Towards Efficient and Elastic Visual Question Answering with Doubly Slimmable Transformer

Zhou Yu, Member, IEEE, Zitian Jin, Jun Yu, Member, IEEE, Mingliang Xu, Jianping Fan

Abstract—Transformer-based approaches have shown great success in visual question answering (VQA). However, they usually require deep and wide models to guarantee good performance, making it difficult to deploy on capacity-restricted platforms. It is a challenging yet valuable task to design an elastic VQA model that supports adaptive pruning at runtime to meet the efficiency constraints of diverse platforms. In this paper, we present the Doubly Slimmable Transformer (DST), a general framework that can be seamlessly integrated into arbitrary Transformer-based VQA models to train one single model once and obtain various slimmed submodels of different widths and depths. Taking two typical Transformer-based VQA approaches, i.e., MCAN [1] and UNITER [2], as the reference models, the obtained slimmable MCAN_{DST} and UNITER_{DST} models outperform the state-of-the-art methods trained independently on two benchmark datasets. In particular, one slimmed MCAN_{DST} submodel achieves a comparable accuracy on VQA-v2, while being \(0.38 \times\) smaller in model size and having \(0.27 \times\) fewer FLOPs than the reference MCAN model. The smallest MCAN_{DST} submodel has 9M parameters and 0.16G FLOPs in the inference stage, making it possible to be deployed on edge devices.

Index Terms—Visual question answering, slimmable network, transformer, multimodal learning, efficient deep learning.

I. INTRODUCTION

Recent advances in deep neural networks enable the machine to tackle complex multimodal tasks that necessitate a fine-grained understanding of both vision and language cues, such as image-text matching [3], visual captioning [4], visual grounding [5], and visual question answering (VQA) [1]. Compared to other multimodal tasks, VQA is a more challenging task as it needs reasoning over the multimodal data to predict an accurate answer.

Current state-of-the-art VQA approaches can be roughly categorized into two lines of research based on whether they are trained from scratch (e.g., MCAN [1] and MUAN [6] in Fig. 1(a) or pre-trained with external multimodal data (e.g., LXMERT [7], ViLBERT [8], and UNITER [2] in Fig. 1(b)).

This work was supported in part by the National Key R&D Program of China under Grant No.2020YFB1406701, in part by the Zhejiang Provincial Natural Science Foundation of China under Grant No.LR22F020001, and in part by the National Natural Science Foundation of China under Grant No.62072147 and Grant No.61836002. (Corresponding author: Jun Yu.) Z. Jin and J. Yu are with Key Laboratory of Complex Systems Modeling and Simulation, School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, 310018, China (e-mail: yujun@hdu.edu.cn; yuz@hdu.edu.cn).
Z. Jin is with HDU-ITMO Joint Institute, Hangzhou Dianzi University, 310018, China (e-mail: jinzj@hdu.edu.cn).
M. Xu is with School of Computer and Artificial IntelligenceZhengzhou University, 450001, China (e-mail: iexumingliang@zzu.edu.cn).
J. Fan is with AI Lab at Lenovo Research, 100094, China (e-mail: jfan1@Lenovo.com).
elastica CNN model with multiple width multipliers to fit different efficiency constraints at runtime [17], we present a Doubly Slimmable Transformer (DST) framework to support model slimming in both the width and depth directions of the Transformer. The DST framework is general enough to be seamlessly integrated with arbitrary Transformer-based VQA models to support training a single model once and then obtaining multiple efficient submodels of various widths and depths at runtime. We take two typical Transformer-based VQA approaches, i.e., MCAN [1] and UNITER [2], as the reference models to incorporate into the DST framework, resulting in the slimmable MCAN_{DST} and UNITER_{DST} models, respectively. As shown in Fig. 1, MCAN_{DST} and UNITER_{DST} either outperform existing state-of-the-art VQA models with similar model sizes or achieve comparable performance with much smaller models.

To our knowledge, our study is the first attempt to explore efficient and elastic models for VQA. The most closely related study to our work is the DynaBERT approach [18], which also investigates slimmable Transformer architectures. In contrast to our DST framework, which supports arbitrary Transformer-based architectures for VQA, DynaBERT focuses on the pretrained BERT model for NLP tasks. In terms of methodology, our DST is different from DynaBERT in terms of slimming strategy and training algorithm, resulting in a higher compression ratio and less training time. More details will be described in the following sections.

Our main contributions can be summarized as follows:

- We suggest a new direction for VQA to learn an efficient-and-elastic model once and obtain diverse efficient submodels that can be fit different platforms adaptively.
- We present a general Doubly Slimmable Transformer (DST) framework that can be seamlessly integrated with any Transformer-based VQA model. DST performs model slimming in both the width and depth dimensions and obtains a set of submodels of different widths and depths.
- We elaborately investigate the effectiveness of the submodels and introduce a simple triangle-based submodel selection strategy to filter out ineffective submodels before training. This selection strategy can significantly reduce the training costs while improving the performance of the remaining submodels.
- We integrate the DST framework with two typical Transformer-based VQA approaches, MCAN [11] and UNITER [2]. The obtained MCAN_{DST} and UNITER_{DST} models outperform existing state-of-the-art models with similar model sizes or achieve comparable performance with much smaller models, highlighting the efficacy, efficiency, and generality of the proposed DST framework.

II. RELATED WORK

In this section, we first briefly review previous studies on VQA, especially those approaches with Transformer architectures. After that, we discuss related work on efficient neural networks and slimmable neural networks.

Visual Question Answering (VQA): The VQA task, which aims to answer a free-form question in natural language with respect to a given image, has been of increasing interest over the last few years. The core of VQA lies in two lines of research, namely multimodal fusion and attention learning. For multimodal fusion, early methods use linear models with element-wise summation or multiplication to fuse the features from different modalities [19][20]. To better characterize the second-order interactions between multimodal features, Fukui et al. [21], Kim et al. [22], and Yu et al. [23] devise different multimodal bilinear pooling models, respectively. For attention learning, question-guided visual attention on image regions has become a standard component in many early VQA approaches. Chen et al. introduce a question-guided attention map to project the question embeddings into the visual space and obtain a dynamic convolutional kernel to focus on specific image regions [24]. Yang et al. propose a stacked attention network to iteratively learn visual attention on different levels [25]. More recently, co-attention models that consider both textual and visual attention have been proposed. Lu et al. introduce a hierarchical co-attention learning paradigm to learn image attention and question attention iteratively [26]. Yu et al. decouple the co-attention learning into a question self-attention stage and a question-conditioned visual attention stage and optimize the two stages in an end-to-end manner [23]. The aforementioned co-attention models are coarse-grained in that they neglect the multimodal interactions at a fine-grained level (i.e., word-region pairs). To address this issue, Nguyen et al. [27] and Kim et al. [11] introduce dense co-attention models that establish complete interactions among word-region pairs.

Transformer-based VQA: The Transformer architecture is initially proposed for the machine translation task in the NLP community [9]. It consists of a sequence of self-attention modules to model the complex and dense interactions within a group of input features. This architecture is general enough to be used not only in various unimodal tasks [28][29][30] but also various multimodal tasks like image captioning and VQA. Yu et al. devise a modular co-attention network (MCAN) for VQA based on the Transformer architecture and deliver new state-of-the-art performance on commonly-used datasets [11]. More recently, the BERT model, which integrates the Transformer architecture with a self-supervised pretraining paradigm, has shown great success in a wide range of NLP tasks [28]. Mirroring the success of BERT, recent studies naturally extend its framework to the multimodal domain to pretrain multimodal-BERT (a.k.a., vision-language pretraining) models [26][7][2][13]. In particular, they first pretrain Transformer-based models on large image-text corpora to learn task-agnostic representations, and then finetune the models on downstream tasks like VQA. To summarize, Transformer-based approaches have dominated the VQA task at present, due to their excellent capability for modeling the complex interactions among multimodal input features. However, Transformer-based VQA models are usually computationally expensive (i.e., of a large number of parameters and FLOPs), hindering their deployment on edge devices with limited memory and computation consumption. This motivates us to explore efficient Transformer architectures for VQA.

Efficient Neural Networks: There has been broad interest in building efficient neural networks in the literature. Existing
approaches can be generally categorized into either compressing pretrained networks [31][32][33] or training efficient networks directly [34][35][36]. The efficient neural networks above are mainly focused on ConvNet architectures. Due to the popularity of Transformer in recent years, efficient Transformer architectures have been investigated from different aspects, e.g., low-rank decomposition [37], weight sharing [38][39], model pruning [40][41], and knowledge distillation [42][43][44]. Despite the success of these approaches, their efficient models are dedicated to one specific scenario, and cannot adapt to different efficiency constraints or different hardware platforms at runtime.

**Slimmable Neural Networks:** Orthogonal to the approaches on efficient neural networks above, slimmable neural networks aim to design dynamic models that can adaptively fit different efficiency constraints at runtime. Given a deep neural network, network slimming can be performed on both the depth and width dimensions. For depth slimming, Wu et al. [45], Liu et al. [33], and Huang et al. [46] learn controllers or gating modules to adaptively drop layers from deep ConvNets. For width slimming, Yu et al. introduce a general framework for a family of ConvNets (e.g., ResNet [47] or MobileNet [35]) that supports a predefined set of width multipliers [17]. After that, they further improve the framework to support model slimming with arbitrary widths [48]. To take a further step, Cai et al. introduce a once-for-all (OFA) method to support width and depth slimming simultaneously in a unified framework [49]. All of the methods above are only for ConvNet architectures, and their strategies cannot be directly applied to Transformer architectures.

The most closely related study to our work is the DynaBERT [18], which also investigates slimmable Transformer architectures. In contrast to our DST framework, which supports arbitrary Transformer-based architectures for VQA, DynaBERT focuses on the pretrained BERT model for NLP tasks. On the methodology, our DST is different from DynaBERT in terms of slimming strategy and training algorithm, obtaining significant advantages in terms of higher compression ratio and less training time.

### III. Doubly Slimmable Transformer (DST)

In this section, we describe the Doubly Slimmable Transformer (DST) framework in detail. Before presenting the DST framework, we first revisit the core components of the Transformer architecture [9]. Then, we introduce the DST framework, including the slimming strategies in width and depth, respectively. Finally, we take two typical VQA models, MCAN [1] and UNITER [2], as examples to integrate with the proposed DST framework. Without loss of generality, our DST framework can be applied to arbitrary VQA models of Transformer-based architectures.

#### A. Preliminaries

The Transformer is a multi-layer network with each layer consisting of the multi-head attention (MHA) and feed-forward networks (FFN) modules [9].

**Multi-Head Attention:** Denote \( m \) query features and \( n \) key-value paired features as \( Q \in \mathbb{R}^{m \times D}, K \in \mathbb{R}^{n \times D}, \) and \( V \in \mathbb{R}^{n \times D} \) respectively, where \( D \) is the hidden dimensionality of these features. The multi-head attention module calculates the attended features \( F \in \mathbb{R}^{m \times D} \) by using \( H \) paralleled attention functions as follows:

\[
F = \text{MHA}(Q,K,V) = [\text{head}_1, \text{head}_2, ..., \text{head}_H]W^o
\]

\[
\text{head}_j = \text{ATT}(QW^j,Q,KW^j,VW^j)
\]

where \( W^Q, W^K, W^V \in \mathbb{R}^{D \times D_H} \) are the projection matrices for the \( j \)-th head, and \( D_H \) is the dimensionality of the features from each head. \( W^o \in \mathbb{R}^{H \times D_H \times D} \) is the projection matrix to aggregate the output features from different heads. We have \( D_H = D/H \) so that the model sizes remain constant when \( H \) varies. The attention function for each head is defined as the scaled dot-products of the query with all keys:

\[
\text{ATT}(Q,K,V) = \text{softmax}(\frac{QK^T}{\sqrt{D_H}})V
\]

which calculates the scaled dot-products of each query with all keys to obtain the attention weights, and then performs weighted summation over the values.

**Feed-Forward Network:** The feed-forward network module is a two-layer MLP model applied to the output features of the MHA module to perform a point-wise nonlinear transformation. Given input features \( X \in \mathbb{R}^{n \times D}, \) the transformed features \( F \in \mathbb{R}^{n \times D} \) are obtained as follows:

\[
F = \text{FFN}(X) = \text{ReLU}(XW_1 + b_1)W_2^T + b_2
\]

where the \( W_1, W_2 \in \mathbb{R}^{D \times 4D} \).

**Transformer Layer:** A typical Transformer layer usually consists of a MHA module and a FFN module as follows:

\[
F = \text{Transformer-layer}(X)
\]

\[
= \text{LN}(\text{FFN}(\tilde{F}) + F)
\]

\[
\tilde{F} = \text{LN}(\text{MHA}(X,X,X) + X)
\]

where residual connection [47] and layer normalization (LN) [50] are applied after the MHA and FFN modules. The LN module takes a \( D \)-dimensional feature \( x \in \mathbb{R}^{D} \) as its input and performs normalization as follows to obtain the output feature:

\[
y = \text{LN}(x) = \gamma \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta
\]

where \( \mu, \sigma^2 \in \mathbb{R} \) are the mean and variance calculated on \( x \). \( \gamma, \beta \in \mathbb{R}^D \) are the learnable weights of the scale and shift terms, respectively.

**Transformer Architectures:** Depending on different composition strategies of Transformer layers above, existing Transformer architectures can be categorized into three classes, namely the encoder [28][51], the decoder [52][53], and the encoder-decoder [9][54].

Taking a sequence of input tokens, the original Transformer [9] adopts an encoder-decoder architecture. The encoder is composed of a cascade of Transformer layers in depth to obtain the bidirectional representations by jointly conditioning on both the left and right contexts, and the decoder takes the representations from the last encoder layer as input to guide the
learning of unidirectional representations by only conditioning on the left context. After that, pure encoder architectures (e.g., BERT [23]) and pure decoder architectures (e.g., GPT [52]) are introduced to integrate with the self-supervised pretraining paradigm, which has become the de-facto standard in the NLP community.

B. The DST Framework

Let an $L$-layer Transformer with hidden dimensionality $D$ be the reference model, where $D$ and $L$ denote the width and depth of the model, respectively. The goal of DST is to obtain one slimmable Transformer model that can adaptively adjust to a set of submodels of different widths and depths in the inference stage. In the following, we introduce the width slimming and depth slimming strategies. An overview of the DST framework is illustrated in Fig. 2a.

Width Slimming: By width slimming, we aim to make each Transformer layer adapt to a set of width slimming ratios with respect to the hidden dimensionality $d$ of the reference model. To achieve this goal, we split the weights of the reference model into different submodels, with each sharing a specific portion of its model weights. As shown in Eq. (1), (3), and (5), the learnable weights in the MHA and FFN modules are all adjusted in a similar manner. The inference stage. In the following, we introduce the width slimming and depth slimming strategies. An overview of the DST framework is illustrated in Fig. 2a.

To perform depth slimming, we first need to determine which layers are to be slimmed when given a specific slimming depth $l \in L$. As shown on the right of Fig. 2a, we first assign an importance score to each layer using some scoring strategies. After that, we select the top-$l$ layers with the largest scores and keep their original order. Here we introduce three scoring strategies, which result in three types of slimming strategies: 1) the slim-random strategy is the most straightforward one that simply sets the importance scores to random values; 2) the slim-first (or slim-last) strategy sets the importance scores in ascending (or descending) order; and 3) the slim-middle strategy sets the smallest scores to the middlemost layer and gradually increases towards the top and bottom layers. This strategy is inspired by the empirical studies in [55] that the layers closer to the input and output are more important than the middle ones in Transformer. We use the slim-middle strategy as the default option.

C. Integrating DST with Off-the-shelf VQA Models

DST is a general framework that can be integrated with arbitrary Transformer-based VQA models in theory. In this paper, we choose two typical Transformer-based models, i.e., MCAN [1] and UNITER [2] shown in Fig. 2b to integrate with the proposed DST framework. Without loss of generality, the DST framework can also be applied to other Transformer-based models beyond the VQA task. Due to space limitations, we will not expand the description further.

MCAN_{DST}: MCAN is the winning solution in the VQA Challenge 2019, which introduces an encoder-decoder-based Transformer architecture to model complex multimodal interactions and perform accurate visual reasoning. Specifically, the input question and image are first represented as a sequence of word embeddings and a group of object embeddings, respectively. After that, the multimodal embeddings are passed through an $L$-layer encoder-decoder to obtain the attended output features. In the $L$-layer question encoder, the word embeddings are transformed with a self-attention mechanism to obtain the attended question features of the same word length. The attended question features, along with the object embeddings, are further fed into an $L$-layer image decoder to obtain the attended image features with a guided-attention mechanism. On top of the attended question features and image features, two attentional reduction modules are devised to obtain a question feature and an image feature, respectively. Finally, the two feature vectors are simply fused and then fed to a linear classifier to predict the answer. The MCAN model can be trained from scratch in an end-to-end manner on a specific VQA dataset like VQA-v2 [56].

Since MCAN’s core components are the standard Transformer layers, the model can be seamlessly integrated with the DST framework to obtain a slimmable MCAN_{DST} model. The width slimming strategy can be directly applied to each encoder and decoder layer in MCAN, and different depth slimming strategies can also be applied to drop a portion of the encoder and decoder layers simultaneously. Furthermore, the model weights in the attention reduction module on top of the encoder-decoder are derived from two-layer MLPs, which
can be slimmed in width using a similar strategy to the FFN module. The weights in the embedder and classifier are not involved in any slimming process, as the dimensionality of the input and output features remains the same.

**UNITER**

UNITER is a representative vision-and-language pretraining (VLP) approach with an $L$-layer Transformer encoder as its backbone. In contrast to MCAN’s `training-from-scratch` mechanism, UNITER utilizes a `pretraining` strategy to learn a generalized backbone model from massive image-text pairs, and then finetunes the backbone to adapt to different multimodal tasks. Specifically for the VQA task, a task-specific head is appended on top of the backbone so that the representation of the predefined `[CLS]` token is fed to a linear classifier to predict the answer. Based on the finetuned UNITER model for VQA, both width slimming and depth slimming are applied to its backbone to further transform it into **UNITER**. Similar to **MCAN**

Note that the embedding layers in **MCAN** and **UNITER** are not slimmable, making the input dimensionality of the first Transformer layer unadjustable. This contradicts our width slimming strategy. To address this issue, we inject a linear projection layer $W_{emb} \in \mathbb{R}^{D\times D}$ between the embedder and backbone, and make it slimmable in width to adapt the DST framework.

### Submodel Complexity Analysis

Given a reference MCAN (or UNITER) model of width $D$ and depth $L$, its model size and FLOPs are both proportional to $O(D^2L)$ approximately. This indicates the computational cost of the smallest submodel $a(1/4D, 1/6L)$ is up to $96 \times$ smaller than the reference model. In practice, the scaling ratio between the submodels and the reference model is not that large. As shown in Fig. 2(a), the slimming strategies are only performed in the backbone while the embedders and the classifier are not involved. Their existence, especially the question embedder, introduces an inescapable cost to all the slimmed submodels, limiting the computational overhead of the small submodels. More detailed results are shown and analyzed in the experimental section.

### IV. Training Algorithm for DST Models

In this section, we introduce the training algorithm for DST in detail, which consists of a submodel architecture selection stage and a self-distillation training stage.

**Submodel Architecture Selection:** By combining each width in $D$ with each depth in $L$, we obtain a set of submodel architectures $A$ of different widths and depths as follows:

$$A = \text{combination}(D, L)$$

where $|A| = |D| \times |L|$. Each architecture $a(d, l) \in A$ corresponds to a combination of a specific width $d \in D$ from and depth $l \in L$. In contrast to previous works that maintain all possible submodel architectures, we hypothesize that not every submodel architecture is effective. By effectiveness of a submodel architecture, we mean its computational cost (e.g., in terms of FLOPs or model size) matches its delivered performance after model training. Devising a heuristic strategy to get rid of such ineffective architectures before DST training can reduce the training costs while improving the performance of the remaining submodels.

According to previous studies on designing efficient Transformers [58][18], `deep-and-narrow` architectures usually deliver better performance than `shallow-and-wide` ones under constrained computational costs. This principle can be explained from two aspects: 1) Transformer requires a relatively deep model to guarantee good performance; and 2) the computational cost of a Transformer model is proportional to $O(LD^2)$, suggesting increasing depth is more economical than width.

To quantize this deep-and-narrow principle, we introduce a simple `triangle selection strategy` to filter out the shallow-and-wide submodel architectures. As shown in Fig. 3, we introduce a 2-D indicator matrix $I \in \{0, 1\}^{[D] \times [L]}$ to track the selection status for all the submodel architectures $A$. $I(d, l) = 1$ indicates the submodel architecture $a(d, l)$ is selected, and 0 otherwise. The indicator matrix $I$ is first initialized with all-one values and then converted to an `upper-triangle` matrix.

---

[1] For the **UNITER** model with pretrained weights, $W_{emb}$ is initialized with an identity matrix and updated along with the entire model in an end-to-end manner.
Fig. 3: The diagram of submodel architecture selection by using the triangle selection filtering strategy. The selected submodel architectures (highlighted with yellow background) follow the deep-and-narrow principle, which is considered to be more effective than the rest (white background).

This simple strategy allows us to filter out six shallow-and-wide submodel architectures, which correspond to the matrix elements above the main diagonal. Formally, the selected submodel architectures $S$ from $\mathcal{A}$ are defined as follows:

$$S = \text{upper-triangle}(\mathcal{A}) = \{a(d, l) | f(d, l) = 1\} \quad (7)$$

Self-distillation Training: After obtaining the selected submodel architectures $S$, we train a slimmable model $M_{\text{DST}}$ that can adapt to any submodel architecture $a \in S$ elastically, where $M$ can be either a MCAN or an UNITER model for VQA. Given the DST model $M_{\text{DST}}$ and a submodel architecture $a$, the obtained submodel is denoted as $M_{\text{DST}}^{(a)}$.

To obtain the DST model, we introduce a self-distillation training mechanism as follows. In general, we first train an ordinary model $M_{\text{teacher}}$ as the teacher model without network slimming. After that, the DST model $M_{\text{DST}}$ is initialized with the model weights from $M_{\text{teacher}}$, which can be viewed as the student model. Each slimmable submodel $M_{\text{DST}}^{(a)}$ shares a specific portion of the model weights from $M_{\text{DST}}$ and is trained with the supervision of the teacher model using the knowledge distillation (KD) strategy [59]. By self-distillation, we mean the teacher model $M_{\text{teacher}}$ and the student model $M_{\text{DST}}^{(a)}$ share the same model architecture.

Different from the existing approaches [18] that decouple the width slimming and depth slimming into two training stages, we use a simpler one-stage training paradigm for DST with standard mini-batch SGD. In each iteration, we pick $k$ submodel architectures from $S$ and feed the same input samples to both the teacher model and the selected $k$ submodels to obtain $(k + 1)$ predictions in total. After that, we apply the KD loss between the predictions of the teacher and each of the $k$ student models and use the accumulated back-propagated gradients to update the model weights of the $k$ selected submodels. To stabilize the DST training, the $k$ selected submodels consist of two determined submodels (i.e., the smallest one and the largest one) and another $(k - 2)$ randomly picked submodels. In our experiments, we use $k=4$.

The detailed training algorithm for DST is illustrated in Algorithm 1.

Algorithm 1: Training algorithm for DST.

**Input:** A reference Transformer architecture with width $D$ and depth $L$. Two predefined sets $\mathcal{D}$ and $\mathcal{L}$ define the slimming widths and depths w.r.t. $D$ and $L$, resp.

The number of sampled submodel architectures $k$ per training iteration. $a_s$ and $a_l$ refer to the smallest and largest submodel architectures, resp.

**Output:** An optimized DST model $M_{\text{DST}}$.

**Stage I: Submodel Architectures Selection;**

$s = \text{combination}(\mathcal{D}, \mathcal{L})$;

**Stage II: Self-distillation Training;**

Train $M_{\text{teacher}}$ to obtain its optimized model weights $\theta$;

Initialize the model weights $\theta_{\text{DST}}$ for $M_{\text{DST}}$ with $\theta$;

for $i = 1$ to max-iter do

Randomly sample a mini-batch of data $x$;

Initialize the sampled architecture set $\Omega = \emptyset$;

# add another $k - 2$ architectures via random sampling.

for $j = 1$ to $k - 2$ do

Randomly sample a submodel architecture $a \sim S_{\mathcal{A}}$;

$\Omega \leftarrow \Omega \cup \{a, a\}$;

# add another $k - 2$ architectures via random sampling.

end

end

# submodel training using knowledge distillation.

Feed-forward the teacher model: $y = M_{\text{teacher}}(x)$;

Freeze $M_{\text{teacher}}$ by stopping gradients: $y$.detach();

foreach $a \in \Omega$ do

Feed-forward the submodel: $\hat{y} = M_{\text{DST}}^{(a)}(x)$;

Compute loss: $\text{loss} = \text{KD}(y, \hat{y})$;

Accumulate backward gradients: $\text{loss.backward}()$;

end

Update model weights $\theta_{\text{DST}}$.

V. EXPERIMENTAL RESULTS

In this section, we conduct experiments to evaluate the performance of our DST framework on two benchmark VQA datasets, namely VQA-v2 [20] and GQA [60]. As mentioned above, we integrate DST with two typical Transformer-based VQA models, namely MCAN [1] and UNITER [2], to demonstrate the effectiveness and universality of the proposed DST framework. Furthermore, we conduct ablation experiments on VQA-v2 to explore the effects of different components in DST.

A. Datasets

VQA-v2 is the most commonly-used VQA dataset [56]. It contains human-annotated QA pairs for MS-COCO images [61]. The dataset is split into three sets: train (80k images with 444k questions); val (40k images with 214k questions); and test (80k images with 444k questions). The test set is further split into test-dev and test-std sets. The reported results include three per-type accuracies (yes/no, number, and other), as well as an overall accuracy.

To make a fair comparison among the compared models, we follow the dataset splitting strategy in UNITER that further splits the val set into a minival subset of 5k images and a trainval subset of the remaining 35k images [2]. All the reported results in the experiments are trained on the augmented train+trainval+vg sets, where vg denotes the augmented VQA samples from Visual Genome [62]. The
TABLE I: Comparison to the state-of-the-art approaches on VQA-v2. All methods use the same bottom-up attention visual features [4]. For a fair comparison, the compared methods are split into two groups depending on whether they are trained from scratch or pretrained on external data (separated by a double-line). The number of parameters is calculated from an entire model, including the embedders, backbone, and classifier. The number of FLOPs is calculated from one single sample.

| model | #params | FLOPs | All | Y/N | Num | Other | All | Y/N | Num | Other |
|-------|---------|-------|-----|-----|-----|-------|-----|-----|-----|-------|
| UpDn  | 22M     | 1.1G  | 65.32 | 81.82 | 44.21 | 56.05 | 65.67 | 82.20 | 43.90 | 56.26 |
| MFB   | 68M     | 2.4G  | 68.40 | 84.78 | 49.05 | 58.82 | -    | -    | -    | -     |
| MFH   | 102M    | 2.5G  | 68.76 | 84.27 | 49.56 | 59.89 | -    | -    | -    | -     |
| BAN   | 112M    | 12.3G | 69.66 | 85.46 | 50.66 | 60.50 | -    | -    | -    | -     |
| MUCAN | 83M     | 17.3G | 70.82 | 86.77 | 54.40 | 60.89 | 71.10 | -    | -    | -     |
| MCAN(D=512,L=6) | 58M | 5.5G | 70.63 | 86.82 | 53.26 | 60.72 | 70.90 | - | - | - |
| MCAN(D,L) | 58M | 5.5G | 71.05 | 87.39 | 52.96 | 61.19 | 71.28 | 87.36 | 52.77 | 61.52 |
| MCAN(1/2D,L) | 22M | 1.5G | 70.45 | 86.84 | 52.89 | 60.43 | - | - | - | - |
| MCAN(1/2D,1/3L) | 14M | 0.6G | 69.42 | 85.68 | 51.96 | 59.48 | - | - | - | - |
| MCAN(1/4D,1/3L) | 10M | 0.2G | 68.16 | 84.84 | 50.27 | 57.95 | - | - | - | - |
| MCAN(DST) , D,L) | 11M | 17.8G | 70.55 | - | - | - | 70.92 | - | - | - |
| MCAN(DST,1/2D,L) | 116M | 20.7G | 71.16 | - | - | - | 72.54 | 88.20 | 54.20 | 63.10 |
| MCAN(DST,1/2D,1/3L) | 116M | 38.6G | 73.16 | - | - | - | 73.44 | - | - | - |
| MCAN(DST,1/4D,1/3L) | 117M | 50.2G | 72.70 | 88.86 | 55.10 | 62.87 | 72.95 | 89.00 | 55.37 | 63.01 |
| UNITER(DST) , D,L) | 117M | 20.2G | 73.27 | 89.01 | 56.73 | 63.57 | 73.46 | 89.17 | 56.28 | 63.73 |
| UNITER(DST,1/2D,L) | 53M | 5.6G | 72.11 | 87.83 | 55.59 | 62.42 | - | - | - | - |
| UNITER(DST,1/2D,1/3L) | 39M | 2.2G | 70.65 | 86.47 | 53.47 | 61.02 | - | - | - | - |
| UNITER(DST,1/4D,1/3L) | 33M | 0.8G | 69.68 | 85.49 | 52.26 | 60.12 | - | - | - | - |

B. Experimental Setup

The model architectures and training hyper-parameters for the MCAN_DST and UNITER_DST models are almost the same as those in their original models, respectively [1], [2]. For MCAN_DST, the hidden dimensionality $D$, number of heads $H$, and number of layers $L$ are set to 512, 8, and 6, respectively. The MCAN_DST models are trained on the VQA-v2 and GQA datasets respectively using slightly different settings. On VQA-v2, binary cross-entropy (BCE) is used as the loss function for both the teacher and the DST model, and both models are trained up to 15 epochs with a batch size of 64 and a base learning rate of 1e-4. The learning rate is warmed-up for 3 epochs and decays by 1/5 every 2 epochs after 10 epochs. On GQA, the learning rate and batch size are the same as those on VQA-v2. KL-divergence is used as the loss function and the teacher and DST models are trained to 11 epochs. The learning rate is warmed-up for 2 epochs and decays by 1/5 every 2 epochs after 8 epochs.

For UNITER_DST, we adopt the network architecture from the UNITER-base model with $D = 768$, $H = 12$, and $L = 12$, respectively. Taking the finetuned UNITER model on VQA-v2 as the teacher, UNITER_DST is initialized from the teacher model and trained up to 130k iterations on VQA-v2 with a batch size of 5120. The AdamW optimizer is used with a base learning rate of 1.5e-4 and a weight decay of 0.01.

C. Main Results

In Tables I and II we compare MCAN_DST and UNITER_DST to the state-of-the-art VQA methods on VQA-v2 and GQA, respectively. For MCAN_DST, the compared methods include UpDn [4], MFB [23], MFH [10], BAN [11], MUAN [6], and evaluated on the test-dev and test-std sets offline.

GQA is a challenging VQA dataset that requires more complex reasoning skills [60]. It consists of 113K images and 1.2M balanced question-answer pairs of assorted types and varying composition degrees, measuring performance on an array of reasoning skills such as object and attribute recognition, spatial reasoning, logical inference, and comparisons. The dataset is split into the following four sets: train (72k images with 943k questions), val (10k images with 132k questions), test-dev (398 images with 12k questions), and undisclosed test-challenge (1.6k images with 50k questions). Following the suggestions in the official GQA guideline [4], all the results are trained on the train+val sets and evaluated on the test-dev set.

https://cs.stanford.edu/people/dorarad/gqa/evaluate.html
and MCAN [11], which are the best performing solutions for the VQA Challenge in recent years. For UNITER$_{DST}$, the compared methods include ViLBERT [8], VLBERT [12], LXMERT [7], OSCAR [13], and UNITER [2], which are the representative vision-language pretraining methods. Due to space limitations, we do not show the results of all the submodels (i.e., the selected ten submodels by the triangle selection strategy in Fig. [7] of MCAN$_{DST}$ and UNITER$_{DST}$ in these tables. Instead, four typical submodels are selected to compare with the state-of-the-art approaches.

From the results in the first group (the upper part) of Table I, we have the following observations: 1) with the same model architecture, the full-sized MCAN$_{DST}(D, L)$ model outperforms the reference MCAN model and another Transformer-based model MUAN without the DST training (Line #7 vs. Line #5 and #6). This improvement benefits from the synergistic effect of the weight-sharing submodel architectures and KD-based training strategy; 2) with only 0.38× model size and 0.27× FLOPs, MCAN$_{DST}(1/2D, L)$ is still competitive with the reference MCAN model (Line #8 vs. Line #6), showing the potential of width slimming; 3) by slimming depth to 1/3L (Line #9), its corresponding model size and FLOPs are respectively reduced to 0.6× and 0.4× of its counterpart in Line #8, at the expense of 1-point accuracy drop. Compared with MFB [23], MFH [10], and BAN [11], MCAN$_{DST}(1/2D, 1/3L)$ achieves superior or comparable performance with up to 0.125× model size and 0.05× FLOPs; and 4) MCAN$_{DST}(1/4D, 1/3L)$ still outperforms UpDn [4] by 2.1 points with an extremely small model size of 10M. This model size is close to the lower bound of MCAN, which includes 7.8M uncompresible model parameters in the embedders and classifier.

From the results in the second group (the lower part) of Table I, we obtain similar observations to those on MCAN$_{DST}$. The slimmable UNITER$_{DST}(D, L)$ outperforms its reference UNITER model (Line #15) and all the other VLP methods. Moreover, the slimmed submodels in Line #17-19 attain a significant reduction in computational costs at the expense of some accuracy drop. These consistent observations verify the effectiveness and robustness of DST over different Transformer architectures (i.e., encoder and encoder-decoder) and different training paradigms (i.e., from-scratch training and vision-language pretraining). To further examine the generalization of DST, we compare MCAN$_{DST}$ to the state-of-the-art methods on GQA. Table II shows that MCAN$_{DST}(D, L)$ is 1.2 points higher than the reference MCAN model without DST. Furthermore, with a 0.17× model size and 0.06× FLOPs, MCAN$_{DST}(1/4D, 1/3L)$ achieves comparable results to the reference MCAN model, surpassing the rest of its counterparts by a distinct margin.

Next, we show the results of all the ten submodels in terms of the DST training and standard independent training. By independent training, we mean each submodel is trained independently without sharing its model weights [4]. From the results in Fig. 4, we can see that all the ten submodels achieved by DST training deliver better performance than their counterparts obtained by independent training. This corroborates with the observations from Table I.

Finally, the submodels obtained by DST training are slimmed from one single model without retraining, outperforming the same submodels by independent training in terms of both the total model size and total training time. From the results in Fig. 5, we can see that the total model size of the ten submodels by DST training is ~25% of that by independent training. Furthermore, the total training time for DST training is ~50% of that for independent training.

For the independent training for UNITER, each submodel is first initialized with a specific portion of the model weights from the pretrained model and then finetuned independently.

---

**Table I: Accuracies of the state-of-the-art methods on GQA.** All entries use the officially provided object features for images and are evaluated on the test-dev split.

| model                  | #params | FLOPs | accuracy |
|------------------------|---------|-------|----------|
| UpDn [4]               | 30M     | 2.8G  | 51.62    |
| BAN [11]               | 120M    | 14.9G | 55.81    |
| MCAN (D=512, L=6) [11] | 59M     | 6.5G  | 56.64    |
| MCAN$_{DST}$ (D, L)    | 59M     | 6.5G  | 57.83    |
| MCAN$_{DST}(1/2D, L)$  | 22M     | 1.9G  | 57.67    |
| MCAN$_{DST}(1/2D, 1/3L)$ | 15M   | 0.8G  | 57.09    |
| MCAN$_{DST}(1/4D, 1/3L)$ | 10M   | 0.4G  | 56.38    |

---

**Fig. 4: Accuracy vs. FLOPs on the VQA-v2 test-dev split.** For each VQA model (i.e., MCAN [11] and UNITER [2]), we report the results of ten submodels obtained from DST training and independent training, respectively.

**Fig. 5: Total number of parameters (left) and total training time (right) of the 10 submodels obtained by DST training and independent training, respectively.**
TABLE III: Ablations of the MCAN\textsubscript{DST} variants models evaluated on the test-dev split of VQA-v2. The default strategies and the best results are bolded.

| slimming strategy | submodel | FLOPs | acc. |
|-------------------|----------|-------|------|
| slim-all          | (1/2D,1/2L) | 1.5G  | 67.98 |
| slim-interm. \cite{18} | (1/AD,1/DL) | 1.6G  | 67.86 |
| slim-all          | (1/2D,1/3L) | 0.6G  | 66.95 |
| slim-interm. \cite{18} | (1/4D,1/3L) | 0.8G  | 66.83 |

(a) Width Slimming: Under the same model depth and similar FLOPs, the obtained submodels trained with the slim-all strategy outperform the counterpart with the slim-intermediate strategy.

(b) Depth Slimming: The slim-middle strategy surpasses over all the counterparts in terms of average accuracy.

(c) Model Training: The teacher initialization and KD strategies show advantages over the random initialization and ID strategies in terms of average accuracy, respectively.

D. Ablation Studies

We run a number of ablations on MCAN\textsubscript{DST} to analyze the effectiveness of the key component in DST. Results are shown in Table III and Fig. 6 are discussed in detail below.

Width Slimming Strategies: In Table IIIa we show the results of the MCAN\textsubscript{DST} variants trained with different width slimming strategies, i.e., the slim-all strategy introduced in this paper and the slim-intermediate strategy introduced in \cite{18}. By comparing two submodels of similar FLOPs, our slim-all strategy delivers better model performance than the slim-intermediate strategy under different model depths.

Depth Slimming Strategies: In Table IIIb we compare the MCAN\textsubscript{DST} variants with different depth slimming strategies (mentioned in Section III-B) in terms of average accuracy over the ten submodels, respectively. From the results, we can see that the slim-middle strategy achieves the best performance among the counterparts, suggesting that the bottom and top layers of Transformer are more important than the middle ones. This observation is consistent with the results in \cite{55}.

Model Training Strategies: Our default training strategy uses the model weights from a teacher model as initialization, and then trains the submodels using a knowledge distillation (KD) strategy to exploit the implicit knowledge from the teacher model. The results in Table IIIc show that both the teacher model initialization and the KD-based model training facilitate the obtained MCAN\textsubscript{DST} model, compared to the model variants trained with random initialization or supervised by the ground-truth answer. In contrast to our KD training strategy that uses a fixed teacher model, an alternative strategy introduces a special in-place distillation (ID) training strategy that takes the largest submodel as the dynamic teacher to perform knowledge distillation \cite{48}. We note that the ID-based strategy achieves worse performance than the KD-based training strategy (65.55% vs. 66.95% in terms of average accuracy), and even underperforms standard training without knowledge distillation (65.55% vs. 65.89%). This suggests a stable teacher model plays a key role in DST training.

Submodel Selection Strategies: In Fig. 6 we compare two MCAN\textsubscript{DST} variants with 10 and 16 submodels, respectively. From the results, we have the following observations: 1) revealing the synergistic effect of different weight-sharing submodels in DST training.

E. Qualitative Analysis

In Fig. 7 we visualize the attention maps from three weight-sharing MCAN\textsubscript{DST} submodels. Due to space limitations, we only show one example and visualize attention maps from the first and last layers of the question encoder and image decoder, respectively. To better understand the effect of the attention mechanism, we highlight some representative attention maps with blue bounding boxes. From the results, we have the following observations.

In general, the slimmed submodels MCAN\textsubscript{DST}(1/4D,1/L) and MCAN\textsubscript{DST}(1/4D,1/3L) have fewer redundant heads (i.e., similar attention maps within one layer) than the full-sized one MCAN\textsubscript{DST}(D,L). This verifies the feasibility and necessity of our DST framework. Moreover, the three submodels, which have different widths and depths, all predict the correct answer. This verifies the effectiveness of both the width and depth slimming strategies, as well as the training paradigm.
Taking a closer look at the attention maps, the attention maps from different submodels share similar properties to those attention maps in the original MCAN paper [11]. Almost all the attention maps from the first layer of the question encoder (i.e., enc-1) attend to the column of words like ‘what’, which acts as the question type classifier. In contrast, some attention maps from the last layer of the question encoder (i.e., enc-6) and image decoder (i.e., dec-1 and dec-6) focus on the columns of keywords like ‘head’.

VI. CONCLUSION

In this paper, we propose a new direction for the VQA task to learn efficient and elastic models that can fit different platforms adaptively. To address this issue, we present a general Doubly Slimmable Transformer (DST) framework that can be seamlessly integrated with any Transformer-based VQA model in theory. By integrating the DST framework with two typical Transformer-based VQA approaches, the resulting slimmable models outperform the state-of-the-art methods with similar model sizes or achieve comparable performance with much smaller models on both VQA-v2 and GQA datasets.

To the best of our knowledge, the proposed DST framework is the first attempt to explore efficient and elastic models for VQA. We hope this general and effective framework will serve as a solid baseline to inspire future research on efficient multimodal learning.

REFERENCES

[1] Z. Yu, J. Yu, Y. Cui, D. Tao, and Q. Tian, “Deep modular co-attention networks for visual question answering,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 6281–6290.
[2] Y.-C. Chen, L. Li, L. Yu, A. E. Kholy, F. Ahmed, Z. Gan, Y. Cheng, and J. Liu, “Uniter: Universal image-text representation learning,” in European Conference on Computer Vision (ECCV), 2020, pp. 104–120.
[3] Z. Yu, Y. Cui, J. Yu, M. Wang, D. Tao, and Q. Tian, “Deep multimodal neural architecture search,” in Proceedings of the 28th ACM International Conference on Multimedia, 2020, pp. 3743–3752.
[4] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang, “Bottom-up and top-down attention for image captioning and visual question answering,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 6077–6086.
[5] L. Yu, Z. Lin, X. Shen, J. Yang, X. Lu, M. Bansal, and T. L. Berg, “Mattenet: Modular attention network for referring expression comprehension,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 1307–1315.
[6] Z. Yu, Y. Cui, J. Yu, D. Tao, and Q. Tian, “Multimodal unified attention networks for vision-and-language interactions,” arXiv preprint arXiv:1908.04107, 2019.
[7] H. Tan and M. Bansal, “Lxmert: Learning cross-modality encoder representations from transformers,” in Conference on Empirical Methods in Natural Language Processing (EMNLP), 2019, pp. 5100–5111.
[8] J. Lu, D. Batra, D. Parikh, and S. Lee, “Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks,” in Neural Information Processing Systems (NIPS), 2019, pp. 13–23.
[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in Neural Information Processing Systems (NIPS), 2017, pp. 6000–6010.
[10] Z. Yu, J. Yu, C. Xiang, J. Fan, and D. Tao, “Beyond bilinear: Generalized multimodal factorized high-order pooling for visual question answering,” IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 12, pp. 5947–5959, 2018.
[11] J.-H. Kim, J. Jun, and B.-T. Zhang, “Bilinear attention networks,” Neural Information Processing Systems (NIPS), vol. 31, 2018.
[12] W. Su, X. Zhu, Y. Cao, B. Li, L. Lu, F. Wei, and J. Dai, “Vi-bert: Pre-training of generic visual-linguistic representations,” in International Conference on Learning Representations (ICLR), 2020.
[13] X. Li, X. Yin, C. Li, P. Zhang, X. Hu, L. Zhang, L. Wang, H. Hu, L. Dong, F. Wei et al., “Oscar: Object-semantic aligned pre-training for vision-language tasks,” in European Conference on Computer Vision (ECCV), 2020, pp. 121–137.
[14] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter,” arXiv preprint arXiv:1910.01108, 2019.
[15] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, “Tinybert: Distilling bert for natural language understanding,” in ACL, 2019.
[16] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou, “Mobilebert: a compact task-agnostic bert for resource-limited devices,” in Annual Meeting of the Association for Computational Linguistics (ACL), 2020, pp. 2158–2170.
[17] J. Yu, L. Yang, N. Xu, J. Yang, and T. Huang, “Simmable neural networks,” in International Conference on Learning Representations (ICLR), 2018.
[18] L. Hou, Z. Huang, L. Shang, X. Jiang, X. Chen, and Q. Liu, “Dynabert: Dynamic bert with adaptive width and depth,” in Neural Information Processing Systems (NIPS), 2020, pp. 9782–9793.
[19] B. Zhou, Y. Tian, S. Sukhbaatar, A. Szlam, and R. Fergus, “Simple baseline for visual question answering,” arXiv preprint arXiv:1512.02167, 2015.
[20] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh, “Vqa: Visual question answering,” in IEEE International Conference on Computer Vision (ICCV), 2015, pp. 2425–2433.
