Deterministic Mini-batch Sequencing for Training Deep Neural Networks

Subhankar Banerjee, Shayok Chakraborty
Department of Computer Science, Florida State University

Abstract

Recent advancements in the field of deep learning have dramatically improved the performance of machine learning models in a variety of applications, including computer vision, text mining, speech processing and fraud detection among others. Mini-batch gradient descent is the standard algorithm to train deep models, where mini-batches of a fixed size are sampled randomly from the training data and passed through the network sequentially. In this paper, we present a novel algorithm to generate a deterministic sequence of mini-batches to train a deep neural network (rather than a random sequence). Our rationale is to select a mini-batch by minimizing the Maximum Mean Discrepancy (MMD) between the already selected mini-batches and the unselected training samples. We pose the mini-batch selection as a constrained optimization problem and derive a linear programming relaxation to determine the sequence of mini-batches. To the best of our knowledge, this is the first research effort that uses the MMD criterion to determine a sequence of mini-batches to train a deep neural network. The proposed mini-batch sequencing strategy is deterministic and independent of the underlying network architecture and prediction task. Our extensive empirical analyses on three challenging datasets corroborate the merit of our framework over competing baselines. We further study the performance of our framework on two other applications besides classification (regression and semantic segmentation) to validate its generalizability.

Introduction

Deep learning algorithms automatically learn a discriminating set of features and have depicted impressive performance in a variety of applications. Architectures such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Generative Adversarial Networks (GANs) etc., have revolutionized multimedia computing and have achieved state-of-the-art performance across a wide range of applications, including image recognition (He et al. 2016a), object detection (Ren et al. 2017), image segmentation (Badrinarayanan, Kendall, and Cipolla 2015) and speech recognition (Deng and Platt 2014) among others. Conventionally, deep neural networks (DNNs) are trained using stochastic gradient descent (SGD), where the back-propagation algorithm is used to compute the gradients. A given training set D is split into a sequence of mini-batches $\{b_1, b_2, \dots b_n\}$ each of a pre-determined size k, where b_t is sampled at random from D. A loss function $\mathcal{L}(w_t)$ (such as the cross-entropy loss) is defined with respect to the current model parameters w_t (at time instance t) and is designed to operate on each mini-batch. The updated network weights, w_{t+1} at time $t+1$, are obtained by minimizing the loss according to SGD:

$$w_{t+1} = w_t - \lambda_t \frac{\partial \mathcal{L}(w_t)}{\partial w_t} \tag{1}$$

where λ_t is the learning rate at time t. Earlier research efforts have established that the selection of the mini-batch b_t that is used to compute the gradient $\frac{\partial \mathcal{L}(w_t)}{\partial w_t}$ at time step t is crucial to improving the generalization performance of the model, motivating the development of intelligent sampling techniques to progressively select the training mini-batches (Joseph et al. 2019; Loshchilov and Hutter 2016). In this paper, we propose an algorithm to generate a deterministic sequence of mini-batches to train a deep neural network. Our rationale is to select the mini-batch b_t such that the data distribution represented by b_t and the already selected mini-batches, is closest to the distribution represented by the unselected training samples. In other words, the exemplar samples from the training data are selected progressively to form the training mini-batches, which can potentially improve the generalization performance of the deep model.

We use the Maximum Mean Discrepancy (MMD) metric (Gretton et al. 2007; Borgwardt et al. 2006; Sriperumbudur et al. 2010) to compute the probability distribution difference between two sets of training samples. MMD is defined as the difference between the means of two distributions after mapping them onto a Reproducing Kernel Hilbert Space (RKHS). It has been successfully used in a variety of applications to compute the distribution difference between training and test data, particularly in the development of transfer learning and active learning algorithms (Chattopadhyay et al. 2013a; Wang and Ye 2013; Chattopadhyay et al. 2012, 2013b; Venkateswara et al. 2017; Tang and Huang 2019). We use this metric to formulate an optimization problem to select a mini-batch B containing k samples from the set of unselected training samples Q, such that the probability distribution difference between $Q \backslash B$ and $P \cup B$ (where P is the set of already selected mini-batches) is minimized. Our

contributions in this paper are summarized below:

- we leverage the MMD metric to derive a sequence of mini-batches to train a deep learning model; to the best of our knowledge, this is the first research effort that uses MMD to select mini-batches for training a DNN
- our mini-batch sequencing strategy is deterministic and independent of the underlying network and prediction task (classification / regression / semantic segmentation etc.). This property is particularly useful as it enables us to determine the mini-batch sequence for a given training data without the need to be meticulously knowledgeable about the task or the optimal deep model for the task
- we conduct an extensive set of experiments to study the performance of our framework against competing baselines on three challenging datasets; we also study the effect of batch size, kernel functions and the performance of our framework on regression and semantic segmentation, to validate its generalizability.

Related Work

In this section, we present a survey of mini-batch sampling algorithms for training deep neural networks. Existing research in this area can be broadly categorized into three groups:

Importance Sampling: Importance sampling aims at increasing the convergence speed of SGD by reducing the variance of the gradient estimates. Zhao and Zhang (Zhao and Zhang 2015) established that the optimal sampling distribution is directly related to the absolute values of the gradient of the samples, for convex objective functions. The simplicity of the optimization problems permitted the authors to sample proportionally to the norm of the inputs. Unfortunately, such simple importance measures do not exist for deep learning and requires clusters of GPU workers just to compute the sampling distribution (Alain et al. 2015). Researchers have therefore resorted to manually tuned sampling schemes for training DNNs. Loss-based sampling techniques have also been explored (Loshchilov and Hutter 2016; Schaul et al. 2016) which maintain a history of loss values of training points and sample either proportionally to the loss or based on the loss ranking. One of the major drawbacks of these techniques is the need to tune a large number of hyper-parameters. Katharopoulos and Fleuret (Katharopoulos and Fleuret 2018) proposed an importance sampling scheme based on the upper bound of the gradient norm. The same authors (Katharopoulos and Fleuret 2017) also proposed to use the loss itself as the importance metric rather than the gradient norm and proved that the loss can be used to create a tighter upper bound to the gradient norm than uniform sampling. However, as evident from the algorithms, both these sampling techniques are dependent on the architecture of the network.

Submodular Optimization and DPP: Recently, submodular optimization and determinantal point process (DPP) based techniques have been explored to generate mini-batch sequences for training a DNN. These methods are based on the intuition that mini-batches with high representativeness / diversity can contribute to better training of the deep neural network; they therefore attempt to select the informative and exemplar training samples progressively in the mini-batches. Joseph *et al.* (Joseph et al. 2019) proposed a submodular optimization framework for selecting mini-batches, which was based on maximizing the uncertainty, diversity, mean closeness and feature match scores of the samples in a mini-batch. Wang *et al.* (Wang et al. 2019) also leveraged submodular optimization strategies and proposed a hierarchical robust partitioning framework to generate a sequence of training mini-batches. A mini-batch selection strategy based on DPPs was proposed by Zhang *et al.* (Zhang, Kjellstrom, and Mandt 2017), which encourages selection of mini-batches with diverse data samples. Along similar lines, the authors also proposed a sampling scheme based on Repulsive Point Processes (RPPs) to reduce the variance of the gradient estimator (Zhang et al. 2018). However, both these techniques are computationally inefficient, as noted in (Joseph et al. 2019; Wang et al. 2019).

Other Sampling Techniques: A few other relevant techniques are summarized in this section. Wu *et al*. proposed a simple margin-based loss, which encourages all positive samples in a batch to be within a specific distance of each other, rather than being as close as possible (Wu et al. 2017). Fan *et al.* used reinforcement learning to train a neural network to select training samples for a target network to optimize the convergence speed (Fan et al. 2017). While the empirical results were promising, the computational overhead of training two deep networks was a major drawback of this method. There have been a few research efforts to accelerate the convergence of SGD through variance reduction (Allen-Zhu 2018; Defazio, Bach, and Lacoste-Julien 2014; Lei et al. 2017). However, these algorithms do not focus on mini-batch selection and demonstrate performance worse than simple SGD in general (Katharopoulos and Fleuret 2018). Curriculum learning (Bengio et al. 2009) and its evolution self-paced learning (Kumar, Packer, and Koller 2010) is another class of methods related to importance sampling which present the classifier with easy samples first (that are likely to have a small loss) and gradually introduce harder samples.

Contrary to most of these methods, our batch selection method is independent of the learning objective and the network architecture; the mini-batch sequence can be precomputed independently for a given learning task. It also does not require extensive hyper-parameter tuning and is based on solving a linear programming (LP) problem. We now describe our framework.

Proposed Framework

Maximum Mean Discrepancy

Our framework is based on the intuition that mini-batches which progressively include the exemplar and informative training samples contribute to better training of the deep neural network (Joseph et al. 2019; Loshchilov and Hutter 2016). Let P denote the set of training samples that have already been selected to form the mini-batches and Q denote the set of unselected training samples from the training dataset D. Let n_P and n_Q be the number of samples in the sets P and Q respectively; then we have, $n_P + n_Q = N$,

the total number of samples in D. Our objective is to select the next mini-batch B containing k samples from Q. The rationale of our framework is to select B in such a way that the joint probability distribution $Prob(X, Y)$ represented by $P \cup B$ is maximally similar to that represented by $Q \backslash B$. This would imply that the selected training samples is a good representation of the unselected training samples, or in other words, the exemplar samples from Q have been selected into the batch B. Since all the training samples are assumed to be derived from the same probability distribution, the conditional probability distribution $Prob(Y|X)$ remains the same for $P \cup B$ and $Q \backslash B$. Thus, ensuring similar joint distribution reduces to ensuring similar marginal distribution $Prob(X)$ between the two sets $P \cup B$ and $Q \backslash B$.

The Maximum Mean Discrepancy (MMD) is a statistical metric to compute the difference in marginal probability between two distributions (Gretton et al. 2007; Borgwardt et al. 2006; Sriperumbudur et al. 2010). It is computed as the difference between the empirical means of the two distributions after mapping onto a Reproducing Kernel Hilbert Space (RKHS). Let $A = \{a_1, a_2, \ldots, a_{n_1}\}$ and $B = \{b_1, b_2, \ldots, b_{n_2}\}$ be two sets of samples drawn randomly from a target population. An empirical estimate of MMD between A and B is obtained as (Borgwardt et al. 2006):

$$\left\| \frac{1}{n_1} \sum_{i=1}^{n_1} \phi(a_i) - \frac{1}{n_2} \sum_{i=1}^{n_2} \phi(b_i) \right\|_{\mathcal{H}}^2 \tag{2}$$

where $\mathcal{H}$ is a universal RKHS (Steinwart 2001) and $\phi : \mathcal{X} \rightarrow \mathcal{H}$ is a mapping from the input space $\mathcal{X}$ to the RKHS $\mathcal{H}$. Our goal is therefore to select a mini-batch B such that the MMD between the sets $P \cup B$ and $Q \backslash B$ is minimized.

Mini-batch Selection Framework

We define a loss function $f(.)$ to compute the utility of a batch of samples B, which captures the MMD between the two sets $P \cup B$ and $Q \backslash B$:

$$f(B) = \left\| \frac{1}{n_P + k} \sum_{i \in P \cup B} \phi(x_i) - \frac{1}{n_Q - k} \sum_{i \in Q \backslash B} \phi(x_i) \right\|_{\mathcal{H}}^2 \tag{3}$$

We define a binary vector $m \in \{0, 1\}^{n_Q \times 1}$ of size n_Q, which denotes which samples from Q should be selected in the mini-batch B. If $m_i = 1$, it implies that sample x_i in Q should be selected in the batch; if $m_i = 0$, then x_i should not be selected. With this definition, minimizing the loss function $f(B)$ can be posed as the following optimization problem:

$$\min_m \left\| \frac{1}{n_P + k} \left\{ \sum_{i \in P} \phi(x_i) + \sum_{i \in Q} m_i \phi(x_i) \right\} - \frac{1}{n_Q - k} \sum_{i \in Q} (\mathbf{1} - m_i) \phi(x_i) \right\|_{\mathcal{H}}^2$$
$$s.t. \quad m_i \in \{0, 1\}, \forall i \text{ and } \sum_{i=1}^{n_Q} m_i = k \tag{4}$$

where $\mathbf{1}$ is a vector of size n_Q with all entries 1 and k is the pre-specified mini-batch size. The first summation in Equation (3) can be split into two terms (summing over the sets P and Q) using the binary variable m, as only the samples in Q that are selected in B by m will contribute to the first summation. Similarly, the second summation can be run entirely over Q as the coefficient $(1 - m_i)$ will ensure that only the samples that are not selected in the batch B contribute to the second summation. Thus, the objectives in Equations (3) and (4) are equivalent. The constraints respectively denote that m is a binary vector and the sum of all elements on m is k, as we are allowed to select only k samples in the mini-batch B. Using the properties of RKHS (Wang and Ye 2013), the objective can be simplified as follows:

$$\min_m m^T \Phi_1 m - \phi_2^T m + \phi_3^T m$$
$$s.t. \quad m_i \in \{0, 1\}, \forall i \text{ and } \sum_{i=1}^{n_Q} m_i = k \tag{5}$$

where the matrix Φ_1 and the vectors ϕ_2 and ϕ_2 are defined as follows (T denotes the vector transpose operator):

$$\Phi_1(i, j) = \phi(x_i, x_j), \text{ where } (x_i, x_j) \in Q, \forall i, j$$
$$\phi_2(i) = \frac{n_P + k}{N} \sum_{j=1}^{n_Q} \phi(x_i, x_j), \text{ where } (x_i, x_j) \in Q, \forall i, j$$
$$\phi_3(i) = \frac{n_Q - k}{N} \sum_{j=1}^{n_P} \phi(x_i, x_j), \text{ where } x_i \in Q, x_j \in P, \forall i, j \tag{6}$$

We note here that the kernel evaluations are all performed in order, that is, the first sample in Q produces the first row of Φ_1 and so on. This is because each entry in the vector m corresponds to a particular sample in Q, which necessitates a correspondence between the samples in Q and the rows of the kernel matrix and vectors.

Due to the binary integer constraints on m, we can combine Φ_1, ϕ_2 and ϕ_3 into a single matrix $Z \in \Re^{n_Q \times n_Q}$ and rewrite the optimization problem in Equation (5) as follows:

$$\min_m m^T Z m$$
$$s.t. \quad m_i \in \{0, 1\}, \forall i \text{ and } \sum_{i=1}^{n_Q} m_i = k \tag{7}$$

where the matrix Z is constructed as follows:

$$Z(i, j) = \begin{cases} \Phi_1(i, j), & \text{if } i \neq j \\ \Phi_1(i, j) - \phi_2(i) + \phi_3(i), & \text{if } i = j \end{cases} \tag{8}$$

This is an integer quadratic programming problem (IQP); the binary integer constraints on m make this IQP NP-hard.

An Efficient LP Relaxation

We define a binary matrix $W \in \Re^{n_Q \times n_Q}$ as $w_{ij} = m_i.m_j$. We thus rewrite the optimization problem in Equation (7) as:

$$\begin{aligned} \min_{m,W} & \sum_{i,j} z_{ij} w_{ij} \\ s.t. \quad & w_{ij} = m_i.m_j \\ & m_i, w_{ij} \in \{0,1\}, \forall i,j \\ & \sum_{i=1}^{n_Q} m_i = k \end{aligned} \tag{9}$$

We now attempt to write the quadratic equality $w_{ij} = m_i.m_j$ as a linear term. From the definition, w_{ij} will be equal to 1 when both m_i and m_j are 1 and will be 0 otherwise. Note that we are solving a minimization problem, where the coefficient matrix Z can have both positive and negative entries (as evident from Equation (8)). We make the following observations:

- When $z_{ij} < 0$, the quadratic equality $w_{ij} = m_i.m_j$ is equivalent to the linear inequality $-m_i - m_j + 2w_{ij} \leq 0$. A simple analysis of the inequality reveals that when m_i and m_j are both 0 or, one of them is 0 and the other one is 1, w_{ij} is forced to be 0. When both m_i and m_j are 1, w_{ij} is free to be 0 or 1. However, we are solving a minimization problem, where the objective is $\sum_{i,j} z_{ij} w_{ij}$ and $z_{ij} < 0$; thus the nature of the problem will force w_{ij} to be 1, as that will produce a better (lower) value of the objective.
- When $z_{ij} \geq 0$, the quadratic equality $w_{ij} = m_i.m_j$ is equivalent to the linear inequality $m_i + m_j \leq 1 + 2w_{ij}$. When both m_i and m_j are 1, w_{ij} is forced to be 1. When both m_i and m_j are 0 or, one of them is 0 and the other one is 1, w_{ij} is free to be 0 or 1. However, we are solving a minimization problem, where the objective is $\sum_{i,j} z_{ij} w_{ij}$ and $z_{ij} \geq 0$; thus the nature of the problem will force w_{ij} to be 0, as that will produce a better (lower) value of the objective. Thus, the quadratic equality and the linear inequalities produce the exact same values of w_{ij} for all possible values of m_i and m_j for both these cases, showing their equivalence.

We can therefore simplify the problem in Equation (9) as:

$$\begin{aligned} \min_{m,W} & \sum_{i,j} z_{ij} w_{ij} \\ s.t. \quad & -m_i - m_j + 2w_{ij} \leq 0 \quad \text{for} \quad z_{ij} < 0 \\ & m_i + m_j \leq 1 + 2w_{ij} \quad \text{for} \quad z_{ij} \geq 0 \\ & m_i, w_{ij} \in \{0,1\}, \forall i,j \\ & \sum_{i=1}^{n_Q} m_i = k \end{aligned} \tag{10}$$

This is an integer linear programming (ILP) problem (as both the objective function and the constraints are linear in the variables), equivalent to the IQP in Equation (7). We relax the integer constraints into continuous constraints and use an off-the-shelf LP solver to solve this problem. After obtaining the continuous solution, we recover the integer solution of our variable of interest m, using a greedy approach where the k highest entries in m are reconstructed as 1 and the other entries as 0. This vector m dictates which samples in Q should be selected in the next mini-batch. The pseudocode is summarized in Algorithm 1 (for one mini-batch selection iteration).

Algorithm 1 The Proposed MMD-based Mini-batch Selection Algorithm

Require: Training data D with N samples, already selected set of training samples P, set of unselected training samples Q, mini-batch size k, kernel function $\phi(.,.)$

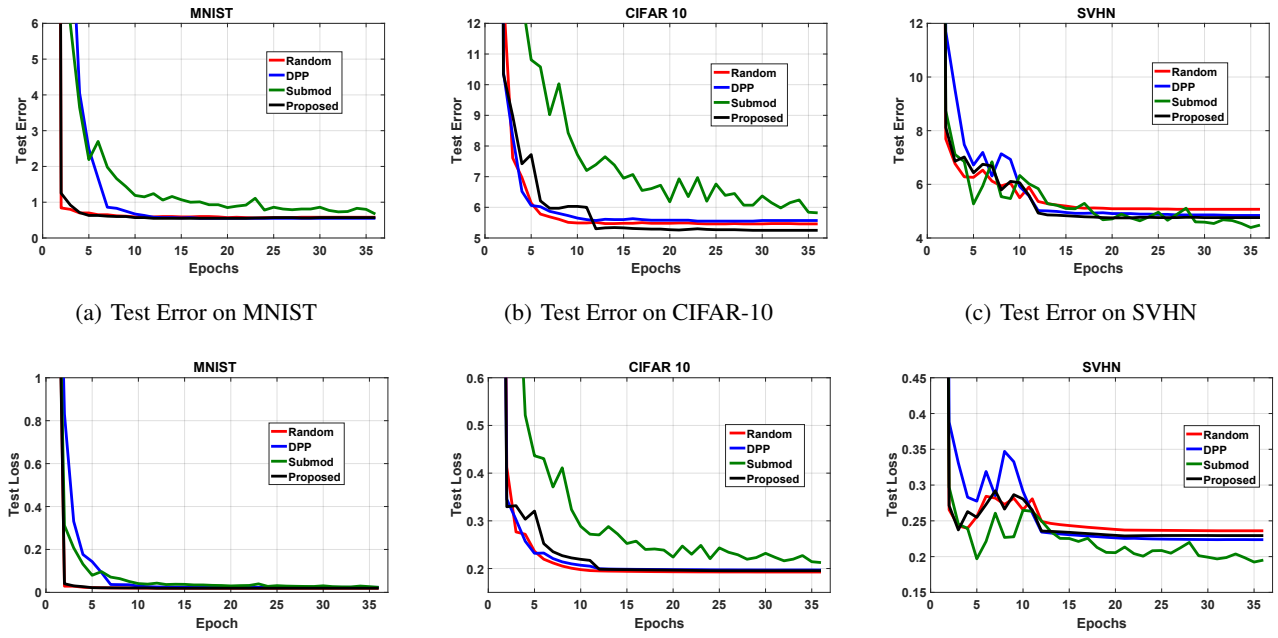
1: Compute the matrix Φ_1 and the vectors ϕ_2 and ϕ_3 as shown in Equation (6)
2: Compute the matrix Z as shown in Equation (8)
3: Solve the LP problem in Equation (10) after relaxing the integer constraints
4: Round the solution to derive the vector m
5: Select the next mini-batch B from Q based on the entries in m

As evident from the algorithm, our framework does not require any knowledge of the network architecture or the prediction task; the mini-batch sequence can be precomputed independently for any given learning problem. Computation of the kernel matrix Φ_1 (Equation (6)) has quadratic complexity and can be challenging for large datasets. We note that Φ_1 needs to be computed just once for the entire training set D. When training samples are selected from D into P, the corresponding rows and columns can be deleted from Φ_1 to get an updated kernel matrix for the unselected training samples. Moreover, the theory of random projections (Vempala 2004) and advanced data structures such as KD-trees can be used to efficiently compute this matrix (this will be investigated as part of future research). Our framework is easy to implement, does not need extensive hyper-parameter tuning, and is thus a promising candidate for practical applications.

Experiments and Results

Datasets and Experimental Setup

We conducted extensive experimental evaluations to study the performance of the proposed MMD formulation for mini-batch selection in training deep neural networks, the effects of mini-batch size, kernel functions and its generalizability across deep learning architectures and learning tasks. We compared the performance of our framework against three baseline methods: (i) *Random Sampling*, where mini-batches are sampled randomly from the training data (as done conventionally); (ii) the *submodular* mini-batch selection framework (Joseph et al. 2019); and (iii) the *DPP* based sampling strategy (Zhang, Kjellstrom, and Mandt 2017). These methods were selected as comparison baselines, as they share the same intuition as the proposed algorithm: mini-batches which progressively include the exemplar training samples contribute to better training of the



(a) Test Error on MNIST (b) Test Error on CIFAR-10 (c) Test Error on SVHN

(d) Test Loss on MNIST (e) Test Loss on CIFAR-10 (f) Test Loss on SVHN

Figure 1: Performance comparison of the *Proposed, Submodular, DPP* and *Random Sampling* techniques on MNIST, CIFAR-10 and SVHN datasets. The x-axis denotes the epoch number and the y-axis denotes test error (in percentage) / test loss. Best viewed in color.

Method	Random	DPP	Submod	Proposed
	MNIST			
Mean Error	**0.60**	6.68	1.53	**0.60**
Final Error	0.58	**0.55**	0.67	0.57
Mean Loss	**0.02**	0.41	0.05	**0.02**
Final Loss	**0.01**	0.02	0.02	**0.01**
	CIFAR-10			
Mean Error	5.82	**5.52**	8.13	5.78
Final Error	5.46	5.57	5.82	**5.25**
Mean Loss	**0.21**	**0.21**	0.30	**0.21**
Final Loss	**0.19**	**0.19**	0.21	**0.19**
	SVHN			
Mean Error	5.45	5.62	5.29	**5.28**
Final Error	5.07	4.84	**4.48**	4.76
Mean Loss	0.24	0.25	**0.22**	0.24
Final Loss	0.23	0.22	**0.19**	0.22

Table 1: Mean and final test error (in percentage) and test loss values over 35 epochs for all the methods and all the datasets. The best values are shown in bold.

deep neural network. The results of *Random Sampling* were averaged over 3 runs.

We studied the performance of our framework on three benchmark datasets: MNIST (LeCun et al. 1998), CIFAR-10 (Krizhevsky 2009) and SVHN (Netzer et al. 2011). The train / test splits given in each of these datasets were used in our experiments. ResNet-18 (He et al. 2016b) was used as the backbone network architecture. The implementations were all performed in Matlab R2019b running on a workstation equipped with an NVIDIA Quadro RTX5000 GPU with 16GB memory. The pre-trained models were obtained from the Matlab Deep Learning Toolbox [1]. The L_2 regularizer was used with regularization parameter 0.0005. We used 0.001 as the initial learning rate and reduced it by a factor of 0.1 every 10 epochs. The stochastic gradient descent with momentum (SGDM) was used as the optimizer, where the momentum parameter was set at 0.9. All the experiments were run for 35 epochs with a mini-batch size of 50. A Gaussian kernel with parameter 1 was used in the MMD computations.

Results

The results are presented in Figure 1 and Table 1. The generalization performance of the network, computed by its test error and test loss, was used as the evaluation metric. As evident from the mean and final error and loss values in the table, the proposed algorithm *always* depicts equal or better performance than *Random Sampling*. Thus, selecting mini-batches sequentially by minimizing the MMD between the already selected mini-batches and the unselected training samples, either outperforms or depicts similar performance as the conventional method of selecting the mini-batches at random, both in terms of generalization error and loss. The *Submodular* selection method sometimes depicts very good

[1] https://www.mathworks.com/help/deeplearning/index.html?s_tid=CRUX_gn_documentation

performance (such as the SVHN dataset, where it achieves the lowest error and loss); however, it is not consistent across the datasets in its performance. The *DPP* technique demonstrates good performance on the CIFAR-10 dataset, where it achieves the lowest error and loss; but, for MNIST and SVHN, it depicts a much slower decrease in generalization error and loss over training epochs, as compared to the other methods. The proposed MMD based framework consistently depicts impressive performance across all the datasets; the generalization error and loss decrease steadily over the training epochs for our algorithm. This is further validated from Table 1 where the MMD-based mini-batch sampling technique achieves the best results in 7 out of the 12 experiments, and the second best results in the remaining 5 experiments. More importantly, as noted in previous research, the *DPP* based selection strategy generates a non-deterministic mini-batch sequence (Joseph et al. 2019; Wang et al. 2019); the *Submodular* subset selection method is also dependent on the network architecture (Joseph et al. 2019). Our algorithm is based on a kernel matrix computation, followed by solving an LP problem; it thus generates a deterministic sequence of representative mini-batches, that is independent of the prediction task and model, corroborating its usefulness in practical applications.

Study of Mini-Batch Size

The goal of this experiment was to study the performance of the mini-batch size on the learning performance. The results with mini-batch size 64, 80 and 100 on the SVHN dataset are shown in Figure 2. The proposed algorithm is compared against the conventional method of selecting the mini-batches randomly from the training data. We note that MMD based selection outperforms *Random Sampling* consistently across all training mini-batch sizes; the performance improvement is more evident for larger batch sizes. This shows the consistency of our framework across different mini-batch sizes.

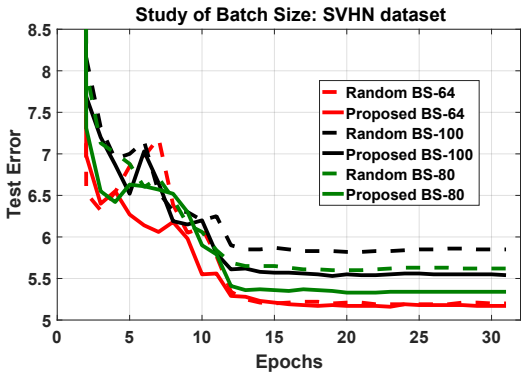


Figure 2: Effect of mini-batch size on learning performance. Best viewed in color.

Study of Kernel Function

The effect of kernel function (used in computing the MMD) on the learning performance was studied in this experiment. Figure 3 depicts the results of Gaussian kernel (with spread 1 and 0.5) and polynomial kernel (with degrees 2 and 3) against *Random Sampling*, on the SVHN dataset. *Random Sampling* depicts marginally better performance than Gaussian kernel with spread 0.5, but is outperformed by all the other kernel parameter settings. Thus, the proposed MMD based batch selection framework outperforms random selection of training batches for a variety of common kernels.

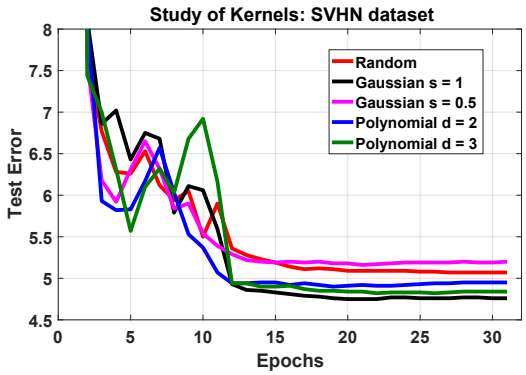


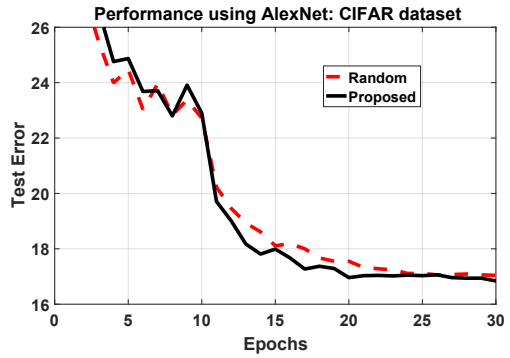
Figure 3: Effect of kernels on learning performance. Best viewed in color.

Performance using Other Deep Learning Architectures

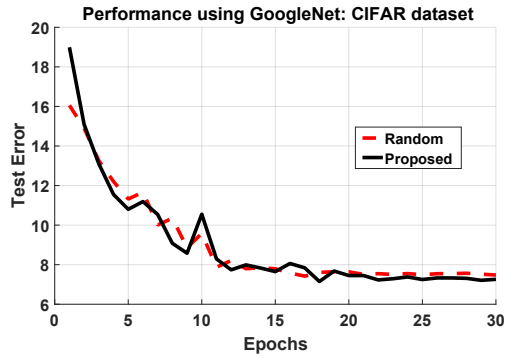
The objective of this experiment was to study the performance of the proposed framework across different common deep learning architectures. The results on the CIFAR-10 dataset using the AlexNet (Krizhevsky, Sutskever, and Hinton 2012) and GoogleNet (Szegedy et al. 2015) architectures are depicted in Figures 4(a) and 4(b) respectively. The results depict a similar pattern as Figure 1(b), with the MMD based mini-batch selection showing overall lower generalization error for both the architectures. This shows the consistency of our framework across different commonly used deep learning architectures.

Performance on Other Learning Tasks

To study its generalizability across learning tasks, we applied the proposed MMD based mini-batch selection framework on a regression problem. The IMDB-Wiki dataset, which contains facial images with age labels of celebrities from the IMDB and Wikipedia databases (Rothe, Timofte, and VanGool 2018) was used for this experiment. We used 7000 samples for training and 3000 samples for testing. The last layer of ResNet was replaced with a single node for the regression application. The results are presented in Figure 5. We note that our framework consistently depicts a lower value of RMSE compared to *Random Sampling* across all the training epochs. This highlights the merit of our deterministic mini-batch sequencing algorithm, which can operate without any knowledge of the underlying prediction task or the network architecture to be used for that task.



(a) Performance using AlexNet. Best viewed in color.



(b) Performance using GoogleNet. Best viewed in color.

Figure 4: Performance using other deep learning architectures. Best viewed in color.

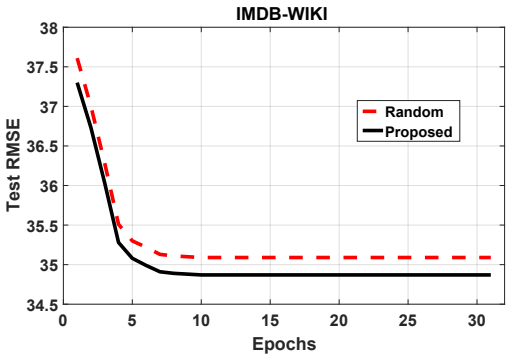


Figure 5: Performance on a regression application (facial age estimation). Best viewed in color.

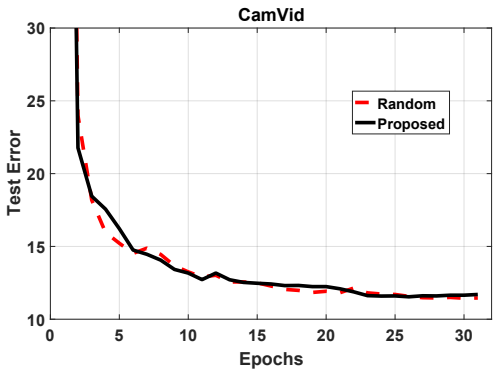


Figure 6: Performance on semantic image segmentation. Best viewed in color.

We also validated the performance of our framework on semantic image segmentation. The CamVid dataset (Brostow et al. 2008) popularly used in image segmentation research was used in this experiment. A DeepLab v3+ network was created based on ResNet-18 and we used 421 images for training and 140 images for testing [2]. We grouped multiple classes from the original dataset to reduce the number of classes from 32 to 11. The pixelwise error over all classes and across all the test images was used as the evaluation metric in this experiment. The results are presented in Figure 6. The proposed framework depicts competitive performance as *Random Sampling*, further corroborating its usefulness in real-world applications.

Conclusion and Future Work

In this paper, we proposed an algorithm to generate a deterministic sequence of mini-batches to train a deep neural network. Our framework is based on minimizing the probability distribution difference (computed using the MMD) between the selected mini-batches and the unselected training samples. The mini-batch selection was reduced to an NP-hard IQP, which was shown to be equivalent to a linear programming problem. Our mini-batch sequencing framework is deterministic and does not depend on the underlying network architecture or the prediction task in question. Our extensive experimental studies corroborated the promise and potential of the framework to improve the generalization capabilities of deep neural networks, over the conventional method of randomly selecting the sequence of training mini-batches.

As part of future work, we plan to study the performance of our framework on other types of deep learning architectures such as region-based CNNs (RCNNs), recurrent neural networks (RNNs) and generative adversarial networks (GANs). We will also explore other strategies to solve the optimization problem in Equation (7) and derive performance bounds on the quality of the solutions.

[2] https://www.mathworks.com/help/vision/examples/semantic-segmentation-using-deep-learning.html

References

Alain, G.; Lamb, A.; Sankar, C.; Courville, A.; and Bengio, Y. 2015. Variance Reduction in SGD by Distributed Importance Sampling. In *arXiv:1511.06481*.

Allen-Zhu, Z. 2018. Katyusha: The First Direct Acceleration of Stochastic Gradient Methods. *Journal of Machine Learning Research (JMLR)* 18.

Badrinarayanan, V.; Kendall, A.; and Cipolla, R. 2015. Seg-

Net: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Bengio, Y.; Louradour, J.; Collobert, R.; and Weston, J. 2009. Curriculum Learning. In *International Conference on Machine Learning (ICML)*.

Borgwardt, K.; Gretton, A.; Rasch, M.; Kriegel, H.; Scholkopf, B.; and Smola, A. 2006. Integrating Structured Biological Data by Kernel Maximum Mean Discrepancy. *Bioinformatics* 22(14).

Brostow, G.; Shotton, J.; Fauqueur, J.; and Cipolla, R. 2008. Segmentation and Recognition using Structure from Motion Point Clouds. In *European Conference on Computer Vision (ECCV)*.

Chattopadhyay, R.; Fan, W.; Davidson, I.; Panchanathan, S.; and Ye, J. 2013a. Joint Transfer and Batch-mode Active Learning. In *International Conference on Machine Learning (ICML)*.

Chattopadhyay, R.; Wang, Z.; Fan, W.; Davidson, I.; Panchanathan, S.; and Ye, J. 2012. Batch Mode Active Sampling based on Marginal Probability Distribution Matching. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*.

Chattopadhyay, R.; Wang, Z.; Fan, W.; Davidson, I.; Panchanathan, S.; and Ye, J. 2013b. Batch Mode Active Sampling Based on Marginal Probability Distribution Matching. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 7(3).

Defazio, A.; Bach, F.; and Lacoste-Julien, S. 2014. SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives. In *Neural Information Processing Systems (NeurIPS)*.

Deng, L.; and Platt, J. 2014. Ensemble Deep Learning for Speech Recognition. In *Interspeech*.

Fan, Y.; Tian, F.; Qin, T.; Bian, J.; and Liu, T. 2017. Learning What Data to Learn. In *International Conference on Learning Representations Workshop (ICLR-W)*.

Gretton, A.; Borgwardt, K.; Rasch, M.; Scholkopf, B.; and Smola, A. 2007. A Kernel Method for the Two-Sample-Problem. In *Neural Information Processing Systems (NeurIPS)*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016a. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Joseph, K.; Teja, V.; Singh, K.; and Balasubramanian, V. 2019. Submodular Batch Selection for Training Deep Neural Networks. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Katharopoulos, A.; and Fleuret, F. 2017. Biased Importance Sampling for Deep Neural Network Training. In *arXiv:1706.00043v2*.

Katharopoulos, A.; and Fleuret, F. 2018. Not All Samples Are Created Equal: Deep Learning with Importance Sampling. In *International Conference on Machine Learning (ICML)*.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images. In *Technical Report, University of Toronto*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Neural Information Processing Systems (NIPS)*.

Kumar, M.; Packer, B.; and Koller, D. 2010. Self-Paced Learning for Latent Variable Models. In *Neural Information Processing Systems (NeurIPS)*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. In *Proceedings of IEEE*.

Lei, L.; Ju, C.; Chen, J.; and Jordan, M. 2017. Non-convex Finite-Sum Optimization Via SCSG Methods. In *Neural Information Processing Systems (NeurIPS)*.

Loshchilov, I.; and Hutter, F. 2016. Online Batch Selection for Faster Training of Neural Networks. In *International Conference on Learning Representations Workshop (ICLR-W)*.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. 2011. Reading Digits in Natural Images with Unsupervised Feature Learning. In *Neural Information Processing Systems Workshop (NeurIPS-W)*.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2017. Faster R-CNN: Towards Real-time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 39(6).

Rothe, R.; Timofte, R.; and VanGool, L. 2018. Deep Expectation of Real and Apparent Age from a Single Image without Facial Landmarks. *International Journal of Computer Vision (IJCV)* 126(2 - 4): 144 – 157.

Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2016. Prioritized Experience Replay. In *International Conference on Learning Representations (ICLR)*.

Sriperumbudur, B.; Gretton, A.; Fukumizu, K.; Scholkopf, B.; and Lanckriet, G. 2010. Hilbert Space Embeddings and Metrics on Probability Measures. *Journal of Machine Learning Research (JMLR)* 11: 1517 – 1561.

Steinwart, I. 2001. On the Influence of the Kernel on the Consistency of Support Vector Machines. *Journal of Machine Learning Research (JMLR)* 2.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going Deeper with Convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tang, Y.; and Huang, S. 2019. Self-Paced Active Learning: Query the Right Thing at the Right Time. In *AAAI Conference on Artificial Intelligence*.

Vempala, S. 2004. The Random Projection Method. In *Americal Mathematical Society*.

Venkateswara, H.; Eusebio, J.; Chakraborty, S.; and Panchanathan, S. 2017. Deep Hashing Network for Unsupervised Domain Adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wang, S.; Bai, W.; Lavania, C.; and Bilmes, J. 2019. Fixing Mini-batch Sequences with Hierarchical Robust Partitioning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Wang, Z.; and Ye, J. 2013. Querying Discriminative and Representative Samples for Batch Mode Active Learning. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*.

Wu, C.; Manmatha, R.; Smola, A.; and Krahenbuhl, P. 2017. Sampling Matters in Deep Embedding Learning. In *IEEE International Conference on Computer Vision (ICCV)*.

Zhang, C.; Kjellstrom, H.; and Mandt, S. 2017. Determinantal Point Processes for Mini-Batch Diversification. In *Uncertainty in Artificial Intelligence (UAI)*.

Zhang, C.; Oztireli, C.; Mandt, S.; and Salvi, G. 2018. Active Mini-Batch Sampling using Repulsive Point Processes. In *arXiv:1804.02772*.

Zhao, P.; and Zhang, T. 2015. Stochastic Optimization with Importance Sampling for Regularized Loss Minimization. In *International Conference on Machine Learning (ICML)*.