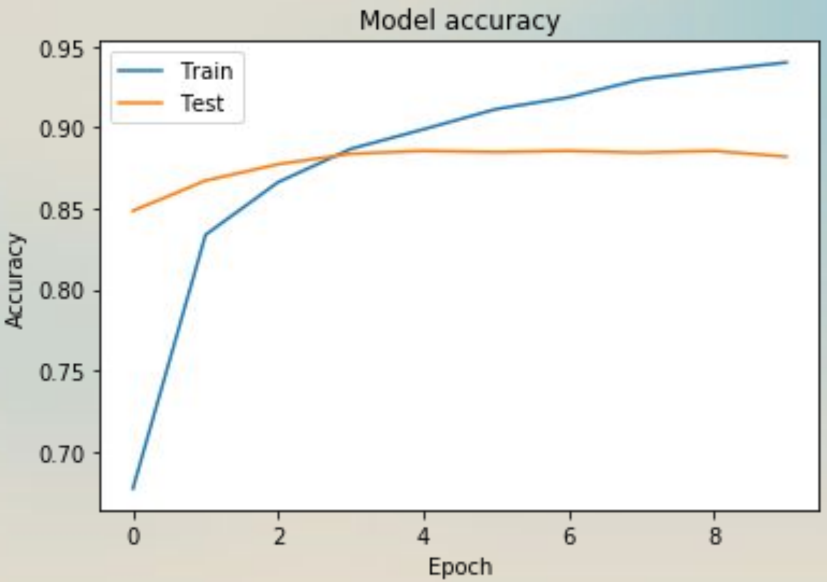
Images of buildings classified as streets

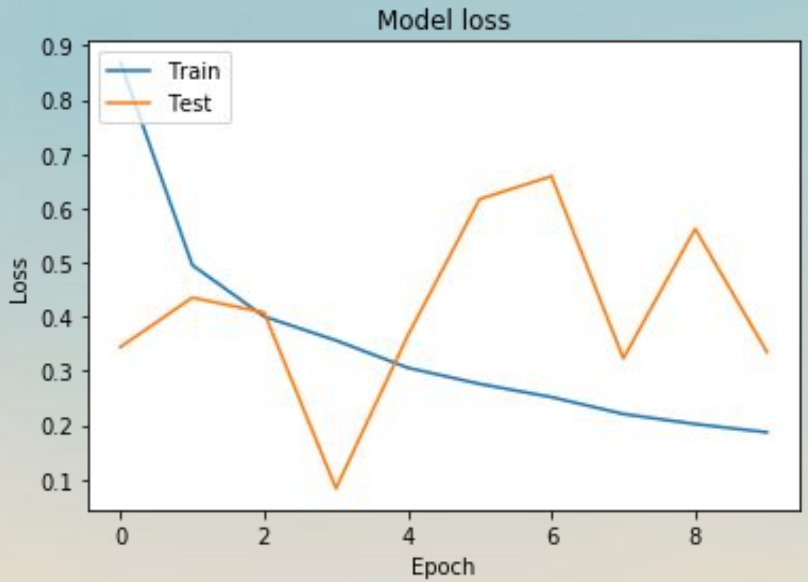
```
#Images of mountains classified as glaciers
errors(y_pred,3,2,10)
```

Images of mountains classified as glaciers

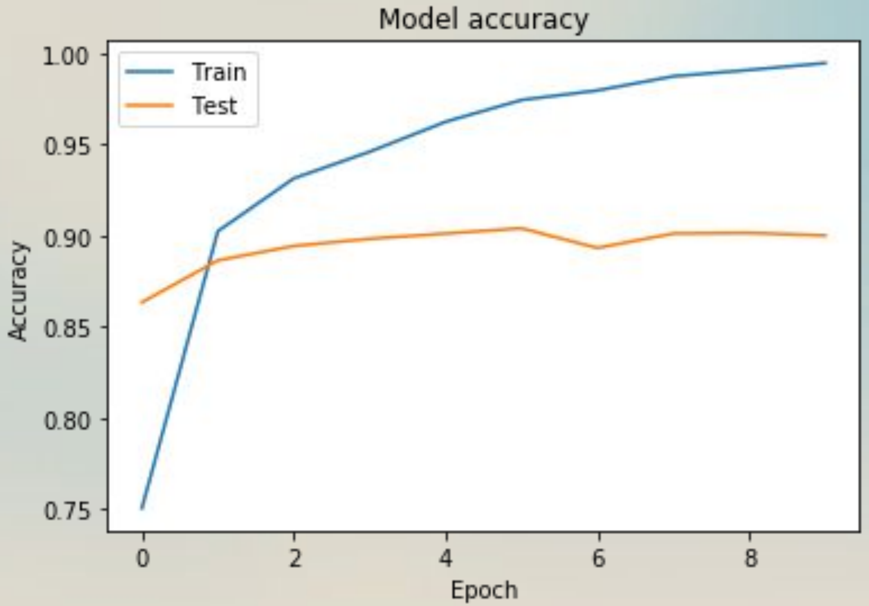
Transfer learning(VGG16)

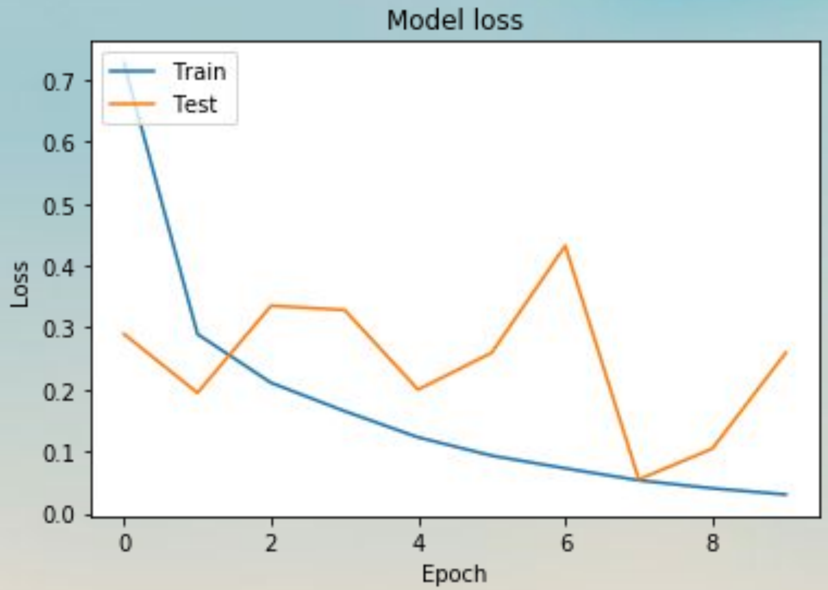
Training:





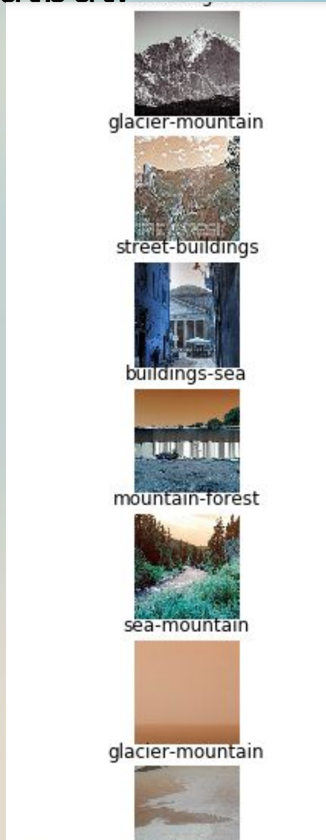
Testing

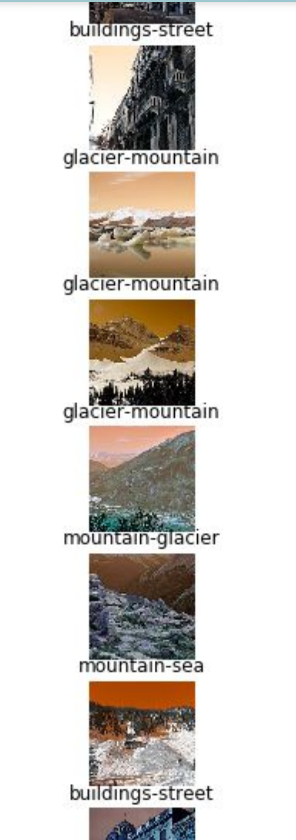

Model accuracy
Train
Test
1.00
0.95
0.90
0.85
0.80
0.75
Accuracy
0
2
4
6
8
Epoch


Model loss
Train
Test
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Loss
0
2
4
6
8
Epoch

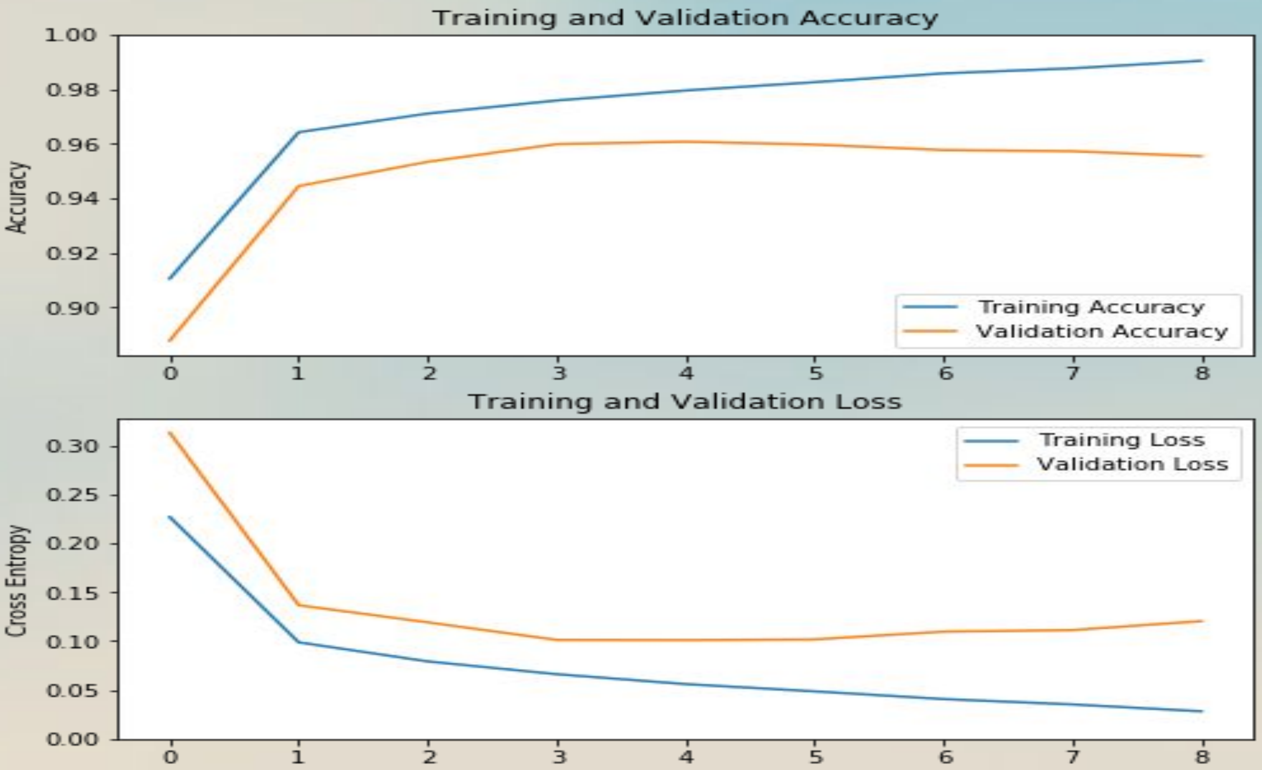
Accuracy:91%

269 wrongly predicted output.





MobileNet



Training and Validation Accuracy
Accuracy
1.00
0.98
0.96
0.94
0.92
0.90
Training Accuracy
Validation Accuracy
0
1
2
3
4
5
6
7
8
Training and Validation Loss
Cross Entropy
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Training Loss
Validation Loss

Conclusion

In this project, we have used different classification models for classifying images into their respective categories. But, in the final classification, a few of the images were misclassified. Therefore, in order to improve the accuracy of classification, it is necessary to include additional knowledge about discarding the image.

In future work, it would be interesting to include additional feature information.

References


THANK YOU