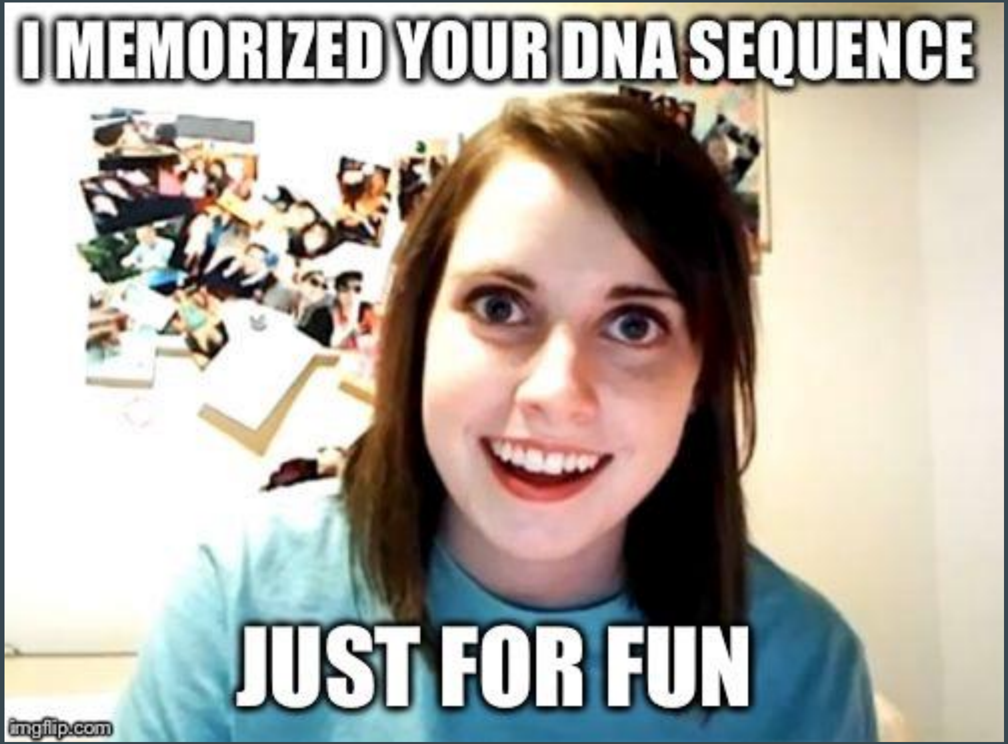
Sequence Modeling

Sepehr Sameni
NeironAI


I MEMORIZED YOUR DNA SEQUENCE
JUST FOR FUN
imgflip.com

Problem Definition

Given an ordered sequence of features, transform them into ...

- ~~A single prediction: sentiment analysis, emoji prediction, ...~~
- Another sequence with same length: PoS prediction, NER, LM, ...
    - In general we can then use these extracted features in the next task
- ~~Another sequence with different length: Summarization, NMT, TTS~~

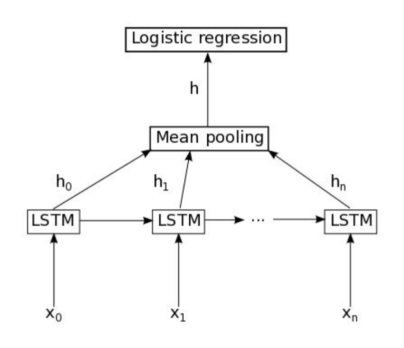
Sequence modeling. Given an input $x_{1:T} = x_1, \ldots, x_T$, a sequence model is any function $G : \mathcal{X}^T \rightarrow \mathcal{Y}^T$ such that

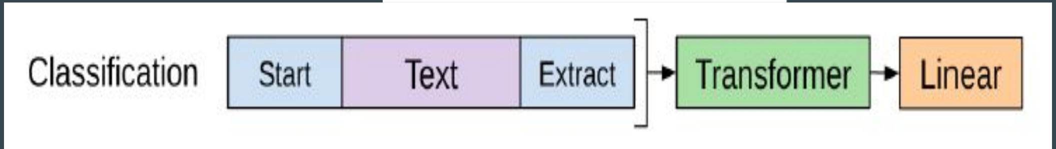
$$y_{1:T} = y_1, \ldots, y_T = G(x_1, \ldots, x_T), \tag{1}$$

where y_t should only depend on $x_{1:t}$ and not on $x_{t+1:T}$ (i.e. no leakage of information from the future). This causality constraint is essential for autoregressive modeling.

What about other problems?

- Single output:
  - Pooling
    - max, min, mean, p-norm, transformer, last
  - Classification Token
- Another sequence output:
  - Old
    - Pool and then generate
  - Attention!
    - Per-step pooling





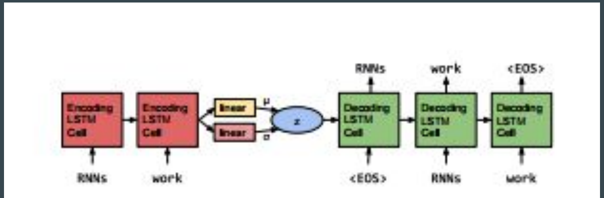
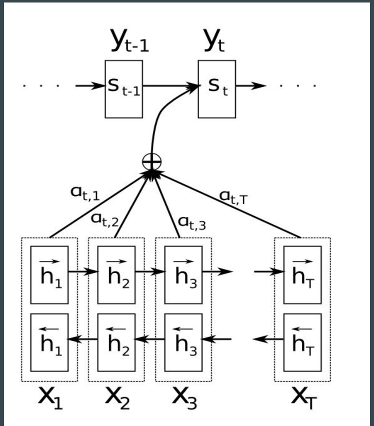


Figure 1: The core structure of our variational autoencoder language model. Words are represented using a learned dictionary of embedding vectors.



RNN

Base formula:

$$
\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)})
\end{aligned}
$$

Problem:

Assume we have a hidden state h_t at time step t. If we make things simple and remove biases and inputs, we have

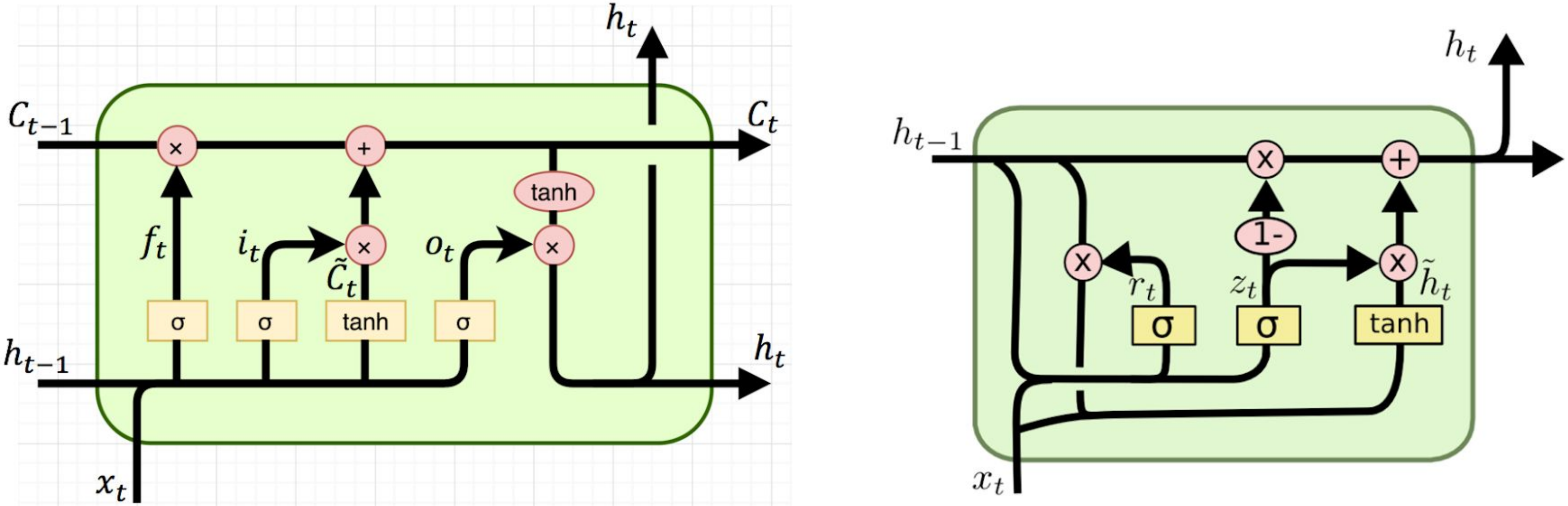
$$h_t = \sigma(w h_{t-1}).$$

Then you can show that

$$
\begin{aligned}
\frac{\partial h_{t'}}{\partial h_t} &= \prod_{k=1}^{t'-t} w\sigma'(w h_{t'-k}) \\
&= \underbrace{w^{t'-t}}_{!!!} \prod_{k=1}^{t'-t} \sigma'(w h_{t'-k})
\end{aligned}
$$

The factored marked with !!! is the crucial one. **If the weight is not equal to 1, it will either decay to zero exponentially fast in $t' - t$, or grow exponentially fast.**

GRU, LSTM



(a) Long Short-Term Memory

(b) Gated Recurrent Unit

RNNs are Awesome

RNNs have a useful Structural bias (like CNNs for image)

RNNs are Awesome

Under review as a conference paper at ICLR 2019

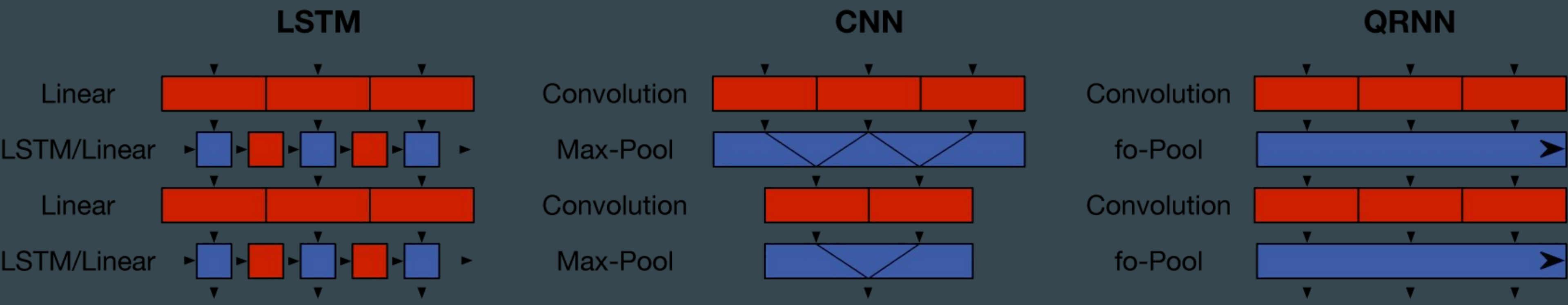
No Training Required: Exploring Random Encoders for Sentence Classification

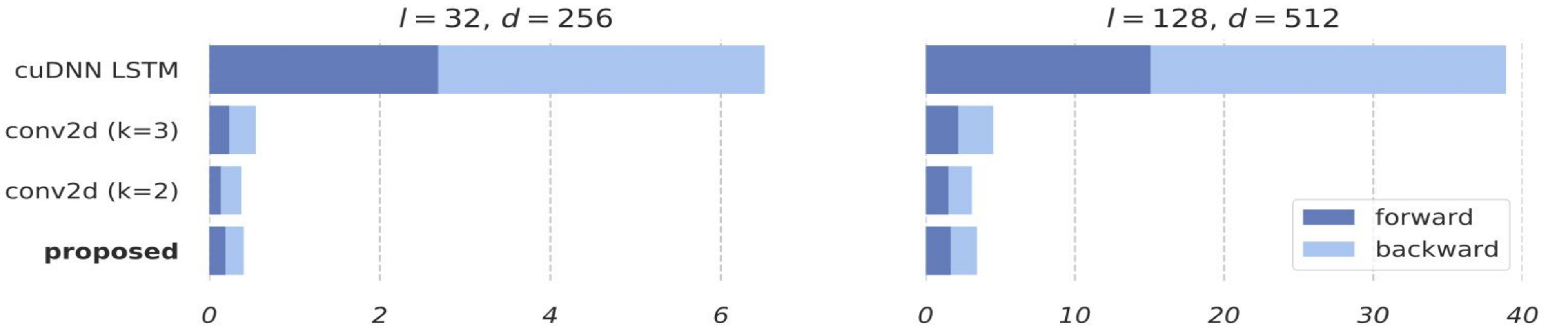
Anonymous authors
Paper under double-blind review

Abstract

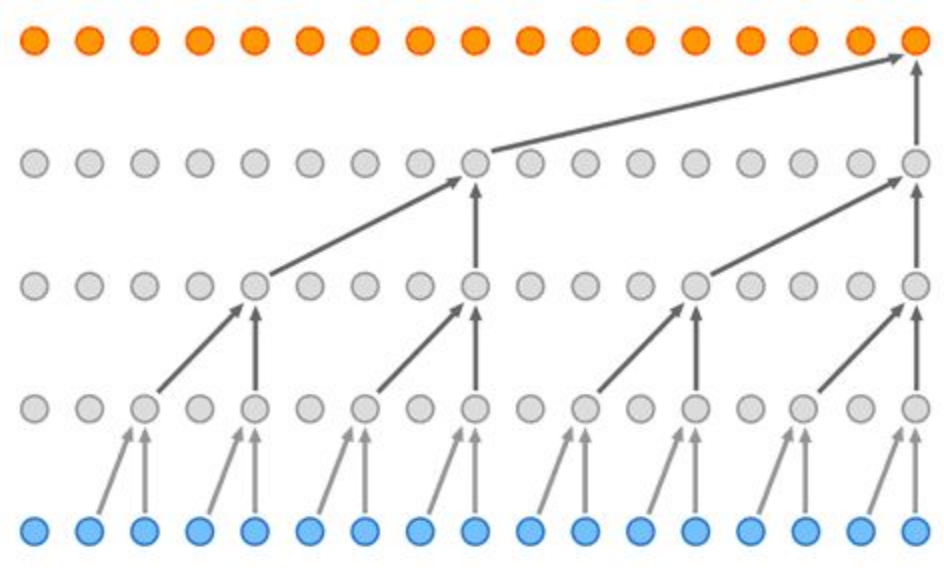
We explore various methods for computing sentence representations from pre-trained word embeddings *without any training*, i.e., using nothing but random parameterizations. Our aim is to put sentence embeddings on more solid footing by 1) looking at how much modern sentence embeddings gain over random methods—as it turns out, surprisingly little; and by 2) providing the field with more appropriate baselines going forward—which are, as it turns out, quite strong. We also make important observations about proper experimental protocol for sentence classification evaluation, together with recommendations for future research.

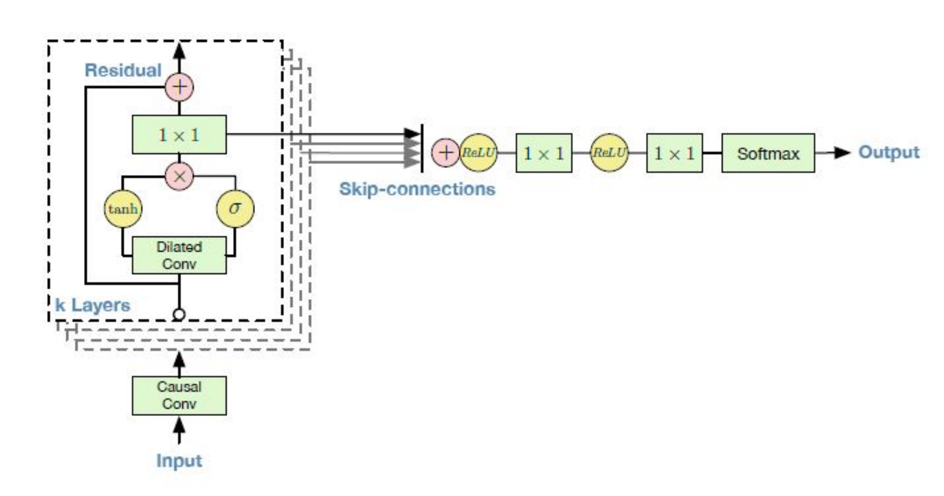
QRNN, SRU


LSTM
Linear
LSTM/Linear
Linear
LSTM/Linear
CNN
Convolution
Max-Pool
Convolution
Max-Pool
QRNN
Convolution
fo-Pool
Convolution
fo-Pool

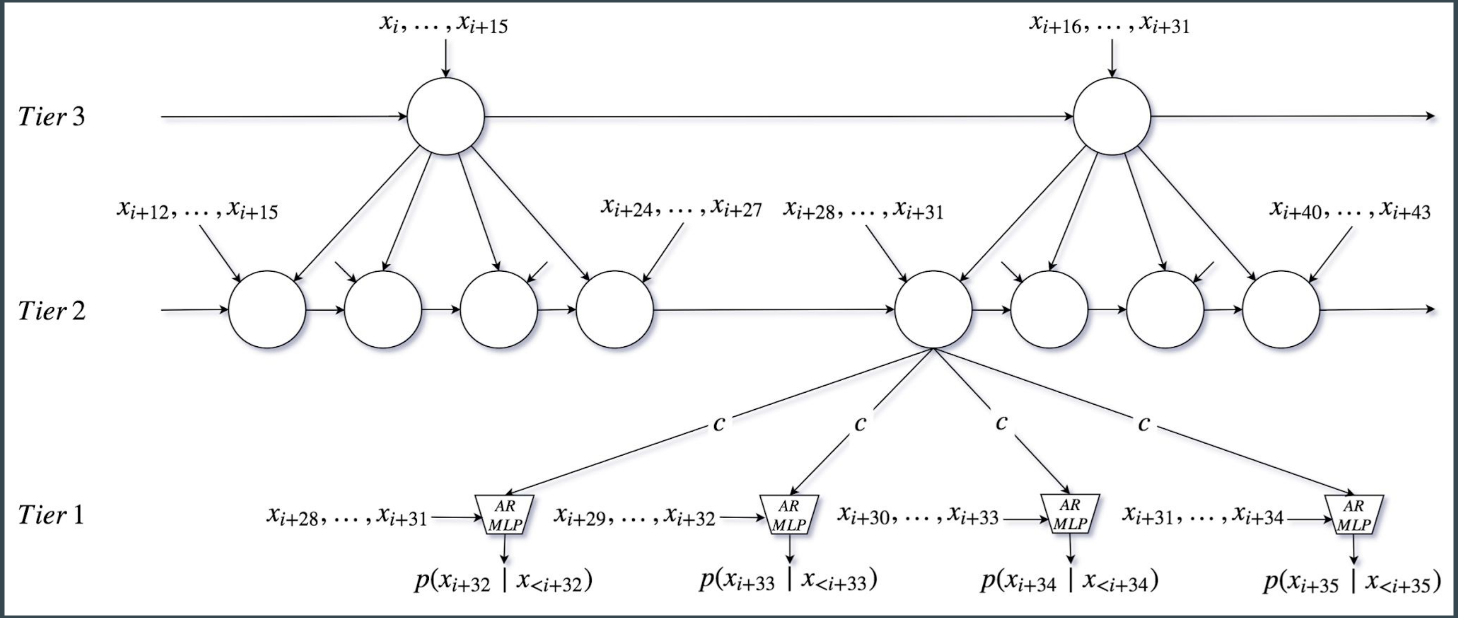

$l = 32, d = 256$
$l = 128, d = 512$
cuDNN LSTM
conv2d (k=3)
conv2d (k=2)
proposed
forward
backward
0
2
4
6
0
10
20
30
40

~~CNN~~ WaveNet



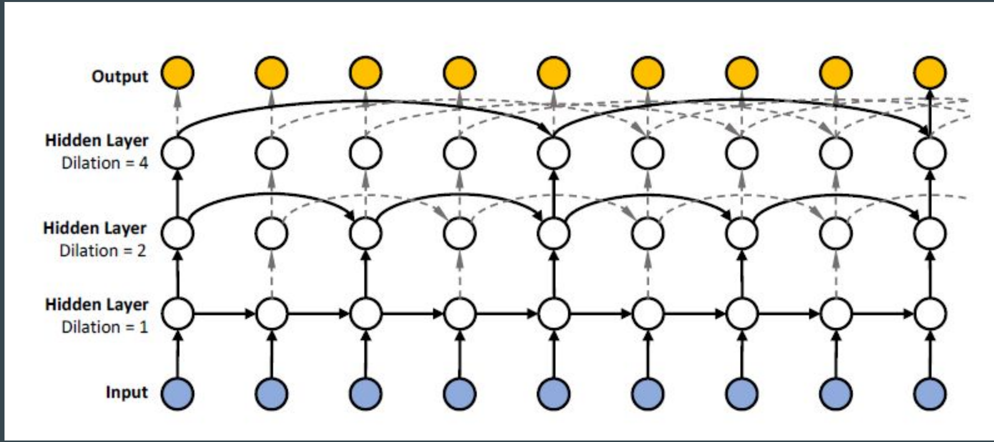

Residual
1 × 1
tanh
σ
Dilated Conv
k Layers
Skip-connections
ReLU
1 × 1
ReLU
1 × 1
Softmax
Output
Causal Conv
Input

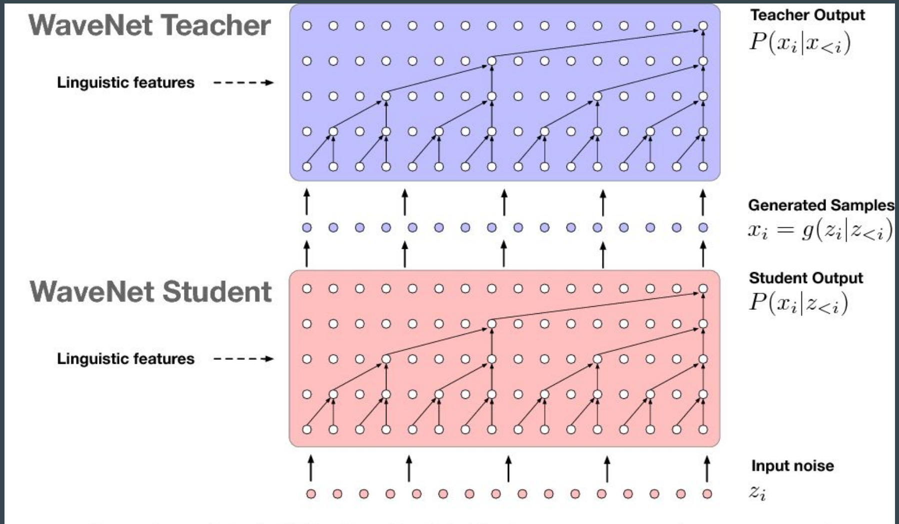
~~Hierarchical RNN~~ SampleRNN



WaveNet++

- TCN
- Wave RNN
- Dilated RNN
- Parallel Wavenet





Neural Architecture Search

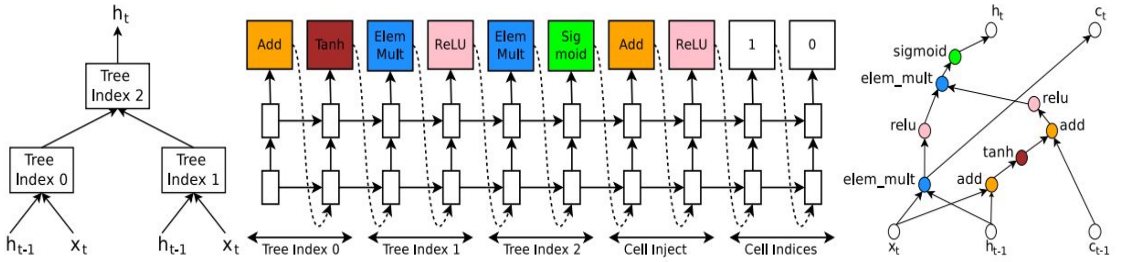


Figure 5: An example of a recurrent cell constructed from a tree that has two leaf nodes (base 2) and one internal node. Left: the tree that defines the computation steps to be predicted by controller. Center: an example set of predictions made by the controller for each computation step in the tree. Right: the computation graph of the recurrent cell constructed from example predictions of the controller.

NAS cells

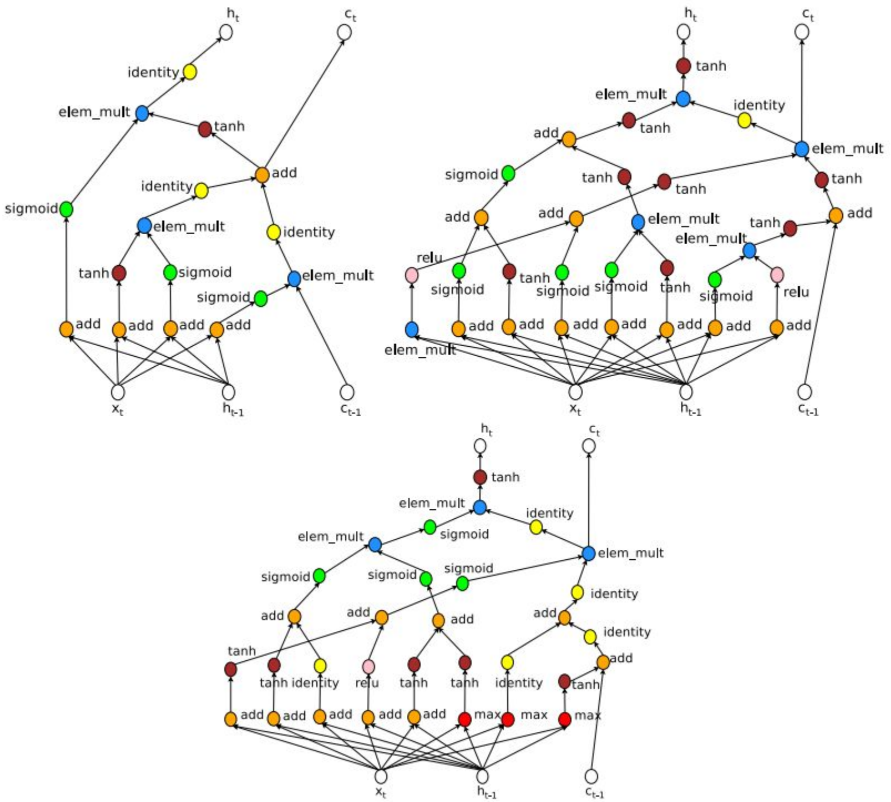


Figure 8: A comparison of the original LSTM cell vs. two good cells our model found. Top left: LSTM cell. Top right: Cell found by our model when the search space does not include max and sin. Bottom: Cell found by our model when the search space includes max and sin (the controller did not choose to use the sin function).

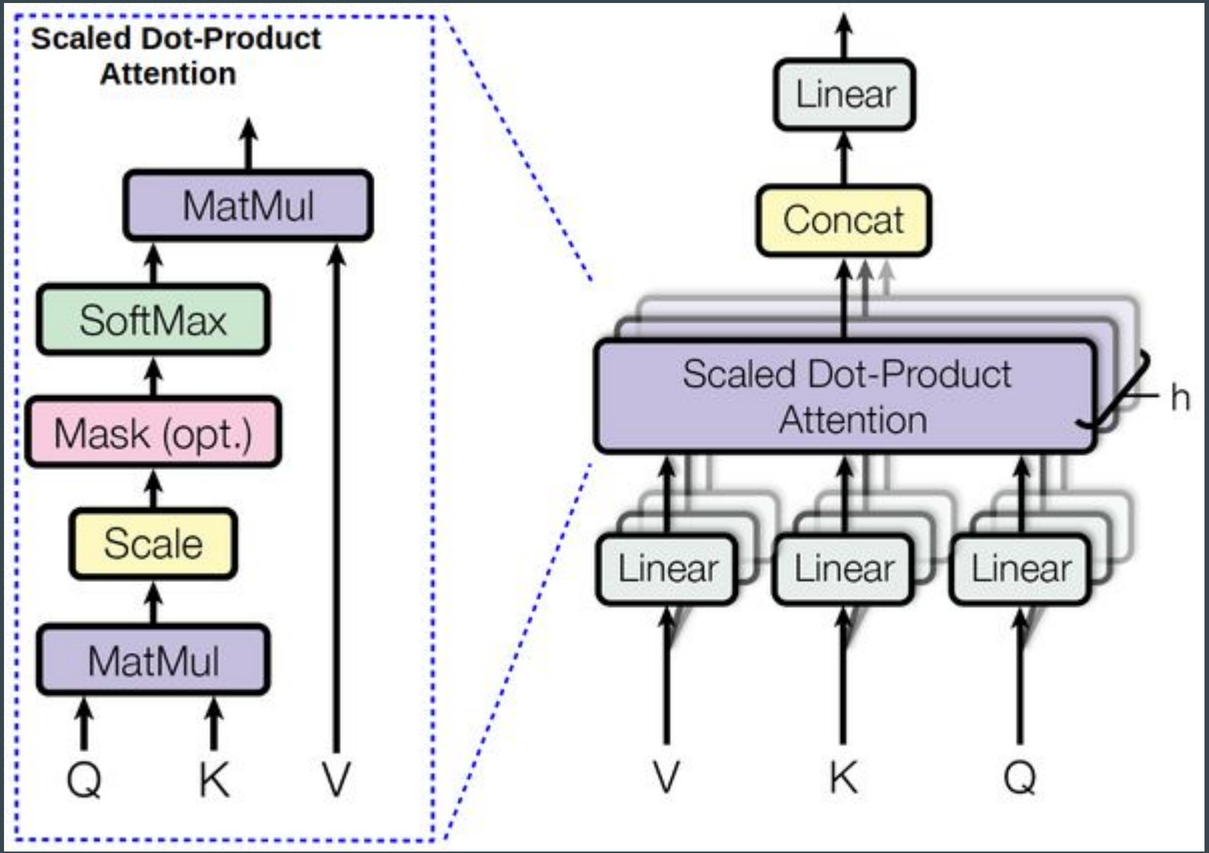
The ~~Empire~~ LSTM Strikes Back!

Model	PTB Validation	PTB Test	WT2 Validation	WT2 Test
AWD-LSTM (tied)	60.0	57.3	68.6	65.8
– fine-tuning	60.7	58.8	69.1	66.0
– NT-ASGD	66.3	63.7	73.3	69.7
– variable sequence lengths	61.3	58.9	69.3	66.2
– embedding dropout	65.1	62.7	71.1	68.1
– weight decay	63.7	61.0	71.9	68.7
– AR/TAR	62.7	60.3	73.2	70.1
– full sized embedding	68.0	65.6	73.7	70.7
– weight-dropping	71.1	68.9	78.4	74.9

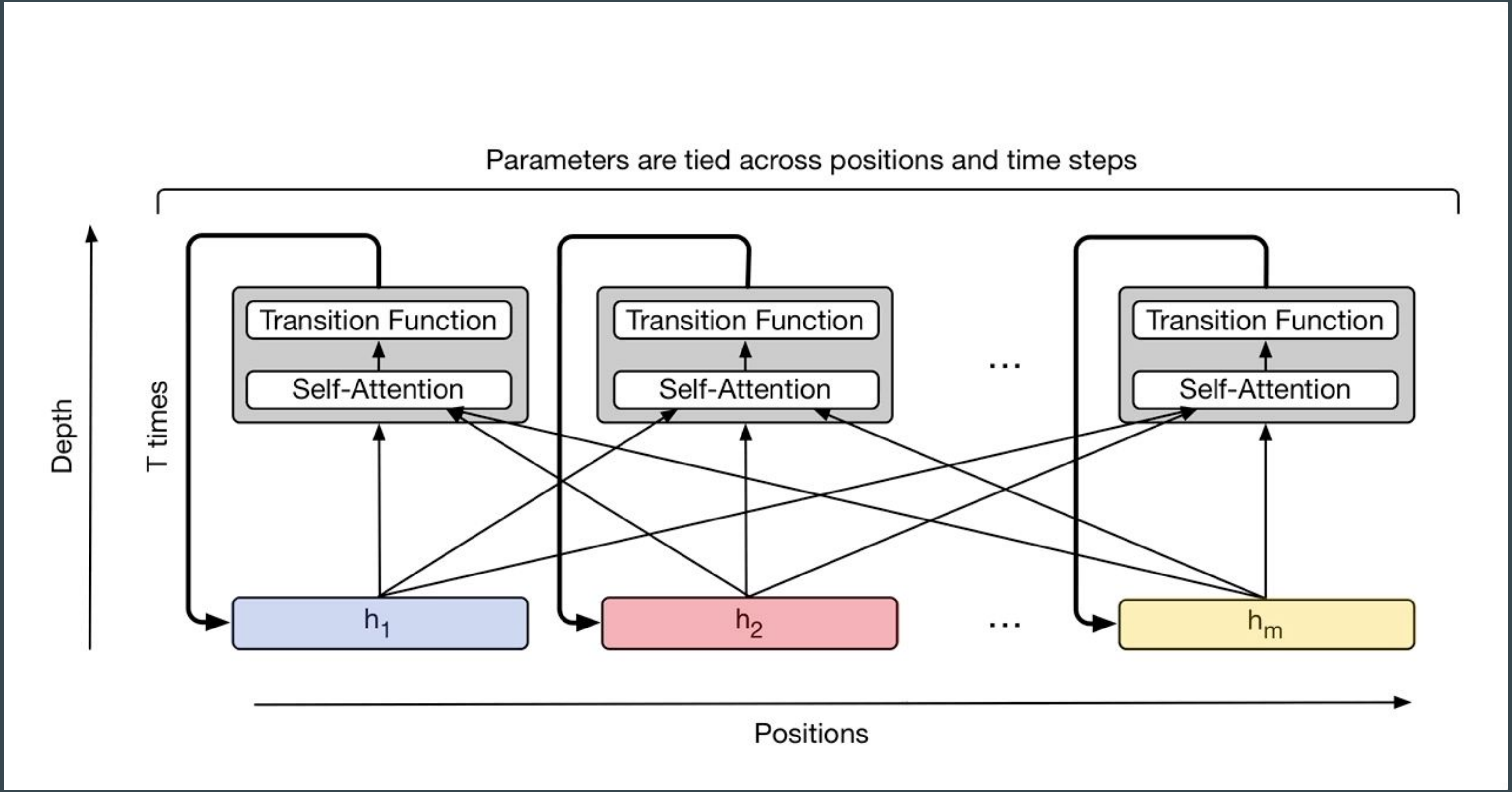
Attention


ALL YOU NEED IS LOVE
FALSE. ALL YOU NEED IS ATTENTION
imgflip.com

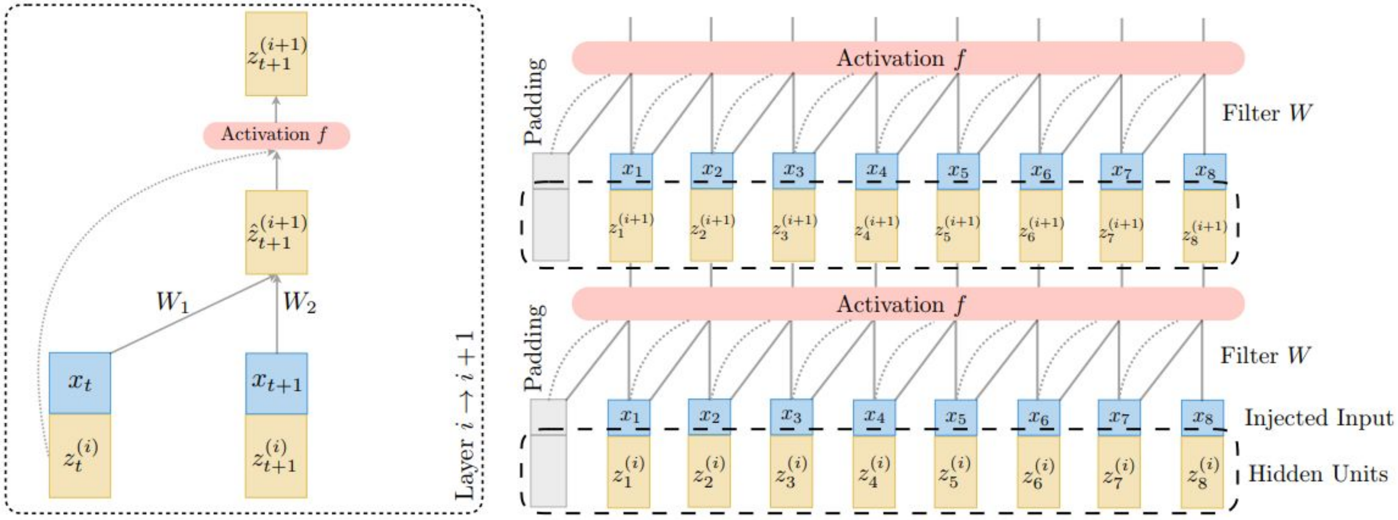
Transformer



Recurrent Transformer



Trellis Net



(a) TrellisNet at an atomic level

(b) TrellisNet on a sequence of units

Figure 1: The interlayer transformation of TrellisNet, at an atomic level (time steps t and $t + 1$, layers i and $i + 1$) and on a longer sequence (time steps 1 to 8, layers i and $i + 1$).

Trellis Net

Table 1: Test perplexities (ppl) on word-level language modeling with the Penn Treebank (PTB) corpus (with and without mixture of softmaxes (MoS) (Yang et al., 2018)). $^{\ell}$ means lower is better.

Word-level PTB without MoS			Word-level PTB with MoS		
Model	Size	Test ppl$^{\ell}$	Model	Size	Test ppl$^{\ell}$
NAS Cell (Zoph & Le, 2017)	54M	62.4	AWD-LSTM-MoC (Yang et al., 2018)	22M	57.55
AWD-LSTM (Merity et al., 2018b)	24M	58.8	AWD-LSTM-MoS (Yang et al., 2018)	24M	55.97
(Black-box tuned) NAS (Melis et al., 2018)	24M	59.7	DARTS (Liu et al., 2018)	23M	56.10
(Black-box tuned) LSTM + skip conn. (Melis et al., 2018)	24M	58.3	ENAS (Pham et al., 2018)	24M	55.80
Ours - TrellisNet	24M	**56.97**	**Ours - TrellisNet-MoS**	25M	**54.67**
Ours - TrellisNet (1.4x larger)	33M	**56.80**	**Ours - TrellisNet-MoS (1.4x larger)**	34M	**54.19**

Table 2: Test perplexities (ppl) on word-level WikiText-103 and test bits-per-character (bpc) on character-level Penn Treebank. $^{\ell}$ means lower is better.

Word-level WikiText-103				Character-level PTB		
Model	Size	Test ppl$^{\ell}$	Epo.	Model	Size	Test bpc$^{\ell}$
LSTM (Grave et al., 2017b)	-	48.7	-	IndRNN (Li et al., 2018)	12.0M	1.23
LSTM+cont. cache (Grave et al., 2017b)	-	40.8	-	Hyper LSTM (Ha et al., 2017)	14.4M	1.219
Generic TCN (Bai et al., 2018)	150M	45.2	-	NAS Cell (Zoph & Le, 2017)	16.3M	1.214
Gated Linear ConvNet (Dauphin et al., 2017)	230M	37.2	60	FS-LSTM-2 (Mujika et al., 2017)	7.2M	1.19
AWD-QRNN (Merity et al., 2018a)	159M	33.0	24	Quasi-RNN (Merity et al., 2018a)	13.8M	1.187
Relational Memory Core (Santoro et al., 2018)	195M	31.6	90	AWD-LSTM (Merity et al., 2018a)	13.8M	1.175
Ours - TrellisNet	180M	**30.35**	**22**	**Ours - TrellisNet**	13.4M	**1.159**

Questions?


ONE DOES NOT SIMPLY FINISH A PRESENTATION
WITHOUT ASKING IF THERE ARE ANY QUESTIONS
emegenerator.net