

Lecture Notes for **Machine Learning in Python**

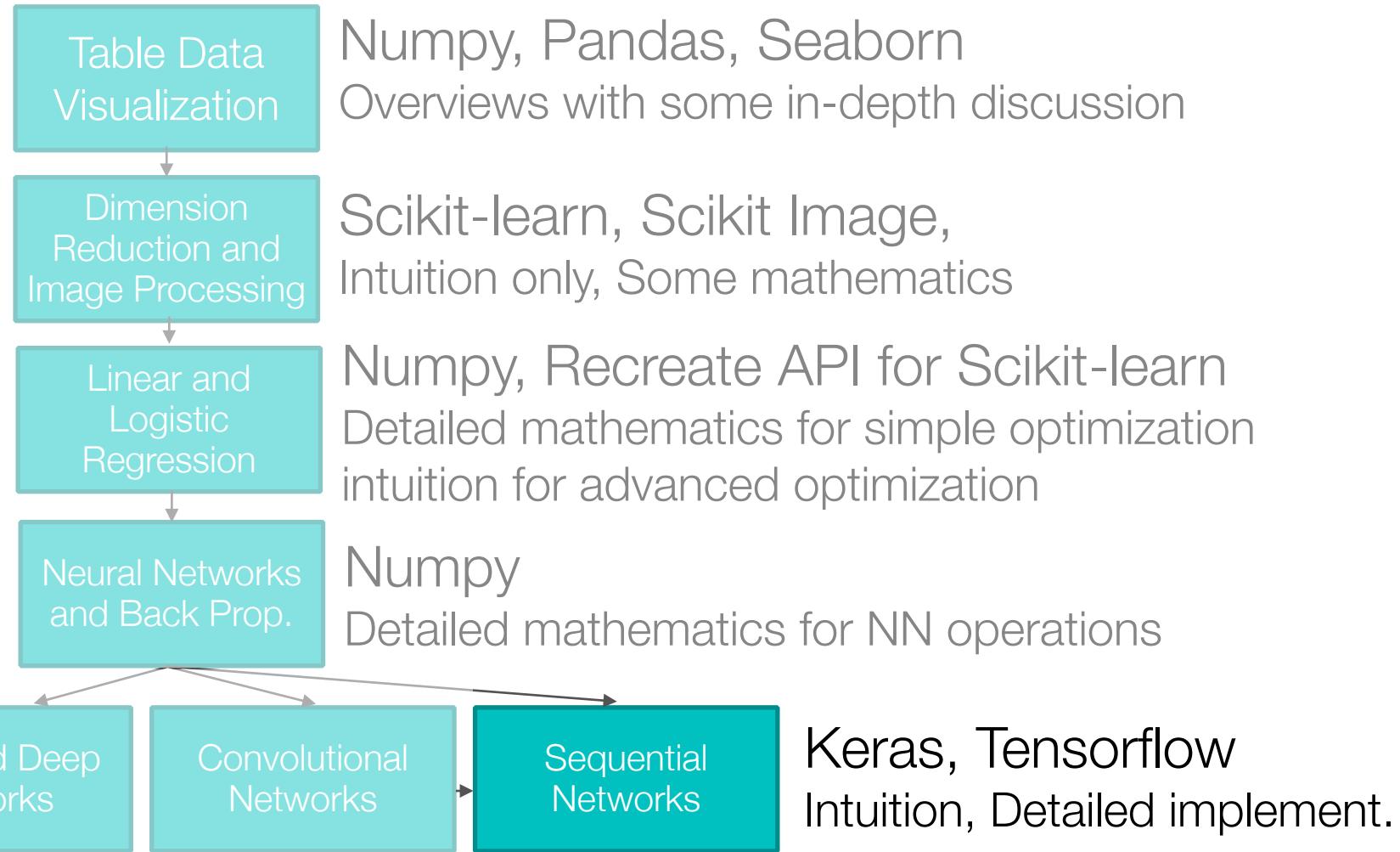
Professor Eric Larson

Final Lecture: Position, Attention, Retrospective

Lecture Agenda

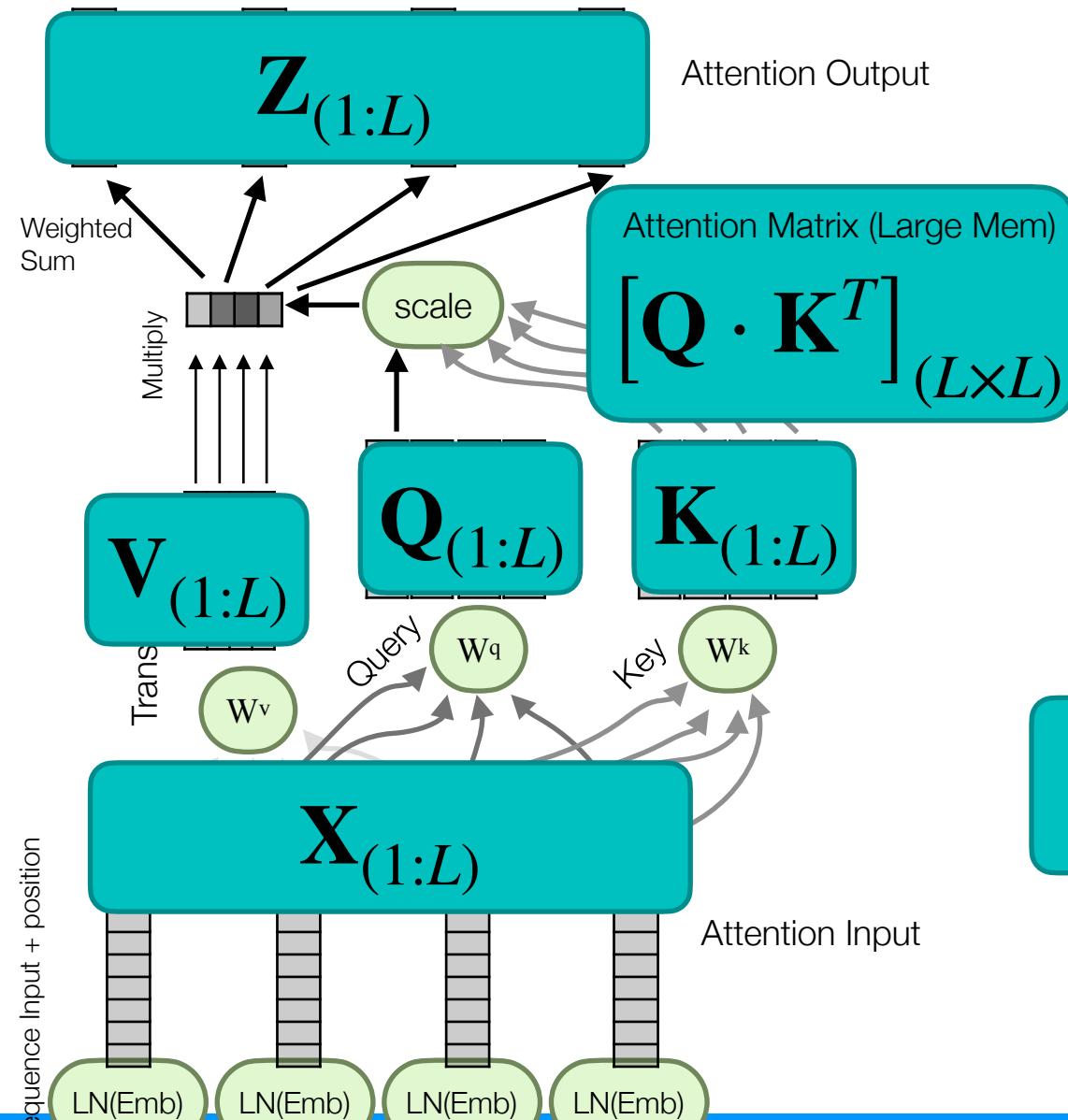
- Logistics
 - Grading Update
 - Sequential Networks due **Last Day of Finals**
 - **Before Midnight on December 14**
 - I will have grades finalized by
late December 15
- Agenda
 - More efficient Transformers
 - Retrospective and Evaluations

Class Overview, by topic



Ethics X

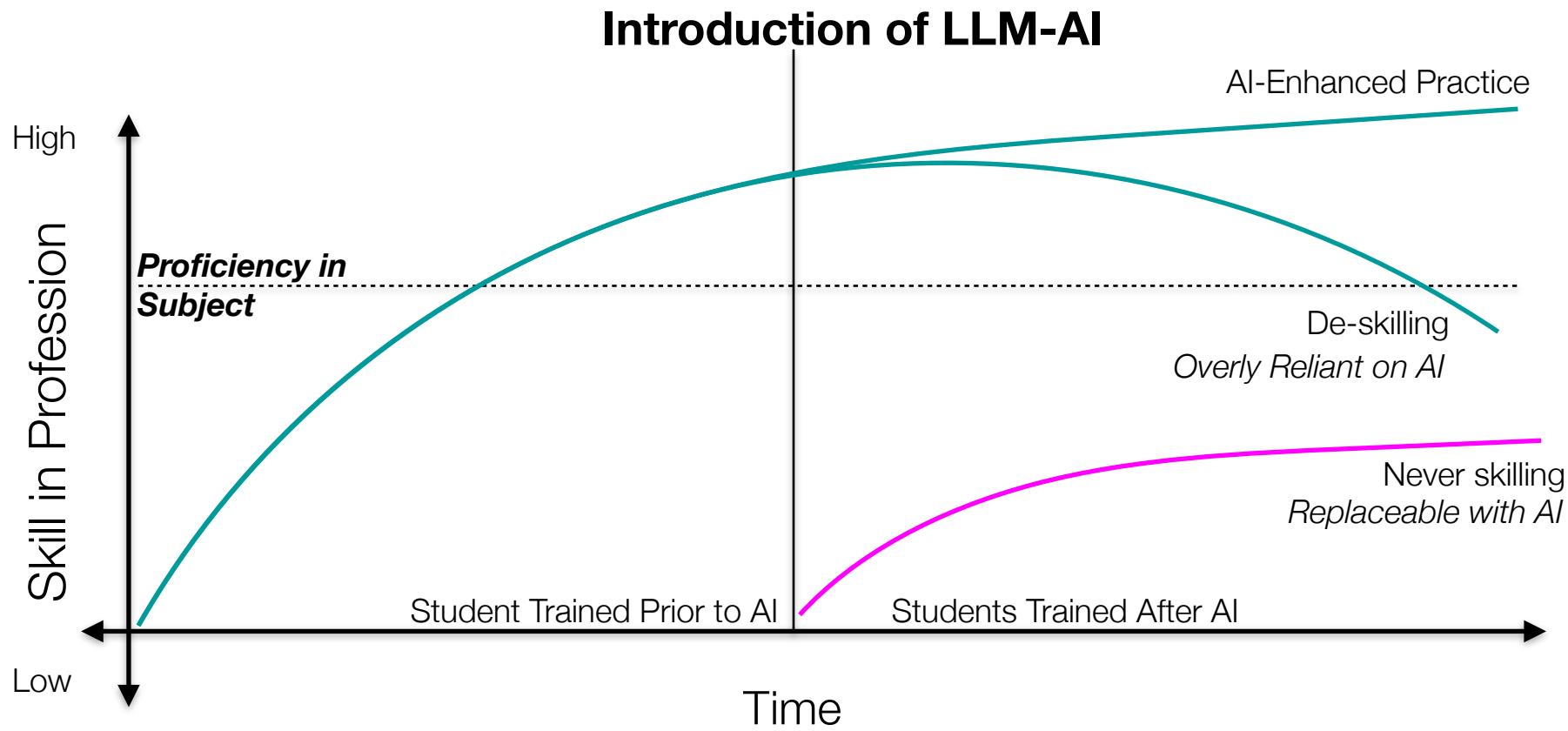
Self Attention Overview (Review)



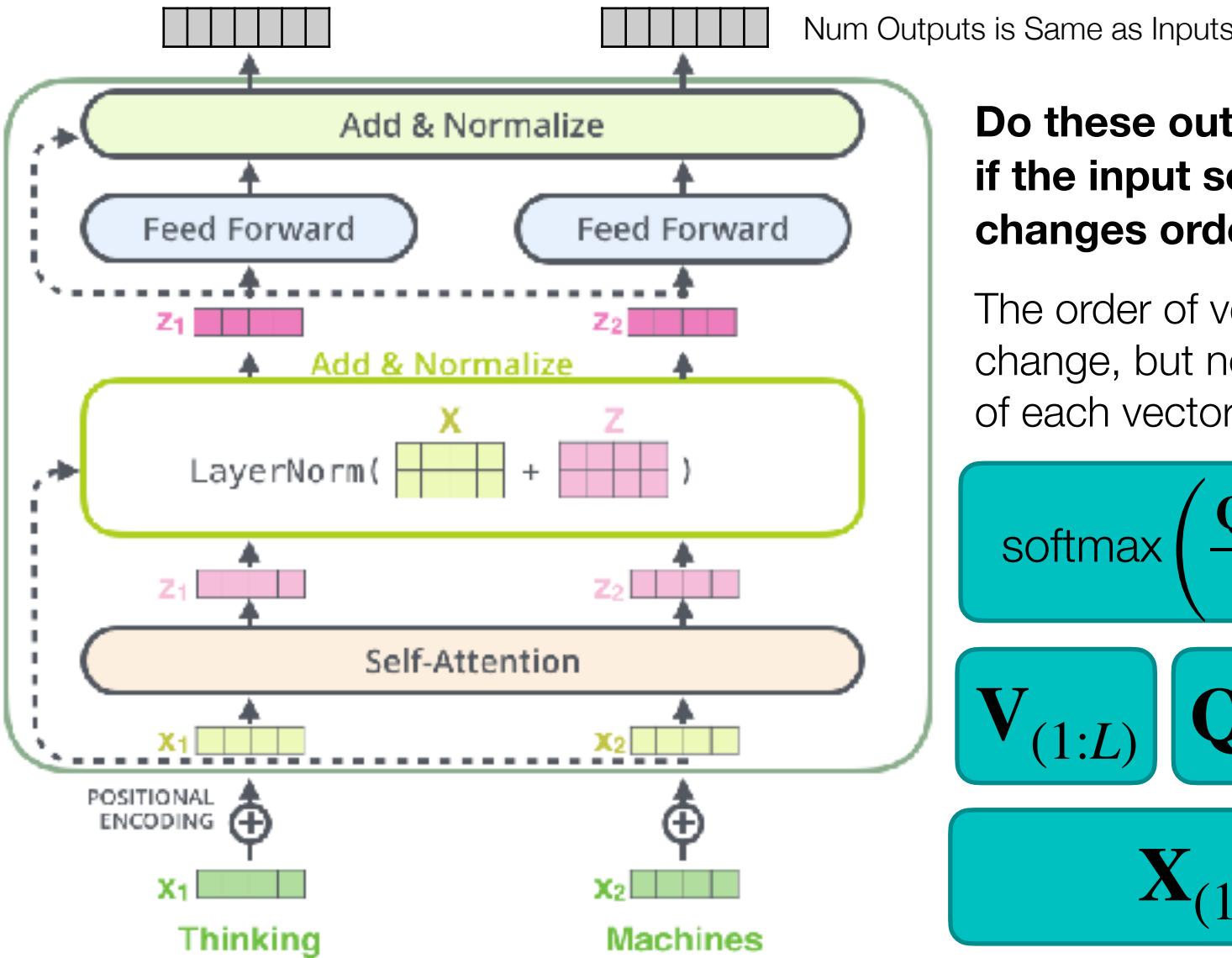
- Trained: $\mathbf{W}^v, \mathbf{W}^q, \mathbf{W}^k$
- Other Parameters:
 - L : length of sequence
 - Query/Key dimension, d_k
 - Value dimension, d_v
 - Type of positional encoding (more later)

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

Positional Encoding



Transformer for Sequence Classification



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

$$\hat{\mathbf{X}} = \underset{\text{to x-former}}{\mathbf{X}} + \underset{\text{word embed}}{\mathbf{PE}} + \underset{\text{position embed}}{\mathbf{PE}}$$

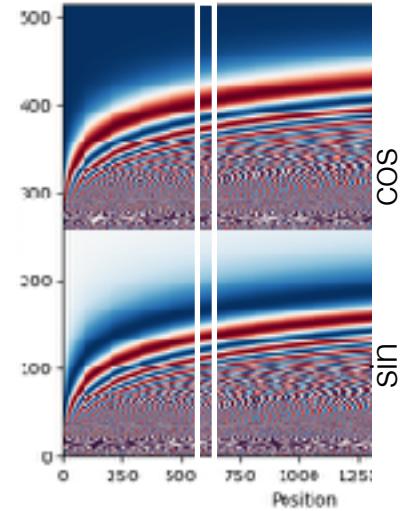
p : in sequence

D : dim of embed

i = index in vector

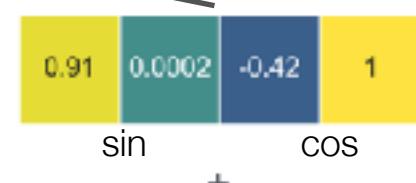
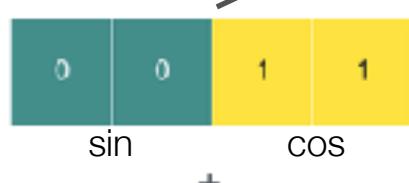
$$PE_{(p,i \in 0 \dots D/2-1)} = \sin(p/10000^{i/(D/2)})$$

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

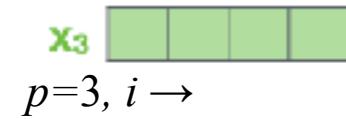
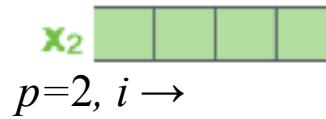
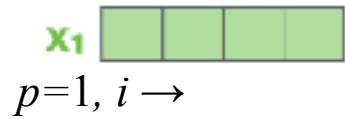


Now use the new embeddings, with position, into transformer architecture

POSITIONAL
ENCODING



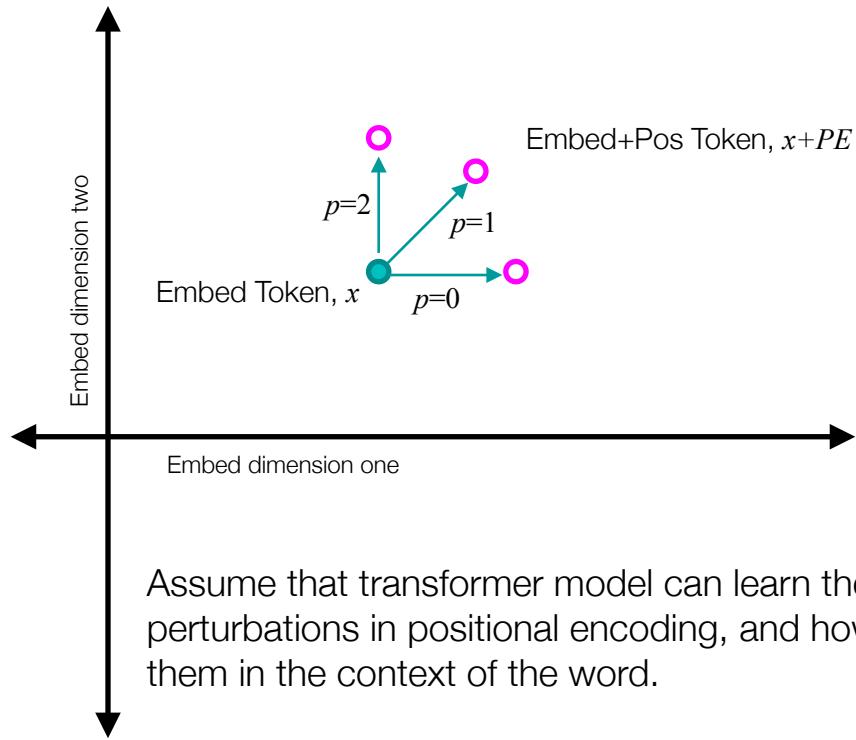
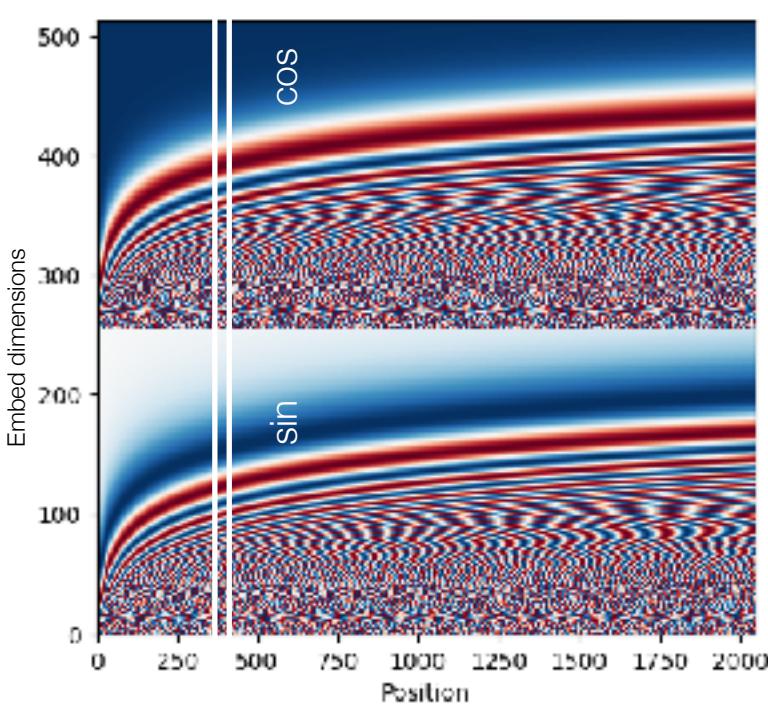
EMBEDDINGS



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.

**POSITIONAL
ENCODING**

0	0	1	1
sin		cos	

EMBEDDINGS

$$x_1 \quad p=1, i \rightarrow$$

0.84	0.0001	0.54	1
sin		cos	

$$x_2 \quad p=2, i \rightarrow$$

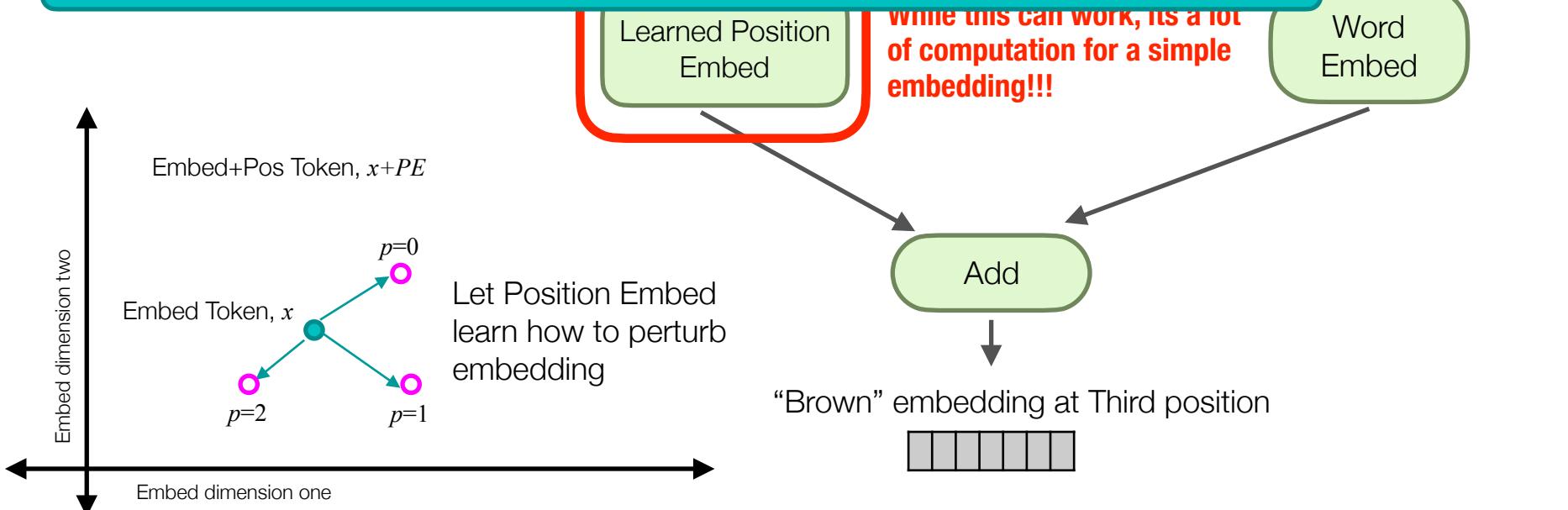
0.91	0.0002	-0.42	1
sin		cos	

$$x_3 \quad p=3, i \rightarrow$$

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:**

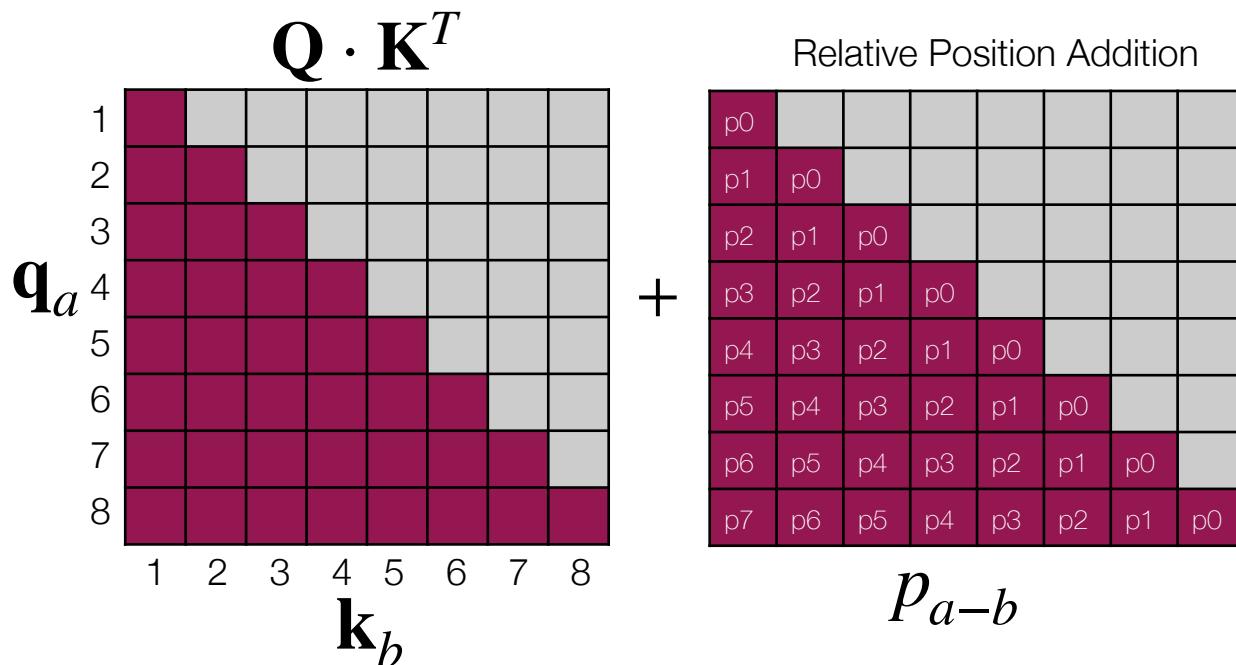
What if, instead of perturbing each embedding based on position, we perturb the multiplication of two embeddings, **based on their relative position?**



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

Relative Positional Encoding

- Don't encode raw position, encode relative position between \mathbf{q}_a and \mathbf{k}_b at position a and b
- practically, attention $\rightarrow \text{softmax}(\mathbf{Q} \cdot \mathbf{K}^T + p_{a-b})$



- (+) Nicely structured relative position information
- (-) Lots more memory
- (-) Couples position and attention (harder to distribute ops)

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

Smart relative position encoding

- Ideally, for sequence $[\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_a, \dots, \mathbf{x}_b, \dots, \mathbf{x}_L]$, relative position attention (positions a and b) is given by:

$$\begin{aligned} f(\mathbf{x}_a, \mathbf{x}_b, a - b) &= \underbrace{\left(\mathbf{W}^{(q)} \cdot \mathbf{x}_a \right)^T}_{\mathbf{q}_a} R_{a-b} \underbrace{\left(\mathbf{W}^{(k)} \cdot \mathbf{x}_b \right)}_{\mathbf{k}_b} \\ &= \mathbf{q}_a^T \cdot R_{a-b} \cdot \mathbf{k}_b \\ &= \mathbf{q}_a^T \cdot R_a R_b \cdot \mathbf{k}_b \\ &= (\mathbf{q}_a^T \cdot R_a)(R_b \cdot \mathbf{k}_b) = \hat{\mathbf{q}}_a^T \cdot \hat{\mathbf{k}}_b \end{aligned}$$

When Multiplied,
still encodes relative pos

Still about as
Efficient as Attention

Precompute $(\mathbf{q}_a \cdot R_a) \rightarrow \hat{\mathbf{q}}_a$
 $(R_b \cdot \mathbf{k}_b) \rightarrow \hat{\mathbf{k}}_b$
softmax($\hat{\mathbf{q}}_a^T \cdot \hat{\mathbf{k}}_b$)
softmax($f(\mathbf{x}_a, \mathbf{x}_b, a - b)$)
relative position attention

- what if R_{a-b} can be decoupled into $R_a R_b$ for efficiency?

$$R_a = e^{j \cdot \theta a} \quad R_b = e^{-j \cdot \theta b} \quad R_a R_b = e^{j \cdot \theta(a-b)} = R_{a-b}$$

one solution: rotations in the complex plain, eigenfunctions

Smart relative position encoding

- but practically $R_a R_b = e^{j \cdot \theta(a-b)} = R_{a-b}$ requires complex valued arithmetic (not ideal for efficiency). So we can get the same benefit via:

$$f(\mathbf{x}_a, \mathbf{x}_b, a - b) = \operatorname{Re}[\mathbf{q}_a^T \cdot R_a R_b \cdot \mathbf{k}_b] = \mathbf{q}_a^T \cdot \operatorname{Re}[R_a R_b] \cdot \mathbf{k}_b$$

- which in the 2 element case reduces to the rotation matrix:

$$R_a \cdot \mathbf{q}_a = \begin{bmatrix} \cos(a \cdot \theta) & -\sin(a \cdot \theta) \\ \sin(a \cdot \theta) & \cos(a \cdot \theta) \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$$

Rotate \mathbf{q} by $a \theta$, precomputed

$$R_b \cdot \mathbf{k}_b = \begin{bmatrix} \cos(b \cdot \theta) & -\sin(b \cdot \theta) \\ \sin(b \cdot \theta) & \cos(b \cdot \theta) \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \end{bmatrix}$$

Rotate \mathbf{k} by $b \theta$, precomputed

- and we effectively get relative position encoding for $(\mathbf{q}_a^T \cdot R_a)(R_b \cdot \mathbf{k}_b)$, but with decoupled operations!!!
 - but, expanding beyond 2D vectors starts to make the operation too computational... so let's only do operation in pairs along \mathbf{q} and along \mathbf{k} separately (lots of 2D rotations)

Rotary Position Encoding, RoPE

- Now we can finally understand RoPE:

$$R_a = \begin{bmatrix} \cos(a \cdot \theta_1) & -\sin(a \cdot \theta_1) & 0 & 0 & \dots & 0 & 0 \\ \sin(a \cdot \theta_1) & \cos(a \cdot \theta_1) & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \cos(a \cdot \theta_2) & -\sin(a \cdot \theta_2) & \dots & \vdots & \vdots \\ & & \sin(a \cdot \theta_2) & \cos(a \cdot \theta_2) & \dots & \vdots & \vdots \\ & & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \text{Rotate by } a \cdot \theta_2 & \dots & 0 & \cos(a \cdot \theta_{D/2}) & -\sin(a \cdot \theta_{D/2}) \\ 0 & 0 & \dots & 0 & 0 & \sin(a \cdot \theta_{D/2}) & \cos(a \cdot \theta_{D/2}) \end{bmatrix}$$

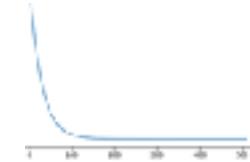
$\theta_i = 10000^{-2(i-1)/D}$ defines the range of rotations, D is size of q, k

Speed: Fast to implement with two point wise vector multiplies and addition (low overhead, still parallel)

$$R_a \cdot \mathbf{q}_a = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ \vdots \\ q_{D-1} \\ q_D \end{bmatrix} \cdot \begin{bmatrix} \cos(a \cdot \theta_1) \\ \cos(a \cdot \theta_1) \\ \cos(a \cdot \theta_2) \\ \cos(a \cdot \theta_2) \\ \vdots \\ \cos(a \cdot \theta_{D/2}) \\ \cos(a \cdot \theta_{D/2}) \end{bmatrix} + \begin{bmatrix} -q_2 \\ q_1 \\ -q_4 \\ q_3 \\ \vdots \\ -q_D \\ q_{D-1} \end{bmatrix} \cdot \begin{bmatrix} \sin(a \cdot \theta_1) \\ \sin(a \cdot \theta_1) \\ \sin(a \cdot \theta_2) \\ \sin(a \cdot \theta_2) \\ \vdots \\ \sin(a \cdot \theta_{D/2}) \\ \sin(a \cdot \theta_{D/2}) \end{bmatrix}$$

In general, produces **better results** (mostly) while being **not too computational** (a “pretty good” relative encoding)

Lots of pairwise rotations, each preserving the property of relative position encoding

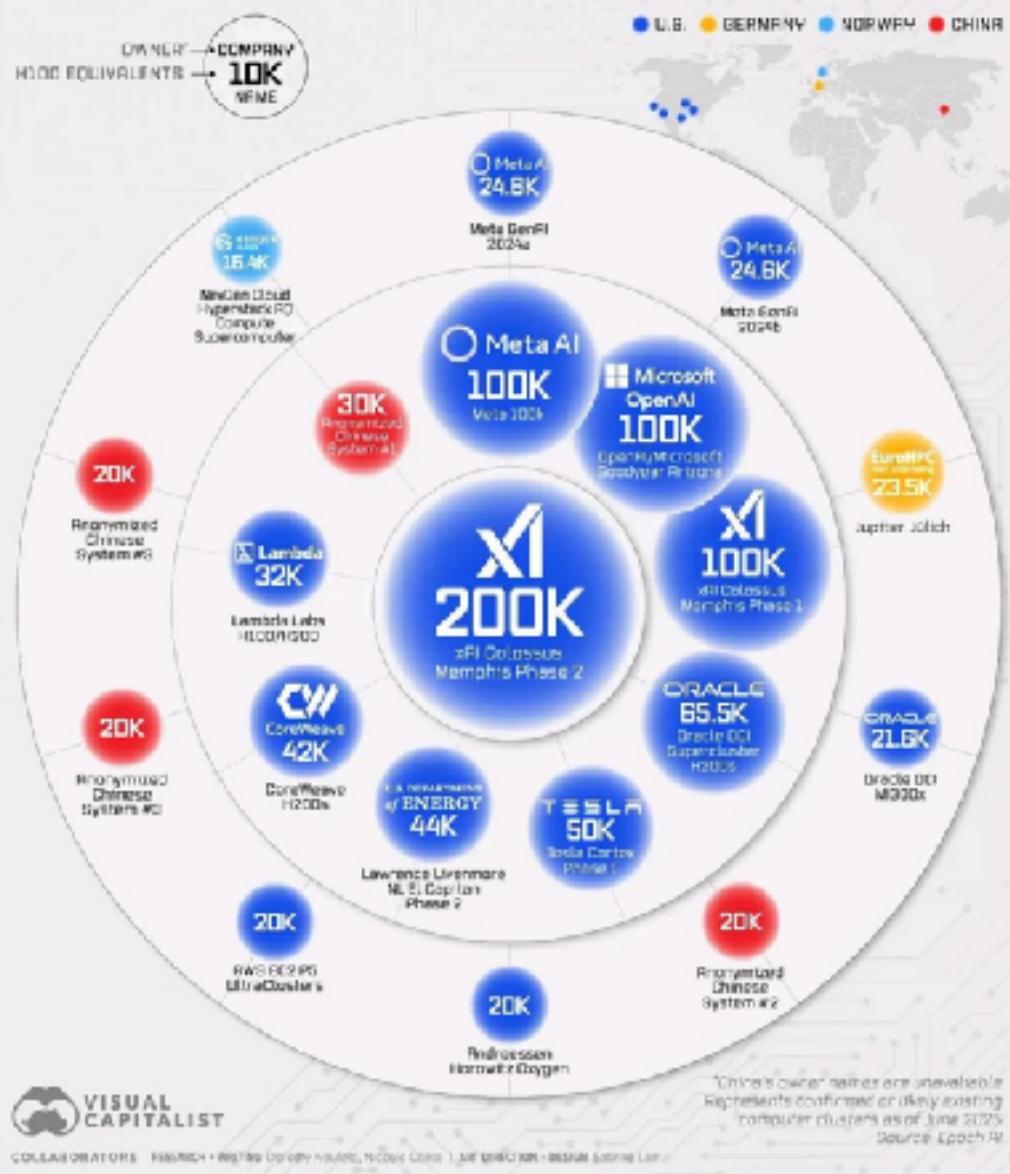


High frequency, sensitive to position

Transformer learns to encode positionally sensitive aspects in high frequency indices...

Low frequency, less sensitive to position

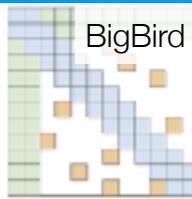
THE WORLD'S MOST POWERFUL Compute Clusters



Altering Self-Attention



Attention Efficiently

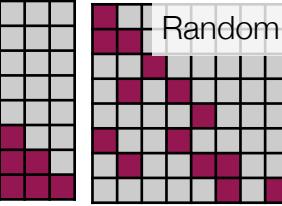
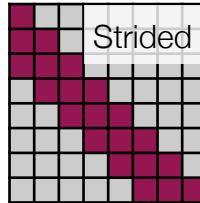


Partial Global + Rand + Strided
Zaheer et al., **BigBird**, NeurIPS 2021

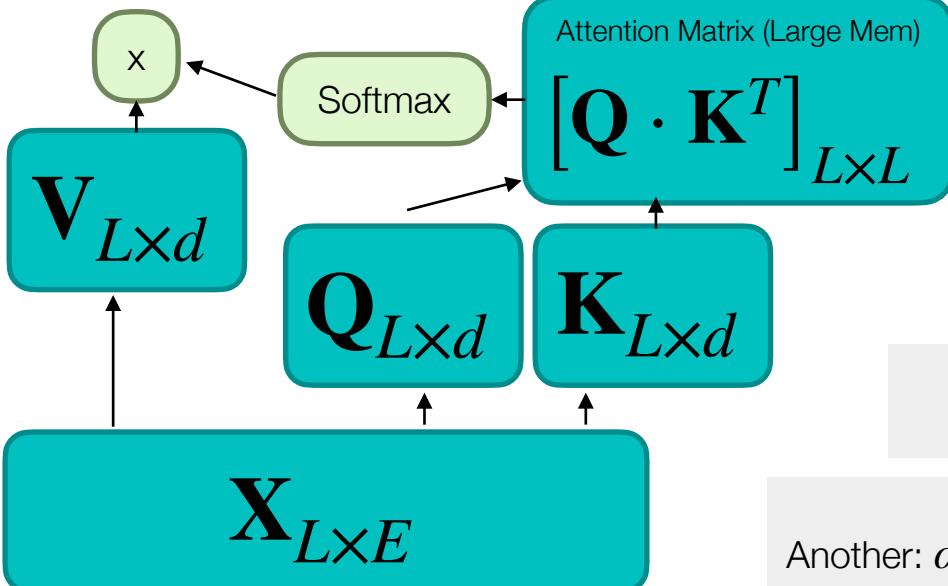
Naive Implementation:

- Computation: $O(L^2 \cdot d)$
- Memory: $O(L^2 + L \cdot d)$

One idea: limit non-zero values of $\mathbf{Q} \cdot \mathbf{K}^T$
Need to define sparsity before computation



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



Another idea: change softmax, to allow associative rule application
 $(\mathbf{Q} \cdot \mathbf{K}^T) \cdot \mathbf{V} = \mathbf{Q} \cdot (\mathbf{K}^T \cdot \mathbf{V})$
 $O(L^2 \cdot d)$ $O(L \cdot d^2)$

Efficient Implementation:

- Computation: $O(L \cdot d^2)$
- Memory: $O(d^2 + L \cdot d)$

but we need a function that satisfies
 $f(\mathbf{Q} \cdot \mathbf{K}^T) = f(\mathbf{Q}) \cdot f(\mathbf{K}^T)$ *linearized attention*

One function: softmax along rows and columns
 $\text{softmax}(\mathbf{Q}) \cdot (\text{softmax}(\mathbf{K})^T \cdot \mathbf{V})$

Exponential Linear Unit
 $f(x) = \max(0, (e^x - 1) \cdot \alpha)$

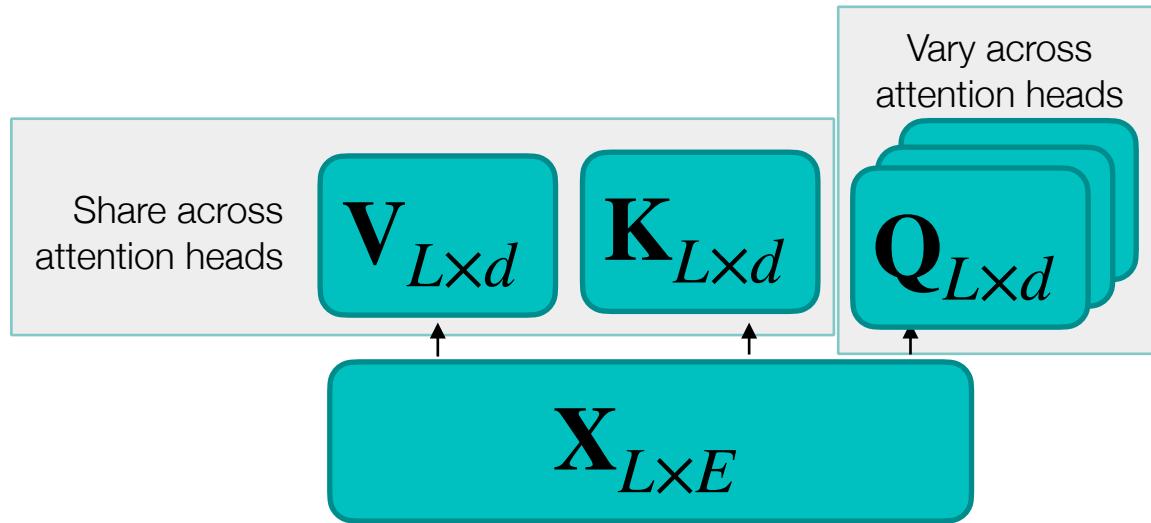
Katharopoulos et al.,
ICLR 2021

Another: $\alpha \frac{\mathbf{Q}}{\|\mathbf{Q}\|} \cdot \left(\frac{\mathbf{K}^T}{\|\mathbf{K}\|} \cdot \mathbf{V} \right) \rightarrow \alpha \frac{\mathbf{Q} \cdot \mathbf{K}^T}{\|\mathbf{Q}\| \|\mathbf{K}\|} \cdot \mathbf{V}$

same as cosine similarity

Mongaras, Dohm, and Larson, **Cottention**, CC 2025

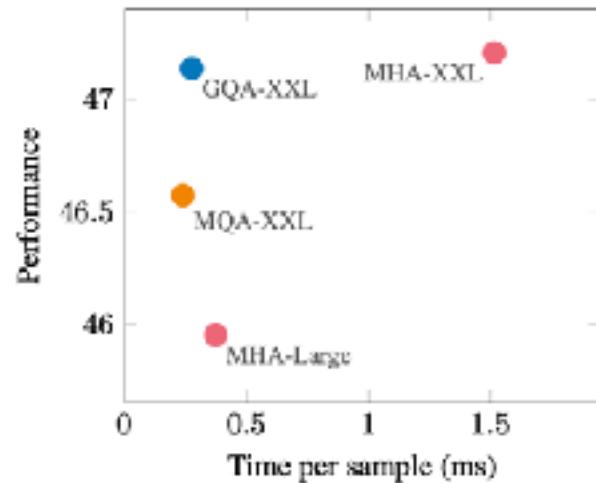
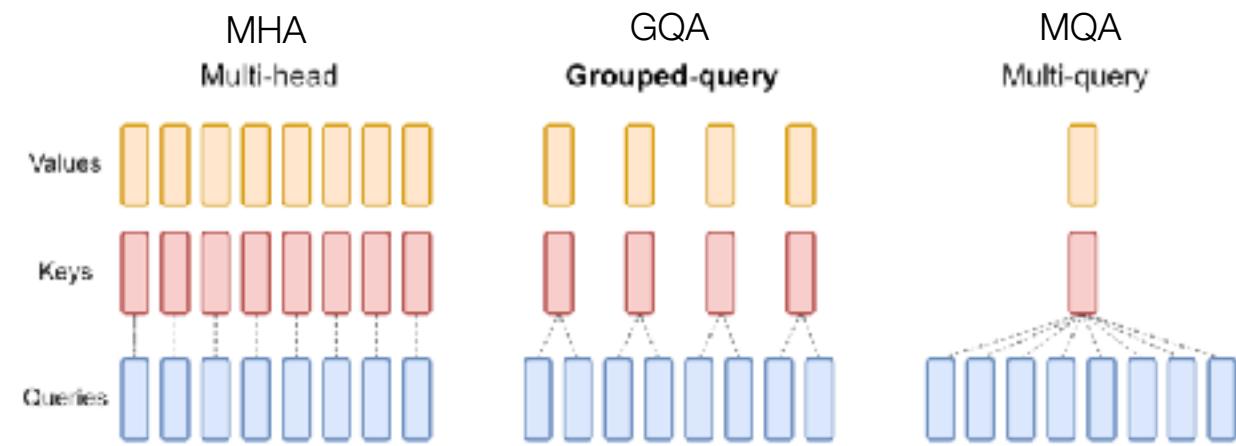
Efficiency, Multi-query Attention (MQA)



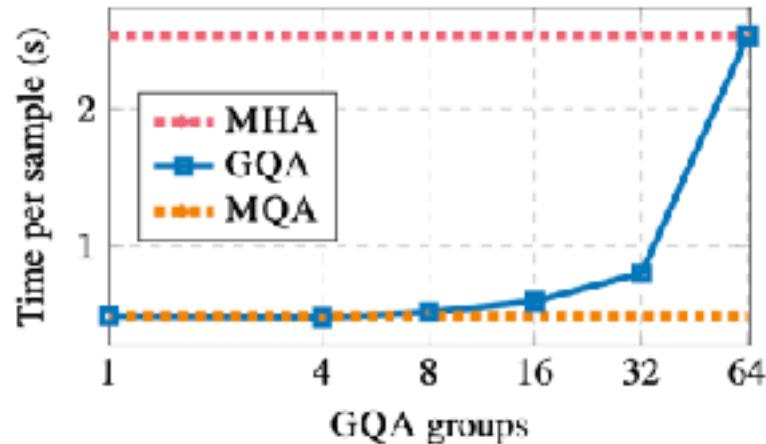
- Share across attention heads
 - Vary across attention heads
 - (-) Slight drop in accuracy for various tasks
 - (+) Allows larger transformer feed forward layers
 - (+) larger context lengths fit in GPU memory
 - (-) No speed up for distributed compute as K, V are copied
-
- Vanilla transformer can store V and K on SRAM of GPU, then just load in Q from high-bandwidth memory (HBM)
 - memory transfer is critical bottleneck for GPU, so you get a huge speed up
 - For linear methods that can calculate $\mathbf{Q}_i \cdot (\mathbf{K}^T \cdot \mathbf{V})$, the entire $\mathbf{K}^T \cdot \mathbf{V}$ matrix can be precomputed (*may not fit in SRAM for long sequences*)

Chelba, Fast Transformer Decoding, ArXiV (2020)

Group Query Attention

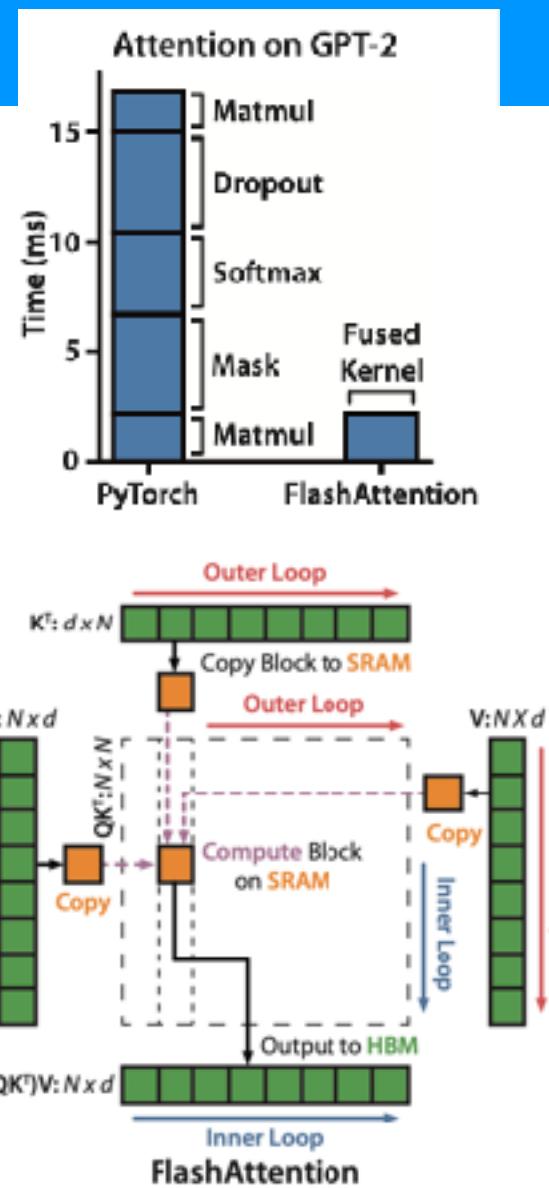


- Can take advantage of distributed computation, parallelize across groups for unique K, V
- Easy to tradeoff performance of MHA with compute of MQA



Flash Attention

- Calculate attention in tiles (local compute)
 - Requires calculation and saving additional variables in each tile
- During tile aggregation, scale the variables properly to get exact softmax across tiles (distributed softmax)
- Tile calculation is a shader function, massive speed up on a GPU
- Back-prop: Only save the attention output and recompute it for back propagation to save memory
 - similar to gradient checkpointing, this adds compute but saves memory
- **Flash Attention 2:**
 - Some small improvements to matrix multiplications
 - Added support for GQA (big speed ups)
 - Flash Attention becomes 9x faster than normal attention for both training and inference



Distributing Softmax in Tiles

$$\mathbf{x} = [a, b, c, d]$$

$$m(\mathbf{x}) = \max([a, b, c, d])$$

$$f(\mathbf{x}) = [e^{a-m(\mathbf{x})}, e^{b-m(\mathbf{x})}, e^{c-m(\mathbf{x})}, e^{d-m(\mathbf{x})}]$$

$$l(\mathbf{x}) = \sum f(\mathbf{x})$$

$$\text{softmax}(\mathbf{x}) = \frac{f(\mathbf{x})}{l(\mathbf{x})} = \left[\frac{e^{a-m(\mathbf{x})}}{l(\mathbf{x})}, \frac{e^{b-m(\mathbf{x})}}{l(\mathbf{x})}, \frac{e^{c-m(\mathbf{x})}}{l(\mathbf{x})}, \frac{e^{d-m(\mathbf{x})}}{l(\mathbf{x})} \right]$$

$$\begin{aligned}\mathbf{x}^{(1)} &= [a, b] & f(\mathbf{x}^{(1)}) &= [e^{a-m(\mathbf{x}^{(1)})}, e^{b-m(\mathbf{x}^{(1)})}] \\ \mathbf{x}^{(2)} &= [c, d] & f(\mathbf{x}^{(2)}) &= [e^{c-m(\mathbf{x}^{(2)})}, e^{d-m(\mathbf{x}^{(2)})}]\end{aligned}$$

$$m(\mathbf{x}) = \max(m(\mathbf{x}^{(1)}), m(\mathbf{x}^{(2)}))$$

Need to track this

$$s_i = e^{m(\mathbf{x}^{(i)}) - m(\mathbf{x})}$$

Slightly more FLOPS, but better utilization of parallelism

$$f(\mathbf{x}) = [s_1 \cdot f(\mathbf{x}^{(1)}), s_2 \cdot f(\mathbf{x}^{(2)}),]$$

$$l(\mathbf{x}) = s_1 \cdot l(\mathbf{x}^{(1)}) + s_2 \cdot l(\mathbf{x}^{(2)})$$

Combine distributed ops:

$$\text{softmax}(\mathbf{x}) = \frac{f(\mathbf{x})}{l(\mathbf{x})}$$

End of lecture material

Course Retrospective

Ilya Sutskever (@ilyasut) Hinton's Student

i find it both obvious and incredible that a neural network is a digital brain that lives inside a computer (and that actually kinda works)

2:10 PM · Jun 19, 2022 · Twitter Web App

721 Retweets 61 Quote Tweets 12.9K Likes

[Reply](#) [Retweet](#) [Like](#) [Share](#)

Tweet your reply

Elon Musk (@elonmusk) · 14h
Replying to @ilyasut
Maybe we're in a computer

4,608 Retweets 2,720 Quote Tweets 33.4K Likes

Microsoft just laid off

Virginia Dignum is also @vdignum · 21h

Replies to @emilybender After LeCunn AGI Evolution

My reply to Yann:
"Is really sad to see CS folk being so misled by our own language. An artificial neural network reassembles a neural network only in name! 🤖
Do you also expect airplanes to evolve into birds just because both fly?
#AI is not intelligence."

4,898 Retweets 1 Quote Tweet 8 Likes 49 Likes

Course Retrospective

Sincerely yours,

Chris Atkeson
Peter Bartlett
Andrew Barto
Jonathan Baxter
Yoshua Bengio
Kristin Bennett
Chris Bishop
Justin Boyan
Carla Brodley
Claire Cardie
William Cohen
Peter Dayan
Tom Dietterich
Jerome Friedman
Nir Friedman
Zoubin Ghahramani
David Heckerman
Geoffrey Hinton
Haym Hirsh
Tommi Jaakkola
Michael Jordan
Leslie Kaelbling
Daphne Koller
John Lafferty
Sridhar Mahadevan
Marina Meila
Andrew McCallum
Tom Mitríchell
Stuart Russell
Lawrence Saul
Bernhard Schoelkopf
John Shawe-Taylor
Yoram Singer
Satinder Singh
Padhraic Smyth
Richard Sutton
Sebastian Thrun
Manfred Warmuth
Chris Williams
Robert Williamson

- AI winters existed because we **over-hyped** power of machine learning, imprecise wording, ... for money and power, and greed
 - **and history will repeat**
- Formal methods around deep learning are **not as formal as other fields**
- At the end of the day, we are still using **back propagation**
- **Open source** is a big reason we can still have advancements without formalism:
 - <http://www.jmlr.org/statement.html>

Leading ML researchers issue statement of support for JMLR.

POST-MORTAL ANALYSIS: THE JOURNAL OF COMPUTATIONAL STATISTICS AND DATA SCIENCE, VOLUME 58, 2013 1180–99
Brought to you by: University of Wisconsin-Madison, from JSTOR's eScholarship platform

Dear colleagues in machine learning,

we hereby support open access journals that publish papers from the field of machine learning. We believe that open access journals are the best way to ensure that our research papers receive the maximum visibility for our authors, and to facilitate access to the implications that our work has for society at large.

We consider JMLR an example of an excellent model of innovative change in the way that research publications are disseminated. JMLR papers are research papers that are submitted online, peer-reviewed online, and published online. They receive a higher citation rate than papers in classical journals such as those in the *Journal of the ACM* (ACM SIGART publication). The funds of the *JMLR* are derived from donations from individuals, foundations, and companies, and the journal makes no institutional affiliation fees or royalties.

Please note that open access journals are not necessarily the best choice for every publication. There are many good reasons to publish in traditional journals, and we do not wish to discourage anyone from doing so. However, we believe that open access journals are the best way to ensure that our work receives the widest possible exposure. We encourage and support the use of open access journals for the dissemination of our work, and encourage our students, postdocs, and colleagues to consider publishing their research in open access journals whenever possible.

Some of the greatest advances in our field have been made by people who have chosen to publish in open access journals. We believe that open access journals are an important part of the future of our field. We invite you to consider publishing your work in an open access journal, and to support the development of these journals.

Topics review

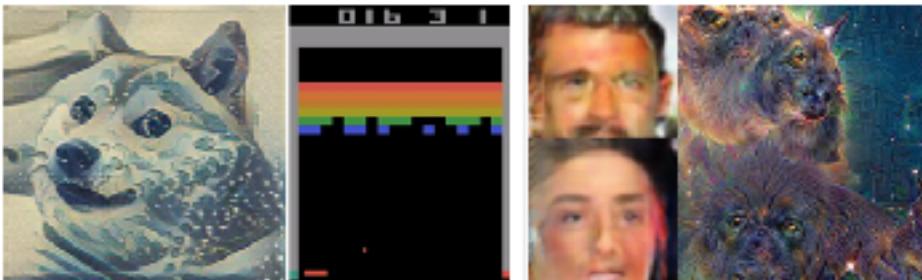
Setup, Review

- Data **munging** in pandas and numpy and **visualization** with matplotlib, pandas, seaborn
- Data preprocessing: **dim reduction**, images, text, categorical features, **embeddings**
- **Linear models**: linear regression, logistic regression, simple neural networks
- **Optimization** strategies: Gradient ascent, Quasi-Newton, Extensions of SGD (RMSProp, AdaM)
- **Back propagation** in MLP (from scratch)
- Tensorflow/Keras for **wide and deep networks**
- **Convolutional** neural networks (up to modern day)
- **Sequential** neural networks (transformers, CNNs)

Deep Learning

Topics Not Covered

- Methods to Assist Ethics in ML
- Transfer/Multi-Task Learning
- Visualizing CNNs
- Fully Convolutional Networks
- Style Transfer (maybe)
- Generative Networks
- Large Language Models



Course Schedule

Week	Lecture #	Content #	Last Date
1	Lecture: Course Introduction and Syllabus	Lecture: Week 1: Neural Networks	
2	Student Presentations (Reviewing: LeCun's Form of Deep Learning, CIFAR-2017) Student Discourse: Generative, TBC	Lecture: CNN Visualization Overview	Lecture: Week 2: Autoencoders, Metrics, Reading: Chapter 5, Section 4
3	Lecture/Demo: Image Transfer		Lecture: Image Style Transfer Overview, Reading: Chapter 5, Sections 7 and 8
4	Student IP presentations (Reviewing: LeCun's Form of Deep Learning, CIFAR-2017) Student Discourse: Generative, TBC	Lecture: Generator and Critic, GANs	Student IP presentations (Reviewing: LeCun's Form of Deep Learning, CIFAR-2017) Student Discourse: Generative, TBC
5	Lecture/Demo: Image Style Transfer in PyTorch		Lecture: Transfer Learning in GANs, Reading: Chapter 5, Section 2 and 3
6	Lecture/Demo: Generative Models on MNIST		Lecture: Multi-modal learning Overview
7	Student IP presentations (Reviewing: Deep Multimodal Learning, Generative Adversarial Networks and GANs)	Lecture: GANs: The Basics	Student IP presentations (Reviewing: Deep Multimodal Learning, Generative Adversarial Networks and GANs)
8	Lecture: Generative Adversarial Networks Overview	Reading: Chapter 5, Sections 1, 2, and 3	Lecture: Multi-Modal and Multi-Task
9	Student Presentations and Readings: Recurrent Principles, Methods for Training GRNNs, Attention and LSTM	Reading: Chapter 5, Sections 1, 2, and 3	Lecture: Multi-Modal and Multi-Task
10	Lecture: Deep Reinforcement Learning Overview		Lecture: Deep Reinforcement Learning Overview
11	Lecture: Dual Policy Policy Optimization		Lecture: Dual Policy Policy Optimization, part 1
12	Student Presentations (Reviewing: Playing Asteroids Using Reinforcement Learning, part 1)	Lecture/Demo: OpenAI Gym, Part 1	Lecture Demo: Playing Deep Reinforcement Learning in Games
13	Lecture: The Future of Deep Learning	Reading: Chapter 9	Lecture: Deep Flow
14	Review of Final Project Paper		Review of Final Project Paper
15	Review of Final Project Paper		Review of Final Project Paper

Learning and Neural Networks

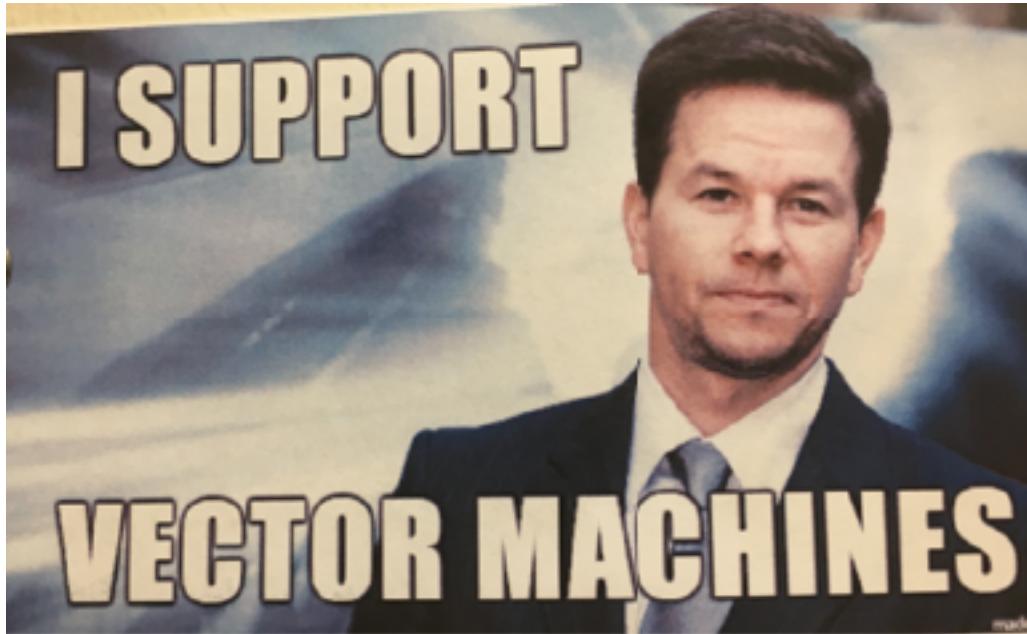
Overview

This course requires basic knowledge of the use of Neural Networks in machine learning beyond simple prediction, especially targeted outputs that are generation or alteration of images, text, and audio. This course emphasizes topics of neural networks in the “deep learning” subdomain. This course will survey of important topics and current areas of research, including transfer learning, multi-task and multi modal learning, image style transfer, neural network visualization, deep convolutional generative adversarial networks, and deep reinforcement learning. For grading, students are expected to complete smaller learn-based projects throughout the semester, present one research paper in a 15-20 minute group presentation (covering topics in the course), and complete a comprehensive final project that involves a number of different deep learning architectures.

Thank you for a great semester!

- but it could **have been better** somehow, right?
 - How could you learn better, more reliably for an interview?
 - When did you feel like you **built proficiency**? Versus when was **critical thought** not attained?
 - what should **not be cut** or **not changed**?
 - **Already cut:** SVMs, Random forests, Boosting, Ensembles, RNNs, many-to-many,
 - Two courses: (1) Intro ML and (2) Deep Learning
 - More APIs? Turi / PyTorch?
 - More flipped Assignments?
 - Fewer coding demos?
 - Self-guided Jupyter notebooks?
 - Exams?

Thank You for an Excellent Semester!



Courtesy of Omar Roa

Please fill out the course evaluations!!!!