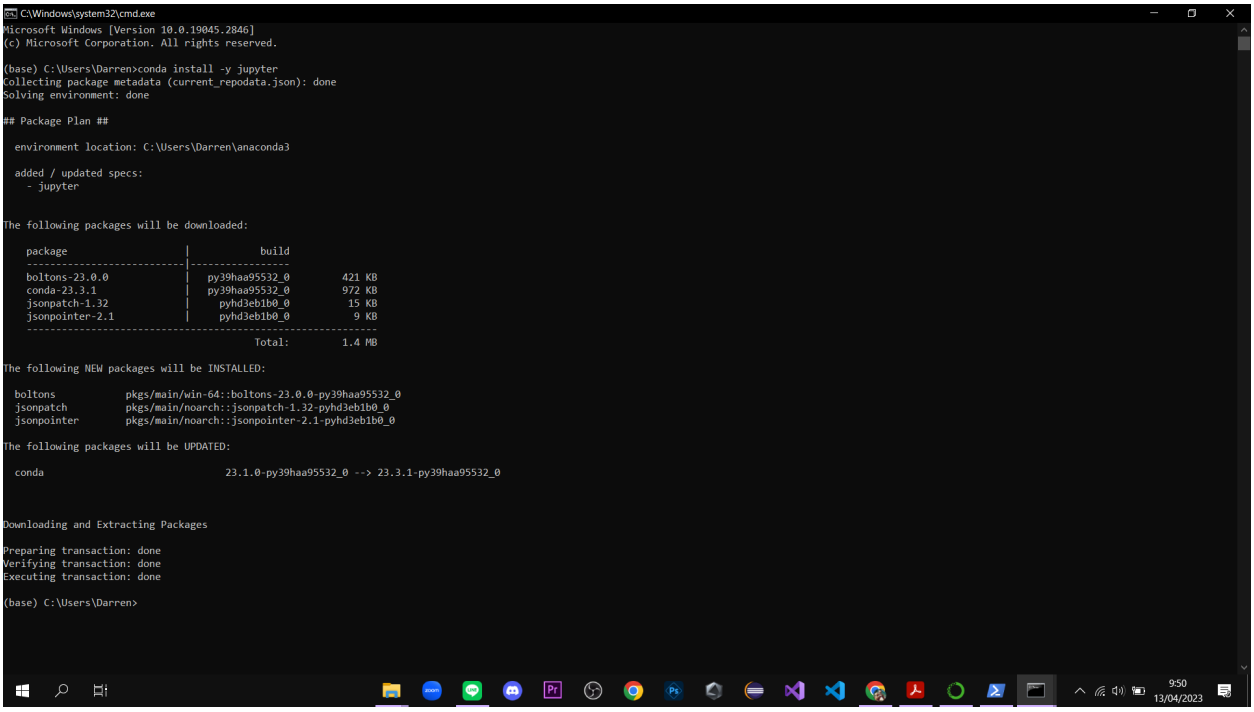
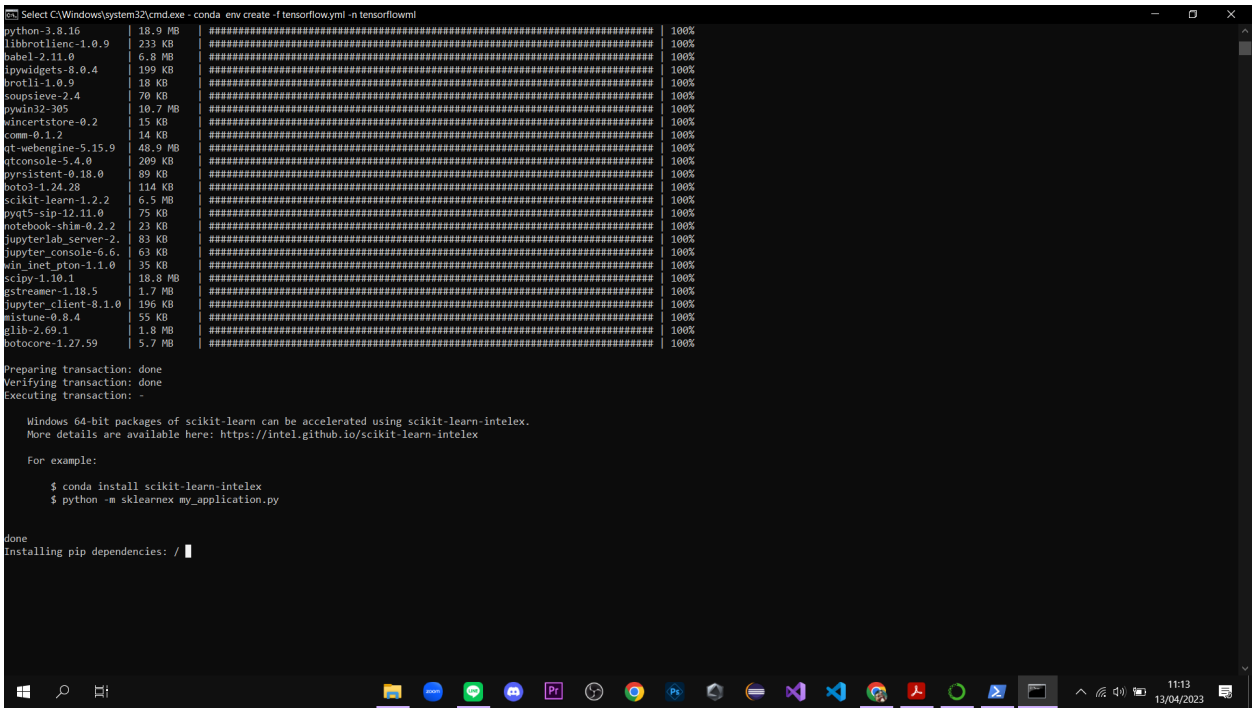
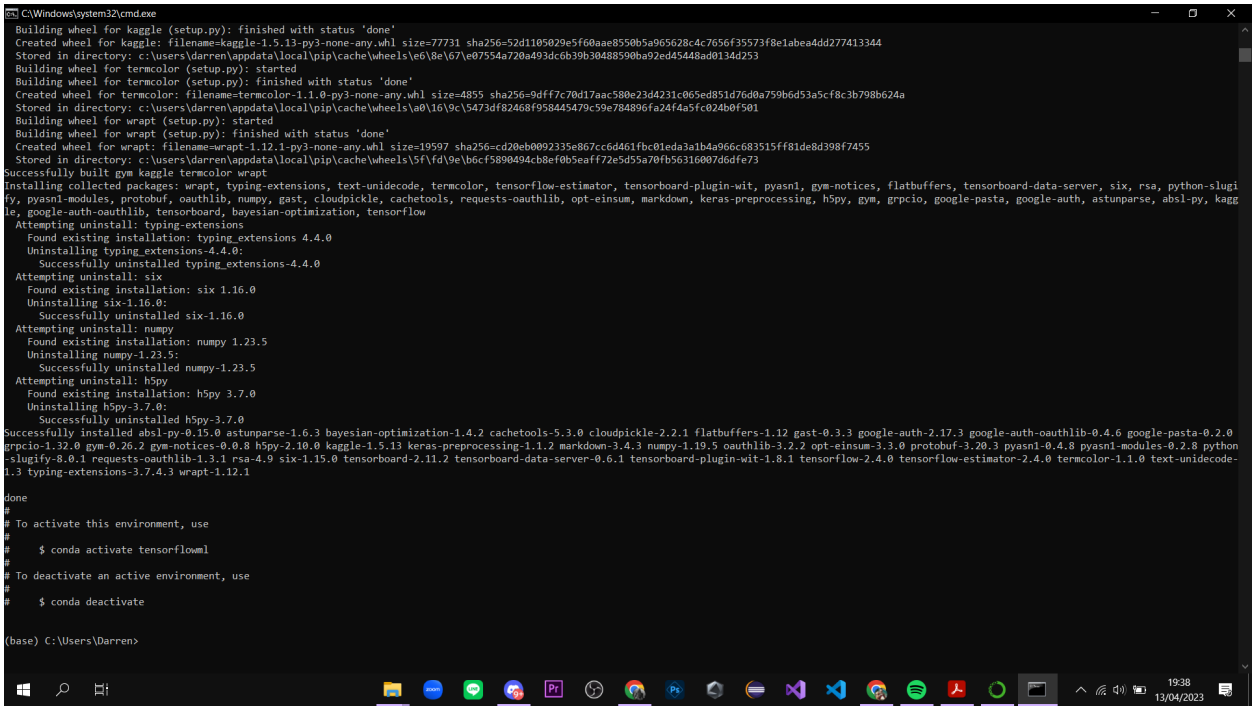
Installing jupyter notebook







Activate tensorflow

```
(base) C:\Users\Darren>$ conda activate tensorflow
'$' is not recognized as an internal or external command,
operable program or batch file.

(base) C:\Users\Darren>conda activate tensorflowml

(tensorflowml) C:\Users\Darren>
```

Install nb_conda

```
(tensorflowml) C:\Users\Darren>conda install nb_conda
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\Darren\anaconda3\envs\tensorflowml

  added / updated specs:
    - nb_conda


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    nb_conda-2.2.1             |           py38_1          34 KB
    nb_conda_kernels-2.3.1     |   py38haa95532_0          28 KB
    ------------------------------------------------------------
                                           Total:          62 KB

The following NEW packages will be INSTALLED:

  nb_conda           pkgs/main/win-64::nb_conda-2.2.1-py38_1
  nb_conda_kernels   pkgs/main/win-64::nb_conda_kernels-2.3.1-py38haa95532_0


Proceed ([y]/n)? y


Downloading and Extracting Packages

Preparing transaction: done
Verifying transaction: done
Executing transaction: | Enabling nb_conda_kernels...
CONDA_PREFIX: C:\Users\Darren\anaconda3\envs\tensorflowml
Status: enabled

| Config option `kernel_spec_manager_class` not recognized by `EnableNBExtensionApp`.
Enabling notebook extension nb_conda/main...
      - Validating: ok
Enabling tree extension nb_conda/tree...
      - Validating: ok
Config option `kernel_spec_manager_class` not recognized by `EnableServerExtensionApp`.
Enabling: nb_conda
- Writing config: C:\Users\Darren\anaconda3\envs\tensorflowml\etc\jupyter
    - Validating...
      nb_conda 2.2.1 ok
```

TUGAS LAB IF540 MACHINE LEARNING

WEEK 08 : Pembelajaran Mendalam

Semester Genap 2022/2023

In [1]:
```python
# Run this code when you restart the machine
# Fill in with YOUR name and NIM
import datetime
import uuid

myName = "Christopher Darren"
myNIM = "00000054804"
```

In [2]:
```python
myDate = datetime.datetime.now()
myDevice = str(uuid.uuid1())

print("Name: \t\t{}".format(myName))
print("NIM: \t\t{}".format(myNIM))
print("Start: \t\t{}".format(myDate))
print("Device ID: \t{}".format(myDevice))
```

```
Name:           Christopher Darren
NIM:            00000054804
Start:          2023-04-13 09:14:43.885427
Device ID:      f4874687-d9a0-11ed-a053-f02f74a116e8
```

Dataset yang dipakai:

1. KMNIST – sumber : https://www.kaggle.com/datasets/sivakrishnathota/sivakmnist?select=kmnist-train-labels.npz

---

Hasil kerja

Lets install Jupyter, which the editor we will use in this course :

In [2]:
```python
#conda install -y jupyter
```

We will actually lunch Jupyter later.

In [3]:
```python
#conda env create -f tensorflow.yml -n tensorflow_env
```

In [4]:
```python
#conda activate tensorflow_env
```

In [5]:
```python
conda info --envs
```

```
# conda environments:
#
base                     C:\Users\Darren\anaconda3
tensorflow_env           C:\Users\Darren\anaconda3\envs\tensorflow_env
tensorflowml          *  C:\Users\Darren\anaconda3\envs\tensorflowml


Note: you may need to restart the kernel to use updated packages.
```

Importing system library

Testing your Environment

Run your Jupyter notebook and check the versions expected.

In [6]:
```python
import sys

import tensorflow.keras
import pandas as pd
import sklearn as sk
import tensorflow as tf
```

From the code, we get print the output with this following code

In [7]:
```python
print(f"Tensor Flow Version: {tf.__version__}")
print(f"Keras Version: {tensorflow.keras.__version__}")
print()
print(f"Python {sys.version}")
print(f"Pandas {pd.__version__}")
print(f"Scikit-Learn {sk.__version__}")
```

```
Tensor Flow Version: 2.4.0
Keras Version: 2.4.0

Python 3.8.16 (default, Mar  2 2023, 03:18:16) [MSC v.1916 64 bit (AMD64)]
Pandas 1.5.3
Scikit-Learn 1.2.2
```

Convolutional Neural Networks (CNN)

CNN is a neural network technology that has profoundly impacted the area of computer vision (CV). This advantage in CNN is due to years of research on biological eyes. In other words, CNN utilize overlapping fields of input to simulate features of biological eyes. Until this breakthrough, AI had been unable to reproduce the capabilities of biological vision.

There are many datasets for computer vision. Two of the most popular are the MNIST digits dataset and the CIFAR image datasets.

MNIST Digits dataset

The MNIST digits data set is very popular in the neural network research community

MNIST Fashion Dataset

Fashion MNIST is a dataset from Zalando's article images- containing of a training set of 70 000 example and a test set of 10 000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. Fashion-MNIST is intended to serve as a direct drop in replacement for the original MNIST dataset for benchmarking machine learning algorithm.

1. Import the libraries

In [8]:
```python
import tensorflow.keras
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras import regularizers
from tensorflow.keras.datasets import mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
print("Shape of x_train: {}".format(x_train.shape))
print("Shape of y_train: {}".format(y_train.shape))
print()
print("Shape of x_test: {}".format(x_test.shape))
print("Shape of y_test: {}".format(y_test.shape))
```

```
Shape of x_train: (60000, 28, 28)
Shape of y_train: (60000,)

Shape of x_test: (10000, 28, 28)
Shape of y_test: (10000,)
```

2 Display the Digits

In [9]:
```python
from IPython.display import display
import pandas as pd

# Display as text
pd.set_option('display.max_columns', 15)
pd.set_option('display.max_rows', 5)

print("Shape for dataset: {}".format(x_train.shape))
print("Labels: {}".format(y_train))

# Single MNIST digit
single = x_train[0]
print("Shape for single: {}".format(single.shape))

pd.DataFrame(single.reshape(28,28))
```

```
Shape for dataset: (60000, 28, 28)
Labels: [5 0 4 ... 5 6 8]
Shape for single: (28, 28)
```

Out[9]:

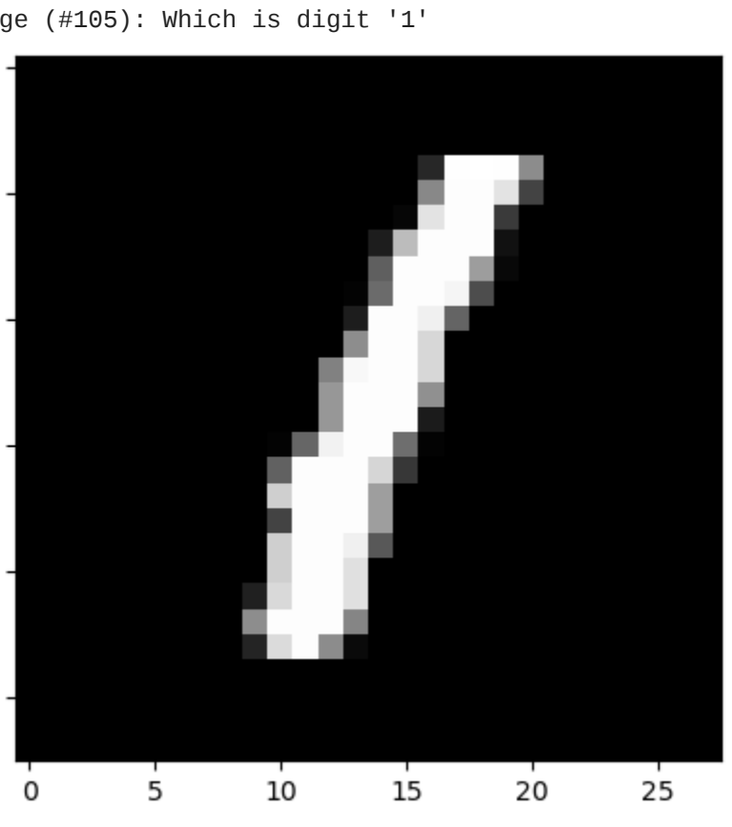
	0	1	2	3	4	5	6	...	21	22	23	24	25	26	27
0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
26	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0

28 rows × 28 columns

3. Lets display as image

In [10]:
```python
# Display as image
import matplotlib.pyplot as plt
import numpy as np
digit = 105 # Change to choose new digit
a= x_train[digit]
plt.imshow(a, cmap='gray', interpolation='nearest')
print("Image (#{}): Which is digit '{}'".format(digit, y_train[digit]))
```

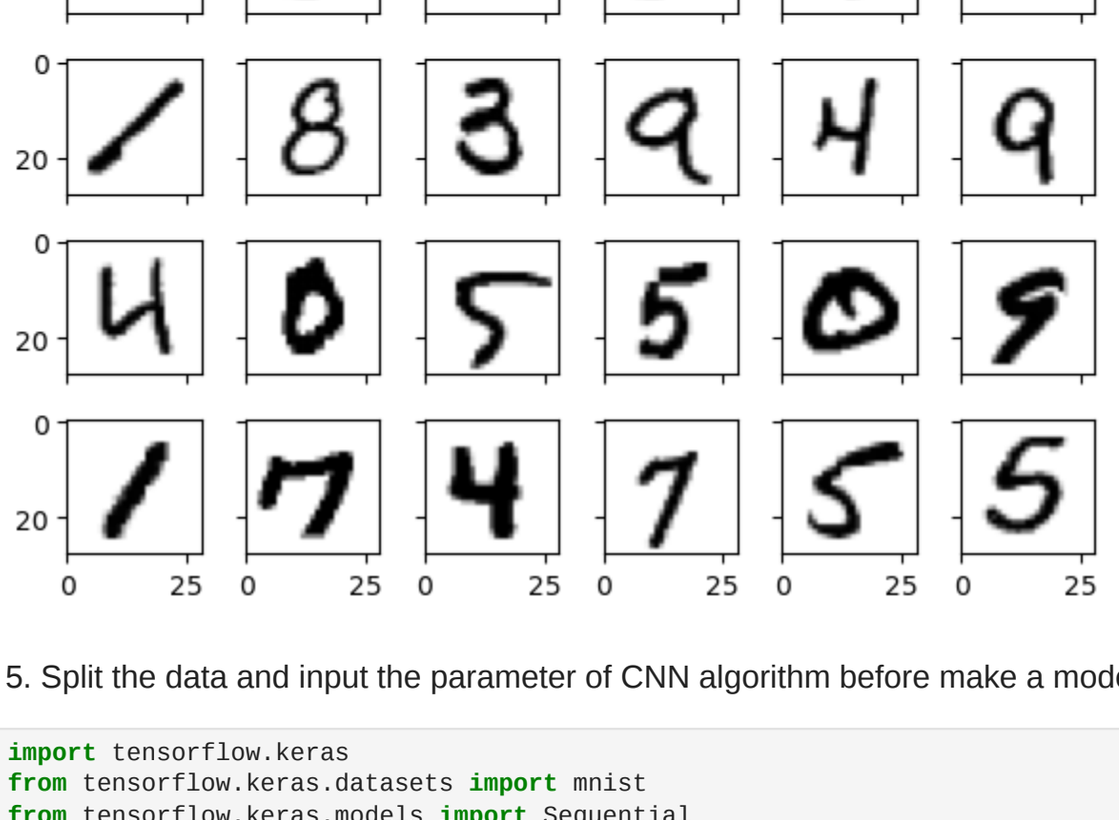
```
Image (#105): Which is digit '1'
```



4. We can show the dataset with random. Using random library to show the figure

In [11]:
```python
import random

ROWS = 6
random_indices = random.sample(range(x_train.shape[0]), ROWS*ROWS)

sample_images = x_train[random_indices, :]

plt.clf()

fig, axes = plt.subplots(ROWS, ROWS,
                         figsize=(ROWS, ROWS),
                         sharex=True, sharey=True)

for i in range (ROWS*ROWS):
    subplot_row = i//ROWS
    subplot_col = i%ROWS
    ax = axes[subplot_row, subplot_col]

    plottable_image = np.reshape(sample_images[i, :], (28,28))
    ax.imshow(plottable_image, cmap='gray_r')

    ax.set_xbound([0,28])

plt.tight_layout()
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```



5. Split the data and input the parameter of CNN algorithm before make a model (train the data)

In [248…
```python
import tensorflow.keras
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras import backend as K
batch_size = 128
num_classes = 10
epochs = 12

img_rows, img_cols = 28, 28
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 = (5, 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 = (5, img_rows, img_cols, 1)
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("Training samples: {}".format(x_train.shape[0]))
print("Test samples: {}".format(x_test.shape[0]))

y_train = tensorflow.keras.utils.to_categorical(y_train, num_classes)
y_test = tensorflow.keras.utils.to_categorical(y_test, num_classes)
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='categorical_crossentropy', optimizer='adam',metrics=['accuracy'])
```

```
x_train shape: (60000, 28, 28, 1)
Training samples: 60000
Test samples: 10000
```

In [245…
```python
model.summary()
```

```
Model: "sequential_33"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_66 (Conv2D)           (None, 5, 26, 26, 32)     320
_________________________________________________________________
conv2d_67 (Conv2D)           (None, 5, 24, 24, 64)     18496
_________________________________________________________________
dropout_44 (Dropout)         (None, 5, 24, 24, 64)     0
_________________________________________________________________
flatten_24 (Flatten)         (None, 184320)            0
_________________________________________________________________
dense_48 (Dense)             (None, 128)               23593088
_________________________________________________________________
dropout_45 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_49 (Dense)             (None, 10)                1290
=================================================================
Total params: 23,613,194
Trainable params: 23,613,194
Non-trainable params: 0
_________________________________________________________________
```

6. Training the CNN – DIGITS datasets. This can take awhile

In [249…
```python
x_train = x_train.reshape(x_train.shape[0], 1, 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 1, 28, 28, 1)
model.compile(optimizer='adam', loss='categorical_crossentropy')
```

In [250…
```python
import tensorflow as tf
import time

start_time = time.time()

model.fit(x_train, y_train,batch_size=batch_size,epochs=epochs,verbose=3,validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss: {}'.format(score[0]))
print('Test accuracy: {}'.format(score[1]))


def hms_string(sec):
    hours = int(sec / 3600)
    minutes = int((sec % 3600) / 60)
    seconds = int(sec % 60)
    return "{:02d}:{:02d}:{:02d}".format(hours, minutes, seconds)

elapsed_time = time.time() - start_time
print("Elapsed time: {}".format(hms_string(elapsed_time)))
```

```
Epoch 1/12
WARNING:tensorflow:Model was constructed with shape (None, 5, 28, 28, 1) for input KerasTensor(type_spec=TensorSpec(shape=(None, 5, 28, 28, 1), dtype=tf.float32, name='conv2d_70_input'), name='conv2d_70_input', description="created by layer 'conv2d_70_input'"), but it was called on an input with incompatible shape (None, 1, 28, 28, 1).
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[250], line 6
      2 import time
      4 start_time = time.time()
----> 6 model.fit(x_train, y_train,batch_size=batch_size,epochs=epochs,verbose=3,validation_data=(x_test, y_test))
      7 score = model.evaluate(x_test, y_test, verbose=0)
      8 print('Test loss: {}'.format(score[0]))

File ~\anaconda3\envs\tensorflowml\lib\site-packages\tensorflow\python\keras\engine\training.py:1100, in Model.fit(self, x, y, batch_size, epochs, verbose, callbacks, validation_split, validation_data, shuffle, class_weight, sample_weight, initial_epoch, steps_per_epoch, validation_steps, validation_batch_size, validation_freq, max_queue_size, workers, use_multiprocessing)
   1093 with trace.Trace(
   1094     'train',
   1095     epoch_num=epoch,
   1096     step_num=step,
   1097     batch_size=batch_size,
   1098     _r=1):
   1099   callbacks.on_train_batch_begin(step)
-> 1100   tmp_logs = self.train_function(iterator)
   1101   if data_handler.should_sync:
   1102     context.async_wait()

File ~\anaconda3\envs\tensorflowml\lib\site-packages\tensorflow\python\eager\def_function.py:828, in Function.__call__(self, *args, **kwds)
    826 tracing_count = self.experimental_get_tracing_count()
    827 with trace.Trace(self._name) as tm:
--> 828   result = self._call(*args, **kwds)
    829   compiler = "xla" if self._experimental_compile else "nonXla"
    830   new_tracing_count = self.experimental_get_tracing_count()

File ~\anaconda3\envs\tensorflowml\lib\site-packages\tensorflow\python\eager\def_function.py:871, in Function._call(self, *args, **kwds)
    868 try:
    869   # This is the first call of __call__, so we have to initialize.
    870   initializers = []
--> 871   self._initialize(args, kwds, add_initializers_to=initializers)
    872 finally:
    873   # At this point we know that the initialization is complete (or less
    874   # interestingly an exception was raised) so we no longer need a lock.
    875   self._lock.release()

File ~\anaconda3\envs\tensorflowml\lib\site-packages\tensorflow\python\eager\def_function.py:725, in Function._initialize(self, args, kwds, add_initializers_to)
    722 self._lifted_initializer_graph = lifted_initializer_graph
    723 self._graph_deleter = FunctionDeleter(self._lifted_initializer_graph)
    724 self._concrete_stateful_fn = (
--> 725     self._stateful_fn._get_concrete_function_internal_garbage_collected(  # pylint: disable=protected-access
    726         *args, **kwds))
    728 def invalid_creator_scope(*unused_args, **unused_kwds):
    729   """Disables variable creation."""

File ~\anaconda3\envs\tensorflowml\lib\site-packages\tensorflow\python\eager\function.py:2969, in Function._get_concrete_function_internal_garbage_collected(self, *args, **kwargs)
   2967   args, kwargs = None, None
   2968 with self._lock:
-> 2969   graph_function, _ = self._maybe_define_function(args, kwargs)
   2970 return graph_function

File ~\anaconda3\envs\tensorflowml\lib\site-packages\tensorflow\python\eager\function.py:3361, in Function._maybe_define_function(self, args, kwargs)
   3357   return self._define_function_with_shape_relaxation(
   3358       args, kwargs, flat_args, filtered_flat_args, cache_key_context)
   3360 self._function_cache.missed.add(call_context_key)
-> 3361 graph_function = self._create_graph_function(args, kwargs)
   3362 self._function_cache.primary[cache_key] = graph_function
   3364 return graph_function, filtered_flat_args

File ~\anaconda3\envs\tensorflowml\lib\site-packages\tensorflow\python\eager\function.py:3196, in Function._create_graph_function(self, args, kwargs, override_flat_arg_shapes)
   3191 missing_arg_names = [
   3192     "%s_%d" % (arg, i) for i, arg in enumerate(missing_arg_names)
   3193 ]
   3194 arg_names = base_arg_names + missing_arg_names
   3195 graph_function = ConcreteFunction(
-> 3196     func_graph_module.func_graph_from_py_func(
   3197         self._name,
   3198         self._python_function,
   3199         args,
   3200         kwargs,
   3201         self.input_signature,
   3202         autograph=self._autograph,
   3203         autograph_options=self._autograph_options,
   3204         arg_names=arg_names,
   3205         override_flat_arg_shapes=override_flat_arg_shapes,
   3206         capture_by_value=self._capture_by_value),
   3207     self._function_attributes,
   3208     function_spec=self.function_spec,
   3209     # Tell the ConcreteFunction to clean up its graph once it goes out of
   3210     # scope. This is not the default behavior since it gets used in some
   3211     # places (like Keras) where the FuncGraph lives longer than the
   3212     # ConcreteFunction.
   3213     shared_func_graph=False)
   3214 return graph_function

File ~\anaconda3\envs\tensorflowml\lib\site-packages\tensorflow\python\framework\func_graph.py:990, in func_graph_from_py_func(name, python_func, args, kwargs, signature, func_graph, autograph, autograph_options, add_control_dependencies, arg_names, op_return_value, collections, capture_by_value, override_flat_arg_shapes)
    987 else:
    988   _, original_func = tf_decorator.unwrap(python_func)
--> 990 func_outputs = python_func(*func_args, **func_kwargs)
    992 # invariant: `func_outputs` contains only Tensors, CompositeTensors,
    993 # TensorArrays and `None`s.
    994 func_outputs = nest.map_structure(convert, func_outputs,
    995                                   expand_composites=True)

File ~\anaconda3\envs\tensorflowml\lib\site-packages\tensorflow\python\eager\def_function.py:634, in Function._defun_with_scope.<locals>.wrapped_fn(*args, **kwds)
    632     xla_context.Exit()
    633 else:
--> 634   out = weak_wrapped_fn().__wrapped__(*args, **kwds)
    635 return out

File ~\anaconda3\envs\tensorflowml\lib\site-packages\tensorflow\python\framework\func_graph.py:977, in func_graph_from_py_func.<locals>.wrapper(*args, **kwargs)
    975 except Exception as e:  # pylint:disable=broad-except
    976   if hasattr(e, "ag_error_metadata"):
--> 977     raise e.ag_error_metadata.to_exception(e)
    978   else:
    979     raise

ValueError: in user code:

    C:\Users\Darren\anaconda3\envs\tensorflowml\lib\site-packages\tensorflow\python\keras\engine\training.py:805 train_function  *
        return step_function(self, iterator)
    C:\Users\Darren\anaconda3\envs\tensorflowml\lib\site-packages\tensorflow\python\keras\engine\training.py:795 step_function  **
        outputs = model.distribute_strategy.run(run_step, args=(data,))
    C:\Users\Darren\anaconda3\envs\tensorflowml\lib\site-packages\tensorflow\python\distribute\distribute_lib.py:1259 run
        return self._extended.call_for_each_replica(fn, args=args, kwargs=kwargs)
    C:\Users\Darren\anaconda3\envs\tensorflowml\lib\site-packages\tensorflow\python\distribute\distribute_lib.py:2730 call_for_each_replica
        return self._call_for_each_replica(fn, args, kwargs)
    C:\Users\Darren\anaconda3\envs\tensorflowml\lib\site-packages\tensorflow\python\distribute\distribute_lib.py:3417 _call_for_each_replica
        return fn(*args, **kwargs)
    C:\Users\Darren\anaconda3\envs\tensorflowml\lib\site-packages\tensorflow\python\keras\engine\training.py:788 run_step  **
        outputs = model.train_step(data)
    C:\Users\Darren\anaconda3\envs\tensorflowml\lib\site-packages\tensorflow\python\keras\engine\training.py:754 train_step
        y_pred = self(x, training=True)
    C:\Users\Darren\anaconda3\envs\tensorflowml\lib\site-packages\tensorflow\python\keras\engine\base_layer.py:1012 __call__
        outputs = call_fn(inputs, *args, **kwargs)
    C:\Users\Darren\anaconda3\envs\tensorflowml\lib\site-packages\tensorflow\python\keras\engine\sequential.py:375 call
        return super(Sequential, self).call(inputs, training=training, mask=mask)
    C:\Users\Darren\anaconda3\envs\tensorflowml\lib\site-packages\tensorflow\python\keras\engine\functional.py:424 call
        return self._run_internal_graph(
    C:\Users\Darren\anaconda3\envs\tensorflowml\lib\site-packages\tensorflow\python\keras\engine\functional.py:560 _run_internal_graph
        outputs = node.layer(*args, **kwargs)
    C:\Users\Darren\anaconda3\envs\tensorflowml\lib\site-packages\tensorflow\python\keras\engine\base_layer.py:998 __call__
        input_spec.assert_input_compatibility(self.input_spec, inputs, self.name)
    C:\Users\Darren\anaconda3\envs\tensorflowml\lib\site-packages\tensorflow\python\keras\engine\input_spec.py:255 assert_input_compatibility
        raise ValueError(

    ValueError: Input 0 of layer dense_50 is incompatible with the layer: expected axis -1 of input shape to have value 184320 but received input with shape (None, 36864)
```

saya sekip modul tutorial karena error

Importing numpy library

In [4]:
```python
import numpy as np
```

---DATASET SENDIRI---

importing dataset

In [230…
```python
# Load the data from the .npz file
x_train = np.load('D:\SEMESTER 4\IF540 Machine Learning\LAB\week8\kmnist-train-imgs.npz')['arr_0']
y_train = np.load('D:\SEMESTER 4\IF540 Machine Learning\LAB\week8\kmnist-train-labels.npz')['arr_0'].reshape(-1, 1)
x_test =  np.load('D:\SEMESTER 4\IF540 Machine Learning\LAB\week8\kmnist-test-imgs.npz')['arr_0']
y_test =  np.load('D:\SEMESTER 4\IF540 Machine Learning\LAB\week8\kmnist-test-labels.npz')['arr_0'].reshape(-1, 1)

print("Shape of x_train: ", x_train.shape)
print("Shape of y_train: ",y_train.shape)
print()
print("Shape of x_test: ", x_test.shape)
print("Shape of y_test: ",y_test.shape)
```

```
Shape of x_train:  (60000, 28, 28)
Shape of y_train:  (60000, 1)

Shape of x_test:  (10000, 28, 28)
Shape of y_test:  (10000, 1)
```

2 Display the Kmnist dataset

In [231…
```python
from IPython.display import display
import pandas as pd

# Display as text
pd.set_option('display.max_columns', 15)
pd.set_option('display.max_rows', 5)

print("Shape for dataset: {}".format(x_train.shape))
print("Labels: {}".format(y_train))

# Single MNIST digit
single = x_train[0]
print("Shape for single: {}".format(single.shape))

pd.DataFrame(single.reshape(28,28))
```

```
Shape for dataset: (60000, 28, 28)
Labels: [[8]
 [7]
 [0]
 ...
 [0]
 [4]
 [9]]
Shape for single: (28, 28)
```

Out[231]:

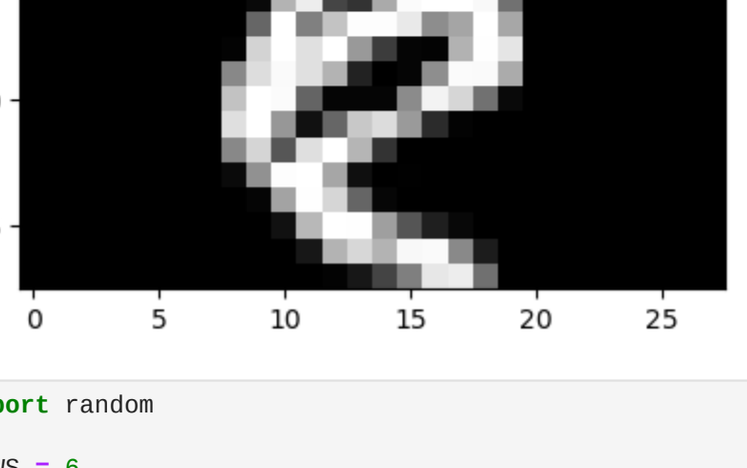
	0	1	2	3	4	5	6	...	21	22	23	24	25	26	27
0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
26	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0

28 rows × 28 columns

3. Lets display as image

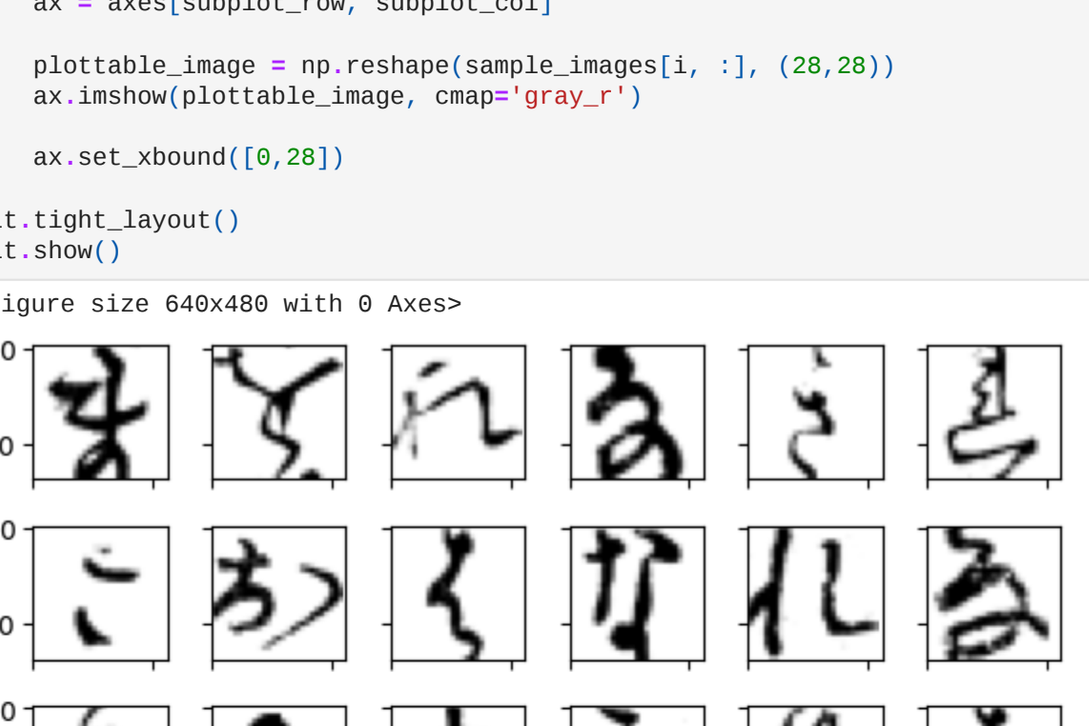
In [232…
```python
# Display as image
import matplotlib.pyplot as plt
import numpy as np
digit = 285 # Change to choose new digit
a= x_train[digit]
plt.imshow(a, cmap='gray', interpolation='nearest')
print("Image (#{}): Which is digit '{}'".format(digit, y_train[digit]))
```

```
Image (#285): Which is digit '[5]'
```



In [233…
```python
import random

ROWS = 6
random_indices = random.sample(range(x_train.shape[0]), ROWS*ROWS)

sample_images = x_train[random_indices, :]

plt.clf()

fig, axes = plt.subplots(ROWS, ROWS,
                         figsize=(ROWS, ROWS),
                         sharex=True, sharey=True)

for i in range (ROWS*ROWS):
    subplot_row = i//ROWS
    subplot_col = i%ROWS
    ax = axes[subplot_row, subplot_col]

    plottable_image = np.reshape(sample_images[i, :], (28,28))
    ax.imshow(plottable_image, cmap='gray_r')

    ax.set_xbound([0,28])

plt.tight_layout()
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```



5. Split the data and input the parameter of CNN algorithm before make a model (train the data)

In [234…
```python
import tensorflow.keras
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras import backend as K

batch_size = 256
num_classes = len(np.unique(y_train))
epochs = 12

img_rows, img_cols = 28, 28
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 = (5, 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 = (5, img_rows, img_cols, 1)
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("Training samples: {}".format(x_train.shape[0]))
print("Test samples: {}".format(x_test.shape[0]))

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

# Define the model architecture
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)))
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='categorical_crossentropy', optimizer='adam',metrics=['accuracy'])
```

```
x_train shape: (60000, 28, 28, 1)
Training samples: 60000
Test samples: 10000
```

In [235…
```python
model.summary()
```

```
Model: "sequential_32"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_64 (Conv2D)           (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_65 (Conv2D)           (None, 24, 24, 64)        18496
_________________________________________________________________
max_pooling2d_29 (MaxPooling (None, 12, 12, 64)        0
_________________________________________________________________
dropout_42 (Dropout)         (None, 12, 12, 64)        0
_________________________________________________________________
flatten_23 (Flatten)         (None, 9216)              0
_________________________________________________________________
dense_46 (Dense)             (None, 128)               1179776
_________________________________________________________________
dropout_43 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_47 (Dense)             (None, 10)                1290
=================================================================
Total params: 1,199,882
Trainable params: 1,199,882
Non-trainable params: 0
_________________________________________________________________
```

6.Training the CNN – KMNITS datasets. This can take awhile

In [236…
```python
import tensorflow as tf
import time

start_time = time.time()

model.fit(x_train, y_train,batch_size=batch_size,epochs=epochs,verbose=3,validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss: {}'.format(score[0]))
print('Test accuracy: {}'.format(score[1]))


def hms_string(sec):
    hours = int(sec / 3600)
    minutes = int((sec % 3600) / 60)
    seconds = int(sec % 60)
    return "{:02d}:{:02d}:{:02d}".format(hours, minutes, seconds)

elapsed_time = time.time() - start_time
print("Elapsed time: {}".format(hms_string(elapsed_time)))
```

```
Epoch 1/12
Epoch 2/12
Epoch 3/12
Epoch 4/12
Epoch 5/12
Epoch 6/12
Epoch 7/12
Epoch 8/12
Epoch 9/12
Epoch 10/12
Epoch 11/12
Epoch 12/12
Test loss: 0.2055513709783554
Test accuracy: 0.9513000249862671
Elapsed time: 00:07:19
```

7. Evaluate accuracy from KMNITS dataset

In [237…
```python
# set the desired Tensorflow output level
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss: {}'.format(score[0]))
print('Test accuracy: {}'.format(score[1]))
```

```
Test loss: 0.2055513709783554
Test accuracy: 0.9513000249862671
```

In [238…
```python
from sklearn import metrics

small_x = x_test[1:100]
small_y = y_test[1:100]
small_y2 = np.argmax(small_y,axis=1)
pred = model.predict(small_x)
pred = np.argmax(pred,axis=1)
score = metrics.accuracy_score(small_y2, pred)
print('Accuracy: {}'.format(score))
```

```
Accuracy: 0.9494949494949495
```

Kesimpulan

Berikan simpulan yang dilakukan dari hasil kerja menggunakan algoritma dan 2 dataset yang dipilih. Simpulan bisa berkisar antara (bisa di modifikasi):

```
- CNN intinya buat klasifikasi gambar
- hasil akurasi dari data KMNIST adalah sebesar 94.9% sedangkan untuk test loss adalah 20%
- kita bisa melihat hasil evaluasi model dengan cara 'model.summary()' hal ini dilakukan supaya bisa melihat hasil evaluasi yang sudah dibuat oleh peneliti.
- Training data dengan waktu epoch di set pada 12 pada device saya membutuhkan waktu Elapsed time: 00:07:19,*perlu diingat hal ini menggunakan CPU dependent
yakni kekuatan cpu penting disini. dan ketika saya mencoba epoch lebih tinggi misal saja 20 epoch maka waktu untuk prosesnya akan semakin lama, sehingga saya
tetap menggunakan rekomendasi modul yakni 12 ephocs saja.*
```

In [251…
```python
# Footer
myDate = datetime.datetime.now()
print("I certify that this is my own work.")
print("Signed by:")
print("Name: \t\t{}".format(myName))
print("NIM: \t\t{}".format(myNIM))
print("Time-stamp:\t{}".format(myDate))
```

```
I certify that this is my own work.
Signed by:
Name:           Christopher Darren
NIM:            00000054804
Time-stamp:     2023-04-19 21:23:11.614866
```

Save the notebook, then convert the notebook to html (by running the next code).

In [ ]:
```python
!jupyter nbconvert --to html "./IF540_Kelas EL_00000054804_Christopher Darren_Week08.ipynb" --output-dir="./"
```

In [ ]: