Optimizing Early Exiting Methods in Deep Neural Networks Using Multi-Armed Bandits: Comparative Analysis

Course: **IE617: Online Machine Learning Bandit Algorithms**

Team Members:

Anubhav Binit (24M1512)
Rahul Kumar Vishwakarma (24M1522)
Dhrubang Utpal Talukdar (24M1517)
Mintu Yadav (23M1529)

1 Problem Description

The problem involves optimizing early exiting methods in deep neural networks (DNNs) to balance accuracy and computational efficiency during inference, particularly under data distribution shifts. Early exiting allows DNNs to make predictions at intermediate layers (exits), reducing computational cost for inputs with sufficient confidence. The decision to exit depends on a confidence metric, with three methods considered:

1. **Confidence-Based (DeeBERT)**: Exits when the highest probability in the output vector exceeds a threshold α. This entropy-based approach balances accuracy (higher α ensures confident predictions) and efficiency (lower α enables earlier exits).
2. **Patience-Based (PABEE)**: Exits when predictions are consistent across t consecutive classifiers. Larger t increases accuracy, while smaller t improves efficiency.
3. **Similarity-Based (MuE)**: Exits if the cosine similarity between hidden representations of consecutive layers falls below a threshold α. Lower α promotes earlier exits, while higher α ensures deeper processing.

The challenge is to learn optimal thresholds (α for DeeBERT and MuE, t for PABEE) that maintain the accuracy-efficiency trade-off during inference. The objective of this work is to learn optimal thresholds that effectively trade off between accuracy and computational efficiency during inference, even when the test dataset distribution changes. We model the problem of selecting thresholds as a multi-armed bandit (MAB) setup for each type of confidence metric, namely Confidence-Based, Patience-Based, and Similarity-Based methods. Finally, we compare these approaches and provide insights into which method performs better under various circumstances.

2 Literature Review

Confidence-Based Methods

DeeBERT [1] introduced an entropy-based early exiting mechanism for BERT, which utilizes the maximum softmax probability to determine whether the model should exit early. By using this confidence metric, DeeBERT was able to reduce inference time by approximately 40% on NLP tasks, such as GLUE, while suffering less than a 2% loss in accuracy. However, a key limitation of DeeBERT is its reliance on static thresholds, which are sensitive to distribution shifts in the input data. This makes the method less robust when the data characteristics change over time or when applied to out-of-distribution inputs.

BranchyNet [2] applied a similar confidence-based approach but focused on Convolutional Neural Networks (CNNs). It achieved a significant speedup of 30–50% in

inference time. However, like DeeBERT, BranchyNet requires careful tuning of the thresholds, particularly when dealing with data distributions that are not represented in the training set, indicating a need for more dynamic thresholding methods that can adapt to new data distributions.

Patience-Based Methods

PABEE [3] introduces a patience-based early exiting strategy that exits the model when predictions across a series of consecutive layers are consistent. This method was shown to achieve about a 50% reduction in floating-point operations (FLOPS) on GLUE tasks, indicating substantial efficiency gains. However, the method is most effective on datasets with stable, predictable distributions. When the dataset undergoes significant distribution shifts, PABEE struggles, as the model may exit prematurely or fail to exit when needed. FastBERT [4] extended the ideas from PABEE, reporting a 45% efficiency gain in terms of inference speedup. The primary challenge of FastBERT, however, lies in the difficulty of selecting an appropriate value for the parameter t (the number of consecutive consistent predictions required), which can be very sensitive to the specific task or dataset.

Similarity-Based Methods

MuE [5] introduces a similarity-based method, where the decision to exit is based on the cosine similarity between the hidden representations of consecutive layers. This method provides a balance between speed and accuracy by reducing inference time by 30–45% on vision tasks, such as those in the ImageNet dataset. MuE works well when the network has learned meaningful representations, but it struggles when the data is noisy or when the representations do not clearly differentiate between different classes. Additionally, the choice of threshold for cosine similarity can significantly impact both the model's performance and efficiency.

SDN [6] extends this idea further, achieving a 20–40% speedup by applying a similar early exiting mechanism. SDN emphasizes the use of adaptive thresholds, which can adjust dynamically based on the input data. This adaptability makes SDN more robust to varying data distributions, offering potential improvements in scenarios where other early exiting methods might fail to achieve optimal performance.

3 Existing Gap

Multi-Armed Bandits in Optimization

Multi-Armed Bandit (MAB) algorithms have seen extensive application in the context of hyperparameter optimization, particularly for tasks like neural architecture search. Li et al. [7] used Upper Confidence Bound (UCB) algorithms to efficiently search

for optimal neural architectures, demonstrating the effectiveness of bandit-based approaches for model optimization. Contextual bandits [8], which extend traditional MAB methods to dynamic environments, have also been employed in personalized recommendations, where the model adapts to changing user preferences in real-time.

Thompson Sampling, a popular MAB technique, was applied by Wang et al. [9] to optimize edge inference tasks. Their approach achieved 35% efficiency gains in resource-constrained environments, such as edge computing, by dynamically adjusting the decision-making process based on the observed performance of different inference strategies. While MAB techniques have been successfully applied to various optimization problems, their use for optimizing early exiting thresholds remains an underexplored area, presenting an opportunity for further research.

4 Problem Setup

In transformer-based models like BERT, early exiting refers to the strategy of terminating inference at an intermediate layer instead of processing all layers, aiming to achieve faster inference while maintaining acceptable accuracy. The key challenge in early exiting methods lies in deciding *when to exit*, i.e., determining how confident the model is at intermediate layers about its prediction. A poor choice of exit criteria can either severely harm the model's predictive performance (if exiting too early) or fail to provide computational savings (if exiting too late).

Thus, the core of the early exit framework revolves around two critical components:

- **Quantifying Confidence at Intermediate Layers:** Developing a reliable method to assess how confident the model is in its prediction at a given intermediate layer.
- **Threshold Selection for Exiting:** Setting an appropriate threshold or rule based on the confidence measure to determine whether to exit or continue processing deeper layers.

There are multiple ways proposed in literature to quantify model confidence for early exiting. We consider the following three major types, inspired by techniques discussed in the literature (including the referenced Survey paper):

Confidence-Based Early Exit

Confidence-based early exiting strategies directly evaluate the model's prediction probability. A typical approach (used in models like DeeBERT) involves computing the maximum probability from the softmax output at a given exit classifier. If this maximum probability exceeds a pre-defined threshold α, the sample exits at that layer.

Confidence Metric:

$$Confidence = \max(Softmax(logits))$$

Exit Criterion: Exit if $Confidence > \alpha$.

Trade-off Control:

- Higher α: More conservative, later exits, higher accuracy but slower inference.
- Lower α: Earlier exits, faster inference but potentially lower accuracy.

Patience-Based Early Exit

Patience-based methods rely on the consistency of predictions across successive layers rather than raw confidence scores. As proposed in PABEE, if a model predicts the same label for t consecutive layers, it is deemed confident enough to exit.

Confidence Metric: Prediction stability over successive layers.

Exit Criterion: Exit if the same prediction is made across t consecutive layers.

Trade-off Control:

- Larger t: Requires more consistent predictions, leading to deeper layers and higher accuracy.
- Smaller t: Exits earlier, favoring efficiency.

Similarity-Based Early Exit

Similarity-based methods measure the semantic similarity between the hidden representations of consecutive layers. The assumption is that once the internal representations stabilize (i.e., stop changing significantly), further computation yields diminishing returns. The MuE framework uses cosine similarity between hidden states to drive exiting decisions.

Confidence Metric: Cosine similarity between hidden states of successive layers.

Exit Criterion: Exit if the similarity exceeds a threshold α.

Trade-off Control:

- Higher similarity thresholds: Exit sooner.
- Lower similarity thresholds: Continue deeper into the model.

Need for Adaptive Thresholds

Although thresholds can be set manually through hyperparameter tuning, this approach may not generalize well to different test datasets or distribution shifts. Ideally, thresholds should adapt automatically to achieve a desired balance between inference speed and model accuracy under varying conditions.

Moreover, the trade-off parameter set during training may not remain optimal during inference, especially when the data distribution differs. This motivates the need for adaptive threshold selection mechanisms rather than static thresholds.

Modeling Threshold Selection as a Multi-Armed Bandit Problem

Threshold selection in early exiting can naturally be modeled as a multi-armed bandit (MAB) problem. In a traditional MAB setup, an agent must choose among several actions (arms), each associated with an unknown reward distribution, to maximize its cumulative reward over time. Drawing this analogy, each possible threshold value (e.g., different levels of confidence, patience, or similarity) is considered an arm.

At each inference instance (i.e., each input sample or batch of samples), the agent selects a threshold (arm), applies it to decide early exiting, and observes a reward based on the trade-off between accuracy and efficiency (e.g., faster inference and correct prediction). Over time, the bandit algorithm learns which thresholds perform best under the current data distribution.

Specifically:

- **Arms:** Different candidate thresholds for early exit decisions (e.g., different values of α or t).
- **Reward:** A function of the prediction correctness and inference efficiency (e.g., negative of computation cost if accurate, heavy penalty if misclassified).
- **Policy:** Use bandit strategies such as ϵ-greedy, Upper Confidence Bound (UCB), or Thompson Sampling to balance exploration of new thresholds and exploitation of known good thresholds.

By framing threshold selection as a MAB problem, the model can automatically adapt to varying data distributions without requiring re-tuning of hyperparameters manually for each new setting.

5 Experimental Setup

Dataset

We use the `Stanford Sentiment Treebank (SST-2)` dataset from the GLUE benchmark for our experiments. The task involves binary sentiment classification of movie review sentences, where each sentence is labeled as either positive or negative. The dataset contains approximately 67,349 training samples and around 872 validation samples. Each sample consists of a sentence and a corresponding label, with 0 indicating negative sentiment and 1 indicating positive sentiment. The dataset is accessed via the Hugging Face `datasets` library. For preprocessing, sentences are tokenized using the `BertTokenizer` from the `bert-base-uncased` model. Inputs are padded or truncated to a maximum sequence length of 128 tokens to ensure consistent input size across samples. The processed data is converted into PyTorch tensors, comprising the fields: `input_ids`, `attention_mask`, and `label`.

Model

We employ the `bert-base-uncased` model as our base architecture for early exiting experiments. This model consists of 12 transformer layers with a hidden size of 768 and 12 attention heads, totaling approximately 110 million parameters. It is sourced from the Hugging Face Transformers library and is chosen for its widespread adoption and sufficient depth, which makes it suitable for studying early exiting strategies. To enable early exiting, we introduce additional exit classifiers at configurable layers, specifically at layers 3, 6, 9, and 12. Each early exit classifier is implemented as a linear layer mapping the [CLS] token's hidden state (of dimension 768) to a 2-dimensional output corresponding to the sentiment classes, following a dropout layer with a probability of 0.1.

6 Algorithm

Early Exit Methods

The experiments compare three early exit strategies for deciding when to terminate inference. The first strategy is **confidence-based** early exiting, as proposed in DeeBERT. In this method, the model exits when the maximum softmax probability from a classifier's output exceeds a predefined threshold. We evaluate 10 threshold values linearly spaced between 0.5 and 0.95. This approach follows the method described by Xin et al. (ACL 2020).

The second strategy is **patience-based** early exiting, inspired by PABEE. Here, the model exits when it produces consistent predictions across t consecutive classifiers. We consider patience values of 1, 2, and 3. This method is based on the approach

introduced by Zhou et al. (NeurIPS 2020).

The third strategy is **similarity-based** early exiting, derived from MuE. In this approach, the model exits when the cosine similarity between the hidden states of consecutive classifiers exceeds a given threshold. We explore 10 threshold values linearly spaced between 0.1 and 0.9. This method follows the work of Liu et al. (ACL 2020).

Threshold Optimization

To dynamically select the optimal thresholds for each early exit method, we utilize the Upper Confidence Bound (UCB) multi-armed bandit algorithm (other algorithms maybe considered during experiment). The purpose of this optimization is to balance the trade-off between maintaining high prediction accuracy and achieving efficient inference by exiting earlier when appropriate.

For the bandit setup, the arms correspond to the candidate thresholds or patience values. Specifically, the confidence-based method uses 10 thresholds between 0.5 and 0.95, the patience-based method uses patience values of 1, 2, and 3, and the similarity-based method uses 10 thresholds between 0.1 and 0.9.

The preliminary reward function we considered is given by:

$$Reward = Accuracy - \lambda \times \frac{l}{L}$$

where:

- $Accuracy \in \{0, 1\}$ indicates whether the model's prediction was correct,
- l is the index of the layer at which the model exited (lower values correspond to earlier exits),
- L is the total number of exit layers,
- λ is a weighting factor (set to 0.1) that controls the trade-off between accuracy and computational efficiency.

Other reward formulations may also be explored during the experiments to better capture different trade-offs between speed and accuracy, depending on empirical observations.

The UCB algorithm selects thresholds based on the following formula:

$$UCB(a) = \hat{\mu}_a + \sqrt{\frac{2 \log t}{n_a + 10^{-5}}}$$

where $\hat{\mu}_a$ is the empirical mean reward for arm a, n_a is the number of times arm a has been selected, and t is the total number of selections made. A small constant 10^{-5} is added for numerical stability.

This approach ensures a balance between exploration (trying thresholds that are less explored) and exploitation (favoring thresholds with high observed rewards).

Comparison Method

The experiment compares the three early exit strategies across three key evaluation metrics.

First, **Accuracy** measures the prediction quality of the model. A higher accuracy indicates better predictive performance; however, achieving higher accuracy may require the model to process deeper layers, potentially reducing inference efficiency.

Second, **Efficiency**, quantified by the *average exit layer*, reflects the computational cost. A lower average exit layer signifies earlier exits and hence more efficient inference by utilizing fewer layers of the model.

Third, **Robustness to Distribution Shift** evaluates how well each early exit method maintains its performance when the input data distribution is altered. Specifically, it compares the performance degradation in terms of accuracy and average exit layer between the original and the shifted test conditions. This metric assesses the reliability and adaptability of the early exit mechanisms under distributional changes.

Extensions

Several potential extensions can be explored if the time permits. One such extension involves **testing additional datasets** such as the Microsoft Research Paraphrase Corpus (MRPC) or the IMDb movie review dataset. These datasets would provide insights into the generalizability of the early exit methods across different domains, particularly when the nature of the text or task changes, such as moving from sentiment classification to paraphrase detection.

Another potential extension is to **incorporate latency or FLOPs (floating-point operations)** into the reward function, which would help capture the real-world computational cost of early exiting. By considering these factors, we can more accurately reflect the trade-off between accuracy and efficiency in terms of actual hardware execution time, making the framework more relevant to practical deployment scenarios.

Lastly, **experiments with different classifier architectures**, such as a simpler two-layer multi-layer perceptron (MLP), could be explored to evaluate the impact of model complexity on early exit performance. These alternative architectures might offer different trade-offs between model capacity, computational cost, and the effectiveness of early exits, providing further insights into optimizing early exit strategies.

References

References

[1] J. Xin et al., "DeeBERT: Dynamic Early Exiting for BERT," *arXiv preprint arXiv:2004.12993*, 2020.

[2] S. Teerapittayanon et al., "BranchyNet: Fast Inference via Early Exiting," in *NeurIPS*, 2016.

[3] W. Zhou et al., "PABEE: Patience-Based Early Exit," *arXiv preprint arXiv:2006.00823*, 2020.

[4] W. Liu et al., "FastBERT: Fast Inference with Early Exit," *arXiv preprint arXiv:2004.02178*, 2020.

[5] M. Phuong and C. Lampert, "MuE: Multi-task Early Exit," in *ICML*, 2021.

[6] Y. Kaya et al., "SDN: Shallow-Deep Networks," in *ICCV*, 2019.

[7] L. Li et al., "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization," *Journal of Machine Learning Research (JMLR)*, 2017.

[8] L. Li et al., "Contextual Bandits for Personalized Recommendation," in *Proceedings of WWW*, 2010.

[9] X. Wang et al., "Thompson Sampling for Edge Inference," *IEEE Transactions on Mobile Computing*, 2023.

[10] R. Schwartz et al., "CALDEE: Calibration-Driven Early Exiting," *arXiv preprint arXiv:2005.04562*, 2020.

[11] Anonymous, "Survey on Early Exiting in Deep Learning," Under Review, 2024.