Communication-efficient View-Pooling for Distributed Multi-View Neural Networks

Manik Singhal, Vijay Raghunathan and Anand Raghunathan
School of Electrical and Computer Engineering, Purdue University
{msingha,vr,raghunathan}@purdue.edu

***Abstract*—Multi-view object detection or the problem of detecting an object using multiple viewpoints, is an important problem in computer vision with varied applications such as distributed smart cameras and collaborative drone swarms. Multi-view object detection algorithms based on deep neural networks (DNNs) achieve high accuracy by *view pooling*, or aggregating features corresponding to the different views. However, when these algorithms are realized on networks of edge devices, the communication cost incurred by view pooling often dominates the overall latency and energy consumption.**

In this paper, we propose techniques for communication-efficient view pooling that can be used to improve the efficiency of distributed multi-view object detection and apply them to state-of-the-art multi-view DNNs. First, we propose *significance-aware feature selection*, which identifies and communicates only those features from each view that are likely to impact the pooled result (and hence, the final output of the DNN). Second, we propose *multi-resolution view pooling*, which divides views into dominant and non-dominant views, and down-scales the features from non-dominant views using an additional network layer before communicating them for pooling. The dominant and non-dominant views are pooled separately and the results are jointly used to derive the final classification. We implement and evaluate the proposed pooling schemes using a model test-bed of twelve Raspberry Pi 3b+ devices and show that they achieve 9X - 36X reduction in data communicated and 1.8X reduction in inference latency, with no degradation in accuracy.

I. Introduction

The great success of Deep Neural Networks (DNNs) in a variety of machine learning tasks and the emergence of low-power DNN processing platforms [1], [2] has prompted interest in deploying them to enable intelligence at the network edge. Many edge intelligence applications, such as distributed smart cameras and drone swarms, rely on synthesizing information from multiple views of the same object to enable more accurate and robust object detection [3]. In recent years, several multi-view object detection algorithms based on DNNs have been proposed [4]–[6]. An integral component of these algorithms is *view pooling*, wherein the intermediate outputs (feature maps) obtained from processing each of the views are pooled (*i.e.,* grouped together) to perform the final classification. This view pooling step, which is key to achieving high accuracy in multi-view object detection algorithms, also becomes a bottleneck when implemented on a distributed network of resource-constrained edge devices. The inputs to the view pooling step, which are typically computed on different edge devices, first need to be transmitted to, and aggregated at, a single location (central server or one of the edge devices) where the subsequent processing and final classification is performed. The high communication cost of this aggregation greatly impacts the latency and energy consumption of multi-view detection. For example, in our implementation of the MVCNN multi-view detection algorithm [5] on a network of 12 Raspberry Pi 3b+ edge devices, we observed that the latency due to communication for view pooling is responsible for 50-90% of the total inference latency. Therefore, there is a crucial need to address this communication bottleneck in the design of distributed multi-view object detection systems.

Prior efforts have attempted to address the communication bottleneck in distributed inference broadly by compressing the data communicated between devices, *e.g.*, by binarizing the communicated features [7] or by applying JPEG encoding [8] to them. We address the communication bottleneck by taking an orthogonal approach that can be used in combination with the above techniques. We observe that the large amount of data communicated in the form of feature maps from each node are immediately reduced by the view pooling operation (typically, max or average pooling in DNNs). We further observe that substantial portions of the transmitted data have little or no impact on the pooled features, and hence the final classification result. Based on this observation, we propose new techniques for communication-efficient view pooling that dramatically reduce the amount of data each node transmits, while maintaining the accuracy of multi-view detection. A recent effort also shares this broad philosophy by turning off entire views based on an estimate of their information content [9]. In contrast to this coarse-grained approach, we propose fine-grained techniques that reduce the volume of data transmitted at a feature level or below. In summary, we make the following contributions:

- We propose a significance-aware feature selection scheme (SAFS) that reduces the amount of data communicated by selectively transmitting only those features from each view that are likely to have an impact on the pooled result.
- We propose multi-resolution view pooling (MRVP), which divides views into dominant and non-dominant ones using an early classification metric [10]. Features from dominant views are transmitted in their original resolution while features from non-dominant views are summarized into a lower resolution. The features from

978-3-9819263-4-7/DATE20/©2020 EDAA

dominant and non-dominant views are pooled separately and concatenated to form inputs to the final classifier.
- Finally, we propose a combined approach that utilizes selective feature transmission but also compensates for the loss in information by concatenating the selected features with a summarized version of the unused features. We refer to this view pooling scheme as enhanced significance-aware feature selection (eSAFS).

We have implemented and evaluated these techniques on a model network of 12 Raspberry Pi 3b+ devices. We used the MVCNN multi-view detection algorithm [5] and the Princeton Modelnet Dataset [4] to evaluate the proposed pooling schemes. Our results indicate that:

- Significance-aware feature selection achieves ˜9x reduction in data communicated with a ˜1.7x reduction in inference latency, while maintaining comparable accuracy ($< 0.5\%$ loss) to the original pooling technique used in MVCNN [5].
- Using multi-resolution view pooling we achieve ˜36x reduction in data communicated with a ˜1.8x reduction in final inference latency but suffer from a ˜2% drop in accuracy.
- Using the enhanced significance-aware feature selection, we achieve a ˜9x reduction in data communicated with a ˜1.7x reduction in final inference latency while achieving *higher accuracy* than *MVCNN* (˜0.5% increase).

The remainder of this paper is organized as follows. In Section II we discuss related work in the field of multi-view object detection algorithms. In Section III and IV we describe and analyze MVCNN and detail our proposed view pooling schemes. In Section V and VI, we discuss our experimental setup and describe our results.

II. Related Work

Su *et al.* [5] proposed the first use of a multi-view convolutional neural network (MVCNN) for object detection. In MVCNN, an intermediate feature representation of each view is first generated using a convolutional neural network. These representations are then pooled using a view-wise max-pooling operation and finally classified using another neural network. If implemented on edge devices, this view-wise pooling technique will result in a large amount of data (Number of views * Number of features per view * Size of each feature) being communicated between devices, incurring substantial latency and energy consumption overheads.

There have been multiple subsequent works that have built upon MVCNN's basic framework, *e.g.*, by building fused representations that carry more information than a single-view representation [6], [11]. These works do not aim to reduce the amount of data being communicated, and can benefit from the techniques proposed herein.

Prior efforts have attempted to address the communication bottleneck in distributed machine learning by compressing the data communicated between devices, *e.g.*, by binarizing the communicated features [7] or by applying JPEG encoding [8] to them, which is complementary to our approach. [12] reduces the communication cost by having a *fixed* sparse network layer when communicating between devices, an approach that is not feasible for the dynamic nature of multi-view applications.

A recent related effort proposed context-aware multi-view convolutional neural networks [9] by exploiting likelihood estimation to decide if a viewpoint's context is important for final classification. This information is used to turn specific viewpoints off, thereby saving on communication time and energy. While our our approach shares the broad philosophy of not transmitting data that is insignificant or unlikely to impact the final result, it differs greatly in how this goal is pursued. Specifically, we adopt more fine-grained approaches to modulate data transmitted at the feature level and below, which results in higher communication savings for similar accuracy.

In summary, multi-view neural networks show great promise in distributed computer vision tasks. However, it is critical to reduce the data communication costs incurred in their implementations on distributed edge-based systems.

III. Background and Motivation

In this section, we describe the view pooling method and network architecture used by MVCNN [5] and analyze the view pooling step to draw insights that motivate our work.

A. Overview of MVCNN

The MVCNN algorithm divides a traditional neural network into two components: a front-end feature extractor ($CNN1$), and a back-end classifier ($CNN2$). Each view (image) is independently passed through $CNN1$ to produce view-specific feature representations. These view-specific feature representations are then pooled to form a fused feature representation. Equation (1) describes this process for the case of max-pooling. If we define the i^{th} feature map of the fused representation as $\mathcal{F}_i$ and the i^{th} feature map of the j^{th} viewpoint as $\mathcal{F}_i^j$, then for each element e in the fused representation's feature map, we have:

$$\mathcal{F}_i[e] = \max_j \mathcal{F}_i^j[e] \tag{1}$$

This fused representation is passed through $CNN2$ to determine a class prediction for the object.

B. Opportunities for Communication-Efficient View Pooling

We analyzed this algorithm by utilizing the Princeton Modelnet Dataset [4] from which we synthesized a 12 view representation of each model as was done in the original MVCNN paper [5]. When calculating the final view-pooled feature representation from Equation 1, we tracked which view was contributing to each element of each feature map. We make two main observations:

- 56% of the elements of each pooled feature map originate from a single node.
- 52% of the elements of the entire set of pooled feature maps come from a single node.

While these numbers suggest that one view may be more important than others, they are not large enough to simply discard other views. Therefore, we propose a fine-grained approach to avoid transmission of data that is unlikely to impact the pooled features.

IV. Communication-Efficient Pooling Schemes

In this section, we describe in detail the proposed techniques for communication-efficient view pooling, which can be realized on a peer-to-peer network of edge devices. We assume that the views are generated by a peer-to-peer network of $\boldsymbol{K}$ edge devices, and that $CNN1$ produces $\boldsymbol{N}$ feature maps, each having $\boldsymbol{m}$ elements.

A. Significance-aware feature selection (SAFS)

Based on the observation in Section III-B, the view-wise max-pooling scheme used in MVCNN can be approximated. Instead of generating a final fused feature map by collecting data from all the viewpoints, we just have to select the most significant viewpoint for each feature map. We find this significant viewpoint by comparing a significance value of corresponding feature maps across the nodes in our network.

To determine these significance values for each feature map, we introduce a global average pooling (GAP) layer after the feature extracting layers of $CNN1$. At every node j, this GAP layer produces a vector $\boldsymbol{s^j}$ of significance values where each element $s^j[i]$ represents the significance for the corresponding feature map $\mathcal{F}^j[i]$.

These significance vectors are then transmitted to and collected (total of K vectors, each of length N) at a node of choice, say node p. These representative vectors are used to calculate a new vector $\boldsymbol{s_{amax}}$ as follows: $s_{amax}[i] = \arg\max_j(s_i^j)$

In other words, for each feature map we select the view with the highest significance. s_{amax} is then broadcast back to all other nodes in the peer-to-peer network. This is the feature selection step as depicted in Figure 1. This step requires communicating a total of $N * K$ elements.

The final fused feature representation, $\boldsymbol{\mathcal{F}}$, is then collected at node p, and it consists of only those features which had the highest relative significance, i.e., each node j transmits all those feature maps i where $s_{amax}[i] = j$. In Figure 1, these chosen features are represented as the colored features after feature selection. The set of pooled feature maps are used as input by $CNN2$ to produce the final classification.

The **SAFS** scheme results in a small accuracy loss compared to the original view pooling scheme (Section III-A). This is because even though the original view-pooling scheme of MVCNN had significant viewpoints that contributed the majority of elements of a feature map, the view-pooled feature map contained some information from non-dominant viewpoints. We present a method to combat this accuracy loss in Section IV-C.

The main benefit of the SAFS scheme is a reduction in the data communicated when compared to MVCNN. The communication cost for MVCNN is a total of $m*N*(K-1)$ elements whereas SAFS incurs a maximum cost of $N * K + m * N$ elements. As the output feature maps of most convolution-based feature extractors are quite large [13]–[15], this results in a net reduction of data communicated by an order of $K-1$.

Note that there is room for further approximation in this scheme. By artificially increasing the significance values for node p's feature maps, we can force node p's features to weigh more in our calculation of s_{amax}. As node p's local features are preferred to features from other nodes, the amount of data being communicated to node p will be reduced. This is studied in Section VI.

B. Multi-resolution view pooling (MRVP)

Our second view pooling scheme finds a dominant viewpoint by drawing inspiration from the techniques used to design early exit networks [10], [16], [17]. Such networks typically feed intermediate features to a fully-connected (FC) layer followed by a softmax layer, whose outputs are used to determine whether to perform an early exit.

We exploit early classification to determine which node(s) in our peer-to-peer network have "dominant" views by comparing each node's belief in the objects being detected. We perform the view pooling step by concatenating the features representing these dominant view(s) with a reduced-resolution version of the features from non-dominant views. As the number of feature elements is high after CNN1, the early classification technique used in [10], [16] would be computationally expensive to use as is. Fortunately, our use case is also less critical (to determine views that should be used in full and reduced resolution, and not to terminate the classification process). Therefore, we devise a lightweight version of the techniques commonly used for early classification.

We use a GAP layer and a small FC layer to scale down the feature maps at each node to a chosen smaller resolution. These chosen resolution feature maps are then transformed using a small FC layer to logits which are passed through a ReLU layer that removes classes with negative beliefs. We empirically found that a simple sum of these positive logits is sufficient for a representative value $\boldsymbol{r}$ for the strength of a viewpoint. After these calculations, for each node j we have the smaller resolution features f^j, the strength of its viewpoint r^j, and the original feature maps $\boldsymbol{\mathcal{F}^j}$.

From each node j, f^j and r^j are then transmitted to a chosen node p. By simply choosing the node with the highest r^j, node p determines the node with the dominant viewpoint (Node d). Node p also calculates a view-pooled smaller resolution representation of all the non-dominant viewpoints: f_{cond}. This pooling is done by either using the average or max of the smaller resolution features from the non-dominant nodes. Node p then transmits to node d the vector f_{cond}.

The MRVP technique incurs a communication cost of $(\mathcal{R}+1)*(K-1)$ elements for transmitting f^j and r^j from each node $j \neq p$, and $\mathcal{R} + K$ elements for broadcasting f_{cond} and the status vector from node p, where $\mathcal{R}$ is the number of elements in the chosen smaller resolution. Overall, this results in a final communication cost of $\mathcal{R} * K + 2 * K - 1$ elements.

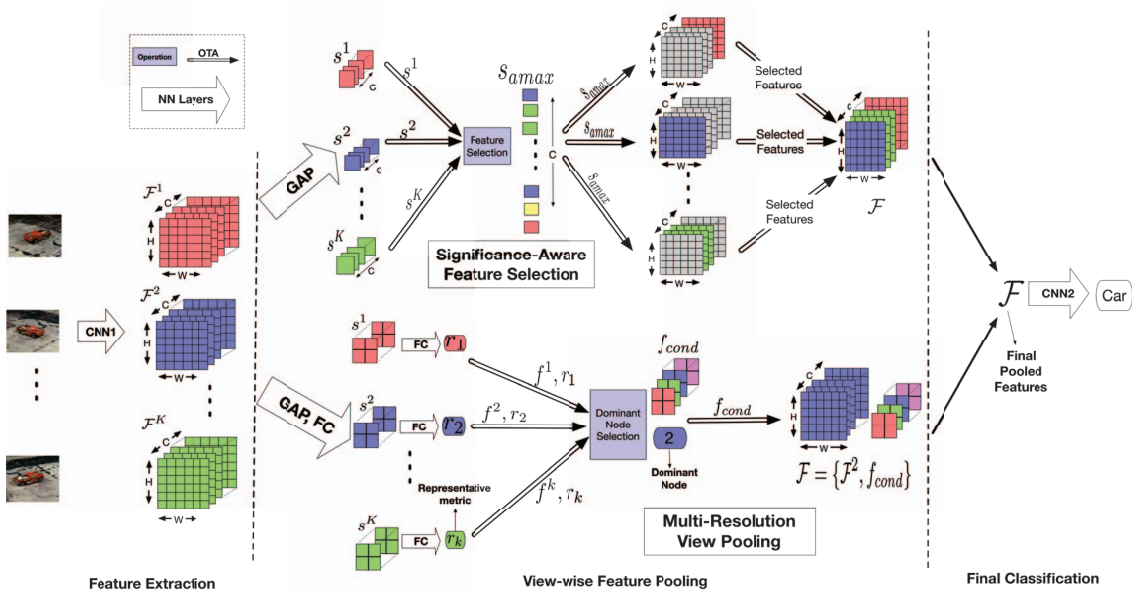

Fig. 1: In **significance-aware feature selection**, each node extracts view-specific feature representations ($\mathcal{F}^i$) and significance vectors (s^i). The significance vectors are transmitted to a single node, which selects the most significant node for each feature. This information is then used to selectively transmit features for final classification. In **Multi-Resolution View Pooling** each node extracts view-specific feature representations $\mathcal{F}^i$, along with a reduced-resolution version f^i and a representative metric r_i, which is used to rank and find the dominant node d. The reduced-resolution summary of all non-dominant nodes (f_{cond}) is transmitted to node d where its concatenated with $\mathcal{F}^d$ and used to perform the final classification.

At node d, the smaller resolution representation, f_{cond} is concatenated with $\boldsymbol{\mathcal{F}^d}$ to form the final pooled feature representation of the multi-view data $\mathcal{F} = \{\mathcal{F}^d, f_{cond}\}$. This final feature representation is passed through $CNN2$ to perform the final classification. $CNN2$ now has a larger model size as the size of its input feature representation has increased.

The key benefit of *MRVP* is a reduction in the volume of data communicated. The communication cost for MVCNN is a total of $m * N * (K-1)$ elements whereas *MRVP* incurs a maximum cost of $\mathcal{R} * K + 2 * K - 1$ elements. This results in a net reduction of data communicated by a factor of roughly $\frac{m*N}{\mathcal{R}}$ (the ratio of the full to the reduced resolution). This ratio represents a knob that can be modulated to tradeoff communication cost for accuracy.

C. Enhanced significance-aware feature selection

As was discussed in Section IV-A, *SAFS* may lose some pertinent information in the final pooled features compared to the original element-wise view pooling scheme of MVCNN. Specifically, elements in lower-significance feature maps that have higher magnitude than the corresponding elements from the most significant feature map are lost. This missing information results in a small accuracy loss (Section VI).

We now present a method that utilizes a technique from *MRVP* (Section IV-B) to overcome this information loss. We view the significance vectors (s^j) themselves as smaller resolution features and combine the significance vectors of all the non-significant features to derive an average vector s_{avg}. This represents a "digest" of those features that we do not collect in the feature transmission step. We simply concatenate s_{avg} with the view-pooled feature representation from *SAFS* to use as inputs for the final classifier.

Compared to *SAFS* this new view-pooling scheme, which we call *eSAFS*, achieves better accuracy without any additional communication costs. We will show in Section VI that this approach outperforms even the original element-wise pooling of MVCNN in terms of accuracy and inference latency. In addition, we conclude that the higher accuracy is not simply due to the increase in model size of $CNN2$.

D. Training

To be able to deploy the proposed pooling schemes in a dynamic environment where different nodes can produce significant features or have dominant views over time, we perform two-step training wherein we first train $CNN1$ (the front-end feature extractor) and then for each pooling scheme train $CNN2$ (the back-end classifier) separately.

V. Experimental Setup

We implement and evaluate our pooling schemes on a representative neural network that has a relatively low inference time on the edge [18], namely Alexnet [13]. Utilizing transfer learning, we take the original model trained on the Imagenet data set and and fine tune the model using our data set. Similar to Su *et al.*'s [5] use of VGG [14], we split Alexnet into two separate networks. The first 5 convolutional layers are chosen as the $CNN1$ and the last 3 fully connected layers are chosen as $CNN2$.

The data set we use is the the Princeton CAD ModelNet data set [4]. Similar to Su *et. al.*, we synthesize data acquisition by taking 12-view images of each object in the Modelnet40 subset of the data set. We ensure that our 12-view 40 class data set is the same one used by [5]. We implemented and analyzed the proposed view pooling schemes using the Pytorch framework [19].

A. Edge device network

We evaluate the proposed pooling schemes on a peer-to-peer network of Raspberry Pi 3b+ devices, each of which has a 1.2 GHz ARM Cortex-A53 cPU with 1GB of RAM, but no native GPU or neural accelerator. Note that the communication cost in the inference latency would be an even larger fraction in devices that specifically target inference at the edge (e.g. Coral TPU, Nvidia Jetson, Intel Movidius, etc) as the computation latency would significantly decrease. We utilise an 802.11ac ad-hoc network between the nodes as our communication fabric with an average measured node-to-node transmission time of 2.05 ms for 1KB.

VI. Results

In this section we analyze the proposed view pooling schemes and discuss their impact on both final inference latency and accuracy.

A. Inference Latency

Figure 2 reports the end-to-end latency for a single inference under various pooling schemes. As we can clearly observe, the communication cost of MVCNN's original view pooling scheme is a very large fraction ($\sim$47%) of the overall inference latency. The proposed pooling schemes greatly cut down the communication costs, resulting in upto $\sim$1.8X lower overall latency.

Both *SAFS* and *eSAFS* have the same communication costs and only differ slightly in their computation costs. The extra computation costs in eSAFS are mostly a result of the extra parameters in CNN2, present in *eSAFS*. Similar overheads are incurred in *MRVP* as well. We observe from Figure 2 that the extra computation cost, marked as *overhead*, is negligible. The *MRVP* variants obtain the lowest communication latency, but as the computation cost of $CNN1$ and $CNN2$ is so high, their overall latency is similar to *SAFS* and *eSAFS*.

B. Accuracy

Table I reports the accuracy for various pooling schemes (for convenience, the inference latency and volume of data communicated are also reported). We can see that *SAFS* incurs a very small drop in accuracy compared to MVCNN (88.75 % vs 88.50%). This is a natural consequence of *SAFS* ignoring the lower significance features when computing the pooled feature representation. On the other hand, *MRVP* incurs a notable 1.75% accuracy drop from the MVCNN baseline. This can be explained by the fact that pertinent information from most viewpoints is lost by choosing a single dominant viewpoint, and the the addition of f_{cond} does not compensate enough. *eSAFS* has a higher accuracy compared to *SAFS* and even the MVCNN baseline. We attribute this improvement to two main effects. First, the concatenation compensates for the information loss that occurs when we approximate the feature pooling step. Second, it gives a weighted importance to features from some viewpoints, thereby going beyond a simple max-pooling scheme.

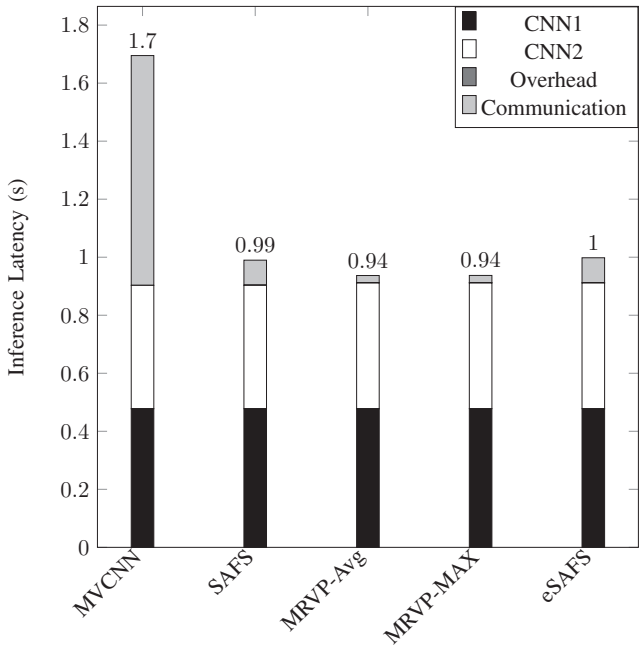

Fig. 2: Inference latency for one input in a 12-view setting. For the MRVP and eSAFS schemes the latency of CNN-2 includes the overhead of the larger model size. MRVP-AVG and MRVP-MAX are variants of MRVP where we use AVG and MAX respectively to calculate the reduced-resolution representation of non-dominant views (f_{cond})

C. Sensitivity analysis

We studied the behavior of *SAFS* and *eSAFS* under further approximations. For this purpose, we varied the strength of node p's features by multiplying the significance vector of node p with an arbitrary constant when we were calculating the vector s_{amax}. This reduced the data that needed to be communicated even further as more of node p's feature maps were chosen for the final feature representation. Figure 3 shows the effect of this approximation on both the volume of data communicated and accuracy. Starting with $p = 1$, even doubling the significance of node p's features does not produce much degradation in accuracy for both *SAFS* and *eSAFS*, whereas it reduces the data communicated by 12.2x over the baseline scheme.

To ensure that the accuracy gains we see in *eSAFS* are not due to just increasing the model size of $CNN2$, we conducted an experiment to analyse the effect of increasing the model size of $CNN2$ used in the view-pooling scheme of MVCNN. We did this by concatenating a condensed representation of

Pooling Scheme	Classification (Accuracy)	Retrieval (mAP)	Inference (Communication) Latency (s)	Data Communicated (KB)
MVCNN (max Pooling)	88.75	79.1%	1.699 (.792)	396
Significance Aware	88.50	80.2%	0.991 (.0858)	42.9
Multi-Resolution - (AVG)	86.88	78.1%	**0.937** (.0246)	**12.3**
Multi-Resolution - (MAX)	87.00	77.8%	**0.937** (.0246)	**12.3**
Enhanced Significance Aware	**89.38**	**82.3**%	0.998 (.0858)	42.9

TABLE I: Comparison of the proposed view pooling schemes. The proposed schemes drastically reduce the communication costs and thereby reduce the final inference latency. We also observe that the accuracies of our proposed schemes are comparable to the original scheme (MVCNN).

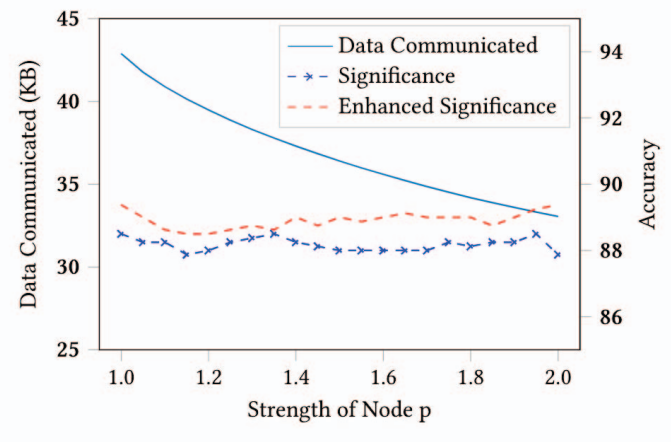

Fig. 3: Data communicated and accuracy *vs.* strength of the feature vector from node p.

all the features to the original pooled features. We then found that increasing the model size does not actually increase the accuracy for the original pooling scheme. We believe this is because all the important information that needs to be collected for the final classification is already present in the traditional pooling schemes and adding the extra condensed information provides no additional benefit.

VII. Conclusions

In this paper, we proposed communication efficient view-pooling schemes to implement multi-view inference on networks of edge devices. These pooling schemes determine the significance of feature maps produced from different views, and use them to determine the feature maps that provide the most information for the back-end classifier. By filtering out less important data, these approaches reduce the communication cost incurred for view pooling significantly (9-36x) and thereby reduce the end-to-end latency by a factor of 1.7x, all while maintaining comparable accuracy to the original pooling scheme.

References

[1] "The Coral dev board takes Googles AI to the edge," *IEEE Spectrum*, April 2019. [Online]. Available: https://spectrum.ieee.org/geek-life/hands-on/the-coral-dev-board-takes-googles-ai-to-the-edge

[2] Intel Movidius Neural Compute Stick. [Online]. Available: https://software.intel.com/en-us/movidius-ncs/

[3] B. Rinner and W. Wolf, "An introduction to distributed smart cameras," *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1565–1575, Oct 2008.

[4] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D shapenets: A deep representation for volumetric shapes," in *Proc. CVPR*, 2015, pp. 1912–1920.

[5] H. Su, et al. "Multi-view convolutional neural networks for 3D shape recognition," in *Proc. ICCV*, 2015.

[6] Y. Feng, Z. Zhang, X. Zhao, R. Ji, and Y. Gao, "GVCNN: Group-view convolutional neural networks for 3D shape recognition," in *Proc. CVPR*, June 2018.

[7] S. Teerapittayanon, B. McDanel, and H. T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *Proc. ICDCS*, June 2017, pp. 328–339.

[8] J. H. Ko, T. Na, M. F. Amir, and S. Mukhopadhyay, "Edge-host partitioning of deep neural networks with feature space encoding for resource-constrained internet-of-things platforms," in *Proc. AVSS*, Nov 2018.

[9] J. Choi, Z. Hakimi, P. W. Shin, J. Sampson, and V. Narayanan, "Context-aware convolutional neural network over distributed system in collaborative computing," in *Proc. DAC*, 2019.

[10] S. Teerapittayanon, B. McDanel, and H. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *Proc. ICPR*, 2016, pp. 2464–2469.

[11] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *Proc. CVPR*, July 2017.

[12] A. Thomas, Y. Guo, Y. Kim, B. Aksanli, A. Kumar, and T. S. Rosing, "Hierarchical and distributed machine learning inference beyond the edge," in *Proc. International Conference on Networking, Sensing and Control (ICNSC)*, May 2019, pp. 18–23.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1097–1105.

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, 2015.

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

[16] P. Panda, A. Sengupta, and K. Roy, "Conditional deep learning for energy-efficient and enhanced pattern recognition," in *Proc. DATE*, March 2016, pp. 475–480.

[17] S. Venkataramani, A. Raghunathan, J. Liu, and M. Shoaib, "Scalable-effort classifiers for energy-efficient machine learning," in *Proc. DAC*, 2015.

[18] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *CoRR*, vol. abs/1605.07678, 2016. [Online]. Available: http://arxiv.org/abs/1605.07678

[19] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *Proc. NIPS Autodiff Workshop*, 2017.