Lite-MDETR: A Lightweight Multi-Modal Detector

Qian Lou, Yen-Chang Hsu, Burak Uzkent, Ting Hua, Yilin Shen, Hongxia Jin
Samsung Research America
{qian.lou, yenchang.hsu, b.uzkent, ting.hua, yilin.shen, hongxia.jin}@samsung.com

Abstract

Recent multi-modal detectors based on transformers and modality encoders have successfully achieved impressive results on end-to-end visual object detection conditioned on a raw text query. However, they require a large model size and an enormous amount of computations to achieve high performance, which makes it difficult to deploy mobile applications that are limited by tight hardware resources. In this paper, we present a Lightweight modulated detector, Lite-MDETR, to facilitate efficient end-to-end multi-modal understanding on mobile devices. The key primitive is that Dictionary-Lookup-Transformormations (DLT) is proposed to replace Linear Transformation (LT) in multi-modal detectors where each weight in Linear Transformation (LT) is approximately factorized into a smaller dictionary, index, and coefficient. This way, the enormous linear projection with weights is converted into efficient linear projection with dictionaries, a few lookups and scalings with indices and coefficients. DLT can be applied to any pre-trained multi-modal detectors, removing the need to perform expensive training from scratch. To tackle the challenging training of DLT due to non-differentiable index, we convert the index and coefficient into a sparse matrix, train this sparse matrix during the fine-tuning phase, and recover it back to index and coefficient during the inference phase. Our experiments on phrase grounding, referring expression comprehension and segmentation, and VQA show that our Lite-MDETR achieves similar accuracy as the prior multi-modal detectors with up to $\sim 4.1\times$ *model size reduction.*

1. Introduction

Recently, multi-modal models on vision and language data have shown increased performance on vision and language tasks thanks to the adoption of transformers and large-scale pre-training [3,7,12,21,32]. These multi-modal transformers substantially benefit from large model complexity and computationally costly pre-training on large-scale datasets. Unfortunately, their large model complexity prohibits us from deploying them on resource-limited platforms, *i.e.*, mobile devices [19,31]. For this reason, improving the efficiency of recent multi-modal models is critical for real-world applications. However, there exists no method that can leverage already pre-trained multi-modal transformers while reducing their size as the existing methods [25,26,29] require pre-training the compressed model on large-scale datasets.

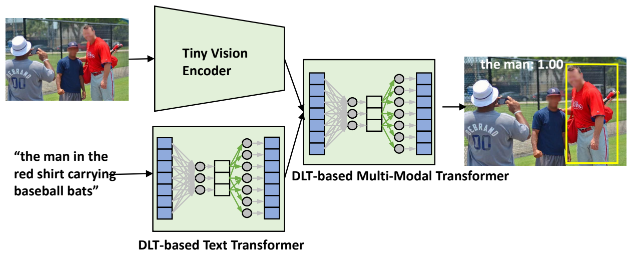
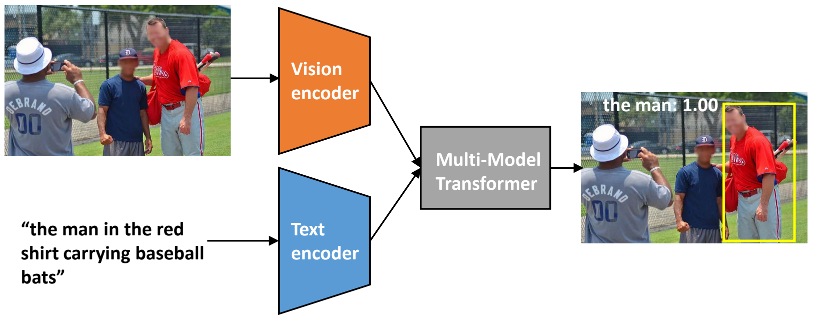

Tiny Vision Encoder
"the man in the red shirt carrying baseball bats"
DLT-based Text Transformer
DLT-based Multi-Modal Transformer
the man: 1.00

Figure 1. Our lightweight multi-modal Lite-MDETR using proposed DLT layers. We replace the linear transformation layers in text encoder and multi-modal transformer with more efficient DLT layers to reduce the model size.

Diving deeper into multi-modal models, we observe that linear transformer layer (LT) in individual transformers dominate large share of the overall model size. For example, a recently published state-of-the-art vision and languge model, called MDETR [12], uses a vision and language transformer based detector together with a language transformer and a CNN as backbones to process text and image inputs. We find out that the LT layers in the text encoder, a RoBERTa transformer [20], and vision and language transformer in MDETR occupy $\sim 90\%$ of the model size. As a result, it is reasonable to replace the traditional LT layers with more efficient layers to reduce the model size of MDETR to achieve high accuracy while being able to deploy it on resource-limited real-world platforms.

Inspired by this finding, we propose Lightweight Dictionary Lookup Transform (DLT) layer that can replace the costly LT layers in transformers in MDETR model. We show integration of our DLT layers in MDETR in Figure 1. As shown in the figure, we use the DLT layers in the text


Vision encoder
Text encoder
"the man in the red shirt carrying baseball bats"
Multi-Model Transformer
the man: 1.00
(a). Prior End-to-End Multi-Modal Detectors Conditioned on a Raw Text Query

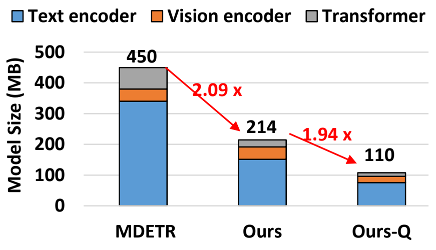


Figure 2. (a) The state-of-the-art multi-modal detector architecture used in MDETR [12]. (b) The model size breakdown of MDETR using ENB-3 as backbone and the effects of our lightweight architecture. Ours-Q means we further use 16-bit quantization technique on our lightweight architecture.

encoder and multi-modal transformer in MDETR. Our DLT layer compresses the representation of LT as a dictionary, scales and sums the sparse lookups on dictionary to recover the compressed representation. To further reduce the model size, we use the orthogonal quantization method already implemented in PyTorch [14, 23]. Our experiments show that our Lite-MDETR using DLT achieves similar accuracy with MDETR using LT on several tasks such as phrase grounding, referring expression comprehension and segmentation, and visual question answering. More spefically, DLT alone can reduce the model size of MDETR by $2.03\times$ with slight loss in accuracy whereas the orthogonal quantization further reduces the model size by $1.94\times$ as shown in Figure 2(b). We want to highlight that we achieve these results without pre-training Lite-MDETR on a large-scale dataset. In conclusion, these results increase the possibility of the application of MDETR on resource-limited platforms, *i.e*, mobile devices.

2. Background and Motivation

2.1. Multi-Modal Detectors

Transformers have recently replaced the CNNs for end-to-end single-stage object detection on images [2, 5, 6, 27, 34]. DETR [2], the first study in this direction, uses transformer to predict fixed number of bounding boxes without using any box proposals. Following this progress, transformers have also been applied to end-to-end object detection on the images conditioned on the language data [7, 12, 18]. For example, MDETR [12], uses the DETR transformers to detect objects given the multi-modal representations of CNN and text encoder, as shown in Figure 2.

In the pre-training stage, it performs end-to-end training for detecting objects mentioned in the image captions. In the fine-tuning stage, MDETR can be specifically tuned for different vision and language tasks that require detection, *i.e.*, referring expression, or not, *i.e.*, VQA. Follow up studies [7, 18] use similar architecture to MDETR as they contain image and text encoder together with multi-modal transformer.

2.2. Limitations of Existing Lightweight Methods for Multi-modal Detectors

One straightforward strategy to achieve lightweight MDETR is replacing the big transformer blocks in MDETR with compact blocks, like TinyBERT [11], DistilBERT [25], then pre-training this new model on a large corpus. Here we highlight that this generic pre-training [25, 26, 29] on a large corpus is necessary for a compact model to perform good performance on the downstream tasks. However, the generic pre-training often costs prohibited computational resources, and as a result, it may not be affordable for everyone. For instance, if we replace the text encoder or multi-modal transformer in MDETR with a compact model, *e.g.*, a TinyBERT [11], it may take almost 5376 NVIDIA V100 GPU hours to get the pre-trained lightweight MDETR [12]. Another line of strategy that can achieve a lightweight MDETR without using a generic pre-training is approximating the pre-trained MDETR parameters, i.e., weights factorization [9, 22], to inherit the knowledge of the big trained MDETR model. However, simply using weights factorization often results in a significant loss in task performance [22]. In this study, different from existing works, we provide a lightweight MDETR in which a generic pre-training is not necessary to match the accuracy of MDETR.

2.3. Motivation

In its original form, the MDETR model consumes almost 450MB model size that prohibits the model deployment to resource-constrained hardware. Thus, there is an urgent need to design a lightweight model. To propose a lightweight model, we investigate the three components in MDETR: vision encoder, text encoder and a multi-modal transformer that are shown in Figure 2(a). The vision en-

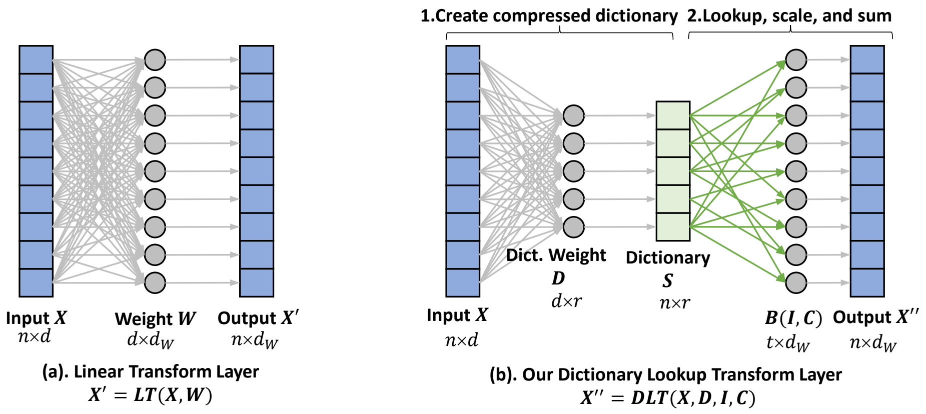

1.Create compressed dictionary
2.Lookup, scale, and sum
Input X $n\times d$
Weight W $d\times d_W$
Output X' $n\times d_W$
(a). Linear Transform Layer $X' = LT(X, W)$
Input X $n\times d$
Dict. Weight D $d\times r$
Dictionary S $n\times r$
$B(I, C)$ $t\times d_W$
Output X'' $n\times d_W$
(b). Our Dictionary Lookup Transform Layer $X'' = DLT(X, D, I, C)$

Figure 3. (a) Storage-bottleneck linear transform layer. (b) Our Lightweight dictionary lookup transformer layer (DLT). In DLT, we compress the representation as a dictionary and then expand the representation by looking up, scaling, and summing the dictionary.

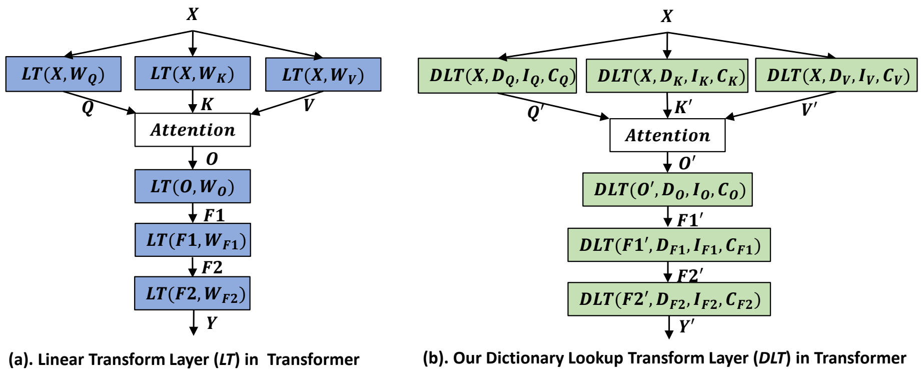


Figure 4. (a) Transformer block used in MDETR. (b) Lightweight transformer block used in our Lite-MDETR. For simplicity, we avoid using residual and normalization operations in (a) and (b) since they are used in both Linear Transform and our Dictionary Lookup Transform layer.

coder employs a small CNN as a backbone, *i.e.*, ENB-3 [28]; Text encoder uses a large transformer encoder, *e.g.*, RoBERTa [20], that occupies $\sim$ 83% of total MDETR model size; the multi-modal transformer is a transformer with hidden size 256. Further investigating the text encoder and the multi-modal transformer, we find out that the Linear transform (LT) layer is the main storage bottleneck as it occupies more than 90% of MDETR model size. For this reason, in this study our motivation is to replace the heavy LT layer with a more efficient layer while maintaining the accuracy of the MDETR and benefiting from the already pre-trained MDETR.

3. Lite-MDETR

3.1. Definition and Overview

For the state-of-the-art multi-modal MDETR, each text encoder block and transformer block shown in Figure 4(a) is heavily dependent on linear transform (LT) layer. Specifically, given each input or hidden state X, corresponding query Q, Key K, and Value V are generated by $LT(X, W_Q)$, $LT(X, W_K)$, and $LT(X, W_V)$, respectively. The $Attention$ module shown in Equation 1 takes Q, K, V as inputs, and generates outputs O that is projected by W_O using $LT(O, W_O)$ to combine multi-head information in transformer. The projected value F_1 is then fed into feed-forward network (FFN) layers as shown in Equation 2 to generate F_2 and next hidden state Y.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d}}V), \quad (1)$$

$$FFN(F1) = max(0, F1W_{F1} + b_{F1})W_{F2} + b_{F2}. \quad (2)$$

As Figure 4(b) shows, our proposed Lite-MDETR replaces each LT in MDETR with lightweight DLT. Instead of using large weights in LT, we propose DLT that depends on smaller dictionary D, index I, and coefficient C. During training, index I and coefficient C is converted to sparse matrix B; during inference and deployment, B

is stored as dense and tiny I and C. Unlike previous lightweight transformers methods that create compact architecture from scratch and knowledge distillation with pre-training [25, 26, 29], our new architecture DLT can inherit the knowledge of existing pre-trained MDETR model. As a result, we remove the need to pre-train the Lite-MDETR on image-text dataset that contains >1M images. This saves us reasonable amount of computations and GPU hours as pre-training MDETR on 8 V100 GPUs takes several weeks [12]. Specifically, given the pre-trained MDETR, we initialize the dictionary, index and coefficient in DLT by factorizing corresponding weights in LT. Compared to low-rank factorization method, DLT has benefits on the sparsity of B, *i.e.* tiny dictionary weight D, index I, and coefficients C and cheap lookups. We introduce the switching method between sparse B and dictionary weight D, index I, and coefficients C in section 3.3.

Figure 3 compares the LT and DLT from a network connection view. As seen in the figure, LT has much more connections than DLT. Specifically, given input X with size of $n \times d$, and the weight W with size of $d \times d_W$, LT has almost $d \times d_W$ connections, where n is the token numbers, d is the hidden size in a transformer block. In contrast, DLT has fewer connections. Dictionary weight D has size of $d \times r$, both coefficient and index size is $t \times d_W$. Given the same index with size of $n \times d$, DLT has $(d \times r + t \times d_W)$ connections. One could dynamically tune dictionary size r and coefficient size t to control the trade-off between performance and model size. As a complement, Figure 5 introduces the difference of LT and DLT from a matrix view, and further illustrates DLT is more lightweight than LT.

Our Lite-MDETR based on DLT has two modes: (I). Inference Mode (II). Training Mode. This is because the lookups on dictionary is not differentiable, thus index I and coefficient C is only used during inference mode and practical deployment. During training mode, instead of index I and C, differentiable B is used with sparsity constraints. We present these two modes in the following section 3.2 and section 3.3, respectively.

3.2. Lite-MDETR Inference

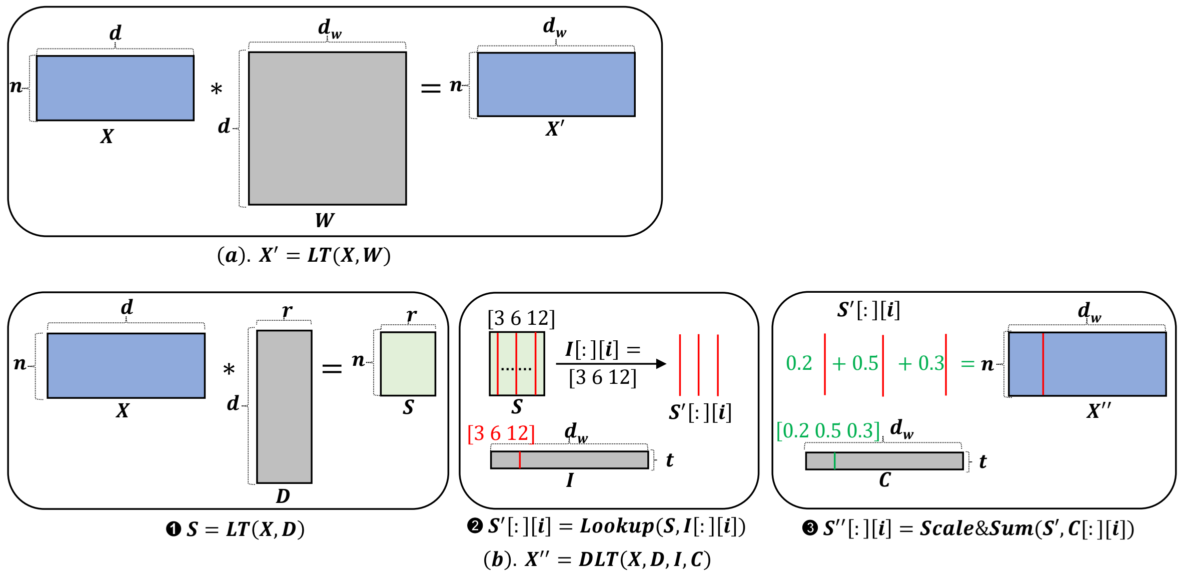
Figure 5 (a) and (b) show the computation processes of LT and DLT that are used in MDETR and our Lite-MDETR, respectively. As Figure 5(a) shows, $LT(X, W)$ is a matrix multiplication between an input X with size of $n \times d$ and a weight W with size of d_W, and the output is X' with size of $n \times d_W$. In contrast, Figure 5(b) represents the computation process of $DLT(X, D, I, C)$ that takes X as input, and generates X'' as output. Specifically, $DLT(X, D, I, C)$ has three steps: ❶ Generating dictionary with small linear projection $LT(X, D) = S$. A small $d \times r$ dictionary weight D is used to compress the input X with size of $n \times d$ as dictionary S with size of $n \times r$ by using $S = LT(X, D)$. Each column i of DLT output $X''[:][i]$ is a linear combination of t columns of dictionary S. This linear combination is performed by looking up $t \times d_W$ index matrix, scaling and summing coefficient matrix C. ❷ Lookups $S'[:][i] = Lookup(S, I[:][i])$. Each column of index matrix I stores the indices that show which columns in dictionary should be looked up for the following scaling and summation. For instance, given $t = 3$, the i-th column is $[3, 6, 12]$, the 3-rd, 6-th, and 12-th columns of dictionary S will be extracted as $S'[:][i]$. ❸ A few scaling and summation $Scale\&Sum(S', C[:][i]) = S''[:][i]$. After each lookups with $I[:][i]$, we get t-column $S'[:][i]$. The t coefficients in the i-th column of coefficient matrix $C[:][i]$ are used to scale with t-column $S'[:][i]$, respectively. And the scaled t columns are summed up in an element-wise manner to generate the i-th column of output of DLT, $X''[:][i]$. For example, given the i-th column of coefficient matrix $C[:][i] = [0.2, 0.5, 0.3]$, DLT scales them with corresponding extracted columns and sum the scaled vectors to generate the $X''[:][i]$. To generate all the X'', one should perform d_W times of lookups, scaling and summation.

Lite-MDETR based on DLT not only has fewer parameters, but also has fewer computations than MDETR based on LT during the practical inference and deployment. Our extensive experimental results in section 5 show that Lite-MDETR has smaller model size with similar accuracy with MDETR. Theoretically, LT requires $\mathcal{O}(d \times d_W)$ parameters and $\mathcal{O}(n \times d \times d_W)$ computation numbers (Multiplication and Addition). In contrast, DLT has $\mathcal{O}(d \times r + t \times d_W)$ parameters and $\mathcal{O}(n \times d \times r + t \times d_W)$ computation numbers and $\mathcal{O}(t \times d_W)$ lookups. In this paper, we set r in the range of $0.2d \sim 0.6d$, and t is around $0.2d$. $\mathcal{O}(t \times d_W)$ lookups on dictionary can be negligible since lookups are small-scale (t is tiny) and lookups are cheap (cache access operations) on both CPU and GPUs as shown in both existing works [1, 4] and our experiments.

3.3. Lite-MDETR Training

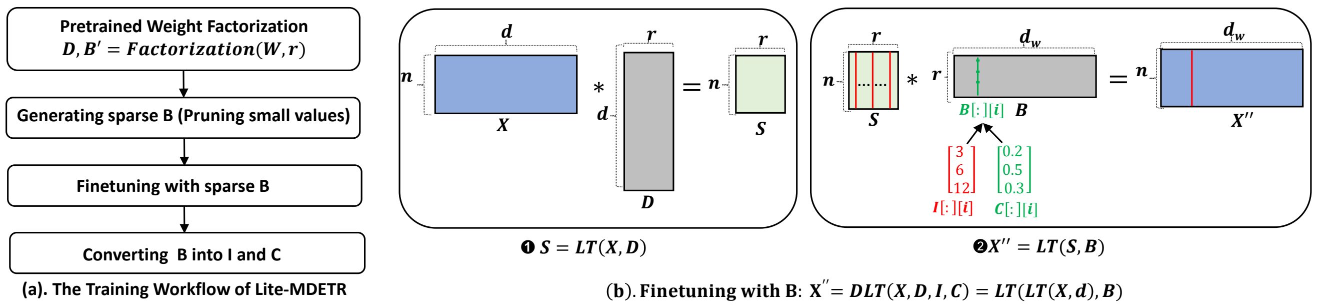
There are two challenges to train Lite-MDETR: (I). How to train DLT that has indifferentiable index matrix I? Directly training DLT requires to jointly train dictionary weight D, index matrix I, and coefficient C. However, index matrix I consists of non-continuous indices values and as a result it is not differentiable during training. (II). Given the pre-trained model MDETR, how to generate our Lite-MDETR without resource-prohibitive retraining from scratch? We show that we can avoid retraining from scratch by reusing the learned knowledge of pre-trained model MDETR. We propose a new Lite-MDETR training workflow shown in Figure 6(a) to address these two challenges.

1. Factorizing a Pre-trained Weight W. To avoid the generic pre-training on a large corpus, Lite-MDETR



(a). X′ = LT(X, W)
❶ S = LT(X, D)
❷ S′[:][i] = Lookup(S, I[:][i])
❸ S″[:][i] = Scale&Sum(S′, C[:][i])
(b). X″ = DLT(X, D, I, C)

Figure 5. A matrix view comparison of (a) LT layer and our (b) DLT inference layer. LT is an operation of applying linear projection W on input X. In contrast, DLT has three steps including smaller projection with dictionary weight D, looking up dictionary S with index I, scaling and summing with C.



Pretrained Weight Factorization
D, B′ = Factorization(W, r)
Generating sparse B (Pruning small values)
Finetuning with sparse B
Converting B into I and C
(a). The Training Workflow of Lite-MDETR
❶ S = LT(X, D)
❷X″ = LT(S, B)
(b). Finetuning with B: X″ = DLT(X, D, I, C) = LT(LT(X, d), B)

Figure 6. (a) The training workflow of Lite-MDETR. Given each pre-trained MDETR weight W, Lite-MDETR factorizes it into two parts: dictionary weight D and matrix B'. Then a sparsity B is generated by removing the small values in B'. Without pre-training on a large corpus, Lite-MDETR supports a direct fine-tuning with B on downstream tasks as shown in (b). After the fine-tuning step, sparse B is converted into storage-efficient I and C for an efficient inference.

enables dictionary weight and coefficients to inherit the knowledge of pre-trained weight W by factorizing W into dictionary weight and coefficients as shown in Equation 4. Given a pre-trained weight W with size of $d \times d_W$, we factorize it and generate a dictionary weight D with size of $d \times r$ and a matrix B' with size of $r \times d_W$ by using a SVD factorization [9, 22] where a weight W is approximated into U with size of $d \times r$, Σ with size of $r \times r$, V_F with size of $r \times d_W$ as shown in Equation 3. Lite-MDETR initializes the dictionary weight $D = U\Sigma$ and lets $B' = V_F^T$.

$$W \approx U \Sigma V_F^T \tag{3}$$

$$D, B' = Factorization(W, r) \approx U\Sigma, V_F^T \tag{4}$$

2. Generating a Sparse Matrix Weight B. Before fine-tuning on the downstream tasks, Lite-MDETR converts the B' generated by factorizing W to a sparse B. Equation 5 shows the conversion process where b', b is an entry in B' and B, respectively, and $value(t)$ is the $(t \times d_W)$-th largest value in B'. As formulated in Equation 5, any b' values larger than or equal to $value(t)$ are kept to B, otherwise the corresponding values b are zeros.

$$b = \begin{cases} b', & \text{if } |b'| >= value(t). \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

3. Fine-tuning with B. To work around indifferentiable index matrix I, we fine-tune the model with B on the downstream tasks. This is because the sparse matrix B can be converted to index matrix I and coefficient matrix C, and

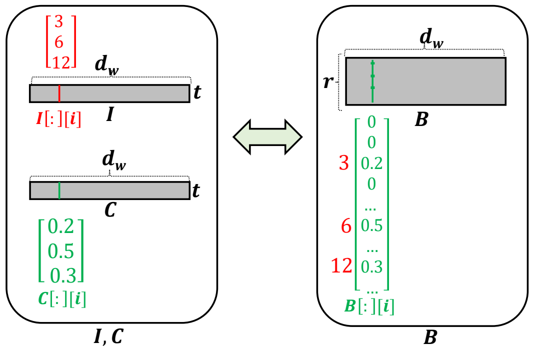


Figure 7. The switching operation between I, C and B. After fine-tuning of B, we convert it to I and C for smaller model size and efficient computation during the inference phase.

vice versa. Specifically, the fine-tuning of DLT has two steps: ❶ Generating dictionary with small linear projection $LT(X, D) = S$. This step is the same as the first step of inference mode. ❷ Multiplying dictionary S with sparse matrix B with size of $r \times d_W$ using $X'' = LT(S, B)$.

4. Converting B into I and C. During the deployment and inference phase, the sparse matrix B with $(r - t) \times d_W$ zeros wastes enormous memory and computation resources. To solve this issue, we convert B into dense and tiny I and C. Figure 7 shows the switching process between B and I, C. Given I and C with size of $t \times d_W$, one can easily generate B with size of $r \times d_W$ by assigning the values in indices of I with corresponding C and other values whose indices are not in index matrix as zeros. For example, given the i-th column of index matrix $I[:][i] = [3, 6, 12]$, and the i-th column of coefficient matrix $C[:][i] = [0.2, 0.5, 0.3]$, the 3-rd, 6-th, and 12-th value of the corresponding $B[:][i]$ is assigned $0.2, 0.5, 0.3$, respectively. The other values of $B[:][i]$ is assigned to zero. Inversely, given the sparse matrix B, one also could easily extract I and C.

4. Experimental Methodology

4.1. Datasets and Metrics

We perform experiments on referring expression comprehension and segmentation, phrase grounding, and visual question answering. We note that in all our experiments, we use the same training/validation/test splits and same evaluation metrics with MDETR [12].

Referring Expression Comprehension. We test Lite-MDETR on a number of referring expression benchmarks including RefCOCO, RefCOCOg, and RefCOCO+ [13,33]. The task in these datasets is to detect an object in an image referred by a language query. RefCOCO, RefCOCOg, and RefCOCO+ expressions have an average length of 3.61, 8.43, and 3.53 words. For evaluation metric, we compute the Jaccard overlap between predicted and ground truth bounding box. If it is above 0.5, the prediction is regarded as correct.

Phrase Grounding. Additionally, we test Lite-MDETR on phrase grounding benchmark, Flickr30K [24]. The task in Flickr30k is slightly different to referring expression. In this case, the goal is to detect all the objects in an image that is mentioned in a language query. We use the same metric with referring expression comprehension for evaluation.

Referring Expression Segmentation. Referring expression segmentation task takes the referring expression comprehension task one step further by segmenting the referred object. For this task, we use the PhraseCut dataset [30] that uses the images from the Visual Genome dataset [16]. For evaluation, we use the mean-IoU metric.

Visual Question Answering. Finally, we test Lite-MDETR on the visual question answering task using the GQA dataset [10]. For evaluation, we use the classification accuracy.

4.2. Implementation Details

Existing Multi-Modal Models. Other than MDETR [12], we also compare our Lite-MDETR with several multi-modal detectors, including VisualBERT [17], UNITER-L [3], VILLA-L [8] and TransVG [7] on referring expression comprehension (REC), referring expression segmentation (RES), and Phrase grounding (PG) tasks where both UNITER and VILLA use large model settings. For these tasks, MDETR uses ENB3 [28] (MDETR-ENB3) as a backbone whereas for visual question answering (VQA) tasks it adapts ENB5 (MDETR-ENB5) as a backbone to extract visual features. For a fair comparison, we use the same backbone and training settings with MDETR.

Lite-MDETR Architecture and Training. We follow the architecture settings of MDETR other than the replacement of the LT layers with DLT layers in Lite-MDETR. For the DLT layers, we study the dictionary size from $r = \lceil 0.2 d_W \rceil$ to $r = \lceil 0.6 d_W \rceil$ in the text encoder and $r = \lceil \frac{d_W}{3} \rceil$ in default to keep a good trade-off between accuracy and model size. We introduce the reasons why we choose this dictionary size and perform experiments with different dictionary size in section 5.1. Here, $\lceil x \rceil$ is an operation to generate an upper-bound integer of x. On the other hand, we set the coefficent matrix size t to $0.2r$. Finally, we follow the parameter settings in MDETR and train Lite-MDETR on 4 NVIDIA V100 GPUs.

Methodologies Study. We also compare our proposed work with baseline results with smaller model sizes, e,g, reducing the model depth by half. We trained an additional baseline with 5 text encoders and 3 encoder and decoder layers in the multi-modal transformer, resulting in a model with 270MB size. We call this baseline, **MDETR-ENB3-Half** in Table 1. The proposed DLT layers can be applied to the different components of the multi-modal detectors. We use **Lite-MDETR-T** to represent the method

Table 1. Comparisons with state-of-the-art multi-modal models on referring expression comprehension (REC) and segmentation (RES), and phrase grounding (PG) tasks. We report validation accuracy, mean-IoU, R@1 for REC, RES, and PG tasks. REC is tested on RefCOCO, RefCOCO+, and RefCOCOg datasets whereas RES and PG are tested on PhraseCut and Flickr30k datasets. We report PG performance using Flickr30K dataset with AnyBox protocol and MergeredBox protocols. Entries with - are not available.

		REC			PG		RES
	Model size	RefCOCO	RefCOCO+	RefCOCOg	Flickr-AnyBox	Flickr-MergedBox	PhraseCut
VisualBERT [17]	440MB	-	72.3	-	71.3	-	-
UNITER-L [3]	1212MB	81.4	75.9	74.9	-	-	-
VILLA-L [8]	1212MB	82.4	76.2	76.2	-	-	-
TransVG [7]	430MB	80.3	63.5	66.6	78.4	-	-
MDETR-ENB3 [12]	450MB	87.5	81.1	83.4	82.9	82.3	53.7
MDETR-ENB3-Half	270MB	84.0	79.2	80.2	-	-	-
SVD-MDETR-TTQ	404MB	83.8	79.3	79.6	79.8	79.5	49.6
Lite-MDETR-T	261MB	85.4	80.8	81.1	80.2	80.6	50.3
Lite-MDETR-TT	215MB	85.1	80.5	80.6	79.7	79.8	49.7
Lite-MDETR-TTQ	110MB	85.1	80.1	79.8	79.3	79.1	49.3

that compresses only the text encoder in MDETR. We also apply Lite-MDETR on both text encoder and multi-modal transformer of MDETR and call this model **Lite-MDETR-TT**. In addition, we also show that our Lite-MDETR can be combined with the orthogonal quantization method [14] without any modification. We name this combination as **Lite-MDETR-TTQ**. The current quantization bit width is chosen as 16 bits that shows that low-precision model compression can be used. Various quantization bits can be explored but it is outside of the scope of this study. Finally, we design a baseline method that uses SVD matrix factorization [9, 22] to MDETR-TTQ, and call it **SVD-MDETR-TTQ**. For fair comparison, we let the dictionary size r in our DLT as the rank of SVD factorization.

Model Size. In this study, we omit the word embedding lookup table from the model size since the embedding lookup table entries are highly dependent on the downstream tasks that often have different vocabulary size. More details can be found in Appendix.

5. Results

5.1. Referring Expression and Phrase Grounding

In Table 1 we show the comparisons of Lite-MDETR and state-of-the-art multi-modal transformers on REC, RES, and PG tasks. As shown in the table, MDETR-ENB3 achieves state-of-the-art results on REC and RES tasks. Since the text encoder in MDETR-ENB3 occupies $\sim 83\%$ model size, Lite-MDETR-T that only compresses the text encoder can still significantly reduce the MDETR-ENB3 model size from 450 MB to 261 MB, achieving 1.72× model size reduction. When tested on RefCOCO, RefCOCOg and RefCOCO+, Lite-MDETR-T attains $3\% - 4\%$ higher accuracy than UNITER-L and VILLA-L that have $4 - 5\times$ larger model size. Finally, Lite-MDETR-T outperforms a recent visual grounding model, TransVG, that has similar architecture to the baseline MDETR by $4 - 13\%$ with $> 2\times$ smaller model.

Based on Lite-MDETR-T, Lite-MDETR-TT further applies our DLT module on the multi-modal transformer. Unlike the text encoder where we used $r = \lceil \frac{d_W}{3} \rceil$, the multi-modal transformer is not the bottleneck of model size and we notice that the multi-modal transformer is more sensitive to dictionary ratio than the text encoder [15]. As a result, we use a larger dictionary size r, *i.e.*, $r = \lceil \frac{d_W}{2} \rceil$ for the multi-modal transformer to keep a better trade-off between total model size and accuracy. With a larger dictionary size r, Lite-MDETR-TT further reduces $\sim 46MB$ model size to $215MB$ with $0.3 - 0.5\%$ accuracy loss on REC tasks. As we show in Table 1, an orthogonal quantization method [14] can be jointly used with our Lite-MDETR-TT to design an even lighter MDETR. With 16-bit quantization on all parts of Lite-MDETR-TT including the vision encoder, a lightweight multi-modal detector Lite-MDETR-TTQ with $110MB$ can be obtained. Lite-MDETR-TTQ attains 85.1%, 80.1%, 79.8% validation accuracy on RefCOCO, RefCOCO+, and RefCOCOg that is still about

Table 2. Comparisons with state-of-the-art MDETR [12] on VQA task. Our Lite-MDETR achieves drastically smaller model size than MDETR without suffering a significant accuracy loss. MDETR adopts ENB5 as a backbone to extract visual features.

		VQA	
	Model size	Test-dev	Test-std
MDETR-ENB5 [12]	490MB	62.95	62.45
Lite-MDETR-T	300MB	60.2	60.3
Lite-MDETR-TT	255MB	59.9	60.1
Lite-MDETR-TTQ	130MB	59.6	59.7

Table 3. Experiments on Lite-MDETR architecture with different dictionary ratios on REC, PG, and RES tasks. We define dictionary ratio as the ratio of dictionary size to weight size d_W. Dictionary ratio decides the architecture of our Lite-MDETR and helps us control the trade-off between model size and accuracy. Entries with ± represent the averages across three runs.

		REC			PG		RES
Dictionary ratio	Model size	RefCOCO	RefCOCO+	RefCOCOg	Flickr-AnyBox	Flickr-MergedBox	PhraseCut
20%	80MB	83.2 ±0.3	78.3 ±0.2	77.6 ±0.3	78.1 ±0.2	75.6 ±0.2	46.5±0.3
33%	110MB	85.1±0.3	80.1±0.3	79.8±0.3	79.3±0.2	79.1±0.2	49.3±0.2
45%	137MB	85.3±0.1	80.2±0.2	79.8±0.2	80.1±0.1	80.6±0.2	50.2±0.3
60%	171MB	85.4±0.3	80.5±0.4	80.2±0.2	80.3±0.2	80.7±0.3	50.3±0.2

$2.7\% - 3.9\%$ higher than the accuracy of VILLA-L and has a 3.81%, 2.23%, 4.15% accuracy loss over MDETR that has $4.1\times$ larger model size. Lite-MDETR-TT with 215MB model size, achieves $\sim 1.1\%$ higher accuracy than MDETR-ENB3-Half.

On the other hand, our baseline, SVD-MDETR-TTQ with r ranks, slightly reduces the model size of MDETR. To be more specific, SVD-MDETR-TTQ still has $404MB$ model size while achieving similar accuracy to the Lite-MDETR-TTQ with $110MB$ model size. By comparing the training workflows of SVD-MDETR-TTQ and our Lite-MDETR-TTQ, we can observe that SVD-MDETR-TTQ lacks two important steps: 1. Generating sparse matrix B and 2. converting sparse matrix into dense index and coefficient matrices, *i.e.*, I and C. A sparse B matrix can be used to reduce the parameters redundancy without hurting the performance. Converting B into I and C can help remove the zeros in the sparse matrix. Therefore, we believe that generating sparse matrix B and converting it into dense index and coefficient matrices are the main reasons why our Lite-MDETR-TTQ can generate more lightweight models than SVD-MDETR-TTQ.

5.2. Visual Question Answering

Next, in Table 2 we show the comparisons of our Lite-MDETR and MDETR-ENB5 [12] on VGA tasks. MDETR-ENB5 [12] with model size $490MB$ adopts ENB5 [28] as a vision encoder backbone. Based on MDETR-ENB5, Lite-MDETR-T replaces the LT layers in the text encoder with the light-weight DLT layers, reducing the model size to $\sim 300MB$. On the other hand, Lite-MDETR-TT compresses the model size to $255MB$ from $300MB$ by further replacing the LT layers with DLT layers in multi-modal transformer of Lite-MDETR-T. We highlight that similar to our settings in section 5.1 DLT layers in multi-modal transformer use a larger dictionary size than the DLT layers in text encoder. We justify the reasons behind this choice in section 5.1. Finally, with orthogonal quantization we reduce the Lite-MDETR-TT model size by $\sim 2\times$ with only 0.3% loss in accuracy. Overall, we reduce the MDETR-ENB5 size from $490MB$ to $130MB$ with only 3.3% loss in accuracy.

5.3. Lite-MDETR-TTQ Architecture Study

Finally, in Table 3 we investigate the Lite-MDETR-TTQ architecture with various dictionary ratios. Dictionary ratio defines the ratio of dictionary size to weight size d_W. For this experiments, we fix the dictionary size in multi-modal transformer as $r = \lceil \frac{d_W}{2} \rceil$ and only tune the dictionary ratio in the text encoder from 20% to 60%. There are two reasons to use a large and fixed dictionary ratio in the multi-model transformer. First, unlike text encoder, the multi-modal transformer is not the bottleneck of model size. Additionally, we notice that the multi-modal transformer model is more sensitive to the accuracy than the text encoder [15]. For these reasons, we fix the dictionary size in multi-modal transformer and only tune the dictionary ratio in the text encoder. As shown in Table 3, a larger dictionary ratio means a larger dictionary size which in turn produces larger model size. As expected, this larger model achieves higher accuracy. In particular, when dictionary ratios are 20% and 60%, Lite-MDETR-TTQ has $80MB$ and $171MB$ model size and these models achieve 83.2% and 85.4% accuracy on RefCOCO. To sum up, our Lite-MDETR-TTQ enables users to pick up architectures that are deployed on mobile devices according to the specifications of hardware resources. In this direction, we can simply tune the dictionary to control the trade-off between model size and performance.

6. Conclusion

We proposed Lite-MDETR, an efficient vision and language model, based on the MDETR architecture. To get Lite-MDETR, we designed a dictionary lookup transformation (DLT) layer to replace the heavy linear transformation layer (LT) in the text encoder and multi-modal transformer in MDETR. We first factorize the LT weights using SVD and learn the sparse matrix in the training time. In inference mode, to improve efficiency we use the sparse matrix as a lookup table to get dictionary indexes and coefficients to perform efficient linear transformation. We perform experiments on vision and language tasks including referring expression comprehension and segmentation, phrase grounding and VQA and we show that our Lite-MDETR reduces MDETR's size drastically with minor loss in the accuracy.

References

[1] Arturo Argueta and David Chiang. Accelerating sparse matrix operations in neural networks on graphics processing units. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6215–6224, Florence, Italy, July 2019. Association for Computational Linguistics. 4

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020. 2

[3] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. Uniter: Universal image-text representation learning. In *ECCV*, 2020. 1, 6, 7

[4] Kyoshin Choo, William Panlener, and Byunghyun Jang. Understanding and optimizing gpu cache memory performance for compute workloads. In *2014 IEEE 13th International Symposium on Parallel and Distributed Computing*, pages 189–196, 2014. 4

[5] Xiyang Dai, Yinpeng Chen, Jianwei Yang, Pengchuan Zhang, Lu Yuan, and Lei Zhang. Dynamic detr: End-to-end object detection with dynamic attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2988–2997, 2021. 2

[6] Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. Up-detr: Unsupervised pre-training for object detection with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1601–1610, 2021. 2

[7] Jiajun Deng, Zhengyuan Yang, Tianlang Chen, Wengang Zhou, and Houqiang Li. Transvg: End-to-end visual grounding with transformers. *arXiv preprint arXiv:2104.08541*, 2021. 1, 2, 6, 7

[8] Zhe Gan, Yen-Chun Chen, Linjie Li, Chen Zhu, Yu Cheng, and Jingjing Liu. Large-scale adversarial training for vision-and-language representation learning. In *NeurIPS*, 2020. 6, 7

[9] Gene H Golub and Christian Reinsch. Singular value decomposition and least squares solutions. In *Linear algebra*, pages 134–151. Springer, 1971. 2, 5, 7

[10] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709, 2019. 6

[11] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019. 2

[12] Aishwarya Kamath, Mannat Singh, Yann LeCun, Ishan Misra, Gabriel Synnaeve, and Nicolas Carion. Mdetr–modulated detection for end-to-end multi-modal understanding. *arXiv preprint arXiv:2104.12763*, 2021. 1, 2, 4, 6, 7, 8

[13] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. Referitgame: Referring to objects in photographs of natural scenes. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 787–798, 2014. 6

[14] Daya Khudia, Jianyu Huang, Protonu Basu, Summer Deng, Haixin Liu, Jongsoo Park, and Mikhail Smelyanskiy. Fbgemm: Enabling high-performance low-precision deep learning inference. *arXiv preprint arXiv:2101.05615*, 2021. 2, 7

[15] Wonjae Kim, Bokyung Son, and Ildoo Kim. Vilt: Vision-and-language transformer without convolution or region supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 5583–5594. PMLR, 2021. 7, 8

[16] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *arXiv preprint arXiv:1602.07332*, 2016. 6

[17] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. What does BERT with vision look at? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5265–5275, Online, July 2020. Association for Computational Linguistics. 6, 7

[18] Muchen Li and Leonid Sigal. Referring transformer: A one-step approach to multi-task visual grounding. *arXiv preprint arXiv:2106.03089*, 2021. 2

[19] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European conference on computer vision (ECCV)*, pages 19–34, 2018. 1

[20] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. 1, 3

[21] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *arXiv preprint arXiv:1908.02265*, 2019. 1

[22] Matan Ben Noach and Yoav Goldberg. Compressing pre-trained language models by matrix decomposition. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 884–889, 2020. 2, 5, 7

[23] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019. 2

[24] Bryan A. Plummer, Liwei Wang, Christopher M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazeb-

nik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. *IJCV*, 123(1):74–93, 2017. 6

[25] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019. 1, 2, 4

[26] Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for bert model compression. *arXiv preprint arXiv:1908.09355*, 2019. 1, 2, 4

[27] Zhiqing Sun, Shengcao Cao, Yiming Yang, and Kris M Kitani. Rethinking transformer-based set prediction for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3611–3620, 2021. 2

[28] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019. 3, 6, 8

[29] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *arXiv preprint arXiv:2002.10957*, 2020. 1, 2, 4

[30] Chenyun Wu, Zhe Lin, Scott Cohen, Trung Bui, and Subhransu Maji. Phrasecut: Language-based image segmentation in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10216–10225, 2020. 6

[31] Zhanghao Wu, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han. Lite transformer with long-short range attention. In *International Conference on Learning Representations*, 2020. 1

[32] Zhengyuan Yang, Yijuan Lu, Jianfeng Wang, Xi Yin, Dinei Florencio, Lijuan Wang, Cha Zhang, Lei Zhang, and Jiebo Luo. Tap: Text-aware pre-training for text-vqa and text-caption. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8751–8761, 2021. 1

[33] Licheng Yu, Patrick Poirson, Shan Yang, Alexander C Berg, and Tamara L Berg. Modeling context in referring expressions. In *European Conference on Computer Vision*, pages 69–85. Springer, 2016. 6

[34] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020. 2