Natural Language Processing with Deep Learning

CS224N/Ling284



**Christopher Manning**

**Lecture 14: More on Contextual Word Representations and Pretraining**

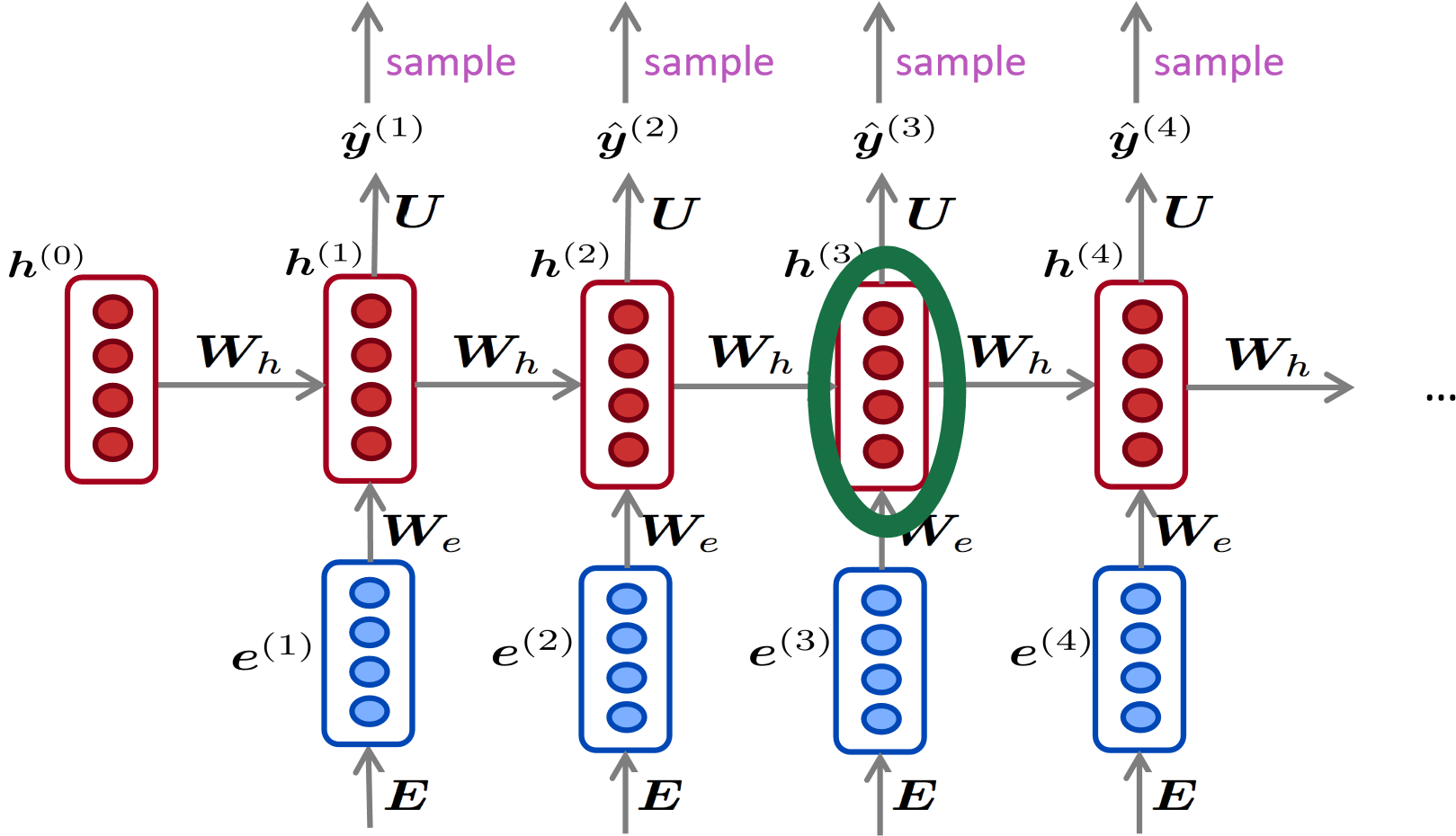
# 1. Representations for a word

* Originally, we basically had one representation of words:
* The word vectors that we learned about at the beginning
* Word2vec, GloVe, fastText
* These have two problems:
* Always the same representation for a **word type** regardless of the context in which a **word token** occurs
* We might want very fine-grained word sense disambiguation
* We just have one representation for a word, but words have different **aspects**, including semantics, syntactic behavior, and register/connotations

**Did we all along have a solution to this problem?**

* In, an NLM, we immediately stuck word vectors (perhaps only trained on the corpus) through LSTM layers
* Those LSTM layers are trained to predict the next word
* But those language models are producing context-specific word representations at each position!

*favorite season is spring*



*my favorite season is spring*

# Context-free vs. contextual similarity

|  |  |  |
| --- | --- | --- |
| **Model** | **Source** | **Nearest Neighbor(s)** |
| GloVe | play | playing, game, games, played, players, plays, player, Play, football, multiplayer |
| BiLM | Chico Ruiz made a spectacular play on Alusik ’s grounder {. . . } | Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent play . |
|  | Olivia De Havilland signed to do a Broadway play for  Garson {. . . } | {. . . } they were actors who had been handed fat roles in a successful play , and had talent enough to fill the roles competently , with nice understatement . |

From Peters et al. 2018 Deep contextualized word representations (ELMo paper)

# 2. Pre-ELMo and ELMo

## Dai and Le (2015) <https://arxiv.org/abs/1511.01432>

• Why don’t we do semi-supervised approach where we train NLM sequence model on large unlabeled corpus, rather than just word vectors and use as pretraining for sequence model

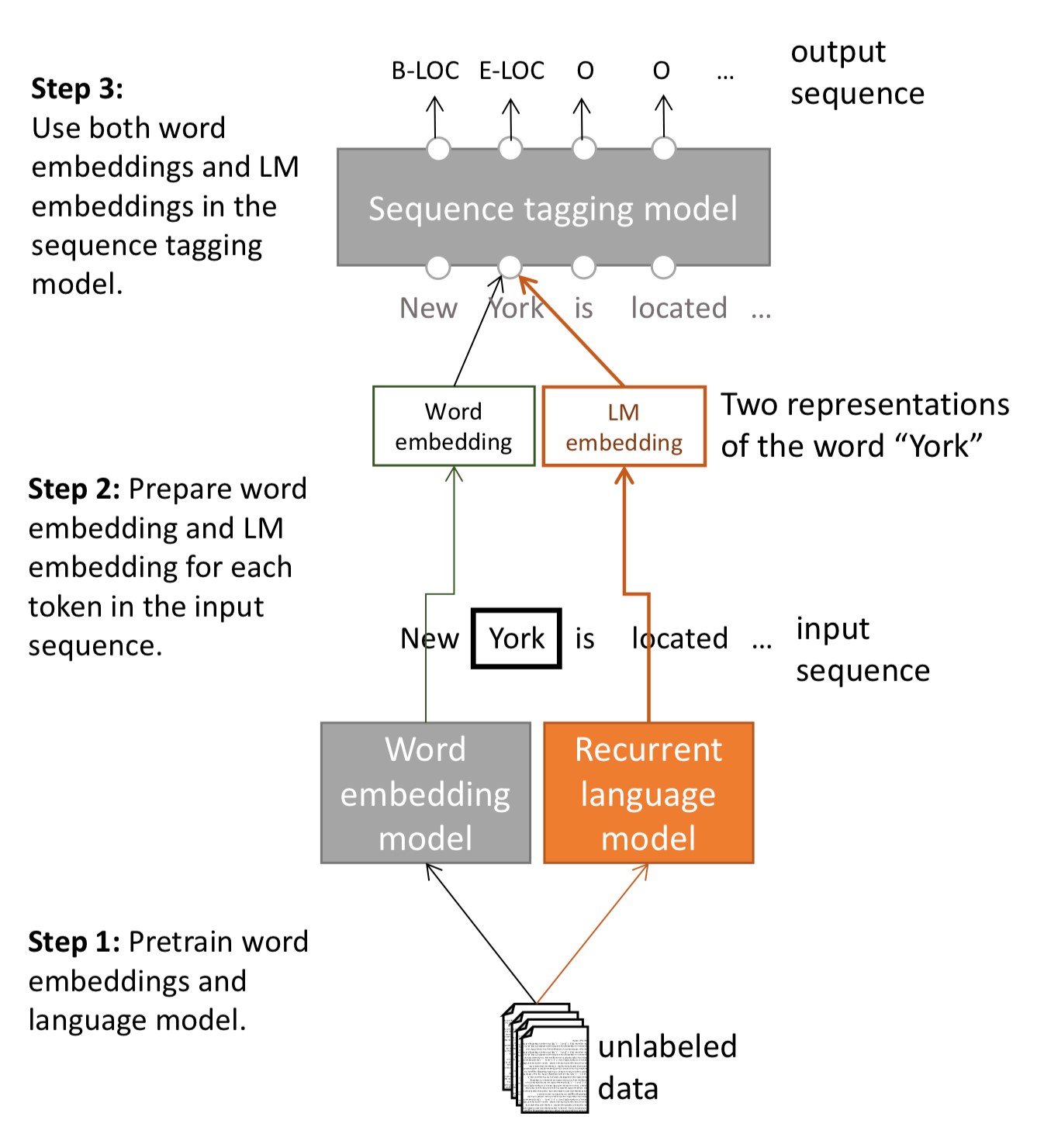
## Peters et al. (2017) <https://arxiv.org/pdf/1705.00108.pdf>

* Idea: Want meaning of word in context, but standardly learn task RNN only on small task-labeled data (e.g., NER)

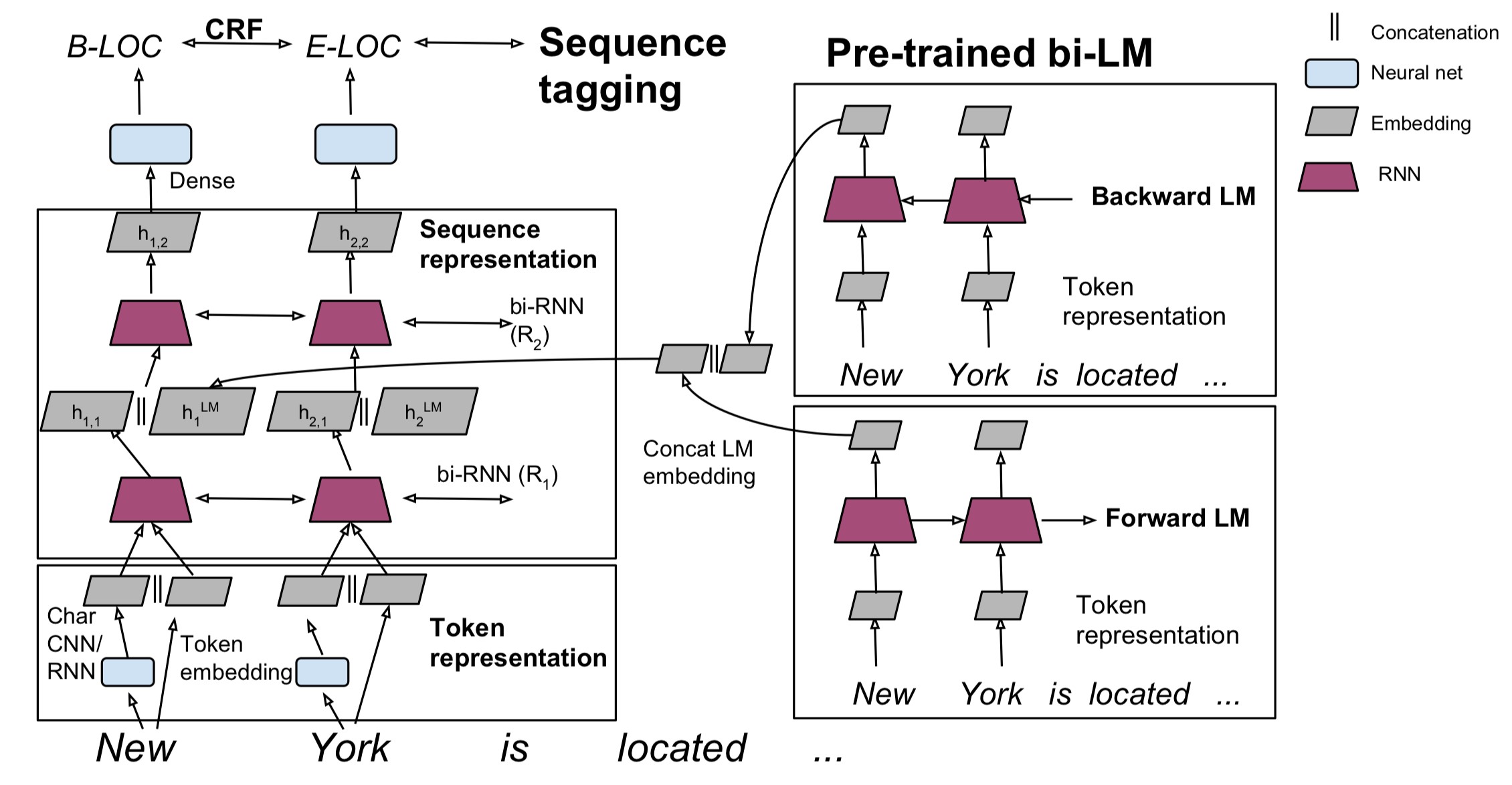
**Howard and Ruder (2018)** Universal Language Model Fine-tuning for Text Classification. <https://arxiv.org/pdf/1801.06146.pdf>

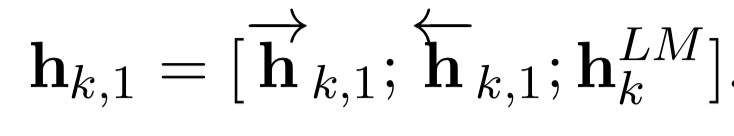
* Same general idea of transferring NLM knowledge
* Here applied to text classification

# Tag LM (Peters et al. 2017)



# Tag LM





# Named Entity Recognition (NER)

* Find and classify names in text, for example:
* The decision by the independent MP Andrew

|  |
| --- |
| Person  Date  Location  Organization |

Wilkie to withdraw his support for the minority Labor government sounded dramatic but it should not further threaten its stability. When, after the 2010 election, Wilkie, Rob Oakeshott, Tony Windsor and the Greens agreed to support Labor, they gave just two guarantees: confidence and supply.

**Peters et al. (2018): ELMo: Embeddings from Language Models**

## CoNLL 2003 Named Entity Recognition (en news testb)

**Name Description Year F1**

[Flair (Zalando)](https://github.com/zalandoresearch/flair) Character-level language model 2018 93.09

BERT Large Transformer bidi LM + fine tune 2018 92.80

CVT Clark Cross-view training + multitask learn 2018 92.61

BERT Base Transformer bidi LM + fine tune 2018 92.40

ELMo ELMo in BiLSTM 2018 92.22

TagLM Peters LSTM BiLM in BiLSTM tagger 2017 91.93

Ma + Hovy BiLSTM + char CNN + CRF layer 2016 91.21

Tagger Peters BiLSTM + char CNN + CRF layer 2017 90.87

# Peters et al. (2017): TagLM – “Pre-ELMo”

Language model is trained on 800 million training words of “Billion word benchmark”

Language model observations

* An LM trained on supervised data does not help
* Having a bidirectional LM helps over only forward, by about 0.2
* Having a huge LM design (ppl 30) helps over a smaller model (ppl 48) by about 0.3

Task-specific BiLSTM observations

* Using just the LM embeddings to predict isn’t great: 88.17 F1
* Well below just using an BiLSTM tagger on labeled data

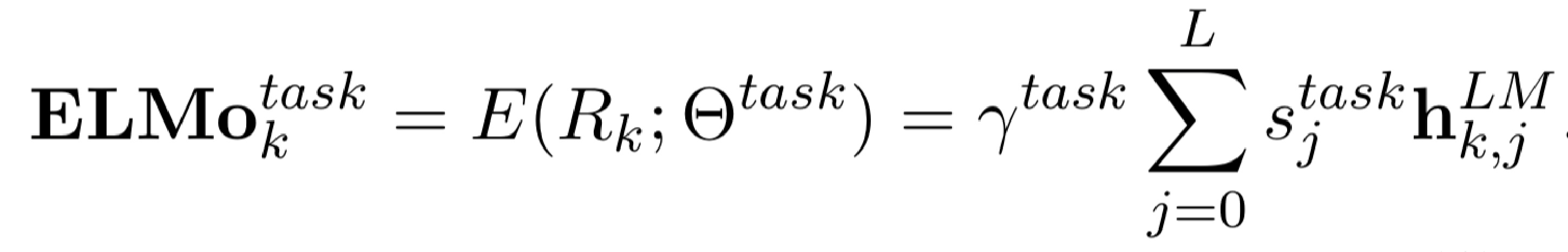
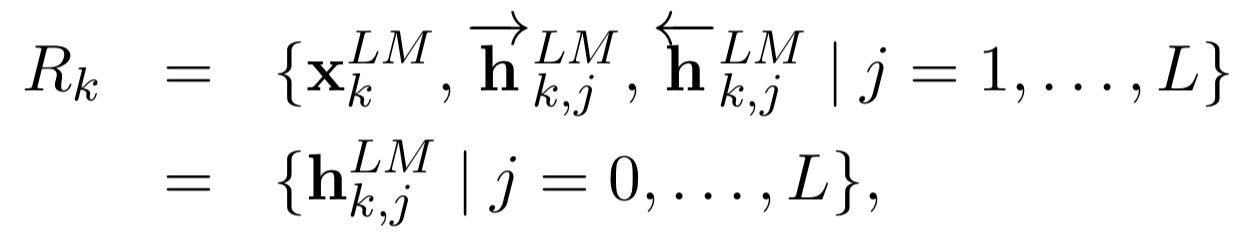
Deep contextualized word representations. NAACL 2018.

https://arxiv.org/abs/1802.05365

* Initial breakout version of **word token vectors** or **contextual word vectors**
* Learn word token vectors using long contexts not context windows (here, whole sentence, could be longer)
* Learn a deep Bi-NLM and use all its layers in prediction



* Train a bidirectional LM
* Aim at performant but not overly large LM:
* Use 2 biLSTM layers
* Use character CNN to build initial word representation (only) • 2048 char n-gram filters and 2 highway layers, 512 dim projection
* Use 4096 dim hidden/cell LSTM states with 512 dim projections to next input
* Use a residual connection
* Tie parameters of token input and output (softmax) and tie these between forward and backward LMs
* ELMo learns task-specific combo of biLM layer representations
* This is an innovation that improves on just using top layer of LSTM stack



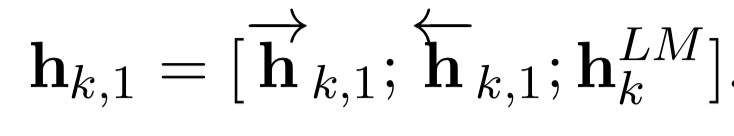
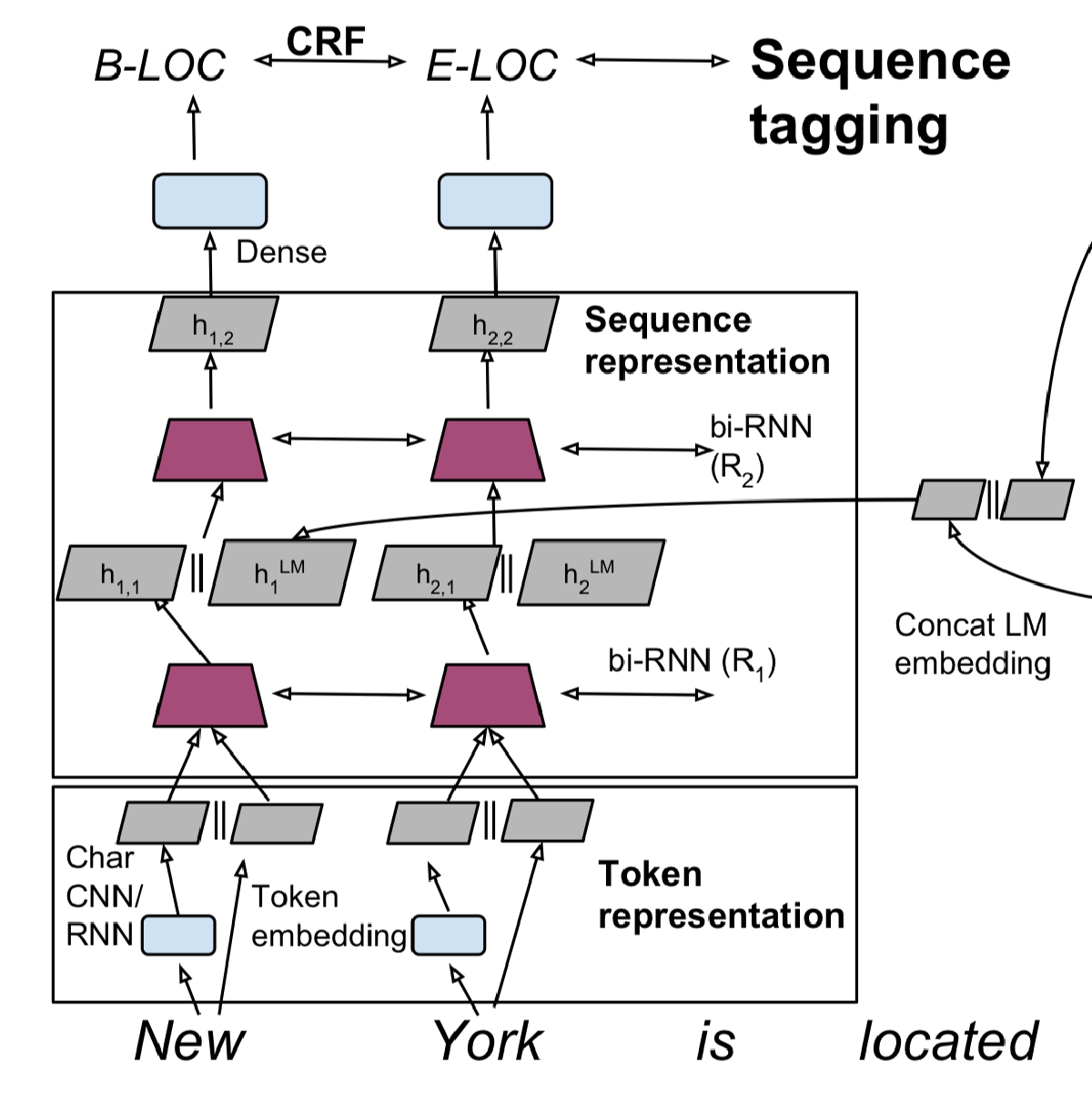
* 𝛾"#$% scales overall usefulness of ELMo to task;
* 𝐬"#$% are softmax-normalized mixture model weights

# Peters et al. (2018): ELMo: Embeddings from Language Models. Use with a task

* First run biLM to get representations for each word
* Then let (whatever) end-task model use them
* Freeze weights of ELMo for purposes of supervised model
* Concatenate ELMo weights into task-specific model
* Details depend on task
* Concatenating into intermediate layer as for TagLM is typical
* Can provide ELMo representations again when producing outputs, as in a question answering system

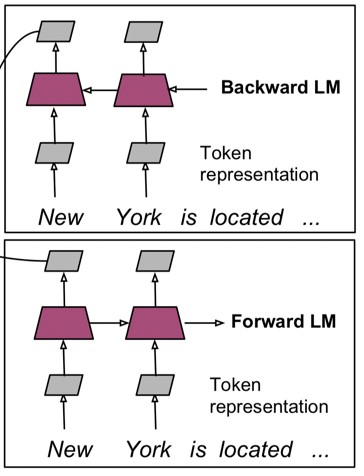
# ELMo used in an NER tagger

Breakout version of **deep contextual word vectors**



ELMo representation:

A deep bidirectional neural LM

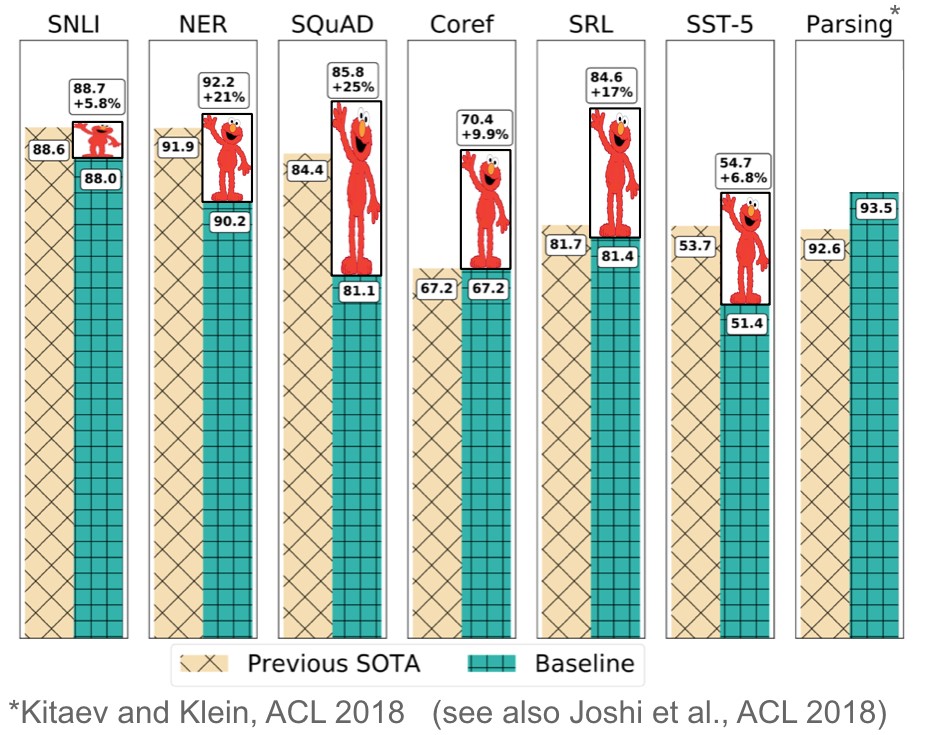


Use learned, task

-

weighted

average of (2) hidden layers



**95.1**

**+1.7**

**%**

## CoNLL 2003 Named Entity Recognition (en news testb)

**Name Description Year F1**

[Flair (Zalando)](https://github.com/zalandoresearch/flair) Character-level language model 2018 93.09

BERT Large Transformer bidi LM + fine tune 2018 92.80

CVT Clark Cross-view training + multitask learn 2018 92.61

BERT Base Transformer bidi LM + fine tune 2018 92.40

ELMo ELMo in BiLSTM 2018 92.22

TagLM Peters LSTM BiLM in BiLSTM tagger 2017 91.93

Ma + Hovy BiLSTM + char CNN + CRF layer 2016 91.21

Tagger Peters BiLSTM + char CNN + CRF layer 2017 90.87

# ELMo: Weighting of layers

* The two biLSTM NLM layers have differentiated uses/meanings • Lower layer is better for lower-level syntax, etc. • Part-of-speech tagging, syntactic dependencies, NER
* Higher layer is better for higher-level semantics
* Sentiment, Semantic role labeling, question answering, SNLI
* This seems interesting, but it’d seem more interesting to see how it pans out with more than two layers of network

# 3. Also in the air: McCann et al. 2017: CoVe

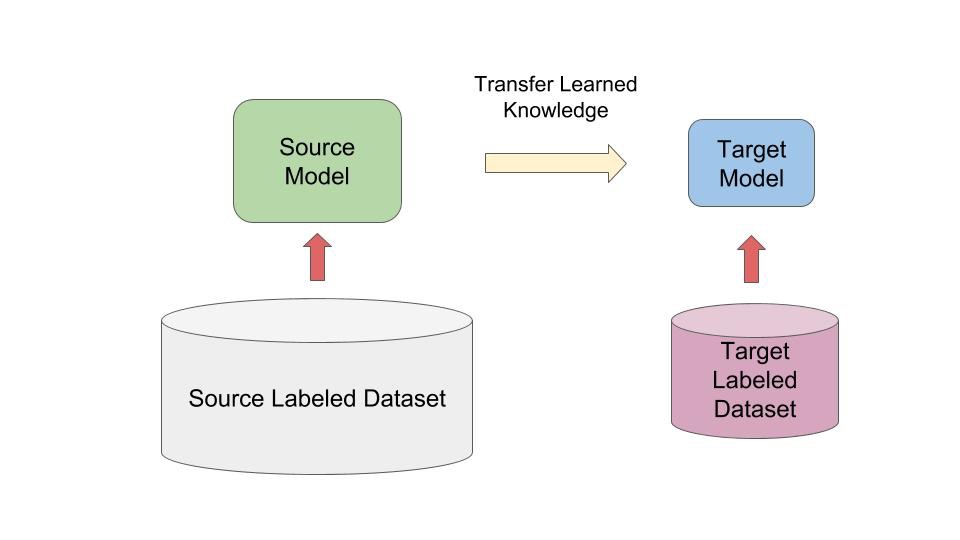
<https://arxiv.org/pdf/1708.00107.pdf>

* Also has idea of using a trained sequence model to provide context to other NLP models
* Idea: Machine translation is meant to preserve meaning, so maybe that’s a good objective?
* Use a 2-layer bi-LSTM that is the encoder of seq2seq + attention NMT system as the context provider
* The resulting CoVe vectors do outperform GloVe vectors on various tasks
* But, the results aren’t as strong as the simpler NLM training described in the rest of these slides so seems abandoned
* Maybe NMT is just harder than language modeling?
* Maybe someday this idea will return?

# Also around: ULMfit

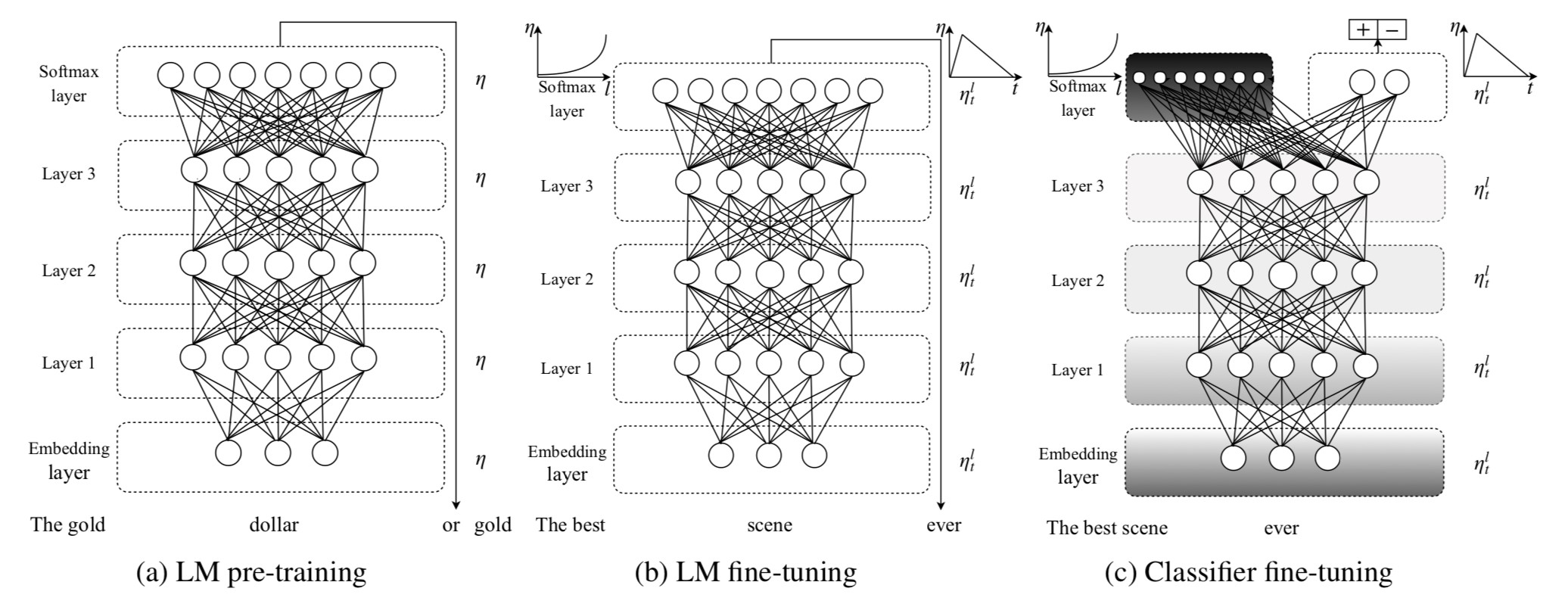
Howard and Ruder (2018) Universal Language Model Fine-tuning for Text Classification. https://arxiv.org/pdf/1801.06146.pdf

* Same general idea of transferring NLM knowledge
* Here applied to text classification



**ULMfit emphases**

Train LM on big general domain corpus (use biLM)

Tune LM on target task data

Fine-tune as classifier on target task

**ULMfit emphases**

Use reasonable-size “1 GPU” language model not really huge one

A lot of care in LM fine-tuning

Classify using concatenation

[

ℎ

)

,

maxpool

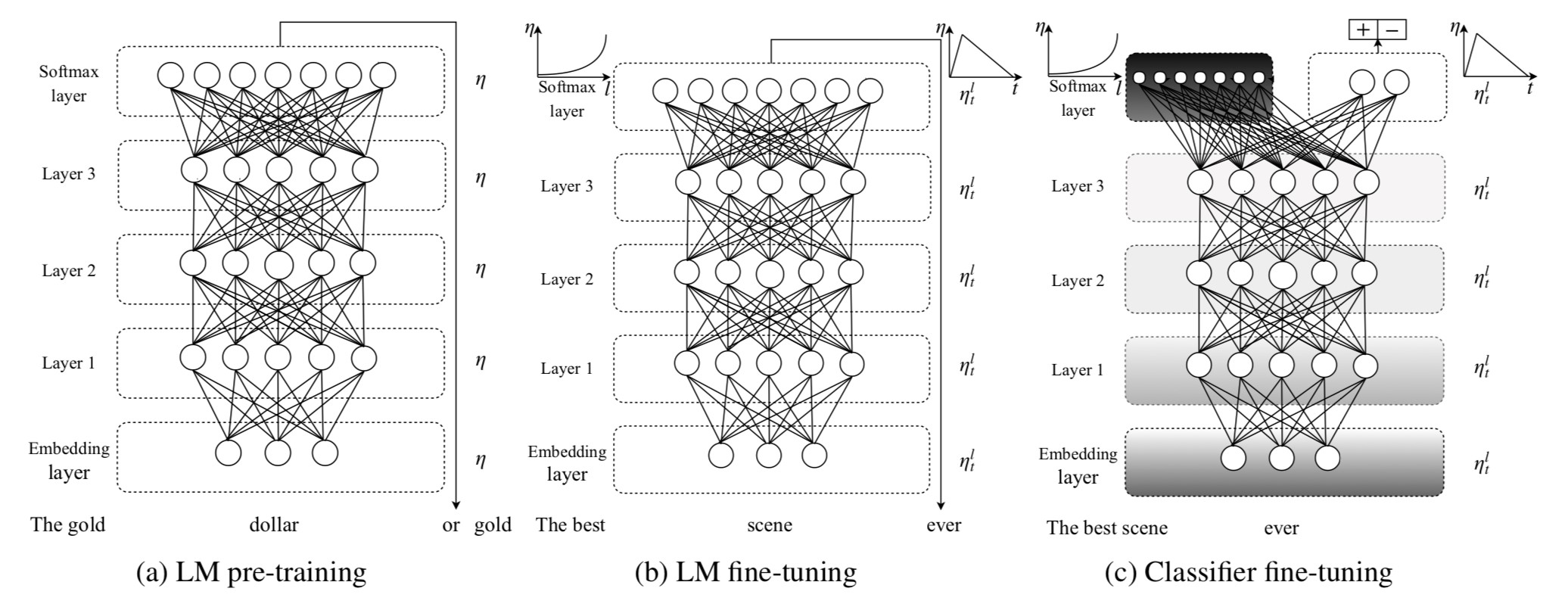
𝐡

,

meanpool

𝐡

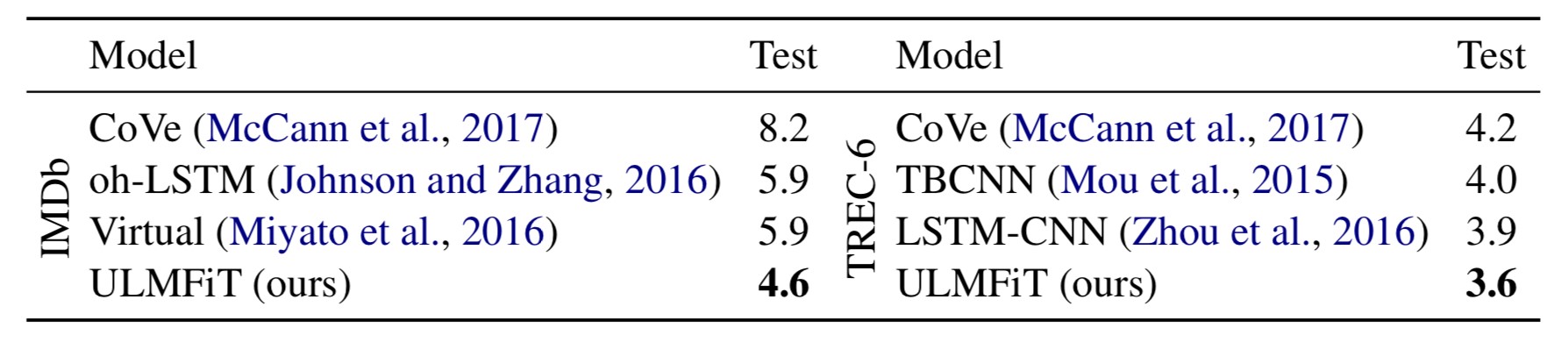
]



Different per-layer learning rates

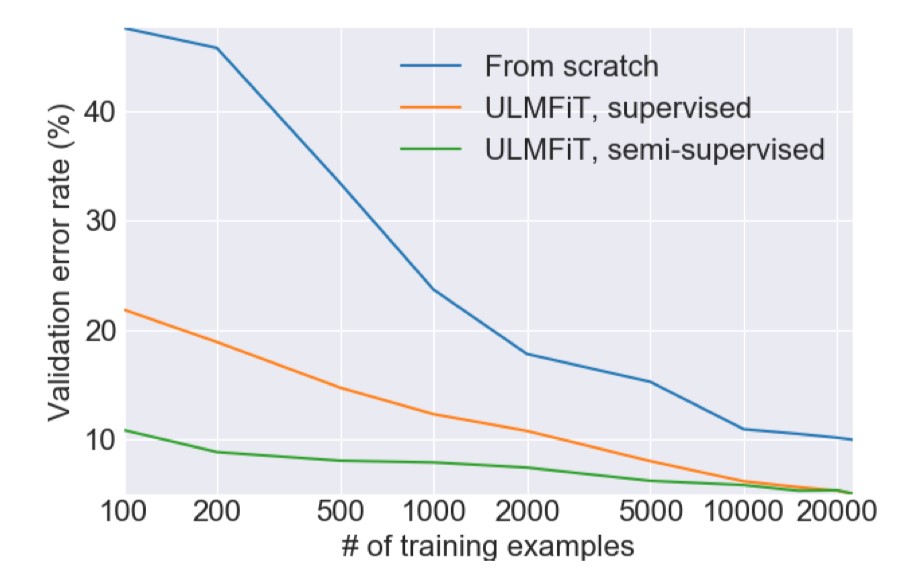
Slanted triangular learning rate (STLR) schedule

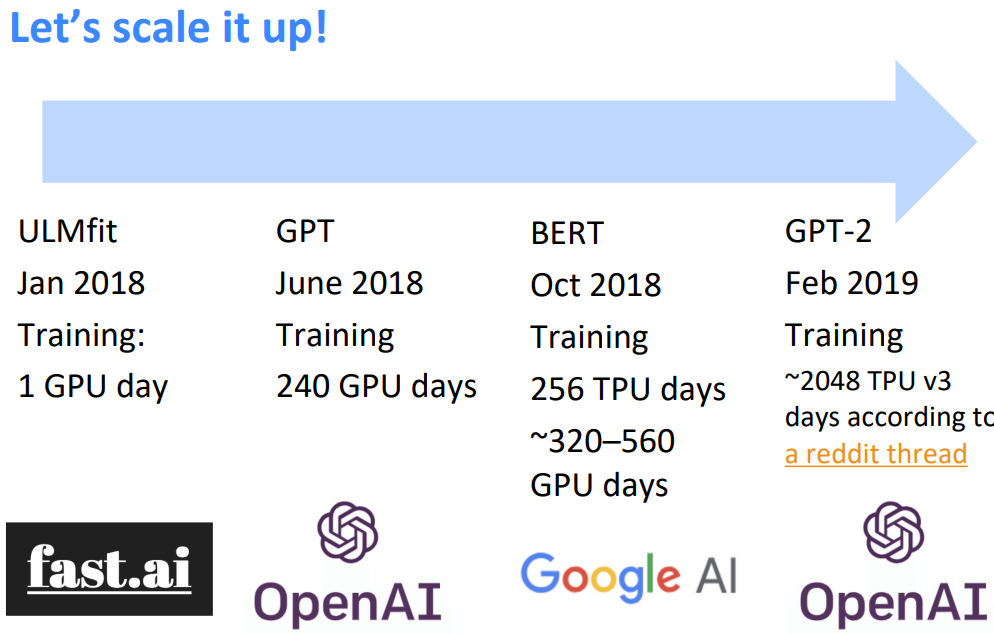
Gradual layer unfreezing and STLR when learning classifier

 **ULMfit performance**

• Text classifier error rates

# transfer learning



**

*researchers was the fact that the* WRITTEN) *unicorns spoke perfect English.*

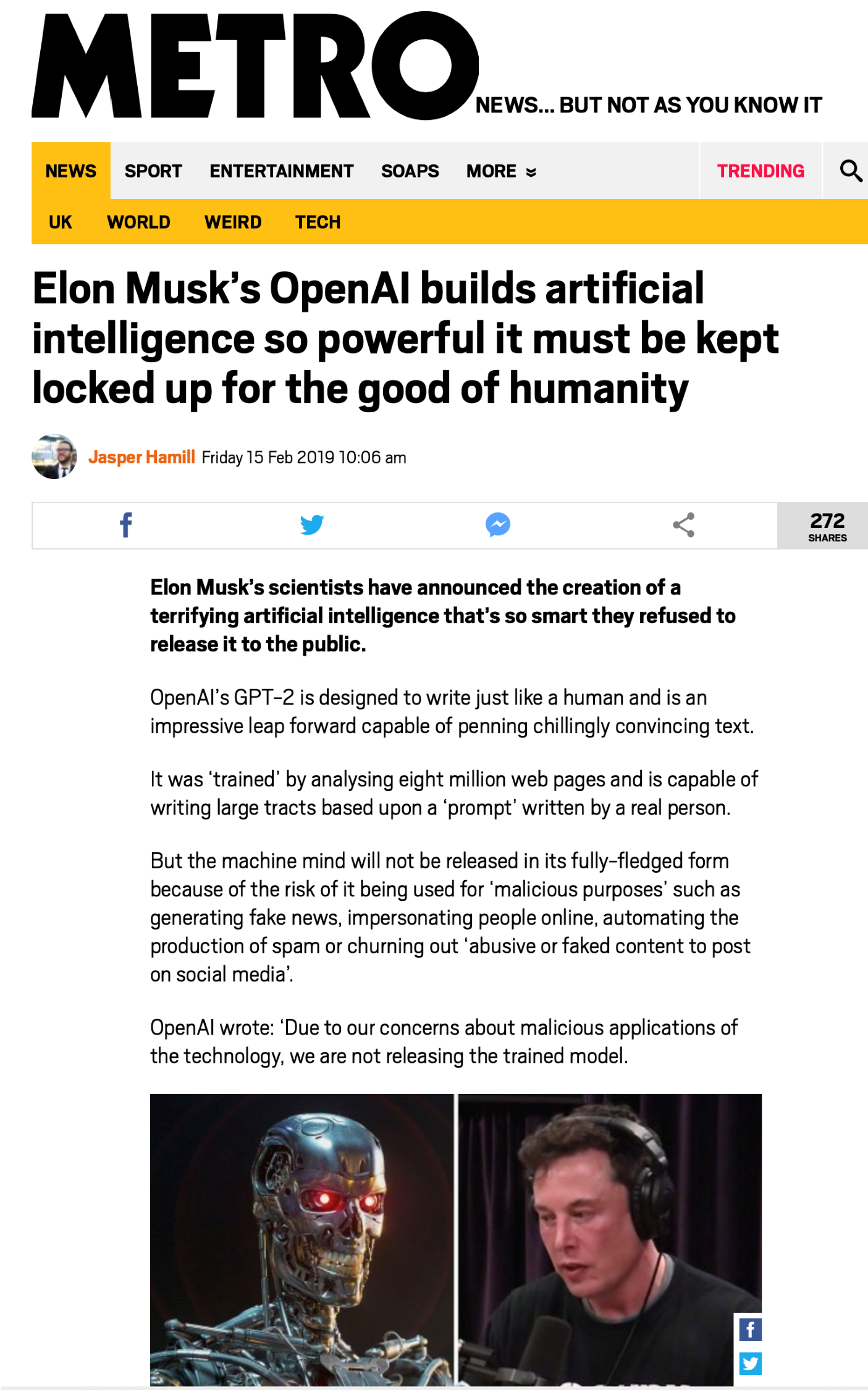
The scientist named the population, after their distinctive horn, MODEL Ovid’s Unicorn. These four-horned, silver-white unicorns were COMPLETION previously unknown to science.

(MACHINE- Now, after almost two centuries, the mystery of what sparked WRITTEN, this odd phenomenon is finally solved.

10 TRIES) Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

Pérez and the others then ventured further into the valley. …

30



## 4. Transformer models

All of

these models

are Transformer

models

ELMo

Oct 2017

Training:

M words

800

GPU days

42

BERT

Oct 2018

Training

3.3

B words

TPU days

256

~320

–

560

GPU days

GPT

-

2

Feb 2019

Training

40

B words

~2048 TPU v3

days according to

[d](https://www.reddit.com/r/MachineLearning/comments/aqlzde/r_openai_better_language_models_and_their/)

a reddit threa

GPT

June 2018

Training

800

M words

240

GPU days



XL

-

Net,

ERNIE,

Grover

RoBERTa

, T5

July 2019

—



32

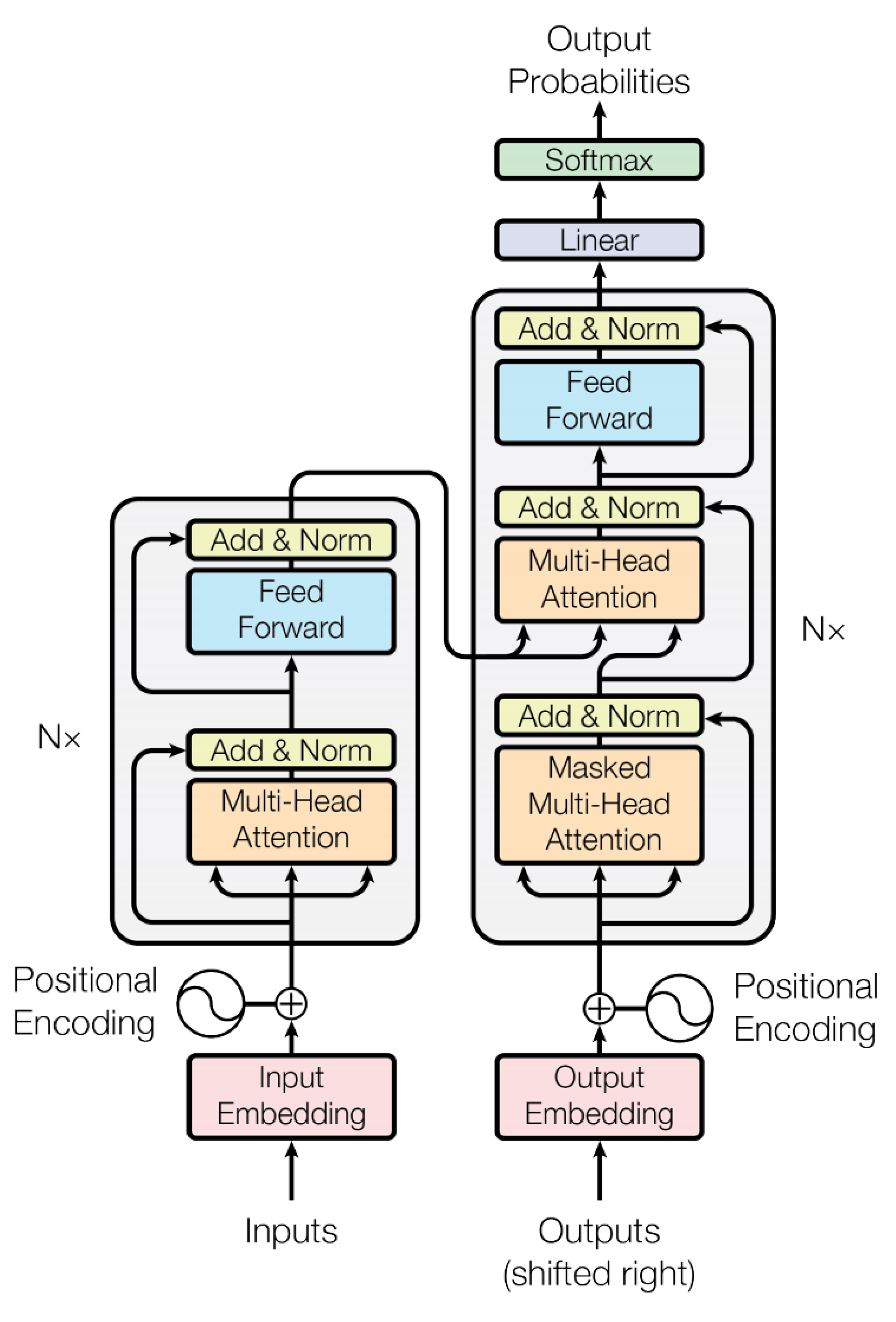
# The Transformer

Attention is all you need. 2017.

Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, Polosukhin https://arxiv.org/pdf/1706.03762.pdf

* Non-recurrent sequence-tosequence encoder-decoder model
* Task: machine translation with parallel corpus
* Predict each translated word • Final cost/error function is standard cross-entropy error on top of a softmax classifier

This and related figures from paper ⇑



# Transformer Basics

* Learning about transformers on your own?
* Key recommended resource:
* <http://nlp.seas.harvard.edu/2018/04/03/attention.html>
* The Annotated Transformer by Sasha Rush
* A Jupyter Notebook using PyTorch that explains everything!

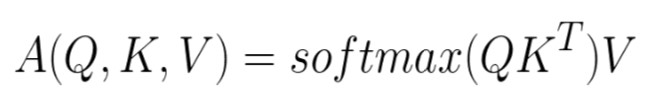
## Dot-Product Attention (Extending our previous def.)

* Inputs: a query q and a set of key-value (k-v) pairs to an output
* Query, keys, values, and output are all vectors
* Output is weighted sum of values, where
* Weight of each value is computed by an inner product of query and corresponding key
* Queries and keys have same dimensionality dk value have dv



# Dot-Product Attention – Matrix notation

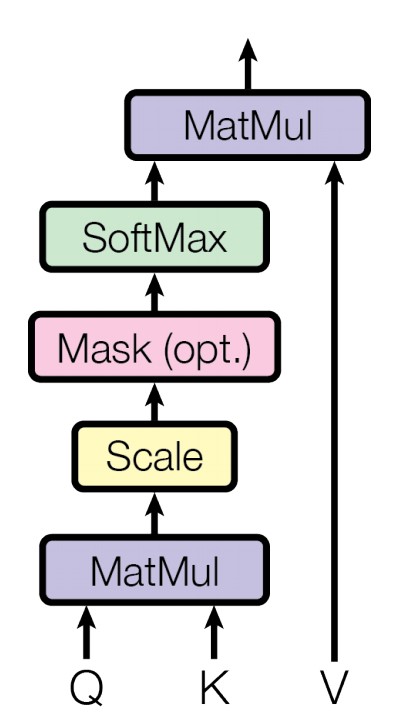
* When we have multiple queries q, we stack them in a matrix Q:
* Becomes:



[|Q| x dk] x [dk x |K|] x [|K| x dv]



# Scaled Dot-Product Attention

* Problem: As dk gets large, the variance of qTk increases à some values inside the softmax get large à the softmax gets very peaked à hence its gradient gets smaller.
* Solution: Scale by length of query/key vectors:



# Self-attention in an encoder

* The input word vectors are the queries, keys and values
* In other words: the word vectors themselves select each other
* Word vector stack = Q = K = V
* They’re separated in the definition so you can different things
* For an NMT decoder, you can do queries from the output with K/V from the encoder

# Multi-head attention

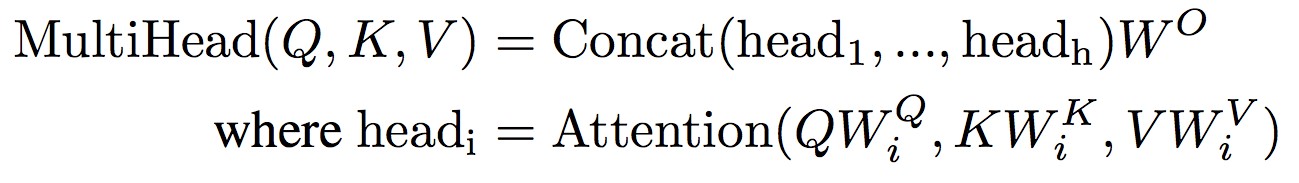
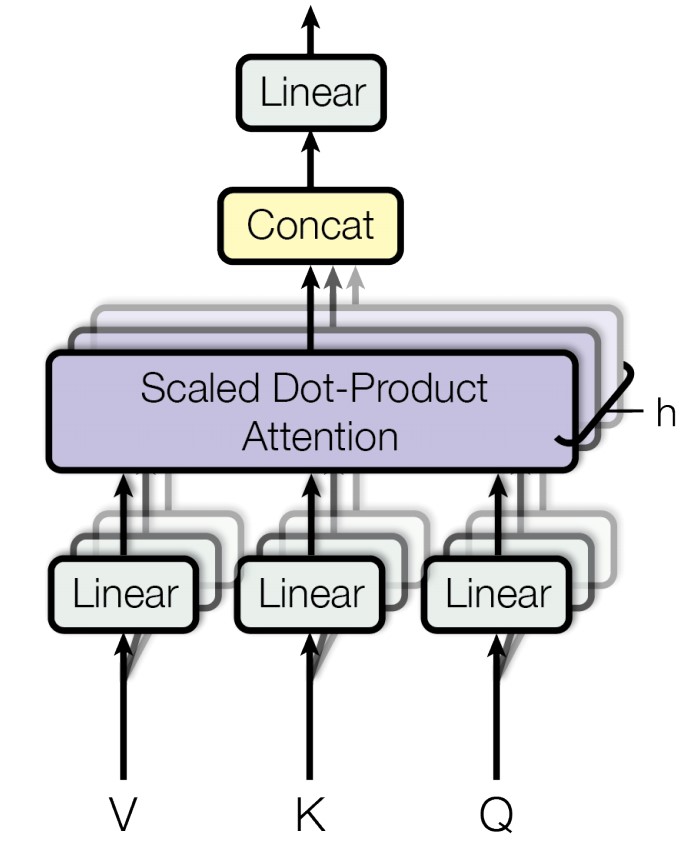
* Problem with simple self-attention:
* Only one way for words to interact with one-another
* Solution: Multi-head attention
* First map Q, K, V into h=8 many lower

dimensional spaces via W matrices

•

Then apply attention, then concatenate

outputs and pipe through linear layer



# Transformer (Vaswani et al. 2017)

Judiciary

Committee

]

MASK

[

Report

]

CLS

[

0

1

2

3

4

*h*

,

0

0

*h*

1

0

,

*h*

2

,

0

*h*

3

,

0

*h*

4

0

,

+

+

+

+

+

V

0

K

0

Q

0

V

1

K

1

Q

1

V

2

K

2

Q

2

V

3

K

3

Q

3

V

4

K

4

Q

4

…

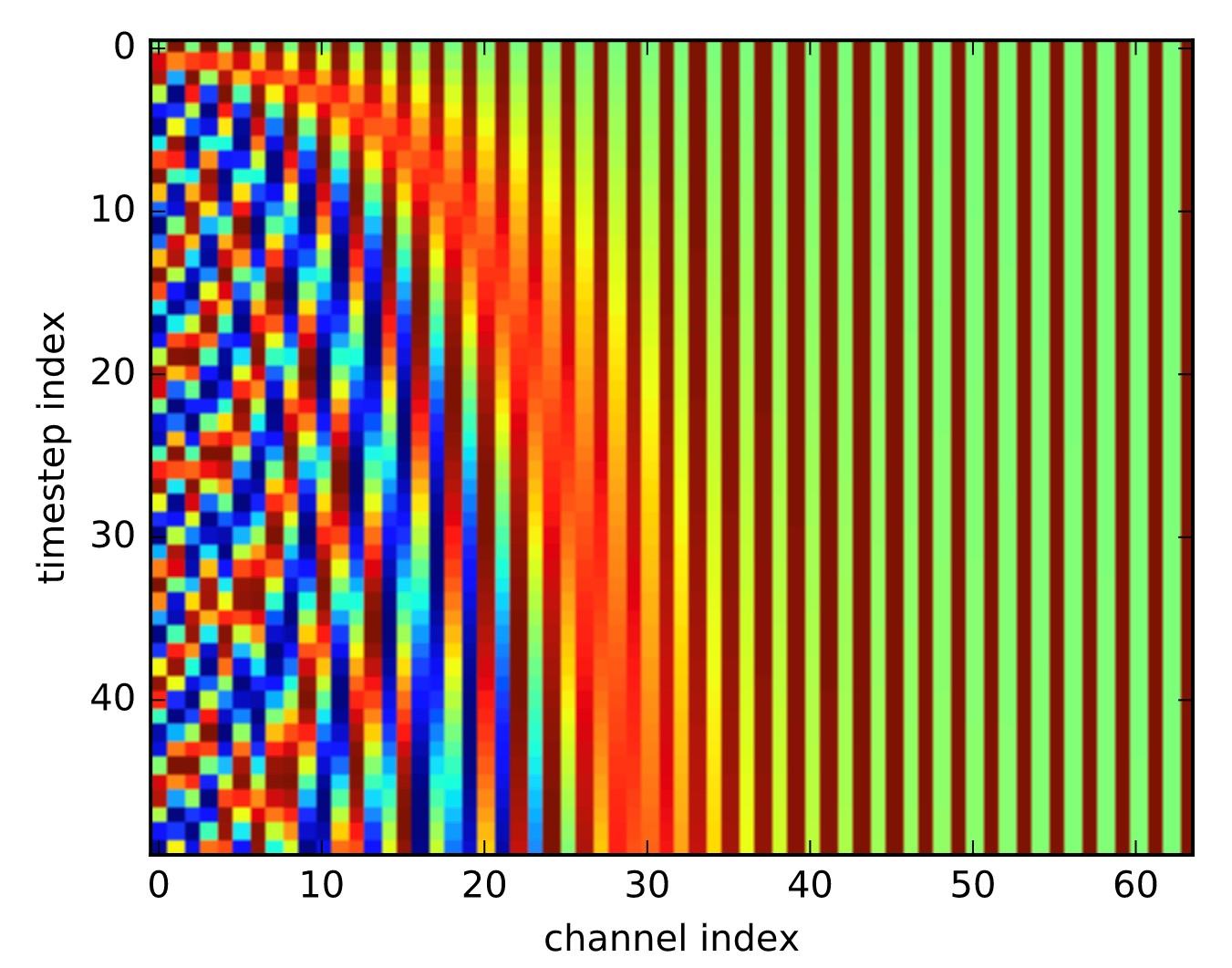
…

*h*

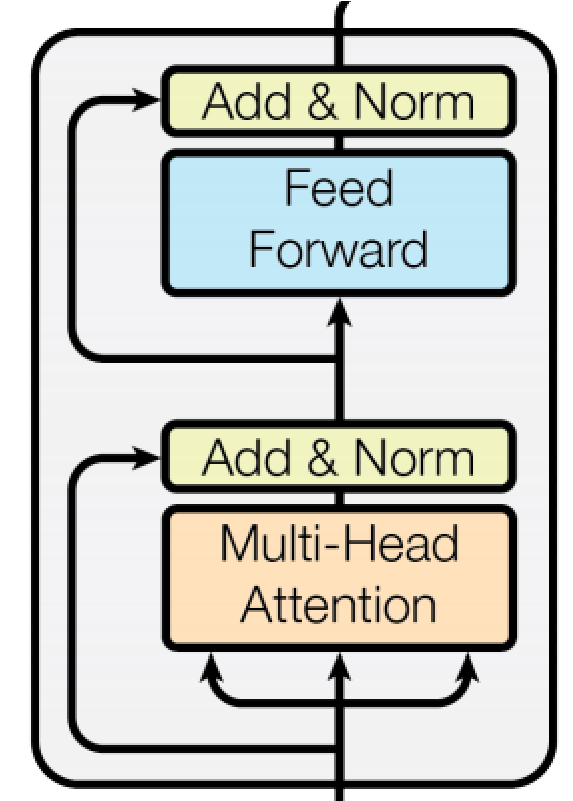
x

# Encoder Input

* Actual word representations are byte-pair encodings
* As in last lecture
* Also added is a **positional encoding** so same words at different locations have different overall representations:

 or learned 

# Complete transformer block

Each block has two “sublayers”

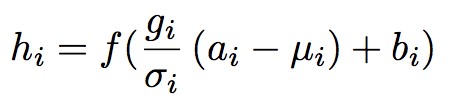
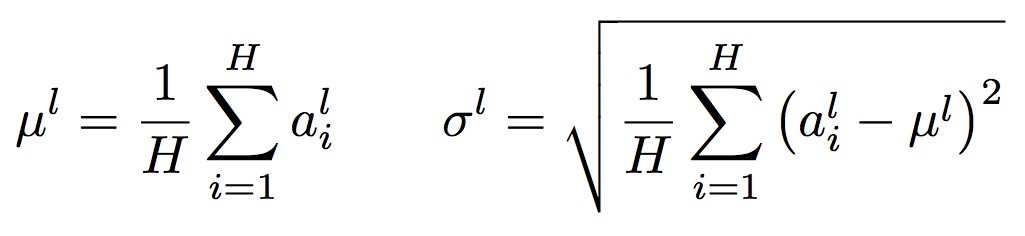
1. Multihead attention
2. 2-layer feed-forward NNet (with ReLU)

Each of these two steps also has:

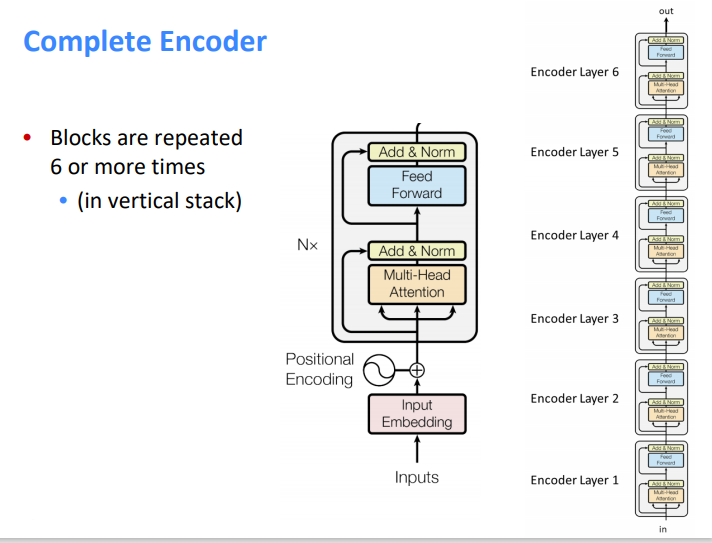
Residual (short-circuit) connection and LayerNorm

LayerNorm(x + Sublayer(x))

Layernorm changes input features to have mean 0 and variance 1 per layer (and adds two more parameters)

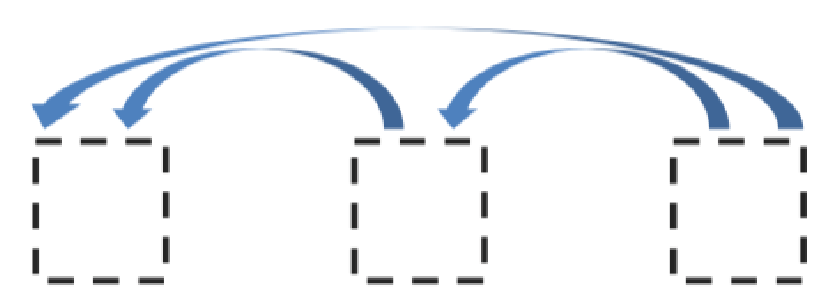


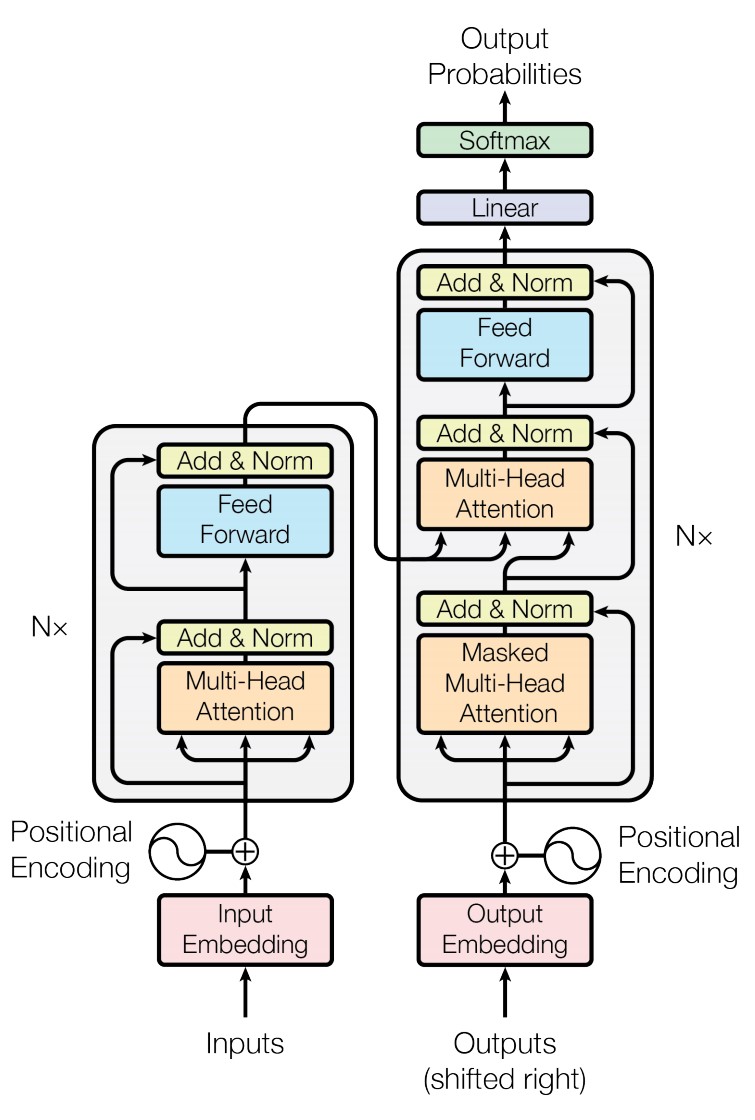
Layer Normalization by Ba, Kiros and Hinton, <https://arxiv.org/pdf/1607.06450.pdf>

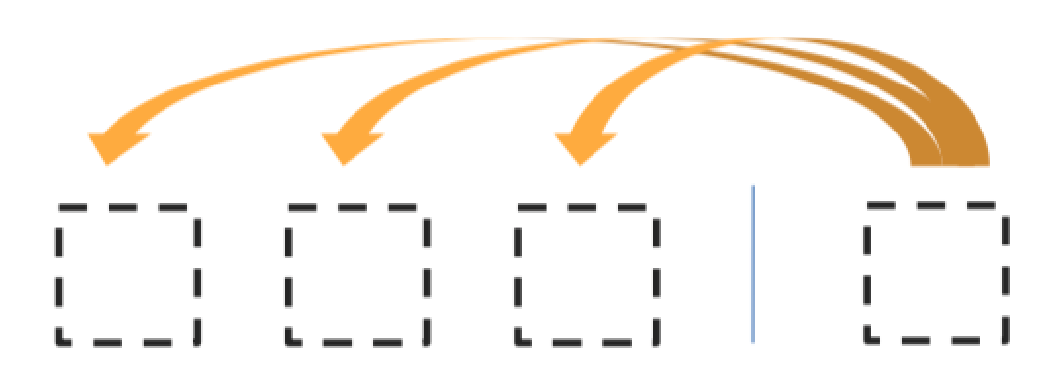


# Transformer Decoder

* 2 sublayer changes in decoder
* Masked decoder self-attention on previously generated outputs:

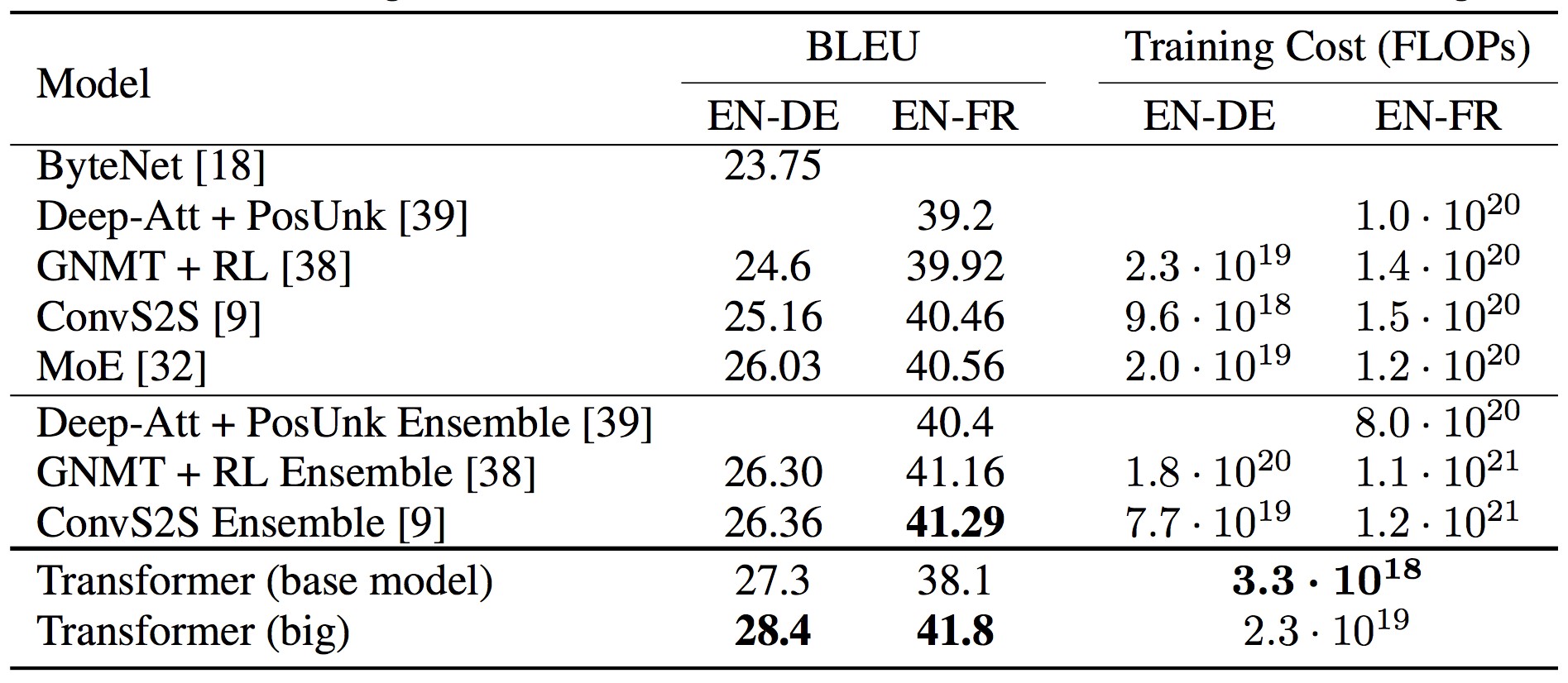


* Encoder-Decoder Attention, where queries come from previous decoder layer and keys and values come from output of encoder



•44 Blocks repeated 6 times also

# Experimental Results for MT



## Some performance numbers: LM on WikiText-103

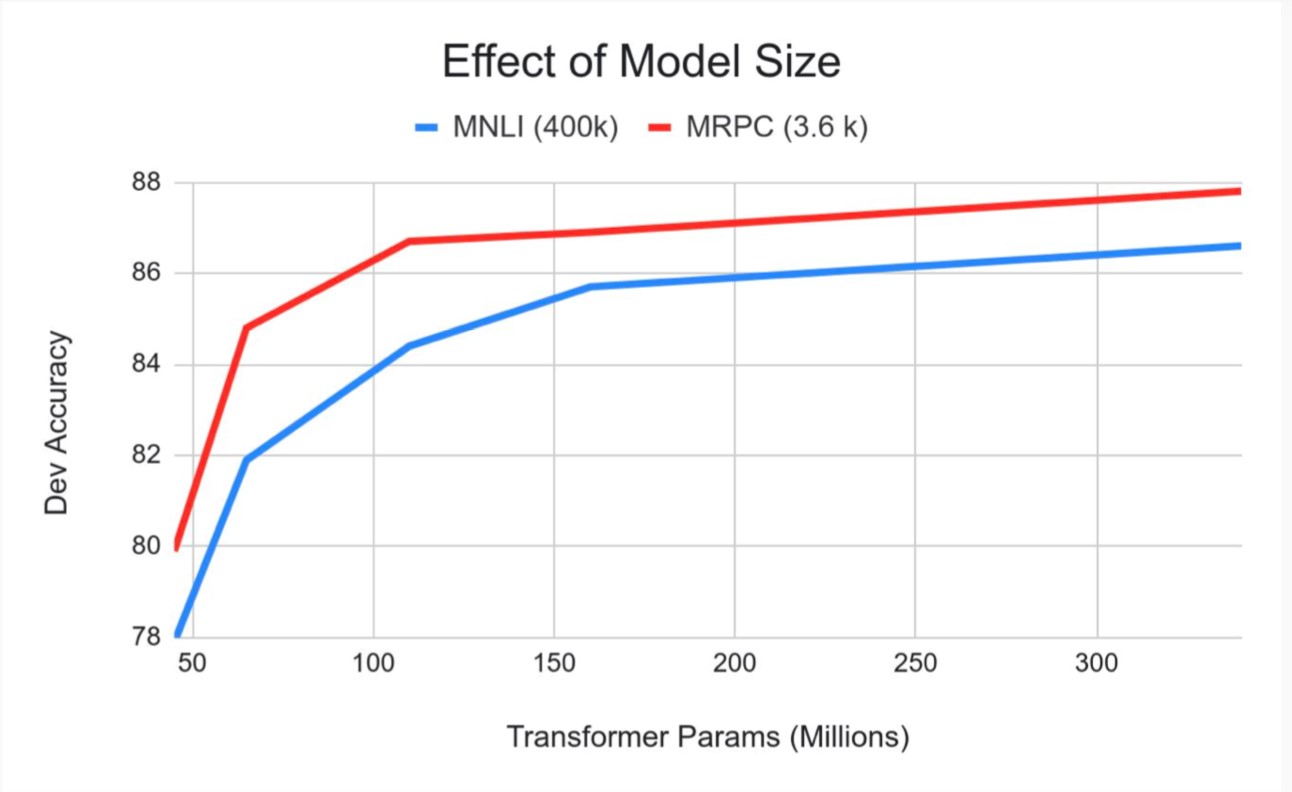
|  |  |  |
| --- | --- | --- |
| **Model** | **# Params** | **Perplexity** |
| Grave et al. (2016) – LSTM |  | 48.7 |
| Grave et al. (2016) – LSTM with cache |  | 40.8 |
| 4-layer QRNN (Merity et al. 2018) | 151M | 33.0 |
| LSTM + Hebbian + Cache + MbPA (Rae et al.) | 151M | 29.2 |
| **Transformer-XL Large (Dai et al. 2019)** | **257M** | **18.3** |
| GPT-2 Large\* (Radford et al. 2019) | 1.5B | 17.5 |

(For gray haired people)

A perplexity of 18 for Wikipedia text is just stunningly low!

# Size matters

* Going from 110M to 340M parameters helps a lot
* Improvements have not yet asymptoted



# 5. BERT: Devlin, Chang, Lee, Toutanova (2018)

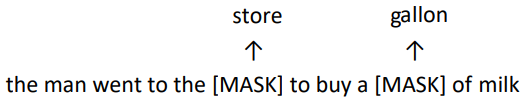
BERT (Bidirectional Encoder Representations from Transformers):

Pre-training of Deep Bidirectional Transformers for Language

Understanding, which is then fine-tuned for a task

Want: truly bidirectional information flow without leakage in a deep model

Solution: Use a cloze task formulation where 15% of words are blanked out and predicted:



# BERT sentence pair encoding

Token embeddings are word pieces

Learned segmented embedding represents each sentence

Positional embedding is as for other Transformer architectures

49

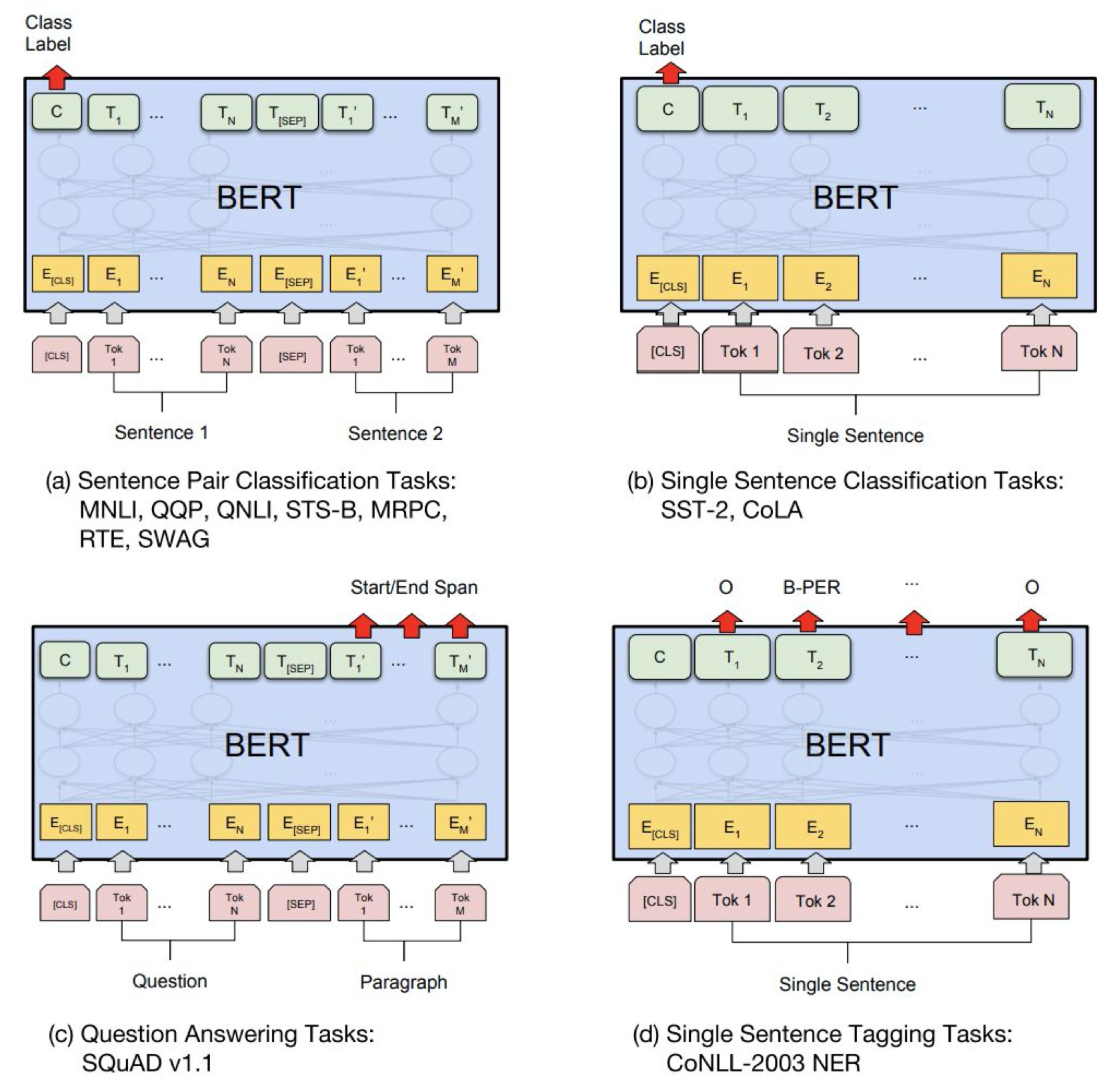
# BERT model architecture and training

* Transformer encoder (as before)
* Self-attention ⇒ no locality bias
  + Long-distance context has “equal opportunity”
* Single multiplication per layer ⇒ efficiency on GPU/TPU
* Train on Wikipedia + BookCorpus
* Train 2 model sizes:
  + BERT-Base: 12-layer, 768-hidden, 12-head
  + BERT-Large: 24-layer, 1024-hidden, 16-head
* Trained on 4x4 or 8x8 TPU slice for 4 days

# BERT model fine tuning

• Simply learn a classifier built on the top layer for each task that you fine tune for

# BERT model fine tuning



## CoNLL 2003 Named Entity Recognition (en news testb)

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **Year** | **F1** |
| [Flair (Zalando)](https://github.com/zalandoresearch/flair) | Character-level language model | 2018 | 93.09 |
| BERT Large | Transformer bidi LM + fine tune | 2018 | 92.8 |
| CVT Clark | Cross-view training + multitask learn | 2018 | 92.61 |
| BERT Base | Transformer bidi LM + fine tune | 2018 | 92.4 |
| ELMo | ELMo in BiLSTM | 2018 | 92.22 |
| TagLM Peters | LSTM BiLM in BiLSTM tagger | 2017 | 91.93 |
| Ma + Hovy | BiLSTM + char CNN + CRF layer | 2016 | 91.21 |
| Tagger Peters | BiLSTM + char CNN + CRF layer | 2017 | 90.87 |
| Ratinov + Roth | Categorical CRF+Wikipeda+word cls | 2009 | 90.80 |
| Finkel et al. | Categorical feature CRF | 2005 | 86.86 |
| IBM Florian | Linear/softmax/TBL/HMM ensemble, gazettes++ | 2003 | 88.76 |
| Stanford | MEMM softmax markov model | 2003 | 86.07 |

**AllenAI ARISTO: Answering Science Exam Questions**

From ‘F’ to ‘A’ on the N.Y. Regents Science Exams: An Overview of the Aristo Project. Peter Clark, Oren Etzioni, Daniel Khashabi, Tushar Khot, Bhavana Dalvi Mishra, Kyle

Richardson, Ashish Sabharwal, Carissa Schoenick, Oyvind Tafjord, Niket Tandon, Sumithra Bhakthavatsalam, Dirk Groeneveld, Michal Guerquin, Michael Schmitz

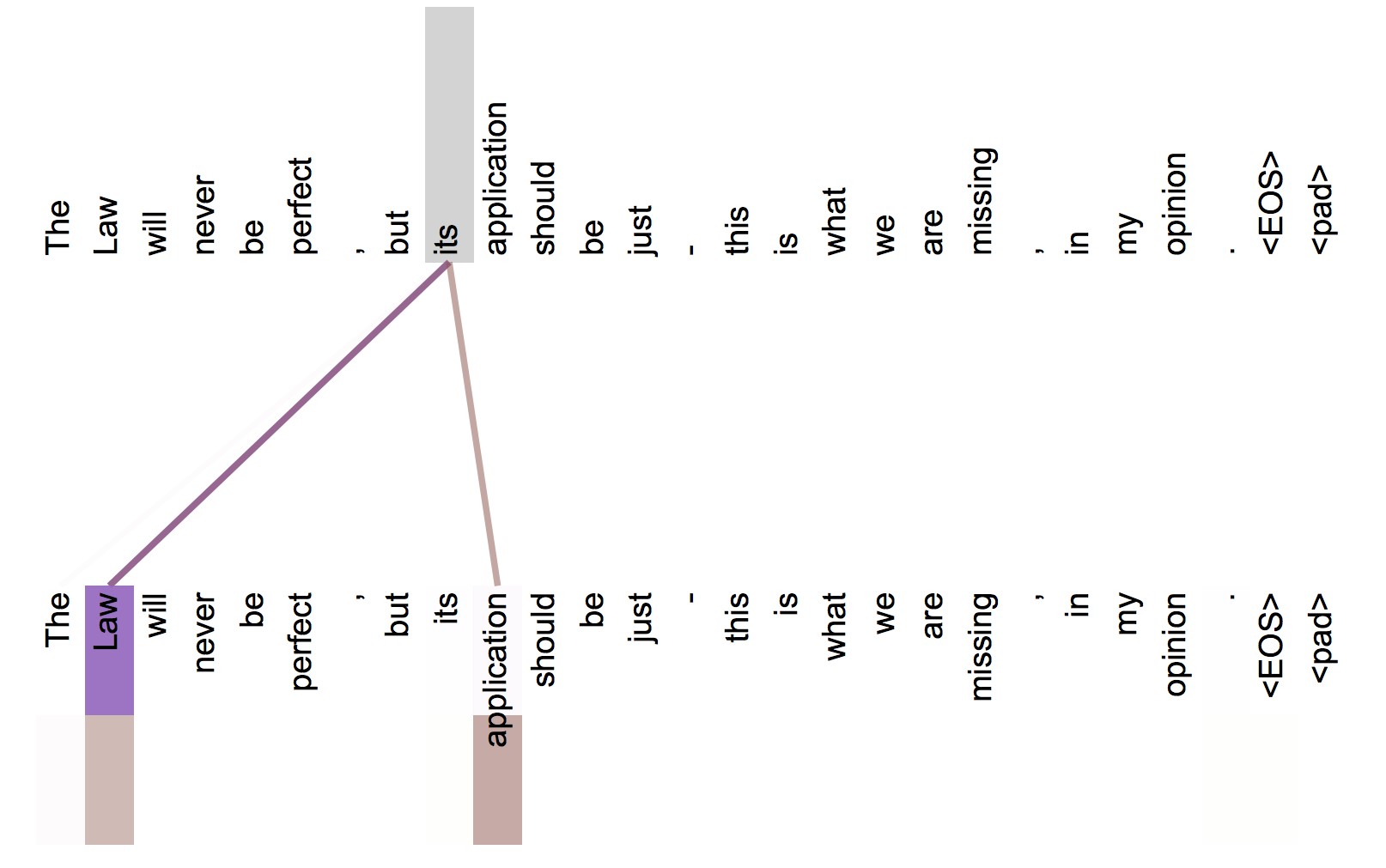
Which equipment will best separate a mixture of iron filings and black pepper? **(1)** magnet **(2)** filter paper **(3)** triplebeam balance **(4)** voltmeter Which process in an apple tree primarily results from cell division?

**(1)** growth **(2)** photosynthesis **(3)** gas exchange **(4)** waste removal

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Set** | **IR** | **TupInf** | **Multee** | **AristoBERT** | **AristoRoBERTa** | **ARISTO** |
| Regents 4th | 64.5 | 63.5 | 69.7 | 86.2 | 88.1 | **89.9** |
| Regents 8th | 66.6 | 61.4 | 68.9 | 86.6 | 88.2 | **91.6** |
| Regents 12th | 41.2 | 35.4 | 56.0 | 75.5 | 82.3 | **83.5** |
| ARC-Challenge | 0.0 | 23.7 | 37.4 | 57.6 | **64.6** | 64.3 |

## Attention visualization: Implicit anaphora resolution

Words start to pay attention to other words in sensible ways



In 5th layer. Isolated attentions from just the word ‘its’ for attention heads 5 and 6.

55 Note that the attentions are very sharp for this word.

**6. How’s the weather?**

## Rapid Progress from Pre-Training (GLUE benchmark)

90

60

ELMo

GPT

BERT

-

Base

BERT

-

Large

XLNet

RoBERTa

ALBERT

GloVe

GLUE Score

Over 3x reduction in error in 2 years, “superhuman” performance

**Yay! We now have strongly performing, deep, generic, pre-trained, neural network stacks for NLP that you can just load**



57

**– in the same way vision has had for 5 years (ResNet, etc.)!**

## But let’s change the x-axis to compute …

90

60

ELMo

GPT

BERT

-

Base

BERT

-

Large

GloVe

GLUE Score

6.4e19

FLOPs

1.9e20

FLOPs

Pre-Train FLOPs

BERT-Large uses 60x more compute than ELMo

## But let’s change the x-axis to compute …

90

60

≈

ç

ELMo

GPT

BERT

-

Base

BERT

-

Large

XLNet

RoBERTa

GloVe

GLUE Score

Pre-Train FLOPs

RoBERTa uses 16x more compute than BERT-Large

**More compute, more better?**

90

60

≈

ç

ELMo

GPT

BERT

-

Base

BERT

-

Large

XLNet

RoBERTa

ALBERT

GloVe

GLUE Score

≈

ç

Pre-Train FLOPs

ALBERT uses 10x more compute than RoBERTa

**The climate cost of modern deep learning**

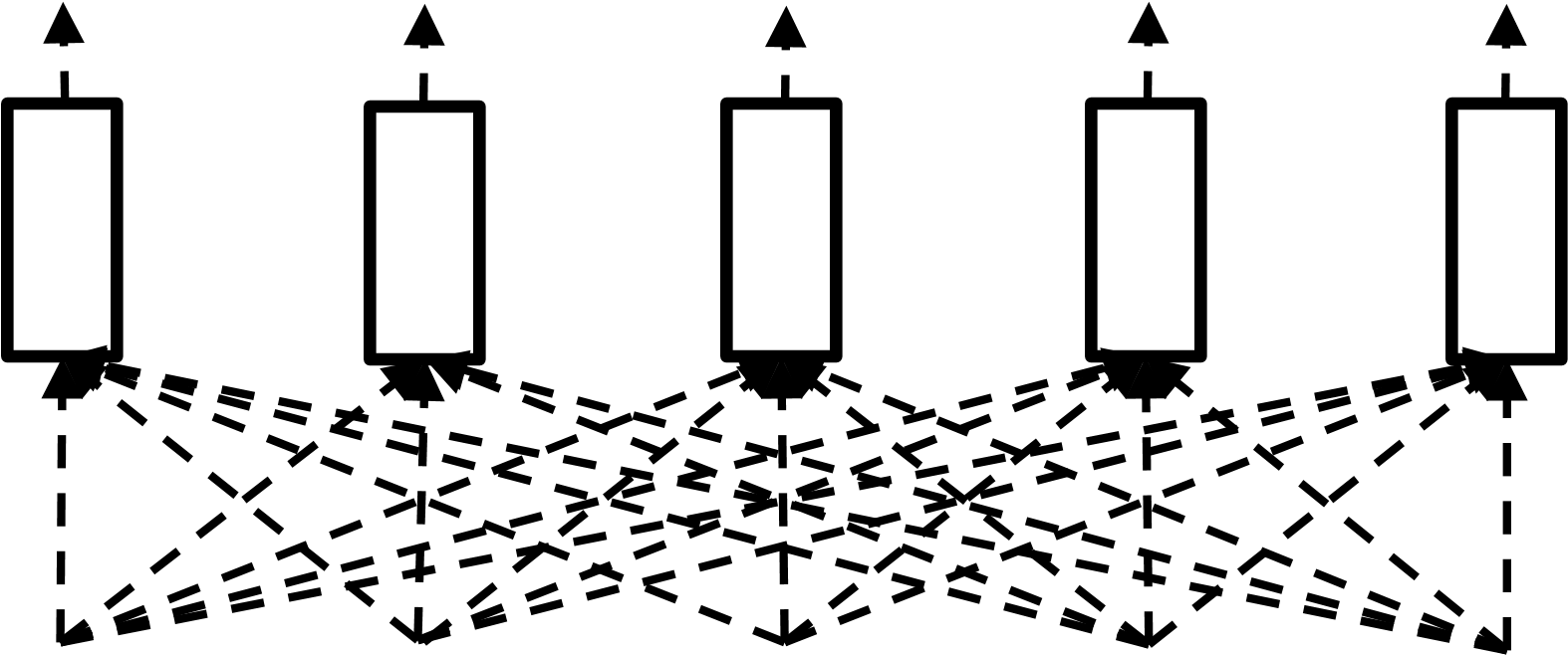


61

**ELECTRA: “Efficiently Learning an Encoder to**

## Classify Token Replacements Accurately” Clark, Luong, Le, and Manning (2020)

Bidirectional model but learn from all tokens original replaced original original replaced



the painter sold the car

## Generating Replacements

Plausible alternatives come from small masked language model (the “generator”) trained jointly with ELECTRA

sample

artist

sold

the

the

car

[

MASK

]



artist

sold

the

artist

[

MASK

]



Generator

(

typically

a

small MLM)

original

original

original

original

replaced

Discriminator

(

ELECT

RA

)

sample

artist

sold

the

the

painting

MLM Loss Binary classification loss

## Results: GLUE Score vs Compute

≈

ç

ELMo

GPT

BERT

-

Base

XLNet

RoBERTa

GloVe

BERT

-

Large

EL

-

Small

EL

-

Base

EL

-

Large

EL

-

Large

100

steps

k

Pre-Train FLOPs