In [ ]:
```
import tensorflow.keras as keras
from tensorflow.keras import layers
from tensorflow.keras import models
from tensorflow.keras.datasets import mnist
import tensorflow.keras.utils as np_utils
from keras.optimizers import SGD
```

In [ ]:
```
(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train = X_train.reshape(X_train.shape[0], 28, 28, 1).astype('float32') / 255
X_test = X_test.reshape(X_test.shape[0], 28, 28, 1).astype('float32') / 255
Y_train = np_utils.to_categorical(y_train, 10)
Y_test = np_utils.to_categorical(y_test, 10)
```

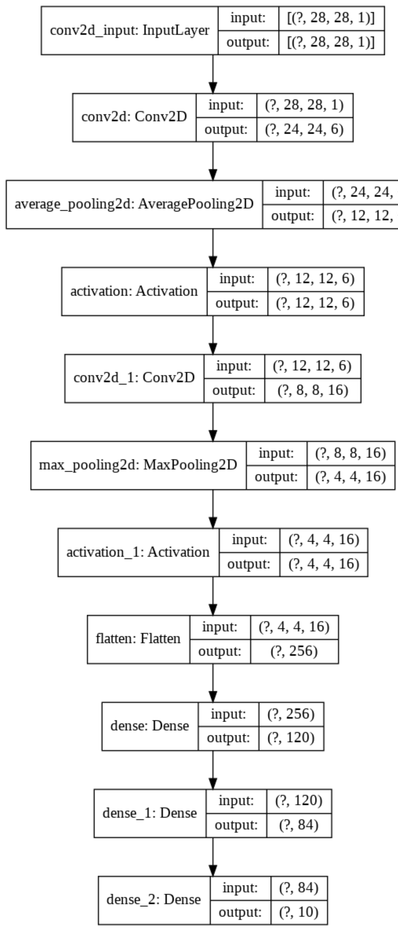
In [ ]:
```
model = models.Sequential()
model.add(layers.Conv2D(6, 5, padding='valid', input_shape = (28, 28, 1)))
model.add(layers.AvgPool2D(pool_size=(2, 2)))
model.add(layers.Activation("sigmoid"))
model.add(layers.Conv2D(16, 5, padding='valid'))
model.add(layers.MaxPool2D(pool_size=(2, 2)))
model.add(layers.Activation("sigmoid"))
model.add(layers.Flatten())
model.add(layers.Dense(120, activation='sigmoid'))
model.add(layers.Dense(84, activation='sigmoid'))
model.add(layers.Dense(10, activation='softmax'))
```

In [ ]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 24, 24, 6)         156
_________________________________________________________________
average_pooling2d (AveragePo (None, 12, 12, 6)         0
_________________________________________________________________
activation (Activation)      (None, 12, 12, 6)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 8, 8, 16)          2416
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 4, 4, 16)          0
_________________________________________________________________
activation_1 (Activation)    (None, 4, 4, 16)          0
_________________________________________________________________
flatten (Flatten)            (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 120)               30840
_________________________________________________________________
dense_1 (Dense)              (None, 84)                10164
_________________________________________________________________
dense_2 (Dense)              (None, 10)                850
=================================================================
Total params: 44,426
Trainable params: 44,426
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:
```
keras.utils.plot_model(model, "LeNet-5.png", rankdir="TB", show_shapes=True)
```

Out[ ]:



In [ ]:
```
l_rate = 1
sgd = SGD(lr=l_rate, momentum=0.8)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
model.fit(X_train, Y_train, batch_size=32, epochs=2, validation_data=(X_test, Y_test))
```

In [ ]:
```
l_rate = 1
sgd = SGD(lr=0.8*l_rate, momentum=0.8)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
model.fit(X_train, Y_train, batch_size=32, epochs=3, validation_data=(X_test, Y_test))
```

In [ ]:
```
l_rate = 1
sgd = SGD(lr=0.4*l_rate, momentum=0.8)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
model.fit(X_train, Y_train, batch_size=32, epochs=3, validation_data=(X_test, Y_test))
```

In [ ]:
```
l_rate = 1
sgd = SGD(lr=0.2*l_rate, momentum=0.8)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
model.fit(X_train, Y_train, batch_size=32, epochs=4, validation_data=(X_test, Y_test))
```

In [ ]:
```
l_rate = 1
sgd = SGD(lr=0.08*l_rate, momentum=0.8)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
model.fit(X_train, Y_train, batch_size=32, epochs=10, validation_data=(X_test, Y_test))
```

```
Epoch 1/10
1875/1875 [==============================] - 31s 16ms/step - loss: 2.3097 - accuracy: 0.1064 - val_loss: 2.3024 - val_
accuracy: 0.1135
Epoch 2/10
1875/1875 [==============================] - 32s 17ms/step - loss: 2.3033 - accuracy: 0.1092 - val_loss: 2.2987 - val_
accuracy: 0.1459
Epoch 3/10
1875/1875 [==============================] - 32s 17ms/step - loss: 0.8688 - accuracy: 0.6993 - val_loss: 0.1502 - val_
accuracy: 0.9532
Epoch 4/10
1875/1875 [==============================] - 33s 17ms/step - loss: 0.1229 - accuracy: 0.9630 - val_loss: 0.0853 - val_
accuracy: 0.9726
Epoch 5/10
1875/1875 [==============================] - 32s 17ms/step - loss: 0.0841 - accuracy: 0.9739 - val_loss: 0.0830 - val_
accuracy: 0.9756
Epoch 6/10
1875/1875 [==============================] - 33s 17ms/step - loss: 0.0682 - accuracy: 0.9787 - val_loss: 0.0602 - val_
accuracy: 0.9814
Epoch 7/10
1875/1875 [==============================] - 33s 18ms/step - loss: 0.0582 - accuracy: 0.9822 - val_loss: 0.0601 - val_
accuracy: 0.9812
Epoch 8/10
1875/1875 [==============================] - 32s 17ms/step - loss: 0.0515 - accuracy: 0.9840 - val_loss: 0.0442 - val_
accuracy: 0.9858
Epoch 9/10
1875/1875 [==============================] - 32s 17ms/step - loss: 0.0455 - accuracy: 0.9856 - val_loss: 0.0457 - val_
accuracy: 0.9852
Epoch 10/10
1875/1875 [==============================] - 33s 18ms/step - loss: 0.0415 - accuracy: 0.9869 - val_loss: 0.0468 - val_
accuracy: 0.9863
```

Out[ ]:
```
<tensorflow.python.keras.callbacks.History at 0x7fa752ab7a20>
```

In [ ]:
```
```