Classifying with Noise: A First Step into Identifying Occluded Digits

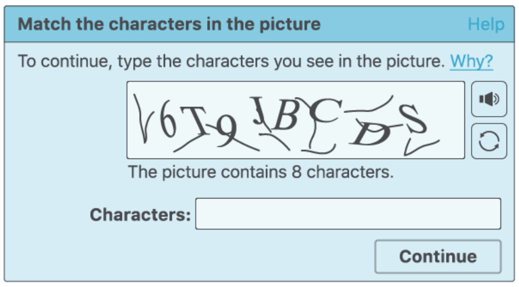
Brayden Carlson, John Z. Zhang
University of Lethbridge, Canada
brayden.carlson, john.zhang@uleth.ca

***Abstract*—The accurate detection and classification of occluded objects in images present an ongoing challenge in the field of computer vision. In our current work, we tackle a situation where we need to identify the objects that are handwritten digits from the MNIST dataset[1]. Our proposed approach is decomposed into two steps. In Step 1, a classification model based on Convolutional Neural Networks is designed and implemented to classify digits that are augmented with noise at varying degrees. In Step 2, we propose to introduce a model based on Single Shot Detector and tailor it to our purpose. Once a possible digit is located in an image, it will be then classified by our model from Step 1. While we are still finalizing Step 2 of our proposed approach, the experiment results from the classification model we propose for Step 1 are promising. In this paper, we are presenting the details of this model, including the architecture of the model, hyperparameters, dataset augmentation, etc. We believe that our proposed CNN-based model for classifying noisy digits is also applicable to classifying other objects, such as letters, characters, etc.**

I. Introduction

The notion of *artificial intelligence* (AI) appears in the 1950s [1]. People attempt to simulate human intelligence in game playing, language translation, natural language processing, etc., by computers. The progress in the field of AI is slow, due to the limitations of computational techniques and computing resources (e.g., processing power and storage equipment). However, the field is re-emerging as one of the most discussed topics in the past decades. Ongoing research and, especially, advancement in computing hardware, make it possible for computers to "learn", coined as *machine learning* [2], and, therefore, to engage in various applications. Classification is one of the core tasks in machine learning [2]: classify new data by learning from the past data by a computer. Applications of classification are ubiquitous, including medical diagnosis that determines whether a tumor is malignant, sentiment analysis that concludes if a review is positive, negative or neutral, image classification that tells what an animal is present in an image, etc.

The development of *Convolutional Neural Networks* (CNNs) [3] has revolutionized the classification methodologies (and in general, the field of artificial intelligence and machine learning). The origins of CNNs can be traced back to the 1980s, with the introduction of the *Neocognitron* by Fukushima et al. [4]. This early neural network is designed to recognize visual patterns. It also lays the groundwork for future developments in CNNs. A CNN is suited for processing grid-structured inputs, including image recognition, object segmentation, object detection and localization, etc. [5]. In object detection, a common challenge is occlusion, where objects can partially or fully obscure other objects [2]. An illustrative example for such a situation is CAPTCHA (Completely Automated Public Turing test to Tell Computers and Humans Apart) [6], [7], as shown in Fig. 1, where digits and letters can overlap, intersect, rotate and even have noisy strokes.


Match the characters in the picture Help

To continue, type the characters you see in the picture. Why?

The picture contains 8 characters.

Characters:

Continue

Fig. 1. An example of a simple CAPTCHA

In our current work, we focus on identifying occluded digits in images. Our road map towards this task consists of two steps. In Step 1, a classification model based on CNNs is designed and implemented, in order to classify digits that are augmented with noise at varying degrees. In Step 2, we plan to introduce a model based on *Single Shot Detector* [8] and tailor it to locate possible digits on an image. A digit, once located, is then classified using our model from Step 1. In this paper, we present the details of our proposed CNN-based model for classification in Step 1.

II. Related works

CNNs represent our continuous efforts to mimic human intelligence through computers. In 1998, LeCun et al. [3] make a significant breakthrough in CNNs. They create the LeNet-5, based on CNN, which is used for digit recognition tasks. This model demonstrates the practical applications of CNNs and its exceptional performance over other pattern recognition methods. CNNs have become *de facto* essential in object detection, and significantly improved upon conventional techniques. A CNN is capable of learning spatial hierarchies of features from images. This capability stems from its deep architecture, which

[1]https://keras.io/api/datasets/mnist/.

consists of multiple layers of convolutional filters, pooling and fully connected layers [9]. Unlike classical algorithms that require manual intervention, CNNs can automate these processes, which then leads to more robust and accurate object detection across a wide range of applications. A thorough discussion of CNNs can be found in [5].

The evolution of CNN-based models has played an important role in overcoming challenges in object detection, particularly in speed and accuracy. The challenges in object detection range from irregular shapes and orientations [10], complex backgrounds [11], objects that overlap in the image plane [12], objects of varying scales and sizes [13], occlusions [11], [14], etc. CNNs are flexible and work well for both 2D and 3D detection. In 2D-object detection, an algorithm or model can be used to identify and locate several objects in images or video frames. This approach to object detection is widely used in various applications including surveillance, vehicle navigation, and automated quality control. 3D object detection identifies and locates objects that are in 3D space [15]. It is important in areas, such as robotics, autonomous driving, and augmented reality [16], [17].

The integration of object detection technology and cybersecurity is illustrated by CAPTCHAs (An example is shown in Section I), reCAPTCHA and its variants. CAPTCHAs are deployed to distinguish between human users and automated bots. CAPTCHAs seek to prevent automated bots from executing unauthorized actions on websites or online services [18], including creating fake accounts, spamming, and automated data scraping [7]. The evolution of CAPTCHAs highlights a competition between cybersecurity measures and advancements in algorithms and machine learning in object detection. The challenge lies in creating CAPTCHAs that are resilient against automated attacks while remaining user-friendly as object detection techniques becomes more powerful. Object detection in the biomedical field is also an important task for the identification of organs, cells, tissues, and other anatomical structures [19]–[21] within medical imaging such CT (Computed Tomography) scans, X-Rays, and Magnetic Resonance Imaging (MRI). This process can assist in medical diagnosis, biomedical research, treatment planning and disease monitoring. CNN-based approaches, such as U-net and its variants, have demonstrated significant success in this domain [21], [22]. These CNNs excel at capturing patterns in image data and contributing to accurate delineation of complex structures, such as tumors in MRI scans or blood vessels in retinal images.

In detecting object, the *Single Shot MultiBox Detector* model presents a significant advancement, as discussed by Liu et al. [8]. This model integrates object localization and classification within a single forward pass. The model is distinct because of its utilization of a fixed set of default bounding boxes, known as “default boxes”, across various aspect ratios and scales at each feature map. This novel approach enables the model to predict the presence of object categories within these boxes and adjust their sizes accordingly to closely match the object shapes [8]. In addition, there are some other object detection techniques that are proposed in the literature, including YOLO (You Only Look Once) by Redmon et al. [23] and R-CNN (Region-based Convolutional Neural Networks) by Girshick et al. [24]. Due to space consideration, we do not discuss them in detail but will present a comprehensive treatment on them when we finish completely our proposed approach for identifying occluded digits, as discussed in Section I.

III. Classifying Noisy Digits: A CNN-based Model

We plan to attack the problem, using CNNs, that detects accurately occluded handwritten digits in grayscale images. The digits on such images can be both overlapping and non-overlapping, which creates conditions that closely resemble real-world object detection where objects may be occluded. The road map of our work is divided into two steps. Due to overlapping among digits, the pixel data of a digit on an image may contain some ”noisy” information, introduced by the other overlapped digits. Our first step is to design and implement a CNN-based model that is able to handle ”contaminated” individual digits, enhancing its resilience against extra noisy information during detection. Such a model will be then used as a black box in our second step. In this paper, we are reporting our work on the first step.

Our CNN-based classification model is structured to process images for the purpose of digit classification by training on noisy images. It is trained on input images of 28×28 pixels, and utilizes a deep learning architecture to classify these images into one of ten (10) categories (0 - 9). The model’s architecture is divided into two primary components: a feature extraction backbone and a classifier. An overview of the model’s architecture can be seen in Figs. 2 and 3.

The feature extraction backbone is built upon a sequence of *convolutional neural network layers* that are designed to capture the patterns and features [3] on images. The backbone consists of multiple stages. Each is comprised of convolutional layers followed by *batch normalization* and *relu activation functions*. The series of convolutional layers use a variety of filter sizes to extract both low-level and high-level features from the input images.

At the end of the feature extraction process, the high-dimensional input images are transformed into a flattened, low-dimensional representation that encapsulates the important information needed for accurate digit classification by the classifier. This process of flattening prepares the data for the fully connected layers of the classifier. Our model employs a classifier which consists of a sequence of fully connected *linear layers* that interpret the extracted features to make a classification decision. The first layer of the classifier takes the flattened feature representation as input and maps it to a higher-dimensional space to facilitate complex relationships between features. This is followed by a relu activation and a *dropout layer* to introduce *non-linearity* and mitigate *overfitting* [9]. Each node on the *output layer* corresponds to one of the ten (10) digit classes. The output of the classifier is passed through a *log softmax* function, which converts the raw class scores into log probabilities. This makes them more

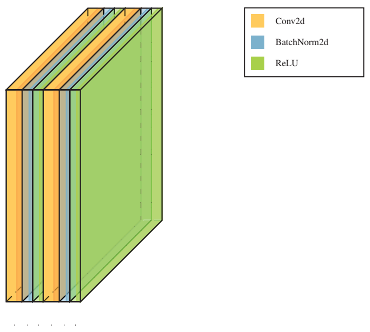


Fig. 2. Classification: Building Block.

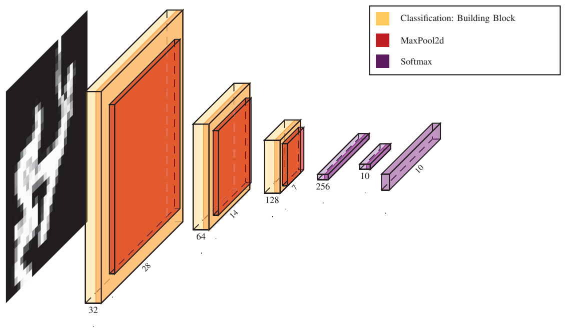


Fig. 3. Classification: Architecture

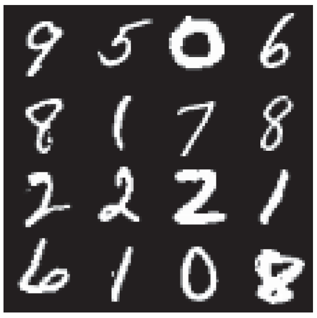


Fig. 4. 16 example digits from the MNIST dataset.

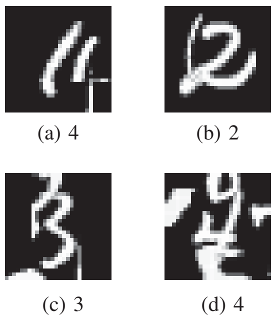


Fig. 5. Introducing noise into the MNIST dataset

interpretable and suitable for calculating loss during training. The log softmax layer guarantees that the output can be used to compute the loss function. For technical details of CNNs, for the sake of space, we refer a reader to [5].

Training data is key in successful deployment of CNN-based models [5]. Our CNN-based classifier is developed using a variation of the MNIST dataset, which has been purposely augmented with noise (See Section IV). The MNIST dataset contains 70,000 images. Each image features a handwritten digit ranging from 0 to 9 and is normalized to a 28×28-pixel image. A collection of 16 example digit images, as shown in Fig. 4, highlights the unique stroke characteristics of individual digits. The handwriting styles vary significantly, even for the same digit. The digits are represented on a white against black background in grayscale format with brightness levels ranging from 0.0 to 1.0. In the first step, to prepare our training data, we introduce random strokes into the digit images from the original MNIST dataset, as shown by the examples in Fig. 5. Our goal is to employ a supervised learning paradigm and construct a CNN model to classify a digit with a noisy background. The challenge behind this is to design a robust neural network capable of learning and adapting to digits' variations. We expect that our model distinguishes between noise and the actual digit and classifies the latter, simulating real-world scenarios in handwritten digits.

IV. Experiments and Results

A. The Computing Environment

Our experiments are conducted on a desktop computer powered by an Intel Core i7-13700K processor and GeForce RTX 4090 graphics card that have 24 GB of video memory. The system also includes 64 GB of RAM using G.Skill Trident Z5 Neo modules and runs Windows 11 Home (22H2) as its operating system. For the development environment, Python 3.11.2[2] serves as the primary programming language. It is supported by the Python ecosystem and libraries which are used for an efficient machine learning development. This includes CUDA 11.8[3] and cuDNN 3.9.3[4] for leveraging GPU acceleration, PyTorch Lightning 2.1.2[5] for structuring the machine learning workflow, and additional tools such as NumPy[6], Pandas[7], Matplotlib[8], and Optuna[9] for numerical operations, data manipulation, visualization, and hyperparameter optimization, respectively.

B. Augmenting Datasets with Noise

We carefully prepare our datasets for training, validation and testing. The original MNIST dataset consists of 60,000 training images and 10,000 testing images. To serve our purpose, the training dataset is further partitioned into a new training dataset and a validation dataset, according to a 80/20 split. The training, validation and testing sets consist of 48,000, 12,000 and 10,000 images, respectively. Afterwards, each digit image from the newly created dataset is decomposed into four

[2]https://www.python.org.
[3]https://developer.nvidia.com/cuda-toolkit.
[4]https://developer.nvidia.com/cudnn.
[5]https://lightning.ai/docs/pytorch/stable.
[6]https://numpy.org.
[7]https://pandas.pydata.org.
[8]https://matplotlib.org.
[9]https://optuna.org.

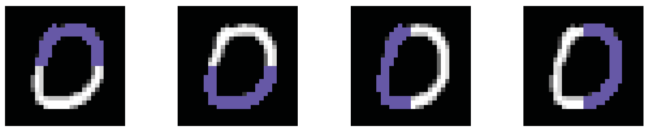


Fig. 6. Decomposing a MNIST digit into segments (blue)

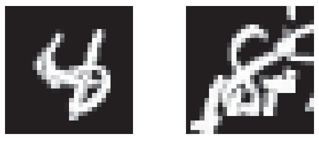


Fig. 7. The overlapping of 2 and 4 “segments”, respectively

segments: top, bottom, left, and right. An example is shown in Fig. 6. A selection process evaluates each segment against a predefined threshold to ensure it contains enough distracting information. This way, we exclude segments that are mostly similar to the background.

If a segment is selected, then it is recentered onto a standard 28 × 28-piexel canvas. This step places the segment onto a transparent background and writes the transparent images to disk for future use, composing a library of possible segments to be used for contamination. The next step randomly select these segments and then paste them onto randomly chosen MNIST digit images. This way, essentially we introduce noise into the digit images and create a new dataset from the original MNIST digits. An example is shown in Fig. 7. We allow control over the number of segments per image. This procedure is repeated, including training, validation, and testing, to generate multiple versions of our datasets with varying degrees of contamination. It is easy to see that we simulate the impact of real-world contamination on image data. Such contamination (noise) aims to enhance the robustness and adaptability of our classification model.

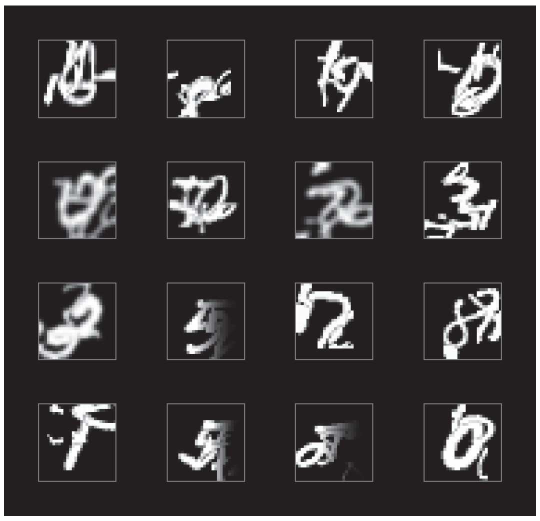


Fig. 8. Some examples of contaminated digits after augmen-tation

We also introduce additional augmentation before training the model in order to improve its robustness. One of the following transformations is applied to each image: a black and white gradient to simulate a fade effect, shifting, Gaussian blurring, contrast adjustment, random erasing, perspective distortion and resized cropping. We show some examples, after these transformations, in Fig. 8. The purpose of this is again to help the model generalize better on unseen noisy digit images. In order to facilitate further analysis, the metadata for each image is generated and stored. This metadata includes the file path of the modified image, the original image from which the segment is derived, the label of the digit, and the specifics of the stroke applied, including its file path and the coordinates of its placement onto the digit image.

C. The Loss Function

The loss function plays an important role in the training of the classification model. It serves as a metric that guides the optimization of the model’s parameters towards accurately classifying digits. For our proposed model, the *Negative Log Likelihood* (NLL) loss is employed [25]. The NLL loss is a measure of how well the model’s predicted probability distribution over the classes aligns with the ground truth distribution. This is done by comparing the logarithm of the model’s probabilities for each class to the one of the true labels’. The objective of training is to minimize this loss so that the model’s classifications closely match the true labels.

D. Training and Evaluation

Our proposed classification model goes through training and evaluation to ensure its accuracy and effectiveness in digit classification. During training, the model processes batches of grayscale images where each image is labeled with the correct digit. For each batch, it computes predictions and measures discrepancies using the NLL loss function. It optimizes the parameters through backpropagation with the *Adam optimizer* [5] to reduce the loss. Cautions are also taken to prevent overfitting and ensure generalization. There are dropout layers interspersed within the model to randomly ignore certain neurons during training. The training step is followed by a validation step where the model’s performance is assessed on a separate set of data it has not seen during training. This step is crucial for tuning the model’s hyperparameters and architecture to balance between underfitting and overfitting. The model’s accuracy and loss on the validation set serve as indicators of its ability to generalize.

In order to visually assess the model’s learning progress and performance, we also integrate visualization techniques into the training and validation steps. At the end of *each epoch*, the model’s capabilities are tested on a subset of images, which are visualized and compared against their true labels. This provides insight into the model’s classification accuracy and highlighted areas for improvement. The optimization process is fine-tuned through the configuration of the learning rate and other parameters of the Adam optimizer. This ensures that the model converges to an optimal set of parameters efficiently. The learning rate is set to a value that balanced the speed of convergence with the stability of the training process.

Upon the completion of the training phase, the model is evaluated in a test step. This final evaluation is critical for confirming the model's capabilities, and ensuring that it performs well on data it has never encountered before.

E. Classification Results

The performance of the classification model we propose demonstrates its capacity to identify individual digit images under varying degrees of complexity. As can be seen in Table I, the model showcases an incremental decrease in accuracy as the number of strokes increased. The model's performance drops from a 98.65% accuracy for images contaminated by one (1) stroke to 91.14% for the ones contaminated by five (5) strokes. This trend demonstrates the challenges posed by increasing visual complexity, even in the absence of overlap. The performance of our classification model is an indication that its feature extraction capabilities could be further optimized. We plan to conduct more experiments on digits with heavier contamination. We leave this to our efforts on the second step of our proposed model for digit detection (Section I).

TABLE I. ACCURACY ON 1 - 5 STROKES

Strokes	Classification Accuracy (%)
1	98.65
2	97.86
3	96.10
4	94.23
5	91.14

V. Summary

The results we are presenting in this paper are from the first step toward our efforts to locate and classify occluded digits in images. As evidenced in our discussions, our proposed CNN-based classification model successfully classifies digits that are contaminated at varying degrees. We believe that it should be applicable to identifying other objects, such as letters, special characters, etc. As discussed in Section I, the model will be later used in our efforts to locate and classify occluded digits in images.

References

[1] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain." *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[2] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson Education, 2020.

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, 1998, pp. 2278–2324.

[4] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, pp. 193–202, 1980.

[5] C. C. Aggarwal, *Neural Networks and Deep Learning*. Springer, 2018.

[6] M. I. Hossen, Y. Tu, M. F. Rabby, M. N. Islam, H. Cao, and X. Hei, "An object detection based solver for Google's image reCAPTCHA v2," in *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*. San Sebastian: USENIX Association, Oct. 2020, pp. 269–284. [Online]. Available: https://www.usenix.org/conference/raid2020/presentation/hossen

[7] R. Shao, Z. Shi, J. Yi, P.-Y. Chen, and C.-J. Hsieh, "Robust text captchas using adversarial examples," 2021.

[8] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, *SSD: Single Shot MultiBox Detector*. Springer International Publishing, 2016, p. 21–37. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46448-0_2

[9] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Neural Information Processing Systems*, vol. 25, p. 84–90, 01 2012.

[10] Z. Guo, X. Zhang, C. Liu, X. Ji, J. Jiao, and Q. Ye, "Convex-hull feature adaptation for oriented and densely packed object detection," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 8, pp. 5252–5265, 2022.

[11] Z. Chen, K. Chen, W. Lin, J. See, H. Yu, Y. Ke, and C. Yang, "Piou loss: Towards accurate oriented object detection in complex environments," 2020.

[12] Y. Liu, L. Liu, H. Rezatofighi, T.-T. Do, Q. Shi, and I. Reid, "Learning pairwise relationship for multi-object detection in crowded scenes," 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:58004654

[13] F. Chabot, Q.-C. Pham, and M. Chaouch, "Lapnet : Automatic balanced loss and optimal assignment for real-time dense object detection," 2020.

[14] P. Viola and M. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, pp. 137–154, 05 2004.

[15] T. Yin, X. Zhou, and P. Krähenbühl, "Center-based 3d object detection and tracking," 2021.

[16] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," 2017.

[17] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4490–4499.

[18] I. G. Mocanu, Z. Yang, and V. Belle, "Breaking captcha with capsule networks," *Neural Networks*, vol. 154, pp. 246–254, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608022002568

[19] S. Bera, "Partially occluded object detection and counting," in *Proceedings of the 2015 Third International Conference on Computer, Communication, Control and Information Technology (C3IT)*, 2015, pp. 1–6.

[20] J. Hung, A. Goodman, D. Ravel, S. Lopes, G. Rangel, O. Nery, B. Malleret, F. Nosten, M. Lacerda, M. Ferreira, L. Renia, M. Duraisingh, F. Costa, M. Marti, and A. Carpenter, "Keras r-cnn: library for cell detection in biological images using deep neural networks," *BMC Bioinformatics*, vol. 21, p. 300, 07 2020.

[21] C. Spahn, R. F. Laine, P. M. Pereira, E. G. de Mariscal, L. von Chamier, M. Conduit, M. G. de Pinho, G. Jacquemet, S. Holden, M. Heilemann, and R. Henriques, "Deepbacs: Bacterial image analysis using open-source deep learning approaches," *bioRxiv*, 2021. [Online]. Available: https://www.biorxiv.org/content/early/2021/11/03/2021.11.03.467152

[22] P. F. Jaeger, S. A. A. Kohl, S. Bickelhaupt, F. Isensee, T. A. Kuder, H.-P. Schlemmer, and K. H. Maier-Hein, "Retina u-net: Embarrassingly simple exploitation of segmentation supervision for medical object detection," 2018.

[23] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016.

[24] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2014.

[25] R. Krishnan and O. Tickoo, "Improving model calibration with accuracy versus uncertainty optimization," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 18 237–18 248. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/d3d9446802a44259755d38e6d163e820-Paper.pdf