Deep Learning for Deciphering Traffic Signs

Group 7 - Yifan Fan, Victor Floriano , Jose Salerno

Problem Statement & Motivation:
As the world advances towards autonomous vehicles, our team has observed the remarkable efforts of large car manufacturers, who are working with data scientists to develop fully autonomous cars. Our team is excited to create neural network models that can classify traffic signs, aiding car makers in overcoming challenges with neural networks for autonomous or assisted driving technologies.We believe autonomous driving to be an important problem to solve due to the great economic benefits it can generate for car manufacturers and the improvement of general driving safety.

Data and Methodology:
Dataset: We utilized the German Traffic Sign Recognition Benchmark (GTSRB), featuring over 50,000 images across 43 traffic sign classes. This publicly accessible dataset was managed through preprocessing for uniformity, data augmentation for robustness, and batch processing for computational efficiency. We also leveraged distributed computing for faster data handling and employed stratified sampling for efficient, representative experimentation.

Our analysis involved four models: Support Vector Machines (SVMs), Multilayer Perceptrons (MLPs), Convolutional Neural Networks (CNNs), and a pre-trained ResNet-18. The SVM provided a non-deep learning baseline, while the MLP introduced us to basic deep learning techniques. CNNs were selected for their effectiveness with image data. The pre-trained ResNet-18 helped compare our custom CNN's performance against a model trained on a diverse dataset, assessing our approach's efficacy in traffic sign recognition. All models were assessed through validation accuracy.

Exploratory Data Analysis:
During the exploratory analysis process, we encountered several challenges with our dataset. One major issue was the inconsistency in data organization. While the training dataset was neatly divided into separate folders for each of the distinct traffic sign classes, the testing dataset contained all images in a single folder, making it difficult for us to read the image files in a consistent manner and apply the right labels. To address this, we restructured the testing dataset to match the organization of the training dataset.

Another challenge was the class imbalance in the training dataset (**Image 1**). Certain classes contained a large number of images (max=2250), while others had far fewer (min=210), leading to potential biases in the model's predictions. To mitigate this, we calculated class weights and used the *`BCEWithLogitsLoss`* function to adjust for the imbalance. This approach helped ensure that the models could accurately predict traffic signs across all classes, reducing the risk of bias towards overrepresented classes. Another challenge we faced was the variation in image resolutions within the training and test data. We analyzed the pixel values of images and found the average resolution to be 56x56, which offered us a baseline for resizing our following models.

Model 1: SMV - Baseline
Our first model was a Support Vector Machine (SVM), traditionally known for its effectiveness in various classification tasks due to its ability to model non-linear relationships. To address the slow training times associated with SVMs, we converted all images from RGB to grayscale, effectively reducing data dimensionality, and resized them to 32x32 pixels to decrease computational demand. We selected the radial basis function (rbf) kernel for its proficiency in handling non-linear data relationships, a common characteristic in image data. This choice proved to be effective as we achieved a surprising **Validation Accuracy of 86%** on the GTSRB dataset, establishing a strong baseline for subsequent comparisons with our neural network models.

Model 2: MLP
As our first neural networks model our team decided to develop a multi-layer perceptron (MLP). We utilized Weights & Biases to conduct a randomized hyperparameter search to experiment with various configurations. Our model design was dynamic, adjusting the number of hidden layers and the dimensions of these hidden layers based on the current settings in the hyperparameter search. Moreover, techniques like data augmentation (horizontal flip, random rotation, random perspective), dropout layers, batch normalization, weight decay, and varying learning rates and batch sizes were also adapted dynamically for each trial to try to optimize model performance.

However, due to time and computational limits, we tested only 9 different configurations for 15 epochs each. As shown in **Image 2** even after extensive hyperparameter tuning, our best model configuration, Sweep 7, achieved a **Validation Accuracy of only 82%**, still falling short of the performance of our baseline SVM model. It was interesting that from our limited trials it appears that simpler model configurations, with smaller hidden layers and a lower number of hidden layers performed better (**Image 3**).

Model 3: CNN
Then, we developed a CNN model and performed various experiments.The model's architecture, with its structured with convolutional and pooling layers, adeptly minimized the complexity of the input while preserving the crucial details needed to differentiate between the numerous sign categories. Importantly, we incorporated dropout layers to avoid overfitting, keeping the model adaptable for the varied and unpredictable real-world images. In particular, we manually adjusted the model's hyperparameters, such as epochs, early-stopping parameters, and batch size, and the first model run achieved a near-perfect **Validation Accuracy of 98.76%** and **Testing Accuracy of 93.62%** (**Image 4).** Interestingly, when data augmentation and balance techniques didn't perform as hoped, we decided to set them aside, relying instead on the solid methods that led our CNN to success.

Model 4: Pre-trained
We opted to experiment with a pre-trained model to see how well it could categorize traffic signs. We selected ResNet-18, part of Microsoft's Residual Network models, which had been pre-trained on the ImageNet dataset comprising 1.2 million training images. During our exploratory data analysis (EDA), we decided on an input size of 56x56 pixels; ResNet-18 originally uses an input size of 256x256 pixels, but automatically resizes images to meet its input

requirements. We modified the output layer of the model to fit our 43 distinct classes. Additionally, we set a batch size of 350 for training, 35 epochs, and a learning rate of 0.001.

We applied transformations to the training dataset in the data augmentation section to improve the model's generalization abilities. Specifically, the pre-trained model uses transformations such as random horizontal flipping, resizing, and random rotation to increase the diversity of the training data. Horizontal flipping allowed us to mirror the image on the horizontal axis, while random rotation created new images with a random rotation of 45 degrees. These transformations allowed us to increase the diversity of the training dataset to help the model. Finally, our results were a **Validation Accuracy of 99.66%** and a **Test Accuracy of 96%**.

Final Results:
Our exploration into traffic sign recognition using neural network models yielded diverse outcomes. Initially, our SVM model served as a robust baseline, surprisingly effective with an 86% validation accuracy. The MLP was the worst performing model at 82% validation accuracy. Our custom CNN was the second best performer, achieving a validation accuracy of 98.76%(93.6% test), and the pre-trained ResNet-18 model, fine-tuned to our specific task, achieved the highest validation accuracy at 99.66%(96% test).

Conclusion:
Throughout our project, we encountered numerous challenges and limitations that shaped our approach and results. Accessing our images through the Google Drive environment was initially too slow, and we discovered that directly loading our database into our coding environment was significantly faster. Computational constraints limited our ability to perform a comprehensive grid search for the CNN and restricted the number of model configurations we could explore during our MLP's random search. We also faced hurdles such as data inconsistencies, class imbalances, and variations in image resolution, which we systematically addressed through dataset restructuring, class weight calculation, and image standardization, ensuring a consistent training regime across our models.

Interestingly, our trials with the MLP model illustrated that despite its limitations in processing the spatial hierarchy of image data, there is potential for performance enhancement through meticulous hyperparameter tuning. Surprisingly, the baseline SVM model outperformed even our best MLP configuration, highlighting that neural networks are not always the superior solution, especially when limited by time or computational resources.

In conclusion, we recommend leveraging the strengths of pre-trained models, particularly when computational resources are scarce, or investing in hardware capable of supporting extensive hyperparameter tuning. We believe that models such as our custom CNN and the modified pre-trained Resnet could be employed in assisted driving, for instance, providing drivers with easier access to current speed limits and highlighting warnings on the roads. Unresolved challenges such as the restrained exploration of hyperparameter space remain areas for future work, setting the stage for advancing autonomous vehicle technology and moving towards a future where safety and efficiency merge in self-driving cars.

References:

1. For this final deliverable AI tools (Gen AI) were used for grammar checks, sentence correction, and to enhance clarity and coherence of the text.
2. Chollet, Francois. “The Keras Blog.” *The Keras Blog ATOM*, blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html. Accessed 24 Apr. 2024.
3. Elhamod, Mohammad. “Transfer_Learning.Ipynb.” *GitHub*, 2024, github.com/elhamod/BA865-2024.git.
4. Mykola. “GTSRB - German Traffic Sign Recognition Benchmark.” *Kaggle*, 25 Nov. 2018, www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign.
5. Poojahira, Hiranandani. “Poojahira/Gtsrb-Pytorch: Pytorch Implementation of Kaggle GTSRB Challenge with 99.8% Accuracy.” *GitHub*, 2018, github.com/poojahira/gtsrb-pytorch.
6. Psomas, Bill. “Billpsomas/Traffic_signs_classification: German Traffic Signs Classification Using Neural Networks (MLP, Lenet, Alexnet, Vggnet, RestrictNet) in Tensorflow Framework.” *GitHub*, 2019, github.com/billpsomas/Traffic_Signs_Classification.
7. Saglani, Vatsal. “Multi-Class Image Classification Using CNN over Pytorch, and the Basics of CNN.” *Medium*, Medium, 20 Apr. 2020, thevatsalsaglani.medium.com/multi-class-image-classification-using-cnn-over-pytorch-and-the-basics-of-cnn-fdf425a11dc0.
8. Stallkamp, Johannes, et al. “German Traffic Sign Recognition Benchmark GTSRB.” *Public Archive: DAAEAC0D7CE1152AEA9B61D9F1E19370*, May 2019, sid.erda.dk/public/archives/daaeac0d7ce1152aea9b61d9f1e19370/published-archive.html
9. Tantai, Hengtao. “Use Weighted Loss Function to Solve Imbalanced Data Classification Problems.” *Medium*, Medium, 27 Feb. 2023, medium.com/@zergtant/use-weighted-loss-function-to-solve-imbalanced-data-classification-problems-749237f38b75.
10. Weights & Bias. “Sweep Configuration Structure: Weights & Biases Documentation.” *Define Sweep Configuration for Hyperparameter Tuning.*, 2024, docs.wandb.ai/guides/sweeps/define-sweep-configuration.

Appendix:

Image 1: Distribution of Images per Class

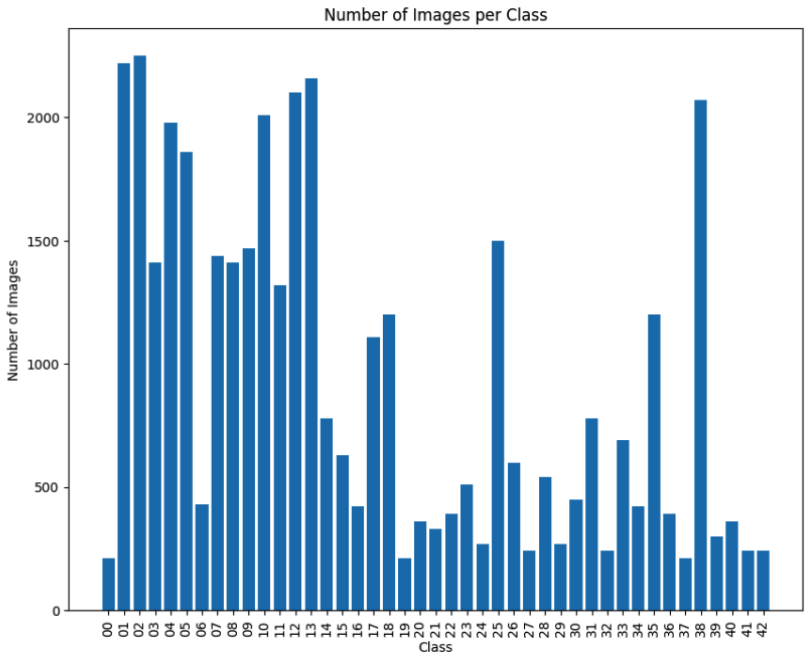


Image 2: MLP - Validation Accuracy per Epoch

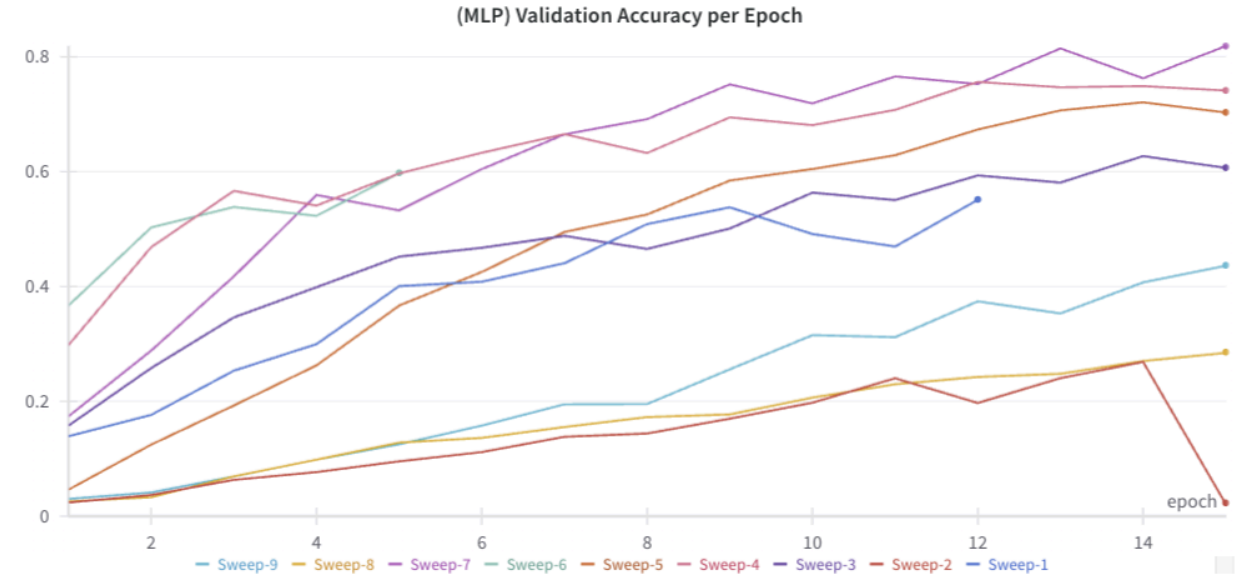


Image 3: MLP - Parameter Importance

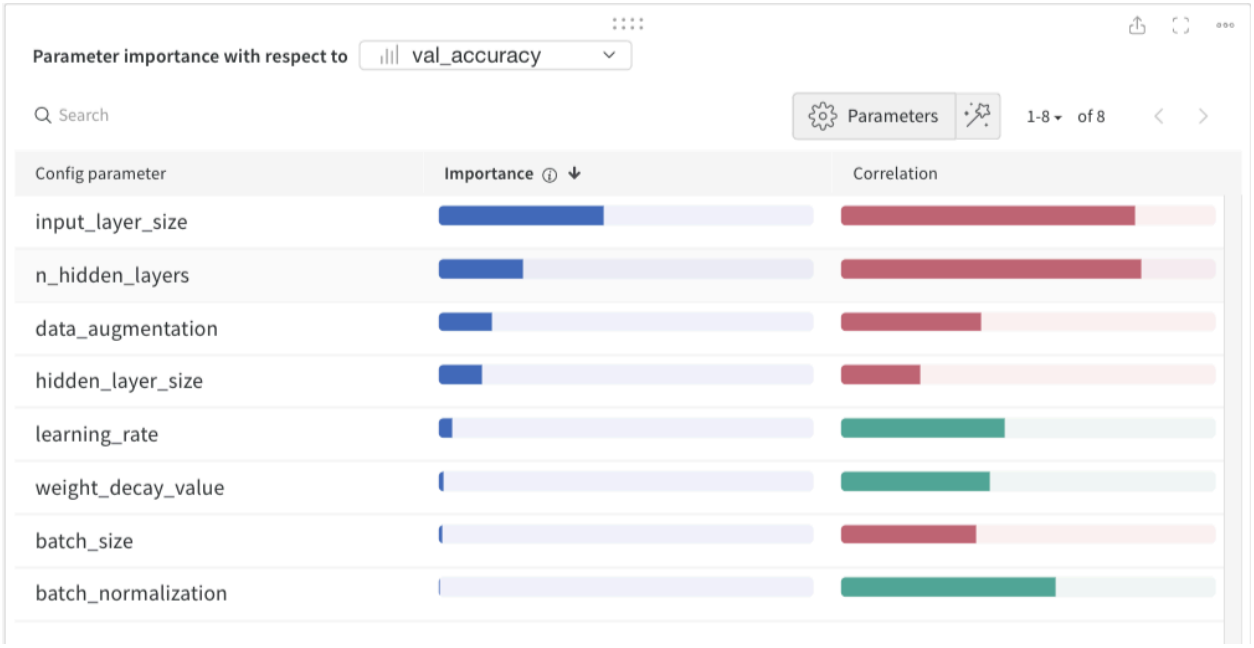
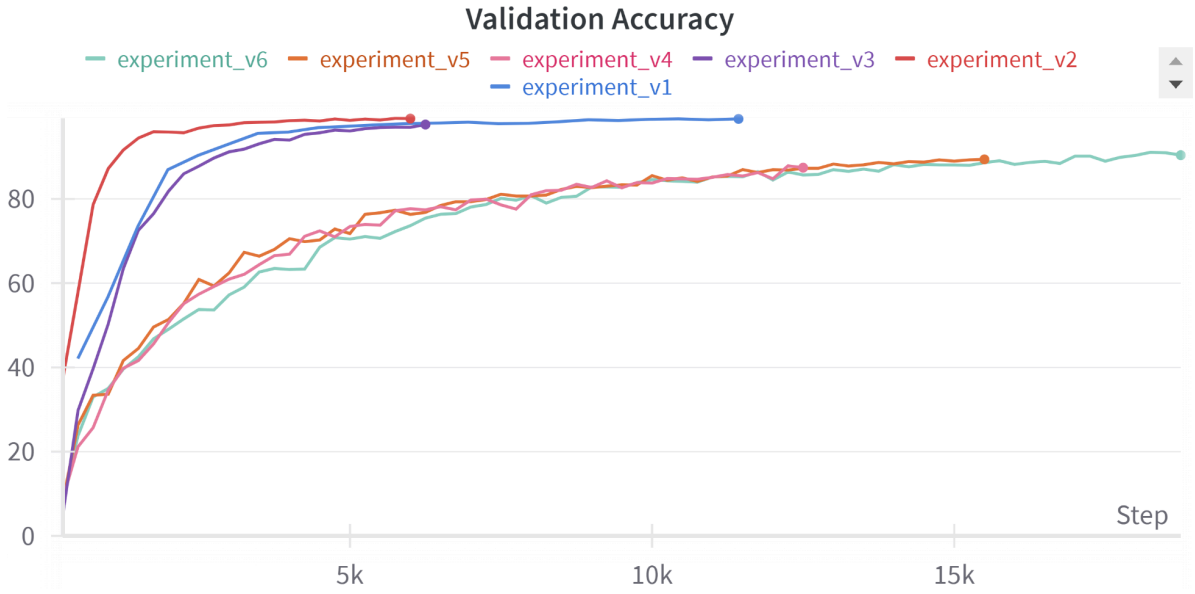


Image 4: CNN - Validation Accuracy per Batch



Breakdown of Project Notebooks:

- EDA Notebook: EDA.ipynb
- Fixing Test Dataset Notebook: Fix_Test_Data.ipynb
- Draft Models Notebook: Draft_Models.ipynb
- SVM Notebook: SVM_Baseline_Model.ipynb
- MLP Notebook: MLP_Models.ipynb
- CNN Notebook: CNN_Models.ipynb
- Resnet-18 Notebook: pre-trained_Resnet18_Model.ipynb

Github Repository Link:
https://github.com/jsale017/Deep-Learning-for-Deciphering-Traffic-Signs.git

Weight & Bias Report:

1. CNN Models: https://api.wandb.ai/links/evafan123/6qme7z4z
2. MLP Models: https://api.wandb.ai/links/victorgfloriano/p66kw1fj

Contributions:

Student Name	**Contributions**
Victor Floriano	- Performed cleaning of the Test Folder - Performed EDA on original data - Created SVM model - Fine tuned Linear MLP Model - Contributed to the Final Write up
Yifan Fan	- Created the final CNN Pytorch model - Ran continuous experiments to improve the CNN model performance - Tuned Parameters of the CNN PyTorch Model - Contributed to the Final Write up
Jose Salerno	- Created the Github Repository - Performed Class Balance - Created the MLP Linear Model on Pytorch - Created the Pre-trained model using Resnet-18 - Contributed to the Final Write up