How does CNN digit recognition work?

As an example, a popular dataset called MNIST was taken to make predictions of handwritten digits from 0 to 9. The dataset was cleaned, scaled, and shaped. Using TensorFlow, a CNN model was created and was eventually trained on the training dataset. Finally, predictions were made using the trained model.

Why CNN is used for classification?

The Convolutional Neural Network (CNN or ConvNet) is a subtype of Neural Networks that is mainly used for applications in image and speech recognition. Its built-in convolutional layer reduces the high dimensionality of images without losing its information. That is why CNNs are especially suited for this use case.

CNN_MNIST.PY

```
 import numpy as np

import argparse

import cv2

from cnn.neural_network import CNN

from keras.utils import np_utils

from keras.optimizers import SGD

# from sklearn.datasets import fetch_mldata

from sklearn.datasets import fetch_openml

from sklearn.model_selection import train_test_split

# Parse the Arguments

ap = argparse.ArgumentParser()

ap.add_argument("-s", "--save_model", type=int, default=-1)

ap.add_argument("-l", "--load_model", type=int, default=-1)

ap.add_argument("-w", "--save_weights", type=str)

args = vars(ap.parse_args())


# Read/Download MNIST Dataset
```

```
print('Loading MNIST Dataset...')
# dataset = fetch_mldata('MNIST Original')
dataset = fetch_openml('mnist_784')
# Read the MNIST data as array of 784 pixels and convert to 28x28 image matrix
mnist_data = dataset.data.reshape((dataset.data.shape[0], 28, 28))
mnist_data = mnist_data[:, np.newaxis, :, :]
# Divide data into testing and training sets.train_img, test_img, train_labels, test_labels = train_test_split(mnist_data/255.0, dataset.target.astype("int"), test_size=0.1)
# Now each image rows and columns are of 28x28 matrix type.img_rows, img_columns = 28, 28
# Transform training and testing data to 10 classes in range [0,classes] ; num. of classes = 0 to 9 = 10 classes
total_classes = 10                    # 0 to 9 labels
train_labels = np_utils.to_categorical(train_labels, 10)
test_labels = np_utils.to_categorical(test_labels, 10)
# Defing and compile the SGD optimizer and CNN model
print('\n Compiling model...')
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
clf = CNN.build(width=28, height=28, depth=1, total_classes=10, Saved_Weights_Path=args["save_weights"] if args["load_model"] > 0 else None)
```

How does SVM work in digit recognition?

The features of input handwritten digit obtained in N3 layer are treated as an input for the SVM classifier. The SVM classifier is initially trained with these new automatically generated features of training images. Finally, the trained SVM classifier is used in recognizing the digits used for testing .

SV.PY

```
import sys
import numpy as np
import pickle
from sklearn import model_selection, svm, preprocessing
from sklearn.metrics import accuracy_score,confusion_matrix
from MNIST_Dataset_Loader.mnist_loader import MNIST
import matplotlib.pyplot as plt
from matplotlib import style
style.use('ggplot')
# Save all the Print Statements in a Log file.
old_stdout = sys.stdout
log_file = open("summary.log","w")
sys.stdout = log_file
# Load MNIST Data
print('\nLoading MNIST Data...')
data = MNIST('./MNIST_Dataset_Loader/dataset/')
print('\nLoading Training Data...')
img_train, labels_train = data.load_training()
train_img = np.array(img_train)
train_labels = np.array(labels_train)
print('\nLoading Testing Data...')
img_test, labels_test = data.load_testing()
test_img = np.array(img_test)
test_labels = np.array(labels_test)

#Features
X = train_img
#Labels
```

```
y = train_labels
# Prepare Classifier Training and Testing Data
print('\nPreparing Classifier Training and Validation Data...')
X_train, X_test, y_train, y_test = model_selection.train_test_split(X,y,test_size=0.1)
# Pickle the Classifier for Future Use
print('\nSVM Classifier with gamma = 0.1; Kernel = polynomial
```