Binary Early-Exit Network for Adaptive Inference on Low-Resource Devices

Aaqib Saeed
Dept. of Computer Science and Technology
University of Cambridge
as3227@cl.cam.ac.uk

Abstract

Deep neural networks have significantly improved performance on a range of tasks with the increasing demand for computational resources, leaving deployment on low-resource devices (with limited memory and battery power) infeasible. Binary neural networks (BNNs) tackle the issue to an extent with extreme compression and speed-up gains compared to real-valued models. We propose a simple but effective method to accelerate inference through unifying BNNs with an early-exiting strategy. Our approach allows simple instances to exit early based on a decision threshold and utilizes output layers added to different intermediate layers to avoid executing the entire binary model. We extensively evaluate our method on three audio classification tasks and across four BNNs architectures. Our method demonstrates favorable quality-efficiency trade-offs while being controllable with an entropy-based threshold specified by the system user. It also results in better speed-ups (latency less than 6ms) with a single model based on existing BNN architectures without retraining for different efficiency levels. It also provides a straightforward way to estimate sample difficulty and better understanding of uncertainty around certain classes within the dataset.

Index Terms: early-exit, audio recognition, binary neural networks

1. Introduction

The rapidly increasing size of deep neural networks, along with advancements in design and training, have made them perform well on a broad spectrum of tasks. The large-scale models produce predictive performance but significantly escalate production costs with slower inference, in particular, severely limiting the adoption of deep models on resource-constrained edge devices like wearables with limited battery and computational power. Binary neural networks (BNNs) have become promising methods for obtaining highly compact and efficient models in deployment on resource-constrained devices due to extreme compression and speed-up gains compared to their real-valued counterparts [1, 2]. A complementary technique for accelerating deep models is a dynamic input-dependent prediction generation, which has recently gained notable traction and has become widely known as early-exit architecture [3, 4, 5, 6, 7, 8, 9, 10].

The key idea behind early-exit models is that the difficulty of classifying an example, and thus the required model capacity to do so, varies greatly in practice. For instance, a speech classifier is likely to be provided with utterances over a wide range of signal-to-noise ratios. The shallow features from initial layers are mostly sufficient to confidently distinguish easy examples of different classes to a certain extent [9, 7, 6]. These models are capable of trading-off computation and accuracy dynamically on a per-instance basis to efficiently utilize available resources. An early-exit architecture adapts its computation to the difficulty of an example by placing auxiliary decision layers (the “exits”) at different depths of its main architecture. At inference, an example passes through each exit in a sequential fashion, and a decision rule (or exit threshold) is typically used to decide whether to use the current exit’s prediction or to continue.

We design an early-exit model to reduce inference cost and further speed-up execution of BNNs based on the aforementioned observation. We focus solely on BNNs due to their greater efficiency and minimal model size that is largely suitable for devices with limited memory [11]. We address the problem of early-exiting from the perspective of varying model sizes and sample (or instance) difficulty. Our approach provides a simple yet effective way to assess the computational difficulty of making a prediction for a given input and pair it with an optimal exit at a specific depth within the model to reduce uncertainty. As a result, not all instances have to pass through the entire model. The ‘easy’ to classify examples are matched with earlier exits. In comparison, the ‘harder’ examples are tackled by deeper exits to maintain good performance and thus save computation on a per-example basis.

We apply our method to a broad range of high-performing and efficient binary networks architectures, which we repurpose for different audio recognition tasks. To the best of our knowledge, our work is the first attempt at proposing and studying binary early-exit models for audio recognition. Specifically, we add classification output layers (or also called exits) at varying depths, while maintaining the rest of the architecture as is. We notice that an exit at the jth layer is efficient though (seemingly) less effective as compared to later exit k, (where k > j). Surprisingly, in some cases we achieve similar or even better performance than standard single-exit models, hinting at ‘over-thinking’ phenomena in neural networks in which the predictions based on earlier layers are correct but become incorrect with progressively deeper layers, resulting in wasteful computation [12]. We provide high-level illustration of the approach in Figure 1.

Our approach of unifying binary networks with early exits accelerates inference substantially and results in saving in terms of computation compared to single-exit models while adding very few learnable parameters to the model. Likewise, it does not result in a penalty in terms of training convergence. Our simple strategy is flexible in practice compared to other techniques, for instance, methods that require re-training to achieve an optimal trade-off between speed-up and accuracy. On the contrary, we can train the early-exit BNNs end-to-end and require only the selection of the appropriate value of decision threshold to decide on exiting or further processing of the input. Likewise, the proposed technique is orthogonal to pruning and other strategies for improving neural network efficiency. It can be easily combined to further enhance the computational
Figure 1: Overview of our early-exit binary neural networks. Grey blocks denote a set of binarized convolutional and other layers, orange and green squares denote decision exits and classification output layers, respectively. Our early-exit model enables adaptive inference for each input sample based on an entropy-based decision threshold to early-exit the input, thus saving computation.

2. Method

We design an early-exit classification architecture with \( E \) exits that produces a list of outputs \( \{p_{e_1}, p_{e_2}, \ldots, p_{e_E}\} \) where each exit \( p_{e_i} \) is a probability distribution over classes. Specifically, for the exits \( 1 < i < E \), an exit \( e_i \) is potentially more accurate and expensive to compute than the previous exit i.e., \( e_{i-1} \). Nevertheless, a slightly less accurate exit can also be used for inference based on a pre-defined threshold of the output to trade-off between accuracy and computation. At inference time, the early-exit model can quantify the exit’s confidence on a per-example basis with an entropy of output probability distribution or temperature-scaled softmax score \( \mu \). When an instance arrives at an exit, the chosen metric is computed and compared with a pre-defined threshold \( S \) to decide to exit earlier or pass on the sample to the next layer. Intuitively, the larger values of the threshold \( S \) result in a faster inference but less accurate prediction and vice versa as the earlier exits can be inaccurate to solve the specified learning task. This strategy is also termed as “early-exiting,” i.e., for a sample, an output can be produced at any time depending on the computational budget or accuracy.

We consider binary convolutional networks [1] as backbone models to augment with early-exit capability for inference on low-resource devices with limited computational power. We represent the weights and input features with \( Y \in \mathbb{R}^{k \times k \times c \times o} \) and \( X \in \mathbb{R}^{w \times h \times c} \), with \( c \) and \( o \) being number of input and output channels, \( k \) denoting kernel height and width, while \( w \) and \( h \) represent spatial dimension of the input. Generally, both the weights and activations of the model are binarized, where convolution operation is performed as \( X \ast W \). For the exits \( i \), the chosen metric is computed and compared with a pre-defined threshold \( \delta \). When an instance arrives at an exit, the chosen metric is computed and compared with a pre-defined threshold \( \delta \) to decide to exit earlier or pass on the sample to the next layer. Intuitively, the larger values of the threshold \( S \) result in a faster inference but less accurate prediction and vice versa as the earlier exits can be inaccurate to solve the specified learning task. This strategy is also termed as “early-exiting,” i.e., for a sample, an output can be produced at any time depending on the computational budget or accuracy.

3. Experiments

3.1. Datasets, Tasks and Implementation Details

We conduct experiments on various audio recognition tasks ranging from spoken language identification, keyword spotting, to spoken commands recognition. We choose Speech Commands [19] and Voxforge [20], and MSWC (Micro-EN) [21] datasets sampled at 16kHz with each comprising 12, 6, and 31 classes, respectively. In all cases, we utilize the standard train-test splits provided with the corresponding datasets. Given an audio input sequence, we randomly select a one-second of audio segment from an entire audio clip to extract log-compressed Mel-filterbanks with a window size of 25 ms, a hop size of 10 ms, and \( N = 64 \) Mel-spaced frequency bins in the range 60-7800 Hz for \( T = 98 \) frames, corresponding to 980 ms. We use these features as input to binary neural networks, i.e., QuickNet, BiRealNet, BinaryDenseNet-28, and MiusNet. Even though these architectures are originally proposed for vision tasks, the 2D input structure of Mel-filterbanks enables us to use
them without further adjustment. We use the Larq [11] framework for implementation of BNNs and utilize Binary Optimizer (Bop) [22] and Adam [23] for training models with a learning rate of 0.001 and batch size of 128 for 100 epochs. During the evaluation, we use each audio sample without any splitting, effectively using a batch size of one to handle input of varying sizes.

3.2. Results

We start with analyzing the generalization of each exit for considered neural architectures and datasets in Table 2. The exits shown with $E_s$ are single exit models where a single output layer after some intermediate layer is added while $E_m$ represents the same exit but trained jointly in an end-to-end fashion. We notice that the performance of individual exits within early-exit models is mostly consistent with the ones trained independently. Further, the single exit model ($E_s$) can also be seen as efficient baselines, which offer a trade-off between accuracy and speed, but unlike with one model like our approach.

In Figure 2, we present our key results achieved with different entropy thresholds for each task and architecture. The curves report accuracy achieved while setting a fixed value of entropy $\delta$ with an early-exit model. We use entropy thresholds of $0.1, 0.25, 0.5, 0.75,$ and $1.0$. We observe that even with high values of $\delta$, the performance stays considerably stable. In the case of SpeechCommands, for all early-exit BNNs, the accuracy is around 91% when $\delta = 0.5$. Interestingly, on Voxforge, we see a significant difference in performance across architectures. Specifically, the BiRealNet accuracy is lower than the rest and degrades more rapidly with increasing uncertainty. However, MeliusNet performs best with $\delta = 0.5$, achieving around 90% accuracy. On the other hand, we notice that architectures achieve optimal performance with $0.6 < \delta < 0.8$. Both the early and late exits results in lower performance, these results echo the finding of [12], that “overthinking” can lead to incorrect predictions.
We show the fraction of instances exiting from each exit of MeliusNet for a fixed entropy threshold in Figure 3. The lower values of $\delta$ force the samples to increasingly utilize later exits. With increasing $\delta$, more examples exit early, thus saving computation. We notice example difficulty varies across tasks. For instance, in the case of SpeechCommands, the first exits generally dominate the inference to classify the majority of examples. However, on MSWC and Voxforge, all exits are mostly utilized. In particular, we observe an interesting pattern for Voxforge that instances mostly either exit early or prefer the last exit with few using the fourth exit. These results highlight that entropy-based exiting optimally selects suitable exit while maintaining a good trade-off between predictive quality and efficiency. In Figure 4, we show the accuracy of binary models with respect to average exit for various entropy thresholds. The dotted gray line represents the performance of the standard single exit model, where all instances exit from the last layer. We observe consistent results with later exits improving performance at the cost of utilizing more compute, with BinaryDenseNet and MeliusNet achieving higher accuracy as compared to the rest. Furthermore, Figure 5 presents the proportion of instances in the Voxforge test set for each semantic class and models for entropy threshold of 0.5. We notice that across models, not all classes utilize all exits. In particular, EN is recognized with early exits, while DE and RU largely used later exits. We also observe that in QuickNet and BiRealNet, difficult classes pushed more towards the last layers, on average using the fourth exit.

4. Conclusions

We unify binary neural networks with the early-exiting approach for audio recognition on devices with low computational resources. To this end, we present a method to enhance their quality-efficiency trade-offs for efficient inference. Our conceptually simple and architecture-agnostic approach augments standard models with early exits at different intermediate layers to make predictions on easy instances by executing fewer layers. Our experimental evaluation on three different tasks and four neural architectures demonstrates the effectiveness of our approach over standard single exit models while maintaining similar performance at a lower computational budget. Further, our technique provides a simple strategy based on entropy for controlling the speed and accuracy of inference with a single model.
5. References


