Budget Active Learning For Deep Networks

Patrick Kinyua Gikunda[1 2] and Nicolas Jouandeau[2]

[1] Department of Computer Science,
Dedan Kimathi University of Technology, Kenya,
kinyuagikunda@gmail.com
[2] Advanced Computer Science Laboratory of Saint Denis , LIASD,
University of Paris 8, France,
n@ai.univ-paris8.fr

Abstract. In the digital world unlabeled data is relatively easy to acquire but expensive to label even with use of domain experts. On the other hand, state-of-the-art Deep Learning methods are dependent on large labeled datasets for training. Recent works on Deep Learning focus on use of Active Learning with uncertainty for model training. Although most uncertainty Active Learning selection strategies are very effective, they fail to take informativeness of the unlabeled instances into account and are prone to querying outliers. In order to address these challenges, we propose a *Budget Active Learning* (BAL) algorithm for Deep Networks that advances active learning methods in three ways. First, we exploit both the uncertainty and diversity of instance using uncertainty and correlation evaluation metrics. Second, we use a budget annotator to label high confidence instances, and simultaneously update the selection strategy. Third, we incorporate Active Learning in Deep Networks and perform classifications on untrained and pretrained models with two classical and a plant-seedling sets of data while minimizing the prediction loss. Experimental results on the three datasets of varying sizes demonstrate the efficacy of the proposed BAL method over other state-of-the-art Deep Active Learning methods.

Keywords: Active Learning, Budget Learning, Deep Network

1 Introduction

Current ICT technologies include Internet of Things [**?**], Remote Sensing [**?**], Cloud Computing [**?**] and Big Data [**?**]. The continuous use of these technologies to collect, monitor, measure, store and analyze data has led to a phenomena of Big Data [**?**] which is in abundance of unlabeled data. Unlabeled data is relatively easy to acquire and it is expensive to label even with use of domain experts. For example, its expensive to hire dermatologists to annotate 129,450 skin cancer images [**?**]. Even when using state-of-the-art computing resources, training a Machine Learning (ML) model on large data sets can take long time. However, like other ML researchers [**?**], we believe that *ML algorithm does not need all of the available data for training*. The main motivation for use of *Active Learning* (AL) is that, if a learning algorithm can pick the data it want to learn from, then a small set of selected data-points can be used for training. Typically this process would involve randomly sampling large amount of data from underlying

distribution for training a model. Collecting large amount of labeled data for training is time consuming and expensive. AL provides methods for analyzing vast amount of data with improved efficiency than other computing approaches, because of the ability to iteratively select the most informative data samples [**?**]. AL is a semi-supervised method meaning that it does not require labels of all the samples in a dataset. In unsupervised methods no labeled samples are used and for fully supervised all samples are labeled. The decision of how much data to use for training a Deep Learning model or alternatively the level of performance required is a resource management decision.

The emphasis in AL is to evaluate the informativeness of an instance, with an assumption that an instance with higher classification uncertainty is more crucial to label. This classical approach usually uses statistical theory such as entropy and margin to measure instance utility, however it fails to capture the data distribution information contained in the unlabeled data. This can eventually cause the classifier to select outlier instances to label Therefore, its important to consider the classification uncertainty as well as instances diversity in a population while developing an AL solution. In this paper, we present a *Budget Active Learning* (BAL) for Deep Networks a new robust AL method created by combining both uncertainty and correlation measure as an instance informativeness evaluation metric. An instance is selected based on its informativeness measure and then a budget annotator is used to label the instance. After each successful labeling the model selection strategy is updated with the new labeled data information. We perform various experiments on batched SVHN, CIFAR10 and plant-seedling-V2 datasets using Deep Networks models : Inception-V3, DenseNet and SqueezeNet.

The rest of the paper is organized as follows: Section **??** highlights related works. Section **??** presents our proposed Budget AL algorithm. Section **??** presents the experiments and results. Section **??** concludes the paper.

2 Literature Review

Successful investigations on ways to reduce labeling cost by use of AL has been going on for years now [**?**]. AL helps reduce the training data by selecting the most informative instances to label for training the model [**?**]. In a typical AL method, learning proceeds sequentially, while actively querying the labels of some instances from the membership queries. In AL there are three scenarios in which the ML algorithm will query the label of an instance, they include: a) *Membership Query Synthesis* that generates constructs of an instance from underlying distribution [**?**]; b) *Stream-Based Selective Sampling* that uses query strategy to determine whether to query the label of an instance or reject it based on informativeness [**?**]; c) *Pool-Based Sampling* that uses instances that are drawn from a pool of unlabeled data according to some informativeness measure [**?**]. Many recent works focuses on use of pool-based sampling approach. The aim is to query labels of the most informative instances, consequently reducing labeling costs and accelerating the learning.

In recent time, there are a number of works focusing on AL strategies to reduce the labeling cost. Yang *et al.* (2017) defines in [**?**] ways to segment biomedical images by combining fully convolutional network and AL to reduce annotation effort by making suggestions on the most effective annotation areas. In their approach, the network is

used to provide uncertainty and similarity information which is used to evaluate the most informative areas for annotation. Sener *et al.* (2017) in [**?**] defines AL as a core-set selection problem by choosing a set of points that the model can use to learn in a batch setting environment. A geometrical method is used to characterize the performance of the selected subset. Wanh *et al.* (2016) in [**?**] introduced a framework for updating the feature representation and the classifier simultaneously. A sample selection strategy is used to improve the classifier performance while reducing the manual annotation. Huang *et al.* (2018) in [**?**] uses fine tuned pretrained model on most useful examples. The examples are estimated based on potential contribution of an instance to feature representation. Iscen *et al.* (2019) in [**?**] introduces a transductive method that uses nearest neighbor graph to make predictions for generating pseudo-labels of the unlabeled dataset. Wang *et al.* (2014) in [**?**] combines AL and transfer learning into a Gaussian process based approach, and sequentially uses predictive covariance to select most suitable queries from the target domain. In their study Kale and Liu in [**?**] uses a combination of AL and Transfer Learning to learn labeled data from source domain for classification in target domain. Kale *et al.* (2015)in [**?**] introduces a framework for generating effective label queries by performing transfer learning. The framework is able to perform both the un-supervised and semi-supervised learning. Cai *et al.* (2019) in [**?**] defines online video recommendation as a multi-view AL problem and they proposed a framework to learn the mapping from visual view to text view. In their work Joshi *et al.* (2009) proposes an uncertainty measure that generalizes margin based uncertainty to the multi-class [**?**]. Chakraborty *et al.* (2011) propose a dynamic-batch-mode-AL combined with selection criteria as a single formulation [**?**].

The conventional way to reduce the cost of designing Deep Learning model and optimizing its parameters is by exploiting available pretrained models. Use of pretrained models trained on large benchmark dataset can helps reduce the training cost by utilizing the learned information. This is also referred to as *Transfer Learning* (TL) [**?**]. In TL, instead of starting the learning process from randomly initialized model weights, learning starts from patterns that have been learned when solving a different problem. This way there is leverage on previous learnings.The information transfer between the source and the target domain is done through feature sharing [**?**] and components transformation [**?**]. Performing batch training with gradient descent optimization helps address the challenge of limited computing power in deep learning. However, it is not possible to train Deep Networks efficiently with large training set. To overcome this challenge, a mini-batch gradient descent is performed by splitting the training set into smaller sets and gradient descent is implemented on each of the batches. This approach make training more faster and efficient. Classical state-of-the-art Deep Network models include: AlexNet [**?**], NIN [**?**], ENet [**?**], ZFNet [**?**], GoogLeNet [**?**] and VGG 16 [**?**]. Modern models include: Inception [**?**], ResNet [**?**], and DenseNet [**?**]. These networks have achieved impressive performance on computer vision, speech and text recognition with effective representations for visual objects [**?**].

From the literature presented, recent AL works focus on selecting a single informative unlabeled instance to label using uncertainty metrics. One main shortcoming of the above approaches is poor generalization for unseen instances in the domain. This is due to the fact that they only select queries based on how the instance related to the

classifier while ignoring unlabeled instances. Also with a large set of instances classification response time can be slow, therefore use of budget annotator will help reduce active selection and labeling time.

3 AL WITH BUDGET ANNOTATION

In this section, we first describe the general AL algorithm, then we introduce our algorithm detailing each component. We will use the following notation in this paper. Let x represents instances and y represents labels, $D = D^L \cup D^U$, D^L denotes labeled instances where $D^L = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, D^U denotes unlabeled instances where $D^U = \{(x_1, ?), (x_2, ?), ..., (x_n, ?)\}$, D^H denotes high confidence instances and Θ denotes the model defined by model parameters. For label space L^S with m classes in D the label of D^U can be expressed as $y_i = l, l \in \{1, 2, ..., m\}$. Therefore, instance selection criteria in this study will be based on probability of x_i belonging to l^{th} class which can be expressed as:

$$p(y_i = l|x_i; \theta) \tag{1}$$

where θ denotes the CNN network weights and Equation **??** denotes the network softmax output for l^{th} class.

Fu *et al.* presents a survey on instance selection and introduces in [**?**] an inefficient general AL algorithm for *Deep Networks*. We present in Algorithm **??** a new generalized form of AL. From lines 4 to 10, the model is iteratively defined according to a budget m.

Algorithm 1: General Active Learning.

1 **Input:** labeled instance set D^L, unlabeled instance set D^U, a budget m;
2 **Output:** a model Θ;
3 $\Theta \leftarrow$ `getModel`(D^L);
4 **while** $|D^L| < m$ **do**
5 $\quad D^U \leftarrow D \setminus D^L$;
6 $\quad$ **for** *each* x_i *in* D^U **do**
7 $\quad\quad u_i \leftarrow u(x_i, \Theta)$;
8 $\quad x^* \leftarrow \underset{i}{\text{argmax}}(u_i)$;
9 $\quad D^L \leftarrow D^L \cup \{x^*\}$;
10 $\quad \Theta \leftarrow$ `getModel`(D^L);
11 **return** Θ;

Figure **??** describes the conceptual representation of our method. The method progressively get data as input from the unlabeled set. While on initial model parameters, the most informative instance is selected from the unlabeled set for labeling by the Budget Annotator. On sucessful selection and labeling the labeled instance is added to the

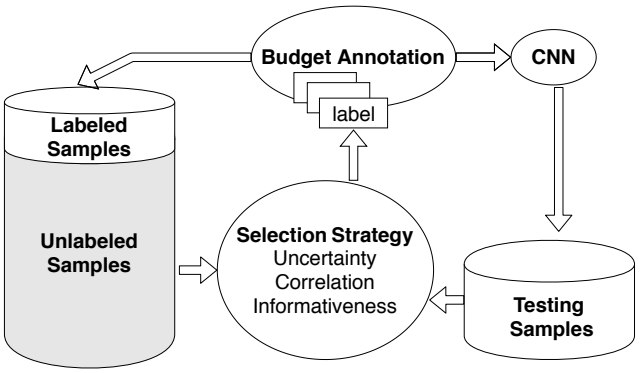


Fig. 1. BAL conceptual representation

training set and the model selection strategy is simultaneously updated and validated. Most informative samples and the classified samples are applied on the classifier output. The process to select and label instances will iterate until the budget is achieved while simultaneously updating the selection strategy.

In order to avoid the problem of generalization of unseen instances and to learn an accurate model, we present a robust approach by combining the strengths of different learning strategies. An AL annotator use evaluation metrics to compute the instance utility in order to select the most appropriate instance to label. The utility metrics considered in this work are uncertainty, correlation and informativeness measure, thus we present four main components: a) an uncertainty measure, b) correlation measure, c) an informative measure and d) and budget annotator.

3.1 Uncertainty Measure

Given a label space L^S the uncertainty measure f_u of a sample (features & label) can be defined as:

$$f_u(x) : \begin{cases} L^S \rightarrow R, & \text{(i) features view} \\ (D^U \times L^S) \rightarrow R, & \text{(ii) features-label view} \end{cases} \tag{2}$$

to a real number space R. From Equation **??**, (i) the uncertainty measure is computed from sample features only while (ii) the measure is computed from both the sample features and label. In our method we consider the uncertainty measure computed from sample features and label view which is considered the most effective [**?**]. Out of the three common uncertainty measure criteria namely least confidence, sample margin and entropy, we considered sampling margin since it integrates the second most probable class label in the uncertainty metric hence able to reduce the error rate by defining the decision boundary. We therefore define uncertainty measure as:

$$f_u(x_i) = p(y_i = l_1|x_i;\theta) - p(y_i = l_2|x_i;\theta) \tag{3}$$

High uncertainty value f_u implies current model have little knowledge of the instance, and including it into the training set can help improve the prediction performance of the model.

3.2 Correlation Measure

When developing efficient AL methods, its is critical to consider samples distribution information [**?**]. The instance diversity information aids in selecting most representative instances. In order to have more information about the unlabeled instances its appropriate to select a candidate instance in a more dense region. In addition, selecting an instance to label only based on uncertainty measure may lead to redundancy, therefore exploiting sample instance diversity will provide an optimal instance to label. Our method is based on the fact that the trade off between instance uncertainty and correlation is an essential AL problem to address. Given a label space L^S, we can define different groups of correlation of an instance x in a set of unlabeled set as;

$$f_c(x) : \begin{cases} D^U \times D^U \rightarrow R, & \text{feature view} \\ L^S \times L^S \rightarrow R, & \text{label view} \\ (D^U, y) \times (D^U, y) \rightarrow R, & \text{combined view} \end{cases} \tag{4}$$

to a real number space R. In Equation **??**, the combination of feature and label correlation is called combined view. Different algorithms exist for exploiting this type of combination [**?**]. Majorly these algorithms are used in a multi-label learning tasks when an instance has more than one label. This setting is ideal for mining tasks on instances with complex structure. On our work we focus on exploiting the pairwise similarities of instances, therefore the informativeness of an instances is weighed by average similarity to its neighbours. Let x_i and x_j be a pair of instances. To cope with the drawback of uncertainty based selection, we then consider the diversity by evaluating the correlation of the instances. Given a label space L^S the correlation measure $f_c(x_i, x_j)$ between a pair of instances in a sample x_i and x_j can be defined as:

$$f_c(x_i) = \frac{1}{D^U} \sum_{x_j \in D^U} (x_i, x_j) \tag{5}$$

The value of $f_c(x_i)$ represents the instance density of x_i in the unlabeled set. The larger the value, the more densely an instance is correlated with others. A low value of the correlation measure indicates an outlier instance which should not be considered for labeling.

3.3 Informativeness Measure

Our motivation is that the most representative instances of a distribution can be very informative for improving the generalization performance. Therefore, given correlation measure $f_c(x_i)$ and uncertainty measure $f_u(x_i)$ the informativeness of an instance can be defined as:

$$f_i(x) = f_u(x_i) \times f_c(x_i) \tag{6}$$

It can be rewritten as:

$$x^* = \underset{i}{\operatorname{argmax}}(u_i.c_i) \tag{7}$$

3.4 Instance evaluation and Budget Labeling

Instance evaluation is based on the instance informativeness in a set. In our method we use query by a single model evaluation learned from the training set. The model is trained on all labeled instances: feature and label views. After quering for an unlabeled instance, a model prediction result is generated based on output probability distribution. Each instance $x_i = \{f_1^i, f_2^i, ...f_q^i, y^i\}$ in labeled set $D^L = \{x_1, x_2, ...x_s\}$ is represented in a feature space F consisting of a feature space and its class label y^i. The size of D^L is denoted by s and x_i denoted the ith instance in D^L. The prediction can be denoted as a mapping function from the feature space F to the class label space Y which can be expressed as;

$$p(x) : F \mapsto Y \tag{8}$$

The query strategy used in this work is based on the value of f_i discussed in equation 6. Instances are ranked based on the value f_i with top ranked instances being the most appropriate to label. Budget annotator is used to pick classes which has maximum predicted probability as if they were true labels. For CNN implementation we use entropy regularization, this way we are able to separate low density between classes. High confidence samples from D^H are selected and then assign predicted labels to them. For l^{th} category we define the budget label y_i as follows;

$$y^* = \underset{i}{\operatorname{argmax}}(p(y_i = l|x_i; \theta_{x,y})) \tag{9}$$

Under the current distribution $p(y_i = l|x_i; \theta)$ each possible instance $(x_1, ?)$ from the selected instances D^H will be labeled with label y_i. When $y_i = 1$, x_i is regarded as a high confidence sample. The model update strategy is to learn a model based on the information provided by model weights computed from model validation of the performance. The Algorithm **??** describes the *Budget Active Learning* (BAL) with budget annotation.

BAL is designed to train a classification model using a small labeled population sample proportion. At first the BAL is trained using the initial set of labeled data D^L, using the initial weights for pretrained models and random initialized weights for untrained models. In Algorithm **??**, the labeling is defined by the budget m with model updates after each iteration (lines 4-16). Instance evaluation is done to identify the most informative and representative instance to label (lines 5-8). This evaluation returns the high confidence instances D^H selected from the unlabeled population (lines 10-12). For each of the selected instance, its label is queried and consequently the labeled set is updated. The model selection strategy is updated with the learned parameters after every iteration.

Algorithm 2: Efficient Budget Active Learning (BAL).

Input: labeled instance set D^L, unlabeled instance set D^U, a budget m;
Output: model Θ;
$\Theta \leftarrow$ `getModel` (D^L);
while $|D^L| < m$ **do**
  for *each* x_i *in* D^U **do**
    $u_i \leftarrow f_u(x_i)$;
    $c_i \leftarrow f_c(x_i)$;
  $x^* \leftarrow \underset{i}{\operatorname{argmax}}(u, c)$;
  $D^H \leftarrow \emptyset$;
  for *each* i *in* D^U **do**
    $x \leftarrow \underset{i}{\operatorname{argmax}} f_i(x)$;
    $D^H \leftarrow D^H \cup \{x\}$;
  $y_i \leftarrow$ `getLabel`(D^H);
  $D^U \leftarrow D^U \setminus \{y_i\}$;
  $D^L \leftarrow D^L \cup \{y_i\}$;
  $\Theta \leftarrow$ `getModel` (D^L);
return Θ;

4 EXPERIMENTS

To examine the efficiency of the proposed algorithm, we have considered public available datasets and state-of-the-art models.

4.1 Datasets

Three public available datasets namely CIFAR10 [**?**], Street View House Numbers (SVHN) [**?**] and plant-seedling-V2 [**?**] datasets are used. The statistical information of the datasets are summarized in Table **??**. For large datasets (CIFAR10 and SVHN), in regards to their size, we split the data into two sets; 20% as labeled set and 80% as unlabeled set. Half of the labeled set is randomly sampled as the training set, and the remaining as the validation set. The testing samples for each of the dataset is as shown in the table. For the other dataset (plant-seedling-V2), due to its very small size, 40% was randomly sampled as labeled set and 70% as the unlabeled set. In both cases, we tried to minimize the size of the training data, in order to demonstrate the efficiency of our budget AL method. For all data input, resize and normalize transformation was done in order to match the models input sizes and shapes.

4.2 Fine-tuning Network Parameters

In order to suite the pretrained network to the dataset classes, the last layer (softmax layer) is truncated and replaced with a layer that matches the dataset classes. Back

Table 1. Selected datasets used in this work.

Data	# instance	# label	# train	# validation	# testing
CIFAR10 [**?**]	50k	10	5k	5k	10k
SVHN [**?**]	73k	10	7k	7k	26k
plant-seedling-V2 [**?**]	6539	12	1k	807	807

propagation is performed to fine-tune the pretrained weights. 10 model updates were carried out with a training batch size of 32 and and a learning rate of 0.05. The number of model updates is sufficient to demonstrate the classification performance and efficiency of our method over the other methods. The training rate is carefully considered to ensure a good training stability and generalization is achieved.

4.3 Models

Table **??** presents six state-of-the-art Deep Networks models that have comparative few model parameters (M) expressed in million. While Deep Networks provide state-of-the-art prediction accuracy to many Machine Learning tasks, it comes at a high computational cost [**?**]. Model with more parameters (i.e. bigger networks and learnable parameters) is slower than a model with less parameters. The experiments were done using three of these models which have achieved best performance in ILSVRC and have lower parameters number (Inception-V3, DenseNet-169 and SqueezeNet). Instead of only training an entire CNN from scratch (with random initialization) we considered also transfer learning in order to leverage the training and then use ConvNet as an initialization and a fixed feature extractor for the task. In our experiments, we have used pretrained models given by Pytorch v1.3.0. In this section we will briefly discuss the architectures of the selected models.

Table 2. Deep Networks models comparison on ImageNet [**?**].

Model	Input Size	M	Top-1 Acc (%)	Top-5 Acc (%)
Inception-V1 [**?**]	224x224	5	70	90
Inception-V2 [**?**]	224x224	5	74	92
Inception-V3 [**?**]	**299x299**	**24**	**78**	**94**
Inception-V4 [**?**]	299x299	46	80	95
DenseNet-169 [**?**]	bf224x224	**14**	**76**	**93**
SqueezeNet [**?**]	**224x224**	**3**	**68**	**88**

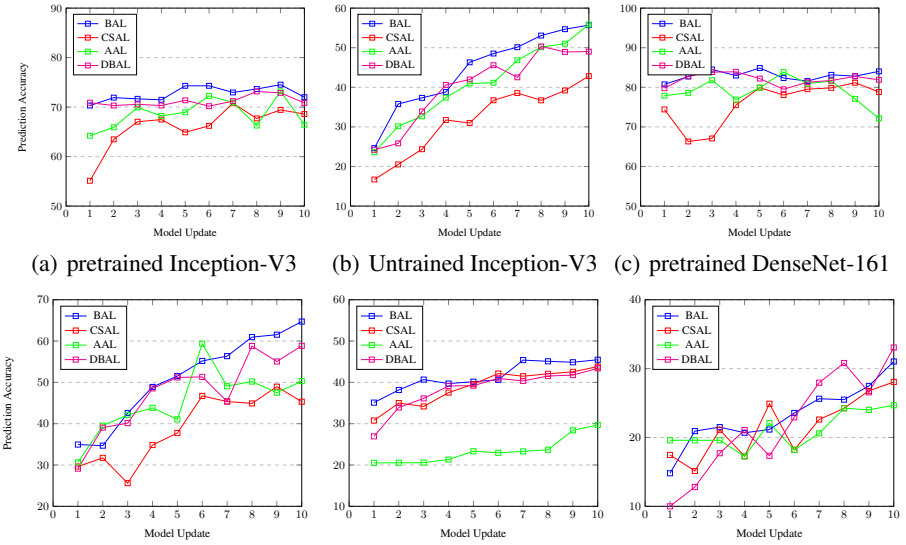
GoogLeNet, a 2014 ILSVRC winner, was inspired by LeNet but implemented a novel Inception module. Their Inception module performs series of convolutions at different scales and subsequently concatenate the results. The module is built with several small convolutions. There has been tremendous efforts done to improve the performance of this architecture: a) Inception-V1 [**?**] has 3 different sizes of filters (1x1, 3x3, 5x5) and max pooling. The outputs are concatenated and sent to the next Inception module; b) Inception-V2 [**?**] and Inception-V3 [**?**] factorize 5x5 convolution to two 3x3 convolution operations to improve computational speed. A 5x5 convolution is 2.78 times costly than a 3x3 convolution; stacking two 3x3 convolutions leads to a boost in performance; c) In Inception-V4 and Inception-ResNet the initial set of operations were modified before introducing the Inception blocks. The figures **??**, **??** and **??** show the prediction accuracy comparison between our approach and other baseline methods on previously cited models.

When Deep Networks start converging, then degradation become apparent challange to performance. Thats means as the network depth increases, the accuracy gets saturated and then degrades rapidly. Deep Residual Neural Network (ResNet), a logical extension of DenseNet [**?**] created by Kaiming *et al.* [**?**] introduced a novel architecture with insert shortcut connections. The connections turns the network into a residual network. This was a breakthrough which enabled the development of much deeper networks. The residual enables the network learn to adjust the input feature map. Following this intuition the authors proposed a pre-activation variant using the insert shortcut connections by the gradients flowing through the shortct connections to the earlier layes unimpended. Each ResNet block is either 2 layer deep or 3 layer deep. It achieves a top-5 error rate of 3.57% which beats human-level performance. DenseNet which is a logical extension of ResNet, brings improved efficiency by concatenating each layer feature map to every successive layer within a dense block [**?**]. This enables feature reuse within the network by allowing later layers within the network to directly leverage the features from earlier layers. Now the feature-maps of all preceding layers can be used as inputs, and its own feature-maps can be used as inputs into all subsequent layers, this helps alleviate the vanishing-gradient problem, feature reuse and consequently reduce number of parameters.

In recent times with use of Internet of Things and Cloud Computing, there is constant communication between the servers and the clients. This brings a need for a smaller sized model with similar or improved efficiency as the state of the art models. SqueezeNet [**?**] achieves AlexNet-level accuracy with 50x fewer parameters [**?**]. Additionally with model compression technique one can achieve 510 times smaller than AlexNet compression. In order to reduce the number of parameters by 9 times, a 3x3 filters are replaced with 1x1 filters. Subsequently, number of input channels is reduced to 3x3 filters. Finally, the feature map is down-sampled in order to have larger activation maps.

4.4 Results

The proposed approach was implemented on NVIDIA Tesla P100 GPU. Using few model update iterations, our method demonstrates impressive prediction accuracy over the other Deep AL methods. In the experiments losses and accuracies per model update



(a) pretrained Inception-V3 (b) Untrained Inception-V3 (c) pretrained DenseNet-161

(d) Untrained DenseNet-161 (e) pretrained SqueezeNet-V1 (f) Untrained SqueezeNet-V1

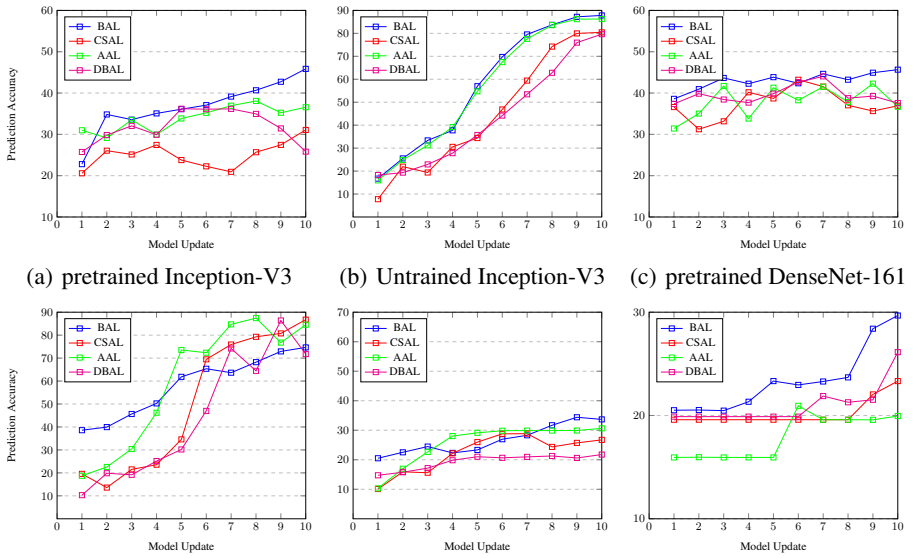
Fig. 2. Prediction comparison on CIFAR10 dataset.

were monitored while comparing the following Deep Active Learning baseline methods:

- **Budget AL (BAL)**: our proposed method.
- **Core-Set AL (CSAL)**: method proposed in [**?**] which defines the AL problem as a competitive sample core-set selection which is then applied to a CNN in a batch setting.
- **Deep Bayesian Active Learning (DBAL)**: a Bayesian framework proposed in [**?**] for high dimensional data which considers Deep Learning problem of dependence on big amount of data.
- **Adversarial AL for Deep Networks (AAL)**: a margin based approach proposed in [**?**] for Deep Networks with intention of reducing the number of queries to the oracle during training.

Impressive performance is recorded by the methods on the pretrained models as compared to the un-trained models. In general, from the results the pretrained DenseNet and Inception models on CIFAR10 leverage much better than SqueezeNet on same dataset. This means that the model weights for DenseNet and Inception model leverage better that those of SqueezeNet to this type of dataset. On all the training instances, BAL performs better than all other baseline active learning methods as shown in Figure **??**. On the un-trained models the prediction performance seem to edge up as the model selection strategy gets updated.

On SVHN dataset, all Deep Active Learning methods performs poorly except on untrained DenseNet and Inception models. The performance on these models improves after the fifth model update. This is so because initially the models weights do not



(a) pretrained Inception-V3 (b) Untrained Inception-V3 (c) pretrained DenseNet-161

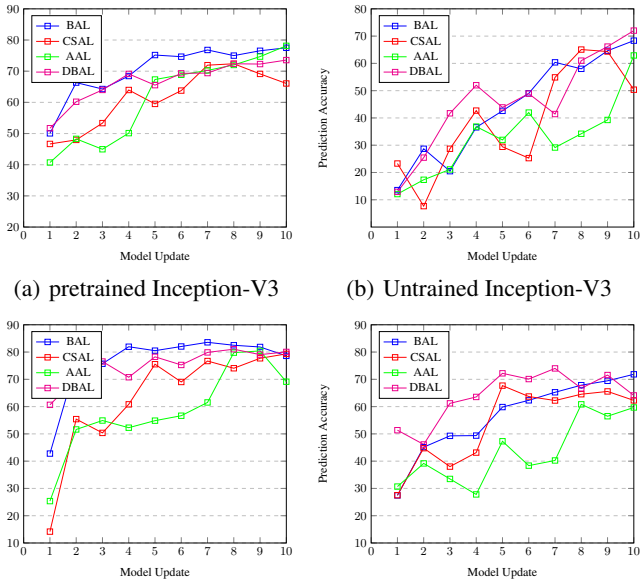
(d) Untrained DenseNet-161 (e) pretrained SqueezeNet-V1 (f) Untrained SqueezeNet-V1

Fig. 3. Prediction comparison on SVHN dataset.

perform well with this dataset but after several self tuning there is improved prediction accuracy. Following the poor performance exhibited in SquezeNet on both CIFAR10 and SVHN data, we did not conduct experiments with SqueezeNet on the plant-weed detection problem.

Plant Weed Detection Agriculture is a critical for human survival and it remains a major driver of many economies around the world. With increase demand for food and other agricultural production challenges, there is sure need to improve on production output. Current agricultural machine vision solutions are faced with accurate and reliable large scale weed detection. In this section we present a plant seedling weed detection problem using a plant-seedling-V2 dataset [**?**]. The plant-seedling dataset contains 6539 images from 960 RGB unique seedlings of plants belonging to 12 species at different growth stages with a physical resolution of 10 pixels/mm. Because of small dataset available, 15% of the set was used for training our algorithm for weed identification and 12% used for validation, the rest of the data used as unlabeled dataset. In addition, in an effort to avoid overfitting the convolutional base of the networks was frozen and its output used in the classifier.

In Figures 4(a) and 4(c) we compare our method with other Deep Active Learning methods in both pretrained Inception-V3 and DenseNet-161 models on plant-seedling dataset. The results indicate the efficiency of our method as compared to performance of other Deep AL methods discussed. Our method is able to adapt better with the pretrained parameters and quickly provide better prediction. Figures 4(b) and 4(d) show the performance on the untrained versions of the models on the same dataset. Using



(a) pretrained Inception-V3 (b) Untrained Inception-V3

(c) pretrained DenseNet-161 (d) Untrained DenseNet-161

Fig. 4. Prediction comparison on plant-seedling-V2 dataset.

the initial pretrained parameters to initialize the models yields to better prediction accuracy quickly within few model updates. The main results are shown in Figures **??**. and **??**. Overall, BAL (in blue line) is able to outperform other Deep AL methods on major datasets including the plant-seedling dataset. By comparing our method with other methods, we notice an apparent increase in classification accuracy which indicate that using both instance uncertainty and correlation measure is more efficient. BAL is able to pick the most representative candidate point from the unlabeled population. In addition, from the plant-seedling shown in Figure **??**, we observe the superiority of our method tends to be more obvious even when the number of instances is small. Its clear that our method can generalize better than other discussed methods by selecting the most representative instances with only few queries.

5 CONCLUSION

The main objective of AL is to label the most informative instance in order to achieve high prediction accuracy with minimum cost. Use of AL in recent technologies is an active research area with efforts to improve on the prediction accuracy while using less data. In this paper, we propose a BAL method for cost-effective training of Deep Networks. Instead of training from scratch with random initialization, a pretrained model parameters can be used to initialize a model to a new target task by fine tuning with a few actively queried examples, thus significantly reducing the cost of designing the network architecture and cost of labeling a large training set. Using BAL, classification

task was able to record 85% prediction accuracy quickly using fairly small amount (10 to 20% of data) of data as training data as compared to conventional AL methods on Deep Network Models. In the future, we plan to apply the approach on more real life datasets and more pretrained models with variational dataset. In addition, the feature transformation on each layer will be further studied while considering different types of data input.

References

1. Weber, R. H., & Weber, R. Internet of things (Vol. 12). Heidelberg: Springer (2010).
2. Ray, A. S. Remote sensing in agriculture. International Journal of Environment, Agriculture and Biotechnology, 1(3) (2016).
3. Jinbo, C., Xiangliang, C., Han-Chi, F., & Lam, A. (2019). Agricultural product monitoring system supported by cloud computing. Cluster Computing, 22(4), 8929-8938.
4. Chi, M., Plaza, A., Benediktsson, J. A., Sun, Z., Shen, J., & Zhu, Y. (2016). Big data for remote sensing: Challenges and opportunities. Proceedings of the IEEE, 104(11), 2207-2219.
5. Chen, M., Mao, S., & Liu, Y. (2014). Big data: A survey. Mobile networks and applications, 19(2), 171-209.
6. Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. Nature, 542(7639), 115.
7. Long, M., Zhu, H., Wang, J., & Jordan, M. I. (2017, August). Deep transfer learning with joint adaptation networks. In Proceedings of the 34th International Conference on Machine Learning-Volume 70 (pp. 2208-2217). JMLR. org.
8. Gal, Y., Islam, R., & Ghahramani, Z. (2017, August). Deep bayesian active learning with image data. In Proceedings of the 34th International Conference on Machine Learning-Volume 70 (pp. 1183-1192). JMLR. org.
9. Cohn, D., Atlas, L., & Ladner, R. (1994). Improving generalization with active learning. Machine learning, 15(2), 201-221.
10. Settles, B. (2009). Active learning literature survey. University of Wisconsin-Madison Department of Computer Sciences.
11. Angluin, D. (1988). Queries and concept learning. Machine learning, 2(4), 319-342.
12. Zhu, X., Zhang, P., Lin, X., & Shi, Y. (2007, October). Active learning from data streams. In Seventh IEEE International Conference on Data Mining (ICDM 2007) (pp. 757-762). IEEE.
13. Nigam, K., & McCallum, A. (1998). Pool-based active learning for text classification. In Conference on Automated Learning and Discovery (CONALD).
14. Yang, L., Zhang, Y., Chen, J., Zhang, S., & Chen, D. Z. (2017, September). Suggestive annotation: A deep active learning framework for biomedical image segmentation. In International conference on medical image computing and computer-assisted intervention (pp. 399-407). Springer, Cham.
15. Sener, O., & Savarese, S. (2017). Active learning for convolutional neural networks: A core-set approach. arXiv preprint arXiv:1708.00489.
16. Wang, K., Zhang, D., Li, Y., Zhang, R., & Lin, L. (2016). Cost-effective active learning for deep image classification. IEEE Transactions on Circuits and Systems for Video Technology, 27(12), 2591-2600.
17. Huang, S. J., Zhao, J. W., & Liu, Z. Y. (2018, July). Cost-effective training of deep cnns with active model adaptation. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 1580-1588). ACM.

18. Iscen, A., Tolias, G., Avrithis, Y., & Chum, O. (2019). Label propagation for deep semi-supervised learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 5070-5079).
19. Wang, X., Huang, T. K., & Schneider, J. (2014, January). Active transfer learning under model shift. In International Conference on Machine Learning (pp. 1305-1313).
20. Kale, D., & Liu, Y. (2013, December). Accelerating active learning with transfer learning. In 2013 IEEE 13th International Conference on Data Mining (pp. 1085-1090). IEEE.
21. Kale, D., Ghazvininejad, M., Ramakrishna, A., He, J., & Liu, Y. (2015, June). Hierarchical active transfer learning. In Proceedings of the 2015 SIAM International Conference on Data Mining (pp. 514-522). Society for Industrial and Applied Mathematics.
22. Cai, J. J., Tang, J., Chen, Q. G., Hu, Y., Wang, X., & Huang, S. J. (2019). Multi-view active learning for video recommendation. Proceedings of IJCAI-19, Macao, China, https://www. ijcai. org/proceedings/2019/0284. pdf.
23. Joshi, A. J., Porikli, F., & Papanikolopoulos, N. (2009, June). Multi-class active learning for image classification. In 2009 IEEE Conference on Computer Vision and Pattern Recognition (pp. 2372-2379). IEEE.
24. Chakraborty, S., Balasubramanian, V., & Panchanathan, S. (2011, June). Dynamic batch mode active learning. In CVPR 2011 (pp. 2649-2656). IEEE.
25. Pan, S. J., & Yang, Q. (2009). A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 22(10), 1345-1359.
26. Tzeng, E., Hoffman, J., Zhang, N., Saenko, K., & Darrell, T. (2014). Deep domain confusion: Maximizing for domain invariance. arXiv preprint arXiv:1412.3474.
27. Pan, S. J., Tsang, I. W., Kwok, J. T., & Yang, Q. (2010). Domain adaptation via transfer component analysis. IEEE Transactions on Neural Networks, 22(2), 199-210.
28. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).
29. Lin, M., Chen, Q., & Yan, S. (2013). Network in network. arXiv preprint arXiv:1312.4400.
30. Paszke, A., Chaurasia, A., Kim, S., & Culurciello, E. (2016). Enet: A deep neural network architecture for real-time semantic segmentation. arXiv preprint arXiv:1606.02147.

31. Zeiler, M. D., & Fergus, R. (2014, September). Visualizing and understanding convolutional networks. In European conference on computer vision (pp. 818-833). Springer, Cham.
32. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

33. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
34. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).
35. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
36. Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).
37. Gikunda, P. K., & Jouandeau, N. (2019, July). State-of-the-Art Convolutional Neural Networks for Smart Farms: A Review. In Intelligent Computing-Proceedings of the Computing Conference (pp. 763-775). Springer, Cham.

38. Fu, Y., Zhu, X., & Li, B. (2013). A survey on instance selection for active learning. Knowledge and information systems, 35(2), 249-283.
39. Huang, S. J., Gao, N., & Chen, S. (2017, August). Multi-instance multi-label aInformation regularization with partially labeled datactive learning. In IJCAI (pp. 1886-1892).
40. Szummer, M., & Jaakkola, T. S. (2003). . In Advances in Neural Information processing systems (pp. 1049-1056).
41. Krizhevsky, A., & Hinton, G. (2010). Convolutional deep belief networks on cifar-10. Unpublished manuscript, 40(7), 1-9.
42. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., & Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning.
43. Giselsson, T. M., Jrgensen, R. N., Jensen, P. K., Dyrmann, M., & Midtiby, H. S. (2017). A public image database for benchmark of plant seedling classification algorithms. arXiv preprint arXiv:1711.05458.
44. Sze, V., Chen, Y. H., Yang, T. J., & Emer, J. S. (2017). Efficient processing of deep neural networks: A tutorial and survey. Proceedings of the IEEE, 105(12), 2295-2329.
45. Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.
46. Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and¡ 0.5 MB model size. arXiv preprint arXiv:1602.07360.
47. Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255). Ieee.
48. Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017, February). Inception-v4, inception-resnet and the impact of residual connections on learning. In Thirty-First AAAI Conference on Artificial Intelligence.
49. Gholami, A., Kwon, K., Wu, B., Tai, Z., Yue, X., Jin, P., ... & Keutzer, K. (2018). Squeezenext: Hardware-aware neural network design. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 1638-1647).
50. Ducoffe, M., & Precioso, F. (2018). Adversarial active learning for deep networks: a margin based approach. arXiv preprint arXiv:1802.09841.