Simultaneously Training and Compressing Vision-and-Language Pre-training Model

Qiaosong Qi, Aixi Zhang, Yue Liao, Wenyu Sun, Yongliang Wang, Xiaobo Li, Si Liu

Abstract—Model compression is an essential step for large-scale pre-training models toward practical application and deployment on the edge device. However, when conventional compression methods following ‘pre-training then compressing’ two-phase pipeline are applied to Vision-and-Language Pre-training (VLP) models, it will lead to a high calculation and memory overhead. In this work, we break the two-phase pipeline and propose an efficient and effective one-phase VLP model compression mechanism, named REDUCER, which stands for ‘simultaneously training and compREssing’ VLP model via progressive module replacing and network Rewiring. Specifically, REDUCER consists of three insightful designs. Firstly, we design a one-phase compression framework to train and compress the VLP model simultaneously to avoid the extra calculation and memory cost caused by an isolated model compression phase in the conventional two-phase pipeline. Secondly, we propose an adaptive progressive module replacing mechanism to compress the model depth free from explicit knowledge distillation losses, relieving the multi-task optimization problems. Thirdly, we integrate pruning techniques into VLP model compression to simultaneously compress the model in width and depth. Overall, we obtain a lightweight VLP model with only one pre-training phase, and it is the first one-phase compression method for VLP models. Extensive experiments have been conducted on representative VLP models, i.e., ClipBERT and VICTOR, and the experimental results show a superior trade-off between performance and efficiency.

Index Terms—Vision-and-Language, Model Compression, Pre-training Model.

I. INTRODUCTION

ARGE-SCALE pre-training technologies [1]–[6] have achieved great successes in Natural Language Processing (NLP). Meanwhile, the pre-training-then-transfer mechanism refreshes the state-of-the-art (SOTA) performance across various NLP tasks and pushes the NLP development to the next stage. The revolutionary success of pre-training strategies in NLP has raised a fashion in many artificial intelligence areas. Especially, the Vision-and-Language (VL) understanding presents a steady breakthrough with the development of pre-training technologies. The VL Pre-training (VLP) models, including text-image level [7]–[13] and text-video level [14]–[17], have significantly pushed the performance boundaries of many VL tasks.

Benefiting from advances in computing resources, the pre-training field is developing in the direction of extremely large scale and extremely large amounts of data to obtain a high-performance and generalizable basic model. However, the training resources and data consumption of extremely large-scale pre-training models limit its further development and practical application. Especially in the VL field, computing power consumption is an order of magnitude increase compared with NLP field. Therefore, it has become a problem worthy of attention to make a trade-off in the performance and computational complexity through appropriate model design and learning strategies. In this paper, we aim to explore how to produce a lightweight but accurate VLP model.

As shown in Figure 1a, conventional VLP model compression methods are mostly with a two-phase ‘pre-training then compressing’ pipeline: First, pre-train the large-scale model to obtain the teacher and abstract a lightweight model from the teacher as the student. Second, train the student supervised by both the ground truths of proxy and the teacher. This two-phase mechanism has proved its satisfactory effect.
for linguistic pre-training model compression [18]–[23], there is still a significant gap to employ conventional model compression techniques in the VLP model compression to achieve a satisfactory performance. We summarize three main challenges for adapting conventional compression methods into the VLP models. **Challenge 1:** enormous extra calculations and memory cost. As mentioned above, two-phase methods need an isolated phase to train the student supervised by the teacher, where the student and teacher are required to forward together in the second phase, seriously increasing calculations and memory. The extra cost is unacceptable for the VLP model during the pre-training. **Challenge 2:** optimization difficulty in multi-task learning. VLP model training usually relies on multiple proxy tasks for a good representation ability. Meanwhile, conventional KD needs to transfer knowledge with various soft targets, *i.e.*, logits-based, feature-based and embedding-based knowledge. Therefore, proxy tasks and soft targets with unbalanced relationships make the model hard to train. **Challenge 3:** suffering from a proper lightweight model design. To ensure the semantic integrity of the model during weights inheriting, manually abstracting the student from the teacher does not alleviate the over-parameterization of model width.

As shown in Figure 1b, to address the above challenges, we propose a novel one-phase VLP model compression method named REDUCER, ‘simultaneously training and compREssing’ VLP model via progressive moDUle replaCing and nEtwork Rewiring during the pre-training stage. In REDUCER, we propose three corresponding solutions for the above three challenges. **Solution 1:** We break the conventional two-phase pipeline and design a one-phase pre-training compression method to compress the model only through a pre-training phase. In this way, we can directly produce a lightweight VLP model after one pre-training phase, thus avoiding huge costs produced by an explicit KD phase. Specially, we save more than one-half training computation cost compared with two-phase methods, significantly speeding up the production process of the lightweight VLP model. **Solution 2:** Inspired by BERT-of-Theseus, we propose an Adaptive Progressive Module Replacing (APMR) mechanism to adaptively compress the model in the pre-training stage. Unlike BERT-of-Theseus, we adopt APMR in the pre-training stage, and our APMR dynamically tunes the replacing rate according to the model convergence degree. In this way, we avoid redesigning the training losses and achieve good adaptability to different backbones. Besides, we propose a self-KD based weight sharing mechanism in the APMR to further reduce memory cost. **Solution 3:** We implement a pruning technique to compress the model width. Specifically, we choose two representative video-linguistic pre-training models, *i.e.*, ClipBERT and VICTOR, to conduct our experiments. The experimental results show that our REDUCER outperforms the existing two-phase model compression methods, *e.g.*, in the common downstream tasks with the same compression ratio 50%. Moreover, compressed by REDUCER with a higher compression ratio 75%, the model retains more than 98% average accuracy of the original model on all the classification, Question Answer (QA) and Multiple-Choice tasks.

The main contributions of REDUCER are summarized as follows:

- We propose to simultaneously pre-train and compress VLP model in a one-phase manner with almost no additional overhead, which is friendly to VLP models with multiple proxy tasks. To the best of our knowledge, REDUCER is the first work of one-phase compression for VLP models.
- We propose Adaptive Progressive Module Replacing (APMR) with self-KD and model width pruning to support the one-phase compression. The model over-parameterization is relieved to a certain extent for both width and depth.
- We conduct extensive experiments to verify the superiority of the performance and efficiency of our proposed REDUCER.

**II. Related Work**


In the multi-modal field, DistillVLM [18] proposed the first knowledge distillation technique to compress large visual-linguistic models. DistillVLM firstly aligned the hidden representations and attention distributions between a large-scale VL model as the teacher and a small VL model as the student, then transferred the aligned knowledge from the teacher into the corresponding student, and finally trained
Our proposed REDUCE have three main improvements over conventional model compression and self-KD methods.

Firstly, we train and compress the VLP model simultaneously only through a pre-training phase. Secondly, we propose APMR and design a self-KD based weight sharing mechanism in the APMR to efficiently compress the VLP models. Thirdly, We combine a rewired network pruning strategy with APMR to simultaneously compress the model in width and depth. Through the above three improvements, we tune REDUCER into the first efficient and effective one-phase VLP model compression mechanism.

### III. METHODS

In this section, we elaborate the pipeline of our ‘simultaneously training and compressing’ one-phase VLP model compression method, REDUCER. We first present an overview of the pre-training process in Sec III-A. Then we introduce the insightful compression techniques in the REDUCER. In specific, Sec III-B introduces the Adaptive Progressive Module Replacing (APMR) mechanism along with a self-KD technique to further save memory cost, and Sec III-C adopts a pruning method to compress the model width with network rewiring.

#### A. One-Phase Compression Pipeline

In this subsection, we introduce the one-phase pipeline, REDUCER, during the pre-training stage. As shown in Figure 2, the initial model is compressed into a lightweight VLP model.
model experiencing several states with the training going. Specifically, we define the original model as Predecessor and the compressed model as Successor, and we group $N$ Predecessor Layers (PL) as a P-Module and one Successor Layer (SL) as an S-Module. We design an Adaptive Progressive Module Replacing (APMR) mechanism to replace part of the original large-scale P-Modules with lightweight S-Modules adaptively and transfer knowledge from the P-Modules to the S-Modules to compress the model in depth. The compression ratio is gradually increasing with the passage of training iters. Besides, we rewire the model network according to importance ranking and then prune the model width for further compression. Finally, all layers of the compressed lightweight model are replaced with the width pruned SLs. The whole training process is supervised with only ground truths of the proxy tasks and no soft targets supervision involved.

B. Adaptive Progressive Module Replacing

The representation learning of the VLP models is usually pre-trained through multiple proxy tasks, where Table I shows the proxy tasks of some mainstream VLP models. Moreover, conventional KD needs to transfer feature-based knowledge with various soft targets. These targets may have a significant impact on the optimization of the model, which leads to great difficulty for cooperative training with multi targets. To address this, we design an adaptive progressive module replacing (APMR) distillation scheme for the VL pre-training stage inspired by BERT-of-Theseus (BOT), as shown in Figure 3. The Replacing scheduler will add the corresponding S-Module or P-Module to the computational graph each time instead of simultaneous training them. In this way, we can compress the model and transfer knowledge without explicit KD losses and avoid additional computational overhead. Then, we replace the original heavy P-Modules with the lightweight S-Modules during the pre-training stage for one-phase compression with a newly designed replacing scheduler for modules

![Replacing scheduler](image)

The replacing rate $r$ indicates the probability distribution for S-Modules to replace P-Modules during compression training, and it follows a standard Binomial distribution $r \sim \text{BinomialDistribution}(n, p)$, where $p$ denotes the replacing probability and $n$ stands for the count

![Replacing probability](image)
of the P-Modules as well as S-Modules. We design an adaptive replacing probability strategy to stably tune $p$ according to the gradient change amplitude, i.e., the gradient momentum change rate. As shown in the Algorithm 1, the strategy adjusts the replacing probability $p$ adaptively and smoothly with the training gradient. Specifically, the probability increases rapidly or slowly when the training gradient decreases steadily or steeply, while it decreases with the gradient increasing. The upper limit of the replacing probability is set to 0.5 to ensure that the P-Modules can be sufficiently trained. We show the adaptive replacing probability $p$ (ARP: the blue line in Figure 4) at each training epochs from the real experiment log. As shown in Figure 4, our ARP can make the VLP model learning stably. Finally, we get the adaptive replacing rate $r$ according to the Binomial distribution $r \sim B(n, p)$. For stability, when the model converges to a more stable stage, we continue to train several epochs with S-Modules only.

Comparison with BOT. BOT implemented model replacing in the fine-tuning stage, and the convergence process is relatively stable and fast, since the P-Modules are well pre-trained. Therefore, BOT designed two kinds of replacing probability as shown in Figure 4: i) constant probability, ii) the probability increasing linearly with the training step. However, we argue that these replacing probabilities are inappropriate for one-phase compression: i) Firstly, it is challenging to select the appropriate magic number for better convergence, including the constant number or the linear slope; ii) Secondly, one-phase compression is usually not stable enough with drastic changes, e.g., if at some iter the model randomly replaces too many S-Modules, the training will irretrievably move towards the direction of irreducible convergence. Comparably, our newly designed adaptive replacing rate is more stable and applicable for one-phase compression.

Self-KD. APMR replaces large-scale modules with lightweight modules gradually in the pre-training stage for saving training time. However, as shown in Figure 3, we need to apply memory for both P-Modules and S-Modules, while we only use part of the memory during inference. Thus, it makes a waste of memory with the amount of compression ratio compared to the training with P-Modules only. For example, when the compression ratio is set to 0.6, we have 1.6 times memory compared with the memory cost for P-Modules only in the training stage. To save the additional memory cost of S-Modules during training, we utilize a strategy of complete module guiding partial module training. Specifically, we design the S-Module to be a part of the P-Module, which means that we can share parameters from P-Module to S-Module. The model design is shown in Figure 5. The same color between P-Modules and S-Modules stands for parameter sharing. To alleviate drastic network changes during module replacing, we add a lightweight MLP layer as a buffer in each S-Module.

C. Model Width Pruning

We conduct the model width pruning after network rewiring according to the parameter importance for a thorough one-stage compression.

Firstly similar with [41], [42], [48], we calculate the importance of the attention heads and neurons in FFN layers according to the variation of the gradient. The importance of the $K$th head is denoted as $I_K$, which is estimated as the gradient of corresponding head mask, i.e., $g_{mK}$. Then the importance set $M_h$ for multi-heads can be estimated as

$$M_h = \{ I_1, I_2, I_3, \ldots, I_N \} = \{ |g_{m1}|, |g_{m2}|, |g_{m3}|, \ldots, |g_{mN}| \}$$

(1)

For a neuron of a FFN layer, we denote $w_i$ stands for the weight of a neuron and $b_i$ stands for the bias, then the parameter set $P^1$ for FFN intermediate layer and $P^2$ for FFN output layer can be expressed as $P^1 = \{ (w_1^1, b_1^1), (w_2^1, b_2^1), \ldots, (w_K^1, b_K^1) \}$ and $P^2 = \{ w_1^2, w_2^2, \ldots, w_K^2 \}$. The importance $M_n$ for a neuron in the FFN layer is formulated as

$$M_n = \sum_{i=1}^{d} g_{i} w_i^1 + g_{i} b_i^1 + \sum_{i=1}^{d} g_{i} w_i^2$$

(2)

where $g_{i}$ denotes the gradient of the neuron.

Then with the calculated importance of heads and FFN neurons, we rewire the partial module’s attention heads and FFN neurons based on their importance. Finally we train the heads and neurons with an abandoning rate equal to the width compression rate ranking by the importance. The model design is shown in Figure 6.

IV. EXPERIMENTS

In this section, we first introduce the datasets and tasks. Then we introduce the pre-training model backbones and experimental settings. Finally we demonstrate the comprehensive experiments and ablation studies on different kinds of downstream tasks.
TABLE II
CROSS-MODAL RETRIEVAL, MULTIPLE-CHOICE, CLASSIFICATION AND QA RESULTS FOR ClipBERT WITH DIFFERENT WIDTH AND DEPTH SETTING $m_d \times m_w$ COMPRESSED BY OUR REDUCER AND THE TWO-PHASE REDUCER(REDUCER$_2$) ON MSRVTT AND YouCook2 DATASETS.

<table>
<thead>
<tr>
<th>Method</th>
<th>$m_d \times m_w$</th>
<th>MSRVTT</th>
<th>Classification</th>
<th>QA</th>
<th>Multiple-Choice</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>R@1↑</td>
<td>R@5↑</td>
<td>R@10↑</td>
<td>ACC↑</td>
</tr>
<tr>
<td>baseline</td>
<td>12 × 768</td>
<td>9.46</td>
<td>28.76</td>
<td>41.22</td>
<td>16.60</td>
</tr>
<tr>
<td>REDUCER$_2$</td>
<td>6 × 384</td>
<td>4.98</td>
<td>17.76</td>
<td>28.70</td>
<td>29.30</td>
</tr>
<tr>
<td>REDUCER</td>
<td>6 × 384</td>
<td>7.55</td>
<td>21.60</td>
<td>33.45</td>
<td>24.00</td>
</tr>
</tbody>
</table>

TABLE III
CROSS-MODAL RETRIEVAL, MULTIPLE-CHOICE, CLASSIFICATION AND QA RESULTS FOR ClipBERT WITH DIFFERENT WIDTH AND DEPTH SETTING $m_d \times m_w$ COMPRESSED BY OUR REDUCER, RANDOM PRUNING AND OTHER REPRESENTATIVE TWO-PHASE COMPRESSION METHODS ON MSRVTT AND YouCook2 DATASETS.

<table>
<thead>
<tr>
<th>Method</th>
<th>$m_d \times m_w$</th>
<th>MSRVTT</th>
<th>Classification</th>
<th>QA</th>
<th>Multiple-Choice</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>R@1↑</td>
<td>R@5↑</td>
<td>R@10↑</td>
<td>ACC↑</td>
</tr>
<tr>
<td>baseline</td>
<td>12 × 768</td>
<td>9.46</td>
<td>28.76</td>
<td>41.22</td>
<td>16.60</td>
</tr>
<tr>
<td>random</td>
<td>6 × 768</td>
<td>3.54</td>
<td>12.20</td>
<td>19.04</td>
<td>55.10</td>
</tr>
<tr>
<td>DistillBERT [19]</td>
<td>6 × 768</td>
<td>8.30</td>
<td>25.60</td>
<td>37.15</td>
<td>22.00</td>
</tr>
<tr>
<td>LayerDrop [28]</td>
<td>6 × 768</td>
<td>7.40</td>
<td>24.15</td>
<td>33.75</td>
<td>22.00</td>
</tr>
<tr>
<td>REDUCER</td>
<td>6 × 768</td>
<td>7.50</td>
<td>24.55</td>
<td>36.40</td>
<td>19.50</td>
</tr>
<tr>
<td>random</td>
<td>12 × 384</td>
<td>5.98</td>
<td>20.24</td>
<td>30.16</td>
<td>29.80</td>
</tr>
<tr>
<td>REDUCER</td>
<td>12 × 384</td>
<td>9.16</td>
<td>26.74</td>
<td>38.80</td>
<td>19.00</td>
</tr>
<tr>
<td>random</td>
<td>6 × 384</td>
<td>1.00</td>
<td>4.20</td>
<td>7.23</td>
<td>143.40</td>
</tr>
<tr>
<td>REDUCER</td>
<td>6 × 384</td>
<td>7.55</td>
<td>21.60</td>
<td>33.45</td>
<td>24.00</td>
</tr>
</tbody>
</table>

Fig. 6. Network rewiring and model width pruning according to the importance of attention heads and neurons. The shade of color represents the strength of importance.

A. Datasets and Tasks

Pre-Train. We pre-train the VL understanding model with the large-scale video-text dataset Howto100M [49]. Since pre-training convergence does not enhance a model’s compressibility [50], we make some simplifications to save computing resources and conduct more adequate experiments, where we use $5M$ subset of the whole Howto100M dataset.

Downstream tasks. We selected four groups of tasks that are most commonly used to measure the representational ability of VLP models, namely Cross-Modal Retrieval [34], [51]–[54], Classification [55]–[58], Question Answering (QA) [59], [60], and Multiple-Choice (MC). We evaluate our REDUCER on the MSRVTT [61] and YouCook2 [62] datasets for the downstream tasks. MSRVTT contains 10K YouTube videos with 200K video descriptions. We follow ClipBERT [14] to train the model with 7K training and validation videos and evaluate the model with 1K test videos. YouCook2 contains 2K YouTube videos with 11K video-text pairs, and we split 8K video-text pairs for training and 3K video-text pairs for testing.
TABLE IV
Cross-Modal Retrieval, Multiple-Choice, classification and QA results for ClipBERT and Victor with multi-level width \( m_w \) compressed by our REDUCER on MSRVTT and YouCook2 datasets.

<table>
<thead>
<tr>
<th>backbone</th>
<th>( m_w )</th>
<th>Cross-Modal Retrieval</th>
<th>Classification</th>
<th>QA</th>
<th>Multiple-Choice</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>MSRVT T</td>
<td>MSRVT T</td>
<td>YouCook2</td>
<td>MSRVT T</td>
</tr>
<tr>
<td>ClipBERT</td>
<td>1.00</td>
<td>9.46</td>
<td>28.76</td>
<td>41.22</td>
<td>16.60</td>
</tr>
<tr>
<td></td>
<td>0.75</td>
<td>9.78</td>
<td>27.80</td>
<td>39.28</td>
<td>19.20</td>
</tr>
<tr>
<td></td>
<td>0.50</td>
<td>9.16</td>
<td>26.74</td>
<td>38.80</td>
<td>19.00</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>7.42</td>
<td>23.98</td>
<td>35.30</td>
<td>23.00</td>
</tr>
<tr>
<td>Victor</td>
<td>1.00</td>
<td>3.68</td>
<td>20.76</td>
<td>32.38</td>
<td>24.20</td>
</tr>
<tr>
<td></td>
<td>0.75</td>
<td>3.04</td>
<td>14.50</td>
<td>25.46</td>
<td>29.80</td>
</tr>
<tr>
<td></td>
<td>0.50</td>
<td>3.24</td>
<td>14.20</td>
<td>24.10</td>
<td>31.80</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>2.98</td>
<td>12.86</td>
<td>22.04</td>
<td>35.40</td>
</tr>
</tbody>
</table>

TABLE V
Cross-Modal Retrieval, Multiple-Choice, classification and QA results for ClipBERT and Victor with multi-level depth \( m_d \) compressed by our REDUCER on MSRVTT and YouCook2 datasets.

<table>
<thead>
<tr>
<th>backbone</th>
<th>( m_d )</th>
<th>Cross-Modal Retrieval</th>
<th>Classification</th>
<th>QA</th>
<th>Multiple-Choice</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>MSRVT T</td>
<td>MSRVT T</td>
<td>YouCook2</td>
<td>MSRVT T</td>
</tr>
<tr>
<td>ClipBERT</td>
<td>1.00</td>
<td>9.46</td>
<td>28.76</td>
<td>41.22</td>
<td>16.60</td>
</tr>
<tr>
<td></td>
<td>0.75</td>
<td>9.78</td>
<td>27.80</td>
<td>39.28</td>
<td>19.20</td>
</tr>
<tr>
<td></td>
<td>0.50</td>
<td>9.16</td>
<td>26.74</td>
<td>38.80</td>
<td>19.00</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>7.42</td>
<td>23.98</td>
<td>35.30</td>
<td>23.00</td>
</tr>
<tr>
<td>Victor</td>
<td>1.00</td>
<td>3.68</td>
<td>20.76</td>
<td>32.38</td>
<td>24.20</td>
</tr>
<tr>
<td></td>
<td>0.75</td>
<td>3.04</td>
<td>14.50</td>
<td>25.46</td>
<td>29.80</td>
</tr>
<tr>
<td></td>
<td>0.50</td>
<td>3.24</td>
<td>14.20</td>
<td>24.10</td>
<td>31.80</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>2.98</td>
<td>12.86</td>
<td>22.04</td>
<td>35.40</td>
</tr>
</tbody>
</table>

B. Experimental Setting

We firstly use ClipBERT [14] as the backbone and utilize video and text of Howto100M for the pre-training experiments. To verify the generality of our REDUCER for transformer-style backbones, we extend the experiments to another VLP model, Victor [17].

We use the 30,000 tokens vocabulary and initialize the ClipBERT backbone with the parameters of the ClipBERT model trained by COCO Captions [63] and Visual Genome Captions [64]. The maximal number of input text is 10, and the maximal number of video frames is 8. For each video, we first cut the segments according to the subtitles, then sample one frame per second for each video segment, and next randomly select three frames to combine with the corresponding subtitle for pre-training. We adopt the setting of transformer blocks and CNN blocks consistently with ClipBERT. The value of the weighted average \( \beta \) of the adaptive replacing rate is 0.999, and the temperature value \( \tau \) is 0.05. We use BERTAdam [1] as the optimizer with an initial learning rate of \( 10^{-4} \) and a batch size of 128 per GPU. For each experiment, we train the VLP model for 10 epochs on 32 NVIDIA Tesla V100 GPUs.

Then we adopt the lightweight VLP model to fine-tune the downstream tasks on with MSRVTT [61] and YouCook2 [62]. We set the learning rate to \( 10^{-5} \) on all tasks and set the batch size to 32 on all tasks except that the batch size is 16 on the cross-Modal retrieval task. We train the classification task with 4 GPUs for 10 epochs, the QA task with 8 GPUs for 10 epochs, and cross-Modal retrieval and multiple-choice tasks with 4 GPUs for 50 epochs. We train all downstream tasks 5 times repeatedly with random seeds from 5671 to 5675 and report the average results.

C. Results

Main Results. We train the ClipBERT baseline with depth and width as \( 12 \times 768 \). REDUCER compresses the model depth to 6 and width to 384, where achieves 75% compression ratio. From the precision perspective, as shown in Table II, for all the four evaluated downstream tasks, the precision degradation is slight and acceptable. For example for classification and multiple-choice with MSRVTT, compared to the baseline, REDUCER degrades accuracy by only 2.35% and 4.46% percentage, respectively. For QA with MSRVTT, REDUCER even outperforms the baseline by 17%. This reflects effective compression on redundant parameters will not damage performance, and sometimes even improve it.

Comparison with Two-Phase Compression. We compare our REDUCER with the two-phase compression process. We split our one-phase training into two separate phases, \textit{i.e.}, firstly train 10 epochs for pre-training, and then implement the progressive module replacing and model width pruning to train another 5 epochs. Table II indicates the performance of REDUCER is significantly better than the two-phase training, marked as REDUCER\(_2\), since two-phase training
Fig. 8. Comparison of (a) Performance v.s. FLOPs and (b) Performance v.s. #Parameters between our proposed REDUCER and other methods on the MSRVTT and YouCook2 datasets. The figure demonstrates performances of ClipBERT compression for different downstream tasks including Cross-Modal Retrieval, Multiple-Choice, classification, and QA.

requires more time to achieve comparable results. Figure 7 compares the Encoder FLOPs during the compression training process. The one-phase training REDUCER requires much less training computation cost since it prunes model depths and model widths much earlier than the two-phase training REDUCER$_2$. The AUC (Area Under The Curve) indicates REDUCER significantly saves more than one-half of training resources compared with REDUCER$_2$. To be noted, this timing of width pruning is set to be around the third epoch, since the original training is nearly converged and it’s easier to converge with an additional width pruning. In addition, we hope that the pruning process can be completed earlier to reduce the frequency of simultaneous compression of depth and width, so we choose to complete width compression at one time.

D. Discussion

Comparison with Previous Two-Phase Methods. We compare our REDUCER with two representative two-phase compression methods of DistillBERT [19] and LayerDrop [28]. We implement DistillBERT and LayerDrop on ClipBERT, compress ClipBERT’s transformer layers by 50% percent on depth and keep the original width since they do not support width pruning. We use ClipBERT pretrained on Howto100M as the teacher model, and initialize the student model with the first 6 layers of the teacher model. When implementing DistillBERT, since the loss of multiple proxy tasks is also used for supervision, we do not adopt the cosine loss between hidden layers for better convergence. Then, we first train the teacher model for 10 epochs as the first phase, and then train the student model for 5 epochs as the second phase. For LayerDrop implementation, we set 0.15 drop rate and train for 10 epochs.

In terms of performance, as shown in Table III, we conduct the comparison with 50% model depth compression ratio, i.e., from baseline 12×768 to target 6×768. Generally, REDUCER outperforms DistillBERT and LayerDrop with large margins on all the four downstream tasks, e.g., 1.25 margin for classification and 2.15 margin for QA with MSRVTT compared to LayerDrop. Although DistillBERT achieves a higher R@1-10 score for the Cross-Modal Retrieval task, REDUCER achieves a higher MdR (Median rank) score that can comprehensively reflect the cross-modal retrieval task. To be noted, REDUCER is the only one that can surpass the baseline on several tasks. Then, REDUCER is able to alleviate the over-parameterization in model width to 6×384, and maintains comparable or better performance with LayerDrop (e.g. 35.84 v.s. 34.57 on QA task) at a higher compression rate (75% v.s. 50%).

In terms of the training cost, Figure 7 compares RE-
DUCER with DistillBERT and LayerDrop. In the first stage, the FLOPs of DistillBERT and LayerDrop are 25.82 G, and the FLOPs of REDUCER gradually decrease from 25.82 G to 6.46 G. In the second phase, the FLOPs of DistillBERT is 38.73 G (25.82 G for Teacher and 12.91 G for Student), and the FLOPs of LayerDrop is 21.95 G (with drop rate 0.15). In the inference stage, the FLOPs of DistillBERT and LayerDrop are 12.91 G, and the FLOPs of REDUCER is 6.46 G.

**Comparison with One-Phase Random Pruning.** We compare REDUCER with random pruning compression, which randomly prunes model depth without APMR and randomly prunes model width without network rewiring. As shown in Table III, REDUCER outperforms random pruning on all downstream tasks with all depth and width settings, which proves the effectiveness of APMR on depth pruning and network rewiring on width pruning. Take classification with MSRVTT as example, REDUCER outperforms random pruning by large accuracy margins of 8.7 for 6 × 708, 4.84 for 12 × 384 and 14.43 for 6 × 384, respectively.

**Analyze Over-Parameterization.** We compress the backbones of ClipBERT and VICTOR at different width and depth compression ratios. Table IV shows the width compression and Table V shows the depth compression. The results prove that both these two kinds of backbones have a large number of redundant parameters on both width and depth dimensions, and pruning these parameters will not introduce large accuracy degradation. For the baseline of 12 × 708, most of the downstream tasks achieve best performance as 0.75 width keeping ratio and 0.5 depth keeping ratio, which indicates the other parameters are redundant and should be pruned. This also explains why REDUCER achieves good performance with large compression ratio, and proves that REDUCER extensively works for different kinds of backbones.

**Performance v.s. Flops and Parameters.** We count the FLOPs (at batch size of 1) and parameters of encoder structures for different compression methods with various compression ratio settings, and show the corresponding performance of downstream tasks in Figure 8. REDUCER at 6 × 384 achieves the best trade-off between performance and computation FLOPs, so as the memory cost when going to the parameters comparison. To be noted, REDUCER(6×384) is almost always on top of the extension trend of REDUCER_D and REDUCER_W, which proves that, compressing model depth and width together can achieve overall best performance. The results show that the depth direction is more robust to compression than the width direction for the VLP model on both the classification and QA tasks, while on the cross-modal retrieval and multiple-choice tasks, the width direction exhibits higher compression robustness.

**V. CONCLUSION**

In this paper, we propose a one-phase ‘simultaneously training and compressing’ method, REDUCER, to generate a lightweight VLP model in the pre-training stage. We design a mechanism of adaptive progressive module replacing and model width pruning with network rewiring to compress the transformer structure in one phase. Thus, we compress the model with almost no additional overhead, which saves pre-training costs, and is friendly to VLP models with multiple proxy tasks. Experiments show that REDUCER alleviates the VLP model’s depth and width over-parameterization and maintains good accuracy at high compression ratios.

**REFERENCES**


[29] J. B. Lei and R. Carluana, “Do deep nets really need to be deep,” in NIPS, 2014. 3


Qiaosong Qi received the Master degree from the Wuhan University of Science and Technology, Wuhan, China, in 2018. He is now a researcher at Taobao (China) Software Co. Ltd., Alibaba Group. His research interests include lightweight neural networks, computer vision and multimodal video understanding.

Aixi Zhang received the Mphil degree from the Hong Kong University of Science and Technology, Hong Kong, China, in 2014. He is now a senior researcher at Taobao (China) Software Co. Ltd., Alibaba Group. His research interests include computer vision, HOI detection and multimodal video understanding. He has published 4 papers at top conferences and journals including NIPS, CVPR, ACM MM and TIP. He was the Champion of CVPR 2021 ActivityNet Homage Challenge and CVPR 2020 DeepFashion2 Fashion Image Retrieval Challenge.
Yue Liao is currently a Ph.D. candidate at School of Computer Science and Engineering, Beihang University. He received his Master degree from Institute of Information Engineering, Chinese Academy of Sciences. His research interests include human-object interaction detection, visual grounding and object detection. He has published more than 10 papers at top journals and conferences including T-PAMI, T-IP, NIPS, CVPR and ECCV, etc. He was the Champion of CVPR 2021 ActivityNet Homage Challenge. He has been serving as a reviewer for numerous academic journals and conferences such as TMM, CVPR, ICCV, ECCV, AAAI and ACM MM.

Wenyu Sun received the Master degree from Peking University, in 2022. He is currently an algorithm engineer at Taobao (China) Software Co. Ltd., Alibaba Group. His research interest includes multimodal models and efficient model compression.

Yongliang Wang received the Master degree from Beijing Jiaotong University, in 2011. He is currently working at Taobao (China) Software Co. Ltd., Alibaba Group as a senior researcher. His research interests include computer vision, image caption and video understanding. He has published 2 top conference papers including NIPS and CVPR. He was also the Champion of 3 CVPR challenges including CVPR 2020 DeepFashion2 Fashion Retrieval, CVPR 2021 ActivityNet Homage, and CVPR 2021 VisWiz.

Xiaobo Li received the Master degree from Peking University, in 2009. He is currently working at Taobao (China) Software Co. Ltd., Alibaba Group as a senior expert. His research interests include computer vision, audio and video encoding, and video understanding. He was the recipient of second prize of National Scientific and Technological Progress Award in 2020.

Si Liu is currently a professor at Beihang University. She received her Ph.D. degree from Institute of Automation, Chinese Academy of Sciences. She has been a research assistant and Postdoc in National University of Singapore. Her research interest includes computer vision and multimedia analysis. She has published over 70 cutting-edge papers on the human-related analysis and vision-language understanding. She was the recipient of Best Paper Award of ACM MM 2021 and 2013, Best Demo Award of ACM MM 2012. She was the Champion of CVPR 2017 Look Into Person Challenge and the organizer of the ECCV 2018, ICCV 2019, CVPR 2021 and ACM MM 2022 Person in Context Challenges. She is the Associate Editor of IEEE TMM and TCSVT.