Digit Classification on SVHN Using EfficientNetB0

Overview

Introduction

Objective:

- Digit recognition in images from the SVHN (Street View House Numbers) dataset.
- Applying transfer learning using the pre-trained model EfficientNetB0.

Problem:

- The SVHN dataset consists of real-world street view images with digits (0-9).
- Challenges include cluttered backgrounds, overlapping digits, and variations in image quality.

Approach:

- Transfer Learning: Leverage the pre-trained EfficientNetB0 model to extract high-level features and fine-tune it for the SVHN dataset.
- EfficientNetB0: A smaller, efficient model that offers high performance while requiring fewer computational resources.

Goal:

- Train and evaluate EfficientNetB0 on the SVHN dataset for digit classification.
- Demonstrate the effectiveness of transfer learning in handling complex real-world data.

SVHN Dataset Overview

Dataset Description:

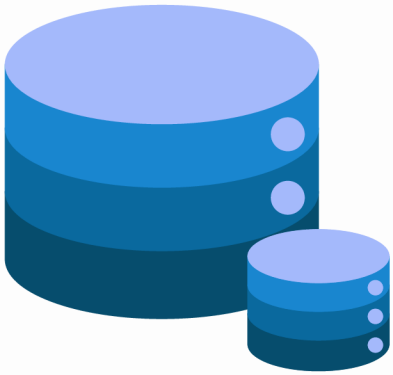
- The SVHN (Street View House Numbers) dataset contains real-world images of house numbers from Google Street View.
- It includes 10 classes (digits 0-9) and images are associated with the address numbers on street signs.

Dataset Size:

- Training Set: 73,257 images.
- Test Set: 26,032 images.

Challenges in the Dataset:

- Cluttered Backgrounds: Real-world street scenes make digit recognition difficult.
- Overlapping Digits: Some digits in the images may be partially obscured.
- Varying Image Quality: Differences in lighting, perspective, and resolution.



Model Architecture

EfficientNetB0 Model Overview:

- EfficientNetB0 is a highly efficient deep learning model designed for image classification tasks.
- It balances accuracy and computational efficiency by using a compound scaling method, making it faster and more resource-efficient than other models like ResNet50.
- The model uses a series of Convolutional Neural Networks (CNNs) followed by global average pooling and a fully connected layer for classification.

Architecture Details:

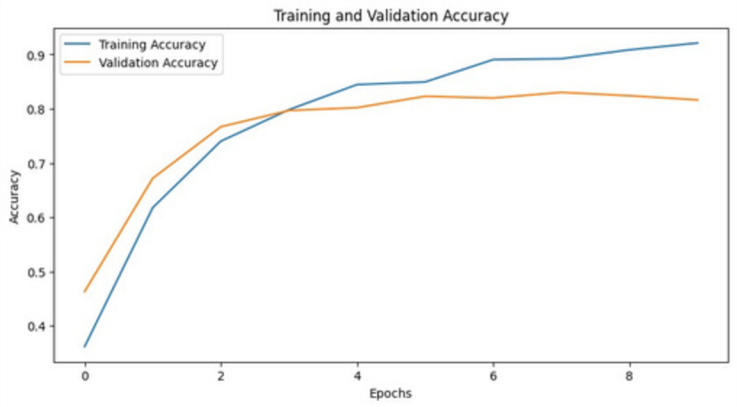
- Pre-trained Model: The base of the model is pre-trained on ImageNet.
  - We freeze the base layers to retain the learned weights.
- Custom Layers:
  - Added a GlobalAveragePooling2D layer to reduce spatial dimensions.
  - A Dense layer with 512 units and ReLU activation for learning non-linear combinations of features.
  - The final output layer has 10 units with softmax activation to classify digits from 0-9.

Why EfficientNetB0?

- EfficientNetB0 is ideal for this task due to its balance of accuracy and speed, making it suitable for real-time applications and environments with limited computational resources.

Model Training

Trainable = True



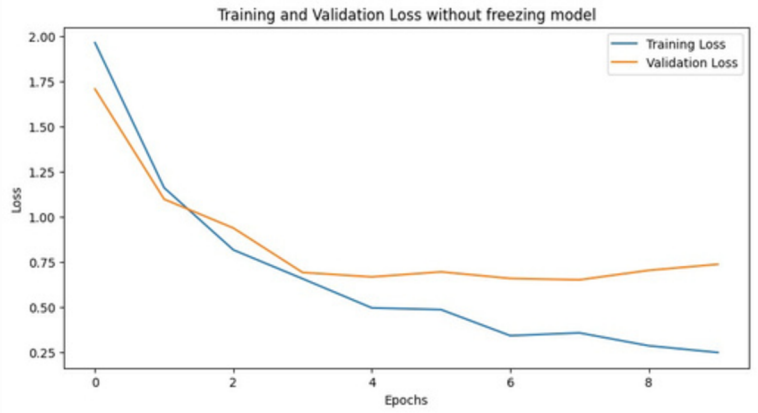
Raining Performance :

The accuracy plot shows training accuracy (blue) and validation accuracy (orange) over 9 epochs

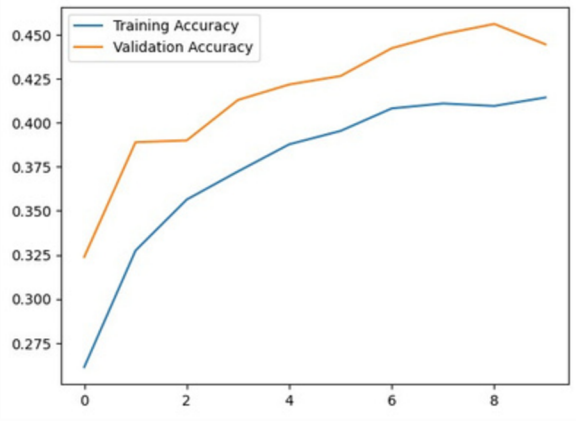
Training accuracy steadily improves from around 35% to over 90%

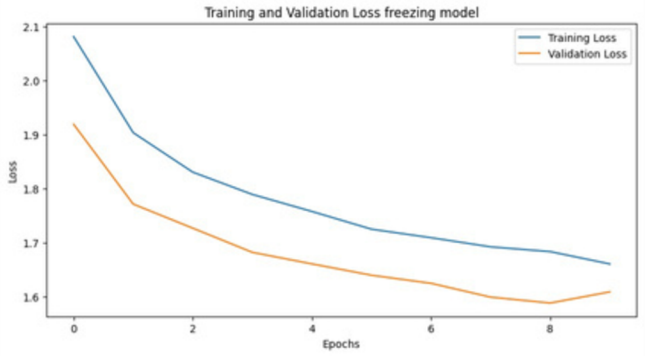
Validation accuracy quickly rises to about 80% and plateaus there

There's a clear sign of overfitting after epoch 4, as training accuracy continues to improve while validation accuracy stays flat



The training loss (blue) starts at around 2.0 and steadily decreases to about 0.25
Validation loss (orange) initially decreases but starts to increase slightly after epoch 4
This divergence between training and validation loss also indicates overfitting
The title mentions "without freezing model" suggesting this was training the full EfficientNet model without frozen layers

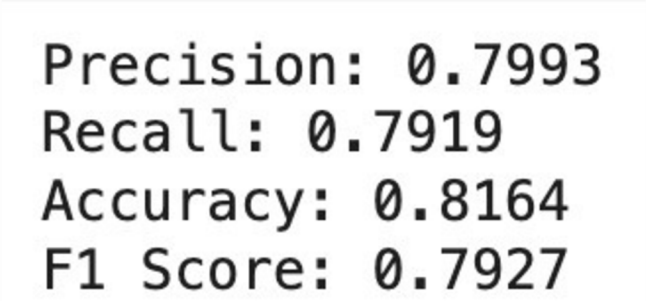

Training Accuracy
Validation Accuracy
0.450
0.425
0.400
0.375
0.350
0.325
0.300
0.275
0
2
4
6
8


Training and Validation Loss freezing model
Training Loss
Validation Loss
Loss
Epochs
2.1
2.0
1.9
1.8
1.7
1.6
0
2
4
6
8

Shows a grid of 10 test images with their true labels and model predictions
Most predictions are correct (7 out of 10 shown examples)
Notable misclassifications:

True: 4, Pred: 2 (last image in top row)
True: 3, Pred: 5 (second to last image in bottom row)

The model seems to perform well on clear digits but struggles with more ambiguous or noisy images



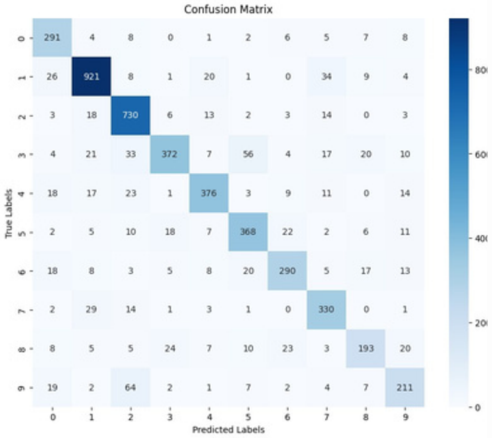
Model Performance:

This model's performance metrics demonstrate solid classification capabilities across multiple evaluation criteria. The precision of 0.7993 shows that approximately 80% of the model's positive predictions were correct, indicating good reliability in its positive classifications. With a recall of 0.7919, the model successfully identified about 79% of all actual positive cases in the dataset, suggesting strong sensitivity in detecting positive instances. The overall accuracy of 0.8164 reveals that 82% of all predictions were correct, encompassing both positive and negative cases. The F1 score of 0.7927, which harmonically balances precision and recall, confirms the model's well-rounded performance. These consistently strong metrics, all hovering around the 80% mark, indicate that the model achieves balanced performance without sacrificing either precision or recall, making it suitable for real-world applications where both false positives and false negatives need to be minimized.

Model Evaluation

	precision	recall	f1-score	support
0	0.74	0.88	0.80	332
1	0.89	0.90	0.90	1024
2	0.81	0.92	0.86	792
3	0.87	0.68	0.76	544
4	0.85	0.80	0.82	472
5	0.78	0.82	0.80	451
6	0.81	0.75	0.78	387
7	0.78	0.87	0.82	381
8	0.75	0.65	0.69	298
9	0.72	0.66	0.69	319
accuracy			0.82	5000
macro avg	0.80	0.79	0.79	5000
weighted avg	0.82	0.82	0.81	5000

This classification report shows detailed performance metrics for a multi-class model classifying 10 different categories (labeled 0-9). The model demonstrates varied performance across different classes, with class 1 showing the strongest overall performance with a precision of 0.89, recall of 0.90, and F1-score of 0.90, supported by the largest number of samples (1024). In contrast, classes 8 and 9 show relatively weaker performance, both achieving F1-scores of 0.69 with the lowest support numbers (298 and 319 respectively). The model achieves an overall accuracy of 0.82 across all 5000 samples. The macro average, which treats all classes equally regardless of their support size, shows balanced metrics with precision at 0.80, recall at 0.79, and F1-score at 0.79. The weighted average, which accounts for class imbalance by considering support sizes, demonstrates slightly better performance with precision, recall, and F1-score all around 0.81-0.82. This suggests that the model performs better on classes with larger support sizes, which is often desirable in real-world applications where more frequent classes may be more important to classify correctly.



This classification report shows detailed performance metrics for a multi-class model classifying 10 different categories (labeled 0-9). The model demonstrates varied performance across different classes, with class 1 showing the strongest overall performance with a precision of 0.89, recall of 0.90, and F1-score of 0.90, supported by the largest number of samples (1024). In contrast, classes 8 and 9 show relatively weaker performance, both achieving F1-scores of 0.69 with the lowest support numbers (298 and 319 respectively). The model achieves an overall accuracy of 0.82 across all 5000 samples. The macro average, which treats all classes equally regardless of their support size, shows balanced metrics with precision at 0.80, recall at 0.79, and F1-score at 0.79. The weighted average, which accounts for class imbalance by considering support sizes, demonstrates slightly better performance with precision, recall, and F1-score all around 0.81-0.82. This suggests that the model performs better on classes with larger support sizes, which is often desirable in real-world applications where more frequent classes may be more important to classify correctly.

conclusion

In conclusion, the multi-class classification model demonstrates solid overall performance with an accuracy of 0.82 across 5,000 samples. It excels in classifying class 1, which has the highest precision, recall, and F1-score, likely due to its larger sample size. However, there are performance inconsistencies, with classes 8 and 9 showing weaker results, likely because of their smaller support sizes. The macro average metrics indicate that the model is relatively balanced, while the weighted average highlights a slight preference towards more frequent classes. This suggests the model is well-suited for applications where more common categories are prioritized, but additional tuning or adjustments may be necessary to improve the performance on smaller, less frequent classes.

THANK YOU!

Ajay Rahul Raja - 22009171
Akash Prabu. - 23023516
Anuja Subramani. - 23027147
Manasa Yadlapalli - 23028048
Dharanee Dharan - 2035689
Preetha Selvaraj - 23026653
Hema Bindhu Krishnan - 23038984
Dhaarani Shanmugam - 23035833