Budgeted Subset Selection for Fine-tuning Deep Learning Architectures in Resource-Constrained Applications

Subhankar Banerjee
Department of Computer Science
Florida State University

Shayok Chakraborty
Department of Computer Science
Florida State University

***Abstract*—The growing success and popularity of deep learning in computer vision have resulted in the availability of several pre-trained deep learning architectures (such as AlexNet, ResNet, VGGNet among others). A common practice in deep learning research is to use one of the pre-trained models and fine-tune it to a given target task, using training data from the target. However, training a deep learning model efficiently necessitates expensive, high-quality GPUs and distributed computing infrastructures. Some applications (such as those running on mobile platforms) are severely limited in terms of memory and computational resources; in these applications, it is a significant challenge to fine-tune a pre-trained deep learning model to a target task, using large amounts of target training data. Cloud services can be leveraged for training, but involve issues with data privacy and cost. In such applications, it is important to select an informative subset of the training data and fine-tune the deep model using only the selected subset. In this paper, we propose a novel framework to address this problem. We pose subset selection as a constrained NP-hard integer quadratic programming problem and derive an efficient linear relaxation to select a subset of exemplar instances. Our extensive empirical studies on three challenging vision datasets (from different application domains) using three commonly used pre-trained deep learning models corroborate the potential of our framework for real-world, resource-constrained applications.**

***Index Terms*—deep learning, subset selection**

I. Introduction

Deep neural networks have revolutionized the field of computer vision and have achieved state-of-the-art performance in a variety of applications, including image recognition [1], object detection [2] and image segmentation [3] among others. The unprecedented success of deep learning has inspired researchers to make their trained deep models publicly available to the community to promote further research using these models. Examples include the AlexNet [4], VGGNet [5] and ResNet [1], all of which are trained on the ImageNet dataset [6] and are publicly available. A common practice in deep learning research is to use one of these pre-trained models as the starting configuration and fine-tune it to the target application in question, using training data from the target, rather than developing a deep network from scratch for the target. This is mainly due to the following two reasons: (i) designing a suitable network architecture from scratch for a given target task requires extensive prior knowledge or domain expertise about the target, which may not be readily available; and (ii) tuning the hyper-parameters involves repeated trial and error, which can affect the learning performance. A pre-trained model, on the other hand, is already equipped with useful knowledge from its original task and comes with a set of optimized hyper-parameters, and thus addresses these issues.

In today's world, deep models with more and more hidden layers and trainable parameters are being successfully applied by researchers in a variety of applications. This exponential growth in the number of trainable parameters has necessitated the development of efficient hardware solutions, such as GPUs and distributed computing infrastructures to accelerate the training process [7]. However, for resource constrained applications (such as applications running on mobile platforms) using high-end GPUs and other computing infrastructures (multiple cores, clusters etc.) to train deep models may not be feasible. These applications are severely limited by the available memory, compute-power and other constraints of the mobile device. Cloud services like the Amazon Web Services, Microsoft Azure and Google Cloud Platform have been explored as an alternative option for training deep learning models. However, they often come with a hefty price tag, as the user needs to pay for the computational as well as storage resources. The data may also contain sensitive and confidential information, which can lead to potential security concerns if uploaded to the cloud. Some applications may require the models to be retrained frequently as new training data becomes available over time; it may thus be cheaper and more beneficial to train the model in a local machine, rather than in a cloud platform.

In situations like this, it is necessary to select an informative subset of the target training data and fine-tune the model using only the selected subset. In this paper, we propose a novel framework to address this problem. Specifically, we attempt to answer the following research question: *We wish to adapt a particular pre-trained deep model to a given target task. We are given a small amount of initial training data, which serves as our prior domain knowledge about the target. We are also given a large number (N) of training samples from the target for fine-tuning the model. However, due to computational and resource constraints, training on all the N samples is prohibitively expensive. We are given a parameter*

$k (k \ll N)$, which is the maximum allowable training set size (in addition to the initial training data), considering the resource constraints. Which k samples should we select from the set of N for fine-tuning, so that we derive a model with maximum generalization capability?

Our algorithm is easy to implement and computationally lightweight, corroborating its usefulness in resource-constrained applications. Although we study the performance of our algorithm on computer vision tasks in this paper, the proposed framework is generic and can be applied with any pre-trained network on any target task. The rest of the paper is organized as follows: we present a survey of related techniques in Section II; the details of our framework are presented in Section III; Section IV depicts the results of our empirical studies; and we conclude with discussions in Section V.

II. Related Work

Deep learning is responsible for many of the recent breakthroughs in AI, such as Google DeepMind's AlphaGo, autonomous cars, intelligent voice assistants among others. While deep neural networks (DNNs) have depicted commendable performance in several applications, training a deep network efficiently, to meet the requirements of practical applications, has proved to be a significant challenge. To address this issue, a variety of techniques have been explored with the goal of speeding up the training time of DNNs. We present a brief survey of these techniques in this section.

Importance Sampling: There is a body of literature on weighing or sampling training instances to speed up the training of deep models and achieve higher accuracy. The core idea is to perform importance sampling on training data points based on the gradient norm [8], loss [9], [10], bound on the gradient norm [11] or approximating the ideal sampling distribution [12]. Most of these techniques involve a large number of hyper-parameters, which are difficult to optimize [9], [10]. Our framework is computationally light and also fundamentally different from these techniques. For instance, the importance sampling scheme proposed in [11] first pre-samples a large batch of data points B uniformly from the given training samples; a subset b of desired batch size is then sampled according to an importance sampling distribution; the network parameters are then updated using stochastic gradient descent (SGD) and the whole process is repeated until convergence to obtain the final trained network. The algorithm requires access to the entire set of samples throughout the training process, which may be a significant challenge, especially in resource-constrained applications. In contrast, in our method, we attempt to select an exemplar subset of instances from the target data before even starting to train the deep model. The instances that are deemed irrelevant by the algorithm can be discarded straightaway and need not be considered at all during training, resulting in efficient usage of memory and computational resources.

Model Compression and Parameter Optimization: Multiple techniques have been proposed to compress large DNNs in order to accelerate the training time; these include distillation [13], weight precision reduction [14], [15], low-rank decomposition [16] and pruning [17], [18]. Some researchers have focused on improving prediction accuracy at a given parameter budget. Algorithms in this category include adoption of skip connections [1], replacement of fully-connected layers with global average pooling layers followed directly by the classifier layer [19], and depth-wise separable convolutions [20]. Cheng *et al.* [21] exploited the redundancy of parameters in deep neural networks by replacing the conventional linear projection in fully-connected layers with the circulant projection, which substantially reduces the memory footprint and enables usage of FFTs to speed up computations. In a related research, Mathieu *et al.* [22] used the FFT to accelerate the computation of convolutional layers, through the convolution theorem. In our method, we attempt to sample an informative set of instances without altering the model architecture or the training algorithm.

Other Related Techniques: Among other related techniques, Zhang *et al.* [23] proposed a hashing based scheme to address the vanishing gradient problem in back-propagation, which directly impacts the computational efficiency. Ioffe and Szegedy [24] introduced the batch normalization technique to normalize each training mini-batch which allowed the usage of much higher learning rates; when applied to state-of-the-art image classification models, this technique achieved the same accuracy with 14 times fewer training steps. Ko *et al.* [25] introduced a novel dropout technique called controlled dropout where a network is trained using compressed matrices of smaller size, resulting in significant training speed improvements.

Although these methods have depicted promising empirical results, most researchers today heavily rely on high-end GPUs and distributed computing infrastructures to train deep learning models efficiently [7], [26], [27]. For instance, Sun *et al.* [28] used 50 NVIDIA K80 GPUs and 17 parameter servers to train a deep learning model for ImageNet. Xie *et al.* [29] trained their deep network using SGD with mini-batch size of 256 on 8 GPUs (32 per GPU). Goyal *et al.* [30] trained the ResNet-50 model in 1 hour using a batch size of 8192, but assumed access to 256 powerful GPUs. Such powerful computational infrastructures are not available in limited memory and resource-constrained applications. To facilitate efficient adaptation of a given pre-trained deep model to a target task in such applications, it is necessary to select a subset of exemplar training samples from the target task and restrict the training of the deep model only to the selected subset. The size of the subset is governed by the memory, power and other available resources of the device running the application and can be assumed to be known apriori. Banerjee and Chakraborty [31] proposed a submodular optimization based framework to fine-tune deep learning architectures. To the best of our knowledge, this is the only prior research addressing this problem. Although the method depicted impressive performance, the performance was validated only in terms of accuracy; the computational overhead of the algorithm was not studied. In this research, we analyze the performance of our framework both in terms of accuracy and computational complexity and use [31] as one of our comparison baselines. We now describe our framework.

III. Proposed Framework

In our problem setup, we are given a model M^0, pre-trained on a particular dataset (such as the AlexNet model trained on ImageNet). Our objective is to adapt this model to a particular target task. We are given a small amount of target data L, which serves as our prior domain knowledge about the target (it is typically sampled at random from the given target data). We are further given a large set Z containing N target instances $\{x_1, x_2, \ldots, x_N\}$. However, due to resource constraints, we are not in a position to fine-tune M^0 using all the N instances. We are given a budget $k (k \ll N)$, which denotes the number of target samples that can be used to train the model (in addition to the initial labeled set L). Our objective is to select a subset of k instances from Z such that when M^0 is fine-tuned using L and the selected subset, it has the maximum generalization capability on the target data. Let C denote the number of classes in the target dataset.

To address this problem, we attempt to quantify the score of a subset of k samples and select the subset furnishing the maximal score. We compute the utility (usefulness from a classification perspective) of every sample in Z and propose to select the samples with the highest utilities in order to augment maximal information to M^0. However, if two samples x_i and x_j individually have high utilities, but are redundant, then we gain sub-optimal information by selecting both of them. We therefore include a diversity metric in our formulation to quantify the uniqueness of every pair of samples. A framework based on these two conditions ensures that the high-utility samples are selected for adapting the model and there is minimal overlap of information during model training.

Computing Utility: The utility of a sample x_i is computed by the classification uncertainty of the model M^0, trained on the initial set L. The class probabilities obtained from the last layer of the network can be used to define a variety of uncertainty metrics based on prediction confidence, margin and entropy [32]:

$$U_{confidence}(x_i) = 1 - P(y_1) \tag{1}$$

$$U_{margin}(x_i) = 1 - (P(y_1) - P(y_2)) \tag{2}$$

$$U_{entropy}(x_i) = -\sum_{y \in C} P(y|x_i) \log P(y|x_i) \tag{3}$$

where $P(y_1)$ and $P(y_2)$ denote the highest and the second highest class probabilities for the data point x_i respectively. We select entropy as the utility metric in this research due to its promising performance in a variety of machine learning applications [32]–[34]. A high value of entropy means high classification uncertainty. We compute a vector $u \in \Re^{N \times 1}$ where $u(i)$ denotes the entropy or utility of sample x_i in Z.

Computing Diversity: We compute a matrix $D \in \Re^{N \times N}$ where d_{ij} denotes the diversity between sample x_i and x_j. In this work, we used the gaussian kernel with parameter 1 to compute the diversity between a pair of samples (other distance metrics can be used, depending on the application). The $(i,j)^{th}$ entry in the matrix D is given by:

$$D(i,j) = \phi(x_i, x_j) \tag{4}$$

where $\phi = (.,.)$ denotes a kernel in the Reproducing Kernel Hilbert Space (RKHS). It is noted that computing D scales quadratically with N. The concept of random projections can be used to reduce the computational overhead. Random projections have been successfully used to speed up computations, where an original data matrix $A \in \Re^{m \times P}$ is multiplied by a random projection matrix $X \in \Re^{P \times p}$ to obtain a projected matrix $B \in \Re^{m \times p}$ in the lower dimensional space p:

$$B = \frac{1}{\sqrt{p}} AX \tag{5}$$

where $p \ll min(m, P)$. X is typically populated using the entries from the standard normal distribution $N(0,1)$, leading to many well-known theoretical results [35]. We leave the study of random projections for reducing the computational complexity for future research.

A. Subset Selection Framework

From our definitions in Equations (3) and (4), we note that both the vector u and the matrix D can only have non-negative entries. Given u and D, our objective is to select a subset of k samples which furnish maximum utility (given by the entries in u) and maximum diversity (given by the entries in D). We define a binary selection vector $x \in \{0,1\}^{N \times 1}$ where each entry corresponds to a particular sample in the set Z; if a particular entry in x is 1 (or 0), it means that the corresponding sample in Z will be selected (not selected) in the subset. Our subset selection problem (with subset cardinality k) can thus be posed as the following integer quadratic programming (IQP) problem:

$$\max_x u^T x + \lambda x^T D x$$

$$\text{s.t.} \quad x_i \in \{0,1\}, \forall i \quad \text{and} \quad \sum_{i=1}^{N} x_i = k \tag{6}$$

where λ is a weight parameter governing the relative importance of the two terms. Due to the binary integer constraints on x, we can combine u and D into a single matrix $Q \in \Re^{N \times N}$ and rewrite the optimization problem as follows:

$$\max_x x^T Q x$$

$$\text{s.t.} \quad x_i \in \{0,1\}, \forall i \quad \text{and} \quad \sum_{i=1}^{N} x_i = k \tag{7}$$

where the matrix Q with non-negative entries is constructed as follows:

$$Q(i,j) = \begin{cases} \lambda D(i,j), & \text{if } i \neq j \\ u(i), & \text{if } i = j \end{cases} \tag{8}$$

The binary integer constraints on x make this IQP NP-hard. We now derive an efficient linear programming (LP) relaxation to solve this NP-hard IQP.

B. An Efficient LP Relaxation

We define a binary matrix $W \in \Re^{N \times N}$ as $w_{ij} = x_i.x_j$. Clearly, each element in W is either 0 or 1, since x is a binary vector. With this definition, we can write the optimization problem in Equation (7) as:

$$\max_{x,W} \sum_{i,j} q_{ij} w_{ij}$$

$$\text{s.t.} \quad w_{ij} = x_i.x_j, \quad x_i \in \{0,1\}, \forall i \quad \text{and} \quad \sum_{i=1}^{N} x_i = k \tag{9}$$

The quadratic equality $w_{ij} = x_i.x_j$ implies that w_{ij} attains a value of 1 when both x_i and x_j are 1 and 0 otherwise. These conditions can be captured by the linear inequality $-x_i - x_j + 2w_{ij} \leq 0, \forall i,j$. From the inequality, it is easy to see that w_{ij} is 0 if either x_i, or x_j or both are equal to 0. When both x_i and x_j are 1, w_{ij} can be 0 or 1. However, the maximization condition in Equation (9) forces w_{ij} to be 1, since $q_{ij} \geq 0$. Thus, the value of w_{ij} obtained from the equality $w_{ij} = x_i.x_j$ and from the inequality $-x_i - x_j + 2w_{ij} \leq 0, \forall i,j$ is exactly the same for every possible values of x_i and x_j. We can therefore replace the quadratic equality with the linear inequality and simplify the problem in Equation (9) as:

$$\max_{x,W} \sum_{i,j} q_{ij} w_{ij}$$

$$\text{s.t.} \quad -x_i - x_j + 2w_{ij} \leq 0, \forall i,j$$

$$x_i, w_{ij} \in \{0,1\}, \forall i,j \quad \text{and} \quad \sum_{i=1}^{N} x_i = k \tag{10}$$

This is an integer linear programming (ILP) problem, as both the objective function and the constraints are linear in the variables. Thus, we have simplified an integer quadratic programming problem into an equivalent integer linear programming problem, which is much easier to solve. However, the binary constraints on x and W still pose computational challenges. To solve such a problem, a common strategy is to use the LP relaxation:

$$\max_{x,W} \sum_{i,j} q_{ij} w_{ij}$$

$$\text{s.t.} \quad -x_i - x_j + 2w_{ij} \leq 0, \forall i,j$$

$$x_i, w_{ij} \in [0,1], \forall i,j \quad \text{and} \quad \sum_{i=1}^{N} x_i = k \tag{11}$$

where the constraints have been relaxed, so that x and W are permitted to have values between 0 and 1. We solve the relaxed problem using an off-the-shelf LP solver and use the rounding technique (where we set the k largest entries as 1 and the remaining as 0) to derive the integer solution. However, there is no guarantee that the rounding technique will produce the optimal solution.

C. The Iterative Truncated Power Algorithm

To further improve the solution obtained using the LP relaxation, we use the iterative truncated-power algorithm [36]. This was originally proposed in the context of the sparse eigenvalue and the densest k-subgraph problems. Starting with an initial solution x_0, it generates a sequence of solutions $x_1, x_2, \ldots$. At each time step t, the solution x_{t-1} is multiplied by the matrix Q and the entries are truncated to zeros except for the k largest entries, which becomes the new solution x_t. The process is repeated until convergence. The algorithm has a guaranteed monotonic convergence for a positive semi-definite (psd) matrix Q. When the Q is not psd, the algorithm can be run on the shifted quadratic function (with a positive scalar added to the diagonal elements) to guarantee a monotonic convergence [36]. The algorithm benefits from a good starting point; we use the solution in Equation (11) as the initial solution x_0. It is also computationally efficient and converges fast. The pseudo-code for the proposed subset selection algorithm is given in Algorithm 1.

Algorithm 1 The Proposed Subset Selection Algorithm

```
Require: Pre-trained model $M^0$, Small initial training set $L$,
    Large ground set $Z$, subset size $k$, weight parameter $\lambda$
 1: Fine-tune $M^0$ using the set $L$
 2: Compute utility vector $u$ (Equation 3) and the diversity
    matrix $D$ (Equation 4)
 3: Compute the matrix $Q$, as described in Equation (8)
 4: Derive the initial solution $x_0$ by solving the relaxed LP
    problem in Equation (11) and rounding the solution
 5: t = 1
 6: repeat
 7:     Compute $x_t^{'} = Q.x_{t-1}$
 8:     Identify $F_t$ as the index set of $x_t^{'}$ with top $k$ values
 9:     Set $x_t$ to be 1 on the index set $F_t$ and 0 otherwise
10:     t = t + 1
11: until Convergence
12: Select a subset of $k$ samples from $Z$ based on the final
    solution $x_t$
13: Fine-tune $M^0$ using $L$ and the selected subset
```

The LP problem can be efficiently solved using a variety of LP solvers. Thus, our algorithm involves minimal computational overhead and is a promising candidate for resource-constrained applications.

IV. Experiments and Results

A. Datasets

We used three challenging datasets from different application domains to validate the performance of our algorithm:

MNIST: We used the MNIST dataset [37] (containing images of handwritten digits from 10 classes), which is extensively used in computer vision research, to study the performance of our framework.

SVHN: We also validated the performance of our framework on the SVHN dataset [38], which also contains 10 classes of digits obtained from house numbers in Google Street View images.

CIFAR-10: We further used the CIFAR-10 dataset [39], containing images of objects from 10 different categories under challenging real-world conditions, in our empirical study.

Our objective was to test the performance of our subset selection algorithm and not to outperform the best error rates on these datasets; we therefore did not follow the precise train / test splits specified for these datasets.

B. Deep Learning Models

We used three commonly-used convolutional neural network architectures (pre-trained models) to study the performance of our algorithm:

AlexNet: The AlexNet [4] model won the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) in 2012. It contains 5 convolutional layers and 3 fully connected layers.

GoogleNet: This is a 22-layer deep CNN which won the ILSVRC challenge in 2014 [40]. It is based on the idea of an inception module and contains significantly lesser number of trainable parameters than the AlexNet.

ResNet-18: This ResNet architecture contains 18 layers and is a variant of the original model with 152 layers, which won the ILSVRC in 2015 [1]. The core idea is the introduction of residual blocks, which facilitates training of very deep networks with much lower computational overhead.

The implementations were all performed in Matlab R2018a. The pre-trained models were obtained from the Matlab *Deep Learning Toolbox* [1]. The number of nodes in the last layer in each model was changed to equal the number of classes for each dataset. The L_2 regularizer was used for each of the models with regularization parameter 0.0005 for AlexNet, 0.001 for GoogleNet and 0.001 for ResNet. We used 0.001 as the initial learning rate for each model. The stochastic gradient descent with momentum (SGDM) was used as the optimizer and the models were trained for 30 epochs. The layer freezing technique [41] was employed to accelerate the training process. The weight parameter λ was taken as 0.3 based on preliminary experiments.

C. Comparison Baselines

While various subset selection criteria have been studied in the literature, optimizing almost all of them is, in general, NP-hard and non-convex [42], [43]. This has motivated the development and study of approximation methods for optimizing these criteria. We used two such commonly used methods as comparison baselines in our work:

DPP: Determinantal Point Process (DPP) is a sampling technique, which assigns a probability measure on the set of all subsets of a ground set, for approximately finding the maximum volume subset [44], [45]. DPP is an appealing tool for modeling diversity in applications such as document summarization, image search and ranking. We used the optimization strategy proposed in [44] for subset selection in this research.

Submodular: Submodular optimization techniques have been extensively used in data subset selection and use greedy approximation algorithms to maximize the value of a submodular function under a budget constraint [46]–[48]. Commonly used submodular functions include graph-cuts, facility location, diversity and coverage among others. We used the coverage and diversity criteria for subset selection and the standard greedy algorithm for optimization (as proposed in [31]), which is guarateed to produce a solution that is $1 - \frac{1}{e} \approx 63\%$ of the optimal.

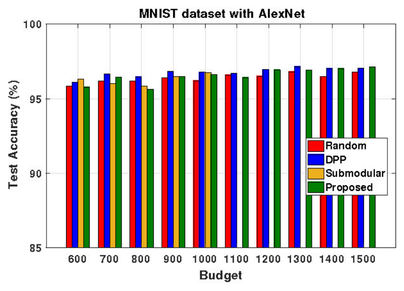
In addition, we also compared our method against **Random Sampling**, where a batch of samples is selected at random. Note that these are the commonly used dataset subset selection techniques and were hence used as baselines in this research. Other methods, such as *Lasso, ILFS,UDFS* have focused on feature selection [49], rather than data subset selection and were thus not included.

D. Subset Selection Performance

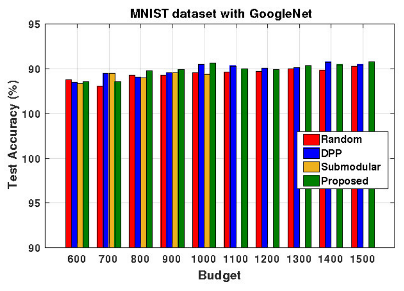
The goal of this experiment was to validate the performance of the proposed subset selection framework. For a given initial training set L and a given ground set Z, we used a parameter k, which denotes the available budget, that is, the number of samples from the ground set that can be selected for fine-tuning the deep models. Once a subset of target samples was selected, the deep models were trained on L and the selected subset and their performance was evaluated on a held out test set. The process was repeated afresh for each value of k. In our empirical study, we used 500 samples as the initial training set ($|L| = 500$), $3,000$ samples as the ground set ($|Z| = 3,000$) and $10,000$ samples as the test set. The samples were selected at random from each dataset. We used ten values of k, from 600 to $1,500$ in steps of 100, to study the performance under different budgets.

Accuracy: The results are depicted in Figure 1. In each figure, the x-axis denotes the specified budget and the y-axis denotes the classification accuracy on the test set. We first note that the test accuracy depicts a general increasing trend with an increase in budget, which is intuitive. *Random Sampling* sometimes depicts good performance, but is not consistent in its performance across datasets, models and subset sizes. Out of the 90 experiments (3 datasets x 3 models x 10 subset sizes), *Random Sampling* achieves the highest accuracy only 6 times. Our framework outperforms *Random Sampling* in

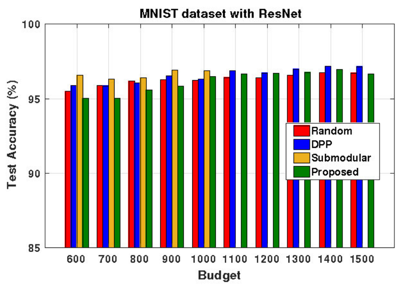
[1]https://www.mathworks.com/products/deep-learning.html



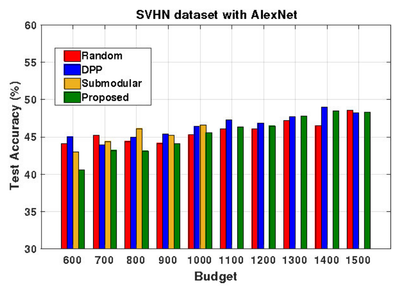
(a) MNIST with AlexNet



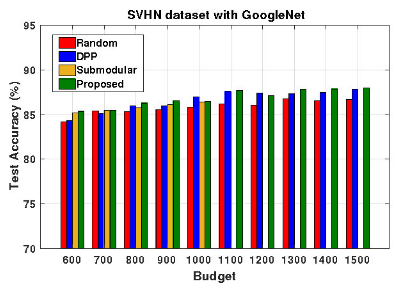
(b) MNIST with GoogleNet



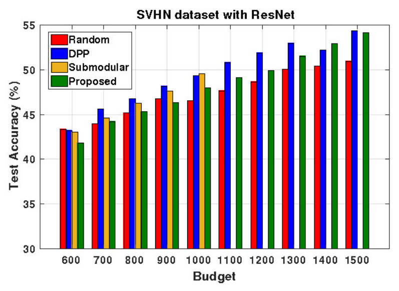
(c) MNIST with ResNet



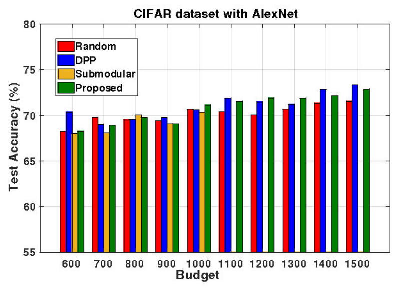
(d) SVHN with AlexNet



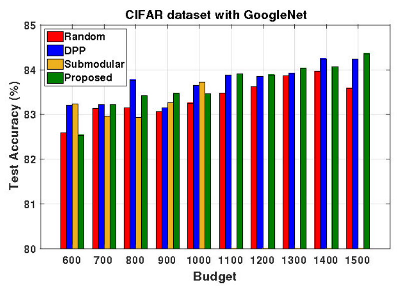
(e) SVHN with GoogleNet



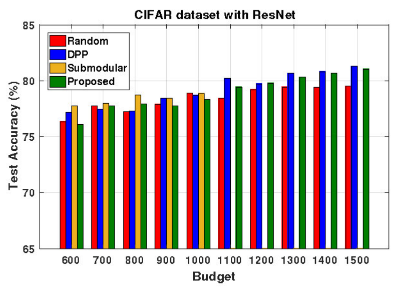
(f) SVHN with ResNet



(g) CIFAR with AlexNet



(h) CIFAR with GoogleNet



(i) CIFAR with ResNet

Fig. 1. Performance comparison on three datasets, using three pre-trained deep learning models. The x-axis denotes the budget for fine-tuning the models and the y-axis denotes the classification accuracy. For budgets greater than $1,000$, the submodular baseline was omitted as the algorithm did not terminate within 9 hours. Best viewed in color.

67 out of the 90 experiments. This shows the merit of our subset selection criteria and the efficiency of the LP relaxation proposed in Section III-B to derive a solution for the NP-hard IQP in Equation (6). *DPP* achieves the best results and obtains the highest accuracy in 43 of the 90 experiments. The *Submodular* sampling strategy obtains the highest accuracy in 16 experiments, while the proposed method in 25 experiments. The proposed method thus depicts much better performance than *Random Sampling* and its performance is comparable to *DPP* and *Submodular*.

Computation Time: An analysis of the computation time (time taken to select a subset of k samples from the ground set) for all the methods is reported in Table I (we exclude *Random Sampling* since there is no computation involved).

We note that the proposed method is computationally much more efficient than the *DPP* and *Submodular* methods. The *Submodular* method depicts the worst computation time, where it takes more than 8 hours to select a subset of size $1,000$. The *DPP* method is more efficient than *Submodular*, but has much higher computational overhead than our algorithm. For instance, on the MNIST dataset with AlexNet, *DPP* takes about 1 hour and 40 minutes to select a subset of size $1,500$, whereas our method takes only about 5 minutes. On the CIFAR dataset with ResNet, the corresponding values are 1 hour and 38 minutes and 7 minutes respectively. The maximum time incurred by our method to select a subset across all

	MNIST with AlexNet									
Subset Size	**600**	**700**	**800**	**900**	**1000**	**1100**	**1200**	**1300**	**1400**	**1500**
DPP	00:07:17	00:07:38	00:15:51	00:21:10	00:30:17	00:40:25	00:55:35	01:15:06	01:29:27	01:41:28
Submodular	05:10:13	06:01:54	06:53:34	07:45:15	08:36:55	-	-	-	-	-
Proposed	00:05:27	00:05:27	00:05:27	00:05:27	00:05:27	00:05:27	00:05:27	00:05:27	00:05:27	00:05:27
	MNIST with GoogleNet									
DPP	00:04:18	00:08:10	00:11:23	00:22:08	00:28:14	00:40:24	00:51:22	01:05:48	01:20:47	01:39:46
Submodular	05:10:13	06:01:54	06:53:34	07:45:15	08:36:55	-	-	-	-	-
Proposed	00:09:53	00:09:53	00:09:53	00:09:53	00:09:53	00:09:53	00:09:53	00:09:53	00:09:53	00:09:53
	MNIST with ResNet									
DPP	00:07:25	00:11:08	00:16:02	00:22:22	00:30:18	00:40:19	00:52:01	01:06:18	01:21:14	01:41:23
Submodular	05:10:13	06:01:54	06:53:34	07:45:15	08:36:55	-	-	-	-	-
Proposed	00:07:46	00:07:46	00:07:46	00:07:46	00:07:46	00:07:46	00:07:46	00:07:46	00:07:46	00:07:46
	SVHN with AlexNet									
DPP	00:04:01	00:06:27	00:09:25	00:12:46	00:18:01	00:36:34	00:30:39	00:38:24	00:48:32	01:00:22
Submodular	05:14:27	06:06:48	06:59:09	07:51:29	08:43:50	-	-	-	-	-
Proposed	00:05:28	00:05:28	00:05:28	00:05:28	00:05:28	00:05:28	00:05:28	00:05:28	00:05:28	00:05:28
	SVHN with GoogleNet									
DPP	00:04:07	00:06:22	00:09:13	00:13:05	00:18:09	00:23:42	00:30:43	00:38:24	00:48:01	01:01:59
Submodular	05:14:27	06:06:48	06:59:09	07:51:29	08:43:50	-	-	-	-	-
Proposed	00:08:51	00:08:51	00:08:51	00:08:51	00:08:51	00:08:51	00:08:51	00:08:51	00:08:51	00:08:51
	SVHN with ResNet									
DPP	00:04:00	00:06:22	00:09:46	00:13:24	00:17:48	00:23:06	00:33:18	00:40:49	00:47:57	00:58:58
Submodular	05:14:27	06:06:48	06:59:09	07:51:29	08:43:50	-	-	-	-	-
Proposed	00:07:19	00:07:19	00:07:19	00:07:19	00:07:19	00:07:19	00:07:19	00:07:19	00:07:19	00:07:19
	CIFAR with AlexNet									
DPP	00:05:01	00:06:13	00:11:10	00:15:36	00:20:18	00:23:44	00:50:47	01:04:09	01:20:43	01:38:01
Submodular	05:18:47	06:11:51	07:04:55	07:57:59	08:51:02	-	-	-	-	-
Proposed	00:09:33	00:09:33	00:09:33	00:09:33	00:09:33	00:09:33	00:09:33	00:09:33	00:09:33	00:09:33
	CIFAR with GoogleNet									
DPP	00:05:06	00:08:19	00:13:26	00:18:50	00:26:12	00:39:06	00:50:30	01:04:05	01:20:08	01:38:16
Submodular	05:18:47	06:11:51	07:04:55	07:57:59	08:51:02	-	-	-	-	-
Proposed	00:08:45	00:08:45	00:08:45	00:08:45	00:08:45	00:08:45	00:08:45	00:08:45	00:08:45	00:08:45
	CIFAR with ResNet									
DPP	00:04:07	00:06:26	00:09:06	00:13:05	00:17:55	00:39:02	00:52:34	01:03:52	01:19:47	01:38:06
Submodular	05:18:47	06:11:51	07:04:55	07:57:59	08:51:02	-	-	-	-	-
Proposed	00:07:32	00:07:32	00:07:32	00:07:32	00:07:32	00:07:32	00:07:32	00:07:32	00:07:32	00:07:32

TABLE I

COMPUTATION TIME ANALYSIS (TIME TAKEN TO SELECT A SUBSET OF k SAMPLES FROM THE GROUND SET). TIME FORMAT **HH:MM:SS**. WE EXCLUDE *Random Sampling* FROM THE ANALYSIS AS IT DOES NOT INVOLVE ANY COMPUTATION. FOR SUBSET SIZES GREATER THAN $1,000$ THE *Submodular* BASELINE DID NOT COMPLETE IN 9 HOURS AND SO, THE RESULTS WERE NOT INCLUDED IN THE TABLE.

experiments is less than 10 minutes. Our algorithm is based on solving an LP problem, followed by the truncated power method, both of which are computationally lightweight, which accounts for its efficient runtime. Further, for the proposed method, the computation time remains more or less constant with increasing subset sizes; this is immensely useful from a practical standpoint, since the subset size can vary depending on the available resources and maybe different for different applications. The *DPP* and *Submodular* methods, on the other hand, show an increasing trend in computation time with increasing subset sizes. In fact, for subset sizes greater than $1,000$, the *Submodular* baseline did not complete in 9 hours and so, the results were not included in the plots in Figure 1 and in Table I. The proposed technique thus outperforms *Random Sampling* comprehensively in terms of accuracy, depicts performance comparable to *DPP* and *Submodular*, but is computationally much more efficient than these two methods. This corroborates the usefulness of our method to select a subset of exemplar samples to fine-tune a deep learning architecture in resource-constrained applications, where computational complexity is an important consideration.

Experiment	**Full Training Set**		50% **Subset**		40% **Subset**		25% **Subset**	
	TT (hours)	**Acc** $(\%)$	**TT (hours)**	**Acc** $(\%)$	**TT (hours)**	**Acc** $(\%)$	**TT (hours)**	**Acc** $(\%)$
GoogleNet on MNIST	17.23	98.3	16.1	99.2	11.17	99.37	8.85	99.17
AlexNet on CIFAR	13.52	81.67	9.93	80.05	8.33	78.78	6.56	77.45

TABLE II

COMPARATIVE ANALYSIS OF APPROXIMATE TRAINING TIME (TIME TAKEN TO SELECT A SUBSET AND TRAIN THE MODEL) AND ACCURACY WHEN USING THE FULL TRAINING SET AND A SUBSETS OF SIZE $50\%, 40\%$ AND 25% OF THE FULL TRAINING SET USING OUR SUBSET SELECTION FRAMEWORK. **TT**: TRAINING TIME, **ACC**: ACCURACY

E. Analysis on the Full Target Dataset

In this experiment, we performed a comparative analysis of the accuracy and training time (time taken to select a subset and train the model) when fine-tuning a deep model using the full target dataset against that obtained when fine-tuning on a subset of the target dataset. Having established the usefulness of our sampling technique over the baselines in terms of accuracy and computation time, we focused only on our method in this experiment. Further, the selection times of *Submodular* and *DPP* will be prohibitively expensive for larger subsets (as evident from Table I) and so, were not included. We randomly selected two datasets (MNIST and CIFAR) and two models (GoogleNet and AlexNet) for this study. The results are shown in Table II. For the MNIST dataset using GoogleNet, the model was first fine-tuned on the full training set of $60,000$ samples and tested on the test set of $10,000$ samples; the accuracy obtained was 98.3% and the training time was approximately 17 hours. Our subset selection framework was then applied to select subsets of size 50%, 40% and 25% of the full training set. With a decrease in subset size, the training time reduced, but the test accuracy was better than that obtained when trained on the full dataset. With 25% training data, the training time was 8.85 hours and the accuracy was 99.17%. We thus achieved a 1.94x speed-up on the training time. Hence, our framework can also be extremely useful in time-critical applications, where training a deep model with large amounts of data over extended periods of time is not feasible. More importantly, with only 25% labeled training data, our subset selection framework outperformed the accuracy obtained using the full training set; our framework thus obtained a higher accuracy with much reduced training time, which shows its efficacy. For the CIFAR dataset on AlexNet, the full training set contained $50,000$ samples and the test set contained $10,000$ samples. Our framework achieved a 1.36x speed-up on the training time with a marginal loss in accuracy of about 1.6% with 50% training data, as compared to the values obtained when trained on the full training set; with 25% training data, the corresponding figures are 2.06x and 4%. These results show the usefulness of our sampling framework to identify the salient and exemplar instances from large amounts of data, discard redundant or duplicate data and obtain a reliable deep model with limited computational and memory resources. Depending on the nature of the dataset, it may often produce better accuracies (or a marginal degradation in accuracy) at a much reduced training time, compared to that obtained when trained on the full training set. This further validates the efficacy of our technique for real-world applications, where the allowable training set size is dictated by the available computational resources and varies from one application to another. Our analysis revealed that computation of the $N \times N$ diversity matrix (Equation (4)) for a ground set of size N was the main bottleneck of our method. As part of future research, we plan to study the performance of random projection algorithms (as mentioned in Section III) to further reduce the training time in our framework. (Note that in all these experiments, we used pre-trained models where some of the layers were frozen to accelerate the training process; it may thus not be fair to compare the accuracy values obtained using the full training set to the best reported accuracies in the literature on these datasets).

F. Performance on a Regression Application

To further study the performance of our framework, we conducted experiments on a regression application. We used the IMDB dataset in this experiment, which contains face images with age labels of celebrities from the IMDB database [50]. We updated the network architecture to include a regression layer at the end of the network, instead of a classification layer. As before, the initial training set L contained $7,000$ samples, the ground set Z had $3,000$ samples and $3,000$ samples constituted the test set, all randomly selected from the dataset. Since entropy does not have an exact analog in the regression setting, the utility vector u in our method (Equation (3)) was computed using a variance-based strategy: each sample in the ground set was passed as an input to the three pre-trained deep models (AlexNet, GoogleNet and ResNet), and the variance of the outputs produced was selected as the utility score of that sample. This is intuitive as a high variance denotes a high degree of disagreement among the models and thus, a sample of high usefulness. The diversity term was computed using Equation (4), as before. We present the results with the ResNet model with a subset size of $2,000$; other models depicted similar performance. Table III depicts the mean squared error and the computation time (time taken to select a subset of size $2,000$ from the ground set). We note that all the methods depict very similar RMSE values for this dataset. The proposed method is much more efficient in terms of computation as compared to *DPP* and *Submodular*. This corroborates the generalizibility of our framework across different applications. This also shows that *Random Sampling* can sometimes be an efficient solution, both in terms of computation and error rate.

Method	Random	DPP	Submodular	Proposed
RMSE	12.76	12.74	12.73	12.74
Computation Time	-	04:07:28	143:49:50	01:46:59

TABLE III
RMSE AND COMPUTATION TIME ON THE IMDB DATASET WITH RESNET FOR SUBSET SIZE $2,000$. TIME FORMAT **HH:MM:SS**.

V. Conclusion and Future Work

In this paper, we addressed the challenging problem of selecting a subset of informative samples to fine-tune a pre-trained deep learning model (trained on a particular dataset) to a different task, under significant resource constraints. This problem is of immense practical importance, as several real-world applications (such as those running on mobile platforms) are severely limited in terms of memory and computational resources. We posed the subset selection as a constrained NP-hard integer quadratic programming problem and derived an efficient linear relaxation to select a subset of informative samples. Our extensive empirical studies on three challenging computer vision datasets using three commonly used pre-trained deep learning models (AlexNet, GoogleNet and ResNet) demonstrated the merit of our method over competing baselines (in terms of accuracy as well as computational overhead). We also studied the performance of our framework on a regression application.

As part of future work, we plan to study the performance of our framework on other popular architectures, such as the ResNeXt [29] and the VGGNet [5]. We also plan to explore other vision applications, such as image segmentation using the SegNet model [3] and object detection using R-CNN [2]. We will study other strategies to solve the optimization problem in Equation (7). For instance, techniques based on semi-definite programming (SDP) relaxation have been explored to solve quadratic optimization problems; while SDP is computationally intensive, efficient and scalable algorithms have been proposed to overcome this challenge [51], [52]. We will also attempt to prove theoretical guarantees on the quality of the solutions.

Further, even though validated in the context of fine-tuning deep architectures, the subset selection algorithm proposed in this work is generic, and can be adapted to other applications with an appropriate modification of the terms in the objective function in Equation (7). For instance, video summarization is essentially a subset selection problem under a given budget (the summary length). Feature selection is another example where the goal is to extract a subset of informative features. As part of future research, we plan to study the performance of our framework on these problems and explore its generalizibility across different application domains.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[2] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 39, no. 6, 2017.

[3] V. Badrinarayanan, A. Kendall, and R. Cipolla, “SegNet: A deep convolutional encoder-decoder architecture for image segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[4] A. Krizhevsky, I. Sutskever, and G. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Neural Information Processing Systems (NIPS)*, 2012.

[5] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *https://arxiv.org/abs/1409.1556*, 2014.

[6] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[7] J. Dean, G. Corrado, and R. M. et al., “Large scale distributed deep networks,” in *Advances of Neural Information Processing Systems (NIPS)*, 2012.

[8] G. Alain, A. Lamb, C. Sankar, A. Courville, and Y. Bengio, “Variance reduction in SGD by distributed importance sampling,” in *International Conference on Learning Representations (ICLR) Workshops*, 2015.

[9] I. Loshchilov and F. Hutter, “Online batch selection for faster training of neural networks,” in *International Conference on Learning Representations (ICLR) Workshops*, 2016.

[10] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, “Prioritized experience replay,” in *International Conference on Learning Representations (ICLR)*, 2016.

[11] A. Katharopoulos and F. Fleuret, “Not all samples are created equal: Deep learning with importance sampling,” in *International Conference on Machine Learning (ICML)*, 2018.

[12] T. Johnson and C. Guestrin, “Training deep models faster with robust, approximate importance sampling,” in *Neural Information Processing Systems (NIPS)*, 2018.

[13] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” in *Deep Learning Workshop at Neural Information Processing Systems (NIPS-W)*, 2014.

[14] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, “Quantized neural networks: Training neural networks with low precision weights and activations,” *Journal of Machine Learning Research*, vol. 18, no. 1, 2017.

[15] M. McDonnell, “Training wide residual networks for deployment using a single bit for each weight,” in *International Conference on Learning Representations (ICLR)*, 2018.

[16] M. Jaderberg, A. Vedaldi, and A. Zisserman, “Speeding up convolutional neural networks with low rank expansions,” in *British Machine Vision Conference (BMVC)*, 2014.

[17] S. Han, H. Mao, and W. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding,” in *International Conference on Learning Representations (ICLR)*, 2016.

[18] T. Zhang, S. Ye, K. Zhang, J. Tang, W. Wen, M. Fardad, and Y. Wang, “A systematic DNN weight pruning framework using alternating direction method of multipliers,” in *European Conference on Computer Vision (ECCV)*, 2018.

[19] M. Lin, Q. Chen, and S. Yan, “Network in network,” in *https://arxiv.org/abs/1312.4400*, 2013.

[20] A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” in *https://arxiv.org/abs/1704.04861*, 2017.

[21] Y. Cheng, F. Yu, R. Feris, S. Kumar, A. Choudhary, and S. Chang, “An exploration of parameter redundancy in deep networks with circulant projections,” in *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[22] M. Mathieu, M. Henaff, and Y. LeCun, “Fast training of convolutional networks through ffts,” in *arXiv:1312.5851*, 2013.

[23] Z. Zhang, Y. Chen, and V. Saligrama, “Efficient training of very deep neural networks for supervised hashing,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[24] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *arXiv:1502.03167*, 2015.

[25] B. Ko, H. Kim, and H. Choi, "Controlled dropout: A different dropout for improving training speed on deep neural network," in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2017.

[26] A. Sergeev and M. Balso, "Horovod: Fast and easy distributed deep learning in TensorFlow," in *arXiv:1802.05799*, 2018.

[27] D. Amodei and D. Hernandez, "Ai and compute," in *https://openai.com/blog/ai-and-compute/*, 2018.

[28] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[29] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[30] P. Goyal, P. Dollar, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: Training imagenet in 1 hour," in *arXiv:1706.02677*, 2017.

[31] S. Banerjee and S. Chakraborty, "Deepsub: A novel subset selection framework for training deep learning architectures," in *IEEE International Conference on Image Processing (ICIP)*, 2019.

[32] B. Settles, *Active Learning*. Morgan and Claypool Publishers, 2012.

[33] S. Chakraborty, V. Balasubramanian, Q. Sun, S. Panchanathan, and J. Ye, "Active batch selection via convex relaxations with guaranteed solution bounds," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 37, no. 10, 2015.

[34] H. Venkateswara, P. Lade, B. Lin, J. Ye, and S. Panchanathan, "Efficient approximate solutions to mutual information based global feature selection," in *IEEE International Conference on Data Mining (ICDM)*, 2015.

[35] S. Vempala, *The Random Projection Method*. American Mathematical Society, 2004.

[36] X. Yuan and T. Zhang, "Truncated power method for sparse eigenvalue problems," *Journal of Machine Learning Research (JMLR)*, vol. 14, no. 1, 2013.

[37] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, 1998.

[38] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng, "Reading digits in natural images with unsupervised feature learning," in *Workshop on Deep Learning and Unsupervised Feature Learning at NIPS*, 2011.

[39] A. Krizhevsky, "Learning multiple layers of features from tiny images," in *Technical Report*, 2009.

[40] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[41] A. Brock, T. Lim, J. Ritchie, and N. Weston, "Freezeout: Accelerate training by progressively freezing layers," in *https://arxiv.org/abs/1706.04983*, 2017.

[42] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge University Press, 1995.

[43] A. Civril and M. Ismail, "On selecting a maximum volume sub-matrix of a matrix and related problems," *Theoretical Computer Science*, vol. 410, no. 47-49, 2009.

[44] A. Kulesza and B. Taskar, "Determinantal point processes for machine learning," *Foundations and Trends in Machine Learning*, vol. 5, no. 2-3, 2013.

[45] B. Gong, W. Chao, K. Grauman, and F. Sha, "Diverse sequential subset selection for supervised video summarization," in *Neural Information Processing Systems (NIPS)*, 2014.

[46] G. Nemhauser, L. Wolsey, and M. Fisher, "An analysis of approximations for maximizing submodular set functions," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[47] K. Wei, R. Iyer, and J. Bilmes, "Submodularity in data subset selection and active learning," in *International Conference on Machine Learning (ICML)*, 2015.

[48] M. Gygli, H. Grabner, and L. V. Gool, "Video summarization by learning submodular mixtures of objectives," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[49] G. Roffo, S. Melzi, U. Castellani, and A. Vinciarelli, "Infinite latent feature selection: A probabilistic latent graph-based ranking approach," in *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[50] A. Rozantsev, M. Salzmann, and P. Fua, "Residual parameter transfer for deep domain adaptation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[51] Z. Luo, W. Ma, A. So, Y. Ye, and S. Zhang, "Semidefinite relaxation of quadratic optimization problems," *IEEE Signal Processing Magazine*, vol. 27, no. 3, 2010.

[52] P. Wang, C. Shen, A. Hengel, and P. Torr, "Large-scale binary quadratic optimization using semidefinite relaxation and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 39, no. 3, 2017.