Understanding the Effects of Distortion on Object Detection Models

Alex Cohen

Overview

- Introduction
- Problem Statement
- Relevant Work
- Project Workflow
- Data Overview
- Model Overview
- Methodology and Evaluation
- Baseline Results
- Fine-Tuning Overview
- Fine-Tuning Results
- Conclusions and Next Steps

Introduction:

Object Detection and Computer Vision

- Computer vision as a field gained a renewed interest after AlexNet won the 2012 ILSVLC (ImageNet) challenge, with an error rate 10% lower than the second-place finisher by using a Deep Neural Net (CNN)-based approach
- Since then, image classification and object detection models have become incredibly common in daily life
    - Object detection has many daily uses, from license plate readers to self-driving car to crowd-counting to video surveillance, with more applications surely on the way
- Historically, object detection was done using pattern-matching and machine learning based approaches, but deep neural networks have become the state-of-the-art in the field

What is Object Detection?

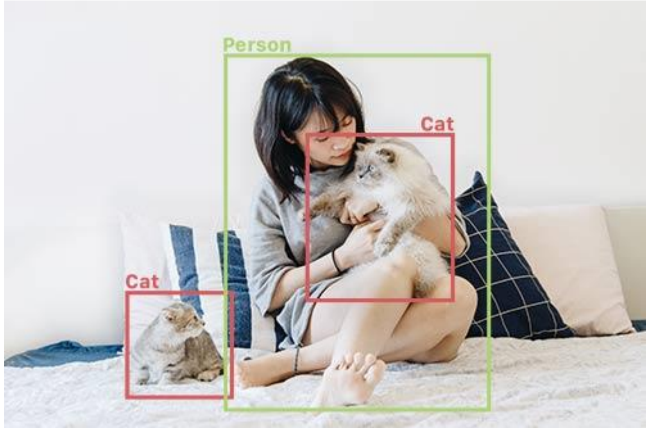
Image classification models attempt to identify the category of object is present in an image

Object detection models attempt to identify and locate one or more objects within an image

Object detection combines the entity labeling of image classification models with a new feature – predicting the object locations within the image, thus require a more complex model architecture

Class='Cat'

Image Classification



Object Detection

Problem Statement

How do different forms of input distortion, specifically noise and blur injection, impact the performance of object detection models, and can different model components be refined to better respond to distorted input?

(Some) Relevant Publications

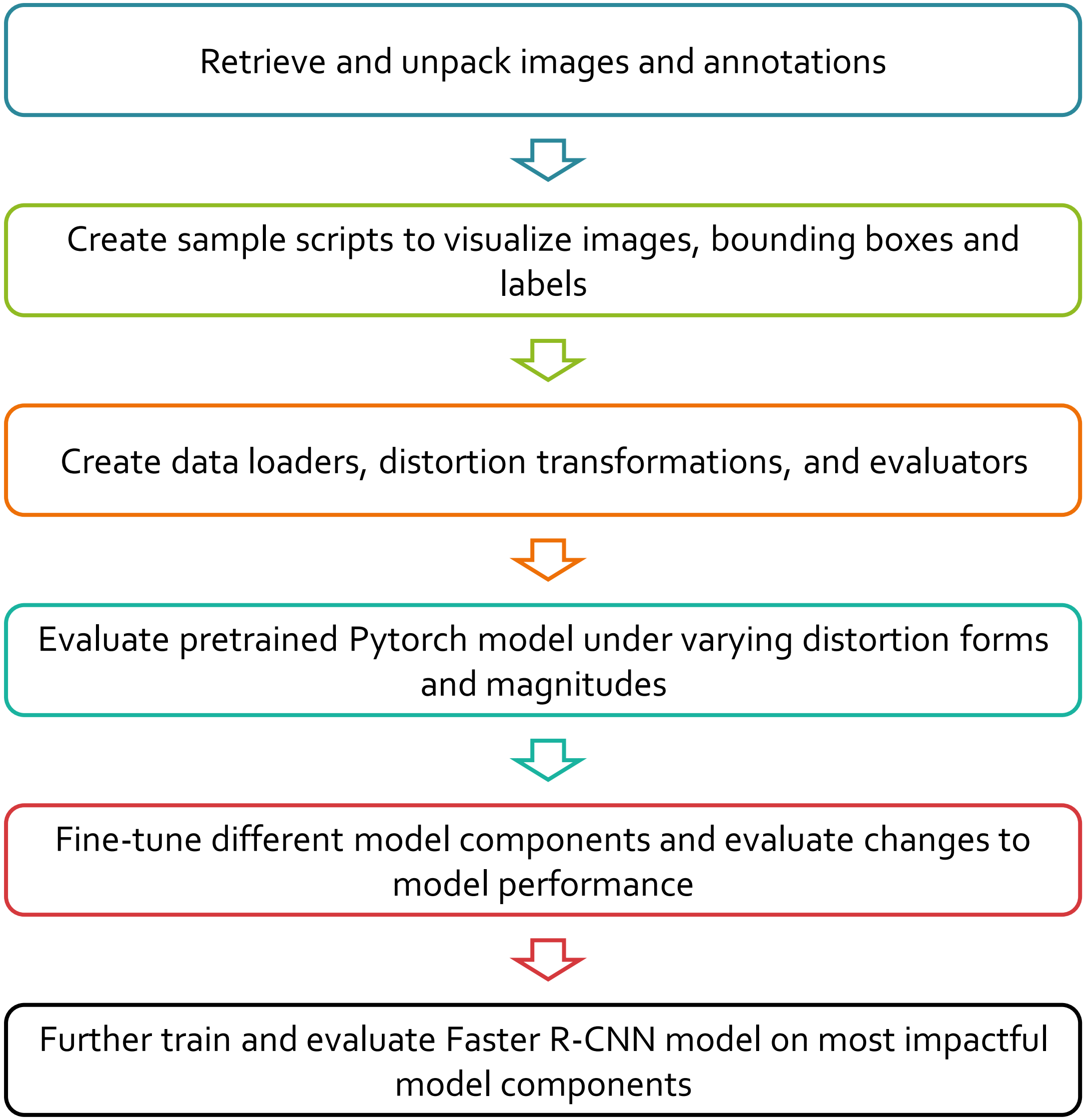
- Lou and Yang showed that deep neural networks are better able to absorb input noise than simpler models, especially if injected earlier in the model

- Dodge and Karam (2016) studied the effect that noise has on various image classification neural network architectures

- Zhou et al. studied the effect that noise had on image classification performance and developed mitigation techniques

Most studies combining computer vision with noise focus on image classification – little work has been done on how noisy input effects object detection models

Overall Project Workflow



Retrieve and unpack images and annotations

Create sample scripts to visualize images, bounding boxes and labels

Create data loaders, distortion transformations, and evaluators

Evaluate pretrained Pytorch model under varying distortion forms and magnitudes

Fine-tune different model components and evaluate changes to model performance

Further train and evaluate Faster R-CNN model on most impactful model components

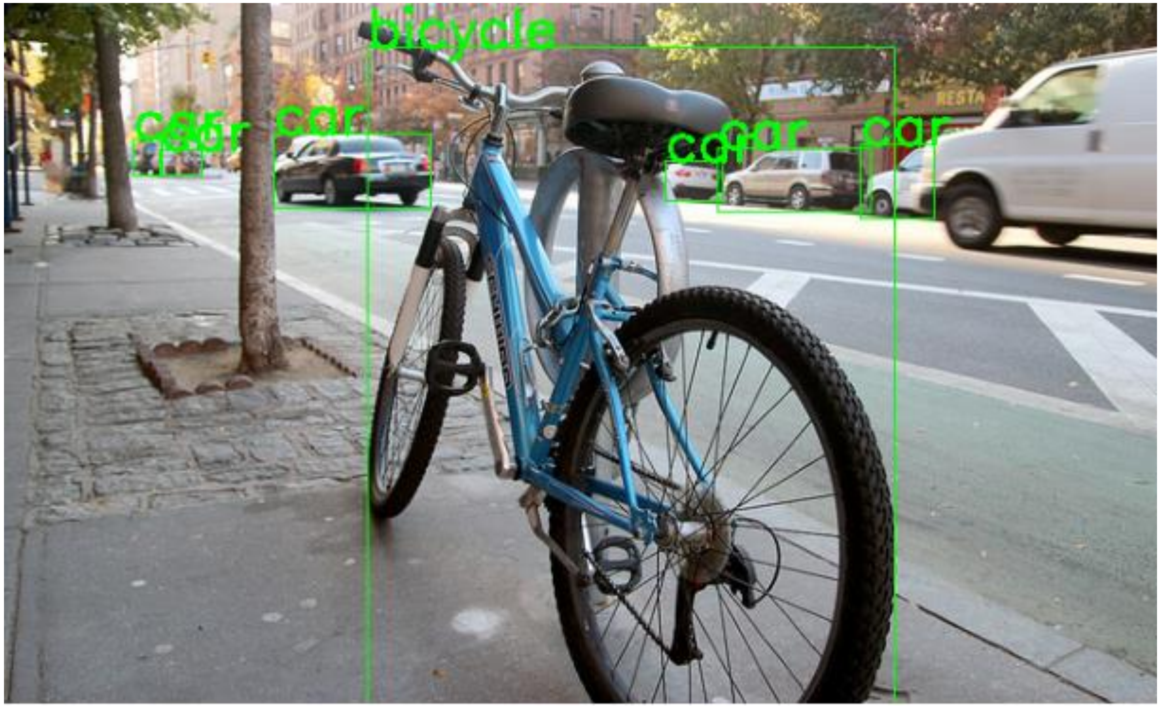
The Data

Microsoft developed the Common Objects in Context (COCO) dataset in 2014 for object detection tasks

The dataset contains over 330,000 images, 80 object classes, and 1.5 million object instances

Classes include labels such as person, bicycle, car, motorcycle, airplane, bus, train, truck, and boat (among others)

For this project, the 2017 dataset was used, which includes 141,000 training images and 5,000 validation images



For each image, the dataset contains a JPG image, and a corresponding annotation file with the following fields:

- Image ID
- File Name
- Width
- Height
- Flickr URL
- Category ID
- Area
- Bounding Box Coordinates

Faster R-CNN

The Faster R-CNN model, developed by Shaoqing Ren et al. (2016), is the current SOTA in the region-based object detection family

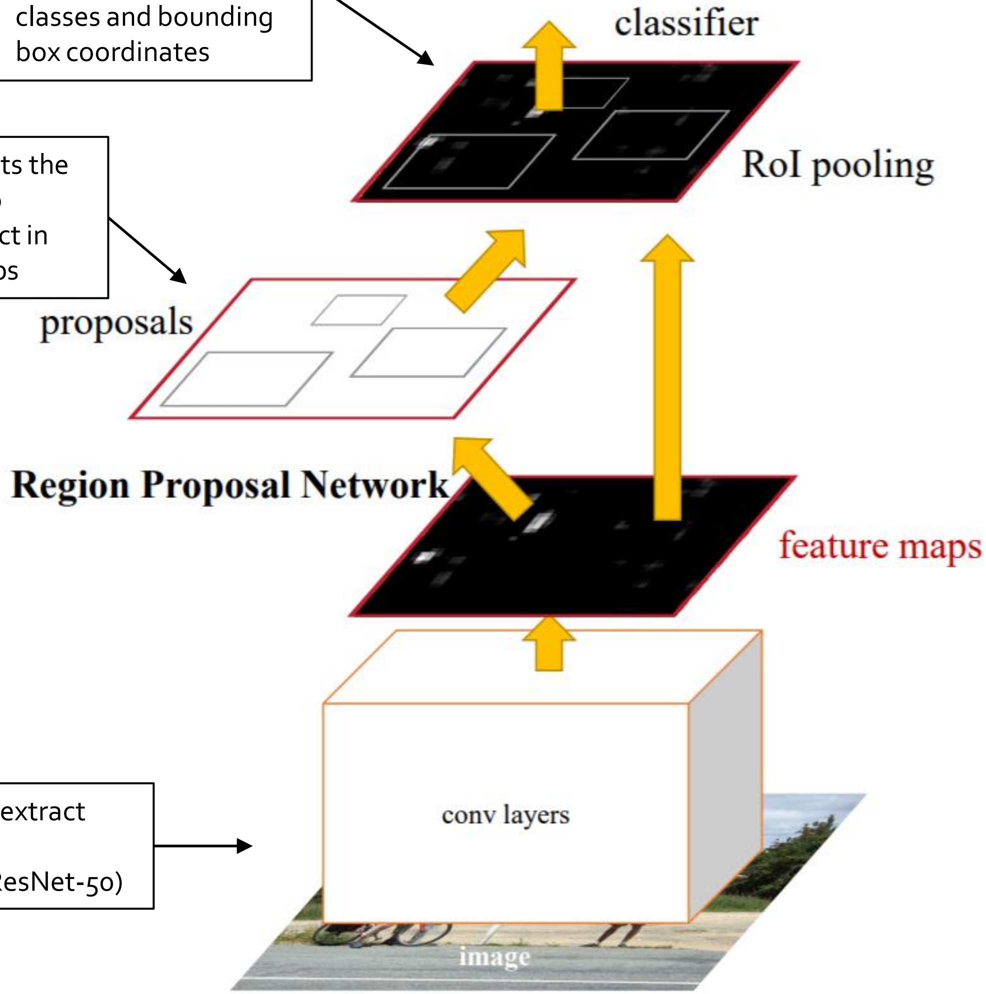
The Faster R-CNN is an extension of the Fast R-CNN model, which was an improvement on the R-CNN model first developed in 2013 by Ross Girshick et al.

The main improvements were the addition of a Region Proposal Network (RPN) to learn the region proposals used when predicting objects, and inverting the order of feature map development

The output layer pools the proposed regions and predicts object classes and bounding box coordinates

The RPN predicts the regions likely to contain an object in the feature maps

A CNN is used to extract feature maps (PyTorch uses a ResNet-50)



Methodology

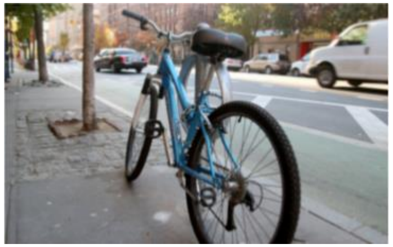
The project is divided into two parts:

- Understand how the object detection model responds to distorted input images
- Use targeted fine-tuning to attempt to mitigate the performance loss caused by distortion

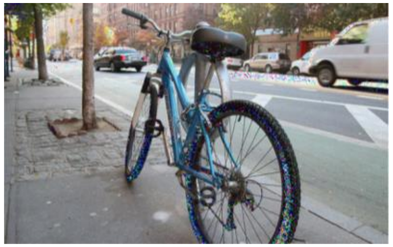
The two forms of distortion being used are noise injection (caused by randomly changing pixel values), and gaussian blur

Varying levels of noise and blur are used to determine how performance decreases by distortion severity

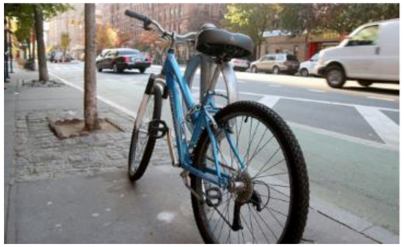
Sample Transformation



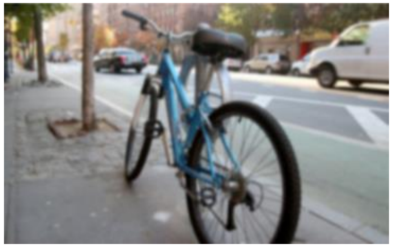
Blur$_{01}$



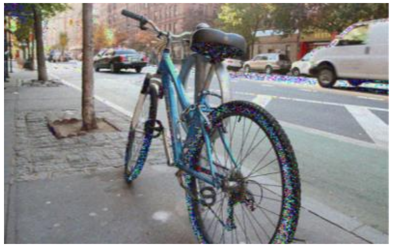
Noise$_{05}$



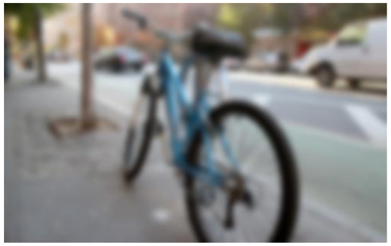
Original Image



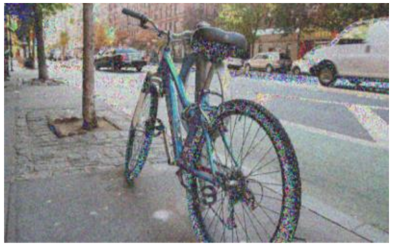
Blur$_{02}$



Noise$_{10}$



Blur$_{05}$



Noise$_{25}$

Noise$_A$: A indicates the standard deviation of the noise mask
Blur$_B$: B indicates the radius of the gaussian blur

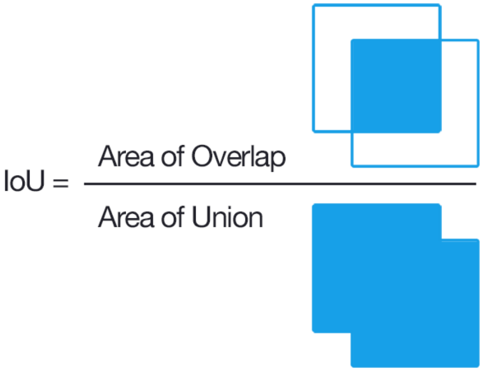
Evaluation Methods

Models will be scored using Mean Average Precision (mAP) implemented by the COCO evaluator class

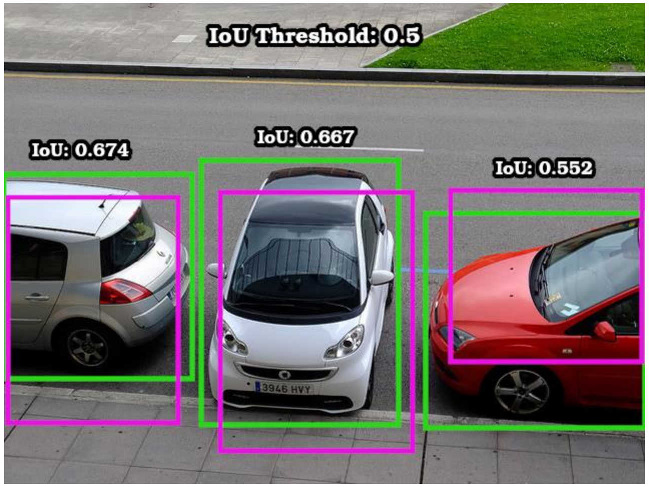
Average precision measures the precision of class predictions across varying levels of bounding box overlap

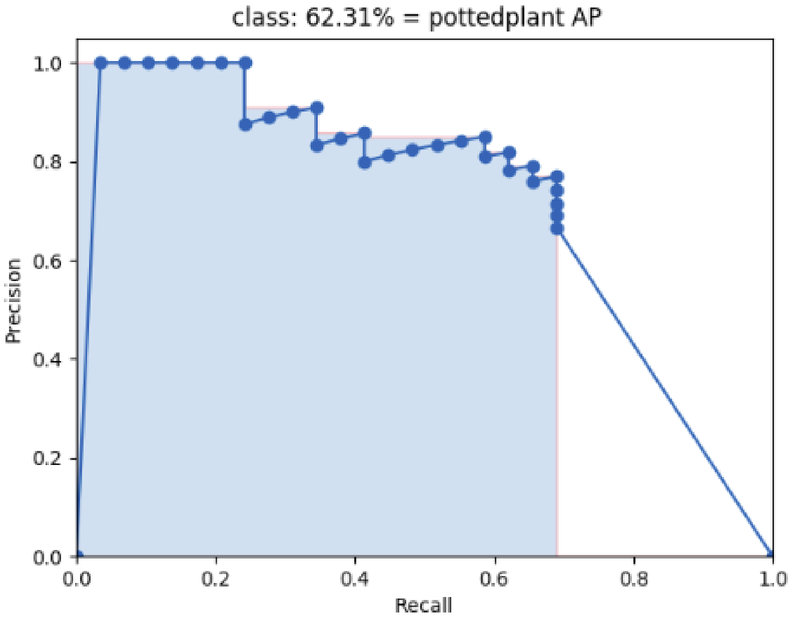
This average precision is then further averaged over all possible predicted classes and IoU thresholds

Results are returned for both the overall object classes, as well as small, medium, and large objects individually



IoU = Area of Overlap / Area of Union





$$AP = \Sigma\,(r_{n+1} - r_n)\,p_{interp}(r_{n+1})$$

$$p_{interp}(r_{n+1}) = \max_{\tilde{r} \geq r_{n+1}} p(\tilde{r})$$

Average precision is interpolated over the precision-recall curve for all object classes and all IoU levels between 0.5 and 0.95 at intervals of 0.05

Baseline Results

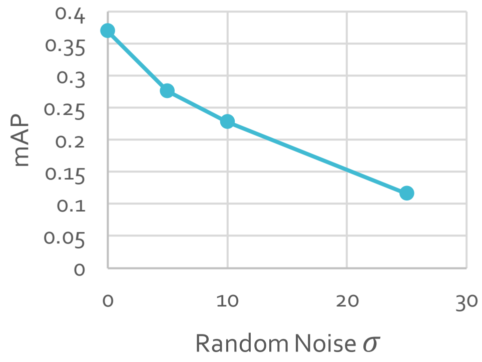
The pretrained PyTorch Faster R-CNN model will be used to set baseline performance values with varied distortion

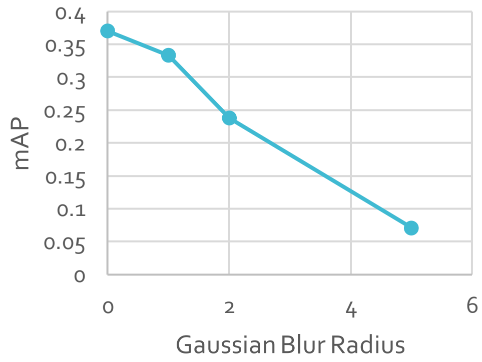
The PyTorch implementation has been pre-trained on the COCO 2017 data, however without the same distorted inputs, just rotations and other minimal adjustments

This output shows the performance of the model when evaluated on non-distorted input (ie. under ideal circumstances)

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

	All Objects	Small (< 32^2px)	Medium (< 96^2px)	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
Blur_{05}	0.071	0.006	0.058	0.161





*Precision is measured over 10 levels of bounding box overlaps, from 0.5 to 0.95 by increments of 0.05

Measure Baseline Errors

Perform Fine Tuning

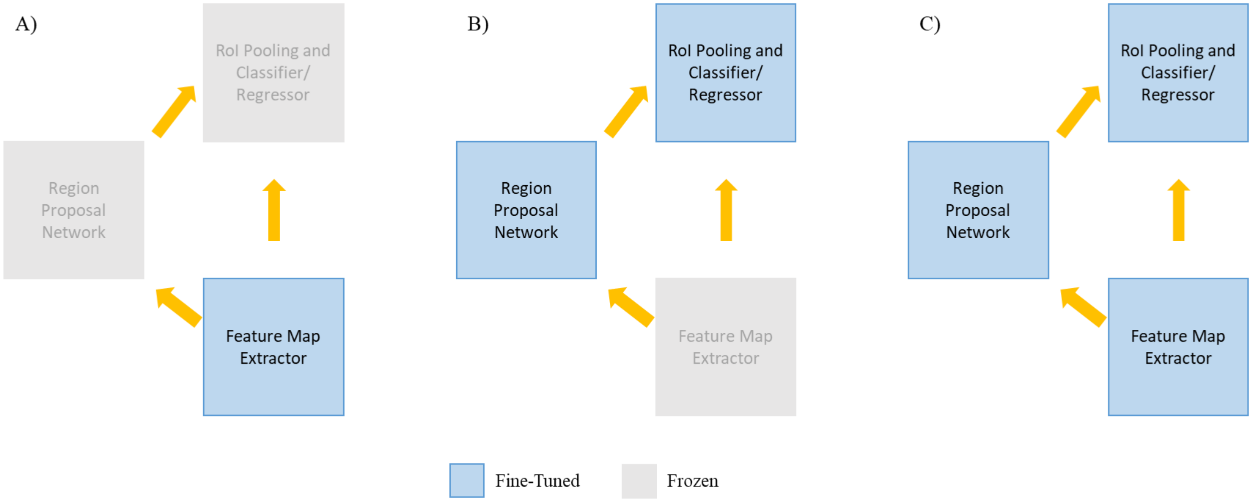
Targeted Fine-Tuning

Model Fine-Tuning Overview

The model is made up of three different components:

- The feature map extractor
- The Region Proposal Network
- The Region of Interest/Pooling and final classification/regression model head

Combinations of these layers will be frozen/unfrozen during model fine-tuning to isolate effects



As the RPN and RoI Pooling and Classifier/Regressor are much smaller than the Feature Map Extractor, they are grouped together for this fine-tuning

Measure Baseline Errors → Perform Fine Tuning → Targeted Fine-Tuning

Model Fine-Tuning

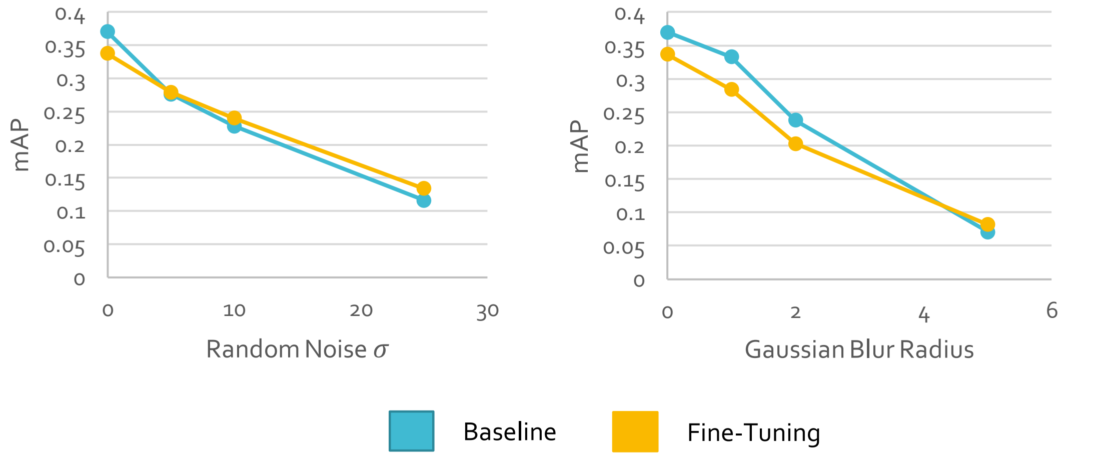
Once baseline performance is established, the next step is to attempt to mitigate the effects of distortion by fine tuning the model to accommodate input distortion

Instead of focusing on the entire model, the RPN and final pooling/classifier layers will be frozen and only the CNN generating the feature maps will be updated

This technique attempts to keep much of the training benefits gained during the initial training

Evaluation of Fine Tuned (Backbone) Faster R-CNN on Distorted Images

	All Objects	Small (< 32^2px)	Medium (< 96^2px)	Large (> 96^2px)
Baseline	0.337	0.188	0.375	0.438
Noise$_{05}$	0.279	0.141	0.309	0.379
Noise$_{10}$	0.240	0.111	0.265	0.341
Noise$_{25}$	0.134	0.045	0.144	0.217
Blur$_{01}$	0.284	0.139	0.316	0.397
Blur$_{02}$	0.203	0.070	0.223	0.315
Blur$_{05}$	0.082	0.011	0.074	0.168



*Precision is measured over 10 levels of bounding box overlaps, from 0.5 to 0.95 by increments of 0.05

Measure Baseline Errors

Perform Fine Tuning

Targeted Fine-Tuning

Model Fine-Tuning (continued)

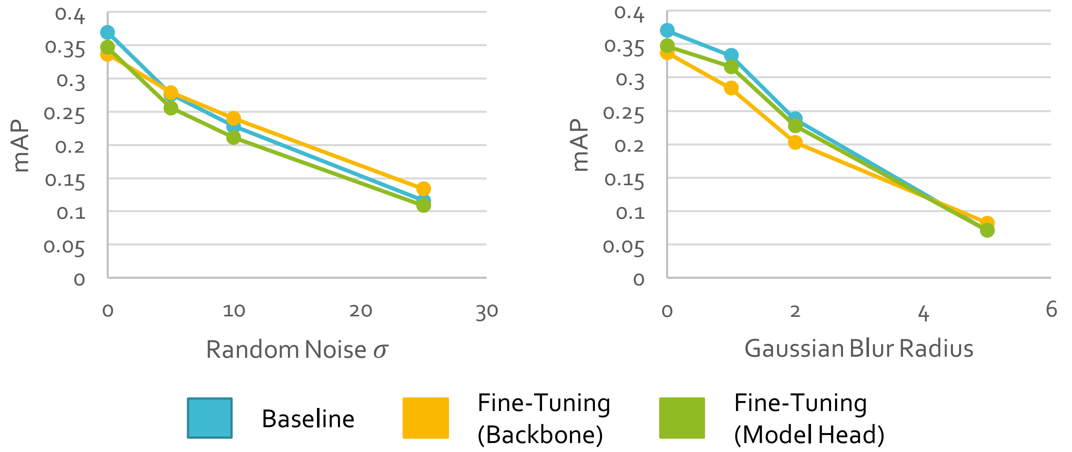
Once baseline performance is established, the next step is to attempt to mitigate the effects of distortion by fine tuning the model to accommodate input distortion

Conversely, instead of fine-tuning the entire model, the RPN and final pooling/ classifier layers will be refined and the CNN generating the feature maps will be frozen

This technique attempts to keep much of the training benefits gained during the initial training, while focusing on a different model component

Evaluation of Fine Tuned (Head) Faster R-CNN on Distorted Images

	All Objects	Small ($< 32^2$px)	Medium ($< 96^2$px)	Large ($> 96^2$px)
Baseline	0.347	0.193	0.388	0.439
Noise$_{05}$	0.256	0.131	0.284	0.342
Noise$_{10}$	0.211	0.098	0.231	0.299
Noise$_{25}$	0.109	0.035	0.118	0.175
Blur$_{01}$	0.316	0.153	0.357	0.423
Blur$_{02}$	0.228	0.079	0.248	0.361
Blur$_{05}$	0.071	0.009	0.059	0.153



*Precision is measured over 10 levels of bounding box overlaps, from 0.5 to 0.95 by increments of 0.05

Measure Baseline Errors → Perform Fine Tuning → Targeted Fine-Tuning

Model Fine-Tuning (continued)

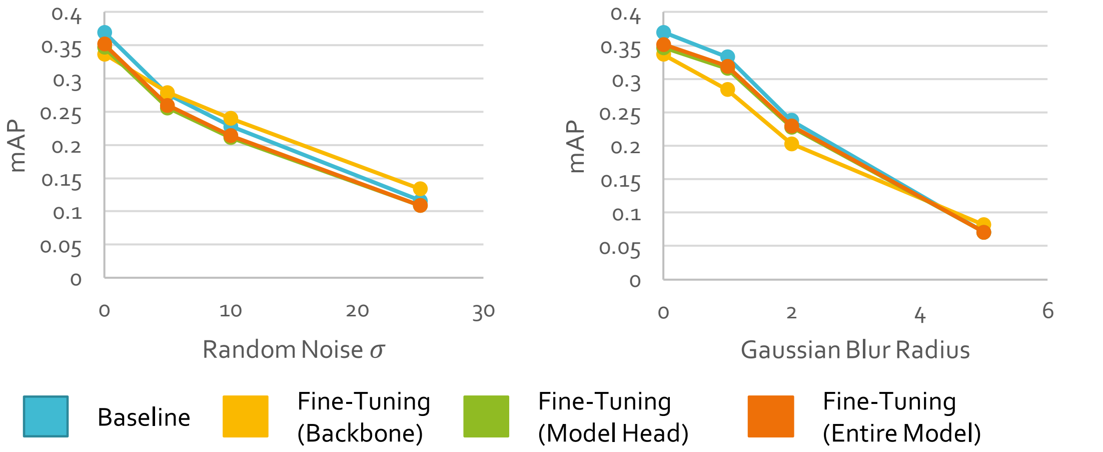
Once baseline performance is established, the next step is to attempt to mitigate the effects of distortion by fine tuning the model to accommodate input distortion

The entire model will be fine tuned on the same number of images, updating weights for all layers throughout the model

This technique attempts to keep much of the training benefits gained during the initial training, and provide the most complete updates to the model weights

Evaluation of Fine Tuned (All) Faster R-CNN on Distorted Images

	All Objects	Small (< 32^2px)	Medium (< 96^2px)	Large (> 96^2px)
Baseline	0.352	0.196	0.391	0.448
Noise_{05}	0.260	0.134	0.285	0.350
Noise_{10}	0.214	0.096	0.234	0.303
Noise_{25}	0.109	0.035	0.117	0.178
Blur_{01}	0.319	0.157	0.358	0.429
Blur_{02}	0.230	0.079	0.248	0.365
Blur_{05}	0.071	0.008	0.059	0.156



*Precision is measured over 10 levels of bounding box overlaps, from 0.5 to 0.95 by increments of 0.05

Measure Baseline Errors → Perform Fine Tuning → Targeted Fine-Tuning

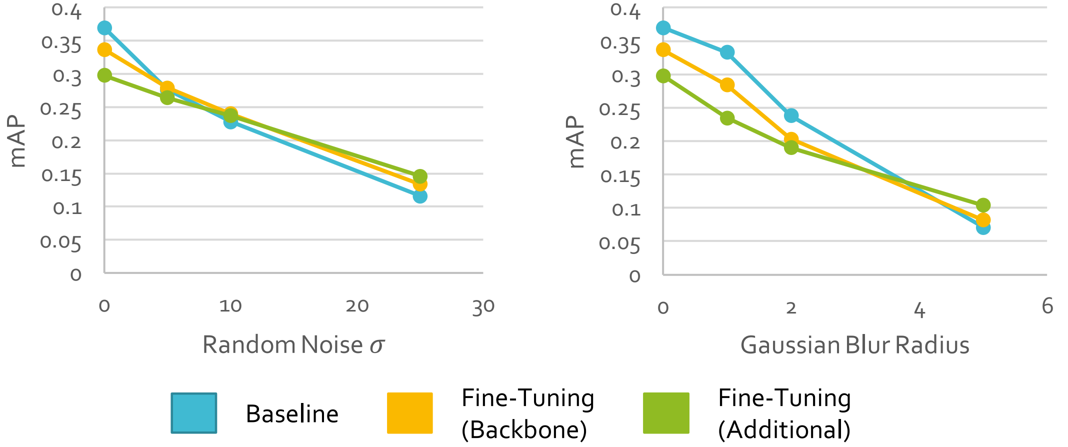
Model Fine-Tuning (continued)

Once it was shown that the feature map extraction layers had the greatest impact on mitigating distortion, additional fine-tuning is focused there

This stage used 10x the training data as the prior fine-tuning efforts to measure the impact additional refinement has on these specific model layers

Evaluation of Fine Tuned (Backbone) Faster R-CNN on Distorted Images (Additional Refinement)

	All Objects	Small ($< 32^2$px)	Medium ($< 96^2$px)	Large ($> 96^2$px)
Baseline	0.298	0.157	0.335	0.392
Noise$_{05}$	0.264	0.127	0.292	0.363
Noise$_{10}$	0.237	0.104	0.261	0.338
Noise$_{25}$	0.146	0.047	0.152	0.237
Blur$_{01}$	0.235	0.104	0.266	0.333
Blur$_{02}$	0.190	0.068	0.212	0.289
Blur$_{05}$	0.104	0.020	0.102	0.189



*Precision is measured over 10 levels of bounding box overlaps, from 0.5 to 0.95 by increments of 0.05

Measure Baseline Errors

Perform Fine Tuning

Targeted Fine-Tuning

Conclusions and Next Steps

- Overall, isolating and fine-tuning solely the feature extraction backbone showed the most success in correctly detecting objects in images with large amounts of distortion
- Using the baseline model (trained on non distorted images) proved to outperform other fine-tuning techniques when presented with images with little to no distortion
- Extended fine-tuning of the feature extraction backbone proves to further improve performance at higher levels of distortion, while further decreasing performance at lower levels
- Future work involves assessing other types of object detection models (e.g. YOLO) as well as substitutions of the model backbone (e.g. VGG)