Empowering Adaptive Early-Exit Inference with Latency Awareness

Xinrui Tan,[1] **Hongjia Li,**[1*] **Liming Wang,**[1] **Xueqing Huang,**[2] **Zhen Xu**[1]
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]New York Institute of Technology, New York, USA
{tanxinrui, lihongjia, wangliming, xuzhen}@iie.ac.cn, xhuang25@nyit.edu

Abstract

With the capability of trading accuracy for latency on-the-fly, the technique of adaptive early-exit inference has emerged as a promising line of research to accelerate the deep learning inference. However, studies in this line of research commonly use a group of thresholds to control the accuracy-latency trade-off, where a thorough and general methodology on how to determine these thresholds has not been conducted yet, especially with regard to the common requirements of average inference latency. To address this issue and enable latency-aware adaptive early-exit inference, in the present paper, we approximately formulate the threshold determination problem of finding the accuracy-maximum threshold setting that meets a given average latency requirement, and then propose a threshold determination method to tackle our formulated non-convex problem. Theoretically, we prove that, for certain parameter settings, our method finds an approximate stationary point of the formulated problem. Empirically, on top of various models across multiple datasets (CIFAR-10, CIFAR-100, ImageNet and two time-series datasets), we show that our method can well handle the average latency requirements, and consistently finds good threshold settings in negligible time.

Introduction

Over the past decade, deep neural networks have reached or even surpassed human accuracy in many machine learning applications. However, this performance superiority is strongly connected to the over-parameterized deep learning models. With a highly complex model, it is generally too computationally intensive to perform the subsequent inference task on smartphones or other resource-constrained Internet-of-Things (IoT) devices. To address the rising challenges of enabling efficient inference with deep neural networks, various techniques have emerged to guarantee the Quality-of-Service (QoS) of latency-sensitive applications, such as quantization (Courbariaux, Bengio, and David 2015), pruning (Han et al. 2015), knowledge distillation (Hinton, Vinyals, and Dean 2015), and lightweight network architectures (Howard et al. 2017; Cai, Zhu, and Han 2018a).

*Hongjia Li is the corresponding author.

Another relatively new, yet promising approach to overcome the resource limitation is the adaptive early-exit inference technique (Panda, Sengupta, and Roy 2016; Teerapittayanon, McDanel, and Kung 2016; Stamoulis et al. 2018; Wu et al. 2020; Huang, Lai, and Chen 2017; Goetschalckx et al. 2018; Cheng et al. 2019; Yokoo, Iizuka, and Fukui 2019; Huang et al. 2017; Zhang et al. 2019; Aketi, Panda, and Roy 2020; Leroux et al. 2015), which is capable of adjusting the computational effort based on each input sample's *inference difficulty* (Venkataramani et al. 2015). This technique builds upon the idea of distributing computing resources unevenly across *easy* and *hard* input samples. In particular, for easy input samples whose patterns can be recognized using the shallow portions of the deep learning model, an early exit is allowed to avoid the subsequent unnecessary computational expense. While for hard input samples that constitute only a small fraction of real-world datasets (Panda, Sengupta, and Roy 2016), the inference task will execute the computation of the full deep learning model. As a result, the adaptive early-exit inference technique can considerably reduce the inference latency for the easy input samples without sacrificing the overall inference accuracy.

To facilitate the implementation of the adaptive early-exit inference technique, existing studies (Panda, Sengupta, and Roy 2016; Teerapittayanon, McDanel, and Kung 2016; Stamoulis et al. 2018; Teerapittayanon, McDanel, and Kung 2017; Wu et al. 2020; Huang, Lai, and Chen 2017; Goetschalckx et al. 2018; Cheng et al. 2019; Yokoo, Iizuka, and Fukui 2019; Huang et al. 2017; Zhang et al. 2019; Aketi, Panda, and Roy 2020; Leroux et al. 2015; Yuan et al. 2019) have concentrated heavily on designing architectures of *early-exit models* with early exit points; most of these studies have employed a threshold-based mechanism to control the adaptive inference processes. Broadly speaking, this mechanism requires a predetermined threshold for each early-exit point, and these thresholds are in turn used in an inference process to determine after which exit point the inference process should stop. Despite the simplicity of the threshold-based mechanism, different threshold settings explicitly lead to different trade-offs between overall inference accuracy and latency. Since the average inference latency is a common requirement for the long-term running models, an on-demand method for appropriately determining the thresholds is needed. However, throughout the lit-

erature, the thresholds are regarded as hardly-learnable hyperparameters, and the ways they impact inference accuracy and latency have been typically treated as black boxes — in a word, there is a lack of mathematical models for determining the thresholds, so that studies have been dedicated to heuristics (Lo et al. 2017; Park et al. 2015; Berestizshevsky and Even 2019; Sun and Pang 2018; Huang et al. 2017) or metaheuristics (Zhang et al. 2019; Jayakodi et al. 2018) for threshold determination.

In general, the methods proposed by these studies suffer from three major limitations. First, almost all of them cannot provide any performance guarantee and thus not applicable to applications with latency or accuracy requirements. Second, they may only work for the specific early exit models that they designed for, and so far, no general-purpose method has been developed. Third, they cannot work well for the recurrent neural networks, which can have a large number of exit points.

To address these limitations, in this paper, we investigate the latency-aware threshold optimization problem, where the goal is to maximize the overall inference accuracy while meeting the average latency requirement; we show that, given an average latency requirement and a well-trained early-exit model, it is feasible for the thresholds to be "trained", *i.e.*, optimized in a data-driven manner and via first-order methods. More specifically, we consider our target problem in a general form to cover a wide range of early-exit models, and approximately formulate it as a constrained non-convex program. Despite the intractability inherent in the non-convex constraint, we develop a threshold determination method based on the inexact proximal-point penalty (iPPP) method of Lin *et al.* (Lin, Ma, and Xu 2019). We further prove that with appropriate parameter settings, our method can output a near-stationary point of the formulated program within a polynomial time.

We empirically evaluate our threshold determination method not only using three benchmark early-exit models on the two CIFAR (Krizhevsky 2009) and ILSVRC-2012 ImageNet (Deng et al. 2009) datasets but also using two recurrent network models on two time-series datasets. For each model, an extensive evaluation over a wide range of average latency requirements is conducted, and it is demonstrated that: (i) our method can effectively handle the average latency requirements; (ii) our method generally outperforms the baseline heuristic of Huang *et al.* (Huang et al. 2017) while performing the same average latency guarantee; (iii) our method induces a computational overhead that is negligible compared with the model training time.

Related Work

Adaptive Early-Exit Inference. Prior studies investigating the adaptive early-exit inference technique have mainly focused on the architecture design (Panda, Sengupta, and Roy 2016; Teerapittayanon, McDanel, and Kung 2016; Stamoulis et al. 2018; Teerapittayanon, McDanel, and Kung 2017; Wu et al. 2020; Huang, Lai, and Chen 2017; Goetschalckx et al. 2018; Cheng et al. 2019; Yokoo, Iizuka, and Fukui 2019; Huang et al. 2017; Zhang et al. 2019; Aketi, Panda, and Roy 2020; Leroux et al. 2015; Yuan et al. 2019) and training (Phuong and Lampert 2019; Li et al. 2019b; Hu et al. 2020) of the early-exit models. In particular, for studies of architecture design, most pioneers (Leroux et al. 2015; Panda, Sengupta, and Roy 2016; Teerapittayanon, McDanel, and Kung 2016) design their architectures by moderately modifying the backbone architectures that have a single exit point. For instance, BranchyNet (Teerapittayanon, McDanel, and Kung 2016) introduces additional branch classifiers at certain intermediate layers of a backbone architecture, thereby obtaining an architecture with multiple exit points. More recently, some more advanced designs have been proposed (Huang et al. 2017; Zhang et al. 2019; Yokoo, Iizuka, and Fukui 2019), where the computation reuse is maximized and during training, the interference between exit-points is reduced. For examples, Huang *et al.* (Huang et al. 2017) designed a two-dimensional multi-scale network architecture that improves the accuracy of early-exit classifiers by maintaining coarse level features throughout the networks; Zhang *et al.* (Zhang et al. 2019) proposed a scalable neural network framework that utilizes attention modules and knowledge distillation to learn good early-exit classifiers. Also, recent studies have leveraged neural architecture search (NAS) to explore the best-performing early-exit architecture (Zhang, Ren, and Urtasun 2018; Ghiasi, Lin, and Le 2019; Yuan et al. 2019). We remark that the intention of this paper is not to present yet another architecture, but to provide a general method that can help the early-exit models to determine the thresholds in a latency-aware fashion.

Threshold Optimization. In adaptive early-exit inference studies employing the threshold-based mechanism, the thresholds play a key role in trading off the inference accuracy and latency. However, most existing heuristics and metaheuristics perform the threshold optimization by simply finding a threshold setting that may yield a good accuracy-latency trade-off (Lo et al. 2017; Park et al. 2015; Berestizshevsky and Even 2019; Sun and Pang 2018; Zhang et al. 2019; Jayakodi et al. 2018; Baccarelli et al. 2020). On one hand, these heuristics either are restricted to simple models with only one early-exit point (Park et al. 2015; Lo et al. 2017), or ignore the impact of shallow exit points on deep exit points (Berestizshevsky and Even 2019; Sun and Pang 2018). On the other hand, Zhang *et al.* (Zhang et al. 2019) and Jayakodi *et al.* (Jayakodi et al. 2018) optimized the thresholds, respectively, based on genetic and Bayesian optimization algorithms, but their derivative-free optimization-based methods, like the aforementioned heuristics, are still unable to handle inference latency requirements. To the best of our knowledge, the only exception that deals with inference latency requirements is the heuristic introduced by Huang *et al.* (Huang et al. 2017) for their multi-scale dense networks (MSDNet). This heuristic, which has been used in other studies (Wu et al. 2020), assumes that the exit probabilities are the same for all exit points. In contrast, our method does not need such an unreasonable assumption. Finally, for recurrent neural networks, the threshold-based mechanism is known as a very intuitive approach to perform early-exit inference (Hartvigsen et al. 2019). However, at present, there is neither a threshold determination method with latency awareness, nor any other early-exit inference

mechanism that can explicitly deal with latency requirements. In the extreme case, a recurrent neural network can have an exit point at each time-step, which results in a large number of exit points, making the threshold determination more difficult. For simplicity, a straightforward method in the literature (Dennis et al. 2018) is to let all exit points share a single threshold; we call this method the single-threshold method for later reference.

Problem Formulation

Throughout the whole paper, we consider the case of multi-class classification with data space $\mathcal{X}$ and finite label space $\mathcal{Y}$, where the cardinality of $\mathcal{Y}$ is C. Furthermore, we consider that the inference is performed with a batch size of 1, which is known to be customary for real-time inference (Han et al. 2016). Given an early-exit model with N exit points, it is conventional to index the exit points with successive integers starting at 1, such that a smaller index corresponds to an exit point located closer to the model's entry point; it is in this sense that the N-th exit point is referred to as the *final exit point*. Once the early-exit model is trained, for each $n \in \{1, 2, \ldots, N\}$, we let $f_n \colon \mathcal{X} \to \mathcal{P}$ denote the output function of the n-th exit point, which computes categorical distribution over $\mathcal{Y}$. Note that here, $\mathcal{P}$ is the probability simplex over $\mathcal{Y}$, and for any $p \in \mathcal{P}$ and $y \in \mathcal{Y}$, we further let p_y represent the probability of y in the distribution p. To perform adaptive inference on the early-exit model, after receiving a sample $\mathbf{x} \in \mathcal{X}$, the inference process starts evaluating $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_{N-1}(\mathbf{x})$ in turn, reusing computation where possible. This continues until the *confidence* of an exit point's output is above the threshold corresponding to the exit point, where the label with the maximum probability in the output is returned. If the inference process is not terminated even after $f_{N-1}(\mathbf{x})$ is evaluated, the full computation of the model will be accomplished, and the label with the maximum probability in $f_N(\mathbf{x})$ will be returned. Formally, the mapping from the outputs of the exit points to the corresponding confidences is denoted by the function $H \colon \mathcal{P} \to S$, where $S \subset \mathbb{R}$ is a compact set bounded by the confidence limits. For concreteness, when using the (negative) entropy as the confidence measure, H can be explicitly defined as

$$H(p) \coloneqq \textstyle\sum_{y \in \mathcal{Y}} p_y \log p_y, \quad \forall p \in \mathcal{P}, \tag{1}$$

and we have $S = [-\log C, 0]$. Alternatively, other confidence measures can also be used, including the more widely employed raw *softmax response* (Cordella et al. 1995). Moreover, we introduce $\mathbf{t} \in S^{N-1}$, whose n-th element ($n \in \{1, 2, \ldots, N-1\}$) represents the threshold specified for the n-th exit point. To help understand how the adaptive early-exit inference is performed, an illustrative example is included in Appendix A.

Now suppose that there is an underlying data distribution $\mathcal{D}$ over pairs of samples $\mathbf{x} \in \mathcal{X}$ and corresponding labels $y \in \mathcal{Y}$. Referring to the aforementioned adaptive inference processes, we can derive the following necessary and sufficient conditions for an arbitrary input sample $\mathbf{x}$ to exit at the n-th exit point, $n \in \{1, 2, \ldots, N-1\}$:

- all the confidences of previous $n-1$ exit points' outputs are respectively no more than their associated thresholds, *i.e.*, $\forall i \in \{1, 2, \ldots, n-1\}$, $H(f_i(\mathbf{x})) \leq \mathbf{t}_i$;
- the confidence of $f_n(\mathbf{x})$ is more than the threshold $\mathbf{t}_n$, *i.e.*, $H(f_n(\mathbf{x})) > \mathbf{t}_n$.

Accordingly, for each $n \in \{1, 2, \ldots, N-1\}$, we have the following function g_n that approximately indicates whether the inference result of a given sample $\mathbf{x} \in \mathcal{X}$ is provided by the n-th exit point or not:

$$g_n(\mathbf{x}, \mathbf{t}) = \sigma(H(f_n(\mathbf{x})) - \mathbf{t}_n) \prod_{i=1}^{n-1} \sigma(\mathbf{t}_i - H(f_i(\mathbf{x}))), \tag{2}$$

where $\sigma \colon \mathbb{R} \to (0, 1)$ is the Verhulst logistic function used to approximate the Heaviside step function to avoid the zero-derivative problem. Formally, σ is defined by

$$\sigma(x) \coloneqq \frac{1}{1 + \exp(-kx)}, \quad \forall x \in \mathbb{R}, \tag{3}$$

where k controls the steepness of the curve, and it is known that σ tends to the Heaviside step function as $k \to \infty$. Thus, it follows that, for a given $\mathbf{t}$ and any $n \in \{1, 2, \ldots, N-1\}$, if $\mathbf{x}$ exits at the n-th exit point, $g_n(\mathbf{x}, \mathbf{t}) \to 1$ as $k \to \infty$; otherwise, $g_n(\mathbf{x}, \mathbf{t}) \to 0$ as $k \to \infty$. Likewise, for the final exit point, we have the following function

$$g_N(\mathbf{x}, \mathbf{t}) = \textstyle\prod_{n=1}^{N-1} \sigma(\mathbf{t}_n - H(f_n(\mathbf{x}))). \tag{4}$$

where it can be seen that $g_N(\mathbf{x}, \mathbf{t}) \to 1$ as $k \to \infty$ iff $\mathbf{x}$ exits at the final exit point, *i.e.*, $H(f_n(\mathbf{x})) \leq \mathbf{t}_n$ for all $n \in \{1, 2, \ldots, N-1\}$.

Due to the fact that the correctness of a sample $\mathbf{x}$'s inference only depends on the output of the exit point at which $\mathbf{x}$ exits, if the corresponding label y is known, we can use the following function to measure the inference correctness:

$$a(\mathbf{x}, y, \mathbf{t}) = \textstyle\sum_{n=1}^{N} g_n(\mathbf{x}, \mathbf{t}) \left[1 - l(f_n(\mathbf{x}), y)\right], \tag{5}$$

where $l \colon \mathcal{P} \times \mathcal{Y} \to \{0, 1\}$ is the 0-1 loss function, *i.e.*, for any categorical distribution p over $\mathcal{Y}$ and any label $y \in \mathcal{Y}$, if y has the maximum probability in p, $l(p, y) = 0$; otherwise, $l(p, y) = 1$. With a sufficiently large k, it is obvious that $a(\mathbf{x}, y, \mathbf{t})$ tends to 1 for a correct inference result of $\mathbf{x}$; while for an incorrect result, $a(\mathbf{x}, y, \mathbf{t})$ is close to 0. Similarly, we can also approximately predict the inference latency, with respect to $\mathbf{x}$ and $\mathbf{t}$, by

$$b(\mathbf{x}, \mathbf{t}) = \textstyle\sum_{n=1}^{N} d_n g_n(\mathbf{x}, \mathbf{t}), \tag{6}$$

where for each $n \in \{1, 2, \ldots, N\}$, d_n denotes the measured or predicted mean latency of the adaptive inference terminated at the n-th exit point on the target hardware. Alternatively, $d_1, d_2, \ldots, d_N$ can also be approximately measured using hardware-agnostic metrics, such as floating-point operation (FLOP) counts. Note that there are various methods in the literature to perform latency prediction, *e.g.*, the analytical performance model, named PALEO (Qi, Sparks, and Talwalkar 2017), and the regression-based methods (Li et al. 2019a; Cai, Zhu, and Han 2018b).

Recall that our goal is to optimize the inference accuracy, and meanwhile ensure the requirement of average inference

latency is met. To this end, the threshold optimization problem can be approximately formulated as

$$\mathbf{P}_1: \quad \min_{\mathbf{t}\in S^{N-1}} \ \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[-a(\mathbf{x},y,\mathbf{t})\right], \tag{7}$$

$$\text{s.t.} \quad \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[b(\mathbf{x},\mathbf{t})\right] \le \Gamma, \tag{8}$$

where Γ is the predefined average latency requirement. In problem $\mathbf{P}_1$, the objective function (7) is to maximize the classification accuracy, and (8) constrains the expected inference latency to be at most Γ. Note that, instead of the average inference latency, the tail inference latency may be of practical concern; in Appendix C, we discuss how the problem can be reformulated to handle a tail latency requirement. In practice, the distribution $\mathcal{D}$ is unknown, and instead we need to rely on a finite number of empirical training samples $(\mathbf{x}_1,y_1),(\mathbf{x}_2,y_2),\ldots,(\mathbf{x}_M,y_M)$. Therefore, the stochastic $\mathbf{P}_1$ is further approximated by

$$\mathbf{P}_2: \quad \min_{\mathbf{t}\in S^{N-1}} \ o(\mathbf{t}) = -\tfrac{1}{M}\textstyle\sum_{m=1}^{M} a(\mathbf{x}_m,y_m,\mathbf{t}), \tag{9}$$

$$\text{s.t.} \quad c(\mathbf{t}) = \tfrac{1}{M}\textstyle\sum_{m=1}^{M} b(\mathbf{x}_m,\mathbf{t}) - \Gamma \le 0. \tag{10}$$

Notice that neither the objective function $o(\mathbf{t})$ nor the constraint $c(\mathbf{t})$ is convex, implying that finding the global optimum of $\mathbf{P}_2$ is intractable (Sahni 1974). In fact, the non-convex constraint in $\mathbf{P}_2$ makes it even more difficult, while recent work on constrained non-convex optimization is still primarily for the convex functional constrained case (Kong, Melo, and Monteiro 2019; Hong 2016; Grapiglia and Yuan 2019). Hence, we will content ourselves with efficiently finding a good approximate stationary point of $\mathbf{P}_2$.

Threshold Determination Method

To deal with $\mathbf{P}_2$, we introduce the following two constants:

$$A = \max\{1, \tfrac{\sqrt{N-1}}{4}\}, \tag{11}$$

$$B = \max\{\Gamma, d_N - \Gamma, \tfrac{d_N\sqrt{N-1}}{4}\}, \tag{12}$$

so that we have, $\forall \mathbf{t}\in S^{N-1}$,

$$A \ge \max\{|o(\mathbf{t})|, \|\nabla o(\mathbf{t})\|\}, \tag{13}$$

$$B \ge \max\{|c(\mathbf{t})|, \|\nabla c(\mathbf{t})\|\}. \tag{14}$$

Moreover, we shall use the following lemma, which is proved in Appendix C. Note that in Appendix B, we give the preliminary definitions necessary for describing our method.

Lemma 1. *Let* $\rho = d_N k^2(N-1)\frac{N(N+4\sqrt{3}+1)+8\sqrt{3}-6}{48}$. *Then,* o *is a* $\frac{\rho}{d_N}$*-weakly convex and* $\frac{\rho}{d_N}$*-smooth function. Also,* c *is a* ρ*-weakly convex and* ρ*-smooth function.*

We next apply the inexact proximal-point penalty (iPPP) method proposed by Lin *et al.* (Lin, Ma, and Xu 2019) to $\mathbf{P}_2$. At a high level, the iPPP is a first-order method consisting of inner-outer iterations, where in each outer iteration a strongly convex subproblem is quickly solved up to a required accuracy with the Nesterov's accelerated proximal gradient (APG) method (Nesterov 2018). More specifically, for our problem, the iPPP method iteratively performs the update:

$$\bar{\mathbf{t}}^{(t+1)} \approx \arg\min\nolimits_{\mathbf{t}\in S^{N-1}} \phi_t(\mathbf{t}), \tag{15}$$

Algorithm 1 iPPP method for threshold determination

Input: initial point $\bar{\mathbf{t}}^{(0)}$, outer iteration number T, proximal parameter γ, penalty parameter β, inner iteration number J, strongly convexity parameter μ and smoothness parameter η

for $t = 0,1,\ldots,T-1$ **do**
  Take $\bar{\mathbf{t}}^{(t)}$ to set ϕ_t by (16)
  $\mathbf{v}^{(0)} \leftarrow \bar{\mathbf{t}}^{(t)}, \mathbf{w}^{(0)} \leftarrow \bar{\mathbf{t}}^{(t)}$
  for $j = 0,1,\ldots,J-1$ **do**
    $\mathbf{v}^{(j+1)} \leftarrow P(\mathbf{w}^{(j)} - \frac{1}{\eta}\nabla\phi_t(\mathbf{w}^{(j)}))$
    $\mathbf{w}^{(j+1)} \leftarrow \mathbf{v}^{(j+1)} + \frac{\eta-\mu}{\eta+\mu}(\mathbf{v}^{(j+1)} - \mathbf{v}^{(j)})$
  end for
  $\bar{\mathbf{t}}^{(t+1)} \leftarrow \mathbf{v}^{(J)}$
end for
$R \leftarrow \arg\min_{0\le t<T} \left\|\bar{\mathbf{t}}^{(t+1)} - \bar{\mathbf{t}}^{(t)}\right\|$

Output: $\bar{\mathbf{t}}^{(R)}$

where, by respectively denoting with $\beta > 0$ and $\gamma > 0$ the penalty and proximal parameters, ϕ_t is given by

$$\phi_t(\mathbf{t}) = o(\mathbf{t}) + \tfrac{\gamma}{2}\left\|\mathbf{t} - \bar{\mathbf{t}}^{(t)}\right\|^2 + \tfrac{\beta}{2}(\max\{0, c(\mathbf{t})\})^2. \tag{16}$$

From the weakly convexity of o and c proved by Lemma 1, it follows that $\frac{\beta}{2}(\max\{0,c(\mathbf{t})\})^2$ is $\beta B\rho$-weakly convex and $\beta B(B+\rho)$-smooth. Therefore, each ϕ_t is η-smooth with

$$\eta = \frac{\rho}{d_N} + \gamma + \beta B(B+\rho). \tag{17}$$

Indeed, we can choose a large enough γ such that each ϕ_t becomes μ-strongly convex, *i.e.*, satisfying

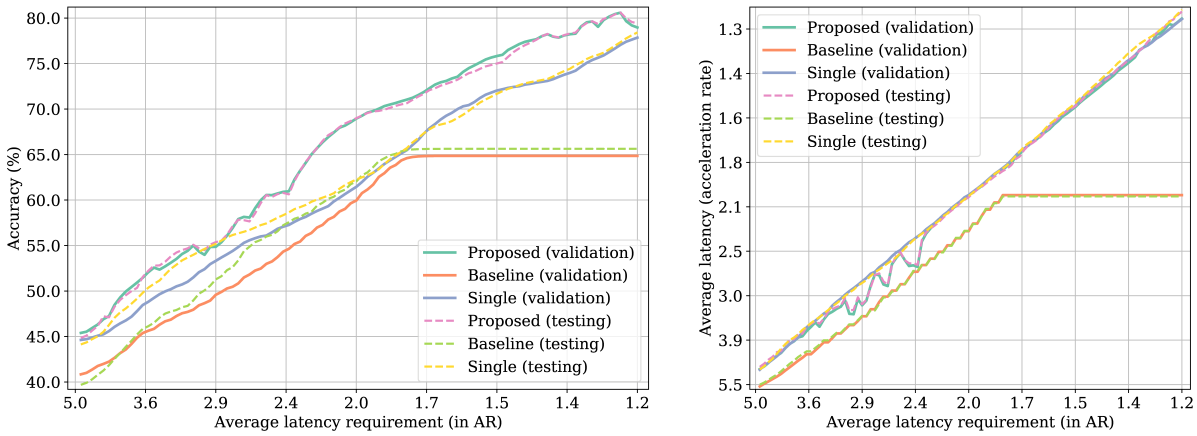
$$\mu = \gamma - \frac{\rho}{d_N} - \beta B\rho > 0. \tag{18}$$

Then, the Nesterov's APG method can converge to the global optimum of the minimization problem in (15) at a linear rate. We formally describe the iPPP method for $\mathbf{P}_2$ in Algorithm 1, where $P\colon \mathbb{R}^N \to S^{N-1}$ denotes the orthogonal projection onto S^{N-1}. Suppose that the target functional accuracy $\epsilon > 0$, we also present the following theorem to show that Algorithm 1 with proper parameters results in a weak ϵ-stationary point defined by Lin *et al.* (Lin, Ma, and Xu 2019), where the complementary slackness conditions are not required to hold approximately when inequality constraints are violated. For the convenience of readers, we formally define the weak ϵ-stationary points of $\mathbf{P}_2$ in Appendix B.

Theorem 1. *Given* $\epsilon > 0$*, in Algorithm 1, let* $\beta = \frac{32A}{\epsilon^2}$, $\gamma = \max\{1, \frac{2\rho}{d_N} + 2\beta B\rho\}$, $T = \lceil\frac{128\gamma A}{9\epsilon^2}\rceil$, μ *be set by* (18)*,* η *be set by* (17)*, and* J *be chosen such that*

$$(1 - \sqrt{\tfrac{\mu}{\eta}})^J[2A + \beta B^2 + \tfrac{\gamma+\mu}{2}(N-1)D^2] \le \delta, \tag{19}$$

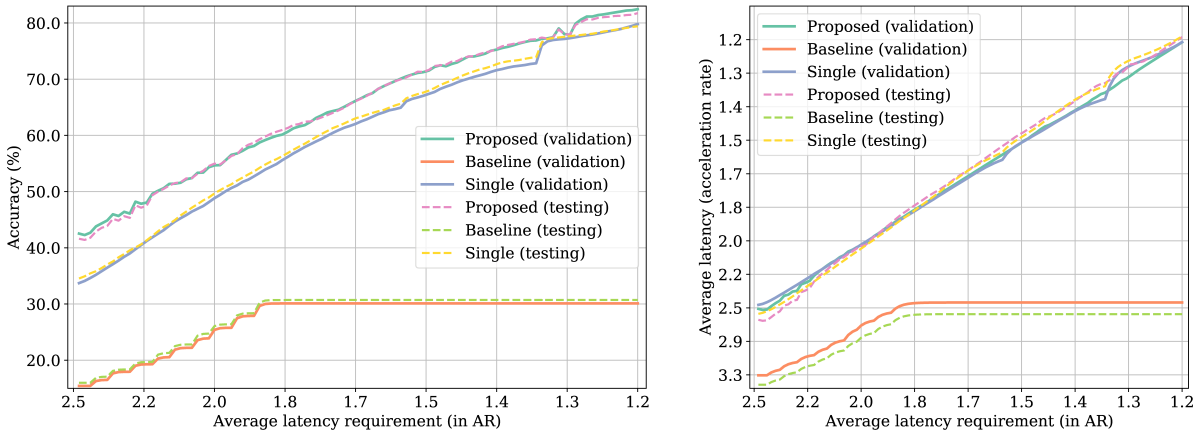
where $\delta = \min\{\frac{\gamma\epsilon^2}{16B^2}, \frac{\epsilon^2}{64\gamma}, \frac{2A}{T}\}$, $D = \max_{x,\tilde{x}\in S}|x-\tilde{x}|$. *Then, if the initial point* $\bar{\mathbf{t}}^{(0)}$ *is feasible, Algorithm 1 returns a weak* ϵ*-stationary point of* $\mathbf{P}_2$ *within* $\mathcal{O}(\epsilon^{-4})$ *proximal gradient steps.*



(a) Inference accuracy

(b) Inference latency

Figure 1: Experimental results on LSTM over DSA-19.



(a) Inference accuracy

(b) Inference latency

Figure 2: Experimental results on LSTM over Google-30.

Proof. We have shown in Lemma 1 that o and c are both smooth and weakly convex. This, together with the compactness of S^{N-1}, implies that $\mathbf{P}_2$ satisfies all the assumptions made in Theorem 4 of Lin *et al.* (Lin, Ma, and Xu 2019). Therefore, under the same setting as Theorem 4 of Lin *et al.*, Algorithm 1 possesses the theoretical property of the iPPP method with weakly-convex constraints, apparently yielding the desired results. □

To enhance the practical efficiency and robustness of Algorithm 1, we further propose a warm-start strategy for providing the initial guess $\bar{\mathbf{t}}^{(0)}$. Specifically, we let $\tilde{\mathbf{t}} \in S^{N-1}$ be the point corresponding to the threshold setting where all the thresholds are set to the lower confidence limit; for instance, when using the softmax response as the confidence measure, we have $\tilde{\mathbf{t}} = \mathbf{0}$. Then, we set $\bar{\mathbf{t}}^{(0)}$ by

$$\bar{\mathbf{t}}^{(0)} = \tilde{\mathbf{t}} + s^* \tilde{\mathbf{d}}, \tag{20}$$

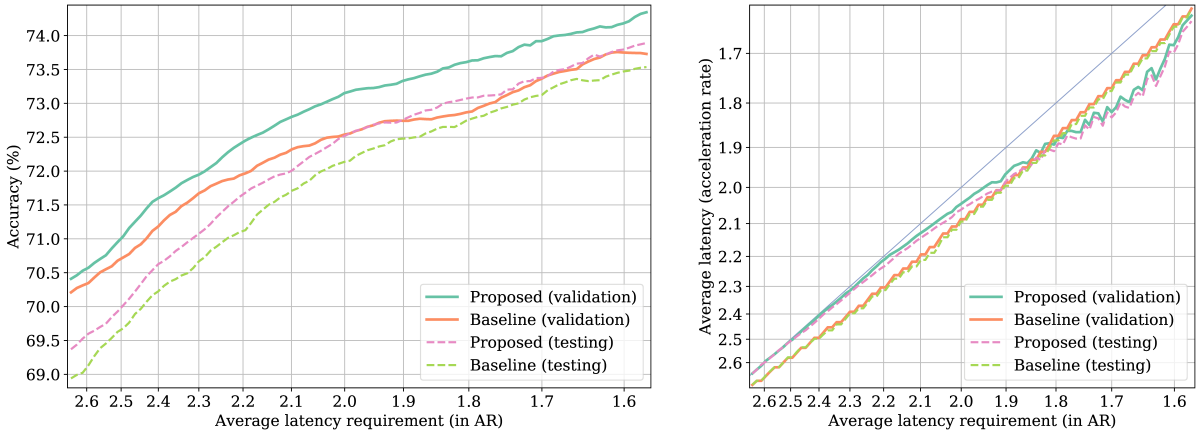
where $\tilde{\mathbf{d}} \in \mathbb{R}^{N-1}_{>0}$ is a pre-specified positive vector, and s^* is the root of the following nonlinear equation with respect to $s \in \mathbb{R}$:

$$c(\tilde{\mathbf{t}} + s\tilde{\mathbf{d}}) = 0. \tag{21}$$

The motivation of this warm-start strategy is to perform a descent step from $\tilde{\mathbf{t}}$ in the descent direction $\tilde{\mathbf{d}}$, where the step size is chosen via exact line search. In Appendix C, we give a more detailed explanation of this strategy. In practice, we find that by setting $\tilde{\mathbf{d}}$ to be the all-ones vector, this strategy often produce a good starting point for Algorithm 1.

Experiments

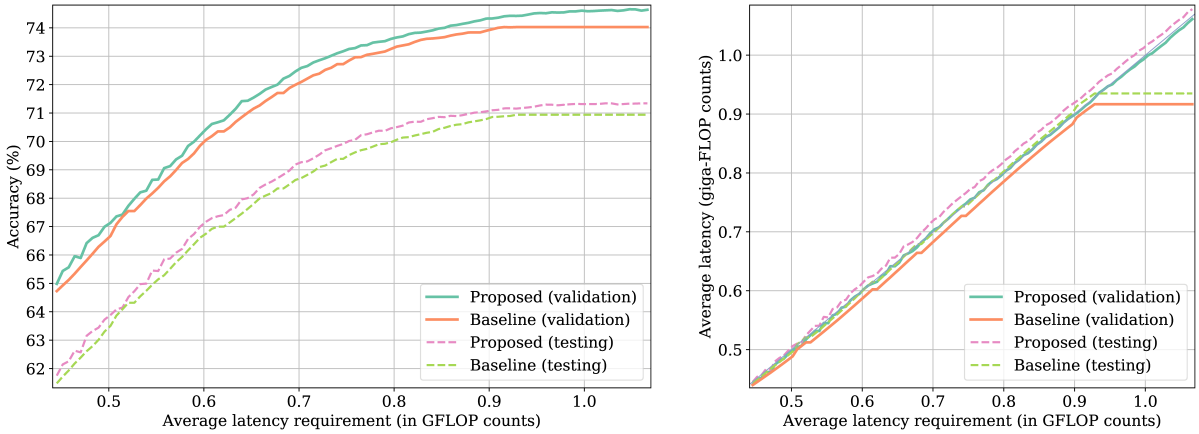
We evaluate the effectiveness and efficiency of our threshold determination method using three representative object recognition early-exit models, *i.e.*, a B-AlexNet (Teerapittayanon, McDanel, and Kung 2016) on CIFAR-10, an S-ResNet-18 (Zhang et al. 2019) on CIFAR-100, and a MSDNet (Huang et al. 2017) on ImageNet. For comparison, the heuristic of Huang *et al.* (Huang et al. 2017) is adopted as



(a) Inference accuracy

(b) Inference latency

Figure 3: Experimental results on S-ResNet-18 over CIFAR-100.



(a) Inference accuracy

(b) Inference latency

Figure 4: Experimental results on MSDNet over ImageNet.

the baseline method. As revealed by the literature review, this heuristic is so far the only method capable of handling average latency requirements. We also evaluate our method using two Long Short-Term Memory (LSTM) models, respectively on two time-series datasets, namely Google-30 (Warden 2018) and DSA-19 (Altun, Barshan, and Tunçel 2010), where the single-threshold method is served as another baseline for comparison. For details on the datasets, model architectures and training, we refer the reader to Appendix D. Note that the exit point numbers of B-AlexNet, S-ResNet-18 MSDNet, the LSTM model over DSA-19 and the LSTM model over Google-30 are respectively 3, 5, 5, 13 and 17. Data and code to reproduce all results are available at https://github.com/XinruiTan/AAAI21.

In our experiments on the three object recognition models, we set $k = 30$, $T = 100$, $\gamma = 10$, $J = 10$, $\mu = 1$, $\eta = 100$ and use all-ones vector as the search direction $\tilde{\mathbf{d}}$ for our warm-start strategy. Since the latency metrics of the three models are different, we choose $\beta = 2 \times 10^7$ for B-AlexNet, $\beta = 500$ for S-ResNet-18 and $\beta = 10^{-14}$ for MSDNet. For the experiments on the two LSTM models, we increase k to 100 and keep all other algorithm parameters the same as in the experiments on S-ResNet-18. We remark that, in practical applications, the penalty parameter β is the only parameter needing to be tuned, and the tuning of β can be carried out efficiently, so that parameter tuning is usually not an issue. Specifically, setting the smoothness parameter η is well-known to be a challenge for Nesterov's APG method, and is also challenging for the iPPP method. However, even for our problem $\mathbf{P}_2$ with a quite large k, we empirically find that, with a proper setting of β and a not-too-large γ, the inner subproblem is always 100-smooth and 1-strongly convex. Moreover, the value of β can be roughly estimated according to the magnitude of $d_N - d_1$.

Effectiveness Evaluation

For each trained model, we use the validation set to determine the thresholds under different average latency require-

ments, and evaluate our method against the baseline methods not only on the validation set to examine whether our method can well tackle our target optimization problem, but also on the test set to examine whether the threshold settings produced by our method can generalize well beyond the data used for threshold determination. The experimental results on S-ResNet-18, MSDNet and the two LSTM models are respectively plotted in Figures 1, 2, 3 and 4, where the curves labeled by 'Baseline', 'Single' and 'Proposed' are, respectively, the results of Huang *et al.*'s baseline heuristic, the single-threshold method and our method. Due to space limitations, we report the experimental results on B-AlexNet in Appendix E. Note that following the original papers (Zhang et al. 2019; Huang et al. 2017), for S-ResNet-18 and MSDNet, we measure the inference latency in terms of acceleration rate (AR) and FLOP counts, respectively; while for the two LSTM models, the inference latency is also measured in terms of AR.

For the experiments on the two LSTM models, as illustrated in Figures 1b and 2b, our method always produces a threshold setting that satisfies the average latency requirement Γ. Meanwhile, the proposed method significantly outperforms the two baseline methods for almost all the Γ settings, see Figures 1a and 2a. In particular, for the Google-30 dataset, the accuracy gains of our method over the single-threshold method on the validation and test sets are, respectively, up to 9.01% and 7.95%; for the DSA-19 dataset, these gains are, respectively, up to 7.89% and 7.07%. Comparing with the baseline heuristic of Huang *et al.*, these achieved accuracy gains are even more dramatic.

For the experiments on S-ResNet-18, we vary the average latency requirement between ARs of 1.57 and 2.67. In Figure 3b, it can be seen that all the threshold settings produced by our method satisfy their corresponding average latency requirements; in Figure 3a, it also can be seen that our method consistently achieves higher accuracies than the baseline heuristic on both the validation and test sets. When handling the same average latency requirement, our method, compared to the baseline heuristic, improves the validation and testing accuracies by up to 0.79% and 0.6%, respectively.

For the experiments on MSDNet, we vary the average latency requirement from 0.43 to 1.07 giga-FLOP (GFLOP) counts. As shown in Figure 4, on the validation set, our method always produces a threshold setting with no or negligible violation of the average latency requirement, implying that the approximation error is small; and under the same average latency requirements, our method consistently outperforms the baseline heuristic with a maximum improvement of 0.63% in accuracy. However, on the test set, it is trivial to compare the proposed and baseline methods, since both methods produce some threshold settings that violate the corresponding average latency requirements. Qualitatively, our method tends to more aggressively trade-off latency for accuracy with respect to the validation set, and thereby leads to higher accuracies and higher violations of the average latency requirement on the test set. We assert that this generalization issue is because the samples in the test set have greater inference difficulty than those in the validation set, which is reflected by the significant gaps between the validation and testing accuracy curves in Figure 4a. It is shown by the additional experimental results in Appendix E that if the samples used for threshold determination have greater inference difficulty than the samples encountered by the model in use, our method would not suffer from such a generalization issue.

From the experimental results, it can be seen that the accuracy improvement by our method is more significant on the LSTM models than that on S-ResNet-18 and MSDNet. This is because of two reasons: first, the LSTM models have more exit points, meaning that the impact of threshold setting on accuracy is much greater; second, the baseline heuristic generally works quite well when the number of exit points is small.

Efficiency Evaluation

We implement our method in Python, and measure its execution time on an Intel quad-core 2.9 GHz CPU. On average, our method requires 0.52, 1.06, 3.82, 2.78 and 8.59 seconds of execution time (with standard deviations of 0.07, 0.12, 0.05, 0.54, 0.41 seconds), respectively, for B-AlexNet, S-ResNet-18, MSDNet, the LSTM model over DSA-19 and the LSTM model over Google-30. The results demonstrate good scalability in both the number of exit points and the size of the validation set. From a technical standpoint, the execution time of our method is dominated by the gradient computation, which can be very efficiently carried out using closed-form expressions. As compared with the baseline heuristic, the proposed method is more time consuming, owing to its iterative nature. However, the execution time required by our method is still negligible as compared with the model training time.

Conclusion

In this paper, we considered the question: how to perform the adaptive early-exit inference with an awareness of the inference latency requirements? To address this question, we presented an approximate formulation of the threshold determination problem for controlling the adaptive early-exit inference processes, where the objective is to find a threshold setting that satisfies the predefined average latency requirement and maximizes the inference accuracy. Furthermore, We introduced a threshold determination method that ensures global convergence to a near-stationary point of the formulated problem, and empirically verified its superior performance over an existing heuristic. As future work, we plan to develop methods with fewer parameters to find threshold settings that have good generalization properties.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (No. 2019YFB1005200).

Broader Impact

We believe that our study can advance the development of adaptive early-exit inference technique in two-folds: first,

we have provided a useful on-demand tool for both the academic and industrial communities to control the accuracy-latency trade-off of early-exit models; second, we hope that our problem formulation will inspire further research on the fine control of the adaptive early-exit inference processes.

References

Aketi, S. A.; Panda, P.; and Roy, K. 2020. Relevant-features based Auxiliary Cells for Energy Efficient Detection of Natural Errors. *arXiv preprint arXiv:2002.11052* .

Altun, K.; Barshan, B.; and Tunçel, O. 2010. Comparative study on classifying human activities with miniature inertial and magnetic sensors. *Pattern Recognition* 43(10): 3605–3620.

Baccarelli, E.; Scardapane, S.; Scarpiniti, M.; Momenzadeh, A.; and Uncini, A. 2020. Optimized training and scalable implementation of Conditional Deep Neural Networks with early exits for Fog-supported IoT applications. *Information Sciences* 521: 107–143.

Berestizshevsky, K.; and Even, G. 2019. Dynamically Sacrificing Accuracy for Reduced Computation: Cascaded Inference Based on Softmax Confidence. In *International Conference on Artificial Neural Networks*, 306–320. Springer.

Cai, H.; Zhu, L.; and Han, S. 2018a. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332* .

Cai, H.; Zhu, L.; and Han, S. 2018b. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332* .

Cheng, B.; Xiao, R.; Wang, J.; Huang, T.; and Zhang, L. 2019. High frequency residual learning for multi-scale image classification. *arXiv preprint arXiv:1905.02649* .

Cordella, L. P.; De Stefano, C.; Tortorella, F.; and Vento, M. 1995. A method for improving classification reliability of multilayer perceptrons. *IEEE Transactions on Neural Networks* 6(5): 1140–1147.

Courbariaux, M.; Bengio, Y.; and David, J.-P. 2015. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, 3123–3131.

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

Dennis, D.; Pabbaraju, C.; Simhadri, H. V.; and Jain, P. 2018. Multiple instance learning for efficient sequential data classification on resource-constrained devices. In *Advances in Neural Information Processing Systems*, 10953–10964.

Ghiasi, G.; Lin, T.-Y.; and Le, Q. V. 2019. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7036–7045.

Goetschalckx, K.; Moons, B.; Lauwereins, S.; Andraud, M.; and Verhelst, M. 2018. Optimized hierarchical cascaded processing. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 8(4): 884–894.

Grapiglia, G. N.; and Yuan, Y.-x. 2019. On the Complexity of an Augmented Lagrangian Method for Nonconvex Optimization. *arXiv preprint arXiv:1906.05622* .

Han, S.; Liu, X.; Mao, H.; Pu, J.; Pedram, A.; Horowitz, M. A.; and Dally, W. J. 2016. EIE: efficient inference engine on compressed deep neural network. *ACM SIGARCH Computer Architecture News* 44(3): 243–254.

Han, S.; Pool, J.; Tran, J.; and Dally, W. 2015. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, 1135–1143.

Hartvigsen, T.; Sen, C.; Kong, X.; and Rundensteiner, E. 2019. Adaptive-halting policy network for early classification. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 101–110.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* .

Hong, M. 2016. Decomposing linearly constrained nonconvex problems by a proximal primal dual approach: Algorithms, convergence, and applications. *arXiv preprint arXiv:1604.00543* .

Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* .

Hu, T.-K.; Chen, T.; Wang, H.; and Wang, Z. 2020. Triple wins: Boosting accuracy, robustness and efficiency together by enabling input-adaptive inference. *arXiv preprint arXiv:2002.10025* .

Huang, G.; Chen, D.; Li, T.; Wu, F.; van der Maaten, L.; and Weinberger, K. Q. 2017. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844* .

Huang, M.-Y.; Lai, C.-H.; and Chen, S.-H. 2017. Fast and accurate image recognition using Deeply-Fused Branchy Networks. In *2017 IEEE International Conference on Image Processing (ICIP)*, 2876–2880. IEEE.

Jayakodi, N. K.; Chatterjee, A.; Choi, W.; Doppa, J. R.; and Pande, P. P. 2018. Trading-off accuracy and energy of deep inference on embedded systems: A co-design approach. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37(11): 2881–2893.

Kong, W.; Melo, J. G.; and Monteiro, R. D. 2019. Complexity of a quadratic penalty accelerated inexact proximal point method for solving linearly constrained nonconvex composite programs. *SIAM Journal on Optimization* 29(4): 2566–2593.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images. *Master's thesis, University of Toronto* .

Leroux, S.; Bohez, S.; Verbelen, T.; Vankeirsbilck, B.; Simoens, P.; and Dhoedt, B. 2015. Resource-constrained classification using a cascade of neural network layers. In *2015 International Joint Conference on Neural Networks (IJCNN)*, 1–7. IEEE.

Li, E.; Zeng, L.; Zhou, Z.; and Chen, X. 2019a. Edge AI: On-Demand Accelerating Deep Neural Network Inference via Edge Computing. *IEEE Transactions on Wireless Communications* .

Li, H.; Zhang, H.; Qi, X.; Yang, R.; and Huang, G. 2019b. Improved Techniques for Training Adaptive Deep Networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 1891–1900.

Lin, Q.; Ma, R.; and Xu, Y. 2019. Inexact Proximal-Point Penalty Methods for Non-Convex Optimization with Non-Convex Constraints. *arXiv preprint arXiv:1908.11518v1* .

Lo, C.; Su, Y.-Y.; Lee, C.-Y.; and Chang, S.-C. 2017. A dynamic deep neural network design for efficient workload allocation in edge computing. In *2017 IEEE International Conference on Computer Design (ICCD)*, 273–280. IEEE.

Nesterov, Y. 2018. *Lectures on convex optimization*, volume 137. Springer.

Panda, P.; Sengupta, A.; and Roy, K. 2016. Conditional deep learning for energy-efficient and enhanced pattern recognition. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 475–480. IEEE.

Park, E.; Kim, D.; Kim, S.; Kim, Y.-D.; Kim, G.; Yoon, S.; and Yoo, S. 2015. Big/little deep neural network for ultra low power inference. In *2015 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, 124–132. IEEE.

Phuong, M.; and Lampert, C. H. 2019. Distillation-Based Training for Multi-Exit Architectures. In *Proceedings of the IEEE International Conference on Computer Vision*, 1355–1364.

Qi, H.; Sparks, E. R.; and Talwalkar, A. 2017. Paleo: A performance model for deep neural networks. In *International Conference on Learning Representations (ICLR)*.

Sahni, S. 1974. Computationally related problems. *SIAM Journal on Computing* 3(4): 262–279.

Stamoulis, D.; Chin, T.-W.; Prakash, A. K.; Fang, H.; Sajja, S.; Bognar, M.; and Marculescu, D. 2018. Designing adaptive neural networks for energy-constrained image classification. In *Proceedings of the International Conference on Computer-Aided Design*, 1–8.

Sun, H.; and Pang, Y. 2018. GlanceNets—Efficient convolutional neural networks with adaptive hard example mining. *Science China Information Sciences* 61(10): 109101.

Teerapittayanon, S.; McDanel, B.; and Kung, H.-T. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, 2464–2469. IEEE.

Teerapittayanon, S.; McDanel, B.; and Kung, H.-T. 2017. Distributed deep neural networks over the cloud, the edge and end devices. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 328–339. IEEE.

Venkataramani, S.; Raghunathan, A.; Liu, J.; and Shoaib, M. 2015. Scalable-effort classifiers for energy-efficient machine learning. In *Proceedings of the 52nd Annual Design Automation Conference*, 1–6.

Warden, P. 2018. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209* .

Wu, W.; He, D.; Tan, X.; Chen, S.; Yang, Y.; and Wen, S. 2020. Dynamic Inference: A New Approach Toward Efficient Video Action Recognition. *arXiv preprint arXiv:2002.03342* .

Yokoo, S.; Iizuka, S.; and Fukui, K. 2019. MLSNet: Resource-Efficient Adaptive Inference with Multi-Level Segmentation Networks. In *2019 IEEE International Conference on Image Processing (ICIP)*, 1510–1514. IEEE.

Yuan, Z.; Wu, B.; Liang, Z.; Zhao, S.; Bi, W.; and Sun, G. 2019. S2DNAS: Transforming Static CNN Model for Dynamic Inference via Neural Architecture Search. *arXiv preprint arXiv:1911.07033* .

Zhang, C.; Ren, M.; and Urtasun, R. 2018. Graph hypernetworks for neural architecture search. *arXiv preprint arXiv:1810.05749* .

Zhang, L.; Tan, Z.; Song, J.; Chen, J.; Bao, C.; and Ma, K. 2019. SCAN: A scalable neural networks framework towards compact and efficient models. In *Advances in Neural Information Processing Systems*, 4029–4038.