

# Deep tokenization

---

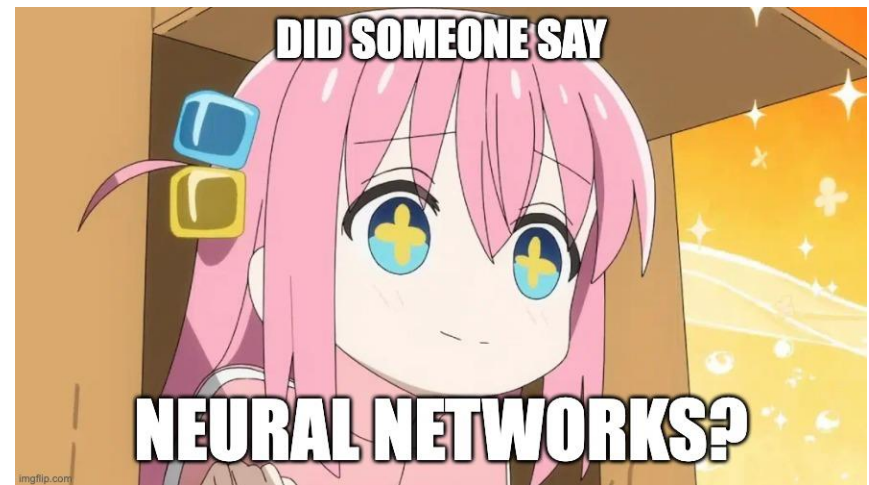
# Tokenization

- Recent tokenization techniques are based on deep learning models
  - Better to handle out-of-vocabulary (OOV), misspellings, etc.

# NEURAL NETWORKS

---

Deep learning = Deep neural networks =  
neural networks



# Why neural networks

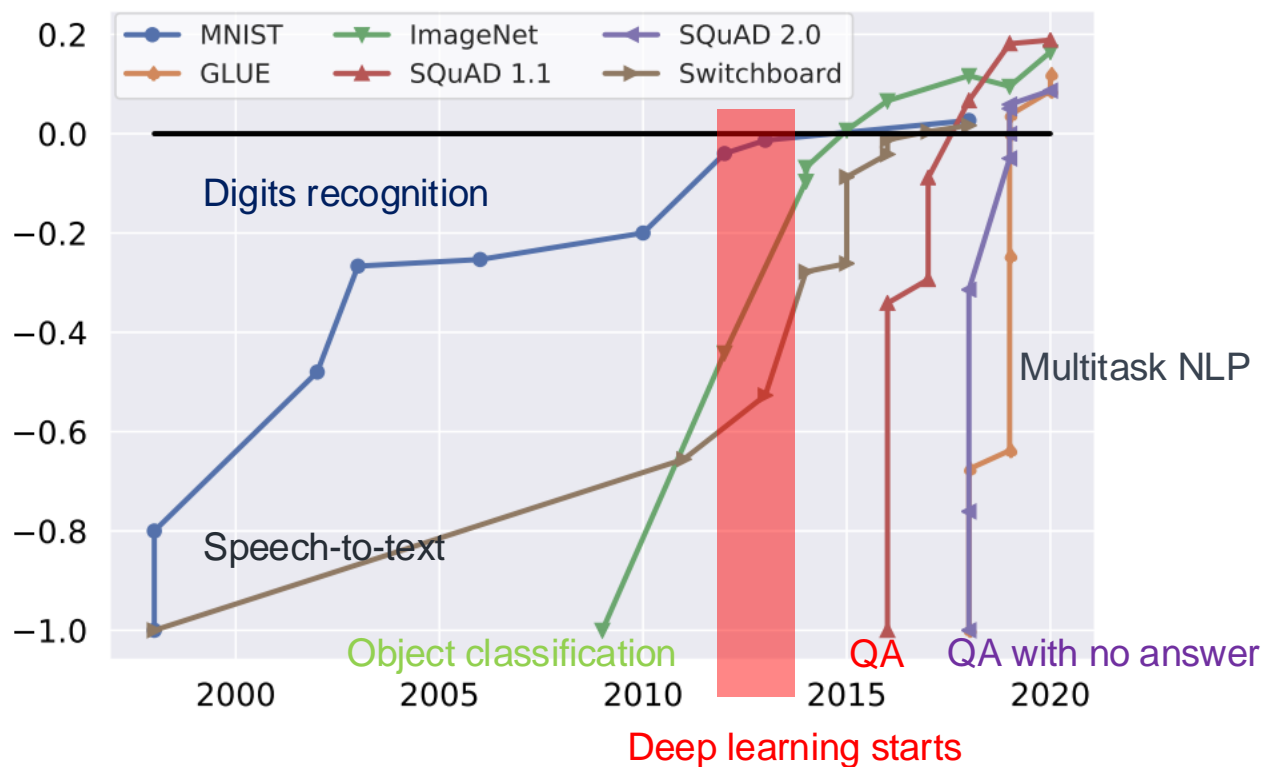
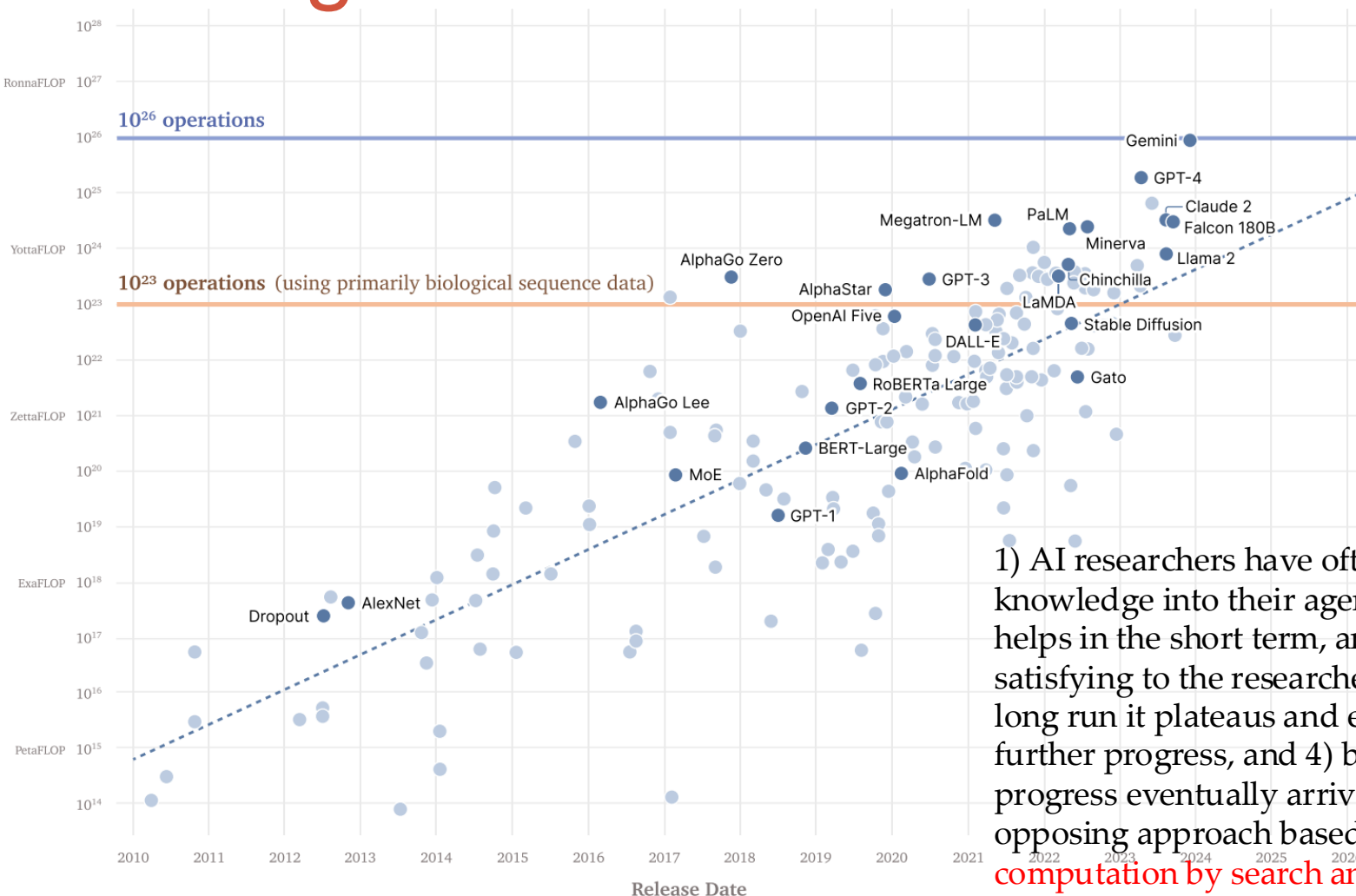


Figure 1: Benchmark saturation over time for popular benchmarks, normalized with initial performance at minus one and human performance at zero.

# Scaling matters

<https://arxiv.org/pdf/2402.08797>



1) AI researchers have often tried to build knowledge into their agents, 2) this always helps in the short term, and is personally satisfying to the researcher, but 3) in the long run it plateaus and even inhibits further progress, and 4) breakthrough progress eventually arrives by an opposing approach based on **scaling computation by search and learning**.

- Richard Sutton  
The Bitter Lesson

**Figure 4: Training compute used for notable ML models has been doubling every six months since the emergence of the Deep Learning Era. Executive Order 14110 introduced a notification requirement for models trained with more than  $10^{26}$  operations (and  $10^{23}$  operations if trained on using primarily biological sequence data).**

# Deep learning in NLP

## Easy task modest gains

|                           | Traditional ML | Deep learning  |
|---------------------------|----------------|--|
| Wisesight Sentiment (th)  | 72%            | 76% (WangchanBERTa)<br>71% (Phayathaibert unsup)<br>(60% QWEN2-72B zeroshot) |
| Topic classification (th) | 67%            | 70%  |
| PoS (th)                  | 96%            | 97%  |

## Harder task larger gains

|                          | Traditional ML | Deep learning |
|--------------------------|----------------|---------------|
| QA                       | 51%*           | 90%           |
| Creating image from text | ???            | very good     |

[wangchanbert](https://www.aclweb.org/anthology/D16-1264/)

<https://www.aclweb.org/anthology/D16-1264/>

<https://openai.com/blog/dall-e/>

TEXT PROMPT

an illustration of a baby daikon radish in a tutu walking a dog

AI-GENERATED IMAGES



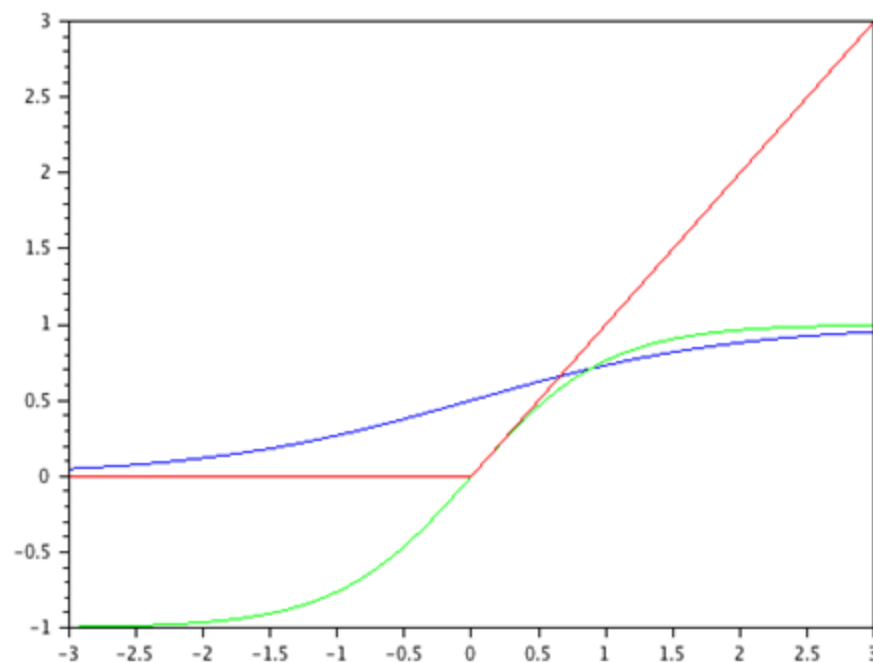
Edit prompt or view more images +

# Neural networks

- Fully connected networks
    - Neuron
    - Non-linearity
    - Softmax layer
    - Dropout
    - Batchnorm
  - CNN, RNN, LSTM, GRU
- 
- If you are unaware of these, please watch last year's Pattern [lecture 7](#) and [7.5](#)

# Non-linearity

- The Non-linearity is important in order to stack neurons
  - If  $F$  is linear, a multi layered network can be collapsed as a single layer (by just multiplying weights together)
- Sigmoid or logistic function
- $\tanh$
- Rectified Linear Unit (ReLU)
  - LeakyReLU, ELU, PreLU
- Sigmoid Linear Units (SiLU)
  - Swish, Mish, GELU, SwiGLU





# SiLU, GELU

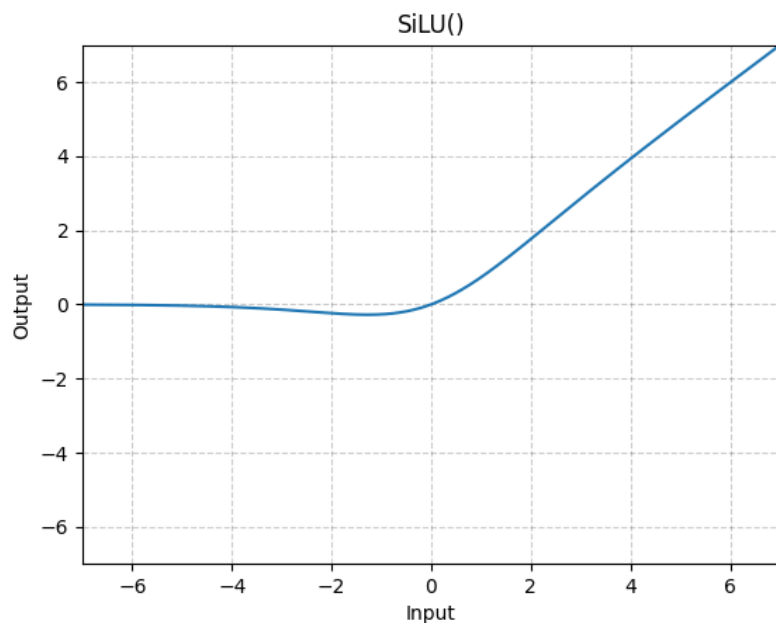
**SiLU** (or Swish – Swish paper comes after SiLU but is more popular)

CLASS `torch.nn.SiLU(inplace=False)` [\[SOURCE\]](#)

Applies the Sigmoid Linear Unit (SiLU) function, element-wise.

The SiLU function is also known as the swish function.

$\text{silu}(x) = x * \sigma(x)$ , where  $\sigma(x)$  is the logistic sigmoid.



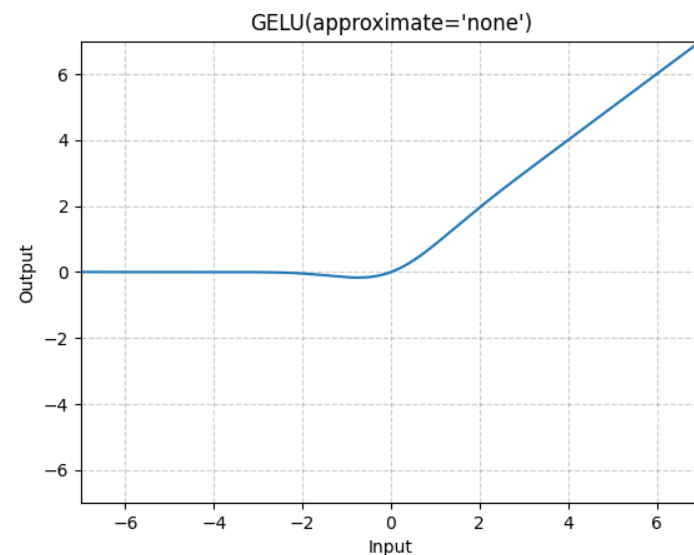
## GELU

CLASS `torch.nn.GELU(approximate='none')` [\[SOURCE\]](#)

Applies the Gaussian Error Linear Units function.

$$\text{GELU}(x) = x * \Phi(x)$$

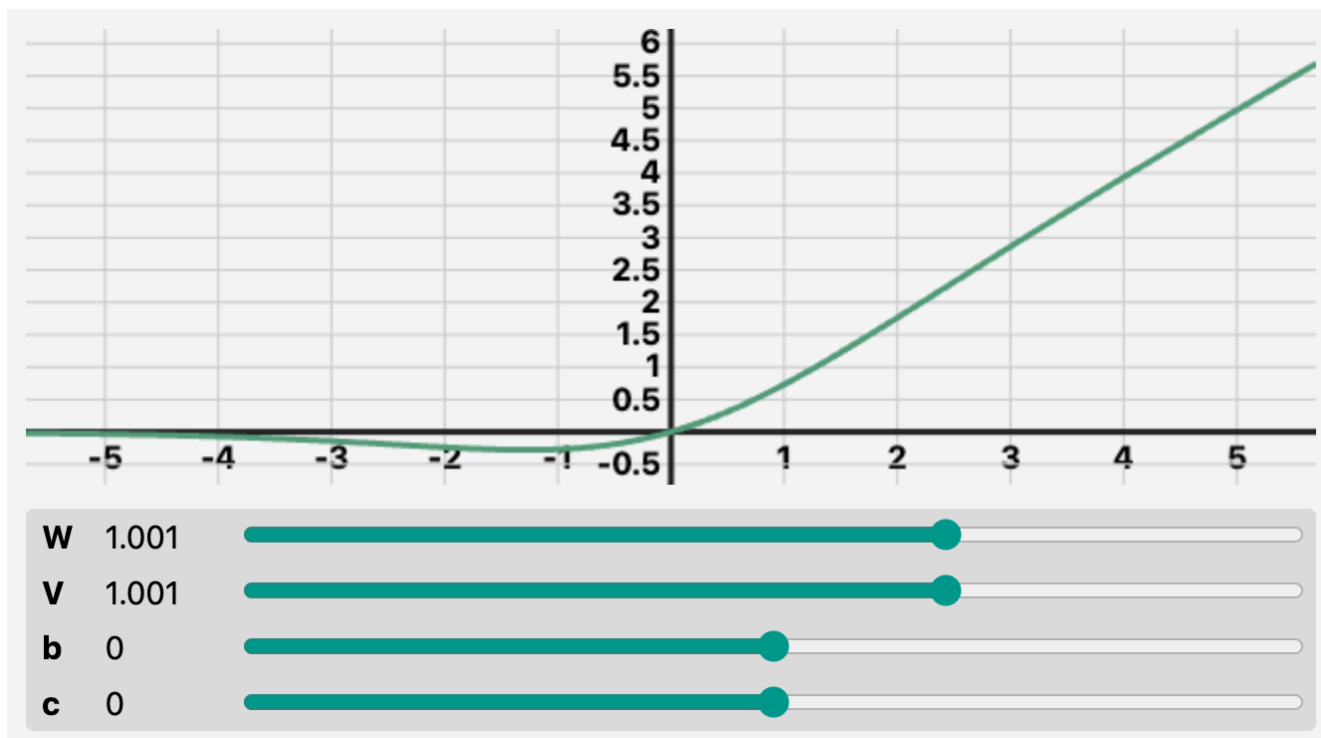
where  $\Phi(x)$  is the Cumulative Distribution Function for Gaussian Distribution.



# GLU (Gated Linear Unit)

- A gated dense layer

$$\text{GLU}(x) = (Wx+b) * \text{sigmoid}(Vx+c)$$



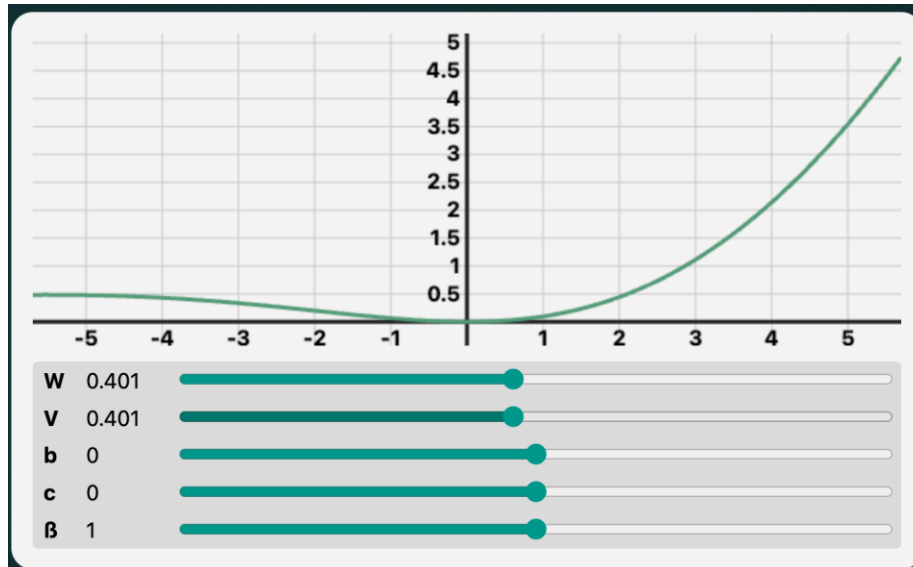
# SwiGLU

- GLU with a Swish/SiLU gating function
- Many LLMs use this. Example: Llama

$$\text{GLU}(x) = (Wx+b) * \text{sigmoid}(Vx+c)$$

$$\text{Swish}(x) = x * \text{sigmoid}(Bx)$$

$$\text{SwiGLU}(x) = (Wx+b) * \text{Swish}(Vx+c)$$



<https://jcarlosroldan.com/post/348/what-is-swigu>

# Batch normalization

- Recent technique for (implicit) regularization
- **Normalize every mini-batch** at various batch norm layers to standard Gaussian (different from global normalization of the inputs)
- Place batch norm layers before non-linearities
- Faster training and better generalizations

For each mini-batch that goes through batch norm

1. Normalize by the mean and variance of the mini-batch for each dimension
2. Shift and scale by learnable parameters

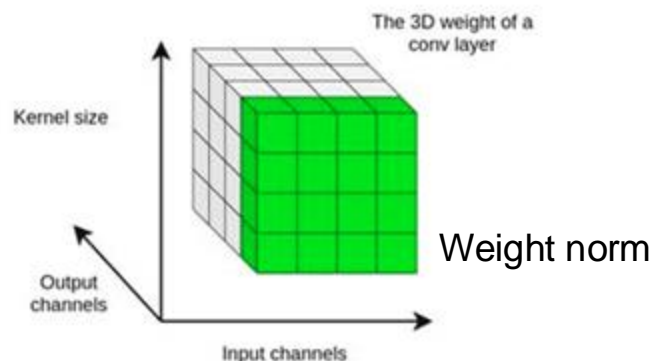
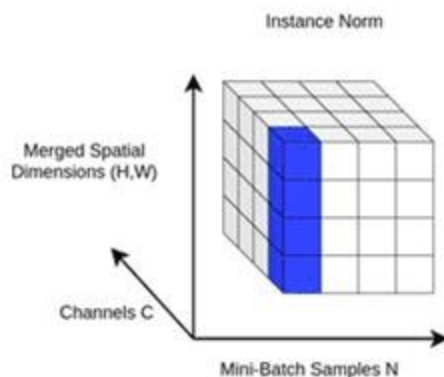
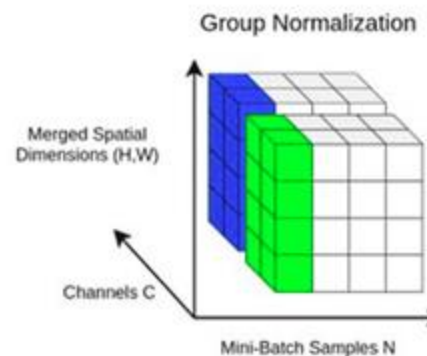
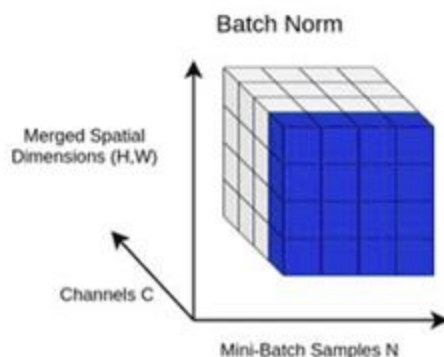
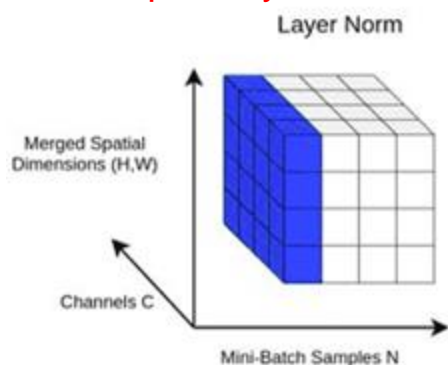
Replaces dropout in some networks

$$\hat{x} = \frac{x - \mu_b}{\sigma_b}$$
$$y = \alpha \hat{x} + \beta$$

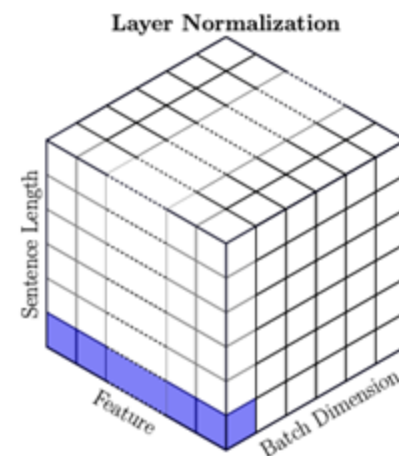
<https://arxiv.org/abs/1502.03167>

# Other normalizations

- Other normalizations are out there  
NLP and CV layer norm are not the same (In NLP, layer norm is applied separately for each element in the sequence)

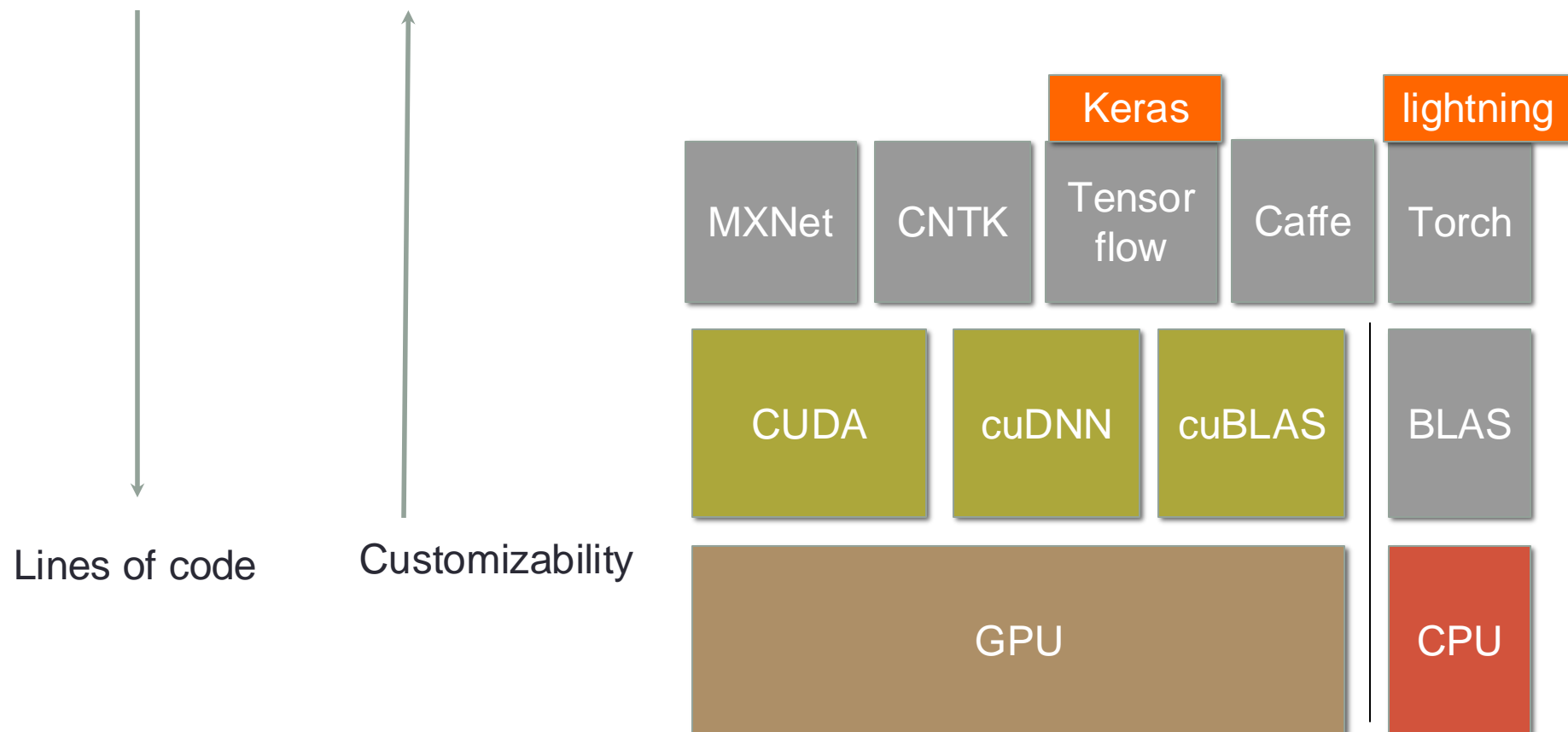


Layer norm in transformers



# What toolkit

Tradeoff between customizability and ease of use




# Pytorch steps

- Setting up dataloader
  - Gives minibatch
- Define a network
  - Init weights
  - Define computation graph
- Setup optimization method
  - Pick LR scheduler
  - Pick optimizer
- Training loop
  - Forward (compute Loss)
  - Backward (compute gradient and apply gradient)
- Let's demo

# Lightning

- Setting up dataloader
  - Gives minibatch
- Define a network
  - Init weights
  - Define computation graph
- Setup optimization method
  - Pick LR scheduler
  - Pick optimizer
- Training loop
  - Forward (compute Loss)
  - Backward (compute gradient and apply gradient)
- Let's demo



Pytorch  
Lightning  
helps with  
this  
and much  
more



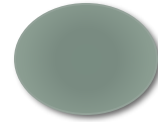
# Lab/HW

- Word segmentation using pytorch
- Given a letter with 10 letters before and after, determine whether it's a start of a word

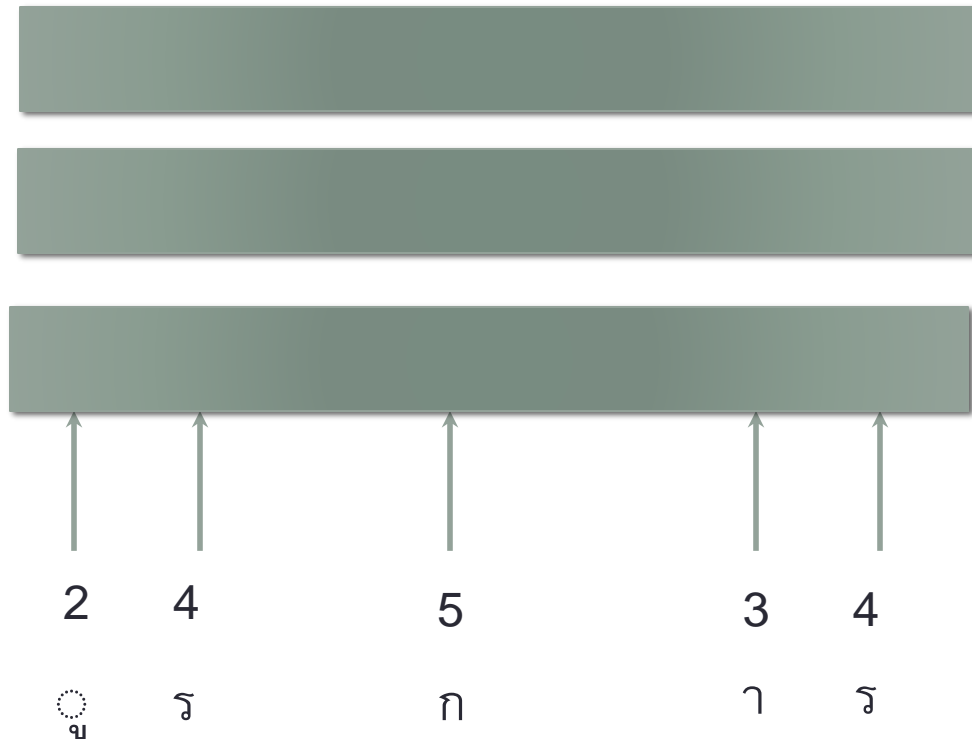
ท ร ง อ ั ย ่า ง แ บ ด แ ช . เ อ ย ั ่า ง บ d i

# Word segmentation with fully connected networks

1 = word beginning, 0 = word middle



Logistic function



# Embeddings

- A way to encode information to a lower dimensional space
  - We can learn about this lower dimensional space through data

|              |              |              |              |
|--------------|--------------|--------------|--------------|
|              |              | DOG          |              |
|              |              | [68, 79, 71] |              |
|              |              |              | PIG          |
|              |              |              | [80, 73, 71] |
| CAT          | CAP          |              |              |
| [67, 65, 84] | [67, 65, 80] |              |              |

# One hot encoding

- Categorical representation is usually represented by **one hot encoding**
  - Categorical representations examples:
    - Words in a vocabulary, characters in Thai language
- Apple -> 1 -> [1, 0, 0, 0, ...]
- Bird -> 2 -> [0, 1, 0, 0, ...]
- Cat -> 3 -> [0, 0, 1, 0, ...]
- **Sparse** representation
    - Sparse means most dimension are zero

# One hot encoding

- Sparse – but lots of dimension
  - Curse of dimensionality
- Does not represent meaning.

Apple -> 1 -> [1, 0, 0, 0, ...]

Bird -> 2 -> [0, 1, 0, 0, ...]

Cat -> 3 -> [0, 0, 1, 0, ...]

$$|\text{Apple} - \text{Bird}| = |\text{Bird} - \text{Cat}|$$

# Getting meaning into the feature vectors

- You can add back meanings by hand-crafted rules
- Old-school NLP is all about feature engineering
- Word segmentation example:
  - Cluster Numbers
  - Cluster letters
- Concatenate them
- $1 = [0\ 0\ 0\ 0\ 1\ 0\ 0\ 0, 1, 0]$
- $\bar{n} = [0\ 0\ 0\ 1\ 0\ 0\ 0\ 0, 0, 1]$
- $\gamma = [1\ 0\ 0\ 0\ 0\ 0\ 0\ 0, 0, 2]$
- Which rules to use?
  - Try as many as you can think of, and do feature selection or use models that can do feature selection

# Dense representation

- We can encode sparse representation into a lower dimensional space
  - $F: \mathbb{R}^N \rightarrow \mathbb{R}^M$ , where  $N > M$

Apple  $\rightarrow 1 \rightarrow [1, 0, 0, 0, \dots] \rightarrow [2.3, 1.2]$

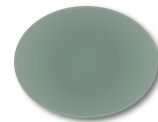
Bird  $\rightarrow 2 \rightarrow [0, 1, 0, 0, \dots] \rightarrow [-1.0, 2.4]$

Cat  $\rightarrow 3 \rightarrow [0, 0, 1, 0, \dots] \rightarrow [-3.0, 4.0]$

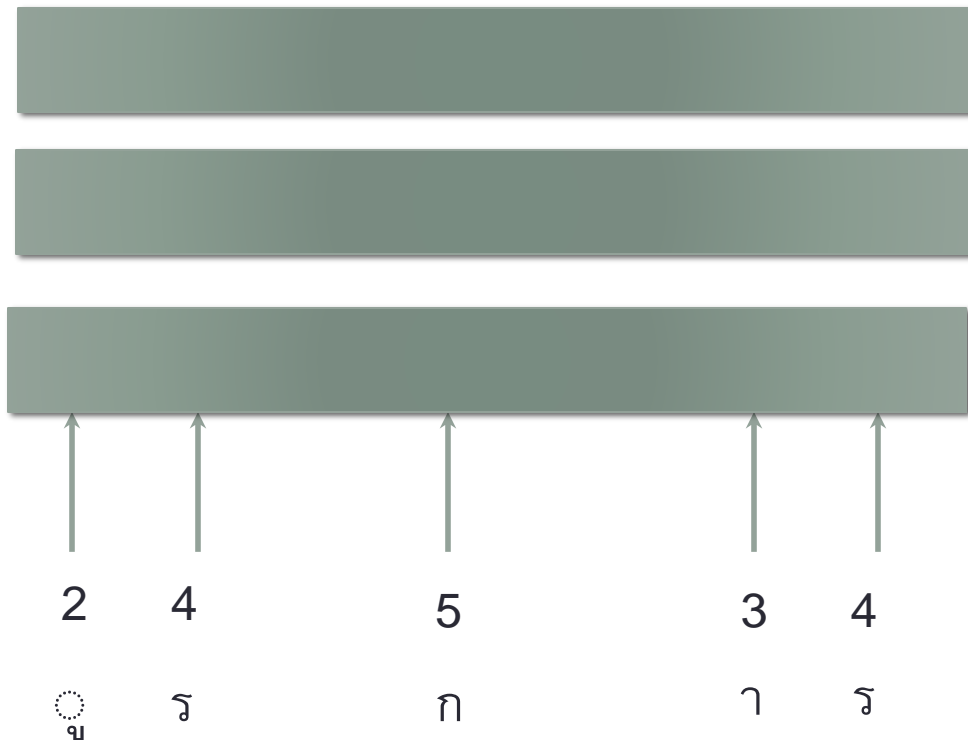
- We can do this by using an embedding layer
  - This is just a (learnable) **lookup table!**

# Word segmentation with fully connected networks

1 = word beginning, 0 = word middle

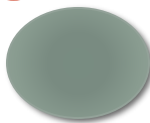


Logistic function





# Adding embedding layer



Embedding layer  
shares the same  
weights

Parameter sharing!

$[1, -1]$   $[3, -2]$   $[5.3, -2.1]$   $[5.3, -3.1]$   $[3, -2]$



2

4

5

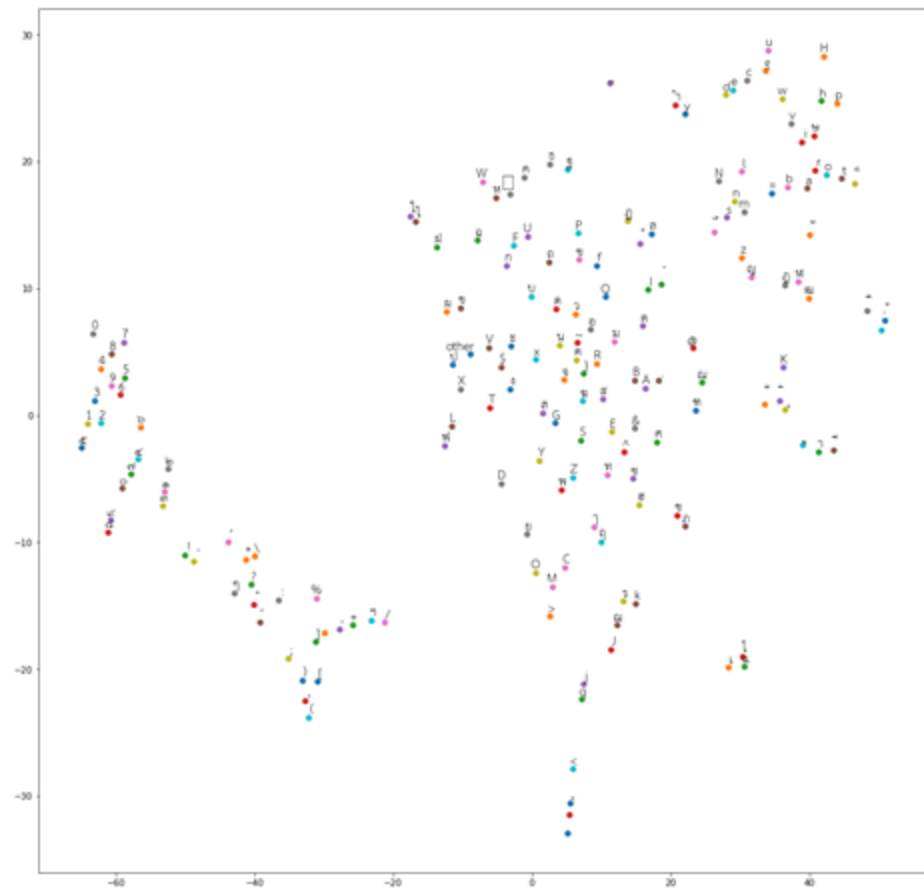
3

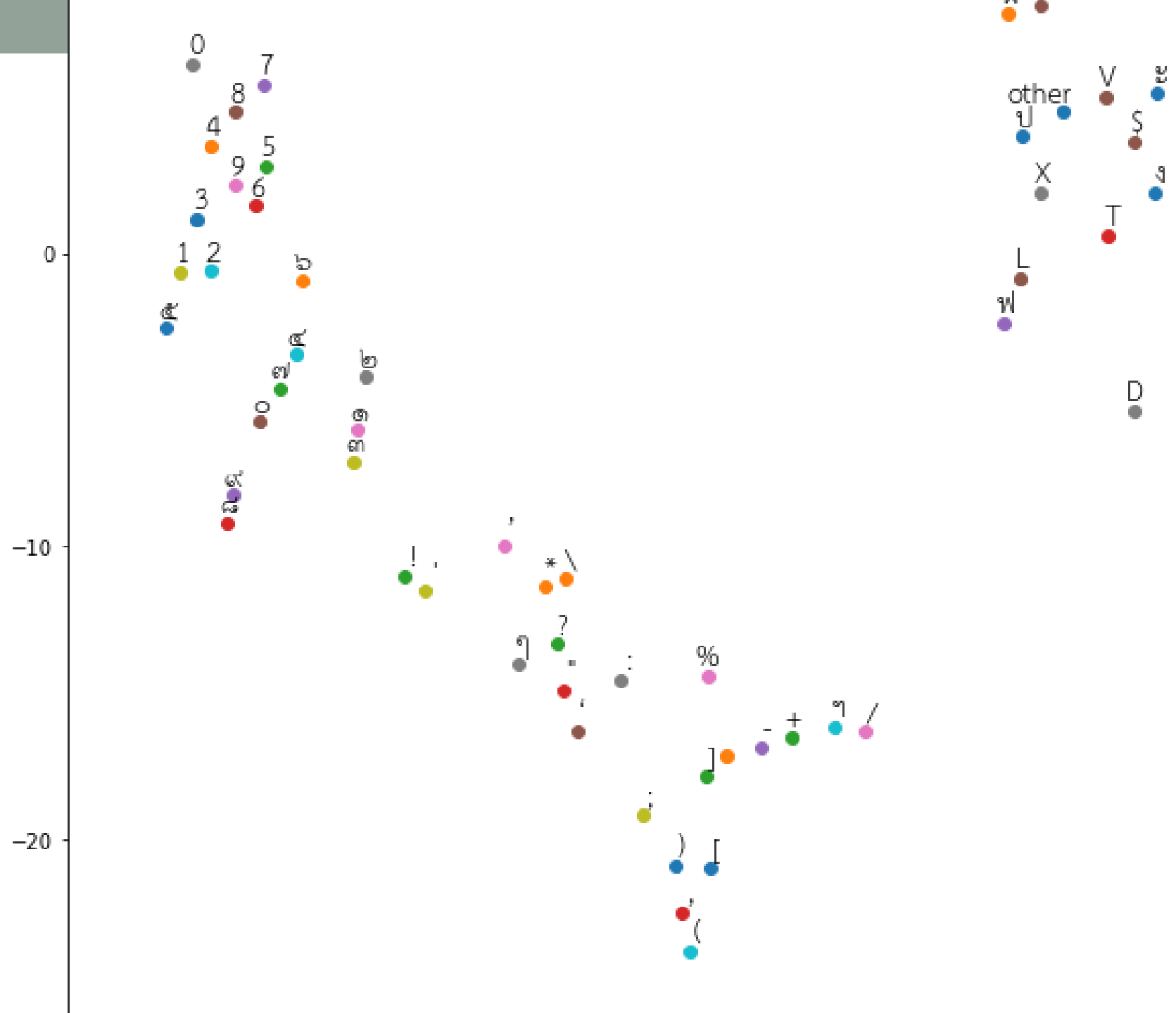
4

More on embeddings  
in the next two  
lectures!

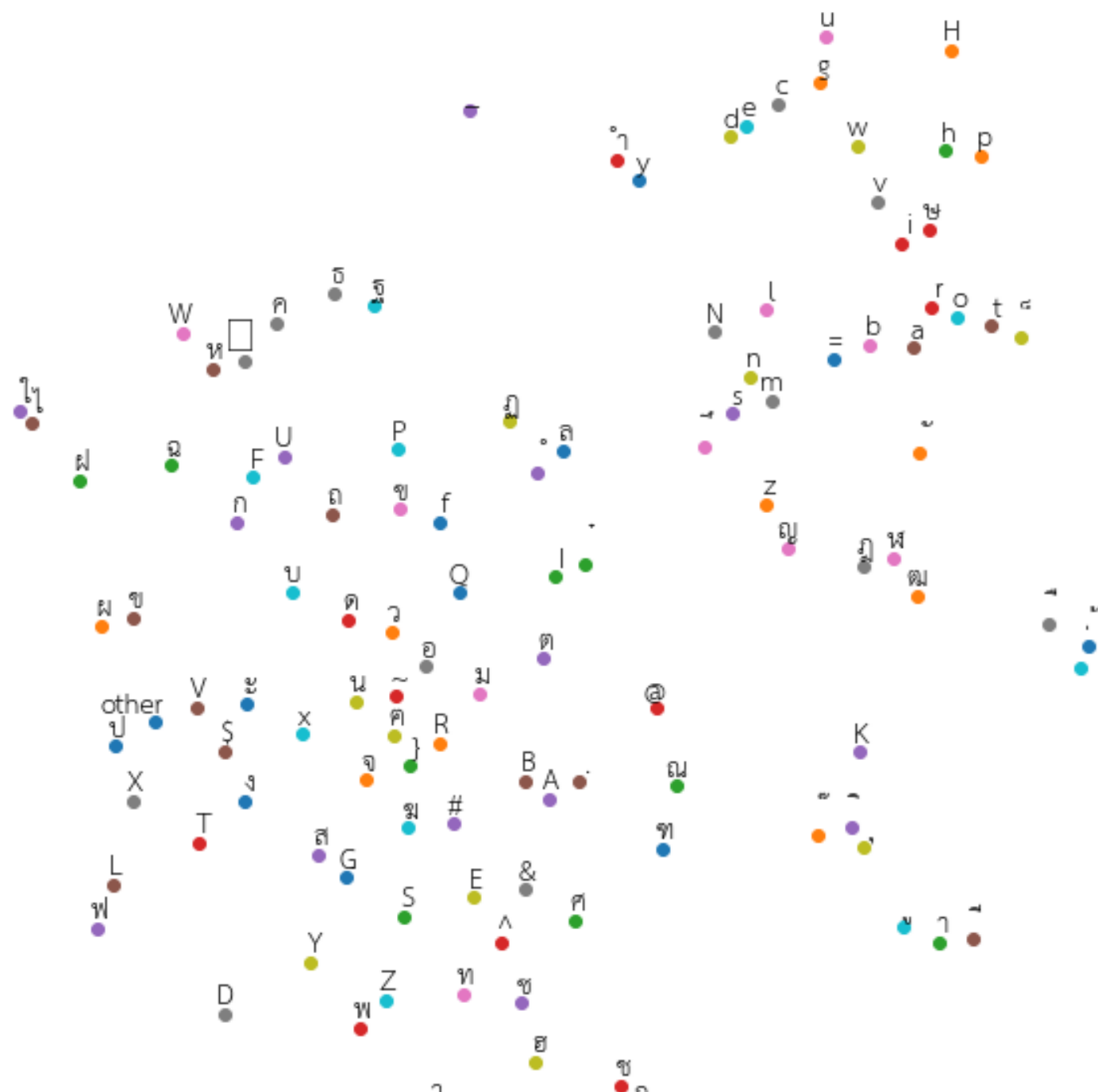
# Embedding and meaning (semantics)

- Meaning is inferred from the task
- Embedding of 32 dimensions -> t-SNE into 2 dimension for visualization
- Automatically!









# Debugging guide

- [https://uvadlc-notebooks.readthedocs.io/en/latest/tutorial\\_notebooks/guide3/Debugging\\_PyTorch.html](https://uvadlc-notebooks.readthedocs.io/en/latest/tutorial_notebooks/guide3/Debugging_PyTorch.html)  
has list of common errors and best practices
- <http://karpathy.github.io/2019/04/25/recipe/>  
has guide for end-to-end model building (start simple and go more advance)



# Back to tokenization...

TABLE II  
RESULTS OF THE SIX BEST TEAMS

| Type of participants                 | F-Measure (%) | Time (mm:ss) |
|--------------------------------------|---------------|--------------|
| <i>Non-Students<sup>a</sup></i>      | 97.94937      | 00:47        |
| <i>Non-Students</i>                  | 97.84097      | 02:46        |
| <i>Non-Students</i>                  | 97.18822      | 00:26        |
| <i>Bachelor Students<sup>b</sup></i> | 95.78162      | 01:08        |
| <i>Master Students</i>               | 95.56670      | 12:14        |
| <i>PhD+Master Students</i>           | 92.02067      | 02:28        |

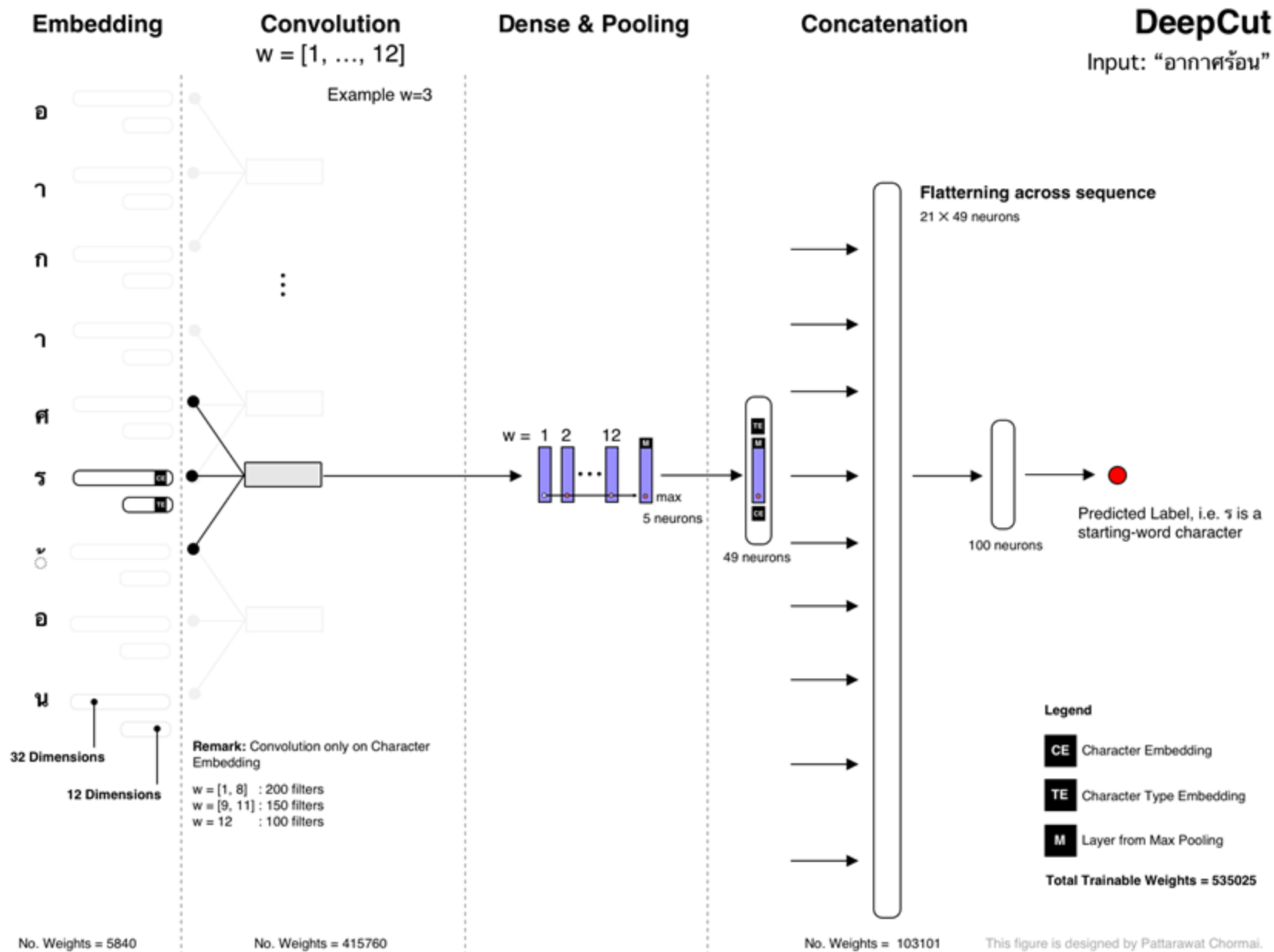
<sup>a</sup>Best of the BEST 2009 Award Winner

<sup>b</sup>BEST Student 2009 Award Winner

**BEST 2009 : Thai word segmentation software contest**

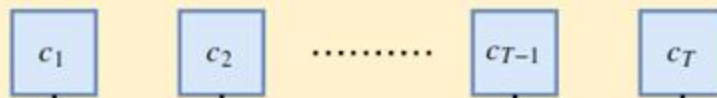
<http://ieeexplore.ieee.org/document/5340941/>

[https://sertiscorp.com/thai-word-segmentation-with-bi-directional\\_rnn/](https://sertiscorp.com/thai-word-segmentation-with-bi-directional_rnn/)

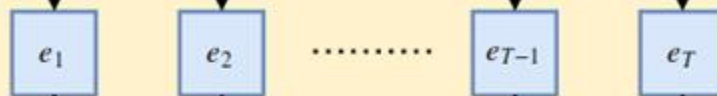




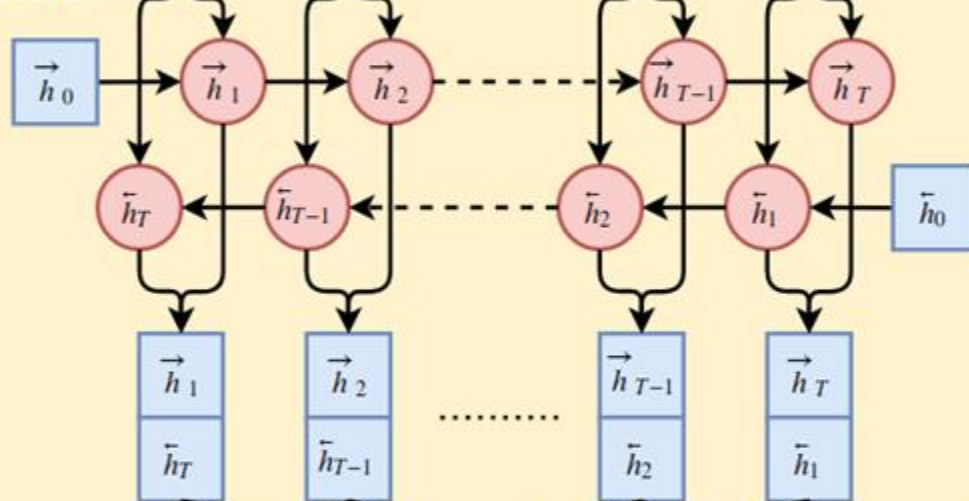
Input sequence



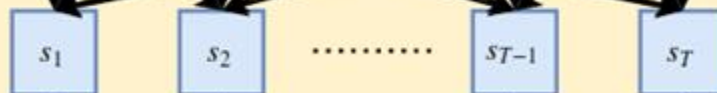
Embedding lookup



Bi-directional RNN



Output score



Softmax output



<https://www.sertiscorp.com/november-20-2017>

$$e_t = W_c c_t$$

$$z_t = \text{sigmoid}(W_z e_t + U_z h_{t-1} + b_z)$$

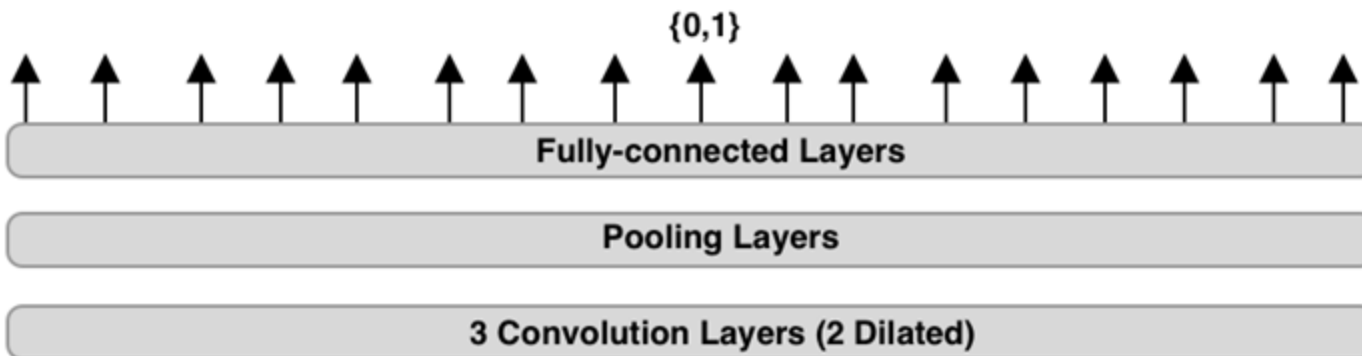
$$r_t = \text{sigmoid}(W_r e_t + U_r h_{t-1} + b_r)$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tanh(W_h e_t + U_h (r_t \odot h_{t-1}) + b_h)$$

$$H_t = [\vec{h}_t, \vec{h}_{T-t+1}]$$

$$s_t = H_t W_s + b_s^T$$

$$p_t = \frac{\exp(s_t)}{\exp(s_{t1}) + \exp(s_{t2})}$$



### Embedding

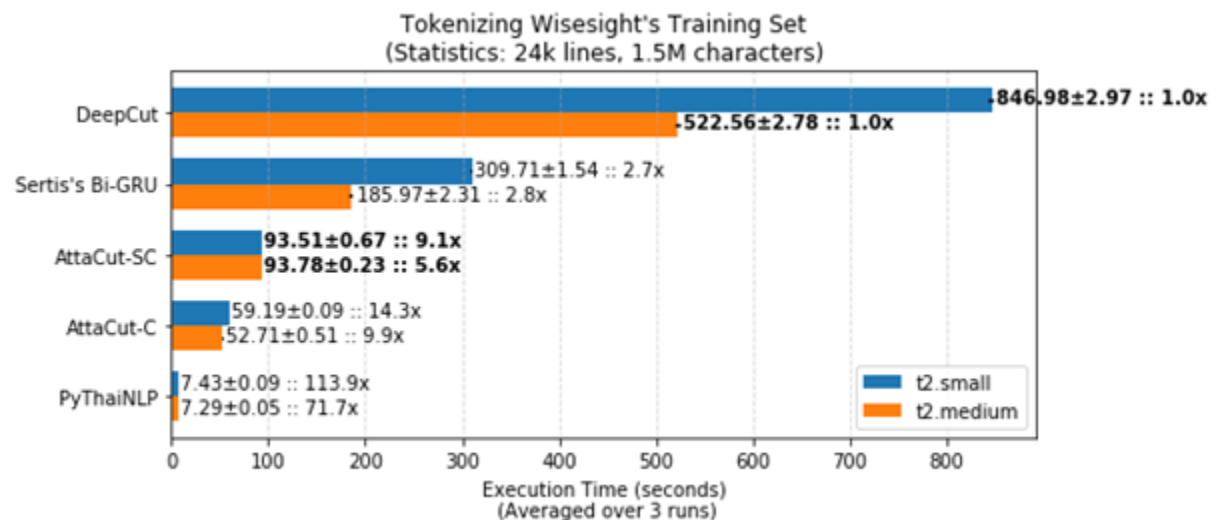
Character

Syllable

ก ฏ ล ะ ฏ ล ฏ

Concatenation

|                          |           | Others             |                  |                  | Ours             |                   |
|--------------------------|-----------|--------------------|------------------|------------------|------------------|-------------------|
| Last Updated: 29/08/2019 |           | PyThaiNLP<br>newmm | Sertis<br>Bi-GRU | DeepCut          | <u>AttaCut-C</u> | <u>AttaCut-SC</u> |
| BEST Validation Set      |           |                    |                  |                  |                  |                   |
| Character-Level          | precision | 0.94±0.11          | 0.95±0.10        | 0.99±0.05        | 0.97±0.07        | 0.98±0.05         |
|                          | recall    | 0.83±0.09          | 0.99±0.02        | 0.99±0.03        | 0.98±0.04        | 0.99±0.03         |
|                          | <b>f1</b> | 0.88±0.08          | 0.97±0.07        | <b>0.99±0.04</b> | 0.98±0.05        | 0.99±0.04         |
| Word-Level               | precision | 0.73±0.16          | 0.91±0.14        | 0.97±0.07        | 0.94±0.10        | 0.96±0.08         |
|                          | recall    | 0.65±0.16          | 0.94±0.10        | 0.97±0.07        | 0.94±0.09        | 0.97±0.08         |
|                          | <b>f1</b> | 0.68±0.15          | 0.93±0.12        | <b>0.97±0.07</b> | 0.94±0.10        | <b>0.97±0.08</b>  |
| BEST Test Set            |           |                    |                  |                  |                  |                   |
| Character-Level          | precision | 0.91±0.15          | 0.92±0.11        | 0.96±0.08        | 0.94±0.10        | 0.95±0.09         |
|                          | recall    | 0.85±0.09          | 0.98±0.04        | 0.98±0.04        | 0.98±0.04        | 0.98±0.04         |
|                          | <b>f1</b> | 0.86±0.11          | 0.95±0.08        | <b>0.97±0.06</b> | 0.96±0.07        | 0.96±0.07         |
| Word-Level               | precision | 0.70±0.19          | 0.85±0.18        | 0.92±0.14        | 0.88±0.17        | 0.91±0.15         |
|                          | recall    | 0.64±0.18          | 0.90±0.14        | 0.93±0.12        | 0.91±0.14        | 0.92±0.13         |
|                          | <b>f1</b> | 0.67±0.19          | 0.87±0.16        | <b>0.93±0.13</b> | 0.89±0.16        | 0.91±0.14         |



ผมเห็นคนวงการนี้เมื่อ 20-30 ปีที่แล้วทำเรื่องตดคำ วงการนี้มันไม่ไปไหนเลยใช่ไหมเนี่ย

มิตรสหาย Business Development ท่านหนึ่ง



# Words of caution

Statistical tokenizers fail on mismatched data

A tokenizer trained on social text might not be able to cut simple words like

<https://www.aclweb.org/anthology/2020.emnlp-main.315/>  
<https://github.com/mrpeerat/OSKut>

มะม่วง มะละกอ

|         | WS160 | TNHC |
|---------|-------|------|
| Deepcut | 93.8  | 93.5 |
| Attacut | 93.5  | 80.8 |

Statistical tokenizers fails unpredictably

หมูกรอบ => |หมู|กรอบ|

ข้าวผัดคะน้าหมูกรอบหนึ่งจาน => |ข้าวผัด|คะน้า|หมู|กรอบ|หนึ่ง|จาน|

Might need rule-based to override (Deepcut has this)

For speed, maximal matching (newmm) is reliable.

- drawbacks?

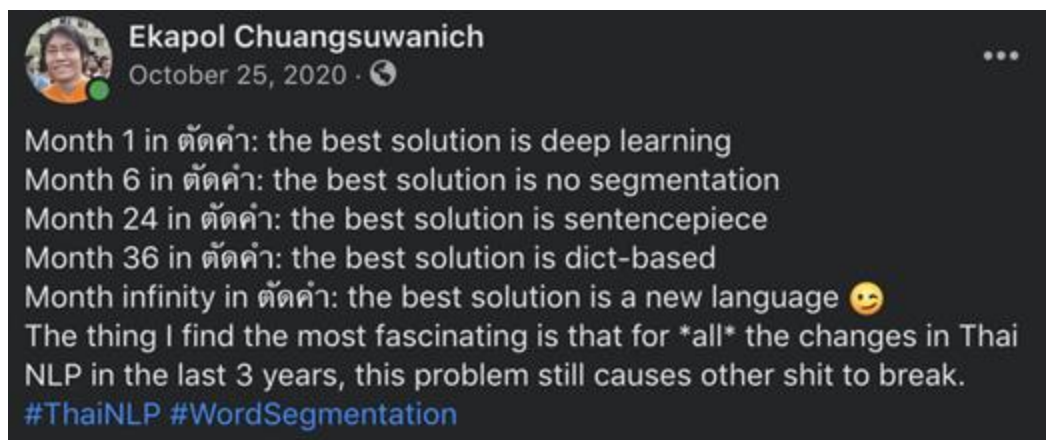
# Words of caution

Tokenization performance effects downstream task performance

Can be small (1%) or large (10%)

Specialized tokenizer can help your downstream task

Example: e-commerce search |หุ|ฟ้ง| |ต่าง|หุ|



# Words of caution

Be careful of what tokenization you used to train the model.  
If there's a mismatch in training and testing tokenization, the results can be devastating.

## TrueVoice

| Training | Testing tokenization |                               |                               |                               |
|----------|----------------------|-------------------------------|-------------------------------|-------------------------------|
|          |                      | Longest matching+<br>noise0.1 | Longest matching+<br>noise0.4 | Longest matching+<br>noise0.7 |
| Deepcut  | Deepcut              | 76.8                          | 60.4                          | 50.9                          |
|          |                      |                               |                               | 42                            |

## Wisesight1000

| Training | Testing |                               |                               |                               |
|----------|---------|-------------------------------|-------------------------------|-------------------------------|
|          |         | Longest matching+<br>noise0.1 | Longest matching+<br>noise0.4 | Longest matching+<br>noise0.7 |
| Manual   | Manual  | 52.1                          | 48.2                          | 38.1                          |
|          |         |                               |                               | 32.7                          |

# Another important note

- In Thai, due to visualization magic, these are the same
  - น + ำ + ั
  - น + ั + ำ
  - ๒ + ๒ + ก
  - ๒๒ + .๑
  - ก + ัก + ัก + ัก + ัก
  - ก + ัก

- You might

`pythainlp.util.normalize(text: str) → str` [\[source\]](#)

Normalize and clean Thai text with normalizing rules as follows:

- Remove zero-width spaces
- Remove duplicate spaces
- Reorder tone marks and vowels to standard order/spelling
- Remove duplicate vowels and signs
- Remove duplicate tone marks
- Remove dangling non-base characters at the beginning of text



# Tokenization - English

- Even English has tokenization issues!
  - Space is usually not enough
  - aren't
    - are + n't
    - aren't
    - arent
    - aren t
    - are + not
  - San Francisco
- Usually includes the text normalization step
- This depends on application
  - “aren't” might be different from “are not” for sentiment analysis

# End-to-end models

- Classical machine learning systems usually break the problem into smaller subtasks
  - Self-driving:
    - Image -> objects detection -> path finding -> steering
  - Speech2speech translation:
    - Speech A -> text A -> text B -> Speech B
- End-to-end models use one large neural networks process the input and generate the desired output
  - Image -> steering
  - Speech A -> Speech B

# End-to-end NLP?

Discourse

Semantics

Syntax: Constituents

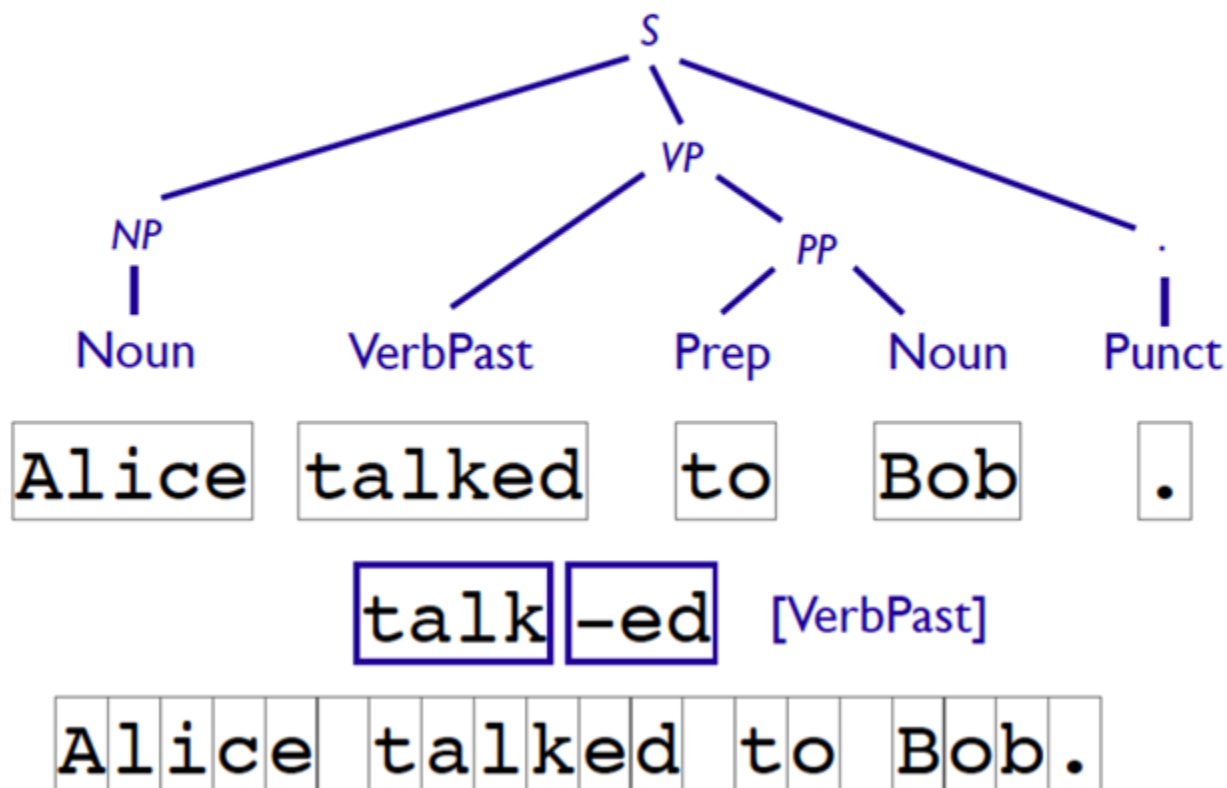
Syntax: Part of Speech

Words

Morphology

Characters

CommunicationEvent(e) SpeakerContext(s)  
Agent(e,Alice) TemporalBefore(e, s)  
Recipient(e, Bob)

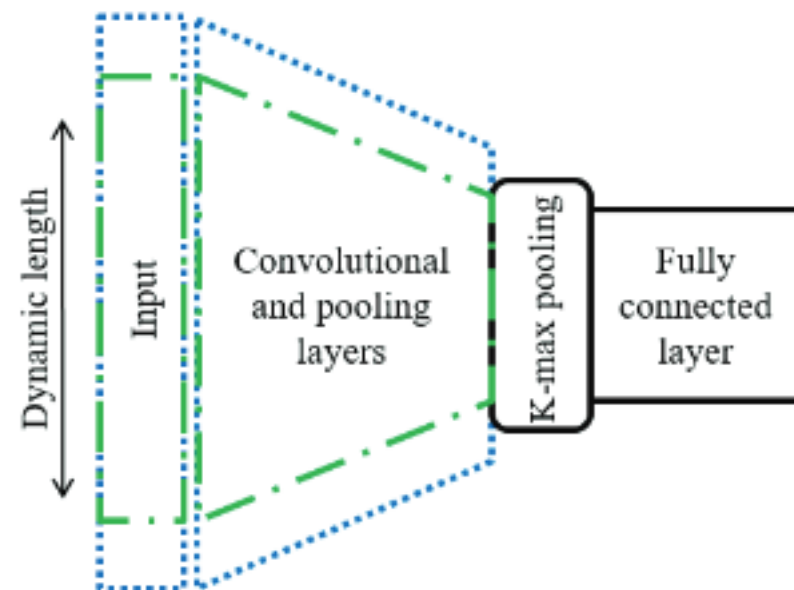


# Towards no tokenization

- Text classification using charCNN on Thai

| Method                           | Accuracy (%) | $F_1$ (%)   |
|----------------------------------|--------------|-------------|
| Naïve Bayes, BoW                 | 87.2         | 87.1        |
| Naïve Bayes, TF-IDF              | 89.0         | 88.9        |
| Logistic Regression, BoW         | 94.8         | 94.8        |
| Logistic Regression, TF-IDF      | 94.7         | 94.7        |
| SVM, BoW                         | 93.7         | 93.7        |
| SVM, TF-IDF                      | 95.2         | 95.2        |
| DCNN (Kalchbrenner et al., 2014) | <b>95.9</b>  | <b>95.9</b> |
| Proposed Char-CNN                | 95.4         | 95.4        |

Word-based methods



A character-level convolutional neural network with dynamic input length for Thai text categorization

<http://ieeexplore.ieee.org/document/7886102/>

# Caveats of end-to-end models

- Requires lots of data for the specific task
- Hard to fix specific mistakes by the model

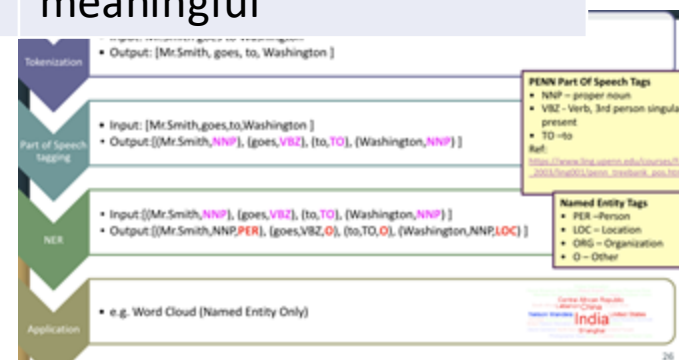
# Things to consider when thinking about tokenization

Know your use cases

Large embedding lookup table

| Word                             | Subword                               | Character   |
|----------------------------------|---------------------------------------|---|
| Large vocabulary (100k)          | Medium vocabulary (20k)               | Small vocabulary (100)                                |
| Can use simpler model            | Moderate complexity in modeling       | Needs a powerful model to learn long range influences |
| High OOVs                        | Few OOVs                              | No OOVs   |
| Individual tokens are meaningful | Individual tokens might be meaningful | Individual tokens are not meaningful                  |

Note: to handle multilingual data, some models use bytes as tokens



# Conclusion

- Tokenization is far from solved but don't let this discourage you
  - No tokenization is perfect
  - Pick one that is suited for your task
    - Speed
    - Robustness to misspelling and unseen words
    - Consistency
    - Certain tools assume you are using a particular type of tokenization, check!