Characterizing the Cost-Accuracy Performance of Cloud Applications

Sunimal Rathnayake
National University of Singapore
Singapore
sunimalr@comp.nus.edu.sg

Lavanya Ramapantulu
Nanyang Technological University
Singapore
r.lavanya@ntu.edu.sg

Yong Meng Teo
National University of Singapore
Singapore
teoym@comp.nus.edu.sg

ABSTRACT

Emergence of applications that produce results with different accuracy allows cloud consumers to leverage the advantages of elastic cloud resources and pay-per-use pricing model. However, the trade-off between cost, accuracy and execution time of cloud applications has not been well studied due to multiple challenges. A key challenge faced by a cloud consumer is tuning the application and determining cloud resource configuration that achieves the desired application accuracy among the configuration space.

This paper proposes an approach to improve the cost-accuracy performance of cloud applications for a given cost and accuracy. To illustrate our approach, we use two popular convolution neural networks' (CNN) inference as examples with pruning as a tuning tool for changing the accuracy, and yield several insights. Firstly, we show the existence of multiple degrees of pruning as "sweet-spots", where inference time and cost can be reduced without losing accuracy. Combining such sweet-spots can halve inference cost and time with one-tenth reduction in accuracy for Caffenet CNN. Secondly, we show that in the large resource configuration space, these "sweet-spots" form the cost-accuracy and time-accuracy Pareto-frontiers whereby a Pareto-optimal configuration can reduce cost and execution time by 55% and 50% respectively for achieving the highest possible inference accuracy. Lastly, to quantify the accuracy performance of cloud applications, we introduce Time Accuracy Ratio (TAR) and Cost Accuracy Ratio (CAR) metrics. We show that using TAR and CAR reduces the time complexity in determining cloud resource configurations from exponential to polynomial-time.

ACM Reference Format:
Sunimal Rathnayake, Lavanya Ramapantulu, and Yong Meng Teo. 2020. Characterizing the Cost-Accuracy Performance of Cloud Applications. In *49th International Conference on Parallel Processing - ICPP : Workshops (ICPP Workshops '20), August 17–20, 2020, Edmonton, AB, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3409390.3409409

1 INTRODUCTION

Competition among today's cloud providers has resulted in vendors offering a large number of cloud services and resource types with pay-per-use charging spanning a multitude of consumer needs. The cost of executing an application remains a key consideration for a cloud consumer in selecting cloud resources. Cloud consumers have the opportunity to minimize the cost and execution time by selecting the optimal cloud configuration for an application. Among these applications, there are applications that produce results with a notion of accuracy or a probabilistic output such as the machine learning applications. With the advent of such applications that produce probabilistic results with different accuracy, there is a new opportunity for cloud consumers to minimize cloud cost through cost-accuracy trade-off. Application and resource scaling on cloud is a well explored topic where the focus has mainly been on resource elasticity. Inspired by Amdahl's [2] and Gustafson's [9] laws on parallel processing, the cloud research community has extended the fixed-workload and fixed-time scaling on the cloud. These work either focus on the impact of cloud resource configuration [8, 15, 21–23, 25, 28] on cost and time, or the impact of scaling problem size on cost and time [26]. However, there is a lack of work on understanding the impact of changing accuracy on both the cost and execution time of the cloud application. This research question is non-trivial due to the large resource configuration presented by the wide spectrum of heterogeneous cloud resources and the different degrees of changing application accuracy. We address this challenge by investigating and quantifying the impact of accuracy with respect to both cost and time.

Big data explosion and advancements in large applications such as real-time image classification, natural language processing, among others, pose new challenges for enterprises. There is a growing demand for compute power and parallel processing with the need to deliver results in real time or within expected time deadlines for Internet-based applications. For example, the need for image detection and filtering processing in social media platforms such as Facebook social media network saw as many as 350 million photo uploads per day in Jan 2019 [4]. Increasingly, before these photos are published they go through a filtering process to determine whether they comply with the rules and regulations [6]. This filtering process has to be completed in near real-time speed for photos to appear on their profiles almost immediately. Today, Convolution Neural Networks (CNNs) are widely used in such applications. As CNNs perform a lot of computations, mainly convolutions, specialized hardware such as Graphics Processing Units (GPUs), Tensor Processing Units and Field Programmable Gate Arrays (FPGAs) are used to improve real-time performance. Results from CNNs are associated with a percentage accuracy metric. Thus the expectation is to produce a "close enough" result [10]. For instance, in image filtering on social media, it would be good enough to say that a given image is violating the rules with a 75% probability, so that the image could be forwarded for manual review. Given the important

ICPP Workshops '20, August 17–20, 2020, Edmonton, AB, Canada

ACM ISBN 978-1-4503-8868-9/20/08…$15.00
https://doi.org/10.1145/3409390.3409409

role CNNs play in today's computing landscape and their suitability as applications that produce results with varying accuracy, we selected CNN as our example application in this paper.

To investigate the trade-off between the cost and accuracy of applications on cloud, we propose a measurement driven approach and use image classification CNN applications, Caffenet and Googlenet as example cloud applications. To vary the accuracy of CNN inference, we change the parameters in CNN layers thereby resulting in sparse layers. This technique is called pruning. The combination of layers pruned by different degrees results in versions of the CNNs that produce classification results with different accuracy. While pruning has been well-studied with respect to CNN algorithms, the impact of tuning such applications on different cloud configurations is non-trivial. Using a measurement-based analytical model we determine cost-accuracy and time-accuracy Pareto-optimal cloud resource configurations. Measurement results are obtained by running 50,000 unique inferences on Caffenet and Googlenet CNN image classification models that were trained with 1.2 million images on Amazon EC2 cloud instances with GPU for parallel processing.

This paper makes the following contributions:

(1) We present a measurement-driven model for investigating the trade-offs between time and accuracy, and, cost and accuracy, and show the existence of "sweet-spot" regions where the time and cost could be reduced with no reduction in application accuracy.
(2) We show that multi-layer pruning is effective in reducing the inference time with minimal accuracy drop. For example, for Caffenet CNN on Amazon EC2, the inference time could be halved with just one-tenth drop in accuracy.
(3) We show the existence of cost-accuracy and time-accuracy Pareto-optimal configurations spanning considerably large time and cost range, and opportunity to reduce cost and time. Selecting the right degree of pruning and resource configuration reduces execution time by up to 50% and cost by up to 55% for obtaining highest possible inference accuracy for Caffenet.
(4) We show that quantifying cost-accuracy and time-accuracy performance measured in TAR and CAR is important in selecting efficient application and resource configurations, and demonstrate their usage as heuristics in a polynomial-time cloud configuration determination algorithm.

The paper is organized as follows. Section 2 discusses related work. Section 3 presents our approach followed by Section 4 where we present results from our approach. Section 5 concludes the paper.

2 RELATED WORK

We discuss the related work under two categories; (i) accuracy of applications where we focus on scaling accuracy of applications, and (ii) impact of scaling on cloud cost.

2.1 Accuracy-Performance of Applications

Changing the accuracy of applications in resource constrained environments is a well-explored resource topic. Especially, in the embedded computing domain where applications are required to deliver results under tight resource constraints (ie. power, storage, memory), trading off the accuracy of applications is a popular approach. Such an approach includes changing the precision of variable representation, lossy computations and compression, data and task sampling, among others [24]. However, outside embedded computing domain, changing accuracy of applications have received comparatively much less attention. The primary reason is that the traditional applications operated in on-premise resources with accesses to large processing capacity, memory, and storage. With the advent of machine learning applications, specially CNNs that produce imprecise results associated with an accuracy percentage, trading off accuracy has become relevant to applications outside embedded systems as well. Li et al. [17] and Luo et al [20] proposes pruning to reduce the number of parameters, resulting in sparse matrices. In this paper, we use pruning to change inference accuracy. Quantization [7, 32] is used to change the length of variables that hold CNN parameters. For example, a parameter data type which is usually represented by 64-bits will be changed to be represented by 32-bits. This has a direct impact on the memory usage of the application. Quantization improves the execution time if there is hardware support for higher speed computations with shorter bit representation. Like Quantization, weight sharing [1] is a technique to cluster parameters in CNNs together based on a "closeness" measure. Multiple parameters that have values close to each other would be reduced to one parameter. This also has a direct impact on the memory and storage usage of the CNN rather than the execution time.

In the context of the cloud where the major concern is the cost, users are more interested in reducing the execution time as it directly translates into cost savings due to pay-per-use charging. Unlike resource constrained environments such as embedded systems, the cloud has ample storage and memory capacity for a relatively inexpensive price. Thus, in this work, we select pruning as the technique for changing the accuracy of CNNs. When it comes to performance of CNNs, the major focus has been on the training phase of the application [14, 18, 29]. Li et al. [18] present a measurement based analysis on training CNNs. They compare different implementations of popular CNN models on on-premise GPUs and investigate performance bottlenecks. In contrast, we focus on CNN inference and conduct our study on cloud GPU instances. It is important to note that existing studies on CNNs have been focusing on the time-performance of CNN hosted on on-premise systems. To the best of our knowledge, there has been no published work that studies the trade-off between cost and CNN inference accuracy on the cloud.

2.2 Application Scaling on Cloud

Unlike resource scaling, application scaling on the cloud has not received much attention. CELIA [25] investigates the scaling of an application under a cost budget and time deadline constraint on the cloud. They propose a measurement-driven analytical modeling approach for determining Pareto optimal cloud resource configurations. Rathnayake et al. [26] focus on the impact of scaling problem size of applications on cloud cost using an analytical model and an optimization algorithm. In comparison to these works, this paper focuses on scaling the accuracy of applications and study its impact on cloud cost and time. Han [11] proposes a framework for developing scalable algorithms called elastic algorithms where they define

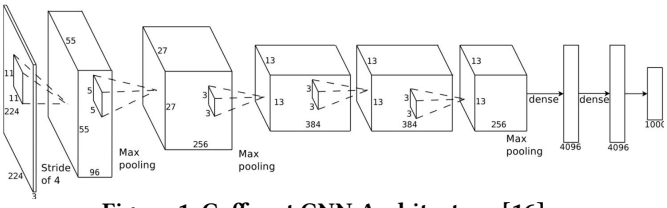

Figure 1: Caffenet CNN Architecture [16]

the accuracy of results with respect to a cost or time investment. Using this method, they generate algorithms that produce results with different accuracy values to match the cost and time investment. In comparison, instead of generating an algorithm we change the accuracy of an already existing application (CNN model) to trade off accuracy for execution time and cost. He et al. [12] proposes scaling of cloud-hosted video streaming applications by varying the quality of the video based on network statistics such as bandwidth and latency. In contrast, we focus on compute intensive CNN applications to study the tradeoff.

In summary, trading-off accuracy for performance is expanding from embedded systems into cloud computing with the advent of machine learning applications such as CNNs. However, there has been limited research to investigate the trade-offs and opportunities present on the cloud for such applications. This paper attempts to bridge this gap and present insights to the cloud consumer on the accuracy-cost and accuracy-time trade-offs on the cloud.

3 APPROACH

In this section, we present our approach which consists of (i) application characterization, (ii) a measurement-driven analysis and (iii) a measurement-driven analytical modeling approach for determining cloud resource configurations for a given accuracy within time and cost deadline constraints, as illustrated in Figure 2.

3.1 Overview

Given an application which produces results of different accuracy for different application configurations, and a set of cloud resources, the objective of our approach is to understand the effect of application accuracy on execution time and cost on cloud. The approach consists of three main stages. Firstly, to characterize the application we conduct a baseline execution and determine various experiment parameters. Secondly, we take time measurements for different application accuracy. Thirdly, we input these measurements into our analytical models for determining execution time and cost for different resource configurations. Cloud resource configurations are filtered using a Pareto optimization filter to determine time and cost Pareto optimal cloud configurations. Finally, with the measurements and analytical model predictions, we derive insights on the time-accuracy and cost-accuracy trade-offs in executing applications on cloud. Since we selected CNN as the representative application, we describe our approach in detail in the context of CNNs.

3.2 Application Characterization

To understand the CNN and to fix experiment parameters such as the number of parallel inferences, we characterize the CNN on cloud. Firstly, to understand the contribution of each CNN layer towards inference time, we analyze the execution time distribution for each layer. Secondly, to determine whether there is still an opportunity to improve inference time, we evaluate the time taken for a single inference and how it changes with pruning. Thirdly, to ensure full utilization of the compute resources, we determine the maximum number parallel inferences on a given cloud resource.

Table 1: Caffenet Layers

Layer	Size	Number of Filters	Filter Size
input	224 x 224 x3	-	-
conv 1	55 x 55 x 96	96	11 x 11 x 3
conv 2	27 x 27 x 256	256	5 x 5 x 48
conv 3	13 x 13 x 384	384	3 x 3 x 256
conv 4	13 x 13 x 384	384	3 x 3 x 192
conv 5	13 x 13 x 256	256	3 x 3 x 192
fc1	4096	-	-
fc2	4096	-	-
fc3	1000	-	-

3.2.1 Pruning of CNNs. CNN layers are stored in the memory as matrices. The CNN training process adjusts the values of the elements of these matrices in order to tune the CNN for the highest accuracy resulting in dense matrices. Pruning removes (changes to zero) selected elements in these matrices. There are different algorithms to determine the parameters (elements) that need to be pruned. Li et al. [17] remove entire regions of convolution layers instead of individual parameters based on L1 norm. Anvar et al. [3] present a similar approach to [17], but with a more complex scoring algorithm to rank parameters. Huang et al. [13] from Nvidia propose modeling the pruning problem as a combinatorial optimization problem with a cost function. Their objective is to determine the subset of parameters that minimizes the cost function when pruned. When CNN layers are pruned, the pruning algorithm selectively changes the matrices' elements to zero, resulting in sparse matrices. For simplicity and implementation convenience, we use the pruning tools developed by Li et al. [17] in this paper. Resulting sparse matrices can be efficiently processed with special sparse matrix computational libraries. In this work, we use an extended version of Caffe framework [31] for efficient sparse matrix computation.

3.2.2 Accuracy of CNN Inference. The accuracy of CNN inference refers to the percentage of correct predictions that the CNN does. For example, for an image classification CNN, the accuracy is the ratio between the number of images the CNN determined the correct label and the total number of images given as input for classifying. Keeping to the standards set by the machine learning research community[19, 27], in this paper, we use two widely used accuracy metrics as follows.

- Top 1 accuracy: The percentage of the number of times that the class with the highest probability is the expected class for the given input.
- Top 5 accuracy: The percentage of the number of times that the expected class for the given input is one of the 5 classes with the highest probability.

3.3 Measurements

For our analysis, we use measurements to determine both execution time and cost for CNN inference on a cloud resource instance for different accuracy. Firstly, for measuring the execution time, we

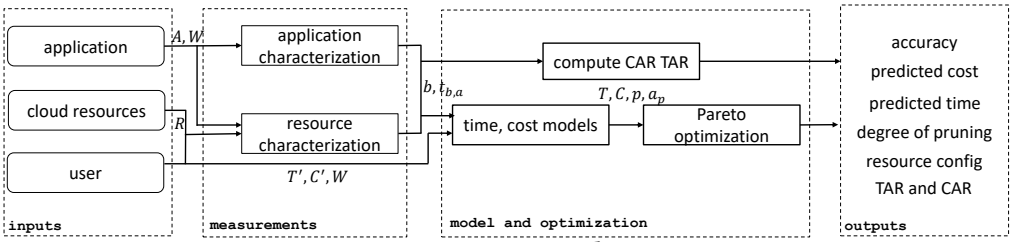

Figure 2: Approach

Symbol	Description
CNN Application and Data	
A	a CNN application
P	set of A pruned with different degrees of pruning
p	a degree of pruning in P
a_p	accuracy of p
W	number of images for inference
n	number of batches
Cloud Resources	
G	set of all cloud resources
R	a cloud resource configuration of G
i	a cloud resource type in R
v_i	number of GPUs in i
c_1	cost per unit time for i
b_i	max parallel inference (batch size) of i
Model	
C'	cost budget for inference of W
T'	time deadline for inference of W
C	total cost for inference of W
T	total time for inference of W
t_{b,a_p}	time for inferring b with a_p

Table 2: Symbols Used

execute CNN inference and record the time. To change the inference time, convolution layers of the CNN are pruned in different degrees. Secondly, to compute the cost, we retrieve the cost per unit time from the cloud provider and compute the inference cost by multiplying the unit cost of the cloud resources used by inference time. Thirdly, to compute the Top1 and Top5 accuracy, we count the number of accurate inferences and compute the metrics as defined in Section 3.2.2. Finally, we compute the TAR and CAR for each application configuration using the formula shown in Section 3.5. At the end of the measurement phase, we output a list of degrees of pruning with their inference time, cost, TAR, and CAR. To minimize the measurement error, we run each experiment three times and record the minimum time measurement.

3.4 Time and Cost Models

This section presents the derivation of our analytical models for determining execution cost and time on cloud resources for a given CNN with different degrees of pruning.

The CNN inference cost on cloud depends on the total inference time and the cost per unit time of the cloud resource configuration.

$$C = T \sum_{i=0}^{|R|-1} c_i \tag{1}$$

where T is the inference time, and c_i is the cost per unit time for cloud resource i. T is ratio between the number of batches to be inferred and the time for inferring one batch for a given accuracy.

$$T = max\left(\frac{n}{t_{b,a}}\right) \tag{2}$$

where n is the number of batches and $t_{b,a}$ denotes the time for a single batch inference for a batch size b and accuracy a. The inference accuracy of a CNN depends on its degree of pruning as described above in section 3.2.1. n relies on the maximum number of parallel inferences possible on the GPU device (the batch size).

$$n = \frac{W}{b} \tag{3}$$

where W is the total number of images to be inferred.

Among cloud resources in a given cloud resource configuration R, inference images are distributed as follows. If W_i denotes the number of inference images for resource i,

$$W_i = \frac{W}{\sum_{j=1}^{|R|} v_j} v_i \tag{4}$$

Configurations predicted by the models are sent through Pareto-optimization to filter cost-accuracy and time-accuracy Pareto optimal configurations that satisfy time deadline T' and cost budget C'.

3.5 Time Accuracy Ratio and Cost Accuracy Ratio

To quantify the trade-offs between time and accuracy, and, cost and accuracy, we introduce two metrics (i) Time Accuracy Ratio (TAR) and (ii) Cost Accuracy Ratio (CAR).
TAR is defined as

$$TAR = \frac{t}{a}$$

where t is the inference time and a denotes inference accuracy.
Similarly, CAR is defined as

$$CAR = \frac{c}{a}$$

where c is the cost incurred on cloud to achieve inference accuracy a.
TAR and CAR represent the time and cost respectively, for achieving a single unit of accuracy where $t, c \in (0, \inf)$ and $a \in [0, 1]$. Higher TAR value means that the time required to achieve a unit of accuracy is higher and higher CAR means that it incurs a higher amount of cost to achieve a unit of accuracy. Hence, for both these metrics a lower value indicates better performance when comparing system configurations.

Table 3: Amazon EC2 Cloud Resource Types

Instance Type	vCPUs	GPUs	Mem (GB)	GPU Mem (GB)	Price ($/hr)	GPU Type
p2.xlarge	4	1	61	12	0.9	NVIDIA K80
p2.8xlarge	32	8	488	96	7.2	
p2.16xlarge	64	16	732	19	14.4	
g3.4xlarge	16	1	122	8	1.14	NVIDIA M60
g3.8xlarge	32	2	244	16	2.28	
g3.16xlarge	64	4	488	32	4.56	

4 EVALUATION

In this section, we present the evaluation of our approach. Firstly, we present the experiment setup including application and cloud resources. Secondly, we show the impact of changing accuracy on the execution time on cloud. Thirdly, we present the impact of changing accuracy on execution cost. Finally, we present two metrics for quantifying the accuracy performance with respect to time and cost and discuss their usage.

4.1 Experiment Setup

4.1.1 Application. We selected two widely used image classification CNNs, Caffenet and Googlenet[30] implemented on caffe machine learning framework as representative applications. Caffenet is a Caffe implementation of Alexnet [16] CNN. As shown in Figure 1, Caffenet consists of five convolution layers and three fully connected layers while Googlenet is much deeper with 56 convolution layers (two main convolution layers and nine "inception" layers each containing six convolution layers). The architecture of Googlenet is not shown due to space constraints. Since we are interested only in inference, we obtained Caffenet and Googlenet trained with a subset of ImageNet [5] dataset containing about 1.2 million. This training dataset contains images from 1000 categories with approximately 1000 images from each category. A breakdown of the properties of each layer is shown in Table 1. `conv` is used to denote a convolution layer while `fc` denotes a fully-connected layer. The convolution layers are non-uniform in size, number of filters and the filter size. Convolution layer 1 is the largest in terms of image size and, convolution layer 2 comes next. The last three convolution layers are approximately equal in size. When it comes the fully-connected layers, unlike convolution layers which are three dimensional, these are vectors. Last fully-connected layer, `fc3`, has 1000 neurons that relate to the number of possible outputs. In contrast to the Caffenet, despite being a deeper CNN, Googlenet has only 4 million parameters in its layers. We use the same RGB input image size of 224x224 pixels for both CNNs. For inference, we use 50000 images from ImageNet dataset which were not included in training.

4.1.2 Cloud Resources. As CNN processing involves a large number of matrix operations, we evaluate our approach on GPU instances from Amazon EC2 cloud. As shown in Table 3, we selected six cloud resource types with GPUs from Amazon EC2 Oregon region. Both `p2` and `g3` resource instances are powered by Intel Xeon E5-2686 v4 CPUs with 2.3GHz base frequency. `p2` instances have NVIDIA k80 GPUs with 2,496 parallel processing cores while `g3` instances are equipped with NVIDIA M60 GPUs with 2048 parallel processing cores. Although the specifications states that GPUs are attached, they are virtual GPUs. Moreover, when charging for resource usage, the hourly price mentioned in the specification is pro-rated to the nearest second.

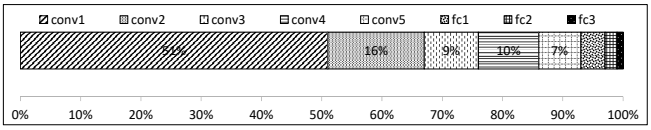

Figure 3: Caffenet Execution Time Distribution of CNN Layers

4.2 Application Characterization

In application characterization, we focus on three parts. Firstly, we determine the execution time distribution across CNN layers to find the most impactful layers. Secondly, we investigate to which extent CNN inference time can be improved with respect to a single inference. Lastly, we determine the maximum number of parallel inferences (batch size) on the GPU device for maximum utilization.

4.2.1 Execution Time Distribution of CNN Layers. To apply pruning, we first determine the layers that have a large execution time by individually measuring the time taken by each layer on caffe framework during inference. Figure 3 shows the distribution of execution time for different layers for Caffenet. We observe that the convolution layers contribute more to the execution time compared to other layers and the contribution of each convolution layer is directly proportional to the values of the image parameters in the convolution layers. As shown in Table 1, `conv1` has the largest image size (Figure 1) and `conv2` comes second. `conv3`, `conv5`, and `conv5` have an approximately similar number of parameters. Following a similar pattern, the execution time contribution is 51% for `conv1`, and 16% for `conv2` followed by 9%, 10% and 7% for the last three convolution layers respectively. The contribution of fully-connected layers is very small compared to convolution layers. Although fully connected layers are dense, as they are not involved in the expensive convolution operation, their contribution to the execution time is smaller.

4.2.2 Has Inference Performance Hit the Wall? To determine whether there is still scope for improvement in the inference time, we apply pruning for a single inference and study the impact on execution time. While varying the prune percentage from 0% to 90% uniformly across all convolution layers, we recorded the execution time for a single inference. The resulting plot is shown in Figure 4. We ob-

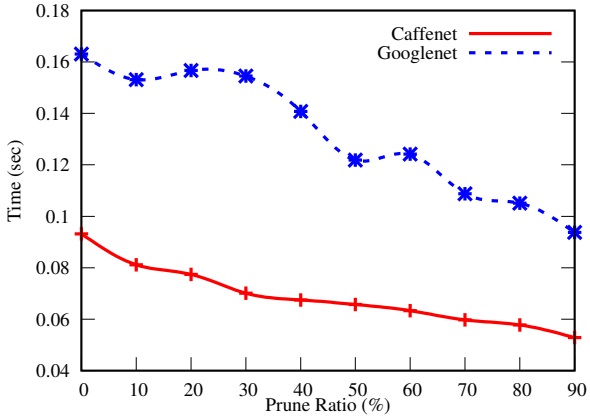

Figure 4: Time for a Single Inference

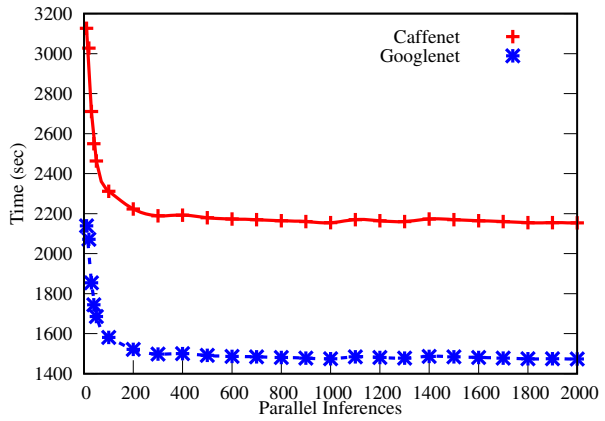

Figure 5: Parallel Inference on a GPU

serve that the inference time decreases with pruning percentage for both CNNs. From 0% to 90% pruning, the execution time of a single inference drops by about half from 0.09s to 0.05 seconds and about one thirds from 0.16s t0 0.1 sec for Googlenet. Thus, we can safely assume that there is still scope for performance improvement in CNN inference.

4.2.3 Parallel Inference on GPU. A GPU consists of hundreds of compute cores that operate in parallel. To get the maximum power of GPUs, processing needs to be parallelized such that the GPU resources are fully utilized. A simple way of increasing GPU utilization in inference is by increasing the number of inferences that run in parallel (also referred to as batch size). Since each GPU has its own characteristics such as the number of cores and memory, it is important to determine the amount of parallelism required to fully utilize the GPU. Although in an on-premise system, a GPU specification would be good enough to estimate this, cloud GPU instance capabilities may vary from the specification due to virtualization, multi-tenancy and other restrictions on the cloud. To estimate this accurately, we experimentally determined the number of parallel inferences required for full utilization of GPU. The results on Amazon EC2 p2.xlarge instance with Nvidia k80 is shown in Figure 5. We observe that the GPU saturates around 300 parallel inferences, thus it is safe to assume that GPU cloud instances will be sufficiently utilized when the number of parallel inferences is 300 or above.

4.3 Effect of Accuracy on Inference Time

To understand the effect of accuracy on CNN inference time, we investigate the change in execution time while changing accuracy. As we use pruning to control accuracy, we prune CNN layers with different degrees and measure the inference time and accuracy. As shown in Figure 3, since convolution layers account for more than 90% of the total inference time, we focus only on the convolution layers. Firstly, we discuss the effect of accuracy on execution time for a CNN model hosted on a single standalone resource with a single GPU. We focus on the impact of pruning individual CNN layers and having dependency among CNN layers. Secondly, we extend to CNN inference hosted on a cloud resource configuration with multiple cloud resources with many GPUs and discuss the impact of accuracy on execution time.

4.3.1 Pruning Single Layer Only. Figures 6 and 7 illustrate the execution time, Top 1 and Top 5 inference accuracy for different pruning ratios for Caffenet and Googlenet, respectively. The subfigures correspond to pruning of each convolution layer. Figure 7 shows only six selected convolution layers of Googlenet from different levels of the CNN due to space constraints. We observe a near-linear decrease in inference time with pruning for all convolution layers of Caffenet whereas Googlenet has decrease in time albeit some fluctuations. The largest reduction in time for Caffenet is when pruning `conv2` layer with one-fourth drop from 19 mins to 14 mins whereas the smallest reduction is for `conv1` layer where the drop is about one-eighth from 19 mins to 16.6 mins. Therefore, by pruning only a single layer, a significant reduction of inference time up to 25% can be achieved for Caffenet. Similarly, `conv2-3x3` of Googlenet has the strongest impact among the selected six layers where the time is reduced by about 30% from 13 mins to 9 mins.

Observation 1: There exist "sweet-spot" regions where inference time reduces with an insignificant drop in accuracy.

Despite the reduction of inference time, we observe that Top 1 and Top 5 accuracy curves in Figure 6 remain unchanged for a range of pruning ratios starting from 0%. For example, as illustrated in Figure 1 (c), in Caffenet both Top 1 and Top 5 accuracy remain almost unchanged until the pruning ratio reaches 50%. After 50%, both curves demonstrate a gradual drop. However, it is evident that for the same range where accuracy remains unchanged, the inference time demonstrates a gradual decrease. A similar pattern could be observed in all convolution layers. For the first six layers of

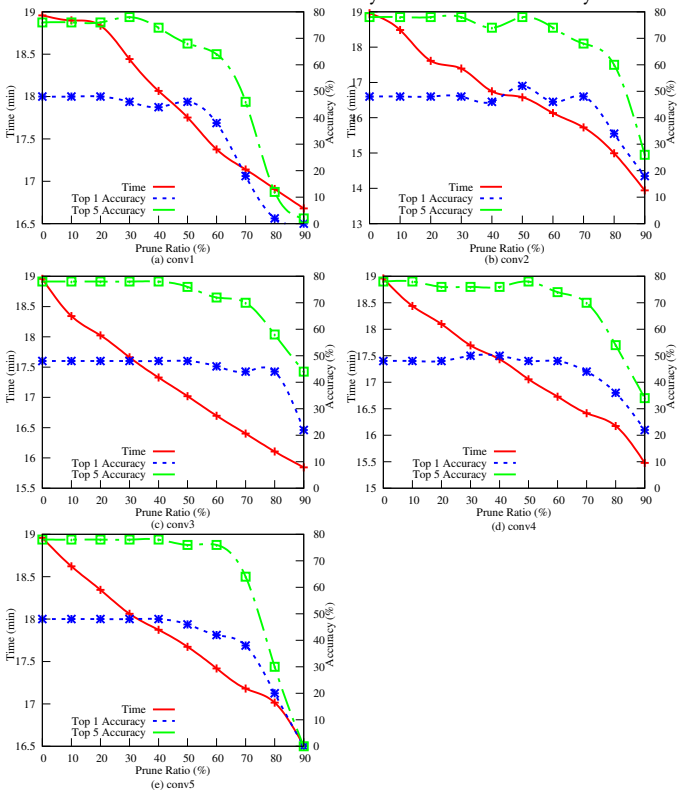

Figure 6: Caffenet Effect of Changing Accuracy with Individual Layer Pruning

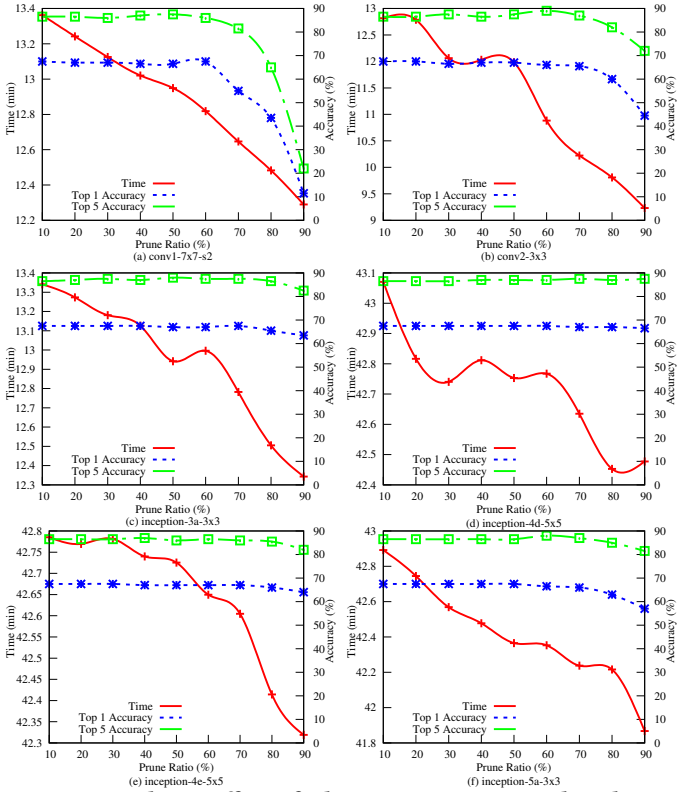

Figure 7: Googlenet Effect of Changing Accuracy with Individual Layer Pruning

Googlenet as shown in Figure 7, the accuracy starts dropping only after 60% of pruning while the time is seen drastically reduced. We call this region where accuracy remains almost unchanged while inference time reduces, sweet-spot region. Existence of sweet-spot regions is important for a cloud consumer to achieve the maximum possible inference accuracy with shorter time by selecting a degree of pruning within the sweet-spot region. For example, for Caffenet CNN inference on Amazon EC2 p2.xlarge instance, we can achieve up to one-eights reduction of inference time with no accuracy change, just by pruning `conv2` layer by 50%.

Observation 2: Impact of pruning on accuracy and execution time is different across convolution layers and does not directly correlate with convolution layer parameter values.

We observe that the variation of Top 1 and Top 5 inference accuracy differs across convolution layers. As shown in Figure 6, we observe the largest change in accuracy is in `conv1` layer where the Top 5 accuracy drops from 80% to 0% when the prune ratio varies from 0% to 90%. This is expected since `conv1` is the layer that receives the input image and also the largest in terms of the size of the image surface parameter. Rest of the layers demonstrate Top 5 accuracy drop from 80% to around 25% for the same prune ratio range. Similar variation pattern is observed for Top 1 accuracy as well. When it comes to execution time, `conv2` demonstrates the largest time range from 19 mins to 14 mins and `conv1` demonstrates smallest execution time range from 19 mins to 16.5 mins. When comparing these observations with the values of the parameters in each layer, it is apparent that the accuracy and execution time variation does not directly relate to the parameter values in the layer. As shown in Table 1 `conv4` in Caffenet contains the highest number of compute operations suggesting that pruning `conv4` would have the highest impact on accuracy and execution time. But our measurements show that this is not the case. Therefore, it is not trivial to determine how to select the best layer and pruning ratio for achieving the highest accuracy with the lowest execution time.

4.3.2 Pruning Multiple Layers. To understand the implications on time and accuracy due to dependency between convolution layers in the CNN, we combine sweet-spots from multiple layers and generate a single CNN model. We take guidance from Figure 3 which shows that in Caffenet, `conv1` and `conv2` account for more than 60% of the inference time. As shown in Figure 8, we compare three degrees of pruning; (i) no-pruning (`nonpruned`), (ii) pruning only `conv1` and `conv2` until the last sweet-spot (`conv1-2`) and (iii) pruning all convolution layers until the last sweet-spot (`all-conv`).

Observation 3: Combining CNN layer sweet-spots from multiple layers results in lower execution time, but may result in a drop in accuracy as well.

Considering the Top5 accuracy, last sweet-spots in Caffenet with lowest execution time for `conv1` and `conv2` are at 30% and 50% prune ratios. As shown in Figure 8, `conv1-2` which has the first two layers pruned recorded an execution time of 13 mins. In comparison, when the layers are pruned individually, the corresponding execution times for `conv1` and `conv2` are 18.4 mins and 16.7 mins respectively. Thus, by pruning the two layers together, we achieve one-fifth reduction in execution time. When it comes to accuracy, `conv1-2` recorded 70% Top5 accuracy, a 10% drop from the original accuracy of 80%. When all five layers are pruned until their last sweet-spots, as shown in `all-conv`, the recorded execution time is 11 mins and the Top5 accuracy is 62%. Thus, pruning all layers resulted in one-third reduction of execution time and 18% drop in accuracy. Similar observations could be made on Top1 accuracy as well. Therefore, in addition to the impact of individual convolution layers, it is important to understand the effect of dependency between convolution layers.

Since there are multiple ways to prune a given CNN, there exist many degrees of pruning. Every degree of pruning is associated with an inference accuracy. Moreover, each of them can be hosted on

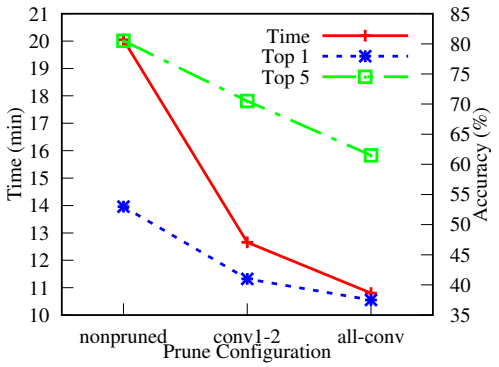

Figure 8: Caffenet Effect of Changing Accuracy with Multi-Layer Pruning

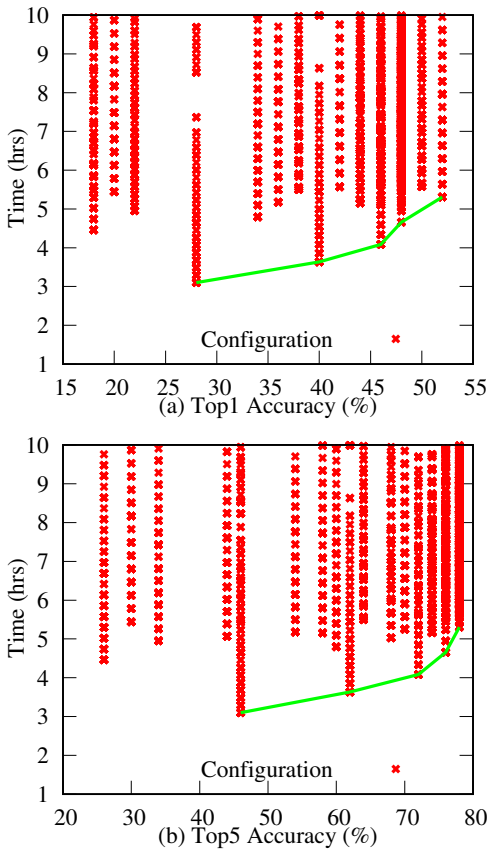

Figure 9: Impact of Accuracy on Cloud Execution Time

many cloud resource configurations. Thus we investigate the impact of inference on accuracy on execution time. For simplicity, we focus on the simpler Caffenet CNN and select a set of 60 versions of Caffenet CNN pruned in different degrees spanning a wide accuracy range. We select a resource configuration space consisting of three Amazon EC2 resource types from p2 category with three resource instances from each type and assume a time deadline of ten hours. Figure 9 is the resulting plot showing all feasible configurations for inferring one million images with Caffenet.

Observation 4: Given a time deadline, there exist many feasible configurations for CNN inference. Among them there are multiple Pareto-optimal configurations.

4.3.3 Time-Accuracy Pareto Frontier. We observe that there are 7654 feasible configurations for executing the Caffenet inference within the 10 hour time deadline. Among them, there are five Pareto-optimal configurations each for Top1 accuracy and Top5 accuracy. The Pareto-frontier is shown as a line in Figures 9 (a) and (b). Even among the Pareto-optimal configurations, Top1 accuracy varies from 27% to 53% whereas Top5 accuracy varies from 45% to 78% with execution time ranging from 3 to 5 hours in both cases. Selecting a Pareto-optimal configuration over other configurations significantly reduces the execution time. For example, as shown in Figure 9 (a), selecting the Pareto-optimal configuration with the highest accuracy reduces execution time by 50% compared to other configurations with the same accuracy.

4.4 Effect of Accuracy on Inference Cost

The effect of accuracy on cloud cost can be divided into two; (i) within a single resource type and (ii) across different resource types. Within a single resource type, the cost of a cloud resource instance solely depends on the time used. Thus, the cost incurred for CNN inference is a function of inference time. i.e. $inference\ cost = inference\ time * cost\ per\ unit\ time$. However, using a combination of resources makes the relationship more complex since we have to take into the cost-accuracy performance of different cloud resources. Figure 10 shows the total number of configurations that are capable of executing Caffenet within a $300 cost budget.

Observation 5: Given a cost budget, there exist many feasible configurations for CNN inference including multiple cost-accuracy Pareto-optimal configurations.

We observe that there are 1042 feasible configurations for executing the Caffenet inference within the $300 cost budget. Among them, there are five Pareto-optimal configurations for both Top1 and Top5 accuracy. Moreover, among the Pareto-optimal configurations, the Top1 accuracy varies from 27% to 53% and Top5 accuracy varies from with cost ranging from $69 to $119. The cost-accuracy Pareto-frontier overlaps with time-accuracy Pareto-frontier due to cost being the restricting factor when allocating resources in both cases. Cloud consumer can save cost significantly by selecting Pareto-optimal configurations over the rest. For example, as

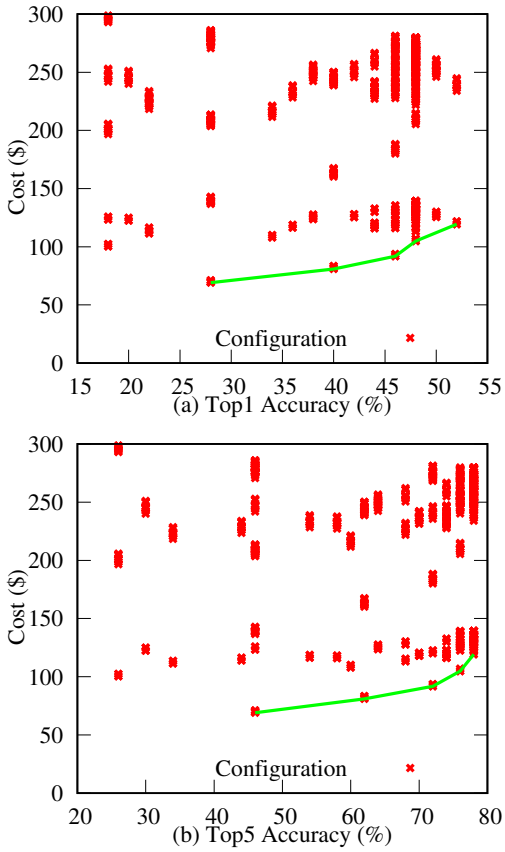

Figure 10: Impact of Accuracy on Cloud Cost

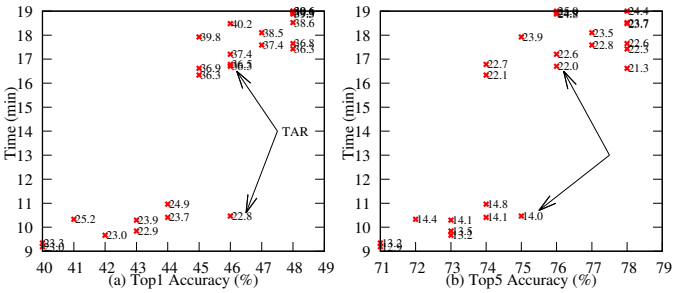

Figure 11: Time-Accuracy of Degrees of Pruning with TAR

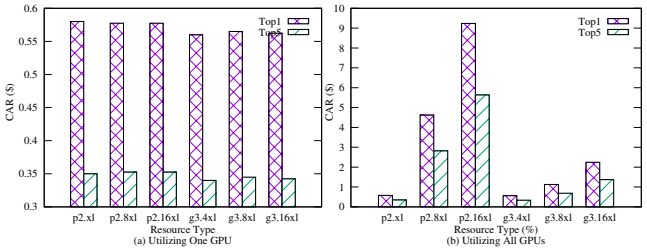

Figure 12: Caffenet CAR Across Resource Types

observed from Figure 10 (a), selecting the Pareto-optimal configuration for the highest accuracy saves up to 55% cost.

4.5 Quantifying Accuracy Performance

As shown in Figures 9 and 10, there exist a series of configurations spanning vertically on time axis and cost axis respectively for a given accuracy. This behaviour is caused by having multiple degrees of pruning that yield same accuracy but with difference performance.

4.5.1 Time-Accuracy Performance. To understand the accuracy performance of the application, we plot the execution time of Caffenet degrees of pruning in Figure 11. We measure the inference time and accuracy by changing the prune ratio for `conv1` and `conv2` of Caffenet. Leveraging on the sweet-spot regions observed in Figure 6, we vary the prune ratio from 0% to 40% for `conv1` and 0% to 50% for `conv2` in 10% increments. The TAR value for each pruning configuration is labeled next to the configuration on the figure. Since TAR represents the relative change between time and accuracy, for a given accuracy, the configuration with a lower TAR gives the least inference time thus, it can be used to determine the trade-offs offered by different degrees of pruning.

4.5.2 Cost-Accuracy Performance. The effect of accuracy on cloud cost can be divided into two parts; (i) within a single resource type and (ii) across different resource types.

Within a single resource type, the cost of a cloud resource instance solely depends on the time used. Thus, the cost incurred for CNN inference is a function of inference time. i.e. $inference\ cost = inference\ time * cost\ per\ unit\ time$. Therefore, TAR is sufficient to understand both the trade-offs between time and accuracy and, cost and accuracy for a single cloud resource type. When it comes to a comparison between different resources types, as shown in Table 3, different cloud resource types have different cost and performance. Thus, to quantify the cost of execution of cloud resource with respect to accuracy, we use CAR metric.

Figure 12 shows computed CAR values for Caffenet with first two convolution layers pruned by 20%, across six types of cloud resource types from two resource categories, `p2` and `g3`. Figure 12 (a) shows the CAR when all GPUs are allocated for inference and Figure 12 (b) shows when only one GPU is allocated. We observe that the CAR is approximately same for resource types within a resource category and varies across resource categories. For example, `p2` has a CAR of approximately $0.57 whereas `g3` has a CAR of approximately $0.35 with all GPUs allocated. When allocating cloud resources for CNN inference, it is ideal to utilize all GPUs in the allocated resource. However there may be circumstances where the application restricts the number of GPUs it can utilize due to requirements such as memory and storage.

4.5.3 Efficient Cloud Resource Allocation. To determine the cloud resource configuration for executing CNN inference efficiently, we propose using TAR and CAR as a heuristics to decide on the order of resource allocation. Given a CNN pruned in different degrees resulting in a set of CNNs with different accuracy (P), a set cloud resource instances (G), and a time deadline (T') and a cost budget (C'), Algorithm 1 illustrates our proposed resource allocation approach. The outputs are the resource configuration R, estimated time T and estimated cost C.

First we order elements in P by descending order of accuracy. If there are multiple elements with same accuracy, then those are ordered in ascending order of TAR. Then, the algorithm iterates through P and allocates cloud resources in the ascending order starting from the lowest CAR until a resource configuration is found that completes the inference within the given time deadline and cost budget. Whenever the cloud resource configuration is changed, the workload distribution across the new resource configuration is performed as described in section 3.

Efficiency. A major challenge in exploring the cloud configuration space is due to the large computational time required [25]. Configuration space exploration is a non polynomial time problem and is upper bounded by $O(2^{|G|})$. By using CAR as a heuristic to pick resources greedily, our algorithm performs under $O(|G|log|G|)$.

Algorithm 1 Resource Allocation with TAR and CAR

```
sort P in (i) accuracy descending order and (ii) TAR ascending
  order for elements with same accuracy
for each p ∈ P do
    sort G in ascending order of CAR
    R ← ∅
    for each g ∈ G do
        R ← g //add resource with lowest CAR
        distribute workload in R
        compute T and C
        if T < T′ and C < C′ then
            return p,R, T, C
        end if
    end for
end for
return ∅ //no feasible resource allocation
```

5 CONCLUSION

This paper presents a measurement-driven approach for investigating the cost-accuracy and time-accuracy trade-offs in executing applications on cloud. To represent applications with variable accuracy, we select CNN as the example and apply our approach on two widely used and well established image detection CNNs, Caffenet and Googlenet. In contrast to existing work that focus on CNN training performance, we investigate the CNN inference performance. We train our CNNs with more than one million images and use parameter pruning technique to change the inference accuracy. We perform our analysis on six types of cloud resources with GPUs from Amazon EC2 cloud with an inference dataset containing 50,000 images.

We show the existence of "sweet-spot" regions where the inference time can be reduced with no or insignificant reduction in inference accuracy. With different application configurations, we investigate the dependency among CNN layers and their impact on inference time. We show that inference time can be halved for just a one-tenth accuracy reduction with multi-layer pruning for Caffenet. We expose the existence of time-accuracy and cost-accuracy Pareto-optimal configurations in the large resource configuration space where the consumer is able to reduce the time by 50% and cost by 55% for achieving the highest possible inference accuracy for Caffenet CNN by selecting Pareto-optimal configurations. For quantifying the trade-off between inference time and accuracy, we introduce Time Accuracy Ratio (TAR) metric and show its usefulness by comparing across different degrees of pruning to save time for a given accuracy bound. To quantify the trade-off between cost and accuracy, we introduce Cost Accuracy Ratio (CAR) metric. We show that TAR and CAR can be utilized to efficiently determine cloud resource configurations for achieving the best inference accuracy within the given time deadline and cost budget constraints in polynomial-time.

ACKNOWLEDGMENTS

This work was supported by Singapore Ministry of Education through the Academic Research Fund Tier 1. The authors thank Amazon Web Services for AWS Cloud research credits.

REFERENCES

[1] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, and Gerald Penn. 2012. Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition. In *2012 IEEE international conference on Acoustics, speech and signal processing (ICASSP)*. IEEE, 4277–4280.

[2] Gene M. Amdahl. 1967. Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. In *Proc. of Spring Joint Computer Conference*. 483–485. http://doi.acm.org/10.1145/1465482.1465560

[3] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. 2017. Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)* 13, 3 (2017), 32.

[4] brandwatch.com. 2019. *53 Incredible Facebook Statistics and Facts*. https://www.brandwatch.com/blog/facebook-statistics/

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[6] Facebook. 2019. *Community Standards*. https://www.facebook.com/communitystandards/

[7] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. 2014. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115* (2014).

[8] Zhenhuan Gong, Xiaohui Gu, and John Wilkes. 2010. Press: Predictive Elastic Resource Scaling for Cloud Systems. In *Proc. of the International Conference on Network and Service Management*. 9–16.

[9] John L Gustafson. 1988. Reevaluating Amdahl's Law. *Commun. ACM* 31, 5 (1988), 532–533.

[10] Jie Han and Michael Orshansky. 2013. Approximate computing: An emerging paradigm for energy-efficient design. In *2013 18th IEEE European Test Symposium (ETS)*. IEEE, 1–6.

[11] Rui Han. 2015. Investigations Into Elasticity in Cloud Computing. *arXiv preprint arXiv:1511.04651* (2015).

[12] Jian He, Yonggang Wen, Jianwei Huang, and Di Wu. 2014. On the Cost–QoE Tradeoff for Cloud Based Video Streaming Under Amazon EC2's Pricing Models. *IEEE Transactions on Circuits and Systems for Video Technology* 24, 4 (2014), 669–680.

[13] Qiangui Huang, Kevin Zhou, Suya You, and Ulrich Neumann. 2018. Learning to prune filters in convolutional neural networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 709–718.

[14] Heehoon Kim, Hyoungwook Nam, Wookeun Jung, and Jaejin Lee. 2017. Performance analysis of CNN frameworks for GPUs. In *2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 55–64.

[15] P Kokkinos, Theodora A Varvarigou, Aristotelis Kretsis, Polyzois Soumplis, and Emmanouel A Varvarigos. 2013. Cost and Utilization Optimization of Amazon EC2 Instances. In *Proc. of 6th International Conference on Cloud Computing*. 518–525.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[17] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2016. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710* (2016).

[18] Xiaqing Li, Guangyan Zhang, H Howie Huang, Zhufan Wang, and Weimin Zheng. 2016. Performance analysis of gpu-based convolutional neural networks. In *2016 45th International Conference on Parallel Processing (ICPP)*. IEEE, 67–76.

[19] Xiaofan Lin, Cong Zhao, and Wei Pan. 2017. Towards accurate binary convolutional neural network. In *Advances in Neural Information Processing Systems*. 345–353.

[20] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. 2017. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*. 5058–5066.

[21] Ming Mao and Marty Humphrey. 2011. Auto-scaling to Minimize Cost and Meet Application Deadlines in Cloud Workflows. In *Proc. of International Conference for High Performance Computing, Networking, Storage and Analysis*. 49.

[22] Ming Mao, Jie Li, and Marty Humphrey. 2010. Cloud Auto-scaling with Deadline and Budget Constraints. In *Proc. of 11th International Conference on Grid Computing*. 41–48.

[23] Aniruddha Marathe, Rachel Harris, David Lowenthal, Bronis R de Supinski, Barry Rountree, and Martin Schulz. 2014. Exploiting Redundancy for Cost-effective, Time-constrained Execution of HPC Applications on Amazon EC2. In *Proc. of 23rd International Symposium on High-performance Parallel and Distributed Computing*. 279–290.

[24] Sparsh Mittal. 2016. A survey of techniques for approximate computing. *ACM Computing Surveys (CSUR)* 48, 4 (2016), 62.

[25] Sunimal Rathnayake, Dumitrel Loghin, and Yong Meng Teo. 2017. CELIA: Cost-time Performance of Elastic Applications on Cloud. In *Proc. of 46th International Conference on Parallel Processing*. 342–351.

[26] Sunimal Rathnayake, Lavanya Ramapantulu, and Yong Meng Teo. 2018. Cost-Time Performance of Scaling Applications on the Cloud. In *2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 30–33.

[27] Suman Ravuri and Oriol Vinyals. 2019. Classification accuracy score for conditional generative models. In *Advances in Neural Information Processing Systems*. 12247–12258.

[28] Upendra Sharma, Prashant Shenoy, Sambit Sahu, and Anees Shaikh. 2011. A Cost-aware Elasticity Provisioning System for the Cloud. In *Proc. of 31st International Conference on Distributed Computing Systems*. 559–570.

[29] Hoo-Chang Shin, Holger R Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald M Summers. 2016. Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging* 35, 5 (2016), 1285–1298.

[30] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.

[31] Wei Wen, Cong Xu, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2017. Coordinating Filters for Faster Deep Neural Networks. In *The IEEE International Conference on Computer Vision (ICCV)*.

[32] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. 2017. Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044* (2017).