CapEEN: Image Captioning with Early Exits and Knowledge Distillation

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

Abstract

Deep neural networks (DNNs) have made significant progress in recognizing visual elements and generating descriptive text in image-captioning tasks. However, their improved performance comes from increased computational burden and inference latency. Early Exit (EE) strategies can be used to enhance their efficiency, but their adaptation presents challenges in image captioning as it requires varying levels of semantic information for accurate predictions. To overcome this, we introduce CapEEN to improve the performance of EE strategies using knowledge distillation. Inference in CapEEN is completed at intermediary layers if prediction confidence exceeds a predefined value learned from the training data. To account for real-world deployments, where target distributions could drift from that of training samples, we introduce a variant A-CapEEN to adapt the thresholds on the fly using Multi-armed bandits framework. Experiments on the MS COCO and Flickr30k datasets show that CapEEN gains speedup of 1.77× while maintaining competitive performance compared to the final layer, and A-CapEEN additionally offers robustness against distortions. The source code is available at https://github.com/Div290/CapEEN

1 Introduction

Image captioning, a multifaceted challenge at the intersection of computer vision and natural language processing, has reaped the benefits of deep neural networks (DNNs), characterized by their increased scale and complexity. This task entails not only the identification of visual elements within an image but also the intricate interpretation of their relationships. Notably, the encoder-decoder framework has made significant strides in sentence generation by anticipating the next word in a sequence (Anderson et al., 2018; Chen et al., 2021; Chen and Lawrence Zitnick, 2015; Fei, 2021; Huang et al., 2019; Vinyals et al., 2015; Xu et al., 2021, 2015; Yao et al., 2018; Zhang et al., 2021; Li et al., 2022, 2023). This predictive modeling considers both the image's content and the preceding partial sentence, resulting in substantial progress in the field (Bai and An, 2018). However, their large size restricts deployment in resource-constrained scenarios requiring fast inference. Early Exit (EE) strategies have emerged as a strategic solution to overcome this challenge.

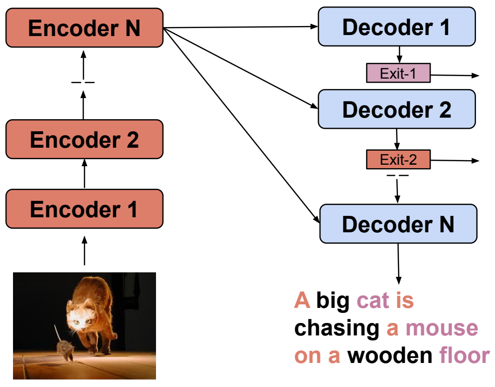


Figure 1: The encoder-decoder framework with attached exits. The figure states that low-level features could be extracted from early classifiers and inferred there, while high-level features are inferred at deeper classifiers. The color of the text in the caption is the same as the color of the classifier after that layer.

In EE strategies, classifiers are attached to the intermediary layers. Each sample can exit from one of them without requiring to pass through all layers (see Fig. 1). This brings down computational requirements and improves latency (Teerapittayanon et al., 2016; Xin et al., 2020). However, this generic approach of attaching exits to the pre-trained backbone may not be suitable for image captioning (Fei et al., 2022)– in the layered hierarchy of representation in transformer-based models, the initial layers focus on extracting low-level features, while deeper layers delve into the complexities of semantic fusion relations (Cornia et al., 2020; Liu et al., 2021c). Consequently, even 'easy samples' require a certain level of high-level information present at deeper layers.

Moreover, the decisions of early exit are based on the confidence at the intermediary layers being above a predefined threshold. The threshold used to compare the confidence levels significantly impacts the amount of latency and accuracy. These thresholds are learned during training and serve as a crucial reference point during inference.

Post-deployment, it may be possible that the distribution of the target sample could drift away from that of training samples. This is often encountered in real-world scenarios, e.g., blurred images due to the camera being out of focus (Dodge and Karam, 2016) during inference. Such drifts could affect the threshold choice and significantly lower the DNNs performance (see figure 2) prompting the question: How to adjust the threshold of deployed pre-trained models when the latent distribution of target samples differs from the training samples due to variation in distortion levels? Also, this adaptation has to be unsupervised, as the ground truth labels may not be available during inference. This motivates a method that 1) gives initial layers high-level information during training and 2) adjusts early exit thresholds during inference for efficiency and robustness against distortions.

We introduce a new approach that extends the idea of knowledge distillation in early exits (Phuong and Lampert, 2019; Zhu, 2021) to image captioning tasks named Image Captioning with Early Exits and KNowledge Distillation (CapEEN) to improve the efficiency of EEs in image captioning tasks. By distilling the knowledge residing in deeper layers (teacher), CapEEN empowers initial classifiers (students) to utilize the richness of deeper representations, which enhances both the performance and speed of the EE model.

To circumvent the issue of threshold choice under distribution change due to distortion in incoming samples during inference, we propose a novel online learning algorithm A-CapEEN based on the Multi-Armed Bandits (MAB) framework (Auer et al., 2002) to learn the optimal threshold as per latent distributions of input samples. A-CapEEN is activated during the inference phase and adapts to various levels of distortions in test samples with minimal computational requirements, making our method more suitable for real-world scenarios.

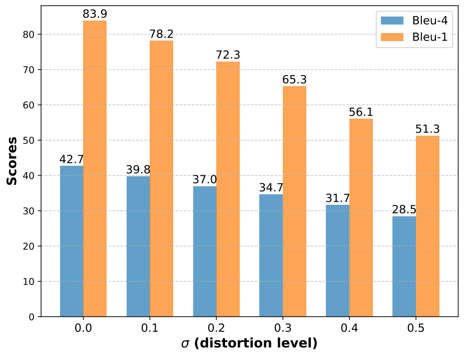


Figure 2: This figure shows the effect of distortion in the performance when the model was trained on undistorted images and tested on images with varying distortion levels (σ models the distortion level).

Our experiments on the MS COCO (Lin et al., 2014) and Flickr30k (Plummer et al., 2015) dataset demonstrate that CapEEN significantly increases speedup while maintaining competitive accuracy performance (1.77x speedup in runtime as compared to the final layer). A-CapEEN further enhances efficiency by dynamically adjusting exit thresholds based on the latent distribution of the test dataset to make it robust to noise present in the datasets during inference.

In summary, our contributions are as follows: 1) We present a novel self-distillation framework named CapEEN tailored for early exiting in image captioning. 2) We introduce an online learning algorithm, A-CapEEN, designed to dynamically choose optimal thresholds as per the latent data distribution utilizing the MAB framework. The algorithm uses the confidence scores to learn the optimal threshold. It makes the method robust to different levels of distortion in the test samples. 3) Comprehensive experiments conducted on the MS-COCO and Flickr30k dataset, reveal the superior performance of CapEEN across all key metrics compared to previous methodologies. Additionally, we validate the effectiveness of A-CapEEN in consistently determining optimal threshold values for various data distributions.

2 Related works

Image Captioning: Recent research has witnessed a surge in exploring efficient text description generation for input images (Bai and An, 2018). Encoder-decoder frameworks have gained prominence for their exceptional performance in text generation tasks, leveraging contextual information (Anderson et al., 2018; Chen et al., 2021; Chen and Lawrence Zitnick, 2015; Fei, 2021; Huang et al., 2019; Vinyals et al., 2015; Xu et al., 2021, 2015; Yao et al., 2018; Zhang et al., 2021; Li et al., 2022, 2023). With the advent of the attention mechanism (Vaswani et al., 2017), the focus has shifted towards employing multiple layers of transformers as both encoders and decoders.

Early-exits. In recent years, the issue of inference latency has gained substantial attention (Matsubara et al., 2022; Guo et al., 2020). To address this issue, DNNs are implemented with internal classifiers in the intermediate layers. Notably, BranchyNet (Teerapittayanon et al., 2016) explores early classification at intermediate layers for images, while SPINN (Laskaridis et al., 2020) uses a mobile cloud setup to split the DNN. SEE (Wang et al., 2019b) performs the early exiting in a service outage scenario. Multiple early exiting frameworks have also been proposed such as (Huang et al., 2017; Yang et al., 2020; Han et al., 2023) for improving early exits for image tasks dynamically choosing the depth of network for different regions for an image. Other works like (Phuong and Lampert, 2019) have utilized knowledge distillation in early exit framework but not for image captioning. Also, it performs joint training of teacher and student that deteriorates the optimality of the backbone.

Numerous early exiting frameworks have been devised for natural language processing tasks (Xin et al., 2020; Liu et al., 2021b, 2020; Wang et al., 2019a; Li et al., 2021; Zhou et al., 2020; Zhu, 2021; Ji et al., 2023; Balagansky and Gavrilov, 2022; Zhang et al., 2022), primarily based on the BERT backbone. DeeCap (Fei et al., 2022) introduces early exiting to image captioning, employing an imitation network to replicate outputs from computationally intensive transformer layers within an encoder-decoder architecture. Similarly, MuE (Tang et al., 2023) applies early exits to OFA (Wang et al., 2022a), a unified vision language model designed for multi-modal applications.

Multi-armed bandits in Early exits. Several works utilize the MAB framework to adapt to different scenarios. Notable methods like LEE (Ju et al., 2021b), DEE (Ju et al., 2021a), and AdaEE (Pacheco et al., 2023) aim to learn the optimal exit points in scenarios like mobile devices with restricted computational resources. Additionally, EPNet (Dai et al., 2020) adopts an offline approach to learning when to exit based on considerations of computational overhead and accuracy. On the other hand, UEEUCB (Hanawal et al., 2022) employs a Multi-Armed Bandit (MAB) framework to dynamically learn the optimal exit strategy in an online and unsupervised manner. UEEUCB relies on the assumption of strong dominance in neural networks, wherein accuracy increases with the layer number in the neural network.

The key differences are: 1) to the best of our knowledge, improving the performance of early exits using knowledge distillation has not been studied for image captioning. 2) Our online algorithm based on the MAB setup overcomes the challenge of choosing the optimal threshold by adapting to the underlying latent distributions of the noise present in the test dataset.

Note: We are the first to apply knowledge distillation for image captioning. Previous methods have applied knowledge distillation for early exits in text and image classification which are much simpler tasks than image captioning and they simultaneously perform distillation in a single-stage training. This cannot be extended to image captioning as it requires high-quality knowledge transfer else it can lead to incorrect learning paths.

Given the complexity of image captioning, we perform a two-stage training that not only transfers high-quality knowledge but also maintains the optimality of the backbone. This approach provides us with a state-of-the-art backbone and serves as a testbed for A-CapEEN.

3 Methodology

In this section, we discuss our method of fusing knowledge distillation with early exit layers by treating the final layer classifier as the teacher and the early exit classifiers as the students.

3.1 Backbone

We use the encoder-decoder framework for building our backbone network motivated by previous works (Liu et al., 2021a; Li et al., 2023). The encoder component comprises a pre-trained Swin-Transformer-base model (Liu et al., 2021c). What sets the Swin-Transformer backbone apart from other vision transformer models (Ranftl et al., 2021) is its Window and Shifted-Window Multi-head Self-Attention (SW-MSA) for the extraction of high-quality rich features. Swin's unique approach has consistently delivered state-of-the-art performance in various vision-related tasks (Wang et al., 2022b). The encoder extracts rich features from the input image and enhances them by capturing their intra-relationships. The output of the Swin-Transformer encoder represents the image in a way that takes into account both local and global context. These features capture details, objects, and their spatial relationships within the image. On the other hand, the decoder component uses the pre-trained GPT-2 (Lagler et al., 2013) model to generate captions in an autoregressive manner, effectively capturing the inter-relationships between words and image features.

3.2 Finetuning Backbone and Training Exits

CAPEEN requires two main training steps: (i) The backbone fine-tuning and (ii) Training of the attached exits using knowledge distillation.

3.2.1 CAPEEN backbone fine-tuning

We start with a pre-trained encoder and decoder. The grid features of the image from the encoder output are passed to the decoder for cross-attention computation. The encoder-decoder backbone is then updated using cross-entropy loss calculated between the predicted token y_i and the ground-truth token y_i^*. The loss function for fine-tuning is formulated as:

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

where θ denotes the collection of all the parameters, I denote the input image, T is the caption length, $y_{1:T}^*$ denotes ground-truth caption, P_N denotes the probability score from the final layer, and N denotes the number of layers in the decoder. We define vocabulary $\mathcal{V}$ as the set of tokens. Once the fine-tuning is complete, we freeze all the backbone parameters. This maintains the optimal quality of the backbone after exits are attached.

3.2.2 CAPEEN Exits Training

After obtaining the fine-tuned backbone from the previous step, we attach task-specific exits at each decoder layer except the final layer. We use a student-teacher setup where the teacher is the final layer, and each intermediate classifier is treated as a student, as visualized in Fig. 3. The weights of the ith exit/student are trained using the loss:

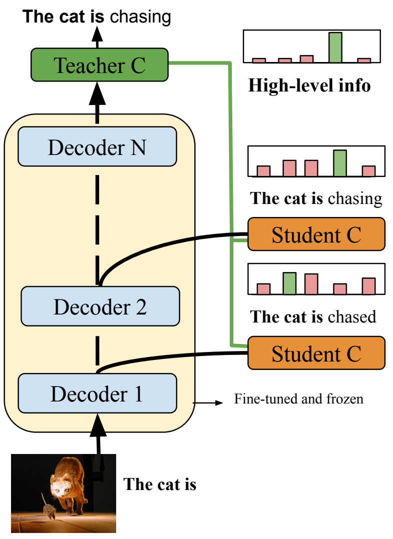


Figure 3: The overall training process for the decoder. Teacher C: Teacher classifier, Student C: Student Classifier, the bars show the probability distribution across different exits.

$$\mathcal{L}_i(I;\theta,\theta_e) = -\frac{1}{T}\sum_{t=1}^{T}(\log(P_i(y_t^*|y_{1:t-1}^*, I;\theta,\theta_e) + KL(p_t^i, p_t^n))$$

where θ_e are the learnable weights for the exits, y_t^* is the tth ground-truth token. p_t^i is the probability vector on the vocabulary for ith student. Its vth component is given by $p_t^i(v) = P_i(v|y_{1:t-1}^*, I;\theta,\theta_e)$ and p_t^n is the probability vector of the teacher model where $p_t^n(v) = P_N(v|y_{1:t-1}^*, I;\theta)$. KL is the KL-Divergence function defined as $KL(p_t^i, p_t^n) = \sum_{v\in\mathcal{V}} p_t^i(v)\log\frac{p_t^i(v)}{p_t^n(v)}$. The early classifiers are then jointly trained using the loss function $\sum_{i=1}^{n-1}\mathcal{L}_i$. In this way, the learning is guided by hard and soft labels from the final layer.

3.3 CAPEEN inference

We predict the caption in an autoregressive manner. This entails making token-by-token predictions for a given image, where the layer at which a token is predicted is determined by the prediction confidence $C_i = \max_{v\in\mathcal{V}} P_i(v|y_{1:t-1}, I;\theta,\theta_e)$. The input to the decoder is processed sequentially through the decoder layers until C_i (the confidence value) is greater than a predefined threshold value α. This threshold is set using the validation data based on the required accuracy-efficiency trade-off. The pseudo-code for inference is given in Algorithm 1. In the algorithm $< bos >$ and $< eos >$ denote the beginning of sentence and end of sentence token, respectively. We rely on fixed confidence threshold values during inference that are tuned during training and applied uniformly across all exits to make decisions of early inference. We denote the prediction word by c and the predicted caption by $\mathbf{C}$.

Algorithm 1 CapEEN Inference

```
1:  Input: Image $I$, Vocabulary $\mathcal{V}$, threshold $\alpha$.
2:  $c = < bos >$, $\mathbf{C} = [< bos >]$
3:  while $c \neq < eos >$ do
4:      for $i \leftarrow 1$ to $N$ do
5:          $C_i \leftarrow \max_{v \in \mathcal{V}} P_i(v|I, \mathbf{C}; \theta, \theta_e)$
6:          if $C_i \geq \alpha$ and $i < N$ then
7:              $c = \arg\max_{v \in \mathcal{V}} P_i(v|I, \mathbf{C}; \theta, \theta_e)$
8:          else if $i = N$ then
9:              $c = \arg\max_{v \in \mathcal{V}} P_N(v|I, \mathbf{C}; \theta)$
10:         end if
11:     end for
12:     $\mathbf{C}.append(c)$
13: end while
14: Return: $\mathbf{C}$
```

4 Learning of Thresholds

Recall the discussion from the introduction that a threshold set using the validation set may not result in better performance when test data distribution drifts from that of the train data distributions, especially when the images in the test data are distorted. As the threshold used in early exit decisions significantly impacts both computational requirements and accuracy, setting it appropriately as per the test data distribution is crucial for optimal performance. We address this challenge by adjusting the threshold as per input data distribution in an online fashion using the MAB framework (Auer et al., 2002).

In the MAB setup, a decision-maker repeatedly selects from a set of arms (or actions) while adapting to the unknown environment. Each arm corresponds to a specific choice or decision. The challenge lies in learning which arms yield the most favourable outcomes (highest reward) over time. This adaptation and learning process, central to MAB setups, aligns with the dynamic nature of online learning problems, where decisions are made sequentially based on incoming data. By leveraging the principles of exploration and exploitation, MAB frameworks facilitate learning the optimal action for the latent distribution.

In this context, we define the action set as k thresholds for each exit as $\mathcal{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$. For exit i, we define the latency factor as the cost of processing the sample from the 1st exit to the ith exit and denote it as o_i. Let $[N]$ denote the set $\{1, 2, \ldots, N\}$. A token will exit the backbone only if the confidence is above the chosen threshold. For a given threshold α where $\alpha \in \mathcal{A}$, suppose that $C_j < \alpha$ for $j \in [i-1]$ and $C_i \geq \alpha$ then it exits at ith exit, and the reward is defined as:

$$r(\alpha) = (C_i - C_1) - \mu o_i \tag{1}$$

where μ is the scaling factor/conversion factor to bring the cost in terms of confidence. If the token does not gain sufficient confidence till the final layer, then $C_j < \alpha$ for all $j \in [N-1]$, and the token will be inferred at the final layer. Then the reward is:

$$r(\alpha) = (C_N - C_1) - \mu o_N \tag{2}$$

The reward could be interpreted as follows: if a sample exits at layer i, then the gain is the confidence gain in the inference at layer i compared to the inference at the first layer, and the cost incurred in processing the token from the first exit to the ith exit. The reward is the net gain, expressed as the difference between gain and cost. The objective is to maximize the expected reward as :

$$\begin{aligned}\mathbb{E}[r(\alpha)] = \sum_{i=1}^{N-1} \mathbb{E}[(C_i - C_1) - \mu o_i|\text{exit}(i)] \\ .P[\text{exit}(i)] + \mathbb{E}[(C_N - C_1) - \mu o_N|\text{exit}(N)] \\ .P[\text{exit}(N)]\end{aligned} \tag{3}$$

where exit(i) denotes the exit from ith layer. The objective is to find an action that maximizes the expected reward function. Note that the reward does not use any label information. The optimal arm is defined as $\alpha^* = \arg\max_{\alpha \in \mathcal{A}} r(\alpha)$. If we consider a policy π that selects threshold $\alpha_t \in \mathcal{A}$ in round t based on past observations. The efficiency of the chosen policy can be expressed in terms of cumulative regret, defined as:

$$R(\pi, T) = \sum_{t=1}^{T} \mathbb{E}[r(\alpha^*) - r(\alpha_t)] \tag{4}$$

4.1 Algorithm

Algorithm 2 A-CapEEN

```
1: Input: $o_i \; \forall i \in [N], \gamma \geq 1, \mathcal{A}$
2: Initialize: For an image obtain $|\mathcal{A}|$ tokens by setting different $\alpha \in \mathcal{A}$ and observe $r(\alpha)$.
3: Set $Q(\alpha) \leftarrow r(\alpha), N(\alpha) \leftarrow 1, \forall \alpha$.
4: $t = |\mathcal{A}| + 1$
5: for $j > 1$ do
6:     For image $I_j$, set $c = \mathbf{C}_j =< bos >$
7:     while $c \neq < eos >$ do
8:         $S_j = \{\mathbf{C}_j, I_j\}$
9:         For next token, set threshold
10:        $\beta_t \leftarrow \arg\max_{\alpha \in \mathcal{A}} \left( Q(\alpha) + \gamma \sqrt{\frac{\ln(t)}{N(\alpha)}} \right)$
11:        $i = 1$ and $o_1 = 0$
12:        for $i = 1$ to $N$ do
13:            Pass $S_j$ till layer $i$
14:            Apply threshold $\beta_t$ and observe $C_i$
15:            if $C_i \geq \beta_t$ and $i < N$ then
16:                Infer at layer $i$ and exit
17:                $r_t(\alpha) \leftarrow (C_i - C_1) - \mu o_i$
18:                $N_t(\alpha) \leftarrow N_{t-1}(\alpha) + 1$
19:                $Q_t(\alpha) \leftarrow \frac{\sum_{j=1}^{t} r_j(\alpha_j) \mathbb{1}_{\{\alpha_j = \alpha\}}}{N_t(\alpha)}$
20:                $c = \arg\max_{v \in \mathcal{V}} P_i(v|S_j; \theta, \theta_e)$
21:                break
22:            else if $i = N$ then
23:                Process till the last layer.
24:                $r_t(\alpha) \leftarrow (C_N - C_1) - \mu o_N$
25:                $N_t(\alpha) \leftarrow N_{t-1}(\alpha) + 1$
26:                $Q_t(\alpha) \leftarrow \frac{\sum_{j=1}^{t} r_j(\alpha_j) \mathbb{1}_{\{\alpha_j = \alpha\}}}{N_t(\alpha)}$
27:                $c = \arg\max_{v \in \mathcal{V}} P_N(v|S_j; \theta)$
28:            end if
29:            $t \leftarrow t + 1$
30:        end for
31:        $\mathbf{C}_j.append(c)$
32:    end while
33:    Return $\mathbf{C}_j$, $j \leftarrow j + 1$
34: end for
```

We introduce an algorithm called Adaptive-CapEEN (A-CapEEN) to adaptively choose the threshold values, and its pseudocode is outlined in Algorithm 2. The input for this algorithm includes latency factors o_i for each exit $i \in [N]$, the exploration parameter γ. The expectation above is with respect to the randomness induced by latent sample distribution in the selection of actions. A policy denoted as π^* is characterized as sub-linear when the average cumulative regret diminishes, that is, $R(\pi^*, T)/T \rightarrow 0$. Our primary goal is to devise a learning algorithm that has a sub-linear regret guarantee.

The initialization of the algorithm involves ensuring that each action is played at least once. In the subsequent rounds, the algorithm selects the arm with the highest UCB index, denoted as β_t (line 5). UCB indices comprise weighted averages of rewards $Q_t(\alpha)$ and incorporate confidence bonuses with γ as the exploration parameter (weight). This confidence threshold is associated with each exit i.e., β_t is the threshold chosen for all exits for the tth token. The token is then processed until the confidence is above β_t (line 10) and if the token never gains sufficient confidence, it is inferred at the final layer (line 16). After the token exits from one of the layers, the average reward of the played arm is updated (line 14). The caption starts with a $< bos >$ token and the next predicted token c is appended to the predicted caption set $\mathbf{C}$ until the $< eos >$ token is predicted. Note that there are two counters, one is for the image, once the $< eos >$ token is predicted the caption for the image is complete and the counter for the image is updated while the token counter is updated every time a token is passed through it and denotes the number of times rewards are updated. Hence, the algorithm learns faster as the thresholds (arms) are learnt on the number of tokens passed instead of the number of images.

Drawing from the analysis of UCB1 (Auer et al., 2002), the regret of A-CapEEN can be shown to be of the order $\mathcal{O}\left(\sum_{\alpha \in \mathcal{A} \setminus \alpha^*} \frac{log(n)}{\Delta\alpha}\right)$, where $\Delta_\alpha = r(\alpha^*) - r(\alpha)$ denotes the sub-optimality gap. For completeness, we provide proof in the Appendix A.1.

5 Experiments

Dataset and Metric: We evaluate the performance of our method using the MS-COCO and Flickr30k datasets for image captioning. Our primary objective is to produce coherent image captions. To maintain consistency with prior studies, we preprocess all captions by converting them to lowercase. Additionally, we filter out words that occur fewer than 5 times in the dataset, ensuring robustness in our evaluation. We report key metrics, including BLEU-4 (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), CIDEr (Vedantam et al., 2015), ROUGE (Lin, 2004) and SPICE (Anderson et al., 2016) scores. To be

Models/Metric	BLEU-1	BLEU-4	METEOR	CIDEr	SPICE	ROUGE-L	Speedup
Final-Exit	82.9	42.3	32.2	147.1	26.7	61.3	1.00×
Decoder-9L	76.5	37.1	29.3	134.8	23.2	57.9	1.33×
DeeBERT	70.1	32.3	26.9	110.2	20.9	50.7	1.35×
ElasticBERT	71.4	32.8	27.6	114.6	21.4	51.6	1.37×
F-PABEE	72.7	33.9	27.9	115.6	21.9	52.3	1.30×
FastBERT	75.0	35.6	28.2	119.5	22.1	53.7	1.42×
LeeBERT	77.3	38.7	29.4	129.2	23.0	55.9	1.39×
DeeCap	77.5	39.2	29.9	132.8	23.2	56.9	1.60×
MuE	79.3	40.5	30.9	139.4	24.9	59.7	1.66×
Ours	**80.7**	**41.3**	**31.6**	**140.3**	**25.5**	**60.1**	**1.77**×

Table 1: Main Results on COCO dataset: CapEEN outperforms other baselines across different metrics.

consistent with previous methods, we report the speedup ratio as the measure of reduction in computational requirements. This metric can easily be converted to the expected time reduction rate.

$$\frac{\sum_{l=1}^{N} w_l^I \times N}{\sum_{l=1}^{N} w_l^I \times l} \tag{5}$$

where N is the number of decoder layers and l is the number of layers after which the token exits the backbone during inference. w_l^I is the number of words that exit at the lth decoder layer for an image I. This metric provides insights into our decoding process's resource utilization and efficiency, a critical aspect discussed in our work.

Training: The encoder-decoder backbone is initially fine-tuned for 10 epochs with a starting learning rate of 1e-5, which decays by 0.5 every 3 epochs. Subsequently, self-critical training is employed for 5 epochs with an initial learning rate of 7e-6, also decaying by 0.5 every 3 epochs. The backbone weights are frozen post-fine-tuning, and exits are added to the decoder, whose weights are further trained for 5 epochs. The Adam optimizer and a batch size of 8 are chosen, with a threshold of 0.6 chosen based on the accuracy-efficiency trade-off on the validation split. Inference is conducted on the Karpathy test split of the MS-COCO dataset with a batch size of 1, using NVIDIA RTX 2070 GPUs. Results are presented in Table 1. More runtime details are in Appendix B.5

Adaptive threshold learning (A-CapEEN): We experiment with two types of noise, Gaussian noise and Gaussian blur. We mimic real-world scenarios by adding different levels of noise to the images of the test set and then proceed to learn the thresholds. In this, we adapt the thresholds based on the level of distortion present in an image using Algorithm 2. For this step, we choose the action set as $\mathcal{A} = \{0.1, 0.2, \ldots, 1.0\}$. We add noise to the dataset's test split. Then we perform learning of threshold values, using A-CapEEN for all the exits. Note that algorithm 2 has small computational complexity and does not add upon latency in the inference process. It maximizes the rewards over a finite set which has negligible complexity.

We set $\mu = 1/N$. The latency cost o_i could be understood as the cost of processing the samples from 1st exit to ith exit. Hence we set the latency cost as $o_i = \lambda i$ where λ is per layer computational cost. Since the value of threshold α is adaptively chosen in this case, λ models the trade-off between accuracy and efficiency. A higher value of λ will provide higher accuracy while a lower value will give high efficiency. We use the value of $\lambda = 1$. A detailed ablation study of this hyperparameter can be found in the Appendix B.4.

Impact of change in Distribution: We evaluate the impact of data distribution shifts on our model's performance (BLEU-1, BLEU-4) using image distortion (Figure 2). We train on undistorted MS-COCO images and introduce Gaussian noise/blur to the test set, simulating real-world imperfections. Noise levels are varied for Gaussian noise ($\sigma \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$) and Gaussian blur ($\sigma \in \{0.5, 1.0, 1.5, 2.0\}$) to assess robustness to context-driven data variations.

Baselines: We establish baseline models for performance evaluation. To assess improvements in speedup as compared to the final decoding layer, we consider this setup as a baseline for our approach (final exit). In this case, all the samples are inferred only at the final layer. We also directly reduce the layer number to 9 in Decoder-9L and use only 9th layer to make an inference. This baseline serves as a lower bound for performance metrics since it does not employ any technique. Since DeeBERT (Xin et al., 2020), ElasticBERT (Liu et al., 2021b), F-PABEE (Zhou et al., 2020), FastBERT (Liu et al., 2020) and LeeBERT (Zhu, 2021) were originally conducted on BERT, we imple-

Model	BLEU-4	METEOR	CIDEr	Speed
Blur intensity = 0.5				
Final	39.2	31.6	148.7	1.00×
DeeCAP	34.7	28.4	133.6	1.33×
MuE	35.5	28.9	137.2	1.47×
Our	36.8	29.5	139.9	1.55×
A-our	**37.3**	**29.8**	**141.8**	**1.61**×
Blur intensity = 1.0				
Final	37.9	30.7	139.5	1.00×
DeeCAP	33.2	27.5	126.3	1.31×
MuE	34.5	28.3	129.4	1.48×
Our	36.1	28.6	130.9	1.53×
A-our	**37.0**	**29.2**	**132.7**	**1.59**×
Blur intensity = 1.5				
Final	34.1	28.6	127.4	1.00×
DeeCAP	28.4	26.8	113.5	1.28×
MuE	29.7	27.4	117.2	1.46×
Our	30.9	27.9	120.6	1.50×
A-Our	**31.7**	**28.5**	**123.0**	**1.63**×
Blur intensity = 2.0				
Final	28.7	22.6	100.4	1.00×
DeeCAP	21.2	18.4	91.7	1.19×
MuE	23.9	20.3	94.5	1.35×
Our	25.4	21.0	98.2	1.39×
A-Our	**26.5**	**21.9**	**101.2**	**1.49**×

Table 2: Results of A-CapEEN when the test sample contains images with different levels of blur.

mented their methods in the decoder part of GPT-2 without changing any hyperparameters. We compare our model with DeeCap (Fei et al., 2022) and MuE (Tang et al., 2023) utilizing the setup from their framework. DeeCap implements an imitation network where as a sample exits the backbone, it passes through multiple MLP layers to regain lost information due to early exiting. MuE is another early exiting model for image captioning that attaches exits in the OFA backbone which is a multimodal unified vision language model. It has the similarity score of consecutive layers as the confidence metric. More details on baselines can be found in Appendix B.3. Note that except for DeeCAP, we have applied the methodology of other baselines to our setup.

We evaluate the performance of CapEEN on the Karpathy test split of the MS-COCO dataset, as done by all the baselines for fair evaluation, and the results are in Table 1. The results of the Flickr30k dataset are in Appendix C and table 5.

5.1 Results

In this section, We discuss the main results (median of 5 independent runs) of our work. Details of the stability of our method are in table 3.

CapEEN. In the context of pristine (undistorted) images, Table 1 presents performance results of early exit models, showcasing our approach's superiority over various baselines. Our method outperforms all previous baselines, including DeeCap and MuE. Unlike DeeCap, our approach does not rely on an imitation network, avoiding noise accumulation as samples exit the main backbone early. Additionally, our method consistently outperforms MuE by leveraging deep representations crucial for semantic correctness. Baselines like DeeBERT, ElasticBERT, and F-PABEE exhibit significant performance drops due to limited access to deep representations. While FastBERT and LeeBERT performs better than these baselines by accessing deep representations, they lacks the appropriate ground-truth information during weight learning for attached exits.

CapEEN enriches early classifiers with higher-level semantic information distilled from the final layer, leading to minimal performance drops and the highest speedup ratio across all metrics.

Adaptive learning of the thresholds (A-CapEEN). In Table 2 and 4, we highlight the impact of adapting threshold values based on changes in data distribution, comparing them with thresholds learned during training and fixed during inference. Our findings demonstrate that fixed thresholds significantly affect inference time and performance, highlighting the need for dynamic threshold learning to accommodate contextual information and inherent image noise.

For the CIDEr metric, A-CapEEN observes minimal occasional gains from the final decoder layer, attributed to overthinking (Zhu, 2021) during inference similar to overfitting during training. A-CapEEN consistently outperforms CapEEN as well as other baselines when the dataset distribution changes due to distortion in images. The gain in performance and speed is observed as A-CapEEN finds the optimal threshold that optimally models the accuracy-efficiency trade-off.

We perform an ablation study and a case study in Appendix B to further prove the effectiveness of our method.

5.2 Analysis of threshold α

We present in the figure 4, the trade-off between accuracy and efficiency. To obtain higher accuracy, we increase the value of α and the time reduction decreases with higher accuracy. On the other hand, decreasing the threshold α will in turn increase the time reduction but with compromising performance. CapEEN performed better than other baselines due to the available knowledge

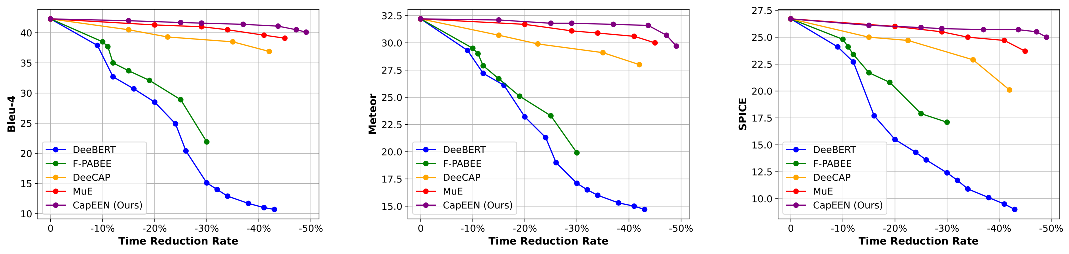


Figure 4: Change in the performance of different metrics with changing time reduction rates. These reductions are observed by changing the threshold parameter α.

from deeper exits. Observe that in figure 4, there is a very minimal drop in performance (mostly constant). This ability comes from the extra information available at the intermediate exits due to knowledge distillation. However, the performance begins to drop when we try to reduce time by more than 50% since then the sample exits from the very initial layers which in turn affects the performance.

6 Conclusion

We introduced a new encoder-decoder backbone for image captioning with early exits and knowledge distillation named CapEEN. Using the student-teacher model, we trained early exit classifiers using knowledge distillation to capture high-level semantic representations available at the deeper layers. We demonstrated that CapEEN offers a significant increase in speedup while maintaining a competitive performance guarantee. Further, we introduced A-CapEEN, where the threshold used for early exit decisions can be adaptively learned for distributions that differ from the training data due to changes in the distortion levels.

7 Limitations

In our work, we have applied a uniform threshold across all exits during inference, simplifying the implementation process. However, extending this to individual thresholds for each exit could offer additional flexibility, albeit with increased complexity. Additionally, while early exit models effectively reduce latency during inference, they do incur higher computational costs during training. This is due to the need for exit classifiers to learn additional weights after each layer, resulting in increased complexity. Nonetheless, post-training, these models significantly enhance inference speed, which becomes the primary consideration post-deployment.

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.

References

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.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6077–6086.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47:235–256.

Shuang Bai and Shan An. 2018. A survey on automatic image caption generation. *Neurocomputing*, 311:291–304.

Nikita Balagansky and Daniil Gavrilov. 2022. Palbert: Teaching albert to ponder. *Advances in Neural Information Processing Systems*, 35:14002–14012.

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.

Long Chen, Zhihong Jiang, Jun Xiao, and Wei Liu. 2021. Human-like controllable image captioning with verb-specific semantic roles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16846–16856.

Xinlei Chen and C Lawrence Zitnick. 2015. Mind's eye: A recurrent visual representation for image caption generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2422–2431.

Marcella Cornia, Matteo Stefanini, Lorenzo Baraldi, and Rita Cucchiara. 2020. Meshed-memory transformer for image captioning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10578–10587.

Xin Dai, Xiangnan Kong, and Tian Guo. 2020. Epnet: Learning to exit with flexible multi-branch network. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 235–244.

Samuel Dodge and Lina Karam. 2016. Understanding how image quality affects deep neural networks. In *2016 eighth international conference on quality of multimedia experience (QoMEX)*, pages 1–6. IEEE.

Zhengcong Fei. 2021. Memory-augmented image captioning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1317–1324.

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.

Longteng Guo, Jing Liu, Xinxin Zhu, Xingjian He, Jie Jiang, and Hanqing Lu. 2020. Non-autoregressive image captioning with counterfactuals-critical multi-agent learning. *arXiv preprint arXiv:2005.04690*.

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*.

Manjesh K Hanawal, Avinash Bhardwaj, et al. 2022. Unsupervised early exit in dnns with multiple exits. *arXiv preprint arXiv:2209.09480*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

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*.

Lun Huang, Wenmin Wang, Jie Chen, and Xiao-Yong Wei. 2019. Attention on attention for image captioning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4634–4643.

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.

Weiyu Ju, Wei Bao, Liming Ge, and Dong Yuan. 2021a. Dynamic early exit scheduling for deep neural network inference through contextual bandits. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 823–832.

Weiyu Ju, Wei Bao, Dong Yuan, Liming Ge, and Bing Bing Zhou. 2021b. 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.

Klemens Lagler, Michael Schindelegger, Johannes Böhm, Hana Krásná, and Tobias Nilsson. 2013. Gpt2: Empirical slant delay model for radio space geodetic techniques. *Geophysical research letters*, 40(6):1069–1073.

Stefanos Laskaridis, Stylianos I Venieris, Mario Almeida, Ilias Leontiadis, and Nicholas D Lane. 2020. Spinn: synergistic progressive inference of neural networks over device and cloud. In *Proceedings of the 26th annual international conference on mobile computing and networking*, pages 1–15.

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. *arXiv preprint arXiv:2301.12597*.

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

Xiaonan Li, Yunfan Shao, Tianxiang Sun, Hang Yan, Xipeng Qiu, and Xuanjing Huang. 2021. Accelerating bert inference for sequence labeling via early-exit. *arXiv preprint arXiv:2105.13878*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

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.

Wei Liu, Sihan Chen, Longteng Guo, Xinxin Zhu, and Jing Liu. 2021a. Cptr: Full transformer network for image captioning. *arXiv preprint arXiv:2101.10804*.

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, Jiawen Wu, Lingling Wu, Xinyu Zhang, Hao Jiang, Zhao Cao, Xuanjing Huang, and Xipeng Qiu. 2021b. Towards efficient nlp: A standard evaluation and a strong baseline. *arXiv preprint arXiv:2110.07038*.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021c. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022.

Yoshitomo Matsubara, Marco Levorato, and Francesco Restuccia. 2022. Split computing and early exiting for deep learning applications: Survey and research challenges. *ACM Computing Surveys*, 55(5):1–30.

Roberto G Pacheco, Mark Shifrin, Rodrigo S Couto, Daniel S Menasché, Manjesh K Hanawal, and Miguel Elias M Campista. 2023. Adaee: Adaptive early-exit dnn inference through multi-armed bandits. In *ICC 2023-IEEE International Conference on Communications*, pages 3726–3731. IEEE.

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.

Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *Proceedings of the IEEE international conference on computer vision*, pages 2641–2649.

René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. 2021. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12179–12188.

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.

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.

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.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164.

Meiqi Wang, Jianqiao Mo, Jun Lin, Zhongfeng Wang, and Li Du. 2019a. 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.

Yiyu Wang, Jungang Xu, and Yingfei Sun. 2022b. End-to-end transformer based model for image captioning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 2585–2594.

Zizhao Wang, Wei Bao, Dong Yuan, Liming Ge, Nguyen H Tran, and Albert Y Zomaya. 2019b. See: Scheduling early exit for mobile dnn inference during service outage. In *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 279–288.

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*.

Guanghui Xu, Shuaicheng Niu, Mingkui Tan, Yucheng Luo, Qing Du, and Qi Wu. 2021. Towards accurate text-based image captioning with content diversity exploration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12637–12646.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell:

Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR.

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.

Ting Yao, Yingwei Pan, Yehao Li, and Tao Mei. 2018. Exploring visual relationship for image captioning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 684–699.

Xuying Zhang, Xiaoshuai Sun, Yunpeng Luo, Jiayi Ji, Yiyi Zhou, Yongjian Wu, Feiyue Huang, and Rongrong Ji. 2021. Rstnet: Captioning with adaptive attention on visual and non-visual words. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15465–15474.

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.

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.

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 Regret Bound

Theorem A.1. *For any $\gamma \geq 1$, the regret of A-CapEEN with K arms in the action set after T rounds is given as:*

$$R(A - \text{CapEEN}, T) \leq 4\gamma \sum_{\alpha \neq \alpha^*} \frac{log(T)}{\Delta_\alpha} + \left(\frac{\pi^2}{3} + 1\right) \sum_{\alpha \neq \alpha^*} \Delta_\alpha \quad (6)$$

where $\Delta_\alpha = r(\alpha^*) - r(\alpha)$.

Proof. The proof follows similar lines as given in the classical UCB (Auer et al., 2002). the instantaneous regret in round t is given as :

$$R_t = r(\alpha_t) - r(\alpha^*)$$

The value of $r(\alpha)$ defined in our problem is a bounded quantity. More specifically $r(\alpha) \in [-1-\lambda N, 1]$ where λ is the processing cost of the layers and N is the number of layers in the decoder.

The bound given in algorithm A.1 can be further improved by adding multiple updates for a single sample. To get an idea of the number of arms that will get updated, we provide the following proposition:

B More experimental details

B.1 Ablation study

In Figure 6a, we conduct an ablation study by learning the early exit weights using different loss combinations. First, we learn the early exit only using the cross-entropy loss where only ground-truth labels are used to train the weights of early exits. Then we consider only knowledge distillation loss and only soft labels guide the early exits. We observe that if we do not use the combination of both soft as well as hard labels then early classifiers get the highest hit in terms of performance as they need rich information from deeper layers. As we move deeper into the backbone all three combinations converge to similar CIDEr scores as deeper layers already have access to rich features.

These observations suggest the significance of the various components within the loss function, ultimately guiding us towards a more comprehensive understanding of the model's behaviour.

B.2 Case Study

We also provide some examples of how the different proposed models annotate an image. We compare the captions generated by the two proposed methods with the captions provided by humans (ground-truth). As we can observe from figure 6c, CapEEN performs well in explaining the content of the image. As compared to the ground-truth, CapEEN provides more details as it also recognizes that the cake is large. A-CapEEN gets a caption which is closer to the CapEEN when there is no noise in the image. However, as we add noise to the image, then the results of A-CapEEN are better than CapEEN. Note that A-CapEEN is tested on this example after it has seen a sufficient amount (around 100 images) of data with similar noise so that it adopts the distribution of the data. Also, observe that there is a slight variation in the captions and some information is lost when there is added noise in the dataset. Still, the prediction

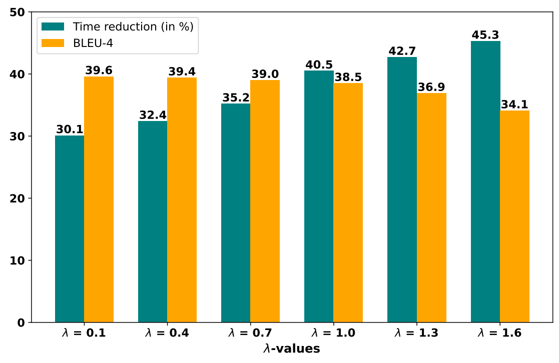


Figure 5: The change in time reduction rate as well as the BLEU-4 scores when the values of the λ are varied.

given by A-CAPEEN is better as it has added 'is cutting' in the sentence while CAPEEN just outputs 'cuts' and CAPEEN observes that it is a large knife which is redundant information.

.

B.3 Baselines

DeeBERT: do a separate training of the backbone and the attached exits in a setup similar to ours. It trains the model by only using the ground-truth labels and is originally trained for text classification tasks. We implement it in the decoder part of our backbone.

ElasticBERT: is similar to DeeBERT but performs a joint training of the backbone i.e. all the exits with the final exit are trained simultaneously. We implement it in the decoder part and provide the results.

F-PABEE: has different exiting criteria from DeeBERT and ElasticBERT, it exits the samples based on the consistency in the prediction from the intermediate classifiers being above a threshold. Its main purpose was to reduce the time as well as overthinking problems in DNNs.

FastBERT: has also been originally implemented on the BERT backbone. For the first few epochs of training, it performs a joint training of the backbone and the exits and then more epochs on distilling the knowledge from deeper exits but without using the ground-truth. We implement it in the GPT-2 i.e. decoder part of our method.

LeeBERT use prediction stability to decide early exits, LeeBERT also distil the knowledge from deeper layers. It perfoms cross level optimization to make the exits and the backbone learn better.

DeeCAP: is specifically made for early exits in image captioning. It learns an imitation network in the form of MLP(Multi-layer Perceptron) layers for each sample as it exits the backbone. Due to a separate MLP layer for each exit, it increase the size of the model to a greater extent. The expected time reduction rate is also affected as each time the sample has to check whether to exit a layer or not from the original backbone it has to exit from the main backbone and pass through the imitation network until the confidence is above a given threshold.

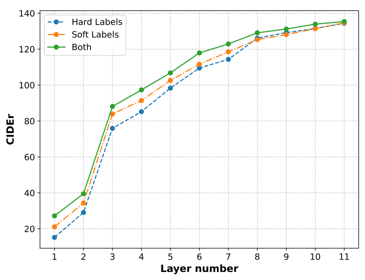
MuE: is an early exit mechanism in DNNs designed for making inferences at the intermediate classifier for the multi-modal tasks. The base model it has used is the OFA model. It sets the similarity in the hidden representations as the exiting criteria. This exiting criterion gives an advantage that it could also be applied to the encoder but has a restriction that the layers of the encoders are identical which might not be always the case (Liu et al., 2021c; He et al., 2016).

B.4 Analysis of the cost λ

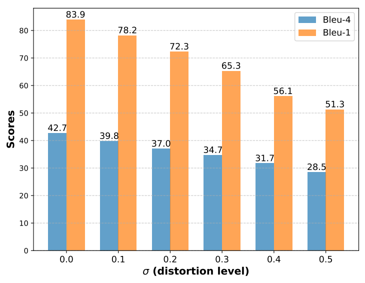
In figure 5, we perform analysis on the cost parameter λ. We introduced the cost parameter λ as a user-defined parameter. It can also be understood as a trade-off factor between accuracy and efficiency. Since a higher value of λ will increase the impact of processing cost and in turn will force samples to make an early exit by lowering the threshold values. This will increase the efficiency of the model but in turn, will decrease the accuracy of predictions. While a smaller value of λ will motivate the samples to gain higher confidence by processing them to deeper layers. The latter method will increase the accuracy while compromising the efficiency of the model.

B.5 Computational requirements

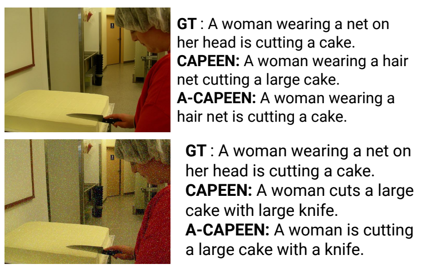
We perform experiments on three NVIDIA RTX 2070 GPUs and the training time takes $\sim$ 10 hours for the backbone fine-tuning and an additional 1 hour for exits training. The model has 205 million parameters and not all are updated as the backbone is pre-trained and not all parameters need to be updated as we are only fine-tuning it. The inference time is $<$ 1 hour on the Karpathy test split. When we also employ A-CapEEN i.e. we adapt the threshold values then the time required is the same as running CapEEN individually since the MAB framework does not add minimal computational complexity.



(a) Effect of Loss Combinations on CIDEr Score across the layers.



(b) Effect of different distortion levels on Bleu-1 and Bleu-4 metrics on final layer.



GT : A woman wearing a net on her head is cutting a cake.
CAPEEN: A woman wearing a hair net cutting a large cake.
A-CAPEEN: A woman wearing a hair net is cutting a cake.

GT : A woman wearing a net on her head is cutting a cake.
CAPEEN: A woman cuts a large cake with large knife.
A-CAPEEN: A woman is cutting a large cake with a knife.

(c) Predicted captions by CAPEEN and A-CAPEEN Ground-truth(GT) caption.

Figure 6: (a) Hard Labels (Ground-truth), Soft Labels (Teacher Model Predictions), and Both (Combined Labels). (b) Image is added with noise and then the captions are predicted, we report the Bleu-1 and Bleu-4 scores. (c) The first image is pristine (undistorted) while the second image is distorted with $\sigma = 0.2$ and the captions are predicted using CAPEENand A-CAPEEN.

Model	BLEU-4	CIDEr	SPICE	ROUGE-L	Speedup
Final-exit	42.7	154.1	26.7	63.3	1.00x
CapEEN	41.3±0.015	140.3±0.3	25.5±0.009	62.3±0.07	1.77x±0.058
A-CapEEN	42.1±0.08	144.2±1.1	26.1±0.08	63.0±0.5	1.76x±0.17

Table 3: This table shows the stability of CAPEEN and A-CAPEEN where 5 independent runs are made.

Model	**BLEU-4**	**METEOR**	**CIDEr**	**Speed**
		σ = **0.5**		
Final-exit	28.5	24.0	99.5	1.00x
Ours	26.1	23.9	91.1	1.43x
A-Ours	**26.7**	**24.1**	**91.9**	**1.51x**
		σ = **0.4**		
Final-exit	31.7	26.5	113.9	1.00x
Ours	28.9	26.4	105.2	1.53x
A-Ours	**29.2**	**26.4**	**107.9**	**1.59x**
		σ = **0.3**		
Final-exit	34.7	28.9	122.1	1.00x
Ours	32.7	28.3	115.8	1.50x
A-Ours	**33.5**	**28.4**	**121.7**	**1.58x**
		σ = **0.2**		
Final-exit	37.1	30.7	134.8	1.00x
Ours	35.7	30.1	127.3	1.51x
A-Ours	**35.8**	**30.1**	**129.4**	**1.58x**
		σ = **0.1**		
Final-exit	39.8	31.9	142.8	1.00x
Ours	38.5	31.1	132.3	1.62x
A-Ours	**38.8**	**31.3**	**135.1**	**1.68x**
		σ = **0.0**		
Final-exit	42.3	32.2	147.1	1.00x
Ours	41.3	31.6	140.3	**1.77x**
A-Ours	**41.8**	**32.0**	**142.6**	1.76x

Table 4: Comparison of CAPEEN and A-CAPEEN with different distortion levels.

C Results on Flickr30k

In table 5, we provide the results on Flickr30k when the experiments are performed in a similar setup as given in the experimental section 5. Our method consistently outperforms the other baselines in terms of all the metrics. Again this gain in performance with decreasing sufficient time is possible due to the higher level of information available at intermediate exit points. This encourages the exit of more samples early and with high confidence. Hence we prove across different datasets that CAPEEN outperforms previous baselines.

Model/Metric	**BLEU-1**	**BLEU-4**	**METEOR**	**CIDEr**	**SPICE**	**ROUGE-L**	**Speed**
Final-Exit	76.9	33.6	25.5	72.7	18.1	55.3	1.00×
DeeBERT	65.2	24.5	20.8	47.5	12.6	45.1	1.24×
ElasticBERT	66.4	25.3	21.1	49.3	13.2	45.9	1.27×
F-PABEE	68.6	26.5	21.9	51.9	13.8	46.3	1.29×
FastBERT	70.1	27.9	22.8	55.3	14.7	48.5	1.33×
DeeCap	72.8	30.1	23.5	64.2	16.1	51.7	1.41×
MuE	74.2	31.9	24.4	66.7	16.9	53.4	1.58×
Ours	**75.1**	**32.8**	**25.0**	**67.2**	**17.4**	**53.9**	**1.65**×

Table 5: Results on Flickr30k dataset