MobiVQA: Efficient On-Device Visual Question Answering

QINGQING CAO, Stony Brook University, USA
PRERNA KHANNA, Stony Brook University, USA
NICHOLAS D. LANE, University of Cambridge & Samsung AI, United Kingdom
ARUNA BALASUBRAMANIAN, Stony Brook University, USA

Visual Question Answering (VQA) is a relatively new task where a user can ask a natural question about an image and obtain an answer. VQA is useful for many applications and is widely popular for users with visual impairments. Our goal is to design a VQA application that works efficiently on mobile devices without requiring cloud support. Such a system will allow users to ask visual questions privately, without having to send their questions to the cloud, while also reduce cloud communication costs. However, existing VQA applications use deep learning models that significantly improve accuracy, but is computationally heavy. Unfortunately, existing techniques that optimize deep learning for mobile devices cannot be applied for VQA because the VQA task is multi-modal—it requires both processing vision and text data. Existing mobile optimizations that work for vision-only or text-only neural networks cannot be applied here because of the dependencies between the two modes. Instead, we design MobiVQA, a set of optimizations that leverage the multi-modal nature of VQA. We show using extensive evaluation on two VQA testbeds and two mobile platforms, that MobiVQA significantly improves latency and energy with minimal accuracy loss compared to state-of-the-art VQA models. For instance, MobiVQA can answer a visual question in 163 milliseconds on the phone, compared to over 20-second latency incurred by the most accurate state-of-the-art model, while incurring less than 1 point reduction in accuracy.

CCS Concepts: • **Human-centered computing** → **Ubiquitous and mobile computing systems and tools**; *Accessibility systems and tools*; **Ubiquitous and mobile computing systems and tools**; *Accessibility systems and tools*; • **Computing methodologies** → *Computer vision*; *Natural language processing*; *Computer vision.*

Additional Key Words and Phrases: mobile computing, visual question answering, edge computing, on-device applications

ACM Reference Format:
Qingqing Cao, Prerna Khanna, Nicholas D. Lane, and Aruna Balasubramanian. 2022. MobiVQA: Efficient On-Device Visual Question Answering. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6, 2, Article 44 (June 2022), 23 pages. https://doi.org/10.1145/3534619

1 INTRODUCTION

Visual Question Answering or VQA is a task of answering a natural language question that a user can ask about any image. VQA has been widely used for several tasks including grocery shopping, locating a specific object in a complex scene, and choosing clothes [8]. VQA is especially useful for users with visual impairments who seek descriptions or ask questions about the physical world around them [9]. VQA empowers over 253 million people worldwide who have vision impairments [5]. With the recent progress in deep learning, VQA models see

Authors' addresses: Qingqing Cao, qicao@cs.stonybrook.edu, Stony Brook University, Stony Brook, NY, USA; Prerna Khanna, pkhanna@cs.stonybrook.edu, Stony Brook University, Stony Brook, NY, USA; Nicholas D. Lane, ndl32@cam.ac.uk, University of Cambridge & Samsung AI, Cambridge, United Kingdom; Aruna Balasubramanian, arunab@cs.stonybrook.edu, Stony Brook University, Stony Brook, NY, USA.

2474-9567/2022/6-ART44 $15.00
https://doi.org/10.1145/3534619

significantly improved accuracy, making them even more useful. However, state-of-the-art VQA models run on a cloud server, often on GPUs, and are resource intensive.

We ask the question: can VQA models run locally on the phone, without requiring any cloud support. The primary motivation for our work is to allow users to ask visual questions about their surroundings without having to give out their private information. For example, a blind user may want to take a picture of their prescription bottle and ask about the type of pill in the bottle. They may not necessarily want to send this information to a cloud provider. A further motivation is the cloud cost. A user may ask questions about an image they captured on their phone, but sending the image to the cloud over a cellular connection will incur large costs. Instead, running a VQA model locally will mean that a user can receive answers to their questions privately and with low cost.

The problem of course is that state-of-the-art VQA models incur large latencies when run on mobile devices. We analyzed three state-of-the-art VQA models—LXMERT [52], X-LXMERT [16], and ViLT [34], released in years 2019, 2020, and 2021, respectively. Running the most accurate models on a smartphone takes over 20 seconds; the less accurate models take an average of 1.4 seconds. For a user to use these models seamlessly for their everyday activities, the latencies need to be in the range of 200 to 300 milliseconds [3, 47]. Therefore, the VQA deep learning model needs to be optimized for resource-constrained environments.

Optimizing deep learning models for mobile devices is a well-studied problem [11, 12, 17, 36]. However, VQA presents unique challenges because of its multi-modal nature. Rather than vision-only tasks, or image-only tasks, VQA models have both a vision processing component and an image processing component. Existing VQA models work by creating a representation of the question using NLP techniques and a representation of the image using vision techniques. These two representations are combined using cross-modal encoding, which is the cause of most of the bottleneck. However, popular vision optimizations such as parallelization [7] does not work because of the dependencies between text and image representation in the cross-modal layer. Optimizations designed to make NLP applications work on mobile devices [13] reduce the total amount of text processing. However, VQA models do not have to process large amounts of text data unlike text-based question answering.

Instead, we design MobiVQA, a set of optimizations that (i) specifically leverage the multimodal nature of the VQA task, and (ii) can be applied to several existing VQA models. The latter is important because VQA models are moving targets[1]. The MobiVQA optimizations work by reducing the model computation by leveraging information from both the image and the question.

The MobiVQA optimizations are based on two intuitions. The first is that visual questions are not equally hard or easy, some questions cannot be answered due to poor image quality or ambiguous questions. The second is that visual questions only need to query *relevant* information based on the image. We do not need to process the irrelevant parts of the image, given a specific question. Based on these intuitions we develop three techniques: (1) **Attention-based two-stage early exit.** MobiVQA uses a two-layer early exit algorithm that is at-once aggressive and conservative. In the first stage, for questions that cannot be answered, MobiVQA is aggressive and stops processing early. In the second stage, MobiVQA leverages the salient information in the question and image representation to decide if further processing will improve accuracy; if not, MobiVQA exits processing, (2) **Question-aware pruning.** VQA models process an image differently for different questions (see example in Figure 1). However, existing VQA models processes the entire image, irrespective of the question. Instead, we determine the part of the image that is most salient to answer a given question. MobiVQA then prunes the rest of the image so that it does not have to be processed, and (3) **Adapting to grid-based models.** The most accurate VQA models require object detection which is expensive on mobile devices. Instead, we adapt these models to grid-based models where the image features are extracted from the image pixels without any object detection.

[1]new models are being developed and released regularly with the goal of improving visual question answering accuracy. Designing system optimizations that work only for one specific VQA model is likely to become obsolete and not useful.

We implement MobiVQA on two mobile platforms: Nvidia TX2 board [54] and an Android smartphone (Pixel 3XL). We evaluate over two popular VQA datasets: VQAv2 [24] and VizWiz [26]. We find that with MobiVQA optimizations, the latency to answer a question is reduced to 163 ms on the smartphone. In fact, when compared to a cloud VQA model, MobiVQA improves latency by 3.5x even on a fast WiFi connection, since the image does not have to be sent to the cloud provider.

We compare MobiVQA to two state-of-the-art VQA models: LXMERT [52] and X-LXMERT [16]. On the VQAv2 dataset, when compared to the most accurate state-of-the-art model LXMERT, the latency reduces from over 20s to 163ms on the phone for less than 1 point drop in accuracy. When compared to the less accurate X-LXMERT, the latency reduces from 1.8s to 163ms with an *improvement* in accuracy of 3 points when using MobiVQA. We also compare MobiVQA to ViLT [34], the newest VQA model released in 2021. This model does not run on the phone, but when run on the board, we see a 3.1x improvement in latency. Results were similar on the VizWiz dataset.

2 BACKGROUND

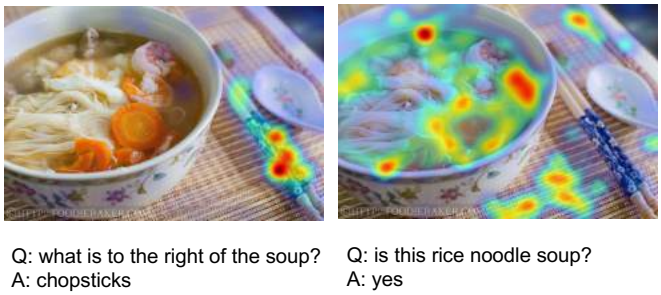


Fig. 1. Two example questions asking information about the same image. The colored overlays are heatmaps that visualize the first layer cross-attention information (which image parts contribute to the answer) in a VQA model.

2.1 VQA Models

A VQA model take as input an image and a natural language question, and produces an answer. Different from image-only vision models or text-only QA models, VQA models need to extract the cross model interaction between the image and the text. For example, Figure 1 shows different questions asked of the same image. The first is the image with the question, "what is to the right of the soup?", for which the answer is "chopsticks". In the second example, the question asked of the *same* image is, "is this rice noodle soup?", to which answers is "yes". The attention given to different parts of the image depends on the question, as shown by the green and orange overlays. We visualize the attention (values are between $[0, 1]$) as heatmap overlays where light color indicates smaller attention values and vice versa. We explain the attention in more detail in §4.2.

The VQA models follow a common pattern in constructing the model architectures as shown in Figure 2. Specifically, the models use three encoders: (1) a vision encoder that uses techniques such as a CNN based object detector to extract object features from the input image, (2) a language encoder such as BERT [18] that take the question tokens as inputs and convert them to a set of word vectors representing the contextual information, and (3) a cross-modal encoder performing the feature fusion between the question and object representations.

The encoders are increasingly being implemented as a transformer [55] that uses self attention to create a rich representation of each modality and its combination. Figure 3 shows the self-attention and cross-attention processes in these encoders.

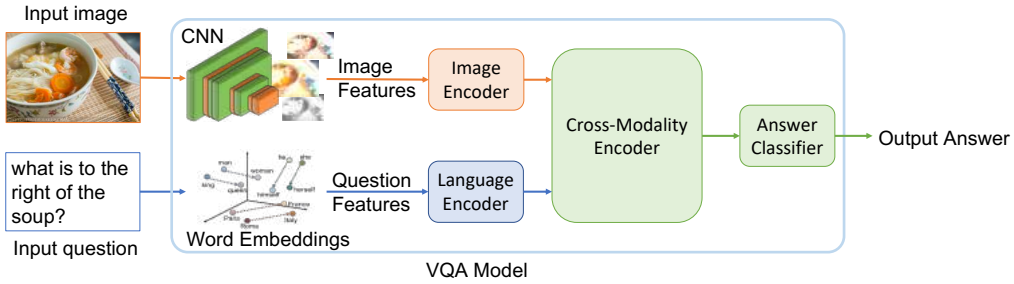


Fig. 2. State-of-the-art VQA model architecture. A VQA model first creates the image and the question features individually, then builds richer representations over the features using single modality encoders, and fuses the two sets of features via a cross-modal encoder, and finally uses an answer classifier to output an answer.

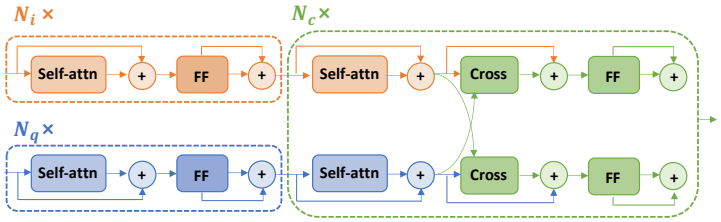


Fig. 3. The self-attention (self-attn) and cross-attention (cross) blocks inside the encoders of a VQA model. "FF" means feed-forward layers. There are N_q layers for the question encoder, N_i layers for the image encoder, and N_c layers for the cross-modal encoder.

2.2 State-of-the-art VQA Models

State-of-the-art VQA models can be categorized as region-based or grid-based; region-based models are more accurate but computationally expensive, while grid-based methods are efficient but less accurate. For example, ViLT [35] and X-LXMERT [16], two state-of-the-art grid-based models have 71.3%[2] and 68.6% accuracy scores which is 1~4 points lower than the state-of-the-art regions-based models LXMERT [52] (accuracy: 72.4%).

Region-based VQA models. Region-based VQA models [14, 41, 52, 66] first detect the objects in the image, using off-the-shelf object detectors Faster-RCNN [49]. They then use the features of each object to model the relationships among the detected objects and use it as the representation. Region-based VQA have higher accuracy compared to grid-based due to fine grained object feature representation via object detection process.

[2]we report the accuracy numbers from the published papers

Grid-based VQA models. Grid-based VQA models [16, 31, 35] do not require the expensive object detection step, and are more efficient compared to region-based models. Instead, the models get features directly from the image pixels without learning the higher level object information. Grid-based VQA models obtain these pixel features from image encoders CNNs [39] or vision Transformers [19] without object detection. But because of the lower accuracy, region-based VQA models are widely used in the VQA community [10, 22]. Given the efficiency versus accuracy tradeoffs in grid-based and region-based VQA models, developing more accurate and faster VQA models is an active research area.

3 ON-DEVICE VQA CHALLENGES

Transformer models are known to consume large amounts of resources even in cloud settings [1], let alone resource-constrained mobile environments. To understand the challenges of running VQA models on the mobile devices, we study three representative state-of-the-art models, one region-based model LXMERT [52] and two grid-based models X-LXMERT [16] and ViLT [35]. We ran these two models on two mobile devices: a next-generation mobile board [54] and a recent, high-end phone [4]. Details are in §6.

We choose LXMERT because it has the highest accuracy. X-LXMERT is a grid version of LXMERT and is touted for the lower latency. ViLT is the newest model released in 2021; it is grid-based and has considerably low latency.

Table 1 shows the latency and number of parameters. LXMERT, the most accurate model, requires nearly 6 seconds to run on the mobile board and over 20 seconds to run on the phone (because of the object detection component). Even the fastest grid-based model requires 651ms on the mobile board. We could not run ViLT on the phone because it has unsupported model operations. However, when run on a Google collab GPU, it took 1.3 seconds. However, to be used seamlessly without perceiving delays, the latencies have to be in the range of 200 to 300ms [47].

Table 1. The latency and accuracy statistics for the one region-based and two grid-based models. *VILT does not run on the phone, so we report numbers from running the model on Google Colab GPU.

Model	Params (million)	Model Storage (MB)	Accuracy (%)	TX2 latency (ms)	Phone latency (s)
LXMERT	237	1446	72.4	5913	>20
X-LXMERT	270	1533	68.6	561	1.8
ViLT	118	449	71.3	651	>1.3*

Table 2 shows the breakdown of the model components in Figure 2 for LXMERT and X-LXMERT. Based on this breakdown, we identify the following challenges in adapting the VQA models to resource-constrained devices.

Dectecting objects is expensive for on-device state-of-the-art VQA models. Region-based models incur significant latency for object detection. Object detectors like FasterRCNN [49] use a ResNet50 backbone to extract feature maps over the image and then use a region proposal network to generate thousands of proposals for potential objects on top of the feature maps, which requires large amounts of computation. VQA models then use the visual feature representations for detected objects. The object detection requires 91% of the total VQA inference processing time on the mobile Jetson TX2 board with a mobile GPU. Since the object detection for our use cases have to be done at runtime, region-based models are not suitable for on-device VQA.

Cross-modal encoding is the bottleneck for grid-based VQA models. Grid-based VQA models remove the need for object detection, but see an accuracy drop. In that case, the key bottleneck is the cross-modal encoder, which takes over 49% of the total inference processing time.

Unfortunately, parallelizing cross-modal layers to speedup the computation is difficult. In the cross-modal encoder, the cross-attention layers are applied to vision and language modalities and cause computational

dependencies that are non-parallelizable. In Figure 3, the cross-attention for each text token or image grid feature relies on the other tokens or grids. Even if the operations (such as matrix multiplications in the self-attention and feedforward layers) inside the cross-modal encoder are parallelizable, the computation dependencies in cross-attention prevent the parallelization of the cross-modal layer. This is in contrast to the question encoder and image encoder in Figure 3 which are independent and can run in parallel. Existing parallelization optimization like MobiRNN [11] addresses computation dependencies in RNNs by offloading to mobile GPUs. However, the RNN dependencies are different from the cross-modal dependencies in Transformers. RNNs process each element of the input data sequentially in each layer at each step, while Transformers process all elements (in our case all image patches and question tokens) at the same time. In Figure 3, the self-attention layer in the cross-modal encoder itself is parallelizable and can process all vision input elements (orange box) or language inputs (blue box) at once, while for RNNs the internal states sequentially depends on each element of the inputs.

Table 2. The latency breakdown for the X-LXMERT and LXMERT VQA model. The second row shows the inference latency (percentage) spent in each component of a VQA model when answering questions from the VQAv2 datasets on the TX2 board. Object detection is the major (>90%) bottleneck for region-based LXMERT VQA model, while for grid-based X-LXMERT VQA model, there is no object detection, cross-modal encoding is the main bottleneck, accounting for nearly half of the total time.

VQA Components	X-LXMERT	LXMERT
1. Image Preprocessing	16.9%	0.5%
2. Object Detection	0%	92.1%
3. Question Encoding	9.1%	0.8%
4. Image Encoding	24.7%	2.2%
5. Cross-modal encoding	49.3%	4.4%

4 MOBIVQA DESIGN

In this work, we introduce MobiVQA, an efficient on-device VQA system that optimizes state-of-the-art VQA models. The resulting model can run efficiently on a mobile device without requiring *any* cloud support. Our goal is to design optimizations that work well across VQA models. We demonstrate the generalizability of our approach for three different state-of-the-art VQA models in §6.2.3. Since most VQA models have a similar architecture (shown in Figure 2), our optimizations work by (a) reducing the computation required by a VQA model and (b) reducing the data that flows through a model. Figure 4 shows the overall architecture of MobiVQA that consists of the following three optimizations:

- **Attention-based two-stage early exit:** Early exit is a well known technique used by several machine learning systems [38, 45, 50, 53]. The idea is to exit the network if the answer cannot be improved by processing more layers. However, we find that existing early exit strategies do not work well *as-is* for the visual QA task. Instead, we design early exit algorithms specifically designed for visual QA.
- **Question-aware image pruning:** A unique characteristic of visual QA is that different parts of the image are salient for different questions. By characterizing the salient regions that are most important to answer a given question, the other image regions can be pruned to reduce processing. We design a question-aware pruning that significantly reduces computation while maintaining accuracy.
- **Adapting to grid-based models:** Region-based models run extremely slow as shown in §6 on mobile devices, our analysis of existing VQA models show that grid-based models are more efficient than region-based models, but at the cost of accuracy. We adapt existing region-based models to grid-based without requiring expensive pre-training, while maintaining the original accuracy.

We discuss the early exit and pruning optimizations first and then describe how we adapt any region-based VQA model to a grid-based model without loss of accuracy.

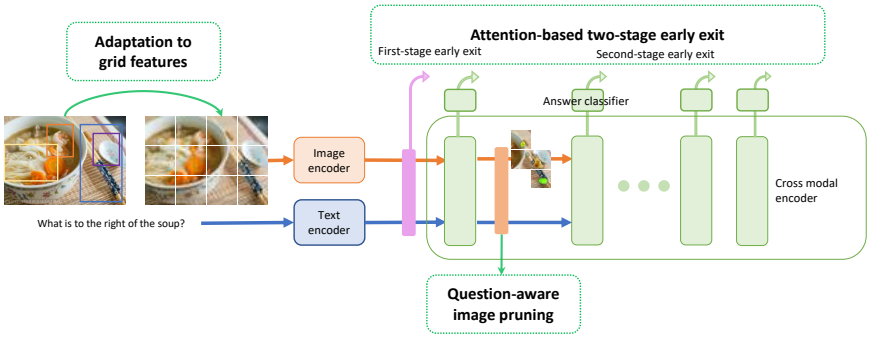


Fig. 4. Overview of MobiVQA design. MobiVQA first converts region-based VQA models to use grid features. Then MobiVQA uses lightweight two-stage early exit and question-aware image pruning to reduce the computation bottleneck in the cross-modal layers of a VQA model.

4.1 Attention-Based Two-Stage Early Exit

Recall that the cross-modal encoding is the biggest bottleneck in existing VQA models (see Figure 2) and optimization techniques such as parallelization cannot be applied here. Instead, in MobiVQA, we develop a two-stage early exit algorithm that exits the cross layer encoding earlier, to avoid expensive computation. The algorithm adaptively decides how much cross-modal processing is needed for different visual questions.

In the first stage, we decide if the input visual questions is too difficult to answer, in which case we fail early and do not enter the cross-modal encoding stage. For the remaining queries, we use an *attention-based* early exit classifier that determines how many layers of the cross-modal encoding is needed. Existing early exit algorithms [38, 45, 50, 53, 61, 68] use prediction probabilities to make decisions on when to exit. However, we find that the prediction probabilities alone are insufficient for VQA tasks.

First Stage Early Exit: We examined the visual questions of two widely studied VQA datasets (VQAv2 and VizWiz [24, 26]). We looked at real questions asked by visually impaired users about their surroundings [25, 26]. 28.6% of these questions are unanswerable because either the image quality is low or the questions are ambiguous [15, 64].

Our goal is to design an end-to-end classifier that uses question and image features to determine if a question cannot be answered. We fuse the image and question features and learn a binary classifier. Specifically, we transform the question features into a single vector $\boldsymbol{q}$ by learning a weighted combination of all question token vectors $\{\boldsymbol{q}_1, \boldsymbol{q}_2, ..., \boldsymbol{q}_i\}$:

$$\boldsymbol{q} = \sum_i a_i \boldsymbol{q}_i \tag{1}$$

$$a_i = \frac{\exp(\boldsymbol{w}_q \boldsymbol{q}_i)}{\sum_{i'} \exp(\boldsymbol{w}_q \boldsymbol{q}_{i'})} \tag{2}$$

and $\boldsymbol{w}_q$ is a weight vector to learn.

The image features are also a sequence of "grid tokens", and each token vector represents a grid feature. Similarly, we map the image grid features $\{\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_j\}$ into a single image vector $\boldsymbol{v}$:

$$\boldsymbol{v} = \sum_j b_j \boldsymbol{v}_j \tag{3}$$

$$b_j = \frac{\exp(\boldsymbol{w}_v \boldsymbol{v}_j)}{\sum_{j'} \exp(\boldsymbol{w}_v \boldsymbol{v}_{j'})} \tag{4}$$

and $\boldsymbol{w}_v$ is also a learnable weight vector.

We then obtain two new feature vectors $\boldsymbol{q}_{new}$ and $\boldsymbol{v}_{new}$ by passing $\boldsymbol{q}$ and $\boldsymbol{v}$ each into a separate linear layer followed by a ReLU activation function. Then we apply element-wise product of $\boldsymbol{q}_{new}$ and $\boldsymbol{v}_{new}$ to fuse the two modality features into a single vector $\boldsymbol{f}$, which is finally fed to a two-class linear classification layer (parameterized by $\boldsymbol{w}_c$) to produce a binary prediction p_f.

$$\boldsymbol{q}_{new} = relu(\boldsymbol{w}_q \boldsymbol{q}) \tag{5}$$

$$\boldsymbol{v}_{new} = relu(\boldsymbol{w}_v \boldsymbol{v}) \tag{6}$$

$$\boldsymbol{f} = \boldsymbol{q}_{new} * \boldsymbol{v}_{new} \tag{7}$$

$$p_f = \boldsymbol{w}_c \boldsymbol{f} \tag{8}$$

The result is a classifier that determines if we can fail early or if we should continue processing a query. We show in the evaluation §4.1 the classifier achieves an average precision score of 0.96 indicating the first-stage classifier effectively identify failure cases.

Attention-based Second Stage Early Exit: For visual questions that can be answered by the VQA model, the correct answer can be obtained at different layers in the cross-modal encoder.

Existing early exit works use prediction probabilities (or derived entropy) as a confidence score for making the exit decisions. However, we find that using prediction probabilities to exit early results in large accuracy drops if bigger inference speedups are desired (see §6). The problem is the nature of the VQA task. This is because the information needed to answer visual questions is not salient in final answer probabilities alone and may rely on more modality relevant signals like the cross-attention between the question and the image. Previous work has shown deep neural networks have the overthinking problem [32] where the shadow layers output correct answers but deeper layers produce wrong answers due to the confusion in the internal model states. Prediction probabilities only capture this confusion before the model's final classification layer but not intermediate states such as cross-attention information that is available in VQA models.

To mitigate this issue, in MobiVQA, we design a lightweight learning based exit classifier that leverages the attention scores in the cross-modal layers as well as the prediction probabilities. *Attention* in deep learning models mimic cognitive human attention to assign larger weights to a specific part of the input information, similarly to how humans pay more attention to salient features. This attention information provides richer features to reduce the confusion in answer predictions and can be used to influence answer confidence. We aggregate the information in the cross-attention between the question and image and define an attention entropy feature. We feed both this attention entropy feature along with the prediction probabilities feature to a second-stage early exit classifier to predict if the answer is correct. For each cross-modal layer, if the exit classifier predicts the answer is correct, we stop further processing, exit the cross-modal encoder and outputs the predicted answer from the answer classifier.

Specifically, we compute the entropy of the attention scores in all attention heads in the cross-modal layers and feed them into a linear classifier together with the prediction probabilities from the answer classifier. We add We denote A^h_{ji} as the attention score over token i to j for the head h for a cross-modal layer, e_h the entropy for head h is $e_h = -\sum_{j=1}^{N_j} \sum_{i=1}^{N_i} A^h_{ji} \log A^h_{ji}$. The classifier is defined as $c = \boldsymbol{w}_1^T \boldsymbol{e} + \boldsymbol{w}_2^T \boldsymbol{p} + b$, where $\boldsymbol{w}_i^T$ and b are

trainable weights, $\boldsymbol{p}$ is the prediction probability vector and $\boldsymbol{e}$ the entropy for all heads. During inference, for each cross-modal layer, we obtain the prediction probabilities from the layer classifier and the attention scores from the cross-modal layer, and compute the exiting probability c, if c is greater than a selected threshold, the classifier exits. The detail procedure is described in Algorithm 1.

Algorithm 1 Attention-based second-stage early exiting

for l in 1...n **do** ▷ n is the number of cross-modal layers
  get attention scores $\boldsymbol{A}_{ji}$ from all heads in cross-modal layer l ▷ $\boldsymbol{A}_{ji} \leftarrow [A^1_{ji}, A^2_{ji}, ..., A^h_{ji}]$
  get prediction probabilities $\boldsymbol{p}_l$ from the answer classifier in cross-modal layer l
  $\boldsymbol{e} \leftarrow -\sum_{j=1}^{N_j}\sum_{i=1}^{N_i} \boldsymbol{A}_{ji} \log \boldsymbol{A}_{ji}$ ▷ compute the entropy of attention scores in cross-modal layer l
  $c \leftarrow \boldsymbol{w}_1^T \boldsymbol{e} + \boldsymbol{w}_2^T \boldsymbol{p} + b$ ▷ compute the exiting probability of cross-modal layer l
  if $c \geq threshold$ **then**
    return argmax($\boldsymbol{p}_l$) ▷ early exit from the cross-modal layer l, return predicted answer
  end if
end for
return argmax($\boldsymbol{p}_n$)

Overall, the **t**wo-stage **e**arly **e**xiting (**TEE**) algorithm works as follows: first, we detect questions that cannot produce a satisfactory answer and for these we fail early and avoid expensive cross-modal computation; then we employ an attention-based exit classifier that predicts which cross-modal layers to exit. We describe the label collection and classifier training process in more details in §5.

4.2 Question-aware Image Grids Pruning

The second optimization leverages the unique characteristic of visual QA. Recall from Figure 1 that given the same image, different questions need to be processed differently to produce different answers. To illustrate this further, we visualize the cross-modal attention for the two different questions in Figure 1 as heatmaps. Darker color (e.g., orange or yellow) overlays show the attention distributes more weights and lighter colors (e.g., blue or purple) indicates the regions are less attended. For the left question, "what is to the right of the soup?", the cross-modal encoder attends most to the chopsticks on the right part of the input image. For the right question, "is this rice noodle soup?", the VQA model scatters the attention to almost the entire input image, indicating the need to process the whole image to produce a good answer.

If we can identify what image regions/grids are needed for further processing earlier, we could potentially reduce the computation by pruning the remaining parts of the image. Based on the intuitions above, we design a **q**uestion-aware **i**mage **p**runing (**QIP**) method.

Specifically, we take the question to image cross attention scores in the first cross-modal layer. For each image grid (each grid can be viewed as a visual "token"), we compute the importance score by summing up the corresponding attention scores over all question tokens, averaged across all attention heads. We sort the image grids and only process the top-k image grids further through the cross-modal layers processing. We define the importance score s_i of image token $\boldsymbol{v}_i$ in attention head h as: $s_i = \frac{1}{N_h}\sum_{h=1}^{N_h}\sum_{j=1}^{N_q} A^h_{ji}$, where N_h is the number of attention heads in the cross-modal layer, j stands for a question token, N_q is the number of question tokens, and A^h_{ji} is the attention score. The image token importance score is defined over the question to image attention information A_{ji} and thus we call this method question-aware image pruning.

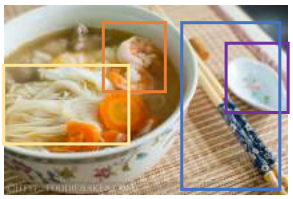
4.3 Adapting Region-based to Grid-based Models without Loss of Accuracy

Recall that grid-based VQA models incur lower latency and are more suited for on-device implementations because they do not perform expensive object detection. However, the model is less accurate compared to region-based models that first detect objects and then answer the visual question. Figure 5 shows the image processing differences between the two types of VQA models. Region-based VQA models process the input image by first detecting the objects and then only use feature representations of the extracted objects for later processing. Grid-based VQA models divide the input image into smaller patches (grids) and represent every image grid for further processing.

Existing grid models such as X-LXMERT [16] adapt a region-based model (in this case, LXMERT [52]) by pre-training the image model using the features from the grid. Pretraining is an expensive large-scale training process where the model is trained on large amount of unlabeled data to learn useful representations for various down-stream tasks like VQA. The problem is that this pre-training causes misalignment in the cross-modal interactions between image objects and textual concepts. The original LXMERT [52] reasons with question tokens using region features (i.e. features for each detected objects) that represent fine-grained visual semantics, such as “chopsticks to the right of noodles". X-LXMERT is pre-trained to reason over coarse-grained grid image features, such as, “pieces of chopsticks to the right of several parts of noodles". This pre-training regularizes the model to perform coarse-grained reasoning that misaligns image objects and question concepts because the model was previously pre-trained to reason over fine-grained features. Recent VQA models like ViLT [35] also design architectures that use grid features from scratch. By pre-training on large-scale (often more than several millions) image captioning datasets, the models learn coarse-grained features that are useful for image and text reasoning tasks like VQA. But grid-based models have lower accuracy due to such coarse-grained feature alignments. Annotating image caption datasets with fine-grained information (such as which object maps to which word) may help the models improve VQA performance but requires substantial efforts. The community is also doing active research on techniques that can exploit more fine-grained alignments automatically between image and text without such fine-grained annotation.

Our goal in MobiVQA is to adapt any given region-based model to grid-based model without incurring large pre-training costs or losing accuracy. Our observation is that the end-to-end pre-training using grid features both results in loss of accuracy and makes adapting region-based models to grid-based models expensive. Instead, MobiVQA adapts any region-based model to grid-based without requiring pre-training. MobiVQA uses a grid-based feature extractor (some extractors are already available [31]) to extract image grid features. We replace the object detection module in region-based VQA models with the grid-based feature extractor. We then fine-tune the adapted model directly for the VQA tasks. Compared to pre-training, fine-tuning is a relatively lightweight training process where a model is trained on the target task (in our case VQA) that has a smaller scale (tens to hundreds of thousands) dataset.

This simple adaptation keeps the accuracy within 1% change; the accuracy of our adapted grid-based LXMERT is 71.5% and the original region-based LXMERT 72.1%. Our adaptation is effective because the fine-tuning adaptation does not break the reasoning skills acquired during original region-based models pre-training. The shorter training time during fine-tuning forces the VQA model to learn the mapping (i.e. adjusting quickly to fit the task labels) between grid and region features for the specific VQA task. This is possible because grid features and region features are essentially learned from the same images, if the reasoning skills acquired during pre-training are not lost, fine-tuning forces this kind of feature realignments thus maintaining similar VQA task performance.



(a) Region-based models process an image by detecting objects (as shown in colored boxes) in the image.

(b) Grid-based models process an image by chopping the image into smaller grids and are not object-aware.

Fig. 5. Region- vs grid- based VQA models.

5 IMPLEMENTATION

We implement MobiVQA using the PyTorch [48] v1.8.1 along with PyTorch Lightning [20] v1.3.8 for training and inference. We implement MobiVQA on both a mobile development board [54] and an Android smartphone. MobiVQA is implemented directly on the board. However, the VQA model, and the MobiVQA optimizations, have to be converted to PyTorch traced mobile format to run on Android. We release the code at https://github.com/SBUNetSys/MobiVQA.

5.1 VQA datasets.

We use two standard VQA datasets in our implementation: VQAv2 [24] and VizWiz [26]. VQAv2 dataset consists of open-ended diverse questions about images both from the MSCOCO [43] dataset (designed for large-scale object detection, segmentation, and captioning) and real world abstract scenes. The V2 version is more challenging than its previous version [6] because of complementary images where most questions are asked about a pair of similar images that result in different answers. Answering these questions requires an understanding of vision, language and commonsense knowledge.

VizWiz dataset was collected in a natural visual question answering setting where blind people were asked to take an image and record a spoken question about it, and then crowd workers answer these visual questions. The questions are often are more conversational and 28.6% of the questions cannot be answered because the images are noisy and low quality.

5.2 MobiVQA optimizations

We implement the three optimizations as follows:

Two-stage early exit classifier. For existing VQA models, there is only one answer classifier at the final layer. However, early exit requires that an answer classifier is added to all layers, so that they can exit from any one of the layers. To this end, for a given VQA model, we add an answer classifier to all the cross-modal layers. The weights are initialized using weights from the original VQA model. We then train this multi-classifier VQA model so that it can exit from any cross-modal layer.

To train the classifiers we need ground truth labels for exit decisions at different layers of the network. The label is collected as follows (1) for first-stage early exit, for a given question, if the model fails to return a correct answer from any of the answer classifiers, we label it as exit, otherwise we label it as continue. Then we train the first stage-early exit classifier algorithm described in §4.1 using a binary cross-entropy loss. (2) in the second-stage early exit, the ground truth exit layer is the earliest cross-modal layer when the answer is correct. We use a multi-class cross-entropy loss to supervise the second-stage exit classifier described in §4.1. In both cases, we set

the exit criterion as a tune-able threshold, such that if the probability from the exit classifier is greater than the threshold then we exit the network.

Implementing the question-aware pruning. We first estimate an *importance* score of each image grid based on the question and the attention information as discussed in §4.2. We then set an optimal pruning ratio that provides the best performance on the validation set. For example, a pruning ratio of 0.1 means we remove 90% of the image grids based on the importance score.

Adapting region-based to grid-based models. We use the grid feature extractor [31] with ResNet50 as the CNN backbone to extract grid features for the input image. To adapt a region-based models to use grid features, we replace the object detection features with extracted grid features as inputs to the VQA model.

5.3 Training and testing

We perform the training tasks under fp16 precision using a cluster with 4 Nvidia V100 GPUs. We use the following hyperparameters during finetuning of all VQA models: training batch size of 256, 3 epochs with a learning rate of $5 * 10^{-5}$ and a warmup ratio of 0.1. We also clip the gradient values to 0.5.

The VQAv2 dataset has 443K training samples, 214K validation samples and 453K testing samples in the form of question-image pairs. To avoid frequent submissions to the evaluation server, we do not use the test set. Instead, from the validation set, we sample 20k examples as our validation set, and another 12895 examples as our test set. This validation set and test set together uses 15% examples of the original validation set, we merge the rest of the 85% examples to the original training set for training.

VizWiz contains 20523 training, 4320 validation, and 8000 test image/question pairs. For development purposes, we do not use the test set, and instead divide the validation set into 2160 examples for our validation set and 2160 examples for our test set. We train on the 20523 training samples. We train all models only on the training set, select hyperparameters based on the accuracy on our validation set, and report the accuracy on our test set.

5.4 Setting threshold and pruning ratio.

There are two parameters that need to be set for MobiVQA. The first is the *exit threshold*. This threshold determines when MobiVQA will exit processing in the first and second stage. We use the validation dataset and different values of exit thresholds and choose the threshold that provides the best performance. We use this value in the test set. For VQAv2 dataset, the optimal threshold is 0.45 for first-stage early exit and 0.37 for second-stage early exit. For VizWiz dataset, the optimal thresholds are 0.65 and 0.58 for first-stage and second-stage early exit respectively. The second parameter is the pruning ratio for question-aware pruning. As before, we experiment with different pruning ratios on the validation set. For the VQAv2 dataset, the pruning ratio was set to 0.9, and for the VizWiz data set the ratio was set to 0.85.

6 EVALUATION

We optimize three state-of-the-art VQA models using MobiVQA optimizations. We then compare the MobiVQA optimized models with the original model in terms of accuracy, latency, and energy consumption. Our experiments are conducted on two different mobile devices and two VQA datasets. Our evaluation results show that

- On an average, MobiVQA takes less than 169ms to answer a question on a smartphone. When compared to cloud VQA , MobiVQA improves the latency of a cloud implementation of MobiVQA by 52x, 6.2x, and 3.5x for 3G, LTE, and WiFi, respectively. This is because, a cloud VQA requires that the image be sent to the cloud which incurs additional latency.
- MobiVQA reduces end-to-end QA time by an average of 66x on the phone (up to 121x speedup), across datasets and baselines. On the mobile board, MobiVQA reduces latency by an average of 6.8x (up to 16x).

The energy reductions are commensurate with the latency reduction while the accuracy drop is less than 1 point.
- MobiVQA's early exit algorithm has lower accuracy loss compared to existing probability-based and entropy-based early exit algorithms [38, 50, 53, 60]

We start by describing the evaluation methodology.

6.1 Evaluation Methodology

We evaluate MobiVQA on two datasets, VQAv2 [24] and VizWiz [26]. We describe the datasets, the training, and testing methodology in the previous section. All results are based on the method described in §5.

6.1.1 Devices. We conduct experiments two devices: (1) Mobile Board: a Nvidia Jetson TX2 [54] development board with quad core ARM 64bit CPU, a 256-core CUDA GPU, 8GB memory, and 128GB storage. (2) Mobile Phone: A Pixel 3XL Android Smartphone running Android 12.0. The phone runs a Snapdragon 845 chipset, with 4GB RAM and 64GB storage. The smartphone also has a Adreno 630 GPU, but we focus on evaluating the performance on CPU due to incomplete support of the PyTorch mobile framework.

6.1.2 Baselines. We compare MobiVQA against three baseline VQA systems. *LXMERT* [52] is a region-based VQA model that has the highest accuracy among VQA models to-date. *X-LXMERT* [16] is the grid-based model adapted from the original region-based LXMERT. The third baseline is the most recent model called *ViLT* [35] released in 2021.

In the case of LXMERT and X-LXMERT, we apply MobiVQA optimizations to LXMERT and compare the performance of the optimized model with the original models. In case of ViLT, we apply the optimization to the ViLT model and then compare the original and optimized models. We apply the grid adaptation optimization only for region-based models.

6.1.3 Evaluation Metrics. The evaluation metrics are as follows:

(1) **Accuracy**: We use the same accuracy definition $min(\frac{N_{ans}}{3}, 1)$ as introduced in the VQAv2 and VizWiz dataset, where N_{ans} is the number of humans annotating the *ans*. This accuracy metric is reported to be more robust to inter-human variability in phrasing the answers.

(2) **Latency**: We define the end-to-end latency as the duration from receiving the text question and input image to producing an answer for the VQA systems. For all latency measurements, we run each model 10 times across 1000 randomly sampled examples from the validation set and report the average inference latency.

(3) **Energy**: We measure energy consumption of the end to end VQA inference using the internal on-board power monitor on the Jetson TX2 board. We use the Android dumpsys [2] tool to report the battery consumption on the phone.

6.2 MobiVQA Benefits

6.2.1 MobiVQA Improves VQA Inference Latency and Causes Minimal Accuracy Loss. Table 3 shows the latency, energy and accuracy results when comparing LXMERT and X-LXMERT, with MobiVQA. Recall that MobiVQA is an optimized version of LXMERT. Compared to region-based LXMERT VQA model on the VQAv2 dataset, MobiVQA reduces the latency by **16x** on the mobile TX2 board and over **121x** on the mobile phone with less than 1% accuracy loss on both VQAv2 and VizWiz datasets.

Compared to X-LXMERT (the grid-based model), MobiVQA provides **1.5x** speedup for TX2 board and **10.9x** speedup for smartphone. The latency benefits are similar for the VizWiz dataset. Note that the speedup for TX2 board is relatively smaller compared to the phone. This is because we do not optimize the CNN (ResNet50) grid feature extractor. Applying orthogonal optimizations such as pruning or quantization would help further reduce the latency but is not our focus in this work. On the phone side, the PyTorch mobile inference runtime uses

JIT-trace[3] based optimized model format that is able to optimize the CNNs and gives a larger speedup. This optimization is available to all models, not just MobiVQA.

In terms of accuracy, MobiVQA sees a less than 1 point drop compared to LXMERT. In contrast, X-LXMERT, the grid-based model that is adapted from LXMERT sees an accuracy drop of 4% on both datasets. In other words, MobiVQA improves latency significantly without significant accuracy drop.

6.2.2 MobiVQA Consumes Less Energy on Mobile Devices. Table 3 shows MobiVQA effectively reduces the VQA energy consumption on the mobile board by **19x** and the smartphone by **18x** respectively for the VQAv2 dataset compared to LXMERT. This result is especially significant for users with visual impairments because blind users are disproportionately worried about battery drain [62]. Even compared with the grid VQA model X-LXMERT, MobiVQA saves the energy by **3.1x** and **2.8x** for the VQAv2 and VizWiz datasets respectively. MobiVQA also saves the energy on the mobile board for both VQA datasets.

For baseline models the latency and energy numbers are the same for different datasets because the model only differs in model weights and not computation complexity. However, MobiVQA can have different latency and energy numbers for different datasets because the two-stage early exit and question-aware image pruning can be configured with different threshold values. In fact, this allows MobiVQA to explore the trade-off space between latency and accuracy as different thresholds result in different trade-offs. We explore this further in §6.5.

Table 3. Comparing the end to end latency, energy and accuracy for MobiVQA with X-LXMERT and LXMERT VQA models. *Running the object detector for the region-based VQA model LXMERT alone takes over 20 seconds on the phone, which translates to more than 50J energy consumption.

	Model	TX2 Latency	TX2 Energy	Phone Latency	Phone Energy	Accuracy (%)
VQAv2	LXMERT	5.9 s	106.2 J	> 20 s *	> 50 J	72.4
	X-LXMERT	531 ms	8.5 J	1.8 s	8.8 J	68.6
	MobiVQA	**361 ms**	5.6 J	**165 ms**	2.8 J	**71.8**
VizWiz	LXMERT	5.9 s	106.2 J	> 20 s *	> 50 J	53.2
	X-LXMERT	531 ms	8.5 J	1.8 s	8.8 J	48.7
	MobiVQA	**382 ms**	6.2 J	**172 ms**	3.1 J	**52.4**

6.2.3 Comparing MobiVQA with Other VQA Models. The VQA models are moving target since the research community is actively developing newer and better model architectures. It is important to understand how MobiVQA optimizations work for other types of VQA architectures. We applied MobiVQA optimizations to ViLT [35], the state-of-the-art VQA model released in 2021. ViLT has the same VQA pipeline as the other models, but does not use CNNs for image feature extraction. Instead, ViLT only uses the transformer architecture to build visual and textual features from the input question and image.

Unlike the other VQA models, ViLT does not have two stages; both the image and question representation and the cross-modal encoding are performed using the same transformer architecture. So, we use the first layer to apply the first-stage early exit, and then apply the second-stage early exit for subsequent layers. The image pruning starts from the second cross-modal layer (since the first layer is used for the first-stage early exit). We tune the exit thresholds and pruning ratios on the local validation set to find the best tradeoffs between the accuracy loss and inference speedups.

[3]See https://pytorch.org/mobile/android/#use-pytorch-jit-interpreter.

Table 4. The latency and energy numbers of ViLT on the TX2 board for the VQAv2 dataset.

Model	TX2 Latency	TX2 Energy	Accuracy (%)
ViLT	651 ms	11.2 J	71.3
MobiVQA-ViLT	**213 ms**	3.3 J	70.5

As shown in Table 4, we observe a 3.1x reduction in inference latency (from 651 ms to 213 ms) for less than a 1 point drop in accuracy for the VQAv2 dataset. We are unable to run ViLT on the mobile device due to unsupported model operations on the mobile.

6.3 MobiVQA vs Cloud VQA

One advantage of MobiVQA is that the image does have to transferred to the cloud, avoiding further cloud latency and cellular data costs. To evaluate the latency when using the cloud, we assume that the image on the users phone needs to be uploaded. We randomly selected 10 images from the validation set of the VQAv2 dataset and the average size of these images are 196KB.

In these experiments, we upload the image of 196KB, run the VQA model on the cloud, and get the response. We performed this experiment on WiFi (bandwidth=169Mbps, RTT=27ms), LTE (bandwidth=58.5Mbps, RTT=78ms), and 3G (bandwidth=10.6Mbps, RTT=200ms). We then run the MobiVQA optimized LXMERT on the cloud device. The cloud device specifications were: 3.4GHz CPU, GTX 1080Ti GPU and 32GB memory. We found that default LXMERT and X-LXMERT is slower on the cloud device compared to MobiVQA, so we compare the performance with a cloud version of MobiVQA.

Table 5 shows that MobiVQA improves latency over the cloud VQA (also using MobiVQA) by 52x, 6.2x, and 3.5x for 3G, LTE, and WiFi, respectively. It is possible that a cloud device will be more powerful than the one we used, but the results suggest that the communication latency is the biggest bottleneck, and even if the cloud compute latency is reduced to 0, the local MobiVQA is 3.2x faster even in the best case, under WiFi.

Table 5. The latency of cloud VQA vs MobiVQA.

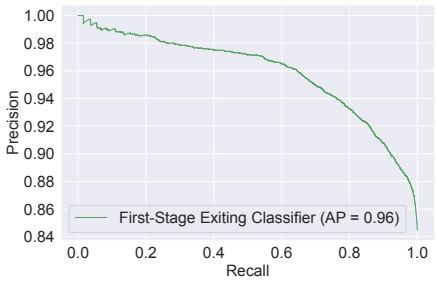
Model	Network Latency	Cloud Compute Latency	Total Latency
MobiVQA 3G	8691 ms	45 ms	8736 ms
MobiVQA LTE	980 ms	45 ms	1025 ms
MobiVQA Wi-Fi	530 ms	45 ms	575 ms
MobiVQA-Local	0 ms	**165 ms**	**165 ms**

6.4 Effectiveness of MobiVQA Optimizations

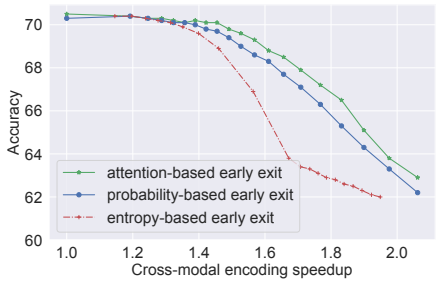
MobiVQA consists of three key optimizations: adapting region-based VQA models to be grid-based (**adaptation**), attention-based two-stage early exit (**TEE**) and question-aware image pruning (**QIP**). Table 6 shows the latency and accuracy numbers when applying each of the three techniques by themselves. Combining all three gives the best inference speedups with minimal accuracy changes.

Specifically, adaptation alone improves the accuracy by 3 points compared to the grid-based X-LXMERT VQA model and maintains the inference speed. Comparing with the region baseline model, adaptation provides over 11x speedups while keeping the accuracy loss to less than 1 point.

Applying attention-based two-stage early exit (TEE) reduces the latency by over 12x compared to the region-based model with negligible loss (0.3%) in accuracy. It is also effective in improving the accuracy by over 3.6% to



(a) Precision-recall of the first-stage early exit. The average precision is 96% indicating the first-stage early exit is able to distinguish between unanswerable questions and questions that can be answered.



(b) Accuracy versus cross-modal encoding speedup for comparing the attention-based (MobiVQA) second stage early exit with probability- and entropy-based exit methods. We vary the exit thresholds to get different samples.

Fig. 6. Comparing the attention-based two-stage early exit algorithm with existing works that use probability- or entropy- to make early exit decisions.

the grid-based model and still provides 1.1x speedup. This is because we tune the two exit thresholds to be less aggressive while maintaining accuracy (we discuss the trade-offs between accuracy and latency in §6.5).

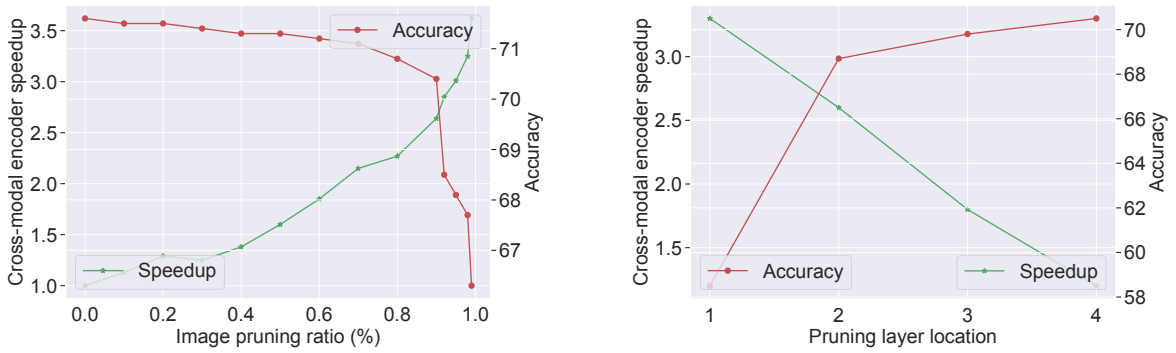
Compared to the region-based model, question-aware image pruning (QIP) has a relatively bigger accuracy loss (1.2%) but comes with a 13x latency speedup. When comparing with the grid-based model, QIP also effectively reduces the latency by 1.1x while providing 2.6% higher accuracy. Again, one can tune the pruning ratios in QIP optimization to trade-off between latency and accuracy.

Table 6. Latency and accuracy results of only applying the each optimization in MobiVQA by itself.

	TX2 Latency	VQAv2 Accuracy (%)
LXMERT baseline	5.9 s	72.4
X-LXMERT	531 ms	68.6
MobiVQA	361 ms	71.8
with only adaptation	531 ms	71.5
with only TEE	482 ms	72.1
with only QIP	462 ms	71.2

6.4.1 Impact of Two-stage Early Exit. In this subsection, we explore the characteristics of the two-stage early exit optimization. The question we ask is—how often does MobiVQA make the right decision to stop processing in the first or second stage If MobiVQA is wrong in many cases, then the users will not get any answer (in the case of first stage exit) or will receive a poor answer (in case of the second stage exit).

Our early exit algorithm is a classification problem: a positive label indicates the question is likely to get an answer and thus needs further processing, a negative label means the model is unlikely to find a correct answer and stops the processing. We use a Precision-Recall curve to characterize the performance of our classification. Figure 6a shows that the first-stage exit classifier is able to identify the failure cases with an average precision of 96%.



(a) Accuracy versus cross-modal encoder speedup for different pruning ratios in the question-aware image pruning.

(b) Accuracy versus cross-modal encoder speedup for applying the question-aware image pruning at different layers.

Fig. 7. Understanding the performance of the question-aware image pruning in MobiVQA on the VQAv2 dataset.

Figure 6b shows the precision of our second stage early exit. Recall that in this stage, MobiVQA uses an attention based early exit algorithm, unlike related work that use prediction probabilities [38, 50] or entropy [53, 60]. We compare the latency and cross-modal encoding speedup of MobiVQA with baseline early exit algorithms that use prediction probabilities or entropy.

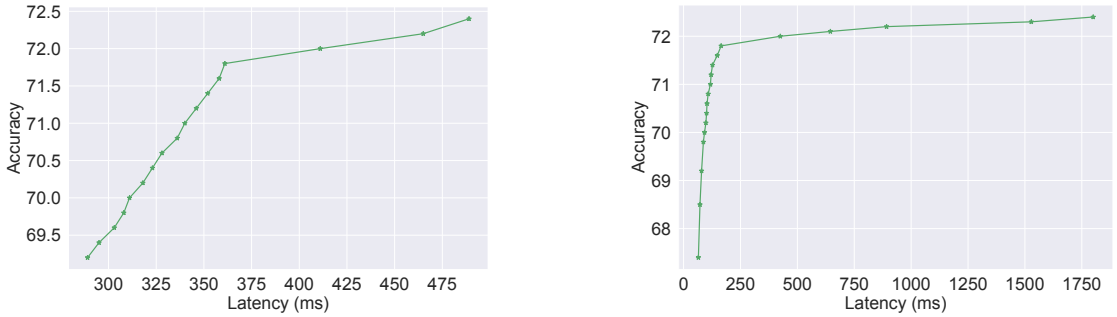
The x-axis shows the inference speedups and y-axis shows the accuracy. The upper right region gives better latency versus accuracy trade-offs. Our attention-based TEE curve lies above the upper right of both probability- and entropy-based curves. In other words, MobiVQA's early exit algorithm performs better than probability or entropy based early exit methods. For example, while providing the same speedup of 1.6x, TEE is 0.6% more accurate than probability-based early exit and over 3% more accurate than entropy-based methods.

6.4.2 Impacts of Question-aware Image Pruning. To quantify the pruning effects in question-aware image pruning (QIP) optimization, we (1) compare pruning at different layers in the cross-modal encoder of a VQA model and (2) vary the pruning ratios to understand the accuracy versus latency tradeoffs. For both experiments, we only apply the adaptation optimization first to the region-based LXMERT model so that the accuracy is close to baseline region-based models. The model has 5 cross-modal layers and has 608 image grids.

Figure 7a shows the accuracy versus latency trade-offs under different pruning ratios. We find pruning up to 80% of the image grids does not hurt the accuracy too much (less than 1%), while still see large latency speedups (over 2x). Nevertheless, pruning over 95% of the image grids causes significant accuracy loss (over 3%). This verifies that only a small portion of the image information is crucial for the model to output a correct answer. In MobiVQA for the VQAv2 dataset, we set the pruning ratio to 90%.

For pruning at different layers, we set a fixed pruning ratio (90% for the VQAv2 dataset), fine-tune the VQA model and measure the accuracy and latency speedups in the cross-modal encoder[4]. Figure 7b shows the trend that pruning at a later layer provides less than 1.5x speedups while maintaining a higher accuracy, pruning earlier (before layer 2) gives better speedups but achieves lower accuracy. For simplicity, in MobiVQA, we prune the image at 1st layer for speedup considerations, but we can tune the pruning ratio to maintain a better accuracy.

[4]we do not compare end to end latency as this will confound the pruning effects because other components such as image feature extraction also contributes to the latency delays.



(a) TX2 board latency versus accuracy tradeoffs for different pruning and exit thresholds.

(b) Phone latency versus accuracy tradeoffs for different pruning and exit thresholds.

Fig. 8. MobiVQA allows accuracy versus latency trade-offs. Numbers are measured on the VQAv2 dataset.

6.5 MobiVQA Allows Flexible Accuracy and Latency Trade-offs

As shown in the §6.4, setting different exit thresholds for the two-stage early exit and different pruning ratios for the question-aware image pruning can produce different accuracy versus latency trade-offs in the cross-modal encoder of a VQA model. In this section, we plot in Figure 8 the end to end latency and accuracy tradeoffs by varying the thresholds. We change the threshold in the range of $[0.2, 0.9]$ with a 0.05 step for both of the two stage early exits. We also change pruning ratios across the ranges from $[0.01, 0.1]$ with a 0.01 step, and from $[0.1, 0.9]$ with a 0.1 step.

MobiVQA is flexible for deployment and allows different accuracy and latency trade-offs. For example, if the user cares more about the latency and can tolerate less accurate predictions, as shown in Figure 8b, we get 15x latency speedup on the phone but with a relatively larger accuracy loss 1.4%. Similarly, if a higher accuracy like 72.0% (lose 0.4%) is desirable, we still get 4x latency benefits on the phone.

6.6 MobiVQA Runtime Overhead

As we show earlier, with the TEE and QIP optimizations, MobiVQA brings both latency and energy benefits. We further study how each of the two optimizations contributes to the on-device runtime inference overhead, i.e. the inference delay of executing the optimization alone. For TEE, the extra computation is in the two early exit classifiers, we empirically find the latency for the classifier is no more than 1.1% (i.e. <5 ms) of the model inference latency and remains constant regardless of the thresholds. For QIP, the extra computation is in the pruning metric calculation (aggregation of attention scores) and top-k selection of image grids. Although the runtime overhead increases as we prune more images (e.g. from 1% to 50%), it remains within 2.7% (i.e. <10 ms) of the the overall model latency.

7 RELATED WORK

Efficient deep learning has seen significant progress in the last few years, especially in vision and NLP. Much of these works focus on optimizing vision-only tasks [29, 37, 63] or NLP/text-only [11, 13, 51] tasks. However, emerging applications such as VQA require different optimizations to address the complex interactions between different modalities. In this section, we discuss related work on deep learning optimizations for vision and NLP.

VQA Models for Mobile. Although there is no work on directly optimizing VQA models for mobile devices, one could combine vision optimizations and text optimizations to answer visual questions. For example, MiniVLM [57]

uses a two-stage efficient object detector to extract region features from the image. This image feature extraction model can be combined with a more efficient text model such as a smaller BERT [18] model to encode the question. Similarly, DistillVLM [21] introduces a small student language vision model to distill the knowledge from a larger teacher model. However, these smaller models lead to significant accuracy loss (over 3%) compared to state-of-the-art models that are specifically designed for visual question answering. More importantly, both MiniVLM and DistillVLM require repeated large scale pre-training on many GPUs which is expensive. Our MobiVQA optimizations do not have such prohibitively expensive pre-training costs.

Early Exit. BranchyNet [53], DenseNet [28], and Shallow-Deep Network [32] are pioneering works in the computer vision community that started the early exit idea for efficient neural network. Recently, the NLP community has also adopted the early exit idea for large transformer language models [45, 50, 60]. SPINN [38] comprehensively uses multi-exits in CNNs for progressive mobile-cloud inference. A more comprehensive study of early exit research can be found here [46]. However, existing early exit algorithms either use the answer probabilities or entropy over the answer probabilities as the exit criterion, our evaluation shows that these decisions result in a drop in accuracy. Instead, MobiVQA uses attention information that takes into account the cross-modal vision and text information.

Caching, Offloading, and Parallelization. Mobile community has studied running DNNs efficiently on mobile device either via computational reuse like caching [63], parallel offloading [67], or parallelization [40]. However, VQA models have inherent data dependencies in the cross-modal interaction layers which prevents the use of these techniques. MobiRNN [11] uses mobile GPUs to efficiently run RNNs that have different input data dependencies than state-of-the-art Transformer models. RNNs process the input sequence data step by step while Transformers use self-attention to process all input data at the same time. Transformers has shown to cause one to two orders of magnitude less training costs whiling achieve better accuracy for many tasks than RNNs [18, 55] A recent work [42] introduces speculative inference for multimodal data application on mobile. However, their speculative inference works for asymmetric multimodal streams such as video or audio and cannot be directly applied to non-streaming applications like VQA. DeQA [13] is a recent work on optimizing deep QA models for mobile devices but works by optimizing the large volume of text data. The question answering task requires that the model searches through a large collection of text data to find the answer. However, in VQA, the amount of data that needs to be processed is small because the answer is contained within one image and not in a large collection.

Optimizing DNNs using Compression. Compressing DNNs by pruning [27, 44, 65] or quantization [30, 58, 59] has been shown to be effective for reducing the model sizes and improve inference latency in resource-constrained environments. Compression methods are orthogonal to our approach and one can apply them on top of MobiVQA for further improvements. Closer to our question-aware image pruning algorithm, token pruning approaches [23, 33, 56] selectively eliminate all possible input word vectors based on the attention scores. Our question-aware image pruning method uses the question to image cross-attention scores to measure the importance of image tokens, which better utilizes the cross-modal information. The MobiVQA pruning also reduces image redundancy instead of removing tokens possibly in the entire input sequence.

8 CONCLUSION

On-device visual question answering or VQA can empower users to get answers for visual questions privately and at low cellular costs. The problem is that existing VQA models use deep learning algorithms that are expensive to run on mobile devices. The VQA task is unique in that it requires multi-modal fusing of text representation as well as image representation. Unfortunately, there are no existing optimizations that adapt multi-modal applications such as VQA for mobile devices. In this paper, we design MobiVQA, a set of optimizations that specifically leverage the multi-modal nature of the VQA task. MobiVQA adapts the accurate but slow region-based VQA

model to faster grid-based model without sacrificing accuracy. MobiVQA then uses an attention-based two-stage early exit algorithm that exploits the cross-modal information in VQA to dynamically exit the model and reduce computation. This is coupled with a question-aware pruning technique that prunes part of the image that cannot be used to answer the question. These optimizations are general and we show that it can be applied to several existing VQA models. Extensive evaluations show that MobiVQA substantially reduces the latency and energy compared to three state-of-the-art VQA models, with minimal accuracy loss.

REFERENCES

[1] [n.d.]. Bing delivers its largest improvement in search experience using Azure GPUs. https://azure.microsoft.com/en-us/blog/bing-delivers-its-largest-improvement-in-search-experience-using-azure-gpus/

[2] [n.d.]. DumpSys. ([n. d.]). https://developer.android.com/studio/command-line/dumpsys.html

[3] [n.d.]. Name that tune: Brain takes just 100 to 300 milliseconds to recognize familiar music. https://www.sciencedaily.com/releases/2019/10/191030073312.htm

[4] 2021. Pixel 3 XL. https://en.wikipedia.org/w/index.php?title=Pixel_3&oldid=1062029816 Page Version ID: 1062029816.

[5] Peter Ackland, Serge Resnikoff, and Rupert Bourne. 2017. World blindness and visual impairment: despite many successes, the problem is growing. *Community Eye Health* 30, 100 (2017), 71–73. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5820628/

[6] Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C. Lawrence Zitnick, Dhruv Batra, and Devi Parikh. 2016. VQA: Visual Question Answering. *arXiv:1505.00468 [cs]* (Oct. 2016). http://arxiv.org/abs/1505.00468

[7] Stefano Aldegheri, Silvia Manzato, and Nicola Bombieri. 2018. Enhancing performance of computer vision applications on low-power embedded systems through heterogeneous parallel programming. In *2018 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*. IEEE, 119–124.

[8] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*. 2425–2433.

[9] Erin Brady, Meredith Ringel Morris, Yu Zhong, Samuel White, and Jeffrey P. Bigham. 2013. Visual challenges in the everyday lives of blind people. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. Association for Computing Machinery, New York, NY, USA, 2117–2126. https://doi.org/10.1145/2470654.2481291

[10] Emanuele Bugliarello, Ryan Cotterell, Naoaki Okazaki, and Desmond Elliott. 2021. Multimodal Pretraining Unmasked: A Meta-Analysis and a Unified Framework of Vision-and-Language BERTs. *Transactions of the Association for Computational Linguistics* 9 (Sept. 2021), 978–994. https://doi.org/10.1162/tacl_a_00408

[11] Qingqing Cao, Niranjan Balasubramanian, and Aruna Balasubramanian. 2017. MobiRNN: Efficient recurrent neural network execution on mobile GPU. In *Proceedings of the 1st International Workshop on Deep Learning for Mobile Systems and Applications*. 1–6.

[12] Qingqing Cao, Alexandru E Irimiea, Mohamed Abdelfattah, Aruna Balasubramanian, and Nicholas D Lane. 2021. Are Mobile DNN Accelerators Accelerating DNNs?. In *Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning*. 7–12.

[13] Qingqing Cao, Noah Weber, Niranjan Balasubramanian, and Aruna Balasubramanian. 2019. DeQA: On-Device Question Answering. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '19)*. Association for Computing Machinery, Seoul, Republic of Korea, 27–40. https://doi.org/10.1145/3307334.3326071

[14] 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 *Computer Vision – ECCV 2020 (Lecture Notes in Computer Science)*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). Springer International Publishing, Cham, 104–120. https://doi.org/10.1007/978-3-030-58577-8_7

[15] Tai-Yin Chiu, Yinan Zhao, and Danna Gurari. 2020. Assessing Image Quality Issues for Real-World Problems. 3646–3656. https://openaccess.thecvf.com/content_CVPR_2020/html/Chiu_Assessing_Image_Quality_Issues_for_Real-World_Problems_CVPR_2020_paper.html

[16] Jaemin Cho, Jiasen Lu, Dustin Schwenk, Hannaneh Hajishirzi, and Aniruddha Kembhavi. 2020. X-LXMERT: Paint, Caption and Answer Questions with Multi-Modal Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 8785–8805. https://doi.org/10.18653/v1/2020.emnlp-main.707

[17] Yunbin Deng. 2019. Deep learning on mobile devices: a review. In *Mobile Multimedia/Image Processing, Security, and Applications 2019*, Vol. 10993. International Society for Optics and Photonics, 109930A.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[19] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. https://openreview.net/forum?id=YicbFdNTTy

[20] William Falcon and The PyTorch Lightning team. 2019. *PyTorch Lightning*. https://doi.org/10.5281/zenodo.3828935

[21] Zhiyuan Fang, Jianfeng Wang, Xiaowei Hu, Lijuan Wang, Yezhou Yang, and Zicheng Liu. 2021. Compressing Visual-Linguistic Model via Knowledge Distillation. 1428–1438. https://openaccess.thecvf.com/content/ICCV2021/html/Fang_Compressing_Visual-Linguistic_Model_via_Knowledge_Distillation_ICCV_2021_paper.html

[22] Stella Frank, Emanuele Bugliarello, and Desmond Elliott. 2021. Vision-and-Language or Vision-for-Language? On Cross-Modal Influence in Multimodal Transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 9847–9857. https://doi.org/10.18653/v1/2021.emnlp-main.775

[23] Saurabh Goyal, Anamitra Roy Choudhury, Saurabh Raje, Venkatesan Chakaravarthy, Yogish Sabharwal, and Ashish Verma. 2020. PoWER-BERT: Accelerating BERT Inference via Progressive Word-vector Elimination. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 3690–3699. https://proceedings.mlr.press/v119/goyal20a.html ISSN: 2640-3498.

[24] 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. 6904–6913. https://openaccess.thecvf.com/content_cvpr_2017/html/Goyal_Making_the_v_CVPR_2017_paper.html

[25] Danna Gurari, Qing Li, Chi Lin, Yinan Zhao, Anhong Guo, Abigale Stangl, and Jeffrey P. Bigham. 2019. VizWiz-Priv: A Dataset for Recognizing the Presence and Purpose of Private Visual Information in Images Taken by Blind People. 939–948. https://openaccess.thecvf.com/content_CVPR_2019/html/Gurari_VizWiz-Priv_A_Dataset_for_Recognizing_the_Presence_and_Purpose_of_CVPR_2019_paper.html

[26] Danna Gurari, Qing Li, Abigale J. Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P. Bigham. 2018. VizWiz Grand Challenge: Answering Visual Questions From Blind People. 3608–3617. https://openaccess.thecvf.com/content_cvpr_2018/html/Gurari_VizWiz_Grand_Challenge_CVPR_2018_paper.html

[27] S. Han, H. Mao, and W. J. Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *ArXiv e-prints* (Oct. 2015). arXiv: 1510.00149 [cs.CV] tex.adsnote: Provided by the SAO/NASA Astrophysics Data System tex.adsurl: http://adsabs.harvard.edu/abs/2015arXiv151000149H.

[28] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Weinberger. 2018. Multi-Scale Dense Networks for Resource Efficient Image Classification. https://openreview.net/forum?id=Hk2aImxAb

[29] Loc N Huynh, Youngki Lee, and Rajesh Krishna Balan. 2017. DeepMon: Mobile GPU-based Deep Learning Framework for Continuous Vision Applications. In *Proc. of the 15th Annual International Conf. on Mobile Systems, Applications, and Services (MobiSys)*. ACM, 82–95.

[30] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2011. Product quantization for nearest neighbor search. *IEEE trans. on pattern analysis and machine intelligence* 33, 1 (2011), 117–128.

[31] Huaizu Jiang, Ishan Misra, Marcus Rohrbach, Erik Learned-Miller, and Xinlei Chen. 2020. In defense of grid features for visual question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10267–10276.

[32] Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. 2019. Shallow-Deep Networks: Understanding and Mitigating Network Overthinking. *arXiv:1810.07052 [cs, stat]* (May 2019). http://arxiv.org/abs/1810.07052 arXiv: 1810.07052.

[33] Sehoon Kim, Sheng Shen, David Thorsley, Amir Gholami, Joseph Hassoun, and Kurt Keutzer. 2021. Learned Token Pruning for Transformers. *arXiv:2107.00910 [cs]* (July 2021). http://arxiv.org/abs/2107.00910 arXiv: 2107.00910.

[34] Wonjae Kim, Bokyung Son, and Ildoo Kim. 2021. Vilt: Vision-and-language transformer without convolution or region supervision. In *International Conference on Machine Learning*. PMLR, 5583–5594.

[35] Wonjae Kim, Bokyung Son, and Ildoo Kim. 2021. ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision. In *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 5583–5594. https://proceedings.mlr.press/v139/kim21k.html ISSN: 2640-3498.

[36] Nicholas D Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, Lei Jiao, Lorena Qendro, and Fahim Kawsar. 2016. Deepx: A software accelerator for low-power deep learning inference on mobile devices. In *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE, 1–12.

[37] Nicholas D Lane, Petko Georgiev, and Lorena Qendro. 2015. DeepEar: robust smartphone audio sensing in unconstrained acoustic environments using deep learning. In *Proc. of the 2015 ACM International Joint Conf. on Pervasive and Ubiquitous Computing*. ACM, 283–294.

[38] 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 (MobiCom '20)*. Association for Computing Machinery, New York, NY, USA, 1–15. https://doi.org/10.1145/3372224.3419194

[39] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (Nov. 1998), 2278–2324. https://doi.org/10.1109/5.726791 Conference Name: Proceedings of the IEEE.

[40] Royson Lee, Stylianos I. Venieris, Lukasz Dudziak, Sourav Bhattacharya, and Nicholas D. Lane. 2019. MobiSR: Efficient On-Device Super-Resolution through Heterogeneous Mobile Processors. In *The 25th Annual International Conference on Mobile Computing and*

Networking (MobiCom '19). Association for Computing Machinery, New York, NY, USA, 1–16. https://doi.org/10.1145/3300061.3345455

[41] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. 2019. VisualBERT: A Simple and Performant Baseline for Vision and Language. (Aug. 2019). https://arxiv.org/abs/1908.03557v1

[42] Tianxing Li, Jin Huang, Erik Risinger, and Deepak Ganesan. 2021. Low-latency speculative inference on distributed multi-modal data streams. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '21)*. Association for Computing Machinery, New York, NY, USA, 67–80. https://doi.org/10.1145/3458864.3467884

[43] 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 (Lecture Notes in Computer Science)*, David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars (Eds.). Springer International Publishing, Cham, 740–755. https://doi.org/10.1007/978-3-319-10602-1_48

[44] Sicong Liu, Yingyan Lin, Zimu Zhou, Kaiming Nan, Hui Liu, and Junzhao Du. 2018. On-Demand Deep Model Compression for Mobile Devices: A Usage-Driven Model Selection Framework. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '18)*. Association for Computing Machinery, Munich, Germany, 389–400. https://doi.org/10.1145/3210240.3210337

[45] Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. 2020. FastBERT: a Self-distilling BERT with Adaptive Inference Time. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 6035–6044. https://doi.org/10.18653/v1/2020.acl-main.537

[46] Yoshitomo Matsubara, Marco Levorato, and Francesco Restuccia. 2021. Split Computing and Early Exiting for Deep Learning Applications: Survey and Research Challenges. *arXiv:2103.04505 [cs, eess]* (March 2021). http://arxiv.org/abs/2103.04505

[47] Albert Ng, Julian Lepinski, Daniel Wigdor, Steven Sanders, and Paul Dietz. 2012. Designing for low-latency direct-touch input. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. 453–464.

[48] PyTorch. 2018. PyTorch. https://pytorch.org/.

[49] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2016. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv:1506.01497 [cs]* (Jan. 2016). http://arxiv.org/abs/1506.01497

[50] Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A. Smith. 2020. The Right Tool for the Job: Matching Model and Instance Complexities. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 6640–6651. https://doi.org/10.18653/v1/2020.acl-main.593

[51] Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices. *arXiv:2004.02984 [cs]* (April 2020). http://arxiv.org/abs/2004.02984

[52] Hao Tan and Mohit Bansal. 2019. LXMERT: Learning Cross-Modality Encoder Representations from Transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 5100–5111. https://doi.org/10.18653/v1/D19-1514

[53] Surat Teerapittayanon, Bradley McDanel, and H.T. Kung. 2016. BranchyNet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*. 2464–2469. https://doi.org/10.1109/ICPR.2016.7900006

[54] Nvidia TX2. 2018. Nvidia TX2. (2018). https://devblogs.nvidia.com/jetson-tx2-delivers-twice-intelligence-edge/

[55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[56] Hanrui Wang, Zhekai Zhang, and Song Han. 2021. SpAtten: Efficient Sparse Attention Architecture with Cascade Token and Head Pruning. *arXiv:2012.09852 [cs]* (Jan. 2021). http://arxiv.org/abs/2012.09852 arXiv: 2012.09852.

[57] Jianfeng Wang, Xiaowei Hu, Pengchuan Zhang, Xiujun Li, Lijuan Wang, Lei Zhang, Jianfeng Gao, and Zicheng Liu. 2020. MiniVLM: A Smaller and Faster Vision-Language Model. *arXiv:2012.06946 [cs]* (Dec. 2020). http://arxiv.org/abs/2012.06946 arXiv: 2012.06946.

[58] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng. 2016. Quantized Convolutional Neural Networks for Mobile Devices. In *2016 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 4820–4828. https://doi.org/10.1109/CVPR.2016.521

[59] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. 2016. Quantized convolutional neural networks for mobile devices. In *Proc. of the IEEE Conf on Computer Vision and Pattern Recognition*. 4820–4828. https://doi.org/10.1109/CVPR.2016.521

[60] Ji Xin, Rodrigo Nogueira, Yaoliang Yu, and Jimmy Lin. 2020. Early Exiting BERT for Efficient Document Ranking. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*. Association for Computational Linguistics, Online, 83–88. https://doi.org/10.18653/v1/2020.sustainlp-1.11

[61] Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. DeeBERT: Dynamic Early Exiting for Accelerating BERT Inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 2246–2251. https://doi.org/10.18653/v1/2020.acl-main.204

[62] Jian Xu, Syed Masum Billah, Roy Shilkrot, and Aruna Balasubramanian. 2019. DarkReader: Bridging the Gap Between Perception and Reality of Power Consumption in Smartphones for Blind Users. In *The 21st International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '19)*. Association for Computing Machinery, New York, NY, USA, 96–104. https://doi.org/10.1145/3308561.3353806

[63] Mengwei Xu, Mengze Zhu, Yunxin Liu, Felix Xiaozhu Lin, and Xuanzhe Liu. 2018. DeepCache: Principled Cache for Mobile Deep Vision. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom '18)*. Association for Computing Machinery, New York, NY, USA, 129–144. https://doi.org/10.1145/3241539.3241563

[64] Chun-Ju Yang, Kristen Grauman, and Danna Gurari. 2018. Visual Question Answer Diversity. In *Sixth AAAI Conference on Human Computation and Crowdsourcing*. https://www.aaai.org/ocs/index.php/HCOMP/HCOMP18/paper/view/17936

[65] Tien-Ju Yang, Yu-Hsin Chen, and Vivienne Sze. 2017. Designing Energy-Efficient Convolutional Neural Networks Using Energy-Aware Pruning. *2017 IEEE Conf on Computer Vision and Pattern Recognition (CVPR)* (Jul 2017). https://doi.org/10.1109/cvpr.2017.643

[66] 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. 5579–5588. https://openaccess.thecvf.com/content/CVPR2021/html/Zhang_VinVL_Revisiting_Visual_Representations_in_Vision-Language_Models_CVPR_2021_paper.html

[67] Wuyang Zhang, Zhezhi He, Luyang Liu, Zhenhua Jia, Yunxin Liu, Marco Gruteser, Dipankar Raychaudhuri, and Yanyong Zhang. 2021. Elf: accelerate high-resolution mobile deep vision with content-aware parallel offloading. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking (MobiCom '21)*. Association for Computing Machinery, New York, NY, USA, 201–214. https://doi.org/10.1145/3447993.3448628

[68] Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. BERT loses patience: Fast and robust inference with early exit. In *Advances in neural information processing systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 18330–18341. https://proceedings.neurips.cc/paper/2020/file/d4dd111a4fd973394238aca5c05bebe3-Paper.pdf