Lecture Notes for

Machine Learning in Python

⟦👨‍🏫, 👨‍💻, 🐍, 👨‍🔬⟧

Professor Eric Larson

Sequential CNN and Transformers

In progress lecture replacing detailed implementation of recurrent networks

Lecture Agenda

- Logistics
  - Grading Update
  - Sequential Networks due **Last Day of Finals**
- Agenda
  - CNNs for Sequential Processing
  - Transformers

Class Overview, by topic

Table Data Visualization

Numpy, Pandas, Seaborn
Overviews with some in-depth discussion

Dimension Reduction and Image Processing

Scikit-learn, Scikit Image,
Intuition only, Some mathematics

Linear and Logistic Regression

Numpy, Recreate API for Scikit-learn
Detailed mathematics for simple optimization
intuition for advanced optimization

Neural Networks and Back Prop.

Numpy
Detailed mathematics for NN operations

Wide and Deep Networks

Convolutional Networks

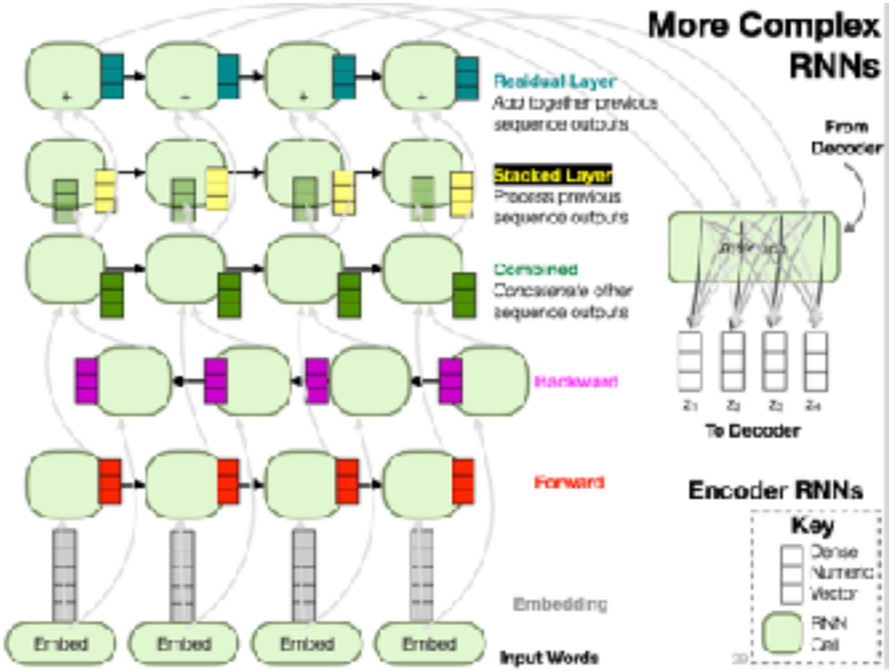
Sequential Networks

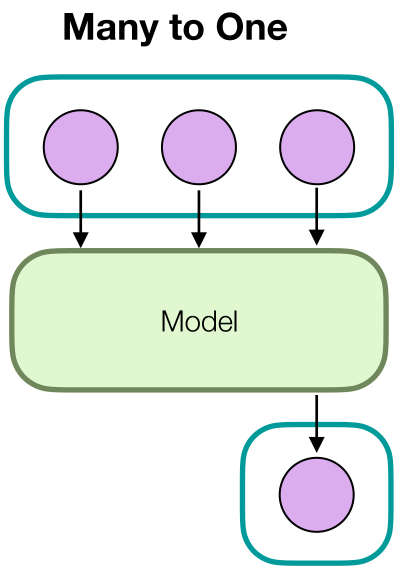
Keras, Tensorflow
Intuition, Detailed implement.

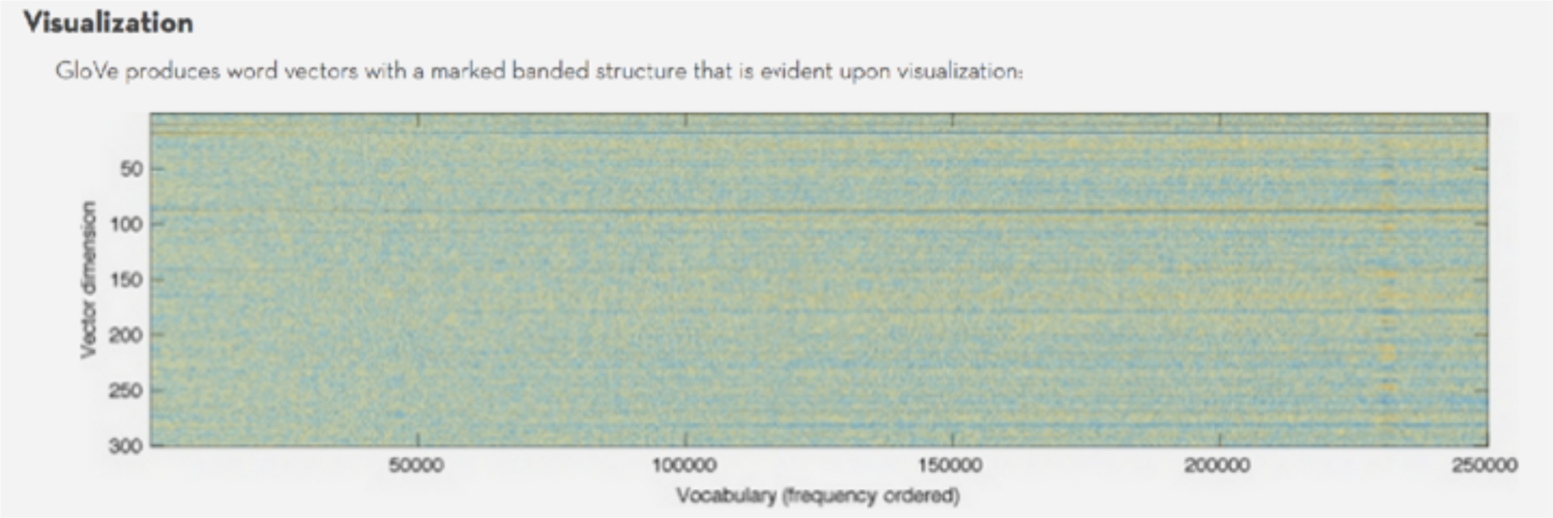
Ethics in Language Models

ConceptNet
Case studies

Last Time:

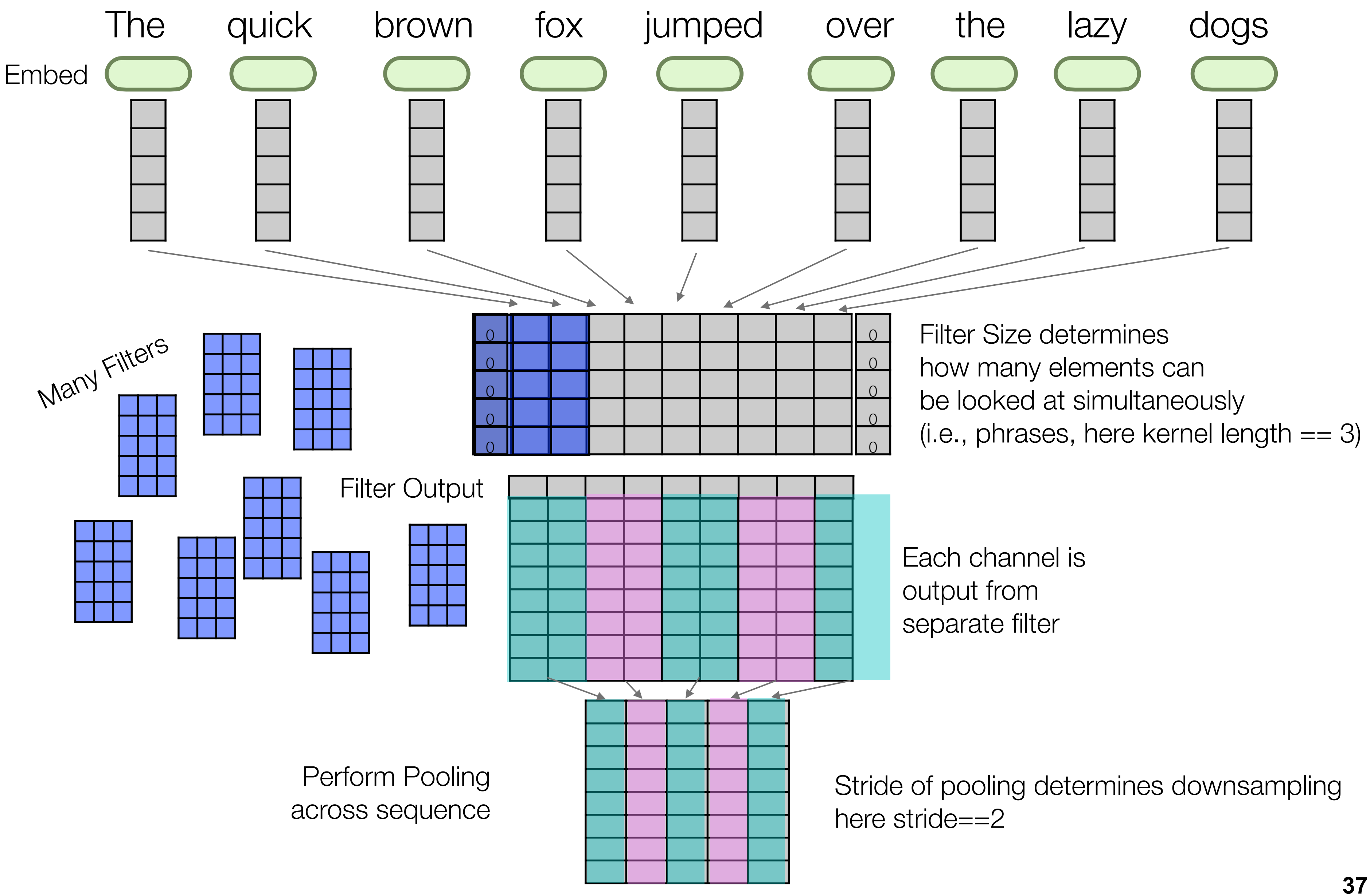

More Complex
RNNs
Residual Layer
Add together previous sequence outputs
Stacked Layer
Process previous sequence outputs
Combined
Concatenate other sequence outputs
Backward
Forward
Embedding
Input Words
Embed
Embed
Embed
Embed
From Decoder
To Decoder
Encoder RNNs
Key
Dense Numeric Vector
RNN Cell


Many to One
Model


Visualization
GloVe produces word vectors with a marked banded structure that is evident upon visualization:
Vector dimension
50
100
150
200
250
300
50000
100000
150000
200000
250000
Vocabulary (frequency ordered)

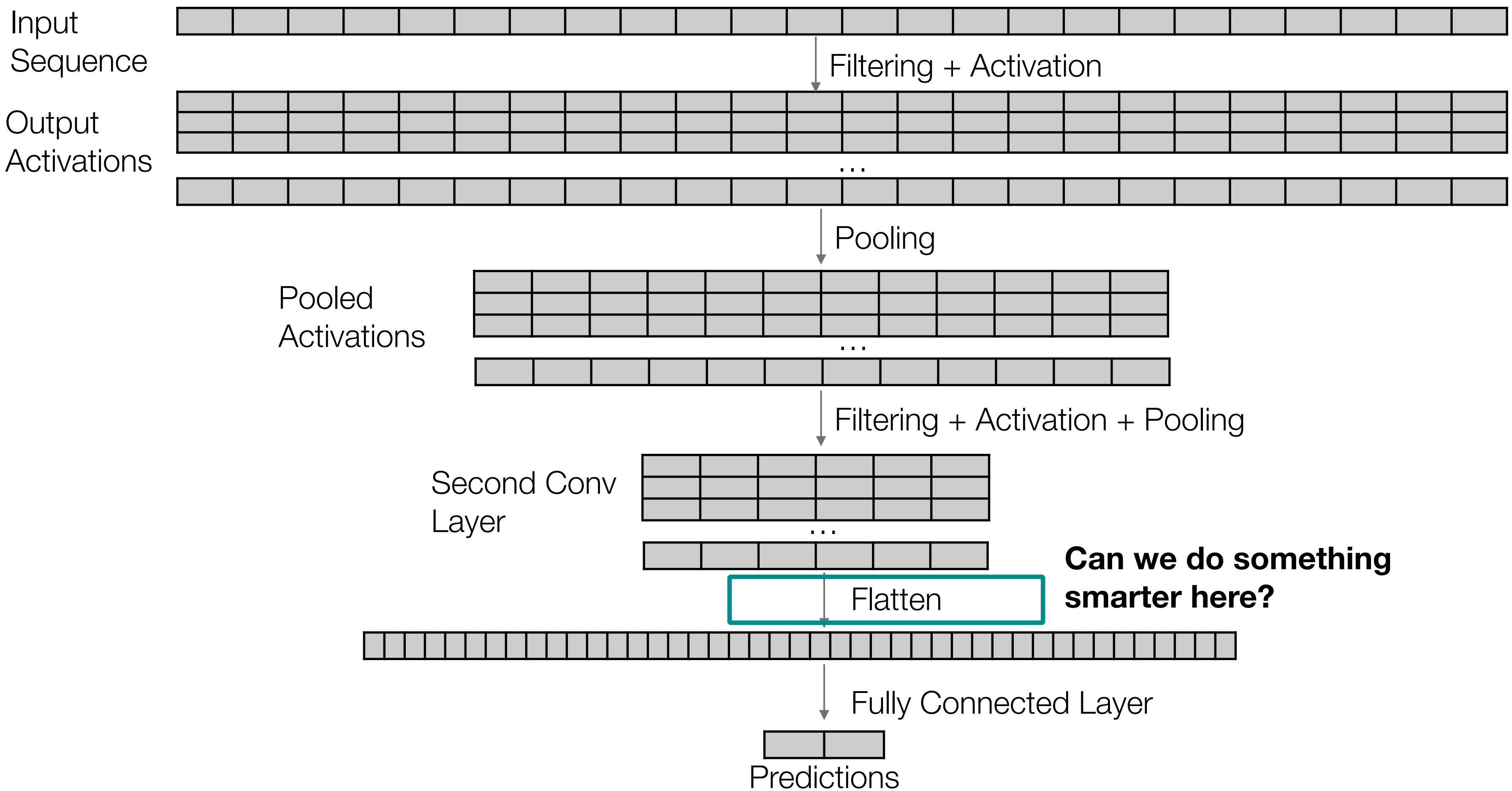
CNNs for Sequences

CNNs for Sequences



CNNs for Sequences

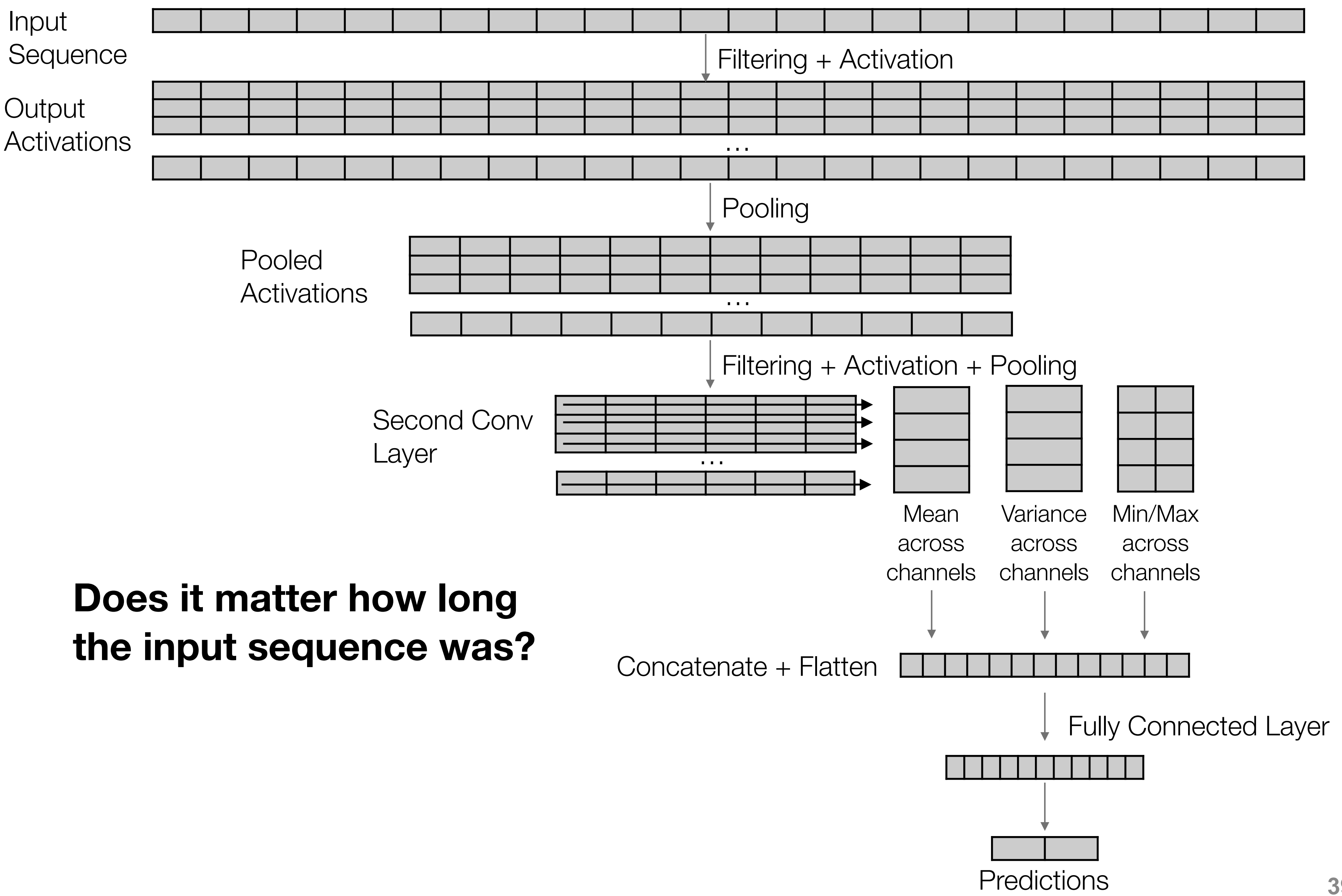
- RNNs are not inherently parallelized or efficient at remembering based on state vector, but CNNs can be run in parallel groups



Can we do something smarter here?

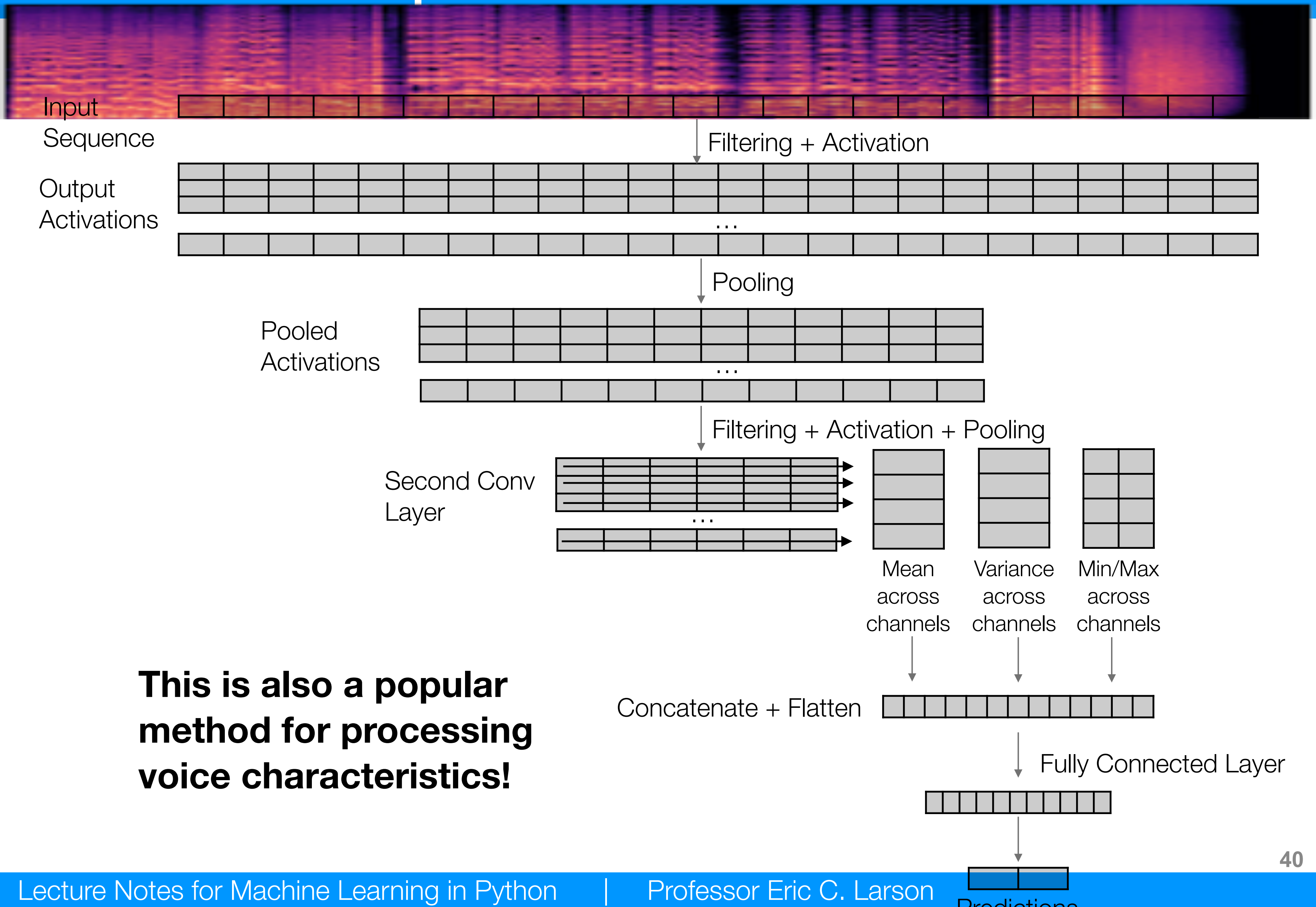
- Everything we learned in 2D CNNs can be applied to 1D CNNs…
- Residuals, separable convolution, squeezing, everything

CNNs for Sequences



Does it matter how long the input sequence was?

CNNs for Sequences



This is also a popular method for processing voice characteristics!

Demo

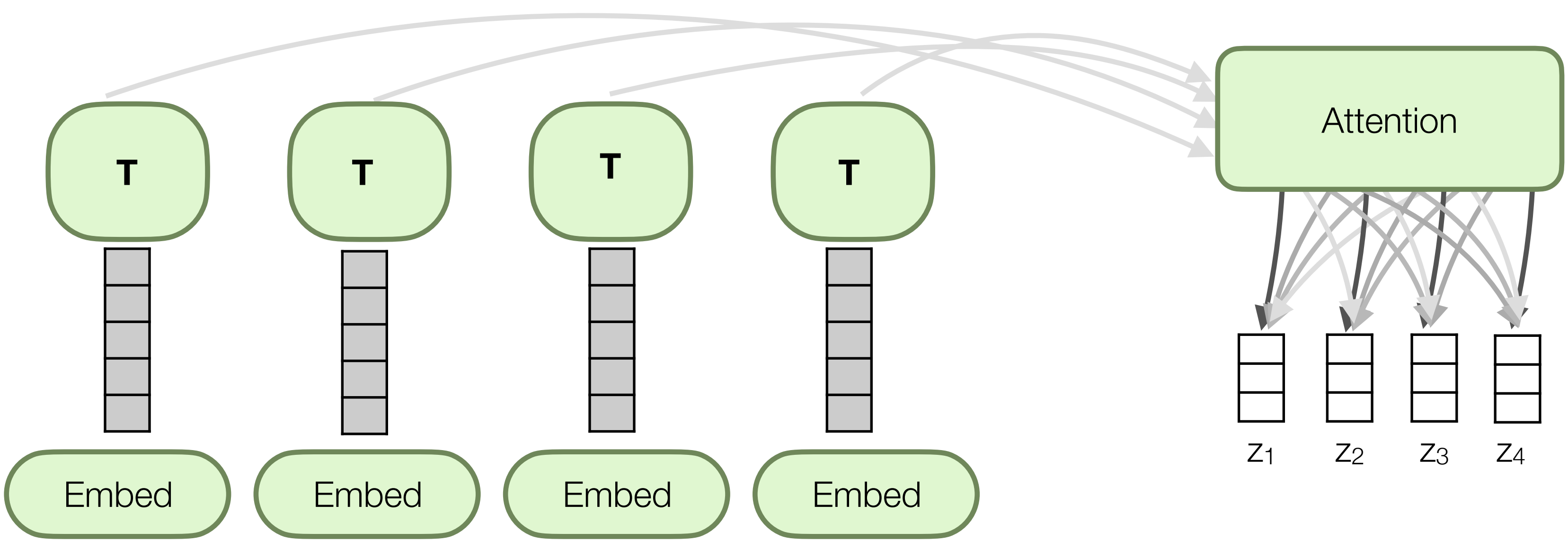
The Sequential CNN
IMdB sentiment analysis

`13a. Sequence Basics [Experimental].ipynb`

Transformers

Transformers Intuition (reminder)

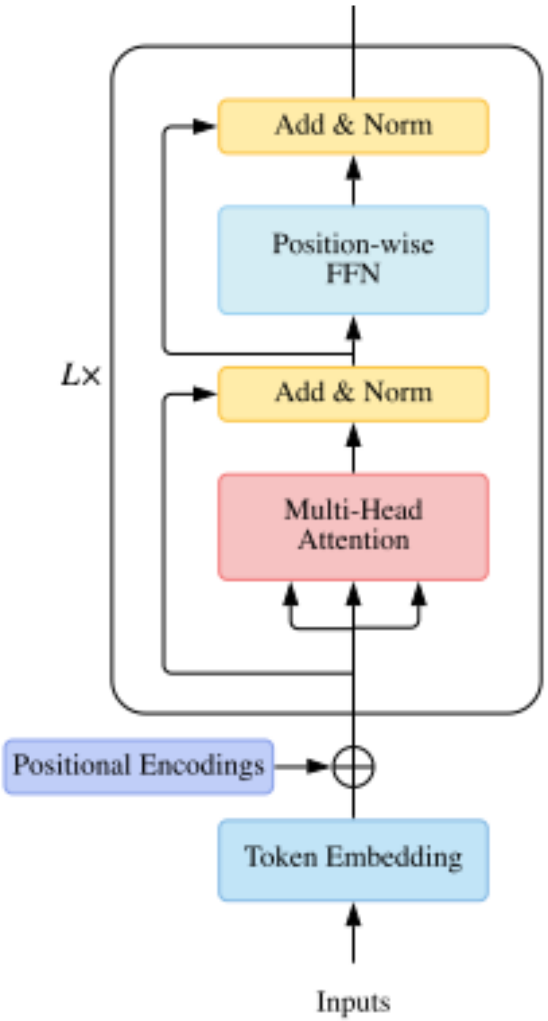
- Recurrent networks track state using an "updatable" state vector, but this takes processing iterative
- Attention mechanism (in RNNs) already takes a weighted sum of state vectors to generate new token in a decoder
- … so why not just use attention on a transformation of the embedding vectors? **Do away with the recurrent state vector all together?**

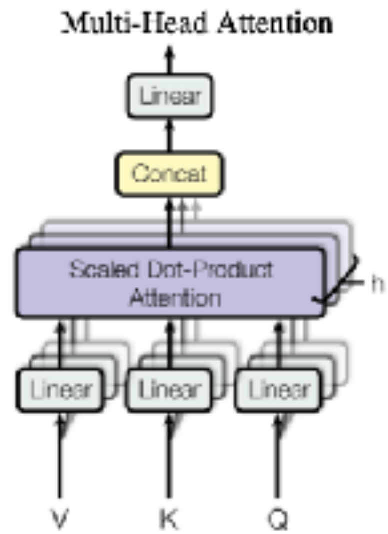


Attention is All You Need

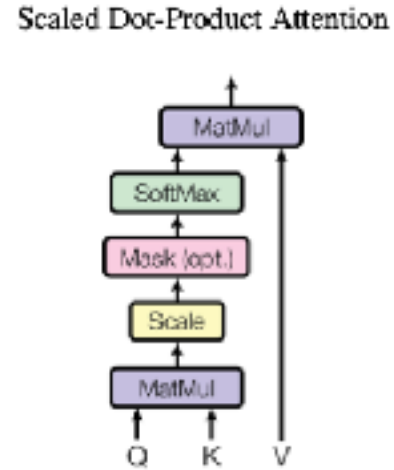
- **Continued Motivation**:
  - RNNs are not inherently parallelized or efficient at remembering based on state vector
  - CNNs are not resilient to long-term word relationships, limited by filter size
- **Transformer Solution:**
  - Build attention into model from the **beginning**
  - Compare all words to each other through **self-attention**
  - Define a notion of “**position**”in the sequence
  - ***Should be resilient to long term relationships and be highly parallelized for GPU computing!!***

Transformer Overview





more than one
Q,K,V use in document



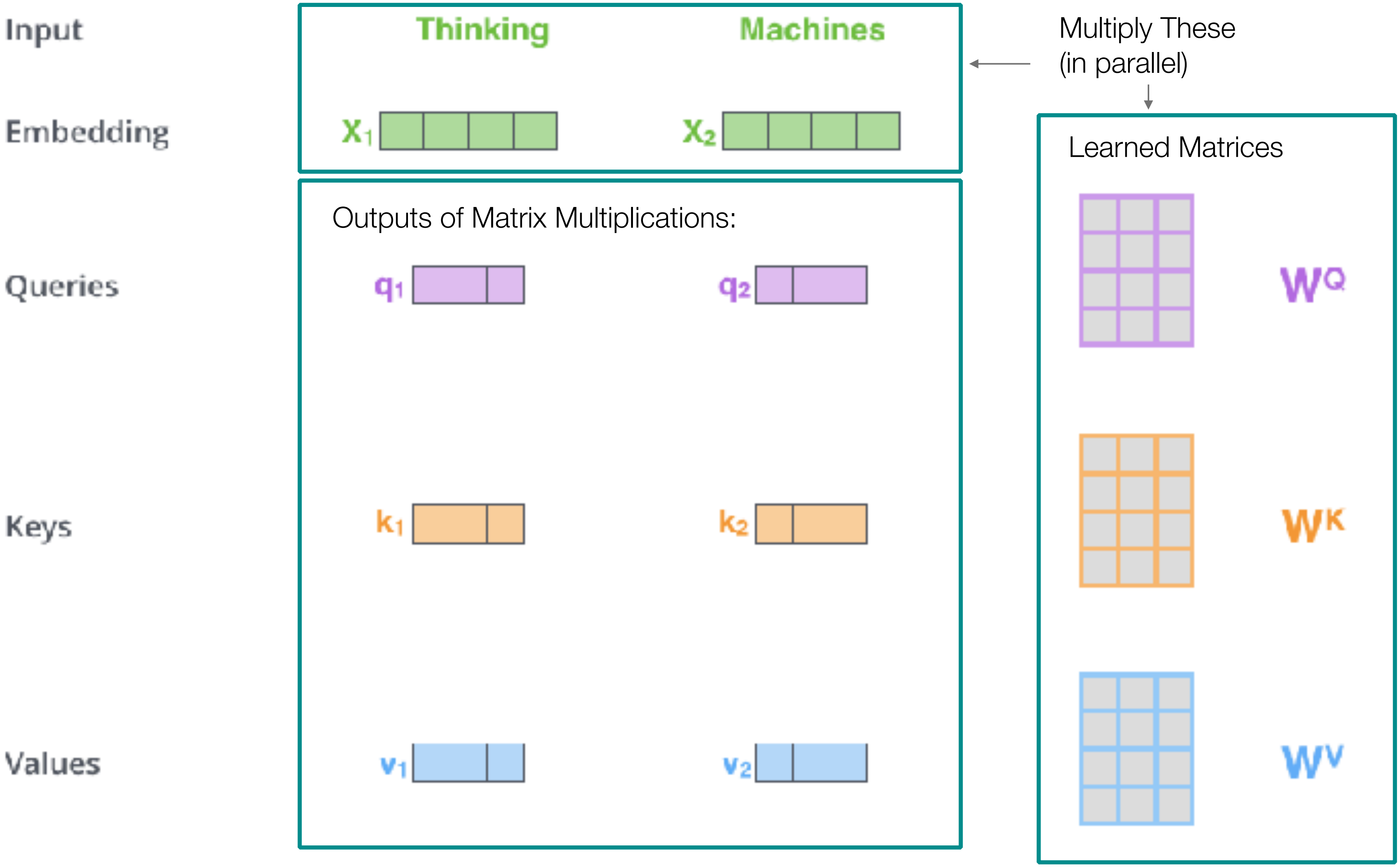
for each word

$$\mathrm{Attention}(Q,K,V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head_1},...,\mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

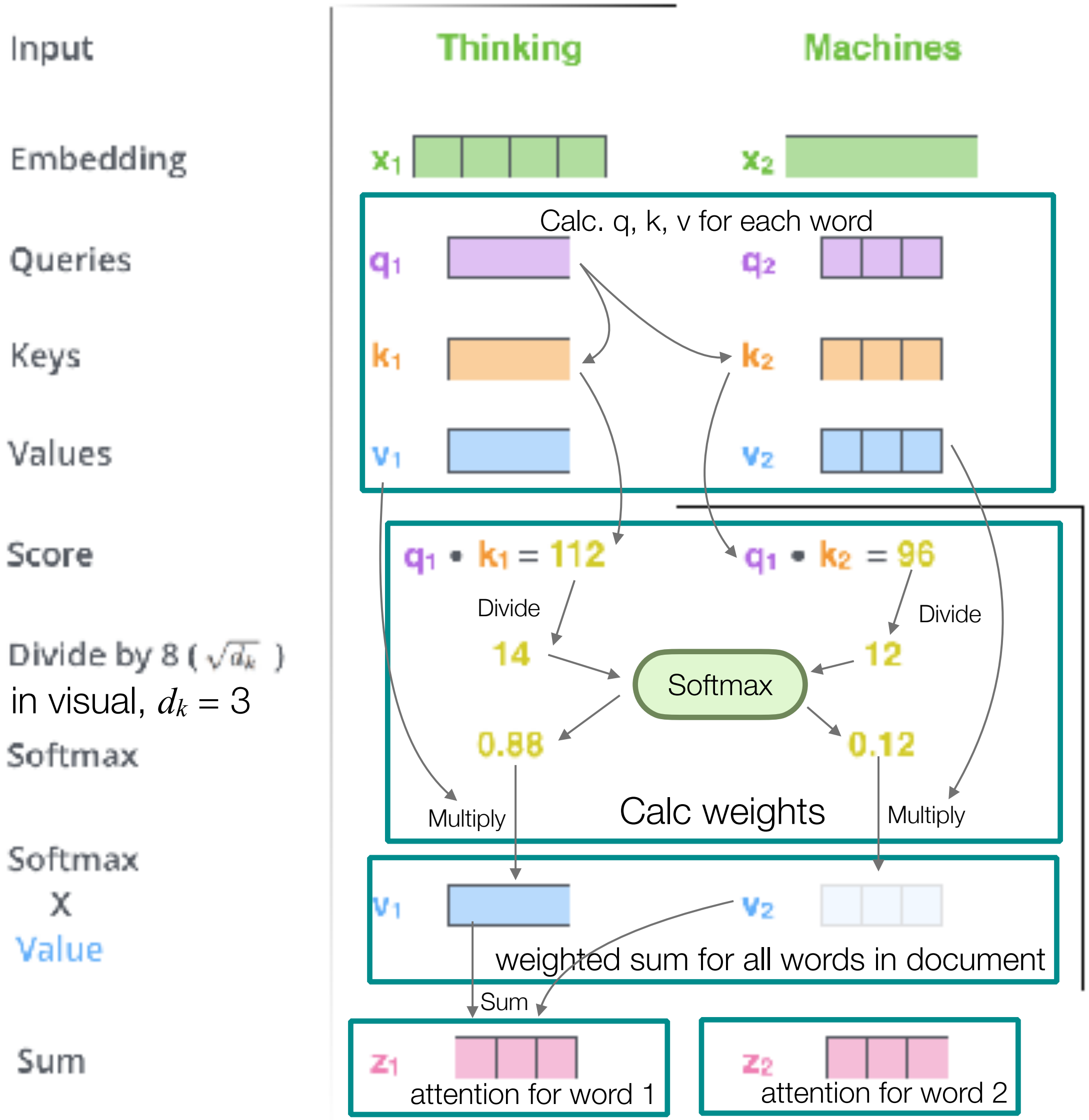
https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

Transformer: in more detail

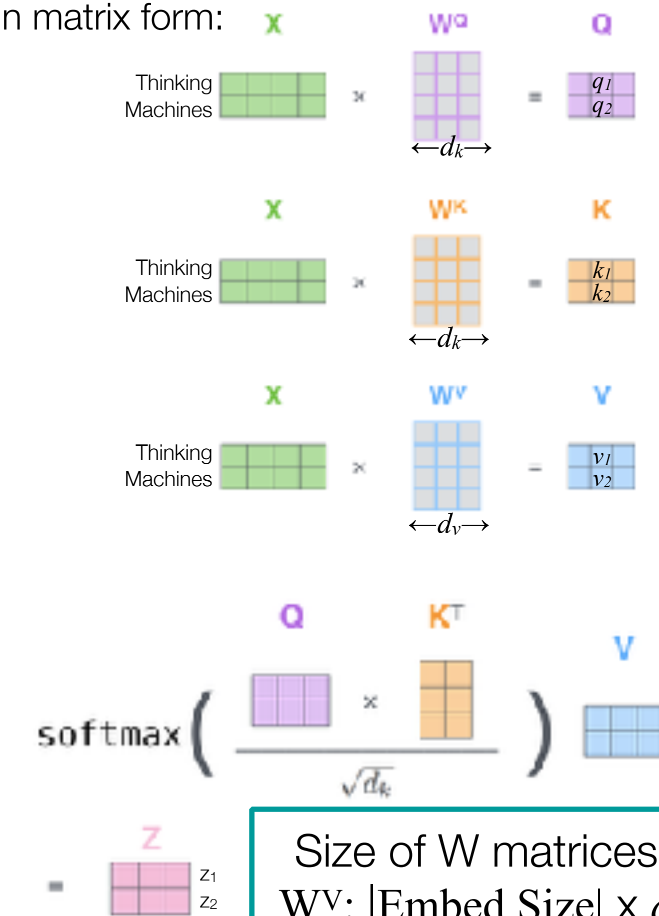


Excellent Blog on Transformers: http://jalammar.github.io/illustrated-transformer/

Transformer: in more detail



Straight forward to do this operation in matrix form:

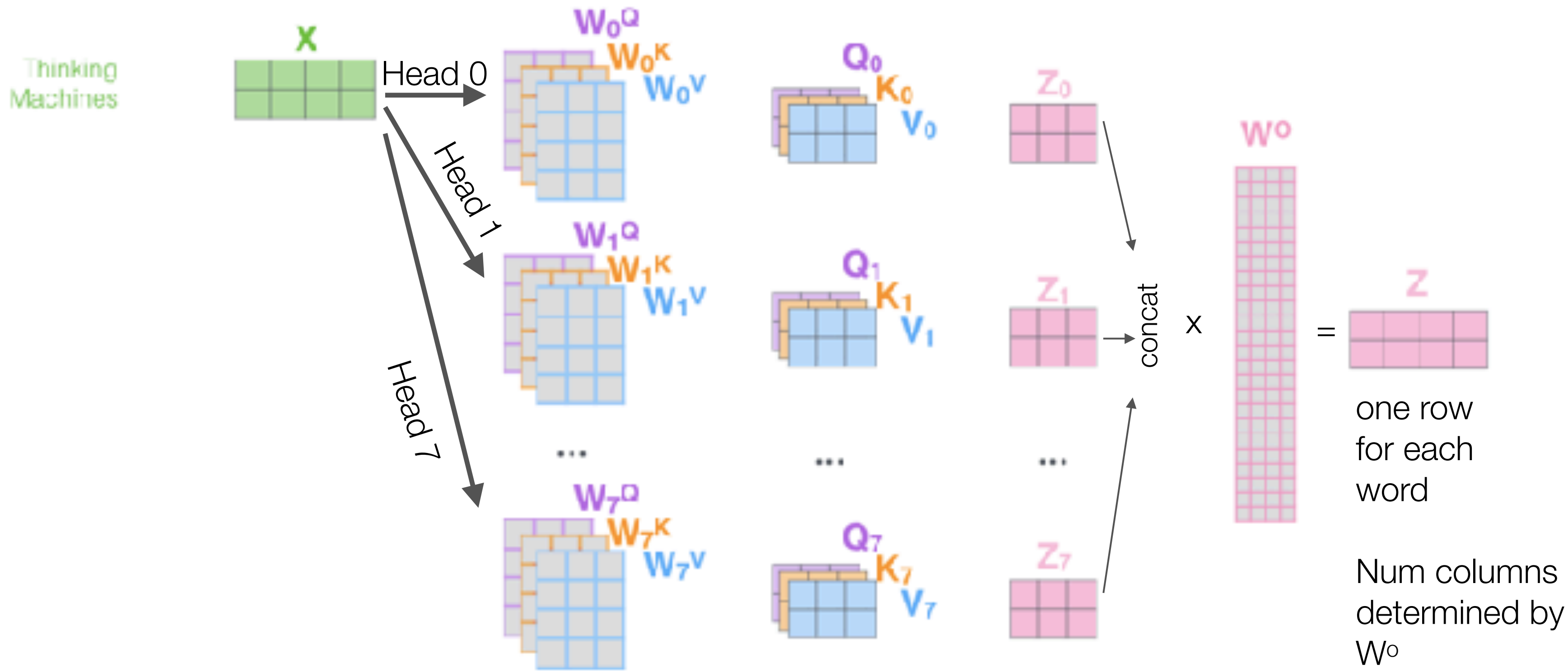


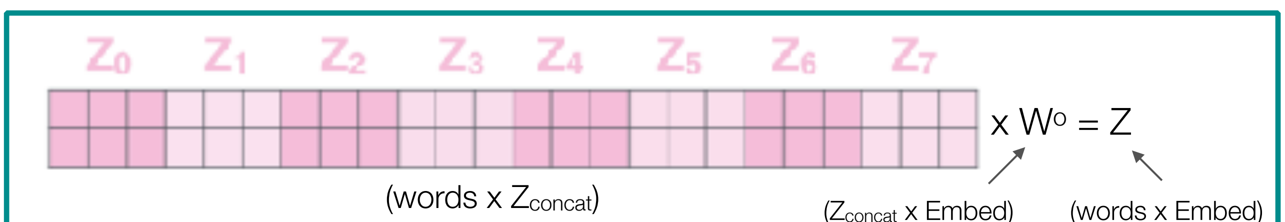
Size of W matrices:
W^V: |Embed Size| x d_v
$W^{Q,K}$: |Embed Size| x d_k

Size of Q,K,V:
|Seq Len| x d_v

Excellent Blog on Transformers: http://jalammar.github.io/illustrated-transformer/

Transformer: Multi-headed Attention





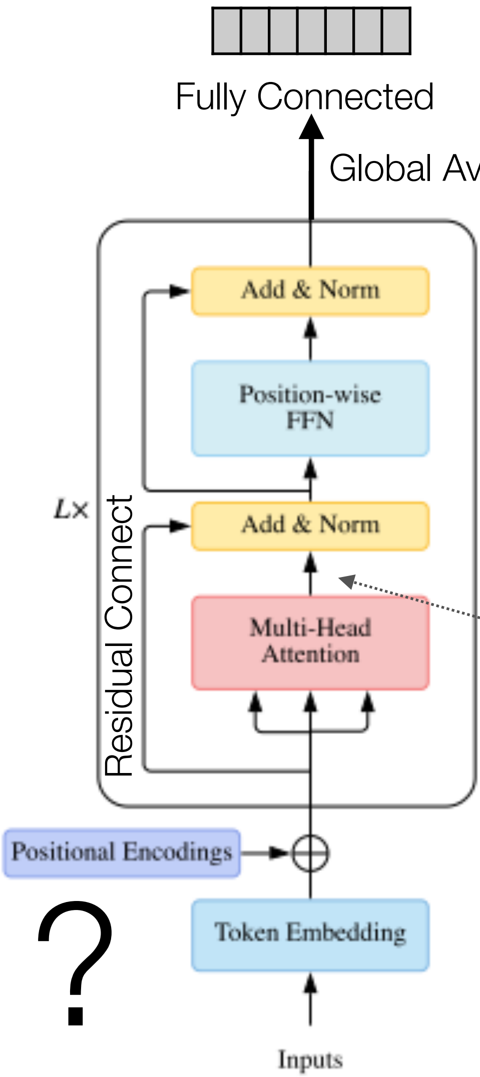
Excellent Blog on Transformers: http://jalammar.github.io/illustrated-transformer/

Putting It Together

Prediction

Fully Connected

Global Average



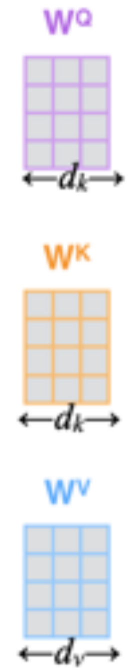
?

Layer Normalization

Fully Connected

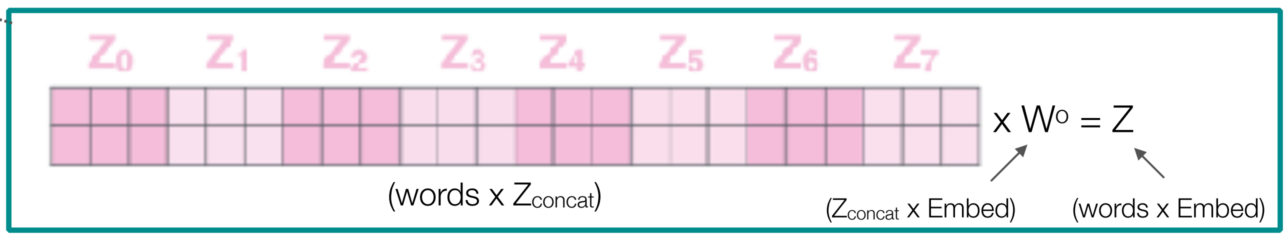
Layer Normalization

```
tf.keras.layers.MultiHeadAttention(
    num_heads,      (Number of heads Z1-Z7)
    key_dim,        (size of query/key dk)
    value_dim,      (size of each dv)
    output_shape,   (Embed size of Z)
  ...
```



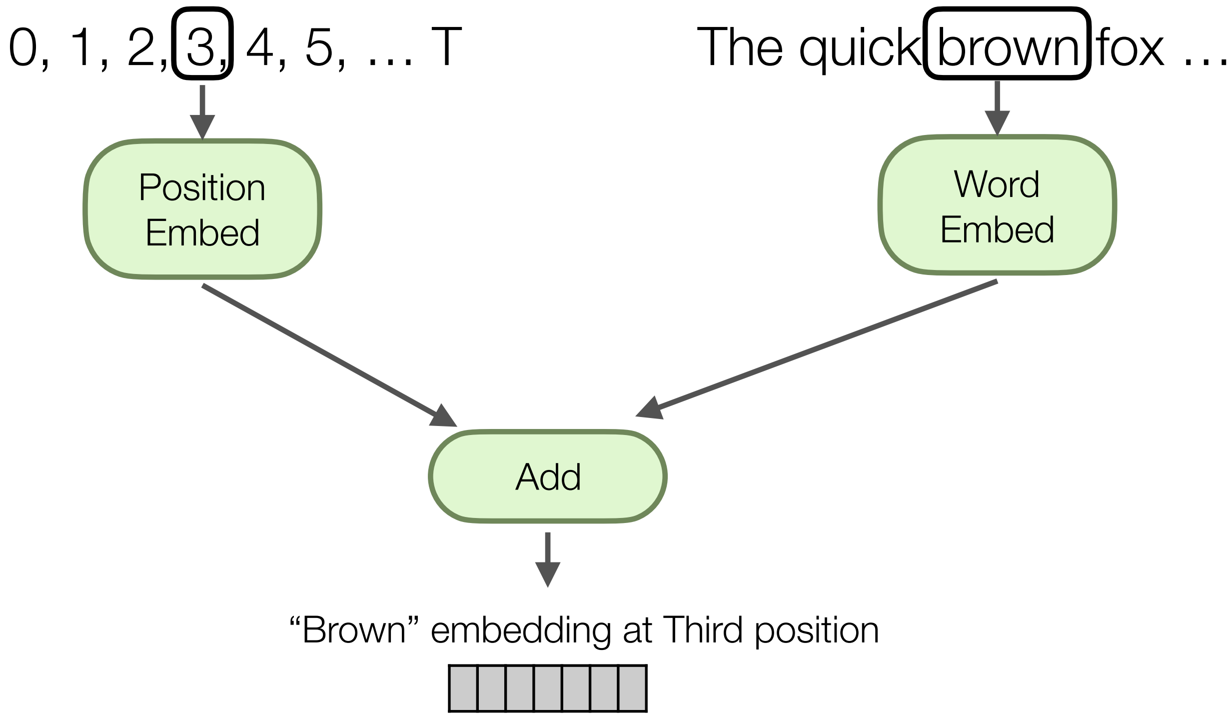
$$LN(Z_{row}) = \alpha \frac{Z_i - \mu_Z}{\sqrt{\sigma_Z^2 + \epsilon}} + \beta$$

Learn to normalize the rows of Z



Transformer: Positional Encoding

- Objective: add notion of position to embedding
- Attempt in original paper: add sin/cos to embedding
- But could be anything that encodes position, like:



Excellent Blog on Transformers: http://jalammar.github.io/illustrated-transformer/

Demo

The Transformer
and 20 news groups with GloVe

`13a. Sequence Basics [Experimental].ipynb`

Lecture Notes for

Machine Learning in Python

⟦👨‍🏫,👨‍💻,🐍,👨‍🔬⟧

Professor Eric Larson

Sequential CNN and Transformers