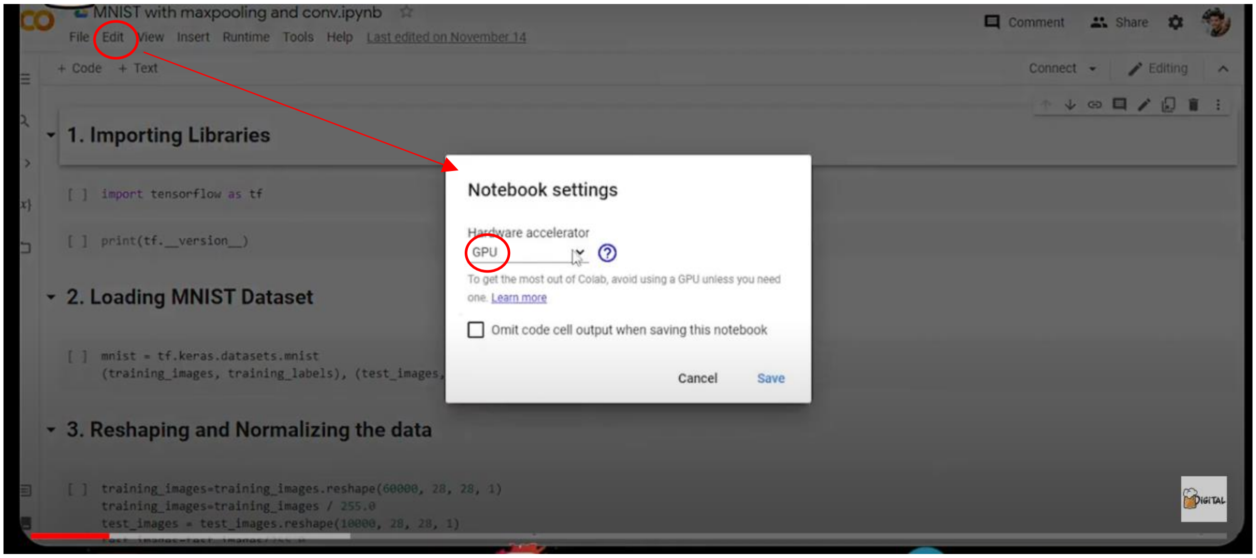
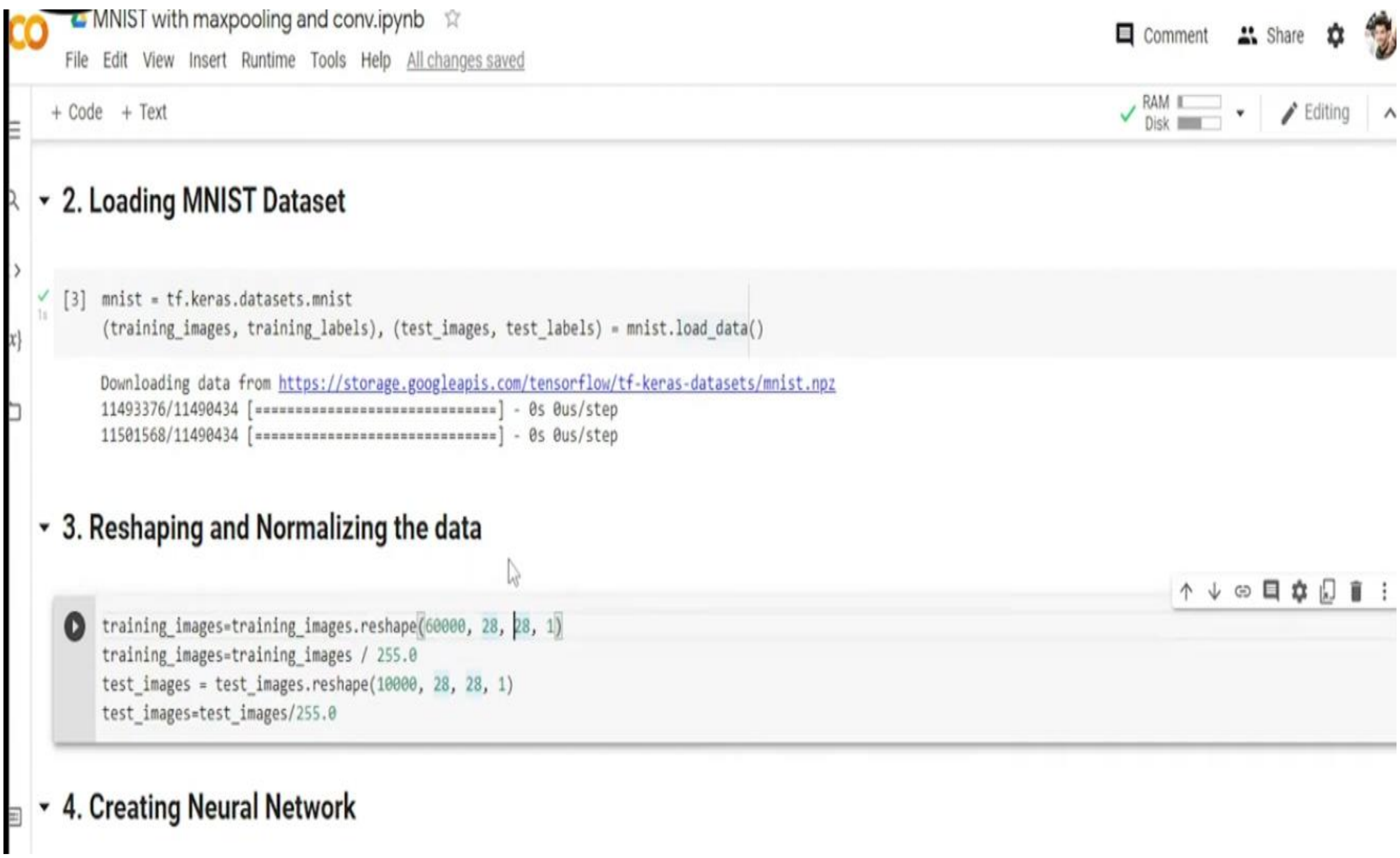
https://www.youtube.com/watch?v=WX4pHOb9P-w





MNIST with maxpooling and conv.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

```
[4] test_images = test_images.reshape(10000, 28, 28, 1)
    test_images=test_images/255.0
```

4. Creating Neural Network

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(28, 28, 1)),
  tf.keras.layers.MaxPooling2D(2, 2),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(10, activation='softmax')
])
```

Activation function = relu

2D = two dimension
32=Filter
3x3 = matrix

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

```
[ ] model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

5. Training and evaluating the data

MNIST with maxpooling and conv.ipynb

File Edit View Insert Runtime Tools Help

Comment Share

+ Code + Text

RAM Disk Editing

```
      tf.keras.layers.Dense(128, activation='relu'),
[5]   tf.keras.layers.Dense(10, activation='softmax')
    ])
```

```
[6] model.summary()
```

#Check summary

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
 )

 flatten (Flatten)           (None, 5408)              0

 dense (Dense)               (None, 128)               692352

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 693,962
Trainable params: 693,962
Non-trainable params: 0
_________________________________________________________________
```

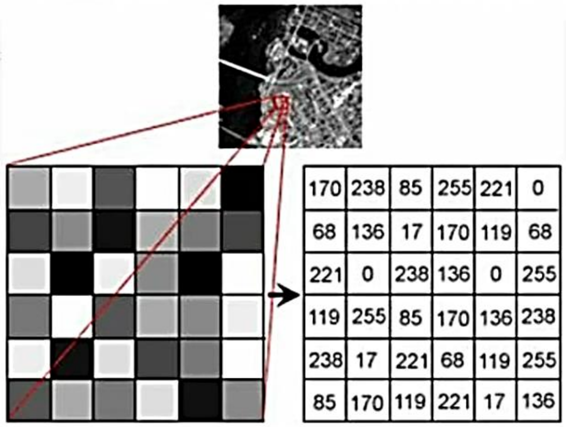
MNIST with maxpooling and conv.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(28,
  tf.keras.layers.MaxPooling2D(2, 2),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(10, activation='softmax')
])
```




```
[6] model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
 )

 flatten (Flatten)           (None, 5408)              0

 dense (Dense)               (None, 128)               692352

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 693,962
Trainable params: 693,962
Non-trainable params: 0
_________________________________________________________________
```

```
[ ] model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

5. Training and evaluating the data

```
[ ] model.fit(training_images, training_labels, epochs=10)
```

MNIST with maxpooling and conv.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

```
[6]
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
 )

 flatten (Flatten)           (None, 5408)              0

 dense (Dense)               (None, 128)               692352

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 693,962
Trainable params: 693,962
Non-trainable params: 0
_________________________________________________________________
```

⇨ ***# Can use with the help of "call back", it is best to use call back***

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

⇨ ***# Compiling of created model***

5. Training and evaluating the data

```
model.fit(training_images, training_labels, epochs=10)

Epoch 1/10
1875/1875 [==============================] - 39s 5ms/step - loss: 0.1427 - accuracy: 0.9583
Epoch 2/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0499 - accuracy: 0.9845
Epoch 3/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0312 - accuracy: 0.9904
Epoch 4/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0207 - accuracy: 0.9932
Epoch 5/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0145 - accuracy: 0.9955
Epoch 6/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0101 - accuracy: 0.9967
Epoch 7/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0075 - accuracy: 0.9976
Epoch 8/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0070 - accuracy: 0.9976
Epoch 9/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0043 - accuracy: 0.9986
Epoch 10/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0039 - accuracy: 0.9988
```

```
test_loss, test_acc = model.evaluate(test_images, test_labels)
print(test_acc)
```

MNIST with maxpooling and conv.ipynb

File Edit View Insert Runtime Tools Help Saving...

Comment Share

+ Code + Text

RAM Disk Editing

[8]

```
Epoch 3/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0312 - accuracy: 0.9904
Epoch 4/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0207 - accuracy: 0.9932
Epoch 5/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0145 - accuracy: 0.9955
Epoch 6/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0101 - accuracy: 0.9967
Epoch 7/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0075 - accuracy: 0.9976
Epoch 8/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0070 - accuracy: 0.9976
Epoch 9/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0043 - accuracy: 0.9986
Epoch 10/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0039 - accuracy: 0.9988
<keras.callbacks.History at 0x7f39730fed50>
```

[9]

```
test_loss, test_acc = model.evaluate(test_images, test_labels)
print(test_acc)
```

```
313/313 [==============================] - 1s 4ms/step - loss: 0.0672 - accuracy: 0.9844
0.9843999743461609
```