LAPORAN AKHIR PRAKTIKUM

Mata Praktikum	: Kecerdasan Artificial
Kelas	: 3IA11
Praktikum ke-	2
Tanggal	: 2310/2024
Materi	: Neural Network
NPM	: 51422161
Nama	: MUHAMMAD TARMIDZI BARIQ
Ketua Asisten	: Gilbert Jefferson Faozato Mendrofa
Paraf Asisten	:
Nama Asisten	: Filbert
Jumlah Lembar	: 11 Lembar

LABORATORIUM TEKNIK INFORMATIKA

UNIVERSITAS GUNADARMA

2024

LISTING PROGRAM

(Copy koding yang sudah dikerjakan)

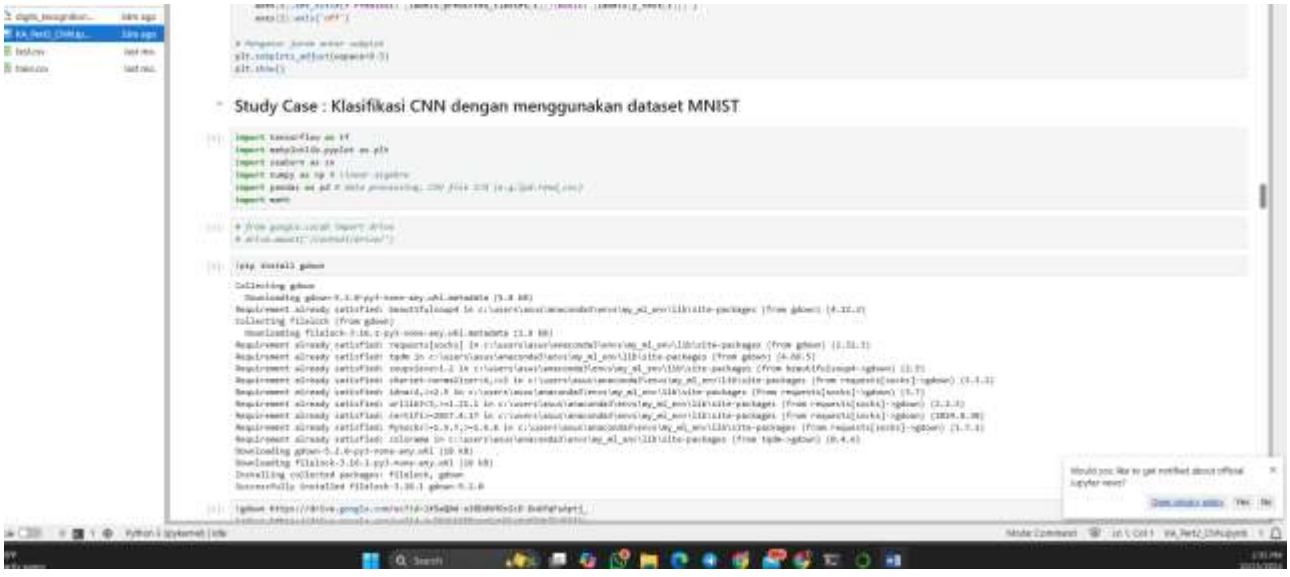
1. Jelaskan tentang apa itu convolution neural network

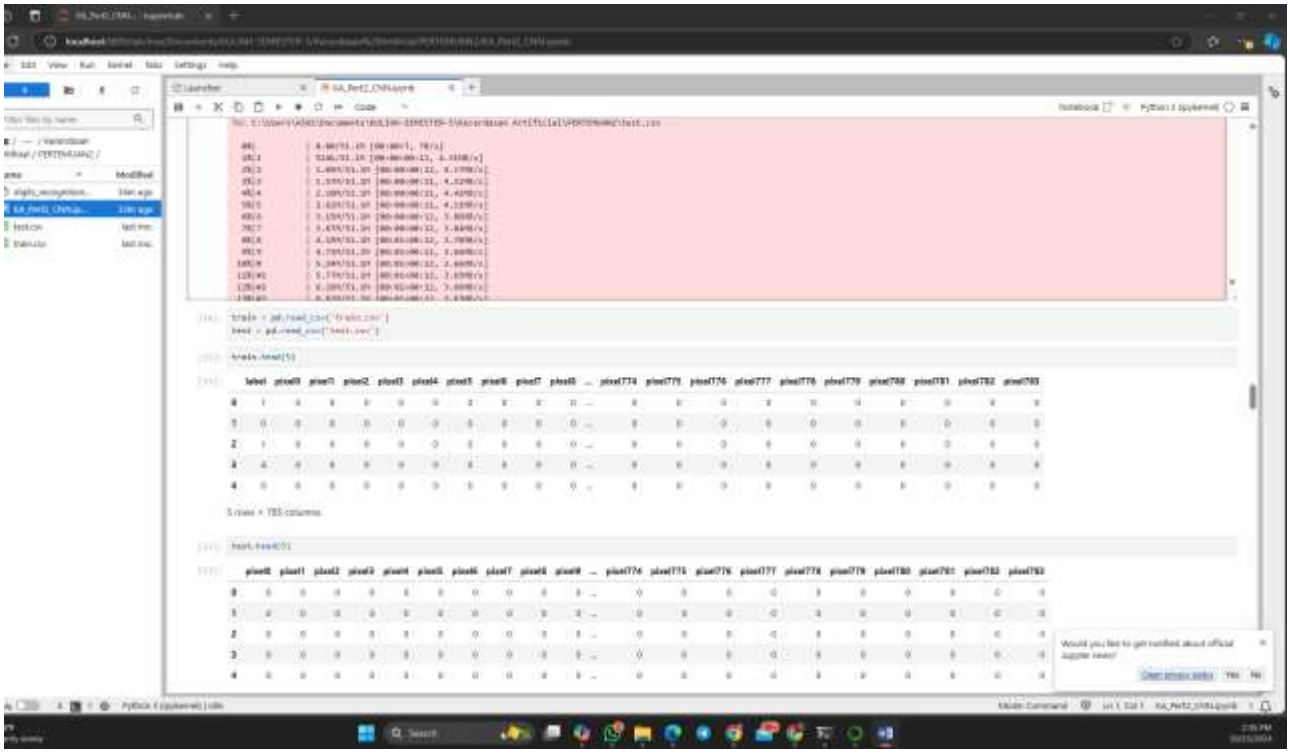
bagian dari machine learning, dan merupakan inti dari algoritma pembelajaran mendalam. Jaringan ini terdiri dari lapisan node, yang terdiri atas sebuah lapisan input, satu atau lebih lapisan tersembunyi, dan lapisan output.

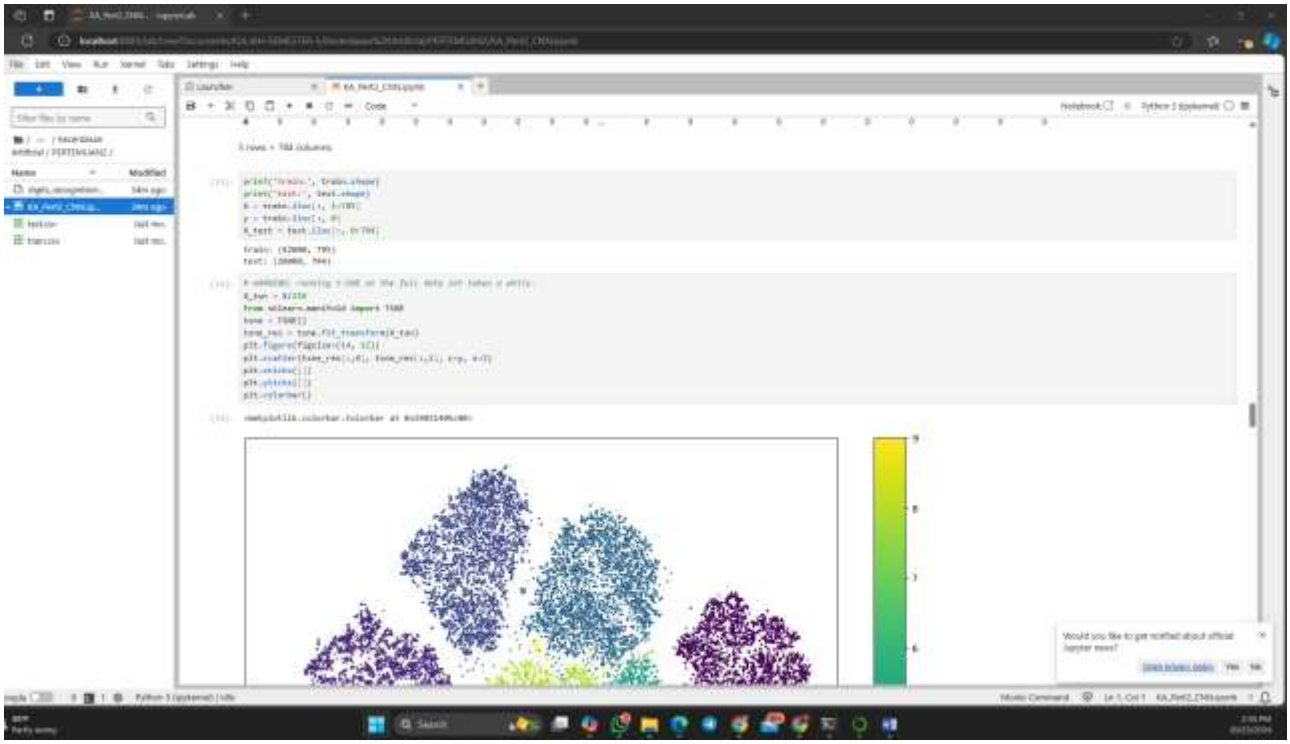
CNN memiliki tiga jenis lapisan utama, yaitu:

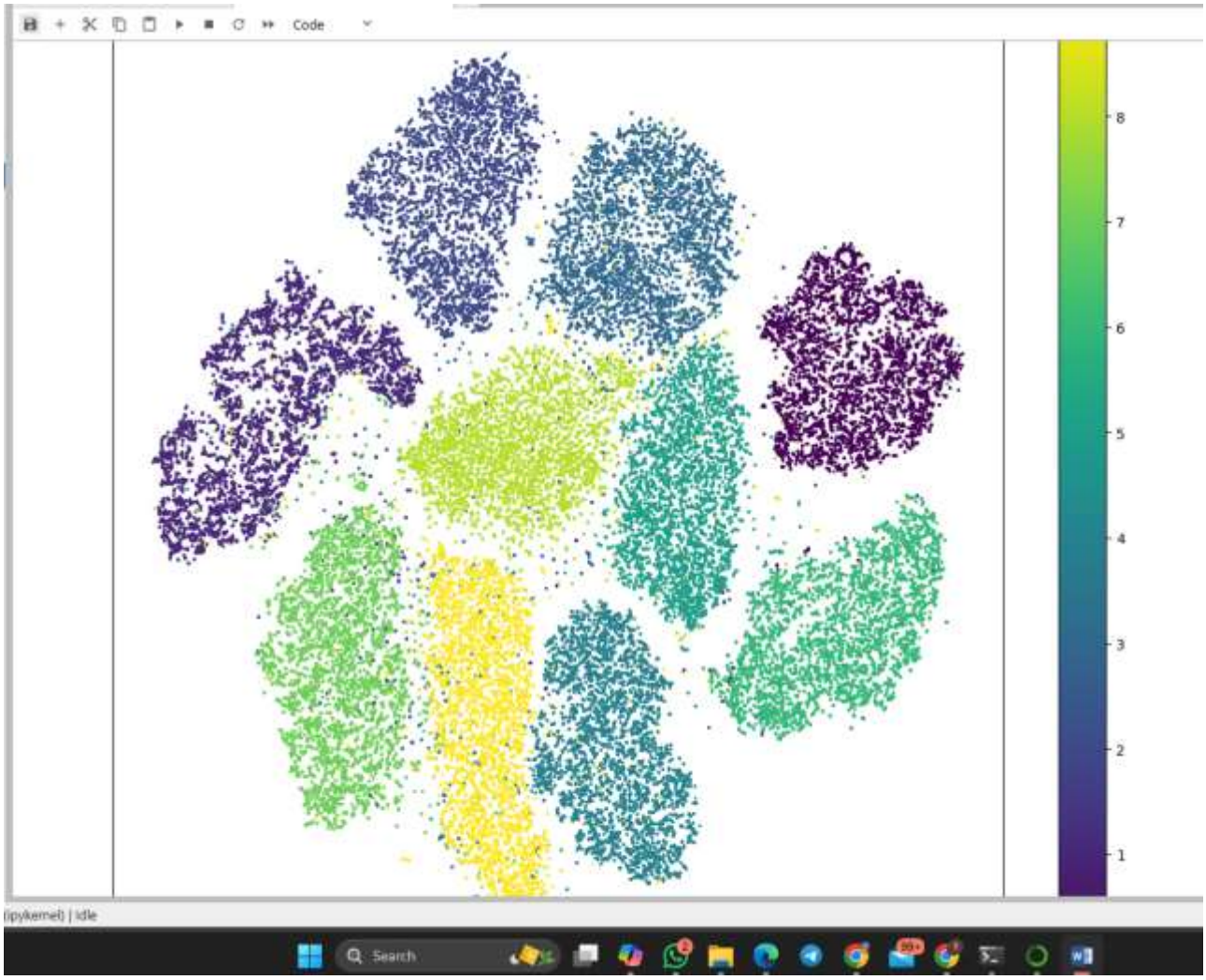
- Lapisan konvolusi : blok bangunan inti dari CNN, dan di sinilah terjadinya sebagian besar komputasi.
- Lapisan penyatuan (pooling) : juga dikenal sebagai downsampling, melakukan reduksi dimensi, mengurangi jumlah parameter dalam input.
- Lapisan yang sepenuhnya terhubung (FC-fully connected) : Lapisan ini melakukan tugas klasifikasi berdasarkan fitur yang diekstraksi melalui lapisan sebelumnya dan filter yang berbeda. Sementara lapisan konvolusional dan penyatuan cenderung menggunakan fungsi ReLu, lapisan FC biasanya memanfaatkan fungsi aktivasi softmax untuk mengklasifikasikan input dengan tepat, menghasilkan probabilitas dari 0 hingga 1.

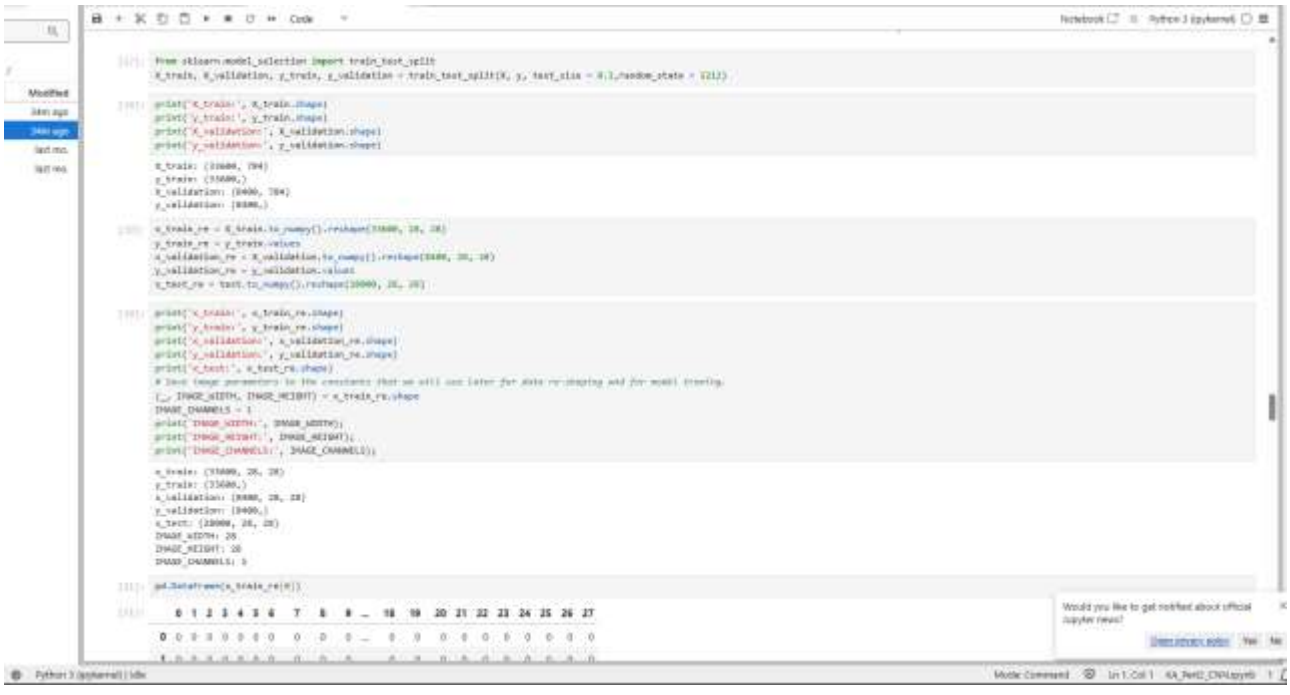
2. Jalankan study Case: kalsifikasi CNN dengan menggunakan dataset MNIST dan screenshot output programnya

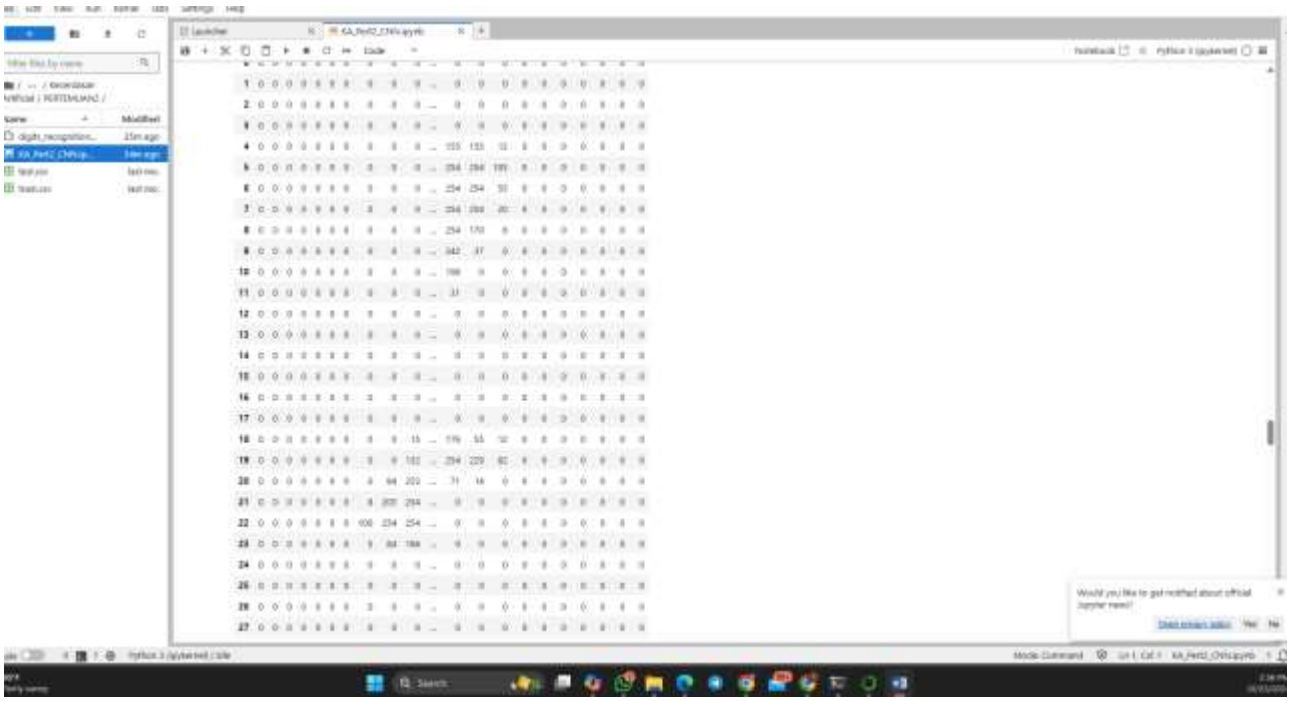


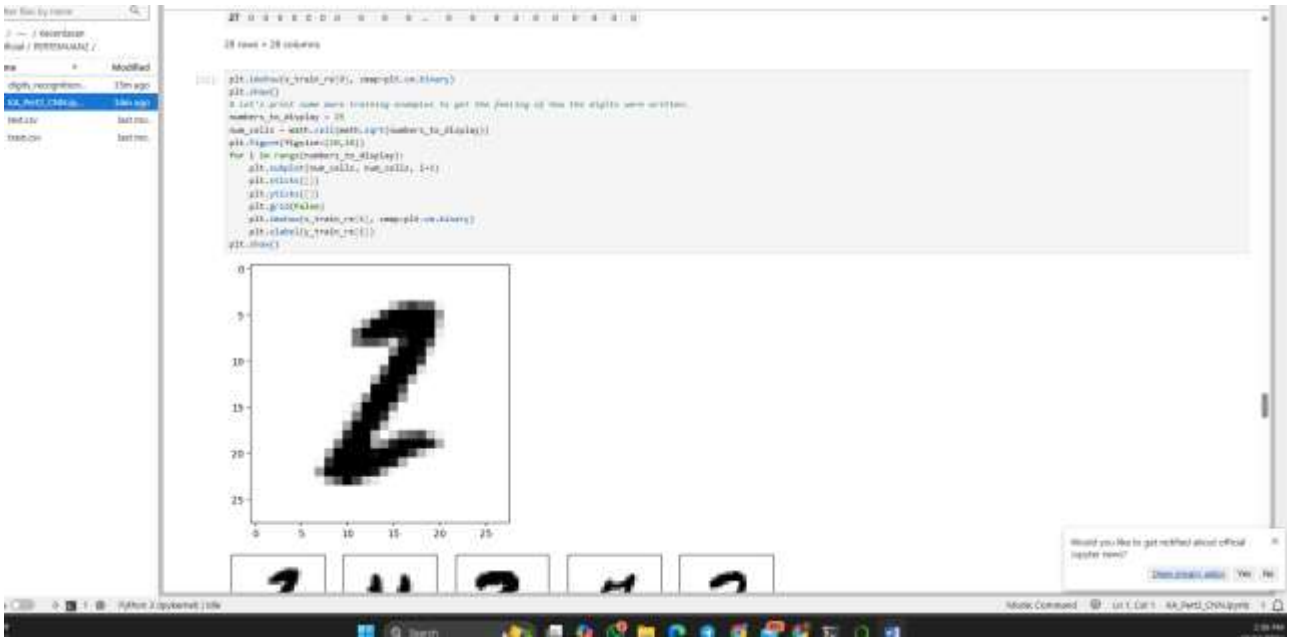


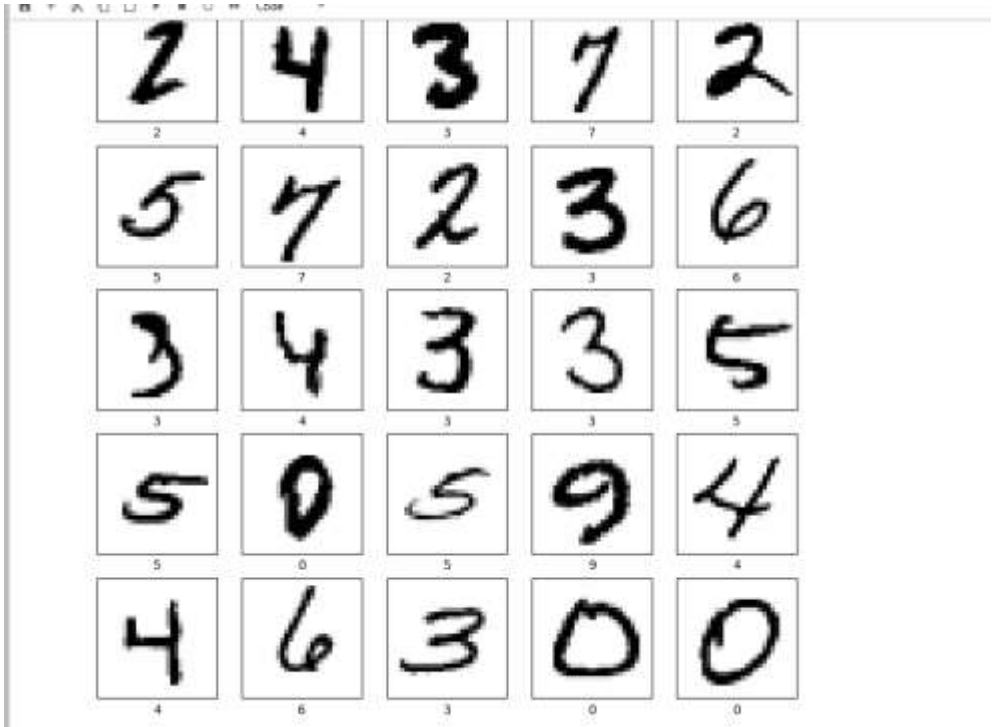



Code
8
7
6
5
4
3
2
1
(ipykernel) | idle
Search


Code
Notebook
Python 3 (ipykernel)
Modified
Mode: Command
Ln 1, Col 1








```
x_train_with_chanels = x_train_re.reshape(
```

Launcher | KA_Part2_CNN.ipynb

Code

```
x_train_with_chanels = x_train_re.reshape(
    x_train_re.shape[0],
    IMAGE_WIDTH,
    IMAGE_HEIGHT,
    IMAGE_CHANNELS
)
x_validation_with_chanels = x_validation_re.reshape(
    x_validation_re.shape[0],
    IMAGE_WIDTH,
    IMAGE_HEIGHT,
    IMAGE_CHANNELS
)
x_test_with_chanels = x_test_re.reshape(
    x_test_re.shape[0],
    IMAGE_WIDTH,
    IMAGE_HEIGHT,
    IMAGE_CHANNELS
)
print('x_train_with_chanels:', x_train_with_chanels.shape)
print('x_validation_with_chanels:', x_validation_with_chanels.shape)
print('x_test_with_chanels:', x_test_with_chanels.shape)
```

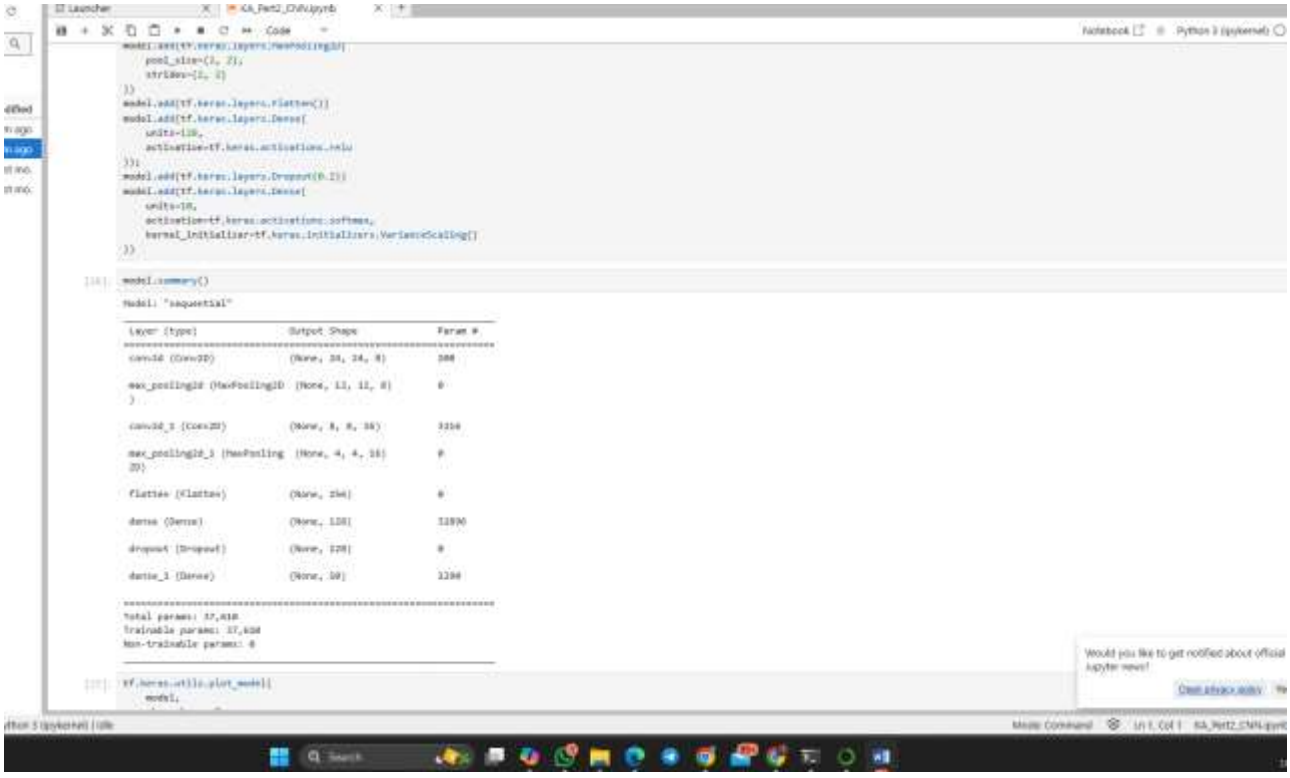
```
x_train_with_chanels: (33600, 28, 28, 1)
x_validation_with_chanels: (8400, 28, 28, 1)
x_test_with_chanels: (28000, 28, 28, 1)
```

```
x_train_normalized = x_train_with_chanels / 255
x_validation_normalized = x_validation_with_chanels / 255
x_test_normalized = x_test_with_chanels / 255
```

```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Convolution2D(
    input_shape=(IMAGE_WIDTH, IMAGE_HEIGHT, IMAGE_CHANNELS),
    kernel_size=5,
    filters=8,
    strides=1,
    activation=tf.keras.activations.relu,
    kernel_initializer=tf.keras.initializers.VarianceScaling()
))
model.add(tf.keras.layers.MaxPooling2D(
    pool_size=(2, 2),
    strides=(2, 2)
))
model.add(tf.keras.layers.Convolution2D(
    kernel_size=5,
    filters=16,
    strides=1,
```

on 3 (ipykernel) | Idle

Search



```
model.add(tf.keras.layers.MaxPooling2D(
    pool_size=(2, 2),
    strides=(2, 2)
))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(
    units=128,
    activation=tf.keras.activations.relu
));
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Dense(
    units=10,
    activation=tf.keras.activations.softmax,
    kernel_initializer=tf.keras.initializers.VarianceScaling()
))
```

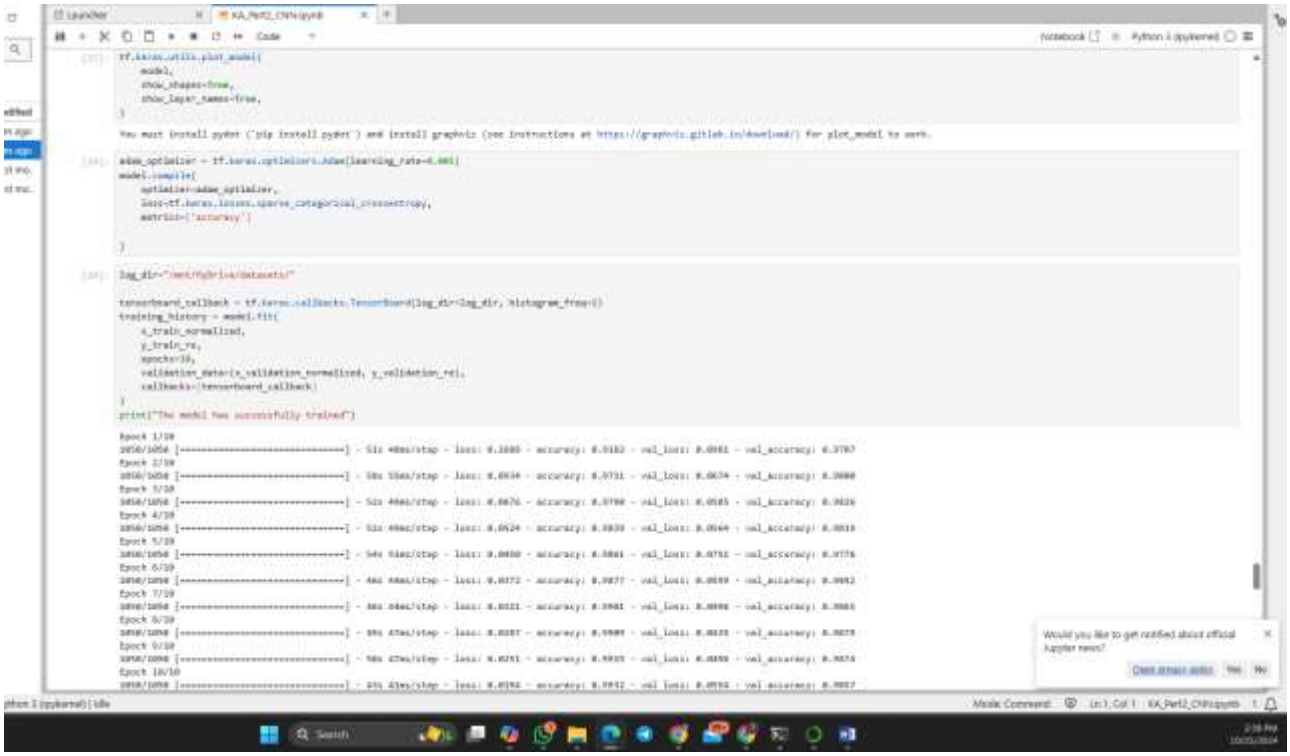
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 24, 24, 8)         208

 max_pooling2d (MaxPooling2D  (None, 12, 12, 8)        0
 )

 conv2d_1 (Conv2D)           (None, 8, 8, 16)          3216

 max_pooling2d_1 (MaxPooling  (None, 4, 4, 16)         0
 2D)

 flatten (Flatten)           (None, 256)               0

 dense (Dense)               (None, 128)               32896

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 37,610
Trainable params: 37,610
Non-trainable params: 0
_________________________________________________________________
```

```
tf.keras.utils.plot_model(
    model,
```

Would you like to get notified about official Jupyter news?

Clear privacy policy Ye



```
tf.keras.utils.plot_model(
    model,
    show_shapes=True,
    show_layer_names=True,
)
```

```
You must install pydot (`pip install pydot`) and install graphviz (see instructions at https://graphviz.gitlab.io/download/) for plot_model to work.
```

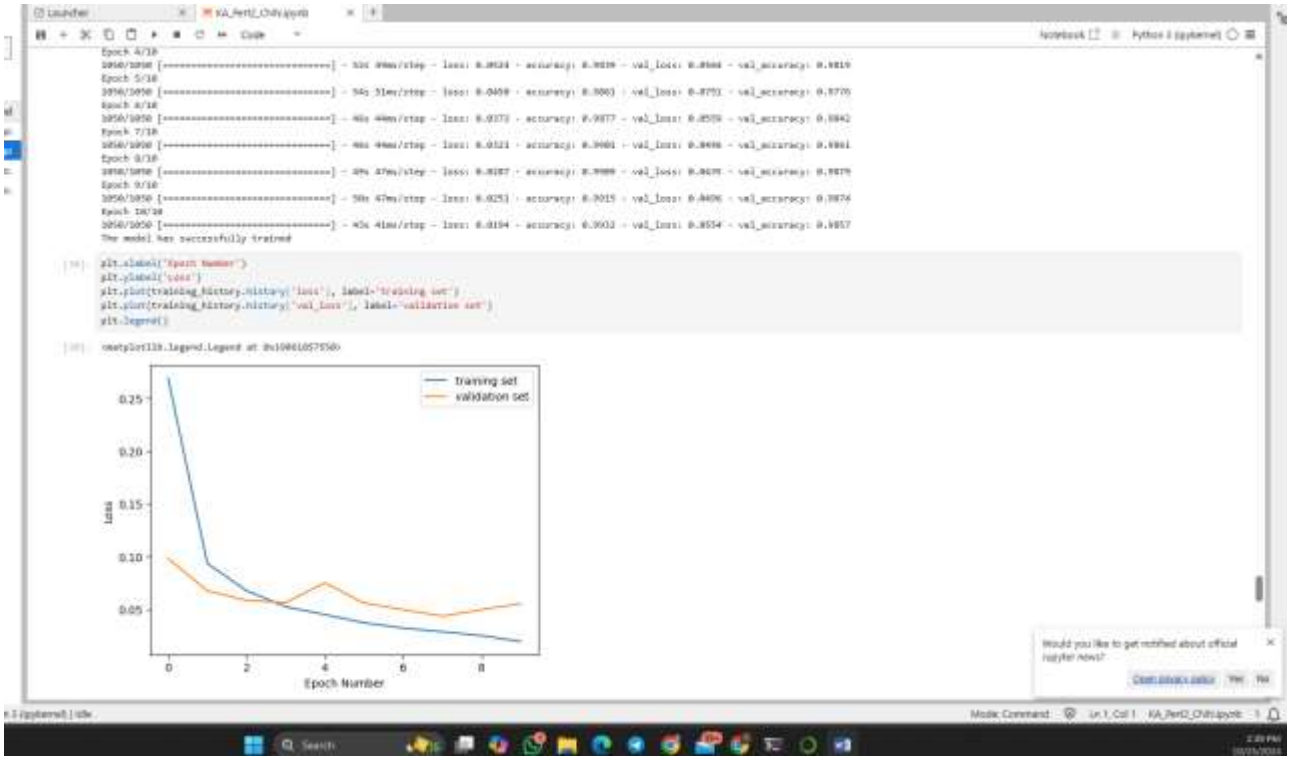
```
adam_optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)
model.compile(
    optimizer=adam_optimizer,
    loss=tf.keras.losses.sparse_categorical_crossentropy,
    metrics=['accuracy']

)
```

```
log_dir="/mnt/Hybrid/Datasets/"

tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)
training_history = model.fit(
    x_train_normalized,
    y_train_re,
    epochs=10,
    validation_data=(x_validation_normalized, y_validation_re),
    callbacks=[tensorboard_callback]
)
print("The model has successfully trained")
```

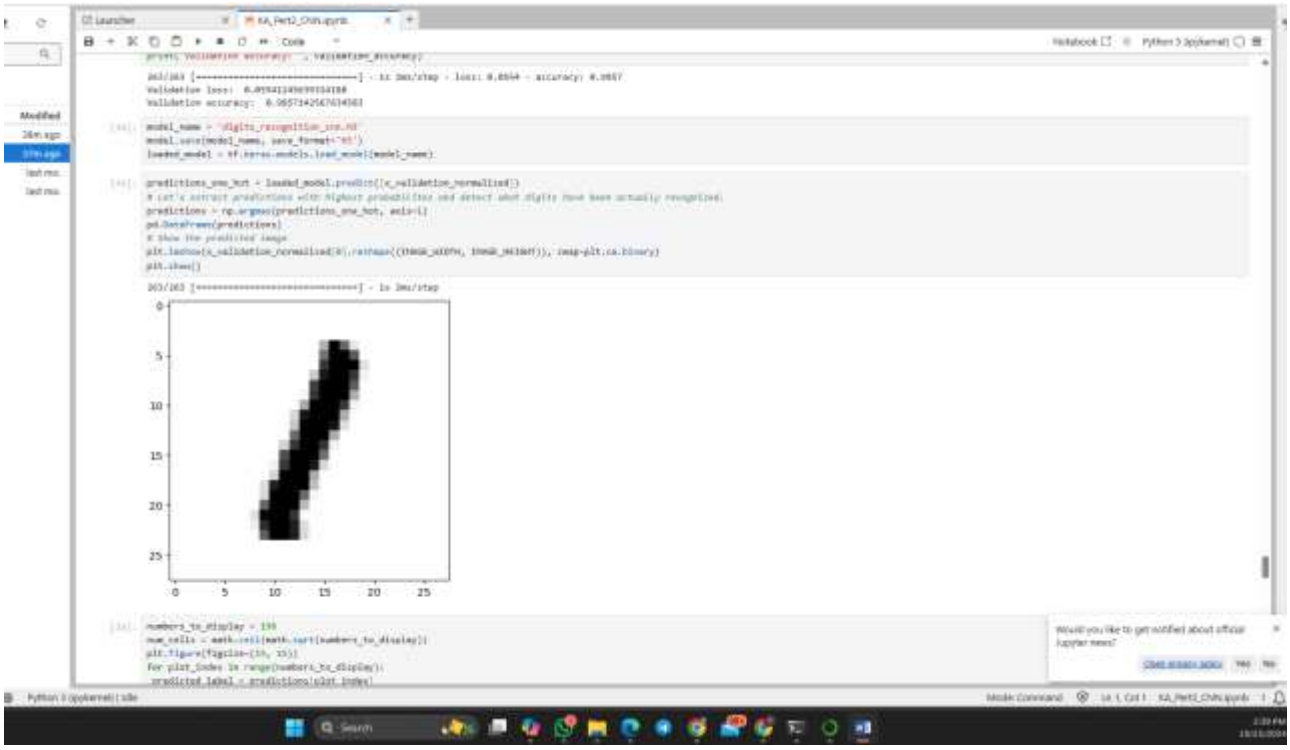
```
Epoch 1/10
1850/1850 [==============================] - 51s 40ms/step - loss: 0.1600 - accuracy: 0.9182 - val_loss: 0.0981 - val_accuracy: 0.9767
Epoch 2/10
1850/1850 [==============================] - 50s 50ms/step - loss: 0.0634 - accuracy: 0.9731 - val_loss: 0.0674 - val_accuracy: 0.9808
Epoch 3/10
1850/1850 [==============================] - 52s 40ms/step - loss: 0.0476 - accuracy: 0.9788 - val_loss: 0.0585 - val_accuracy: 0.9826
Epoch 4/10
1850/1850 [==============================] - 52s 40ms/step - loss: 0.0524 - accuracy: 0.9839 - val_loss: 0.0644 - val_accuracy: 0.9819
Epoch 5/10
1850/1850 [==============================] - 54s 51ms/step - loss: 0.0400 - accuracy: 0.9861 - val_loss: 0.0751 - val_accuracy: 0.9776
Epoch 6/10
1850/1850 [==============================] - 46s 44ms/step - loss: 0.0372 - accuracy: 0.9877 - val_loss: 0.0699 - val_accuracy: 0.9862
Epoch 7/10
1850/1850 [==============================] - 46s 44ms/step - loss: 0.0321 - accuracy: 0.9901 - val_loss: 0.0696 - val_accuracy: 0.9865
Epoch 8/10
1850/1850 [==============================] - 49s 47ms/step - loss: 0.0287 - accuracy: 0.9909 - val_loss: 0.0838 - val_accuracy: 0.9879
Epoch 9/10
1850/1850 [==============================] - 50s 47ms/step - loss: 0.0291 - accuracy: 0.9915 - val_loss: 0.0898 - val_accuracy: 0.9874
Epoch 10/10
1850/1850 [==============================] - 47s 47ms/step - loss: 0.0291 - accuracy: 0.9912 - val_loss: 0.0956 - val_accuracy: 0.9857
```

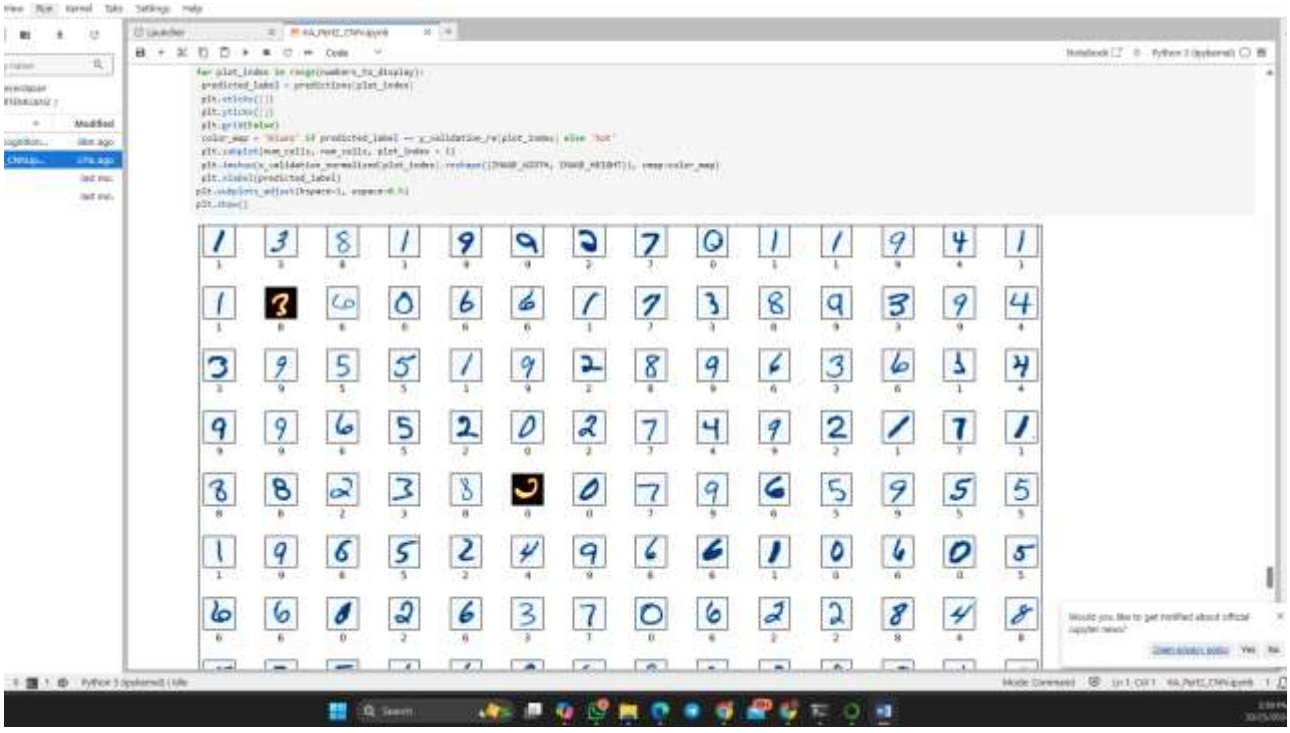
Would you like to get notified about official Jupyter news?

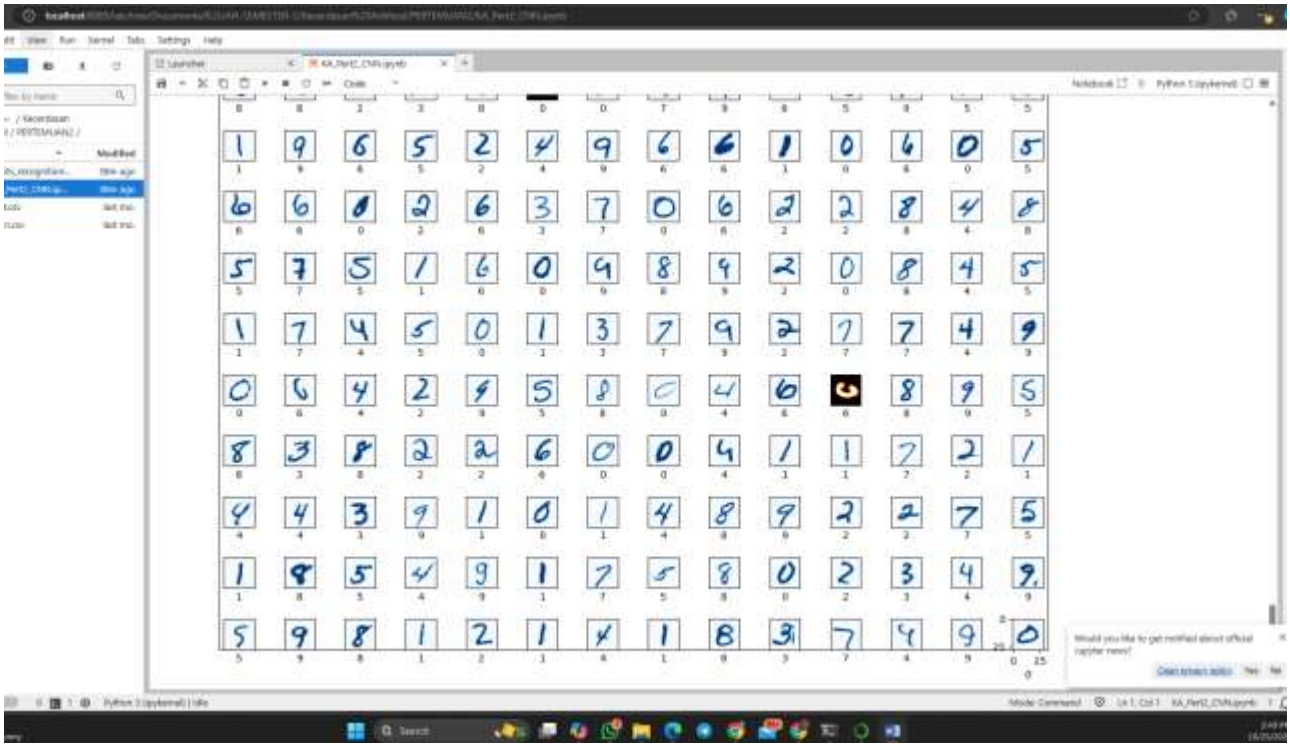
Clear privacy policy Yes No

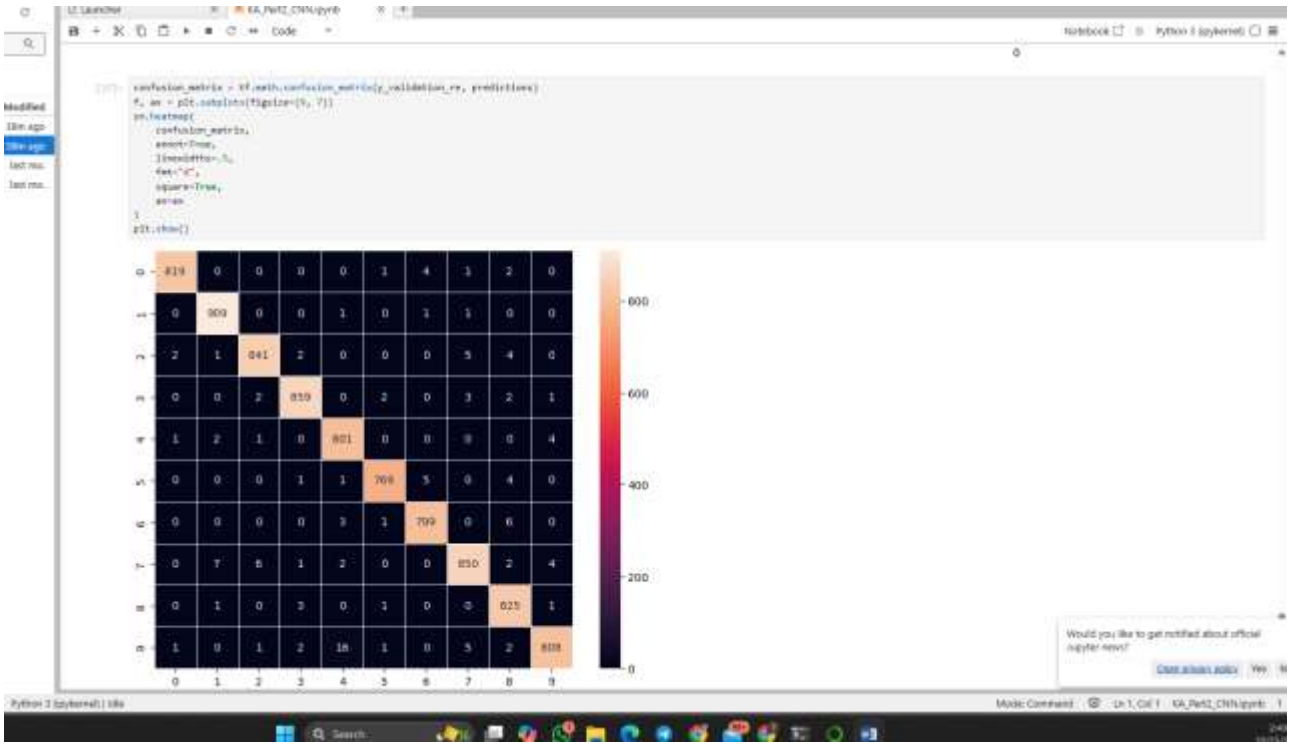

training set
validation set
Loss
Epoch Number
0.25
0.20
0.15
0.10
0.05
0
2
4
6
8


training set
validation set
Accuracy
Epoch Number
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0.92
0
2
4
6
8








confusion_matrix = tf.math.confusion_matrix(y_validation_re, predictions)
f, ax = plt.subplots(figsize=(9, 7))
sn.heatmap(
confusion_matrix,
annot=True,
linewidths=.5,
fmt="d",
square=True,
ax=ax
)
plt.show()
819 0 0 0 0 1 4 1 2 0
0 809 0 0 1 0 1 1 0 0
2 1 841 2 0 0 0 5 4 0
0 0 2 859 0 2 0 3 2 1
1 2 1 0 801 0 0 0 0 4
0 0 0 1 1 769 5 0 4 0
0 0 0 0 3 1 799 0 6 0
0 7 6 1 2 0 0 850 2 4
0 1 0 3 0 1 0 0 825 1
1 0 1 2 16 1 0 5 2 808
0 1 2 3 4 5 6 7 8 9
800
600
400
200
0