Active Learning for Object Detection

Hsuan Ling Chen
University of Toronto
Department of Aerospace Engineering
celene.chen@mail.utoronto.ca

***Abstract*—From computer vision to natural language processing, machine learning has helped solve many complicated human tasks over the past few years in many various applications. Both the collection and labeling of the data can be time-consuming and expensive to obtain. Moreover, not all data is equally useful for training the algorithms. Active learning is a machine learning algorithm that can achieve greater accuracy with fewer training labels if it is allowed to choose the data from which it learns. This paper aims to explore the application of active learning on object detection in order to reduce the data size and annotation effort and obtain an accurate object detector like the SSD, using only a fraction of the training images. The method proposed showed evidence for a successful active learning objective, outperforming the baseline method in both its final performance and data size reduction. The active learning method is able to reach a performance of 70 mPA with only about 60**

I. Introduction

From computer vision to the natural language process, deep learning neural networks have helped to solve complicated human tasks and have achieved great success over the past few years in many various applications. It is no secret that machine learning models, especially deep learning models, need lots of training data. The huge size of data in deep learning has always been a challenge in learning the underlying distribution and tackling various human tasks.

In the real world, autonomous vehicle systems are one of the most data-hungry AI applications. They need to be trained to recognize every object and environmental variable they have any chance of encountering in real-world scenarios. This includes paying attention to factors like differences in rules, road signs, area wildlife, unusual obstacles, and weather conditions. Large autonomous car developers, like Waymo, deploy fleets of test vehicles to capture data. Unfortunately, collecting all this data in-house using a limited fleet of test vehicles is not sufficient and can be redundant.[1] This is because test vehicles are typically operated in limited geographies such as a small group of cities. Obtaining enough structured and diversified data to make autonomous vehicles a reality remains an extraordinary challenge.

Moreover, even while the data is readily available, the sheer size of the data sets makes manual labeling impractical. There are situations in which unlabeled data is abundant but manually labeling is expensive. For example, in order to obtain the ground-truth training set for the neural network, in image classification, the human annotator will have to identify and group the images according to their correct classes. In object detection, the annotator will not only have to identify and label the objects of interest in the image, but also to generate an accurate bounding box for them. In semantic segmentation, the labeling cost is even higher due to assignments of the image at pixel-level. The immense time and manual effort to annotate the images can become extremely laborious and expensive in the long run. Furthermore, not all training images are the same, some are more useful and informative than others. [1][2]

For these reasons, an effective way to deal with large data is data selection identifying the most representative training samples that aim at improving data efficiency of machine learning techniques. This type of iterative supervised learning is called active learning. Since the learner chooses the examples, the number of examples to learn a concept can often be much lower than the number required in normal supervised learning. With this approach, there is a risk that the algorithm is overwhelmed by uninformative examples.

This paper will focus specifically on the use of active learning methods to show that if a learning algorithm can choose the data it wants to learn from, it can perform better than traditional methods with substantially less data for training for object detection and its theoretical usefulness in Autonomous Vehicle application.

A. Background

In perception for robotics and computer vision, the most commonly categorized recognition tasks are: image classification, object localization, object detection, semantic segmentation, and instance segmentation. [3] This project focuses on object detection which usually involves the identification and localization of multiple objects from an image, creating a bounding box for each instance of each object category of interest.

B. Object Detection

Object detection is the task of determining where objects are located and to what class each object belongs to in a given image. A detection is formalized as Di = ci,bi, where ci is a probability distribution over the classes and bi are the bounding box coordinates. There have been many successful approaches to object detection. For example, the sliding window approach, also known as exhaustive search, operates where an entire image is scanned using sliding windows at multiple scales. The more recent methods such as the Convolutional Neural Network (CNN) based classifiers further decrease the classifiable object windows drastically by generating a smaller amount of well chosen bounding boxes and refining those, allowing the use of computationally more expensive classifiers. YOLO is a single neural network trained end to end that takes an image as input and predicts bounding boxes and class labels for each bounding box directly at real-time detection speed. This technique, however, offers lower predictive accuracy. [4]

The Single Shot multibox Detector (SSD) [3] has the advantages of all of these previous methods mentioned, achieving a high-accuracy at real-time detection speed. The standard SSD300 framework uses a VGG-16 base network pre-trained on ImageNet with an extra 4 convolutional layers without the fully-connected layers. Unlike YOLO's fixed grid cells, SSD utilizes default boxes applied to feature maps at different scales to help handle different object scales using convolutional filters of different network layers. SSD's bounding boxes are class-agnostic and each box has a class probability distribution calculated. The predictions are obtained from the greedy NMS if the class probability is higher than some confidence threshold. SSD is the object detection framework used in this project, it will be treated in greater detail in later sections.

1) Metrics: The mean average precision (mAP) is the most commonly used evaluation metric for object detection. First, the localization performance of the object detector is evaluated by the set of pixels in the predicted bounding box A must be compared with the set of pixels in the ground truth bounding box G. [5] This is accomplished by comparing the IoU of the sets of pixels of the two bounding boxes:

$$IoU(A, G) = |AG|/|AG|$$

The value of this metric will be larger when the bounding boxes have big overlap, and smaller when there are pixels that do not overlap. The higher this metric value is, the more it corresponds to the ground truth. An IoU of 1 means that the prediction is exactly the same as the ground truth. Following the Pascal VOC rules, if a single ground truth overlaps with multiple predictions, the prediction with the highest IoU is considered a correct prediction (TP) and all others are false predictions (FP). The accumulative TPs and FPs are then used to calculate the precision and the recall: Precision = TP/TP+FP, Recall = TP/TP+FN. The average precision(AP) is essentially calculated by the integration of the precision recall curve. [5]

Finally, the mean Average Precision (mAP) can be calculated by taking the arithmetic mean of the APs over all classes:

$$mAP = 1/class \sum(AP)$$

C. Active Learning

Active learning is a machine learning algorithm that can make the process of labeling new data more efficient by selecting unlabeled samples which, when labeled, are expected to improve the model the most. This way, the model is able to learn more with a relatively small sample size. Its motivation comes from many modern machine learning problems such as the autonomous vehicles, where unlabeled data may be abundant, but labeling is difficult and expensive to obtain. [6]

Active learning works in an iterative manner, starting on a small set of preliminary labeled data, the model is trained and makes predictions on a small group of samples in the unlabeled pool. Then the sample selection algorithm selects the samples that it deems the most informative based on the query strategy. These selected samples are then sent to a human annotator, also called an oracle, who will then append the newly labeled data to the training model to be retrained in the next iteration. This model is then again used to make predictions on the, now slightly smaller, pool of unannotated data. After each iteration, the model's accuracy will incrementally improve, and can be stopped once it reaches a good enough performance. This process is illustrated in Figure 1. More details on active learning will be discussed in later sections of the project. [6] [7]

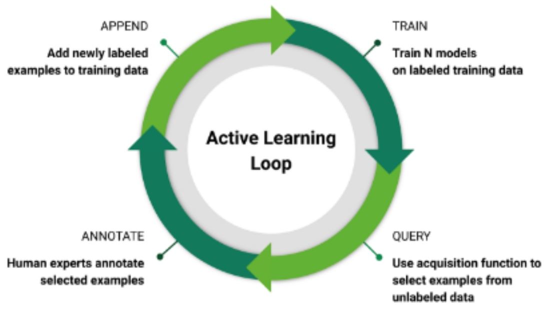

APPEND
Add newly labeled examples to training data
TRAIN
Train N models on labeled training data
Active Learning Loop
ANNOTATE
Human experts annotate selected examples
QUERY
Use acquisition function to select examples from unlabeled data

Fig. 1: Active Learning Loop [1]

1) Scenarios: In active learning, there are typically three settings when the learner will query the labels of instances. The first scenario is the membership query synthesis where the learning algorithm can request labels for any unlabeled instance in the input space, including those it generates itself. In stream-based active learning scenario, it is assumed that obtaining an unannotated sample has no cost and typically the instances are shown one at a time and the learning algorithm must decide each time whether to request its label or to discard it, typically used in online learning settings. Lastly, in the pool-based active learning scenario, in which there is a small initial set of annotation and a large pool of unannotated data. The pool of unannotated data is evaluated entirely and ranked in order to determine the best samples to be annotated. It is also the most common scenario in the active learning community, and hence explored in this project. [8]

II. Methodology

In this section, the method for object detection is briefly discussed, then the method to adopt active learning with object detection is proposed.

A. Object Detection Method

In this project, all experiments are performed using the Single Shot MultiBox Detector (SSD) as object detectors. Since the objective of this project is to study the effect of the sample selection process for object detection and not on object detection itself, existing detection frameworks can be used. The open source code for SSD [9] research was used and followed as is with only slight alterations in the training regime and benchmarking. The SSD is among the best performing object detection architectures in terms of its training speed and mAP performance on the Pascal VOC 2007 and 2012 datasets. [9] The SSD300 standard is selected for the initial model setup, which has an original set of 300 iterations using the SGD optimizer. The fast and accurate detection by SSD300 can help to guarantee a satisfied reference performance and maintain an acceptable computation cost after the implementation of active learning. All other hyperparameters involved including data augmentation are all kept unchanged during the rest of the implementation in the project.

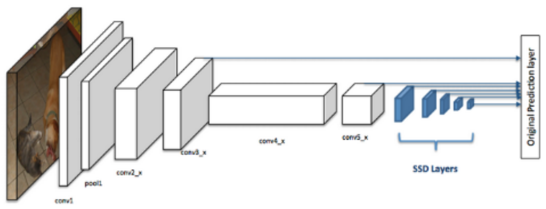


Fig. 2: SSD Architecture [3]

B. Active Learning Method

This section proposes how the datasets will be divided and the sampling strategies for active learning. Figure 3 explains the overview process of the active learning method. Similar to the earlier explanation of the active learning process in earlier sections, the required steps are performed to build the active learning system. Since this project is just an experimentation on active learning and readily available labeled datasets will be used. First the training sets are split into subsets, the model trains on the first labeled subsets and evaluates its own performance on the validation set. Record this performance and then test the model on the next training subset which are already labeled. The labels are evaluated and corrected, then depending on the query strategy selected, create a new training set using the selected samples. Repeat and iterate by training the model on these new subset until the model reaches a desired performance.

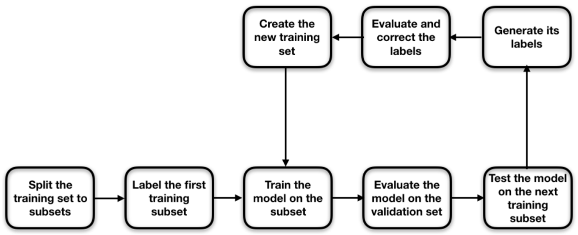

Create the new training set
Evaluate and correct the labels
Generate its labels
Split the training set to subsets
Label the first training subset
Train the model on the subset
Evaluate the model on the validation set
Test the model on the next training subset

Fig. 3: Active Learning Process [10]

1) Dataset Preparation: To prepare the dataset for training the model, there are first the training and validation sets split from the original dataset. There needs to be a large enough validation set in order to keep track of the model's performance metrics after each training process since the training set is being constantly updated. The training sets would also need to be big enough to be split into multiple subsets which will be used iteratively to retrain the model.

In active learning, the first training subset is labeled with the objects that need to be detected. After each training on the training subset, the performance of the model needs to be validated on the validation set and its performance metrics will be stored. It is technically not necessary to have a test set in active learning applications because the subsets of the training sets are going to take the role of testing sets each time. The labels generated by previously trained models are evaluated and corrected with their correct labels. Once a new training set is created, combine the old training subsets with the new ones to train the model. However, to have a fair comparison between the baseline and the active learning method, the project will use a common test set to evaluate each model's performance.

2) Query Strategy: The key component in a pool-based active learning framework is that the model evaluates the class probabilities of the unlabeled samples, and selects the ones that it determines as informative to be labeled for the next training iteration. This selection process is referred to as the sample selection strategy or query strategy. These sampling strategies can be either based on the confidence of the model on detecting objects in the image sample or based on the underperformance of the model on a certain class which assumes that the model m is required to be a probabilistic learner that outputs prediction probabilities : $m(x) = (\mathbb{P}(y_i|x))_{1\leq i\leq n}$ for n classes. Here are three of the most regularly used active learning strategies that can be easily used with neural networks since their last layer has a softmax activation, representing class-probabilities. [11]

The first strategy is called the Least Confidence (LC), where in this method, the average of the classification uncertainties of the predicted bounding boxes is computed and the images with high mean classification uncertainty are chosen for querying. This means that the learner selects the instance for which it has the least confidence in, if the confidence is lower than a certain threshold, the sample is sent off for labeling and used for the next iteration training as the new training subsets. The expression for the LC method to return the samples least confident samples is:

$$X_L C = \underset{x}{\operatorname{argmax}} 1 - \mathbb{P}(\hat{y}|x)$$

Where x represents a sample , the model predicts $\hat{y}$ with probability $\underset{y}{\operatorname{argmax}} \mathbb{P}(y|x)$.

However, notice that using the least confidence method neglects all the information about other class probabilities. Sometimes the original training set can cause underperformance of a certain type of classes due to class imbalance of the dataset or due to some of the classes having many similarities making it difficult for the model to distinguish them in the samples. To account for this is to use the margin between the predicted class $\hat{y}_1$ and the second top prediction $\hat{y}_2$, this incorporating the posterior of the second most likely label by calculating the average, maximum and sum of the predicted bounding boxes and the learner selects the instance that has the smallest difference between the first and second most probable labels. This is called margin sampling strategy.

$$X_M = \underset{x}{\text{argmin}} \mathbb{P}(\hat{y}_1|x) - \mathbb{P}(\hat{y}_2|x)$$

However, in order to optimize the use of all the label probabilities, the entropy sampling strategy applies an entropy formula to each instance and returns the instance with the largest entropy for labeling. Hence this method is different from the other two in a way that it would not return an instance where a label is very unlikely.

$$X_E = \underset{x}{\text{argmax}} - \sum_y \mathbb{P}(y|x) \log \mathbb{P}(y|x)$$

It can be seen from these common query strategies that it is better to use entropy-based sampling if the objective is to reduce loss. [12] It is better to use margin sampling when trying to reduce classification error. When applied with active learning, these sampling methods ensure that in future appearances of any of the previously mentioned cases, the sample is saved with a label in the new training subset, evaluated and then is used to train the model to ensure better performance on similar future appearances. [11]

3) Evaluation: For the evaluation, the mean average precision (mAP) will be evaluated and validated after each selected amount of sample subsets.

III. Implementation and Analysis

As seen from the previous section, there is a lot to account for when applying active learning. The dataset will need to be divided and the samples need to be queried. It is also important to keep track of the performance of the model for each training since the model is constantly updating and introducing new data. Hence before diving into Object Detection, a small demonstration test build is made on the image classification task to verify active learning method's practicability and test the implementation algorithm.

A. Proof of Concept using Object Classification

The test setup build is made on the image classification task on the simple MNIST dataset which contains 60000 samples. First, the initial baseline model is set up without a query strategy where it will just randomly select a subset of 500 images from the training set for each iteration of the active learning model to evaluate on. This gives a sense for how quickly the model trains for each iteration without active learning sampling selection. At random selection, it took about 12000 samples for the model to reach 95% accuracy. For each of the other query strategies, Least Confidence, Least Margin and Entropy methods discussed in the Query Strategy section. Starting off with 500 labeled data, then based on the selected uncertainty sampling, the model evaluates and chooses the next 500 samples that it predicts are the most productive to train on. The performance of the model after each sample increment is monitored to reach a certain performance. The MNIST dataset is very easy to train on and is able to achieve really high accuracy results. Here in this experiment, the benchmark accuracy is set as 95%. Table 1 shows the results from the four models trained on MNIST dataset. It can be seen that all three active learning sampling methods significantly reduce the size of data required to train the model while obtaining the same performance accuracy. It also took significantly less time for the models to train as less dataset was needed to reach the performance goal. Here, only 3500 samples were needed to train the model with 95% accuracy using the Least Margin method.

TABLE I: Query Strategies on MNIST

Method	Sample size when accuracy reached 95%
Random(Baseline)	12000
Least Confidence	4000
Highest Entropy	4500
Least Margin	3400

Figure 4 shows the accuracy plot for the training models with different query strategies. It demonstrates the example that shows the implementation of active learning can not only reduce the data size by training on only the most informative samples, it can also improve the overall model performance.

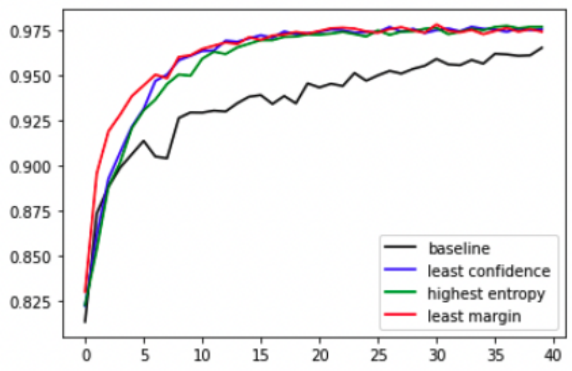


Fig. 4: Performance of Query Strategies on MNIST dataset

B. Datasets Setup

The object detection of this project is assessed on PASCAL VOC 2007 and 2012 datasets which consists of bounding boxes of 20 object categories. These two datasets were said to work well with the SSD detector. The training and validation sets of VOC 2007 and 2012 are combined to create a larger training and validation set. The joining of datasets created a training set of size 8218 and validation set of size 8333. Finally, the VOC 2007 test set of size 4952 is used to test the model's performance. Here the training set will be regarded as the unlabeled pool of data.

C. Object Detector Setup

Similar to the image classification task on the MNIST dataset, an initial baseline model is required to provide a sense of how well the model is performing at random sampling. Since class scores are available, using the classification metric on the object detection is straightforward. Using the open source code for SSD, before the experimental run, the datasets are divided randomly into batches of 5% of the total training set. This ensures some diversity within each batch to prevent very similar images from being selected for the same batch. The model is being evaluated for its performance with the training set for everytime a new batch is trained. The results for each of these evaluations are recorded. This assignment simulates the regular training process while keeping track of the model performance. 5% data size is a reasonable size due to trade-off between tight feedback loop and computational efficiency.

D. Active Learning Setup

The SSD object detection network is capable of accessing the class scores, hence using the classification metric on the object detection is straightforward. The proposed implementation is inspired by the results of the classification of the MNIST experiment in the earlier section. Since the Least Margin produced the best results in data size reduction in the previous section, this query is selected to be applied on the active learning object detection model.

The idea behind the implementation is that if a high-scoring bounding box is discovered by a convolution layer and overlaps with a ground truth, then the other convolution layers will also tend to have high scoring bounding boxes in the region since the loss criterion is defined as the spatial overlap with the ground truth bounding box. Using this fact, the least margin concept can be implemented within the algorithm. For each of the bounding boxes discovered, its margin is calculated with all of the other neighbouring boxes using the difference of their confidence scores. The higher this value is, it means there are more disagreements between the convolution layers. As the class probabilities are included in the calculation of image margin, this emphasises the classifier disagreements of important classes. The corresponding bounding box with the least margin with the current bounding box is the closest disagreements between convolution layers, therefore least confident, and hence, will be selected as the queried samples for active learning. [4]

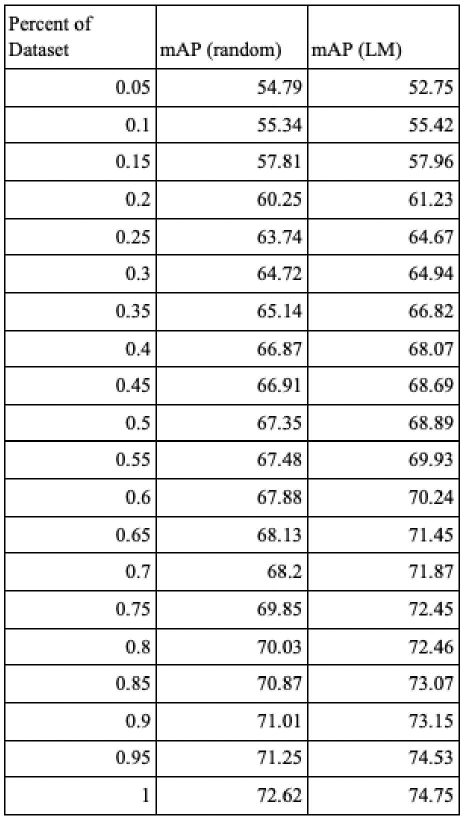
Algorithm 1 Active Learning Implementation

```
Data: SSD Model, unlabeled image pool U, number of classes
      C, batch size k
Result: New training subset
for each image $i$ in $U$ do
    //SSD gets bounding boxes $B$ for each class $c$ after nms;
    $B$ = getBoundindBoxesPerClassfromSSD();
    for each class $c$ in $C$ do
        for each bounding box $b$ in $B$ do
            source = getConvolutionLayer();
            for each layer $l$ do
                if $l == source$ then
                    continue;
                end
                findBoundingBoxInLayer();
                margin[l] = $\underset{x}{\operatorname{argmin}} \mathbb{P}(\hat{y}_1|x) - \mathbb{P}(\hat{y}_2|x)$
end
```

E. Results

On the original SSD paper, the performance for the baseline model reached 77.2 mPA. However, in this experiment, the baseline model without active learning achieved a 72.62 mPA final performance using all of the training sets. The final performance of the object detection with active learning implementation achieved a final result of 74.75 mPA. This section displays the quantitative results from the project implementation.

TABLE II: Implementation Results



Percent of Dataset	mAP (random)	mAP (LM)
0.05	54.79	52.75
0.1	55.34	55.42
0.15	57.81	57.96
0.2	60.25	61.23
0.25	63.74	64.67
0.3	64.72	64.94
0.35	65.14	66.82
0.4	66.87	68.07
0.45	66.91	68.69
0.5	67.35	68.89
0.55	67.48	69.93
0.6	67.88	70.24
0.65	68.13	71.45
0.7	68.2	71.87
0.75	69.85	72.45
0.8	70.03	72.46
0.85	70.87	73.07
0.9	71.01	73.15
0.95	71.25	74.53
1	72.62	74.75

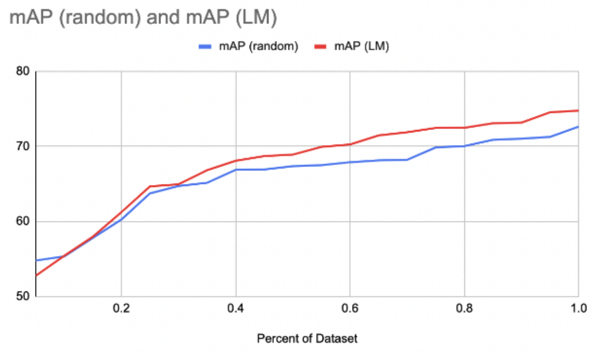


Fig. 5: Performance of Least Margin on Object Detection

F. Comparison and Discussion

The results of the application of Least Margin Sampling and the original results are compared. Figure 5 shows the mAP on the test set plotted against the number of labeled samples for models trained with and without active learning on the PASCAL VOC. Through the history of the iterations, it can be seen that training with active learning outperforms the baseline model as early as only 15% of the data set was used for training. But before 15% data size was used, both methods did not have a big difference in performance with one another. This could be because the samples in the dataset vary greatly and there are many images with high similarity. After 15% data size, the active learning model performs steadily at a better mAP than the baseline model.

From the evaluation table in Table 2, it can be seen that while it took the baseline model (random sampling) about 80% data size to reach the performance of 70 mPA, while it only took the Least Margin sampling method 60% data size to achieve the same performance.Therefore, active learning is capable of reducing dataset size while maintaining a comparable performance in object detection through only training the model on more useful and informative data.

However, it is necessary to recall that in this project, the use of readily available labeled dataset is involved. In reality, the model would need to wait for the annotators to label these samples. This means even though the data size required to reach the same performance becomes smaller, the implementation time of active learning and the training time can get more expensive.

IV. Conclusion

In conclusion, it can be seen from this project that active learning is an effective way to reduce data size requirements to train a model while maintaining comparable performance. Furthermore, the quality of data can also affect the performance of the machine learning model and shows that more data is not always better. If the active learning model is able to select the more helpful data earlier in the training, and the least helpful ones at the end or not trained at all, it can outperform the original passive model that had access to the entire dataset from start. On the contrary, outliers and samples that are not as informative may lead to a bad model performance since there is a risk that the algorithm is overwhelmed by uninformative examples. For this project, only the least margin sampling was explored and tested, there are still many other ways to query the samples aside from the ones discussed in this paper. There are also even more challenges in real-world object detection applications such as class imbalance [11] among the selected samples across each active learning iterations and events where the number of classes is unknown or absent of representative samples. This paper proved that active learning is effective in the application of object detection. More future work includes studying this method in the other domains of object detection.

[5] [13]

References

[1] Elmar Haussmann et al. "Scalable active learning for object detection". In: *2020 IEEE intelligent vehicles symposium (iv)*. IEEE. 2020, pp. 1430–1435.

[2] Hamed H Aghdam et al. "Active learning for deep detection neural networks". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 3672–3680.

[3] Jason Brownlee. Jan. 2021. URL: https : / / machinelearningmastery . com / object - recognition - with-deep-learning/.

[4] Soumya Roy, Asim Unmesh, and Vinay P Namboodiri. "Deep active learning for object detection." In: *BMVC*. 2018, p. 91.

[5] Jasper Bakker. "Active Learning for Object Detection With Localization Uncertainty from Sampling-Based Probabilistic Bounding Boxes". In: 2019.

[6] Ying Li et al. "Deep active learning for object detection". In: *Information Sciences* 579 (2021), pp. 418–433.

[7] Burr Settles. "Active learning literature survey". In: (2009).

[8] Clemens-Alexander Brust, Christoph Käding, and Joachim Denzler. "Active learning for deep object detection". In: *arXiv preprint arXiv:1809.09875* (2018).

[9] Wei Liu et al. "Ssd: Single shot multibox detector". In: *European conference on computer vision*. Springer. 2016, pp. 21–37.

[10] Moustafa Ayoub. Jan. 2020. URL: https://medium.com/ixorthink/how-to-setup-an-active-learning-framework-for-your-object-detection-model-ae46ef502f5f.

[11] Soumya Roy, Vinay P Namboodiri, and Arijit Biswas. "Active learning with version spaces for object detection". In: *arXiv preprint arXiv:1611.07285* (2016).

[12] In So Kweon Donggeun Yoo. "Learning Loss for Active Learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 93–102.

[13] Aibo Tian and Matthew Lease. "Active learning to maximize accuracy vs. effort in interactive information retrieval". In: *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. 2011, pp. 145–154.