SYNTHESIZER: Rethinking Self-Attention in Transformer Models

Yi Tay
Google Research
Mountain View
yitay@google.com

Dara Bahri
Google Research
Mountain View
dbahri@google.com

Donald Metzler
Google Research
Mountain View
metzler@google.com

Da-Cheng Juan
Google Research
Mountain View
dacheng@google.com

Zhe Zhao
Google Research
Mountain View
zhezhao@google.com

Che Zheng
Google Research
Mountain View
chezheng@google.com

Abstract

The dot product self-attention is known to be central and indispensable to state-of-the-art Transformer models. But is it really required? This paper investigates the true importance and contribution of the dot product-based self-attention mechanism on the performance of Transformer models. Via extensive experiments, we find that (1) random alignment matrices surprisingly perform quite competitively and (2) learning attention weights from token-token (query-key) interactions is not that important after all. To this end, we propose SYNTHESIZER, a model that learns synthetic attention weights without token-token interactions. Our experimental results show that SYNTHESIZER is competitive against vanilla Transformer models across a range of tasks, including MT (EnDe, EnFr), language modeling (LM1B), abstractive summarization (CNN/Dailymail), dialogue generation (PersonaChat) and Multi-task language understanding (GLUE, SuperGLUE).

1 Introduction

Transformer models [Vaswani et al., 2017] have demonstrated success across a wide range of tasks. This has resulted in Transformers largely displacing once popular auto-regressive and recurrent models in recent years. At the heart of Transformer models lies the query-key-value dot product attention. The success of Transformer models is widely attributed to this self-attention mechanism since fully connected token graphs, which are able to model long-range dependencies, provide a robust inductive bias.

But is the dot product self-attention really so important? Do we need it? Is it necessary to learn attention weights via expensive pairwise dot products? This paper seeks to develop a deeper understanding of the role that the dot product self-attention mechanism plays in Transformer models.

The fundamental role of dot product self-attention is to learn self-alignment, i.e., to determine the relative importance of a single token with respect to all other tokens in the sequence. To this end, there have been memory metaphors and analogies constructed to support this claim. Indeed, the terms *query*, *keys*, and *values* imply that self-attention emulates a content-based retrieval process which leverages pairwise interactions at its very core. This paper rethinks this entire process.

Moving against convention, this paper postulates that we can not only do without dot product self-attention but also content-based *memory-like* self-attention altogether. Traditionally, attention weights are learned at the instance or sample level, where weights are produced by instance-level by pairwise

Preprint. Under review.

interactions. As a result, these instance-specific interactions often fluctuate freely across different instances as they lack a consistent global context.

This paper proposes SYNTHESIZER, a new model that learns to synthesize the self-alignment matrix instead of manually computing pairwise dot products. We propose a diverse suite of synthesizing functions and extensively evaluate them. We characterize the source information that these synthesizing functions receive, i.e., whether they receive information from individual tokens, token-token interactions, and/or global task information. Intuitively, different source inputs to the synthesizing functions should capture diverse views, which may be useful when employed in conjunction.

Aside from generalizing the standard Transformer model, we show that it is possible to achieve competitive results with fully global attention weights that do not consider token-token interactions or any instance-level (local) information at all. More specifically, a *random* matrix SYNTHESIZER model achieves a 27.27 BLEU score on WMT 2014 English-German[1]. We observe that the popular and well-established dot-product content-based attention can be replaced with simpler variants without sacrificing much performance in some cases. In general, we believe our findings will spur further investigation and discussion about the true role and utility of the self-attention mechanism in Transformer models.

SYNTHESIZER is completely transformation-based, only relies on simple feed-forward layers, and completely dispenses with dot products and explicit token-token interactions. To reiterate, this work moves away from the implied notion of a query-key-value memory store and shows that randomized alignment matrices are sufficient for many tasks in practice.

Our Contributions Our key contributions are described as follows:

- We propose Synthetic Attention, a new way of learning to attend without explicitly attending (i.e., without dot product attention or content-based attention). Instead, we generate the alignment matrix independent of token-token dependencies and explore a potpourri of parameterized functions for synthesizing attention matrices.
- We propose SYNTHESIZER, a new model that leverages Synthetic Attention. The model performs competitive to state-of-the-art Transformer models on a wide range of language tasks, including machine translation and language modeling.
- Moreover, We show that (1) random learnable alignment matrices perform competitively and (2) token-token dependencies are not necessary to achieve good performance with Transformer models on certain tasks.

2 Related Work

Attention-based models are used across a wide spectrum of problem domains. Such models are especially popular, due to their effectiveness, in the language and vision domains. Attention models can be traced back to the machine translation models of [Bahdanau et al., 2014] and [Luong et al., 2015], where attention is employed to learn soft word alignments between language pairs. The intuition behind the attention mechanism is deeply-rooted in the notion of memory-based retrieval [Graves et al., 2014, Weston et al., 2014], in which soft differentiable addressing of memory was initially proposed.

The paradigm of learning self-alignments, also known as self-attention, has been largely popularized by Transformer models [Vaswani et al., 2017]. This technical narrative has also been explored by a number of other recent studies, including those on intra-attention [Parikh et al., 2016], self-matching networks [Wang et al., 2017], and LSTMN [Cheng et al., 2016]. To this end, Transformer models, which function primarily based on self-attention and feed-forward layers, generally serve as a reliable replacement for autoregressive recurrent models.

The self-attention layer itself has been the subject of many recent technical innovations. For example, recent studies have investigated improving the layer's overall efficiency via sparsification and reducing the complexity of computing the alignment matrix [Child et al., 2019, Kitaev et al., 2020, Huang et al., 2018, Tay et al., 2020, Beltagy et al., 2020]. These methods are tightly coupled with the query-key-value paradigm, employing a form of memory-based content retrieval as an attention

[1] The originally reported result is 27.30.

mechanism. On the other end of the spectrum, there have been studies that advocate for replacing self-attention with convolution [Wu et al., 2019]. The recent surge in interest in simplifying the attention mechanism raises important questions about the role and utility of the pairwise dot products, which are one the defining characteristics of self-attention models.

Our work is a novel take on the self-attention mechanism in Transformer models. We delve deeper, starting with replacing the pairwise dot products with what we call synthesizing functions that learn attention matrices that may or may not depend on the input tokens. The most closely related work is [Raganato et al., 2020], in which the authors propose using fixed (i.e., not learned) attention patterns in Transformer encoders. However, the scope of their work is limited to encoders and relies on manually defined patterns that seem to work well. Our work takes this intuition further and expands on this narrative.

3 Our Proposed Method

This section introduces our proposed SYNTHESIZER model. At its core, our model is essentially a Transformer model with self-attention modules replaced with our Synthetic Attention modules. Figure 3.1 illustrates the key ideas behind (a) Transformer (b) Dense Synthesizers and (c) Random Synthesizers.

3.1 Synthesizer Model

This section introduces Synthetic Attention, our proposed self-attention module. Our model removes the notion of query-key-values in the self-attention module and directly synthesizes the alignment matrix instead.

Dense Synthesizer Let us consider the simplest variation of the SYNTHESIZER model which is conditioned on each input token. Overall, our method accepts an input $X \in \mathbb{R}^{\ell \times d}$ and produces an output of $Y \in \mathbb{R}^{\ell \times d}$. Here, ℓ refers to the sequence length and d refers to the dimensionality of the model. We first adopt $F(.)$, a parameterized function, for projecting input X_i from d dimensions to ℓ dimensions.

$$B_i = F(X_i) \tag{1}$$

where $F(.)$ is a parameterized function that maps $\mathbb{R}^d$ to $\mathbb{R}^\ell$ and i is the i-th token of X. Intuitively, this can be interpreted as learning a token-wise projection to the sequence length ℓ. Essentially, with this model, each token predicts weights for each token in the input sequence. In practice, we adopt a simple two layered feed-forward layer with ReLU activations for $F(.)$:

$$F(X) = W(\sigma_R(W(X) + b)) + b \tag{2}$$

where σ_R is the ReLU activation function. Hence, B is now of $\mathbb{R}^{\ell \times \ell}$. Given B, we now compute:

$$Y = \text{Softmax}(B)G(X). \tag{3}$$

where $G(.)$ is another parameterized function of X that is analogous to V (value) in the standard Transformer model.

This approach eliminates the dot product altogether by replacing $QK^\top$ in standard Transformers with the synthesizing function $F(.)$.

Random Synthesizer The previous variant learns synthetic attention by conditioning on each input of X and projecting to ℓ dimensions. Hence, the Dense Synthesizer conditions on each token independently, as opposed to pairwise token interactions in the vanilla Transformer model. We consider another variation of SYNTHESIZER where the attention weights are not conditioned on any input tokens. Instead, the attention weights are initialized to random values. These values can then either be trainable or kept fixed (denoted as *Fixed*).

Let R be a randomly initialized matrix. The Random Synthesizer is defined as:

$$Y = \text{Softmax}(R)G(X). \tag{4}$$

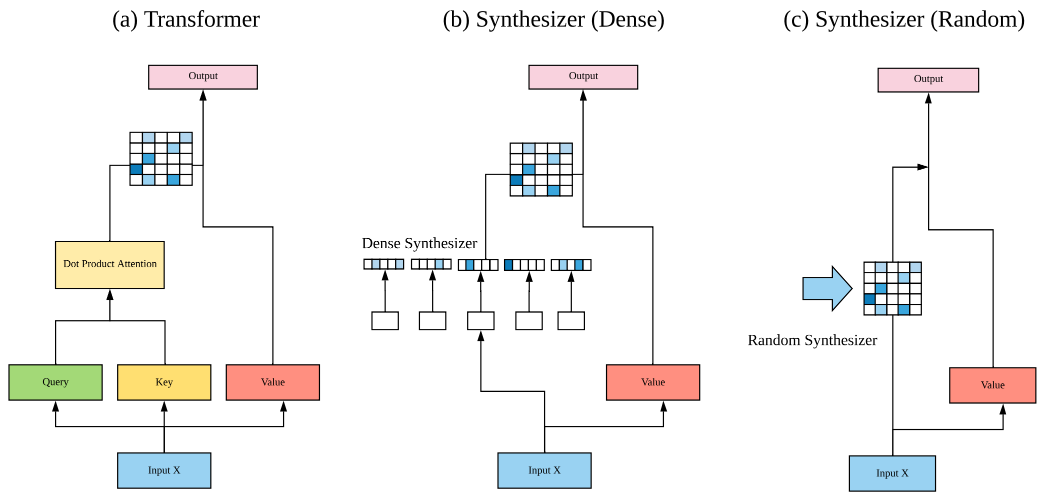


Figure 1: Our proposed SYNTHESIZER model architecture.

where $R \in \mathbb{R}^{\ell \times \ell}$. Notably, each head adds ℓ^2 parameters to the network. The basic idea[2] of the Random Synthesizer is to not rely on pairwise token interactions or any information from individual token but rather to learn a task-specific alignment that works well globally across many samples. This is a direct generalization of the recently proposed fixed self-attention patterns Raganato et al. [2020].

Factorized Models The Dense Synthesizer adds $d \times \ell$ parameters to the network. On the other hand, the Random Synthesizer adds $\ell \times \ell$ parameters. Here, note that we omit the Q, K projections in the standard Transformer which results in further parameter savings. Despite these savings, synthesized models can be cumbersome to learn when ℓ is large. Hence, we propose factorized variations of the SYNTHESIZER models and show that these variants perform comparably in practice.

Factorized Dense Synthesizer Factorized outputs not only slightly reduce the parameter cost of the SYNTHESIZER but also aid in preventing overfitting. The factorized variant of the dense synthesizer can be expressed as follows:

$$A, B = F_A(X_i), F_B(X_i) \tag{5}$$

where $F_A(.)$ projects input X_i into a dimensions, $F_B(.)$ projects X_i to b dimensions, and $a \times b = \ell$. The output of the factorized module is now written as:

$$Y = \text{Softmax}(C)G(X). \tag{6}$$

where $C = H_A(A) * H_B(B)$ where H_A, H_B are tiling functions and $C \in \mathbb{R}^{\ell \times \ell}$. The tiling function simply duplicates the vector k times, i.e., $\mathbb{R}^{\ell} \rightarrow \mathbb{R}^{\ell k}$. In this case, $H_A()$ is a projection of $\mathbb{R}^a \rightarrow \mathbb{R}^{ab}$ and $H_B()$ is a projection of $\mathbb{R}^b \rightarrow \mathbb{R}^{ba}$. To avoid having similar values within the same block, we compose the outputs of H_A and H_B.

Factorized Random Synthesizer Similar to Factorized Synthesizers, we are also able to factorize R into low rank matrices $R_1, R_2 \in \mathbb{R}^{\ell \times k}$.

$$Y = \text{Softmax}(R_1 R_2^{\top})G(X). \tag{7}$$

Therefore, it is easy to see that, for each head, this reduces the parameter costs from ℓ^2 to $2(\ell k)$ where $k << \ell$ and hence helps prevent overfitting. In practice, we use a small value of $k = 8$.

[2]We were not expecting this variation to work at all, but it turns out to be a strong baseline.

Mixture of Synthesizers Finally, we note that all of the proposed synthetic attention variants can be mixed in an additive fashion. This can be expressed as:

$$Y = \text{Softmax}(\alpha_1 S_1(X) + \cdots \alpha_N S_N(X))G(X). \tag{8}$$

where $S(.)$ is a parameterized synthesizing function and the α (where $\sum \alpha = 1$) are learnable weights. In the case of mixing Random Factorized with standard Dense Synthesizers, this is expressed as:

$$Y = \text{Softmax}(R_1 R_2^\top + F(X))G(X). \tag{9}$$

We investigate several Mixture of Synthesizers variants in our experiments.

3.2 Discussion

This paper asks fundamental questions about the attention matrix A and whether it is possible to synthesize A by alternate means other than pairwise attention. It is worth noting that the regular dot product attention can also be subsumed by our SYNTHESIZER framework, i.e., SYNTHESIZER generalizes the Transformer model. In the case of the Transformer, the synthesizing function in question is $S(X) = F_Q(X)F_K(X)^\top$.

Model	$S(X)$	Condition On	Sample	Interact	$\lvert\theta\rvert$
Dot Product Attention	$F_Q(X)F_K(X_i)^\top$	$X_j \ \forall j$	Local	Yes	$2d^2$
Random	R	N/A	Global	No	$.\ \ell^2$
Factorized Random	$R_1 R_2^\top$	N/A	Global	No	$2\ell k$
Dense	$F_1 \sigma(F_2(X_i))$	X_i	Local	No	$d^2 + d\ell$
Factorized Dense	$H_A(F_A(X_i))) * H_B(F_B(X_i)))$	X_i	Local	No	$d^2 + d(k_1 + k_2)$

Table 1: Overview of all Synthesizing Functions.

Table 1 lists the different model variants explored within our SYNTHESIZER framework. The 'condition on' column refers to whether the synthesized output is produced as a function of X_i or every X_i, X_j pair. The 'sample' column indicates whether a given variant leverages local or global context. Random Synthesizers are global because they share the same global alignment patterns across all samples. Dense Synthesizers are considered to be local as they are conditioned on X_i, which makes the alignment pattern dependent on each individual sample. To this end, it is imperative for synthesized models to have multiple heads to be effective. Finally, we note that Random Synthesizers are related to Relative Positional Representations [Shaw et al., 2018], which typically augmented standard self-attention mechanisms. The key difference is that Random Synthesizers capture positional (relative) information without relying on token-token semantics.

4 Experiments

This section outlines our experimental setup and results.

4.1 Machine Translation

We conduct experiments on WMT'14 English-German (EnDe) and WMT'14 English-French (EnFr), which are well-established machine translation benchmarks. The WMT EnDe dataset is comprised of 4.5 million sentence pairs, while the EnFr dataset consists of 36 million sentence pairs. We implement our models in Tensor2Tensor using the standard **base** hyperparameter settings. Further details can be found in the appendix.

Experimental Results on Machine Translation Table 2 reports results on machine translation. First, we observe that our Random Synthesizer baseline achieves 27.27 on EnDe and 41.12 on EnFr. The non-trainable (i.e., fixed) variant performs substantially worse, but still yields surprisingly strong ≈ 24 BLEU with fixed random attention weights. Most other SYNTHESIZER variants achieve competitive performance, although with slight performance degradation compared to Transformers. An interesting finding is that the Mixture model of Random and Dense synthesizer outperforms vanilla Transformers on EnDe. When mixing the standard dot product attention, performance further increases by $+0.8$ BLEU points on EnDe.

In general, the performance of SYNTHESIZER variants are competitive with standard Transformers for this task. Furthermore, SYNTHESIZER variants have reduced computational complexity and parameter costs that are about 10% lower than Transformers. When taken together, synthetic attention is an appealing alternative to traditional dot product self-attention.

	NMT (BLEU)			LM (PPL)	
Model	# Params	EnDe	EnFr	# Params	LM1B
Transformer [Vaswani et al., 2017]	68M	27.30	38.10	-	-
Transformer (Our run)	68M	27.67	41.57	70M	38.21
Transformer (Control)	73M	27.97	41.83	-	-
Synthesizer (Fixed Random)	61M	23.89	38.31	53M	50.52
Synthesizer (Random)	67M	27.27	41.12	58M	40.60
Synthesizer (Factorized Random)	61M	27.30	41.12	53M	42.40
Synthesizer (Dense)	62M	27.43	41.39	53M	40.88
Synthesizer (Factorized Dense)	61M	27.32	41.57	53M	41.20
Synthesizer (Random + Dense)	67M	27.68	41.21	58M	42.35
Synthesizer (Dense + Vanilla)	74M	27.57	41.38	70M	**37.27**
Synthesizer (Random + Vanilla)	73M	**28.47**	**41.85**	70M	40.05

Table 2: Experimental Results on WMT'14 English-German, WMT'14 English-French Machine Translation tasks and Language Modeling One Billion (LM1B).

	Summarization			Dialogue				
Model	Rouge-1	Rouge-2	Rouge-L	Bleu-1/4	Rouge-L	Meteor	CIDr	Emb
Transformer	38.24	**17.10**	35.77	12.03/3.20	13.38	5.89	18.94	83.43
Synthesizer (R)	35.47	14.92	33.10	14.64/2.25	15.00	6.42	19.57	84.50
Synthesizer (D)	36.05	15.26	33.70	**15.58/4.02**	**15.22**	**6.61**	**20.54**	**84.95**
Synthesizer (D+V)	38.57	16.64	**36.02**	14.24/3.57	14.22	6.32	18.87	84.21
Synthesizer (R+V)	**38.57**	16.24	35.95	14.70/2.28	14.79	6.39	19.09	84.54

Table 3: Experimental results on Abstractive Summarization (CNN/Dailymail) and Dialogue Generation (PersonaChat).

4.2 Language Modeling

We experiment on the well-established task of subword level language modeling. We use the Language Modeling One Billion (LM1B) dataset. Our baselines are similar to the ones used for machine translation except they only involve the decoder in the context of the LM task. We implement our models in Tensor2Tensor. We train our models on $300K$ steps on 16 TPU V2 chips. Further details can be found in the appendix.

Experimental Results on LM1B Table 2 reports our results on LM1B (perplexity). We find that the Random Synthesizers perform within $1-2$ perplexity points away from the vanilla Transformer model. The best performing model is the Synthesizer (Dense + Vanilla), which achieves the best performance on this setting.

4.3 Text Generation

Next, we evaluate SYNTHESIZER on two text generation tasks – abstractive summarization using the CNN/Dailymail dataset and dialogue generation using the PersonaChat dataset [Zhang et al., 2018]. The model used is a simple Seq2Seq Transformer model. We leverage our SYNTHESIZER in both the encoder and decoder. All models use the *base* size setting. For the dialogue generation task, due to the smaller dataset size, we train a *small* model for $20K$ steps. For the summarization task, we use the well-established metrics, i.e., Rouge-1, Rouge-2 and Rouge-L. For the dialogue generation task, we use NLG-Eval[3] [Sharma et al., 2017] and report BLEU-1, BLEU-4, Rouge-L, Meteor, CIDr and Embedding based similarity scores (Emb).

[3] `https://github.com/Maluuba/nlg-eval`.

Results on Summarization Table 3 reports results for the summarization and dialogue generation tasks. For summarization, we find that the (R) and (D) variants do not outperform Transformers. The performance of the (D) model is ≈ 2 Rouge-L points below Transformers. Hence, we postulate that the local sample-wise pairwise interactions are important for the summarization task. On the other hand, the utility of synthesized attention can also be observed, i.e., the (R+V) and (R+D) models both outperform Transformers.

Results on Dialogue Generation On this task, Synthesizers (R) and (D) both outperform vanilla Transformers by a reasonable margin ($\approx$ 1-3) points across most/all metrics. The best performing model here is the (D) variant. Surprisingly, unlike most other tasks, the (+V) variants do not perform well, signifying that dot product self-attention may actually be harmful for this task.

4.4 Multi-Task Natural Language Processing

Finally, we evaluate our SYNTHESIZER model on multi-task language understanding (GLUE [Wang et al., 2018] and SuperGLUE [Wang et al., 2019]) following the T5 (text-to-text Transformer) [Raffel et al., 2019] methodology. Our experiments are based on the T5 repository[4] and are implemented in Mesh Tensorflow [Shazeer et al., 2018]. We pre-train the vanilla T5 models and our models for 524288 steps using the span denoising objective. We then co-train the model on multiple tasks. We co-train on the `en_mix` mixture (SuperGLUE and GLUE) for $100k$ steps with a constant learning rate of 10^{-3}.

Model	Glue	CoLA	SST	MRPC	STSB	QQP	MNLI	QNLI	RTE
T5 (Base)	83.5	53.1	**92.2**	**92.0/88.7**	89.1/88.9	88.2/91.2	84.7/**85.0**	91.7	76.9
Syn (R)	75.1	41.2	91.2	85.9/79.4	74.0/74.3	85.5/89.0	77.6/78.1	87.6	59.2
Syn (D)	72.0	18.9	89.9	86.4/79.4	75.3/75.5	85.2/88.3	77.4/78.1	86.9	57.4
Syn (D+V)	82.6	48.6	92.4	91.2/87.7	88.9/89.0	88.6/91.5	84.3/84.8	91.7	75.1
Syn (R+V)	**84.1**	**53.3**	**92.2**	91.2/87.7	**89.3/88.9**	**88.6/91.4**	**85.0**/84.6	**92.3**	**81.2**

Table 4: Experimental results (dev scores) on multi-task language understanding (GLUE benchmark) for *small* model and `en-mix` mixture. Note: This task has been co-trained with SuperGLUE.

Model	SGlue	BoolQ	CB	CoPA	MultiRC	ReCoRD	RTE	WiC	WSC
T5 (Base)	70.3	78.2	72.1/83.9	59.0	73.1/32.1	**71.1/70.3**	77.3	**65.8**	**80.8**
Syn (R)	61.1	69.5	54.6/73.2	60.0	63.0/15.7	58.4/57.4	67.5	64.4	66.3
Syn (D)	58.5	69.5	51.7/71.4	51.0	66.0/15.8	54.1/53.0	67.5	65.2	58.7
Syn (D+V)	69.7	79.3	74.3/85.7	64.0	73.8/33.7	69.9/69.2	78.7	64.3	68.3
Syn (R+V)	**72.2**	**79.3**	**82.7/91.1**	**64.0**	**74.3/34.9**	70.8/69.9	**82.7**	64.6	75.0

Table 5: Experimental results (dev scores) on multi-task language understanding (SuperGLUE benchmark) for *small* model and `en-mix` mixture. Note: This task has been co-trained with GLUE.

Results on GLUE and SuperGLUE Tables 4 and 5 report results on the GLUE and SuperGLUE benchmarks. We note that the (R) and (D) variants of SYNTHESIZER do not achieve reasonable performance. This can be largely attributed to the fact that the encoder self-attention in the T5 setting also functions as a cross-sentence attention. For example, in the entailment or reading comprehension tasks, the premise and hypothesis are concatenated together and self-attention effectively acts as cross-sentence attention. Optimistically, we observe that Syn (R+V) outperforms the T5 model by a substantial margin (+1.9 points on SuperGLUE and +0.6 points on GLUE).

4.5 Overall Summary of Quantitative Results

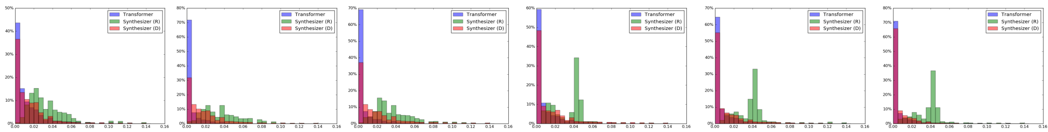
On all evaluated tasks, we showed that synthesized attention functions competitively, i.e., it achieves performance reasonably close to the dot product self-attention. On one task (dialogue generation), the dot product self-attention is found to actually degrade performance. Amongst the other tasks, machine translation is the least affected by the removal of the vanilla dot product. These findings allow us to introspect about whether pairwise comparisons for self-attention are even necessary. We would like to emphasize that this solely refers to self-attention and not cross-attention. On the multi-task language understanding benchmark, the self-attention functions as a form of cross-attention

[4] `https://github.com/google-research/text-to-text-transfer-transformer`

by concatenating sentence pairs. Hence, synthesize attention performance is considerably worse than vanilla Transformers. However, complementing the base T5 model with synthetic attention boosts performs, showing that synthesized attention provides additional value to current state-of-the-art models.

4.6 Analysis

In this section, we perform a deeper analysis of the SYNTHESIZER model.



Enc L1 Enc L3 Enc L5 Dec L1 Dec L3 Dec L5

Figure 2: Histogram of Encoder and Decoder Attention Weights on MT (WMT EnDe). L denotes the layer number and Enc/Dec denotes encoder or decoder.

Distribution of Weights We are interested in investigating how the synthetically generated attention weights differ from the dot product attention weights. Figure 2 shows the attention histograms on trained Transformer and SYNTHESIZER models. We report histograms at layers 1, 3, and 5 of a 6 layered (Transformer or SYNTHESIZER) model at $50K$ steps. We found that the weight distributions remain relatively identical thereafter. Figure 3 shows the initialization state. We observe that there are distinct differences in the weight distribution of SYNTHESIZER and Transformer models. The variance of the SYNTHESIZER weights tends to be higher. On the other hand, the weights on the Transformer model tends to gravitate near 0 and have smaller variance. There are also notable differences across the (R) and (D) SYNTHESIZER variants. Specifically, the (D) model in general has greater max values with more values in the 0.1-0.2 range while the values of the R model tends to stay closer to 0.

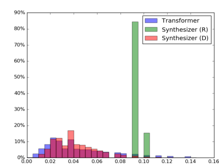


Figure 3: Init Decoder weights (Reference)

Effect of Number of Heads We also investigate the impact of the number of heads on performance. We trained three Random Synthesizer models for the `small` version of the machine translation tasks using the T5 framework without pretraining. For simplicity, evaluation is done via greedy decoding. We report scores on the development set. Table 6 reports the results on varying the number of heads on performance.

Heads	EnDe	EnFr	EnRo
Syn h=2	19.43	34.12	18.67
Syn h=4	20.42	35.26	19.78
Syn h=8	20.88	34.92	20.28
Syn h=16	21.71	35.26	20.43
Syn h=32	**21.72**	**36.01**	**20.52**

Table 6: Effect of number of heads on multi-task MT. Increasing the number of heads improves performance.

5 Conclusion

This paper proposed SYNTHESIZER, a new Transformer model that employs Synthetic Attention. We conducted a principled study to better understand and evaluate the utility of global alignment and local, instance-wise alignment (e.g., independent token and token-token based) in self-attention. We show that, on multiple tasks such as machine translation, language modeling and dialogue generation, synthetic attention demonstrates competitive performance compared to vanilla self-attention. Moreover, for the dialogue generation task, pairwise interactions actually hurt performance. Notably, we reemphasize that this study refers to self-attention. We found that we are not able to replace cross-attention with simpler variants in most cases. Overall, we hope our study will encourage further investigations into the component-wise effectiveness of well-established Transformer models.

6 Supplementary Material

6.1 Detailed Setup for Experiments

Machine Translation We implement our models in Tensor2Tensor, using the standard **base** hyper-parameter settings. Specifically, we use byte-pair encoding (BPE), 6-layered Transformer networks with hidden size 512, filter size of 2048 and 8 heads. We use label smoothing of 0.1. The maximum sequence length is set to 256. Training is performed using 8 x V100 GPUs. We train all models for $250K$ steps and report results at the last checkpoint. We use a length penalty of 0.6 and beam size of 4 following the default settings. We also compare with standard Transformer models. In the interest of keeping a consistent, fair evaluation across all model settings, we do not use checkpoint averaging or tune the decoding hyperparameters although this generally leads to better performance. We evaluate BLEU scores using `sacrebleu`.

Language Modeling We implement our models in Tensor2Tensor using the packed TPU setup of sequence length 256. We train our models on $300K$ steps on 16 TPU V2 chips. We use the `lmx_base` model setting for fair comparison across all model variations. The model has 6 layers and 8 heads, along with a filter width of 2048 and hidden size of 512. We used `conv_relu` for the positional feed-forward layers across all baselines since we find them to perform slightly better. We report results (subword level perplexity scores) on the test set at the final checkpoint.

Summarization For the summarization task, we train all models for $300K$ steps and a batch size of 128. All models use the *base* size setting. For the dialogue generation task, due to the smaller dataset size, we train a *small* model for $20K$ steps. All results are reported on the test set. For the summarization task, we use the well-established metrics, i.e., Rouge-1, Rouge-2 and Rouge-L. Experiments are conducted using Mesh Tensorflow.

Dialogue Generation For the dialogue generation task, we train our models on the small size for $20K$ steps. Experiments are conducted in Tensor2Tensor. We use NLG-Eval[5] [Sharma et al., 2017] and report BLEU-1, BLEU-4, Rouge-L, Meteor, CIDr and Embedding based similarity scores (Emb).

Multi-Task Language Understanding Our experiments are based on the T5 repository[6] implemented in Mesh Tensorflow [Shazeer et al., 2018]. We pre-train the vanilla T5 models and our models for 524288 steps using the span denoising objective. We then co-train the model on multiple tasks. We co-train on the `en_mix` mixture (SuperGLUE and GLUE) for $100k$ steps with a constant learning rate of 10^{-3}. Embedding and Softmax output layer parameters are kept fixed. The maximum sequence length is set to 512. We evaluate on the `en_mix` mixture as defined in the original codebase which is comprised of training GLUE, SuperGLUE and SQuAD in a single model.

6.2 Additional Variants of Synthesizer

We report results of several additional variants of SYNTHESIZER, most of which we found to have marginal or no improvement over the simple dense/random variations.

- Convolution - Applying a 1D convolution instead of a 2 layer nonlinear network. We vary the filter width in our experiments.
- Bottleneck - Converting the 2 layered feed forward network to a bottleneck layer, e.g., $512 \rightarrow 16 \rightarrow 512$. We also experiment with a convolutional variant of bottleneck, i.e., projecting to low dimension space and then projecting back to high dimensions.
- Gated Linear Units (GLU), applying the GLU units of [Dauphin et al., 2017] as the Synthesizing function.

[5] `https://github.com/Maluuba/nlg-eval`.
[6] `https://github.com/google-research/text-to-text-transfer-transformer`

Variant	BLEU
Transformer	27.67
Random	27.27
Dense	27.43
Conv ($f = 3$) Linear	27.43
ConvReluConv ($f = 3$)	27.51
ConvReluConv ($f = 5$)	27.56
ConvReluConv ($f = 3, 5$)	27.49
Bottleneck + Dense	27.43
Bottleneck + ConvReluConv	27.72
GLU	27.43

Table 7: Results for additional SYNTHESIZER variants on WMT EnDe (BLEU scores)

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv:2004.05150*, 2020.

Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 933–941. JMLR. org, 2017.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer. *arXiv preprint arXiv:1809.04281*, 2018.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*, 2016.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

Alessandro Raganato, Yves Scherrer, and Jörg Tiedemann. Fixed encoder self-attention patterns in transformer-based machine translation. *arXiv preprint arXiv:2002.10260*, 2020.

Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *CoRR*, abs/1706.09799, 2017. URL `http://arxiv.org/abs/1706.09799`.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

Noam Shazeer, Youlong Cheng, Niki Parmar, Dustin Tran, Ashish Vaswani, Penporn Koanantakool, Peter Hawkins, HyoukJoong Lee, Mingsheng Hong, Cliff Young, et al. Mesh-tensorflow: Deep learning for supercomputers. In *Advances in Neural Information Processing Systems*, pages 10414–10423, 2018.

Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. Sparse sinkhorn attention. *arXiv preprint arXiv:2002.11296*, 2020.

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, pages 3261–3275, 2019.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198, 2017.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. *arXiv preprint arXiv:1901.10430*, 2019.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. Personalizing dialogue agents: I have a dog, do you have pets too? *arXiv preprint arXiv:1801.07243*, 2018.