CNN MODEL FOR DIGIT RECOGNITION

- trained with 10 000 samples images from the MNIST dataset

In [94]:

```
from IPython.display import Image
from datetime import datetime
import numpy as np
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.callbacks import TensorBoard
```

In [95]:

```
#loading features and labels
features = np.load("C:/Users/AKUMA/Desktop/DESKTOP/UNITY/Python Scripts/SAVED NUMPY
ARRAYS/MNIST_features_v1.npy")
label = np.load("C:/Users/AKUMA/Desktop/DESKTOP/UNITY/Python Scripts/SAVED NUMPY
ARRAYS/MNIST_label_v1.npy")

#model init
features = features/255.0
n_classes = len(set(label))
layer_sizes = [32,64,128]
dense_layer_options = [0,1,2]
conv_layer_options = [1,2,3]
kernel_options = [(2,2),(3,3),(4,4)]


version_index = 0
NAME_LIST_DEBUG = []
print(label.shape)
print(features.shape)
```

```
(10000,)
(10000, 28, 28, 1)
```

DYNAMIC MODEL CONFIGURATION - QUICK DEBUGGING

In [96]:

```
# 3 parameters with 3 possibility to adjust, this, is giving us a 3x3x3 for loop
now = datetime.now()
timestamp = now.strftime("%d-%m-%Y_%H-%M-%S")


for dense_layer in dense_layer_options:
    for n_node in layer_sizes:
        for conv_layer in conv_layer_options:
            name = "mnist-digit-recog-cnn_{}-conv-{}-nodes-{}-dense_{}-timestamp".format(conv_layer
, n_node, dense_layer, timestamp)
            NAME_LIST_DEBUG.append(name)



print("number of models to train: ",len(NAME_LIST_DEBUG))
print("first name of the list: ", NAME_LIST_DEBUG[0])
```

```
number of models to train:  27
first name of the list:  mnist-digit-recog-cnn_1-conv-32-nodes-0-dense_16-04-2020_23-37-19-
timestamp
```

DYNAMIC CONFIGURATION RESEARCH - TRAINING

In [97]:

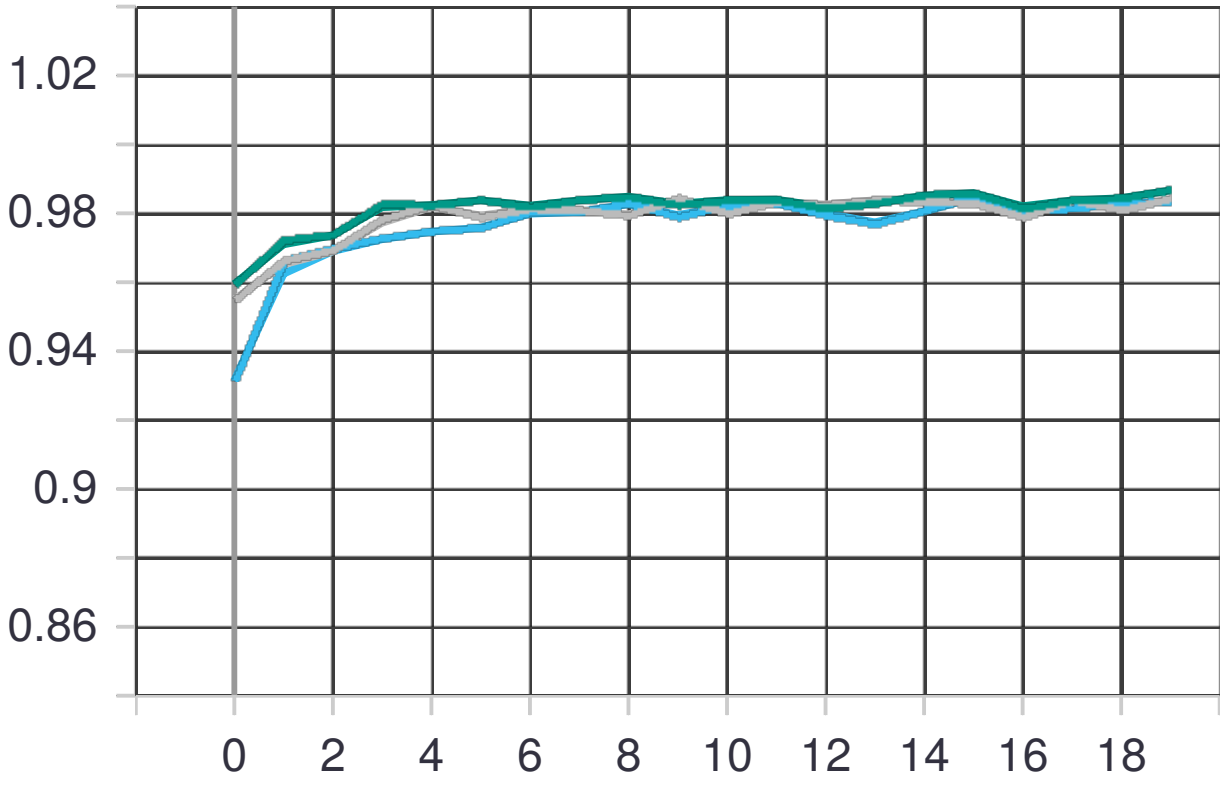
```
for dense_layer in dense_layer_options:
    for n_node in layer_sizes:
        for conv_layer in conv_layer_options:


            #INITIALIZATION
            #setting up the name of the model with the current date (dd/mm/YY H:M:S)
            now = datetime.now()
            timestamp = now.strftime("%d-%m-%Y_%H-%M")
            name = "mnist_digit_recog_cnn_{}-conv-{}-nodes-{}-dense-{}_Ver{}".format(conv_layer, n_
node, dense_layer, timestamp, str(version_index))
            version_index += 1

            #defining path of the future saved model
            model_path ="C:/Users/AKUMA/Desktop/DESKTOP/UNITY/Python Scripts/saved model/"
            model_name = ("{}{}.h5".format(model_path, name))

            #model initilization
            tensorboard = TensorBoard(log_dir="C:\\logs\\train\\{}".format(name))
            model = Sequential()

            #debug
            print("name", model_name)




            #1st Convolutional layer - this is the input layer
            model.add(Conv2D(n_node, (3,3), input_shape = features.shape[1:])) # (3,3) here is the
kernel size(or feature detector)
            model.add(Activation("relu"))
            model.add(MaxPooling2D(pool_size = (2,2)))



            #iterates through our different convolutional layer configurations
            for layer in range(conv_layer - 1): # -1 cuz we already created one above
                model.add(Conv2D(n_node, (3,3)))        # we put 'n_node' here, but it might be the
number of features to detect
                model.add(Activation("relu"))
                model.add(MaxPooling2D(pool_size = (2,2)))



            #we flatten the image before feeding it to the fully-connected Neural Networks
            model.add(Flatten())



             #fully connected  layer
            for layer in range(dense_layer):
                model.add(Dense(n_node))
                model.add(Activation("relu"))
                model.add(Dropout(0.3))


            #final output layer
            model.add(Dense(n_classes, activation='softmax'))




            model.compile(loss="sparse_categorical_crossentropy", optimizer= "adam", metrics=['accu
racy'])

            model.fit(features, label, batch_size=32, epochs=20, validation_split=0.2, callbacks=[t
ensorboard])

            model.save(model_name)
print("done.")
```

```
name C:/Users/AKUMA/Desktop/DESKTOP/UNITY/Python Scripts/saved model/mnist_digit_recog_cnn_3-conv-64-nodes-1-dense-17-04-2020_00-04_Ver14.h5
Train on 8000 samples, validate on 2000 samples
Epoch 1/20
8000/8000 [==============================] - 7s 892us/sample - loss: 0.8262 - accuracy: 0.7346 - val_loss: 0.2587 - val_accuracy: 0.9230
Epoch 2/20
8000/8000 [==============================] - 7s 860us/sample - loss: 0.2852 - accuracy: 0.9139 - val_loss: 0.2053 - val_accuracy: 0.9370
Epoch 3/20
8000/8000 [==============================] - 7s 855us/sample - loss: 0.2050 - accuracy: 0.9392 - val_loss: 0.1548 - val_accuracy: 0.9475
Epoch 4/20
8000/8000 [==============================] - 7s 857us/sample - loss: 0.1627 - accuracy: 0.9511 - val_loss: 0.1763 - val_accuracy: 0.9435
Epoch 5/20
8000/8000 [==============================] - 7s 836us/sample - loss: 0.1243 - accuracy: 0.9625 - val_loss: 0.1247 - val_accuracy: 0.9630
Epoch 6/20
8000/8000 [==============================] - 7s 829us/sample - loss: 0.1054 - accuracy: 0.9680 - val_loss: 0.1156 - val_accuracy: 0.9650
Epoch 7/20
8000/8000 [==============================] - 7s 823us/sample - loss: 0.0970 - accuracy: 0.9703 - val_loss: 0.1218 - val_accuracy: 0.9655
Epoch 8/20
8000/8000 [==============================] - 7s 821us/sample - loss: 0.0780 - accuracy: 0.9759 - val_loss: 0.1683 - val_accuracy: 0.9535
Epoch 9/20
8000/8000 [==============================] - 7s 823us/sample - loss: 0.0746 - accuracy: 0.9768 - val_loss: 0.0938 - val_accuracy: 0.9710
Epoch 10/20
8000/8000 [==============================] - 7s 821us/sample - loss: 0.0609 - accuracy: 0.9809 - val_loss: 0.1148 - val_accuracy: 0.9745
Epoch 11/20
8000/8000 [==============================] - 7s 826us/sample - loss: 0.0535 - accuracy: 0.9849 - val_loss: 0.1198 - val_accuracy: 0.9685
Epoch 12/20
8000/8000 [==============================] - 7s 830us/sample - loss: 0.0444 - accuracy: 0.9865 - val_loss: 0.1191 - val_accuracy: 0.9680
Epoch 13/20
8000/8000 [==============================] - 7s 830us/sample - loss: 0.0426 - accuracy: 0.9862 - val_loss: 0.0939 - val_accuracy: 0.9735
Epoch 14/20
8000/8000 [==============================] - 7s 836us/sample - loss: 0.0367 - accuracy: 0.9889 - val_loss: 0.0987 - val_accuracy: 0.9745
Epoch 15/20
8000/8000 [==============================] - 7s 845us/sample - loss: 0.0342 - accuracy: 0.9881 - val_loss: 0.1070 - val_accuracy: 0.9700
Epoch 16/20
8000/8000 [==============================] - 7s 842us/sample - loss: 0.0334 - accuracy: 0.9898 - val_loss: 0.0947 - val_accuracy: 0.9745
Epoch 17/20
8000/8000 [==============================] - 7s 857us/sample - loss: 0.0206 - accuracy: 0.9939 - val_loss: 0.1152 - val_accuracy: 0.9725
Epoch 18/20
8000/8000 [==============================] - 7s 855us/sample - loss: 0.0241 - accuracy: 0.9934 - val_loss: 0.1005 - val_accuracy: 0.9775
Epoch 19/20
8000/8000 [==============================…
```

RESULT

ACCURACY

- 2 convolutions
- 64 nodes
- 2 dense layers #### seems to be the best configurations among the 27 existing. ###### (top 3 are display below)



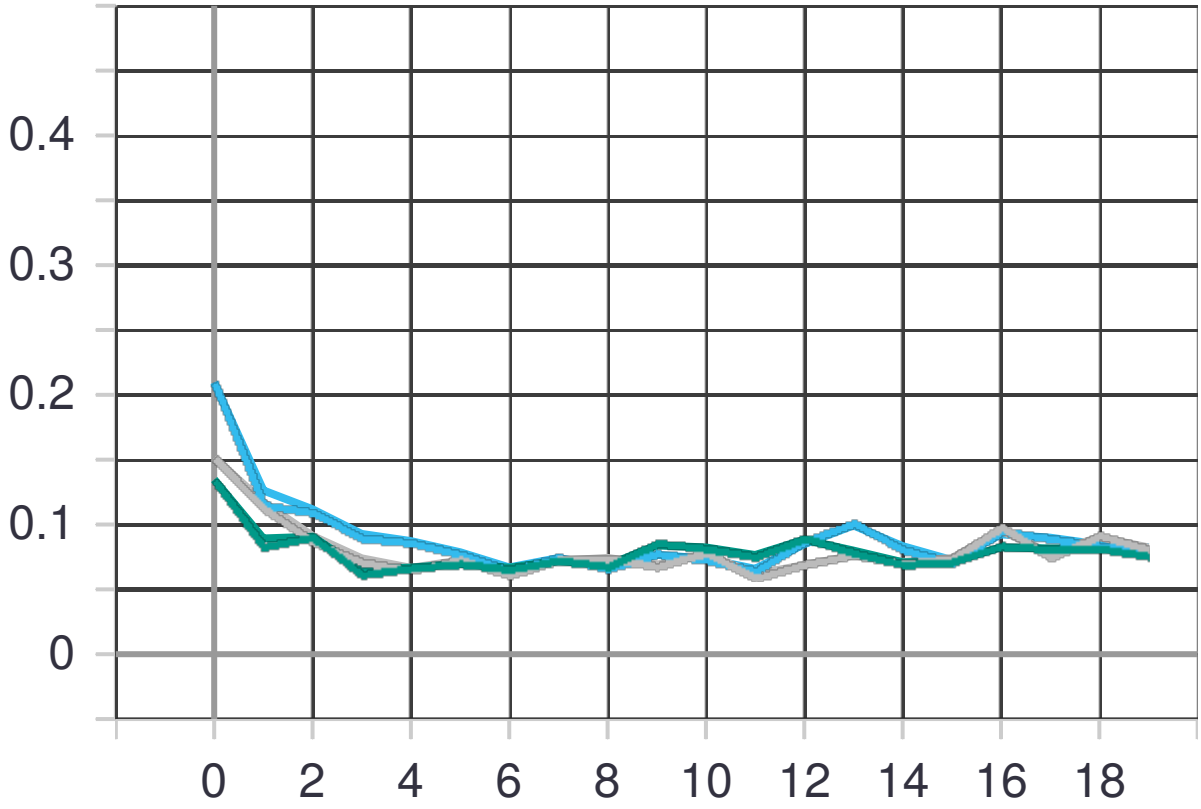
model accuracy: ~ 98%

Name	Smoothed	Value	Step	Time	Relative
train\mnist_digit_recog_cnn_2-conv-128-nodes-1-dense-17-04-2020_00-09_Ver16\validation	0.9867	0.987	19	Fri Apr 17, 00:14:42	4m 37s
train\mnist_digit_recog_cnn_2-conv-64-nodes-1-dense-17-04-2020_00-02_Ver13\validation	0.9841	0.9845	19	Fri Apr 17, 00:04:21	2m 1s

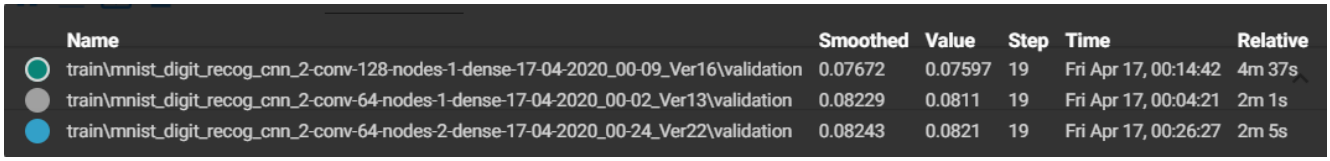
train\mnist_digit_recog_cnn_2-conv-64-nodes-2-dense-17-04-2020_00-24_Ver22\validation 0.9834 0.9835 19 Fri Apr 17, 00:26:27 2m 5s

epoch_accuracy

LOSS



model loss: ~ 0.07%



Name	Smoothed	Value	Step	Time	Relative
train\mnist_digit_recog_cnn_2-conv-128-nodes-1-dense-17-04-2020_00-09_Ver16\validation	0.07672	0.07597	19	Fri Apr 17, 00:14:42	4m 37s
train\mnist_digit_recog_cnn_2-conv-64-nodes-1-dense-17-04-2020_00-02_Ver13\validation	0.08229	0.0811	19	Fri Apr 17, 00:04:21	2m 1s
train\mnist_digit_recog_cnn_2-conv-64-nodes-2-dense-17-04-2020_00-24_Ver22\validation	0.08243	0.0821	19	Fri Apr 17, 00:26:27	2m 5s