DEED: Dynamic Early Exit on Decoder for Accelerating Encoder-Decoder Transformer Models

Peng Tang[†] **Pengkai Zhu**[†] **Tian Li**[‡*] **Srikar Appalaraju**[†]
Vijay Mahadevan[†] **R. Manmatha**[†]
[†]AWS AI Labs [‡]University of California San Diego
{tangpen, zhpengka, srikara, vmahad, manmatha}@amazon.com tianli@ucsd.edu

Abstract

Encoder-decoder transformer models have achieved great success on various vision-language (VL) and language tasks, but they suffer from high inference latency. Typically, the decoder takes up most of the latency because of the auto-regressive decoding. To accelerate the inference, we propose an approach of performing Dynamic Early Exit on Decoder (DEED). We build a multi-exit encoder-decoder transformer model which is trained with deep supervision so that each of its decoder layers is capable of generating plausible predictions. In addition, we leverage simple yet practical techniques, including shared generation head and adaptation modules, to keep accuracy when exiting at shallow decoder layers. Based on the multi-exit model, we perform step-level dynamic early exit during inference, where the model may decide to use fewer decoder layers based on its confidence of the current layer at each individual decoding step. Considering different number of decoder layers may be used at different decoding steps, we compute deeper-layer decoder features of previous decoding steps just-in-time, which ensures the features from different decoding steps are semantically aligned. We evaluate our approach with three state-of-the-art encoder-decoder transformer models on various VL and language tasks. We show our approach can reduce overall inference latency by 20%-74% with comparable or even higher accuracy compared to baselines.

1 Introduction

Transformer models with auto-regressive decoders have shown great success on vision-language (VL) (Wang et al., 2022; Biten et al., 2022; Chen et al., 2022; Appalaraju et al., 2024; Tang et al., 2024) and language tasks (Raffel et al., 2020; Tay et al., 2022; Radford et al., 2019; Vaswani et al., 2017). Among many successful models tackling these tasks, encoder-decoder transformer models (Wang et al., 2022; Biten et al., 2022; Chen et al., 2022; Appalaraju et al., 2021, 2024; Tang et al., 2024; Raffel et al., 2020; Tay et al., 2022) usually show the best accuracy thanks to the strong representation ability of encoder and the strong generative ability of the decoder.

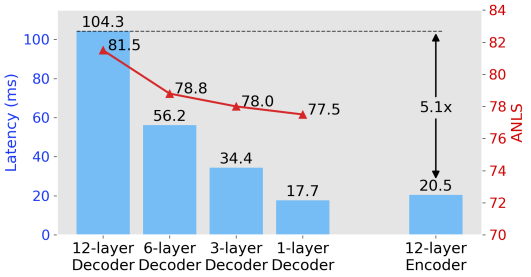


Figure 1: Inference latency and accuracy (ANLS (Mathew et al., 2021)) of LaTr++ (Biten et al., 2022) using different number of decoder layers on the DocVQA validation set. The decoder takes most of the inference time compared to the encoder (104.3 ms *vs*. 20.5 ms). In addition, even using one decoder layer can achieve decent accuracy (77.5% *vs*. 81.5%), implying that most examples do not need all decoder layers during inference.

Nevertheless, encoder-decoder models rely on the auto-regressive decoding to bring its ability into full play at inference. With auto-regressive decoding, each output token is generated conditioned on previous tokens. Therefore, it has to generate tokens one after another, and repeat the feed-forward in each layer as many times. This mechanism leads to high inference latency in the decoder, and makes the decoder take up most of the total inference latency, as shown in Figure 1. Interestingly, even using only one decoder layer, an encoder-decoder model can still get decent prediction accuracy (see Figure 1), which means samples got correct by the one decoder layer do not need the excessive computations in the deeper decoder layers.

[*]Work conducted during an internship at Amazon.

Inspired by these facts, we propose an approach to dynamically allocate adequate amount of computation at a particular decoding step in order to speed-up inference without sacrificing accuracy. Specifically, we build Dynamic Early Exit on Decoder (DEED), a multi-exit model with an early exit strategy to let the model decide whether or not to exit at a specific decoder layer at each decoding step dynamically. Following existing work (Xin et al., 2020; Liu et al., 2021, 2020; Zhang et al., 2022; Geng et al., 2021; Xin et al., 2021; Zhou et al., 2020), we employ confidence-based dynamic early exit where the decoder may decide to exit when it is confident about its prediction. Unlike encoder acceleration, the dynamic early exit for auto-regressive decoder is more challenging. The challenge is two-fold:

- **Multi-exit model**, *i.e.*, a model that can exit / make prediction at each layer. To get accurate predictions out of dynamic early exit, we must build and train a strong multi-exit encoder-decoder model, where each of the decoder layers have strong generative ability.
- **Semantic misalignment at inference**. Tokens can be generated at different decoder layers at different decoding steps. But the auto-regressive decoding requires n-layer features from all the previous steps if the current step is inferring at layer n. They won't be available if the previous decoding steps exit at shallower layers. This semantic misalignment between different layers imposes difficulties when applying naive early exit strategy, leading to degraded accuracy.

Previous approaches address the first challenge by using different prediction heads after each transformer layer (Schwartz et al., 2020; Xin et al., 2020; Geng et al., 2021; Xin et al., 2021). In contrast, we build our multi-exit model by sharing the generation head among different decoder layers and training with deep supervision. The generation head generates the output sequence prediction, *e.g.*, the answer text for visual question answering (Biten et al., 2022; Chen et al., 2022; Alayrac et al., 2022; Lu et al., 2022) or box coordinates for referring expression comprehension (Wang et al., 2022; Lu et al., 2022). In addition, we insert an adaptation module between decoder layers and the generation head. This design helps to strengthen the generative ability of shallow decoder layers by sharing the common generation knowledge among different decoder layers. Moreover, to maintain the generative ability when exiting at the final layer, we proposes a loss function that emphasizes the learning of the final layer. These simple yet effective techniques help to improve the accuracy of shallow decoder layers without sacrificing the accuracy at the final decoder layer.

To address the second challenge, we propose a novel algorithm that dynamically computes required deeper-layer features for previous decoding steps just-in-time. This algorithm effectively resolves the semantic misalignment among different layers at different generation steps. In contrast, the existing work, DAT (Elbayad et al., 2019) and CALM (Schuster et al., 2022), which uses the shallow-layer features as the deeper-layer features directly for later decoding steps, failed to mitigate such semantic misalignment and thus substantially undermine the generative ability of the model.

Our contributions are summarized as follows:

- We propose DEED, a multi-exit model with step-level dynamic early exit on decoder to speed-up inference without sacrificing accuracy for encoder-decoder transformer models.
- We apply our approach to three state-of-the-art encoder-decoder transformer models and evaluate on various VL and language tasks. Our approach is able to reduce 20%-74% overall inference latency with comparable or even higher accuracy compared to baseline models and other dynamic early exit approaches.

2 Related Work

Encoder-Decoder Models for Vision-Language and Language Tasks Transformer models, when first proposed, consist of encoder and decoder and are mainly for language tasks (Vaswani et al., 2017). Following researches improve encoder-decoder transformer models by introducing better unsupervised pre-training strategies and small model architecture changes (Raffel et al., 2020; Tay et al., 2022). Recently, encoder-decoder transformer models have pushed the edge for Vision-Language (VL) tasks (Alayrac et al., 2022; Wang et al., 2022; Biten et al., 2022; Lu et al., 2022; Chen et al., 2022; Appalaraju et al., 2024; Tang et al., 2024) because of strong representation ability of encoder and generative ability of decoder. For example, Flamingo (Alayrac et al., 2022) uses a vision encoder to encode input images and a text decoder to generate text predictions for various VL tasks. LaTr (Biten et al., 2022) utilizes the

sequence generation ability in decoder and layout in multi-modality learning and achieves state-of-the-art accuracy on text-based VQA tasks. OFA (Wang et al., 2022) proposes a unified sequence-to-sequence learning framework to incorporate various VL tasks into the encoder-decoder scheme. Our work focuses on accelerating the decoder inference for this type of encoder-decoder transformer models on VL and language tasks.

Dynamic Early Exit Using Dynamic Early Exit (DEE) is a popular strategy to reduce the inference latency of transformer models (Xin et al., 2020; Liao et al., 2021; Liu et al., 2021, 2020; Zhang et al., 2022; Geng et al., 2021; Xin et al., 2021; Zhou et al., 2020; Li et al., 2021; Hou et al., 2020; Kim and Cho, 2021; Akbari et al., 2022). For example, DeeBERT (Xin et al., 2020) and RomeBERT (Geng et al., 2021) applies DEE to BERT (Kenton and Toutanova, 2019) based on classification confidence scores from different encoder layers. BERxiT (Xin et al., 2021) learns a policy for dynamic early exit. TOKEE (Li et al., 2021) introduces a token-level early exit approach for sequence labelling. However, these encoder-focused approaches cannot be applied to transformer decoders directly, due to the challenges imposed by the auto-regressive mechanism in decoder models.

DAT (Elbayad et al., 2019) is one approach tackling decoder early exit. It introduces a halt-and-copy approach, which halts the computation at a layer if the prediction is confident, and copies the feature from shallow decoder layers to deeper layers in later decoding steps when needed. CALM (Schuster et al., 2022) follows the same halt-and-copy approach for decoder early exit. However, this approach suffers strong semantic misalignment because the semantic information from different decoder layers are not compatible. Thus the later decoding step at deeper layers cannot obtain meaningful features from previous steps, leading to significant accuracy drops. In contrast, our approach dynamically computes the deeper-layer features from earlier steps just-in-time to resolve the semantic misalignment and to achieve high accuracy.

Multi-exit Models The most straightforward way of building multi-exit models is adding deep supervision to each layer (Lee et al., 2015; Teerapittayanon et al., 2016; Schwartz et al., 2020). Nonetheless, it often degrades the accuracy of the final prediction layer. To preserve the final layer accuracy, DeeBERT (Xin et al., 2020) proposes a two-stage training strategy, in which the final prediction layer and the backbone are trained firstly, and other prediction layers are trained secondly with the rest of the parts frozen. However, this two-stage training strategy leads to reduced accuracy of shallow layers. RomeBERT (Geng et al., 2021) designs an approach to increase the accuracy of shallow layers using self-distillation and gradient regularization. BERxiT (Xin et al., 2021) uses an alternating training scheme to improve the accuracy of shallow layers. It alternates between two training objectives: the loss of the final layer only and the loss of all layers. Unlike previous work, we build the multi-exit model by sharing the prediction head among all layers and inserting adaptation modules to align the feature spaces. Our approach shows the best trade-off between final layer accuracy and shallow layer accuracy.

Other Directions for Latency Reduction Apart from dynamic early exit, there are attempts in other directions to reduce latency for transformers. For example, knowledge distillation (Hinton et al., 2015; Jiao et al., 2020; Lin et al., 2022; Sanh et al., 2019) is applied to reduce the model size and latency by distilling information from a large teacher model to a small student model. Model pruning (Gordon et al., 2020; Michel et al., 2019) reduces model size by removing redundant parameters. Non-autoregressive generation (Gu et al., 2018; Qian et al., 2021) avoids the time-consuming step-by-step generation by decoding the predictions in parallel. These directions are orthogonal to dynamic early exit, hence they are not our focus.

3 Approach

We propose DEED, a dynamic early exit on decoder approach to accelerate encoder-decoder transformer models for VL and language tasks. Specifically, we leverage confidence-based step-level dynamic early exit to decide which decoder layer to exit based on how confident we are at each decoding step. At training, we train our multi-exit model with deep supervision (Lee et al., 2015), where the output features of each decoder layer are input to a shared generation head and supervised using the ground truth. At inference, we apply dynamic early exit on the auto-regressive decoder. At each decoding step, the model decides how many decoder layers to use based on its confidence about the output token – hence different number of layers may be used at different decoding steps. In the follow sections, we first introduce the auto-

regressive decoding process and the challenge of semantic misalignment in dynamic early exit on decoder in Section 3.1. Then we describe our multi-exit model architecture and our training strategy in Section 3.2. Finally we show how we resolve the semantic misalignment problem with just-in-time computation of decoder features in Section 3.3.

3.1 Background

Auto-Regressive Decoding At inference, decoder typically generates the prediction in an auto-regressive decoding way, *i.e.*, decoder generates tokens step-by-step and the generated token in each step is conditioned on the previously generated tokens. Theoretically, all the previous tokens are supposed to be input to the decoder to generate the current token, which would cause redundant computation for the previous tokens as their features have been computed at previous decoding steps. In common practice, to reduce redundant computation, the key-value features in the multi-head self-attention layers are all saved and provided for later steps. This practice decreases computation complexity and reduces inference latency, by avoid re-computing key-value features of earlier decoder steps at later steps.

Semantic Misalignment In step-level dynamic early exit, each decoding step can use a different number of decoder layers. As a result, the past key-value features may not always be available for every layer. This misalignment makes it difficult to implement the step-level dynamic early exit, as it cannot retrieve the cached key-value features from previous steps when the current step uses deeper layers. One option is to copy shallower-layer features to deeper-layers (Elbayad et al., 2019; Schuster et al., 2022). However, the deeper-layer features encode higher-level semantics compared to shallower-layer features. A mixture of them across decoding steps will cause semantic misalignment and undermine the generative ability of the model. An easy workaround is to constrain the model to always exit at the same decoder layer, but this would upset our observation that some tokens are harder to generate than others. In experiments, we will show that this constrained approach is not desirable in terms of accuracy and latency. While we have to stick to step-level dynamic early exit and solve semantic misalignment, pre-computing the deeper-layer key-value features is not efficient because we do not know how many layers the following steps will use. To address this issue, we do step-level dynamic early exit with just-in-time computation, see Section 3.3.

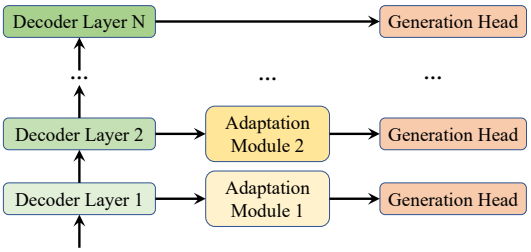


Figure 2: Decoder architecture of our multi-exit model. We share the generation head across different decoder layers and insert adaptation modules between early decoder layers and the generation head.

3.2 Multi-exit Model

To perform step-level dynamic early exit, it is crucial to have a multi-exit model to ensure each decoder layer is capable of generating plausible predictions. So we introduce our multi-exit model here before moving on to how we do step-level dynamic early exit.

Model Architecture In our multi-exit encoder-decoder transformer model, we have a generation head that maps decoder features into tokens. In contrast to existing work (Geng et al., 2021; Xin et al., 2020, 2021), we share the generation head across different decoder layers to share the common generation knowledge among different decoder layers, which strengthens the generative ability of shallow decoder layers. In addition, we insert separate adaptation modules between the shallow decoder layers and the generation head to adapt the features from shallow decoder layers to the semantic space of features from the final decoder layer (see Figure 2). Specifically, the adaptation module is composed of a linear layer followed by layer normalization.

Model Training To train the multi-exit model, the most straightforward way is to add deep supervision (Lee et al., 2015) after outputs of each decoder layer as follows

$$\mathcal{L}_{avg} = \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}_n, \tag{1}$$

where $N, \mathcal{L}_n, \mathcal{L}_{avg}$ correspond to the total number of decoder layers, the loss for the n-th decoder layer, and the average loss across all decoder layers, respectively. However, this approach does not optimize the model for the final decoder layer solely. As a result, the model suffers from degraded accuracy of the final decoder layer, which will cap the

Algorithm 1 Step-level dynamic early exit with just-in-time computation

Require: Current decoding step i, saved past key-value features $\mathcal{P} = \{\mathbf{p}_n^{i'}\}$, saved hidden states $\mathcal{H} = \{\mathbf{h}_n^{i'}\}$, decoder layers $\mathcal{D} = \{D_n\}$, the number of decoder layers N, confidence score threshold τ.

Ensure: Decoded token output t^i.

1: **for** $n = 1$ **to** N **do**
2: Get saved past key-value features $\mathbf{p}_n^{1:j}$ and hidden states $\mathbf{h}_{n-1}^{j+1:i}$.
3: Feed $\mathbf{p}_n^{1:j}$ and $\mathbf{h}_{n-1}^{j+1:i}$ into D_n to compute $\mathbf{p}_n^{j+1:i}$, $\mathbf{h}_n^{j+1:i}$, t_n^i with confidence score c_n^i.
4: Save $\mathbf{p}_n^{j+1:i}$ to $\mathcal{P}$ and $\mathbf{h}_n^{j+1:i}$ to $\mathcal{H}$.
5: **if** $c_n^i > \tau$ **then**
6: Set t^i to t_n^i and terminate the **for** loop.
7: **end if**
8: **end for**

accuracy of our approach. To address this issue, we emphasize the loss of the final layer so as to maintain high accuracy for the final decoder layer. To this end, we add the final decoder layer loss to the training objective as follows

$$\mathcal{L} = \mathcal{L}_{avg} + \mathcal{L}_N. \quad (2)$$

3.3 Step-Level Dynamic Early Exit with Just-in-Time Computation

We perform step-level dynamic early exit at inference on top of the multi-exit model. To avoid semantic misalignment and improve efficiency, we design an algorithm to compute the past key-value features just-in-time. For step i and decoder layer n, we denote the decoder layer n as D_n, the past key-value features as $\mathbf{p}_n^i$, the decoded token output as t_n^i, the corresponding confidence score as c_n^i, and the confidence score threshold as τ. We use colon separated numbers to denote intervals, *e.g.*, $i : j$ denotes the decoding steps from i to j (inclusive). Apart from $\mathbf{p}_n^i$, we also save any output hidden states $\mathbf{h}_n^i$ of D_n at step i if it is computed.

As shown in Algorithm 1, for each decoding step, we go through the decoder one layer per iteration. At decoding step i, first we prepare saved past key-value features $\mathbf{p}_n^{1:j}$ and hidden states $\mathbf{h}_{n-1}^{j+1:i}$ (for the decoding steps where the key-value features are absent), where j ($< i$) corresponds to the sequence length of saved past key-value features for D_n, see line 2 in Algorithm 1. Next we feed

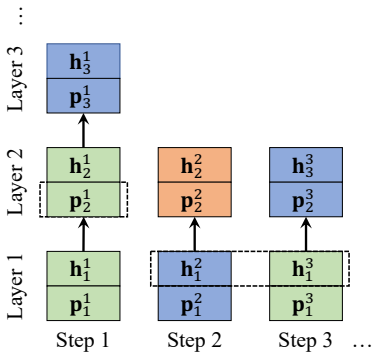


Figure 3: Step-level dynamic early exit with just-in-time computation. Blue boxes: the decoder layers where the model exits. Green boxes: the internal decoder layers. Orange boxes: the layer used for just-in-time computation. Features/hidden states in the dashed boxes are inputs to decoder layer 2 at decoding step 3.

$\mathbf{p}_n^{1:j}$ and $\mathbf{h}_{n-1}^{j+1:i}$ into D_n to compute key-value features $\mathbf{p}_n^{j+1:i}$, hidden states $\mathbf{h}_n^{j+1:i}$, decoded token output t_n^i, and the corresponding confidence score c_n^i, see line 3 in Algorithm 1. We save these newly computed $\mathbf{p}_n^{j+1:i}$ and $\mathbf{h}_n^{j+1:i}$ for future use, see line 4 in Algorithm 1. Taking the decoding process in Figure 3 as an example, at decoding step 3 when the model is about to enter layer 2, the past key-value features $\mathbf{p}_2^1$ are available but p_2^2 are absent, so $\mathbf{p}_2^1$ along with the saved hidden states $\mathbf{h}_1^{2:3}$ will be fed into decoder layer 2. We repeat the same process for every decoder layer until the predicted confidence score c_n^i is larger than a threshold τ, where c_n^i is computed by the classification score after softmax, see line 5-6 in Algorithm 1. Note that although the deeper-layer features are computed for the previous decoding steps, the previous token outputs will not be updated with those features, because each token is supposed to be dependent on the past and any change in the previous tokens will break the dependency.

One may notice that our approach assumes the availability of $\mathbf{h}_{n-1}^{j+1:i}$ at decoding step i. This is assured by our per-layer traversal - the hidden states are always computed and saved at the previous decoder layer.

4 Experiments

We evaluate DEED on LaTr++ (Biten et al., 2022) and OFA (Wang et al., 2022) with various vision-language tasks and on T5 (Raffel et al., 2020) with various language tasks. We do auto-regressive prediction for all the tasks.

	DocVQA			OCR-VQA		
	ANLS↑	Dec. Latency↓	Tot. Latency↓	Accuracy↑	Dec. Latency↓	Tot. Latency↓
Original-b	81.5	104.3	124.6	68.4	109.7	125.6
CALM-b (Schuster et al., 2022)	80.1	73.7	94.1	67.0	73.1	89.0
SLEX-b	81.4	90.1	111.4	**68.3**	109.0	124.7
FTEX-b	81.2	47.1	67.4	67.1	53.8	70.3
DEED-b	**81.9**$_{+0.4}$	**46.1**$_{-55.8\%}$	**66.5**$_{-48.6\%}$	68.1$_{-0.3}$	**52.4**$_{-52.2\%}$	**68.5**$_{-45.5\%}$
Original-L	83.5	181.5	216.3	70.1	202.5	229.6
CALM-L (Schuster et al., 2022)	81.2	81.5	115.6	68.8	93.5	122.5
SLEX-L	83.7	154.3	190.6	69.6	111.5	139.8
FTEX-L	83.1	58.6	91.5	68.6	79.6	108.1
DEED-L	**83.8**$_{+0.3}$	**49.2**$_{-72.9\%}$	**82.8**$_{-61.7\%}$	**69.7**$_{-0.4}$	**79.2**$_{-60.9\%}$	**107.5**$_{-53.2\%}$

Table 1: Accuracy and latency (in ms) on DocVQA and OCR-VQA validation sets. The best results are in **bold face**. The percentage reductions are w.r.t. the original model.

	ST-VQA			Text-VQA		
	ANLS↑	Dec. Latency↓	Tot. Latency↓	Accuracy↑	Dec. Latency↓	Tot. Latency↓
Original-b	69.7	71.9	88.6	61.1	71.7	89.0
CALM-b (Schuster et al., 2022)	69.6	59.1	76.2	57.8	48.3	66.1
SLEX-b	69.8	60.4	77.3	59.6	51.9	68.9
FTEX-b	69.5	41.7	59.0	60.0	45.5	62.4
DEED-b	**69.9**$_{+0.2}$	**33.5**$_{-53.4\%}$	**50.1**$_{-43.5\%}$	**61.0**$_{-0.1}$	**43.5**$_{-39.3\%}$	**61.4**$_{-31.0\%}$
Original-L	70.3	136.5	164.2	63.1	136.5	165.5
CALM-L (Schuster et al., 2022)	70.5	104.7	133.0	59.1	87.1	117.2
SLEX-L	70.2	85.0	114.5	61.3	93.5	122.9
FTEX-L	70.4	65.9	96.4	61.8	79.3	108.9
DEED-L	**71.5**$_{+1.2}$	**50.0**$_{-63.4\%}$	**78.5**$_{-52.2\%}$	**63.6**$_{+0.5}$	**72.1**$_{-47.2\%}$	**102.7**$_{-37.9\%}$

Table 2: Accuracy and latency (in ms) on ST-VQA and Text-VQA validation sets. The best results are in **bold face**. The percentage reductions are w.r.t. the original model.

4.1 DEED on LaTr++

LaTr (Biten et al., 2022) is the state-of-the-art approach for text-based visual question answering (text-VQA). LaTr uses multi-modal encoder-decoder transformer models with OCR text, layout, and visual features as inputs. We improve LaTr by using a better vision backbone and adding better unsupervised pre-training tasks, see Section A.2 for more details. We refer to the improved LaTr as LaTr++ here. Following LaTr, we focus on the text-VQA task.

4.1.1 Settings

We evaluate on four text-VQA datasets: DocVQA (Mathew et al., 2021), OCR-VQA (Mishra et al., 2019), ST-VQA (Biten et al., 2019), and TextVQA (Singh et al., 2019), using accuracy and latency as the metric. For accuracy, we follow the standard protocol to report the metrics on each dataset, *i.e.*, Average Normalized Levenshtein Similarity (ANLS) (Biten et al., 2019; Mathew et al., 2021) for DocVQA and ST-VQA, and accuracy of exact text match between groundtruth and prediction for OCR-VQA and TextVQA. For latency, we report both the total inference latency and the decoder-only latency, as our approach only affects the decoder inference. The latency is measured w.r.t. wall-clock time on the same machine which has 1 Nvidia A100 GPU with 40GB memory. All approaches are implemented in Pytorch (Paszke et al., 2017) with Huggingface (Wolf et al., 2019). To measure the most accurate per sample latency, we use batch size 1 in inference to avoid unnecessary padding. See Section A.3 for more details.

Baselines We compare DEED to the original model, the SOTA approach CALM (Schuster et al., 2022), and two strong baselines SLEX and FTEX we proposed and built:

- **Original**: the vanilla LaTr++ model, on which no early-exit or deep-supervision is applied.
- **CALM** (Schuster et al., 2022): CALM is the state-of-the-art decoder speed-up algorithm. At the step when the model exits at a deeper layer, it simply copies the features of the shallow layer from previous steps to all deeper layers.

	VQA				RefCOCO				
	test-dev↑	test-std↑	Dec. Lat.↓	Tot. Lat.↓	val ↑	testA ↑	testB ↑	Dec. Lat.↓	Tot. Lat.↓
OFA	79.3	79.4	753.5	811.3	90.6	92.5	85.9	132.8	187.1
DEED	79.0	79.1	480.7	538.5	90.2	92.4	85.1	79.5	133.8
	RefCOCO+					RefCOCOg			
	val ↑	testA ↑	testB ↑	Dec. Lat.↓	Tot. Lat.↓	val-u ↑	test-u ↑	Dec. Lat.↓	Tot. Lat.↓
OFA	85.7	89.9	78.6	142.7	197.9	87.2	87.6	132.5	187.6
DEED	85.3	89.6	77.9	61.6	117.9	87.0	87.4	83.0	138.1

Table 3: Accuracy and latency (in ms) of OFA and DEED on various multi-modal tasks using the large size model.

- **Sequence-level early exit (SLEX)**: the decoder always exits at layer m at each decoding step. m is chosen by the accumulated confidence score of the entire sequence. More precisely, for each decoder layer, SLEX needs to infer all decoding steps to get the accumulated confidence score, which makes this baseline unpractical. This baseline is similar to ReasoNet (Shen et al., 2017) for machine comprehension.
- **First-token early exit (FTEX)**: the decoder always exit at layer m at each decoding step. Unlike SLEX, m is chosen based on the confidence score of the first token, which makes FTEX more practical than SLEX because FTEX only needs to infer the first decoding step to make the decision.

Note that in our experiments, SLEX, FTEX, and CALM use the same multi-exit model trained for DEED for fair comparisons, which improves the accuracy of the shallow layers.

Implementation Details We evaluate our approach DEED on both base (-b) and large (-L) variations of LaTr++. The base version has 12 encoder and 12 decoder layers, and the large version has 24 encoder and 24 decoder layers. We follow LaTr (Biten et al., 2022) to do pre-training first and fine-tuning later. We add deep supervision loss in Eq. (2) in both pre-training and fine-tuning. The confidence score threshold τ is selected using cross-validation, specifically, 0.99 on DocVQA and 0.95 for other datasets. See Section A.4 for more details.

4.1.2 Results

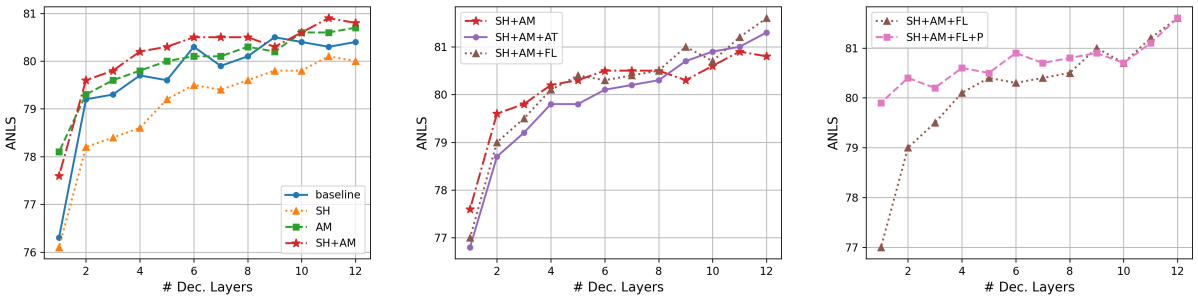
Table 1 and Table 2 show the comparisons of accuracy and latency among DEED and baselines.

Our approach shows excellent performance compared to the original model. It consistently reduces the inference latency for both base and large variations, while maintaining the evaluation accuracy on all benchmark datasets. The latency reduction on decoder is between 40% and 73% across all model and dataset combinations. Specifically, on DocVQA, DEED reduces the decoder latency on the larger variation from 181.5ms to 49.2ms, achieving a large 72.9% reduction, while its ANLS is 0.3 higher than the original LaTr++. DEED also always outperforms other baseline approaches with clear margins. CALM (Schuster et al., 2022) reduces the decoder latency slightly, but it suffers from major accuracy degradation, due to the semantic misalignment introduced by the copy mechanism. SLEX can maintain high accuracy as it makes the decision based on the entire sequence, but its latency improvement is minor compared to DEED . FTEX can reach significant inference acceleration as it can decide to use a shallow layer after the first decoding step. However, it often sacrifices more accuracy because the layer with the maximum first token confidence might not have the best generation for the entire sequence. In contrast, our approach makes exit decisions at each layer and each step, and recomputes the deeper features when necessary, which helps it achieve the best accuracy and latency comparing to all other approaches.

Notice that there is usually more the latency reduction on the large model, because the large model has more decoder layers and early exit still happens at very shallow layers instead of going deeper. In addition, our approach can improve the accuracy of the vanilla LaTr++ in most cases, because our multi-exit model with deep supervision pre-training significantly improves the accuracy for shallow layers (see Section 4.4), and DEED often chooses the layer with the best generative ability based on the confidence scores. In fact, shallow layers can have better predictions than deeper layers on certain examples. If we can choose which layer to make the prediction according to groundtruth, the base model can obtain ANLS 85.0 on DocVQA.

	SQuAD			CNN/DailyMail			SamSum		
	F1↑	Dec. Lat.↓	Tot. Lat.↓	Rouge-L↑	Dec. Lat.↓	Tot. Lat.↓	Rouge-L↑	Dec. Lat.↓	Tot. Lat.↓
T5	92.1	186.8	221.9	41.1	1879.6	1975.7	49.1	694.7	755.6
CALM	90.0	24.4	59.2	20.9	1346.0	1446.1	27.1	556.8	616.8
DEED	**91.6**	**22.5**	**57.6**	**40.7**	**1183.3**	**1283.3**	**47.6**	**544.8**	**605.0**

Table 4: Accuracy and latency (in ms) on language tasks using the large size T5 model.



(a) Ablation on model architecture. (b) Ablation on training objective. (c) Ablation on pre-training.

Figure 4: Ablation studies on the techniques for building and training the multi-exit model. The figure plots the ANLS when exiting at the specific decoder layer. SH: share generation head, AM: adaptation module, AT: alternating training, FL: Additional final layer loss, P: Pre-training with deep supervision, Baseline: the baseline model with unshared generation heads and without the adaptation module.

4.2 DEED on OFA

OFA (Wang et al., 2022) is an encoder-decoder model that unifies multiple modalities and multiple VL tasks with a single paradigm. The encoder and decoder are chosen based on the task.

Experimental Setup Following (Wang et al., 2022), we evaluate DEED with OFA on various multi-modal downstream tasks, specifically, VQAv2 (Goyal et al., 2017) for VQA, and RefCOCO/RefCOCO+/RefCOCOg (Yu et al., 2016; Mao et al., 2016) for referring expression comprehension. We also report the accuracy, the total inference latency, and the decoder-only latency and compare DEED to the baseline model, as described in Section 4.1. For each dataset, the latency is averaged on samples from all splits. Here we only compare to the original OFA model. We use the original pre-trained OFA model and the same fine-tuning procedure as in (Wang et al., 2022) to reproduce the OFA results and train DEED. We do not pre-train the model with deep supervision due to its overwhelming computational costs. We use the large size OFA model and the threshold τ is chosen via cross-validation, *i.e.*, 0.96 for VQA and 0.1 for RefCOCO/RefCOCO+/RefCOCOg.

Results The accuracy and latency of the original OFA and DEED are shown in Table 3. The results of OFA are reproduced using the official code, which are very close to the reported numbers. Again DEED consistently reduces the decoder inference latency with marginal accuracy drops. Specifically, it achieves an average 36.2% and 44% decoder latency reduction on the VQA task and the referring expression comprehension task respectively. In addition, even without deep-supervision pre-training, DEED obtains comparable accuracy compared to the original OFA. The accuracy of DEED should be boosted if we do deep-supervision pre-training for OFA as well. These results demonstrate that our approach can be generalized to different encoder-decoder transformer models and various VL tasks.

4.3 DEED on T5

Experimental Setup Following (Bae et al., 2023), we evaluate DEED with the large size T5 model on various language tasks, specifically, SQuAD (Rajpurkar et al., 2016) for text question answering, CNN/DailyMail (See et al., 2017) and SamSum (Gliwa et al., 2019) for text summarization. We report the accuracy, the total inference latency, and the decoder-only latency and compare DEED to the baseline model and CALM (Schuster et al., 2022). For each dataset, the latency is averaged on samples from all splits. We pre-train T5 with deep supervision on 5M tokens from C4 (Raffel et al., 2020) first and follow the same fine-tuning

procedure as in (Bae et al., 2023) to reproduce the T5 results and train DEED.

Results The accuracy and latency of the original T5, CALM (Schuster et al., 2022), and DEED are shown in Table 4. Similar to vision-language experiments, DEED reduces the decoder inference latency with comparable accuracy compared to the original T5 large model without early exit. Specifically, DEED achieves 74% overall latency reduction on the text question answering dataset SQuAD. These results confirm that our DEED approach can be generalized to other encoder-decoder models and more than vision-language tasks.

4.4 Ablation Study

We study the contribution of each component in DEED. All ablation studies are conducted on DocVQA with LaTr++ base variation. Also see Section B for more ablation studies on the confidence score threshold, the distribution of tokens exiting at each layer, and the number of parameters.

Model Architecture We inspect the effect of shared generation head (SH) and the adaptation module (AM) for multi-exit model. We compare the models of using SH only, using AM only, and using both (SH + AM), to the baseline trained with unshared generation heads without the adaptation module. We use $\mathcal{L}_{avg}$ in Eq. 1 for training and we do not do deep-supervision pre-training. Figure 4a shows results of different approaches. By using both the shared generation head and the adaptation module, the model achieves consistently better accuracy than the baseline, except for the 9-th layer. Notice that the improvement on the first layer is the greatest (>1%), which hugely contributes to the overall latency reduction as more examples can exit at layer 1 without sacrificing the accuracy. However, without the adaptation module, the shared generation head has inferior performance due to the mis-alignment between the generation head and the intermediate features for generation.

Training Objective In Figure 4b, we visualize the ANLS of models trained with the vallina deep supervision $\mathcal{L}_{avg}$ in Eq. 1, alternating training (AT) (Xin et al., 2021), and the additional final layer loss (FL) $\mathcal{L}_N$ in Eq. 2. We can see both AT and FL can improve the accuracy of the deep (> 8) layers, which helps DEED achieve the same or even better accuracy compared to the original model. FL gives better accuracy for most layers than AT, which confirms the effectiveness of our proposed training objective.

Pre-training In our experiments, we found that pre-training the model with deep supervision can significantly improve the accuracy of the shallow layers, as shown in Figure 4c. The magenta curve is the model pre-trained with the deep supervision while the brown curve is the one without. Pre-training with deep supervision increases the accuracy of the first layer by 3%. It also consistently increases the ANLS between layer 2 and layer 8. We argue that deep supervision during the pre-training stage helps the model learn strong generative ability in the shallow layers.

5 Conclusions

We propose DEED, a multi-exit model with step-level dynamic early exit on decoder for encoder-decoder transformer model acceleration. DEED leverages confidence-based step-level dynamic early exit to reduce the computation at each decoding step. To improve the accuracy when exiting at shallow layers, we build a multi-exit model leveraging multiple techniques including deep supervision, shared generation head, adaptation modules, and emphasizing the learning of the final decoder layer. We apply our approach to three state-of-the-art encoder-decoder transformer models. Results on various vision-language and language tasks show that our approach significantly reduces the inference latency with comparable or even higher accuracy compared to baselines. In the future, we will explore DEED for decoder-only models.

References

Mohammad Akbari, Amin Banitalebi-Dehkordi, and Yong Zhang. 2022. E-lang: Energy-based joint inferencing of super and swift language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5229–5244.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, et al. 2022. Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*.

Srikar Appalaraju, Bhavan Jasani, Bhargava Urala Kota, Yusheng Xie, and R Manmatha. 2021. Docformer: End-to-end transformer for document understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 993–1003.

Srikar Appalaraju, Peng Tang, Qi Dong, Nishant Sankaran, Yichu Zhou, and R. Manmatha. 2024.

Docformerv2: Local features for document understanding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(2):709–718.

Sangmin Bae, Jongwoo Ko, Hwanjun Song, and Se-Young Yun. 2023. Fast and robust early-exiting framework for autoregressive language models with synchronized parallel decoding. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5910–5924.

Ali Furkan Biten, Ron Litman, Yusheng Xie, Srikar Appalaraju, and R Manmatha. 2022. Latr: Layout-aware transformer for scene-text vqa. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16548–16558.

Ali Furkan Biten, Ruben Tito, Andres Mafla, Lluis Gomez, Marçal Rusinol, Ernest Valveny, CV Jawahar, and Dimosthenis Karatzas. 2019. Scene text visual question answering. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4291–4301.

Fedor Borisyuk, Albert Gordo, and Viswanath Sivakumar. 2018. Rosetta: Large scale system for text detection and recognition in images. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 71–79.

Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, et al. 2022. Pali: A jointly-scaled multilingual language-image model. *arXiv preprint arXiv:2209.06794*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. 2019. Depth-adaptive transformer. *arXiv preprint arXiv:1910.10073*.

Shijie Geng, Peng Gao, Zuohui Fu, and Yongfeng Zhang. 2021. Romebert: Robust training of multi-exit bert. *arXiv preprint arXiv:2101.09755*.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 70–79.

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

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*.

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

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2020. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33:9782–9793.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling bert for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Gyuwan Kim and Kyunghyun Cho. 2021. Length-adaptive transformer: Train once with length drop, use anytime with search. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6501–6511.

Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. 2015. Deeply-supervised nets. In *Artificial intelligence and statistics*, pages 562–570. PMLR.

Xiaonan Li, Yunfan Shao, Tianxiang Sun, Hang Yan, Xipeng Qiu, and Xuan-Jing Huang. 2021. Accelerating bert inference for sequence labeling via early-exit. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 189–199.

Kaiyuan Liao, Yi Zhang, Xuancheng Ren, Qi Su, Xu Sun, and Bin He. 2021. A global past-future early exit method for accelerating inference of pre-trained language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2013–2023.

Sihao Lin, Hongwei Xie, Bing Wang, Kaicheng Yu, Xiaojun Chang, Xiaodan Liang, and Gang Wang. 2022. Knowledge distillation via the target-aware transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10915–10924.

Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. 2020. Fastbert: a self-distilling bert with adaptive inference time. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6035–6044.

Yijin Liu, Fandong Meng, Jie Zhou, Yufeng Chen, and Jinan Xu. 2021. Faster depth-adaptive transformers. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(15):13424–13432.

Jiasen Lu, Christopher Clark, Rowan Zellers, Roozbeh Mottaghi, and Aniruddha Kembhavi. 2022. Unified-io: A unified model for vision, language, and multi-modal tasks. *arXiv preprint arXiv:2206.08916*.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, and Kevin Murphy. 2016. Generation and comprehension of unambiguous object descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 11–20.

Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. 2021. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2200–2209.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32.

Anand Mishra, Shashank Shekhar, Ajeet Kumar Singh, and Anirban Chakraborty. 2019. Ocr-vqa: Visual question answering by reading text in images. In *2019 international conference on document analysis and recognition (ICDAR)*, pages 947–952. IEEE.

Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. 2019. When does label smoothing help? *Advances in neural information processing systems*, 32.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Rafał Powalski, Łukasz Borchmann, Dawid Jurkiewicz, Tomasz Dwojak, Michał Pietruszka, and Gabriela Pałka. 2021. Going full-tilt boogie on document understanding with text-image-layout transformer. In *International Conference on Document Analysis and Recognition*, pages 732–747.

Lihua Qian, Hao Zhou, Yu Bao, Mingxuan Wang, Lin Qiu, Weinan Zhang, Yong Yu, and Lei Li. 2021. Glancing transformer for non-autoregressive neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1993–2003.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. 2022. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35:17456–17472.

Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A Smith. 2020. The right tool for the job: Matching model and instance complexities. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6640–6651.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1047–1055.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. 2019. Towards vqa models that can read. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8317–8326.

Peng Tang, Srikar Appalaraju, R Manmatha, Yusheng Xie, and Vijay Mahadevan. 2024. Multiple-question multiple-answer text-vqa. *NAACL*.

Yi Tay, Mostafa Dehghani, Vinh Q Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Siamak Shakeri, Dara Bahri, Tal Schuster, et al. 2022. Ul2: Unifying language learning paradigms. *arXiv preprint arXiv:2205.05131*.

Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd international conference on pattern recognition (ICPR)*, pages 2464–2469.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Peng Wang, An Yang, Rui Men, Junyang Lin, Shuai Bai, Zhikang Li, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. 2022. Ofa: Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. In *International Conference on Machine Learning*, pages 23318–23340. PMLR.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. Deebert: Dynamic early exiting for accelerating bert inference. *arXiv preprint arXiv:2004.12993*.

Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. 2021. 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*, pages 91–104.

Licheng Yu, Patrick Poirson, Shan Yang, Alexander C Berg, and Tamara L Berg. 2016. Modeling context in referring expressions. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 69–85. Springer.

Zhen Zhang, Wei Zhu, Jinfan Zhang, Peng Wang, Rize Jin, and Tae-Sun Chung. 2022. PCEE-BERT: Accelerating BERT inference via patient and confident early exiting. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 327–338, Seattle, United States. Association for Computational Linguistics.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. Bert loses patience: Fast and robust inference with early exit. *Advances in Neural Information Processing Systems*, 33:18330–18341.

This is the appendix for the main DEED paper. Here we discuss the architecture and results of LaTr++, and the results of our reproduced OFA *vs*. the original OFA results.

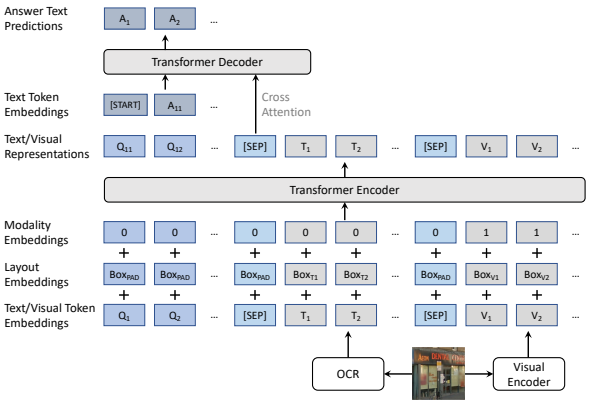


Figure 5: The architecture of LaTr++ for text-VQA.

A LaTr++

LaTr (Biten et al., 2022) obtains the state-of-the-art results on the text-based visual question answering (text-VQA) task. LaTr uses multi-modal encoder-decoder transformer models which takes OCR text, layout, and visual features as inputs. We improve LaTr by replacing the ViT-based vision backbone (Dosovitskiy et al., 2020) with simple multi-layer perceptrons and adding more unsupervised pre-training tasks, following DocFormerv2 (Appalaraju et al., 2024). We refer to the improved LaTr as LaTr++. See Figure 5 for the architecture of LaTr++ and more details below.

A.1 Architecture

In LaTr++, given an input image, we resize the image to size 500x384 and split the image into 196 patches with patch size 32x32. Instead of using ViT (Dosovitskiy et al., 2020), we simply use a linear projection layer to generate 196 visual token embeddings for each patch. We further use one more linear layer with the intention of compressing the extracted 196 visual tokens to only 128 visual tokens. These visual tokens are then concatenated with word embeddings, from here the architecture is identical to LaTr (Biten et al., 2022) and T5 (Raffel et al., 2020). Arguably our LaTr++ architecture is much more simpler than LaTr (Biten et al., 2022) as we do not have a pre-trained ViT as a dependency, hence our model has less number of parameters for equal model size compared to LaTr (Biten et al., 2022).

A.2 Pre-training

We use the IDL dataset[1] described in the main paper to pre-train the LaTr++ models. We use

[1] https://www.industrydocuments.ucsf.edu/

	ST-VQA (ANLS) ↑	TextVQA (Accuracy) ↑	OCR-VQA (Accuracy) ↑
$\text{LaTr}_{\text{base}}$	68.3	59.5	67.5
$\text{LaTr++}_{\text{base}}$	**69.7**	**61.1**	**68.4**
$\text{LaTr}_{\text{large}}$	70.2	61.1	-
$\text{LaTr++}_{\text{large}}$	**70.3**	**63.1**	**70.1**

Table 5: Accuracy comparison between LaTr++ and LaTr on ST-VQA, TextVQA, and OCR-VQA validation sets. The best results are in **bold face**.

	VQA		RefCOCO			RefCOCO+			RefCOCOg	
	test-dev↑	test-std↑	val ↑	testA ↑	testB ↑	val ↑	testA ↑	testB ↑	val-u ↑	test-u ↑
OFA (original)	79.4	79.5	90.1	92.9	85.3	85.8	89.9	79.2	85.9	86.6
OFA (reproduced)	79.3	79.4	90.6	92.5	85.9	85.7	89.9	78.6	87.2	87.6

Table 6: Accuracy comparisons between the original OFA and our reproduced OFA.

the standard T5 denoising pre-training task (Raffel et al., 2020) as in the original LaTr paper (Biten et al., 2022). In addition, to make the LaTr++ a more competitive baseline we add two more unsupervised pre-training tasks at the encoder: a) Line prediction task - in order to teach the model the relative position semantic information between text tokens, we randomly pick two text tokens and ask the model to predict how many lines are between them. There are only three labels: 0, 1 and 2. Any text token pairs that have more than 2 lines between them are assigned to 2 because distant text tokens are not related and the model does not need the precise number of lines between them. b) Token-to-grid task - To utilize global information the task involves creating a virtual 3x3 grid and asking the network to predict which grid each text token falls in. Losses of all three tasks, *i.e.*, standard denoising, line prediction, and token-to-grid, are added to form the final pre-training loss for LaTr++.

A.3 Settings

We evaluate on four text-VQA datasets: DocVQA (Mathew et al., 2021), OCR-VQA (Mishra et al., 2019), ST-VQA (Biten et al., 2019), and TextVQA (Singh et al., 2019). DocVQA is a VQA dataset dedicated to document text understanding, and OCR-VQA focuses on question-answering on book covers. ST-VQA and TextVQA contain natural images of everyday scenes with textual information and require the understanding of the text in the image to answer the question. Following (Biten et al., 2022), we use Amazon Textract[2] for DocVQA, Amazon Text-in-Image[3] for ST-VQA and TextVQA, and Rosetta (Borisyuk et al., 2018) for OCR-VQA, to extract text information from images.

A.4 Implementation Details

We pre-train our models on the Industrial Document Library (IDL) dataset[4], using the tasks described in Section A.2. We add deep-supervision loss of the T5 denoising task on all decoder layers for DEED, because we found it considerably improves the generative ability of shallow layers, as discussed in our main paper. We pre-train the base version with deep supervision on 5M IDL data for 30 epochs. For the large variation, to reduce the computational costs while achieving competitive performance, we firstly pre-train the model on 64M IDL data for 1.5 epochs without deep supervision, and then pre-train it on 64M IDL data with deep supervision using batch size 18 for 60k steps. The models are then fine-tuned on each dataset following the same settings as in (Powalski et al., 2021; Biten et al., 2022). We follow the convention of fine-tuning on the combination of ST-VQA and TextVQA training sets when evaluating on these two datasets (Biten et al., 2022). Label smoothing is used to calibrate the confidence scores (Müller et al., 2019). The confidence score threshold τ is selected using cross-validation, specifically, 0.99 on DocVQA and 0.95 for other three datasets.

[2] https://aws.amazon.com/textract/

[3] https://docs.aws.amazon.com/rekognition/latest/dg/text-detecting-text-procedure.html

[4] https://www.industrydocuments.ucsf.edu/

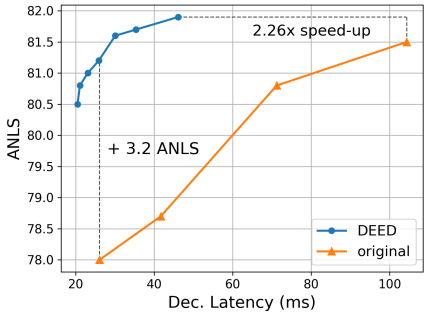


Figure 6: ANLS against decoder latency for DEED and the original model on the DocVQA validation set. DEED are obtained by tuning the confidence threshold τ . The original model is trained with different numbers of decoder layers.

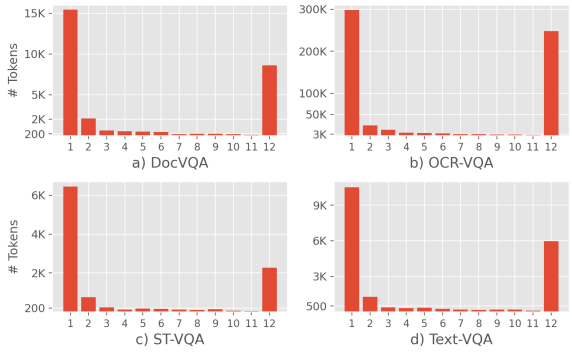


Figure 7: Histogram of layer at which our model exits when evaluated on the validation set of different VQA datasets.

	# Parameters		
	Baseline	Ours	Previous
LaTr++ (-b)	232M	239M	503M
LaTr++ (-L)	750M	774M	1507M

Table 7: # parameters of the baseline model (Baseline), our approach (Ours), and previous approaches (Previous).

A.5 Results

Here we compare LaTr++ to LaTr on three text-VQA datasets: ST-VQA (Biten et al., 2019), TextVQA (Singh et al., 2019), and OCR-VQA (Mishra et al., 2019). For ST-VQA and TextVQA, we train LaTr++ on the combination of ST-VQA and TextVQA training sets, following LaTr (Biten et al., 2022). For OCR-VQA, we train LaTr++ on the OCR-VQA training set only. All results are reported on the validation sets of these three datasets. As we can see in Table 5, LaTr++ obtains better results than state-of-the-art approach LaTr on the text-VQA task.

B More Ablation Studies

B.1 The Influence of the Confidence Score Threshold

DEED can realize different trade-offs between the accuracy and latency by tuning the confidence score threshold τ, to fit in different use cases without retraining the model. In Figure 6, we visualize the decoder latency and ANLS of DEED w.r.t. different thresholds ([0.5, 0.99]). We compare DEED to the original LaTr++ trained with 2, 4, 8, and 12 decoder layers. When using the threshold 0.99, our approach reaches the highest ANLS score of 81.9, which exceeds the vanilla 12-layer LaTr++, while achieving 2.26X decoder speed-up. At the other end of the spectrum, DEED can reduce the decoder latency to 25.9ms (4.03X speed-up *vs.* 12-layer LaTr++) with ANLS score of 81.2. In contrast, to reduce the latency to 26ms, the original model can only use 2 decoder layers, resulting in a significant 3.2 ANLS drop compared to DEED.

B.2 The Distribution of Tokens Exiting at Each Layer

We visualize the distribution of tokens exit at each layer on four text-VQA datasets in Figure 7. The majority of the tokens exit at the first layer, then the last layer. Only a small amount of tokens exit at middle layers. This shows that the model exits at shallow layers for easy predictions, which aligns with our observation that most samples do not need all decoder layers during inference. In addition, from Figure 4 in the main paper, the second most samples are hard samples that can be correctly predicted by the final decoder layer or even the final decoder layer fails, so the second peak of the histogram appears at the final decoder layer (*i.e.*, decoder layer 12).

B.3 Analyses on the Number of Parameters

Compared to the previous multi-exit models, one advantage of our multi-exit model is fewer number of parameters. For LaTr++, the base version (-b) has 232M parameters with hidden size (d_{model}) 768, 12 encoder layers, and 12 decoder layers. The large version (-L) has 750M parameters with d_{model} 1024, 24 encoder layers, and 24 decoder layers. Each adaptation module consists of a linear layer ($d_{model} \times d_{model}$ parameters) followed by layer normalization (d_{model} parameters). There-

questionID	documentID	Question	Baseline	DEED
2193	gsxk0226_3	what is the name of the firm mentioned at the top in bold letters?	merrill lynch pierce fenner & smith inc.	merrill lyner fenner fenner & smith
6010	fllg0224_1	name the materials for which this procedure note is given	g14-9a and g19-33a	g14tho
7590	flxn0020_1	what is the "street adress(no po box)" ?	409 n. main st.	409 lake plaza dr. main st.
22076	fnnp0227_6	who is the applicant?	gerard jean dubois	gerard jean jean jean jean jean - b
42105	fkxn0226_9	what is the name of the consulting agency given below the logo?	"accumyn"	"accaccyn"
45675	glxn0226_4	which mechanism is used to lift the insert out of the shell?	suction cup mechanism	suction cup
50469	fgfl0228_4	what type of plant is scgp 1?	demonstration plant	demonstration
51502	gjhp0000_1	what is the objective given in the document?	announce and explain forsyth's labor day 1998 wholesale promotion.	announce and explain fors labor day extension.
52622	fglc0003_1	what is the full form for ncciu?	north carolina center for international understanding	north carolina center for international
55281	fsgj0223_63	what is written on the top left corner of the page?	"gtc"	"gtl"
55304	glvj0223_10	what is subheading a?	related party with whom transaction have taken place during the year	related party of persons persons persons persons persons during the year
58295	fyvw0217_3	what is the text on the top right corner of the page?	achieving clarity, renewing confidence	achieving clarity, renewal of confidence
58312	fqvw0217_39	what is the adverse effects of using megestrol acetate?	menstrual bleeding in women after discontinuation	menstrual bleeding in women after completion
59990	ggbm0227_2	what is the short form for esquire?	"esq."	"esquin"
63045	fhjc0228_2	what is the name of the bank advertised?	first american national bank	first american

Table 8: Error cases of DEED vs. baseline (LaTr++) with base model size on DocVQA. The baseline outputs are the identical to the ground truth for these examples.

fore, each adaption module only has 0.6M parameters (-b) and 1.05M parameters (-L). The adaption modules from all layers only increase ∼3% of the full model parameters. In contrast, a generation head has 24.7M parameters (-b) and 32.9M parameters (-L) due to the output dimension (32128). Using unshared generation heads increases the total number of parameters by >100%. So our adaptation module has much fewer parameters than unshared generation heads in previous approaches. See Table 7 for more details.

C OFA Results

We use the original pre-trained OFA model and the same fine-tuning procedure as in (Wang et al., 2022) to reproduce the OFA results and train DEED, using the official code[5]. The only exception is RefCOCOg - we fine-tune our model on top of the RefCOCO fine-tuned model, because RefCOCOg has fewer training samples than RefCOCO and RefCOCO+, which makes the accuracy on RefCOCOg inferior if we fine-tune our model on RefCOCOg directly. Here we compare our reproduced OFA results and the original OFA results on VQAv2 (Goyal et al., 2017) for VQA and RefCOCO/RefCOCO+/RefCOCOg (Yu et al., 2016; Mao et al., 2016) for referring expression comprehension. As we can see, our reproduced results are close to the reported numbers.

D Qualitative Results

We provide some qualitative results of DEED on DocVQA to help understand the behavior of our method and its impact on the accuracy. Specifically, we list the error cases of DEED where the predictions of the baseline model are correct. The outputs from DEED and the baseline model are tabulated in Tab. 8. From the error cases, we observed a few

[5] https://github.com/OFA-Sys/OFA

patterns of the errors from DEED :

1. Missing words at the end of the prediction (e.g., `2193`, `45675`, `63045`, etc.).
2. Making errors on rare words like names (e.g., `51502`: `"forsyth"` → `"fors"`, `42105`: `"accumyn"` → `"accaccyn"`, etc.).
3. Repeating the same word (e.g., `22076`: `gerard jean` → `gerard jean jean ...`, etc.).