In [2]:
```
from __future__ import print_function
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
```

```
Using TensorFlow backend.
```

In [0]:
```
batch_size = 128
num_classes = 10
epochs = 12
```

In [0]:
```
# input image dimensions
img_rows, img_cols = 28, 28
```

In [5]:
```
# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
```

In [0]:
```
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)
```

In [7]:
```
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
```

In [0]:
```
# convert class vectors to binary class matrices

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

1. Model 1 - 2 Hidden Layers + Adam + Max Pooling + Kernel 3X3

In [9]:
```
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(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.Adadelta(), metrics=['accurac
y'])

model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1, validation_data=(x_test, y_test))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 154s 3ms/step - loss: 0.2555 - accuracy: 0.9209 - val_loss: 0.0573
- val_accuracy: 0.9812
Epoch 2/12
60000/60000 [==============================] - 153s 3ms/step - loss: 0.0859 - accuracy: 0.9747 - val_loss: 0.0384
- val_accuracy: 0.9871
Epoch 3/12
60000/60000 [==============================] - 153s 3ms/step - loss: 0.0649 - accuracy: 0.9808 - val_loss: 0.0337
- val_accuracy: 0.9887
Epoch 4/12
60000/60000 [==============================] - 153s 3ms/step - loss: 0.0547 - accuracy: 0.9840 - val_loss: 0.0349
- val_accuracy: 0.9888
Epoch 5/12
60000/60000 [==============================] - 153s 3ms/step - loss: 0.0458 - accuracy: 0.9861 - val_loss: 0.0305
- val_accuracy: 0.9890
Epoch 6/12
60000/60000 [==============================] - 152s 3ms/step - loss: 0.0421 - accuracy: 0.9871 - val_loss: 0.0278
- val_accuracy: 0.9903
Epoch 7/12
60000/60000 [==============================] - 152s 3ms/step - loss: 0.0378 - accuracy: 0.9883 - val_loss: 0.0333
- val_accuracy: 0.9884
Epoch 8/12
60000/60000 [==============================] - 154s 3ms/step - loss: 0.0337 - accuracy: 0.9894 - val_loss: 0.0292
- val_accuracy: 0.9906
Epoch 9/12
60000/60000 [==============================] - 154s 3ms/step - loss: 0.0313 - accuracy: 0.9903 - val_loss: 0.0270
- val_accuracy: 0.9909
Epoch 10/12
60000/60000 [==============================] - 152s 3ms/step - loss: 0.0283 - accuracy: 0.9908 - val_loss: 0.0245
- val_accuracy: 0.9923
Epoch 11/12
60000/60000 [==============================] - 154s 3ms/step - loss: 0.0268 - accuracy: 0.9918 - val_loss: 0.0275
- val_accuracy: 0.9916
Epoch 12/12
60000/60000 [==============================] - 154s 3ms/step - loss: 0.0254 - accuracy: 0.9923 - val_loss: 0.0284
- val_accuracy: 0.9922
```

Out[9]:
```
<keras.callbacks.callbacks.History at 0x7f83b355c898>
```

In [10]:
```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.028409935376906652
Test accuracy: 0.9922000169754028
```

In [0]:
```

```

1. Model 2 - 2 Hidden Layers + Adam + Max Pooling + Kernel 5X5

In [11]:
```
model_2 = Sequential()
model_2.add(Conv2D(32, kernel_size=(5, 5), activation='relu', input_shape=input_shape))
model_2.add(Conv2D(64, (3, 3), activation='relu'))
model_2.add(MaxPooling2D(pool_size=(2, 2)))
model_2.add(Dropout(0.25))
model_2.add(Flatten())
model_2.add(Dense(128, activation='relu'))
model_2.add(Dropout(0.5))
model_2.add(Dense(num_classes, activation='softmax'))

model_2.compile(loss=keras.losses.categorical_crossentropy, optimizer='adam', metrics=['accuracy'])

model_2.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1, validation_data=(x_test, y_test))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 135s 2ms/step - loss: 0.2336 - accuracy: 0.9287 - val_loss: 0.0549
- val_accuracy: 0.9834
Epoch 2/12
60000/60000 [==============================] - 134s 2ms/step - loss: 0.0802 - accuracy: 0.9767 - val_loss: 0.0320
- val_accuracy: 0.9891
Epoch 3/12
60000/60000 [==============================] - 136s 2ms/step - loss: 0.0612 - accuracy: 0.9819 - val_loss: 0.0275
- val_accuracy: 0.9905
Epoch 4/12
60000/60000 [==============================] - 136s 2ms/step - loss: 0.0492 - accuracy: 0.9848 - val_loss: 0.0276
- val_accuracy: 0.9914
Epoch 5/12
60000/60000 [==============================] - 135s 2ms/step - loss: 0.0407 - accuracy: 0.9879 - val_loss: 0.0261
- val_accuracy: 0.9910
Epoch 6/12
60000/60000 [==============================] - 135s 2ms/step - loss: 0.0355 - accuracy: 0.9891 - val_loss: 0.0282
- val_accuracy: 0.9911
Epoch 7/12
60000/60000 [==============================] - 135s 2ms/step - loss: 0.0333 - accuracy: 0.9899 - val_loss: 0.0281
- val_accuracy: 0.9922
Epoch 8/12
60000/60000 [==============================] - 137s 2ms/step - loss: 0.0301 - accuracy: 0.9905 - val_loss: 0.0244
- val_accuracy: 0.9923
Epoch 9/12
60000/60000 [==============================] - 135s 2ms/step - loss: 0.0255 - accuracy: 0.9915 - val_loss: 0.0251
- val_accuracy: 0.9926
Epoch 10/12
60000/60000 [==============================] - 136s 2ms/step - loss: 0.0258 - accuracy: 0.9920 - val_loss: 0.0216
- val_accuracy: 0.9933
Epoch 11/12
60000/60000 [==============================] - 136s 2ms/step - loss: 0.0227 - accuracy: 0.9930 - val_loss: 0.0245
- val_accuracy: 0.9921
Epoch 12/12
60000/60000 [==============================] - 136s 2ms/step - loss: 0.0217 - accuracy: 0.9931 - val_loss: 0.0244
- val_accuracy: 0.9927
```

Out[11]:
```
<keras.callbacks.callbacks.History at 0x7f83b267ee10>
```

In [12]:
```
score = model_2.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.024443474179803117
Test accuracy: 0.9926999807357788
```

In [0]:
```

```

1. Model 2 - 2 Hidden Layers + Adam + Max Pooling + Kernel 7X7

In [13]:
```
model_3 = Sequential()
model_3.add(Conv2D(32, kernel_size=(7, 7), activation='relu', input_shape=input_shape))
model_3.add(Conv2D(64, (3, 3), activation='relu'))
model_3.add(MaxPooling2D(pool_size=(2, 2)))
model_3.add(Dropout(0.25))
model_3.add(Flatten())
model_3.add(Dense(128, activation='relu'))
model_3.add(Dropout(0.5))
model_3.add(Dense(num_classes, activation='softmax'))

model_3.compile(loss=keras.losses.categorical_crossentropy, optimizer='adam', metrics=['accuracy'])

model_3.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1, validation_data=(x_test, y_test))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 121s 2ms/step - loss: 0.2402 - accuracy: 0.9282 - val_loss: 0.0476
- val_accuracy: 0.9846
Epoch 2/12
60000/60000 [==============================] - 122s 2ms/step - loss: 0.0811 - accuracy: 0.9757 - val_loss: 0.0355
- val_accuracy: 0.9881
Epoch 3/12
60000/60000 [==============================] - 121s 2ms/step - loss: 0.0627 - accuracy: 0.9815 - val_loss: 0.0274
- val_accuracy: 0.9901
Epoch 4/12
60000/60000 [==============================] - 121s 2ms/step - loss: 0.0489 - accuracy: 0.9857 - val_loss: 0.0287
- val_accuracy: 0.9898
Epoch 5/12
60000/60000 [==============================] - 120s 2ms/step - loss: 0.0426 - accuracy: 0.9869 - val_loss: 0.0235
- val_accuracy: 0.9914
Epoch 6/12
60000/60000 [==============================] - 120s 2ms/step - loss: 0.0369 - accuracy: 0.9887 - val_loss: 0.0238
- val_accuracy: 0.9915
Epoch 7/12
60000/60000 [==============================] - 119s 2ms/step - loss: 0.0318 - accuracy: 0.9902 - val_loss: 0.0250
- val_accuracy: 0.9926
Epoch 8/12
60000/60000 [==============================] - 119s 2ms/step - loss: 0.0310 - accuracy: 0.9906 - val_loss: 0.0209
- val_accuracy: 0.9931
Epoch 9/12
60000/60000 [==============================] - 118s 2ms/step - loss: 0.0259 - accuracy: 0.9918 - val_loss: 0.0224
- val_accuracy: 0.9926
Epoch 10/12
60000/60000 [==============================] - 120s 2ms/step - loss: 0.0247 - accuracy: 0.9923 - val_loss: 0.0243
- val_accuracy: 0.9926
Epoch 11/12
60000/60000 [==============================] - 120s 2ms/step - loss: 0.0220 - accuracy: 0.9929 - val_loss: 0.0228
- val_accuracy: 0.9924
Epoch 12/12
60000/60000 [==============================] - 120s 2ms/step - loss: 0.0227 - accuracy: 0.9925 - val_loss: 0.0200
- val_accuracy: 0.9944
```

Out[13]:
```
<keras.callbacks.callbacks.History at 0x7f83b2498898>
```

In [14]:
```
score = model_3.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.019960603120292627
Test accuracy: 0.9944000244140625
```

In [0]:
```

```

1. Model 4 - 3 Hidden Layers + Adadelta + Max Pooling + Kernel 3X3

In [15]:
```
model_4 = Sequential()
model_4.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
model_4.add(Conv2D(64, (3, 3), activation='relu'))
model_4.add(MaxPooling2D(pool_size=(2, 2)))
model_4.add(Conv2D(128, (3, 3), activation='relu'))
model_4.add(MaxPooling2D(pool_size=(2, 2)))
model_4.add(Dropout(0.25))
model_4.add(Flatten())
model_4.add(Dense(256, activation='relu'))
model_4.add(Dropout(0.5))
model_4.add(Dense(num_classes, activation='softmax'))

model_4.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.Adadelta(), metrics=['accura
cy'])

model_4.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1, validation_data=(x_test, y_test))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 219s 4ms/step - loss: 0.2324 - accuracy: 0.9265 - val_loss: 0.0477
- val_accuracy: 0.9834
Epoch 2/12
60000/60000 [==============================] - 217s 4ms/step - loss: 0.0662 - accuracy: 0.9797 - val_loss: 0.0309
- val_accuracy: 0.9900
Epoch 3/12
60000/60000 [==============================] - 218s 4ms/step - loss: 0.0476 - accuracy: 0.9851 - val_loss: 0.0234
- val_accuracy: 0.9918
Epoch 4/12
60000/60000 [==============================] - 219s 4ms/step - loss: 0.0386 - accuracy: 0.9884 - val_loss: 0.0261
- val_accuracy: 0.9916
Epoch 5/12
60000/60000 [==============================] - 218s 4ms/step - loss: 0.0327 - accuracy: 0.9899 - val_loss: 0.0215
- val_accuracy: 0.9924
Epoch 6/12
60000/60000 [==============================] - 219s 4ms/step - loss: 0.0304 - accuracy: 0.9909 - val_loss: 0.0190
- val_accuracy: 0.9930
Epoch 7/12
60000/60000 [==============================] - 218s 4ms/step - loss: 0.0245 - accuracy: 0.9928 - val_loss: 0.0198
- val_accuracy: 0.9936
Epoch 8/12
60000/60000 [==============================] - 219s 4ms/step - loss: 0.0226 - accuracy: 0.9929 - val_loss: 0.0174
- val_accuracy: 0.9942
Epoch 9/12
60000/60000 [==============================] - 217s 4ms/step - loss: 0.0183 - accuracy: 0.9942 - val_loss: 0.0154
- val_accuracy: 0.9950
Epoch 10/12
60000/60000 [==============================] - 218s 4ms/step - loss: 0.0189 - accuracy: 0.9936 - val_loss: 0.0177
- val_accuracy: 0.9947
Epoch 11/12
60000/60000 [==============================] - 219s 4ms/step - loss: 0.0157 - accuracy: 0.9952 - val_loss: 0.0195
- val_accuracy: 0.9932
Epoch 12/12
60000/60000 [==============================] - 219s 4ms/step - loss: 0.0160 - accuracy: 0.9951 - val_loss: 0.0172
- val_accuracy: 0.9947
```

Out[15]:
```
<keras.callbacks.callbacks.History at 0x7f83b2262d30>
```

In [17]:
```
score = model_4.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.01719610899976833
Test accuracy: 0.994700014591217
```

In [0]:
```

```

1. Model 5 - 3 Hidden Layers + Adam + Max Pooling + Kernel 5X5

In [18]:
```
model_5 = Sequential()
model_5.add(Conv2D(32, kernel_size=(5, 5), activation='relu', input_shape=input_shape))
model_5.add(Conv2D(64, (5, 5), activation='relu'))
model_5.add(MaxPooling2D(pool_size=(2, 2)))
model_5.add(Conv2D(128, (5, 5), activation='relu'))
model_5.add(MaxPooling2D(pool_size=(2, 2)))
model_5.add(Dropout(0.25))
model_5.add(Flatten())
model_5.add(Dense(256, activation='relu'))
model_5.add(Dropout(0.5))
model_5.add(Dense(num_classes, activation='softmax'))

model_5.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.Adadelta(), metrics=['accura
cy'])

model_5.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1, validation_data=(x_test, y_test))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 304s 5ms/step - loss: 0.2448 - accuracy: 0.9212 - val_loss: 0.0383
- val_accuracy: 0.9882
Epoch 2/12
60000/60000 [==============================] - 307s 5ms/step - loss: 0.0617 - accuracy: 0.9816 - val_loss: 0.0260
- val_accuracy: 0.9912
Epoch 3/12
60000/60000 [==============================] - 302s 5ms/step - loss: 0.0434 - accuracy: 0.9873 - val_loss: 0.0418
- val_accuracy: 0.9863
Epoch 4/12
60000/60000 [==============================] - 307s 5ms/step - loss: 0.0337 - accuracy: 0.9899 - val_loss: 0.0212
- val_accuracy: 0.9936
Epoch 5/12
60000/60000 [==============================] - 303s 5ms/step - loss: 0.0272 - accuracy: 0.9917 - val_loss: 0.0188
- val_accuracy: 0.9936
Epoch 6/12
60000/60000 [==============================] - 298s 5ms/step - loss: 0.0244 - accuracy: 0.9926 - val_loss: 0.0187
- val_accuracy: 0.9942
Epoch 7/12
60000/60000 [==============================] - 303s 5ms/step - loss: 0.0211 - accuracy: 0.9936 - val_loss: 0.0171
- val_accuracy: 0.9948
Epoch 8/12
60000/60000 [==============================] - 305s 5ms/step - loss: 0.0177 - accuracy: 0.9944 - val_loss: 0.0252
- val_accuracy: 0.9923
Epoch 9/12
60000/60000 [==============================] - 303s 5ms/step - loss: 0.0161 - accuracy: 0.9951 - val_loss: 0.0170
- val_accuracy: 0.9947
Epoch 10/12
60000/60000 [==============================] - 308s 5ms/step - loss: 0.0133 - accuracy: 0.9959 - val_loss: 0.0192
- val_accuracy: 0.9940
Epoch 11/12
60000/60000 [==============================] - 306s 5ms/step - loss: 0.0127 - accuracy: 0.9959 - val_loss: 0.0172
- val_accuracy: 0.9947
Epoch 12/12
60000/60000 [==============================] - 305s 5ms/step - loss: 0.0113 - accuracy: 0.9965 - val_loss: 0.0171
- val_accuracy: 0.9950
```

Out[18]:
```
<keras.callbacks.callbacks.History at 0x7f83b1f37940>
```

In [21]:
```
score = model_5.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.01709874028202305
Test accuracy: 0.9950000047683716
```

In [0]:
```

```

1. Model 6 - 3 Hidden Layers + Adam + Max Pooling + Kernel 7X7

In [23]:
```
model_6 = Sequential()
model_6.add(Conv2D(32, kernel_size=(7, 7), activation='relu', input_shape=input_shape))
model_6.add(Conv2D(64, (7, 7), activation='relu'))
model_6.add(MaxPooling2D(pool_size=(2, 2)))
model_6.add(Conv2D(128, (7, 7), activation='relu'))
model_6.add(MaxPooling2D(pool_size=(2, 2)))
model_6.add(Dropout(0.25))
model_6.add(Flatten())
model_6.add(Dense(256, activation='relu'))
model_6.add(Dropout(0.5))
model_6.add(Dense(num_classes, activation='softmax'))

model_6.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.Adadelta(), metrics=['accura
cy'])

model_6.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1, validation_data=(x_test, y_test))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 334s 6ms/step - loss: 0.2568 - accuracy: 0.9190 - val_loss: 0.0446
- val_accuracy: 0.9874
Epoch 2/12
60000/60000 [==============================] - 333s 6ms/step - loss: 0.0610 - accuracy: 0.9827 - val_loss: 0.0322
- val_accuracy: 0.9904
Epoch 3/12
60000/60000 [==============================] - 337s 6ms/step - loss: 0.0417 - accuracy: 0.9876 - val_loss: 0.0333
- val_accuracy: 0.9898
Epoch 4/12
60000/60000 [==============================] - 338s 6ms/step - loss: 0.0328 - accuracy: 0.9905 - val_loss: 0.0232
- val_accuracy: 0.9930
Epoch 5/12
60000/60000 [==============================] - 337s 6ms/step - loss: 0.0264 - accuracy: 0.9923 - val_loss: 0.0244
- val_accuracy: 0.9922
Epoch 6/12
60000/60000 [==============================] - 337s 6ms/step - loss: 0.0204 - accuracy: 0.9941 - val_loss: 0.0220
- val_accuracy: 0.9929
Epoch 7/12
60000/60000 [==============================] - 336s 6ms/step - loss: 0.0175 - accuracy: 0.9948 - val_loss: 0.0201
- val_accuracy: 0.9946
Epoch 8/12
60000/60000 [==============================] - 337s 6ms/step - loss: 0.0158 - accuracy: 0.9953 - val_loss: 0.0229
- val_accuracy: 0.9942
Epoch 9/12
60000/60000 [==============================] - 340s 6ms/step - loss: 0.0126 - accuracy: 0.9965 - val_loss: 0.0259
- val_accuracy: 0.9926
Epoch 10/12
60000/60000 [==============================] - 343s 6ms/step - loss: 0.0116 - accuracy: 0.9966 - val_loss: 0.0253
- val_accuracy: 0.9931
Epoch 11/12
60000/60000 [==============================] - 349s 6ms/step - loss: 0.0093 - accuracy: 0.9972 - val_loss: 0.0282
- val_accuracy: 0.9937
Epoch 12/12
60000/60000 [==============================] - 350s 6ms/step - loss: 0.0093 - accuracy: 0.9974 - val_loss: 0.0271
- val_accuracy: 0.9934
```

Out[23]:
```
<keras.callbacks.callbacks.History at 0x7f83b196d7b8>
```

In [24]:
```
score = model_6.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.027129932529083498
Test accuracy: 0.993399977684021
```

In [0]:
```

```

1. Model 7 - 5 Hidden Layers + Adadelta + Max Pooling + Kernel 3X3

In [48]:
```
model_7 = Sequential()
model_7.add(Conv2D(16, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
model_7.add(Conv2D(32, (3, 3), activation='relu'))
model_7.add(MaxPooling2D(pool_size=(2, 2)))
model_7.add(Conv2D(64, (3, 3), activation='relu'))
model_7.add(MaxPooling2D(pool_size=(2, 2)))
model_7.add(Conv2D(32, (3,3 ), activation='relu'))
model_7.add(Dropout(0.2))
model_7.add(Conv2D(16, (3, 3), activation='relu'))
model_7.add(Dropout(0.2))
model_7.add(Flatten())
model_7.add(Dense(256, activation='relu'))
model_7.add(Dropout(0.5))
model_7.add(Dense(num_classes, activation='softmax'))

model_7.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.Adadelta(), metrics=['accura
cy'])

model_7.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1, validation_data=(x_test, y_test))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 76s 1ms/step - loss: 0.5951 - accuracy: 0.8008 - val_loss: 0.0858 -
val_accuracy: 0.9755
Epoch 2/12
60000/60000 [==============================] - 76s 1ms/step - loss: 0.1648 - accuracy: 0.9500 - val_loss: 0.0508 -
val_accuracy: 0.9850
Epoch 3/12
60000/60000 [==============================] - 79s 1ms/step - loss: 0.1257 - accuracy: 0.9633 - val_loss: 0.0417 -
val_accuracy: 0.9868
Epoch 4/12
60000/60000 [==============================] - 78s 1ms/step - loss: 0.1020 - accuracy: 0.9704 - val_loss: 0.0391 -
val_accuracy: 0.9868
Epoch 5/12
60000/60000 [==============================] - 79s 1ms/step - loss: 0.0887 - accuracy: 0.9735 - val_loss: 0.0435 -
val_accuracy: 0.9879
Epoch 6/12
60000/60000 [==============================] - 76s 1ms/step - loss: 0.0767 - accuracy: 0.9773 - val_loss: 0.0359 -
val_accuracy: 0.9892
Epoch 7/12
60000/60000 [==============================] - 79s 1ms/step - loss: 0.0685 - accuracy: 0.9804 - val_loss: 0.0265 -
val_accuracy: 0.9919
Epoch 8/12
60000/60000 [==============================] - 78s 1ms/step - loss: 0.0642 - accuracy: 0.9811 - val_loss: 0.0287 -
val_accuracy: 0.9919
Epoch 9/12
60000/60000 [==============================] - 76s 1ms/step - loss: 0.0590 - accuracy: 0.9834 - val_loss: 0.0273 -
val_accuracy: 0.9920
Epoch 10/12
60000/60000 [==============================] - 80s 1ms/step - loss: 0.0558 - accuracy: 0.9841 - val_loss: 0.0244 -
val_accuracy: 0.9936
Epoch 11/12
60000/60000 [==============================] - 76s 1ms/step - loss: 0.0508 - accuracy: 0.9858 - val_loss: 0.0238 -
val_accuracy: 0.9930
Epoch 12/12
60000/60000 [==============================] - 75s 1ms/step - loss: 0.0474 - accuracy: 0.9864 - val_loss: 0.0271 -
val_accuracy: 0.9930
```

Out[48]:
```
<keras.callbacks.callbacks.History at 0x7f83b05085f8>
```

In [49]:
```
score = model_7.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.027066458926092492
Test accuracy: 0.9929999709129333
```

In [0]:
```

```

1. Model 8 - 5 Hidden Layers + Adam + Max Pooling + Kernel 5X5

In [9]:
```
model_8 = Sequential()
model_8.add(Conv2D(16, kernel_size=(5, 5), activation='relu', input_shape=input_shape))
model_8.add(Conv2D(32, (5, 5), activation='relu'))
model_8.add(Dropout(0.25))
#model_8.add(MaxPooling2D(pool_size=(2, 2)))
model_8.add(Conv2D(64, (5, 5), activation='relu'))
model_8.add(Dropout(0.25))
#model_8.add(MaxPooling2D(pool_size=(2, 2)))
model_8.add(Conv2D(64, (5, 5), activation='relu'))
#model_8.add(MaxPooling2D(pool_size=(2, 2)))
model_8.add(Conv2D(32, (5, 5), activation='relu'))
#model_8.add(MaxPooling2D(pool_size=(2, 2)))
model_8.add(Dropout(0.25))
model_8.add(Flatten())
model_8.add(Dense(256, activation='relu'))
model_8.add(Dropout(0.5))
model_8.add(Dense(num_classes, activation='softmax'))

model_8.compile(loss=keras.losses.categorical_crossentropy, optimizer='adam', metrics=['accuracy'])

model_8.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1, validation_data=(x_test, y_test))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 421s 7ms/step - loss: 0.2515 - accuracy: 0.9206 - val_loss: 0.0494
- val_accuracy: 0.9864
Epoch 2/12
60000/60000 [==============================] - 421s 7ms/step - loss: 0.0700 - accuracy: 0.9794 - val_loss: 0.0337
- val_accuracy: 0.9896
Epoch 3/12
60000/60000 [==============================] - 421s 7ms/step - loss: 0.0526 - accuracy: 0.9846 - val_loss: 0.0287
- val_accuracy: 0.9913
Epoch 4/12
60000/60000 [==============================] - 421s 7ms/step - loss: 0.0436 - accuracy: 0.9868 - val_loss: 0.0314
- val_accuracy: 0.9910
Epoch 5/12
60000/60000 [==============================] - 420s 7ms/step - loss: 0.0399 - accuracy: 0.9885 - val_loss: 0.0245
- val_accuracy: 0.9920
Epoch 6/12
60000/60000 [==============================] - 421s 7ms/step - loss: 0.0353 - accuracy: 0.9896 - val_loss: 0.0292
- val_accuracy: 0.9912
Epoch 7/12
60000/60000 [==============================] - 421s 7ms/step - loss: 0.0302 - accuracy: 0.9912 - val_loss: 0.0344
- val_accuracy: 0.9884
Epoch 8/12
60000/60000 [==============================] - 421s 7ms/step - loss: 0.0289 - accuracy: 0.9916 - val_loss: 0.0249
- val_accuracy: 0.9925
Epoch 9/12
60000/60000 [==============================] - 421s 7ms/step - loss: 0.0286 - accuracy: 0.9917 - val_loss: 0.0332
- val_accuracy: 0.9907
Epoch 10/12
60000/60000 [==============================] - 422s 7ms/step - loss: 0.0260 - accuracy: 0.9918 - val_loss: 0.0334
- val_accuracy: 0.9902
Epoch 11/12
60000/60000 [==============================] - 421s 7ms/step - loss: 0.0251 - accuracy: 0.9926 - val_loss: 0.0211
- val_accuracy: 0.9941
Epoch 12/12
60000/60000 [==============================] - 421s 7ms/step - loss: 0.0207 - accuracy: 0.9938 - val_loss: 0.0275
- val_accuracy: 0.9925
```

Out[9]:
```
<keras.callbacks.callbacks.History at 0x7fed1a3a5908>
```

In [10]:
```
score = model_8.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.027493627056613512
Test accuracy: 0.9925000071525574
```

In [0]:
```

```

1. Model 9 - 5 Hidden Layers + Adam + Max Pooling + Kernel 7X7 + padding

In [18]:
```
model_9 = Sequential()
model_9.add(Conv2D(16, kernel_size=(7, 7), activation='relu', input_shape=input_shape))
model_9.add(Conv2D(24, (7, 7), activation='relu',padding='same'))
model_9.add(MaxPooling2D(pool_size=(2, 2)))
model_9.add(Dropout(0.25))
model_9.add(Conv2D(32, (7, 7), activation='relu',padding='same'))
model_9.add(MaxPooling2D(pool_size=(2, 2)))
model_9.add(Conv2D(24, (7, 7), activation='relu',padding='same'))
model_9.add(MaxPooling2D(pool_size=(2, 2)))
model_9.add(Conv2D(16, (7, 7), activation='relu',padding='same'))
model_9.add(MaxPooling2D(pool_size=(2, 2)))
model_9.add(Dropout(0.25))
model_9.add(Flatten())
model_9.add(Dense(256, activation='relu'))
model_9.add(Dropout(0.5))
model_9.add(Dense(num_classes, activation='softmax'))

model_9.compile(loss=keras.losses.categorical_crossentropy, optimizer='adam', metrics=['accuracy'])

model_9.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1, validation_data=(x_test, y_test))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 242s 4ms/step - loss: 0.5484 - accuracy: 0.8182 - val_loss: 0.0831
- val_accuracy: 0.9756
Epoch 2/12
60000/60000 [==============================] - 242s 4ms/step - loss: 0.1629 - accuracy: 0.9523 - val_loss: 0.0460
- val_accuracy: 0.9859
Epoch 3/12
60000/60000 [==============================] - 241s 4ms/step - loss: 0.1184 - accuracy: 0.9648 - val_loss: 0.0533
- val_accuracy: 0.9861
Epoch 4/12
60000/60000 [==============================] - 241s 4ms/step - loss: 0.0908 - accuracy: 0.9735 - val_loss: 0.0354
- val_accuracy: 0.9900
Epoch 5/12
60000/60000 [==============================] - 243s 4ms/step - loss: 0.0794 - accuracy: 0.9770 - val_loss: 0.0425
- val_accuracy: 0.9882
Epoch 6/12
60000/60000 [==============================] - 241s 4ms/step - loss: 0.0719 - accuracy: 0.9801 - val_loss: 0.0273
- val_accuracy: 0.9931
Epoch 7/12
60000/60000 [==============================] - 240s 4ms/step - loss: 0.0618 - accuracy: 0.9827 - val_loss: 0.0300
- val_accuracy: 0.9915
Epoch 8/12
60000/60000 [==============================] - 241s 4ms/step - loss: 0.0565 - accuracy: 0.9842 - val_loss: 0.0306
- val_accuracy: 0.9919
Epoch 9/12
60000/60000 [==============================] - 241s 4ms/step - loss: 0.0555 - accuracy: 0.9848 - val_loss: 0.0257
- val_accuracy: 0.9924
Epoch 10/12
60000/60000 [==============================] - 241s 4ms/step - loss: 0.0485 - accuracy: 0.9863 - val_loss: 0.0263
- val_accuracy: 0.9934
Epoch 11/12
60000/60000 [==============================] - 241s 4ms/step - loss: 0.0459 - accuracy: 0.9871 - val_loss: 0.0265
- val_accuracy: 0.9926
Epoch 12/12
60000/60000 [==============================] - 241s 4ms/step - loss: 0.0408 - accuracy: 0.9888 - val_loss: 0.0254
- val_accuracy: 0.9929
```

Out[18]:
```
<keras.callbacks.callbacks.History at 0x7f762b2f4198>
```

In [19]:
```
score = model_9.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.025357187755659833
Test accuracy: 0.992900013923645
```

In [22]:
```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Hidden Layers","Parameters","Accuracy"]
x.add_row([2,'CNN + ReLU +  Adadelta + Max Pooling + Kernel 3X3 + Dropout ', 0.992])
x.add_row([2,'CNN + ReLU +  Adam + Max Pooling + Kernel 5X5 + Dropout ', 0.992])
x.add_row([2,'CNN + ReLU +  Adam + Max Pooling + Kernel 7X7 + Dropout', 0.994])
x.add_row([3,'CNN + ReLU +  Adadelta + Max Pooling + Kernel 3X3 + Dropout ', 0.994])
x.add_row([3,'CNN + ReLU +  Adam + Max Pooling + Kernel 5X5 + Dropout', 0.995])
x.add_row([3,'CNN + ReLU +  Adam + Max Pooling + Kernel 7X7 + Dropout', 0.993])
x.add_row([5,'CNN + ReLU +  Adadelta + Max Pooling + Kernel 3X3 + Dropout', 0.992])
x.add_row([5,'CNN + ReLU +  Adam + Max Pooling + Kernel 5X5 + Dropout', 0.992])
x.add_row([5,'CNN + ReLU +  Adam + Max Pooling + Kernel 7X7 + Dropout + padding', 0.992])
print(x)
```

```
+---------------+-----------------------------------------------------------------+----------+
| Hidden Layers |                           Parameters                            | Accuracy |
+---------------+-----------------------------------------------------------------+----------+
|       2       |    CNN + ReLU +  Adadelta + Max Pooling + Kernel 3X3 + Dropout    |  0.992   |
|       2       |      CNN + ReLU +  Adam + Max Pooling + Kernel 5X5 + Dropout      |  0.992   |
|       2       |      CNN + ReLU +  Adam + Max Pooling + Kernel 7X7 + Dropout      |  0.994   |
|       3       |    CNN + ReLU +  Adadelta + Max Pooling + Kernel 3X3 + Dropout    |  0.994   |
|       3       |      CNN + ReLU +  Adam + Max Pooling + Kernel 5X5 + Dropout      |  0.995   |
|       3       |      CNN + ReLU +  Adam + Max Pooling + Kernel 7X7 + Dropout      |  0.993   |
|       5       |    CNN + ReLU +  Adadelta + Max Pooling + Kernel 3X3 + Dropout    |  0.992   |
|       5       |      CNN + ReLU +  Adam + Max Pooling + Kernel 5X5 + Dropout      |  0.992   |
|       5       | CNN + ReLU +  Adam + Max Pooling + Kernel 7X7 + Dropout + padding |  0.992   |
+---------------+-----------------------------------------------------------------+----------+
```

In [0]: