ECEN 493R (Machine Learning) Project 3: Neural Networks Are All You Need

Arjun Singh

***Abstract*—This paper talks about Convolutional Neural Networks and is part of the course ECEn 493R at BYU in Machine Learning in the Electrical and Computer Engineering department**

I. Introduction

THIS paper is about training convolutional neural networks for traffic sign detection in self driving cars. This paper addresses a classification problem as there are only a limited number of traffic signs. Despite the pixel data of each image being continuous, the output labels are discrete and limited to a small list. The signs included in the dataset are 52 in number each in a different directory. This also makes it difficult to parse the inputs and outputs, but there is an additional data loader that I have coded that will parse each image to be used by PyTorch. The images are 32X32 pixels in size and different kernels in the CNN will be able to find different features of the picture. Some kernels may detect lines, some may detect color and other prominent patterns.

An accurate model on this dataset will help evolve self-driving cars and improve object detection across the board not only traffic signs. In addition, an accurate model will also help reduce accidents as a high percentage of cars are using cameras for assisting the drivers, even the ones without complete self-driving features. Hence, reducing accidents, among pedestrians and cars. The inputs themselves are images and will potentially help detect different features of the road signs for real time understanding. This will give any car an extra set of eyes which can potentially be even better than humans and faster than humans.

II. Procedure for Training Model

A. *Training Stage*

A.0 Data Loading and Setup

Before the training for the Convolutional Neural Network starts, A Data loader and Dataset class need to be made with loading methods to load the images from the "ppn" format to a format that can be inputted in the CNN layers.

In particular the Load Data method had two nested for loops to read each image and generate its specific label, which was depended on the folder the image was in. IN total there are 43 image types and this is given by the constant BATCH SIZE declared in the starting of the Jupyter Notebook. Some other constants are also declared in the same window at the start of the code. When looking at the Dataset class you will also find two Booleans named train and test, these are there to separate the dataset into train and test parts to be validated at the end. This also lets me build a train loader and a test loader.

The dataset class has 4 Booleans each controlling a different aspect of testing. The data is first split in 90% for final training and 10% for final testing. Then out of the 90% I split the data gain to do sweep training, and the 10% out of the final training is used for validation loss. After all the hyperparameters are swept the you can set the final Boolean high, this will merge the train and validation sets back and training the data on the total 90% of images and then finally test it on the 10% of images.

A.1 Training Steps

The training stage consists of the following steps

- Iterating over multiple epochs
- Iterate over all batches from your data loaders
- Zeroing the Gradients
- Performing a forward pass at each iteration
- Performing a backwards pass at each iteration
- Taking an optimizer step at each iteration

These steps are inside a for loop and are completed for a certain Batch Size and certain number of Epochs. I will be using two different optimizers and a number of different hyperparameters. In total the end result should produce a total of 600 neural networks.

A.2 Hyperparameter Sweep

After The Training Process is setup, I sweep over the following Hyperparameters using another for loop on the training layer. I choose three different Hyperparameter sweeps:

- Number of Epochs - [2,4,8,10,14]
- Learning Rate – [0.1,0.01,0.001,0.0001, 0.00001]
- Batch Size – [10,20,32,64,128]
- *Optimizer – ["adam", "sgd"]*
- *Dropout or Not*

A.3 Dropout

The above training steps will be consistent between the dropout and non-dropout training process. Dropout will be used as another hyperparameter, and the difference between the dropout and the non-dropout training processes will be outlined. At the end of the training, in total, there will be 600 Neural networks that will have been trained each with a different combination of hyperparameters. The results of each of the hyperparameter combinations are lined up in section B.

All of these hyperparameter sweeps produced a loss/ error in the training data which is in part C.

B. Validation Stage

Firstly, the Neural Networks trained without dropout were put against new/test data that they had never seen. The resulting graphs are found in section C. These graphs are labeled as C.WD [Without Dropout]. These graphs show the validation loss of each of the hyperparameter sweeps. For the neural network without dropout, Learning rate consistently had higher importance and had a negative correlation of negative 0.328. Hence, as the learning rate went up the loss went down. Learning rate was also impacting the neural networks a lot more than any other hyperparameter. Between the two Optimizers Stochastic Gradient Descent had lower loss overall compared to the Adam Optimizer. The minimum validation loss was 0.007923. The hyperparameter combination that produced this validation loss is shown below:

- Number of Epochs - 2
- Learning Rate – 0.00001
- Batch Size – 64
- *Optimizer – "Stochastic Gradient Descent"*

Secondly, the neural Networks trained with dropout were put up against new/test data that they had never seen. The resulting graphs are found in section C. These graphs are labelled as C.D [Dropout]. These graphs show the validation loss of each of the hyperparameter sweeps. Some interesting correlation I found when training with dropout was that as the batch size increased the loss went down. There was a negative correlation. Learning rate was still one of the most important hyperparameters and the correlation will validation loss was negative. The minimum validation loss was 0.00417. The hyperparameter combination that produced this validation loss is shown below:

- Number of Epochs - 2
- Learning Rate – 0.1
- Batch Size – 10
- *Optimizer – "Stochastic Gradient Descent"*

C. Figures

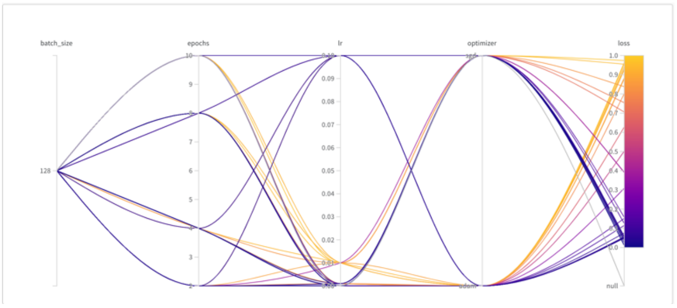

Figure 1.C.WD Batch Size = 128

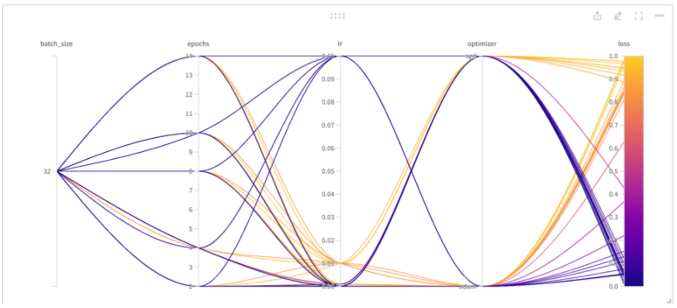

Figure 2. C.WD Batch Size = 32

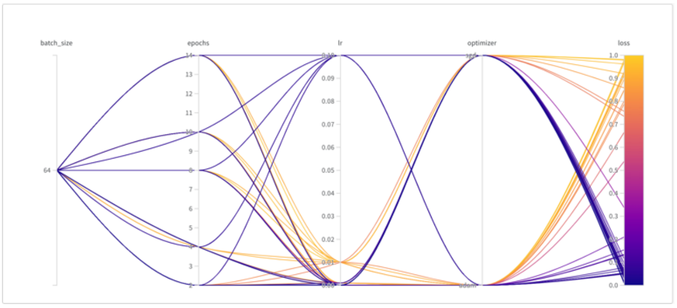

Figure 3. C.WD Batch Size = 64

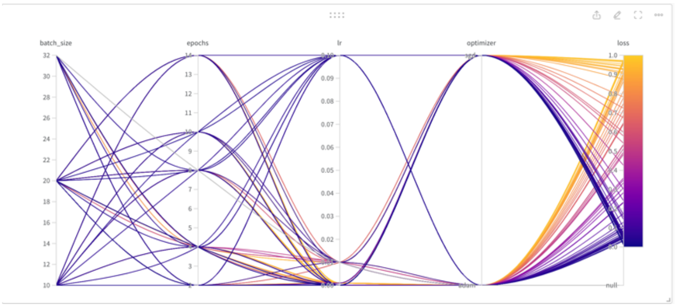

Figure 4. C.WD Ba tch Size = [10,20,32]

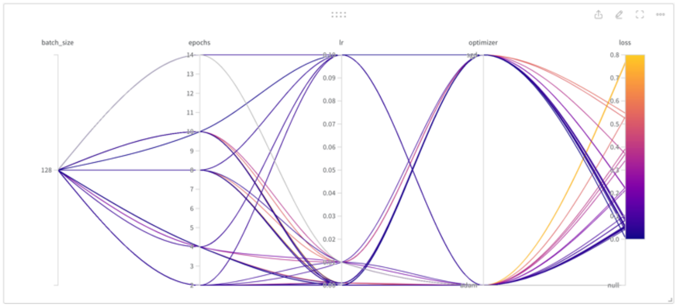

Figure 5. C.D Batch Size = 128

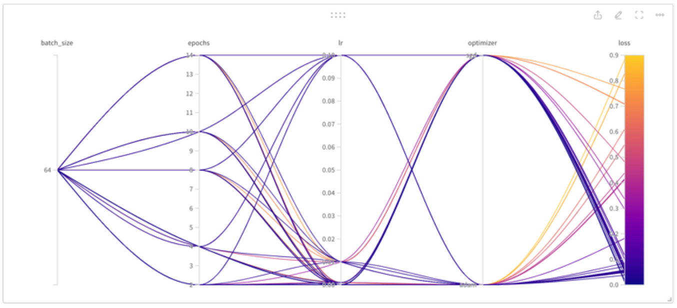

Figure 6. C.D Batch Size = 64

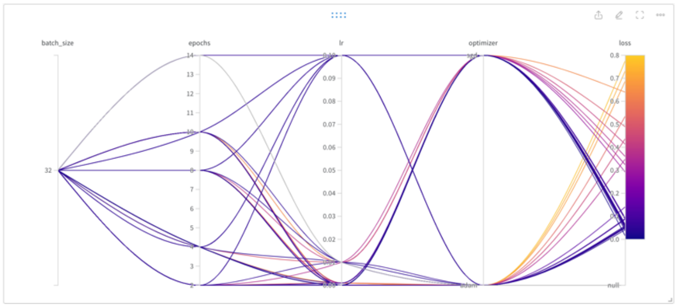

Figure 7. C.D Batch Size = 32

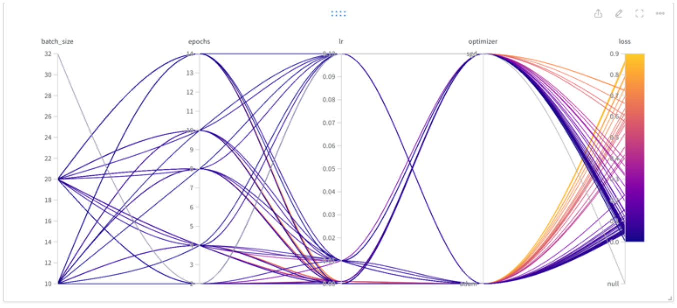

Figure 8. C.D Batch Size = [10,20]

D. Test Stage

Import your source files in one of the following: Microsoft Word, Microsoft PowerPoint, Microsoft Excel, or Portable Document Format (PDF); you will be able to submit the graphics without converting to a PS, EPS, or TIFF files. Image quality is very important to how yours graphics will reproduce. Even though we can accept graphics in many formats, we cannot improve your graphics if they are poor quality when we receive them. If your graphic looks low in quality on your

The table below shows the loss and accuracy of each model on new/test data.

Base Model	-> 0.95
Without Dropout	-> 0.04767
Dropout	-> 0.005255

E. Conclusion

The model with Dropout performed the best on the test set. Both the neural networks outperformed the baseline model, because it was a random image checker. There was 1/43 chance that the baseline model would have had chosen the correct image. To improve the paper, a much nicer and more efficient baseline model can be tested against the neural networks. Some examples are clustering and decision trees. To understand and see the differences between complexity and loss of the model. The models with Dropout generally had a smaller loss function.

In the end, however, the validation losses for both sides of the hyperparameter were significantly close. In addition, this small performance increase can be attributed to dropout and how it handles and fights overfitting in data. The learning rate for the model with the dropout was also much higher than the one without dropout. On the other, hand with a smaller batch size. This was step that needed to be taken though to get the little extra performance boost in the model. Especially when it is about life and death, every point counts.

One other reason on how it fought the outliers and was more robust. I am very confident in the model, however the model is built for image detection and not real-time image detection. Those are two different things and hence, could potentially cause some minor issues as it may not have the best performance. Ideally a model with less computation will be preferred. On top of that, this model is only for German traffic signs, so I would be comfortable putting the model out for others to test and detect different traffic signs.

F. References

[1] DATASET: https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign
[2] Tools used -> WandB API
[3] Tools used -> Google Colab

G. Abbreviations and Acronyms

Define abbreviations and acronyms the first time they are used in the text, even after they have already been defined in the abstract. Abbreviations such as IEEE, SI, ac, and dc do not have to be defined. Abbreviations that incorporate periods should not have spaces: write “C.N.R.S.,” not “C. N. R. S.” Do not use abbreviations in the title unless they are unavoidable (for example, “IEEE” in the title of this article).

H. Equations

Number equations consecutively with equation numbers in parentheses flush with the right margin, as in (1). First use the equation editor to create the equation. Then select the “Equation” markup style. Press the tab key and write the equation number in parentheses. To make your equations more compact, you may use the solidus (/), the exp function, or appropriate exponents. Use parentheses to avoid ambiguities in denominators. Punctuate equations when they are part of a sentence, as in

$$\begin{aligned}\int_0^{r_2} F(r,\varphi)\, dr\, d\varphi = [\sigma r_2 / (2\mu_0)] \\ \cdot \int_0^{\infty} \exp(-\lambda |z_j - z_i|)\, \lambda^{-1} J_1(\lambda r_2)\, J_0(\lambda r_i)\, d\lambda .\end{aligned} \tag{1}$$

Be sure that the symbols in your equation have been defined before the equation appears or immediately following. Italicize symbols (*T* might refer to temperature, but T is the unit tesla). Refer to “(1),” not “Eq. (1)” or “equation (1),” except at the beginning of a sentence: “Equation (1) is ... .”

I. Other Recommendations

III. Some Common Mistakes

The word “data” is plural, not singular. The subscript for the permeability of vacuum μ_0 is zero, not a lowercase letter “o.” The term for residual magnetization is “remanence”; the adjective is “remanent”; do not write “remnance” or “remnant.” Use the word “micrometer” instead of “micron.” A graph within a graph is an “inset,” not an “insert.” The word “alternatively” is preferred to the word “alternately” (unless you really mean something that alternates). Use the word “whereas” instead of “while” (unless you are referring to simultaneous events). Do not use the word “essentially” to mean “approximately” or “effectively.” Do not use the word “issue” as a euphemism for “problem.” When compositions are not specified, separate chemical symbols by en-dashes; for example, “NiMn” indicates the intermetallic compound $Ni_{0.5}Mn_{0.5}$ whereas “Ni–Mn” indicates an alloy of some composition Ni_xMn_{1-x}.

Be aware of the different meanings of the homophones “affect” (usually a verb) and “effect” (usually a noun), “complement” and “compliment,” “discreet” and “discrete,” “principal” (e.g., “principal investigator”) and “principle” (e.g., “principle of measurement”). Do not confuse “imply” and “infer.”

Prefixes such as “non,” “sub,” “micro,” “multi,” and “ultra” are not independent words; they should be joined to the words they modify, usually without a hyphen. There is no period after the “et” in the Latin abbreviation “*et al.*” (it is also italicized). The abbreviation “i.e.,” means “that is,” and the abbreviation “e.g.,” means “for example” (these abbreviations are not italicized).

An excellent style manual and source of information for science writers is [9]. A general IEEE style guide and an *Information for Authors* are both available at http://www.ieee.org/web/publications/authors/transjnl/index.html