Captcha Bots Traffic Sign Classification Project Report

Kyle Buchholz
Dept. of Mechanical and Aerospace Engineering
University of Florida
Gainesville, Florida
buchhky@ufl.edu

Conrad Hellwege
Dept. of Electrical and Computer Engineering
University of Florida
Gainesville, Florida
conrad.hellwege@ufl.edu

Phil Promsurin
Dept. of Mechanical and Aerospace Engineering
University of Florida
Gainesville, Florida
ppromsurin@ufl.edu

Alan Smith
Dept. of Mechanical and Aerospace Engineering
University of Florida
Gainesville, Florida
smithal6092@ufl.edu

***Abstract*—Quick and accurate classification of traffic signs is a fundamental prerequisite for the safe implementation of autonomous vehicles. This paper explores two classes of algorithm—Support Vector Machine (SVM) and Convolutional Neural Network (CNN)—that are promising for this type of image classification. Experimentation with dimensionality reduction and image pre-processing yielded an SVM model capable of achieving over 82% accuracy on a dataset containing 10 classes of common traffic signs and signals in the United States. Alternatively, transfer learning and hyperparameter tuning with publicly available CNN architectures yielded a model capable of achieving over 97% accuracy on the same set. The success of the CNN model emphasizes the value of this class of algorithm for traffic sign classification and related image classification tasks.**

***Index Terms*—image classification, autonomous vehicle, convolutional neural network, support vector machine**

I. Introduction

Traffic sign classification has become an important task as the automotive industry approaches the widespread integration of self-driving vehicles. Any autonomous vehicle operating on existing roadways will need to recognize traffic signs and adapt its driving behavior according to sign specifications. This recognition must occur rapidly to ensure the vehicle has adequate time to respond. One solution is to run the image data collected by the vehicle's sensors through an algorithm pre-trained on similar inputs. While model training can be extremely time-consuming and computationally expensive, testing an input against a pre-trained model is much quicker. For this reason, algorithms such as Support Vector Machines (SVMs), and Convolution Neural Networks (CNNs) have been developed to approach real-time traffic sign classification [1]. The greatest challenges faced by these algorithms are the variability of the signs themselves and the many types of noise. Signs are not standardized internationally or even within nations. They differ in color and symbols, increasing the difficulty of training a model for a generalized traffic sign classifier [2]. Therefore, to train a classifier to recognize a class with such an extreme amount of variance requires large amounts of training data.

One of the fundamental algorithms utilized in machine learning is SVM—a supervised learning model often used in regression analysis and classification. SVMs utilize kernel functions to map linear or non-linear data to higher dimensions to achieve separable decision boundaries. Decision boundaries (lines, planes, or hyperplanes) maximize the distance between data points near the decision boundary, called support vectors, in order to optimize class distinction. As only support vectors are necessary for distinction, SVMs are computationally manageable as they do not rely on the entire data set for classification. The radial basis function (RBF) is one of the most widely used kernels in SVM as it projects data to infinite dimensional space. Therefore, at the risk of overfitting, SVMs with a RBF kernel ensure separability and demonstrate high performance on large feature spaces. SVMs have previously exhibited high performance in traffic sign and general image classification, serving as a befitting algorithm for testing [3].

Another widely utilized class of algorithm is the Convolutional Neural Network (CNN). A CNN is a specific instance of an Artificial Neural Network that convolves low-dimensional kernels with input images to extract useful features. For example, a specially determined two-dimensional kernel can be convolved with an input image to map horizontal or vertical lines in the input to bright pixels in the output. With successive layers of the CNN, simple extracted features are combined to produce more complex features that may be present in a given class of image. Traffic signs are distinguishable by their geometric shape and color, features that CNNs can be designed to detect [4]. It is noted that the convolution process detects features without considering their exact location within the image. This is beneficial for traffic sign classification because the sign will not always be centrally located in the image.

Both SVM and CNN algorithms were designed and tested for the traffic sign classification task of this report. Pre-

processing experiments and significant feature reduction was conducted for the SVM algorithm to determine how certain features influenced classification accuracy. Varieties of features were selected from the input images and either appended to the original set of features or used as a standalone input for the SVM. The image filters evaluated in pre-processing experiments were as follows: Gaussian blur, Canny filter, Sobel filter, and noise reduction. CNN experiments focused on the selection of a transfer learning base model and hyperparameter tuning. The Xception and VGG16 architectures were tested for use as a base. For each architecture, randomized search cross-validation was performed to determine the optimal number of additional fully-connected layers, the number of neurons per additional layer, the learning rate, and the dropout rate.

II. Implementation

Our group's final model was constructed with standard keras and sklearn packages. The inputted 300x300x3 images undergo a slight random rotation and random horizontal flip to introduce noise that increases the robustness of the final model. The data is then scaled to range from -1 to 1 for input to the Xception base model. This model was imported without its top layer and frozen to prevent tuning of its parameters. A subsequent `GlobalAveragePooling2D` step and `Dropout` step reduce the number and size of filters feeding into an appended dense layer with 64 neurons. Another `Dropout` step is performed and then an output layer with 10 neurons (one per class) utilizing the softmax activation function is added as the new top layer.

The model was trained over 100 epochs using cross validation across the number of dense hidden layers to append, the number of neurons per dense layer, the learning rate, and the dropout rate (see Experiments section). The best model was saved and briefly retrained (20 epochs) with the Xception base unfrozen and a very small learning rate (10^{-5}).

For the extra credit competition, our team devised a threshold strategy based on the open-set classification literature [5]. When running images through the trained model described above, if the probability of an inputted image belonging to one of the original 10 classes does not exceed 0.80, the image is assigned to the 11th class. This strategy was simple to implement and avoided the computational strain that would have been imposed by retraining the model with a supplemented data array with random object images.

III. Experiments

In working towards a final algorithm, our team broke into two groups: one focusing on the use of SVMs with various image pre-processing and the other on transfer learned CNNs with limited pre-processing and greater hyperparameter tuning.

A. SVM

The SVM group selected SVM as an optimal classifier following experimentation with fundamental algorithms used in machine learning. Principal Component Analysis (PCA) was performed and SVM, Naive-Bayes, k-Means, and kNN classifiers were trained and evaluated. The SVM performed best, decisively, with a test accuracy of 68.3% (see Table I).

TABLE I
Fundamental Classifiers: Comparison

Accuracy	kNN	k-Means	Naive-Bayes	SVM
Training	54.1%	12.2%	47.7%	100.0%
Testing	55.9%	9.7%	39.8%	68.3%

While SVM performs well with large feature-space datasets, reducing computational complexity took priority during experimentation. Significant feature reduction was necessary to form a base input that could be fed into the algorithms without exceeding HiPerGator memory allocations. The first technique utilized was image compression whereby the original image (Fig. 1: Left) was compressed from 300x300 pixels to 50x50 pixels (per color channel). This left 7,500 remaining features (reduction factor of 36). Image resolution was reduced, but dominant colors in each region of the image remained, and color is one of the most useful features for traffic sign classification [6]. Next, the files were converted to gray scale to reduce images down to a single channel, leaving 2,500 features (Fig. 1: Right). Following this step, PCA was performed. PCA is a dimensionality-reduction technique which finds the vectors along which there is the most explained variance in the data. In implementing PCA, it was determined that 115 features are necessary to preserve 90% of the explained variance in the dataset (i.e., 115 is the minimum number of features which can sufficiently serve as a basis for differentiating the ten classes). To conclude the base-level SVM feature processing, the SVM hyperparameters C and gamma were optimized through a randomized search and cross-validation with an RBF SVM kernel. C is the regularization parameter that governs the trade-off between some misclassification (underfitting) and no misclassification (overfitting) during training. Gamma defines the spread or radius of the decision region for the kernel, where low values result in broad decision regions and vice versa for high values. The optimal values for C and gamma were determined to be 8.9 and 5.8e-8, respectively.

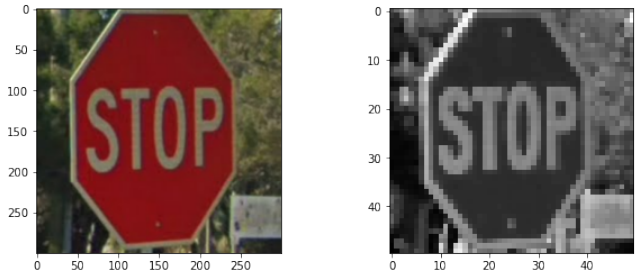

Fig. 1. Left: unaltered image sample (300x300 image size, 3 color channels). Right: compressed and grayscaled image sample (50x50 image size, 1 channel).

Following base dimensionality reduction experimentation, pre-process experimentation, utilizing a variety of additional image filters, was conducted. Such methods included image

normalization (Gaussian Blur), edge detectors (Canny and Sobel filters), and noise reduction. The goal was to normalize the images at the input and apply a sequence of filters to emphasize the differences among the signs while reducing the impact of background noise and color variance. The difficulty in this pre-processing approach was finding the proper balance between adding and removing features from the input. In the case of edge detectors, the visual distinction was clear for a human viewer (Fig. 2); however, the detectors, used independently, reduced useful information from the model. The edge detectors limited color variance of the signs and failed to promote features for out-of-focus signs, resulting in poor classification. Similarly, noise reduction resulted in an insignificant change to performance or blurred images to the point of extremely poor performance. Optimal hyperparameter selection was beneficial to all models, yet imposed computational strain, of which was already significant. The accuracy results of each experiment are provided in Table II.

TABLE II
SVM FEATURE PROCESSING: COMPARISON

Accuracy	Feat. Red.	Canny	Sobel	Noise	Gauss. Blur
Train	100.0%	85.3%	70.2%	100.0%	28.5%
Test	82.5%	58.2%	45.3%	72.4%	22.5%

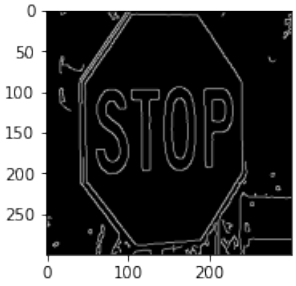

Fig. 2. Sample image pre-processed with Canny filter.

B. CNN

The CNN group only applied minor augmentation before inputting training traffic sign images into the architecture. CNNs are designed to use matrices acting like traditional image filters (e.g., edge detectors, colormaps) to extract useful features from raw images [7]. Therefore, extensive pre-processing techniques like PCA or Fisher's LDA may destroy the spatial features from which CNNs derive their success. Other than scaling the data to range from -1 to 1, only slight random rotations and horizontal flips were applied to the data prior to input. These augmentation practices make the model more robust by improving its ability to identify angled test images [8]. They also enable the training dataset to expand by using variations of the actual collected images (this practice was not employed in the training of the final model for this report but can be helpful when there is clearly insufficient training data).

The augmented image inputs were passed to models built upon one of two base architectures: Xception and VGG16. Xception, designed by Francois Chollet, was the first architecture to use depthwise separable convolutions as a means to efficiently implement a deep neural network (DNN) [9]. Since Xception outperforms the other available keras models in both top-1 and top-5 accuracy on the versatile ImageNet dataset, it was a promising foundation for this project's traffic sign classifier [11]. VGG16 was also evaluated given its success in classifying the ImageNet data [10]; however, the depth and number of dense layers of the VGG16 architecture made it difficult to work with (i.e., training time was long and the kernel often died during HiperGator sessions).

Subsequent experimentation revolved around striking a balance between improving computational time and appending layers to the frozen base. An early test proved that a `GlobalAveragePooling2D` and `Dropout` operation could be performed on the base model without impacting performance. New dense hidden layers could then be added to create the relationships among base features necessary to classify the traffic signs. First tests were performed with a range of 0-10 additional layers; however, it quickly became apparent that architectures with fewer layers (0-2) achieved the same performance as more complex architectures and had less computational expense. The training script cross validation only tests 0, 1, and 2 as values for the `n_hidden` hyperparameter for this reason. The number of neurons within each of these dense layers was also tuned as a hyperparameter (`n_neurons`) in the range 1-100. Larger numbers of neurons caused computational errors in HiPerGator or were not selected by the cross validation scheme.

A `Dropout` operation was encoded following the addition of each of the new dense layers to the architecture; however, instead of setting the dropout rate to the standard 0.5, an experiment was conducted to determine the optimal rate. Lower dropout rates were found to deliver slightly better accuracy for the end model without significantly increasing computational time, so the training script only tests the range 0.15–0.3 in increments of 0.05 during cross-validation. These low rates enable the most redundant features to be discarded but preserve the majority for input to the critical output layer.

Learning rates were sampled from a reciprocal distribution with parameters $3\text{x}10^{-4}$ and $3\text{x}10^{-2}$ during cross-validation. The tested values ended up ranging between 10^{-4} and 10^{-2}. The learning rates that produced the most accurate models were small (on the order of 10^{-4}).

The outcome of the experiments discussed above was an Xception-based architecture that achieved an accuracy of about 93% with the test set and a VGG16-based architecture that achieved an accuracy of about 91%. Our group moved forward with the Xception-based architecture and decided to revisit the parameters of the previously frozen Xception layers. Using a small learning rate (1e-5), these parameters were tuned with a short training period (maximum 20 epochs). The result was the final model which achieved an accuracy of 97% on the augmented test set. Figure 3 depicts Francois Chollet's original

Xception architecture and the additional dense hidden layer and output layer added for the traffic sign classification task.

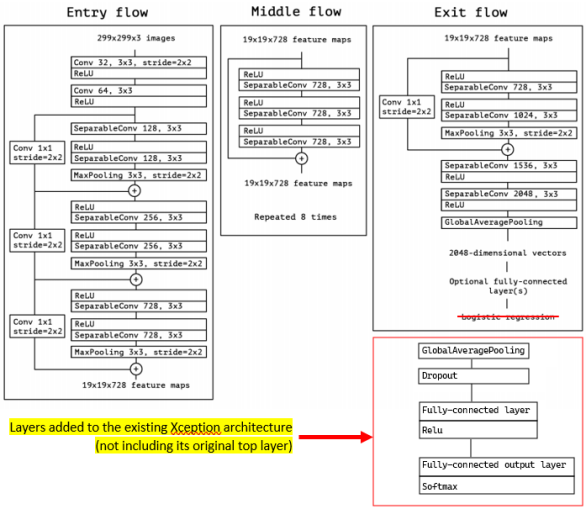

Fig. 3. Xception model architecture with an additional GlobalAveragePooling2D step, Dropout step, and appended dense layer and output layer. Base architecture image from Francois Chollet's original paper on Xception[9].

Conclusion

For this project, we were tasked with classifying 300x300 RGB images into ten distinct classes of traffic signs. 70% of the original dataset images were used for training and cross validation and 30% were used for testing of the trained image classifiers. Our group saw promising performance with both Support Vector Machines and Convolution Neural Networks and both of these models were the foundational elements of the classifier pipelines outlined in this report.

The HiPerGator supercomputer was an invaluable resource for rapid prototyping of classification models and techniques. The resources required exceeded the computational power of an average computer and without the HiPerGator, success would have been hindered. The significant hurdles associated with the supercomputer primarily arose from the lack of pre-allocated resources. When working with our training data set, both our SVM and CNN classifier implementations experienced dying kernels. Machine learning is an industry in which computing resources are the limiting factor and with the resources we were allotted, this was apparent.

The primary challenges with SVM involved selecting appropriate features without exceeding computational limits or removing critical features. A significant hurdle to the preprocessing approach was the ever increasing feature array with each applied image filter. Using filters like Canny and Sobel increased our feature arrays by a factor equal to the number of input images. In order to preserve the spatial positioning of each filters output, these arrays could not be altered any differently than our input images. SVM was extremely burdened by large feature arrays and, in order to produce a model that could run in a realistic time frame, feature reduction techniques like PCA and gray scale were necessary. As the number of filters utilized in the model's pipeline increased, the computation time exceeded reasonable limits. Additionally, each filter increased the number of hyperparameters to search through and thus increased the difficulty in finding an optimal solution. Finally, critical features such as color and object distinction were also impacted by the specified filters, due to the nature of the filter or insufficient hyperparameter tuning.

While inferior to the CNN model, the best SVM model compressed the input image by a factor of 36, applied the gray scale image filter, and used PCA to extract 115 elements from the processed feature array. This model, based on general feature reduction principals, performed well with 82.5% accuracy while maintaining low computational expense and thus allowed for an exhaustive search of hyperparameters.

The final Xception-based CNN classifier was the best performing model of this project with an associated accuracy of up to 97% on an augmented (rotated and flipped) test set and up to 99% on a standard test set. The Xception model was trained with about 1 million images and learned to detect features inherent in the vast majority of image classes. Therefore, the addition of a few new dense layers and an output layer was sufficient to achieve the high accuracy figures stated above. As highly refined, robust models can quickly be leveraged for related tasks, transfer learning continues to be an extremely valuable technique for image processing.

References

[1] Y. Yang, H. Luo, H. Xu and F. Wu, "Towards Real-Time Traffic Sign Detection and Classification," in IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 7, pp. 2022-2031, July 2016

[2] M. Swathi and K. V. Suresh, "Automatic traffic sign detection and recognition: A review," 2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET), 2017, pp. 1-6

[3] R. Kumaraswamy, L.V. Prabhu, K. Suchithra and P.S.S. Pai, "SVM Based Classification of Traffic Signs for Realtime Embedded Platform," Communications in Computer and Information Science (CCIS), vol. 193, 2011

[4] Palak and A. L. Sangal, "Traffic Signs Classification using Convolutional Neural Networks: A review," 2021 2nd International Conference on Secure Cyber Computing and Communications (ICSCCC), 2021, pp. 450-455

[5] A. Bendale and T. E. Boult, "Towards Open Set Deep Networks," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 1563-1572

[6] M. A. A. Sheikh, A. Kole and T. Maity, "Traffic sign detection and classification using colour feature and neural network," 2016 International Conference on Intelligent Control Power and Instrumentation (ICICPI), 2016, pp. 307-311

[7] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1-6

[8] T. A. Korzhebin and A. D. Egorov, "Comparison of Combinations of Data Augmentation Methods and Transfer Learning Strategies in Image Classification Used in Convolution Deep Neural Networks," 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus), 2021, pp. 479-482

[9] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1800-1807

[10] "Keras Applications." keras.io. https://keras.io/api/applications/ (accessed Aug. 1, 2022)

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," ICLR 2015