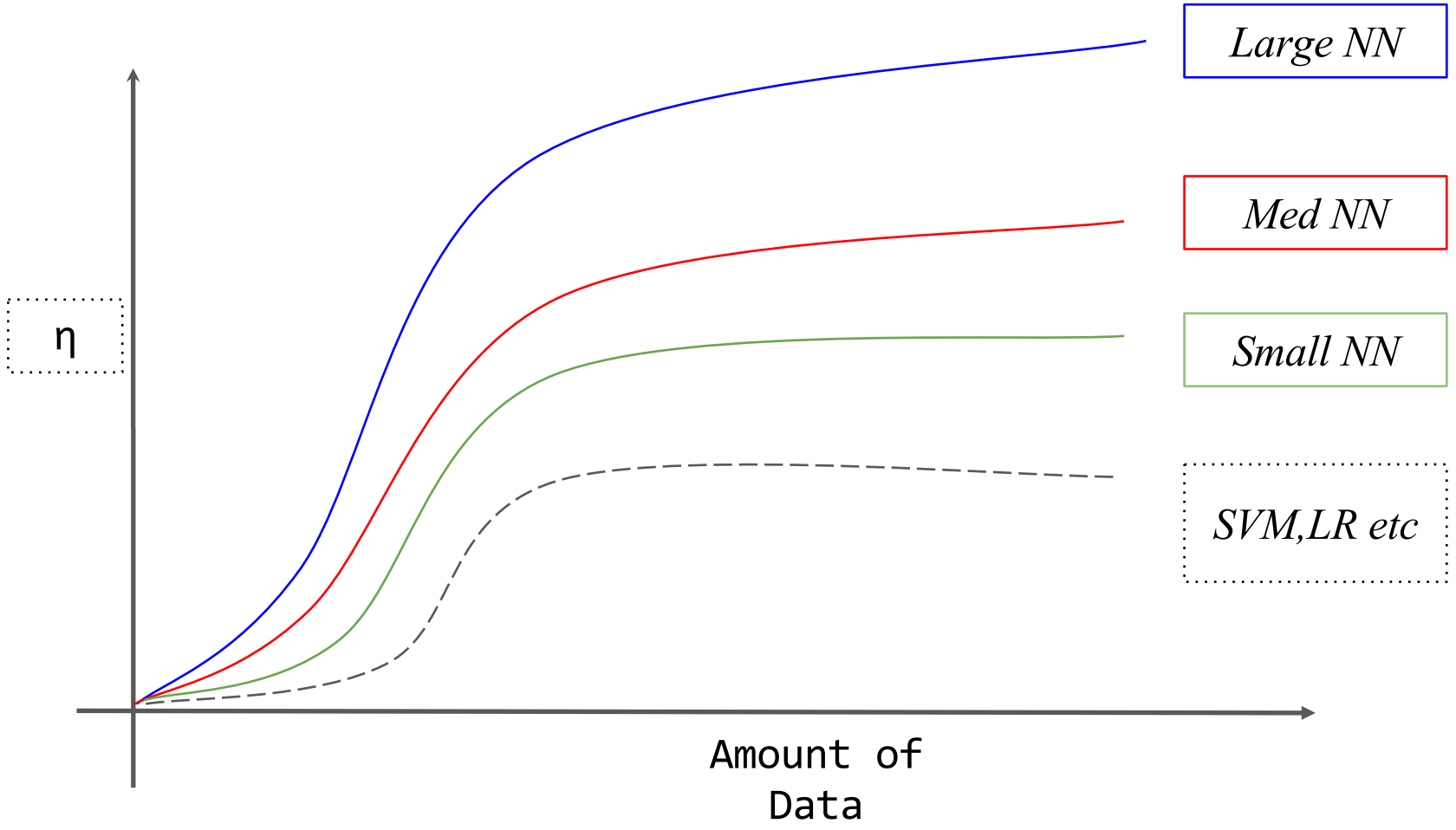
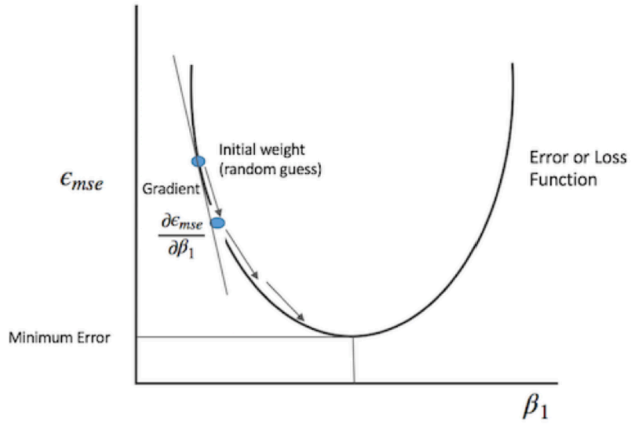
Deep Learning using Keras

Manaranjan Pradhan



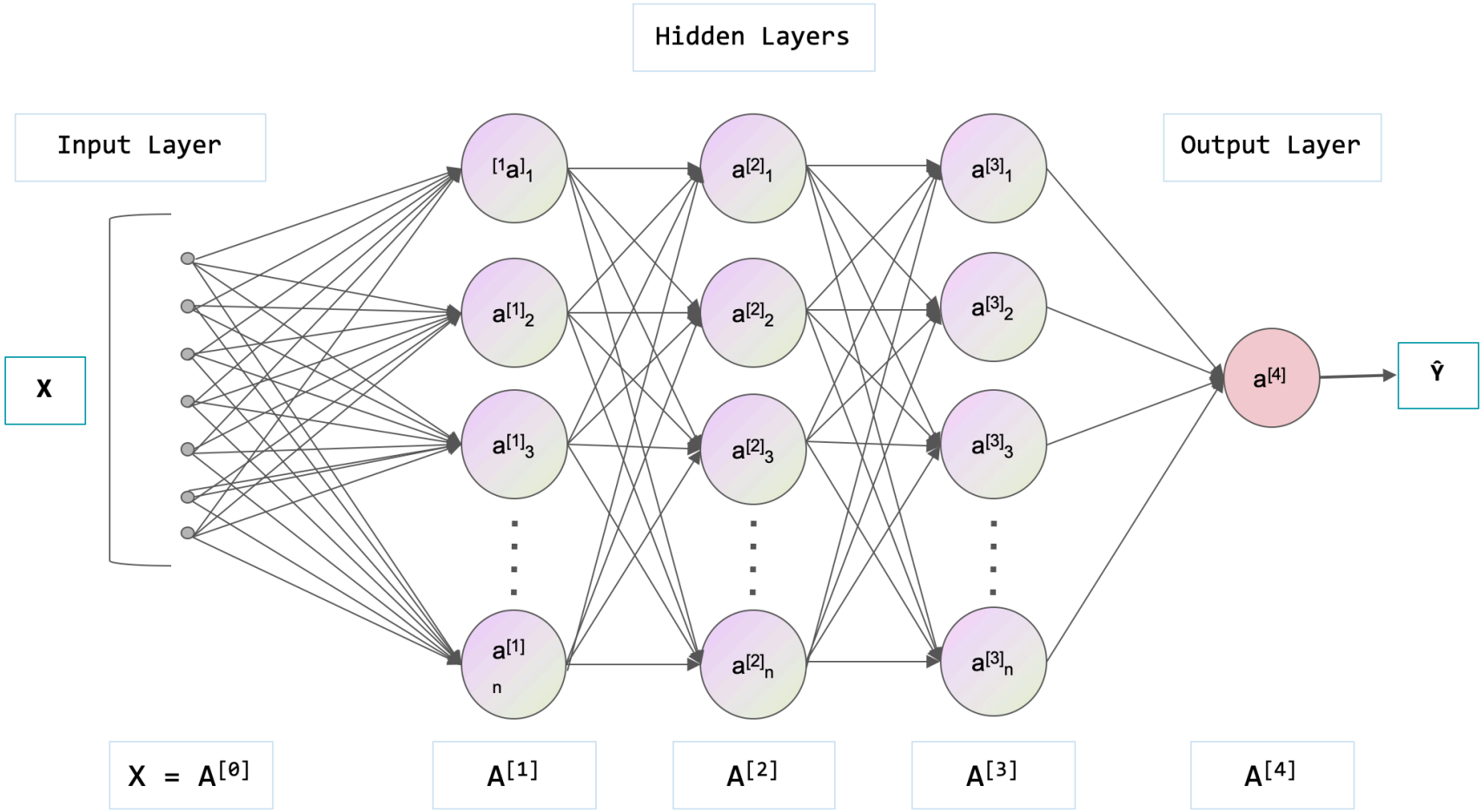
Why does Deep learning work?

Gradient Descent



Beta Updates	Beta Derivatives
$\beta_0 = \beta_0 - \alpha * \frac{\partial \epsilon_{mse}}{\partial \beta_0}$	$\frac{\partial \epsilon_{mse}}{\partial \beta_0} = \frac{2}{N} \sum_{i=1}^{n} (Y_i - (\beta_0 + \beta_1 X_i)) = \frac{2}{N} \sum_{i=1}^{n} (Y_i - \hat{Y})$
$\beta_1 = \beta_1 - \alpha * \frac{\partial \epsilon_{mse}}{\partial \beta_1}$	$\frac{\partial \epsilon_{mse}}{\partial \beta_1} = \frac{2}{N} \sum_{i=1}^{n} (Y_i - (\beta_0 + \beta_1 X_i)) * X_i = \frac{2}{N} \sum_{i=1}^{n} (Y_i - \hat{Y}) * X_i$

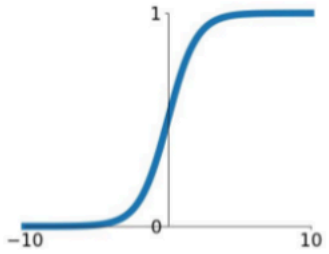
Neural Network Architecture



Activations

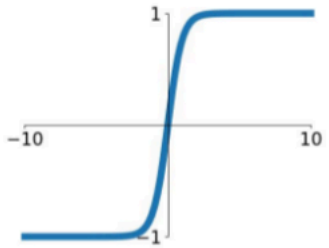
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



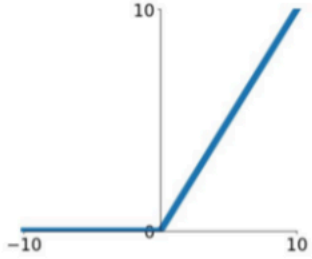
tanh

$$\tanh(x)$$



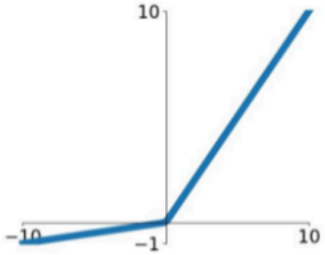
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

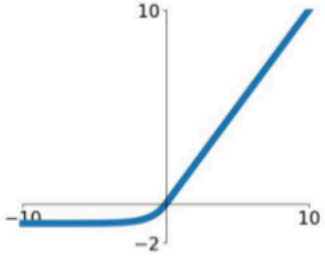


Maxout

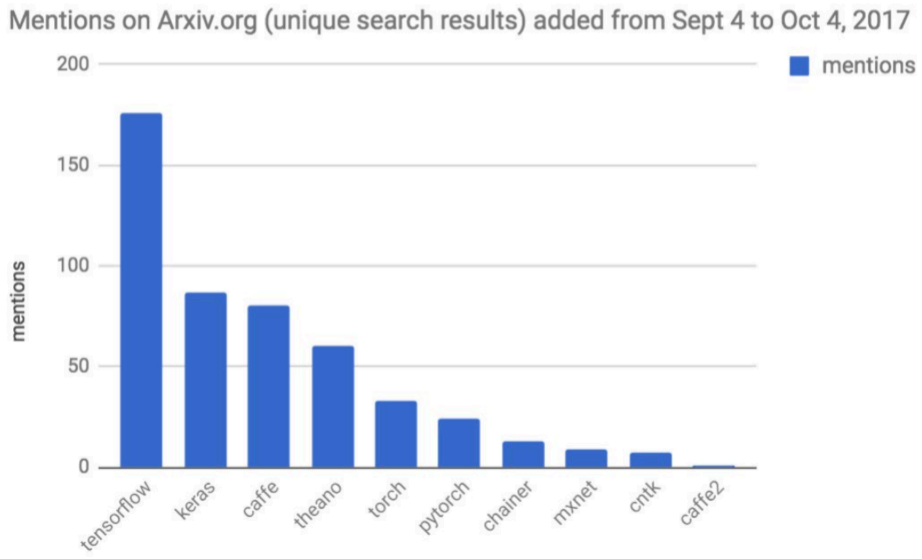
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Deep Learning Platforms



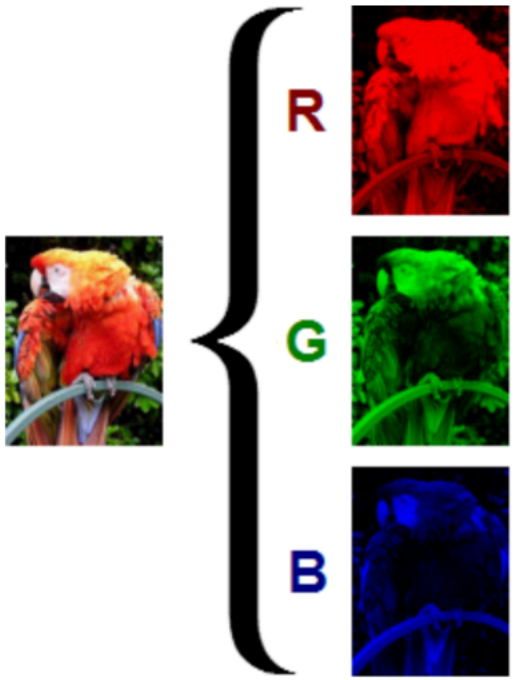
- Anaconda 5.0

  https://www.anaconda.com/download/#macos

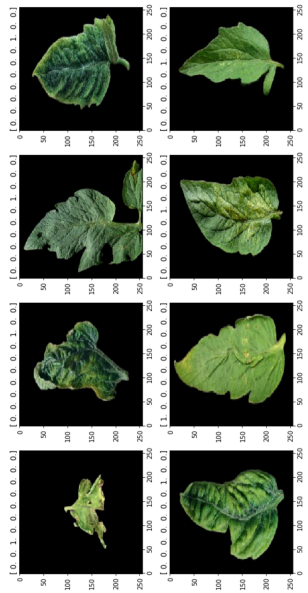
- Python 3.5
- Tensorflow 1.4
- Keras 2.1
- OpenCV2
- Jupyter Notebook is an editor

Dealing with Images

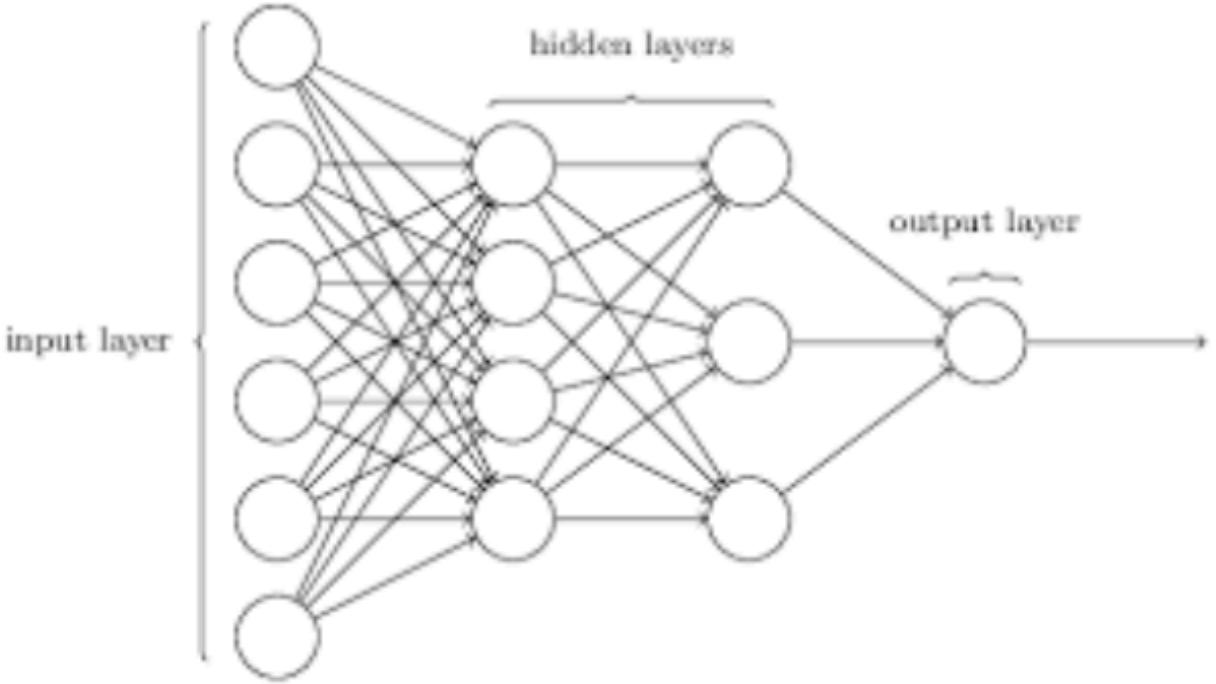
- Images are matrixes with 3 dimensions
  - height x width x channels
  - height x width depends on number of pixels
  - There are 3 channels – RGB
- High Dimensional Dataset (Image Size: 256x256x3)
  - Feature Set is very large
- Dimensionality Reduction is an alternate
  - But loss of information
  - Have other problems
- Convolution Neural Network (CNN) is most widely used
  - http://yann.lecun.com/exdb/lenet/
  - [LeCun et al., 1998]



Too many parameters to train



256x256x3 = 196608 input pixels



Number of weights to be trained only for first hidden layer: **196608 X 256 = 50,331,648**

Identify the Images

- Can identify images from the edges
- All pixel values may not be required
- But how to extract these features like edges, lines etc. from the images?

Extracting Features using Filters

- Applying ***Convolution Filters***
- A filter is either 3x3 or 5x5 matrix
- Predefined filters to detect features like edges, lines etc.

-1	-1	-1
-1	8	-1
-1	-1	-1

This is an Edge Filter

In Convolution Neural Networks, the filters are learnt by the neural networks. And typically multiple filters are used.

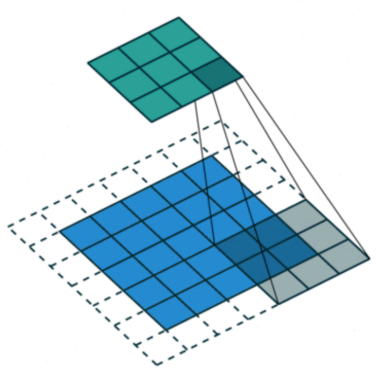
Applying Filters

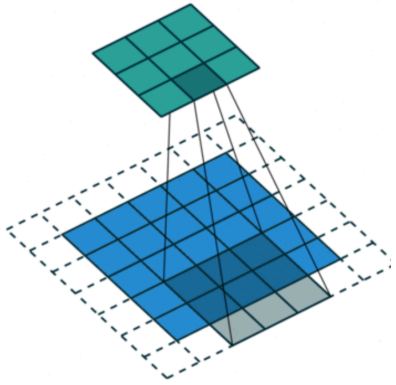
3_0	3_1	2_2	1	0
0_2	0_2	1_0	3	1
3_0	1_1	2_2	2	3
2	0	0	2	2
2	0	0	0	1

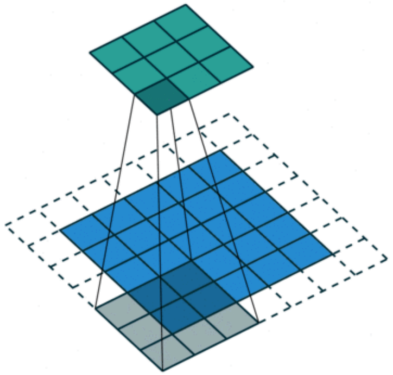
12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3_0	2_1	1_2	0
0	0_2	1_2	3_0	1
3	1_0	2_1	2_2	3
2	0	0	2	2
2	0	0	0	1

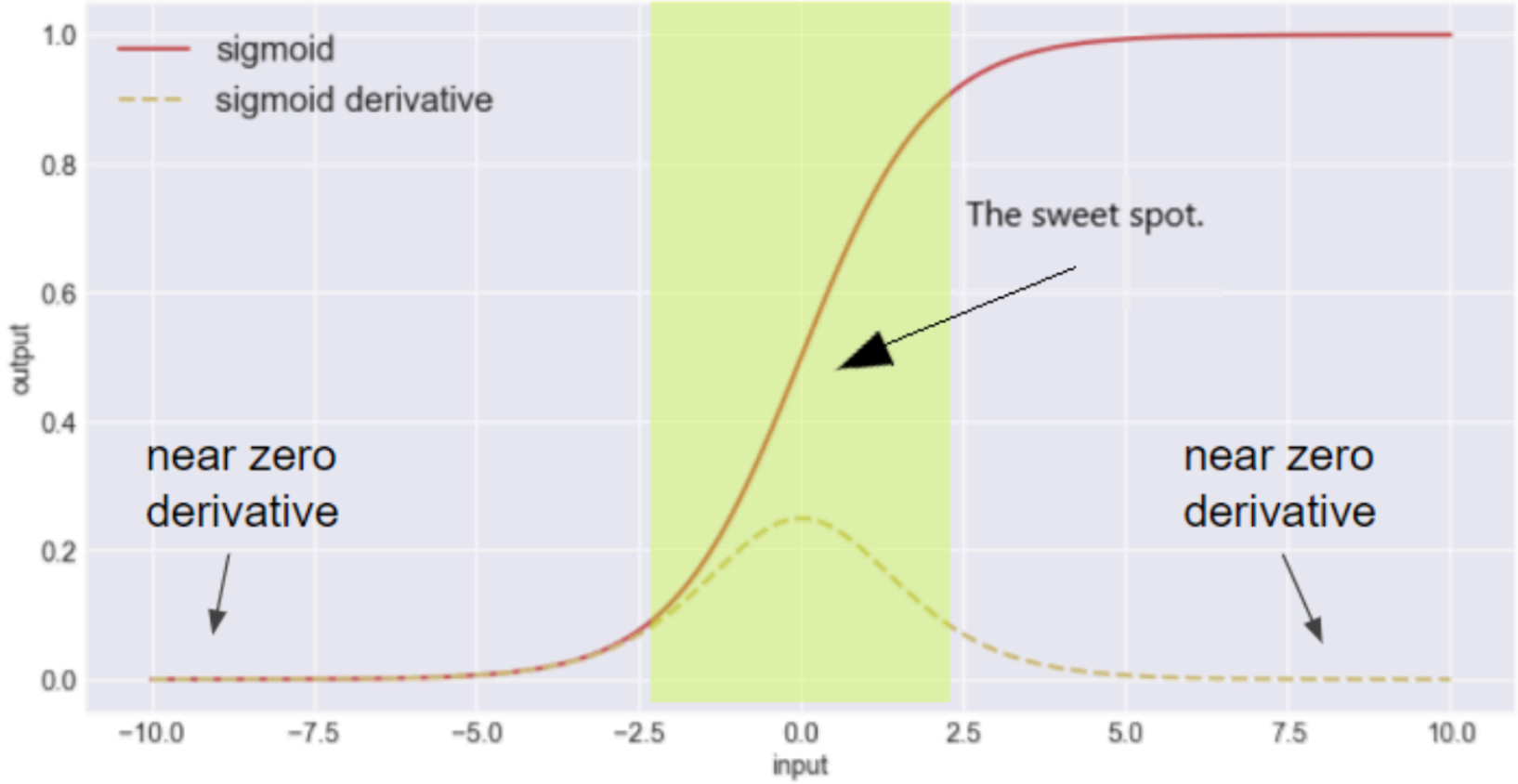
12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0







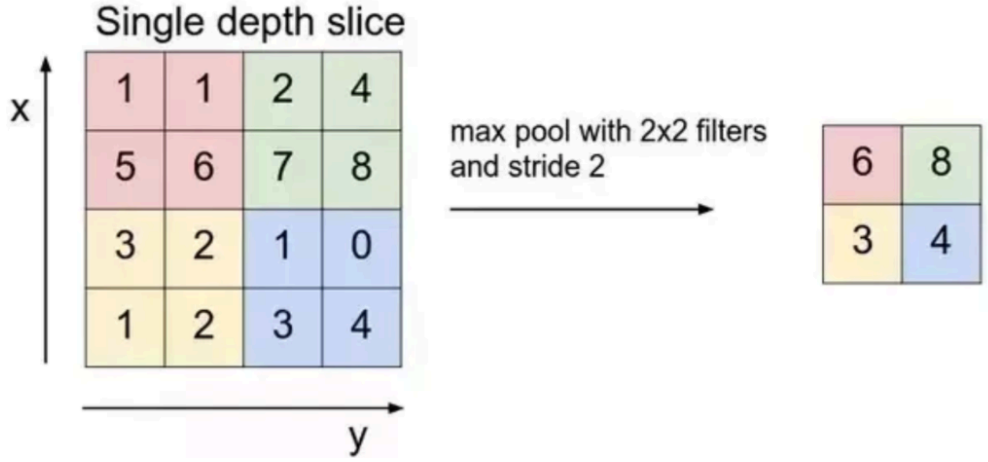
Vanishing Gradient Problem



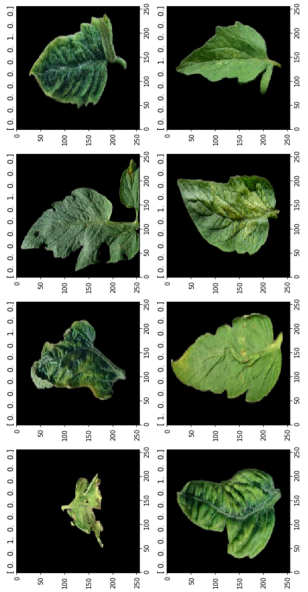
The Sigmoid Sweet Spot

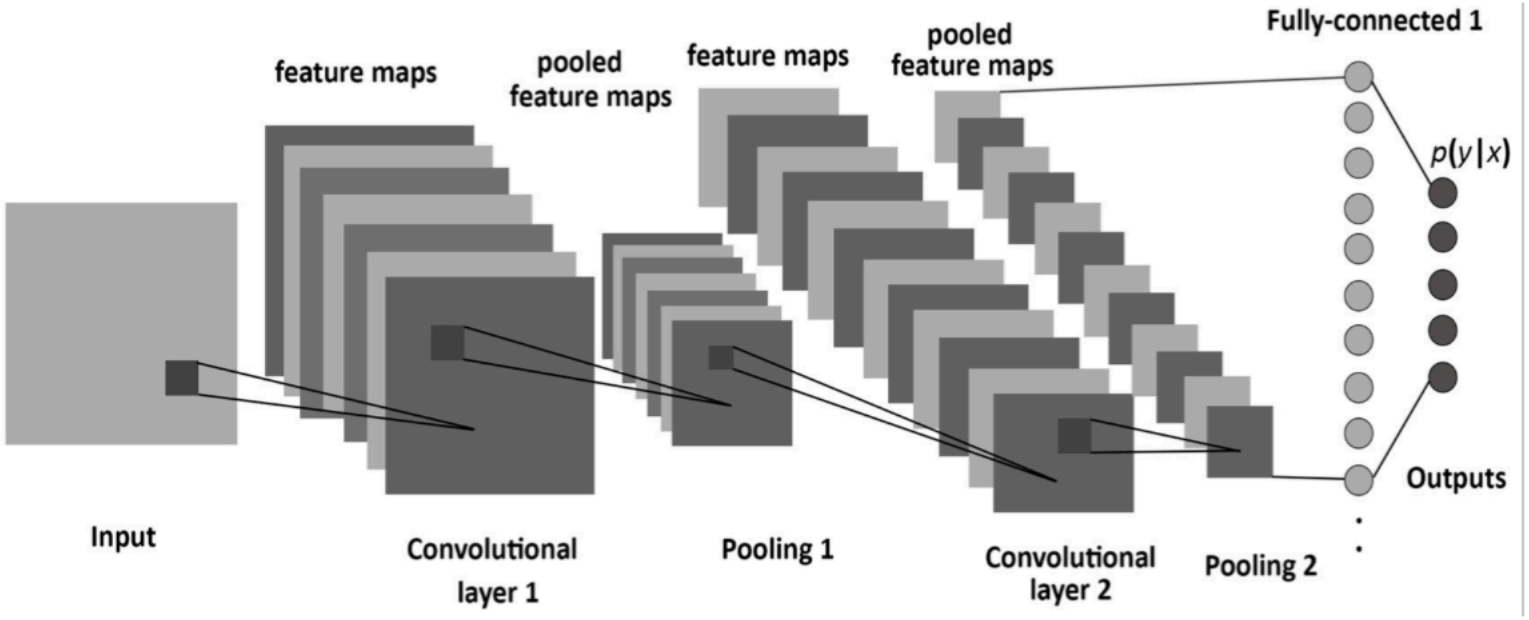
Max Pooling helps Translation Invariance

- Typically applied after convolution filters.
- Reduces input size significantly

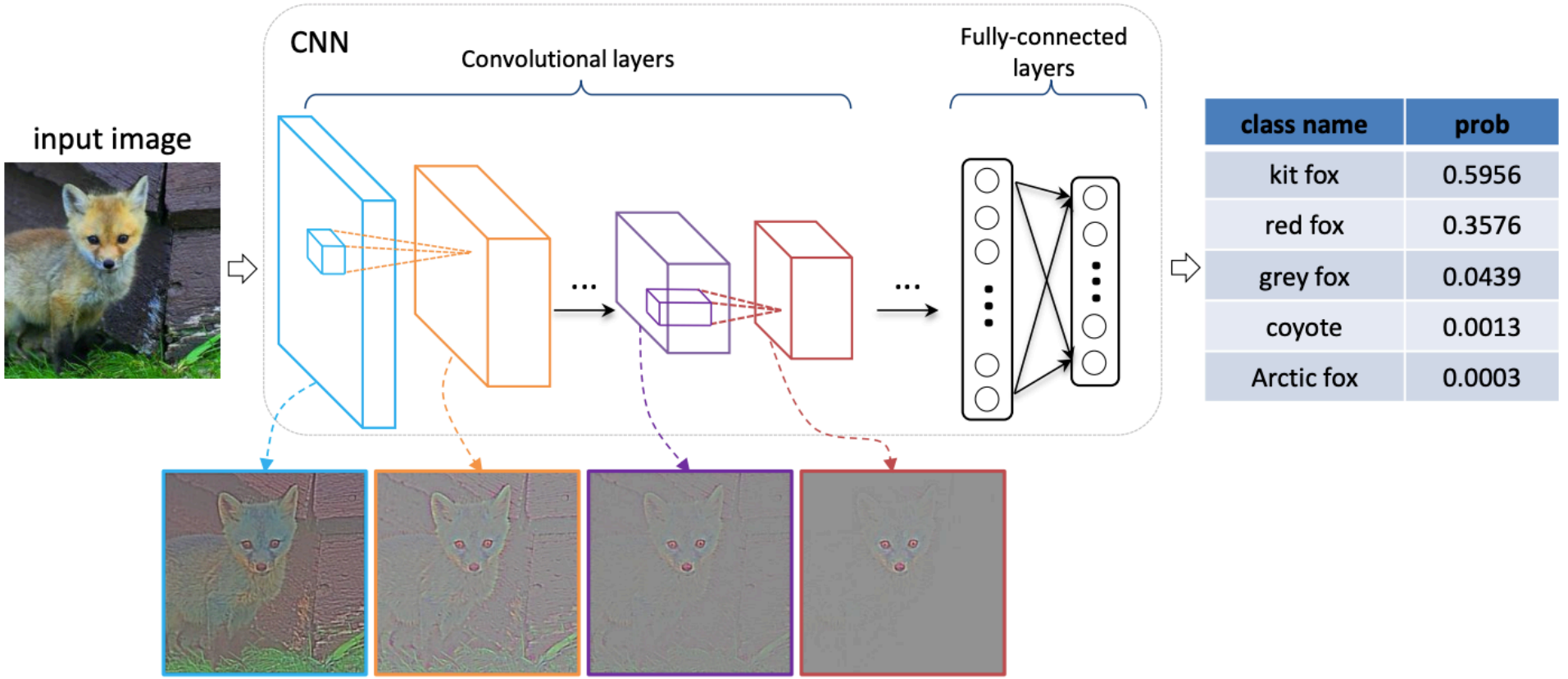


Convolution Neural Network Architecture





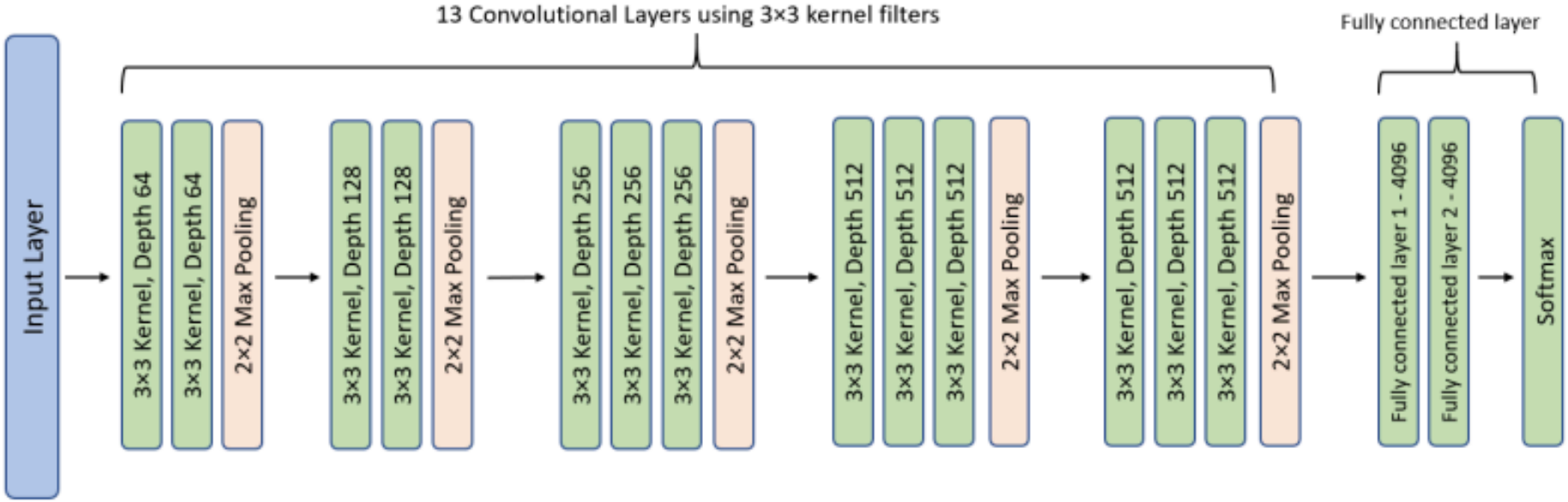

```
Bacterial_spot: 0
Early_blight: 1
Late_blight: 2
Leaf_Mold: 3
Septoria_leaf_spot: 4
Spider_mites: 5
Target_Spot: 6
Leaf_Curl_Virus: 7
Tomato_mosaic_virus: 8
healthy: 9
```



class name	prob
kit fox	0.5956
red fox	0.3576
grey fox	0.0439
coyote	0.0013
Arctic fox	0.0003

Paper: Visualizing and Comparing AlexNet and VGG using Deconvolutional Layers

VGG-16

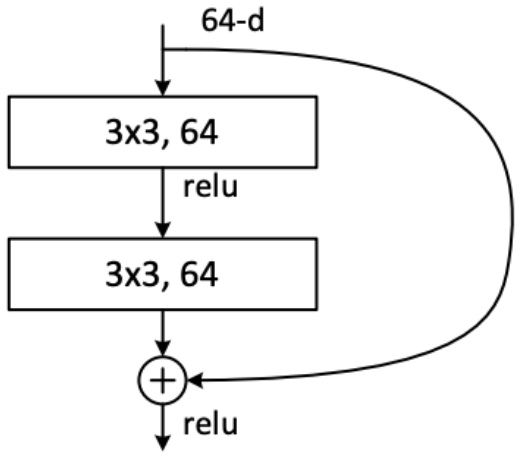


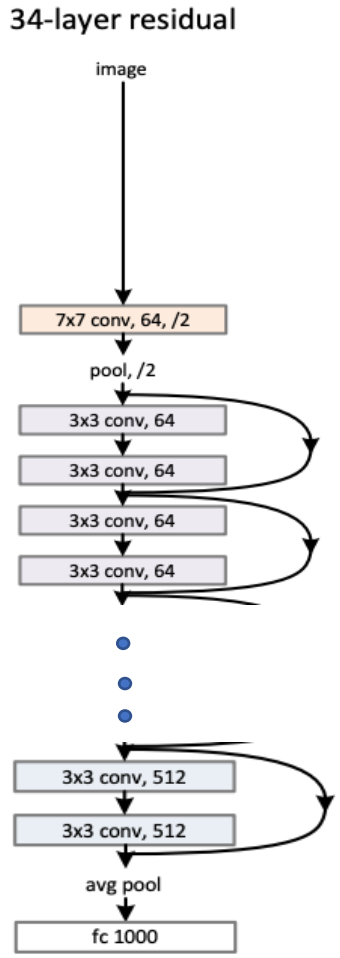
VGG-16 model architecture – 13 convolutional layers and 2 Fully connected layers and 1 Softmax classifier

http://dx.doi.org/10.29322/IJSRP.9.10.2019.p9420

ResNet50

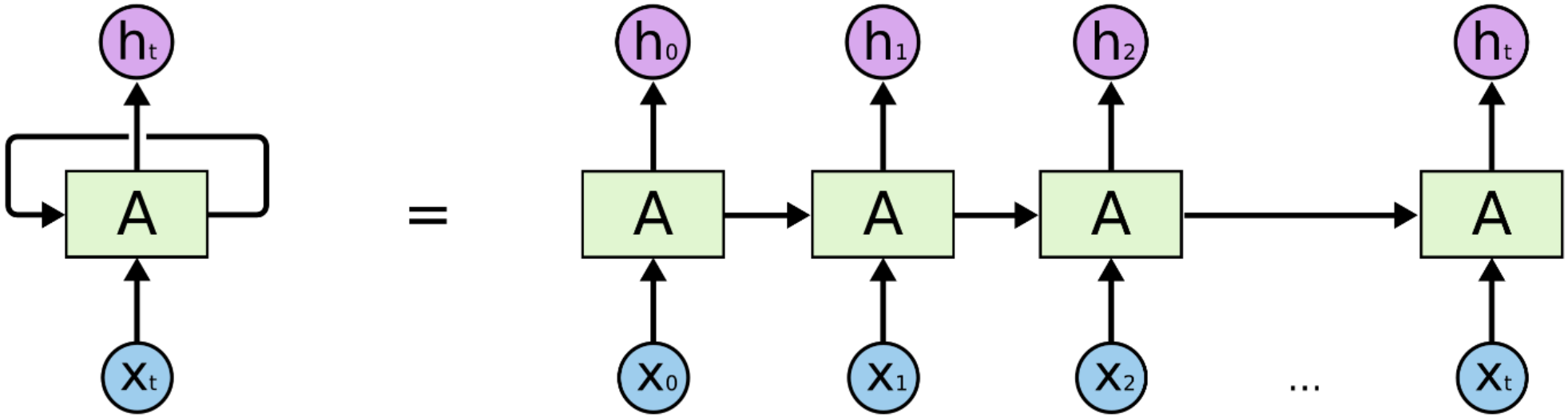
ResNet50 won the top position at the ILSVRC 2015 classification competition with an error of only **3.57%**. Additionally, it also came first in the ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation in the ILSVRC & COCO competitions of 2015.





https://arxiv.org/pdf/1512.03385.pdf

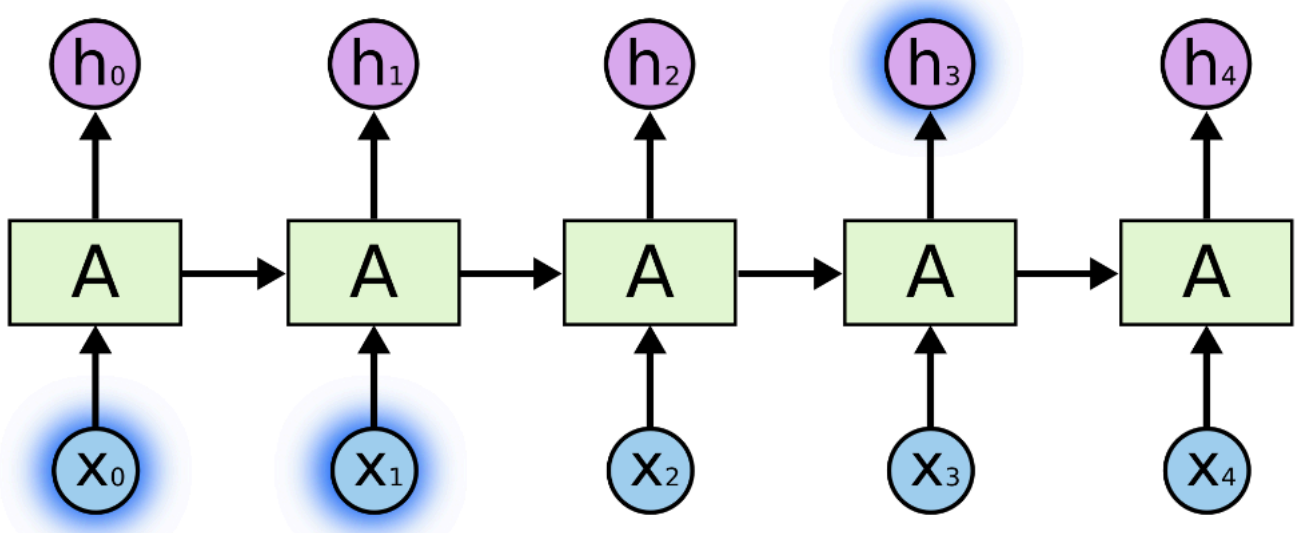
RNN



An unrolled recurrent neural network.

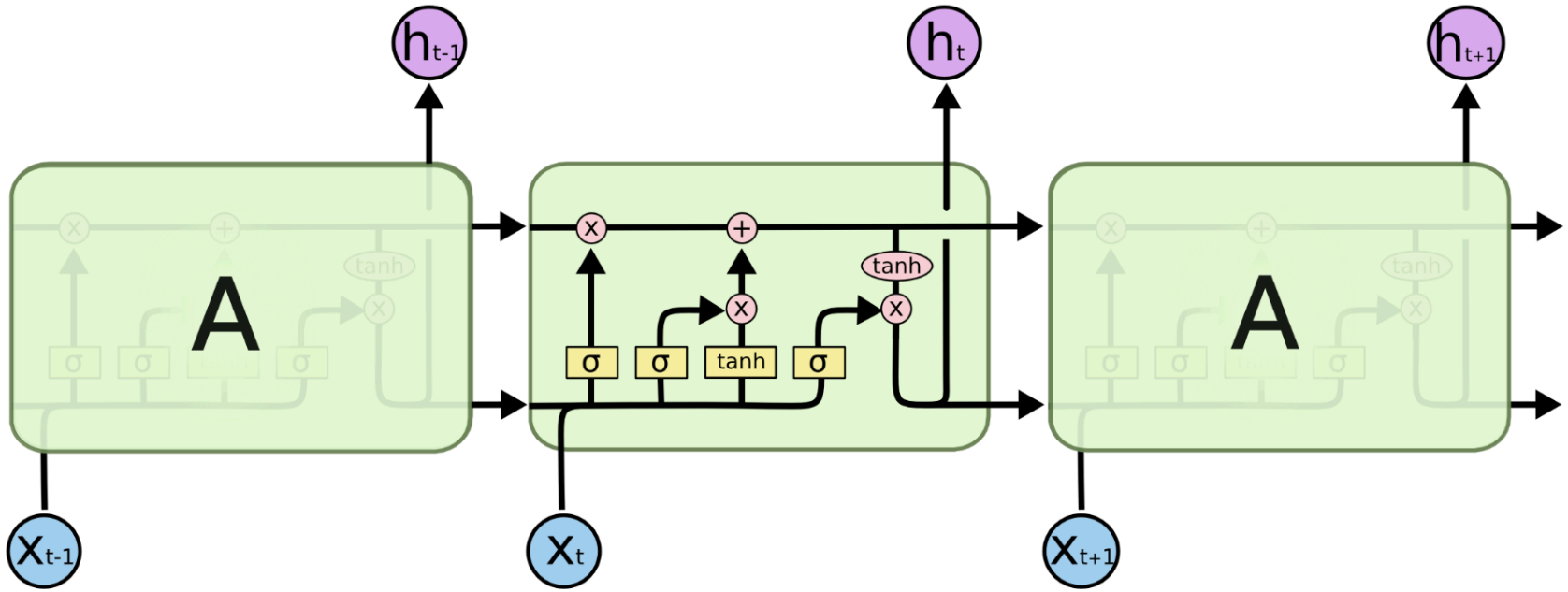
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long term dependencies



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM

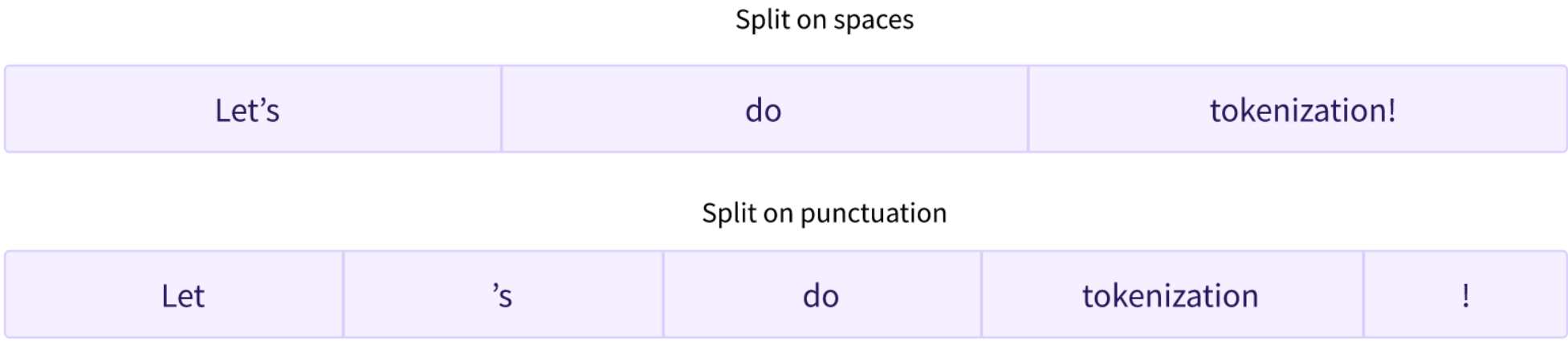


https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Tokenization

- Tokenizer
  - https://huggingface.co/learn/nlp-course/chapter2/4?fw=pt
  - https://blog.floydhub.com/tokenization-nlp/

Word Based Tokenization

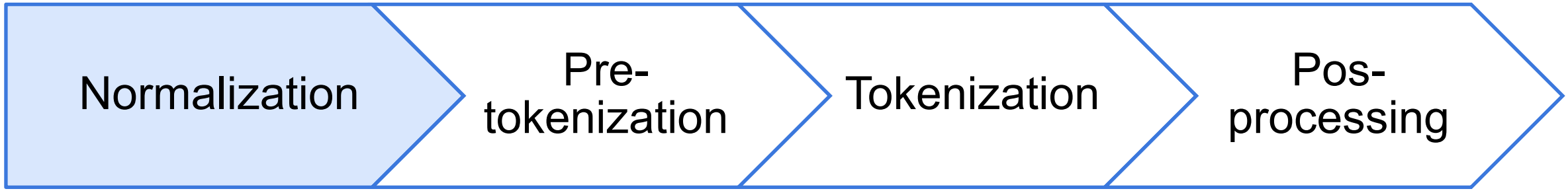


Character Based Tokenization

Subword Tokenization

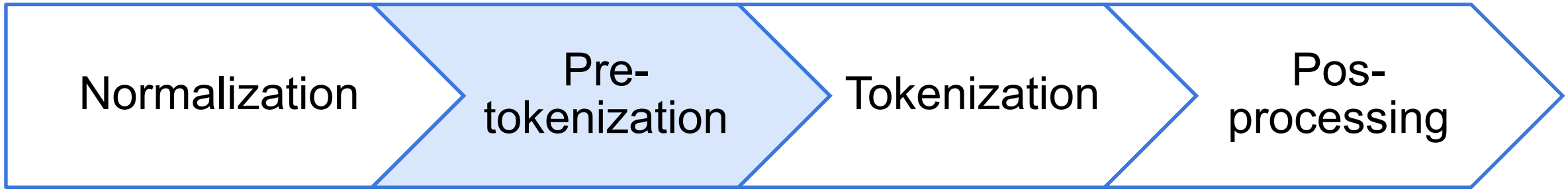
Let’s </w>	do</w>	token	ization</w>	!</w>

The Tokenization Pipeline



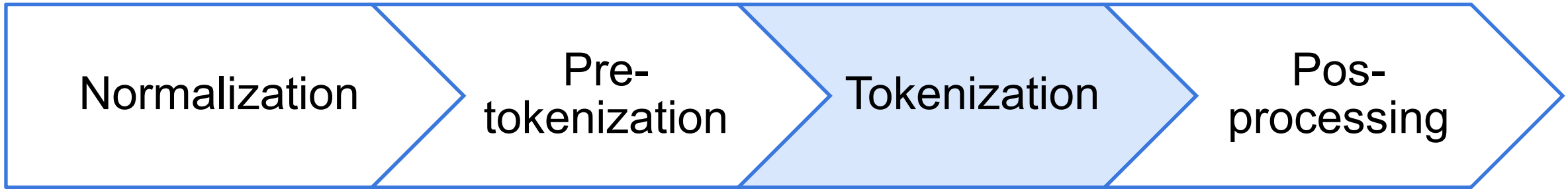
- Strip extra spaces
- Unicode normalization, ...

The Tokenization Pipeline



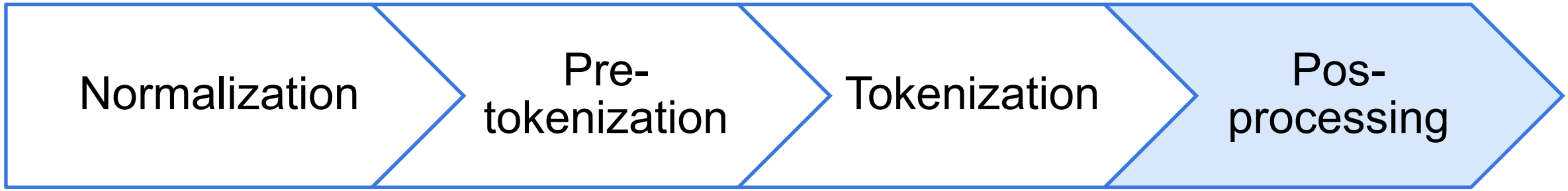
- White spaces between words and sentences
- Punctuations
- ...

The Tokenization Pipeline



- BPE, …. (will discuss this in a second)

The Tokenization Pipeline



- Add special tokens: for example [CLS], [SEP] for BERT
- Truncate to match the maximum length of the model
- Pad all sentences in a batch to the same length

Byte-pair Encoding (BPE)

- An algorithm for forming subword tokens based on a collection of raw text.

```
and there are no re ##fueling stations anywhere
One of the city’s more un ##princi ##pled real state agents
```

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding (BPE): Example

- Form base vocabulary of all characters that occur in the training set.
- *Example:*

Our (very fascinating🙄) training data: "`jhu jhu jhu hopkins hop hops hops`"
Base vocab: `h, i, j, k, n, o, p, s, u`
Tokenized data: `j h u j h u j h u h o p k i n s h o p h o p s h o p s`

Does not show the word separator for simplicity.

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (2)

- Count the frequency of each token pair in the data
- *Example:*

  Our (very fascinating🙄) training data: "`jhu jhu jhu hopkins hop hops hops`"
  Base vocab: `h, i, j, k, n, o, p, s, u`
  Tokenized data: `j h u j h u j h u h o p k i n s h o p h o p s h o p s`
  Token pair frequencies:

  - j + h -> 3
  - h + u -> 3
  - h + o -> 4
  - o + p -> 4
  - p + k -> 1
  - k + i -> 1
  - ….

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (3)

- Choose the pair that occurs more, merge them and add to vocab.
- *Example:*

Our (very fascinating🙄) training data: "`jhu jhu jhu hopkins hop hops hops`"
Base vocab: `h, i, j, k, n, o, p, s, u`
Tokenized data: `j h u j h u j h u h o p k i n s h o p h o p s h o p s`
Token pair frequencies:

- `j + h -> 3`
- `h + u -> 3`
- `h + o -> 4` 
- `o + p -> 4`
- `p + k -> 1`
- `k + i -> 1`
- `....`

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (4)

- Choose the pair that occurs more, merge them and add to vocab.
- *Example:*

Our (very fascinating🙄) training data: "`jhu jhu jhu hopkins hop hops hops`"
Base vocab: `h, i, j, k, n, o, p, s, u, ho` ⬅
Tokenized data: `j h u j h u j h u h o p k i n s h o p h o p s h o p s`
Token pair frequencies:

- j + h -> 3
- h + u -> 3
- h + o -> 4 ⬅
- o + p -> 4
- p + k -> 1
- k + i -> 1
- ....

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (5)

- Retokenize the data
- *Example:*

Our (very fascinating🙄) training data: "`jhu jhu jhu hopkins hop hops hops`"
Base vocab: `h, i, j, k, n, o, p, s, u, ho`
Tokenized data: `j h u j h u j h u ho p k i n s ho p ho p s ho p s`
Token pair frequencies:

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (6)

- Count the token pairs and merge the most frequent one
- *Example:*

Our (very fascinating🙄) training data: "`jhu jhu jhu hopkins hop hops hops`"
Base vocab: `h, i, j, k, n, o, p, s, u, ho`
Tokenized data: `j h u j h u j h u ho p k i n s ho p ho p s ho p s`
Token pair frequencies:

- j + h -> 3
- h + u -> 3
- ho + p -> 4
- p + k -> 1
- k + i -> 1
- i + n -> 1
- ….

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (7)

- Count the token pairs and merge the most frequent one
- *Example:*

Our (very fascinating🙄) training data: "`jhu jhu jhu hopkins hop hops hops`"
Base vocab: `h, i, j, k, n, o, p, s, u, ho`
Tokenized data: `j h u j h u j h u ho p k i n s ho p ho p s ho p s`
Token pair frequencies:

- j + h -> 3
- h + u -> 3
- ho + p -> 4 
- p + k -> 1
- k + i -> 1
- i + n -> 1
- ….

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (7)

- Count the token pairs and merge the most frequent one
- *Example:*

Our (very fascinating🙄) training data: "`jhu jhu jhu hopkins hop hops hops`"
Base vocab: `h, i, j, k, n, o, p, s, u, ho, hop` ⬅
Tokenized data: `j h u j h u j h u ho p k i n s ho p ho p s ho p s`
Token pair frequencies:

- j + h -> 3
- h + u -> 3
- ho + p -> 4 ⬅
- p + k -> 1
- k + i -> 1
- i + n -> 1
- ….

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (7)

- Count the token pairs and merge the most frequent one
- *Example:*

Our (very fascinating🙄) training data: "`jhu jhu jhu hopkins hop hops hops`"
Base vocab: `h, i, j, k, n, o, p, s, u, ho, hop` ⬅
Tokenized data: `j h u j h u j h u hop k i n s hop hop s hop s` ⬅
Token pair frequencies:

- j + h -> 3
- h + u -> 3
- ho + p -> 4 ⬅
- p + k -> 1
- k + i -> 1
- i + n -> 1
- ….

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (8)

- Count the token pairs and merge the most frequent one
- *Example:*

Our (very fascinating🙄) training data: "`jhu jhu jhu hopkins hop hops hops`"
Base vocab: `h, i, j, k, n, o, p, s, u, ho, hop`
Tokenized data: `j h u j h u j h u hop k i n s hop hop s hop s`
Token pair frequencies:

- j + h -> 3 ⬅
- h + u -> 3
- hop + k -> 1
- hop + s -> 2
- k + i -> 1
- i + n -> 1
- n + s -> 1
- ....

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (8)

- Count the token pairs and merge the most frequent one
- *Example:*

Our (very fascinating🙄) training data: "`jhu jhu jhu hopkins hop hops hops`"
Base vocab: `h, i, j, k, n, o, p, s, u, ho, hop, jh` 
Tokenized data: `j h u j h u j h u hop k i n s hop hop s hop s`
Token pair frequencies:

- j + h -> 3 
- h + u -> 3
- hop + k -> 1
- hop + s -> 2
- k + i -> 1
- i + n -> 1
- n + s -> 1
- ….

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (8)

- Count the token pairs and merge the most frequent one
- *Example:*

Our (very fascinating🙄) training data: "`jhu jhu jhu hopkins hop hops hops`"
Base vocab: `h, i, j, k, n, o, p, s, u, ho, hop, jh` ⬅
Tokenized data: `jh u jh u jh u hop k i n s hop hop s hop s` ⬅
Token pair frequencies:

- j + h -> 3 ⬅
- h + u -> 3
- hop + k -> 1
- hop + s -> 2
- k + i -> 1
- i + n -> 1
- n + s -> 1
- ....

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (8)

- Count the token pairs and merge the most frequent one
- *Example:*

Our (very fascinating🙄) training data: "`jhu jhu jhu hopkins hop hops hops`"
Base vocab: `h, i, j, k, n, o, p, s, u, ho, hop, jh`
Tokenized data: `jh u jh u jh u hop k i n s hop hop s hop s`
Token pair frequencies:

- j h+u -> 3 ⬅
- hop + k -> 1
- hop + s -> 2
- k + i -> 1
- i + n -> 1
- n + s -> 1
- ….

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (8)

- Count the token pairs and merge the most frequent one
- *Example:*

Our (very fascinating🙄) training data: "`jhu jhu jhu hopkins hop hops hops`"
Base vocab: `h, i, j, k, n, o, p, s, u, ho, hop, jh, jhu` ⬅
Tokenized data: `jh u jh u jh u hop k i n s hop hop s hop s`
Token pair frequencies:

- j h+u -> 3 ⬅
- hop + k -> 1
- hop + s -> 2
- k + i -> 1
- i + n -> 1
- n + s -> 1
- ….

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (8)

- Count the token pairs and merge the most frequent one
- *Example:*

Our (very fascinating🙄) training data: "`jhu jhu jhu hopkins hop hops hops`"
Base vocab: `h, i, j, k, n, o, p, s, u, ho, hop, jh, jhu` 
Tokenized data: `jhu jhu jhu hop k i n s hop hop s hop s` 
Token pair frequencies:

- j h+u -> 3 
- hop + k -> 1
- hop + s -> 2
- k + i -> 1
- i + n -> 1
- n + s -> 1
- ....

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Language Model

Thou shalt not make a machine in the likeness of a human mind

Sliding window across running text

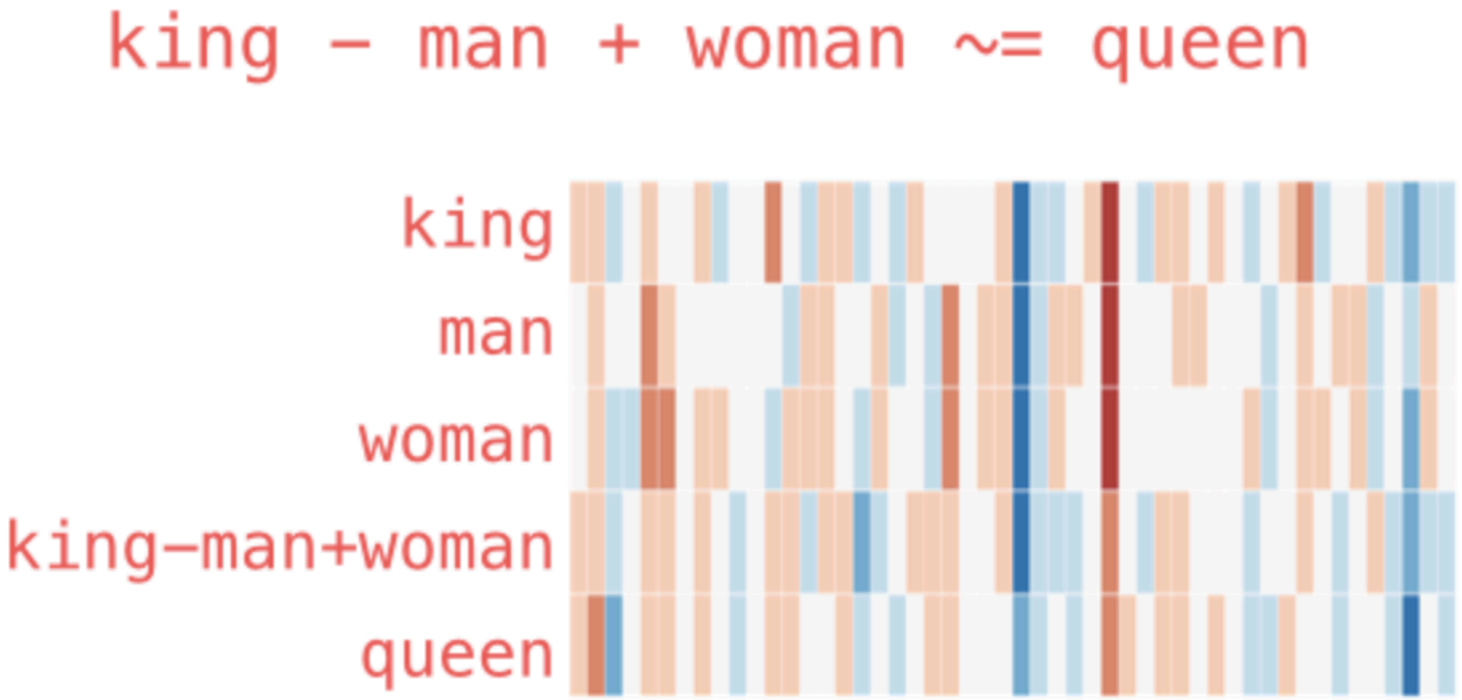
thou	shalt	not	make	a	machine	in	the	…
thou	shalt	not	make	a	machine	in	the	
thou	shalt	not	make	a	machine	in	the	
thou	shalt	not	make	a	machine	in	the	
thou	shalt	not	make	a	machine	in	the	

Dataset

input 1	input 2	output
thou	shalt	not
shalt	not	make
not	make	a
make	a	machine
a	machine	in

https://jalammar.github.io/illustrated-word2vec/

Word Vector



https://jalammar.github.io/illustrated-word2vec/

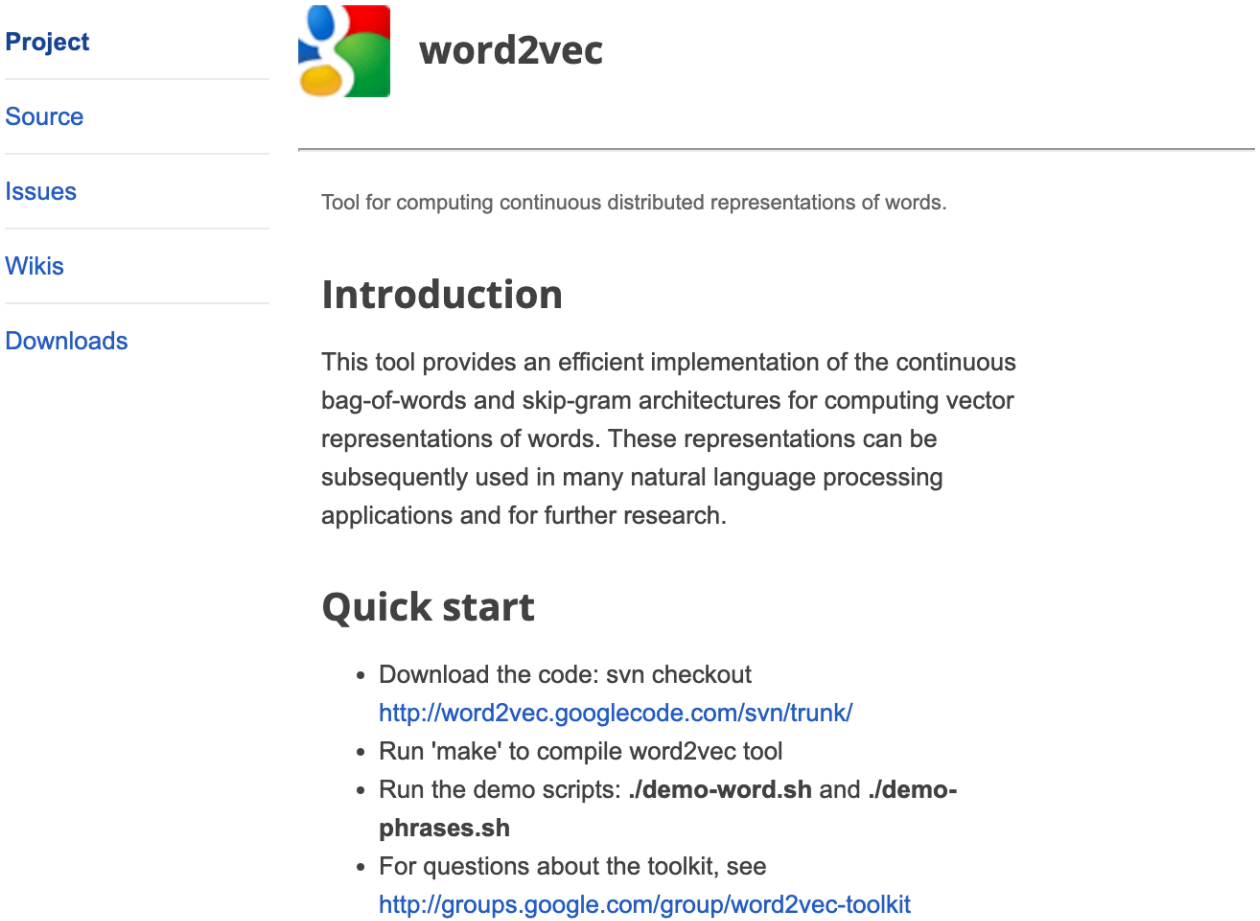
Glove

Download pre-trained word vectors

- Pre-trained word vectors. This data is made available under the Public Domain Dedication and License v1.0 whose full text can be found at: http://www.opendatacommons.org/licenses/pddl/1.0/.
  - Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 50d, 100d, 200d, & 300d vectors, 822 MB download): glove.6B.zip
  - Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors, 1.75 GB download): glove.42B.300d.zip
  - Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors, 2.03 GB download): glove.840B.300d.zip
  - Twitter (2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 100d, & 200d vectors, 1.42 GB download): glove.twitter.27B.zip
- Ruby script for preprocessing Twitter data

https://nlp.stanford.edu/projects/glove/

word2vec



Project

Source

Issues

Wikis

Downloads

word2vec

Tool for computing continuous distributed representations of words.

Introduction

This tool provides an efficient implementation of the continuous bag-of-words and skip-gram architectures for computing vector representations of words. These representations can be subsequently used in many natural language processing applications and for further research.

Quick start

- Download the code: svn checkout http://word2vec.googlecode.com/svn/trunk/
- Run 'make' to compile word2vec tool
- Run the demo scripts: **./demo-word.sh** and **./demo-phrases.sh**
- For questions about the toolkit, see http://groups.google.com/group/word2vec-toolkit

https://code.google.com/archive/p/word2vec/

fastText

Library for efficient text classification and representation learning

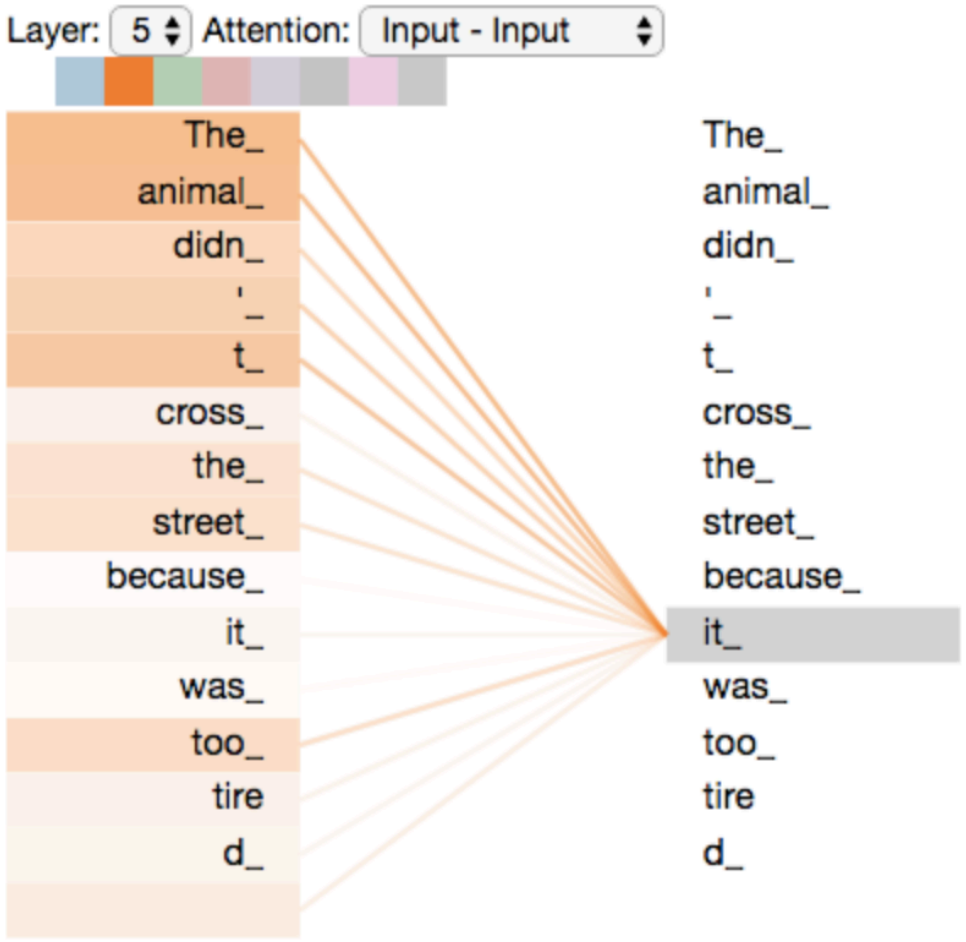
GET STARTED | DOWNLOAD MODELS

Word vectors for 157 languages

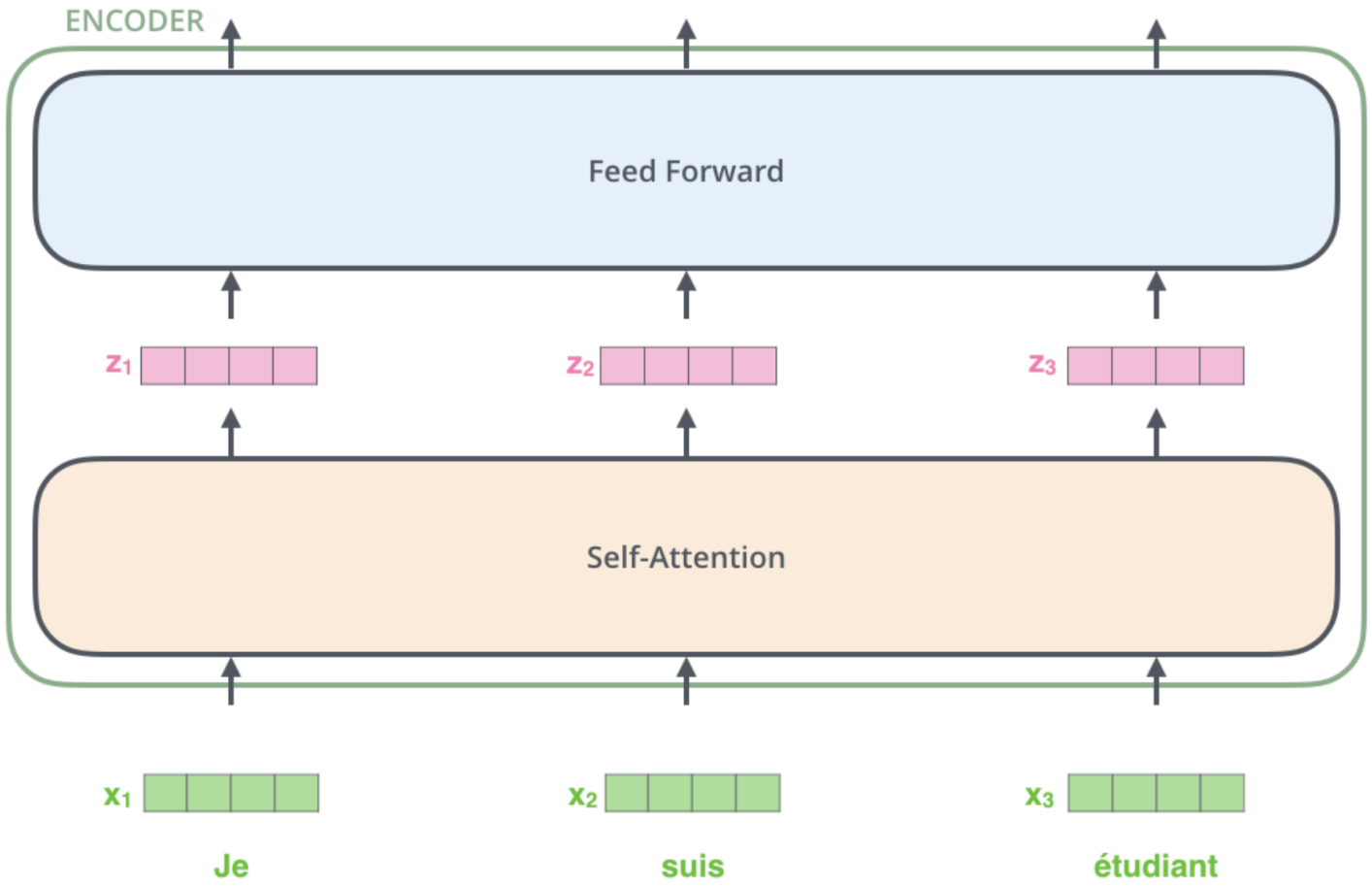
We distribute pre-trained word vectors for 157 languages, trained on *Common Crawl* and *Wikipedia* using fastText. These models were trained using CBOW with position-weights, in dimension 300, with character n-grams of length 5, a window of size 5 and 10 negatives. We also distribute three new word analogy datasets, for French, Hindi and Polish.

https://fasttext.cc/docs/en/crawl-vectors.html

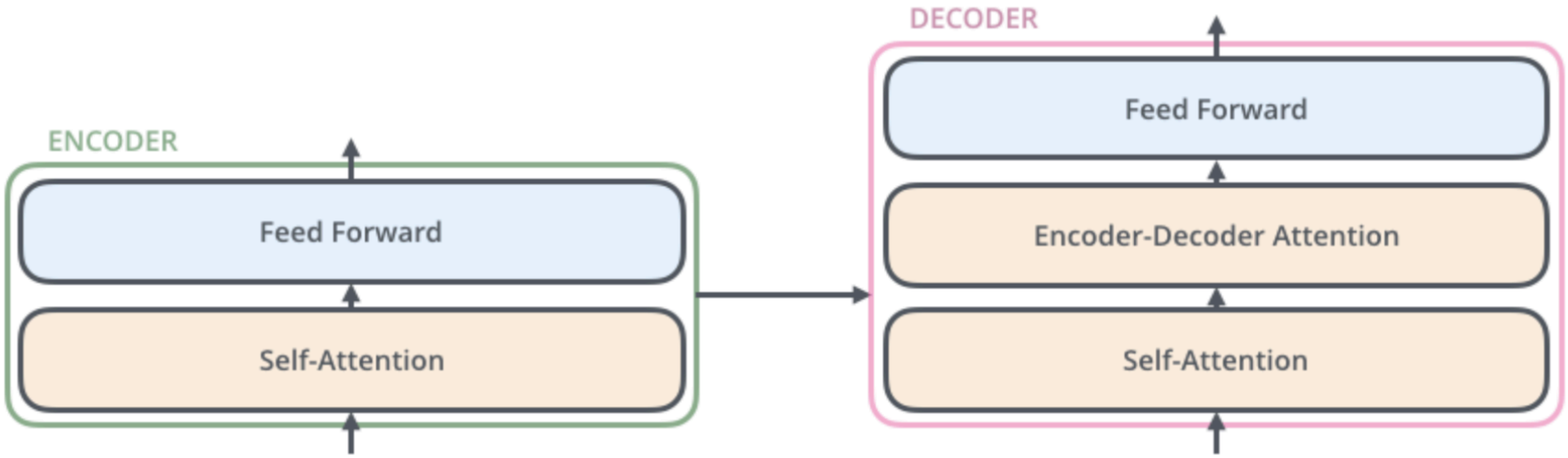
Attention


Layer: 5
Attention: Input - Input
The_
animal_
didn_
'_
t_
cross_
the_
street_
because_
it_
was_
too_
tire
d_
The_
animal_
didn_
'_
t_
cross_
the_
street_
because_
it_
was_
too_
tire
d_

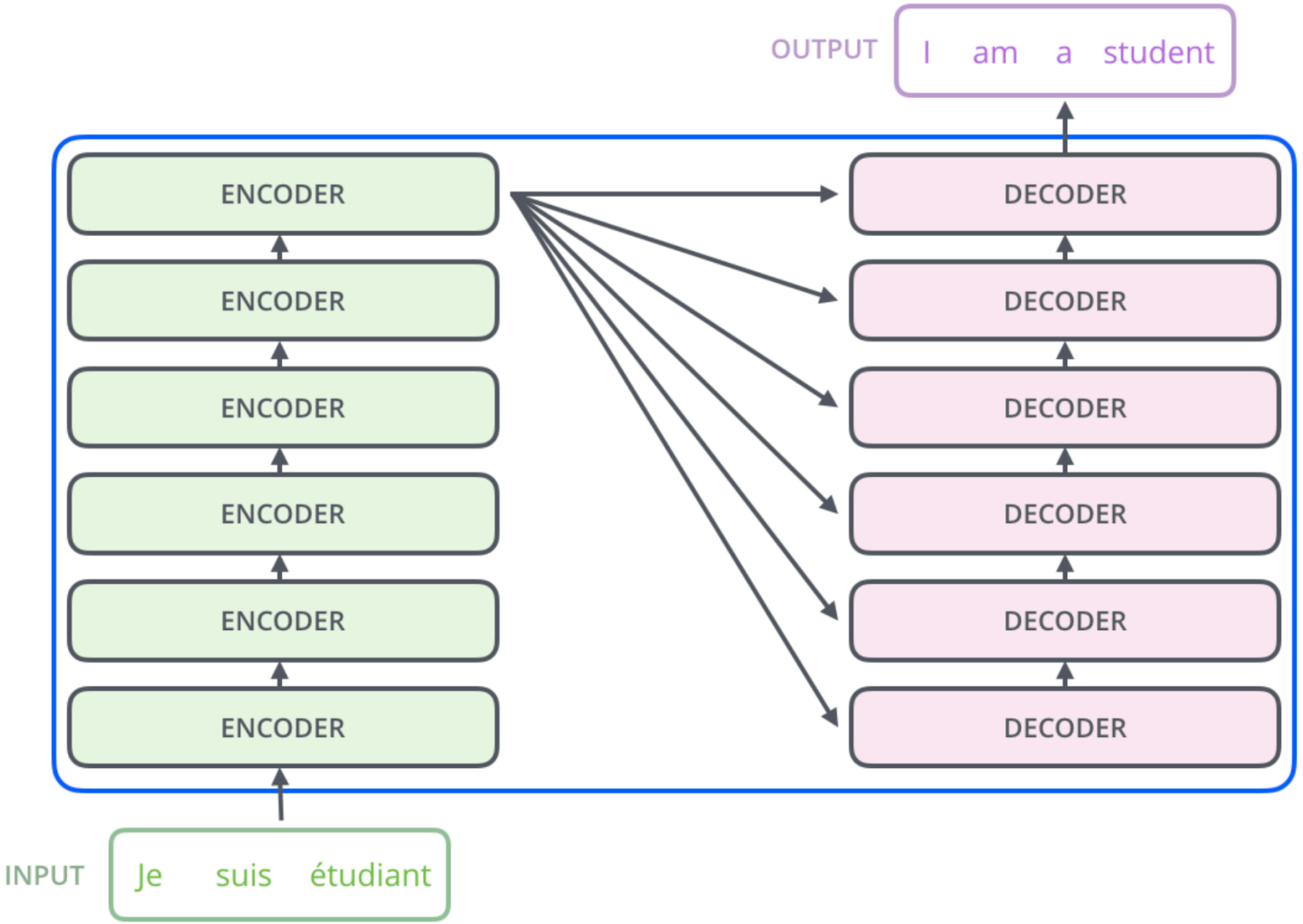
Encoder


ENCODER
Feed Forward
z1
z2
z3
Self-Attention
x1
x2
x3
Je
suis
étudiant

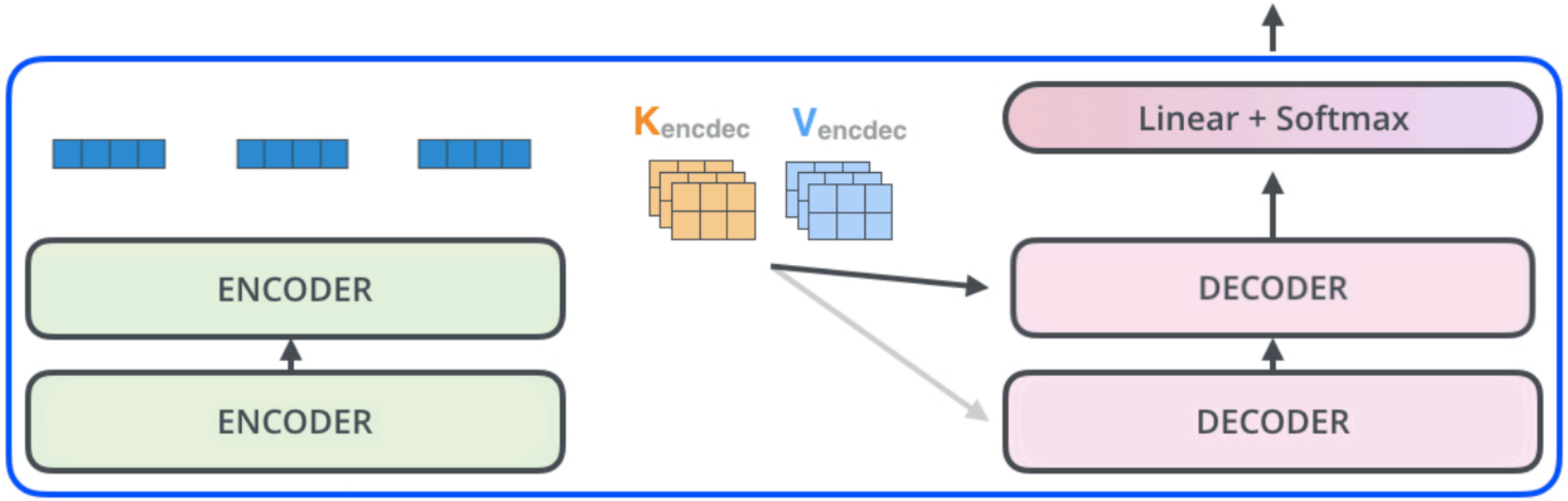
Encoder-Decoder



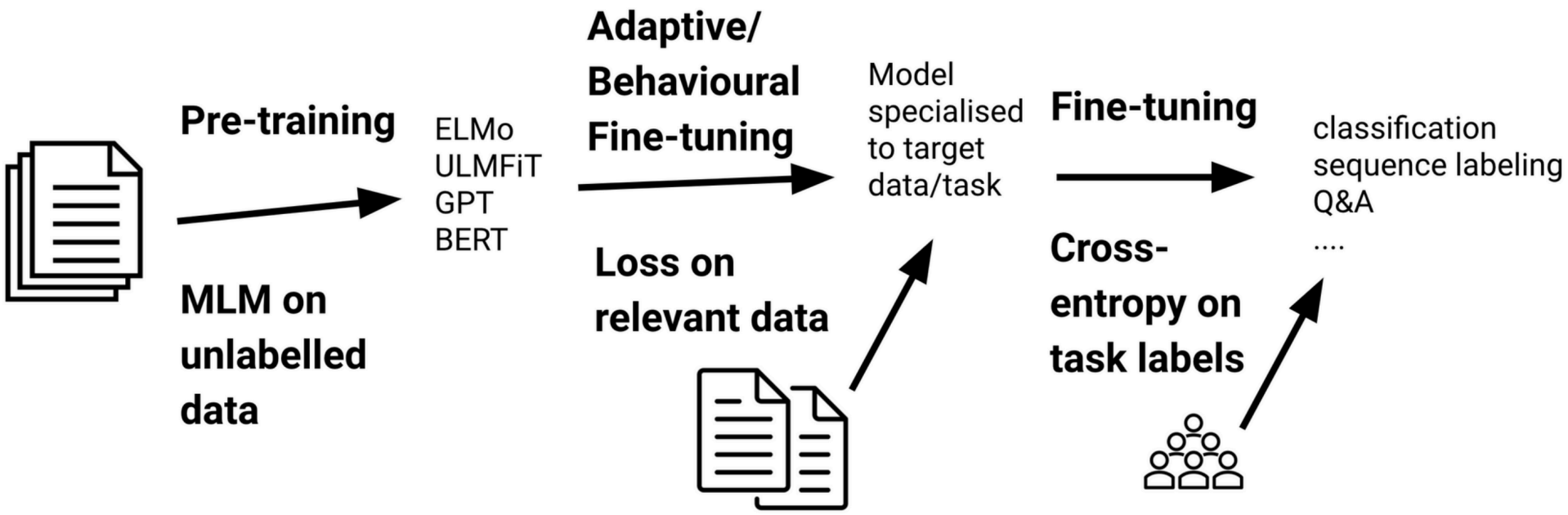
Transformer



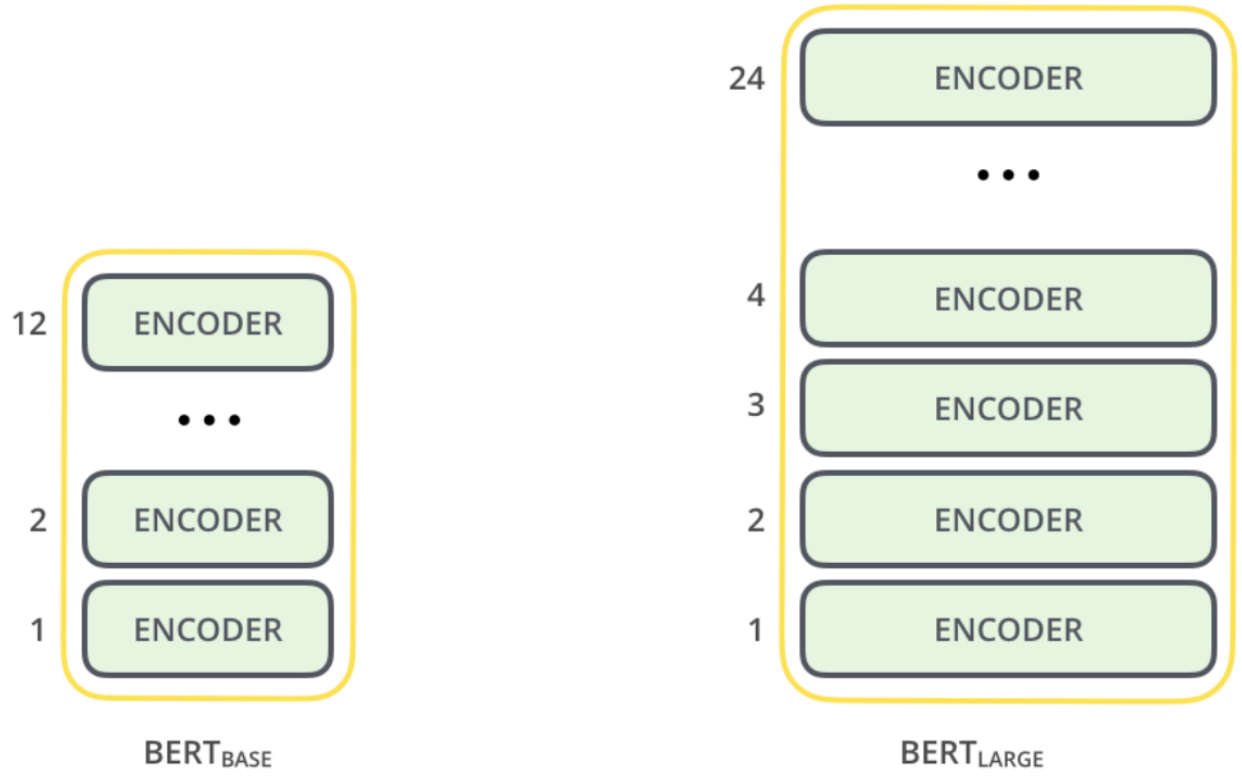
NLP Task


Linear + Softmax
K_{encdec}
V_{encdec}
ENCODER
ENCODER
DECODER
DECODER

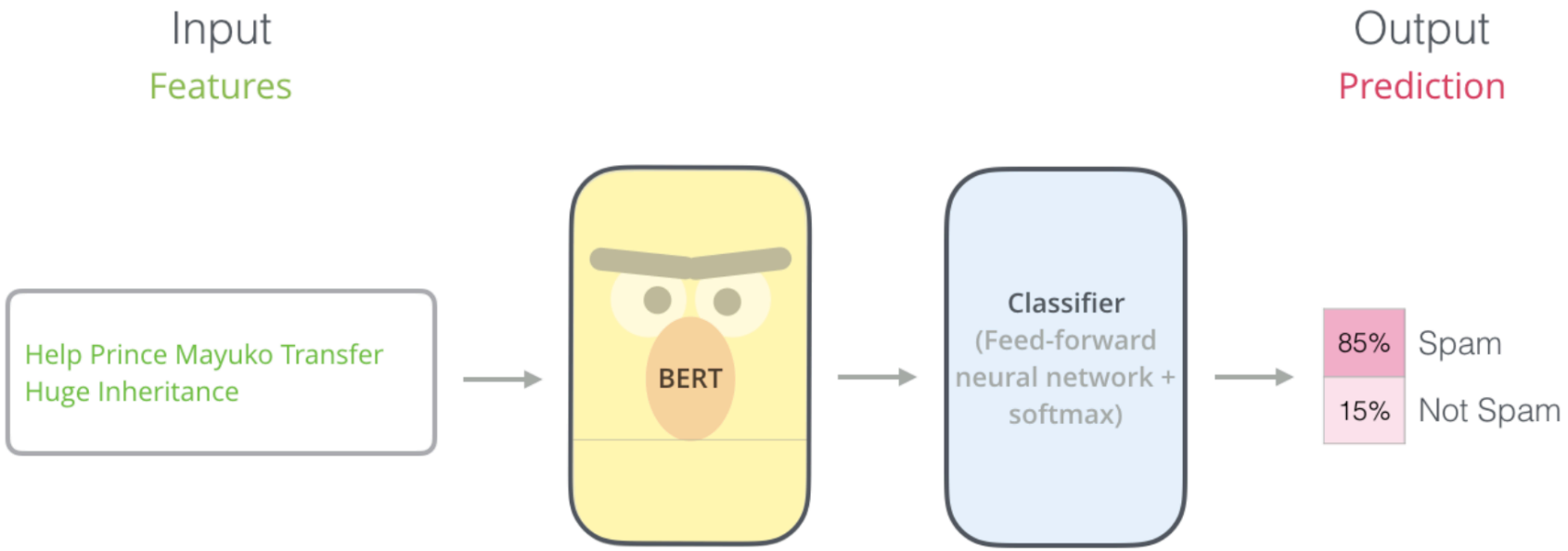
Pre-training and Fine-tuning



BERT Base and Large

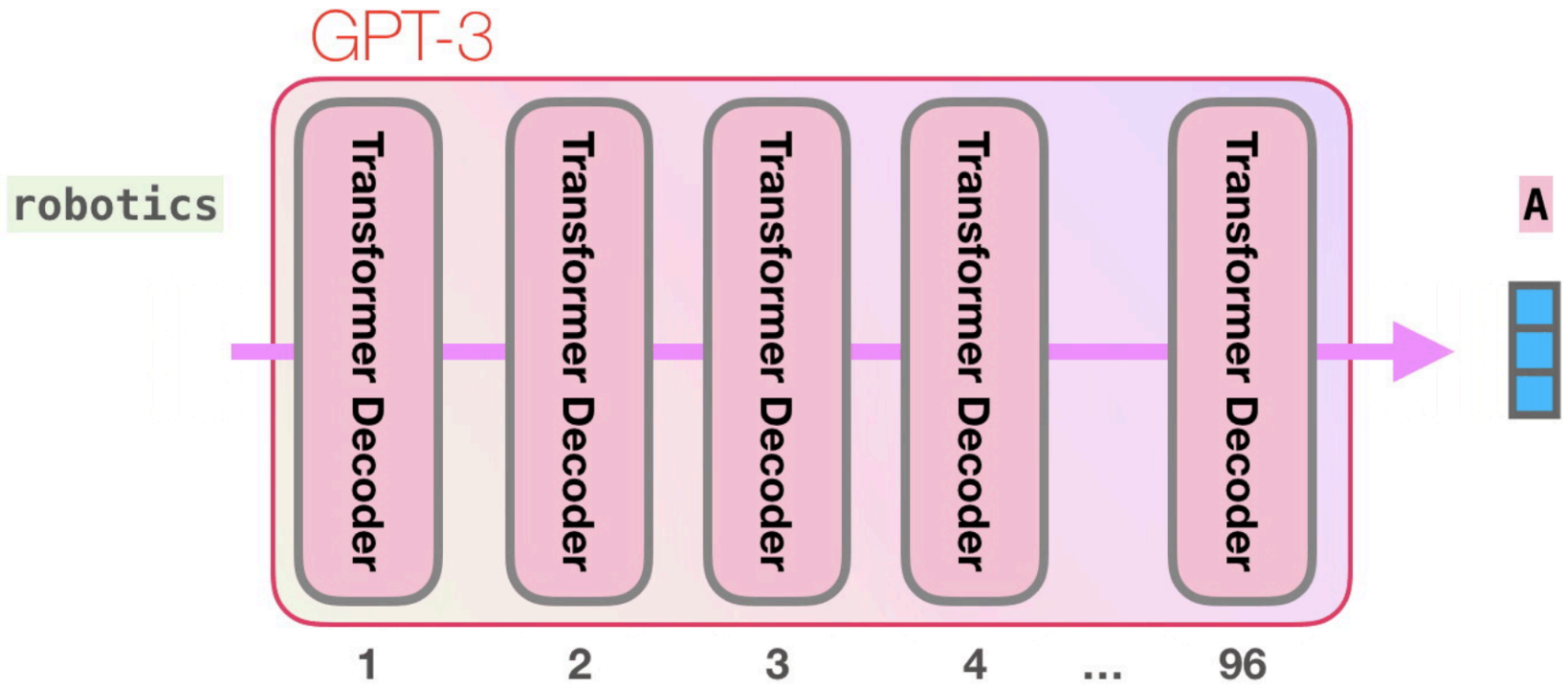


BERT Base and Large



https://jalammar.github.io/illustrated-bert/

GPT-3



https://jalammar.github.io/how-gpt3-works-visualizations-animations/

NLP Tasks

https://nlpprogress.com/