Echo-Attention: Attend Once and Get N Attentions for Free

Anonymous ACL submission

Abstract

This paper proposes echo-attention layers, an efficient method for improving the expressiveness of the self-attention layers without incurring significant parameter or training time costs. The key idea is to iteratively refine the attentional activations via stateful repeated computation, i.e., we compute the activations once and get N refinements (echo-attentions) at a relatively cheap cost. To this end, we introduce an update and state transition function that operates over these attentional activations. Via a set of extensive experiments, we show that this the proposed Echoformer model demonstrates widespread benefits across 21 datasets including language modeling, machine translation, language understanding and question answering.

1 Introduction

Transformer architectures (Vaswani et al., 2017) have become the defacto model choice for sequence modeling and have garnered widespread adoption across natural language processing (Devlin et al., 2018; Raffel et al., 2019; Brown et al., 2020) and computer vision (Dosovitskiy et al., 2020) applications. Fundamentally, Transformers are characterized by stacked blocks of self-attention and standard transformation layers which repeatedly transforms the input sequence as it propagates deeper into the network hierarchy.

At each layer of a standard Transformer model, the self-attention mechanism acts as a form of message passing (Joshi, 2020) and/or routing operation and learns a re-alignment of the intermediate representations. This can also be interpreted as a form of contextual mixing of tokens/representations and is well-established to be crucial to the reasoning capabilities of the Transformer inductive bias. Conceptually, the depth of a Transformer model is an intriguing and important concept since the maximum number of sequential operations in a Transformer model is upper bounded by its depth.

Intuitively, the depth of a Transformer model not only influences its expressiveness but also its ability to handle recursive and/or sequential structures in data. As such, a single layer of self-attention, partially due to its parallel nature, would not be suitable for tasks that require more than a single step of reasoning or processing. Consider the case where a Transformer model has to resolve a nested equation, i.e., $S = \max(a, b, \min(\max(c, d), e))$, a single layer of self-attention would not be able to process this in a single pass since this requires resolving segments of S independently and composing them in subsequent steps (e.g., $S_2 = \max(c, d)$ first and then $\min(S_2, e)$). To this end, Transformer models are usually of some minimum depth, say 4 or more layers, in order to be viable across a wide range of tasks.

In this paper, we propose a new echo-attention layer. The key idea is to have a form of stateful, sequential and separable self-attention module while retaining efficiency and parallelizablility of the Transformer model. Concretely, we decompose and separate the attention activation process into a multi-step iterative process, enabling it to deal with multiple steps of processing within a single attention layer. In contrast to stacking new layers or repeating computation of entire block (Dehghani et al., 2018), our method is more feasible and efficient in practice due to its lightweight nature. We call our model the Echoformer, or echo-attention because the repeated stateful computation resembles an *echo*.

To this end, the intermediate and repeated refinement of the attention modules in a neural network is well-established to be a useful inductive bias and is often linked to reasoning (Sukhbaatar et al., 2015; Cui et al., 2016; Gong and Bowman, 2017; Hu et al., 2017). In our experiments, our qualitative and quantiative analysis shows that the proposed echo-attention has self-refining properties as it learns to continuously update its activations to

become more confident (sharper) if needed. All in all, the proposed echo-attention is more expressive and flexible, and subsumes the vanilla attention by having the capability to revert to it if need be.

We conduct extensive experiments across **21** diverse datasets and tasks in the areas of natural language processing such as machine translation, language modeling (Chelba et al., 2013), question answering (Rajpurkar et al., 2016), summarization (See et al., 2017), and language understanding (Wang et al., 2019). Our overall finding is that Echoformer consistently outperforms vanilla Transformers on all 21 tasks/datasets.

On machine translation and language modeling, we perform systematic studies pertaining to compute-performance trade-offs. We find that on these tasks, Echoformers can outperform Transformers that are deeper and/or have more parameters. For instance, $4L$ Echoformers can outperform $5L$ Transformers on machine translation tasks while being more compute efficient. More often than not, Echoformers with L layers are more performant yet compute efficient as compared to using Transformer model with $L+1$ layers. This demonstrates the improved expressiveness of the Echoformer model. Notably on machine translation tasks, a $4L$ Echoformer can outperform a Transformer model with twice its number of layers (i.e., $8L$).

Our contributions The overall contributions of this work can be summarized as follows:

- We propose echo-attention layers and Echoformer, a new efficient way of imbuing sequential order, state and repeated computation within the self-attention layer. This is achieved by learning to *echo*, reusing existing activations and iteratively refining them via repeated computation.
- The proposed echo-attention is relatively lightweight and has negligible parameter costs. On three systematic studies, we demonstrate that Echoformer can outperform Transformer models that have additional layers. Moreover, with the same number of layers, our experiments show that Echoformers always outperform Transformers.
- We conduct extensive experiments on **21** NLP tasks and datasets. Echoformer consistently outperforms Transformers on all 21 NLP tasks.

2 Echoformers

This section introduces Echoformers and echo-attention. Figure 1 provides a brief illustration of the echo-attention layer.

2.1 Vanilla Self-Attention

We first describe the vanilla self-attention mechanism (Vaswani et al., 2017). The self-attention module first projects the input tensor X to query, key and values. Namely, for each head h and each layer ℓ, this is written as:

$$Q_{h,\ell}, K_{h,\ell}, V_{h,\ell} = W_{q,h,\ell}X, W_{k,h,\ell}X, W_{v,h,\ell}X, \quad (1)$$

where $W_{q,h,\ell}, W_{k,h,\ell}, W_{v,h,\ell}$ are learned parameters. Recall that in the vanilla scaled dot-product self-attention (Vaswani et al., 2017), for layer ℓ and head h, this can be also written as:

$$Y_{h,\ell,i} = \sum_{j=0}^{N} a_{ij,h,\ell} \cdot V_{j,h,\ell}, \quad (2)$$

where $A_{h,\ell} = \text{Softmax}\left(\frac{Q_{h,\ell}K_{h,\ell}^{\top}}{\sqrt{d_k}}\right)$. $A_{h,\ell}$ acts as a form of routing matrix, that guides the routing of representations at layer ℓ. The output at each layer ℓ is defined as $o_\ell = W_o \,\text{concat}\,(Y_{1,\ell} \cdots Y_{H,\ell}) + b_o$. A layer normalization and residual connector to the previous layer wraps around this module, followed by a two-layer positional-wise feed-forward network with ReLU activations.

2.2 Echo Attention

The key idea is to learn to echo, in which we reuse the logits $A_{h,\ell}$ and construct an interpolation (or mixture) of echoed activations as the final output. The echo-attention layer similarly acts upon query, keys and values, which are first learned via linear transformations.

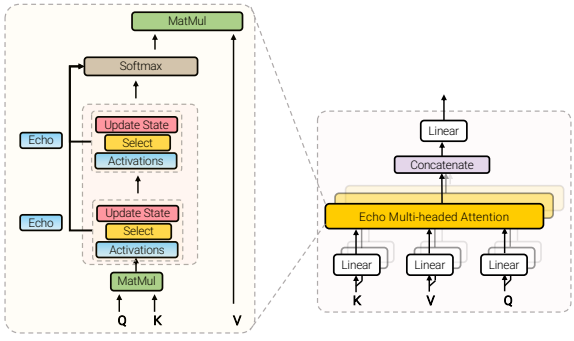

Figure 1: Overview of the proposed Echo-Attention mechanism with two echoes (iterations).

Echoed attention activations are lightweight echos of the original attention activations that constructed in a sequential fashion. There are two design principles and motivators for echo attention.

Learning what to Echo At each step, we learn what to echo. Activations that are selected in step k are already activated and have lesser presence in subsequent echoes. In short, echos are sequentially activated with $E_{0,h,\ell} = A_{h,\ell}$.

Stateful Echo The echo-attention is stateful, i.e., echos are step k are distinct from other steps. We use a state function $U(.)$ to distinguish between echos at each step. The model learns to control of activation strength across echoes.

Generally, for a single head h and layer ℓ, the overall echo attention is defined as:

$$Y_{h,\ell} = \text{Softmax}(E_{0,h,\ell} + E_{1,h,\ell} + \cdots E_{S,h,\ell})V_{h,\ell},$$

where final S is the number of echo steps and $E_{0,h,\ell}$ is the initial activations defined as $Q_{h,\ell}K_{h,\ell}^{\top}$. The function defining $E_{k+1,h,\ell}$ is defined as:

$$E_{k+1,h,\ell} = {}_{h,\ell}\,(E_{k,h,\ell}, Q_{h,\ell}).$$

In short, the $E_{k+1,h,\ell}$ echo is conditioned and produced from $E_{k,h,\ell}$. ${}_{h,\ell}(.)$ is a parameterized function that accepts the previous $E_{k,h,\ell}$ and query tensor as an input. The function ${}_{h,\ell}(.)$ is responsible for (1) learning what to echo and (2) producing the echo activations for step k.

2.2.1 Learning What to Echo

In each step, we learn *what to echo*. When activations are echo-ed, they are softly *erased* (gated) from the matrix and have a lesser presence in the next echo step. The echo-ed attention can be seen as selectively choosing activations to activate (echo) across multiple steps and acts as a form of gating mechanism.

Given $E_{k-1,h,\ell}$ for the $(k-1)$-th echo step for head h and layer ℓ, we first learn a priority score for each activation that denotes an activation's involvement in the current echo. This can be done for each activation, or across row/column dimensions. For simplicity, we tie the priority scores at the token-level so all P_{i*} have the same values for all values of j. To learn the associated priority scores of the ij-th logit at the k-th iteration, we adopt a simple linear projection to a scalar value:

$$p_{k,ij,h,\ell} = (W_{k,h,\ell}Q_{k,i,h,\ell}),$$

where $W_{k,h,\ell} \in \mathbb{R}^d$ are learnable parameters. Intuitively $p_{k,i,j,h,\ell}$ can be interpreted as a form of gating mechanism that learns to re-weight the attention matrix. Given the priority matrix, the echo-ed logits at step k is defined as:

$$E_{k,h,\ell} = P_{k,h,\ell} \odot \hat{E}_{k-1,h,\ell}$$

where $\hat{E}_{k-1}$ is the activation matrix passed to the next echo step after taking into consideration the decision made at step k. This can be expressed as:

$$\hat{E}_{k-1,h,\ell} = U_{k,h,\ell}((1 - P_{k-1,h,\ell})) \odot E_{k-1,h,\ell})), \tag{3}$$

where U maps from $\mathbb{R}^{N\times N} \rightarrow \mathbb{R}^{N\times N}$ is a parameterized function that maps an $N \times N$ matrix to another $N \times N$ matrix. $U_{k,h,\ell}$ is the state function that is used to denote a transition from step k to step $k+1$.

Notably, equation (3) is also reminiscent of the gated linear unit formulation (Dauphin et al., 2017), i.e., $\mathcal{F} = \mathcal{F}_1(.) \odot \mathcal{F}_2(.)$. Moreover, this step infuses nonlinearity into $E_{k,h,\ell}$, which is eventually passed to the Softmax function. Infusing nonlinearity in the Softmax formulation, i.e., $\frac{\sigma(x)e^x}{\sum \sigma(x)e^x}$ improves the expressiveness of the model by enhancing the output set (range) of the log-softmax activation, as demonstrated in the sigsoftmax activation (Kanai et al., 2018).

State Transition Our choice of function $U_{k,h,\ell}(.)$ is a simple learned scaling of its input. The model has the flexibility to learn to control this knob in which the activations can either become stronger or weaker. This can be interpreted as a form of *temperature*. We denote this by:

$$U_{k,h,\ell}(X) = \alpha_{k,h,\ell}X. \tag{4}$$

We experiment with multiple choices where $\alpha \in \mathbb{R}$ can be a scalar parameter that is used to scale the matrix X up or down. We also consider row or column wise position based scaling where $\alpha_{k,h,\ell} \in \mathbb{R}^N$ is a vector and is either broadcast in a row or column fashion. In short, with $U(.)$, we learn to assign a specific magnitude (large or small) with each echo. The model has the flexibility to anneal the activation strength, or learn to increase it over echoes.

Parameter and Computation Costs A defining characteristic of echo-attention layers is that they are very lightly parameterized. For each layer, each

head, each echo step adds d_k parameters for learning $P_{k,i,h,\ell}$. The state update function $U_{k,h,\ell}(.)$ only costs 1 parameter for the scalar version of α and N parameters for the vector (row or column) variation. To keep the echo steps efficient, we do not use a matrix variation of $N \times N$. In the end, each echo attention layer only adds kd_k parameters which is often extremely negligible in the grand scheme of things as most models are in the range of millions of parameters. Echo-attention layers are also relatively cheap, adding $h \times d_k$ operations for learning $P_{k,i,j,h,\ell}$, $N \times N$ operations for summing $E_{k,h,\ell}$. and N^2 operations for scaling via $U_{k,h,\ell}(.)$.

2.3 Discussion

This section discusses several interpretations and qualities of the proposed echo-attention layer.

2.3.1 Echoformer is more expressive and subsumes Transformers

This section shows that Echoformer is more expressive (or at least equally expressive) than vanilla self-attention. We first show that echo-attention subsumes the vanilla self-attention by setting

$$P_{k,h,\ell} = 1 \text{ AND } \alpha_{k,h,\ell} = 1, \ \forall k \geq 1$$

we recover the vanilla self-attention model. Hence, the model has the option to (learn to) revert back to the vanilla self-attention model.

2.3.2 Relation to Sparse and Hard Attention

Sparse and hard attention (Xu et al., 2015) is a form of attention mechanism where attention weights concentrate on minimal number of tokens. The attention distribution is sharper with more probability mass concentrated on some tokens instead of being more uniformly distributed. This section shows that the echo-attention is better at express sparse and/or hard attention, largely due to the iterative updating using the function and the properties of the sigmoid activation.

At each echo step, $P_{k,\ell}$ applies a gating to the attentional activations in order to select which activations to echo in that current step k. Let σ_j be the activations at activation j, then the $\mathrm{Softmax}$ function is written as: $\mathrm{Softmax}(\sigma x) = \frac{e^{\sigma_i x_i}}{\sum_{j=1}^{N} e^{\sigma_j x_j}}$. With $\sigma \in [0, 1]$, the summation denominator of the $\mathrm{Softmax}$ function decreases, the weights that are selected in the echo step k will have sharper distributions than in standard attention. Since the function is biased towards $[0, 1]$, this would intuitively make the distribution of each echo-attention even sharper. We show in our qualitative analysis that echo-attention has the capability to refine and produce sharper distributions when necessary. Likewise, when $P_{k,i,j,\ell}$ is equal to 1, the echo-attention reverts the vanilla self-attention model. Hence, the echo-attention has the option of maintaining the same extent of sparsity as the vanilla attention.

2.3.3 On levels of abstraction: From token-token attention to token-sequence attention

Many early works use attention mechanism as a means to learn relative importance (Nadaraya, 1964; Bahdanau et al., 2014). Given a sequence of tokens, this often refers to a form of parameterized pooling, i.e., $y = \sum^{N} a_i x_i$. That said, in modern self-attention, this notion of relative comparison is shifted to the *token-token* level. Essentially, the activations in self-attention dictate a routing path for each token in the sequence and is normalized (via $\mathrm{Softmax}$) as such. Here, it is easy to see that the model loses relative importance at the *token-sequence* level as routing decisions are made at the token-level. There is no relative importance amongst routing paths.

In echo-attention, the function provides an inductive bias to learn relative importance at a different level of abstraction. In this case, this refers to the *token-sequence* level since the values of $p_{k,h,\ell}$ are tied for each token i. Hence, the design of the proposed inductive bias brings back some flavour of the standard vanilla attention.

2.3.4 Quasi-depth vs Real depth

It is clear that the induced echo-attention mechanism creates a form of pseudo quasi-depth. To simulate *'real'* depth, one would have to use intermediate echo-attentions to create new repesentations, i.e., by multiplying by V before the next echo-step. Our implementation parallelizes this by a initializing a lightweight stateful transition function and avoids this multiplication. We note that this is the intended functionality of the proposed echo-attention as one of its main goals is to **remain efficient**. We postulate that a lightweight stateful mechanism can already bring benefits without having to explicitly create 'real' depth.

3 Experiments

This section describes our experiments. Overall, we conduct experiments on language modeling, machine translation and large-scale pretraining and

Table 1: Controlled experiments on One Billion language modeling (LM1B) benchmark. Keeping the number of params constant, we evaluate the effect of Echo-Attention compared with several alternatives of improving representation capability.

Model	$L=4$		$L=6$		$L=8$	
	#Params	Perplexity	#Params	Perplexity	#Params	Perplexity
Transformer (L)	45M	34.97	51M	33.41	58M	31.65
Transformer $(L+1)$	46M	34.86	51M	33.34	58M	31.54
Echoformer$_S$ (x2)	45M	**34.35** (+1.8%)	51M	32.77 (+2.0%)	58M	31.60 (+0.2%)
Echoformer$_S$ (x4)	45M	34.87 (+0.3%)	51M	33.06 (+1.1%)	58M	31.17 (+1.5%)
Echoformer$_S$ (x6)	45M	35.13 (- 0.5%)	51M	32.74 (+2.0%)	58M	31.22 (+1.4%)
Echoformer$_V$ (x2)	45M	34.97 (+0.0%)	51M	32.74 (+1.8%)	58M	29.37 (+7.7%)
Echoformer$_V$ (x4)	45M	34.95 (+0.1%)	51M	32.87 (+1.6%)	58M	31.14 (+1.6%)
Echoformer$_V$ (x6)	45M	34.87 (+0.3%)	51M	**32.62** (+2.4%)	58M	**29.06** (+8.9%)

finetuning experiments using state-of-the-art T5 models (Raffel et al., 2019).

3.1 Language Modeling Experiments

We conduct experiment on the Google One Billion Language Modeling benchmark (LM1B) (Chelba et al., 2013).

3.1.1 Experimental Setup

We train models for $30K$ steps with a batch size of 256. The maximum sequence length is 256. Models have 8 heads, d_{model} of 1024 and d_{ffn} of 2048. We vary the number of layers $L \in \{4, 6, 8\}$ and the number of echoes for Echoformer in $\{2, 4, 6\}$. We implement our models in FLAX (Heek et al., 2020) and JAX (Bradbury et al., 2018) and train them on 16 TPUv3 chips. We train models for a total of $30K$ steps and report validation (subword) perplexity. In order to verify that echo-attention layers are more efficient as compared to stacking layers, we compare with $L+1$ under *equal* parameterization. We do this by scaling d_{ffn} accordingly. We experiment with both S (scalar) and V (vector) variants of Echoformers.

3.1.2 Experimental Results on Autoregressive Language Modeling

Table 1 reports our results on the LM1B benchmark. We report a systematic study of varying number of layers and comparing Echoformers with Transformers at L and $L+1$ (one additional Transformer layer). Across $L \in \{4, 6, 8\}$ we find that (1) Echoformers consistently outperform Transformers at both L and $L+1$. This ascertains that echo-attention layers are powerful enough to emulate the stacking of an additional Transformer layer.

Table 2: Experimental results on WMT English German. We report quality (BLEU) along with speed (steps per second). Echoformer outperforms Transformers at all levels. Models with $*$ are Echoformer models that outperform Transformers at L+1.

# Layer	Quality (Bleu)		Speed	
	Trans.	Echo.	Trans.	Echo.
3	26.88	**27.40**	3.43	3.06
4	28.10	**28.17***	2.89	2.43
5	27.70	**28.59***	2.19	2.16
6	28.36	**28.53**	1.93	1.84

At the same number of layers, Echoformer also outperforms Transformer of up to $+8.9\%$ relative improvement in terms of validation perplexity at 8 layers. In terms of relative gains across different number of layers, we observe that the gains at 8 layers is much greater than at 6 layers and likewise 4 layers. The relationship between gains from echo-attention and number of standard layers is clearly non-linear. Moreover, the trend of number of echo steps and the number of standard layers is also interesting. Increasing echoes at $L=4$ seems to degrade performnance while the converse is true for $L=\{6, 8\}$. Hence, we believe the ratio of echo layers to standard layers is quite crucial to the effect on model quality. We conduct more extensive studies regarding number of echoes in subsequent sections.

3.2 Machine Translation Experiments

We conduct experiments on machine translation on the WMT'16 English-German task.

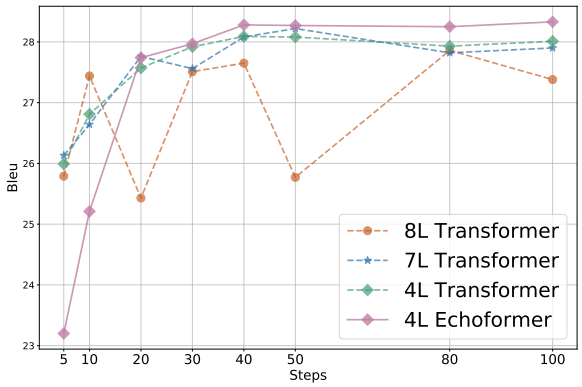

Figure 2: Comparison of 4 layer Echoformer against Transformers of $4, 7, 8$ layers. y-axis denotes BLEU score and y-axis denotes training steps (in thousands).

3.2.1 Experimental Setup

We train on WMT'16 and evaluate on *newstest2014* set. We implement large Transformer and Echoformer models with varied number of layers from $\{3, 4, 5, 6\}$. We use a sequence length of 512 and batch size of 256. Models have d_{ffn} of 4096, 16 heads, dropout of 0.1 and d_{model} of 1024. We train models with *bfloat16* optimizations and report BLEU scores using sacrebleu (Post, 2018) with beam size 4 and length penalty 0.6 at 50K steps. Models are trained on 16 TPUv3 chips. Echoformers are implemented with 4 echoes using the vector variant of the state function. We implement models in Python and JAX (Bradbury et al., 2018). Note that the purpose of this attempt was not to go for state-of-the-art performance, but to understand the effect of echo attention on performance with respect to the number of layers.

3.2.2 Experimental Results on Machine Translation

Table 2 reports results (quality and speed) on our MT experiments. When varying the number of layers, we find that Echoformer outperforms Transformers with the same number of layers. Notably, Echoformers with 4 and 5 layers can outperform Transformers with $L + 1$ layers. This is similarly observed to the experiments in language modeling. As for speed, the cost of using 4 echoes per layer is not substantial and is faster than adding one more Transformer layer. This is even when using 4 echoes per layer and both the encoder and decoder. Hence, we believe that echo-attentions are relatively cost-negligible.

Table 3: Experimental results on question answering, SuperGLUE and summarization tasks.

Model	Transformer	Echoformer
SQuAD	82.2 / 89.7	**82.8 / 90.5**
TriviaQA	22.7 / 27.0	22.7 / **27.1**
WebQA	28.1 / 34.9	**28.6 / 35.5**
CosmosQA	69.4	**69.9**
SGLUE	71.2	**72.8**
BoolQ	78.9	**79.7**
CB	84.8 / 91.1	**89.0 / 91.1**
CoPA	63	**65**
MultiRC	74.5 / 34.1	74.5 / **34.6**
Record	71.0 / 70.0	**71.3 / 70.4**
RTE	79.8	**80.9**
WiC	66.1	**67.2**
WSC	69.2	**74.0**
XSum	39.3/17.3/32.0	**39.4/17.4/32.1**
CNN/DM	40.9/19.0/38.4	**41.2/19.2/38.6**
MultiNews	35.3/13.5/20.4	**35.5/13.7/20.6**

Convergence and Stability Figure 2 compares a $4L$ Echoformer with deep Transformers. We plot the results on machine translation of the four compared models across training iterations. Out of the four variants, we find that the 4L Echoformer performs the best, even outperforming deep Transformers of 7 or 8 layers. Moreover, we find that the Echoformer maintains stability across training epoches, unlike the 8 layer Transformer which demonstrated some extent of instability (fluctuating performance). We also note that the 4L Echoformer takes a little longer to converge, but ends up taking a good position with respect to overall (final) performance.

3.3 Large-Scale Pretraining and Finetuning Experiments

We conduct pretraining and finetuning experiments based on the state-of-the-art T5 model (Raffel et al., 2019) using the open source Mesh Tensorflow[1] and T5[2] library (Shazeer et al., 2018). Specifically, we replace the Transformers in T5 with Echoformers.

3.3.1 Experimental Setup

This section desribes the experimental setup for large-scale experiments.

[1] https://github.com/tensorflow/mesh

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

Pretraining We pretrain on a mixture of unsupervised C4 (Collosal Cleaned Common Crawl corpus) (Raffel et al., 2019) and Glue/SuperGLUE tasks for one million (1M) steps with a batch size of 65536 and sequence length of 512. Experiments are run on 16 TPU-V3 chips and each pretraining takes roughly around 36 hours for pretraining. We adopt Adafactor (Shazeer and Stern, 2018) as the optimizer and use a learning rate schedule inversely proportionate to the square root of the number of steps.

Finetuning This section describes our finetuning experiments. We finetune on SuperGLUE (Wang et al., 2019), which comprises of 8 difficult language understanding benchmarks. Additionally, we also finetune on four question answering benchmarks, (SQuAD (Rajpurkar et al., 2016), TriviaQA (Joshi et al., 2017), WebQA (Berant et al., 2013) and CosmosQA (Huang et al., 2019)), along with three summarization benchmarks (XSum (Narayan et al., 2018), CNN/Dailymail (See et al., 2017) and MultiNews (Fabbri et al., 2019)). All datasets can be found on `https://www.tensorflow.org/datasets/catalog/overview`. In total, evaluate on 17 challenging NLP tasks for finetuning experiments. We train all models for a maximum of $200K$ steps with a constant learning rate tuned amongst $\{0.001, 0.0005\}$. We report peak validation performance.

Model Details We generally use the Transformer base model for experiments which have approximately 220M parameters, have 12 layers, 12 heads, $d_{ffn} = 3076$ and $d_{model} = 768$. Models are sequence to sequence models with both an encoder and decoder. In Echoformer, we replace both encoder and decoder attention with the proposed method. We tune the number of echo layers amongst $\{2, 3, 4\}$. In general, we use the scalar variation of the echo-attention.

3.3.2 Experimental Results on Large-Scale Finetuning

Table 3 report results on large-scale pretraining and finetuning. For QA, we report EM/F1 metrics except for CosmoQA where we only report accuracy. For summarization, we report Rouge-1/2/L. SuperGLUE tasks are mainly accuracy metrics except for Record and MultiRC where we report EM/F1. Results show that Echoformer outperforms a strong Transformer baseline on 17 challenging and difficult NLP tasks. While, performance gains are relatively modest across certain collections of tasks (i.e., summarization), we believe the consistent improvement here relays the strength of the proposed inductive bias.

3.4 Analysis

In this section, we study the effects of echo-attention. We are primarily interested in the inner workings of the echo-attention mechanism.

3.4.1 Effect of State Transition Parameters α

In this section, we study the effect of α across echo steps. We analyze a model trained on language modeling task for $30K$ steps and plot values of learned α. We plot values of α from two different heads across all 6 layers. Figure 3 illustrates the learned α values for a model with 4 echoes.

We observe that learned α differs across layers and heads, i.e., there is a sufficiently distinct behaviour across different heads and layers. We observe that there is both increasing and decreasing echoes, i.e., the trend is not strictly going up or going down for each α value. Moroever, we also observe that most the model learns to diminish α at the fourth echo. We find this intriguing, as this is reminiscent of soft adaptive computation (Graves, 2016) - if the model does not require echoes $\geq N$, it has the complete flexibility to assign a low α to echoes beyond that point.

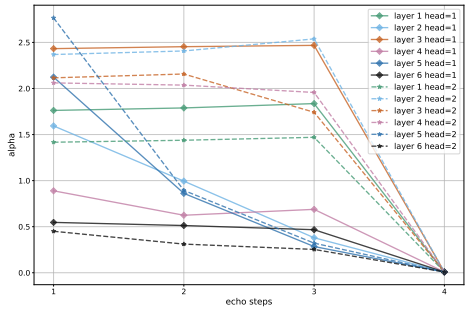

Figure 3: Effect of state transition parameter α across echoes.

3.4.2 Analysis of Attention Activations

Figure 4 and Figure 5 illustrates the attention activations $E_{k,h,\ell}$ for $k \in \{0, 1, 2, 3\}$, $h = 0$ and $\ell = \{2, 4\}$. From the activations, we observe that echo-attention has an effect that forces a sharper distribution of attention scores as opposed the standard attention ($k = 0$). Hence, this reinforces the idea that echo-attention learns to *refine* and iteratively updates the attention weights across echoes.

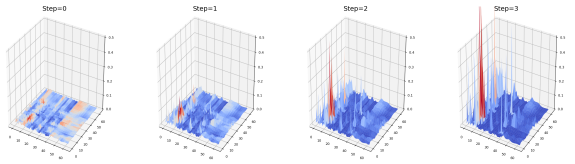

Figure 4: Effect of $E_{k,0,2}$ matrix after each echo step (head 0, layer 2) for $k = \{0, 1, 2, 3\}$.

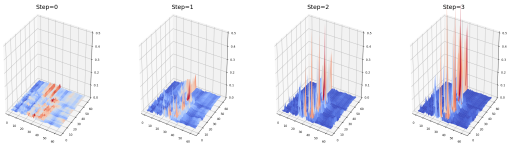

Figure 5: Effect of $E_{k,0,4}$ matrix after each echo step (head 0, layer 4) for $k = \{0, 1, 2, 3\}$.

3.4.3 Analysis of echo priority P

Figure 6 depicts examples of the priority scores P created from the selection function. We extracted weights for 3 samples (denoted sample $\{1, 2, 3\}$ and extract the P values across 4 echoes. Based on our observation, we find that (1) there is a diverse value of P at every echo step, i.e., the echo-attention has a diverse selection of row (token) at each echo step. However, this does not prevent it from putting weight on similar rows, as shown in Figure 6. Next, we also find that the priority distribution across samples is quite different. This perhaps demonstrates that the conditioning on Q enables the model to learn context-dependent priority scores.

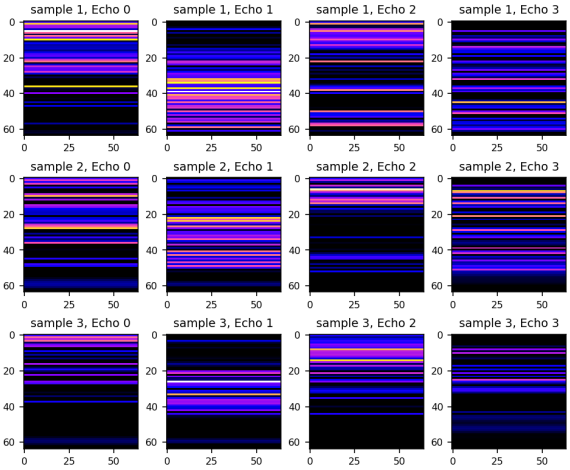

Figure 6: Effect of P matrix across echoes.

4 Related Work

Transformer models (Vaswani et al., 2017) are the dominant choice for sequence processing and has made a tremendous impact on countless of machine learning applications and domains such as natural language processing (Devlin et al., 2018; Brown et al., 2020; Raffel et al., 2019) and computer vision (Dosovitskiy et al., 2020).

Despite the success of Transformers in a wide range of applications, there are some limitations that drive researchers to improve this class of models, hoping to address these constraints. One of the studied aspects is the computational power and expressivity of Transformers (Pérez et al., 2019; Hahn, 2020; Bhattamishra et al., 2020). Hahn (2020) showed that self-attention is computationally restricted and cannot model periodic finite-state languages, nor hierarchical structure, unless the number of layers or heads increases with input length. Some prior works (Tran et al., 2018; Abnar et al., 2020) empirically showed that feed-forward self-attention based models are not as capable as recurrent models in modeling hierarchical structure, which can be crucial in solving some tasks.

To address this issue, Dehghani et al. (2018) proposed the Universal Transformer, introducing a recurrence in depth by repeating the computation of the whole block of a Transformer iterative (Lan et al., 2019), which not only injects a new form of inductive bias into the model, but also increase its computational power. Although this might enables Transformers to better model inherently sequential/hierarchical inputs, it comes with a computational cost that is not necessarily needed to gain such a benefit.

5 Conclusion

We proposed a new echo-attention mechanism. The key idea behind echo-attention is to repeatedly and iteratively refine the attention activations, simulating multiple layers of self-attention within a single layer. This is done in an efficient manner without incurring significant parameter or runtime costs. In order to do so, we introducing priority scoring and state update (transition) functions for modifying the logits. We evaluate Echoformers on 21 diverse and challenging NLP tasks ranging from NLU and commonsense reasoning (i.e., SuperGLUE), generation, question answering, summarization, machine translation and language modeling. For most tasks, we conduct experiments with both large scale pre-training and fine-tuning. On all 21 tasks, Echoformer consistently outperforms a competitive pre-trained T5 Transformer.

References

Samira Abnar, Mostafa Dehghani, and Willem Zuidema. 2020. Transferring inductive biases through knowledge distillation. *arXiv preprint arXiv:2006.00555*.

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

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Satwik Bhattamishra, Arkil Patel, and Navin Goyal. 2020. On the computational power of transformers and its implications in sequence modeling. *arXiv preprint arXiv:2006.09286*.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. JAX: composable transformations of Python+NumPy programs.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2016. Attention-over-attention neural networks for reading comprehension. *arXiv preprint arXiv:1607.04423*.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941. PMLR.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2018. Universal transformers. *arXiv preprint arXiv:1807.03819*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Alexander R. Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir R. Radev. 2019. Multi-news: a large-scale multi-document summarization dataset and abstractive hierarchical model.

Yichen Gong and Samuel R Bowman. 2017. Ruminating reader: Reasoning with gated multi-hop attention. *arXiv preprint arXiv:1704.07415*.

Alex Graves. 2016. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*.

Michael Hahn. 2020. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171.

Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. 2020. Flax: A neural network library and ecosystem for JAX.

Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. 2017. Reinforced mnemonic reader for machine reading comprehension. *arXiv preprint arXiv:1705.02798*.

Lifu Huang, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Cosmos QA: Machine reading comprehension with contextual commonsense reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Chaitanya Joshi. 2020. Transformers are graph neural networks. *The Gradient*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Sekitoshi Kanai, Yasuhiro Fujiwara, Yuki Yamanaka, and Shuichi Adachi. 2018. Sigsoftmax: Reanalysis of the softmax bottleneck. *arXiv preprint arXiv:1805.10829*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Elizbar A Nadaraya. 1964. On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *ArXiv*, abs/1808.08745.

Jorge Pérez, Javier Marinković, and Pablo Barceló. 2019. On the turing completeness of modern neural network architectures. *arXiv preprint arXiv:1901.03429*.

Matt Post. 2018. A call for clarity in reporting bleu scores. *arXiv preprint arXiv:1804.08771*.

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. *CoRR*, abs/1704.04368.

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

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. *arXiv preprint arXiv:1503.08895*.

Ke Tran, Arianna Bisazza, and Christof Monz. 2018. The importance of being recurrent for modeling hierarchical structure. *arXiv preprint arXiv:1803.03585*.

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

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint arXiv:1905.00537*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.