Tensorized Attention for Understanding Multi-Object Relationships

Makoto Nakatsuji[1], Yasuhiro Fujiwara[2], Atsushi Otsuka[1], Narichika Nomoto[1], Yoshihide Sato[1]

[1]NTT Human Informatics Laboratories,
[2]NTT Communication Science Laboratories
makoto.nakatsuji@ntt.com

Abstract

Attention mechanisms have played a crucial role in the success of Transformer models, as seen in platforms like ChatGPT. However, since they compute attentions from relationships between only one or two object types, they fail to effectively capture multi-object relationships in real-world scenarios, resulting in low prediction accuracy. In fact, they cannot calculate attention weights among diverse object types, such as the 'comments,' 'replies,' and 'subjects' that naturally constitute conversations on platforms like Reddit or X, although their relationships are simultaneously observed in real-world contexts. To overcome this limitation, we introduce the Tensorized Attention Model (TAM), which leverages the Tucker decomposition to calculate attention weights across various object types and seamlessly integrates them into the Transformer models. Evaluations show that TAM significantly outperforms existing encoder methods, and its integration into the LoRA adapter for Llama2 enhances fine-tuning accuracy.

Introduction

The Transformer architecture (Vaswani et al. 2017) is a popular attention mechanism that has achieved extraordinary success in a wide range of natural language processing (NLP) tasks. In particular, BERT (Devlin et al. 2019) is a model utilizing attention-based mechanisms and pretrained on massive datasets. Its introduction marked a significant advancement in attention-centric models, such as OpenAI's ChatGPT (OpenAI 2023), Google's Gemini (Geminiteam et al. 2023), and Meta's LLaMA3 (Meta 2024). These models have contributed to the evolution of attention mechanisms in NLP language models.

Current attention-based models typically focus on relationships involving only one or two object types, like query and memory objects, as seen in self-attention or source-target attention mechanisms (Vaswani et al. 2017). However, real-world interactions often involve multi-object relationships, connecting three or more object types. For instance, two connected sentences in a Wikipedia article typically share a common topic, geo-tagged tweets may trigger replies related to specific locations, and multiple utterances on Reddit can stem from a shared conversational context. Incorporating additional information from such diverse object types into current two-dimensional attention mechanisms is crucial for unlocking their latent capabilities.

An effective representation of such multi-object relationships can be achieved using tensors (Nakatsuji et al. 2016). However, existing attention-based methods fail to utilize tensors to represent multi-object relationships. For example, (Ma et al. 2019) introduced a Tensorized Transformer model that incorporates a specialized self-attention encoder layer along with the block-term tensor decomposition (BTD) technique (De Lathauwer 2008). This approach alleviates the computational burden by compressing the extensive parameter set used in multi-head attention into a set of third-order tensors through a low-rank approximation. Their study, which focuses specifically on self-attention, however, did not address handling relationships among more than two types of object when computing the attention weights.

Even if we use the Tensorized Transformer (Ma et al. 2019) for attention computations among multiple object types, we still face two limitations. First, the approach imposes a restriction that the side lengths of the core tensor must match the dimension size of the input query. This is because it analyzes the relationships among the query, key, and value components by compressing them within the core tensor, which must be cubic. As a result, when predicting the attention output based on the query length using a cubic tensor of this dimension size, it fails to exploit the correspondences between tokens in the query stream and those in the attention output, leading to reduced training accuracy. Second, this approach necessitates directly computing the transformer output from attention weight tensors. Therefore, it could not obtain transformations from source objects to target objects, reducing its versatility.

This paper introduces the Tensorized Attention Model (TAM). It addresses these limitations by extending the Tensorized Transformer (Ma et al. 2019) to incorporate attention weights among multi-object relationships. It introduces a third object type, termed "semantic objects," to complement the traditional query and memory objects. Specifically, TAM modifies the existing Tensorized Transformer in two key ways. First, TAM employs the Tucker decomposition with a core tensor, where one side corresponds to the query length (i.e., the explicit token length), and the other two sides correspond to the hidden dimension sizes in the key and semantic components. This approach aggregates infor-

mation from memory and semantic components, aligning it with the query token sequence. This alignment enables explicit updates to the query and smooth transitions to the next layer. Second, TAM computes multi-dimensional attentions among the query, key (derived from memory), and semantic components, facilitating the learning of transformations from the source (value from memory) to the target (query).

Since our approach introduces semantic objects, it effectively captures multi-object relationships in complex real-world scenarios. In conventional 2D attention models, the query utilizes keys to determine where to pull information from memory (Vaswani et al. 2017). In contrast, TAM adopts a multi-dimensional framework by incorporating semantic objects into the Tensorized Transformer. Since semantic objects capture essential aspects of multi-object relationships, they facilitate a seamless connection between the query and key. Consequently, TAM can capture multi-dimensional relationships effectively and understand "concurrent occurrences" among objects accurately. For example, TAM can effectively model a Reddit comment and its replies by considering the shared title as a semantic object. Similarly, in X discussions, it can use a specific location shared between a tweet and its replies as a semantic object to improve accuracy.

To showcase its effectiveness, we integrated TAM into the Transformer encoder and evaluated its performance in response selection and question-answering tasks. Our experiments followed two paths: training the TAM model from scratch and augmenting it with pre-trained Transformer models. When tested with NFL and Politics datasets compiled from Reddit as well as the TweetQA dataset (Xiong et al. 2019), TAM consistently outperformed existing Transformer-based methods in accuracy. We also integrated TAM into the LoRA adapter (Hu et al. 2022) for fine-tuning large language model (LLM), Llama2 (Touvron et al. 2023). In evaluations on Reddit and SQuAD (Rajpurkar et al. 2016) datasets, we observed that TAM enhances accuracy in both response and answer generation tasks.

Related Work

Several studies have explored tensor decomposition within transformer architectures to capture complex relationships for parameter compression (Bilgin et al. 2022; Shen et al. 2019; Panahi, Saeedi, and Arodz 2021; Ye et al. 2020; Hawkins, Liu, and Zhang 2022). However, despite the efforts to reduce parameters, the experimental results have indicated a decrease in accuracy (Shen et al. 2019; Hawkins, Liu, and Zhang 2022). Further, these techniques have not been applied to Transformer attention mechanisms.

In the multimodal research (Lu et al. 2019; Zhu and Yang 2020; Chefer, Gur, and Wolf 2021), which uses both visual and textual inputs, ViLBERT (Lu et al. 2019) introduced a co-attentional transformer layer, where the key-value pairs from one source are passed to the other source's attention block to act as the new key-value pairs. ActBERT (Zhu and Yang 2020) enhances ViLBERT by encoding three objects; action, regional, and linguistic features. It first blends action features from linguistic ones and guides action features from regional ones. It then computes source-target attentions from these blended or guided features to each target feature. In contrast, TAM emphasizes the simultaneous observations of three distinct features.

Preliminary

Let us now explain the notation used in this paper and introduce Tensorized Transformer (Ma et al. 2019).

We use Euler script letter $\mathcal{A}$ to denote a third-order tensor, which can be thought of as a multi-array extending the concept of a matrix to three dimensions. This paper will use third-order tensors for simplicity. However, it is worth noting that this approach can be extended to higher-dimensional tensors. In this specific context, an element in a third-order tensor is denoted as $\mathcal{A}_{d_1,d_2,d_3}$. We use ":" as a dummy index to fix certain dimensions in the tensor, while leaving variable a face composed of the remaining dimensions; e.g. $\mathcal{A}_{d_1,d_2,:}$ represents a vector with the first and second dimensions fixed and the third dimension left variable.

Tensorized Transformer assumes that the query, key, and value can be mapped into sets of three orthogonal basis vectors. Each factor matrix ($\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$) has dimensions of $N \times D$, where N represents the sequence length and D represents the dimensionality of the matrix. It initializes a core tensor $\mathcal{G}$ of rank R, where R is typically set to D in practice (Ma et al. 2019). The attention weight tensor $\mathcal{A}(\mathcal{G};\mathbf{Q},\mathbf{K},\mathbf{V})$, which stores attention weights among the query, key, and value as its elements, is defined as:

$$\mathcal{A}= \sum_{i,j,l=1}^{D} \mathcal{G}_{i,j,l} \cdot (\mathbf{Q}_{:,i} \odot \mathbf{K}_{:,j} \odot \mathbf{V}_{:,l})_{:,i,:,j,:,l} \qquad (1)$$

The symbol $\odot$ represents the outer product. $\mathbf{Q}_{:,i}$, $\mathbf{K}_{:,j}$, and $\mathbf{V}_{:,l}$ are column vectors extracted from $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$, respectively. $\mathcal{G}$ forms a weight vector $\mathbf{g}$ with trainable elements g_r along its diagonal that are computed using the softmax function such that $\sum_{r=1}^{R} g_r = 1$. The size of the core tensor $\mathcal{G}$ is set to $\mathbb{R}^{D\times D\times D}$, and while the column vector $\mathbf{Q}_{:,i}$ corresponds to the query length Q, the relationships among $\mathbf{Q}, \mathbf{K}$, and $\mathbf{V}$ are maintained within the latent space of dimension D. This setup aligns these relationships to the dimension D (rather than the query-length side Q).

The Tensorized Transformer employs block-term tensor decomposition (De Lathauwer 2008) to construct its multi-head attention mechanism. Ultimately, it computes the attention output matrix $\mathbf{O}(\mathcal{G};\mathbf{Q},\mathbf{K},\mathbf{V})$ as follows:

$$\mathbf{O}= \text{Split\&Concat}\left(\frac{\mathcal{T}_1 + \ldots \mathcal{T}_h \ldots + \mathcal{T}_H}{H}\right)\mathbf{W}^O \qquad (2)$$

$$\text{s.t.} \quad \mathcal{T}_h = \mathcal{A}(\mathcal{G}_h; \mathbf{Q}\mathbf{W}^q, \mathbf{K}\mathbf{W}^k, \mathbf{V}\mathbf{W}^v)$$

This equation computes the attention output matrix $\mathbf{O}(\mathcal{G};\mathbf{Q},\mathbf{K},\mathbf{V})$ directly from the attention weight tensors, $\mathcal{A}$s, by applying a linear function. Split&Concat$(\cdot)$ is a function which achieves the concatenation after splitting for a third-order tensor. $\mathbf{W}^O$ is a parameter matrix which is a fully connected layer and is correlated to the attention output. $\mathbf{W}^q$, $\mathbf{W}^k$, and $\mathbf{W}^v$ are parameter matrices that are used to adjust the dimension sizes of $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$. They are shared by Tensorized Transformer when multi-core tensors are constructed.

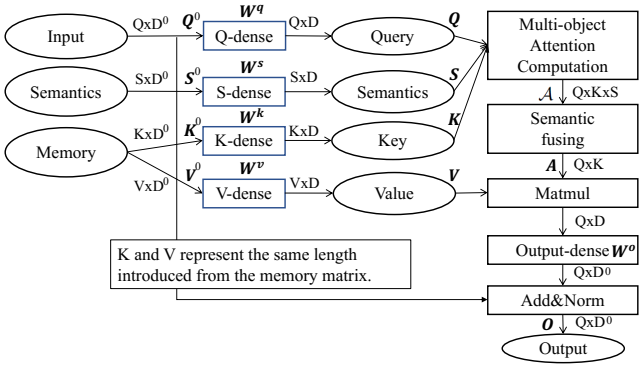


Figure 1: The attention architecture of the TAM.

Proposed Method

This section explains the details of TAM.

Basic Idea and our Attention Architecture

First, we introduce our idea: a novel multi-object attention mechanism that captures real-world relationships. Conventional attention mechanisms employ a query ($\mathbf{Q}$), a key from memory ($\mathbf{K}$), and a value from memory ($\mathbf{V}$). Our approach introduces an additional component, "semantics" ($\mathbf{S}$), which is distinct from the existing $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ components to represent multi-object relationships. Specifically, our tensorized attention framework ensures that $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{S}$ are distinct from each other, while $\mathbf{K}$ and $\mathbf{V}$ remain the same as they come from the same source, memory. This approach enables our attention mechanism to capture complex dependencies and interactions effectively by leveraging the semantic information encapsulated in $\mathbf{S}$. As a result, TAM accurately models natural relationships in real-world contexts through multi-object attention computation, as described next.

The architecture is depicted in Fig. 1. Unlike conventional attention models, TAM takes not only query and memory components, but also semantic components as inputs. The query, key, value, and semantic embeddings ($\mathbf{Q}^0$, $\mathbf{K}^0$, $\mathbf{V}^0$, and $\mathbf{S}^0$) with dimension size D^0 are mapped to their respective embeddings ($\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$, and $\mathbf{S}$) with size D using dense layers with weight matrices $\mathbf{W}^q$, $\mathbf{W}^k$, $\mathbf{W}^v$, and $\mathbf{W}^s$. Following the dense networks, the model calculates a three-dimensional attention tensor $\mathcal{A}$ relating the query matrix $\mathbf{Q}$, semantics matrix $\mathbf{S}$, and key matrix $\mathbf{K}$ by using a multi-object attention computation. To incorporate these three-dimensional attention weights into the value matrix $\mathbf{V}$, it converts $\mathcal{A}$ into a two-dimensional matrix $\mathbf{A}$ through semantic fusion, which involves averaging the $q \times k$ matrices along the semantic direction. Subsequently, matrix multiplication (Matmul) is performed on $\mathbf{A}$ and $\mathbf{V}$, followed by application of Add&Norm (Vaswani et al. 2017) to the updated value and the original query through a dense network. Finally, TAM produces the attention output $\mathbf{O}$.

Multi-object Attention Computation

TAM computes three-dimensional attention weights among $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{S}$ by using the Tucker decomposition. The three-dimensional attention weights are then reflected in the attention output $\mathbf{O}$ through semantic-fused 2D attention.

Unlike the Tensorized Transformer (Ma et al. 2019), which relates the query and memory components in a fixed latent space, TAM introduces a transformative approach that aligns the query, semantic, and key components with the query token sequence. To achieve this, we extend Eq. (1) to Eq. (3), introducing Multi-object Attention Computation:

$$\mathcal{A}=\sum_{i,j,l=1}^{D}\mathcal{G}_{q,j,l}\cdot\left(\mathbf{Q}_{q,:}\odot\mathbf{K}_{:,j}\odot\mathbf{S}_{:,l}\right)_{q,:,:,j,:,l} \tag{3}$$

This equation represents q as the row vector $\mathbf{Q}_{q,:}$ from the query matrix $\mathbf{Q}$ and uses the column vectors $\mathbf{K}_{:,j}$ and $\mathbf{S}_{:,l}$ extracted from the matrices $\mathbf{K}$ and $\mathbf{S}$, respectively. Eq. (3) can be implemented by the torch einsum function like: einsum('qjl, qi, kj, sl -> qks', [core, query, key, semantic]).

The size of the core tensor $\mathcal{G}$ is set to $\mathbb{R}^{Q\times D\times D}$, where the length Q in $\mathbf{Q}$ is retained as the side length of the core tensor. This novel approach enables an *alignment centered around* Q for its relationships with $\mathbf{K}$ and $\mathbf{S}$ when computing the outer product among the three matrices and performing summation along the D dimension with respect to $\mathbf{Q}$. In this context, Eq. (3) activates the first, fourth, and sixth dimensions in $(\mathbf{Q}_{q,:}\odot\mathbf{K}_{:,j}\odot\mathbf{S}_{:,l})$ during the computation of the outer product $(\mathbf{Q}_{q,:}\odot\mathbf{K}_{:,j}\odot\mathbf{S}_{:,l})_{q,:,:,j,:,l}$. A single core tensor $\mathcal{G}$ has diagonal elements g_ds forming a D-length trainable weight vector $\mathbf{g}$ and is initialized as

$$\mathcal{G}_{q,j,l}=\begin{cases}0 & \text{if } q\neq j \text{ or } q\neq l \text{ or } j\neq l \text{ or } D<q\\ g_d & \text{if } q=j=l=d\end{cases} \tag{4}$$

The diagonal element g_d is computed using a softmax function such that $\sum_{d=1}^{D} g_d = 1$. If $D < Q$, then the elements where $q > D$ become zero.

The previous study (Ma et al. 2019) employed a method called "split&concat," which splits the three-dimensional attention obtained by Tucker decomposition along the V-axis and then concatenates them to form a 2D-attention matrix of size $\mathbb{R}^{Q\times(V\cdot K)}$. This 2D matrix is then transformed into the transformer output via a linear layer. However, this method diverges from the Transformer, particularly when handling transformations from one type of object to another. Therefore, it fails to effectively learn the transformations that the Transformer adeptly handles, where $\mathbf{V}$ is transformed using attention output and seamlessly integrated with $\mathbf{Q}$.

To overcome this limitation, TAM introduces Semantic Fusion, which merges the semantic axis of the multi-dimensional attention tensor into a 2D matrix $\mathbf{A}\in\mathbb{R}^{Q\times K}$ by re-expressing Eq. (3) as follows:

$$\mathbf{A}=\frac{1}{S}\sum_{s=1}^{S}\left[\sum_{i,j,l=1}^{D}\mathcal{G}_{q,j,l}\cdot\left(\mathbf{Q}_{q,:}\odot\mathbf{K}_{:,j}\odot\mathbf{S}_{:,l}\right)_{q,:,:,j,:,l}\right]_{:,:,s}.$$

Thus, TAM calculates the 2D attention between $\mathbf{Q}$ and $\mathbf{K}$ on the basis of a semantically rich 3D attention tensor. Moreover, by computing the inner-product between $\mathbf{A}_{:,k}$ and $\mathbf{V}$, the model can capture the transformation induced by the attention mechanism directly within $\mathbf{V}$ itself. Finally, the

residual connection is performed by an 'Add&Norm' operation with the parameter matrix $\mathbf{W}^O$ to generate the final output $\mathbf{O}$ as $\mathbf{O} = \text{Add\&Norm}((\mathbf{A}_{:,k}\mathbf{V}) \cdot \mathbf{W}^O, \mathbf{Q})$, as is done in (Vaswani et al. 2017).

Training TAM as an Encoder Model

TAM pretrains the masked language model (MLM) and next segment prediction (NSP) using input, memory, and semantic components, optimized with cross-entropy loss. For fine-tuning on response selection, TAM uses the hidden vector $\mathbf{O}_{cls}$ from the first cls token in $\mathbf{O}$ and computes the matching score between the utterance context and response using a single-layer neural network, $\sigma(\mathbf{W} \cdot \mathbf{O}_{cls})$, where $\mathbf{W}$ is a trainable parameter. The model is optimized with cross-entropy loss.

TAM effectively handles real-world relationships in multiple dimensions, and can also be combined with pre-trained language models based on conventional 2D-attention mechanisms like BERT. We can realize this integration by feeding the output from the pre-trained language model, along with the corresponding semantic information, into TAM as inputs. Thereby, TAM can also serve as a pre-trained model, extending its utility across various applications.

TAM as an LLM Adapter

It is important to demonstrate the TAM's applicability to LLMs like Llama2. In particular, since it needs high computational cost in pre-training LLMs, we applied TAM to the LoRA adapter model (Hu et al. 2022). LoRA freezes the pre-trained model weights and introduces trainable rank decomposition matrices into each layer of the Transformer architecture, which significantly reduces the number of trainable parameters for downstream tasks.

The procedure of our LoRA-TAM adapter is as follows: (1) compress the LLM's input embeddings and semantic embedding $\mathbf{Q}^l$, $\mathbf{K}^l$, $\mathbf{V}^l$, and $\mathbf{S}^l$, which have dimension size D^l, into $\mathbf{Q}^0$, $\mathbf{K}^0$, $\mathbf{V}^0$, and $\mathbf{S}^0$ with smaller dimension size D^0 using the weight matrix $\mathbf{A}$ of the LoRA adapter (Hu et al. 2022); (2) perform TAM calculations to facilitate the reflection of relationships in simultaneous co-occurences in the input and semantic streams in the multi-dimensional attention model; (3) restore the output embedding of TAM to the original number of dimensions by using the weight matrix $\mathbf{B}$ of the LoRA adapter and passing it to the attention layer of the LLM.

In the above procedure, the output embedding of LoRA-TAM adapter, $\mathbf{H}$, is expressed as follows:

$$\begin{aligned}\mathbf{H} &= \mathbf{W}_0 \cdot \mathbf{Q}^l + \Delta\mathbf{W} \cdot \mathbf{Q}^l \\ &= \mathbf{W}_0 \cdot \mathbf{Q}^l + \mathbf{B} \cdot \text{TAM}(\mathbf{A}\mathbf{Q}^l, \mathbf{A}\mathbf{K}^l, \mathbf{A}\mathbf{V}^l, \mathbf{A}\mathbf{S}^l).\end{aligned} \tag{5}$$

Here, the function TAM represents the TAM model, which takes $\mathbf{Q}^0$, $\mathbf{K}^0{=}\mathbf{V}^0$, and $\mathbf{S}^0$ as inputs and outputs $\mathbf{O}$ (see Fig. 1). $\mathbf{W}_0 \in \mathbb{R}^{D^l \times Q}$ is a pre-trained weight matrix and its update is represented by a low-rank decomposition as follows:

$$\mathbf{W}_0 + \Delta\mathbf{W} = \mathbf{W}_0 + \mathbf{B}\mathbf{A} \tag{6}$$

where $\mathbf{B} \in \mathbb{R}^{D^l \times D^0}$ and $\mathbf{A} \in \mathbb{R}^{D^0 \times D^l}$ such that $D^0 \ll D^l$; D^0 serves the same role as RoLA rank (Hu et al. 2022). During training, $\mathbf{W}_0$ is frozen and does not receive gradient updates, while $\mathbf{A}$ and $\mathbf{B}$ contain trainable parameters. Note that both $\mathbf{W}_0$ and $\Delta\mathbf{W} = \mathbf{B}\mathbf{A}$ are multiplied with the same input, and their respective output vectors are summed coordinate-wise.

Evaluation of the Encoder Model

This section evaluates TAM on the encoder model.

Datasets

NFL and Politics Two datasets, "NFL" and "Politics", were compiled by sampling the posts from their respective communities on Reddit between September 2018 and February 2019 for the response selection evluation. The dataset is accessible through BigQuery (Henderson et al. 2019). We focused on active speakers and paired their comments with responses, and divided the data into training and test datasets. Those comments and their responses form "a threaded dialogue". The NFL dataset has 230,060 dialogues in the training set (averaging 4.2 utterances and 56.3 words) and 13,765 dialogues in the test set (averaging 4.2 utterances and 57.6 words). The Politics includes 290,020 dialogues in the training set (averaging 4.8 utterances and 81.1 words) and 19,040 dialogues in the test set (averaging 4.9 utterances and 81.5 words). The response selection task aims to identify the last utterance (response) in a dialogue based on the utterance context leading up to that response. Our evaluation focused on the following three object relationships: (1) 'The entire history of a dialogue', which was derived from the word embedding stream for the current dialogue and served as $\mathbf{Q}$; (2) 'the context of the current dialogue', which was created by applying GRU (Cho et al. 2014) to the word embedding streams in both the utterance context and the response. This is represented as either $\mathbf{K}$ or $\mathbf{V}$; (3) 'the topic under discussion in the dialogue', which was obtained from the word embedding stream for the subject title of the dialogue and represented as $\mathbf{S}$. We set the dialogue length to 70 (180) and the title length to 35 (60) for the NFL (Politics) dataset.

TweetQA We also used the TweetQA dataset (Xiong et al. 2019). TweetQA is a large-scale dataset designed for automated question-answering, permitting abstractive responses over social media content. It consists of question-passage-answer triples. The passages are tweets that have been used by journalists in news articles, implying that they contain useful information. It includes 10,692 training triples and 1,979 test triples. The average question length is 6.95, and the average answer length is 2.45. We prepared the following three-object relationships: (1) 'the entire topic of a passage' was derived from the word embedding stream for the passage and served as $\mathbf{Q}$; (2) 'the context of the passage-answer pair' was created by applying GRU to the word embedding streams in both the passage and the answer, and was represented as either $\mathbf{K}$ or $\mathbf{V}$; (3) 'the question assigned for the passage' was obtained from the word embedding stream for the question and was represented as $\mathbf{S}$. We set the passage-answer length to 48 and the question length to 15.

Compared Methods

We integrate *TAM* into the BERT implementation by replacing its attention layer. Here, by directly comparing *TAM* with

Method	*NFL*				*Politics*				*TweetQA*			
Metrics	$R_{10}@1$	$R_{10}@2$	$R_{10}@5$	#params	$R_{10}@1$	$R_{10}@2$	$R_{10}@5$	#params	$R_{10}@1$	$R_{10}@2$	$R_{10}@5$	#params
BERT(CR)	34.08	51.28	81.56	146,167,049	37.62	55.46	83.37	146,167,269	19.98	33.79	63.72	146,167,005
BERT(SCR)	35.76	53.39	82.44	146,167,049	37.09	54.43	82.75	146,167,269	23.39	37.48	69.52	146,167,005
TTrans(CR)	35.19	52.39	81.97	160,170,977	34.24	51.37	80.15	223,873,677	14.18	26.80	58.38	157,474,437
TTrans(SCR)	32.05	49.39	79.32	169,342,001	36.37	53.54	81.73	315,644,781	19.61	36.19	72.01	158,952,501
ViLBERT	35.22	52.90	82.13	375,642,947	46.69	62.49	86.14	375,643,167	20.07	33.79	64.09	375,642,903
ActBERT	43.89	60.12	84.66	641,214,848	48.05	63.66	86.57	428,597,340	29,74	49,63	84,53	641,214,804
TAM	***45.61***	***61.86***	***85.57***	156,228,041	***52.12***	***66.88***	***88.15***	153,066,213	***47.15***	***56.63***	***75.14***	155,047,197

Table 1: Comparison of the methods when learning from scratch.

Method	*NFL*			*Politics*			*TweetQA*		
Metrics	$R_{10}@1$	$R_{10}@2$	$R_{10}@5$	$R_{10}@1$	$R_{10}@2$	$R_{10}@5$	$R_{10}@1$	$R_{10}@2$	$R_{10}@5$
$BERT^P$(CR)-BERT(CR)	65.70	78.50	93.80	70.11	81.63	94.73	60.31	70.72	85.08
$BERT^P$(SCR)	66.80	79.27	94.24	70.95	82.33	95.06	81.03	90.15	97.61
$BERT^P$(SCR)-BERT(SCR)	68.04	80.36	94.78	71.73	83.01	95.19	80.76	90.79	97.33
$BERT^P$(SCR)-ActBERT	67.51	80.11	94.60	71.90	83.32	95.71	80.85	90.24	96.87
$BERT^P$(SCR)-TAM	***68.54***	***80.94***	***94.89***	***72.65***	***83.98***	***95.93***	***81.95***	***90.42***	***97.33***

Table 2: Comparison of a case when a pretrained language model is used for initialization.

the BERT model, we can effectively isolate and analyze the differences in multi-dimensional attention. To ensure a fair evaluation, we utilize two variants of BERT. The first one, denoted as *BERT(CR)*, which inputs 'the entire history of a dialogue' from NFL and Politics, or 'passage-answer pairs' from TweetQA as a query, key, and value. The second one, denoted as *BERT(SCR)*, inputs a concatenation of 'the topic under discussion in the dialogue' and 'the entire history of a dialogue' for NFL and Politics, or a concatenation of the 'question' and 'passage-answer pair' for TweetQA. We also made two Tensorized Transformer variants, *Tensorized Transformer(CR)* and *Tensorized Transformer(SCR)* following a similar configuration to that of BERT. We also compared *TAM* with *ViLBERT* (Lu et al. 2019) and *ActBERT* (Zhu and Yang 2020). *ViLBERT* computes attention from just two sources. Thus, for this comparison, we removed one of the three sources from the dataset when using *ViLBERT*: for Politics or NFL, we excluded 'the context of the current dialogue,' whereas for TweetQA, we excluded 'the context of the passage-answer pair.' *ActBERT* "indirectly" computes relationships between three sources by overlaying 2D source-target attentions. All models used 12 transformer encoder layers, except for *ActBERT* and *TAM* on the Politics dataset. Due to high memory usage, *ActBERT* was limited to 8 layers, and *TAM* was also set to 8 layers for consistency.

Metrics

The evaluation metric was $\text{Recall}_{10}@k$ ($R_{10}@k$) (Qian et al. 2021). Here, given ten responses (or answers), this metric measures if the relevant response (answer) is ranked among the top-k candidates for response selection (question-answering). It uses 9 responses (answers) randomly sampled from the test dataset as negatives. The embedding size of words, D^0, was set to 768 as per (Devlin et al. 2019). The learning rate was set to 1×10^{-5}. We used the AdamW optimizer with beta values of 0.9 and 0.999, respectively, and an epsilon of 1×10^{-8}, following (Loshchilov and Hutter 2019). Sufficient convergence was achieved with 20 (100) epochs for pre-training and 15 (20) epochs for fine-tuning on the NFL and Politics (TweetQA) datasets. We set the dimension size of $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$, and $\mathbf{S}$ to 192 for the NFL dataset, 256 for the Politics dataset, and 160 for the TweetQA dataset. The batch size was 96 for all methods and datasets. *Tensorized Transformer* and *TAM* utilized two core tensors for optimal performance, as relying on a single core tensor often leads to suboptimal results. The experiments were performed using an NVIDIA A100 GPU with 80GB of memory.

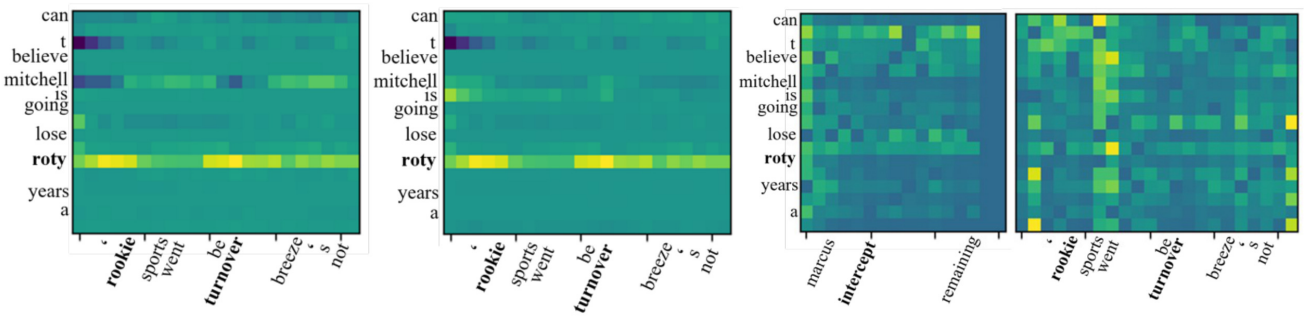
Results when Learning from Scratch

Table 1 presents results in the case of training the model from scratch, without any pre-trained language models. "TTrans" in the table stands for "Tensorized Transformer."

First, *BERT(SCR)* was more accurate than *BERT(CR)* on the NFL and TweetQA datasets, whereas *BERT(CR)* outperformed *BERT(SCR)* on the Politics dataset. These results suggest that semantic information may be effective in improving accuracy, but the current attention model has difficulty in leveraging the potential benefits of different object types. Second, both *Tensorized Transformer(CR)* and *Tensorized Transformer(SCR)* showed only modest improvements or slight decreases compared with BERT across all datasets. This is expected because the main goal of *Tensorized Transformer* is to improve memory efficiency in self-attention for single objects, rather than enhancing accuracy in multi-object scenarios. *ViLBERT* improved accuracy more than *BERT* on the NFL and Politics datasets but was less accurate than *TAM* and *ActBERT* since it fail to effectively use three input sources. *ActBERT* calculates attention for each of the three sources separately, resulting in significantly more parameters compared with *TAM*. *ActBERT* is less accurate than *TAM* since it computes the relationships among the three input sources indirectly. Finally,

Method	*NFL*			*Politics*			*TweetQA*		
Metrics	$R_{10}@1$	$R_{10}@2$	$R_{10}@5$	$R_{10}@1$	$R_{10}@2$	$R_{10}@5$	$R_{10}@1$	$R_{10}@2$	$R_{10}@5$
TAM	45.61	61.86	85.57	52.12	66.88	88.15	47.15	56.63	75.14
w/o Semantic fusing	42.74	59.41	84.21	46.25	61.97	85.65	40.61	54.14	76.61
w/o query aligned	38.01	55.05	81.98	40.30	57.34	83.81	14.73	26.52	61.79

Table 3: Ablation study when learning from scratch.



(a) TAM (intercepts) (b) TAM (summation) (c) BERT

Figure 2: Visualization examples of attention weights for TAM and $BERT$.

TAM achieved high accuracy on all datasets when trained from scratch, with notably better results on the TweetQA dataset compared to other methods. This improvement is due to TAM's effective learning of token co-occurrences in $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{S}$ from the limited data in TweetQA. These results highlight the benefit of using multi-object tensors to represent real-world relationships. The *Tensorized Transformer* has significantly more parameters than *TAM*, particularly on the Politics dataset due to length-dependent $\mathbf{W}^o$ values associated with $K \times V$. Conversely, *TAM* slightly exceeds *BERT* in parameter count due to additional dense network and core tensor weights, while Longformer-based layers (Beltagy, Peters, and Cohan 2020) could further improve adaptability.

Results of TAM combined with Pre-trained Models

We performed experiments with pre-trained models. We took three approaches: (1) we created a hybrid model by integrating the outputs of 12 layers of *BERT^P(SCR)* into the TAM model, denoted as *BERT^P(SCR)-TAM*. (2) we used the output from *BERT(CR)* or *BERT(SCR)* as input for *BERT(CR)* or *BERT(SCR)*, labeled as *BERT^P(CR)-BERT(CR)* or *BERT^P(SCR)-BERT(SCR)*, respectively. (3) we compared our model with a single 12-layer *BERT^P(SCR)*. Here, the term "BERT^P" refers to the pre-trained BERT-base model with 12 layers. In addition, "BERT", "ActBERT", and "TAM" indicate unpretrained models with four layers which achieved superior performance and computational efficiency compared with the 12-layer model. *BERT^P(SCR)-TAM* was pre-trained and fine-tuned for 6 epochs on the NFL and Politics datasets, while other methods required 15 epochs for fine-tuning. This was sufficient for model convergence.

Table 2 presents the results. *BERT^P(SCR)-BERT(SCR)* outperformed *BERT^P(CR)-BERT(CR)* and *BERT^P(SCR)* in terms of accuracy. Additionally, *BERT^P(SCR)-TAM* was more accurate than *BERT^P(SCR)-BERT(SCR)*. These findings suggest that learning multi-object attentions is beneficial when combined with a pre-trained language model. *BERT^P(SCR)-TAM* was more accurate than *BERT^P(SCR)-ActBERT*. This is because ActBERT computes the relationships among the three input sources "indirectly".

Ablation Study

We designed two ablation studies. *w/o Semantic Fusing*, which uses *split&concat* to compute the transformer output directly from the attention weight tensor using q-k-s attention (Ma et al. 2019). *w/o Query Aligned*, which has a tensor decomposition with a core tensor size of $\mathbb{R}^{D\times D\times D}$ instead of $\mathbb{R}^{Q\times D\times D}$, implying a lack of alignment around Q. Table 3 presents the results. *w/o Query Aligned* exhibited lower accuracy than *TAM* since it cannot aggregate information from other objects along the token stream $\mathbf{Q}$ through tensor decomposition. This limitation arises because it cannot explicitly update the query token stream, thus failing to align with the transformer's approach to query updating. In contrast, *TAM* can explicitly update the query token stream while incorporating the latent semantics derived from the correlations between $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{S}$.

Meaningful Results for the Encoder Model

Here, we provide visual examples of the attention weights computed by *TAM* and *BERT(SCR)* for the NFL dataset in Fig. 2. The colors in the visualization represent attention weight levels, with yellow indicating a high level of attention, green representing a moderate level, and blue suggesting low attention. The plots for *TAM* represent utterance contexts on the x-axis and responses on the y-axis, while those for *BERT(SCR)* depict semantics and utterance

	Tea						*SQuAD*					
Metrics	Rouge1	Rouge2	RougeL	BLEU-1	BLEU-2	WordCount	Rouge1	Rouge2	RougeL	BLEU-1	BLEU-2	WordCount
LoRA	13.39	3.38	12.21	12.90	5.30	13.95	75.87	47.60	75.79	65.93	49.40	4.85
TTrans	8.84	1.19	7.71	10.04	3.39	14.17	75.21	46.67	75.16	65.08	48.46	4.77
ViLBERT	13.72	3.48	12.48	13.72	5.61	15.07	75.72	47.54	75.63	65.79	49.37	4.86
ActBERT	13.16	3.35	12.17	12.84	5.23	12.10	75.60	47.17	75.53	65.43	48.90	4.76
TAM	***14.06***	***4.55***	***12.82***	***13.83***	***5.79***	14.22	***76.12***	***48.28***	***76.06***	***66.18***	***49.93***	4.93

Table 4: Comparison of the methods.

contexts on the x-axis. We have selectively labeled certain words on the x- and y-axes for improved readability. In this example, the semantics are "[Highlight] Marcus Peters intercepts Mahomes with 1:13 remaining", the utterance context is "He's a rookie. Sports went up +'s be turnovers.", and the reply is "Can't believe donovan mitchell is going to lose roty 2 years in a row".

Fig. 2-(a) displays the sliced $q \times k$ matrix corresponding to the semantic token "intercept" by *TAM*. Fig. 2-(b) presents the sum of all $q \times k$ matrices for semantic tokens by *TAM*. In (a), *TAM* effectively captures relationships among the word "intercepts" in the semantics, "rookie" and "turnovers" in the utterance context, and "roty (rookie of the year)" in the response. These relationships are crucial, as they pertain to impressive interceptions and turnover plays that are fundamental for a rookie. In (b), *TAM* retains these relationships even after summing the weights across the semantic tokens. In contrast, *BERT(SCR)* fails to identify such relationships among the three different object types, as seen in Fig. 2-(c).

Evaluation of the LLM Adapter

This section evaluates TAM's application to LLM.

Datasets

Tea "Tea" was compiled by sampling the Reddit posts in order to evaluate response generation in the same as way we did for NFL and Politics datasets. It has 44,802 dialogues in the training set (averaging 3.9 utterances and 22.7 words) and 2,533 dialogues in the test set (averaging 3.0 utterances and 20.9 words). The response generation task aims to generate the last utterance in a dialogue based on the utterance context leading up to that response. We set the dialogue length to 190 and the title length to 40. Since Llama2 is a self-attention model, we treated the context-response pair (i.e., dialogue) as the query and memory components, with the title serving as semantics.

SQuAD "SQuAD 1.1" dataset is a collection of question-answer pairs derived from Wikipedia articles. In the dataset, the correct answers of questions can be any sequence of tokens in the given text (called as passage). It contains 87,599 question-answer-passage triples in the training set and 10,570 in the test set. The answer generation task aims to generate the answers based on the questions and their assigned passages leading up to that response. We set the question-answer length to 41 and the passage length to 350. We treat the question-answer pair as the query and memory components, with the passage serving as semantics.

Compared Methods

To ensure fair inputs across methods, we concatenate semantic descriptions before the context-response pair for all models and fed those concatenations into Llama2. The compared methods were as follows: (1) *LoRA* adapter (Hu et al. 2022); (2) *TTrans* adapter, which incorporated Tensorized Transformer (Ma et al. 2019) into the LoRA adapter in spite of TAM; (3) *ViLBERT* adapter, where we implemented a ViLBERT (Lu et al. 2019) within the LoRA adapter; the vision-side embedding stream of ViLBERT served as the semantic input and the language-side as the context-response input; (4) *ActBERT* adapter, where we implemented an ActBERT (Zhu and Yang 2020) within the LoRA adapter; the allocation of two object types as three inputs to the model was designed the same way as TAM. (5) our model, *TAM* adapter.

Metrics

The evaluation metrics were Rouge (Lin 2004) and Bleu (Papineni et al. 2002), following (Touvron et al. 2023; Hu et al. 2022). We implemented the LoRA-TAM adapter on Llama2-13B, adhering to its parameter settings. For the LoRA adapter, we set D^0 to 64, following (Hu et al. 2022), with α set to 16 and a batch size of 16. The dimension size of the TAM model was set to 8 for Tea and 256 for SQuAD, as these values yielded the highest accuracy. The larger dimension size required for SQuAD likely reflects its broader data scope compared to Tea. Sufficient convergence was achieved within one epoch for each dataset. The results are averaged over three random seeds, following (Hu et al. 2022).

Results

Table 4 shows *TAM*'s Rouge score exceeded *LoRA*'s by more than 5% on the Tea dataset, with statistical significance (t-test, $\alpha < 0.05$). This Reddit dataset, characterized by informal user interactions, poses challenges in understanding context. While Rouge scores on the Reddit dataset tended to be low, as seen in previous evaluations (Yang et al. 2021; Liu et al. 2023), *TAM* could observe and predict semantic descriptions and tokens within a dialogue simultaneously using its multi-dimensional attention mechanism. Although the SQuAD dataset has more structured passages and QA relationships than Reddit, *TAM* had higher accuracy in this dataset. *ViLBERT* and *ActBERT*, on the other hand, had lower accuracy than *TAM* because they indirectly measure the relationship between context-response and semantics, while *TAM* directly leverages the simultaneous co-occurrence of context-response and semantics during the attention computation. *TAM* had significantly higher accuracy

compared with *TTrans*. This improvement is due to *TAM*'s capability to aggregate information from various objects around $\mathbf{Q}$ and facilitate transformations between source and target objects.

We also conducted experimental evaluations on the Tea dataset, focusing on informativeness and naturalness, following (Wu et al. 2020). We cross-checked human and LLM judgments as per (Zheng et al. 2023) in 100 random assessments. Our findings indicate that *TAM* produced over 15% more informative outputs than *LoRA* while preserving naturalness (see supplemental material for details).

Conclusion

This paper introduced the Tensorized Attention Model (TAM), which employs Tucker decomposition to compute attention weights across diverse object types. Our evaluation demonstrated that TAM outperforms co-attention models and previous tensorized transformers in accuracy for both encoder and decoder models. In the future, we plan to leverage TAM for pre-training LLMs on larger datasets.

References

Beltagy, I.; Peters, M. E.; and Cohan, A. 2020. Longformer: The Long-Document Transformer. *CoRR*, abs/2004.05150.

Bilgin, O.; Maka, P.; Vergutz, T.; and Mehrkanoon, S. 2022. TENT: Tensorized Encoder Transformer for Temperature Forecasting.

Chefer, H.; Gur, S.; and Wolf, L. 2021. Generic Attention-model Explainability for Interpreting Bi-Modal and Encoder-Decoder Transformers. In *Proc. ICCV'21*, 387–396.

Cho, K.; van Merriënboer, B.; Bahdanau, D.; and Bengio, Y. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proc. SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 103–111.

De Lathauwer, L. 2008. Decompositions of a Higher-Order Tensor in Block Terms—Part II: Definitions and Uniqueness. *SIAM Journal on Matrix Analysis and Applications*, 30(3): 1033–1066.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. NAACL-HLT'19*, 4171–4186.

Gemini-team; Anil, R.; Borgeaud, S.; Wu, Y.; and et al. 2023. Gemini: A Family of Highly Capable Multimodal Models. volume cs.CL, 2312.11805.

Hawkins, C.; Liu, X.; and Zhang, Z. 2022. Towards Compact Neural Networks via End-to-End Training: A Bayesian Tensor Approach with Automatic Rank Determination. *SIAM Journal on Mathematics of Data Science*, 4(1): 46–71.

Henderson, M.; Budzianowski, P.; Casanueva, I.; Coope, S.; Gerz, D.; Kumar, G.; Mrksic, N.; Spithourakis, G.; Su, P.; Vulic, I.; and Wen, T. 2019. A Repository of Conversational Datasets. *CoRR*, abs/1904.06472.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *Proc. ICLR'22*.

Lin, C.-Y. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, 74–81.

Liu, S.; Cho, H.; Freedman, M.; Ma, X.; and May, J. 2023. RECAP: Retrieval-Enhanced Context-Aware Prefix Encoder for Personalized Dialogue Response Generation. In *Proc. ACL'23*, 8404–8419.

Loshchilov, I.; and Hutter, F. 2019. Decoupled Weight Decay Regularization. *arXiv*, cs.LG, 1711.05101.

Lu, J.; Batra, D.; Parikh, D.; and Lee, S. 2019. ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks. In *NeurIPS*, 13–23.

Ma, X.; Zhang, P.; Zhang, S.; Duan, N.; Hou, Y.; Zhou, M.; and Song, D. 2019. A Tensorized Transformer for Language Modeling. In *Proc. NeurIPS'19*, 2229–2239.

Meta. 2024. Meta Llama 3, https://llama.meta.com/llama3/. Accessed: June 1, 2024.

Nakatsuji, M.; Toda, H.; Sawada, H.; Zheng, J.; and Hendler, J. A. 2016. Semantic sensitive tensor factorization. *Artif. Intell.*, 230: 224–245.

OpenAI. 2023. GPT-4 Technical Report. *ArXiv*, abs/2303.08774.

Panahi, A.; Saeedi, S.; and Arodz, T. 2021. Shapeshifter: a Parameter-efficient Transformer using Factorized Reshaped Matrices. In *Proc. NeurIPS'21*, volume 34, 1337–1350.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proc. ACL'02*, 311–318.

Qian, H.; Dou, Z.; Zhu, Y.; Ma, Y.; and Wen, J. 2021. Learning Implicit User Profile for Personalized Retrieval-Based Chatbot. In *Proc. CIKM'21*, 1467–1477.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proc. EMNLP'16*, 2383–2392.

Shen, T.; Zhou, T.; Long, G.; Jiang, J.; and Zhang, C. 2019. Tensorized Self-Attention: Efficiently Modeling Pairwise and Global Dependencies Together. In *Proc. NAACL'19*, 1256–1266.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; and et.al. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv*, cs.CL, 2307.09288.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is All you Need. In *Proc. NIPS'17*, volume 30.

Wu, S.; Li, Y.; Zhang, D.; Zhou, Y.; and Wu, Z. 2020. Diverse and Informative Dialogue Generation with Context-Specific Commonsense Knowledge Awareness. In *Proc. ACL'20*, 5811–5820.

Xiong, W.; Wu, J.; Wang, H.; Kulkarni, V.; Yu, M.; Guo, X.; Chang, S.; and Wang, W. Y. 2019. TweetQA: A Social Media Focused Question Answering Dataset. In *Proc. ACL'19*.

Yang, Z.; Wu, W.; Hu, H.; Xu, C.; Wang, W.; and Li, Z. 2021. Open Domain Dialogue Generation with Latent Images. In *Proc. AAAI'21*, 14239–14247.

Ye, J.; Li, G.; Chen, D.; Yang, H.; Zhe, S.; and Xu, Z. 2020. Block-term Tensor Neural Networks. *CoRR*, abs/2010.04963.

Zheng, L.; Chiang, W.; Sheng, Y.; Zhuang, S.; Wu, Z.; Zhuang, Y.; Lin, Z.; Li, Z.; Li, D.; Xing, E. P.; Zhang, H.; Gonzalez, J. E.; and Stoica, I. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Proc. NeurIPS'23*.

Zhu, L.; and Yang, Y. 2020. ActBERT: Learning Global-Local Video-Text Representations. In *Proc. CVPR'20*, 8743–8752.