All Context Aware Reservoir Transformer Possible

Anonymous EMNLP submission

Abstract

The commitment of language processing is largely restricted by knowing the context around it. However, Transformer, as one of the most powerful neural network architectures, has its input length restricted due to a quadratic time and memory complexity. Despite rich work advancing its efficiency, long context is still an issue that requires large computational resources in training. We realize a novel reservoir Transformer that bounds the learning in linear time by handling different input lengths in a cascaded way. For a long-term context, the reservoir with non-linear readout learns sample dependencies from the beginning to the end of a sequential dataset; To learn more accurately the medium-term context such as previous sentences, we apply a recurrent memory mechanism; and finally for the short-term dependencies in one sentence, we learn with the Transformer. Experiments show that our reservoir Transformer improves BERT and Blenderbot performance and significantly increases our prediction accuracy in (small) language modeling, text classification, and chatbot tasks over the state-of-the-art methods. This shows that a reservoir Transformer makes it possible to efficiently learn from extremely long context.

1 Introduction

Transformer has updated state-of-the-art in a wide range of AI tasks including but not limited to NLP, computer vision, bioinformatics, etc. (Vaswani et al., 2017; Devlin et al., 2018; Dosovitskiy et al., 2020). One important limitation of Transformer is the quadratic time and memory complexity of the input length, e.g., BERT has a restriction of 512 input tokens, and GPT-3 2048 for efficiency. Even LlaMA 3 (Meta, 2024), Gemma (Team et al., 2024), GPT-4 (Achiam et al., 2023), and Mistral (Jiang et al., 2023) have maximum input tokens as $8K$. However, long sequential inputs can be extremely useful for learning contextual information. For example, in language understanding, words have different meanings in different contexts; In dialogue modeling, the lack of effective contextual understanding can lead to incoherent or irrelevant responses in longer conversations. Therefore, the Transformer's length restriction must be solved so that long histories can be retained and utilized.

Several studies have investigated how to increase Transformer input lengths, such as (Kitaev et al., 2020; Kim and Cho, 2020; Beltagy et al., 2020; Choromanski et al., 2020; Katharopoulos et al., 2020; Zhou et al., 2021; Guo et al., 2021; Ma et al., 2021; Hua et al., 2022; Tay et al., 2022; Bertsch et al., 2023; Liu et al., 2023; Li et al., 2023; Mohtashami and Jaggi, 2023; Ainslie et al., 2023; Bulatov et al., 2023; Liu and Abbeel, 2024; Munkhdalai et al., 2024; Tworkowski et al., 2024; Bertsch et al., 2024; Martins et al., 2021; Han et al., 2023; Mohtashami and Jaggi, 2024). Existing solutions, however, either modifying the attention model with heuristic assumptions or projecting long input into a fixed dimension. Since most work does not consider temporal patterns of the input, there is still room to reduce the information loss learned from the long context and improve the prediction accuracy and efficiency.

In this work, we introduce a novel approach that enhances the Transformer with reservoir computing to efficiently handle long sequences. Reservoir Computing (RC) is a class of simple and efficient Recurrent Neural Networks where internal weights are fixed at random, and only a linear output layer is trained. Reservoir computing requires a small number of training data samples and computing resources with great advantages for processing sequential data in linear time and constant space (Gauthier et al., 2021). Here, we improve reservoir with non-linear readout to take long conversational context into account for time and memory efficiency (Gauthier et al., 2021).

Figure 1 shows the architecture of our Reservoir Transformer. We handle the input sequence

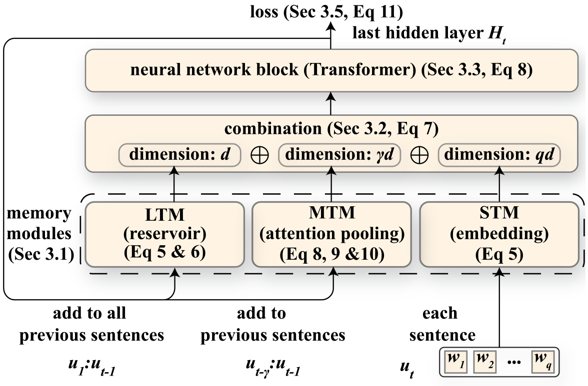

Figure 1: A schematic representation of the integrated memory system combining Reservoir, LTM, MTM, and STM with the Neural Network Block. The Long-Term Memory module (reservoir) processes all the previous states $u_1 : u_{t-1}$. The Medium-Term Memory module processes only the immediately preceding states $u_{t-\gamma} : u_{t-1}$. The Short-Term Memory module (Embedding) processes the current state u_t with the window size k and tokens $w_1, \cdots, w_k$.

at three cascaded processes depending on the input context length: For long context, e.g., the total training data like the full article, our reservoir reads sentence by sentence sequentially from the entire training data; for intermediate long context such as the previous five sentences, we apply a recurrent neural network (RNN) for a more accurate learning; for short context, i.e., the current sentence, we will maintain learning the token dependency using a Transformer.

Figure 1 shows our system architecture. The main novelty of our model is the three memory modules, i.e. short-term (STM), medium-term (MTM), and long-term (LTM) to represent different context lengths. For time step t, the current sentence u_t is fed only to the STM which embeds it to get the sentence's embedding matrix $e(u_t)$. The MTM takes γ previous sentences $u_{t-\gamma}, \cdots, u_{t-1}$ to an Attention Pooling. The LTM is a reservoir that processes the Transformer encoder's final layer output H_i for all the previous sentences in the whole dataset $u_1, \cdots, u_{t-1}$. The output of these three modules is concatenated together and then fed to the Neural Network Block model, such as a BERT, Blenderbot, and BART.

Our reservoir method makes it possible for the Transformer to process an infinite number of input tokens. Our contributions include the following: (a) We introduce reservoir computing to handle arbitrary long input of Transformer; (b) We enhance the conventional reservoir computing model by replacing linear with a non-linear readout for dimension reduction and better feature learning; (c) We introduce integrating the reservoir, RNN, and Transformer to handle long, intermediate, and short contexts, respectively; (d) We collect experimental evidence that our reservoir Transformer significantly enhances the performance and generalizes the model learning robustness of the Transformer on various NLP tasks showcasing an improvement of 2.7 in perplexity score compared to the Transformer-XL (Dai et al., 2019) Large model and 2.4% points in accuracy compared to LONGFORMER (Beltagy et al., 2020), for the language modelling and the text classification tasks respectively.

2 Problem Definition

Given a sequence of words $w_1^K = w_1 w_2 \cdots w_k \cdots w_K$ ($k \in 1, 2, \cdots, K$), where K is the input length, and its segmentation boundaries k_1^J, a corpus can be represented in the form of a sequence of sentences $u_1^J = u_1 u_2 \cdots u_j \cdots u_J$ ($j \in 1, 2, \cdots, J$), and each individual sentence u_j is defined as $u_j = w_{k_{j-1}+1} \cdots w_{k_j} = w_{k_{j-1}}^{k_j}$, where u_1^J is composed of two sources of information, the word sequence and its sentences.

Taking language model as an example, we predict the masked token $\bar{w}_i$ ($k_{j-1} < i < k_j + 1$) within a sentence j in the context of discriminative language modeling, and for generative language modeling, we predict the next token considering the last token is the masked token, i.e., $\bar{w}_i \equiv w_{k_j}$. we aim to predict masked tokens $\hat{w}_i$ ($k_{j-1} < i < k_j + 1$) within a sentence j, as expressed by:

$$Pr(w_i | u_1, u_2, \cdots, w_{k_{j-1}+1} \cdots \bar{w}_i \cdots w_{k_j}) \quad (1)$$
$$= \text{softmax}(y_{ji}), \quad (2)$$

where t is the reservoir state time step, and is also the sentence index, i.e. $t \equiv j$ and $y_{ji} \equiv y_{ti}$ in Equation 10. Besides language modeling, we apply the similar analogy to text classification, dialogue modeling, and text summarization.

Complexity: Conventional Transformers to encapsulate dependencies across these long sequences resulting in the time complexity of $O(K^2 \times d)$, with d as the model dimension. As the sequence length K is extremely long, therefore quadratic time complexity becomes impractical. To counter this problem, we propose a novel framework for Reservoir Transformer (RT). This consists of the memory module which deal with handling three different lengths of context. The short-term memory module (STM) handles only the current sentence u_t. The medium-term memory module (MTM) handles γ previous sentences $u_{t-\gamma}, \cdots, u_{t-1}$. The long-term memory module (LTM) handles all the previous sentences $u_1, \cdots, u_{t-1}$. The time complexity of LTM is n^2 where n is the number of neurons in the reservoir. The time complexity for MTM is $\gamma \times qd$ and the time complexity of STM is $q^2 d$, where q is the sentence length. Therefore both LTM and MTM are linear in terms of the input length and even though the total complexity is still quadratic, however in practice as we set q to a small value of 512 while the max value of K we experiment with is $29K$ but can potentially be even higher.

3 Reservoir Transformer

We propose a novel architecture we call Reservoir Transformer (RT). It combines the idea from reservoir computing (Gallicchio et al., 2017) with the Transformer architecture. The novelty of our approach is that we propose three different memory mechanisms to capture information at different levels. The long-term, medium-term, and short-term memory modules focus on different lengths of context to balance the precision and the efficiency. We also discuss the non-linear readout that significantly improves the linear readout of the RC.

3.1 Memory Modules

We propose a novel idea of using three memory modules as part of a context-aware memory framework that not only addresses the challenge of handling long context but also efficiently controls which previous input sentence should be given more importance especially when the previous context length becomes very large. These three memory modules help to capture dependency at different levels. The long-term memory module handles the whole context and therefore can capture long dependency present in all the previous input sentences. The medium-term memory module captures dependency from immediately preceding input sentences. The short-term memory module captures the local dependencies existing within a single sentence.

We introduce three modules to handle different ranges of input lengths to balance the efficiency and accuracy. These three mechanisms are (i) Short-Term Memory (STM), (ii) Medium-Term Memory (MTM), and (iii) Long-Term Memory (LTM).

Short-Term Memory (STM) - Transformer: Unlike LTM which handles the long-term context, the STM only inputs the t'th input sentence u_t. Each of the q tokens $w_1, \cdots, w_q$ are fed to the embedding layer as input and we get the sentence embedding $e(w_1), \cdots, e(w_q)$ as the output.

Here $e(w_i)$ is the embedding layer output for the token w_i. As Short-Term Memory does not learn any relation between sentences, this step makes the training much more efficient when dealing with shorter inputs. We use Transformer with the self-attention mechanism, which computes the attention scores with the input itself.

Medium-Term Memory (MTM): LTM allows the handling long context as input. However, the modeling is not precise enough. Therefore, we add the medium-term memory module to handle the medium-term context so that far-away samples will be taken care of by the reservoir and close-by previous samples will be learned by the Medium-Term Memory.

The MTM focuses on capturing and processing only the immediately preceding states for the current context. For a given current state t, MTM considers the γ immediate preceding hidden states, represented as $H_{t-\gamma}, \ldots, H_{t-2}, H_{t-1}$. While the output of the Reservoir Transformer for each state input is a $q \times d$ dimensional vector, the MTM requires scalar inputs. Conventionally, max pooling is used to convert these vector outputs into scalar forms; however, this method potentially omits valuable multidimensional data from the Transformer's output. To address this, we adopt attention pooling, as proposed by Alam et al. (2023), which offers improved performance by preserving more information.

The objective of the attention pooling mechanism is to construct a condensed representation, $\beta_{t-\gamma}, \ldots, \beta_{t-2}, \beta_{t-1} \in \mathbb{R}^{\gamma \times d}$, from the inputs $H_{t-\gamma}, \ldots, H_{t-2}, H_{t-1}$ and the output is used for the MTM. We achieve this by emphasizing the most significant frames in the context of the sequence. Specifically, for the state H_{t-1}, the attention pooling is defined as:

$$\beta_{t-1} = \sum_{i=1}^{q} \alpha_i H_{t-1}^i, \tag{3}$$

Here each $\alpha_i \in [0, 1]$ represents the normalized attention weight allocated to the frame H_{t-1}^i. These weights are computed through a softmax function, ensuring they sum to 1, as shown in Equation 4. The intrinsic non-linearity of the softmax function within the attention mechanism ensures the model's capacity to capture complex, hierarchical dependencies.

$$\alpha_i = \frac{\exp(e_i)}{\sum_{i=1}^{d} \exp(e_i)}, \tag{4}$$

Here, the score e_i is derived from H_{t-1}^i using the learnable parameters v_i, W_i, and b_i. This transformation, followed by the application of a hyperbolic tangent function, is expressed as:

$$e_i = v_i \tanh(W_i \cdot H_{t-1}^i + b_i), \tag{5}$$

These parameters (v_i, W_i, b_i) are fine-tuned during the training phase, allowing the attention pooling to dynamically assign optimal weights to each frame, tailored to the specific task.

Long-Term Memory (LTM) using Reservoir Computing (RC): Reservoir Computing (RC) is a framework within Recurrent Neural Networks (RNN) that capitalizes on the high-dimensional non-linear dynamics of neurons to process sequences. We use the LTM module to model all input sentences to a fixed reservoir memory. The LTM module treats each input sentence as a unique state, enabling it to incorporate historical information efficiently without increasing the input sequence length. This helps not only in processing lengthy sequences but also in reducing the RC output dimension for better performance.

The reservoir network (Gallicchio et al., 2017) in Equation 6 processes all previous context.

$$x_t = (1-\kappa)x_{t-1} + \kappa \tanh(W_r x_{t-1} + W_i H_{t-1}) \tag{6}$$

Here H_{t-1} is output of the Transformer's last layer for the $t-1$'th sentence, x_{t-1} is the previous reservoir network state, $\kappa \in [0, 1]$ is the leaky parameter, and $W_r \in \mathbb{R}^{n\times n}$ and $W_i \in \mathbb{R}^{n\times s}$ represent the reservoir and input weight matrices, respectively. These matrices are fixed and generated randomly, with each weight being drawn from an i.i.d. Gaussian distribution with variances ${\sigma_r}^2$ and ${\sigma_i}^2$, respectively.

Instead of a linear readout of $o_t = W_o x_t$, we propose a non-linear readout given in Equation 7 for enhanced prediction capabilities.

$$o_t = \sigma(W_o x_t) \tag{7}$$

Here, σ is a non-linear activation. We use ReLU as default non-linear activation but we compare with other activation functions in Section 4.2, and $W_o \in \mathbb{R}^{r\times n}$ denotes the output weights, where r represents the output dimension.

The output of non-linear readout is then passed to the self-attention mechanism.

$$Q = W^Q o_t, \quad K = W^K o_t, \quad V = W^V o_t$$
$$o'_t = \frac{\text{softmax}(QK^T)V}{\sqrt{d_k}} \tag{8}$$

Here W^Q, W^K, W^V are learnable weight matrices. d_k is the dimensions of the key.

3.2 Combining Memory Modules

The outputs of each of the memory modules are given as input to a concatenation layer. We add a small trainable weight parameter to the concatenation layer for each module. Equation 9 shows the concatenation layer. $\oplus$ is the concatenation operator and the coefficients $\mu_1, \mu_2, \mu_3 \in [0, 1]$ act as controlling parameters.

$$z_t = \mu_1 \cdot o'_t \oplus \mu_2 \cdot (\beta_{t-\gamma} \oplus \cdots \oplus \beta_{t-2} \oplus \beta_{t-1}) \oplus \mu_3 \cdot (e(w_1) \oplus e(w_2) \oplus \cdots \oplus e(w_{tK})) \tag{9}$$

Here, o'_t is from Equation 8 in LTM, $\beta_{t-\gamma}, \cdots, \beta_{t-2}, \beta_{t-1}$ are from Equation 3 in MTM, and $e(w_1), e(w_2), \cdots, e(w_{tK})$ are the embedding output from STM. $\oplus$ is the concatenation operator and the coefficients $\mu_1, \mu_2, \mu_3 \in [0, 1]$ act as controlling parameters. These parameters determine the relative influence of the LTM, MTM, and STM on the present state. By adjusting these parameters, the model can learn the balance between relying on long-term, medium-term, or short-term context.

3.3 Neural Network Block

A neural network block can be any neural network architecture for classification or generation. Here, we use BERT, Blenderbot, and BART, respectively for different tasks. The concatenated memory representation in Equation 9 is then fed into the Neural Network Block to perform the prediction task. The output of the concatenation layer is fed to a Neural Network block:

$$y_{ti} = \mathbb{M}(z_t; w_{ti}) \tag{10}$$

Here, given z_t is the output from Equation 9 and $\mathbb{M}$ is the neural model. For this neural model, we experiment with BERT (Devlin et al., 2018) as the Transformer encoder-only model, and Blenderbot (Xu et al., 2021a) and BART (Lewis et al., 2019) as the Transformer encoder-decoder models.

Figure 1 shows the model architecture when using the vanilla Transformer (Vaswani et al., 2017), however, our system is agnostic of the type of the neural network used. Therefore, in theory, we can replace it with any other neural network. In practice, we experimented with two different transformer-based models (see Section 4). For encoder-decoder architectures, e.g. BART (Lewis et al., 2019), we use the same architecture with both encoder and decoder. Then, we extract the final layer's hidden states and give it to the attention-pooling layer.

3.4 Training and Parallelize STM

Training Loss In this work, given y^* as true labels, we use the cross entropy loss function as our objective:

$$\mathcal{L}(y^*, y) = \frac{1}{T} \sum_{t=1}^{T} y_t^* \log(y_t) \tag{11}$$

Training models with integrated reservoir needs to process the whole dataset sequentially to learn the inherent memory dependency between samples. Traditional batch training is impractical as each sample's computation is contingent on its predecessor. Therefore, we introduce a batch training to accelerate the training process.

As in Figure 1, a sentence is processed by a Transformer to learn within sentence dependency, then this embedded sentence is further fed into RNN and reservoir. In reservoir, each new incoming embedded sentence is fed into the model and added to the old memory based on all previous embedded sentences. Thus reservoir learns the full contextual dependency in the whole dataset. This means that reservoir needs to wait for the Transformer to process each sentence which is very time consuming. To accelerate this process, we parallelize the training process of the Transformer, where we embed S sentences at the same time, and then fed them together to the reservoir. In this way, the training time of waiting is reduced by S.

For example, for a sequence of sentence input $w_1, w_2, \cdots, w_T$. We feed w_1 to the first STM, then w_2 to the second STM, after that w_3 to the third STM, until w_S to the S-th STM. These STM outputs are collected and fed together to the reservoir. Afterwards, We feed w_{S+1} to the first STM, w_{S+2} to the second STM, until w_{2S} to the S-th STM, then fed their output to the reservoir as the second batch of the input. This is done interatively until all sentences are read, where STM processing time is reduced by S due to the parallelization.

4 Experiments and Results

In this section, we discuss the experiments we carried out to evaluate our proposed Reservoir Transformer as well as the results we obtained. Specifically, we want to verify that RT can handle context of any length, so we experiment with three NLP tasks, (i) language modeling, (ii) dialogue modeling, and (iii) text classification. The maximum length for each of the dataset varies from 1.4K for the language modeling task to more than 29K for the text summarization task. This allows us to test each of the memory module mentioned in Section 3. The language modeling task will test how our model handles long-term context, dialogue modeling will verify the medium-term context and text classification will test for the short-term context. All the training details, including the hyperparameters are discussed in the Appendix A.1.

4.1 NLP Tasks

We experimented with four NLP tasks, (1) language modeling, (2) dialogue modeling, (3) text classification, and (4) text summarization.

4.1.1 Language Modeling

Data and Pre-processing: We conduct language modeling experiments on the WikiText-103 dataset (Merity et al., 2018), which consists of 103 million words extracted from English Wikipedia articles, to assess the performance of various language models. We convert the data into sequential batches (see Section 3.4) and then use BERT tokenizer for the pre-processing.

Model Training: We adopt the parameter settings described in the original BERT paper (Devlin et al., 2018). For the reservoir (LTM), we use 3000 units with a spectral radius of 0.50. The leaky rate is set to 0.35. The reservoir readout size is set to 50, and the medium-term memory (MTM) is 50.

Results: Table 1 provides an overview of the models utilized in the experiments, along with their

Model Name	PPL
Transformer-N (Sun and Iyyer, 2021)	25.2
Transformer-XL Standard (Dai et al., 2019)	24.0
Feedback Transformer (Fan et al., 2020)	22.4
BERT-Large-CAS (Wang et al., 2019a)	20.4
Transformer-XL Large (Dai et al., 2019)	18.3
Feedback Transformer (Fan et al., 2020)	18.2
Shortformer (Press et al., 2020)	18.2
Sandwich Transformer (Press et al., 2019)	18.0
SegaTransformer-XL (Bai et al., 2021)	17.1
Compressive Transformer (Rae et al., 2019)	17.1
Hybrid H3 (Dao et al., 2022)	16.9
kNN-LM (Khandelwal et al., 2019)	15.8
Routing Transformer (Roy et al., 2021)	15.8
Reservoir Transformer (RT)	**15.6**

Table 1: Language modeling on WikiText-103.

Model	METERO	GBLEU	BLEU	ROUGE1/2/L/Lsum
BB	24.9	10.1	7.0	27.5/10.2/23.0/23.1
BB+RT	**25.2**	**10.3**	**7.2**	**28.0/10.5/23.3/23.3**

Table 2: BB+RT outperforms baseline Blenderbot (BB). GBLEU is Google's BLEU score.

Model/Dataset	HND	20N	E57K
BERT	92.0	84.8	73.1
BERT+TextRank (Park et al., 2022)	91.2	85.0	72.9
BERT+Random (Park et al., 2022)	89.2	84.6	73.2
ToBERT (Pappagari et al., 2019)	89.5	85.5	67.6
CogLTX (Ding et al., 2020a)	94.8	84.6	70.1
LONGFORMER (Beltagy et al., 2020)	94.8	83.4	54.5
BIG BIRD (Zaheer et al., 2020)	92.2	-	-
Reservoir Transformer	**97.2**	**89.7**	**74.0**

Table 3: Text classification on three datasets.

corresponding perplexity scores. Using our proposed Reservoir Transformer, which merges as the highest-performing model, achieving a deduction of 9.6 perplexity score, i.e., 38.0% relatively compared to the Transformer-N model.

4.1.2 Dialogue Modeling

Data and Pre-processing: We verify our model's performance for medium-length context on a dialogue modeling task. We create a custom conversational data by prompting GPT 3.5. The dataset containing total 400K conversation covering a wide range of topics including sports, history, travel, art, music, health, and wellness etc. We randomly select 30 conversations for test.[1]

Model Training: We use Blenderbot (Xu et al., 2021a) as the baseline model and compare it with RC and Blenderbot+RC, where we follow Roller et al. (2020) for the fine-tuning. For the reservoir, we use 1000 units with a spectral radius of 0.50. The leaky rate is set to 0.35. We set the readout size to 50 and for MTM we set the memroy size to 15. Each fine-tuning is run for 3 epochs and the final evaluation is carried out using BLEU score and ROUGE score.

Results: Experiments show that our method outperforms the baseline method in conversational modeling. We observe a consistent improvement of both our methods above the baseline model. Blenderbot+RT shows the highest improvement with +0.3 BLEU score and +0.5 ROUGE1 score higher than the Blenderbot. This comparative analysis allows us to evaluate the improvements attained by our chatbot in terms of its ability to generate coherent and contextually appropriate responses.

[1]We will release the test data along with the paper.

4.1.3 Text Classification

Data and pre-processing: For the text classification task, we present the results on three dataset; hyperpartisan news detection (HND), 20Newsgroups (20N), and EURLEX-57K (E57K). Each dataset is split into training, validation, and test following the approach outlined in Park et al. (2022).

Model Training: We use BERT for the neural network block. We set the model dropout to 0.1, attention dropout to 0.1, and weight decay to 0.05. Additionally, we set the reservoir size to 500 with a spectral radius of 0.7. The leaky rate is set to 0.35. The readout size is set to 20 and the MTM memory size is set to 5. We compare our method with various transformer-based models and also compare with LLMs including LONGFORMER and BIG BIRD.

Results: Table 3 shows the performance of RT compared to other state-of-the-art models. We can see that our system consistently achieves a higher performance for all the three tasks. For HND, we get an improvement of at least 0.6% points above the ERNIE-DOC-LARGE model. Similarly, we get an improvement of 1.2% points above SGC model for 20N task and 0.8% improvement above BERT+Random model for E57K task.

4.1.4 Text Summarization

Data and Pre-processing: To verify our model on very long context, we also experiment with the text summarization task. We used the XSum dataset (Narayan et al., 2018) for training and testing our approach. The maximum context length for data is up to 29K tokens.

Model Training: We use the BART (Lewis et al., 2019) model as the Neural Network Block for the

Model	ROUGE-1
BART (Lewis et al., 2019)	42.43
BERTSumExtAbs (Liu and Lapata, 2019)	16.30
EXT-ORACLE (Narayan et al., 2018)	29.79
Reservoir Transformer	**44.54**

Table 4: Text Summarization results for XSum dataset

Memory Modules	HND	20N
STM	92.0	84.8
MTM+STM	96.3	89.0
LTM+STM	95.5	88.3
LTM+MTM+STM	**97.2**	**89.7**

Table 5: System performance by removing specific memory modules from the network.

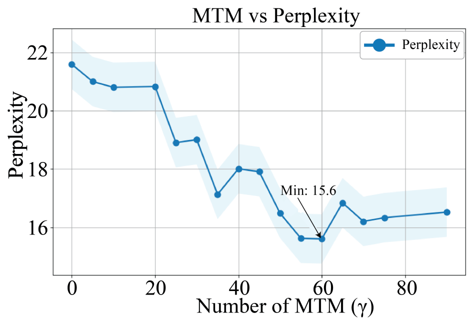

Figure 2: Perplexity score versus γ for MTM.

Reservoir Transformer. We apply the Figure 1 architecture separately for the encoder and decoder part of the BART model.

Results: Table 4 shows the results for the text summarization.The RT model gets a higher performance of +2.11% points above only BART model.

4.2 Ablation Study

Comparison of Memory Modules: For the ablation study, we compare how each of the three memory modules influences the model's performance. We try various combinations of the memory modules by removing one at a time. We experiment with the text classification task using the HND and the 20N datasets. Table 5 shows the reduction in performance when each of the memory modules is 'switched off'. The row 'LTM+MTM+STM' is our default setting where we use all three modules (same as Table 3).

Optimizing γ for MTM: We also try to optimize the value of γ to find the optimal number of previous steps given as input to the recurrent memory (MTM). Figure 2 shows the results for the language modeling task (WikiText-103). We can see the minimum perplexity is setting the $\gamma = 60$.

Additionally, we also optimize γ for the text classification task. For these experiments, the default value of *gamma* is set to four. However, we also experimented with changing the number of these previous sentences used to see how the performance changes. Table 6 shows the results on the number of previous sentences used in MTM, which helps to further improve the prediction accuracy on three tasks.

Dataset	No. Sentences 1	2	3	4
HND	92.04	94.63	**97.18**	96.92
20N	84.83	88.63	**89.67**	89.07
E57K	68.56	70.82	73.37	**73.97**

Table 6: Performance for number of sentences in MTM.

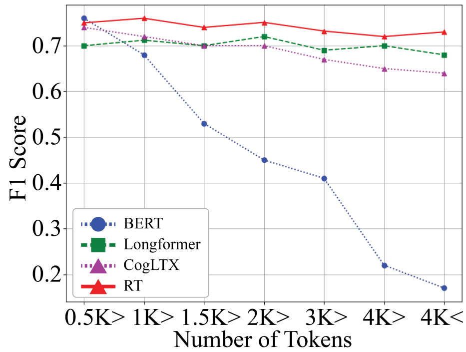

Figure 3: Increasing the number of short-term recurrent memory leads to a gradual reduction in perplexity, indicating improved performance.

Changing Token Length: Figure 3 shows a comparative analysis of the performance of four NLP models—BERT, LONGFORMER, CogLTX, and RT— for long sequences classification task. The Pan Trigger Detection dataset comprises texts ranging from 50 to 6,000 words, each tagged with one or more out of 32 distinct trigger warnings. These warnings follow a long-tailed frequency distribution, where a few labels are highly frequent, whereas the majority are increasingly scarce. We use F1 Score as an evaluation metric, revealing that BERT's effectiveness wanes with longer texts. In contrast, LONGFORMER demonstrates remarkable consistency across varying text lengths. CogLTX experiences a slight drop in performance as text length increases. RT stands out with robust performance, showing only a slight reduction in longer documents. In summary, LONGFORMER and RT prove to be more adept at managing extended sequences compared to BERT and CogLTX.

Linear vs Non-linear Readout: Table 7 compares the performance of different activation functions of the reservoir readout layer across experimented datasets; WikiText-103, HND, 20N, and E57K. The activation functions tested are Linear,

Activation Function	Datasets			
	WikiText-103	HND	20N	E57K
Linear	18.4	94.1	85.8	73.3
Tanh	17.4	94.9	86.7	73.3
Relu	16.1	95.8	89.1	73.5
Leaky Relu	16.7	95.8	88.6	73.6
Attention	**15.6**	**97.2**	**89.7**	**74.0**

Table 7: Comparative analysis of activation function efficacy in the reservoir readout layer, with WikiText-103 results measured by perplexity and remaining datasets evaluated based on accuracy.

Model	Time Complexity	Memory Complexity
Transformer	$O(K^2d)$	$O(Kd + K^2)$
RNN	$O(Kd^2)$	$O(Kd)$
LONGFORMER	$O(Kd^2 + gKd)$	$O(Kqd)$
Mamba	$O(Krd)$	$O(rd^2)$
RT	$O(Kqd)$	$O(qd + q^2 + n^2)$

Table 8: Comparison of time and memory complexity.

Tanh, Relu, Leaky Relu, and Attention. Across all datasets, the Attention activation function consistently outperforms the others. This shows that non-linear activation functions like Attention can enhance network performance in language processing tasks.

Time Complexity Comparison Table 8 shows the time and memory comparison of our method with other popular models. K is the input length, q is the sentence length (or window size in case of LONGFORMER), g is tokens used for global attention, r is the rank in low-rank projection of the state space for the Mamba (Gu and Dao, 2023) model, and n is the number of neurons in reservoir.

5 Related Work

A common approach when dealing with long context is to modify the attention mechanism using heuristics (Liu and Abbeel, 2024; Zaheer et al., 2020) and represent very long context as fixed-length representation (Peters et al., 2018; Devlin et al., 2018). These approaches by themselves cause information loss and thus result in lower performance for downstream tasks (Li et al., 2024).

As the amount of input data increases, a naive idea to handle long context is to make the model architecture much larger, e.g. LlaMa 3 (Meta, 2024) and Mistral (Jiang et al., 2023) which can handle up to 8K context length. However, this approach cannot be scaled to an ever-increasing context length (Mohtashami and Jaggi, 2024; Kryściński et al., 2021). Therefore, a common approach to handle long context is to represent the previous history as a fixed size representation (Kanerva, 1988) or to modify the attention mechanism using heuristics (Liu and Abbeel, 2024; Zaheer et al., 2020; Beltagy et al., 2020) and represent very long context as fixed-length representation (Peters et al., 2018; Devlin et al., 2018). For example, (Munkhdalai et al., 2024) use this idea and empirically verify for up to 1 million length input sequence. One idea is to offload cross-attention to a single k-NN index (Bertsch et al., 2024). Tworkowski et al. (2024) modify the LlaMA model to handle long context. Other ideas modify the attention mechanism, including block-wise computation (Liu and Abbeel, 2024) of the self-attention mechanism, ring attention (Liu et al., 2023), and sparse attention(Zaheer et al., 2020). Researchers have also used RNN blocks within a deep neural network (Munkhdalai et al., 2019) or the transformer model (Feng et al., 2024; Bulatov et al., 2022; Wang et al., 2019b; Kim et al., 2018; Bulatov et al., 2023). State-space models represent the model's state as fixed-size representation (Gu and Dao, 2023). These approaches by themselves cause information loss and there is certainly room to improve on downstream tasks (Li et al., 2024).

Researchers have tried integrating traditional Reservoir Computing (RC) models (Jaeger, 2001; Maass et al., 2002; Xia et al., 2023) with state-of-the-art models for processing temporal data(Wang et al., 2023). This integration has shown promise in fields like speech recognition (Nako et al., 2023; Ibrahim et al., 2021) and time series prediction (Shahi et al., 2022; Bianchi et al., 2020; Platt et al., 2022; Shen et al., 2020). However, to the best of our knowledge, this idea has not been used for textual input. We propose the novel idea of capturing dependencies from three different context lengths and representing them as fixed-length representations.

6 Conclusion

We propose a Reservoir Transformer model that can handle long input sequences without increasing training data sets or training time. The novelty of our approach is the memory module which helps the model represent variable-length context. This ensures that the model can capture these temporal dependencies within text thus improving the model's performance on downstream tasks. We integrate our method with two different transformer architectures; BERT and BlenderBot, and show significant improvement for the language modeling, dialogue modeling, and text classification tasks.

7 Limitations

The Reservoir Transformer model presents a notable step forward in processing extensive sequences in natural language tasks. However, it is important to acknowledge its constraints. Primarily, its proficiency in managing shorter sequences or tasks that do not significantly depend on long-span connections may not be as pronounced. Furthermore, a traditional Transformer model that considers every token theoretically could outperform the Reservoir model, albeit with a substantial increase in computational demands. This positions the Reservoir approach as a balance between efficiency and performance. Nonetheless, there remains a potential for loss of information, particularly with dependencies that extend over very long terms.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Joshua Ainslie, Tao Lei, Michiel de Jong, Santiago Ontañón, Siddhartha Brahma, Yury Zemlyanskiy, David Uthus, Mandy Guo, James Lee-Thorp, Yi Tay, et al. 2023. Colt5: Faster long-range transformers with conditional computation. *arXiv preprint arXiv:2303.09752*.

Jahangir Alam, Woo Hyun Kang, and Abderrahim Fathan. 2023. Hybrid neural network with cross-and self-module attention pooling for text-independent speaker verification. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

He Bai, Peng Shi, Jimmy Lin, Yuqing Xie, Luchen Tan, Kun Xiong, Wen Gao, and Ming Li. 2021. Segatron: Segment-aware transformer for language modeling and understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12526–12534.

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

Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew Gormley. 2024. Unlimiformer: Long-range transformers with unlimited length input. *Advances in Neural Information Processing Systems*, 36.

Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew R Gormley. 2023. Unlimiformer: Long-range transformers with unlimited length input. *arXiv preprint arXiv:2305.01625*.

Filippo Maria Bianchi, Simone Scardapane, Sigurd Løkse, and Robert Jenssen. 2020. Reservoir computing approaches for representation and classification of multivariate time series. *IEEE transactions on neural networks and learning systems*, 32(5):2169–2179.

Aydar Bulatov, Yuri Kuratov, Yermek Kapushev, and Mikhail S Burtsev. 2023. Scaling transformer to 1m tokens and beyond with rmt. *arXiv preprint arXiv:2304.11062*.

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. 2022. Recurrent memory transformer. *Advances in Neural Information Processing Systems*, 35:11079–11091.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, and Ion Androutsopoulos. 2019. Large-scale multi-label text classification on eu legislation. *arXiv preprint arXiv:1906.02192*.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. 2020. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Tri Dao, Daniel Y Fu, Khaled K Saab, Armin W Thomas, Atri Rudra, and Christopher Ré. 2022. Hungry hungry hippos: Towards language modeling with state space models. *arXiv preprint arXiv:2212.14052*.

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

Ming Ding, Chang Zhou, Hongxia Yang, and Jie Tang. 2020a. Cogltx: Applying bert to long texts. *Advances in Neural Information Processing Systems*, 33:12792–12804.

Siyu Ding, Junyuan Shang, Shuohuan Wang, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2020b. Ernie-doc: A retrospective long-document modeling transformer. *arXiv preprint arXiv:2012.15688*.

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

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

Leo Feng, Frederick Tung, Hossein Hajimirsadeghi, Mohamed Osama Ahmed, Yoshua Bengio, and Greg Mori. 2024. Attention as an rnn. *arXiv preprint arXiv:2405.13956*.

Claudio Gallicchio, Alessio Micheli, and Luca Pedrelli. 2017. Deep reservoir computing: A critical experimental analysis. *Neurocomputing*, 268:87–99.

Daniel J Gauthier, Erik Bollt, Aaron Griffith, and Wendson AS Barbosa. 2021. Next generation reservoir computing. *Nature communications*, 12(1):5564.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.

Mandy Guo, Joshua Ainslie, David Uthus, Santiago Ontanon, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. 2021. Longt5: Efficient text-to-text transformer for long sequences. *arXiv preprint arXiv:2112.07916*.

Chi Han, Qifan Wang, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2023. Lm-infinite: Simple on-the-fly length generalization for large language models. *arXiv preprint arXiv:2308.16137*.

Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc Le. 2022. Transformer quality in linear time. In *International Conference on Machine Learning*, pages 9099–9117. PMLR.

Hemin Ibrahim, Chu Kiong Loo, and Fady Alnajjar. 2021. Speech emotion recognition by late fusion for bidirectional reservoir computing with random projection. *IEEE Access*, 9:122855–122871.

Herbert Jaeger. 2001. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148(34):13.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Pentti Kanerva. 1988. *Sparse distributed memory*. MIT press.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*.

Gyuwan Kim and Kyunghyun Cho. 2020. Length-adaptive transformer: Train once with length drop, use anytime with search. *arXiv preprint arXiv:2010.07003*.

Hyunwoo Kim, Jack Hessel, Liwei Jiang, Peter West, Ximing Lu, Youngjae Yu, Pei Zhou, Ronan Le Bras, Malihe Alikhani, Gunhee Kim, Maarten Sap, and Yejin Choi. 2022. Soda: Million-scale dialogue distillation with social commonsense contextualization. *ArXiv*, abs/2212.10465.

Seungryong Kim, Stephen Lin, Sang Ryul Jeon, Dongbo Min, and Kwanghoon Sohn. 2018. Recurrent transformer networks for semantic correspondence. *Advances in neural information processing systems*, 31.

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

Wojciech Kryściński, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. 2021. Booksum: A collection of datasets for long-form narrative summarization. *arXiv preprint arXiv:2105.08209*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Jia-Nan Li, Quan Tu, Cunli Mao, Zhengtao Yu, Ji-Rong Wen, and Rui Yan. 2024. Streamingdialogue: Prolonged dialogue learning via long context compression with minimal losses. *arXiv preprint arXiv:2403.08312*.

Shanda Li, Chong You, Guru Guruganesh, Joshua Ainslie, Santiago Ontanon, Manzil Zaheer, Sumit Sanghai, Yiming Yang, Sanjiv Kumar, and Srinadh Bhojanapalli. 2023. Functional interpolation for relative positions improves long context transformers. *arXiv preprint arXiv:2310.04418*.

Yuxiao Lin, Yuxian Meng, Xiaofei Sun, Qinghong Han, Kun Kuang, Jiwei Li, and Fei Wu. 2021. Bertgcn: Transductive text classification by combining gcn and bert. *arXiv preprint arXiv:2105.05727*.

Hao Liu and Pieter Abbeel. 2024. Blockwise parallel transformers for large context models. *Advances in Neural Information Processing Systems*, 36.

Hao Liu, Matei Zaharia, and Pieter Abbeel. 2023. Ring attention with blockwise transformers for near-infinite context. *arXiv preprint arXiv:2310.01889*.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*.

Yang Liu, Jiaxiang Liu, Li Chen, Yuxiang Lu, Shikun Feng, Zhida Feng, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2022. Ernie-sparse: Learning hierarchical efficient transformer through regularized self-attention. *arXiv preprint arXiv:2203.12276*.

Xuezhe Ma, Xiang Kong, Sinong Wang, Chunting Zhou, Jonathan May, Hao Ma, and Luke Zettlemoyer. 2021. Luna: Linear unified nested attention. *Advances in Neural Information Processing Systems*, 34:2441–2453.

Wolfgang Maass, Thomas Natschläger, and Henry Markram. 2002. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, 14(11):2531–2560.

Pedro Henrique Martins, Zita Marinho, and André FT Martins. 2021. ∞-former: Infinite memory transformer. *arXiv preprint arXiv:2109.00301*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. An analysis of neural language modeling at multiple scales. *arXiv preprint arXiv:1803.08240*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Meta. 2024. Llama 3.

Amirkeivan Mohtashami and Martin Jaggi. 2023. Landmark attention: Random-access infinite context length for transformers. *arXiv preprint arXiv:2305.16300*.

Amirkeivan Mohtashami and Martin Jaggi. 2024. Random-access infinite context length for transformers. *Advances in Neural Information Processing Systems*, 36.

Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. 2024. Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143*.

Tsendsuren Munkhdalai, Alessandro Sordoni, Tong Wang, and Adam Trischler. 2019. Metalearned neural memory. *Advances in Neural Information Processing Systems*, 32.

Eishin Nako, Kasidit Toprasertpong, Ryosho Nakane, Mitsuru Takenaka, and Shinichi Takagi. 2023. Reservoir computing system with hzo/si fefets in parallel configuration: Experimental demonstration of speech classification. *IEEE Transactions on Electron Devices*.

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

Raghavendra Pappagari, Piotr Zelasko, Jesús Villalba, Yishay Carmiel, and Najim Dehak. 2019. Hierarchical transformers for long document classification. In *2019 IEEE automatic speech recognition and understanding workshop (ASRU)*, pages 838–844. IEEE.

Hyunji Hayley Park, Yogarshi Vyas, and Kashif Shah. 2022. Efficient classification of long documents using transformers. *arXiv preprint arXiv:2203.11258*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Jason A Platt, Stephen G Penny, Timothy A Smith, Tse-Chun Chen, and Henry DI Abarbanel. 2022. A systematic exploration of reservoir computing for forecasting complex spatiotemporal dynamics. *Neural Networks*, 153:530–552.

Ofir Press, Noah A Smith, and Omer Levy. 2019. Improving transformer models by reordering their sublayers. *arXiv preprint arXiv:1911.03864*.

Ofir Press, Noah A Smith, and Mike Lewis. 2020. Shortformer: Better language modeling using shorter inputs. *arXiv preprint arXiv:2012.15832*.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. 2019. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Kurt Shuster, Eric M Smith, et al. 2020. Recipes for building an open-domain chatbot. *arXiv preprint arXiv:2004.13637*.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68.

Shahrokh Shahi, Flavio H Fenton, and Elizabeth M Cherry. 2022. Prediction of chaotic time series using recurrent neural networks and reservoir computing techniques: A comparative study. *Machine learning with applications*, 8:100300.

Sheng Shen, Alexei Baevski, Ari S Morcos, Kurt Keutzer, Michael Auli, and Douwe Kiela. 2020. Reservoir transformers. *arXiv preprint arXiv:2012.15045*.

Eric Michael Smith, Mary Williamson, Kurt Shuster, Jason Weston, and Y-Lan Boureau. 2020. Can you put it all together: Evaluating conversational agents' ability to blend skills.

Simeng Sun and Mohit Iyyer. 2021. Revisiting simple neural probabilistic language models. *arXiv preprint arXiv:2104.03474*.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. Efficient transformers: A survey. *ACM Computing Surveys*, 55(6):1–28.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. 2024. Focused transformer: Contrastive training for context scaling. *Advances in Neural Information Processing Systems*, 36.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Chenguang Wang, Mu Li, and Alexander J Smola. 2019a. Language models with transformers. *arXiv preprint arXiv:1904.09408*.

Tiancheng Wang, Huaping Liu, Di Guo, and Xi-Ming Sun. 2023. Continual deep residual reservoir computing for remaining useful life prediction. *IEEE Transactions on Industrial Informatics*.

Zhiwei Wang, Yao Ma, Zitao Liu, and Jiliang Tang. 2019b. R-transformer: Recurrent neural network enhanced transformer. *arXiv preprint arXiv:1907.05572*.

Ji Xia, Junyu Chu, Siyang Leng, and Huanfei Ma. 2023. Reservoir computing decoupling memory–nonlinearity trade-off. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 33(11).

Jing Xu, Arthur Szlam, and Jason Weston. 2021a. Beyond goldfish memory: Long-term open-domain conversation. *arXiv preprint arXiv:2107.07567*.

Peng Xu, Xinchi Chen, Xiaofei Ma, Zhiheng Huang, and Bing Xiang. 2021b. Contrastive document representation learning with graph attention networks. *arXiv preprint arXiv:2110.10778*.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*.

A Equations

Reservoir:
$x_t = \frac{1}{\sqrt{n}} f(W_r x_{t-1} + W_i H_{t-1})$

Linear Readout:
$o_t = W_o x_t$

Non-Linear Readout:
$o'_t = \sigma(W_o x_t)$

Concatenation:
$z_t = \mu_1 \cdot o_t \oplus \mu_2 \cdot (\beta_{t-\gamma} \oplus \cdots \oplus \beta_{t-2} \oplus \beta_{t-1})$
$\oplus \mu_3 \cdot (e(w_1) \oplus e(w_2) \oplus \cdots \oplus e(w_{tK}))$

Neural Network:
$y_{ti} = \mathbb{M}(z_t; w_{ti})$

Loss Function:
$\mathcal{L}(y^*, y) = \frac{1}{T} \sum_{t=1}^{T} y_t^* \log(y_t)$

Attention Pooling:
$\beta_{t-1} = \sum_{i=1}^{s} \alpha_i H_{t-1}^i,$

Attention Pooling Softmax:
$\alpha_i = \frac{\exp(e_i)}{\sum_{i=1}^{d} \exp(e_i)}$

Attention Pooling Score:
$e_i = v_i \tanh(W_i \cdot H_{t-1}^i + b_i)$

A.1 Training details

We utilize three Transformer implementations: BERT, BART, and on Blenderbot. The architecture and hyperparameter settings are the default ones from the original papers (Devlin et al., 2018; Roller et al., 2020). During the training process, we utilize the Adam optimizer with a decay of 0.01 and a linear schedule learning rate starting from $2e-5$. However, in mask language modeling (MLM) tasks, the cross-entropy loss is commonly employed to optimize the model's predictions. In MLM, a certain percentage of input tokens are randomly masked to train the model to predict the masked tokens based on their surrounding context.

Mathematically, the cross-entropy loss is defined as follows:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{T} \sum_{i=1}^{T} \sum_{j=1}^{V} y_{ij} \log(p_{ij}), \qquad (12)$$

where T is the total number of instances, V is the size of the vocabulary, y_{ij} is the binary indicator (0 or 1) for whether the true label is j for the i-th instance, and p_{ij} is the predicted probability of the i-th instance belonging to class j.

In mask language modeling, the input sequences are modified by randomly replacing some tokens with a special [MASK] token. The model's objective is then to predict the original tokens based on the context provided by the surrounding tokens. The cross-entropy loss is calculated by comparing the predicted probabilities of the masked tokens with their true labels.

Additionally, BERT often includes next-sentence prediction (NSP) as an auxiliary task during pretraining. NSP determines whether two sentences in a pair are contiguous in the original text. This task helps the model capture relationships between sentences. Cross-entropy loss is also used to optimize the predictions of sentence pairs for the NSP task.

Furthermore, in token generation models like BlenderBot, the Cross-entropy loss is again employed to train these models, comparing the predicted probability distribution of tokens in the generated sequence with the target sequence.

We train our model by adopting the methodology outlined in Algorithm 1. The pretraining phase encompasses two models: BERT and Blenderbot.

For BERT, we adopt the parameter settings described in the original BERT paper by Devlin et al. (Devlin et al., 2018). Our primary focus lies in optimizing the Masked Language Model (MLM) and Next Sentence Prediction (NSP) objectives for the pretraining model M. Pretraining, the Reservoir Bert model, involves employing the Book Corpus dataset (Zhu et al., 2015) and fine-tuning with the WikiText-103 dataset (Merity et al., 2016). In the reservoir setting, we use 3000 units with a spectral radius of 0.5. The leaky rate is set to 0.35. Furthermore, we allocate 50 units of memory for recurrent settings and 50 for long-term memory which is reservoir readout size.

Similar to BERT, training the Blenderbot model adheres to the original Blenderbot hyperparameters outlined by Roller et al. (Roller et al., 2020). Pretraining of Reservoir Blenderbot involves utilizing the Soda dataset (Kim et al., 2022), followed by fine-tuning with the Blend Skills Talk dataset (Smith et al., 2020). In the reservoir setting of Blenderbot, we use 1000 units with a spectral radius of 0.5. The leaky rate is set to 0.35. Additionally, we reserve 15 units of memory for recurrent settings and 50 for long-term memory.

For the text classification dataset, we use 500 reservoir size with a spectral radius of 0.7. The leaky rate is set to 0.35. We set 5 recurrent memory and 20 for long-term memory. For all experiments, we have used the Transformer model dropout 0.1, attention dropout of 0.1, and weight decay 0.05.

B Training Algorithm

Algorithm 1 Training algorithm of Deep Reservoir Computing with Recurrent Transformer

Require:
- $U_{i:T}, Y_{i:T}$: Dataset
- **function** R(u_t) ▷ Reservoir
  - $x_t \leftarrow \frac{1}{\sqrt{N}} f(W_r x_{t-1} + W_i u_t)$ ▷ By Equation 6
  - **return** o_t ▷ By equation 7
- **end function**

Ensure: Optimize $Pr(Y_t|S_{1:t})$ distribution by learning a model $F(.)$
- $W_i, W_r \leftarrow \mathcal{N}(0,1)$ ▷ Weights initialization of Reservoir
- $\sigma \leftarrow \mathcal{N}(0,1)$ ▷ Weights initialization of Reservoir non-linear output
- $\phi \leftarrow \mathcal{N}(0,1)$ ▷ Weights initialization of Transformer $F(.)$
- **while** $epoch < epochs$ **do**
  - **while** $t < T$ **do**
    - $o_t \leftarrow R(u_t; W_i, W_r, \sigma)$ ▷ Non-linear readout from $R(.)$ reservoir described in Equation 7
    - $\beta_{t-\gamma} = \sum_{i=1}^{s} \alpha_i H^i_{t-\gamma}$ ▷ Get STM from Equation 3
    - $z_t = \mu_1 \cdot o_t \oplus \mu_2 \cdot (\beta_{t-1} \oplus \beta_{t-2} \oplus \ldots \oplus \beta_{t-\gamma}) \oplus \mu_3 \cdot (e(w_1) \oplus e(w_2) \oplus \cdots \oplus e(w_q))$ ▷ Concatenation of all embedding described in described in Euqation 9
    - $\bar{y}_t \leftarrow F(z_t; \phi)$
  - **end while**
  - $loss \leftarrow \mathcal{L}_{\text{CE}}(\bar{y}_{1:T}, y_{1:T})$ ▷ Calculating loss for every time step z by equation 12
  - $\phi, \sigma, e, \kappa \leftarrow update$ ▷ Update parameters
- **end while**

C Additional Results

Table 9 shows additional baseline results for the text classification task.

Model/Dataset	HND	20N	E57K
GRAPH-ROBERTA (Xu et al., 2021b)	96.2	-	-
ERNIE-DOC-LARGE (Ding et al., 2020b)	96.6	-	-
ERNIE-SPARSE (Liu et al., 2022)	92.8	-	-
RMT BERT (Bulatov et al., 2022)	94.3	-	-
TextGCN (Lin et al., 2021)	-	86.3	-
BertGAT (Lin et al., 2021)	-	87.4	-
RoBERTaGAT (Lin et al., 2021)	-	86.5	-
SGC (Lin et al., 2021)	-	88.5	-
ZERO-BIGRU-LWAN (Chalkidis et al., 2019)	-	-	65.2
BIGRU-LWAN (L2V) (Chalkidis et al., 2019)	-	-	71.1
BIGRU-LWAN (ELMO) (Chalkidis et al., 2019)	-	-	71.9
Reservoir Transformer	**97.2**	**89.7**	**74.0**

Table 9: Text classification on three datasets.

D Dataset Statistics

Table 10 shows the training data samples for all the datasets we used including our custom conversational dataset.

Dataset	Training Samples
WikiText-103	1.8M
Custom Dialog data	600K
XSum	204K
HND	600K
20N	20K
E57K	57K

Table 10: Training data samples for each dataset

E Length comparison

Figure 4 shows the maximum sequence length for all the datasets we used in our experiments. 'Dialog' is our custom generated data we use for the dialogue modeling task. As shown in the plot, we experimented with varying length of context from 1.4K tokens for WikiTest-103 up to 11.7K tokens for 20N dataset.

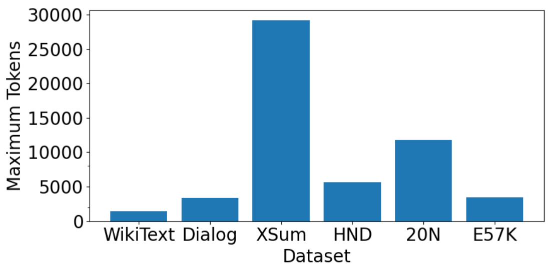

Figure 4: Maximum length of a single data sample for all the four datasets.