CHEATSHEET
CONVOLUTIONAL NEURAL NETWORKS (CNN)

It is also termed as ConvNet that is a **Deep learning** algorithm that inputs an image, draws different feature maps using different kernels that allocates learnable weights to different objects in the image so as to differentiate from one another. The algorithm is widely used for various Image related tasks like classification, segmentation etc. Let us understand more about it.

Convolution Layers

This layer has a different number of kernels with different sizes that are responsible to extract feature maps from the image. There can be any number of convolution layers and it depends upon the programmer.

#Implementation

```
model.add(Convolution2D(32,(3,3)))
```

There are a total 32 filters of each having size 3X3.

Pooling Layers

This layer is usually added after the convolution layer that reduced the dimensions of feature maps.

MaxPooling

This type of pooling layers takes the maximum of the pixel that lies under the kernel.

#Implementation

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

Average Pooling

```
#Implementation
model.add(AveragePooling2D(pool size=2))
```

Stride

It tells how much the kernel should move in each step while capturing the feature maps.

#Implementation

```
model.add(Convolution2D(32,(3,3,stride=(1,1)))
```

Padding

This is used when the kernel does not fit correctly over the input image. In this case, 0s are added to the input image matrix. It has two values that can either be the same or valid.

```
#Implementation
model.add(Convolution2D(32,(3,3,stride=(1,1),padding='same'))

model.add(Convolution2D(32,(3,3,stride=(1,1),padding='valid'))
```

Flatten Layer

This layer is used to flatten the incoming data into 1 dimensional arrays that can be passed to F.C layers.

```
#Implementation
model.add(Flatten())
```

Fully Connected Layers

The flatten layer output is passed to F.C layers that consist of dense layers and the last layer outputs the desired class using activation function.

```
#Implementation
model.add(Dense(units=128,activation='relu')
model.add(Dense(units=128,activation='relu')
model.add(Dense(units=2,activation='softmax')
```