Alex Cohen

Understanding the Effects of Distortion on Object Detection Models

George Washington University, Data Science Program

alexcohen@gwu.edu

Abstract

While much work has been done to understand the effect of distorted visual input on image classification models, there has been little exploration of the same effects on object detection models. This study attempts to understand the impact of varying degrees of two common types of distortion (image blur and noise injection) on the ability to identify objects in the Microsoft Common Objects in Context dataset. This experiment involves applying distortions of two types and three different magnitudes and observing the changes in object detection task performance across different object sizes. Additionally, this study examines the impact that targeted fine-tuning has on the Faster R-CNN object detection model, and quantifies the improvements achieved by focusing on the feature extractor, Region Proposal Network and pooling layers, and overall model. This study shows immediate decreases in performance with the slightest injection of distortion, highlighting the limited robustness of common pretrained object detection models, and identifies the feature extraction layers as the component most easily fine-tuned to mitigate this decrease in performance at higher levels of distortion.

Keywords: Object Detection, Image Distortion, Faster R-CNN, Computer Vision

Disclosures

Funding: Not applicable

Conflicts of Interest/Competing Interests: Not applicable

Availability of data and material: public at https://cocodataset.org/#download

Code Availability: public at https://github.com/alexjcohen/Capstone/tree/master/Code

Introduction

Computer vision work reached a turning point in the early 2010s with the introduction of deep convolutional neural networks in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC). [1] The innovation of this challenge was the use of a common, large-scale, and easily accessible set of everyday images from which image classifiers could be trained and evaluated. This challenge, which ended in 2017, pushed the idea that the data on which models are trained is just as important as the models themselves. In parallel to this change, the entry of the AlexNet model in the 2012 competition spurred renewed interest in deep learning for image classification. AlexNet far outpaced the rest of the competition, winning the 2012 ImageNet challenge and producing an error rate over 10% lower than the second-place competitor when using a deep neural network (DNN) based approach. [2] Thus, the field of computer vision was revisited with a renewed interest.

There are two main resources to rely on when undertaking this study: understanding the effect of distortion on other computer vision tasks, and further studies involving the state-of-the-art Faster R-CNN model, which is the model used in this study. Prior work evaluating the effects of noise and distortion on other computer vision tasks can provide an understanding of differing techniques and potential effects. For example, in In Deep Learning with Noise [3] Lou and Yang showed that more complex neural networks are better able to absorb input noise when comparing small noise injections into smaller machine learning classifiers – including Logistic Regression, a Multi-Layer Perceptron (MLP), and Convolutional Neural Networks. Expanding further, Nguyen et al. have shown that DNNs can be easily fooled and images can be created that are unidentifiable to humans, yet machines can classify them with over 99% confidence [4], highlighting the gap that still exists between human and computer performance on visual tasks. A major study done on the effect that distortion has on image classification was performed by Dodge and Karam in 2016, [5] which explored the susceptibility of various CNN architectures to distortions such as gaussian blur, noise injection, varying contrast levels, and JPEG image quality. The most comparable study was performed by Zhou et al. in their paper *On Classification of Distorted Images with Deep Convolutional Networks* [6], which not only quantified the impact of different manners of distortion on image classification models, but suggested training strategies to mitigate the corresponding loss in performance when presented with distorted input. All this related work, however, focuses on distortion in the context of image classification models, whereas this study focuses on image distortion in object detection tasks.

Additional work involving the Faster R-CNN model include the application of the Faster R-CNN to 360^{o} images [7], pre-processing techniques to mitigate distortion effects [8], the identification of locations containing motion blur or distortion in images for additional pre-processing [9], and the study of rain's impact on camera-based object detection [10]. Furthermore, studies have been conducted on understanding the impact of training imbalance on Faster R-CNN detection capabilities [11], as well as extending the Faster R-CNN model to the generation of object masks using the Mask R-CNN model [12].

This paper proposes a model training and evaluation strategy for the Faster R-CNN model based on the injection of two different kinds of noise: Gaussian Blur (substitute for motion blur), and Randomized Pixelated Noise Injection (substitute for image corruption or extreme temperatures/lighting) to understand performance impacts. Additionally, model refinement was performed in an attempt to both understand which model components were most impacted by distortion and strategies to mitigate performance changes with the addition of various sources of distortion.

This paper is outlined as follows: Section 2 introduces the data that is used during this experiment. The overview of the Faster R-CNN model is presented in Section 3 and the experimental framework in Section 4. Section 5 details the results of the experiment. Finally, Section 6 discusses the conclusions of this paper and identifies future work.

Object Detection Data

The images all come from the Microsoft Common Objects in Context (COCO) dataset, which was developed specifically to advance the state-of-the-art in object recognition by placing objects in their everyday scenes.[13] The entire dataset contains over 330,000 images and 1.5 million object instances across over 80 different object categories. The specific categories used in this project include:

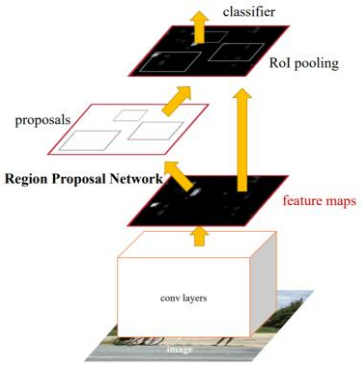
person, bicycle, car, motorcycle, airplane, bus,
train, truck, boat, traffic light, fire hydrant, stop sign,
parking meter, bench, bird, cat, dog, horse, sheep, cow,
elephant, bear, zebra, giraffe, backpack, umbrella,
handbag, tie, suitcase, frisbee, skis, snowboard, sports ball,
kite, baseball bat, baseball glove, skateboard, surfboard, tennis racket,
bottle, wine glass, cup, fork, knife, spoon, bowl,
banana, apple, sandwich, orange, broccoli, carrot, hot dog, pizza,
donut, cake, chair, couch, potted plant, bed, dining table,
toilet, tv, laptop, mouse, remote, keyboard, cell phone,
microwave, oven, toaster, sink, refrigerator, book,
clock, vase, scissors, teddy bear, hair drier, toothbrush

For this specific project, only the 2017 TRAIN and VAL image sets were used. The TRAIN image set consists of 141,000 images, and the VAL image set consists of 5,000 images. The VAL set is used to determine the baseline object detection scores, the detection scores after distortions, and the scores on both unaltered and distorted images when using a fine-tuned model. The TRAIN image set is used for refining the pretrained object detection models, with subsets of various sizes being taken to determine the optimal training parameters.

Model Overview

The model selected for this experiment is the Faster R-CNN framework, first introduced by Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun in 2016 [14]. This model used as its origin the R-CNN first developed by Girshick et al. in 2014 [15] and then further improved by Girshick et al. in 2015 with the introduction of the Fast R-CNN model [16]. The main improvement on the previous versions of region-based model was the introduction of the Region Proposal Network (RPN) for the identification of regions of interest within the underlying image. By training a second network instead of using more traditional selective search algorithms, the authors were able to achieve object detection results at 17 frames per second (with the ZF-Net backbone) instead of the previous 0.5 frames per second, enabling near real-time capabilities [14]. The model training in this paper relied on the PASCAL VOC 2007 data for training, without mention of the types of transformations performed during the model development process. [14]

Fig. 1 [14]



The Faster R-CNN model is composed of three primary components: the feature map backbone, the Region Proposal Network (RPN), and the Region of Interest (RoI) pooling/output layer. In the PyTorch implementation, which was used in this study, the feature map backbone is a ResNet-50 [18], with the final flattening and softmax layers removed to output just the feature-maps for further use in the model [17]. On top of these feature maps, the RPN generates predicted bounding boxes for objects using a series of anchor points, and then a classifier is trained to predict the “object-ness” of the contents of each bounding box and a regressor for the bounding-box adjustments

[14]. This RPN is trained using stochastic gradient decent and a similar multi-task loss as was first demonstrated in the Fast-RCNN model [15], and shown below:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*).$$

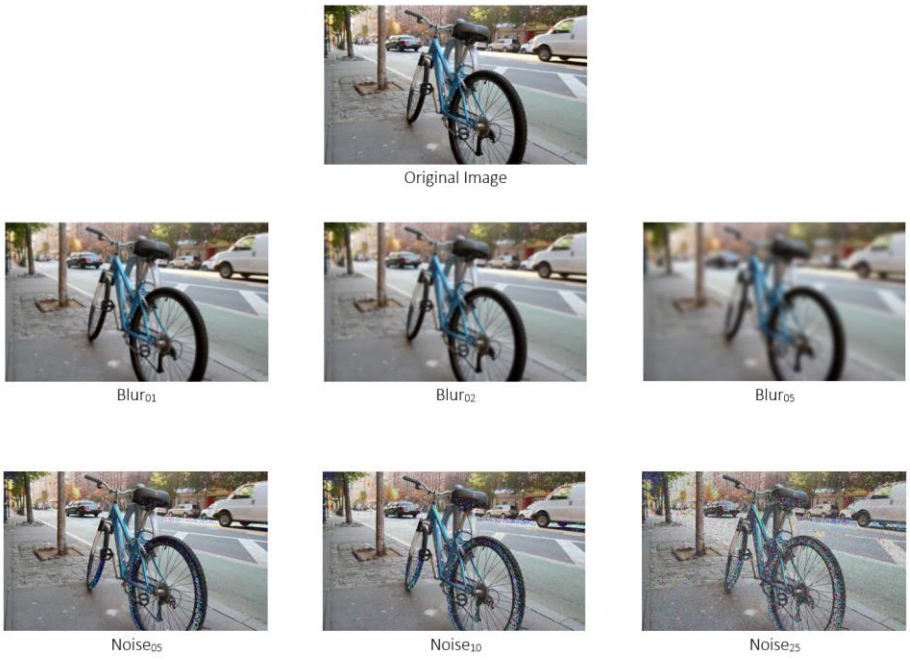
The loss is a weighted combination of the binary cross-entropy loss of classifying the "object-ness" of a particular bounding box (L_{cls}) and the smooth L1 loss of the bounding box adjustments on only non-background classes (L_{reg}) [14]. Once learned, the RPN outputs regions likely to contain an object (after reducing the total number of proposals using non-maximum suppression [14]), which are then combined in the RoI layer with the previously extracted feature maps, flattened using a fully connected layer, and then passed to the final classifiers and bounding-box regressors to generate the final object class and location predictions, which is also trained using a similar multi-task loss function to the RPN [14]. While the final model layers are quite similar to that of the Fast R-CNN [15], the key improvement of the Faster R-CNN is the implementation of the RPN to more efficiently generate the proposed regions for final classification and location identification.

Experimental Framework

There are two phases to this project: the exploration and measurement of baseline values, and then the attempt to improve upon those metrics by finetuning using transfer learning to improve model performance on distorted images. The initial phase of baseline performance measurement requires passing images with varying levels of distortion to the pretrained Faster R-CNN model, and capturing the mAP across all validation pictures. The specific images used for this task were the VAL 2017 set of Microsoft COCO images – containing 5,000 photos. Two different types of distortion were used in this study: blur and noise injection.

The blur transformation is more straightforward and takes an input image tensor and applies gaussian blur to the image to increase the level of distortion. The randomized pixelated noise injection applies a mask of random pixel value fluctuations to the original input image. First, the input image is represented as a three-dimensional array (w x h x color channels). From there, three equally sized masks are created using a random normal array of size w x h, with a mean of zero and standard deviation of *sigma*, and this mask is added to each of the original image channels. To avoid overflow (since pixels can only have values between 0 and 255), if the resulting sum for each pixel is less than zero or more than 255, the mask is "flipped" to instead subtract the pixel mask value, effectively clipping the noise injection. The radii used to generate the varying levels of blur were 1 px (low), 2 px (medium) and 5 px (high); the standard deviation of pixel values used to generate the varying levels of noise were 5 (low), 10 (medium), and 25 (high). Figure 2 shows examples of these transformations across the varying levels of distortion.

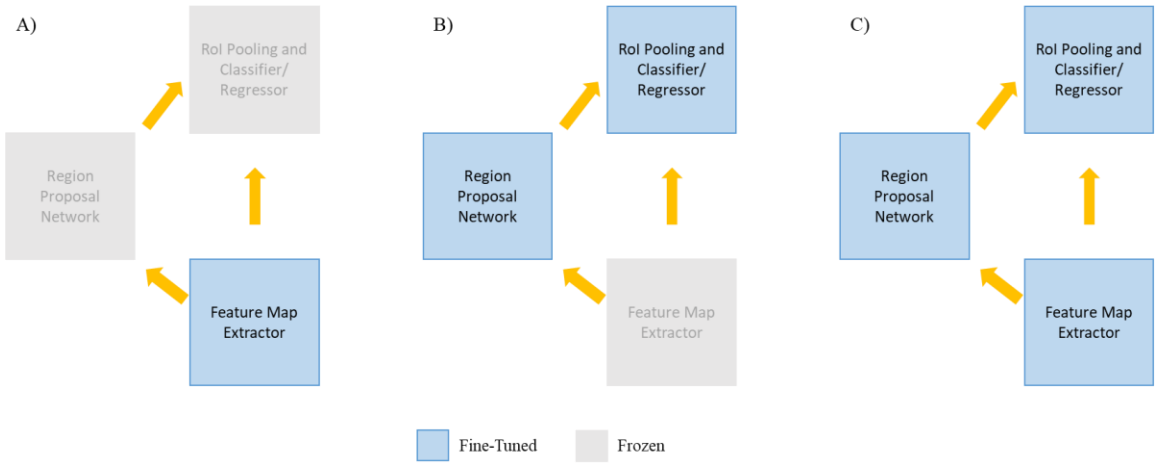
Fig 2.



Evaluating results involves calculating the mean Average Precision (mAP) of object detection and classification tasks for the given model, which is the metric by which object detection models are scored in the annual COCO challenge. This process involves calculating the precision and recall of the predicted bounding boxes across differing areas of overlap, calculated using the Intersect over Union (IoU) metric. The IoU metric required for inclusion in the calculation was varied between 0.50 and 0.95 at increments of 0.05, meaning there are 10 categories of predictions across all 90 object categories (not counting the background class) that are all averaged to produce the final mAP value. This is the scoring method used in the COCO challenge [13], and is similar to the mAP at solely an IoU threshold of 0.5 used in the PASCAL VOC challenge [19].

Once the baseline metrics are calculated for varying levels of noise injection on the pretrained model, it was then fine-tuned in order to see if additional training on blurred and noise-injected images would improve object detection capabilities when attempting to detect objects in distorted input. Different components of the model have their associated weights frozen during training to limit weight updates to specific layers within the model and better understand component effects. The three different fine-tuning rounds included a) isolated feature extraction b) isolated RPN/model head c) full model. Various combination of hyperparameters were used to differing levels of success – which are discussed further in the results section.

Fig. 3.



It is worth noting that this strategy is different than typical approaches when training models on noisy or distorted data. Traditional computer vision tasks often attempt to sharpen, refine, or improve the images used prior to predictions [8-9, 11], or train models using simpler distorted involving image rotations and transformations such as contrast adjustments. This approach looks to improve the object detection model weights directly using this distorted input.

Discussion of Results

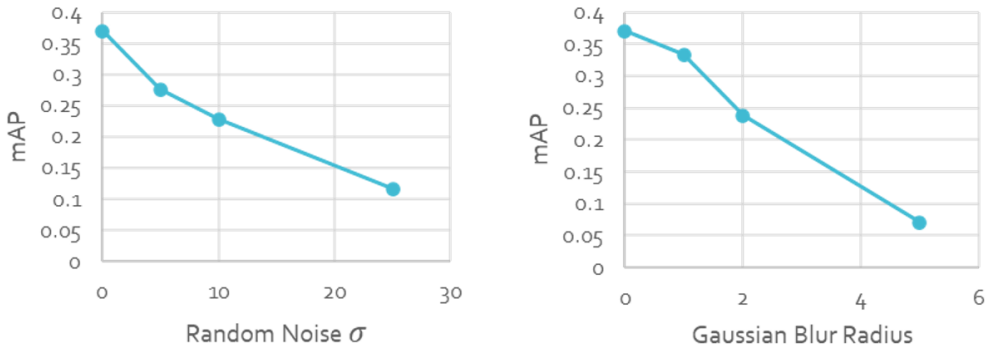
These results are further broken down by object size, measured in the number of pixels in the image the object occupies. All four of these metrics (mAP, and mAP across small, medium, and large objects) are reported for the baseline model (without any fine tuning or additional training) across the undistorted images, and images distorted varying amounts. Additionally, any subscripts, e.g. $blur_{01}$ or $noise_{05}$, represent the gaussian blur radius and the standard deviation of the randomly pixelated noise injected, respectively. The metric is reported across all object classes and all IoU thresholds between 0.5 and 0.95 at 0.05-unit increments.

Table 1.

Evaluation of Pre-Trained Faster R-CNN on Distorted Images

	AP[0.50:0.95:0.05] All	*AP[0.50:0.95:0.05] Small (< 32^2px)*	*AP[0.50:0.95:0.05] Medium (< 96^2px)*	*AP[0.50:0.95:0.05] Large (> 96^2px)*
Baseline	**0.370**	**0.211**	**0.403**	**0.482**
$Noise_{05}$	0.276	0.142	0.300	0.385
$Noise_{10}$	0.228	0.103	0.246	0.325
$Noise_{25}$	0.116	0.036	0.122	0.197
$Blur_{01}$	0.333	0.161	0.368	0.463
$Blur_{02}$	0.238	0.079	0.256	0.389
$BIur_{05}$	0.071	0.006	0.058	0.161

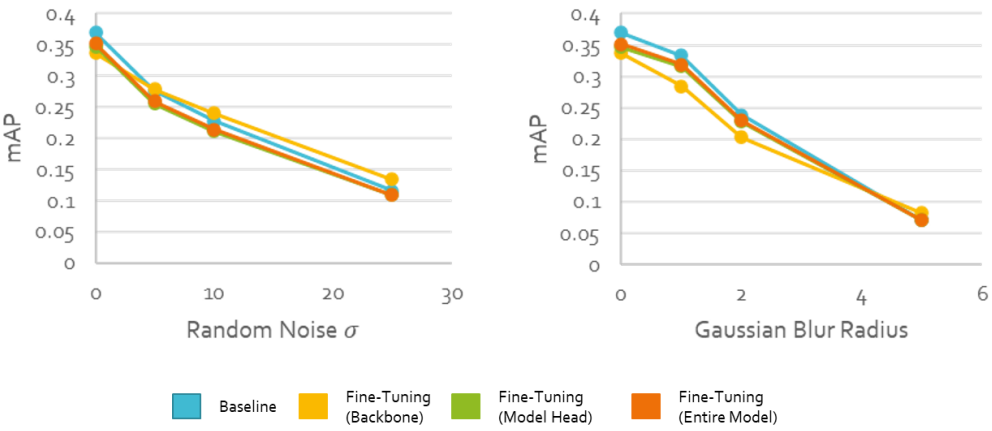
Fig 4.



There is an immediate, rapid decrease in object detection performance upon the slightest injection of distortion. With minimal random-noise injection, the overall model performance decreases by approximately 25%, and this is amplified for smaller objects in images (decrease of approximately 33%). A similar pattern is observed for the blur transformations; however, the decrease is not as drastic both overall and for each object size, with an overall performance decrease of just 10% (decrease of 24% for small objects). While minor noise seems to have a larger effect than minor blur, the highest level of noise and blur injections show the opposite property – the maximum level of blur distortion used showed an overall performance decrease of 81%, whereas the maximum level of noise injection showed a precision decrease of just 69% across the 5,000 validation images.

In an attempt to mitigate this decrease in performance, model fine-tuning was performed using distorted images as input. The fine-tuning was performed in three stages: fine-tuning the feature extractor (ResNet-50 backbone), fine-tuning the RPN and RoI Pooling/final classification layers, and fine-tuning the entire model. The results of each are shown below:

Fig 5.

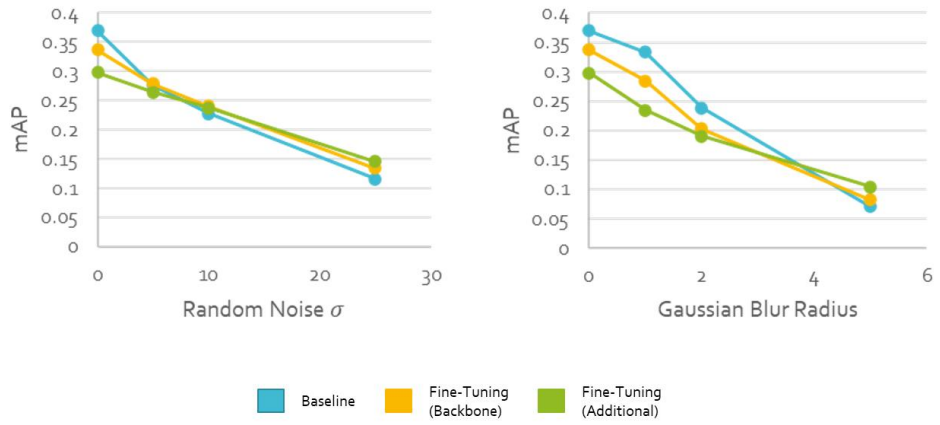


Across the different fine-tuning methods, each done with constant training parameters, differences begin to appear as various components of the model are more sensitive to refinement than others, as well as the two different distortion techniques. Overall, refinement of the model backbone (the ResNet50 feature extractor in the PyTorch implementation [18]), when fine-tuned, is best able to mitigate the decrease in performance associated with random noise injection at levels of distortion above zero. This refinement outperforms the baseline model (without fine-tuning on distorted input) across all levels of random pixel noise injection, except for the unaltered images. Fine-tuning the backbone is the only refinement technique that outperforms the baseline model when noise is injected, as the full-model refinement and model head refinement both show performance decreases when compared to the unrefined model. The conclusion that can be drawn from this is that refining the model head (the RPN and then

pooling/final predictive layers) is the component of the model that responds most poorly to noise injection and causes a larger decrease in performance when fine-tuned on distorted images.

Opposite conclusions appear when locating objects in images transformed with Gaussian blur. The model backbone fine-tuning appears to accelerate the decrease in performance as lower-levels of blur when compared to the baseline model (e.g. mAP of 0.284 for fine-tuned backbone vs baseline of 0.333 with $blur_{01}$). The performance of the model-head-refinement and fully-refined model show more comparable performance (albeit slightly worse) to the baseline model, which is an opposite result to the noise-injected images. However, at the highest levels of distortion recorded ($blur_{05}$), isolating fine-tuning to just the feature-extraction backbone outperforms all other refinement techniques as well as the baseline model, proving to mitigate more of the performance decrease than other techniques. This result means there appears to be a trade-off with blurred object detection: improved performance at lower levels of distortion with a minimally refined model (or simply use of the baseline model), and improved performance with the backbone-refined model at higher levels of distortion. This appears to be a counter-result to the noise-injection results, where the backbone-refined model outperforms all other models at all levels of distortion (with comparable or slightly decreased performance with no distortion). For this reason, the impact of additional backbone fine-tuning is also explored.

Fig. 6.



For the noise-injected data, the increased training (10x the training data) appears to cause the model to greatly underperform the baseline model at lower levels of distortion and outperform other techniques at higher levels of distortion. This runs counter to the previous results, which found that fine-tuning just the model backbone resulted in comparable results at lower levels of noise injection and improved performance at higher distortion levels for both random pixel noise and blur. For images distorted with pixelated noise, there is a noticeable decrease in performance at lower distortion levels, comparable performance at intermediate levels, and then a further outperformance at higher levels of random pixelated noise.

With the blurred images, similar underperformance is noted at lower levels of distortion, however the degree to which the model underperforms is only increased with the additional refinement. In images without distortion, the mAP value drops from 0.37 (baseline) and 0.337 (slight fine-tuning) to 0.298 (extended fine-tuning), a decrease of slightly over 20%. This decrease in performance continues at low levels of distortion ($blur_{01}$), yet becomes comparable to the slightly-refined model at medium levels of distortion. At the highest levels of image blur recorded ($blur_{05}$), the model further improves its performance over the moderately-refined model, improving the mAP from 0.082 to 0.104 – an increase of approximately 25%. The additional refinement again shows that there is a tradeoff in model performance – the more fine-tuned models tend to outperform the pretrained/baseline model at these higher levels of distortion, whereas they underperform or only perform comparably to that baseline at lower levels of distortion.

Conclusions and Future Work

This paper proposes the use of fine-tuned Faster R-CNN models for the purposes of object detection at higher levels of distortion, specifically with Gaussian blur due to motion within images and pixelated noise injection due to high temperatures or image corruption. The experimental results show the improvements in performance that can be achieved at higher levels of distortion when using a Faster R-CNN model which has a refined feature extraction backbone trained on distorted input, albeit with decreased performance at lower levels of distortion. Future work will involve comparing changes in performance when using different model backbones (instead of the ResNet-50), as well as determining if these results generalize to other families of object detection models.

References:

[1] Russakovsky, Olga, et al. “ImageNet Large Scale Visual Recognition Challenge.” International Journal of Computer Vision, vol. 115, no. 3, 2015, pp. 211–252., doi:10.1007/s11263-015-0816-y.

[2] Krizhevsky, Alex, et al. “ImageNet Classification with Deep Convolutional Neural Networks.” Communications of the ACM, vol. 60, no. 6, 2017, pp. 84–90., doi:10.1145/3065386.

[3] Luo, Yixin and Fan Yang. “Deep Learning With Noise.” (2014).

[4] Nguyen, Anh, et al. “Deep Neural Networks Are Easily Fooled: High Confidence Predictions for Unrecognizable Images.” 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, doi:10.1109/cvpr.2015.7298640.

[5] Dodge, Samuel, and Lina Karam. “Understanding How Image Quality Affects Deep Neural Networks.” 2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX), 2016, doi:10.1109/qomex.2016.7498955.

[6] Zhou, Yiren, et al. “On Classification of Distorted Images with Deep Convolutional Neural Networks.” 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017, doi:10.1109/icassp.2017.7952349.

[7] Wang, Kuan-Hsun, and Shang-Hong Lai. “Object Detection in Curved Space for 360-Degree Camera.” ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019, doi:10.1109/icassp.2019.8683093.

[8] Yang, Ruiduo; Shih, Yichang; Liang, Chia-Kai; and Hasinoff, Sam, "Improved Object Detection in an Image by Correcting Regions with Distortion", Technical Disclosure Commons, (April 01, 2020) https://www.tdcommons.org/dpubs_series/3090

[9] Daisuke Uchida, Atsushi Yamashita, and Hajime Asama "Detecting image frames which contain a moving object from a severely distorted video stream using dynamic mode decomposition", Proc. SPIE 11515, International Workshop on Advanced Imaging Technology (IWAIT) 2020, 1151510 (1 June 2020); https://doi.org/10.1117/12.2566797

[10] Hamzeh, Y., El-Shair, Z., and Rawashdeh, S., "Effect of Adherent Rain on Vision-Based Object Detection Algorithms," SAE Technical Paper 2020-01-0104, 2020, https://doi.org/10.4271/2020-01-0104.r

[11] Chen, Yuhua, et al. “Domain Adaptive Faster R-CNN for Object Detection in the Wild.” 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, doi:10.1109/cvpr.2018.00352.

[12] He, Kaiming, et al. “Mask R-CNN.” 2017 IEEE International Conference on Computer Vision (ICCV), 2017, doi:10.1109/iccv.2017.322.

[13] Lin, Tsung-Yi, et al. “Microsoft COCO: Common Objects in Context.” Computer Vision – ECCV 2014 Lecture Notes in Computer Science, 2014, pp. 740–755., doi:10.1007/978-3-319-10602-1_48.

[14] Ren, Shaoqing, et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, 2017, pp. 1137–1149., doi:10.1109/tpami.2016.2577031.

[15] Girshick, Ross, et al. “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation.” 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, doi:10.1109/cvpr.2014.81.

[16] Girshick, Ross. “Fast R-CNN.” 2015 IEEE International Conference on Computer Vision (ICCV), 2015, doi:10.1109/iccv.2015.169.

[17] He, Kaiming, et al. “Deep Residual Learning for Image Recognition.” ArXiv.org, 10 Dec. 2015, arxiv.org/abs/1512.03385.

[18] “Torchvision.models.” Torchvision.models - PyTorch 1.6.0 Documentation, pytorch.org/docs/stable/torchvision/models.html.

[19] Everingham, Mark, et al. “The Pascal Visual Object Classes (VOC) Challenge.” International Journal of Computer Vision, vol. 88, no. 2, 2009, pp. 303–338., doi:10.1007/s11263-009-0275-4.