Convultion Neural Network

```
import numpy as np
from numpy import asarray
```

```
data =[
    [3,3,2,1,0],
    [0,0,1,3,1],
    [3,1,2,2,3],
    [2,0,0,2,2],
    [2,0,0,0,1]
]
```

```
data = np.asarray(data)
data
```

```
     array([[3, 3, 2, 1, 0],
            [0, 0, 1, 3, 1],
            [3, 1, 2, 2, 3],
            [2, 0, 0, 2, 2],
            [2, 0, 0, 0, 1]])
```

```
data.shape
```

```
     (5, 5)
```

```
data_flatten = data.reshape(1,5,5,1)
data_flatten
```

```
     array([[[[3],
              [3],
              [2],
              [1],
              [0]],

             [[0],
              [0],
              [1],
              [3],
              [1]],

             [[3],
              [1],
              [2],
              [2],
              [3]],

             [[2],
               [0],
               [0],
               [2],
               [2]],

              [[2],
               [0],
               [0],
               [0],
               [1]]]])
```

```
data_flatten.shape
```

```
     (1, 5, 5, 1)
```

weights is also know as a filters

```
kernel = [
    [ [[0]],[[1]],[[2]] ],
    [ [[2]],[[2]],[[0]] ],
    [ [[0]],[[1]],[[2]] ]

]
```

```
weights = [asarray(kernel),asarray([0.0])]
weights
```

```
     [array([[[[0]],

               [[1]],

               [[2]]],


              [[[2]],

               [[2]],

               [[0]]],


              [[[0]],

               [[1]],

               [[2]]]]), array([0.])]
```

Simple CNN

```
from keras.models import Sequential
from keras.layers import Conv2D

model = Sequential()


model.add(Conv2D(1,(3,3),input_shape=(5,5,1)))


model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 3, 3, 1)           10        
                                                                 
=================================================================
Total params: 10
Trainable params: 10
Non-trainable params: 0
_________________________________________________________________
```

```
model.set_weights(weights)


pred = model.predict(data_flatten)
pred
```

```
1/1 [==============================] - 1s 562ms/step
array([[[[12.],
         [12.],
         [17.]],

        [[10.],
         [17.],
         [19.]],

        [[ 9.],
         [ 6.],
         [14.]]]], dtype=float32)
```

```
for i in range(pred.shape[3]):
  print(i)
```

```
0
```

```
pred.shape[0]
```

```
1
```

```
pred.shape[1]
```

```
3
```

```
pred.shape[3]
```

```
1
```

```
pred.shape
```

```
(1, 3, 3, 1)
```

```
for r in range(pred.shape[1]):
  print([pred[0,r,c,0] for c in range(pred.shape[2])])
```

```
[12.0, 12.0, 17.0]
[10.0, 17.0, 19.0]
[9.0, 6.0, 14.0]
```

CNN With Stride 2 : Default stride is 1 we change it 2

```
model1 = Sequential()
model1.add(Conv2D(1,(3,3),input_shape=(5,5,1),strides=(2,2)))
model1.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_1 (Conv2D)           (None, 2, 2, 1)           10

=================================================================
Total params: 10
Trainable params: 10
Non-trainable params: 0
_________________________________________________________________
```

```
model1.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_1 (Conv2D)           (None, 2, 2, 1)           10

=================================================================
Total params: 10
Trainable params: 10
Non-trainable params: 0
_________________________________________________________________
```

```
model1.set_weights(weights)
```

```
pred=model1.predict(data_flatten)
pred
```

```
1/1 [==============================] - 0s 18ms/step
array([[[[12.],
         [17.]],

        [[ 9.],
         [14.]]]], dtype=float32)
```

```
for r in range(pred.shape[1]):
  print([pred[0,r,c,0] for c in range(pred.shape[2])])
```

```
[12.0, 17.0]
[9.0, 14.0]
```

CNN with Padding

```
model2  = Sequential()
model2.add(Conv2D(1,(3,3),padding='same',input_shape=(5,5,1)))
model2.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 5, 5, 1)           10

=================================================================
Total params: 10
Trainable params: 10
Non-trainable params: 0
_________________________________________________________________
```

```
model2.set_weights(weights)
```

```
pred = model2.predict(data_flatten)
pred
```

```
1/1 [==============================] - 0s 37ms/step
array([[[[ 6.],
         [14.],
         [17.],
         [11.],
           [ 3.]],

          [[14.],
           [12.],
           [12.],
           [17.],
           [11.]],

          [[ 8.],
           [10.],
           [17.],
           [19.],
           [13.]],

          [[11.],
           [ 9.],
           [ 6.],
           [14.],
           [12.]],

          [[ 6.],
           [ 4.],
           [ 4.],
           [ 6.],
           [ 4.]]]], dtype=float32)
```

```
pred.shape
```

```
(1, 5, 5, 1)
```

```
for r in range(pred.shape[1]):
  print([pred[0,r,c,0] for c in range(pred.shape[2])])
```

```
[6.0, 14.0, 17.0, 11.0, 3.0]
[14.0, 12.0, 12.0, 17.0, 11.0]
[8.0, 10.0, 17.0, 19.0, 13.0]
[11.0, 9.0, 6.0, 14.0, 12.0]
[6.0, 4.0, 4.0, 6.0, 4.0]
```

CNN with Pooling

```
from keras.layers import MaxPooling2D
```

```
model3 = Sequential()
model3.add(Conv2D(1,(3,3),padding='same',input_shape=(5,5,1)))
model3.add(MaxPooling2D((2,2)))
model3.summary()
```

```
Model: "sequential_6"
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 5, 5, 1)           10

 max_pooling2d (MaxPooling2D  (None, 2, 2, 1)          0
 )

=================================================================
Total params: 10
Trainable params: 10
Non-trainable params: 0
_________________________________________________________________
```

```
model3.set_weights(weights)
```

```
pred=model3.predict(data_flatten)
pred
```

```
1/1 [==============================] - 0s 100ms/step
array([[[[14.],
         [17.]],

        [[11.],
         [19.]]]], dtype=float32)
```

```
for r in range(pred.shape[1]):
  print([pred[0,r,c,0] for c in range(pred.shape[2])])
```

```
[14.0, 17.0]
[11.0, 19.0]
```

CNN with Flatten

```
from keras.layers import Flatten
from keras.layers import AveragePooling2D
```

```
model4 = Sequential()
model4.add(Conv2D(1,(3,3),padding='same',input_shape=(5,5,1)))
model4.add(AveragePooling2D())
model4.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_7 (Conv2D)           (None, 5, 5, 1)           10
 average_pooling2d (AverageP  (None, 2, 2, 1)         0
 ooling2D)

=================================================================
Total params: 10
Trainable params: 10
Non-trainable params: 0
_________________________________________________________________
```

```
model4.set_weights(weights)
```

```
pred=model4.predict(data_flatten)
pred
```

```
WARNING:tensorflow:5 out of the last 6 calls to <function Model.make_predict_function.<l
1/1 [==============================] - 0s 45ms/step
array([[[[11.5 ],
         [14.25]],

        [[ 9.5 ],
         [14.  ]]]], dtype=float32)
```

```
for r in range(pred.shape[1]):
  print([pred[0,r,c,0] for c in range(pred.shape[2])])
```

```
[11.5, 14.25]
[9.5, 14.0]
```

✓ 0s completed at 9:36 PM