A. Appendix

A.1. Additional Method Details

Position Embedding Relative position embeddings (Shaw et al., 2018) make it possible to condition on the order of inputs by modifying the attention to $a_{ti} = \text{Softmax}(\mathbf{q}_t^\top \mathbf{k}_i + \mathbf{q}_t^\top \mathbf{p}_{t-i})$. However, because this second term is computed for the whole block in parallel for efficiency, it can become expensive for a large L even when the average memory size $|C_t|$ is small. Our solution is to remove position embeddings from older memories $i < t - K$ (where K is the block size), which empirically does not affect performance. The computational complexity of the position embeddings is then $\mathcal{O}(K)$, thus allowing us to increase the maximum span L. This modification makes training EXPIRE-SPAN more efficient, but does not improve accuracy.

Training with Small Initial Spans EXPIRE-SPAN scales to long attention spans as it quickly learns to expire irrelevant content. However, at the beginning of training, the long span can use large quantities of GPU memory. To circumvent this, we initialize the bias term b with a negative value. This prevents large memory usage at the beginning of training, after which the model quickly learns to expire and the memory usage is no longer problematic.

A.2. Additional Experimental Results

Efficiency for Instruction Task We include a comparison of EXPIRE-SPAN to Adaptive-Span and Compressive Transformer in Table 5 and show that EXPIRE-SPAN has stronger performance, is faster, and saves GPU memory.

Wikitext-103 Language Modeling The Wikitext-103 word-level language modeling benchmark (Merity et al., 2016) consists of a collection of Wikipedia articles and a fixed vocabulary size of 270K. We set the max attention span for EXPIRE-SPAN to 8K. We compare EXPIRE-SPAN to existing work in Table 6 and show that even fairly small models trained with EXPIRE-SPAN achieve competitive results. Next, we analyze the performance of EXPIRE-SPAN on Wikitext-103 as the memory size increases. We compare to a Transformer-XL model in Figure 10 — even with far smaller memory, EXPIRE-SPAN performs much better.

Expire-span Performance and Analysis on Enwik8 In Figure 11, we analyze multiple layers of a trained model and show that different layers memorize different types of information. Several layers retain summarizing information about sentences or sections by increasing the expire-spans of spaces, new lines, and section titles.

Additionally, we did an ablation by running our large Expire-Span model without LayerDrop. Its validation performance dropped from 0.98bpb to 1.00bpb.

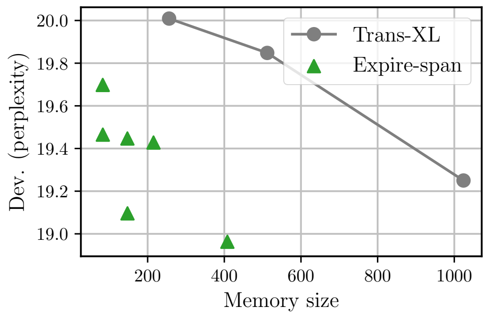

Figure 10. **Performance as a function of Memory Size on Wikitext-103**

Importance of Structured Dropout for Regularization We analyze the importance of structured dropout to regularize the large memory capacity provided by EXPIRE-SPAN. In an experiment on enwiki8, Figure 12 shows that loss on a portion of validation data was incredibly large. This part corresponds to a 66K token long table. We hypothesize that the model likely never encountered such a table during training. During validation, this caused all non-table tokens to expire. Without regularizing the model memory size during training, the model can easily overfit.

Colliding Objects, An Easy Version We experiment with an easier version of the Colliding Objects task where objects do not have colors. The model has to predict either the last collision, or a mapping of the last 3 collisions. In contrast to the harder task, there are no color switches and any collision prediction is valid. As this version is less memory intensive, the EXPIRE-SPAN model almost solves it with a shorter maximum span, as shown in Table 7.

A.3. Additional Implementation Details

A.3.1. REINFORCEMENT LEARNING TASKS

We used MazeBase (Sukhbaatar et al., 2015a) to construct tasks in grid world. Agents can observe its surrounding 3×3 area and move in the four cardinal directions. Every objects and their properties are described by words such as "agent","block", "blue", etc. Thus, the input to the model is a binary tensor of size $3 \times 3 \times$ vocabulary-size.

We train 2-layer Transformers with 64 hidden units using actor-Critic algorithm. We used a BPTT length of 100, and an entropy cost of 0.0005.

Corridor Task The corridor length is sampled from $\mathcal{U}(3, 200)$. All models are trained for 100M steps. We used RMSProp optimizer with a learning rate of 0.0001 and a batch size of 64. For the expire-span models, we set the

	Model	Performance	GPU Memory (GB)	Time/Batch (ms)
Instruction Task	Compressive Transformer	71% Acc	10	210
	Adaptive-Span	64% Acc	14	240
	EXPIRE-SPAN	**74%** Acc	**8**	**90**

Table 5. **Efficiency of EXPIRE-SPAN**. We report peak GPU memory usage and per-batch training time, fixing the batch size. We evaluate the mean pooling version of the Compressive Transformer.

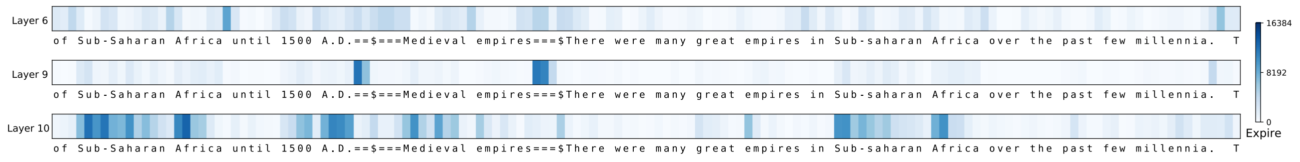

Figure 11. **Per-Layer EXPIRE-SPAN values on Enwik8**. We visualize the expire-spans of different layers: layer 6 gives long span to spaces, layer 9 memorizes special tokens like newlines and section titles, and layer 10 retains named entities in memory.

Model	Params	Test
DEQ-Trans. (Bai et al., 2019)	110M	23.3
Trans-XL (Dai et al., 2019)	257M	18.3
Feedback Trans. (Fan et al., 2020b)	77M	18.3
Trans.+LayerDrop (Fan et al., 2020a)	423M	17.7
Compressive Trans. (Rae et al., 2020)	277M	17.1
Routing Trans. (Roy et al., 2020)	-	15.8
EXPIRE-SPAN	140M	19.6

Table 6. **Wikitext-103 Results.** We report perplexity on test.

Model	Maximum Span	Test Error (%)
Transformer-XL	1k	39.1
EXPIRE-SPAN	1k	19.5
	2k	9.1
	4k	3.2

Table 7. **Colliding Objects Results**. We report test error.

maximum span L to 200, the loss coefficient α to 5e-6, and the ramp length R to 16.

Multi-Room Portal In this task, there are 50 rooms sequentially connected together. Each room is 5×5 in size, and have two doors with different colors. If agent go to the correct door, it will be teleported to the next room, but if it is the wrong door, the agent will be teleported back to the first room and have to start over. Which of the two doors is correct in each room is randomly decided and fixed throughout the episode. This information is not visible to the agent, thus can only be discovered by trial and error within each episode. The current room number is visible to the agent.

When the agent successfully transitions from the k-th room to the next, it receives a reward of $0.1k$. The episode ends if the agent makes two mistakes in the same room, reaches the last room, or when the number of steps reach 1000. A reward discount of 0.98 is used. All models are trained with Adam optimizer with a learning rate of 5e-4, and a batch size of 1024, with gradients are clipped at 0.1. We set $L = 100$, $R = 16$ and $\alpha =$1e-6 for the expire-span models.

A.3.2. INSTRUCTION TASK IN LIGHT

We train 6-layer models with a hidden size of 512 and 8 attention heads. To train, we use the Adam optimizer with a learning rate of 7e-4 and 8000 warmup updates. We set the expire-span ramp R to 64 and the expire-span loss α to 2e-6.

A.3.3. COLLISION TASK

At the start of the simulation, each particle samples a Gaussian Normal velocity and position uniform inside a 16×16 grid. At each time step the particles' position is updated by adding its velocity (unless it would go off the grid, in which case its velocity is re-sampled). There are 5 different colors, and a particle can change its color randomly at each step with 0.05 probability. A collision happens when the two particles have the same rasterized locations, but it does not affect the movement.

Given a question specifying two colors, the task is to report in which of the four quadrants of the grid the last collision of the specified-colors occurred. To make the task easier to learn, 40% of the queries will have the matching colors as the last collision.

The model is given an input sequence of tokens that has 8 entries per timestep. The first 4 are the rounded and rasterized (x, y) locations of the two particles, and next 2

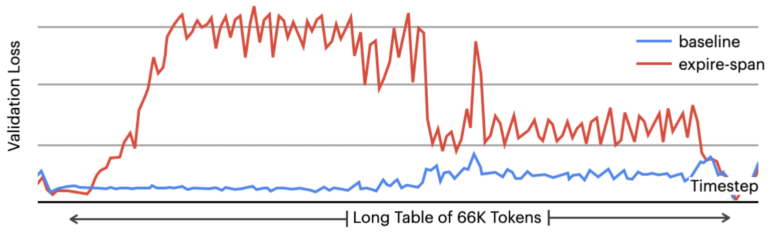

Figure 12. **Extreme Overfitting** on part of validation occurs without proper regularization.

are tokens representing the colors of the particles. The last 2 entries are "question" tokens that specify the colors of the collision. The output sequence has a token for each quadrant. We generate 50M steps for training, which equals to 400M entries.

Easy Version: The particles have no color in this version. There are two types of questions, in which the task is to report either: **(i)** in which of the four quadrants of the grid the last collision occurred, or **(ii)** the label mapping of the last 3 collisions.

A.3.4. LANGUAGE MODELING DETAILS

Enwik8 Our small model has 12 layers with a hidden size of 512 and 8 attention heads. To train, we use Adam optimizer with a learning rate of 7e-4, a batch size of 512, a block size of 512 and 8000 warmup updates. All models are trained for 100k updates. The model in Table 2 is further fine-tuned for another 10k updates with a 10x smaller LR. The baseline models used for comparison are the same size model following the same training protocol.

The large EXPIRE-SPAN model Table 2 is a 24-layer model with a hidden size of 768 and 4096 feedforward units. It is trained with a learning rate of 4e-4 and 16k warmup steps. In addition to 0.5 dropout, we also add 0.2 layer-drop. The EXPIRE-SPAN parameters are $L = 32k$, α =3e-7, and $R = 128$. We used the version of Eq. 6 due to the very long maximum span.

Character-level PG-19 Besides the maximum span, all model parameters and training parameters were held constant. Each model had 12 layers, a hidden size of 512, a feedforward size of 2048, 8 attention heads, and processed a block of 512 characters at a time. We initialized the weights using a uniform distribution as described by (Glorot & Bengio, 2010), used dropout of 0.2, clipped the gradients at 0.3, warmed up the learning rate linearly for 8000 steps, and used cosine annealing to decay the learning rate after warmup (Loshchilov & Hutter, 2016). For the EXPIRE-SPAN models, we used a ramp of $R = 128$ and an expiration loss coefficient of α =1e-6 (3e-7) for $L = 8k$ ($16k$).

Wikitext-103 All models have 8 layers and 1024 hidden units (4096 in feedforward layers). In addition to the dropout of 0.3 applied to attention and ReLU activation, outputs from the embedding layer and the last layer had a dropout of 0.2. We used the adaptive input (Baevski & Auli, 2019) and the adaptive softmax (Grave et al., 2017) for reducing the number of parameters within word embeddings. The models are trained for 300k updates with a block size of 256, and gradients are clipped at 0.1. The other hyperparameters are the same as the small Enwik8 experiments.