electronics

MDPI

Article

Self-Adaptive Approximate Mobile Deep Learning

Timotej Knez, Octavian Machidon and Veljko Pejović *

Faculty of Computer and Information Science, University of Ljubljana, 1000 Ljubljana, Slovenia; tk7225@student.uni-lj.si (T.K.); octavian.machidon@fri.uni-lj.si (O.M.)

* Correspondence: veljko.pejovic@fri.uni-lj.si

Abstract: Edge intelligence is currently facing several important challenges hindering its performance, with the major drawback being meeting the high resource requirements of deep learning by the resource-constrained edge computing devices. The most recent adaptive neural network compression techniques demonstrated, in theory, the potential to facilitate the flexible deployment of deep learning models in real-world applications. However, their actual suitability and performance in ubiquitous or edge computing applications has not, to this date, been evaluated. In this context, our work aims to bridge the gap between the theoretical resource savings promised by such approaches and the requirements of a real-world mobile application by introducing algorithms that dynamically guide the compression rate of a neural network according to the continuously changing context in which the mobile computation is taking place. Through an in-depth trace-based investigation, we confirm the feasibility of our adaptation algorithms in offering a scalable trade-off between the inference accuracy and resource usage. We then implement our approach on real-world edge devices and, through a human activity recognition application, confirm that it offers efficient neural network compression adaptation in highly dynamic environments. The results of our experiment with 21 participants show that, compared to using static network compression, our approach uses 2.18× less energy with only a 1.5% drop in the average accuracy of the classification.

Keywords: mobile sensing; neural networks; dynamic optimization; quantization; DNN slimming

Citation: Knez, T.; Machidon, O.; Pejović, V. Self-Adaptive Approximate Mobile Deep Learning. *Electronics* **2021**, *10*, 2958. https://doi.org/10.3390/electronics10232958

Academic Editor: Arturo de la Escalera Hueso

Received: 8 November 2021
Accepted: 25 November 2021
Published: 28 November 2021

1. Introduction

The ongoing technological developments and the expansion of the Internet of Things have led to an overwhelming amount of data being collected in real time by a wide range of sensors and end devices, such as smartphones. This raises the need for the real-time processing of data using AI in order to perform knowledge extraction and transform the data into meaningful information which can be acted upon. Hence, yesterday's sensors and embedded computing devices are transcending into today's edge computing nodes, with the aim of processing the data as close to the originating source as possible.

The most powerful machine learning technology used today for data processing and analysis is deep earning, which has been shown to be successful in a variety of application domains ranging from computer vision [1] and human activity recognition [2] to natural language processing [3] and others. The concept of edge intelligence—running deep learning on edge devices—has the potential to ameliorate several drawbacks associated with traditional centralized computing, among which are network latency, scalability, privacy and data security concerns.

However, several important challenges hinder the performance of deep learning at the edge. High accuracy and high resource consumption are defining characteristics of deep learning, which inherently come into conflict with the resource-constrained nature and (as in the cases of smartphones) limited battery capacity of edge computing devices. To achieve high accuracy, a DNN (deep neural network) requires significant amounts of memory to store all the model parameters and will have a high latency because of all the matrix multiplications being performed. In addition, Graphics Processing Units (GPUs)

play a critical role in efficient DNN inference and training. Some of the most widely used GPU software libraries, such as CUDA [4] and cuDNN [5] (which is targeted towards deep learning), are not widely available on many edge computing nodes such as today's smartphones [6].

The last decade's research efforts have begun to question the necessity of such a large number of parameters and network layers in deep learning models [7]. Consequently, focus has shifted towards reducing the computational complexity of deep learning models through several design strategies, such as knowledge distillation [8], early exit [9] and network compression [10]. Nevertheless, all such neural network complexity reduction methods come with the downside of decreasing inference accuracy as the network compression rate increases. In addition, traditional approaches apply a single static compression to the network, which then operates in a permanently impaired state regardless of any changes in circumstances that may arise and call for a more or less accurate network inference.

Recent advances mitigate this issue by proposing adaptive compression techniques that can dynamically adjust the network compression rate during runtime. These approaches, such as Slimmable Neural Networks [11] or Any-Precision Deep Neural Networks [12], promise to facilitate the flexible deployment of deep learning models in scenarios where a trade-off between inference accuracy and runtime efficiency are sought. These approaches enable the on-demand engagement of fewer or more deep learning parameters, or an on-demand usage of a lower or a higher numerical precision, and thus balance the precision and the execution speedup. To date, however, these approaches have only been examined at a very high level, and their practical utility in real-world applications and on actual edge devices is unexplored. Moreover, while the above approaches, at least theoretically, enable an accuracy–resource usage trade-off, it remains unclear how to dynamically adapt the operation along the trade-off line in order to maximize the achieved accuracy with minimal resource usage.

In this paper, inspired by the Approximate Mobile Computing principles [13], we propose an innovative approach for dynamic mobile deep learning adaptation that aims to bridge the gap between the theoretical resource savings brought by adaptive neural network compression techniques and the requirements of a real-world mobile application. We devise original algorithms that continuously analyze the difficulty of the inference task and for each input dynamically guide the neural network compression rate in a mobile human activity recognition scenario. Our approach has the potential to foster the deployment of deep learning on resource-constrained IoT edge devices by enabling neural networks to scale their resource usage and thus energy consumption, according to the environmental factors and context of use, with tolerable accuracy drops.

The main contributions of this work are summarized as follows:

- We investigate the inference accuracy and the energy saving potential as well as the practicality of two orthogonal neural network compression techniques (quantization and slimming) in the context of mobile human activity recognition;
- We devise dynamic neural network compression adaptation algorithms that achieve a comparable inference accuracy with up to 33% fewer network parameters compared to the best-performing static compression;
- We conduct a 21-person experiment and assess the utility of our deep learning adaptation approach on a previously unseen dataset, while also demonstrating the real-world usability of the implementation and the potential of real-world energy savings it brings.

Given that one of the main challenges in machine learning-related research is to ensure that the presented and published results are sound and reliable, we performed our research with reproducibility in mind [14], and we have made the code and collected experimental data publicly available to the research community at: https://gitlab.fri.uni-lj.si/lrk/mobilenn-slimming-and-quantization (accessed on 4 November 2021).

2. Related Work

2.1. Deep Learning Model Optimization

Knowledge distillation is based on the principle that a smaller deep neural network (DNN) can successfully capture the knowledge of a larger one if coached by it [15]. The technique involves creating a smaller DNN which is trained using the output predictions of the larger DNN, ultimately leading to the smaller DNN approximating the function learned by the larger network [15]. Nevertheless, the minimized network represents only a single possible approximation of the large DNN. Early exit is another technique whereby the computational complexity of the inference is reduced by not computing all network layers; it is based on the observation that features learned at an early layer of a network may often be sufficient for the classification of many data points, and early exit branches are used to predict the result when the sample can already be inferred with sufficient accuracy at an intermediate point in the network [9]. If implemented as a cascade of early exits, this technique actually enables a finer control over the accuracy vs. approximation trade-off.

Compressing the DNN model is another strategy to enable running DL on resource-constrained edge devices. Such techniques aim at compressing the network model while minimizing the accuracy loss compared to that achieved by the original model. DeepX, for instance, uses singular value decomposition (SVD) to compress the network's layers and enable the execution of the model on resource-constrained devices [16]. In the absence of a direct metric, the authors propose using the weight reconstruction error in order to assess the compression quality. However, compression comes at the price of a ≈5% drop in inference accuracy—a downside which is nearly always present in the case of neural network compression techniques.

The feasibility of "trimming" complex neural networks to operate on edge devices is also demonstrated by other compression solutions based on intelligent weight pruning [17], weight virtualization [18] and quantization [19]. DeepIoT describes a pruning-based compression technique dedicated to running DNNs on edge devices that compresses all commonly used deep learning structures for sensing applications, including fully-connected, convolutional and recurrent neural networks, as well as their combinations [20]. The compressed model generated by DeepIoT can directly use existing deep learning libraries that run on embedded and mobile systems without further modifications, in order to reduce the resource consumption of the sensing system without hurting the performance.

The relationship between the amount of compression and the model performance is seldom straightforward. FastDeepIoT is a framework designed to uncover the non-linear relation between neural network structure and execution time and exploit this relation by finding network configurations that significantly improve the trade-off between execution time and accuracy on mobile and embedded devices, resulting in reduced execution time and energy consumption [21]. In contrast, AdaDeep harnesses reinforcement learning to uncover the best combination of compression techniques, including pruning and the special filter structures borrowed from MobileNet and SqueezeNet, that should be employed for the model to meet application requirements and satisfy mobile resource constraints, such as energy consumption, accuracy or latency [22]. DeepMon, a mobile deep learning inference system, integrates a suite of optimization techniques, including combining quantization with caching of results from intermediate layers on GPUs, to efficiently offload convolutional layers to mobile GPUs and accelerate the processing [23].

2.2. Dynamic Model Compression

The main drawback of the above approaches is that once the identified compression settings are applied, the resulting compressed network is fixed, representing only one of the possible configurations of the accuracy–resource usage trade-off curve; should the inference context change—e.g., DNN runs on a smartphone and the phone is connected to a charger—the changes in requirements call for yet another round of network training and compression. In edge/ubiquitous computing scenarios, this is a very common situation because of the dynamic environment and the ever-changing context of use.

More recently, researchers have tried different strategies to mitigate this issue. Among the first solutions proposed was MCDNN, which builds a collection of compressed models using various compression settings, stores them in the cloud, and according to the changing system requirements downloads the most appropriate model [24]. However, this adds inherent overheads due to the additional resources used for transferring the models. Bolukbasi et al. [25] adopted a different approach and proposed the use of a cascade of increasingly more complex neural network classifiers, with the model selection based on the reliability of the inference (measured by the classifier confidence value) for each inferred sample. This approach also exhibits an overhead that might hinder the energy savings due to the necessity of storing multiple models on the device.

A more efficient approach is the Slimmable Neural Network (SNN) proposed by Yu et al. [11], who designed a general method to train a single neural network executable at different widths (number of channels in a layer), permitting instant and adaptive accuracy–efficiency trade-offs at runtime, with the network adjusting its width on the fly according to on-device benchmarks and resource constraints, rather than downloading and offloading different models. Runtime adaptivity is also proposed by the Any-Precision Deep Neural Networks approach, in which the network is trained with a new method that allows the learned DNNs to be flexible in terms of numerical precision during inference [12]. The Any-Precision network can be flexibly and directly set to different bit-widths during runtime to support a dynamic speed and accuracy trade-off.

The adaptive network compression techniques, such as Slimmable Neural Networks or Any-Precision, have demonstrated the potential to facilitate the flexible deployment of deep learning models in real-world applications, where trade-offs between model accuracy and runtime efficiency are often sought. However, their actual suitability and performance in ubiquitous or edge computing applications has not, to this date, been evaluated. In this work, we aim to bridge the gap between the theoretical resource savings promised by these two approaches and the requirements of a real-world mobile application by investigating their performance on mobile human activity recognition (HAR), a field where deep learning is achieving tremendous success [2,26]. To exploit the flexibility of the two compression frameworks, we focus on the development of algorithms that guide the compression rate (slimming and quantization level, respectively) based on the continuously changing contextual factors in a mobile HAR scenario.

3. Methodology Preliminaries

3.1. Any-Precision and Slimmable Neural Networks

Any-Precision deep neural networks are the first neural network complexity reduction approach examined in our work. This is a training and inference methodology that allows DNNs to be flexible in numerical precision during inference; i.e., the same model during runtime can be set to different bit-widths (without the need for retraining), thus enabling a dynamic speed vs. accuracy trade-off (an overview of this approach is illustrated in Figure 1a). To achieve this, Any-Precision trains individual batch normalization layers for each bit-width and then dynamically switches between these layers at runtime, according to the chosen precision level. Any-Precision allows for the same neural network to be flexible without performance degradation: when all layers are set to low-bits, the authors show that the model achieves an accuracy comparable to dedicated models trained at the same precision. This approach avoids the infeasible re-training of the model during inference or the need to burden the storage budget by keeping multiple models with different resource usage on the same device.

The same principle of having a single neural network model that can flexibly "adapt" during inference is also behind the second neural network compression framework used in this work, namely the Slimmable Neural Networks (SNNs). SNNs exploit the observation that an increase or decrease in the number of weights disproportionally impacts the classification accuracy and that a graceful degradation of inference accuracy can be achieved with a significant reduction in the number of weights [11]. The SNN approach enables a reduc-

tion in the number of active network parameters on the fly to a fraction of the network's width selected from a pre-defined subset (as illustrated in Figure 1b). To avoid the need to re-train the model after each configuration change, SNNs also rely on switchable batch normalization (S-BN) layers, similar to the approach used by Any-Precision. During the training, S-BN privatizes all batch normalization layers for each network width, meaning that all (sub-)networks are jointly trained at all the different widths.

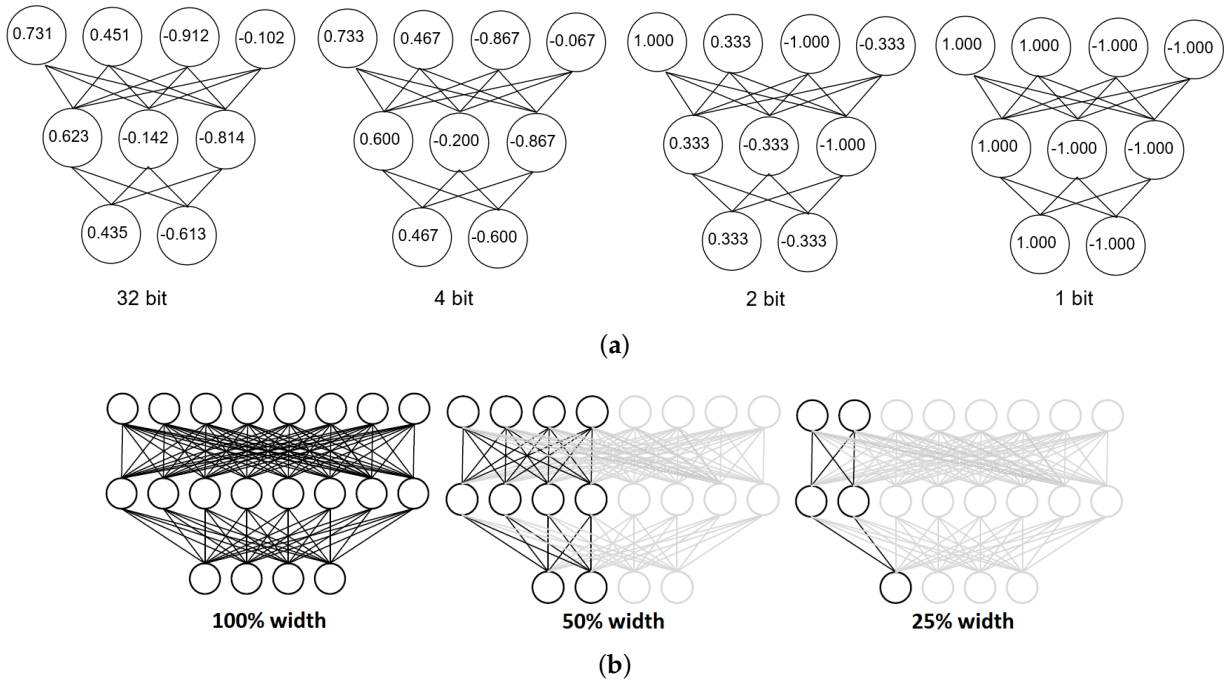


Figure 1. Illustration of the two neural network compression techniques used in our experiments. (**a**) Any-Precision Deep Neural Networks. (**b**) Slimmable Neural Network.

Both Any-Precision and SNNs use switchable batch normalization layers to solve the problem of feature aggregation inconsistency between different widths by independently normalizing the feature mean and variance during testing. This ensures that both approaches maintain acceptable inference accuracy values as the network complexity reduction increases. The main difference between the two approaches is in the real-world energy savings: while SNNs use just a fraction of the network parameters, which translates directly into a reduction of the number of FLOPS (floating point operations per second), Any-Precision truncates the least-significant bits of the weights off to implement dynamic quantization. Since modern consumer-off-the-shelf mobile computers store weights as 32-bit values, in practice, quantization does not transcend into real energy savings. This is not to say that the savings are not going to be shown in future systems—with a great deal of compiler-level support for packing quantized data in full bit-width words (e.g., recent advances in the TVM compiler [27]) and with hardware support for sub-precise computation, we might see energy savings from quantization in the next-generation mobile devices. Nevertheless, even without full mobile system support, the Any-Precision network is functional in illustrating the accuracy vs. resource usage (number of bits used for storing the network parameters) tradeoff. As such, both Any-Precision and SNNs are used for the theoretical evaluation of our dynamic mobile deep learning adaptation framework on a pre-recorded dataset, with the SNNs used for the real-world study given their potential to transcend the network compression into actual energy savings.

3.2. Use-Case: Human Activity Recognition

In our experiments, we evaluate both compression techniques in a Human Activity Recognition (HAR) scenario by training two networks on the publicly available UCI HAR dataset: Any-Precision ResNet-50 (with 1, 2, 4, 8 and 32 bit quantization levels) and SNN MobileNet-V2 (with 0.35, 0.5, 0.75 and 1.0 network widths). The dataset contains recordings of 30 volunteers performing 6 different activities (walking, walking upstairs, walking downstairs, sitting, standing and lying) while carrying a waist-mounted smartphone with embedded inertial sensors [28]. In the original dataset, the collected data were de-noised and sampled in fixed-width sliding windows of 2.56s with 50% overlap (128 readings/window). We used the dataset's predefined train and test partitioning (random partitioning where 70% of the volunteers were selected for generating the training data and 30% the test data), and 20% of the train data were randomly selected as validation data, which translates into 7352 training and 2947 test set instances. Each dataset record consists of the triaxial acceleration from the accelerometer (total acceleration), the estimated body acceleration, the triaxial angular velocity from the gyroscope and a 561-feature vector with time and frequency domain variables. When performing inference during our experiments, we always use the raw acceleration and angular velocity (nine values) as input data, while some of our adaptation algorithms shown in Section 4 also rely on the extracted features.

To train the networks on the UCI data set, we added an additional pre-processing step to re-shape the input data so they matched the networks' input shape requirements. Resnet and MobileNet are traditionally used for image analysis, so by default the expected input shape is $n_c \times w \times h$, where n_c is the number of color channels, w is the image width and h is the image height. We follow an approach similar to that described by Nutter et al. [29], in which the authors created *2D signal images* from the windowed samples of a HAR data set. In our case, we re-shaped the input data into *3D signal images* "mimicking" three color channels and a size of 32 by 32 pixels. In other words, we packed 3072 time-domain signal values to each image, which was then given as an input to the networks. An example of such a resulting image is shown in Figure 2.

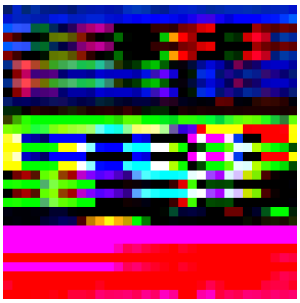


Figure 2. An RGB illustration of a three-channel image consisting of motion data.

The default partitioning of the UCI HAR dataset has the signals divided into windows containing 128 consecutive values. Since we use the 9 acceleration and velocity values, each window thus contains 1152 values. To "fill" the entire 3D image, we combine three consecutive windows together by individually concatenating each signal, resulting in $128 \times 3 = 384$ values for each of the 9 signals. We then construct tuples by taking all the signal values for the same axis in the order of body acceleration, angular velocity and total acceleration, resulting in 3 tuples of length $384 \times 3 = 1152$ each. Each tuple is then truncated to the first 1024 values by discarding the last 128 values in each tuple representing total acceleration values. We find this acceptable since the total acceleration signal remains quasi-constant during one activity, since it generally describes the orientation of the device. The resulting 1024 values from each of the 3 tuples are then re-shaped into a 32×32 area, resulting in a $3 \times 32 \times 32$ signal image. As the image contains data on 5.3 s of movement, cases in which sequential windows did not contain the same activity were discarded. The creation of the 3D images with the time-domain signals is illustrated in Figure 3.

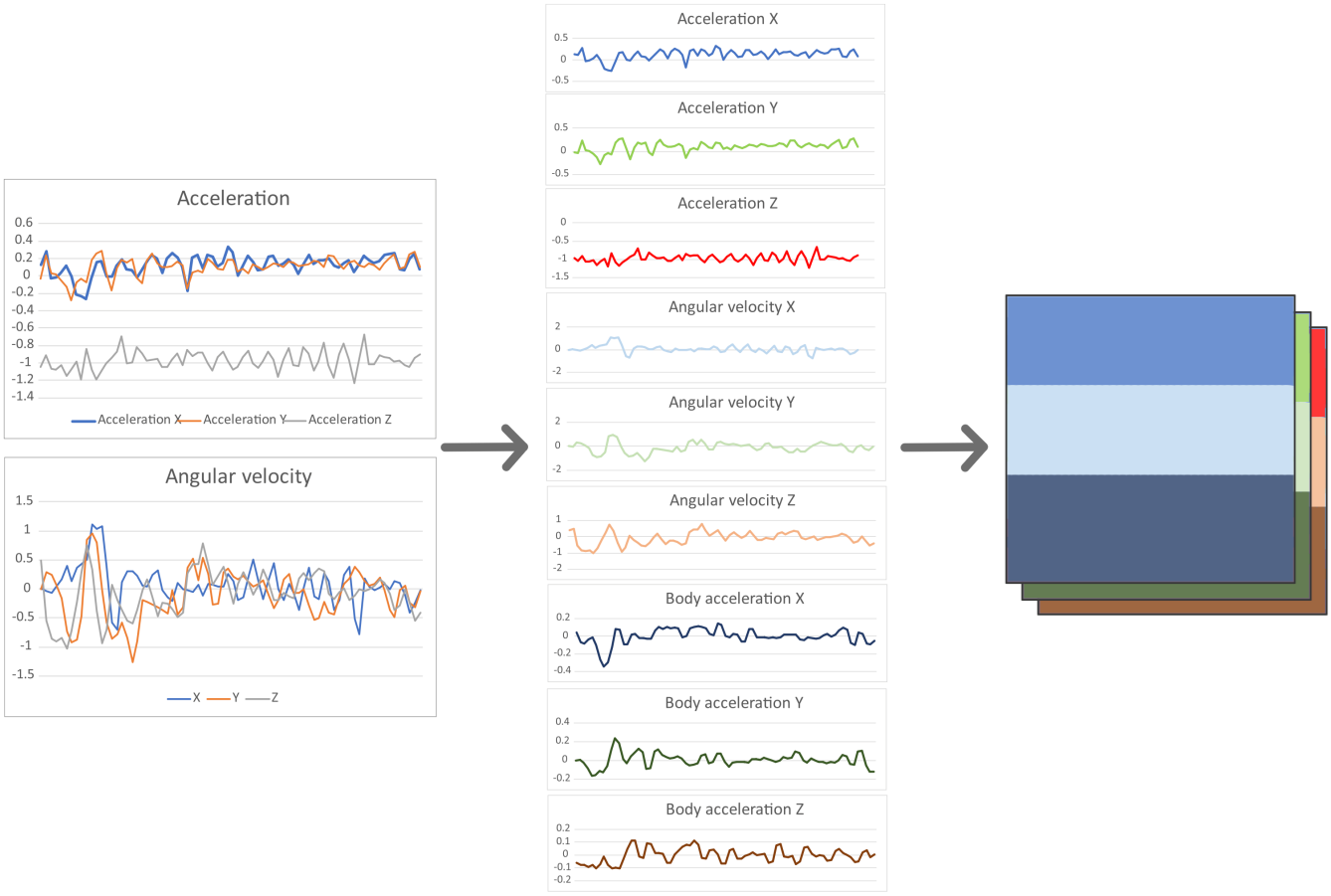


Figure 3. Converting the time-domain acceleration and angular velocity signals into 3D images.

4. Dynamic Neural Network Adaptation Algorithms

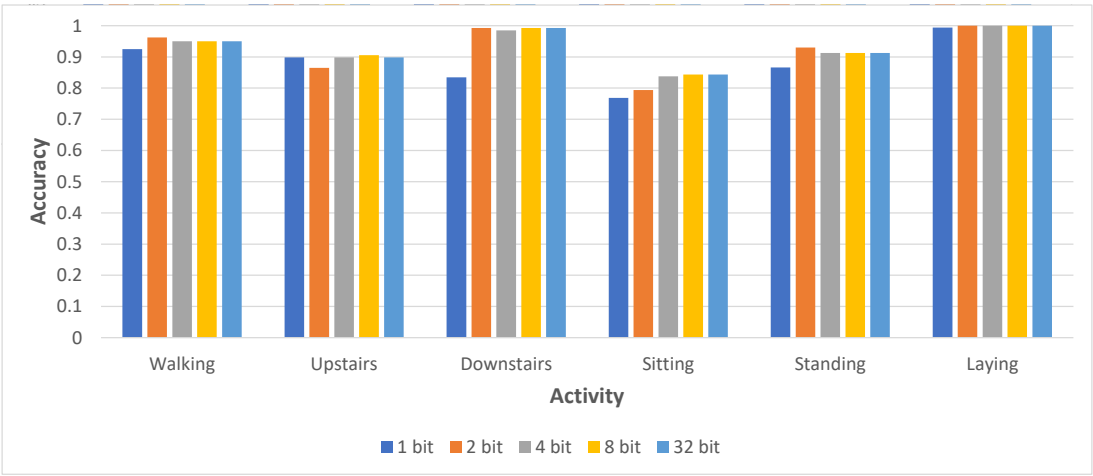
The techniques described in Section 3.1 provide a means for trading off accuracy for reduced resource usage; however, none of the techniques prescribes the manner in which the adaptation should be performed in order to achieve a particular target accuracy or, in accordance to the goals of our work, minimize resource usage with a negligible accuracy loss. The key issue lies in the fact that the decision on how much a neural network should be compressed before processing a particular input needs to be made before the input is classified. Furthermore, even after the classification, we can seldom tell whether the classification was correct or not. In this section, we examine different metrics proxying the performance of a compressed neural network and devise methods for automatic neural network compression adaptation.

4.1. Input Difficulty—Properties Impacting Compressed Classifier Performance

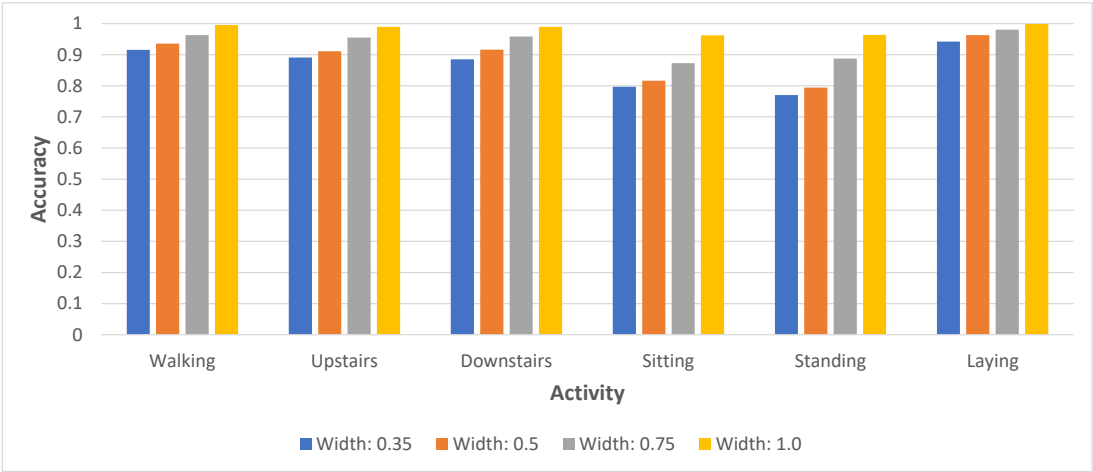
We investigate what factors impact the accuracy of the prediction, since this information plays an important role in selecting the appropriate compression level for the neural network model before inference. As such, we consider the specific physical activity from the UCI-HAR dataset and the user performing it as the specific contextual factors that could make a given input more or less difficult for the network to correctly classify.

We start by analyzing the role played by different activities. For both neural network compression approaches—Any-Precision and SNN—we compute the per-activity average accuracy for all the compression levels and illustrate the results in Figure 4. In the case of Any-Precision, we notice that some activities are almost always easy to classify correctly, regardless of the number of bits used in quantization, such as the case for lying down or going downstairs (here, only the 1-bit quantized model exhibits an accuracy drop). At

the same time, the lowest accuracy is reported for sitting, where the top three quantized networks (4, 8 and 32-bit) all score similarly, at around 85%. Surprisingly, for walking downstairs and standing, the 2-bit quantized model yields the highest accuracy, which could be due to the regularization effect of quantization. In the case of SNN, we notice a more visible correlation between the compression level and the inference accuracy (as we slim the network, the accuracy drops); however, we again notice that some activities are more difficult to be accurately classified (e.g., standing) than others (e.g., laying), regardless of the network width. These results indicate that the appropriate compression level of the network should be chosen based on what we term *the difficulty* of the input, which is influenced by the class to which the data point belongs.



(**a**)

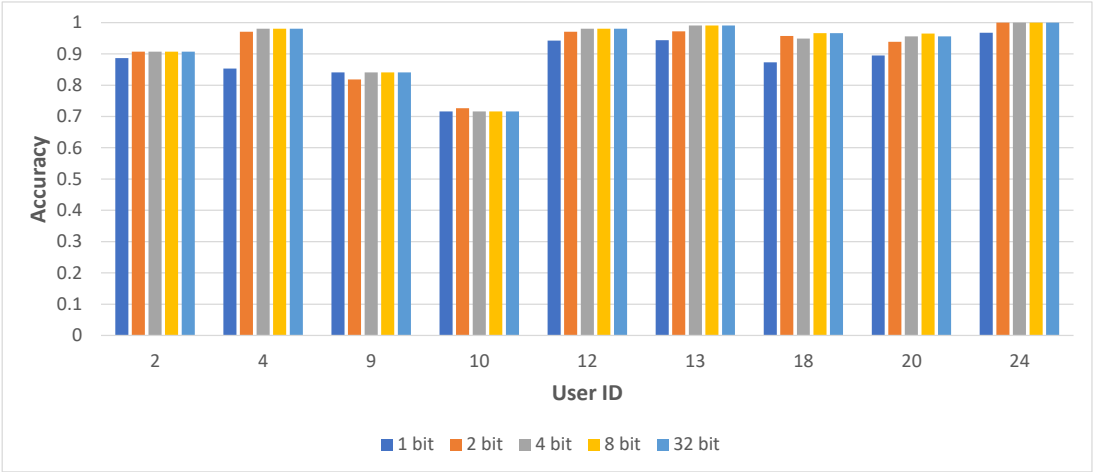


(**b**)

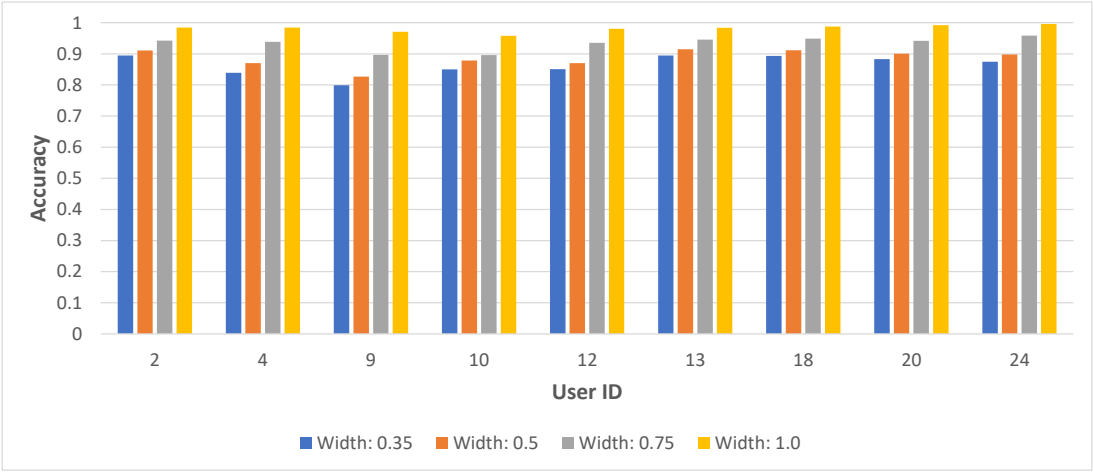
Figure 4. Per-activity accuracy of both networks for various compression levels. (**a**) Any-Precision deep neural networks. (**b**) Slimmable Neural Network.

Next, we investigate the per-user differences in classification accuracy. Figure 5 illustrates the accuracy per user (we select nine users randomly for illustrative purposes) and for each compression level for both networks. For Any-Precision, we notice important variations among users, with user 10 standing out due to a considerably lower accuracy for all networks (just above 70%) compared to the rest of the users (where on average the overall accuracy was higher than 90%). Interestingly, while the more compressed networks usually perform worse, the prominence of this effect is not the same across different users—for instance, the accuracy of inference for User 9 appears to exhibit no variation between

1-bit and 32-bit compression. In the case of SNN, we notice the same linear decrease in accuracy with the increase in slimming that is observable in the per-activity breakdown chart above. This relationship is present for all users, and the variations across users are smaller than in the case of Any-Precision.



(**a**)



(**b**)

Figure 5. Per-user accuracy of both networks for various compression levels. (**a**) Any-Precision deep neural networks. (**b**) Slimmable Neural Network.

The above analysis indicates that the classification success with different levels of compression varies from input to input and is influenced by contextual factors. These, according to our investigation, include the current physical activity and user-inherent factors; however, in different situations, other factors may play a role (e.g., device placement, etc.). Thus, our goal is to *employ the maximum possible compression level that does not hurt the classification accuracy.*

We first conduct a preliminary analysis on the feasibility of *predicting the difficulty of the input*, where a higher difficulty calls for a less compressed network. We aim to use a basic machine learning classifier in order to minimize the computational overhead of difficulty prediction.

Our difficulty predictors are trained as follows. First, we assign a label to each input sample, with a score between 0 and 1 representing the fraction of all the network's compressed variants that achieved correct classification for that particular input. For example, if for Any-Precision and a given input datapoint, 3 out of the 5 levels of quantization correctly classified that input, the label would be 0.6. For SNN, if 3 out of the 4 network widths

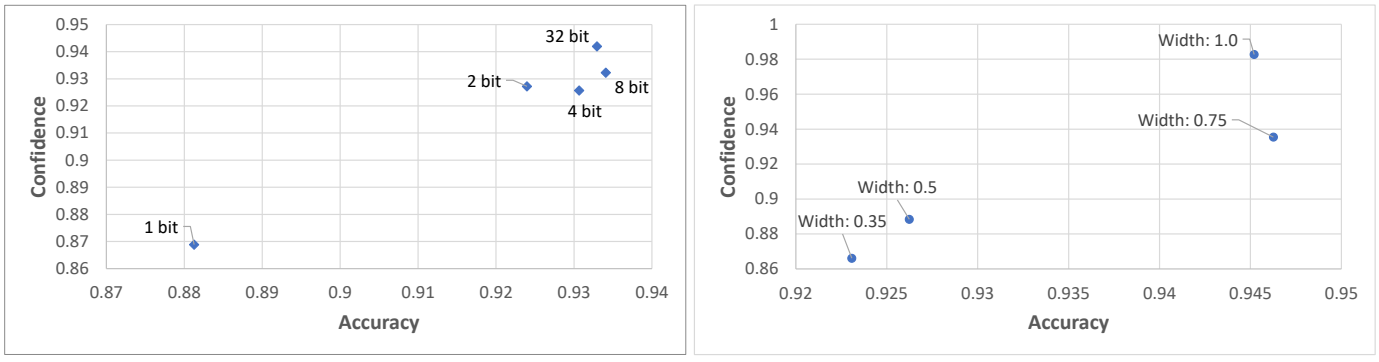
had correct predictions, the label would be 0.75. Next, we use the Orange data mining toolbox [30] and train four basic machine learning models on the data: k-nearest neighbors (kNN), random forest (RF), support vector method (SVM) and linear regression. The goal was to correctly predict the difficulty score for the given input, and for comparison, we also included a dummy regressor which always predicted a constant value. Each input sample is comprised of all the 561 features pre-computed in the UCI HAR database [28]. Finally, all 5 models were evaluated using five-fold cross-validation, and the mean squared error (MSE) and root mean squared error (RMSE) were used to assess the accuracy of the prediction. The results are provided in Table 1 and show that the k-nearest neighbors algorithm delivers the best results in terms of correctly predicting the difficulty score for a given input sample.

Table 1. MSE and RMSE for the five machine learning models trained to predict the difficulty of each input sample.

ML Model	MSE	RMSE
kNN	0.039	0.198
SVM	0.051	0.225
Random forest	0.051	0.226
Linear regression	0.056	0.236
Constant regressor	0.056	0.238

While the above method enables input difficulty prediction, another proxy for this metric comes from the deep neural network itself. In a deep learning classifier, the final layer's softmax function generates normalized classification scores, often termed confidence. The capability of the softmax confidence to accurately reflect the true confidence of the classifier was shown by Gu et al. [31]. However, the authors note that calibration is needed in order to make it highly correlated with the expected accuracy. At the same time, however, Gal and Ghahramani argue that a model can be uncertain in its predictions even with a high softmax output, thus arguing against the term "confidence" [32]. While we acknowledge that the interpretation of this term might be subject to debate, we pursue an analysis of how the softmax confidence scales with the accuracy that a neural network achieves for inputs of different activities when the network is compressed to a varying level.

Figure 6 plots the average confidence vs. average accuracy for the various compression levels for both Any-Precision and SNN on the UCI HAR dataset. This plot shows that there is a positive correlation overall between confidence and accuracy; that is, the network is more confident in the cases where it is also more accurate.



(**a**) (**b**)

Figure 6. Comparison of the neural network accuracy and its softmax-based confidence for different compression levels on the UCI HAR dataset. (**a**) Any-Precision. (**b**) SNN (Slimmable Neural Network).

The relationship is not completely linear, and the accuracy cannot be robustly predicted from the softmax confidence. Nevertheless, there appears to be a certain connection between the softmax confidence and the accuracy. Besides, with the above standard machine learning methods, we examine the use of the network's softmax confidence for guiding the network's compression level.

4.2. Guiding Dynamic Network Compression with kNN

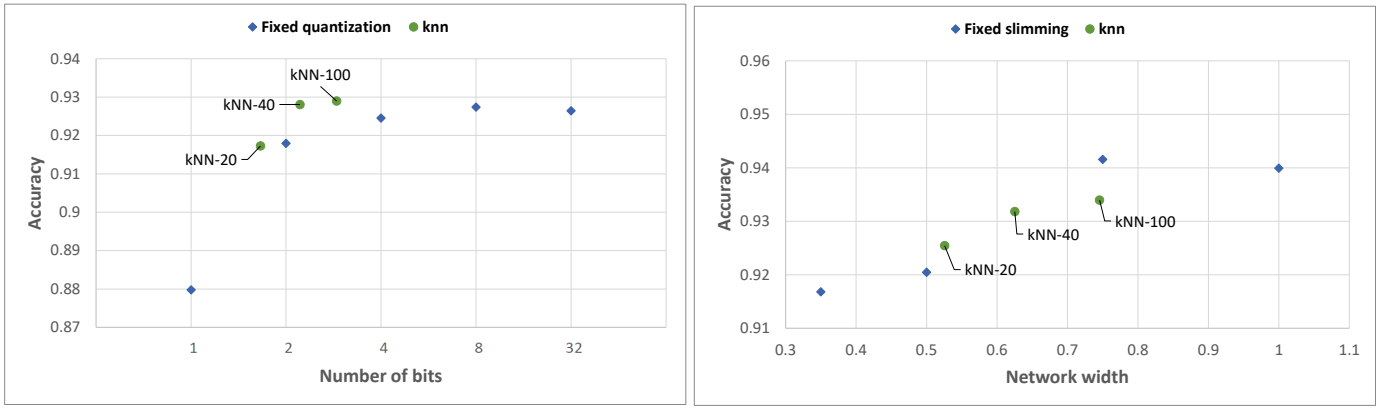
Returning to our goal of *employing the maximum possible compression level that does not hurt the classification accuracy*, we devise a method for automatically selecting the most appropriate DNN compression level for a given input.

We start from the observation that the kNN algorithm can forecast the difficulty of a given input for classification, and we evaluate the kNN's utility for selecting the network's compression level. For each sample from the training set, we compute the HAR prediction outcome (correct or incorrect) with each of the network compression levels employed. During testing, for each input, using the Euclidean distance between the features of the samples, the algorithm finds the k closest examples from the training set and selects the compression level which achieved the highest accuracy. In case of ties, the highest compression level is selected.

Three versions of the kNN algorithm are evaluated—for 20, 40 and 100 nearest neighbors—for both Any-Precision ResNet-50 and SNN MobileNet-V2. We choose these particular network models because they were previously successfully validated with the respective compression techniques [11,12] and were shown to achieve tolerable accuracy drop with increased compression (quantization/slimming).

The results of the kNN algorithm are illustrated in Figure 7 and show the average compression level selected by the kNN method, together with the HAR activity classification accuracy achieved.

In the case of Any-Precision ResNet-50, the results show that, for all flavors of the kNN algorithm, we improve the accuracy–compression trade-off; in addition, the green points remain above the Pareto front of the blue points. More specifically, kNN-driven adaptation achieves a higher classification accuracy compared to the accuracy achieved by the nearest static quantization level. Furthermore, the algorithm using $k = 100$ scores even higher in terms of accuracy compared to any static quantization level. This paradox is explainable by those situations in which a more quantized network (that may be selected by the kNN algorithm) for a given input is more accurate than a non-quantized network.



(**a**) (**b**)

Figure 7. Accuracy of k-NN based compression level selection algorithm. The blue dots show the accuracy that can be achieved with static compression. (**a**) Any-Precision. (**b**) SNN (Slimmable Neural Network).

In the case of SNN MobileNet-V2, we notice a lower performance of the k-NN compression selection algorithm. The algorithm slightly outperformed static slimming for

values of k equal to 20 and 40, but for k = 100, it failed to reach the accuracy achieved by the 0.75-width network.

4.3. Guiding Dynamic Network Compression with Softmax Confidence

The above kNN-based approach may lead to very good results (i.e., simultaneous energy savings and accuracy increase in case of Any-precision), but it does not allow the fine-tuning of the operational point—it always aims for the highest possible accuracy. Often, however, the accuracy may be traded for increased energy efficiency. Thus, we now develop another approach that in a hierarchical manner first determines to which difficulty group an input point belongs and then allows the selection of an appropriate neural network compression level yielding the desired accuracy/energy trade-off.

We now use the softmax confidence to guide the compression level selection algorithm. Based on the near-linear relationship between the accuracy and the confidence, we group inputs with similar prediction confidence together, as this indicates that these inputs are of a similar difficulty, thus tolerating the same level of neural network compression in order to be classified accurately. We empirically establish the number of groups and confidence interval for each group based on the prediction confidence obtained using the most compressed network (i.e., 1-bit Any-precision or 0.35%-width SNN) on the training set. We choose the group boundaries in order to obtain a balanced distribution of the samples in each group. The grouping based on the softmax confidence of the training set input samples for both networks is illustrated in Figure 8.

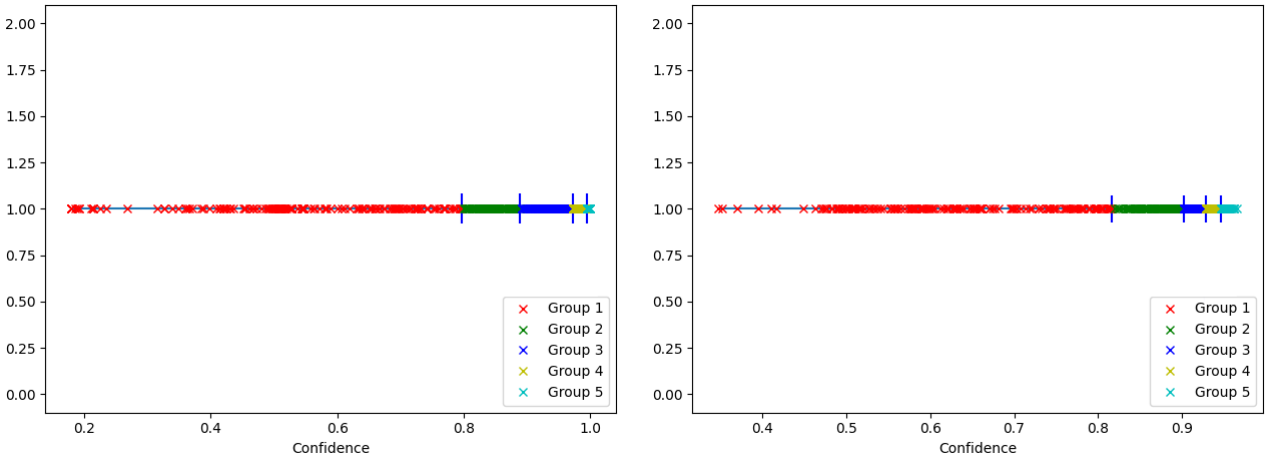


Figure 8. Grouping based on softmax confidence values. (**a**) Any-Precision. (**b**) SNN.

After assigning each training input to a group, we then compute the average accuracy obtained by each network compression level for each of the groups. For each group, we then construct a linear model defining the relationship between the compression levels (which are encoded as numerical values; i.e., 1b–1, 2b–2, 4b–3, 8b–4, 32b–5 for Any-Precision, and 35%–1, 50%–2, 75%–3 and 100%–4 for SNN) and the corresponding accuracy achieved at a certain compression level. To build the model, we use the average accuracy values recorded on the training set when different network compression levels are employed. We then use this model to predict the most appropriate compression level for a given target accuracy.

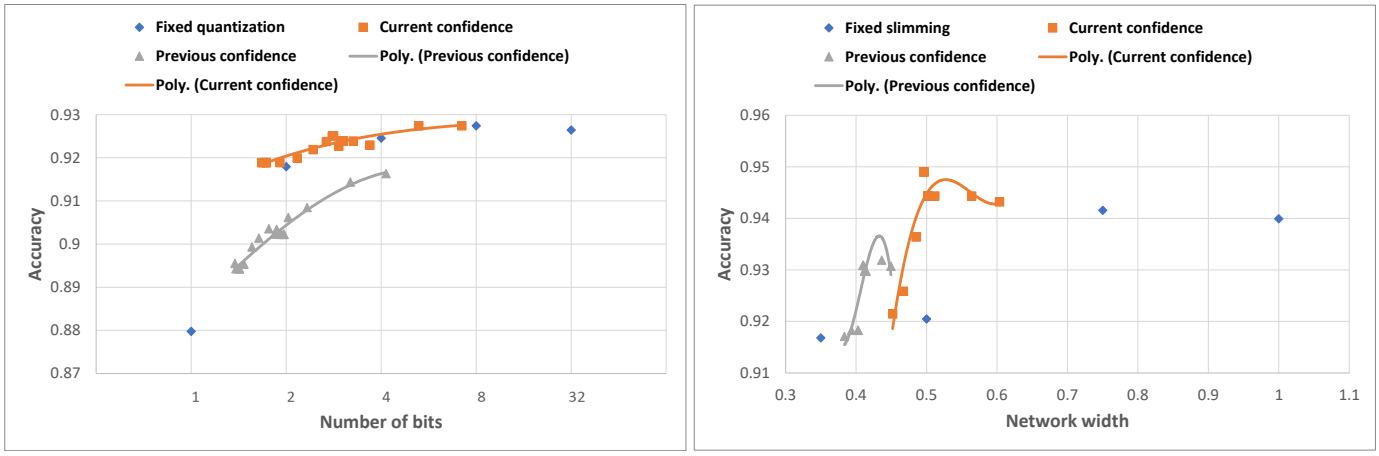
The above set of linear relationships between the expected classification accuracy and the employed compression level that we have constructed for each of the input difficulty groups allows us to quickly identify the most appropriate compression level, should we have the softmax confidence information to hand. Thus, in the initial softmax-based approach, we first classify an incoming sample using the most compressed network model

and then, based on the confidence of that prediction, we assign the input to one of the difficulty groups. If the corresponding linear model indicates that the most compressed model is the most appropriate (for the desired target accuracy), we keep the calculated classification result. Otherwise, we repeat the classification using the compression level indicated by the linear model.

The downside of this approach is that for some input samples the classification will be performed twice—in case the linear model indicates that the most compressed network is the best choice. Repeated classification may induce an overhead, diminishing the energy savings achieved through NN compression. Therefore, we devise an alternative approach where the compression level selection is guided by the confidence of the previous datapoint prediction. We base this approach on the fact that natural human mobility is not characterized by rapid variations; thus, rapid variance among subsequent data points is unlikely [33,34]. To further validate this hypothesis, we compute the correlations between each pair of consecutive prediction confidence values when running the network on consecutive training samples from the dataset, and we obtain a value of $r = 0.397$ indicating a medium-strong correlation [35].

We evaluate both approaches on the UCI HAR dataset for both Any-Precision and SNN. The target accuracy threshold can be adjusted according to the requirements for achieving either a higher accuracy or a higher energy efficiency. In our evaluationl we varied the target accuracy from 0.8 to 0.99 to illustrate how the accuracy–energy savings trade-off translates into actual benefits brought by the dynamic adaptation framework.

The evaluation results are displayed in Figure 9. In the case of Any-Precision Resnet-50, we notice that the first approach (using the current confidence) achieves average accuracy scores slightly higher than those obtained using fixed quantization, but using more compressed networks on average, and thus saving more resources. Using this approach, a maximum average accuracy of 92.7% can be achieved with an average quantization level of 5.6 bits, which is equal to the maximum average accuracy obtainable using fixed 8-bit quantization. The results of the second approach (using the prediction confidence of the previous sample) are worse, however—the average accuracy is 1–2% lower than that obtained using fixed quantization, for the same quantization level. A possible explanation for this could relate to a poor correlation between the confidence of consecutive samples, very likely caused by the fact that grouping intervals are determined using the confidence scores from the most compressed network, while the algorithm selects the group based always on the previous confidence, which is computed using various network compression levels.



(**a**) (**b**)

Figure 9. Results of the compression level selection algorithm based on softmax confidence. The blue line shows the accuracy that can be achieved with static compression. (**a**) Any-Precision. (**b**) SNN.

In the case of the SNN MobileNet-V2, both approaches yield better accuracy results compared to setting the same compression level for a static network. The first approach outperforms even the case of always running the 100% width network, with an average width of the compression levels of 50%. The second approach again underperforms for the same reasons as stated above, but in this case, it still scores better in terms of accuracy than choosing the equivalent static network compression level.

4.4. Guiding Dynamic Network Compression with LDA

So far, we have developed network compression adaptation methods based on the similarity among input datapoints' features; i.e., the kNN approach and based on the grouping of inputs according to the softmax confidence scores. None of the methods is without its weaknesses. The kNN approach requires that the whole training set containing 561-dimensional data points should be stored on the device, which may be prohibitively expensive for resource-restricted mobile devices. The confidence-based approach, on the other hand, may require repeated classifications of the same input, essentially wasting energy.

We therefore develop an alternative approach for grouping inputs in difficulty levels by using a well established statistical method—linear discriminant analysis (LDA). LDA finds a linear combination of features that best separates classes observed in the training set. We develop two flavors of the LDA-based dynamic network compression adaptation algorithm. In both cases, we apply LDA on the space of input samples from the UCI HAR training set. Given that the input samples have a considerable number of features (561) which may result in the overfitting of the LDA algorithm, we reduce the dimensionality of the inputs using the principal component analysis (PCA) to 10 values per input sample before applying LDA.

The first flavor of our algorithm uses LDA to find the subspace that best separates the inputs based on the difficulty of classifying them. To obtain a label that guides the LDA grouping, we classify each sample with all compression levels of the network. For each input, we get a set of predictions, some of which may be correct and some incorrect. We then assign each sample a score between 0 and 1 based on the number of compression levels that correctly classified that sample. We then determine the groups by combining the elements according to their position in the resulting subspace, so that inputs with a similar "difficulty" level (how likely they are to be classified correctly) are aggregated together in the same group. An illustrative example for this "correctness"-based grouping is presented in Figure 10.

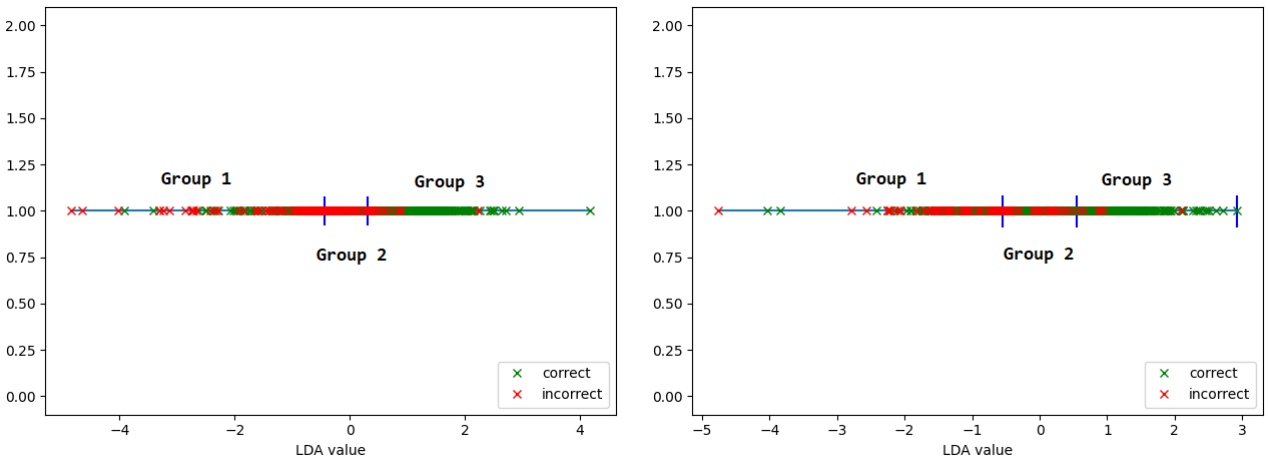
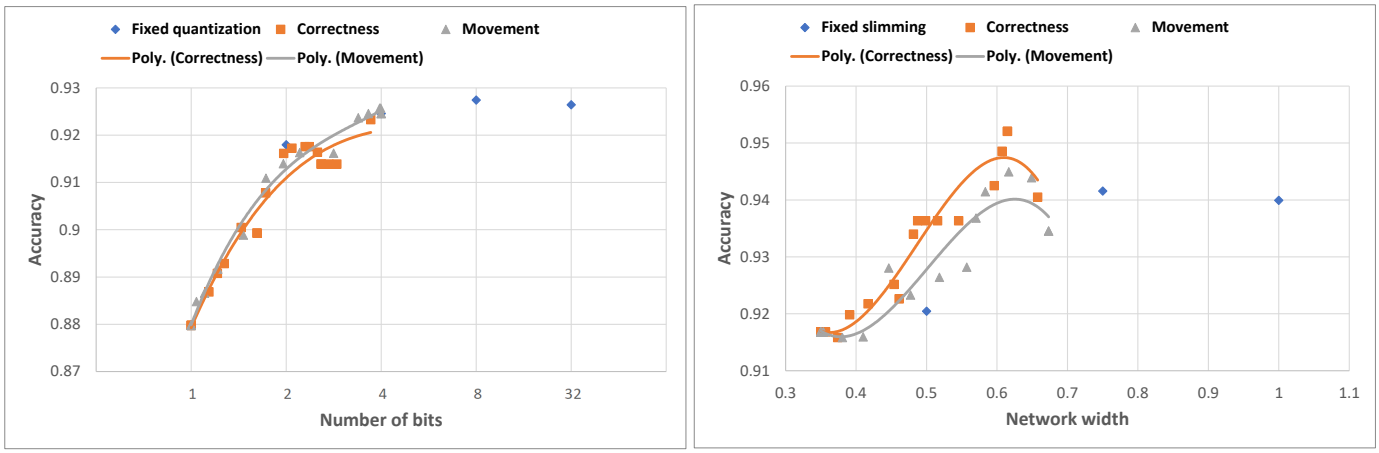


Figure 10. Grouping based on LDA subspace separating inputs by "correctness" of classification. The LDA label for each sample is obtained as the majority vote for correctly vs. incorrectly classified across all the network's compression levels. (**a**) Any-Precision. (**b**) SNN.

In the second flavor of the algorithm, we start from the previous observation that the actual physical activity influences the difficulty of the input and ultimately the accuracy of the network. As such, we apply LDA to separate the input vectors according to their activity class label, where we combine walking, walking upstairs and walking downstairs into *dynamic* and sitting, standing and lying down into *static* classes. The same principle for LDA grouping as with the first algorithm flavor is then applied, resulting in the "movement"-based grouping.

After establishing the groups, in both versions of the algorithm, the approach for constructing and assigning a linear accuracy-compression level model to each group (described in Section 4.3) is used. We evaluated both algorithms on both networks for the UCI HAR dataset, and the results are illustrated in Figure 11.



(**a**) (**b**)

Figure 11. Results of the compression level selection algorithm based on LDA subspace projections. (**a**) Any-Precision. (**b**) SNN.

For Any-Precision ResNet-50, the "movement"-based version of the algorithm scores very similarly in terms of average accuracy and average quantization level to the fixed quantization approach for 1, 2 and 4-bit quantization. However, it does not reach the highest average accuracy achievable with the fixed 8-bit quantization. The "correctness"-based algorithm generally scores slightly lower in terms of average accuracy compared to the "movement"-based counterpart for the same average quantization levels.

In the case of SNN MobileNet-V2, however, the results show that both versions of the algorithm are able to clearly outperform the results obtained using fixed slimming in terms of average accuracy, for all fixed slimming ratios, thus being significantly more resource-efficient. The highest average accuracy of 95.2% is achieved by the "correctness"-based algorithm with an average network width of 61.4%, which is higher than the maximum average accuracy achieved using fixed slimming (94.1% obtained using the 75% width network). The "movement"-based algorithm scores slightly lower in terms of average accuracy than its "correctness"-based counterpart, but still outperforms fixed slimming across all network widths.

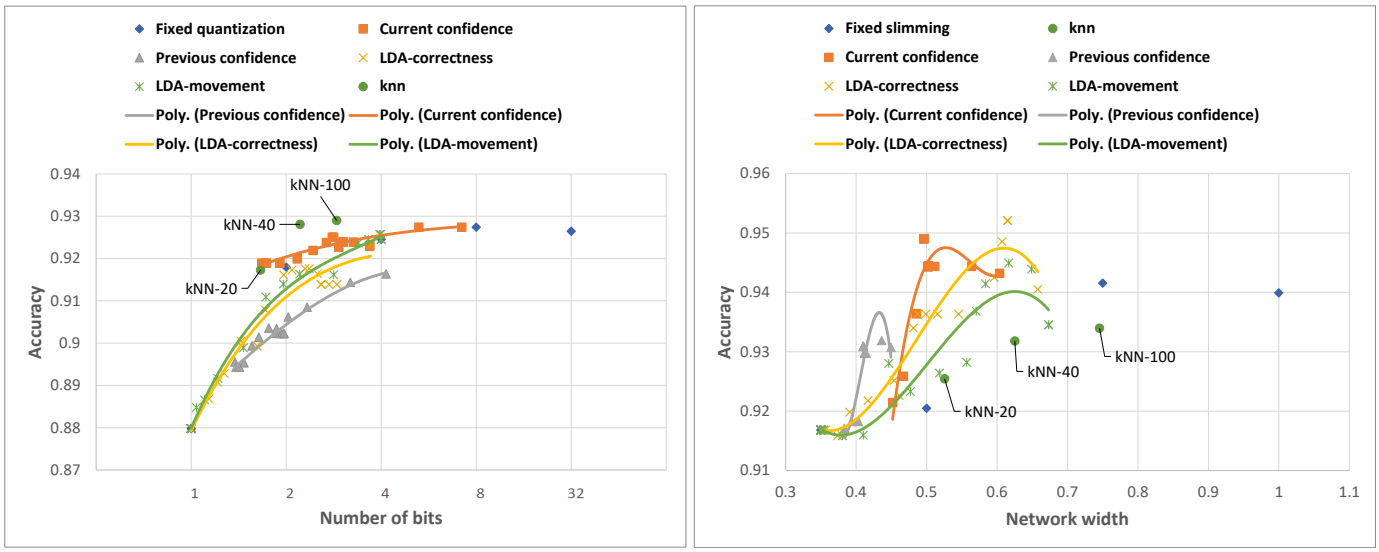
4.5. Comparative Analysis

Following the evaluation of all the proposed algorithms for both Any-Precision and SNN networks, we provide a comparative analysis of the results in Figure 12. We note that the kNN-based adaptation always aims for the best possible accuracy with the smallest amount of resources used, while the confidence-based and LDA-based adaptation allows the fine-tuning of the trade-off between the compression and the inference accuracy. Thus, for the kNN, we observe only individual points in the compression (e.g., number of bit or

network width) vs. accuracy space, whereas the other approaches draw a trade-off line in this space.

We observe that the algorithms perform differently on the two networks: for Any-Precision ResNet-50, compression level adaptation based on kNN-100 delivers an improvement in both the accuracy and the resource usage over any single static compression choice. The remaining algorithms indeed allow the fine-tuning of the accuracy–resource usage trade-off, but they do not appear to deliver the improvements brought by the kNN approach. This is not to say that such approaches do not have their place—indeed, they allow us to achieve intermediate accuracies not supported by the discrete compression levels implemented for a particular neural network.

For SNN MobileNet-V2, the kNN-based approach does not improve the classification accuracy beyond that delivered via static compression. On the other hand, the methods harnessing LDA and confidence-based adaptation not only enable the fine-tuning of the accuracy–compression trade-off but also result in accuracies higher than those delivered by the static compression (or even an uncompressed network).



(**a**) (**b**)

Figure 12. Comparative illustration of the results of the three compression selection algorithms on the UCI HAR dataset, for both Any-Precision ResNet-50 and SNN MobileNet-V2. (**a**) Any-Precision. (**b**) SNN.

When analyzing these results, it is important to take into account the overhead induced by individual algorithms in terms of resource/energy consumption. The softmax confidence-based algorithm involves running the inference on the same input twice for some cases, while the k-NN has the disadvantage of high memory requirements due to the need to store all the training data on the device. As such, choosing the best compression level selection algorithm is not a trivial task and requires taking into account the resources available on the device, the specifics of the application and the context of use. Due to its low storage requirements and the ability to adapt the compression level without the need for multiple classifications of the same data point, in the rest of the paper, we focus on the LDA-based algorithm and proceed with its practical implementation.

5. Dynamic DNN Compression on Mobile Devices

After successfully validating various methods for dynamic DNN compression adaptation on the publicly available UCI HAR dataset [28], we proceed to demonstrate its applicability in a real-world experiment. We focus on the MobileNet-V2 Slimmable Neural Network. Network slimming directly maps to a reduced number of operations and thus is bound to bring certain energy savings. The Any-Precision approach, as discussed in

Section 3.1, relies on quantization, which need not translate to actual energy savings on 32-bit or 64-bit hardware. We also focus on a single dynamic compression adaptation algorithm—correctness-guided LDA—as it provides the balance between the computational overhead and the classification accuracy on the SNN MobileNet-V2 HAR network (see Section 4.5).

5.1. Implementation

We implement our approach on the UDOO Neo Full board, a compact IoT embedded computing device equipped with a three-axis accelerometer and a digital gyroscope. We chose this platform as its processing hardware corresponds to that found in today's low-end smartphones; however, the platform itself runs Linux enabling a quick prototyping of dynamic DNN adaptation (at the time of our experiments, Android's TensorFlow Lite framework did not allow dynamic neural network graph reconfiguration). We sample the acceleration and angular velocity in all three axes from the board's sensors with a frequency of 50 Hz using the Neo.GPIO library [36]. The raw readings were saved to a local file (for subsequent additional analysis) but also went through a processing pipeline similar to the one used in the UCI HAR dataset. This consisted of noise removal followed by the separation of the acceleration into high and low-frequency components, resulting in nine numeric values for each sample: the three total acceleration values (high-frequency components), three angular velocities and three body acceleration values (low-frequency components). The samples were grouped into windows of 128 measurements with an overlap of 64 measurements to match the pre-processing of the original dataset. The measurements were then combined to form 3D images in the same way as described in Section 3, which were then taken as inputs by the SNN.

In addition, from the filtered signals, we also calculated the values of 55 features (see Table 2) that we used as an input for the compression level selection algorithm. This represents a sub-set of the total of 561 features used when evaluating the compression rate selection algorithms in Section 4; we chose only a subset in order to reduce the computational burden given the resource constraints of the UDOO board. The chosen compression rate selection algorithm was the LDA based on "correctness" grouping, which was shown to yield the best trade-off between the accuracy of the classification and resource/energy efficiency following the comparative evaluation of the algorithms performed on the UCI HAR dataset (as shown in Section 4.5). In the experiments, the target accuracy level for the algorithm was set to 92.32%. To select the most relevant features, we used the RReliefF algorithm [37] and evaluated all the features based on how accurately they describe the correctness of the classification. This analysis revealed that the time-domain features yield significantly better results than the frequency-domain features. Hence, we disregarded the computation of the latter and considerably improved the overall computation speed of the feature extraction process.

Consequently, for each input, our algorithm computes the 55 features, applies the compression rate selection algorithm and selects the width of the SNN which then performs activity inference. The entire framework's workflow—i.e., acquisition, pre-processing, compression level selection and activity inference—was performed on the UDOO Neo Full board in real time. The software saved the inference results in a local log file but also printed them to a terminal for real-time inspection via a wireless connection.

Table 2. The 55 features used for inference in the user study.

Nr.	Feature Type	Variables
1–4	Body acceleration—average	X, Y, Z, Magnitude
5–8	Body acceleration—standard deviation	X, Y, Z, Magnitude
9–11	Body acceleration—correlation	XY, XZ, YZ
12–15	Gravity acceleration—average	X, Y, Z, Magnitude
16–19	Gravity acceleration—standard deviation	X, Y, Z, Magnitude
20–22	Gravity acceleration—correlation	XY, XZ, YZ
23–26	Body acceleration jerk—average	X, Y, Z, Magnitude
27–30	Body acceleration jerk—standard deviation	X, Y, Z, Magnitude
31–33	Body acceleration jerk—correlation	XY, XZ, YZ
34–37	Angular velocity—average	X, Y, Z, Magnitude
38–41	Angular velocity—standard deviation	X, Y, Z, Magnitude
42–44	Angular velocity—correlation	XY, XZ, YZ
45–48	Angular velocity jerk—average	X, Y, Z, Magnitude
49–52	Angular velocity jerk—standard deviation	X, Y, Z, Magnitude
53–55	Angular velocity jerk—correlation	XY, XZ, YZ

5.2. User Study Details

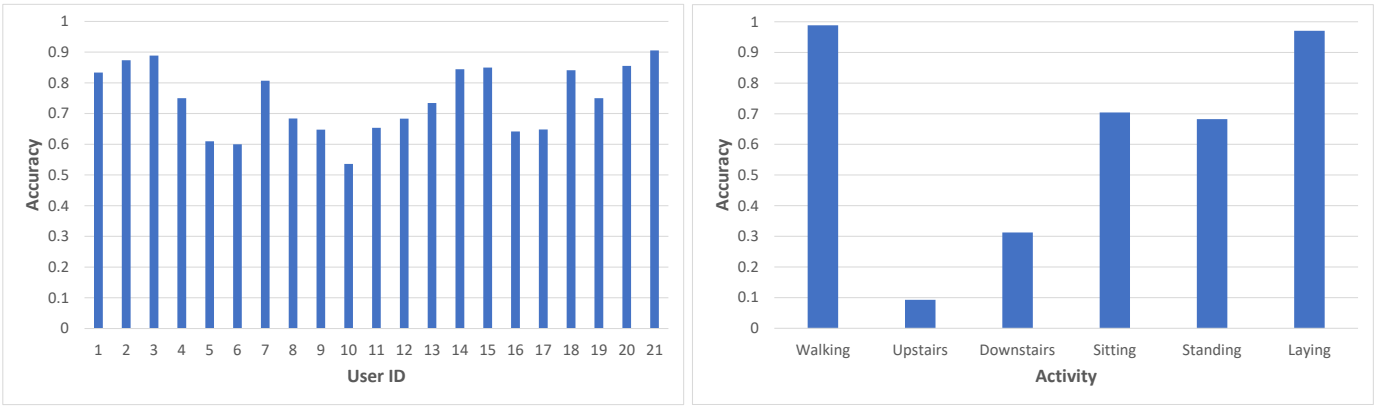
We evaluate the performance of our approach in terms of inference accuracy vs. energy savings enabled by the dynamic DNN compression selection algorithm on the human activity detection task. Our study involved 21 volunteers: 13 male and 8 female participants, with an average age of 29 and a standard deviation of 12 years. Our adaptation approaches were initially developed and evaluated on the UCI HAR dataset (see Section 4; thus, we ensure that our study is conducted under similar conditions to those described in the original study. Namely, the participants were equipped with a battery-powered UDOO board placed to replicate, as accurately as possible, the positioning of the smartphone in the original UCI HAR experiment, and the activities conducted in our guided scenario are the same as those used in the original UCI HAR experiment.

The experiments were conducted in a university campus building at our institution's premises. The volunteers performed all six activities in a row. They first sat in a chair for two minutes without leaning against a table. They then stood still for two minutes, followed by two more minutes of lying down. After completion of these three static activities, the volunteers walked up and down a hallway, which lasted from two to three minutes. This was followed by walking down and up the stairs. In doing so, we were limited by the total number of stairs, so the walk in each direction took about 45 s.

5.3. Experimental Results

Our dynamic mobile deep learning adaptation network achieved an overall average accuracy of 74.6% for all users in this study. This accuracy is lower than that achieved by the framework on the UCI HAR dataset, which is to be expected given that the network was trained on that dataset and now used to infer the activity based on samples collected in a different environment, on a different target group of users and with different hardware. Factors such as the different positioning of the device on the body of the user, different geometry of the chair and stairs compared to those used for creating the dataset may all impact the outcome of the classification, which overall still achieved relatively high accuracy.

In particular, our study showed the framework to exhibit high variations between users, as shown in Figure 13a. For nine users, the average accuracy was over 80% (with a max accuracy of 91%), while for five users, the values were below 70%, with the lowest being 54%.

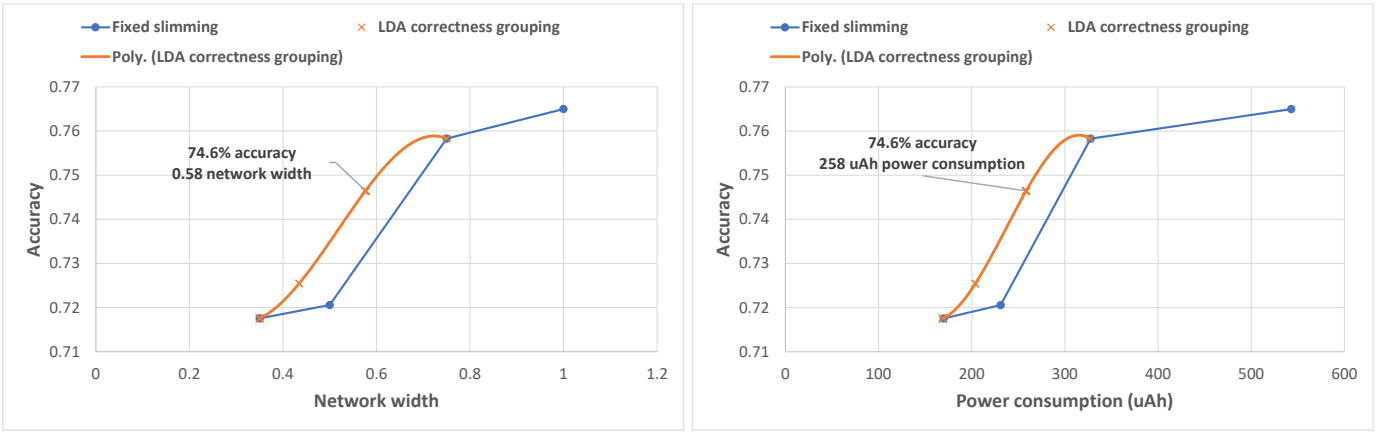


(**a**) (**b**)

Figure 13. Accuracy of the classification for the user study. (**a**) Per user. (**b**) Per activity.

We analyze the accuracy variation among the different activity states and illustrate it in Figure 13b. This analysis also reveals important variations: activities such as lying or walking can be inferred with a high accuracy (almost 100%), while other activities are more difficult to classify correctly. Sitting and standing are often confused by the network with one another and thus exhibit a slightly lower accuracy (around 70%); this can be explained by the similarities in terms of acceleration and angular velocities for these two activities. The worst classification accuracy is reported for walking upstairs and downstairs (under 40%), with one possible reason being the geometry of the staircase used for performing the experiments (a half landing staircase—U shaped—which implies that the user performs a 180° turn every half floor).

We then further explored the analysis of the efficiency of our LDA-based compression level selection algorithm during this user study and investigated the relationship between the average classification accuracy and the average network width and average power consumption, respectively. During the experiments, the sequence of network width selections performed by the algorithm resulted in an average accuracy of 74.6% and an overall average width of the SNN of 58% (see Figure 14a). In terms of accuracy, this scores higher than the accuracy obtained by using a fixed width of either 35% (71.8%) or using 50% of the network (72.1%) and slightly lower than using either the 75% or the 100% fixed width networks (which achieved an accuracy of 75.9%, and 76.5%, respectively).



(**a**) (**b**)

Figure 14. Experimental results obtained for the user study. (**a**) Accuracy vs. Network width. (**b**) Accuracy vs. Power consumption.

In addition, we performed the post processing of the results in order to illustrate the performance of our algorithm for the entire range of possible target accuracy values in more depth. We plot all resulting datapoints together with their corresponding trend-line in Figure 14a and notice that our algorithm outperforms the results obtained by using fixed slimming, achieving a higher accuracy (the trend-line is always above the fixed slimming line) for the same network width, and thus is more resource-efficient.

5.4. Power Consumption Evaluation

In order to assess the power consumption of our LDA-based compression level selection algorithm on the MobileNet-V2 SNN, we measure the power consumption of running inference using each SNN width, together with the additional power consumption required by the adaptation algorithm. We perform the measurements on the same device used during the real-world experiments, the UDOO Neo board, using the Monsoon power monitor tool [38]—a high sampling frequency platform commonly used for power measurements in mobile and embedded computing [39]. The power consumption for performing inference using each of the fixed slimming widths of the MobileNet-V2 SNN is depicted in Figure 14b.

To assess the overhead induced by the LDA-based compression level selection algorithm, we performed power measurements on the UDOO board during inference with the MobileNet-v2 SNN with and without the algorithm and compared the results. Using the algorithm induced no measurable overhead in terms of power consumption, with the resulting power consumption readings for the two cases being almost identical (4.88 mAh vs. 4.87 mAh for running classification on all the UCI HAR test-set samples with and without the compression level selection algorithm, respectively).

Based on these measurements, we were able to assess the energy savings brought by our algorithm during the real-world study. These results, illustrated in Figure 14b, show that compared to to the top two network widths, the adaptation algorithm achieved considerable energy savings: an average power consumption of 258 uAh compared to 327 uAh for the fixed 75% width network and 543 uAh for the fixed 100% network). Analyzing the power consumption across the entire range of possible target accuracy values, we again notice that our algorithm outperforms fixed slimming in terms of energy efficiency: to achieve the same average accuracy, the algorithm consumes less power.

6. Discussion

Our framework enabling self-adaptive mobile deep learning solutions can easily be integrated with any neural network compression method that enables real-time adjustment of the compression level. As such, it has the potential to foster the deployment of neural networks on resource-constrained devices where the operation is hindered by the networks' inherent complexity. By enabling neural networks to operate, albeit approximately, under even scarcer resources, a wide range of embedded devices can develop to become edge computing platforms: from wearables, smart speakers and smart appliances, to earables and smart toothbrushes. Consequently, our approach represents an important contribution to the Artificial Intelligence of Things (AIoT) [40], the newly emerging research area resulting from the synergy between AI and IoT. In this context, our solution enables DL models to self-adapt in real time to the dynamic and varied IoT application scenarios, contexts of use and platform resources (i.e., computation, storage and battery resources) available in each particular use case.

In the experiments described in this paper, the selection of the appropriate network compression level at each inference step was driven by the estimated "difficulty" of the input; however, our algorithms also allow for an environmental adaptation of the network's compression, driven for example in an IoT use case by the constantly changing environmental context including application data, knowledge base, task-related performance requirements and platform-imposed resource constraints. This enables an on-demand model compression which can achieve the desired balance between the model's performance and

the environment's budget, reducing costs and improving computational efficiency with negligible performance degradation.

In addition, our approach's applicability is not limited to resource-constrained embedded computing nodes in the IoT but can be applied in any "sensing" scenario where computing resources or the available energy budget are limited. One such example is Unmanned Aerial Vehicle (UAV)-based Earth Observation imagery applications, where the on-board resource usage directly impacts the flight autonomy of the device. In such cases, our algorithms can self-adapt the network compression to optimize the resource usage based on the *difficulty* of the image for the given task (e.g., in weed detection applications, the crop particularities such as type or growth stage significantly impact the difficulty of the processing task [41], opening the door to important energy savings if *easier* images are processed using the *more compressed* networks).

While we demonstrated our framework on both Any-Precision and Slimmable Neural Networks, only the latter showed its practical viability for achieving important energy savings. For the latter, the compiler-level support for enabling the quantization of the network's parameters to be translated into an actual reduction in the number of bits used for computations at the CPU level is still missing. As of the time of writing this manuscript, the native support for dynamic quantization during inference provided by PyTorch remains limited to 8 bits on CPUs (GPU support is not available). Future research directions could address these limitations through custom implementations using custom machine learning compiler frameworks, such as TVM [27].

7. Conclusions

In this paper, we introduced a framework for dynamic mobile deep learning adaptation which aims to translate the theoretical advantages advertised by the recent neural network compression techniques into real-world resource and energy savings on ubiquitous computing devices.

We exemplify the operation of the framework using two of the most promising neural network compression techniques—Any-Precision [12] and Slimmable Neural Networks [11]—both of which allow the dynamic selection of the compression rate during inference (either the number of bits used for quantization or the width of the network, respectively). As a use case, we choose human activity recognition on a smartphone, and we train both networks on the publicly available UCI HAR dataset [28].

Starting from the analysis on context-related factors impacting the accuracy of the classification and the difficulty of the input, we introduced three algorithms for the dynamic selection of the compression rate during inference (kNN, softmax confidence and LDA subspace projections) and assessed their performance for both neural network compression techniques on the UCI HAR dataset. The three-way comparison of these algorithms confirmed their feasibility in offering a scalable trade-off between the inference accuracy and the resource usage and revealed that selecting the most appropriate algorithm depends on the neural network model and compression technique used, but also on the constraints in terms of hardware resources and context of use.

The user study we conducted to confirm these theoretical findings validated the proposed framework both in terms of the accuracy of the classification and in terms of the efficiency of the compression level selection algorithm compared to using a static neural network compression level. In the study, we configured our framework with the LDA correctness-based grouping algorithm on the SNN MobileNet-V2 network and achieved an accuracy of 74.6% while using only 58% of the network width. Compared to applying a fixed network width of 100%, our framework used 2.18× less energy with only a 1.5% drop in the average accuracy of the classification.

The results obtained and presented in this paper confirm that our self-adaptive approximate mobile deep learning approach has the potential to foster the deployment of deep learning on resource-constrained computing devices by enabling neural networks to

scale their resource usage and thus their energy consumption, according to environmental factors and context of use, with negligible accuracy drops.

Author Contributions: Conceptualization, T.K., O.M. and V.P.; methodology, O.M. and V.P.; software, T.K.; validation, T.K., O.M. and V.P.; formal analysis, T.K. and O.M.; investigation, T.K. and O.M.; resources, T.K., O.M. and V.P.; data curation, T.K.; writing—original draft preparation, O.M.; writing—review and editing, V.P.; visualization, T.K. and O.M.; supervision, O.M. and V.P.; project administration, V.P.; funding acquisition, V.P. All authors have read and agreed to the published version of the manuscript.

Funding: The research presented in this paper was funded by the project "Bringing Resource Efficiency to Smartphones with Approximate Computing" (project number: N2-0136) and "Context-Aware On-Device Approximate Computing", both funded by the Slovenian National Research Agency (ARRS), and by the Slovenian Research Agency research core funding No. P2-0098.

Institutional Review Board Statement: Ethical review and approval were waived for this study, due to the nature of the data collection. More specifically, the study was fully voluntary, the data was fully anonymized, and no personal data were collected in the process. The ethics review board of the Faculty of Computer and Information Science, University of Ljubljana's preliminary opinion was that accelerometer readings from a fully scripted experiment do not warrant in-depth ethical considerations, thus the project was not further examined by the board.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the user study.

Data Availability Statement: The code and collected experimental data are available at: https://gitlab.fri.uni-lj.si/lrk/mobilenn-slimming-and-quantization (accessed on 4 November 2021).

Acknowledgments: The authors would like to thank the volunteers who participated in the user study.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Voulodimos, A.; Doulamis, N.; Doulamis, A.; Protopapadakis, E. Deep learning for computer vision: A brief review. *Comput. Intell. Neurosci.* **2018**, *2018*, 7068349. [CrossRef] [PubMed]
2. Ronao, C.A.; Cho, S.B. Human activity recognition with smartphone sensors using deep learning neural networks. *Expert Syst. Appl.* **2016**, *59*, 235–244. [CrossRef]
3. Young, T.; Hazarika, D.; Poria, S.; Cambria, E. Recent Trends in Deep Learning Based Natural Language Processing [Review Article]. *IEEE Comput. Intell. Mag.* **2018**, *13*, 55–75. [CrossRef]
4. CUDA Zone. Available online: https://developer.nvidia.com/cuda-zone (accessed on 23 September 2021).
5. NVIDIA cuDNN. Available online: https://developer.nvidia.com/cudnn (accessed on 23 September 2021).
6. Chen, J.; Ran, X. Deep Learning With Edge Computing: A Review. *Proc. IEEE* **2019**, *107*, 1655–1674. [CrossRef]
7. Ba, L.J.; Caruana, R. Do deep nets really need to be deep? In *Proceedings of the 27th International Conference on Neural Information Processing Systems—Volume 2*; MIT Press: Cambridge, MA, USA, 2014; pp. 2654–2662.
8. Gou, J.; Yu, B.; Maybank, S.J.; Tao, D. Knowledge distillation: A survey. *Int. J. Comput. Vis.* **2021**, *129*, 1789–1819. [CrossRef]
9. Teerapittayanon, S.; McDanel, B.; Kung, H.T. Branchynet: Fast inference via early exiting from deep neural networks. In Proceedings of the 2016 23rd International Conference on Pattern Recognition (ICPR), Cancun, Mexico, 4–8 December 2016; pp. 2464–2469.
10. Han, S.; Mao, H.; Dally, W.J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv* **2015**, arXiv:1510.00149.
11. Yu, J.; Yang, L.; Xu, N.; Yang, J.; Huang, T. Slimmable neural networks. *arXiv* **2018**. arXiv:1812.08928.
12. Yu, H.; Li, H.; Shi, H.; Huang, T.S.; Hua, G. Any-precision deep neural networks. *Eur. J. Artif. Intell.* **2020**, *1*, 10–37686.
13. Machidon, O.; Fajfar, T.; Pejovic, V. Implementing Approximate Mobile Computing. In Proceedings of the 2020 Workshop on Approximate Computing Across the Stack (WAX), Online, 17 March 2020 ; pp. 1–3.
14. Pineau, J.; Vincent-Lamarre, P.; Sinha, K.; Larivière, V.; Beygelzimer, A.; d'Alché Buc, F.; Fox, E.; Larochelle, H. Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program). *J. Mach. Learn. Res.* **2021**, *22*, 1–20.
15. Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. *arXiv* **2015**, arXiv:1503.02531.
16. Lane, N.D.; Bhattacharya, S.; Georgiev, P.; Forlivesi, C.; Jiao, L.; Qendro, L.; Kawsar, F. Deepx: A software accelerator for low-power deep learning inference on mobile devices. In Proceedings of the 2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), Vienna, Austria, 11–14 April 2016; pp. 1–12.

17. Niu, W.; Ma, X.; Lin, S.; Wang, S.; Qian, X.; Lin, X.; Wang, Y.; Ren, B. Patdnn: Achieving real-time DNN execution on mobile devices with pattern-based weight pruning. In Proceedings of the Twenty-Fifth International Conference on ACM ASPLOS, Lausanne, Switzerland, 16–20 March 2020.
18. Lee, S.; Nirjon, S. Fast and scalable in-memory deep multitask learning via neural weight virtualization. In Proceedings of the 18th International Conference on ACM MobiSys, 2020, Toronto, ON, Canada, 15–19 June 2020.
19. Wu, J.; Leng, C.; Wang, Y.; Hu, Q.; Cheng, J. Quantized convolutional neural networks for mobile devices. In Proceedings of the IEEE Conference on CVPR, Las Vegas, NV, USA, 27–30 June 2016.
20. Yao, S.; Zhao, Y.; Zhang, A.; Su, L.; Abdelzaher, T. Deepiot: Compressing deep neural network structures for sensing systems with a compressor-critic framework. In Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems, Delft, The Netherlands, 6–8 November 2017; pp. 1–14.
21. Yao, S.; Zhao, Y.; Shao, H.; Liu, S.; Liu, D.; Su, L.; Abdelzaher, T. Fastdeepiot: Towards understanding and optimizing neural network execution time on mobile and embedded devices. In Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems, Shenzhen, China, 4–7 November 2018.
22. Liu, S.; Lin, Y.; Zhou, Z.; Nan, K.; Liu, H.; Du, J. On-demand deep model compression for mobile devices: A usage-driven model selection framework. In Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services ACM MobiSys, Munich, Germany, 10–15 June 2018.
23. Huynh, L.N.; Lee, Y.; Balan, R.K. Deepmon: Mobile gpu-based deep learning framework for continuous vision applications. In Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services, New York, NY, USA, 19–23 June 2017; pp. 82–95.
24. Han, S.; Shen, H.; Philipose, M.; Agarwal, S.; Wolman, A.; Krishnamurthy, A. Mcdnn: An approximation-based execution framework for deep stream processing under resource constraints. In Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services ACM MobiSys, Singapore, 26–30 June 2016.
25. Bolukbasi, T.; Wang, J.; Dekel, O.; Saligrama, V. Adaptive neural networks for efficient inference. In Proceedings of the International Conference on Machine Learning (ICML), Sydney, Australia, 6–11 August 2017.
26. Plötz, T.; Guan, Y. Deep Learning for Human Activity Recognition in Mobile Computing. *Computer* **2018**, *51*, 50–59. [CrossRef]
27. Chen, T.; Moreau, T.; Jiang, Z.; Zheng, L.; Yan, E.; Shen, H.; Cowan, M.; Wang, L.; Hu, Y.; Ceze, L.; et al. {TVM}: An automated end-to-end optimizing compiler for deep learning. In Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18), Carlsbad, CA, USA, 8–10 October 2018; pp. 578–594.
28. Anguita, D.; Ghio, A.; Oneto, L.; Parra, X.; Reyes-Ortiz, J.L. A public domain dataset for human activity recognition using smartphones. In Proceedings of the 21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), Bruges, Belgium, 24–26 April 2013; Volume 3, p. 3.
29. Nutter, M.; Crawford, C.H.; Ortiz, J. Design of Novel Deep Learning Models for Real-time Human Activity Recognition with Mobile Phones. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
30. Demšar, J.; Curk, T.; Erjavec, A.; Gorup, Č.; Hočevar, T.; Milutinovič, M.; Možina, M.; Polajnar, M.; Toplak, M.; Starič, A.; et al. Orange: data mining toolbox in Python. *J. Mach. Learn. Res.* **2013**, *14*, 2349–2353.
31. Guo, C.; Pleiss, G.; Sun, Y.; Weinberger, K.Q. On calibration of modern neural networks. In Proceedings of the International Conference on Machine Learning, PMLR, Sydney, Australia, 6–11 August 2017; pp. 1321–1330.
32. Gal, Y.; Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Proceedings of the International Conference on Machine Learning, PMLR, New York, NY, USA, 20–22 June 2016; pp. 1050–1059.
33. Reyes-Ortiz, J.L.; Oneto, L.; Samà, A.; Parra, X.; Anguita, D. Transition-aware human activity recognition using smartphones. *Neurocomputing* **2016**, *171*, 754–767. [CrossRef]
34. Jabla, R.; Buendía, F.; Khemaja, M.; Faiz, S. Balancing Timing and Accuracy Requirements in Human Activity Recognition Mobile Applications. *Proceedings* **2019**, *31*, 15. [CrossRef]
35. Akoglu, H. User's guide to correlation coefficients. *Turk. J. Emerg. Med.* **2018**, *18*, 91–93. [CrossRef] [PubMed]
36. Smerkous, D. Neo.GPIO. Available online: https://github.com/smerkousdavid/Neo.GPIO (accessed on 23 September 2021).
37. Robnik-Šikonja, M.; Kononenko, I. Theoretical and empirical analysis of ReliefF and RReliefF. *Mach. Learn.* **2003**, *53*, 23–69.. [CrossRef]
38. Monsoon Solutions High Voltage Power Monitor. Available online: http://msoon.github.io/powermonitor/HVPM.html (accessed onn 23 September 2021).
39. Schuler, A.; Anderst-Kotsis, G. Examining the energy impact of sorting algorithms on Android: An empirical study. In Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, MobiQuitous '19, Houston, TX, USA, 12–14 November 2019; ACM: New York, NY, USA, 2019; pp. 404–413.
40. Zhang, J.; Tao, D. Empowering Things With Intelligence: A Survey of the Progress, Challenges, and Opportunities in Artificial Intelligence of Things. *IEEE Internet Things J.* **2021**, *8*, 7789–7817. [CrossRef]
41. Wu, Z.; Chen, Y.; Zhao, B.; Kang, X.; Ding, Y. Review of Weed Detection Methods Based on Computer Vision. *Sensors* **2021**, *21*, 3647. [CrossRef] [PubMed]