Building Neural Network to learn AI

Create a Neural Network , then train it with sample data and see how it works, recognizing handwritten numbers with TipsMake.com!

Neural Network - Neural Network is an important concept in the field of artificial intelligence and machine learning. They consist of interconnected nodes, arranged in layers and mimicking the way the human brain works. Nodes represent human nerves.

You can create your own multilayer, simple forwarding neural network. Train it to classify handwritten digits using the MNIST dataset. You can then use computer vision to classify your handwritten digits.

What is multiclass classification?

Multiclass classification is a type of machine learning that can categorize data into different categories. Neural networks use softmax classifiers to distribute probabilities or ratios over possible classes.

You can use multiclass classification for images from the MNIST dataset into 10 different categories. They correspond to the numbers 0 to 9.

What is MNIST Dataset?

This is a popular benchmark dataset for machine learning and computer vision algorithms. It contains 70,000 handwritten images at 28x28 pixels. Handwritten numbers range from 0 to 9.

Before building any machine learning pattern, you need to understand what the dataset contains. Understanding datasets will give you a better way of handling data.

Prepare the development environment

To follow the tutorial, you need to know the basics of Python and have a background in machine learning. Finally, feel free to use Jupyter Notebook or Google Colab.

Create a new Jupyter Notebook or sign up for Google Colab. Run this command to install the required package:

```
!pip install numpy matplotlib tensorflow opencv-python
```

You will use:

1. Matplotlib for data visualization.
2. NumPy for manipulating arrays.
3. TensorFlow to create and train your model.
4. OpenCV to give the model your own handwritten digits.

Import the required module

Import the packages you have installed in the environment. This allows you to call and use functions and their modules in code later.

```
import tensorflow as tf from tensorflow import keras import matplotlib.pyplot as
```

The second line of code imports the Keras module from the Google TensorFlow library. You will use Keras for intensive neural network training with TensorFlow as the backend.

Download and view Dataset

The MNIST dataset is integrated into Keras. Load the MNIST dataset and split it into training and test components. You will use train to orient the model and test to evaluate the accuracy of the sample in classifying new unseen images.

```
(X_train, y_train) , (X_test, y_test) = keras.datasets.mnist.load_data()
```

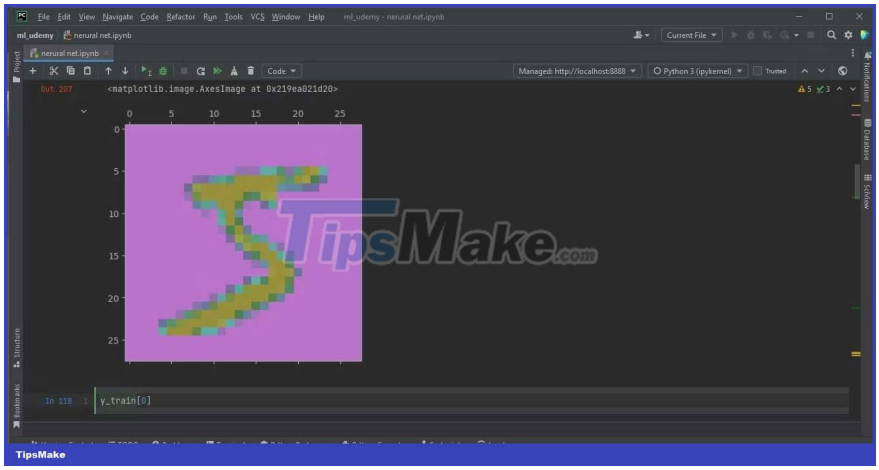
Check the length of the train and test sets. The MNIST dataset has 60,000 images for training and 10,000 images for testing.

```
len(X_train) len(X_test)
```

Check the shape of the first image in the MNIST dataset at 28x28 pixels. Then print the pixel values ??and illustrate it using Matplotlib.

```
X_train[0].shape X_train[0] plt.matshow(X_train[0]) y_train[0]
```

The following results:



Data pre-processing

Before using the data in the dataset to train and test the model, you need to process it first. This process improves accuracy by normalizing the data.

Normalize pixel values

Normalize the pixel values ??of the images in the dataset by dividing each value by 255. The pixel values ??of the unnormalized dataset range from 0 to 255 with 0 being black and 255 being white . Dividing each pixel value by 255 ensures each pixel is in the range between 0 and 1. This makes it easier for the model to learn related features and patterns in the data.

```
X_train = X_train / 255 X_test = X_test / 255
```

Then print the pixel values ??of the first photo.

```
X_train[0]
```

They are now in the range between 0 and 1.

Convert image matrix to 1D . array

The input layer of a neural network usually requires 1D input, so create a 1D array for the pixel values ??in the image. To do this, use the reshape() function along with the number of raw images to set the number of images in the dataset.

```
X_train_flattened = X_train.reshape(len(X_train), 28 * 28) X_test_flattened = X_
```

Your images are now ready to train and test the model.

In-depth neural network prototyping

Create a sequential model with Tensorflow's Keras module using 1 input layer, 2 hidden layers, and 1 output layer. Set the input shape to 28 x 28 as this is the shape of the original image in the dataset. Use 128 nodes for hidden layers. The output layer should have only 10 neurons because you are only classifying digits from 0 to 9.

```
model = keras.Sequential([ ????keras.layers.Flatten(input_shape=(28, 28)),
????keras.layers.Dense(128, activation='relu'), ????
keras.layers.Dense(128, activation='relu'), ????
keras.layers.Dense(10, activation='softmax') ])
```

Compile the model using the **adam** optimizer , **sparse_categorical_crossentropy** as the loss function and metric to evaluate the model's performance as the accuracy. Then fit the training data into the model and set the number of stages to 5.

```
model.compile(optimizer='adam', ?????????????
loss='sparse_categorical_crossentropy', ?????????????
metrics=['accuracy']) model.fit(X_train, y_train, epochs=5)
```

You need to wait a few minutes to finish 'training' the model. After finishing, evaluate its effectiveness in the test section.

```
model.evaluate(X_test, y_test)
```

The evaluation function will return the loss and precision of the model. This sample gives 98% accuracy.

Use model to classify handwritten numbers

You need to prepare the image to match the numbers in the MNIST dataset. Failure to do so will result in the model being created inefficiently.

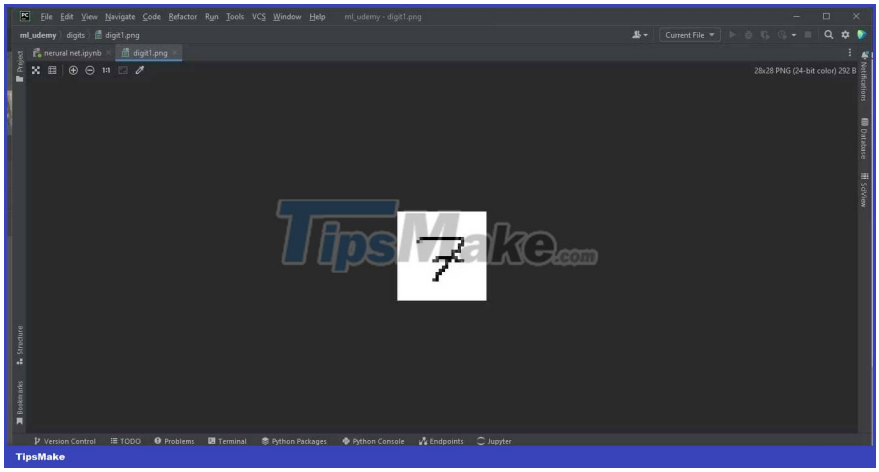
To preprocess the image:

1. Load an image with numbers using OpenCV.
2. Convert it to grayscale and resize it at 28x28 pixels.
3. Flip and normalize pixel values.
4. Finally flatten the image with a 1D array.

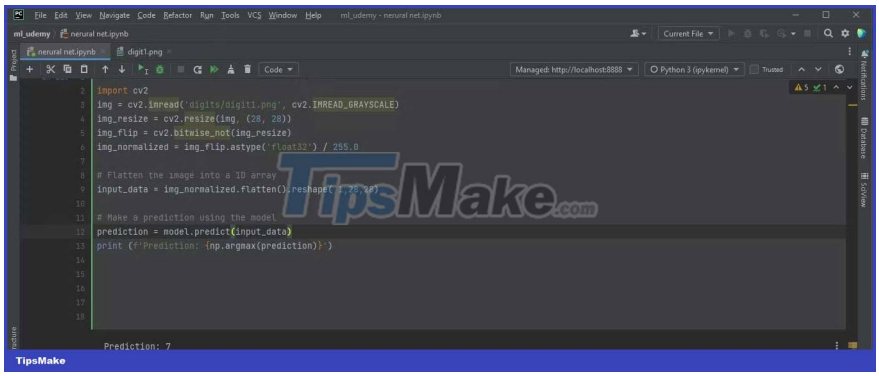
Transfer the preprocessed image to the model for prediction and print the predicted value on the screen.

```
img = cv2.imread('digits/digit1.png', cv2.IMREAD_GRAYSCALE) img_resize = cv2.resi
```

Pass a preprocessed image containing a number into the template:



Result:



Above is **how to build an artificial neural network** . Hope the article is useful to you.

You finished reading the article "**Building Neural Network to learn AI**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.
