2024 IEEE International Conference on Industrial Technology (ICIT), 25-27 March 2024, Bristol, UK
DOI: 10.1109/ICIT58233.2024.10540817

Reading and understanding house numbers for delivery robots using the "SVHN Dataset"

1st Omkar Pradhan
SATM
Cranfield University
Cranfield, UK
omkarnilesh.pradhan.867@cranfield.ac.uk

2nd Dr. Gilbert Tang
SATM
Cranfield University
Cranfield, UK
g.tang@cranfield.ac.uk

3rd Christos Makris
Ocado Technology
Hatfield, UK
christos.makris@ocado.com

4th Radhika Gudipati
Ocado Technology
Hatfield, UK
radhika.gudipati@ocado.com

***Abstract*— Detecting street house numbers in complex environments is a challenging robotics and computer vision task that could be valuable in enhancing the accuracy of delivery robots' localisation. The development of this technology also has positive implications for address parsing and postal services. This project focuses on building a robust and efficient system that deals with the complexities associated with detecting house numbers in street scenes. The models in this system are trained on Stanford University's SVHN (Street View House Numbers) dataset. By fine-tuning the YOLO's (You Only Look Once) nano model results with an effective detection range from 1.02 meters to 4.5. The optimum allowance for angle of tilt was $\pm 15^{o}$. The inference resolution was obtained to be $2160 * 1620$ with inference delay of 35 milliseconds**

***Index Terms*—Artificial Intelligence, Character Recognition, Computer Vision, Object Detection, YOLO, SVHN.**

1. Introduction

In the dynamic field of artificial intelligence (AI) development, rapid advancements in computational capabilities and sophisticated algorithms propel the creation of AI and machine learning (ML) models, meticulously designed for precise object categorization to emulate human cognitive abilities. Across industries, the widespread adoption of AI is driven by its unparalleled accuracy and minimal downtime, contributing transformative benefits in fields such as disease diagnosis, fraud detection, and autonomous vehicles. This project explores multiple facets of machine learning and machine vision, with applications extending to various domains, notably in the optimization of delivery robots. The fusion of AI and computer vision equips these robots with the capability to navigate complex environments, efficiently recognize and handle objects, and seamlessly interact with their surroundings. This integration enhances the precision of object and number detection, fostering the evolution of smart, efficient, and adaptive delivery systems. The project objectives are as follows:

1) Literature review: This part develops the baseline of the development by reviewing existing work on this research topic.
2) Pre-processing: The image data is being prepared before any machine learning processing.
3) Deciding the optimum AI model to train on for the best results: this part will compare the results for multiple possible AI models based on the necessary factors as follows.
    a) Considering the inference time
    b) Increasing the efficiency by tuning the hyper-parameters
    c) Real-time object detection analysis
4) Hardware implementation: in this part, the model needs to be implemented by using a camera and thus doing the inferencing of live camera feed.

1.1. Related works

The state-of-the-art computational techniques can match human-level accuracy in pattern recognition and object detection when tested in a controlled environment, but this gap widens as we go towards complex scenarios. To deal with number identification in the real-world environment, we need to have robust systems. [1] Used Convolution networks architecture to deal with the issue. Instead of implementing max pooling, Lp pooling was implemented, and multi-stage features were used along with the training using stochastic gradient descent (SGD). With the implementation of this architecture, and accuracy of pattern recognition improved to 94.97%.

To determine the most suitable machine learning model for a specific application, researchers referred to various papers that summarized findings. In a study by [2], five machine learning models (Neural Network, K-Nearest Neighbour, Random-forest, Decision tree, and bagging with gradient boost) were compared using the MNIST dataset. They applied multiple pre-processing techniques and found that Neural Networks achieved the highest accuracy at 95.73%, but struggled with poorly written digits. [3] also used the MNIST dataset, comparing Linear SVM, Multilayered Perception, and Convolutional Neural Networks (CNN). They considered execution time, complexity, accuracy rate,

epochs, and hidden layers. SVM provided the fastest results for simple data, while CNN excelled in accuracy and efficiency for more complex data. [4] and [5] also trained on MNIST, using Ensemble systems, TFE-SVM, C-NN, and Large C-NN+. Their results reinforced that Neural networks, particularly CNN, consistently outperformed other models in terms of efficiency and accuracy.

[6] conducted a classification analysis using 18 Deep Neural Network (D-NN) models, including ResNets, DenseNets, MobileNets, Nasnets, VGG Nets, and Alex Nets. They trained these models on the ImageNet Dataset, generating adversarial images from a random sample of 1000 images. Evaluation criteria included attack success rate, distortion using l_2 and l_∞ norms, CLEVER scores, and transferability, providing a comprehensive understanding of model performance. [7], in a separate study, compared VGG16, VGG19, and ResNet50 models trained on a custom dataset of 6000 images with five classes. The results showed accuracies of 0.9667, 0.9707, and 0.9733.

Object detection combines localization and classification and is often implemented using CNN architecture. In [8], Faster-RCNN, YOLO$_{v5}$, and SSD were compared using an automobile training dataset. Faster-RCNN demonstrated better accuracy but was unsuitable for real-time applications due to its 2-stage nature, making YOLO the top performer. In another comparison by [9] in 2021, YOLO$_{v6}$ was pitted against SSD, with SSD outperforming YOLO in terms of Frames Per Second (FPS) and achieving a higher mean Average Precision (mAP) score.

[10] delved into the SVHN (Street View House Numbers) dataset from Stanford, consisting of 73,257 images in the training set and 26,032 in the test set. An additional SVHN extra dataset, with 531,131 less complex images, introduces potential model bias. Feature learning, utilizing methods such as Histogram of Oriented Gradients (HOG) and Sauvola binarization, is employed to detect features in these complex images. Post-processing involves comparing results from algorithms like HOG binary features, K-means, and Stacked Sparse Auto-Encoders. The findings favor the K-means-based system, although a notable challenge is the continuous failure of the binarization algorithm to separate characters from their surrounding backgrounds.

1.2. Research Methodology

The study followed a general methodology that involved collecting relevant papers in the AI field, laying the foundation for the research direction. Once the basics were established, specific models were chosen for the project. Data analysis and preparation were conducted to facilitate model training. During the training and testing phases, the selected models were individually trained and fine-tuned for improved performance. Challenges encountered at each stage were addressed through the study of GitHub libraries and literature. This iterative process continued until the desired results were achieved.

2. Implementation

To successfully implement the training, the data was prepared in the format required for the training of the specific model (also known as pre-processing), as required for using YOLOv8. The process of data pre-processing, augmenting and tuning parameters are defined in the further subsections 2.1, 2.2, 2.3, 2.4 and 2.5

2.1. Bounding Box Labels

As stated in section 1.1, the study converged towards the solution that YOLO is the optimum model to begin with. According to Ultralytics, a YOLO model is to be trained with a training and validation dataset. The bifurcated dataset's image needs to be accompanied by the text file with the information about the classes and the bounding box information defined in it. Ex. (if an image is having the name 'xyz.jpg', the text file should be 'xyz.txt'). The information in the text file is to be exactly in the form which is required to train the YOLO model. The information is shown in table1.

TABLE 1. TEXT FILE FORMAT

C1	CBB1_X_norm	CBB1_Y_norm	W1_norm	H1_norm
C2	CBB2_X_norm	CBB2_Y_norm	W2_norm	H2_norm
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
Cn	CBBn_X_norm	CBBn_Y_norm	Wn_norm	Hn_norm

Where, Cn =nth Class number CBBn_X_norm = Normalised nth Centre co-ordinate of Bounding Box in X-axis CBBn_Y_norm = Normalised nth Centre co-ordinate of Bounding Box in Y-axis Wn_norm = Normalised nth Bounding Box Width Hn_norm = Normalised nth Bounding Box Height

Stanford University's SVHN dataset comes in two formats: one with all images formatted to 32x32 pixels, heavily cropped to show only a single number, and another with images in a general form including parts of the environment. The 'digitstruct.mat' file provides data on bounding boxes and classes. To create individual text files for each image, a MATLAB code was developed. This code reads image dimensions, extracts information from the mat file, and normalizes and formats the data according to the required specifications.

To get to know about the number of instances for which the classes appear in the training and validation set in total, the graph is plotted.

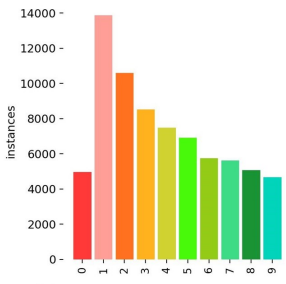

Figure 1. Total number of instances of classes in training and validation

The graph, when plotted, shows that the highest number of instances that appear are for number '1', and it gradually goes on decreasing till number '9' and is equal to the number '0'.

2.2. YAML file

The YAML file contains crucial details about the dataset, specifying the locations of the training and testing datasets and defining classes. **Path** indicates the absolute location of the dataset folder. **Train** and **Val** denote the relative locations of the training and validation datasets. **Names** assign encoding to each class (e.g., 7 represents the encoded value for 'H'). These encodings are reflected in the Cn section of the Bounding Box Labels file, detailed in Section 2.1.

2.3. structure of the Files

To train the model, the dataset with its labels and images needs to be organised in a specific order for the training. The order is shown in Figure 2 The absolute path of the

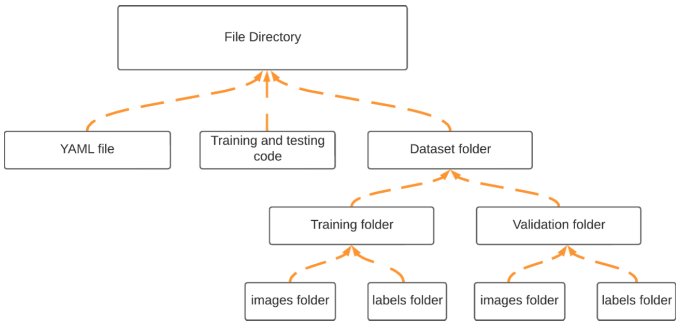

Figure 2. Structure of the files

file director is to be given in the YAML file, as explained in section 2.2. The names of the files matter the most while training the YOLO models using the Ultralytics library. There is a compulsion to have folders named 'Images' and 'Labels' in both training and validation because the Utralytics library searches for the folders with exact names. The names of the train and validation folder can be user-defined, considering you define the exact name in the YAML file with the specified location.

2.4. Hyper-parameters

During the training of a model, multiple hyper-parameters can be changed to check the impact of the variance of those parameters on the inference. In this study, the hyper-parameters that are studied and tuned are shown in table 2. The models were tuned and analysed using the different combinations.

TABLE 2. TUNED HYPER-PARAMETERS

[HTML]D5DCE4Hyper-parameter	[HTML]D5DCE4Values
Optimiser	Auto, SGD, RAdams, Adamax
Pretrained	True / False
Degrees	±5, ±15, ±45
Imgsz	240, 320, 640, 720, 1080
Model	YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8x

3. Analysis and Discussion

Before hyper-parameter tuning, the initial step involves identifying the best-performing models for the SVHN dataset using their default configurations. This preliminary performance analysis aims to streamline the process by excluding slower or less efficient models. The training and testing procedures are carried out on the hardware specifications detailed in Table 3.

TABLE 3. HARDWARE SPECIFICATIONS

[HTML]D5DCE4Hardware Specification	
CPU	intel i5-11400H 6 cores 12 threads
GPU	Nvidia RTX3050 Laptop GPU 4 GB
RAM	32 GB DDR4 3200 MHz
Max Power TDP	180W
[HTML]D5DCE4Software Specification	
OS	Ubuntu 22.04.2 Jammy Jellybean
IDE	VS Code
Python version	3.10
Cuda version	11.8.0
Cudnn version	8.6.0.163
Tensorflow version	2.12.0

3.1. Default Model Performance Analysis

To decide which model works the best, multiple aspects need to be considered. To get the overall working of the different versions of the YOLOv8 models, parameters like training time, precision vs epochs, mAP vs epochs, and Recall vs epochs were compared.

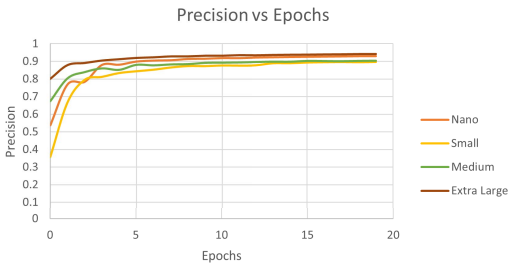

Figure 3. Precision vs Epochs

$$Precsion = TP/(TP + FP) \quad (1)$$

The top three performers, YOLOv8x, YOLOv8n, and YOLOv8m, were further examined. The study delved deeper into these models, analyzing Precision-Confidence and Recall-Confidence curves to gain a comprehensive understanding of their overall performance.

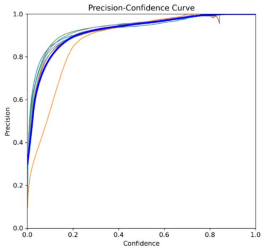

Figure 4. Precision-Confidence Curve of YOLOv8x, YOLOv8n, YOLOv8m

Precision confidence curves for the three models revealed a consistent lag in precision for the digit '1,' with the widest gap in YOLOv8x, narrowing in YOLOv8n, and further reducing in YOLOv8m. As confidence levels increased, the gap diminished, and all models peaked at approximately 82% confidence. Despite '1' having the highest training instances, the lower precision is puzzling. To address this, a detailed analysis of the confusion matrix is recommended for insights into class predictions, including true and false predictions across all classes.

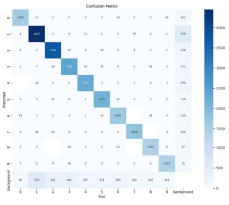

Figure 5. Confusion Matrix for YOLOv8x

In the case of YOLOv8m, it was observed that 20% of the time, it identified the number '1' in the background. This indicates that the problem of misclassification is more related to errors in the dataset rather than how the models are trained.

Comparing the models, YOLOv8x emerges as the top performer, but for tuning purposes, YOLOv8n is selected as the best model. This decision is based on YOLOv8n having precision levels almost equivalent to YOLOv8x while being significantly lighter in size. This choice results in reduced computational expense and enables implementation on mobile systems.

3.2. Hyper-Parameters Tuning Analysis

YOLOv8n was trained with a fixed Epoch of 20 and a batch size of -1, optimizing CUDA memory usage. A fixed positive batch size can either slow down training or exceed memory capacity. By setting it to -1, the system dynamically determines simultaneous training images, optimizing memory usage without trial and error.

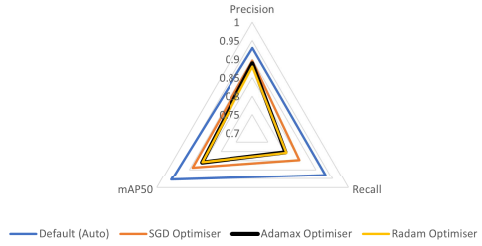

Figure 6. Precision-recall-mAP comparison of Nano models for different Optimiser

When plotting Precision, Recall, and mAP, the graph with the largest area indicates the optimum and best results. In this scenario, the Default configuration consistently delivers the best results. The Optimiser with high Precision, Recall, and mAP collectively outperforms the alternative, resulting in fewer overall losses. Therefore, the Default (Auto) configuration is the most suitable choice in this case. Enabling tilt for the models revealed that as the tilt angles increased, losses also increased. The model with no rotation had the lowest loss, while the model with a 45° tilt had the highest, with the 15° tilt falling in between.

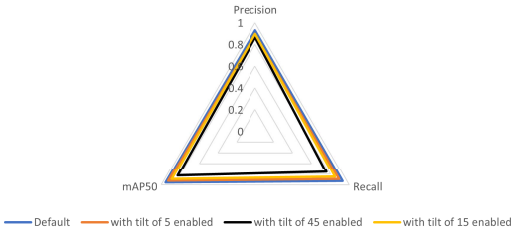

Figure 7. Precision-recall-mAP comparison of Nano models for different tilt angles

Training models with various training resolutions, ranging from 240 pixels width to 1080 pixels, led to an unexpected outcome: the model trained with a 240-pixel resolution performed the best. This result contradicts the assumption that higher training resolution leads to better results, which typically holds true when the actual image resolution is sufficiently high. It's crucial for the training resolution to closely match the actual image resolution.

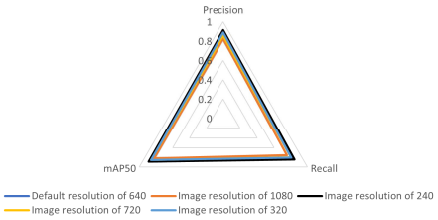

Figure 8. Precision-recall-mAP comparison of Nano models for different training resolutions

To address this, a code was developed to identify the highest and lowest image resolutions in the training dataset, as well as the average resolution across all images.

The analysis revealed that the average image resolution is 128 pixels in width and 50 pixels in height. This indicates that when the model is trained with a resolution higher than the actual image resolution, it introduces significant noise, leading to increased losses and reduced accuracy. Therefore, the optimal model, taking resolution into account, is one trained with a 240-pixel width resolution.

3.3. Inference Timings and Min/Max Distance

To evaluate the real-time performance of the models, all trained models with various hyper-parameters, as discussed in section 3.2, were configured for inference. The recorded metrics include inference time and the closest and farthest distances at which classes are successfully classified. These measurements were collected while altering the inference resolution, spanning 240, 320, 640, 720, 1080, 2160, and 3840-pixel widths.

Various models were assessed for effective classification and localization distances through experiments in a lab. A complex-font image with numbers 0 to 9 was affixed to a movable structure. A webcam, paired with a laser-guided distance meter, ensured accurate measurements. Multiple experiments, with varying hyper-parameters and inference resolutions, were conducted. Results were tabulated, and final outcomes were derived by averaging data from three experiment repetitions.

RAdam demonstrated standout performance in optimizer evaluations, achieving an inference distance of 7.389 meters and a lower bound of 1.396 meters at a training resolution of 3840 pixels, albeit with a noticeable 100-millisecond inference lag. Subsequent reevaluation at an inference resolution of 2160 pixels yielded a range from 0.967 to 5.483 meters. Optimal results were obtained with a tilt angle of 15°, aligning with previous comparisons, offering flexibility within ±15° and an inference distance spanning 1.11 to 3.171 meters. Similarly, the 240-pixel training resolution echoed earlier findings, providing the best performance with a distance range of 1.1 to 4.9 meters.

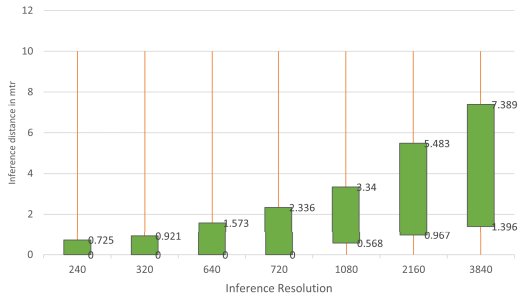

Figure 9. Inference distance for YOLOv8n with RAdam Optimiser

3.4. Compound Hyper-parameters

The analysis in Section 3.3 indicates that a training resolution of 240 pixels width and the RAdam optimizer produce optimal results within their respective segments. A tilt angle of 15° is chosen for a balance between robustness and a reduction in inference distance. Subsequently, a model is trained with these optimal hyperparameter settings, comprising a training resolution of 240 pixels width, the RAdam optimizer, and a ±15° tilt angle.

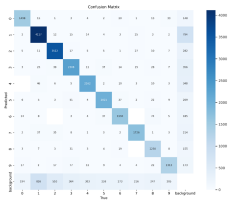

Figure 10. Confusion matrix for YOLOv8 with compound tuning

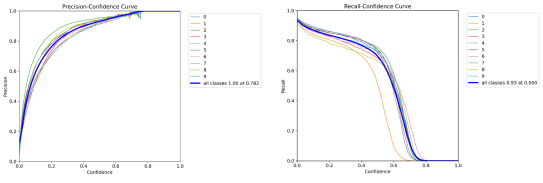

Figure 11. Precision-Confidence curve and Recall-Confidence curve for YOLOv8 with compound tuning

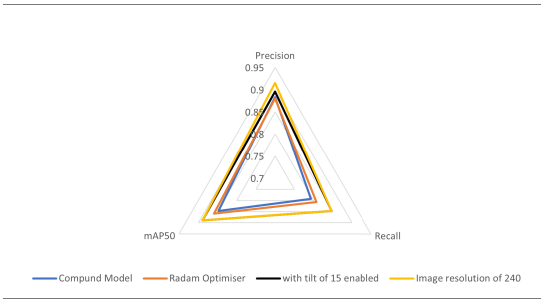

Figure 12. Precision recall mAP comparison of Compound tuned model with individual tuned models

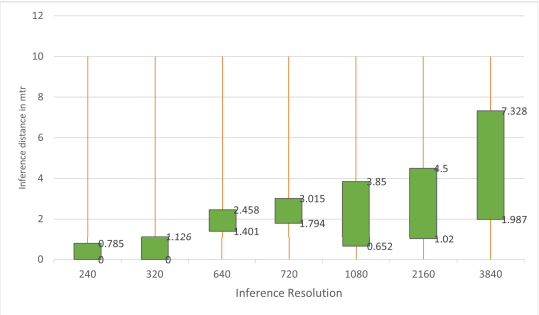

Figure 13. Inference distance for YOLOv8n with Compound Model Tuning

4. Results and challenges

4.1. Results

After the successful completion of the tuning, YOLOv8n was selected with hyperparameters as defined.

TABLE 4: Finalised Hyper-parameters

Hyperparameter Key	Value
Imgsz (Training)	240
Optimiser	RAdam
Degrees	15

This model gave the optimum results during the analysis with the effective inference distance of 1.02 meters to 4.5 meters with an average latency of 32 milliseconds.

4.2. Challenges

4.2.1. Configuration errors. Setting up the environment demands careful configuration. Neglecting to ensure compatibility among versions of PyTorch, cuDNN, CUDA, and TensorFlow can lead to a chain of failures and potentially a malfunctioning setup. It is not recommended to install the latest version using the pip command, as it can result in compatibility issues or failure to recognize specific packages. To address this, it's advisable to follow the comprehensive list provided by TensorFlow to verify the compatibility of versions.

4.2.2. Classification of Number '1'. In Section 3.2, an average image resolution of 128 x 50 pixels was determined. Training involves convolution and pooling layers to preserve features while reducing dimensions. The unclassified image section, representing the background class, may transform the number '1' into a background-like feature due to lower resolution, causing misclassification and reduced precision. This requires later detection or closer proximity for '1' than other classes, impacting overall performance. A proposed solution is outlined in Section 6.

5. Conclusion

The successful implementation of house number recognition in a complex environment was achieved using the SVHN (Street View House Numbers) dataset. A robust system was developed utilizing a webcam to detect and localize numbers with a remarkable 90% accuracy, covering an inference distance of up to 4.5 meters.

6. Future work

6.0.1. Recognition. To address the inference issue related to number '1', a potential solution is to create a custom image dataset by gathering images of number '1' from various open sources and incorporating them into the training dataset, along with their corresponding bounding box information. This approach can be used to assess whether adding more number '1' images to the dataset resolves the issue and whether it has any unintended consequences or alterations on the results for the other classifications.

6.0.2. Model Fine Tuning. Much finer tuning of the models can be done by changing the parameters more gradually, thus resulting in a more extensive analysis.

6.0.3. Delivery Robots. The trained algorithm after furthermore fine-tuning, can be implemented on the actual robots such as delivery robots to check practical implementation of the system. The fusion of GPS and recognition can be implemented and analysed for the enhancement in the accuracy of the current systems in use.

References

[1] P. Sermanet, S. Chintala, and Y. LeCun, "Convolutional neural networks applied to house numbers digit classification," *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pp. 3288–3291, 2012.

[2] S. Chen, R. Almamlook, Y. Gu, and L. wells, "Offline handwritten digits recognition using machine learning," *European Conference on Computer Vision (ECCV)*, 2018.

[3] R. Dixit, R. Kushwah, and S. Pashine, "Handwritten digit recognition using machine and deep learning algorithms," *International Journal of Computer Applications*, vol. 176, pp. 27–33, 07 2020.

[4] A. Shrivastava, I. Jaggi, S. Gupta, and D. Gupta, "Handwritten digit recognition using machine learning: A review," *2019 2nd International Conference on Power Energy, Environment and Intelligent Control (PEEIC)*, pp. 322–326, 2019.

[5] R. KARAKAYA and S. Çakar, "Handwritten digit recognition using machine learning," *Sakarya University Journal of Science*, vol. 25, 10 2020.

[6] D. Su, H. Zhang, H. Chen, J. Yi, P.-Y. Chen, and Y. Gao, *Is Robustness the Cost of Accuracy? – A Comprehensive Study on the Robustness of 18 Deep Image Classification Models: 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part XII*, pp. 644–661. 09 2018.

[7] S. Mascarenhas and M. Agarwal, "A comparison between vgg16, vgg19 and resnet50 architecture frameworks for image classification," in *2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON)*, vol. 1, pp. 96–99, Nov 2021.

[8] J. ah Kim, J.-Y. Sung, and S.-H. Park, "Comparison of faster-rcnn, yolo, and ssd for real-time vehicle type recognition," *2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia)*, pp. 1–4, 2020.

[9] M. Shetty, "A review on deep learning object detection: Yolo vs ssd," *International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)*, vol. 5, 2021.

[10] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

Reading and understanding house numbers for delivery robots using the ”SVHN Dataset”

Pradhan, Omkar N.

2024-06-05

Pradhan O, Tang G, Makris C, Gudipati R. (2024) Reading and understanding house numbers for delivery robots using the “SVHN Dataset”. In: 2024 IEEE International Conference on Industrial Technology (ICIT), 25-27 March 2024, Bristol, UK

https://doi.org/10.1109/ICIT58233.2024.10540817