Lecture Notes for
Machine Learning in Python

Professor Eric Larson

Transformer Demo + Positional Encoding

Lecture Agenda

- Logistics
  - Grading Update
  - Sequential Networks due **see canvas**
- Agenda
  - Sequential Networks Demo
    - Extended Demo
  - Final Town Hall
  - (if time) More effective position encoding
  - (if time) More efficient attention
  - Next time (if not behind):
    - Finish above lecture topics
    - Retrospective and Evaluations

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

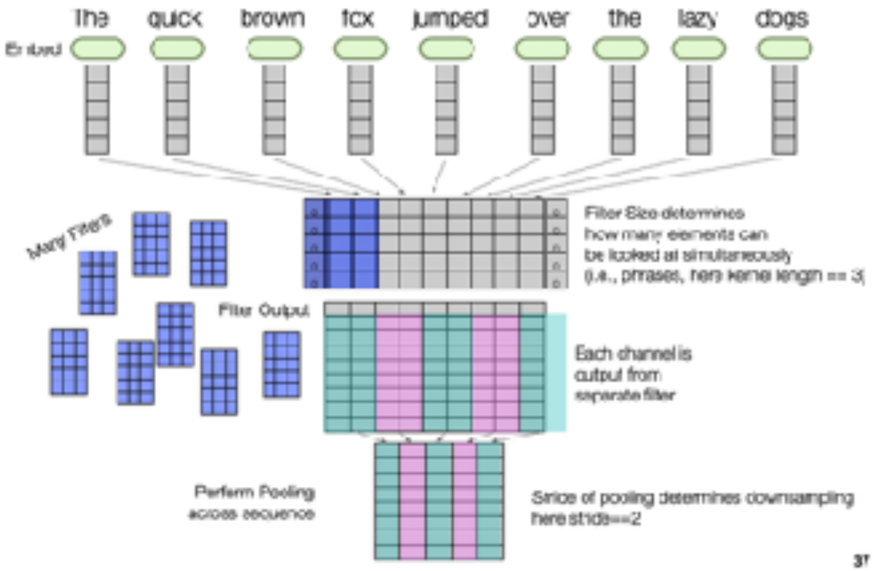
Linear and Logistic Regression
Numpy, Recreate API for Scikit-learn
Detailed mathematics for simple optimization
intuition for advanced optimization

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

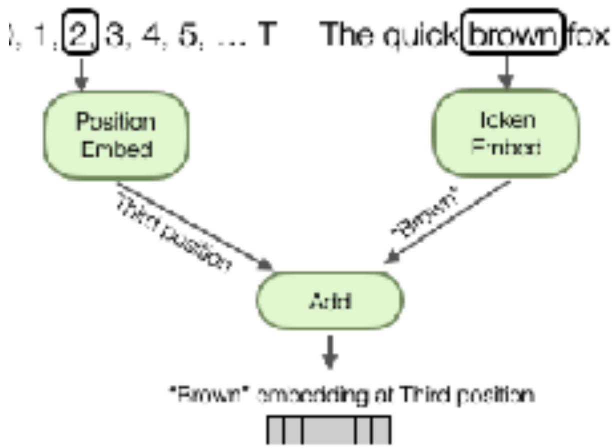
Wide and Deep Networks | **Convolutional Networks** | **Sequential Networks**
Keras, Tensorflow
Intuition, Detailed implement.

~~Ethics~~

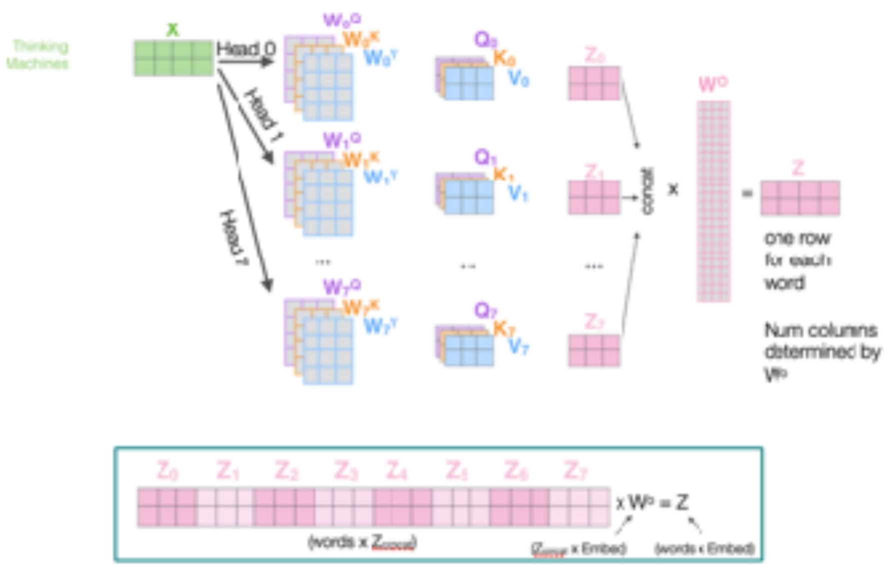
Last Time



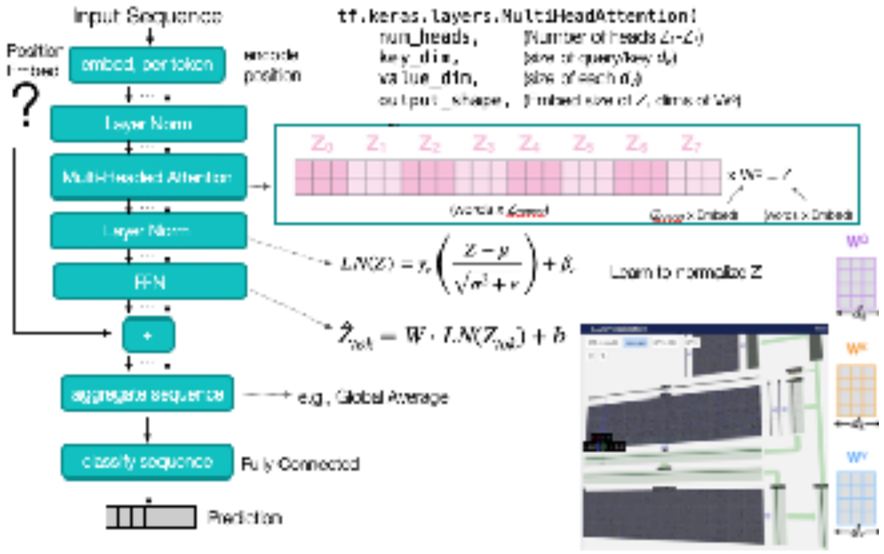
The
quick
brown
fox
jumped
over
the
lazy
dogs
Embed
Many Filters
Filter Size determines how many elements can be looked at simultaneously (i.e., phrases, here kernel length == 3)
Filter Output
Each channel is output from separate filter
Perform Pooling across sequence
Stride of pooling determines downsampling here stride==2



The quick brown fox
Position Embed
Token Embed
Third position
"Brown"
Add
"Brown" embedding at Third position

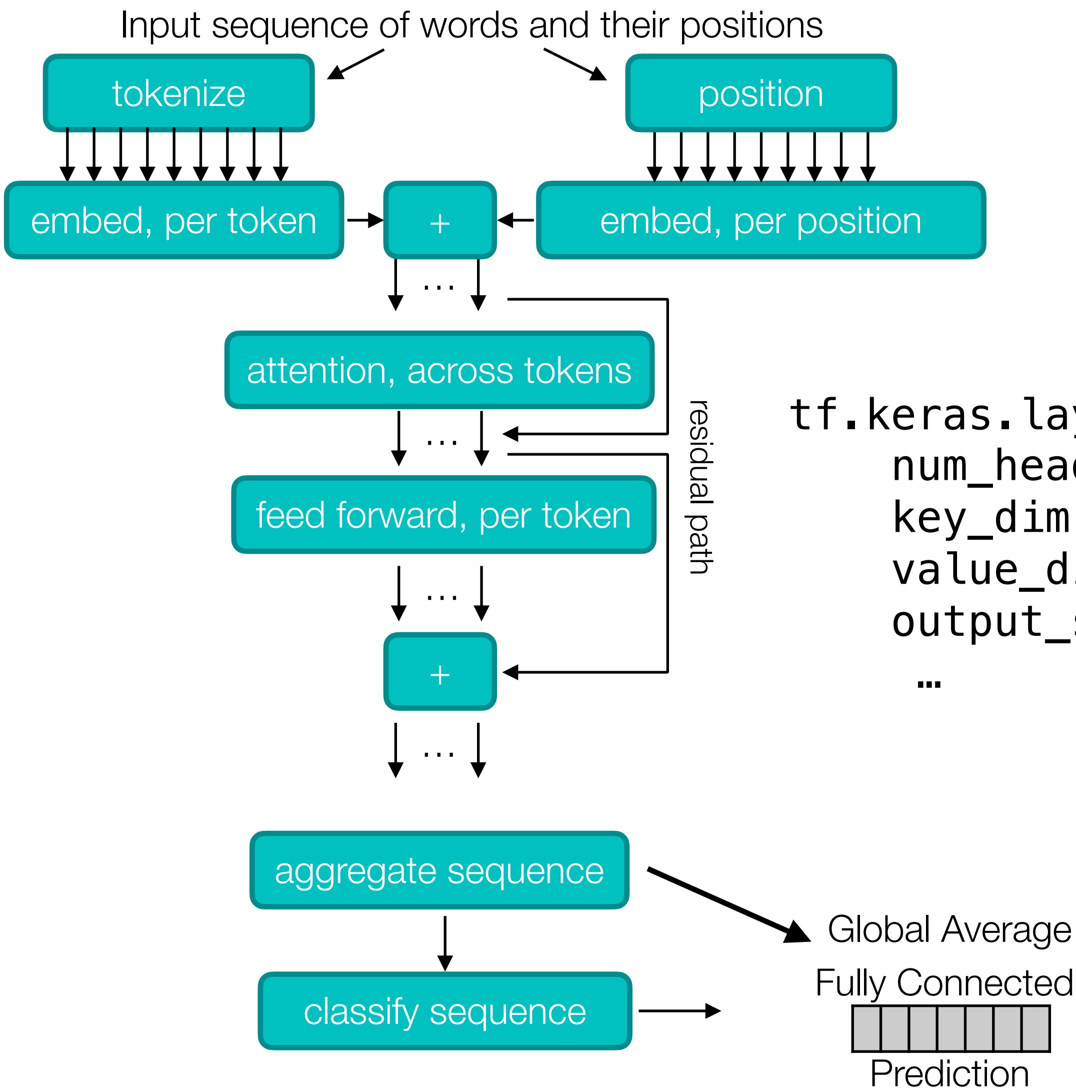


Thinking Machines
X
Head 0
Head 1
Head 7
concat
one row for each word
Num columns determined by W°
x W° = Z



Input Sequence
embed per token
encode position
Layer Norm
Multi-Headed Attention
FFN
aggregate sequence
e.g., Global Average
classify sequence
Fully Connected
Prediction
tf.keras.layers.MultiHeadAttention(
Learn to normalize Z

Demo




```
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)
     …
```

The Transformer and 20 news groups with GloVe

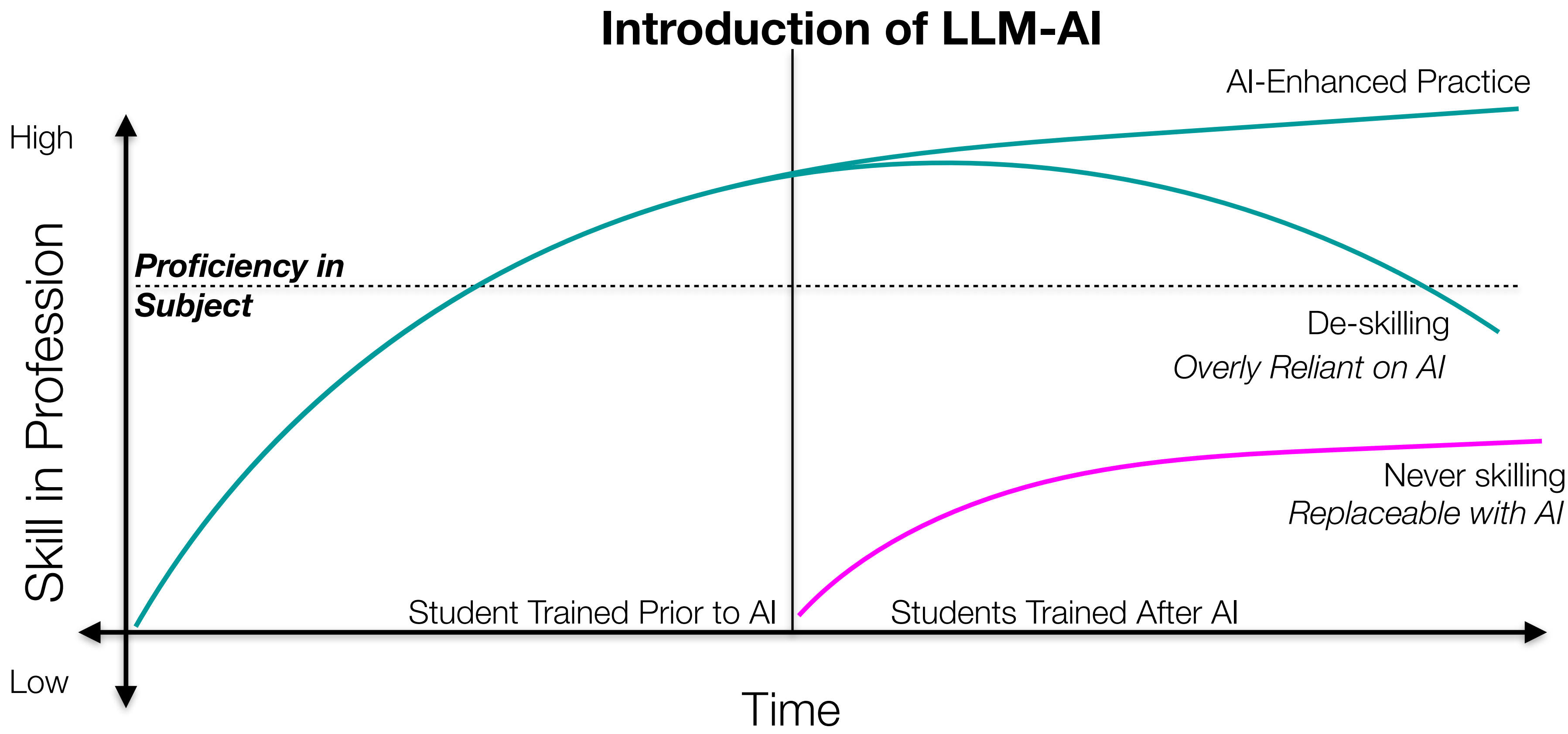
`13a. Sequence Basics [Experimental].ipynb`

Sequential Networks Town Hall

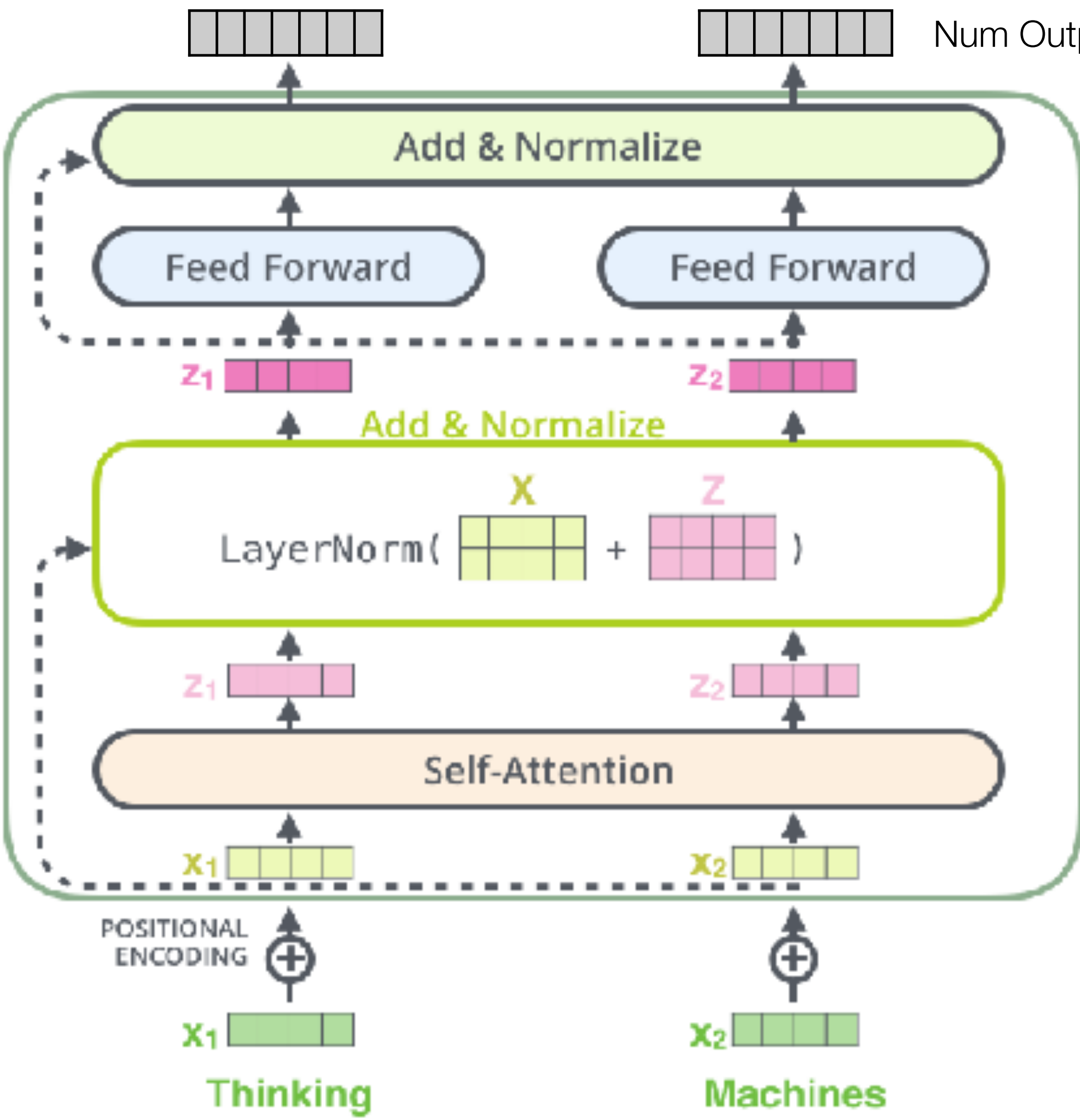
CNN, RNN, LSTM, GAN,
Test time data,
Early stopping,
Data augmentation,
Dropout, Batch norm,
Gradient clipping

Attention

Positional Encoding



Transformer for Sequence Classification



Num Outputs is Same as Inputs

Do these outputs change, if the input sequence changes order?

The order of vectors will change, but not the values of each vector…

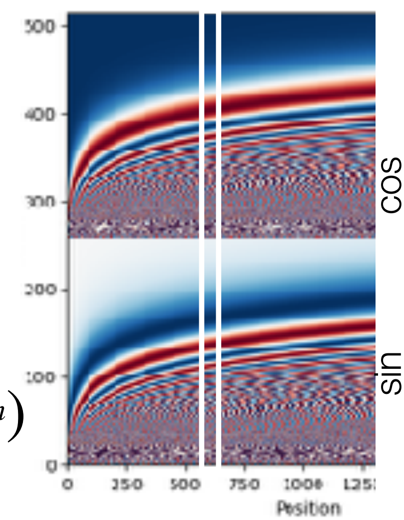
$$\text{softmax}\left(\frac{\mathbf{Q}\cdot\mathbf{K}^T}{\sqrt{d_k}}\right)\cdot\mathbf{V}$$

$\mathbf{V}_{(1:L)}$ $\mathbf{Q}_{(1:L)}$ $\mathbf{K}_{(1:L)}$

$\mathbf{X}_{(1:L)}$

Transformer: First Positional Encoding

- Objective: add notion of position to embedding
- Attempt in paper: add sin/cos to embedding

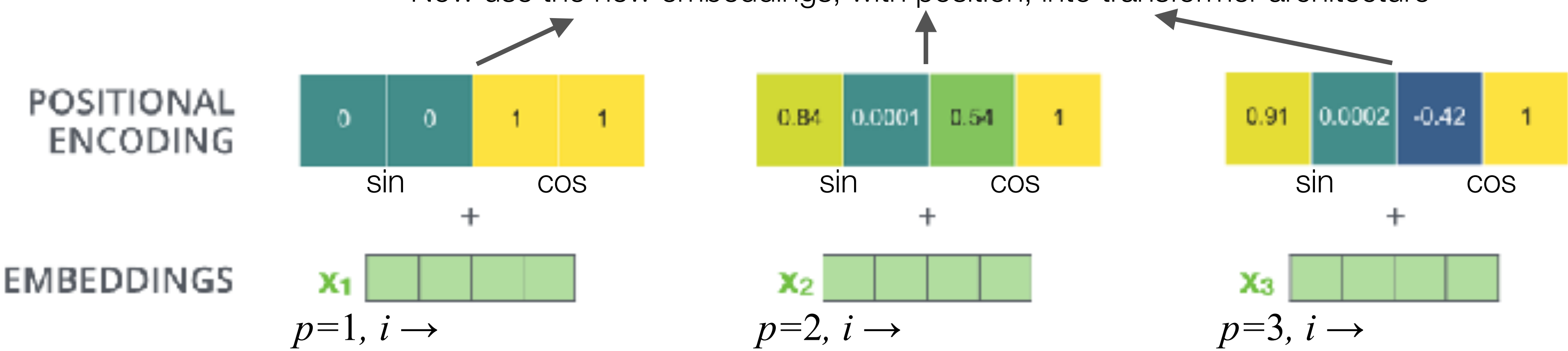


p: in sequence
d_m: 1/2 dim of embed
i = index in vector

$$PE_{(p,i\in 0\dots d_m-1)} = \sin(p/10000^{i/d_m})$$

$$PE_{(p,i\in d_m\dots 2d_m)} = \cos(p/10000^{(i-d_m)/d_m})$$

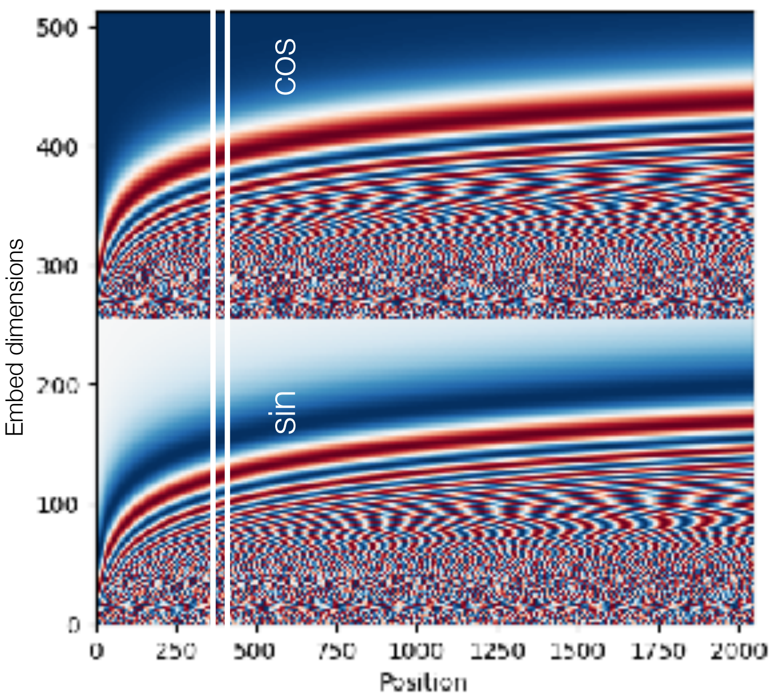
Now use the new embeddings, with position, into transformer architecture

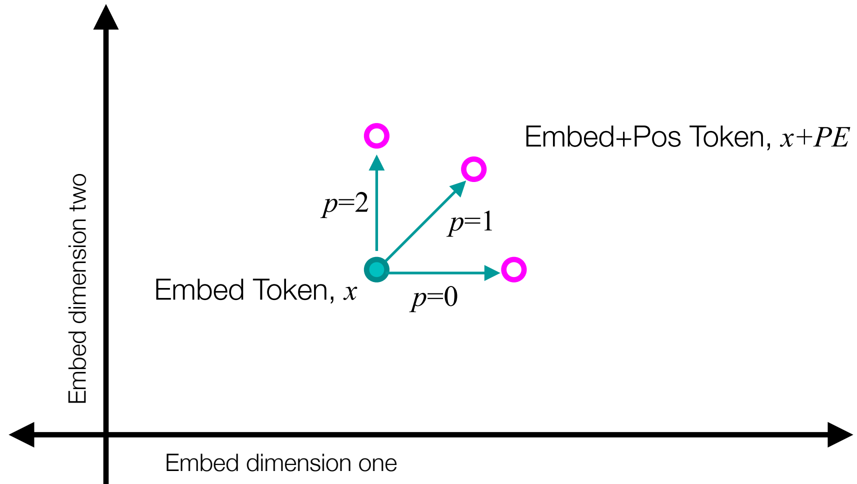


Hypothesis: Now the word proximity is encoded in the embedding matrix, with other pertinent information. Well, it does help… so it could be true that this is a good way to do it.

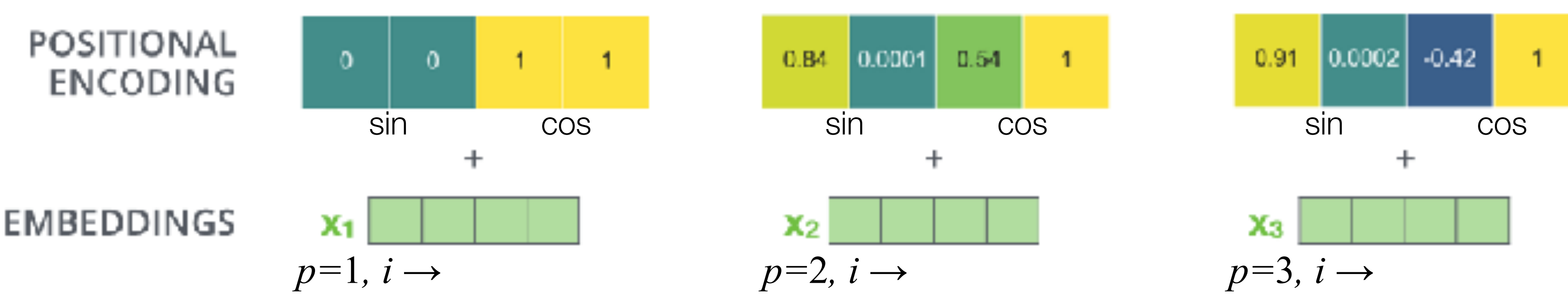
Excellent Blog on Transformers: http://jalammar.github.io/illustrated-transformer/

Positional Intuition, Geometrically



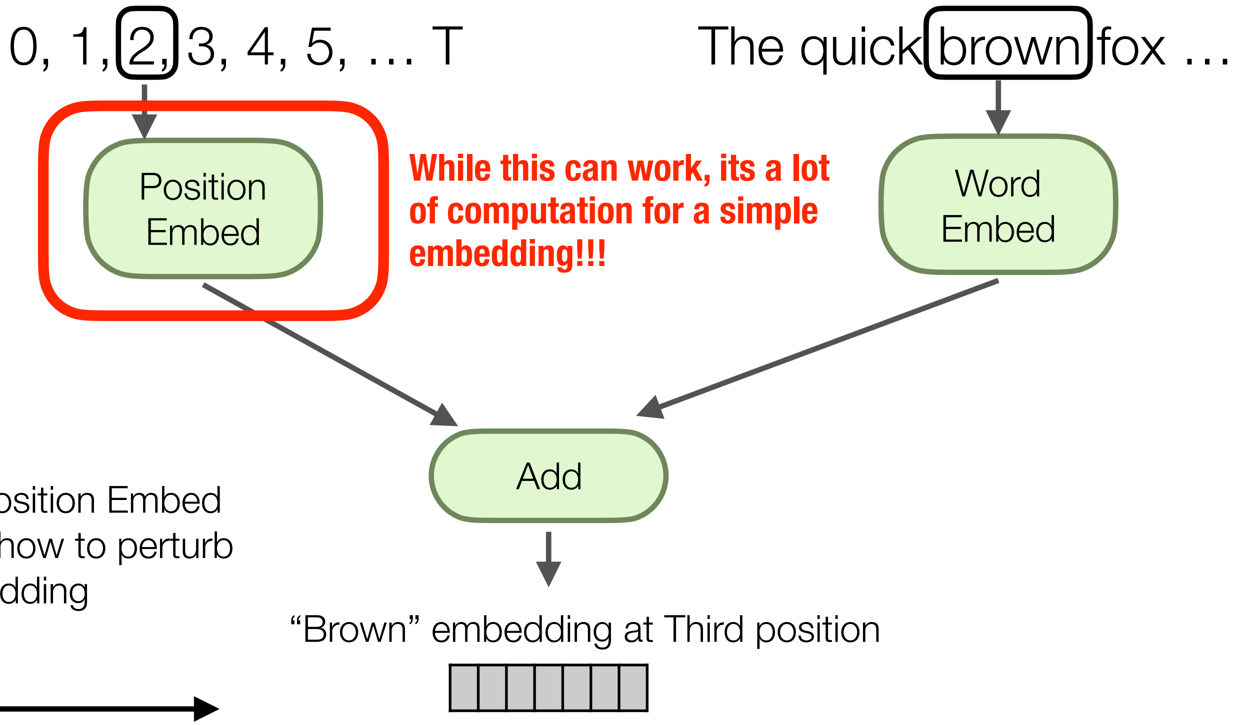


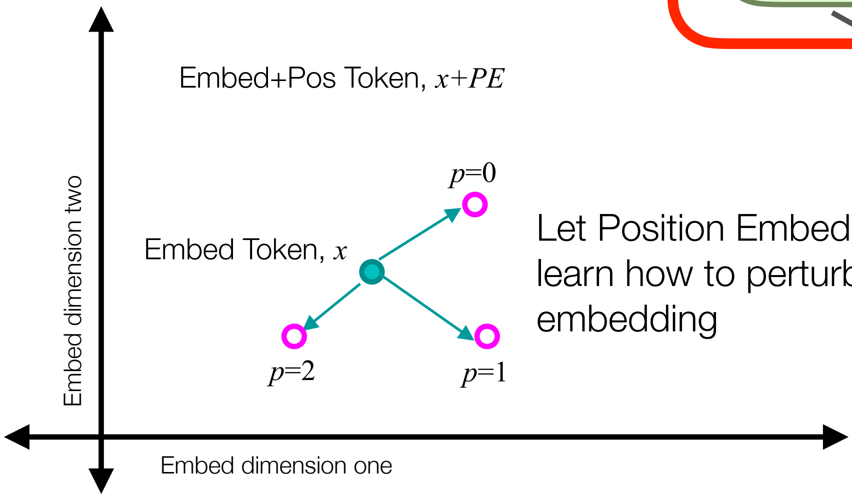
Assume that transformer model can learn the small perturbations in positional encoding, and how to use them in the context of the word.



Transformer: Positional Embedding

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



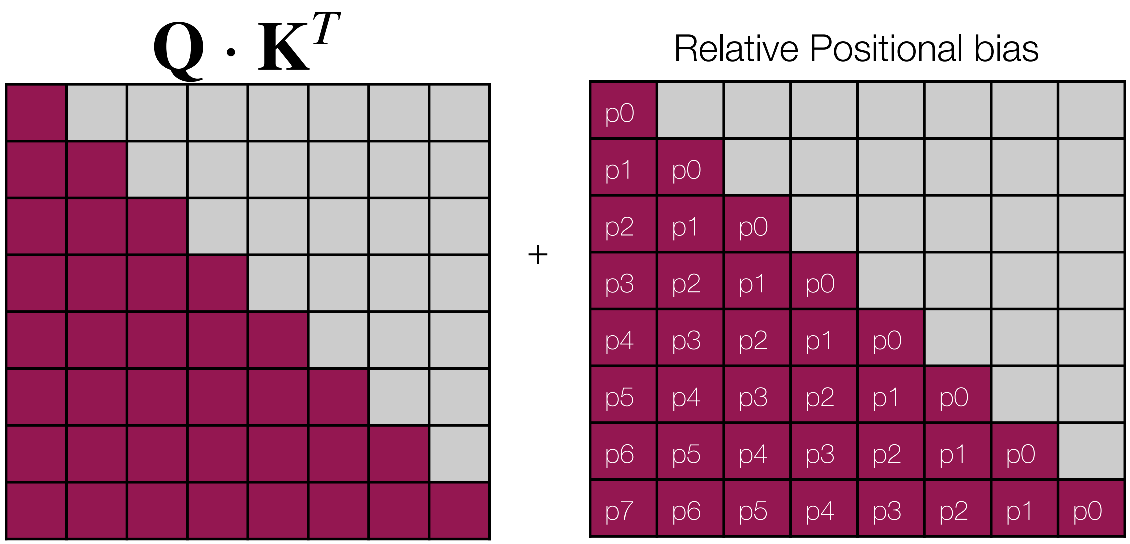


Let Position Embed learn how to perturb embedding

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

Relative Positional Encoding

- Relative position encoding:
  add relative words differences into $\mathbf{Q} \cdot \mathbf{K}^T$



- (+) nicely structured position information
- (-) Slow, lots more memory
- (-) fragments ops further, more KV cache misses

- **How might we still encode relative position, without all the overhead?**