Investigating Recurrent Transformers with Dynamic Halt

Anonymous authors
Paper under double-blind review

Abstract

In this paper, we comprehensively study the inductive biases of two major approaches to augmenting Transformers with a recurrent mechanism: (1) the approach of incorporating a depth-wise recurrence similar to Universal Transformers; and (2) the approach of incorporating a chunk-wise temporal recurrence like Temporal Latent Bottleneck. Furthermore, we propose and investigate novel ways to extend and combine the above methods - for example, we propose a global mean-based dynamic halting mechanism for Universal Transformers and an augmentation of Temporal Latent Bottleneck with elements from Universal Transformer. We compare the models and probe their inductive biases in several diagnostic tasks, such as Long Range Arena (LRA), flip-flop language modeling, ListOps, and Logical Inference. The code is released in the supplementary.

1 Introduction

Transformer (Vaswani et al., 2017) is a highly successful general-purpose attention-based architecture that aims to eliminate any recurrence or convolution. Nevertheless, several works have questioned whether the complete elimination of recurrence has been the right move (Han et al., 2021; Hao et al., 2022; Merrill et al., 2022). Although Transformers generally perform better than Recurrent Neural Networks (RNNs) given enough data[1], several works have shown that RNNs can still outperform Transformers in specific contexts - especially in algorithmic and structure-sensitive tasks under out-of-distribution (OOD) settings (Tran et al., 2018; Han et al., 2021; Shen et al., 2019; Bhattamishra et al., 2023; Liu et al., 2023a; Deletang et al., 2023). Theoretical reasons (Han et al., 2021; Hao et al., 2022; Merrill et al., 2022; Liu et al., 2023b; Merrill & Sabharwal, 2023; Feng et al., 2023) are also suggested for such limitations of Transformers. Overall, these factors naturally raise the question of whether we can get "the best of both worlds" by combining recurrence and Transformers in the same model.

As a motivating example, consider a task like ListOps (Nangia & Bowman, 2018) where we have to solve a nested list of mathematical operations such as: `MAX(1,3,SUM(4,5,MIN(9,7)),4))`. Note that in such a task, unconstrained all-to-all (all tokens to all tokens) interactions cannot accommodate for fixed layer depth in Transformers (Vaswani et al., 2017) compared to RNNs where layer depth can extend adaptively based on sequence length. To solve this task, a model has to process the sequence sequentially because the operations in the outer lists can only be executed once the inner list operations are executed. Keeping that in mind, we can list a few potentially necessary *conditions* or requirements to fulfill for a task like this (and other tasks like program synthesis, program execution, mathematical reasoning, algorithmic reasoning, etc.)[2]:

- **C1:** Ability to operate in arbitrary orders (for example, the innermost operations may occur in any position in the input).
- **C2:** Being equipped with a gating mechanism to keep some hidden states unchanged if needed (for example, outermost values may need to remain unchanged for a while to wait for inner list operations to be completed).

[1] Newer RNNs like RWKV (Peng et al., 2023) or Mamba (Gu & Dao, 2024) could be potential exceptions.
[2] A similar list of requirements was also motivated in Csordás et al. (2022).

- **C3:** Having an adaptive number of layers (according to the input) such that the number of layers can potentially increase indefinitely with higher demand (for example, when a higher depth of reasoning is required according to sequence length as longer sequences are more complex).

Transformers (Vaswani et al., 2017) with their fixed layers (failing C3) and lacking a gating mechanism to control information flow (failing C2) tend to struggle in tasks like ListOps (Tran et al., 2018; Shen et al., 2019; Tay et al., 2021; Csordás et al., 2022) especially in out-of-distribution contexts - for example, when evaluated on data of higher sequence length compared to the length of training samples. Zhou et al. (2023) show that standard Transformers are most suited for length generalization when the problem can be solved by a short RASP-L program, failing otherwise. According to Merrill & Sabharwal (2023), log-precision non-recurrent Transformers fall under complexity class logspace-uniform TC_0.

Given these issues, we consider two broad directions of including recurrence in Transformers to tackle these limitations—one via adding depth-wise recurrence (§4.1, §5.1) and another via adding chunk-wise recurrence (§4.2, §5.2).

1. Depth-wise Recurrence: One way to introduce recurrence is to *repeat the same Transformer block over all tokens*. Such a strategy is employed by Universal Transformer (UT) (Dehghani et al., 2019). In the complete implementation, this is accompanied by a dynamic halting mechanism (Graves, 2016) that adaptively decides when to halt based on the input complexity. We consider Universal Transformer with a modern variant of its halting function (Tan et al., 2023) as the representative model to study for this direction (baseline UT).

2. Chunk-wise Recurrence: Another way to introduce recurrence is to *incorporate temporal recurrence just like RNNs do* except with a stack of Transformer blocks as the recurrent cell, using the past hidden states or some compressed representation as the internal RNN-like memory (Fan et al., 2020). Typically, this temporal recurrence is done at the level of chunks (subsequences) rather than single tokens to exploit the parallelism of Transformers (Didolkar et al., 2022; Ju et al., 2022; Hutchins et al., 2022; Bulatov et al., 2022). In other words, these models can recurrently process one whole chunk at a time rather than single tokens during training. We consider Temporal Latent Bottleneck (TLB) (Didolkar et al., 2022) as the representative model to study for this direction (baseline TLB).

Chunk-wise recurrence can also be backed by some practical engineering-related motivations. For example, Transformer XL (Dai et al., 2019) enables a recurrent cache mechanism to compress past information into a fixed number of memory slots for efficient language modeling. This bypasses the need for expensive attention to an indefinitely growing history of tokens. Other methods have extended this strategy with gradient propagation through the memory slots for long-range language modeling (Hutchins et al., 2022; Ju et al., 2022; Bulatov et al., 2022; Hwang et al., 2024). Some recent works have suggested that recurrent augmentation provides better memory for Transformers (Kuratov et al., 2024; Bulatov et al., 2024).

Contributions: Our main aim in this paper is to investigate and compare the potential and trade-offs of different recurrent inductive biases for Transformers in algorithmic tasks generally found as challenging by standard Transformers. In doing so, we also propose and investigate some reasonable extensions and variations for earlier models. For instance, We propose Gated Universal Transformer (GUT) that uses a transition-aware global dynamic halting mechanism for Universal Transformer and integrates it with a gating mechanism and transition-dynamics-aware halting function. We also propose Gated Universal Transformer Latent Bottleneck (GUTLB) as a combination of GUT (which is a Depth-wise Recurrent Transformer) with TLB (which is a Chunk-wise Recurrent Transformer). We comprehensively investigate the advantages and disadvantages of the two forms of recurrences and our modifications in several diagnostic tasks such as Long Range Arena (LRA), flip-flop language modeling, ListOps, and Logical Inference.

We find that GUT tends to perform better than baseline UT in most of the algorithmic tasks tested here, but so far, we have not found much advantage for using GUTLB over TLB. We also show that chunk-wise recurrence with its fixed recurrent attention window tends to show more robustness than depth-wise recurrence in length generalization and flip-flop language modeling but worse IID performance in certain structure-sensitive tasks like ListOps and logical inference. Note that our purpose is not to chase state-of-the-art results with our proposal but to investigate where they stand empirically.

2 Related Works

Dynamic Halting: Schmidhuber (2012) initially proposed the idea of using a halting neuron for self-delimiting neural programs. The work on Adaptive Computation Mechanism (ACT) by Graves (2016) laid the foundation for a lot of modern dynamic halting approaches. ACT was adapted into the original Universal Transformer (Dehghani et al., 2019) and later modified under probabilistic principles for PonderNet (Banino et al., 2021). Recently, Tan et al. (2023) adapted the ideas similar to Banino et al. (2021) and integrated them with a sparse Mixture of Expert mechanisms in their Sparse Universal Transformer. Our baseline halting mechanism is based on the halting algorithm used in Tan et al. (2023). In another direction, Deep Equilibrium Networks (DEQ) (Bai et al., 2019) implements UT in an implicit layer framework, which can be interpreted as having an implicit dynamic global-halting mechanism that stops the model when the hidden states converge. Xue et al. (2023) explored another alternative direction to standard dynamic halting where they focus on dynamically increasing the input sequence using additional representations generated by a prior RNN representing intermediate computational states to adapt to input complexity. Several works use some form of dynamic halting mechanism for early exit and accelerated inference (Han et al., 2022; Tan & Sim, 2016; Elbayad et al., 2020; Hou et al., 2020; Zhou et al., 2020; Xin et al., 2021; Schuster et al., 2021; 2022; Ainslie et al., 2023; Ji et al., 2023).

Universal Transformer: Universal Transformer (UT) strives to be Turing-complete like earlier works such as Neural Turning Machines (Graves et al., 2014) or Neural GPU (Kaiser & Sutskever, 2016). Several works have attempted to extend UT - some of which we already discussed above (Bai et al., 2019; Banino et al., 2021; Tan et al., 2023). ALBERT (Lan et al., 2020) scales UT with BERT-like training but forgoes the dynamic halting - although certain extensions of it do incorporate it (Balagansky & Gavrilov, 2022). Neural Data Router (Csordás et al., 2022) extends UT with geometric attention and gating. It gets strong results in several algorithmic tasks; however, it also forgoes dynamic halting, relying mainly on a gating mechanism to implicitly halt as needed while making it run up to some user-set upper bound number of layers.

Gating: Besides the work of Csordás et al. (2022), gating was also used in Transformers in an earlier work (Chai et al., 2020). Modern attention methods also integrate it with certain forms of gating (Shazeer, 2020; Hua et al., 2022; Qin et al., 2023a; Yang et al., 2023) inspired by the Gated Linear Unit (GLU) activation function (Dauphin et al., 2017).

3 Preliminaries

Here, we describe the building blocks of our models.

Transformer: In this paper, we focus on Transformer-based encoding models (Vaswani et al., 2017). Our Transformer block typically constitutes an attention layer followed by a feed-forward network. Given a sequence $H_l = (h_l^{(0)}, h_l^{(1)}, \ldots, h_l^{(n-1)}) \in I\!R^{n \times d}$ from layer l, we can formalize the layers as:

$$A_{l+1} = \text{Attention}(Q = H_l, K = H_l, V = H_l) \tag{1}$$

$$H_{l+1} = \text{FeedForward}(A_l) \tag{2}$$

The $\text{Attention}(Q, K, V)$ layer can be formalized more generally as:

$$A = \text{MHA}(LN(Q), LN(K), LN(V)) + Q \tag{3}$$

Here, LN represents layer normalization (Ba et al., 2016) and MHA is the standard multiheaded attention (Vaswani et al., 2017). Eq. 3 shows pre-normalization with residual addition.

The $\text{FeedForward}(A)$ layer can be formalized as:

$$H = (\text{GeLU}(LN(A)W_1 + b_1)W_2 + b_2) + A \tag{4}$$

Here, $W_1 \in I\!R^{d \times d_{ff}}, W_2 \in I\!R^{d_{ff} \times d}, b_1 \in I\!R^{d_{ff}}$, and $b_2 \in I\!R^d$. LN is again layer normalization. GeLU is an activation function (Hendrycks & Gimpel, 2016). Eq. 4 shows a two-layered perceptron network with a pre-normalization strategy and a residual addition.

Algorithm 1 Baseline Halting mechanism at a given timestep t

$\hat{\alpha}_{-1}^{(t)} = 0$
for $l = 1$ **to** L **do**
  if $\sum_{l'=1}^{l-1} \alpha_{l'}^{(t)} < \alpha_{\text{thresh}}$ **then**
    $\mathbf{A}_l = \text{Attention}(\underbrace{\mathbf{H}_{l-1}}_{Q}, \underbrace{\mathbf{S}_{l-1}}_{K}, \underbrace{\mathbf{S}_{l-1}}_{V})$ # here, $S_{l-1} = (s_{l-1}^{(0)}, s_{l-1}^{(1)}, \ldots, s_{l-1}^{(n-1)})$ and $H_{l-1} = (h_{l-1}^{(0)}, h_{l-1}^{(1)}, \ldots, h_{l-1}^{(n-1)})$
    $\mathbf{H}_l = \text{FeedForward}(\mathbf{A}_l)$
    $\hat{\alpha}_{l-1}^{(t)} = \text{halt}(\mathbf{h}_{l-1}^{(t)})$ # token-level halt
    $\alpha_{l-1}^{(t)} = \hat{\alpha}_{l-1}^{(t)} \prod_{l'=-1}^{l-2} (1 - \hat{\alpha}_{l'}^{(t)})$
    $\mathbf{s}_l^{(t)} = \left(1 - \sum_{l'=0}^{l-1} \alpha_{l'}^{(t)}\right) \cdot \mathbf{h}_l^{(t)} + \left(\sum_{l'=0}^{l-1} \alpha_{l'}^{(t)} \cdot \mathbf{h}_{l'}^{(t)}\right)$
  else
    $\mathbf{h}_l^{(t)} = \mathbf{h}_{l-1}^{(t)}$
    $\mathbf{s}_l^{(t)} = \mathbf{s}_{l-1}^{(t)}$
  end if
end for

4 Prior Works

Here, we discuss prior works implementing depth-wise and chunk-wise recurrences in Transformers that we use as baselines.

4.1 Depth-wise Recurrence

4.1.1 Universal Transformer

One approach to tackling condition C3 above via depth-wise recurrence is to use Universal Transformers (UT) (Dehghani et al., 2019) instead of the standard Transformer. Briefly, UT repeatedly applies the same Transformer block in every layer while a halting mechanism decides when to terminate the repetition based on the hidden state. Reusing the same block (thus, sharing the parameters) can allow better function reuse (as similar functional operations can be required in different layers) and allows the model to adapt flexibly to task complexity without any hard preset upper bound. In principle, it is also possible to simulate any vanilla Transformer with unshared parameters with parameter-shared Transformers given enough hidden state size as shown in Bai et al. (2019). However, we still need to set some upper bounds for the number of layers in practice. In the case of UT, we can set an arbitrarily high upper bound (possibly even something that varies with the input length) without worrying about the parameter size because of the sharing mechanism. However, running every input through an arbitrarily high number of layers would still be impractical. Thus, we require some halting mechanism to halt early on as per input requirements efficiently.

Dynamic Halting: The baseline algorithm that we use for UT is shown in Algorithm 1. The algorithm is mainly adapted from Tan et al. (2023) with minor presentational modifications for consistency with our formalism. The standard halting mechanism used in UT is token-level, i.e., different token representations ($h_l^{(t)}$) at different time steps (t) can halt at different layers l. After halting, the token representation will no longer be updated. We implement the $\text{halt}(h_{l-1}^{(t)})$ function (as used in Algorithm 1) as follows:

$$\text{halt}(h_{l-1}^{(t)}) = \sigma_2(\sigma_1(h_{l-1}^{(t)} W_1 + b_1) W_2 + b_2) \tag{5}$$

Here, $h_{l-1}^{(t)} \in \mathbb{R}^d$, $W_1 \in \mathbb{R}^{d \times d}, W_2 \in \mathbb{R}^{d \times 1}, b_1 \in \mathbb{R}^d$, and $b_2 \in \mathbb{R}$. σ_1 is GeLU activation and σ_2 is sigmoid activation. The algorithm implements a soft probabilistic halting procedure to compute halting probability per layer. Here, $\hat{\alpha}_{l-1}^{(t)}$ represents the probability of halting at layer $l-1$ and position t conditioned on the event that the model has not already halted at position t. Thus, the unconditional halting probability at

layer $l-1$ and position t ($\alpha_{l-1}^{(t)}$) can be calculated from probabilistic principles as done in Algorithm 1:

$$\alpha_{l-1}^{(t)} = \hat{\alpha}_{l-1}^{(t)} \prod_{l'=-1}^{l-2} (1 - \hat{\alpha}_{l'}^{(t)}) \tag{6}$$

The final output ($S_L = (s_l^{(0)}, s_l^{(1)}, \ldots, s_l^{(n-1)})$ in Algorithm 1) is a marginalization of all layer outputs (produced before the unconditional halting probability exceeds a threshold α_{thresh}) weighted based on their halting probability which represents the probability of the corresponding representation being the *final* representation. In other words, we softly select the final output per position. As shown in Algorithm 1, this is mathematically represented as:

$$\mathbf{s}_l^{(t)} = \left(1 - \sum_{l'=0}^{l-1} \alpha_{l'}^{(t)}\right) \cdot \mathbf{h}_l^{(t)} + \left(\sum_{l'=0}^{l-1} \alpha_{l'}^{(t)} \cdot \mathbf{h}_{l'}^{(t)}\right) \tag{7}$$

We can early halt (before going through all the layers until the upperbound) if the unconditional halting probability (which is bound to monotonically increase with layer number) at a layer exceeds a hyperparameter threshold (α_{thresh}) because we do not use any representation after that layer for the final output. This way, we can save compute compared to running the model for the maximum number of layers. We allow early stopping both during training and inference.

The baseline algorithm captures the ideas from Banino et al. (2021), which re-formalizes an earlier halting mechanism Graves (2016); Dehghani et al. (2019) in a probabilistically principled manner.

Auxiliary Loss: Similar to Tan et al. (2023), we also set an auxiliary cost (loss) which would discourage halting too late:

$$\mathcal{L}_{\text{ACT}} = \frac{1}{T} \sum_{t=1}^{T} \sum_{l=1}^{L} \alpha_l^{(t)} \cdot l. \tag{8}$$

Here t represents the position in a sequence, T is the sequence length, L is the total number of layers, l is the layer number, $\alpha_l^{(t)}$ is the probability of halting for position t at layer l. Late halting is equivalent to putting more probability mass (high $\alpha_l^{(t)}$ values) to later layers (higher l). Thus, late halting corresponds to high $\mathcal{L}_{\text{ACT}}$ - making it work as a penalty function to discourage late halting. The model is fully terminated when the cumulative halting probability of every timestep passes a certain threshold α_{thresh}.

4.2 Chunk-wise Recurrence

4.2.1 Temporal Latent Bottleneck

Another approach to introducing recurrence and dynamic depth to Transformers is to make Transformers temporally recurrent (Fan et al., 2020), similar to standard Recurrent Neural Networks (RNNs). However, processing one token at a time recurrently with a non-linear Transformer block can be exorbitantly slow during training. Thus, to keep some recurrence for its benefits and also gain practical benefits of parallelism from Transformer - a popular strategy is to first divide the given sequence into temporal chunks (or subsequences) and then recurrently process one whole chunk at a time with a Transformer-based recurrent cell. This results in a chunk-wise recurrent processing. Many approaches explore a similar direction (Hutchins et al., 2022; Ju et al., 2022; Bulatov et al., 2022; Rae et al., 2020; Wu et al., 2022b; Lei et al., 2020). Here, we consider a recent method - Temporal Latent Bottleneck (TLB) (Didolkar et al., 2022) as a representative[3] given that its performance has been well demonstrated in a wide variety of tasks.

The basic algorithm used in TLB is detailed in Algorithm 2. Attention1, Attention2, and Attention3 in the algorithm are the same attention functions (as MHA described before) structurally, just named differently to

[3]We also briefly explored Recurrent Memory Transformer (Bulatov et al., 2022) in LRA in preliminary experiments and found it lacking compared to TLB.

Algorithm 2 Transformer Latent Bottleneck

GIVEN: $\mathbf{M^0}$ #initial memory state
GIVEN: g # g = chunk size (hyperparameter)
$k = \lceil len(H_0)/g \rceil$
chunks = torch.chunk($\mathbf{H}, \mathbf{k}$) #k = number of chunks
for $\mathbf{t}, \mathbf{C}_0$ **in** enumerate(**chunks**) **do**
  #Initially t=0
  # Chunk Processing
  for $l = 1$ **to** L **do**
    $\mathbf{A}_l = \text{Attention1}_l(\underbrace{\mathbf{C}_{l-1}}_{Q}, \underbrace{\mathbf{C}_{l-1}}_{K}, \underbrace{\mathbf{C}_{l-1}}_{V})$
    # Integrating past memory
    $\mathbf{A}_l = \text{Attention2}_l(\underbrace{\mathbf{A}_l}_{Q}, \underbrace{\mathbf{M}^t}_{K}, \underbrace{\mathbf{M}^t}_{V})$
    $\mathbf{C}_l = \text{FeedForward}_l(\mathbf{A}_l)$
  end for
  # Recurrent memory update
  $\mathbf{M}^t_0 = \mathbf{M}^t$
  for $l = 1$ **to** U **do**
    $\mathbf{A}_l = \text{Attention3}_l(\underbrace{\mathbf{M^t}_{l-1}}_{Q}, \underbrace{\mathbf{C}_L}_{K}, \underbrace{\mathbf{C}_L}_{V})$
    $\mathbf{M^t}_l = \text{FeedForward}_l(\mathbf{A}_l)$
  end for
  $\mathbf{M}^{t+1} = \mathbf{M}^t_U$
end for

denote that they have different parameters. $M^t \in I\!R^{k \times d}$ is a sequence of hidden states representing an RNN-like internal memory compressing past information into some fixed memory slots (k represents the number of slots). We slightly modified the original algorithm to make it more comparable to the proposed extension that we discuss later (§5.2). Particularly, in the original method (Didolkar et al., 2022), the Attention2 is not used in every layer - but only periodically, and separate feedforward functions are used after each attention function (unlike in our case). Our implementation baseline simplifies the design.

TLB is interleaved with both self-attention and cross-attention layers. TLB uses self-attention to refine the hidden states in a given chunk. TLB uses cross-attention to integrate information from the past chunks with the hidden states representing the current chunk by having the current chunk states attend to the recurrently encoded memory states representing the past chunks. At the end of processing a given chunk, TLB updates the recurrent memory states with information from the current chunk by having the memory states encoding the past chunks attend to the last hidden states created for the current chunk through the earlier interleaved self-attention and cross-attention layers.

4.3 Limitations

Depth-wise Recurrence: In the standard UT, a token-level halting mechanism is implemented; that is, the halting probability is tracked separately for each time step or position. The model is only fully terminated when all the positions are individually halted (or some upper bound is reached). Moreover, once halted, they are permanently halted - in other words, the representation at the halted position becomes read-only for all subsequent layers.

However, in practice, for hierarchical tasks like mathematical reasoning or program synthesis, the model may need to only temporarily restrict updates at positions (e.g., while waiting for computation in the inner lists to be completed in a task like ListOps). Moreover, the token-level halt increases the number of halting decisions needed to determine the final halt, thus potentially increasing the chances of halting errors per sample (making some positions halt too early). In addition, the token-level halt can limit the writable memory (each position can also be considered as a memory slot with read-write capacities) by halting certain

Algorithm 3 Proposed Global Halting mechanism at a given timestep t

$\hat{\alpha}_{-1}^{(t)} = 0$
for $l = 1$ **to** L **do**
  if $\sum_{l'=1}^{l-1} \alpha_{l'} < \alpha_{\text{thresh}}$ **then**
    $\mathbf{A}_l = \text{Attention}(\underbrace{\mathbf{H}_{l-1}}_{Q}, \underbrace{\mathbf{H}_{l-1}}_{K}, \underbrace{\mathbf{H}_{l-1}}_{V})$ # here, $H_{l-1} = (h_{l-1}^{(0)}, h_{l-1}^{(1)}, \ldots, h_{l-1}^{(n-1)})$
    $\mathbf{H}_l = \text{GatedFeedForward}(\mathbf{H}_{l-1}, \mathbf{A}_l)$ # gating
    $trans = [mean(\mathbf{H}_{l-1}); mean(\mathbf{H}_l)]$ # transition dynamics
    $\hat{\alpha}_{l-1} = \text{halt}(trans)$ # mean-based global halt
    $\alpha_{l-1} = \hat{\alpha}_{l-1} \prod_{l'=-1}^{l-2} (1 - \hat{\alpha}_{l'})$
    $\mathbf{S}_l = \left(1 - \sum_{l'=0}^{l-1} \alpha_{l'}\right) \cdot \mathbf{H}_l + \left(\sum_{l'=0}^{l-1} \alpha_{l'} \cdot \mathbf{H}_{l'}\right)$
  else
    break
  end if
end for

positions too early but without providing any computational saving in return because the model still has to wait for *all* the positions to halt to terminate fully.

Furthermore, typically, in UT, the scoring for halting at any layer l is computed only using the hidden state at layer l ($\mathbf{H}_l$). Thus, it can lack foresight from future layers as to whether the next transition can be helpful or not in deciding whether to halt at the current time.

Chunk-wise Recurrence: To an extent, chunk-wise recurrent Transformers can adapt to input complexity dynamically - for example, the number of recurrent steps they execute depends on the number of chunks that exist, which depends on the given input sequence size. However, it lacks flexibility in the degree to which it adapts. For instance, some chunks may require more compute than others. One of the original motivations for an early dynamic halting mechanism (Adaptive Computation Time or ACT for short) (Graves, 2016) was to adapt compute based on the significance of the token being processed in the given time step. A similar motivation applies to chunk-wise recurrence, too. Different chunks may require different levels of compute resources based on the nature of the input chunk at the time step.

5 Proposed Extensions

In this section, we present our proposed extensions that address the aforementioned limitations.

5.1 Depth-wise Recurrence: Gated Universal Transformer

Given the aforementioned limitations for depth-wise recurrence, we propose Gated Universal Transformer (GUT). In GUT, we make the following modifications:

1. We use an explicit **gating mechanism** so that the model can learn to temporarily restrict updates to positions in a cleaner manner without permanently halting them. This also better addresses C2.
2. We introduce a **global halting mechanism** to decide halting based on a single global state representation.
3. We make the halting decision **transition-aware** by using the information from the immediate future layer.

We show the algorithm of GUT in Algorithm 3. The main changes in Algorithm 3 over Algorithm 1 are highlighted in blue. We discuss the modifications in more detail below.

(1) Gating: To implement gating, we replace the standard feed-forward network with a gated mechanism (GatedFeedForward in the algorithm). The gating mechanism that we use is similar to that in Csordás et al. (2022), but they lack any dynamic halting mechanism for early termination. The processing of $H_l = \text{GatedFeedForward}(H_{l-1}, A_l)$ can be broken down to the following equations:

$$H_{inter} = \text{FeedForward}(A_l) \tag{9}$$

$$G = \sigma_2(\sigma_1(LN(A_l)W_{g1} + b_{g1})W_{g2} + b_{g2}) \tag{10}$$

$$H_l = G \odot H_{inter} + (1 - G) \odot H_{l-1} \tag{11}$$

Here, $W_{g1} \in I\!R^{d \times d_{gff}}, W_{g2} \in I\!R^{d_{gff} \times d}, b_{g1} \in I\!R^{d_{gff}}$, and $b_{g2} \in I\!R^{d}$. σ_1 is GeLU activation and σ_2 is sigmoid activation. The FeedForward function is the same as before.

(2) Global Halt: In our global halt mechanism, the halting function determines the halting probability based on some pooling over the whole hidden sequence rather than the hidden state of a particular time step:

$$\hat{\alpha}_{l-1} = \text{halt}(pool(\mathbf{H^{l-1}})) \tag{12}$$

We use mean-pooling in our specific implementation:

$$\hat{\alpha}_{l} = \text{halt}(mean(\mathbf{H^{l-1}})) \tag{13}$$

Here, $\hat{\alpha}_{l-1}$ represents the probability of halting at layer $l-1$ conditioned on the event the model has not already halted. Note that the current halting probability is no longer indexed with t because it does not vary from position to position. Rather, we have a global halting probability for all positions in a specific layer.

Similar to before, the unconditional halting probability at layer $l-1$ (α_{l-1}) can be calculated from $\hat{\alpha}_{l-1}$ using probabilistic principles (as shown in Algorithm 3):

$$\alpha_{l-1} = \hat{\alpha}_{l-1} \prod_{l'=-1}^{l-2} (1 - \hat{\alpha}_{l'}) \tag{14}$$

The final output S_l is again the marginalization of all layer outputs that are produced before the threshold (α_{thresh}) is exceeded based on the corresponding halting probabilities as shown in Algorithm 3:

$$\mathbf{S}_l = \left(1 - \sum_{l'=0}^{l-1} \alpha_{l'}\right) \cdot \mathbf{H}_l + \left(\sum_{l'=0}^{l-1} \alpha_{l'} \cdot \mathbf{H}_{l'}\right) \tag{15}$$

Early halting works the same way as before. The auxiliary loss for discouraging late halting in Eq. 8 can also be simplified as $\mathcal{L}_{\text{ACT}} = \sum_{l=1}^{L} \alpha_l \cdot l$.

Besides this, when gating is present, it can also simulate token-level halting, if needed, by continuously zeroing out certain positions in every layer.

(3) Transition-based halt: Here, we seek to enrich the input to the halt function (to decide whether to halt at some layer $l-1$) by entering both the hidden state of layer $l-1$ and also the immediate future hidden state ($\mathbf{H}_l$). Thus, the halting function is informed by the future hidden state to consider whether to halt now. Combined with mean-pooling-based global halting, transition-based scoring for the halting-probability can be represented as:

$$\hat{\alpha}_{l-1} = \text{halt}([mean(\mathbf{H^{l-1}}); mean(\mathbf{H^{l}})]) \tag{16}$$

Here, $[;]$ represents concatenation. The shape of the first layer weight W_{h1} in halt function is changed to $I\!R^{2d \times d}$ to account for the increased vector size. Balagansky & Gavrilov (2022) also used a form of transition-aware halting, but their approach lacks gating mechanism and global halting.

Algorithm 4 Gated Universal Transformer Latent Bottleneck

GIVEN: $\mathbf{M^0}$ #initial memory state
GIVEN: g # g = chunk size (hyperparameter)
$k = \lceil len(H_0)/g \rceil$
chunks = torch.chunk($\mathbf{H_0}, \mathbf{k}$) #k = number of chunks
$\hat{\alpha}_{-1}^{(t)} = 0$
for $\mathbf{t}, \mathbf{C}_0$ **in** enumerate(**chunks**) **do**
  #Initially t=0
  for $l = 1$ **to** L **do**
    if $\sum_{l'=1}^{l-1} \alpha_{l'} < \alpha_{\text{thresh}}$ **then**
      $\mathbf{A}_l = \text{Attention1}(\underbrace{\mathbf{C}_{l-1}}_{Q}, \underbrace{\mathbf{C}_{l-1}}_{K}, \underbrace{\mathbf{C}_{l-1}}_{V})$
      $\mathbf{A}_l = \text{Attention2}(\underbrace{\mathbf{A}_l}_{Q}, \underbrace{\mathbf{M}^t}_{K}, \underbrace{\mathbf{M}^t}_{V})$
      $\mathbf{C}_l = \text{GatedFeedForward}(\mathbf{H}_{l-1}, \mathbf{A}_l)$
      $trans = [mean(\mathbf{C}_{l-1}); mean(\mathbf{C}_l)]$
      $\hat{\alpha}_{l-1} = \text{halt}(trans)$ #mean-based global halt
      $\alpha_{l-1} = \hat{\alpha}_{l-1} \prod_{l'=-1}^{l-2} (1 - \hat{\alpha}_{l'})$
      $\mathbf{S}_l = \left(1 - \sum_{l'=0}^{l-1} \alpha_{l'}\right) \cdot \mathbf{C}_l + \left(\sum_{l'=0}^{l-1} \alpha_{l'} \cdot \mathbf{C}_{l'}\right)$
    else
      break
    end if
  end for
  $\mathbf{M}_0^t = \mathbf{M}^t$
  # Recurrent memory update
  for $l = 1$ **to** U **do**
    $\mathbf{A}_l = \text{Attention3}_l(\underbrace{\mathbf{M^t}_{l-1}}_{Q}, \underbrace{\mathbf{S}_L}_{K}, \underbrace{\mathbf{S}_L}_{V})$
    $\mathbf{M^t}_l = \text{FeedForward}_l(\mathbf{A}_l)$
  end for
  $\mathbf{M}^{t+1} = \mathbf{M}_U^t$
end for

5.2 Chunk-wise Recurrence: Gated Universal Temporal Latent Bottleneck

Based on the limitation of chunk-wise recurrence that we discussed, we introduce a dynamic halting mechanism at every chunk. More precisely, we replace the transformer-based recurrent function in TLB with a GUT-based function. We call this combination the Gated Universal Transformer Latent Bottleneck (GUTLB). The algorithm for the combined model is shown in Algorithm 4. The main changes compared to TLB are highlighted in blue.

GUTLB can also be seen as a depth-wise recurrent Transformer-like UT where the attention is locally windowed, and the sequence length regulates the halting time. For instance, the longer the sequence, the more chunks there will be to process, and thus, there will be more recurrent layers of processing until complete termination, irrespective of how early the per chunk-level processing is halted. This also makes good sense as an inductive bias for halting because, intuitively, higher sequence length, in general, can indicate higher complexity and, thus, more computational need.

5.3 Depth-wise vs. Chunk-wise

Here, we discuss one salient distinction between standard depth-wise recurrence (UT, GUT) and chunk-wise recurrence (TLB, GUTLB) that motivates our comparison. For UT or GUT - all tokens are available for attention at every level. This may introduce noise through the dense softmax-based attention, making it

harder to generalize to higher lengths or deal with sensitive tasks (where one token change can change the ground truth) (Liu et al., 2023a; Bhattamishra et al., 2023). Attention sparsifying and sharpening techniques have been used in consideration for making the attention mechanism more robust for these contexts (Liu et al., 2023a). But another alternative can be to bound the attention receptive field to a fixed chunk size by using chunk-wise recurrence (TLB, GUTLB). There is already some evidence that chunk-wise recurrent/recursive Transformers can better perform state tracking (Ju et al., 2022) in sensitive synthetic tasks, parity detection (Chi et al., 2023), and length generalization (Didolkar et al., 2022). We further explore the potential of chunk-wise recurrence here compared to depth-wise recurrence - mostly in contexts where they are not yet evaluated side-by-side.

6 Experiments and Results

6.1 Implementation details

For all models, we implement the multiheaded attention mechanism with FlashAttention2 (Dao, 2024). For positional encoding, we mainly used xPos (Sun et al., 2023b). Other hyperparameter details are presented in Appendix B.

6.2 Choice of Models

Our main aim is to investigate the effects of incorporating different recurrent inductive biases into moderately standard Transformers. Based on that, we concentrate on mainly relevant models and baselines for our experiments. Thus, we ignore other unorthodox Transformers, recursive neural networks, state space models, and such, even though some of them may achieve state-of-the-art results in the datasets we explore. However, we discuss future works related to some of these models in §9.

6.3 Task Motivations

We empirically investigate the models in tasks like ListOps Nangia & Bowman (2018), logical inference (Bowman et al., 2015), Flip-flop language modeling (Liu et al., 2023a), and Long Range Arena (LRA) (Tay et al., 2021). ListOps and logical inference are interesting to explore because Transformers have been shown to underperform in this task despite its simplicity (Shen et al., 2019)[4]. This task can also be a diagnostic testbed for evaluating how well the dynamic halting function is being learned to adapt to the underlying tree depths of the tasks. Solving this task is not as easily hackable through shortcuts or spurious correlations - instead, the model needs to learn to halt at the right time. Flip-flop language modeling is an interesting diagnostic task where vanilla Transformers tend to struggle (Liu et al., 2023a) - thus making it a useful testbed for exploring the benefits of recurrent inductive bias in standard Transformers. LRA is a useful testbed for evaluating the models on a few moderately realistic tasks and also test long-range processing capabilities.

6.4 ListOps

Description: ListOps or List Operations is an arithmetic task for solving nested list of mathematical operations as exemplified in §1. ListOps represents the ListOps split set up similarly to Ray Chowdhury & Caragea (2023a). The training split is the original ListOps (Nangia & Bowman, 2018) with any sample with length > 100 size being filtered (except for the case of results copied from (Shen et al., 2019)). The testing splits consist of several splits of higher lengths, including the original testing split ('near-IID') of mixed lengths (mainly similar size lengths to the training split), some split with a higher number of maximum arguments for each list operation, and the testing split from Long Range Arena (LRA) (Tay et al., 2021) which is both longer and has more arguments.

[4]Neural Data Router (Csordás et al., 2022) is a Transfromer-based model that shows decent performance in ListOps, but they train with $10\times$ more data and short sequence lengths (~ 50). They still struggle with generalization to higher sequence lengths or argument generalization (Ray Chowdhury & Caragea, 2023a).

Model	**near-IID**	**Length Gen.**			**Argument Gen.**		**LRA**
(Lengths)	≤ 1000	200-300	500-600	900-1000	100-1000	100-1000	2000
(Arguments)	≤ 5	≤ 5	≤ 5	≤ 5	10	15	10
Transformer *	$57.4_{0.4}$	—	—	—	—	—	—
UT *	$71.5_{7.8}$	—	—	—	—	—	—
Our Implementations							
Transformer	$73_{0.79}$	$15.6_{1.5}$	$10.4_{0.71}$	$9.6_{0.67}$	$15.8_{1.1}$	$10.3_{1.2}$	$8.1_{0.75}$
UT	$65.8_{7.3}$	$14.1_{1.6}$	$10.3_{0.99}$	$10_{0.44}$	$13.9_{0.59}$	$10.7_{0.29}$	$8.6_{0.81}$
GUT	$\mathbf{83.2_{1.1}}$	$17.6_{2.5}$	$11.1_{0.96}$	$10.5_{0.84}$	$16.2_{2.1}$	$14.2_{2.4}$	$9.8_{1.1}$
−Global Halt	$58.9_{0.89}$	$11_{0.67}$	$11.6_{0.3}$	$9.8_{0.29}$	$10.7_{0.66}$	$10.7_{1.9}$	$10.2_{1.3}$
−Gate	$59.1_{2.7}$	$15.2_{2.5}$	$12.9_{1.9}$	$11.2_{1.8}$	$14.7_{1.8}$	$12.7_{1.1}$	$10.7_{1.2}$
−Transition	$77.7_{5.5}$	$13.8_{0.36}$	$11.9_{0.84}$	$9.7_{0.33}$	$15.5_{1.7}$	$11.9_{1.9}$	$9.6_{0.82}$
TLB	$58.3_{0.73}$	$\mathbf{37.7_{1.6}}$	$\mathbf{30.5_{1.1}}$	$\mathbf{25.0_{2.0}}$	$\mathbf{37.3_{0.75}}$	$\mathbf{37.0_{1.1}}$	$\mathbf{31.1_{1.3}}$
GUTLB	$63.6_{3.5}$	$35.8_{2.4}$	$28.4_{3.4}$	$21_{4.8}$	$34.5_{2.8}$	$33.8_{3.9}$	$26.4_{5.3}$

Table 1: Accuracy on ListOps. ListOps uses the original training set (Nangia & Bowman, 2018) (after removing sequences of length > 100) with the length generalization splits from Havrylov et al. (2019), the argument generalization splits from Ray Chowdhury & Caragea (2023a), and the ListOps-LRA test set from Tay et al. (2021). * represents that the results are taken from Shen et al. (2019). Our models were trained on lengths ≤ 100, depth ≤ 20, and arguments ≤ 5. For our models, we report the mean of 3 runs. We bold the best results in the first and second blocks separately. Subscript represents standard deviation. As an example, $90_{1.0} = 90 \pm 1.0$.

Results: We focus mainly on Transformer-based models here with a standard attention mechanism. We show the results in Table 1. Our Transformer and UT baseline with xPos positional encoding is quite strong - outperforming prior reports with Transformer or UT. Our proposed GUT performs the best in the near-IID setting, but like most other models, it is still poor in out-of-distribution generalization. Our ablations show each of our proposed modifications is helpful for GUT. Replacing a global halt with a token-level halt in GUT (GUT−Global Halt), removing the gating (GUT−Gate), or removing the transition dynamics information (GUT−Transition) - all harm the performance severely in the near-IID setting, and none improves the OOD performance substantially. The transition-based halting choice has the least impact on performance from the ablation.

Interestingly, TLB and GUTLB perform worse in the near-IID setting but better than others for OOD generalization. Their better generalization supports our intuitions that we discussed in §5.3 that chunk-wise recurrent with bound attention could be more robust. One reason why GUT and UT perform better in the near-IID settings could be that they better satisfy condition C1, which is beneficial for the structure of the task, but higher noise from dense softmax in longer contexts may still harm its length generalization (as we discussed in §5.3). Moreover, the chunking in TLB/GUTLB is not input-dependent - thus, may lead to bad chunks unsuited for the structure of the inputs. GUTLB does not benefit much over TLB.

6.5 Logical Inference

Description: Logical Inference (Bowman et al., 2015) is another structure-sensitive task where Transformers struggle to generalize (Tran et al., 2018). In this task, the model has to classify the logical relationship between two given sequences in propositional logic formalism. The models are trained on data with ≤ 6 logical operators and tested on data with a higher number of operators 8-12.

Results: We show the results in Table 2. We find similar results here as in ListOps except TLB and GUTLB perform worse in length generalization than others. Moreover, transition dynamics turn out to be less useful for GUT in this task. Sparse Universal Transformer (SUT) (Tan et al., 2023) recently achieved

Model	**Number of Operations**					
	7	8	9	10	11	12
Transformer*	51	52	51	51	51	48
UT*	51	52	51	51	51	48
SUT†	**98**	**97**	**94**	**90**	**88**	**81**
Our Implementations						
Transformer	$90.88_{0.41}$	$81.26_{1.80}$	$72.56_{2.4}$	$64.71_{4.54}$	$58.45_{3.1}$	$\mathbf{53.88_{4.4}}$
UT	$76.65_{1.24}$	65.47_{1}	$57.4_{1.73}$	$51.02_{1.42}$	$50.08_{1.95}$	$46.62_{0.11}$
GUT	$\mathbf{96.36_{0.23}}$	$\mathbf{84.84_{1.42}}$	$\mathbf{74.80_{1.65}}$	$66.81_{2.43}$	$58.99_{3.69}$	$51.97_{2.83}$
− Global Halt	$80.30_{8.06}$	$67.55_{8.33}$	$58.69_{8.04}$	$54.18_{6.43}$	$50.35_{5.81}$	$45.92_{4.6}$
− Gate	$93.12_{1.30}$	$82.65_{1.55}$	$72.08_{3.12}$	$65.63_{2.48}$	$58.06_{1.9}$	$50.8_{1.94}$
− Transition	$95.24_{0.39}$	$84.79_{0.86}$	$74.62_{1.27}$	$\mathbf{67.68_{1.82}}$	$\mathbf{59.72_{4.1}}$	$52.33_{4.45}$
TLB	$76.66_{0.77}$	$69.83_{1.36}$	$66.22_{1.25}$	$60.09_{0.23}$	$55.36_{1.28}$	$49.75_{1.65}$
GUTLB	$76.17_{2.72}$	$71.39_{1.65}$	$67.37_{1.5}$	$60.87_{1.6}$	$57.95_{0.72}$	$52.36_{2.43}$

Table 2: Test accuracy of models trained on samples with ≤ 6 logical operators and tested on out-of-distribution samples (number of logical operators between 7-12). * indicates that the results are taken from Shen et al. (2019). † indicates that the results are taken from Tan et al. (2023). We bold the best results per block separately. We show the mean of 3 different runs. Subscript represents standard deviation. As an example, $90_{1.0} = 90 \pm 1.0$.

	Length 512			**Length 1024**		
	p_i			p_i		
Model	0.1	0.8	0.98	0.1	0.8	0.98
Transformer	**100**	**100**	90	**100**	**100**	87
UT	**100**	92.8	59	**100**	92.7	59
GUT	**100**	**100**	82	93	**100**	80
TLB	**100**	**100**	94	**100**	**100**	95
GUTLB	**100**	**100**	**95**	**100**	**100**	**96**
TLB Fixed Recurrence	**100**	**100**	93.97	99.99	99.6	92.71

Table 3: Median of models trained on 512 sequence length and $p_i = 0.8$ for flip-flop language modeling (Liu et al., 2023a). Our models are trained for 3 runs.

strong performance in this task, but it uses a sparse MoE alongside dynamic halting. We keep this for future exploration (see §9).

6.6 Flip-flop Language Modeling

Description: Recently Liu et al. (2023a) showed that Transformer-based language models can be brittle. They proposed a minimal synthetic benchmark in the form of Flipflop language modeling that can diagnose some aspect of their architectural brittleness. An example of a sample from this task is "`w 0 i 1 w 1 r 1`". The write instruction (`w`) tells the model to write the following number into the memory, whereas the read instruction (`r`) tells the model to read the latest written number and print it out in the next position. The number following the ignore instruction (`i`) is to be ignored - essentially, they serve as distractors. The task can become challenging for Transformers as its attention mechanism has to ignore the influence of the distractors and zero in on the number associated with the last write instruction, irrespective of the sequence length and the receptive field of attention. We generate the data for this task as follows: We set p_i as the probability of generating the ignore (`i`) instruction in any appropriate context. The probability for generating read or write instruction is then set as $(1-p_i)/2$ similar to Liu et al. (2023a). We set $p_i = 0.8$ and sequence length $= 512$ for the training data; we then test on samples with different p_i ($p_i = 0.1, 0.8, 0.98$) and samples with both the same sequence length (512) and higher sequence length (1024). We generate $160,000$ training samples and $10,000$ samples per test split. The validation set is a mixture of $1,000$ samples

Model	ListOps	Text	Retrieval	Image	Pathfinder
Transformer*	36.37	64.27	57.46	42.44	71.40
Transformer‡	38.37	61.95	80.69	40.57	65.26
TLB†	37.05	81.88	76.91	**57.51**	**79.06**
Our Implementations					
Transformer	$45.05_{1.3}$	$68.01_{0.78}$	$78.65_{0.85}$	$40.09_{0.52}$	$70.26_{1.3}$
UT	$28.50_{12.4}$	$67.33_{0.8}$	$79.15_{0.34}$	$40.60_{0.029}$	$67.17_{1.6}$
GUT	$17.80_{0.0}$	$67.95_{0.06}$	$68.84_{12.9}$	$34.82_{1.5}$	$60.81_{3.5}$
TLB	$\mathbf{50.35}_{0.07}$	$\mathbf{83.55}_{0.15}$	$\mathbf{86.48}_{0.05}$	$53.39_{0.85}$	$69.18_{13.0}$
GUTLB	$40.88_{0.74}$	$82.25_{0.26}$	$85.09_{0.83}$	$53.22_{0.43}$	$69.02_{0.29}$

Table 4: Accuracy on LRA (Tay et al., 2021). * represents that the results are copied from (Tay et al., 2021). ‡ represents that the results are copied from (Chen et al., 2021). † represents that the results are copied from (Didolkar et al., 2022). For our models, we report the mean of 3 runs. Subscript represents standard deviation. As an example, $90_{1.0} = 90 \pm 1.0$.

from each distribution setting evaluated in testing. We show the accuracy of the models in predicting the last read value. We generated the data using the same parameters as used in Liu et al. (2023a) except we also generated higher length test splits for checking length generalization capacity.

Results: We show the results in Table 3. GUT again performs better than UT, but interestingly, basic Transformers outperform both GUT and UT, suggesting that both the latter models are potentially struggling to properly learn the halting function.We find both TLB and GUTLB perform quite well compared to simpler baselines and even GUT - supporting our hypothesis (§5.3) that chunk-wise recurrence may help make Transformers more robust by bounding the explicit size of the receptive field to a fixed chunk size - making it largely invariant to the change of sequence length. Thus, these models can also be considered alternatives to attention-sharpening techniques (Liu et al., 2023a).

To investigate the importance of bounding the chunk size more deeply, we also set up another model variant - TLB with fixed recurrence or a fixed number of chunks (TLB Fixed Recurrence in Table 3). In this model, instead of fixing the chunk size, we fix the number of recurrent steps (or, equivalently, the number of chunks) to five. For this model, Algorithm 2 is be modified such that k would be directly set as 5 and made independent of sequence length ($len(H_0)$) and chunk size (g).

In the standard TLB, when sequence length increases, the number of chunks (and thus, the number of required recurrent steps) increases when the chunk size remains constant. Thus, the Transformer block has to manage the same length chunk in length-generalization settings. On the other hand, in TLB Fixed Recurrence, when sequence length increases, the chunk size increases, whereas the number of chunks (equivalently, the number of recurrent steps) remains constant. Here, the Transformers blocks have to handle larger length chunks in length-generalization settings.

Supporting our hypothesis further, TLB with fixed recurrence performs worse than TLB in length generalization settings. This highlights the importance of bounding the chunk size over and beyond simply having non-linear recurrence.

6.7 LRA

Description: Long Range Arena (LRA) (Tay et al., 2021) is a useful set of tasks to investigate the long-range processing capabilities of different models. We investigate the standard LRA task suite, which has no test set to check length generalization or Out-of-distribution generalization capacities.

Results: We evaluate our key models in the Long Range Arena (LRA) (Tay et al., 2021) in Table 4. Our xPos-based Transformer and TLB baselines, even with the same hyperparameters as in prior work, greatly outperform their prior reported performance and also newer baselines (Chen et al., 2021) in natural language

GUT No Halt Upperbound	**GUT Upperbound**	**GUT**
Upperbound Layers: 10		
2:27 min	3:10 min	2:77 min
Upperbound Layers: 20		
4:40 min	6:10 min	4:53 min
Upperbound Layers: 30		
6:52 min	8:59 min	4:20 min
Upperbound Layers: 40		
9:04 min	11:57 min	5.04 min

Table 5: Training runtime on ListOps for different model per epoch. For GUT we report the average of 5 epochs.

tasks. However, we do not find the same benefit on the vision tasks with any of our models. In fact, some models like TLB with xPos perform significantly worse[5]. GUT mostly keeps up with Transformer, but as expected, it does not perform as well here in a more realistic setting due to limited parameters[6]. Surprisingly, GUT failed to train well in LRA ListOps compared to other models and is generally outperformed by UT. Leaving the ListOps subtasks aside, it is possible that GUT is not particularly helpful in LRA over UT because the other tasks in LRA are still relatively simple and do not require non-linear recurrence and dynamic halting. Notably, our xPos-based TLB shows the strongest performance in natural language tasks among all models, while GUTLB performs worse with its further parameter sharing. GUTLB possibly performs better than GUT because it has more parameters due to two separate attention layers for chunk processing and a separate memory update attention layer with different parameters.

7 Dynamic Halt Runtime Analysis

In Table 5, we analyze the effects of dynamic halt mechanisms on training runtimes. We compare three models: - 1) Gated Universal Transformer (GUT) that we discussed before, 2) GUT Upperbound - which is exactly the same as GUT but without any hard stop (no break statement) when the halting threshold (α_{thresh}) is reached, and 3) GUT No Halt Upperbound - which is again like GUT but lacks any halting function altogether including any computation of halting probability or layer marginalization (it just returns the final layer after going through a fixed number of upperbound layers).

We compare the different models under different upperbound layers (10, 20, 30, and 40). GUT Upperbound is functionally equivalent to GUT. They share the same halting mechanism with layer marginalization at the end. But GUT upperbound has some redundant computation (that does not affect the output) of layers until the final upperbound layer is reached even if the halting threshold is exceeded. Thus, GUT Upperbound reflects the worst-case scenario (where GUT fails to halt at all) for GUT in terms of the empirical runtime.

GUT No Halt Upperbound shows the result of simply running GUT through the full upperbound number of layers without any halting mechanism at all. Thus, in this model, all the halting-related computations (e.g., per layer computation of halting probability) are removed, making each layer faster than that of the full GUT. Thus, GUT No Halt Upperbound can sometimes be faster than GUT even if the former goes through more layers (since each layer of the former is cheaper to compute).

[5]To be fair, TLB's median on Pathfinder is still ~ 77. Only, in one run, it failed to learn the task.

[6]Similar limitations for UT were shown in prior works Narang et al. (2021). Matching the parameters will require either increasing per layer compute by increasing the hidden state size or some additional mixture of expert mechanism (Tan et al., 2023), which we keep for future exploration.

We show the empirical training runtime of one epoch on ListOps for different models and different upperbound layers in Table 5. However, since the layers used in GUT are adaptive and can vary in different epochs, we report the mean empirical training runtime for the first five epochs in the case of GUT.

From the results, unsurprisingly, we find GUT upperbound tends to have more runtime than GUT. However, if the learned halting layer numbers for GUT tend to be not too far from the upperbound layer number, the difference between the three models can be minimal in terms of the runtime. Moreover, because GUT No Halt Upperbound has cheaper layers, it is sometimes empirically faster than GUT, even if GUT may halt a bit earlier. However, we note that as the upper bound is increased to 30 or 40, the runtime of GUT starts to saturate around 5 minutes, whereas the runtime of other models keeps increasing proportionally with the increasing upperbound.

Thus, in cases where we do not have a good estimate for the upper bound a priori or when the ideal number of layers varies wildly (with some examples requiring a low number of layers and some requiring much higher), it can be more viable to run GUT with a very high upper bound rather than GUT No Halt Upperbound (GUT without any halting mechanism) or other model setups.

8 Conclusion

In this paper, we compare side-by-side two forms of recurrences adapted to Transformers - depth-wise recurrence (with Universal Transformer (UT) as a representative) and chunk-wise recurrence (with Transformer Latent Bottleneck (TLB) as a representative). Furthermore, we investigate new extensions to depth-wise and chunk-wise recurrence by creating Gated Universal Transformer (GUT) representing depth-wise recurrence and Gated Universal Latent Transformer Bottleneck (GUTLB) representing chunk-wise recurrence. We find that GUT with global dynamic halting generally tends to perform better than token-level halting in UT in algorithmic tasks. However, all models still struggle with length generalization in ListOps and Logical Inference. GUT/UT (depth-wise recurrence) also performs better than TLB/GUTLB (chunk-wise recurrence) in ListOps and Logical Inference. On the other hand, GUTLB/TLB (chunk-wise recurrence) performs better in LRA and Flip-Flop language modeling.

9 Limitations and Future Work

In this paper, we intended to do a focused study, limiting our scope to consider a few variables - mainly recurrence style and dynamic halting. As discussed below, there are other variables, tasks, and avenues to consider for future study.

More tasks: We only have a few experiments on larger realistic tasks. This is because Universal Transformer-based models are harder to scale parameter-wise (as discussed below) and tend to underperform in more realistic tasks where more parameters are beneficial (Narang et al., 2021; Tay et al., 2023). This would limit any insight we can gain from such tasks when comparing the different models. A potential approach can be to use something like a Mixture of Experts (as suggested below) for better scaling. Still, we keep such architectural choices for future investigation because they are out of the scope of our core focus in the chapter, which is about the recurrence structure and dynamic halting.

Mixture of Experts (MoE): The limitation of UT/GUT is scaling their parameters. The only way to increase their parameters is to increase per-layer compute (for example, by increasing hidden state size). However, since that same layer is repeated, the overall computation cost is increased by multiple folds compared to adding a new layer in a vanilla Transformer. One way to address this issue is to keep a set of neural modules and dynamically select a fixed number of a sparse subset of them in every layer (Cases et al., 2019; Pfeiffer et al., 2023; Le & Venkatesh, 2020; Shen et al., 2023; Tan et al., 2023; Rahaman et al., 2021; Weiss et al., 2022; Csordás et al., 2024). This way, we can increase the parameters by increasing the number of modules while keeping the per-layer compute roughly the same. Such strategies can be further explored in future works in combination with GUT. Such models can also be better explored in more realistic tasks.

Alternative Attention: We mainly focused on vanilla attention efficiently implemented using FlashAttention2 (Dao, 2024). However, there are other variations to consider that are understudied in some of these

diagnostic tasks - for example, geometric attention (Csordás et al., 2022), linear attention (Yang et al., 2023; Qin et al., 2023a; Sun et al., 2023a; Katharopoulos et al., 2020; Wu et al., 2022a), and others (Zimerman & Wolf, 2023; Tay et al., 2022).

Recursive Structures: Besides chunk-wise recurrence, it is also possible to explore more general chunk-wise recursive tree-like structures - for example, within a recursion-in-recursion framework (Ray Chowdhury & Caragea, 2023b) or using the strategy proposed by Chi et al. (Chi et al., 2023). Augmenting Transformers with a stack-based memory for simulating recursive capabilities can also be another avenue to explore (DuSell & Chiang, 2024; Li et al., 2024).

Linear RNNs: Recently, linear RNN based models are also gaining traction (Gu & Dao, 2024; Gu et al., 2022b; Martin & Cundy, 2018; Peng et al., 2023; Gu et al., 2022a; Orvieto et al., 2023; Smith et al., 2023; Qin et al., 2023b; 2024; Beck et al., 2024). There is room for further investigating them in similar contexts with dynamic halting and also further exploring RNN-Transformer hybridization (Ma et al., 2023; Ren et al., 2023; Pilault et al., 2023; Mehta et al., 2023; De et al., 2024; Lieber et al., 2024). However, some recent works suggest that Linear RNNs and adjacent models can still be limited compared to non-linear RNNs for state-tracking purposes (Merrill et al., 2024).

Deep Equilibrium Networks: Another alternative approach to dynamic halting or dynamic computation could be to frame Transformers as a Deep Equilibrium Network (Bai et al., 2019), which can implicitly and adaptively increase recursive iterations based on the sample input.

Large Language Models and Chain of Thoughts: Large Language Models (LLMs) with Chain-of-Thought (COT) reasoning prompts (Wei et al., 2022; Nye et al., 2021) and its extensions (Huang & Chang, 2023) can potentially counteract some of the original problems with fixed-layer Transformers (Zhou et al., 2023; Feng et al., 2023). Essentially, the chain of thought (literally like a chain of thought expressed in text) can serve as intermediate computation results (Merrill & Sabharwal, 2024). LLMs can dynamically vary the size of COT as per the input prompt, thus adapting computation based on the input. In essence, this can be an alternative way to incorporate adaptive computation into Transformers without any adaptive layer increase. However, Transformers may still benefit from dynamically varying the number of internal layers for more flexibility and for better capacity to learn from training examples that lack explicit COT style reasoning or model tasks where the dynamic reasoning aspect is not typically explicitly fully verbalized in natural symbolic language.

References

Joshua Ainslie, Tao Lei, Michiel de Jong, Santiago Ontanon, Siddhartha Brahma, Yury Zemlyanskiy, David Uthus, Mandy Guo, James Lee-Thorp, Yi Tay, Yun-Hsuan Sung, and Sumit Sanghai. CoLT5: Faster long-range transformers with conditional computation. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://openreview.net/forum?id=h96N32OkAx.

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *ArXiv*, abs/1607.06450, 2016. URL http://arxiv.org/abs/1607.06450.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/01386bd6d8e091c2ab4c7c7de644d37b-Paper.pdf.

Nikita Balagansky and Daniil Gavrilov. PALBERT: Teaching ALBERT to ponder. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_WQ6XkVP23f.

Andrea Banino, Jan Balaguer, and Charles Blundell. Pondernet: Learning to ponder. In *8th ICML Workshop on Automated Machine Learning (AutoML)*, 2021. URL https://openreview.net/forum?id=1EuxRTe0WN.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory. *arXiv preprint arXiv:2405.04517*, 2024.

Satwik Bhattamishra, Arkil Patel, Varun Kanade, and Phil Blunsom. Simplicity bias in transformers and their ability to learn sparse Boolean functions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5767–5791, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.317. URL https://aclanthology.org/2023.acl-long.317.

Samuel R. Bowman, Christopher D. Manning, and Christopher Potts. Tree-structured composition in neural networks without tree-structured architectures. In *Proceedings of the 2015th International Conference on Cognitive Computation: Integrating Neural and Symbolic Approaches - Volume 1583*, COCO'15, pp. 37–42, Aachen, DEU, 2015. CEUR-WS.org.

Aydar Bulatov, Yuri Kuratov, and Mikhail Burtsev. Recurrent memory transformer. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=Uynr3iPhksa.

Aydar Bulatov, Yuri Kuratov, Yermek Kapushev, and Mikhail Burtsev. Beyond attention: Breaking the limits of transformer context length with recurrent memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17700–17708, Mar. 2024. doi: 10.1609/aaai.v38i16.29722. URL https://ojs.aaai.org/index.php/AAAI/article/view/29722.

Ignacio Cases, Clemens Rosenbaum, Matthew Riemer, Atticus Geiger, Tim Klinger, Alex Tamkin, Olivia Li, Sandhini Agarwal, Joshua D. Greene, Dan Jurafsky, Christopher Potts, and Lauri Karttunen. Recursive routing networks: Learning to compose modules for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3631–3648, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1365. URL https://aclanthology.org/N19-1365.

Yekun Chai, Shuo Jin, and Xinwen Hou. Highway transformer: Self-gating enhanced self-attentive networks. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6887–6900, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.616. URL https://aclanthology.org/2020.acl-main.616.

Yifan Chen, Qi Zeng, Heng Ji, and Yun Yang. Skyformer: Remodel self-attention with gaussian kernel and nyström method. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 2021.

Ta-Chung Chi, Ting-Han Fan, Alexander I Rudnicky, and Peter J Ramadge. Transformer working memory enables regular language reasoning and natural language length extrapolation. *arXiv preprint arXiv:2305.03796*, 2023.

Róbert Csordás, Kazuki Irie, and Jürgen Schmidhuber. The neural data router: Adaptive control flow in transformers improves systematic generalization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KBQP4A_J1K.

Róbert Csordás, Kazuki Irie, Jürgen Schmidhuber, Christopher Potts, and Christopher D Manning. Moeut: Mixture-of-experts universal transformers. *arXiv preprint arXiv:2405.16039*, 2024.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1285. URL https://aclanthology.org/P19-1285.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=mZn2Xyh9Ec.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. volume 70 of *Proceedings of Machine Learning Research*, pp. 933–941, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/dauphin17a.html.

Soham De, Samuel L Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, et al. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *arXiv preprint arXiv:2402.19427*, 2024.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyzdRiR9Y7.

Gregoire Deletang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Chris Cundy, Marcus Hutter, Shane Legg, Joel Veness, and Pedro A Ortega. Neural networks and the chomsky hierarchy. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WbxHAzkeQcn.

Aniket Rajiv Didolkar, Kshitij Gupta, Anirudh Goyal, Nitesh Bharadwaj Gundavarapu, Alex Lamb, Nan Rosemary Ke, and Yoshua Bengio. Temporal latent bottleneck: Synthesis of fast and slow processing mechanisms in sequence learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=mq-8p5pUnEX.

Brian DuSell and David Chiang. Stack attention: Improving the ability of transformers to model hierarchical patterns. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=XVhm3X8Fum.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJg7KhVKPH.

Angela Fan, Thibaut Lavril, Edouard Grave, Armand Joulin, and Sainbayar Sukhbaatar. Addressing some limitations of transformers with feedback memory. *arXiv preprint arXiv:2002.09402*, 2020.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. Towards revealing the mystery behind chain of thought: A theoretical perspective. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=qHrADgAdYu.

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

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *ArXiv*, abs/1410.5401, 2014. URL http://dblp.uni-trier.de/db/journals/corr/corr1410.html#GravesWD14.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024. URL https://openreview.net/forum?id=AL1fq05o7H.

Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 35971–35983. Curran Associates, Inc., 2022a. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/e9a32fade47b906de908431991440f7c-Paper-Conference.pdf.

Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=uYLFoz1vlAC.

Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *Advances in Neural Information Processing Systems*, 34:15908–15919, 2021.

Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7436–7456, 2022. doi: 10.1109/TPAMI.2021.3117837.

Yiding Hao, Dana Angluin, and Robert Frank. Formal language recognition by hard attention transformers: Perspectives from circuit complexity. *Transactions of the Association for Computational Linguistics*, 10: 800–810, 2022. doi: 10.1162/tacl_a_00490. URL https://aclanthology.org/2022.tacl-1.46.

Serhii Havrylov, Germán Kruszewski, and Armand Joulin. Cooperative learning of disjoint syntax and semantics. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 1118–1128, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1115. URL https://aclanthology.org/N19-1115.

Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *arXiv*, abs/1606.08415, 2016.

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. Dynabert: Dynamic bert with adaptive width and depth. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 9782–9793. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6f5216f8d89b086c18298e043bfe48ed-Paper.pdf.

Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc Le. Transformer quality in linear time. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9099–9117. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/hua22a.html.

Jie Huang and Kevin Chen-Chuan Chang. Towards reasoning in large language models: A survey. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 1049–1065, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.67. URL https://aclanthology.org/2023.findings-acl.67.

DeLesley Hutchins, Imanol Schlag, Yuhuai Wu, Ethan Dyer, and Behnam Neyshabur. Block-recurrent transformers. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=uloenYmLCAo.

Dongseong Hwang, Weiran Wang, Zhuoyuan Huo, Khe Chai Sim, and Pedro Moreno Mengibar. Transformerfam: Feedback attention is working memory. *arXiv preprint arXiv:2404.09173*, 2024.

Yixin Ji, Jikai Wang, Juntao Li, Qiang Chen, Wenliang Chen, and Min Zhang. Early exit with disentangled representation and equiangular tight frame. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 14128–14142, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.889. URL https://aclanthology.org/2023.findings-acl.889.

Da Ju, Stephen Roller, Sainbayar Sukhbaatar, and Jason E Weston. Staircase attention for recurrent processing of sequences. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=NiCJDYpKaBj.

Lukasz Kaiser and Ilya Sutskever. Neural gpus learn algorithms. In *ICLR 2016*, 2016. URL http://arxiv.org/abs/1511.08228.

A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. In search of needles in a 10m haystack: Recurrent memory finds what llms miss. *arXiv preprint arXiv:2402.10790*, 2024.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

Hung Le and Svetha Venkatesh. Neurocoder: Learning general-purpose computation using stored neural programs. *arXiv preprint arXiv:2009.11443*, 2020.

Jie Lei, Liwei Wang, Yelong Shen, Dong Yu, Tamara Berg, and Mohit Bansal. MART: Memory-augmented recurrent transformer for coherent video paragraph captioning. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2603–2614, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.233. URL https://aclanthology.org/2020.acl-main.233.

Jiaoda Li, Jennifer C White, Mrinmaya Sachan, and Ryan Cotterell. A transformer with stack attention. *arXiv preprint arXiv:2405.04515*, 2024.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, et al. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*, 2024.

Bingbin Liu, Jordan T Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Exposing attention glitches with flip-flop language modeling. *arXiv preprint arXiv:2306.00946*, 2023a.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=De4FYqjFueZ.

Xuezhe Ma, Chunting Zhou, Xiang Kong, Junxian He, Liangke Gui, Graham Neubig, Jonathan May, and Luke Zettlemoyer. Mega: Moving average equipped gated attention. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=qNLe3iq2El.

Eric Martin and Chris Cundy. Parallelizing linear recurrent neural nets over sequence length. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HyUNwulC-.

Harsh Mehta, Ankit Gupta, Ashok Cutkosky, and Behnam Neyshabur. Long range language modeling via gated state spaces. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=5MkYIYCbva.

William Merrill and Ashish Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11:531–545, 2023. doi: 10.1162/tacl_a_00562. URL https://aclanthology.org/2023.tacl-1.31.

William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NjNGlPh8Wh.

William Merrill, Ashish Sabharwal, and Noah A. Smith. Saturated transformers are constant-depth threshold circuits. *Transactions of the Association for Computational Linguistics*, 10:843–856, 2022. doi: 10.1162/tacl_a_00493. URL https://aclanthology.org/2022.tacl-1.49.

William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. *arXiv preprint arXiv:2404.08819*, 2024.

Nikita Nangia and Samuel Bowman. ListOps: A diagnostic dataset for latent tree learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pp. 92–99, New Orleans, Louisiana, USA, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-4013. URL https://aclanthology.org/N18-4013.

Sharan Narang, Hyung Won Chung, Yi Tay, Liam Fedus, Thibault Fevry, Michael Matena, Karishma Malkan, Noah Fiedel, Noam Shazeer, Zhenzhong Lan, Yanqi Zhou, Wei Li, Nan Ding, Jake Marcus, Adam Roberts, and Colin Raffel. Do transformer modifications transfer across implementations and applications? In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5758–5773, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.465. URL https://aclanthology.org/2021.emnlp-main.465.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. *arXiv preprint arXiv:2303.06349*, 2023.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, et al. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023.

Jonas Pfeiffer, Sebastian Ruder, Ivan Vulić, and Edoardo Maria Ponti. Modular deep learning. *arXiv preprint arXiv:2302.11529*, 2023.

Jonathan Pilault, Mahan Fathi, Orhan Firat, Christopher Pal, Pierre-Luc Bacon, and Ross Goroshin. Block-state transformers. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=XRTxIBs2eu.

Zhen Qin, Dong Li, Weigao Sun, Weixuan Sun, Xuyang Shen, Xiaodong Han, Yunshen Wei, Baohong Lv, Fei Yuan, Xiao Luo, et al. Scaling transnormer to 175 billion parameters. *arXiv preprint arXiv:2307.14995*, 2023a.

Zhen Qin, Songlin Yang, and Yiran Zhong. Hierarchically gated recurrent neural network for sequence modeling. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023b. URL https://openreview.net/forum?id=P1TCHxJwLB.

Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. Hgrn2: Gated linear rnns with state expansion. *arXiv preprint arXiv:2404.07904*, 2024.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SylKikSYDH.

Nasim Rahaman, Muhammad Waleed Gondal, Shruti Joshi, Peter Vincent Gehler, Yoshua Bengio, Francesco Locatello, and Bernhard Schölkopf. Dynamic inference with neural interpreters. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=IUjt25DtqC4.

Jishnu Ray Chowdhury and Cornelia Caragea. Modeling hierarchical structures with continuous recursive neural networks. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 1975–1988. PMLR, 18–24 Jul 2021. URL http://proceedings.mlr.press/v139/chowdhury21a.html.

Jishnu Ray Chowdhury and Cornelia Caragea. Beam tree recursive cells. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 28768–28791. PMLR, 23–29 Jul 2023a. URL https://proceedings.mlr.press/v202/ray-chowdhury23a.html.

Jishnu Ray Chowdhury and Cornelia Caragea. Recursion in recursion: Two-level nested recursion for length generalization with scalability. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023b. URL https://openreview.net/forum?id=o6yTKfdnbA.

Liliang Ren, Yang Liu, Shuohang Wang, Yichong Xu, Chenguang Zhu, and ChengXiang Zhai. Sparse modular activation for efficient sequence modeling. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=TfbzX6I14i.

Jürgen Schmidhuber. Self-delimiting neural networks. *arXiv*, abs/1210.0118, 2012. URL http://arxiv.org/abs/1210.0118.

Tal Schuster, Adam Fisch, Tommi Jaakkola, and Regina Barzilay. Consistent accelerated inference via confident adaptive transformers. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 4962–4979, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.406. URL https://aclanthology.org/2021.emnlp-main.406.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=uLYc4L3C81A.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Yikang Shen, Shawn Tan, Arian Hosseini, Zhouhan Lin, Alessandro Sordoni, and Aaron C Courville. Ordered memory. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/d8e1344e27a5b08cdfd5d027d9b8d6de-Paper.pdf.

Yikang Shen, Zheyu Zhang, Tianyou Cao, Shawn Tan, Zhenfang Chen, and Chuang Gan. Moduleformer: Learning modular large language models from uncurated data. *arXiv preprint arXiv:2306.04640*, 2023.

Jimmy T.H. Smith, Andrew Warrington, and Scott Linderman. Simplified state space layers for sequence modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Ai8Hw3AXqks.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *ArXiv*, abs/2307.08621, 2023a. URL https://api.semanticscholar.org/CorpusID:259937453.

Yutao Sun, Li Dong, Barun Patra, Shuming Ma, Shaohan Huang, Alon Benhaim, Vishrav Chaudhary, Xia Song, and Furu Wei. A length-extrapolatable transformer. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14590–14604, Toronto, Canada, July 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.816. URL https://aclanthology.org/2023.acl-long.816.

Shawn Tan and Khe Chai Sim. Towards implicit complexity control using variable-depth deep neural networks for automatic speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5965–5969, 2016. doi: 10.1109/ICASSP.2016.7472822.

Shawn Tan, Yikang Shen, Zhenfang Chen, Aaron Courville, and Chuang Gan. Sparse universal transformer. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 169–179, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.12. URL https://aclanthology.org/2023.emnlp-main.12.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena : A benchmark for efficient transformers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=qVyeW-grC2k.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Comput. Surv.*, 55(6), dec 2022. ISSN 0360-0300. doi: 10.1145/3530811. URL https://doi.org/10.1145/3530811.

Yi Tay, Mostafa Dehghani, Samira Abnar, Hyung Chung, William Fedus, Jinfeng Rao, Sharan Narang, Vinh Tran, Dani Yogatama, and Donald Metzler. Scaling laws vs model architectures: How does inductive bias influence scaling? In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 12342–12364, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.825. URL https://aclanthology.org/2023.findings-emnlp.825.

Ke Tran, Arianna Bisazza, and Christof Monz. The importance of being recurrent for modeling hierarchical structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4731–4736, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1503. URL https://aclanthology.org/D18-1503.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_VjQlMeSB_J.

Martin Weiss, Nasim Rahaman, Francesco Locatello, Chris Pal, Yoshua Bengio, Bernhard Schölkopf, Erran Li Li, and Nicolas Ballas. Neural attentive circuits. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 7741–7754. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/32f227c41a0b4e36f65bebb4aeda94a2-Paper-Conference.pdf.

Haixu Wu, Jialong Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Flowformer: Linearizing transformers with conservation flows. In *International Conference on Machine Learning*, 2022a.

Qingyang Wu, Zhenzhong Lan, Kun Qian, Jing Gu, Alborz Geramifard, and Zhou Yu. Memformer: A memory-augmented transformer for sequence modeling. In Yulan He, Heng Ji, Sujian Li, Yang Liu, and Chua-Hui Chang (eds.), *Findings of the Association for Computational Linguistics: AACL-IJCNLP 2022*, pp. 308–318, Online only, November 2022b. Association for Computational Linguistics. URL https://aclanthology.org/2022.findings-aacl.29.

Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. BERxiT: Early exiting for BERT with better fine-tuning and extension to regression. In Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty (eds.), *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 91–104, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.8. URL https://aclanthology.org/2021.eacl-main.8.

Fuzhao Xue, Valerii Likhosherstov, Anurag Arnab, Neil Houlsby, Mostafa Dehghani, and Yang You. Adaptive computation with elastic input sequence. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 38971–38988. PMLR, 23–29 Jul 2023. URL `https://proceedings.mlr.press/v202/xue23e.html`.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023.

Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Josh Susskind, Samy Bengio, and Preetum Nakkiran. What algorithms can transformers learn? a study in length generalization. *arXiv preprint arXiv:2310.16028*, 2023.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. Bert loses patience: Fast and robust inference with early exit. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 18330–18341. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/d4dd111a4fd973394238aca5c05bebe3-Paper.pdf`.

Itamar Zimerman and Lior Wolf. On the long range abilities of transformers. *arXiv preprint arXiv:2311.16620*, 2023.

A Appendix

B Hyperparameter Details

For UT, GUT, or GUTLB the full training objective is: $\mathcal{L} = \mathcal{L}_{\text{main}} + \beta \mathcal{L}_{\text{ACT}}$. Here, $\mathcal{L}_{\text{main}}$ is the main task loss, and β is a scalar trade-off factor for weighing $\mathcal{L}_{\text{ACT}}$. For ListOps (original), we use the same hyperparameters as used in the LRA version of ListOps Tay et al. (2021). We also keep the same hyperparameters for UT, and GUT as Transformers where possible. We tune the β for UT based on validation accuracy on ListOps in $[1, 0.1, 0.01, 0.001]$ and set the value as 0.1. We use the same β in other tasks as well, except in Logical Inference, where we found it better to set $\beta = 0$. We set α_{thresh} to 0.999. We use the same hyperparameters for TLB in ListOps as used in LRA ListOps for TLB in the original repository[7]. For GUTLB we set $L = 5$ but otherwise keep the same hyperparameters as TLB. We generally set $d_{gff} = d_{ff}$ for gated models. We augment all the sequences with a special START and END token before and after the input sequence for all tasks except flipflop. Generally, we use mean pooling for all models for all tasks, except for GUT/GUTLB, where we use the representation in the END position. The decisions are made based on the accuracy of validation on ListOps.

For Logical Inference, we use a Siamese model setup similar to Ray Chowdhury & Caragea (2021). For all the models, we use a mini-batch size of 256, a learning rate of $7e-4$, an AdamW optimizer with weight decay $1e-1$, and linear warmup for 4000 steps. For TLB/GUTLB, we use a chunk size of 10 and 10 memory slots. For UT or GUT, we use an upperbound layer as 15. Other setting details are consistent with ListOps unless mentioned otherwise above.

We use the same hyperparameters for Flipflop language modeling as Liu et al. (2023a). For TLB/GUTLB, we set the chunk size and memory slots as 10. For this task, we train the models in a causal language modeling setup. During training, we only send loss signals to predict the read outputs. The mean-based global halting in GUTLB and GUT is appropriately constrained in this context using cumulative sum to prevent leakage of future data.

For LRA tasks, we use the same hyperparameters as in the original LRA repository for the corresponding models for the natural language tasks. For GUT, we use an upperbound layer as 15. We transfer the rest of the hyperparameters of Transformers to GUT, and of TLB to GUTLB. One exception is that we use a bigger chunk size (100) for the Retrieval task for training efficiency for TLB/GUTLB than originally used; otherwise, they are exorbitantly slower to train. We set $L = 2$ for GUTLB for Retrieval/Text tasks in LRA. For Pathfinder in LRA, we found it better to use the hyperparameters from an alternative work Chen et al. (2021) for Transformer, UT, and GUT.

Each experiment was done in a single Nvidia RTX A6000.

[7] `https://github.com/google-research/long-range-arena`