Why deep learning becoming more so popular?

1. Data Growth-deep learning works better if volumne of data is high
2. Hardware Advancements - computer hardwares have advanced much GPU AND TPU
3. Python and open source ecosysytem
4. Cloud and AI boom

Deep learning is a subfield of machine learning that is inspired by the structure and function of the brain, and is concerned with algorithms and models that can learn representations of data with multiple levels of abstraction.

Deep learning models are typically composed of multiple layers of artificial neural networks, each layer building on the outputs of the previous layer to learn more complex features and representations. The "deep" in deep learning refers to the fact that these models often have many layers, allowing them to learn highly complex and abstract representations of the data.

Deep learning has been successfully applied to a wide range of tasks, including image and speech recognition, natural language processing, and even game-playing. It has also shown promising results in areas such as drug discovery, personalized medicine, and autonomous vehicles.

# The Linear Unit

y=x*w+b(x is the feature ,w is the weight and b is the bias)

in a datset if u have a single feature then the feature gets multiplied with the weight and gets added withn bias and volla u will get the result The equation y=x*w+b ------------- is similar with the straight line equationy=m*x+c

example:- Let's think about how this might work on a dataset like 80 Cereals. Training a model with 'sugars' (grams of sugars per serving) as input and 'calories' (calories per serving) as output, we might find the bias is b=90 and the weight is w=2.5. We could estimate the calorie content of a cereal with 5 grams of sugar per serving like this:

And, checking against our formula, we have calories=2.5×5+90=102.5 , just like we expect

## Multiple Inputs

The 80 Cereals dataset has many more features than just 'sugars'. What if we wanted to expand our model to include things like fiber or protein content? That's easy enough. We can just add more input connections to the neuron, one for each additional feature. To find the output, we would multiply each input to its connection weight and then add them all together. The formula for this neuron would be y=w0x0+w1x1+w2x2+b . A linear unit with two inputs will fit a plane, and a unit with more inputs than that will fit a hyperplane.

```
In [1]: pip install tensorflow
```

```
Requirement already satisfied: tensorflow in c:\users\shruthy\anaconda3\lib\site-packages (2.12.0)
Requirement already satisfied: tensorflow-intel==2.12.0 in c:\users\shruthy\anaconda3\lib\site-packages (from tensorflow) (2.12.
0)
Requirement already satisfied: six>=1.12.0 in c:\users\shruthy\anaconda3\lib\site-packages (from tensorflow-intel==2.12.0->tenso
rflow) (1.16.0)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in c:\users\shruthy\anaconda3\lib\site-packages (from tensorflow-intel==2.12.
0->tensorflow) (1.14.1)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\shruthy\anaconda3\lib\site-packages (from tensorflow-intel==2.12.0-
>tensorflow) (1.6.3)
Requirement already satisfied: jax>=0.3.15 in c:\users\shruthy\anaconda3\lib\site-packages (from tensorflow-intel==2.12.0->tenso
rflow) (0.4.10)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in c:\users\shru
thy\anaconda3\lib\site-packages (from tensorflow-intel==2.12.0->tensorflow) (3.20.3)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in c:\users\shruthy\anaconda3\lib\site-packages (from tensor
flow-intel==2.12.0->tensorflow) (0.31.0)
Requirement already satisfied: setuptools in c:\users\shruthy\anaconda3\lib\site-packages (from tensorflow-intel==2.12.0->tensor
flow) (63.4.1)
Requirement already satisfied: keras<2.13,>=2.12.0 in c:\users\shruthy\anaconda3\lib\site-packages (from tensorflow-intel==2.12.
0->tensorflow) (2.12.0)
Requirement already satisfied: flatbuffers>=2.0 in c:\users\shruthy\anaconda3\lib\site-packages (from tensorflow-intel==2.12.0->
tensorflow) (23.5.9)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\shruthy\anaconda3\lib\site-packages (from tensorflow-intel==2.12.
0->tensorflow) (1.54.2)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in c:\users\shruthy\anaconda3\lib\site-packages (from tensorflow-intel==2.12.
0->tensorflow) (0.4.0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\shruthy\anaconda3\lib\site-packages (from tensorflow-intel==2.12.0->
tensorflow) (16.0.0)
Requirement already satisfied: packaging in c:\users\shruthy\anaconda3\lib\site-packages (from tensorflow-intel==2.12.0->tensorf
low) (21.3)
Requirement already satisfied: h5py>=2.9.0 in c:\users\shruthy\anaconda3\lib\site-packages (from tensorflow-intel==2.12.0->tenso
rflow) (3.7.0)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\shruthy\anaconda3\lib\site-packages (from tensorflow-intel==2.12.0-
>tensorflow) (3.3.0)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\shruthy\anaconda3\lib\site-packages (from tensorflow-intel==2.12.
0->tensorflow) (0.2.0)
Requirement already satisfied: tensorflow-estimator<2.13,>=2.12.0 in c:\users\shruthy\anaconda3\lib\site-packages (from tensorfl
ow-intel==2.12.0->tensorflow) (2.12.0)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\shruthy\anaconda3\lib\site-packages (from tensorflow-intel==2.12.0->te
nsorflow) (1.4.0)
Requirement already satisfied: numpy<1.24,>=1.22 in c:\users\shruthy\anaconda3\lib\site-packages (from tensorflow-intel==2.12.0-
>tensorflow) (1.23.5)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\shruthy\anaconda3\lib\site-packages (from tensorflow-intel==2.12.0->
tensorflow) (2.3.0)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\shruthy\anaconda3\lib\site-packages (from tensorflow-intel==
2.12.0->tensorflow) (4.3.0)
Requirement already satisfied: tensorboard<2.13,>=2.12 in c:\users\shruthy\anaconda3\lib\site-packages (from tensorflow-intel==
2.12.0->tensorflow) (2.12.3)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\shruthy\anaconda3\lib\site-packages (from astunparse>=1.6.0->tenso
rflow-intel==2.12.0->tensorflow) (0.37.1)
Requirement already satisfied: ml-dtypes>=0.1.0 in c:\users\shruthy\anaconda3\lib\site-packages (from jax>=0.3.15->tensorflow-in
tel==2.12.0->tensorflow) (0.1.0)
Requirement already satisfied: scipy>=1.7 in c:\users\shruthy\anaconda3\lib\site-packages (from jax>=0.3.15->tensorflow-intel==
2.12.0->tensorflow) (1.9.1)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\shruthy\anaconda3\lib\site-packages (from tensorboard<2.13,>=2.12->te
nsorflow-intel==2.12.0->tensorflow) (2.0.3)
Requirement already satisfied: google-auth<3,>=1.6.3 in c:\users\shruthy\anaconda3\lib\site-packages (from tensorboard<2.13,>=2.
12->tensorflow-intel==2.12.0->tensorflow) (2.18.0)
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in c:\users\shruthy\anaconda3\lib\site-packages (from tensorboard<
2.13,>=2.12->tensorflow-intel==2.12.0->tensorflow) (1.0.0)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\shruthy\anaconda3\lib\site-packages (from tensorboard<2.13,>=2.12
->tensorflow-intel==2.12.0->tensorflow) (2.28.1)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in c:\users\shruthy\anaconda3\lib\site-packages (from tenso
rboard<2.13,>=2.12->tensorflow-intel==2.12.0->tensorflow) (0.7.0)
Requirement already satisfied: markdown>=2.6.8 in c:\users\shruthy\anaconda3\lib\site-packages (from tensorboard<2.13,>=2.12->te
nsorflow-intel==2.12.0->tensorflow) (3.3.4)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\shruthy\anaconda3\lib\site-packages (from packaging->tensorf
low-intel==2.12.0->tensorflow) (3.0.9)
Requirement already satisfied: rsa<5,>=3.1.4 in c:\users\shruthy\anaconda3\lib\site-packages (from google-auth<3,>=1.6.3->tensor
board<2.13,>=2.12->tensorflow-intel==2.12.0->tensorflow) (4.9)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in c:\users\shruthy\anaconda3\lib\site-packages (from google-auth<3,>=1.6.
3->tensorboard<2.13,>=2.12->tensorflow-intel==2.12.0->tensorflow) (5.3.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\shruthy\anaconda3\lib\site-packages (from google-auth<3,>=1.6.3
->tensorboard<2.13,>=2.12->tensorflow-intel==2.12.0->tensorflow) (0.2.8)
Requirement already satisfied: urllib3<2.0 in c:\users\shruthy\anaconda3\lib\site-packages (from google-auth<3,>=1.6.3->tensorbo
ard<2.13,>=2.12->tensorflow-intel==2.12.0->tensorflow) (1.26.11)
Requirement already satisfied: requests-oauthlib>=0.7.0 in c:\users\shruthy\anaconda3\lib\site-packages (from google-auth-oauthl
ib<1.1,>=0.5->tensorboard<2.13,>=2.12->tensorflow-intel==2.12.0->tensorflow) (1.3.1)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\shruthy\anaconda3\lib\site-packages (from requests<3,>=2.21.0->ten
sorboard<2.13,>=2.12->tensorflow-intel==2.12.0->tensorflow) (2022.9.14)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\shruthy\anaconda3\lib\site-packages (from requests<3,>=2.21.
0->tensorboard<2.13,>=2.12->tensorflow-intel==2.12.0->tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\shruthy\anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorboa
rd<2.13,>=2.12->tensorflow-intel==2.12.0->tensorflow) (3.3)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\users\shruthy\anaconda3\lib\site-packages (from pyasn1-modules>=0.2.1-
>google-auth<3,>=1.6.3->tensorboard<2.13,>=2.12->tensorflow-intel==2.12.0->tensorflow) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\shruthy\anaconda3\lib\site-packages (from requests-oauthlib>=0.7.0->g
oogle-auth-oauthlib<1.1,>=0.5->tensorboard<2.13,>=2.12->tensorflow-intel==2.12.0->tensorflow) (3.2.2)
Note: you may need to restart the kernel to use updated packages.
```

```
[notice] A new release of pip is available: 23.0.1 -> 23.1.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [41]:
```
import tensorflow
import tensorflow as tf
```

In [42]:
```
from tensorflow import keras
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
```

In [6]:
```
(X_train,y_train),(X_test,y_test)=keras.datasets.mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 1s 0us/step
```

In [7]:
```
len(X_train)
```

Out[7]:
```
60000
```

In [8]:
```
len(X_test)
```

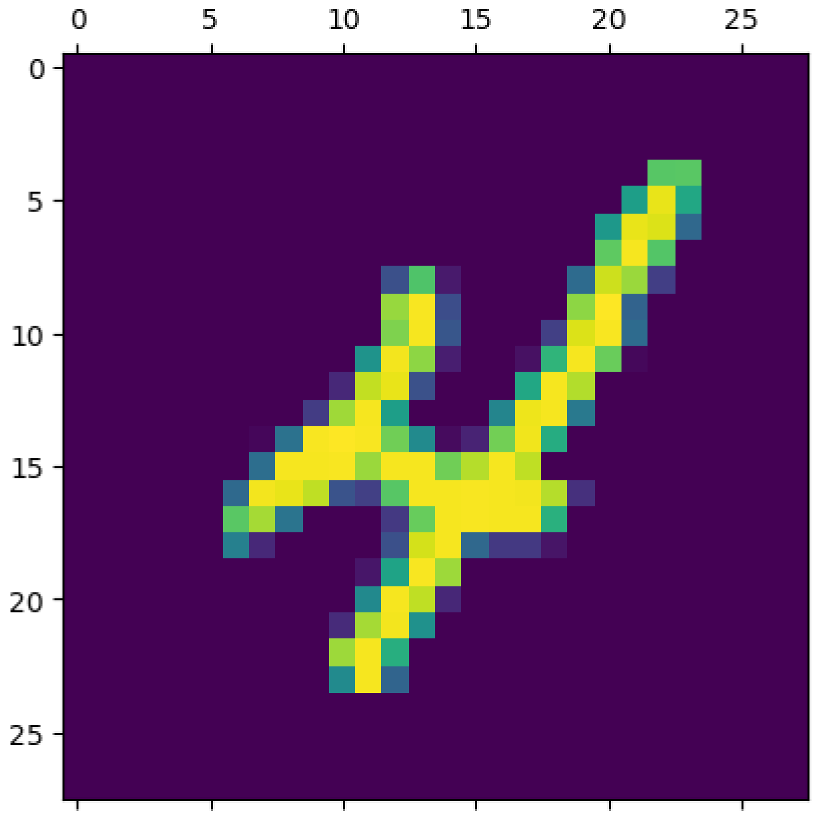
Out[8]:
```
10000
```

In [10]:
```
X_train[0].shape
```

Out[10]:
```
(28, 28)
```

In [12]:
```
plt.matshow(X_train[9])
```

Out[12]:
```
<matplotlib.image.AxesImage at 0x1fe5452b940>
```



```
In [18]: X_train
```

```
Out[18]: array([[[0, 0, 0, ..., 0, 0, 0],
                 [0, 0, 0, ..., 0, 0, 0],
                 [0, 0, 0, ..., 0, 0, 0],
                 ...,
                 [0, 0, 0, ..., 0, 0, 0],
                 [0, 0, 0, ..., 0, 0, 0],
                 [0, 0, 0, ..., 0, 0, 0]],

                [[0, 0, 0, ..., 0, 0, 0],
                 [0, 0, 0, ..., 0, 0, 0],
                 [0, 0, 0, ..., 0, 0, 0],
                 ...,
                 [0, 0, 0, ..., 0, 0, 0],
                 [0, 0, 0, ..., 0, 0, 0],
                 [0, 0, 0, ..., 0, 0, 0]],

                [[0, 0, 0, ..., 0, 0, 0],
                 [0, 0, 0, ..., 0, 0, 0],
                 [0, 0, 0, ..., 0, 0, 0],
                 ...,
                 [0, 0, 0, ..., 0, 0, 0],
                 [0, 0, 0, ..., 0, 0, 0],
                 [0, 0, 0, ..., 0, 0, 0]],

                ...,

                [[0, 0, 0, ..., 0, 0, 0],
                 [0, 0, 0, ..., 0, 0, 0],
                 [0, 0, 0, ..., 0, 0, 0],
                 ...,
                 [0, 0, 0, ..., 0, 0, 0],
                 [0, 0, 0, ..., 0, 0, 0],
                 [0, 0, 0, ..., 0, 0, 0]],

                [[0, 0, 0, ..., 0, 0, 0],
                 [0, 0, 0, ..., 0, 0, 0],
                 [0, 0, 0, ..., 0, 0, 0],
                 ...,
                 [0, 0, 0, ..., 0, 0, 0],
                 [0, 0, 0, ..., 0, 0, 0],
                 [0, 0, 0, ..., 0, 0, 0]],

                [[0, 0, 0, ..., 0, 0, 0],
                 [0, 0, 0, ..., 0, 0, 0],
                 [0, 0, 0, ..., 0, 0, 0],
                 ...,
                 [0, 0, 0, ..., 0, 0, 0],
                 [0, 0, 0, ..., 0, 0, 0],
                 [0, 0, 0, ..., 0, 0, 0]]], dtype=uint8)
```

In [24]:
```
X_train=X_train/255
X_test=X_test/255
```

scaling inc the accuracy of a model

In [25]:
```
#flatten our training data
X_train_flatten=X_train.reshape(len(X_train),28*28)
X_train_flatten.shape
```

Out[25]:
```
(60000, 784)
```

In [26]:
```
X_test_flatten=X_test.reshape(len(X_test),28*28)
X_test_flatten.shape
```

Out[26]:
```
(10000, 784)
```

In [27]:
```
X_train_flatten
```

Out[27]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

In [28]:
```
model=keras.Sequential([
    keras.layers.Dense(10,input_shape=(784,),activation='sigmoid')
])
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
             )
model.fit(X_train_flatten,y_train,epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 2s 842us/step - loss: 0.4663 - accuracy: 0.8786
Epoch 2/5
1875/1875 [==============================] - 2s 826us/step - loss: 0.3042 - accuracy: 0.9149
Epoch 3/5
1875/1875 [==============================] - 2s 840us/step - loss: 0.2832 - accuracy: 0.9212
Epoch 4/5
1875/1875 [==============================] - 2s 898us/step - loss: 0.2731 - accuracy: 0.9238
Epoch 5/5
1875/1875 [==============================] - 1s 788us/step - loss: 0.2667 - accuracy: 0.9255
Out[28]: <keras.callbacks.History at 0x1fe55961760>
```

Sequential means which creates a neural network as a stack of layers.

optimizer allows you to train effecentially

The 'sparse' part of the name refers to the fact that the class labels are integers, rather than one-hot encoded vectors. This means that the label for each data point is a single integer, which indicates the class that the data point belongs to.

The 'categorical' part of the name refers to the fact that the model is performing a classification task with multiple categories.

The 'crossentropy' part of the name refers to the use of the cross-entropy loss function, which is commonly used for classification problems.

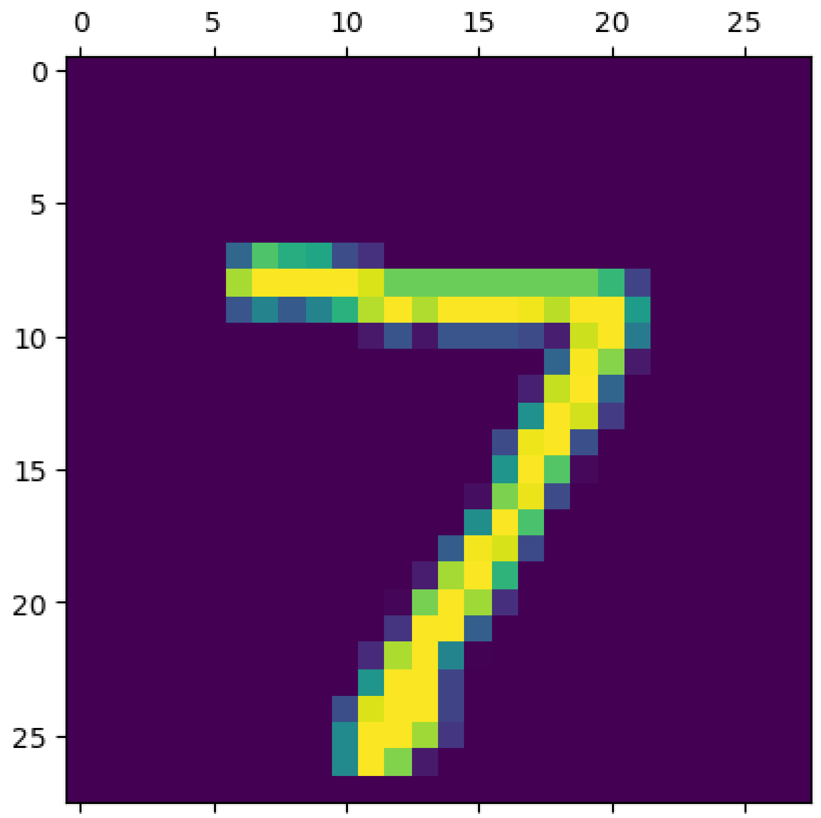
An epoch is one iteration of the training process, where the model updates its parameters based on the errors it makes on the training data.

```
In [29]: model.evaluate(X_test_flatten,y_test)
```

```
313/313 [==============================] - 0s 1ms/step - loss: 0.2708 - accuracy: 0.9236
Out[29]: [0.27076292037963867, 0.9236000180244446]
```

```
In [36]: plt.matshow(X_test[0])
```

```
Out[36]: <matplotlib.image.AxesImage at 0x1fe50fef100>
```



```
In [33]: y_pred=model.predict(X_test_flatten)
         y_pred[0]
```

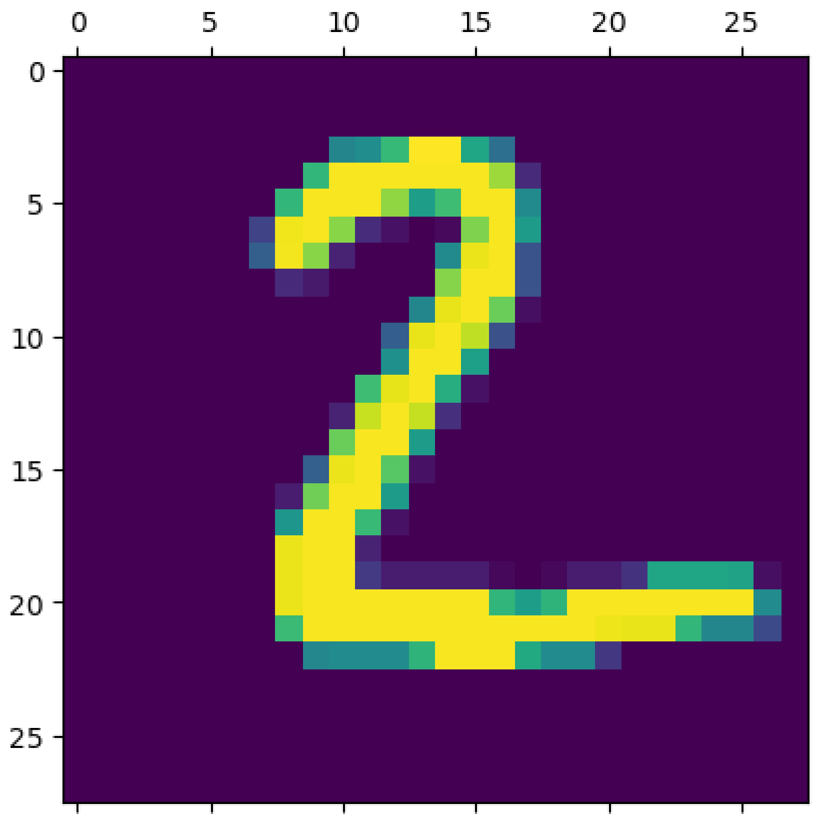
```
313/313 [==============================] - 0s 682us/step
Out[33]: array([1.9997513e-02, 4.0298613e-07, 4.4423062e-02, 9.5133698e-01,
                3.3678527e-03, 1.3316876e-01, 8.7793364e-07, 9.9979097e-01,
                9.2596143e-02, 6.2394232e-01], dtype=float32)
```

```
In [37]: np.argmax(y_pred[0])
```

```
Out[37]: 7
```

```
In [38]: plt.matshow(X_test[1])
```

```
Out[38]: <matplotlib.image.AxesImage at 0x1fe50db4100>
```



In [39]:
```
y_pred=model.predict(X_test_flatten)
y_pred[1]
```

```
313/313 [==============================] - 0s 613us/step
```

Out[39]:
```
array([3.4900558e-01, 5.4057436e-03, 9.9945676e-01, 3.7030986e-01,
       9.8652209e-10, 9.1583520e-01, 9.0523803e-01, 8.4348587e-13,
       2.4223754e-01, 2.4915048e-09], dtype=float32)
```

In [40]:
```
np.argmax(y_pred[1])
```

Out[40]:
```
2
```

In [58]:
```
y_pred_labels=[np.argmax(i) for i in y_pred]
y_pred_labels[:6]
```

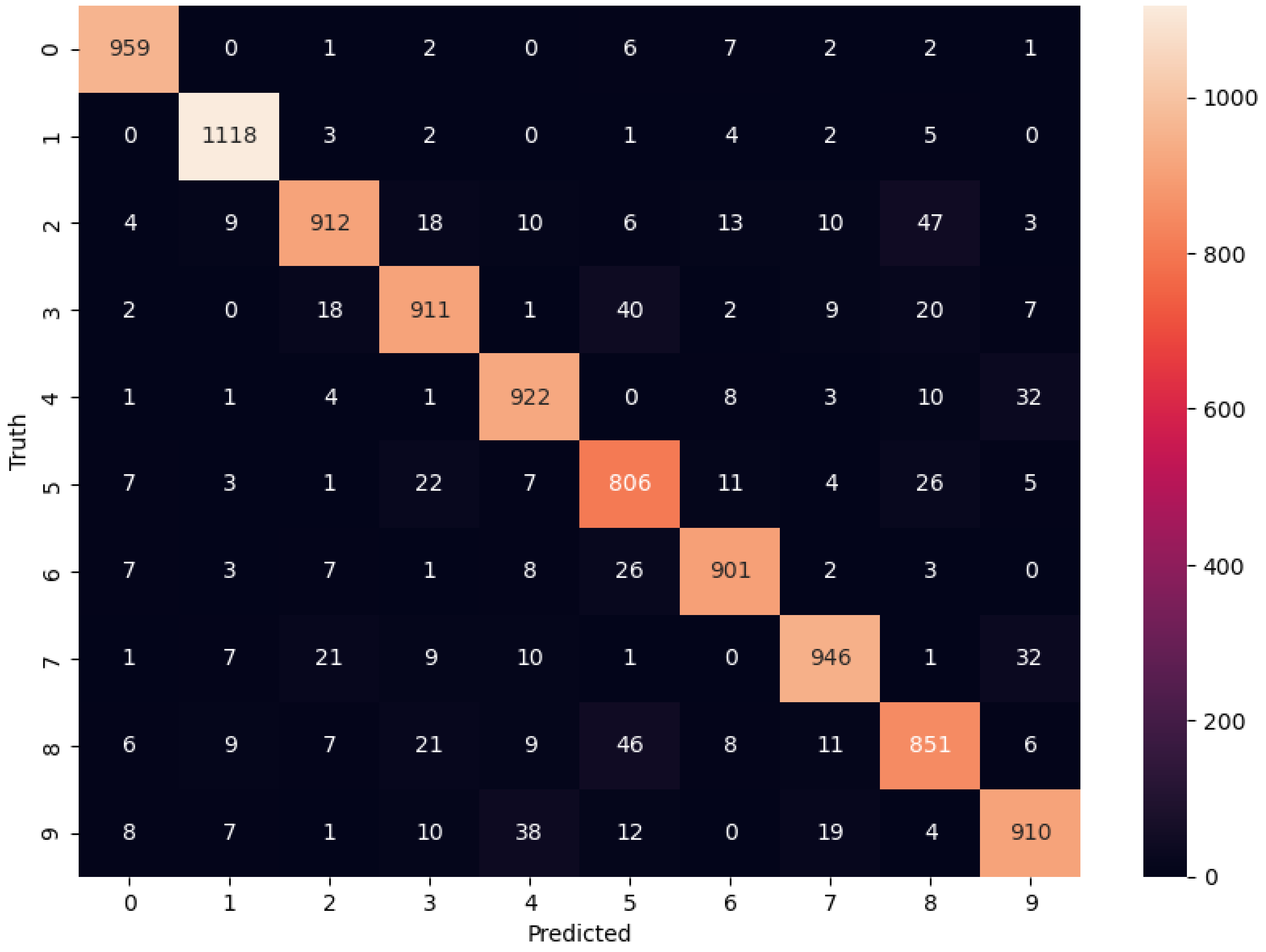
Out[58]:
```
[7, 2, 1, 0, 4, 1]
```

In [46]:
```
cm=tf.math.confusion_matrix(labels=y_test,predictions=y_pred_labels)
cm
```

Out[46]:
```
<tf.Tensor: shape=(10, 10), dtype=int32, numpy=
array([[ 959,    0,    1,    2,    0,    6,    7,    2,    2,    1],
       [   0, 1118,    3,    2,    0,    1,    4,    2,    5,    0],
       [   4,    9,  912,   18,   10,    6,   13,   10,   47,    3],
       [   2,    0,   18,  911,    1,   40,    2,    9,   20,    7],
       [   1,    1,    4,    1,  922,    0,    8,    3,   10,   32],
       [   7,    3,    1,   22,    7,  806,   11,    4,   26,    5],
       [   7,    3,    7,    1,    8,   26,  901,    2,    3,    0],
       [   1,    7,   21,    9,   10,    1,    0,  946,    1,   32],
       [   6,    9,    7,   21,    9,   46,    8,   11,  851,    6],
       [   8,    7,    1,   10,   38,   12,    0,   19,    4,  910]])>
```

In [47]:
```
import seaborn as sns
plt.figure(figsize=(10,7))
sns.heatmap(cm,annot=True,fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Out[47]:
```
Text(95.72222222222221, 0.5, 'Truth')
```



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 959 | 0 | 1 | 2 | 0 | 6 | 7 | 2 | 2 | 1 |
| 1 | 0 | 1118 | 3 | 2 | 0 | 1 | 4 | 2 | 5 | 0 |
| 2 | 4 | 9 | 912 | 18 | 10 | 6 | 13 | 10 | 47 | 3 |
| 3 | 2 | 0 | 18 | 911 | 1 | 40 | 2 | 9 | 20 | 7 |
| 4 | 1 | 1 | 4 | 1 | 922 | 0 | 8 | 3 | 10 | 32 |
| 5 | 7 | 3 | 1 | 22 | 7 | 806 | 11 | 4 | 26 | 5 |
| 6 | 7 | 3 | 7 | 1 | 8 | 26 | 901 | 2 | 3 | 0 |
| 7 | 1 | 7 | 21 | 9 | 10 | 1 | 0 | 946 | 1 | 32 |
| 8 | 6 | 9 | 7 | 21 | 9 | 46 | 8 | 11 | 851 | 6 |
| 9 | 8 | 7 | 1 | 10 | 38 | 12 | 0 | 19 | 4 | 910 |

## Introducing hidden layer

In [54]:
```
model=keras.Sequential([
    keras.layers.Dense(100,input_shape=(784,),activation='relu'),
    keras.layers.Dense(10,activation='sigmoid')
])
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
             )
model.fit(X_train_flatten,y_train,epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 3s 1ms/step - loss: 0.2775 - accuracy: 0.9215
Epoch 2/5
1875/1875 [==============================] - 2s 1ms/step - loss: 0.1261 - accuracy: 0.9627
Epoch 3/5
1875/1875 [==============================] - 2s 1ms/step - loss: 0.0873 - accuracy: 0.9737
Epoch 4/5
1875/1875 [==============================] - 2s 1ms/step - loss: 0.0669 - accuracy: 0.9798
Epoch 5/5
1875/1875 [==============================] - 3s 1ms/step - loss: 0.0535 - accuracy: 0.9839
```

Out[54]:
```
<keras.callbacks.History at 0x1fe006002b0>
```

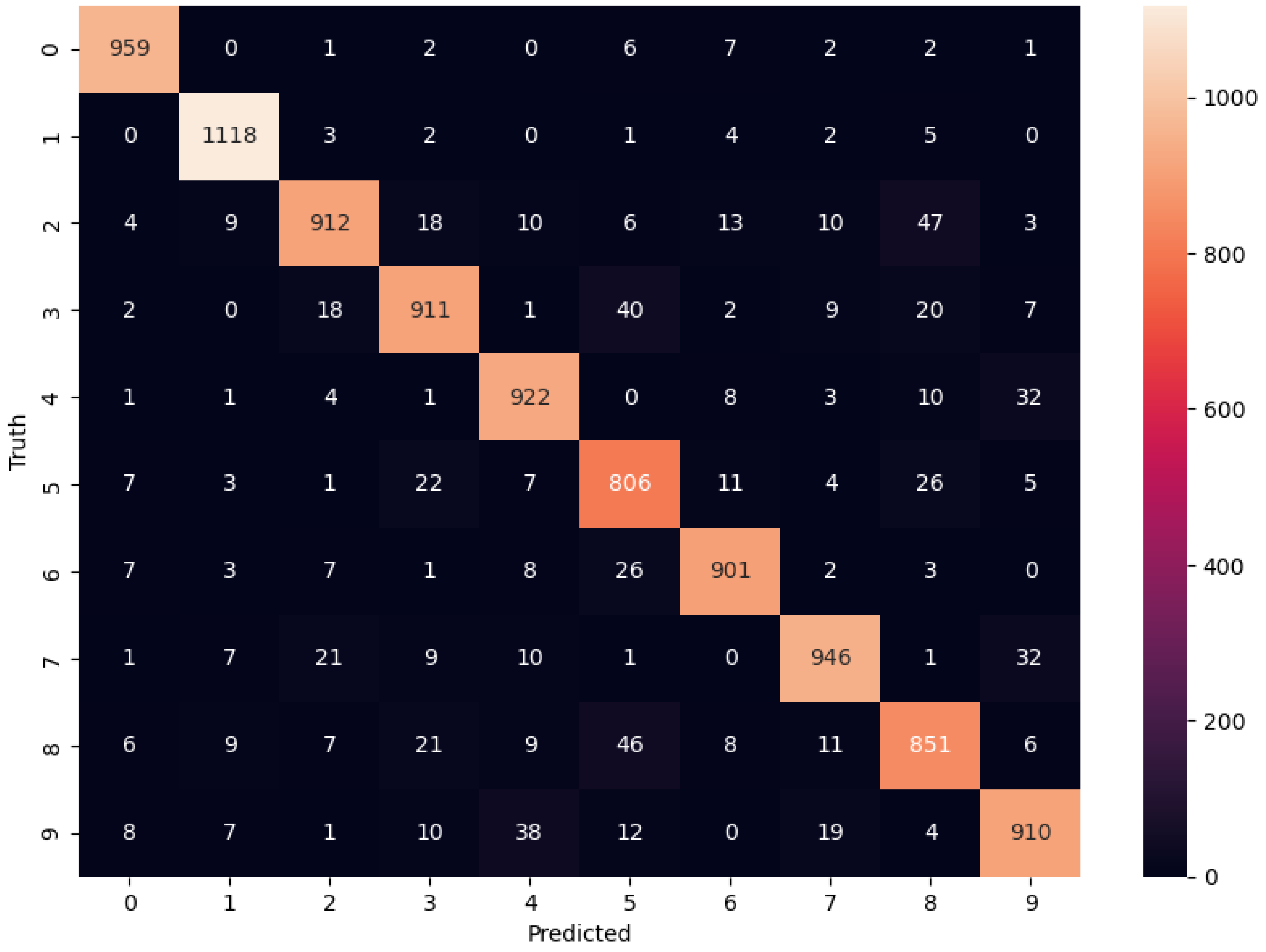
In [55]:
```
model.evaluate(X_test_flatten,y_test)
```

```
313/313 [==============================] - 0s 805us/step - loss: 0.0821 - accuracy: 0.9753
```

Out[55]:
```
[0.0820598229765892, 0.9753000140190125]
```

In [56]:
```
y_pred_labels=[np.argmax(i) for i in y_pred]
cm=tf.math.confusion_matrix(labels=y_test,predictions=y_pred_labels)
import seaborn as sns
plt.figure(figsize=(10,7))
sns.heatmap(cm,annot=True,fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Out[56]:
```
Text(95.72222222222221, 0.5, 'Truth')
```



| Truth \ Predicted | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 959 | 0 | 1 | 2 | 0 | 6 | 7 | 2 | 2 | 1 |
| 1 | 0 | 1118 | 3 | 2 | 0 | 1 | 4 | 2 | 5 | 0 |
| 2 | 4 | 9 | 912 | 18 | 10 | 6 | 13 | 10 | 47 | 3 |
| 3 | 2 | 0 | 18 | 911 | 1 | 40 | 2 | 9 | 20 | 7 |
| 4 | 1 | 1 | 4 | 1 | 922 | 0 | 8 | 3 | 10 | 32 |
| 5 | 7 | 3 | 1 | 22 | 7 | 806 | 11 | 4 | 26 | 5 |
| 6 | 7 | 3 | 7 | 1 | 8 | 26 | 901 | 2 | 3 | 0 |
| 7 | 1 | 7 | 21 | 9 | 10 | 1 | 0 | 946 | 1 | 32 |
| 8 | 6 | 9 | 7 | 21 | 9 | 46 | 8 | 11 | 851 | 6 |
| 9 | 8 | 7 | 1 | 10 | 38 | 12 | 0 | 19 | 4 | 910 |

```
In [ ]:
```