Stabilizing Transformers for Reinforcement Learning

Emilio Parisotto [1] **H. Francis Song** [2] **Jack W. Rae** [2] **Razvan Pascanu** [2] **Caglar Gulcehre** [2] **Siddhant M. Jayakumar** [2] **Max Jaderberg** [2] **Raphaël Lopez Kaufman** [2] **Aidan Clark** [2] **Seb Noury** [2] **Matthew M. Botvinick** [2] **Nicolas Heess** [2] **Raia Hadsell** [2]

Abstract

Owing to their ability to both effectively integrate information over long time horizons and scale to massive amounts of data, self-attention architectures have recently shown breakthrough success in natural language processing (NLP). Harnessing the transformer's ability to process long time horizons of information could provide a similar performance boost in partially observable reinforcement learning (RL) domains, but the large-scale transformers used in NLP have yet to be successfully applied to the RL setting. In this work we demonstrate that the standard transformer architecture is difficult to optimize, which was previously observed in the supervised learning setting but becomes especially pronounced with RL objectives. We propose architectural modifications that substantially improve the stability and learning speed of the original Transformer and XL variant. The proposed architecture, the Gated Transformer-XL (GTrXL), surpasses LSTMs on challenging memory environments and achieves state-of-the-art results on the multi-task DMLab-30 benchmark suite, exceeding the performance of an external memory architecture. We show that the GTrXL has stability and performance that consistently matches or exceeds a competitive LSTM baseline, including on more reactive tasks where memory is less critical.

1. Introduction

It has been argued that self-attention architectures (Vaswani et al., 2017) deal better with longer temporal horizons than recurrent neural networks (RNNs): by construction, they avoid compressing the whole past into a fixed-size hidden state and they do not suffer from vanishing or exploding gradients in the same way as RNNs. Recent work has empirically validated these claims, demonstrating that self-attention architectures can provide significant gains in performance over the more traditional recurrent architectures such as the LSTM (Dai et al., 2019; Radford et al., 2019; Devlin et al., 2019; Yang et al., 2019). In particular, the Transformer architecture (Vaswani et al., 2017) has had breakthrough success in a wide variety of domains: language modeling (Dai et al., 2019; Radford et al., 2019; Yang et al., 2019), machine translation (Vaswani et al., 2017; Edunov et al., 2018), summarization (Liu & Lapata, 2019), question answering (Dehghani et al., 2018; Yang et al., 2019), multi-task representation learning for NLP (Devlin et al., 2019; Radford et al., 2019; Yang et al., 2019), and algorithmic tasks (Dehghani et al., 2018).

The repeated success of the transformer architecture in domains where sequential information processing is critical to performance makes it an ideal candidate for partially observ-

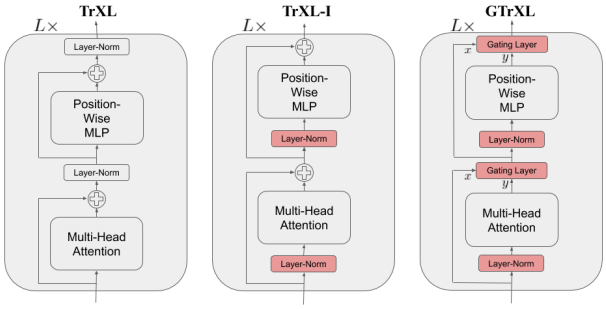

Figure 1. Transformer variants, showing just a single layer block (there are L layers total). **Left:** Canonical Transformer(-XL) block with multi-head attention and position-wise MLP submodules and the standard layer normalization (Ba et al., 2016) placement with respect to the residual connection (He et al., 2016a). **Center:** TrXL-I moves the layer normalization to the input stream of the submodules. Coupled with the residual connections, there is a gradient path that flows from output to input without any transformations. **Right:** The GTrXL block, which additionally adds a gating layer in place of the residual connection of the TrXL-I.

[1]Department of Machine Learning, Carnegie Mellon University, Pittsburgh, USA [2]Google DeepMind, London, UK. Correspondence to: Emilio Parisotto <eparisot@cs.cmu.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

able RL problems, where episodes can extend to thousands of steps and the critical observations for any decision often span the entire episode. Yet, the RL literature is dominated by the use of LSTMs as the main mechanism for providing memory to the agent (Espeholt et al., 2018; Kapturowski et al., 2019; Mnih et al., 2016). Despite progress at designing more expressive memory architectures (Graves et al., 2016; Wayne et al., 2018; Santoro et al., 2018) that perform better than LSTMs in memory-based tasks and partially-observable environments, they have not seen widespread adoption in RL agents perhaps due to their complex implementation, with the LSTM being seen as the go-to solution for environments where memory is required. In contrast to these other memory architectures, the transformer is well-tested in many challenging domains and has seen several open-source implementations in a variety of frameworks [1].

Motivated by the transformer's superior performance over LSTMs and the widespread availability of implementations, in this work we investigate the transformer architecture in the RL setting. In particular, we find that the canonical transformer is significantly difficult to optimize, often resulting in performance comparable to a random policy. This difficulty in training transformers exists in the supervised case as well. Typically a complex learning rate schedule is required (e.g., linear warmup or cosine decay) in order to train (Vaswani et al., 2017; Dai et al., 2019), or specialized weight initialization schemes are used to improve performance (Radford et al., 2019). These measures do not seem to be sufficient for RL. In Mishra et al. (2018), for example, transformers could not solve even simple bandit tasks and tabular Markov Decision Processes (MDPs), leading the authors to hypothesize that the transformer architecture was not suitable for processing sequential information. Our experiments have also verified this observed critical instability in the original transformer model.

However in this work we succeed in stabilizing training with a reordering of the layer normalization coupled with the addition of a new gating mechanism to key points in the submodules of the transformer. Our novel gated architecture, the Gated Transformer-XL (GTrXL) (shown in Figure 1, Right), is able to learn much faster and more reliably and exhibit significantly better final performance than the canonical transformer. We further demonstrate that the GTrXL achieves state-of-the-art results when compared to the external memory architecture MERLIN (Wayne et al., 2018) on the multitask DMLab-30 suite (Beattie et al., 2016). Additionally, we surpass LSTMs significantly on memory-based DMLab-30 levels while matching performance on the more reactive set of levels, as well as significantly outperforming LSTMs on memory-based continuous control and navigation environments. We perform extensive ablations on the GTrXL in challenging environments with both continuous actions and high-dimensional observations, testing the final performance of the various components as well as the GTrXL's robustness to seed and hyperparameter sensitivity compared to LSTMs and the canonical transformer. We demonstrate a consistent superior performance while matching the stability of LSTMs, providing evidence that the GTrXL architecture can function as a drop-in replacement to the LSTM networks ubiquitously used in RL.

[1] www.github.com/tensorflow/tensor2tensor

2. Transformer Architecture and Variants

The transformer network consists of several stacked blocks that repeatedly apply self-attention to the input sequence. The transformer layer block itself has remained relatively constant since its original introduction (Vaswani et al., 2017; Liu et al., 2018; Radford et al., 2019). Each layer consists of two submodules: an attention operation followed by a position-wise multi-layer network (see Figure 1 (left)). The input to the transformer block is an embedding from the previous layer $E^{(l-1)} \in \mathbb{R}^{T \times D}$, where T is the number of time steps, D is the hidden dimension, and $l \in [0, L]$ is the layer index with L being the total number of layers. We assume $E^{(0)}$ is an arbitrarily-obtained input embedding of dimension $[T, D]$, e.g. a word embedding in the case of language modeling or a visual embedding of the per-timestep observations in an RL environment.

Multi-Head Attention: The Multi-Head Attention (MHA) submodule computes in parallel H soft-attention operations for every time step. A residual connection (He et al., 2016a) and layer normalization (LN) (Ba et al., 2016) are then applied to the output (see Appendix D for more details):

$$Y^{(l)} = \text{LN}(E^{(l-1)} + \text{MHA}(E^{(l-1)})) \tag{1}$$

Multi-Layer Perceptron: The Multi-Layer Perceptron (MLP) submodule applies a 1×1 temporal convolutional network $f^{(l)}$ (i.e., kernel size 1, stride 1) over every step in the sequence, producing a new embedding tensor $E^{(l)} \in \mathbb{R}^{T \times D}$. As in (Dai et al., 2019), the network output does not include an activation function. After the MLP, there is a residual update followed by layer normalization:

$$E^{(l)} = \text{LN}(Y^{(l)} + f^{(l)}(Y^{(l)})) \tag{2}$$

Relative Position Encodings: The basic MHA operation does not take sequence order into account explicitly because it is permutation invariant. Positional encodings are a widely used solution in domains like language where order is an important semantic cue, appearing in the original transformer architecture (Vaswani et al., 2017). To enable a much larger contextual horizon than would otherwise be possible, we use the relative position encodings and memory scheme used in (Dai et al., 2019). In this setting, there is an additional $\mathcal{T}$-step memory tensor $M^{(l)} \in \mathbb{R}^{\mathcal{T} \times D}$, which is held

constant during weight updates. The MHA submodule then becomes:

$$Y^{(l)} = \text{LN}(E^{(l-1)} + \text{RMHA}(\text{SG}(M^{(l-1)}), E^{(l-1)})) \quad (3)$$

where SG is a stop-gradient function that prevents gradients flowing backwards during backpropagation. We refer to Appendix D for a more detailed description.

3. Gated Transformer Architectures

3.1. Motivation

While the transformer architecture has achieved breakthrough results in modeling sequences for supervised learning tasks (Vaswani et al., 2017; Liu et al., 2018; Dai et al., 2019), a demonstration of the transformer as a useful RL memory has been notably absent. Previous work has highlighted training difficulties and poor performance (Mishra et al., 2018). When transformers have not been used for temporal memory but instead as a mechanism for attention over the input space, they have had success—notably in the challenging multi-agent Starcraft 2 environment (Vinyals et al., 2019). Here, the transformer was applied solely across Starcraft units and not over time.

In order to alleviate these difficulties, we propose the introduction of powerful gating mechanisms in place of the residual connections within the transformer block, coupled with changes to the order of layer normalization in the submodules. Our gated architecture is motivated by multiplicative interactions having been successful at stabilizing learning across a wide variety of architectures (Hochreiter & Schmidhuber, 1997; Srivastava et al., 2015; Cho et al., 2014), and we empirically see these same improvements in our proposed gated transformer.

Additionally, we propose that the key benefit of the "Identity Map Reordering", where layer normalization is moved onto the "skip" stream of the residual connection, is that it enables an identity map from the input of the transformer at the first layer to the output of the transformer after the last layer. This is in contrast to the canonical transformer, where there are a series of layer normalization operations that nonlinearly transform the state encoding. One hypothesis as to why the Identity Map Reordering improves results is, assuming that the submodules at initialization produce values that are in expectation near zero, the state encoding is passed un-transformed to the policy and value heads, enabling the agent to learn a Markovian policy at the start of training (i.e., the network is initialized such that $\pi(\cdot|s_t, \dots, s_1) \approx \pi(\cdot|s_t)$ and $V^\pi(s_t|s_{t-1}, \dots, s_1) \approx V^\pi(s_t|s_{t-1})$). The original positioning of the layer normalization, on the other hand, would scale down at each layer the information flowing through the skip connection, forcing the model to rely on the residual path. In many environments, reactive behaviours need to be learned before memory-based ones can be effectively utilized, i.e., an agent needs to learn how to walk before it can learn how to remember where it has walked. We provide a more in depth discussion and empirical results demonstrating the re-ordered layer norm allows untransformed state embeddings in App. F.

3.2. Identity Map Reordering

Our first change is to place the layer normalization on only the input stream of the submodules, a modification described in several previous works (He et al., 2016b; Radford et al., 2019; Baevski & Auli, 2019). The model using this *Identity Map Reordering* is termed TrXL-I in the following, and is depicted visually in Figure 1 (center). Because the layer norm reordering causes a path where two linear layers are applied in sequence, we apply a ReLU activation to each sub-module output before the residual connection (see Appendix D for equations). The TrXL-I already exhibits a large improvement in stability and performance over TrXL (see Section 4.3.1).

3.3. Gating Layers

We further improve performance and optimization stability by replacing the residual connections in Equations 3 and 2 with gating layers. We call the gated architecture with the identity map reordering the *Gated Transformer(-XL)* (GTrXL). The final GTrXL layer block is written below:

$$\begin{aligned}
\overline{Y}^{(l)} &= \text{RMHA}(\text{ LN}([\text{SG}(M^{(l-1)}), E^{(l-1)}])) \\
Y^{(l)} &= g^{(l)}_{\text{MHA}}(E^{(l-1)}, \text{ReLU}(\overline{Y}^{(l)})) \\
\overline{E}^{(l)} &= f^{(l)}(\text{LN}(Y^{(l)})) \\
E^{(l)} &= g^{(l)}_{\text{MLP}}(Y^{(l)}, \text{ReLU}(\overline{E}^{(l)}))
\end{aligned}$$

where g is a gating layer function. A visualization of our final architecture is shown in Figure 1 (right), with the modifications from the canonical transformer highlighted in red. In our experiments we ablate a variety of gating layers with increasing expressivity:

Input: The gated input connection has a sigmoid modulation on the input stream, similar to the short-cut-only gating from (He et al., 2016b):

$$g^{(l)}(x, y) = \sigma(W_g^{(l)} x) \odot x + y$$

Output: The gated output connection has a sigmoid modulation on the output stream:

$$g^{(l)}(x, y) = x + \sigma(W_g^{(l)} x - b_g^{(l)}) \odot y$$

Highway: The highway connection (Srivastava et al., 2015) modulates both streams with a sigmoid:

$$g^{(l)}(x, y) = \sigma(W_g^{(l)} x + b_g^{(l)}) \odot x + (1 - \sigma(W_g^{(l)} x + b_g^{(l)})) \odot y$$

Sigmoid-Tanh: The sigmoid-tanh (SigTanh) gate (Van den Oord et al., 2016) is similar to the Output gate but with an additional tanh activation on the output stream:

$$g^{(l)}(x, y) = x + \sigma(W_g^{(l)} y - b) \odot \tanh(U_g^{(l)} y)$$

Gated-Recurrent-Unit-type gating: The Gated Recurrent Unit (GRU) (Chung et al., 2014) is a recurrent network that performs similarly to an LSTM (Hochreiter & Schmidhuber, 1997) but has fewer parameters. We adapt its powerful gating mechanism as an untied activation function in depth:

$$r = \sigma(W_r^{(l)} y + U_r^{(l)} x), \qquad z = \sigma(W_z^{(l)} y + U_z^{(l)} x - b_g^{(l)})$$
$$\hat{h} = \tanh(W_g^{(l)} y + U_g^{(l)}(r \odot x))$$
$$g^{(l)}(x, y) = (1 - z) \odot x + z \odot \hat{h}.$$

Gated Identity Initialization: We have claimed that the Identity Map Reordering aids policy optimization because it initializes the agent close to a Markovian policy / value function. If this is indeed the cause of improved stability, we can explicitly initialize the various gating mechanisms to be close to the identity map. This is the purpose of the bias $b_g^{(l)}$ in the applicable gating layers. We later demonstrate in an ablation that initially setting $b_g^{(l)} > 0$ can greatly improve learning speed.

4. Experiments

In this section, we provide experiments on a variety of challenging single and multi-task RL domains: DMLab-30 (Beattie et al., 2016), Numpad and Memory Maze (see App. Fig. 8 and 9). Crucially we demonstrate that the proposed Gated Transformer-XL (GTrXL) not only shows substantial improvements over LSTMs on memory-based environments, but suffers no degradation of performance on reactive environments. The GTrXL also exceeds MERLIN (Wayne et al., 2018), an external memory architecture which used a Differentiable Neural Computer (Graves et al., 2016) coupled with auxiliary losses, surpassing its performance on both memory and reactive tasks.

For all transformer architectures except when otherwise stated, we train relatively deep 12-layer networks with embedding size 256 and memory size 512. These networks are comparable to the state-of-the-art networks in use for small language modeling datasets (see enwik8 results in (Dai et al., 2019)). We chose to train deep networks in order to demonstrate that our results do not necessarily sacrifice complexity for stability, i.e. we are not making transformers stable for RL simply by making them shallow. Our networks have receptive fields that can potentially span any episode in the environments tested, with an upper bound on the receptive field of 6144 (12 layers $\times$ 512 memory (Dai et al., 2019)). Future work will look at scaling transformers in RL even further, e.g. towards the 52-layer network in (Radford et al., 2019). See App. C for experimental details.

Model	Mean HNR	Mean HNR, 100-capped
LSTM	99.3 ± 1.0	84.0 ± 0.4
TrXL	5.0 ± 0.2	5.0 ± 0.2
TrXL-I	107.0 ± 1.2	87.4 ± 0.3
MERLIN@100B	115.2	89.4
GTrXL (GRU)	117.6 ± 0.3	89.1 ± 0.2
GTrXL (Input)	51.2 ± 13.2	47.6 ± 12.1
GTrXL (Output)	112.8 ± 0.8	87.8 ± 0.3
GTrXL (Highway)	90.9 ± 12.9	75.2 ± 10.4
GTrXL (SigTanh)	101.0 ± 1.3	83.9 ± 0.7

Table 1. Final human-normalized return (HNR) averaged across all 30 DMLab levels for baselines and GTrXL variants. We also include the 100-capped score where the per-level mean score is clipped at 100, providing a metric that is proportional to the percentage of levels that the agent is superhuman. We see that the GTrXL (GRU) surpasses LSTM by a significant gap and exceeds the performance of MERLIN (Wayne et al., 2018) trained for 100 billion environment steps. The GTrXL (Output) and the proposed reordered TrXL-I also surpass LSTM but perform slightly worse than MERLIN and are not as robust as GTrXL (GRU) (see Sec. 4.3.2). We sample 6-8 hyperparameters per model. We include standard error over runs.

For all experiments, we used V-MPO (Song et al., 2020), an on-policy adaptation of Maximum a Posteriori Policy Optimization (MPO) (Abdolmaleki et al., 2018a;b) that performs approximate policy iteration based on a learned state-value function $V(s)$ instead of the state-action value function used in MPO. Rather than directly updating the parameters in the direction of the policy gradient, V-MPO uses the estimated advantages to first construct a target distribution for the policy update subject to a sample-based KL constraint, then calculates the gradient that partially moves the parameters toward that target, again subject to a KL constraint. V-MPO was shown to achieve state-of-the-art results for LSTM-based agents on multi-task DMLab-30. As we want to focus on architectural improvements, we do not alter or tune the V-MPO algorithm differently than the settings originally presented in that paper for the LSTM architecture, which included hyperparameters sampled from a wide range.

4.1. Transformer as Effective RL Memory Architecture

We first present results of the best performing GTrXL variant, the GRU-type gating, against a competitive LSTM baseline, demonstrating a substantial improvement on the DMLab-30 domain (Beattie et al., 2016). DMLab-30 (shown in App. Fig. 9) is a large-scale, multitask benchmark comprising 30 first-person 3D environments with image observations and has been widely used as a benchmark for architectural and algorithmic improvements (Wayne et al.,

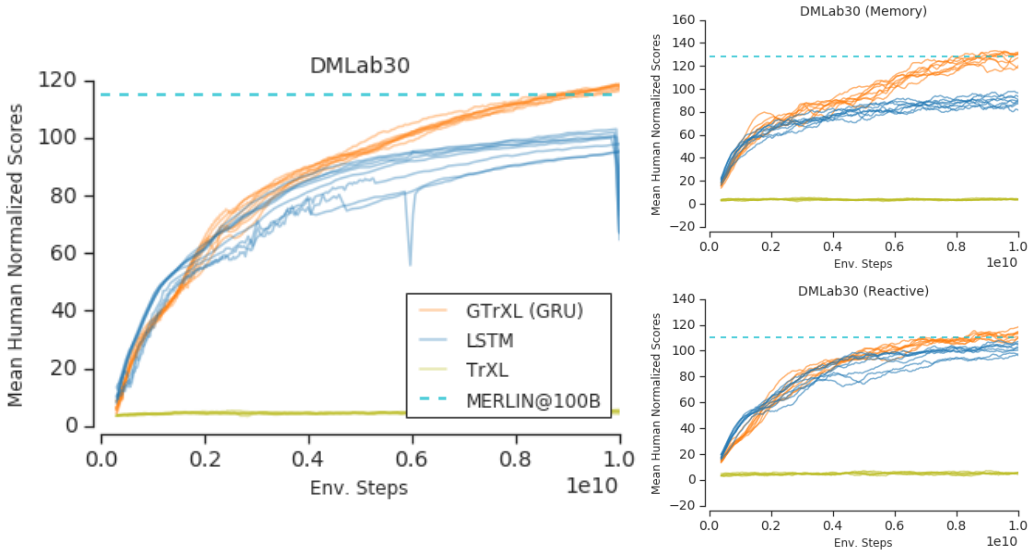

Figure 2. Average return on DMLab-30, re-scaled such that a human has mean 100 score on each level and a random policy has 0. **Left:** Results averaged over the full DMLab-30 suite. **Right:** DMLab-30 partitioned into a "Memory" and "Reactive" split (described in Table 14, Appendix). The GTrXL has a substantial gain over LSTM in memory-based environments, while even slightly surpassing performance on the reactive set. We plot 6-8 hyperparameter settings per architecture (see Appendix C). MERLIN scores obtained from personal communication with the authors.

2018; Espeholt et al., 2018; Kapturowski et al., 2019; Hessel et al., 2018). The levels test a wide range of agent competencies such as language comprehension, navigation, handling of partial observability, memory, planning, and other forms of long horizon reasoning, with episodes lasting over 4000 environment steps. We choose the DMLab-30 benchmark over Atari-57 (Bellemare et al., 2013) due to tasks in DMLab-30 explicitly being constructed to test an agent's memory and long-horizon reasoning, in contrast to Atari-57 which contains mostly fully-observable games. Nevertheless, we also report Atari-57 results in App. E. Figure 2 shows mean return over all levels as training progresses, where the return is human normalized as done in previous work (meaning a human has a per-level mean score of 100 and a random policy has a score of 0), while Table 1 has the final performance at 10 billion environment steps. The GTrXL has a significant gap over a 3-layer LSTM baseline trained using the same V-MPO algorithm. Furthermore, we included the final results of a previously-published external memory architecture, MERLIN (Wayne et al., 2018). Because MERLIN was trained for 100 billion environment steps with a different algorithm, IMPALA (Espeholt et al., 2018), and also involved an auxiliary loss critical for the memory component to function, the learning curves are not directly comparable and we only report the final performance of the architecture as a dotted line. Despite the differences, our results demonstrate that the GTrXL can match the previous state-of-the-art on DMLab-30. An informative split between a set of memory-based levels and more reactive ones (listed in App. Table 14) reveals that our model specifically has large improvements in environments where memory plays a critical role. Meanwhile, GTrXL also shows improvement over LSTMs on the set of reactive levels, as memory can still be used in some of these levels.

4.2. Scaling with Memory Horizon

We next demonstrate that the GTrXL scales better compared to an LSTM when an environment's temporal horizon is increased, using the "Numpad" continuous control task of (Humplik et al., 2019) which allows an easy combinatorial increase in the temporal horizon. In Numpad, a robotic agent is situated on a platform resembling the 3x3 number pad of a telephone (generalizable to $N \times N$ pads). The agent can interact with the pads by colliding with them, causing them to be activated (visualized in the environment state as the number pad glowing). The goal of the agent is to activate a specific sequence of up to N^2 numbers, but without knowing this sequence a priori. The only feedback the agent gets is by activating numbers: if the pad is the next one in the sequence, the agent gains a reward of +1, otherwise all activated pads are cleared and the agent must restart the sequence. Each correct number in the sequence only provides reward once, i.e. each subsequent activation of that number will no longer provide rewards. Therefore the agent must explicitly develop a search strategy to determine the correct pad sequence. Once the agent completes the full sequence, all pads are reset and the agent gets a chance to repeat the sequence again for more reward. This means higher reward directly translates into how well the pad sequence has been memorized. An image of the scenario is provided in App. Figure 8. There is the restriction that contiguous pads in the

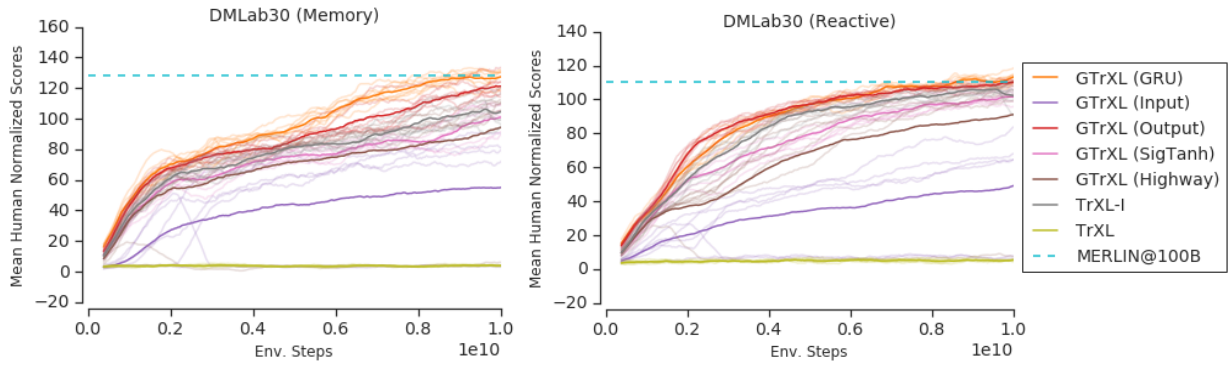

Figure 3. Learning curves for the gating mechanisms, along with MERLIN score at 100 billion frames as a reference point. We can see that the GRU performs as well as any other gating mechanism on the reactive set of tasks. On the memory environments, the GRU gating has a significant gain in learning speed and attains the highest final performance at the fastest rate. We plot both mean (bold) and the individual 6-8 hyperparameter samples per model (light).

sequence must be contiguous in space, i.e. the next pad in the sequence can only be in the Moore neighborhood of the previous pad. No pad can be pressed twice in the sequence.

We present two results in this environment in Figure 5. The first measures the final performance of the trained models as a function of the pad size. We can see that LSTM performs badly on all 3 pad sizes, and performs worse as the pad size increases from 2 to 4. The GTrXL performs much better, and almost instantly solves the environment with its much more expressive memory. On the center and right images, we provide learning curves for the 2×2 and 4×4 Numpad environments, and show that even when the LSTM is trained twice as long it does not reach GTrXL's performance.

4.3. Gating Variants + Identity Map Reordering

We demonstrated that the GRU-type-gated GTrXL can achieve state-of-the-art results on DMLab-30, surpassing both a deep LSTM and an external memory architecture, and also that the GTrXL has a memory which scales better with the memory horizon of the environment. However, the question remains whether the expressive gating mechanisms of the GRU could be replaced by simpler alternatives. In this section, we perform extensive ablations on the gating variants described in Section 3.3, and show that the GTrXL (GRU) has improvements in learning speed, final performance and optimization stability over all other models, even when controlling for the number of parameters.

4.3.1. Performance Ablation

We first report the performance of the gating variants in DMLab-30. Table 1 and Figure 3 show the final performance and training curves of the various gating types in both the memory / reactive split, respectively. The canonical TrXL completely fails to learn, while the TrXL-I improves over the LSTM. Of the gating varieties, the GTrXL (Output) can recover a large amount of the performance of the GTrXL (GRU), especially in the reactive set, but as shown in Sec. 4.3.2 is generally far less stable. The GTrXL (Input) performs worse than even the TrXL-I, reinforcing the identity map path hypothesis. Finally, the GTrXL (Highway) and GTrXL (SigTanh) are more sensitive to the hyperparameter settings compared to the alternatives, with some settings doing worse than TrXL-I.

Model	Mean Human Norm. Score	# Param. Millions
LSTM	99.3 ± 1.0	9.25M
Large LSTM	103.5 ± 0.9	51.3M
TrXL	5.0 ± 0.2	28.6M
TrXL-I	107.0 ± 1.2	28.6M
Thin GTrXL (GRU)	111.5 ± 0.6	22.4M
GTrXL (GRU)	117.6 ± 0.3	66.4M
GTrXL (Input)	51.2 ± 13.2	34.9M
GTrXL (Output)	112.8 ± 0.8	34.9M
GTrXL (Highway)	90.9 ± 12.9	34.9M
GTrXL (SigTanh)	101.0 ± 1.3	41.2M

Table 2. Parameter-controlled ablation. We report standard error of the means of 6-8 runs per model.

Model	% Diverged
LSTM	0%
TrXL	0%
TrXL-I	16%
GTrXL (GRU)	0%
GTrXL (Output)	12%

Table 3. Percentage of the 25 parameter settings where the training loss diverged within 2 billion env. steps. We do not report numbers for GTrXL gating types that were unstable in DMLab-30. For diverged runs we plot the returns in Figure 6 as 0 afterwards.

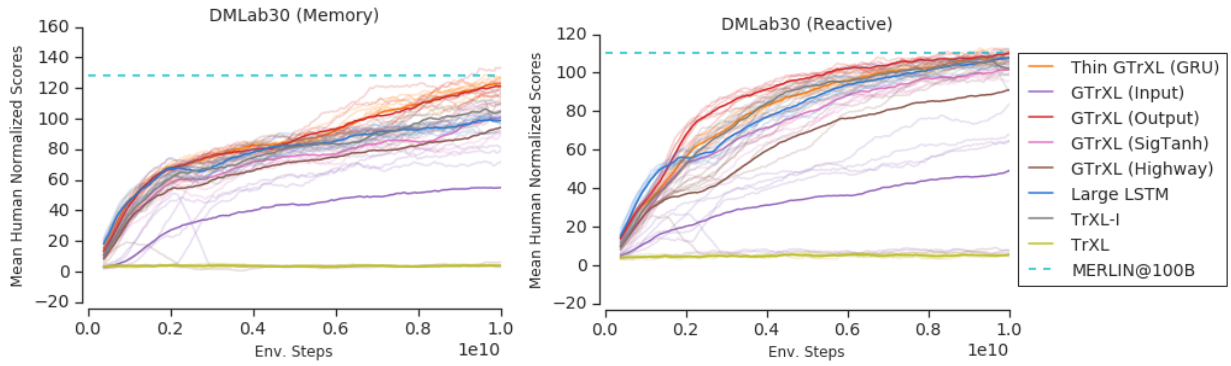

Figure 4. Learning curves comparing a thinner GTrXL (GRU) with half the embedding dimension of the other presented gated variants and TrXL baselines. The Thin GTrXL (GRU) has fewer parameters than any other model presented but still matches the performance of the best performing counterpart, the GTrXL (Output), which has over 10 million more parameters. We plot both mean (bold) and 6-8 hyperparameter settings (light) per model.

4.3.2. Hyperparameter and Seed Sensitivity

Beyond improved performance, we next demonstrate a significant reduction in hyperparameter and seed sensitivity for the GTrXL (GRU) compared to baselines and other GTrXL variants. We use the "Memory Maze" environment (App. Fig.8), a memory-based navigation task in which the agent must discover the location of an apple randomly placed in a maze. The agent receives a positive reward for collecting the apple and is then teleported to a random location in the maze, with the apple's position held fixed. The agent can make use of landmarks situated around the room to return as quickly as possible to the apple for subsequent rewards. Therefore, an effective mapping of the environment results in more frequent returns to the apple and higher reward.

We chose to perform the sensitivity ablation on Memory Maze because (1) it requires the use of long-range memory to be effective and (2) it includes both continuous and discrete action sets (details in Appendix B) which makes optimization more difficult. In Figure 6, we sample 25 independent V-MPO hyperparameter settings from a wide range of values and train the networks to 2 billion environment steps (see Appendix C). Then, at various points in training (0.5B, 1.0B and 2.0B), we rank all runs by their mean return and plot this ranking. Models with curves which are both higher and flatter are thus more robust to hyperparameters and random seeds. Our results demonstrate that (1) the GTrXL (GRU) can learn this challenging memory environment in much fewer environment steps than LSTM, and (2) that GTrXL (GRU) beats the other gating variants in stability by a large margin, thereby offering a substantial reduction in necessary hyperparameter tuning. The values in Table 3 list what percentage of the 25 runs per model had losses that diverged to infinity. The only model reaching human performance in 2 billion environment steps is the GTrXL (GRU), with 10 runs having a mean score > 8.

4.3.3. Parameter Count-Controlled Comparisons

For the final gating ablation, we compare transformer variants while tracking their total parameter count to control for the increase in capacity caused by the introduction of additional parameters in the gating mechanisms. To demonstrate that the advantages of the GTrXL (GRU) are not solely due to an increase in parameter count, we halve the number of attention heads (which also effectively halves the embedding dimension due to the convention that the embedding size is the number of heads multiplied by the attention head dimension). The effect is a substantial reduction in parameter count, resulting in less parameters than even the canonical TrXL. Fig. 4 and Tab. 2 compare the different models to the "Thin" GTrXL (GRU), with Tab. 2 listing the parameter counts. We include a parameter-matched LSTM model with 12 layers and 512 hidden size. The Thin GTrXL (GRU) surpasses every other model except the GTrXL (GRU), even surpassing the next best-performing model, the GTrXL (Output), with over 10 million less parameters.

5. Related Work

Gating has been shown to be effective to address the vanishing gradient problem and thus improve the learnability of recurrent models. LSTM networks (Hochreiter & Schmidhuber, 1997; Graves, 2013) rely on an input, forget and output gate that protect the update of the cell. GRU (Chung et al., 2014; Cho et al., 2014) is another popular gated recurrent architecture that simplifies the LSTM cell, reducing the number of gates to two. Finding an optimal gating mechanism remains an active area of research, with other existing proposals (Krause et al., 2016; Kalchbrenner et al., 2015; Wu et al., 2016), as well as works trying to discover optimal gating by neural architecture search (Zoph & Le, 2017)

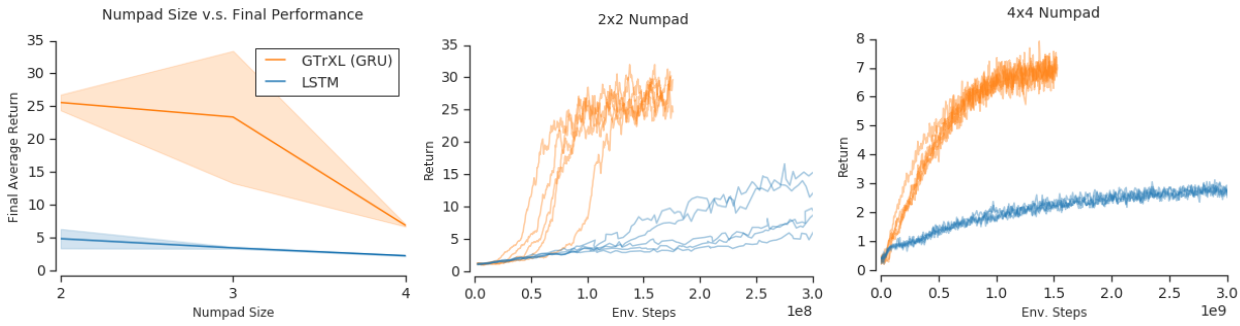

Figure 5. Numpad results demonstrating that the GTrXL has much better memory scaling properties than LSTM. **Left:** As the Numpad environment's memory requirement increases (because of larger pad size), the GTrXL suffers much less than LSTM. However, because of the combinatorial nature of Numpad, the GTrXL eventually also starts dropping in performance at 4x4. We plot mean and standard error of the last 200 episodes after training each model for 0.15B, 1.0B and 2.0B environment steps for Numpad size 2, 3 and 4, respectively. **Center, Right:** Learning curves for the GTrXL on 2×2 and 4×4 Numpad. Even when the LSTM is trained for twice as long, the GTrXL still has a substantial improvement over it. We plot 5 hyperparameter settings per model for learning curves.

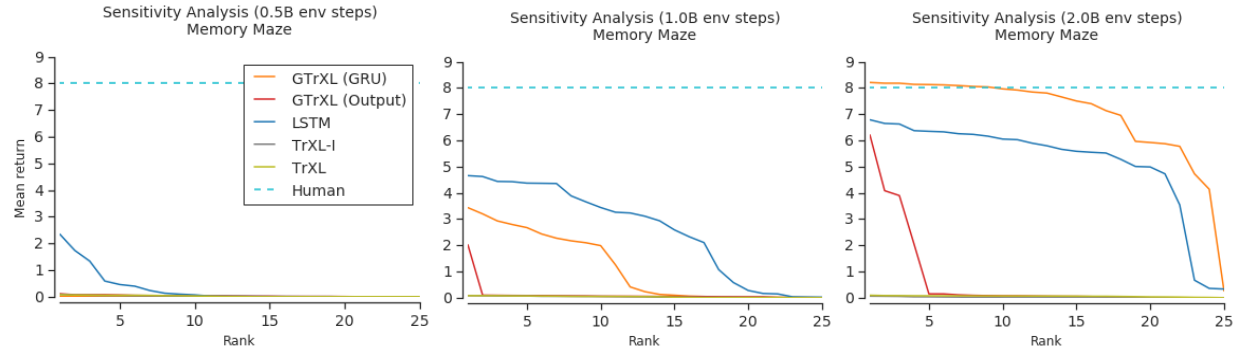

Figure 6. Sensitivity analysis of GTrXL variants versus TrXL and LSTM baselines. We sample 25 different hyperparameter sets and seeds and plot the ranked average return at 3 points during training (0.5B, 1.0B and 2.0B environment steps). Higher and flatter lines indicate more robust architectures. The same hyperparameter sampling distributions were used across models (see Appendix C). We plot human performance as a dotted line.

More generally, gating and multiplicative interactions have a long history (Rumelhart et al., 1986). Gating has been investigated previously for improving the representational power of feedforward and recurrent models (Van den Oord et al., 2016; Dauphin et al., 2017), as well as learnability (Srivastava et al., 2015; Zilly et al., 2017). Initialization has played a crucial role in making deep models trainable (LeCun et al., 1998; Glorot & Bengio, 2010; Sutskever et al., 2013).

There has been a wide variety of work looking at improving memory in reinforcement learning agents. External memory approaches typically have a regular feedforward or recurrent policy interact with a memory database through read and write operations. Priors are induced through the design of the specific read/write operations, such as those resembling a digital computer (Wayne et al., 2018; Graves et al., 2016) or an environment map (Parisotto & Salakhutdinov, 2018; Gupta et al., 2017). An alternative non-parametric perspective to memory stores an entire replay buffer of the agent's past observations, which is made available for the agent to itself reason over either through fixed rules (Blundell et al., 2016) or an attention operation (Pritzel et al., 2017). Others have looked at improving performance of LSTM agents by extending the architecture with stacked hierarchical connections / multiple temporal scales and auxiliary losses (Jaderberg et al., 2019; Stooke et al., 2019) or allowing an inner-loop update to the RNN weights (Miconi et al., 2018). Other work has examined self-attention in the context of exploiting relational structure within the input-space (Zambaldi et al., 2019) or within recurrent memories (Santoro et al., 2018).

Concurrent with this paper, several works have examined stabilizing transformers in the supervised learning setting. (Xiong et al., 2020) demonstrated that the "pre-norm" placement, which we referred to as "TrXL-I" in this work, provided stabilization benefits and avoided the otherwise necessary use of a learning rate schedule for large language models. They further provided some theoretical analysis on the gradient path when using the different layer norm

placements. A similar examination was done by (Nguyen & Salazar, 2019), which showed that the prenorm placement improved stability but sometimes at the cost of lower final performance. Finally, Working Memory Graphs (Loynd et al., 2019) applied a transformer over a recurrent memory bank and demonstrated similar improvements in RL applications over LSTM baselines.

6. Conclusion

In this paper we provided evidence that confirms previous observations in the literature that standard transformer models are unstable to train in the RL setting and often fail to learn completely (Mishra et al., 2018). We presented a new architectural variant of the transformer model, the GTrXL, which has increased performance, more stable optimization, and greater robustness to initial seed and hyperparameters than the canonical architecture. The key contributions of the GTrXL are reordered layer normalization modules and a gating layer instead of the standard residual connection. We performed extensive ablation experiments testing the robustness, ease of optimization and final performance of the gating layer variations, as well as the effect of the reordered layer normalization. These results empirically demonstrate that the GRU-type gating performs best across all metrics, exhibiting comparable robustness to hyperparameters and random seeds as an LSTM while still maintaining a performance improvement. Furthermore, the GTrXL (GRU) learns faster, more stably and achieves higher final performance (even when controlled for parameters) than the other gating variants on the challenging multitask DMLab-30 benchmark suite. Having demonstrated substantial and consistent improvement in DMLab-30, Numpad and Memory Maze over the ubiquitous LSTM architectures currently in use, the GTrXL makes the case for wider adoption of transformers in RL. A core benefit of the transformer architecture is its ability to scale to very large and deep models, and to effectively utilize this additional capacity in larger datasets. In future work, we hope to test the limits of the GTrXL's ability to scale in the RL setting by providing it with a large and varied set of training environments.

7. Acknowledgments

We thank Alexander Pritzel, Chloe Hillier, Vicky Langston and many others at DeepMind for discussions, feedback and support during the preparation of the manuscript.

References

Abdolmaleki, A., Springenberg, J. T., Degrave, J., Bohez, S., Tassa, Y., Belov, D., Heess, N., and Riedmiller, M. Relative Entropy Regularized Policy Iteration. *arXiv preprint*, 2018a.

Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Munos, R., Heess, N., and Riedmiller, M. Maximum a Posteriori Policy Optimisation. *Int. Conf. Learn. Represent.*, 2018b.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Baevski, A. and Auli, M. Adaptive input representations for neural language modeling. *Int. Conf. Learn. Represent.*, 2019.

Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Blundell, C., Uria, B., Pritzel, A., Li, Y., Ruderman, A., Leibo, J. Z., Rae, J., Wierstra, D., and Hassabis, D. Model-free episodic control. *arXiv preprint arXiv:1606.04460*, 2016.

Buchlovsky, P., Budden, D., Grewe, D., Jones, C., Aslanides, J., Besse, F., Brock, A., Clark, A., Colmenarejo, S. G., Pope, A., Viola, F., and Belov, D. TF-Replicator: Distributed Machine Learning for Researchers. *arXiv preprint*, 2019.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., and Salakhutdinov, R. 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.

Dauphin, Y. N., Fan, A., Auli, M., and Grangier, D. Language modeling with gated convolutional networks. In

Proceedings of the 34th International Conference on Machine Learning-Volume 70, pp. 933–941. JMLR. org, 2017.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, Ł. Universal transformers. *Int. Conf. Learn. Represent.*, 2018.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers 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. 4171–4186, 2019.

Edunov, S., Ott, M., Auli, M., and Grangier, D. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 489–500, 2018.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., and Kavukcuoglu, K. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, pp. 1406–1415, 2018.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS10). Society for Artificial Intelligence and Statistics*, 2010.

Google. Cloud TPU, 2018.

Graves, A. Generating sequences with recurrent neural networks, 2013.

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S. G., Grefenstette, E., Ramalho, T., Agapiou, J., et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471, 2016.

Gupta, S., Davidson, J., Levine, S., Sukthankar, R., and Malik, J. Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2616–2625, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016a.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016b.

Hessel, M., Soyer, H., Espeholt, L., Czarnecki, W., Schmitt, S., and van Hasselt, H. Multi-task Deep Reinforcement Learning with PopArt. *arXiv preprint*, 2018.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Humplik, J., Galashov, A., Hasenclever, L., Ortega, P. A., Teh, Y. W., and Heess, N. Meta reinforcement learning as task inference. *arXiv preprint*, 2019.

Jaderberg, M., Czarnecki, W. M., Dunning, I., Marris, L., Lever, G., Castaneda, A. G., Beattie, C., Rabinowitz, N. C., Morcos, A. S., Ruderman, A., et al. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019.

Kalchbrenner, N., Danihelka, I., and Graves, A. Grid long short-term memory. 07 2015.

Kapturowski, S., Ostrovski, G., Quan, J., Munos, R., and Dabney, W. Recurrent experience replay in distributed reinforcement learning. *Int. Conf. Learn. Represent.*, 2019.

Krause, B., Lu, L., Murray, I., and Renals, S. Multiplicative lstm for sequence modelling, 2016.

LeCun, Y., Bottou, L., Orr, G. B., and Müller, K.-R. Efficient backprop. In *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, pp. 9–50, London, UK, UK, 1998. Springer-Verlag. ISBN 3-540-65311-2.

Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., and Shazeer, N. Generating wikipedia by summarizing long sequences. *Int. Conf. Learn. Represent.*, 2018.

Liu, Y. and Lapata, M. Text summarization with pretrained encoders. *Conference on Empirical Methods in Natural Language Processing*, 2019.

Loynd, R., Fernandez, R., Celikyilmaz, A., Swaminathan, A., and Hausknecht, M. Working memory graphs. *arXiv preprint arXiv:1911.07141*, 2019.

Miconi, T., Stanley, K., and Clune, J. Differentiable plasticity: training plastic neural networks with backpropagation. In *International Conference on Machine Learning*, pp. 3556–3565, 2018.

Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. A Simple Neural Attentive Meta-Learner. *Int. Conf. Learn. Represent.*, 2018.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.

Neunert, M., Abdolmaleki, A., Wulfmeier, M., Lampe, T., Heess, N., Hafner, R., and Riedmiller, M. Continuous-Discrete Deep Reinforcement Learning for Hybrid Control in Robotics. *To appear in Conf. on Robot Learn. (CoRL)*, 2019.

Nguyen, T. Q. and Salazar, J. Transformers without tears: Improving the normalization of self-attention. *arXiv preprint arXiv:1910.05895*, 2019.

Parisotto, E. and Salakhutdinov, R. Neural map: Structured memory for deep reinforcement learning. *Int. Conf. Learn. Represent.*, 2018.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., dAlché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Pritzel, A., Uria, B., Srinivasan, S., Badia, A. P., Vinyals, O., Hassabis, D., Wierstra, D., and Blundell, C. Neural episodic control. In *International Conference on Machine Learning*, pp. 2827–2836, 2017.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019.

Rumelhart, D. E., Hinton, G. E., and McClelland, J. L. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter 2: A General Framework for Parallel Distributed Processing, pp. 45–76. MIT Press, Cambridge, MA, USA, 1986. ISBN 0-262-68053-X.

Santoro, A., Faulkner, R., Raposo, D., Rae, J., Chrzanowski, M., Weber, T., Wierstra, D., Vinyals, O., Pascanu, R., and Lillicrap, T. Relational recurrent neural networks. In *Advances in Neural Information Processing Systems*, pp. 7299–7310, 2018.

Song, H. F., Abdolmaleki, A., Springenberg, J. T., Clark, A., Soyer, H., Rae, J. W., Noury, S., Ahuja, A., Liu, S., Tirumala, D., Heess, N., Belov, D., Riedmiller, M., and Botvinick, M. M. V-mpo: On-policy maximum a posteriori policy optimization for discrete and continuous control. In *International Conference on Learning Representations*, 2020.

Srivastava, R. K., Greff, K., and Schmidhuber, J. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

Stooke, A., Dalibard, V., Jayakumar, S. M., Czarnecki, W. M., and Jaderberg, M. Perception-prediction-reaction agents for deep reinforcement learning. *SPIRL Workshop*, 2019.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. On the importance of initialization and momentum in deep learning. In Dasgupta, S. and McAllester, D. (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., and Kavukcuoglu, K. Conditional image generation with pixelcnn decoders. In *Advances in neural information processing systems*, pp. 4790–4798, 2016.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Vinyals, O., Babuschkin, I., Chung, J., Mathieu, M., Jaderberg, M., Czarnecki, W., Dudzik, A., Huang, A., Georgiev, P., Powell, R., Ewalds, T., Horgan, D., Kroiss, M., Danihelka, I., Agapiou, J., Oh, J., Dalibard, V., Choi, D., Sifre, L., Sulsky, Y., Vezhnevets, S., Molloy, J., Cai, T., Budden, D., Paine, T., Gulcehre, C., Wang, Z., Pfaff, T., Pohlen, T., Yogatama, D., Cohen, J., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Apps, C., Kavukcuoglu, K., Hassabis, D., and Silver, D. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II, 2019.

Wayne, G., Hung, C.-C., Amos, D., Mirza, M., Ahuja, A., Grabska-Barwinska, A., Rae, J., Mirowski, P., Leibo, J. Z., Santoro, A., Gemici, M., Reynolds, M., Harley, T., Abramson, J., Mohamed, S., Rezende, D., Saxton, D., Cain, A., Hillier, C., Silver, D., Kavukcuoglu, K., Botvinick, M., Hassabis, D., and Lillicrap, T. Unsupervised predictive memory in a goal-directed agent. *arXiv preprint arXiv:1803.10760*, 2018.

Wu, Y., Zhang, S., Zhang, Y., Bengio, Y., and Salakhutdinov, R. R. On multiplicative integration with recurrent neural networks. In *Advances in neural information processing systems*, pp. 2856–2864, 2016.

Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., and Liu, T.-Y. On layer normalization in the transformer architecture. *arXiv preprint arXiv:2002.04745*, 2020.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., and Le, Q. V. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in Neural Information Processing Systems*, 2019.

Zambaldi, V., Raposo, D., Santoro, A., Bapst, V., Li, Y., Babuschkin, I., Tuyls, K., Reichert, D., Lillicrap, T., Lockhart, E., Shanahan, M., Langston, V., Pascanu, R., Botvinick, M., Vinyals, O., and Battaglia, P. Deep reinforcement learning with relational inductive biases. In *International Conference on Learning Representations*, 2019.

Zilly, J. G., Srivastava, R. K., Koutník, J., and Schmidhuber, J. Recurrent highway networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 4189–4198. JMLR. org, 2017.

Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. In *ICLR*, 2017.