Transfer Learning Object Detection Framework for Indoor Environments

Tanmay Singhal

Regeneron-Westchester Science and Engineering Fair (WESEF)

Research Report

All figures were created by student researcher unless otherwise noted

ACKNOWLEDGMENTS

I would like to extend my gratitude towards my mentor, Dr. Ujwal Krothapalli, at Virginia Tech/EY for his support in the process of starting my research and for helping me throughout the entire process. Special thanks to my peers, Matthew Mione and Sally Mestrich, for all their help in my dataset collection. Additionally, I would also like to express my gratitude towards my Science Research teachers, Angelo Piccirillo and Valerie Holmes, and my family and friends for their endless support.

CONTENTS

List of Figures

List of Tables

I. Abstract

The recent increase in interest in building autonomous mobile robots has made detecting and recognizing objects a very important task. Object detectors aim to mimic the function of the human eye; however, they are limited by their visual acuity. In complex indoor environments, object detectors remain prone to classification and localization errors as they frequently fail to account for different variations in color, texture, occlusions, and viewpoints in objects; thus, they are unable to provide a consistently reliable cue. Leading object detectors leverage recent progress in Convolutional Neural Networks (CNNs) to simultaneously detect and categorize objects of interest in dynamic scenes. We developed a novel classification and localization dataset compiled from eight open-source datasets. An additional real-world dataset was created using 5,000 crowdsourced images obtained from over 30 freelance volunteers. Different strategies to train ResNet50-based models were explored to achieve better object detection accuracy (mAP) using five classes - doors, door handles, chairs, tables, and extraneous objects. The architectures were trained using the developed classification and object localization dataset consisting of over 80k images. To quantify the robustness of the model, cross-validation was performed on test set images, and model robustness tests (image corruptions) were used as performance indexes. Experimentation demonstrated a 22% lift (mAP) as compared to industry standard object detectors for the ResNet50 pre-trained on ImageNet and fine-tuned with our developed classification set. Ultimately, the method described can be implemented to train future object detectors for specific tasks while achieving better accuracy results.

II. INTRODUCTION

Technological improvements over the last decade have allowed mobile robots to become predominantly autonomous and capable of performing a diverse range of tasks with limited human intervention. A critical aspect of these autonomous and intelligent systems is computer vision-based object detection [1-4]. However, the task of object detection is made very challenging due to visually complex environments containing large amounts of clutter, variations in lighting, and occlusions.

Existing approaches for object detection are based upon machine learning (ML) techniques that rely heavily on local and global feature descriptors. These techniques include Scale Invariant Feature Transform (SIFT) [5], Histogram Orient Gradient (HOG) [6], Space-time Interest points (STIP) [7] and use of linear classification models such as Support Vector Machines (SVMs). These methods, although accurate, are highly application specific, difficult to customize, and tend to perform poorly over diverse datasets. Furthermore, these methods have proven to be computationally expensive and complex to implement in real-world scenarios [8-10].

The use of neural networks for object detection has gained traction due to their capability of generalization over a large amount of data. Their independence from hand-crafted features such as SIFT, HOG, and STIP has led to large-scale adoption for computer vision tasks including object classification, detection, and segmentation [14-15]. These neural networks require training on large amounts of data to attain their optimum level of accuracy and require the labeling of large numbers of training images with objects appearing in various backgrounds and poses with bounding boxes (Figure 1). Improving upon existing recognition models is still in its infancy, partly due to the limited availability of large training datasets. Thus, the critical task of developing datasets with variation in samples is extremely important to improve the quality of these algorithms being trained.

Leading object detectors exploit the use of convolutional neural networks (CNNs) and are either trained end-to-end or fine-tuned for accurate object detection [16]. These neural networks generate hierarchical feature representations from pixel data which is “learned” from training data and allow accurate detections in complex situations. These algorithms, however, often overfit or

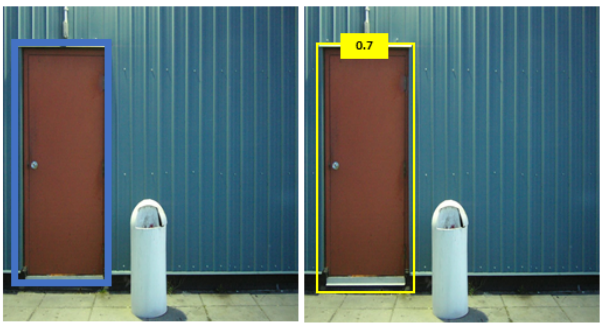


FIG. 1: An example of object detection: Ground truth label or annotated image (left) and processed image (right). Object detectors provide resulting bounding boxes and the model's confidence on resulting detection.

underfit to data making them unreliable. Thus, in addition to the need for large and diverse datasets, training these algorithms to generalize to unfamiliar objects remains a formidable challenge.

In this research paper, we analyzed the performance of three ResNet50 based models trained using a novel classification and localization dataset consisting of over 80k+ images. The novel dataset was developed using various open-source datasets and introduced new image data by crowdsourcing it from students from a local suburban high school. Three training strategies (conventional, fine-tune, and end-to-end) were used to determine the most effective model in terms of overall performance. Average Localization-Recall-Precision (ALRP) [17], a novel object detection loss function, was utilized for training the models. We aimed to identify the methodology where the object detection pipeline is optimized to have the best overall performance.

The contents of this paper are organized as follows. In Section II, we discuss our methodology. In Section III, we discuss our experimentation results. The overall findings of this research and suggestions for future work are discussed in Section IV; and we present our conclusions in Section V.

III. METHODS

In this section, descriptions of the hardware and software, the dataset used in our research, the metrics for calibration used for our CNNs, and other implementation details are described. For the purposes of this research, three ResNet-50 based models [18] were developed for experimentation.

A. Hardware and Software

The dataset and algorithms were programmed in Microsoft Visual Studio Code (version 1.5.7) and executed using Google Collaboratory Pro [19]. The host environment ran Ubuntu 18.04.5, Cuda 11.0, Python 3.9.6, and PyTorch 1.9.0. All experiments were run on NVIDIA P100 and NVIDIA Tesla T4 graphical processing units (GPUs) with 16 GBs of memory. Python was the programming language of choice because of the vast amount of computer vision and machine learning libraries available for use. PyTorch, an open-source machine-learning (ML) framework, was used to develop the models. Additional Python libraries including NumPy, Matplotlib, Python Image Library (PIL), Sea-born, Py-CM were used. Additional dependencies part of the ALRP Loss repository were also employed. All local machine work was done on a DELL i5-6780, 12-GB RAM running Windows 10 Version 20H2.

B. Dataset

Convolutional Neural Networks (CNNs) based object detectors require large amounts of annotated data for training [26], due to the sizeable number of parameters that need to be learned. For object classification and detection, image data should include variations in the object's viewpoints and other parameters such as lighting, occlusion, and clutter.

Image data for this research was gathered using the following keywords: doors, doorknobs and/or door handles, chairs, and tables. Image data for training, validation, and testing was obtained from two main sources – existing databases and a novel dataset. Over 5,000 new images were crowdsourced by student freelance photograph volunteers from a suburban high school (samples of crowdsourced images are available in Figure 2 and Appendix A). Additional images were downloaded from eight open-source datasets: Open Images V6+ (more specifically, the object detection set) [27], Door Gym [28], DeepDoors2 [29], Door-Detect Dataset [30], My Nursing Home [31], Bonn Furniture [32], Ikea-Dataset [33], and MS-COCO [34]. Class distribution is represented in Figure 3. This dataset presented variation throughout every image as data was extracted from various locations, taken under different lighting conditions (daylight/artificial light/night), varied in size, and many included extraneous objects (lamp, bike, railing, etc.).

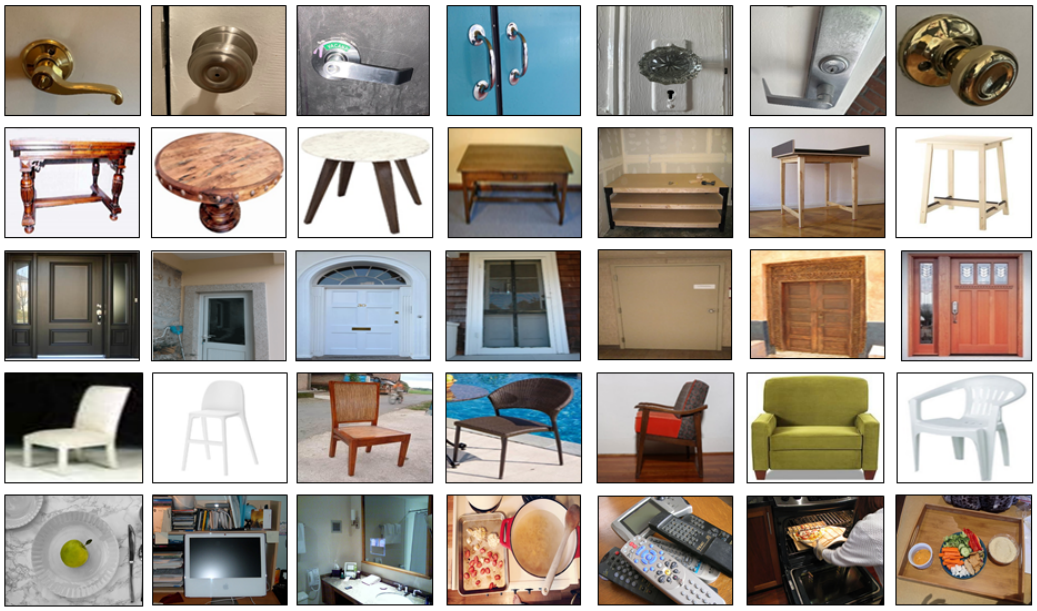


FIG. 2: Illustration of intra-class variation. Images from the compiled classification and localization dataset, which included samples with different poses, colors, lighting situations, and features.

For the classification task, an additional category with extraneous objects (not including our target objects) was added to address the possibility that none of the trained objects were present. Similarly, for the localization task, extraneous objects from the MS-COCO (Common Objects in Context) dataset were added. Extraneous objects included common household objects such as remote controls, baseball bats, teddy bears, wine glasses, bicycles, dogs, deserts, etc. (Figure 2).

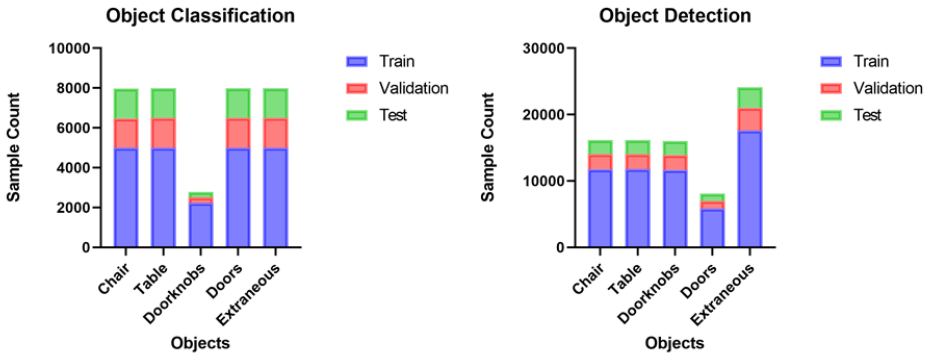


FIG. 3: Sample quantities for object classification (left) and object localization (right). Images were gathered from multiple sources detailed in Section 3.2. To address the disparity in doorknobs in the classification set, a higher weight was used while training.

To maintain consistency and to standardize the data, all images were resized to 416 x 416 using Pillow [29], a Python Imaging Library. Additionally, a Python program was developed to give each JPG or JPEG a random name (ex. 000aabd8888cfde7.jpg). To address the disparity in the doorknobs/door handles category, a higher weight was applied in the cross-entropy loss function during classification training.

All images in the dataset were manually annotated using the Robo-Flow Annotation Tool [35]. Each image was annotated using a bounding box and given one of the following labels: door, door handle, chair, or table (Figure 4). For extraneous objects downloaded through MS-COCO, labels were downloaded from the source website. The MS-COCO dataset format (JSON file) was used to store annotations locally.

All images were annotated with the following criteria: 1) The box should be drawn as tightly as possible; 2) If there are multiple instances of an object in an image, the bounding box should be placed only on all instances present in the image.

FIG. 4: Annotated Images from the Robo-Flow platofrm; annotations were downloaded using the Microsoft COCO format - JSON File.

C. Base Network: ResNet50

There are many common models such as AlexNet, GoogLeNet, and VGG-16 that have been implemented for object detection and have demonstrated their state-of-the-art results [36]. These models stack many convolutional layers with the same number of filters. As these networks

get deeper and more complex with multiple layers, problems start to arise including vanishing gradients [37] and degradation [38]. Eventually, the gradient value shrinks to 0, which causes weights to fail to update, and no learning to occur.

To address this problem, we employed Residual Networks (ResNets) which deploy skip connections (or shortcuts) to jump over layers allowing gradients to flow through unhindered. This addition performs identity mapping padded for extra dimensions and introduces no additional parameters to be trained. Due to these skip connections, ResNet50s show less training error when depth increases as compared to "plain" networks (which simply stack convolutional layers) [18]. Figure 5 illustrates these skip connections. Newer architectures compared to ResNet50s were available; however, the scope of this research was not to compare architectures but to identify a methodology for object detection that provides the best overall performance.

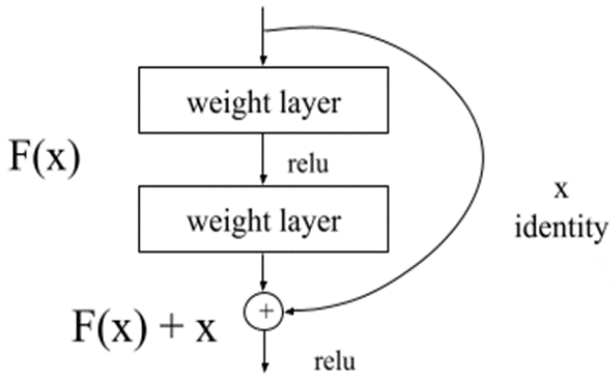


FIG. 5: A residual block; These blocks add shortcuts that skip 2 or 3 layers at a time and change how gradients are calculated for learning. In short, in some layers training is skipped due to the addition of these blocks.

In this research paper, three ResNet50 based models [18] models were employed for the object detection task. The ResNet50 structure is effective for complicated tasks and has increased detection accuracy. See Table 1 for a description of our phased training approach.

TABLE I: Phases of model training for developed models

Model	Backbone	Phase 1	Phase 2	Phase 3
Model 1	ResNet50	ImageNet	-	Developed Object Detection Set
Model 2	ResNet50	ImageNet	Developed Classification	Developed Object Detection Set
Model 3	ResNet50	Developed Classification	-	Developed Object Detection Set

D. Pre-training

Each of the three ResNet50's training was modified for experimental purposes. For Model 1, ResNet50 was downloaded from PyTorch pre-trained with the ImageNet dataset. No fine-tuning was performed for this model. For Model 2, ResNet50 was downloaded from PyTorch pretrained with the ImageNet dataset with the addition of a new classifier. For Model 3, ResNet50 was trained end-to-end with the addition of a new classifier. Random weights for Model 3 were initialized using the Kaiming weight initialization method [39].

"Network surgery" was performed on each model because ResNet50s were developed to predict 1000 classes. For our purposes, we required a prediction of only 5 classes. To modify the models, a fully connected layer (in_features = 2048, out _features=1000) was removed from the original ResNet50. This layer was replaced with a fully connected layer (in _features = 2048, out _features=5). Refer to Figure 6 for a visual representation of the ResNet50s detailed in this section.

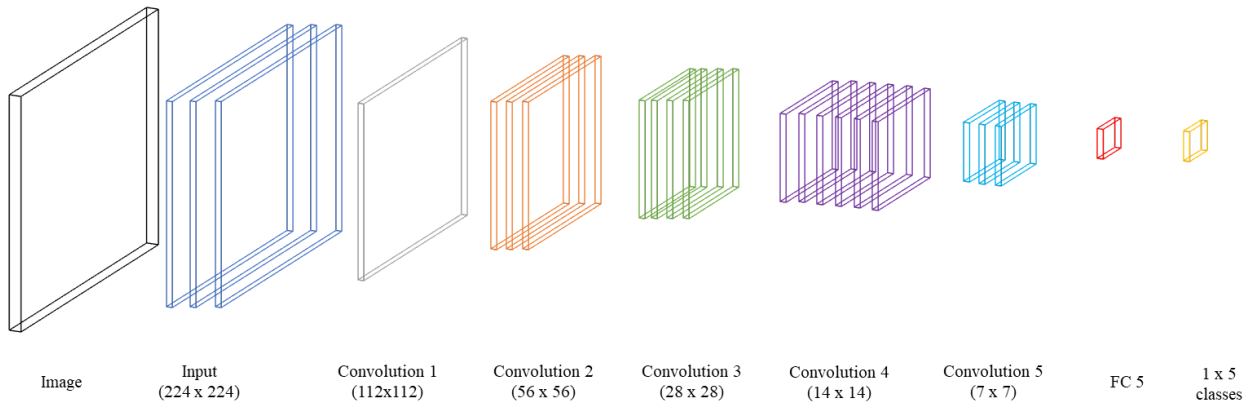


FIG. 6: ResNet50 based model with last 1000 fully connected (fc) soft max layer replaced by a 5 fully connected layer. The last layer was modified as the dataset used in this research had 5 classes, and not 1000 classes as the ImageNet dataset. The parameters of the convolutional layers were not frozen during the training process.

In the case of Model 2, the first 49 layers of ResNet-50 were unfrozen, with the addition of a new classifier. For Model 3, gradients were also not frozen as the model was not previously trained on any network. The parameters used for training are detailed in Table II. The loss function used was Cross-Entropy Loss, often referred to as log loss, measures the performance of a classifier between 0 and 1. To account for minority classes, weights were manually rescaled within the Cross Entropy Loss Function. For optimization, stochastic gradient descent (SGD)

was used, as it converges faster and can perform updates on hyperparameters frequently. Refer to Equation 1 for mathematical representation of SGD. Each model was evaluated using the validation dataset following each iteration.

$$\theta_j = \theta - n\frac{du}{dx} \tag{1}$$

Where θ_j describes the weights after update, θ and weights before update, n describes the learning rate, and $\frac{du}{dx}$ is the gradient value.

TABLE II: Parameters for classification training in models

Parameter	Model 1	Model 2	Model 3
Batch Size	-	32	32
Epoch	-	25	25
Loss Function	-	Cross Entropy	Cross Entropy
Optimizer	-	SGD	SGD
Learning Rate	-	0.01	0.01
Momentum	-	0.9	0.9

E. Object Detection Training

The average Localization-Recall-Precision (aLRP) Loss Function [17] was implemented for object detection training. aLRP is a novel ranking-based loss function which handles classification and localization errors in a unified manner. aLRP loss features one hyperparameter which does not need to be tuned compared to around 6-8 hyperparameters in other state-of-the-art loss functions. aLRP enforces high-quality localization for high-precision classification and guarantees balanced training. This loss function has shown to improve baselines for both classification and localization tasks significantly by simplifying parameter tuning and outperforming one-stage detectors.

All three models were trained using the aLRP Loss Function. The parameters used for training are detailed in Table 3. While training, the following metrics were collected: time elapsed per batch in each epoch, running loss after each batch, and memory usage.

TABLE III: Parameters for object detection training in models

Parameter	Model 1	Model 2	Model 3
Batch Size	32	32	32
Epoch	100	100	100
Optimizer	SGD	SGD	SGD
Learning Rate	0.003	0.003	0.0003
Momentum	0.9	0.9	0.9

F. Model Quality Assessment

Classification models were evaluated on the developed object classification's test set. Object classification results were quantified by F1-scores, precision, and recall.

A higher F-1 score indicates better overall model performance of the network. F1-score is calculated per **Equation 2**:

$$F1Score = \frac{2 * precision * recall}{precision + recall} \tag{2}$$

where Precision is calculated per **Equation 3**:

$$n, Precision = \frac{TP}{TP + FP} \tag{3}$$

where TP is the number of true positives, and FP is the number of False Positives. Similarly, Recall is calculated per **Equation 4**:

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

Where TP is as above, and FN is the number of False Negatives.

Object localization models were evaluated on the object localization test set. Object detection results were quantified based upon on mean Average Precision (mAP). mAP was calculated at multiple intersection over union (IoU) thresholds including 0.50:0.95, 0.50, and 0.75. IoU refers to the overlap of a predicted versus ground trurth bound box for an object. The closer the predicted bounding box values are to the ground truth, the greater the IoU value.

Higher mAP indicates performance of the network. mAP is calculated per **Equation 5**:

$$mAP = \frac{1}{n} * \sum_{k=1}^{k} = nAP_k \tag{5}$$

Where AP_k refers to the average precision of class k and n is the number of classes.

Image corruptions were also performed in order to understand the model's robustness when provided with noisy data. Seventeen corruptions were consisting of gaussian noise, shot noise, impulse noise, defocus blur, snow, frost, fog, brightness, contrast, JPEG compression, speckle noise, spatter, saturate. mAP was the metric to evaluate these image corruptions. Refer to Figure 7 for sample corruptions.

FIG. 7: Sample image corruptions using brightness, motion blur, and speckle noise. Severities include 0 (original image) to 5 (highest).

IV. EXPERIMENTAL RESULTS

A. *Classification Results*

All three ResNet50 based models were evaluated using the PYCM Confusion matrix using the developed classification test set which contained 7,129 images. Model 1 achieved an F-1 score of 0.138, Model 2 achieved an F-1 score of 0.632, and Model 3 achieved an F-1 score of 0.348. Table IV contains per class breakdown of F1-score, Precision, and Recall for all three

developed ResNet50 based models. Model 2, pretrained on ImageNet and fine-tuned on the developed classification training set, achieved a higher F-1 score (thus, higher precision and recall) compared to Model 1 and Model 3.

TABLE IV: F-1 scores, precision, and recall using the PYCM Confusion Matrix for our three ResNet50 base models. Higher values indicate better model performance for F1-scores, Precision, and Recall. The results demonstrate that Model 2 performed significantly better than Model 1 and 3 achieving an overall F1-score of 0.632, Precision of 0.622, and Recall of 0.682.

	Model 1			Model 2			Model 3		
	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall
Overall	0.138	0.11	0.198	**0.632***	**0.622***	**0.682***	0.348	0.344	0.406
Chair	0.42	0.3	0.7	**0.65***	**0.6***	**0.71***	0.32	0.29	0.35
Table	0.01	0.01	0.01	**0.3***	**0.42***	**0.23***	0.16	0.24	0.12
Doorknobs	0.01	0.03	0.02	**0.59***	**0.44***	**0.89***	0.27	0.17	0.58
Doors	0.03	0.02	0.01	**0.68***	**0.7***	**0.66***	0.46	0.49	0.44
Extraneous	0.22	0.19	0.25	**0.94***	**0.95***	**0.92***	0.53	0.53	0.54

B. Object Detection Results

The mAP (mean-average precision) of all three ResNet50 based models were evaluated upon object localization test set (10,529 image samples) at IoU thresholds 0.5:0.95, 0.5, and 0.75 and mAP was the metric. Please refer to Section II for an explanation on mAP and IoU thresholds. The conventional training method (Model 1) achieved 0.203 mAP at IoU threshold=0.5:0.95 achieving lower accuracy than Model 2. The finetuning method (Model 2) was found to have significantly higher accuracy as compared to the other models and demonstrates a 22% lift compared to Model 1 and a 33% lift to Model 3 at IoU=0.5:0.95. Model 3, trained end-to-end, achieved a mAP at 0.186 at IoU=0.5:0.95. Results demonstrate that Model 2 was the most effective training strategy.

TABLE V: All three developed ResNet50 based models were evaluated using mAP at three IoU thresholds: 0.5:0.95,0.5,0.75 using the object localization test set. Results indicated that Model 2 was the highest scoring method and performed 22% compared to Model 1 and 33% compared to Model 3.

Model	IoU=0.5:0.95	IoU=0.5	IoU=0.75
Model 1	0.203	0.358	0.202
Model 2	**0.248***	**0.398***	**0.242***
Model 3	0.186	0.292	0.186

The mAP was also evaluated for small, medium, and large objects at at IoU=0.5:0.95. Table VI contains the breakdown for small, medium, and large objects. All three models performed poorly with small objects achieving less than 0.02 mAP. Model 1 and 2 performed similarly on medium objects while Model 3 struggled with medium-sized objects. Model 2 performed significantly better with large objects than the other models achieving a 0.316 mAP.

TABLE VI: A combination of small, medium, and large objects comprised the object localization test set. The three ResNet50 based models were evaluated at IOU thresholds 0.5:0.95 using mAP. Results indicate that all three of the models performed poorly on small and medium objects. Model 2 performed the best on "large" objects.

Model	Small	Medium	Large
Model 1	0.019	**0.097***	0.272
Model 2	0.015	0.081	**0.316***
Model 3	**0.02***	0.019	0.224

C. Qualitative Results

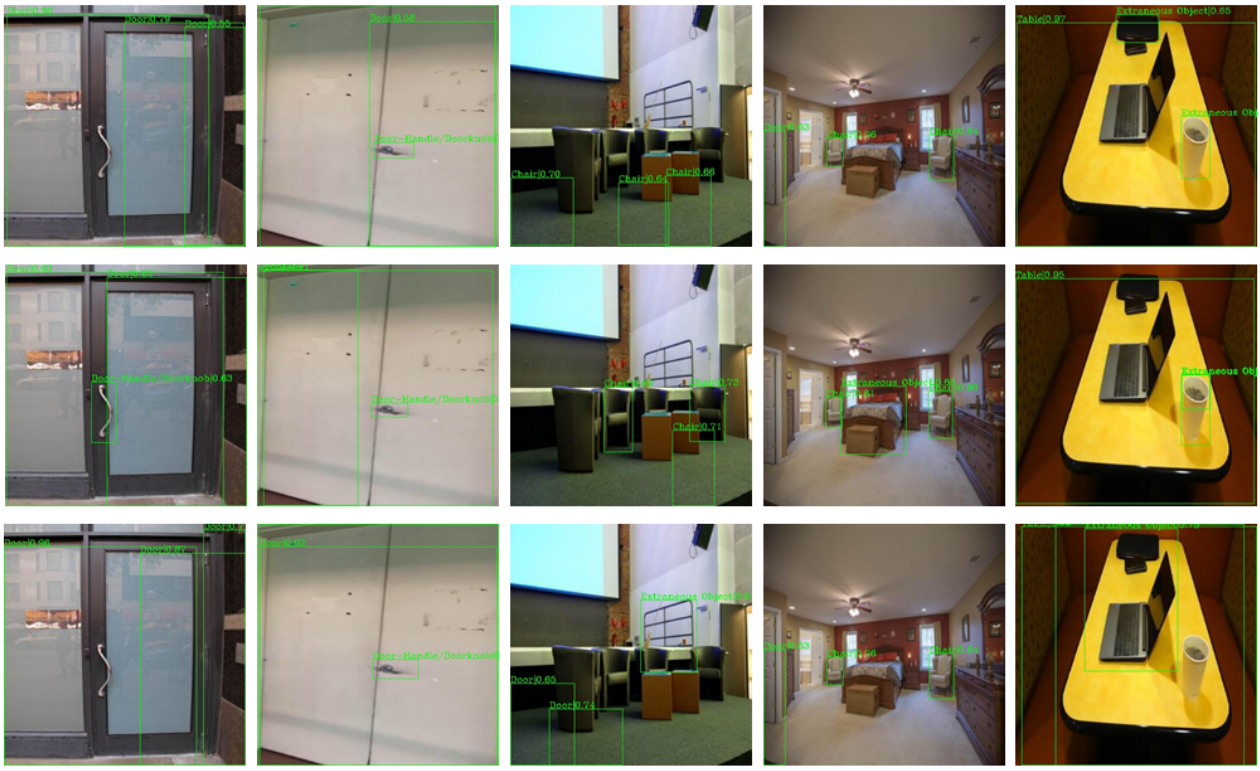


FIG. 8: Qualitative Results. Sample detection results from the three developed ResNet50 based models. Row 1 depicts results from Model 1, Row 2 depicts results from Model 2, and Row 3 depicts results from Model 3.

TABLE VII: 17 image corruptions were performed on the object localization's test set images and mAP was evaluated at IoU thresholds 0:5:0:95. mAP metrics demonstrate that Model 2 performed better than the other models in most instances.

	Model 1	Model 2	Model 3
Noise (severity=1)			
Brightness	0.218	**0.258***	0.18
Contrast	0.183	**0.223***	0.159
Defocus Blur	0.183	0.168	**0.186***
Elastic Transform	0.2	**0.246***	0.193
Fog	0.204	**0.235***	0.175
Frost	0.219	0.26*	0.16
Gaussian Blur	0.203	**0.211***	0.185
Gaussian Noise	**0.239***	0.159	0.204
Impulse Noise	0.154	0.081	**0.194***
JPEG Compression	0.225	**0.261***	0.189
Motion blur	0.194	**0.209***	0.184
Pixelate	0.257	**0.275***	0.186
Saturate	0.188	**0.237***	0.185
Shot Noise	**0.229***	0.151	0.2
Snow	0.197	**0.244***	0.095
Spatter	0.206	**0.263***	0.181
Speckle Noise	**0.233***	0.13	0.198
Noise (severity=2)			
Brightness	0.218	**0.258***	0.172
Contrast	0.176	**0.215***	0.139
Defocus blur	0.165	0.153	**0.184***
Elastic Transform	0.199	**0.234***	0.193
Fog	0.206	**0.232***	0.167
Frost	0.203	**0.207***	0.149
Gaussian Blur	0.173	0.177	**0.185***
Gaussian Noise	0.2	0.132	**0.198***
Impulse Noise	0.157	0.101	**0.186***
JPEG Compression	0.223	**0.253***	0.188
Motion blur	0.175	**0.187***	0.18
Pixelate	0.267	**0.279***	0.185
Saturate	0.178	**0.229***	0.183
Shot Noise	**0.189***	0.12	0.187
Snow	0.182	**0.205***	0.125
Spatter	0.192	**0.235***	0.107
Speckle Noise	**0.212***	0.087	0.189
Noise (severity=3)			
Brightness	0.211	**0.252***	0.161
Contrast	0.161	**0.197***	0.109
Defocus Blur	0.141	0.125	**0.179***
Elastic Transform	0.193	**0.210***	0.192
Fog	0.219	**0.245***	0.159
Frost	**0.183***	0.170	0.138
Gaussian Blur	0.147	0.148	**0.177***
Gaussian Noise	0.160	0.117	**0.178***
Impulse Noise	0.149	0.094	**0.174***
JPEG Compression	0.212	**0.242***	0.189
Motion blur	0.154	0.165	**0.169***
Pixelate	0.264	**0.283***	0.188
Saturate	0.207	**0.249***	0.193
Shot Noise	0.156	0.105	**0.157***
Snow	0.163	**0.193***	0.08
Spatter	0.178	**0.224***	0.121
Speckle Noise	**0.163***	0.025	0.151

All three models were evaluated with image corruptions performed on the test set images (see Section II for more details more about the image corruptions). Table VII illustrates the mAP of the models with the image corruptions at three noise severity levels. Our results indicate that Model 2 outperformed the other models 71% of the time at noise severity 1. At noise severity 2, Model 2 outperformed the other models 65% of the time. At noise severity 3, Model 2 outperformed the other models 52% of the time. These results indicate that even with severely corrupted images, Model 2 performed better in terms of mAP.

V. Discussion

Over the last decade, technological improvements have allowed mobile robots to become greatly autonomous and capable of performing a diverse range of tasks. In order to improve the autonomy of these mobile robots, object detection models need to be further refined in order to able to operate in complex and dynamic scenarios and consistently provide an accurate detection. This research developed and tested three deep learning methods and demonstrated a significant improvement using a particular training strategy.

This research demonstrates that Model 2's training method was the most effective strategy to train the object detection pipeline to identify objects in a visually complex scene. In our phased training approach, the classification results demonstrated that Model 2 was extremely effective in classifying objects achieving an F1-score of 0.632 which was significantly better than the other models. The object detection test results indicated that Model 2 outperformed Model 1 and 3 by 22% and 33%, respectively. This significant increase in mAP can be accredited to the transfer learning approach which refers to applying knowledge from a general domain and using it for a task-specific domain. This, in part, also helped Model 2 to generalize to unknown images as seen by the higher mAP across all three Intersection over Union (IoU) thresholds.

Interestingly, when all three models were evaluated upon small, medium, and large objects, all three models struggled with small objects. All three models did not achieve a mAP > 0.02; however, this is considered a common limitation among major object detection models (YOLOv4, EfficientDet-D2). [40] This situation, to a certain degree, is due to image resolution and how many pixels are captured within the ground truth bounding box. This creates an additional issue

as if the ground truth bounding box is not large enough, the signal produced will be small during training, and identification of these objects will be omitted. Future efforts should aim toward increasing image resolution and model input resolution to potentially increase mAP for smaller objects.

Image corruptions performed on the test set images revealed in most Model 2 outperformed Model 1 and Model 3. As noise severity levels increased, Model 2 continued to perform at a high mAP despite the image corruptions. Model 2 demonstrated a greater generalizability than Model 1 and 3 and capable of detecting through noisy conditions that it has never encountered before. Future efforts should focus on performing these image corruptions and adding them to the training set to further improve the generalizability of the models.

One key contribution of this research is the development of a novel dataset by compiling images from 8 open-source datasets and introducing a novel dataset obtained from student volunteers at a local suburban high school. Consisting of 5 classes, this dataset provided images from various locations, taken under different lighting conditions, and included extraneous objects (lamp, bike, railing, etc.) Some classes had limited sample counts and the scarcity of data possibly limited the generalizability of the models. Due to this limitation, future research must be done to expand on this dataset, more classes and varied training/validation/test data to validate these results further. The novel crowdsourced dataset will be available for public use through GitHub repository.

The method of transfer learning developed in this research could make training object detectors more robust and capable of detection objects with higher accuracy. This approach can be applied to other domains such as facial recognition and pedestrian detection.

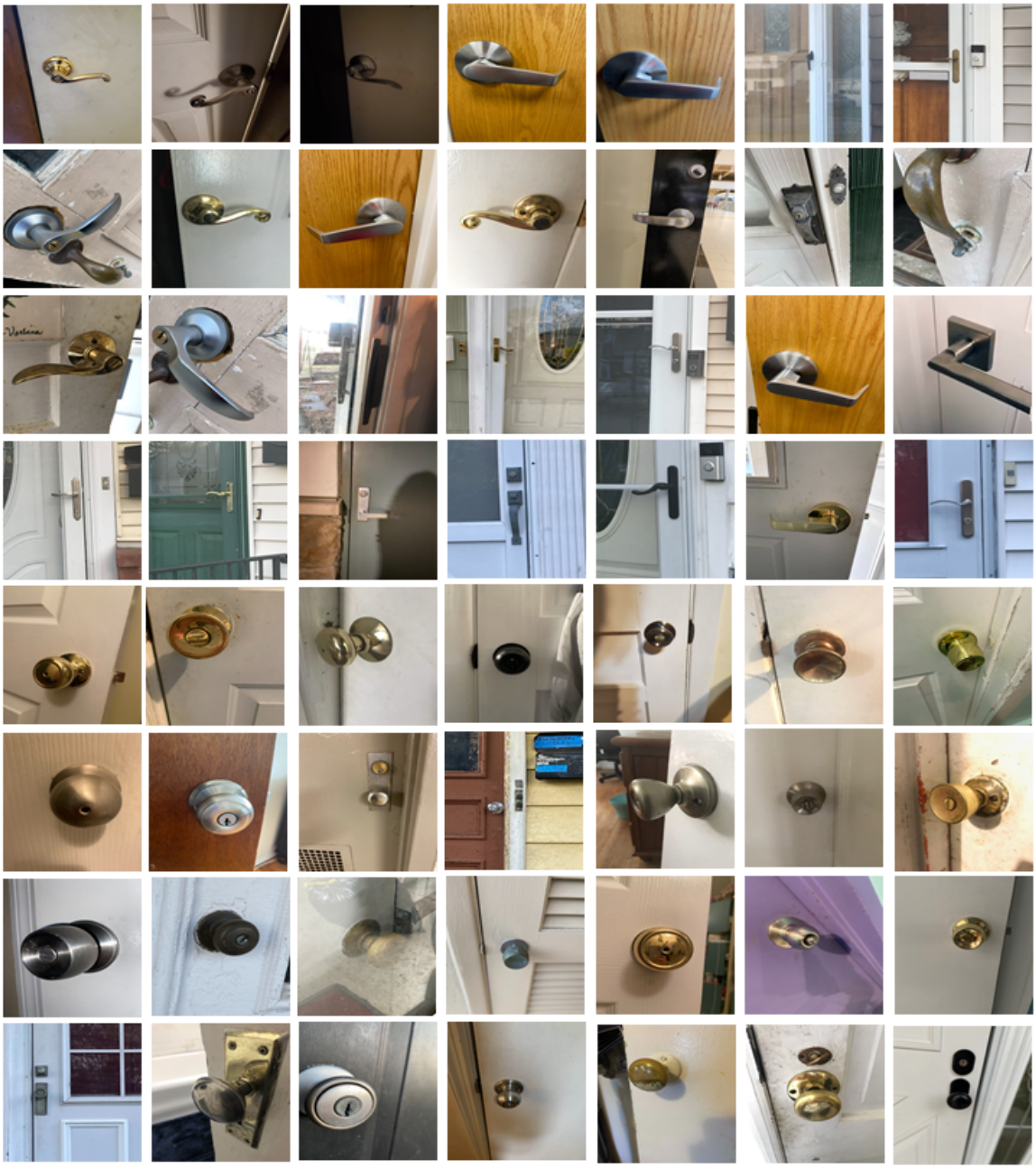
VI. Conclusion

This research investigated and improved the most effective object detection pipeline for indoor object detection in terms of overall performance (mAP). In our novel design, we developed and built a new classification and localization dataset to carry out our training for our models. The test and application experiments demonstrated that Model 2 (the transfer learning method) developed within this research improves the deep learning model compared to conventional methods of

training CNNs (Model 1 and Model 3). Model 2 shows a 22% lift in comparison to Model 1 and 33% in comparison to Model 3. The method described in this research can be implemented to train object detection algorithms for a wide array of tasks and used in robotic applications.

VII. Appendix

A. *Sample Images Collected Through Crowdsourcing*



B. *Classification Model Training Code*

```
//Data directories
data_validation = "/content/classification/validation"
data_train= "/content/classification/train"
data_test= "/content/classification/test"

//Define Transformations
transformations = transforms.Compose([
  transforms.Resize((416,416)),
  transforms.ToTensor(),
  transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])

//Perform Transformations
train_set = datasets.ImageFolder(data_train, transform = transformations)
validation_set = datasets.ImageFolder(data_validation, transform = transformations)
test_set = datasets.ImageFolder(data_test, transform = transformations)

//Define Batchsize and Input into Train/Validation/Test Loader
batchsize = 32
train_loader = torch.utils.data.DataLoader(train_set, batch_size=batchsize, shuffle=True)
validation_loader = torch.utils.data.DataLoader(validation_set, batch_size=batchsize, shuffle=False)
test_loader = torch.utils.data.DataLoader(test_set, batch_size=batchsize, shuffle=True

//Downloading Pretrained Model(Model 1 the pretrained value was set to False)
model = models.resnet50(pretrained=True)
//Model Surgery (Redefining last layer to 5 outputs)
model.fc = nn.Linear(2048, 5)

//Model Training Parameters
learn_rate = 0.01
momentum_rate = 0.9
epoch_no=25
weights = [1.0,1.0,2.4,1.0,1.0]
class_weights = torch.FloatTensor(weights).cuda()
criterion = nn.CrossEntropyLoss(weight=class_weights)
optimizer = optim.SGD(model.parameters(), lr=learn_rate, momentum=momentum_rate)

//Sending Model to GPU
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)

//Training(developed through PyTorch Docs)
for epoch in range (epoch_no):
    train_loss = 0
    val_loss = 0
    accuracy = 0
    print("Epoch:", epoch+1)
    model.train()
    counter = 0
    for inputs, labels in train_loader:
            inputs, labels = inputs.to(device), labels.to(device)
            optimizer.zero_grad()
            output = model.forward(inputs)
            loss = criterion(output, labels)
            loss.backward()
            optimizer.step()
            train_loss += loss.item()*inputs.size(0)
            counter += 1
    model.eval()
    counter = 0
    with torch.no_grad():
        for inputs, labels in validation_loader:
          inputs, labels = inputs.to(device), labels.to(device)
          output = model.forward(inputs)
          valloss = criterion(output, labels)
          val_loss += valloss.item()*inputs.size(0)
          output = torch.exp(output)
          top_p, top_class = output.topk(1, dim=1)
          equals = top_class == labels.view(*top_class.shape)
          accuracy += torch.mean(equals.type(torch.FloatTensor)).item()
          counter += 1
          print(counter, "/", len(validation_loader))
```

REFERENCES

[1] S. H. Khan, M. Hayat, M. Bennamoun, R. Togneri, F. A. Sohel, "A discriminative representation of convolutional features for indoor scene recognition," IEEE Transactions on Image Processing, vol. 25, no. 7, pp. 3372-3383, 2016.

[2] S. S. Srinivasa, D. Ferguson, C. J. Helfrich, D. Berenson, A. Collet, R. Diankov, et al., "HERB: a home exploring robotic butler," Autonomous Robots, vol. 28, no. 1, pp. 5- 20, 2010

[3] A. Ramisa, G. Alenyà, F. Moreno-Noguer, C. Torras, "Learning RGB-D descriptors of garment parts for informed robot grasping," Engineering Applications of Artificial Intelligence, vol. 35, pp. 246-258, 2014.

[4] A. Collet, D. Berenson, S. S. Srinivasa, D. Ferguson, "Object recognition and full pose registration from a single image for robotic manipulation," in Robotics and Automation, ICRA'09. IEEE International Conference on. IEEE, pp. 48-55, 2009.

[5] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, A. W. M. Smeulders, "Selective search for object recognition," International Journal of Computer Vision, vol. 104, no. 2, pp. 154-171, 2013.

[6] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, T. Poggio, "Robust Object recognition with cortex-Like mechanisms," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, no. 3, pp. 411-426, 2007.

[7] Y. LeCun, Fu Jie Huang, L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on, Vol. 2, pp. 97-104, 2004.

[8] M. Ranzato, F. J. Huang, Y. L. Boureau, Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on, pp. 1-8, 2007.

[9] L. L. Nan, K. Xie, A. Sharf, "A search-classify approach for cluttered indoor scene understanding," ACM Transactions on Graphics, vol. 31, no. 6, Article no. 137, 2012. H. Y. Wang, S. Gould, D. Roller, "Discriminative learning with latent variables for cluttered indoor scene understanding," Communications of the ACM, vol. 56, no. 4, pp. 92-99, 2013.

[10] L. D. Pero, J. Bowdish, D. Fried, B. Kermgard, E. Hartley, K. Barnard, "Bayesian geometric modeling of indoor scenes," in Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pp. 2719-2726, 2012.

[11] H. Y. Wang, S. Gould, D. Roller, "Discriminative learning with latent variables for cluttered indoor scene understanding," Communications of the ACM, vol. 56, no. 4, pp. 92-99, 2013.

[12] L. D. Pero, J. Bowdish, D. Fried, B. Kermgard, E. Hartley, K. Barnard, "Bayesian geometric modeling of indoor scenes," in Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pp. 2719-2726, 2012.

[13] M. Schwarz, H. Schulz, S. Behnke, "RGB-D object recognition and pose estimation based on pre-trained convolutional neural network features," in Robotics and Automation (ICRA), 2015 IEEE International Conference on, pp. 1329-1335, 2015.

[14] N. Neverova, C. WolfGraham, W. Taylor, F. Nebout, "Multi-scale deep learning for gesture detection and localization," in Workshop at the European Conference on Computer Vision, 2014, ECCV 2014, Springer International Publishing, pp. 474-490, 2014

[15] C. Couprie, C. Farabet, L. Najman, Y. LeCun, "Indoor semantic segmentation using depth information," arXiv preprint arXiv:1301.3572, pp. 1-8, 2013.

[16] S. Gupta, R. Girshick, P. Arbeláez, J. Malik, "Learning rich features from RGB-D images for object detection and segmentation," in European Conference on Computer Vision, ECCV 2014, Springer International Publishing, pp. 345-360, 2014.

[17] Oksuz, Kemal, et al. "A ranking-based, balanced loss function unifying classification and localisation in object detection." arXiv preprint arXiv:2009.13592 (2020).

[18] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[19] Bisong E. (2019) Google Colaboratory. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform. Apress, Berkeley, CA.

[20] PyTorch, pytorch.org/.

[21] NumPy, numpy.org/.

[22] “Visualization with Python.” Matplotlib, matplotlib.org/.

[23] Pillow, pillow.readthedocs.io/en/stable/.

[24] “Statistical Data Visualization.” Seaborn, seaborn.pydata.org/.

[25] “PyCM.” Http://Www.pycm.ir, www.pycm.ir/.

[26] Gu, Jiuxiang, et al. "Recent advances in convolutional neural networks." Pattern Recognition 77 (2018): 354-377.

[27] Kuznetsova, Alina, et al.; The open images dataset v4." International Journal of Computer Vision (2020): 1-26.

[28] Urakami, Yusuke, et al."Doorgym: A scalable door opening environment and baseline agent." arXiv preprint arXiv:1908.01887 (2019).

[29] Gasparramoa. (n.d.). Gasparramoa/deepdoors2. Retrieved March 29, 2021, from https://github.com/gasparramoa/DeepDoors2

[30] MiguelARD. (n.d.). MiguelARD/DoorDetect-Dataset. Retrieved March 29, 2021, from https://github.com/MiguelARD/DoorDetect-Dataset

[31] Ismail, asmida (2020), “MyNursingHome”, Mendeley Data, V1, doi: 10.17632/fpctx3svzd.2021, https://cvml.comp.nus.edu.sg/furniture/index.html.

[32] Aggarwal, Divyansh, et al. "Learning style compatibility for furniture." German Conference on Pattern Recognition. Springer, Cham, 2018.

[33] "Ivonatau/Ikea";. Github, 2021, https://github.com/IvonaTau/ikea.

[34] "Common Objects in Context." COCO, cocodataset.org.

[35] Clark, A. (2015). Pillow (PIL Fork) Documentation. readthedocs. Retrieved from https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf

[36] “Give Your Software the Power to See Objects in Images and Video.” Roboflow, roboflow.com/.

[37] Habibzadeh M, Jannesari M, Rezaei Z, Baharvand H, Totonchi M. Automatic white blood cell classification using pre-trained deep learning models: Resnet and inception. InTenth International Conference on Machine Vision (ICMV 2017) 2018 Apr 13 (Vol. 10696, p. 1069612). International Society for Optics and Photonics.

[38] Wichrowska O, Maheswaranathan N, Hoffman MW, Colmenarejo SG, Denil M, de Freitas N, Sohl-Dickstein J. Learned

optimizers that scale and generalize. InProceedings of the 34th International Conference on Machine Learning-Volume 70 2017 Aug 6 (pp. 3751-3760). JMLR. org.

[39] He, Kaiming, et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." Proceedings of the IEEE international conference on computer vision. 2015.

[40] Solawetz, Jacob. "Tackling the Small Object Problem in Object Detection." Roboflow Blog, Roboflow Blog, 27 Oct. 2021, blog.roboflow.com/detect-small-objects/.