It's always personal: Using Early Exits for Efficient On-Device CNN Personalisation

Ilias Leontiadis*[†], Stefanos Laskaridis*[†], Stylianos I. Venieris*[†], Nicholas D. Lane[†,‡]
[†]Samsung AI Center, Cambridge [‡]University of Cambridge
*Indicates equal contribution.
{i.leontiadis,stefanos.l,s.venieris,nic.lane}@samsung.com

ABSTRACT

On-device machine learning is becoming a reality thanks to the availability of powerful hardware and model compression techniques. Typically, these models are pretrained on large GPU clusters and have enough parameters to generalise across a wide variety of inputs. In this work, we observe that a much smaller, *personalised* model can be employed to fit a specific scenario, resulting in both higher accuracy and faster execution. Nevertheless, on-device training is extremely challenging, imposing excessive computational and memory requirements even for flagship smartphones. At the same time, on-device data availability might be limited and samples are most frequently unlabelled.

To this end, we introduce PersEPhonEE, a framework that attaches early exits on the model and personalises them on-device. These allow the model to progressively bypass a larger part of the computation as more personalised data become available. Moreover, we introduce an efficient on-device algorithm that trains the early exits in a semi-supervised manner at a fraction of the whole network's personalisation time. Results show that PersEPhonEE boosts accuracy by up to 15.9% while dropping the training cost by up to 2.2× and inference latency by 2.2-3.2× on average for the same accuracy, depending on the availability of labels on-device.

ACM Reference Format:
Ilias Leontiadis, Stefanos Laskaridis, Stylianos I. Venieris, Nicholas D. Lane. 2021. It's always personal: Using Early Exits for Efficient On-Device CNN Personalisation. In *The 22nd International Workshop on Mobile Computing Systems and Applications (HotMobile '21), February 24–26, 2021, Virtual, United Kingdom.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3446382.3448359

HotMobile '21, February 24–26, 2021, Virtual, United Kingdom
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8323-3/21/02…$15.00
https://doi.org/10.1145/3446382.3448359

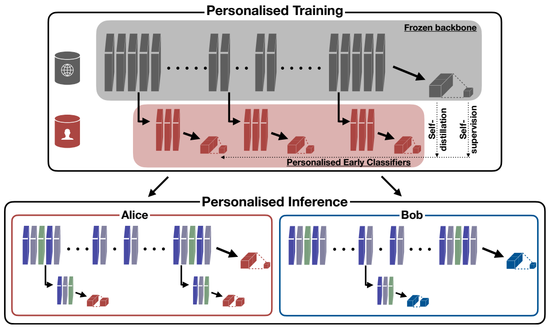


Figure 1: Overview of early-exit personalisation.

1 INTRODUCTION

The recent progress of deep learning has enabled the development and ubiquitous deployment of numerous novel systems and apps. At the same time, mobile chipsets are increasingly equipped with dedicated processing units (*e.g.* NPUs) for efficient execution of DNNs [1, 11]. As such, on-device execution is becoming an emerging approach for meeting the latency, energy and privacy requirements of such systems.

Typically, deep models are trained centrally on standard datasets with the aim to generalise across samples and users. For instance, facial landmark detectors are trained to capture various demographics, speech recognisers to accommodate different accents and voices, and home-assistant robots to work reliably across diverse household configurations. However, with data not being independent and identically distributed (IID) across devices in reality, the global model results in varying accuracy across the samples encountered in the wild, often failing catastrophically on unexpected inputs.

Conventional solutions typically rely on cloud- or edge-based setups to perform *model personalisation*. Under this scheme, each device transmits the user-specific data to a remote server, where personalisation takes place through additional training rounds. Although this approach takes advantage of the resource-rich server, it comes with significant overheads. First, the exposure of both the user data and the resulting personalised model to the remote side raises privacy concerns [16, 18, 21, 23]. Second, on the service-provider side, using cloud/edge infrastructure comes with significant operating costs [24], due to the high demand of DNN training workloads for compute, memory and bandwidth resources.

A promising approach to remedy this situation is *on-device model personalisation*, aiming to tailor the DNN to each individual user or context. Despite the algorithmic progress, there are still important limitations. First, the excessive resource demands of DNN training and the device hardware heterogeneity makes on-device personalisation challenging. Second, user data are frequently unlabelled, making supervised learning impossible. Third, personalisation can result in undetected catastrophic forgetting for non-frequent inputs.

To address these limitations, we propose PersEPhonEE, a novel framework that converts a pretrained CNN into a multi-exit network and personalises its early exits to the specific user's data distribution (Fig. 1). Personalisation takes place purely on-device and aims at producing classifiers along the depth of the network that are specialised for the user's data. At inference time, the network can exit early if it is confident on its early output, or progressively refine the quality of the result. Training can take place with or without ground-truth labels in a self-supervised manner, using the output of the network's last exit. By only personalising the early classifiers and keeping the backbone network frozen, we render the training process lightweight enough to take place overnight, while the device is plugged in. Finally, as the backbone is not altered, we can assess the quality of each personalised classifier and capture out-of-distribution samples at run time.

We evaluate our system across two networks and datasets. Results indicate that we can achieve up to 3.2× inference throughput gain, 3.1× fewer FLOPs and 25.1× fewer parameters while maintaining similar accuracy. Moreover, training an exit can be up to 23× faster than the whole network.

2 BACKGROUND AND RELATED WORK

Recently, an increasing body of work has focused on the design of early-exit networks, *i.e.* DNNs with intermediate classifiers along their depth that provide varying accuracy-latency trade-offs. The goal of this class of models is to offer adaptive accuracy-latency behaviour, either through an input-dependent execution with each sample stopping at the appropriate exit based on its difficulty or by extracting a subnetwork. Existing efforts span from hand-crafted early-exit models [10, 28] to model-agnostic [13, 25] and deployment-optimised frameworks [14, 15]. Focusing on transfer learning, REDA [12] employs self-distillation to efficiently adapt earlier exits to tasks from different domains. While this line of work focuses on speeding up inference by training a single global model offline, PersEPhonEE attempts to personalise exits in order to build a user-specific network that can save both computation and energy. Tangentially, [26] specialises CNNs by dynamically dropping layers based on *offline* class clustering. Nevertheless, to repurpose it for personalisation, the number of users and the most common classes for each user need to be known *a priori*, leading to poor scalability.

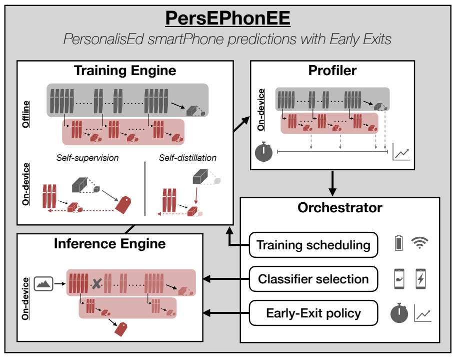


Figure 2: PersEPhonEE's system architecture.

On the personalisation front, several methods have been proposed, leveraging user data to specialise generic pre-trained models [4, 20]. Currently, this space is dominated by meta-learning approaches [9]. Such algorithms train a DNN so that it ends up with a good parameter initialisation that is amenable to rapid specialisation without requiring an excessive amount of data. Our approach, instead, uses personalisation as a way of progressively accelerating both on-device inference and training as more user-specific data become available. Orthogonal meta-learning techniques can be combined with PersEPhonEE to further improve performance. Resembling PersEPhonEE's self-supervised technique, another line of work employs online model distillation to obtain a user-adapted lightweight model [19]. The proposed two-model method is tailored for videos and requires the separate and frequent execution of both a costly teacher model and a lightweight student. Although this is suitable for high-end platforms, it is prohibitive for mobile devices.

3 SYSTEM ARCHITECTURE

PersEPhonEE aims at creating an efficient personalised multi-exit model for each user. The processing flow (Fig. 2) starts by attaching M intermediate classifiers, or early exits, along the depth of a given CNN. Next, the *Training Engine* trains both the early exits and the backbone network if it has not been pretrained. Upon deployment to a user's device, the *Profiler* collects statistics on the on-device execution. The *Orchestrator* examines the *Profiler*'s accumulated data and dictates when an on-device personalisation round will be launched. After a personalisation round, the *Orchestrator* considers the accuracy-latency characteristics of each early exit and customises the execution by configuring the *Inference Engine* with the selected exits and the early-exit policy to be used when processing new incoming data.

3.1 Training Engine

The *Training Engine* comprises a dual structure: *i)* an *offline* component (Sections 3.1.1 & 3.1.2) that derives a global multi-exit variant given a CNN and trains it using a generic dataset and *ii)* an *on-device* component (Section 3.1.3) that adapts the global multi-exit model to work well for the current user.

3.1.1 *Deriving a multi-exit model.*

Given a CNN model, PersEPhonEE converts it into a multi-exit network by attaching a number of early exits along its path. For the exit architecture, we utilise a uniform design for all early exits, adopting the intermediate classifier structure of MSDNet [10]. Furthermore, we follow a platform-agnostic approach with equidistant placement of the M classifiers along the depth of the backbone network in terms of FLOP count, *i.e.* at $i/(M+1)$-th, where $i \in [1, M]$ is the ordinal of the classifier.

3.1.2 *Training the global model.*

The multi-exit global model is then trained offline by utilising a generic training set for the target task. If the supplied backbone network has been pretrained, we apply early-exit-only training, thus freezing the backbone's parameters and training only the intermediate classifiers' layers [15]. If the supplied CNN is not trained, we jointly train the backbone and intermediate exits from scratch using the cost function introduced in [13]. In terms of overhead, training from scratch the multi-exit model spans between 1.2×-2.5× the time of the backbone network, depending on the architecture and number of exits, with higher overhead when $\frac{FLOPs_{\text{early exit}}}{FLOPs_{\text{backbone}}}$ is larger. Nonetheless, as training takes place once for all users upfront, it is rapidly amortised by the runtime gains of confident samples from early exiting. The resulting model is the trained *global* model, serving as initialisation for client devices to personalise. The global model is then deployed to the mobile and embedded devices.

3.1.3 *On-device early-exit personalisation.*

On-device personalisation is hindered by the limited compute, storage and memory capabilities of mobile devices. Moreover, device diversity calls for more elastic models that can adapt computation and energy usage. At the same time, obtaining ground-truth labels for user data is often impractical. Finally, prolonged fine-tuning can lead to catastrophic forgetting.

To overcome these limitations, PersEPhonEE introduces personalised multi-exit networks and an efficient on-device training scheme. We adopt a *frozen-backbone* training approach that updates the parameters of the early exits only (Fig 2). This strategy has a twofold gain: First, early exits and their gradients only occupy a fraction of the global model's memory usage and require a significantly lower number of FLOPs at training time. Second, as the network path to the last exit remains unmodified, the original output of the global model is maintained as a fail-safe, partly counteracting catastrophic forgetting by allowing for situations where the input sample does not follow the user-specific distribution.

To remedy the shortage of ground-truth labels in realistic scenarios, we design an objective function that personalises the multi-exit network using: *i)* supervision, *ii)* self-supervision or *iii)* self-distillation. When no hard labels are available, either the soft labels (*i.e.* the softmax distribution) (*self-distillation*) or the top-1 prediction (*self-supervision*) from the last exit – which is typically the most accurate classifier in the global dataset – can be used to "teach" the early exits. The intuition is that the early exits, despite having smaller learning capacity, can progressively approach the accuracy of the last classifier for the *personalised* input (*i.e.* a subset of the input distribution). If ground-truth labels are available, we can leverage them (*supervision*) to further fine-tune the exits and, thus, achieve even higher accuracy.

To this end, we introduce a hybrid loss function that combines the three training schemes with a tunable weighting. Specifically, we define the personalisation loss of exit i as:

$$\begin{aligned}\mathcal{L}^{(i)}_{\text{personal}}(\hat{\boldsymbol{y}}^{(i)}_T, \hat{\boldsymbol{y}}^{(M+1)}_T, \boldsymbol{y}) = \quad & \quad (1)\\ excl(\alpha,\gamma)\cdot\mathcal{L}^{(i)}_{\text{superv}}\left(\hat{\boldsymbol{y}}^{(i)}_{T=1}, \boldsymbol{y}\right) & \\ +\,\beta\cdot\mathcal{L}^{(i)}_{\text{self-distill}}\left(\hat{\boldsymbol{y}}^{(i)}_T, \hat{\boldsymbol{y}}^{(M+1)}_T\right) & \\ +\,excl(\gamma,\alpha)\cdot\mathcal{L}^{(i)}_{\text{self-superv}}\left(\hat{\boldsymbol{y}}^{(i)}_{T=1}, \text{top1}(\hat{\boldsymbol{y}}^{(M+1)}_{T=1})\right) &\end{aligned}$$

where $\mathcal{L}^{(i)}_{\text{superv}}$ is the cross-entropy loss between the hard label, $\boldsymbol{y}$, and the output of the exit i, $\hat{\boldsymbol{y}}^{(i)}_{T=1}$, $\mathcal{L}^{(i)}_{\text{self-distill}}$ is the KL divergence between the output of exit i, $\hat{\boldsymbol{y}}^{(i)}_T$, and the last exit, $\hat{\boldsymbol{y}}^{(M+1)}_T$, smoothed by temperature T [8], $\mathcal{L}^{(i)}_{\text{self-superv}}$ is the cross-entropy loss treating the last exit's top-1 prediction as the ground-truth label, and $excl(x,y) = (x = 0 \lor y = 0)$ ensures that x and y are mutually exclusive. Hyperparameters α, β and γ determine the importance of the three components, weighing the supervised, self-distillation and self-supervised loss, respectively. By replacing cross-entropy with another loss, the *Trainer* can also target other tasks such as regression. Finally, the overall training objective is to minimise the sum of losses of all M exits across the user-specific data.

3.1.4 *Confidence calibration.*

For the early-exit policy, we estimate a classifier's *confidence* for a given input using the top-1 output value of its softmax layer [6], defined as $\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$ for the i-th classifier. An input takes the i-th exit if the prediction confidence is higher than a tunable threshold, thr_{conf}, following $\arg_i\{\max_i\{\text{softmax}_i\} > thr_{\text{conf}}\}$. The exact value of thr_{conf} provides a trade-off between the latency and accuracy of the multi-exit model and determines the *early-exit policy*. If none of the classifiers reaches the confidence threshold the original (last) classifier is used as a fail-safe for non-personalised inputs.

PersEPhonEE re-calibrates its confidence threshold after every personalisation round. This is performed by maintaining a calibration set, obtained from the user's input distribution. After the early-exit personalisation, the calibration set is used to measure the average loss of each exit with respect to the original model output, and the exit rate of each exit which will affect the overall average inference latency. With these measurements, PersEPhonEE performs a Pareto-front analysis in the accuracy-average latency space and chooses the smallest threshold that maintains the model's accuracy within a specific drop tolerance. Furthermore, early exits that do not contribute much to the user-specific distribution (*e.g.* they have a high loss) are pruned.

3.2 Inference Engine

The *Inference Engine* is responsible for executing the forward pass of the multi-exit network over a supplied set of input samples. These are either incoming samples to be classified or stored past samples to be used for personalisation. The *Orchestrator* configures the *Inference Engine* by defining which early exits to use and the early-exit policy, *i.e.* the threshold above which an exit's prediction is considered confident.

3.3 Profiler

In order for PersEPhonEE to make informed decisions about the cost (*i.e.* latency, memory usage) and benefits (*i.e.* accuracy) of using each classifier, we need to measure these metrics for a personalised model on the target device. For this reason, the *Profiler* gets invoked after a training session has completed, while the phone is still plugged in. Specifically, the profiler conducts one forward pass of the calibration set, estimating *1)* the latency, *2)* top confidence and *3)* accuracy of each classifier (i), using the final prediction as the ground-truth if no hard labels are present. Subsequently, these metrics are passed to the *Orchestrator* (Section 3.4) to decide the early-exit policy during inference.

3.4 Orchestrator

The *Orchestrator* is a key system component that measures the run-time performance and is responsible for scheduling the different operating phases of PersEPhonEE and configuring the *Inference Engine*. Three basic phrases are defined:

Inference phase: The selected exits and threshold are used to configure the *Inference Engine*, saving computation time and energy on the new samples.

Exploration phase: PersEPhonEE periodically re-evaluates the quality of the early exits in the background, with a given probability p_{expl}. This is achieved by stochastically extracting the output of the last classifier and calculating the loss function for each of the selected early exits. If the average loss of an intermediate classifier is consistently higher, there has probably been a domain shift. To mitigate this, we raise the confidence threshold to thr'_{conf} and schedule the trainer to run on the newly encountered samples when the device is plugged in. The values of p_{expl} and thr'_{conf} can be selected based on the transient load and battery level of the device as well as the loss increase of the early classifier.

Personalisation phase: The on-device personalisation task is scheduled overnight [2] and only when there are sufficient new data on the device (Section 4.3) or when the active threshold (Section 4.4.2) thr'_{conf} deviates significantly from the validation set threshold thr_{conf}, as this indicates that the user input distribution might have shifted since the last time the exits were trained. Given the aforementioned conditions, personalisation typically takes place on infrequent intervals, *e.g.* monthly, and only after the device is plugged to a power adapter and fully charged. Hence, the personalisation energy cost is minimal compared to the resulting inference energy savings during normal usage, *i.e.* when on battery power.

4 EVALUATION

We have developed PersEPhonEE on top of *PyTorch* and *torchvision* and targeted two CNNs, namely ResNet-50 [7] and MobileNetV2 [22]. All reported measurements are taken on an Nvidia Jetson Xavier board equipped with a Quad-core Arm Cortex-A57 and a dual-core NVIDIA Denver2 CPU. We only use the embedded CPUs as they are comparable to today's flagship phones. For our experiments, we attach six early exits ($M = 6$) in addition to the original final classifier, giving seven total possible output locations. We pretrain the global model (including the exits) on the global labelled dataset in an offline manner and we only personalise early classifiers on-device with or without the presence of ground-truth labels, leaving the final output untouched. We fine-tune the exits for 10 epochs with learning rate 0.01.

4.1 Datasets

To demonstrate the ability of early exits to personalise we evaluate PersEPhonEE on two different datasets:

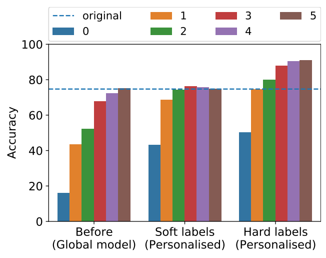
Personalised ImageNet: Based on ILSVRC 2012 [5], this is a personalised variant created with the assumption that not all users experience all 1000 labels with the same frequency. To this end, each personalised user is assumed to be exposed to images with labels whose frequency follow a Gaussian distribution to better capture such biases [27]. After generating the user-specific label popularity, we sample images from ImageNet to create each user's personalised dataset.

FEMNIST: This is a real-world dataset aimed at hand-written text recognition [3]. It contains 80K 28×28 images from 3.5K users (225 images per user) belonging to 62 different classes (10 digits, 26 lowercase, 26 uppercase letters). We mix the data from 3250 random users to train the global model and evaluate the personalisation results on each of the remaining 250 users, holding out 100 images per user for testing.

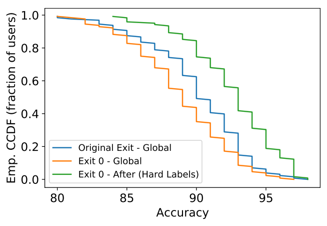
4.2 Accuracy on Personalised Exits

First, we evaluate the impact of on-device personalisation on the accuracy of individual early exits, using ResNet-50 on ImageNet and MobileNetV2 on FEMNIST (Fig. 3).

Fig. 3a-Before shows the accuracy of the *pretrained* exits on the personalised ImageNet dataset, whereas the dotted line shows the original model's accuracy. As expected, the

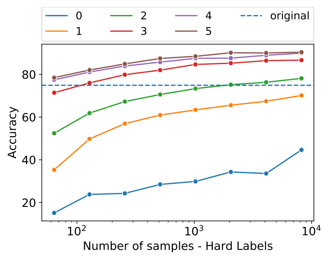


(a) Personalised ResNet-50 on ImageNet.

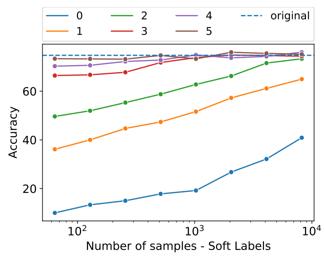


(b) Exit-0 of personalised MobileNetV2 on FEMNIST.

Figure 3: Accuracies per early classifier.

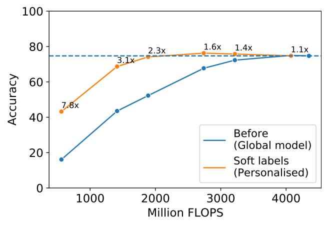


(a) Hard labels

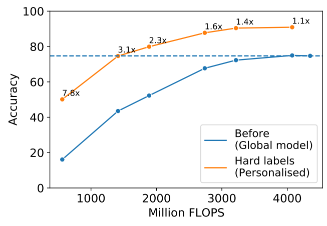


(b) Self-distillation

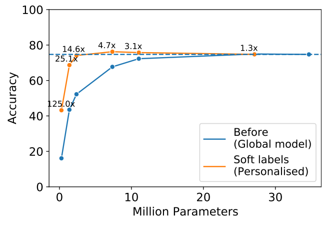
Figure 4: Accuracy vs. #personalisation samples.



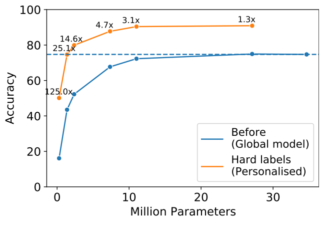
(a) FLOPs (Self-distillation)



(b) FLOPs (Hard labels)



(c) Params. (Self-distillation)



(d) Params. (Hard labels)

Figure 5: Accuracy vs. #parameters and FLOPs for each exit, before and after personalisation.

earlier the exits are attached to the backbone, the lower the accuracy is, ranging from 16% for exit-0 to 74.95% for exit-5, similar to the original top-1 accuracy of 74.96%. By personalising the early exits on-device with user-specific input, we can greatly improve their accuracy for similar (personalised) inputs. Under the presence of ground-truth labels (Fig. 3a-Hard Labels), PersEPhonEE applies supervised personalisation with exit-1 reaching the original accuracy and pruning a large part of the computation. Later exits attain even higher accuracy than the global model (up to 90.9% for exit-5) as they are now specialised for the user's frequently seen labels. When no labels are available on-device, PersEPhonEE uses self-distillation. In this case, although the early exits do not surpass the accuracy of the global model, some of the them still reach its accuracy significantly faster (*e.g.* exit-2).

Fig. 3b depicts a different view, where we fix the personalised classifier and look into the empirical CDF of users reaching a specific accuracy. It can be observed that by using supervised personalisation, the same users can achieve higher accuracy faster. For instance, 75% of the users attain 90% accuracy at exit-0, while without personalisation this was for 40% and 60% for the non-personalised final exit.

4.3 Training Samples Required

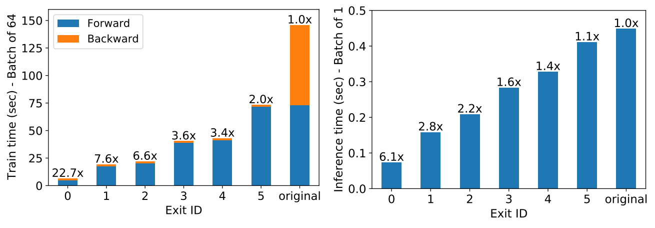
Different users might need different early-exit policies as they might have varying amounts of data to train the model as well as different input complexity. Therefore, it is important to understand how much data are required to train a personalised multi-exit model. Fig. 4, depicts the accuracy achieved per early exit for varying number of samples. First, we note that the accuracy of every exit is gradually improving as more data become available. Moreover, the earlier the exit, the larger the improvement. As a result, different exits can be used for users with different amount of data. For example, users with 2K samples or more could utilise exit-2 to achieve similar performance to the original model, whereas users with solely 150 samples can use exit-3, still saving more than 60% of the computational cost.

4.4 Performance Evaluation

4.4.1 Performance - Accuracy Trade-off. Here, we correlate the achieved accuracy of our personalised early-exit model with the amount of computation (FLOPs) and memory (#parameters) required. In Fig. 5, we depict this relationship for each of the exits. With ground-truth (hard) labels, we observe that the personalised classifiers can reach the original accuracy with 25.1× fewer parameters (Fig. 5b) and 3.1× fewer FLOPs (Fig. 5d). Furthermore, accuracy is improved by 13 percentage points (pp) (to 87.8%) while using 4.7× fewer parameters and 1.6× fewer FLOPs, resulting in a configuration that provides significant speedup, energy and accuracy gains. With no labels available, on-device self-distillation results in similar accuracy to the original network with 14.6× fewer parameters (Fig. 5a) and 2.3× less FLOPs (Fig. 5d), exploiting the more narrow, personalised input distribution.

4.4.2 On-device performance. In this section, we evaluate the training and inference performance on mobile CPUs to assess the feasibility of personalising CNNs on device.

Training: On-device personalisation does not require full re-training of the model at hand. With the network's backbone frozen, we only personalise the early classifiers leading to significant gains in performance. As witnessed in Fig. 6a, multi-exit personalisation can be between 2× and 22.7× faster than full model training, with the main cost



(a) Per-exit training time. (b) Per-exit inference time.

Figure 6: Execution times for training and inference of personalised multi-exit ResNet-50 on mobile CPU.

allocated to the forward pass. This speedup is mainly attributed to not re-training the whole model. This way, we avoid the computational and memory overheads of tracking activations and gradients for the full model.

PersEPhonEE aims to train networks overnight, when the device is charged and idle [2]. Our experiments show that such a training on a mobile platform (Jetson's Arm Cortex CPUs) is possible. For instance, to train Exit-2 overnight with 2048 samples only requires 1.7 hours, whereas personalising all the exits simultaneously takes 6.8 hours. Finally, notice that personalisation will not be required frequently, but only when a data distribution shift is detected by the orchestrator.

Inference: In Fig. 6b, we observe that the inference time grows linearly with respect to the early-exit FLOPs. As a result, on-device personalisation can deliver significant speedups, leading to up to 3× higher inference throughput when hard labels are used in our ImageNet-based experiments.

PersEPhonEE allows multiple exits to be attached upon deployment, with the exit of a sample to be chosen at run time by comparing with a confidence threshold. This way, easier examples do not need to pass through the full depth of the CNN, while out-of-distribution inputs (*e.g.* a new environment) propagate until the last classifier and, therefore, maintaining the original model accuracy. Fig. 7 shows the accuracy-latency trade-off for confidence thresholds $thr_{\text{conf}} \in [0, 1]$ (increments of 0.05). We see that PersEPhonEE's orchestrator can explore this space depending on the device capabilities and performance of the personalised model. For example, when the ground truth is available for personalisation, the orchestrator can choose configurations with similar performance to the original network while achieving 3.2× higher inference throughput (2.2× with soft labels).

5 DISCUSSION AND FUTURE WORK

With the wide availability of on-device data and the ever-increasing concern about privacy, on-device training constitutes a strong competitor to centralised solutions. In addition, mobile devices are now equipped with unprecedented compute capabilities. In this context, ML and systems researchers and developers of today have to rethink the status quo assumptions, *e.g.* IID-ness of user data, and take advantage of the untapped compute available on users' devices.

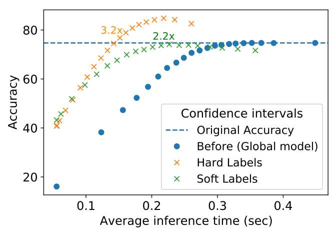


Figure 7: Accuracy vs. inference latency for various confidence thresholds.

PersEPhonEE offers a new perspective to the problem of lifelong learning through a system that exploits on-device data and multi-exit neural architectures to tailor models to a specific user, improving both accuracy and efficiency. Two of the key components in this endeavour are the ability to adapt CNNs to a specific environment *with or without* labelled data and the automated monitoring of both the training and inference parameters by the system orchestrator, continuously exploring opportunities to further improve performance.

At the same time, recent approaches that embrace the heterogeneity of data and the abundance of devices in the wild are becoming increasingly relevant. Meta-learning [9] attempts to find good initialisation for personalised models and to leverage expertise on different domains. Federated learning (FL) [17] aims to train globally accurate models without directly accessing the users' data. We believe that our work offers a missing piece in the equation, boosting the efficiency in the deployment of personalised models.

As future work, we plan to integrate PersEPhonEE with the aforementioned approaches to solve emerging challenges, such as avoiding catastrophic forgetting of less frequent knowledge or finding good initialisation for the early classifiers via meta-learning. Incorporation with FL will allow PersEPhonEE to share the personalised knowledge to constantly improve the global model while respecting user privacy. Moreover, tighter system integration with mobile accelerators will allow PersEPhonEE to further optimise the energy and latency footprint of AI training and inference.

6 CONCLUSIONS

This paper presents a framework for efficiently personalising CNNs using solely on-device resources. The proposed system introduces multi-exit networks that allow the customisation of the CNN based on the device capabilities and significantly reduce the computational overhead of training. Through an efficient on-device training algorithm that leverages the last exit's output to distill personalised knowledge to the earlier exits, the proposed system counteracts the common shortage of ground-truth labels on the user device. Evaluation shows that PersEPhonEE boosts the accuracy of early exits on user-specific samples while delivering significant speedup for both inference and training, making a decisive step towards on-device personalisation of CNNs on mobile platforms.

REFERENCES

[1] Mario Almeida et al. 2019. EmBench: Quantifying Performance Variations of Deep Neural Networks Across Modern Commodity Devices. In *EMDL*.

[2] Keith Bonawitz et al. 2019. Towards Federated Learning at Scale: System Design. In *Proceedings of Machine Learning and Systems (MLSys)*.

[3] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. 2017. EMNIST: Extending MNIST to handwritten letters. In *IJCNN*.

[4] Marc Delcroix, Keisuke Kinoshita, Atsunori Ogawa, Takuya Yoshioka, Dung T Tran, and Tomohiro Nakatani. 2016. Context Adaptive Neural Network for Rapid Adaptation of Deep CNN Based Acoustic Models. In *INTERSPEECH*. 1573–1577.

[5] Jia Deng et al. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*.

[6] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On Calibration of Modern Neural Networks. In *ICML*.

[7] K He, X Zhang, S Ren, and J Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*.

[8] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2014. Distilling the Knowledge in a Neural Network. In *NeurIPS Deep Learning Workshop*.

[9] Timothy Hospedales et al. 2020. Meta-Learning in Neural Networks: A Survey. In *arXiv*.

[10] Gao Huang et al. 2018. Multi-Scale Dense Networks for Resource Efficient Image Classification. In *ICLR*.

[11] Andrey Ignatov et al. 2019. AI Benchmark: All About Deep Learning on Smartphones in 2019. In *ICCVW*.

[12] Junguang Jiang, Ximei Wang, Mingsheng Long, and Jianmin Wang. 2020. Resource Efficient Domain Adaptation. In *MM*.

[13] Yigitcan Kaya et al. 2019. Shallow-Deep Networks: Understanding and Mitigating Network Overthinking. In *ICML*.

[14] Stefanos Laskaridis et al. 2020. SPINN: Synergistic Progressive Inference of Neural Networks over Device and Cloud. In *MobiCom*.

[15] Stefanos Laskaridis, Stylianos I. Venieris, Hyeji Kim, and Nicholas D. Lane. 2020. HAPI: Hardware-Aware Progressive Inference. In *ICCAD*.

[16] Taegyeong Lee, Zhiqi Lin, Saumay Pushp, Caihua Li, Yunxin Liu, Youngki Lee, Fengyuan Xu, Chenren Xu, Lintao Zhang, and Junehwa Song. 2019. Occlumency: Privacy-Preserving Remote Deep-Learning Inference Using SGX. In *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom)*.

[17] Brendan McMahan et al. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*.

[18] Fan Mo, Ali Shahin Shamsabadi, Kleomenis Katevas, Soteris Demetriou, Ilias Leontiadis, Andrea Cavallaro, and Hamed Haddadi. 2020. DarkneTZ: Towards Model Privacy at the Edge Using Trusted Execution Environments. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services (MobiSys)*. 161–174.

[19] R. T. Mullapudi, S. Chen, K. Zhang, D. Ramanan, and K. Fatahalian. 2019. Online Model Distillation for Efficient Video Inference. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*.

[20] Aditya Rajagopal and Christos-Savvas Bouganis. 2020. Now That I Can See, I Can Improve: Enabling Data-Driven Finetuning of CNNs on the Edge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

[21] Mark D. Ryan. 2011. Cloud Computing Privacy Concerns on Our Doorstep. *Commun. ACM* 54, 1 (2011), 36–38.

[22] Mark Sandler et al. 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *CVPR*.

[23] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. 2016. Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal (JIOT)* 3, 5 (2016), 637–646.

[24] Asoke K Talukder, Lawrence Zimmerman, et al. 2010. Cloud economics: Principles, costs, and benefits. In *Cloud computing*. Springer, 343–360.

[25] Surat Teerapittayanon et al. 2016. BranchyNet: Fast Inference via Early Exiting from Deep Neural Networks. In *ICPR*.

[26] Ravi Teja Mullapudi, William R Mark, Noam Shazeer, and Kayvon Fatahalian. 2018. HydraNets: Specialized Dynamic Architectures for Efficient Inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 8080–8089.

[27] Kaiyu Yang, Klint Qinami, Li Fei-Fei, Jia Deng, and Olga Russakovsky. 2020. Towards Fairer Datasets: Filtering and Balancing the Distribution of the People Subtree in the ImageNet Hierarchy. In *FAT**.

[28] Linfeng Zhang et al. 2019. SCAN: A Scalable Neural Networks Framework Towards Compact and Efficient Models. In *NeurIPS*.