Correctness-preserving Compression of Datasets and Neural Network Models

Vinu Joseph, Nithin Chalapathi, Aditya Bhaskara, Ganesh Gopalakrishnan, Pavel Panchekha and Mu Zhang
School of Computing, University of Utah, Salt Lake City, UT 84112, USA

***Abstract*—Neural networks deployed on edge devices must be efficient both in terms of their model size and the amount of data movement they cause when classifying inputs. These efficiencies are typically achieved through *model compression*: pruning a fully trained network model by zeroing out the weights. Given the overall challenge of neural network correctness, we argue that focusing on *correctness preservation* may allow the community to make measurable progress. We present a state-of-the-art model compression framework called Condensa around which we have launched correctness preservation studies. After presenting Condensa, we describe our initial efforts at understanding the effect of model compression in semantic terms, going beyond the top n% accuracy that Condensa is currently based on. We also take up the relatively unexplored direction of *data compression* that may help reduce data movement. We report preliminary results of learning from decompressed data to understand the effects of compression artifacts. Learning without decompressing input data also holds promise in terms of boosting efficiency, and we also report preliminary results in this regard. Our experiments centered around a state-of-the-art model compression framework called Condensa and two data compression algorithms, namely JPEG and ZFP, demonstrate the potential for employing model- and dataset compression without adversely affecting correctness.**

***Index Terms*—Model Compression, Data Compression, Machine Learning, Correctness Verification**

I. Introduction

As we face the convergence of HPC and data intensive methods [1], an inescapable reality is that we must begin addressing the overall correctness of machine learning systems. Most of today's methods to judge the correctness of networks relies on their *top n% classification accuracy* that captures how well a network trained on its training data performs in the deployment context.

Unfortunately, these methods do not take into account the *semantic content* of what is being classified, and thus are unable to provide its users a sufficiently useful metric of reliability. For example, if one network makes errors with respect to images *within* one semantic class (e.g., "dogs") while another network starts conflating images *across* classes (e.g., "dogs" versus "weapons"), the former network is naturally considered more reliable. While no single real-world decision is likely to be based on the conclusion of a single sensor or classifier (in multi-sensor situations, cross sensor constraints are likely to be used to resolve ambiguities), it still behooves the community to be making each sensor do a better job with respect to the semantic content of what is being classified.

Supported in part by NSF Awards 1704715, 1817073 and 1918497. Vinu Joseph is supported in part by an NVIDIA Graduate Research Fellowship. Nithin Chalapathi is supported by a fellowship from the Undergraduate Research Opportunities Program, University of Utah. Correspondence to Vinu Joseph: vinu@cs.utah.edu

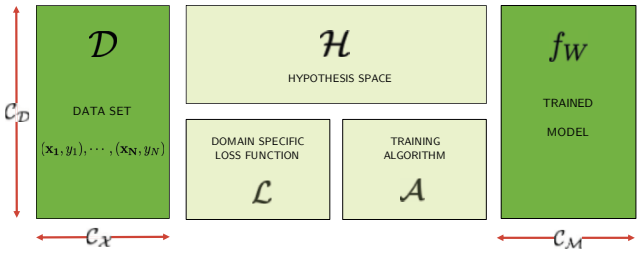

Fig. 1. Illustrates the Learning System used to obtain a reference model f_w that will undergo compression represented by $\mathcal{C}_M$. $\mathcal{C}_D$ refers to one axis of Data compression, where we find a dataset of minimal cardinality to characterize the optimal parameters of a model by using only most influential points (similar to Support Vectors in SVM) and discarding the rest. $\mathcal{C}_X$ refers to data compression by using a compact representation (for example: Using JPEG, ZFP) to store these examples.

Unfortunately, correctness taken as a whole is a daunting challenge: there are literally thousands of network architectures and input classes one must confront. Given this vast and open-ended nature of network correctness, the community is better off making progress with respect to correctness in restricted settings that may arise naturally during the training and deployment of networks. In this work, we examine what it takes for networks to be *deployably efficient* and pursue the direction of *correctness preservation*: ensure that the networks are doing "similar" classification before and after such efficiency measures are introduced. In the technical portion of this position paper, we present two directions of research-in-progress in our group toward attaining deployable levels of efficiency, namely *model compression* and *data compression*. More broadly, our correctness-preserving compression methods promise to provide important insights about when compressed models are qualified to make decisions on real world inputs. Our tools could be used to expose atypical examples for further human inspection [2], choose not to classify certain examples when the compressed model is uncertain [3]–[6], or to aid in explaining the behavior of compressed models. [7]–[10].

A. Model Training

In Figure (1) $\mathbf{x}$ is the input data to the unknown target function $f : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{X}$ is the input space (set of all possible inputs $\mathbf{x}$), and $\mathcal{Y}$ is the output space (set of all possible outputs). There is a dataset $\mathcal{D}$ of input-output

978-0-7381-1044-8/20/$31.00 ©2020 IEEE
DOI 10.1109/Correctness51934.2020.00006

examples $(\mathbf{x_1}, y_1), \cdots, (\mathbf{x_N}, y_N)$, where $y_n = f(\mathbf{x_n})$ for $n = 1, \cdots, N$. The examples are often referred to as data points. There is the learning algorithm $\mathcal{A}$ that uses the data set $\mathcal{D}$ to select a model $\boldsymbol{f_W} : \boldsymbol{\mathcal{X}} \rightarrow \boldsymbol{\mathcal{Y}}$ that approximates the unknown f. The algorithm chooses $\boldsymbol{f_W}$ from a set of candidate models (hypothesis set $\mathcal{H}$) by minimizing a loss function $\mathcal{L}$. The same training setup is used in in the Model Compression pipeline as described in the next section.

B. Model Compression ($\mathcal{C}_M$)

The main approach pursued for attaining deployable efficiency—in both academic research [11]–[18] and in major industrial practices [19], [20]—is *model compression*: taking a trained model that has desired accuracy, safety, security, and privacy properties, and then applying a compression algorithm ($\boldsymbol{\pi}$) to fit the network, $\boldsymbol{f_\theta}$ onto a mobile platform or edge device. Today's state-of-the-art compression methods mainly aim to achieve high levels of network sparsity as well as high overall classification accuracy.

In this work, we briefly examine the process of model compression (§II) and how we can bring in additional aspects of correctness into the picture based on mining *how a network arrives at its decisions*.

C. Data Compression ($\mathcal{C}_D$ and $\mathcal{C}_X$)

The sizes of data sets to be handled by deep networks is witnessing an explosion. Medical image data sets for CT-scan are known to be more than 2 Terabytes in size with a representative workload containing around 2.4 million images [21]. MovieLens 20M is a typical dataset for recommendation systems that includes more than 20,000,000 ratings [22]. Finding a dataset of minimal cardinality to characterize the optimal parameters of a model is of paramount importance in machine learning and distributed optimization over a network [23]. A recent article [24] further elaborates on the growth of data set sizes. The goal of $\mathcal{C}_D$ is to investigate the compressibility of large datasets. More specifically, can we jointly learn the input-output mapping as well as the most representative samples of the dataset (i.e., sufficient dataset)? Given the progress in data compression methods, it is natural to ask the question of whether one can incorporate compression into deep learning. There are two natural directions in which to take this idea: (1) *storing* data in a compressed format ($\mathcal{C}_X$), and (2) *learning directly from compressed data*. There is growing awareness in the community in the effects of compression-induced artifacts and how to deal with misclassifications caused by it [25]. Learning directly from compressed data has recently received attention [26] with the authors studying the advantages of learning directly from JPEG-compressed data. There is however significant room to continue these lines of research to encompass other compression algorithms, and even explore the idea of choosing compression algorithms that minimize misclassification error. We provide our preliminary results on these topics.

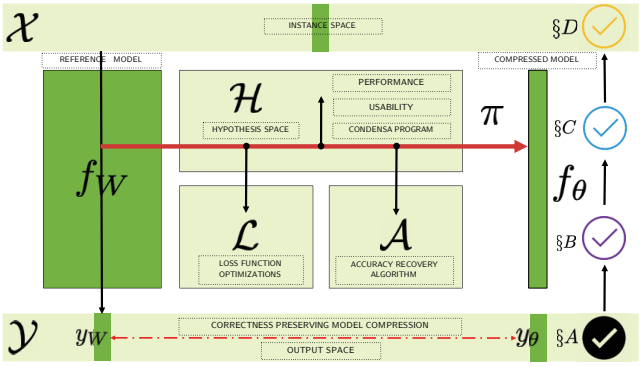

Fig. 2. Condensa Extensions: Delivering the Trifecta of Correctness-preserving Model Compression: Performance, Usability and Correctness. $\boldsymbol{\mathcal{X}}$ represents the input space and the green box on the top bar represents a batch of inputs, whose outputs when passed through the reference model $\boldsymbol{f_W}$ are represented by $\boldsymbol{y_W}$ and the same batch produces $\boldsymbol{y_\theta}$,when passed through $\boldsymbol{f_\theta}$: both are subsets of the output space represented by $\boldsymbol{\mathcal{Y}}$. The correctness-preservation means $\boldsymbol{y_W} == \boldsymbol{y_\theta}$. $\boldsymbol{f_\theta}$ is obtained using a Condensa compression program $\boldsymbol{\pi}$ represented by the red horizontal arrow. Users can specify various compression related strategies like compression scheme to apply, accuracy recovery algorithm $\mathcal{A}$, Loss functions optimizations $\mathcal{L}$ to incorporate correctness-preserving constraints. In section §A- §D below we will describe the various correctness-preserving strategies.

Roadmap

In §II, we discuss how the semantic effects of model compression can be systematically studied based on a state-of-the-art programmable compression framework. In §III, we discuss our initial studies of a data compression framework called ZFP [27], [28] that is widely being studied in the arena of HPC for reducing storage and data movement overheads. We also have preliminary results comparing ZFP against JPEG [29]. In §IV, we provide directions for future work stemming from these discussions.

II. Model Compression

Network compression is the process of converting a neural network f for a certain task to a network $\tilde{f}$ that achieves similar performance as f, but is considerably "simpler" with respect to the memory footprint of the network or time (the number of operations needed to classify an input using the model). Many of the state-of-the art networks turn out to be compressible to reasonably high extents [11], [18], [30] in both these dimensions. This is because redundancy is often built into networks, partly because it helps in the training process to find better minima [31]. Space reduction can be achieved in many ways including sparsifying individual layers by removing certain weights, by reducing the number of bits used in the floating point representation of the weights, by removing "unimportant" features or convolutional filters, by using matrix/tensor approximations to obtain a compact representation of the connection matrix at every layer, and so on. We now provide a brief overview of Condensa [32] (Figure 3) that our experiments are based on. The user provides the pretrained model f, a compression scheme, and an objective function η. Condensa uses a novel acquisition function called *Domain*

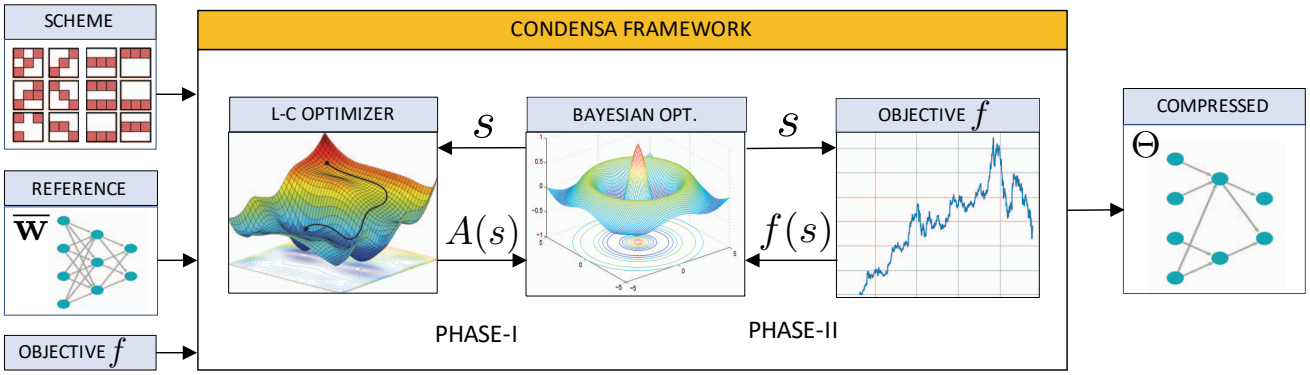

Fig. 3. The Condensa Framework for Model Compression (figure courtesy [30]). The user provides the pre-trained model f_W, a compression scheme, and an objective function f. uses the Bayesian and an L-C optimizer to infer an optimal target sparsity s^* and corresponding compressed model f_θ. This entire process is denoted as the $\mathcal{C}_M$ in Fig-(1)

Restricted Upper Confidence Bound (DR-UCB) in its Bayesian optimization [33] formulation and LC optimization [34] to infer an optimal target sparsity s^* and corresponding compressed model $\tilde{f}$. The compression schemes are chosen from a repertoire supported in the Condensa library. It is specified in Python, and includes schemes such as pruning and quantization. The objective function η (throughput, FLOPs, memory footprint, etc.) is also specified in Python using operators in the Condensa library. Overall, Condensa follows the two-phase approach of (1) using Bayesian Optimization with acquisition function A to find an initial sparsity value s_{acc} that meets the target accuracy within a user-specified threshold; (2) it then constrains the sparsity search space to $(0, s_{acc})$ while optimizing η to maximize performance.

Condensa is a state-of-the-art framework that understands the microarchitecture of many modern GPUs and allows users to *programmatically* build, train, and test neural networks, while also lending significant assistance with hyperparameter tuning. On realistic networks Condensa has delivered memory footprint improvements of $188\times$ and throughput improvements of $2.59\times$ using at most 10 Bayesian Optimization samples per search [32]; these indicate the rapidity with which compressed models can be arrived at. As to our choice of LC, recent work [35], [36] has shown that for high-sparsity regimes, LC performs really well, especially when the optimization becomes difficult with stringent resource constraints imposed.

Due to the resource constraints of deploying models to mobile phones or embedded devices [37], [38] model compression is now is commonly used. An understanding of the *Correctness Ramifications of Compression* becomes particularly important in sensitive domains like health care diagnostics [39], [40], self driving cars [41] and hiring [42], [43], because the introduction of compression may contradict safety, security and privacy objectives.In the sections to follow, we discuss the various correctness criteria that go beyond simple classification accuracy as the acceptance metric. When comparing two neural network models with different depths, number of edges,

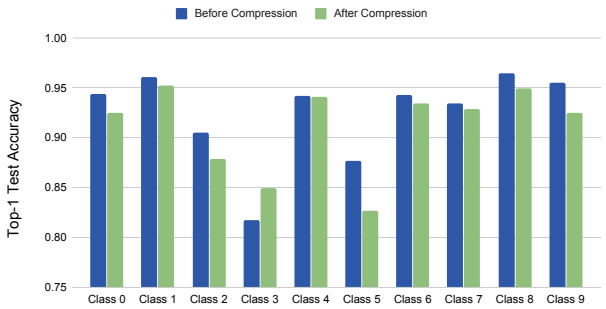

Fig. 4. Output or Functional level Analysis : Effect of compression on class-wise accuracy (CIFAR-10 with ResNet56 architecture, this artifact depends on the kind of compression scheme used and dataset distribution itself.)

number of filters, etc., the naïve view of a network as a set of parameters does not provide much information; neither do simple parameters such as the training loss and test accuracy. We need to look closely at what features of the input are captured and how they impact classification. The measures we provide below show a natural progression, going from a focus on outputs, to intermediate layer activations, to input space partitions, and aim to compare different semantic aspects.

A. *Output or functional level.*

The most black-box way to understand a network is to study the classification it provides for different input classes. This does not tell us how the network arrives at the classification, but it contains more fine-grained information than the aggregate accuracy. Figure 4 illustrates how a state-of-the-art compression tool [30] affects the error in certain classes more significantly than others — something we wish to avoid as it may result in unfair treatment [25].

While class-wise accuracy numbers are more nuanced than the overall accuracy, they are still a coarse representation of the underlying issues. For example, we often observe "structure" in the misclassifications: some classes are much more likely

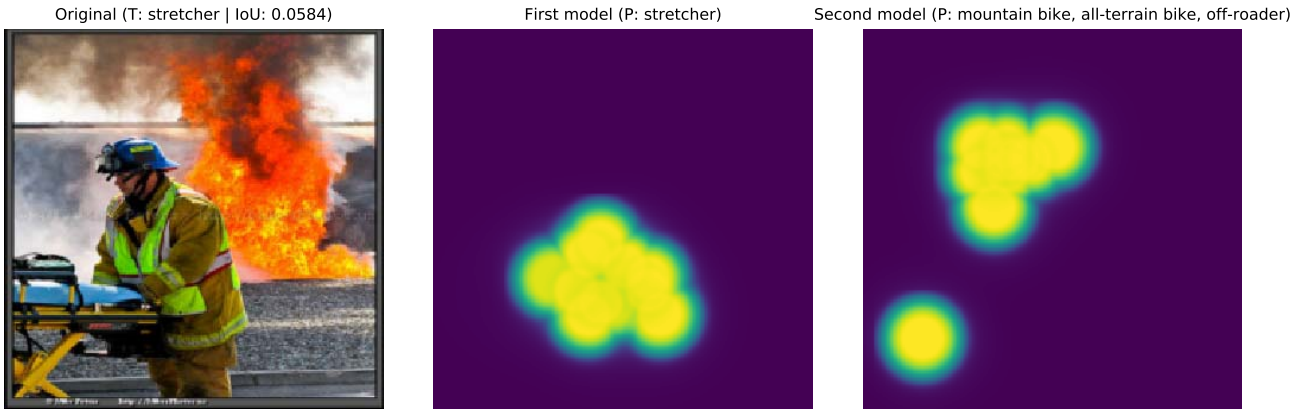

Fig. 5. Cross layer comparisons: Image from ImageNet's *stretcher* class, along with saliency maps showing the parts of the image "most responsible" for classification by the reference and compressed models. Former obtains the correct class, while the latter classifies as *mountain bike, all-terrain bike*. `T:label` represents Ground Truth, `IoU:value` represents the Jaccard similarity indices (Intersection Over Union) between the two attribution maps. `P:label` represents the Predicted labels: The middle attribution map is obtained from the reference model and the attribution map on the right is obtained from the compressed model. Under *Correctness-preserving model compression*, both should be identical, resulting in an `IoU=1.0` here `IoU≅0` indicting a mismatch. This attributions matching problem is critical in domains like Healthcare and self-driving applications, which we tackle in our framework using optimizations in the $\mathcal{L}$ block.

to be misclassified as a few other classes than the rest of the classes. That is, compression may result in one mammal being misclassified as another mammal, but very rarely as an object. This distinction can be important for safety and security, and depending on the application (e.g., autonomous driving, home cameras), some misclassifications are more costly than others. We capture this distinction using an application-specific metric or a distance function $d(y_1, y_2)$ between classes y_1 and y_2 (see [44]).

Denoting the reference and the compressed networks by f and $\widetilde{f}$ respectively (these are functions mapping the inputs $\mathbf{x}$ to labels y), the total misclassification cost of $\widetilde{f}$ relative to f is

$$\Delta_{\mu,d}(\widetilde{f}, f) := \sum_{\mathbf{x}} \mu(\mathbf{x}) d(\widetilde{f}(\mathbf{x}), f(\mathbf{x})),$$

where $\mu(\mathbf{x})$ is the weight of the input $\mathbf{x}$. The main questions we study are: *(1)* how do known compression algorithms perform with respect to the above metric for different choices of d and the reference network? *(2)* how can we incorporate this metric into the compression procedure?

Answering the first question will lead to an understanding of which compression scheme is better for a given application. Also, it may suggest that some reference networks are more compressible than others while maintaining semantic information. Such trends have been observed in our current framework Condensa [30]. Answers to the second question will lead us to better domain-specific compression.

B. Extension to feature space embeddings.

The layer below the final classifier is often used for obtaining embeddings of inputs in the *feature space* or a latent space. This embedding is useful because it captures a network's "knowledge" about an image better than the raw classification. The popular notion of word-embeddings [45], [46] in natural language processing, and the neural image captioner [47], [48] used for generating captions for images are both based on last-layer embeddings. In simple examples like MNIST, visualizations of the feature embeddings reveal a strong structure [49] in the embeddings for different digits. We ask if compression can be made to *preserve* such structure among embeddings. Studies in [50] indicate that this can be highly effective in making the reference and compressed models agree in predictions.

C. Cross-layer comparisons.

Both of the metrics discussed so far involved the final two layers of the network – ones most directly tied to classification. In many safety-critical applications, one may have constraints such as "*the two networks focus on the same aspects of an image*," or "*certain aspects of an image are not used for classification*." To incorporate such constraints, we propose the using methods from interpretability research such as saliency maps and influence attribution [51]–[54]. These techniques give a way to "trace" the factors in an input that contributed to the overall classification. Given a network and an input image $\mathbf{x}$, a method such as [53], [54] outputs the influence (or attribution score) of each pixel on the overall classification.

Interestingly, we find that in many cases of discrepancy between the reference and the compressed model (e.g., Figure 5), the attributions differ significantly, indicating that the two models focus on different regions of the input. Given the attribution scores α and α', we consider the weighted Jaccard

similarity (a generalization of the natural *intersection-over-union* (IoU)) metric

$$\text{sim}(\alpha, \alpha') = \frac{\sum_i \min(\alpha_i, \alpha'_i)}{\sum_i \max(\alpha_i, \alpha'_i)}.$$

When deploying a compressed network, a natural goal is to have high value for this similarity. This can help avoid possible security issues during deployment and in general lead to a higher agreement between the models.

D. Neighborhoods in the Input space.

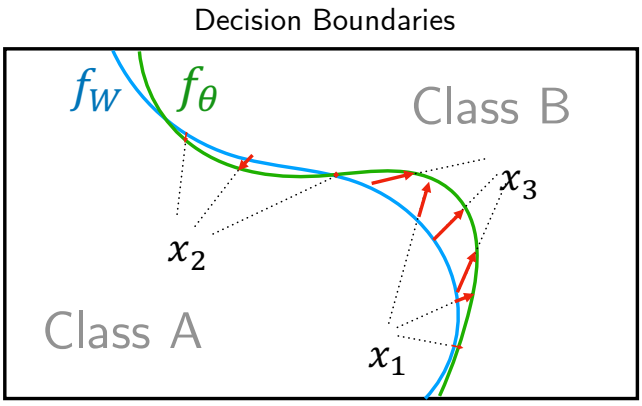

Fig. 6. Input Space Analysis: Here f_W and f_θ represent the decision boundaries of the models before and after compression optimizations $\mathcal{C}_M$: Computing these are intractable in general, but in our formulations we restrict distance computations to linear directions. The change in distance, along a random direction, from training data to the decision boundary captures changes in robustness.

Robustness against perturbations is a critical parameter for deployability. Robustness can be understood by studying how close different inputs are to the class boundaries. Prior works such as [25], [55], [56] study the average distance to the class boundary along random and worst-case directions, and use this information for improving models. A key feature of our framework with its correctness-extensions will be to incorporate robustness specifications into the compression pipeline. Our goals here are: *(1)* to study the effect of compression on the distances to class boundaries for different forms of compression, and *(2)* to incorporate robustness specifications for adversarial, benign and random perturbations into compression.

The first goal will help build a theory of when compression preserves robustness. For example, preliminary experiments on CIFAR-10 with filter pruning show that the distance to decision boundaries along random directions are approximately preserved for many, but not all, images (see Figure 6).

III. Data Compression

While model compression reduces compute and memory requirements, compression of the input dataset can also yield efficiency gains. It is well known that early convolutional layers learn Gabor filters while latter layers learn more abstract features [57], [58]. Moreover, as neural networks gain traction in domains with large datasets (i.e. CT-scans [21]), dataset size becomes a bottleneck in terms of decompression (Decode block in NVIDIA-DALI [59] for example), storage, and I/O operations. As such, dataset compression becomes a promising area for efficiency improvements, both during training and inference. [26] studies the JPEG codec and draws an analogy between the discrete cosine transformation (DCT) and the initial layers of a convolutional neural network. They introduce several architectural changes to ResNet-50 to take advantage of the DCT. By leveraging the DCT coefficients, they reduce the size of ResNet-50 and increase its throughput (frames per second). However, this observation is not specific to the JPEG codec. In particular, ZFP [27] has become a common lossy compression technique used in HPC. ZFP performs a similar near-orthogonal transformation to the DCT. ZFP is a promising dataset compression candidate due to recent work focused on providing error analysis [28], [60] and stability results of iterative methods using ZFP compression [61]. ZFP is a lossy compression algorithm that was designed specifically for floating-point arrays [28]. Other competitive lossy compression algorithms typically require global information. For example SZ [62] compresses each data value using a predictive model by using the adjacent data points in multidimensional space, requiring an ordering of values that inherently limits the data streaming capabilities that could be utilized for machine learning. However, ZFP, first introduced in [27], and modified in [28], only uses local information by first partitioning the d-dimensional scalar array into blocks of 4^d scalars, which are compressed and decompressed independently. ZFP uses a custom near-orthogonal transform on the 4^d-size blocks to remove redundancies in the data through a change in basis. By using negabinary (base -2) to represent the basis coefficients and reordering the coefficients, the leading zeros of small valued coefficients are grouped together when ordered by bit-plane instead of by coefficient. Each bit-plane is then losslessly compressed using an embedded coding scheme [27] which emits one bit at a time until some stopping criterion is satisfied. The stopping criterion for ZFP has three modes: fixed-precision, fixed-rate, and fixed-accuracy. The fixed-rate mode compresses a block to a fixed number of bits, the fixed-precision mode compresses to a variable number of bits while retaining a fixed number of bit planes, and the fixed-accuracy mode compresses a block with relation to the tolerated maximum error. Further work will be needed to investigate how to extend existing error analysis techinques of ZFP to neural networks. Beyond the theoretical guarantees afforded by ZFP, there are a number of systems benefits including an publicly available open-source CUDA implementation, the ability to determine compression level at decode time, random array access, and various compression modes for customization.

A. System Benefits of Dataset compression

First, for various network architectures, we find a compression level such that a compressed CIFAR-10 dataset is

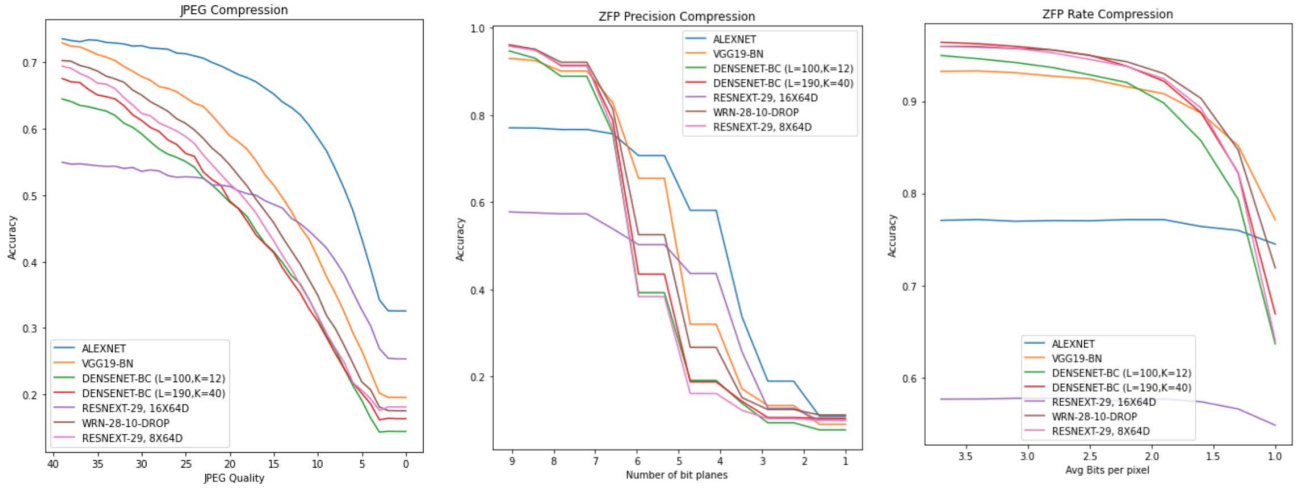

Fig. 7. Illustrates Correctness-preservation using **Output level Analysis**: (Left) Top-1 Test Accuracy vs. JPEG quality parameter on CIFAR 10. Top-1 Test Accuracy vs. ZFP Precision (Middle) and Rate Compression (Right) on CIFAR 10. Details in Section C below.

TABLE I
SYSTEM BENEFITS OF DATASET COMPRESSION: CIFAR 10 COMPRESSED SIZE IN PROPORTION TO THE UNCOMPRESSED DATASET AT VARIOUS ERRORS, R: ZFP-RATE COMPRESSION SCHEME AND P:ZFP-PRECISION SCHEME, HERE ϵ REPRESENTS THE % TOP-1 ACCURACY THE USER IS WILLING TO LET GO AFTER $\mathcal{C}_{\boldsymbol{X}}$ OPTIMIZATION.

COMPR	DNN ARCHITECTURE	$\epsilon = 2\%$	$\epsilon = 5\%$
JPEG	ALEXNET	2.3%	2.1%
ZFP - R	ALEXNET	1.9%	**1.5%**
ZFP - P	ALEXNET	**1.6%**	1.6%
JPEG	VGG19-BN	3.4%	2.8%
ZFP - R	VGG19-BN	**3.2%**	**2.3%**
ZFP - P	VGG19-BN	3.5%	2.5%
JPEG	DENSENET-BC (L=100, K=12)	**3.9%**	**3.2%**
ZFP - R	DENSENET-BC (L=100, K=12)	4.1%	**3.2%**
ZFP - P	DENSENET-BC (L=100, K=12)	4.6%	3.5%
JPEG	DENSENET-BC (L=190, K=40)	3.9%	3.1%
ZFP - R	DENSENET-BC (L=190, K=40)	3.6%	**2.8%**
ZFP - P	DENSENET-BC (L=190, K=40)	**3.5%**	3.5%
JPEG	RESNEXT-29, 16X64D	2.3%	2.1%
ZFP - R	RESNEXT-29, 16X64D	**1.9%**	**1.5%**
ZFP - P	RESNEXT-29, 16X64D	2.4%	1.6%
JPEG	WRN-28-10-DROP	3.5%	3.0%
ZFP - R	WRN-28-10-DROP	**3.2%**	2.8%
ZFP - P	WRN-28-10-DROP	3.5%	**2.5%**
JPEG	RESNEXT-29, 8X64D	**3.5%**	3.0%
ZFP - R	RESNEXT-29, 8X64D	3.6%	2.8%
ZFP - P	RESNEXT-29, 8X64D	**3.5%**	**2.5%**

within 2% and 5% of the original validation accuracy. Second, we compare storage requirements of the compressed images as a percent of the uncompressed dataset (see Table I). On balance, *ZFP is able to achieve a higher rate of compression at equivalent error tolerances*. Finally we also expect that when we learn directly from compressed data the Decode compute which is a mixed compute workload CPU-GPU can be eliminated – today this step is part of the NVIDIA DALI pipeline that decompresses the JPEG data before feeding it to the training or inference kernels. **The end user only cares about end-to-end speed up during training and inference of the neural network, whether it came from data compression or neural network compresssion is irrelevant.** It is therefore important that we carefully utilize profiling tools like nvprof and Rapids Extensions [63] and identify the bottlenecks and fix them.

B. Learning Directly from Compressed Data

While these results indicate the potential of ZFP, we can gain additional compute and memory savings by learning directly from the ZFP coefficients, similar to [26] where they learn directly from DCT coefficients. With no hyperparameter tuning or architectural changes, we are able to recover 90+% accuracy on CIFAR-10. We believe that, with sufficient searching over the hyperparameter space and investigation using the aforementioned debugging stack, the loss in accuracy can be recovered and that the compression parameter can be increased. We continue to explore the correctness ramifications of using ZFP compressed datasets and learning directly from ZFP coefficients and the system benefits it will offer. In the next sections we present our initial results on Correctness-preserving techniques on such compact networks.

C. Correctness-preservation using **Output level Analysis**

In Figure 7, we plot the accuracy versus compression parameter curves for both JPEG and ZFP rate mode. In this setup, we load the image from disk, compress and decompress the image, then pass it through various neural networks. This provides a higher level picture of the accuracy drop off when introducing compression artifacts. Of particular interest here is AlexNet and ResNext-29 8x64D in the JPEG plot. These

TABLE II
LEARNING DIRECTLY FROM COMPRESSED DATA: MODEL VALIDATION ACCURACY TRAINED ON ZFP COEFFICIENTS

ZFP PRECISION	VGG19-BN	RESNET-110	PRERESNET-110	WRN-28-10-DROP	DENSENET-BC (L=100, K=12)
2	40.75%	40.67%	41.28%	40.98%	40.83%
3	64.84%	63.76%	65.14%	66.33%	64.84%
4	76.91%	76.91%	77.42%	79.58%	77.84%
5	83.51%	84.55%	84.48%	87.61%	86.05%
6	87.79%	87.62%	88.03%	91.55%	90.04%
7	89.23%	88.53%	89.67%	92.77%	91.2%
8	89.87%	89.03%	90.4%	93.54%	92.35%
9	89.99%	89.67%	90.4%	93.68%	92.59%
10	89.79%	90.46%	90.88%	93.79%	92.4%
11	90.43%	90.2%	90.42%	93.76%	93.18%
12	90.15%	90.4%	90.66%	94.09%	92.49%
RAW IMAGES	93.43%	93.41%	94.02%	96.12%	95.24%

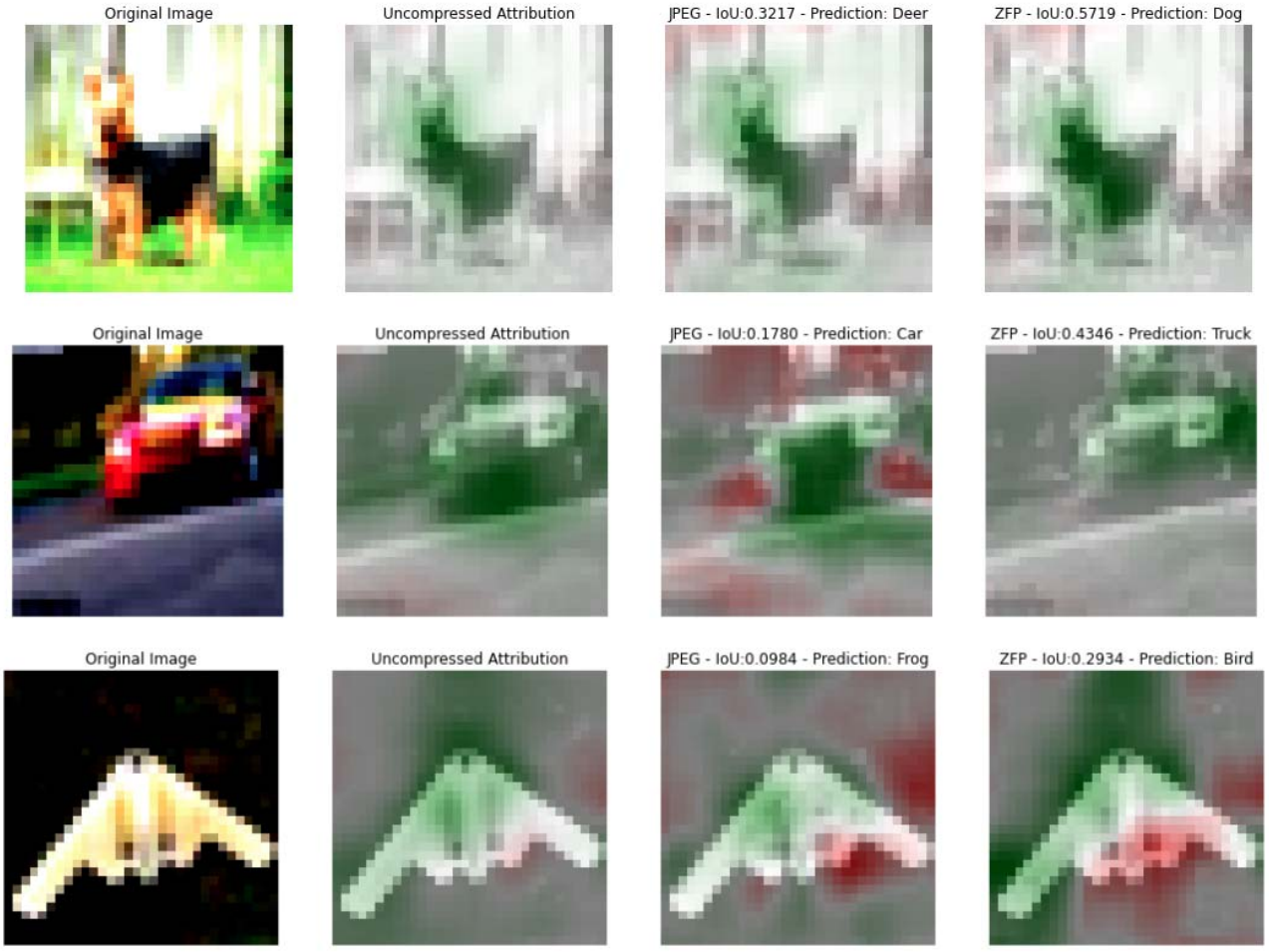

Fig. 8. Illustrates Correctness-preservation using **Cross layer comparisons**: Attributions of misclassified CIFAR-10 images when compressed with ZFP and JPEG when using ResNeXt-29, 8x64d. Original labels are, from top to bottom, dog (JPEG misclassification), car (ZFP misclassification), plane (ZFP and JPEG misclassification). Details in Section D below.

are the only two networks without batch normalization. As a result, they are more resilient to the noise introduced by JPEG.

Batch normalization (BN) layers computes the mean and the standard deviation of the activations and projects them to activations with zero mean and unit norm [64]. These are then projected back using the learned parameters. The interesting part is the normalization phase. Since the network always saw images which were without any JPEG or ZFP compression noise, the learned normalization values are based on those clean images. Once the level of noise increases, this renders the computed mean and standard deviation to be meaningless, resulting in a very significant drop in performance. Therefore, whenever BN layers are employed, they will be very sensitive to changes in the input values. Even minor perturbations can adversely effect the network. We view BN layers to be just an amplification of the minor perturbations. On the other hand, when there is no normalization, there is no explicit noise amplification resulting in much higher resilience against noise and perturbations.

D. Correctness-preservation using **Cross-layer comparisons**

The levels of debugging and analysis in §II can be applied at the dataset level as well. We conducted a preliminary study to review the compression artifacts introduced by ZFP and compare it with JPEG. In Figure 8 we demonstrate *cross-layer comparisons* to get a better view of the type of artificating ZFP creates in comparison to JPEG. We overlay attribution maps generated using occlusion [65] and an IoU score is computed with respect to the uncompressed attribution.

IV. Conclusions, Future Work

In this paper we presented correctness-preserving compression methods for compression of large datasets and over-parameterized neural network models. The correctness abstractions that we have developed were naturally suitable for both model compression (C_M) and data compression optimizations ($\mathcal{C}_D, \mathcal{C}_X$). A user of our framework can use these layers of abstractions to debug failure modes of both these optimizations alike. We illustrated the use of two such debugging techniques on CIFAR-10. On the model compression side, we are developing optimization algorithms and additional loss terms ($\mathcal{L}$ block) by which we automatically determine the optimal weights when we incorporate correctness-preservation as constraints. On the data compression side, we will continue to explore the benefits and trade-offs when using ZFP to compress floating point datasets and learning directly from the compressed space. A challenge that arises in this specific domain is how to comprehensively compare different compression spaces (i.e. DCT Coefficients vs. ZFP Coefficients). Although disk utilization is one simple metric, more nuanced measures are clearly needed to analyze the benefits of various compression schemes.

Acknowledgment

The authors thank Saurav Muralidharan, Michael Garland, Animesh Garg, Shoaib Ahmed Siddiqui, Alyson Fox and Peter Lindstrom for discussions. This work is based on ideas published in [32] and other items of work in progress under various collaborations.

References

[1] G. Fox, A. Adiga, J. Chen, O. Beckstein, S. Jha, J. A. Glazier, J. Kadupitiya, V. Jadhao, M. Kim, J. Qiu, and et al., "Learning everywhere: Pervasive machine learning for effective high-performance computation," *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, May 2019. [Online]. Available: http://dx.doi.org/10.1109/IPDPSW.2019.00081

[2] C. Leibig, V. Allken, M. S. Ayhan, P. Berens, and S. Wahl, "Leveraging uncertainty information from deep neural networks for disease detection," *Scientific reports*, vol. 7, no. 1, pp. 1–14, 2017.

[3] P. L. Bartlett and M. H. Wegkamp, "Classification with a reject option using a hinge loss," *Journal of Machine Learning Research*, vol. 9, no. Aug, pp. 1823–1840, 2008.

[4] C. Cortes, G. DeSalvo, and M. Mohri, "Boosting with abstention," in *Advances in Neural Information Processing Systems*, 2016, pp. 1660–1668.

[5] ——, "Learning with rejection," in *International Conference on Algorithmic Learning Theory*. Springer, 2016, pp. 67–82.

[6] C. Cortes, G. DeSalvo, C. Gentile, M. Mohri, and S. Yang, "Online learning with abstention," *arXiv preprint arXiv:1703.03478*, 2017.

[7] B. Kim, R. Khanna, and O. O. Koyejo, "Examples are not enough, learn to criticize! criticism for interpretability," in *Advances in neural information processing systems*, 2016, pp. 2280–2288.

[8] K. S. Gurumoorthy, A. Dhurandhar, G. Cecchi, and C. Aggarwal, "Efficient data representation by selecting prototypes with importance weights," in *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2019, pp. 260–269.

[9] R. Caruana, "Case-based explanation for artificial neural nets," in *Artificial Neural Networks in Medicine and Biology*. Springer, 2000, pp. 303–308.

[10] S. Hooker, D. Erhan, P.-J. Kindermans, and B. Kim, "A benchmark for interpretability methods in deep neural networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 9734–9745.

[11] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," in *Proceedings of ICLR 2016*, ser. ICLR'16, 2016.

[12] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[13] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡1mb model size," *ArXiv*, vol. abs/1602.07360, 2017.

[14] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 1135–1143. [Online]. Available: http://papers.nips.cc/paper/5784-learning-both-weights-and-connections-for-efficient-neural-network.pdf

[15] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *Computer Vision – ECCV 2016*, 2016.

[16] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: Efficient inference engine on compressed deep neural network," in *Proceedings of the 43rd International Symposium on Computer Architecture*, ser. ISCA '16, 2016.

[17] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[18] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. [Online]. Available: https://openreview.net/forum?id=rJl-b3RcF7

[19] https://www.tensorflow.org/lite, 2020.

[20] https://pytorch.org/tutorials/intermediate/pruning_tutorial.html, 2020.

[21] Oracle, "Performance evaluation of storage and retrieval of dicom image content in oracle database 11g using hp blade servers and intel processors," *White Paper*, 2010.

[22] GroupLens, "https://grouplens.org/datasets/movielens/20m/," *Documentation*.

[23] H. Shokri-Ghadikolaei, H. Ghauch, C. Fischione, and M. Skoglund, "Learning and data selection in big datasets," in *36th International Conference on MachineLearning, Long Beach, California, PMLR 97, 2019.*, 2019.

[24] I. Tobore, J. Li, L. Yuhang, Y. Al-Handarish, A. Kandwal, Z. Nie, and L. Wang, "Deep learning intervention for health care challenges: Some biomedical domain considerations," Aug. 2019. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6696854/

[25] S. Hooker, A. Courville, Y. Dauphin, and A. Frome, "Selective brain damage: Measuring the disparate impact of model pruning," *arXiv preprint arXiv:1911.05248*, 2019.

[26] L. Gueguen, A. Sergeev, B. Kadlec, R. Liu, and J. Yosinski, "Faster neural networks straight from jpeg," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 3933–3944. [Online]. Available: http://papers.nips.cc/paper/7649-faster-neural-networks-straight-from-jpeg.pdf

[27] P. Lindstrom, "Fixed-rate compressed floating-point arrays," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2674–2683, 2014.

[28] J. Diffenderfer, A. L. Fox, J. A. Hittinger, G. Sanders, and P. G. Lindstrom, "Error analysis of zfp compression for floating-point data," *SIAM Journal on Scientific Computing*, vol. 41, no. 3, pp. A1867–A1898, 2019. [Online]. Available: https://doi.org/10.1137/18M1168832

[29] [Online]. Available: https://jpeg.org

[30] V. Joseph, S. Muralidharan, A. Garg, M. Garland, and G. Gopalakrishnan, "A programmable approach to model compression," 2019.

[31] B. Neyshabur, Z. Li, S. Bhojanapalli, Y. LeCun, and N. Srebro, "The role of over-parametrization in generalization of neural networks," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=BygfghAcYX

[32] V. Joseph, G. L. Gopalakrishnan, S. Muralidharan, M. Garland, and A. Garg, "A programmable approach to neural network compression," *IEEE Micro*, vol. 40, no. 5, pp. 17–25, 2020.

[33] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17, 2017.

[34] M. A. Carreira-Perpinán, "Model compression as constrained optimization, with application to neural nets. part I: General framework," *arXiv preprint arXiv:1707.01209*, 2017.

[35] A. Renda, J. Frankle, and M. Carbin, "Comparing rewinding and fine-tuning in neural network pruning," *arXiv preprint arXiv:2003.02389*, 2020.

[36] D. Blalock, J. J. G. Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?" *arXiv preprint arXiv:2003.03033*, 2020.

[37] A. Estava, B. Kuprel, R. Novoa *et al.*, "Dermatologist level classification of skin cancer with deep neural networks [j]," *Nature*, vol. 542, p. 115, 2017.

[38] A. See, M.-T. Luong, and C. D. Manning, "Compression of neural machine translation models via pruning," *arXiv preprint arXiv:1606.09274*, 2016.

[39] R. Gruetzemacher, A. Gupta, and D. Paradice, "3d deep learning for detecting pulmonary nodules in ct scans," *Journal of the American Medical Informatics Association*, vol. 25, no. 10, pp. 1301–1310, 2018.

[40] H. Xie, D. Yang, N. Sun, Z. Chen, and Y. Zhang, "Automated pulmonary nodule detection in ct images using deep convolutional neural networks," *Pattern Recognition*, vol. 85, pp. 109–119, 2019.

[41] NHTSA, *Technical report, U.S. Department of Transportation, National Highway Traffic, Tesla Crash Preliminary Evaluation Report Safety Administration*, 2017. [Online]. Available: https://static.nhtsa.gov/odi/inv/2016/INCLA-PE16007-7876.PDF

[42] D. Harwell, *A face-scanning algorithm increasingly decides whether you deserve the job*, 2019 (accessed March 23, 2019). [Online]. Available: https://wapo.st/2X3bupO

[43] J. Dastin, *Amazon scraps secret ai recruiting tool that showed bias against women*, 2018 (accessed March 23, 2019). [Online]. Available: https://reut.rs/2p0ZWqe

[44] C. Qin, K. D. Dvijotham, B. O'Donoghue, R. Bunel, R. Stanforth, S. Gowal, J. Uesato, G. Swirszcz, and P. Kohli, "Verification of non-linear specifications for neural networks," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=HyeFAsRctQ

[45] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[46] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[47] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3156–3164, 2015.

[48] K. Xu, J. L. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, p. 2048–2057.

[49] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 212–220.

[50] B. Chen, W. Liu, A. Garg, Z. Yu, A. Shrivastava, J. Kautz, and A. Anandkumar, "Angular visual hardness," 2019.

[51] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2014. [Online]. Available: http://arxiv.org/abs/1312.6034

[52] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 818–833.

[53] B. Kim, W. M., J. Gilmer, C. C., W. J., , F. Viegas, and R. Sayres, "Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV) ," *ICML*, 2018.

[54] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. JMLR.org, 2017, p. 3319–3328.

[55] F. Tramèr, P. Dupré, G. Rusak, G. Pellegrino, and D. Boneh, "Adversarial: Perceptual ad blocking meets adversarial machine learning," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19, 2019.

[56] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan, "Theoretically principled trade-off between robustness and accuracy," 2019.

[57] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[58] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" *CoRR*, vol. abs/1411.1792, 2014. [Online]. Available: http://arxiv.org/abs/1411.1792

[59] NVIDIA, "Nvidia dali documentation," *Documentation*.

[60] P. Lindstrom, "Error Distributions of Lossy Floating-Point Compressors," in *Joint Statistical Meetings*, 2017, pp. 2574–2589.

[61] A. Fox, J. Diffenderfer, J. Hittinger, G. Sanders, and P. Lindstrom, "Stability Analysis of Inline ZFP Compression for Floating-Point Data in Iterative Methods," *SIAM Journal on Scientific Computing*, 2020.

[62] X. Liang, S. Di, D. Tao, S. Li, S. Li, H. Guo, Z. Chen, and F. Cappello, "Error-controlled lossy compression optimized for high compression ratios of scientific datasets," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 438–447.

[63] NVIDIA, "https://github.com/rapidsai/jupyterlab-nvdashboard," *Github Link*.

[64] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[65] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *CoRR*, vol. abs/1311.2901, 2013. [Online]. Available: http://arxiv.org/abs/1311.2901