CNN - Convolution Neural Networks

- It is used if the inputs are in the form of image.

Types of Images:

1. Binary Images - Two types of colours ie. Black and White

1. Gray Scale Image - It has an intensity range added to two existing colours. Black and White and Intensity of colours between Black and White

1. Colored Image - It has RGB - Red, Green and Blue colours

CNN Layers:

1. Convolution Layer
2. Pooling Layer
3. Flatten Layer
4. Fully Connected Layer

1. Convolution Layer

When you apply convolution (*) on an image and a Feature Detector it will create a Feature Map.

Image * Feature Detector = Feature Map

Once we got this Feature Map:

2. Pooling Layer

- We will be using **Max Pooling** on Feature Map to identify features in different type of pictures.
- Our Model must be robust enough to identify the image even with variations i.e. Spatial Invariance.
- Here, Important features will be kept as **Pooled Featured Maps** and unimportant features (noise) will be removed.
- Doing this we are reducing 75% of the unwanted feature

3. Flatten Layer

- Here, the 'n' no. of Pooled Featured Map will be converted to one dimensional image.
- This Flatten Layer will the input for the ANN.

4. Fully Connected Layer

- This is the 'n' no. of Pooled Featured Maps acting as an Input to ANN

Flatten Layers --> Hidden Layers --> Output Layer

To Check a visual represenation in Google type 'scs.ryerson.ca aharley'

Step 1: Importing Libraries

In [1]:

```
from keras.models import Sequential  #initializing the layers
from keras.layers import Dense  #building layers
from keras.layers import Conv2D  #Creating Convolution Layer
from keras.layers import MaxPooling2D  #Max Pooling 2D
from keras.layers import Flatten  #Flatten Layer
```

```
Using TensorFlow backend.
```

Step 2: Initializing the Model with Sequential Layer

In [2]:

```
model = Sequential ()
```

Step 3: Adding Convolution Layer

In [3]:

```
model.add(Conv2D(32,3,3,input_shape = (64,64,3), activation = 'relu'))


#1st parameter in conv2D = no. of Feature detectors
#2nd &3rd parmater = size of feat. Detect.
#4th parameter = Expected input image shape
#5th parameter =Activation
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
UserWarning: Update your `Conv2D` call to the Keras 2 API: `Conv2D(3
2, (3, 3), input_shape=(64, 64, 3..., activation="relu")`
  """Entry point for launching an IPython kernel.
```

Step 4: Adding Max Pooling Layer

In [4]:

```
model.add(MaxPooling2D(pool_size = (2,2)))
```

Step 5: Flatten Layer

It converts any 'n' dimensional image into one dimentional image.

In [5]:

```
model.add(Flatten())
```

To check the number of layers added into the Model of Neural Network

In [6]:

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 62, 62, 32)        896       
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 31, 31, 32)        0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 30752)             0         
=================================================================
Total params: 896
Trainable params: 896
Non-trainable params: 0
_________________________________________________________________
```

Step 6: ANN Layers or Fulling Connected Layers

In [7]:

```
model.add(Dense(output_dim = 128, activation = 'relu', init = 'random_uniform'))
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
UserWarning: Update your `Dense` call to the Keras 2 API: `Dense(act
ivation="relu", units=128, kernel_initializer="random_uniform")`
  """Entry point for launching an IPython kernel.
```

Output Layer

In [8]:

```
model.add(Dense(output_dim  = 1, activation = 'sigmoid', init = 'random_uniform'
))
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
UserWarning: Update your `Dense` call to the Keras 2 API: `Dense(act
ivation="sigmoid", units=1, kernel_initializer="random_uniform")`
  """Entry point for launching an IPython kernel.
```

In [9]:

```
model.compile(optimizer = 'adam', loss ='binary_crossentropy', metrics=['accurac
y'])
```

In [10]:

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 62, 62, 32)        896       
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 31, 31, 32)        0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 30752)             0         
_________________________________________________________________
dense_1 (Dense)              (None, 128)               3936384   
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 129       
=================================================================
Total params: 3,937,409
Trainable params: 3,937,409
Non-trainable params: 0
_________________________________________________________________
```

Step 7: Image Processing - Image Data Generation

In [11]:

```
from keras.preprocessing.image import ImageDataGenerator
```

In [12]:

```
train_datagen = ImageDataGenerator (rescale = 1./255, shear_range = 0.2, zoom_ra
nge = 0.2, horizontal_flip = True)
test_datagen = ImageDataGenerator (rescale = 1./255)
```

Step 8: Input Images Data

Here,

batch_size - How many image to be sent at once for processing

class_mode - Number of types of image categories, if its two types use 'binary' and if its more than two types use'category'.

In [13]:

```
x_train = train_datagen.flow_from_directory(r'dataset/training_set', target_size
=(64,64), batch_size = 32, class_mode = 'binary')
#if types of d=images data is more ethan 2 class_mode - 'categorical'

x_test = test_datagen.flow_from_directory(r'dataset/test_set', target_size =(64,
64), batch_size = 32, class_mode = 'binary')
```

```
Found 8100 images belonging to 3 classes.
Found 2056 images belonging to 3 classes.
```

In [14]:

```
print(x_train.class_indices)
```

```
{'cats': 0, 'dogs': 1, 'elephant': 2}
```

In [15]:

```
#samples_per_epochs is the number of input data images
#validation_data is used for testing parallely with the test data

model.fit_generator(x_train, samples_per_epoch = 8000, epochs = 1, validation_da
ta = x_test, nb_val_samples=2000)
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:4:
UserWarning: The semantics of the Keras 2 argument `steps_per_epoch`
is not the same as the Keras 1 argument `samples_per_epoch`. `steps_
per_epoch` is the number of batches to draw from the generator at ea
ch epoch. Basically steps_per_epoch = samples_per_epoch/batch_size.
Similarly `nb_val_samples`->`validation_steps` and `val_samples`->`s
teps` arguments have changed. Update your method calls accordingly.
  after removing the cwd from sys.path.
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:4:
UserWarning: Update your `fit_generator` call to the Keras 2 API: `f
it_generator(<keras.pre..., epochs=1, validation_data=<keras.pre...,
steps_per_epoch=250, validation_steps=2000)`
  after removing the cwd from sys.path.

Epoch 1/1
250/250 [==============================] - 335s 1s/step - loss: 0.67
07 - accuracy: 0.5820 - val_loss: 0.5776 - val_accuracy: 0.5118
```

Out[15]:

```
<keras.callbacks.callbacks.History at 0x63a04d750>
```

In [16]:

```
model.save('cat_dog_elephant_cnn.h5')
```

In [ ]:

```
```