Freie Universität Berlin

Bachelor Thesis at the Department of Mathematics and Computer Science

Dahlem Center for Machine Learning and Robotics

# Traffic Light Detection with Convolutional Neural Networks and 2D Camera Data

Dennis Hein
Student ID: 4754788
dennis.hein@fu-berlin.de

| | |
|---|---|
| Advisors: | Fritz Ulbrich, |
| | Prof. Dr. Daniel Göhring |
| First examiner: | Prof. Dr. Daniel Göhring |
| Second examiner: | Prof. Dr. Dr. (h.c.) habil. Raúl Rojas |

Berlin, January 20, 2020

## Abstract

Self-driving cars are the next step towards safe and convenient travel, but, as with all machine learning applications, require loads of training data. It would be desirable if the Freie Universität Berlin could use readily available datasets to prototype new machine learning models instead of creating their own dataset from test drives of one of their self-driving cars.

Multiple traffic light detection models were trained on the popular datasets BSTLD and DTLD using the Tensorflow Research repository. The evaluation revealed that predictive power achieved in one dataset generally transfers over to another similar dataset with minimal loss in performance. Neither geographical differences between datasets (e.g. traffic lights at the beginning or the end of an intersection) nor architecture choices seem to impact this result. Some ideas to further reduce performance penalties are given for future work.

## Declaration of Academic Honesty

I hereby declare that this bachelor's thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute. Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references is given in the bibliography.

Berlin, January 20, 2020

Dennis Hein

# Contents

# 1 Introduction

Modern machine learning algorithms require massive amounts of data to learn anything significant. But, given they get sufficient training data, often manage to achieve or even surpass human level performance on that one particular task [1]. Popular examples include AlphaGo [2] [3] and AlphaStar [4] [5].

AlphaGo beat Lee Sedol (18 world titles) convincingly with 4-1 in the Chinese board game Go. AlphaStar beat Liquid'Mana (rank 13 in that years World Championship Series) flawlessly with 5-0 in the real-time strategy game StarCraft 2.

However, obtaining quality training data can be a task in itself. Visual datasets in particular are tedious to create because the annotations for each image need to be created either by hand or automatically by an algorithm.

Both approaches are error prone. Humans can only concentrate for so long, especially on repetitive and monotone tasks [6]. Annotations created by an algorithm can only be as good as that algorithms understanding of the image, which likely is not perfect. A hybrid approach where a human checks and corrects the annotations created by an algorithm is also a possibility, but still not foolproof.

Using publicly available datasets to create and test new machine learning models is therefor desirable. These datasets often allow users to report erroneous annotations, which means that the quality of these datasets should even improve over time.

However, it is not always clear if the performance on one dataset translates to comparable performance on another similar dataset.

In the case of the self-driving cars of the Freie Universität Berlin it would be interesting to see if it is possible to train and test new approaches to traffic light detection on a readily available dataset instead of extracting and labeling the images of the test drives and thereby creating a new dataset from scratch.

A possible challenge are differences in the traffic lights appearance and location in different geographical regions. German traffic lights for example are located at the beginning of an intersection, while traffic lights in the United States of America are located at the end of an intersection.

This thesis will explore how architecture choices and datasets from different geographical regions impact performance on the data of a test drive of one of the self-driving cars from the Freie Universität Berlin. It will attempt to answer if it is possible to prototype traffic light detection approaches for our cars on readily available datasets.

# 2 Fundamentals

## 2.1 2D Camera Data

The term 2D camera data refers to images that do not include any depth information. The two dimensions in the name are width and height. When the general public thinks of digital images they think of 2D camera data.

That does not necessarily mean that the machine representation of that picture is two dimensional, though. Grayscale images can in fact be represented as a single matrix. However, RGB images are represented with a matrix for every color (channel), which adds a third dimension to the representation.

2D camera data therefor fits quite naturally into the machine learning ecosystem, which operates on vectors, matrices and tensors.

## 2.2 Neural Networks

(Artificial) neural network is a blanket term for many different kinds of networks. Conceptually, all types of networks work in similar fashion. Some sort of input, encoded as a vector, matrix, or tensor, is passed through any number of hidden layers. The output is the networks current solution to the task it was given. The quality of this solution depends heavily on how far in the training process the model is [7] [8].

Training is the process of minimizing a loss function. This is done with some form of gradient descent, be it stochastic gradient descent or (mini-)batch gradient descent. The loss function can be any arbitrary function that describes how far the current solution is from the optimal solution [9].

Sometimes the terms (artificial) neural network and feed-forward neural network are used interchangeably. If only some layers of the architecture are a feed-forward neural network those layers are called fully-connected layers [7] [10].

In a feed-forward neural network every neuron of every layer is connected with every neuron of the subsequent layer, as can be seen in Figure 1. Each connection has its own weight, which results in massive amounts of variables that need to be trained.

The trivial example in Figure 1 already has 18 ($4 \cdot 3 + 3 \cdot 2$) variables for the connections and 5 ($3 + 2$) variables for the biases.

Feed-forward neural networks can only work with input vectors. This means that if you want to work with multidimensional data you have to flatten it first. The implications of that and a quick estimation of the amount of variables needed quickly reveal that feed-forward neural networks are a sub-optimal choice when working with 2D camera data.

The example in Figure 1 used 4 input neurons. Even with greyscale images this would mean that the input image has a total of 4 pixels. This is totally unrealistic.

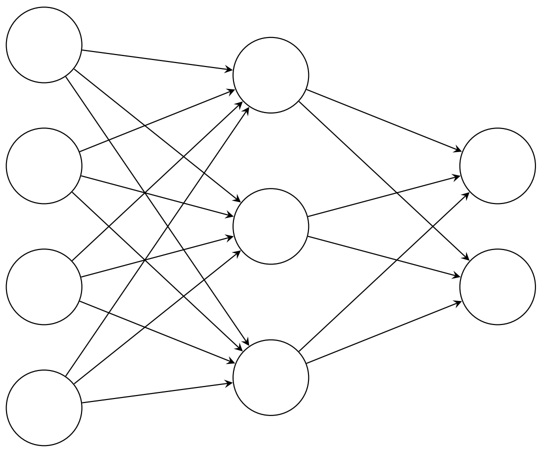


Figure 1: Example of a 3-layer feed forward neural network with 4 input, 3 hidden, and 2 output neurons

The cameras of the self-driving cars from Freie Universität Berlin deliver RGB images with a width of 1920 pixels and a height of 1200 pixels. A feed-forward neural network working with these images would therefor have 6 912 000 ($1920 \cdot 1200 \cdot 3$) input neurons.

The other layers would also need a lot more neurons to be able to learn something from this huge input. At this point, the number of variables increases beyond all reasonable limits.

Flattening the input image implies the loss of spatial information. However, this information is key when trying to understand images. A clump of red pixels might indicate a red light, while some stripes of red pixels with gaps in between could mean anything.

It is possible, although unlikely, that the network will learn how these spaced out features relate to each other. The time needed to train this network would undoubtedly grow substantially.

Another disadvantage of feed-forward neural networks is the fixed input size. This becomes apparent in two scenarios.

As described in the introduction, finding a suitable dataset for training and evaluation can already be a challenge. Adding an additional constraint by requiring the exact resolution of the cameras used by the car narrows the choice down even further. If the camera uses an exotic resolution this might even mean that no such dataset exists.

When the cameras of the car ever get upgraded the resolution of the new cameras might be different. This would mean that the network would need to be retrained from scratch. Alternatively, the camera choice for the upgrade would be constrained to a camera with the same resolution as the old one.

### 2.2.1 Convolutional Neural Networks

Convolutional neural networks are the state-of-the-art when it comes to working with images. It uses many small filters (usually 7x7, 5x5, 3x3 or 1x1), instead of having a single layer of many neurons that are connected to every input neuron. These filters slide over the input image and produce feature maps [10] [11] [12].

These feature maps are modified versions of the image. Visualizing them can help in understanding what that particular filter is looking for in an image.

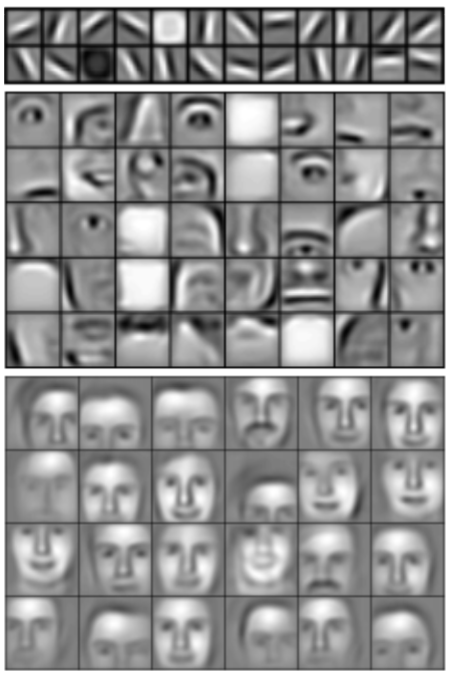


Figure 2: Example activation maps learned from a convolutional deep belief network. Source: [13]

As can be seen in Figure 2, in the early stages of the network the filters look for rudimentary features like orientation of light and dark spots (top). Filters in the later stages work on the activation maps of the earlier stages and start to look for higher level features like eyes and noses (middle). Near the end of the network the filters start to look for even higher level features like the form of a face (bottom).

The powerful thing is that these filters do not need to be defined beforehand. The convolutional neural network learns them based on the training data.

Convolutional neural networks solve all the disadvantages of the feed-forward neural network.

The amount of variables does not grow with the input size. The filters have a fixed size. Assume the network uses 64 3x3 filters in the first layer. That would result in 576 ($3 \cdot 3 \cdot 64$) variables that need to be trained. A significant reduction to the $6912000 \cdot (x + 1)$ variables needed for the first layer of a feed-forward neural network with the same input [10].

The input is not being flattened, which means that the spatial information is preserved. In fact, the input can not be flattened because the filters (apart from 1x1) could not convolve over a $n$x1 vector.

The preservation of the spatial information in the input image is the main reason why convolutional neural networks perform so well on images. Spatial patterns like shadows are preserved and can be generalized in the model.

Convolutional neural networks are also not necessarily fixed in input size. If the architecture includes a fully-connected layer some precautions have to be made. Either the image needs to be preprocessed (resized, cropped,...) or spatial pyramid pooling [14] needs to be used. If the architecture is fully convolutional the only requirement is that the image is larger than the largest filter.

### 2.2.2 Supervised Learning

Supervised learning describes a training methodology used to train artificial neural networks in general. In essence, the datasets consist of input and output pairs, where the output is the expected result. This result is also called the groundtruth. The goal is to minimize the loss function by changing the weights of the network in a way that the output more closely matches the groundtruth [15] [10].

It is assumed that this groundtruth is actually the correct result. That is not necessarily always the case, as talked about in the introduction. Too many wrong groundtruths will be detrimental to the performance of the network.

### 2.2.3 Transfer Learning

Training deep neural networks from scratch costs a lot of time, power and money. Transfer learning remedies that by allowing the use of a pre-trained network [15].

Many uses of artificial neural networks share some commonalities. These commonalities can be leveraged to reuse parts of already trained networks when working on something new.

For example, a network that tries to detect traffic lights can reuse parts of a network that learned to correctly classify images of traffic lights. That new network will not be able to detect traffic lights right away, but it does not have to learn the features that make up a traffic light anymore, which speeds up the training process. The weights of the pre-trained network are frozen (variables turned to constants), in order to prevent the training process from negatively impacting the learned representation.

## 2.3 Object Detection

The aim of object detection is to find all instances of known classes in an image and put bounding boxes around them. For that, two things need to happen. The network needs to be able to correctly identify regions of interest in an image and it needs to correctly classify those regions [16].

There are many approaches to these tasks. The state-of-the-art in image classification is some form of convolutional neural network. The identification of regions of interest does not have a clear answer. There are trade-offs to consider and many approaches are viable. The following will describe the two approaches that were used in this thesis.

### 2.3.1 Faster R-CNN

Faster R-CNN [17] identified the region proposal step as the bottleneck for real-time object detection and proposed a way to combat that.

Previous methods worked on the results of the classification network and were often implemented on the CPU, while classification networks usually run on the GPU. This results in quite a bit of computational overhead when switching between the processors and means that the region proposal often needed the same, if not more, time as the actual classification.

In contrast, Faster R-CNN attaches a region proposal network (RPN) to the classification network. The RPN is itself a convolutional neural network. This allows the region proposal to also run on the GPU and even share parts of the classification computation.

This reportedly results in region proposals being computed in milliseconds rather than seconds.

### 2.3.2 Single Shot MultiBox Detector (SSD)

Single Shot MultiBox Detector (SSD) [18] removes the region proposal step that is often present in other approaches.

Instead, a set of default bounding boxes in many different sizes and aspect ratios is used. The network predicts the presence of an object in those bounding boxes, as well as adjustments to the bounding box to better match the shape of the object.

Removing the region proposal step and combining the entire object detection process into a single network reportedly yields a dramatic speed improvement.

## 2.4 Evaluation Of Classifiers

When evaluating classifiers special care has to be given to the chosen metric. One could naively choose to evaluate the model based on the accuracy, which is the ratio of correctly classified examples to the total number of examples. Problems with that arise quickly, especially when dealing with skewed datasets.

Assume a model was trained to detect cancer. Naturally, there are a lot more negative examples (patients that do not have cancer) than positive ones (patients who do have cancer). This model might achieve an accuracy of well over 90% by always predicting *no cancer*. However, it would not be very useful as it would never fulfill its main purpose (detecting cancer).

### 2.4.1 Confusion Matrix

A confusion matrix is a table that allows the visualization of predictions made by the model. The columns of this table represent the predicted labels, while the rows represent the groundtruth labels. The cells contain counts on how often the given class was classified correctly or incorrectly and what incorrect class was predicted [19].

Sometimes the columns and rows are swapped. The result is the same, but the matrix would be transposed and would need to be read column by column instead of row by row.

Table 1: An example confusion matrix taken and slightly modified from [20]

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Dog** | **Cat** | **Plane** | **Tree** |
| Groundtruth | **Dog** | 70 | 10 | 15 | 5 |
| | **Cat** | 8 | 67 | 20 | 5 |
| | **Plane** | 0 | 11 | 88 | 1 |
| | **Tree** | 4 | 10 | 14 | 72 |

The perfect classifier would have zeros everywhere but on the diagonal. This is of course highly unlikely and would raise the question if the validation dataset was part of the training dataset.

If the validation dataset is very big the confusion matrix can become quite illegible. If that is the case it can be beneficial to convert the entries into percentages. This allows readers to get an idea of the classifiers performance at first glance regardless of the datasets size.

### 2.4.2 Precision

Precision denotes the probability that the result was correct. It is calculated by dividing the true positives (TP) with the sum of true positives and false positives (FP) [19].

$$Precision = \frac{TP}{TP + FP}$$

It is possible to calculate the precision from a confusion matrix. For that the number on the diagonal corresponding to the class is divided by the sum of that column.

In the example given in Table 1, the precision for the class *Dog* would be $\frac{70}{70+8+0+4} \approx 0.85$.

This can be done for every class in the confusion matrix. The result would be four precision values. The overall precision of this classifier would be the so-called macro precision, which is the arithmetic mean of all the precision values [21].

The overall precision of the classifier in Table 1 would therefor be $\frac{0.85+0.68+0.64+0.87}{4} \approx 0.76$.

### 2.4.3 Recall

Recall denotes how many of the classes examples were correctly classified. It is calculated by dividing the true positives with the sum of true positives and false negatives (FN) [19].

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

It is possible to calculate the recall from a confusion matrix. For that the number on the diagonal corresponding to the class is divided by the sum of that row.

In the example given in Table 1, the recall for the class *Dog* would be $\frac{70}{70+10+15+5} = 0.7$.

This can be done for every class in the confusion matrix. The result would be four recall values. The overall recall of this classifier would be the so-called macro recall, which is the arithmetic mean of all the recall values [21].

The overall recall of the classifier in Table 1 would therefor be $\frac{0.7+0.67+0.88+0.72}{4} \approx 0.74$.

### 2.4.4 Precision vs. Recall

Precision and recall are related to each other. Improving one usually worsens the other. This can be easily seen when looking at the extremes.

A perfect recall can be trivially achieved by always predicting the class, no matter the input. This will result in every instance of the class being classified correctly (perfect recall), but every other class being classified incorrectly (abysmal precision).

The reverse is also true. Picking a very high confidence threshold will result in very few, but correct classifications (perfect precision). However, the recall will be low because the classifier will miss a lot of instances from that class where confidence was not high.

A trade-off has to be made, since not both precision and recall can be very high. So what is more important? That really depends on the type of application being built.

Going back to the cancer example, a high precision would mean that patients that are predicted to have cancer are very likely to actually have it. Fewer patients would get the shock of the diagnosis and undergo the painful and costly treatment. But, the lower recall would also mean that many more patients with cancer will not receive the treatment they need.

On the other hand, a high recall, accompanied by low precision, would mean that many patients who do not have cancer get the shocking diagnosis and might even undergo a treatment that they do not need.

# 3 Training

## 3.1 Datasets

### 3.1.1 Bosch Small Traffic Light Dataset

The Bosch Small Traffic Light Dataset (BSTLD) [22] contains about 13 400 images and about 24 000 annotated traffic lights from Palo Alto, California.

The annotations are composed of the coordinates for the bounding box, the current state of the traffic light, and whether or not the traffic light is (partially) occluded.

The approximately 5 100 images from the training set cover various challenges that one can expect to face when driving in urban environments. These challenges include road-works, dense stop-and-go traffic, multiple visible traffic lights, suburban multilane roads with varying traffic density, and different weather conditions.

The test set was recorded in a single trip and on a route that was not traveled on in the training set, but is from the same general area. This would be suboptimal for a production system that learns exclusively from this dataset, but should be fine for the use-case of this thesis.

The test set only includes four possible states (off, green, yellow, red), while the training set has a total of 15 possible states. The additional states are more specific versions of the four base states (e.g. RedLeft). For the purposes of this thesis the specification is not necessary and was ignored (e.g. RedLeft simply interpreted as red).

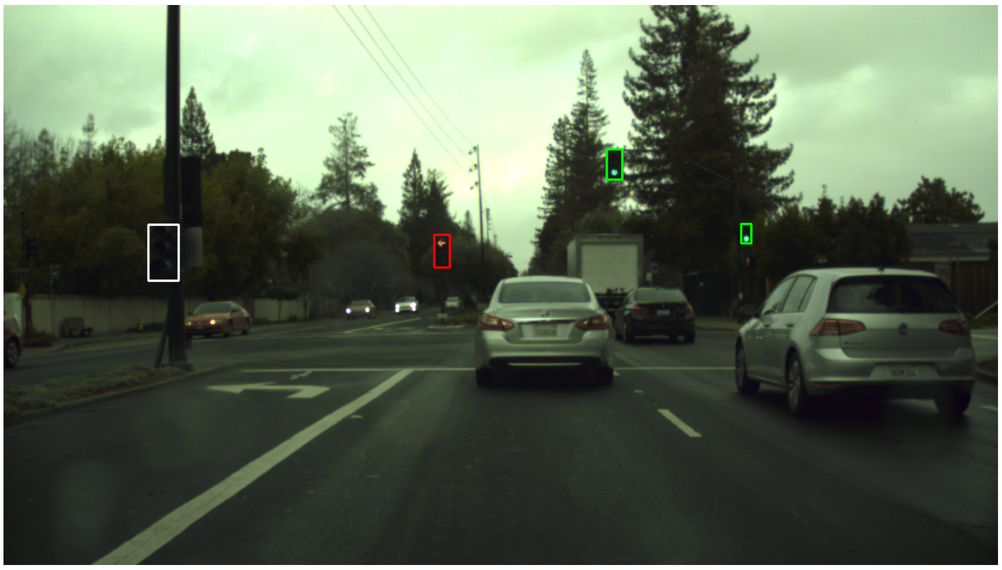


Figure 3: Example image from the training set of the BSTLD dataset with labels

Table 2: Details about the BSTLD dataset split by training and test set

| | **Training set** | **Test set** |
|---|---|---|
| **# of images** | 5093 | 8334 |
| **# of traffic lights** | 10756 | 13486 |
| **# off** | 726 | 442 |
| **# green** | 5422 | 7569 |
| **# yellow** | 444 | 154 |
| **# red** | 4164 | 5321 |
| **Avg. width** | 11.2 | 9.4 |
| **Avg. height** | 24.2 | 26.8 |
| **(Partially) Occluded** | 170 | 2088 |

### 3.1.2 DriveU Traffic Light Dataset

The DriveU Traffic Light Dataset (DTLD) [23] contains about 41 000 images and about 232 000 annotated traffic lights from 11 German cities.

The annotations are composed of the coordinates for the bounding box and a 6-digit class id. Every digit represents a category with multiple possible values. The categories include state and occlusion, but also categories like the pictogram (circular, arrow, pedestrian,...) and viewpoint orientation, which describes what part of the traffic light (front, back, left, right) is visible.

Additionally, the dataset only includes images were at least one traffic light changed compared to the last frame. This means that waiting at a red traffic light only includes a single frame of waiting.

The variety in cities and the sheer amount of scenes present in this dataset ensure that typical challenges that arise when driving in German cities are represented. Different degrees of traffic density are also present due to the nature of driving in smaller cities like Kassel and bigger cities like Berlin.

Each of the cities had unique routes and every intersection of those routes was saved separately. This enables the test and training set to contain intersections from every city, while ensuring that no intersection of the training set is part of the test set and vice versa.

For a production system a real split between cities would probably be better, but again should be fine for the use-case of this thesis.

For this thesis the annotations were translated to be in the same format as the annotations provided by the BSTLD dataset.

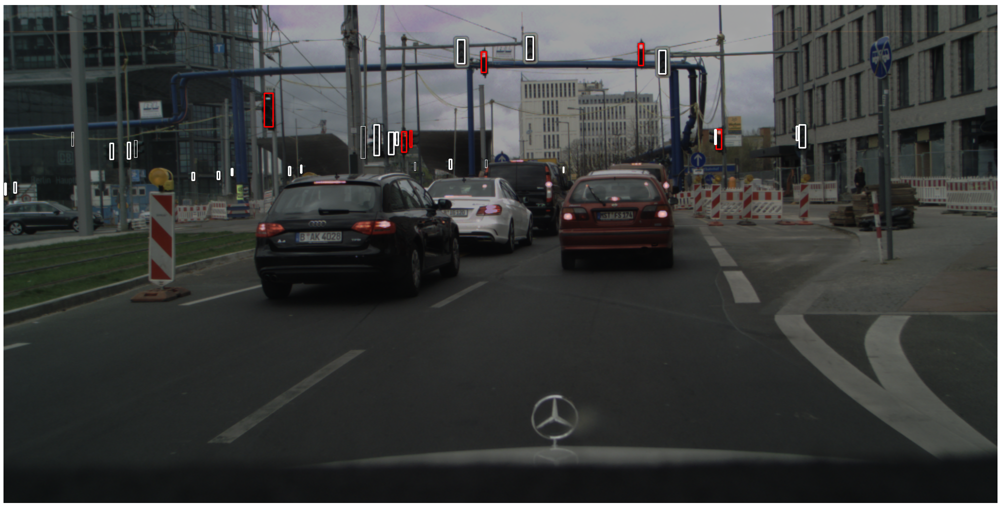


Figure 4: Example image from the training set of the DTLD dataset with labels

Table 3: Details about the DTLD dataset split by training and test set

| | Training set | Test set |
|---|---|---|
| **# of images** | 28526 | 12453 |
| **# of traffic lights** | 159902 | 72137 |
| **# off** | 35801 | 18696 |
| **# green** | 67534 | 29618 |
| **# yellow** | 7636 | 3033 |
| **# red** | 48931 | 20790 |
| **Avg. width** | 11.2 | 11.2 |
| **Avg. height** | 35.2 | 34.8 |
| **(Partially) Occluded** | 23189 | 10663 |

## 3.2 Architectures

Training was done using the open source deep learning framework Tensorflow. Tensorflow provides a research repository with scripts for various tasks, including object detection. These scripts implement a convenient way to take advantage of transfer learning and evaluating the resulting models.

A handful of architectures are supported and snapshots of those architectures, trained on popular datasets, can be downloaded from the Tensorflow website.

The snapshots used for this thesis were trained on the COCO dataset [24], which includes traffic lights as one of their 91 categories.

Object detection itself is the combination of two tasks. Objects within the image need to be classified and located.

Additionally, in the context of self-driving cars, the object detection needs to take place in a timely fashion. Ideally, a self-driving car would be able to detect objects of interest (e.g. traffic lights) in real-time. This is especially challenging because of the limited computational power.

A trade-off between speed and accuracy is therefore necessary.

The architecture choice for this thesis attempts to test that trade-off. SSD Mobilenet v1 leans more towards speed, while Faster RCNN Inception v2 leans more towards accuracy.

### 3.2.1 SSD Mobilenet v1

Localization of objects is done with a Single Shot MultiBox Detector (SSD).

Classification of objects is done with the first version of Mobilenet [25]. Or, more accurately, feature extraction is done with Mobilenet, since SSD replaces the classification layers of its base network.

Mobilenet was invented with mobile and embedded applications in mind. The goal was to create a network architecture that is fast, but still has competitive accuracy.

This is achieved by using a special form of convolution, which reduces the amount of parameters in the model drastically and spends most (95%) of the computation in 1x1 convolutions that can be implemented with a highly optimized algorithm.

### 3.2.2 Faster RCNN Inception v2

Localization of objects is done with Faster R-CNN.

Classification of objects is done with the second version of the convolutional neural network codenamed Inception [26].

Given enough training data, deeper neural networks usually perform better than less deep ones. This comes at the cost of more parameters and higher computational cost.

The second version of Inception explored ways to reduce the computational cost and amount of parameters of its layers without compromising the accuracy of the network. This allows for even deeper networks with minimal increase in computational cost.

## 3.3 Process

Both architectures were trained on each dataset for 200 000 global steps and with multiple batch sizes (1 and 4). Global steps is Tensorflow terminology for number of batches seen (respectively number of weight updates) during training. The number of epochs is therefor the product of batch size and global steps divided by the number of images in the training set.

$$Epochs = \frac{\text{batch size} \cdot \text{global step}}{\text{\# images in training set}}$$

The DTLD dataset already only includes images with traffic lights in them. The BSTLD dataset however has empty frames, meaning there are images without traffic lights. The provided scripts for BSTLD filter the dataset (remove images without traffic lights) when creating the TFRecord files. The models for this thesis were trained on both filtered and unfiltered versions of the BSTLD dataset. This results in four models based on the DTLD dataset and eight models based on the BSTLD dataset.

# 4 Evaluation

All trained models were evaluated on the test sets of both BSTLD and DTLD, as well as one of the test drives of a self-driving car from the Freie Universität Berlin. The following sections will only show the interesting results. The complete results can be found in the appendix.

## 4.1 Bosch Small Traffic Light Dataset

The performance on the test set from BSTLD varied widely. The best performance with a weighted mAP of 0.5684 was achieved by the SSD Mobilenet trained on the filtered BSTLD dataset with a batch size of 4 {1}. The worst performance with a weighted mAP of 0.1996 was achieved by the same architecture and training data with a batch size of 1 {2}.

SSD Mobilenet trained on the DTLD dataset with batch size 4 was only marginally worse than its BSTLD counterpart with a weighted mAP of 0.5517 {3}. This might indicate that regional differences in the appearance and position of traffic lights do not significantly impact the performance of a model.

Table 4: Best and worst performing models for BSTLD

| **Model** | **Weighted mAP** | **mAP** | **Off** | **Green** | **Yellow** | **Red** |
|---|---|---|---|---|---|---|
| {1} | 0.5684 | 0.3674 | 0.0001 | 0.666 | 0.3383 | 0.4653 |
| {2} | 0.1996 | 0.1054 | 0.0 | 0.3096 | 0.0001 | 0.1118 |
| {3} | 0.5517 | 0.3594 | 0.0 | 0.6519 | 0.3121 | 0.4736 |

The individual AP scores (Table 4) do not reveal any major differences between the best performing models. This further supports the theory that the generalizations learned by the models are independent of regional differences.

When comparing the P/R curves for {1} and {3} (Figure 5 and Figure 6), one can see that the curves for red and green look very similar. Especially the curves for green are on a good way to the optimum. This is also reflected in Table 4.

However, the curves for yellow look different. {3} seems to have perfect precision for very high confidence thresholds. Once the threshold decreases it quickly adds false positives though, which destroys its precision and usefulness. {1} on the other hand adds false positives at a more steady pace, but the precision is never outstanding. Overall, {1} is probably nicer to work with as its performance deteriorates more slowly and it therefor has a relatively useful output for longer.

The curves for {2} (Figure 7) look just as depressing as one would expect from its AP values (Table 4).

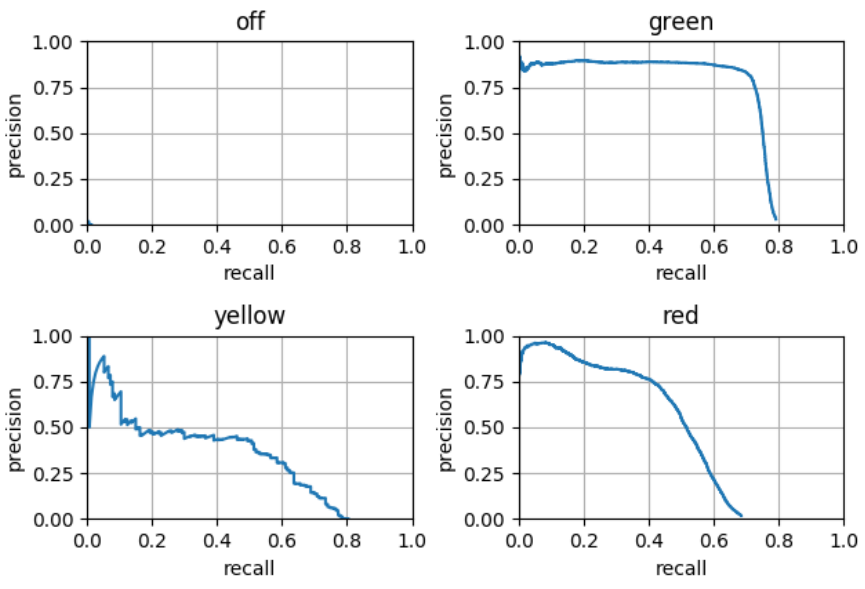


Figure 5: P/R curves for SSD Mobilenet trained on BSTLD (filtered) with batch size 4

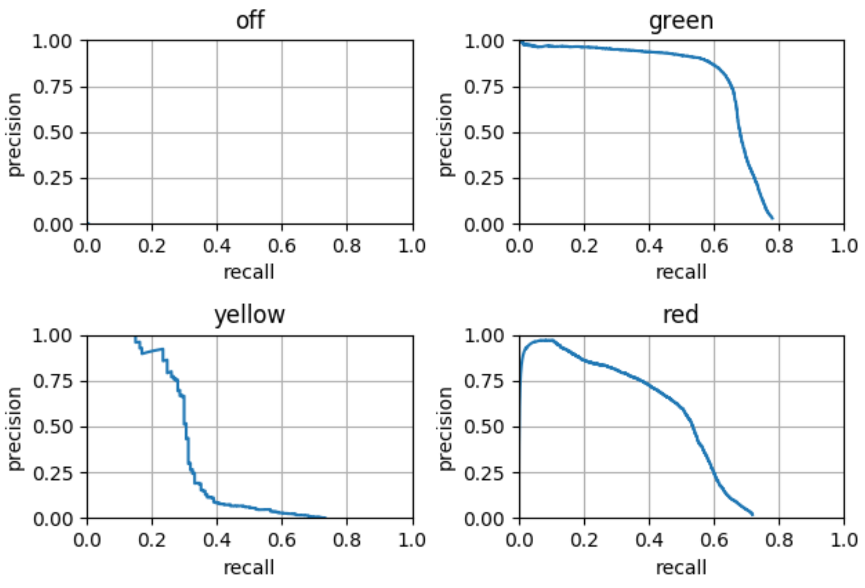


Figure 6: P/R curves for SSD Mobilenet trained on DTLD with batch size 4

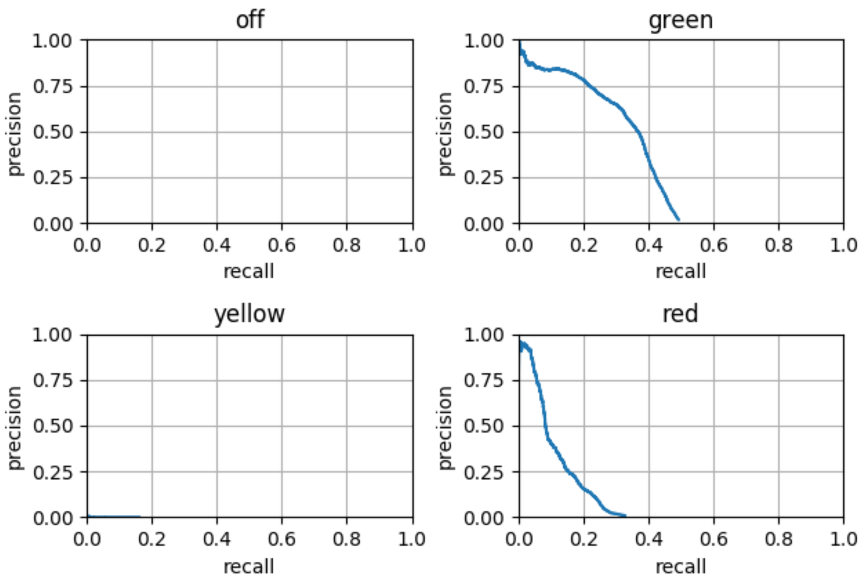


Figure 7: P/R curves for SSD Mobilenet trained on BSTLD (filtered) with batch size 1

## 4.2 DriveU Traffic Light Dataset

The models in general performed worse on this dataset than on BSTLD. The best result came from the Faster RCNN Inception architecture trained on DTLD with a batch size of 4 (weighted mAP of 0.4397) {4}. The second best, with a weighted mAP of 0.4054, is the same architecture, but trained on BSTLD (all) with a batch size of 1 {5}.

Although the margin is bigger this time, the performance is still relatively close between the two best performing models. This is further indication that datasets from different geographical regions do not impact model performance in a significant way.

The SSD Mobilenet architecture seems to struggle on DTLD data. While the best model achieves a somewhat respectable weighted mAP of 0.3496 the remaining five models achieve between 0.1266 and 0.0136. It is not immediately obvious why that would be the case. One assumption could be that the traffic lights in DTLD are smaller, making it harder to detect them, but in fact the opposite is the case. Another possibility is that the DTLD images are too busy, as they contain on average $\approx 3.6$ times as many traffic lights per image as their BSTLD counterparts.

Table 5: Best performing models for DTLD

| Model | Weighted mAP | mAP | Off | Green | Yellow | Red |
|---|---|---|---|---|---|---|
| {4} | 0.4397 | 0.3947 | 0.0281 | 0.627 | 0.408 | 0.5158 |
| {5} | 0.4054 | 0.3696 | 0.0271 | 0.5964 | 0.383 | 0.472 |

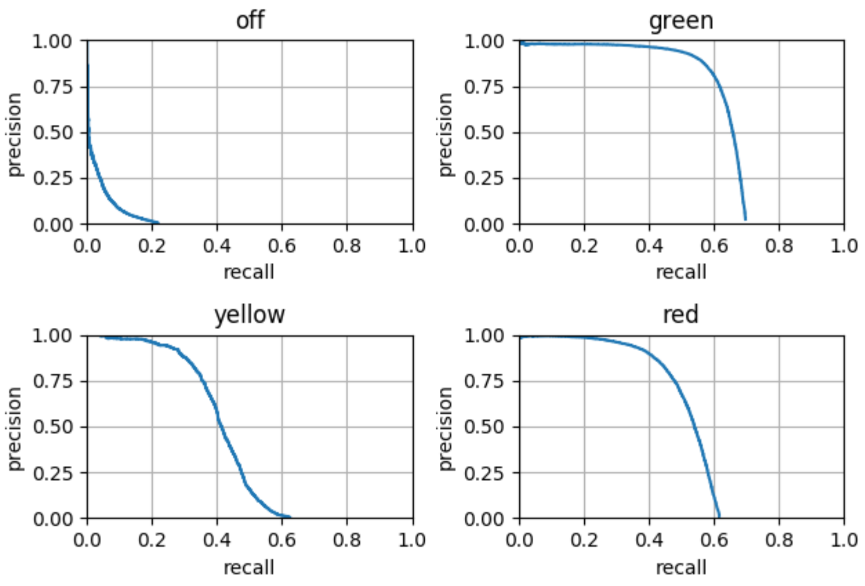


Figure 8: P/R curves for Faster RCNN Inception trained on DTLD with batch size 4

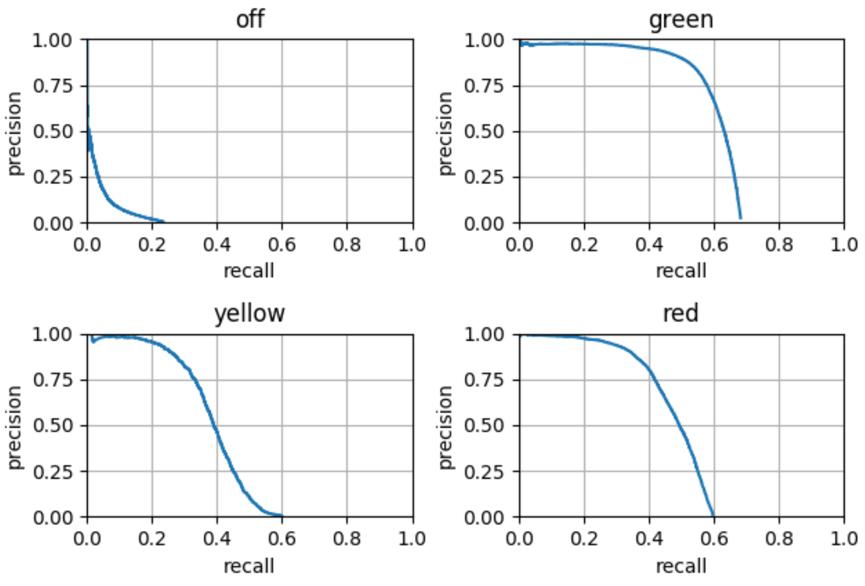


Figure 9: P/R curves for Faster RCNN Inception trained on BSTLD (all) with batch size 1

Looking at the P/R curves (Figure 8 and Figure 9) one could easily think that the two models are identical. But, after some careful inspection subtle differences can be made out. When looking at the corresponding AP values (Table 5) the difference is a lot clearer.

Both models seem to be good candidates for further training. The P/R curves look promising and especially the top performing model has had barely any training ($\frac{4 \cdot 200000}{159902} \approx 5$ epochs).

## 4.3 Freie Universität Berlin Test Drive Data

The test drive data had to be labeled in order to be able to evaluate the performance of the models objectively. The test drive contains 4 201 images and 4 226 traffic lights, which were labeled by hand. The test drive was recorded on the Thielallee in front of the Rost- and Silberlaube.

This rather short drive does not include turned off traffic lights and very few yellow traffic lights. The average width and height of the traffic lights is about twice as big as the ones in DTLD and BSTLD.

Table 6: Details about the labeled test drive data from the Freie Universität Berlin

| | |
|---|---:|
| **# of images** | 4201 |
| **# of traffic lights** | 4226 |
| | |
| **# off** | 0 |
| **# green** | 2211 |
| **# yellow** | 174 |
| **# red** | 1841 |
| | |
| **Avg. width** | 20.4 |
| **Avg. height** | 45.4 |
| | |
| **(Partially) Occluded** | 0 |

Running the evaluation revealed very disappointing results (Table 7). Even the models that had decent performance on BSTLD and DTLD perform abysmal on the test drive data. Bad performance of models trained on BSTLD might have been to geographical differences (traffic lights in the United States of America look different and are positioned differently), but DTLD even includes traffic lights from Berlin, so the performance drop should intuitively not be that big.

Taking a look at images of the test drive (Figure 10) reveals a major difference in the images of BSTLD and DTLD. Both BSTLD and DTLD had their cameras perfectly centered and leveled in the car. The cameras of the self-driving car from Freie Universität Berlin are pointed more towards the sky and rotated by a couple degrees, which makes the traffic lights appear diagonal in the image.

Table 7: Previously best / worst performing models for BSTLD and DTLD

| Model | Weighted mAP | mAP | Off | Green | Yellow | Red |
|---|---|---|---|---|---|---|
| {1} | 0.1001 | 0.0656 | 0.0 | 0.127 | 0.0 | 0.0698 |
| {2} | 0.1066 | 0.0911 | 0.0 | 0.0868 | 0.0039 | 0.1827 |
| {3} | 0.0306 | 0.106 | 0.0 | 0.0258 | 0.2352 | 0.0571 |
| {4} | 0.0059 | 0.0046 | 0.0 | 0.0139 | 0.0 | 0.0 |
| {5} | 0.0082 | 0.0055 | 0.0 | 0.0166 | 0.0 | 0.0 |

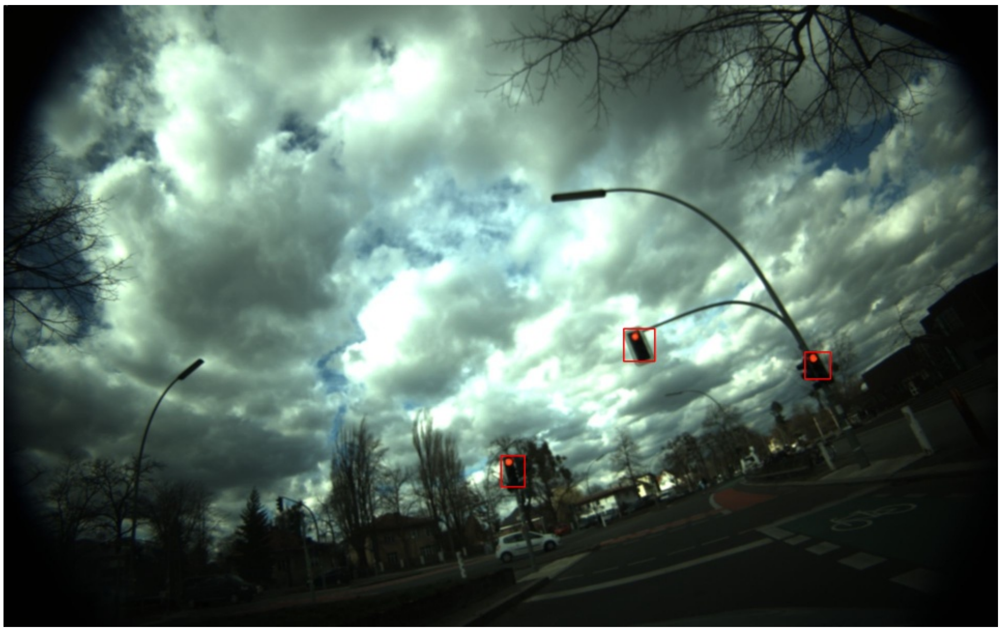


Figure 10: Example image from the test drive with labels

While the focus towards the sky could not be changed, the rotation can be counteracted by pre-processing the images before feeding them to the model. A rotation by 20 degrees was chosen by experimenting with the values until the traffic lights appeared visually to be level (Figure 11).

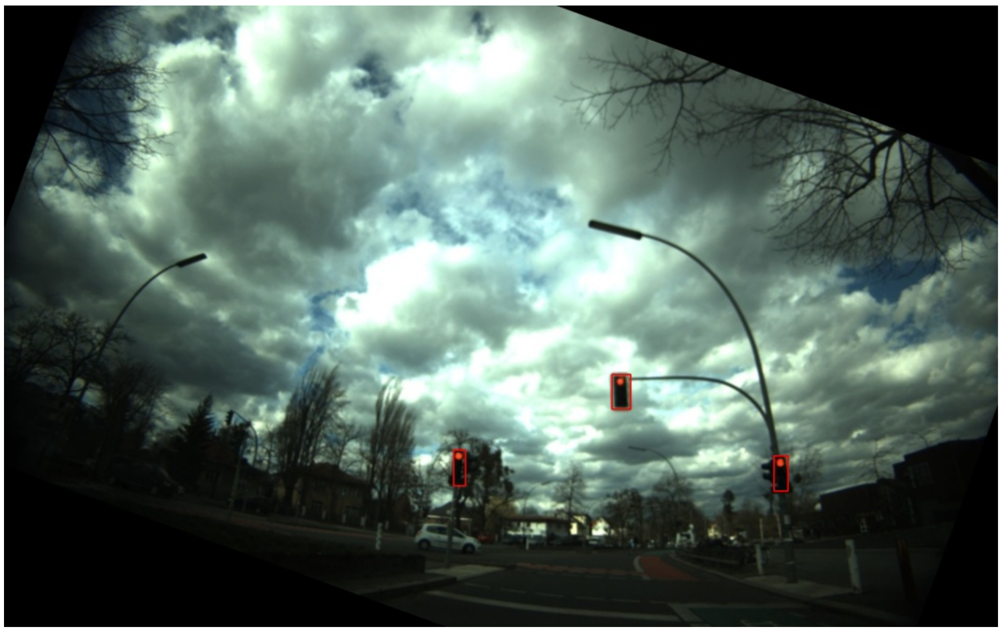


Figure 11: Example rotated image from the test drive with labels

Running the evaluation again, but this time with the rotated images yielded significant improvements in model performance (Table 8). It appears that the models can not handle traffic lights in unusual orientations (yet). This could lead to very dangerous situations should the self-driving car encounter damaged traffic lights (Figure 12).

Table 8: Previously best / worst performing models for BSTLD and DTLD on rotated FU data

| **Model** | **Weighted mAP** | **mAP** | **Off** | **Green** | **Yellow** | **Red** |
|---|---|---|---|---|---|---|
| {1} | 0.4727 | 0.3537 | 0.0 | 0.3274 | 0.0001 | 0.7335 |
| {2} | 0.4091 | 0.3545 | 0.0 | 0.27 | 0.1901 | 0.6035 |
| {3} | 0.3721 | 0.5298 | 0.0 | 0.187 | 0.8047 | 0.5977 |
| {4} | 0.0125 | 0.0107 | 0.0 | 0.0279 | 0.0029 | 0.0014 |
| {5} | 0.0036 | 0.0047 | 0.0 | 0.0058 | 0.0046 | 0.0036 |

The biggest surprise is the performance improvement from model {2}. After consistently being the worst performing model in both BSTLD and DTLD it jumps to rank 4 of the 12 models on the Freie Universität test drive data. The 4.7x improvement from model {1} is also quite impressive.

When comparing the P/R curves for model {1} on the BSTLD data and Freie Universität data (Figure 5 and Figure 13) a major difference is visible. While model {1} could detect yellow traffic lights reasonably well on the BSTLD data it completely misses

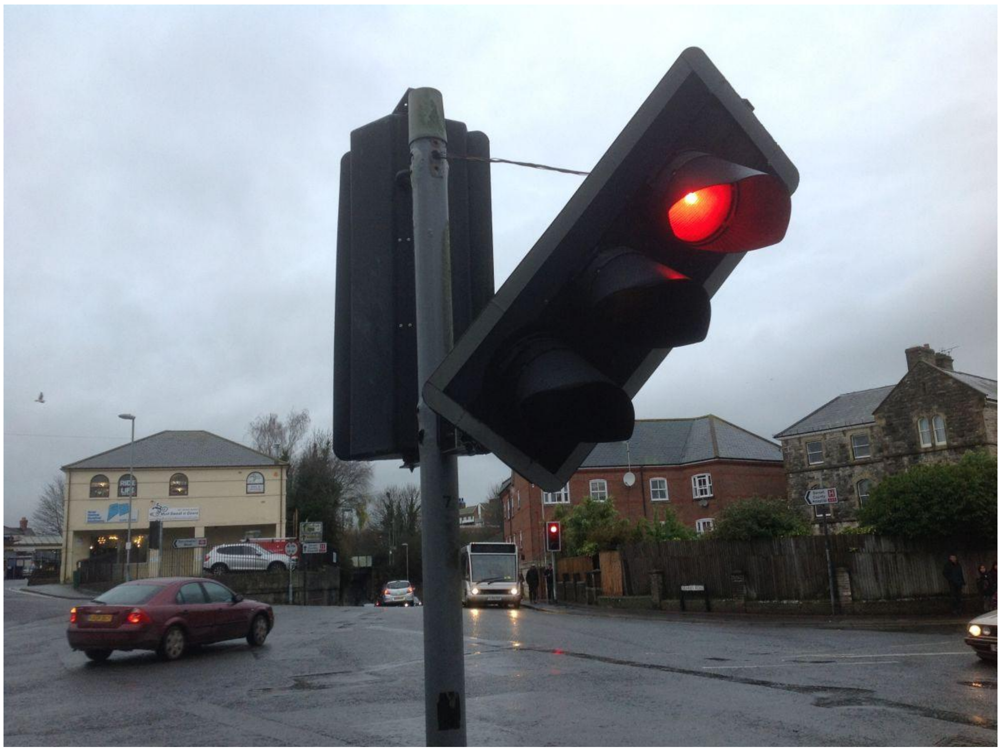


Figure 12: A damaged traffic light that could throw off the model. Source: [27]

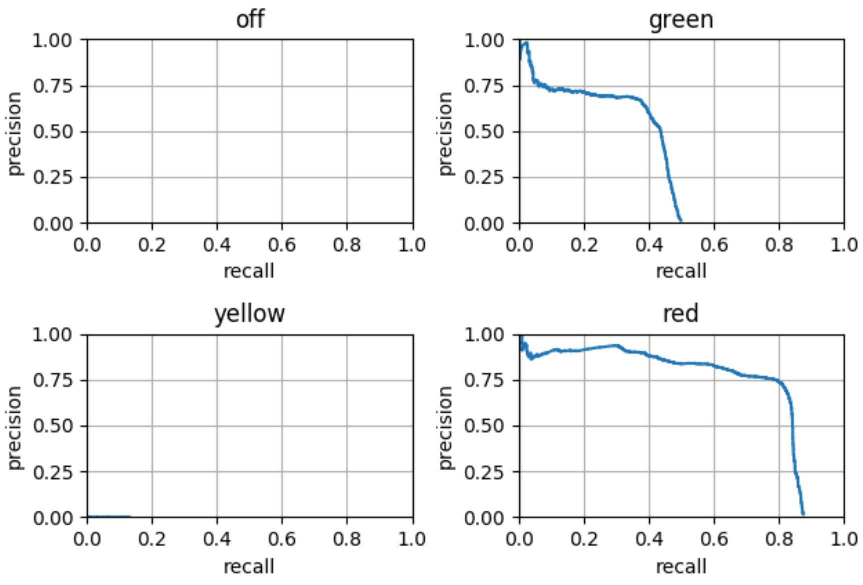


Figure 13: P/R curves for SSD Mobilenet trained on BSTLD (filtered) with batch size 4 evaluated on the rotated Freie Universität test drive data

them on the Freie Universität data. Since green and red traffic lights still get detected the only reasonable explanation is that the higher saturation in the Freie Universität images throws off the model.

Increasing the saturation of yellow in RGB color space increases the red and green values and decreases the blue value. The model will most likely confuse the higher saturation yellow with either green or red. Since the green AP suffered quite substantially compared to the BSTLD evaluation it probably confuses the higher saturation yellow with green. If that is the case, it should be visible in the confusion matrix.

Table 9: Confusion matrix of the Mobilenet model trained on BSTLD (filtered) with batch size 4

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0 | 0 | 0 | 0 |
| | **Green** | 277 | 1070 | 491 | 373 |
| | **Yellow** | 17 | 133 | 2 | 22 |
| | **Red** | 445 | 650 | 89 | 657 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.00 | 0.00 | 0.00 | 0.00 |
| | **Green** | 0.13 | 0.48 | 0.22 | 0.17 |
| | **Yellow** | 0.10 | 0.76 | 0.01 | 0.13 |
| | **Red** | 0.24 | 0.35 | 0.05 | 0.36 |

As the confusion matrix shows (Table 9), about 76% of the yellow traffic lights get detected as green traffic lights. It should be safe to say that the assumption was correct.

# 5 Conclusion and Future Work

Trained models from one dataset seem to be able to reach their performance on another similar dataset. Problems arise when the objects appear in an orientation that was not present in the training data or when the color changes too dramatically.

Some of the trained models showed promising results. With more training time they should be able to achieve decent performance. The models that fell flat might just need more training time as well, but it's probably not worth the effort, time and money.

In the context of self-driving cars it is not only important to get accurate predictions, but also to get them in a timely fashion. The architectures and resulting models should be checked for computational performance on realistic hardware. On powerful, modern hardware (RTX 2070) both architectures provide (close to) real-time performance, with SSD Mobilenet reaching about 50FPS and Faster RCNN Inception reaching about 22 FPS. An RTX 2070 will be too expensive for car manufacturers though.

Creating the dataset from a test drive was very time-consuming and most likely introduced some mislabeled traffic lights. The Freie Universität Berlin has access to a testing ground where special traffic lights are installed. These traffic lights report their location and current state to cars that are listening. Leveraging this technology it should be possible to create labels for test drives automatically, which would allow for more extensive testing and longer routes.

Some models have a hard time when the color saturation changes between datasets. It would be interesting to see if this could be remedied by increasing / decreasing the saturation of the training dataset to more closely match the saturation of the expected model input in production / evaluation.

The trained models were also susceptible to traffic lights in non-standard orientation. Natural forces, accidents and vandalism can always alter the standard appearance of traffic lights. Not being able to handle that will sooner or later result in catastrophic malfunctions. It should be explored how the resulting models can be made more resilient against changed appearance (e.g. by randomly cropping / rotating the training images).

# Bibliography

[1] Rene Schaub. What are neural networks made of? *CoRR*, abs/1909.09588, 2019. URL `http://arxiv.org/abs/1909.09588`.

[2] Google DeepMind. AlphaGo. `https://deepmind.com/research/case-studies/alphago-the-story-so-far`, 2019. Accessed: 13-December-2019.

[3] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. ISSN 1476-4687. doi: 10.1038/nature16961. URL `https://doi.org/10.1038/nature16961`.

[4] Google DeepMind. AlphaStar: Mastering the real-time strategy game StarCraft II. `https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii`, 2019. Accessed: 13-December-2019.

[5] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019. ISSN 1476-4687. doi: 10.1038/s41586-019-1724-z. URL `https://doi.org/10.1038/s41586-019-1724-z`.

[6] Robert Langner and Simon B. Eickhoff. Sustaining attention to simple tasks: a meta-analytic review of the neural mechanisms of vigilant attention. *Psychological bulletin*, 139(4):870–900, Jul 2013. ISSN 1939-1455. doi: 10.1037/a0030694. URL `https://www.ncbi.nlm.nih.gov/pubmed/23163491`.

[7] Carlos Gershenson. Artificial neural networks for beginners. *CoRR*, cs.NE/0308031, 2003. URL `http://arxiv.org/abs/cs.NE/0308031`.

[8] C. A. L. Bailer-Jones, R. Gupta, and H. P. Singh. An introduction to artificial neural networks. *CoRR*, 2001. URL `http://arxiv.org/abs/astro-ph/0102224`.

[9] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016. URL `http://arxiv.org/abs/1609.04747`.

[10] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015. URL `http://arxiv.org/abs/1511.08458`.

[11] Md Zahangir Alom, Tarek M. Taha, Chris Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Mahmudul Hasan, Brian C. Van Essen, Abdul A. S. Awwal, and Vijayan K. Asari. A state-of-the-art survey on deep learning theory and architectures. *Electronics*, 8(3), 2019. ISSN 2079-9292. doi: 10.3390/electronics8030292. URL `https://www.mdpi.com/2079-9292/8/3/292`.

[12] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, and Gang Wang. Recent advances in convolutional neural networks. *CoRR*, abs/1512.07108, 2015. URL `http://arxiv.org/abs/1512.07108`.

[13] Honglak Lee, Roger B. Grosse, Rajesh Ranganath, and Andrew Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, pages 609–616, 2009. URL `https://doi.org/10.1145/1553374.1553453`.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *CoRR*, abs/1406.4729, 2014. URL `http://arxiv.org/abs/1406.4729`.

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[16] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *CoRR*, abs/1807.05511, 2018. URL `http://arxiv.org/abs/1807.05511`.

[17] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. URL `http://arxiv.org/abs/1506.01497`.

[18] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015. URL `http://arxiv.org/abs/1512.02325`.

[19] David Powers and Ailab. Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation. *J. Mach. Learn. Technol*, 2:2229–3981, 01 2011. doi: 10.9735/2229-3981.

[20] Pablo Diez. Chapter 1 - Introduction. In *Smart Wheelchairs and Brain-Computer Interfaces*, pages 1 – 21. Academic Press, 2018. ISBN 978-0-12-812892-3. doi: https://doi.org/10.1016/B978-0-12-812892-3.00001-7. URL `http://www.sciencedirect.com/science/article/pii/B9780128128923000017`.

[21] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information processing & management*, 45(4):427–437, 2009.

[22] Karsten Behrendt, Libor Novak, and Rami Botros. A deep learning approach to traffic lights: Detection, tracking, and classification. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1370–1377. IEEE, 2017.

[23] Andreas Fregin, Julian Muller, Ulrich Krebel, and Klaus Dietmayer. The DriveU traffic light dataset: Introduction and comparison with existing datasets. In *Robotics and Automation (ICRA), 2018 IEEE International Conference on*, pages 3376–3383. IEEE, 2018.

[24] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL `http://arxiv.org/abs/1405.0312`.

[25] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. URL `http://arxiv.org/abs/1704.04861`.

[26] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015. URL `http://arxiv.org/abs/1512.00567`.

[27] Dorset ECHO. Traffic lights damaged in winds, 2014. URL `https://www.dorsetecho.co.uk/news/11663153.traffic-lights-damaged-in-winds/`. Accessed: 13-January-2020.

# List of Figures

## List of Tables

# Appendix A Confusion Matrices

### SSD Mobilenet trained on BSTLD (all) with batch size 1 and evaluated on BSTLD

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 22 | 47 | 265 | 108 |
| | **Green** | 1014 | 2924 | 2028 | 1603 |
| | **Yellow** | 0 | 152 | 0 | 2 |
| | **Red** | 277 | 1921 | 1933 | 1190 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.05 | 0.11 | 0.60 | 0.24 |
| | **Green** | 0.13 | 0.39 | 0.27 | 0.21 |
| | **Yellow** | 0.00 | 0.99 | 0.00 | 0.01 |
| | **Red** | 0.05 | 0.36 | 0.36 | 0.22 |

### SSD Mobilenet trained on BSTLD (all) with batch size 1 and evaluated on DTLD

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 3179 | 3824 | 3409 | 8284 |
| | **Green** | 7732 | 4303 | 5023 | 12560 |
| | **Yellow** | 822 | 905 | 304 | 1002 |
| | **Red** | 6407 | 3446 | 3063 | 7874 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.17 | 0.20 | 0.18 | 0.44 |
| | **Green** | 0.26 | 0.15 | 0.17 | 0.42 |
| | **Yellow** | 0.27 | 0.30 | 0.10 | 0.33 |
| | **Red** | 0.31 | 0.17 | 0.15 | 0.38 |

### SSD Mobilenet trained on BSTLD (all) with batch size 1 and evaluated on FU

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0 | 0 | 0 | 0 |
| | **Green** | 473 | 166 | 1164 | 407 |
| | **Yellow** | 0 | 174 | 0 | 0 |
| | **Red** | 1210 | 423 | 110 | 98 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.00 | 0.00 | 0.00 | 0.00 |
| | **Green** | 0.21 | 0.08 | 0.53 | 0.18 |
| | **Yellow** | 0.00 | 1.00 | 0.00 | 0.00 |
| | **Red** | 0.66 | 0.23 | 0.06 | 0.05 |

### SSD Mobilenet trained on BSTLD (all) with batch size 1 and evaluated on FU (rotated)

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0 | 0 | 0 | 0 |
| | **Green** | 459 | 143 | 1403 | 206 |
| | **Yellow** | 0 | 174 | 0 | 0 |
| | **Red** | 1504 | 121 | 179 | 37 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.00 | 0.00 | 0.00 | 0.00 |
| | **Green** | 0.21 | 0.06 | 0.63 | 0.09 |
| | **Yellow** | 0.00 | 1.00 | 0.00 | 0.00 |
| | **Red** | 0.82 | 0.07 | 0.10 | 0.02 |

**SSD Mobilenet trained on BSTLD (all) with batch size 4 and evaluated on BSTLD**

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 43 | 61 | 46 | 292 |
| | **Green** | 478 | 3528 | 1384 | 2179 |
| | **Yellow** | 1 | 121 | 9 | 23 |
| | **Red** | 337 | 1845 | 958 | 2181 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.10 | 0.14 | 0.10 | 0.66 |
| | **Green** | 0.06 | 0.47 | 0.18 | 0.29 |
| | **Yellow** | 0.01 | 0.79 | 0.06 | 0.15 |
| | **Red** | 0.06 | 0.35 | 0.18 | 0.41 |

**SSD Mobilenet trained on BSTLD (all) with batch size 4 and evaluated on DTLD**

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 2163 | 7613 | 3324 | 5596 |
| | **Green** | 3627 | 8915 | 6929 | 10147 |
| | **Yellow** | 315 | 1168 | 613 | 937 |
| | **Red** | 2333 | 7143 | 4256 | 7058 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.12 | 0.41 | 0.18 | 0.30 |
| | **Green** | 0.12 | 0.30 | 0.23 | 0.34 |
| | **Yellow** | 0.10 | 0.39 | 0.20 | 0.31 |
| | **Red** | 0.11 | 0.34 | 0.20 | 0.34 |

**SSD Mobilenet trained on BSTLD (all) with batch size 4 and evaluated on FU**

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0 | 0 | 0 | 0 |
| | **Green** | 320 | 471 | 397 | 1022 |
| | **Yellow** | 0 | 174 | 0 | 0 |
| | **Red** | 132 | 843 | 369 | 497 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.00 | 0.00 | 0.00 | 0.00 |
| | **Green** | 0.14 | 0.21 | 0.18 | 0.46 |
| | **Yellow** | 0.00 | 1.00 | 0.00 | 0.00 |
| | **Red** | 0.07 | 0.46 | 0.20 | 0.27 |

**SSD Mobilenet trained on BSTLD (all) with batch size 4 and evaluated on FU (rotated)**

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0 | 0 | 0 | 0 |
| | **Green** | 383 | 416 | 520 | 892 |
| | **Yellow** | 0 | 174 | 0 | 0 |
| | **Red** | 383 | 367 | 346 | 745 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.00 | 0.00 | 0.00 | 0.00 |
| | **Green** | 0.17 | 0.19 | 0.24 | 0.40 |
| | **Yellow** | 0.00 | 1.00 | 0.00 | 0.00 |
| | **Red** | 0.21 | 0.20 | 0.19 | 0.40 |

## SSD Mobilenet trained on BSTLD (filtered) with batch size 1 and evaluated on BSTLD

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 23 | 39 | 150 | 230 |
| | **Green** | 302 | 2658 | 1785 | 2824 |
| | **Yellow** | 0 | 145 | 8 | 1 |
| | **Red** | 142 | 1423 | 1819 | 1937 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.05 | 0.09 | 0.34 | 0.52 |
| | **Green** | 0.04 | 0.35 | 0.24 | 0.37 |
| | **Yellow** | 0.00 | 0.94 | 0.05 | 0.01 |
| | **Red** | 0.03 | 0.27 | 0.34 | 0.36 |

## SSD Mobilenet trained on BSTLD (filtered) with batch size 1 and evaluated on DTLD

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 1643 | 2437 | 8714 | 5902 |
| | **Green** | 3999 | 3520 | 11468 | 10631 |
| | **Yellow** | 415 | 565 | 990 | 1063 |
| | **Red** | 2973 | 2476 | 7854 | 7487 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.09 | 0.13 | 0.47 | 0.32 |
| | **Green** | 0.14 | 0.12 | 0.39 | 0.36 |
| | **Yellow** | 0.14 | 0.19 | 0.33 | 0.35 |
| | **Red** | 0.14 | 0.12 | 0.38 | 0.36 |

## SSD Mobilenet trained on BSTLD (filtered) with batch size 1 and evaluated on FU

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0 | 0 | 0 | 0 |
| | **Green** | 122 | 152 | 1229 | 707 |
| | **Yellow** | 0 | 174 | 0 | 0 |
| | **Red** | 79 | 1000 | 237 | 525 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.00 | 0.00 | 0.00 | 0.00 |
| | **Green** | 0.06 | 0.07 | 0.56 | 0.32 |
| | **Yellow** | 0.00 | 1.00 | 0.00 | 0.00 |
| | **Red** | 0.04 | 0.54 | 0.13 | 0.29 |

## SSD Mobilenet trained on BSTLD (filtered) with batch size 1 and evaluated on FU (rotated)

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0 | 0 | 0 | 0 |
| | **Green** | 183 | 338 | 1240 | 450 |
| | **Yellow** | 0 | 174 | 0 | 0 |
| | **Red** | 359 | 520 | 266 | 696 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.00 | 0.00 | 0.00 | 0.00 |
| | **Green** | 0.08 | 0.15 | 0.56 | 0.20 |
| | **Yellow** | 0.00 | 1.00 | 0.00 | 0.00 |
| | **Red** | 0.20 | 0.28 | 0.14 | 0.38 |

### SSD Mobilenet trained on BSTLD (filtered) with batch size 4 and evaluated on BSTLD

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 99 | 137 | 94 | 112 |
| | **Green** | 770 | 3906 | 1356 | 1537 |
| | **Yellow** | 9 | 111 | 16 | 18 |
| | **Red** | 619 | 1975 | 820 | 1907 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.22 | 0.31 | 0.21 | 0.25 |
| | **Green** | 0.10 | 0.52 | 0.18 | 0.20 |
| | **Yellow** | 0.06 | 0.72 | 0.10 | 0.12 |
| | **Red** | 0.12 | 0.37 | 0.15 | 0.36 |

### SSD Mobilenet trained on BSTLD (filtered) with batch size 4 and evaluated on DTLD

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 2034 | 8575 | 3908 | 4179 |
| | **Green** | 3888 | 11410 | 5731 | 8589 |
| | **Yellow** | 329 | 1398 | 516 | 790 |
| | **Red** | 2564 | 8495 | 3589 | 6142 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.11 | 0.46 | 0.21 | 0.22 |
| | **Green** | 0.13 | 0.39 | 0.19 | 0.29 |
| | **Yellow** | 0.11 | 0.46 | 0.17 | 0.26 |
| | **Red** | 0.12 | 0.41 | 0.17 | 0.30 |

### SSD Mobilenet trained on BSTLD (filtered) with batch size 4 and evaluated on FU

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0 | 0 | 0 | 0 |
| | **Green** | 175 | 890 | 992 | 153 |
| | **Yellow** | 0 | 174 | 0 | 0 |
| | **Red** | 204 | 938 | 598 | 101 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.00 | 0.00 | 0.00 | 0.00 |
| | **Green** | 0.08 | 0.40 | 0.45 | 0.07 |
| | **Yellow** | 0.00 | 1.00 | 0.00 | 0.00 |
| | **Red** | 0.11 | 0.51 | 0.32 | 0.05 |

### SSD Mobilenet trained on BSTLD (filtered) with batch size 4 and evaluated on FU (rotated)

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0 | 0 | 0 | 0 |
| | **Green** | 277 | 1070 | 491 | 373 |
| | **Yellow** | 17 | 133 | 2 | 22 |
| | **Red** | 445 | 650 | 89 | 657 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.00 | 0.00 | 0.00 | 0.00 |
| | **Green** | 0.13 | 0.48 | 0.22 | 0.17 |
| | **Yellow** | 0.10 | 0.76 | 0.01 | 0.13 |
| | **Red** | 0.24 | 0.35 | 0.05 | 0.36 |

**Inception trained on BSTLD (all) with batch size 1 and evaluated on BSTLD**

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 148 | 87 | 32 | 175 |
| | **Green** | 3067 | 1861 | 1146 | 1495 |
| | **Yellow** | 45 | 31 | 23 | 55 |
| | **Red** | 1988 | 992 | 531 | 1810 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.33 | 0.20 | 0.07 | 0.40 |
| | **Green** | 0.41 | 0.25 | 0.15 | 0.20 |
| | **Yellow** | 0.29 | 0.20 | 0.15 | 0.36 |
| | **Red** | 0.37 | 0.19 | 0.10 | 0.34 |

**Inception trained on BSTLD (all) with batch size 1 and evaluated on DTLD**

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 6522 | 2406 | 1805 | 7963 |
| | **Green** | 7609 | 8413 | 4903 | 8693 |
| | **Yellow** | 811 | 683 | 560 | 979 |
| | **Red** | 5336 | 3609 | 3203 | 8642 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.35 | 0.13 | 0.10 | 0.43 |
| | **Green** | 0.26 | 0.28 | 0.17 | 0.29 |
| | **Yellow** | 0.27 | 0.23 | 0.18 | 0.32 |
| | **Red** | 0.26 | 0.17 | 0.15 | 0.42 |

**Inception trained on BSTLD (all) with batch size 1 and evaluated on FU**

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0 | 0 | 0 | 0 |
| | **Green** | 386 | 378 | 561 | 885 |
| | **Yellow** | 22 | 2 | 147 | 3 |
| | **Red** | 209 | 380 | 1142 | 110 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.00 | 0.00 | 0.00 | 0.00 |
| | **Green** | 0.17 | 0.17 | 0.25 | 0.40 |
| | **Yellow** | 0.13 | 0.01 | 0.84 | 0.02 |
| | **Red** | 0.11 | 0.21 | 0.62 | 0.06 |

**Inception trained on BSTLD (all) with batch size 1 and evaluated on FU (rotated)**

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0 | 0 | 0 | 0 |
| | **Green** | 518 | 415 | 343 | 935 |
| | **Yellow** | 41 | 35 | 94 | 4 |
| | **Red** | 129 | 451 | 947 | 314 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.00 | 0.00 | 0.00 | 0.00 |
| | **Green** | 0.23 | 0.19 | 0.16 | 0.42 |
| | **Yellow** | 0.24 | 0.20 | 0.54 | 0.02 |
| | **Red** | 0.07 | 0.24 | 0.51 | 0.17 |

**Inception trained on BSTLD (all) with batch size 4 and evaluated on BSTLD**

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 201 | 188 | 25 | 28 |
| | **Green** | 2638 | 2371 | 1606 | 954 |
| | **Yellow** | 25 | 31 | 60 | 38 |
| | **Red** | 1508 | 1418 | 631 | 1764 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.45 | 0.43 | 0.06 | 0.06 |
| | **Green** | 0.35 | 0.31 | 0.21 | 0.13 |
| | **Yellow** | 0.16 | 0.20 | 0.39 | 0.25 |
| | **Red** | 0.28 | 0.27 | 0.12 | 0.33 |

**Inception trained on BSTLD (all) with batch size 4 and evaluated on DTLD**

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 10676 | 4307 | 1185 | 2528 |
| | **Green** | 12737 | 7582 | 4057 | 5242 |
| | **Yellow** | 967 | 857 | 512 | 697 |
| | **Red** | 7742 | 5393 | 2192 | 5463 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.57 | 0.23 | 0.06 | 0.14 |
| | **Green** | 0.43 | 0.26 | 0.14 | 0.18 |
| | **Yellow** | 0.32 | 0.28 | 0.17 | 0.23 |
| | **Red** | 0.37 | 0.26 | 0.11 | 0.26 |

**Inception trained on BSTLD (all) with batch size 4 and evaluated on FU**

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0 | 0 | 0 | 0 |
| | **Green** | 715 | 956 | 357 | 182 |
| | **Yellow** | 63 | 60 | 33 | 18 |
| | **Red** | 1032 | 415 | 114 | 280 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.00 | 0.00 | 0.00 | 0.00 |
| | **Green** | 0.32 | 0.43 | 0.16 | 0.08 |
| | **Yellow** | 0.36 | 0.34 | 0.19 | 0.10 |
| | **Red** | 0.56 | 0.23 | 0.06 | 0.15 |

**Inception trained on BSTLD (all) with batch size 4 and evaluated on FU (rotated)**

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0 | 0 | 0 | 0 |
| | **Green** | 586 | 764 | 654 | 207 |
| | **Yellow** | 62 | 57 | 49 | 6 |
| | **Red** | 844 | 161 | 497 | 339 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.00 | 0.00 | 0.00 | 0.00 |
| | **Green** | 0.27 | 0.35 | 0.30 | 0.09 |
| | **Yellow** | 0.36 | 0.33 | 0.28 | 0.03 |
| | **Red** | 0.46 | 0.09 | 0.27 | 0.18 |

### Inception trained on BSTLD (filtered) with batch size 1 and evaluated on BSTLD

| Groundtruth | Predicted: Off | Green | Yellow | Red |
|---|---|---|---|---|
| **Off** | 264 | 45 | 15 | 118 |
| **Green** | 2805 | 2222 | 1083 | 1459 |
| **Yellow** | 49 | 40 | 50 | 15 |
| **Red** | 2054 | 1350 | 961 | 956 |

| Groundtruth | Predicted: Off | Green | Yellow | Red |
|---|---|---|---|---|
| **Off** | 0.60 | 0.10 | 0.03 | 0.27 |
| **Green** | 0.37 | 0.29 | 0.14 | 0.19 |
| **Yellow** | 0.32 | 0.26 | 0.32 | 0.10 |
| **Red** | 0.39 | 0.25 | 0.18 | 0.18 |

### Inception trained on BSTLD (filtered) with batch size 1 and evaluated on DTLD

| Groundtruth | Predicted: Off | Green | Yellow | Red |
|---|---|---|---|---|
| **Off** | 10037 | 2721 | 1984 | 3954 |
| **Green** | 8953 | 8782 | 5706 | 6177 |
| **Yellow** | 941 | 691 | 795 | 606 |
| **Red** | 6840 | 3886 | 4366 | 5698 |

| Groundtruth | Predicted: Off | Green | Yellow | Red |
|---|---|---|---|---|
| **Off** | 0.54 | 0.15 | 0.11 | 0.21 |
| **Green** | 0.30 | 0.30 | 0.19 | 0.21 |
| **Yellow** | 0.31 | 0.23 | 0.26 | 0.20 |
| **Red** | 0.33 | 0.19 | 0.21 | 0.27 |

### Inception trained on BSTLD (filtered) with batch size 1 and evaluated on FU

| Groundtruth | Predicted: Off | Green | Yellow | Red |
|---|---|---|---|---|
| **Off** | 0 | 0 | 0 | 0 |
| **Green** | 919 | 505 | 391 | 395 |
| **Yellow** | 96 | 55 | 15 | 8 |
| **Red** | 657 | 382 | 644 | 158 |

| Groundtruth | Predicted: Off | Green | Yellow | Red |
|---|---|---|---|---|
| **Off** | 0.00 | 0.00 | 0.00 | 0.00 |
| **Green** | 0.42 | 0.23 | 0.18 | 0.18 |
| **Yellow** | 0.55 | 0.32 | 0.09 | 0.05 |
| **Red** | 0.36 | 0.21 | 0.35 | 0.09 |

### Inception trained on BSTLD (filtered) with batch size 1 and evaluated on FU (rotated)

| Groundtruth | Predicted: Off | Green | Yellow | Red |
|---|---|---|---|---|
| **Off** | 0 | 0 | 0 | 0 |
| **Green** | 1032 | 497 | 299 | 383 |
| **Yellow** | 86 | 68 | 15 | 5 |
| **Red** | 492 | 654 | 546 | 149 |

| Groundtruth | Predicted: Off | Green | Yellow | Red |
|---|---|---|---|---|
| **Off** | 0.00 | 0.00 | 0.00 | 0.00 |
| **Green** | 0.47 | 0.22 | 0.14 | 0.17 |
| **Yellow** | 0.49 | 0.39 | 0.09 | 0.03 |
| **Red** | 0.27 | 0.36 | 0.30 | 0.08 |

## Inception trained on BSTLD (filtered) with batch size 4 and evaluated on BSTLD

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 105 | 55 | 9 | 273 |
| | Green | 2425 | 2336 | 1243 | 1565 |
| | Yellow | 36 | 39 | 49 | 30 |
| | Red | 1321 | 978 | 1297 | 1725 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 0.24 | 0.12 | 0.02 | 0.62 |
| | Green | 0.32 | 0.31 | 0.16 | 0.21 |
| | Yellow | 0.23 | 0.25 | 0.32 | 0.19 |
| | Red | 0.25 | 0.18 | 0.24 | 0.32 |

## Inception trained on BSTLD (filtered) with batch size 4 and evaluated on DTLD

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 5988 | 2481 | 2186 | 8041 |
| | Green | 9968 | 5598 | 5256 | 8796 |
| | Yellow | 861 | 621 | 724 | 827 |
| | Red | 6183 | 3789 | 4174 | 6644 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 0.32 | 0.13 | 0.12 | 0.43 |
| | Green | 0.34 | 0.19 | 0.18 | 0.30 |
| | Yellow | 0.28 | 0.20 | 0.24 | 0.27 |
| | Red | 0.30 | 0.18 | 0.20 | 0.32 |

## Inception trained on BSTLD (filtered) with batch size 4 and evaluated on FU

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 0 | 0 | 0 | 0 |
| | Green | 648 | 496 | 501 | 565 |
| | Yellow | 48 | 60 | 53 | 13 |
| | Red | 673 | 338 | 709 | 121 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 0.00 | 0.00 | 0.00 | 0.00 |
| | Green | 0.29 | 0.22 | 0.23 | 0.26 |
| | Yellow | 0.28 | 0.34 | 0.30 | 0.07 |
| | Red | 0.37 | 0.18 | 0.39 | 0.07 |

## Inception trained on BSTLD (filtered) with batch size 4 and evaluated on FU (rotated)

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 0 | 0 | 0 | 0 |
| | Green | 835 | 467 | 596 | 313 |
| | Yellow | 29 | 91 | 49 | 5 |
| | Red | 909 | 213 | 617 | 102 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 0.00 | 0.00 | 0.00 | 0.00 |
| | Green | 0.38 | 0.21 | 0.27 | 0.14 |
| | Yellow | 0.17 | 0.52 | 0.28 | 0.03 |
| | Red | 0.49 | 0.12 | 0.34 | 0.06 |

### SSD Mobilenet trained on DTLD with batch size 1 and evaluated on BSTLD

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 57 | 56 | 247 | 82 |
| | Green | 1991 | 2212 | 2032 | 1334 |
| | Yellow | 5 | 147 | 1 | 1 |
| | Red | 1635 | 893 | 1546 | 1247 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 0.13 | 0.13 | 0.56 | 0.19 |
| | Green | 0.26 | 0.29 | 0.27 | 0.18 |
| | Yellow | 0.03 | 0.95 | 0.01 | 0.01 |
| | Red | 0.31 | 0.17 | 0.29 | 0.23 |

### SSD Mobilenet trained on DTLD with batch size 1 and evaluated on DTLD

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 2531 | 1630 | 12023 | 2512 |
| | Green | 5383 | 3112 | 15501 | 5622 |
| | Yellow | 621 | 828 | 1225 | 359 |
| | Red | 5223 | 2121 | 10105 | 3341 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 0.14 | 0.09 | 0.64 | 0.13 |
| | Green | 0.18 | 0.11 | 0.52 | 0.19 |
| | Yellow | 0.20 | 0.27 | 0.40 | 0.12 |
| | Red | 0.25 | 0.10 | 0.49 | 0.16 |

### SSD Mobilenet trained on DTLD with batch size 1 and evaluated on FU

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 0 | 0 | 0 | 0 |
| | Green | 97 | 149 | 869 | 1095 |
| | Yellow | 2 | 147 | 22 | 3 |
| | Red | 941 | 572 | 172 | 156 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 0.00 | 0.00 | 0.00 | 0.00 |
| | Green | 0.04 | 0.07 | 0.39 | 0.50 |
| | Yellow | 0.01 | 0.84 | 0.13 | 0.02 |
| | Red | 0.51 | 0.31 | 0.09 | 0.08 |

### SSD Mobilenet trained on DTLD with batch size 1 and evaluated on FU (rotated)

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 0 | 0 | 0 | 0 |
| | Green | 158 | 326 | 894 | 833 |
| | Yellow | 6 | 168 | 0 | 0 |
| | Red | 1488 | 190 | 142 | 21 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 0.00 | 0.00 | 0.00 | 0.00 |
| | Green | 0.07 | 0.15 | 0.40 | 0.38 |
| | Yellow | 0.03 | 0.97 | 0.00 | 0.00 |
| | Red | 0.81 | 0.10 | 0.08 | 0.01 |

**SSD Mobilenet trained on DTLD with batch size 4 and evaluated on BSTLD**

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 12 | 100 | 279 | 51 |
| | Green | 116 | 3805 | 3174 | 474 |
| | Yellow | 0 | 109 | 37 | 8 |
| | Red | 255 | 1240 | 2988 | 838 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 0.03 | 0.23 | 0.63 | 0.12 |
| | Green | 0.02 | 0.50 | 0.42 | 0.06 |
| | Yellow | 0.00 | 0.71 | 0.24 | 0.05 |
| | Red | 0.05 | 0.23 | 0.56 | 0.16 |

**SSD Mobilenet trained on DTLD with batch size 4 and evaluated on DTLD**

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 3048 | 1302 | 12751 | 1595 |
| | Green | 3138 | 9236 | 15346 | 1898 |
| | Yellow | 159 | 1604 | 1039 | 231 |
| | Red | 1934 | 3085 | 10540 | 5231 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 0.16 | 0.07 | 0.68 | 0.09 |
| | Green | 0.11 | 0.31 | 0.52 | 0.06 |
| | Yellow | 0.05 | 0.53 | 0.34 | 0.08 |
| | Red | 0.09 | 0.15 | 0.51 | 0.25 |

**SSD Mobilenet trained on DTLD with batch size 4 and evaluated on FU**

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 0 | 0 | 0 | 0 |
| | Green | 185 | 272 | 1701 | 52 |
| | Yellow | 35 | 69 | 68 | 2 |
| | Red | 58 | 812 | 933 | 38 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 0.00 | 0.00 | 0.00 | 0.00 |
| | Green | 0.08 | 0.12 | 0.77 | 0.02 |
| | Yellow | 0.20 | 0.40 | 0.39 | 0.01 |
| | Red | 0.03 | 0.44 | 0.51 | 0.02 |

**SSD Mobilenet trained on DTLD with batch size 4 and evaluated on FU (rotated)**

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 0 | 0 | 0 | 0 |
| | Green | 166 | 358 | 1613 | 74 |
| | Yellow | 6 | 116 | 21 | 31 |
| | Red | 114 | 760 | 804 | 163 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 0.00 | 0.00 | 0.00 | 0.00 |
| | Green | 0.08 | 0.16 | 0.73 | 0.03 |
| | Yellow | 0.03 | 0.67 | 0.12 | 0.18 |
| | Red | 0.06 | 0.41 | 0.44 | 0.09 |

**Inception trained on DTLD with batch size 1 and evaluated on BSTLD**

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 129 | 92 | 24 | 197 |
| | **Green** | 2303 | 2093 | 1059 | 2114 |
| | **Yellow** | 81 | 35 | 20 | 18 |
| | **Red** | 2211 | 1203 | 738 | 1169 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.29 | 0.21 | 0.05 | 0.45 |
| | **Green** | 0.30 | 0.28 | 0.14 | 0.28 |
| | **Yellow** | 0.53 | 0.23 | 0.13 | 0.12 |
| | **Red** | 0.42 | 0.23 | 0.14 | 0.22 |

**Inception trained on DTLD with batch size 1 and evaluated on DTLD**

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 6355 | 4635 | 1529 | 6177 |
| | **Green** | 7200 | 9913 | 4546 | 7959 |
| | **Yellow** | 801 | 843 | 639 | 750 |
| | **Red** | 4944 | 5528 | 3105 | 7213 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.34 | 0.25 | 0.08 | 0.33 |
| | **Green** | 0.24 | 0.33 | 0.15 | 0.27 |
| | **Yellow** | 0.26 | 0.28 | 0.21 | 0.25 |
| | **Red** | 0.24 | 0.27 | 0.15 | 0.35 |

**Inception trained on DTLD with batch size 1 and evaluated on FU**

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0 | 0 | 0 | 0 |
| | **Green** | 344 | 806 | 431 | 629 |
| | **Yellow** | 22 | 39 | 79 | 34 |
| | **Red** | 405 | 675 | 535 | 226 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.00 | 0.00 | 0.00 | 0.00 |
| | **Green** | 0.16 | 0.36 | 0.20 | 0.28 |
| | **Yellow** | 0.13 | 0.22 | 0.45 | 0.20 |
| | **Red** | 0.22 | 0.37 | 0.29 | 0.12 |

**Inception trained on DTLD with batch size 1 and evaluated on FU (rotated)**

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0 | 0 | 0 | 0 |
| | **Green** | 512 | 653 | 275 | 771 |
| | **Yellow** | 57 | 40 | 63 | 14 |
| | **Red** | 322 | 478 | 674 | 367 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | **Off** | **Green** | **Yellow** | **Red** |
| Groundtruth | **Off** | 0.00 | 0.00 | 0.00 | 0.00 |
| | **Green** | 0.23 | 0.30 | 0.12 | 0.35 |
| | **Yellow** | 0.33 | 0.23 | 0.36 | 0.08 |
| | **Red** | 0.17 | 0.26 | 0.37 | 0.20 |

**Inception trained on DTLD with batch size 4 and evaluated on BSTLD**

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 127 | 217 | 39 | 59 |
| | Green | 2073 | 2594 | 1571 | 1331 |
| | Yellow | 53 | 28 | 36 | 37 |
| | Red | 1567 | 1402 | 917 | 1435 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 0.29 | 0.49 | 0.09 | 0.13 |
| | Green | 0.27 | 0.34 | 0.21 | 0.18 |
| | Yellow | 0.34 | 0.18 | 0.23 | 0.24 |
| | Red | 0.29 | 0.26 | 0.17 | 0.27 |

**Inception trained on DTLD with batch size 4 and evaluated on DTLD**

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 8149 | 3762 | 1886 | 4899 |
| | Green | 8776 | 8673 | 5410 | 6759 |
| | Yellow | 846 | 686 | 717 | 784 |
| | Red | 6030 | 4239 | 3833 | 6688 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 0.44 | 0.20 | 0.10 | 0.26 |
| | Green | 0.30 | 0.29 | 0.18 | 0.23 |
| | Yellow | 0.28 | 0.23 | 0.24 | 0.26 |
| | Red | 0.29 | 0.20 | 0.18 | 0.32 |

**Inception trained on DTLD with batch size 4 and evaluated on FU**

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 0 | 0 | 0 | 0 |
| | Green | 455 | 847 | 429 | 479 |
| | Yellow | 20 | 37 | 60 | 57 |
| | Red | 274 | 381 | 690 | 496 |

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 0.00 | 0.00 | 0.00 | 0.00 |
| | Green | 0.21 | 0.38 | 0.19 | 0.22 |
| | Yellow | 0.11 | 0.21 | 0.34 | 0.33 |
| | Red | 0.15 | 0.21 | 0.37 | 0.27 |

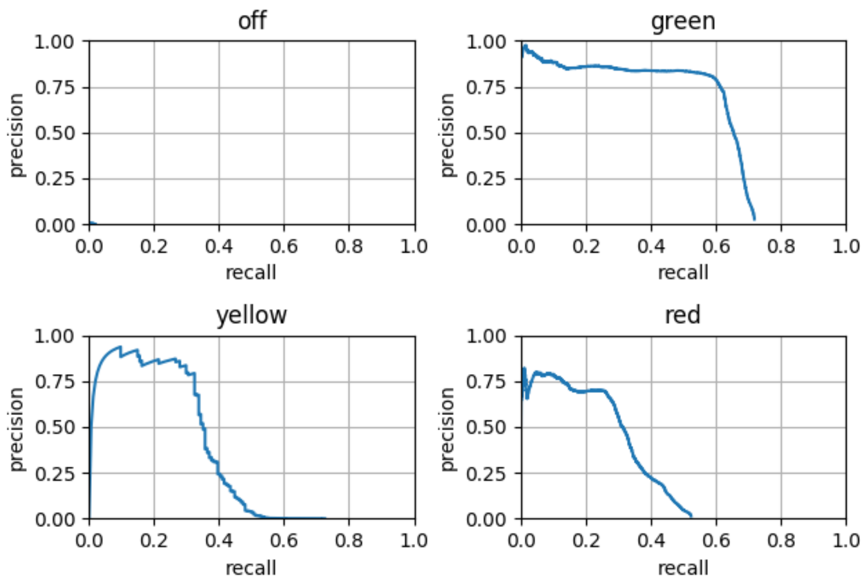
**Inception trained on DTLD with batch size 4 and evaluated on FU (rotated)**

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 0 | 0 | 0 | 0 |
| | Green | 700 | 615 | 394 | 502 |
| | Yellow | 5 | 58 | 88 | 23 |
| | Red | 233 | 416 | 777 | 415 |

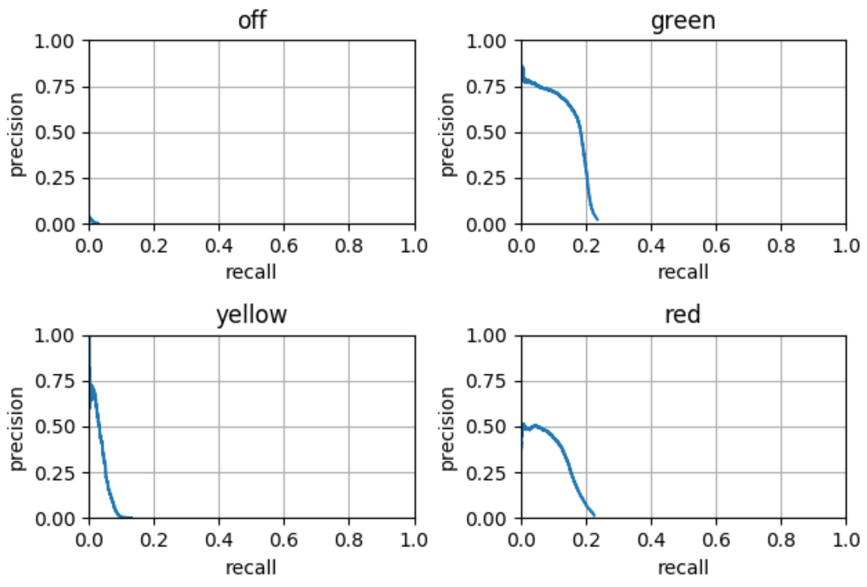
| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Off | Green | Yellow | Red |
| Groundtruth | Off | 0.00 | 0.00 | 0.00 | 0.00 |
| | Green | 0.32 | 0.28 | 0.18 | 0.23 |
| | Yellow | 0.03 | 0.33 | 0.51 | 0.13 |
| | Red | 0.13 | 0.23 | 0.42 | 0.23 |

# Appendix B P/R Curves

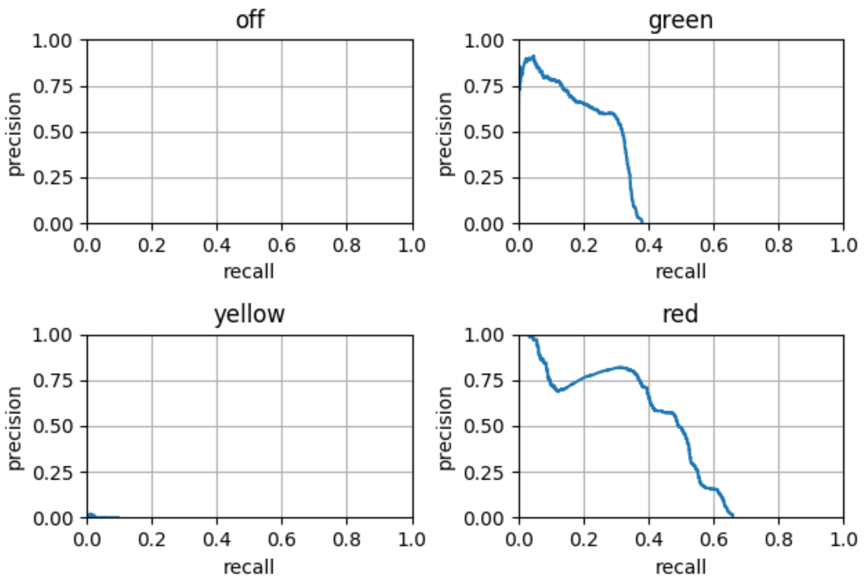
**SSD Mobilenet trained on BSTLD (all) with batch size 1 and evaluated on BSTLD**



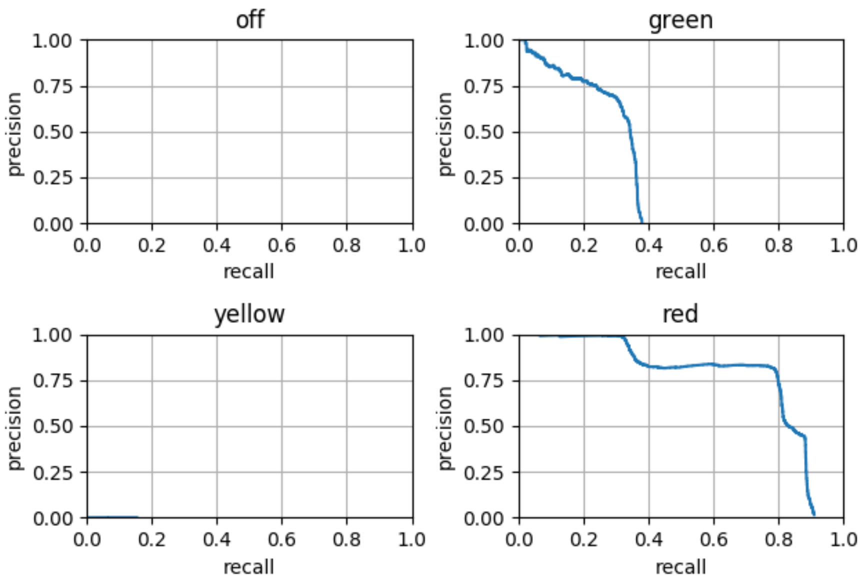
**SSD Mobilenet trained on BSTLD (all) with batch size 1 and evaluated on DTLD**



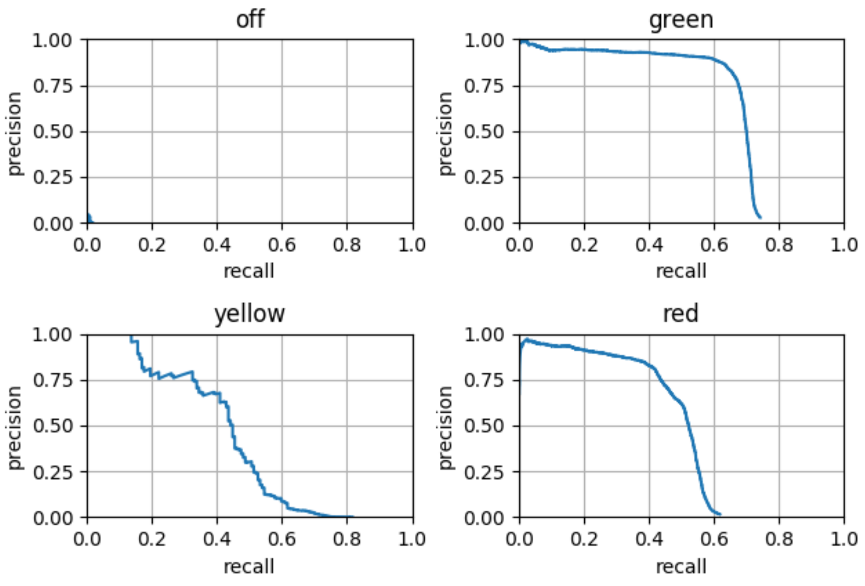
**SSD Mobilenet trained on BSTLD (all) with batch size 1 and evaluated on FU**



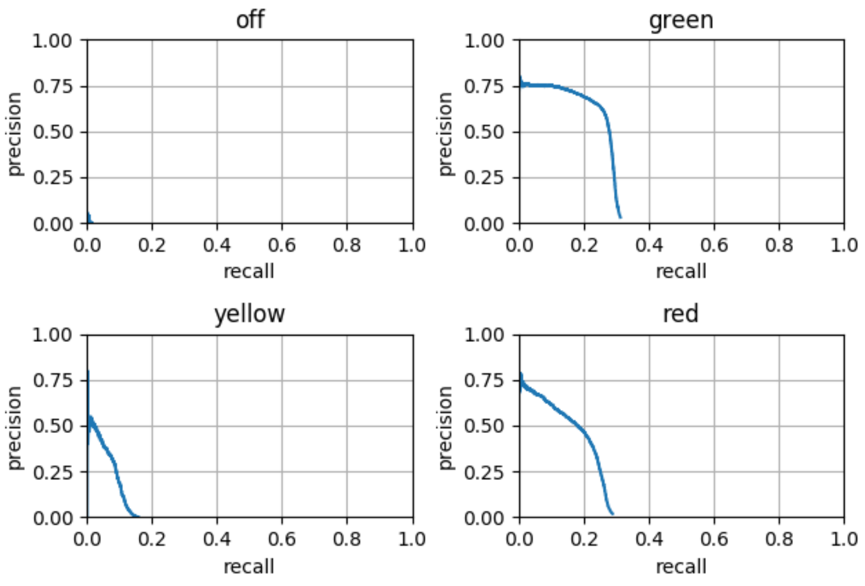
**SSD Mobilenet trained on BSTLD (all) with batch size 1 and evaluated on FU (rotated)**



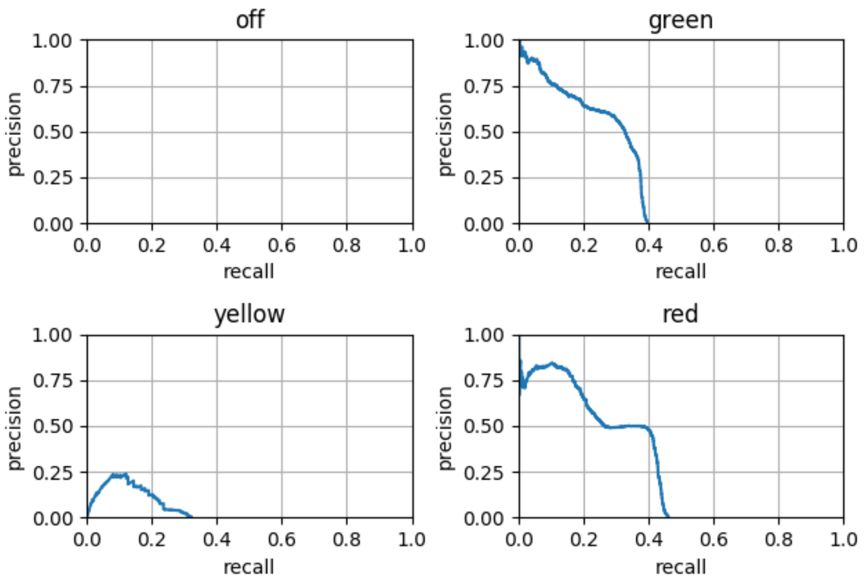
**SSD Mobilenet trained on BSTLD (all) with batch size 4 and evaluated on BSTLD**



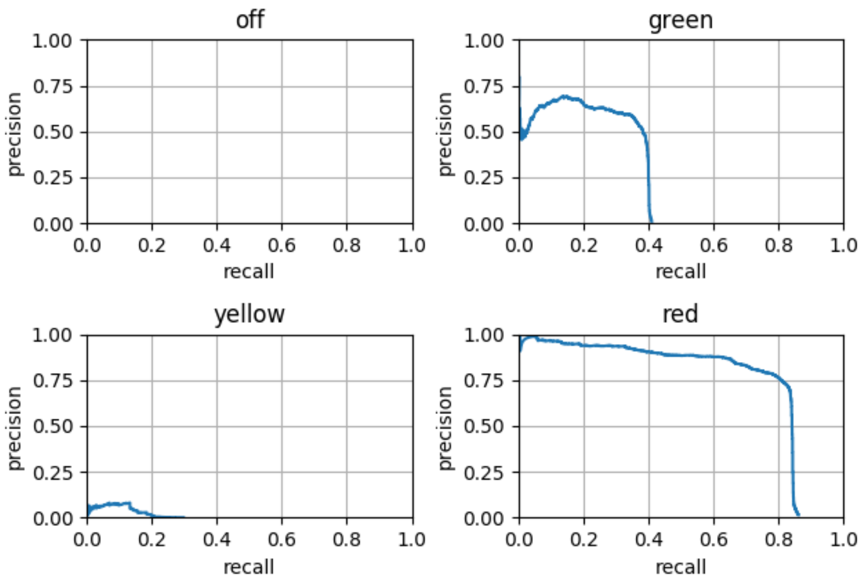
**SSD Mobilenet trained on BSTLD (all) with batch size 4 and evaluated on DTLD**



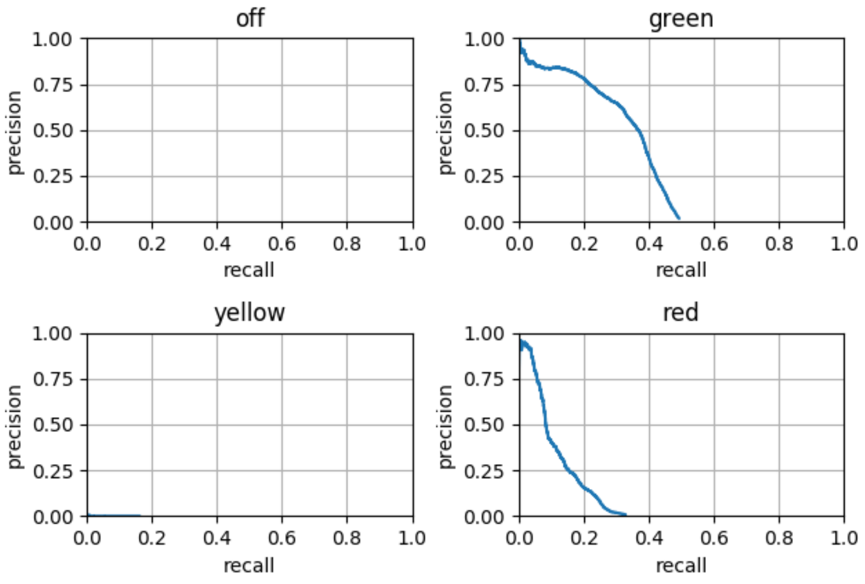
**SSD Mobilenet trained on BSTLD (all) with batch size 4 and evaluated on FU**



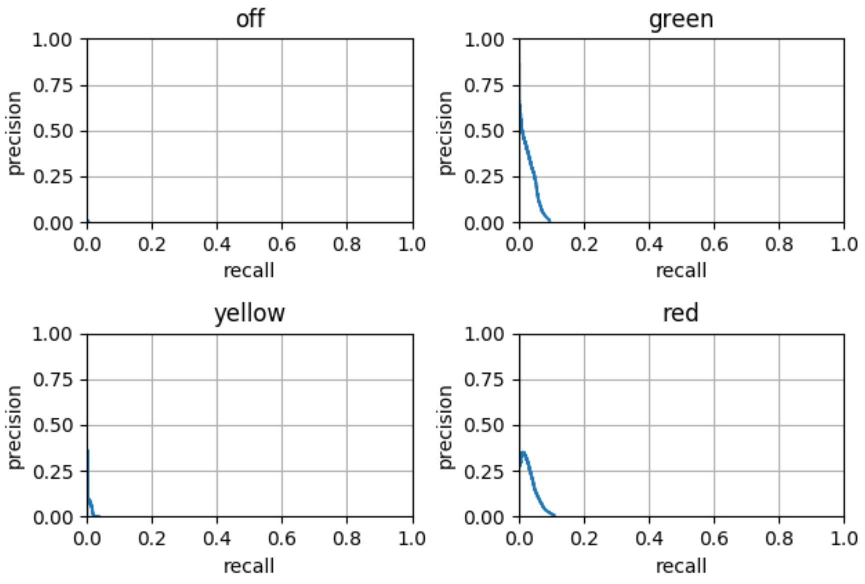
**SSD Mobilenet trained on BSTLD (all) with batch size 4 and evaluated on FU (rotated)**



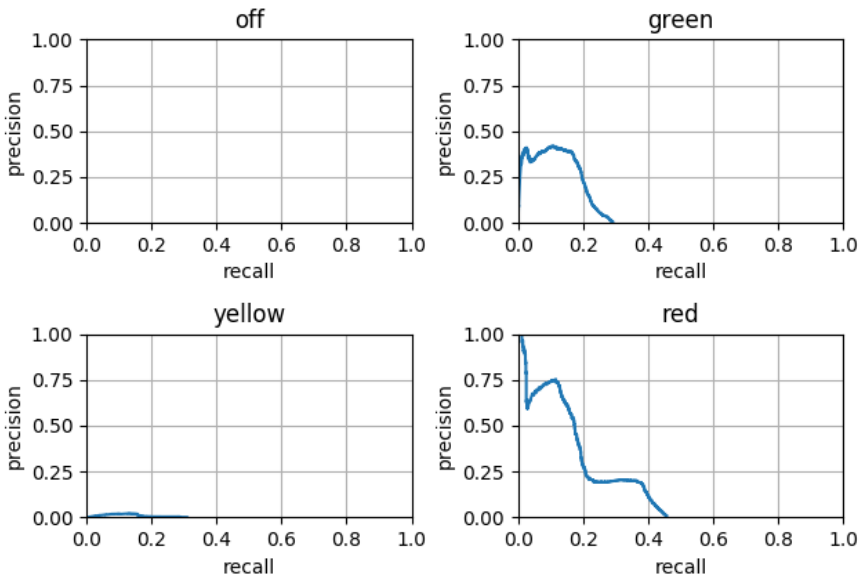
**SSD Mobilenet trained on BSTLD (filtered) with batch size 1 and evaluated on BSTLD**



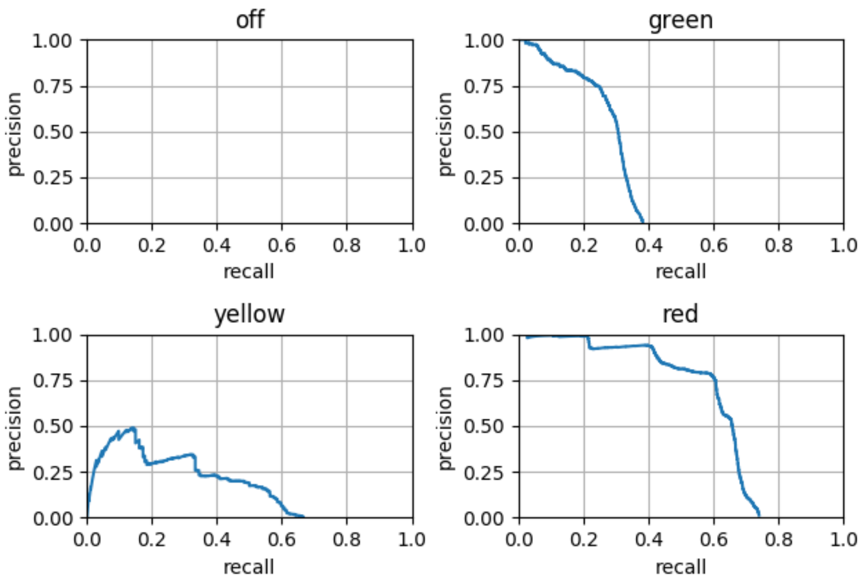
**SSD Mobilenet trained on BSTLD (filtered) with batch size 1 and evaluated on DTLD**



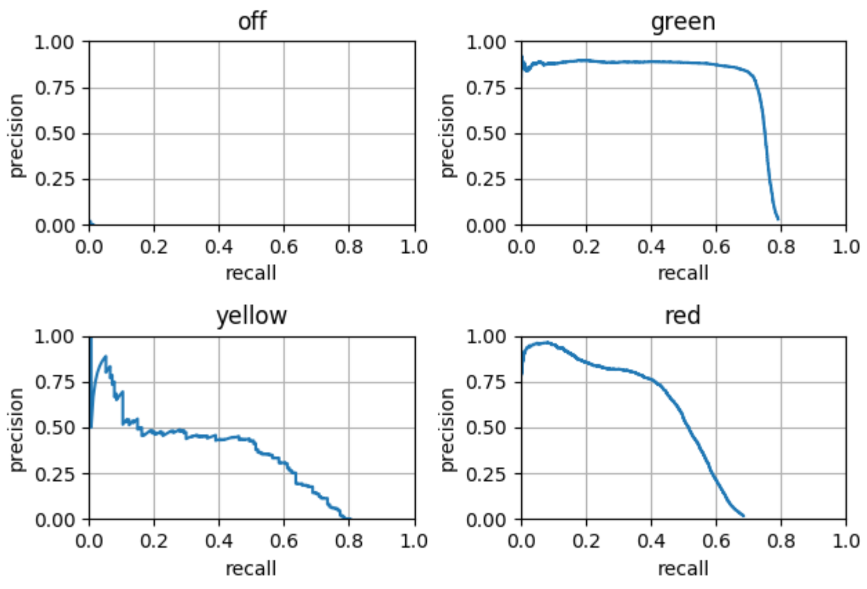
**SSD Mobilenet trained on BSTLD (filtered) with batch size 1 and evaluated on FU**



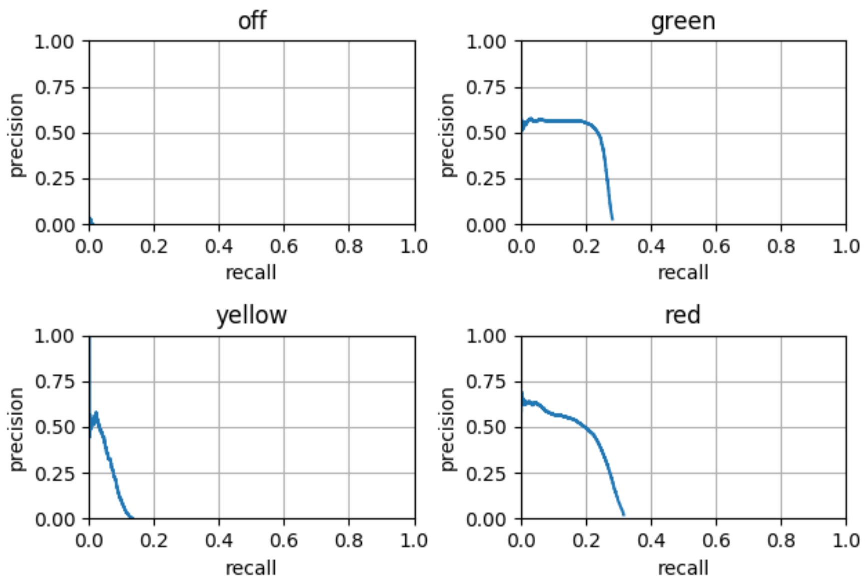
**SSD Mobilenet trained on BSTLD (filtered) with batch size 1 and evaluated on FU (rotated)**



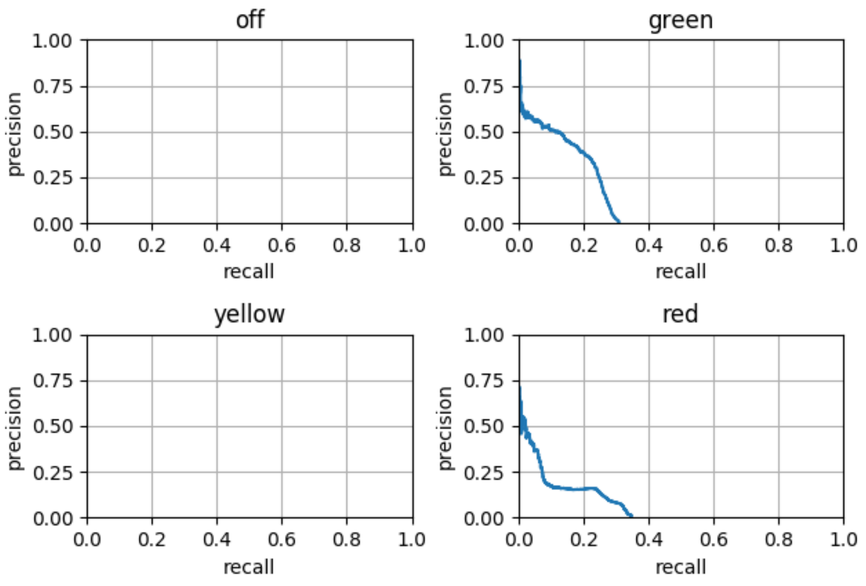
**SSD Mobilenet trained on BSTLD (filtered) with batch size 4 and evaluated on BSTLD**



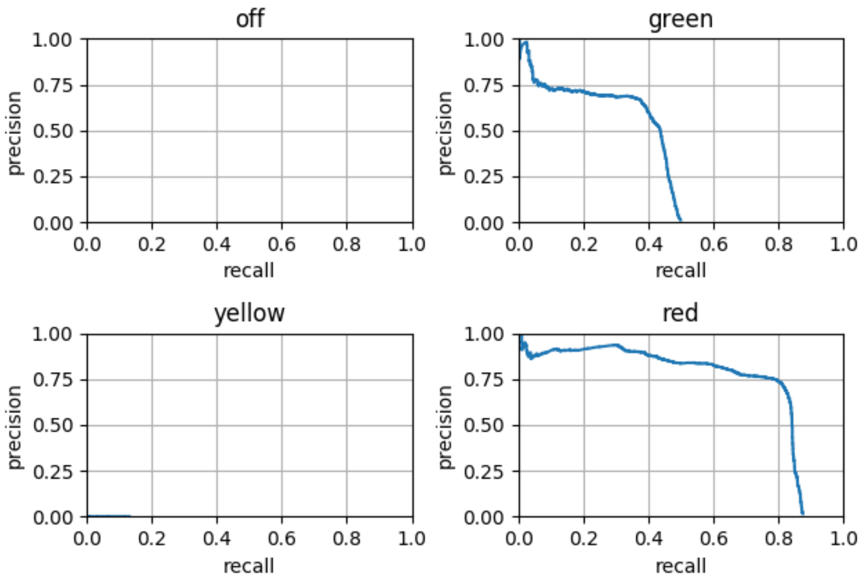
**SSD Mobilenet trained on BSTLD (filtered) with batch size 4 and evaluated on DTLD**



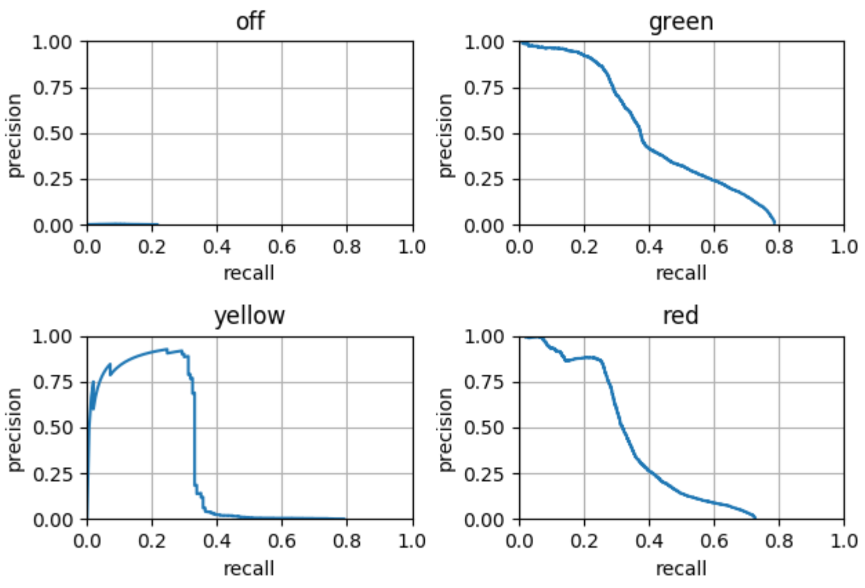
**SSD Mobilenet trained on BSTLD (filtered) with batch size 4 and evaluated on FU**



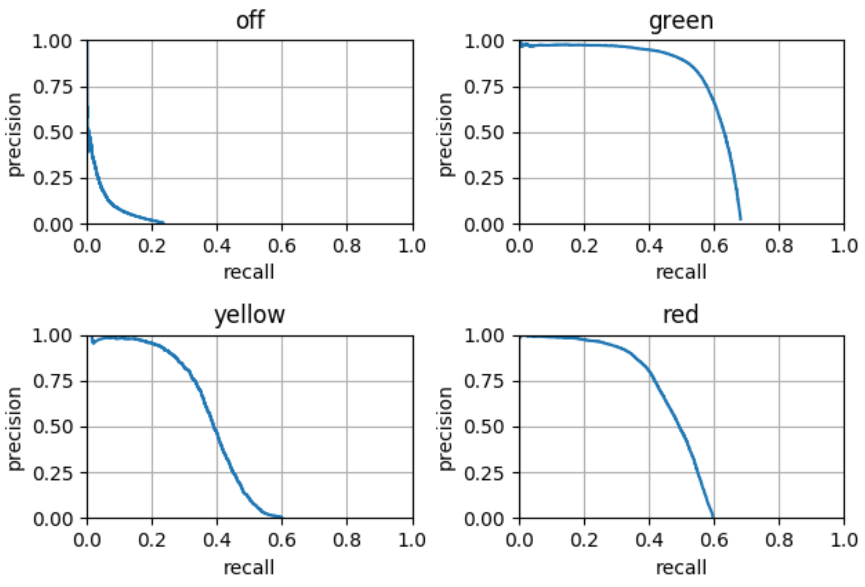
**SSD Mobilenet trained on BSTLD (filtered) with batch size 4 and evaluated on FU (rotated)**



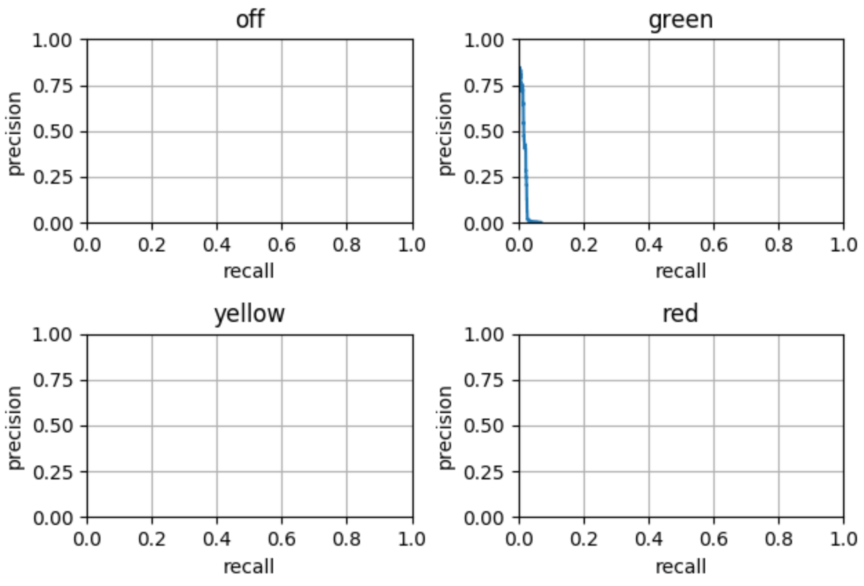
**Inception trained on BSTLD (all) with batch size 1 and evaluated on BSTLD**



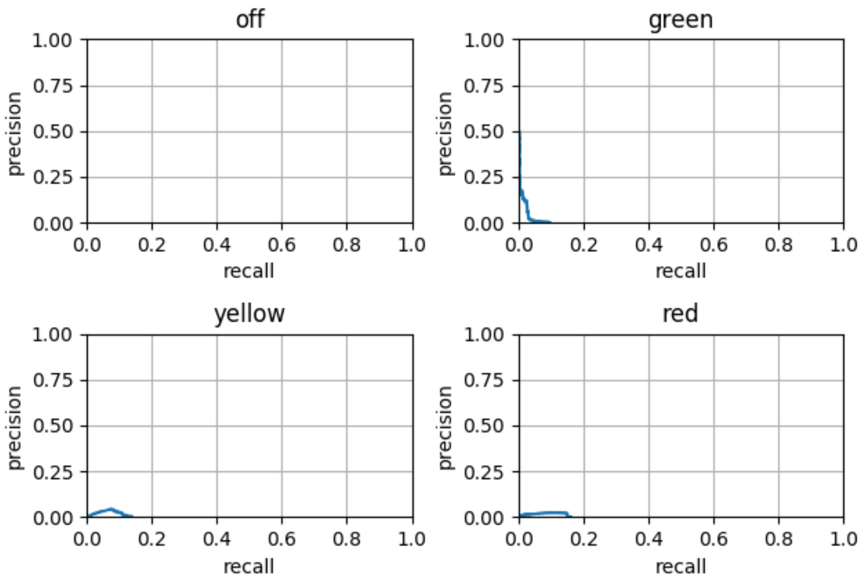
**Inception trained on BSTLD (all) with batch size 1 and evaluated on DTLD**



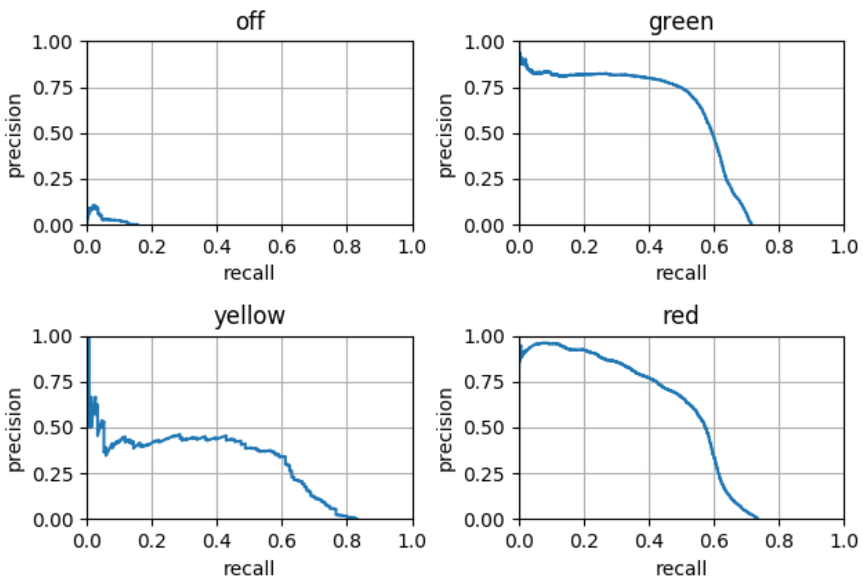
**Inception trained on BSTLD (all) with batch size 1 and evaluated on FU**



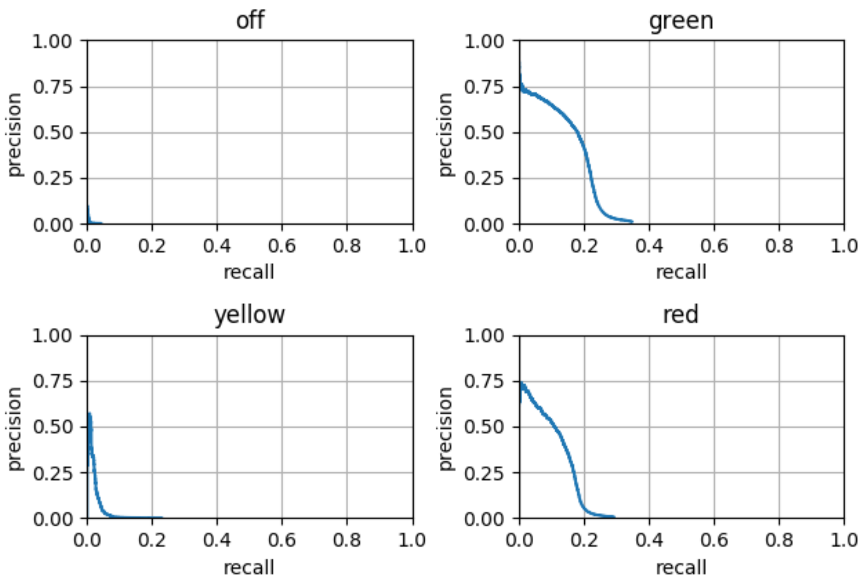
**Inception trained on BSTLD (all) with batch size 1 and evaluated on FU (rotated)**



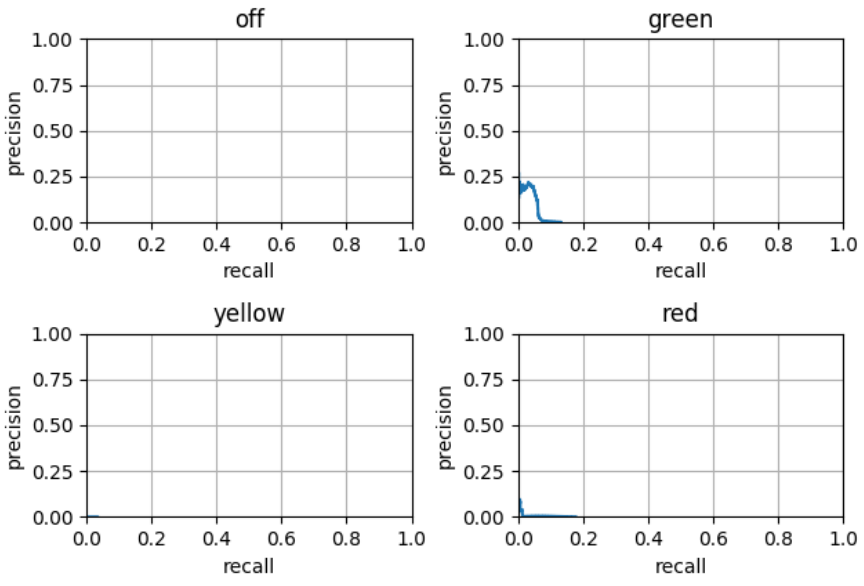
**Inception trained on BSTLD (all) with batch size 4 and evaluated on BSTLD**



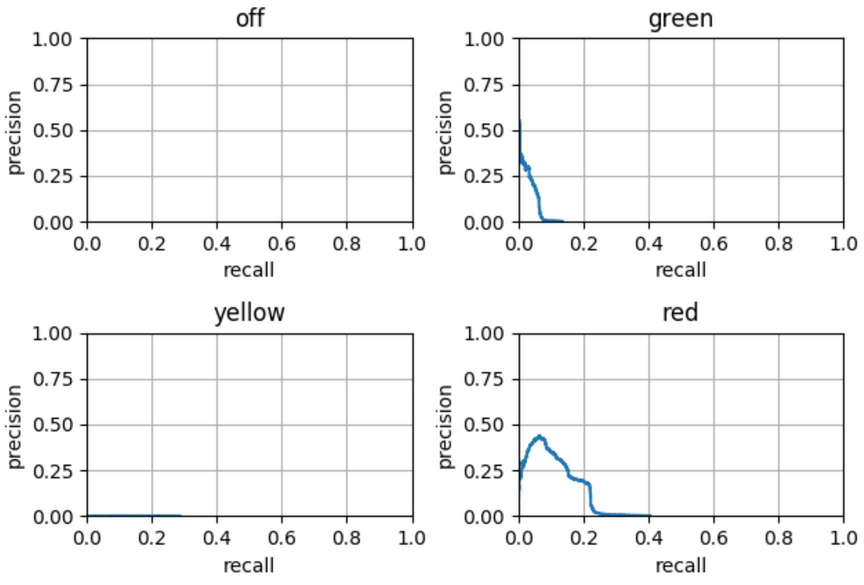
**Inception trained on BSTLD (all) with batch size 4 and evaluated on DTLD**



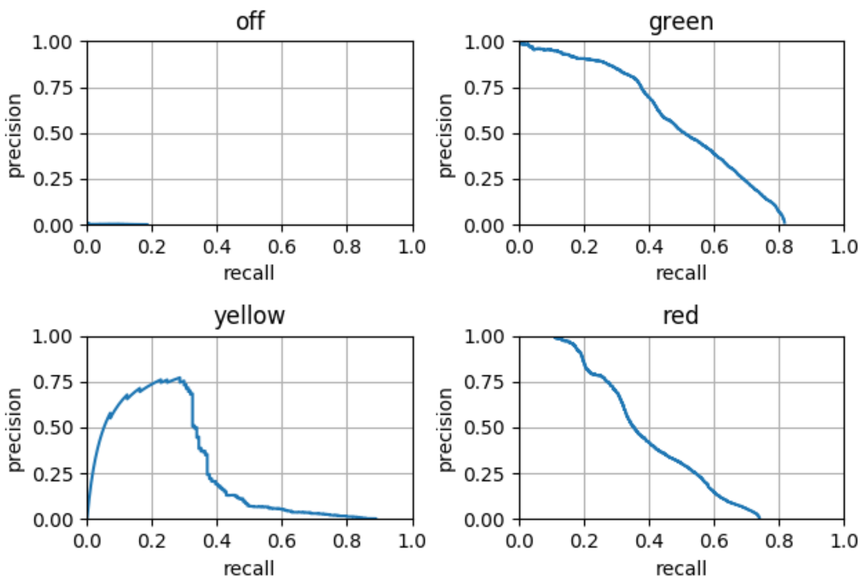
**Inception trained on BSTLD (all) with batch size 4 and evaluated on FU**



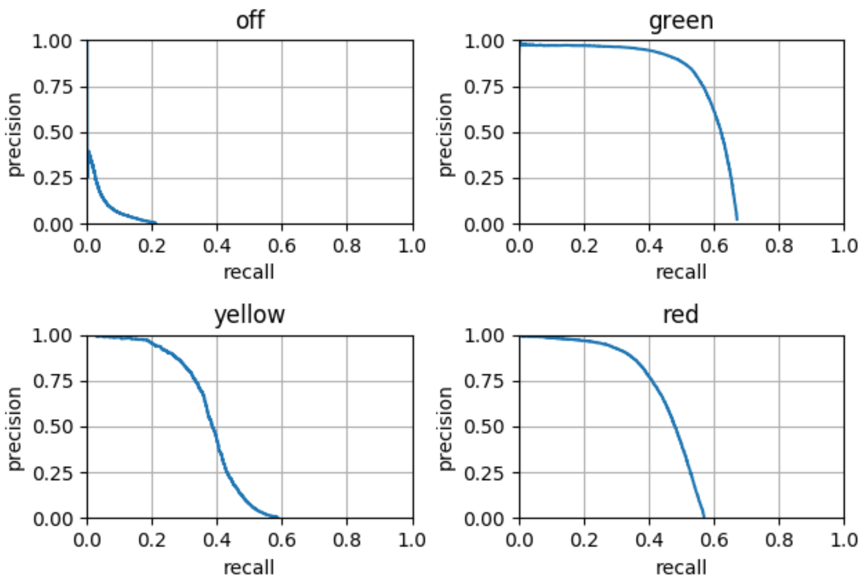
**Inception trained on BSTLD (all) with batch size 4 and evaluated on FU (rotated)**



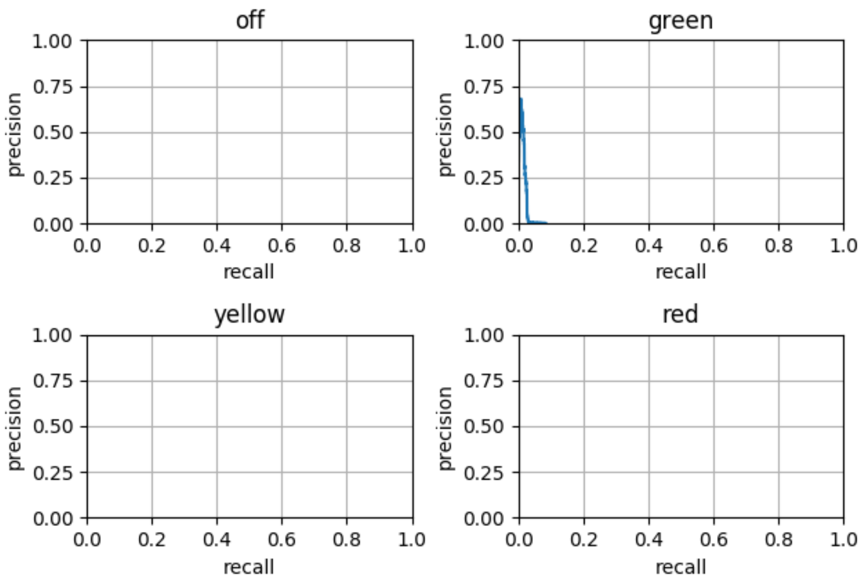
**Inception trained on BSTLD (filtered) with batch size 1 and evaluated on BSTLD**



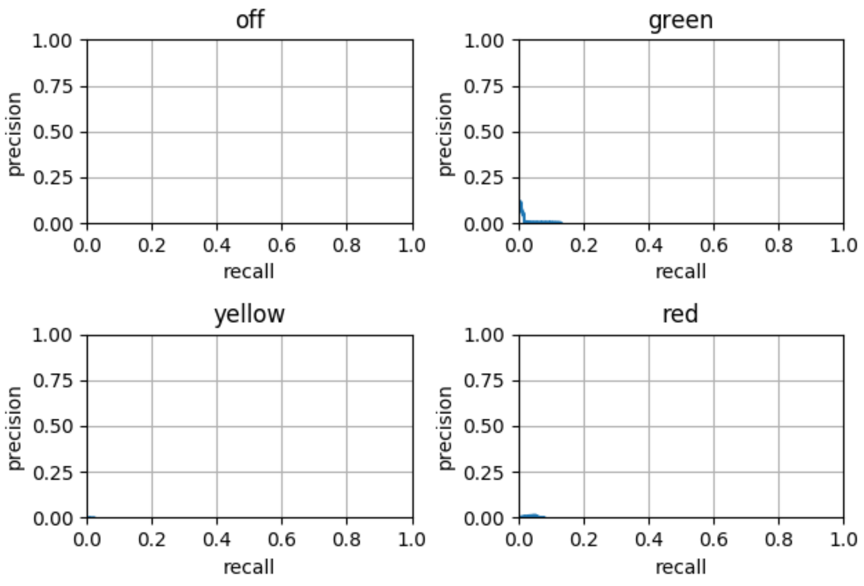
**Inception trained on BSTLD (filtered) with batch size 1 and evaluated on DTLD**



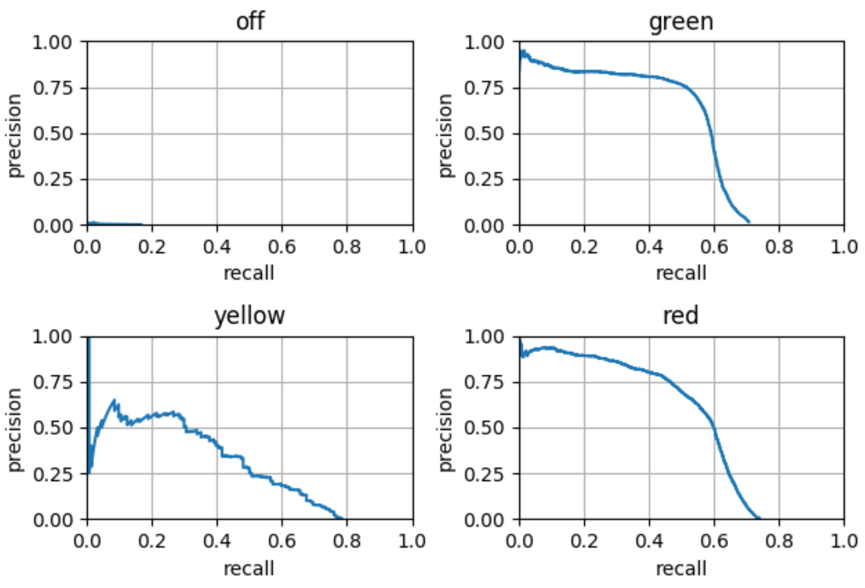
**Inception trained on BSTLD (filtered) with batch size 1 and evaluated on FU**



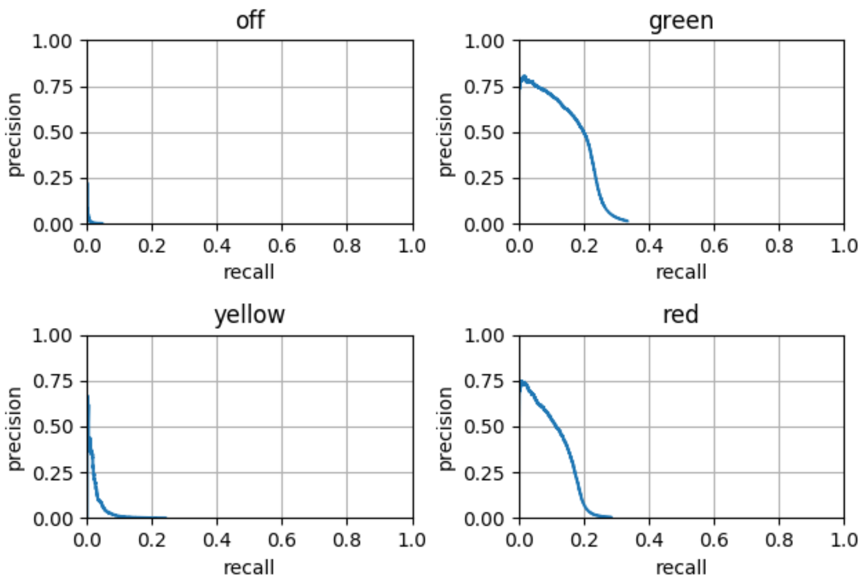
**Inception trained on BSTLD (filtered) with batch size 1 and evaluated on FU (rotated)**



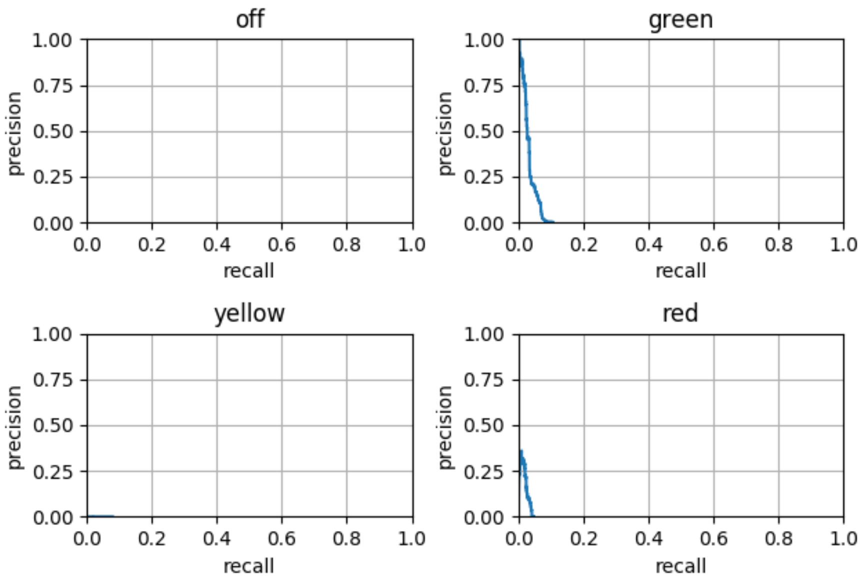
**Inception trained on BSTLD (filtered) with batch size 4 and evaluated on BSTLD**



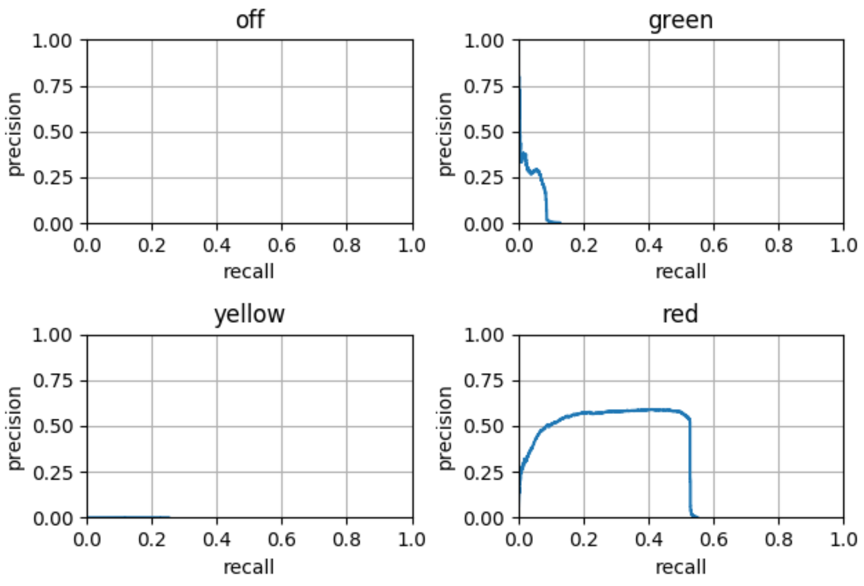
**Inception trained on BSTLD (filtered) with batch size 4 and evaluated on DTLD**



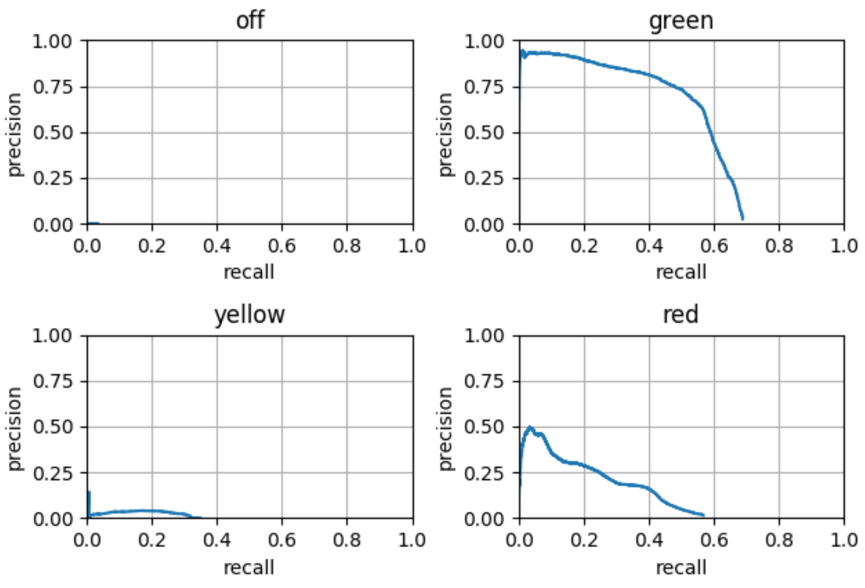
**Inception trained on BSTLD (filtered) with batch size 4 and evaluated on FU**



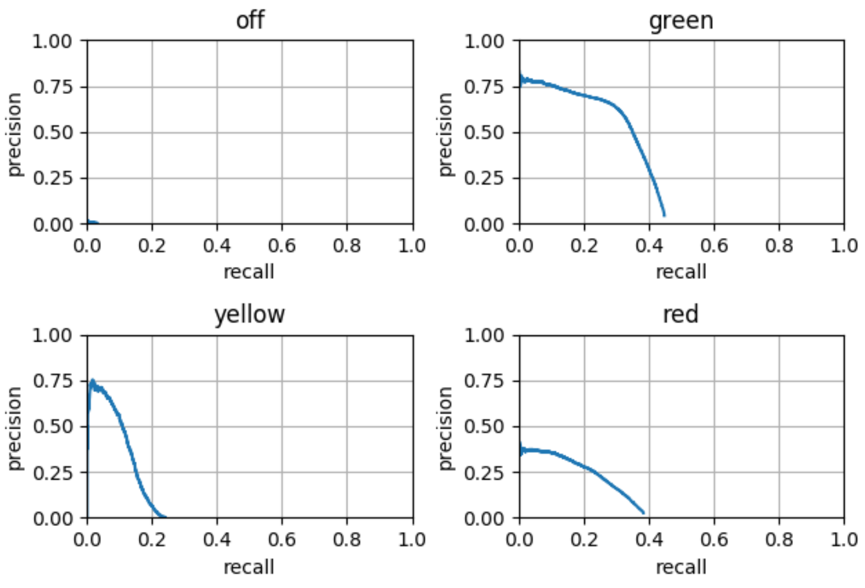
**Inception trained on BSTLD (filtered) with batch size 4 and evaluated on FU (rotated)**



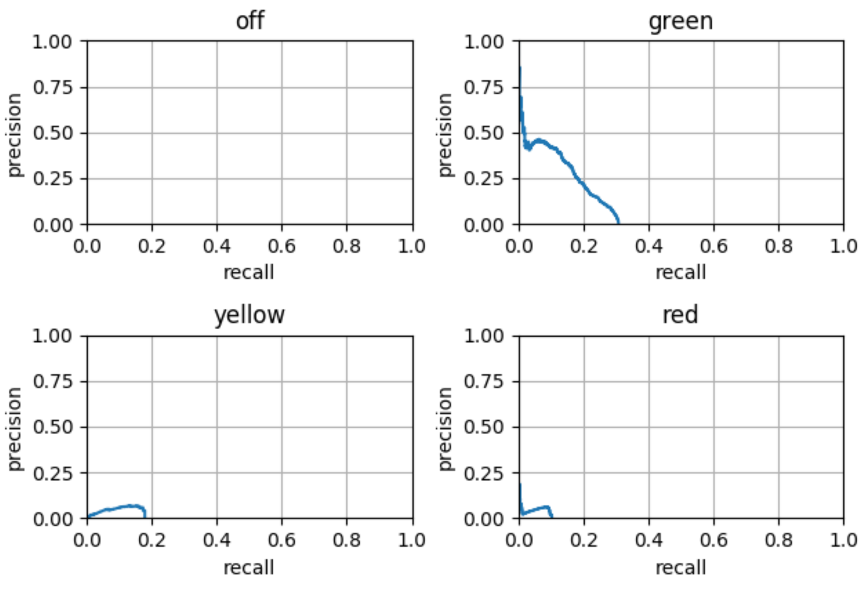
**SSD Mobilenet trained on DTLD with batch size 1 and evaluated on BSTLD**



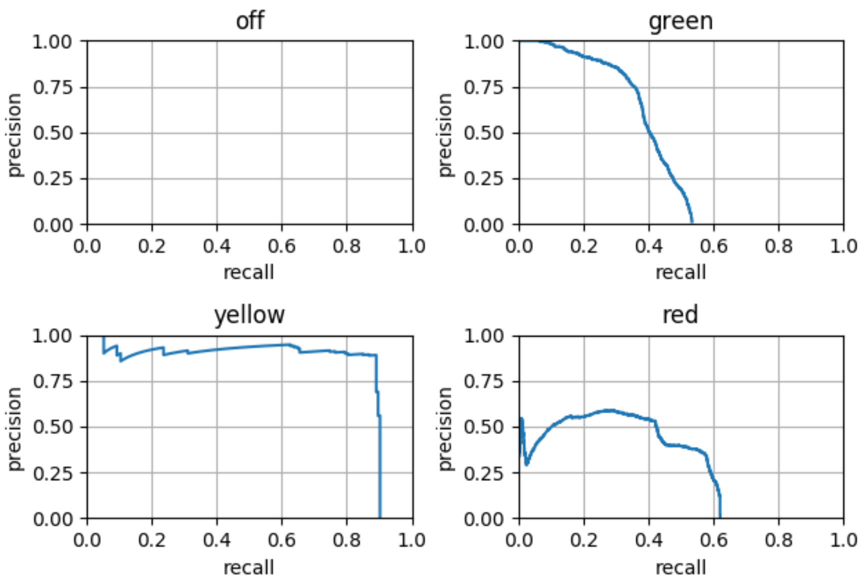
**SSD Mobilenet trained on DTLD with batch size 1 and evaluated on DTLD**



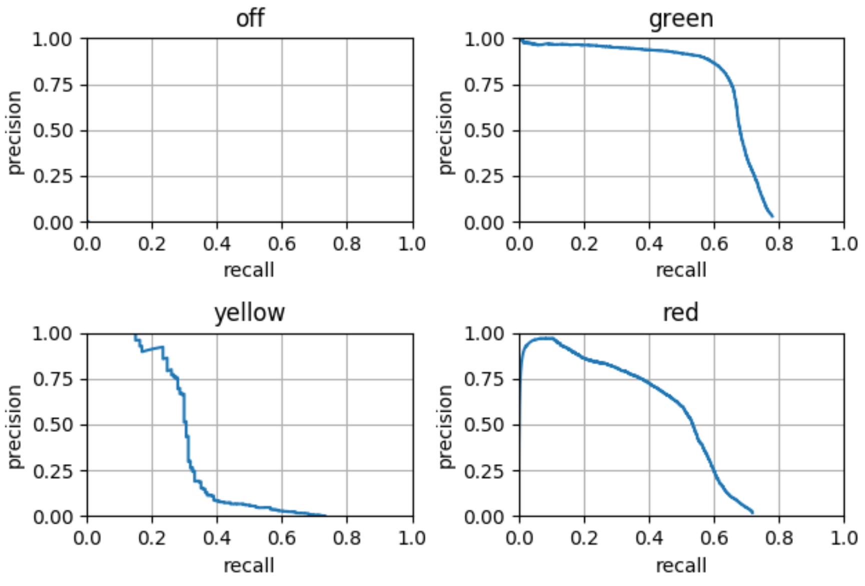
**SSD Mobilenet trained on DTLD with batch size 1 and evaluated on FU**



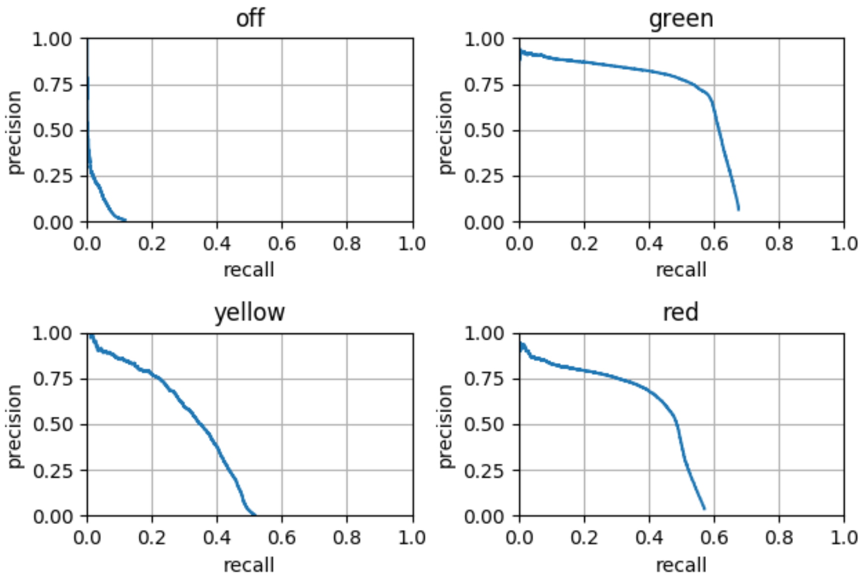
**SSD Mobilenet trained on DTLD with batch size 1 and evaluated on FU (rotated)**



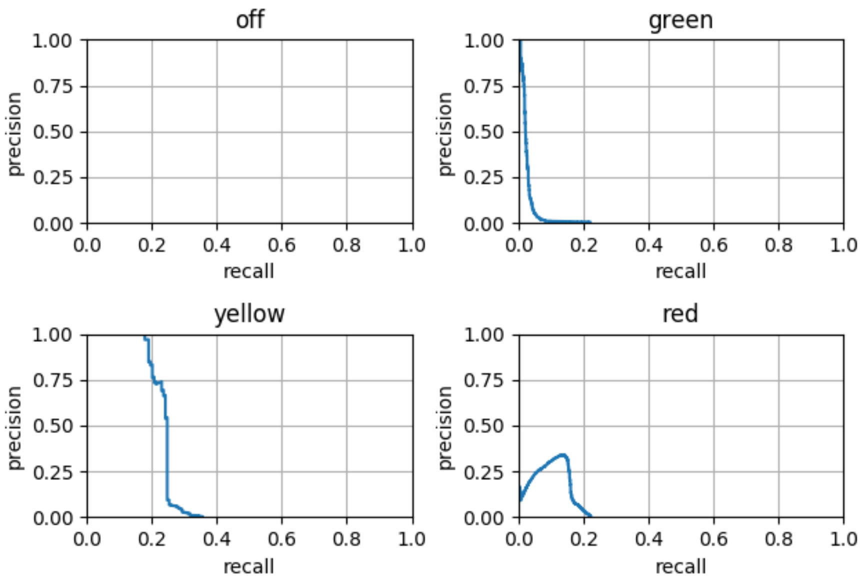
**SSD Mobilenet trained on DTLD with batch size 4 and evaluated on BSTLD**



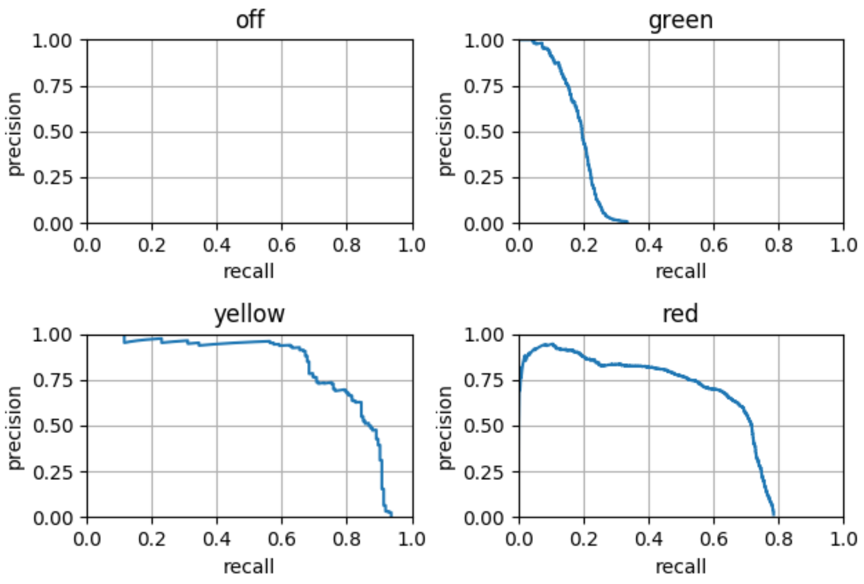
**SSD Mobilenet trained on DTLD with batch size 4 and evaluated on DTLD**



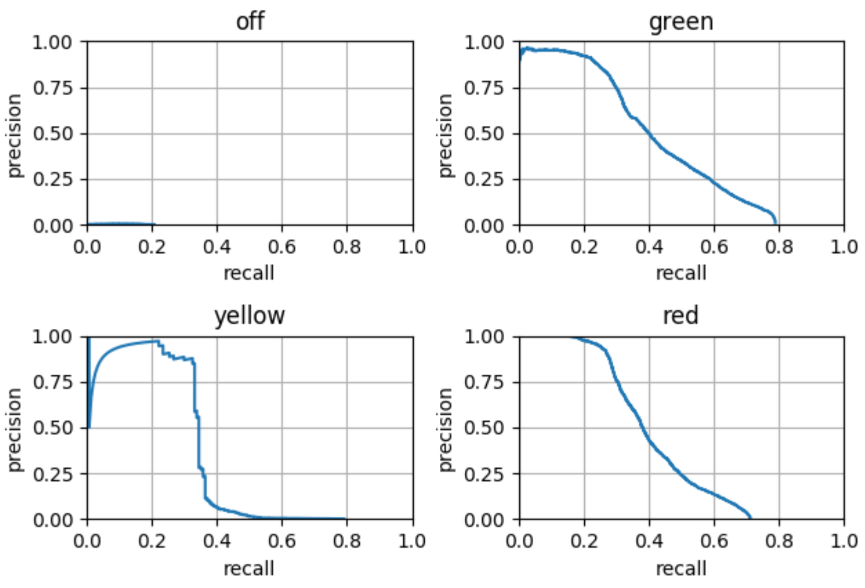
**SSD Mobilenet trained on DTLD with batch size 4 and evaluated on FU**



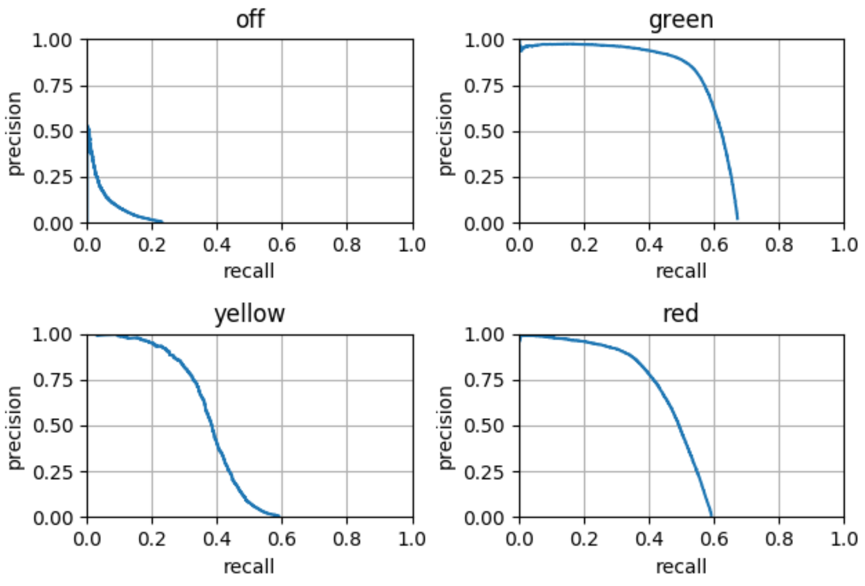
**SSD Mobilenet trained on DTLD with batch size 4 and evaluated on FU (rotated)**



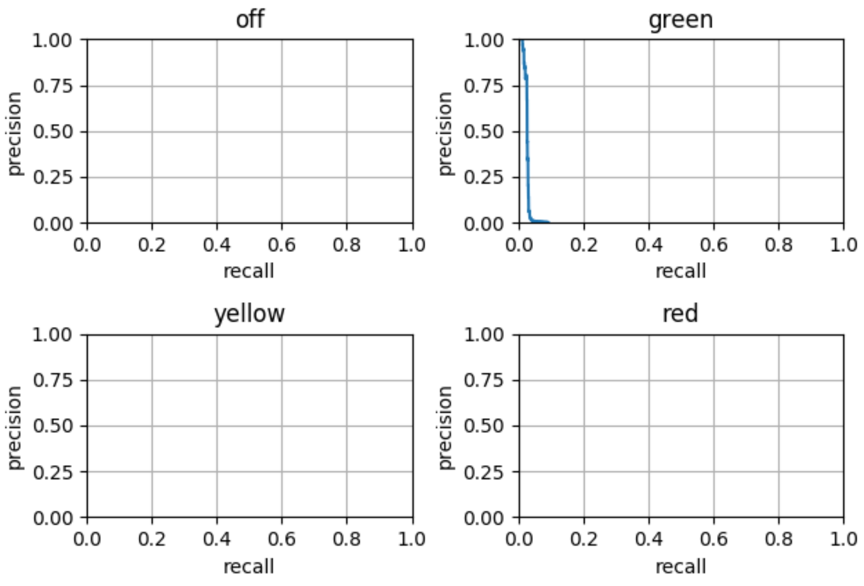
**Inception trained on DTLD with batch size 1 and evaluated on BSTLD**



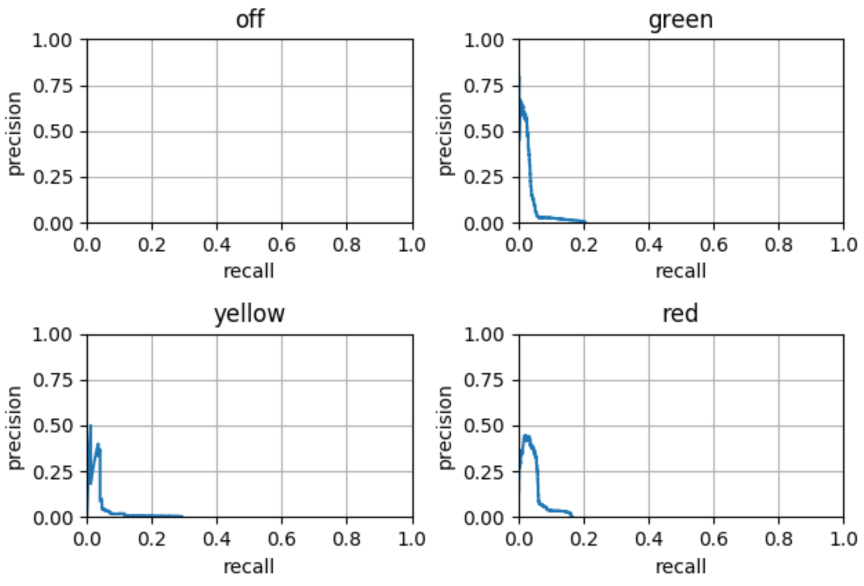
**Inception trained on DTLD with batch size 1 and evaluated on DTLD**



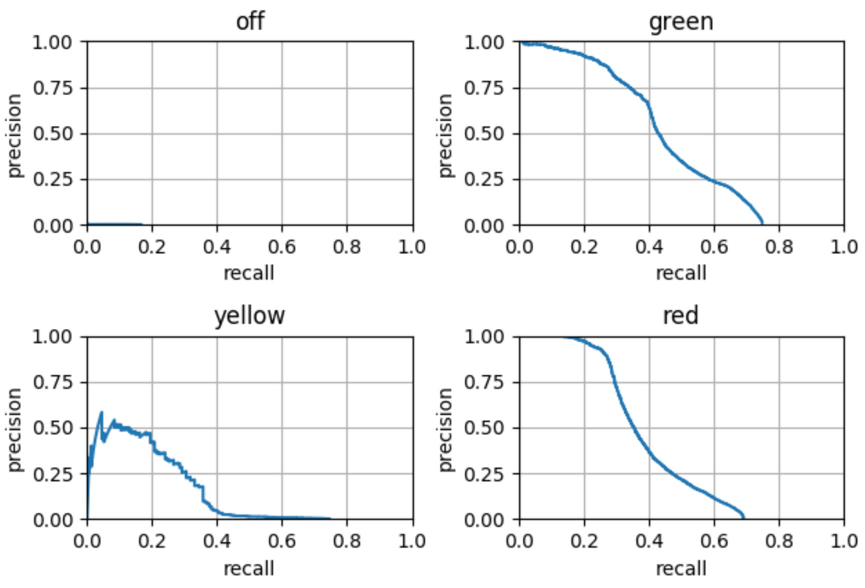
**Inception trained on DTLD with batch size 1 and evaluated on FU**



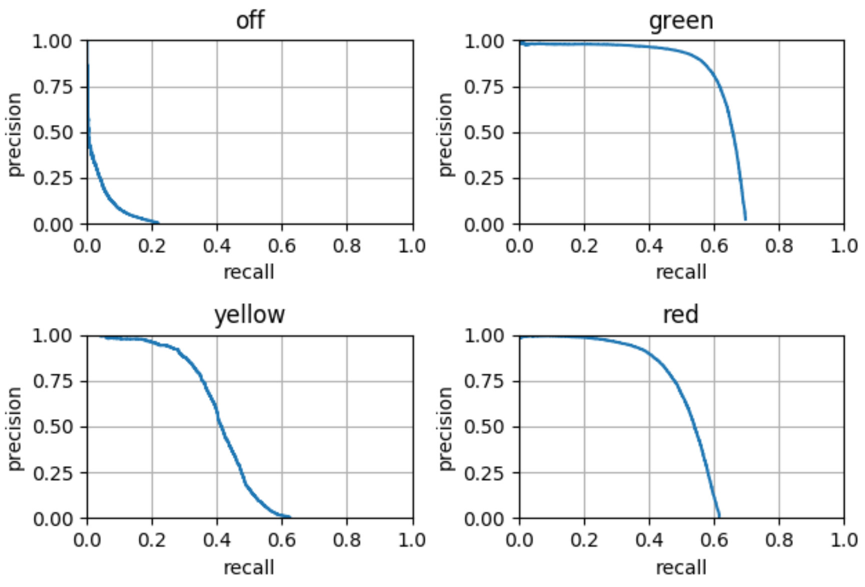
**Inception trained on DTLD with batch size 1 and evaluated on FU (rotated)**



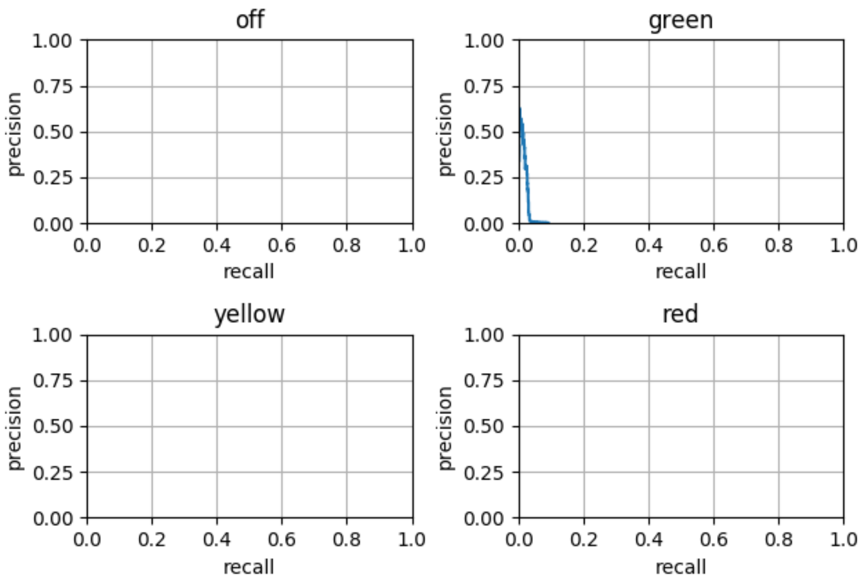
**Inception trained on DTLD with batch size 4 and evaluated on BSTLD**



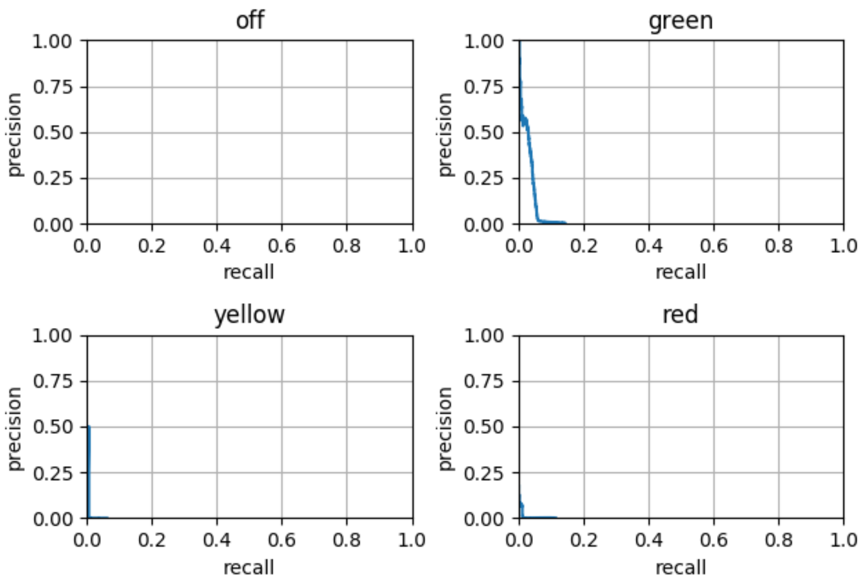
**Inception trained on DTLD with batch size 4 and evaluated on DTLD**



## Inception trained on DTLD with batch size 4 and evaluated on FU



## Inception trained on DTLD with batch size 4 and evaluated on FU (rotated)



# Appendix C (Weighted) mAP

**SSD Mobilenet v1 trained on BSTLD (all) with batch size 1**

| Evaluation Dataset | Weighted mAP | mAP | Off | Green | Yellow | Red |
|---|---|---|---|---|---|---|
| BSTLD | 0.4348 | 0.294 | 0.0 | 0.568 | 0.3366 | 0.2714 |
| DTLD | 0.0805 | 0.0631 | 0.0003 | 0.1416 | 0.0331 | 0.0775 |
| FU | 0.3026 | 0.223 | 0.0 | 0.2427 | 0.0004 | 0.4259 |
| FU (rotated) | 0.4724 | 0.3509 | 0.0 | 0.2838 | 0.0001 | 0.7689 |

**SSD Mobilenet v1 trained on BSTLD (all) with batch size 4**

| Evaluation Dataset | Weighted mAP | mAP | Off | Green | Yellow | Red |
|---|---|---|---|---|---|---|
| BSTLD | 0.5492 | 0.3822 | 0.0004 | 0.6491 | 0.4122 | 0.4674 |
| DTLD | 0.1199 | 0.0994 | 0.0003 | 0.2065 | 0.0441 | 0.1469 |
| FU | 0.2499 | 0.1957 | 0.0 | 0.2562 | 0.0488 | 0.2823 |
| FU (rotated) | 0.4433 | 0.3421 | 0.0 | 0.2573 | 0.0131 | 0.7558 |

**SSD Mobilenet v1 trained on BSTLD (filtered) with batch size 1**

| Evaluation Dataset | Weighted mAP | mAP | Off | Green | Yellow | Red |
|---|---|---|---|---|---|---|
| BSTLD | 0.1996 | 0.1054 | 0.0 | 0.3096 | 0.0001 | 0.1118 |
| DTLD | 0.0136 | 0.0111 | 0.0 | 0.0247 | 0.0018 | 0.0179 |
| FU | 0.1066 | 0.0911 | 0.0 | 0.0868 | 0.0039 | 0.1827 |
| FU (rotated) | 0.4091 | 0.3545 | 0.0 | 0.27 | 0.1901 | 0.6035 |

**SSD Mobilenet v1 trained on BSTLD (filtered) with batch size 4**

| Evaluation Dataset | Weighted mAP | mAP | Off | Green | Yellow | Red |
|---|---|---|---|---|---|---|
| BSTLD | 0.5684 | 0.3674 | 0.0001 | 0.666 | 0.3383 | 0.4653 |
| DTLD | 0.1036 | 0.0849 | 0.0002 | 0.1471 | 0.0405 | 0.1519 |
| FU | 0.1001 | 0.0656 | 0.0 | 0.127 | 0.0 | 0.0698 |
| FU (rotated) | 0.4727 | 0.3537 | 0.0 | 0.3274 | 0.0001 | 0.7335 |

**Faster RCNN Inception v2 trained on BSTLD (all) with batch size 1**

| Evaluation Dataset | Weighted mAP | mAP | Off | Green | Yellow | Red |
|---|---|---|---|---|---|---|
| BSTLD | 0.3368 | 0.2721 | 0.0004 | 0.4286 | 0.3109 | 0.3483 |
| DTLD | 0.4054 | 0.3696 | 0.0271 | 0.5964 | 0.383 | 0.472 |
| FU | 0.0082 | 0.0055 | 0.0 | 0.0166 | 0.0 | 0.0 |
| FU (rotated) | 0.0036 | 0.0047 | 0.0 | 0.0058 | 0.0046 | 0.0036 |

**Faster RCNN Inception v2 trained on BSTLD (all) with batch size 4**

| Evaluation Dataset | Weighted mAP | mAP | Off | Green | Yellow | Red |
|---|---|---|---|---|---|---|
| BSTLD | 0.4755 | 0.3276 | 0.006 | 0.498 | 0.303 | 0.5033 |
| DTLD | 0.0867 | 0.0637 | 0.0004 | 0.1417 | 0.0146 | 0.0983 |
| FU | 0.0053 | 0.0047 | 0.0 | 0.0126 | 0.0 | 0.0015 |
| FU (rotated) | 0.0353 | 0.0311 | 0.0 | 0.0188 | 0.0004 | 0.0742 |

**Faster RCNN Inception v2 trained on BSTLD (filtered) with batch size 1**

| Evaluation Dataset | Weighted mAP | mAP | Off | Green | Yellow | Red |
|---|---|---|---|---|---|---|
| BSTLD | 0.416 | 0.3021 | 0.0004 | 0.5103 | 0.3007 | 0.3971 |
| DTLD | 0.3968 | 0.3597 | 0.0207 | 0.5848 | 0.3777 | 0.4554 |
| FU | 0.0046 | 0.0045 | 0.0 | 0.0134 | 0.0 | 0.0 |
| FU (rotated) | 0.0009 | 0.001 | 0.0 | 0.0021 | 0.0 | 0.0008 |

**Faster RCNN Inception v2 trained on BSTLD (filtered) with batch size 4**

| Evaluation Dataset | Weighted mAP | mAP | Off | Green | Yellow | Red |
|---|---|---|---|---|---|---|
| BSTLD | 0.4849 | 0.3315 | 0.0005 | 0.5005 | 0.304 | 0.521 |
| DTLD | 0.0945 | 0.0692 | 0.0005 | 0.159 | 0.0143 | 0.1029 |
| FU | 0.0128 | 0.0137 | 0.0 | 0.0318 | 0.0 | 0.0092 |
| FU (rotated) | 0.1529 | 0.1131 | 0.0 | 0.0278 | 0.0001 | 0.3113 |

**SSD Mobilenet v1 trained on DTLD (all) with batch size 1**

| Evaluation Dataset | Weighted mAP | mAP | Off | Green | Yellow | Red |
|---|---|---|---|---|---|---|
| BSTLD | 0.321 | 0.1657 | 0.0 | 0.5191 | 0.0125 | 0.1314 |
| DTLD | 0.1266 | 0.1181 | 0.0002 | 0.2757 | 0.0965 | 0.0999 |
| FU | 0.0321 | 0.0379 | 0.0 | 0.0951 | 0.012 | 0.0066 |
| FU (rotated) | 0.3529 | 0.5185 | 0.0 | 0.3952 | 0.844 | 0.3165 |

**SSD Mobilenet v1 trained on DTLD (all) with batch size 4**

| Evaluation Dataset | Weighted mAP | mAP | Off | Green | Yellow | Red |
|---|---|---|---|---|---|---|
| BSTLD | 0.5517 | 0.3594 | 0.0 | 0.6519 | 0.3121 | 0.4736 |
| DTLD | 0.3496 | 0.3122 | 0.0159 | 0.5278 | 0.3136 | 0.3916 |
| FU | 0.0306 | 0.106 | 0.0 | 0.0258 | 0.2352 | 0.0571 |
| FU (rotated) | 0.3721 | 0.5298 | 0.0 | 0.187 | 0.8047 | 0.5977 |

**Faster RCNN Inception v2 trained on DTLD (all) with batch size 1**

| Evaluation Dataset | Weighted mAP | mAP | Off | Green | Yellow | Red |
|---|---|---|---|---|---|---|
| BSTLD | 0.3774 | 0.2946 | 0.0008 | 0.4307 | 0.3344 | 0.4125 |
| DTLD | 0.3963 | 0.3631 | 0.0258 | 0.5856 | 0.3763 | 0.4647 |
| FU | 0.0129 | 0.009 | 0.0 | 0.0269 | 0.0 | 0.0 |
| FU (rotated) | 0.0257 | 0.0252 | 0.0 | 0.0261 | 0.0208 | 0.0287 |

**Faster RCNN Inception v2 trained on DTLD (all) with batch size 4**

| Evaluation Dataset | Weighted mAP | mAP | Off | Green | Yellow | Red |
|---|---|---|---|---|---|---|
| BSTLD | 0.3835 | 0.2503 | 0.0002 | 0.4512 | 0.1565 | 0.3935 |
| DTLD | 0.4397 | 0.3947 | 0.0281 | 0.627 | 0.408 | 0.5158 |
| FU | 0.0059 | 0.0046 | 0.0 | 0.0139 | 0.0 | 0.0 |
| FU (rotated) | 0.0125 | 0.0107 | 0.0 | 0.0279 | 0.0029 | 0.0014 |