CMC MSU Department of Algorithmic Languages
Samsung Moscow Research Center

Recent approaches to natural language processing with neural networks
Современные подходы к обработке текстов с помощью нейронных сетей

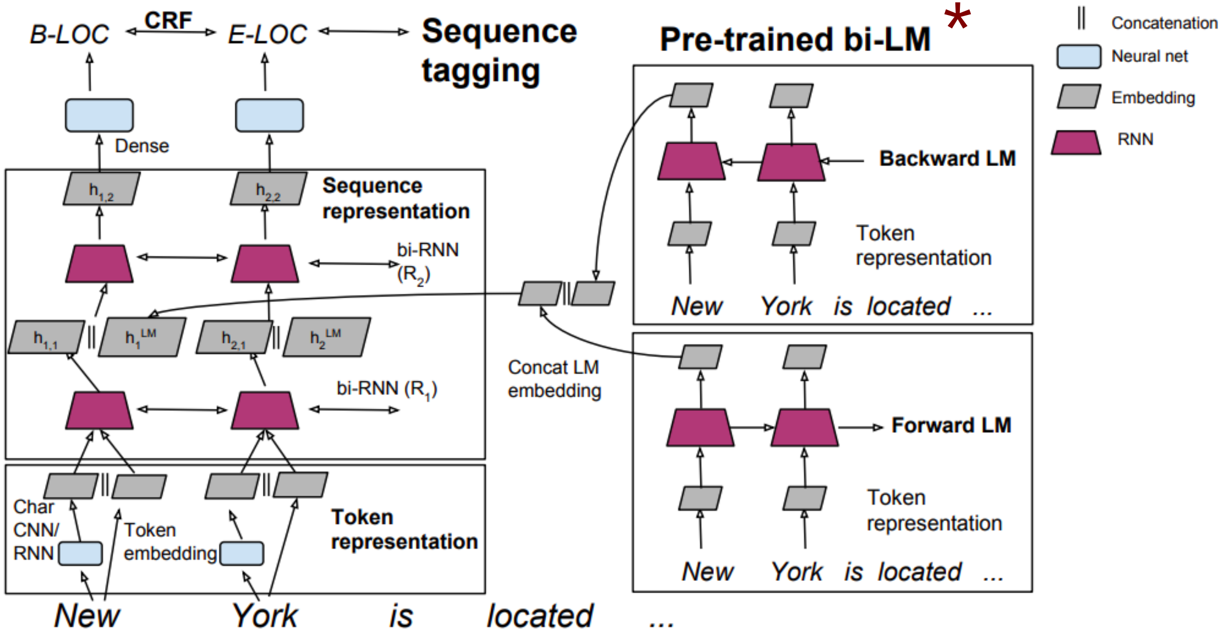
Lecture 8. Knowledge transfer in NLP: ELMo and ULMFiT

Arefyev Nikolay

CMC MSU Department of Algorithmic Languages &
Samsung Moscow Research Center

ELMo

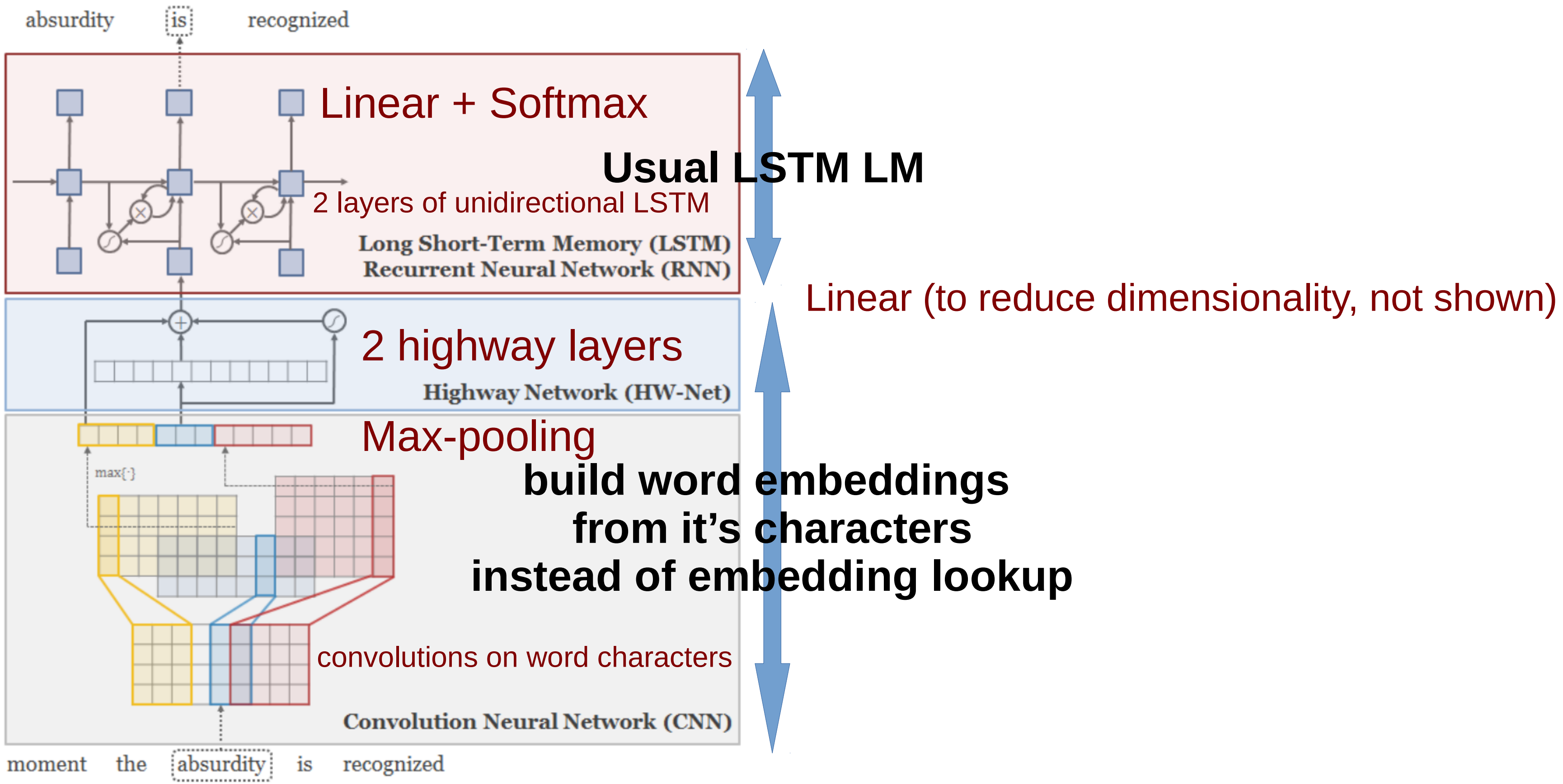
ELMo = embeddings from language models



* This is really TagLM, the predecessor of ELMo. There are technical differences, but idea is the same!

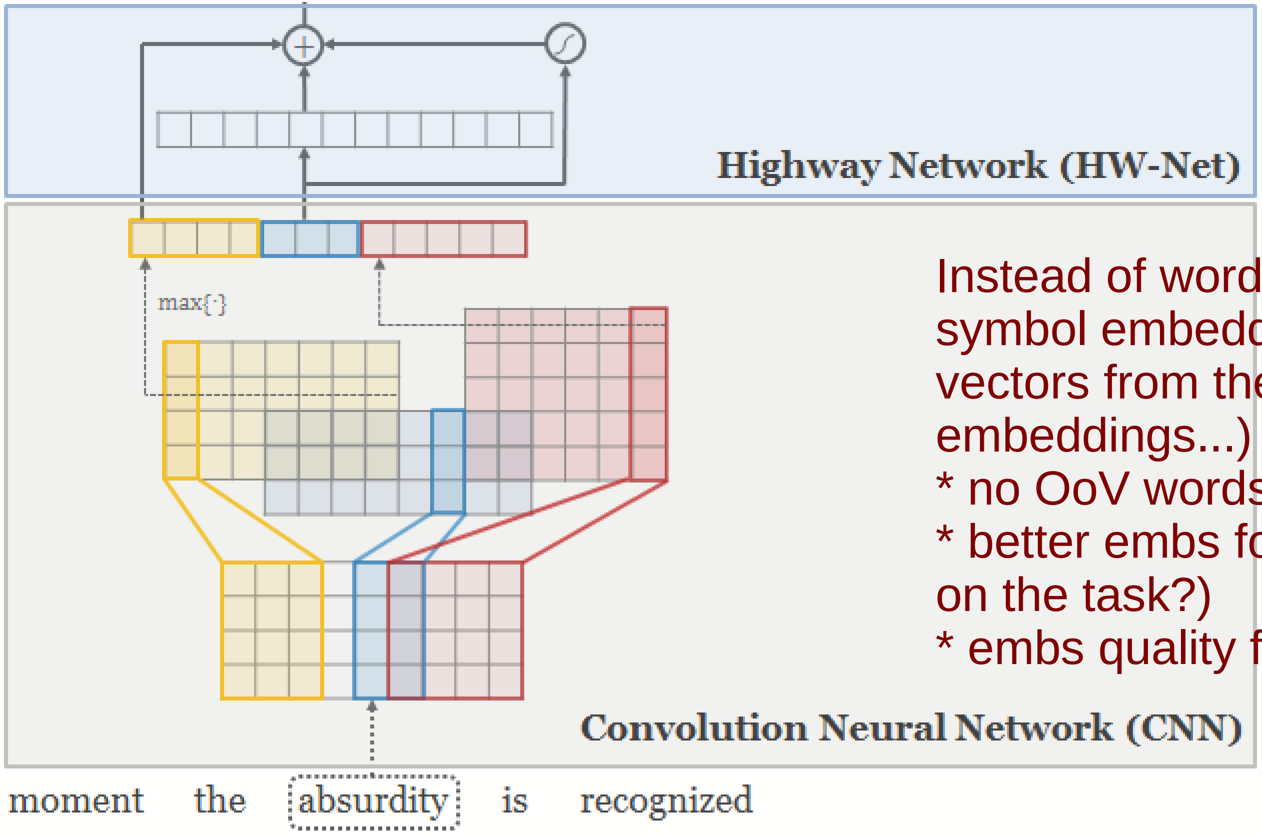
Pictures from https://en.wikipedia.org/wiki/Elmo and Peters et al. Semi-supervised sequence tagging with bidirectional language models, 2017

ELMo Language Models architecture



Kim et al. Character-Aware Neural Language Models, 2016 @ https://nlp.seas.harvard.edu/slides/aaai16.pdf

Character-level CNN (CharCNN)



Instead of word embeddings use symbol embeddings + NN to build word vectors from them (still named word embeddings...)
* no OoV words
* better embs for rare words (depends on the task?)
* embs quality for frequent words?

Kim et al. Character-Aware Neural Language Models, 2016 @ https://nlp.seas.harvard.edu/slides/aaai16.pdf

Character-level CNN (CharCNN)

a b s u r d i t y

Kim et al. Character-Aware Neural Language Models, 2016 @ https://nlp.seas.harvard.edu/slides/aaai16.pdf

Character-level CNN (CharCNN)

$C \in \mathbb{R}^{d \times l}$: Representation of *absurdity*

0.4	-0.8	2.2	0.1	0.5	-0.4	0.4	-0.4	0.1
0.1	1.2	1.5	-0.8	-1.5	0.2	0.1	1.2	0.7
0.2	0.1	-1.2	0.2	-0.2	0.3	0.2	-1.3	-0.1
-0.2	-0.5	0.1	0.2	-0.3	0.3	-0.1	1.0	-0.3

. . . .

a b s u r d i t y

Kim et al. Character-Aware Neural Language Models, 2016 @ https://nlp.seas.harvard.edu/slides/aaai16.pdf

Character-level CNN (CharCNN)

$H \in \mathbb{R}^{d \times w}$: Convolutional filter matrix of width w = 3

-0.1	0.5	2.2
0.7	0.9	0.3
-0.2	-0.2	0.7
1.3	-0.1	-1.1

0.4	-0.8	2.2	0.1	0.5	-0.4	0.4	-0.4	0.1
0.1	1.2	1.5	-0.8	-1.5	0.2	0.1	1.2	0.7
0.2	0.1	-1.2	0.2	-0.2	0.3	0.2	-1.3	-0.1
-0.2	-0.5	0.1	0.2	-0.3	0.3	-0.1	1.0	-0.3

. . . .

a b s u r d i t y

Kim et al. Character-Aware Neural Language Models, 2016 @ https://nlp.seas.harvard.edu/slides/aaai16.pdf

Character-level CNN (CharCNN)

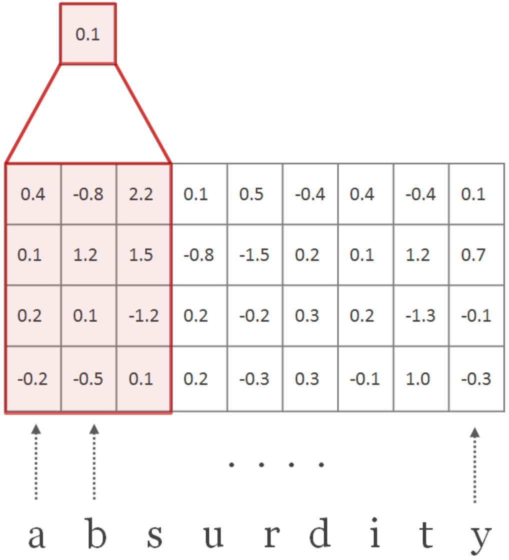
$$f[1] = \langle C[*, 1:3], H \rangle$$

0.4	-0.8	2.2	0.1	0.5	-0.4	0.4	-0.4	0.1
0.1	1.2	1.5	-0.8	-1.5	0.2	0.1	1.2	0.7
0.2	0.1	-1.2	0.2	-0.2	0.3	0.2	-1.3	-0.1
-0.2	-0.5	0.1	0.2	-0.3	0.3	-0.1	1.0	-0.3
a	b	s	u	r	d	i	t	y

Kim et al. Character-Aware Neural Language Models, 2016 @ https://nlp.seas.harvard.edu/slides/aaai16.pdf

Character-level CNN (CharCNN)

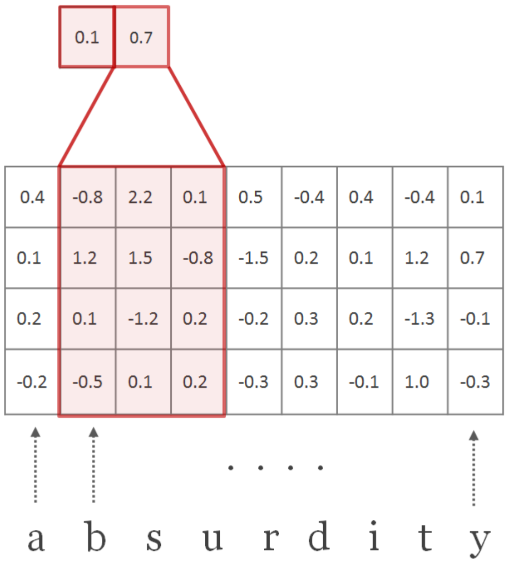
$$f[1] = \langle C[*, 1:3], H \rangle$$



Kim et al. Character-Aware Neural Language Models, 2016 @ https://nlp.seas.harvard.edu/slides/aaai16.pdf

Character-level CNN (CharCNN)

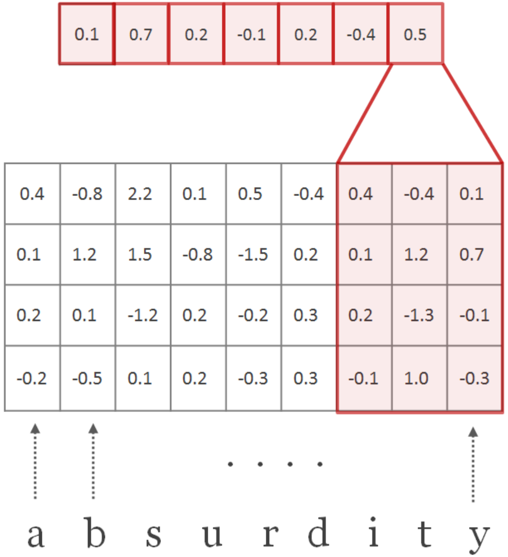
$$f[2] = \langle C[*, 2:4], H \rangle$$



Kim et al. Character-Aware Neural Language Models, 2016 @ https://nlp.seas.harvard.edu/slides/aaai16.pdf

Character-level CNN (CharCNN)

$$f[T-2] = \langle C[*, T-2:T], H \rangle$$



Kim et al. Character-Aware Neural Language Models, 2016 @ https://nlp.seas.harvard.edu/slides/aaai16.pdf

Character-level CNN (CharCNN)

$$y[1] = \max_i \{f[i]\}$$

0.1	0.7	0.2	-0.1	0.2	-0.4	0.5	→ 0.7

0.4	-0.8	2.2	0.1	0.5	-0.4	0.4	-0.4	0.1
0.1	1.2	1.5	-0.8	-1.5	0.2	0.1	1.2	0.7
0.2	0.1	-1.2	0.2	-0.2	0.3	0.2	-1.3	-0.1
-0.2	-0.5	0.1	0.2	-0.3	0.3	-0.1	1.0	-0.3

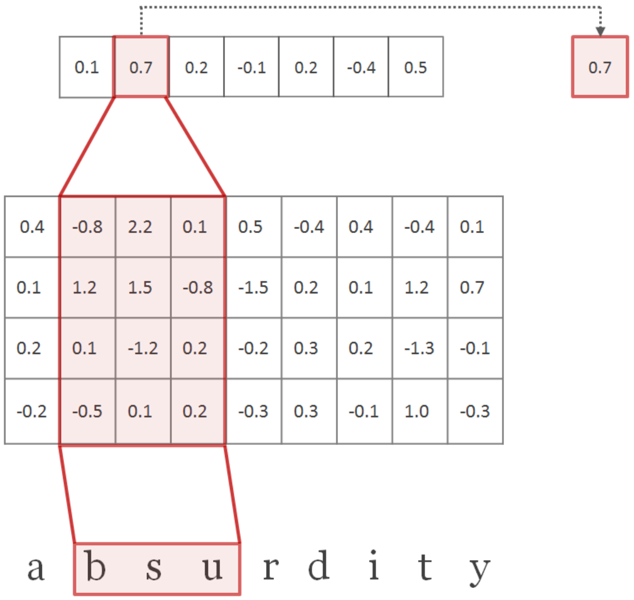
↑ ↑ . . . . ↑

a b s u r d i t y

Kim et al. Character-Aware Neural Language Models, 2016 @ https://nlp.seas.harvard.edu/slides/aaai16.pdf

Character-level CNN (CharCNN)

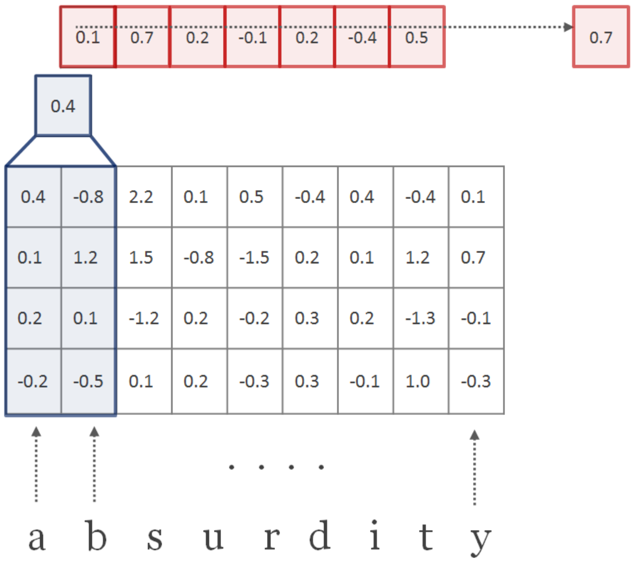
Each filter picks out a character n-gram



Kim et al. Character-Aware Neural Language Models, 2016 @ https://nlp.seas.harvard.edu/slides/aaai16.pdf

Character-level CNN (CharCNN)

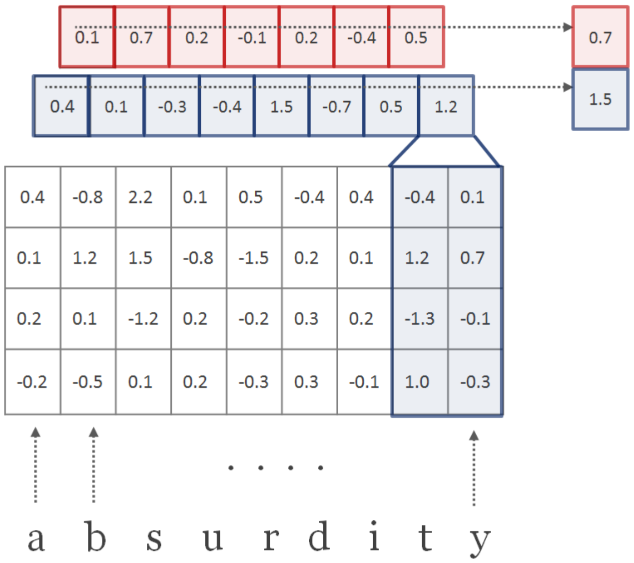
$$f'[1] = \langle C[*, 1:2], H' \rangle$$



Kim et al. Character-Aware Neural Language Models, 2016 @ https://nlp.seas.harvard.edu/slides/aaai16.pdf

Character-level CNN (CharCNN)

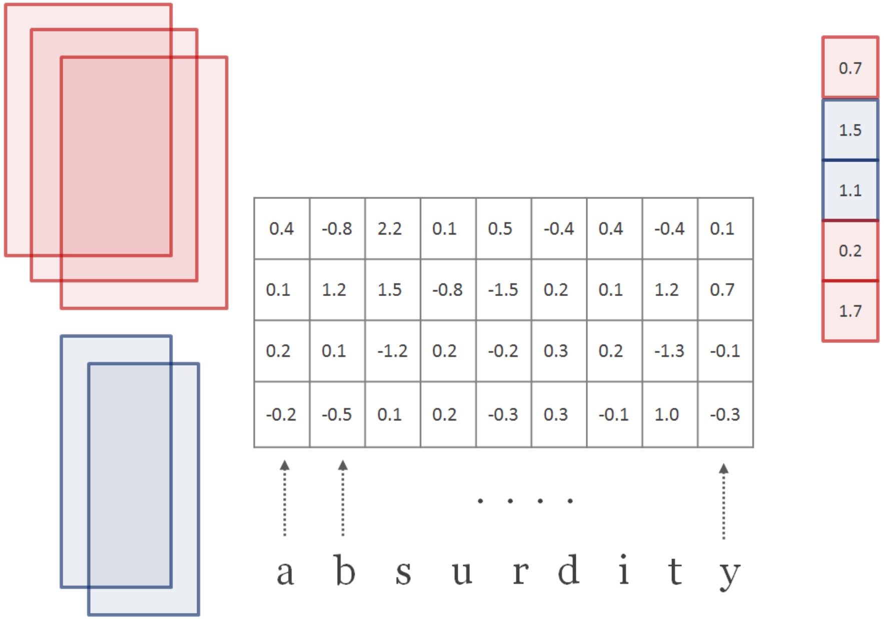
$$y[2] = \max_i \{f'[i]\}$$



Kim et al. Character-Aware Neural Language Models, 2016 @ https://nlp.seas.harvard.edu/slides/aaai16.pdf

Character-level CNN (CharCNN)

Many filter matrices (25–200) per width (1–7)



Kim et al. Character-Aware Neural Language Models, 2016 @ https://nlp.seas.harvard.edu/slides/aaai16.pdf

Character-level CNN (CharCNN)

Add bias, apply nonlinearity

$$\tanh\left(\begin{bmatrix}0.7\\1.5\\1.1\\0.2\\1.7\end{bmatrix} + b\right) = \begin{bmatrix}0.8\\1.0\\0.9\\0.5\\1.1\end{bmatrix}$$

CharCNN is slower, but convolution operations on GPU have been very optimized.

Kim et al. Character-Aware Neural Language Models, 2016 @ https://nlp.seas.harvard.edu/slides/aaai16.pdf

Character-level CNN (CharCNN)

$\mathbf{C} \in \mathbb{R}^{d \times l}$: Matrix representation of word (of length l)

$\mathbf{H} \in \mathbb{R}^{d \times w}$: Convolutional filter matrix

d : Dimensionality of character embeddings (e.g. 15)

w : Width of convolution filter (e.g. 1–7)

1. Apply a convolution between $\mathbf{C}$ and $\mathbf{H}$ to obtain a vector $\mathbf{f} \in \mathbb{R}^{l-w+1}$

$$\mathbf{f}[i] = \langle \mathbf{C}[*, i : i + w - 1], \mathbf{H} \rangle$$

where $\langle \mathbf{A}, \mathbf{B} \rangle = \mathrm{Tr}(\mathbf{A}\mathbf{B}^T)$ is the Frobenius inner product.

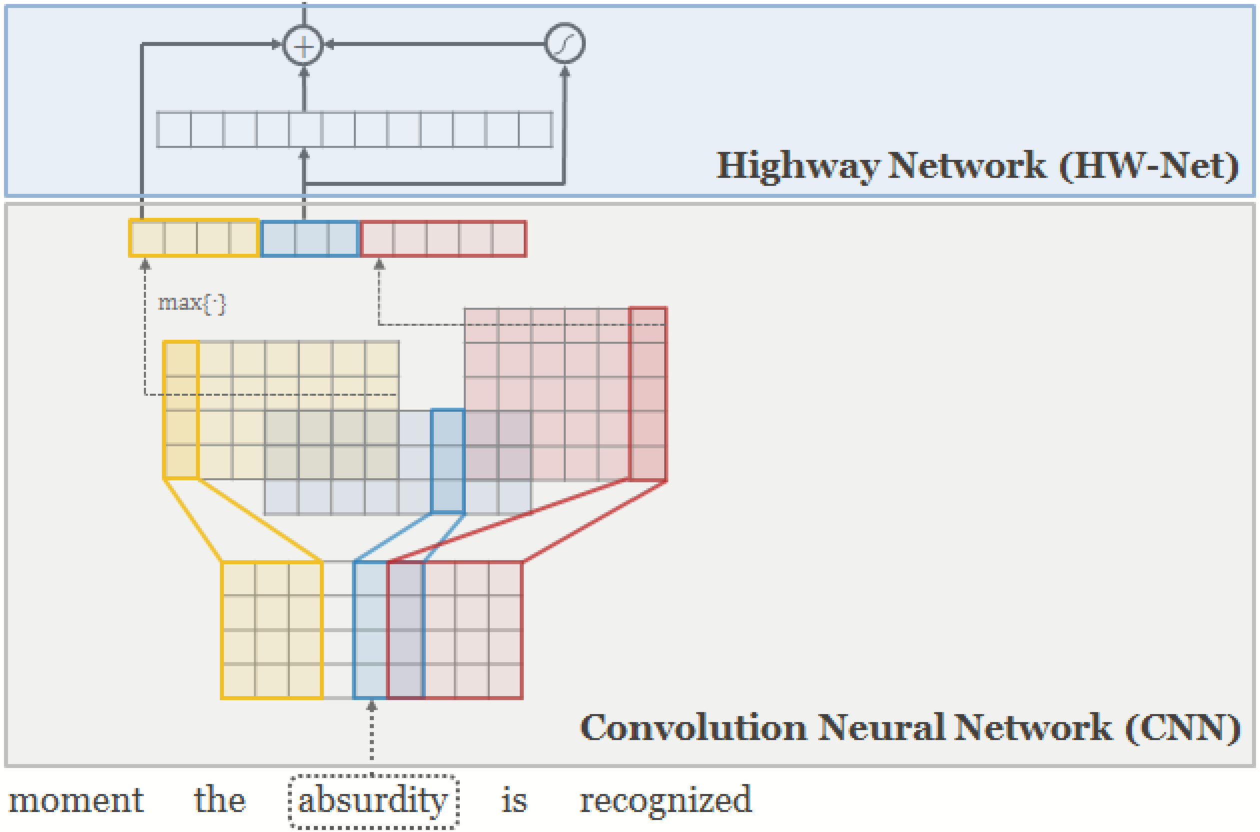
2. Take the *max-over-time* (with bias and nonlinearity)

$$y = \tanh(\max_i \{\mathbf{f}[i]\} + b)$$

as the feature corresponding to the filter $\mathbf{H}$ (for a particular word).

Kim et al. Character-Aware Neural Language Models, 2016 @ https://nlp.seas.harvard.edu/slides/aaai16.pdf

Character-level CNN (CharCNN)



Kim et al. Character-Aware Neural Language Models, 2016 @ https://nlp.seas.harvard.edu/slides/aaai16.pdf

Highway Network

$\mathbf{y}$: output from CharCNN

Multilayer Perceptron

$$\mathbf{z} = g(\mathbf{W}\mathbf{y} + \mathbf{b})$$

Highway Network

(Srivastava, Greff, and Schmidhuber 2015)

$$\mathbf{z} = \mathbf{t} \odot g(\mathbf{W}_H\mathbf{y} + \mathbf{b}_H) + (\mathbf{1} - \mathbf{t}) \odot \mathbf{y}$$

$\mathbf{W}_H, \mathbf{b}_H$: Affine transformation

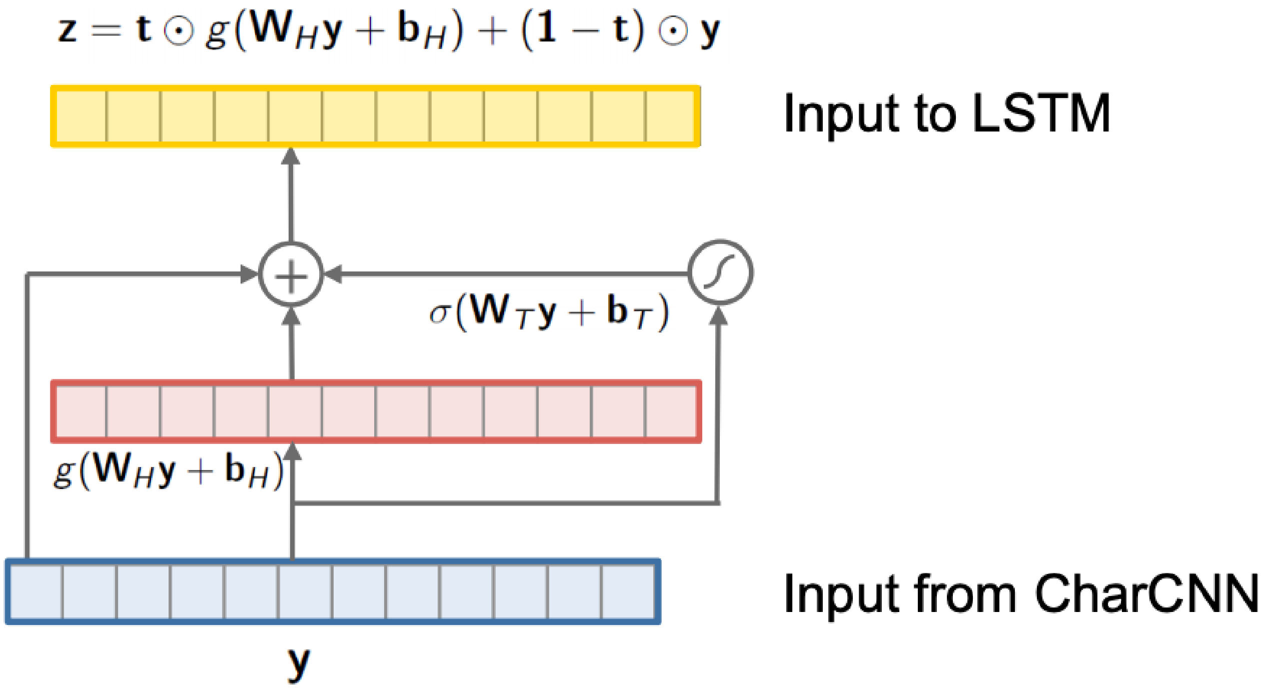
$\mathbf{t} = \sigma(\mathbf{W}_T\mathbf{y} + \mathbf{b}_T)$: *transform* gate

$\mathbf{1} - \mathbf{t}$: *carry* gate

Hierarchical, adaptive composition of character *n*-grams.

Kim et al. Character-Aware Neural Language Models, 2016 @ https://nlp.seas.harvard.edu/slides/aaai16.pdf

Highway Network



Kim et al. Character-Aware Neural Language Models, 2016 @ https://nlp.seas.harvard.edu/slides/aaai16.pdf

Highway Network

Perplexity on the Penn Treebank

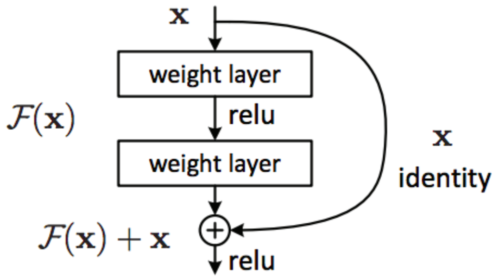
	LSTM-Char	
	Small	Large
No Highway Layers	100.3	84.6
One Highway Layer	92.3	79.7
Two Highway Layers	90.1	78.9
One MLP Layer	111.2	92.6

Table 7: Perplexity on the Penn Treebank for small/large models trained with/without highway layers.

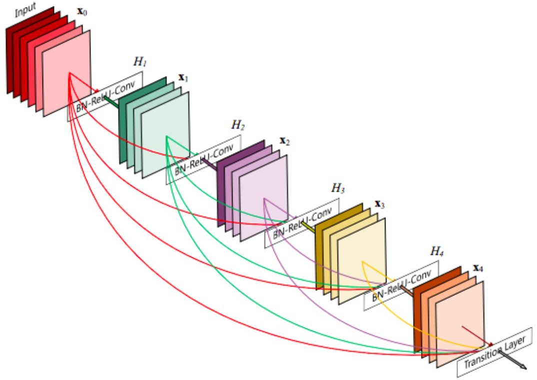
Kim et al. Character-Aware Neural Language Models, 2016 @ https://nlp.seas.harvard.edu/slides/aaai16.pdf

Help your gradients flow!

- In deep nets gradient can vanish/explode when it backpropagates (due to chain rule)
- To solve vanishing gradient problem:
  - Skip / residual connections
    - Helped building ResNet – 152-layer CNN winning ILSVRC 2015 (ImageNet classification competition)
  - Densely-connected networks
    - Each layer receives all previous layers' outputs concatenated, number of parameters quadratically grows with the number of layers
    - Used in DenseNet, QRNNs
  - LSTMs
  - Highway Networks



ResNet building block.
Figure from He et al. Deep residual learning for image recognition, 2015



DenseNet. Figure from Huang et al. Densely connected convolutional networks, 2017

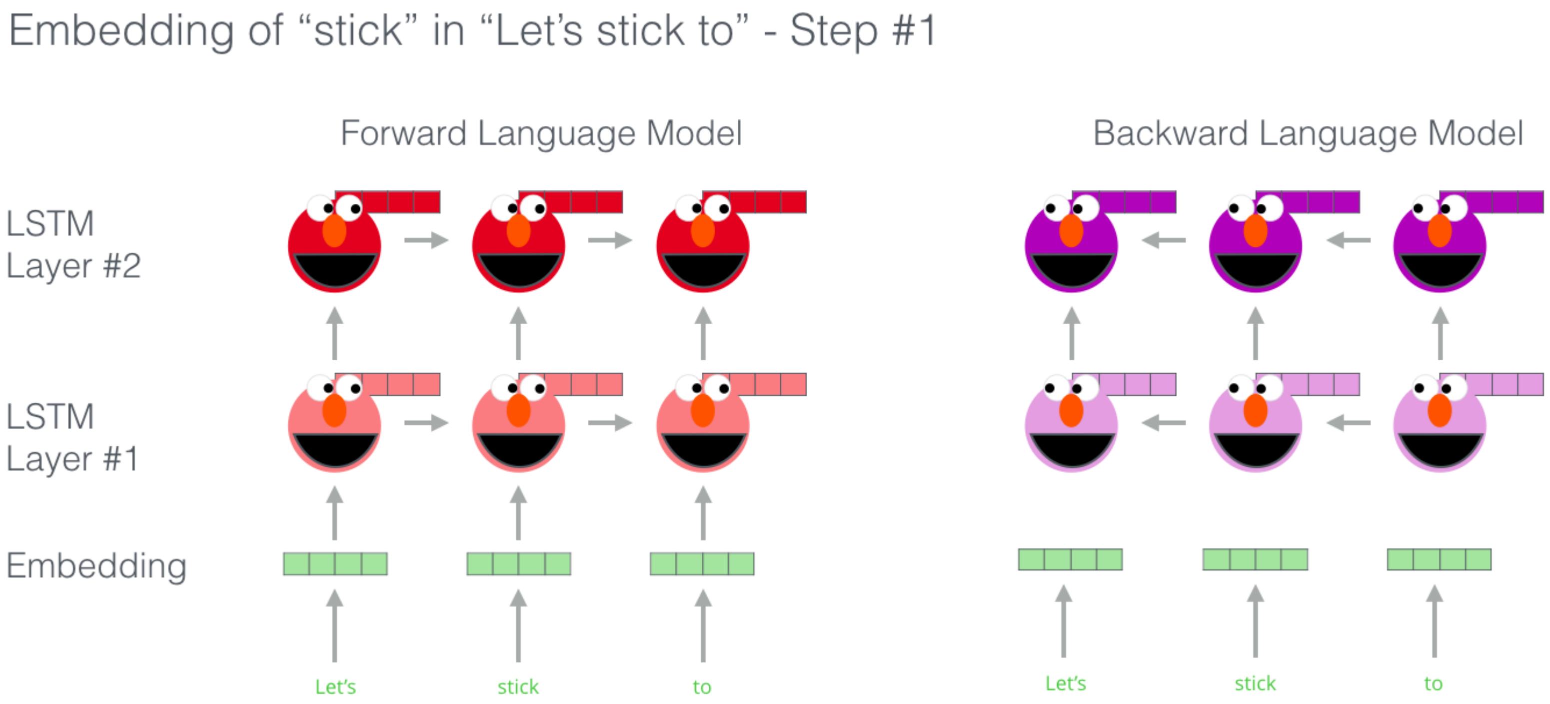
ELMo LM inputs

1. The context insensitive type representation:
    a. Char embedding dim - 16
    b. 2048 character n-gram convolutional filters (with a width from 1 to 7)
    c. Two highway layers (dim - 2048)
    d. Linear projection down to a 512 representation
2. These filters are used in CharCNN:

Width	1	2	3	4	5	6	7	Total
Num	32	32	64	128	256	512	1024	2048

Peters et al. Deep contextualized word representations, 2018 @ https://www.slideshare.net/shuntaroy/a-review-of-deep-contextualized-word-representations-peters-2018

ELMo LM



Peters et al. Deep contextualized word representations, 2018 @ http://jalammar.github.io/illustrated-bert/

ELMo LM

1. Bidirectional language model with two LSTM layers in each direction:
2. L = 2 LSTM layers for each direction:
    a. 4096 units
    b. 512 input_size
    c. 512 dimension projections
3. Residual connection from the first to second layer

Peters et al. Deep contextualized word representations, 2018 @ https://www.slideshare.net/shuntaroy/a-review-of-deep-contextualized-word-representations-peters-2018

LSTM with a projection layer

LSTM

$$i_t = \sigma\left(W_i x_t + U_i h_{t-1}\right)$$
$$f_t = \sigma\left(W_f x_t + U_f h_{t-1}\right)$$
$$o_t = \sigma\left(W_o x_t + U_o h_{t-1}\right)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tanh(W_c x_t + U_c h_{t-1})$$
$$h_t = o_t \circ \tanh(c_t)$$
$$W_f \in \mathbb{R}^{n_c \times n_i}, U_f \in \mathbb{R}^{n_c \times n_h}$$
$$W_i \in \mathbb{R}^{n_c \times n_i}, U_i \in \mathbb{R}^{n_c \times n_h}$$
$$W_o \in \mathbb{R}^{n_c \times n_i}, U_o \in \mathbb{R}^{n_c \times n_h}$$
$$W_c \in \mathbb{R}^{n_c \times n_i}, U_c \in \mathbb{R}^{n_c \times n_h}$$
$$n_i = 512, n_c = 4096, n_h = 4096$$

LSTM with projection

$$i_t = \sigma\left(W_i x_t + \underline{\boldsymbol{Q_i}} r_{t-1}\right)$$
$$f_t = \sigma\left(W_f x_t + \underline{\boldsymbol{Q_f}} r_{t-1}\right)$$
$$o_t = \sigma\left(W_o x_t + \underline{\boldsymbol{Q_o}} r_{t-1}\right)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tanh\left(W_c x_t + \underline{\boldsymbol{Q_c}} r_{t-1}\right)$$
$$h_t = o_t \circ \tanh(c_t)$$
$$\underline{\boldsymbol{r_t = W_{rh} h_t}}$$
$$W_f \in \mathbb{R}^{n_c \times n_i}, \underline{\boldsymbol{Q_f \in \mathbb{R}^{n_c \times n_r}}}$$
$$W_i \in \mathbb{R}^{n_c \times n_i}, \underline{\boldsymbol{Q_i \in \mathbb{R}^{n_c \times n_r}}}$$
$$W_o \in \mathbb{R}^{n_c \times n_i}, \underline{\boldsymbol{Q_o \in \mathbb{R}^{n_c \times n_r}}}$$
$$W_c \in \mathbb{R}^{n_c \times n_i}, \underline{\boldsymbol{Q_c \in \mathbb{R}^{n_c \times n_r}}}$$
$$\underline{\boldsymbol{W_{rh} \in \mathbb{R}^{n_r \times n_c}}}$$
$$n_i = 512, n_c = 4096, \underline{\boldsymbol{n_r = 512}}$$

LSTM with a projection layer

Comparing LSTM and LSTM with projection parameters

LSTM with projection has a separate linear projection layer after the LSTM layer

Parameters		Comment
LSTM	LSTM with projection	
4nc x (ni + nh)	4nc x (ni + nr) + nc x nr	
4*4096 x (512 + 4096) = 75.4M	4*4096 x (512 + 512) + 4096 x 512 = 18.8M	ELMo (nc=4096, nr=ni=512) LSTM / LSTMP = 4

Similar reduction in computational complexity

ELMo LM

- LMs share input charNN weights and output embeddings (softmax weights), only LSTM weights differ
- LMs are optimized jointly, by minimizing the loss:

$$\sum_{k=1}^{N} \Big(\log p(t_k \mid t_1, \ldots, t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) \\ + \log p(t_k \mid t_{k+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s) \Big)$$

Peters et al. Deep contextualized word representations, 2018 @ https://www.slideshare.net/shuntaroy/a-review-of-deep-contextualized-word-representations-peters-2018

ELMo LM training

1. Training 10 epochs on the 1B Word Benchmark
2. The average forward and backward perplexities is 39.7
3. Sampled softmax with num_samples = 8192
4. Optimizer - Adam
5. Gradient clipping by value - 1.0
6. ‘Noam’ learning rate schedule

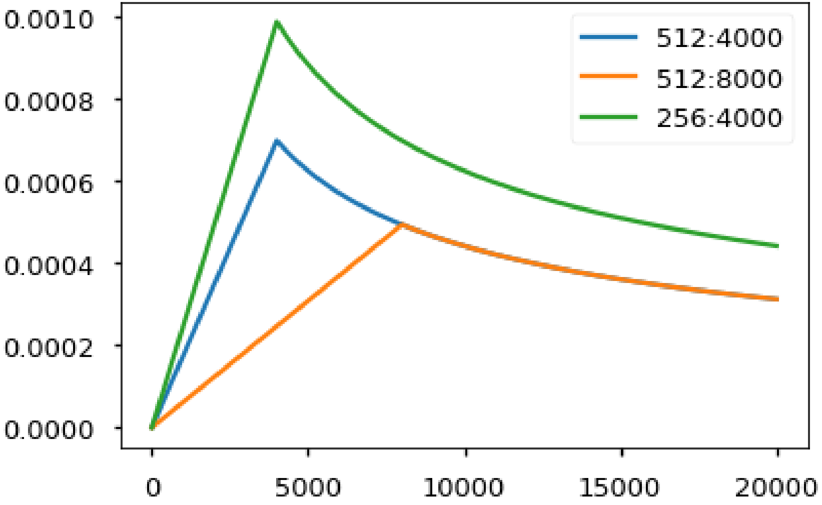
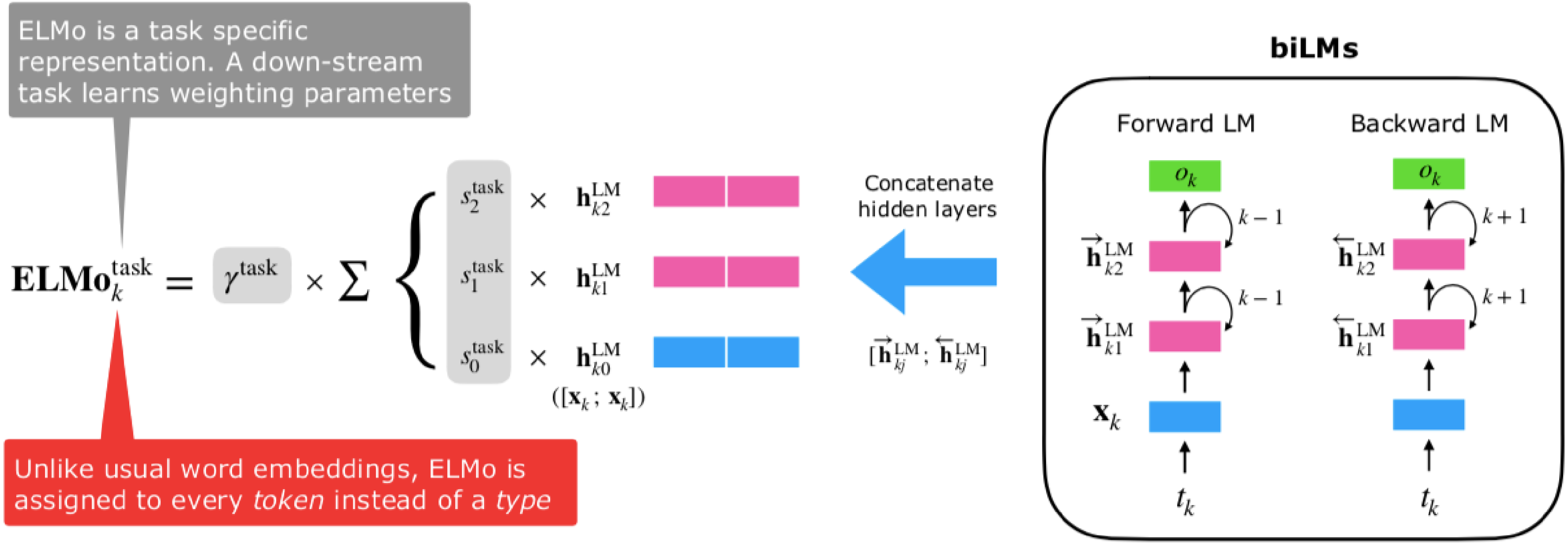


Figure from Vaswani et al. Attention Is All You Need, 2017 @ http://nlp.seas.harvard.edu/2018/04/03/attention.html

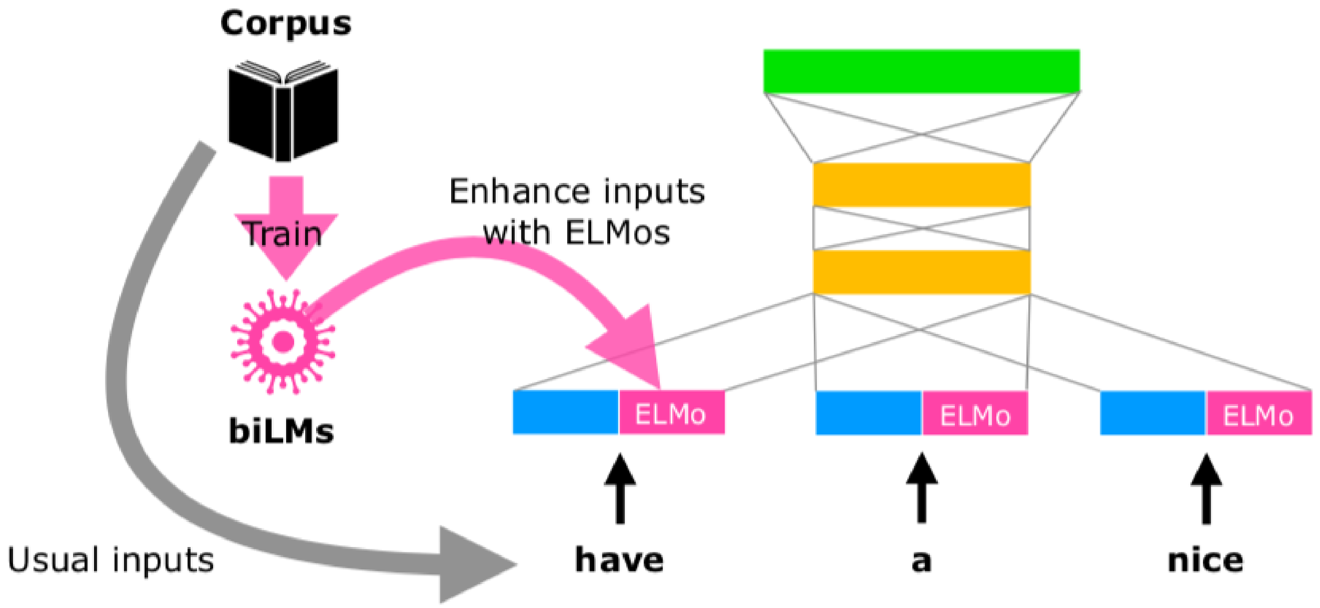
ELMo contextualized word embeddings



Peters et al. Deep contextualized word representations, 2018 @ https://www.slideshare.net/shuntaroy/a-review-of-deep-contextualized-word-representations-peters-2018

ELMo

ELMo can be integrated to almost all neural NLP tasks with simple concatenation to the embedding layer



Peters et al. Deep contextualized word representations, 2018 @ https://www.slideshare.net/shuntaroy/a-review-of-deep-contextualized-word-representations-peters-2018

ELMo results

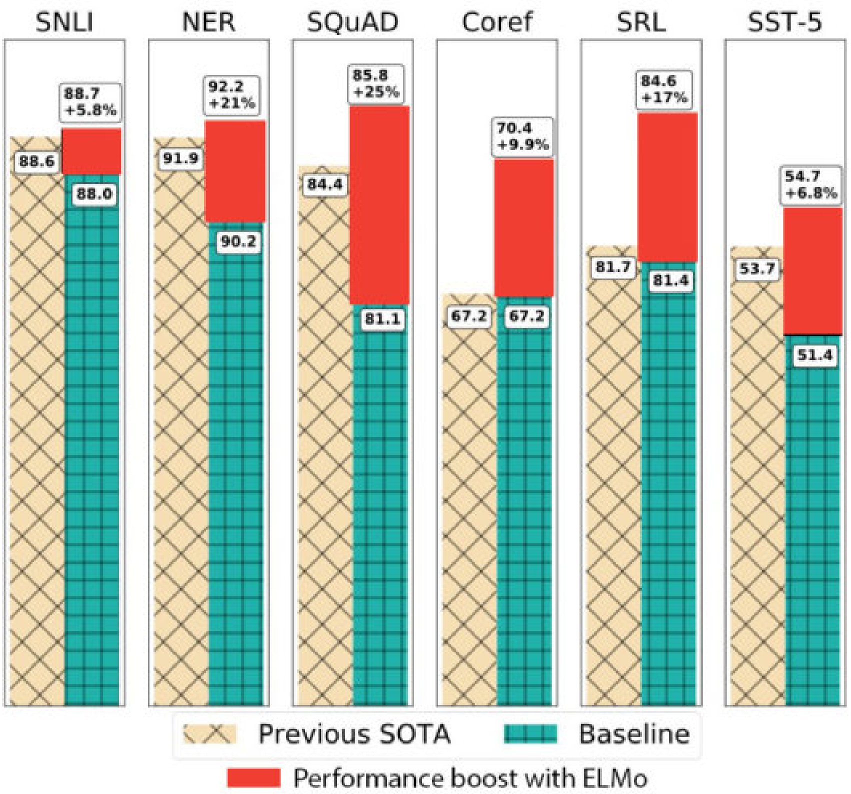
Many linguistic tasks are improved by using ELMo

	Task	Previous SOTA		Our baseline	ELMo + baseline	Increase (absolute/ relative)
Q&A	SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
Textual entailment	SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
Semantic role labelling	SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coreference resolution	Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
Named entity recognition	NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
Sentiment analysis	SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

Table 1: Test set comparison of ELMo enhanced neural models with state-of-the-art single model baselines across six benchmark NLP tasks. The performance metric varies across tasks – accuracy for SNLI and SST-5; F_1 for SQuAD, SRL and NER; average F_1 for Coref. Due to the small test sizes for NER and SST-5, we report the mean and standard deviation across five runs with different random seeds. The "increase" column lists both the absolute and relative improvements over our baseline.

Peters et al. Deep contextualized word representations, 2018 @ https://www.slideshare.net/shuntaroy/a-review-of-deep-contextualized-word-representations-peters-2018

ELMo results



Peters et al. Deep contextualized word representations, 2018 @ https://blogs.nvidia.com/blog/2018/08/27/nlp-deep-learning/

Which ELMo layers to include?

- Traditionally (in computer vision) only last layer's output is used as input to the classifier

Larger L2-penalty =>more uniform distribution

Task	Baseline	Last Only	All layers λ=1	All layers λ=0.001
SQuAD	80.8	84.7	85.0	**85.2**
SNLI	88.1	89.1	89.3	**89.5**
SRL	81.6	84.1	84.6	**84.8**

Average / weighted

Table 2: Development set performance for SQuAD, SNLI and SRL comparing using all layers of the biLM (with different choices of regularization strength λ) to just the top layer.

Peters et al. Deep contextualized word representations, 2018

Where to include ELMo?

- Concatenating ELMo embeddings to both first and last layer of the target model can improve results (for target models with attention on top?)

Task	Input Only	Input & Output	Output Only
SQuAD	85.1	**85.6**	84.8
SNLI	88.9	**89.5**	88.7
SRL	**84.7**	84.3	80.9

Table 3: Development set performance for SQuAD, SNLI and SRL when including ELMo at different locations in the supervised model.

Peters et al. Deep contextualized word representations, 2018

Analysis

The higher layer seemed to learn semantics while the lower layer probably captured syntactic features

Word sense disambiguation

Model	F_1
WordNet 1st Sense Baseline	65.9
Raganato et al. (2017a)	69.9
Iacobacci et al. (2016)	**70.1**
CoVe, First Layer	59.4
CoVe, Second Layer	*64.7*
biLM, First layer	67.4
biLM, Second layer	*69.0*

Table 5: All-words fine grained WSD F_1. For CoVe and the biLM, we report scores for both the first and second layer biLSTMs.

PoS tagging

Model	Acc.
Collobert et al. (2011)	97.3
Ma and Hovy (2016)	97.6
Ling et al. (2015)	**97.8**
CoVe, First Layer	*93.3*
CoVe, Second Layer	92.8
biLM, First Layer	*97.3*
biLM, Second Layer	96.8

Table 6: Test set POS tagging accuracies for PTB. For CoVe and the biLM, we report scores for both the first and second layer biLSTMs.

Peters et al. Deep contextualized word representations, 2018 @ https://www.slideshare.net/shuntaroy/a-review-of-deep-contextualized-word-representations-peters-2018

Analysis

The higher layer seemed to learn semantics while the lower layer probably captured syntactic features

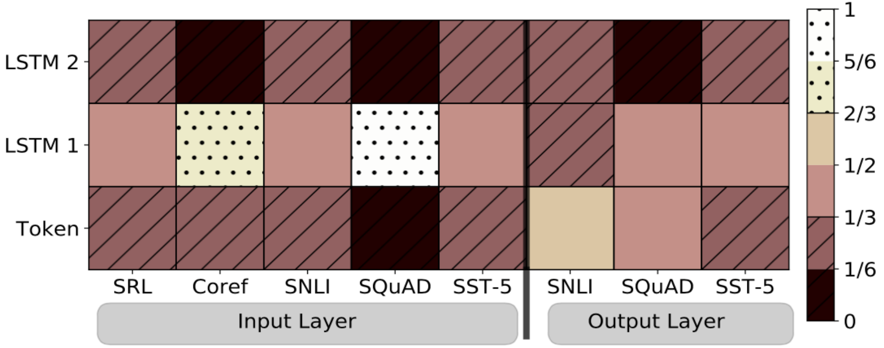


Figure 2: Visualization of softmax normalized biLM layer weights across tasks and ELMo locations. Normalized weights less then 1/3 are hatched with horizontal lines and those greater then 2/3 are speckled.

Peters et al. Deep contextualized word representations, 2018 @ https://www.slideshare.net/shuntaroy/a-review-of-deep-contextualized-word-representations-peters-2018

Analysis

The higher layer seemed to learn semantics while the lower layer probably captured syntactic features

Most models preferred "syntactic (probably)" features

Even in sentiment analysis

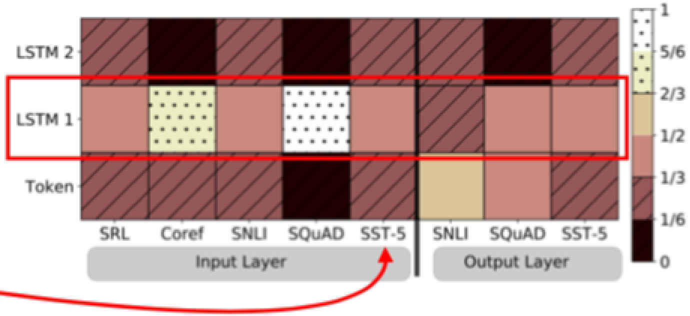


Figure 2: Visualization of softmax normalized biLM layer weights across tasks and ELMo locations. Normalized weights less then 1/3 are hatched with horizontal lines and those greater then 2/3 are speckled.

Peters et al. Deep contextualized word representations, 2018 @ https://www.slideshare.net/shuntaroy/a-review-of-deep-contextualized-word-representations-peters-2018

Fine tuning biLM

Fine tuning the biLM on task specific data typically resulted in significant drops in perplexity and sometimes improves target metric (88.9%→89.5% for SNLI, but not for sentiment classification).

Dataset		Before tuning	After tuning
SNLI		72.1	16.8
CoNLL 2012 (coref/SRL)		92.3	-
CoNLL 2003 (NER)		103.2	46.3
SQuAD	Context	99.1	43.5
	Questions	158.2	52.0
SST		131.5	78.6

Table 7: Development set perplexity before and after fine tuning for one epoch on the training set for various datasets (lower is better). Reported values are the average of the forward and backward perplexities.

Peters et al. Deep contextualized word representations, 2018 @ https://www.slideshare.net/shuntaroy/a-review-of-deep-contextualized-word-representations-peters-2018

ELMo overview

- Propose a new type of deep contextualised word representations (**ELMo**) that model:
  - Complex characteristics of word use (e.g., syntax and semantics)
  - How these uses vary across linguistic contexts (i.e., to model polysemy)
- Show that ELMo can improve existing neural models in various NLP tasks
- Argue that ELMo can capture more abstract linguistic characteristics in the higher level of layers

Peters et al. Deep contextualized word representations, 2018 @ https://www.slideshare.net/shuntaroy/a-review-of-deep-contextualized-word-representations-peters-2018

ELMo References

Presentations:

1. Character-Aware Neural Language Models: https://nlp.seas.harvard.edu/slides/aaai16.pdf
2. Deep contextualized word representations: https://www.slideshare.net/shuntaroy/a-review-of-deep-contextualized-word-representations-peters-2018

Papers:

3. Kim et al. Character-Aware Neural Language Models, 2015
4. Jozefowicz et al. Exploring the Limits of Language Modeling, 2016
5. Peters et al. Semi-supervised sequence tagging with bidirectional language models, 2017
6. Peters et al. Deep contextualized word representations, 2018

ELMo code: https://github.com/allenai/allennlp/blob/master/allennlp/modules/elmo.py

ULMFiT = Universal LM Fine-Tuning

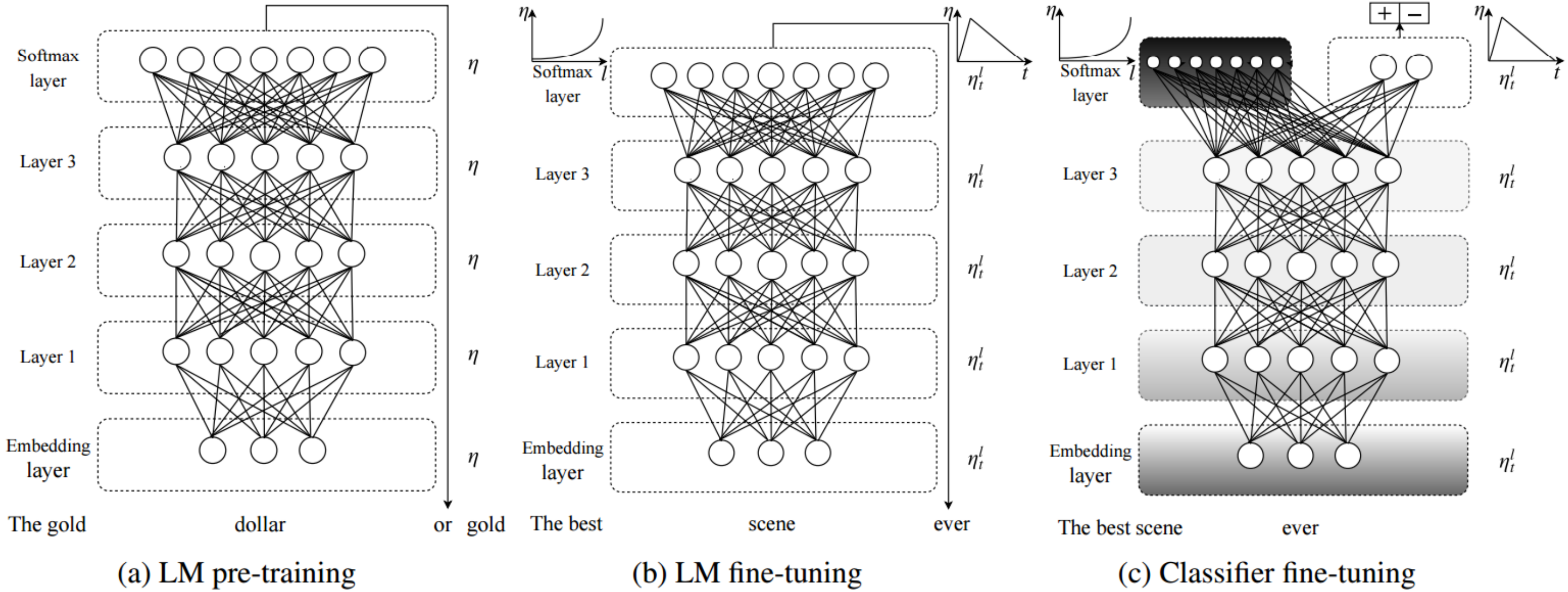


Figure 1: ULMFiT consists of three stages: a) The LM is trained on a general-domain corpus to capture general features of the language in different layers. b) The full LM is fine-tuned on target task data using discriminative fine-tuning (‘*Discr*’) and slanted triangular learning rates (STLR) to learn task-specific features. c) The classifier is fine-tuned on the target task using gradual unfreezing, ‘*Discr*’, and STLR to preserve low-level representations and adapt high-level ones (shaded: unfreezing stages; black: frozen).

Figure from Howard&Ruder. Universal Language Model Fine-tuning for Text Classification, 2018

Fine-tuning Tricks

- Discriminative fine-tuning = discriminative learning rates
  - Smaller learning rates for lower layers: $lr^{(l-1)} = lr^{(l)} / 2.6$
  - Lower layers capture more common features which need less adaptation to the task / domain
- Slanted triangular learning rates
  - lr warmup + lr decay
  - remember Noam
- Gradual unfreezing
  - Used for classifier finetuning only

    1. train only classifier weights (randomly initialized)
    2. classifier weights and last LM layer
    3. classifier weights and two last LM layers

    ...

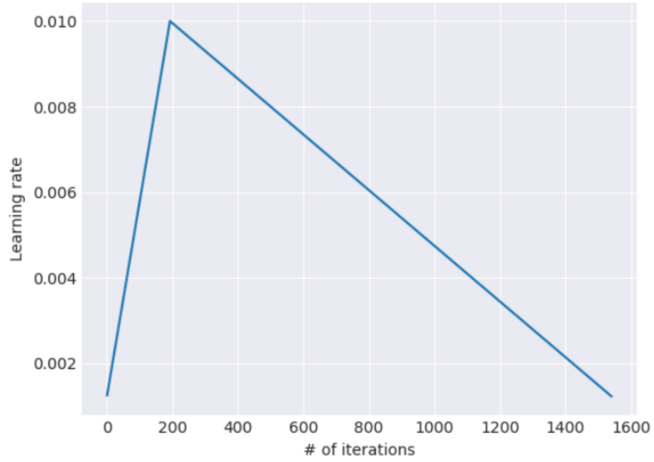


Figure 2: The slanted triangular learning rate schedule used for ULMFiT as a function of the number of training iterations.

Figure from Howard&Ruder. Universal Language Model Fine-tuning for Text Classification, 2018

More Tricks

- Bidirectional LM helps (5.3 → 4.58 ERR on IMDB)
- Classifier is FFNN:
  - 1 hidden layer of size 50
  - **[h_T , maxpool(H), meanpool(H)]** as input
- BPTT for Text Classification = BPT3C
  - Look at code to understand …

Figure from Howard&Ruder. Universal Language Model Fine-tuning for Text Classification, 2018

ULMFiT results

	Model	Test		Model	Test
IMDb	CoVe (McCann et al., 2017)	8.2	TREC-6	CoVe (McCann et al., 2017)	4.2
	oh-LSTM (Johnson and Zhang, 2016)	5.9		TBCNN (Mou et al., 2015)	4.0
	Virtual (Miyato et al., 2016)	5.9		LSTM-CNN (Zhou et al., 2016)	3.9
	ULMFiT (ours)	**4.6**		ULMFiT (ours)	**3.6**

Table 2: Test error rates (%) on two text classification datasets used by McCann et al. (2017).

	AG	DBpedia	Yelp-bi	Yelp-full
Char-level CNN (Zhang et al., 2015)	9.51	1.55	4.88	37.95
CNN (Johnson and Zhang, 2016)	6.57	0.84	2.90	32.39
DPCNN (Johnson and Zhang, 2017)	6.87	0.88	2.64	30.58
ULMFiT (ours)	**5.01**	**0.80**	**2.16**	**29.98**

Table 3: Test error rates (%) on text classification datasets used by Johnson and Zhang (2017).

Figure from Howard&Ruder. Universal Language Model Fine-tuning for Text Classification, 2018

ULMFiT results

- From scratch: No LM pretraining
- Supervised: LM pretraining on WikiText-103 + LM finetuning on labeled data
- Semi-supervised: LM pretraining WikitText-103 + LM finetuning on all task data

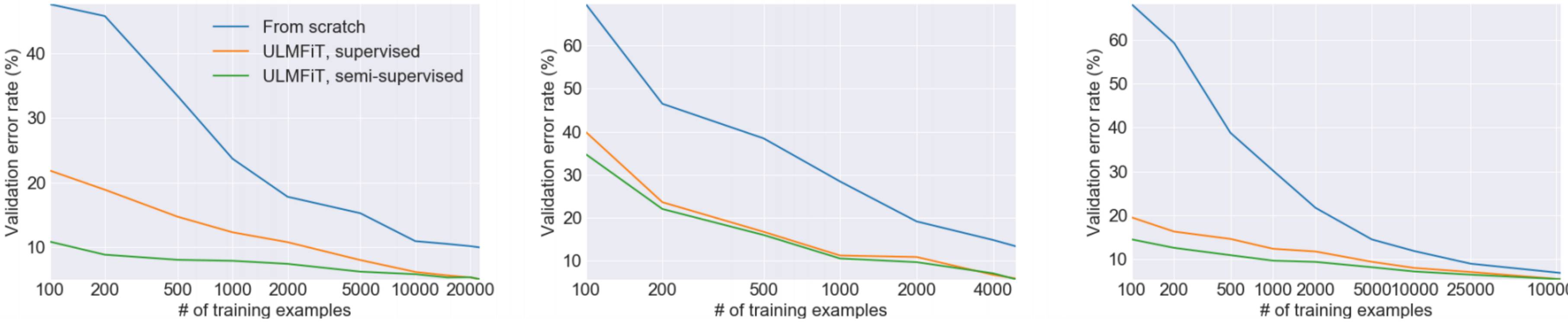


Figure 3: Validation error rates for supervised and semi-supervised ULMFiT vs. training from scratch with different numbers of training examples on IMDb, TREC-6, and AG (from left to right).

Figure from Howard&Ruder. Universal Language Model Fine-tuning for Text Classification, 2018

ULMFiT results

- For larger datasets (IMDB) we can skip pretraining on WikiText-103 and train LM only on target task data

Pretraining	IMDb	TREC-6	AG
Without pretraining	5.63	10.67	5.52
With pretraining	**5.00**	**5.69**	**5.38**

Table 4: Validation error rates for ULMFiT with and without pretraining.

Figure from Howard&Ruder. Universal Language Model Fine-tuning for Text Classification, 2018

ULMFiT results

- For larger datasets (IMDB) we can skip pretraining on WikiText-103 and train LM only on target task data

LM fine-tuning	IMDb	TREC-6	AG
No LM fine-tuning	6.99	6.38	6.09
Full	5.86	6.54	5.61
Full + discr	5.55	6.36	5.47
Full + discr + stlr	**5.00**	**5.69**	**5.38**

Table 6: Validation error rates for ULMFiT with different variations of LM fine-tuning.

Classifier fine-tuning	IMDb	TREC-6	AG
From scratch	9.93	13.36	6.81
Full	6.87	6.86	5.81
Full + discr	5.57	6.21	5.62
Last	6.49	16.09	8.38
Chain-thaw	5.39	6.71	5.90
Freez	6.37	6.86	5.81
Freez + discr	5.39	5.86	6.04
Freez + stlr	5.04	6.02	5.35
Freez + cos	5.70	6.38	**5.29**
Freez + discr + stlr	**5.00**	**5.69**	5.38

Table 7: Validation error rates for ULMFiT with different methods to fine-tune the classifier.

Figure from Howard&Ruder. Universal Language Model Fine-tuning for Text Classification, 2018

ULMFiT results

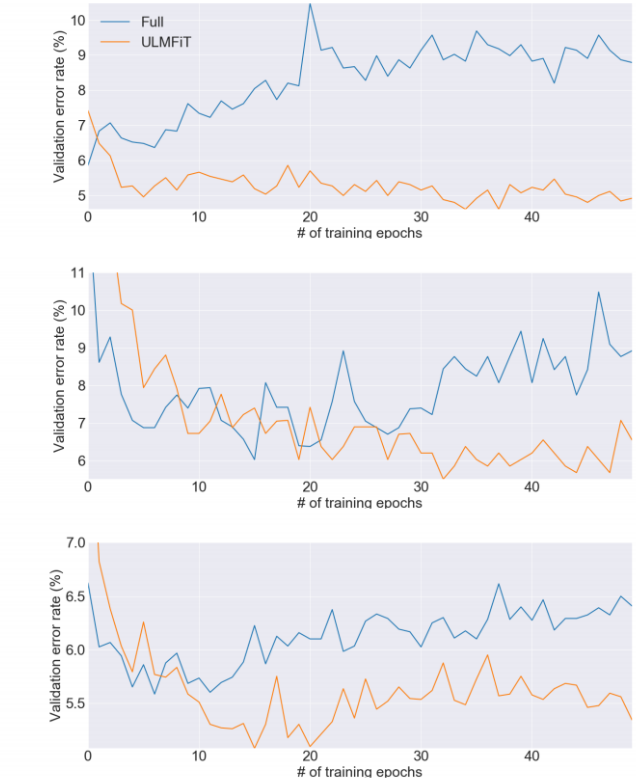


Figure 4: Validation error rate curves for fine-tuning the classifier with ULMFiT and '*Full*' on IMDb, TREC-6, and AG (top to bottom).

Figure from Howard&Ruder. Universal Language Model Fine-tuning for Text Classification, 2018