Active Learning for Deep Object Detection

by

Tze Fung Christopher Chan
Supervisor: Professor Steven L. Waslander

April, 2020

B.A.Sc. Thesis

Division of Engineering Science
UNIVERSITY OF TORONTO

Abstract

Active learning is a subfield in machine learning in which a model will iteratively seek the most informative samples to include in its training dataset. Advances in this subfield has the potential to save millions of hours in data labelling costs for today's data hungry deep learning models. This thesis aims to explore the use of active learning in object detection tasks; a largely unexplored application for active learning.

Through our studies with the VOC2007 and MNIST datasets we have made three key contributions to the field. Firstly, we benchmarked the effectiveness of recent active learning techniques applied to object detection. Secondly, we provided an in-depth analysis on the type of data favored by each of these active learning techniques. Finally, we highlight the importance of re-initializing network weights after acquiring new data in an active learning scenario.

Acknowledgements

I would like to thank my supervisor, Professor Steven L. Waslander for supervising me for my undergraduate thesis and providing me with very insightful feedback. I would also like to thank John Willes for helping me setup remote access to TRAIL Lab's resources.

Contents

Nomenclature

D	Dataset
X	Dataset inputs
Y	Dataset output/labels
ENT	Entropy
LS	Localization Stability
MS	Margin Sampling
MSTD	Mean Standard Deviation
MSTDB	Mean Standard Deviation with Bounding Boxes
QBC	Query-by-committee
VR	Var Ratio

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

Deep learning has become the de-facto standard for many object detection tasks on 2D images. However, drawing individual bounding boxes in an image can be a tedious process and these methods often require large amounts of labelled training data [1].

To reduce labelling efforts for object detectors, researchers have mainly focused on two strategies [2]. The first strategy is weakly supervised learning; it involves building a model that will generate bounding boxes for the user to label. Examples of weak supervision include yes/no verification [3], center-clicking schemes [4] and image-level labelling [5]. The second strategy is active learning; the model will determine which of the unlabelled data will provide the biggest improvement to its model parameters [6]. Past attempts include [7], [8] [9]. While the first strategy can drastically reduce labelling time, it can often lead to lower quality detectors. Thus, we focus our efforts on active learning as it will likely produce higher quality detectors and potentially help us gain insights into how we can improve our unsupervised models.

1.2 Research Problem

There have been many works that employ active learning for image classification [10] [11] [12], segmentation [13] [14], but very few works that focus on object detection [7], [8] [9]. To the best of our knowledge, there has not been a comparison between recent active learning strategies for object detection. Furthermore, many of the active learning methods for image recognition tasks seem to suffer from a similar training bottleneck: to prevent convergence at a local minima, model weights are re-initialized after every query for new labelled data [7] [15] [11]. It should also be noted that some authors did not re-initialize their model weights after acquisition of new images in their experiments [9] [8].

1.3 Objective

This thesis has two objectives. The first objective is to explore and compare recent advances in active learning for object detection in 2D images. In doing so, we hope to better understand why certain active learning metrics perform better than others. The second objective is to explore the impact of network weight re-initialization in active learning scenarios. In particular, we would like to verify a claim made by Gal et al. in which they suggested that resetting model weights upon acquisition can prevent local minima [11].

Chapter 2

Background

2.1 Deep Learning for Object Detection

A 2D object detection algorithm takes an image as an input and outputs a set of rectangular bounding boxes that outline the scale, location and type of object in the image. The rising popularity of object detection can be attributed to the recent advancements in convolutional neural networks (CNNs) [16] as well as the plethora of annotated object detection data that has been released in the past few years MSCOCO [17], PASCAL VOC [18], KITTI [19], etc.

An in-depth surveys on modern object dectectors can be found in [20]. State-of-the-art deep object detectors can be categorized into two categories: two-stage detectors and one-stage detectors [7]. In two-stage detectors, region proposals are generated from an image, CNN features are then extracted from these regions and a final classifier is used to determine the category of the proposal. On the other hand, one-stage detectors directly output a set of object location and classes without an intermediate proposal stage. One-stage detectors such as YOLOv3 [21] and SSD [22] tend to have less complex network architectures. As such, one-stage detectors tend to train faster and have lower inference times at the cost of lower accuracy.

We will focus on experimenting with one-stage detectors for our experiments as this will enable faster experimentation and prototyping. In particular, we have chosen to use SSD. This was also the network of choice for several previous works on active learning for object detection [7] [9].

2.2 Active Learning for Object Detection

Active learning is a subfield in machine learning in which the algorithm is able to query an oracle to label certain unlabelled instances [6]. The key idea is: if a machine can choose the data it learns from, then the number of training instances required for it to learn will be much less than if a human were to choose the data it should learn from. The typical workflow of an active learning algorithm is illustrated in Figure 2.1. We typically begin training our model with a small set of instances in our labelled training set $\mathcal{L}$. Once the model has learned from $\mathcal{L}$, it will query from the unlabelled pool $\mathcal{U}$ for instances based on a

predefined acquisition function and ask an oracle to label these instances. This cycle repeats itself until the model has achieved a satisfactory performance. A detailed survey of active learning can be found in [6].

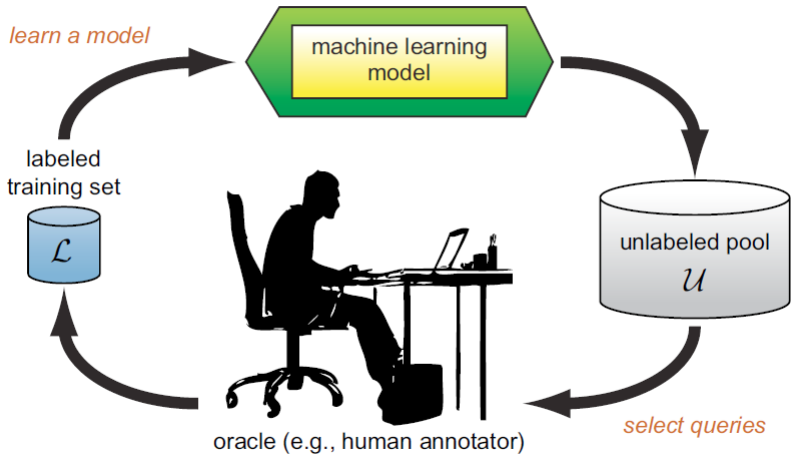


Figure 2.1: Pool Based Active Learning [6]

Past attempts at applying active learning for deep object detectors have been heavily based on uncertainty sampling and query-by-committee methods. Uncertainty sampling involves labelling the samples that our model is least certain of [6]. MC-Dropout [15], margin-sampling [8] and localization stability [7] are examples of uncertainty sampling applied to active learning in object detection. Query-by-commitee methods seek to minimize the version space of the model [6]. These methods assume that there is a committee of models that represent the various version spaces of the data. The instances with the greatest disagreement between the various models are selected for querying. Roy et al. applied this method to object detectors and found an improvement in results over black-box methods such as entropy [9].

2.2.1 Network Uncertainty Methods

Recent findings that suggest softmax produces over-confident predictions for unseen data instances [23] [24] have prompted a surge of research interest into quantifying network uncertainty [23] [11] [15]. Two popular techniques for quantifying network uncertainty are MC-Dropout [11] and Deep Ensembles [23].

MC-Dropout extracts the predictive uncertainty of a network by performing multiple forward passes at test time with dropout being applied. For a test point x, the network will perform T inferences with dropout applied as per Equation 2.1, where the network weights for the t-th inference is given by W_t. Contrary to MC-Dropout, Deep Ensembles will take the mean of the softmax output from an ensemble of networks. Both methods apply an element of stochasticity to the softmax output of a network to generate an uncertainty estimate.

$$p(y|x) \approx \frac{1}{T}\sum_{t=1}^{T} softmax_{W_t}(x) \tag{2.1}$$

Entropy, BALD, var ratio and mean STD are all examples of acquisition functions that can utilize the uncertainty of a network [11]. Gal et al. applied MC-Dropout in an image classification setting and found a significant improvement in the network's ability to classify images over a random acquisition function [11]. Feng et al. used MC-Dropout [25] and Deep Ensembles [23] to estimate the predictive probability of their object detectors in an active learning setting [15] and their experiments showed that using entropy as an estimate for the model's predictive uncertainty resulted in significant savings in the amount of data that needed to be labelled. Although Feng et al. experimented with a 3D object detector, their methods can easily be extended to 2D object detectors.

Seeing that Feng et al.'s experiments indicate that MC-Dropout and Deep Ensembles have comparable performance [15], we will only be experimenting with MC-Dropout in this thesis to compute entropy, BALD, var ratio and mean STD.

The sections below describe how to apply network uncertainty to active learning in a classification setting. We can easily extend each of these methods to an object detection setting by computing the entropy, BALD, var ratio or mean STD for each bounding box and then aggregating them by using the maximum, average or sum of the scores among all bounding boxes.

Entropy

The predictive entropy of a classifier can be calculated using Equation 2.2 [11], in which we sum the scaled log probability of the output belonging to each class. In the context of deep learning classifier, this is typically calculated at the softmax output layer of a deep network. We typically take the samples with the largest entropy for querying.

$$H[y|x, D_{train}] := -\sum_{c} p(y = c|x, D_{train}) \log p(y = c|x, D_{train}) \tag{2.2}$$

The MC-Dropout approximation for entropy computation is given by Equation 2.3 with $\hat{p}_c^t$ representing the probability of class c for inference t.

$$H[y|x, D_{train}] \approx -\sum_{c} (\frac{1}{T}\sum_{t} \hat{p}_c^t) \log (\frac{1}{T}\sum_{t} \hat{p}_c^t) \tag{2.3}$$

BALD

A detailed derivation for the BALD ratio can be found in [11]. Given model weights ω, we can approximate the BALD ratio using Equation 2.4 [11]. This metric is similar to entropy, except we include an additional term that takes into account the expected entropy of the system. The first term characterizes the uncertainty of the model, while the second term measures the uncertainty given the a certain draw of the model parameters. Our BALD score is high when the we have high entropy (model is uncertain) and the expectation of sample entropy is high (per sample uncertainty is high).

$$I[y, \omega | x, D_{train}] = H[y|x, D_{train}] - E_{p(\omega|D_{train})}[H[y|x, D_{train}]] \\ \approx -\sum_c (\frac{1}{T}\sum_t \hat{p}_c^t) \log(\frac{1}{T}\sum_t \hat{p}_c^t) + \frac{1}{T}\sum_{c,t} \hat{p}_c^t \log \hat{p}_c^t \tag{2.4}$$

Var Ratio

The var ratio measures a lack of confidence in the softmax predictions [11]. We seek to label instances with the largest var ratio: the most confident class in bounding box has a low priority.

$$var\_ratio[x] := 1 - \max_y p(y|x, D_{train}) \tag{2.5}$$

Mean STD

The mean standard deviation metric simply computes the variance of the probabilities across the T inferences and averages the variances among all classes [11].

2.2.2 Margin Sampling

Margin sampling (also known as 1v2 sampling) seeks out unlabelled data that will help the model refine it's decision boundary the best. The margin for a bounding box is given by Equation 2.6. Intuitively, this metric tries to find the samples where the difference between the top two class probabilities in a bounding box is minimized.

$$v_{1vs2}(x) = (1 - (\max_{c_1 \in K} \hat{p}(c_1 \mid x) - \max_{c_2 \in K \setminus c_1} \hat{p}(c_2 \mid x)))^2 \tag{2.6}$$

Brust et al. experimented with using the average, sum and maximum as a means of aggregating the bounding box margins for each image. They found that the three aforementioned aggregation techniques performed better than the baseline of random acquisition [8]. In our experiments, we will be using using the maximum margin among all bounding boxes in an image as our metric.

2.2.3 Localization Stability

All of the aforementioned acquisition functions deal with the classification aspect of object detection but do not quantify the quality of the object-location hypothesis. Localization stability serves to fill this void. Localization stability measures how sensitive to image noise the detected bounding boxes are [7]. By measuring the overlap between the bounding box in the original image and the bounding box in the noisy images, we can determine our model's confidence.

Figure 2.2 illustrates the process of localization stability. We first run the image through a detector without any noise and then we run N noisy images with varying noise levels through

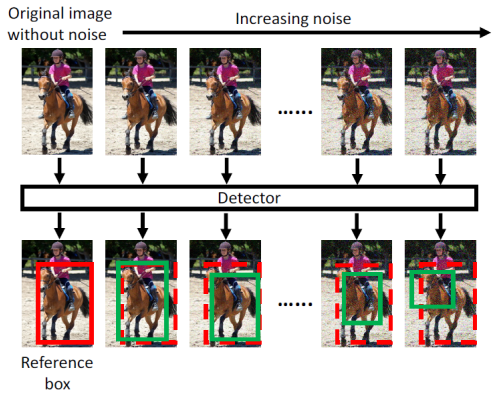


Figure 2.2: Localization Stability [7]. The red bounding boxes are the reference bounding boxes and the green bounding boxes represent the detectors predictions in the noisy image.

our detector. In the author's experiments, they chose N=6 and the standard deviations of each level were {8, 16, 24, 32, 40, 48}.

Let B_0^j represent the j-th bounding box in the image, let $C_n(B_0^j)$ represent the bounding box computed at noise level n, then the localization stability, $S_B(B_0^j)$, of this bounding box is given by Equation 2.7.

$$S_B(B_0^j) = \frac{\sum_{n=1}^{N} IoU(B_0^j, C_n(B_0^j))}{N} \tag{2.7}$$

We aggregate the localization stability for every bounding box to compute the localization stability of an unlabelled image I_i. Let M be the number of bounding boxes in an image, let each bounding box be weighted by the maximum class probability of that box, then the localization stability of an image will be given by $S_I(I_i)$ as per Equation 2.8.

$$S_I(I_i) = \frac{\sum_{j=1}^{M} P_{max}(B_0^j) S_B(B_0^j)}{\sum_{j=1}^{M} P_{max}(B_0^j)} \tag{2.8}$$

In the experiments of Kao et al., combining localization stability and classification uncertainty to quantify network uncertainty produced the best results [7]. The classification uncertainty of an image is given by Equation 2.9. It is the maximum difference between the most confident class of all bounding boxes and 1. We will select images with high classification uncertainty and low localization stability for annotation: $U_C(I_i) - S_I(I_i)$.

$$U_C(I_i) = \max_j (1 - P_{max}(B_0^j)) \tag{2.9}$$

2.2.4 Query-By-Committee

Roy et al.'s query-by-commitee (QBC) method can be illustrated using Figure 2.3. Imagine a scenario where our network outputs three bounding boxes around the car. A model with a large version space would mean that all three bounding boxes have a dissimilar softmax probabilities. QBC clusters overlapping bounding boxes of the same class and tries to

look for images with a large version space among its detections to be labelled in order to minimize the version space of the model

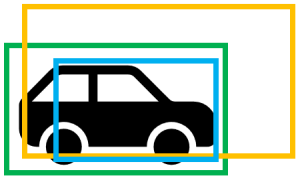


Figure 2.3: Sample bounding box detection on object.

SSD does object detection at multiple scales using its various convolutional layers and QBC uses these convolutional layers to form a committee a classifiers. Similar bounding boxes are clustered together and the difference in softmax score between the two most probable bounding boxes in the same cluster is used as a metric in our query. The full algorithm developed by Roy et al. is reproduced in Algorithm 1 [9].

Algorithm 1: Query-By-Committee Algorithm

input : SSD model, batch size k, unannotated image pool $\mathcal{U}$, set of classes C, overlap threshold j
output: active selection from $\mathcal{U}$

for *each image i in* $\mathcal{U}$ **do**
  apply SSD on i and acquire bounding boxes B_c for each class c after non-max suppression.
  for *each class c in C* **do**
    for *each bounding box b in* B_c **do**
      `// discard overconfident or unconfident predictions`
      if *prob(b) <= 0.1 or prob(b) >= 0.9* **then**
        continue
      find convolution layer that generated it and call it s
      $a_l = \{\}$
      for *each convolutional layer l other than s* **do**
        find bounding boxes that have more than j jaccard overlap with b and add them to A.
        add maximum softmax probability of auxiliary bounding box to a_l
      secondmax = max value in A
      box_margin[b] = prob(b) - secondmax
    class_margin[c] = $\frac{\sum_{b \in B_c} box\_margin[b]}{|B_c|}$
    class_confidence[c] = $\max_{b \in B_c} prob(b)$
  image_margin[i] = $\sum_{c \in C} \frac{class\_margin[c]}{\sum_{c_1 \in C} class\_margin[c_1]} \times class\_margin[c]$
  select k images with largest image margins

2.3 Summary of Acquisition Functions

From our literature review, we can categorize each acquisition function for object detection in two different ways. Firstly, does the acquisition function seek to measure the confidence of the network in it's detection or the consistency in it's detection? Confidence is typically measured by looking at the maximum softmax probability of a bounding box or measuring the spread (i.e. entropy) of the softmax probability distribution for a bounding box. Consistency is measured by looking at how model output varies in the network. Mean STD looks at how softmax probabilities vary across MC-Dropout inferences, QBC seeks to produce a consistent version space across all bounding box detections, and localization stability seeks to produce consistent bounding box detections across noisy images. Secondly, does the acquisition function focus solely on the classification aspect of the network or does the acquisition function incorporate the localization and classification aspect of the network? We categorize the aforementioned acquisition functions in Table 2.1.

Table 2.1: Categorization of acquisition functions for object detection.

Acquisition Function	Confidence vs. Consistency	Classification vs. Localization
Entropy	Confidence	Classification
BALD	Confidence	Classification
Var Ratio	Confidence	Classification
Mean STD	Consistency	Classification
Margin Sampling	Confidence	Classification
Localization Stability	Consistency	Both
QBC	Consistency	Both

2.4 Reinitialization of Network Weights

Aside from variations in query methods, many researchers have varied their network training process. Some researchers have re-initialized their network weights upon acquisition of new data [15] [7] [11], while some researchers did not re-initialize network weights upon acquisition of new data [9] [8]. According to our literature review, there are no such studies that explore the necessity of network weight re-initialization in an active learning setting for object detection. As such, this thesis will also explore the impact of weight re-initialization on deep object detectors upon acquisition of new data.

Chapter 3

Method

Our exploration and comparison of recent advances in active learning consists of two stages. First, we compared the effectiveness of these methods in a classification setting using the MNIST dataset [26]. We wanted to see how well these techniques translate from a classification setting into an object detection setting which requires the localization and classification of objects in an image. Second, we compared the effectiveness of these methods in an object detection setting using the PASCAL VOC2007 dataset [18]. Using the PASCAL VOC2007 dataset, we also investigated the effects of network weight re-initialization on our active learning process.

All experiments were conducted using a custom built desktop computer with a quad core Intel(R) Core i7-7700K CPU @ 4.2GHz, NVIDIA GeForce GTX 1070 8GB GDDR5 GPU and with 32GB of RAM.

3.1 General Experimental Protocol

Algorithm 2: Active Learning Experimental Protocol

input: model, batch size (b), acquisition iterations (ai), acquisition function (af), number of epochs (n), queries (q), queries to pick (p), pool_set, evaluation_set

```
training_set = pool_set.initializeTrainingSet()
model.train(training_set, n, b)
model.evaluate(evaluation_set)
for i = 1 to ai do
    new_data = pool_set.acquireData(af, q, p, model)
    training_set.insert(new_data)
    model.train(training_set, n, b)
    model.evaluate(evaluation_set)
```

The general protocol for each of our active learning experiments is outlined in Algorithm 2. Since we used datasets consisting of training images that have already been annotated, we were able to emulate the existence of an oracle without having to label our own data.

We start off by partitioning our data into a pool set and an evaluation set. At the beginning of the algorithm (line 1 to 3), we will initialize the training set by extracting a fixed number of samples from the pool set, train on it and evaluate our performance. At every acquisition iteration (i), we will sample from the pool set using our acquisition function for new data. This data is then removed from the pool and added to the training set. In essence, this emulates the process of the a model querying from an unlabelled pool and seeking an oracle for labelling as in Figure 2.1. When we acquire data from our pool data set we will only run our acquisition function on a fixed number of queries (q) and select p of them (line 5). Once the new data is added to training set, we will train the model for a fixed number of epochs and then evaluate our model. We repeat the process of acquiring new data and training a new model until we have reached the maximum number of acquisition iterations ai.

3.2 MNIST

All experiments on the MNIST dataset were done using the same model structure: convolution-relu-convolution-relu-max_pool-dropout-dense-relu-dropout-softmax, with 32 convolution kernels of 3x3 kernel size, 64 convolutions of 3x3 kernel size, 2x2 pooling, dropout layer with probability of 0.25, a dense layer of 128 units and another dropout layer with probability of 0.5. The scripts to reproduce our MNIST experiments are available on Github [1].

Our training method is similar to the method used by Gal et. al [11] in which we used a standard test set of 10K images and we use the rest of the 40K points as pool points. Our training set initialization involves selecting 2 images of each of the 10 digits for a total of 20 training examples. We used a batch size of 128 images (b=128), 50 epochs (n=50), 99 acquisition iterations (ai). At each acquisition iteration we sampled from the entire pool dataset (q = size of pool dataset) and selected 10 images (p = 10).

The acquisition functions that were tested using the MNIST dataset were: R (random), ENT (entropy), BALD, VR (var ratio), MSTD (mean STD), MS (margin sampling) and CS (classification stability). The calculation of ENT, BALD, VR, MSTD and MS have been detailed in Sections 2.2.1 and 2.2.2. We used 10 inferences for the computation of MC-Dropout. CS is a method we invented to emulate localization stability in a classification setting. This method applies zero mean Gaussian noise with a variance of 0, 2, 4, 6, and 8 to each image. The mean of the variance in softmax probability for each class is used to score each image. Similar to the intuition behind localization stability, we selected images with detections that were the most susceptible to noise (high variance score).

3.3 PASCAL VOC2007

Similar to Kao et al. [7], we studied the aforementioned active learning techniques using the VOC2007 trainval set [18] to create an unlabelled pool of 5011 images. We use the 4,952 images in the VOC2007 test set to evaluate the performance of our models.

[1] `https://github.com/CtfChan/keras_mnist_al`

Our training set initialization involves selecting 500 random training examples and training model for 40 epochs at a learning rate of 1e-03 (line 1 and 2 of Algorithm 2). Following the notation in Algorithm 2, we used a batch size of 8 images (b=8), 28 epochs (n=28), 14 acquisition iterations (ai). At each acquisition iteration we sampled 500 images from the pool dataset (q = 500) and selected 250 images (p = 250).

All active learning algorithms were tested with the Single Shot multibox Detector (SSD) [22] using VGG16 as the pre-trained network in each experiment. We also used the 'image expansion data augmentation' trick [22] proposed by Liu et al. in our training pipeline. The code for our VOC experiments are also available on Github [2]. All input images from the dataset are resized to 300×300 before being fed into the network. In order to be able to do MC-Dropout, we added dropout with a probability of 0.1 to FC6 and FC7 of the VGG16 backbone of the network. This network generates a tensor of size $(8732 \times (4 + 21))$ which represents 8732 priors on object locations. The 4 is used to represent the bounding box offset (along x and y axis) and spatial extent (height and width) of each prior. The 21 is used to represent the softmax probabilities of the 21 possible classes (20 VOC classes + background class) for each bounding box.

The acquisition functions that were tested on the VOC2007 object detection dataset include: R (random), MS (margin sampling), BALD, ENT (entropy), VR (var ratio), MSTD (mean STD), MSTDB (mean STD with bounding box), QBC (query-by-committee) and LS (localization stability). Dropout was only enabled at test time for ENT, VR, MSTD, BALD and MSTDB. Similar to our experiments with MNIST, we used T=10 inferences to do MC-Dropout.

BALD and ENT score each image by summing up the BALD/ENT score for every bounding box in an image. MSTD and MSTDB were computed by taking the average mean standard deviation scores for all bounding boxes, the difference being, MSTD computes the variance across only class scores while MSTDB computes the variance across the class scores and the four bounding box parameters. VR is computed by taking the max var ratio across all bounding boxes in an image. QBC and LS are implemented as described in Section 2.

A total of 6 trials were conducted on the PASCAL VOC2007 dataset for each acquisition function. The first 3 trials involved no re-initialization of network weights. For the last three trials, we re-initialized the network weights before line 7 of Algorithm 2 to the weights of the network after training on the initial 500 images on line 2.

[2] `https://github.com/CtfChan/pytorch_ssd_active_learning`

Chapter 4

Results and Discussion

4.1 MNIST

4.1.1 Test Set Performance

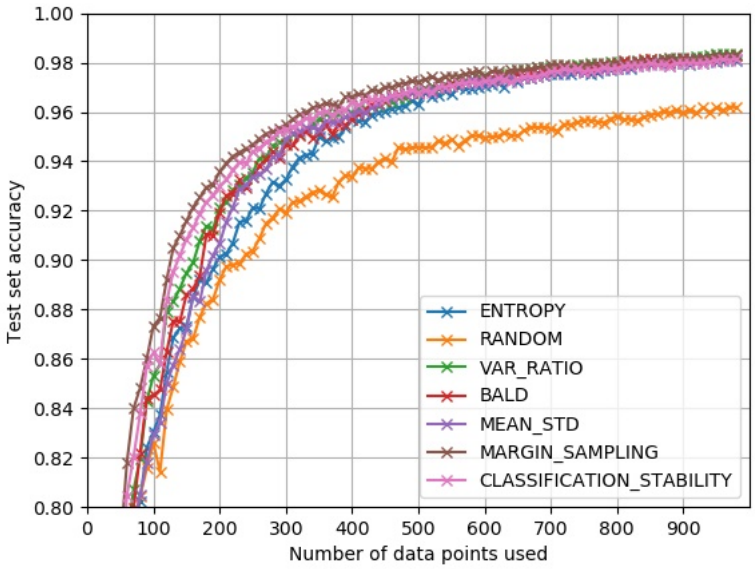


Figure 4.1: Test set error of various acquisition functions on MNIST.

Fig 4.1 shows the test accuracy of each acquisition function as a function of the number of data points used. The test accuracies have been averaged over the course of five trials for each acquisition function. We found that all of the methods that were tested outperformed our baseline of R. Not only did the other acquisition functions enable faster learning when compared with R, they also enabled the network to outperform the test set accuracy of R.

In Table 4.1, we provide the number of images required for each acquisition type to reach test errors of 5% and 10%. MS outperformed all other methods in both categories, followed by CS. The MC-Dropout based methods (ENT, BALD, VR and MSTD) for active learning were the least effective at acquiring new samples, but still more effective than a random acquisition strategy. Nonetheless, this table illustrates the effectiveness of active learning -

Acquisition Function	10% error	5% error
R	260	700
ENT	240	400
VR	190	360
BALD	180	380
MSTD	230	370
MS	**130**	**270**
CS	150	290

Table 4.1: Number of images required to achieve X % test set error.

a machine learning engineer using an MS acquisition function to select images to label could save more than half of his/her labelling efforts. For example, to achieve a 5% test error, MS only required 39% of the 700 images used by R.

4.1.2 Importance of Acquiring the Right Data

To further reinforce the importance of acquiring the right images for your model to train with, we illustrate the effect of acquiring the least valuable training examples from the unlabelled pool in Figure 4.2. For example, rather than picking samples with the maximal entropy, we picked those with the least entropy. We can see that there is a significant reduction in test set accuracy in all of our acquisition functions other than R. In fact, for certain acquisition functions such as entropy and BALD, we see virtually no improvement in accuracy as move samples are added to the training set.

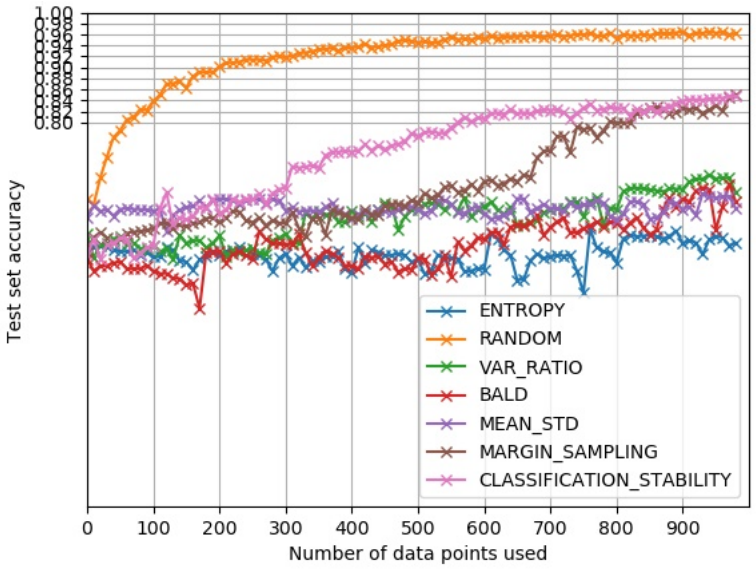
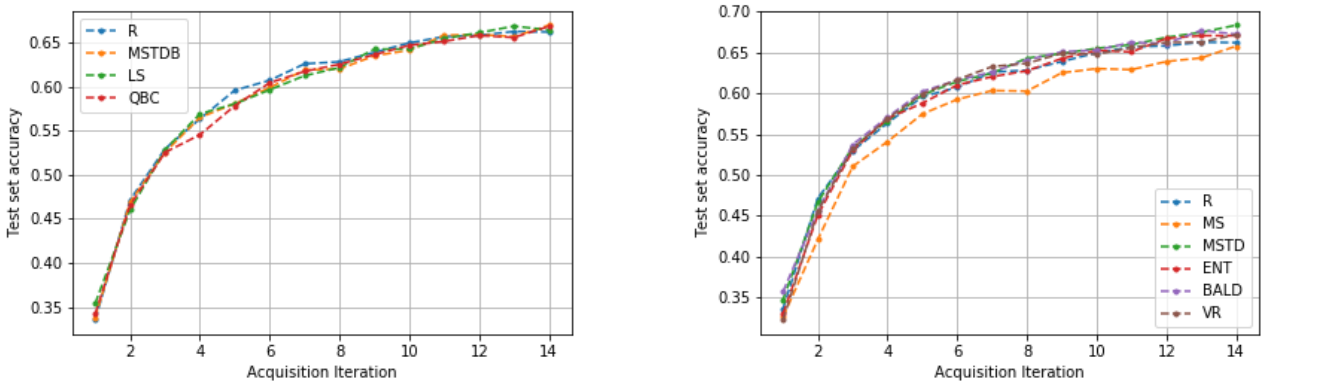


Figure 4.2: Test set error of various acquisition functions on MNIST using an incorrect metric.

4.2 PASCAL VOC 2007

4.2.1 Test Set Performance without Weight Re-Initialization

The performance of each acquisition function without weight re-initialization is displayed in Figure 4.3. For ease of visual comparison, we have chosen to show the methods that incorporate localization and the methods that do not incorporate localization in two separate plots. The test set accuracy refers to the averaged mAP performance of each acquisition at each acquisition iteration over the course of 3 trials. Unlike our experiments in MNIST, we did not see a large offset in performance between random acquisition and our other acquisition functions. Surprisingly, the mAP scores of methods that incorporate localization were sometimes below that of R. The best performing acquisition function in our MNIST experiments, MS, actually achieved the lowest mAP scores across all acquisition iterations. This highlights the fact that acquisition functions that perform well in classification tasks do not necessarily perform well in an object detection tasks. We also note that the performance improvement of the acquisition functions over R is noticeably less than in MNIST.



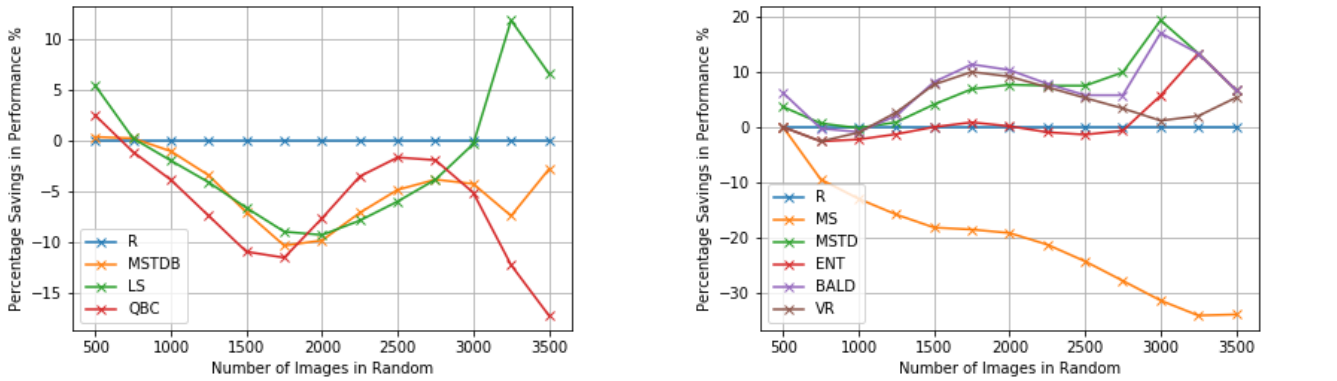
(a) Methods that incorporate localization (b) Methods that do not incorporate localization

Figure 4.3: Test set performance for VOC2007 without weight re-initialization.

The relative savings of each active learning strategy relative to R in shown Figure 4.4 and the averaged relative savings across all acquisition iterations is shown in Table 4.2 below. We computed the relative savings at each acquisition iteration by comparing the number of images required for R and the acquisition function of interest to reach a certain mAP score. From a relative savings standpoint, none of the methods that incorporate localization provided a positive benefit over R. On the other hand, all of the methods that did not incorporate localization, with the exception of MS, provided a positive relative savings over R. Contrary to the potential maximum relative savings of 39% found in MNIST, the maximum average relative savings we were able to achieve was about 6% on VOC2007.

Our findings differ from what was found in Kao et al and Roy et al. In Kao et al. they found that LS provided a relative savings of 12 to 22 % over R [7]. In Roy et al. they found that [9] QBC was able to learn faster than ENT. There are a variety of potential reasons for these differences such as different parameters and different implementations. For

instance, Roy et al. also integrated an exploration/exploitation framework into their active learning experiments [9]. We chose not to include this framework because it would distort our comparison of vanilla acquisition functions. We were unable to verify the source of these discrepancies as the code for these works was not made public at the time of our research.



(a) Methods that incorporate localization

(b) Methods that do not incorporate localization

Figure 4.4: Plot of relative savings over random acquisition on VOC2007 without weight re-initialization.

Acquisition Function	Average Relative Savings over Random in %
MS	-20.468
QBC	-5.37
MSTDB	-4.383
LS	-2.687
ENT	0.59
VR	4.438
MSTD	5.255
BALD	6.152

Table 4.2: Average relative savings for each acquisition function without weight re-initialization.

4.2.2 Analysis on Acquired Images and Class Performance

Class-wise AP Performance

In Table 4.3 we present the class-wise AP performance of each acquisition function at the end of our first trial. Similar to the findings in Kao et al., the boat, bottle, plant and chair classes were particularly hard for our networks to learn [7]. The maximum AP we achieved on these four classes was below 0.6. The classes for which we achieved the strongest AP were car, cat, horse and train; we achieved a maximum AP of above 0.8 for all the these classes.

Given that MSTD achieved the highest average mAP score for the final acquisition iteration in our previous experiments, it is not surprising that MSTD achieved the highest mAP in many of the categories: MSTD achieved the best AP score in eight out of the 20 classes in the dataset.

Class	MS	R	LS	MSTD	ENT	BALD	VR	MSTD	QBC
aeroplane	0.728	0.704	0.718	0.718	0.69	0.708	**0.739**	0.712	0.711
bicycle	0.774	0.736	0.78	0.783	0.772	**0.798**	0.772	0.793	0.787
bird	0.641	0.617	0.639	0.659	0.563	0.629	**0.663**	0.656	0.609
boat*	0.518	0.564	0.56	**0.588**	0.572	0.552	0.55	0.579	0.569
bottle*	0.242	0.322	0.348	**0.374**	0.373	0.326	0.362	0.34	0.319
bus	0.77	0.767	0.747	0.774	0.776	0.783	0.763	**0.783**	0.76
car	**0.819**	0.803	0.798	0.807	0.804	0.797	0.79	0.803	0.797
cat	0.78	0.769	0.775	0.755	0.753	0.775	0.78	0.748	**0.8**
chair*	0.358	0.41	0.45	**0.46**	0.45	0.452	0.451	0.438	0.444
cow	0.661	0.589	0.677	0.708	0.663	0.711	0.696	0.705	**0.726**
diningtable	0.625	0.605	0.648	**0.709**	0.686	0.704	0.683	0.651	0.661
dog	0.713	0.687	0.749	0.673	0.696	0.726	**0.752**	0.729	0.739
horse	0.815	0.807	0.809	0.82	0.809	0.815	0.803	**0.829**	0.823
motorbike	0.757	0.699	0.771	**0.794**	0.772	0.779	0.786	0.776	0.768
person	0.716	0.719	0.72	**0.737**	0.705	0.726	0.733	0.709	0.72
pottedplant*	0.391	**0.444**	0.376	0.433	0.404	0.425	0.407	0.413	0.38
sheep	0.611	0.562	0.633	**0.681**	0.632	0.658	0.652	0.639	0.668
sofa	0.644	0.693	0.684	0.697	0.665	0.684	**0.726**	0.622	0.681
train	0.808	0.786	0.785	0.786	0.804	0.802	**0.819**	0.802	0.764
tvmonitor	0.63	0.663	0.674	**0.694**	0.678	0.679	0.693	0.668	0.651

Table 4.3: Class-wise AP performance of each acquisition function. Asterisks (*) are used represents difficult class, which we defined as those with a maximum AP less than or equal to 0.6. Bolded AP scores represent the highest achieved AP score for a certain class.

Class-wise Objects Acquired

We further our analysis of the class-wise AP performance by investigating the total number of objects acquired by each acquisition function and developing a class-wise breakdown of the objects acquired in Table 4.4 and in Table 4.5 respectively.

Acquisition Function	MS	QBC	MSTDB	LS	R	VR	BALD	ENT	MSTD
Total Objects Acquired	11179	12119	12527	12534	12559	12999	13763	13806	13812

Table 4.4: Total number of objects acquired by each acquisition function. Sorted in ascending order from left to right.

We found that the number of object acquired did not directly translate to better mAP performance. MSTD, one of the better performing acquisition functions, acquired approximately 2600 more objects than MS, one of the worst performing acquisition functions. However, ENT, another acquisition function that performed poorly in our experiments, acquired

Class	MS	R	LS	MSTD	ENT	BALD	VR	MSTDB	QBC
aeroplane	301	253	**316**	243	232	249	270	290	273
bicycle	332	335	278	363	**381**	362	317	332	364
bird	479	456	475	516	480	498	519	**554**	444
boat	343	317	**383**	342	350	338	351	344	296
bottle	340	519	537	587	**595**	577	570	488	449
bus	247	212	228	236	235	237	209	**249**	213
car	**1448**	1321	1309	1365	1367	1341	1060	1337	1287
cat	299	299	**346**	278	232	246	294	311	344
chair	664	1118	1305	1324	1359	**1371**	1334	1126	1034
cow	196	294	246	319	294	316	268	**321**	296
diningtable	154	247	277	291	292	**298**	292	237	223
dog	398	430	413	440	393	406	406	398	**449**
horse	343	327	281	**364**	335	355	302	363	363
motorbike	235	302	256	**361**	336	353	319	324	321
person	3900	4427	4209	4889	**5049**	4909	4675	4163	4290
pottedplant	439	547	461	600	594	**604**	580	524	356
sheep	274	283	221	298	273	308	259	**313**	253
sofa	265	341	**392**	385	389	372	391	297	322
train	288	259	277	279	**296**	290	243	292	266
tvmonitor	234	272	324	332	324	333	**340**	264	276

Table 4.5: Number of objects acquired by each acquisition function by class.

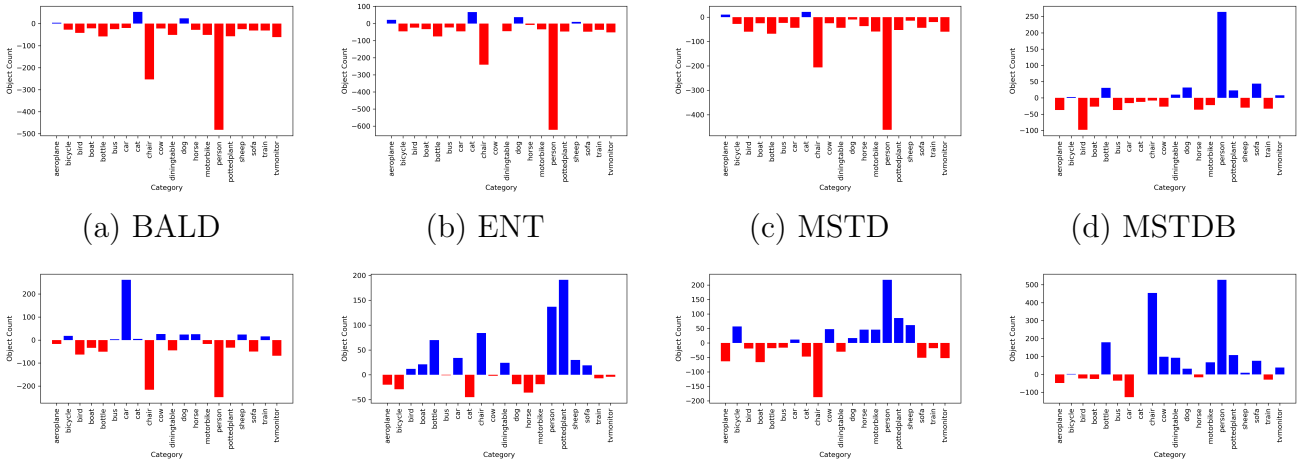
the second most objects out of all the acquisition functions we tested. It should also be noted that many of the MC-Dropout based methods such as MSTD, ENT, BALD and VR were among the acquisition functions that acquired the greatest number of objects and many of these acquisition functions were also among the best performing acquisition functions.

Another interesting finding was that acquiring more objects of a certain class did not correlate with better performance in that class. ENT acquired the most person objects (5049), yet it's model did not achieve the highest AP for this class. MS, only acquired 77% of the number of person objects ENT acquired and it's AP was only 0.009 behind that of ENT. In some cases we also found situations where a lack of samples drawn from a certain class potentially led to large deficit in performance. For instance, MS acquired half the number of chair objects as BALD. As a result, BALD's AP score for chairs was 30% higher than that of MS.

Difference in Number of Objects Acquired

Figure 4.5 shows the difference in the number of objects acquired between each acquisition function and random acquisition. An enlarged version of these plots is available in Appendix A. A blue bar indicates that random acquired more objects than a certain acquisition function, while a red bar indicates that the acquisition function of interest acquired more objects than random acquisition. All MC-Dropout based methods with the exception of MSTDB heavily favored selecting images with persons in them. Most MC-Dropout based methods tend to have many red bars, suggesting that they acquired more objects than random acquisition across all categories. Many of the best performing acquisition functions tended to show many red bars, while many of the worst performing acquisition functions tended to have many blue bars. A large number of red bars indicate that an acquisition func-

tion tended to acquire more objects than random across a wide array of classes, suggesting a greater diversity in the training data that was selected



(a) BALD (b) ENT (c) MSTD (d) MSTDB

(e) VR (f) QBC (g) LS (h) MS

Figure 4.5: Class-wise comparison between images acquired between random and the other acquisition functions.

t-SNE Exploration

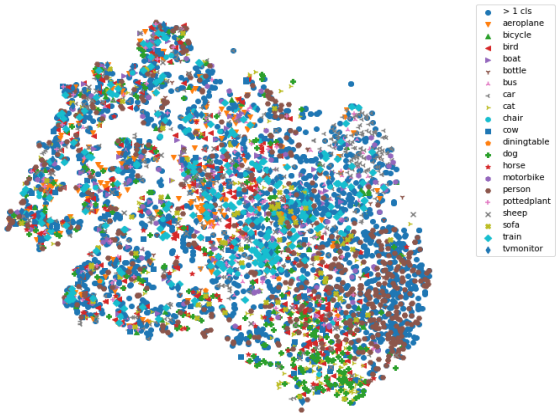


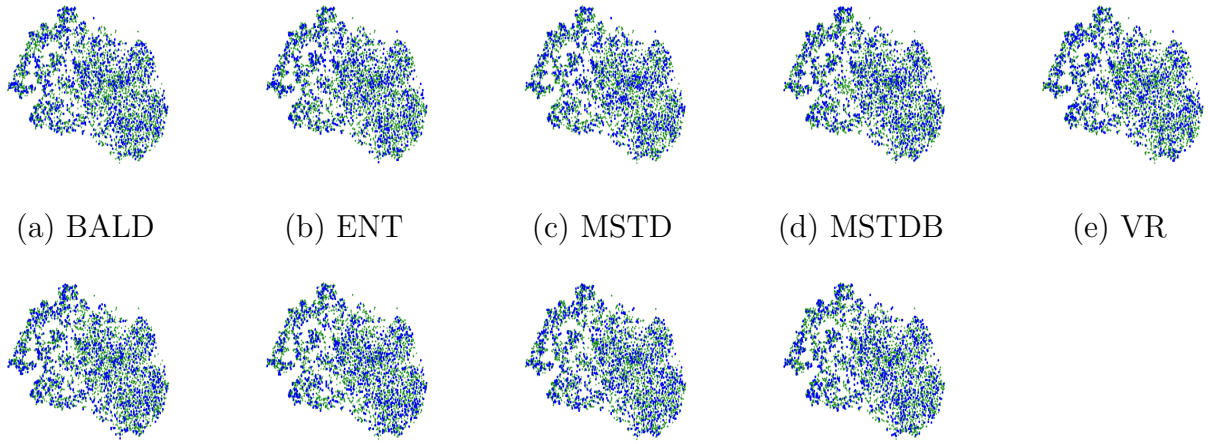
Figure 4.6: t-SNE Visualization for entire VOC2007 dataset.

We also conducted a visual exploration of the acquisition functions using t-SNE, which is a method of visualizing high dimensional data in lower dimensions. The enlarged version of our t-SNE figures are available in Appendix A.

Our t-SNE visualization for the entire training set of the VOC2007 dataset is shown in Figure 4.6. Following the methodology of Kao et. al., we used the features extracted from the conv5_3 layer as the high dimensional vector in our t-SNE plot [7]. Due to memory

constraints, we first had to perform PCA on this high dimensional vector to reduce it to a 30 dimensional vector before using t-SNE to project these vectors onto a 2D plane. If an image has more than one class present, it is labelled as a "> 1 cls" category, otherwise it is categorized using the only class present in that image.

We visualize the first 1000 acquired images in the first trial for each acquisition function in Figure 4.7. The blue dots represent the selected images, while the green dots represent all possible image selections on the plane. As expected, the random acquisition function is spread throughout the entire feature space. On the other hand, other acquisition functions tend to have selections that congregate in one area; ENT has a high density where the persons images are. Many of the MC-Dropout based methods also tend to select data points that contain more than one class.



(a) BALD (b) ENT (c) MSTD (d) MSTDB (e) VR

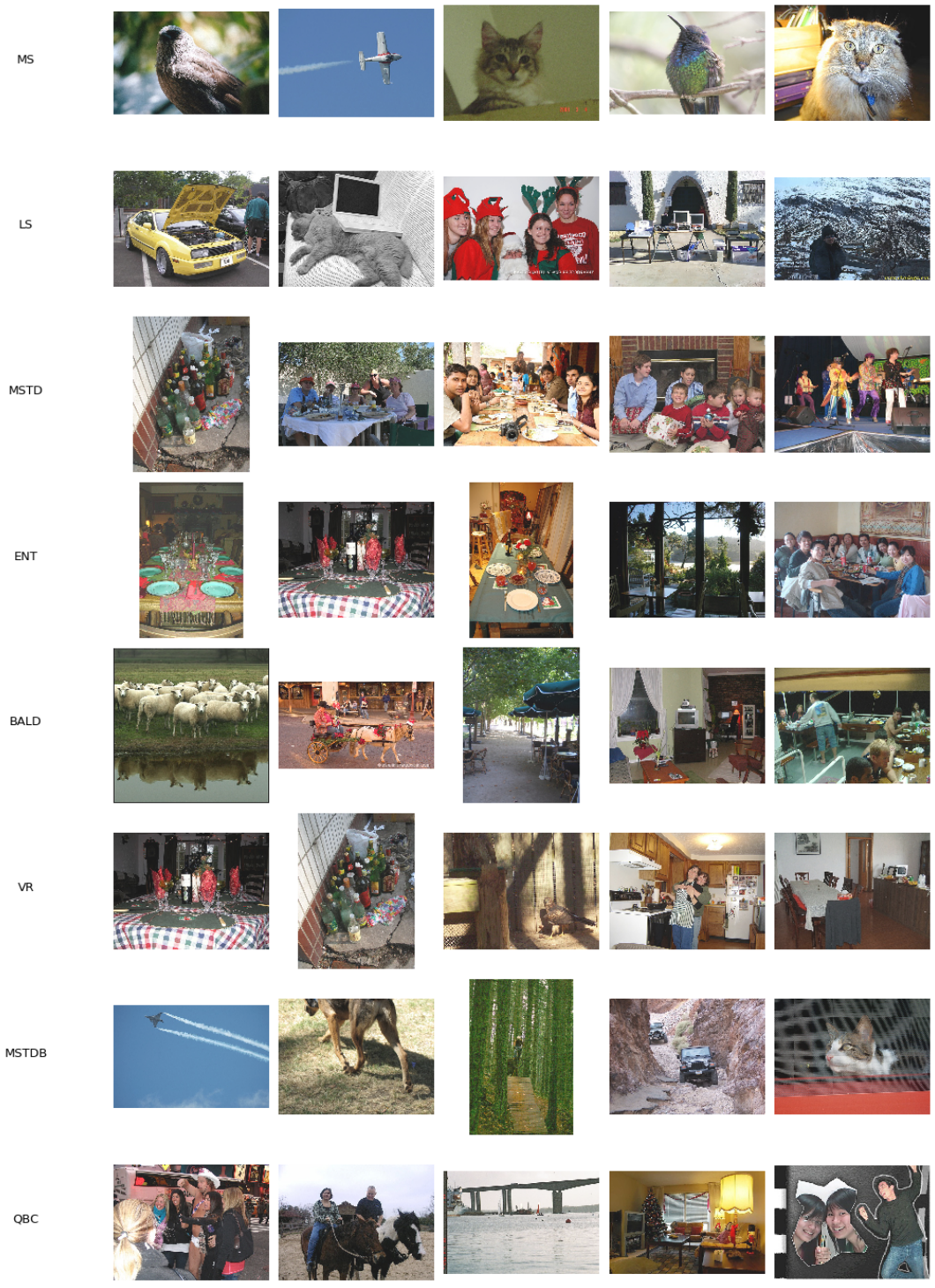
(f) QBC (g) LS (h) MS (i) R

Figure 4.7: t-SNE visualization for each acquisition function.

Visualizing Acquired Images

In Figure 4.8, we visualize the top 5 highest scoring examples from the pool subset in the final acquisition iteration. Here we see that MS focuses heavily on images that have single objects in them. Meanwhile, many of the better performing acquisition functions focused on acquiring images with many objects. MSTD picked images with many people and bottles in them. Similarly, VR picked images with many people, chairs and bottles in them.

Figure 4.8: Five highest scoring examples from each acquisition function at the final acquisition iteration of the first trial.



4.2.3 Timing Analysis

Table 4.6 below summarizes the computational time required for each acquisition function. MS was the cheapest non-trivial acquisition function to compute, requiring only 0.0285s per image. LS and QBC were the most computationally expensive methods, requiring 0.423s and 1.20s of processing time respectively. It should be noted that our implementation of these acquisition functions is single-threaded so there is room to optimize for time. Nonetheless, the table below allows us to gauge the magnitude of additional computational effort required by more complex acquisition functions.

Acquisition Function	500 Images Compute Time (s)	Compute Time Per Image (s)
Random	0.000111	2.22e-07
Localization Stability	211	0.423
Mean STD	58.0	0.116
Entropy	59.5	0.119
BALD	60.1	0.120
Var Ratio	58.2	0.116
Margin Sampling	14.5	0.0285
QBC	600	1.20

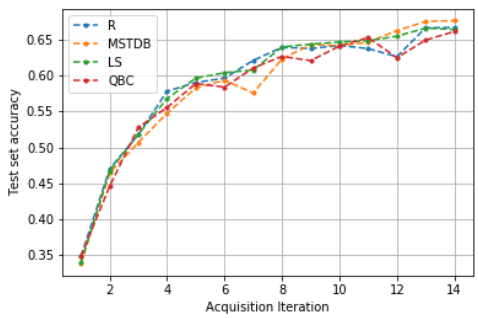
Table 4.6: Computational time per image for each acquisition function.

4.2.4 Weight Re-initialization

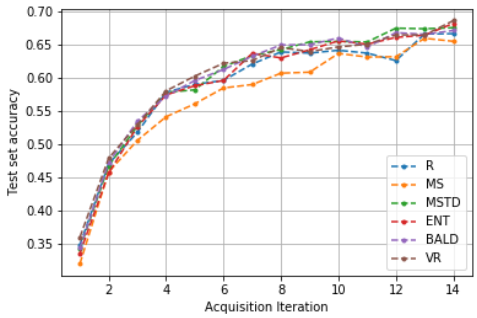
We repeat the previous experiments above, except we re-initialize our network weights to the model trained on line 2 of Algorithm 2. The test set performance plot, relative savings plot and average relative savings averaged over three trials are shown in Figure 4.9, Figure 4.10 and Table 4.7. Many of the best performing acquisition functions such as BALD, VR, MSTD and ENT in our previous experiments experienced an increase in the average relative savings. LS, which previously provided no benefit over R, provided a positive relative savings in our weight re-initialization experiments. Similar to what was previously observed, MS still performed the worse in these experiments.

Acquisition Function	Average Relative Savings over Random in %
MS	-20.802
QBC	-9.956
MSTDB	-4.392
ENT	4.451
LS	5.341
MSTD	8.655
VR	9.114
BALD	10.082

Table 4.7: Average relative savings for each acquisition function with weight reinitialization.

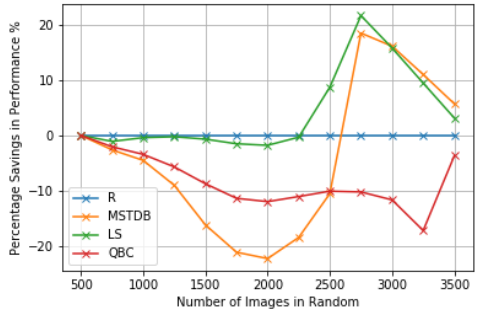


(a) Methods that incorporate localization

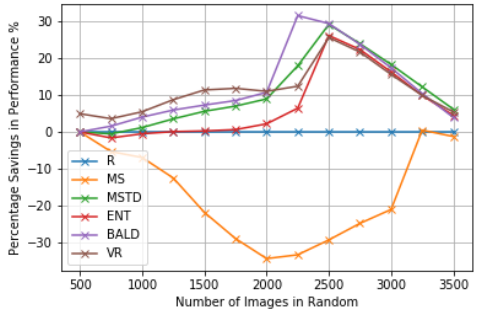


(b) Methods that do not incorporate localization

Figure 4.9: Test set performance on VOC2007 with weight re-initialization.



(a) Methods that incorporate localization



(b) Methods that do not incorporate localization

Figure 4.10: Plot of relative savings over random acquisition on VOC2007 with weight re-initialization.

Chapter 5

Conclusion

5.1 Summary of Findings

In this work we have compared the effectiveness of various acquisition functions in an object detection setting. We summarize our main findings below:

- MC-Dropout based methods such as BALD, MSTD and VR tend to require the less labelled images to achieve a high mAP score than other acquisition functions. Their effectiveness can be attributed to their propensity to select images with many objects and to select images from a wider array of classes.
- Current active learning methods that try to quantify the effectiveness of the network's localization (QBC, LS, and MSTDB) provide little benefit in an active learning scenario for object detection.
- An acquisition function's effectiveness in a classification setting does not directly translate to an object detection scenario. We observed this phenomenon in the MS acquisition function, which performed the best in our MNIST experiments, but performed the worst in our VOC2007 experiments.
- Re-initializing weights after the acquisition of new training data in an active learning scenario does enhance the effectiveness of the acquisition function. This supports a claim made by Gal et al. sugesting that resetting model weights upon acquisition can prevent local minima [11].

5.2 Future Work

Active learning for object detection remains a largely unexplored field of study. We outline some of key areas for future work below:

- Investigate better metrics for labelling effort. In this study, we used the number of labelled images as a heuristic for the labelling effort required to train a model. We found in our analysis that better performance often correlated with acquiring a greater

number of objects. Although a hypothetical oracle labels less images, they might be required to spend more time labelling images with many objects. Future metrics for labelling effort should also incorporate the number of objects, size of objects and diversity of objects in an image.

- There remains many traditional active learning query strategy frameworks that have not been applied to object detection. For instance, the estimated error reduction and expected model change technique mentioned in Settle's active learning survey [6] have not been applied to object detection before and could prove to be far more effective than existing methods. Future studies could try to apply these frameworks to object detection.

- Some researchers have recently suggested the use of adaptive frameworks for active learning in object detection [27]. Desai et al. proposed switching between strong labelling (drawing bounding boxes around objects) and weak labelling (center-clicking objects) to save labelling efforts. Comparing these adaptive methods to existing active learning approaches that rely solely on strong labelling would also make an interesting study.

Bibliography

[1] H. Su, J. Deng, and L. Fei-Fei, "Crowdsourcing Annotations for Visual Object Detection," en, p. 7,

[2] D. P. Papadopoulos, J. R. R. Uijlings, F. Keller, and V. Ferrari, "We don't need no bounding-boxes: Training object class detectors using only human verification," *arXiv:1602.08405 [cs]*, Feb. 2016, arXiv: 1602.08405. [Online]. Available: `http://arxiv.org/abs/1602.08405` (visited on 10/05/2019).

[3] ——, "We don't need no bounding-boxes: Training object class detectors using only human verification," *arXiv:1602.08405 [cs]*, Apr. 2017, arXiv: 1602.08405. [Online]. Available: `http://arxiv.org/abs/1602.08405` (visited on 01/06/2020).

[4] ——, "Training object class detectors with click supervision," *arXiv:1704.06189 [cs]*, May 2017, arXiv: 1704.06189. [Online]. Available: `http://arxiv.org/abs/1704.06189` (visited on 01/06/2020).

[5] Y. Zhu, Y. Zhou, Q. Ye, Q. Qiu, and J. Jiao, "Soft Proposal Networks for Weakly Supervised Object Localization," *arXiv:1709.01829 [cs]*, Sep. 2017, arXiv: 1709.01829. [Online]. Available: `http://arxiv.org/abs/1709.01829` (visited on 01/06/2020).

[6] B. Settles, "Active Learning Literature Survey," en, p. 46,

[7] C.-C. Kao, T.-Y. Lee, P. Sen, and M.-Y. Liu, "Localization-Aware Active Learning for Object Detection," *arXiv:1801.05124 [cs]*, Jan. 2018, arXiv: 1801.05124. [Online]. Available: `http://arxiv.org/abs/1801.05124` (visited on 10/05/2019).

[8] C.-A. Brust, C. Käding, and J. Denzler, "Active Learning for Deep Object Detection," *arXiv:1809.09875 [cs]*, Sep. 2018, arXiv: 1809.09875. [Online]. Available: `http://arxiv.org/abs/1809.09875` (visited on 10/05/2019).

[9] S. Roy, "Deep active learning for object detection," en, p. 12,

[10] A. Freytag, E. Rodner, and J. Denzler, "Selecting Influential Examples: Active Learning with Expected Model Output Changes," en, in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., ser. Lecture Notes in Computer Science, Springer International Publishing, 2014, pp. 562–577, ISBN: 978-3-319-10593-2.

[11] Y. Gal, R. Islam, and Z. Ghahramani, "Deep Bayesian Active Learning with Image Data," *arXiv:1703.02910 [cs, stat]*, Mar. 2017, arXiv: 1703.02910. [Online]. Available: `http://arxiv.org/abs/1703.02910` (visited on 10/05/2019).

[12] W. H. Beluch, T. Genewein, A. Nurnberger, and J. M. Kohler, "The Power of Ensembles for Active Learning in Image Classification," en, in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT: IEEE, Jun. 2018, pp. 9368–9377, ISBN: 978-1-5386-6420-9. DOI: 10.1109/CVPR.2018.00976. [Online]. Available: https://ieeexplore.ieee.org/document/8579074/ (visited on 10/05/2019).

[13] K. Konyushkova, R. Sznitman, and P. Fua, "Introducing Geometry in Active Learning for Image Segmentation," en, in *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile: IEEE, Dec. 2015, pp. 2974–2982, ISBN: 978-1-4673-8391-2. DOI: 10.1109/ICCV.2015.340. [Online]. Available: http://ieeexplore.ieee.org/document/7410697/ (visited on 10/06/2019).

[14] S. D. Jain and K. Grauman, "Active Image Segmentation Propagation," en, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 2864–2873, ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.313. [Online]. Available: http://ieeexplore.ieee.org/document/7780682/ (visited on 10/06/2019).

[15] D. Feng, X. Wei, L. Rosenbaum, A. Maki, and K. Dietmayer, "Deep Active Learning for Efficient Training of a LiDAR 3d Object Detector," *arXiv:1901.10609 [cs]*, Jan. 2019, arXiv: 1901.10609. [Online]. Available: http://arxiv.org/abs/1901.10609 (visited on 10/05/2019).

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf (visited on 01/11/2020).

[17] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft COCO: Common Objects in Context," *arXiv:1405.0312 [cs]*, Feb. 2015, arXiv: 1405.0312. [Online]. Available: http://arxiv.org/abs/1405.0312 (visited on 01/08/2020).

[18] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," en, *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010, ISSN: 0920-5691, 1573-1405. DOI: 10.1007/s11263-009-0275-4. [Online]. Available: http://link.springer.com/10.1007/s11263-009-0275-4 (visited on 01/13/2020).

[19] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," en, in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI: IEEE, Jun. 2012, pp. 3354–3361, ISBN: 978-1-4673-1228-8 978-1-4673-1226-4 978-1-4673-1227-1. DOI: 10.1109/CVPR.2012.6248074. [Online]. Available: http://ieeexplore.ieee.org/document/6248074/ (visited on 10/05/2019).

[20] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep Learning for Generic Object Detection: A Survey," *arXiv:1809.02165 [cs]*, Aug. 2019, arXiv: 1809.02165. [Online]. Available: http://arxiv.org/abs/1809.02165 (visited on 01/11/2020).

[21] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," en, p. 6,

[22] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," *arXiv:1512.02325 [cs]*, vol. 9905, pp. 21–37, 2016, arXiv: 1512.02325. DOI: 10.1007/978-3-319-46448-0_2. [Online]. Available: http://arxiv.org/abs/1512.02325 (visited on 01/08/2020).

[23] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., Curran Associates, Inc., 2017, pp. 6402–6413. [Online]. Available: http://papers.nips.cc/paper/7219-simple-and-scalable-predictive-uncertainty-estimation-using-deep-ensembles.pdf (visited on 10/06/2019).

[24] Y. Gal, "Uncertainty in Deep Learning," en, p. 174,

[25] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," *arXiv:1506.02142 [cs, stat]*, Jun. 2015, arXiv: 1506.02142. [Online]. Available: http://arxiv.org/abs/1506.02142 (visited on 10/06/2019).

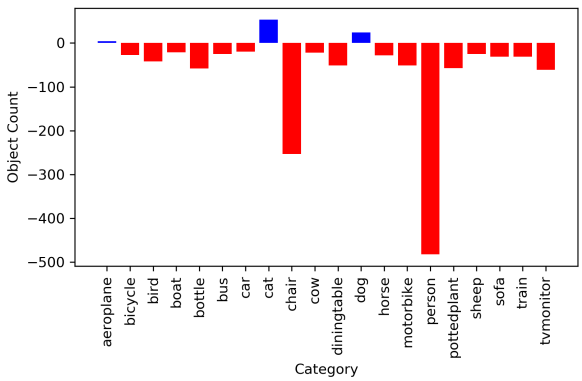
[26] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/.

[27] S. V. Desai, A. Chandra, W. Guo, S. Ninomiya, and V. Balasubramanian, "An adaptive supervision framework for active learning in object detection," Aug. 2019.

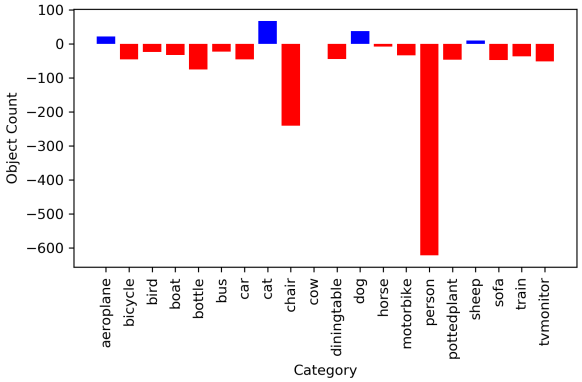
Appendices

Appendix A

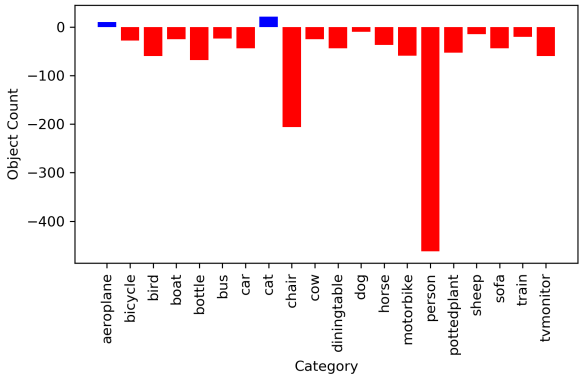
Enlarged photos of data visualizations



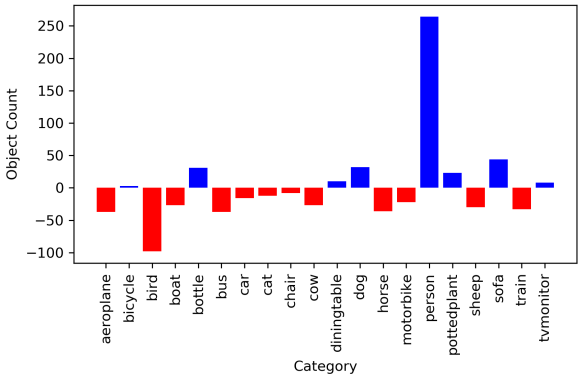
(a) BALD



(b) ENT

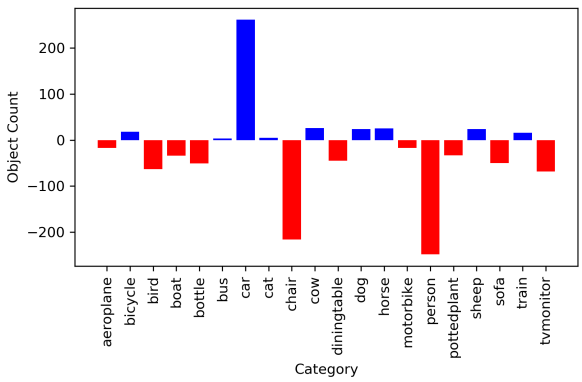


(c) MSTD

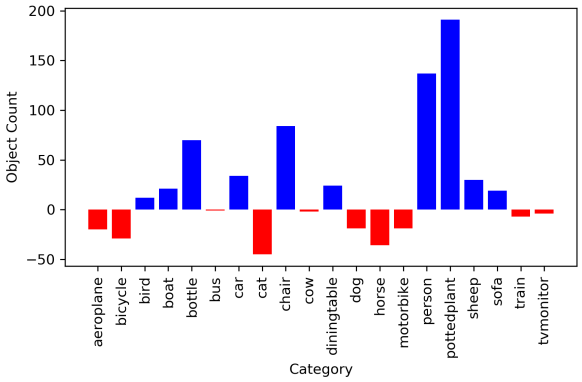


(d) MSTDB

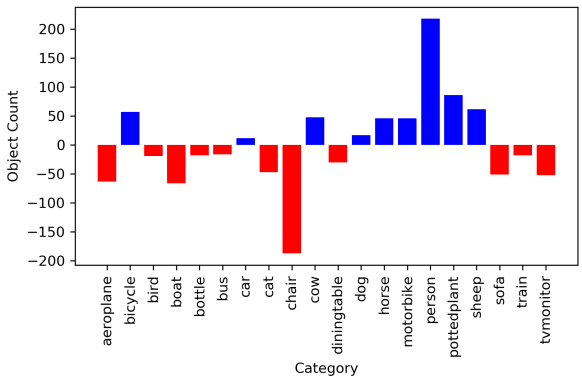
Figure A.1: Class-wise comparison between images acquired between random and the other acquisition functions (part 1).



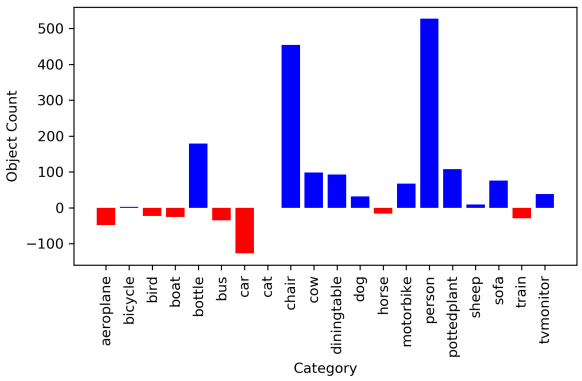
(e) VR



(f) QBC



(g) LS



(h) MS

Figure A.2: Class-wise comparison between images acquired between random and the other acquisition functions (part 2).

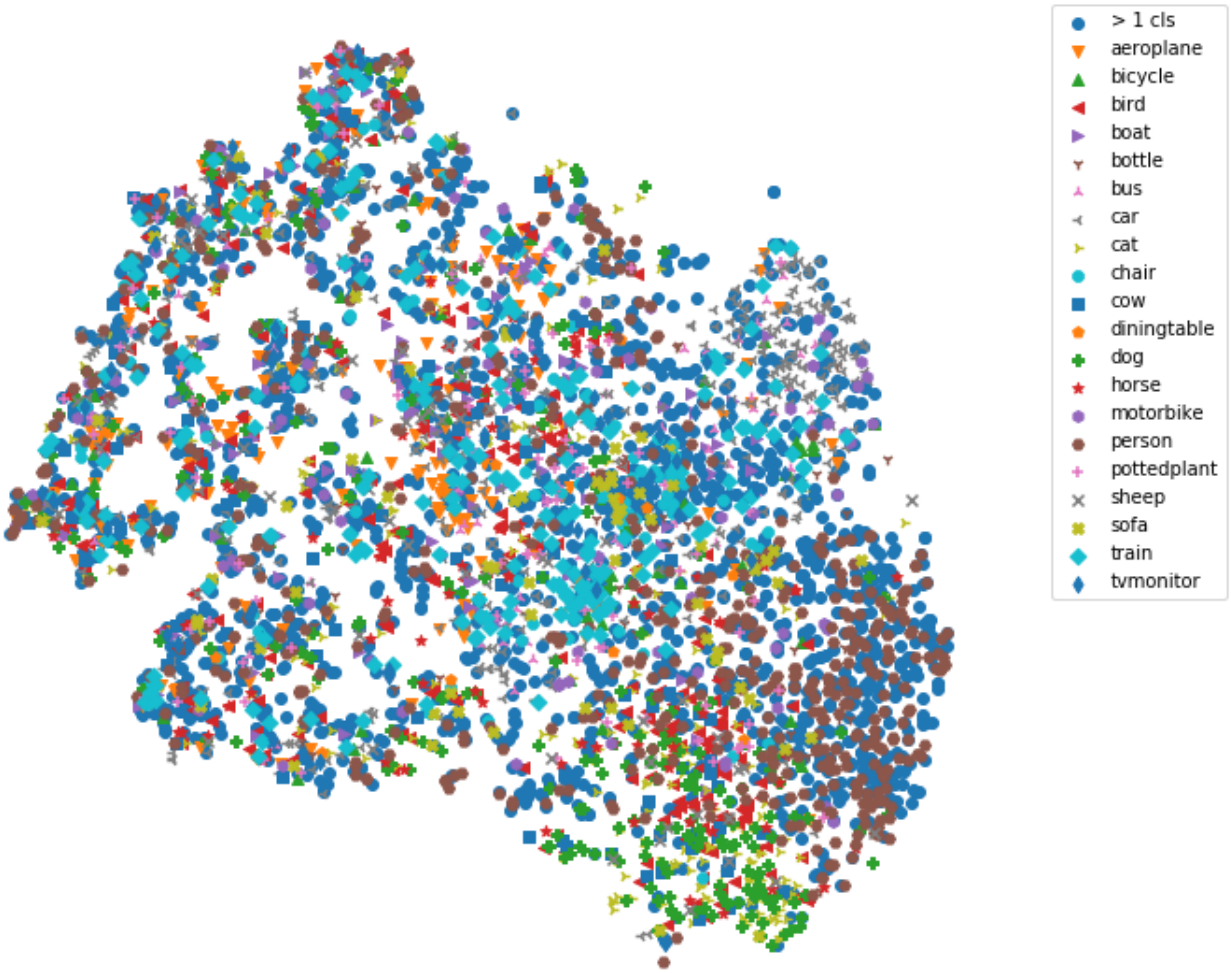
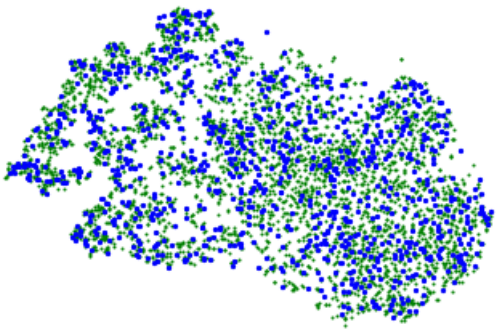
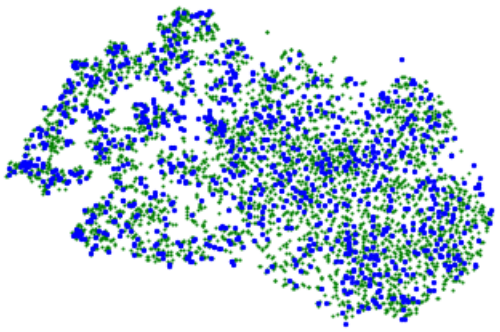


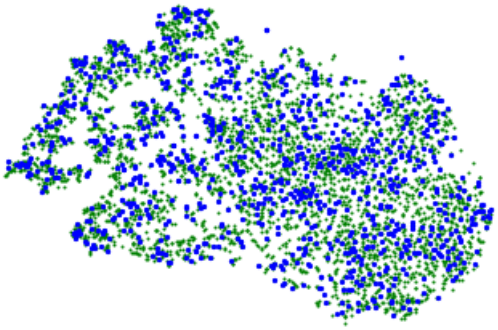
Figure A.3: t-SNE Visualization for entire VOC2007 dataset.



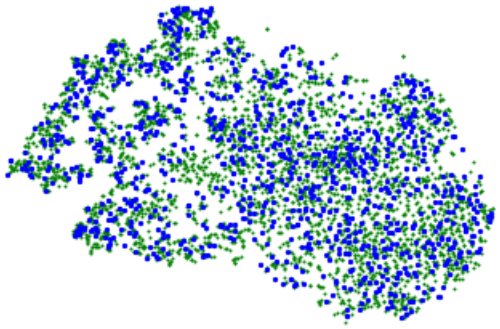
BALD



ENT

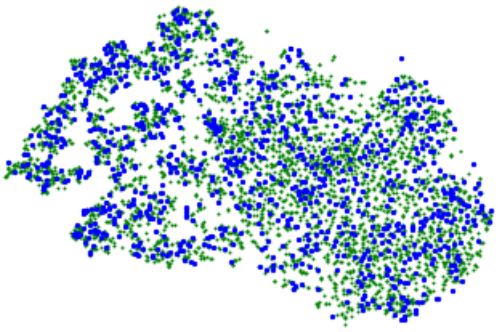


MSTD

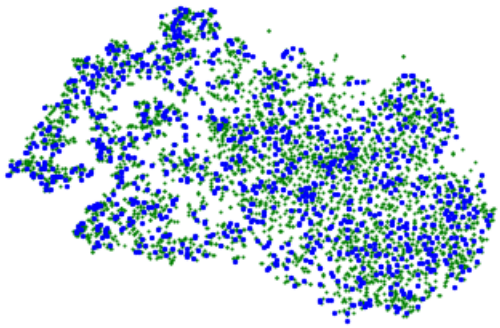


MSTDB

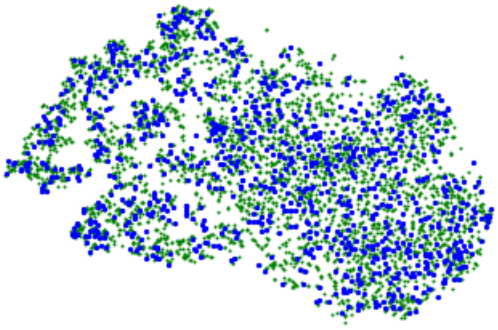
Figure A.4: t-SNE visualization for each acquisition function (part 1)



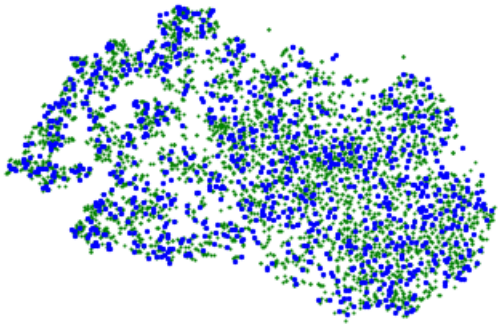
VR



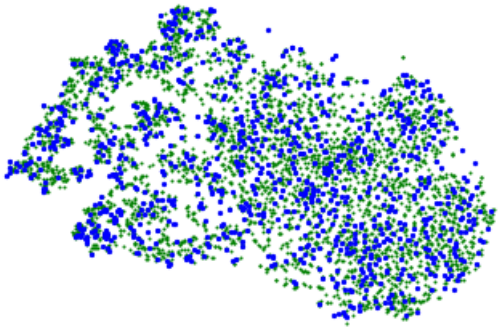
QBC



LS



MS



R

Figure A.5: t-SNE visualization for each acquisition function (part 2).