JMLR: Workshop and Conference Proceedings 101:300–315, 2019 ACML 2019

Efficient Diversified Mini-Batch Selection using Variable High-layer Features

Wanming Huang WANMING.HUANG@STUDENT.UTS.EDU.AU
Faculty of Engineering and IT
University of Technology Sydney
81 Broadway, Sydney, Australia

Richard Yi Da Xu YIDA.XU@UTS.EDU.AU
Faculty of Engineering and IT
University of Technology Sydney
81 Broadway, Sydney, Australia

Ian Oppermann IANOPPER@OUTLOOK.COM
NSW Data Analytics Centre
2-24 Rawson Pl, Sydney, Australia

Editors: Wee Sun Lee and Taiji Suzuki

Abstract

Stochastic Gradient Descent (SGD) has been widely adopted in training Deep Neural networks of various structures. Instead of using a full dataset, a so-called *mini-batch* is selected during each gradient descent iteration. This aims to speed up the learning when a large number of training data is present. Without the knowledge of its true underlying distribution, one often samples the data indices uniformly. Recently, researchers applied a diversified mini-batch selection scheme through the use of Determinantal Point Process (DPP), in order to avoid having highly correlated samples in one batch (Zhang et al. (2017)). Despite its success, the attempts were restrictive in the sense that they used fixed features to construct the Gram-matrix for DPP; using the raw or fixed higher-layer features limited the amount of potential improvement over the convergence rate. In this paper, we instead proposed to use variable higher-layer features which are updated at each iteration when the parameter changes. To avoid the high computation cost, several contributions have been made to speed up the computation of DPP sampling, including: (1) using hierarchical sampling to break down a single DPP sampling with large Gram-matrix into many DPP samplings of much smaller Gram-matrix and (2) using Markov k-DPP to encourage diversity across iterations. Empirical results show a much more diversified mini batch in each iteration in addition to a much improved convergence compared with the previous approach.

Keywords: Image classification, Sampling methods

1. Introduction

Mini-batch and SGD are common tools used in machine learning. Often, the size of training data becomes so large that it is impractical to render batch optimization (i.e., using the entire dataset in every iteration). For example, Krizhevsky et al. (2012) trained a CNN to classify 1.2 million images from ImageNet. Sutskever et al. (2014) built a sequence-

© 2019 W. Huang, R.Y.D. Xu & I. Oppermann.

to-sequence neural translation model with 12 million sentences that contained 348 million French words and 304 million English words. Therefore, mini batch training was employed to reduce the communication cost and parallelize the learning process (Li et al. (2014)).

Despite its wide usage, the most standard form of selecting mini-batches in SGD is to sample data uniformly in each iteration and perform an update as follows:

$$w_{t+1} \leftarrow w_t - \frac{\eta}{n} \sum_{i=1}^{n} \nabla \phi_i(w_t) \tag{1}$$

where w_t is the parameter at time step t, ϕ is the loss function, η is the learning rate and n is the size of the mini batch. As standard SGD employs uniform sampling, the stochastic gradient is an unbiased estimate of the true gradient. It nonetheless, introduces variance between iterations, which negatively affects the performance (Zhao and Zhang (2015)).

Using large mini batches can mitigate the problem by reducing the variance but it also slows down the actual convergence (Byrd et al. (2012)). There have been several attempts made to optimize the performance of mini batch training: Li et al. (2014) added a conservative constraint to the loss function in order to control differences between parameters across iterations and γ is a coefficient:

$$w_t = argmin[\phi(w) + \frac{\gamma}{2}||w - w_{t-1}||_2^2] \tag{2}$$

Johnson and Zhang (2013) proposed a method named Stochastic Variance Reduced Gradient (SVRG) that replaced the training target with a new random vector that had the same expectation but a smaller variance. Alain et al. (2015) applied distributed importance sampling where the sampling weight was proportional to the L2-norm of the gradient. Gopal (2016) proposed a similar idea by separating the data into bins using side-information and maintaining the distribution at a class level rather than an instance level. Yin et al. (2017) studied the role of dissimilarity between concurrent gradient updates in the mini-batch SGD performance and proposed a data-dependent scheme to choose the batch size. Xie et al. (2016) analyzed one-hidden-layer neural network and proposed a novel regularization function that could potentially lead to better generalization.

Other researchers reduced variances through modifying sampling methods. Zhao and Zhang (2014) proposed an algorithm named Stochastic Gradient Descent with Stratified Sampling (SGD-ss) that applied clustering to separate the dataset into clusters before uniformly sampling for mini batches within each cluster. In addition, the method required data in each cluster belonging to the same category. The weight update formula now becomes:

$$w_{t+1} \leftarrow w_t - \frac{\eta}{n} \sum_{i=1}^{k} \sum_{s=1}^{b} \nabla \phi_s(w_t) \tag{3}$$

where k is the total number of clusters and b is the number of samples to be selected within each cluster.

While most research in the space of SGD variance reduction assumes the loss function to be convex and smooth, such as in Shamir (2011), recent works by Johnson and Zhang (2013) and Reddi et al. (2016) also proved that SVRG can be applied to non-convex functions.

Although these works can theoretically accelerate the training process, there is still the requirement for an underlying data distribution to be known for the weight updates.

In order to improve the efficiency of SGD convergence without the assumption of any underlying data distributions, Zhang et al. proposed a strategy called DM-SGD that applied k-DPP to mini-batch sampling.

Given fully-connected feature vectors W and corresponding one-hot labels H, DM-SGD calculates the feature vector for each data sample as a weighted concatenation between the two, i.e. feature vectors are calculated as:

$$F = [(1-w)W \quad wH],\ 0 \leq w \leq 1 \tag{4}$$

The Gram-matrix L for the sampling is defined as $L = FF^\top$. In addition, authors proved that: For all data points x_i and x_j and all parameters θ, if:

$$\forall_{i \neq j} : C_{ij} g(x_i, \theta)^\top g(x_j, \theta) < 0 \tag{5}$$

Then DM-SGD has a lower variance than SGD. Here, g represents the gradient, C_{ij} is the correlation between the two data points, which is negative when data points are similar and positive when dissimilar. Therefore, applying k-DPP sampling guarantees variance reduction.

Authors demonstrated the performance on several datasets including the MNIST (LeCun et al. (1998)) and Oxford flower 102 datasets (Nilsback and Zisserman (2008b)). In the MNIST dataset, authors used the raw image pixels to define feature vectors. In Oxford Flower 102 dataset, the feature vector for each image was defined using the first fully-connected layer of the pre-trained VGG-16 network (Simonyan and Zisserman (2014)). Authors then trained a linear softmax classification with off-the-shelf CNN features. There are two drawbacks in DM-SGD:

Firstly, when raw input features are used, it measures diversities in terms of the Euclidean distance in the input space. This is prohibitive; and in fact, the very reason neural network is applied, is to transform data non-linearly from its original space into a layer before the output, such that the features now becomes a lot more expressive. Therefore, if any features are to be used to guide the mini-batch selection, we need to use them before the last layer instead of the first input. However, this also creates another problem in that the network parameters are not static and are updated throughout the iterations; Therefore, we must use the updated features at each iteration as opposed to what DM-SGD proposes.

Secondly, mini-batch selection using DPP may become prohibitively slow when the entire set of data Ω is used to compute its Gram-matrix, even if this operation is only computed once. Therefore, the "fixed" feature approach is somewhat only theoretically plausible.

For these reasons, we must update the features to compute the Gram-matrix at each iterations. Given sampling DPP once is already far too much computation, sampling at each iteration with an updated Gram-matrix would further increase the already-infeasible complexities. Therefore, it is natural to use an approximated DPP to dramatically improve its computation time. By observing this in the last or higher layer, we found that features within each class have a higher tenancy to stay closer and data between classes tend to separate. Therefore we chose the last fully-connected layer to construct the Gram-matrix in our algorithms. We also took the so-called Class-Dependent DPP sampling by using

hierarchical sampling to break down a single DPP sampling with large Gram-matrix into many DPP sampling of much smaller Gram-matrix. In the lower hierarchy, each sampling is to be performed on data within its own class. To further improve the computational efficiency, we also used Markov k-DPP to encourage diversity across iterations. Accordingly, we propose five separate mini-batch selection algorithms, which are explained in section 3; We also show their empirical effectiveness, where these mini-batch selection schemes are applied to classification problems on the Oxford 102 (Nilsback and Zisserman (2008a)), Stanford Dogs (Khosla et al. (2011)) and Caltech 101 dataset (Fei-Fei et al. (2007)).

The rest of the paper is organized as follows. Section 2 reviews approaches to sample DPP. Section 3 explains the three approaches we propose to enhance the performance of mini batch training. Section 4 demonstrates the results and comparison between algorithms. Section 5 is the conclusion.

2. Background

For completeness, we briefly outline DPP, which is a random process used to model a subset $\mathbf{Y}$ selected from a base set $\mathcal{Y}$. In particular, DPP encourages $\mathbf{Y}$ to contain a diverse set of items, and is therefore applied to some sampling and summarization tasks where diversity is preferred, for example, detecting people and their poses in an image (Kulesza et al. (2012)).

Consider a random subset $\mathbf{Y}$ drawn from $\mathcal{Y}$ according to $\mathcal{P}$, for every subset $A \subseteq \mathcal{Y}$:

$$\mathcal{P}(\mathbf{Y} \supseteq A) = \det(K_A) \tag{6}$$

where K is called the marginal kernel which is a real, symmetric, positive semidefinite $N \times N$ matrix indexed by elements of $\mathcal{Y}$.

DPPs can be constructed alternatively with a real, symmetric matrix L indexed by the element of $\mathcal{Y}$ instead of the marginal kernel K, in which case L is called L-ensembles. Let L be the Gram-matrix, constructed from B, the feature vector for each element in the base set $\mathcal{Y}$, i.e., $L = B^\top B$:

L-ensembles directly specifies the probability for a possible subset A as follows:

$$\mathcal{P}_L(\mathbf{Y} = A) = \frac{\det(L_A)}{\det(L + I)} \tag{7}$$

where L_A is a part of L that is indexed by elements in A, and I is the identity matrix.

2.1. K-DPP

k-DPP models the distribution over subsets with size k from $\mathcal{Y}$ Kulesza and Taskar (2011), which ensures that the number of data sampled is of a fixed size.

$$P_L^k(Y) = \frac{\det(L_Y)}{\sum_{|Y'|=k} \det(L_{Y'})} \tag{8}$$

2.2. Markov DPP

Markov DPP further encourages diversity between two subset selections. Suppose Y_t is the subset sampled at timestep t, and Y_{t-1} is the subset sampled at timestep t, Markov DPP

tries to maximize the conditional probability of sampling Y_t given Y_{t-1}. In other words, it tries to make the sample diverse from previous samples as shown below.

The sampling strategies for normal DPP sampling, k-DPP sampling and Markov k-DPP sampling are demonstrated in Kulesza et al. (2012) and Affandi et al. (2012).

3. Methodology

Deep Neural networks, such as CNN can be thought as a process of data projection to serve the functions of its final layer, for example, a Softmax. The learning requires back-propagation, which can be very computational when the data size is large. For this reason, a mini-batch is used instead at every iteration to approximate a "batch method":

$$\theta_{t+1} = \theta_t - \eta \mathbb{E}[\nabla f(x, \theta)] \tag{9}$$

by using:

$$\{x^{(i)} \sim p(x)\} \qquad \theta_{t+1} = \theta_t - \eta \frac{1}{M_{mb}} \sum_{i=1}^{M_{mb}} \nabla f(x^i, \theta) \tag{10}$$

where M_{mb} is the mini-batch size. In most instances, we do not know the underlying distribution $p(X)$. Therefore, a uniform sampling of data indices has been used instead. In our work, we use DPP to select a diversified set of data points within each category for each SGD iteration.

Although it may be possible to allow the size of mini-batches to vary in each iteration, it is practical to keep them the same across iterations. Therefore, in this work, we used a k-DPP, where each DPP draw generates a subset having equal cardinality k.

3.1. Gram-Matrix Construction from variable higher-layer features

DPP sampling requires a Gram-matrix which defines similarities between data points. The past literature offers two forms of features to construct a Gram matrix, namely the *raw* feature and *fixed higher layer* feature.

In terms of the *raw* feature, it uses the input directly as feature vectors. For example, DM-SGD (Kulesza et al. (2012)) uses the Gram-matrix from raw MNIST image pixels which are first reshaped into 1D. This method does not require additional feature extractions, and the eigen-decomposition for the matrix only needs to be calculated once without the presence of neural network.

In terms of *fixed higher layer* feature, it is obtained by feeding each data sample to a pre-trained deep neural network such as VGG-16. In the same way as the raw data, the Gram-matrix is not updated during training.

3.1.1. Variable Higher-Layer Features and the Advantages

The first two constructions are computationally effective since the Gram-matrix and eigen-decomposition only need to be computed once. However, using *raw* feature, measuring diversities in terms of the Euclidean distance in the input space can be mis-leading as the purpose of neural network is to project data from its original space to a layer before output,

to better serve the function of the output layer. Therefore, we ought to use data of the last (or higher) layer instead of the input. Since the parameters changes over iterations, using *fixed higher feature* to construct a fixed Gram-matrix, does not reflect the parameter update and makes it entirely dependent on the initial parameter guess.

For this reason, we used the output from the last fully-connected layer as the feature vector to construct the Gram-matrix. These features are recomputed since the network is updated through back-propagation. Thus we name them the *variable higher-layer* feature.

We demonstrated the advantage of using *variable higher-layer* feature, and its remarkable expressive power over using *raw* or *fixed higher-layer* features. In Figure 1, we plotted the feature vectors used to construct the three Gram-matrices for the Oxford Flower 102 datasets. In terms of *variable higher-layer* features, we used the value of the last iteration.

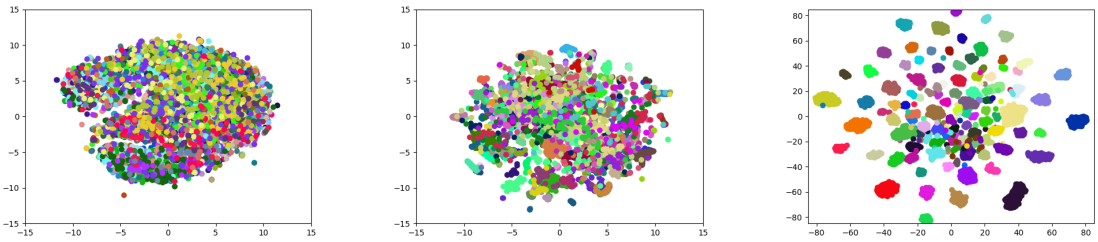


Figure 1: A visualization of three feature vectors being used for constructing Gram-matrix. From left to right, each figure represents *raw*, *fixed higher-layer* and *variable higher-layer* features respectively. Each category is represented in a unique color. Features are dimension reduced using t-SNE.

The figure shows that *raw* or *fixed higher-layer* features provide very high intra-class variances as data of the same class tend *not* to be close. On the contrary, *variable higher-layer* feature have much lower intra-class variance. Numerically, we measured the Calinski-Harabaz score for the three feature vectors as in Table 1. The Calinski-Harabaz score is defined as a ratio between the within-cluster dispersion and the between-cluster dispersion. A higher score shows dense and well separated clusters, and in a classification problem, the class labels of training data provide the cluster information. Not surprisingly, the *variable higher-layer* feature has much higher Calinski-Harabaz score than the other two.

	Raw pixels	Fixed Features	Variable Features
Oxford 102	10.3827	26.7641	78.4164

Table 1: Calinski-Harabaz score for *raw*, *fixed higher-layer* and *variable higher-layer* features in Oxford 102.

3.1.2. Computationally-Feasible Way to Construct Variable Gram-Matrix

Knowing that we ought to construct Gram-matrix using *variable higher-layer* features has barely solved any problems, as generating random samples from a DPP with a large Gram-matrix can be prohibitively expensive, since the operation involves eigen-decomposition. For

these reasons, we proposed five different mini-batch approximation schemes below, which is also the key contribution of our paper.

In each of these methods the Gram-matrix sizes are drastically reduced, allowing the algorithm to significantly accelerate training. Remarkably, whilst we update Gram-matrix construction at each iteration, our experiment shows that our method achieves much quicker convergence compared with previous approaches. In this paper, we provide two broad categories of approaches, namely, the Full-set DPP sampling and Class Dependent DPP sampling.

3.1.3. Full-set DPP Sampling

In this category, one does not take into account of the class label information when sampling DPP. The subsets are selected independent of their labels. There are two versions of algorithms, which we named FULL-SINGLE-DPP and FULL-MARKOV-DPP respectively:

FULL-SINGLE-DPP In here, in each i^{th} iteration, one first samples a larger subset $\tilde{S}$ uniformly from the full training set Ω, before sampling a random subset S_i using k-DPP from $\tilde{S}$. Compared with DPP sampling from Ω directly, this method has drastically sped up its computation. The choice of $|\tilde{S}|$ is arbitrary as long as it is larger than K: we have chosen it to be a multiple of K. Obviously, a larger $|\tilde{S}|$ results in a longer duration, but at the same time it also increases its performance in terms of a faster convergence and accuracy as well as the Calinski-Harabaz score. Therefore, one may select an optimized and appropriate trade-off value given the size of the dataset. This approach is summarized in Algorithm 1.

In addition, in order to collect the feature matrix W_i, a feed-forward process is required for $\tilde{S}$. The Gram-matrix for the k-DPP sampling is thus calculated as $W_i W_i^\top$.

Algorithm 1 FULL-SINGLE-DPP sampling

Data: Data Ω, mini-batch size K
for i = *1 to MaxIter* **do**
  $\tilde{S} \sim U(\Omega)$
  Obtain the feature matrix W_i in feed-forward($\tilde{S}$)
  $S_i \sim$ k-DPP($W_i W_i^\top$)
  Perform updates according to equation 10
end

FULL-MARKOV-DPP FULL-SINGLE-DPP only provides samples with diversities within a single iteration. However, diversities are also needed between consecutive iterations. For this reason, we also proposed the cross-iteration diversity sampling scheme which we named FULL-MARKOV-DPP. In here, it uses conditional Markov k-DPP to perform sampling on S_i conditioning on all previous samples $\{S_{t<i}\}$. Therefore, in addition to encouraging samples in each mini batch to be as diverse as possible, it also encourages samples to be more diverse across the iterations. Details are in Algorithm 2.

The original Markov k-DPP works on a fixed Gram-matrix across time steps. However, as we apply sampling on a set of randomly selected data points, the Gram-matrix also needs to be re-calculated. From the second iteration, the Gram-matrix is calculated from

Algorithm 2 FULL-MARKOV-DPP sampling

Data: Data Ω, mini-batch size K
for *i = 1 to MaxIter* **do**
  $\tilde{S} \sim U(\Omega)$
  Obtain the feature matrix W_i in feed-forward($\tilde{S}$)
  if *iter* $= 1$ **then**
    $S_i \sim$ k-DPP($W_i W_i^\top$)
  else
    $S_i \sim$ markov-kDPP($W_i W_i^\top, \{S_{t<i}\}$)
  end
  Perform updates according to equation 10
end

the uniform random subset $\tilde{S}_i$ and conditioning factors $\{S_{t<i}\}$ that are mini batch samples from the previous iterations.

3.1.4. Class-Dependent Stochastic DPP

The second broad category of our proposed methods is Class-Dependent stochastic DPP. Given categories $1, \ldots, C$, it takes a hierarchical approach to sampling: in the *first hierarchy*, a sampling is first performed within each class to obtain d initial data points. In the *second hierarchy*, a second k-DPP is applied to sample the final mini-batch with size K. This hierarchical approach of DPP sampling breaks down a single DPP sampling with large Gram-matrix into many DPP sampling of much smaller Gram-matrix, hence enhancing its performance dramatically.

We propose three methods in this category as shown in Algorithm 3, differing only in terms of the *first hierarchy*: **CLASS-SINGLE-DPP** and **CLASS-MARKOV-DPP** applies k-DPP and Markov k-DPP respectively as in the Full-set DPP sampling case in section 3.1.3.

CLASS-IMBALANCE: this is to handle scenarios where there are class-imbalance problems. Some classes may have significantly more data points than other sets, therefore, DPP sampling is only applied to "smaller" class data, and uniform sampling is applied to the rest. In here, Markov k-DPP is only applied to classes with numbers of samples below a threshold δ. The Class-Dependent Stochastic DPP is summarized in Algorithm 3.

where

$$\text{SELECT}(W^c, \Omega^c) = \begin{cases} \text{kDPP}(W^c W_c^\top) & \textbf{CLASS-SINGLE-DPP} \\ \text{markov-kDPP}(W^c W_c^\top, \{S_{t<i}\}) & \textbf{CLASS-MARKOV-DPP} \\ \begin{cases} \text{markov-kDPP}(W^c W_c^\top), & n_c < \delta \\ U(\Omega^c), & n_c \geq \delta \end{cases} & \textbf{CLASS-IMBALANCE} \end{cases}$$

In conclusion, the differences between the proposed methods relative to DM-SGD lie in the following perspectives.

Algorithm 3 Class-Dependent Stochastic DPP

Data: Data Ω, number of samples in each categories $\{n_1, \ldots, n_C\}$
for $i = 1$ *to* $MaxIter$ **do**
  for $c = 1$ *to* C **do**
    obtain W_i^c in feed-forward(Ω^c)
    $S_i^c \sim$SELECT(W_i^c, Ω^c))
  end
  Obtain the feature matrix W_i by concatenating features of $[S_i^1, \ldots S_i^c]$
  sample $S_i \sim$ k-DPP($W_i W_i^\top$)
  Perform updates according to equation 10
end

1. The feature vectors W used to compute the Gram matrix are not coming from *raw* or *fixed higher-layer* features and instead they are *variable higher-layer* features, which are updated as parameters change in each iteration.

2. Unlike DM-SGD as in Equation 4, the label information is not used in defining the feature vector for each data sample.

3. Various techniques are employed to avoid DPP sampling on the full training set which significantly saves the time cost in performing sampling.

4. Markov k-DPP is employed to encourage diversity across iterations as well.

In terms of the variance reduction, Zhang et al. (2017) proved that DM-SGD, which employs k-DPP sampling, has a lower gradient variance than vanilla SGD. The same proof can be easily applied to our proposed method. The proposed methods also employ distance-dependent similarity kernel and sample diverse points, thus naturally inherit the properties of k-DPP.

3.2. Computation Complexity Analysis

In this section, we give the sampling duration computation complexity of the proposed methods and compare it with DM-SGD.

Assume that the entire dataset Ω contains N data samples and each mini-batch contains K items. In each iteration, DM-SGD requires $O(N^3)$ to perform the eigen-decomposition and $O(NK + K^2)$ to perform the sampling in each iteration. The total time cost after I iterations is:

$$T_{\text{DMSGD}} = O(N^3 + (NK + K^2) \cdot I) \tag{11}$$

Full-set DPP performs sampling on a much smaller subset S_i in iteration i. Assume that the size of S_i is T of the original set Ω and $T \in (0, 1]$ (e.g. $T = 1/3$). The proposed methods do not require the eigen-decomposition to be performed on the $N \times N$ similarity kernel prior to the training. The computation cost in each iteration is $O((NT)^3 + NTK + K^2)$. The total computation cost for I iterations is:

$$T_{\text{FULL\_SET}} = O\left(\left((NT)^3 + NTK + K^2\right) \cdot I\right) \tag{12}$$

If we compare the two values T_{DMSGD} and $T_{\text{FULL\_SET}}$, we can easily derive that the condition for $T_{\text{DMSGD}} > T_{\text{Full\_SET}}$ is:

$$(1 - T^3 I) \cdot N^2 + (1 - T) \cdot KI > 0 \tag{13}$$

As $(1 - T) \cdot KI$ is always positive, if we only focus on the first term, then as long as $T^3 < 1/I$, we can have $T_{\text{DMSGD}} > T_{\text{Full\_SET}}$. For example, when I is $10,000$, as long as S_i is smaller than $1/22$ of the entire dataset, FULL-set DPP is more time efficient than DM-SGD. This is quite an easy constraint to be met. Taking the second term into consideration should further loosen this constraint.

Class-dependent DPP is slightly more complex as it performs separate sampling in each category of data. If we simplify the problem setting by assuming that C categories equally split the entire dataset, and the subset S_i^c sampled from category c contains T of total samples in this category, the total computation cost is

$$T_{\text{Class-dependent}} = \left(\left((\frac{N}{C})^3 + (\frac{N}{C}) \cdot (\frac{N}{C} \cdot T) + (\frac{N}{C} \cdot T)^2 \right) \cdot C + (NT)^3 + (NT) \cdot K + K^2 \right) \cdot I \tag{14}$$

Therefore, the required condition for $T_{\text{DMSGD}} > T_{\text{Class-dependent}}$ is:

$$\left(\frac{I}{C^2} + I \cdot T^3 - 1 \right) \cdot N^2 + \left(\frac{T}{C} + \frac{T^2}{C} \right) \cdot NI + (T - 1) \cdot KT < 0 \tag{15}$$

If we take the Stanford Dogs dataset, whose statistics are shown in Table 2, as an example and we set $K = 50$, $T < 1/32$ is required such that Class-dependent DPP is more computationally efficient than DM-SGD.

In Figure 2, we plot the training duration, which includes the sampling, feed-forward and back-propagation duration, over different T when the total iterations I is $10,000$. All experiments are performed on the Oxford 102 dataset using the full VGG-16 network. DMSGD2 is the baseline model, the details and explanation of which can be found in Section 4.2. It is clear to see that the computation cost of the proposed algorithms is lower when T is larger. When $T < 1/10$, all four proposed methods are more efficient than the baseline model.

4. Experiments

4.1. Datasets

Experiments were performed on several image classification problems using the Oxford 102 Flower (Nilsback and Zisserman (2008a)), Stanford Dogs (Khosla et al. (2011)), Caltech 101 datasets (Fei-Fei et al. (2007)) and MNIST dataset. The statistics of these datasets are as below.

Following the original paper from Zhang et al. (2017). The training of Oxford 102 Flower dataset was performed on the 6,149 testing images and the testing was performed

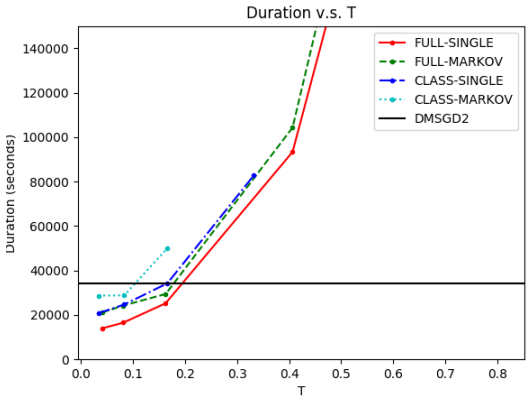


Figure 2: Duration over T on the Oxford 102 dataset

Dataset	Oxford 102			Stanford Dogs			Caltech 101	MNIST	
#categories	102			120			101	10	
#samples	train	test	val	train	test	val	9,145	train	test
	1,020	6,149	1,020	12,000	8,580	8,580		60,000	10,000

Table 2: Statistics for Datasets

on the 1,020 training images, i.e. the training and testing sets were interchanged in our experiments. In addition, as Caltech 101 dataset does not provide the train-test split, we randomly selected 80% of the data from each category as the training set and we equally split the rest of the data in each category as the testing and validation set. Lastly, performing eigen-decomposition on the entire Gram-matrix generated from MNIST image features is extremely time consuming. Therefore, we randomly selected 20% from each category to perform the training, another 20% as the validation set. We used the original test set for the testing. The results were collected from multiple experiments with random selections.

4.2. Baseline Models

We compared the proposed methods against two baseline models on the first three datasets. The first baseline model is the DM-SGD from Zhang et al. (2017), which uses the off-the-shelf last fully-connected features and the label information to construct the Gram-matrix. As the design from the original paper only trains the layer before the softmax layer, we also compared the proposed algorithms with the second baseline model, which we named as "DMSGD2". DMSGD2 allows all layers of deep neural networks to be trained while the sampling still relies on the off-the-shelf last fully-connect features. For both DM-SGD and DMSGD2, Equation 4 is applied to calculate the feature vector F for each data sample. The proposed methods construct the Gram-matrix using the last fully-connected features as shown in the pseudo-codes.

In terms of the MNIST dataset, we compared the proposed methods against one baseline model DM-SGD. Following Zhang et al. (2017), DM-SGD uses the raw image pixels and the label information to construct a RBF kernel for the sampling. It also trains a full 5-layer

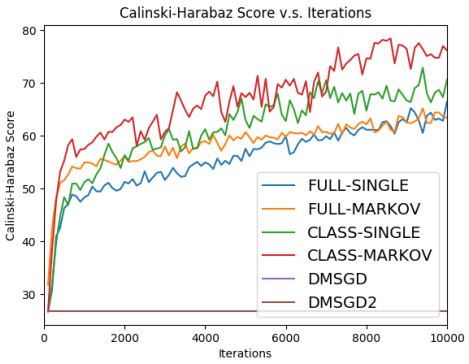
CNN network instead of only the last layer as on the previous datasets. The proposed method still uses the Gram-matrix generated from the last fully-connected features and trains the same 5-layer network.

4.3. Performance Evaluation on Oxford 102 Flower

This section evaluates the performances of the proposed methods, by comparing them against the baseline models.

Comparisons were made in terms of accuracy, Calinski-Harabaz scores and time costs. Following the original paper, the results are demonstrated via fine tuning the pre-trained VGG-16 model. Experiments were performed with $10,000$ iterations, learning rate is fixed as $1e-5$ and the batch size is set to be $K = 50$. The same setting is applied to all datasets.

Experiment	Calinski-Harabaz Score
FULL-SINGLE	66.4588 ± 1.6767
FULL-MARKOV	68.1966 ± 2.5753
CLASS-SINGLE	72.8985 ± 3.5527
CLASS-MARKOV	78.4164 ± 2.0534
DM-SGD	26.7737
DMSGD2	26.7737



Experiment	Accuracy	Duration(seconds)
FULL-SINGLE	0.9010 ± 0.0027	$17,587.096 \pm 77.5$
FULL-MARKOV	0.9080 ± 0.0065	$24,867.031 \pm 96.0$
CLASS-SINGLE	0.9157 ± 0.0027	$24,671.139 \pm 106.3$
CLASS-MARKOV	0.9333 ± 0.0040	$28,760.66 \pm 151.6$
DM-SGD	0.8852 ± 0.0038	$31,257.426 \pm 82.8$
DMSGD2	0.8983 ± 0.0044	$33,994.164 \pm 143.5$

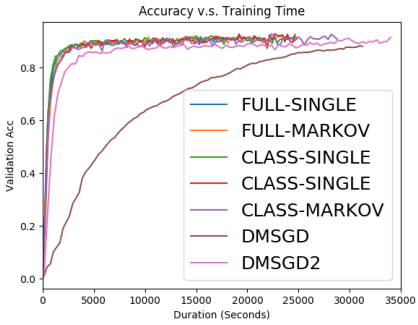


Table 3: Demonstration of performances on the Oxford 102 Flower dataset. The top row shows the Calinski-Harabaz Score of the training set over iterations and the final score. The bottom row shows the validation accuracy over the training duration in addition to the final testing accuracy and total duration.

In terms of the accuracy, Table 3 shows that the proposed algorithms were able to achieve faster convergence rates and higher final testing accuracies. The proposed methods achieved up to 3.50% higher accuracy than baseline models.

In addition, compared with baseline models which used fixed Gram-matrices, we were able to achieve a much higher Calinski-Harabaz score during training.

Table 3 also reports validation accuracies over time costs and the total training duration. The displayed time costs for the proposed methods include both sampling and back-propagation duration in each iteration while omitting the data processing duration.

Results show that all four proposed methods spent less time in sampling than the DM-SGD, even when the eigen-decomposition needed to be performed in each iteration. In particular, FULL-SINGLE spends 56.27% time costs of the original DM-SGD and achieves a higher testing accuracy.

Compared among the four proposed methods on the four datasets, class dependent stochastic sampling achieves a better performance than full-set sampling on all four datasets. Methods that employs markov-kDPP performs better than using k-DPP.

4.4. Performance Evaluation on the Stanford Dogs Dataset

Table 4 reports the validation accuracy over training iterations and the training duration on the Stanford Dogs Dataset. The best performed CLASS-MARKOV were 5.5% higher in accuracy than DM-SGD and 3.0% higher than DMSGD2. In terms of the training duration, all four proposed methods require less training duration compared to DM-SGD. It is clear in the figure that the proposed algorithms have a faster learning curve and a clear advantage in computation efficiency.

Experiment	Accuracy	Duration(seconds)
FULL-SINGLE	0.7738 ± 0.0228	$25,268.7 \pm 110.1$
FULL-MARKOV	0.8114 ± 0.0135	$29,497.8 \pm 83.5$
CLASS-SINGLE	0.8130 ± 0.0072	$30,831.7 \pm 84.8$
CLASS-MARKOV	0.8165 ± 0.0032	$36,814.6 \pm 78.5$
DMSGD	0.7612 ± 0.0115	$47,817.7 \pm 118.0$
DMSGD2	0.7867 ± 0.0014	$49,763.4 \pm 79.1$

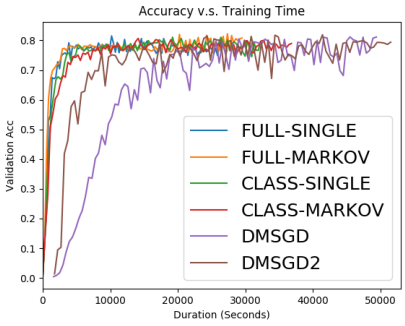


Table 4: Performances for the Stanford Dogs Dataset. The table reports the final testing accuracy and training duration, the right graph shows the validation accuracy over the training duration.

4.5. Performance Evaluation on the Caltech 101 Dataset

Table 5 reports the validation F_1 score over training iterations and the training duration on the Caltech 101 Dataset. Since the dataset is highly imbalanced, the results are measured by F_1 score rather than accuracy. The best performed CLASS-SINGLE results in a 7.2% higher F_1 than DM-SGD and 2.4% higher than DMSGD2. In terms of the training duration, FULL-SINGLE requires approximately of the 49.8% time cost compared to DM-SGD, and all proposed methods spent less time than the two baseline models. It is clear to see that the proposed methods achieve a better performance on this highly imbalanced dataset.

Experiment	F_1	Duration(seconds)
FULL-SINGLE	0.9165 ± 0.0317	$20,231.5 \pm 96.6$
FULL-MARKOV	0.9182 ± 0.0116	$20,464.8 \pm 108.0$
CLASS-SINGLE	0.9232 ± 0.0093	$26,548.0 \pm 82.1$
CLASS-MARKOV	0.9221 ± 0.0023	$26,809.6 \pm 143.3$
CLASS-IMBALANCE	0.9097 ± 0.0359	$26,250.9 \pm 65.6$
DM-SGD	0.8515 ± 0.0102	$40,604.2 \pm 84.5$
DMSGD2	0.8990 ± 0.0092	$45,491.3 \pm 167.0$

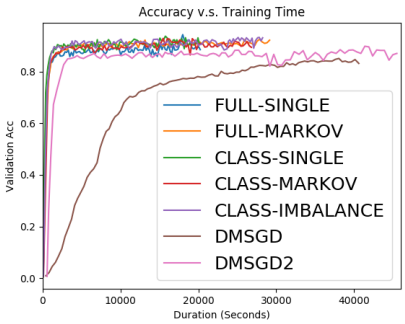


Table 5: Performances on the Caltech 101 dataset. The table reports the final F_1 score and training duration, the right graph shows the validation accuracy over the training duration.

In addition, it can be noted that class-dependent stochastic sampling methods perform better than the full-set sampling methods on this dataset, while CLASS-MARKOV performs slightly worse than CLASS-SINGLE in terms of the testing F_1. This is because images in the CALTECH 101 dataset has high intra-class variance. While class-dependent sampling performs seperate sampling across categories, further encouraging diversity across iterations does not necessarily improve the performance as on other datasets.

4.6. Performance Evaluation on MNIST

Table 6 reports the performance on the MNIST dataset. The proposed method still outperforms DM-SGD in terms of the final testing accuracy. As for the computational cost, class-dependent DPP actually require a longer duration. This finding is also supported by our analysis in Section 3.2, as MNIST only has 10 categories. On the contrary, the proposed full-set DPP methods only require less than 30% total training duration of DM-SGD.

Experiment	Accuracy	Duration (seconds)
FULL-SINGLE	0.9704 ± 0.0025	$6,841.92 \pm 163.57$
FULL-MARKOV	0.9703 ± 0.0011	$11,109.3 \pm 160.67$
CLASS-SINGLE	0.9712 ± 0.0015	$36,204.4 \pm 120.07$
CLASS-MARKOV	0.9714 ± 0.0034	$52,640.0 \pm 247.16$
DM-SGD	0.9609 ± 0.0004	$32,224.8 \pm 202.95$

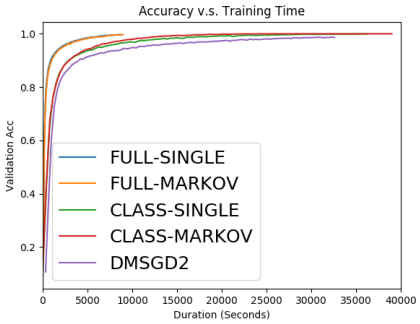


Table 6: Performances on the MNIST dataset. The table reports the final testing accuracy and training duration, the right graph shows the validation accuracy over the training duration.

5. Conclusion

Diversified mini-batch selection at each Deep Neural networks iterations results in a faster algorithm convergence compared with uniform selection. In this paper, we proposed several fast, approximated DPP sampling strategies by taking advantage of available class-label information, which, in turn, allowed us to sample DPP using a Gram-matrix that is constructed from the variable higher layer features, updated at each iteration as parameters change. We detailed the five approaches employed in this paper and the rationale behind how these methods have significantly reduced the time costs. We demonstrated that all proposed algorithms were able to achieve a faster convergence rate and a higher accuracy compared to the previous state-of-the art "fixed" feature approach.

References

Raja Hafiz Affandi, Alex Kulesza, and Emily B. Fox. Markov determinantal point processes. *CoRR*, abs/1210.4850, 2012. URL http://arxiv.org/abs/1210.4850.

G. Alain, A. Lamb, C. Sankar, A. Courville, and Y. Bengio. Variance Reduction in SGD by Distributed Importance Sampling. *ArXiv e-prints*, 2015.

Richard H Byrd, Gillian M Chin, Jorge Nocedal, and Yuchen Wu. Sample size selection in optimization methods for machine learning. *Mathematical programming*, 134(1):127–155, 2012.

Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer vision and Image understanding*, 106(1):59–70, 2007.

Siddharth Gopal. Adaptive sampling for sgd by exploiting side information. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 364–372, New York, New York, USA, 20–22 Jun 2016. PMLR. URL http://proceedings.mlr.press/v48/gopal16.html.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 315–323. Curran Associates, Inc., 2013.

Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Alex Kulesza and Ben Taskar. k-dpps: Fixed-size determinantal point processes. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1193–1200, 2011.

Alex Kulesza, Ben Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J Smola. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 661–670. ACM, 2014.

M-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008a.

M-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008b.

Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. Stochastic variance reduction for nonconvex optimization. *arXiv preprint arXiv:1603.06160*, 2016.

Ohad Shamir. Making gradient descent optimal for strongly convex stochastic optimization. *CoRR*, abs/1109.5647, 2011. URL http://arxiv.org/abs/1109.5647.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

Bo Xie, Yingyu Liang, and Le Song. Diversity leads to generalization in neural networks. *CoRR*, abs/1611.03131, 2016. URL http://arxiv.org/abs/1611.03131.

Dong Yin, Ashwin Pananjady, Maximilian Lam, Dimitris S. Papailiopoulos, Kannan Ramchandran, and Peter Bartlett. Gradient diversity empowers distributed learning. *CoRR*, abs/1706.05699, 2017. URL http://arxiv.org/abs/1706.05699.

Cheng Zhang, Hedvig Kjellström, and Stephan Mandt. Stochastic learning on imbalanced data: Determinantal point processes for mini-batch diversification. *CoRR*, abs/1705.00607, 2017. URL http://arxiv.org/abs/1705.00607.

Peilin Zhao and Tong Zhang. Accelerating minibatch stochastic gradient descent using stratified sampling. *arXiv preprint arXiv:1405.3080*, 2014.

Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1–9, 2015.