Team members:

- *Ahmad Mohammad Abdel-Hafeez*
- *Ahmad Osama Mohammad*
- *Asmaa Ali El-Sheikh*
- *Ayaalla Mohammad El-Tabey*
- *Bassant Ehab Moustafa*

Project Description :

- The scene recognition is an area of visual recognition where recognizes and identifies scene of the image. The scene recognition identifies what the image as a whole is about, or where it is taken, like if it's a picture of sea, image of zoo, photo of a party, and so on. It doesn't classify a certain object in the picture, but tries to recognize the whole scene of the picture.

- **The dataset for the project consists of 6 classes :**
    - Buildings
    - Forest
    - Glacier
    - Mountain
    - Sea
    - Street

The project is a contest on Kaggle

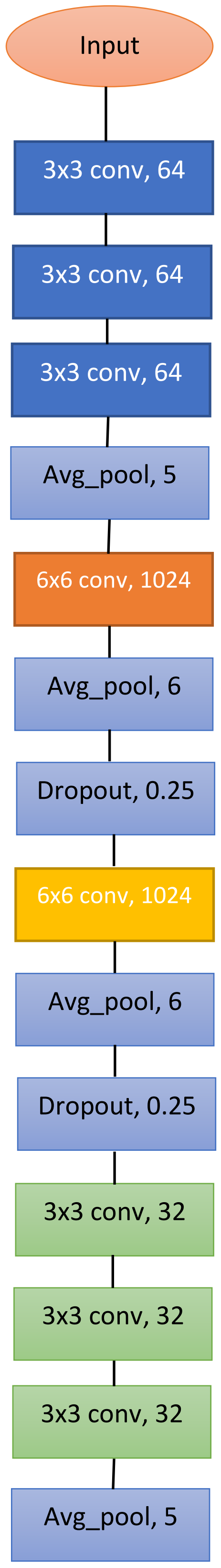
Kaggle is a community and site for hosting machine learning competitions. Competitive machine learning can be a great way to develop and practice your skills, as well as demonstrate your capabilities

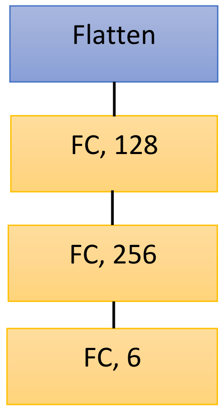
Using Google Colab as IDE for Python Notebook :

Google Colab is a free cloud service and now it supports free GPU! You can: improve your Python programming language coding skills.

Solution method :

We create CNN model with many layers and its architecture is like the following




Flatten
FC, 128
FC, 256
FC, 6

Used layers:-

Conv

Conv2D: is a 2D Convolution Layer, this layer creates a convolution kernel that is wind with layers input which helps produce a tensor of outputs.
Kernel: In image processing kernel is a convolution matrix or masks which can be used for blurring, sharpening, embossing, edge detection, and more by doing a convolution between a kernel and an image.

```
Conv_2d(filters, kernel_size,activation=None, use_bias=True)
```

Pooling

Avg_pool: is a layer, reducethe spatial dimensionality of each feature map (but not depth) (reduce the amount of parameters and computation in the network) and build in invariance to small transformations.

```
Avg_Pool(pool_size)
```

Dropout

dropout: is a layer which randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by 1/(1 - rate) such that the sum over all inputs is unchanged.

```
dropout(rate)
```

Flatten

Flatten: is a layer which used to flatten the input. For example, if flatten is applied to layer having input shape as (2,2), then the output shape of the layer will be (4).

```
flatten(input)
```

Fully Connected

fully_connected: is a traditional Multilayer Perceptron MLP that uses a Softmax activation function in the output layer.

```
fully_connected(input,num_outputs,activation_fn)
```

The regression

Regression: is a layer which used in TFLearn to apply a regression (linear or logistic) to the provided input. It requires to specify a TensorFlow gradient descent optimizer 'optimizer' that will minimize the provided loss function 'loss' (which calculate the errors). A metric can also be provided, to evaluate the model performance.

```
regression(input, optimizer, learning_rate, loss, name)
```

Deep Neural Network Model

```
tflearn.DNN (input, tensorboard_dir)
```

To improve our model, we used data augmentation.

Data augmentation: in data analysis are techniques used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data. It acts as a regularizer and helps reduce overfitting when training a machine learning model. It is closely related to oversampling in data analysis.

```
datagen = ImageDataGenerator(brightness_range = (0.5, 1.5))
for img in train:
  x = img[0]
  x = x.reshape((1, ) + x.shape)
  i = 0
  for batch in datagen.flow(x, batch_size = 1):
    trainSet.append([batch, img[1]])
    i += 1
    if i > 1:
      break
```

which increase the accuracy.

Trails:

Update in model	Accuracy
Gray image 150x150, 2 FC	81%
3 FC	82%
Avg_ pool instead of Max_pool	84%
RGB image 150x150	86%
Testing on 5000 instead of 4000	87%
RGB image 200x200	87.5%
Data augmentation zoom=0.3	89%
zoom=0.3, brightness (0.5,1.5)	90%
Only brightness (0.5, 1.5)	91%