Convolutional Neural netwok(CNN) ¶

The convolutional layer is the core building block of a CNN, and it is where the majority of computation occurs. It requires a few components, which are input data, a filter, and a feature map. Let’s assume that the input will be a color image, which is made up of a matrix of pixels in 3D. This means that the input will have three dimensions—a height, width, and depth—which correspond to RGB in an image. We also have a feature detector, also known as a kernel or a filter, which will move across the receptive fields of the image, checking if the feature is present. This process is known as a convolution.

In [1]:

```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K
import numpy as np
```

In [2]:

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datas
ets/mnist.npz (https://storage.googleapis.com/tensorflow/tf-keras-datasets/mni
st.npz)
11490434/11490434 [==============================] - 2s 0us/step
```

In [3]:

```
img_rows, img_cols = 28, 28

if K.image_data_format() == 'channels_first':
   x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
   x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
   input_shape = (1, img_rows, img_cols)
else:
   x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
   x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
   input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255

y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)
```

In [5]:

```
model = Sequential()
model.add(Conv2D(32, kernel_size = (3, 3),
   activation = 'relu', input_shape = input_shape))
model.add(Conv2D(64, (3, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.25)) , model.add(Flatten())
model.add(Dense(128, activation = 'relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation = 'softmax'))
```

In [6]:
```
model.compile(loss = keras.losses.categorical_crossentropy,
   optimizer = keras.optimizers.Adadelta(), metrics = ['accuracy'])
```

In [7]:
```
model.fit(
   x_train, y_train,
   batch_size = 128,
   epochs = 12,
   verbose = 1,
   validation_data = (x_test, y_test)
)
```

```
Epoch 1/12
469/469 [==============================] - 97s 205ms/step - loss: 2.2782 - acc
uracy: 0.1654 - val_loss: 2.2462 - val_accuracy: 0.3393
Epoch 2/12
469/469 [==============================] - 91s 193ms/step - loss: 2.2283 - acc
uracy: 0.2842 - val_loss: 2.1860 - val_accuracy: 0.5698
Epoch 3/12
469/469 [==============================] - 96s 204ms/step - loss: 2.1653 - acc
uracy: 0.3792 - val_loss: 2.1047 - val_accuracy: 0.6451
Epoch 4/12
469/469 [==============================] - 96s 205ms/step - loss: 2.0771 - acc
uracy: 0.4524 - val_loss: 1.9928 - val_accuracy: 0.6710
Epoch 5/12
469/469 [==============================] - 96s 204ms/step - loss: 1.9633 - acc
uracy: 0.5031 - val_loss: 1.8476 - val_accuracy: 0.7002
Epoch 6/12
469/469 [==============================] - 97s 208ms/step - loss: 1.8186 - acc
uracy: 0.5452 - val_loss: 1.6696 - val_accuracy: 0.7383
Epoch 7/12
469/469 [==============================] - 96s 205ms/step - loss: 1.6567 - acc
uracy: 0.5794 - val_loss: 1.4714 - val_accuracy: 0.7690
Epoch 8/12
469/469 [==============================] - 98s 210ms/step - loss: 1.4867 - acc
uracy: 0.6086 - val_loss: 1.2771 - val_accuracy: 0.7910
Epoch 9/12
469/469 [==============================] - 98s 209ms/step - loss: 1.3380 - acc
uracy: 0.6332 - val_loss: 1.1076 - val_accuracy: 0.8102
Epoch 10/12
469/469 [==============================] - 99s 211ms/step - loss: 1.2136 - acc
uracy: 0.6551 - val_loss: 0.9707 - val_accuracy: 0.8220
Epoch 11/12
469/469 [==============================] - 100s 213ms/step - loss: 1.1104 - ac
curacy: 0.6790 - val_loss: 0.8624 - val_accuracy: 0.8308
Epoch 12/12
469/469 [==============================] - 101s 214ms/step - loss: 1.0256 - ac
curacy: 0.6982 - val_loss: 0.7784 - val_accuracy: 0.8389
```

Out[7]:
```
<keras.callbacks.History at 0x21d4ce70c40>
```

In [8]:
```
score = model.evaluate(x_test, y_test, verbose = 0)

print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.7784239053726196
Test accuracy: 0.8389000296592712
```

In [9]:

```
pred = model.predict(x_test)
pred = np.argmax(pred, axis = 1)[:5]
label = np.argmax(y_test,axis = 1)[:5]

print(pred)
print(label)
```

```
313/313 [==============================] - 4s 12ms/step
[7 2 1 0 4]
[7 2 1 0 4]
```

In [ ]: