Context-Aware Convolutional Neural Network over Distributed System in Collaborative Computing

Jinhang Choi, Zeinab Hakimi, Philip W Shin, Jack Sampson, Vijaykrishnan Narayanan

Pennsylvania State University

University Park, PA, USA

{jpc5731,zuh17,pws5345,sampson,vijay}@cse.psu.edu

ABSTRACT

As the computing power of end-point devices grows, there has been interest in developing distributed deep neural networks specifically for hierarchical inference deployments on multi-sensor systems. However, as the existing approaches rely on latent parameters trained by machine learning, it is difficult to preemptively select front-end deep features across sensors, or understand individual feature's relative importance for systematic global inference. In this paper, we propose multi-view convolutional neural networks exploiting likelihood estimation. Proof-of-concept experiments show that our likelihood-based context selection and weighted averaging collaboration scheme can decrease an endpoint's communication and energy costs by a factor of 3 $\times$, while achieving high accuracy comparable to the original aggregation approaches.

CCS CONCEPTS

• **Computing methodologies** $\rightarrow$ **Neural networks**; *Distributed artificial intelligence*; *Feature selection*; • **Computer systems organization** $\rightarrow$ **Neural networks**; *Distributed architectures*;

KEYWORDS

Convolutional neural network, distributed intelligence system

ACM Reference Format:

Jinhang Choi, Zeinab Hakimi, Philip W Shin, Jack Sampson, Vijaykrishnan Narayanan . 2019. Context-Aware Convolutional Neural Network over Distributed System in Collaborative Computing. In *The 56th Annual Design Automation Conference 2019 (DAC '19)*, June 2–6, 2019, Las Vegas, NV, USA. ACM, New York, NY, USA, 6 pages. <https://doi.org/10.1145/3316781.3317792>

1 INTRODUCTION

Deep Neural Networks (DNNs) have been proposed in recent years that automatically discover deep features providing rich representations for intelligent recognition. Endpoint devices have started employing DNN models as and where hardware resources have become more capable of processing a DNN's computation-intensive operations [8, 13]. In the Internet of Things (IoT) era, such devices attached to sensors offer tremendous opportunities for distributed system design. Suppose that we design a security system performing face recognition across multiple cameras. The final classification cannot be reliably accomplished unless at least one camera sensor takes a picture of a human's face in the near-front view without

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

DAC '19, June 2–6, 2019, Las Vegas, NV, USA

© 2019 Association for Computing Machinery.

ACM ISBN 978-1-4503-6725-7/19/06...\$15.00

<https://doi.org/10.1145/3316781.3317792>

Figure 1: Each view has discriminative visual information corresponding to class instance's property. Viewpoint-wise images are collected from viewpoint controlled dataset [2].

glare or active motion. Even though data acquisition against dynamic and adversarial environments is frequently imperfect due to noisy and unknown conditions, we know that the deployed subset of cameras are expected to capture at least some front view images. By providing each IoT device with a DNN evaluating data acquisition, we can control data credibility and have a chance to make features that combine multimodal data.

Three-dimensional, real world object recognition is improved by a number of well-defined data sources with respect to geometric variance [3]. Recent studies in translation of viewpoint-wise 2D image projections to deep features demonstrate promising performance in terms of both accuracy and retrieval [4, 9, 16]. DNNs can learn discriminative information from each view as shown in Figure 1. The more accurate a level of recognition we target, the more viewpoints should be provided. Since we do not know which view at a given time will be the most informative to a DNN, however, current multi-view approaches deploy omni-viewpoint feature extractors relying on latent variables trained by viewpoint augmentation. Consequently, feeding every viewpoint 2D image projection to the extractor is imperative to acquire strong priori.

Meanwhile, Figure 1 also implies that training each endpoint feature extractor on constrained view data can produce inherently viewpoint-specific IoT devices. The endpoint devices can make a fast decision with high confidence [18]: Classification is terminated at an endpoint if a sensor's observation is likely close to the front-end DNN's intrinsic context. However, it is worth noting that all features do not have the same importance (e.g. the front view features are the most important in the case of face recognition). *What if we have multiple distinctive endpoint nodes stating that their data acquisition is competitive to represent features for classification?* The existing hierarchical architecture designs have the following limitations:

- (1) If the most confident features are activated among endpoint nodes, we have to complete an exhaustive inter-node comparison, which increases energy consumption significantly. If such features are just activated from a single node, then we will lose other nodes' prediction flexibility by removing sensor fusion, which is error-prone.

Figure 2: Overview of MVCNN-Wavg architecture in distributed system design (Bold box indicates front-end DNN deployment.)

- (2) If the back-end system equivalently gives credit to all endpoint features, then we do not know how much each node’s context is important. As a result, prevalent, but less important features within many views may overshadowing discriminative visual features in a few views, and we would have to always feed all extracted features to the classifier for accurate recognition, incurring communication costs.

To tackle such design difficulties, we propose a collaborative Multi-View Convolutional Neural Network architecture using a weighted mean of likelihoods (MVCNN-Wavg). We train MVCNN-Wavg as a whole DNN model in a framework of end-to-end optimization similar to proposed distributed DNN designs [19]. Then, the model is separated so that we can allocate each submodule to the desired position in a distributed system. As depicted in Figure 2, each feature extractor (CNN- A_n), at inference, contains viewpoint context corresponding to latent DNN parameters, which is encoded indirectly by the effect of context group-wise regularization during training due to view-specific dataset. Conceptually, each front-end node should have its own local feature extractor, but be expected to have some commonality between neighboring nodes in a context group. When sensing data, each node determines its feature quality by retrieving entropy as a proxy of context representativeness. More specifically, each feature is weighted with respect to its importance in terms of likelihood. Then, any nodes having less important features can be pruned, while they still have a chance of contributing to recognition by regulating the likelihood cut-off level. If more than two context-wise features are confident, MVCNN-Wavg proportionally aggregates them to establish feature normalization in weighted pooling. The back-end DNN system (CNN-B) uses the multimodal feature representation to make global prediction.

The contributions of this paper include:

- A context-aware feature aggregation scheme, weighted pooling, is integrated to an end-to-end DNN training framework, where each endpoint node can establish a sensor’s modality-based context explicitly. By adjusting the granularity of context grouping, we can achieve 98% classification accuracy when its confidence score is higher than 56th percentile.
- We demonstrate that likelihood estimation at endpoint nodes significantly decreases communication and energy costs in a proposed distributed system. Likelihood-based node pruning removes 66.7% of sensor node activations in 12-view

MVCNN and 91.9% in 80-view MVCNN while sustaining 89% top-1 classification accuracy in tested datasets. The reduced endpoint node reservation offers new opportunities to increase application coverage or reliability accordingly.

2 BACKGROUND AND OVERVIEW

Distributed DNNs (DDNNs) use end devices to collect data and, while performing some fraction of inference processing locally on these end devices, rely on cloud resources for training and higher inference accuracy. Recent research on DDNNs has investigated efficient mapping of larger networks onto end nodes and other less compute-capable platforms. Assuming that early stage features have sufficient information to make decisions for some inputs, BranchyNet [18] introduced an architecture for early termination to trade off substantial computation reductions for limited loss in accuracy. Branches are exploited in each layer to stop its inference in the early stage, which negates more confident nodes that might come after the early termination point. Teerapittayanon *et al.* [19] offload the early exit point features to edge and cloud, then aggregate them across end devices in hierarchical DNN design if the condition of early termination is not satisfied. Even though such a dynamic network hierarchy would decrease communication cost between edges and cloud, collecting the features necessitates participation from all front-end nodes. In contrast, we assume a fixed depth for classification, but exploit learned models of positional feature redundancy and input-dependent feature saliency in multi-node scenarios to elide computation on features with low expected marginal information contributions.

2.1 Likelihood estimation using entropy

In machine learning, minimizing negative log-likelihood of multivariate probability distribution presents a DNN model’s optimization. To express how much uncertainty a probability distribution $P(\mathbf{x}; \Theta)$ contains, Shannon entropy [6] is defined as:

$$\text{entropy}(y) = -\mathbb{E}_{\mathbf{x} \sim P} [\log(y)] = - \sum_{c \in C} y_c \log P(\mathbf{x}; \Theta), \quad (1)$$

where y_c is a probability instance of all possible DNN inferences in C classes¹. As uncertainty of a probability distribution decreases, the latent DNN parameters, Θ , construct classification results based

¹The machine learning literature generally uses softmax regression function [6].

Figure 3: Various views in a scene from KITTI Dataset [5]

on the context of training dataset. Consequently, entropy is capable of representing the proximity to training dataset when it is close to 0 ($entropy < 1$). BranchyNet introduces an estimation of uncertainty in the distribution of inferences by replacing $P(x; \Theta)$ to y_c in computing entropy. Note that entropy does not measure confidence, but uncertainty. In other words, a clear classification result essentially corresponds to the level of internal context likelihood.

2.2 Context-aware feature selection

The quantity of entropy does not give any information except likelihood to us, but its comparison provides us with explicit context-awareness. Suppose that we train multiple DNN feature extractors by separating training dataset with respect to each sensor's modality, viewpoint. Table 1 depicts which context of DNN parameter is selected among four viewpoint-wise VGG-16 models [15] when we test eight hundred sample images of four viewpoint-specific images. Training/testing images are sampled from iLab-80M dataset [2]. We selected a model which has the smallest entropy of inferences. The preliminary result indicates that the context predicted by the DNN model at an endpoint is significantly correlated with their raw data. Since the sensor itself is a priori agnostic, DNN models in addition to sensors bolster reliable data acquisition. For example, geographically dispersed cameras in a cross street are responsible for monitoring vehicles in specific views. Cameras can filter out a region of unrelated objects (e.g. human sitting on bench in Figure 3) when they send an image to the cloud as the region is out of the context that the camera has to focus on. By associating data with a desired context, we can sustain prior information that can be signaled to the system in an abstract level [9]. In comparison to the end-to-end entangled DNN system design, context-awareness gives a chance to group relevant data into representative features.

2.3 Context grouping

Although the most significant features are proposed by the major modal sensors, auxiliary sensors would also capture important data to support feature representation due to data similarity. As shown in Figure 4, for instance, view4 context of the entire samples is selected higher than 90% in netV4 (front left view) model, whereas a significant fraction of view2 context of the boat class samples can be selected in either netV2 (front right view, 52%) or netV3 (side view, 36%) models. Various context properties trained on a feature extractor provide robust representation while simplifying feature

Table 1: Example of entropy-based context awareness: view-point selection of trained DNN model against test images

Selected DNN Model (4 VGG-16s [15])	Viewpoint of 800 test images			
	View1	View2	View3	View4
VGG 16–View1 (netV1)	88%	1%	1%	0%
VGG 16–View2 (netV2)	4%	87%	6%	2%
VGG 16–View3 (netV3)	1%	7%	89%	2%
VGG 16–View4 (netV4)	7%	5%	4%	96%

	View1 test images					View2 test images			
	Boat	Bus	Car	Plane		Boat	Bus	Car	Plane
netV1	82%	80%	92%	88%	netV1	6%	0%	0%	0%
netV2	8%	8%	2%	4%	netV2	52%	96%	100%	90%
netV3	0%	0%	6%	0%	netV3	36%	0%	0%	2%
netV4	10%	12%	0%	8%	netV4	6%	4%	0%	8%

	View3 test images					View4 test images			
	Boat	Bus	Car	Plane		Boat	Bus	Car	Plane
netV1	2%	2%	0%	2%	netV1	0%	0%	0%	0%
netV2	12%	0%	0%	22%	netV2	6%	0%	2%	2%
netV3	82%	98%	100%	64%	netV3	2%	4%	2%	2%
netV4	4%	0%	0%	12%	netV4	92%	96%	96%	96%

Figure 4: Per-instance VGG-16 model selection in Table 1 – Salient (highlighted row) vs. potential contexts (dashed box)

description [4]. However, architectural design to support context clustering is often intractable as internal DNN parameters are black boxes trained by supervised or unsupervised learning. To address ambiguity in local estimation, Hu *et al.* proposed to use weighted pooling in DNN design [7]. While forwarding an importance of features to the back-end DNN layers, the weighted pooling can be entangled with the end-to-end network framework as it is differentiable in the process of back-propagation. As a result, by grouping similar sensor data onto a training dataset per feature extractor, we can intrinsically generalize context groups with respect to the designated granularity in the training phase.

3 COLLABORATIVE NETWORK IN DISTRIBUTED SYSTEM

Multi-View Convolutional Neural Networks (MVCNNs) have been investigated for many years. Su *et al.* [16] proposed a MVCNN approach that generates a descriptor for each view individually and combines different viewpoints through dimensional concatenation and maximum value activation in feature pooling for collaborative recognition. Subsequent research [4, 9] has extended this initial model, but these works all perform view-point pooling in a manner that treats all views equally, resulting in viewpoint-wise context ignorance. RotationNet [9] predicts category and viewpoint from each single view feature space, and associates candidate viewpoints with its corresponding feature for the following inference. However, RotationNet restricts each viewpoint to a specific context in the network schema, which leads to viewpoint selection that neglects less important viewpoints' contribution to specific context establishment. GVCNN [4] uses viewpoint grouping with respect to their content and discriminative information discovered in the training phase. Unfortunately, GVCNN requires an exhaustive combination of feature sets for aggregation since we do not know which feature contexts will be grouped. Our proposed approach addresses limitations in context-agnostic feature contribution by using weighted mean to keep track of all the viewpoint context likelihoods.

3.1 Architecture of MVCNN-Wavg

MVCNN-Wavg is not constrained to any specific DNN architecture. As described in Figure 2, likelihood estimation is a residual layer attached to the original feature extractors². To emphasize the importance of viewpoint-wise features, we need to find which view has more informative features to be selected for the following DNN layers. Based on entropy-based likelihood estimation ξ , we define

²Please refer to <https://github.com/cepiross/MVCNN-TensorFlow> for details

importance weight λ for each sensor's modality ϕ as:

$$\xi(y; \phi) = \sum_c \sigma(-y_{c;\phi} \log(y_{c;\phi} + \epsilon)) \quad (2)$$

$$\lambda_{y;\phi} = \frac{1}{\xi(y; \phi) + \rho}, \quad (3)$$

where $\epsilon = 1.0 \times 10^{-5}$, $\rho = 0.01$. σ is a rectified linear unit (ReLU) activation to suppress negative impact undefined in likelihoods. In other words, we clip statistically low confidence of inference as 0.001%, to avoid numerical precision error, and set the limit of an importance weight to 100. Then, we apply likelihood thresholding δ to the importance weight such that

$$\delta(y, \mu) = \sigma(\lambda_{y;\phi} - \mu) = \max(\lambda_{y;\phi} - \mu, 0) \quad (4)$$

by manipulating the likelihood cut-off level μ . If data acquisition from a sensor is not likely to be relevant in a context group, then we can signal it to the following feature pooling layer dynamically. Since ReLU-based neural networks have proven to be robust in learning frameworks, we use ReLU for our importance regularization function. Any proposed nonlinear activation functions used throughout the machine learning literature could be used instead; Sensitivity to choice of activation function is an interesting exploration, but outside the scope of this paper.

To estimate the feature's likelihood against a group of viewpoint contexts, memory-intensive inner product operations would be deployed in the front-end devices. As presented in [23], we use class activation mapping (CAM) technique for end devices so that can replace memory-intensive operations with computation-intensive convolutions with little degradation of inference accuracy in computing likelihood. It is important to note that likelihood estimation at the endpoint node is not the final classifier, but an auxiliary classifier like network regularizer [17], which improves training convergence in DNN generalization. Therefore, such accuracy deviation in likelihood estimation will have limited propagation through the remainder of the network. Still, the back-end DNN layers have the original fully-connected neurons to benefit from comprehensive correlation between multimodal features in fused representation.

Finally, clustered context groups aggregate their features by normalizing the value in the weighted pooling:

$$\text{Pooling}_{\text{avg}} = \sum_i c_i X_i \quad (5)$$

such that $c_i = \delta_i / \sum_\phi \delta_\phi$. As the importance weights are only determined by likelihood estimation, there is no trained DNN parameter for our proposed pooling scheme. Nevertheless, likelihood-based importance weights populate the collaboration between context groups, which in turn leads to sensor fusion at the back-end DNN layers. Moreover, negated likelihood quantities proposed by (4) enable us to selectively skip endpoint devices without exhaustive node comparison nor computation at the stage of feature aggregation. This is conceptually similar to stochastic short-circuiting designs [10], but proposed at distributed system-level.

3.2 Training MVCNN-Wavg

We assume synchronous stochastic gradient descents updated by back-propagation, where MVCNN-Wavg is trained as a whole DNN model with intermediate modality augmentation in the end-to-end training framework for distributed system design. Instead of selectively optimizing system design in network dissection [20], our proposed approach jointly optimizes front-end feature extractors

Table 2: Profile of computation (and memory footprint for trained DNN parameters) cost in DNN dissection

DNN Model	front-end feature/sensor		back-end classifier	comm. traffic
	extractor	likelihood		
AlexNet	1.1B Ops ¹ (9 MiB)	600M Ops ² (14 MiB)	55M Ops (209 MiB)	# sensors × 36KiB
GoogLeNet	1.0B Ops (10 MiB)	3.4M Ops (8.4 MiB)	290M Ops (13 MiB)	# sensors × 404KiB

¹Billion, ²Million arithmetic operations

and back-end classifiers. First, for given object instances, every front-end feature extractor receives sensor-wise data projections per instance as a batch of training data. For instance, if we have 12 viewpoints and aim at constructing 6 context groups, then instance images of 2 neighbor viewpoints are fed to each feature extractor. Second, the final classifier receives the output of weighted pooling when $\mu = 0$, expecting that all context groups of feature extractors can contribute to a fused feature representation. Third, since training is synchronous, the order of training data and its corresponding label are exactly the same in both front- and back-end. We only supervise association of each sensor's data with the expected context group, where we implicitly know which context is trained. Due to needing only a simple training framework with minimal data augmentation, it is possible to train the distributed DNN model even in a single powerful GPU.

4 EXPERIMENTAL SETUP

We deploy two representative CNN designs on MVCNN-Wavg: AlexNet [11] and GoogLeNet [17] to evaluate collaborative DNN inference with sensor fusion. More specifically, network parameters of the original models trained on the Imagenet dataset [14] are extracted via transfer learning [22]. The first five convolution layers of AlexNet are transferred to each viewpoint-wise front-end feature extractor, and two fully connected layers of AlexNet are transferred to the back-end final classifier. In the case of GoogLeNet, the first three convolution layers and six inception layers are transferred to the front-end, the remaining three inception layers are transferred to the back-end. Table 2 depicts computation, memory footprint, and communication costs between the front-end feature extractor and the back-end classifier when individual DNN submodules are deployed in a distributed system. Please note that memory and communication costs are based on single precision floating point weight and activations. With respect to training costs, the size of front-end network parameters is proportional to viewpoint context groups. The size of feature space does not change while we keep using the same number of viewpoints. Therefore, the potential training overhead depends on how many viewpoint contexts we want to construct. For the proposed number of viewpoint groups, the additional network parameters account for 6.4% and 36.5% memory overheads for AlexNet and GoogLeNet respectively, which are amortized across the entire deployment lifetime of the system.

For a proof-of-concept environment, we use Princeton 3D CAD ModelNet dataset [21] to synthesize data acquisition from 12-view and 80-view sensors as proposed by Su *et al.* [16]. Consequently, we can evaluate how many endpoint sensors should be located in a distributed system design. The iLab-80M [2] dataset presents more realistic data by taking pictures of an object instance with control of viewpoint placement, camera focusing, and lighting source in the lab. Among the permutations of 88 different viewpoints, we

Figure 5: Sensor placements with respect to viewpoint setup
narrow down the target viewpoints similar to the ModelNet experiment so that we can explore the impact of context grouping on the distributed system in detail. In short, training/testing 3D CAD objects of 40 classes use 3,183 training object instances and 800 testing object instances (38,196/9,600 12-view data and 254,640/64,000 80-view data), and 2,377/600 iLab-80M object instances of 10 classes are used with respect to viewpoint-wise 256×256 image projections. We assume static object placement aligned with context groups.

We use TensorFlow [1] to train and test the two fully annotated datasets individually on an NVIDIA Tesla V100 GPU. To consistently train DNN designs, we follow prior work [17] in setting up hyperparameters such as data random cropping and gradient clipping. Finally, we evaluate inference accuracy and communication costs by assuming deployment of front-end feature extractors on each endpoint device, and the back-end classifier on a cloud system.

5 RESULTS AND DISCUSSION

Table 3 describes the impact of viewpoint-wise sensor deployments depicted in Figure 5 on MVCNN object classification against 3D CAD ModelNet40 dataset. For a fair comparison, the original feature aggregation schemes using either maximum (max) and average (avg) pooling are also placed in front of the classifier. As we place more sensors, finer differences will be collected, which causes redundancy in raw data. Since an omni-viewpoint context feature extractor (single group) absorbs all visual information from every viewpoint-wise training data, the most discriminative features will survive the process of training. Naturally, max pooling is almost the best approach if we presume a single context. On the other hand, avg pooling receives each sensor’s information equally. Since frequent features overshadow important features, avg pooling necessitates many sensors’ placement in reliable positions. For instance, compared to max pooling, avg pooling loses top-1 accuracy more than 10% with 12 view sensors, whereas it maintains accuracy loss less than 2% with 80 view sensors. Our proposed weighted average

Table 3: Impact of sensor deployments on top-1 classification accuracy of MVCNN(AlexNet) against ModelNet dataset

Context Groups	12 endpoint views			80 endpoint views		
	max	avg	wavg	max	avg	wavg
orig.	88.9%	85.6%	88.3%	88.8%	86.0%	86.8%
2	88.1%	88.5%	88.4%	—	—	—
3	88.3%	79.4%	88.5%	—	—	—
4	89.4%	73.8%	89.4%	88.5%	87.5%	88.4%
6	87.6%	72.9%	82.4%	—	—	—
10	—	—	—	89.0%	87.6%	88.5%
12	87.0%*	76.0%*	87.8%*	—	—	—
16	—	—	—	89.3%	88.9%	88.0%
40	—	—	—	89.8%	89.0%	89.0%

*Each sensor has its own feature context without grouping.

Figure 6: Precision and recall of classification with respect to context group in 12-view MVCNN (AlexNet vs. GoogLeNet)

(wavg) pooling lies between max and avg pooling. While giving credit to important features, it also considers prevalent features at the same time. Therefore, when we increase the number of context groups, the trend of accuracy improvement in wavg pooling is more similar to max pooling than avg pooling. The realistic object instances in the iLab-80M dataset also follow this tendency. With 12-view setups, the best accuracy of max pooling reaches 90.2% in 2 context groups (6 sensor grouping), and wavg pooling reaches 89.7% in both 2 and 6 context groups.

Context grouping provides a distributed system with the potential for improving recognition capability. This is because features agreed as important within a context group tend to be an important property of an object. Note, however, that, while commonly observed, feature frequency is not a *necessary* condition for feature saliency. In our experiment, we arbitrarily group neighboring sensors taking account of information locality. Though such locality-based grouping does not always retrieve a representative context, the result shows that the best top-1 classification accuracy in max pooling reaches 89.4% with 4 context groups in 12 views (3 sensor grouping), and the accuracy reaches 89.8% with 40 context groups in 80 views (2 sensor grouping). When every sensor node has its own specialized feature extractor, sensor contexts have too many details to make a generalized feature representation, which results in 87.0% accuracy. Unlike max pooling, avg pooling does not take advantage of context groups effectively. In comparison to 80 views, where the top-1 accuracy keeps increasing, the accuracy of avg pooling in 12 views deteriorates below 80% due to lack of redundancy in reliable view sensors. However, both of max and avg pooling do not benefit from context grouping significantly. As shown in Figure 6, the context-agnostic feature aggregation attached to AlexNet achieves 46 percentile high confidence score to give 98% accuracy in max pooling and 20 percentile high score in avg pooling, whereas our proposed context-aware aggregation in wavg pooling achieves 56 percentile high score to give the same accuracy when we use context groups. Furthermore, with wavg pooling, 12 viewpoint-wise AlexNet-based feature extractors associated with 6 context groups promises robustness similar to GoogLeNet.

If the base DNN model is GoogLeNet, 78/60/72 percentile high confidences gives 98% top-1 accuracy in max/avg/wavg pooling respectively because of realizing a more powerful feature extractor, where the best accuracy reaches 90.5% in max/avg pooling, and 90.4% in wavg pooling. While the results do not exhibit large sensitivity to feature contexts choices in simple system designs, context-awareness offers new opportunities in system reliability

Figure 7: Trade-off between accuracy and communication cost with likelihood thresholding (GoogLeNet on 12-view MVCNN-Wavg with 1, 2, 3, 4, 6, and 12 context groups)

and energy saving. For instance, when 50% of sensors' inputs are subjected to random noise on iLab-80M dataset, our wavg pooling scheme retains $87.9 \pm 0.9\%$ accuracy while the original reduces to $85.3 \pm 1.3\%$ (max) / $72.9 \pm 1.5\%$ (avg) in GoogLeNet of 4 context groups. Note that each scheme's best accuracies are 91.2%/92.3%/91.7%, respectively. Figure 7 depicts the top-1 accuracy and communication cost between front-end feature extractor and back-end classifier with wavg pooling in ModelNet dataset. Because we use representative features with context grouping, it is possible to consider sensor pruning from nodes not relevant to the target context. As MVCNN-Wavg requires an auxiliary classifier to determine likelihood, there is inevitably an additional computation cost, up to 54.5% at each endpoint sensor node in AlexNet. However, the computation penalty is smaller than 1% in the case of GoogLeNet since the existing DNN layers are reused. According to likelihood thresholding tests, we can turn off 66.7% of sensor nodes while sustaining the same accuracy (90.4%) in 12-view setups. If more aggressive energy saving is targeted, then we can turn off 83.3% of sensor nodes with 2.7% accuracy loss by assigning independent feature contexts to each sensor. In the case of 80-view AlexNet, we turn off 91.9% of sensor nodes with 89% classification accuracy.

Such preemptive node pruning also provides communication cost reduction. Table 2 describes that the amount of feature data is proportional to the active sensors. Since communication cost dominates energy consumption in IoT devices, a dynamic trade-off between computation and communication on end nodes improves a distributed system's processing capacity [12]. Similarly, while alleviating wireless network congestions in IoT environment, 3 – 6× smaller sensor node activations provide the idle nodes with the potential of decentralized inter-node collaboration to relieve computation requirements. As a topic of future work, we are exploring the possibility of each node to independently evaluate alternative context group membership to adapt to dynamic environments.

6 CONCLUSION

In this paper, we propose MVCNN-Wavg, a method for augmenting distributed DNN inference wherein the endpoint nodes use likelihood estimation to decide whether their context provides value to the next layer of computation. Our approach can reduce computational effort while maintaining overall system accuracy by automatically constructing context-awareness information from grouped multimodal sensor data in the end-to-end training framework. This allows the centralized back-end of a DNN system to

focus on a representative subset of context node data instead of needing to observe all sensors' data acquisitions. We demonstrate a decrease in communication and energy costs of more than 3× with the same level of inference accuracy.

ACKNOWLEDGMENTS

This work was supported in part by NSF Expeditions in Computing Program: Visual Cortex on Silicon CCF 1317560, and Semiconductor Research Corporation (SRC) Center for Brain-inspired Computing Enabling Autonomous Intelligence (C-BRIC). We are particularly appreciative to Prof. Laurent Itti (USC), who provided us with the iLab-80M dataset.

REFERENCES

- [1] Martin Abadi et al. 2016. TensorFlow: A System for Large-scale Machine Learning. In *Proc. of the 12th Conf. on Operating Systems Design and Implementation (OSDI)*. USENIX Association, Savannah, GA, 265–283.
- [2] Ali Borji, Saeed Izadi, and Laurent Itti. 2016. iLab-20M: A Large-Scale Controlled Object Dataset to Investigate Deep Learning. In *Proc. of Conf. on Comp. Vis. and Patt. Recog. (CVPR)*. IEEE, 2221–2230.
- [3] Jinhang Choi et al. 2018. Stochastic Functional Verification of DNN Design through Progressive Virtual Dataset Generation. In *Proc. of Int'l Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–5.
- [4] Yifan Feng et al. 2018. GVCNN: Group-View Convolutional Neural Networks for 3D Shape Recognition. In *Proc. of Conf. on Comp. Vis. and Patt. Recog. (CVPR)*. IEEE, 264–272.
- [5] Andreas Geiger et al. 2013. Vision meets Robotics: The KITTI Dataset. *Int'l Journal of Robotics Research (IJRR)* 32, 11 (2013), 1231–1237.
- [6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press, Cambridge, MA. <http://www.deeplearningbook.org>.
- [7] Yuanming Hu, Baoyuan Wang, and Stephen Lin. 2017. FC4: Fully Convolutional Color Constancy With Confidence-Weighted Pooling. In *Proc. of Conf. on Comp. Vis. and Patt. Recog. (CVPR)*. IEEE, 330–339.
- [8] Shubham Jain et al. 2018. Compensated-DNN: Energy Efficient Low-precision Deep Neural Networks by Compensating Quantization Errors. In *Proc. of the 55th Annual Design Automation Conference (DAC)*. ACM, New York, NY, 38:1–38:6.
- [9] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. 2018. RotationNet: Joint Object Categorization and Pose Estimation Using Multiviews From Unsupervised Viewpoints. In *Proc. of Conf. on Comp. Vis. and Patt. Recog. (CVPR)*. IEEE, 5010–5019.
- [10] Kyoungheon Kim et al. 2016. Dynamic Energy-accuracy Trade-off Using Stochastic Computing in Deep Neural Networks. In *Proc. of the 53rd Annual Design Automation Conference (DAC)*. ACM, New York, NY, 124:1–124:6.
- [11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25 (NIPS)*. Curran Associates, Inc., 1097–1105.
- [12] Kaisheng Ma et al. 2018. NEOFog: Nonvolatility-Exploiting Optimizations for Fog Computing. In *Proc. of the 23rd Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. ACM, New York, NY, 782–796.
- [13] Burhan A. Mudassar, Jong Hwan Ko, and Saibal Mukhopadhyay. 2018. Edge-Cloud Collaborative Processing for Intelligent Internet of Things: A Case Study on Smart Surveillance. In *Proc. of the 55th Design Automation Conference (DAC)*. ACM, New York, NY, 146:1–146:6.
- [14] Olga Russakovsky et al. 2015. ImageNet Large Scale Visual Recognition Challenge. *Int'l Journal of Computer Vision (IJCV)* 115, 3 (2015), 211–252.
- [15] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR* abs/1409.1556 (2014).
- [16] Hang Su et al. 2015. Multi-view convolutional neural networks for 3d shape recognition. In *Proc. of Int'l Conf. on Computer Vision (ICCV)*. IEEE, 945–953.
- [17] Christian Szegedy et al. 2015. Going Deeper With Convolutions. In *Proc. of Conf. on Comp. Vis. and Patt. Recog. (CVPR)*. IEEE, 1–9.
- [18] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. 2016. BranchyNet: Fast inference via early exiting from deep neural networks. In *Proc. of the 23rd Int'l Conf. on Pattern Recognition (ICPR)*. IEEE, 2464–2469.
- [19] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. 2017. Distributed Deep Neural Networks Over the Cloud, the Edge and End Devices. In *Proc. of 37th Int'l Conf. on Distributed Computing Systems (ICDCS)*. IEEE, 328–339.
- [20] Wei-Yu Tsai et al. 2017. Co-training of Feature Extraction and Classification Using Partitioned Convolutional Neural Networks. In *Proc. of the 54th Annual Design Automation Conference (DAC)*. ACM, New York, NY, 58:1–58:6.
- [21] Zhirong Wu et al. 2015. 3D ShapeNets: A Deep Representation for Volumetric Shapes. In *Proc. of Conf. on Comp. Vis. and Patt. Recog. (CVPR)*. IEEE, 1912–1920.
- [22] Jason Yosinski et al. 2014. How Transferable Are Features in Deep Neural Networks?. In *Proc. of the 27th Int'l Conf. on Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 3320–3328.
- [23] Bolei Zhou et al. 2016. Learning Deep Features for Discriminative Localization. In *Proc. of Conf. on Comp. Vis. and Patt. Recog. (CVPR)*. IEEE, 2921–2929.