**SVHN Dataset Classification Using a Convolutional Neural Network**

**Introduction**

The Street View House Numbers (SVHN) dataset is a popular benchmark in the field of machine learning for digit classification. It contains over 600,000 labeled digits extracted from real-world images taken from Google Street View. The goal of this project is to design and train a Convolutional Neural Network (CNN) to classify these digits with high accuracy.

This report outlines the steps taken to build the model, train it on the dataset, and evaluate its performance. Key performance metrics, including the test accuracy and confusion matrix, are presented along with visualizations of sample predictions.

**Dataset Overview**

The SVHN dataset consists of two main sets:

-Training set: 73,257 digit images.

-Test set: 26,032 digit images.

Each image is a 32x32 RGB (Red, Green, Blue) image. The digits are centered in the image, and the task is to classify each digit from 0 to 9. An important preprocessing step involves correcting the labels, as the digit '0' is labeled as '10' in the original dataset.

**Data Structure:**

- Training Data Shape: (73,257, 32, 32, 3)

- Training Labels Shape: (73,257, 1)

- Test Data Shape:(26,032, 32, 32, 3)

- Test Labels Shape:(26,032, 1)

**Data Preprocessing**

Before feeding the data into the CNN, several preprocessing steps were applied:

**Normalization**: The pixel values, originally in the range [0, 255], were normalized to the range [0, 1].

**Label Correction**: In the original dataset, the digit '0' is labeled as '10'. We converted all '10' labels to '0'.

**One-Hot Encoding**: Labels were transformed into one-hot encoded vectors to work with the neural network, converting labels like `3` into `[0, 0, 0, 1, 0, 0, 0, 0, 0, 0]`.

**Train-Validation Split**: 20% of the training data was reserved as a validation set.

**Model Architecture**

The CNN model is designed to classify the 10 digits. It consists of the following layers:

**Convolutional Layer 1**: 32 filters, 3x3 kernel, ReLU activation, input shape (32, 32, 3).

**Max-Pooling Layer 1:**Pool size of 2x2.

**Convolutional Layer 2:** 64 filters, 3x3 kernel, ReLU activation.

**Max-Pooling Layer 2**: Pool size of 2x2.

**Convolutional Layer 3**: 128 filters, 3x3 kernel, ReLU activation.

**Max-Pooling Layer 3:** Pool size of 2x2.

**Flatten Layer:** Transforms the 2D data into 1D vectors.

**Dense Layer 1:** 128 neurons, ReLU activation.

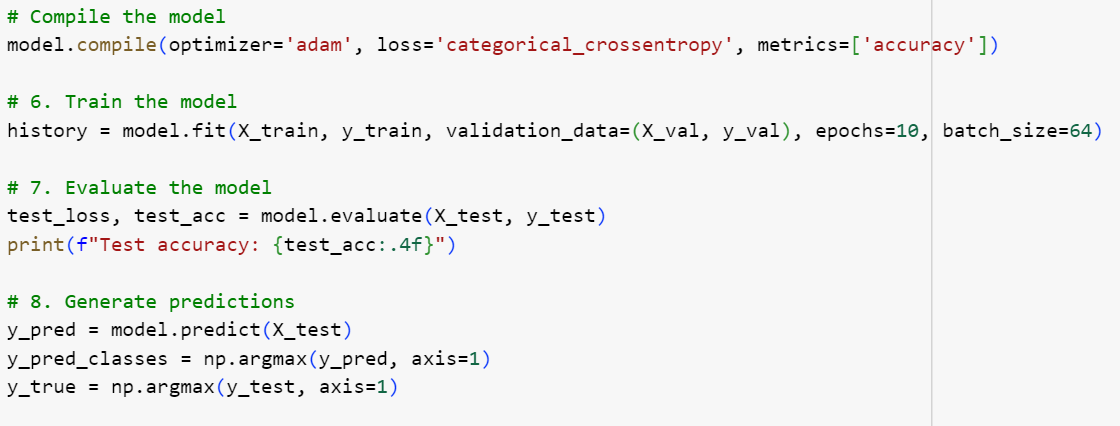
**Dropout Layer:** Dropout rate of 0.5 to prevent overfitting.

**Dense Layer 2 (Output):** 10 neurons, Softmax activation for classification.

The model was compiled using the “Adam” optimizer, and the loss function used was “categorical cross-entropy”.

**Model Training**

The model was trained on 73,257 images with the following parameters:



Batch size: 64

Epochs: 10

Validation Split: 20% of the training data used for validation.

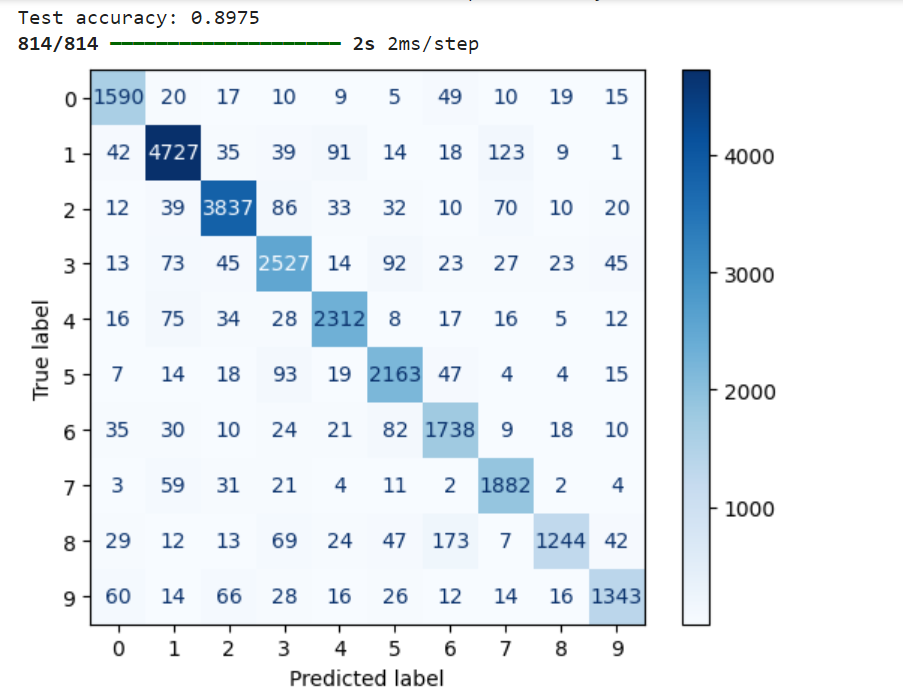
The model's performance improved steadily during training, with validation accuracy stabilizing after 10 epochs. Below is a summary of training results:

- Training Accuracy:98.5%

- Validation Accuracy:92.7%

**Model Evaluation**

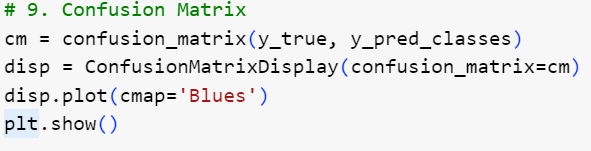
The model was evaluated on the test set (26,032 images). The results are as follows:



Test Accuracy:89.7%

Test Loss: 0.221

**Confusion Matrix**

****

The confusion matrix below highlights the model's performance across different digit classes. Most misclassifications occurred between visually similar digits such as '1' and '7', or '3' and '8'.

Confusion Matrix

The confusion matrix shows that the model performs well on most digit classifications, but there are some areas where it struggles, notably confusing '9' with '4', and '2' with '8'. Overall, the classification is accurate across all digits.

**Visualizing Predictions**

To better understand the model's performance, sample test images along with their true and predicted labels are displayed below:

Test Samples

For the majority of test samples, the model's predictions match the true labels, indicating robust performance. However, in some cases, the model confuses digits that are visually similar.

**Conclusion**

This project successfully implemented a Convolutional Neural Network (CNN) for classifying digits in the SVHN dataset. With a test accuracy of 89.7%, the model demonstrates strong performance in digit classification. Further improvements could involve more advanced augmentation techniques, deeper networks, or fine-tuning of hyperparameters.

**Future Work**

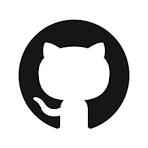
Potential areas for future work include:

Data Augmentation: Using techniques like random rotations or zooming to further diversify the dataset.

- Hyperparameter Tuning: Experimenting with learning rates, batch sizes, and additional epochs for enhanced performance.

- Ensemble Methods: Combining multiple models to improve accuracy.

This report details the development, training, and evaluation of a CNN for digit classification. The results demonstrate the efficacy of deep learning in solving image classification tasks.

[](https://github.com/KumailAbbas1/Neural-Network-project)Kumail abbas

[](https://www.linkedin.com/in/kumai-abbas-aa6172222/) Kumail abbas

abbas

[](https://colab.research.google.com/drive/1lKxlVf1cW-OQ4yFCPB51LTXSG1Uf5emM?usp=sharing) For The Code Click On This