Condar: Context-Aware Distributed Dynamic Object Detection on Radar Data

Ian Harshbarger
University of California Irvine
Irvine, USA
iharshba@uci.edu

Colin Zejda
University of California Irvine
Irvine, USA
czejda@uci.edu

Marco Levorato
University of California Irvine
Irvine, USA
levorato@uci.edu

***Abstract*—Deep neural networks are an established class of algorithms widely used for real-time data analysis in computer vision-based mobile applications. However, on one hand the constrained resources - computing power, energy reservoir and memory - available to mobile devices clash with the complexity of high-performance neural models. On the other hand, offloading the execution of the computing task to a wirelessly connected edge/cloud server requires the transfer of information rich signals over volatile and capacity constrained channels. To address this issue, the Split Computing (SC) paradigm has been proposed. In SC frameworks, a baseline DNN model is used to create two models, namely *head* and *tail* models, that are executed at a mobile device and edge/cloud server, respectively. In this work, a SC paradigm - *Condar* - is presented that introduces several core design and conceptual innovations: (i) we target object detection on radar data rather than 2D RGB images; (ii) we design encoders/decoder structures that are specialized to the specific operating environment; (iii) we develop a multi-branched model that can dynamically select the best performing encoder while executing the main model. Our design is based on the CenterNet model with a split ResNet50 backbone trained on the RADIATE dataset's radar images of resolution 1152x1152. Our results show that Condar has performance analogous to that of a Generalized CenterNet Model with a difference of 4.41%, 1.72%, and 2.18% on mAP@50, mAP@75, and mAP@COCO respectively, while simultaneously performing compression on the features of the data to 15.5KB. This datasize is smaller than lowest possible compression data size of JPEG compression at 21KB, which results in a considerable performance degradation.**

***Index Terms*—Dynamic Deep Neural Networks, Edge Computing, Split Computing, Radar, Object Detection,**

I. Introduction

Deep Neural Networks (DNN) are an established class of algorithms widely used for real-time data analysis in computer vision-based mobile applications. Three predominant paradigms have been proposed to support the execution of these compute-heavy algorithms in mobile computing applications: Edge Computing (EC), Local Computing (LC), and a hybrid approach between the two often referred to as Split Computing (SC). In LC, the mobile devices execute the computing tasks in isolation, whereas in EC the execution is offloaded to compute-capable devices - the edge servers - interconnected to the mobile devices by means of communication links. While the former paradigm suffers from the limited resources available to mobile devices - computing power, energy reservoir and memory - the latter requires the transmission of information rich signals (e.g., 2D images and LiDAR scans) over volatile and capacity constrained wireless channels. Note that EC involves offloading all computational processes to an edge server, whereas LC relies entirely on the client device for computation.

Split Computing, a fusion of these two approaches, leverages the strengths of both edge and client-based computation while reducing the impact from the constraints of each. SC [1] emerged as an intermediate strategy, where the mobile device and edge server share the computing burden. At a high level, the core idea is to divide DNN models into two sections - head and tail models - respectively executed by the mobile device and edge server. The original SC approach in [2] partitions an unaltered DNN model to create the head and tail models, so that the tensor at the partition point is transmitted over the channel. Unfortunately, such a strategy results in highly sub-optimal computing/compression profiles. More recent approaches such as [3]–[5], modify the architecture and training to create optimized compression points whose objective is to minimize the computing load at the mobile device, while achieving high compression gains. At an intuitive level, the core advantage of high-performance SC compared to traditional neural compression is that compression is optimized for the task.

However, some limitations emerge in current state-of-the-art SC for computer vision: (1) SC models are almost exclusively targeted to 2D RGB images; (2) the models are trained over generalized datasets - e.g., [6], [7], and [8] for object detection. The former issue results in a lack of general understanding of whether SC strategies are applicable to more general data structures. The latter issue results in the models failure to leverage characteristics of specific environments, and thus data, to maximize the compression gain, while also limiting their ability to generalize due to the usually consistent set of features in the datasets.

In this work, we introduce *Condar*, a context-aware SC model for object detection on radar data that can seamlessly adapts sub-sections of the models to match specific characteristics of the environment. Remarkably, such capability is embedded in the model itself in the form of a multi-branch model where a classifier stems from the early layers of the

This paper has been partially supported by the National Science Foundation under grants CCF 2140154 and CNS 2134567

model to select how data will be routed across the layers.

Specifically, we make the following contributions:

- We design the first SC model for radar data, and show that the achieved compression gain outperforms widespread compression approaches;
- We demonstrate that context-specific models can achieve an improved performance compared to generalized SC models;
- We design an innovative multi-branched dynamic architecture capable of selecting the optimal encoding/decoding structure while executing the main model;
- We develop a novel three-stage training methodology: (1) Foundation Calibration: Generalized Encoder/Decoder Training, (2) Iterative Condar Tail Optimization, and (3) Contextual Insight Distillation.

Our results show that Condar has performance analogous to that of a Generalized CenterNet Model with a difference of 4.41%, 1.72%, and 2.18% on mAP@50, mAP@75, and mAP@COCO respectively, while simultaneously performing compression on the features of the data to 15.5KB. This data size is smaller than lowest possible compression data size of JPEG compression at 21KB, which results in a considerable performance degradation. These results show that Condar supports high-performance distributed object detection over challenged wireless channels - e.g., low capacity and high variance air-ground channels.

II. Related Work

In this Section, we summarize relevant literature to the approach we propose in this paper.

A. Split Computing

Similar to edge computing, split computing assumes an asymmetric system, where a compute-capable device (e.g., a fixed server, or a large mobile device) assists a resource constrained mobile device in executing complex DNN models. In the early development of split computing, DNN models were simply split into two sub-models, head and tail, composed of unaltered layers of the original network [2]. The mobile device executes the head model, and transmits its tensor output to the edge server, which then executes the tail model. The issue with this approach is that depending on the stage of the network, the intermediate features can be much larger than the original network inputs, as "compression" is achieved only in later layers. In more recent works [3], [4] and [9], split computing (SC) focuses on injecting *bottlenecks* in networks as encoder/decoder-based structures. These are trained to compress and reconstruct the data of a particular layer within the model, then fine-tuning the rest of the model to accommodate the error produced through reconstruction. This structure is then split into two models; the head comprising the encoder, and the tail comprising the decoder and the rest of the original model. The encoder's design aims to compress the data representation of the input as small as possible while preserving important features for the tail model's extraction. This small representation, generated by a client device, can then be efficiently sent over a channel to a more powerful edge device for the larger tail's execution, before sending the results back, allowing for highly efficient computation across devices.

An important concept in the training of these injected bottlenecks found in SC models is knowledge distillation (KD) [10]. The form of KD used in SC is that of an L2 loss. This guides the layer feature reconstruction (model head + decoder) of the student network in the first stage of training to produce similar features between the student's tail layers and the later portions of the teacher model layers while keeping them both frozen.

B. Context-Aware Object Detection

Another concept that is relevant to our design is context-specific object detection. Object detection in adverse conditions, such as extreme weather, has been a major roadblock for efficient perception systems, as observed in scenarios such as those experienced by autonomous vehicles and satellite systems. Previous attempts to solve this issue involve sensor fusion [11] and domain adaptation [12]. The issue with these approaches is that none of them address the core problem: that each context is unique, with characteristics that span spatial, depth, and temporal dimensions.

Current context-aware models are insufficient for handling these complexities. For example, LiDAR sensor performance degrades significantly in heavy fog and rain situations due to signal scattering. In these situations, combining multiple sensor inputs or enhancing sensor robustness fails to fully compensate for the noisy loss of meaningful features. In recent works, the use of adversarial training to enhance the robustness of object detectors is used. This involves generating adversarial examples that mimic adverse conditions and training models to recognize and focus on these perturbations.

The survey in [13] identifies the limitations of the current state of the art context-aware models, explaining that they often lack the adaptability required to effectively manage real-time shifts in weather patterns. More advanced models are required to map contextual information across the multiple dimensions of adverse weather.

III. The Condar Framework

At a high level, Condar is an object detection architecture – and specifically a SC architecture – whose substructures are activated based on the notion of *context*. Considering an overall set of training data $\mathbb{D}$, we assume that subsets of data can be grouped into specific *domains* or *contexts* such that, $\mathbb{D} = \{\mathbb{D}_{C_1}, ..., \mathbb{D}_{C_n}\}$ for n amount of contexts. For instance, in computer vision, context is often defined as illumination and weather conditions, the number of objects and the complexity of structures presents in the data. We assume that the context is classifiable though clear discernible features and/or by some function $f_{cls}(d_{C_i})$ where $d_{C_i} \in \mathbb{D}_{C_i}$. Given the n contexts, context-specific models can be built that outperform a generalized model trained on all data [11].

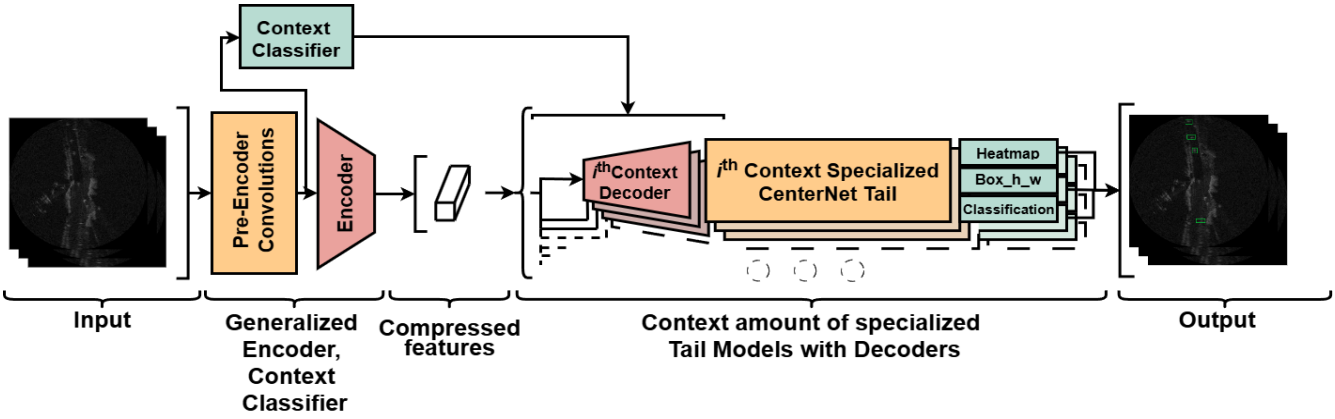

Fig. 1. Architecture of Condar realizing a multi-branched dynamical split DNN. The architecture of the original model is altered to include a decoder/decoder-like structure. Different from traditional SC models, our architecture branch out after the encoder to include context-specific decoder and tail models that are selected by an auxiliary gate model detecting the operating environment.

The general concept we introduce here is that context-specific split computing models can be built using a generalized encoder structure at the head network $f_{gEnc}(d)$ and set of context-specific decoders focused on specific data subsets $F_{dec} = \{f_{dec}^{C_1}(f_{gEnc}(d)), ..., f_{dec}^{C_n}(f_{gEnc}(d))\}$. Here the encoder compresses the features such that each decoder can receive them and reconstruct a pre-selected i^{th} layer $\mathbb{L}_i$ of the context models. The remaining layers after the $\mathbb{L}_i$ for use in Condar we refer to as the tail model(s) $F_{dec} = \{f_{tail}^{C_1}(f_{dec}^{C_1}(f_{gEnc}(d))), ..., f_{tail}^{C_n}(f_{dec}^{C_n}(f_{gEnc}(d)))\}$. We remark how the choice of the layers that are altered to build the encoder/decoder structures within the model is quintessential to balance computational complexity at the mobile device and amount of data transmitted over the channel. The training methodology we propose in this paper leads to extremely compact payload transmitted over the channel with marginal performance loss compared to the original model.

The model, then, is equipped with an internal gate that selects the decoder structure based on the detected context. We embed the detector in the early stages of the model itself to minimize complexity.

A. Model Architecture

The Condar model is expressed as a CenterNet [8] framework with a ResNet50 [14] backbone described in Fig.1. We choose to build a bottleneck after the first `maxpooling` layer in the ResNet50 backbone with a trailing convolutional layer for branching off of to the context classifier head. The bottleneck will replace and reconstruct the features found at the output of the `layer1` set of convolutional layers of the ResNet50 backbone structure. Here, unless specified otherwise, we deploy all convolutions layers we create with a batch-normalization layer and ReLU operation.

Condar consists of one head model comprised of the all layers up until the final encoder layer including the context classification head. The tail of this model consists of n different pairs of decoders and the subsets of layers after the `layer1` of ResNet50 backbone for the n context-specific CenterNet models. The Encoder is composed of convolutional layers with varying strides compressing the feature size of the image sample to be [$\mathbb{B}$, $\mathbb{C}$, $\frac{H}{R}$, $\frac{W}{R}$] where; $\mathbb{B}$ is the batch size, $\mathbb{C}$ is the compressed channel size, H and W are the height and width of the image, R is the reduction ratio based on encoder design and implementation. Further we perform a quantization over the values of the features compressing them to 8-bit integer representations. The Decoder is comprised of a combination of convolutional transpose as up-scaling layers followed by one or more convolutional layers to assist the target layer reconstruction. While the design of the Context classifier head can vary we set it to be comprised of four convolutional layers, an average pooling operation, and linear layer. Through its design, it approximates a trained ResNet50 model on the context classification task. The decoder's input consists of the output from the preceding convolutional layer before the Encoder. The Classification Head detects which context is most likely represented and signals which decoder and tail pair to use.

B. Training Procedures

The training procedure for Condar is structured into three distinct stages, each tailored to optimize different components for efficient context-based object detection. As split-computing utilizes knowledge distillation we pretrain n context specialized models for this training and label them as $\mathbb{M} = M_{C_1}, ..., M_{C_n}$

1) Stage 1: Generalized Encoder and Decoder Training: This initial stage focuses on training the encoder and decoder modules of the CenterNet. We copy and freeze all layers of the tails from $\mathbb{M}$, and train to minimize the reconstruction error of not only the target layer but through select proceeding layers of `layer2`, `layer3`, and `layer4` outputs of the ResNet50 backbone. In addition to these, we also include the final layer of the CenterNet just before the heat-map, class regression, and bounding-box heads. We use a combined mean-squared error (MSE) loss between teacher models output and the reconstructed outputs to learn over. Additionally we utilize

gradient accumulation [15] to combine this loss for each n^{th} context decoder. By using gradient accumulation we are able to increase $\mathbb{B}$ and stabilize the training process.

As a second fine-tuning step in this stage, we freeze the head and encoder structure and further train each context decoder on the combined MSE loss for only their particular context-specific tail.

2) Stage 2: Condar Tail Training: This second stage we select an i^{th} context tail and unfreeze it while simultaneously freezing the head model and all decoders. We then perform a retraining of the tail model's weights by passing the model input through the head model and decoder. The reconstructed features from the decoder are then treated as input for the rest of the tail model, which is retrained for object detection on this new input.

3) Stage 3: Context Head Classifier: Similar to the object detection model, we create a teacher context detector T_c. For this work, T_c will be a base ResNet50 model. As the design of the context head mimics the structure of T_c we use a MSE loss for knowledge distillation. We do the loss between the each of the four sequential convolutional layers of our context head and the output of the sets of layers in ResNet50 referred to as; `layer1`, `layer2`, `layer3`, and `layer4` respectively.

IV. Results

A. Dataset

The RADIATE [16] dataset is a multi-sensor autonomous-driving dataset with a focus on scenario specific contexts of Cities, Traffic junctions, Motorways, Night time driving Raining, Foggy, Rural, and Snowy conditions. Within this dataset exists 39,416 labeled radar images for object detection which we will use in this work. The radar images in this dataset are collected with a Navtech CTS350-X radar set at a range of 100 meters and azimuth sampling of 0.9°

It is important to note that RADIATE does not have an equal distribution of data for each particular context and can be unbalanced in this regard. To stabilize our training processes we pool the datasets by their context labels. For each context we assigned the first 75% of the images for training data in that context and the remaining 25% for validation. In a combined context we pool all splits of 75% as the combined training split, and 25% as the combined validation split. When training we redefine an epoch to be a static 15000 randomly sampled images.

The distribution of images in an epoch will change depending on the training or validation focus context. For any focused context training we randomly sample 75% of the 15000 images to be from that context training split and an equal amount of images from each different context such that they sum to 25% of the 15000 images within an epoch. If there are less images in a context than the 11250 for the focus context or 535 for the non-focus contexts, we over-sample images from that context such that we fully expand the whole context dataset from the context split before oversampling again. For the generalized or combined dataset training we sample 12.5% of the 15000 images for each individual context following the sample principles above. When validating a model we use all data in the combined or a specific context dataset with the Mean Average Precision (mAP) as the performance metric for evaluation. Furthermore to reduce the need for additional model complexity we straighten the bounding boxes of all objects by generating a new bounding box target based on the maximum and minimum x and y pixel positions of existing rotated bounding box.

When training any of our models we transform the images using an Affine Transformation to reduce the image height and width from 1152x1152 to 512x512. We do this to decrease training time and allow us to increase our maximum batch size per training. No other alterations to the images are performed and all model evaluations are done at the original untransformed image size.

B. Encoder and Decoder Specification

A radar image from the RADIATE dataset is of size 1152x1152 in the height and width dimensions. When compressing this to JPEG at 1% we obtain an average minimum sized image over the dataset as 22KB. We design our encoder and decoder structures to achieve a better compression than 22KBs. We implement Condar with a ResNet50 backbone and design the encoder as 4 convolutional layers with kernel sizes of 3 and padding of 1. the last 3 of which have a stride of 2. With the initial layers of the backbone in place and a selected bottleneck channel size of 12, we achieve an $R = 32$ and a resulting feature size of [$\mathbb{B}$, 12, 36, 36]. With quantization, the data size of these features is 15.5KB.

To assist reconstruction of the target layers we implement the decoder with 3 convolutional transpose layers with a kernel size of 3, padding of 0, and stride of 1. We follow these with two regular convolutional layers with the first having a kernel size of 3, the second of 4, and both with a padding of 0 and stride of 1.

1) Hardware: All training of Condar was done on an NVIDIA GeForce RTX 3090 GPU. Execution times of Condar are collected on this GPU and a Jetson ORIN Nano and displayed in Tab.II.

C. Teacher Models and JPEG Compression

For each context we train a CenterNet model for 100 epochs with an Adam Optimizer [17] at learning rate of 5e-4 which we reduce by half at the epochs 55, 75, and 85, and batch size of 40. We then take the best performing model from an evaluation every 5 epochs. Upon training these models we do see a increase in performance of the specialized context models for their particular contexts when compared to the generalized model as seen in Tab. I. We notice a peculiarity of these models however: when evaluating the models with the images compressed at various JPEG ratios we see a gradual performance drop off until a compression ratio of 55% where the models perform better than using the uncompressed image seen in our results in fig.3 and Tab. I only to have a further decrease in performance to 1% JPEG compression.

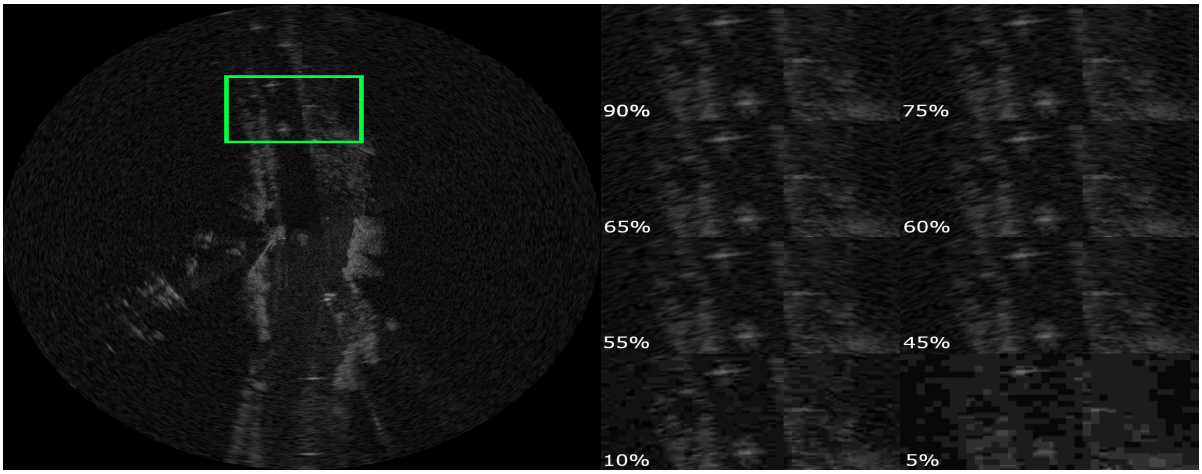

Fig. 2. Impact of JPEG compression on raw radar images. At 55% JPEG compress we observe a noise reduction effect.

TABLE I
PERFORMANCE OF MODELS ON CONTEXT TASKS WITH UNCOMPRESSED IMAGES.
SPEC: CONTEXT SPECIFIC MODEL, CONDAR: CONDAR TAIL MODELS, GEN: GENERIC MODEL
PERFORMANCE IN DESCENDING ORDER; **BLUE**, **GREEN**, **RED**

		Night			**Rain**			**Snow**			**City**		
	mAP(%)	Spec	Condar	Gen	Spec	Condar	Gen	Spec	Condar	Gen	Spec	Condar	Gen
Uncompressed	@50%	49.53	44.63	32.16	60.40	49.63	47.30	63.19	64.42	53.32	28.37	20.05	18.28
	@COCO	23.40	18.15	11.88	27.24	18.45	16.19	23.31	20.77	18.29	10.57	6.78	6.72
	@75%	19.01	11.79	5.05	21.50	8.28	4.65	15.96	7.29	13.93	4.78	2.54	3.15
JPEG 55% Compression	@50%	55.86	52.50	43.64	70.11	56.75	52.44	81.61	67.75	63.01	39.47	26.95	25.04
	@COCO	29.22	24.82	18.84	34.74	25.76	24.92	39.03	29.59	24.72	17.19	10.23	10.53
	@75%	28.17	19.16	10.19	31.64	19.73	19.55	32.76	15.42	13.81	12.55	5.29	6.74

		Junction			**Motorway**			**Rural**			**Fog**		
	mAP(%)	Spec	Condar	Gen	Spec	Condar	Gen	Spec	Condar	Gen	Spec	Condar	Gen
Uncompressed	@50%	45.19	34.19	26.69	48.77	34.30	30.59	68.66	54.00	53.63	69.05	67.12	44.19
	@COCO	18.15	12.46	9.06	17.23	10.68	10.00	25.46	18.68	22.18	26.39	21.85	18.90
	@75%	11.66	5.99	2.86	6.64	2.92	3.82	11.99	6.42	11.68	10.39	8.68	12.40
JPEG 55% Compression	@50%	57.30	35.65	38.50	60.81	41.32	43.26	73.26	65.00	61.82	73.62	70.42	66.26
	@COCO	25.39	14.13	15.40	25.08	14.29	15.11	34.76	27.50	30.77	32.36	27.21	32.45
	@75%	18.73	6.79	8.55	14.93	5.56	6.21	25.75	15.07	29.46	20.23	11.14	23.82

TABLE II
EXECUTION TIMES

	Head(ms)	Tail(ms)	Combined(ms)
RTX3090	2.60	8.29	10.89
ORIN Nano	128.71	300.52	429.23

With further inspection of the images as shown in fig. 2, we can see the noise of the radar get progressively worse until the 55% compression ratio where the compression actually smooths the image out as noise reduction. This effect persists until we see a gradual increase in artifacts due to the low compression ratio seen best at 5%.

D. Context Split Model

When training the head model and decoder in the first stage of training, we train with an Adam Optimizer for 15 epochs at a learning rate of 5e-5 and batch size of 10. We similarly train the tail models for 100 epochs with a initial decreased learning rate of 2e-5, which we reduce by half at the epochs 40, 65, and 85 and a batch size of 40. The results of our training can be seen in Table I.

When comparing the performance of each specialized tail model in relation to the non-split generalized model we see mostly similar performances, if not slightly worse Tab. I. Furthermore, the noise reduction trait of the JPEG compression is still be observed in Condar as also seen in Tab. I.

E. Context Classification Head

For training of the Context Classification Head we first trained a teacher ResNet50 model to classify the context of images in the dataset for 80 epochs at a learning rate of 5e-4 with a batch size of 40. Our teacher model achieves an 96.01% accuracy on the combined validation set. We use this model

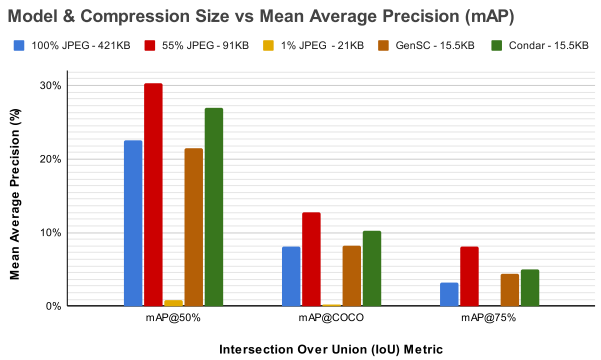

Fig. 3. Compressed Data sizes vs mAP@ of Condar, Generalized Model, and Generalized SC Model. Condar (far right) is outperforming both Generalized Model and Generalized SC model with JPEG compressed images at 55%. We also see the effects of JPEG as de-noising increasing the performance of the generalized model (at 91KB or JPEG 55% Compression).

for knowledge distillation in our Context Classification Head under the same hyper parameters as the teacher and achieve a 90.72% accuracy.

F. Condar vs Generalized model

For evaluation of Condar we first used as imput the radar images at 55% JPEG compression, for the noise reduction performance increase, and then selected the highest scoring index value from the Context Classification Head prediction vector matching the execution with the decoder of that index. We then ran the combined validation dataset though the Condar and achieved a mAP@50% 20.83%, mAP@75% 2.43%, and mAP@COCO% 6.86% for uncompressed images and a mAP@50% 27.01%, mAP@75% 4.95%, and mAP@COCO% 10.23% for JPEG compressed images at 55%. This does not beat the raw mAP performance of the Generalized model at 55% JPEG compression as seen from the results in Tab.I but as shown in Fig.3 Condar is capable of operating below the the JPEG compression limit data size with minimal impact on performance, behaving similarly to the Generalized model. For additional comparison we also generate a SC model [3] based on the Generalized model, with an encoder and decoder designed with same structure as Condar, but with only one decoder and tail model for comparison. Furthermore, we train this SC model with the same hyper-parameters used for the comparable stages of Condar's training. As seen in Fig.3, Condar out performs the generalized SC model performance of mAP@50% 21.51%, mAP@75% 4.39%, and mAP@COCO% 8.21% for JPEG compressed images at 55%.

V. Conclusions

In this work, we presented Condar, a novel split computing framework architecture for radar data. Unique to our design, Condar realizes the first multi-branched split architecture, where the output of the encoder is routed to different decoder/tail models to match the current operating environment. This logic is supported by an auxiliary gate structure that detects features of the data mapped to a specific context.

Condar model is expressed as a CenterNet framework with a ResNet50 backbone trained on the RADIATE dataset. Our results show that Condar has performance analogous to that of a Generalized CeneterNet Model with a difference of 4.41%, 1.72%, and 2.18% on mAP@50, mAP@75, and mAP@COCO respectively, while simultaneously performing compression on the features of the data to 15.5KB. This data size is smaller than lowest possible compression data size of JPEG compression at 21KB, which results in considerable performance degradation.

References

[1] Y. Matsubara, M. Levorato, and F. Restuccia, "Split computing and early exiting for deep learning applications: Survey and research challenges," *ACM Computing Surveys*, vol. 55, no. 5, pp. 1–30, 2022.

[2] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 615–629.

[3] Y. Matsubara, D. Callegaro, S. Singh, M. Levorato, and F. Restuccia, "Bottlefit: Learning compressed representations in deep neural networks for effective and efficient split computing," in *2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Jun. 2022.

[4] F. Cunico, L. Capogrosso, F. Setti, D. Carra, F. Fummi, and M. Cristani, "I-split: Deep network interpretability for split computing," in *2022 26th Int. Conference on Pattern Recognition (ICPR)*. IEEE, Aug. 2022.

[5] A. Bakhtiarnia, N. Milošević, Q. Zhang, D. Bajović, and A. Iosifidis, "Dynamic split computing for efficient deep edge intelligence," 2022, dynamic Neural Networks Workshop (DyNN) at ICML 2022.

[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016, advances in Neural Information Processing Systems 28 (NeurIPS) 2015.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conf. on computer vision and pattern recognition*, 2016, pp. 779–788.

[8] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint triplets for object detection," 2019, iEEE/CVF International Conference on Computer Vision (ICCV 2019).

[9] A. Furtuanpey and et al., "Frankensplit: Efficient neural feature compression with shallow variational bottleneck injection for mobile edge computing," *IEEE Trans. on Mobile Computing*, p. 1–17, 2024.

[10] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[11] A. V. Malawade, T. Mortlock, and M. A. A. Faruque, "Hydrafusion: Context-aware selective sensor fusion for robust and efficient autonomous vehicle perception," *CoRR*, vol. abs/2201.06644, 2022. [Online]. Available: https://arxiv.org/abs/2201.06644

[12] J. Li, R. Xu, J. Ma, Q. Zou, J. Ma, and H. Yu, "Domain adaptation based enhanced detection for autonomous driving in foggy and rainy weather," *ArXiv*, vol. abs/2307.09676, 2023.

[13] Y. Zhang, A. Carballo, H. Yang, and K. Takeda, "Perception and sensing for autonomous vehicles under adverse weather conditions: A survey," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 196, p. 146–177, Feb. 2023.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[15] J. R. Hermans, G. Spanakis, and R. Möckel, "Accumulated gradient normalization," in *Asian Conference on Machine Learning*. PMLR, 2017, pp. 439–454.

[16] M. Sheeny, E. De Pellegrin, S. Mukherjee, A. Ahrabian, S. Wang, and A. Wallace, "Radiate: A radar dataset for automotive perception in bad weather," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 1–7.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.