Image Classification using CNN

Report on SVHN Dataset  
Rahat Karim  
DSAI-258

Rahat Karim

Contents

[1. Introduction 1](#_Toc178953202)

[2. Data Overview 1](#_Toc178953203)

[ Training Data Shape 1](#_Toc178953204)

[ Test Data Shape: 1](#_Toc178953205)

[3. Data Preprocessing 2](#_Toc178953206)

[ Reshaping: 2](#_Toc178953207)

[ Normalization: 2](#_Toc178953208)

[ Label Correction: 2](#_Toc178953209)

[ One-hot Encoding: 2](#_Toc178953210)

[Data Split 2](#_Toc178953211)

[4. Model Architecture 2](#_Toc178953212)

[Model Compilation 2](#_Toc178953213)

[5. Model Training 2](#_Toc178953214)

[6. Model Evaluation 3](#_Toc178953215)

[Confusion Matrix 3](#_Toc178953216)

[7. Visualizing Test Results 3](#_Toc178953217)

[8. Results 3](#_Toc178953218)

[9. Conclusion 5](#_Toc178953219)

[Resources: https://github.com/Rahat-karim/CNN 5](#_Toc178953220)

## 1. Introduction

This project involves the use of a Convolutional Neural Network (CNN) to classify images from the Street View House Numbers (SVHN) dataset. The SVHN dataset consists of color images of house numbers, which are typically used for digit recognition tasks. The dataset was downloaded from a public source and contains both training and test data.

## 2. Data Overview

The SVHN dataset is composed of images of house numbers with corresponding labels:

* Training Data Shape: (32, 32, 3, 73257) - This consists of 73,257 samples, with each image having a shape of 32x32 pixels and 3 channels (RGB).
* Test Data Shape: (32, 32, 3, 26032) - The test set contains 26,032 samples.
* The labels range from 0-9, representing the digits shown in the images.

After loading the dataset, the images were reshaped to fit TensorFlow's expected input format and normalized to a range of [0, 1].

## 3. Data Preprocessing

* Reshaping: The images were reshaped from (32, 32, 3, num\_samples) to (num\_samples, 32, 32, 3), which is the required format for TensorFlow’s CNN input.
* Normalization: All pixel values were normalized by dividing by 255 to ensure that the values are in the range [0, 1].
* Label Correction: In the SVHN dataset, the digit "0" is labeled as "10." This was corrected by mapping all occurrences of the label "10" to "0."
* One-hot Encoding: Labels were one-hot encoded, converting them from scalar values to binary class matrices with 10 classes.

### Data Split

The training data was split into a training set and a validation set using an 80-20 ratio.

### 4. Model Architecture

The CNN model is composed of the following layers:

* Conv2D Layer 1: 32 filters, (3x3) kernel size, with ReLU activation, followed by a **MaxPooling2D** layer (2x2).
* Conv2D Layer 2: 64 filters, (3x3) kernel size, with ReLU activation, followed by another **MaxPooling2D** layer (2x2).
* Conv2D Layer 3: 128 filters, (3x3) kernel size, with ReLU activation, followed by a **MaxPooling2D** layer (2x2).
* Flatten Layer: Converts the 2D feature maps into 1D vectors.
* Dense Layer: 128 units with ReLU activation.
* Dropout Layer: Dropout rate of 0.5 to prevent overfitting.
* Output Layer: A Dense layer with 10 units (one for each class) and softmax activation to output class probabilities.

### Model Compilation

The model was compiled using the **Adam optimizer**, **categorical cross-entropy** as the loss function, and **accuracy** as the evaluation metric.

## 5. Model Training

The model was trained for **10 epochs** with a batch size of 64. Below is a summary of the training process:

* Epoch 1: Accuracy 0.3688, Validation Accuracy 0.8277
* Epoch 5: Accuracy 0.8749, Validation Accuracy 0.8901
* Epoch 10: Accuracy 0.8890, Validation Accuracy 0.8969

The model's performance improved steadily during training, with validation accuracy reaching around **90%** after the final epoch.

## 6. Model Evaluation

The trained model was evaluated on the test set, achieving a **test accuracy of 89.69%**, which indicates a strong ability to generalize to unseen data.

### Confusion Matrix

A confusion matrix was generated to assess the model’s performance on individual classes. The matrix provides insights into which digits the model misclassifies more often and helps diagnose potential areas of improvement.

## 7. Visualizing Test Results

To further analyze the model's predictions, we visualized some test samples along with their true and predicted labels. The model was able to correctly predict most of the samples, and the visualizations allowed us to see how well it distinguished between different digits.

### Sample code

# 4. Split the train set into training and validation sets

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X\_train, y\_train, test\_size=0.2, random\_state=42)

# 5. Define the CNN model

model = Sequential([

Conv2D(32, (3, 3), activation='relu', input\_shape=(32, 32, 3)),

MaxPooling2D(pool\_size=(2, 2)),

Conv2D(64, (3, 3), activation='relu'),

MaxPooling2D(pool\_size=(2, 2)),

Conv2D(128, (3, 3), activation='relu'),

MaxPooling2D(pool\_size=(2, 2)),

Flatten(),

Dense(128, activation='relu'),

Dropout(0.5),

Dense(10, activation='softmax')

])

# Compile the model

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

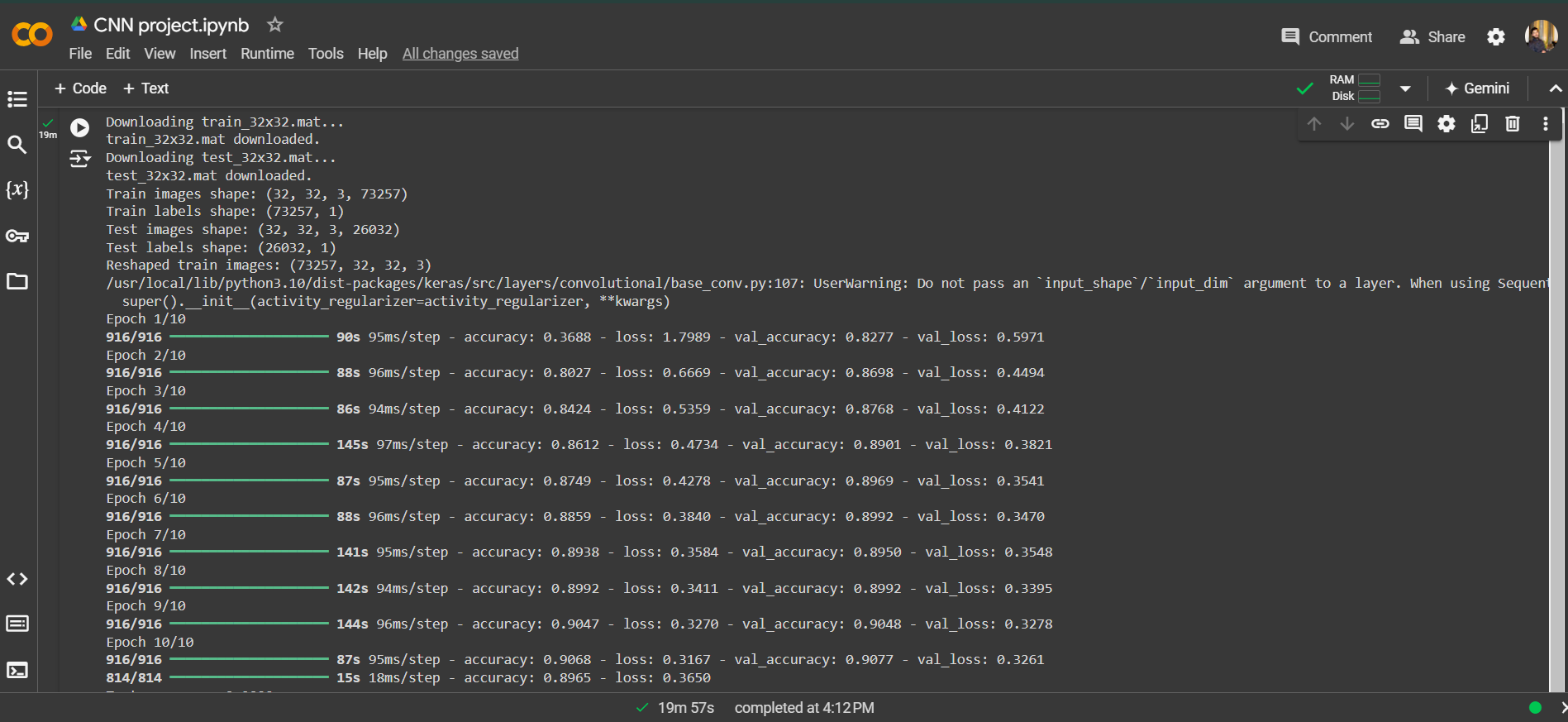
# 6. Train the model

history = model.fit(X\_train, y\_train, validation\_data=(X\_val, y\_val), epochs=10, batch\_size=64)

# 7. Evaluate the model

test\_loss, test\_acc = model.evaluate(X\_test, y\_test)

print(f"Test accuracy: {test\_acc:.4f}")

8. Results  
  
A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

## 9. Conclusion

This project demonstrated the application of a Convolutional Neural Network for digit classification using the SVHN dataset. After training for 10 epochs, the model achieved high accuracy on both training and validation sets, and a test accuracy of **89.69%**. The confusion matrix provided insights into misclassifications, and overall, the results indicate the effectiveness of CNNs for image classification tasks.

## Resources: <https://github.com/Rahat-karim/CNN>