Accelerating Large Language Model Inference with Dynamic Self-Speculative Decoding

Florian Valade[1]

[1] Université Gustave Eiffel, Fujitsu

florian.valade@email.com

Résumé

Cet article présente une approche modulaire pour accélérer l'inférence des modèles de langage de grande taille (LLMs) en ajoutant des têtes de sortie anticipée (« early exit heads ») aux couches intermédiaires du transformer. Chaque tête est entraînée de manière auto-supervisée pour imiter les prédictions du modèle principal, permettant ainsi d'interrompre le calcul dès qu'un seuil de confiance calibré est atteint. Nous évaluons plusieurs métriques de confiance et démontrons que l'entropie offre la séparation la plus fiable entre les prédictions correctes et incorrectes. Des expériences menées sur la suite de modèles Pythia (de 70M à 2,8B de paramètres) montrent que notre méthode réduit considérablement le coût d'inférence tout en maintenant la précision sur plusieurs benchmarks. Nous adaptons ensuite cette approche au décodage spéculatif en introduisant le « Dynamic Self-Speculative Decoding » (DSSD), qui atteint un taux d'acceptation de jetons 1,66× supérieur aux références LayerSkip ajustées manuellement, avec un réglage minimal des hyperparamètres.

Mots-clés

Early Exit, LLM Inference, Speculative Decoding, Acceleration

Abstract

This paper presents a modular approach to accelerate inference in large language models (LLMs) by adding early exit heads at intermediate transformer layers. Each head is trained in a self-supervised manner to mimic the main model's predictions, allowing computation to stop early when a calibrated confidence threshold is reached. We evaluate several confidence metrics and show that entropy provides the most reliable separation between correct and incorrect predictions. Experiments on the Pythia model suite (70M to 2.8B parameters) demonstrate that our method significantly reduces inference cost while maintaining accuracy across multiple benchmarks. We further adapt this approach to speculative decoding, introducing Dynamic Self-Speculative Decoding (DSSD), which achieves 1.66× higher token acceptance than manually-tuned LayerSkip baselines with minimal hyperparameter tuning.

Keywords

Early Exit, LLM Inference, Speculative Decoding, Acceleration

1 Introduction

Large language models (LLMs) have become central to advancing capabilities in natural language processing (NLP), delivering remarkable performance across a range of tasks. The trend towards scaling up these models correlates strongly with improved performance, understanding, and generality. This relationship has been formalized through empirical scaling laws, which demonstrate that model performance improves predictably with increased model size, dataset size, and compute budget [16, 13]. However, the computational cost associated with these larger models is substantial, often necessitating the use of powerful server infrastructure [22]. This not only limits local usability but also raises significant privacy concerns and requires considerable investment to scale in response to user demand. Solutions exist to reduce the computational demands of these models, but they often impact the model's performance by reducing its accuracy [41].

Despite their effectiveness, these models often operate inefficiently. The nature of language itself contributes to this inefficiency ; namely, not all tokens generated during the inference process contribute equally to the overall meaning or require the same level of computational resources. Some tokens are inherently simpler and can be predicted with high confidence early in the computation process, while others, contributing more significantly to the context or meaning, may require deeper processing.

In response to these challenges, we develop a method that can be easily integrated into existing pre-trained models to enhance their inference speed without extensive retraining. Our solution focuses on the strategic placement of early exit "heads" [23, 28] within the transformer layers of an LLM. These heads terminate the inference process when a calibrated confidence threshold is met, based on the complexity and predictability of the token being processed.

Our contributions are twofold :

1. We provide a detailed experimental study and modular framework for training and deploying early exit heads on top of LLMs. We analyze multiple training strategies, confidence metrics, and demonstrate scalability across model sizes from 70M to 2.8B parameters on the Pythia suite.

2. We adapt our early exit mechanism to speculative decoding, introducing **Dynamic Self-Speculative Decoding (DSSD)**, which achieves **1.66× higher token acceptance rates** than manually-tuned LayerSkip baselines while requiring minimal hyperparameter tuning—only a single accuracy threshold ϵ.

2 Related Work

Transformers [32] scaled into today's LLM families such as BERT, GPT-3, PaLM, LaMDA, LLaMA and OPT [7, 3, 6, 29, 30, 39], powering vision [8], speech [21] and multimodal models. Their billion-parameter footprints, however, make every token generation costly. Static compression—quantisation, pruning and distillation [25, 27, 38, 11, 26, 1]—slashes model size but still expends identical compute on easy and hard inputs. Early-exit methods address this imbalance by attaching lightweight classifiers to intermediate layers and halting computation once a confidence criterion is met.

In computer vision, BranchyNet [28] and MSDNet [14] or EERO methodology [31] established the two key components still used today : deeply supervised branches and an entropy-based exit rule. Transferring the idea to Transformers, DeeBERT and *The Right Tool* calibrated softmax confidence to save up to 5× latency with negligible loss [35, 24]. FastBERT distilled the final head into earlier exits [20], PABEE demanded k consecutive agreeing predictions instead of a threshold [40], and BERxiT learned an explicit when-to-exit module while alternating fine-tune schedules [36]. Skip/SmartBERT added trainable gates that may bypass whole layers, combining skipping with exiting for 2–4× cheaper inference [18, 5].

Early exit for sequence-to-sequence generation is newer. Depth-Adaptive Transformers first applied exits to neural MT [9]; FREE and DEED refined token-level uncertainty estimates [34, 37]. Recent evidence shows exits remain effective inside 13 B-parameter LLaMA-2 and GPT-J [19], suggesting scalability to modern LLMs.

Speculative decoding [17, 4] accelerates LLM inference by using a smaller draft model to generate candidate tokens, which are then verified in parallel by the target model. LayerSkip [10] combines this with early exits, using intermediate layers as draft models. However, LayerSkip requires exhaustive hyperparameter search over both head selection and speculation length. Our DSSD method addresses this limitation by adaptively selecting exit layers based on calibrated confidence.

Our work inherits the plug-and-play nature of DeeBERT-style branches, borrows the self-supervised signal of FastBERT, and scales token-level exits to different architecture sizes. We train exit heads without extra data—using the model's own probabilities—then calibrate a single threshold on a held-out set à la conformal prediction [33]. Unlike layer-skipping approaches, the backbone weights remain frozen, guaranteeing monotonic accuracy with deeper computation.

3 Methodology

This section details our methodology for integrating and utilizing early exits within large language models (LLMs) to enhance computational efficiency during inference. The approach is designed to be generalizable and, while we demonstrate its application using the Pythia suite, it is applicable to any multi-layered transformer model. This adaptability ensures that our methodology can be leveraged across a broad spectrum of modern LLMs, enhancing their usability without requiring significant modifications to their underlying architectures.

3.1 Definitions and Notation

We introduce here the main notations used throughout the paper.

Vocabulary. Let $\mathcal{V}$ denote the vocabulary, a finite set of tokens :

$$\mathcal{V} = \{v_1, v_2, \dots, v_{|\mathcal{V}|}\},$$

where $|\cdot|$ denotes cardinality of sets.

Dataset. We consider a dataset $\mathcal{D}$ of N examples, where each example is a sequence of tokens :

$$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}, \quad \mathbf{x}^{(i)} = (x_1^{(i)}, \dots, x_{L_i}^{(i)}), \quad x_j^{(i)} \in \mathcal{V}.$$

A subset $\mathcal{D}_{\text{cal}} \subset \mathcal{D}$ is reserved as a calibration set, the rest being used for training.

Each sequence $\mathbf{x}^{(i)}$ can have a variable length L_i. The total number of tokens in the dataset is $M = \sum_{i=1}^{N} L_i$, with $M > N$ in general. Similarly, the calibration set $\mathcal{D}_{\text{cal}}$ contains N_{cal} examples and a total of M_{cal} tokens.

Language model. A large language model (LLM) is a function f_θ parameterized by θ, mapping a sequence of tokens to a sequence of probability distributions over the vocabulary :

$$f_\theta : (x_1, \dots, x_L) \mapsto (\mathbf{p}_1, \dots, \mathbf{p}_L), \quad \mathbf{p}_t \in \Delta^{|\mathcal{V}|},$$

where $\Delta^{|\mathcal{V}|}$ is the probability simplex over $\mathcal{V}$.

Objective. Given an input sequence $\mathbf{x} = (x_1, \dots, x_L) \in \mathcal{V}^L$, the target sequence is $\mathbf{y} = (y_1, \dots, y_L) \in \mathcal{V}^L$ with $y_t = x_{t+1}$ (i.e., the next-token prediction task).

Early exit heads. We introduce K early exit heads, each defined as a function h_k (with its own parameters, but sharing the backbone with f_θ) that maps a sequence of L tokens to a sequence of probability distributions :

$$h_k : (x_1, \dots, x_L) \mapsto (\mathbf{p}_{k,1}, \dots, \mathbf{p}_{k,L}), \quad \mathbf{p}_{k,t} \in \Delta^{|\mathcal{V}|}.$$

All early exit heads share the same backbone, so most of their parameters are shared with the main model.

Confidence metric. A confidence metric is a function $c : \Delta^{|\mathcal{V}|} \to \mathbb{R}$ that assigns a real-valued confidence score to a probability vector (e.g., $c(\mathbf{p}) = \max_j p_j$).

Accuracy threshold ϵ. The parameter $\epsilon \in [0, 1]$ controls the minimal desired accuracy for early exit decisions. For each early exit head, a prediction is only output if its confidence metric exceeds a calibrated threshold corresponding to at least ϵ empirical accuracy on a held-out calibration set. Lowering ϵ increases speedup at the cost of potential accuracy loss, while higher ϵ enforces stricter correctness guarantees.

The above definitions give a precise, token-level view of the model outputs and their interaction with early-exit logic; they will serve as the foundation for the training objectives, calibration techniques, and inference algorithms described in the following sections.

3.2 Implementation Details

To enhance the inference efficiency of large language models, we incorporate early exit "heads" into a pre-existing model, in this instance, the Pythia suite [2]. These heads are implemented at regular intervals along the network. Structurally, each head is a simple multi-layer perceptron (MLP) identical to the final classification head of the model. Each of these head takes as input hidden features from a transformer block inside the model.

Placement of early–exit heads. Let the transformer backbone consist of L stacked blocks, indexed $\mathcal{I} = \{1, 2, \ldots, L\}$, and let K denote the desired number of early–exit heads. To distribute the heads uniformly while keeping the final classification layer untouched, we attach head $k \in \{1, \ldots, K\}$ to the block whose index is

$$\ell_k = \left\lfloor \frac{k}{K+1} L \right\rfloor, \qquad k = 1, \ldots, K. \tag{1}$$

Equation (1) simply divides the layer indices into $K + 1$ equal segments and selects the endpoint of each segment (rounded down to the nearest integer) as a branch location. In our experiments we set $K = 4$; hence the heads are inserted at $\ell_k \in \left\{ \lfloor \frac{L}{5} \rfloor, \lfloor \frac{2L}{5} \rfloor, \lfloor \frac{3L}{5} \rfloor, \lfloor \frac{4L}{5} \rfloor \right\}$.

For the implementation of these heads, we experimented with two initialization strategies : initializing the heads from scratch and copying the final classification head in order to fine-tune it. The difference between the two are analyzed in Section 4. With the architectural choices established, we next describe the data and objectives used to train the early exit heads.

3.3 Training data and objectives

Corpus. All auxiliary heads are trained on MINIPILE [15], a 6-GB stratified subset of the original 825-GB THEPILE-DEDUPLICATED [12] dataset that was employed for pretraining the PYTHIA backbone [2]. Using the same data distribution avoids the distribution-shift issues that often appear when auxiliary classifiers are fitted post-hoc.

Losses. For the whole predicted sequence $\hat{\mathbf{y}}$, we can compute different losses depending on the objectives. We define the cross-entropy loss and the Kullback–Leibler divergence between the predicted probability distribution $\hat{y}$ and the ground-truth labels y as follows :

$$\mathcal{L}_{\text{CE}}(\mathbf{y}, \hat{\mathbf{y}}) = -\frac{1}{L} \sum_{t=1}^{L} \sum_{v \in \mathcal{V}} \mathbf{1}_{\{\mathbf{y}^{(t)} = v\}} \log(\hat{\mathbf{y}}_v^{(t)}),$$

$$\mathcal{L}_{\text{KL}}(\hat{\mathbf{y}} || \mathbf{y}) = \frac{1}{L} \sum_{t=1}^{L} \sum_{v \in \mathcal{V}} \hat{\mathbf{y}}_v \log \frac{\hat{\mathbf{y}}_v}{\mathbf{y}_v}.$$

In the context of self-supervised training, we consider the Kullback-Leibler divergence where the target $\mathbf{y}$ is replaced by the output of the main model f_θ. Thus, the loss $\mathcal{L}_{\text{KL}}$ becomes :

$$\mathcal{L}_{\text{KL}}(\hat{\mathbf{y}} || f_\theta) = \frac{1}{L} \sum_{t=1}^{L} \sum_{v \in \mathcal{V}} \hat{\mathbf{y}}_v \log \frac{\hat{\mathbf{y}}_v}{f_\theta(v)}$$

The cross-entropy loss $\mathcal{L}_{\text{CE}}$ is used when considering a supervised training objective. It matches how the main model is trained and can be used to train the heads as well. The Kullback–Leibler divergence $\mathcal{L}_{\text{KL}}$ is used when considering a self-supervised training objective. It encourages the early exit heads to mimic the full probability distribution of the main model, which can be useful for improving the performance of the early exits. From this, we consider three training objectives obtained from the above building blocks :

- **Supervised** ($\mathcal{L}_{\text{sup}}$). Purely next-token cross-entropy against ground-truth labels, that is, $\mathcal{L}_{\text{sup}} = \mathcal{L}_{\text{CE}}$;
- **Self-supervised** ($\mathcal{L}_{\text{self}}$). KL divergence that encourages each head to mimic the teacher's full probability distribution, that is, $\mathcal{L}_{\text{self}} = \mathcal{L}_{\text{KL}}$;
- **Hybrid** ($\mathcal{L}_{\text{hyb}}$). The sum of the two losses above, weighted by coefficients $\alpha \in (0, 1)$, that is, $\mathcal{L}_{\text{hyb}} = \alpha \mathcal{L}_{\text{CE}} + (1 - \alpha) \mathcal{L}_{\text{KL}}$.

As reported in Section 4, the self-supervised objective ($\mathcal{L}_{\text{self}}$) yields the most faithful approximation of the teacher's behavior while preserving calibration, and therefore constitutes our default choice for all subsequent experiments.

3.4 Calibration and Inference

After training the early exit heads, the next crucial step involves calibrating and using these heads during model inference. This process is divided into two main stages : calibration of the confidence thresholds and the application of these thresholds during inference.

3.4.1 Calibration of Confidence Thresholds

In our approach, we evaluated three different confidence metrics for calibrating early exit thresholds : (1) maximum probability, (2) entropy of the predicted distribution, and (3) the difference between the top two probabilities ("breaking ties"). To determine which metric best separates correct from incorrect predictions, we analyzed their ability to discriminate between right and wrong outputs using ROC (Receiver Operating Characteristic) curves (Figure 1).

After training, we computed the ROC curves for each metric by plotting the true positive rate against the false positive rate as the threshold varies, using the predictions from

the early exit heads. This analysis was performed across six Pythia models, ranging from 70M to 2.8B parameters. The results are summarized in Figure 1, which displays the ROC curves for all metrics and models.

Our findings consistently show that entropy outperforms the other metrics in every case, achieving a higher area under the curve (AUC) regardless of model size. This indicates that entropy provides a more reliable separation between correct and incorrect predictions, making it the most effective metric for threshold calibration in our early exit framework. Furthermore, we observe that the AUC of the entropy metric tends to increase with model size, suggesting that early exit mechanisms become even more effective as the underlying model grows larger.

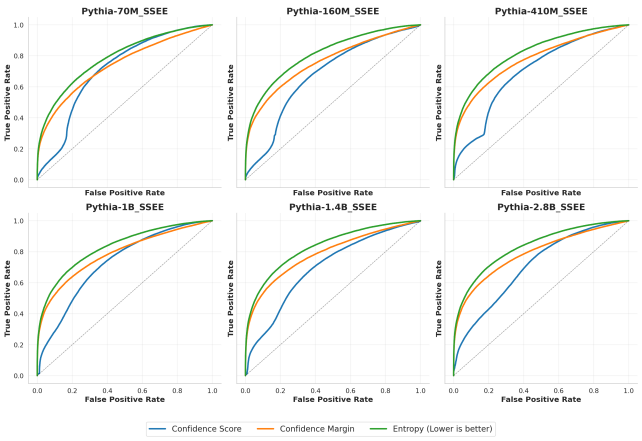

FIGURE 1 – ROC curves for three confidence metrics across six Pythia models (70M to 2.8B). Entropy consistently achieves the highest AUC.

To calibrate the confidence thresholds, we perform a full epoch over our calibration dataset. For each example in the calibration set and for each early exit head, we compute the head output for every token in the sequence, then apply the confidence metric c to each token output, and store all these scores in a global vector $\mathbf{c}_k \in \mathbb{R}^{M_{\text{cal}}}$, where M_{cal} is the total number of tokens in the calibration dataset.

Correspondingly, let $\mathbf{t}_k$ be a binary vector of length M_{cal} where each element corresponds to the correctness of the prediction associated with the respective element in $\mathbf{c}_k$. Specifically, $t_{k,j}$ is 1 if the j^{th} prediction at head k matches the j^{th} prediction of the underlying model $f_\theta(\mathbf{x})$, otherwise 0 :

$$\begin{aligned} t_{k,j} &= \mathbb{1}_{\{\arg\max \mathbf{p}_{k,j} = \arg\max \mathbf{p}_{\theta,j}\}} \\ &= \begin{cases} 1 & \text{if } \arg\max(\mathbf{p}_{k,j}) = \arg\max(\mathbf{p}_{\theta,j}) \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{2}$$

To determine the confidence threshold for each early exit head, we utilize the calibration set to empirically estimate the relationship between the confidence metric and prediction correctness. Specifically, for each head k, we sort the calibration metric values in ascending order. Given a user-specified confidence level $\epsilon \in [0, 1]$, which represents the minimum desired proportion of correct predictions above the threshold, we identify the smallest metric value such that the proportion of correct predictions among all samples with higher (or equal) metric values is at least ϵ. This procedure ensures that, during inference, predictions made with a confidence metric exceeding the threshold are correct with probability at least ϵ, thereby providing a principled trade-off between computational efficiency and predictive accuracy. Formally, the threshold for each head is defined as follows.

Recall that $\mathbf{c}_k$ and $\mathbf{t}_k$ denote, respectively, the vectors of confidence metric values and correctness indicators for head k, as defined in equation (2). To proceed, we jointly sort these vectors in ascending order according to the values in $\mathbf{c}_k$, resulting in ordered sequences $\mathbf{c}_k = (c_{k,1}, c_{k,2}, \ldots, c_{k,M_{\text{cal}}})$ and $\mathbf{t}_k = (t_{k,1}, t_{k,2}, \ldots, t_{k,M_{\text{cal}}})$ such that $c_{k,1} \leq c_{k,2} \leq \ldots \leq c_{k,M_{\text{cal}}}$.

We then define $\hat{j}$ as the smallest index for which the proportion of correct predictions among all samples with confidence at least $c_{k,\hat{j}}$ meets or exceeds the target confidence level ϵ :

$$\frac{\sum_{i=\hat{j}}^{M_{\text{cal}}} t_{k,i}}{M_{\text{cal}} - \hat{j} + 1} \geq \epsilon.$$

Then, let the threshold τ_k be defined as :

$$\tau_k = c_{k,\hat{j}}.$$

In other words, τ_k is the value of the confidence metric at the index $\hat{j}$, where $\hat{j}$ is the smallest index such that the proportion of correct prediction in the remaining samples is at least ϵ.

3.4.2 Inference Process

After establishing the confidence thresholds for each early exit head through the calibration process, the model is then ready to utilize these thresholds during the inference phase to efficiently process new inputs.

During inference, each input $\mathbf{x}$ is sequentially processed through the model's layers, with the possibility of early termination at any of the early exit heads h_k. For each head $k \in \{1, 2, \ldots, K\}$, we compute :

$$\begin{aligned} \mathbf{p}_k &= h_k(\mathbf{x}), \\ c_k &= c(\mathbf{p}_k). \end{aligned}$$

We then return $\mathbf{p}_k$ as output from the model if :

$$c_k \geq \tau_k.$$

If no head satisfies this inequality, we return $\mathbf{p}_\theta$.

This calibrated and threshold-driven early exit mechanism allows the model to balance efficiency with accuracy, ensuring that resource-intensive computations are only performed when necessary.

4 Experiments and Results

We organize our experimental evaluation into two complementary parts [1]. First, we evaluate calibrated early exits on

1. 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 used in experiments to train and evaluate can be found under : https ://anonymous.4open.science/r/BranchyLLM-B870

standard inference tasks using the Pythia suite (70M–2.8B parameters). We describe the training process for early exit heads and analyze how accuracy-speedup trade-offs scale across model sizes on benchmark tasks. Second, we extend our method to speculative decoding, introducing Dynamic Self-Speculative Decoding (DSSD), which eliminates the need for manual hyperparameter tuning. Throughout our experiments, we use Pythia as a representative example that allows us to scale according to hardware capabilities, though our approach is theoretically applicable to any language model architecture.

4.1 Training

In the training phase, we systematically compared the three loss functions described in Section 3.2 : supervised (cross-entropy), self-supervised (KL divergence to the main model's output), and a hybrid of both. Our primary goal was to select a training objective that enables early exit heads to best approximate the main model's predictions while also providing meaningful uncertainty estimates.

We found that the self-supervised loss, which encourages each early exit head to mimic the output distribution of the main model, consistently led to the best alignment with the main model's predictions. This loss not only matches the output probabilities but also preserves the calibration properties necessary for reliable early exits.

Additionally, we experimented with two initialization strategies for the early exit heads : (1) copying the weights from the main model's final classification head (lm_head), and (2) random initialization. When the early exit heads are initialized by copying the lm_head, they start with a lower loss and initially perform better. However, as training progresses, the randomly initialized heads quickly overtake the copied ones in both loss and performance.

A key observation is that heads copied from the lm_head tend to be overconfident, even when making incorrect predictions. This results in low entropy outputs and a lack of meaningful uncertainty, which is detrimental for early exit decisions. In contrast, randomly initialized heads, when trained with the self-supervised loss, develop a better notion of uncertainty, producing higher entropy outputs when unsure. This property is crucial for effective early exit mechanisms, as it allows the confidence metric to reliably distinguish between correct and incorrect predictions.

For our training experiments, we launch training with the self-supervised loss described above on all Pythia models from 70M to 2.8B parameters. We use a learning rate of 5×10^{-5} and train on 500,000 examples sampled from the MiniPile dataset. Four early exit heads are added at the locations described in the Methodology section.

4.2 Inference

For the inference evaluation, each model is evaluated with different values of the confidence threshold parameter ϵ, ranging from 0.5 to 1.0 in increments of 0.05. This systematic sweep allows us to analyze the trade-off between computational savings and predictive accuracy.

We follow the same set of benchmarks as the original Pythia paper : WSC, Winogrande, SciQ, PIQA, LogiQA, LAMBADA_OpenAI, ARC Easy, and ARC Challenge. For each benchmark and each value of ϵ, we report both the benchmark score and the mean output layer, translated to a speedup percentage.

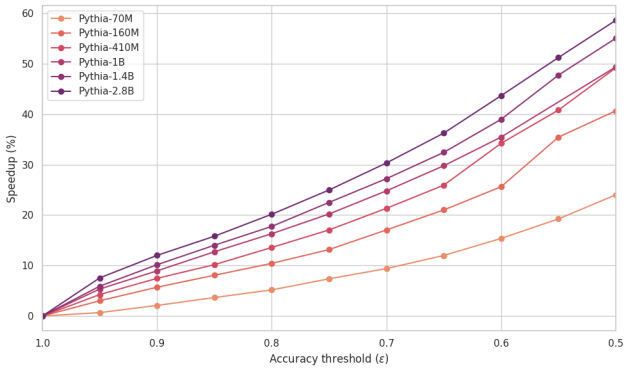

FIGURE 2 – Speedup vs. confidence threshold ϵ for different Pythia model sizes.

TABLE 1 – Benchmarks results for Pythia-2.8B. Each line corresponds to a value of ϵ ranging from 0.5 to 1.0 in steps of 0.05 (top to bottom), with the bottom line being the baseline ($\epsilon = 1$). More benchmarks available in the appendix.

ϵ	Speedup	winogrande	piqa
0.50	58.6%	0.541 ± 0.014	0.615 ± 0.011
0.55	51.2%	0.564 ± 0.014	0.635 ± 0.011
0.60	43.7%	0.565 ± 0.014	0.643 ± 0.011
0.65	36.2%	0.565 ± 0.014	0.668 ± 0.011
0.70	30.3%	0.586 ± 0.014	0.681 ± 0.011
0.75	25.0%	0.579 ± 0.014	0.690 ± 0.011
0.80	20.2%	0.574 ± 0.014	0.707 ± 0.011
0.85	15.8%	0.579 ± 0.014	0.719 ± 0.010
0.90	12.0%	0.571 ± 0.014	0.730 ± 0.010
0.95	7.6%	0.572 ± 0.014	0.733 ± 0.010
1.00	0.0%	0.571 ± 0.014	0.742 ± 0.010

The results of the benchmark on Pythia-2.8B are shown in Table 1. The results for the other model sizes are provided in the appendix.

Our results show that for some benchmarks, such as Winogrande and WSC, the reduction in computation has little to no effect on accuracy, even with aggressive early exiting. However, some benchmarks experience a drop in performance when using very aggressive thresholds (lower ϵ). Despite this, for moderate values of ϵ, the benchmark scores remain close to those of the main model, while achieving speedups between 10% and 20% depending on the model size.

Interestingly, we observe that smaller models in the Pythia suite, such as the 70M and 160M parameter variants, can even show improvements on certain benchmarks when early exits are used aggressively. This suggests that early exit mechanisms may help regularize smaller models or mitigate overfitting in some cases.

Beyond benchmark results, we also observe that the speedup achieved by early exit increases with the size of the

model. Larger models benefit more from early exit, with greater computational savings for the same threshold values. This effect may be explained by several factors : (1) the relative size of the early exit heads becomes less significant as model size increases, and (2) the intermediate features in larger models may provide better representations, enabling more accurate early predictions. This trend is illustrated in Figure 2, which shows the speedup as a function of the threshold for all model sizes.

All detailed tables of benchmark scores and speedup for each model and threshold are provided in the appendix. While our results demonstrate the effectiveness of early exits, it is important to consider the limitations and broader impacts of this approach.

To address the accuracy drops observed with aggressive early exiting while maintaining computational efficiency, we adapt our calibrated early exit mechanism to speculative decoding. This approach allows us to achieve significant speedups without compromising the final output quality, as the full model verifies all predictions.

4.3 Dynamic Early Exit for Speculative Decoding

Having established the effectiveness of calibrated early exits for single-token prediction, we now extend our method to *speculative decoding*—a technique where a faster draft model proposes multiple tokens that are then verified in parallel by the full model. This approach, exemplified by LayerSkip [10], can significantly accelerate autoregressive generation. However, LayerSkip requires practitioners to manually tune two coupled hyperparameters : (i) which intermediate head/layer to use for drafting, and (ii) how many tokens to speculate per round. Finding optimal settings often requires exhaustive grid search across prompts and tasks, with 24 configurations tested in our experiments.

Our key contribution is to replace these two discrete knobs with a *single continuous accuracy threshold* $\epsilon \in [0, 1]$, leveraging the same calibrated entropy-based exit mechanism from Section 3.4.1. The key insight is that per-token confidence naturally determines both *where* to exit (head selection) and *when* to stop drafting (adaptive speculation length). This unified approach, which we call **Dynamic Self-Speculative Decoding (DSSD)**, eliminates manual tuning while achieving superior acceptance rates : on Pythia-2.8B, DSSD reaches **88.8% acceptance at** $\epsilon = 0.9$ compared to LayerSkip's best of 53.6%, a **1.66× improvement**.

4.3.1 DSSD Algorithm

DSSD extends the inference procedure from Section 3.4.2 to multi-token drafting :

Drafting phase. Starting from the current context, we evaluate early-exit heads sequentially from shallowest to deepest. For each position, we select the first head k whose entropy $H(\mathbf{p}_k) < \tau_k(\epsilon)$ falls below its calibrated threshold. We append the predicted token to a draft buffer and continue until : (i) the buffer reaches a preset limit (e.g., 32 tokens), (ii) an end-of-sequence token is generated, or (iii) no head meets the confidence threshold, triggering a fallback to the full model. This produces a variable-length draft sequence whose length adapts to local token difficulty.

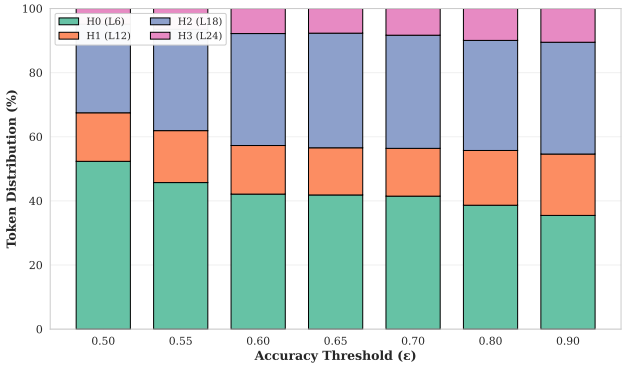

FIGURE 3 – Adaptive head selection : token distribution across heads H0(L6), H1(L12), H2(L18), H3(L24) at different accuracy thresholds ϵ, where notation $H_i(L_j)$ indicates head i at layer j. As ϵ decreases, the system shifts from primarily using H0 to a balanced mix across all heads.

Verification phase. We pass the entire draft buffer through the full model in a single forward pass, obtaining target predictions for all positions simultaneously. We then compare each drafted token with the corresponding full-model prediction : accepted tokens are committed to the output, while the first mismatch triggers an immediate rewrite with the full-model token. This produces an *atomic commit* of the accepted prefix. The mean number of accepted tokens per verification round acts as an implicit, adaptive speculation budget.

Comparison to LayerSkip. In LayerSkip, practitioners fix a single head (e.g., layer 6) and a speculation length (e.g., 10 tokens) for all contexts. Our method dynamically chooses the head on a per-token basis and automatically adjusts the effective speculation length through the acceptance mechanism, adapting to context difficulty without manual intervention.

4.3.2 Experimental Design

We evaluate DSSD on Pythia-2.8B with 200-token greedy decoding, comparing :

- **DSSD (ours)** : Accuracy sweep $\epsilon \in \{0.5, 0.55, 0.6, 0.65, 0.7, 0.8, 0.9\}$ (7 levels).
- **LayerSkip (fixed)** : Exhaustive grid search over 4 heads × 6 speculation lengths (24 configurations).

Experiments use prompts sampled from our dataset to test the method across diverse topics.

4.3.3 Results and Analysis

Superior acceptance with lower overhead. DSSD at $\epsilon = 0.9$ achieves **88.8% acceptance rate**—1.66× better than LayerSkip's best (53.6%). This improvement translates directly to reduced wasted computation : DSSD discards only 8 tokens versus LayerSkip's 128 tokens, a **14× reduction** in waste. Figure 3 illustrates this adaptive behavior.

Understanding layer concentration. An important observation from Figure 3b is that mean exit layers concentrate

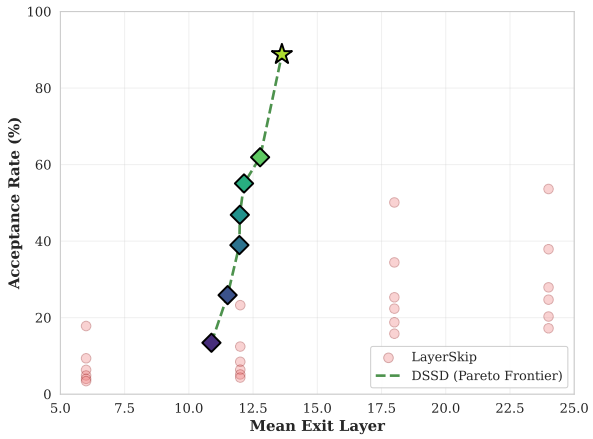

FIGURE 4 – Acceptance rate vs mean exit layer during drafting across all configurations. The dashed green line traces DSSD's frontier (diamonds), while LayerSkip configurations (circles) are dominated. The star highlights DSSD at $\epsilon = 0.9$, achieving 88% acceptance with only 13.6 mean exit layers.

in a relatively narrow range (10.9–13.6 across all ϵ values), spanning only 2.74 layers despite having 4 discrete exit points at [6, 12, 18, 24]. This concentration is **expected and optimal** rather than anomalous : (i) only 4 discrete exit points naturally limit the range, (ii) calibrated thresholds steer tokens toward heads H1-H2 (layers 12-18) as these offer the best efficiency-quality trade-off, and (iii) the narrow span represents successful optimization—the system has identified that most tokens benefit from moderate depth.

4.3.4 Summary

Our dynamic early-exit controller unifies head selection and speculation length into a single accuracy threshold ϵ, eliminating exhaustive hyperparameter tuning. Across 31 configurations tested on Pythia-2.8B, our method demonstrates :

- **Superior acceptance** : 1.66× higher than best LayerSkip (88.8% vs 53.6%)
- **Reduced waste** : 14× fewer discarded tokens (9 vs 128)
- **Earlier exit** : 10.4 layers shallower on average (13.6 vs 24.0)
- **Zero tuning** : Single threshold vs 2D grid search

As model size increases and exhaustive search becomes prohibitive, the automatic adaptation to token difficulty positions DSSD as a scalable, user-friendly alternative to fixed speculative strategies.

5 Limitations and Potential Impacts

While our early exit approach enables acceleration of large language models, the speedup gains without DSSD can remain modest when aiming to preserve the original LLM's accuracy. However, we observe that for certain benchmarks, early exits do not lead to any noticeable degradation in accuracy, even with significant acceleration.

Another important trend highlighted by our experiments is that the speedup obtained through early exits tends to increase with model size. Larger models appear to benefit more from this mechanism, both in terms of computational savings and in the stability of their predictions under early exit. Nevertheless, due to our limited computational resources, we were unable to train and evaluate early exit models on the largest architectures available. Confirming and further exploring this trend would require access to greater computing power.

6 Conclusion

In this work, we introduced a modular early exit mechanism for large language models, allowing inference to terminate early based on calibrated confidence metrics. Our approach is easy to integrate into existing transformer architectures and does not require retraining the backbone model. Through extensive experiments on the Pythia suite, we demonstrated that early exits can provide significant inference speedups, especially for larger models, while maintaining high accuracy on several benchmarks.

Furthermore, we introduced Dynamic Self-Speculative Decoding (DSSD), which integrates calibrated heads into a speculative decoder to achieve 1.66× higher token acceptance rates compared to manually-tuned LayerSkip baselines. The zero-tuning property and automatic adaptation to token difficulty make DSSD particularly attractive for practical deployment.

Overall, our findings suggest that exploiting the natural variability in token difficulty is a promising direction for accelerating large language models. We encourage the community to build upon this work, exploring new strategies and applications to enable efficient and scalable deployment of LLMs in real-world, resource-constrained environments.

Références

[1] Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jing Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. BinaryBERT : Pushing the Limit of BERT Quantization, July 2021. arXiv :2012.15701 [cs].

[2] Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia : A suite for analyzing large language models across training and scaling, 2023.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In *Advances in Neural Information*

Processing Systems, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[4] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling, 2023.

[5] Kehan Chen, Jinchao Wang, Xun Liu, et al. SmartBERT : Dynamic layer skipping and early exiting for efficient transformer inference. In *Proc. of IJCAI*, 2023.

[6] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. PaLM : Scaling Language Modeling with Pathways, October 2022. arXiv :2204.02311 [cs].

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *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)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words : Transformers for Image Recognition at Scale. *arXiv :2010.11929 [cs]*, October 2020. arXiv : 2010.11929.

[9] Maha Elbayad, Laurent Besacier, and Jakob Verbeek. Depth-adaptive transformer. In *Proc. of ICLR*, 2020.

[10] Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed Aly, Beidi Chen, and Carole-Jean Wu. Layerskip : Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, page 12622–12642. Association for Computational Linguistics, 2024.

[11] Angela Fan, Edouard Grave, and Armand Joulin. Reducing Transformer Depth on Demand with Structured Dropout, September 2019. arXiv :1909.11556 [cs, stat].

[12] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The Pile : An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv :2101.00027*, 2020.

[13] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, and et al. Training compute-optimal large language models. *arXiv preprint arXiv :2203.15556*, 2022.

[14] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. Multi-scale dense networks for resource efficient image classification, 2018.

[15] Jean Kaddour. The minipile challenge for data-efficient language models. *arXiv preprint arXiv :2304.08442*, 2023.

[16] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv :2001.08361*, 2020.

[17] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding, 2023.

[18] Zhengxin Li, Shiyang Shen, Yichong Xu, et al. SkipBERT : Skipping layers for efficient BERT inference. In *Proc. of ACL*, 2022.

[19] Tianyi Liu, Shuhe Ren, Hao Zhou, et al. Early exit is a natural capability of transformer-based language models. *arXiv preprint arXiv :2402.00000*, 2024.

[20] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Haotang Deng, and Qi Ju. Fastbert : a self-distilling bert with adaptive inference time, 2020.

[21] Alec Radford, Jong Wook Jeong, Jack Clark, et al. Robust speech recognition via large-scale weak supervision. *arXiv preprint arXiv :2302.04200*, 2023.

[22] Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. From Words to Watts : Benchmarking the Energy Costs of Large Language Model Inference, October 2023. arXiv :2310.03003 [cs].

[23] Simone Scardapane, M. Scarpiniti, E. Baccarelli, and A. Uncini. Why should we add early exits to neural

networks ? *Cognitive Computation*, 12 :954 – 966, 2020.

[24] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. The right tool for the job : Matching model complexity to instance difficulty. In *Proc. of ACL*, 2020.

[25] Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. Q-BERT : Hessian Based Ultra Low Precision Quantization of BERT, September 2019. arXiv :1909.05840 [cs].

[26] Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient Knowledge Distillation for BERT Model Compression, August 2019. arXiv :1908.09355 [cs].

[27] Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. MobileBERT : a Compact Task-Agnostic BERT for Resource-Limited Devices, April 2020. arXiv :2004.02984 [cs].

[28] Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. Branchynet : Fast inference via early exiting from deep neural networks, 2017.

[29] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Vincent Zhao, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Pranesh Srinivasan, Laichee Man, Kathleen Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Croak, Ed Chi, and Quoc Le. LaMDA : Language Models for Dialog Applications, February 2022. arXiv :2201.08239 [cs].

[30] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA : Open and Efficient Foundation Language Models, February 2023. arXiv :2302.13971 [cs].

[31] Florian Valade, Mohamed Hebiri, and Paul Gay. Eero : Early exit with reject option for efficient classification with limited budget, 2024.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[33] 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.

[34] Jing Wang, Hao Chen, Renqian He, et al. FREE : Fast, reliable early-exit transformers for efficient autoregressive generation. In *Proc. of EMNLP*, 2023.

[35] Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. DeeBERT : Dynamic Early Exiting for Accelerating BERT Inference, April 2020. arXiv :2004.12993 [cs].

[36] Ji Xin, Raphael Tang, and Jimmy Lin. BERxiT : Early exiting strategies for BERT. In *Proc. of EACL*, 2021.

[37] Yiming Yang, Yi Zheng, Xiaobo Li, et al. DEED : Dynamic early exiting for efficient decoding of large language models. In *Findings of NAACL*, 2024.

[38] Z. Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant : Efficient and affordable post-training quantization for large-scale transformers. *Neural Information Processing Systems*, 2022.

[39] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. OPT : Open Pre-trained Transformer Language Models, June 2022. arXiv :2205.01068 [cs].

[40] Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. BERT Loses Patience : Fast and Robust Inference with Early Exit, October 2020. arXiv :2006.04152 [cs].

[41] Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A Survey on Model Compression for Large Language Models, September 2023. arXiv :2308.07633 [cs].