



Reducing inference energy consumption using dual complementary CNNs

Michail Kinnas^a, John Violos^{b,*}, Ioannis Kompatsiaris^b, Symeon Papadopoulos^b

^a Department of Informatics and Telematics, Harokopio University of Athens, Omirou 9, Tavros, 177 78, Greece

^b Information Technologies Institute, Centre for Research and Technology, 6th km Harilaou - Thermi, Thessaloniki, 57001, Greece

ARTICLE INFO

Keywords:

On-device AI applications
Convolutional Neural Networks
Energy consumption
Complementarity
Confidence score
Perceptual hash

ABSTRACT

Energy efficiency of Convolutional Neural Networks (CNNs) has become an important area of research, with various strategies being developed to minimize the power consumption of these models. Previous efforts, including techniques like model pruning, quantization, and hardware optimization, have made significant strides in this direction. However, there remains a need for more effective on device AI solutions that balance energy efficiency with model performance. In this paper, we propose a novel approach to reduce the energy requirements of inference of CNNs. Our methodology employs two small Complementary CNNs that collaborate with each other by covering each other's "weaknesses" in predictions. If the confidence for a prediction of the first CNN is considered low, the second CNN is invoked with the aim of producing a higher confidence prediction. This dual-CNN setup significantly reduces energy consumption compared to using a single large deep CNN. Additionally, we propose a memory component that retains previous classifications for identical inputs, bypassing the need to re-invoke the CNNs for the same input, further saving energy. Our experiments on a Jetson Nano computer demonstrate an energy reduction of up to 85.8% achieved on modified datasets where each sample was duplicated once. These findings indicate that leveraging a complementary CNN pair along with a memory component effectively reduces inference energy while maintaining high accuracy.

1. Introduction

Energy efficiency is essential for edge devices in resource-constrained smart environments, where reducing energy consumption is critical for extending device life and lowering maintenance costs [1]. Although techniques like network condition estimation and transmission power adjustment have been explored to improve energy efficiency at the network edge, optimizing deep neural network (DNN) models for on-device AI applications remains underexplored [2]. DNNs have proven to be highly effective for solving complex problems in smart environments, leveraging advancements from various fields, such as computer vision [3]. As Internet of Things (IoT), smart homes, and edge computing devices become more widespread, there is an increasing need to run DNN models directly on edge devices to enable real-time processing and minimize latency [4]. However, deploying large DNNs on these devices presents significant challenges due to their limitations, such as low-power processors and limited memory, making it difficult to manage intensive computations and large storage needs [5]. Given that these devices typically operate on batteries, the high energy demands of DNNs can rapidly drain their power, reducing their operational longevity.

To address these limitations, one strategy is to compress DNN models into smaller ones. Smaller models, with fewer parameters and

layers, require less computational power and energy, making them more suitable for resource-constrained devices. However, their limited capacity can restrict their ability to capture complex patterns and nuances in data, potentially leading to lower accuracy on challenging tasks [6]. In contrast, large DNN models, with more parameters and layers, have a greater capacity to learn intricate features and relationships, often resulting in superior performance on complex problems. This increased complexity, however, demands substantial computational resources and memory [7], posing a significant challenge for deployment on edge devices.

Efforts to mitigate the computational and memory burdens of DNNs have led to numerous works in the area of DNN compression [8]. These aim to reduce the model's computational and memory demands without compromising its performance. However, in practice, these efforts typically involve a trade-off between reducing model size and computational resources against the potential loss in model performance and predictive accuracy [9].

A wide range of smart environment applications, from homes to cities, are driven by advancements in the field of computer vision [10]. In computer vision, several methods have been developed to build efficient CNNs [11]. Quantization reduces the precision of weight

* Corresponding author.

E-mail address: violos@iti.gr (J. Violos).

values to conserve memory. Pruning involves the removal of redundant or insignificant neurons to decrease the computational load [12]. Convolutional filter compression and matrix factorization reduce model size by optimizing the structure of the network. Network Architecture Search (NAS) is another approach that finds DNNs optimized for individual devices, ensuring good performance trade-offs. Additionally, knowledge distillation involves training a smaller *student* neural network to mimic the behavior of a larger *teacher* model, with the goal of maintaining performance while reducing model size [13]. All these approaches have several limitations, including loss of accuracy in the compressed model, inefficient capturing of the teacher model's knowledge, and often considerable computational requirements, increased complexity and time needed for training, fine-tuning, and knowledge distillation [14].

Our goal is to implement a CNN architecture that achieves the performance of a large deep CNN while maintaining the energy efficiency of smaller, more compact models. We explore deploying two small CNN models on resource-constrained devices in order to attain higher predictive accuracy than each model could achieve individually. To this end, we investigate the concept of complementarity. Given a pair of small CNN models trained on the same dataset, complementarity is present if each model correctly predicts distinct subsets of the dataset. This suggests that each model has learned different aspects of the dataset. By leveraging the correct predictions from each model and discarding the incorrect ones, we aim to maximize the coverage of accurate dataset predictions.

To evaluate the validity of a prediction, we employed the confidence score based on the output logits of the model's prediction. This provides a reasonably accurate estimate of prediction validity during inference, while having the least computational and energy requirements than other more advanced methods [15]. By utilizing this confidence score for both deployed models, we select the prediction that is deemed more accurate.

To further reduce energy consumption, we investigate the possibility of bypassing classification by invoking the CNNs only if the same input was not previously encountered. We focus on perceptual hashing [16], a technique that generates a hash value from an image's visual content, enabling similar images to have similar hash values. This method facilitates rapid identification and comparison of images. By storing the hash and classification for each previous prediction, we are able to avoid unnecessary CNN inferences.

While our experiments are conducted on CNN architectures, we also refer to DNNs throughout the manuscript because CNNs are a specific subcategory of DNNs. Our contributions include the following:

- We propose using two complementary CNNs to improve accuracy by addressing each other's prediction weaknesses.
- We introduce a confidence-based mechanism that dynamically selects the more accurate CNN prediction.
- We present a memory system that stores previous classifications, allowing the system to bypass the more energy-intensive CNNs.
- We propose an architecture that integrates memory with inference, activating its components progressively to minimize energy consumption and response time.

The structure of this paper is as follows: Section 2 reviews related work, Section 3 describes our proposed methodology, Section 4 reports our experimental results, and Section 5 presents our conclusions.

2. Related work

The objective of our research is to reduce the energy consumption and response time of DNN inference, making DNN solutions suitable for resource-constrained edge devices without compromising prediction accuracy. As we will show in this section, the literature reveals a gap in solutions that can run on a single edge device, are not

hardware-specific, and do not rely exclusively on compression techniques. Although compression techniques are the go-to approach, they involve a computationally intensive process like knowledge distillation or fine-tuning of pruned models, and they often result in significant performance degradation.

Apart from pruning and knowledge distillation, the other two prominent compression approaches are quantization, and low-rank factorization [8]. These compression methods do not require a computational heavy process but they still result in performance degradation. Recently, researchers have been investigating approaches such as automating the deactivation of DNN layers, offloading computational workloads to other computing nodes, and co-designing hardware and software. We will briefly present these approaches along with their limitations to better contextualize our work within the relevant literature.

Hu et al. [17] present a method aimed at improving the energy efficiency of DNN inference on edge devices by dynamically adapting the network structure in real time. This technique, known as *adaptive DNN surgery*, selectively activates or deactivates certain layers and neurons based on the current computational load and resource availability. By doing so, the network can optimize its inference speed and reduce energy consumption, making it well-suited for energy-constrained edge environments. The dynamic adjustments ensure that only the necessary parts of the network are active at any given time, thus minimizing unnecessary computations and associated energy costs. The limitation of this method is that it is trained for on a specific prediction model and for every new use case the process should be retrained to activate and deactivate the layers and neurons.

Zeng et al. [18] present a method to enhance energy efficiency in DNN inference by offloading tasks across multiple complementary edge devices. CoEdge dynamically distributes the inference workload based on each device's computational capabilities and current energy levels, ensuring optimal resource utilization and minimizing overall energy consumption. Real-time adaptability and dynamic load balancing enable efficient resource allocation. In this context, Adaptive Stochastic Learning Automata can dynamically distribute resources to achieve optimal performance [19]. Additionally, Multi-Agent Deep Learning has been proposed to adjust resource allocation and enhances the performance of processing nodes [20]. These "cooperative" approaches prevent any single device from becoming overburdened, extending the battery life of all devices involved.

Zhang et al. [21] introduce the *coarse-to-fine* method to enhance the efficiency of DNN inference on resource-constrained edge devices. This operates by initially deploying a lightweight, coarse-grained model to make fast, preliminary predictions. When the confidence score, also known as the confidence level, of these initial predictions is sufficiently high, the results are accepted. If the confidence score is low, indicating uncertainty, the system escalates the task to a more complex, fine-grained model for further analysis and refined predictions. This tiered approach ensures that only a fraction of the tasks incur the heavy computational load of the large model, thus optimizing the use of limited edge resources. By performing initial coarse analysis on edge devices and offloading expensive analysis to more powerful servers if needed, coarse-to-fine optimizes resource utilization across the edge network.

While the two above task offloading methods provide improvements in energy requirements they rely on a group of devices on the edge network or to the cloud for the more demanding tasks. Our work surpasses this limitation because it enhances the performance on a single device for all possible inference tasks without burdening other computing devices.

Other studies address the DNN optimization problem through hardware/software co-design. This is an integrated approach that involves jointly designing both the hardware and software components of a system to optimize performance and energy efficiency. Hao et al. [22] introduce a method to improve energy efficiency and performance of

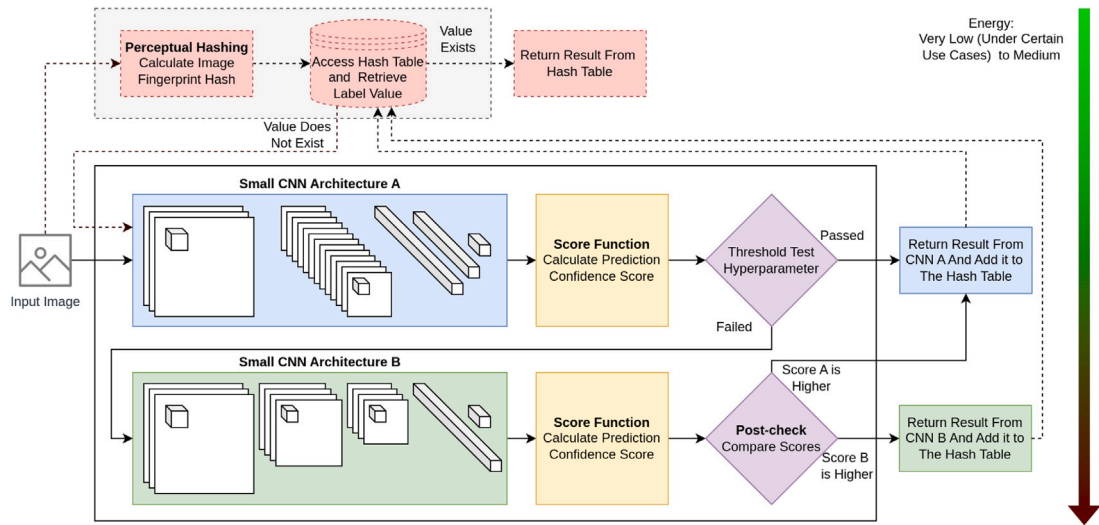


Fig. 1. Overview of our proposed methodology, which consists of two complementary small CNN's with a memory component.

DNNs on edge devices by co-designing both hardware (FPGA) and software DNN. The approach leverages the reconfigurable nature of FPGAs to create customized, energy-efficient hardware accelerators tailored specifically for DNN tasks. It also employs algorithmic techniques such as model compression and quantization to optimize the DNNs for reduced computational load. This co-design ensures that both the hardware and software are highly tuned to work together, significantly lowering energy consumption and enhancing processing speed.

Lee et al. [23] outline a strategy to enhance the energy efficiency of DNN processors on mobile devices through the integrated optimization of hardware and algorithms. By designing specialized, low-power processors and employing algorithmic techniques like model compression, quantization, and pruning, this approach reduces computational demands while maintaining high accuracy. The co-design ensures mutual optimization, significantly cutting down energy consumption and making sophisticated DNN models feasible for real-time, on-device artificial intelligence applications.

The research limitation we address, compared to the two approaches mentioned above, is that we aim to provide a general method for improving the inference energy requirements of DNNs in a hardware-agnostic manner, with minimal model optimizations or modifications. Our goal is to enable rapid deployment on a single device using already trained DNN models.

The closest research to our work was presented by [24]. This employs a dual network architecture that dynamically switches between a “small” CNN for simpler tasks and a “big” CNN for complex tasks based on whether the confidence score of the small CNN falls below a certain threshold. This method significantly reduces energy consumption during inference without sacrificing accuracy. The idea of using two CNN models and selecting the most suitable prediction for a specific input inspired us to explore how two network architectures can collaborate effectively. Our paper distinguishes itself from this work by making further research contributions through complementarity, post-check mechanism, and perceptual hashing. Additionally, a limitation of the work by [24] is that they used a simulator for their experimental evaluation – in particular, they rely on a Verilog-emulated hardware accelerator – whereas we conducted our experiments on an actual edge device and a power meter.

A core concept we use to enable efficient collaboration between the two models is based on the confidence score. Jayakodi et al. [25] introduced a method for calculating a confidence estimate of a model's prediction. Specifically they present a method to balance the accuracy and energy consumption of DNN inference on embedded systems. We

applied this method to decide which DNN will provide the prediction outcome.

For the function of our memory component we explored the use of perceptual hashing. Tang et al. [26] presents a method for generating perceptual hashes for color images based on invariant moments. The proposed technique computes these moments from the color image to create a unique hash value that reflects the image's visual content. We empirically found that these moments are not fully invariant, i.e. they slightly vary under various transformations. A further improvement of the invariant moments is presented in the work of [27], where they implement invariants from complex moments. These complex invariants are fully invariant to transformations, and as such we adopted them in our methodology.

While model compression techniques such as pruning, knowledge distillation, and quantization aim to create a smaller model that replaces a larger one, our approach is fundamentally different. Instead of focusing on compressing a single model, our work emphasizes the effective collaboration between two complementary small models, each with strong accuracy and confidence in distinct parts of the data samples. This approach is more effective than dynamic switching between a small and large model, as the latter often relies on a large model that may not be suitable for edge devices. Additionally, our method incorporates a memory component, which is absent from other techniques in the literature. This memory component works in synergy with the inference process. Our proposed method selectively activates the memory and inference components as needed, optimizing energy consumption and response time. This approach further distinguishes our work from traditional compression methods.

3. Proposed methodology

Our proposed methodology leverages two complementary CNNs to optimize prediction performance while minimizing computational and energy demands. These CNNs operate sequentially: for a given input, the first CNN is initially invoked. The output of the first CNN is evaluated by a prediction confidence score based on the CNN's logits vector. This score is then compared against a threshold, and if it falls short, the second CNN is invoked. The threshold is a predefined value that determines the extent of usage of the second CNN as described in Section 3.4. For the second CNN, a prediction confidence score is similarly calculated using the score function. If both CNNs are utilized for a prediction, their respective confidence scores are compared, and the final decision is based on the more confident prediction.

We also employ a memory component designed to store prior classifications for identical or highly similar inputs. When an input first passes through the memory component, a unique image fingerprint is generated. This fingerprint is used as a key to access a hash table. If the hash table contains a value for the key, this is returned without invoking the CNNs. If no value exists, indicating a new input, the prediction process as described above is initiated. After classification using the CNNs, the classification label is saved to the hash table with the image fingerprint as the key. This component is illustrated in the upper part of Fig. 1, enclosed in a dashed box. Note that the proposed dual-CNN architecture can also operate without this component if it is considered preferable in specific application contexts.

Fig. 1 illustrates an overview of our solution's architecture, which comprises three main components: the small CNN architecture A, the small CNN architecture B, and the memory component. These components are activated based on score comparisons. When an input image arrives, it is first processed by the memory component, specifically the perceptual hashing module, to generate a fingerprint hash of the image. If this fingerprint hash already exists in the access hash table, the corresponding label is retrieved, and the result is returned from the hash table. If the hash is not found, the image is passed to the small CNN architecture A, which makes a prediction and computes a confidence score. If this score exceeds a predefined threshold, the prediction is returned and the hash table is updated. Otherwise, the image is forwarded to the small CNN architecture B, which also generates a prediction and a confidence score. The confidence scores are compared in a post-check, and the prediction with the highest score is returned, while the hash table is updated accordingly.

An important aspect, shown by the energy arrow on the right of Fig. 1, is the varying energy consumption. If the image already exists in the hash table, the energy usage is minimal. When the image is processed by the small CNN architecture A, the energy increases, and it rises further if the small CNN architecture B is also engaged. The following subsections describe each component and the checks of the proposed solution in more detail.

3.1. Two complementary CNNs

Our methodology incorporates two distinct and compact CNN architectures. This strategy capitalizes on the diverse design principles of each model, allowing them to complement each other's deficiencies. In instances of classification ambiguity, the utilization of an alternative CNN mitigates such limitations, given that each neural network captures unique facets of the dataset's information. Opting for two small models, as opposed to a mixture of large and small ones, further minimizes power consumption as we will see in the experimental evaluation.

The concept of complementarity is illustrated in Fig. 2. The gray box represents the whole dataset, of which we consider that the volume outside the circles represent the percentage of labels that are incorrectly predicted by both models and the correct predictions made by a pair of CNNs represented by blue and green circles as subsets of the dataset. The complementarity is function of the symmetric difference between the correct predictions of the two CNNs. The four parts of the Fig. 2 show four examples that help us understand intuitively the idea of complementarity.

In the case of Fig. 2a, both models correctly predict a large portion of the dataset, but their predictions overlap significantly. This substantial intersection indicates low complementarity, as the models frequently predict the same samples. Consequently, this results in redundant resource utilization and increased energy consumption.

Fig. 2b depicts a scenario with high complementarity and extensive correct prediction coverage. However, the size disparity between the models is not optimal, as the larger green model requires higher energy consumption.

Fig. 2c illustrates a fully complementary pair with no overlap in their correct predictions. However, their overall prediction coverage is low, resulting in limited accuracy. Despite their perfect complementarity and equal size, the limited complexity and small size of these models lead to suboptimal accuracy.

Finally, Fig. 2d presents the ideal scenario, where both models have similar size, and together they cover a large portion of the dataset with minimal overlap, thus achieving high complementarity and optimal resource utilization.

We define the complementarity of a pair of CNN models with the formula given in Eq. (1)

$$\text{complementarity}(a, b) = \frac{n(a \cup b) - n(a \cap b) - |n(a) - n(b)|}{N} \quad (1)$$

Given all the correct predictions from a dataset as represented by the set N , then the subsets a and b represent the correct predictions of models a and b respectively.

The first term is defined as:

$$n(a \cup b) = \sum_{i=1}^N \mathbb{I}^1(y_i^a, y_i^b, y_i^{true}) \quad (2)$$

where the indicator function \mathbb{I}^1 is defined as:

$$\mathbb{I}^1(y^a, y^b, y^{true}) = \begin{cases} 1, & \text{if } y^a = y^{true} \\ 1, & \text{if } y^b = y^{true} \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

The second term is defined as:

$$n(a \cap b) = \sum_{i=1}^N \mathbb{I}^2(y_i^a, y_i^b, y_i^{true}) \quad (4)$$

where the indicator function \mathbb{I}^2 is defined as:

$$\mathbb{I}^2(y^a, y^b, y^{true}) = \begin{cases} 1, & \text{if } y^a = y^{true} \text{ and } y^b = y^{true} \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

The third term components are defined as:

$$n(a) = \sum_{i=1}^N \mathbb{I}^3(y_i^a, y_i^{true}) \quad (6)$$

$$n(b) = \sum_{i=1}^N \mathbb{I}^4(y_i^b, y_i^{true}) \quad (7)$$

where the indicator functions \mathbb{I}^3 and \mathbb{I}^4 are defined as:

$$\mathbb{I}^3(y^a, y^{true}) = \begin{cases} 1, & \text{if } y^a = y^{true} \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

and

$$\mathbb{I}^4(y^b, y^{true}) = \begin{cases} 1, & \text{if } y^b = y^{true} \\ 0, & \text{otherwise} \end{cases} \quad (9)$$

respectively.

For the above Eqs. (2), (4), (6) and (7), y_i^a is the prediction from model a , y_i^b is the prediction from model b and y_i^{true} is the true label for a given input i .

The formula (1) consists of three terms in the numerator. The first term, $n(a \cup b)$, represents the total number of correct predictions covered by either of the two models in the given dataset. The objective is to maximize this term to ensure that the combined predictive capability of the models is as comprehensive as possible.

The second term, $n(a \cap b)$, represents the overlap of correct predictions between the two models, meaning, the instances that both models predict correctly. The objective is to minimize this overlap since it is subtracted from the complementarity score. The rationale is that reducing the intersection of correct predictions helps eliminate redundancy, optimizing the use of computational resources. Ideally, the goal is to use smaller models with minimal overlap, ensuring

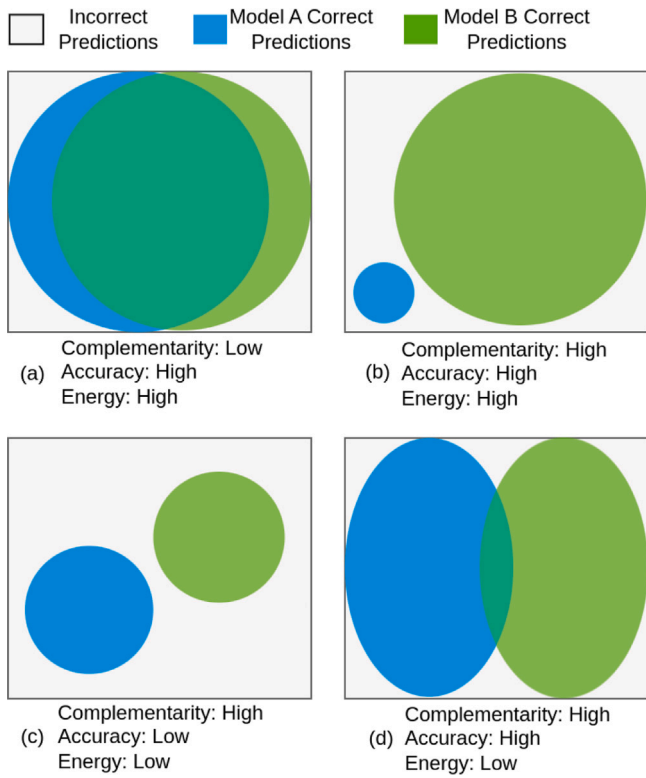


Fig. 2. Complementarity based on the predictions of two CNN models.

they together cover the same prediction space as larger models. The two models become accurate on different data subsets, covering each other's weaknesses without leading to poor generalization. The models' capacity to generalize is preserved because they do not overfit, neither on samples where they exhibit high confidence nor on samples where they exhibit low confidence.

The third term, $|n(a) - n(b)|$, measures the disparity in the sizes of the two models in terms of number of correct predictions. This term should also be minimized. A case where a significant size disparity exists, where one large model may cover the majority of predictions (e.g., 95%), while a much smaller model covers the remainder (e.g., 5%), could also be considered a complementary pair. However, this would lead to inefficient resource usage, as it relies heavily on a large, resource-intensive model. Ideally, the models should be of comparable size, each covering distinct portions of the dataset's predictions to promote balanced and efficient use of computational resources.

3.2. Confidence score functions

CNNs in a classification problem produce a logits vector \vec{z} , which, upon passing through a softmax function (10), is converted into a probability distribution vector \vec{p} .

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^N e^{z_j}} \quad (10)$$

where i and j are the i th and j th element of the logits vector.

Each dimension of this vector corresponds to a class, with the value $\sigma(\vec{z})_i$ indicating the probability that the input belongs to that class. We then operate on this probability vector applying one of three score functions, as presented in [25]:

(a) The Max Probability score function (11) simply selects the highest value from the probability distribution vector. When the CNN is confident in its prediction, this top probability will approach 1.

$$\text{score}(\vec{p}) = \max(\{p_1, \dots, p_n\}) \quad (11)$$

(b) The Difference score function (12) calculates the disparity between the first and second highest values in the output probability vector.

$$\text{score}(\vec{p}) = p_i - p_j \quad (12)$$

where p_i is the largest value in the probability vector \vec{p} and p_j is the second largest value. A larger difference signifies greater confidence in the prediction, as the highest value approaches 1 while the second highest approaches 0. The Difference score function sometimes outperforms the Max Probability in confident predictions, as the top probability may be high, but the second may also hold a small yet significant value.

(c) The Entropy score function (14) computes the entropy of the output probability vector. A lower entropy value corresponds to higher prediction confidence. In theory, lower entropy suggests higher confidence in prediction, indicating that the CNN has confidently assigned a single class. Conversely, higher entropy suggests a more evenly spread distribution, indicating lower confidence.

$$\text{entropy} = - \sum_{i=1}^N p_i \ln(p_i) \quad (13)$$

We employed a normalized version of the standard entropy formula to convert the score values into the range of [0, 1], making them more comparable to the other two score functions:

$$\text{score}(\vec{p}) = \frac{- \sum_{i=1}^N p_i \ln(p_i)}{- \sum_{i=1}^K \frac{i}{K} \ln\left(\frac{i}{K}\right)} \quad (14)$$

where K is the total number of classes of the dataset.

The selection of a score function among the three available options depends on the specific CNN pair under consideration, as the performance of each function can vary across different pairs. Empirically, we have determined that the Difference score function (12) generally yields the best performance in the majority of cases. This observation aligns with the findings reported by [24].

3.3. Score comparison & post-check

The confidence score is utilized in two steps. First, it is employed to compare the score value of the initial CNN against a predetermined threshold. This comparison determines whether to trigger the subsequent CNN. Second, after the invocation of the second CNN, a second confidence score is computed. The two confidence scores are compared, and the prediction of the CNN with the highest (or lowest, if the entropy score function is applied) score is selected. We name this comparison "post-check", indicating an assessment that follows the inference and confidence score of the second CNN. This approach is advantageous as we will see in the experimental evaluation because in some cases the initial CNN may show greater confidence and potentially higher accuracy in its predictions than the second CNN, even if it fails to surpass the threshold test.

3.4. Threshold hyper-parameter

The threshold hyperparameter is a fixed value that determines the extent of usage of the second CNN. By employing lower values (or higher in the case of the entropy score function), the utilization of the second CNN is reduced, thereby decreasing energy consumption but also affecting accuracy. However, by employing a higher threshold value, although the invocation of the second CNN becomes more frequent, this does not necessarily result in higher accuracy, as evidenced by [24].

Our selection of CNN architectures is primarily influenced by the complementary nature of the two CNNs. Since both are nearly equal in size, prediction accuracy, and capabilities, there is no inherent advantage of the second CNN over the first; however, since the first CNN is

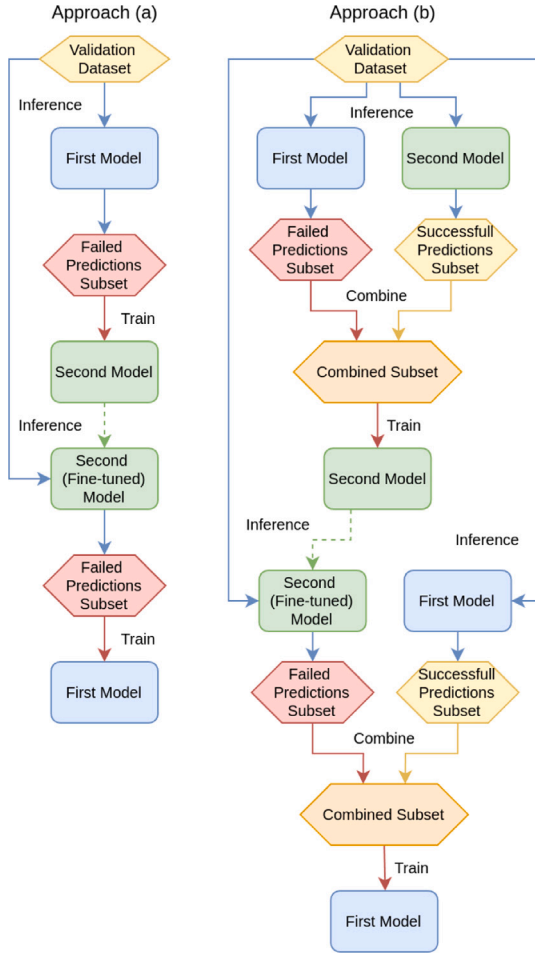


Fig. 3. Fine-tuning approaches to increase complementarity.

used 100% of the time, it is preferable to refrain from using the second as much as possible. Nonetheless, there exists an optimal trade-off that could maximize accuracy. In our methodology, we compute the value that leads to maximum accuracy for the selected CNNs architectures by using the following method in the training dataset.

If we define the achieved accuracy result of our setup, given N samples and a λ value as:

$$acc(\lambda) = \frac{1}{N} \sum_{i=1}^N \begin{cases} 1, & \text{if } H(a(x_i), b(x_i), s, \lambda) = y_i^{true} \\ 0, & \text{otherwise} \end{cases} \quad (15)$$

where the function H represents our methodology setup that returns a prediction \hat{y} in a single inference and $a(x_i)$ and $b(x_i)$ are the predictions of model a and model b respectively of the selected CNN pair given an input x_i , $s = score(\hat{p})$ the selected of the three scoring functions and λ the provided threshold hyper-parameter. The optimal value of λ , denoted as λ^* , is determined as the one that maximizes the accuracy of the selected setup and can be calculated:

$$\lambda^* = \underset{\lambda}{\operatorname{argmax}}(acc(\lambda)), \text{ for } 0 < \lambda < 1 \quad (16)$$

where $acc(\lambda)$ is the Eq. (15).

To determine the optimal threshold hyperparameter, we first select a pair of CNNs exhibiting a high complementarity score based on equation (1) and one of the confidence score functions (Section 3.2). We then apply our methodology iteratively for various λ values within the range $[0, 1]$. This procedure is repeated for each score function, allowing us to identify the optimal combination of score function and λ^* that maximizes the accuracy for the chosen CNN pair.

3.5. Memory component

As an enhancement to our CNN complementary methodology, we incorporate a memory component designed to reduce energy consumption during predictions under specific conditions. This component aims to recall whether a previous classification has been made for a given input, thereby bypassing the need to invoke the CNN when possible. As noted by [7], data movement through memory is a highly energy-intensive process. To address this, we utilized a combination of perceptual hashing [26], which generates a unique fingerprint based on the contents of an image, and a hash table data structure to store and retrieve classification labels for each input. For each input, the required memory access operations are either one read, if the image classification label exists in the hash table, or one read and one write operation, if the input image is new. This approach ensures that energy consumption is kept to a minimum.

According to Fig. 1, every new input image first passes through the memory component, where a hash value is calculated for that image. This is then used as a key to either read or add it, if it does not exist, to a key-value store. The value corresponds to the classification label of the image. If the same image has been classified before, then the read operation using the calculated hash value from the hash table will return the class label, thus skipping the invocation of the CNNs. If no such key exists in the store, then the image is a new input and is thus provided to the CNNs for classification. After the classification, the label, along with the calculated image hash, is added to the key-value store.

To calculate the image hash code, we explored two Perceptual Hashing methods [27]:

1. Difference Hash
2. Invariants from Complex Moments.

The Difference Hash method is the simplest and least computationally intensive approach. It functions by detecting gradients within the image. A bit sequence is computed wherein each bit denotes a change in brightness between adjacent pixels. The setting of each bit is determined by whether the brightness of the left pixel surpasses that of the right one. This sequence is subsequently serving as the image fingerprint and key for the hash map. Although the Difference Hash method offers rapid and energy-efficient hashing, it is susceptible to alterations in the image, including significant fluctuations in brightness and contrast, as well as rotations and mirroring, resulting in distinct calculated fingerprints.

The second method utilizes complex moments of an image to calculate fully rotation-invariant features that remain unchanged despite spatial transformations, as described in [27]. We obtain six invariant features represented as a six-dimensional vector of floating points. All of these values are fully invariant to rotations, and four of them are both fully invariant to rotations and additionally to mirroring in any axis, meaning that even if the image is rotated or mirrored these four values remain the same. We sum these four fully invariant values to receive a single floating-point number that represents the image and is considered the fingerprint, subsequently used as the key in the key-value store, as described above.

3.6. Enhancing complementarity

To enhance the complementarity of a chosen pair, we fine-tune the CNN models. Our objective is to “push” each model toward different areas of the dataset, thereby improving accuracy gains and overall performance. The core idea is to use the failed prediction instances from one model to train the other model for a few epochs, and vice versa. We explore two primary approaches to fine-tuning: (a) using only the instances where predictions failed and (b) using both the failed and successful prediction instances.

In the first approach, shown on the left side of Fig. 3, we select a complementary pair of models and pass the validation dataset through the first model. We then collect all instances where the model makes incorrect predictions into a separate subset. This subset is used to train the second model for a few epochs. After fine-tuning the second model, we reverse the process. We pass the validation dataset through the second, now fine-tuned model, collect the failed predictions into a separate subset, and use this subset to fine-tune the first model.

In the second approach, shown on the right side of Fig. 3, we again select a complementary pair of models and pass the validation dataset through the first model as before. However, this time, we also pass the validation dataset through the second model, capturing the instances where predictions are correct. We then combine the incorrect predictions from the first model with the correct predictions from the second model into a single subset, which is used to train the second model. Next, we reverse the process by passing the validation dataset through both models again. This time, we collect the failed instances from the fine-tuned second model and the correct instances from the first model, combine them into a single subset, and use it to train the first model. This method aims to prevent the overfitting observed in the first approach, which occurs when training on a small subset of data.

4. Experimental evaluation

In this section, we present the experimental evaluation of our proposed methodology. Our objective is to assess the performance and efficiency of our method, utilizing Accuracy, Precision, Recall, and F1 Score for performance evaluation, as well as measuring energy consumption, current, and tail latency.

Sections 4.1, 4.2, and 4.3 detail the experimental edge device, the datasets used, and the evaluation metrics, respectively. The CNN models used in our proposed methodology are described in Section 4.4. In Section 4.6, we summarize the experimental outcomes and discuss the evaluation results.

4.1. Edge device

We conducted our experiments on a Jetson Nano computer with a Quad-core ARM Cortex-A57 MPCore processor, 4GB RAM, and 128 NVIDIA CUDA cores, operating in 5 W mode. The system OS was Ubuntu 20.04.6 LTS, and as the machine learning framework we used PyTorch 1.13.0 with Python 3.6. The experiments' source code is available on GitHub.¹ The Jetson was powered through the USB at 5.15V, and we used a USB power meter capable of measuring milliamps (mAh) and watt-hours (Wh) to the second decimal digit.

4.2. Datasets

We conducted our experiments on CIFAR-10 [28], comprising 10,000 validation images of size 3x32x32 spread across 10 classes, Intel Image Classification,² comprising 3,000 validation images of size 3 × 150 × 150 spread across 6 classes, FashionMNIST [29], comprising 10,000 images of size 1 × 28 × 28 spread across 10 classes and ImageNet [30], comprising 50,000 images of size 3 × 224 × 224 spread across 1,000 classes. More specifically, the ImageNet model architectures are designed to receive 3 × 224 × 224 size inputs, but the actual images in the ImageNet dataset have higher varying widths and heights. The vast majority of these images contain 3 channels (RGB), while a few are grayscale with only 1 channel. The Jetson Nano was unable to load all 50,000 test images into main memory, at their default dimensions, as this would require significantly more than the

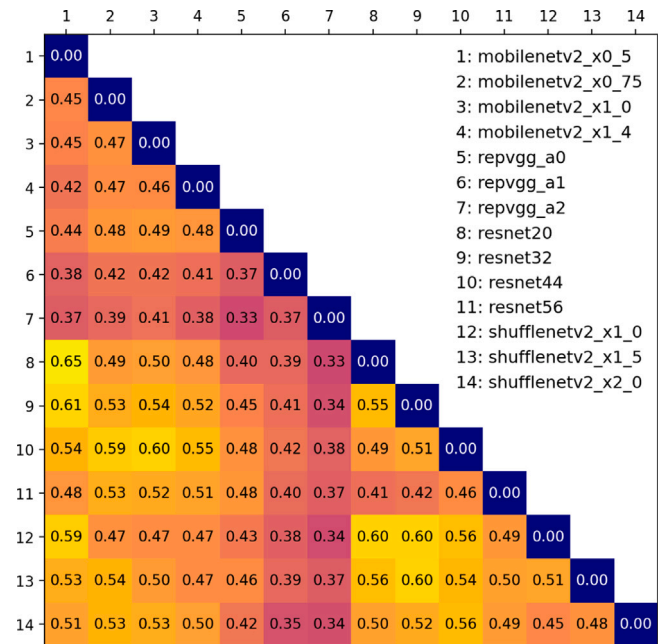


Fig. 4. Complementarity matrix of CIFAR-10 available pretrained PyTorch models.

4GB of available RAM. To address this limitation, we selected a subset of 10,000 images from the test dataset, comprising 10 images from each of the 1,000 classes. We pre-processed these images by resizing them to 224 × 224 pixels and converting grayscale images to 3-channel images.

To determine the optimal hyperparameter λ^* for achieving the highest accuracy, as mentioned in Eq. (16), we can use a validation dataset or a part of the training dataset. For ImageNet, we utilized the additional 40,000 images from the test set as our validation dataset. However, for the CIFAR-10 dataset, which does not have a designated validation dataset, we employed the training set for this purpose. To test the memory component we duplicated our samples for each dataset at different ratios, while also applying simple transformations such as rotations and mirroring on the duplicates, and run the same experiments once using the difference hash method and once using the invariants from complex moment method as described in Section 3.5.

4.3. Evaluation metrics

We measured performance using four metrics: Accuracy, Precision, Recall, and F1-score. Accuracy is defined as the ratio of correctly predicted instances to the total instances. Precision is the ratio of correctly predicted positive instances to the total predicted positives. Recall is the ratio of correctly predicted positive instances to all actual positives. F1-score is the harmonic mean of Precision and Recall, providing a single metric that balances both concerns. Precision, Recall, and F1-score are calculated from the perspective of a specific class, but for our multi-class classification task, we averaged these metrics across all classes. For energy consumption, we measured the current in milliamps (mAh) and energy consumption in watt hours (Wh) used throughout the entire inference process.

We conducted the inference process for each dataset, processing one sample at a time to measure response times. We recorded the mean response time in milliseconds (ms), as well as the 95-th and 99-th percentile tail latencies, defined as the maximum response times that 95% and 99% of the input samples respectively experienced from the time a sample enters the system to when it produces a result. Tail latency [31] is a more important evaluation metric than mean response

¹ <https://github.com/michaelkinnas/Reducing-Inference-Energy-Consumption-Using-Two-Complementary-CNNs>

² <https://www.kaggle.com/datasets/puneet6060/intel-image-classification>

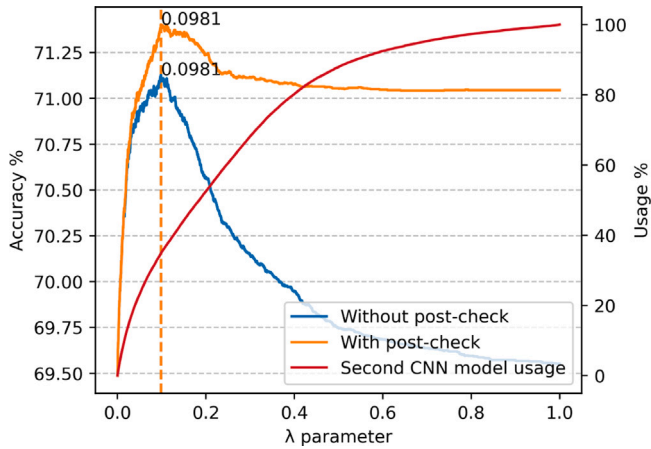


Fig. 5. Post-check Evaluation, using the Difference score function on the ImageNet validation dataset using our configuration I3.

time in latency-sensitive edge computing systems. The reason is that tail latency directly impacts user satisfaction and service level agreements by ensuring that even a small percentage of users do not face unacceptably long delays. It provides a more comprehensive evaluation metric of on-device artificial intelligence applications by highlighting extreme values in the response time distribution, which traditional metrics may overlook.

4.4. CNN models

We utilized benchmark CNN models available from PyTorch Hub.³ Our selection criteria ensured that the largest model could run on the Jetson Nano device while achieving high prediction accuracy, along with its smaller variant. Two models from PyTorch Hub will be used as the basis for our big/little implementation in the comparisons. The large CNN also serves as the single large model in our experimental analysis. Additionally, we included a pruned version of this large model, along with a smaller model trained through knowledge distillation from the large model. This allows us to compare our methodology with established model compression techniques for a comprehensive evaluation. More specifically, with the pruned model, we began with the large baseline model and systematically pruned it to a size comparable to the combined size of the CNN pair used in our methodology, ensuring fair comparison. For the small distilled model, we started with a single, similarly-sized small model and applied a standard knowledge distillation method, using the large model as the teacher. It is important to note that we do not include a distilled model for the ImageNet dataset due to the substantial challenges in training with this large dataset, particularly in terms of time and computational resources. However, we have provided a distilled model for the other three datasets, CIFAR-10, Intel and FashionMNIST.

We did not find any comprehensive list of pretrained models for the Intel and FashionMNIST datasets, therefore, we used the already pretrained models from CIFAR-10 and applied standard transfer learning, fine-tuning all layers for a few epochs.

To identify suitable pairs of CNNs for our proposed methodology, we applied the complementarity formula (1) to the models available in PyTorch Hub, resulting in the complementarity matrix as shown in Fig. 4. This matrix displays all possible combinations of model pairs and their corresponding complementarity scores for the CIFAR-10 dataset.

The highest score was observed between ResNet-20 and MobileNetV2-0.5, making this pair our selection for our methodology. For better illustration, all values are multiplied by 10. The diagonal elements of the matrix have a complementarity score of 0, as identical models produce identical predictions, representing full homogeneity.

We employed a similar process for selecting the model pair for the ImageNet dataset which we do not illustrate for the sake of brevity. After selecting the most suitable models, we evaluated each one individually to establish a baseline. The results are summarized in Table 1.

Table 2 details all the experimental configurations discussed in this paper including a single large model, a pair of a large and a small model and four pairs of small complementary models for every dataset. Each configuration is identified by a prefix letter representing the dataset (CI for CIFAR-10, IM for ImageNet, IN for Intel and FM for FashionMNIST) and a suffix number indicating its sequence in the table. Our proposed methodology configurations are denoted by identifiers CI3 to CI5, IM3 to IM5 and IN3 to IN5 and FM3 to FM5 while the configurations used for comparison are identified as CI1, IM1, IN1 and FM1 for the single large model, CI2, IM2, IN2 and FM2 for the big/little representation, CI6, IM6, IN6 and FM6 for the single large pruned model and CI7, IN7 and FM7 for the small distilled model. We will use this terminology in the Sections 4.6 of outcomes and discussions.

Confidence score functions and threshold values for all experimental configurations using two CNNs were determined by applying the formula (16). The final combination of score function and threshold value was selected based on the maximum achieved accuracy for each configuration.

4.5. Fine-tuning CNN pairs

To implement the fine-tuning methods, we first selected five pairs from the ImageNet dataset, with the highest complementarity using the complementarity matrix. We then applied the fine-tuning approaches outlined in Section 3.6. Once the fine-tuning was completed, we evaluated the fine-tuned pairs using our primary methodology as described in the rest of Section 3. Finally, we compared the accuracy of the fine-tuned pairs against the same models with their default pre-trained weights obtained from PyTorch Hub.

4.6. Results

4.6.1. Post-check ablation

First, we want to examine the application of the post-check mechanism, described in Section 3.3. Fig. 5 shows the accuracy in the validation dataset (left axis) with post-check (orange line) and without post-check (blue line) in the IM3 configuration, which includes the models MnasNet-1.3 and Densenet-121 using the ImageNet dataset. The vertical lines show the selected threshold hyperparameter λ^* , 0.0983 in this case, which achieves the best accuracy, for this specific configuration. By incorporating post-check, we observe a slight enhancement in the total accuracy of the setup. In addition we see the percentage of usage (right axis) of the second model (red line) as a function of the λ hyperparameter. As the value of λ increases, the usage of the second model also increases.

4.6.2. λ sensitivity analysis

We examine the relationship between accuracy and energy consumption as a function of the λ parameter. As shown in Fig. 9,

³ <https://pytorch.org/hub/>

Table 1
Baseline measurements of performance and efficiency metrics on our selection of CNN models.

Dataset	Model	N. Parameters (M.)	Response times (ms)			Current (mAh)	Power (Wh)	Accuracy (%)
			mean	95th	99th			
CIFAR-10	Resnet 20	0.27	22.3	23.3	23.8	48	0.25	92.60
CIFAR-10	MobileNetV2-0.5	0.70	42.4	43.8	44.3	81	0.41	93.12
CIFAR-10	RepVGG-A0	7.84	51.9	53.8	54.4	125	0.64	94.46
CIFAR-10	RepVGG-A2	26.82	64.8	65.3	67.0	225	1.20	95.27
CIFAR-10	ShuffleNet V2 X-1.0 (distilled)	1.26	53.0	54.4	56.1	40	0.75	77.36
ImageNet	MnasNet-1.3	6.3	46.6	47.2	85.1	165	0.88	69.17
ImageNet	DenseNet-121	7.9	131.0	141.1	149.0	398	2.12	69.65
ImageNet	RegNet-X-800MF	7.3	118.4	127.7	133.1	283	1.50	67.89
ImageNet	RegNet-X-8GF	39.6	235.1	237.8	238.7	845	4.50	73.11
Intel	MobileNetV2-X1.0	2.24	43.6	45.3	52.8	0.13	24	86.70
Intel	Resnet 44	0.66	44.6	47.2	48.2	0.11	20	85.60
Intel	RepVGG-A0	7.84	54.4	55.2	56.3	0.16	30	81.83
Intel	RepVGG-A2	26.82	65.0	65.5	87.6	0.28	54	88.03
Intel	ShufflenetV2-X1.5 (distilled)	2.48	49.3	54.0	55.6	0.12	23	82.20
FashionMNIST	MobileNetV2-X1.0	2.24	44.8	46.8	47.1	0.34	65	86.86
FashionMNIST	Resnet 44	0.66	43.5	45.2	45.5	0.38	72	86.59
FashionMNIST	RepVGG-A0	7.84	54.0	54.6	55.3	0.52	99	88.88
FashionMNIST	RepVGG-A2	26.82	64.8	65.4	68.6	0.93	176	90.00
FashionMNIST	ShufflenetV2-X1.5 (distilled)	2.48	52.0	53.0	55.3	0.40	76	87.76

Table 2
Experimental configurations.

Identifier	Description	First model	Second model	Score function	Threshold value	Memory
CI1	Single large model	RepVGG-A2	–	–	–	–
CI2	Big/little representation	RepVGG-A0	RepVGG-A2	Difference	0.99	–
CI3	Small complementary pair	Resnet 20	MobileNetV2-0.5	Difference	0.8724	No
CI4	Small complementary pair	Resnet 20	MobileNetV2-0.5	Difference	0.8724	DHash
CI5	Small complementary pair	Resnet 20	MobileNetV2-0.5	Difference	0.8724	Invariants
CI6	Single large model (pruned)	RepVGG-A2	–	–	–	–
CI7	Single small model (distilled)	ShuffleNet V2 x1.5	–	–	–	–
IM1	Single large model	RegNet-X-8GF	–	–	–	–
IM2	Big/little representation	RegNet-X-800MF	RegNet-X-8GF	Difference	0.9074	–
IM3	Small complementary pair	MnasNet-1.3	Densenet-121	Difference	0.0983	No
IM4	Small complementary pair	MnasNet-1.3	Densenet-121	Difference	0.0983	DHash
IM5	Small complementary pair	MnasNet-1.3	Densenet-121	Difference	0.0983	Invariants
IM6	Single large model (pruned)	RegNet-X-8GF	–	–	–	–
IN1	Single large model	RepVGG-A2	–	–	–	–
IN2	Big/little representation	RepVGG-A0	RepVGG-A2	Difference	0.6	–
IN3	Small complementary pair	MobileNetV2-1.0	Resnet 44	Difference	0.2166	No
IN4	Small complementary pair	MobileNetV2-1.0	Resnet 44	Difference	0.2166	DHash
IN5	Small complementary pair	MobileNetV2-1.0	Resnet 44	Difference	0.2166	Invariants
IN6	Single large model (pruned)	RepVGG-A2	–	–	–	–
IN7	Single small model (distilled)	ShuffleNet V2 x1.5	–	–	–	–
FM1	Single large model	RepVGG-A2	–	–	–	–
FM2	Big/little representation	RepVGG-A0	RepVGG-A2	Difference	0.9	–
FM3	Small complementary pair	Resnet 44	MobileNetV2-1.0	Max Probability	0.7917	No
FM4	Small complementary pair	Resnet 44	MobileNetV2-1.0	Max Probability	0.7917	DHash
FM5	Small complementary pair	Resnet 44	MobileNetV2-1.0	Max Probability	0.7917	Invariants
FM6	Single large model (pruned)	RepVGG-A2	–	–	–	–
FM7	Single small model (distilled)	ShuffleNet V2 x1.5	–	–	–	–

increasing the λ parameter tightens the acceptance threshold for predictions from the first model. As a result, the second model is invoked more frequently, leading to higher energy consumption. In this figure, the blue curve represents the accuracy as λ varies, with its corresponding y-axis on the left, while the red curve depicts the total energy consumed during inference on the test set, with its y-axis on the right. When λ approaches 0, the first model predominantly generates the predictions, resulting in lower overall accuracy, similar to the performance of using only the first model, as only its predictions are accepted. As λ increases, the second model is invoked more frequently. Additionally, the figure shows that accuracy remains largely stable, except for very small values of λ . This stability is maintained by the Post-check mechanism, which ensures that the most reliable prediction is selected for each input. Without this mechanism, as shown by the blue curve in Fig. 5, accuracy declines because the second model's predictions are accepted unconditionally when the confidence threshold of the first model is not met.

4.6.3. Score functions comparison

To gain a deeper understanding of the selection process among the three available score functions, we present a comparative performance analysis using the λ^* parameter search function (Eq. (16)). Two additional figures are included to illustrate the Post-check evaluation for the Max Probability score function (Eq. (11)) in Fig. 10 and for the Entropy score function (Eq. (14)) in Fig. 11. These are compared alongside the previously discussed Difference score function (Eq. (12)) shown in Fig. 5.

From the comparison of these figures, it is evident that for our specific ImageNet configuration (IM3), utilizing the pair of CNNs MnasNet-1.3 and Densenet-121, the best performance is achieved with the Difference score function at $\lambda^* = 0.0981$, yielding the highest accuracy on the validation dataset. Notably, the Entropy score function performs the worst for this particular CNN pair. Additionally, it is observed that in all three cases, the use of the Post-check mechanism consistently results in either improved or, at minimum, comparable performance across the full range of λ values.

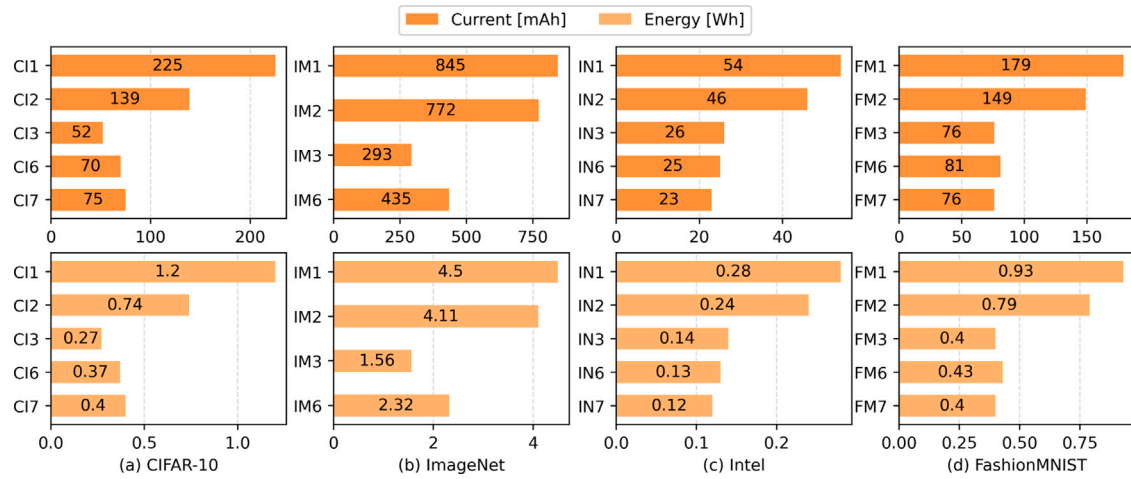


Fig. 6. Current and Energy consumption measurements of tested configurations for (a) CIFAR-10, (b) ImageNet, (c) Intel and (d) FashionMNIST datasets.

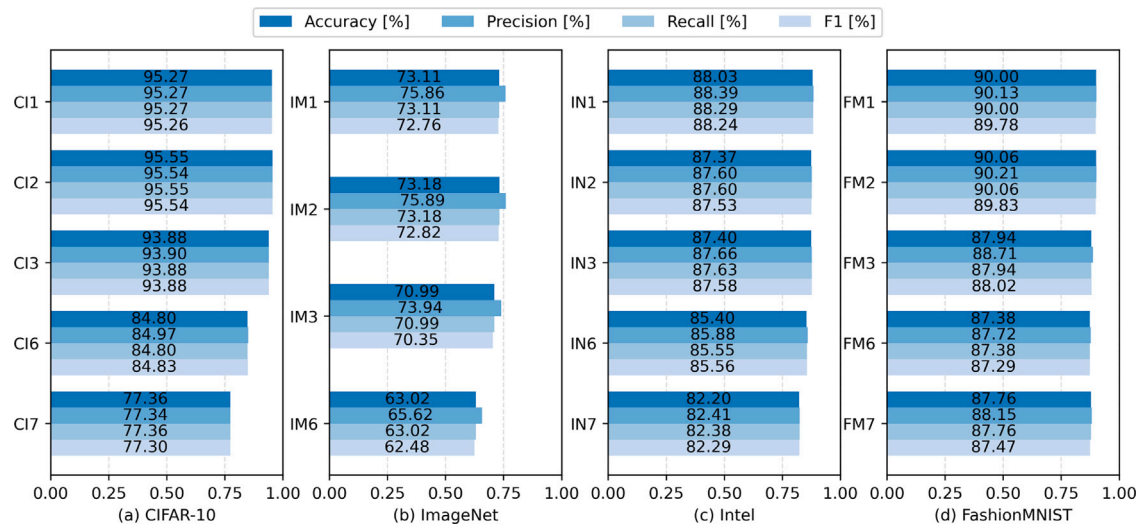


Fig. 7. Performance of tested configurations for (a) CIFAR-10, (b) ImageNet, (c) Intel and (d) FashionMNIST datasets.

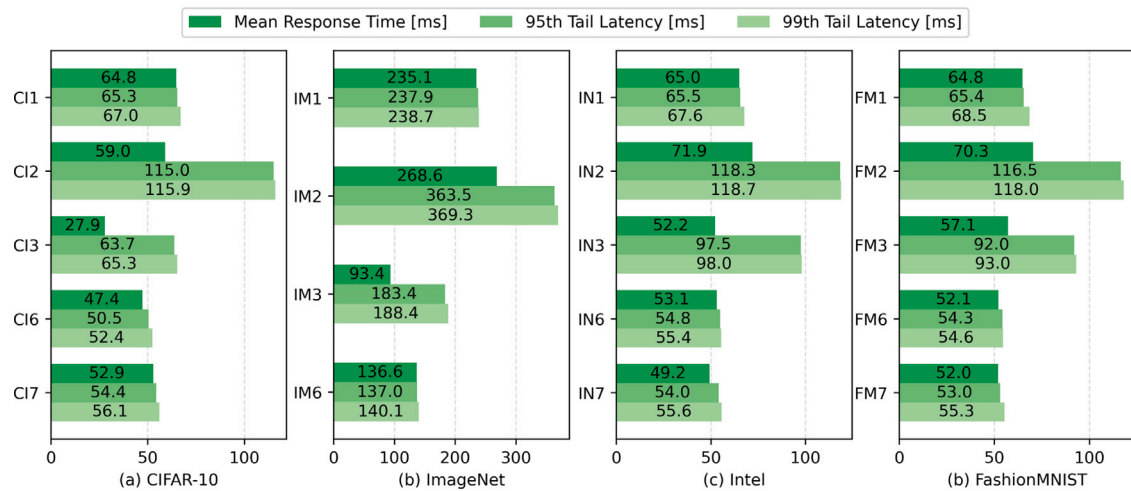


Fig. 8. Response times of tested configurations for (a) CIFAR-10, (b) ImageNet, (c) Intel and (d) FashionMNIST datasets.

4.6.4. Energy evaluation

Fig. 6 presents a comparison of energy consumption across different experimental configurations (Table 2) for the CIFAR-10, ImageNet, Intel and FashionMNIST datasets. For the CIFAR-10 dataset, our configuration (CI3) demonstrates a substantial reduction in energy consumption, achieving a 62.6% decrease compared to the big/little configuration (CI2) and a 76.9% decrease relative to the single large model (CI1). Additionally, it reduces energy consumption by 25.6% compared to the large pruned model (CI6) and by 30.7% when compared to the small distilled model (CI7).

A similar trend is observed with the ImageNet dataset. Our configuration (IM3) reduces energy consumption by 62% relative to the big/little configuration (IM2) and by 65.3% when compared to the single large model (IM1). In comparison to the large pruned model (IM6), energy consumption is decreased by 32.6%.

For the Intel dataset, our configuration (IN3) achieves a 43.5% reduction in energy consumption compared to the big/little configuration (IN2) and a 52.9% reduction relative to the single large model (IN1). However, it shows a slight 4% increase in energy consumption compared to the large pruned model (IN6) and a 13% increase compared to the small distilled model (IN7).

Lastly, for the FashionMNIST dataset, our configuration (FM3) results in a 49% decrease in energy consumption compared to the big/little configuration (FM4) and a 57.5% reduction relative to the single large model (FM1). It also achieves a 6.2% reduction in energy consumption compared to the large pruned model (FM6), while consuming an equivalent amount of energy as the small distilled model (FM7).

To evaluate the system's performance in a dynamic IoT setting with constantly changing inputs, we conducted the following experiment. We provided 10 randomly selected batches (each containing 100 images) from CIFAR-10 and 10 batches (each containing 100 images) from ImageNet, which were fed to the Jetson Nano device in random order. The Jetson Nano forwarded each image to the DNNs presented in Table 2. This setup created a highly dynamic scenario with continuously changing and unpredictable input streams. Power consumption was measured using a power meter, and the results showed that the approach with the Single large models consumed 0.57 Wh, the Big/Little models consumed 0.484 Wh, and the small complementary pairs consumed 0.177 Wh. These findings demonstrate that the proposed approach effectively balances energy efficiency in high-throughput, variable data streams and outperforms other methods in this dynamic environment.

4.6.5. Performance evaluation

Fig. 7 illustrates the inference performance metrics. For the CIFAR-10 dataset, our configuration (CI3) shows a 1.67% performance degradation compared to the big/little configuration (CI2) and a 1.39% degradation relative to the single large model (CI1). However, when compared to the large pruned model (CI6), our configuration shows a 9.0% improvement in performance, and a 16.5% increase when compared to the small distilled model (CI7).

A similar trend is observed on the ImageNet dataset. Our configuration (IM3) shows a 2.19% performance degradation relative to the big/little setup (IM2) and a 2.12% degradation compared to the single large model (IM1). In contrast, our configuration demonstrates an 8.0% performance improvement over the large pruned model (IM6).

For the Intel dataset, our configuration (IN3) shows a slight performance increase of 0.03% compared to the big/little configuration (IN2), though there is a 0.57% decrease in performance relative to the single large model (IN1). However, it achieves a 2.0% performance improvement over the large pruned model (IN6) and a 3.2% improvement compared to the small distilled model (IN7).

Last, for the FashionMNIST dataset, our configuration (FM1) exhibits a 2.12% decrease in performance relative to the big/little configuration (FM2) and a 2.06% decrease compared to the single large model

(FM1). In contrast, it demonstrates a 0.56% performance improvement over the large pruned model (FM6) and a 0.18% increase relative to the small distilled model (FM7).

We should clarify that classification in ImageNet generally has lower performance than in CIFAR-10, Intel and FashionMNIST due to the significantly larger and more diverse set of images and classes in ImageNet, which increases the complexity and difficulty of the classification task. Although our methodology results in a slight decrease in performance metrics close to 2% due to the use of significantly smaller models, it achieves substantial improvements in energy consumption close to 70%, which is the primary objective of this research.

4.6.6. Complementarity evaluation

To evaluate the improvement in accuracy achieved by leveraging the complementarity of the CNN pairs we made experimental comparisons of the CNNs pairs against the single CNNs they include and six CNNs pairs with different complementarity values. In Fig. 7 we see the performance of the pair configurations against the baseline performance of each individual model within each pair, as shown in Table 1.

For the CIFAR-10 dataset, our implementation (CI3) exhibits a 0.76% increase in prediction accuracy over the best-performing individual model it includes (i.e. MobileNetV2-0.5), while the big/little configuration (CI2) shows a 0.28% improvement against the single big CNN (i.e. RepVGG-A2).

Similarly, for the ImageNet dataset, our implementation (IM3) achieves a 1.34% increase in prediction accuracy compared to the highest-performing single model (i.e. DenseNet-121), whereas the big/little configuration (IM2) demonstrates only a 0.07% enhancement compared to the single large CNN (RegNet-X-8GF).

For the Intel dataset our configuration (IN2) achieves a 0.7% increase in prediction accuracy over the best-performing individual model in includes (i.e. MobileNetV2-X1.0), while the big/little configuration (IN3) shows a 0.9% decrease in predictive performance against the single big CNN (i.e. RepVGG-A2).

Last, for the FashionMNIST dataset our implementation (FM3) shows a 1.35% increase in prediction accuracy over the best-performing individual model it includes (i.e. Resnet44) whereas the big/little configuration (FM3) shows only an 0.06% increase in prediction accuracy compared to the single large model (i.e. RepVGG-A2).

The experiments using six pairs of CNNs with different complementarity values are illustrated in Fig. 12. The green and orange segments of the stacked bars represent the accuracy of the individual CNNs. The purple segment indicates the accuracy of CNN pairs based on our proposed methodology. The numbers inside the stacked bar charts show the accuracy gain of the CNN pairs compared to the best-performing individual CNN. The outcomes show that utilizing a pair of CNNs with a relatively higher complementarity factor leads to improved accuracy. This is achieved by covering a larger portion of the dataset, despite employing smaller and less complex models as we described in the Section 3.1.

4.6.7. Response time evaluation

Additionally, our methodology improves response times, as shown in Fig. 8. For the CIFAR-10 dataset, our configuration (CI3) has 52.7% lower mean response time, 44.6% lower 95-th percentile tail latency and 43.7% lower 99-th percentile tail latency compared to the big/little configuration (CI2). Similarly it has 56.9% lower mean response time, 2.5% lower 95-th percentile tail latency and 2.5% lower 99-th percentile tail latency compared to the single large model (CI1). Compared to the large pruned model (CI6) it has 41.1% lower mean response time but a 26.1% higher 95-th percentile tail latency and 24.6% higher 99-th percentile tail latency. Compared to the small distilled model (CI7) it has 47.3% lower mean response time but 17.1% higher 95-th percentile tail latency and 16.4% 99-th percentile tail latency.

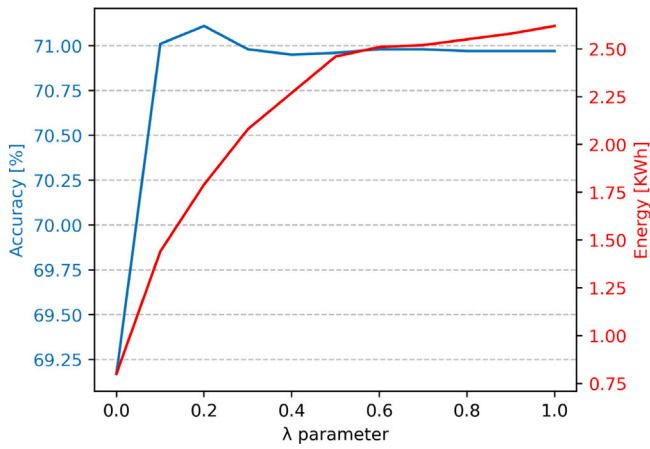


Fig. 9. Accuracy and energy curves as a function of λ parameter, for IM3 configuration on ImageNet test set.

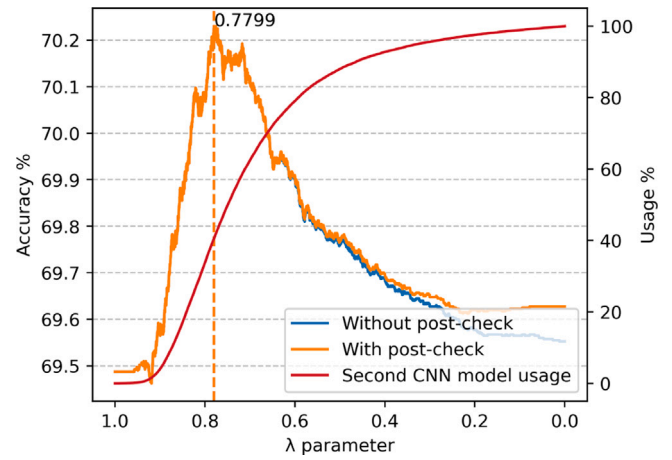


Fig. 11. Post-check evaluation for the Entropy score function on the ImageNet validation dataset using configuration IM3.

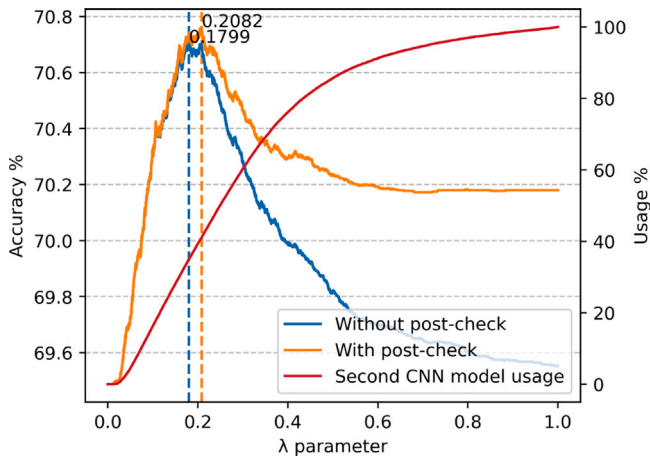


Fig. 10. Post-check evaluation for the Max Probability score function on the ImageNet validation dataset using configuration IM3.

We have the same observations with the ImageNet dataset: our configuration (IM3) has 65.2% lower mean response time, 49.5% lower 95-th percentile tail latency and 49% lower 99-th percentile tail latency compared to the big/little setup (IM2) and 60.3% lower mean response time, 22.9% lower 95-th percentile tail latency and 21.1% lower 99-th percentile tail latency compared to the single large model (IM1). Compared to the large pruned model (IM6) it has 31.6% lower mean response time but 33.8% higher 95-th percentile tail latency and 34.5% increase in 99-th percentile tail latency.

For the Intel dataset, our configuration (IN3) achieves a 27.4% reduction in mean response time, a 17.6% decrease in 95th percentile tail latency, and a 17.4% decrease in 99th percentile tail latency compared to the big/little setup (IN2). When compared to the single large model (IN1), our implementation shows an 18.2% reduction in mean response time, though it exhibits a 48.9% increase in 95th percentile tail latency and a 45.0% increase in 99th percentile tail latency. Relative to the large pruned model (IN6), it features a 1.7% decrease in mean response time but has 77.9% higher 95th percentile tail latency and 76.9% higher 99th percentile tail latency. Compared to the small distilled model (IN7), it shows a 5.7% increase in mean response time, alongside an 80.6% increase in 95th percentile tail latency and a 76.3% increase in 99th percentile tail latency.

Last, for the FashionMNIST dataset, our configuration (FM3) delivers an 18.8% reduction in mean response time, a 40.7% decrease in

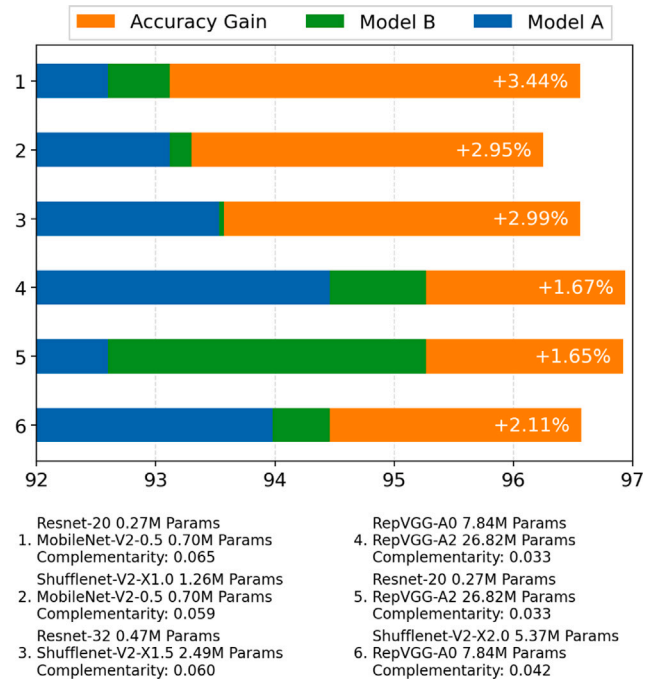


Fig. 12. Accuracy gains as a function of the complementarity of two CNN pairs.

95th percentile tail latency, and a 35.8% decrease in 99th percentile tail latency compared to the big/little setup (FM2). Compared to the single large model (FM1), our implementation shows an 11.9% reduction in mean response time, but with a 40.0% increase in 95th percentile tail latency and a 35.8% increase in 99th percentile tail latency. In comparison to the large pruned model (FM6), it has a 9.6% increase in mean response time, with 69.4% higher 95th percentile tail latency and 70.3% higher 99th percentile tail latency. Relative to the small distilled model (FM7), it exhibits a 9.8% increase in mean response time, a 73.6% increase in 95th percentile tail latency, and a 40.5% increase in 99th percentile tail latency.

Using two CNNs results in an increase in response time and tail latency compared to a single CNN, as shown in Table 1. This occurs because the second CNN is triggered when the first CNN's prediction confidence is low, introducing an additional delay in response time. This effect is observed in both our methodology and the big/little configuration. Despite the increased tail latency, our implementation

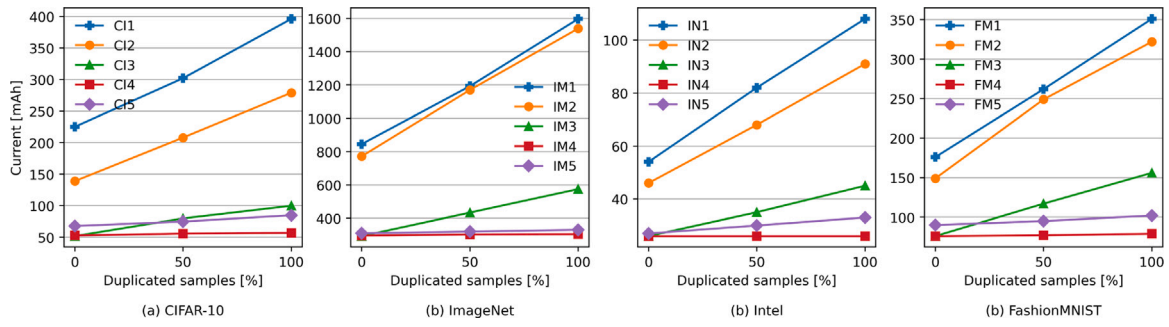


Fig. 13. Energy consumption comparisons using memory component with duplicated samples.

Table 3
Enhancing complementarity evaluation.

	CNN pairs	Individual accuracy	Accuracy using our methodology		Fine-tuned approach (b)
			Default pre-trained weights	Fine-tuned approach (a)	
1	MnasNet-1.3	69.17%	70.33%	69.33%	70.31%
	DenseNet-121	69.65%			
2	MobileNet-V3-Large	66.94%	69.12%	71.20%	70.78%
	GoogleNet	65.27%			
3	MnasNet-1.3	65.57%	68.01%	66.95%	68.65%
	GoogleNet	65.27%			
4	RegNet-X-400MF	66.84%	69.39%	69.38%	70.82%
	RegNet-Y-400MF	67.12%			
5	RegNet-Y-400MF	67.12%	69.03%	69.03%	69.74%
	GoogleNet	65.27%			

achieves a lower overall response time when compared to configurations with similar performance, such as the single large model and the big/little configuration. In comparison to the pruned model and the small distilled model, our configuration shows a higher tail latency due to the extra time needed to invoke the second CNN; however, its performance is higher.

4.6.8. Perceptual hashing ablation

Fig. 13 presents the energy consumption comparisons when using the memory component. The comparison includes a single large model (CI1, IM1, IN1 and FM1) and a big/little architecture (CI2, IM2, IN2 and FM2) against our implementation without the memory component (CI3, IM3, IN3 and FM3), our implementation using the memory component with the Difference Hash (DHash) method (CI4, IM4, IN4 and FM4), and our implementation using the memory component with the Invariants from Complex Moments method (CI5, IM5, IN5 and FM5). The configurations without a memory component (CI1 to CI3, IM1 to IM3, IN1 to IN3 and FM1 to FM3) exhibit a linear increase in energy consumption as the number of duplicated samples increases. In contrast, our implementations utilizing the memory component (CI4, CI5, IM4, IM5, IN4, IN5, FM4, and FM5) show almost no increase in energy consumption, with the slopes of the corresponding lines being very close to zero. This indicates that leveraging the memory component in an environment with identical, mirrored and rotated inputs significantly reduces energy consumption. Additionally, it is noteworthy that while the Invariants method is robust to image transformations, it demands more computational and energy resources compared to the DHash method.

We also evaluated the computational overhead introduced by the memory component. To quantify this, we measured the energy consumption and current draw using the standard dataset without duplicated samples, allowing us to determine the additional energy required for the memory component's operation. As shown in Fig. 14, for the CIFAR-10 dataset, the Difference Hash method led to a 1.9% increase in energy consumption, while the Invariants From Complex Moments method resulted in a 30.9% increase compared to not using the memory

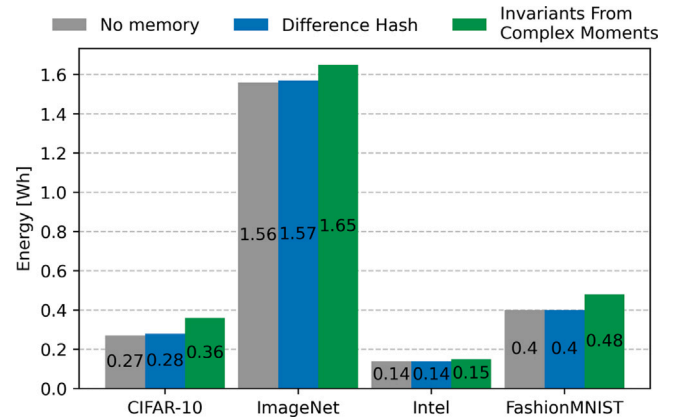


Fig. 14. Memory component computational overhead.

component. For the ImageNet dataset, energy consumption increased by 1.0% with the Difference Hash method and by 5.8% with the Invariants From Complex Moments. In the case of the Intel dataset, the Difference Hash method did not show any measurable increase in energy consumption, whereas the Invariants From Complex Moments led to a 7.1% increase. Finally, for the FashionMNIST dataset, there was no measurable increase in energy consumption with the Difference Hash method, while the Invariants From Complex Moments increased energy consumption by 20%.

4.6.9. Enhancing complementarity evaluation

Table 3 summarizes the results of our methodology, using fine-tuned model pairs from both approaches outlined in Section 3.6 to enhance complementarity. The table presents five CNN pairs with their default weights, the individual accuracy of each CNN, the accuracy of each pair of CNNs, and the results after applying the first and second fine-tuning approaches. Some pairs exhibit marginally better performance with the

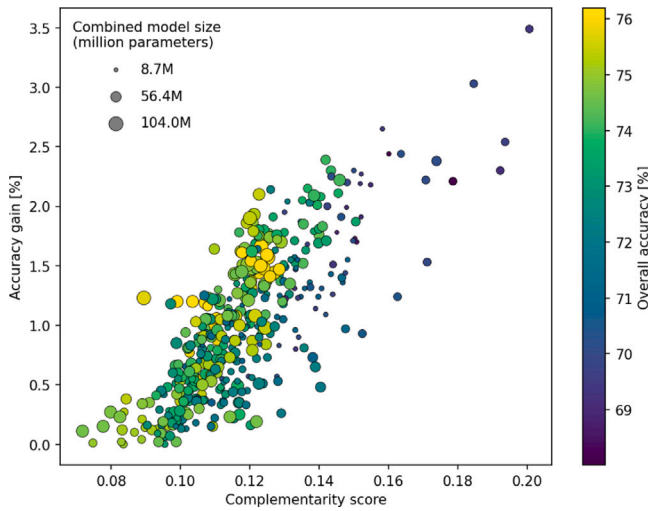


Fig. 15. Correlation between complementarity and accuracy gain.

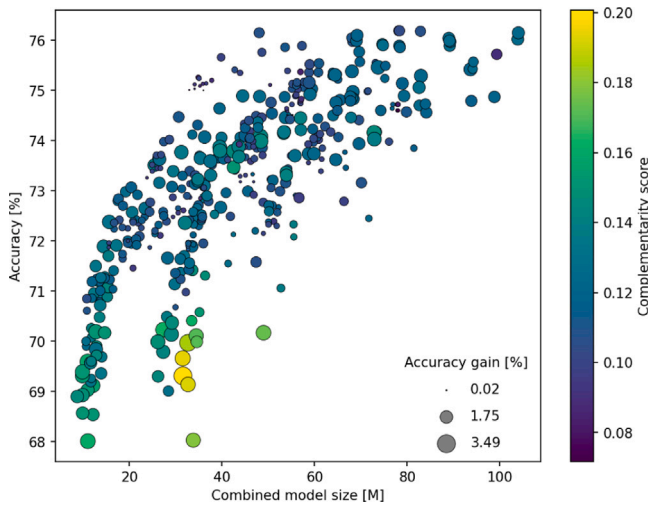


Fig. 16. Correlation between combined model size and overall accuracy.

first approach, while others perform slightly better with the second approach, with a performance gain of approximately 1% in most cases. A notable result is observed with the second pair, where performance increased by 2.08%, achieving an overall accuracy of 71.20%, which surpassed even our main selected ImageNet configuration IM3 (see Table 2). Given that the highest-performing individual model in the pair achieved an accuracy of 66.94%, this results in an overall accuracy improvement of 4.26% using our methodology with fine-tuned models.

4.6.10. Correlation between complementarity & increased performance

To better understand the correlation between complementarity scores and accuracy gains, we applied our methodology, as described in Section 3, to all possible pair combinations of the ImageNet dataset models used in the complementarity matrix, to assess performance outcomes. The results are illustrated in Fig. 15. Each point on the graph represents a CNN pair, with the X-axis showing the complementarity score calculated using the complementarity formula (1), and the Y-axis showing the accuracy gain—defined as the additional accuracy achieved through our methodology compared to the best-performing model in the pair.

To provide additional insight, we included two additional dimensions of information: the size of each point reflects the combined number of parameters (M) of both models, and the color indicates

the overall accuracy achieved by the pair. The graph reveals a positive linear correlation between complementarity scores and accuracy gains, suggesting that utilizing more complementary CNN pairs leads to improved accuracy. When examining overall performance, the largest models achieve the highest absolute accuracy, as expected. Despite this, their low complementarity scores result in minimal accuracy gains. This indicates that, in addition to their higher energy demands, these large pairs of models are also less efficient, as a single large model would provide nearly the same level of performance.

To further explore the nature of complementarity, in Fig. 16, we plotted the same results but with different axes. In this graph, the X-axis represents the combined model size in millions of parameters (M), and the Y-axis represents the overall accuracy achieved by the pair using our methodology. Additionally, the size of each point corresponds to the accuracy gain, while the color indicates the complementarity score. The graph reveals a logarithmic correlation, rather than a linear one, between model size and performance. This implies that although larger models tend to improve accuracy, the gains become progressively smaller as size increases, leading to a higher energy cost per unit of accuracy. Based on these insights, we can conclude that using a highly complementary pair of smaller CNN models results in increased accuracy with lower energy demand.

4.7. Discussion

Our methodology contributes to advancing knowledge in the field of energy-efficient on-device AI applications by making research on a relatively unexplored approach: dynamically switching between small DNNs based on confidence scores. This strategy significantly enhances energy efficiency while maintaining predictive accuracy, offering a novel solution for resource-constrained smart environments. Two innovative aspects of our approach have the potential to shape future research directions. First, we illustrate how an effective collaboration between complementary DNNs can improve prediction accuracy by leveraging the strengths of each model to compensate for the weaknesses of the other. This work demonstrates the potential for DNNs to collaborate on the same task, setting the stage for further exploration of cooperative model architectures. Second, we highlight the importance of integrating a memory component that retains previous decisions, allowing the system to bypass redundant inference processes for repeated inputs. This synergy between inference and memory opens new opportunities for optimizing both energy efficiency and predictive performance in AI-driven edge computing environments.

In terms of effectiveness, unlike traditional compression techniques or the big-little architecture, the proposed dual complementary CNN methodology leverages two smaller models that compensate for each other's weaknesses. This design achieves accuracy comparable to larger, more complex models while significantly reducing energy consumption. By dynamically selecting predictions based on confidence scores, the method ensures that only the most accurate predictions are used, enhancing overall inference reliability. This approach effectively maintains high accuracy without the computational overhead typically associated with large models.

In terms of efficiency, the methodology is specifically designed to reduce energy consumption and response time, a critical concern for edge devices with limited power and computational resources. The dual-CNN setup selectively activates the second CNN based on prediction confidence, ensuring it is only invoked when necessary. Additionally, the integration of a memory component, which stores and recalls previous classifications, further reduces energy consumption by bypassing repeated inference on identical or similar inputs.

In terms of applicability, the methodology is broadly suitable for resource-constrained devices, such as those in IoT, smart homes, or edge computing environments. Unlike hardware-specific optimizations, such as co-design approaches requiring custom hardware or intensive

model compression techniques (e.g., pruning, quantization), this dual-CNN method is hardware-agnostic and can be deployed on a wide range of devices without substantial modifications. The use of perceptual hashing in the memory component also makes it ideal for dynamic environments where inputs may frequently repeat or vary slightly, providing practical benefits in real-world scenarios.

Our proposed methodology focuses on multi-class classification, but not multi-label classification, specifically using visual data. At present, it does not support data samples that can be assigned multiple labels or categories simultaneously. Our design has the limitations of being tailored and evaluated for visual data, opening opportunities for further research to explore other data modalities such as audio, biometric, geo-location, and sensor time series data in smart environments. Additionally, we acknowledge the limitation that our methodology operates on two small prediction models, and if such models are unavailable, a transfer learning process will be necessary. In cases where the transfer learning models have low accuracy, our experimental results demonstrate that our methodology significantly improves their performance.

To ensure the robustness and validity of our experimental results, we conducted experiments using four different datasets. We carefully managed data splitting by employing random shuffling, stratification, and preventing data leakage. The evaluation was performed on unseen data to provide unbiased performance estimates. Furthermore, we compared the performance of our proposed model against well-established baseline models, including pruning, knowledge distillation, and the big/little model [24]. We also ensured the reproducibility of our experiments by testing with a variety of architectural pairs, thereby confirming the consistency of the improvements observed across different model synergies.

5. Conclusion and future work

In this study, we propose a methodology for reducing the energy requirements of on-device CNNs inference through the utilization of two complementary CNNs integrated with a memory component. Each CNN addresses the weaknesses of the other, while the memory component retains previous classifications, thereby bypassing the need for repeated CNN invocations. Our implementation demonstrates up to 85.8% reduction in energy consumption compared to a single large CNN for inferences with multiple identical inputs on the CIFAR-10 dataset up to 80.9% energy reduction on the ImageNet dataset, up to 76.0% energy reduction on the Intel dataset and up to 77.5% energy reduction on the FashionMNIST dataset with a minor loss of accuracy.

Our future work involves adapting the concept of complementarity to various data modalities and applications in smart environments. Further research directions include examining complementarity based not only on the number of predictions made by the models in a validation dataset but also by focusing solely on confidence scores. Additionally, we aim to develop a new complementarity formula that considers the intrinsic structural characteristics of the CNNs. Last, since the concept of complementarity is a significant contribution of our work, we aim to explore its application for enhancing accuracy, rather than solely focusing on reducing energy consumption.

CRedit authorship contribution statement

Michail Kinnas: Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **John Violos:** Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology, Investigation, Formal analysis, Conceptualization. **Ioannis Kompatsiaris:** Supervision, Project administration, Funding acquisition. **Symeon Papadopoulos:** Writing – review & editing, Project administration, Methodology, Investigation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was funded by the European Union's Horizon Europe research and innovation program under grant agreement No. 101120237 (ELIAS - European Lighthouse of AI for Sustainability).

Data availability

The experiments conducted in this article utilized publicly available datasets. As such, no additional data were generated or analyzed that require sharing.

References

- [1] N. Tekin, A. Aris, A. Acar, S. Uluagac, V.C. Gungor, A review of on-device machine learning for IoT: An energy perspective, *Ad Hoc Netw.* 153 (2024) 103348, <http://dx.doi.org/10.1016/j.adhoc.2023.103348>, URL <https://www.sciencedirect.com/science/article/pii/S1570870523002688>.
- [2] P. Rahmani, M. Arefi, Improvement of energy-efficient resources for cognitive internet of things using learning automata, *Peer-to-Peer Netw. Appl.* 17 (1) (2024) 297–320, <http://dx.doi.org/10.1007/s12083-023-01565-y>.
- [3] J. Chai, H. Zeng, A. Li, E.W. Ngai, Deep learning in computer vision: A critical review of emerging techniques and application scenarios, *Mach. Learn. Appl.* 6 (2021) 100134.
- [4] A. Bimpas, J. Violos, A. Leivadreas, I. Varlamis, Leveraging pervasive computing for ambient intelligence: A survey on recent advancements, applications and open challenges, *Comput. Netw.* 239 (2024) 110156, <http://dx.doi.org/10.1016/j.comnet.2023.110156>, URL <https://www.sciencedirect.com/science/article/pii/S1389128623006011>.
- [5] P. Qi, D. Chiaro, F. Piccialli, Small models, big impact: A review on the power of lightweight Federated Learning, *Future Gener. Comput. Syst.* 162 (2025) 107484, <http://dx.doi.org/10.1016/j.future.2024.107484>, URL <https://www.sciencedirect.com/science/article/pii/S0167739X24004400>.
- [6] X. Ma, T. Yao, M. Hu, Y. Dong, W. Liu, F. Wang, J. Liu, A survey on deep learning empowered IoT applications, *IEEE Access* 7 (2019) 181721–181732.
- [7] T.-J. Yang, Y.-H. Chen, J. Emer, V. Sze, A method to estimate the energy consumption of deep neural networks, in: 2017 51st Asilomar Conference on Signals, Systems, and Computers, IEEE, 2017, pp. 1916–1920.
- [8] J.O. Neill, An overview of neural network compression, 2020, arXiv preprint [arXiv:2006.03669](https://arxiv.org/abs/2006.03669).
- [9] A.E. Brownlee, J. Adair, S.O. Haraldsson, J. Jabbo, Exploring the accuracy–energy trade-off in machine learning, in: 2021 IEEE/ACM International Workshop on Genetic Improvement, GI, IEEE, 2021, pp. 11–18.
- [10] C. González García, D. Meana-Llorián, B.C. Pelayo G-Bustelo, J.M. Cueva Lovelle, N. García-Fernández, Midgar: Detection of people through computer vision in the Internet of Things scenarios to improve the security in Smart Cities, Smart Towns, and Smart Homes, *Future Gener. Comput. Syst.* 76 (2017) 301–313, <http://dx.doi.org/10.1016/j.future.2016.12.033>, URL <https://www.sciencedirect.com/science/article/pii/S0167739X16308652>.
- [11] A. Goel, C. Tung, Y.-H. Lu, G.K. Thiruvathukal, A survey of methods for low-power deep learning and computer vision, in: 2020 IEEE 6th World Forum on Internet of Things, WF-IoT, IEEE, 2020, pp. 1–6.
- [12] B.J. Eccles, P. Rodgers, P. Kilpatrick, I. Spence, B. Varghese, DNNShifter: An efficient DNN pruning system for edge computing, *Future Gener. Comput. Syst.* 152 (2024) 43–54, <http://dx.doi.org/10.1016/j.future.2023.09.025>, URL <https://www.sciencedirect.com/science/article/pii/S0167739X23003576>.
- [13] S. Tsanakas, A. Hameed, J. Violos, A. Leivadreas, A light-weight edge-enabled knowledge distillation technique for next location prediction of multitude transportation means, *Future Gener. Comput. Syst.* 154 (2024) 45–58, <http://dx.doi.org/10.1016/j.future.2023.12.025>, URL <https://www.sciencedirect.com/science/article/pii/S0167739X23004867>.
- [14] J. Violos, S. Papadopoulos, I. Kompatsiaris, Towards Optimal Trade-offs in Knowledge Distillation for CNNs and Vision Transformers at the Edge, 2024, <http://dx.doi.org/10.48550/arXiv.2407.12808>, URL <http://arxiv.org/abs/2407.12808>, arXiv:2407.12808 [cs].
- [15] J. Gawlikowski, C.R.N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher, M. Shahzad, W. Yang, R. Bamler, X.X. Zhu, A survey of uncertainty in deep neural networks, *Artif. Intell. Rev.* 56 (1) (2023) 1513–1589, <http://dx.doi.org/10.1007/s10462-023-10562-9>.

- [16] L. Du, A.T. Ho, R. Cong, Perceptual hashing for image authentication: A survey, *Signal Process., Image Commun.* 81 (2020) 115713.
- [17] C. Hu, W. Bao, D. Wang, F. Liu, Dynamic adaptive DNN surgery for inference acceleration on the edge, in: *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, IEEE, 2019, pp. 1423–1431.
- [18] L. Zeng, X. Chen, Z. Zhou, L. Yang, J. Zhang, Coedge: Cooperative dnn inference with adaptive workload partitioning over heterogeneous edge devices, *IEEE/ACM Trans. Netw.* 29 (2) (2020) 595–608.
- [19] Q. Wang, W. Li, A. Mohajer, Load-aware continuous-time optimization for multi-agent systems: toward dynamic resource allocation and real-time adaptability, *Comput. Netw.* 250 (2024) 110526, <http://dx.doi.org/10.1016/j.comnet.2024.110526>, URL <https://www.sciencedirect.com/science/article/pii/S138912862400358X>.
- [20] S. Kuang, J. Zhang, A. Mohajer, Reliable information delivery and dynamic link utilization in MANET cloud using deep reinforcement learning, *Trans. Emerg. Telecommun. Technol.* 35 (9) (2024) e5028, <http://dx.doi.org/10.1002/ett.5028>, eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/ett.5028>, URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.5028>.
- [21] Z. Zhang, Y. Zhang, W. Bao, C. Li, D. Yuan, Coarse-to-fine: A hierarchical DNN inference framework for edge computing, *Future Gener. Comput. Syst.* 157 (2024) 180–192.
- [22] C. Hao, X. Zhang, Y. Li, S. Huang, J. Xiong, K. Rupnow, W.-m. Hwu, D. Chen, FPGA/DNN co-design: An efficient design methodology for IoT intelligence on the edge, in: *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.
- [23] J. Lee, S. Kang, J. Lee, D. Shin, D. Han, H.-J. Yoo, The hardware and algorithm co-design for energy-efficient DNN processor on edge/mobile devices, *IEEE Trans. Circuits Syst. I. Regul. Pap.* 67 (10) (2020) 3458–3470.
- [24] E. Park, D. Kim, S. Kim, Y.-D. Kim, G. Kim, S. Yoon, S. Yoo, Big/little deep neural network for ultra low power inference, in: *2015 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, 2015, pp. 124–132, <http://dx.doi.org/10.1109/CODESIS.2015.7331375>, URL <https://ieeexplore.ieee.org/document/7331375>.
- [25] N.K. Jayakodi, A. Chatterjee, W. Choi, J.R. Doppa, P.P. Pande, Trading-off accuracy and energy of deep inference on embedded systems: A co-design approach, *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* 37 (11) (2018) 2881–2893, <http://dx.doi.org/10.1109/tcad.2018.2857338>.
- [26] Z. Tang, Y. Dai, X. Zhang, Perceptual hashing for color images using invariant moments, *Appl. Math. Inf. Sci.* 6 (2S) (2012) 643S–650S.
- [27] J. Flusser, B. Zitova, T. Suk, *Moments and Moment Invariants in Pattern Recognition*, John Wiley & Sons, 2009.
- [28] A. Krizhevsky, *Learning Multiple Layers of Features from Tiny Images*, University of Toronto, 2012.
- [29] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017, arXiv preprint [arXiv:1708.07747](https://arxiv.org/abs/1708.07747).
- [30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, ImageNet: A large-scale hierarchical image database, in: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, (ISSN: 1063-6919) 2009, pp. 248–255, <http://dx.doi.org/10.1109/CVPR.2009.5206848>, URL <https://ieeexplore.ieee.org/document/5206848>.
- [31] T. Theodoropoulos, J. Violos, A. Makris, K. Tserpes, A new approach for evaluating the performance of distributed latency-sensitive services, in: *2024 IEEE International Conference on Communications Workshops, ICC Workshops*, 2024, pp. 365–370, <http://dx.doi.org/10.1109/ICCWshops59551.2024.10615818>.



Michail Kinnas works as a Researcher Intern at Centre for Research & Technology Hellas, Information Technologies Institute. He received his B.Sc. (hons.) degree from Harokopio University of Athens, Department of Informatics and Telematics, Greece, in 2024. His research interests include Machine Learning, Artificial Neural Networks and Computer Vision.



John Violos works at the Center for Research & Technology Hellas. He is also adjunct professor at Harokopio University of Athens and at National and Kapodistrian University of Athens. His previous positions include: Senior researcher in the Dept. of Software Engineering and Information Technology at École de Technologie Supérieure, Université du Québec in Montreal, Canada; Senior researcher at National Technical University of Athens, Greece. He was a member of the European Commission's Digital Single Market working group on the code of conduct for switching and porting data between cloud service providers. He has participated as work package leader, task leader and researcher in more than fourteen research projects funded by the European Union, Korea and Canada. He has more than fifty publications in top-tier conferences and journals. He received the Best Paper Award of the IEEE CISOSE 2023 Conference, the Best Paper Award of the IEEE iThings 2021 Conference, the Best Course Teaching by an adjunct professor Award of Harokopio University of Athens 2021 and Thomaïdion Award 2017.



Ioannis Kompatsiaris is a Research Director at the CERTH-ITI, the Head of the Multimedia Knowledge and Social Media Analytics Laboratory, and Deputy Director of the institute. His research interests include image and video analysis, Big Data and social media analytics, semantics, human computer interfaces (AR and BCI), multimedia, eHealth, security, and culture applications. He is the co-author of 171 papers in refereed journals, 63 book chapters, 8 patents, and more than 500 papers in international conferences. Since 2001, Dr. Kompatsiaris has participated in 89 National and European research programs, 18 of which he has been the Project Coordinator. He has also been the PI in 14 research collaborations with industry, including Motorola US and UK. He has been the co-chair of various international conferences and workshops, including the 13th IEEE Image, Video, and Multidimensional Signal Processing (IVMSP 2018) Workshop, and he has served as a regular reviewer and associate and guest editor for a number of journals and conferences, currently working as an associate editor of IEEE Transactions on Image Processing. He is a member of the Scientific Advisory Board of the CHIST-ERA funding program, an elected member of the IEEE Image, Video, and Multidimensional Signal Processing—Technical Committee (IVMSP - TC), a senior member of the IEEE, and a member of the ACM.



Symeon Papadopoulos is a Principal Researcher with the Centre for Research and Technology Hellas (CERTH), the Information Technologies Institute, and part of the Multimedia Knowledge and Social Media Analytics Lab (MKLab). He is currently leading the MeVer group, a team of approximately 30 researchers and engineers working in a number of projects at the intersection of Artificial Intelligence, Media, Multimedia Retrieval and Intelligent Network Analytics. In particular, during the last years, he has been interested in the topics of AI bias, online disinformation, media verification, and trend detection and forecasting using social media signals. He has been coordinating and leading a number of collaborative research and innovation projects. He has co-authored more than 200 publications, has served as TPC member in more than 50 conferences and he is a regular reviewer in more than 20 journals. Since 2014, he is one of the co-founders of the Infalia Private Company, a spin-off company of CERTH.