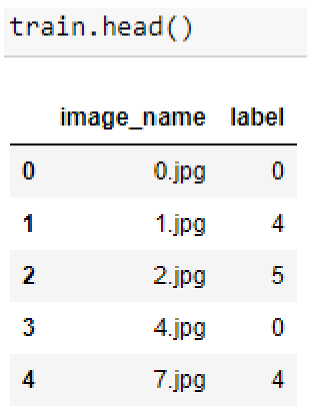
Approach file

• Data Description:

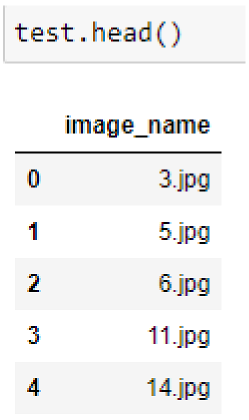
There are **17034 images in train and 7301 images in test** data. The categories of scenes and their corresponding codes in the dataset are as follows -

```
{Building:0,
 Forest: 1,
 Glacier: 2,
 Mountain: 3,
 Sea: 4,
 Street: 5}
```



```
train.head()
```

	image_name	label
0	0.jpg	0
1	1.jpg	4
2	2.jpg	5
3	4.jpg	0
4	7.jpg	4

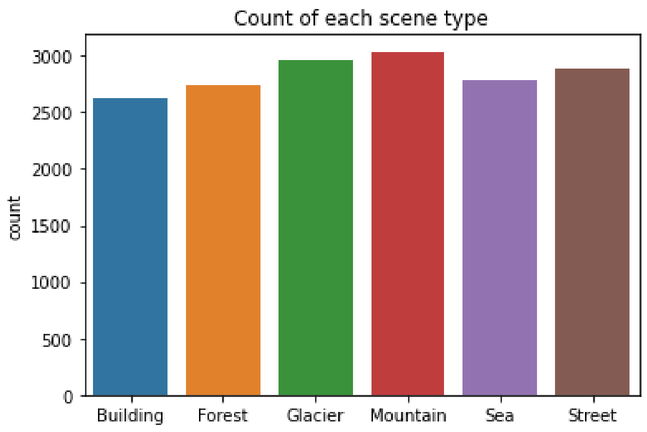


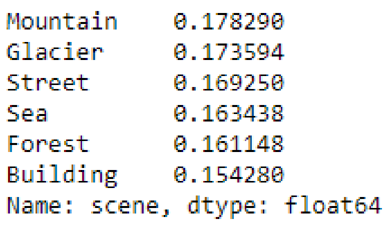
```
test.head()
```

	image_name
0	3.jpg
1	5.jpg
2	6.jpg
3	11.jpg
4	14.jpg

• EDA and Data preprocessing:

1. There are no null values in the training set.
2. The distribution of data in the various category of scenes are as follows:



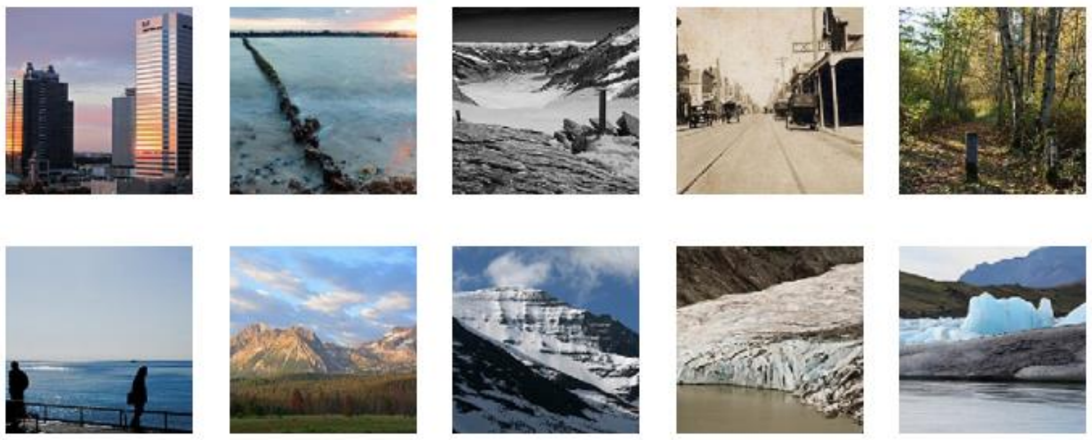


```
Mountain     0.178290
Glacier      0.173594
Street       0.169250
Sea          0.163438
Forest       0.161148
Building     0.154280
Name: scene, dtype: float64
```

So, all the classes have same proportion of data. There is no class imbalance.

3. While loading the image, we reshape each image to a particular shape, because they have to be of same size while feeding to model. Here we choose the size (224,224,3). As , they are colored images, the number of channels are kept 3.
4. The images are changed into 2D array, with the pixel values normalized. Because the computation of high numeric values is more complex. The pixel values are made to have value between 0-1. The pixel values of RGB image lies from 0 to 255. So, we divide each pixel value with 255.
5. Some images from the train and test set are visualized.

Training set

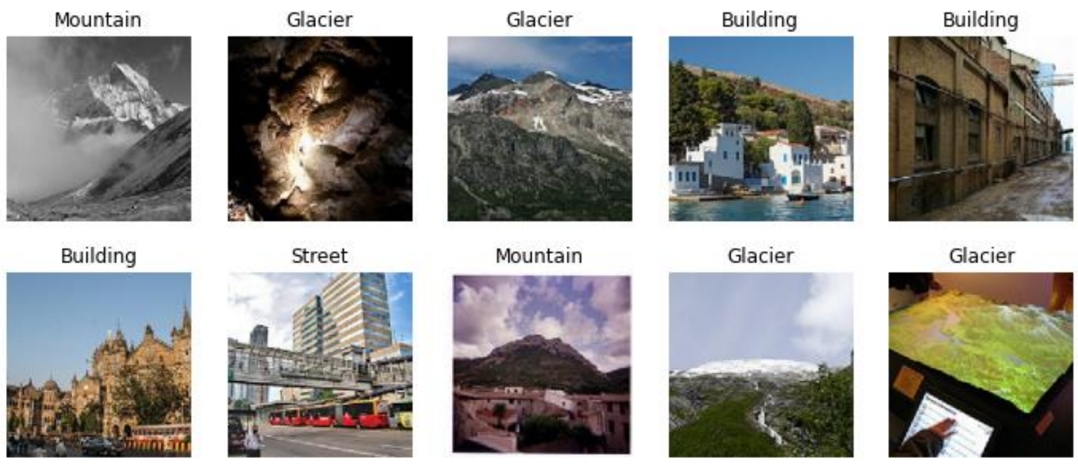


Test Set

So, in the test set, there are greyscale images with colour images. Also, some images contain a large portion of the background. So, to address these issues, we will need to perform data augmentation to create a more robust training set. Data augmentation create modified versions of training set images, by cropping a part of the picture, changing the intensity of RGB channels, flipping them etc. This helps creating a more generalized training set, so that accuracy on the unseen test data increases. This will also prevent overfitting.

6. There may be some confusing images in the training set. Because the images labelled street has a portion of building. Also, the pictures of mountain and glacier can resemble a lot. So, we remove images with wrong predictions having confidence less than 0.55.

   We train a xception pretrained model for 2 epochs on the whole training set and it gives 90% accuracy on training set. Then we use this model to predict the probability of each training images. We find 327 such images with less than 0.55 confidence. Such images are shown below.



So, we can see that the third image can be both mountain and glacier but labeled as glacier. Or, the 7th image of street has a large proportion of building. So, we drop these images and make a new data set for training. So, now we have 16707 images in training set.

7. The category column contains values from 0 to 5. As these are labels, we hot encode the column into six different columns. As, we will use the data for train test split, we convert it into numpy array.

• Splitting the data into train and validation set:

The data is split in 80% training and 20% validation set. We use stratify as mentioned earlier.

```
X_train, X_val, y_train, y_val = train_test_split(new_imgs, new_y, test_size=0.20, random_state=1)
```

• Data Augmentation:

The training data is augmented and a generator is made for feeding into model.

```
BS = 32
gen = ImageDataGenerator(rotation_range=45,
                         horizontal_flip=True,
                         width_shift_range=0.5,
                         height_shift_range=0.5)
```

• Transfer learning:

CNN model contains **convolutional** layers followed by **fully connected** layers. Convolution layers extract features from the image and fully connected layers classify the image using extracted features. When we train a **CNN** on image data, It is seen that top layers of the network learn to extract **general** features from images such as edges, distribution of colours, etc. As we keep going deep in the network, the layers tend to extract more **specific** features. Now we can use these pretrained models which already know how to extract features and avoid the training from scratch.

So, similarly here pretrained models are used. Xception,Resnet50 and Inception Resnetv2 all are convolutional neural networks with respectively 71, 50 and 164 layers. They are already pretrained on millions of images. So, we keep the pretrained weights in the model. We use a sequential model with base as the pretrained models. Then we attach our own classifier, so we disable the default classifier by setting include_top=False. After the convolutional layer, a average pooling layer is used. The layer also turns the output 2D, as the Dense layer only accepts 2D. Then we use softmax at Dense layer, since it is a non-binary classification.

Since the model is pretrained, we don't need to train for large epochs. So, here 20 epochs are run with initial learning rate as 0.0001, which keeps decreasing at each epoch.

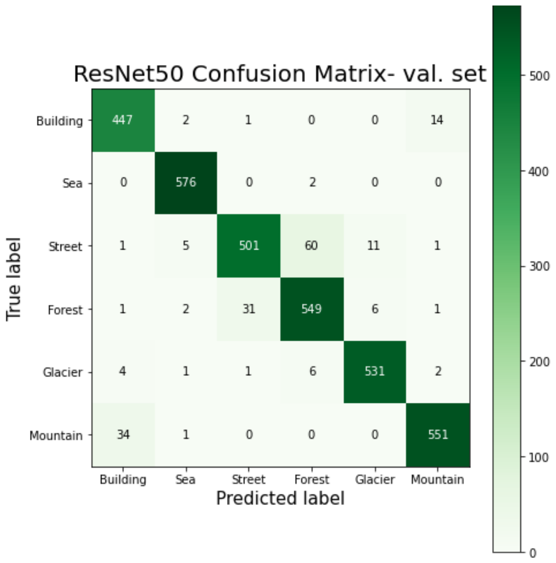
To prevent overfitting we use early stopping with patience 5 and criteria as minimum validation loss. So, if the validation less does not decrease for 5 epochs, the training stops. Also, we use model checkpoint to save the best model with minimum validation loss.

• Model Evaluation:

The accuarcy and confusion matrix are shown below:

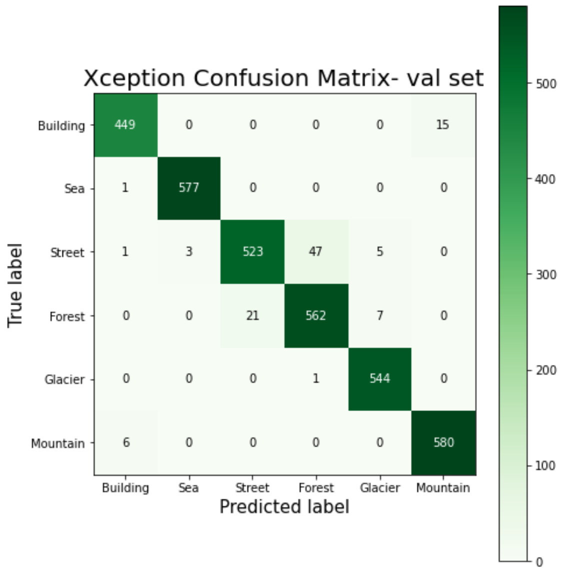
1) ResNet50:

```
Training accuracy:  0.9426139282307184
Validation accuracy:  0.9292632814793073
```



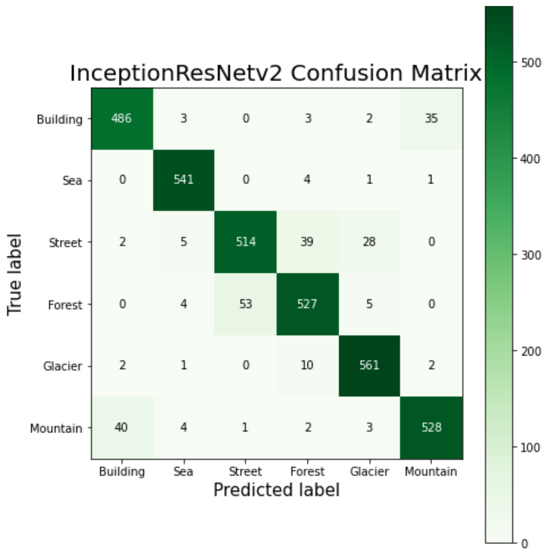
2) Xception:

```
Training accuracy:  0.9708666617744184
Validation accuracy:  0.9424713824479014
```



3) Inception Resnet v2:

```
Training accuracy:  0.9600058706978792
Validation accuracy:  0.9266216612855885
```



- **Ensemble:**

  We all ensemble all the models which gives more than 93% accuracy on test set and get a 96.75% accuracy ultimately.

- **Conclusion:**

  - We can see from the confusion matrices, that forest and street are the most confusing classes.
  - Ensembling all the models with good val accuracy increases test accuracy.
  - As the model is pretrained, we could have frozen some layers , because it is taking a really long time to train the whole model .
  - Optimal learning rate could have been applied.