Toward Sampling for Deep Learning Model Diagnosis

***Abstract*—Deep learning (DL) models have achieved paradigm-changing performance in many fields with high dimensional data, such as images, audio, and text. However, the black-box nature of deep neural networks is not only a barrier to adoption in applications such as medical diagnosis, where interpretability is essential, but it also impedes diagnosis of under performing models. The task of diagnosing or explaining DL models requires the computation of additional artifacts, such as activation values and gradients. These artifacts are large in volume, and their computation, storage, and querying raise significant data management challenges.**

In this paper, we develop a novel data sampling technique that produces approximate but accurate results for these model debugging queries. Our sampling technique utilizes the lower dimension representation learned by the DL model and focuses on model decision boundaries for the data in this lower dimensional space.

I. Introduction

Deep learning (DL) models have enabled unprecedented breakthroughs in developing artificial intelligence systems for analyzing high-dimensional data, such as text, audio, and images. Building such models is a *data intensive* task. To build an effective model, a machine learning (ML) practitioner needs to proceed in an iterative fashion, building and tuning dozens of models before selecting one. While naive selection of the *best* model could be based on statistical measures such as, accuracy, F1 score, etc., examining what the model is learning and why it is making mistakes requires access to artifacts, such as model activations and gradients. Activation values, or *activations*, are learned representations of input data. *Gradients* are partial derivatives of the target output (e.g., the true label of the input data) with respect to the input data. At a high level, activations and gradients are high dimensional vectors with sizes that depend on input data dimensionality and DL model architecture. While activations depict what the deep learning model *sees*, gradients depict *areas of high model sensitivity*.

The naive solution of pre-computing and storing all artifacts required for model diagnosis scales as the product of size of input data and number of parameters of the deep learning model. For even small data sets and models this can be up to three orders of magnitude larger than the input data per model [18]. This explosion in the volume of data makes it difficult to efficiently perform diagnosis tasks, often preventing interactive diagnosis. Thus, with hundreds of gigabytes of artifacts per model, building, diagnosing, and selecting a DL model becomes a large-scale data management challenge.

Previous attempts at solving this problem either pre-generated all data required to provide interactive query times [10], [12] or utilized a variety of storage optimization techniques to manage the storage footprint [14], [20]. Both approaches require pre-generated artifacts. Several visualization tools pre-generate some of the aggregates and severely limit the type of queries that can be posed, while others simply do the latter. Systems with storage optimizations [20] reduce the storage required for this data by utilizing techniques such as de-duplication and quantization, etc.

Here, we explore the idea of utilizing samples to diagnose DL models. Instead of relying on generating artifacts on the entire dataset we generate artifacts on a carefully selected sample. Sampling is a fast and flexible database technique for approximate query processing, it works well in high dimensions [1], [8] and is a potential candidate for this workload.

A key challenge with answering model debugging queries over input data samples is that those queries frequently ask for outlier values such as the top-k maximally activated neurons for a given layer and output label (see [18] for a more detailed workload characterization). Common sampling techniques that rely on uniform random sampling fail to adequately capture information about such values.

In this paper, we outline an approach to create samples that addresses the above problem and that can serve to debug any DL model where a lower dimensional representation of the input data is learned in a supervised, semi-supervised or unsupervised manner. We compare our technique to standard sampling techniques such as uniform, random sampling and stratified sampling as well as a variety of state-of-the-art alternatives from the literature.

II. Approach

To enable interactive model diagnosis, our approach creates a sample. We compute the results of a query on this sample instead of the entire data. In this section, we describe our approach and present other baseline techniques for selecting these samples.

The key insight that we utilize to select a data sample and avoid generating and storing all activation values is that DL models learn a lower dimension representation of the data, and a classifier. DL training transforms the input data, creating a new representation with each layer. Training criteria encourage training set neighbors, such as data points from the same class, to have similar representations. Leveraging this lower dimensional representation learned by the model has the dual benefit of reducing the dimensionality of the data and focusing on the representation learned by the model. Since the objective of the workload is to diagnose this model, we hypothesize that

leveraging the learned latent space to select a sample will be key to understanding what the model has learned. For model diagnosis we view the training, and test data points in the *latent space*, i.e., instead of viewing the data in the high dimensional original format of images, audio or text, we utilize this lower dimensional representation of the data learned by the model's last hidden layer to create samples.

Our goal is to diagnose the model, which implies that a subset of the queries will focus on what the model got wrong, as we discuss in more detail in the full version of this paper [18]. In a classification problem with multiple classes, the decision boundary partitions the underlying vector space into multiple regions, one for each class. Decision boundaries are where the output label of a classifier is ambiguous, i.e., where errors and mis-classifications occur. The diagnosis of a DL model requires exploration of the *decision boundary* for a model [7], [22].

Thus, our approach is based on utilizing the lower dimensional representation and focusing on decision boundaries in the *latent space* when selecting the data points to include in our sample. Different types of queries serve for DL model diagnosis. One type of debugging query that is especially difficult to answer from a sample are queries that ask for the top-k maximally activated neurons or the distribution of maximally activated neurons. Top-k queries are generally an important area of database research. The best known general-purpose algorithm for identifying top-k items is the Threshold Algorithm [5], which operates on the complete, sorted multi-dimensional data required to compute the top-k elements. Approximate algorithms for top-k retrieval require building probabilistic models to fit the score distribution of the underlying data as proposed in [19]. In contrast to these approaches, we wish to avoid computing and storing activations for the entire data set in the first place.

A. Baselines

Since our key insight to selecting a sample is to leverage the lower dimensional representation of the data learned by the model, the first baseline that we consider is a simple method for sampling from the latent space. The naive way of selecting a sample that covers the n-dimensional latent space is to create a grid in that space and sample from each cell. However, the latent space is typically high dimensional, e.g., for the MNIST dataset, the latent space is $84D$. Even if we divide each dimension into two buckets, we get a total of $2^{84} \sim 1.93x10^{25}$ buckets. Instead, we reduce the dimensionality of data (to $5D$ for evaluation) in the latent space for this analysis before collecting an equal number of instances at random from each underlying cell. We call this naive technique *simple latent space* sampling.

Another way to lower the dimensions is to utilize the classification result. Each data point is classified by the model as belonging to a class. This result is encapsulated in a *confusion matrix* (a.k.a. the error matrix), which tabulates the performance of a classification algorithm. For a binary classifier, the confusion matrix counts the number of true positives, false positives, true negatives, and false negatives. For multiple labels, the confusion matrix generalizes this concept. Each row of the matrix represents a predicted class, while each column represents a true class. In this technique, we sample based on cells in the confusion matrix. We call this technique *stratified by confusion matrix (CM)*.

In database systems such as BlinkDB [1], strata are defined over a subset of columns that typically correspond to categorical valued attributes, e.g. `city`. For DL model diagnosis, the underlying data can be considered a relation, with each row representing a data item (e.g., one image) and each column a value of interest, such as the activation value for a neuron in the model. Each row can be extended with metadata, such as the predicted class and the class label. The *stratified by CM* sample thus serves as a stratified sample baseline.

In addition, we use two other techniques from the literature as baselines. First, we use visualization aware sampling (VAS) for large scale data visualization, such as scatter and map-plots. VAS is based on the interchange algorithm [15], which selects tuples that minimize a visualization-inspired loss function. Visualization-inspired loss is based on three common visualization goals: regression, density estimation and clustering. The interchange algorithm creates a sample that maximixes visual fidelity of the data at arbitrary zoom levels. Second, we use explicable boundary (EB) trees [21] to create a single sample from input data. This method constructs a boundary tree to approximate the complicated deep neural network models with high fidelity. EB trees provide a single sample for a dataset and a model which explains the boundary between each class learned by the DL model.

B. Clustering in Latent Space

An important part of our approach to selecting a sample for DL model diagnosis is to ensure that model decision boundaries are represented in the sample. To determine boundaries in latent space, we cluster data in latent space and fit a model to estimate the parameters for each class in that space. We do this in both supervised and unsupervised manners. When fitting a supervised model, we use the class labels. In the unsupervised case, we use parameterized models so we utilize the number of unique classes present.

In both supervised and unsupervised cases the models fitted to the latent space provide us with the likelihood that and object belongs to a class or cluster. For binary classification to determine whether an object belongs to class A or class B, let $P(A|x_i)$ be the likelihood that a data instance x_i belongs to class A. In this case, the points on the decision boundary of class A and class B are those for which the ratio $\frac{P(A|x_i)}{P(B|x_i)}$ is ≈ 1. A lower value of likelihood ratio would imply that $P(B|x_i) > P(A|x_i)$ in which case x_i would be assigned to cluster or class B. The higher the likelihood that an object belongs to class A, the higher the ratio $\frac{P(A|xi)}{P(B|x_i)}$ will be.

For a multi-class classifier, where a data point x_i may belong to classes $\subset \quad a, b, c, \ldots$, this ratio would be,

Algorithm 1: Clustering and Sampling

Data: input data in latent space, f,k, j

// k num class labels, f is sample size

$Clusters \leftarrow None$

$sample \leftarrow None$

Clusters = ClusterAndSortData(data,k)

foreach $cluster_i$ *in Clusters* **do**

  $s1 \leftarrow data.head(f * j)$

  $s2 \leftarrow data.tail(f * (1 - j))$

  $sample \leftarrow s1 + s2 + sample$

end

return sample

$\frac{P(A|x_i)}{\sum_{z \subset b,c,d,\ldots} P(Z|x_i)}$, or

$$\frac{P(A|x_i)}{P(\neg A|x_i)}$$

Our sampling technique clusters the data in the latent space, then sorts data in each cluster or class by the ratio of likelihood of belonging to that particular class. This sorted list thus consists of exemplars on the higher end and outliers on the lower end of the list. We utilize a tuning parameter j to determine the proportion of exemplars and outliers in our sample. We select $j\%$ from the outliers and $1 - j\%$ from the exemplars. Algorithm 1 describes this approach in further detail.

For the unsupervised technique, we utilize a parameterized clustering technique, the Gaussian Mixture Model (GMM). These models offer a probabilistic way to represent normally distributed sub-populations within an overall population. We set the number of clusters in GMM to be equal to the number of unique classes in the dataset. We utilize variational estimation for the GMM [3], where the effective number of components can be inferred from the data.

For the supervised technique, we use max-margin classifiers to classify the data in the latent space. Margin classifiers are a class of supervised classification algorithms that utilize distance from the decision boundary to bound the classifier's generalization of error. Support vector machine (SVM) [17] is an example of this category of classifiers, which learns boundaries based on labels so that the examples of the separate classes are divided by a clear gap that is as wide as possible. SVMs utilize kernel functions [11]; these help to projecting data to a higher dimensional space where points can be linearly separated. DL models do not have non-linear activation functions after the last hidden layer, so the latent representation from last the hidden layer should enable discovery of linear boundaries. Thus, we utilize a linear kernel for SVM [6], which has the dual advantage of being faster than non-linear kernels and less prone to over-fitting. Results of the classifier are turned into a probability distribution over classes by using Platt scaling [16], [23]. These probabilities are used to sort the data items in each cluster or predicted class and then select a sample.

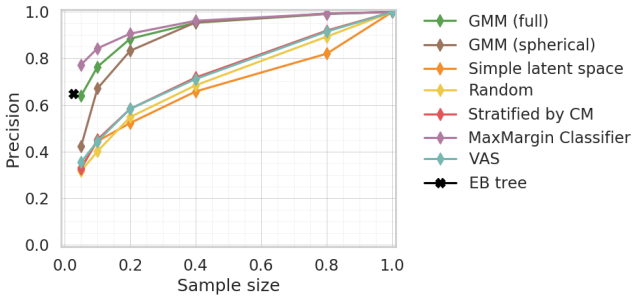

Fig. 1: Precision for top-10 neurons for MNIST. The X-axis shows the sample size as a fraction of the entire data set.

III. Preliminary Evaluation

Here, we empirically evaluate our hypotheses from our sampling approach, namely sampling evenly from the latent space is not sufficient; model decision boundaries are the most important region of this latent space for answering model diagnosis queries and must be well represented in a reliable sample. Model diagnosis queries are posed over sets [10] e.g. top-10 maximally activated neurons for layer x, for all correctly classified items for class a. Instead of considering the immense set of subsets possible we limit our evaluation to all combinations of layers of DL model, number of classes in the input data and classification (correct or incorrect). Thus, to measure accuracy of a query for a sample, we first compute the query results for each of these combinations (layer, class and classification). Next, we compute a metric comparing the results from the sample with the results for the same combination on the entire data set. Finally, we calculate the over-all query set accuracy for each query set by averaging the value of the corresponding metric over the combinations.

For the evaluation we compare five baseline techniques with two sampling techniques described in Section II. We use a top-k query and to measure how well our sample performs we use *precision* as the metric. Precision is the fraction of top-k results from the sample that belong to the true top-k result. Precision lies between $[0, 1]$. A precision value of 0 implies that the sample top-k does not contain any of the full data top-k neurons. We evaluate our sampling technique on MNIST data set that consists of $28x28$ pixel gray-scale images of handwritten numerical digits with a training and test set of 60K and 10K images, respectively. We train a 6 layer model. In our evaluation we measure precision for top-10 maximally activated neurons for every combination of 6 layers and ten classes which are correctly and incorrectly classified. Figure. 1 shows the results. As the figure shows, our sampling approaches outperform all baselines for all sample sizes.

IV. Related Work

Our work is related to model diagnosis systems and approximate query processing. Model diagnosis systems either pre-generated all data required to provide interactive query times [10] or utilized a variety of storage optimization techniques to manage the storage footprint [20]. Both approaches

require pre-generated artifacts. These tools would benefit from our sampling techniques, as sampling would help reduce the scale of data required to support model diagnosis. Activis [10], for instance selectively pre-computes values for nodes of interest to save computation and storage. Sampling techniques such as ours will enable ML practitioners using tools such as Activis to avoid making such compromises. Algorithms for exact top-k queries are defined by the seminal work on the threshold algorithm (TA) [5], which require access to the indexed attribute(s) for a data set. Efficient processing of the top-k queries over samples is a challenging task [9]. Related work in this category includes top-k processing techniques that operate on deterministic data but report approximate answers in favor of performance. Algorithms presented in [19] are an approximate adaptation of TA where the approximate answers to the top-k query is associated with probabilistic guarantees. However, like TA this algorithm requires access to sorted attributes for the underlying data, which we do not want to generate and store for DL models. Another approach to approximate top-k answers is considered in similarity search for multi-media databases [2]. This method uses a proximity measure to determine if a data region should be inspected. This utilizes the underlying data distribution rather than individual column value and in that sense is closer to our approach (i.e., instead of examining the underlying data, we utilize the latent space to create a sample).

V. Conclusion and Future work

Deep learning models have become an indispensable tool for a wide range of tasks, such as image classification, object recognition, speech analysis, machine translation, and more. The task of diagnosis for these purportedly black-box models requires additional artifacts, such as activations. These additional artifacts must be generated, stored, and queried for each DL model being debugged. The addition of these artifacts, which can be up to three orders of magnitude larger than the input data size for each model being diagnosed, turns the process of building, diagnosing, and selecting DL models in to a large-scale data management challenge. In this work, we present a novel sample creation technique that reduces the time and complexity required to accomplish these tasks.

The sampling technique we present in this paper focus on sampling input data points, e.g. rows from the relation of data points and activations. The ML literature supports the notion of reducing the number of neurons for which activations need to be calculated [12], [13] and queried. We would like to explore this avenue in future work. The sampling technique described in this paper works well with supervised learning models, i.e. DL models built with labeled data. In future work, we would like to explore our sampling technique and their efficacy for *unsupervised* DL models, such as generative models, auto-regressive models, etc. [4] A large body of scientific data is unlabeled and requires unsupervised learning techniques, and extending our sampling technique in this direction could be beneficial to the scientific community working on newer data sets.

References

[1] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica. Blinkdb: queries with bounded errors and bounded response times on very large data. In *EuroSys '13*, 2013.

[2] G. Amato, F. Rabitti, P. Savino, and P. Zezula. Region proximity in metric spaces and its use for approximate similarity search. *ACM Transactions on Information Systems (TOIS)*, 21(2):192–227, 2003.

[3] H. Attias. A variational baysian framework for graphical models. In *Advances in neural information processing systems*, pages 209–215, 2000.

[4] https://deepmind.com/blog/unsupervised-learning/.

[5] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. *Journal of computer and system sciences*, 66(4):614–656, 2003.

[6] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 2008.

[7] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5), Aug. 2018.

[8] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. In *SIGMOD '97, Proceedings*, 1997.

[9] I. F. Ilyas, G. Beskales, and M. A. Soliman. A survey of top-k query processing techniques in relational database systems. *ACM Computing Surveys (CSUR)*, 40(4):11, 2008.

[10] M. Kahng et al. Activis: Visual exploration of industry-scale deep neural network models. *IEEE TVCG*, 2018.

[11] https://en.wikipedia.org/wiki/Kernel_method.

[12] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu. Towards better analysis of deep convolutional neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 2017.

[13] R. Maithra et al. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *NIPs*, 2017.

[14] H. Miao, A. Li, L. S. Davis, and A. Deshpande. Modelhub: Deep learning lifecycle management. *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, 2017.

[15] Y. Park, M. J. Cafarella, and B. Mozafari. Visualization-aware sampling for very large databases. In *32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016*, 2016.

[16] J. Platt. Probabilistic outputs for svms and comparisons to regularized likehood methods, advances in large margin classifiers, 1999.

[17] https://en.wikipedia.org/wiki/Support-vector_machine.

[18] https://github.com/pm-anon/Papers/blob/master/anon_tech_report.pdf.

[19] M. Theobald, G. Weikum, and R. Schenkel. Top-k query evaluation with probabilistic guarantees. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 648–659. VLDB Endowment, 2004.

[20] M. Vartak, J. M. F. da Trindade, S. Madden, and M. Zaharia. Mistique: A system to store and query model intermediates for model diagnosis. In *SIGMOD Conference*, 2018.

[21] H. Wu, C. Wang, J. Yin, K. Lu, and L. Zhu. Interpreting shared deep learning models via explicable boundary trees. *ArXiv*, abs/1709.03730, 2017.

[22] H. Wu, C. Wang, J. Yin, K. Lu, and L. Zhu. Sharing deep neural network models with interpretation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, 2018.

[23] T.-F. Wu, C.-J. Lin, and R. C. Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5(Aug):975–1005, 2004.