Traffic Sign Classification

Thiago de Andrade
Department of Computer Science
University of Florida
Gainesville, Florida
tdeandradebezerr@ufl.edu

Yujuan Gao
Department of Computer Science
University of Florida
Gainesville, Florida
yujuan.gao@ufl.edu

Rui Guo
Department of Computer Science
University of Florida
Gainesville, Florida
rui.guo@ufl.edu

Cody Haby
Department of Aerospace Engineering
University of Florida
Gainesville, Florida
codyhaby@ufl.edu

***Abstract*—This paper details the development of a Convolutional Neural Network (CNN), a shift invariant artificial neural network (SIANN) utilizing convolution operations instead of matrix multiplication, with the goal of classifying ten unique traffic signs. A well-balanced data set of photos with equivalent resolution was used to train and validate the neural network to determine appropriate hyperparameters for optimal performance, accurate classification greater than ninety percent. The CNN was developed using packages found within the Tensorflow library in Python, including convolution, pooling, and dense layers. Additionally, this paper documents specific experiments conducted during the design and training which led to the final architecture of the neural network. The CNN will be shown to have an accuracy of greater than ninety-four (94) percent during training and validation.**

***Index Terms*—Machine learning, Classifier design and evaluation**

I. Introduction

Deep Learning algorithms have evolved in recent history to enable machines to observe their environment as humans do and utilize this knowledge to perform tasks, such as classification. A Convolutional Neural Networks (CNN) is a deep learning algorithm that can train large datasets with millions of parameters, in the form of 2D images as input, and convolve it with filters to produce the desired output [1]. CNNs are composed of convolutional layers, pooling layers, and dense layers. Convolution layers capture the spatial and temporal aspects of the input to extract high level features, with each additional/deeper layer allowing further adaptation to the high level features of the input. Pooling layers are used to reduce the spatial size of the convolved features, thus reducing the required computational power, while also extracting dominant features. Rather than requiring the user to declare the features of interest, convolutional and pooling layers work together to define, optimize, and learn the features/kernels of the input data automatically. Dense, or "Fully Connected", layers are then utilized to learn non-linear combinations of the features representing the output classification over a series of epochs with backpropagation [2].

The architectural design of this CNN was developed through experimental design performed prior to final training. Specific experiments conducted include preprocessing strategies, training data augmentation, architecture definition, and hyperparameter tuning. Further details are available in Section III.

II. Implementation

The machine learning system developed follows a basic CNN architecture. The final training model used alternating convolutional layers and pooling layers, five (5) and four (4) respectively, followed by two (2) dense layers. The convolutional layers utilized a Rectified Linear Unit (ReLU) activation function with a kernel size of thirty-two (32) for the first layer, sixty-four (64) for the following 2 layers, and one-hundred and twenty-eight (128) for the final two layers. Every pooling layers utilized "same" padding. The first dense layer used a ReLU activation with an output dimensionality of sixty-four (64) while the final dense layer, the output, used a softmax activation function with an output dimensionality of ten (10). Additionally, during training dropout regularization was used after the first set of convolutional and pooling layers with a dropout rate set to twenty percent (0.20). Adaptive Moment Optimization (Adam) was used as the optimizer in leu of gradient descent since it makes use of both momentum optimization and adaptive learning rate. Sparse categorical crossentropy was the loss function used during backpropagation to minimize the error between the actual and predicted results. The final model training used thirty (30) epochs and a batch size of thirty-two (32) to extract the high level features. The initial values of the hyperparameters for the neural network (number of hidden layers, number of neurons per layer, and the learning rate) were chosen based on literature recommendations and were then fine tuned for better performance. Unfortunately, due to issues with the University of Florida's (UF) High Performance Computing (HPC) cluster, all training/testing had to be completed locally. For this reason, typical cross-validation techniques such as GridSearchCV() and RandomizedSearchCV() could not be performed due to limited computer RAM. No attempt

was made to flag signs that were not part of the baseline ten (10) classes.

To implement the model on a test set, the trained model should be loaded and the test data passed into the model. Note, the test data should be augmented equivalently to that performed on the training set.

III. Experiments

A. Data Preprocessing

Fig. 1. Class 0-9: Stop, Yield, Red Light, Green Light, Roundabout, Right Turn Only, Do Not Enter, Crosswalk, Handicap Parking, No Parking

Several experimental designs were performed to help define and optimize the machine learning system. These experiments were performed on both the training data and on the model to increase accuracy and decreasing computational time.

Data preprocessing was performed on the initial training set of six-thousand one-hundred and ninety-five (6,195) samples which represented the ten (10) traffic sign classes shown in Fig. 1. Images that were mislabeled were corrected to represent the correct class. Images that did not correspond to any class and images that contained more than one class were removed from the training data.

B. Data Augmentation

Fig. 2. Original Image (Left), Augmented Image (Right)

The sequential class inside of the Keras application programming interface (API) was utilized to create linear layers to resize, rescale, randomly flip, randomly rotate, and randomly zoom in the order defined. The images new resized shape, 150x150, was an additional hyperparameter that was tuned during training. The red green blue (RGB) images were rescaled by two-hundred and fifty-five (255) so that all pixels are between zero (0) and one (1). The random flips were performed in both the vertical and horizontal planes. Random rotations equating to twenty (20) percent of one full rotation were performed in both the clockwise and counter-clockwise directions. Finally, both vertical and horizontal zooming, in and out, of twenty (20) percent were randomly performed on the images. Fig. 2 shows an example of an image before and after data augmentation.

TABLE I
Effects of Data Augmentation

DataAugmentation	*Training* (%)	*Testing* (%)
With	95	94
Without	90	82

The data augmentation's effect on classification accuracy was quantified by comparing the results of the CNN (architecture "A" detailed further in the following sections) when trained with and without augmented data. These results are shown in Table I. The CNN that was trained with the augmented data outperformed the CNN trained without the augmented data in both training and testing accuracy.

C. Neural Network Architecture

In the development of the neural network architecture, three different designs were compared. The first, architecture "A", is the CNN described previously in Section II and was the eventual winner with the highest accuracy. The second, architecture "B", was similar to architecture "A" with convolutional, pooling, and dense layers present. The primary difference between these two architectures was that architecture "B" stacked convolutional layers, had an additional dense layer, and used dropout regularization between each dense layer. The third architecture made use of transfer learning by utilizing the already existing Xception pre-trained model.

TABLE II
Neural Network Architecture Accuracy

Architecture	*Training* (%)	*Testing* (%)
A	95	94
B	94	92
XCeption	96	93

Table II details the accuracy in training and in testing that was achieved by each of the aforementioned neural network architectures. Based on these results, architecture "A" outperformed the other two models in testing. While the pre-trained XCeption model outperformed architecture "A" in training, the lower accuracy obtained by the pre-trained XCeption model in testing indicates it may be overfitting to the training data. Architecture "B" performed the worst despite the advantages of stacked convolutional layers and additional regularization techniques.

Fig. 3 illustrates the final architecture selected for classification task.

D. Hyperparameter Tuning

The final experiments conducted in the design of the neural network for the classification of traffic signs was the fine

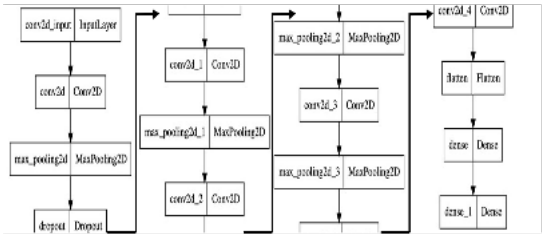

Fig. 3. Architecture "A"

tuning of hyperparameters. Hyperparameters are parameters (independent variables) whose values control the learning process and determine the values of model parameters that a learning algorithm ends up learning [3]. The hyperparameters adjusted in the design were the number of hidden layers, number of neurons per layer, and the learning rate. Additionally, the resized shape of the input pictures during data augmentation was varied.

TABLE III
HYPERPARAMATER VALUES

$HyperParamaters$	$TestRange$	$FinalValue$
Image Size	UPDATE	150x150
Number of Hidden Layers	UPDATE	UPDATE
Number of Neurons per Layer	UPDATE	UPDATE
Learning Rate	UPDATE	0.001

Table III details the range of values that each hyperparamter was tested and the final value selected.

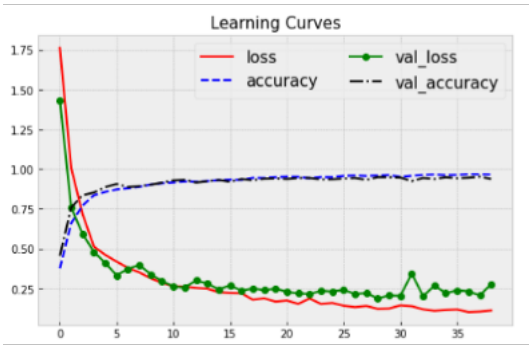

Fig. 4. Learning Curves of Architecture A

Fig. 4 shows the learning curves associated with architecture "A" during training with the final hyperparameters. The accuracy increased to 95% in training and the loss function decreased smoothly towards approximately 10%. The loss function utilized was the sparse categorical crossentropy because due to the sparse labels (i.e., for each instance, there is just a target class index, from 0 to 9 in this case), and the classes are exclusive.

IV. CONCLUSIONS

REFERENCES

[1] R. Chauhan, K. K. Ghanshala and R. C. Joshi, "Convolutional Neural Network (CNN) for Image Detection and Recognition," 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC), 2018, pp. 278-282, doi: 10.1109/ICSCCC.2018.8703316.

[2] S. Saha, "A Comprehensive Guide to Convolutional Neural Networks - the ELI5 way", unpublished

[3] K. Nyuytiymbiy, "Parameters and Hyperparameters in Machine Learning and Deep Learning", unpublished