Automated Road Asset Data Collection and Classification using Consumer Dashcams

Michael Sieverts[1], Yoshihiro Obata[1], Mohammad Farhadmanesh[1], David Sacharny[2], and Thomas C. Henderson[1]

***Abstract*— A growing remote sensing network comprised of consumer dashcams presents Departments of Transportation (DOTs) worldwide with opportunities to dramatically reduce the costs and effort associated with monitoring and maintaining hundreds of thousands of sign assets on public roadways. However, many technical challenges confront the applications and technologies that will enable this transformation of roadway maintenance. This paper highlights an efficient approach to the problem of detection and classification of more than 600 classes of traffic signs in the United States, as defined in the Manual on Uniform Traffic Control Devices (MUTCD). Given the variability of specifications and the quality of images and metadata collected from consumer dashcams, a deep learning approach offers an efficient development tool to small organizations that want to leverage this data type for detection and classification. This paper presents a two-step process, a detection network that locates signs in dashcam images and a classification network that first extracts the bounding box from the previous detection to assign a specific sign class from over 600 classes of signs. The detection network is trained using labeled data from dashcams in Nashville, Tennessee, and a combination of real and synthetic data is used to train the classification network. The architecture presented here was applied to real-world image data provided by the Utah Department of Transportation and Blyncsy, Inc., and achieved modest results (test accuracy of 0.47) with a relatively low development time.**

I. INTRODUCTION

Traffic signs are an integral component of ensuring safety for all drivers. Different types of signs communicate unique messages to drivers; the signs are grouped into three main categories: warning, regulatory, and guide signs. These signs warn drivers of upcoming road conditions, regulate laws on the road, and provide information about nearby cities, food, and attractions. For drivers, traffic signs create a safe and fast-flowing driving environment. For departments of transportation (DOTs), knowing the location, the type, and the state of repair of these signs is essential for inventory and maintenance purposes.

While access to information about traffic signs is helpful to DOTs, cataloging the data related to sign location is tedious, costly, and recorded infrequently. Sign locations do not often change, but new signs are added as cities grow and temporary signs are placed in construction zones. Additionally, traffic signs are only cataloged and updated once every two years, so temporary signs and signs in disrepair would not be accounted for in a timely fashion. When signs are cataloged, it is time-consuming to classify them by hand due to the many unique signs found on roadways. An automated approach for determining the location and the type of signs in near-real time would be valuable for DOTs across the United States and worldwide.

Detecting and classifying traffic signs is a complex problem to automate due to many sign classifications and the variation in sign appearance. According to the Manual on Uniform Traffic Control Devices (MUTCD) from the US DOT Federal Highway Administration (FHWA), there are over 600 different types of signs on US roads [1]. In addition to many different signs, the traffic sign's appearance can vary due to the sign's content or the conditions of the image of the sign. These challenges add complexity to sign detection and classification; however, deep learning approaches show promise in overcoming these challenges.

Deep learning methods have had great success solving problems like object detection and classification. Useful technologies exist to leverage deep learning methods quickly. There are robust frameworks for performing deep learning like Pytorch [2] and Tensorflow [3]. Pre-trained models can also be used for transfer learning or fine-tuning a well-trained general model to accomplish a specific task. These technologies allow for the accelerated development of deep learning solutions.

This project aims to create a deep learning pipeline for detecting and classifying traffic signs from dashcam images in collaboration with Blyncsy, Inc. In this work, we trained a model using a training dataset of dashcam images of Tennessee roads and applied it to detect signs in dashcam images of Utah roads. Due to insufficient examples for image classification, we created synthetic data to supplement actual sign data. We used a two-stage architecture where we trained one model for sign detection using real-world image data, and we trained another model separately for sign classification using a combination of synthetic and real-world sign images. This approach achieved promising results with a relatively low development effort.

II. PREVIOUS WORK

Recently, a real-world European dataset for traffic sign classification was introduced with 80,000 images containing 164 classes from six European countries [4]. This dataset is focused on the classification of traffic signs and cannot be used to evaluate detection models. Other previous works almost unanimously used a publicly available traffic sign

[1]University of Utah, Salt Lake City, UT tch@cs.utah.edu

[2]Chief Technology Officer, Blyncsy, Inc., 175 W 200 S STE 1000, Salt Lake City, UT david.sacharny@blyncsy.com

dataset named German Traffic Sign Recognition Benchmark (GTSRB), with 43 classes developed for detection and classification [5]. There are numerous approaches developed for increased accuracy and speed [6]–[11].

Our study aimed to detect and classify traffic signs commonly used in the United States as defined by FHWA. These traffic signs include three main categories, regulatory, warning, and guide, with over 600 classes. This task presents some unique challenges, such as many classes compared to the existing benchmark datasets and the amount of variation in sign shape and color. Our solution to these challenges was to employ deep learning models to learn the detection and classification tasks.

Deep learning model development requires a large number of images per class for the training data. However, there is not a publicly available dataset for all traffic signs used on US roadways. As a result, we synthesized a traffic sign dataset using the existing design-level templates to prepare a set of annotated data. This approach differs from the previous attempts to produce realistic synthesized training pictures using actual traffic sign images. Dewi et al. [12], [13] applied generative adversarial networks (GANs) such as DCGAN, LSGAN, and WGAN to actual traffic sign pictures to create synthetic images of those traffic signs, focusing on no entry, no stopping, no parking, and speed limit signs (all a circle shape). The most similar works to ours are those efforts made to apply a series of transformations to the pictograms of the traffic signs [14]–[20]. This research generally focuses on a small set of traffic sign classes, such as rare classes [21], so they do not prove their effectiveness on FHWA-defined traffic signs with more than 600 different classes.

III. MATERIALS & METHODS

A. Datasets and resources

The training dataset for sign detection consisted of manually labeled dashcam footage from cars obtained from the Tennessee Department of Transportation (TDOT). Images were provided by Blyncsy, Inc. and manually annotated with bounding boxes and classification labels according to the MUTCD [1] for warning, regulatory, and guide signs. The manually labeled dataset consisted of 620 labeled dashcam images with 1578 labeled signs from 98 different classes. Of the 98 classes represented in the labeled data, the median number of signs in a class was 4. In the MUTCD pdfs, there are over 600 categories of signs. Since our labeled data contained few examples of only 98 types of signs, we supplemented the real-world data with synthetic data to perform the classification task.

B. Synthetic sign data created to supplement sign classification task

Synthetic data were created using images from the MUTCD pdfs. First, example images of each class were prepared from the pdf by deleting the annotations, masking the sign, and cropping the image. These example images were saved in the PNG format to preserve the mask information in the alpha channel. These images were then processed into synthetic data following two approaches. The simpler approach included placing the image on a randomly chosen background of black, white, or noise, adding randomly weighted noise and Gaussian blur to the image, and then randomly resizing the image. The more involved approach included randomly skewing the image, and its mask, randomly adding noise and Gaussian blur to the image, randomly resizing the image, randomly placing the image on a dashcam image that did not contain any signs, and allowing a random amount of the background to show through the sign to add texture. We generated approximately 385,000 synthetic images for training using the two methods described. Before classification, all images were resized to 64×64 to enable easy image batching.

C. Sign detection using transfer learning with a Faster R-CNN model

Traffic signs were detected in dashcam images using transfer learning. The models used for this task were all pre-trained on COCO [22] and accessed through Pytorch's torchvision package [2]. These models were adjusted to only have two classes, background and sign, and trained using the composite loss function in pycocotools [23]. Three Faster R-CNN network backbones were selected to test, ResNet-50-FPN, MobileNetV3-Large FPN, and MobileNetV3-Large FPN for mobile platforms. These models were trained and validated using the TDOT dashcam image dataset. Each model was trained for five epochs with the optimizer Adam [24] using a learning rate of 0.0001 and a batch size of 4. After five epochs, all models' validation performance was compared to select an appropriate model for detection. Once the different architectures had been compared and the hyperparameters had been tuned, the final model selected for sign detection was the Faster R-CNN model with the ResNet-50-FPN backbone. This model was trained for 30 epochs using Adam [24] as the optimizer, a learning rate of 0.0001, and a batch size of 4. The model weights were stored at each epoch where there was an improvement in the validation loss. These weights were then loaded into the final model for prediction.

D. Sign classification using transfer learning

Traffic signs were classified into their MUTCD categories using transfer learning. The models used for this task were all pre-trained on ImageNet [25] and accessed through Pytorch's torchvision package [2]. When fine-tuning these models, the final layer in the model architecture was adjusted from 1000 classes to 608 classes to reflect the number of classes in our MUTCD signs dataset. All models were trained using cross-entropy as the loss function because the task was a multi-class classification. The models were trained using primarily synthetic data, with some real examples included where possible. The performance of six different model architectures was tested to determine an appropriate model architecture for the sign classification task. The model architectures included SqueezeNet, MobileNet, EfficientNet, ResNet, ResNext, and Wide-ResNet [26]–[31]. The validation accuracy of these

models was recorded after allowing them to train for five epochs to assist in model selection. The model was selected based on its ability to achieve high validation accuracy with fewer parameters to mitigate overfitting. After testing the different architectures and tuning hyperparameters, the final model used was an EfficientNet [28] with Adam as the optimizer [24] a learning rate of 0.001 trained over ten epochs with a batch size of 256. During training, the weights were stored at each epoch when there was an improvement in validation loss.

IV. RESULTS

A. Sign detection has success with a small training dataset

The model selected for sign detection was the Faster R-CNN model with the ResNet-50-FPN backbone. The ResNet-50-FPN backbone outperformed the MobileNet backbones on validation accuracy for each bounding box size. This performance is expected because the ResNet-50-FPN backbone has the most trainable parameters of the backbones tested. Despite the relatively small amount of training data, the network trained for sign detection had high accuracies. We observed that small signs are not well detected, with only a peak accuracy of about 25%. As bounding box size increases to medium and large, accuracy increases. Intuitively, this trend makes sense because as the bounding box size increases, more image information of the sign is available for the network to use to classify. Medium size bounding boxes had an accuracy of about 58%, and large bounding boxes had an accuracy of about 75% on the validation dataset. The $F_{0.5}$ scores of the network during these bouts of training were very similar (Fig. 1). These scores are slightly higher than the training accuracies, indicating that the network does not sacrifice accuracy for recall.

B. Sign classification has success using synthetic data

The model selected for the classification task was EfficientNet B0 [28]. This network was chosen because it achieved high validation accuracy with fewer trainable parameters than the other tested models. The EfficientNet B0 architecture balanced high validation accuracy with a few parameters. Training the EffiencentNet architecture for ten epochs resulted in high validation accuracy. Three instances of the EfficeintNet network were trained for ten epochs to analyze performance better. The model with the highest validation accuracy reached an accuracy of 0.994. This network was selected for use in the final product. The model's accuracy on the UDOT test dataset was lower than the accuracy on the validation set, with a test accuracy of 0.470. To further characterize the model's performance on the test set, we implemented a top k prediction accuracy. This approach shows if the correct label is found in the top k predictions (Fig. 2). This analysis showed that the model accuracy improved quickly with the top k predictions. When k=10, the model achieved an accuracy of 0.73.

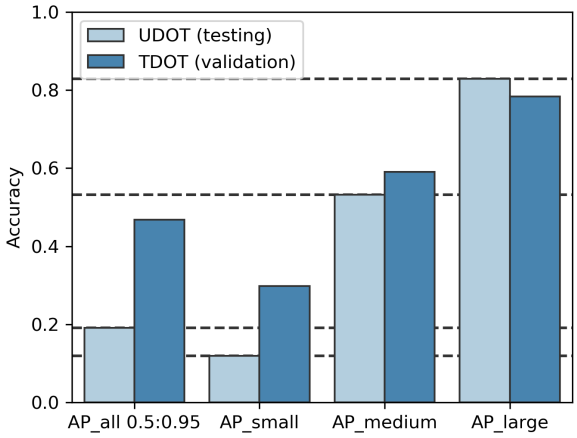

Fig. 1. Comparison of detection accuracies of the TDOT validation dataset and UDOT test dataset. TDOT validation performed on the highest-accuracy epoch is shown in dark blue. After training, the network was applied to the UDOT dashcam dataset, shown in light blue. While overall sign detection accuracy is lower than the training accuracy, the network generalizes to images from another state quite well, as shown in the medium and large size sign accuracy. The network even performs more accurately on large signs from the UDOT testing dataset.

C. Final product detects and classifies MUTCD signs

The final product combines the Faster R-CNN detection network and EfficentNet classification network to detect and classify MUTCD signs. We set thresholds for both the detection and classification networks to avoid false positives. The thresholds for detection included a minimum score of 0.7, a minimum bounding box side of 15 pixels, and a minimum aspect ratio of 0.1. For classification, a minimum score threshold of 0.45 was set. An example of the final model's performance can be seen in Fig. 3.

V. DISCUSSION

The Faster R-CNN network architecture with the ResNet-50-FPN backbone achieved good detection results with limited training data [32]. The success of this model stems from the pre-trained weights used as a starting point for training. These initial weights are already well situated for general object detection tasks. By fine-tuning this model with the small set of traffic sign training data, the model successfully learned to detect signs in dashcam images. Another factor that could contribute to this model's success is that the signs generally contrasted well with their surroundings. During qualitative observations of the model's performance, we saw that the model generally excluded objects that could be mistaken as signs like billboards or the sides of trucks. However, there were instances where the model would misclassify objects as a sign, such as traffic lights. These mistakes are likely due to sparse examples of intersections with traffic lights in our training data. With additional training data, this model would likely successfully learn to exclude objects that are not signs.

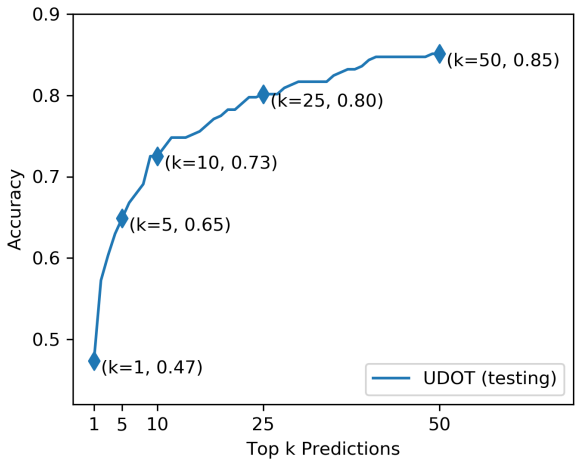

Fig. 2. Classification accuracy on the UDOT test dataset for the top k predictions. For example, at k=5 the accuracy 0.65 reflects the true label being in the top 5 predictions. The accuracy of a single prediction is lower (0.47), but the accuracy quickly increases with k.

The EfficientNet achieved good classification results using the synthetic data. The EfficientNet model showed promise of performing the classification task well by quickly achieving high validation accuracy during training. However, when we applied the trained model to the hold-out test data, we observed much lower accuracy than what was observed on the validation data. This reduced performance on the test dataset is likely due to the model overfitting on the training data. Since most training data was synthetic, most of the training images were very similar to one another. These similarities likely allowed the model to memorize aspects of the training data instead of learning more meaningful representations of the classes. Overfitting was not immediately detected during training because the model achieved high validation accuracy. This high validation accuracy was likely a result of data leakage, or information being shared between training and validation data. Because most of the training and validation datasets consisted of synthetic images, the images in both datasets were very similar. In future work, the input data should be supplemented with as much real-world input data as possible, and the synthesis of synthetic data should mimic real-world images more than current methods. Despite the model overfitting to the synthetic data, the model still provides valuable classifications of real-world data. The accuracy of the model on the test data was 0.47, which is somewhat low; still, if we look for the correct class being in the top ten predictions, then the accuracy rises to 0.73 (Fig. 2), which is very respectable considering the sizable number of potential classes in this task. While looking at the top ten predictions would not work for the final application of automatic sign classification, these predictions could be used to assist in labeling real-world data to improve the model's performance.

The use of synthetic data allowed us to build a functioning classification network with minimal data labeling. After labeling the data, many unrepresented classes of signs and few examples in the represented classes made deep learning impossible. We mitigated this challenge using synthetic data to balance the classes and increase the number of examples in each class. These synthetically generated sign images allowed us to train a classification model; however, there were limitations in training on the synthetic data. One of these limitations is that the synthetic signs all contained the same text information. For example, all synthetic speed limit signs showed a limit of 50 MPH, all interstate route signs showed interstate 20, and all hill warning signs warned of an 8% grade. We expect that the example signs containing the same text data hindered how well the model generalizes to real-world examples; however, we observed instances of correct sign classifications that had different text than the template examples. Another limitation of the synthetic data is that the synthetic images look a bit unnatural. One of the main contributing factors to this abnormal appearance was our inability to apply the natural lighting conditions in real images. We tried to overcome this issue by applying a gamma correction on the images and allowing some background to show through the signs. However, the synthetic data still had a distinct appearance compared to actual images of signs. This challenge could be overcome with more resources using a generative adversarial network. Despite these limitations, the classification network could still achieve good classification results on real-world images.

VI. CONCLUSIONS AND FUTURE WORK

In conclusion, using a small amount of labeled data supplemented with synthetic data, we created a two-stage network that performs well at both sign detection and classification for many prominent sign classes. While the performance of this network is modest, it could be used to assist in labeling data to train a more robust model in the future. This work highlights the strength of transfer learning to create valuable models with relatively low development time.

In future work, we intend to work to increase the performance of the system developed here. As part of that, single-unit ablation studies can be performed to determine the impact on the variety of sign classes. There is some evidence [33] that ablation of specific units may increase the classification performance for some of the sign classes.

ACKNOWLEDGMENT

This work was supported in part by Blyncsy, Inc. The opinions and findings of the authors do not necessarily reflect the view and opinions of Blyncsy, Inc. In addition, we would like to thank the State of Utah for the computational and other resources provided for the Deep Learning in AI and Robotics program.

REFERENCES

[1] W. Stout, "Manual on Uniform Traffic Control Devices–Signs," 2015.

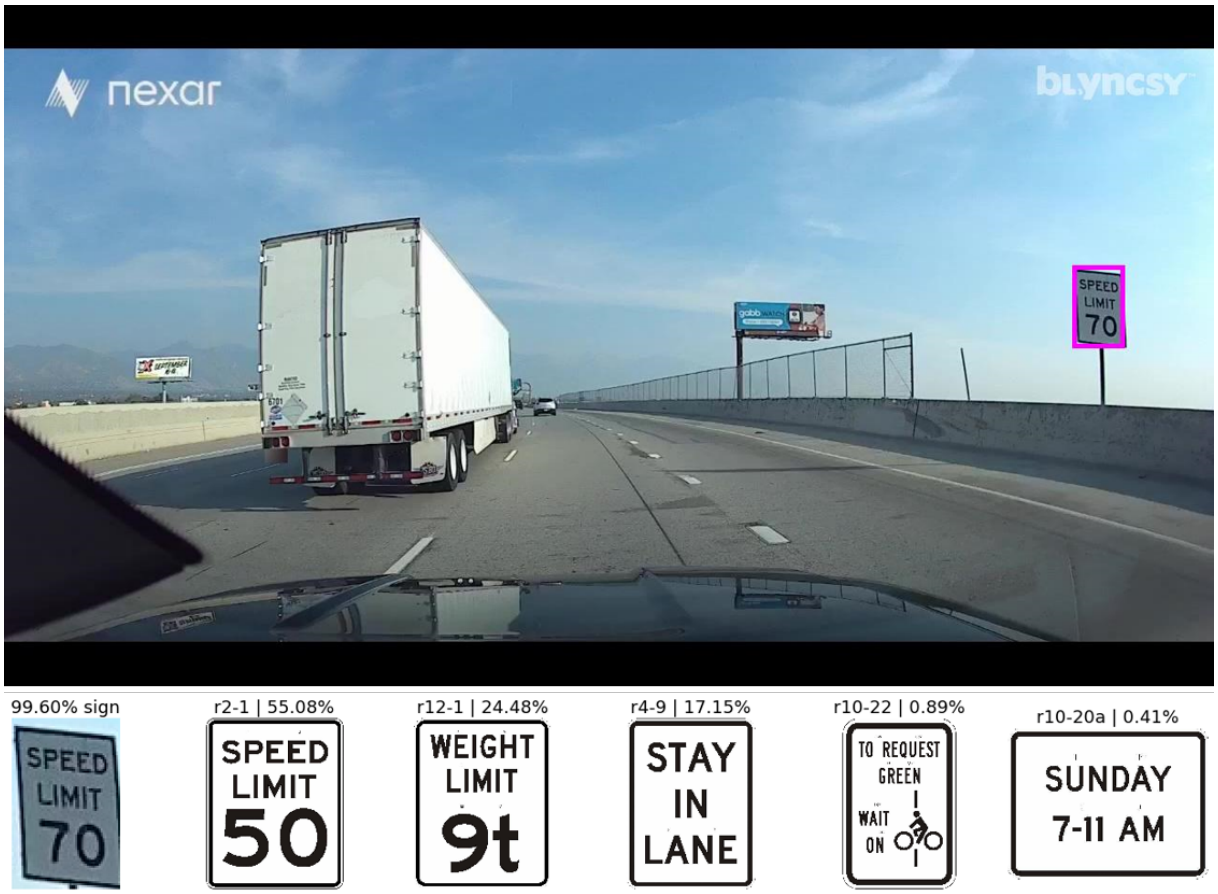

Fig. 3. Example of final model's sign detection and classification capabilities. At the top, a dashcam image is shown with the sign detected in a magenta bounding box. At the bottom, the extracted sign is show with the top 5 predictions of the signs classification.

[2] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[3] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[4] C. G. Serna and Y. Ruichek, "Classification of traffic signs: The european dataset," *IEEE Access*, vol. 6, pp. 78 136–78 148, 2018.

[5] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural networks*, vol. 32, pp. 323–332, 2012.

[6] Y. Jin, Y. Fu, W. Wang, J. Guo, C. Ren, and X. Xiang, "Multi-feature fusion and enhancement single shot detector for traffic sign recognition," *IEEE Access*, vol. 8, pp. 38 931–38 940, 2020.

[7] J. Greenhalgh and M. Mirmehdi, "Real-time detection and recognition of road traffic signs," *IEEE transactions on intelligent transportation systems*, vol. 13, no. 4, pp. 1498–1506, 2012.

[8] A. Shustanov and P. Yakimov, "Cnn design for real-time traffic sign recognition," *Procedia engineering*, vol. 201, pp. 718–725, 2017.

[9] J. Jin, K. Fu, and C. Zhang, "Traffic sign recognition with hinge loss trained convolutional neural networks," *IEEE transactions on intelligent transportation systems*, vol. 15, no. 5, pp. 1991–2000, 2014.

[10] F. Zaklouta and B. Stanciulescu, "Real-time traffic sign recognition in three stages," *Robotics and autonomous systems*, vol. 62, no. 1, pp. 16–24, 2014.

[11] A. Ellahyani, M. El Ansari, and I. El Jaafari, "Traffic sign detection and recognition based on random forests," *Applied Soft Computing*, vol. 46, pp. 805–815, 2016.

[12] C. Dewi, R.-C. Chen, Y.-T. Liu, X. Jiang, and K. D. Hartomo, "Yolo v4 for advanced traffic sign recognition with synthetic training data generated by various gan," *IEEE Access*, vol. 9, pp. 97 228–97 242, 2021.

[13] C. Dewi, R.-C. Chen, Y.-T. Liu, and S.-K. Tai, "Synthetic data generation using dcgan for improved traffic sign recognition," *Neural Computing and Applications*, pp. 1–16, 2021.

[14] B. Moiseev, A. Konev, A. Chigorin, and A. Konushin, "Evaluation of traffic sign recognition methods trained on synthetically generated data," in *International Conference on Advanced Concepts for Intelligent Vision Systems*. Springer, 2013, pp. 576–583.

[15] L. T. Torres, T. M. Paixão, R. F. Berriel, A. F. De Souza, C. Badue, N. Sebe, and T. Oliveira-Santos, "Effortless deep training for traffic sign detection using templates and arbitrary natural images," in *2019 international joint conference on neural networks (IJCNN)*. IEEE, 2019, pp. 1–7.

[16] D. Horn, L. Janssen, and S. Houben, "Automated selection of high-quality synthetic images for data-driven machine learning: A study on traffic signs," in *2021 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2021, pp. 832–837.

[17] A. Stergiou, G. Kalliatakis, and C. Chrysoulas, "Traffic sign recognition based on synthesised training data," *Big Data and Cognitive*

Computing, vol. 2, no. 3, p. 19, 2018.

[18] O. Araar, A. Amamra, A. Abdeldaim, and I. Vitanov, "Traffic sign recognition using a synthetic data training approach," *International Journal on Artificial Intelligence Tools*, vol. 29, no. 05, p. 2050013, 2020.

[19] A. Konushin, B. Faizov, and V. Shakhuro, "Road images augmentation with synthetic traffic signs using neural networks," *arXiv preprint arXiv:2101.04927*, 2021.

[20] G. Villalonga, J. Van de Weijer, and A. M. López, "Recognizing new classes with synthetic data in the loop: application to traffic sign recognition," *Sensors*, vol. 20, no. 3, p. 583, 2020.

[21] V. Shakhuro, B. Faizov, and A. Konushin, "Rare traffic sign recognition using synthetic training data," in *Proceedings of the 3rd International Conference on Video and Image Processing*, 2019, pp. 23–26.

[22] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[23] G. Chitnis, "cocoapi," https://github.com/gautamchitnis/cocoapi, 2020.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[25] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[26] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[27] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[28] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[30] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[31] E. Zerhouni, D. Lányi, M. Viana, and M. Gabrani, "Wide residual networks for mitosis detection," in *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*. IEEE, 2017, pp. 924–928.

[32] X. Chen and A. Gupta, "An implementation of faster rcnn with study for region sampling," *arXiv preprint arXiv:1702.02138*, 2017.

[33] R. Meyes, C. W. de Puiseau, and T. Meisen, "Ablation Studies in Artificial Neural Networks," 2019. [Online]. Available: https://arxiv.org/abs/1901.08644