Introduction:
As larger models on larger datasets have often shown better performance [Li2020], the parameter size of NLP models is growing over time. For instance, RoBERTa [Liu2019, Strubell2019] was trained using a thousand GPUs, and it cost nearly five million dollars to train the final GPT-3 model [Brown2020]. Toward this end, In this project, we revisit an old idea in deep learning: echo state networks [Jaeger2001], where we add fixed layers with random features to Transformers. These layers increase the depth of the model and consequently improve its representational power, but are much more computationally efficient. The proposed method shows improvements in wall-clock compute time until convergence, as well as overall performance, on various machine translation and (masked) language modelling tasks.

Proposed Method:
Our approach is based on a very simple idea. Neural networks are trained via backpropagation, which involves consecutive steps of matrix addition and multiplication, i.e.,

$$\theta_{t+1} \leftarrow \theta_t - \eta \frac{\partial J}{\partial \theta_t}; \frac{\partial J}{\partial \theta_t} = \frac{\partial J}{\partial L_n} \frac{\partial L_n}{\partial L_{n-1}} \cdots \frac{\partial L_1}{\partial L_0} \frac{\partial L_0}{\partial x}$$

$$\theta_{t+1} \leftarrow \theta_t - \eta \frac{\partial J}{\partial \theta_t}; \frac{\partial J}{\partial \theta_t} = \frac{\partial J}{\partial L_n} \frac{\partial L_n}{\partial L_{n-1}} \cdots \frac{\partial L_1}{\partial L_0} \frac{\partial L_0}{\partial x}$$

for some objective *J*, parameterization θ and learning rate η, with the gradient computed via the chain rule, where *Li* is the *i*-th layer of the neural network and x is the input. Let *L* = Transformer(X) be a single layer in a Transformer network, i.e.,

$$\begin{aligned} H &= \text{MultiHeadSelfAttn}(\text{LayerNorm}(X)) + X \\ L &= \text{FFN}(\text{LayerNorm}(H)) + H \end{aligned}$$

$$\begin{aligned} H &= \text{MultiHeadSelfAttn}(\text{LayerNorm}(X)) + X \\ L &= \text{FFN}(\text{LayerNorm}(H)) + H \end{aligned}$$

Now, during every "backward pass", we compute the Jacobian for parameters *θL* at layer *L*, which are used to update the parameters of *θL*, as well as to compute the next layer's Jacobian, thus back-propagating the gradients. In this project, however, for some of the layers, we still backpropagate through them to compute gradients for earlier layers, but we never update their parameters. As a result, these layers stay fixed at their random initialization, saving computational resources.

Results:

We evaluate the proposed approach on a variety of well-known tasks in natural language processing, namely: machine translation, language modelling and masked language model pre-training. We propose to measure this trade-off via the area under the convergence curve (AUCC): similarly to how the area under the receiver operating characteristic (AUC-ROC) measures a classifier's performance independent of the classification threshold, AUCC measures a model's performance independent of the specific compute budget. All the experiments are benchmarked using the same hardware and set all the hyper-parameters the same as in fairseq.

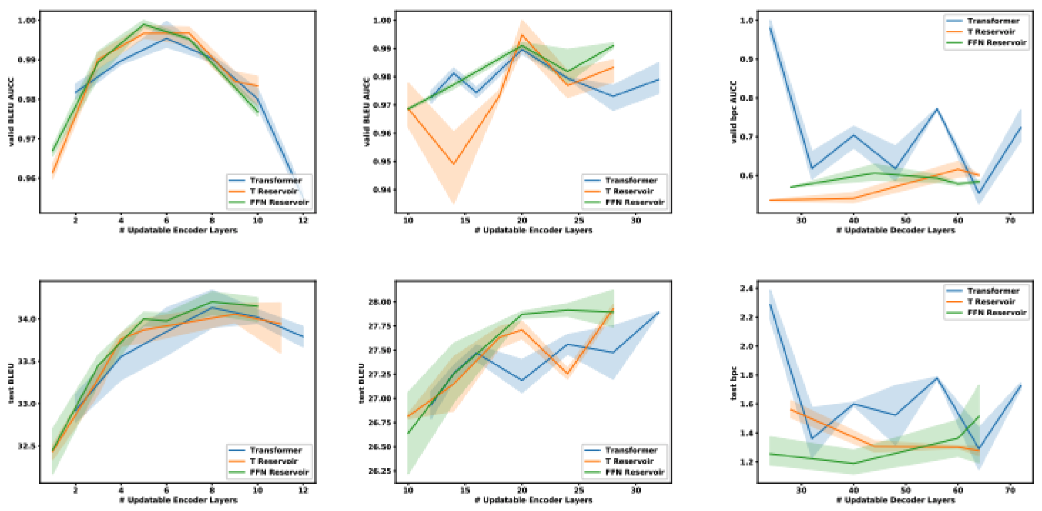
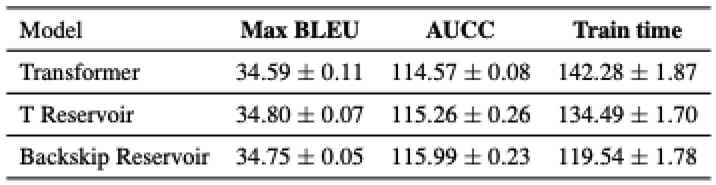


Figure 1 here shows the results for IWSLT (left) and WMT (middle), enwiki8 (right). On the *y-axis* we show validation AUCC for the BLEU metric; on the *x-axis* we show the number of updatable layers in the encoder. At test time with BLEU and during training with AUCC, reservoir transformers outperform regular transformers for almost all encoder depths. The FFN Reservoir seems to work best in both cases, which is surprising because it does not have any self-attention component at all. This finding shows that self-attention, or the mechanism to summarize context information, should be learned if present. Once the context features have been gathered, a random projection via a fixed FFN module appears to be beneficial.

Model	# Layers	Frozen	Max BLEU	Train time **until max** (in hours)	Ratio	# Params **Trainable** (Total)	Train Time each **epoch** (in hours)
Transformer	12	0	24.46 ± 0.04	15.15 ± 0.15	1	75.6M	0.505 ± 0.005
	16	0	24.52 ± 0.03	16.05 ± 0.18	1	88.2M	0.643 ± 0.006
	24	0	24.69 ± 0.05	17.61 ± 0.85	1	113.4M	0.877 ± 0.029
	32	0	24.83 ± 0.04	18.42 ± 0.28	1	138.6M	1.036 ± 0.010
T Reservoir	12	4	24.26 ± 0.08	14.11 ± 0.21	0.93	72.4M (75.6M)	0.472 ± 0.007
	16	4	24.50 ± 0.05	15.25 ± 0.28	0.95	75.6M (88.2M)	0.596 ± 0.009
	24	4	25.11 ± 0.07	15.89 ± 0.74	0.90	100.8M (113.4M)	0.776 ± 0.024
	32	4	24.66 ± 0.04	16.38 ± 0.24	0.88	126.0M (138.6M)	0.998 ± 0.009
FFN Reservoir	12	4	24.42 ± 0.05	14.01 ± 0.09	0.92	72.4M (71.4M)	0.441 ± 0.003
	16	4	24.65 ± 0.07	14.53 ± 0.17	0.91	75.6M (83.9M)	0.524 ± 0.006
	24	4	24.93 ± 0.04	12.62 ± 1.53	0.71	100.8M (109.2M)	0.743 ± 0.018
	32	4	24.98 ± 0.03	13.96 ± 0.19	0.73	126.0M (134.4M)	0.964 ± 0.007
LayerDrop	12	4	24.27 ± 0.03	14.61 ± 0.14	0.96	72.4M (75.6M)	0.489 ± 0.006
	16	4	24.15 ± 0.06	15.55 ± 0.54	0.97	75.6M (88.2M)	0.597 ± 0.017
	24	4	24.37 ± 0.05	16.25 ± 0.36	0.92	100.8M (113.4M)	0.823 ± 0.013
	32	4	23.84 ± 0.03	15.27 ± 0.38	0.83	126.0M (138.6M)	1.028 ± 0.012

The above table presents the wall-clock time (averaged over multiple runs) saved for WMT for different model types and encoder depths. We save as much as 27% time until convergence of a 24 layer model on WMT as much with the same number of updateable layers. One other noticeable point is that we can see that the T Reservoir achieves a similar performance to LayerDrop on IWSLT and WMT in terms of wall-clock per epoch and wallclock time to the best performance. However, on both tasks, FFN Reservoir performs much better than LayerDrop in terms of efficiency per epoch. As a point of reference, a three-hour gain on WMT translates to a gain of several days in the training of bigger transformer models like GPT-3 [Brown2020].

Furthermore, we extend the idea with a synthetic gradient predictor that allows for skipping the *backward pass.* As shown in the following table, the *backskip* reservoir approach leads to a higher maximum BLEU score than the regular transformer, with a much higher AUCC and much lower training time. This finding opens up intriguing possibilities for having parts of neural networks be completely frozen both in the forward as well as in the backward pass, while still contributing to the overall model computation.



Model	Max BLEU	AUCC	Train time
Transformer	34.59 ± 0.11	114.57 ± 0.08	142.28 ± 1.87
T Reservoir	34.80 ± 0.07	115.26 ± 0.26	134.49 ± 1.70
Backskip Reservoir	34.75 ± 0.05	115.99 ± 0.23	119.54 ± 1.78

Pre-trained masked language modelling architectures like RoBERTa [Liu2019] can benefit from having some of their layers frozen, both during pre-training as well as when fine-tuning on downstream tasks in our complete paper. For the next steps, we are extending these findings to find subnetworks with random initialized Transformer with pruning techniques that may still preserve the decent performance.

Milestones:
This work has been published at the ACL2021 main conference. https://arxiv.org/pdf/2012.15045.pdf

Reference:

[Liu2019] Liu, Yinhan, et al. "Roberta: A robustly optimized bert pretraining approach." arXiv preprint arXiv:1907.11692 (2019).
[Brown2020] Brown, Tom B., et al. "Language models are few-shot learners." arXiv preprint arXiv:2005.14165 (2020).

[Li2020] Li, Zhuohan, et al. "Train large, then compress: Rethinking model size for efficient training and inference of transformers." in ICML (2020).

[Wieting2019] Wieting, John, and Douwe Kiela. "No training required: Exploring random encoders for sentence classification." in ICLR (2019).

[Jaeger2001] Jaeger, Herbert. "The “echo state” approach to analysing and training recurrent neural networks-with an erratum note." in GMD Technical Report (2001).

[Strubell2019] Emma Strubell, Ananya Ganesh and Andrew McCallum. “Energy and Policy Considerations for Deep Learning in NLP.” in ACL (2019).