Lecture Notes for

Machine Learning in Python

⟦👨‍🏫, 👨‍💻, 🐍, 👨‍🔬⟧

Professor Eric Larson

Sequential CNN and Transformers

Lecture Agenda

- Logistics
  - Grading Update
  - Lectures
  - Sequential Networks due **during finals**
- Agenda
  - CNNs for Sequential Processing (review)
  - 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

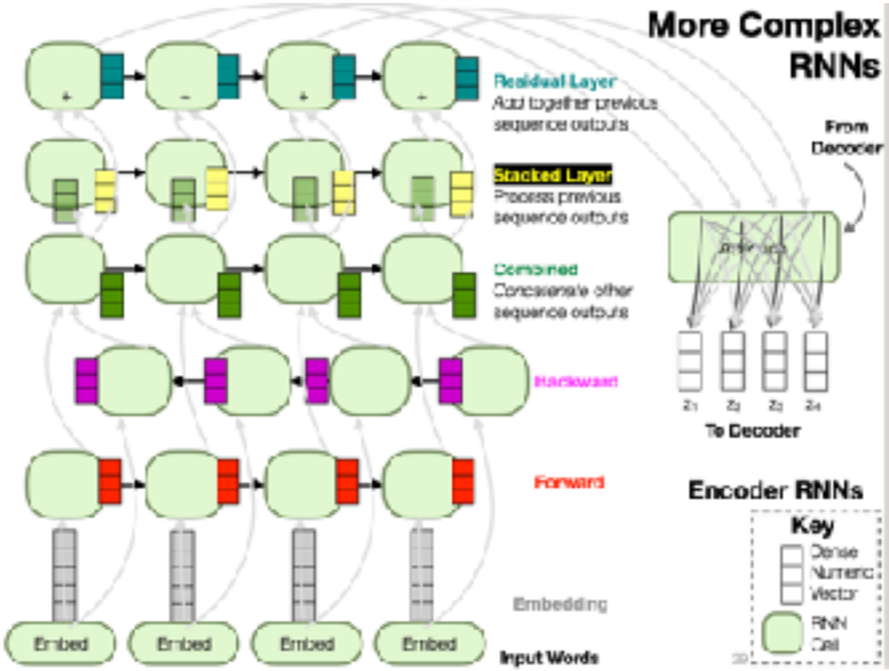
Linear and Logistic Regression
Numpy, Recreate API for Scikit-learn
Detailed mathematics for simple optimization
intuition for advanced optimization

Neural Networks and Back Prop.
Numpy
Detailed mathematics for NN operations

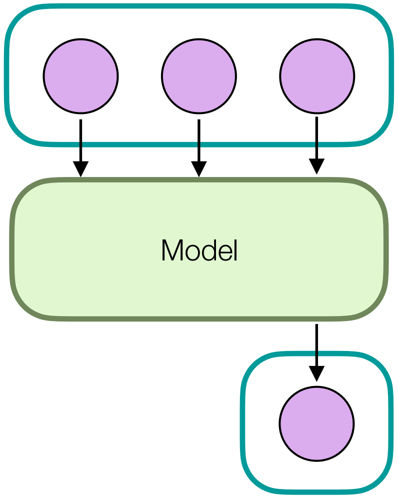
Wide and Deep Networks | **Convolutional Networks** | **Sequential Networks**
Keras, Tensorflow
Intuition, Detailed implement.

Ethics in Language Models
ConceptNet
Case studies

Last Time:

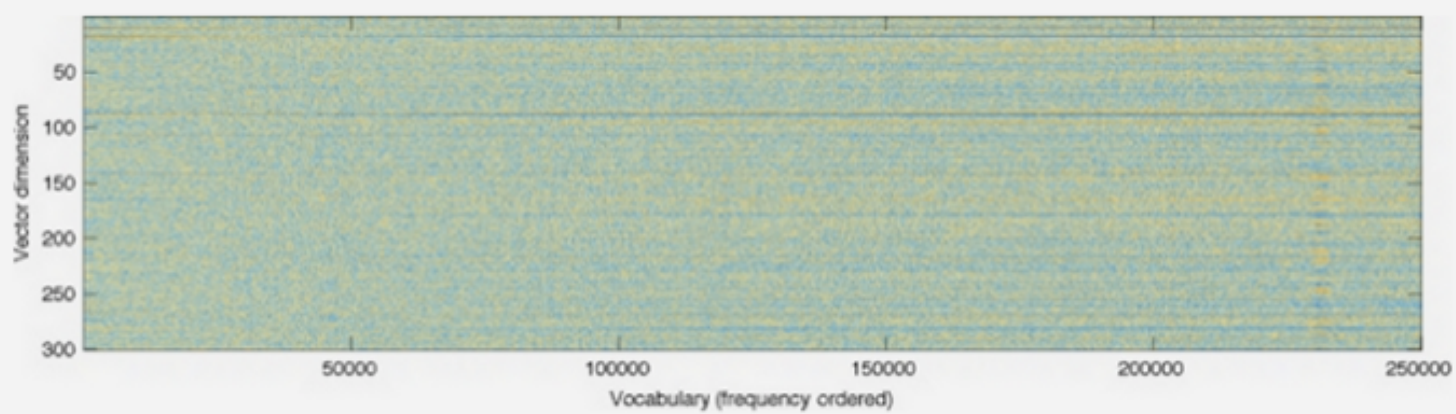


Many to One



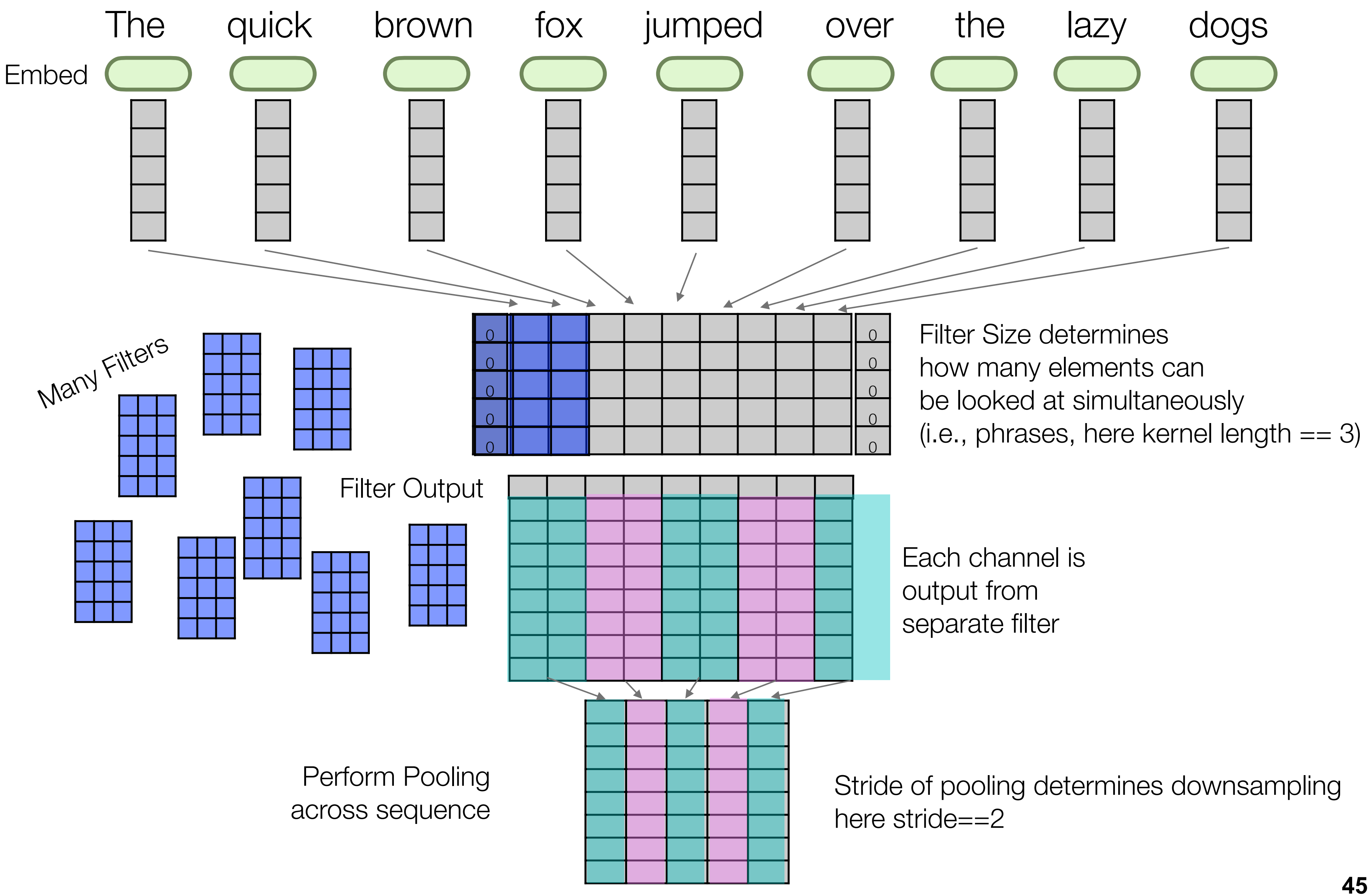
Visualization

GloVe produces word vectors with a marked banded structure that is evident upon visualization:



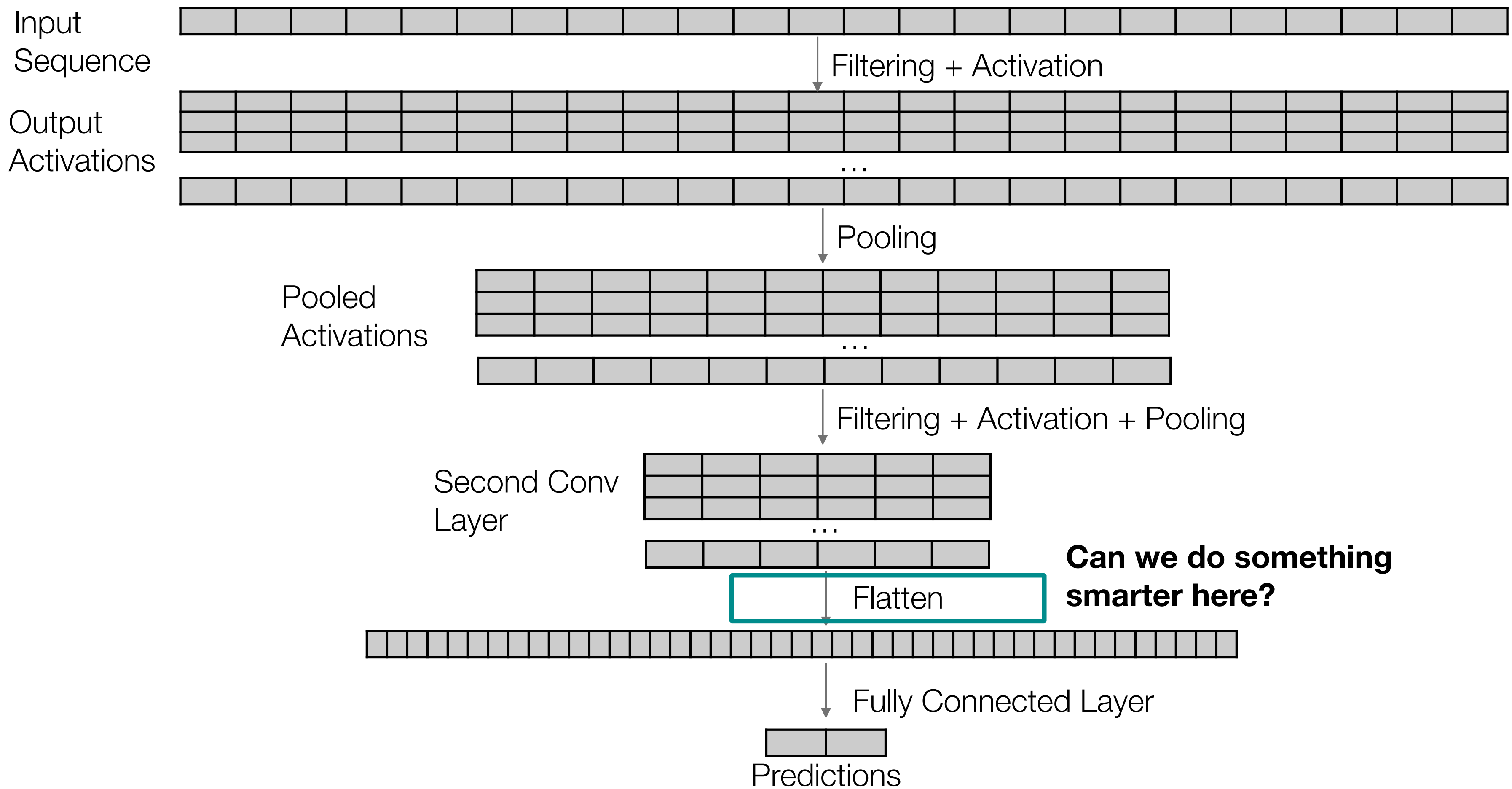
CNNs for Sequences

CNNs for Sequences



CNNs for Sequences

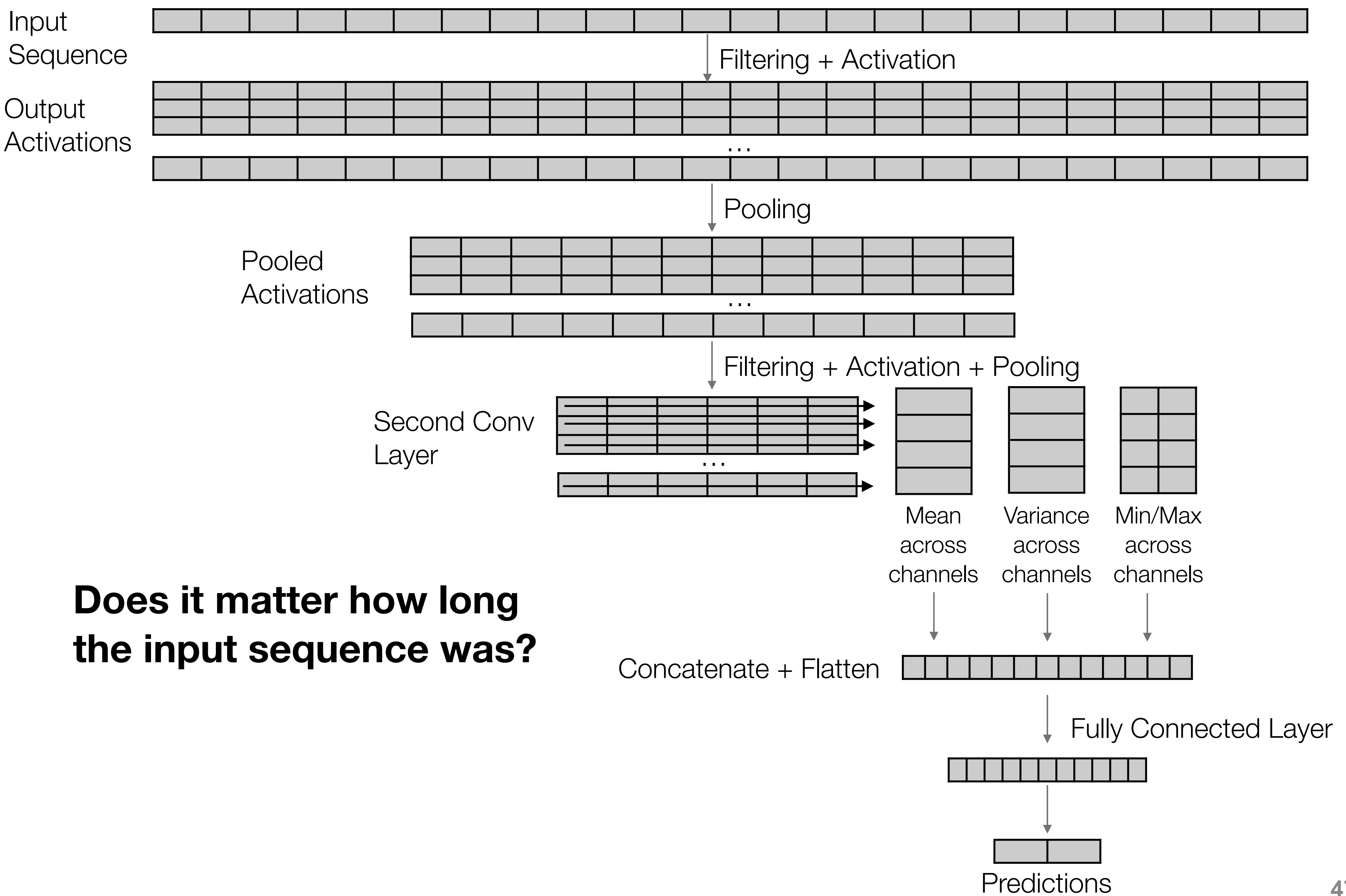
- RNNs are not inherently parallelized or efficient at remembering based on state vector, but CNNs can be run in parallel groups



Can we do something smarter here?

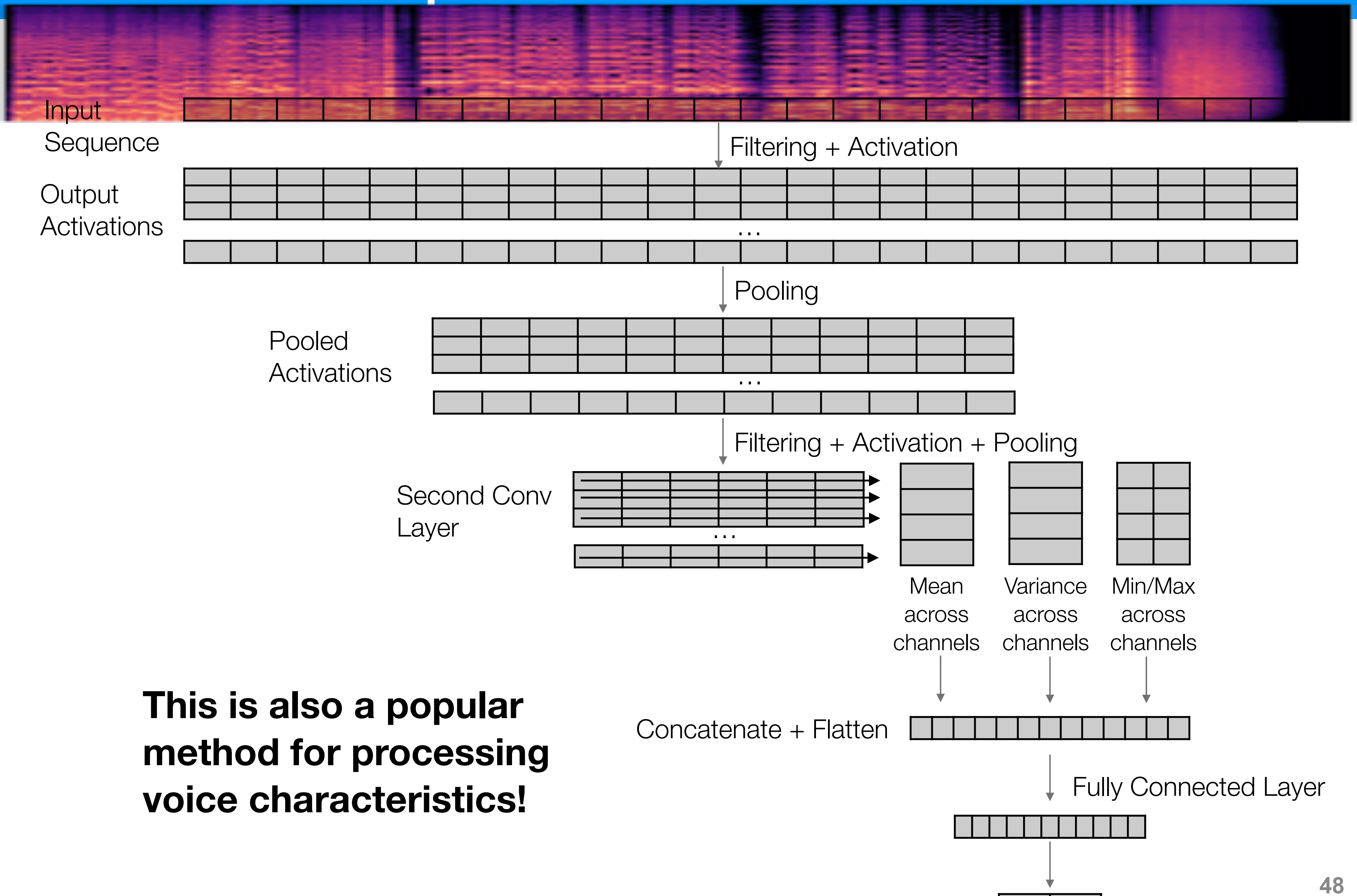
- Everything we learned in 2D CNNs can be applied to 1D CNNs…
- Residuals, separable convolution, squeezing, everything

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

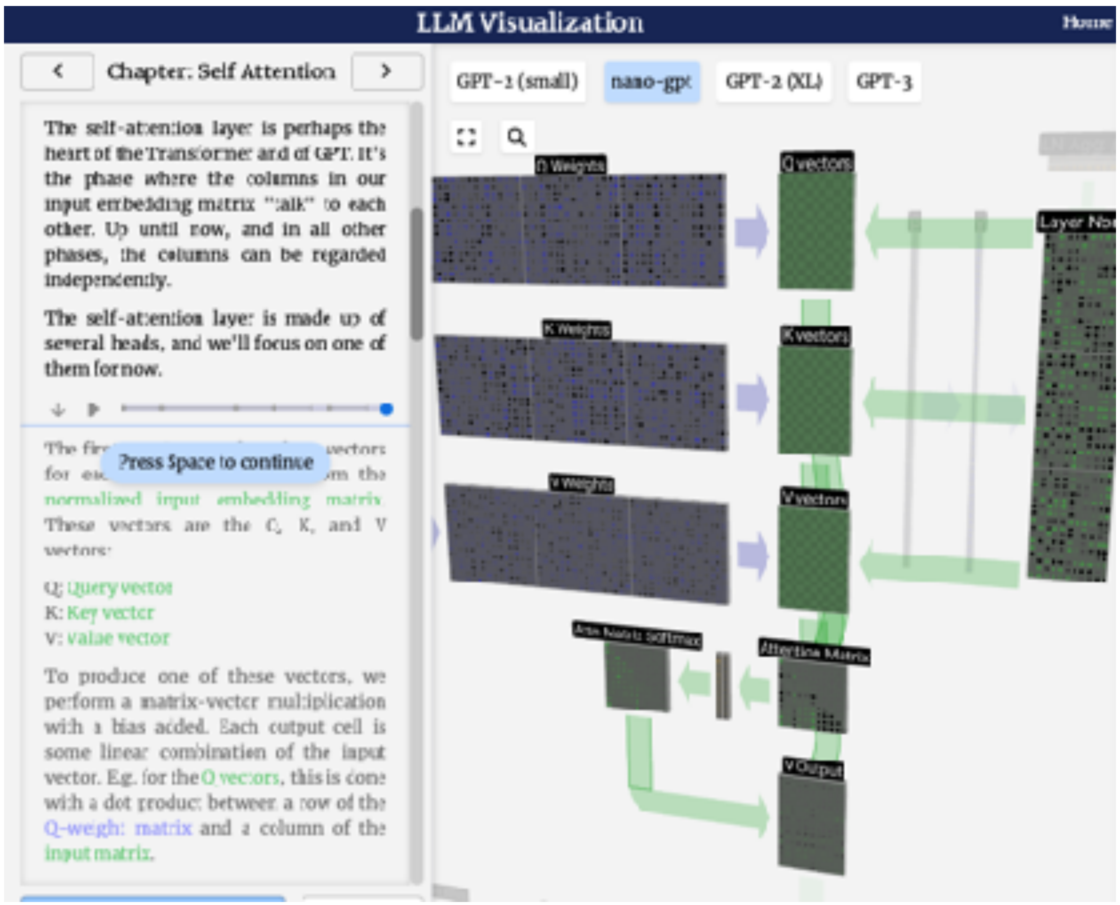
`13a. Sequence Basics [Experimental].ipynb`

Marked as “Experimental” because it does not yet have a textbook definition of transformers.

This link is perhaps the greatest tutorial on X-formers I have ever seen

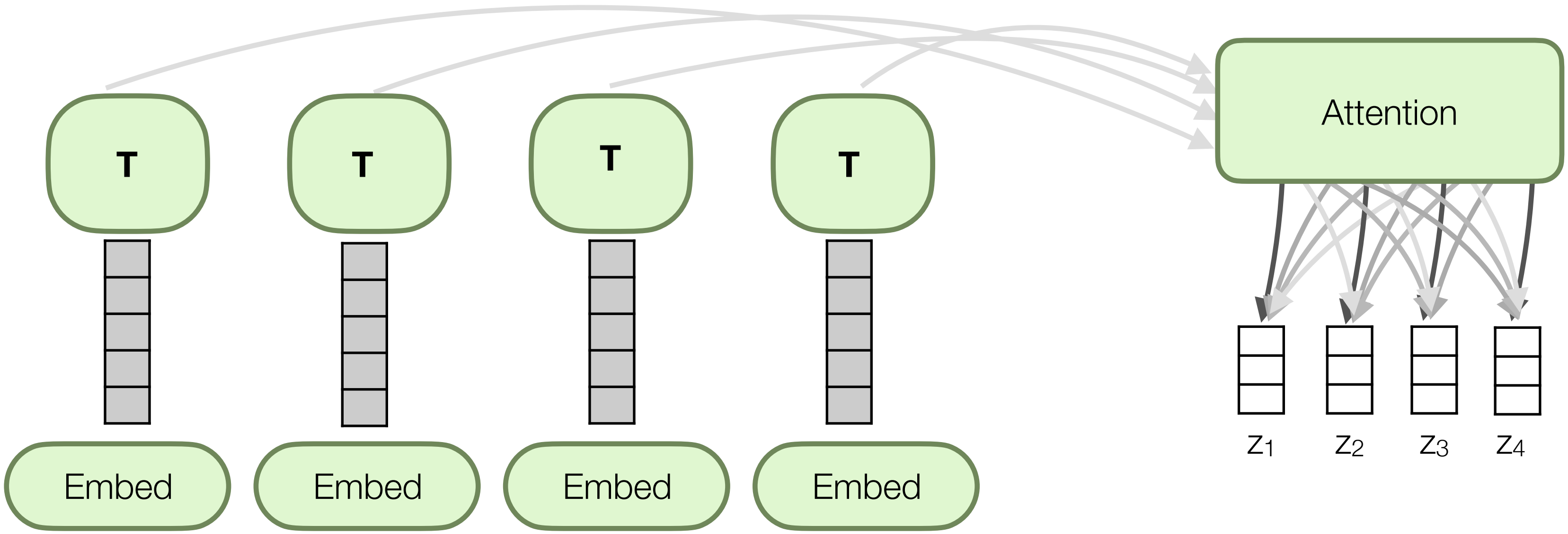
Transformers

https://bbycroft.net/llm



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?**

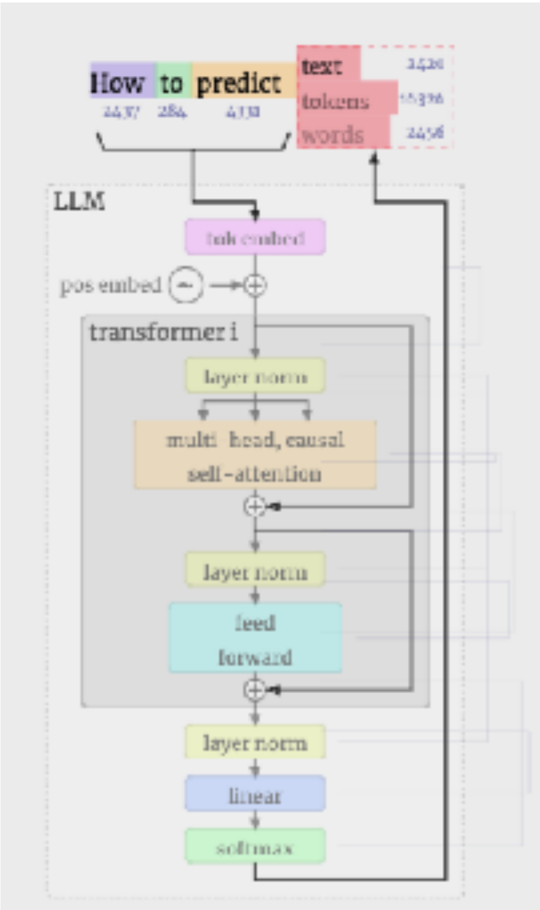


Attention is All You Need

- **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!!***

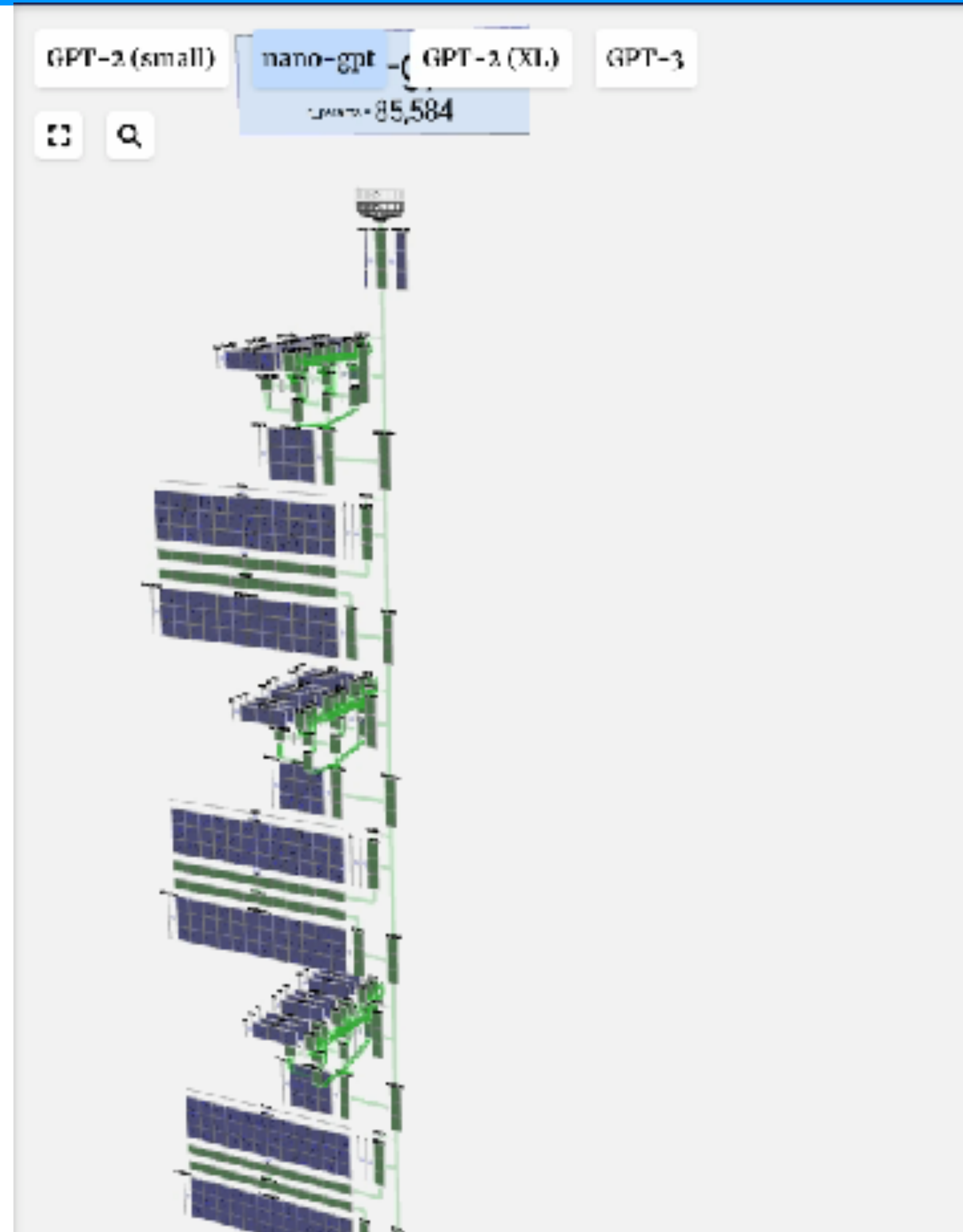
Transformer Overview

https://bbycroft.net/llm



Components

- Embedding
- Layer Norm
- Self Attention
- Projection
- MLP
- Transformer
- Softmax
- Output



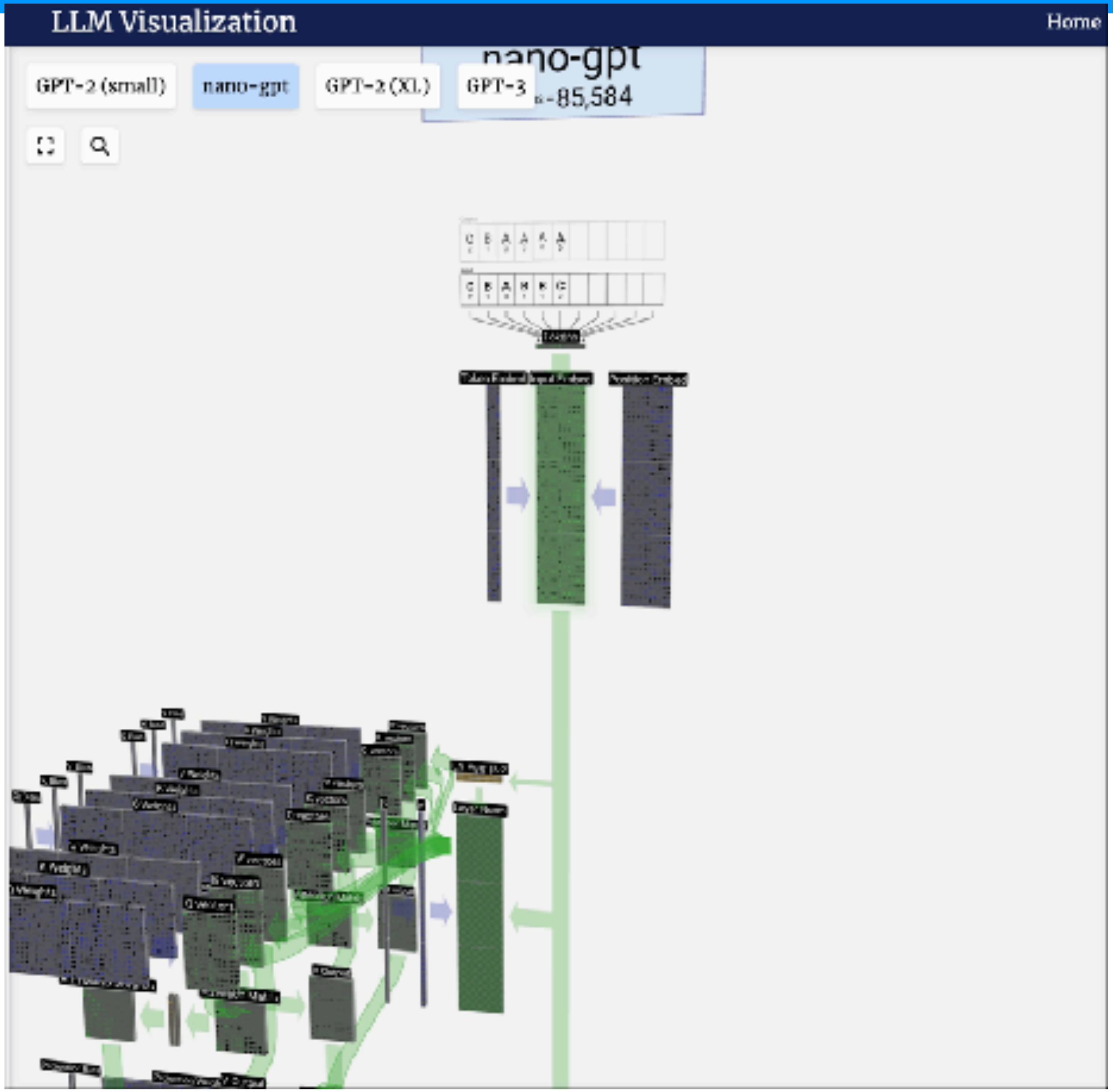
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O$$
$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

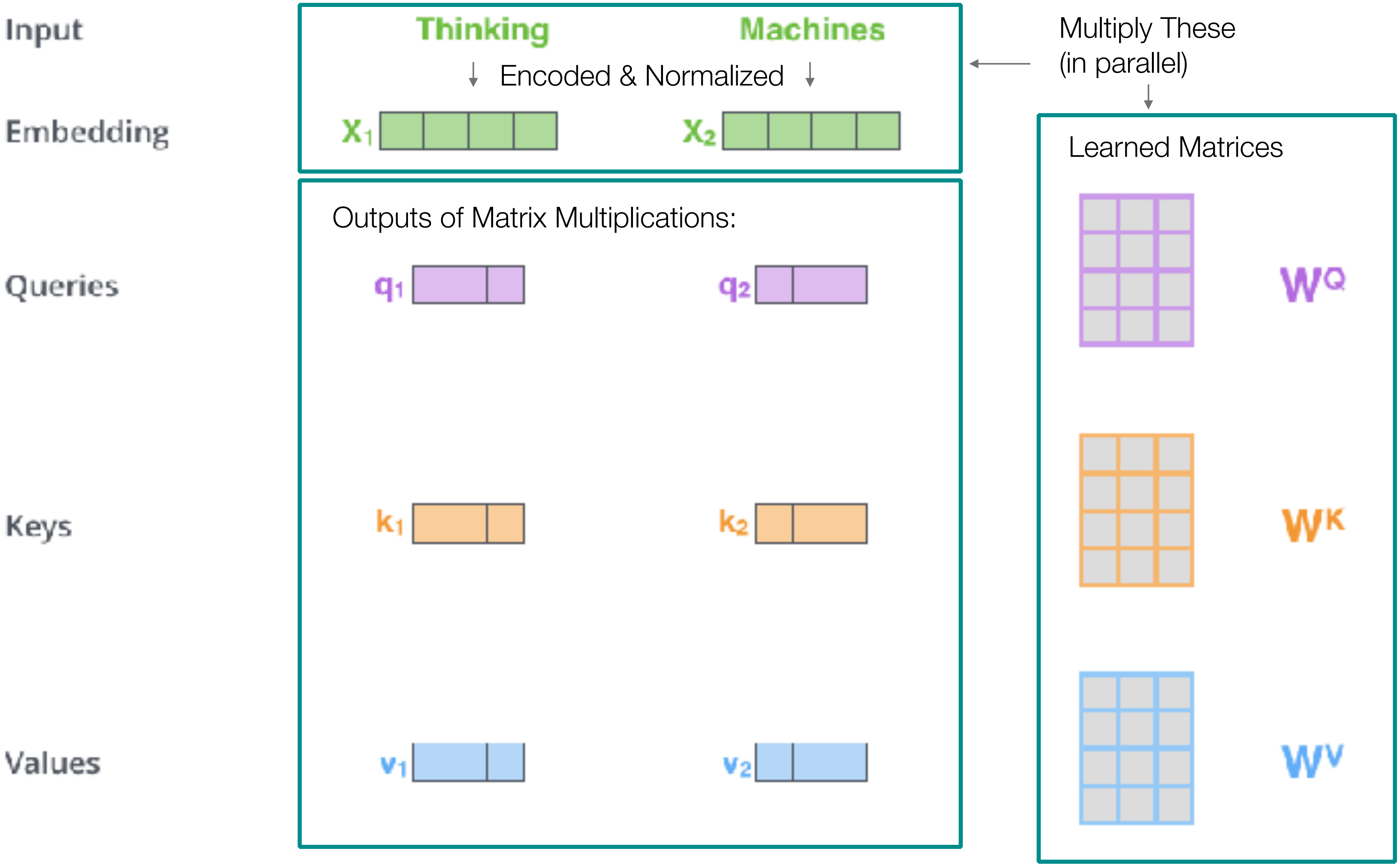
Layer Norm

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

Learn to normalize along a dimension of Z

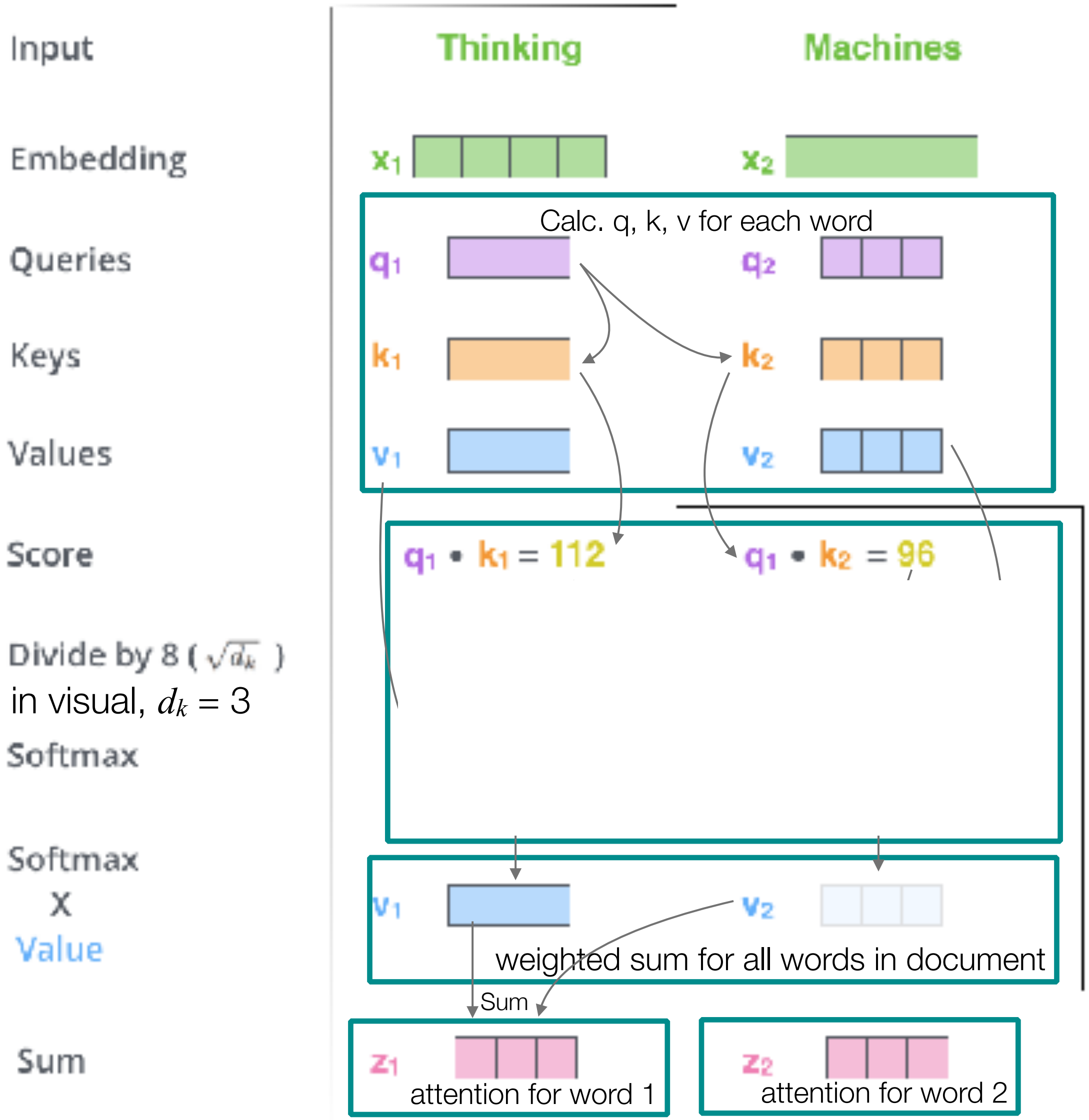


Transformer: in more detail

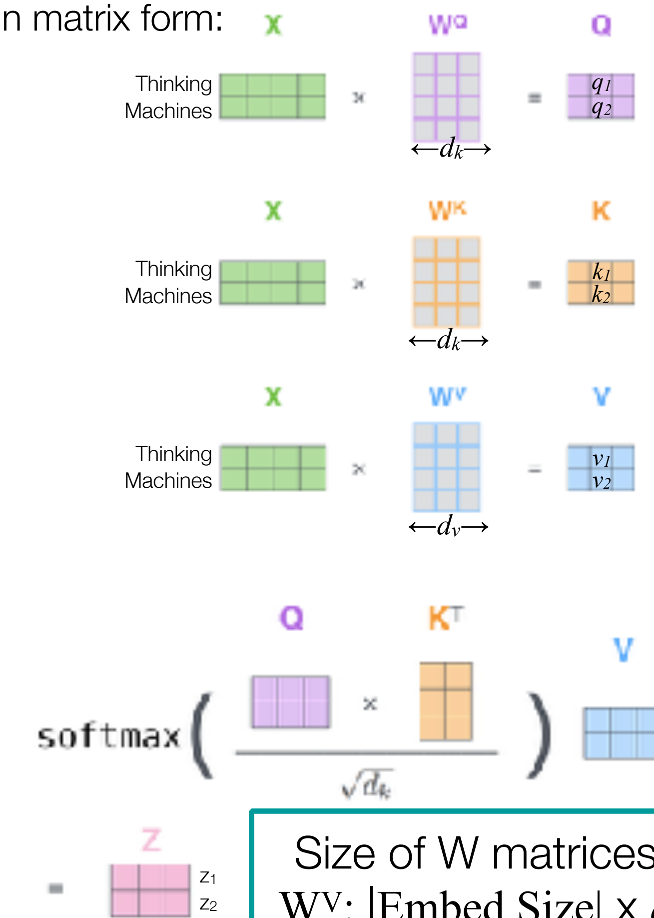


Excellent Blog on Transformers: http://jalammar.github.io/illustrated-transformer/

Transformer: in more detail



Straight forward to do this operation in matrix form:



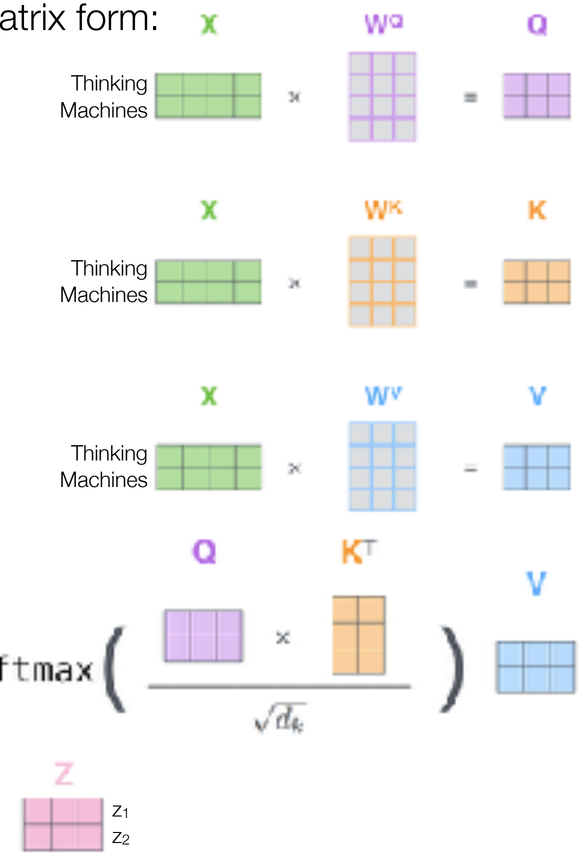
Size of W matrices:
W^V: |Embed Size| x d_v
$W^{Q,K}$: |Embed Size| x d_k

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

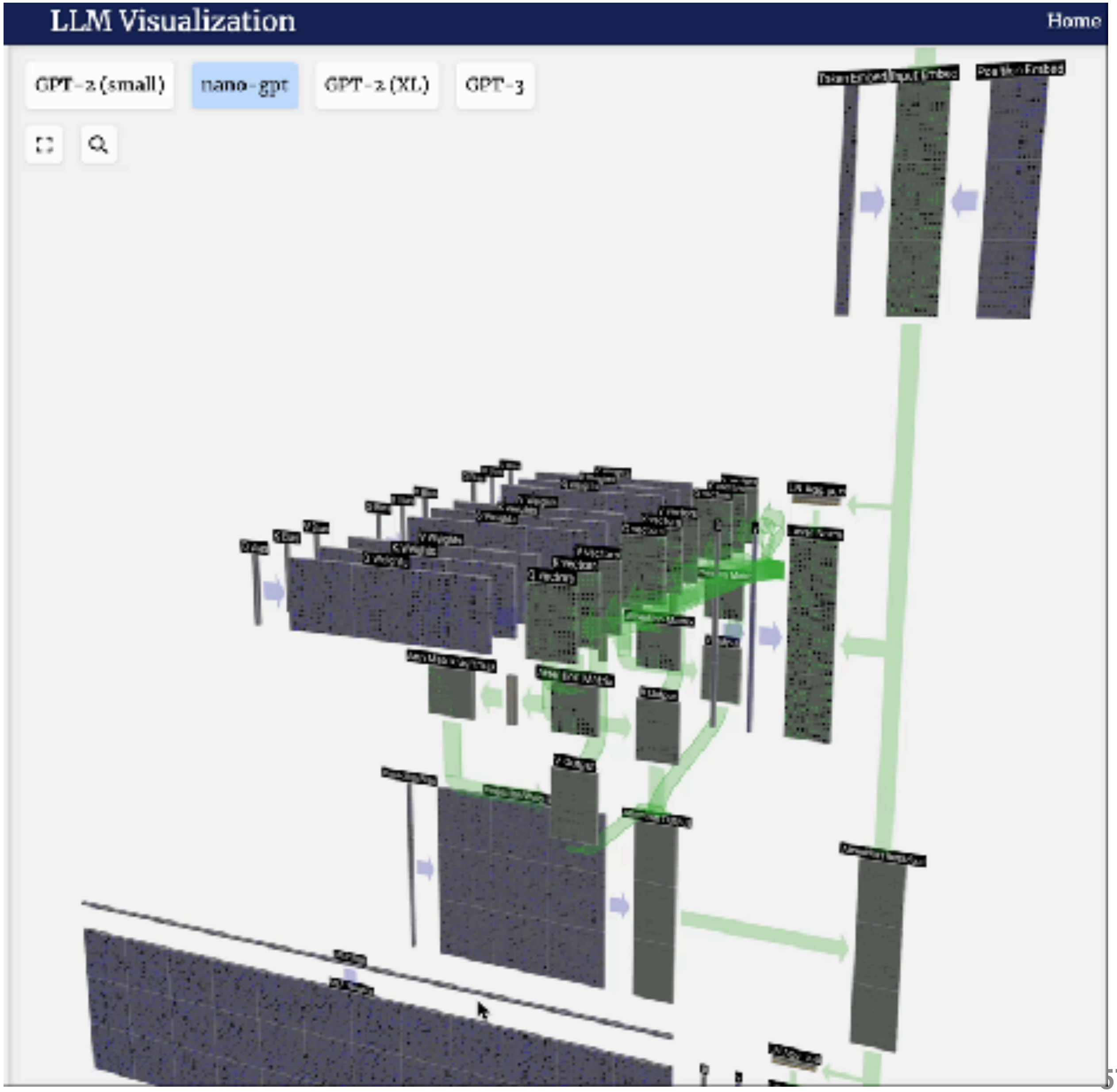
Excellent Blog on Transformers: http://jalammar.github.io/illustrated-transformer/

Self Attention: From https://bbycroft.net/llm

ight forward to do this operation
atrix form:



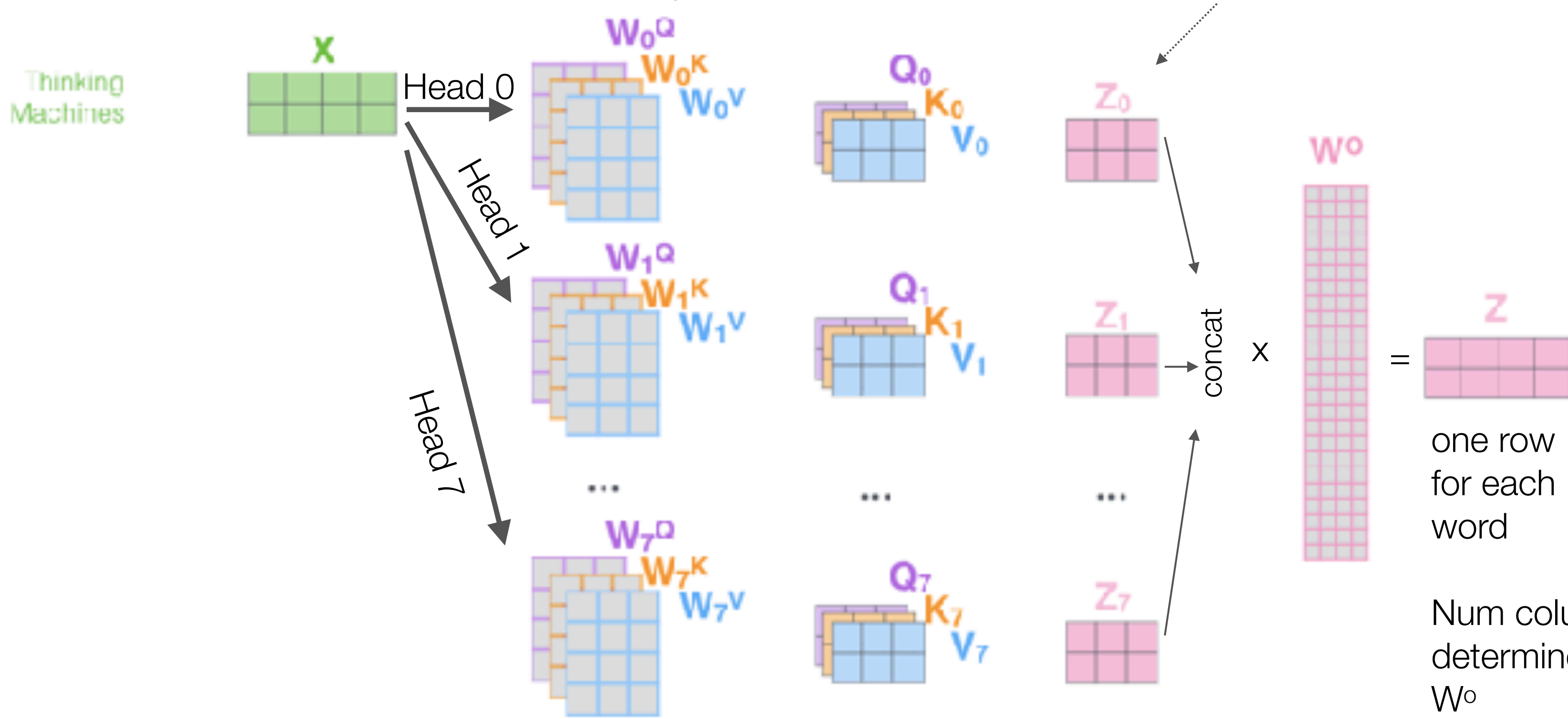
output of each head is Z_i



Transformer: Multi-headed Attention

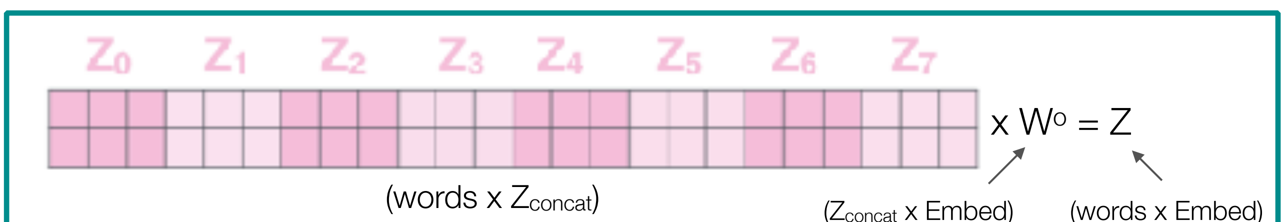
output of each head is Z_i

Z is transposed from video example



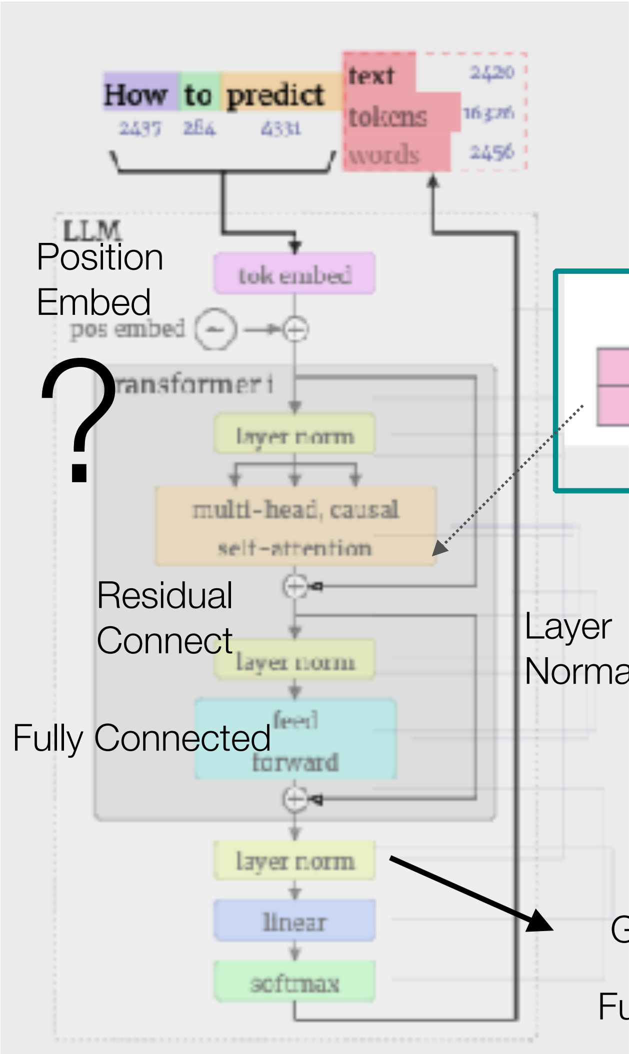
one row for each word

Num columns determined by W^O



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

Putting It Together



Position Embed

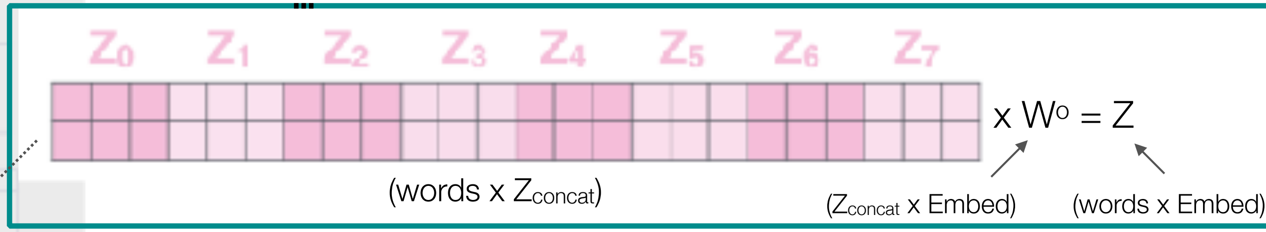
?

Residual Connect

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, dims of Wo)
    …
```

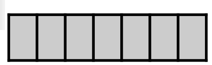


$$LN(Z_{row}) = \gamma \frac{Z_i - \mu_Z}{\sqrt{\sigma_Z^2 + \epsilon}} + \beta$$

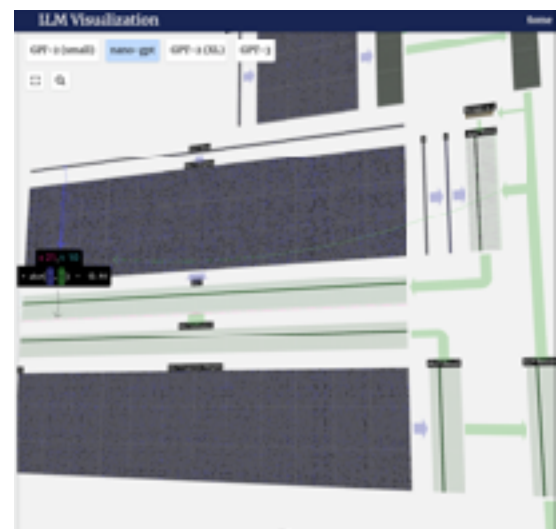
Learn to normalize the rows of Z

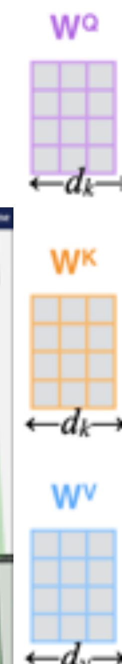
Global Average

Fully Connected



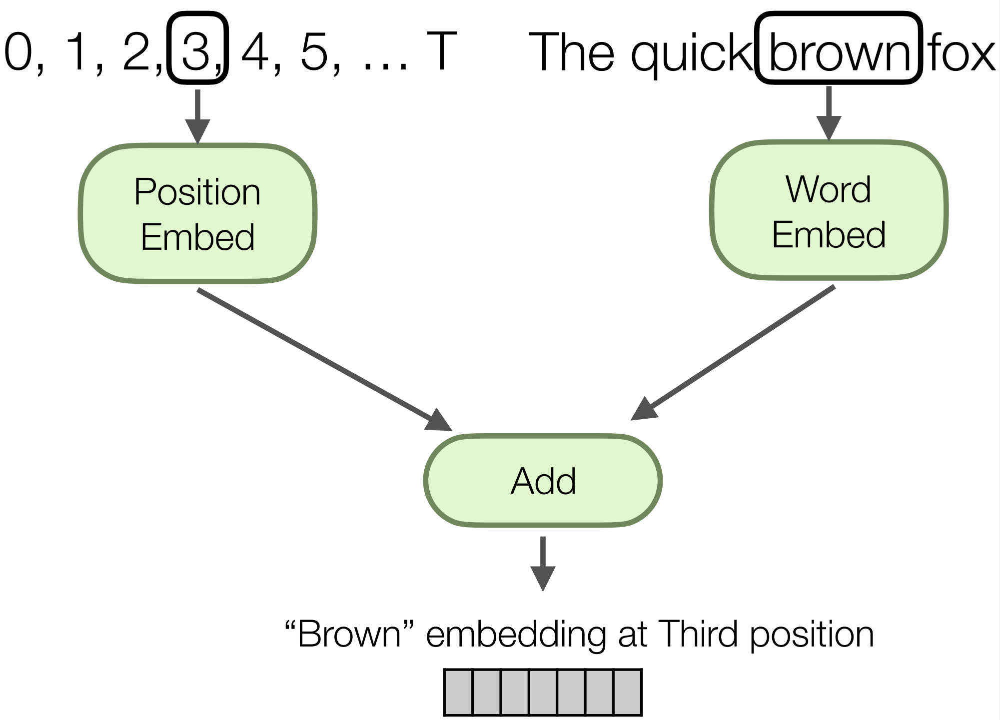
Prediction

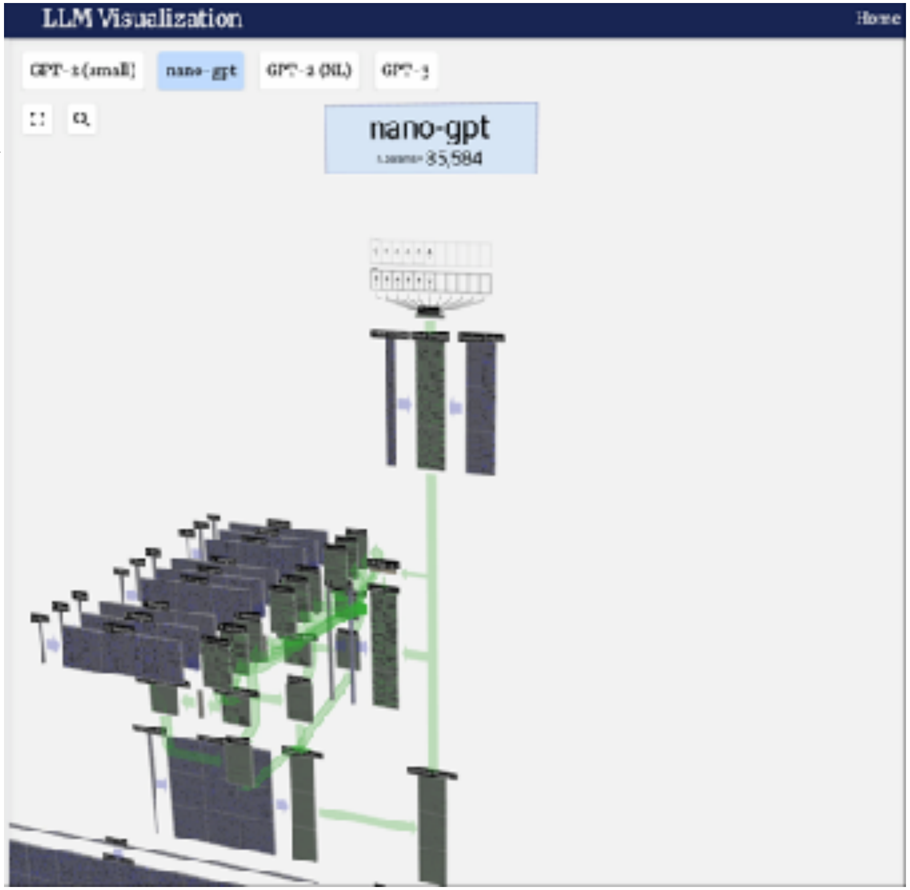




Transformer: Positional Encoding

- 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