Derin Öğrenme Modellerinin Doğruluk, Süre ve Boyut Temelli Ödünleşme Değerlendirmesi

Araştırma Makalesi/Research Article

İsmail ARI, Mustafa Barış ÇAMLI

Bilgisayar Mühendisliği, Özyeğin Üniversitesi, İstanbul, Türkiye
ismail.ari@ozyegin.edu.tr, mustafa.camli@ozu.edu.tr

(Geliş/Received:22.08.2021; Kabul/Accepted:01.08.2022)
DOI: 10.17671/gazibtd.976711

Özet— Makine öğrenmesi ve özellikle derin öğrenme modellerinin gerçek-zamanlı saha uygulamalarında operasyona alınması için üç ana kriterin aynı anda optimizasyonu gerekmektedir. Bunlar modelin tahmin doğruluğu, eğitim-test süreleri ile dosya boyutu olup ilgili çalışmalarda sadece iki kriter (örnek: doğruluk-süre) beraber göz önüne alınmıştır. Ancak, modellerin tahmin doğruluğunu artırmak için oluşturulan derin sinir ağlarının (DSA) eğitim süresi ve boyutunu artırdığı, boyutunu küçültmek için yapılan çalışmaların ise doğruluğunu düşürdüğü gözlemlenmiştir. Bu üç kriter arasında bir ödünleşme yapılması gerekmektedir.

Farklı optimizasyon tekniklerinin modelin performansına etkisini göstermek için, bu makalede DSA araştırma alanında sıklıkla kullanılan ResNet50, ResNet101, VGG16, VGG19 ve EfficientNet ön-eğitimli modellerini CIFAR10, CIFAR100 görsel veri kümeleriyle test ettik. Google Colab Pro ve Tensorflow sistemi üzerinde yaptığımız başarım çalışmalardan elde edilen önemli sonuçların arasında ağırlık quantizasyonun çok-boyutlu optimizasyonunda şu ana kadarki en başarılı teknik olduğu, ağırlık kümeleme ve transfer öğrenimi tekniklerinin ise ancak 2-boyutta fayda sağladıkları söylenebilir. Çalışmamızda ayrıca, literatürde ilk defa DSA'lar için bir operasyonel skor ve modelden-modele katman aktarımı metodunu tasarlayıp, sınadık. Oluşturulan çerçevenin, yeni geliştirilen DSA modellerinin operasyona sokulmadan önce çok-boyutlu değerlendirilebilmeleri için bir referans teşkil etmesi umuyoruz.

Anahtar Kelimeler— operasyonel makine öğrenmesi, derin öğrenme, derin sinir ağları, ön-eğitimli modeller, ResNet, VGG, CIFAR, doğruluk, eğitim süresi.

Tradeoff Assessment of Deep Learning Models based on Accuracy, Time and Size

Abstract— Machine Learning and especially deep learning models need to be optimized over three main criteria concurrently, to be operationalized in real-time field applications. These criteria are model's accuracy, training-testing times and file size. Related work only considers two criteria (*e.g.* accuracy-time) together. However, it is observed that deep neural networks (DNN) designed to improve model accuracy can increase training time and size, while efforts to reduce model size can lead to lower accuracy. A trade-off needs to be made among these three criteria.

In this paper, to demonstrate the effects of different optimization techniques on model performance, we tested ResNet50, ResNet101, VGG16, VGG19, EfficientNet pre-trained models with CIFAR10, CIFAR100 image datasets, which are commonly utilized in the DNN research field. Important performance results obtained over Google Colab Pro and TensorFlow system show that weight quantization is the most successful technique so far in multi-dimensional optimization, while weight clustering and transfer learning techniques remain useful in 2-dimensions. In addition, we designed and tested a new DNN operational score and model-to-model layer transfer method for the first time in literature. We hope that our framework will constitute a multi-dimensional evaluation reference for DNN models before they are operationalized.

Keywords— operational machine learning, deep learning, deep neural networks, pre-trained models, ResNet, CIFAR, VGG, accuracy, training time.

1. INTRODUCTION

Machine Learning (ML) models can be broadly categorized as supervised or unsupervised: supervised models are trained and tested over labelled data, whereas unsupervised models depend on purely statistical properties and patterns in the datasets. Classifier models fall under supervised category, whereas clustering algorithms are unsupervised. In this paper, we focus on Deep Neural Network (DNN)-based classifier models, which have a high potential for use in real-time applications including autonomous vehicles [1], industrial quality control & maintenance [2], automated financial trading [3], IT security [4], Natural Language Processing (NLP) [5] and many more [6].

In ML classifiers, three performance criteria that need to be optimized simultaneously to achieve the highest operational efficiency are model accuracy, training-testing times, and file size. Figure 1 illustrates our framework for *Operational Machine Learning (OpML)*, where the hypothetical optimal point for all three criteria would be the *<A:0, T:0, S:0>* point, which represents the highest accuracy ratio obtained (*1.0-accuracy*) with minimal training–testing times (*0.0 sec*) and smallest model size (*0 MB*). For example, a *random* prediction that requires no training or state-keeping, would meet the time and size criteria, but have very low accuracy. As illustrated in Figure 1 with dashed arrows, trying to improve model accuracy usually results in increased model complexity followed by more training and testing times. In addition, trying to reduce the model size *(e.g.* by weight clustering or quantization) reduces accuracy, if not handled carefully. Finally, to improve model testing times researchers use more data, which increases the corpus size as well as model size, while also carrying the risk of model overfitting. When designing new models and/or tuning pre-trained models, the operational goal should be to stay within the inner triangles in Figure 1, while pushing the limits on either one of these criteria.

We can combine 3-dimensions into a simple, but efficient operational score (*OpScore*) as shown in Equation 1:

$$OpScore = -\log_{10}(1/(1-Acc)) + \log_{10} Time + \log_{10} Size(MB) \qquad (1)$$

where the optimal score is close to zero and between [0,5] for most practical cases (see Sec.4). Note that, increasing accuracy in Equ. 1 decreases the score, whereas increasing model time and size values increase the score, denoting increasing operational costs. The operational score can be used separately for training and testing times, a weighted average or total of both depending on the operational scenario. We will discuss model training, testing (or inference) and even pre-training (tuning & pruning) times in this paper to understand their implications on real-time field applications. While OpScore is directly applicable to different DNN models, a value-based comparison is done over the same dataset for fairness.

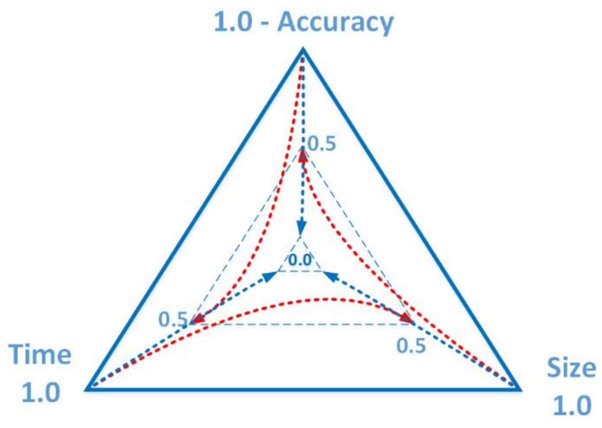


Figure 1. Dimensions of model performance criteria to be optimized concurrently are model accuracy, time and size. We use this scheme as a reference to compare novel DNN models and optimization techniques in terms of their operational efficiency.

Figure 2 illustrates the general structure and components of a DNN classifier model. As an example, the image of an animal such as a cat or a dog is passed through the DNN to generate a correct classification "cat" in text format. The *depth* of a DNN model is measured by the number of hidden layers. As DNN models get deeper they contain hundreds of convolutional layers as well as activation, pooling, regularization, attention, normalization, and dense layers that carry thousands of functions and millions of weights. These facts are valid for Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and their hybrids called R-CNN.

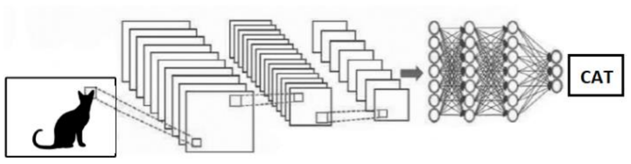


Figure 2. DNN, CNN or RNN models have layered structures containing millions of weights, which account for 100s of millions of floating-point operations over multiple epochs of training and testing.

After training, the models are serialized and saved in storage systems using special Hierarchical Data Formats (HDF) [7] or Open Neural Network Exchange (ONNX) [8] file formats. ONNX is becoming the de facto standard for easily converting and sharing pre-trained models. These models can be uploaded to public model repositories called "model zoo" [9] for online sharing. They are later downloaded locally, possibly in edge or Internet of Things (IoT) devices and operationalized by deserializing the model file into the memory. Next, every input (*e.g.* image) gets turned into a matrix of float-type variables by the input layer and passed through model layers to deliver correct classification results at the output layer. Convolutional layers generate feature maps from the input. Pooling layers (average or max pooling) down sample feature maps to provide translational invariance. In between these two, there are activation functions (ReLU, Sigmoid, *etc.*) that are used to introduce nonlinearity. It takes millions of sample images to train accurate models. DNN models can take days to train over traditional Central Processing Units (CPU). Fortunately, massively parallel Graphical Processing Unit (GPU) and novel Tensor Processing Units (TPU) [10, 11] come to rescue to reduce these model training times to hours and minutes.

One potential caveat in model training is over-fitting, where the model strictly and only learns the training data. To avoid over-fitting, techniques such as data cross-validation and activation functions are used. Models may need to be retrained with new data in the field for continuous improvement, yet mobile edge devices [12] are usually not as powerful as the central servers or cloud resources. Therefore, field success depends on optimization of model training time as well as its accuracy and size, concurrently. Model testing does not require weight updates and therefore it is faster than model training. However, since inference time effects field performance it must be measured and optimized carefully.

Contributions of our paper are as follows:

- We develop a 3D framework and a score for evaluation and ranking of DNN model to be operationalized in the field,
- We present a detailed multi-dimensional evaluation of commonly used DNN models including ResNet, VGG and EfficientNet with different datasets,
- We calculate an *OpScore* and rank models using the same unified metric,
- We evaluate the performance effects of modern CPU accelerator chips, including GPU and TPU over DNN training and testing times,
- We quantify the cost of model optimization techniques such as weight clustering,
- We report on sensitivity of model accuracy over increasing number of dense & hidden layers “unfrozen” to be retrained,
- We test a new method that we call “model-to-model layer transfer”.

The rest of the paper is organized as follows. Section 2 gives the background on DNN models and their Key Performance Indicators (KPI) including model accuracy, training and testing times as well as size reduction techniques. Section 3 describes our methodology including experimental setup, datasets and the concept of transfer learning. Section 4 presents the detailed performance assessment of several DNN models and optimization techniques including model quantization and clustering using proposed tradeoff assessment scheme. Most of the contribution listed above are realized in this section. Section 5 discusses related work on model optimizations in comparison to our proposed framework and findings. Section 6 concludes the paper also describing future work.

2. BACKGROUND

In this section, we first give a quick background on KPI for OpML (model accuracy, time, size). Next, we describe properties of reference DNN models used in this research.

2.1. Model Accuracy

When a set of N inputs are fed into a classifier, the four possible outcomes of the classification are True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN), thus $N=(TP+FP+TN+FN)$. Accuracy (Eq.2) is simply measured as the ratio of “truly classified” instances over N. Model’s accuracy, recall, precision and F1-measure formulas are given below by Equations 2-5, respectively.

$$\text{Accuracy} = (\text{TP+TN})/\text{N} \tag{2}$$

$$\text{Recall}= \text{TP}/(\text{TP+FN}) \tag{3}$$

$$\text{Precision} = \text{TP}/(\text{TP+FP}) \tag{4}$$

$$\text{F1-score} = 2/(1/\text{Precision}+1/\text{Recall}) \tag{5}$$

These are commonly used for model accuracy measures but have small semantic differences and implications. In cases where the items to catch are extremely rare (*e.g.*, a medical test to detect a rare cancer type) precision and recall should also be employed. Recall (Eq.3) focuses on model’s ability to correctly detect (*i.e.*, “not miss” or make “False Negative”) items that should be placed in the selected class, while Precision (Eq.4) denotes model’s ability “not to misplace or confuse” (FP) items belonging to other sets into the selected class. F1-score (Eq.5) is the harmonic average of recall and precision values, giving a unified view of both success metrics.

2.2. Model Training and Testing Time

NN are difficult to deploy on IoT and embedded systems with limited resources since they are both compute and memory intensive. Most of the recent R&D efforts in OpML focus on reducing model training and testing (*a.k.a.* inference, serving) times. CPU-GPU manufacturers also produce ML-accelerator chips such as Tensor Processing Units (TPU) to speed up floating point operations in DNN. In cases where a model contains mixed-precision values (float16, float32, float64) together, chip-level support for mixed-precision operations [13] is also crucial for high performance. Otherwise, every arithmetic-logic operation also requires a value conversion (among float16-32) having a diverse performance effect.

2.3. Model Size Reduction Techniques

Modern pretrained DNN model files generally range between 10 Megabytes (MB) to 1 Gigabytes (GB) in size. We compared two techniques for model size reduction in this paper, namely quantization [l4, 15] and weight sharing (*a.k.a* weight clustering) [16]. Quantization is a technique to reduce model size by reducing the number of bytes used for each NN weight value. For example, if we use a float16 value (2-bytes) instead of float32 (4-bytes) the model size should be approximately reduced by half. However, since the weights are highly tuned to make accurate predictions and reducing weight precision could also result in loss of accuracy.

In weight sharing technique [16], all the weights in a model are clustered using K-means algorithm (or alike) and centroid values of each cluster are used as a single, shared value for the weights that belong to that cluster. Thus, the number of DNN weights to maintain is reduced from millions to thousands. Potentially, this also results in caching of weight values in L1-L2-L3 caches, which could speed up model training and testing times. However, like quantization, replacing thousands of values with their shared value also results in loss of model accuracy. We test this hypothesis and present comparative results in Sec. 4.

2.4. Models: ResNet50-101, VGG16-19, EfficientNet

Most image classification DNN models including ResNet50, ResNet101 and VGG were originally trained over the ImageNet dataset [17], but they can be retrained and used with other datasets. Table 1 shows their original and quantized sizes as well as layer counts. We can generally observe the 2x reduction in size due to float32 to float16 type change over all models. ResNet50 consists of 176 layers, which are inside 50 blocks (hence the name ResNet50) each consisting of a sequential group of <Conv-2D, Batch-Normalization, ReLU> layers, whereas ResNet101 consists of 346 layers inside 101 such carefully-engineered blocks. ResNet50 original model file is 103 MB whereas ResNet101 model is 180 MB, which is approximately proportional to the number of layers. Since we used the CIFAR datasets, which were different from the ImageNet dataset that was originally used to pretrain ResNet50-101 models, we had to retrain and test them. This would be the case with any operational ML scenario when pretrained models are employed in field applications. Note that the architecture of models (blocks and layers) are not affected by weight quantization techniques.

VGG16 was the winner of ImageNet classification competition in 2014 [18]. While it doesn't have as many layers as ResNet, it has very wide convolutional layers that carry a total of 138 Million parameters that reflect its original model size of 528 MB. VGG19 model is very similar in structure to VGG16 except 3 more convolutional layers in the middle, which increases VGG19 size to 549MB for a slightly better accuracy. When we downloaded models into Keras [19], we omitted the top-layers of the pretrained models, therefore Table 1 reports smaller model sizes without the top layers, obtained before transfer learning.

EfficientNet [20,21] studies model scaling issues and identifies that "carefully balancing network depth, width, and resolution can lead to better performance". They proposed a method that uniformly scales all dimensions using a compound coefficient using which they designed a range of EfficientNet from B0→B7. They demonstrated effectiveness of this method up to *6x* in speed and *8x* in size over ResNet50-101-152. Table 1 shows the modest size of B0 compared to ResNets and VGG. We'll compare their time-accuracy performances in Section 4.

Table 1: Originally used and quantized model sizes (MB).

Model Name	Original Size (MB)	Quantized Size (MB)	Number of Layers
ResNet50	103	48	176
ResNet101	180	86	346
VGG16	59	29	16
VGG19	80	40	19
EfficientNet-B0	16	8	238

3. METHODOLOGY

This section describes our methodology for tradeoff assessment of different DNN models over different benchmark datasets. Compared techniques include quantization, weight sharing (*clustering)*, transfer learning using top (dense + hidden) layer unfreezing (Top-1, Top-10), and direct model-to-model layer-based reuse of weights (*e.g.*, from ResNet101 to ResNet50). We trained each model for 10 Epochs, which allowed for accuracy convergence.

3.1. Experimental Systems and Software

We used Google Colab Pro cloud system [10], TensorFlow [11] Keras [19] and Scikit-learn [22] Python libraries to calculate our model accuracies, training times and sizes. Google Colab Pro provides Intel Xeon® CPU @ 2.30GHz 16-core 32GB DRAM machines with different GPU generations: NVIDIA Tesla Pascal (P100) PCIE-16GB and Volta (V100) SXM2-16GB. A major difference between these two GPU's is that the newer V100 supports mixed-precision operations, while P100 does not. We will discuss the effects of system differences in performance.

3.2. Datasets: CIFAR10 and CIFAR100

Canadian Institute For Advanced Research (CIFAR) has published labelled image datasets that have become the de-facto benchmark for image classification models. CIFAR10 and CIFAR100 include 60,000 32x32 bit color images, belonging to 10 different classes and 100 different classes, respectively. Image contents exclusively show animals (bird, cat, dog, horse, *etc.*), vehicles (automobiles, planes, ships, trucks), flowers, vegetables, trees and people. CIFAR10 has 6000 images from each of 10 class types, whereas CIFAR100 has 600 images from each of 100 types. In both datasets, 50,000 images are used for training and 10,000 for testing purposes. These datasets are identically and independently distributed (IID) [23], basically having no label and quantity skews. Since we compare models over a centralized system architecture in this paper, having non-IID datasets would be equivalent to leaving some classes out of the question and render our results incomparable with respect to other related work in this domain. Yet, we included a sensitivity analysis having different splits (Table 3: 50-50%, 70-30%, 90-10%) for training and testing CIFAR datasets. In another study [24], we currently tested models with non-IID datasets with respect to their impact on distributed, federated learning scenarios.

3.3. Unfreezing Model Layers for Transfer Learning

Transfer learning allows a model, pretrained with a former dataset, to be unfrozen and retrained with the new dataset at hand. Top layers are the ones closer to the model output, which are also the fully-connected, dense layers. Since the input layers are the same for all models (ingesting 32x32 bit images), the output layer and the hidden layers are of utmost importance in improving model performance. Therefore, we start by retraining models' weights from Top-to-Bottom, where Top denotes the model output. We used the term "unfreeze" to denote the number of layers to be retrained. For example, when we "unfreeze" Top-1 (dense) layer the dense layer is retrained, but hidden layers are not affected, whereas when we unfreeze Top-10 layers, we update and adapt 1 dense layer + 9 hidden layers. VGG16-VGG19 models we used had 2 fully-connected (dense) layers [25] and 13-16 hidden layers, respectively. For fairness, we unfroze and retrained the same number of dense & hidden layers in each model type.

After downloading selected pretrained models into Keras for transfer learning, we initialized models' top layers with Keras' default Glorot (Xavier) uniform initializer. This initializer is known to fix the exploding & vanishing gradient problems associated with random initialization [26]. We can regard our new model-to-model layer transfer strategy as a coarse-granularity initialization technique where the top-layers of successor models (e.g. ResNet101 to ResNet50) are transferred with the presumptions that (a) there is an evolutionary relation between the two model generations which guarantees input-output compatibility (2) the successor model has higher accuracy and the predecessor model has smaller size. We report the results in Section 4.

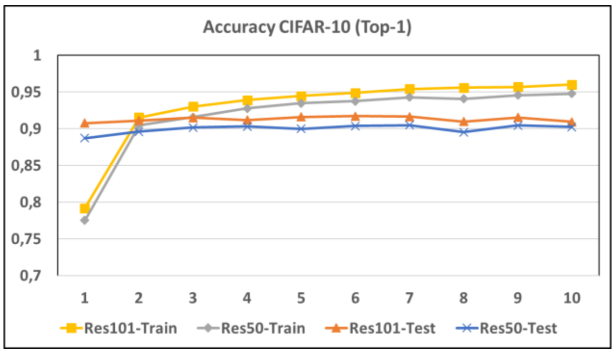
4. RESULTS AND DISCUSSION

In this section, we first present comparisons of the baseline ResNet50-ResNet101 models over selected datasets. We start by comparing their accuracies and continue with training-testing times over two different GPU systems. Then, we compare different model optimization techniques in terms of their impact on model accuracy, time, and size. We fix both training and testing epochs at 10 for all experiments for reference. While training models for 10 epochs provides weight updates and accuracy convergence, testing models for 10 epochs serves a statistical validation for accuracy and inference timing measurements.

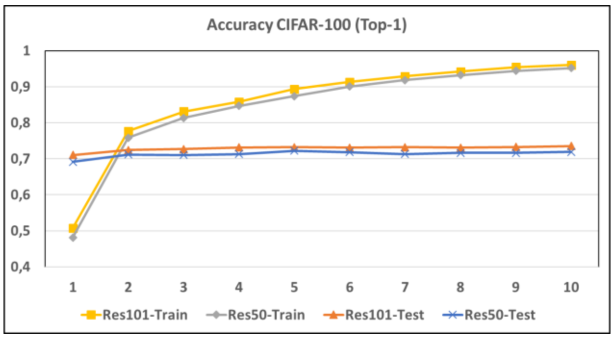
4.1. Analysis and Comparison of Model Accuracies

Figure 3 shows the comparison of classification accuracies for ResNet50-101 models over CIFAR10 and CIFAR100 datasets while retraining Top-1 (Dense) and Top-10 (1 Dense + 9 Hidden) layers. First, we observe in Figure 3a-3b that retraining of even the last (Top-1) layer in both ResNet50 and ResNet101 results in a significant improvement of accuracy. Since original models were trained using ImageNet, this case shows the immediate benefits of transfer learning for both CIFAR10 (from ~0.78-to-0.96) and CIFAR100 (from ~0.5-to-0.95).

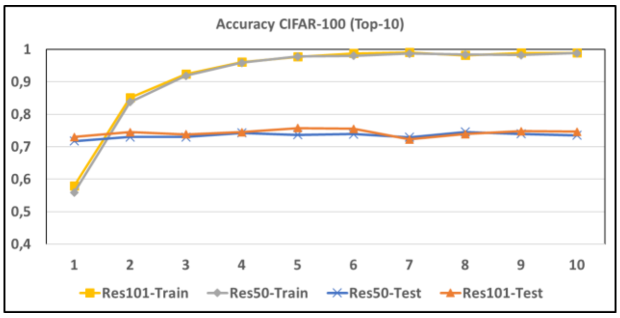
Second, training accuracies are always higher than testing accuracies and the difference between them denotes the amount of overfitting (since we have IID data). Third, ResNet101 only has *slightly higher* training and testing



(a) Accuracy with CIFAR10 dataset Top-1 retraining.



(b) Accuracy with CIFAR100 dataset Top-1 retraining.



(c) Accuracy with CIFAR100 dataset Top-10 retraining.

Figure 3. Comparison of accuracies for Resnet50 and Resnet101 models with (a) CIFAR10 Top-1, (b) CIFAR100 Top-1, and (c) CIFAR100 Top-10 layers retraining.

Table 2: ResNet50-101 accuracy values over CIFAR10-100 datasets using Recall, Precision and F1 measures.

Model	Dataset	Recall	Precision	F1
CIFAR10	Res50-Top10	0.92	0.92	0.92
	Res101-Top10	0.99	0.99	0.99
CIFAR100	Res50-Top10	0.72	0.76	0.74
	Res101-Top10	0.73	0.77	0.75

accuracy compared to ResNet50. However, the dataset's image complexity has a bigger role in the final result, *i.e.* CIFAR100 being more complex than CIFAR10 leads to lower accuracies (0.7 *vs.* 0.9). Otherwise, ResNet50-101 models have very close validation accuracies in both datasets. This proves that additional efforts and complexity added to create ResNet101 model *did not contribute significantly to higher* accuracy while it caused 60-80% increase in model training time and sizes. ResNet101 takes 1.6-1.8x the training time of ResNet50 (see Table3: P100: 470sec *vs.* 840sec, V100: 280sec vs. 470sec) and increases model size by ~1.74x (see: Table1: 103MB→180MB). We therefore find that operationalizing models in the field to train & test them over different datasets can be as important as offline feature and model engineering.

In Figure 3b-3c, a comparison of models over retraining of Top-1 vs. Top-10 layers with CIFAR100 shows faster convergence in Top-10 layers (*e.g.,* 0.9 point reached at 3 epochs in Top-10 vs. 5 epochs in Top-1) and a slightly higher accuracy result with Top-10 at the end (Top-1: 0.95 vs. Top-10:0.98). Note that we refer to unfrozen layers and not the probably of a class prediction being in Top-1 vs. Top-5 lists, since Top-5 result in an unacceptably higher accuracies for field applications.

In Table 2, we briefly report different accuracy measures including Recall, Precision and F1 for ResNet50-ResNet101 models over different datasets. Since these benchmark datasets were IID for all class types, all of the measures gave similar accuracy results: ~0.9 for CIFAR10 and ~0.7 for CIFAR100. There are no label or quantity skews to cause reductions in precision or recall. We tested these issues in our recent work on distributed federated learning [24] where non-IID datasets [27,28] can have more impact on federated model performance [29].

While not shown here for brevity, we also tested a Random Model (*i.e.* naive reference) with CIFAR100 dataset for cross-examination and correctness control and we obtained accuracy: 0.01 (1/100), recall: 0.01, precision: 0.01, Area Under Curve (AUC): 0.50 as expected.

4.2. Model Training-Testing Times and GPU Effects

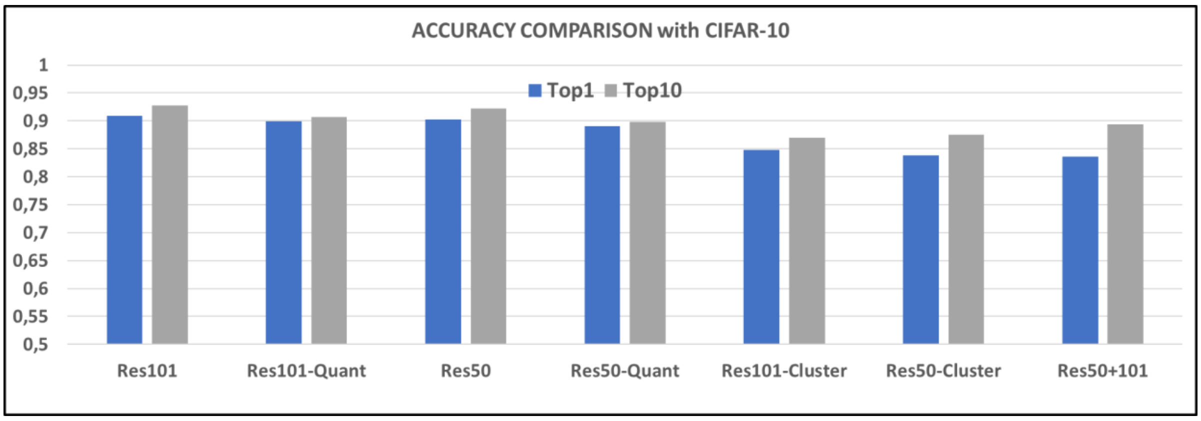
In this part, we compare training and testing times of the baseline ResNet50-ResNet101 models over CIFAR100 dataset and by unfreezing Top10 layers. As a part of our measurement and sensitivity analysis we run the experiments over two different GPUs (P100-V100) and change the ratio of training-testing datasets (50-50%, 70-30%, 90-10%) to validate per mini-batch processing times. We used TensorFlow Keras library and its *evaluate()* function. The *batch_size* was 128 items for both training and testing.

Table 3 shows that the training time for ResNet50 was 26.8 sec/30,000 images (~0.9 ms/image) and for ResNet101 42.7 sec/30,000 images (~1.4 ms/image). For example, if a real-time application sets the inference time limit for the DNN model to be 1ms, ResNet50 would be eligible for field deployment whereas ResNet101 would not qualify. Depending on the ratio of increase in training data the training times increase and the testing times decrease, but the total time remains consistent around ~53 seconds for ResNet50 and ~84 second for ResNet101 over P100 GPU. We observe that both training and testing times of Resnet101 are 1.6-1.8x slower than ResNet50 for all data sizes (*e.g.* Res101/Res50 testing for 90-10%: 8.2/4.9=1.67), which is due to the model size and complexity.

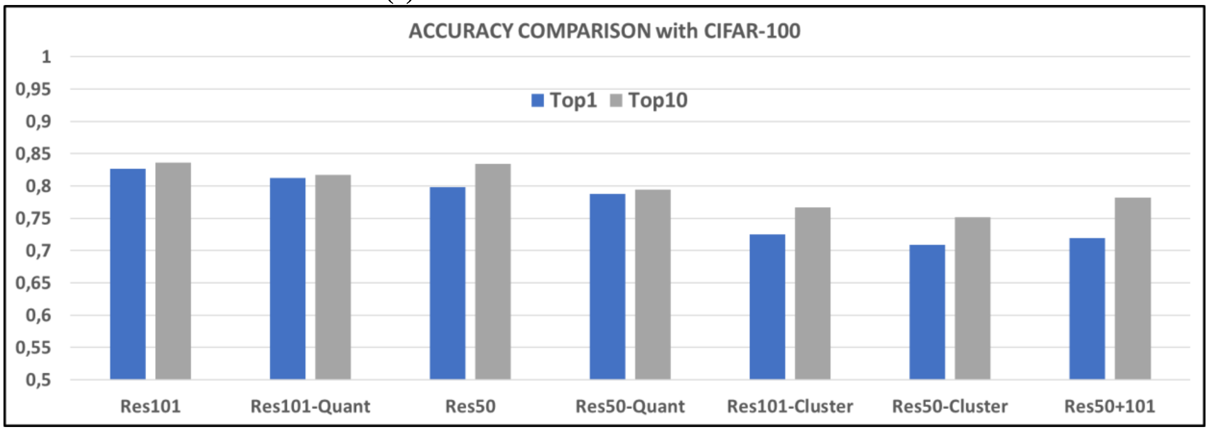
Training and testing times improved by ~45% for both models when we used Volta (V100) GPU, since it has ~45% more GPU cores (+TPU cores), higher memory bandwidth as well as memory capacity (32GB vs 16GB). For example, Res50 50-50% was 14.5 sec vs. 26.8 sec. In an offline analysis, we also compared Adam or SGD optimizers and observed no performance impact on training time. The total time was similarly ~29 sec. for ResNet50 and ~47 sec. for ResNet101 with V100.

Table 3: Effects of different GPU accelerators (P100,V100) on model training-testing time for different data ratios (50-50%, 70-30%, 90-10%) using CIFAR100 dataset and Top-10 layers for retraining.

GPU	Ratio (%)	Train Time (sec)		Test Time (sec)		Total Time (sec)	
		Res50	Res101	Res50	Res101	Res50	Res101
P100 (Pascal)	50-50	26.8	42.7	27.8	40.5	64.6	83.2
	70-30	37.4	59.9	14.7	24.4	52.1	84.3
	90-10	48.1	76.9	4.9	8.2	53.0	85.1
V100 (Volta)	50-50	14.5	23.7	15.0	22.7	29.5	46.4
	70-30	20.3	33.1	8.1	20.6	28.4	53.7
	90-10	25.9	42.5	2.7	5.2	28.6	47.7



(a) Accuracies of models with CIFAR10 dataset.



(b) Accuracies of models with CIFAR100 dataset.

Figure 4. Accuracy comparison of different pre-trained DNN models and their optimized (quantized, clustered, transfer learning, model-to-model layer transfer) versions after retraining of Top-1 and Top-10 layers.

4.3. Comparison of Different Model Optimization Techniques on Accuracy

Figure 4 shows the accuracy results for the basic Resnet50-Resnet101 models and their optimized (quantized, weight-clustered, transferred) versions over CIFAR10 (Fig.4a) and CIFAR100 (Fig.4b) datasets, respectively and in descending order of their accuracy achievements. We report the accuracies obtained after 10 epochs of retraining.

First, we observed that unfreezing Top-10 Layers resulted in higher accuracy compared to unfreezing only Top-1 layer. For Res50-cluster and Resnet101-cluster models, although we used a relatively large number of clusters K=8192 to obtain high-precision weights, the accuracy drop over the original model was significant (~6%) compared to the quantization optimization. Finally, transferring top layers directly from ResNet101 to the ResNet50 model (Res50+101 model-2-model) resulted in the lowest accuracy before retraining. However, this strategy was still more promising than the Res50-Cluster model after retraining with both CIFAR10 and CIFAR100. This technique requires more investigation by careful selection of the transferred blocks.

Also, accuracies of all unmodified (Res50-Res101) and optimized (quantized, clustered, transferred) models are higher for CIFAR10 (~0.85-0.92), then CIFAR100 (~0.65-0.75). Surprisingly, quantized versions of Res50-Res101 from float32-to-float16 resulted in a very small decrease (<2%) in the accuracy, although we gained a significant size reduction (see Sec 4.5).

Based on this subtle, but important difference we decided to conduct another sensitivity analysis of model depth over Resnet ad VGG. Figure 5 shows that ResNet benefited more as we unfroze more layers, whereas VGG did not.

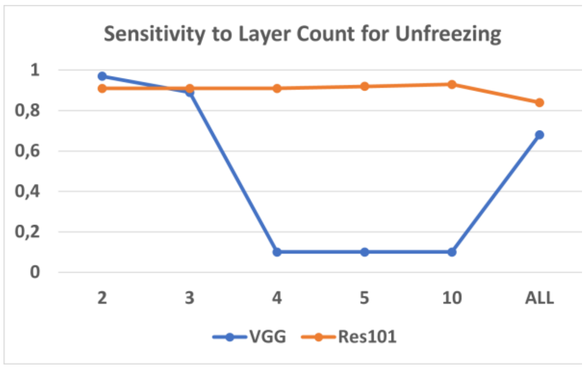


Figure 5: Sensitivity of Resnet101 and VGG model testing accuracy over different count of Top layers unfrozen and retrained for 10 epochs.

Table 4: Training and testing times with different NVIDIA Tesla GPUs (P100-V100).

GPU	Dataset-Layers	Resnet50	Res50-Quant	Res50-Cluster	Resnet101	Res101-Quant	Res101-Cluster	Res50+101
P100	CIF10-Top1	472	630	471	842	1062	847	838
	CIF100-Top1	476	642	476	841	1048	861	841
	CIF10-Top10	528	671	540	894	1108	892	896
	CIF100-Top10	540	691	524	895	1121	898	902
V100	CIF10-Top1	286	226	286	474	329	470	477
	CIF100-Top1	286	218	285	493	325	470	477
	CIF10-Top10	318	238	323	509	343	501	510
	CIF100-Top10	319	238	319	509	343	507	514

Including dense (Top 1-3) classifier layers to transfer learning is generally a good idea but moving further into the hidden layers does not always guarantee accuracy gains. This may be due to the fact that model architectures are originally designed and fit for their initial datasets, which is ImageNet dataset in this case.

4.4. Comparisons of Training-Testing Times

Table 4 shows the total times for 10 training + 10 testing epochs with 50,000 training and 10,000 testing images (ratio ~83-17%) of original ResNet50-101 models and their optimized (quantized, clustered, transfer learned, and model-2-model layer transfer) versions over P100 and V100 GPUs. There are two main findings within these results: (1) clustering does not change the train-test times, but there is a preparation time cost for these models, (2) quantization is highly-sensitive to GPU's mixed-precision support at the hardware level: if there is support (V100) the resulting times are 25% faster, but if not (P100) the train-test times can be 30% slower. Resnet50+101 model-to-model layer transfer has similar time performance to weight clustering.

4.5. 3D Comparisons Including Size Reduction

In Table 1 we showed that quantization can reduce model sizes by 50% (or *2x*). Note that the accuracies presented in Figure 4 were comparable within 2% of original models. Quantization with mixed-precision support leads to the best results in multi-dimensional assessment. However, edge nodes currently don't have GPUs with mixed-precision support. These high-end GPUs are powerful, but expensive and energy-consuming devices. So, their operational use at the edge for mass scale is currently not feasible.

Figure 6 shows the model inference time (X-axis) *vs.* model accuracy (Y-axis) and bigger bubble sizes refer to the bigger model sizes. High accuracy, small inference time and smallest model sizes are preferable. From Fig. 6 we can see that EfficientNet-B0 Top-1-10 quant models have the highest accuracy, smallest inference time and size. In comparison, VGG19 models are at the outer rim of the chart with lower accuracy, higher inference times and relatively larger bubble sizes. Finally, Resnet101 models have relatively larger sizes and inference times compared to EfficientNet-B0 and Resnet50 models (respectively, their quant versions) although their accuracies are also high. For each model, we observe that quantization increases the inference times while it makes the model sizes smaller. While Fig 6. is illustrative, it cannot help us make a final decision in model selection. Therefore, we use a new operational score to combine conflicting dimensions and rank our selected models. Table 5 shows the model rankings based on OpScore. We added a fictitious MIN and MAX model in the table, which picks the best and worst values from among the analyzed models in the table. These values can later be used for MIN-MAX normalization purposes for each application. Theoretically, our OpScore can take negative or relatively high values, but it will be between [0,5] for almost all practical cases. For example, a very poor performing model with *<Acc:0,01; Size=1GB; Time=100sec>* has OpScore=5.0, whereas a relatively good model with *<Acc:0,99; Size=10MB; Time=10sec>* has OpScore=0.0. OpScore values of all the real pretrained models for CIFAR10 are between [0,67-2,63] in Table 5.

Figure 7 shows the performance results of all models over CIFAR100 dataset using a bubble chart for visual comparison. Accuracy values have dropped for all the models due to classification complexity of CIFAR100 as expected. Yet, the general patterns and findings for CIFAR10 reported in Figure 6 are still valid. Inference times and model sizes are similar. Table 6 show the results ranked by OpScore, respectively. Due to the decrease in accuracy, OpScore values of all models with CIFAR100 are higher between [1,15-3,10]. There are only small changes in the ordering of the models.

Figure 8 presents the sensitivity analysis of clustering technique (K=512→16,384) on model accuracy. Note that the accuracy drops from ~0.9 to ~0.6 for CIFAR10 and ~0.7 to ~0.3 as the number of weight clusters decrease from K=8192 to K=1024. This is due to the immense loss of information in NN weights. Table 7 presents the same analysis for effects on model size and model clustering time. Note that this clustering time is a "preparation" time that has to be spent before the model is used for training and testing. Therefore, it is an additional cost. Most of the related work do not report their model preparation times, which could become a significant limit in practice. The main goal of clustering is to reduce model size. Table 7 shows that this goal is achieved (103MB→39MB), but at the expense of accuracy loss and added clustering times.

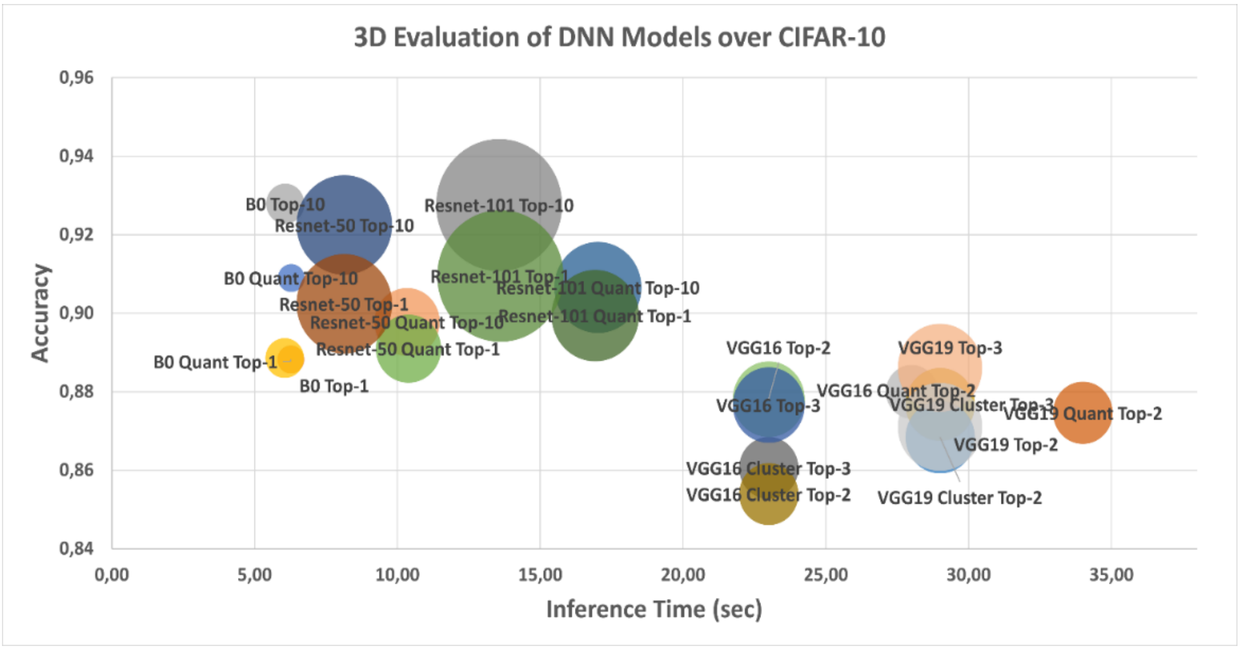


Figure 6. 3D Bubble chart comparison of selected DNN models with CIFAR10.

Table 5. Ranking of evaluated models based on 3D KPI and unified OpScore using CIFAR10 dataset. MIN-MAX values are obtained by selecting the best and worst value for each column and generating a score for this fictitious model as reference; this can be used for normalization. Note that best and worst OpScore values range between 0-5.

Model Name	Accuracy	Size (MB)	Training Time (sec)	Inference Time (sec)	Score
MIN	0,93	8,10	27,72	6,06	0,54
B0 Quant Top-10	0,91	8,10	31,77	6,28	0,67
B0 Quant Top-1	0,89	8,10	27,98	6,27	0,75
B0 Top-10	0,93	16,60	29,92	6,07	0,86
B0 Top-1	0,89	16,60	27,72	6,06	1,05
B0 Cluster Top-10	0,40	11,00	29,34	6,07	1,60
Resnet-50 Quant Top-10	0,90	48,00	56,41	10,34	1,71
B0 Cluster Top-1	0,21	11,00	27,75	6,07	1,72
Resnet-50 Quant Top-1	0,89	48,00	51,51	10,39	1,74
Resnet-50 Top-10	0,92	103,00	44,48	8,14	1,81
Resnet-50 Top-1	0,90	103,00	40,09	8,14	1,91
VGG16 Quant Top-2	0,88	29,50	136,00	28,00	2,00
Resnet-50 Cluster Top-10	0,75	61,00	44,48	8,14	2,09
VGG16 Cluster Top-3	0,86	39,90	118,00	23,00	2,11
VGG16 Cluster Top-2	0,85	39,90	114,00	23,00	2,13
Resnet-101 Quant Top-10	0,91	86,00	89,78	17,02	2,14
Resnet-101 Quant Top-1	0,90	86,00	84,29	16,92	2,17
Resnet-50 Cluster Top-1	0,70	61,00	40,16	8,15	2,17
VGG16 Top-2	0,88	58,90	114,13	23,00	2,22
VGG16 Top-3	0,88	58,90	117,78	23,00	2,22
VGG19 Quant Top-2	0,87	40,10	167,48	34,00	2,23
Resnet-101 Top-10	0,93	180,00	71,18	13,56	2,25
VGG19 Cluster Top-3	0,88	54,30	144,85	29,00	2,29
VGG19 Cluster Top-2	0,87	54,30	141,33	29,00	2,32
Resnet-101 Top-1	0,91	180,00	67,51	13,60	2,35
VGG16 Quant Top-3	0,73	29,50	140,00	28,00	2,35
VGG19 Top-3	0,89	80,20	144,80	29,00	2,42
VGG19 Top-2	0,87	80,20	141,23	29,00	2,48
VGG19 Quant Top-3	0,77	40,10	171,31	34,00	2,50
Resnet-101 Cluster Top-10	0,77	115,00	71,25	13,56	2,55
Resnet-101 Cluster Top-1	0,73	115,00	67,51	13,63	2,63
MAX	0,21	180,00	171,00	34,00	3,68

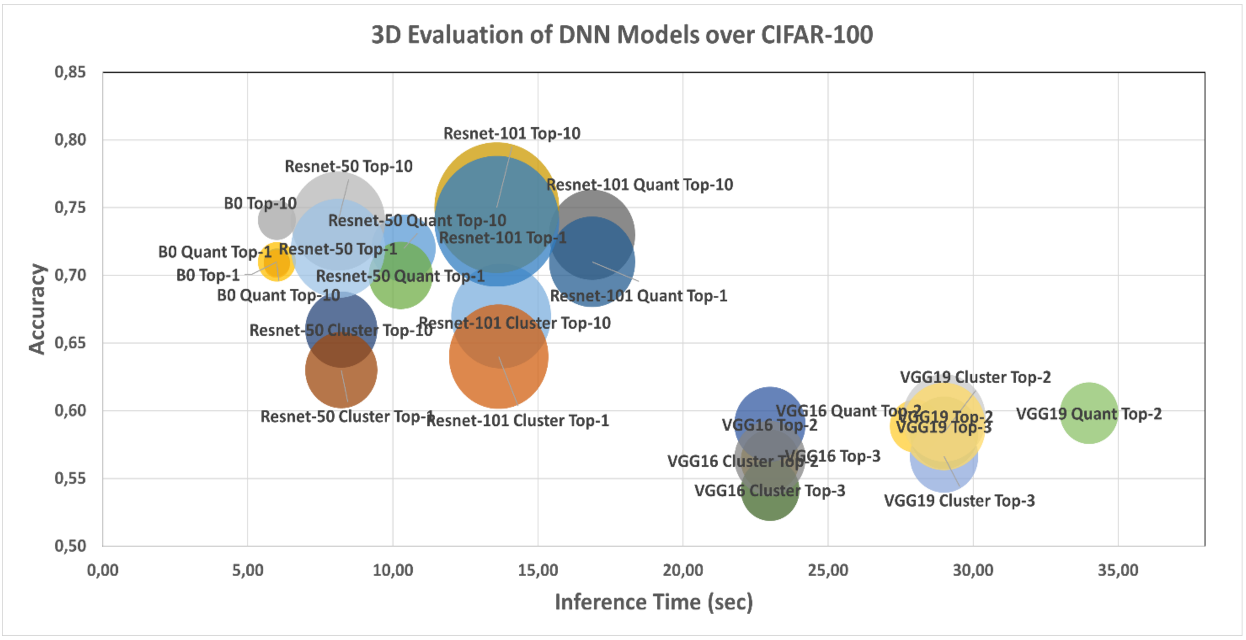
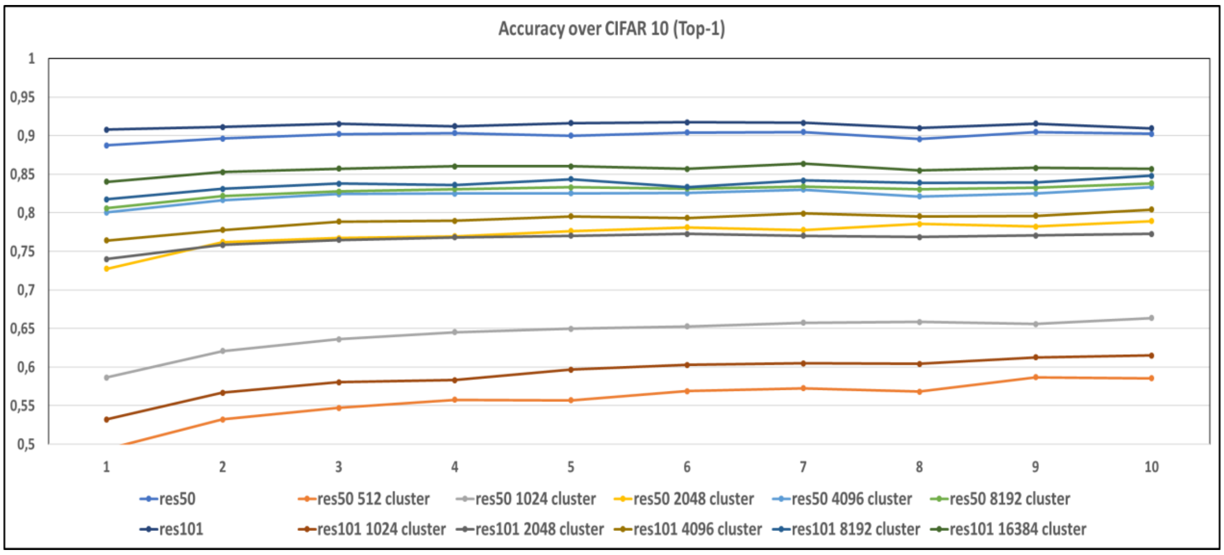


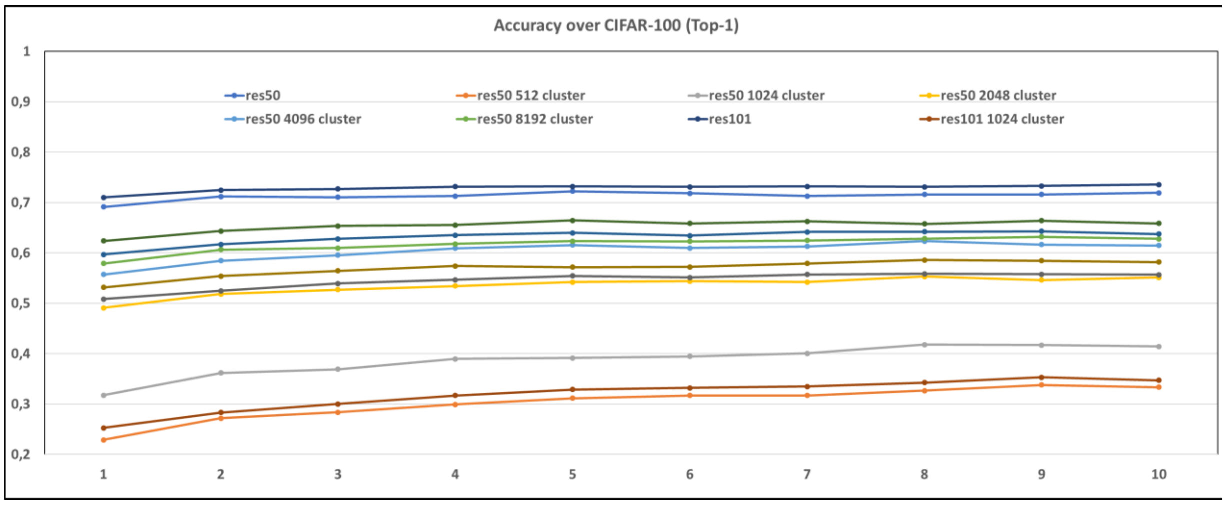
Figure 7. 3D Bubble chart comparison of selected DNN models with CIFAR100.

Table 6. Table 5. Ranking of evaluated models based on 3D KPI and unified OpScore using CIFAR100 dataset. MIN-MAX values are obtained by selecting the best and worst value for each column and generating a score for this fictitious model as reference; this can be used for normalization. Note that best and worst OpScore values range between 0-5.

Model Name	Accuracy	Size (MB)	Training Time (sec)	Inference Time (sec)	Score
MIN	0,75	8,10	27,28	6	1,08
B0 Quant Top-1	0,71	8,10	27,77	6,00	1,15
B0 Quant Top-10	0,71	8,10	30,94	6,00	1,15
B0 Top-10	0,74	16,60	29,63	6,00	1,41
B0 Top-1	0,71	16,60	27,36	6,00	1,46
B0 Cluster Top-10	0,12	11,00	29,59	6,00	1,76
B0 Cluster Top-1	0,05	11,00	27,28	6,00	1,80
Resnet-50 Quant Top-10	0,72	48,00	56,63	10,37	2,14
Resnet-50 Quant Top-1	0,70	48,00	51,05	10,27	2,17
Resnet-50 Cluster Top-10	0,66	61,00	44,61	8,22	2,23
Resnet-50 Cluster Top-1	0,63	61,00	40,50	8,22	2,27
Resnet-50 Top-10	0,74	103,00	44,47	8,11	2,34
VGG16 Quant Top-2	0,59	29,50	136,27	28,00	2,53
Resnet-101 Quant Top-10	0,73	86,00	88,43	16,87	2,59
VGG16 Cluster Top-2	0,56	39,90	113,84	23,00	2,60
Resnet-101 Quant Top-1	0,71	86,00	84,04	16,87	2,62
VGG16 Cluster Top-3	0,54	39,90	117,50	23,00	2,62
Resnet-101 Cluster Top-10	0,67	115,00	71,25	13,73	2,72
VGG19 Quant Top-2	0,60	40,1	167,37	34	2,74
VGG16 Top-2	0,59	58,90	113,92	23,00	2,74
Resnet-101 Cluster Top-1	0,64	115,00	67,50	13,65	2,75
VGG16 Top-3	0,56	58,90	117,63	23,00	2,77
Resnet-101 Top-10	0,75	180,00	70,18	13,58	2,79
Resnet-101 Top-1	0,74	180,00	67,3	13,58	2,80
VGG19 Cluster Top-2	0,58	54,3	141,1	29	2,82
VGG19 Cluster Top-3	0,57	54,3	144,5	29	2,83
VGG16 Quant Top-3	0,12	29,50	139,65	28,00	2,86
VGG19 Top-2	0,59	80,20	141,24	29,00	2,97
VGG19 Top-3	0,59	80,20	144,92	29,00	2,98
Resnet-50 Top-1	0,72	103,00	40,06	8,11	3,06
VGG19 Quant Top-3	0,07	40,1	171	34	3,10
MAX	0,07	180,00	167,37	29	3,69



(a) Sensitivity of ResNet accuracy to number of weight clusters over CIFAR10 dataset.



(b) Sensitivity of ResNet accuracy to number of weight clusters over CIFAR100 dataset.

Figure 8. Accuracy comparison of clustered Resnet50-101 pre-trained models over different cluster counts.

Table 7: Sensitivity of Resnet50-101 models per number of clusters over size (MB) and time (sec).

Model Name - Cluster#	Size (MB)	Clustering Time (sec)
Res50 – Original	103	-
Resnet50 -512	39	398
Resnet50 -1024	42	561
Resnet50 -2048	47	911
Resnet50 -4096	55	1556
Resnet50 -8192	61	2894
Res101 – Original	180	-
Resnet101 -512	76	1014
Resnet101 -1024	82	1560
Resnet101 -2048	93	2692
Resnet101 -4096	115	4808
Resnet101 -8192	130	7800

5. RELATED WORK

While increasing the depth in DNN, generally increases the accuracy of image classification, this accuracy gets saturated and then even drops as more layers are added. He *et al.* [17] proposed to reformulate plain or stacked layers as residual functions leading to deeper networks without increasing the costs of training. These networks were called Residual Networks or *ResNet* in short. The state-of-the art DNN including ResNets [17], DenseNet [30], SqueezeNet [31] and EfficientNet [19,20] addressed models' size, accuracy and inference time balance, but the outcomes were not operationalized in real-time field applications, because of resource limitations at the edge and the need for continuous architectural adaptation. Methods that only consider improvements in inference time, assume that model training or tuning would be done once at the beginning or infrequently. However, modern distributed [32] or decentralized scenarios teach us that frequent model updates may be necessary. Our work quantifies the evaluation and ranking of DNN's for this purpose.

DeepCPU [33] developed by Microsoft also focuses on operational models, but only addresses model serving time reduction. They denote, users can tolerate long training times since this step is offline but improving serving time is crucial since this makes the biggest difference between *shippable vs. non-shippable* model. Models need to conform to Service-Level Agreements (SLA) in real-time and thus must make fast and accurate predictions. DeepCPU proposes cache-aware partitioning to optimize L2-L3 data movements and weight-centric streamlining. Specifically, they demonstrate 10x-60x speedups in serving time of different NLP models. There are other researchers that address model performance issues at the hardware level. HardNet [34] aimed to reduce the traffic to DRAM by reducing the number of accesses to feature maps in CNN. This operation reduced the number of floating-point operations (flops) and inference latency.

In this paper, we evaluated the performance benefits and potential hardware dependencies of quantization. While our work is orthogonal to scalar and vector-based quantization [35], it is still useful to summarize related work in this group. Fixed scalar quantization of weights stored in floating-point variables can reduce performance as the amount of information is reduced. Researchers analyzed resilience of DNN under quantization [36] and effects of retraining models to alleviate this problem. Zhang *et al.* [15] developed a strategy called Learned Quantization Networks (LQ-Nets) that can change the fixed point and adapt quantization level to balance the tradeoff between size and accuracy. Han, *et al.* [16] reduced model sizes further via deep compression while also trying to address power issues. They designed a three-stage pipeline of model pruning, trained quantization, and Huffman coding to reduce the storage requirement of NN without affecting their accuracy.

Nath, *et al.* [37] claim that most of these methods suffer from two strict requirements, which hinder their operational performance: (1) they require special hardware to be effective or (2) they have to modify their model structures and/or weights via pruning [38] and retraining which is quite costly to handle during operation. In their paper titled "better together" [37], they proposed "adjoint networks" where a large DNN continuously acts as a teacher to a smaller DNN. Their approach is orthogonal to quantization and neural architectural search [32] methods. A similar approach by Shen, *et al.* [39] called MealV2 uses an ensemble of teachers for "knowledge distillation" coupled with a good initialization of the student model. Our model-to-model layer transfer method was inspired by these approaches, but we wanted to further minimize the cost of retraining by quick layer transfers and stitching before retraining instead of weight initialization. We could resemble our new approach to creation of a "Frankenstein" model, which borrows layers of the architecture model different related models.

Other orthogonal but related work include model generalization issues [40,41] and DNN performance surveys [42,43]. Recht, *et al.* [40] performed a longitudinal study for model generalization or robustness using ImageNet classifiers using CIFAR10 and ImageNet datasets. After creating a new labelled dataset *called CIFAR 10.1*, they found that (1) there was a significant performance drop in all models developed during 2013-2018 (a very active 5 years for ML research), (2) drops were due to models' inability to generalize to slightly harder images and not due to model adaptability to new data (*i.e.* overfitting to old datasets). Zhang, *et al.* proposed adversarial auto-augmentation [41] which is a data augmentation technique to help target NN learn hard features and improve model generalization

Li, *et al*. [43] gave a comprehensive survey of deep learning (DL) compilers. They listed properties of DL frameworks such as TensorFlow, PyTorch, Keras, *etc.* and DL accelerator hardware (by Google, Intel, Amazon). It is a useful survey for understanding general OpML issues.

6. CONCLUSION AND FUTURE WORK

We described a framework for multi-dimensional tradeoff assessment of DNN model performances based on accuracy, time, and size. Comparison of modern quantization and weight sharing (clustering) techniques revealed that quantization can provide size savings without loss of accuracy, yet it needs GPU support for mixed-precision float operations for achieving acceptable train & test times. If this can be provided, then its operational efficiency and viability is high. Yet, these high-end GPUs are currently not feasible for use in edge devices, especially at large-scale. NN weight sharing via clustering can save sizes by up to 60%, but its accuracy is sensitive to number of clusters used. While higher cluster counts give better accuracy, they eventually lose benefit in size and time dimensions. Finally, direct layer transfers among models requires careful selection. Otherwise, their accuracy can be lower than the original models.

In the future, we plan to extend our sensitivity analysis to different models and model parameters. Specifically, we are currently investigating the effect of decentralized, non-IID and unbalanced datasets in distributed federated learning settings [44,45,46]. Our work proposes a 3D evaluation scheme for these and other comparable techniques. We hope that DNN, AI, ML, OpML researchers can repeat these assessments over different DNN models and datasets.

REFERENCES

[1] F. Fabio, G. Lami, A. M. Costanza. "Deep learning in automotive software." *IEEE Software* 34(3), 56-63, 2017.

[2] J. Villalba-Diez, D. Schmidt, R. Gevers, J. Ordieres-Meré, M. Buchwitz, W. Wellbrock, "Deep learning for industrial computer vision quality control in the printing Industry 4.0". *Sensors*, 19(18), 3987, 2019.

[3] Z. Hu, Y.Zhao, M. Khushi. "A survey of forex and stock price prediction using deep learning." *Applied System Innovation* 4(1), 9, 2021.

[4] J. Kim, Y. Shin, E. Choi. "An intrusion detection model based on a convolutional neural network." *Journal of Multimedia Information System* 6(4), 165-172, 2019.

[5] Deng, L., Liu, Y. (Eds.). **Deep learning in natural language processing**. Springer, 2018.

[6] Bashar, A. "Survey on evolving deep learning neural network architectures". *Journal of Artificial Intelligence*, 1(02), 73-82, 2019.

[7] A. Collette, **Python and HDF5,** O'Reilly Media, Inc., November ISBN: 9781449367831, 2013.

[8] Internet: Open Neural Network Exchange (ONNX), The open standard for machine learning interoperability, https://onnx.ai, 24.10.202.

[9] Internet: Model Zoo, Open source deep learning code and pretrained models. https://modelzoo.co, 24.10.202.

[10] Internet: Google Colaboratory, https://colab.research.google.com/, 24.10.202

[11] M. Abadi, P. Barham, J.Chen, Z. Chen, A. Davis, J. Dean, M. Devin et al. "Tensorflow: A system for large-scale machine learning." **In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI),** 265-283. 2016.

[12] Yu, R., Li, P. "Toward resource-efficient federated learning in mobile edge computing". *IEEE Network*, 35(1), 148-155, 2021.

[13] M. , Paulius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, et al. "Mixed precision training." *arXiv preprint* arXiv:1710.03740 (2017).

[14] D. Lin, T. Sachin Talathi, A. Sreekanth "Fixed point quantization of deep convolutional networks." **In International Conference on Machine Learning, PMLR,** 2849-2858, 2016.

[15] D. Zhang, J. Yang, D. Ye,, G. Hua, "LQ-Nets: Learned quantization for highly accurate and compact deep neural networks". **In Proceedings of the European Conference on Computer Vision (ECCV)** 365-382, 2018.

[16] S. Han, H. Mao, W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding". *arXiv preprint* arXiv:1510.00149, 2015.

[17] K. He, X. Zhang, S. Ren, J. Sun, "Deep residual learning for image recognition". **In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR),** 770-778, 2016.

[18] Simonyan, K., Zisserman, A. "Very deep convolutional networks for large-scale image recognition". *arXiv preprint* arXiv:1409.1556, 2014.

[19] Internet: Keras Applications, https://keras.io/api/applications, 24.10.202

[20] M. Tan, Q. Le. "EfficientNet: Rethinking model scaling for convolutional neural networks." **In International Conference on Machine Learning (ICML),** 6105-6114. PMLR, 2019.

[21] Tan, M., Le, Q. "EfficientNetv2: Smaller models and faster training". **In International Conference on Machine Learning**, 10096-10106, PMLR, 2021

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. "Scikit-learn: Machine learning in Python." *The Journal of Machine Learning Research* V.12, 2825-2830, 2011.

[23] He, Y., Shen, Z., Cui, P. "Towards non-IID image classification: A dataset and baselines". *Pattern Recognition*, 110, 107383, 2021.

[24] M.B. Çamlı, I. Ari, "Sensitivity Analysis of Federated Learning over Decentralized Data and Communication Rounds", **7. Ulusal Yüksek Basarimli Hesaplama Konferansı** (BAŞARIM), no.14, 2022

[25] Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H. P. "Pruning filters for efficient ConvNets". *arXiv preprint* arXiv:1608.08710, 2016.

[26] Glorot, X., Bengio, Y. "Understanding the difficulty of training deep feedforward neural networks". **In Proceedings of the 13th international conference on artificial intelligence and statistics**, 249-256, JMLR Workshop and Conference Proceedings, 2010.

[27] Luo, J., Wu, X., Luo, Y., Huang, A., Huang, Y., Liu, Y., Yang, Q. "Real-world image datasets for federated learning". *arXiv preprint* arXiv:1910.11089, 2019

[28] Zhu, H., Xu, J., Liu, S., Jin, Y. "Federated Learning on Non-IID Data: A Survey". *arXiv preprint* arXiv:2106.06843, 2021.

[29] Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., Chandra, V. "Federated learning with non-IID data".*arXiv preprint* arXiv:1806.00582, 2018.

[30] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, K. Keutzer, "DenseNet: Implementing efficient convnet descriptor pyramids" *arXiv preprint* arXiv:1404.1869, 2014.

[31] F. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, K. Keutzer."SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size." *arXiv preprint* arXiv:1602.07360, 2016.

[32] H.R. Roth , D. Yang, W. Li, A. Myronenko, W. Zhu, Z. Xu, X. Wang, D. Xu, "Technique to perform neural network architecture search with federated learning", WO/2021-247338A1, WIPO Patent, 2021.

[33] M. Zhang, S. Rajbhandari, W. Wang, Y. He, "DeepCPU: Serving RNN-based deep learning models 10x faster". **In 2018 USENIX Annual Technical Conference (ATC),** 951-965, 2018.

[34] P. Chao, C. Kao, Y. Ruan, C. Huang, Y. Lin. "Hardnet: A low memory traffic network" **In Proceedings of the IEEE/CVF International Conference on Computer Vision,** 3552-3561. 2019.

[35] Sung, W., Shin, S., Hwang, K. "Resiliency of deep neural networks under quantization". *arXiv preprint* arXiv:1511.06488, 2015.

[36] Gong, Y., Liu, L., Yang, M., Bourdev, L. "Compressing deep convolutional networks using vector quantization". *arXiv preprint* arXiv:1412.6115, 2014.

[37] Nath, U., Kushagra, S. “Better Together: Resnet-50 accuracy with 13x fewer parameters and at 3x speed”. *arXiv preprint* arXiv:2006.05624, 2020.

[38] Ma, X., Yuan, G., Lin, S., Li, Z., Sun, H., Wang, Y. “ResNet can be pruned 60×: Introducing network purification and unused path removal (p-rm) after weight pruning”. **In 2019 IEEE/ACM International Symposium on Nanoscale Architectures,** 1-2, IEEE, 2019.

[39] Shen, Z., Savvides, M. “Meal v2: Boosting vanilla Resnet-50 to 80%+ Top-1 accuracy on Imagenet without tricks”. *arXiv preprint* arXiv:2009.08453, 2020.

[40] B. Recht, R. Roelofs, L. Schmidt, L., V. Shankar, “Do Imagenet classifiers generalize to Imagenet?” **In International Conference on Machine Learning,** 5389-5400, PMLR, 2019.

[41] Zhang, X., Wang, Q., Zhang, J., Zhong, Z. “Adversarial autoaugment”. arXiv preprint arXiv:1912.11188, 2019.

[42] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, C. Liu, “A survey on deep transfer learning”. **In International conference on artificial neural networks** 270-279, Springer, 2018.

[43] M. Li, Y. Liu, X. Liu, Q. Sun, X. You, H. Yang, Z. Luan, L. Gan, G. Yang, D. Qian. "The deep learning compiler: A comprehensive survey." *IEEE Transactions on Parallel and Distributed Systems*, Vol. 32, No. 3, 708-727, 2020.

[44] McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B. A. “Communication-efficient learning of deep networks from decentralized data”. *In Artificial intelligence and statistics*, 1273-1282, PMLR, 2017.

[45] T. Li, A. K. Sahu, A. Talwalkar, V. Smith, V. “Federated learning: Challenges, methods, and future directions”. *IEEE Signal Processing Magazine*, 37(3), 50-60, 2020.

[46] Maraş, A. Erol, Ç. “Emerging Trends in Classification with Imbalanced Datasets: A Bibliometric Analysis of Progression”. *Bilişim Teknolojileri Dergisi*, 15(3), 275-288, 2022.