EENET: LEARNING TO EARLY EXIT FOR ADAPTIVE INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Budgeted adaptive inference with early exits is an emerging technique to improve the computational efficiency of deep neural networks (DNNs) for edge AI applications with limited resources at test time. This method leverages the fact that different test data samples may not require the same amount of computation for a correct prediction. By allowing early exiting from full layers of DNN inference for some test examples, we can reduce latency and improve throughput of edge inference while preserving performance. Although there have been numerous studies on designing specialized DNN architectures for training early-exit enabled DNN models, most of the existing work employ hand-tuned or manual rule-based early exit policies. In this study, we introduce a novel multi-exit DNN inference framework, coined as EENet, which leverages multi-objective learning to optimize the early exit policy for a trained multi-exit DNN under a given inference budget. This paper makes two novel contributions. First, we introduce the concept of early exit utility scores by combining diverse confidence measures with class-wise prediction scores to better estimate the correctness of test-time predictions at a given exit. Second, we train a lightweight, budget-driven, multi-objective neural network over validation predictions to learn the exit assignment scheduling for query examples at test time. The EENet early exit scheduler optimizes both the distribution of test samples to different exits and the selection of the exit utility thresholds such that the given inference budget is satisfied while the performance metric is maximized. Extensive experiments are conducted on five benchmarks, including three image datasets (CIFAR-10, CIFAR-100, ImageNet) and two NLP datasets (SST-2, AgNews). The results demonstrate the performance improvements of EENet compared to existing representative early exit techniques. We also perform an ablation study and visual analysis to interpret the results.

1 INTRODUCTION

Deep neural networks (DNNs) have shown unprecedented success in various fields such as computer vision and NLP, thanks to the advances in computation technologies (GPUs, TPUs) and the increasing amount of available data to train large and complicated DNNs. However, these models usually have very high computational cost, which leads to many practical challenges in deployment on edge computing applications, especially for edge clients with limited resources such as smartphones, IoT devices, and embedded devices (Goodfellow et al., 2016; Laskaridis et al., 2021; Teerapittayanon et al., 2017). With this motivation, there has been a significant research focus on improving computational efficiency of DNN models, especially at inference time. To this end, several efficient techniques, such as model quantization (Gholami et al., 2022), neural network pruning (Ghosh et al., 2022), knowledge distillation (Hinton et al., 2015) and early exiting (Teerapittayanon et al., 2016), have been introduced in the literature. Among these, early exiting has emerged as an efficient and customizable approach for deploying complex DNNs on edge devices, thanks to modular implementation and flexibility in terms of supporting multi-fidelity application scenarios.

Early exiting employs the idea of injecting early exit classifiers into some intermediate layers of a deep learning model and gaining the capability to adaptively stop inference at one of these early exits in runtime (Laskaridis et al., 2021). In particular, this technique enables several different application scenarios for efficient inference such as subnet-based inference where a single-exit submodel is selected and deployed based on the constraints of the edge device. Another use case is the budget-

constrained adaptive inference where the multi-exit DNN model is deployed under a given inference budget. At model training phase, we need to train the DNN model with the additional multiple exit branches through joint loss optimization. During inference, the early-exit enabled DNN model can elastically adjust how much time to spend on each sample based on an early exit scheduling policy to maximize the overall performance metric under the given inference budget. In this setting, through learning an early exit policy, early exiting has the potential to leverage the fact that all input samples do not have to require the same amount of computation for a correct prediction. This approach provides efficient utilization of the provided inference resources on heterogeneous edge devices by exiting earlier for easier and later for more challenging data samples, as shown for some example images in Figure 1. Even though there is a significant line of research on improving the performance of early-exit neural networks through designing specialized DNN architectures (Huang et al., 2018; Veniat & Denoyer, 2018; Yang et al., 2020; Elbayad et al., 2020) and DNN training algorithms (Li et al., 2019; Phuong & Lampert, 2019), work on optimizing early exit policies is very limited. In the literature, most methods still consider hand-tuned or heuristics-based approaches in early exiting.

With this motivation in mind, we introduce EENet, the first lightweight and budget-driven early exit policy optimization framework, to learn the optimal early-exit policy for adaptive inference given a trained multi-exit model and inference budget. In particular, our approach employs a two-branch neural network optimized on validation predictions to estimate the correctness of a prediction and exit assignment of a sample. The design of EENet makes two original contributions. First, EENet introduces the concept of exit utility scores, and computes the exit utility score for each test input by jointly evaluating and combining two complimentary statistics: (i) the multiple confidence scores that quantify the correctness of the early exit prediction output and (ii) the class-wise prediction scores. This enables EENet to handle the cases with statistical differences among prediction scores for different classes. Second, based on exit utility scoring and inference budget constraint, the EENet early exit scheduler optimizes the distribution of test samples to different exits and auto-selects the exit utility threshold for each early exit such that the test performance is maximized while the inference budget is satisfied. The design of EENet is model-agnostic and hence, applicable to all pre-trained multi-exit DNN models. In addition, EENet enables flexible splitting of multi-exit DNN models for edge clients with heterogeneous computational resources by running only a partial model until a certain early exit.

Figure 1: Example easy/difficult images from ImageNet on four different classes.

We conduct extensive experiments to evaluate EENet with multiple DNN architectures (ResNet (He et al., 2016), DenseNet (Huang et al., 2017), MSDNet (Huang et al., 2018), BERT (Devlin et al., 2019)) on three image classification benchmarks (CIFAR10, CIFAR100, ImageNet) and two NLP benchmarks (SST-2, AgNews). We demonstrate the improvements of EENet in terms of test accuracy under a given inference budget (average latency), compared to existing representative approaches, such as BranchyNet (Teerapittayanon et al., 2016), MSDNet (Huang et al., 2018) and PABEE (Zhou et al., 2020), which has specifically been introduced for NLP tasks. We consider average latency as the budget definition as it usually reflects the inference constraints in real-life applications, compared to most of the existing studies that only analyze #FLOPs. We also provide an ablation study and visual analysis to interpret the behavior of our approach. Lastly, we report the computational cost statistics in space and time in terms of number of parameters and #FLOPs.

2 Related Work

BranchyNet (Teerapittayanon et al., 2016) is the first to explore the idea of early exits. It considers the entropy of prediction scores as the measure of confidence and sets the early exit thresholds heuristically. MSDNet (Huang et al., 2018) is the most representative architecture-specific early exit DNN training solution, which uses maximum prediction score instead of entropy as the exit confidence measure. These manually defined confidence measures may be suboptimal, especially in the existence of statistical differences in prediction scores for different classes. Furthermore, these studies only consider the image classification task. Recent studies apply early exiting on NLP tasks. For example, PABEE (Zhou et al., 2020) shows that the manual prediction score-based exit confidence measuring approaches may cause substantial performance drop for NLP tasks. Hence,

PABEE proposes an early exit criterion based on the agreement among early exit classifiers, which stops the inference when the number of predictions on the same output reaches a certain patience threshold. However, this method may require having a high number of early exits to produce meaningful scores so that the samples can be separated with a higher resolution for the exit decision. Nevertheless, all these methods introduce manually defined task-specific rules, which do not include any optimization of the early exit policies in terms of scoring function and threshold computation.

Some recent efforts propose other task-dependent confidence measures (Lin et al., 2021; Li et al., 2021) or modifying the training objective to include exit policy learning during the training of a multi-exit DNN (Dai et al., 2020; Chen et al., 2020). EPNet (Dai et al., 2020) proposes to model the problem using Markov decision processes however they add an early exit classifier at each exit to increase the number of states, which is computationally unfeasible since each early exit introduces an additional computational cost during both training and inference, especially for deeper models. (Chen et al., 2020) proposes a variational Bayesian approach to learn when to stop predicting during training. Another drawback of these approaches is to require a larger number of early exits to produce meaningful scores. To the best of our knowledge, EENet is the first to learn optimal early exit policies independent of the multi-exit DNN training process.

3 EENET ARCHITECTURE AND METHODOLOGY

Given a pre-trained DNN model with N layers, one can inject multiple exits and finetune the model. Three key questions are (i) what number of exits K is most suitable, (ii) how to determine which layers l_k to place each exit k, and (iii) which optimization algorithms to use for training. The first two questions remain open, hence the number and location of exits is manually selected in existing work. Although most existing research addresses the question (iii) by developing complex training algorithms, they use manually tuned exit policies during inference. We argue that a model-agnostic approach should focus on lightweight adaptive learning of optimal early exit policies that can be applied to any pre-trained multi-exit model. To this end, we perform the optimization of early exit policies upon the completion of multi-exit model training. In this section, we provide the details of EENet by first explaining the multi-exit model training approach and then describing the model-agnostic adaptive inference optimization methodology.

3.1 MULTI-EXIT MODEL TRAINING

To enable the given pre-trained DNN classifier to perform early exiting, we first inject early exit classifiers into the model at certain intermediate layers. We set the exit locations l_k following even spacing principle such that $l_k = l_0 + kL$ for $k \in \{1, 2, \dots K-1\}$, where l_0 is the location of the first exit, $L = \lfloor \frac{N-l_0}{K-1} \rfloor$ and $l_K = N$ is the location of last exit, i.e. the full model. Let f denote a multi-exit classification model capable of outputting multiple predictions in one forward pass after the injection of early exit subnetworks f_k^e. The architecture of f_k^e should be designed in a way that the additional cost is negligible compared to the full model. Therefore, we employ 3-layer CNNs and a 1-layer fully-connected layer for image and text classification models respectively.

Let us denote the set of output probability scores of f for one input sample $\boldsymbol{x}$ as $\{\hat{\boldsymbol{y}}_k\}_{k=1}^K$ and the corresponding label as $y \in \mathcal{C}$, where K is the number of exits and $\mathcal{C} = \{1, 2, \dots C\}$ is the set of classes. Here, at each exit k, $\hat{\boldsymbol{y}}_k = f_k(\boldsymbol{x}) = [\dots \hat{y}_{k,c} \dots] \in \mathbb{R}^C$ is the vector of prediction scores for each class c, where $f_k \triangleq f_k^e \circ f_k^c \circ \dots f_1^c$ with f_k^c the kth core subnetwork and f_k^e the kth early exit subnetwork of the multi-exit model f. During the training/finetuning of these models, we minimize the weighted average of cross-entropy losses from each exit: $\mathcal{L}_{multi\_exit} = \sum_{k=1}^K \gamma_k \mathcal{L}_{CE_k}$, where $\mathcal{L}_{CE_k}$ is the cross-entropy loss and $\gamma_k = \frac{k}{K(K+1)}$ is the loss weight of the kth exit.

3.2 ADAPTIVE EARLY-EXIT INFERENCE OPTIMIZATION

Problem Definition: After training the multi-exit classification model f with K exits, we can move forward to generate an early exit policy under given budget constraints. To this end, we consider a given average per-sample inference budget B (in terms of latency, #FLOPs etc.), and the vector of inference costs $\boldsymbol{c} \in \mathbb{R}^K$ of f until each exit. On a dataset with N examples, $\mathcal{D} = \{(\{\hat{\boldsymbol{y}}_{n,k}\}_{k=1}^K, y_n)\}_{n=1}^N$ containing model prediction scores on validation samples and corresponding

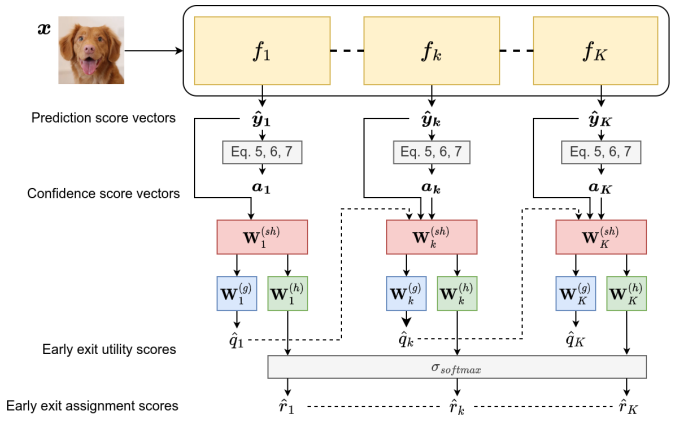


Figure 2: System architecture of EENet.

labels, the goal is to find the optimal exit utility scoring functions $(\{g_k\}_{k=1}^K)$ and the thresholds $\boldsymbol{t} \in \mathbb{R}^K$ that maximizes the accuracy as follows:

$$\boldsymbol{t}, g = \underset{\boldsymbol{t}\in\mathbb{R}^K, \{g_k:\mathbb{R}^D\to\mathbb{R}\}_{k=1}^K}{\arg\max} \frac{1}{N}\sum_{n=1}^{N} \mathbf{1}_{\hat{y}_{n,k_n}=y_n}, \tag{1}$$

$$k_n = min\{k|g_k(\hat{\boldsymbol{y}}_{n,k}) \geq t_k\} \tag{2}$$

while satisfying the given average per-sample inference budget B such that $\frac{1}{N}\sum_{n=1}^{N} c_{k_n} \leq B$. Here, k_n denotes the minimum exit index where the computed utility score was greater or equal to the threshold of that exit, i.e. the assigned exit for the nth sample. The pair of exit utility scoring functions $(g_1, g_2 \ldots g_K)$ and thresholds $(t_1, t_2, \ldots t_K)$ that maximizes the validation accuracy while satisfying the given average budget are then used for early-exit enabled inference.

EENet Architecture: Figure 2 gives an overview of our early exit policy learning architecture for adaptive inference. We solve the problem of optimizing an early exit policy by developing a multi-objective optimization approach with the target variables q_k and r_k, representing the correctness of a prediction and exit assignment at the kth exit such that

$$q_k = \begin{cases} 1 & \text{if} \quad \hat{y}_k = y \\ 0 & \text{if} \quad \hat{y}_k \neq y \end{cases} \tag{3}$$

$$r_k = \begin{cases} \frac{1}{\sum_{k'=1}^{K} q_{k'}} & \text{if} \quad \hat{y}_k = y \\ \frac{1}{K} & \text{if} \quad \hat{y}_{k'} \neq y \quad \forall k' \in \{1 \ldots K\} \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

where $\hat{y}_k \triangleq \arg\max_{c\in\mathcal{C}} \hat{y}_{k,c}$. In addition, let us denote the confidence score vector $\boldsymbol{a}_k$ containing different measures based on maximum score, entropy and voting such that

$$a_k^{(max)} = \max_{c\in\mathcal{C}} \hat{y}_{k,c}, \tag{5}$$

$$a_k^{(entropy)} = 1 + \frac{\sum_{c'=1}^{C} \hat{y}_{k,c'} \log \hat{y}_{k,c'}}{\log C}, \tag{6}$$

$$a_k^{(vote)} = \frac{1}{k}\max_{c\in\mathcal{C}} \sum_{k'=1}^{k} \mathbf{1}_{\hat{y}_{k'}=c}. \tag{7}$$

At each exit k, using the prediction score vector $\hat{\boldsymbol{y}}_k$ and the confidence score vector $\boldsymbol{a}_k$, we compute utility score $\hat{q}_k$ and assignment score $\hat{r}_k$ as follows:

$$\boldsymbol{s}_k = \sigma_{lrelu}(\mathbf{W}_k^{(sh)}[\hat{\boldsymbol{y}}_k, \boldsymbol{a}_k, \boldsymbol{b}_k]), \tag{8}$$

$$\hat{q}_k = g_k(\hat{\boldsymbol{y}}_k, \boldsymbol{a}_k, \boldsymbol{b}_k) = \sigma_{sig}(\mathbf{W}_k^{(g)}\boldsymbol{s}_k), \tag{9}$$

$$\tilde{r}_k = h_k(\hat{\boldsymbol{y}}_k, \boldsymbol{a}_k, \boldsymbol{b}_k) = \mathbf{W}_k^{(h)}\boldsymbol{s}_k, \quad \hat{r}_k = \frac{e^{\tilde{r}_k}}{\sum_{k'=1}^{K} e^{\tilde{r}_{k'}}}, \tag{10}$$

where $\boldsymbol{b}_k = [\hat{q}_1, \dots \hat{q}_{k-1}]$ for $k > 1$ and an empty vector for $k = 1$. $\sigma_{lrelu}(x) = \max(0, x) + 0.01 * \min(0, x)$ is the leaky ReLU activation function. Here, $\mathbf{W}_k^{(g)}, \mathbf{W}_k^{(h)} \in \mathbb{R}^{1 \times D_h}$ and $\mathbf{W}_k^{(sh)} \in \mathbb{R}^{D_h \times D}$ are the fully-connected layer weights for exit utility score function g_k, exit assignment function h_k and the shared part of the network for these functions. Thus, the model weights for kth exit can be denoted as $\boldsymbol{\theta}_k^{(g)} = \{\mathbf{W}_k^{(g)}, \mathbf{W}_k^{(sh)}\}$ and $\boldsymbol{\theta}_k^{(h)} = \{\mathbf{W}_k^{(h)}, \mathbf{W}_k^{(sh)}\}$. Here, $D = C + k + 2$ is the number of input features and D_h is the hidden layer size.

Optimization: After computing the target values q_k and r_k representing the correctness of a prediction and exit assignment using Equations equation 3 and equation 4, we compute the multi-objective loss defined as $\mathcal{L} = \mathcal{L}_g + \mathcal{L}_h$, where $\mathcal{L}_g$ and $\mathcal{L}_h$ are the losses observed by the exit utility scoring functions g_k, and exit assignment estimator functions h_k respectively. We define $\mathcal{L}_g$ as follows:

$$\mathcal{L}_g = \frac{1}{K} \sum_{n=1}^{N} \sum_{k=1}^{K} w_{n,k} \ell_g(\hat{q}_{n,k}, q_{n,k}) \text{ such that} \tag{11}$$

$$\ell_g(\hat{q}_k, q_k) = q_k \log(\hat{q}_k) + (1 - q_k) \log(1 - \hat{q}_k) \text{ and} \tag{12}$$

$$w_{n,k} = \frac{1 - \sum_{k'=1}^{k-1} \hat{r}_{k',n}}{\sum_{n'=1}^{N} (1 - \sum_{k'=1}^{k-1} \hat{r}_{k',n})}, \tag{13}$$

where N is the number of validation data samples and $w_{n,k}$ is the loss weight for the nth sample at the kth exit. This weighting scheme based on the survival possibilities up to that exit encourages exit score estimator functions to specialize in their respective subset of data. The computation of weights in Equation equation 13 is performed outside the computation graph. Lastly, we define $\mathcal{L}_h = \alpha \mathcal{L}_{budget} + \beta \mathcal{L}_{CE}$ such that

$$\mathcal{L}_{budget} = \frac{1}{B} |B - \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} \hat{r}_{n,k} c_k| \text{ and} \tag{14}$$

$$\mathcal{L}_{CE} = -\frac{1}{NK} \sum_{n=1}^{N} \sum_{k=1}^{K} \log(\hat{r}_{n,k}) r_{n,k}, \tag{15}$$

where $\alpha, \beta > 0$ are the loss weighting parameters and r_k is defined in equation 3.2. We learn $\{\boldsymbol{\theta}_k^{(g)}\}_{k=1}^K$ and $\{\boldsymbol{\theta}_k^{(h)}\}_{k=1}^K$ by minimizing $\mathcal{L}$ on $\mathcal{D} = \{(\{\hat{\boldsymbol{y}}_{n,k}\}_{k=1}^K, y_n)\}_{n=1}^N$ using stochastic gradient descent. We provide the pseudocode for the optimization of the utility function g and thresholds $\boldsymbol{t}$ in Algorithm 1. Here, given the validation dataset with model predictions/labels, budget and inference costs, we first learn the exit utility scoring and assignment functions by minimizing $\mathcal{L}$. Then for each exit, we let the most utilizable samples exit until the quota assigned to that exit is full. We set the threshold to the exit utility score of the last exited sample with the lowest score. We also provide Algorithm 2 for early-exit enabled adaptive inference with EENet in Appendix A. During inference, at each exit k, we compute the exit utility score using the optimized exit utility scoring function g_k from the output of Algorithm 1 and stop the inference if the score is above the threshold t_k.

4 EXPERIMENTS

We conduct extensive experiments to evaluate EENet and report the performance improvements obtained by EENet for budget-constrained adaptive inference on five benchmarks (CIFAR-10, CIFAR-100, ImageNet, SST-2 and AgNews). We demonstrate that EENet consistently outperforms existing representative multi-exit solutions, such as BranchyNet (Teerapittayanon et al., 2016), MSDNet (Huang et al., 2018) and PABEE (Zhou et al., 2020). Our ablation study and visual analysis further interpret the design features of EENet.

Datasets and Preprocessing: In image classification experiments, we work on CIFAR-10/100 (Krizhevsky, 2009) and ImageNET (Deng et al., 2009) datasets. CIFAR-10 and CIFAR-100 contain 50000 train and 10000 test images with 32x32 resolution from 10 and 100 classes respectively. ImageNET contains 1.2 million train and 150000 validation images (used for test) with 224x224 resolution from 1000 classes. We hold out randomly selected 5000 images from CIFAR-10/100 train set and 25000 images from ImageNET train set for validation. We follow the data augmentation

Algorithm 1 `Early Exit Inference Optimization Algorithm`

Inputs: $\mathcal{D} = \{(\{\hat{\boldsymbol{y}}_{n,k}\}_{k=1}^{K}, y_n)\}_{n=1}^{N}$ (validation predictions and labels), B (average inference budget per sample), $\boldsymbol{c} \in \mathbb{R}^K$ (inference costs per sample until each exit)
Output: $\{g_k\}_{k=1}^{K}$ (exit utility score functions), $\boldsymbol{t} \in \mathbb{R}^K$ (thresholds)

Initialize: $\boldsymbol{h} \leftarrow \texttt{zeros}(N)$, $\boldsymbol{t} \leftarrow \texttt{ones}(k) * 1e8$
Learn $\{g_k\}_{k=1}^{K}$ and $\{h_k\}_{k=1}^{K}$ by minimizing $\mathcal{L}$ on $\mathcal{D}$.
Compute exit scores: $\mathbf{C} \triangleq (\hat{q}_{n,k}) \in \mathbb{R}^{N \times K}$ using equation 9
$\mathbf{S} = (s_{n,k}) \in \mathbb{R}^{N \times K} \leftarrow \texttt{argsort}(\mathbf{C}, 1)$
for *exit index* $k = 1$ **to** K **do**
    $c \leftarrow 0$
    Estimate exit distribution: $p_k \leftarrow \frac{1}{N} \sum_{n=1}^{N} \hat{r}_{n,k}$ using equation 10
    for *sample index* $n = 1$ **to** N **do**
        if $h_{s_{n,k}} = 0$ **then**
            $c \leftarrow c + 1$
            $h_{s_{n,k}} \leftarrow 1$
            if $c = \texttt{round}(Np_k)$ **then**
                $t_k \leftarrow \hat{q}_{s_{n,k},k}$
                break
$t_K \leftarrow -1e8$
return $\{g_k\}_{k=1}^{K}, \boldsymbol{t}$

techniques applied in (He et al., 2016), zero padding, center cropping, random horizontal flip with 0.5 probability. In text classification experiments, we consider SST-2 (Socher et al., 2013) and AGNews (Zhang et al., 2015) datasets. SST-2 contains 67349 train, 872 validation and 1821 test sentences with positive or negative labels. AGNews contains 120000 train and 7600 test sentences from four classes. We hold randomly selected 5000 sentences for validation. For tokenization, we use the pre-trained tokenizer for BERT model provided by open-source HuggingFace (Wolf et al., 2020).

Experimental Setup: We perform experiments with ResNet (He et al., 2016) on CIFAR-10 and with DenseNet121 (Huang et al., 2017) on CIFAR-100. We use the default ResNet settings for 56-layer architecture and insert two evenly spaced early exits at the 18th and 36th layers. For DenseNet, we follow the default settings for 121-layer configuration and insert three early exit layers at the 12th, 36th and 84th layers after transition layers. We train these models using Adam optimizer (Kingma & Ba, 2015) for 150 epochs (first 20 epochs without early exits) and the batch size of 128, with the initial learning rate of 0.1 (decays by 0.1 at 50-th and 100th epochs). On ImageNet dataset, we use MSDNet (Huang et al., 2018) with 35 layers, 4 scales and 32 initial hidden dimensions. We insert four evenly spaced early exits at the 7th, 14th, 21th and 28th layers. Each early exit classifiers consist of three 3x3 convolutional layers with ReLU activations. We use the pre-trained BERT (Devlin et al., 2019) provided by the open-source HuggingFace library (Wolf et al., 2020) on SST-2 and AgNews datasets. We insert three evenly spaced early exits at the 3rd, 6th and 9th layers. Each early exit classifier consists of a fully-connected layer. We finetune the models for 20 epochs using gradient descent with the learning rate of 3e-5 and batch size of 16. For EENet, we set $D_h = 0.5D$, $\alpha = 1e-1$ and $\beta = 1e-3$ for image classification experiments and we optimize the weights using Adam optimizer with the learning rate of $3e-5$ on validation data. For text classification experiments, we set $D_h = 2D$, $\alpha = 1e-2$ and $\beta = 1e-4$, and use the learning rate of $1e-4$. We use L2 regularization with the weight of 0.01. In all experiments, we stop the training if the loss does not decrease for 50 consecutive epochs on the validation set. For MSDNet (Huang et al., 2018), BranchyNet (Teerapittayanon et al., 2016) and PABEE (Zhou et al., 2020), we use maximum score $a_k^{(max)}$, entropy-based $a_k^{(entropy)}$ and agreement-based $a_k^{(vote)}$ scores as provided in equations 5, 6 and 7. To compute the thresholds for these methods, we follow the approach in (Huang et al., 2018) and assume that exit assignment of samples will follow the geometric distribution that satisfies the budget on validation data. Our implementation is on Python 3.7 with PyTorch 1.12 library. Each latency measurement is carried out 100 times on a machine with 8-core 2.9GHz CPU. The extra inference time caused by the computations of equations equation 8 and equation 9 are also included in the reported latency measurements, and the cost is much smaller compared to the cost of the forward pass of the model as shown in Table 2. The source code of EENet is available at [anonymized].

Dataset, Model and Base Performance	Average Latency Budget per sample	Accuracy (%)			
		BranchyNet	**MSDNet**	**PABEE**	**EENet**
CIFAR-10	3.50 ms	93.76	93.81	93.69	**93.84**
ResNet56 w/ 3 exits	3.00 ms	92.57	92.79	91.85	**92.90**
93.90% @ 4.70 ms	2.50 ms	87.55	88.76	84.39	**88.90**
CIFAR-100	7.50 ms	73.96	74.01	73.68	**74.08**
DenseNet121 w/ 4 exits	6.75 ms	71.65	71.99	68.10	**72.75**
75.08% @ 10.20 ms	6.00 ms	68.13	68.70	61.13	**70.15**
ImageNet	1.50 s	74.10	74.13	74.05	**74.18**
MSDNet35 w/ 5 exits	1.25 s	72.44	72.70	72.40	**72.75**
74.60% @ 2.35 s	1.00 s	69.32	69.76	68.13	**69.88**
SST-2	2.50 s	90.86	91.00	90.75	**92.07**
BERT w/ 4 exits	2.00 s	87.66	87.71	86.99	**91.45**
92.36% @ 3.72 s	1.75 s	84.33	84.30	80.99	**90.45**
AgNews	2.50 s	92.95	92.98	92.57	**93.84**
BERT w/ 4 exits	2.00 s	85.58	84.93	85.22	**93.75**
93.98% @ 3.72 s	1.50 s	75.08	73.07	74.67	**90.63**

Table 1: Image and text classification experiment results in terms of accuracy obtained at different budget levels (average latency per sample).

Validation of EENet with Comparison: We compare EENet with BranchyNet (Teerapittayanon et al., 2016), MSDNet (Huang et al., 2018) and PABEE (Zhou et al., 2020) in terms of the accuracy obtained under different budget constraints. We consider average latency per sample as the budget definition throughout the experiments. To this end, we collect results for each method within the budget range of $B \in [c_1, c_K]$. Table 1 contains the results on CIFAR-10, CIFAR-100, ImageNet, SST-2 and AgNews datasets and as demonstrated, EENet consistently performs better compared to other early exit approaches. For example, consider CIFAR-100 (row 2), the original pre-trained DenseNet121 with target accuracy of 75.08% has the average inference time of 10.20ms. We set three levels of early-exit inference time budgets: 7ms, 6.75ms, 6ms. For CIFAR-100 under the low budget setting with 6 ms, EENet yields 1.45% higher accuracy compared to MSDNet, the second best performer, and outperforms PABEE by over 9% in accuracy gain, showing that the optimal early exit policy learned by EENet is more efficient compared to the respective hand-tuned early exit policy used in BranchyNet, MSDNet and PABEE. In addition, we observe that EENet achieves greater performance gains as the budget tightens. These observations are consistent over all benchmarks. In NLP experiments, the performance gains brought by EENet are more significant compared to the existing methods, with the improvements ranging from 1% (at high budget) to 15% (at low budget). For example, for SST-2, under the low inference budget of 1.75s, EENet achieves 90.45% accuracy compared to 84.33% by the second best performing multi-exit approach BranchyNet. Under the 2s inference budget, EENet achieves 91.45% accuracy compared to 87.71% by the second best multi-exit approach MSDNet. Similar observations are also found in AgNews, showing EENet offers more consistent and stable performance improvement for all five benchmarks.

Visual Analysis of Some EENet Design Features: In Figure 3, we analyze through examples to show why the exit utility scores produced by EENet are more effective in finding optimal early exit policy by comparing with the maximum prediction scoring method for early exit used in MSDNet and others in the literature. Randomly selected ten classes are listed on the x-axis sorted by the accuracy achieved using the full model on the corresponding class. From the left figure, using maximum prediction scores to determine exit utility may lead to missing some good early exit opportunities. For example, consider those classes that the predictor model produces relatively low maximum prediction scores (less than 0.7), such as lizard, man, butterfly and fox. Even though the predictor can predict them correctly but the relative confidence is not very high. In comparison, EENet defines the exit utility scoring by combining three quality measures (entropy, maximum prediction confidence and voting). Hence, the exit utility scores obtained by EENet reflect the easiness of test examples more accurately. For example, those classes that have lower maximum prediction scores on the true predictions, such as lizard, man, butterfly and fox, will have high early exit utility scores in EENet as shown in the right figure highlighted in the blue oval. Similarly, the classes that make the false prediction with high maximum confidence, such as castle, highlighted in the lower right corner, will have low exit utility score in EENet. This shows one of the novel features of EENet for learning optimal early exit policies by leveraging high-quality exit-utility function and ranking under given

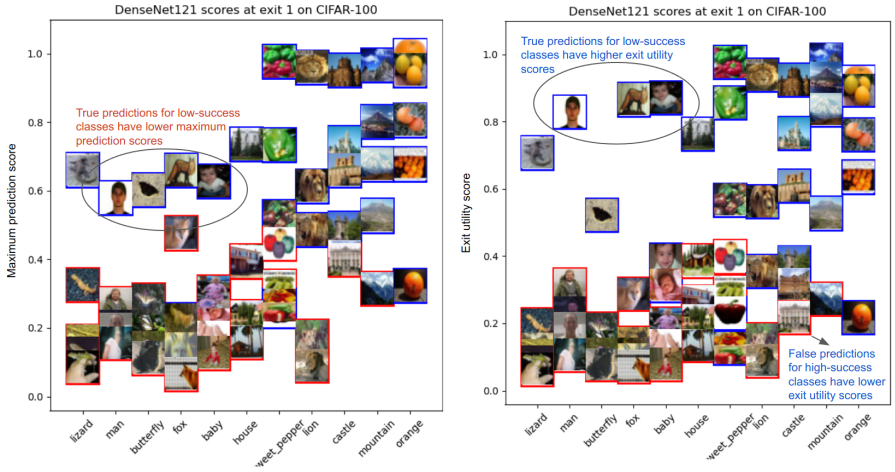


Figure 3: Analysis on the benefit of exit utility scores obtained by EENet, which provides a clearer separation of true and false predictions for all classes, compared to maximum prediction score based confidence, which is popularly used in the literature. Images with blue/red borders are predicted correctly/incorrectly at the first exit of DenseNet121.

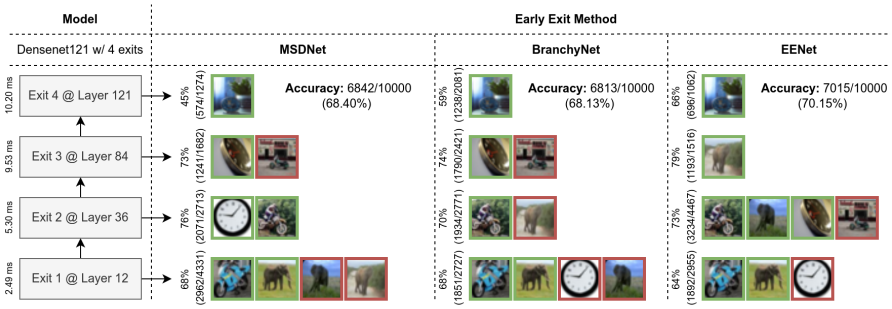


Figure 4: Visual comparison of the early exit approaches on CIFAR-100 test data with DenseNet121 (4 exits) for the average latency budget of 6 ms. We illustrate the randomly selected nine samples from three classes and the exit location that they were assigned. Images with green/red borders are predicted correctly/incorrectly at the corresponding exit. We also report the number of correct predictions and exited samples at each exit. In this case, EENet obtains the performance gain by allowing more samples to exit at the second exit.

accuracy and inference latency budget constraints. Further analysis on CIFAR-100 and AgNews are provided in Section B.

Figure 4 provides a visual comparison of EENet (right) with MSDNet (left) and BranchyNet (middle) on CIFAR-100 test data, with four exits of the respective early exit models. The visual comparison is using the average latency budget of 6 ms (recall Table 1). We use the randomly selected nine examples from three classes in the test set and display the exit location that they were assigned by EENet (right) and by MSDNet (left). Images with green/red borders are predicted correctly/incorrectly at the corresponding exit. We also report the number of correct predictions and the number of exited text examples at each exit. In this case, EENet obtains the performance gain over MSDNet by allowing more correct predictions to exit earlier at the second exit.

Ablation Study: We also analyze the effect of different components in EENet on performance by in-depth investigation of the results for SST-2 and AgNews. Figure 5 provides the plots of average inference time vs. accuracy for two additional variants of EENet and compares with MSDNet and BranchyNet. The first variant of EENet shows the results of EENet without optimizing the exit utility scoring, and instead, directly using maximum prediction scores. The second variant shows the results of EENet without optimizing exit distributions through our budget-constrained learning, and instead, directly using geometric distribution. For both SST-2 and AgNews, EENet outperform its two variants. All three versions of EENet outperforms MSDNet and BranchyNet.

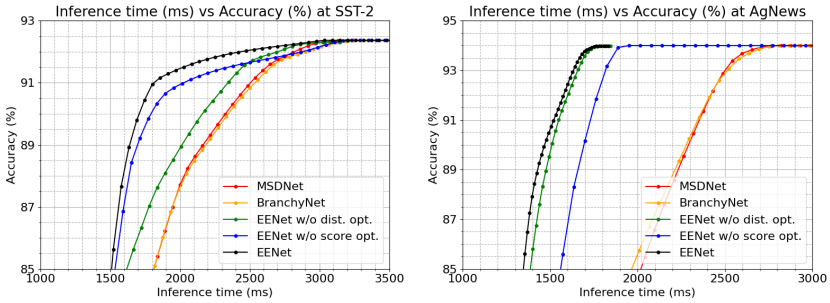


Figure 5: Average latency (ms) vs Accuracy (%) results at SST-2 and AgNews datasets for BranchyNet, MSDNet and EENet variations (without distribution/scoring optimization).

Model	Exit-1		Exit-2		Exit-3		Exit-4		Exit-5		Base Model	
	#PRMs	#FLOPs	#PRMs	#FLOPs	#PRMs	#FLOPs	#PRMs	#FLOPs	#PRMs	#FLOPs	#PRMs	#FLOPs
ResNet56	0.06M	55M	0.28M	112M	0.96M	156M	-	-	-	-	0.86M	126M
(w/ EENet)	(+0.07K)	(+0.11K)	(+0.09K)	(+0.13K)	(+0.11K)	(+0.15K)	-	-	-	-	-	-
DenseNet121	0.06M	55M	0.25M	94M	0.86M	125M	1.17M	131M	-	-	1.04M	126M
(w/ EENet)	(+5.25K)	(+7.80K)	(+5.36K)	(+7.96K)	(+5.47K)	(+8.11K)	(+5.57K)	(+8.27K)	-	-	-	-
MSDNet35	8.76M	0.61B	20.15M	1.43B	31.73M	2.28B	41.86M	2.96B	62.31M	3.25B	58.70M	3.02B
(w/ EENet)	(+0.25M)	(+0.31M)	(+0.25M)	(+0.31M)	(+0.25M)	(+0.31M)	(+0.25M)	(+0.31M)	(+0.25M)	(+0.31M)	-	-
BERT	45.69M	5.46B	67.55M	10.9B	89.40M	16.4B	111.26M	21.8B	-	-	109.90M	20.9B
(w/ EENet)	(<200)	(<500)	(<200)	(<500)	(<200)	(<500)	(<200)	(<500)	-	-	-	-

Table 2: Model statistics in terms of number of parameters (#PRMs) and the number of floating point operations (#FLOPs) until each exit and the base model configuration without early exits. The cost associated with EENet is also provided in parantheses.

Flexible Deployment on Heterogeneous Edge Clients: EENet by design provides model-agnostic adaptive early exit inference, applicable to all pre-trained early exit models. Another EENet design goal is to enable flexible NN splitting by early exits, enabling edge clients with limited resources to benefit from early exit models. Table 2 reports the number of parameters (#PRMs) and number of floating point operations (#FLOPs) of the models used in the experiments until each exit for four multi-exit DNNs. For each model, we also provide the additional computational cost of EENet in employing the budgeted adaptive early exit policy. First, the increase of #FLOPs caused by EENet is negligible ($< 0.5\%$) compared to the cost of the forward pass of the original pre-trained DNN model. For the application scenarios with hard constraints (storage/RAM limitations) for edge deployment, the partial multi-exit model with EENet split at a certain exit can be delivered, with the partial model size meeting the edge deployment constraints. For this subnetwork with EENet, those test examples with exit utility score below the learned exit threshold will be passed to the next level edge server in the hierarchical edge computing infrastructure, which has higher computational capacity to continue the multi-exit inference.

5 Conclusion

We have presented EENet, a novel, lightweight and model-agnostic early exit policy optimization framework for budgeted adaptive inference. The paper makes a number of original contributions. First, our approach introduces early exit utility scoring function which combines a set of complimentary early exit confidence measures and class-wise prediction scores. Second, we optimize the assignment of test data samples to different exits by learning the optimal early exit distribution and the adaptive thresholds for test-time early exit scheduling. As opposed to previous manually defined heuristics-based early exit techniques, which may be suboptimal based on the specific multi-exit DNN architecture, our approach is model-agnostic and can easily be used in different learning tasks with pre-trained DNN models in vision and NLP applications. Extensive experiments on five benchmarks (CIFAR-10, CIFAR-100, ImageNet, SST-2, AgNews) demonstrate that EENet consistently outperforms the existing representative techniques represented by MSDNet, BranchyNet, PABEE, and the performance improvements get more significant as the given average latency budget per sample tightens. Lastly, our ablation study and visual analysis further demonstrate the effects of core components of EENet design.

REFERENCES

Xinshi Chen, Hanjun Dai, Yu Li, Xin Gao, and Le Song. Learning to stop while learning to predict. In *ICML*, pp. 1520–1530, 2020. URL http://proceedings.mlr.press/v119/chen20c.html.

Xin Dai, Xiangnan Kong, and Tian Guo. Epnet: Learning to exit with flexible multi-branch network. In *Proceedings of the 29th ACM International Conference on Information amp; Knowledge Management*, CIKM '20, pp. 235–244, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450368599. doi: 10.1145/3340531.3411973. URL https://doi.org/10.1145/3340531.3411973.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJg7KhVKPH.

Asghar Gholami, Sehoon Kim, Dong Zhen, Zhewei Yao, Michael Mahoney, and Kurt Keutzer. *A Survey of Quantization Methods for Efficient Neural Network Inference*, pp. 291–326. 01 2022. ISBN 9781003162810. doi: 10.1201/9781003162810-13.

Sayan Ghosh, Karthik Prasad, Xiaoliang Dai, Peizhao Zhang, Bichen Wu, Graham Cormode, and Peter Vajda. Pruning compact convnets for efficient inference, 2022. URL https://openreview.net/forum?id=_gZ8dG4vOr9.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2017. doi: 10.1109/CVPR.2017.243.

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Weinberger. Multi-scale dense networks for resource efficient image classification. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Hk2aImxAb.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Stefanos Laskaridis, Alexandros Kouris, and Nicholas D. Lane. Adaptive inference through early-exit networks: Design, challenges and directions. In *Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning*, EMDL'21, pp. 1–6, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450385978. doi: 10.1145/3469116.3470012. URL https://doi.org/10.1145/3469116.3470012.

Hao Li, Hong Zhang, Xiaojuan Qi, Ruigang Yang, and Gao Huang. Improved techniques for training adaptive deep networks. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1891–1900, 2019.

Shuang Li, Jinming Zhang, Wenxuan Ma, Chi Harold Liu, and Wei Li. Dynamic domain adaptation for efficient inference. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

Ziqian Lin, Sreya Dutta Roy, and Yixuan Li. Mood: Multi-level out-of-distribution detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

Mary Phuong and Christoph Lampert. Distillation-based training for multi-exit architectures. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1355–1364, 2019. doi: 10.1109/ICCV.2019.00144.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://aclanthology.org/D13-1170.

Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 2464–2469, 2016.

Surat Teerapittayanon, Bradley McDanel, and H.T. Kung. Distributed deep neural networks over the cloud, the edge and end devices. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pp. 328–339, 2017. doi: 10.1109/ICDCS.2017.226.

T. Veniat and L. Denoyer. Learning time/memory-efficient deep architectures with budgeted super networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3492–3500, Los Alamitos, CA, USA, jun 2018. IEEE Computer Society. doi: 10.1109/CVPR.2018.00368. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00368.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. Resolution adaptive networks for efficient inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/250cf8b51c773f3f8dc8b4be867a9a02-Paper.pdf.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. Bert loses patience: Fast and robust inference with early exit. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 18330–18341. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/d4dd111a4fd973394238aca5c05bebe3-Paper.pdf.

A Adaptive Inference Algorithm with Early Exits

Algorithm 2 `Early Exit Inference Algorithm`

Inputs: $\boldsymbol{x}$ (test input sample), $\{f_k\}_{k=1}^{K}$ (predictor functions), $\{g_k\}_{k=1}^{K}$ (early exit utility score estimator functions), $\boldsymbol{t}$ (early exit thresholds)
Output: $\hat{y}$ (predicted label)
for *exit index* $k = 1$ **to** K **do**
  Obtain predictor model output: $\hat{\boldsymbol{y}}_k \leftarrow f_k(\boldsymbol{x})$
  Compute exit score $\hat{q}_k$ with g_k using equation 9
  if $\hat{q}_k \geq t_k$ **then**
    $\hat{y}_k \leftarrow \arg\max \hat{\boldsymbol{y}}_k$
    return $\hat{y}_k$
return $\hat{y}_K$

B Exit Distribution Behavior Analysis on CIFAR-100 and AgNews

Here, we visualize the exit distribution behavior of EENet on CIFAR-100 under different average latency budget levels (6ms, 6.5ms, 7ms, 8ms). In the scatter plots provided in Figures 6, 7, 8 and 9, at each exit, we plot the validation samples with x-axis the class of the sample and y-axis the exit utility score. Green/red color represents correct/incorrect predictions at the corresponding exit whereas yellow cross marker is used to indicate that the sample has already exited. Computed thresholds are drawn using horizontal blue lines and the percentages of exiting samples are provided in subplot titles. We also analyze the results on AgNews by comparing the exit assignments by MSDNet, BranchyNet and EENet in Figure 10.

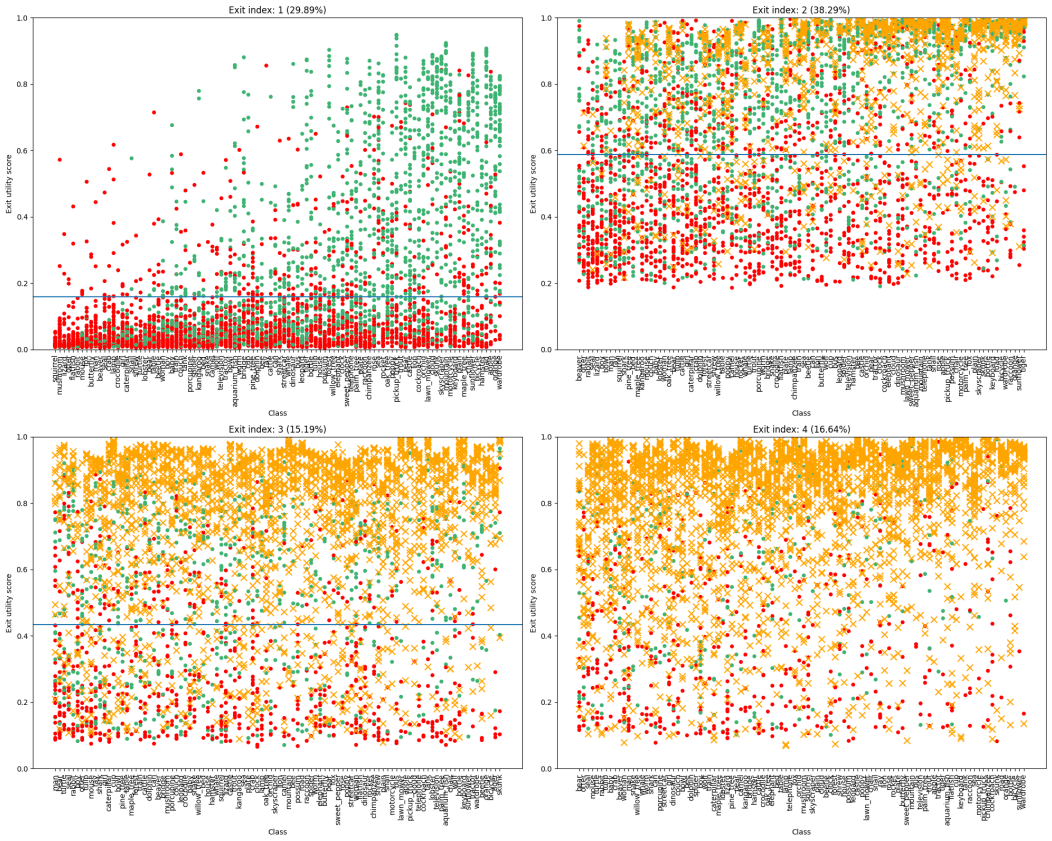


Figure 6: Distribution of samples to different exits under the average latency budget of 6 milliseconds.

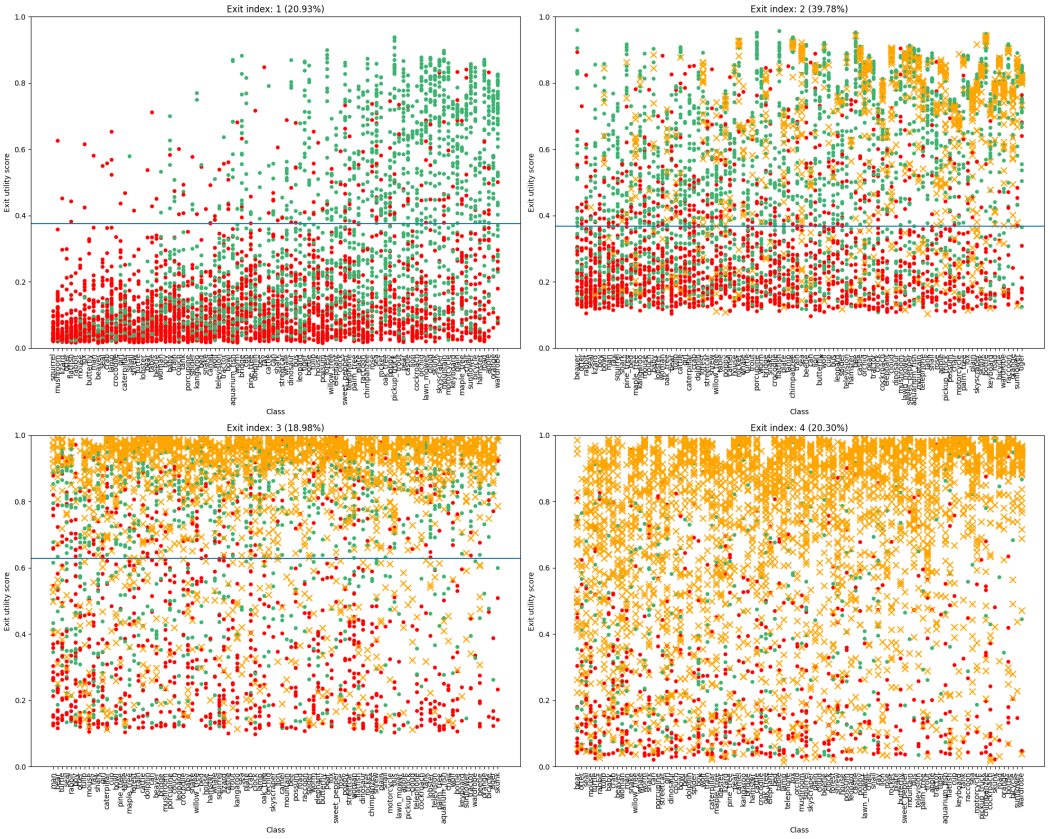


Figure 7: Distribution of samples to different exits under the average latency budget of 6.5 milliseconds.

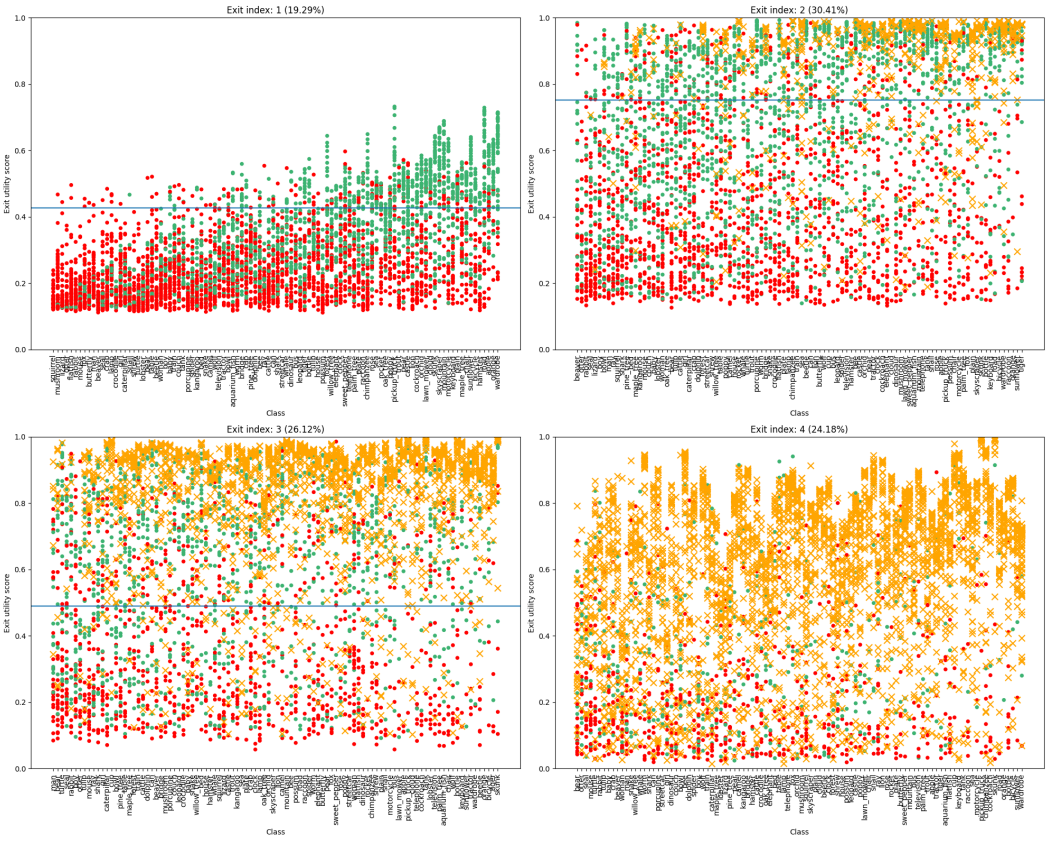


Figure 8: Distribution of samples to different exits under the average latency budget of 7 milliseconds.

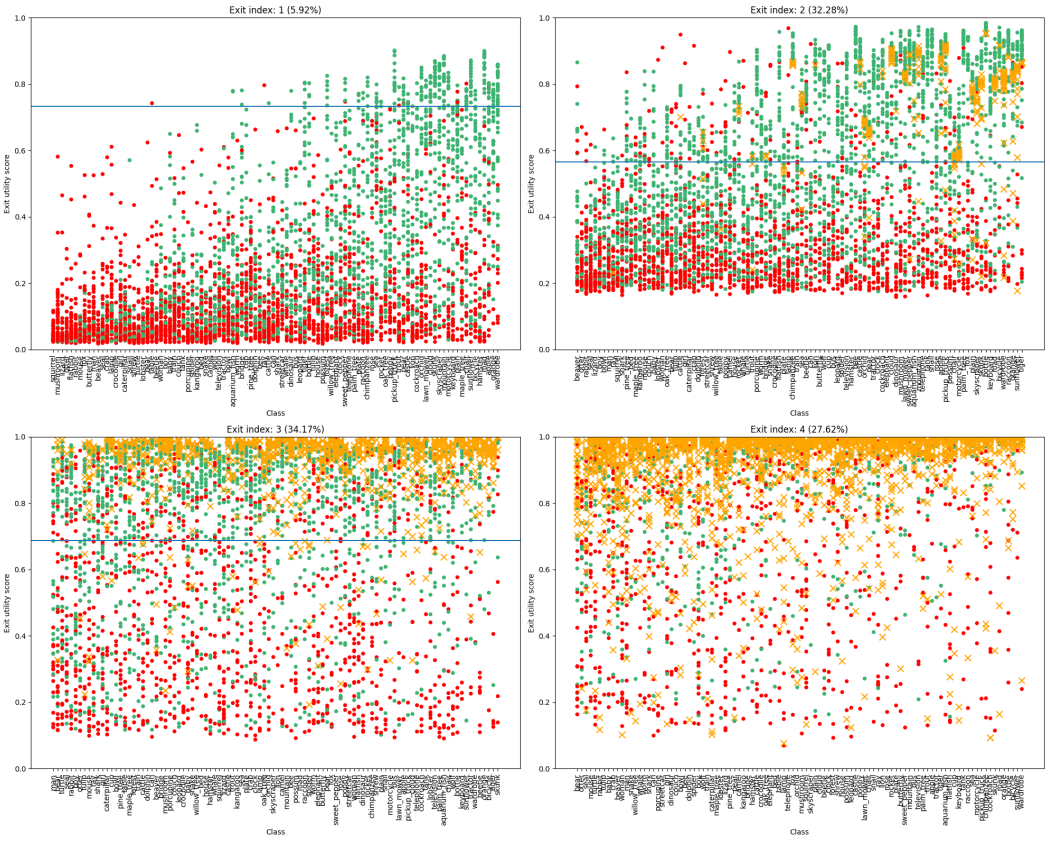
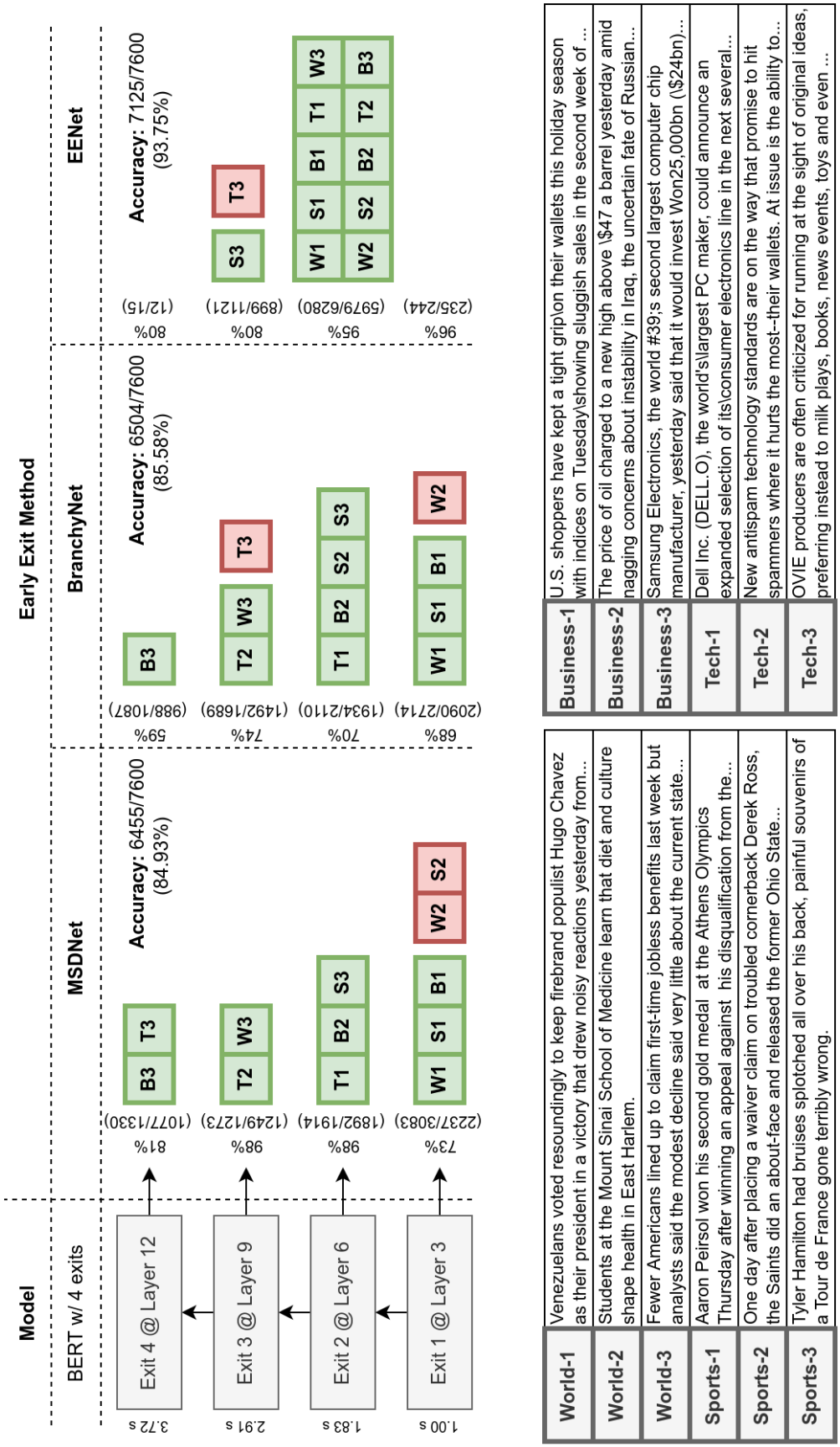


Figure 9: Distribution of samples to different exits under the average latency budget of 8 milliseconds.



World-1	Venezuelans voted resoundingly to keep firebrand populist Hugo Chavez as their president in a victory that drew noisy reactions yesterday from...
World-2	Students at the Mount Sinai School of Medicine learn that diet and culture shape health in East Harlem.
World-3	Fewer Americans lined up to claim first-time jobless benefits last week but analysts said the modest decline said very little about the current state...
Sports-1	Aaron Peirsol won his second gold medal at the Athens Olympics Thursday after winning an appeal against his disqualification from the...
Sports-2	One day after placing a waiver claim on troubled cornerback Derek Ross, the Saints did an about-face and released the former Ohio State...
Sports-3	Tyler Hamilton had bruises splotched all over his back, painful souvenirs of a Tour de France gone terribly wrong.

Business-1	U.S. shoppers have kept a tight grip\on their wallets this holiday season with indices on Tuesday\showing sluggish sales in the second week of ...
Business-2	The price of oil charged to a new high above \$47 a barrel yesterday amid nagging concerns about instability in Iraq, the uncertain fate of Russian...
Business-3	Samsung Electronics, the world #39;s second largest computer chip manufacturer, yesterday said that it would invest Won25,000bn (\$24bn)...
Tech-1	Dell Inc. (DELL.O), the world's\largest PC maker, could announce an expanded selection of its\consumer electronics line in the next several...
Tech-2	New antispam technology standards are on the way that promise to hit spammers where it hurts the most--their wallets. At issue is the ability to...
Tech-3	OVIE producers are often criticized for running at the sight of original ideas, preferring instead to milk plays, books, news events, toys and even ...

Figure 10: Visual comparison of the early exit approaches on AgNews test data with BERT (4 exits) for the average latency budget of 2 seconds. We illustrate the randomly selected twelve samples from four classes and the exit location that they were assigned. Images with green/red borders are predicted correctly/incorrectly at the corresponding exit. We also report the number of correct predictions and exited samples at each exit. EENet does not make costly assignments to the last two exits and uses the second exit more effectively.