Switchable Decision: Dynamic Neural Generation Networks

Shujian Zhang [1] **Korawat Tanwisuth** [1] **Chengyue Gong** [1] **Pengcheng He** **Mingyuan Zhou** [1]

Abstract

Auto-regressive generation models achieve competitive performance across many different NLP tasks such as summarization, question answering, and classifications. However, they are also known for being slow in inference, which makes them challenging to deploy in real-time applications. We propose a switchable decision to accelerate inference by dynamically assigning computation resources for each data instance. Automatically making decisions on where to skip and how to balance quality and computation cost with constrained optimization, our dynamic neural generation networks enforce the efficient inference path and determine the optimized trade-off. Experiments across question answering, summarization, and classification benchmarks show that our method benefits from less computation cost during inference while keeping the same accuracy. Extensive experiments and ablation studies demonstrate that our method can be general, effective, and beneficial for many NLP tasks.

1. Introduction

Large-scale pre-trained language models such as BART (Lewis et al., 2019)

(Lewis et al., 2019) have demonstrated a significant performance gain to the natural language processing (NLP) community but generally come with the cost of a heavy computational burden. Besides pre-training and fine-tuning, inference of such a large model also comes with a heavy computational cost. On IoT (Internet of things) devices and real-world applications, lower computation cost tolerance and restricted computation resource during inference impede these models from deployment.

Recent efforts of efficient inference mainly focus on pruning or compressing the model parameters, *e.g.*, pruning unimportant parts of the neural model weights (Han et al., 2015b; Fan et al., 2019; Gordon et al., 2020), quantizing the number of bits needed (Lin et al., 2016; Shen et al., 2020), distilling from large teacher models to small student models (Hinton et al., 2015; Jiao et al., 2019). These methods produce only one small model with a predetermined target size. Another direction is to switch the model parameters for different data instances, *e.g.*, the mixture of experts (Shazeer et al., 2017), and switch transformer (Fedus et al., 2021). Early exiting, which adaptively produces a series of small models for different data instances, is one of the most common practices. Most previous work makes exit decisions based on either the confidence of output probability distributions or a trained agent. In this work, we propose a carefully designed candidate space for encoder-decoder auto-regressive models and enhance the optimization strategies when training the agent.

In this spirit, we explore the problem of dynamically allocating computation across a generation model. In particular, we consider a standard encoder-decoder transformer auto-regressive generation model. It comprises a stacked structure with multiple layers, each having a multi-head attention layer followed by a feed-forward network (FFN) layer (Zhang et al., 2021b;a; Dai et al., 2022; Tanwisuth et al., 2023). To this end, we introduce a dynamic neural network for the auto-regressive generation models, which includes the attention, feed-forward, and input sequence as the candidate space for switchable decisions. Our method generates an input-dependent inference strategy for each data. For each input sequence, the reinforcement learning agent outputs all the decisions for skipping or keeping each candidate. With the first-layer hidden representations as the input, the policy network is trained to maximize a reward that incentives the use of as few blocks or tokens as possible while preserving the prediction accuracy.

We propose learning optimal switchable strategies that simultaneously preserve prediction accuracy and minimal computation usage based on input-specific decisions. The constrained optimization is utilized as a more principled approach for trading off these two targets (quality *v.s.* efficiency). We target keeping the predicted quality while achieving better efficiency as far as possible. A gradient-based constrained optimization algorithm is implemented

[1]The University of Texas at Austin. Correspondence to: Shujian Zhang <szhang19@utexas.edu>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

under our framework.

We run extensive experiments across summarization, *e.g.*, XSum (Narayan et al., 2018) and CNN/DM (Hermann et al., 2015), question answering, *e.g.*, SQuAD 1.1 (Rajpurkar et al., 2016) and SQuAD 2.0 (Rajpurkar et al., 2018)), and GLUE (Wang et al., 2018a) classification tasks. ❶ Our method not only shows comparable performance across different tasks and datasets but also accelerates model inference by up to 40% with negligible model quality degradation. ❷ Furthermore, we provide extensive ablation studies on different design choices for the proposed method, including the encoder-only or decoder-only switchable schemes. ❸ Our analysis shows the switchable decision contributes the efficiency improvement and accuracy consistency, helping the generation model to choose the inference path and candidates dynamically. ❹ To the best of our knowledge, we present the first switchable decision in the language generation model setting by dynamically making the inference decisions in summarization, question answering, and classification. Our contributions are summarized as follows:

- Present a dynamic network for switchable decisions embracing attention, feed-forward, and input sequence as skipping candidates.
- Propose an efficient and effective way to train the skipping strategies, which can optimize the trade-off between computation and quality.
- Verify the effectiveness and general applicability of the proposed method in various NLP tasks, $e.g.$, summarization, question answering, and classification benchmarks, and provide a rich analysis of our method with various design choices.

2. Related Work and Background

Compact Network Design and Model Compression For model compression, pruning removes unimportant parts of the neural network (Han et al., 2015a; Fan et al., 2019; Gordon et al., 2020), quantization targets the number of bits needed to operate a neural network (Shen et al., 2020), and distillation transfers knowledge from large teacher models to small student models (Chen et al., 2017; Jiao et al., 2019). Efficient network architectures such as MobileBERT (Sun et al., 2020) and ALBERT (Lan et al., 2020) have also been explored for lightweight neural network architectures. Compared to these previous approaches, we focus on dynamic networks with the switchable candidate design to best reduce total computation without degrading prediction accuracy.

Dynamic Networks Dynamic networks enable adaptive computation for various input instances that have been conducted for natural language tasks. Text skimming (Campos et al., 2017; Hansen et al., 2019) learns to skip state updates and shortens the effective size of the computational graph. Dynamic jumping (Yu et al., 2018; Fu & Ma, 2018) strategically skips some tokens without reading them, and directly jumps to an arbitrary location. Early exiting for pretrained models has been explored by previous literature. RTJ (Schwartz et al., 2020), DeeBERT (Xin et al., 2020), and FastBERT (Liu et al., 2020a) make early exiting decisions based on confidence (or its variants) of the predicted probability distribution and are therefore limited to classification tasks. PABEE (Zhou et al., 2020) and BERxiT (Xin et al., 2021) propose patience-based early exiting by exploiting the layer information. Runtime Neural Pruning (Lin et al., 2017), SkipNet (Wang et al., 2018b), and BlockDrop (Wu et al., 2018) use reinforcement learning (RL) to decide whether to execute a network module. Inspired by them, we incorporate lightweight reinforcement learning to make input-dependent decisions and build a diversified switchable candidate space. With the constrained optimization approach, our method saves computational costs without loss of accuracy.

3. Method

Our switchable decision (Figure 1) network focuses on speeding up the inference time for an autoregressive language generation model. Specifically, we suggest a general recipe for the switchable decision: 1) construct the versatile decision space, 2) utilize the input-dependent reinforcement learning agent, and 3) propose the lexicographic (lexico) optimization strategy.

Denote input $\mathbf{o} = (\mathbf{o}_0, \cdots, \mathbf{o}_n)$. With a series of n tokens, a transformer-based language generation model, $\mathcal{M}$ with L layers, first embeds the tokens to form a matrix $O_e \in \mathbf{R}^{n \times e}$, where e is the dimension of the embedding space. These token representations then go through the encoders and decoders of the language model. To speed up the inference time while maintaining similar high quality, we decide whether each input data should skip one layer. This decision problem grows exponentially as we increase the number of layers. Moreover, because of the discrete nature of the decision space, optimization becomes challenging. In this section, we outline our unique design choices to accomplish our goal and overcome optimization challenges.

3.1. Construct Discrete Decision Space

We propose learning the best configurations of *(input, inference paths)* pair for each example using a switchable decision network to speed up inference time. We consider three search space candidates, namely, the attention layer, the feed-forward layer, and query inputs after the first layer. We now explain the details of each search space below.

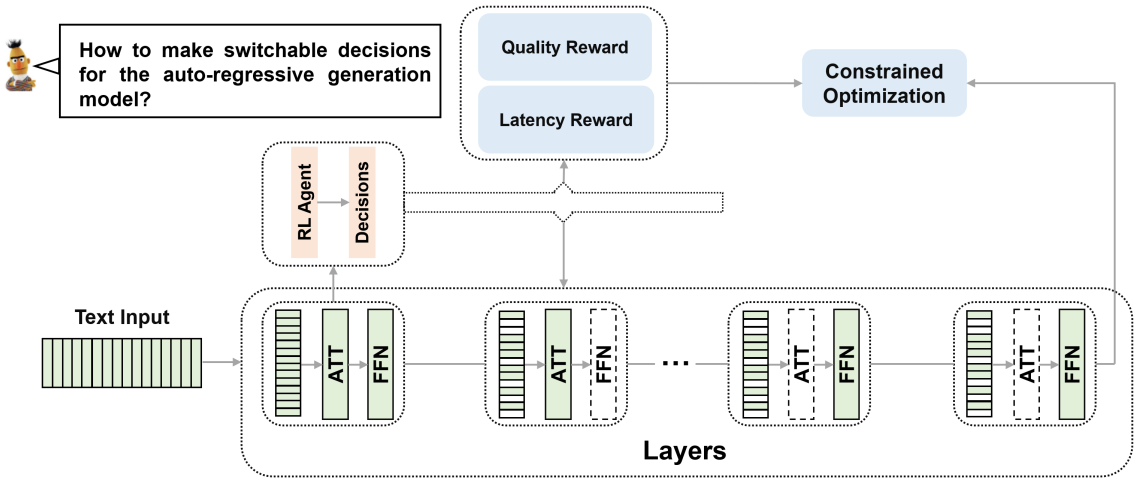

Figure 1. Overview of the dynamic network. Some notations are labeled along with corresponding components. 'Layers' refers to layers within the auto-regressive generation model. 'ATT' refers to the attention candidate, 'FFN' refers to the feed-forward candidate, 'Text Input' refers to the token candidate, and 'Decisions' refers to the skipping decisions from the reinforcement learning agent. The green color represents not skipping. The no-fill in the text input and the dashed line with the no-fill color box represents the skipping.

Attention Candidate. A key component of a transformer-based language model is the attention layer. Zhang et al. (2019) discover that some layers are redundant. To decide whether to skip a certain layer, we model these decisions as a sequence of i.i.d. Bernoulli random variables parameterized by a policy network q. Let $\mathbf{b}_l$ denote the switchable decision of the l^{th} layer, defined as

$$\mathbf{b}_l = \begin{cases} 1 & \text{with probability} \quad g()_l \\ 0 & \text{with probability} \quad 1 - g()_l \end{cases}, \tag{1}$$

where $\in \mathbf{R}^e$ denotes the input of the decision unit, and we apply the first encoder layer output as . The policy network, g, learns instance-specific probabilities of keeping the hidden representations of each layer. To perform skipping, we sample from this distribution and broadcast the indicators, $\mathbf{b}_l^{\text{att}}$, to the input representations of attention layers.

Feed-Forward Candidate. In the same spirit, the feed-forward layers may contain redundant information. Thus, we consider skipping these layers using the same approach as that done in the attention. We decide whether to skip or not based on the indicator $\mathbf{b}_l^{\text{ffn}}$. The design of the policy network is the same as that of the attention layer.

Token Candidate. In addition to skipping the layers, skipping the tokens can also be an alternative way to save computation costs. We create two token skipping strategies: ① skipping the last $p\%$ tokens and ② uniformly skipping $p\%$ tokens. For the former, we set p to 10, 20, and 30. For the latter, p is equal to 25, 33, and 50. To decide which strategy to use, we optimize a categorical random variable parameterized by a function $h(\cdot)$. The input of $h(\cdot)$ is the same as $g(\cdot)$, and the output of $h(\cdot)$ is a distribution over all six candidate decisions.

Encoder and Decoder Structure. Our interested architecture contains encoders and decoders. For the encoders, we apply attention skipping and feed-forward skipping together with token skipping. For the decoders, since every token is meaningful for the final outputs, we only apply attention skipping and feed-forward skipping. When making decisions, we sample from the outputs of our policy network, and broadcast the decisions to the hidden representations of each layer.

3.2. Reinforcement Learning Agent

Policy Network Architecture. Since we aim to speed up the inference process, a simple design for the policy network is adopted. We utilize a one-layer MLP with layer normalization and ReLU activation function. To output a Binomial distribution over decisions, we apply the sigmoid activation to the outputs of the network for attention and feed-forward candidates. We use the softmax function to output the distribution over the choices for token candidates.

Parameterization. During the training process, we sample from the decision distributions, which are parameterized by the policy network. The distribution of the switchable decisions for the layers can be represented as a $2L$-dimensional Bernoulli distribution, which can be written as:

$$\pi(|) = \prod_{l=1}^{2L} g_l()^{s_l} (1 - g_l())^{1-s_l}, \tag{2}$$

where $= \{\mathbf{b}_l^{\text{att}}\}_{l=1}^{L} \bigcup \{\mathbf{b}_l^{\text{ffn}}\}_{l=1}^{L}$. Similarly, the distribution of the token skipping decisions can be represented as a categorical distribution, which can be formalized as:

$$\eta(a\ |) = \prod_{j=1}^{J} h_j()^{\mathbf{1}(a=j)}, \tag{3}$$

where a denotes the choice of the skipping strategy, and J indicates the total number of strategies. We apply seven candidates in practice.

Reward. We define the reward function (Yang et al., 2022b;a; Feng et al., 2023) as a trade-off between quality and computational cost. Given an inference path and a data instance, the reward can be computed from the computation (estimated FLOPs). Intuitively skipping layers will have high reward. We further refer quality as accuracy and loss in the following way:

$$R(, a) = \text{quality} + \lambda \text{computation}, \tag{4}$$

where quality is $-\text{loss}$, computation is the estimated FLOPs (floating point operations), and λ is a coefficient. The overall loss function is defined as the expected value of the reward:

$$J = \mathbf{E}_{\sim\pi,\ \sim\eta}[R(, a)], \tag{5}$$

where π and η are defined in (2) and (3), respectively.

Optimization. To optimize our policy network, we apply policy gradient to compute the gradient of J, and update the parameters of the policy network. We use a self-critical baseline to reduce the variance of the gradients. The constraint-optimization strategy is further applied on the quality and computation. Details are in the next section.

During Inference. Unlike the training process, we do not sample the skipping decisions during inference. Instead, we choose the decisions which maximize the likelihood function.

3.3. Constrained Optimization

Trade-off is a Problem. In the joint training of the main network and the policy network, a trade-off between quality and computation is important. The linear combination of multiple objectives is the most widely used approach. However, the coefficient of the combination requires manual tuning, and it is theoretically unsuitable for non-convex functions. In this work, we consider constrained optimization on trading off two objectives, with a special emphasis on lexicographic (lexico) optimization.

Algorithm 1 Switchable Decision (SD)

1: **Input:** Text o. Auto-regressive generation model $\mathcal{M}$ parameter w with learning rate α_t, policy network parameter θ with learning rate γ_t, number of iterations T.
2: **for** $t = 0$ to T **do**
3: $\quad w \leftarrow w - \alpha_t \nabla(w)$,
4: $\quad \theta$ is updated via Eqn (7),
5: **end for**

Our Equation. To optimize the trade-off between quality and computation in Eqn (4), we propose to use lexicographic optimization, in which the parameters are iteratively updated as

$$\theta_{t+1} \leftarrow \theta_t - \gamma_t e_t, \tag{6}$$

where $\gamma_t \geq 0$ is an adaptive step size and $e_t \in \mathbb{R}^d$ is an update direction to be chosen to balance the minimization of f and constraint satisfaction on q. One of the objectives (say f which is computation in our case) is of secondary importance w.r.t. the other one (say q which is quality). The design criterion for the constrained optimization is when the constraint is not satisfied (i.e., $q(\theta_t) \geq c$), the focus becomes decreasing q to satisfy the constraint as soon as possible; in the meantime, f performs as a secondary objective indicating that f should be minimized to the degree that it does not hurt the descent of q. Therefore, we apply the following update rule to obtain such a goal:

$$\theta_{t+1} \leftarrow \theta_t - \gamma_t (\nabla \text{quality} + \lambda \nabla \text{computation}\,(\theta_t)), \tag{7}$$

where $\nabla\text{computation}$ and $\nabla\text{quality}$ are estimated by score function, and the λ can be computed as $\lambda = \max\left(\frac{\phi(\theta_t) - \nabla\text{quality}(\theta_t)^\top \nabla\text{computation}(\theta_t)}{\|\nabla\text{computation}(\theta_t)\|^2},\ 0\right)$, where $\phi(\theta_t)$ equals to $q(\theta_t) - c$ and the c represents the minimal loss.

The Proposed Algorithm. Our switchable decision (SD) with efficient candidate space and constrained optimization is shown in Algorithm 1. We iteratively update the auto-regressive model and the policy network in a single-loop manner. The policy network parameter θ is updated by Eqn (6) in a direction to balance the optimization of quality and constraint satisfaction on computation.

4. Experimental Settings

Table 1 shows the experimental data configuration.

4.1. Task and Evaluation Metrics

Summarization. We use CNN/DailyMail (Hermann et al., 2015) and XSum (Narayan et al., 2018) to evaluate our

method. CNN/DailyMail consists of 287,226 documents for training, 13,368 documents for validation, and 11,490 documents for testing. XSum has 226,711 news articles accompanied with a one-sentence summary, answering the question "What is this article about?". Following the splits of Narayan et al. (2018), it contains 204,045 train, 11,332 dev, and 11,334 test. Following prior work (Lewis et al., 2019), we use ROUGE (Lin & Hovy, 2003) as our primary metric. We report the unigram ROUGE1 (R-1) and bigram ROUGE-2 (R-2) overlap to assess the informativeness, and the longest common subsequence ROUGE-L (R-L) score to assess the fluency.

Question Answering. The Stanford Question Answering Datasets (SQuAD) v1.1 and v2.0 (Rajpurkar et al., 2016; 2018; Fan et al., 2020) are popular machine reading comprehension benchmarks. For the SQuAD v2.0 dataset, it contains examples where the answer to the question cannot be derived from the provided context. Similar to previous settings (Devlin et al., 2018; Lewis et al., 2019), we use concatenated question and context as input to the encoder of BART, and additionally pass them to the decoder. We report Exact Match (EM) and F1 score for evaluation (Lewis et al., 2019).

Classification. The General Language Understanding Evaluation (GLUE) benchmark is a collection of natural language understanding (NLU) tasks. As shown in Table 1, we include Multi-Genre NLI (MNLI; (Williams et al., 2017b; Zhang et al., 2021d)), Recognizing Textual Entailment (RTE; (Dagan et al., 2005)), and Stanford Sentiment Treebank (SST; (Socher et al., 2013)). The diversity of the tasks makes GLUE very suitable for evaluating the generalization and robustness of our proposed method (Liu et al., 2020b). Accuracy is adopted as our evaluation metric.

Task	Dataset	Train	Val	Test
Summarization	CNN/DailyMail	287.2K	13.4K	11.5k
	XSum	204K	11.3K	11.3K
Question Answering	SQuAD 1.1	87.6K	10.5K	9.5k
	SQuAD 2.0	130.3K	11.9K	8.9K
Classification	RTE	2.5K	276	3k
	MNLI	393K	20K	20K
	SST	67K	872	1.8K

Table 1. Dataset Configuration. The top block is for summarization, the middle block is for question answering, and the bottom block is the classification tasks.

4.2. Implementation Details

Following Lewis et al. (2019), we take the pre-trained BART model as the backbone and utilize the provided checkpoint for finetuning on the downstream datasets. BART is a pre-trained sequence-to-sequence model based on the masked source input and auto-regressive target output, which contains 12 layers of transformer encoder and 12 layers of transformer decoder. Its embedding size is 1,024 and feed-forward size is 4,096. We follow the hyper-parameters used in Lewis et al. (2019). Specifically, in summarization, we set the training steps as 50k and the number of warm-up steps as 500. The max number of tokens and the update frequency are set to be 2,048 and 4, respectively. The learning rate is set to 3×10^{-5}. For the question answering (SQuAD 1.1/2.0). We set the total number of updates and warm-up updates as 5,430 and 326, respectively. The max number of sentences is 3 per device with an update frequency of 2. The learning rate is 1.5×10^{-5}. We refer the readers to Appendix A for classification hyper-parameter configurations, and more details about the settings.

5. Experiments

We evaluate the performance of our switchable dynamic network. In each table, we bold the best result within each column block and the results of our method are obtained with three trials to determine the variance. See Appendix A for full results with error bars.

5.1. Summarization

Table 2 reports our results on two summarization datasets. ① The top block displays the performance of baselines on CNN/DailyMail and XSum datasets, and the bottom block shows the results of incorporating the switchable dynamic networks. We report the results upon the BART large setting in Lewis et al. (2019). ② Summaries in the CNN/DailyMail tend to resemble source sentences and summaries in XSUM are highly abstractive. Baseline models such as BART (Lewis et al., 2019), UniLM (Dong et al., 2019), and BERTSUM (Liu & Lapata, 2019) do well enough, and even the baseline of the first-three source sentences is highly competitive for CNN/DailyMail. Our method can reduce the computation cost while having little or no drop on ROUGE. For example, we even have a 0.2 increase on R1 for CNN/DailyMail and a 0.1 increase on R1 for XSum, while reducing 39% and 18% computation costs, respectively. For the quality of the sentence generations, our method has almost outperformed all the baselines. Especially, for the CNN/DailyMail, we achieve better ROUGE with less than two-thirds FLOPs cost, compared to the original BART-large model (*e.g.*, R1: $44.16 \rightarrow 44.31$, RL: $40.90 \rightarrow 41.01$ on CNN/DailyMail). ③ These results further confirm that SD can work as an effective module to be incorporated into the auto-regressive generation models. SD on improving the inference can also be seen as a complementary module to works focusing on improving pre-training components (Hou et al., 2022; Ge et al., 2022).

Model	CNN/DailyMail				XSum			
	R1 ↑	R2 ↑	RL ↑	FLOPs (%) ↓	R1 ↑	R2 ↑	RL ↑	FLOPs (%) ↓
Lead-3	40.42	17.62	36.67	-	16.30	1.60	11.95	-
UniLM	43.33	20.21	40.51	-	-	-	-	-
BERTSUM	42.13	19.60	39.18	-	38.81	16.50	31.27	-
BART	44.16	**21.28**	40.90	100	45.14	**22.27**	37.25	100
Ours large	**44.31**	21.18	**41.01**	**61.1**	**45.20**	22.16	**37.30**	**81.9**

Table 2. Comparison to models on CNN/DailyMail and XSum. ROUGE are reported for each model. 'BART' represents the BART large model.

Comparison with inference reduction methods. We adopt several methods from the conventional early-exiting method (CALM; (Schuster et al., 2022)), Fast and Robust EarlyExiting (FREE) (Bae et al., 2023), Pegasus (Shleifer & Rush, 2020) (pruning and distillation) and DQ-Bart (Li et al., 2022) (quantization and distillation) and compare them with Ours (SD). ❶ In Shleifer & Rush (2020), it utilizes the shrink and finetune methods: BART-student, Pegasus, and BART on CNN/DailyMail. ❷ In Li et al. (2022), it uses quantization and distillation. It reports the BART (8-8-8 6-1). The number at here represents the number of bits for weights, word embedding, activations, the number of encoder layers, and the number of decoder layers. The results shown in Table 3 demonstrate our switchable decision achieves a good trade-off between quality and computation. These results verify that our method contributes to efficiency and accuracy, helping the generation model to choose the inference path and candidates dynamically. ❸ Further, combining quantization or distillation method, our effective method of improving the language generation model can also be seen as a complementary and plug-in module. We leave this as a future work.

Data	ROUGE-L	FLOPs (%)
BART-student	41.01	93.1
Pegasus	40.34	93.1
DQ-Bart	40.05	18.2
CALM	40.54	80.5
FREE	40.69	76.8
Our Switchable Decision	41.01	61.1

Table 3. Comparison SD with different inference cost reduction methods on CNN/DailyMail.

5.2. Classification

We further show the experimental results on the GLUE in Table 4. The Multi-Genre NLI (MNLI; (Williams et al., 2017b)), Recognizing Textual Entailment (RTE; (Dagan et al., 2005)), and Stanford Sentiment Treebank (SST; (Socher et al., 2013)) are included. We adopt several baselines from the existing literature. ① For BERT, following Devlin et al. (2019), it introduces masked language modeling, which allows pre-training to learn interactions between left and right context words. ② UniLM (Dong et al., 2019), the baseline, fine-tunes BERT with an ensemble of masks, some of which allow only leftward context. ③ RoBERTa, following Liu et al. (2019a), is pretrained with dynamically changing the mask. ④ For BART (Lewis et al., 2019), it is a bi-directional encoder-decoder structure.

Table 4 first displays that SD yields a better trade-off between accuracy and computational efficiency. Ours shows comparable performance over BART and a clear-margin gain over other baselines, while sufficiently lower FLOPs. For example, SD achieves 87.2% accuracy *v.s.* BART's 87.0% accuracy with only 83.6% FLOPs. For the various GLUE benchmarks, our dynamic network demonstrates the strong capability of making skipping decisions for auto-regressive generation models. It further verifies that our method can work for different datasets and can generalize to different input types and fields.

Model	MNLI		RTE		SST	
	m/mm ↑	FLOPs (%) ↓	Acc ↑	FLOPs (%) ↓	Acc ↑	FLOPs (%) ↓
BERT	86.6/-	-	70.4	-	93.2	-
UniLM	87.0/85.9	-	70.9	-	94.5	-
RoBERTa	90.2/90.2	-	86.6	-	96.4	-
BART	**89.9/90.1**	100	87.0	100	**96.6**	100
Ours	89.7/90.0	**82.4**	**87.2**	**83.6**	**96.6**	**80.7**

Table 4. Performance on GLUE. We report the accuracy. All language models here are large size. 'm/mm' and 'Acc' denotes accuracy on matched/mismatched version MNLI and accuracy, respectively.

5.3. Question Answering

For both SQuAD v1.1 and v2.0, following Lewis et al. (2019), we feed the complete documents into the encoder and decoder, and use the top hidden state of the decoder as a representation for each word. This representation is used to classify the token. Table 5 shows our experiment results. The BART large is used as the primary baseline, and the recent baselines (Devlin et al., 2019; Dong et al., 2019; Liu et al., 2019b) are reported. We load the official checkpoint from Fairseq with the official pre-processed SQuAD data. On question answering, by dynamically skipping the candidates from attention layers, feed-forward layers, and input tokens, our model achieves a similar EM and F1 score as BART. Different from the above tasks, here the input is concatenated question and context and additionally passed to the decoder. Although the input is organized in different formats, it is interesting to see the consistent computation cost improvement of our proposed switchable decision in question answering. It further demonstrates that SD can be utilized in general NLP tasks.

6. Analysis

Can we use the proposed dynamic network with the different auto-regressive generation models? As dis-

Model	SQuAD 1.1		SQuAD 2.0	
	EM/F1 ↑	FLOPs (%) ↓	EM/F1 ↑	FLOPs (%) ↓
BERT	84.1/90.9	-	79.0/81.8	-
UniLM	-/-	-	80.5/83.4	-
RoBERTa	**88.9/94.6**	-	**86.5/89.4**	-
BART	88.8/**94.6**	100	86.1/89.2	100
Ours	88.7/94.5	**80.5**	86.0/89.3	**83.3**

Table 5. Results across different strategies on SQuAD v1.1 and v2.0. Answers are text spans extracted from a given document context. [2]

cussed in Section 3, our proposed method targets the auto-regressive generation model. Thus, can our method be adapted to other auto-regressive generation models? We select the GPT-2 (Radford et al., 2019) base and T5 (Raffel et al., 2020) base to study the performance after adapting our proposed switchable decisions. The results are presented in Table 6. It indicates our method is insensitive to different generation models. This confirms our discussion in Section 3 that SD can serve as an efficient alternative dynamic network for versatile generation models. We also analyze the impact of making decisions based on different hidden representations. More details about LLaMA (Touvron et al., 2023) models are included in Appendix A.

Data	ROUGE	FLOPs (%)
BART	44.16/21.28/40.90	100
+ Ours	44.31/21.18/41.01	61.1
GPT-2	37.55/15.53/25.81	100
+ Ours	37.76/15.68/25.93	74.5
T5	42.05/20.34/39.40	100
+ Ours	41.98/20.38/39.61	74.5

Table 6. The proposed method for different generation models on CNN/DailyMail.

What are the differences between encoder-only, decoder-only, and token-only architecture search space? We test if our results are sensitive to the choice of architectures: encoder-only, decoder-only, and encoder-decoder. We create the following scenarios: ① For encoder-only, we incorporate the attention and feed-forward as the skipping candidates. ② For decoder-only, similarly, the attention and feed-forward are included. ③ For token-only, the token candidate is utilized. Then we compare these three designs with SD and BART large to see the impact of incorporating our designed decision space into these different model architectures. As shown in Table 7, we observe distinct FLOPs (reducing 10%) saving by only adding our skipping attention and feed-forward strategies for encoder-only and decoder-only. By only including the token skipping for the encoder-decoder structure, we observe the larger FLOPs (reducing 29%) saving while delivering the comparable ROUGE to BART. We refer the readers to Appendix 6.1 for the detailed skipping percentage of each candidate. These results confirm our analysis and motivation for the switchable decision that using a combination of all these architectural search spaces comes to the best efficiency and accuracy trade-off.

Architecture	ATT	FFN	Token	FLOPs (%)	ROUGE
BART				100	44.16/21.28/40.90
Encoder-Only	✓	✓		91.9	44.21/**21.32**/40.95
Decoder-Only	✓	✓		90.3	44.13/21.08/40.86
Token-Only			✓	71.5	44.09/21.26/40.92
Ours	✓	✓	✓	**61.1**	**44.31**/ 21.18/**41.01**

Table 7. Results of skipping strategies on different architecture spaces for CNN/DailyMail. BART (Izacard & Grave, 2021) large model is presented.

Ablation studies on the components in SD. We conduct the ablation study to examine the role of constrained optimization. For ablation, instead of automatically searching the trade-off between the quality and computation, we manually set the λ in Eqn (7) as $0.2, 0.5, 0.8$. We also include the random selection strategy. The random selection strategy is not learning switchable decisions and would not dynamically assign computation for each data instance. ❶ Table 8 shows that the constrained optimization of our method brings clear benefits. ❷ We find that without CO, ' CO' with different manually tuned λ value shows an unstable trade-off between the ROUGE and FLOPs across all λ values, indicating that manually tuned λ value can not bring both optimized quality and computation together. ❸Empirically, we randomly select a policy from our decision space candidates and use the same other parameters. These result in a degradation in performance and lower FLOPs reduction. It demonstrates the necessity and effectiveness of the constrained optimization for the switchable candidate set in SD structure.

Data	ROUGE	FLOPs (%)
BART	44.16/21.28/40.90	100
Random	41.77/19.02/38.72	75.3
Ours	44.31/21.18/41.01	61.1
- CO, $\lambda = 0.2$	44.12/21.30/40.88	77.8
- CO, $\lambda = 1.0$	42.89/21.02/40.57	68.5
- CO, $\lambda = 1.5$	41.35/19.87/38.39	49.4

Table 8. Comparison of different λ values for the manually tuned trade-off between computation and quality vs. Ours. 'CO' denotes constrained optimization.

Efficiency and time. We provide the parameter sizes, average GPU memory per device, per step training time, and inference time comparisons between the baseline and SD during the finetuning. Experiments in this part are performed on eight Tesla V100 GPUs. ❶ Table 9 shows that SD keeps the parameter size at the same level as the BART large during finetuning. The GPU memory per device and

training time of SD are slightly higher (2.7% for memory and 1.6% for running time) than BART. SD gives the best inference FLOPs, outperforming BART while keeping the comparable ROUGE score and running time. ❷ For the inference time, we evaluate our method and BART large on CNN/DailyMail following the same setting and device with batch size 1. For each iteration, 5.1 seconds (Ours) vs. 10.3 seconds (BART). Our dynamic network demonstrates the strong capability of making skipping decisions. ❸ With the constrained optimization and the reinforcement learning agent, our switchable decision is still computationally productive as the design of our optimization and agent (*e.g.,* applying one-layer MLP for policy network) has almost negligible finetuning computational cost.

Model	ROUGE ↑	Params ↓	GPU memory ↓	s/step ↓	IT ↓
BART	44.16/21.28/40.90	406M	16.8G	1.20	10.3
Ours	44.31/21.18/41.01	423M	17.6G	1.48	5.1

Table 9. Results of parameter size, GPU memory per device, and step time for BART and ours finetuning on CNN/DailyMail. 's/step' represents training step time (second/per step).'IT' represents inference time (second) for each iterations.

6.1. Contributions of Search Space Candidates.

To further identify the contributions of our search space candidates for efficiency improvements and inference acceleration, we present the details skipping percentage of each candidate for CNN/DailyMail, SQuAD 1.1, and SST in Table 10. For CNN/DailyMail, we observe around 8% attention skipping of total attention, 11% feed-forward skipping of total feed-forward, and 29% token skipping of total tokens. The similar skipping percentage holds for question answering. However, we have seen an obvious contrast in the token skipping percentage in classification tasks. The key observation is that the skipping percentages for tokens are high for both CNN/DailyMail and SQuAD 1.1. In addition, our method generally takes around 5K iterations for the reinforcement learning algorithm to converge on CNN/DailyMail. This confirms our conjecture in Section 5.1. For summarization and question answering tasks, the first few parts of inputs are more representative. Thus, it perfectly serves as the candidate for our switchable network to make the skipping decisions.

Dataset	**ATT**	**FFN**	**Token**
CNN/DailyMail	8.50%	11.13%	28.75%
SST	13.30%	13.54%	7.18%
SQuAD 1.1	10.21%	11.93%	9.02%

Table 10. Skipping percentage of each candidate. For example, 8.50% indicates that there are 8.50% of total attention skipped.

The impact of making decisions based on different hidden representations. In Section 3.1, we consider three skipping candidates' hidden representations (attention, feed-forward, and query) after the first layer as the input for our reinforcement learning agent to make switchable decisions. Here, we demonstrate that using hidden representations from different layers comes to the same results, and therefore we pick the easiest one. We set up a baseline here, in which whether to skip the following layer is dependent on the nearby previous layer outputs. We experiment on Ours (based on the output from the first layer) and Ours Layer Wise (layer-wise decisions based on the output from the nearby previous layers). The difference between these two cases is small in Table 11. The layer-wise design requires more computation as it needs to make decisions at each layer. Therefore, it further demonstrates that the design of ours is capable of making skipping decisions and imposing less computational cost.

Data	ROUGE	FLOPs (%)
Ours	44.31/21.18/41.01	61.1
Ours Layer Wise	44.38/21.22/40.97	61.8

Table 11. Comparison of different layer-wise decision of SD on CNN/DailyMail. 'Ours' represents the decision based on the hidden after the first layer. 'Ours Layer Wise' represents the decision based on the hidden representation from the nearby previous layer.

7. Conclusion

Our work demonstrates the benefits of introducing a switchable decision of the dynamic network. The proposed method can dramatically increase the inference efficiency and still enable the model performance. Noticeable FLOPs saving and consistent performance are observed across summarization, question answering, and classification benchmarks. We further conduct a detailed study with the proposed switchable strategy in different settings, *e.g.*, comparing with different architecture search spaces, providing more evidence for making decisions based on hidden representations, and verifying the impact of components. To summarize, the proposed SD is effective and general, with the potential to be incorporated into existing generation models for various NLP tasks.

Impact Statement

The gaps and biases between training and testing data can be significant in real-world settings. Thus, the models may lead to poor performance and unintended consequences on the unseen data. To mitigate these potential issues and reduce the impact of bias in the data, it is essential to utilize the techniques such as data preprocessing, augmentation, and regularization. In addition, the usage of environmental and computational resources should also be considered. This would further lead to the usability and accessibility of the models for different user groups.

References

Bae, S., Ko, J., Song, H., and Yun, S.-Y. Fast and robust early-exiting framework for autoregressive language models with synchronized parallel decoding. *arXiv preprint arXiv:2310.05424*, 2023.

Campos, V., Jou, B., Giró-i Nieto, X., Torres, J., and Chang, S.-F. Skip rnn: Learning to skip state updates in recurrent neural networks. *arXiv preprint arXiv:1708.06834*, 2017.

Chen, G., Choi, W., Yu, X., Han, T., and Chandraker, M. Learning efficient object detection models with knowledge distillation. *Advances in neural information processing systems*, 30, 2017.

Dagan, I., Glickman, O., and Magnini, B. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pp. 177–190. Springer, 2005.

Dai, Y., Tang, D., Liu, L., Tan, M., Zhou, C., Wang, J., Feng, Z., Zhang, F., Hu, X., and Shi, S. One model, multiple modalities: A sparsely activated approach for text, sound, image, video and code. *arXiv preprint arXiv:2205.06126*, 2022.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: pre-training of deep bidirectional transformers for language understanding. arxiv. *arXiv preprint arXiv:1810.04805*, 2018.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

Dong, L., Yang, N., Wang, W., Wei, F., Liu, X., Wang, Y., Gao, J., Zhou, M., and Hon, H.-W. Unified language model pre-training for natural language understanding and generation. *Advances in Neural Information Processing Systems*, 32, 2019.

Fan, A., Grave, E., and Joulin, A. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*, 2019.

Fan, X., Zhang, S., Chen, B., and Zhou, M. Bayesian attention modules. *arXiv preprint arXiv:2010.10604*, 2020.

Fedus, W., Zoph, B., and Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2021.

Feng, Y., Yang, S., Zhang, S., Zhang, J., Xiong, C., Zhou, M., and Wang, H. Fantastic rewards and how to tame them: A case study on reward learning for task-oriented dialogue systems. *arXiv preprint arXiv:2302.10342*, 2023.

Fu, T.-J. and Ma, W.-Y. Speed reading: Learning to read forbackward via shuttle. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4439–4448, 2018.

Ge, T., Xia, H., Sun, X., Chen, S.-Q., and Wei, F. Lossless acceleration for seq2seq generation with aggressive decoding. *arXiv preprint arXiv:2205.10350*, 2022.

Gordon, M. A., Duh, K., and Andrews, N. Compressing bert: Studying the effects of weight pruning on transfer learning. *arXiv preprint arXiv:2002.08307*, 2020.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015a.

Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015b.

Hansen, C., Hansen, C., Alstrup, S., Simonsen, J. G., and Lioma, C. Neural speed reading with structural-jump-lstm. *arXiv preprint arXiv:1904.00761*, 2019.

Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28, 2015.

Hinton, G., Vinyals, O., Dean, J., et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

Hou, L., Pang, R. Y., Zhou, T., Wu, Y., Song, X., Song, X., and Zhou, D. Token dropping for efficient bert pretraining. *arXiv preprint arXiv:2203.13240*, 2022.

Izacard, G. and Grave, E. Distilling knowledge from reader to retriever for question answering. In *ICLR 2021, 9th International Conference on Learning Representations*, 2021.

Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F., and Liu, Q. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. Albert: A lite bert for self-supervised learning of language representations. *ArXiv*, abs/1909.11942, 2020.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

Li, Z., Wang, Z., Tan, M., Nallapati, R., Bhatia, P., Arnold, A., Xiang, B., and Roth, D. Dq-bart: Efficient sequence-to-sequence model via joint distillation and quantization. *arXiv preprint arXiv:2203.11239*, 2022.

Lin, C.-Y. and Hovy, E. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 human language technology conference of the North American chapter of the association for computational linguistics*, pp. 150–157, 2003.

Lin, D., Talathi, S., and Annapureddy, S. Fixed point quantization of deep convolutional networks. In *International conference on machine learning*, pp. 2849–2858. PMLR, 2016.

Lin, J., Rao, Y., Lu, J., and Zhou, J. Runtime neural pruning. *Advances in neural information processing systems*, 30, 2017.

Liu, W., Zhou, P., Zhao, Z., Wang, Z., Deng, H., and Ju, Q. Fastbert: a self-distilling bert with adaptive inference time. *arXiv preprint arXiv:2004.02178*, 2020a.

Liu, X., Wang, Y., Ji, J., Cheng, H., Zhu, X., Awa, E., He, P., Chen, W., Poon, H., Cao, G., et al. The microsoft toolkit of multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:2002.07972*, 2020b.

Liu, X., Gong, C., Wu, L., Zhang, S., Su, H., and Liu, Q. Fusedream: Training-free text-to-image generation with improved clip+ gan space optimization. *arXiv preprint arXiv:2112.01573*, 2021.

Liu, Y. and Lapata, M. Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*, 2019.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019a.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv e-prints*, pp. arXiv–1907, 2019b.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., and McClosky, D. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pp. 55–60, 2014.

Narayan, S., Cohen, S. B., and Lapata, M. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*, 2018.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. Squad: 100,000+ questions for machine comprehension of text. *Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

Rajpurkar, P., Jia, R., and Liang, P. Know what you don't know: Unanswerable questions for squad. *Annual Meetings of the Association for Computational Linguistics (ACL)*, 2018.

Schuster, T., Fisch, A., Gupta, J., Dehghani, M., Bahri, D., Tran, V., Tay, Y., and Metzler, D. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35:17456–17472, 2022.

Schwartz, R., Stanovsky, G., Swayamdipta, S., Dodge, J., and Smith, N. A. The right tool for the job: Matching model and instance complexities. *arXiv preprint arXiv:2004.07453*, 2020.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Shen, S., Dong, Z., Ye, J., Ma, L., Yao, Z., Gholami, A., Mahoney, M. W., and Keutzer, K. Q-bert: Hessian based ultra low precision quantization of bert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8815–8821, 2020.

Shleifer, S. and Rush, A. M. Pre-trained summarization distillation. *arXiv preprint arXiv:2010.13002*, 2020.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. 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, 2013.

Sun, Z., Yu, H., Song, X., Liu, R., Yang, Y., and Zhou, D. Mobilebert: a compact task-agnostic bert for resource-limited devices. *arXiv preprint arXiv:2004.02984*, 2020.

Tanwisuth, K., Zhang, S., Zheng, H., He, P., and Zhou, M. Pouf: Prompt-oriented unsupervised fine-tuning for large pre-trained models. In *International Conference on Machine Learning*, pp. 33816–33832. PMLR, 2023.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018a.

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

Williams, A., Nangia, N., and Bowman, S. R. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*, 2017a.

Williams, A., Nangia, N., and Bowman, S. R. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017b.

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

Xin, J., Tang, R., Lee, J., Yu, Y., and Lin, J. Deebert: Dynamic early exiting for accelerating bert inference. *arXiv preprint arXiv:2004.12993*, 2020.

Xin, J., Tang, R., Yu, Y., and Lin, J. Berxit: Early exiting for bert with better fine-tuning and extension to regression. In *Proceedings of the 16th conference of the European chapter of the association for computational linguistics: Main Volume*, pp. 91–104, 2021.

Yang, S., Feng, Y., Zhang, S., and Zhou, M. Regularizing a model-based policy stationary distribution to stabilize offline reinforcement learning. In *International Conference on Machine Learning*, pp. 24980–25006. PMLR, 2022a.

Yang, S., Zhang, S., Feng, Y., and Zhou, M. A unified framework for alternating offline model training and policy learning. *Advances in Neural Information Processing Systems*, 35:17216–17232, 2022b.

Yang, S., Zhang, S., Xia, C., Feng, Y., Xiong, C., and Zhou, M. Preference-grounded token-level guidance for language model fine-tuning. *Advances in Neural Information Processing Systems*, 36, 2024.

Yu, K., Liu, Y., Schwing, A. G., and Peng, J. Fast and accurate text classification: Skimming, rereading and early stopping. 2018.

Zhang, C., Bengio, S., and Singer, Y. Are all layers created equal? 2019.

Zhang, S., Fan, X., Chen, B., and Zhou, M. Bayesian attention belief networks. In *International Conference on Machine Learning*, pp. 12413–12426. PMLR, 2021a.

Zhang, S., Fan, X., Zheng, H., Tanwisuth, K., and Zhou, M. Alignment attention by matching key and query distributions. *Advances in Neural Information Processing Systems*, 34:13444–13457, 2021b.

Zhang, S., Gong, C., and Choi, E. Knowing more about questions can help: Improving calibration in question answering. *arXiv preprint arXiv:2106.01494*, 2021c.

Zhang, S., Gong, C., and Choi, E. Learning with different amounts of annotation: From zero to many labels. *arXiv preprint arXiv:2109.04408*, 2021d.

Zhang, S., Gong, C., and Liu, X. Passage-mask: A learnable regularization strategy for retriever-reader models. *arXiv preprint arXiv:2211.00915*, 2022a.

Zhang, S., Gong, C., Liu, X., He, P., Chen, W., and Zhou, M. Allsh: Active learning guided by local sensitivity and hardness. *arXiv preprint arXiv:2205.04980*, 2022b.

Zhang, S., Wu, L., Gong, C., and Liu, X. Language rectified flow: Advancing diffusion language generation with probabilistic flows. *arXiv preprint arXiv:2403.16995*, 2024.

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

A. Experimental details

A.1. Full Results With Error Bar

We report the full results of our method with the error bar for summarization and question answering in Table 12 and 14, respectively. The full result of classification is demonstrated in Table 13.

Model	CNN/DailyMail				XSum			
	R1 ↑	R2 ↑	RL ↑	FLOPs (%) ↓	R1 ↑	R2 ↑	RL ↑	FLOPs (%) ↓
Lead-3	40.42	17.62	36.67	-	16.30	1.60	11.95	-
UniLM	43.33	20.21	40.51	-	-	-	-	-
BERTSUM	42.13	19.60	39.18	-	38.81	16.50	31.27	-
BART	44.16	21.28	40.90	100	45.14	22.27	37.25	100
Ours large	44.31±0.1	21.18±0.2	41.01±0.2	61.1	45.20±0.1	22.16±0.2	37.30±0.2	81.9

Table 12. Full results on CNN/DailyMail and XSum. ROUGE is reported for each model. 'BART' represents the BART large model.

Model	MNLI		RTE		SST	
	m/mm ↑	FLOPs (%) ↓	Acc ↑	FLOPs (%) ↓	Acc ↑	FLOPs (%) ↓
BERT	86.6/-	-	70.4	-	93.2	-
UniLM	87.0/85.9	-	70.9	-	94.5	-
RoBERTa	90.2/90.2	-	86.6	-	96.4	-
BART	89.9/90.1	100	87.0	100	96.6	100
Ours	89.7±0.2/90.0±0.3	82.4	87.2±0.1	83.6	96.6±0.2	80.7

Table 13. Full performance on GLUE. We report the accuracy of each dataset. All language models here are large size. 'm/mm' and 'Acc' denotes accuracy on matched/mismatched version MNLI and accuracy, respectively.

Model	SQuAD 1.1		SQuAD 2.0	
	EM/F1 ↑	FLOPs (%) ↓	EM/F1 ↑	FLOPs (%) ↓
BERT	84.1/90.9	-	79.0/81.8	-
UniLM	-/-	-	80.5/83.4	-
RoBERTa	88.9/ 94.6	-	86.5/ 89.4	-
BART	88.8/ 94.6	100	86.1/89.2	100
Ours	88.7±0.3/94.5±0.4	80.5	86.0±0.3/89.3±0.3	83.3

Table 14. Full results across different strategies on SQuAD v1.1 and v2.0. Answers are text spans extracted from a given document context.

A.2. Experimental Datasets

Summarization. CNN/DailyMail contains news articles and associated highlights as summaries. Following the standard splits from Hermann et al. (2015) for training, validation, and testing, we have 90,266/1,220/1,093 CNN documents and 196,961/12,148/10,397 DailyMail documents, respectively. The sentence is split by using the Stanford CoreNLP toolkit (Manning et al., 2014). For XSum (Narayan et al., 2018), summaries are professionally written by the authors of the documents. We also use the preprocessing and data splits from (Narayan et al., 2018; Yang et al., 2024).

Question Answering. Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016; 2018; Zhang et al., 2021c; 2022a) is an extractive question answering task, consisting of questions posed by crowdworkers on a set of Wikipedia articles. The answers, given the questions, are text span from the given reading passage. The SQuAD 1.1 contains around 100,000 question-answer pairs on about 500 articles. The SQuAD v2.0 dataset includes unanswerable questions about the same paragraphs.

Classification. GLUE (Wang et al., 2018a; Zhang et al., 2022b) comprises a collection of text classification tasks meant to test general language understanding abilities. We adopt the three datasets for our experiments: natural language inference (MNLI (Williams et al., 2017a) and RTE (Dagan et al., 2005)) and sentiment analysis (SST-2 (Socher et al., 2013)).

A.3. Experimental Settings

For summarization, we follow the setting in (Lewis et al., 2019) and initialize our models with the pretrained BART large checkpoint. The checkpoint is from the Fairseq library [3]. T5 (Raffel et al., 2020) is also used in Section 6. We adopt the T5 base from the HuggingFace Transformer library[4]. Following Lewis et al. (2019) , the Adam optimizer (Kingma & Ba, 2014; Liu et al., 2021; Zhang et al., 2024) is utilized for optimizing the model parameter with the learning rate 3×10^{-5}. The training step is 50k and the warmup step is 500. Both dropout and attention dropout are set as 0.1. For classification, the detailed training settings are presented in Table 15.

Model	MNLI	RTE	SST-2
NC	3	2	2
LR	5×10^{-6}	1×10^{-5}	5×10^{-6}
BSZ	128	32	128
TS	30,968	1,018	5,233
WS	1,858	61	314

Table 15. Experiment setting for MNLI, RTE, and SST-2 (LR: learning rate, BSZ: batch size, NC: number of classes, TS: total number of training steps, WS: warm-up steps).

Data	ROUGE	FLOPs (%)
BART	44.16/21.28/40.90	100
+ Ours	44.31/21.18/41.01	61.1
GPT-2	37.55/15.53/25.81	100
+ Ours	37.76/15.68/25.93	74.5
T5	42.05/20.34/39.40	100
+ Ours	41.98/20.38/39.61	74.5
LLaMA	-/-/46.68	100
+ Ours	-/-/46.73	77.6

Table 16. The proposed method for different generation models on CNN/DailyMail.

[3] https://github.com/facebookresearch/fairseq/tree/main/examples/bart

[4] https://github.com/huggingface/transformers

A.4. More comparisons

As discussed in Section 6, We select the GPT-2 (Radford et al., 2019) base and T5 (Raffel et al., 2020) to study the performance after adapting our proposed switchable decisions. We also included LLaMA (Touvron et al., 2023) as an additional comparison in Table 16.