III [].	## Name: Suraj jauhari ## Roll no: I4268 ## Subject:LP-IV(DL)
In []: In [1]:	!pip install tensorflow #importing necessary libraries
	<pre>import tensorflow as tf from tensorflow import keras import pandas as pd import numpy as np import matplotlib.pyplot as plt</pre>
In [2]:	<pre>import random %matplotlib inline</pre>
111 [2].	<pre>#import dataset and split into train and test data mnist = tf.keras.datasets.mnist (x_train, y_train), (x_test, y_test) = mnist.load_data()</pre>
<pre>In [3]: Out[3]:</pre>	<pre>plt.matshow(x_train[1]) <matplotlib.image.axesimage 0x216d2a9c9d0="" at=""></matplotlib.image.axesimage></pre>
	0 5 10 15 20 25
	10 - 15 -
	20 -
In [4]:	plt.imshow(-x_train[0], cmap="gray")
Out[4]:	<pre><matplotlib.image.axesimage 0x216d321cac0="" at=""> 0 -</matplotlib.image.axesimage></pre>
	10 -
	15 - 20 - 20 - 20 - 20 - 20 - 20 - 20 - 2
	25 -
In [5]:	<pre>x_train = x_train / 255 x_test = x_test / 255</pre>
In [6]:	<pre>model = keras.Sequential([keras.layers.Flatten(input_shape=(28, 28)), keras.layers.Dense(128, activation="relu"), keras.layers.Dense(10, activation="softmax")</pre>
	<pre>model.summary()</pre>
	Model: "sequential" Layer (type)
	dense (Dense) (None, 128) 100480 dense_1 (Dense) (None, 10) 1290
	Total params: 101,770 Trainable params: 101,770 Non-trainable params: 0
In [7]:	<pre>model.compile(optimizer="sgd", loss="sparse_categorical_crossentropy", metrics=['accuracy'])</pre>
In [8]:	<pre>history=model.fit(x_train, y_train,validation_data=(x_test,y_test),epochs=10)</pre>
	Epoch 1/10 1875/1875 [====================================
	Epoch 3/10 1875/1875 [====================================
	1875/1875 [====================================
	Epoch 8/10 1875/1875 [====================================
In [9]:	1875/1875 [====================================
	313/313 [===================================
In [10]:	<pre>n=random.randint(0,9999) plt.imshow(x_test[n]) plt.show()</pre>
	0 - 5 -
	10 -
	20 -
In [11]:	0 5 10 15 20 25 x_train
	<pre>array([[[0., 0., 0.,, 0., 0., 0.],</pre>
	[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.]]])
<pre>In [12]: Out[12]:</pre>	x_test 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.], [1., 0., 0.,, 0., 0.], [1., 0., 0.,, 0., 0.], [1., 0., 0.,, 0., 0.], [1., 0., 0.,, 0., 0.], [1., 0., 0.,, 0., 0.], [1., 0., 0., 0., 0.], [1., 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.],
In [13]:	[0., 0., 0.,, 0., 0.], [0., 0., 0.,, 0., 0.]]) predicted_value=model.predict(x_test)
	<pre>plt.imshow(x_test[n]) plt.show() print(predicted_value[n])</pre>
	313/313 [===================================
	10 -
	20 -
	0 5 10 15 20 25 [5.5406590e-06 9.9045217e-01 9.6445804e-04 1.1881288e-03 5.5083026e-05 1.0767796e-03 5.1782565e-04 5.0224096e-04 4.8863539e-03 3.5130000e-04]
In [14]:	<pre># history.history() history.history.keys() # dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])</pre>
	<pre>plt.plot(history.history['accuracy']) plt.plot(history.history['val_accuracy']) plt.title('model accuracy') plt.ylabel('accuracy')</pre>
	<pre>plt.xlabel('epoch') plt.legend(['Train', 'Validation'], loc='upper left') plt.show()</pre>
	model accuracy Train Validation
	0.92 - \(\frac{1}{2} \) 0.90 - \(\frac{1}{2} \) 0.88 -
	0.86 - 0.84 -
	0 2 4 6 8 epoch
In [15]:	<pre># history.history() history.keys() # dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy']) plt_plot(history_bistory['loss'])</pre>
	<pre>plt.plot(history.history['loss']) plt.plot(history.history['val_loss']) plt.title('model loss') plt.ylabel('loss') plt.xlabel('epoch')</pre>
	plt.legend(['Train', 'Validation'], loc='upper left') plt.show() model loss
	0.6 - Train Validation
	0.5 - <u>§</u> 0.4 -
	0.2
In []:	0 2 4 6 8 epoch