SVHN Classification with CNN — Detailed Project Report

A. Introduction

The objective of this project is to build a robust digit recognition model using the **Street View House Numbers (SVHN)** dataset, a real-world dataset composed of digit images from Google Street View. Unlike MNIST, SVHN images are in color and contain complex backgrounds, making this a more challenging and realistic classification problem. Each image is a cropped digit (0–9) from house numbers in street scenes, presented as a 32×32 RGB image.

Applications include:

- - Address and street sign recognition
- - Automated postal sorting
- - OCR in navigation and mapping apps
- - Smart cameras and intelligent transportation systems

B. Dataset Overview

- - **Source:** SVHN (Cropped Digits) from `.mat` files
- - **Train Samples:** ~73,000
- - **Test Samples:** ~26,000
- - **Classes:** 10 digits (0–9); label '10' is mapped to '0'
- - **Format:** 32×32 RGB images; cropped digits

Key characteristics:

- - **Complex backgrounds:** Unlike MNIST, SVHN includes background clutter
- - **Color variance:** Input is 3-channel (RGB), not grayscale
- - **Real-world noise:** Some images include neighboring digits or lighting issues

C. Methodology (Detailed)

1. Data Loading

Data is loaded from MATLAB `.mat` files using `scipy.io.loadmat`. Each file contains:

- - `X`: 4D image tensor with shape (32, 32, 3, N)
- - `y`: Label vector with shape (N,)

To prepare this data:

- - Transpose `X` to shape (N, 32, 32, 3)
- - Normalize pixel values to [0, 1] by dividing by 255
- - Replace label '10' with '0'

2. Preprocessing

After loading, preprocessing includes:

a. One-hot encoding:

- - Converts numeric labels (0–9) to categorical vectors for softmax output layer compatibility.

b. Train-validation split:

- - To prevent overfitting, 20% of training data is used for validation using `train_test_split`.

c. Data augmentation: *(optional but beneficial)*

Using `ImageDataGenerator`, the dataset is artificially expanded with real-time augmentation:

- - Rotation (±10°)
- - Zoom (±10%)
- - Horizontal and vertical shifts (±10%)

This helps the model generalize better on unseen data by exposing it to slight distortions.

3. CNN Model Architecture

A custom Convolutional Neural Network (CNN) is built using `Sequential` API. It includes:

- - **Convolutional layers (Conv2D):**

- - Extract spatial features from the image
- - Filters of sizes 32, 64, and 128 progressively learn low to high-level features

- - **BatchNormalization:**

- - Standardizes outputs of layers to stabilize and speed up training

- - **MaxPooling2D:**

- - Downsamples the feature maps, reducing spatial dimensions

- - **Dropout:**

- - Randomly disables neurons to prevent overfitting

- - **Dense (Fully Connected) layers:**

- - Translates feature maps into class probabilities

- - **Softmax output:**

- - Produces a probability distribution over 10 classes

Optimizer: **Adam** (adaptive learning rate)

Loss Function: **Categorical Crossentropy**

4. Training Strategy

- - **Batch size:** 128
- - **Epochs:** 50 (with early stopping)
- - **Callbacks:**
- - `EarlyStopping`: Stops training if validation loss doesn't improve
- - `ReduceLROnPlateau`: Reduces learning rate on performance plateau

Training is monitored for both loss and accuracy on training and validation sets. This allows tracking of underfitting or overfitting behavior.

D. Model Evaluation

1. Accuracy

Model achieved ~96% test accuracy — a strong result considering the complexity of SVHN.

2. Confusion Matrix

A 10x10 matrix showing true labels vs. predicted labels, used to identify:

- - Which digits are misclassified most
- - Confusion trends (e.g. 3 vs. 5, 8 vs. 0)

3. Classification Report

Includes:

- - **Precision**: Correct positive predictions / total predicted positives
- - **Recall**: Correct positive predictions / total actual positives
- - **F1-score**: Harmonic mean of precision and recall

4. Misclassified Examples

10 incorrectly predicted images are visualized. Most errors are due to:

- - Blurred or partially occluded digits
- - Side digits not removed completely during cropping
- - Background color blending with digit

E. Results Summary

Digit-wise Precision, Recall, F1-Score:

0: 96%, 97%, 96%

1: 98%, 99%, 99%

2: 95%, 94%, 95%

3: 94%, 92%, 93%

4: 96%, 95%, 95%

5: 93%, 92%, 93%

6: 95%, 96%, 96%

7: 96%, 95%, 96%

8: 94%, 95%, 95%

9: 94%, 93%, 94%

Macro average: Precision = 0.95, Recall = 0.95, F1 = 0.95

F. Challenges and Limitations

- - **Cluttered backgrounds**: Some cropped images still contain side digits
- - **Digit similarity**: 3 vs 5, 8 vs 0 misclassifications
- - **Small variations**: Rotations and occlusions affect predictions

G. Future Improvements

1. 1. **Use the Extra Set**: SVHN provides ~500k additional labeled samples.

2. **Switch to Transfer Learning**: Use EfficientNetB0, MobileNet, or ResNet with pretrained weights.

3. **CTC-based digit string recognition**: Instead of cropped digits, build models that detect and recognize multi-digit strings.

4. **Hyperparameter tuning**: Explore optimizers, dropout rates, and learning rate schedules.

5. **Ensemble models**: Combine predictions of multiple CNNs.

H. Conclusion

This project demonstrates that a well-designed CNN can perform strongly on real-world digit recognition using the SVHN dataset. The model handles noise and complexity well, achieving ~96% accuracy. With more data and fine-tuning, it could be deployed in real applications like address recognition and smart OCR.