Graph-based Deep Learning Analysis and Instance Selection

Keisuke Nonaka  
KDDI Corporation  
Tokyo, Japan  
ki-nonaka@kddi.com

Sarath Shekkizhar and Antonio Ortega  
University of Southern California  
California, USA  
shekkizh@usc.edu, aortega@usc.edu

Abstract—While deep learning is a powerful tool for many applications, there has been only limited research about selection of data for training, i.e., instance selection, which enhances deep learning scalability by saving computational resources. This can be attributed in part to the difficulty of interpreting deep learning models. While some graph-based methods have been proposed to improve performance and interpret behavior of deep learning, the instance selection problem has not been addressed from a graph perspective. In this paper, we analyze the behavior of deep learning outputs by using the K-nearest neighbor (KNN) graph construction. We observe that when a directed KNN graph is constructed, instead of the more conventional undirected KNN, a large number of instances become isolated nodes, i.e., they do not belong to the directed neighborhoods of any other nodes. Based on this, we propose two new instance selection methods, that both lead to fewer isolated nodes, by either directly eliminating them (minimization approach) or by connecting them more strongly to other points (maximization). Our experiments show that our proposed maximization method leads to better performance than random selection and recent methods for instance selection.

Index Terms—Deep learning, K-nearest neighbor, Directed graph, Instance selection

I. INTRODUCTION

Due to its excellent performance results, deep learning continues to be applied in an ever growing number of applications, including image classification, object recognition or automatic translation. An important step in the development of a deep learning system is the selection of the training dataset, or instance selection, which directly affects performance. Selecting the “right” data reduces computational complexity and memory usage, while preserving accuracy, and can make deep learning more scalable. However, instance selection is a major challenge and, often, datasets are used “as is”, with randomized selection methods to break data into training, validation and testing sets. Due to the low interpretability of deep learning methods, a specific data selection can only be evaluated based on output results. In this paper, our goal is to develop a better understanding of dataset structure, and in particular its impact on deep network training, and to use this to propose improved instance selection.

Constructing graphs from a set of non-uniform data points [1] is required to apply graph signal processing (GSP) methods [2] in machine learning applications, such as semi-supervised learning [3]. A deep learning system can be viewed as an operation in multiple stages that maps input data to a high-dimensional space and performs classification. In this paper we apply graph concepts following the ideas in [4]–[6], where it was noted that deep learning systems can be characterized by a data graph, which changes after each layer and each epoch of training, and a graph signal (formed by the labels associated to points in the dataset), which does not change.

The K-nearest neighbor (KNN) method is a simple and widely used technique for graph construction, and has been applied in instance selection [7]–[10]. For example, a method proposed in [8] detects and removes a noisy instance by using majority voting of its neighbors, a method that is said to make boundaries between classes smooth. Another method repeats this procedure until the number of iterations meets a certain criterion [9]. In [10], a method that accelerates a conventional selection method [7] has been proposed, leading to the observation that an instance with noise is more important than an interior instance within a class. However, these methods only considered KNN connections in the original data space without considering connections in the transformed space given by the penultimate layer of the model (the one before the fully connected layer). We will focus on the penultimate layer, as this layer most impacts final classification performance.

Note that active learning (AL) [11]–[13] is concerned with prioritizing data to be labeled. For example, [11] presents data acquisition methods for Deep Bayesian AL scheme based on model uncertainty defined using dropout. Unlike AL where data selection is prioritized for labelling, we focus our efforts in this work on supervised neural network models and how their performance is affected by the choice of data samples used for training. It is important to note that our instance selection procedure, however, does not depend on the labels used for training and can potentially be used in conjunction with AL.

In this paper, we use KNN constructions to analyze the behavior of a popular deep learning framework, convolutional neural network (CNN). In particular, we study the characteristics of a directed KNN graph constructed on the penultimate layer of three simple CNN models. We show the evolution of this graph representation and the underlying data over the course of training in each model. We observe that input data is progressively transformed, similar data examples are clustered,
fully clear which instance contributes to improvement of CNN. We do not use data augmentation during training to identify and weights of the CNNs are initialized according to [15].

The proposed heuristics follow seemingly conflicting ideas: criterion based on the spread (degree) associated to each node. leverage this observation to propose two instance selection while examples that are only loosely tied to these clusters and uncertainty based selection method of [11]. robust model performance in comparison to random selection proposed instance selection methods lead to better and more outliers are no longer selected. Experiments show that our selects more points inside dense clusters, so that minimization clusters, thus reducing cluster density for training, ii) spread.

ple, and thus suitable for analysis, and include commonly used nonlinearity

A. CNN Structure settings

Throughout this paper, we analyze three CNNs that are simple, and thus suitable for analysis, and include commonly used regularization layers (max pooling and batch normalization):

- Basic CNN: This consists of 6 convolutional layers and fully connected layer (Figure 1). Convolutional layers and fully connected layers are activated by rectified linear units (ReLU) and soft max units, respectively. The size of the convolutional kernel is $n$.
- MP CNN: We add $2 \times 2$ max pooling layer after layer 3 and layer 6 of the basic CNN. This is inspired by [4].
- BN CNN: We add batch normalization layer after the max pooling layer of MP CNN described above.

We use ADAM optimizer [14] with a learning rate of $1e^{-3}$ and weights of the CNNs are initialized according to [15]. We do not use data augmentation during training to identify clearly which instance contributes to improvement of CNN.

B. Graph Construction on CNN

Graphs are a powerful framework that can represent complex distributions with non-uniform sampling points. The output of a CNN layer can be considered as points scattered in a high-dimensional space and can be modelled using a graph. In particular we define a directed weighted graph $G = (V, E)$, generated by the set of outputs at the penultimate layer, where the set $V = \{x_1, x_2, \ldots, x_M\}$ contains the output vectors (nodes) associated to each data point in the training set and $E$ is the set of edges connecting nodes. For the case in Figure 1, each $x_i$ is a tensor with dimension $20 \times 20 \times n$. The adjacency matrix $A$ that represents the graph structure is given as follows:

$$A_{i,j} = \begin{cases} w_{i,j} & \text{if } (i, j) \in E \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

where $A_{i,j}$ is the entry at position $(i, j)$ of matrix $A$ and $w_{i,j}$ is edge weight between $x_i$ to $x_j$.

III. ANALYSIS OF CNN BEHAVIOR

To analyze the behavior of a CNN, we train it by using well-known datasets such as CIFAR-10 [16] and MNIST [17]. CIFAR-10 has 50000 32 x 32 images for training and 10000 images for test, whereas the MNIST has 60000 28 x 28 images for training and 10000 images for test.

First, we train all CNNs described in Section II-A for a fixed set of epochs (30) for both datasets. We use 10% of the training set as validation data to examine over-fitting behavior. After training, we pick a subset of the training dataset (1000 randomly selected images) and use it to analyze the network behaviour. For a given input, intermediate outputs after layer provide intermediate representations of that instance. Observing how these intermediate representations change from layer to layer (as well as during different training epochs) helps us understand how the CNN optimizes its representations [4]. In this paper, we only focus on the penultimate layer, which directly affects final classification results, and construct graphs based on different sets of output data (corresponding to different input sets). Note, in particular, that a different graph is obtained at each epoch of training and for each set of inputs (e.g., different choices made by a given instance selection method).

For simplicity, we use the conventional KNN approach for graph construction, although alternative methods may be considered in future work (e.g., [1]). A KNN graph is constructed by calculating pairwise distances between nodes and then selecting the $K$ closest nodes to connect with a given node. In our approach a KNN graph is constructed where each node corresponds to one output of the penultimate layer. Note that if node $i$ is one of the $K$ nearest neighbors of node $j$, the reverse is not guaranteed to happen, so that KNN graphs are directed. While in the literature it is common to use undirected KNN graphs, in this paper we make use of the directed nature of the KNN graph and do not use symmetrization. In particular, using directed KNN graphs allows us to identify isolated nodes (nodes that are not among the $K$ nearest neighbors of any other

Fig. 1. Structure of basic CNN.
nodes). The weights of directed KNN graph are obtained with a simple similarity metric:  

\[ w_{i,j} = \exp(-\|x_i - x_j\|^2), \]  

(2)

where the vectors \(x_i\) and \(x_j\) are obtained by vectorizing the tensors corresponding instances \(i\) and \(j\) (associated to the corresponding graph nodes). Denoting \(N_i\) the set of \(K\) nearest neighbors of \(x_i\), we can say that \(x_i\) is an isolated node if we have that \(\{x_i\} \cap N_j = \emptyset\) for all \(j \neq i\), as shown in Figure 2. That is, \(x_i\) is not selected as a neighbor by any other nodes.

We observe experimentally the number of isolated nodes in the penultimate layer for both CIFAR-10 and MNIST datasets, with \(K = 5\) and \(K = 50\) (Figures 3 to 6). The dimension of the output of each CNN and the number of isolated nodes are in Table I. These results show that for CIFAR-10 and \(n = 32\) the outputs of Basic CNN are widely spread in high dimensional space so that most of the nodes (about 80% of the total) are isolated. In other words, the non-isolated nodes refer to each other, suggesting that they form a relatively compact cluster. In contrast, MP CNN and BN CNN with CIFAR-10 produce fewer isolated nodes, so that the outputs have different characteristics from those observed for Basic CNN.

As for MNIST, there are fewer isolated nodes as compared with CIFAR-10. This is because datapoints in the MNIST dataset have a similar structure, e.g., a black background. Thus, although the dimension of \(x_i\) is high, the number of entries in \(x_i\) that are necessary to distinguish it from other data points is relatively small, i.e., for this data a lower dimensional space may be sufficient for classification.

These observations suggest that, higher the dimension and more complex the data is, the more likely it is that isolated nodes may be produced as a way to achieve better data separation and good classification performance. Thus we can see that the number of isolated nodes is higher for CIFAR-10 than for MNIST, at the initial stages of training, indicating that the latter dataset is simpler and the data naturally clusters. Also note that as training progresses the number of isolated nodes increases, which clearly shows that training increases separation between instances and thus the number of isolated nodes grows. After training ends the number of isolated nodes remains substantially larger for CIFAR-10.

IV. INSTANCE SELECTION

Based on the previous analysis, we consider refining the dataset by selecting instances that can improve CNN performance. Our experiments showed that if the dimension of output is high, the number of isolated nodes significantly increases. We can interpret this from two different perspectives, each leading to a different instance selection strategy.

First, we may view isolated nodes as outliers, that is, datapoints that are not strongly connected to others. If we assume that outliers are not representative, then a possible instance selection approach would seek to avoid including many of these outliers in the training set. This can be accomplished...
by performing an instance selection that favors denser areas by seeking to minimize the spread between nodes. Second, we may instead view the clusters as being too dense, which explains why data points in those areas are not connected to the outliers, which become disconnected as a consequence. In this scenario, the goal should be to reduce the density of cluster areas, so that they do not have too much influence on the classifier. Based on this idea, an alternative technique for instance selection is to maximize the spread (thus reducing the density of points in the cluster areas).

Based on this, we propose two alternative methods, which maximize or minimize the spread of all nodes. To define a notion of spread, consider the adjacency matrix $A$ obtained by KNN, and let $a_i$ be the sum of all similarities between node $x_i$ and nodes in the set $N_i$, its $K$ nearest neighbors. Note that if $a_i$ is large then the neighbor weights are large and thus the area surrounding $i$ is dense, with small spread, while small $a_i$ indicates large distances to neighbors. Thus, an approach to minimize spread would remove nodes with small $a_i$, while a maximization of spread would remove first nodes in compact neighborhoods (large $a_i$).

V. EXPERIMENTS

In this section, we evaluate the performance of the proposed instance selection methods for the convolutional architectures presented in Section II-A. We perform preliminary training of CNNs for 30 epochs to obtain their behavior as in Section III. Then, based on the trained CNN, we test the two proposed instance selection methods. Specifically, we process a batch of 1000 training images, selecting a subset (900 images) based on the node degree of each image at the penultimate layer as described in Section IV. Then, the selected instances are grouped into a new training set, the models are reinitialized and the system is evaluated by retraining the CNNs from scratch using only the selected instances. Moreover, the subset is removed from the initial training dataset. We repeat this selection procedure in batches for the entire training dataset. For example, in CIFAR-10, the selection procedure would require 50 (= 50000/1000) iterations for a complete pass over the training dataset (see Table II). Note that we do not use test dataset for either training or instance selection. Further, the selection procedure, given a trained model, does not require any label information and is kept separate from the neural network weight optimization.

We compare our method to random selection and to an approach that maximizes Variation Ratios with Monte Carlo dropout (Dropout VarRatio), which is shown to perform better than previous instance selection functions [11]. For a fair comparison, we only modify the number of instances that are selected at one iteration in Dropout VarRatio so that we can compare our method to Dropout VarRatio where both methods select the same number of instances. Specifically, we select 900 instances from 2000 training data, which is the default value of the implementation [18] distributed by the authors of [11], for Dropout VarRatio and repeat this selection process until we have the same number of selected dataset instances as our method.

Finally, to evaluate the selection methods, we train a neural network over 20 epochs using the selected instances (45000 images) and observe the model accuracy on test data for both CIFAR-10 and MNIST. For the random selection method, training and testing are executed 10 times and the average score is reported in the results. Note that we do not use the trained weights that were obtained earlier as part of the initial training for instance selection in our method for evaluation. In other words, we are interested in evaluating the performance of a neural network trained only on the selected subset.

Test accuracy during training epochs is shown in Figures 7 and 8. We observe that the proposed maximizing method outperforms other selection procedures for all network settings as seen in Figure 7). This is inline with our intuition about isolated nodes: reducing the density of data clusters and making data more widely spread should help improve performance. However, Figures 7(b) and 7(c) show that the difference between the proposed maximizing and minimizing approaches is minimal for better regularized models with the same number of convolutional filters ($n$). We believe that this is due to max pooling and batch normalization, which explicitly avoids overfitting to the cluster, and thus reduces the benefits of instance selection. This is further supported in Figures 7(d) to 7(f) where we regularize the model by reducing the number of parameters in the model. The fact seems to indicate that when a low dimensional, well regularized CNN is used, there is only a small difference between the two proposed strategies (removing outliers and reducing cluster density).

The Dropout VarRatio method in Figure 7 shows lower per-
Fig. 7. CIFAR-10 test accuracy for different network architectures using various instance selection procedures. (a)-(c): three types of CNN with $n = 32$. (d)-(f): three types of CNN with $n = 8$. The parameter of KNN for our methods is $K = 50$.

Fig. 8. MNIST test accuracy for different network architectures using various instance selection procedures. (a)-(c): three types of CNN with $n = 32$. (d)-(f): three types of CNN with $n = 8$. The parameter of KNN for our methods is $K = 50$.

Formance than our method for CIFAR-10. This is because the method aims to sample data optimally by updating the model gradually with data batches that have already been sampled in the previous iteration. Thus to improve the performance, very small batches need to be used (e.g. 10 samples every iteration). However, as the required amount of data increases, so does the number of training iterations. This leads to low performance and limits the scalability of this method. For instance, if we use a small batch with 10 samples, training will have to be done 4500 times so as to acquire the total number of required data in these experiments, which is not practical.

Moreover, our method has advantages over Dropout VarRatio even if both use the same number of iterations. In particular, our method requires only one inference for obtaining KNN for each batch, while the Dropout VarRatio requires one training for updating the model per each batch. Clearly, training is very data intensive, e.g. several tens of thousands of images are needed, and has significant impact on complexity.

For the MNIST case, as shown in Figure 8, contrary to the case of CIFAR10, the Dropout VarRatio shows better performance in most cases except the case of BN CNN with $n = 32$. If a well regularized CNN (as MP and BN in Figures
8(e) and 8(f)) is used and the dimension of the CNN is low, the Dropout VarRatio is clearly better than others, while our maximizing method also shows its advantage compared with minimization and random choice.

The results of the Dropout VarRatio with MNIST are better, however, the fluctuations in performance which is seen in the CIFAR-10 case are a drawback of the Dropout VarRatio. Overall, the proposed maximizing method is a practical strategy with robust performance over different CNN architectures. This supports our consideration that the number of isolated nodes increase as the training progresses and that reducing dense clusters helps improve model performance.

VI. CONCLUSION

We present a framework to characterize the behavior of deep learning systems using a simple directed KNN graph, constructed on the outputs of neural network layer. Our analysis shows the number of isolated nodes in a neural network is affected by pooling which in turn affects the performance of the model. We propose a heuristic method, based on our study, for instance selection based on the degree spread of nodes connected using a directed KNN. We validate the superior performance of our proposed methods over random selection and recently proposed uncertainty based instance selection procedure. Our method is computationally simple and is robust under different model settings. In particular, even in the presence of large number of isolated node in high dimensions, careful instance selection using the proposed method shows consistent improvement in performance. In the future, we would like to investigate instance selection procedure and isolated nodes for the purpose of active, semi-supervised learning and scale our framework to more complex neural network models.

REFERENCES