CAE-Net: Enhanced Converting Autoencoder based Framework for Low-latency Energy-efficient DNN with SLO-constraints

Hasanul Mahmud, Peng Kang, Palden Lama, Kevin Desai and Sushil K. Prasad
Department of Computer Science
University of Texas at San Antonio
San Antonio, Texas
Email: {hasanul.mahmud, peng.kang, palden.lama, kevin.desai, sushil.prasad}@utsa.edu

***Abstract*—As deep neural networks (DNNs) continue to be used on resource-limited edge devices with low latency requirements for interactive applications, there is a growing need to reduce inference time and energy consumption while maintaining acceptable prediction accuracy. In response, we introduce a novel framework, CAE-Net, for designing and training lightweight and energy-efficient deep neural networks (DNNs) for image classification on edge devices. The proposed framework consists of two parts: (1) a new Enhanced Converting Autoencoder that employs entropy-based intraclass clustering to learn the key image features by transforming the hard images into easy representative images, and (2) a composite lightweight CAE-Net classifier employing the pre-trained encoder of the Converting Autoencoder followed by a few classification layers from a baseline DNN trained using knowledge transfer. Unlike many state-of-the-art models, our experimental results using popular image-classification datasets, MNIST and CIFAR10 demonstrate that CAE-Net can satisfy the inference latency target of 10-20ms on Raspberry Pi and 5-10 ms on Nvidia Jetson Nano. Compared with the competing models meeting the SLO targets, CAE-Net achieves over 4-fold energy reduction and inferencing latency speedups on the CIFAR-10 dataset compared to AlexNet and its pruned/distilled variants and other DNNs on Raspberry Pi and about 6-fold on Jetson Nano while maintaining similar or higher accuracy.**

***Index Terms*—Converting Autoencoder, Low latency, Energy efficiency, Edge devices, DNN compression**

I. Introduction

Deep neural networks (DNNs) are increasingly deployed on edge devices for low-latency access to data sources, real-time applications, and data privacy [17]. DNN models deployed on edge devices are often associated with real-time applications with strict latency requirements such as biometric authentication, surveillance, autonomous driving, drone navigation, augmented reality, etc. However, our study shows that state-of-the-art DNN models deployed on edge devices can have up to 10-fold higher inference latency than a typical 10-20 ms service-level-objective (SLO) target of interactive applications [4]. Furthermore, the high energy/power consumption of these models makes it difficult to achieve longer operation times for battery-powered edge devices such as smart wearables, smartphones, autonomous drones, etc., or to maintain reasonable operating temperatures. It is challenging to achieve low-latency and energy-efficient DNN inference on resource-constrained edge devices while maintaining acceptable prediction accuracy [16].

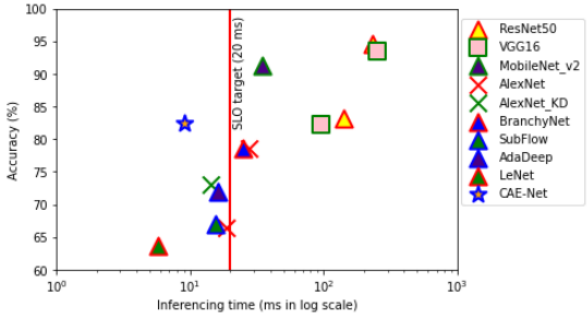


Fig. 1: Performance of the state-of-art DNN models and their aggressively pruned derivatives to meet a SLO target of 20 ms for interactive applications on Raspberry Pi over CIFAR-10 dataset and their resulting loss of accuracy. Our framework meets SLO target and beats its baseline AlexNet in accuracy.

Existing model compression techniques, including pruning, knowledge distillation (KD) [2], [10], and streamlined DNN architectures [11], [15], [18] come at the cost of reduced model accuracy. Figure 2 shows our performance analysis of various DNN models using CIFAR10 dataset on a Raspberry Pi 4. We observe that VGG16 [24], ResNet50 [9],and MobileNet_v2 [23] have an average inference latency of 252.7 ms, 223.4 ms and 77.7 ms respectively. Even with aggressive pruning, these models cannot achieve the SLO target of 20 ms, and they suffer from a significant loss of accuracy. Smaller models, including LeNet [14], aggressively pruned AlexNet, knowledge distillation version of AlexNet (AlexNet-KD) [2], [10], Subflow [15] and AdaDeep [18] meet the SLO target. However, their accuracy is much worse than the baseline AlexNet [13] model. In comparison, our proposed model, CAE-Net, comfortably beats all the smaller models in accuracy and inferencing time.

2024 IEEE Cloud Summit | 979-8-3503-7006-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/CLOUD-SUMMIT61220.2024.00028

979-8-3503-7006-5/24/$31.00 ©2024 IEEE
DOI 10.1109/Cloud-Summit61220.2024.00028

We have recently introduced a concept of a simple "Converting Autoencoder" (CAE) trained to transform images into more easily classifiable representations of their classes [19][1]. Employing an early-exiting DNN such as a BranchyNet [25], CAE identifies easy images as those taking its first exit and all others as hard. We demonstrated that a trained CAE concatenated with a BranchyNet's initial layers leading to its first exit ("CBNet") [19] beats the baseline Lenet, BranchyNet, and others in latency and energy reduction on smaller datasets like MNIST and FMNIST. However, for more complex datasets such as CIFAR10, the random allocation of training targets for Converting Autoencoder-based image transformation and dissimilarities between images within a class result in poor performance. Addressing these limitations, we present CAE-Net, a novel framework with the improved components as in the following key contributions:

1) We propose an enhanced Converting Autoencoder with significantly improved design and training by (a) identifying easy images based on their classification entropy on a baseline network, thus achieving independence from early-exiting DNNs, and (b) employing intraclass clustering for complex datasets such as CIFAR10 to obtain representative easy images within each subclass, as shown in Figure 2, thus reducing reconstruction loss.
2) We propose a lighter-weight design of CAE-Net classifier by concatenating the pre-trained encoder block of the enhanced Converting Autoencoder with a few classification layers from the baseline DNN and better training using knowledge transfer.
3) We evaluate the performance of CAE-Net and compare it with state-of-the-art models on Raspberry Pi 4 and Nvidia Jetson Nano devices in terms of inference latency and energy efficiency (as opposed to their cloud-based emulations as previously carried out). We show it is more suitable for real-time applications on edge devices with low latency constraints.
4) Our comprehensive experiments with two popular image-classification datasets, MNIST and CIFAR10, demonstrate that our method outperforms streamlined network architectures (Subflow [15], AdaDeep [18], MobileNetV2 [23]), early-exit DNN (BranchyNet [25]) and lightweight derivatives of ResNet50 [9], VGG16 [24], AlexNet [13] and LeNet [14]. CAE-Net achieves superior energy efficiency compared to competing models, meeting 10-20ms and 5-10ms of service-level-objective (SLO) targets for inference latency on Raspberry Pi 4 and Nvidia Jetson Nano, respectively. These latency targets are generally acceptable for interactive applications [4].

The remainder of the paper is organized as follows. Section II describes the background and related work. Section III describes the design and implementation details of our framework. Section IV details the testbed setup, experimental results, and quantitative analysis of our proposed method. Section V presents the conclusion and future work.

[1]This work is an extension of an accepted workshop paper at IPDPS'24 (PAISE Workshop).https://arxiv.org/abs/2403.07036. Due to the 6-page limit, we focus on our substantive additional work.

Fig. 2: Images from the airplane class of CIFAR10, which are divided into 3 clusters using the intraclass clustering approach. Each cluster shows one easy image (left) and one hard image (right), which were determined using entropy values of baseline AlexNet.

II. Background and Related work

In this section, we review some of the significant methods that focus on reducing the computational burden of DNN inferencing on resource-constrained devices.

A. Autoencoders

An autoencoder is an artificial neural network that learns efficient encodings of unlabeled data. It consists of an Encoder, which learns how to efficiently encode data into a reduced representation, and a Decoder, which learns how to reconstruct the data back to a representation that is as close to the original input as possible [27]. Autoencoders and their variants are widely used for dimensionality reduction, denoising, data augmentation, and anomaly detection. Unlike conventional autoencoders that learn to reconstruct the same input image, possibly with some noise, we train an autoencoder to convert a hard image into an easier one within the same class to learn the essential features of the image class. We present a synergistic system approach leveraging an autoencoder trained for hard-to-easy image transformation and transfer of its knowledge to yield lightweight and energy-efficient DNNs suitable for edge devices.

B. Early-exit DNNs

An early-exiting framework uses initial neural network layers to accurately classify easy data samples. Teerapittayanon et al. [25] proposed BranchyNet which adds branches at various exit points, allowing confident test samples to exit the network early and improving overall performance by jointly training branches with the original network. In [20], the authors presented the Branch-pruned Conditional Neural Network (BPNet) and its methodology for determining the number and placement of auxiliary classifiers.

Our approach is inspired by early-exit methods. However, a major limitation of existing early-exit DNNs is that they can be inefficient when a significant portion of the dataset consists of hard inputs, which cannot be handled by early exits. We address this issue with a novel approach that enables lightweight DNNs

to efficiently classify both hard and easy images with the help of the knowledge transferred from an autoencoder trained for hard-to-easy image transformation.

C. Model Compression

Model compression techniques can reduce the DNN model's computation and storage. Han et al. [7] proposed "Deep Compression," a three-stage pipeline of pruning, trained quantization, and Huffman coding. Network pruning technique has been widely used to make neural networks lightweight and energy-efficient [1], [30]. Energy-aware pruning was introduced in [29] by removing the layers based on the energy consumption. Initially proposed by Hinton et al. [10], knowledge distillation is a model compression method that distills the knowledge in a large network or ensemble of networks into a small student network by forcing the student's predictions to match those of the teacher. The effectiveness of distillation has been demonstrated in many studies, e.g. [3], [21], using various student and teacher architecture patterns with different depths and widths. Our framework is an alternative approach that can provide lower latency and higher energy efficiency while maintaining similar or higher accuracy.

D. Streamlined DNN Architectures

Several works have proposed methods to streamline DNN architectures for resource-constrained platforms. Adadeep [18] is a usage-driven, automated DNN compression framework that systematically explores the trade-off between performance and resource constraints, and selects the optimal combination of compression techniques and hyperparameters for each layer of a given DNN. SubFlow [15] uses subnetwork pruning to find the optimal subnetworks for each layer that can preserve the accuracy of the original network, and fine-tunes them with a sparsity regularization term. MobileNet [11] and MobileNetV2 [23] are DNN architectures designed for mobile and embedded vision applications, which, respectively, use depth-wise separable convolutions and inverted residual blocks with bottle-necking features to reduce the number of parameters and computations. Experimental evaluation shows that CAE-Net is superior to the competing models in the context of DNN inferencing on edge devices with SLO constraints.

III. CAE-Net Framework Design

A. Enhanced Converting Auto-Encoder

1) CAE Design and Training: In our prior work [19], we introduced a Converting Autoencoder (CAE) focused on expediting the early-exiting DNNs by converting all input images to easily classifiable representative images. This was sufficient for smaller datasets like MNIST and FMNIST, but not for more complex datasets such as CIFAR10 as mentioned earlier. Our primary goal now is to minimize reconstruction errors while efficiently mapping easy representative targets. The representative image within a cluster/subclass refers to the image with the lowest entropy measured by the baseline DNN, such as LeNet and AlexNet. For efficient training of the conversion, we derive our converting autoencoder from a baseline UNet [22] model. Leveraging UNet's inherent encoder and decoder architecture, we've streamline the baseline UNet and designe a lightweight autoencoder that avoids introducing additional overhead. Our autoencoder comprises five convolutional layers and five deconvolutional layers. Following the autoencoder's design, we map input images to their respective easy representative images for each cluster within a class.

Our final objective is to devise a DNN capable of inferring efficiently to meet SLO targets while maintaining good accuracy. To match the target SLO, we uses two sets of pruning techniques during the autoencoder training. The neuron connections are first pruned, making the model sparse and effectively reducing inference latency. Secondly, structural pruning is applied, which aggressively prunes the CAE-Net model, particularly when the SLO target is considerably smaller than the baseline CAE-Net's inference time. We currently perform manual exploration for structural pruning to identify efficient substructures within CAE-Net. In future, we intend to conduct an exhaustive neural architecture search (NAS). We re-train the CAE-Net with a pruned encoder to see if it satisfies the SLO target, else the process is repeated.

2) Intraclass Clustering: Complex datasets such as CIFAR10 can have high dissimilarity among images of the same class, which poses a challenge for hard-to-easy image transformation with autoencoders. It is more difficult and costly to transform a hard image of a fighter jet into an easy image of a commercial airplane than into an easy image of a fighter jet. To tackle this problem, we introduce intraclass clustering that groups images of the same class into different clusters based on their similarity. This facilitates effective training of our autoencoder to perform hard-to-easy image transformation within each cluster/subclass. We used a pre-trained VGG16 model [24] to extract important features of the input image for dimensionality reduction and efficient clustering. The VGG16 model is a CNN model trained on ImageNet [5] for image recognition. The input image had a shape of (32,32,3), which means 32 pixels of width and height and 3 color channels. We removed the last few layers of the VGG16 model, including the fully connected and classification layers, and used the rest as a feature extractor. We passed input images to the feature extractor and obtained a feature map of shape (1, 1, 512) from the 13th layer. This smaller shape was more suitable for clustering, which groups similar data points based on their features. We employed three clustering techniques: K-means, Cosine Similarity, and DBSCAN. For Cosine Similarity, we measured the cosine distance between images and clustered them based on their similarity using various similarity index values. We also utilized DBSCAN and HDBSCAN, fine-tuning hyperparameters such as random state and minimum number of samples. Additionally, we experimented with K-means clustering, selecting different numbers of centroids, which provided

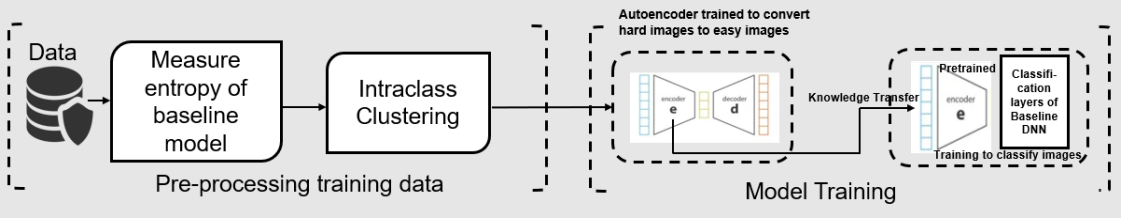


Fig. 3: CAE-Net framework. The final model includes the encoder layers and a thin subnetwork of the baseline model.

the best performance in clustering the data within the same class. Applying K-means clustering to the extracted features improved accuracy by 1% to 3% compared to other clustering techniques.

We used the Elbow method [26] to find the optimal number of clusters for k-means clustering. The elbow point was 3 for CIFAR10 datasets. We evaluated the effectiveness of our converting autoencoder using CIFAR10 dataset with and without intraclass clustering. The autoencoder's performance was evaluated in terms of reconstruction loss, which measures the similarity between the output and the target image. The target image is the easiest image within its cluster for a given input image. As a result of the intraclass clustering, the reconstruction loss of our autoencoder dropped by 46%, from 0.012 to 0.0065.

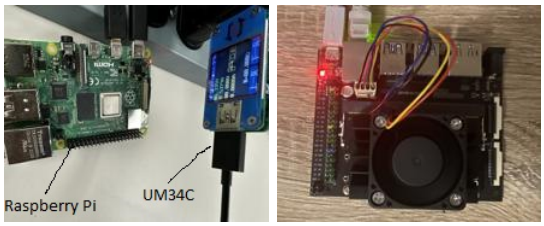
B. Knowledge Transfer from Trained CAE

In our framework, the encoder layers of an autoencoder capture the essential features common to both hard and easy images that can aid in efficient image classification. Therefore, we employ transfer learning to create a new combined DNN model that consists of the pre-trained encoder layers and additional layers obtained through pruning a baseline DNN model. By freezing the pre-trained encoder layers and only training the remaining layers of the network, we utilize the learned representations from the autoencoder and adapt them for image classification. CAE-Net model is used for the classification of MNIST dataset and CIFAR10 dataset based on LeNet and AlexNet respectively. We compared the prediction accuracy of our DNN models trained with and without transfer learning from our pre-trained autoencoder. We observed that transferring the knowledge of our autoencoder improves EncodeNet-LeNet's prediction accuracy on MNIST dataset from **96% to 98.2%**. The transfer learning vastly improves EncodeNet-AlexNet's prediction accuracy on CIFAR10 dataset from **62% to 82.4%**.

IV. Evaluation

A. Experimental Testbeds

Our experiments were conducted on two different edge devices: a Raspberry Pi 4, which is equipped with a Quad-core Cortex-A72 processor and 4GB of RAM, and an Nvidia Jetson Nano, which features a Quad-core ARM Cortex-A57 processor, 4GB of RAM, and an NVIDIA Maxwell GPU. Figure 4 shows our testbed.



(a) Raspberry Pi 4 (b) Jetson Nano

Fig. 4: Experimental test beds

B. Datasets and Evaluation Method

1) Datasets: We evaluated CAE-Net and other competing models on two widely used image-classification datasets: MNIST [6], and CIFAR10 [12]. The MNIST dataset is a large dataset of handwritten digits. It includes a training set of 60K examples and a test set of 10K examples. The CIFAR10 dataset contains 60K 32x32 colored images, of which 50K are used for training and 10K for testing.

2) Evaluation method and competing models: We compared the total number of non-zero parameters, model sizes, accuracy, inference time, and energy efficiency of CAE-Net with state-of-the-art models. The competing models include streamlined network architectures (Subflow [15], AdaDeep [18], MobileNetV2 [23]), early-exit DNN (BranchyNet [25]), ResNet50 [9], ResNet32, ResNet18, VGG16 [24], AlexNet [13] and LeNet [14]. In addition, we evaluated pruned versions VGG16, ResNet50, and lightweight derivatives of AlexNet and LeNet obtained through weight pruning [8], [28], weight clustering [7], and knowledge distillation [10]. The models derived from LeNet using weight pruning, weight clustering and knowledge distillation are denoted as LeNet-WP, LeNet-WC, and LeNet-KD respectively. Similarly, the models derived from AlexNet using these compression techniques are denoted as AlexNet-WP, AlexNet-WC, and AlexNet-KD.

C. Measurement of Inference Time and Energy Consumption

We measured the energy usage, in Joules, as a product of the average power consumption, in Watts, and the inference time, in seconds. All experiments were conducted using an inference

TABLE I: Comparison with the state-of-the-art on CIFAR10 dataset using AlexNet as the baseline.

Methods	Raspberry pi 4 (SLO = 20 ms)		Jetson Nano (SLO = 10 ms)		Parameters (M)	Model Sizes (MB)	Accuracy (%)
	Inference Time [ms] (Speed Up)	Energy [mJ] (Energy ratio)	Inference Time [ms] (Speed Up)	Energy [mJ] (Energy ratio)			
AlexNet [13]	37.1 (1)	154.73 (1)	5.09 (1)	32.1 (1)	9.3	32.97	78.46
AlexNet-KD [10]	9.51 (3.90)	38.11 (4.06)	0.9 (5.7)	**5.21 (6.15)**	**0.55**	**2.22**	73.15
AlexNet-WP [8]	39.05 (0.95)	164.6 (0.94)	5.19 (0.98)	30 (1.07)	5.81	32.97	77.75
AlexNet-WC [7]	39.05 (0.95)	164.6 (0.94)	5.78 (0.88)	36.89 (0.87)	5.81	5.53	80.20
BranchyNet [25]	24.73 (1.5)	N/A	2.12 (2.4)	N/A	9.3	37.3	75
AdaDeep [18]	16.13 (2.3)	N/A	2.82 (1.8)	N/A	3.1	8.1	72
SubFlow [15]	15.45 (2.4)	N/A	1.59 (3.2)	N/A	0.93	3.73	67
CAE-Net	**9.16 (4.05)**	**37.73 (4.1)**	**0.89 (5.7)**	5.30 (6.05)	0.9	3.5	**82.4**
Larger models well beyond SLO latency targets and their derivatives							
VGG16 [24]	252.7 (0.14)	1068 (0.14)	83.9 (0.06)	529.49 (0.07)	37.9	112	93.56
VGG16-pruned (0.4)	135.67 (0.38)	496.08 (0.31)	46.8 (0.11)	320.2 (0.10)	28	61.45	82.34
ResNet50 [9]	223.4 (0.16)	1435 (0.10)	37.6 (0.13)	177.78 (0.18)	25.4	98.2	93.78
ResNet50-pruned (0.4)	140.26 (0.26)	784 (0.20)	22.9 (0.22)	164.61 (0.195)	10	40.69	81.23
ResNet32	161.29 (0.22)	990 (0.15)	27.15(0.20)	226 (0.142)	16	45.6	92.68
ResNet18	155.29 (0.24)	923 (0.16)	29.36 (0.173)	123.46 (0.26)	11	39.52	92.56
MobileNetv2 [23]	77.7 (0.48)	416.47 (0.37)	12.68 (0.40)	91.7 (0.35)	2.2	27.2	91.31

TABLE II: Comparison with the state-of-the-art on MNIST dataset using LeNet as the baseline.

Methods	Raspberry Pi 4 (SLO = 10 ms)		Jetson Nano (SLO = 5 ms)		Parameters (K)	Model Sizes (KB)	Accuracy (%)
	Inference Time [ms] (Speed Up)	Energy [mJ] (Energy ratio)	Inference Time [ms] (Speed Up)	Energy [mJ] (Energy ratio)			
LeNet [14]	5.2 (1)	18.73 (1)	0.47 (1)	1.87 (1)	89.6	328	99.10
LeNet-KD [10]	3.71 (1.4)	12 (1.56)	0.48 (0.98)	**1.78 (1.05)**	12.6	76.1	98.60
LeNet-WP [8]	5.14 (1.01)	18.54 (1.01)	0.48 (0.99)	1.90 (0.98)	56.1	107.9	99.02
LeNet-WC [7]	5.14 (1.01)	18.73 (1)	0.49 (0.97)	1.87 (1)	56.1	52.69	**99.15**
BranchyNet [25]	1.03 (5.4)	N/A	**0.10 (4.7)**	N/A	89.6	1130	98
AdaDeep [18]	2.88 (1.8)	N/A	N/A	N/A	59.7	**63**	97
SubFlow [15]	4 (1.3)	N/A	0.14 (3.2)	N/A	**8.96**	113	97
CAE-Net	**3.37 (1.54)**	**10.52 (1.78)**	0.472 (1)	1.79 (1.04)	110	409	98.20

batch size of one. The power consumption and inference time were measured over five rounds of experiments, and the average values were reported. To measure the inference time on Nvidia Jetson Nano, we used Tensorflow Profiler, which is embedded into Tensorflow's visualization tool, Tensorboard. In the case of Raspberry Pi, we timestamped our source code to measure the inference time since the device did not have sufficient resources for using the Tensorflow Profiler. To measure the power consumption of Raspberry Pi, we used a UM34C USB power meter [2] which was connected to the power source and supplied current to the device as shown in Figure 4 (a). We utilized the rd-usb library [3] to read the logs generated by the power meter from a local server connected via Bluetooth. For Nvidia Jetson Nano, we used the built-in command "tegrastats"[4] to measure the power consumption.

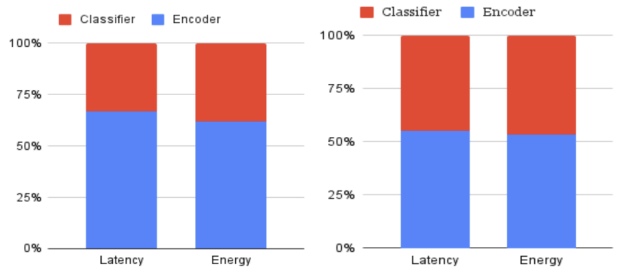
D. Results

Table I presents a comparison of CAE-Net with competing models using CIFAR10 dataset and inference batch size of one. We considered AlexNet [13] model, as the baseline to measure the speedup and energy usage ratio. We observe that CAE-Net achieves a 4x speedup in inference time and a 4.1x reduction in energy usage compared to the baseline AlexNet model on Raspberry Pi. On Nvidia Jetson Nano, CAE-Net archives a 5.7x speedup in inference time and 6x reduction in energy usage. Large models and their variants show high inference latency and energy consumption, significantly exceeding the SLO target of 20 ms for Raspberry Pi 4 and 10 ms for Nvidia Jetson Nano. VGG16 [24], ResNet50 [9], ResNet32, and ResNet18 are examples of such models. Even MobileNet_v2 [23], which has only 2.2 million parameters and 53 layers, fails to meet the SLO target. CAE-Net is the best model among those that meet the SLO targets, with 82.4% accuracy. Notably, CAE-Net achieves 1.7X speedup in inference latency compared to Subflow [15] and AdaDeep [18] models, and 2.7X speedup over BranchyNet [25] on Raspberry Pi 4. Similarly, on Jetson Nano, CAE-Net achieves 1.8X speedup over Subflow [15], 3X speedup over AdaDeep [18], and 2.4X speedup over BranchyNet [25]. Table II compares CAE-Net with competing models using LeNet [14] as the baseline for measuring speedup and energy efficiency. Compared to the baseline, CAE-Net achieves a 1.54x speedup in inference time and a 1.78x reduction in energy usage on Raspberry Pi. CAE-Net did not show substantial speedup over competing models with the MNIST dataset since the baseline inference time is relatively small. Also, the MNIST dataset has a significantly larger proportion of easily classifiable data than hard data. The

[2]UM34C, https://github.com/sebastianha/um34c

[3]rd-usb, https://github.com/kolinger/rd-usb

[4]https://docs.nvidia.com/jetson/archives/r35.2.1/DeveloperGuide/index.html



(a) CAE-Net based on ALexNet (b) CAE-Net based on LeNet

Fig. 5: Per-component breakdown of our model's inference latency and energy usage.

CAE-Net framework is more efficient in reducing inference time when there is an adequate amount of hard data, as the converting autoencoder transforms these difficult data samples into easier ones.

E. Discussion

We analyzed the efficiency of our method by comparing the activation values of the last convolutional layer of AlexNet and CAE-Net during inference. We found that about 90% of activations in AlexNet's last convolutional layer are zero-valued, meaning that only 10% of the computations are useful for inference. On the other hand, our framework based on AlexNet has no zero-valued activations. Thus, it achieves higher accuracy and lower complexity by eliminating unnecessary computations. The inference latency and energy usage, as detailed in Tables I and II, were measured for the complete CAE-Net model, encompassing both the encoder and classifier components. Figure 5 shows the percentage breakdown of inference latency and energy consumption for each element within CAE-Net. We observe that the encoder component can be attributed to a greater portion of the inference latency and energy usage. This is especially true for CAE-Net based on ALexNet. We also assessed the impact of intra-class clustering on the performance of CAE-Net. Intra-class clustering reduces the reconstruction loss of the Converting Autoencoder and is scalable for datasets like CIFAR100 and ImageNet. However, intra-class clustering can introduce additional targets when training on complex datasets such as ImageNet, which contains 1000 different categories. This can place a burden on the Converting Autoencoder (CAE), designed to convert difficult samples into easier ones. Therefore, the design of the autoencoder is crucial for the successful scaling of this approach. The CAE-Net framework is particularly efficient when the datasets contain a higher proportion of hard-to-classify samples, indicating significant potential for further exploration.

F. Limitations of the work

The Converting Autoencoder must first accurately predict the class of an image and then convert the image into the appropriate target. In the CAE-Net framework, the encoder of the converting autoencoder needs to efficiently extract features, while the decoder regenerates an easier target for the given image. If the encoder is not effective, the performance of the converting autoencoder degrades significantly, leading to poor overall performance of CAE-Net. This design approach may limit the classification accuracy of the final model, depending on how efficiently and robustly the converting autoencoder is designed and trained. Using the proposed approach, highly efficient smaller models can be developed with low memory and computation usage. However, further exploration is needed to generalize and scale the framework for large datasets like ImageNet.

V. Conclusion and Future Work

This paper presented CAE-Net, a novel framework for designing and training lightweight and energy-efficient DNNs for resource-limited edge devices. It is based on our enhanced Converting Autoencoder which employs entropy-based intraclass clustering, training for hard-to-easy image transformation. We evaluated our method on two popular datasets, MNIST and CIFAR10, using two resource-constrained edge devices, Raspberry Pi 4 and Nvidia Jetson Nano. Our experimental results demonstrate that it outperforms state-of-the-art techniques in meeting strict latency requirements of real-time applications while providing superior energy efficiency and higher accuracy. In the future, we plan to extend CAE-Net suitable for more complex datasets such as ImageNet. We will also explore the potential integration of the presented framework with existing model compression techniques.

References

[1] A. Aghasi, A. Abdi, N. Nguyen, and J. Romberg. Net-trim: Convex pruning of deep neural networks with performance guarantee. In *Proc. of the Int'l Conference on Neural Information Processing Systems*, 2017.

[2] L. Beyer, X. Zhai, A. Royer, L. Markeeva, R. Anil, and A. Kolesnikov. Knowledge distillation: A good teacher is patient and consistent. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[3] J. H. Cho and B. Hariharan. On the efficacy of knowledge distillation. In *2019 IEEE/CVF Int'l Conference on Computer Vision (ICCV)*, 2019.

[4] D. Crankshaw, X. Wang, G. Zhou, M. J. Franklin, J. E. Gonzalez, and I. Stoica. Clipper: A low-latency online prediction serving system. In *Proc. of NSDI*, 2017.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE conference on computer vision and pattern recognition*, pages 248–255, 2009.

[6] L. Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[7] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *arXiv: Computer Vision and Pattern Recognition*, 2016.

[8] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
[10] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.
[11] 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. *arXiv preprint arXiv:1704.04861*, 2017.
[12] A. Krizhevsky, V. Nair, and G. Hinton. Cifar-10 (canadian institute for advanced research).
[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. of the 25th Int'l Conference on Neural Information Processing Systems - Volume 1*, 2012.
[14] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278–2324, 1998.
[15] S. Lee and S. Nirjon. Subflow: A dynamic induced-subgraph strategy toward real-time dnn inference and training. In *2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2020.
[16] E. Li, L. Zeng, Z. Zhou, and X. Chen. Edge ai: On-demand accelerating deep neural network inference via edge computing. *IEEE Transactions on Wireless Communications*, PP:1–1, 10 2019.
[17] H. Li, K. Ota, and M. Dong. Learning iot in edge: Deep learning for the internet of things with edge computing. *IEEE Network*, 32(1):96–101, 2018.
[18] S. Liu, J. Du, K. Nan, Z. Zhou, A. Wang, and Y. Lin. Adadeep: A usage-driven, automated deep model compression framework for enabling ubiquitous intelligent mobiles. *CoRR*, abs/2006.04432, 2020.
[19] H. Mahmud, P. Kang, K. Desai, P. Lama, and S. Prasad. A converting autoencoder toward low-latency and energy-efficient dnn inference at the edge. *ArXiv - Accepted for publication at the PAISE workshop at IPDPS*, abs/2403.07036, 2024.
[20] K. Park, C. Oh, and Y. Yi. Bpnet: Branch-pruned conditional neural network for systematic time-accuracy tradeoff. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2020.
[21] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. Fitnets: Hints for thin deep nets. *CoRR*, abs/1412.6550, 2014.
[22] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *ArXiv*, abs/1505.04597, 2015.
[23] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
[24] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.
[25] S. Teerapittayanon, B. McDanel, and H. T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. *2016 23rd Int'l Conference on Pattern Recognition (ICPR)*, pages 2464–2469, 2016.
[26] R. Thorndike. Who belongs in the family? *Psychometrika*, 18(4):267–276, 1953.
[27] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.
[28] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning structured sparsity in deep neural networks. *Advances in neural information processing systems*, 29, 2016.
[29] T.-J. Yang, Y.-H. Chen, and V. Sze. Designing energy-efficient convolutional neural networks using energy-aware pruning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
[30] S. Yao, Y. Zhao, A. Zhang, L. Su, and T. Abdelzaher. Deepiot: Compressing deep neural network structures for sensing systems with a compressor-critic framework. In *Proc. of the 15th ACM Conference on Embedded Network Sensor Systems*, SenSys '17, 2017.