In [24…
```
import matplotlib.pyplot as plt
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten,
from tensorflow.keras.optimizers import SGD, Adam
import os
from tensorflow.keras.regularizers import l2
def getData(filename, dirname):
    file_dir = os.path.join('/home/student/Desktop', dirname)
    csv = os.path.join(file_dir, filename)
    return pd.read_csv(csv)
```

In [25…
```
train_set = getData('train.csv', 'digit-recognizer')
test_set = getData('test.csv', 'digit-recognizer')
```

In [26…
```
train_set.shape
x_train, x_test = np.array(train_set.drop("label", axis = 1)) / 255, np.a
y_train = np.array(train_set.label)
```

In [27…
```
x_train.shape
X = x_train.reshape(x_train.shape[0], 28, 28,1)
X.shape
```

Out[27…
```
(42000, 28, 28, 1)
```

In [28…
```
x_test.shape
```

Out[28…
```
(28000, 784)
```

In [29…
```
#reshapes it
X_test = x_test.reshape(x_test.shape[0], 28, 28 ,1)
```

In [30…
```
X_test.shape
```

Out[30…
```
(28000, 28, 28, 1)
```

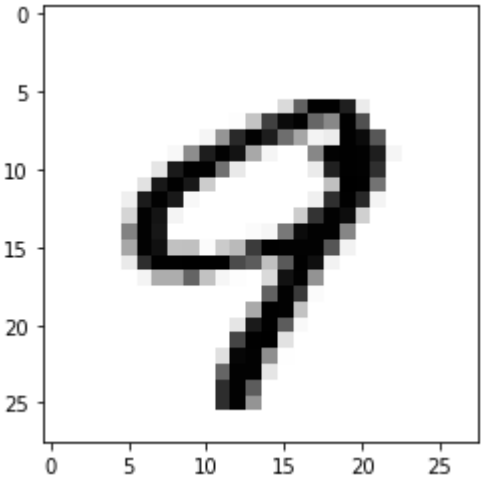
In [ ]:

In [ ]:

In [ ]:

In [31…
```
plt.imshow(X[100], cmap="Greys")
```

Out[31…
```
<matplotlib.image.AxesImage at 0x7f4377847490>
```



In [ ]:

In [ ]:

In [ ]:

In [32…

```
#simple cnn
def create_model():
    with tf.name_scope("cnn"):
        model = Sequential()
        model.add(Conv2D(64, (1,1), activation="relu", input_shape=(28,28
        model.add(Conv2D(64, (3,3), activation="relu",))
        model.add(MaxPooling2D((2,2)))
        model.add(Conv2D(128,(1,1), activation="relu"))
        model.add(Conv2D(128,(3,3), activation="relu"))
        model.add(MaxPooling2D((2,2)))
        model.add(Conv2D(256, (1,1), activation="relu"))
        model.add(Conv2D(256, (3,3), activation="relu"))
        model.add(Dropout(0.5))
        model.add(Flatten())
        model.add(Dense(84, activation="relu"))
        model.add(Dense(42, activation='relu'))
        model.add(Dense(10, activation="softmax"))
        adam = Adam(learning_rate=1e-3)
        model.compile(optimizer=adam, loss="categorical_crossentropy", me
    return model
```

In [33…

```
model = create_model()
model.summary()
```

```
2022-09-06 10:30:57.237297: W tensorflow/stream_executor/platform/default
/dso_loader.cc:64] Could not load dynamic library 'libcuda.so.1'; dlerro
r: libcuda.so.1: cannot open shared object file: No such file or director
y
2022-09-06 10:30:57.237360: W tensorflow/stream_executor/cuda/cuda_drive
r.cc:269] failed call to cuInit: UNKNOWN ERROR (303)
2022-09-06 10:30:57.237409: I tensorflow/stream_executor/cuda/cuda_diagno
stics.cc:156] kernel driver does not appear to be running on this host (s
tudent-OptiPlex-3050): /proc/driver/nvidia/version does not exist
2022-09-06 10:30:57.272202: I tensorflow/core/platform/cpu_feature_guard.
```

```
cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Netwo
rk Library (oneDNN) to use the following CPU instructions in performance-
critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropria
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 64)        128

 conv2d_1 (Conv2D)           (None, 26, 26, 64)        36928

 max_pooling2d (MaxPooling2D  (None, 13, 13, 64)       0
 )

 conv2d_2 (Conv2D)           (None, 13, 13, 128)       8320

 conv2d_3 (Conv2D)           (None, 11, 11, 128)       147584

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 128)        0
 2D)

 conv2d_4 (Conv2D)           (None, 5, 5, 256)         33024

 conv2d_5 (Conv2D)           (None, 3, 3, 256)         590080

 dropout (Dropout)           (None, 3, 3, 256)         0

 flatten (Flatten)           (None, 2304)              0

 dense (Dense)               (None, 84)                193620

 dense_1 (Dense)             (None, 42)                3570

 dense_2 (Dense)             (None, 10)                430

=================================================================
Total params: 1,013,684
Trainable params: 1,013,684
Non-trainable params: 0
_________________________________________________________________
```

In [35…

```
EPOCHS = 5
BATCH_SIZE = 128
#accidental error in the number of batches, but if it'll work , then what
y_one_hot_train = tf.one_hot(y_train, 10)
# encoding , without it the model got a 10% accuracy
history = model.fit(X,y_one_hot_train, epochs=EPOCHS, batch_size = BATCH_
# these values could be do to overfitting or other variances in the data,
#easy way to improve accuracy could be to explore removing some excess of
```

```
Epoch 1/5
329/329 [==============================] - 113s 342ms/step - loss: 0.0805
- accuracy: 0.9757
Epoch 2/5
329/329 [==============================] - 112s 341ms/step - loss: 0.0544
- accuracy: 0.9826
Epoch 3/5
329/329 [==============================] - 112s 340ms/step - loss: 0.0424
- accuracy: 0.9864
Epoch 4/5
329/329 [==============================] - 115s 350ms/step - loss: 0.0364
```

```
- accuracy: 0.9886
Epoch 5/5
329/329 [==============================] - 112s 341ms/step - loss: 0.0289
```

In [36…

```
print(sum(history.history["accuracy"]) / EPOCHS)
```

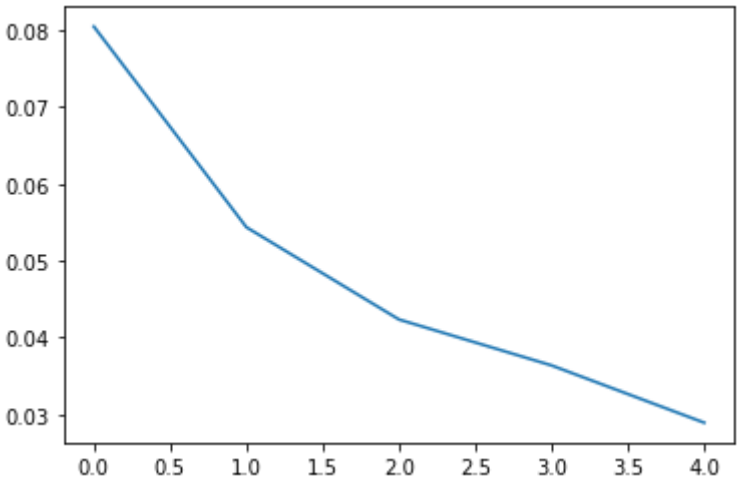
```
0.9848095297813415
```

In [37…

```
plt.plot(history.history["loss"])
plt.show
```

Out[37…

```
<function matplotlib.pyplot.show(close=None, block=None)>
```

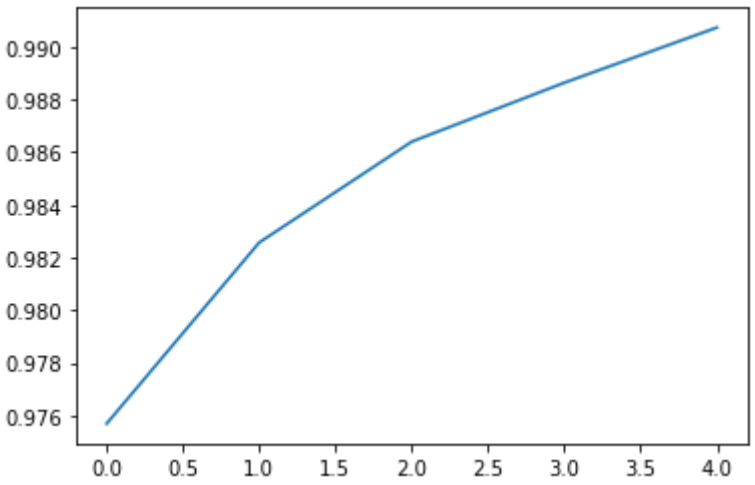


In [38…

```
plt.plot(history.history["accuracy"])
plt.show
```

Out[38…

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [39…

```
pred = model.predict(X_test)
```

```
875/875 [==============================] - 13s 15ms/step
```

In [40…

```
new_pred = tf.math.argmax(pred, axis = -1)
sub = {'ImageId': test_set.index + 1, 'Label': new_pred}
basic_sub = pd.DataFrame(data=sub)
basic_sub.to_csv("submission.csv", index=False)
basic_sub.head()
```

Out[40…

	ImageId	Label
0	1	2
1	2	0
2	3	9
3	4	9
4	5	3

In [ ]: