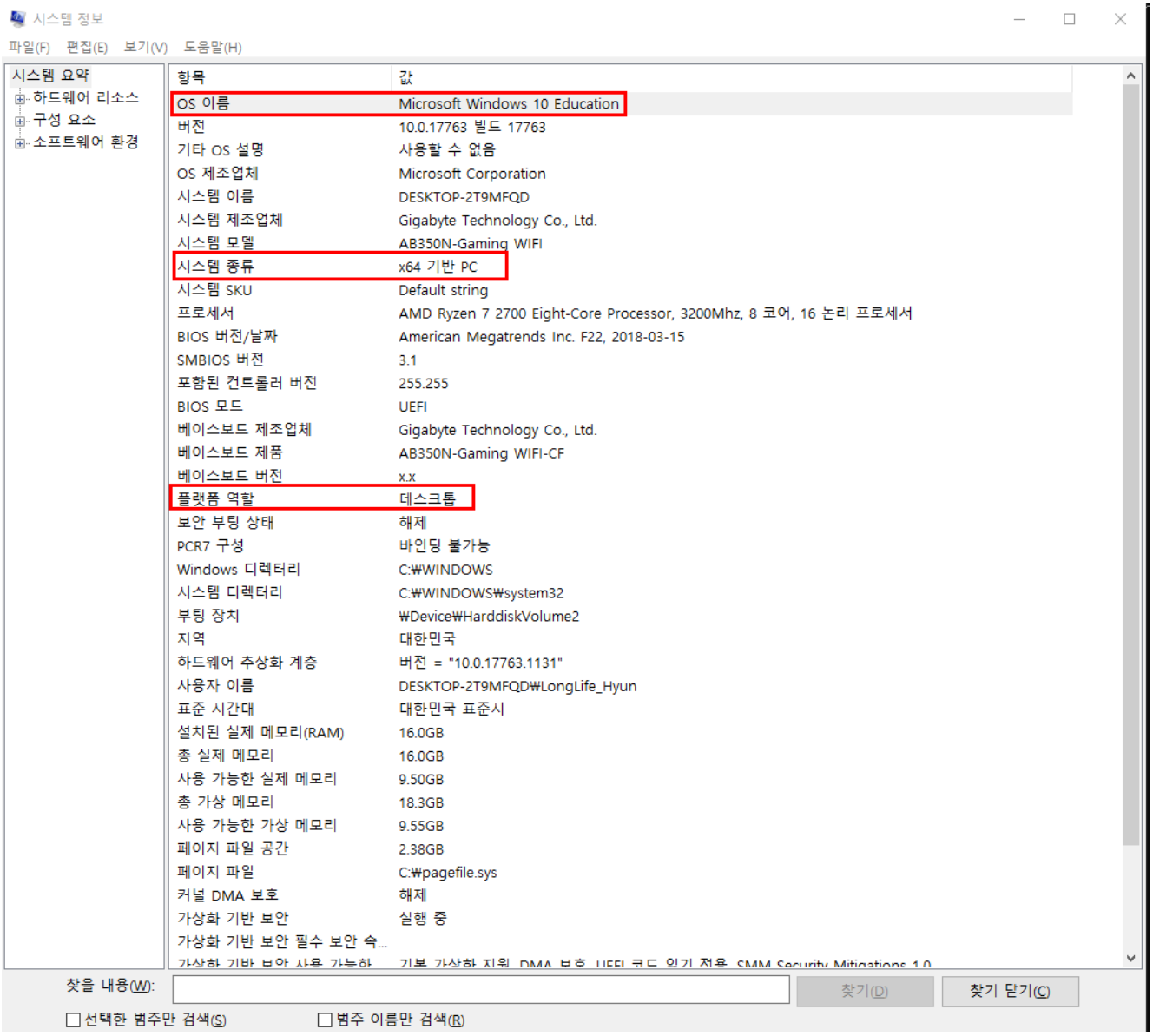
머신러닝 과제 3

201724557 장수현

1. 딥러닝 개발환경 (Development Environment)

딥러닝 개발을 위한 나의 개발환경은 다음과 같다.

PC

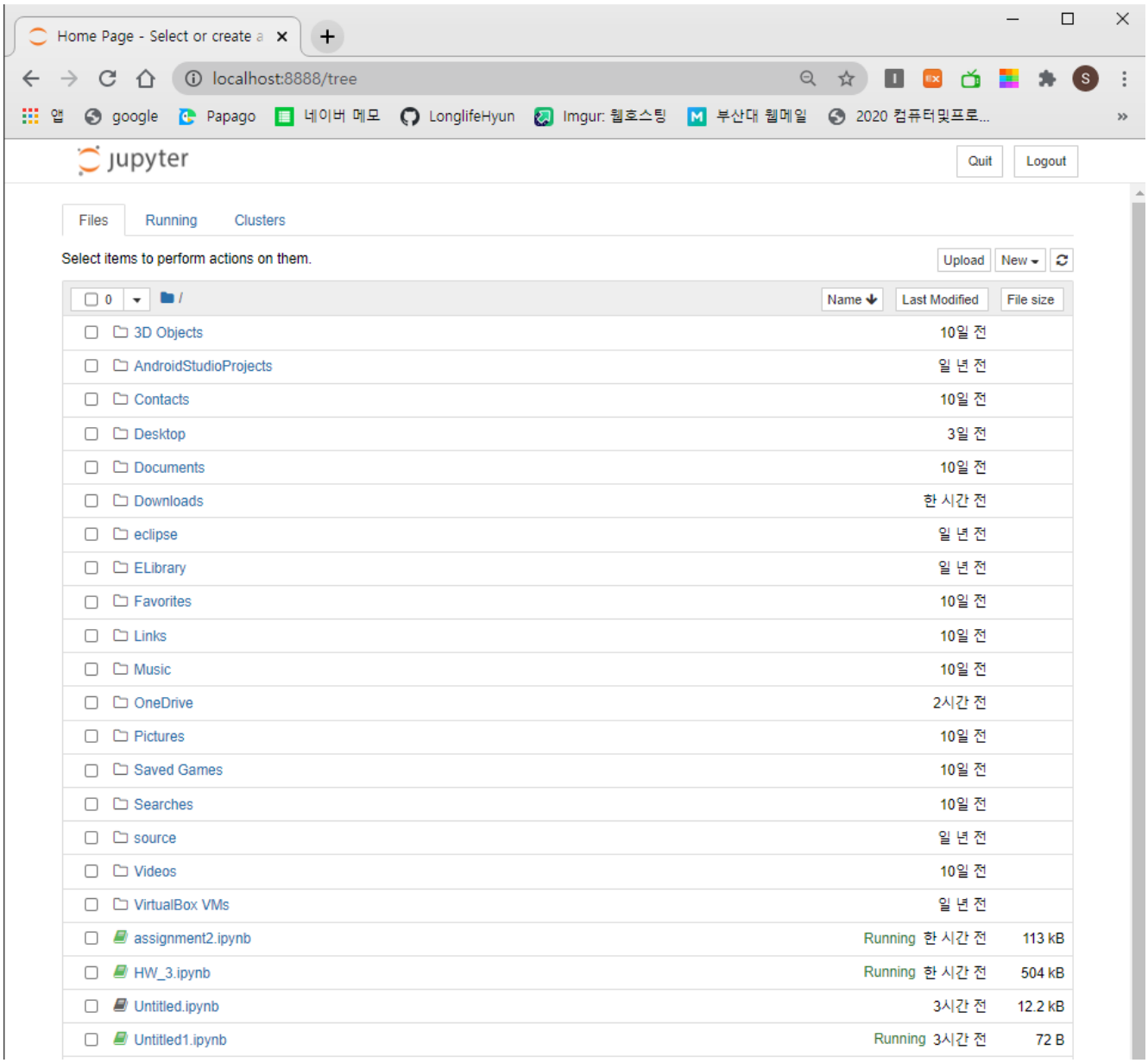


PC : 64비트 기반 데스크탑
OS : MS Windows 10 Education

jupyter notebook & Python & Tensorflow

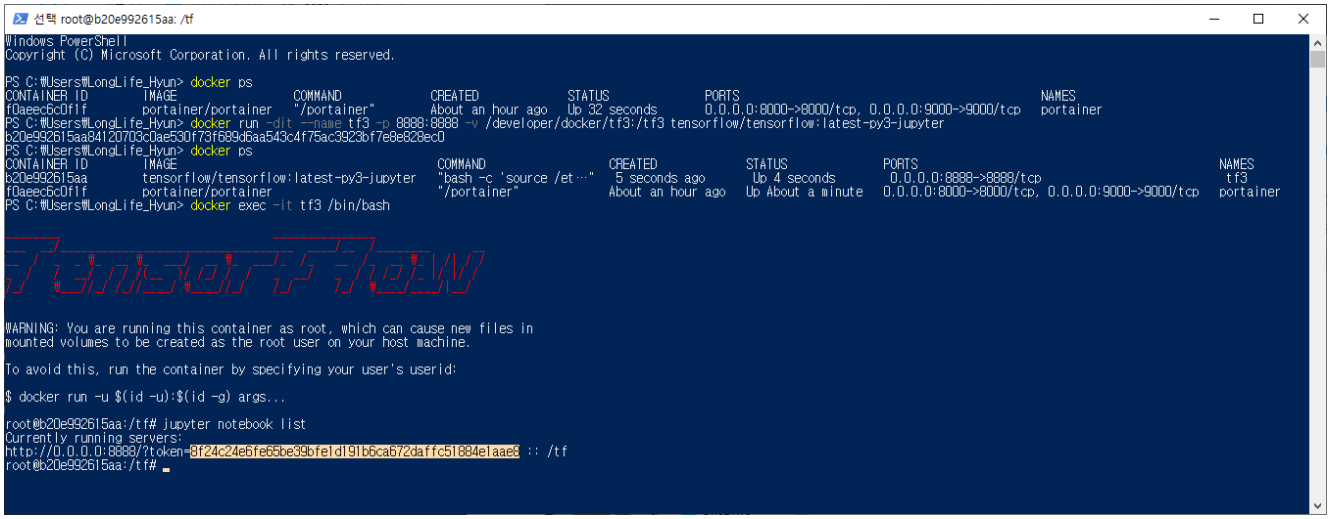
```
6.0.3

C:₩Users₩LongLife_Hyun>python --version
Python 3.7.3

C:₩Users₩LongLife_Hyun>python
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
2020-06-26 22:43:44.450963: W tensorflow/stream_executor/platform/default/dso_loader.cc:55] Could
 not load dynamic library 'cudart64_101.dll'; dlerror: cudart64_101.dll not found
2020-06-26 22:43:44.465494: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cud
art dlerror if you do not have a GPU set up on your machine.
>>> print(tf.__version__)
2.2.0
>>>
```

jupyter notebook 실행 화면



기타

원래는 Docker for Window로 개발 환경을 구축하려 했다. 그러나 graphviz path관련 오류를 해결하지 못하여 결국 windows 환경에서 과제를 진행하게 되었다.



2. 모델 생성 (Making Models)

tensorflow 불러오기

`import tensorflow as tf` 를 사용하여 tensorflow를 tf로 사용한다.(파이썬 문법)
tensorflow 중에서 **keras**를 사용하여 모델을 만들어 보겠다.

In [1]:

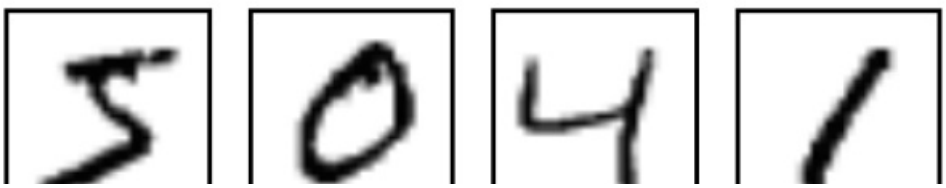
```
import tensorflow as tf
from tensorflow.keras import Input, Model
from tensorflow.keras.layers import *
```

Python을 사용하는데 필요한 라이브러리 호출하기

In [2]:

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

MNIST - 사람의 손글씨를 모아 놓은 데이터



훈련 데이터 6만개, 테스트 데이터 1만개로 구성되어 있으며 tensorflow의 내장 함수를 사용하여 쉽게 불러올 수 있다.
각 데이터는 2차원의 이미지 형태 이며, 가로 28픽셀 세로 28픽셀로 28x28데이터 이다.
각 데이터 포인트는 0~255사이 값을 가지고 있다.

In [26]:

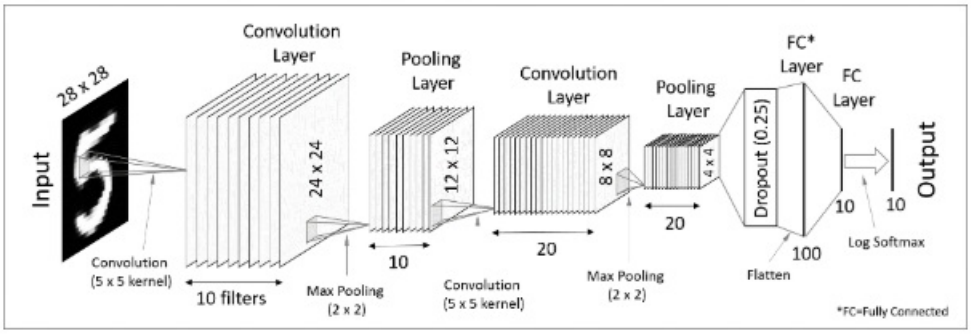
```
mnist = tf.keras.datasets.mnist
```

In [4]:

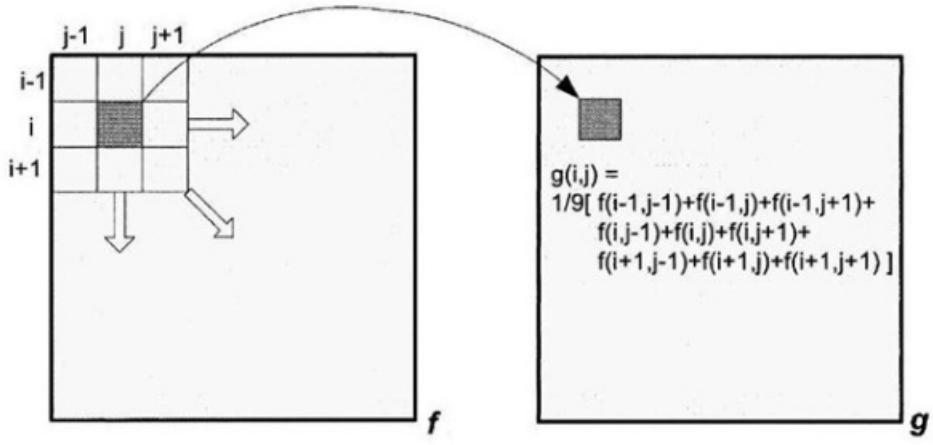
```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
```

Deep learning을 할 때 데이터가 0~1 사이의 값을 가지는 것이 학습이 잘된다.
따라서 학습 데이터를 최대값 255로 나누어서 0~1 사이 값을 가지도록 만든다.

CNN 사용하여 모델 만들기



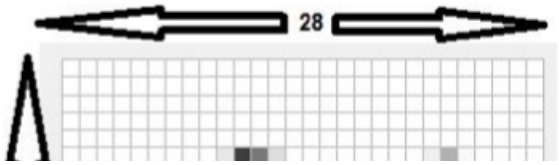
CNN 모델은 2차원 데이터는 2차원 형태로 분석하고자 하는 것이다.
과거 이미지 처리에서 사용한 mask를 사용하여 이미지를 분석하던 것과 비슷한 방법이다.

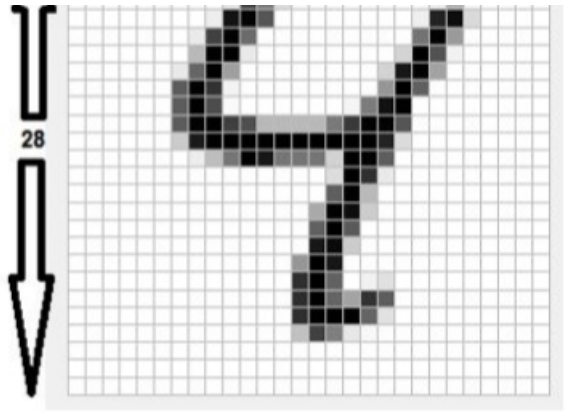


CNN은 이 mask를 deep learning을 사용하여 학습해 보는 것이다.

데이터 변환

먼저 CNN을 사용하기 위해서는 3차원 데이터 형태로 만들어야 한다.
2차원으로 각 데이터의 위치를 지정해주고 데이터의 값 차원을 추가해서 총 3차원으로 만든다.





In [5]:

```
train_img = x_train.reshape((-1,28,28,1))
test_img = x_test.reshape((-1,28,28,1))
```

CNN 모델 생성

CNN layer + MaxPooling layer + Dropout layer를 1개의 셋트로 총 2개 만들고
최종 결과를 출력하기 위해서 Flatten layer와 Dense layer를 통해 판별한다.

Case1 : Dropout = 0.1

In [6]:

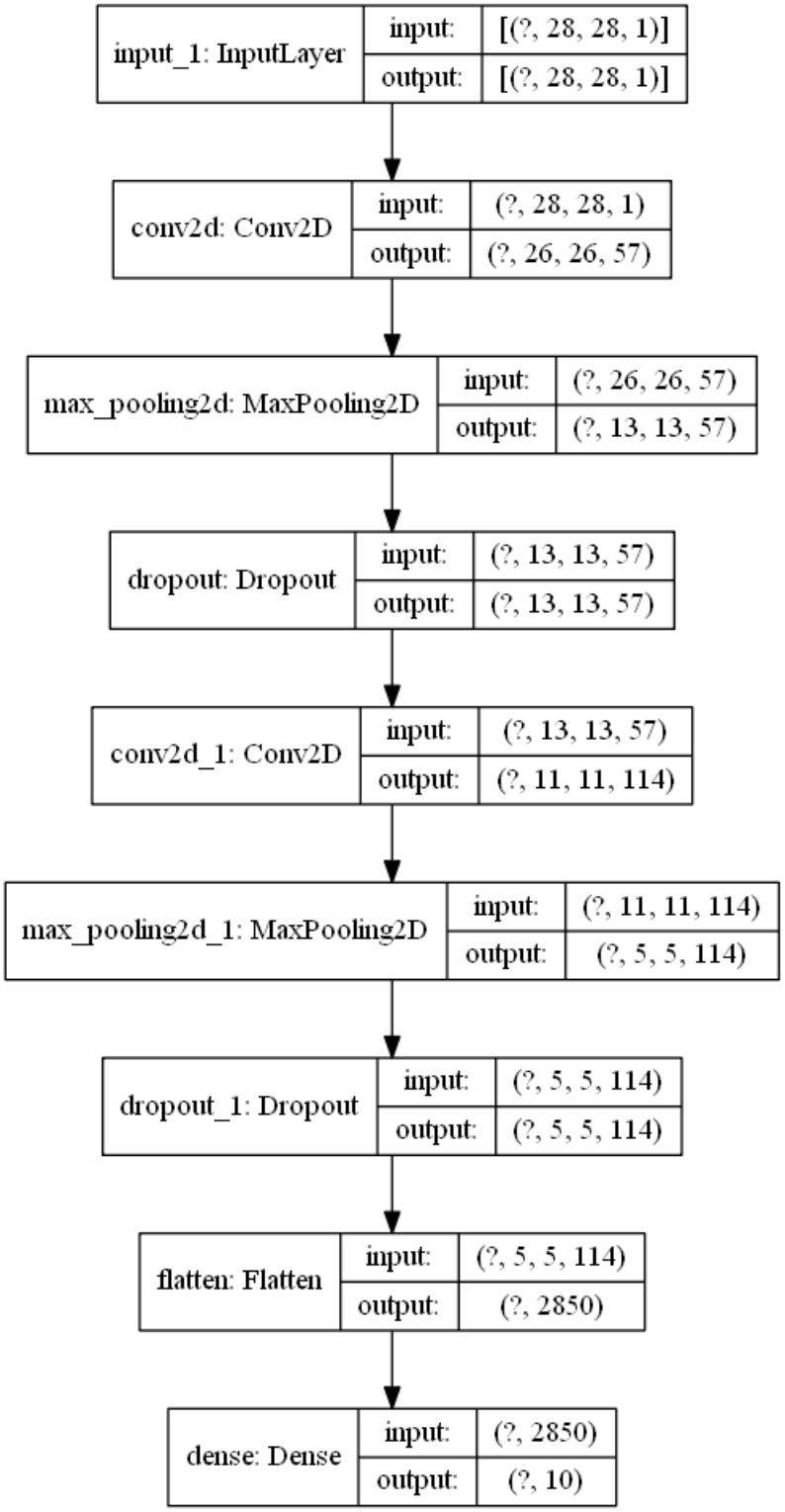
```
CNN_input = Input(shape = (28,28,1))
CNN = Conv2D(57,(3,3),activation='relu')(CNN_input)
CNN = MaxPool2D((2,2))(CNN)
CNN = Dropout(0.1)(CNN)
CNN = Conv2D(114,(3,3),activation='relu')(CNN)
CNN = MaxPool2D((2,2))(CNN)
CNN = Dropout(0.1)(CNN)
CNN = Flatten()(CNN)
CNN = Dense(10,activation='softmax')(CNN)
CNN_model = Model(CNN_input,CNN)
CNN_model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 28, 28, 1)]       0
_________________________________________________________________
conv2d (Conv2D)              (None, 26, 26, 57)        570
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 57)        0
_________________________________________________________________
dropout (Dropout)            (None, 13, 13, 57)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 114)       58596
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 114)         0
_________________________________________________________________
dropout_1 (Dropout)          (None, 5, 5, 114)         0
_________________________________________________________________
flatten (Flatten)            (None, 2850)              0
_________________________________________________________________
dense (Dense)                (None, 10)                28510
=================================================================
Total params: 87,676
Trainable params: 87,676
Non-trainable params: 0
_________________________________________________________________
```

In [7]:

```
tf.keras.utils.plot_model(CNN_model, show_shapes=True)
```

Out[7]:



In [8]:

```
CNN_model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
CNN_model.fit(train_img,y_train, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 40s 21ms/step - loss: 0.1391 - accuracy: 0.9576
Epoch 2/5
1875/1875 [==============================] - 40s 21ms/step - loss: 0.0493 - accuracy: 0.9847
Epoch 3/5
1875/1875 [==============================] - 40s 21ms/step - loss: 0.0357 - accuracy: 0.9888
```

```
Epoch 4/5
1875/1875 [==============================] - 40s 21ms/step - loss: 0.0279 - accuracy: 0.9915
Epoch 5/5
1875/1875 [==============================] - 41s 22ms/step - loss: 0.0217 - accuracy: 0.9934
```

Out[8]:

```
<tensorflow.python.keras.callbacks.History at 0x29ff9de1a58>
```

In [9]:

```
CNN_model.evaluate(test_img,  y_test, verbose=2)
```

```
313/313 - 2s - loss: 0.0270 - accuracy: 0.9909
```

Out[9]:

```
[0.027023665606975555, 0.9908999800682068]
```

Case2 : Dropout = 0.4

In [10]:

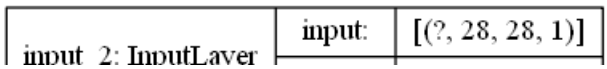
```
CNN_input = Input(shape = (28,28,1))
CNN = Conv2D(57,(3,3),activation='relu')(CNN_input)
CNN = MaxPool2D((2,2))(CNN)
CNN = Dropout(0.4)(CNN)
CNN = Conv2D(114,(3,3),activation='relu')(CNN)
CNN = MaxPool2D((2,2))(CNN)
CNN = Dropout(0.4)(CNN)
CNN = Flatten()(CNN)
CNN = Dense(10,activation='softmax')(CNN)
CNN_model = Model(CNN_input,CNN)
CNN_model.summary()
```

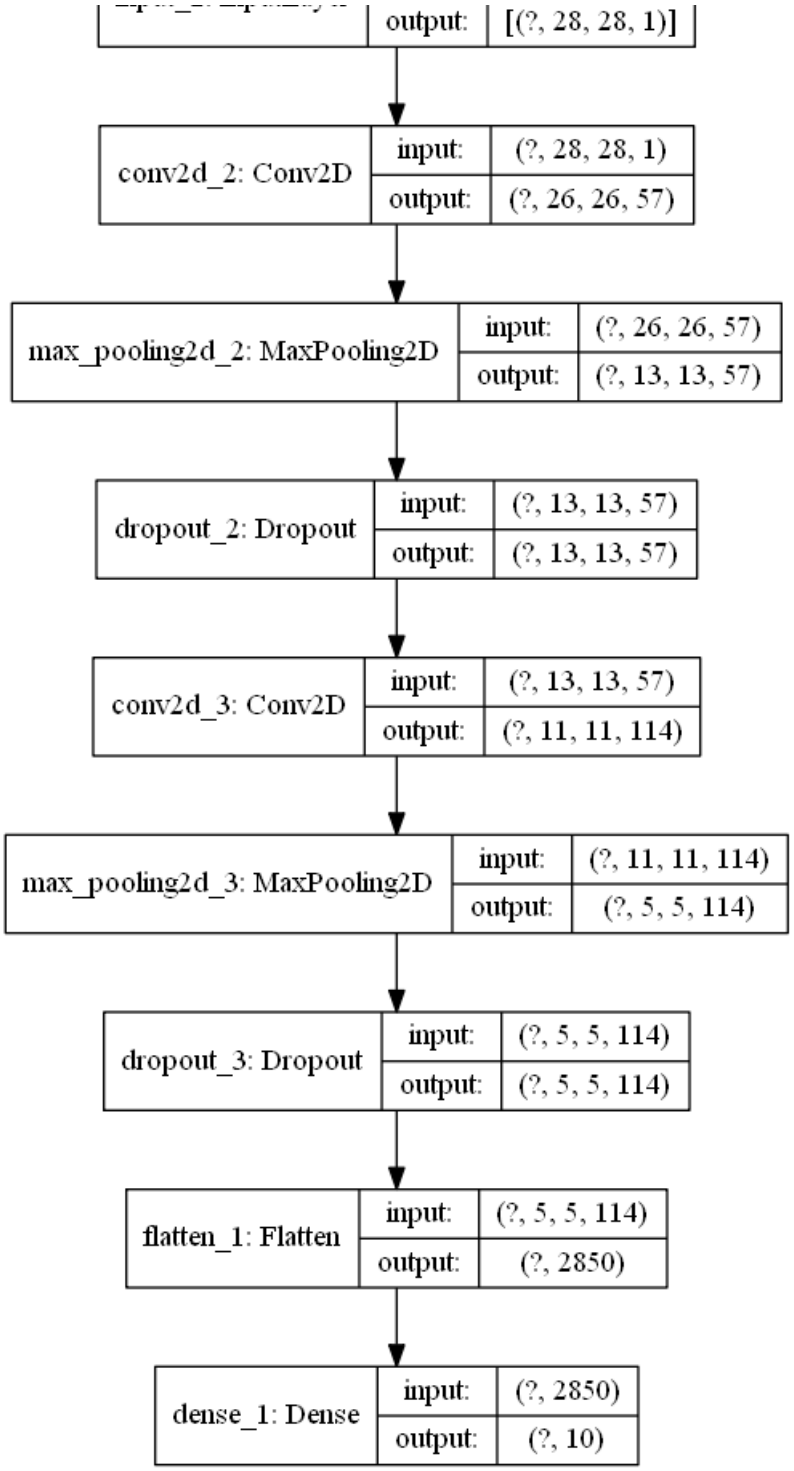
```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 28, 28, 1)]       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 26, 26, 57)        570
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 13, 13, 57)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 13, 13, 57)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 11, 11, 114)       58596
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 5, 5, 114)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 5, 5, 114)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2850)              0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                28510
=================================================================
Total params: 87,676
Trainable params: 87,676
Non-trainable params: 0
_________________________________________________________________
```

In [11]:

```
tf.keras.utils.plot_model(CNN_model, show_shapes=True)
```

Out[11]:





In [12]:

```
CNN_model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
CNN_model.fit(train_img,y_train, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 42s 22ms/step - loss: 0.1910 - accuracy: 0.9400
Epoch 2/5
1875/1875 [==============================] - 42s 22ms/step - loss: 0.0712 - accuracy: 0.9778
Epoch 3/5
1875/1875 [==============================] - 42s 23ms/step - loss: 0.0570 - accuracy: 0.9822
Epoch 4/5
1875/1875 [==============================] - 41s 22ms/step - loss: 0.0493 - accuracy: 0.9852
Epoch 5/5
1875/1875 [==============================] - 41s 22ms/step - loss: 0.0422 - accuracy: 0.9866
```

Out[12]:

```
<tensorflow.python.keras.callbacks.History at 0x29ffb0b3208>
```

In [13]:

```
CNN_model.evaluate(test_img,  y_test, verbose=2)
```

```
313/313 - 2s - loss: 0.0315 - accuracy: 0.9890
```

Out[13]:

```
[0.03151492029428482, 0.9890000224113464]
```

Case3 : Dropout = 0.9

In [14]:

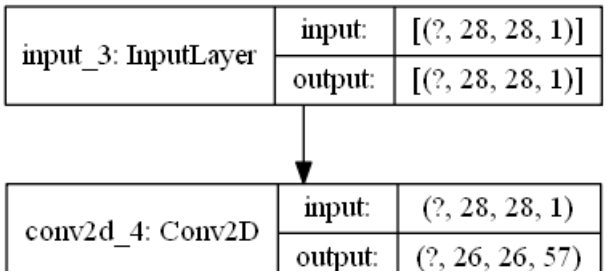
```
CNN_input = Input(shape = (28,28,1))
CNN = Conv2D(57,(3,3),activation='relu')(CNN_input)
CNN = MaxPool2D((2,2))(CNN)
CNN = Dropout(0.9)(CNN)
CNN = Conv2D(114,(3,3),activation='relu')(CNN)
CNN = MaxPool2D((2,2))(CNN)
CNN = Dropout(0.9)(CNN)
CNN = Flatten()(CNN)
CNN = Dense(10,activation='softmax')(CNN)
CNN_model = Model(CNN_input,CNN)
CNN_model.summary()
```

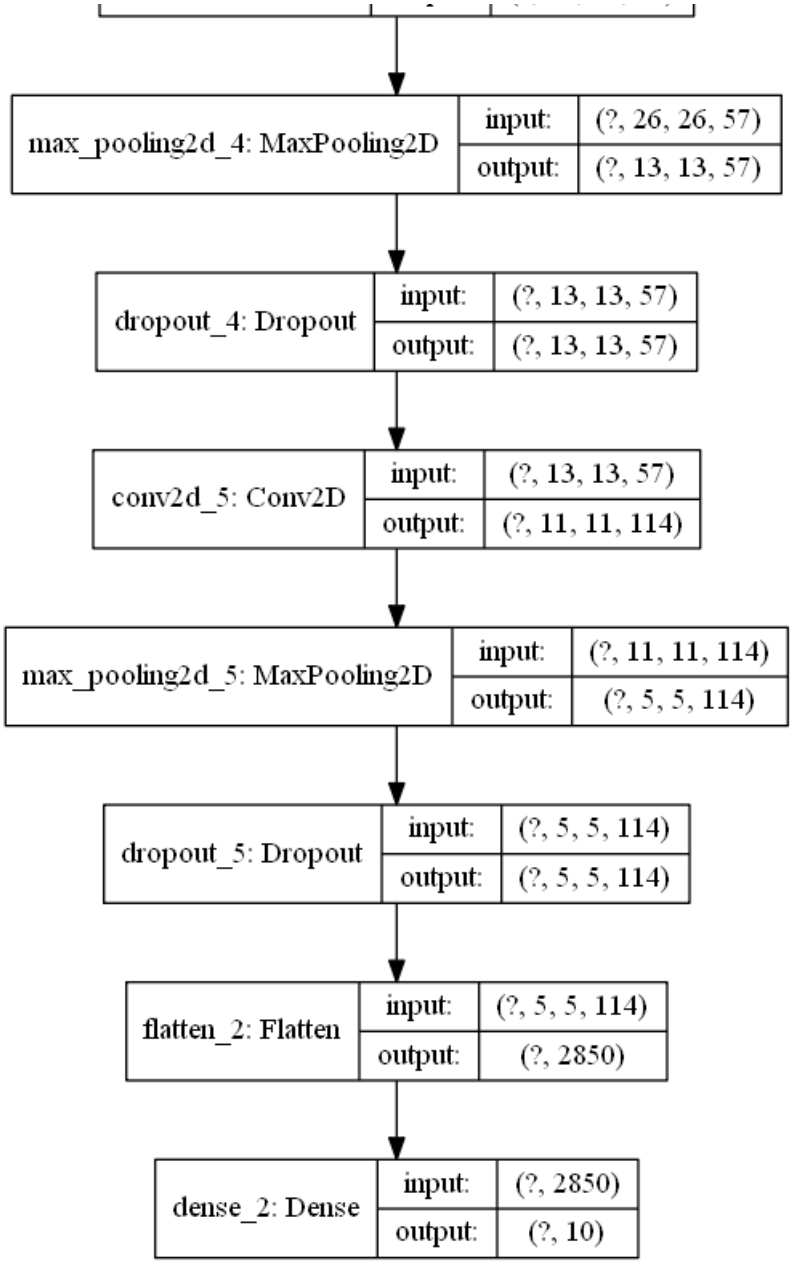
```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 28, 28, 1)]       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 26, 26, 57)        570
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 13, 13, 57)        0
_________________________________________________________________
dropout_4 (Dropout)          (None, 13, 13, 57)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 11, 11, 114)       58596
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 5, 5, 114)         0
_________________________________________________________________
dropout_5 (Dropout)          (None, 5, 5, 114)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 2850)              0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                28510
=================================================================
Total params: 87,676
Trainable params: 87,676
Non-trainable params: 0
_________________________________________________________________
```

In [15]:

```
tf.keras.utils.plot_model(CNN_model, show_shapes=True)
```

Out[15]:





In [16]:

```
CNN_model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
CNN_model.fit(train_img,y_train, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 41s 22ms/step - loss: 0.7939 - accuracy: 0.7386
Epoch 2/5
1875/1875 [==============================] - 41s 22ms/step - loss: 0.4098 - accuracy: 0.8722
Epoch 3/5
1875/1875 [==============================] - 42s 22ms/step - loss: 0.3680 - accuracy: 0.8863
Epoch 4/5
1875/1875 [==============================] - 42s 22ms/step - loss: 0.3465 - accuracy: 0.8921
Epoch 5/5
1875/1875 [==============================] - 42s 22ms/step - loss: 0.3291 - accuracy: 0.8975
```

Out[16]:

```
<tensorflow.python.keras.callbacks.History at 0x29ffb1b5198>
```

In [17]:

```
CNN_model.evaluate(test_img,  y_test, verbose=2)
```

```
313/313 - 2s - loss: 0.1785 - accuracy: 0.9676
```

Out[17]:

```
[0.17854127287864685, 0.9675999879837036]
```

Case4 : Dropout = 0.99

In [18]:

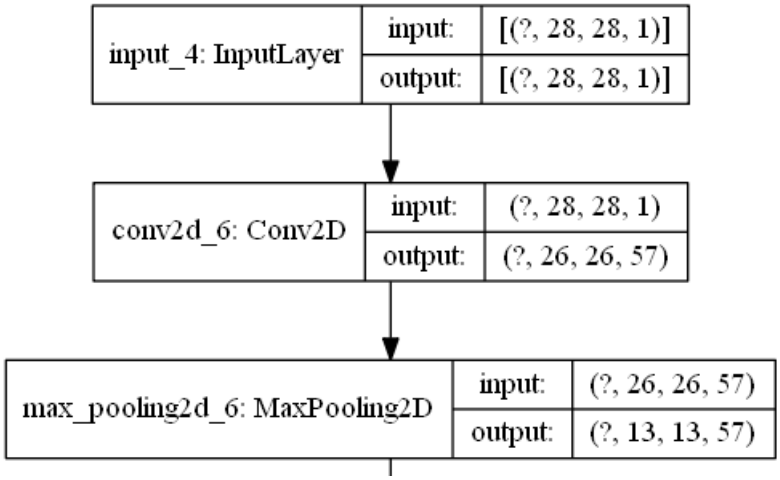
```
CNN_input = Input(shape = (28,28,1))
CNN = Conv2D(57,(3,3),activation='relu')(CNN_input)
CNN = MaxPool2D((2,2))(CNN)
CNN = Dropout(0.99)(CNN)
CNN = Conv2D(114,(3,3),activation='relu')(CNN)
CNN = MaxPool2D((2,2))(CNN)
CNN = Dropout(0.99)(CNN)
CNN = Flatten()(CNN)
CNN = Dense(10,activation='softmax')(CNN)
CNN_model = Model(CNN_input,CNN)
CNN_model.summary()
```

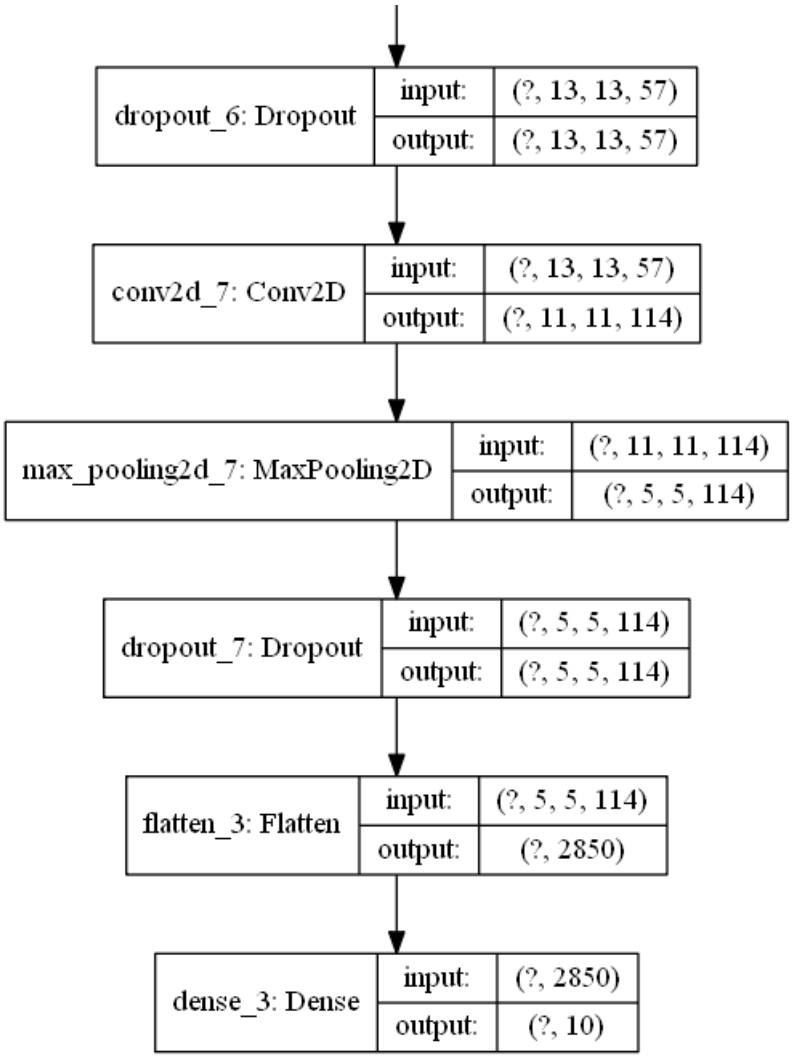
```
Model: "model_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         [(None, 28, 28, 1)]       0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 26, 26, 57)        570
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 13, 13, 57)        0
_________________________________________________________________
dropout_6 (Dropout)          (None, 13, 13, 57)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 11, 11, 114)       58596
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 5, 5, 114)         0
_________________________________________________________________
dropout_7 (Dropout)          (None, 5, 5, 114)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 2850)              0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                28510
=================================================================
Total params: 87,676
Trainable params: 87,676
Non-trainable params: 0
_________________________________________________________________
```

In [19]:

```
tf.keras.utils.plot_model(CNN_model, show_shapes=True)
```

Out[19]:





In [20]:

```
CNN_model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
CNN_model.fit(train_img,y_train, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 41s 22ms/step - loss: 2.4827 - accuracy: 0.1145
Epoch 2/5
1875/1875 [==============================] - 41s 22ms/step - loss: 2.3017 - accuracy: 0.1125
Epoch 3/5
1875/1875 [==============================] - 41s 22ms/step - loss: 2.3015 - accuracy: 0.1125
Epoch 4/5
1875/1875 [==============================] - 41s 22ms/step - loss: 2.3015 - accuracy: 0.1123
Epoch 5/5
1875/1875 [==============================] - 41s 22ms/step - loss: 2.3017 - accuracy: 0.1121
```

Out[20]:

```
<tensorflow.python.keras.callbacks.History at 0x29ffb38aa90>
```

In [21]:

```
CNN_model.evaluate(test_img,  y_test, verbose=2)
```

```
313/313 - 2s - loss: 2.3011 - accuracy: 0.1135
```

Out[21]:

```
[2.3011062145233154, 0.11349999904632568]
```

Case5 : Dropout = 0.95

In [23]:

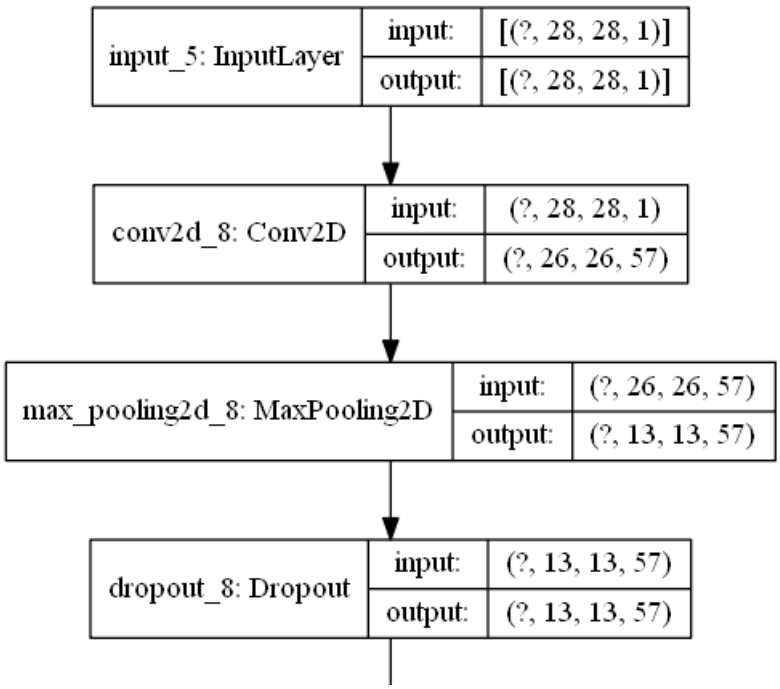
```
CNN_input = Input(shape = (28,28,1))
CNN = Conv2D(57,(3,3),activation='relu')(CNN_input)
CNN = MaxPool2D((2,2))(CNN)
CNN = Dropout(0.95)(CNN)
CNN = Conv2D(114,(3,3),activation='relu')(CNN)
CNN = MaxPool2D((2,2))(CNN)
CNN = Dropout(0.95)(CNN)
CNN = Flatten()(CNN)
CNN = Dense(10,activation='softmax')(CNN)
CNN_model = Model(CNN_input,CNN)
CNN_model.summary()
```

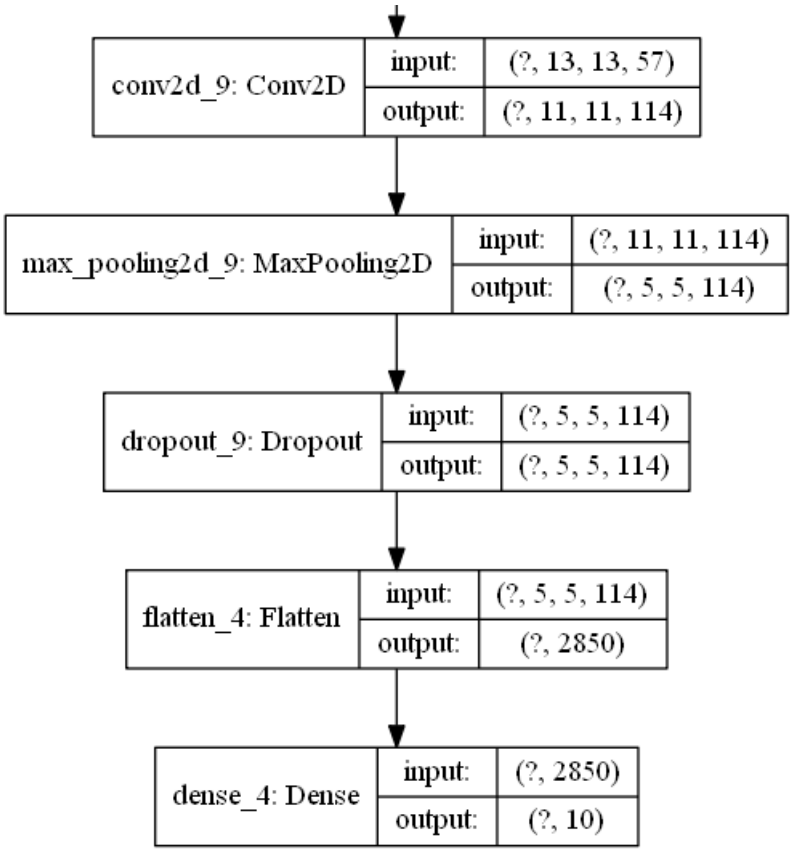
```
Model: "model_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_5 (InputLayer)         [(None, 28, 28, 1)]       0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 26, 26, 57)        570
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 13, 13, 57)        0
_________________________________________________________________
dropout_8 (Dropout)          (None, 13, 13, 57)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 11, 11, 114)       58596
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 5, 5, 114)         0
_________________________________________________________________
dropout_9 (Dropout)          (None, 5, 5, 114)         0
_________________________________________________________________
flatten_4 (Flatten)          (None, 2850)              0
_________________________________________________________________
dense_4 (Dense)              (None, 10)                28510
=================================================================
Total params: 87,676
Trainable params: 87,676
Non-trainable params: 0
_________________________________________________________________
```

In [24]:

```
tf.keras.utils.plot_model(CNN_model, show_shapes=True)
```

Out[24]:





In [25]:

```
CNN_model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
CNN_model.fit(train_img,y_train, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 41s 22ms/step - loss: 1.3608 - accuracy: 0.5375
Epoch 2/5
1875/1875 [==============================] - 40s 22ms/step - loss: 0.8108 - accuracy: 0.7407
Epoch 3/5
1875/1875 [==============================] - 41s 22ms/step - loss: 0.7192 - accuracy: 0.7693
Epoch 4/5
1875/1875 [==============================] - 41s 22ms/step - loss: 0.7021 - accuracy: 0.7794
Epoch 5/5
1875/1875 [==============================] - 41s 22ms/step - loss: 0.6919 - accuracy: 0.7808
```

Out[25]:

```
<tensorflow.python.keras.callbacks.History at 0x29ff9dc0e48>
```

In [28]:

```
CNN_model.evaluate(test_img,  y_test, verbose=2)
```

```
313/313 - 2s - loss: 0.4982 - accuracy: 0.9088
```

Out[28]:

```
[0.498157799243927, 0.9088000059127808]
```

Case6 : Dropout = 0.96

In [29]:

```
CNN_input = Input(shape = (28,28,1))
CNN = Conv2D(57,(3,3),activation='relu')(CNN_input)
CNN = MaxPool2D((2,2))(CNN)
```

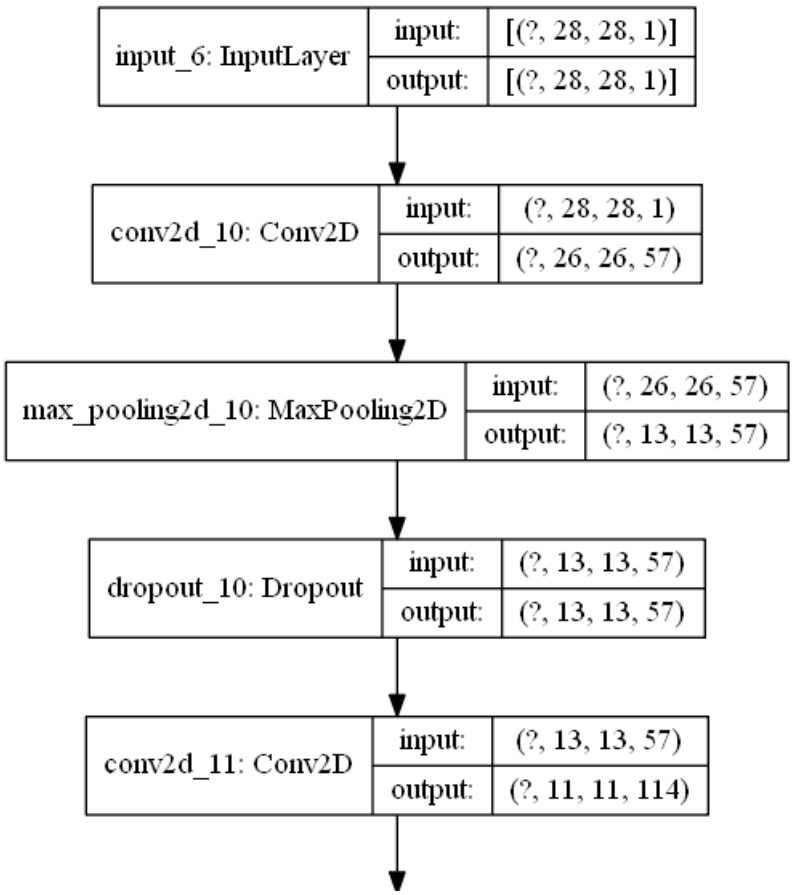
```
CNN = Dropout(0.96)(CNN)
CNN = Conv2D(114,(3,3),activation='relu')(CNN)
CNN = MaxPool2D((2,2))(CNN)
CNN = Dropout(0.96)(CNN)
CNN = Flatten()(CNN)
CNN = Dense(10,activation='softmax')(CNN)
CNN_model = Model(CNN_input,CNN)
CNN_model.summary()
```

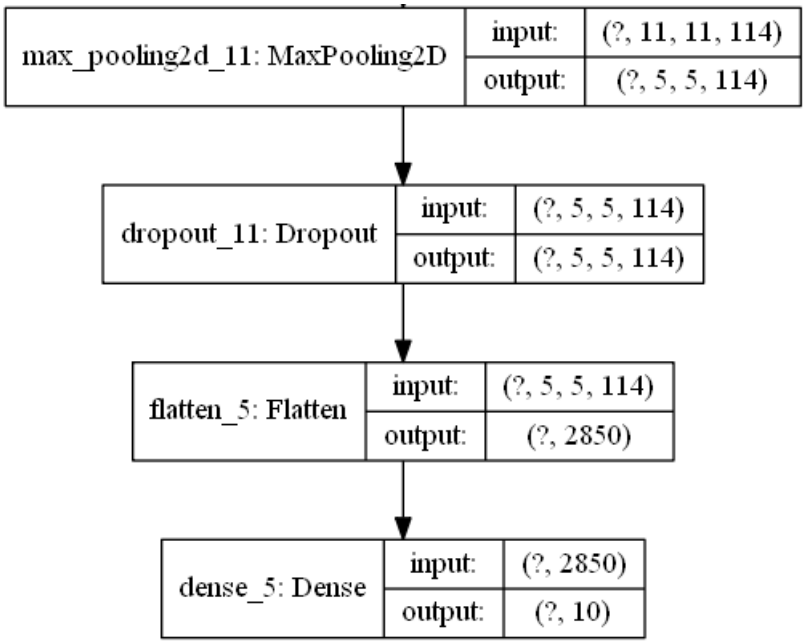
```
Model: "model_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_6 (InputLayer)         [(None, 28, 28, 1)]       0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 26, 26, 57)        570
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 13, 13, 57)        0
_________________________________________________________________
dropout_10 (Dropout)         (None, 13, 13, 57)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 11, 11, 114)       58596
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 5, 5, 114)         0
_________________________________________________________________
dropout_11 (Dropout)         (None, 5, 5, 114)         0
_________________________________________________________________
flatten_5 (Flatten)          (None, 2850)              0
_________________________________________________________________
dense_5 (Dense)              (None, 10)                28510
=================================================================
Total params: 87,676
Trainable params: 87,676
Non-trainable params: 0
_________________________________________________________________
```

In [30]:

```
tf.keras.utils.plot_model(CNN_model, show_shapes=True)
```

Out[30]:





In [31]:

```
CNN_model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
CNN_model.fit(train_img,y_train, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 40s 22ms/step - loss: 1.4473 - accuracy: 0.5248
Epoch 2/5
1875/1875 [==============================] - 41s 22ms/step - loss: 0.9847 - accuracy: 0.6971
Epoch 3/5
1875/1875 [==============================] - 40s 21ms/step - loss: 0.9109 - accuracy: 0.7158
Epoch 4/5
1875/1875 [==============================] - 40s 22ms/step - loss: 0.8927 - accuracy: 0.7209
Epoch 5/5
1875/1875 [==============================] - 40s 22ms/step - loss: 0.8781 - accuracy: 0.7231
```

Out[31]:

```
<tensorflow.python.keras.callbacks.History at 0x29fa567bf60>
```

In [35]:

```
CNN_model.evaluate(test_img,  y_test, verbose=2)
```

```
313/313 - 2s - loss: 1.4124 - accuracy: 0.4810
```

Out[35]:

```
[1.4123649597167969, 0.48100000619888306]
```

3. 결과

CNN모델 생성 및 학습 결과는 다음과 같다.

Case	Dropout 확률	시간	정확성
1	0.1	201.106	0.9909
2	0.4	208.111	0.9890
3	0.9	208.110	0.9676
4	0.99	205.11	0.1135

5	0.95	204.11	0.9088
6	0.96	201.109	0.4810

Dropout 확률을 달리하여 6가지 경우에 대한 CNN모델을 적용해보았다. 결론적으로는 Dropout가 높을 수록 최종 정확성이 높아진다. Case1 ~ Case3의 경우 모두 90%가 훨씬 넘는 정확성을 가지고 있다. 그 중에서도 Dropout의 확률이 0.1로 가장 낮은 Case1의 정확성이 약 99%로 제일 높았다.

정확도가 90%이하로 떨어지기 위해서는 Dropout이 어느 정도 되어야 하는지 확인하기 위해 Case4에서 Dropout을 0.99로 높였더니 약 11%로 매우 낮은 정확성을 얻게 되었다. 그래서 Case3과 Case4의 중간 Dropout인 0.95로 Case4에서 확인한 결과 약 91%의 정확성을 얻게 되었다. 그리고 Case 6에서 아주 조금 Dropout을 증가시키니 정확성이 Case 5의 절반이 됨을 확인할 수 있었다. 따라서 약 0.95의 이상의 값을 Dropout값으로 주어야 정확성이 최소 90% 이상이 나오게 된다.

의외로 시간은 Dropout, 정확성과 연관이 떨어졌다. 정확성과 반비례할 것이라는 예측과는 달랐는데 정말 연관이 없다기 보다는 예제로 학습한 데이터가 매우 작았기 때문일 것이라 추측한다.

마지막으로, 모든 Case에서 학습이 진행될수록 정확도가 높아지는 것을 확인할 수 있었다.