FREE: Fast and Robust Vision Language Models with Early Exits

Divya Jyoti Bajpai and Manjesh Kumar Hanawal
Department of IEOR, IIT Bombay
{divyajyoti.bajpai, mhanawal}@iitb.ac.in

Abstract

In recent years, Vision-Language Models (VLMs) have shown remarkable performance improvements in Vision-Language tasks. However, their large size poses challenges for real-world applications where inference latency is a concern. To tackle this issue, we propose employing Early Exit (EE) strategies in VLMs. However, training exit classifiers in VLMs is challenging, particularly with limited labeled training data. To address this, we introduce FREE, an adversarial training approach within a GAN-based framework. Here, each exit consists of a transformer layer and a classifier. The transformer layer is adversarially trained to produce feature representations similar to the final layer, while a feature classifier serves as the discriminator. Our method focuses on performing input-adaptive inference that increases inference speed with minimal drop in performance. Experimental results demonstrate the effectiveness of our approach in enhancing accuracy and model robustness by mitigating overthinking and the phenomenon of *mid-crisis* that we highlight. We experimentally validate that our method speeds up the inference process by more than $1.51\times$ while retaining comparable performance. The source code is available at https://github.com/Div290/BLIPEE.

1 Introduction

Vision-language pre-training (VLP) has evolved significantly with the emergence of sophisticated pre-trained Vision Language Models (VLMs). These models have consistently pushed the performance boundaries across various vision-language tasks. However, their demanding computational requirements and inference latency pose challenges for real-time applications. Several models, such as BLIP-2 (Li et al., 2023), MiniGPT (Zhu et al., 2023) etc., leverage off-the-shelf large-scale pre-trained models as building components with their parameters frozen. This reduces VLMs training parameters but makes them extremely slow during inference, as highlighted in (Bajpai and Hanawal, 2024a) leading to higher inference time.

The use of the Language Model (LM) component with frozen parameters not only makes VLM susceptible to overthinking but also to another phenomenon that we term *mid-crisis* (Elhoushi et al., 2024). This phenomenon occurs when intermediate layers suffer performance drops due to the search for irrelevant features. While initial layers capture shallow representations and syntactic information, and deep layers learn semantic-fusion relations (Fei et al., 2022), intermediary layers tend to capture dataset patterns that degrade their performance, even losing the information learned by initial layers, and the model has to regain the lost information again in deeper layers.

We illustrate this phenomenon in the left figure of Fig. 3 and Sec. 3.1, showcasing accuracy on the VQAv2 (visual question answering) (Das et al., 2017) dataset across different exits when trained with dedicated classifiers as in vanilla early exit setup for the intermediate layers. This raises the question: how can we mitigate mid-crisis and overthinking to enhance the accuracy and efficiency of VLMs?

We address this using Early Exit (EE) techniques (Xin et al., 2020; Zhou et al., 2020; Zhu, 2021), an input-adaptive method that reduces computational costs by bypassing certain layers while preserving the knowledge encoded in large-scale models. Since real-world datasets contain both "easy" and "hard" samples, EE ensures adaptive computation per sample.

However, applying EE to VLMs presents challenges: (1) Exit classifiers introduce significant overhead—e.g., a single exit for $\text{OPT}_{2.7\text{B}}$ (Zhang et al., 2022a) decoder adds approximately 130M parameters; (2) Training these exits requires sub-

Findings of the Association for Computational Linguistics: ACL 2025, pages 23599–23615

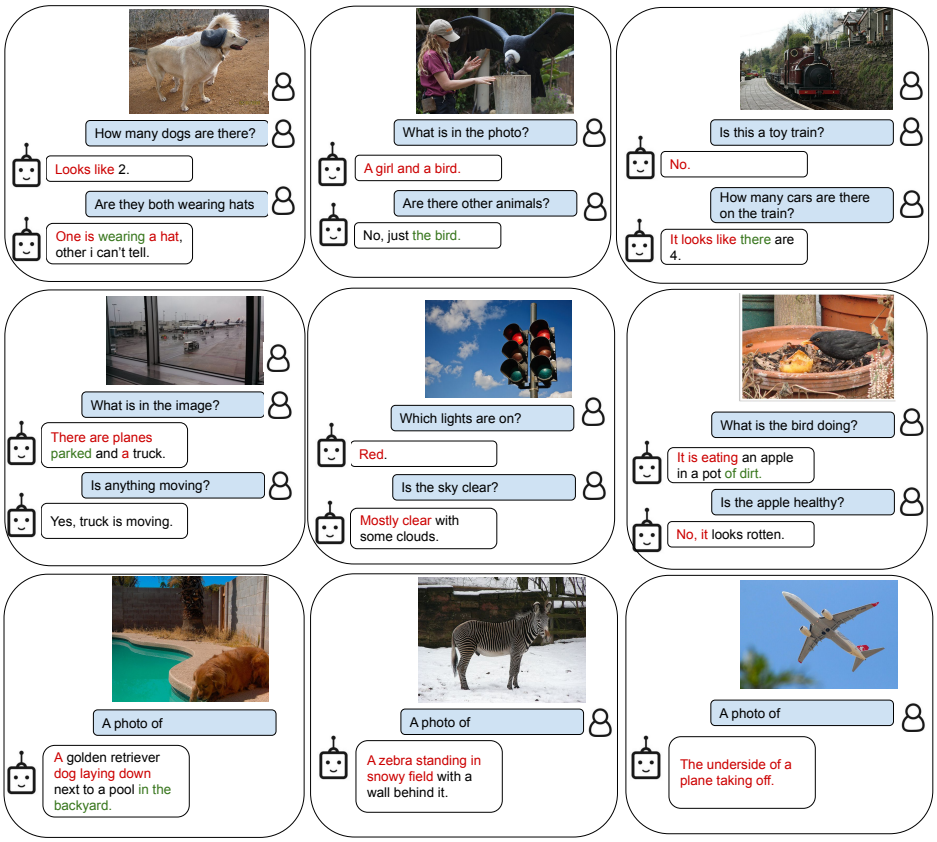


Figure 1: This figure provides some example outputs of FREE ViT-g $OPT_{2.7B}$. The different colours show the difficulty levels of tokens in captions. Red: Easy to predict and predicted at initial (1-12) layers. Green: Mediocre hard, predicted at intermediary (13-24) layers. Black: Hard to predict, predicted at deeper (25-32) layers.

stantial labeled data, limiting EE adoption in zero-shot VLMs that require minimal fine-tuning.

We propose FREE Fast and Robust Vision-Language Models with Early Exits. This efficient EE training framework minimizes training costs while maintaining accuracy. Our method employs a Generative Adversarial Network (GAN)-based (Creswell et al., 2018) framework, leveraging the pre-trained VLM outputs to align feature representations between intermediate exits and the final layer. Unlike cosine similarity, which hampers generalization, the adversarial setup improves feature consistency, ensuring robust exit predictions.

Our method attaches exits and *Feature Classifiers* (FCs) to intermediary layers of the decoder of the VLM. Each exit consists of a single *exit transformer* (ET) and an *exit classifier* (EC). The exit transformer is a replica of the layers in the decoder LM of the VLM. They are used as generators and feature classifiers as discriminators in a GAN-based setup, as shown in Fig. 2. The task of the feature classifier is to correctly classify if the input is from the exit or final layer, and the task of the exit transformer is to generate representations similar to the final layer to fool the feature classifier of that exit.

As the exit transformer is trained to generate representations similar to that of the final layer, we can use the final layer classifier at all exits as an EC with frozen parameters, which are used to map the outputs of the exit transformer to class probabilities. As the size of ETs is significantly smaller than that of ECs, it substantially reduces the number of parameters. In this way, a single LM layer attached to the exits helps produce similar feature representations and reduces the training parameters by utilizing a frozen final layer classifier. By attaching EEs, our method reduces the chances of mid-crisis or overthinking and makes the inference process faster. Fig. 1 shows how our method can speed up inference while maintaining comparable performance by using the EE methods.

Adversarial training methods are prone to *catastrophic forgetting* (Kirkpatrick et al., 2017; Ryu et al., 2022) and *mode collapse*, which can hinder exit training. To address these challenges, we propose both supervised and unsupervised strategies. In the supervised setting, when a small

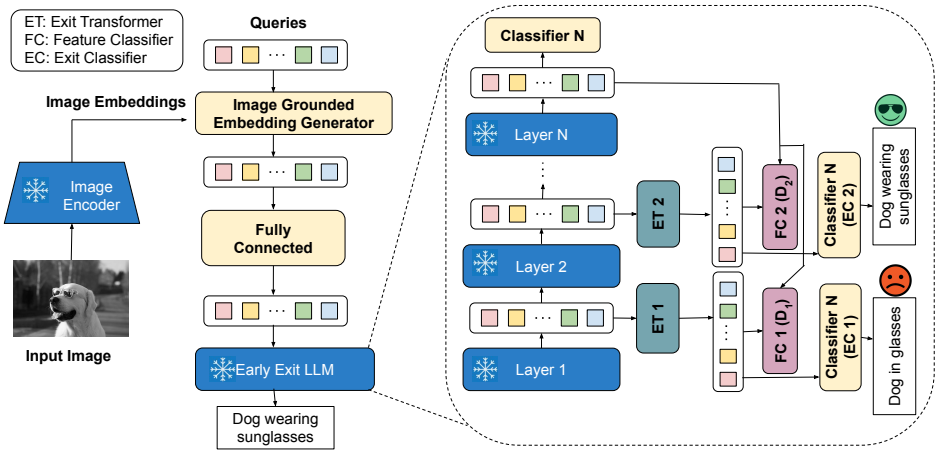


Figure 2: This figure illustrates our model's architecture. The Q-Former output and previous tokens pass through the backbone. The final layer and ET outputs go to feature classifiers, refining ET to match final representations. During inference, the final classifier is used, and confident exits generate the caption.

labeled dataset is available, we use *hard labels* to stabilize training, preventing catastrophic forgetting and avoiding local optima. In the unsupervised case, when labeled data is unavailable, we utilize *soft labels* from vanilla VLM or generate high-quality synthetic labels using *CapFilt* (Li et al., 2022b), as employed in BLIP and BLIP-2.

Our method remains effective across different scenarios. When a labeled dataset is available, it is used to mitigate catastrophic forgetting and mode collapse. If no labeled data is present, we employ knowledge distillation to solve this issue. When both labeled data and annotations are unavailable but computational resources are accessible, we generate synthetic labels using CapFilt. By providing solutions for all data constraints, our approach ensures stable and effective exit training while enhancing efficiency and generalization. In summary, our contributions are as follows:

- We introduce an EE strategy named FREE for VLMs to effectively mitigate inference latency by reducing unnecessary computations inherent in their large-scale architecture.
- We propose an efficient training strategy to improve performance at EE classifiers. FREE emulates the behaviour of the final layer at the exits through adversarial learning. This reduces the need for labeled training datasets.
- Our model reduces the number of trainable parameters of the exits by reutilizing the frozen final layer classifier at the exits.
- We experiment both qualitatively (see Fig. 1 (**recommended**)) and quantitatively on various tasks such as image captioning, visual question-answering and visual dialogue dataset. Our method provides inference speed $> 1.51\times$ with comparable accuracy than vanilla VLM inference. We show the robustness of our method in Appendix A.4.

2 Related works

We discuss the VLPs with LM components and EE strategies related to our work below.

Vision-language Pre-training: Different model architectures have been proposed for specific downstream tasks in VLPs, including dual-encoder architectures (Radford et al., 2021; Jia et al., 2021), encoder-decoder architectures (Cho et al., 2021; Wang et al., 2021; Chen et al., 2022b). Various pre-training objectives have also been introduced, such as image-text contrastive learning (Radford et al., 2021; Yao et al., 2021; Li et al., 2021, 2022b), the image-text matching (Ju et al., 2021; Li et al., 2022b; Bao et al., 2021), and masked language modeling (Li et al., 2021, 2022b; Yu et al., 2022; Wang et al., 2022b). However, these end-to-end models are inflexible to leverage readily available pre-trained models, such as LLMs (Brown et al., 2020; Zhang et al., 2022a; Chung et al., 2024).

Recent approaches in vision-language pre-training have adopted the strategy of utilizing off-the-shelf pre-trained models and keeping them frozen during training. Some methods freeze only the image encoder (Chen et al., 2020; Li et al.,

2020; Zhang et al., 2021), and recent LiT (Zhai et al., 2022) which uses a frozen pre-trained encoder for CLIP (Radford et al., 2021) pre-training, while others freeze the language model to leverage knowledge from language-only pre-trained models for vision-to-language generation tasks (Tsimpoukelli et al., 2021; Alayrac et al., 2022; Chen et al., 2022a; Mañas et al., 2022; Tiong et al., 2022; Guo et al., 2022). The primary challenge lies in aligning visual features with text space. To address this challenge, techniques like Frozen (Tsimpoukelli et al., 2021) finetune image encoders or insert new cross-attention layers into language models to incorporate visual features. BLIP-2 (Li et al., 2023) employs both frozen image encoders and language models for vision-language tasks, achieving strong performance.

Early Exits: To minimize inference latency in deep neural networks, BranchyNet (Teerapittayanon et al., 2016) introduced attaching exits classifiers at intermediary layers. This concept was extended by Shallow-deep (Kaya et al., 2019), which effectively determines when to exit based on confidence distribution at each exit classifier. Approaches like (Huang et al., 2017; Yang et al., 2020; Han et al., 2023) improve EEs for image tasks by dynamically choosing different depths for different regions of the image. Approaches like (Phuong and Lampert, 2019) have employed knowledge distillation for image classification.

For NLP tasks, several early exit frameworks have emerged (Xin et al., 2020; Liu et al., 2021; Zhou et al., 2020; Liu et al., 2020; Wang et al., 2019, 2020; Zhu, 2021; Ji et al., 2023; Zhang et al., 2022b; Bajpai and Hanawal, 2024b; Bajpai et al., 2023, 2024), primarily built on the BERT backbone. DeeCap (Fei et al., 2022) introduces EE for image captioning tasks, employing an imitation network to replicate outputs from transformer layers. MuE (Tang et al., 2023) applies EE to OFA (Wang et al., 2022a), a VLM tailored for multi-modal applications. CapEEN (Bajpai and Hanawal, 2024a) makes the EEs robust to noise by adapting to the exiting threshold.

The key differences in our work are: 1) We employ adversarial training for efficient learning of EE models. 2) Our method can work under both supervised and unsupervised setups by utilizing the zero-shot capabilities of the VLMs, while previous methods require a good amount of high-quality labeled training data.

3 Methodology

We begin with a pre-trained Vision-Language Model (VLM) comprising three key components: a visual encoder, a projection layer, and a language model (LM). The visual encoder processes the input image, which may be a standalone image encoder (Wang et al., 2022a) or include an image-grounded text encoder (e.g., Q-Former in BLIP-2). The projection layer then transforms the visual features to align with the LM's input space which is passed as an input to the decoder. As autoregressive decoding requires multiple forward passes of LM, our focus will primarily be on optimizing the LM component for efficient inference.

We assume that LM consists of N layers, and we attach exits to the K chosen layers. Each exit consists of one Exit Transformer (ET) layer with the same configuration as one LM layer and an Exit Classifier (EC). The exit layer is such the parameters of the ET are trainable, and those of EC are frozen. Before indulging into architectural details, we motivate our method by highlighting the major concerns on the well-known BLIP-2 model.

3.1 Motivation

In Fig. 3, we show the VQA accuracy for various layers on the VQAv2 dataset on the BLIP-2 model. In the left side of the figure, we show the performance of vanilla EE methods where just an EC is attached to the intermediate layer output. As seen, performance dips at the middle layers after the initial improvement. This is due to the LM component, which is of large size and kept frozen during training. During pre-training, BLIP-2 aligns the features of text and images using the Q-former, the querying transformer. However, the Q-former provides the image-grounded text embeddings to the LM component in such a way that it produces high-quality predictions only at the final layer and not the intermediary layer.

To address this, we enhance exits by incorporating a transformer layer instead of relying solely on an EC. This layer mimics the final layer's behavior, enabling intermediate exits to access deep-level representations. By doing this, we are not directly passing the information of intermediate layers to the classifiers instead we first enhance the information through ETs and then pass the hidden representations to classifiers which improves the performance of exits. The right side of Fig. 3 demonstrates how our approach reduces mid-crisis

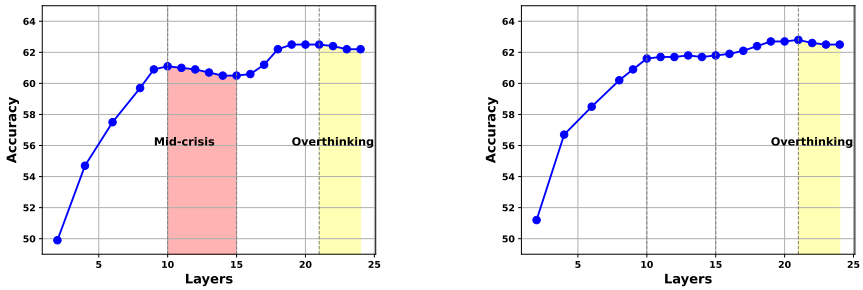


Figure 3: Left: VQA accuracy on the VQAv2 dataset with BLIP-2-ViT-g-FlanT5$_{XL}$ showcasing mid-crisis and overthinking. Right: Our training process improves the mid-crisis. Overthinking can then be solved using EE.

by leveraging deeper layer information. Overthinking is further alleviated by performing inference through EEs using thresholds.

Previous EE methods that attach a trainable classifier at exits introduce substantial parameter overhead, given the large vocabulary size. For instance, a single classifier added to BLIP-2 with OPT$_{2.7B}$ contributes $130M$ trainable parameters, and 7 exits would scale this up to $900M$ parameters. In contrast, our method only trains the LM layer at each exit, adding $63M$ parameters per exit and $588M$ for 7 exits—reducing the trainable parameters by approximately 52% (see Appendix D.1). This efficiency is achieved by using the final layer classifier and training only the LM layer. These issues also translate to other VLMs such as MiniGPT (Zhu et al., 2023) and InstructBLIP (Dai et al., 2024) due to similar architectures.

We next discuss our method which consists of two parts: 1) backbone finetuning and 2) exit training.

3.2 Backbone finetuning

The backbone is fine-tuned using the cross-entropy loss between the predicted token and the ground truth token. The loss function for fine-tuning could be formulated as:

$$\mathcal{L}(I) = \sum_{t=1}^{T} \log P_N(y_t^* | y_{1:t-1}^*, I) \qquad (1)$$

where I denotes the input image, $y_{1:T}^*$ is the true caption and T is the caption length. P_N denote the probability output from the final layer. In this step, the backbone learns to produce high-quality features at the final layer and a classifier C_N to map the feature representations at the final layer to class probabilities. Note that C_N is part of the backbone. Once the fine-tuning is complete, we freeze the parameters of the backbone. This is done to maintain the optimality of the backbone.

3.3 Exits training

After fine-tuning the backbone, we attach K exits to the LM component of the VLM. We denote the set of exit indices by $[K]$. At each exit, we use a feature classifier D_i that discriminates the feature representations of the transformer layer of the ith exit from that of the final layer. Specifically, it provides a score to an input feature representation, whether it is from the final layer. We have a separate feature classifier for every exit as feature representations at different exits can differ.

In our setup, the feature classifier acts as a discriminator, and the exit transformer layer as a generator; the goal of the transformer layer is to generate feature representations similar to that of the final layer. We train them alternately as in the original GAN framework. This training problem can be set up as an unconstrained optimization problem. Let E_i denote the transformer layer in the ith exit. The feature classifier loss for an exit $i \in [K]$ and an input image I could be formulated as:

$$\mathcal{L}_i^{fc}(h_t^i, h_t^N \mid y_{1:t-1}^*, I) = -\log D_i(h_t^N \mid y_{1:t-1}^*, I) \\ -\log(1 - D_i(E_i(h_t^i \mid y_{1:t}^*, I)))$$

where h_t^i is the feature (hidden) representation at ith layer and h_t^N is the feature representation at the final layer of the LM for the tth token in the sentence. The overall loss across all exits could be written as $\mathcal{L}^{fc} = \sum_{i \in [K]} \mathcal{L}_i^{fc}$. It simultaneously updates the feature classifiers across all the exits.

The generator loss for the transformer layer in ith exit could be formulated as:

$$\mathcal{L}_i^{gen}(h_t^i | y_{1:t-1}^*, I) = -\log D_i(E_i(h_t^i | y_{1:t-1}^*, I)))$$

However, because the weights of the transformer layer of the exits are untied from the original backbone, they can face the issue of catastrophic for-

getting or mode collapse. To circumvent these issues, we utilize the small labeled dataset to finetune the backbone. It guides the model to correct the learning trajectory and not let it get stuck to the local minima. The labeled data not only removes the issue of catastrophic forgetting but also helps in reducing overthinking as exits mimic the final layer and learn from the hard labels. The loss could be written as:

$$\mathcal{L}_i^{CE}(I, y^*_{1:t-1}) = \log P_i(y^*_t | y^*_{1:t-1}, I) \quad (2)$$

where P_i denotes the probability score at the ith exit. The loss at exit i will be $\mathcal{L}_i = \mathcal{L}_i^{CE} + \mathcal{L}_i^{gen}$. The overall loss for exits training will be $\mathcal{L} = \sum_{i \in [K]} \mathcal{L}_i$. After this step, the backbone is learned with attached exits.

3.4 Unsupervised setup

Recall from the previous section that labeled data was utilized to reduce the issue of catastrophic forgetting and mode collapse. As the VLMs has good zero-shot performance, we can utilize it to either distill the knowledge at the final layer or create a small set of pseudo labels to fulfill the requirements of the labeled dataset. We provide two methods for unsupervised learning.

Using Knowledge Distillation: In this case, we can directly utilize the soft labels from the final layer to distill the knowledge to the exits. The knowledge distillation loss for the ith layer could be formulated as:

$$\mathcal{L}_i^{KL} = KL(p_t^i, p_t^N) \quad (3)$$

where $p_t^i = C_N(E_i(h_t^i | y^*_{1:t-1}, I))$, $p_t^N = C_N(h_t^N | y^*_{1:t-1}, I)$ and KL is the KL-divergence loss defined as $KL(p_t^i, p_t^N) = \sum_{v \in \mathcal{V}} p_t^i \log \frac{p_t^i}{p_t^N}$ where $\mathcal{V}$ is the vocabulary. We can train the exits by replacing the $\mathcal{L}_i^{CE}$ by $\mathcal{L}_i^{KL}$. Then the overall loss for exit i is $\mathcal{L}_i = \mathcal{L}_i^{KL} + \mathcal{L}_i^{gen}$.

This method also utilizes the zero-shot capabilities of the VLMs model. However, this method has a slightly lower performance than the CapFilt method proposed next, still, it comes with lower computational cost and has comparable performance to vanilla VLM inference.

Using CapFilt: CapFilt (Li et al., 2022b, 2023) is a method that is used in both the original BLIP and BLIP-2 models to generate high-quality synthetic captions. We use similar ideas to generate the labeled dataset. In this step, a sample is passed through the BLIP-2 model, which then provides us with 10 possible captions for the given samples. We then use the CLIP ViT-L/14 model to rank the synthetic captions based on the image-text similarity produced by the CLIP model. We then keep the top-2 captions and keep them as synthetic captions that can be later utilized for training the exits by treating the synthetic captions as true captions. Creating synthetic captions using the CapFilt is more accurate but computationally heavy (Li et al., 2022b). Hence it is recommended when the computational resources are available.

3.5 Inference

We perform captioning in an autoregressive manner. This entails making a token-by-token prediction for a given image, where the layer at which the token is predicted is determined by the confidence score $S_i = \max_{v \in \mathcal{V}} C_N(E_i(h_t^i | y_{1:t-1}, I))(v)$ where $\mathcal{V}$ is the vocabulary. The input to the decoder is processed sequentially through the decoder layers until the confidence score S_i is greater than a predefined threshold value α. The inference starts with the begin of the sentence token and the next token is predicted either at the exits or at the final layer. The inference process stops when the end of the sentence token is predicted either at intermediary layers or at the final layer. Note that if the prediction confidence is below α for all the exits, then the sample is predicted at the final layer, irrespective of the confidence in the prediction.

4 Experiments

Dataset and Metric: We evaluate the performance of our method using the COCO (Lin et al., 2014) and NoCaps dataset (Agrawal et al., 2019) for image captioning. For Visual Question-answering tasks, we utilize the VQAv2 (Goyal et al., 2017), OK-VQA (Marino et al., 2019) and GQA (Hudson and Manning, 2019) datasets. For visual dialogue, we use the VisDial (Das et al., 2017) dataset. We report key metrics including BLEU-4 (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), CIDEr (Vedantam et al., 2015) and SPICE (Anderson et al., 2016) scores for captioning. For VQA tasks, we report the VQA accuracy and for the Visual Dialog, we use the Mean Reciprocal Rank (MRR) (Dai et al., 2024). To effectively consider the speedup, we define the speedup as inverse of the fraction of parameters used for inference on average. The

Models	#Train Params	NoCaps Zero-shot								Spd
		in-domain		near-domain		out-domain		full-dataset		
		C	S	C	S	C	S	C	S	
Vin VL	345M	102.9	14	94.8	13.7	88.1	11.9	95.1	13.2	-
BLIP	446M	114.9	15.2	110.6	14.6	114.8	14.3	112.8	14.7	-
SimVLM	1.4B	113.7	14.9	110.6	14.2	114.6	14.4	112.1	14.3	-
BLIP-2 ViT $O_{2.7B}$	1.1B	123.0	15.8	117.8	15.4	123.2	15.0	119.6	15.4	1.07×
BLIP-2 ViT $FT5_{XL}$	1.1B	123.7	16.3	**120.2**	**15.9**	124.8	15.1	121.6	15.8	1.00×
Early Exit models										
DeeBLIP	1.8B	115.2	15.3	111.5	14.7	115.4	14.5	112.4	14.5	1.41×
PABEE-BLIP	1.8B	117.7	15.4	114.2	14.8	117.6	14.7	112.9	14.6	1.29×
LeeBLIP	1.8B	119.4	15.5	115.8	14.8	120.1	14.9	116.3	15.1	1.38×
MuE	1.8B	118.1	15.4	115.3	14.8	118.7	14.8	114.8	14.9	1.44×
FREE ViT $O_{2.7B}$	1.5B	122.7	15.7	118.1	15.5	123.9	15.1	119.9	15.6	**1.63**×
FREE ViT $FT5_{XL}$	1.4B	**124.3**	**16.5**	120.0	**15.9**	**125.5**	**15.4**	**122.7**	**16.1**	1.51×

Table 1: Results on the Nocaps dataset during zero-shot transfer when the model is trained on the COCO dataset across various domains. Spd is the speedup achieved by the model. $O_{2.7B}$ is $OPT_{2.7B}$ and $FT5_{XL}$ is $FlanT5_{XL}$.

Models	#Train Params	VQAv2 train	VQAv2 test	Spd
Without Exits				
ALBEF	314M	72.3	71.5	-
BLIP	385M	73.9	72.1	-
OFA	930M	75.7	75.6	-
Flamingo80B	10.6B	77.9	78.1	-
BLIP-2 V-O	1.2B	78.3	78.5	1.07×
BLIP-2 V-F	1.2B	78.8	78.7	1.00×
Early Exit models (on BLIP-2)				
DeeBLIP	1.9B	75.4	75.9	1.52×
PABEE-BLIP	1.9B	77.4	77.1	1.39×
LeeBLIP	1.9B	78.1	77.8	1.65×
FREE-V-O	1.6B	78.7	79.0	**1.77**×
FREE-V-F	1.5B	**78.9**	**79.1**	1.71×

Table 2: Results of semi-supervised application of our model to Visual-Question Answering tasks.

speedup is formulated as:

$$\text{Speedup} = \frac{\text{Total parameters}}{\text{Average number of parameters used}}$$

where the average number of parameters could be written as $\frac{1}{M}\sum_{I=1}^{M}\sum_{i=1}^{N_I} w_i \times (i+k) \times p$ where p denotes the number of parameters in one layer of the LM component, N_I denotes the number of predicted words in the caption for the image I, M denotes the total number of input images and $k = 1$ is the number of LM layers in the exit. Total parameters denote the total number of parameters in the backbone. The baseline for comparing speedup for BLIP-2 models is BLIP-2 ViT-g $FlanT5_{XL}$. We only compare the speedup of early exiting methods and the BLIP-2 variants. In the Appendix B and Table 5 and 8, we show the results of the MiniGPT (Zhu et al., 2023) model and in Appendix C and Table 7, we show the results on the InstructBLIP (Liu et al., 2024) model.

Training: In our setup for BLIP-2, we utilize two variations of the BLIP-2 model with the same image encoder (ViT-g/14 (Dosovitskiy et al., 2020)) and frozen LLMs that are OPT-2.7B (Zhang et al., 2022a) and FlanT5-XL (Chung et al., 2024). We use the LAVIS (Li et al., 2022a) library for implementation, training and evaluation. For training, we use the validation split of the datasets. We use 80% of validation split for training and the remaining 20% for development.

We use labels of the validation dataset when the task is semi-supervised, else we just use the samples without labels. First, the backbone is fine-tuned for 10 epochs with a starting learning rate of 1e-5, which decays by 0.5 every 3 epochs. The backbone weights are then frozen post-fine-tuning and exits are attached to the backbone. We train exit weights for a further 3 epochs. We employ the Adam (Kingma and Ba, 2014) optimizer and a batch size of 16. Similar to Bajpai and Hanawal (2024c), we use a feature classifier, with one hidden linear layer with a hidden state of size 3072 and a LeakyReLU activation function.

For the unsupervised tasks, we train the model for 5 epochs on the validation dataset (without labels) with knowledge distillation from the final layer. Optimizers and learning rates are kept the same as given above. Note that in CapFilt we apply the CapFilt method on the validation dataset (without labels) and generate synthetic labels. After this, we perform a similar procedure by treating the synthetic labels as the true labels as done for the semi-supervised tasks. We do not finetune the backbone in an unsupervised setup.

Inference: Inference is conducted with a batch size of 1. We provide the results on the test dataset. For image captioning, we use a prompt as 'a photo of'. The threshold is cho-

sen as the best-performing threshold from the set $\{0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$ on the held-out split of the validation dataset. More details on the hyperparameter setting can be found in Table 10 and in the Appendix A.2 with the values of hyperparameters. All experiments were performed with a combination consisting of two NVIDIA RTX A6000 and four NVIDIA GeForce RTX 3080 Ti GPUs.

Baselines: We establish baseline models for performance evaluation, including vanilla BLIP-2 inference. Additionally, we compare with multimodal models VinVL (Zhang et al., 2021), ALBEF (Li et al., 2021), SimVLM (Wang et al., 2021), OFA (Wang et al., 2022a), and Flamingo (Alayrac et al., 2022). We also assess state-of-the-art early exit methods originally proposed for the BERT backbone, such as DeeBERT (Xin et al., 2020), PABEE (Zhou et al., 2020), and LeeBERT (Zhu, 2021), adapted to the BLIP-2 backbone as DeeBLIP, PABEE-BLIP, and LeeBLIP, respectively. DeeBLIP uses confidence-based exiting, PABEE-BLIP employs patience-based exiting, and LeeBLIP combines knowledge distillation from the final layer with hard label learning. Furthermore, we apply the MuE (Tang et al., 2023) early exiting method to the BLIP-2 backbone, using exits from the better-performing BLIP-2 variant for our baselines.

5 Results

Visual Question Answering: We provide results on unsupervised (see table 3) as well as semi-supervised setups (see table 2). We observe that our method outperforms all early exit methods in terms of accuracy and speedup even with less number of trainable parameters. We even outperform the vanilla BLIP-2 inference due to overthinking in the BLIP-2 backbone which is mitigated by our input-adaptive inference. We also provide results on an unsupervised visual dialogue dataset where the task is similar to VQA but there is an additional context before the question i.e. a dialogue history between the user and the model.

Image Captioning: We provide results of semi-supervised and unsupervised setup in table 1 and 4 respectively. We clearly outperform the existing models in terms of both accuracy as well as speedup. For the NoCaps dataset, the model is fine-tuned on the COCO dataset. The speedup for NoCaps dataset is lower as there is a domain change from the training which lowers the confidence in prediction taking more samples to deeper exits for inference.

We observe performance improvement over previous baselines as we attached exits to the BLIP-2 model and by performing input-adaptive inference, we perform better than the BLIP-2 model, and as BLIP-2 outperforms other models, FREE also outperforms others. For the early exiting baselines on BLIP-2, we outperform them as we have an additional component in the exits rather than just a linear classifier which helps in better performance of exits in terms of both performance and speedup. Note that there is a decrease in accuracy when we are in an unsupervised setup, as our model mimics the final layer hence some amount of overthinking still remains. Still, we are comparable to the BLIP-2 inference. On the other hand, the labeled dataset in semi-supervised tasks helps the model learn the hardness of the incoming sample. This helps the model to overcome the overthinking issue.

We have utilized two versions of the BLIP-2 model that have decoder as $\text{FlanT5}_{\text{XL}}$ and $\text{OPT}_{\text{2.7B}}$. We observe that the speedup in BLIP-2 with $\text{OPT}_{\text{2.7B}}$ was higher as there are more layers in this hence they are more susceptible to overthinking issues. The speedup for VQA tasks was higher as these tasks are simpler than image captioning tasks. We have not reported the speedup of the models other than the variants of BLIP-2 as they have different types of architectures. In terms of speedup, our objective is to make BLIP-2 faster.

In table 9, we have shown the result of using the CapFilt method to generate synthetic captions in the absence of the labeled dataset. We have reported the CIDEr score over the NoCaps dataset. We can observe that the model has improved upon the performance using CapFilt and the speedup has significantly increased. This effect is due to the good quality captions that help the exits learn better, hence it outputs more samples early increasing the speedup as compared to knowledge distillation.

6 Conclusion

In this study, we introduced a novel inference technique FREE, which leverages adversarial training of exits alongside the zero-shot capabilities of the VLMs. By employing FREE, we effectively reduce the dependency on a vast amount of labeled training data typically required for exit training.

Model	#Total params	VisDial test	VQAv2 train	VQAv2 test	OK-VQA test	GQA test	VizWiz test	Speed
Without exits								
Flamingo3B	3.2B	-	53.2	49.4	41.5	-	28.9	1.28×
Flamingo9B	9.3B	-	55.7	51.8	44.7	-	28.8	0.44×
Flamingo80B	80B	-	59.1	56.2	**50.4**	-	**31.5**	0.05×
BLIP-2 ViT-g $OPT_{2.7B}$	3.8B	34.1	54.6	52.0	31.2	34.2	27.0	1.07×
BLIP-2 ViT-g $OPT_{6.7B}$	7.8B	37.5	55.9	53.7	36.1	36.4	27.2	0.52×
BLIP-2 ViT-g $FlanT5_{XL}$	4.1B	**45.9**	**64.9**	**62.5**	40.6	**44.5**	29.8	1.00×
Early Exit models (on BLIP-2 ViT-g $FlanT5_{XL}$)								
DeeBLIP	4.7B	33.4	41.3	42.8	23.4	27.8	20.1	1.39×
PABEE-BLIP	4.7B	35.7	49.6	51.3	31.2	34.3	23.6	1.31×
LeeBLIP	4.7B	39.1	57.7	57.1	37.1	39.7	26.4	1.29×
MuE	4.7B	36.6	55.4	53.6	33.7	37.1	24.7	1.36×
FREE ViT-g $OPT_{2.7B}$	4.3B	32.3	55.5	53.4	35.6	44.7	26.8	**1.51**×
FREE ViT-g $FlanT5_{XL}$	4.5B	**45.5**	**64.5**	**62.1**	**40.3**	**44.0**	**29.5**	1.45×

Table 3: Results of the unsupervised Visual-Question Answering and VisDial dataset. For VQA tasks, we report the VQA accuracy and for the visual dialogue, we report the Mean Reciprocal Rank(MRR).

Models	COCO Karpathy test				
	B@4	C	S	M	Spd
OFA	37.5	130.3	25.2	31.1	-
Flamingo	38.5	134.1	24.1	27.8	-
SimVLM	38.6	138.3	24.8	29.8	-
BLIP-2-V-O	41.7	139.8	25.5	30.5	1.07×
BLIP-2-V-F	40.4	141.5	25.2	29.1	1.00×
Early Exit models					
DeeBLIP	32.8	115.1	20.9	25.3	1.65×
PABEE-BLIP	34.2	119.8	21.4	26.2	1.45×
LeeBLIP	37.4	132.0	22.8	27.6	1.59×
MuE	37.9	137.5	23.6	28.5	1.41×
FREE-V-O	**41.9**	**142.5**	**25.2**	**30.8**	**1.75**×

Table 4: Results of semi-supervised training on the Karpathy test split of the COCO dataset.

Our approach involves adversarially training exits to generate representations similar to those of the final layer, thereby minimizing the need for extensive labeled data. Moreover, our exit design reduces the number of trainable parameters, resulting in lower computational costs. Experimental results demonstrate that our method significantly enhances inference speed while yielding high-quality outputs.

7 Limitations

For attaching exits to a large model such as BLIP-2, the crucial part is to decide where to attach exits within a given budget, i.e., what could be the best places for an exit in the LM component of the backbone without exceeding a certain amount of parameters. We answered that question by explaining the mid-crisis. However, the placements of exits with given budget criteria still remain unexplored, which can make these models even faster within computational boundaries.

Acknowledgements

Divya Jyoti Bajpai is supported by the Prime Minister's Research Fellowship (PMRF), Govt. of India. Manjesh K. Hanawal thanks funding support from SERB, Govt. of India, through the Core Research Grant (CRG/2022/008807) and MATRICS grant (MTR/2021/000645), and DST-Inria Targeted Programme. We also thank funding support from Amazon IIT-Bombay AI-ML Initiative (AIAIMLI).

References

Harsh Agrawal, Karan Desai, Yufei Wang, Xinlei Chen, Rishabh Jain, Mark Johnson, Dhruv Batra, Devi Parikh, Stefan Lee, and Peter Anderson. 2019. Nocaps: Novel object captioning at scale. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8948–8957.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. 2022. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. Spice: Semantic propositional image caption evaluation. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V 14*, pages 382–398. Springer.

Divya J Bajpai, Vivek K Trivedi, Sohan L Yadav, and Manjesh K Hanawal. 2023. Splitee: Early exit in deep neural networks with split computing. *arXiv preprint arXiv:2309.09195*.

Divya Jyoti Bajpai and Manjesh Kumar Hanawal. 2024a. Capeen: Image captioning with early

exits and knowledge distillation. *arXiv preprint arXiv:2410.04433*.

Divya Jyoti Bajpai and Manjesh Kumar Hanawal. 2024b. Ceebert: Cross-domain inference in early exit bert. In *To appear in proceedings of the 62nd conference of the Association for computational linguistics: Findings Volume*.

Divya Jyoti Bajpai and Manjesh Kumar Hanawal. 2024c. Dadee: Unsupervised domain adaptation in early exit plms. *arXiv preprint arXiv:2410.04424*.

Divya Jyoti Bajpai, Aastha Jaiswal, and Manjesh Kumar Hanawal. 2024. I-splitee: Image classification in split computing dnns with early exits. *arXiv preprint arXiv:2401.10541*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Hangbo Bao, Wenhui Wang, Li Dong, Qiang Liu, Owais Khan Mohammed, Kriti Aggarwal, Subhojit Som, and Furu Wei. 2021. Vlmo: Unified vision-language pre-training with mixture-of-modality-experts. *arXiv preprint arXiv:2111.02358*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Jun Chen, Han Guo, Kai Yi, Boyang Li, and Mohamed Elhoseiny. 2022a. Visualgpt: Data-efficient adaptation of pretrained language models for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18030–18040.

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

Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. Uniter: Universal image-text representation learning. In *European conference on computer vision*, pages 104–120. Springer.

Jaemin Cho, Jie Lei, Hao Tan, and Mohit Bansal. 2021. Unifying vision-and-language tasks via text generation. In *International Conference on Machine Learning*, pages 1931–1942. PMLR.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. 2018. Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65.

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale N Fung, and Steven Hoi. 2024. Instructblip: Towards general-purpose vision-language models with instruction tuning. *Advances in Neural Information Processing Systems*, 36.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José MF Moura, Devi Parikh, and Dhruv Batra. 2017. Visual dialog. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 326–335.

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. *arXiv preprint arXiv:2010.11929*.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. 2024. Layer skip: Enabling early exit inference and self-speculative decoding. *arXiv preprint arXiv:2404.16710*.

Zhengcong Fei, Xu Yan, Shuhui Wang, and Qi Tian. 2022. Deecap: Dynamic early exiting for efficient image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12216–12226.

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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913.

Jiaxian Guo, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Boyang Li, Dacheng Tao, and Steven CH Hoi. 2022. From images to textual prompts: Zero-shot vqa with frozen large language models. *arXiv preprint arXiv:2212.10846*.

Yizeng Han, Dongchen Han, Zeyu Liu, Yulin Wang, Xuran Pan, Yifan Pu, Chao Deng, Junlan Feng, Shiji Song, and Gao Huang. 2023. Dynamic perceiver for efficient visual recognition. *arXiv preprint arXiv:2306.11248*.

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*.

Drew A Hudson and Christopher D Manning. 2019. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709.

Yixin Ji, Jikai Wang, Juntao Li, Qiang Chen, Wenliang Chen, and Min Zhang. 2023. Early exit with disentangled representation and equiangular tight frame. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 14128–14142.

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. 2021. Scaling up visual and vision-language representation learning with noisy text supervision. In *International conference on machine learning*, pages 4904–4916. PMLR.

Weiyu Ju, Wei Bao, Dong Yuan, Liming Ge, and Bing Bing Zhou. 2021. Learning early exit for deep neural network inference on mobile devices through multi-armed bandits. In *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 11–20. IEEE.

Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. 2019. Shallow-deep networks: Understanding and mitigating network overthinking. In *International conference on machine learning*, pages 3301–3310. PMLR.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Dongxu Li, Junnan Li, Hung Le, Guangsen Wang, Silvio Savarese, and Steven CH Hoi. 2022a. Lavis: A library for language-vision intelligence. *arXiv preprint arXiv:2209.09019*.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. 2022b. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pages 12888–12900. PMLR.

Junnan Li, Ramprasaath Selvaraju, Akhilesh Gotmare, Shafiq Joty, Caiming Xiong, and Steven Chu Hong Hoi. 2021. Align before fuse: Vision and language representation learning with momentum distillation. *Advances in neural information processing systems*, 34:9694–9705.

Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, et al. 2020. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXX 16*, pages 121–137. Springer.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024. Visual instruction tuning. *Advances in neural information processing systems*, 36.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Haotang Deng, and Qi Ju. 2020. Fastbert: a self-distilling bert with adaptive inference time. *arXiv preprint arXiv:2004.02178*.

Xiangyang Liu, Tianxiang Sun, Junliang He, Lingling Wu, Xinyu Zhang, Hao Jiang, Zhao Cao, Xuanjing Huang, and Xipeng Qiu. 2021. Towards efficient nlp: A standard evaluation and a strong baseline. *arXiv preprint arXiv:2110.07038*.

Oscar Mañas, Pau Rodriguez, Saba Ahmadi, Aida Nematzadeh, Yash Goyal, and Aishwarya Agrawal. 2022. Mapl: Parameter-efficient adaptation of unimodal pre-trained models for vision-language few-shot prompting. *arXiv preprint arXiv:2210.07179*.

Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. 2019. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, pages 3195–3204.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Mary Phuong and Christoph H Lampert. 2019. Distillation-based training for multi-exit architectures. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1355–1364.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish

Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Minho Ryu, Geonseok Lee, and Kichun Lee. 2022. Knowledge distillation for bert unsupervised domain adaptation. *Knowledge and Information Systems*, 64(11):3113–3128.

Shengkun Tang, Yaqing Wang, Zhenglun Kong, Tianchi Zhang, Yao Li, Caiwen Ding, Yanzhi Wang, Yi Liang, and Dongkuan Xu. 2023. You need multiple exiting: Dynamic early exiting for accelerating unified vision language model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10781–10791.

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. IEEE.

Anthony Meng Huat Tiong, Junnan Li, Boyang Li, Silvio Savarese, and Steven CH Hoi. 2022. Plug-and-play vqa: Zero-shot vqa by conjoining large pretrained models with zero training. *arXiv preprint arXiv:2210.08773*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Maria Tsimpoukelli, Jacob L Menick, Serkan Cabi, SM Eslami, Oriol Vinyals, and Felix Hill. 2021. Multimodal few-shot learning with frozen language models. *Advances in Neural Information Processing Systems*, 34:200–212.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Meiqi Wang, Jianqiao Mo, Jun Lin, Zhongfeng Wang, and Li Du. 2019. Dynexit: A dynamic early-exit strategy for deep residual networks. In *2019 IEEE International Workshop on Signal Processing Systems (SiPS)*, pages 178–183. IEEE.

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

Wenhui Wang, Hangbo Bao, Li Dong, Johan Bjorck, Zhiliang Peng, Qiang Liu, Kriti Aggarwal, Owais Khan Mohammed, Saksham Singhal, Subhojit Som, et al. 2022b. Image as a foreign language: Beit pretraining for all vision and vision-language tasks. *arXiv preprint arXiv:2208.10442*.

Zirui Wang, Jiahui Yu, Adams Wei Yu, Zihang Dai, Yulia Tsvetkov, and Yuan Cao. 2021. Simvlm: Simple visual language model pretraining with weak supervision. *arXiv preprint arXiv:2108.10904*.

Zizhao Wang, Wei Bao, Dong Yuan, Liming Ge, Nguyen H Tran, and Albert Zomaya. 2020. Accelerating on-device dnn inference during service outage through scheduling early exit. *Computer Communications*, 162:69–82.

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*.

Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. 2020. Resolution adaptive networks for efficient inference. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2369–2378.

Lewei Yao, Runhui Huang, Lu Hou, Guansong Lu, Minzhe Niu, Hang Xu, Xiaodan Liang, Zhenguo Li, Xin Jiang, and Chunjing Xu. 2021. Filip: Fine-grained interactive language-image pretraining. *arXiv preprint arXiv:2111.07783*.

Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. 2022. Coca: Contrastive captioners are image-text foundation models. *arXiv preprint arXiv:2205.01917*.

Xiaohua Zhai, Xiao Wang, Basil Mustafa, Andreas Steiner, Daniel Keysers, Alexander Kolesnikov, and Lucas Beyer. 2022. Lit: Zero-shot transfer with locked-image text tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18123–18133.

Pengchuan Zhang, Xiujun Li, Xiaowei Hu, Jianwei Yang, Lei Zhang, Lijuan Wang, Yejin Choi, and Jianfeng Gao. 2021. Vinvl: Revisiting visual representations in vision-language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5579–5588.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022a. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Zhen Zhang, Wei Zhu, Jinfan Zhang, Peng Wang, Rize Jin, and Tae-Sun Chung. 2022b. Pcee-bert: accelerating bert inference via patient and confident early exiting. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 327–338.

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.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. 2023. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*.

Wei Zhu. 2021. Leebert: Learned early exit for bert with cross-level optimization. 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 2968–2980.

A Appendix

A.1 Ablation Study

Qualitative analysis: In Figure 1, we provide some examples of the output provided by the FREE model. The figure shows how the early exit models increase the speedup by predicting easier tokens earlier. For instance, the image in the last row and last column of the figure is an example of an easy sample where the tokens are predicted at initial layers. Similarly, for the image with a zebra, it can easily predict the easier token such as '*A zebra standing in a snowy field*' at the initial layers while the part of the image that is not easy to predict '*with a wall behind*' is predicted at deeper layers and predicting a high-quality caption overall while speeding up inference using the easiness of sample as well as token.

Also note that the common tokens are mostly predicted from the initial layers while the rare tokens are predicted from the deeper layers. This observation suggests that the tokens that occur more number of times are considered as easy by the model while the token that have rare appearances are the ones treated as hard.

A.2 Accuracy vs speedup

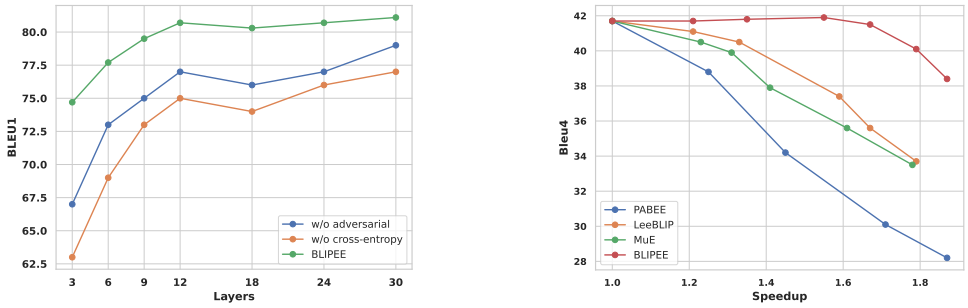
In figure 4b, we show the accuracy vs speedup curve which could be obtained by changing the threshold parameter α. As we decrease the threshold parameter, samples exit from the initial layers even with less confidence, in this way all the samples are more prone to be incorrect decreasing the accuracy but as the threshold is decreased more samples exit from the initial layers and increase the speedup. One key observation is as we start decreasing the threshold, we observe that sometimes the performance even increases, this is the effect of overthinking, where some samples are correctly predicted at initial layers and might become wrong as they reach the final layer. We have also plotted the curves for other exiting methods and observed that our method has better stability as compared to other early exiting methods.

A.3 Importance of different components

In figure 4a, we show the importance of different components of our method. We observe that there is a huge performance drop if we remove the knowledge distillation or cross-entropy loss from the overall loss function. This occurs due to catastrophic forgetting or mode collapse where

Model	COCO Karpathy test split				Speed
	BLEU4	CIDEr	S	M	
MiniGPT	38.6	133.9	24.2	30.8	1.00×
DeeMini	31.4	107.0	20.1	25.9	1.42×
PABEE-Mini	33.5	117.3	20.9	27.3	1.40×
LeeMini	37.3	126.8	22.8	29.1	1.49×
MuE	37.8	129.5	23.2	29.7	1.52×
FREE	**38.3**	**132.2**	**23.9**	**30.5**	**1.67**×

Table 5: Results of MiniGPT-4 model on the COCO Karpathy test split.



(a) Different combinations of loss function

(b) Speedup vs accuracy curve.

Figure 4: Left: BLEU-1 score for COCO with model BLIP-2-ViT-L-OPT$_{2.7B}$ with different components. Right: Speedup vs BLEU-4 curve for COCO dataset with ViT-L-OPT$_{2.7B}$.

Noise	BLEU-4	METEOR	CIDEr	Speed
		$\sigma = 0.0$		
BLIP-2	40.4	29.1	141.5	1.00×
BLIPEE	40.5	29.1	142	1.77×
		$\sigma = 0.5$		
BLIP-2	38.6	28	137.9	1.00×
BLIPEE	39.8	28.8	139.5	1.64×
		$\sigma = 1.0$		
BLIP-2	35.3	26.4	130.8	1.00×
BLIPEE	36.7	27.9	134.2	1.59×
		$\sigma = 2.0$		
BLIP-2	28.6	20.5	112.6	1.00×
BLIPEE	31.1	22.7	120.8	1.45×

Table 6: Results on BLIP-2 and FREE on COCO dataset when some level of noise σ is added into the images during inference phase.

the model gets stuck into local minima. On the other hand, if we remove the adversarial training part, there is again a performance drop, as we only train the classifier but we are not mapping the feature representations of the final layer and the exits hence exits only have low-level features which is insufficient to make correct predictions, hence resulting in a performance drop.

A.4 Robustness of FREE

In table 6, we provide results to prove the robustness of FREE. To obtain these results, we perform inference on images with Gaussian noise σ added to it. The higher the value of σ, the more noise is present in the image. Observe from the table that when the level of noise is increased in the image the performance of the BLIP-2 model is affected. However, the impact of the noise is smaller for FREE.

The reason for the smaller impact of noise on FREE is that FREE uses outputs from multiple exits and performs inference only when a classifier is confident enough. This makes it robust to noise present during the inference phase.

B Results on MiniGPT

In Table 5 and 8, we show the results of FREE on the MiniGPT backbone. First note that the performance of MiniGPT is not as good as the BLIP-2 model. The reason is stated in the Appendix of the MiniGPT-4 paper as the number of training parameters of the model is significantly lower than the full BLIP-2 model. However, it has a better performance over long sequence generation tasks. We use the Llama (Touvron et al., 2023) model in its decoder.

From the results, we observe that our method can be well generalized over MiniGPT as the results are similar to the BLIP-2 backbone. Note that during fine-tuning MiniGPT, we have used similar hyperparameters as used to train BLIP-2 as the overall architecture of MiniGPT is similar to BLIP-2 except for the number of training param-

Model	COCO Karpathy test split				Speed
	BLEU4	CIDEr	S	M	
InstructBLIP	42.6	140.5	24.3	31.2	1.00×
DeeIB	35.2	123.7	20.8	26.8	1.39×
PABEE-IB	36.5	126.3	21.7	28.0	1.35×
LeeIB	38.0	132.8	22.9	29.9	1.47×
MuE	39.4	136.2	23.4	30.6	1.42×
FREE	**42.1**	**138.9**	**23.8**	**31.0**	**1.58**×

Table 7: Results of InstructBLIP model on the COCO Karpathy test split.

Model	VQA	OKVQA	GQA	Speed
MiniGPT	68.6	65.2	41.9	1.00×
DeeMini	59.8	59.1	35.7	1.53×
PABEE-Mini	61.4	60.8	36.9	1.49×
LeeMini	65.7	63.2	39.0	1.61×
MuE	67.8	64.6	39.9	1.59×
FREE	**68.2**	**65.1**	**41.6**	**1.63**×

Table 8: Results on MiniGPT for Visual Question Answering tasks.

eters. Note that the baselines are similar to those given in the main body with DeeBERT replaced as DeeMini, PABEE as PABEE-Mini, and LeeBERT as LeeMini. All the hyperparameters were kept the same for all the baselines with their approach applied.

C Results on InstructBLIP

InstructBLIP is an upgraded version of the BLIP-2 model where the underlying architecture is same but the difference is with the instruction given to generate the output. InstructBLIP can perform better than BLIP-2 over long sequence generation tasks however takes a slight hit on the performance of the image captioning tasks. The hyperparameters for fine-tuning InstructBLIP were the same as for BLIP-2.

However, the behavior of our method applied to this model is the same as it shows minimal performance drop as compared to other EE methods while keeping the speedup better than them. This is the result of the access to deeper level knowledge as our method tries to mimic the behavior of the final layers at the exits. Also, we have used only the validation split to train the exits which makes our method less resource-heavy. Note that the baselines are similar to those given in the main body with DeeBERT replaced as DeeIB, PABEE as PABEE-IB, and LeeBERT as LeeIB. All the hyperparameters were kept the same for all the baselines with their approach applied.

Note that both MiniGPT and InstructBLIP share same architectural details as BLIP-2 model with different training strategies, hence we do not explicitly explain the architectural details. Also the training and inference procedure with hyperparameters is similar to the BLIP-2 model.

D Analysis on computational cost

Training stage: In our method, the fine-tuning stage is similar to vanilla fine-tuning of the backbone. After this step, we have additional parameters that are used at the exits consisting of one transformer layer similar to the decoder of the model. While other methods use a classifier at all the exits, note that due to huge size of the vocabulary the classifier size is very large even larger than one transformer layer. Due to this our method has lesser number of training parameters. Now as we train in a GAN-based setup, we alternatively train the exit transformer layer, where it is trained to generate similar representations as the final layer. The generator is the exit transformer layer and a separate feature classifier is also added with a job to discriminate if the features are from the final layer or exit layer. **The full training procedure along with fine-tuning takes approximately** 26 **hours to complete on a setup of** 6 **GPUs with** 4 **NVIDIA RTX 3090 and** 2 **NVIDIA RTX A6000 GPUs while on a similar setup training of vanilla BLIP-2 training requires** 18 **hours of training.**

Inference: While we have a heavy training setup, post deployment the inference process is very simple and fast. Once the image has been passed through the image encoder as well as the query generator. After this step the query embedding are passed through decoder after the projection layer. Then the decoder generates the text autoregressively. The next token is generated at one of the attached exits where the input to the decoder is processed sequentially at the exits. If an exit is confident on its prediction, it stops the inference process by assigning the token generated by it as the next predicted token by the model. This im-

Model	No Caps Zero-shot				
	in-domain	near-domain	out-domain	full-dataset	Spd
w/o CapFilt	122.3	118.9	123.1	120.7	1.45×
Capfilt	123.5	120.4	124.7	122.0	1.77×

Table 9: Difference between CapFilt and Knowledge distillation method for an unsupervised setup.

LLM	OPT	FlanT5
Exit Config	[3, 6, 9, 12, 24, 27, 30]	[3, 5, 7, 9, 12, 20, 22]
AdamW beta	[0.9, 0.999]	[0.9, 0.999]
Threshold	0.8	0.8
Inference beam size	5	5
Warmup Steps	500	500

Table 10: More hyperparameter details of FREE on different LM component in the BLIP-2 model. Note that the thresholds are chosen from the set $\{0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$

Dataset	**Layer 3**	**Layer 6**	**Layer 9**	**Layer 12**	**Layer 24**	**Layer 27**	**Layer 30**
VQA-v2	0.02	0.07	0.13	0.09	0.15	0.18	0.17
Ok-VQA	0.03	0.09	0.11	0.09	0.13	0.15	0.18
VizWiz	0.01	0.06	0.09	0.08	0.11	0.17	0.22

Table 11: Fraction of samples exiting from different layers of the backbone across various datasets.

Component	**Parameters per Layer**	**Total Parameters(32-layers)**
Transformer Layer	∼62.8M	∼2.01B
Layer Norms & Embeddings	-	∼560M
LM Head	-	∼128.7M
Total	-	**2.7B**

Table 12: OPT-2.7B Parameter Breakdown

proves the speed benefits of the full model where not all tokens are processed till the final layer. We observed that more common tokens usually exit at earlier exits while rare tokens are processed deeper into the backbone. As formation of sentence require a large number of common tokens such as 'the', 'and', etc., the early exits makes the generation process significantly faster. The fraction of token exited from different layers is given in Table 11. **The inference time over the Karpathy test split of the COCO dataset was** 7 **minutes of wall clock time approximately while for vanilla BLIP-2, it was approximately** 11 **minutes. This also proves the importance of exits.**

In summary, during training, we have additional exits as compared to vanilla BLIP-2 model as well as feature classifiers at the exits. The feature classifiers are dropped during inference and are not further required. While the attached exits help the model to make adaptive predictions by performing inference using some fraction of layers from the full model based on the input sample and token complexity.

D.1 OPT-2.7B Architecture

OPT-2.7B is a decoder-only transformer model with 2.7 billion parameters. Each transformer layer consists of multi-head self-attention, a feed-forward network (FFN), and layer normalization. The detailed architecture is as follows:

- **Hidden Size** (d_{model}): 2560
- **Number of Layers**: 32
- **Attention Heads**: 32
- **FFN Inner Size**: $4 \times d_{\text{model}} = 10240$
- **LM Head Size**: $2560 \times 50272 \approx 128.7$M parameters

The total parameter breakdown is summarized in Table 12. Now if we use classical exit setup, that uses independent exit classifiers for attaching exits it adds $128.7M$ parameters at every exit. While in our setup, we attach exit transformers and exit classifiers at intermediate layers where the exit classifier have shared weights as final layer classifier and is frozen. Since the exit transformer is a replica of transformer layer of the model, its adds only $62.8M$ parameters. This reduces the training parameter size by 52%.