EERO: Early Exit with Reject Option for Efficient Classification with limited budget

Florian Valade[1,2] **Mohamed Hebiri**[1] **Paul Gay**[3]

[1]LAMA , Université Gustave Eiffel
[2]Fujitsu
[3]Université De Pau Et Des Pays De L'adour

Abstract

The increasing complexity of advanced machine learning models requires innovative approaches to manage computational resources effectively. One such method is the Early Exit strategy, which allows for adaptive computation by providing a mechanism to shorten the processing path for simpler data instances. In this paper, we propose EERO, a new methodology to translate the problem of early exiting to a problem of using multiple classifiers with reject option in order to better select the exiting head for each instance. We calibrate the probabilities of exiting at the different heads using aggregation with exponential weights to guarantee a fixed budget. We consider factors such as Bayesian risk, budget constraints, and head-specific budget consumption. Experimental results demonstrate that our method achieves competitive compromise between budget allocation and accuracy.

1 INTRODUCTION

Nowadays, vision models are increasing in size rising the issue of their complexity and computation costs. There exist different strategies to train lighter deep learning networks, such as quantization and pruning [Liang et al., 2021], distillation [Touvron et al., 2021], and dynamic inference where the network adapts its topology on the fly to the input data [Han et al., 2021]. Among them, Early Exit [Laskaridis et al., 2021] is an orthogonal approach which aims at adapting the amount of computation to each input data point, exploiting that most neural networks can be approximated as a stack of layers which process the data sequentially. The idea is to add auxiliary heads at regular intervals along the network (see Figure 1) which are able to produce a prediction with the current state of the features. The intuition is that easy cases can be processed with the first few layers. Those heads can also be used to analyze model's layers [Chen et al., 2020, Kaya et al., 2019] and improve training [Teerapittayanon et al., 2016].

Despite its advantages, one of the main challenges in applying Early Exit during inference is to determine the appropriate moment to exit for a given input. This objective is intricately connected to assessing confidence in the predictions of neural networks. Importantly it requires to build a rule – that usually relies on a test that asks whether the score for the prediction is higher or not than a given threshold – at the level of each auxiliary head of the network and one possible ultimate goal may be to calibrate all thresholds appropriately in order to meet an overall objective. Specifically, in our contribution, we focus on *Budgeted Batch classification* [Huang et al., 2018], a strategy where a fixed computational budget is allocated for processing a batch of data. The strategy involves calculating thresholds to allocate these resources efficiently across different data points enhancing overall accuracy. In principle, this approach offers improved performance because it allows for the conservation of computational resources on simpler cases, which can then be reallocated to improve accuracy on more complex cases. However, practical investigations into this framework have been limited, with most studies concentrating on architectural design or the reliability of confidence scores. Those few that have addressed Budgeted Batch Classification often make additional assumptions for practicality [Huang et al., 2018] or utilize less-than-ideal algorithms without fully exploring the associated mathematical challenges [Wang et al., 2021].

Contributions In this work, we present EERO (Early Exit with Reject Option), a novel framework designed to augment Early Exit strategies in Budgeted Batch Classification for inference tasks. The EERO methodology progresses through a series of steps: Initially, we train auxiliary heads to refine decision-making. Subsequently, we translate the budget limitation to probabilities of classification at each head

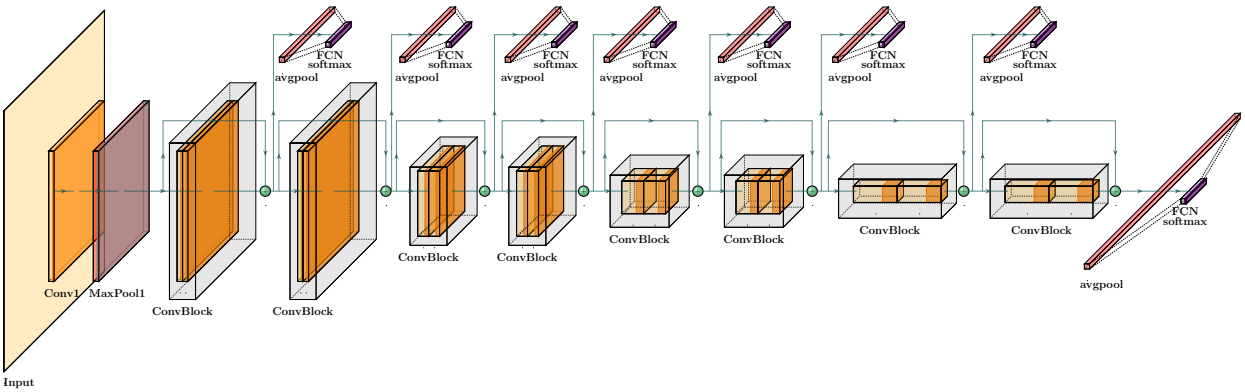


Figure 1: Illustration of the Early Exit principle in a convolutional architecture.

based on aggregation with exponential weights [Dalalyan and Tsybakov, 2008]. We then use a calibration set and those probabilities to calibrate head specific exit thresholds. We based this calibration of the thresholds on learning with rejection option arguments [Chow, 1957]. (Also named *learning with abstention* and later *selective learning* in the literature.) Finally, for each data point and at the level of each head we compute a score. If this score exceeds the threshold, we take this head output as the final prediction. Otherwise, we proceed with the following head in a similar way.

Our key contributions are: (i) Developing an optimal classification process within a specified GFlops[1] budget, ensuring efficient data processing while balancing accuracy; (ii) Adapting and generalizing EERO to various architectures like ResNet [He et al., 2016], ConvNext [Liu et al., 2022], and MSDNet [Huang et al., 2018], suitable for multiple auxiliary heads; (iii) Validating our approach through extensive benchmarks on CIFAR-100 and ImageNet datasets, proving its effectiveness in maintaining resource constraints, reducing computational load, and enhancing model accuracy.

The paper is organized as follows: Section. 2 reviews relevant literature. Section. 3 introduces our proposed EERO method, including its statistical framework (Section 3.1) and detailed methodology (Section 3.2). Section. 4 presents experimental results, applying EERO on ImagrNet.

2 RELATED WORK

One of the first works on Early Exit [Teerapittayanon et al., 2016] proposed to use auxiliary heads and a weighted sum of losses. The focus was essentially on the design and the positions of the auxiliary heads. MSDNet [Huang et al., 2018] is an architecture where the heads are carefully designed so that early features which are assumed to be unsuitable for classification are refined. Although this leads to better performances, the complex resulting heads reduce their number, and therefore the flexibility of the model, and makes it more difficult to adapt to new architectures. Overall, the position and the number of the auxiliary heads remains an open discussion for convolutional networks [Lin et al., 2022] and transformers [Bakhtiarnia et al., 2021].

Determining the optimal timing for an early exit in neural network inference is a crucial aspect of efficient model design. A widely used method is the *Threshold-Based Approach*, where the exit decision is based on a predefined threshold on auxiliary heads output probabilities, focusing on different metrics of the probability distribution [Huang et al., 2018, Bolukbasi et al., 2017]. Another strategy is the *Patience-Based Strategy*, requiring consecutive heads to agree on a prediction before exiting, thus leveraging sequential predictions for reliability [Zhou et al., 2020]. Additionally, more sophisticated methods involve *Halting Scores*, where scores are accumulated from each layer's output, integrating model confidence into the exit process [Figurnov et al., 2017]. In complex scenarios, *Reinforcement Learning Techniques* are employed to finely balance computational efficiency and prediction accuracy, especially in dynamic inference settings [Wang et al., 2018, Wu et al., 2018].

In the context of *Budgeted batch classification*, thresholds are not determined for individual images but for a batch as a whole, facilitating a more strategic deployment of computational resources. To date, the literature presents two notable algorithms addressing this problem in the realm of Early Exit strategies. The authors in [Wang et al., 2021] employ a genetic algorithm to optimize these thresholds. Conversely, the work [Huang et al., 2018] operates under the assumption that each exit point in the network has an equal and predetermined likelihood of accurately classifying an image. While this simplifies the optimization problem, it introduces an additional hyper-parameter that is not inherently related to computational complexity.

[1]Flops: floating point operations per second.

In addition, other contexts have been well studied in the literature, such as model cascading or learning-to-defer. Model cascading involves using separate, independent models of increasing complexity, where each model decides whether to classify an instance or defer to the next model in the sequence [Jitkrittum et al., 2024, Gupta et al., 2024]. Learning to defer, on the other hand, transfers decision-making to human operators when the model's confidence is low [Okati et al., 2021, Charusaie and Samadi, 2024]. Our work differs by presenting a comprehensive formulation that is grounded on the principles of reject option learning theory on a multi headed classification model with a focus on minimizing cost.

Learning with the reject option, a method to abstain from predictions under uncertainty, is crucial in our work. Initially explored in [Chow, 1957] and advanced through conformal prediction [Vovk et al., 1999, 2005], this concept has evolved significantly [Herbei and Wegkamp, 2006, Naadeem et al., 2010, Grandvalet et al., 2009, Yuan and Wegkamp, 2010, Lei, 2014, Cortes et al., 2016, Denis and Hebiri, 2019]. It generally encompasses strategies for predefined coverage, rejection rates, or a balance of both. In EERO, rejection is adapted at the head level, deciding if an instance $\mathbf{x}$ should exit early or continue processing. This application to deep learning for energy efficiency is novel, though the reject option has been previously applied to various learning problems [Denis et al., 2020, 2022].

3 METHOD

In this section, we introduce the Early Exit with Reject Option (EERO) framework, applying reject option learning theory to enable efficient early exits in neural networks. We detail the statistical underpinnings and describe how EERO strategically manages resources to balance accuracy with computational expenditure. This approach not only optimizes performance but also ensures strict compliance with predefined computational constraints.

3.1 STATISTICAL FRAMEWORK – CLASSIFICATION WITH REJECT OPTION

This section describes the mathematical framework for Early Exit. Let $(\mathbf{X}, Y)$ be a random couple distributed according to a distribution $\mathbb{P}$ on $\mathcal{X} \times [K]$, where $[K] := \{1, \dots, K\}$.

Here, $\mathcal{X} \subset \mathbb{R}^d$ is the feature space, and Y is the label corresponding to the feature $\mathbf{X}$. We focus on the problem of K-class classification with $K \geq 2$ and one of our goals is to build a prediction rule $g : \mathcal{X} \to [K]$ that reduces the misclassification risk $\mathbb{P}(g(\mathbf{X}) \neq Y)$. This risk is minimized by the Bayes rule g^* that is given, in the multi-class setting, for all $\mathbf{x} \in \mathcal{X}$ by

$$g^*(\mathbf{x}) = \underset{k=1,\dots,K}{\operatorname{argmax}}\, p_k(\mathbf{x}) \ , \tag{1}$$

where $p_k(\mathbf{x}) = \mathbb{P}(Y = k|\mathbf{X} = \mathbf{x})$ are the conditional probabilities. Because the distribution $\mathbb{P}$ is unknown, the Bayes rule itself is unknown, and we need to approximate it. In general, this approximation seeks to maximize the classification accuracy. In our case, the goal is also to reduce the energy consumption of the model, and thus, we assume a constraint on the computation budget available. Notably, we will show that the latter is strongly connected to classification with reject option (classification with abstention) that we describe in Section 3.2.

The framework of classification with reject option assumes that classifiers are allowed to abstain from classifying (on the empirical side, this means that the classifier abstains on a given proportion of the data). Let $\varepsilon \in (0, 1)$ be a parameter that denotes the probability of classifying an instance. The optimal rule that abstains with probability $1 - \varepsilon$ is given by:

Definition 3.1. Let $\varepsilon \in (0, 1)$. The optimal classifier with $1 - \varepsilon$ rejection rate is defined as

$$h^*_\varepsilon \in \underset{h}{\operatorname{argmin}} \{\mathbb{P}(\{h(\mathbf{X}) \neq Y\} \cap \{h(\mathbf{X}) \neq \mathfrak{R}\}) \quad \text{s.t. } \mathbb{P}(h(\mathbf{X}) = \mathfrak{R}) = 1 - \varepsilon\} \ , \tag{2}$$

where h is a classifier that is allowed to reject, that is, $h : \mathcal{X} \to [K] \cup \{\mathfrak{R}\}$ and $\mathfrak{R}$ is the output when the classifier rejects all elements from $[K]$.

The opportunity of using the reject option is important in applications where ambiguity occurs between classes, which is often the case when the total amount of classes K is large. There are several ways to handle this reject option. In this paper, we constrain the rejection rate as it is in accordance with the resource limitation (see Sections 3.2.1 and 3.2.2).

Now, we aim at providing the explicit expression of the optimal rule given by Definition 3.1. Let us denote by s, the score function defined for each $\mathbf{x} \in \mathcal{X}$ by

$$s(\mathbf{x}) = \max_{k \in [K]} \{p_1(\mathbf{x}), \dots, p_K(\mathbf{x})\} \ . \tag{3}$$

From the definition of the Bayes rule (1), we can establish the following characterization of the optimal rule.

Proposition 3.2. *Assume the cumulative distribution function (CDF) F_s of $s(\mathbf{X})$ is continuous. Then, for all $\mathbf{x} \in \mathcal{X}$*

$$h^*_\varepsilon(\mathbf{x}) = \begin{cases} g^*(\mathbf{x}) & \text{if} \quad F_s(s(\mathbf{x})) \geq 1 - \varepsilon \\ \mathfrak{R} & \text{otherwise.} \end{cases}$$

In particular, we have $\mathbb{P}(h^*_\varepsilon(\mathbf{X}) = \mathfrak{R}) = 1 - \varepsilon$.

Proposition 3.2 extends the result established in [Denis and Hebiri, 2019] (*c.f.* Proposition 1) to multi-class setting. Its

proofs can be found in the Sec. D of the Appendix. The main assumption used to build this result is the continuity assumption on the CDF of $s(\mathbf{X})$ and requires that the random variable has no atoms. It ensures that the classifier h^*_ε has a rejection rate exactly $1-\varepsilon$. In the next section, we will see that from an empirical point of view, this condition can always be satisfied through randomization.

3.2 DATA-DRIVEN PROCEDURE

Before considering the reject option arguments and since we deal with an Early Exit strategy, we train a neural network with M possible exits that correspond to $M-1$ auxiliary heads and the classical output of the last layer, as illustrated in Figure 1. To this end, we collect a *labeled* dataset $\mathcal{D}_{n_1}$, that consists of $n_1 \in \mathbb{N}$ *i.i.d.* copies of $(\mathbf{X}, Y)$, and train for each head ℓ an estimator $(\hat{p}_1^\ell, \ldots, \hat{p}_K^\ell)$ of the conditional probability vector $(p_1, \ldots, p_K)$. Since larger ℓ means that we go deeper in the network, it is reasonable to assume that for all layers indices $\ell, \ell' \in [M-1]$ with $\ell < \ell'$, the auxiliary head ℓ' consumes more resources than the auxiliary head ℓ and in general, yet this increase in consumption might come with a better accuracy.

3.2.1 Classification rule based on reject option

Our methodology highlights that early exiting can be efficiently performed, borrowing tools from learning with reject option. Let us then define, for each auxiliary head ℓ, a rejection rate $1-\varepsilon^\ell \in (0,1)$ – that will be specified later. In order to specify the prediction rule for each head, we will mimic the optimal rule provided by Proposition (3.2) using the plug-in principle. This step requires collecting a sample of *unlabeled* instances $\mathcal{D}_N$, that consists in N *i.i.d.* copies of $\mathbf{X}$. Since the process is the same at each auxiliary head, let us develop our methodology for a specific head ℓ.

The goal is to understand whether the head ℓ is a good early exit for a given instance $\mathbf{x} \in \mathcal{X}$. Translating this into the classification with reject option vocabulary, the question becomes whether classifier

$$\hat{g}^\ell(\cdot) = \underset{k\in[K]}{\operatorname{argmax}} \left\{\hat{p}_k^\ell(\cdot) + u_k\right\} \ , \tag{4}$$

should classify the instance $\mathbf{x}$ or reject it. The u_k variable is introduced to randomize the $\hat{p}_k^\ell$ estimations for a technical reason that we now explain. Let us denote by $(u_k)_{k\in[K]}$ *i.i.d.* variables which follow a uniform distribution on $[0,u]$ with u being a non-negative real number which is usually chosen very small. (In practice, we set $u = 10^{-5}$.) It ensures that the random variables $p_k^\ell(\mathbf{X}) + u_k$ has no atoms which is the key argument (see proof in appendix D) to control suitably the rejection rate of the produced classifier with reject option. Formally, this classifier is the empirical counterpart of the classifier with reject option given by Proposition 3.2 and is based on the empirical score function given for all $\mathbf{x}$ by

$$\hat{s}^\ell(\mathbf{x}) = \max_{k\in[K]} \left\{\hat{p}_1^\ell(\mathbf{x}) + u_1, \ldots, \hat{p}_K^\ell(\mathbf{x}) + u_K\right\} \ . \tag{5}$$

Then we can derive the plug-in estimator of the classifier with reject option h^*_ε at the level of the ℓ-th auxiliary head.

Definition 3.3. For all $\mathbf{x} \in \mathcal{X}$

$$\hat{h}^\ell_{\varepsilon^\ell}(\mathbf{x}) = \begin{cases} \hat{g}^\ell(\mathbf{x}) & \text{if} \quad \hat{F}_{\hat{s}^\ell}(\hat{s}^\ell(\mathbf{x})) \geq 1-\varepsilon^\ell \\ \mathfrak{R} & \text{otherwise,} \end{cases}$$

where conditional on the dataset $\mathcal{D}_n$, we denote by $\hat{F}_{\hat{s}^\ell}(t) = \frac{1}{N}\sum_{\mathbf{X}\in\mathcal{D}_N} \mathbb{1}_{\{\hat{s}^\ell(\mathbf{X})\leq t\}}$ the empirical CDF of $\hat{s}^\ell(\mathbf{X})$ on the *unlabeled* sample $\mathcal{D}_N$.

Remark 3.4. The score function $\hat{s}^\ell$ in the above definition can be replaced by any suitable metrics providing an estimation of the prediction confidence. We tried entropy-based confidence and breaking ties [Luo et al., 2005] (*i.e.,* the difference between the two highest scores) in our experiments and selected the latter which performed slightly better.

According to the above definition, the classifier $\hat{h}^\ell_{\varepsilon^\ell}$ abstains when it is not confident about the classification. In our case, having $\hat{h}^\ell_{\varepsilon^\ell}$ assigns the output $\mathfrak{R}$ to a given instance $\mathbf{x}$ means that the observation should not be treated by $\hat{h}^\ell_{\varepsilon^\ell}$, but rather should be delayed to next auxiliary head treatment. In contrast, when $\hat{h}^\ell_{\varepsilon^\ell}(\mathbf{x}) = \hat{g}^\ell(\mathbf{x})$, then we use the early exit at the head ℓ and the observation $\mathbf{x}$ does not go through the rest of the network, thus reducing the computation. We can establish the following result whose proof can be found in Section D of the Appendix.

Proposition 3.5. *For all $\ell \in [M]$ and all $\varepsilon^\ell \in (0,1)$, there exists a constant $C > 0$ such that, whatever the distribution $\mathbb{P}$ of the data and whatever the estimators $\hat{s}^\ell$ of the score function s we consider, we have*

$$\left|\mathbb{P}\left(\hat{h}^\ell_{\varepsilon^\ell}(\mathbf{X}) = \mathfrak{R}\right) - (1-\varepsilon^\ell)\right| \leq \frac{C}{\sqrt{N}} \ .$$

The above result confirms that the rejection rate of the classifier $\hat{h}^\ell_{\varepsilon^\ell}$ is indeed of the right order. However, this result suggests that the head ℓ might reject more data than it should (by a proportion of order $C/\sqrt{N}$ which is not suitable from the budget perspective. In order to solve this issue, it is sufficient to impose a smaller rate of rejection. More precisely, if we replace ε^ℓ by $\tilde{\varepsilon}^\ell = \varepsilon^\ell + C/\sqrt{N}$ when we run our algorithm, we force the rejection rate to be less than $1-\varepsilon^\ell$. From our numerical study, we observed that $C = \varepsilon^\ell$ leads to good results in order to ensure the budget limitation.

Remark 3.6. Our methodology, applicable for semi-supervised learning, capitalizes on both labeled and unlabeled data, making it ideal when acquiring labels is costly. If only labeled data is available, we advise splitting the dataset, facilitating budget control theoretically.

Remark 3.7. The result in Proposition 3.5 is valid when $M > 2$ only in the situations where the sets of classified instances by all heads are disjoint, meaning that each image is rejected by all heads but one. Since this is unlikely in practice, we need to adjust the rejection rate at the level of the head ℓ based on the rejection rates of the earlier heads. We will elaborate on this in Section 3.2.3.

3.2.2 Calibration of the rejection rates

To fully exploit the layered complexity of deep learning networks, our EERO methodology must handle multiple exits. We incorporate an aggregation with exponential weights [Cesa-Bianchi and Lugosi, 2006, Dalalyan and Tsybakov, 2008] to optimize the decision-making process at various network depths. This adaptation is crucial for leveraging the diverse representational capabilities of neural networks, ensuring computational efficiency, and maintaining high accuracy across multiple exit points. We recall that our main constraint here is the maximum allowed budget B in GFlops.

Let us then develop the process to build the vector $\hat{\boldsymbol{\varepsilon}} = (\hat{\varepsilon}^1, \ldots, \hat{\varepsilon}^M)$ of discrete probabilities that provides us the rates of classifying at each head (in other words, the $1 - \hat{\varepsilon}^\ell$ are the rejection rates). As inputs, we assume

- we have already trained all heads classifiers that are called $\hat{g}^1 \ldots \hat{g}^M$ (based on the first labeled dataset $\mathcal{D}_{n_1}$);
- we have already computed for each of the classifiers $\hat{g}^\ell$ an evaluation of its risk $\hat{R}^\ell = \frac{1}{n_2} \sum_{(\mathbf{X},Y)\in\mathcal{D}_{n_2}} \mathbb{1}_{\{\hat{g}^\ell(\mathbf{X})\neq Y\}}$ based on a second labeled dataset $\mathcal{D}_{n_2}$ that consists in *i.i.d.* copies of $(\mathbf{X}, Y)$ – notice that these error rates are already computed during any classic deep learning training;
- we have a prior distribution $\boldsymbol{\pi} = (\pi^1, \ldots, \pi^M)$ on the simplex Λ_{M-1} defined as $\pi^\ell = \frac{\left(\hat{B}^\ell\right)^{-1}}{\sum_{j=1}^M \left(\hat{B}^\ell\right)^{-1}}$, where $\hat{B}^\ell$ is the budget required by the head classifier $\hat{g}^\ell$ to provide an inference for one instance,
- we have fixed the overall budget B we are allowed to use and the size T of the batch of new data points we need to predict.

Our proposal is based on aggregation with exponential weights. The vector $\hat{\boldsymbol{\varepsilon}} = (\hat{\varepsilon}^1, \ldots, \hat{\varepsilon}^M) \in \Lambda_{M-1}$ is solution in $\boldsymbol{\varepsilon} = (\varepsilon^1, \ldots, \varepsilon^M)$ of the following minimization problem:

$$\min_{\boldsymbol{\varepsilon}\in\mathbb{R}^M} \sum_{\ell=1}^M \varepsilon^\ell \hat{R}^\ell + \beta \sum_{\ell=1}^M \varepsilon^\ell \log\left(\frac{\varepsilon^\ell}{\pi^\ell}\right), \tag{6}$$

$$s.t. \qquad \varepsilon^\ell \geq 0, \qquad \sum_{\ell=1}^M \varepsilon^\ell = 1, \qquad \sum_{\ell=1}^M \varepsilon^\ell \hat{B}^\ell \leq \bar{B}, \tag{7}$$

where $\beta \geq 0$ is a tuning parameter that controls the strength of the Kullback-Leibler divergence between $\boldsymbol{\varepsilon}$ and $\boldsymbol{\pi}$ and $\bar{B} = B/T$ is the average budget we can spend to infer one instance. Notice that the constraint $\sum_{\ell=1}^M \varepsilon^\ell \hat{B}^\ell \leq \bar{B}$ reads as $\sum_{\ell=1}^M (T\varepsilon^\ell)\hat{B}^\ell \leq B$. In particular, $T\varepsilon^\ell$ interprets as the number of data points that should be classified by the head ℓ so that the total budget remains less (or equal) than the allocated budget B. Considering the Lagrangian of this problem, we can exhibit the following form of the probability vector $\hat{\boldsymbol{\varepsilon}}$.

Proposition 3.8. *For all $\ell \in [M]$, the ℓ-th coordinates of the rejection rates vector $\hat{\boldsymbol{\varepsilon}}$ is given by*

$$\hat{\varepsilon}^\ell = \frac{\pi^\ell \exp\left\{-\frac{\hat{R}^\ell + \hat{\mu}\hat{B}^\ell}{\beta}\right\}}{\sum_{j=1}^M \pi^j \exp\left\{-\frac{\hat{R}^j + \hat{\mu}\hat{B}^j}{\beta}\right\}} \ ,$$

where $\hat{\mu} = \max\{0, \bar{\mu}\}$, with $\bar{\mu}$ being solution of

$$\sum_{\ell=1}^M (\bar{B} - \hat{B}^\ell)\pi^\ell \exp\left\{-\frac{\hat{R}^\ell + \bar{\mu}\hat{B}^\ell}{\beta}\right\} = 0 \ .$$

The only tuning parameter in the above procedure is the temperature β. Higher values force the probability vector $\hat{\boldsymbol{\varepsilon}}$ to get closer to the prior distribution $\boldsymbol{\pi}$ that is created to take into account the budget required by each head to produce a prediction. Last but not least, The empirical risk factors $\hat{R}^\ell$ in Equation (7) allow the rejection rates $\hat{\boldsymbol{\varepsilon}}^\ell$ to depend on the head performances on the training data, *i.e.,* layers with good classifiers will be selected more often.

One main advantage of aggregation with exponential weights is that it is supported by strong theoretical performance. Let us focus on the misclassification risk $\mathcal{R}(g) := \mathbb{P}\left(g(\mathbf{X}) \neq Y\right)$. Following the analysis in Dalalyan and Tsybakov [2008], Rigollet and Tsybakov [2012], we can show the following result.

Theorem 3.9. *Consider $\hat{g}^{EERO}$ the classifier resulting from our aggregation procedure (whose formal definition is given in*(16)*). Conditionally on $\mathcal{D}_{n_1}$, the data used to train the heads $\hat{g}^1 \ldots \hat{g}^M$, we have*

$$\mathcal{R}(\hat{g}^{EERO}) \leq \inf_{\ell\in[M]:\ \hat{B}^\ell\leq\bar{B}} \left\{\mathcal{R}(\hat{g}^\ell) + \beta\log(1/\pi^\ell)\right\} \ .$$

Notice that the above Oracle inequality shows that our aggregation procedure $\hat{g}^{\text{EERO}}$ is at least as good (up to a log factor) as the best head that satisfies the budget constraint. In addition, without *prior* knowledge on the budget, one may consider the uniform probabilities $\pi^\ell = 1/M$ and recover the classical bound of order $\log(M)$.

Remark 3.10. Our methodology fits well for the problem of Budgeted Batch Classification. However, the notion of batch is not important for our purpose. As expressed in the

above algorithm (6)-(7) (see also Proposition 3.8), the only information that is required to calibrate the probability of rejections vector $\hat{\boldsymbol{\varepsilon}}$ is the average budget for classifying one instance.

Remark 3.11. EERO methodology takes advantage of aggregation with exponential weights to compromise between accuracy and budget consumption. However in the particular case of only one auxiliary head, there is no need for aggregation – the budget is split between the auxiliary head and the last layer. We have developed such particular case in Appendix C and run experiments in that case as well using the ResNet-18 architecture illustrated in Figure 1 on CIFAR-100.

3.2.3 Heads with reject option

In the case of multiple heads, it is difficult to ensure the validity of Proposition 3.5 for all heads (*c.f.* Remark 3.7). However, we can guarantee a weaker result that is sufficient to ensure the good control on the allocated budget. Based on the previous section, we built M head classifiers on one hand and a probability vector $\hat{\boldsymbol{\varepsilon}}$ whose ℓ-th components gives the rates of classification at the ℓ-th auxiliary head on the other hand. The probability vector $\hat{\boldsymbol{\varepsilon}}$ takes into account both the accuracy and the resources of each head and allows achieving the suitable control on the budget. However, in the case of multiple auxiliary heads, a careful analysis of our methodology imposes some modifications according to the sequential calibration of the probabilities of classification.

The calibration of the classification rates of all heads is based on the estimation of the CDF F_s of the score function. Since there are $M-1$ auxiliary heads, we estimate $M-1$ times the function F_s. Importantly, all these estimates are built on the same dataset $\mathcal{D}_N$. While the calibration at the level of the first head is perfectly valid, the calibration of the classification rate starting from the second head needs to be adjusted to get the $\hat{\varepsilon}^\ell$ classification rate. In particular, for all $\ell \in \{2, \dots, M-1\}$, we enforce a higher classification rate as

$$\hat{\varepsilon}^\ell_{\text{seq}} = \sum_{j=1}^{\ell} \hat{\varepsilon}^j \ . \tag{8}$$

This choice is motivated by the fact that we need, for each head ℓ, to calibrate the threshold for the rejection rule so that at most a proportion $1 - \sum_{j=1}^{\ell} \hat{\varepsilon}^j$ of the data in $\mathcal{D}_N$ is rejected. If we consider this adjustment together with the correction we have detailed right after Proposition 3.5, we can show that for all $\ell \in [M]$, if we replace the probability of classification $\hat{\varepsilon}^\ell$ in Definition 3.3 by

$$\tilde{\varepsilon}^\ell_{\text{seq}} = \hat{\varepsilon}^\ell_{\text{seq}} + C/\sqrt{N} \ . \tag{9}$$

We have that $\mathbb{P}\left(\hat{h}^\ell_{\tilde{\boldsymbol{\varepsilon}}^\ell_{\text{seq}}}(\mathbf{X}) = \mathfrak{R}\right) \leq 1 - \sum_{j=1}^{\ell} \hat{\varepsilon}^j$ so that we can deduce the following result.

Theorem 3.12. *For all $\ell \in [L]$, the classification procedure at the head ℓ is such that*

$$\mathbb{P}\left(\hat{h}^\ell_{\tilde{\boldsymbol{\varepsilon}}^\ell_{\text{seq}}}(\mathbf{X}) = \mathfrak{R}\right) \leq \sum_{j=\ell+1}^{M} \hat{\varepsilon}^j \ .$$

The above statement means that after using the head ℓ, there is at most a proportion $\sum_{j=\ell+1}^{M} \hat{\varepsilon}^j$ of the data that remains to classify. This result will be illustrated in the next section through numerical experiments.

Algorithm 1 EERO method: Calibration phase and classification of a batch of data points

Require: Batch of data points: $\mathbf{X}_1, \dots, \mathbf{X}_T \in \mathcal{X}$;
Calibration sets $\mathcal{D}_N$; Model p with M heads,
Risk of heads: R, Allowed budget: B
Ensure: Prediction: $\mathcal{P} \in [K]^T$

```
F_s ← ComputeCDF(D_N)                                  {Eq. (5)}
ε̃_seq ← Aggregation(M, R, B)                           {Prop 3.8, Eq (8)-(9)}
P ← empty list
for i = 1 to T do
    for ℓ = 1 to M do
        s^ℓ, g^ℓ ← CompOutput(X_i, p_ℓ)                {Eq. (4)-(5)}
        if F_{s^ℓ}(s^ℓ(x)) ≥ 1 − ε̃^ℓ_seq then
            P ← [P, g^ℓ]
            break
        end if
    end for
end for
return P
```

4 EXPERIMENTS

Building upon the methodological foundations established for EERO, this section seeks to empirically substantiate our approach on ImageNet dataset with the ConvNext and MSDNet network architectures. We also compare our EERO approach with MSDNet strategy to compute the exit thresholds. Those experiments[2] validate our theory and demonstrate the scalability of EERO across diverse neural network configurations.

4.1 EERO WITH MULTIPLE HEADS

We recall that, after training the heads, the procedure consists of two steps. The first one focuses on determining the

[2]All computations are run on a server with an Intel(R) Xeon(R) Gold 5120 CPU and a Tesla V100 GPU with 32GB of Vram and 64GB of RAM. Code associated with paper can be found in repository here : `https://github.com/FlorianVal/Early-Exit-With-Reject-Option`

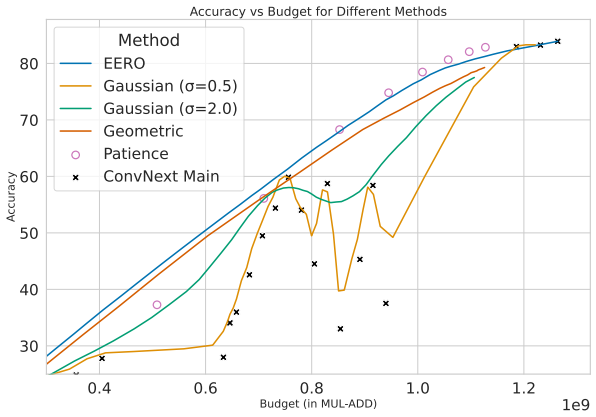


Figure 2: Accuracy *w.r.t.* the budget for our EERO methodology based on Convnext against other known methods such as Patience [Zhou et al., 2020, Zhang et al., 2022], Geometric distribution [Huang et al., 2018, Elbayad et al., 2020] or Gaussian [Li et al., 2022] distribution of weights on each head. ConvNext Main points correspond to the accuracy of each head alone.

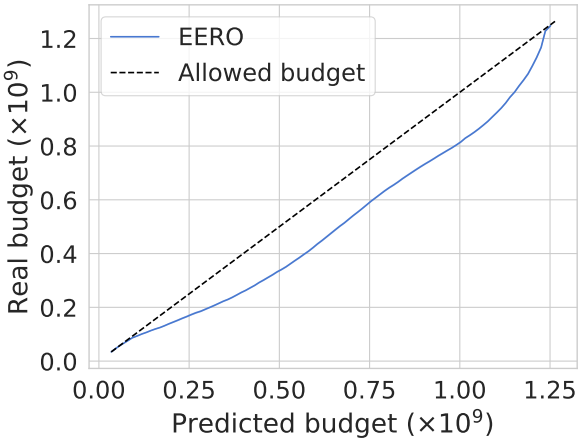


Figure 3: Measured and allowed budget of our algorithm on Convnext. This figure shows that our method accurately follows the budget given by never exceeding it.

rejection rates at each head. The second one consists in specifying the rejection rule (that is, calibrating the exit thresholds) at the level of each given head, *c.f.,* Definition 3.3.

In particular, we exploit the methodology based on aggregation with exponential weights developed in Section 3.2.2 to specify the rejection rates.

Following the protocol from Section 3.2, EERO is applied to the ConvNext architecture using the ImageNet dataset. The model is adapted with auxiliary heads – See Appendix A for details on extra computational cost induced by training the auxiliary heads. The training is based on pretrained weights and a sum of cross-entropy losses from various exits. For complete hyperparameter details, refer to Appendix Tables 1 and 2.

We plot the results for the different heads and for different computation budgets in Figure 2. Several observations can be made. First, we highlight an interesting phenomenon: for a given budget, our proposed multiple heads approach (Blue curve) improves the accuracy from the different exits used independently (black cross). Such observation confirms our statement in Theorem 3.9. Moreover, regarding the accuracy of the auxiliary heads alone, placing an exit further in the model does not necessarily result in a better accuracy when the model is not tuned for early exit.

We further check the budget used by our model in Figure 3 and show that our methodology respects the allowed budget by always being lower or equal to it, validating our theory from Proposition 3.5 and Theorem 3.12. On some budgets, the gap between the allowed budget and the actual consumption suggests that a better accuracy could be obtained. We also found that the rejection rates $\hat{\varepsilon}^\ell$ behave as expected as for example lower budgets favor early heads, and heads with higher risk $\hat{R}^\ell$ have lower values, thus decreasing their likelihood to be selected for the final classification score as depicted on Figure 4. (See also Appendix B).

4.2 COMPARISON WITH EXISTING EARLY EXIT STRATEGIES

We extend our evaluation by comparing EERO with existing early exit strategies, demonstrating its general applicability and effectiveness across different models and scenarios.

4.2.1 Methods Using specific distributions of Exits

Some early exit strategies employ predefined distributions, such as Gaussian [Li et al., 2022] or geometric [Huang et al., 2018, Elbayad et al., 2020] distributions, to determine the probability of exiting at each head. We show on Figure 2 that these methods lack awareness of the individual risks

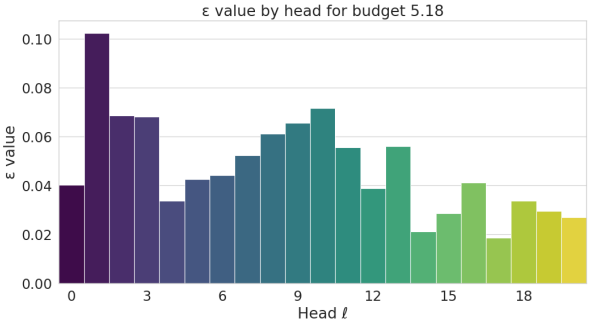


Figure 4: Value of the aggregation weights $\hat{\varepsilon}^\ell$ on an intermediate budget for the ConvNext model for EERO. Each bar represents an exit head ℓ.

associated with each head, which can significantly impact overall accuracy. While they may achieve comparable results on networks specifically tuned so that deeper exits yield better accuracy, they are less effective on general architectures. The lack of adaptation to the actual performance of each head can lead to suboptimal allocation of computational resources, potentially favoring less accurate exits. This is highlighted in Figure 2 where our method outperforms specific-distribution methods.

4.2.2 Patience-Based Strategies

Patience-based methods [Zhou et al., 2020, Zhang et al., 2022] exit after a certain number of consecutive confident predictions, leveraging the model's consistency across layers. While these strategies can outperform EERO in specific cases, they do not allow for explicit computational budget selection. Budget control is indirect and limited to discrete levels corresponding to different patience values, which restricts fine-grained resource allocation and may not align with strict budget constraints.

4.2.3 Comparison with MSDNet

In our experiments (see Appendix 5a, 5b), we compared EERO with the Multi-Scale Dense Network (MSDNet) [Huang et al., 2018], a well-known architecture specifically designed for early exiting that uses geometric distribution. The results show that EERO achieves similar accuracy and computational budget consumption as MSDNet (Figure 5a). However, there are key differences between the two approaches. MSDNet requires a specialized architecture and tuning to ensure that exits placed deeper in the network provide better accuracy. Moreover, MSDNet does not allow for the direct specification of a computational budget; instead, it requires iterative adjustments of exit probabilities to approximate the desired budget, which can be impractical. In contrast, EERO is model-agnostic and can be adapted to any network architecture with auxiliary heads. It allows for explicit selection of the computational budget beforehand, ensuring that the resource constraints are strictly met (Figure 5b). This flexibility makes EERO more suitable for applications where budget compliance is critical.

4.2.4 Advantages of EERO

EERO addresses the limitations of these methods by providing a general framework that is both model-agnostic and budget-aware. By incorporating the risks of each head through our aggregation method (Section 3.2.2), EERO optimally allocates computational resources to maximize accuracy under a given budget constraint. This ensures that more computational effort is devoted to heads with better performance, regardless of their position in the network. Our experimental results (Figure 2) demonstrate that EERO matches or outperforms the performance of specialized methods without requiring architecture-specific tuning. It provides precise budget adherence (Figure 3) and maintains competitive accuracy across various budgets and architectures.

5 CONCLUSION

In this paper, we presented EERO, a novel mathematical framework devised to construct classification rules for Early Exit in deep learning networks. This framework targets the pivotal challenge of optimizing computational efficiency while enhancing model performance. EERO approaches the problem by modeling it as a classification with a reject option, providing a feasible solution for scenarios involving a single Early Exit. For multi-exit scenarios, our innovative approach hinges on an aggregation with exponential weights in order to tune the exiting probabilities. A key strength of EERO lies in its flexibility and generalizability, enabling it to be applied across various model architectures. This adaptability makes EERO a versatile tool, well-suited for diverse applications in the realm of efficient deep learning.

Our experiments on ImageNet and CIFAR-100 using the Resnet-18, MSDNet and the ConvNext architectures respectively demonstrated the efficiency of our framework. The results not only shed light on the intricate workings of multi-head deep neural networks but also offer practical strategies to enhance model accuracy while reducing computational budget. This has important implications for edge computing and making the field of deep learning more sustainable.

ACKNOWLEDGEMENTS

This work was conducted as part of a CIFRE PhD funded by Fujitsu and the French National Association for Research and Technology (ANRT). We would like to thank Fujitsu for their support and collaboration throughout this research.

References

A. Bakhtiarnia, Q. Zhang, and A. Iosifidis. Multi-exit vision transformer for dynamic inference. *arXiv preprint arXiv:2106.15183*, 2021.

T. Bolukbasi, J. Wang, O. Dekel, and V. Saligrama. Adaptive neural networks for fast test-time prediction. *arXiv preprint arXiv:1702.07811*, 1(3), 2017.

N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

Mohammad-Amin Charusaie and Samira Samadi. A unifying post-processing framework for multi-objective learn-to-defer problems. In A. Globerson, L. Mackey, D. Bel-

grave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 23705–23755. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/2a5a41a536d3ada8fbf61a9d6fbf18d2-Paper-Conference.pdf.

X. Chen, H. Dai, Y. Li, X. Gao, and L. Song. Learning to stop while learning to predict. In *International Conference on Machine Learning*, pages 1520–1530. PMLR, 2020.

C. Chow. An optimum character recognition system using decision functions. *IRE Transactions on Electronic Computers*, EC-6(4):247–254, 1957.

C. Cortes, G. DeSalvo, and M. Mohri. Learning with rejection. In *International Conference on Algorithmic Learning Theory*, pages 67–82. Springer, 2016.

A. Dalalyan and A. B. Tsybakov. Aggregation by exponential weighting, sharp pac-bayesian bounds and sparsity. *Machine Learning*, 72(1–2):39–61, April 2008. ISSN 1573-0565. doi: 10.1007/s10994-008-5051-0. URL http://dx.doi.org/10.1007/s10994-008-5051-0.

C. Denis and M. Hebiri. Consistency of plug-in confidence sets for classification in semi-supervised learning. *Journal of Nonparametric Statistics*, 2019.

C. Denis, M. Hebiri, and A. Zaoui. Regression with reject option and application to knn. *arXiv preprint arXiv:2006.16597*, 2020.

C. Denis, M. Hebiri, B. Njike, and X. Siebert. Active learning algorithm through the lens of rejection arguments. *arXiv preprint arXiv: arXiv:2208.14682*, 2022.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer, 2020. URL https://arxiv.org/abs/1910.10073.

M. Figurnov, M. Collins, Y. Zhu, L. Zhang, J. Huang, D. Vetrov, and R. Salakhutdinov. Spatially adaptive computation time for residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1039–1048, 2017.

Y. Grandvalet, A. Rakotomamonjy, J. Keshet, and S. Canu. Support vector machines with a reject option. In *NIPS*, pages 537–544, 2009.

Neha Gupta, Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Aditya Krishna Menon, and Sanjiv Kumar. Language model cascades: Token-level uncertainty and beyond, 2024. URL https://arxiv.org/abs/2404.10136.

Y. Han, G. Huang, S. Song, L. Yang, H. Wang, and Y. Wang. Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

R. Herbei and M. Wegkamp. Classification with reject option. *Canad. J. Statist.*, 34(4):709–721, 2006.

G. Huang, D. Chen, T. Li, F. Wu, L. Van Der Maaten, and K. Weinberger. Multi-scale dense networks for resource efficient image classification. In *6th International Conference on Learning Representations, ICLR 2018*. OpenReview.net, 2018.

Wittawat Jitkrittum, Neha Gupta, Aditya Krishna Menon, Harikrishna Narasimhan, Ankit Singh Rawat, and Sanjiv Kumar. When does confidence-based cascade deferral suffice?, 2024. URL https://arxiv.org/abs/2307.02764.

Y. Kaya, S. Hong, and T. Dumitras. Shallow-deep networks: Understanding and mitigating network overthinking. In *International conference on machine learning*, pages 3301–3310. PMLR, 2019.

A. Krizhevsky, V. Nair, and G. Hinton. Cifar-100. *(Canadian Institute for Advanced Research)*, 2009.

S. Laskaridis, A. Kouris, and N. Lane. Adaptive inference through early-exit networks: Design, challenges and directions. In *Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning*, pages 1–6, 2021.

J. Lei. Classification with confidence. *Biometrika*, 101(4):755–769, 2014.

Xiangjie Li, Chenfei Lou, Zhengping Zhu, Yuchi Chen, Yingtao Shen, Yehan Ma, and An Zou. Predictive exit: Prediction of fine-grained early exits for computation- and energy-efficient inference, 2022. URL https://arxiv.org/abs/2206.04685.

T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403, 2021.

S. Lin, B. Ji, R. Ji, and A. Yao. A closer look at branch classifiers of multi-exit architectures. *arXiv preprint arXiv:2204.13347*, 2022.

Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022.

T. Luo, K. Kramer, D. Goldgof, L. Hall, S. Samson, A. Remsen, T. Hopkins, and D. Cohn. Active learning to recognize multiple types of plankton. *Journal of Machine Learning Research*, 6(4), 2005.

M. Naadeem, J.D. Zucker, and B. Hanczar. Accuracy-rejection curves (ARCs) for comparing classification methods with a reject option. In *MLSB*, pages 65–81, 2010.

Nastaran Okati, Abir De, and Manuel Gomez-Rodriguez. Differentiable learning under triage, 2021. URL `https://arxiv.org/abs/2103.08902`.

P. Rigollet and A. B. Tsybakov. Sparse Estimation by Exponential Weighting. *Statistical Science*, 27(4):558 – 575, 2012.

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

H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021.

V. Vovk, A. Gammerman, and C. Saunders. Machine-learning applications of algorithmic randomness. In *In Proceedings of the Sixteenth International Conference on Machine Learning*, pages 444–453. Morgan Kaufmann, 1999.

V. Vovk, A. Gammerman, and G. Shafer. *Algorithmic learning in a random world*. Springer, New York, 2005.

X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 409–424, 2018.

Y. Wang, R. Huang, S. Song, Z. Huang, and G. Huang. Not all images are worth 16x16 words: Dynamic transformers for efficient image recognition. *Advances in Neural Information Processing Systems*, 34:11960–11973, 2021.

Z. Wu, T. Nagarajan, A. Kumar, S. Rennie, L. Davis, K. Grauman, and R. Feris. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8817–8826, 2018.

M. Yuan and M. Wegkamp. Classification methods with reject option based on convex risk minimization. *J. Mach. Learn. Res.*, 11:111–130, 2010.

Zhen Zhang, Wei Zhu, Jinfan Zhang, Peng Wang, Rize Jin, and Tae-Sun Chung. PCEE-BERT: Accelerating BERT inference via patient and confident early exiting. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 327–338, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-naacl.25. URL `https://aclanthology.org/2022.findings-naacl.25`.

W. Zhou, C. Xu, T. Ge, J. McAuley, K. Xu, and F. Wei. Bert loses patience: Fast and robust inference with early exit. In *Advances in Neural Information Processing Systems*, volume 33, pages 18330–18341, 2020.

Supplementary Material

Florian Valade[1,2] **Mohamed Hebiri**[1] **Paul Gay**[3]

[1]LAMA , Université Gustave Eiffel
[2]Fujitsu
[3]Université De Pau Et Des Pays De L'adour

A IMPLEMENTATION DETAILS OF EXPERIMENTS

In Tables 1 and 2, we provide the key experimental details for our implementations, including model architectures, datasets, and training parameters. Our implementation is done by freezing the pretrained model and only training the added auxiliary heads. Involving the extra head in our process increases the computational cost by less than 4% and 2.5% in the case of ResNet-18 and ConvNext respectively. In all our experiments we set $\beta = 0.04$ and we mention that our method is not significantly affected by changes of this parameter ; notice that our theory suggests a value of order $1/\sqrt{N}$.

Table 1: Basic Setup of Experiments

Experiment	**Model**	**Dataset**	**Key Modifications**
ResNet with CIFAR-100	ResNet	CIFAR-100	Auxiliary heads at 7 positions, Flatten outputs
ConvNext with ImageNet	ConvNext	ImageNet	Auxiliary heads at 19 positions, Adaptive Average pooling, Flatten outputs

Table 2: Detailed Training Parameters

Experiment	**Training Details**	**Calibration Details**
ResNet with CIFAR-100	SGD, LR: 0.003, Batch size: 128 Train data : 49.000	Calibration set: 1,000 examples
ConvNext with ImageNet	Pretrained weights, Fine-tuned for 300 epochs, SGD, Momentum: 0.9, Weight decay: 10^{-4}, Train data : Full Imagenet	Calibration set: 5,000 examples

B ANALYSIS OF AGGREGATION WEIGHTS

We examine the values of the obtained thresholds $\hat{\varepsilon}$ as it offers interesting insights. We first display the values for all the layers and for different budgets in Table 3. The results highlights that: i) when higher budget are allocated, the weights produced by the aggregation put higher weights on heads with lower risks; ii) heads with high risks are assigned smaller weights by the aggregation, see for instance the weights corresponding to the heads 13 and 18 that have a higher risk and so a lower value for the ε^{ℓ} threshold.

This is also visible in Figure 6 where we show the $\hat{\varepsilon}$ values for our EERO method together with the MSDNet like strategy.

First line shows a budget that does not allow usage of the computationally costly heads and so most data will be output on the two first heads. Second line corresponds a medium budget, where we can see that our EERO approach and the MSDNet

			$6.55 * 10^{12}$ Flops		$3.21 * 10^{13}$ Flops		$5.64 * 10^{13}$ Flops	
ℓ	$\hat{R}^\ell$	π^ℓ	$\hat{\varepsilon}^\ell$	$\hat{\varepsilon}^\ell_{\text{out}}$	$\hat{\varepsilon}^\ell$	$\hat{\varepsilon}^\ell_{\text{out}}$	$\hat{\varepsilon}^\ell$	$\hat{\varepsilon}^\ell_{\text{out}}$
1	0.93	0.46	0.69	0.68	0.18	0.17	0.00	0.00
2	0.81	0.13	0.17	0.28	0.07	0.19	0.00	0.01
3	0.75	0.04	0.03	0.04	0.03	0.16	0.00	0.01
4	0.72	0.04	0.03	0.00	0.03	0.12	0.00	0.01
5	0.72	0.02	0.01	0.00	0.03	0.08	0.00	0.02
6	0.66	0.02	0.01	0.00	0.03	0.07	0.00	0.01
7	0.64	0.02	0.01	0.00	0.03	0.05	0.01	0.02
8	0.57	0.02	0.01	0.00	0.03	0.04	0.01	0.02
9	0.51	0.02	0.01	0.00	0.04	0.03	0.01	0.02
10	0.46	0.02	0.01	0.00	0.04	0.02	0.01	0.01
11	0.40	0.02	0.01	0.00	0.04	0.02	0.01	0.02
12	0.46	0.02	0.01	0.00	0.04	0.03	0.01	0.04
13	0.55	0.02	0.00	0.00	0.03	0.02	0.01	0.03
14	0.41	0.02	0.01	0.00	0.04	0.01	0.02	0.03
15	0.67	0.02	0.00	0.00	0.03	0.01	0.02	0.08
16	0.55	0.02	0.00	0.00	0.04	0.00	0.02	0.15
17	0.42	0.02	0.00	0.00	0.04	0.00	0.03	0.09
18	0.62	0.02	0.00	0.00	0.03	0.00	0.03	0.13
19	0.17	0.01	0.00	0.00	0.06	0.00	0.20	0.05
20	0.17	0.01	0.00	0.00	0.06	0.00	0.27	0.06
21	0.16	0.01	0.00	0.00	0.06	0.00	0.33	0.20

Table 3: Illustration of the evolution of the probability vector of classification for the different heads on ImageNET dataset with Convnext architecture and for three different budgets: each line represents an exit head, $\boldsymbol{\pi}$ is the prior distribution, $\hat{\boldsymbol{\varepsilon}}$ are the weights produced by the aggregation, and $\hat{\boldsymbol{\varepsilon}}_{\text{out}}$ are the proportion of the data that was actually classified by each head.

strategy show different behaviors. Whereas the latter assumes that a sample that *reaches* a head classifier has the same constant probability to exit independently from its layer position, resulting in a monotonic behavior, our approach exploits the information of the empirical risks $\hat{R}^\ell$ (see Equation (6)) to modulate the importance of the heads. As we saw in Figure 6 and 7, a high risk is correlated with a low epsilon. Third line shows a permissive budget where most data will be output at the last heads as they have more accuracy.

C EERO WITH ONLY 1 AUXILIARY HEAD

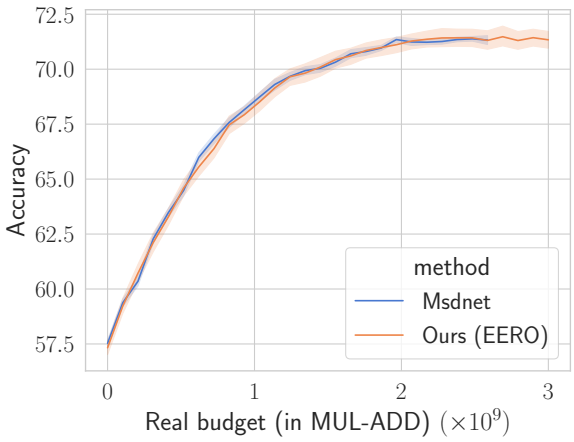
In the case of one auxiliary head, computations of the rate of classifying $\hat{\varepsilon}^1$ (of this single head) can be simplified. In particular, we can express it analytically without the need for aggregation with exponential weights. Let $B > 0$ be the amount of *GFlops* we were allocated for the task of labeling T data points. Assume that the classifiers $\hat{h}^1_{\hat{\varepsilon}^1}$ and $\hat{g}^2$ burn off respectively $\hat{B}^1$ and $\hat{B}^2$ *GFlops* at each call with $\hat{B}^1 < \hat{B}^2$. (We assume that $B \geq T\hat{B}^1$ so that we are guaranteed to label all T points).

Proposition C.1. *Let* $\hat{\varepsilon}^1 = \frac{\frac{B}{T} - \hat{B}^2}{\hat{B}^1 - \hat{B}^2}$. *Then the total amount of* GFlops *used to label* T *instances is not larger than* B.

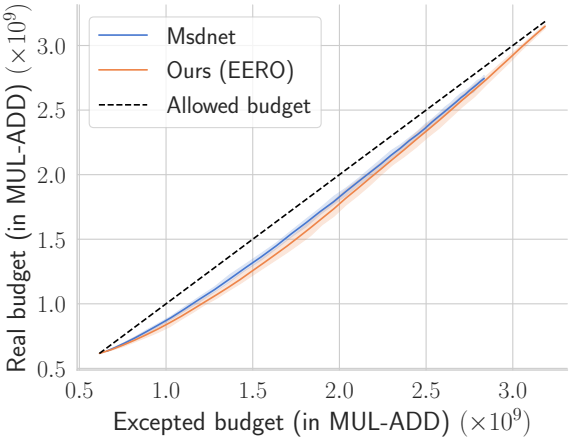
This result highlights that our strategy succeeds to comply with the constraint of budget we imposed. The proof of the result lies in the fact that we use the early exit on a proportion $\hat{\varepsilon}^1$ of the data. Then we consume $T\hat{\varepsilon}^1\hat{B}^1$ *GFlops* (up to a $1/\sqrt{N}$ additive term – see the comment after Proposition 3.5 for a correction). The rest of the data is treated by the classifier $\hat{g}^2$ and then uses $T(1-\hat{\varepsilon}^1)\hat{B}^2$ *GFlops*. In total, we then get

$$T(\hat{\varepsilon}^1\hat{B}^1 + (1-\hat{\varepsilon}^1)\hat{B}^2) = B \ . \tag{10}$$

Remark C.2. In this section, we enforce the rejection rate to exactly burn off the whole budget B – see Equation (10). We make this choice to explain some broader effects that illustrate the importance of using auxiliary heads (see below and



(a) Accuracy *w.r.t.* the real budget based on MSDNet architecture – comparison of EERO methodology to the original MSDNet algorithm.



(b) Measured budget on MSDNet architecture – comparison of EERO methodology to the original MSDNet algorithm.

Figure 5: Subfigures (a) and (b) compare EERO with MSDNet's original method, focusing on accuracy and budget metrics. Error bars were made using bootstrap on data.

Figure 8). On the other hand, we notify that it is extremely simple to modify the algorithm so that it consumes less than or equal to the total budget, that is, $(\hat{\varepsilon}^1 T)\hat{B}^1 + ((1-\hat{\varepsilon}^1)T)\hat{B}^2 \leq B$. In this case, the accuracy curve would always be increasing *w.r.t.* the budget.

We implement EERO using a ResNet [He et al., 2016] model on the CIFAR-100 [Krizhevsky et al., 2009] dataset, with auxiliary heads placed at seven different positions for early exit testing (see Figure 1). The model training involve a sum of cross-entropy losses from these exits. Detailed hyperparameters of this experiment are provided in Appendix A in Tables 1 and 2.

We plot the accuracy for the different exits in Figure 8 for different budgets. First, we can see a general trend where our procedure gradually improves the accuracy as the budget increases since images wrongly classified by the auxiliary head and correctly classified by the final layer are forwarded from the Early Exit to the latter. Then there is an accuracy decrease since augmenting the budget only affects images which are wrongly classified by both Early Exit and final model exit. Secondly, we notice that the oracle quickly outperforms the accuracy of the main model for only a fraction of the total budget required to process all images with the full model. This fact shows that Early Exits can have valid predictions, whereas the last model output is wrong. This phenomenon is known as over-thinking and motivates the potential of Early Exit methods, as a way of better understanding the flaws of deep learning models. Here, this over-thinking situation seems important, as some of the Early Exits have a better accuracy than the last model output, *c.f.,* Remark C.2. A second reason explaining the oracle improvements is its capacity to save computation budget by choosing an Early Exit when both early and last exit predictions are wrong.

This advantage, only achievable in our specific context of Budgeted Batch Classification, aligns well with our primary objective of reducing computational power for our algorithms. Furthermore, it indirectly enhances the overall accuracy of our model.

D PROOFS

In this section, we provide the proofs of our main results.

Proof 1 (Proposition 3.2)**.** *Let us write*

$$\mathcal{R}(h) = \mathbb{P}\left(h(\mathbf{X}) \neq Y \ , \ h(\mathbf{X}) \neq \mathfrak{R}\right) \ ,$$

for short. We then need to solve the problem

$$h^*_\varepsilon = \underset{h}{\operatorname{argmin}} \ \{\mathcal{R}(h) : \mathbb{P}(h(\mathbf{X}) = \mathfrak{R}) = 1 - \varepsilon\} \, ,$$

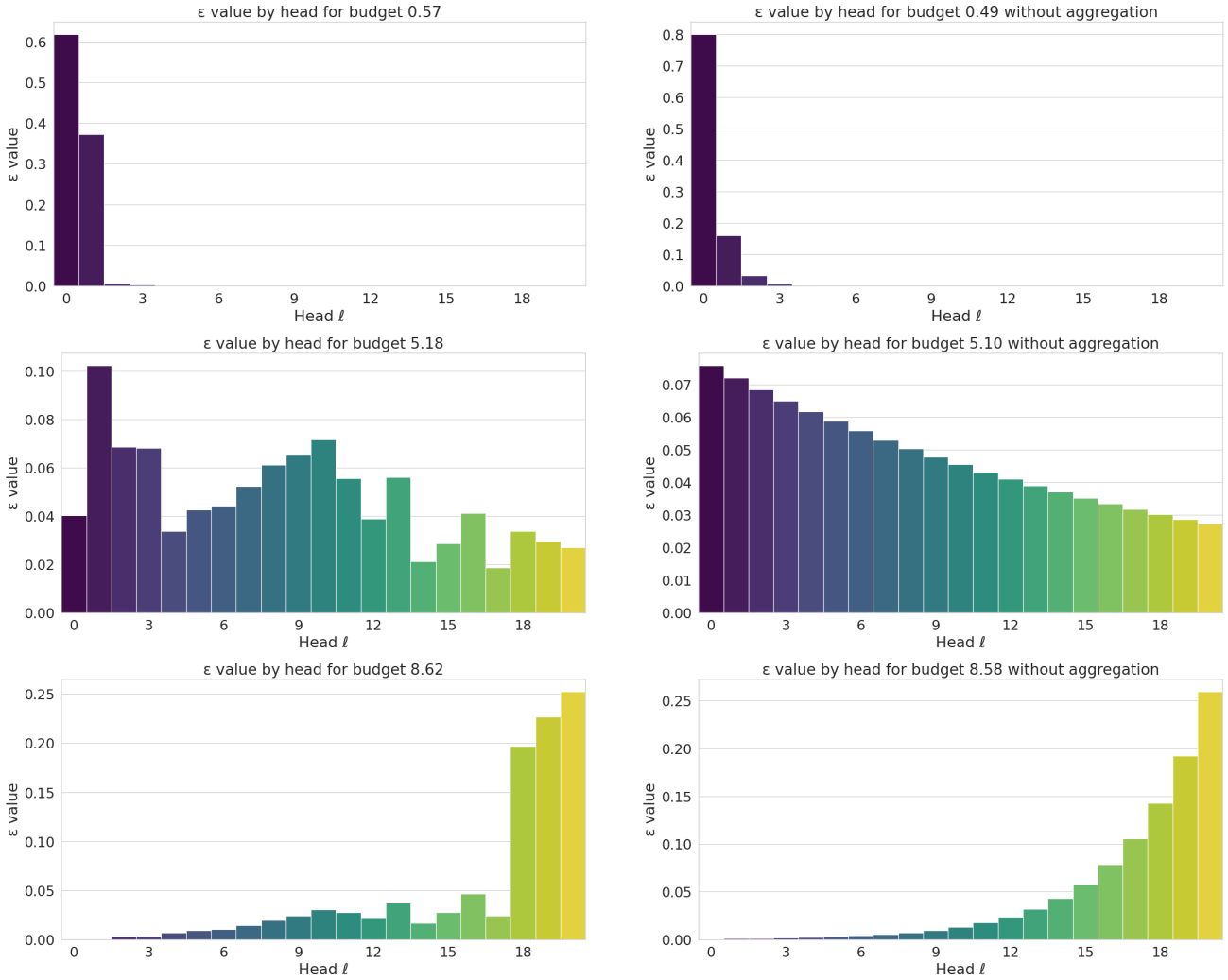


Figure 6: Value of the aggregation weights $\hat{\varepsilon}^{\ell}$ on three different budgets for the ConvNext model for EERO (left column) and with the MSDNet strategy (right column). Each bar represents an exit head ℓ. The lines corresponds to different budgets ranging from small (top), to medium (centre) and high (top).

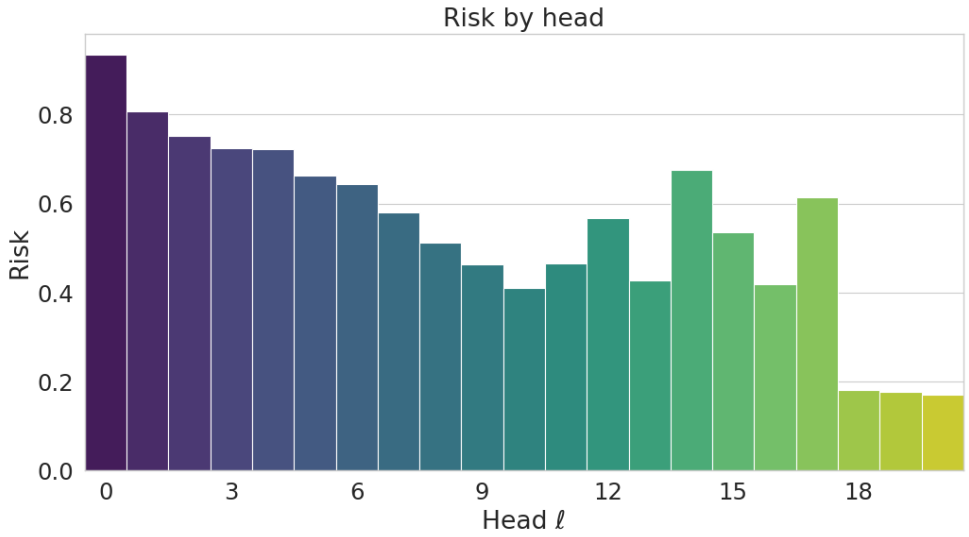


Figure 7: Risk $\hat{R}^{\ell}$ of each head on Convnext model

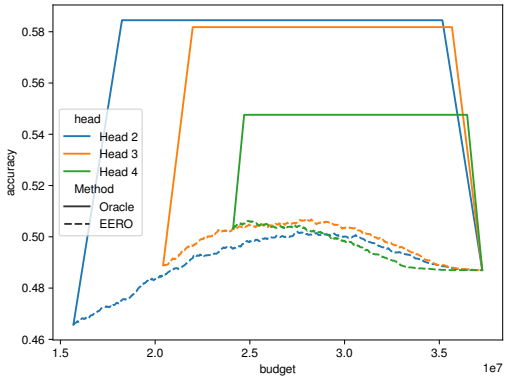


Figure 8: Accuracy versus computation budget for our multi-headed ResNet-18 (given in Figure 1). In this illustration, we consider only a two heads procedure by selecting a specific auxiliary head – 2, 3, or 4 – among the initial 7. Each curve shows how the accuracy evolves according to the probability of use of the auxiliary head. For each color, we display our algorithm as well as the oracle.

where the minimum is taken over all measurable functions. Considering the Lagrangian of the above problem, we solve

$$\min_h \max_{\lambda\in\mathbb{R}^+} \underbrace{\{\mathcal{R}(h) + \lambda\left(\mathbb{P}(h(\mathbf{X}) = \mathfrak{R}) - (1-\varepsilon)\right)\}}_{:=\mathcal{L}(h,\lambda)} \ ,$$

where $\lambda \in \mathbb{R}$ is the dual variable. Observe that by weak duality we have

$$\min_h \max_{\lambda\in\mathbb{R}^+} \mathcal{L}(h,\lambda) \geq \max_{\lambda\in\mathbb{R}^+} \min_h \mathcal{L}(h,\lambda) \ ,$$

we then consider first the minimization problem over h of $\mathcal{R}$. We have

$$\begin{aligned}\mathcal{R}(h) = \mathbb{P}\left(h(\mathbf{X}) \neq Y \ , \ h(\mathbf{X}) \neq \mathfrak{R}\right) &= \mathbb{E}\left[\sum_{k\in[K]} \mathbb{1}_{\{Y=k\}}\mathbb{1}_{\{h(\mathbf{X})\neq k\}}\mathbb{1}_{\{h(\mathbf{X})\neq\mathfrak{R}\}}\right] \\ &= \mathbb{E}\left[\sum_{k\in[K]} p_k(\mathbf{X})\mathbb{1}_{\{h(\mathbf{X})\neq k\}}\mathbb{1}_{\{h(\mathbf{X})\neq\mathfrak{R}\}}\right] = \mathbb{P}\left(h(\mathbf{X}) \neq \mathfrak{R}\right) - \mathbb{E}\left[\sum_{k\in[K]} p_k(\mathbf{X})\mathbb{1}_{\{h(\mathbf{X})=k\}}\mathbb{1}_{\{h(\mathbf{X})\neq\mathfrak{R}\}}\right]\end{aligned}$$

Moreover,

$$\mathbb{P}(h(\mathbf{X}) \neq \mathfrak{R}) = \mathbb{E}\left[\sum_{k\in[K]} \mathbb{1}_{\{h(\mathbf{X})=k\}}\mathbb{1}_{\{h(\mathbf{X})\neq\mathfrak{R}\}}\right].$$

Therefore, we can write

$$\begin{aligned}\mathcal{L}(h,\lambda) &= \lambda - \lambda(1-\varepsilon) + (1-\lambda)\mathbb{P}(h(\mathbf{X}) \neq \mathfrak{R}) - \mathbb{E}\left[\sum_{k\in[K]} p_k(\mathbf{X})\mathbb{1}_{\{h(\mathbf{X})=k\}}\mathbb{1}_{\{h(\mathbf{X})\neq\mathfrak{R}\}}\right] \quad (11)\\ &= \lambda\varepsilon - \mathbb{E}\left[\sum_{k\in[K]} \left(p_k(\mathbf{X}) - (1-\lambda)\right)\mathbb{1}_{\{h(\mathbf{X})=k\}}\mathbb{1}_{\{h(\mathbf{X})\neq\mathfrak{R}\}}\right] \ . \quad (12)\end{aligned}$$

We need to maximize the expectation w.r.t. *h that first leads to the optimum h^*_λ is such that*

$$h^*_\lambda(\mathbf{x}) \neq \mathfrak{R} \iff \sum_{k\in[K]} \left(p_k(\mathbf{x}) - (1-\lambda)\right)\mathbb{1}_{\left\{h^*_\lambda(\mathbf{x})=k\right\}} > 0 \ , \quad (13)$$

for all $\mathbf{x} \in \mathcal{X}$. Moreover we have that on the event $\{h^*_\lambda(\mathbf{x}) \neq \mathfrak{R}\}$, the mapping h that maximizes $h \mapsto \sum_{k\in[K]} \left(p_k(\mathbf{x}) - (1-\lambda)\right)\mathbb{1}_{\{h(\mathbf{x})=k\}}$ is simply $h^*_\lambda(\mathbf{x}) = \operatorname{argmax}_{k\in[K]} p_k(\mathbf{x})$. At this level of the proof, we have shown that the problem $\min_h \mathcal{L}(h,\lambda)$ leads to the rule

$$h^*_\lambda(\mathbf{x}) = \begin{cases} \operatorname{argmax}_{k\in[K]} p_k(\mathbf{x}) & \text{if} \quad \max_{k\in[K]} p_k(\mathbf{x}) \geq 1-\lambda \\ \mathfrak{R} & \text{otherwise} \ . \end{cases} \quad (14)$$

Now, we deal with the maximization of $\mathcal{L}(h^*_\lambda,\lambda)$ w.r.t. *λ. Substituting the above value of $h = h^*_\lambda$ in* (11)*, we can show that*

$$\mathcal{L}(h^*_\lambda,\lambda) = \lambda\varepsilon - \mathbb{E}\left[\left(\max_{k\in[K]}\left\{p_k(\mathbf{X}) - (1-\lambda)\right\}\right)_+\right] \ ,$$

where for all $a \in \mathbb{R}$, we write $(a)_+ = \max\{a, 0\}$. The above function is then concave in λ, therefore we can write the first order optimality condition as $0 \in \partial\mathcal{L}(h^*_{\lambda^*},\lambda^*)$. Observe that because we assumed that the r.v. $p_k(X)$ has no atom for all $\in [K]$, we have that $\mathbb{P}\left(\exists\, j \in [K] : p_k(X) = p_j(X)\right) = 0$ and then the subgradient reduces to the gradient. As a consequence, we obtain the following condition on λ^*

$$\mathbb{P}\left(\exists k \in [K] : p_k(\mathbf{X}) > \max\left\{\max_{j\in[K]}\left\{p_j(\mathbf{X})\right\}; 1-\lambda^*\right\}\right) = \varepsilon \ .$$

Notice that this last condition can rewrite as

$$\varepsilon = \mathbb{P}\left(s(\mathbf{X}) > 1 - \lambda^*\right) = \mathbb{P}\left(h^*_{\lambda^*}(\mathbf{X}) \neq \mathfrak{R}\right) \ ,$$

where $s(\mathbf{X}) = \max_{k\in[K]} p_k(\mathbf{X})$*, which guarantee that the optimal rule has indeed the correct rejection rate. Moreover, from the above relation, we can exhibit the value of* λ*. Indeed, using the continuity condition on the CDF* F_s *of* $s(\mathbf{X})$*, we have*

$$\mathbb{P}\left(s(\mathbf{X}) > 1 - \lambda^*\right) = \varepsilon \quad \Longleftrightarrow \quad 1 - F_s\left(1 - \lambda^*\right) = \varepsilon \quad \Longleftrightarrow \quad \lambda^* = 1 - F_s^{-1}\left(1 - \varepsilon\right),$$

where F_s^{-1} *is the generalized inverse of* F_s*. We conclude the proof substituting this value into the expression of the optimal rule given by* (14)*.*

Remark D.1. If we replace the equality by an inequality, everything is the same expect the part of λ^*. We need to consider the case where $\lambda^* = 0$ separately, and in this case, we would get $\mathbb{P}\left(h^*_{\lambda^*}(\mathbf{X}) \neq \mathfrak{R}\right) \geq \varepsilon$.

Proof 2 (Proposition 3.5)**.** *First, observe that conditionally to the training dataset* $\mathcal{D}_n$ *and due to the continuity of the CDF of* $\hat{s}^\ell(\mathbf{X})$*, the random variable* $F_{\hat{s}^\ell}(\hat{s}^\ell(\mathbf{X}))$ *is uniformly distributed. Therefore, for any* $u \in [0,1]$*, we have* $\mathbb{P}\left(F_{\hat{s}^\ell}(\hat{s}^\ell(\mathbf{X})) \leq u\right) = u$*. We then can write*

$$\begin{aligned}
\left|\mathbb{P}\left(\hat{h}^\ell_{\varepsilon^\ell}(\mathbf{X}) = \mathfrak{R}\right) - \left(1 - \varepsilon^\ell\right)\right| &= \left|\mathbb{E}\left[\mathbb{1}_{\left\{\hat{F}_{\hat{s}^\ell}(\hat{s}^\ell(\mathbf{X})) \leq 1-\varepsilon^\ell\right\}} - \mathbb{1}_{\left\{F_{\hat{s}^\ell}(\hat{s}^\ell(\mathbf{X})) \leq 1-\varepsilon^\ell\right\}}\right]\right| \\
&\leq \left|\mathbb{E}\left[\mathbb{1}_{\left\{\left|F_{\hat{s}^\ell}(\hat{s}^\ell(\mathbf{X})) - (1-\varepsilon^\ell)\right| \leq \left|\hat{F}_{\hat{s}^\ell}(\hat{s}^\ell(\mathbf{X})) - F_{\hat{s}^\ell}(\hat{s}^\ell(\mathbf{X}))\right|\right\}}\right]\right| \\
&\leq \left|\mathbb{E}\left[\mathbb{1}_{\left\{\left|F_{\hat{s}^\ell}(\hat{s}^\ell(\mathbf{X})) - (1-\varepsilon^\ell)\right| \leq \left\|\hat{F}_{\hat{s}^\ell} - F_{\hat{s}^\ell}\right\|_\infty\right\}}\right]\right| \leq 2\mathbb{E}\left\|\hat{F}_{\hat{s}^\ell} - F_{\hat{s}^\ell}\right\|_\infty \ ,
\end{aligned}$$

where we used again in the last line the fact that $F_{\hat{s}^\ell}(\hat{s}^\ell(\mathbf{X}))$ *is uniformly distributed. Moreover, we wrote* $\|\hat{F}_{\hat{s}^\ell} - F_{\hat{s}^\ell}\|_\infty = \sup_{t\in\mathbb{R}} |\hat{F}_{\hat{s}^\ell}(t) - F_{\hat{s}^\ell}(t)|$*. We conclude the proof using the Dvoretzky–Kiefer–Wolfowitz inequality, that states that*

$$\mathbb{E}\left\|\hat{F}_{\hat{s}^\ell} - F_{\hat{s}^\ell}\right\|_\infty \leq \sqrt{\frac{\pi}{2N}} \ .$$

Proof 3 (Proposition 3.8)**.** *The Lagrangian of the minimization problem* (6)-(7) *is given by*

$$\mathcal{L}(\boldsymbol{\varepsilon}, \lambda, \mu) = \sum_{j=1}^M \varepsilon^j \hat{R}^j + \beta \sum_{j=1}^M \varepsilon^j \log\left(\frac{\varepsilon^j}{\pi^j}\right) + \lambda\left(\sum_{j=1}^M \varepsilon^j - 1\right) + \mu\left(\sum_{j=1}^M \varepsilon^j \hat{B}^j - \bar{B}\right) \ ,$$

with $(\lambda, \mu) \in \mathbb{R} \times \mathbb{R}^+$*. Considering the KKT condition of the problem we get for all* $j \in [M]$

$$\hat{R}^j + \beta\left(\log\left(\frac{\hat{\varepsilon}^j}{\pi^j}\right) + 1\right) + \hat{\lambda} + \hat{\mu}\hat{B}^j = 0,$$

that leads in turns to

$$\hat{\varepsilon}^j_{\hat{\lambda},\hat{\mu}} = \pi^j \exp\left(-\frac{\hat{R}^j + \hat{\lambda} + \hat{\mu}\hat{B}^j}{\beta} - 1\right) \ . \tag{15}$$

Plug-in these values into the equality constraint leads to

$$\begin{aligned}
\sum_{j=1}^M \hat{\varepsilon}^j_{\hat{\lambda},\hat{\mu}} = 1 \quad &\Longleftrightarrow \quad \sum_{j=1}^M \pi^j \exp\left(-\frac{\hat{R}^j + \mu\hat{B}^j}{\beta} - 1\right) = \exp\left(\frac{\hat{\lambda}}{\beta}\right) \\
&\Longleftrightarrow \quad \hat{\lambda} = \beta \log\left(\sum_{j=1}^M \pi^j \exp\left(-\frac{\hat{R}^j + \mu\hat{B}^j}{\beta} - 1\right)\right) \ .
\end{aligned}$$

Substituting back this value into (15)*, we get*

$$\hat{\varepsilon}^j_{\hat{\mu}} = \frac{\pi^j \exp\left(-\frac{\hat{R}^j + \hat{\mu}\hat{B}^j}{\beta}\right)}{\sum_{k=1}^M \pi^k \exp\left(-\frac{\hat{R}^k + \hat{\mu}\hat{B}^k}{\beta}\right)} \ .$$

According to the parameter $\hat{\mu}$, we need to consider the constraints $\hat{\mu} \geq 0$ and $\sum_{j=1}^{M} \hat{\varepsilon}_{\hat{\mu}}^{j} \hat{B}^{j} \leq \bar{B}$ together with the complementary condition $\hat{\mu}\left(\sum_{j=1}^{M} \hat{\varepsilon}_{\hat{\mu}}^{j}(\hat{B}^{j} - \bar{B})\right) = 0$. Therefore, when $\hat{\mu} \neq 0$, this parameter should be taken such that

$$\sum_{j=1}^{M} \hat{\varepsilon}_{\hat{\mu}}^{j}(\hat{B}^{j} - \bar{B}) = 0 \quad \iff \quad \sum_{j=1}^{M} \pi^{j}(\hat{B}^{j} - \bar{B}) \exp\left(-\frac{\hat{R}^{j} + \hat{\mu}\hat{B}^{j}}{\beta}\right) = 0 \ .$$

Otherwise $\hat{\mu} = 0$ and in this case, $\hat{\varepsilon}_{\hat{\mu}}^{j}$ becomes

$$\hat{\varepsilon}_{\hat{\mu}}^{j} = \frac{\pi^{j} \exp\left(-\hat{R}^{j}/\beta\right)}{\sum_{k=1}^{M} \pi^{k} \exp\left(-\hat{R}^{k}/\beta\right)} \ .$$

Proof 4 (Theorem 3.9)**.** *Let us first give a formal definition of the resulting classifier $\hat{g}^{EERO}$ from our procedure. Introduce for all $\mathbf{x} \in \mathcal{X}$, the index $\bar{\ell}(\mathbf{x}) = \min\left\{\ell \in [M] : \ \hat{F}_{\hat{s}^{\ell}}(\hat{s}^{\ell}(\mathbf{x})) \geq 1 - \hat{\varepsilon}^{\ell}\right\}$ as the first moment where the we do not reject. Then, the classification by EERO is exactly provided by the head that corresponds to $\bar{\ell}$, that is*

$$\hat{g}^{EERO}(\mathbf{x}) = \hat{g}^{\bar{\ell}(\mathbf{x})}(\mathbf{x}) \ . \tag{16}$$

The goal now is here to demonstrate that the choice of $\hat{\boldsymbol{\varepsilon}}$ provides a good compromise in terms of risk and computations budget. Recall that for all $\ell \in [M]$, we have

$$\hat{\varepsilon}^{\ell} = \frac{\pi^{\ell} \exp\left\{-\frac{\hat{R}^{\ell} + \hat{\mu}\hat{B}^{\ell}}{\beta}\right\}}{\sum_{j=1}^{M} \pi^{j} \exp\left\{-\frac{\hat{R}^{j} + \hat{\mu}\hat{B}^{j}}{\beta}\right\}} = c \cdot \pi^{\ell} \exp\left\{-\frac{\hat{R}^{\ell} + \hat{\mu}\hat{B}^{\ell}}{\beta}\right\} \ ,$$

where $c = \sum_{j=1}^{M} \pi^{j} \exp\left\{-\frac{\hat{R}^{j} + \hat{\mu}\hat{B}^{j}}{\beta}\right\}$ is the normalization constant. Then

$$\log\left(\frac{\hat{\varepsilon}^{\ell}}{\pi^{\ell}}\right) = \log(c) - \frac{\hat{R}^{\ell} + \hat{\mu}\hat{B}^{\ell}}{\beta} \ .$$

Rearranging terms we get

$$\hat{R}^{\ell} = \beta \log(c) - \beta \log\left(\frac{\hat{\varepsilon}^{\ell}}{\pi^{\ell}}\right) - \hat{\mu}\hat{B}^{\ell} \ .$$

Notice that the same relation holds true for the optimal head ℓ^* – the feasible head that minimizes the risk. Therefore, subtracting the two equality gives

$$\hat{R}^{\ell} = \hat{R}^{\ell^*} + \beta\left(\log\left(\frac{\hat{\varepsilon}^{\ell^*}}{\pi^{\ell^*}}\right) - \log\left(\frac{\hat{\varepsilon}^{\ell}}{\pi^{\ell}}\right)\right) + \hat{\mu}\left(\hat{B}^{\ell^*} - \hat{B}^{\ell}\right) \ .$$

Multiplying by $\hat{\varepsilon}^{\ell}$ and summing out over ℓ we get

$$\frac{1}{n_2} \sum_{(\mathbf{X},Y) \in \mathcal{D}_{n_2}} \mathbb{1}_{\{\hat{g}^{EERO}(\mathbf{X}) \neq Y\}} = \sum_{\ell=1}^{M} \hat{\varepsilon}^{\ell} \hat{R}^{\ell} = \hat{R}^{\ell^*} + \beta\left(\log\left(\frac{\hat{\varepsilon}^{\ell^*}}{\pi^{\ell^*}}\right) - \sum_{\ell=1}^{M} \hat{\varepsilon}^{\ell} \log\left(\frac{\hat{\varepsilon}^{\ell}}{\pi^{\ell}}\right)\right) + \hat{\mu}\left(\hat{B}^{\ell^*} - \sum_{\ell=1}^{M} \hat{\varepsilon}^{\ell}\hat{B}^{\ell}\right) \ . \tag{17}$$

Let us first consider the term $\hat{\mu}\left(\hat{B}^{\ell^*} - \sum_{\ell=1}^{M} \hat{\varepsilon}^{\ell}\hat{B}^{\ell}\right)$. Recalling the previously stated complementary condition $\hat{\mu}\left(\sum_{\ell=1}^{M} \hat{\varepsilon}^{\ell}(\hat{B}^{\ell} - \bar{B})\right) = 0$, we either have $\hat{\mu} = 0$ or $\sum_{\ell=1}^{M} \hat{\varepsilon}^{\ell}\hat{B}^{\ell} = \bar{B}$. In the latter case, the term

$$\hat{B}^{\ell^*} - \sum_{\ell=1}^{M} \hat{\varepsilon}^{\ell}\hat{B}^{\ell} = \hat{B}^{\ell^*} - \bar{B} \leq 0 \ ,$$

since $\hat{B}^{\ell^*} \leq \bar{B}$ by the fact that it corresponds to the budget of the ℓ-th head with a weight $\hat{\varepsilon}^{*} = 1$ (all other weights are 0). Therefore, in both cases $\hat{\mu}\left(\hat{B}^{\ell^*} - \sum_{\ell=1}^{M} \hat{\varepsilon}^{\ell}\hat{B}^{\ell}\right) \leq 0$. On the other, we have by the properties of the KL divergence,

$$\sum_{\ell=1}^{M} \hat{\varepsilon}^{\ell} \log\left(\frac{\hat{\varepsilon}^{\ell}}{\pi^{\ell}}\right) \leq 0 \ ,$$

and then Equation (17) *becomes*

$$\frac{1}{n_2} \sum_{(\mathbf{X},Y)\in\mathcal{D}_{n_2}} \mathbb{1}_{\{\hat{g}^{EERO}(\mathbf{X})\neq Y\}} \leq \hat{R}^{\ell^*} + \beta \log(1/\pi^{\ell^*}) \ ,$$

since $\log(\hat{\varepsilon}^{\ell^*}) = 0$ *and this ends the proof by taking the expectation from both side.*