Deep Learning and Convolutional Neural Network

Project Report

Project Title: Train CNN on the SVHN dataset Classification

Student Name: Aftab Nafees

Course: Artificial Intelligence and Data Science

GitHub: https://github.com/aftab1038

LinkedIn: https://pk.linkedin.com/in/aftab1038

Repository: https://shorturl.at/rMKIX

Date: October 04, 2024

Table of Contents

Abstract

This project presents an image classification model based on a Convolutional Neural Network (CNN) trained on the SVHN (Street View House Numbers) dataset. The aim of the project is to accurately classify digits from house number plates using a deep learning approach. The model was trained using the train_32x32 dataset and tested on the test_32x32 dataset, achieving significant accuracy.

Introduction

The objective of this project is to develop a robust image classification model using CNN to classify digits from the SVHN dataset. The dataset contains images of digits from real-world street scenes, making it an ideal benchmark for machine learning models. The project involves training a CNN on the dataset to automatically recognize the digits present in the images.

Dataset Description

The SVHN dataset is a real-world image dataset obtained from house numbers in Google Street View images. It is similar to the MNIST dataset but contains a larger set of 32x32 RGB images, making it more challenging. This project uses the train_32x32.mat and test_32x32.mat files, where the training set contains 73,257 labeled images, and the test set contains 26,032 labeled images. The goal is to predict the correct digit (0-9) present in each image.

Methodology

The project uses a Convolutional Neural Network (CNN) implemented using TensorFlow and Keras libraries. The CNN consists of three convolutional layers followed by max-pooling layers. A dropout layer is introduced to prevent overfitting. The final output layer uses a softmax activation function to predict the class (digit) for each input image. Data augmentation techniques like rotation, zoom, and width/height shifts are applied to enhance the model's generalization capability.

Code

The following code shows the CNN architecture and data preprocessing steps:

```
import numpy as np
from scipy.io import loadmat
from tensorflow.keras.utils import to_categorical
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.image import
ImageDataGenerator

# Load dataset
train_data = loadmat('train_32x32.mat')
test_data = loadmat("test_32x32.mat")
```

```
X_train = train_data['X']
y_train = train_data['y'].flatten()
X_test = test_data['X']
y_test = test_data['y'].flatten()

# Data preprocessing
X_train = np.moveaxis(X_train, -1, 0)
X_test = np.moveaxis(X_test, -1, 0)
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0
y_train = to_categorical(y_train, num_classes=10)
y_test = to_categorical(y_test, num_classes=10)

# Data augmentation
train_datagen = ImageDataGenerator(rotation_range=10,
zoom_range=0.1, width_shift_range=0.1,
height_shift_range=0.1)
train_generator = train_datagen.flow(X_train, y_train,
batch_size=64)

# CNN architecture
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(10, activation='softmax'))

# Compile and train model
model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(train_generator, epochs=20,
validation_data=(X_test, y_test))
```

```
# Save model
model.save('trained_model.h5')
```

Model Performance and Evaluation

The CNN model was trained over 20 epochs with the following results:

1. Initial accuracy was 26.54% with a validation accuracy of 77.39% in the first epoch.
2. The model gradually improved and reached a final training accuracy of 85.97% and a validation accuracy of 92.08% after 20 epochs.
3. The model's loss decreased from 2.06 in the first epoch to 0.4777 by the last epoch, with a validation loss of 0.2836.

This performance indicates that the model successfully learned to classify the digits in the SVHN dataset with high accuracy, with further potential for improvement.

Results

The CNN model was trained for 20 epochs using the Adam optimizer. The training accuracy improved over the epochs, and the model achieved a significant accuracy on the test set. The final trained model was saved as 'trained_model.h5'. Further analysis using confusion matrices and classification reports could be conducted to assess the model's performance.

Discussion

The use of CNNs for digit classification from real-world images proved effective, achieving high accuracy. Data augmentation was instrumental in preventing overfitting and improving generalization. Further improvements could include experimenting with deeper network architectures or fine-tuning hyperparameters.

Conclusion

This project successfully developed a CNN model to classify digits from the SVHN dataset with significant accuracy. The application of data augmentation and dropout contributed to the model's performance. Future work could focus on improving the model through additional techniques or applying it to other image classification tasks.