Efficient deep learning using richer context, with applications to *handwriting recognition* and *scholarly document quality prediction*

Gideon Maillette de Buy Wenniger

Thomas van Dongen
Eleri Aedmaa
Herbert Teun Kruitbosch
Edwin A. Valentijn
Lambert Schomaker

Andy Way

university of groningen

Open Universiteit
www.ou.nl

Date: 6-7-2021

Research Map

Family

Ranjita

Anika

My position within AI and research themes

- Theme 1: More context for predictive models.
- Theme 2: Efficient data-structures & Algorithms.
- Theme 3: Application to business relevant real-world problems.

PART 0: Deep Learning Basics

Data, model and training

- **Data**: collection of [Input,Label] pairs [x,y]
- **Model**: A nonlinear function x -> y*,
  - Implemented by multi-layer network

    with weights

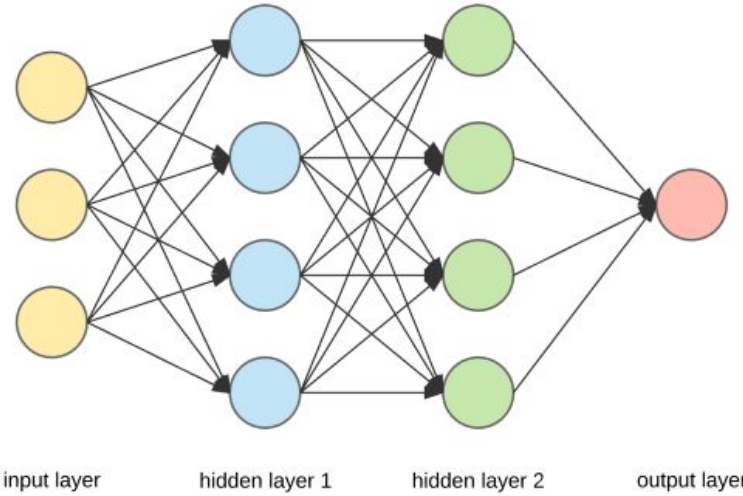


Image Source:
https://towardsdatascience.com/everything-you-need-to-know-about-neural-networks-and-backpropagation-machine-learning-made-easy-e5285bc2be3a

- **Training (**for batches b of examples [x_b,y_b]):
  - Feed model examples x_b, returns prediction y_b*
  - Compare: y_b*, y_b => loss (loss=0 when equal)
  - Compute gradient of loss with respect to network weights
  - Apply loss gradients: change weights to decrease loss

PART 1: Scholarly Document Quality Prediction

Based on:

Gideon Maillette de Buy Wenniger, Thomas van Dongen, Eleri Aedmaa, Herbert Teun Kruitbosch, Edwin A. Valentijn and Lambert Schomaker. 2020. **Structure-Tags Improve Text Classification for Scholarly Document Quality Prediction.** First Workshop on Scholarly Document Processing (SDP 2020), at EMNLP 2020. pages 158--167. https://aclanthology.org/2020.sdp-1.18/

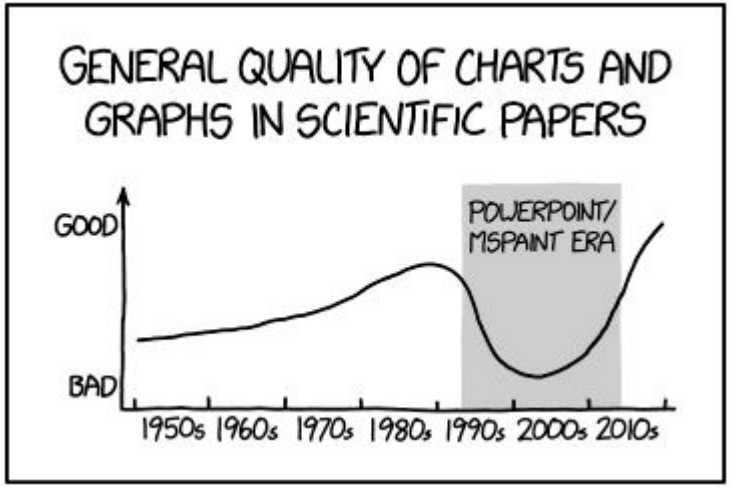
Thomas van Dongen, Gideon Maillette de Buy Wenniger and Lambert Schomaker. 2020. **SChuBERT: Scholarly Document Chunks with BERT-encoding boost Citation Count Prediction.** First Workshop on Scholarly Document Processing (SDP 2020), at EMNLP 2020. pages 148--157. https://aclanthology.org/2020.sdp-1.17/

Scholarly Document Quality Prediction

- Predict quality from the document alone

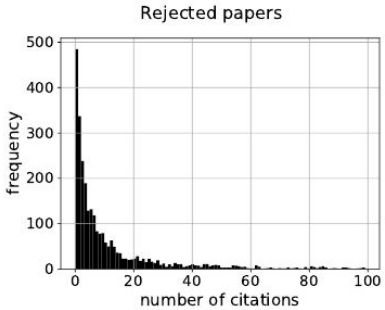
- What indicators of quality to predict?
  - Accept/Reject
    - Simple and well understood
    - Scarce data

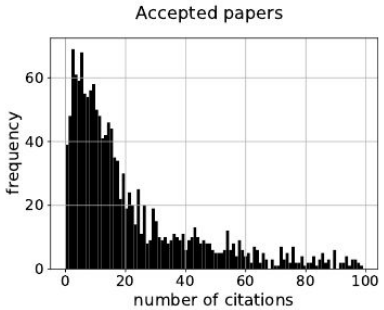
  - Number of Citations
    - Large data availability



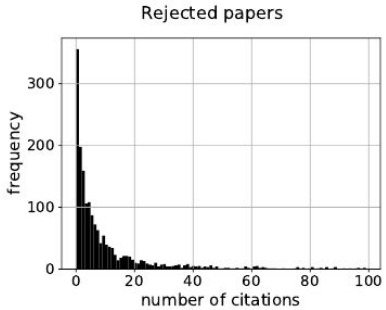
Source:https://m.xkcd.com/1945/

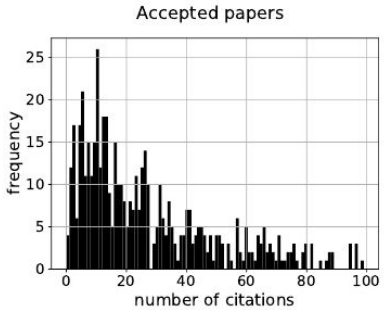
Motivation: Correlation paper acceptance and number of citations





(a) Machine Learning domain.





(b) Computation and Language domain.

Domain	Average number of citations		Spearman rank-order correlation
	rejected articles	accepted articles	coefficient (ρ), p-value
Machine Learning	24.0 ± 127.3	61.0 ± 232.6	$0.375, 5 \times 10^{-153}$
Computation and Language	14.8 ± 44.3	59.0 ± 105.9	$0.466, 1.6 \times 10^{-128}$

(c) Global statistics.

Word embeddings

Vocabulary:
Man, woman, boy, girl, prince, princess, queen, king, monarch

	1	2	3	4	5	6	7	8	9
man	1	0	0	0	0	0	0	0	0
woman	0	1	0	0	0	0	0	0	0
boy	0	0	1	0	0	0	0	0	0
girl	0	0	0	1	0	0	0	0	0
prince	0	0	0	0	1	0	0	0	0
princess	0	0	0	0	0	1	0	0	0
queen	0	0	0	0	0	0	1	0	0
king	0	0	0	0	0	0	0	1	0
monarch	0	0	0	0	0	0	0	0	1

Each word gets a 1x9 vector representation

Try to build a lower dimensional embedding

Vocabulary:
Man, woman, boy, girl, prince, princess, queen, king, monarch

	Femininity	Youth	Royalty
Man	0	0	0
Woman	1	0	0
Boy	0	1	0
Girl	1	1	0
Prince	0	1	1
Princess	1	1	1
Queen	1	0	1
King	0	0	1
Monarch	0.5	0.5	1

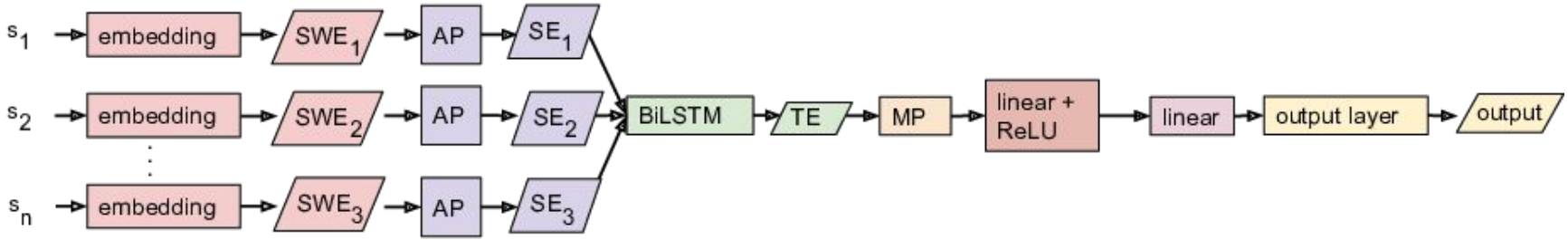
Each word gets a 1x3 vector

Similar words... similar vectors

@shane_a_lynn | @TeamEdgeTier

Image Source:
https://www.r-craft.org/r-news/get-busy-with-word-embeddings-an-introduction/

Methods: BiLSTM-based model (Shen et al., 2019)

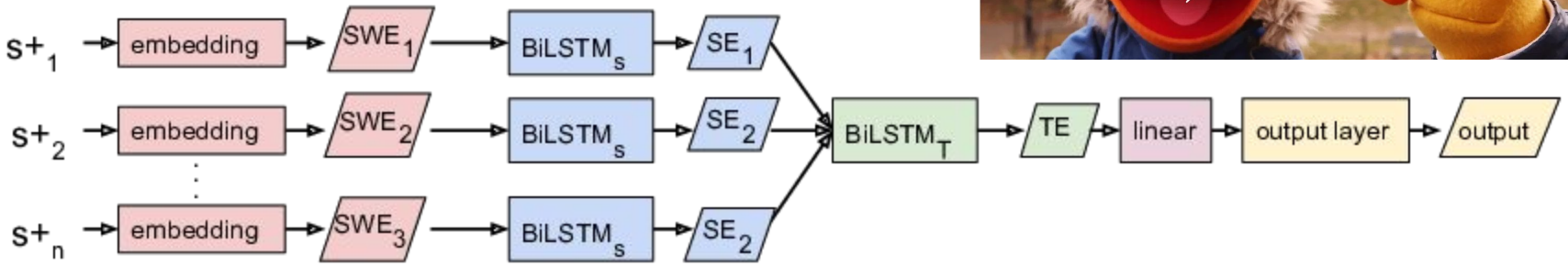


S = Input (sentence segmented text, one-hot encoded)
SWE = Sentence words embedding
AP = Average-pooling
SE = Sentence embedding

BiLSTM = Bidirectional LSTM
TE = Text embedding
MP = Maxpooling Layer
output layer = Softmax or Leaky Relu

linear + ReLU = Rectified Linear Hidden Layer

Methods: HAN-based predictive model



Legend

S+ = Input (sentence segmented text with structure tags, one-hot encoded)

SWE = Sentence words embedding

$BiLSTM_s$ = Sentence level bidirectional-LSTM with attention

SE = Sentence embedding

$BiLSTM_T$ = Text level bidirectional-LSTM

TE = Text embedding

output layer = Softmax or Leaky ReLu

Methods: structure tags

- Tags added at begin and end every sentence
- Indicate the origin in the text structure:
  - Title, Abstract, Body_Text
- Similar to principle of "**command-string**" in: "Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation" (Johnson et. al, 2016)

<*TITLE*>Cross-Task Knowledge-Constrained Self Training
</*TITLE*>
<*ABSTRACT*> Abstract </*ABSTRACT*>
<*ABSTRACT*> We present an algorithmic framework for learning multiple related tasks. </*ABSTRACT*>
...
<*BODY_TEXT*> 1 Introduction </*BODY_TEXT*>
<*BODY_TEXT*> When two NLP systems are run on the same data, we expect certain constraints to hold between their outputs. </*BODY_TEXT*>

Experiments

- PeerRead accept/reject prediction
  - Standard benchmark
  - 3-domains: AI, ML, CL
  - Small datasets: 5.0K (ML), 2.6K, 4.1K (AI)
  - Imbalanced

- Number of citations prediction:
  - Predict **log(Number of citations + 1)**
  - 88K examples

PeerRead accept/reject prediction

Table 1: Data sizes and division between the ratio of accepted and rejected papers for the arXiv subsets

	training		validation		testing		total
	num	acc:rej	num	acc:rej	num	acc:rej	
machine learning	4543	36.4% : 63.6%	252	36.5% : 63.5%	253	32.0% : 68.0%	5048
computation & language	2374	24.3% : 75.7%	132	22.0% : 78.0%	132	31.1% : 68.9%	2638
artificial intelligence	3682	10.5% : 89.5%	205	8.3% : 91.7%	205	7.8% : 92.2%	4092

- **PeerRead accept/reject prediction: small and unbalanced datasets**
  - **But still the standard, for lack of good alternatives**

- **Accept/Reject labels are heuristically defined:**
  - **based on publication at top conferences or not**

Model performance on PeerRead

Table 6: PeerRead accept/reject prediction accuracy and *AUC* (area under ROC curve) scores for our models.

arXiv sub-domain dataset	metric	Majority class prediction	Average Word Embeddings	BiLSTM (re-implemented)	HAN	HAN_{ST}
artificial intelligence	accuracy	92.2%	74.1 ± 0.49%	**92.4 ± 1.02%**	88.9 ± 1.97 %	89.6 ± 1.02%
	AUC	0.50	**0.793 ± 0.0143**	**0.711 ± 0.0771**	0.625 ± 0.042	0.705 ± 0.055
computation & language	accuracy	68.9%	73.7 ± 0.87%	80.1 ± 1.91%	80.3 ± 2.00%	**81.8 ± 1.91%**
	AUC	0.50	0.740 ± 0.010	0.744 ± 0.056	0.712 ± 0.029	**0.745 ± 0.011**
machine learning	accuracy	67.9%	72.9 ± 0.60%	**79.6 ± 3.19%**	76.7 ± 2.77%	78.7 ± 0.69%
	AUC	0.50	0.662 ± 0.003	0.743 ± 0.025	0.743 ± 0.019	**0.758 ± 0.0149**

HAN_{ST} : comparison to state of the art

Table 5: PeerRead accept/reject prediction accuracy: comparison of HAN_{ST} against state-of-the-art.

arXiv sub-domain dataset	Majority class prediction	Benchmark (Kang et al., 2018)	BiLSTM (Shen et al., 2019)	Joint (Shen et al., 2019)	HAN_{ST}
artificial intelligence	92.2%	92.6%	91.5 ± 1.03%	**93.4 ± 1.07%**	89.6 ± 1.02%
computation & language	68.9%	75.7%	76.2 ± 1.30%	77.1 ± 3.10%	**81.8 ± 1.91%**
machine learning	68.0%	70.7% +	**81.1 ± 0.83%**	79.9 ± 2.54%	78.7 ± 0.69%

Results Summary

Both tasks:

- HAN_{ST} outperforms HAN in all experiments

Accept/Reject prediction PeerRead:

- HAN_{ST} best on CL domain
- BiLSTM and joint (textual+visual) model best on other two domains

Number of citation prediction:

- HAN_{ST} outperforms other models on number of citation prediction

Conclusion

- Presented Hierarchical Attention Networks with structure tags HAN_{ST}
  - Outperforms HAN and best on number of citations prediction

- Number of citations prediction:
  - Predict **Log(#Citations + 1)**, large data available, strong correlation with accept/reject

PART 1B: Saliency Maps for Scholarly Document Quality Prediction

Motivation and setting

- **Goal:** obtain a heatmap of what inputs are most important for a prediction

- **Setting**: Input consists of word embeddings or full-sentence embeddings (when working with BERT)

Data

- **PeerRead data: accept/reject prediction**
  - Small dataset: beneficial for faster development
  - Accept/Reject labels: binary prediction easier to work with than regression labels, for saliency map producing methods
  - https://github.com/allenai/PeerRead

- **Stanford Sentiment Treebank**
  - Easy to interpret, in terms of saliency at the word level
  - Reported benchmark scores
  - Small dataset:
    - 5-class prediction: train (8544), dev (1101) and test splits (2210)
    - Binary (positive/negative): train (**6920**), dev (**872**), test (**1821**) sentences
  - https://nlp.stanford.edu/sentiment/

Selected Literature

- **Gradient-Based Attribution Methods**
  - Marco Ancona, Enea Ceolini, Cengiz Öztireli, Markus Gross
  - Chapter 9 “Explaining AI: Interpreting, Explaining and Visualizing Deep Learning

- **TOWARDS BETTER UNDERSTANDING OF GRADIENT-BASED ATTRIBUTION METHODS FOR DEEP NEURAL NETWORKS**
  - Marco Ancona, Enea Ceolini, Cengiz Öztireli, Markus Gross
  - https://arxiv.org/pdf/1711.06104.pdf
- **Visualizing and Understanding Neural Models in NLP**
  - Jiwei Li, Xinlei Chen, Eduard Hovy and Dan Jurafsky
  - https://www.cs.cmu.edu/~./hovy/papers/16HLT-visualizing-NNs.pdf

Explored approach

- Baseline approach: saliency as **Gradient X Input**
- Why?
  - Easy to implement
  - Easy to understand
  - **Passes Sanity checks for Saliency Maps**
    - Sanity Checks for Saliency Maps
    - **Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, Been Kim**
    - https://papers.nips.cc/paper/2018/file/294a8ed24b1ad22ec2e7efea049b8737-Paper.pdf

- Later: explore different approaches:
  - Integrated Gradients
  - Layer-wise Relevance Propagation
  - Other

Gradient-based methods: multiplying with the input

- Sensitivity methods: how does the output changes when one or more of the inputs is changes?
  - i.e. how to get more cat features in the input?

- Saliency methods: effect of feature on the output for same input with feature removed
  - i.e. how to explain prediction for this input?

- Multiplying gradient with input yields a saliency method:
  $R_i(x) = x_i \times \delta y_c(x) / \Delta x_i$

Source: M. Ancona, E. Ceolini, A.C. Öztireli and M.H. Gross. Gradient-Based Attribution Methods. Explainable AI: Interpreting, Explaining and Visualizing Deep Learning. https://link.springer.com/chapter/10.1007/978-3-030-28954-6 13

Benchmark results original paper

Model	Fine-grained		Positive/Negative	
	All	Root	All	Root
NB	67.2	41.0	82.6	81.8
SVM	64.3	40.7	84.6	79.4
BiNB	71.0	41.9	82.7	83.1
VecAvg	73.3	32.7	85.1	80.1
RNN	79.0	43.2	86.1	82.4
MV-RNN	78.7	44.4	86.8	82.9
RNTN	**80.7**	**45.7**	**87.6**	**85.4**

Table 1: Accuracy for fine grained (5-class) and binary predictions at the sentence level (root) and for all nodes.

Reproducing the benchmark scores

- Using a slightly different model (not RNN):
  - BiLSTM+MaxPooling+Linear+ReLu
  - Model worked well on PeerRead Dataset :(Shen et.a, 2010) baseline)
- number of examples: 1821

	Average Word Embedding	BiLSTM	BiLSTM+MaxPooling+Linear+ReLu
Accuracy	77.2%	**79.8%**	79.2%
AUC Score	.772	**0.798**	0.792

Still not at the benchmark score level, but getting there

Stanford Sentiment Treebank Saliency: an impression

307	words:	@start@	a	sour	,	nasty	offering	.	@end@
308	saliency_scores_average_	-4.55E-11	-6.18E-11	2.01E-10	-7.17E-11	4.02E-10	-6.55E-11	1.16E-11	-5.83E-12
309	saliency_scores_l2_norm_	7.43E-11	1.47E-10	1.60E-10	5.38E-11	2.08E-10	7.29E-11	1.74E-11	3.88E-11
310	Summary_label	example_numbe	total_saliency_s	prediction	actual_label	correct			
311	<<<Example_Summary>>	1	3.64E-10	**0**	**0**	TRUE			
312									
313	words:	@start@	a	waste	of	good	performances	.	@end@
314	saliency_scores_average_	-0.00096389418	0.000875569065	0.006339158863	0.000939710938	-0.00233269855	-0.00274255219	-0.00047290811	-0.00053284555
315	saliency_scores_l2_norm_	0.000834381440	0.00131617696	0.003468679963	0.001130487188	0.001936187502	0.001603858662	0.000519084162	0.000524401373
316	Summary_label	example_numbe	total_saliency_s	prediction	actual_label	correct			
317	<<<Example_Summary>>	15	0.001109540259	0	0	TRUE			
318									
319	words:	@start@	a	wildly	funny	prison	caper	.	@end@
320	saliency_scores_average_	2.62E-08	-2.37E-08	-1.54E-08	1.17E-07	-3.46E-08	-5.78E-09	-3.69E-09	1.84E-08
321	saliency_scores_l2_norm_	2.39E-08	3.50E-08	6.36E-08	7.63E-08	2.79E-08	1.66E-08	6.03E-09	1.86E-08
322	Summary_label	example_numbe	total_saliency_s	prediction	actual_label	correct			
323	<<<Example_Summary>>	14	7.88E-08	1	1	TRUE			
324									
325	words:	@start@	alas	,	it	s	neither	.	@end@
326	saliency_scores_average_	-1.02E-05	8.16E-06	-3.01E-05	-1.59E-06	-7.98E-06	5.46E-05	8.06E-06	-1.11E-05
327	saliency_scores_l2_norm_	3.26E-05	4.28E-05	2.09E-05	1.94E-05	2.49E-05	2.93E-05	1.10E-05	2.15E-05
328	Summary_label	example_numbe	total_saliency_s	prediction	actual_label	correct			
329	<<<Example_Summary>>	4	9.81E-06	0	0	TRUE			

PeerRead Saliency: architecture impression

- The same technique can be applied to multiple-sentence documents
- Quality of saliency predictions is work in progress...

0	@title_start@ bayesian models of data streams with hierarchical power priors @title_end@	-2.54E-08
1	@abstract_start@ abstract making inferences from data streams is a pervasive problem in many modern data analysis applications . @abstract_end@	2.08E-05
2	@abstract_start@ but it requires to address the problem of continuous model updating , and adapt to changes or @@UNKNOWN@@ in the underlying data generating dis	3.27E-05
3	@abstract_start@ in this paper , we approach these problems from a bayesian perspective covering general conjugate exponential models . @abstract_end@	2.96E-07
4	@abstract_start@ our proposal makes use of non - conjugate hierarchical priors to explicitly model temporal changes of the model parameters . @abstract_end@	0.0001937837951
5	@abstract_start@ we also derive a novel variational inference scheme which overcomes the use of non - conjugate priors while maintaining the computational efficiency of	-1.46E-05
6	@abstract_start@ the approach is validated on three real data sets over three latent variable models . @abstract_end@	0.000185916826
7	@body_text_start@ 1 . @body_text_end@	-5.09E-05
8	@body_text_start@ introduction flexible and computationally efficient models for streaming data are required in many machine learning applications , and in this paper we p	1.92E-05
9	@body_text_start@ specifically , we are interested in models suitable for domains that exhibit changes in the underlying generative process (@@UNKNOWN@@ et al . ,	0.0001789636444
10	@body_text_start@ we @@UNKNOWN@@ a situation , where one receives batches of data at discrete points in time . @body_text_end@	0.0001455553138
11	@body_text_start@ as each new batch arrives , we want to @@UNKNOWN@@ information from the new data , while also retaining relevant information from the historica	0.0001939011709
12	@body_text_start@ our modelling is inspired by previous works on bayesian recursive estimation (@@UNKNOWN@@ et al . , 2013 ; @@UNKNOWN@@ , 2014) , powe	-0.0005119945272
13	@body_text_start@ correspondence to : @body_text_end@	-3.49E-05
14	@body_text_start@ @@UNKNOWN@@ @@UNKNOWN@@ @body_text_end@	-9.94E-05

PART 1C: SDP with full text: BERT-based models

SChuBERT

Scholarly Document Chunks with BERT-encoding boost Citation Count Prediction

Thomas van Dongen,

Gideon Maillette de Buy Wenniger,

Lambert Schomaker

(Based on slides by Thomas van Dongen)

Open Universiteit
www.ou.nl

Introduction

Scholarly document quality prediction
Content-based citation prediction

In this paper we show the benefits of:

- More (and better) data: ACL-BiblioMetry dataset
- Pre-trained language models: SChuBERT model

ACL-BiblioMetry dataset

Title + abstract + full text information
Citations and log citations labels
All papers scraped from ACL (CL + NLP papers)

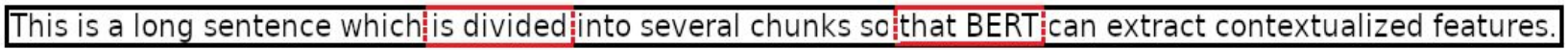
	Train	Test	Validation
Number of papers	27853	1549	1548

BERT: brief overview

State-of-the-art language model released by Google in 2018.
Several pre-trained models have been released.
Can be used out-of-the-box to generate contextualized embeddings for sentences.

Issues with BERT

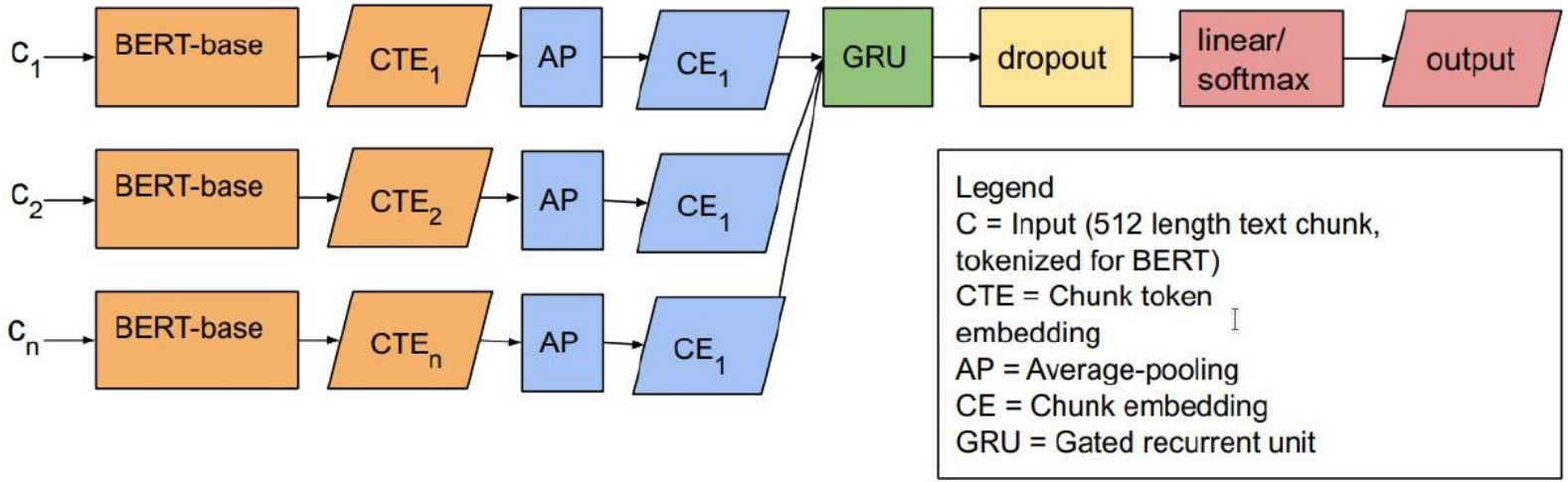
BERT has a time complexity that is quadratic with respect to the input length.
Max tokens for BERT-base: 512
Average tokens in our dataset: around 20000
Solution: chunking



Chunk 1: This is a long sentence which is divided
Chunk 2: is divided into several chunks so that BERT
Chunk 3: that BERT can extract contextualized features

Methods

Model architecture:



Experiments

Longer vs. shorter input texts:
- Abstract only vs. full text
- All chunks vs. a portion of chunks

Larger vs. smaller training data set:
- 100% vs. 50% vs. 10% of training data

SChuBERT is compared to two state-of-the-art models: HAN and a BiLSTM

Main Results

Table 4: Results on the full data and with full input.

	BiLSTM	HAN	SChuBERT (5 chunk)	SChuBERT (6 chunk)	SChuBERT
R^2 score	0.319 ± 0.013	0.339 ± 0.013	0.369 ± 0.009	0.380 ± 0.004	$\mathbf{0.398 \pm 0.006}$
MSE	1.110 ± 0.021	1.080 ± 0.021	1.032 ± 0.015	1.013 ± 0.006	$\mathbf{0.985 \pm 0.010}$
MAE	0.824 ± 0.009	0.820 ± 0.009	0.805 ± 0.005	0.798 ± 0.005	$\mathbf{0.789 \pm 0.005}$

Results: Does full input text help?

Table 5: Results on the full data and with abstract text only.

	BiLSTM	HAN	SChuBERT
R^2 score	0.158 ± 0.006	0.248 ± 0.014	$\mathbf{0.249 \pm 0.002}$
MSE	1.377 ± 0.010	$\mathbf{1.230 \pm 0.023}$	$\mathbf{1.230 \pm 0.004}$
MAE	0.933 ± 0.002	0.885 ± 0.008	$\mathbf{0.884 \pm 0.002}$

Results: Does more training data help?

Table 6: Results for SChuBERT on a subset of the data and with full input.

	SChuBERT 50% data	SChuBERT 10% data
R^2 score	0.327 ± 0.007	0.205 ± 0.026
MSE	1.058 ± 0.011	1.473 ± 0.048
MAE	0.809 ± 0.005	0.923 ± 0.027

Other advantages

SChuBERT training is much faster:

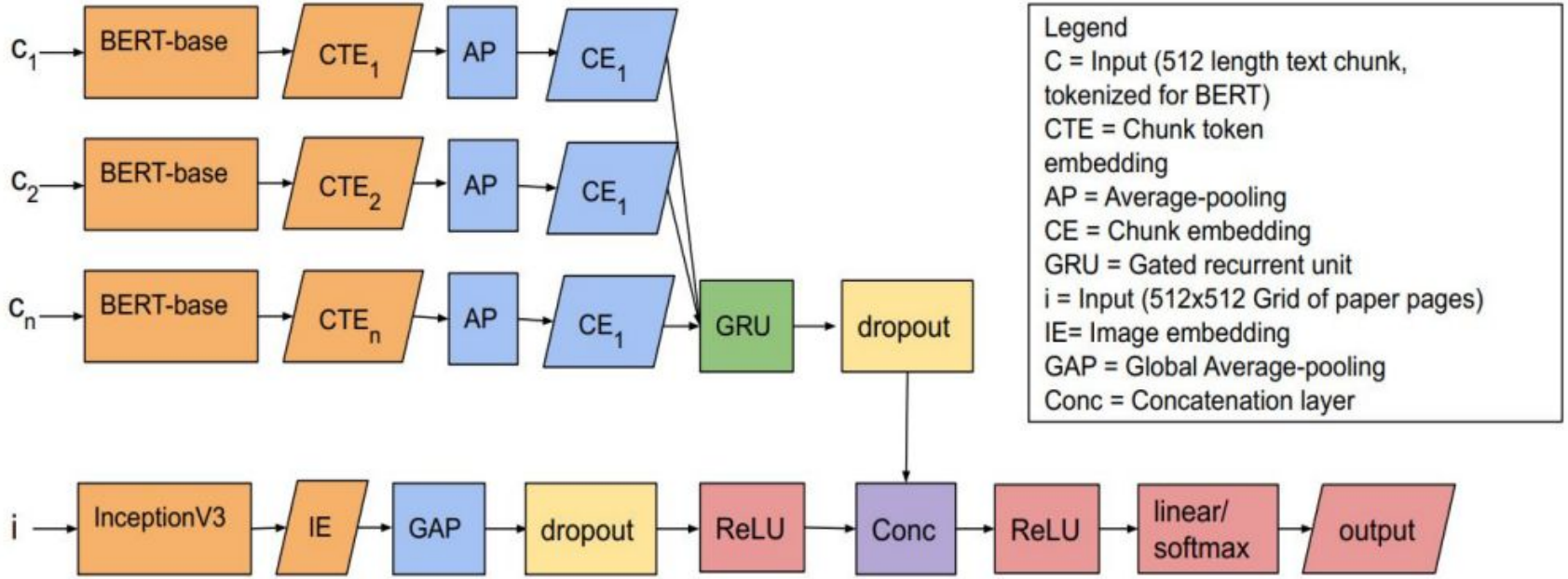
	BiLSTM	HAN	SchuBERT
Time in seconds	1048	1921	12

SchuBERT has far less (trainable) parameters:

Hidden size	BiLSTM	HAN	SchuBERT
192	1170949	2059525	N/A
256	N/A	N/A	788225
512	N/A	N/A	969665

SChuBERT converges much faster:
40 epochs vs ~100 epochs

Beyond SChuBERT$_{Joint}$: Multi-modal prediction



Joint Results

Accept/reject prediction:

Dataset	BiLSTM	HANst	JOINT	SChuBERT	SChuBERT$_{JOINT}$
AI	91.5 ± 1.03%	89.6 ± 1.02%	**93.4 ± 1.07%**	92.4 ± 0.84%	93.3 ± 0.70%
CL	76.2 ± 1.30%	81.8 ± 1.91%	77.1 ± 3.10%	80.8 ± 2.60%	**83.0 ± 2.67%**
LG	81.1 ± 0.83%	78.7 ± 0.69%	79.9 ± 2.54%	80.2 ± 1.39%	**82.8 ± 2.86%**

	Majority	BiLSTM	HAN$_{ST}$	JOINT	SChuBERT$_{JOINT}$
Average	76.3%	82.9 ± 1.05%	83.5 ± 1.20%	83.5 ± 2.24%	**86.4 ± 2.08%**

Citation prediction:

	SChuBERT	SChuBERT$_{JOINT}$
R2	0.398 ± 0.006	**0.422 ± 0.005%**
MSE	0.985 ± 0.010	**0.946 ± 0.008%**
MAE	0.789 ± 0.005	**0.770 ± 0.001%**

Conclusion

Advantages of more data:

- Larger training sets lead to much better results.

Advantages of pre-trained language models for scholarly document quality prediction:

- SChuBERT outperforms the other models significantly and has other benefits.
- Image + Text information are complementary:
  - $\text{SChuBERT}_{\text{Joint}}$ best model so far.

Future work

- Different (more powerful) language models: longformer/reformer
- More training data
- Adding context

PART 2: Neural Handwriting Recognition

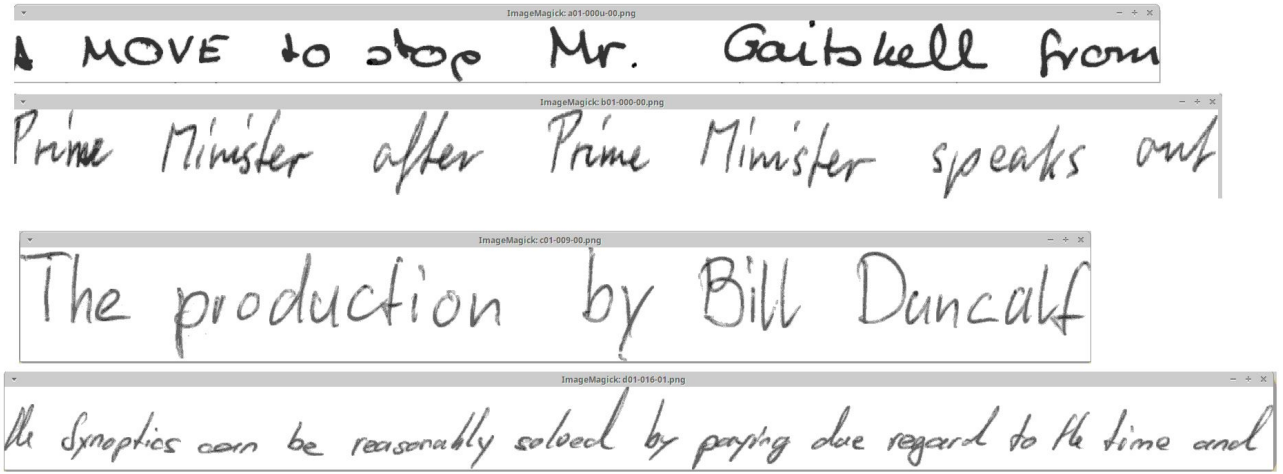
Based on:

Gideon Maillette de Buy Wenniger, Lambert Schomaker and Andy Way. 2019. "**No Padding Please: Efficient Neural Handwriting Recognition**" 2019 International Conference on Document Analysis and Recognition (ICDAR). Sydney, Australia. pages 355--362. doi: 10.1109/ICDAR.2019.00064.

https://ieeexplore.ieee.org/document/8978156

The handwriting recognition task

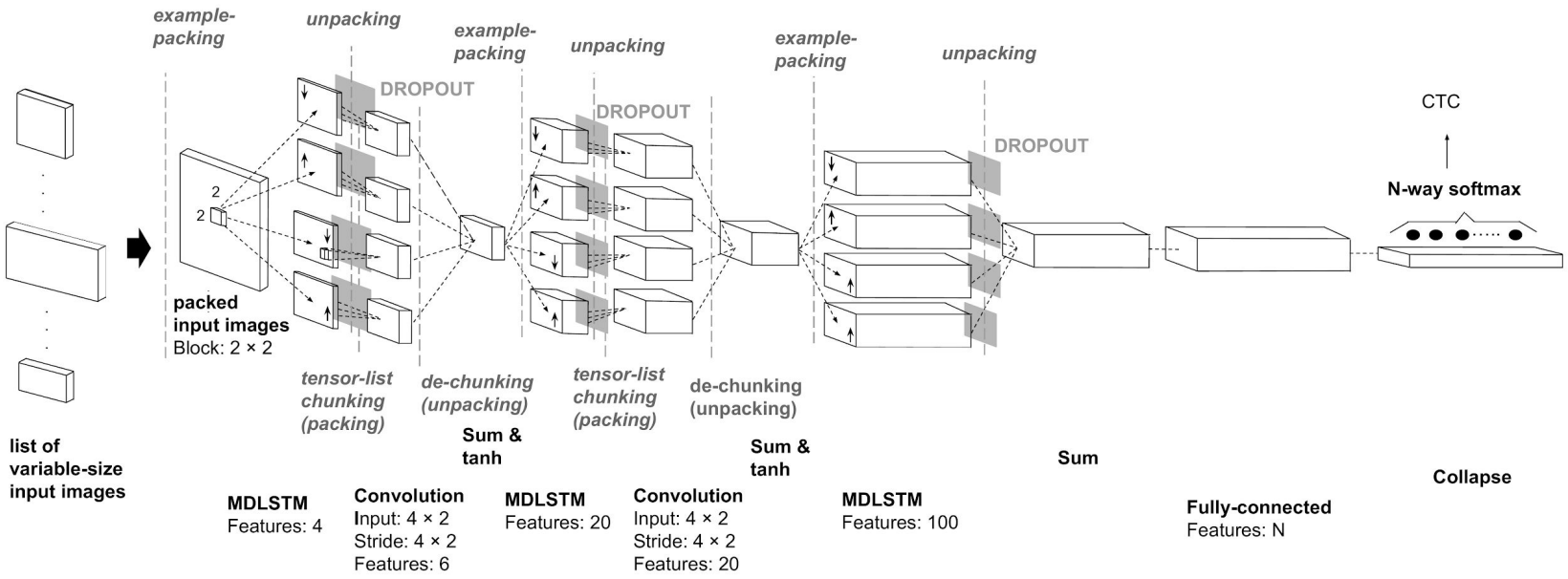
Input:



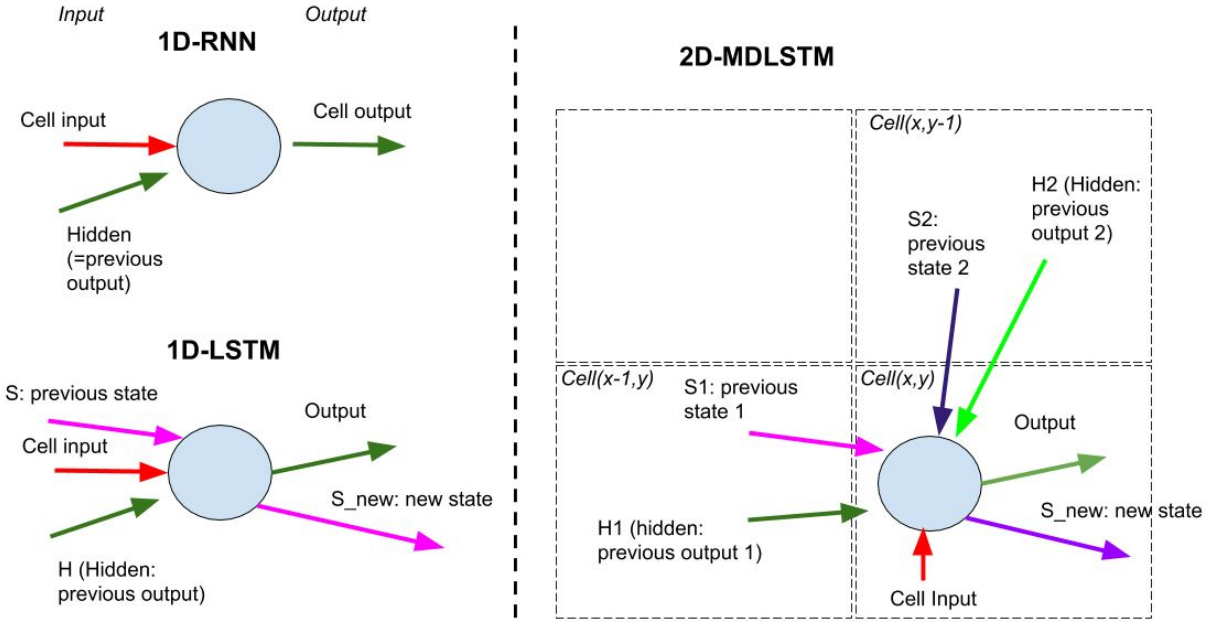
Output (Hopefully…)

- A|MOVE|to|stop|Mr.|Gaitskell|from
- Prime|Minister|after|Prime|Minister|speaks|out
- The|production|by|Bill|Duncalf
- the|Synoptics|can|be|reasonably|solved|by|paying|due|regard|to|the|time|and

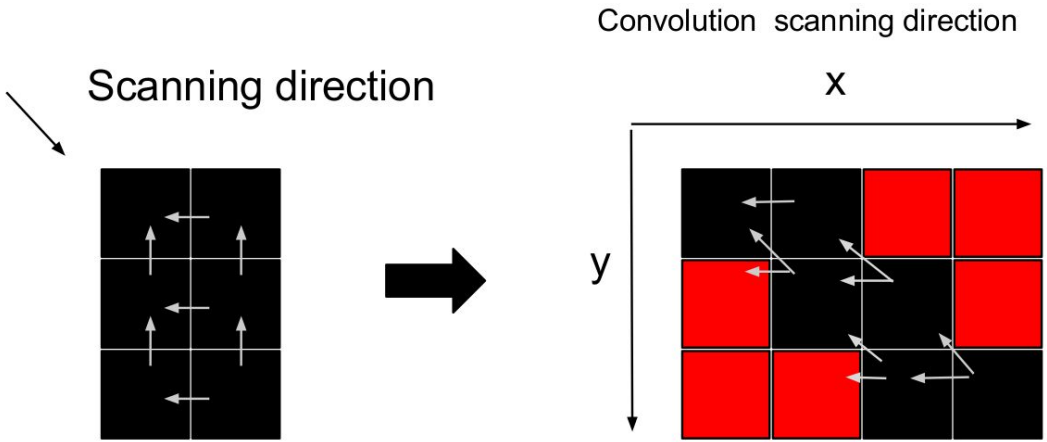
MDLSTM-based handwriting recognon



What are MDLSTMs?



Efficient MDLSTM computation by convolution[1]



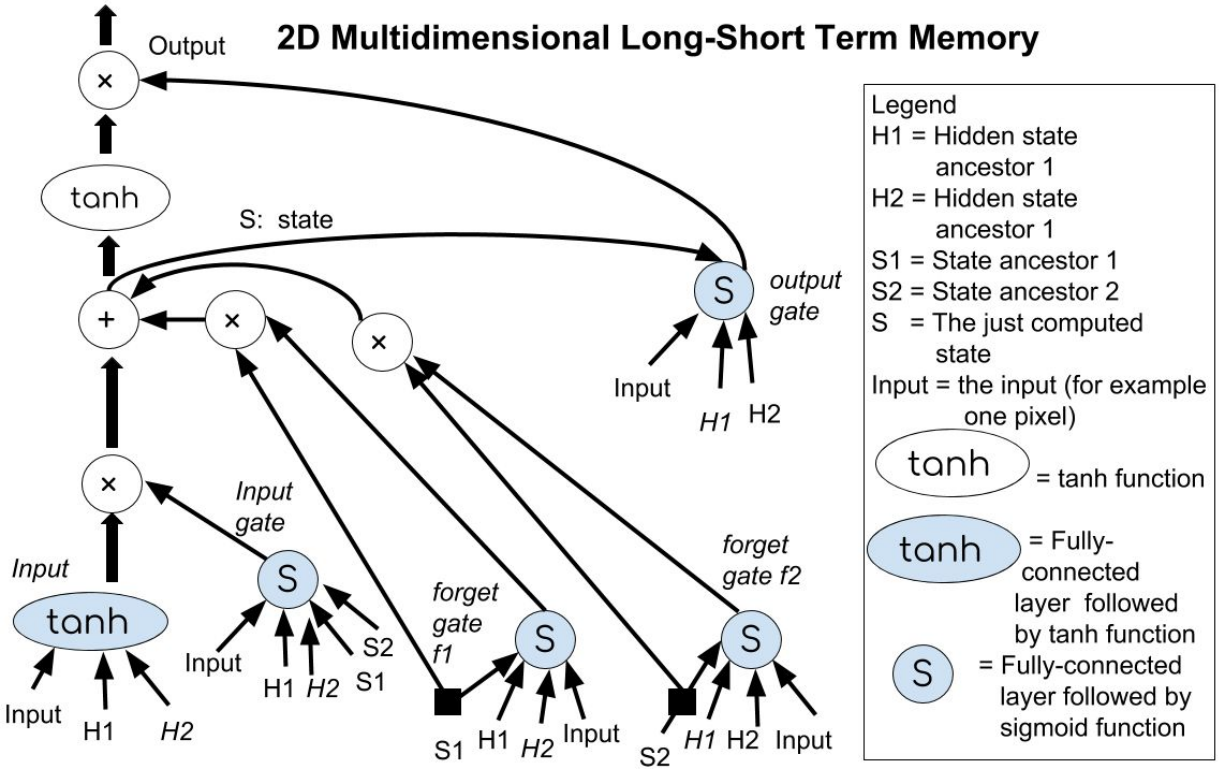
The red entries are invalid inputs that should be masked

Original Image

Diagonally skewed for convolutional computation MDLSTM

[1] First proposed in *Pixel Recurrent Neural Networks. (Van Den Oord et.al, 2016)*

MDLSTM Cell

- 

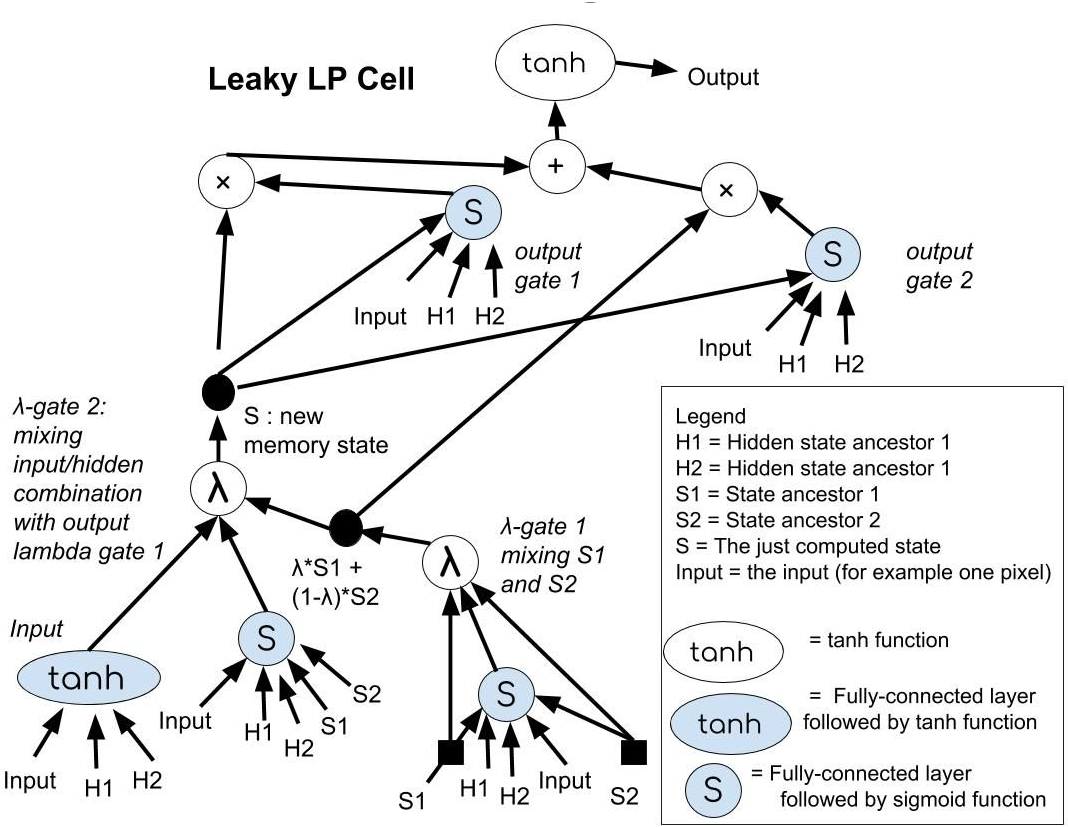
ity

MDLSTM Problems

- State can grow over time causing instability
  - Gradient clipping cannot solve this
- Multiplying S1 and S2 by 0.5? => State decays too fast…
- Better solution is needed that:
  - Prevents the state from growing to much and becoming instable
  - Still enables preserving state over a long time

  => Leaky LP Cells

Stable MDLSTM cells: Leaky LP



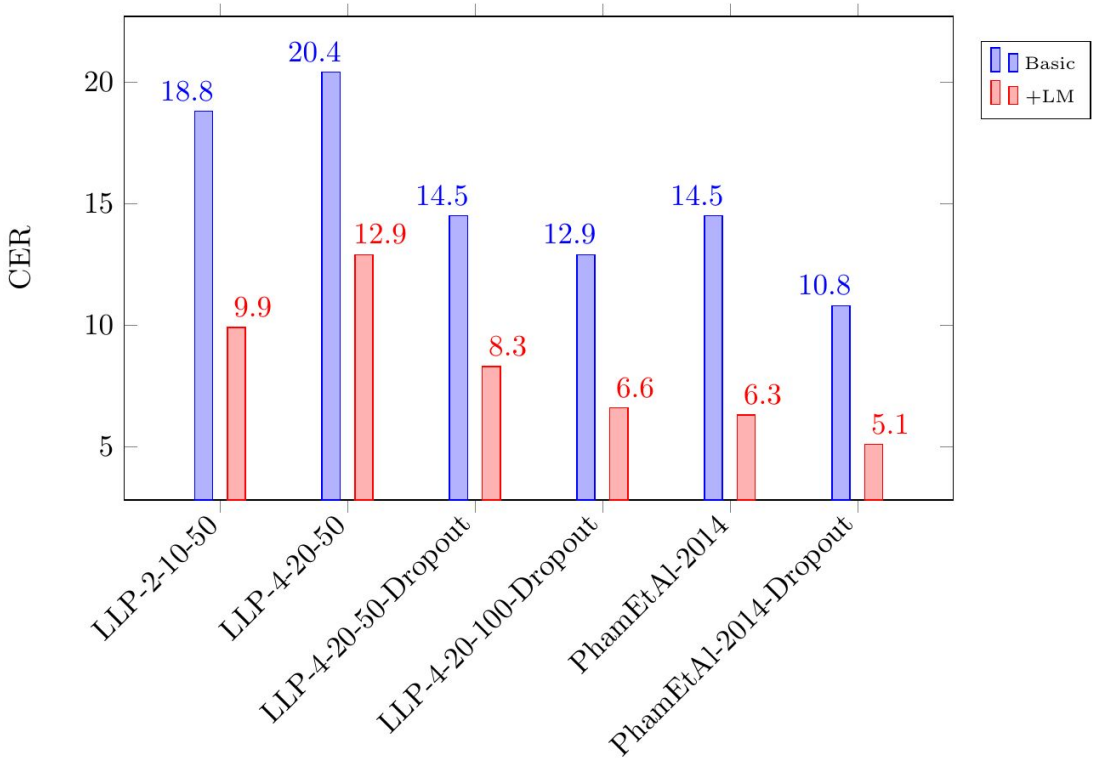
Experiments

- Line-strip handwriting recognition on the IAM benchmark dataset
- Material: lines taken from 1M word Lancaster-Oslo-Bergen (LOB) corpus
- Lines written by multiple writers

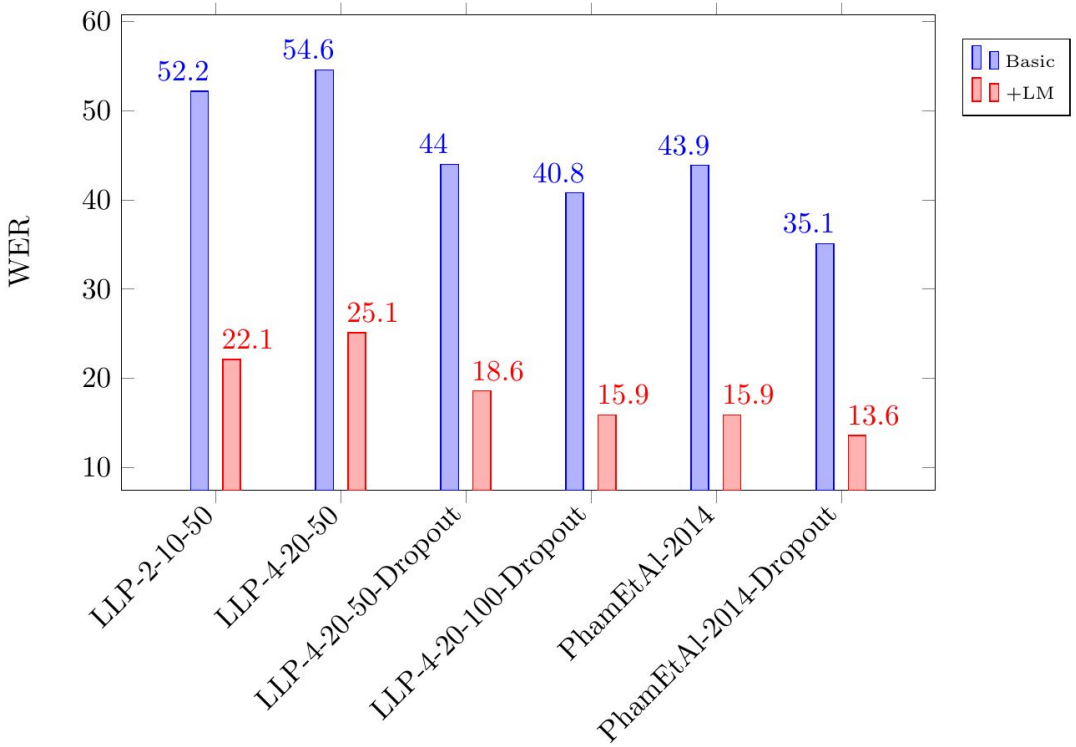
	IAM	
	words	lines
Training	80 421	6 482
Validation	16 770	976
Evaluation	17 991	2 915

- 3-gram language model trained on unused parts LOB corpus + Brown corpus.
- Results with/without language model

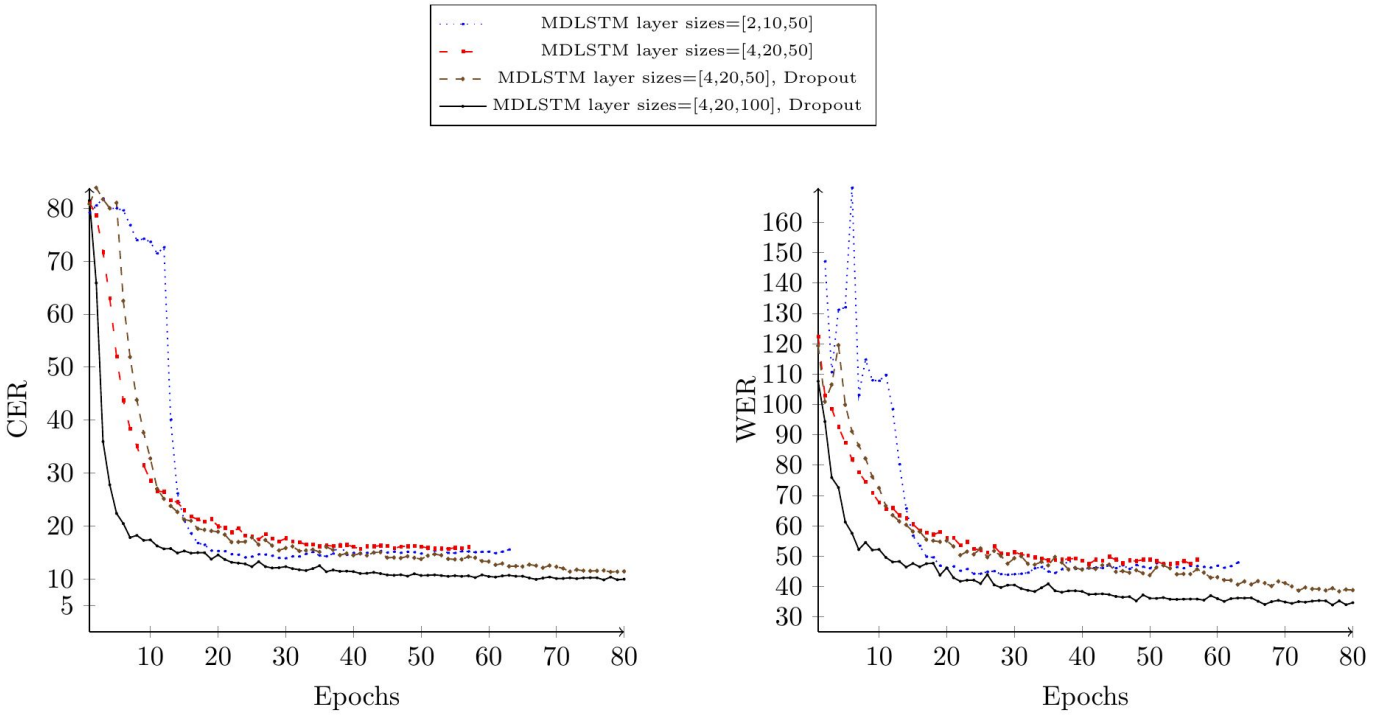
CER on IAM test-set



WER on IAM test-set



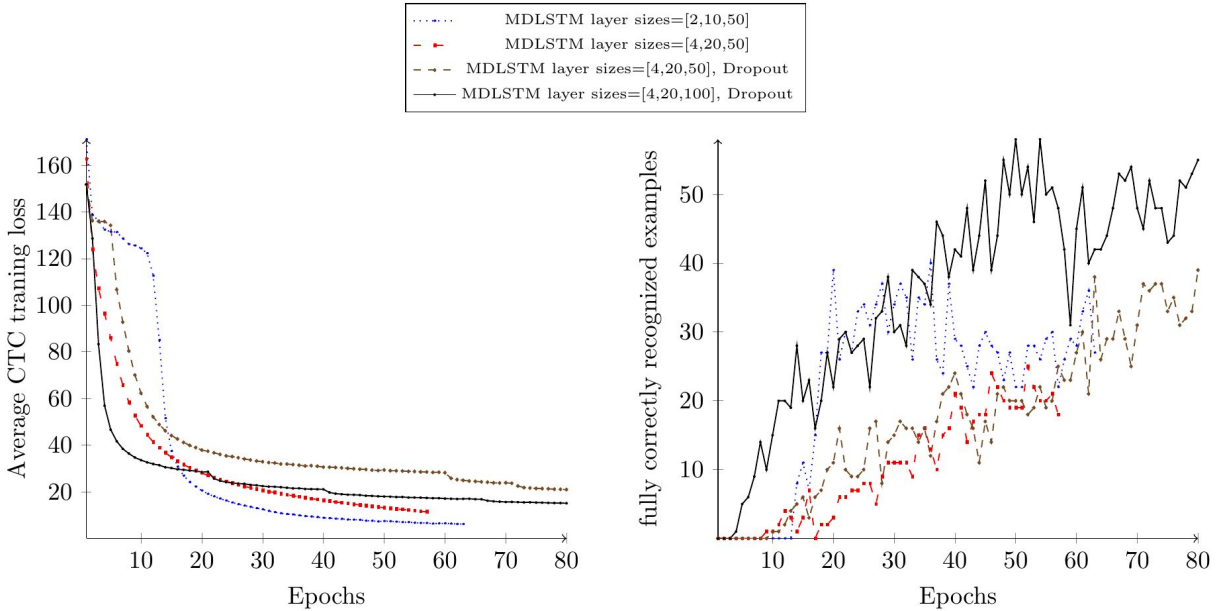
Handwriting recognition quality



(a) Character Error Rate (CER)

(b) Word Error Rate (WER)

IAM results (continued)



(c) Average CTC Loss per minibatch

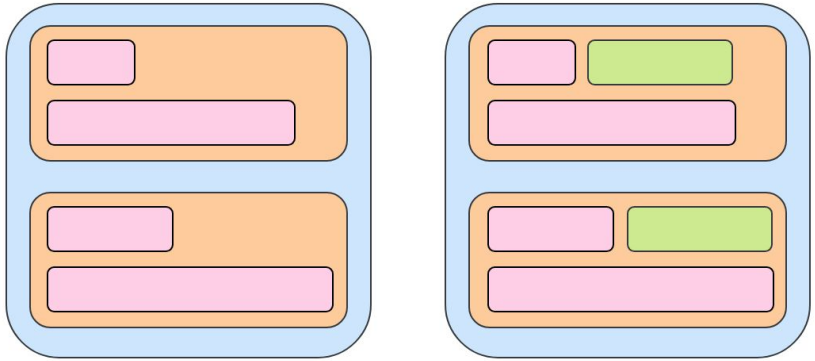
(d) Number of fully correctly recognized examples

Lessons learned

- Right combination of weight initialization scheme, optimizer and learning rate is crucial
- *Xavier Glorot* (Xavier Glorot and Joshua Bengio, 2010) *uniform* weight initialization in combination with *Adam* optimizer works well.
- Using *Leaky LP cell* (Leifert et. al, 2014) variant of MDLSTMs essential
- Dropout is required to get real good results and avoid overfitting

Padding

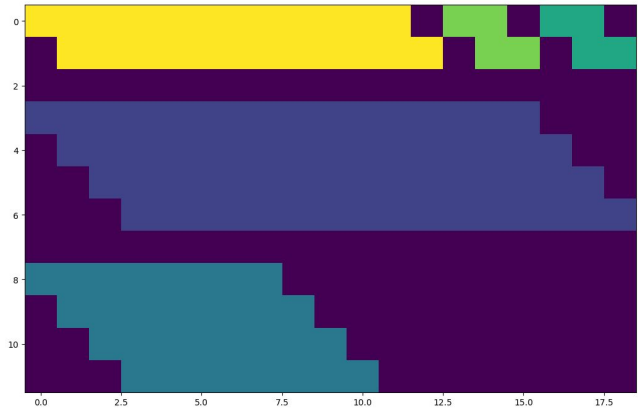
- Many neural models expect equal-sized inputs
- But input lengths differ
- Padding is a solution
  - But wastes a lot of computation

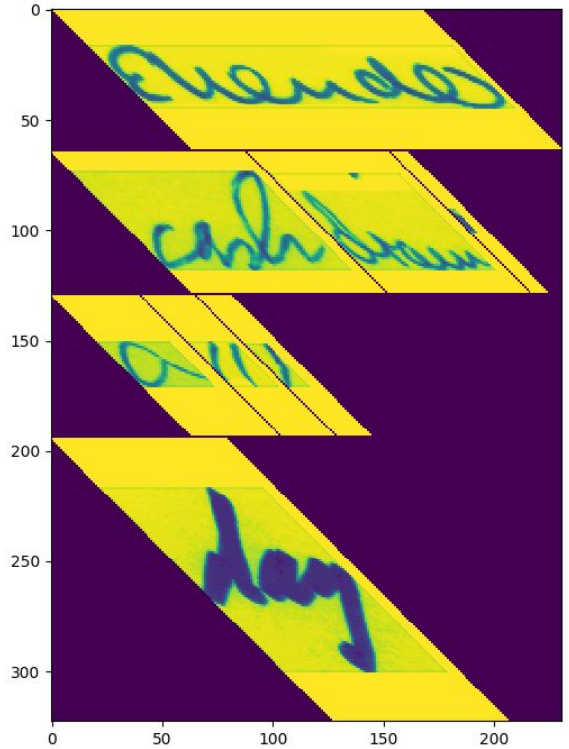


Original Padding

Solution : Example Packing

Idea: Pack variable-sized examples together, to minimize padding





Some observations and details

- Every row is filled greedily up to the maximum width
- Examples within a row must share same height, but different rows are allowed to have different heights
- Major gains especially in word-based handwriting recognition setting (due to large variance in word lengths)

Some observations and details (continued)

- Packing/unpacking done in pairs:
  - packing: List → Tensor
  - unpacking: Tensor → List
- Packing done before every MDLSTM layer, unpacking after it

- For block-strided convolution layers (after MDLSTM layers): use simplified packing/unpacking algorithms

Speedup of example packing

TABLE IV: Memory and time usage for models with and without example packing, with batch sizes chosen the maximal possible given the observed maximum GPU memory usage.

Preparation of batch examples	batch size	time per epoch (HH:MM:SS)	examples per second	max GPU1 memory use (MB)	max GPU2 memory use (MB)
IAM lines					
batch-padding	8	07:24:06	0.243	10824	10675
example-packing	12	05:04:45	0.355	10694	10780
IAM words					
batch-padding	20	06:26:48	2.38	11074	11144
example-packing	200	00:58:22.	16.1	10827	10849

Conclusions

- MDLSTM-based handwriting recognition
  - Importance of *Leaky LP cells, Dropout, Xavier weight initialization, optimizer and learning rate*

- Example packing: making better
- use of computational resources:
- 6.4 times speedup on words

- Full MDLSTM-based neural handwriting recognition in pytorch, open source

Questions?

Acknowledgements

This research has been supported by the ADAPT Centre for Digital Content Technology which is funded under the SFI Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Fund.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 713567.

Many thanks to Joost Bastings for invaluable consultation on deep learning technology and best practices. Special thanks to Paul Voigtlaender and Théodore Bluche for their helpful advise which has been important in getting MDLSTMs for handwriting recognition (from scratch) to work.

Extra Slides *Structure Tags* Paper

Number of citations prediction

- Predict **log(Number of citations + 1)**

  Why?

  - Compensates for Zipfian character number of citations
  - Always number in range [0, infinity], suitable for regression models
  - Alternative: predicting fixed categories (“low”, “medium”, “high” etc)
    - requires outlier analysis
    - more domain-dependent
    - Potential instability for articles with number of citations on the border of two classes/bins

S2ORC experiments number of citations prediction

- Large number of examples
- Retrieve number of citations any paper from open Semantic Scholar database
- But text length remains limited in our experiments for this paper

Table 8: S2ORC dataset size statistics.

data subset	num examples	avg num words
training	78894	839.1 ± 473.7
validation	4383	849.1 ± 477.5
testing	4382	856.4 ± 489.0

Number of citations prediction results

Table 9: Test scores for the log number of citations prediction on the S2ORC dataset.

	Average Word Embeddings	BiLSTM (re-implemented)	HAN	$\text{HAN}_{\text{struct-tag}}$
R^2 score	0.238 ± 0.0005	0.267 ± 0.007	0.275 ± 0.008	$\mathbf{0.285 \pm 0.002}$
mean squared error	1.261 ± 0.0008	1.214 ± 0.009	1.201 ± 0.007	$\mathbf{1.184 \pm 0.002}$
mean absolute error	0.867 ± 0.0002	0.842 ± 0.001	0.833 ± 0.003	$\mathbf{0.831 \pm 0.001}$

Related Work

Number of citations prediction:

- Fu and Aliferis (2008): citation count prediction using paper content (title, abstract and keywords) + bibliometric information.
- Li et al. (2019) and Plank and van Dale (2019): improved results using review information.

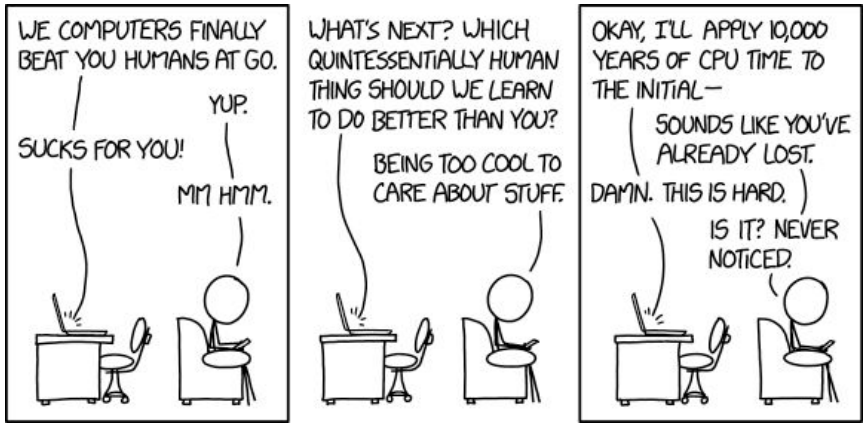
Accept/Reject prediction

- Shen et al. (2017) perform hybrid hand-crafted features + text content DL-based quality prediction of Wikipedia articles.
- Shen et al. (2019) combine visual and textual content using a CNN and LSTM respectively. Wikipedia and the PeerRead arXiv datasets.

Extra Slides *No Padding Please* Paper

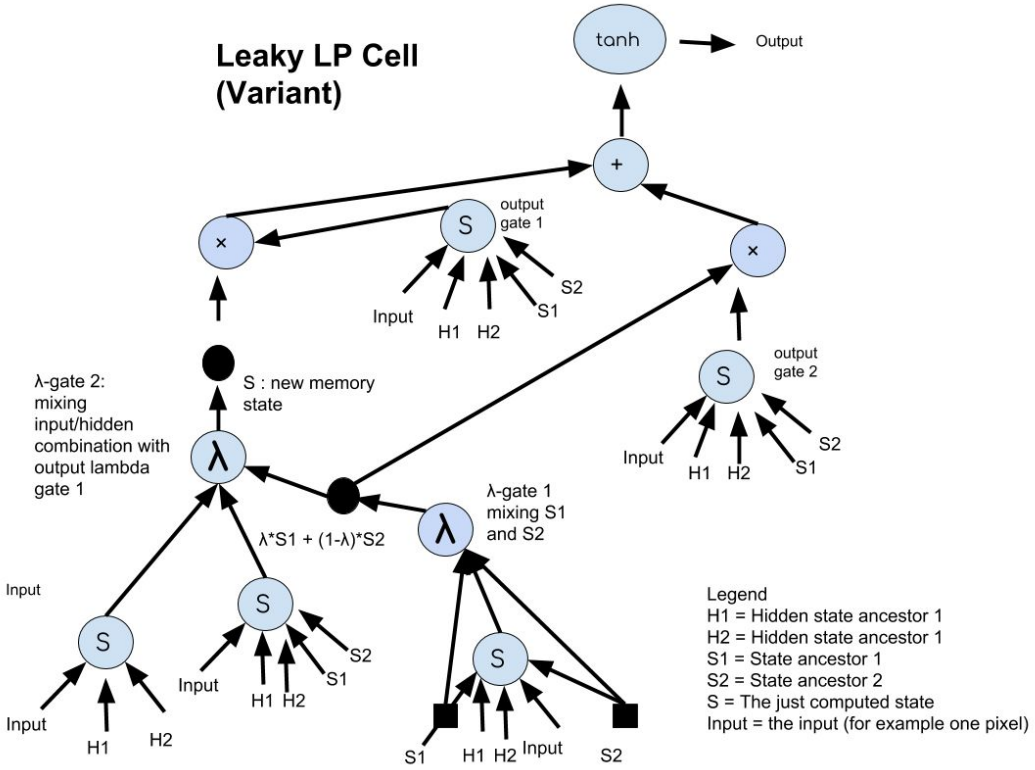
Bright and dark side of deep learning

- Unprecedented performance with same technology and similar models on many tasks



- Training models: takes long time and often inefficient in use of computational resources

Leaky LP Cell - variant



Handwriting results

>>> evaluate_mdrnn - output: "for|himself|only|,|instead|of|all|to|help|to|gather|your"

reference: "for|himself|only|,|instead|of|all|to|help|to|gather|your" --- correct

>>> evaluate_mdrnn - output: "midst|of|plenty|.|Hal|will|not|be|easily|forth"

reference: "midst|of|plenty|.|Help|will|not|be|easily|forth-" --- wrong

>>> evaluate_mdrnn - output: "coming|for|the|people|in|need|.|They|will"

reference: "coming|for|the|people|in|need|.|They|will" --- correct

>>> evaluate_mdrnn - output: "think|of|the|animals|first|(|which|is|of|course"

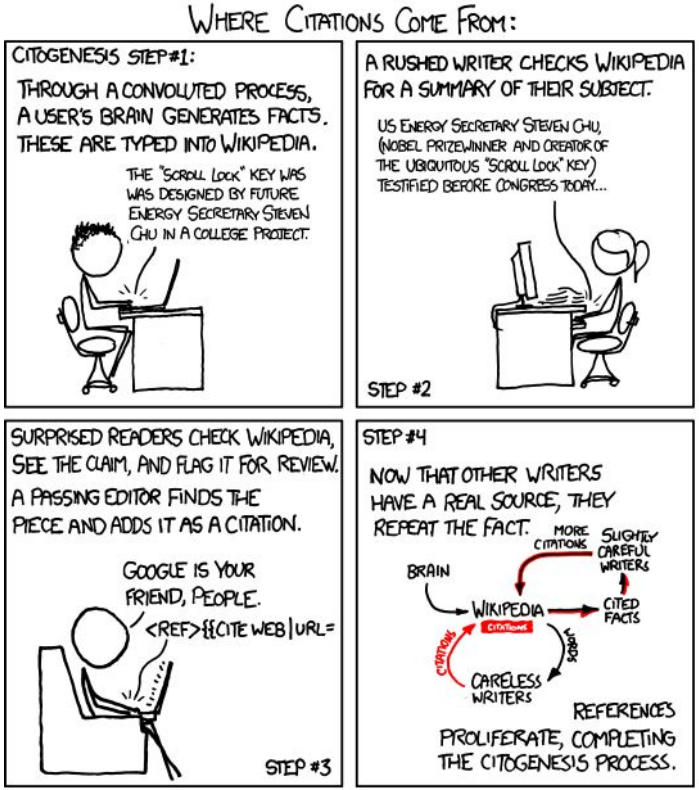
reference: "think|of|the|animals|first|(|which|is|of|course" --- correct

>>> evaluate_mdrnn - output: "our|duty|)|.|Of|course|the|individual|will"

reference: "our|duty|)|.|Of|course|the|individual|will" --- correct

Bonus Material

Where citations come from



Are all structure-tags necessary?

- Ablation experiment 2 tags: merge Title and Abstract tags

- Result: performance loss on all PeerRead datasets
- Sometimes performing worse than plain HAN

Table 7: Results of the HAN_{ST} model with a reduced structure-tag set of only two tags.

metric \ domain	artificial intelligence	computation & language	machine learning
accuracy	$89.6 \pm 1.57\%$	$79.3 \pm 0.14\%$	$77.2 \pm 1.21\%$
AUC	0.610 ± 0.067	0.727 ± 0.015	0.759 ± 0.017

Hyperparameters used

	PeerRead classification	S2ORC regression
optimizer, learning rate	Adam, 0.005	
maximum input characters	20000	
vocabulary size	10000	
weight initialization		
general	Xavier uniform	
lstm	Xavier normal	
bias	zero	
word embeddings	GloVe	
loss function	cross entropy	MAE
dropout probability	0.5	0.2
BiLSTM hidden size	256	100
batch size	4	64
embedding size	50	300