Attention

Mert Pilanci

March 2, 2026

Electrical Engineering
Stanford University

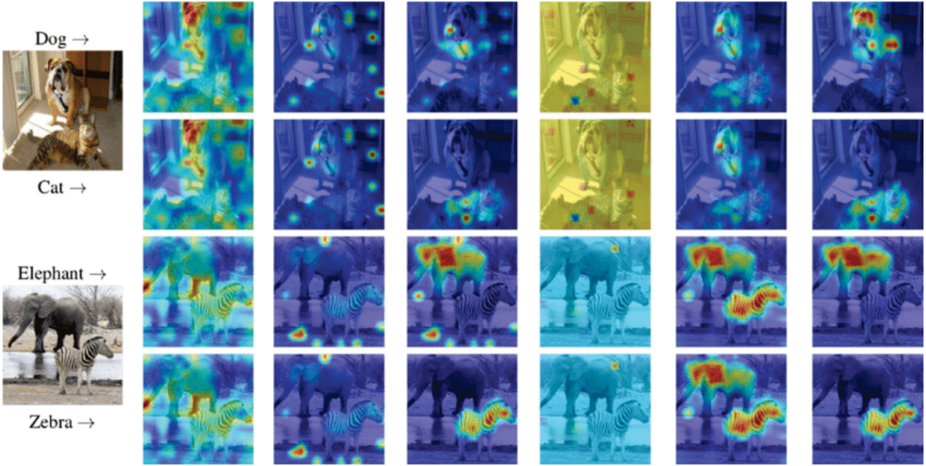
EE269

Outline

- attention mechanism
- transformer networks
- applications in signal processing
- text-conditioned diffusion
- model compression

Attention Mechanism: Motivation

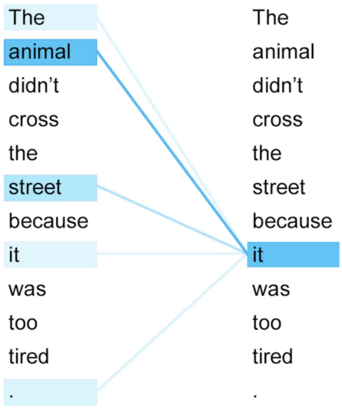
- visual attention: classifying dog vs cat and elephant vs zebra



Transformers in computational visual media: A survey, Xu et al., 2022

Attention Mechanism: Motivation

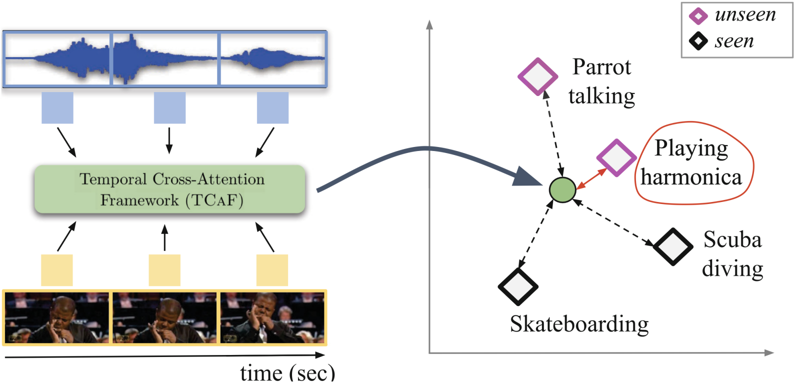
- textual attention
- consider the sentence **"The animal didn't cross the street because it was too tired."**



Transformer: A Novel Neural Network Architecture for Language Understanding, Jakob Uszkoreit, 2017

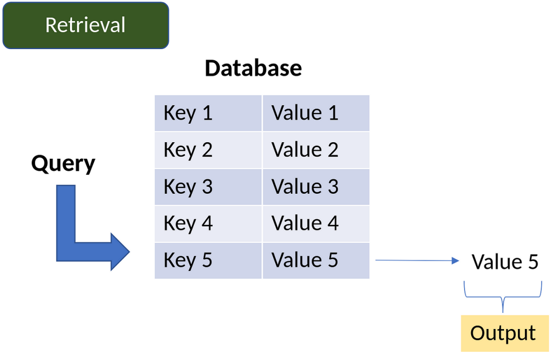
Attention Mechanism: Motivation

- audio-visual attention
- consider paired audio and video features



Temporal and Cross-modal Attention for Audio-Visual Zero-Shot Learning, Mercea et al., 2022

Analogy: Data retrieval in a database



Terminology

- given a set of **value** vectors and a **query** vector, **attention** is a way to compute a **weighted sum of the values dependent on the query**.
- the query determines what values to focus on,

  we say: the query **“attends”** to the values
- the keys are used to compute the attention scores
- the values are used to compute the output vector

Attention Mechanism in a Neural Network

- suppose we have two input matrices X, Y representing n vectors of dimension d
  $X = [x_1, ..., x_n]^T$ and $Y = [y_1, ..., y_n]^T$
- introduce trainable linear weights W_K, W_V, W_Q
- **keys:** $k_i = W_K y_i \in \mathbb{R}^{d_k}$
- **values:** $v_i = W_V y_i \in \mathbb{R}^{d_v}$
- **query:** $q = W_Q x \in \mathbb{R}^{d_k}$
- attention scores are given by

$$a_i = \frac{1}{\sqrt{d_k}} k_i^T q \quad \text{for } i = 1, ..., n$$

$$\text{output:} \quad \sum_{i=1}^{n} \mathrm{softmax}_i(a)\, v_i$$

- linear weights are learned via backpropagation
- normalization ensures $\mathrm{Var}\left(\frac{1}{\sqrt{d_k}} k_i^T q\right) = 1$

Standard Attention Layer (Cross-Attention)

- two input matrices X, Y representing n vectors (tokens) $\in \mathbb{R}^d$
  $X = [x_1, ..., x_n]^T$ and $Y = [y_1, ..., y_n]^T$
- define

$$Q = XW_Q, \quad K = YW_K, \quad V = YW_V$$

- output is the matrix Z given by

$$A = \mathrm{softmax}\Big(\frac{1}{\sqrt{d_k}} QK^T\Big)$$

$$Z = AV$$

- $\mathrm{softmax}$ is applied row-wise
- $W_Q \in \mathbb{R}^{d \times d_k}$, $W_K \in \mathbb{R}^{d \times d_k}$, $W_V \in \mathbb{R}^{d \times d_v}$ are trainable weight matrices

Neural Machine Translation by Jointly Learning to Align and Translate, Bahdanau et al., 2016

Self-Attention Layer in Explicit Matrix Form

- take $Y = X$

  $X = [x_1, ..., x_n]^T \in \mathbb{R}^{n \times d}$
- attention weights and output:

$$Z = \operatorname{softmax}\Big(\frac{1}{\sqrt{d_k}} X W_Q W_K X^T\Big) X W_V$$

Convex Combination Interpretation

- $Q = XW_Q$: query matrix
- $K^T = XW_K$: key matrix
- $V = XW_V$: value matrix

$$Z = \mathrm{softmax}\Big(\frac{1}{d_k} QK^T\Big)V$$

- $Z_i = \sum_j p_{ij} V_j$ is a convex combination of the columns of V
- $p_{ij} \geq 0$ and $\sum_j p_{ij} = 1$ are input dependent mixture weights

Attention as Adaptive Matched Filtering

- consider one query q and keys $k_1, \ldots, k_n$
- attention scores:

$$a_i = \frac{1}{\sqrt{d_k}} k_i^T q$$

- this is a **matched filter** (correlation detector) which is optimal for detecting a fixed waveform in Gaussian noise
- softmax produces adaptive weights:

$$\alpha_i = \frac{e^{a_i}}{\sum_j e^{a_j}}$$

- output $z = \sum_{i=1}^{n} \alpha_i v_i$
- interpretation:

  keys are a learned dictionary

  attention selects the most correlated atoms

  output is a data-dependent convex combination

Copy Previous Token Task

- input sequence of length 4

$$x_1, x_2, x_3, x_4$$

- task:

$$y_t = x_{t-1}$$

- desired output:

$$Z = \begin{bmatrix} 0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Copy Previous Token Task

- input sequence of length 4

$$x_1, x_2, x_3, x_4$$

- suppose each token is represented in the delta basis:

$$x_1 = e_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \ x_2 = e_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \ x_3 = e_3, \ x_4 = e_4$$

- stack them into matrix

$$X = \begin{bmatrix} e_1^T \\ e_2^T \\ e_3^T \\ e_4^T \end{bmatrix} = I_4$$

Attention Can Implement the Copy Operation

- choose

$$W_Q = I, \quad W_K = S, \quad W_V = I$$

- where S is the shift matrix

$$S = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

- then

$$Q = XW_Q = I$$
$$K = XW_K = S$$

- compute attention logits:

$$QK^T = IS^T = S^T$$

- observe: each row of S^T is one-hot

Toy Example

- consider a toy problem of sequence-to-sequence prediction
- the target averages the heights in each pair of shapes

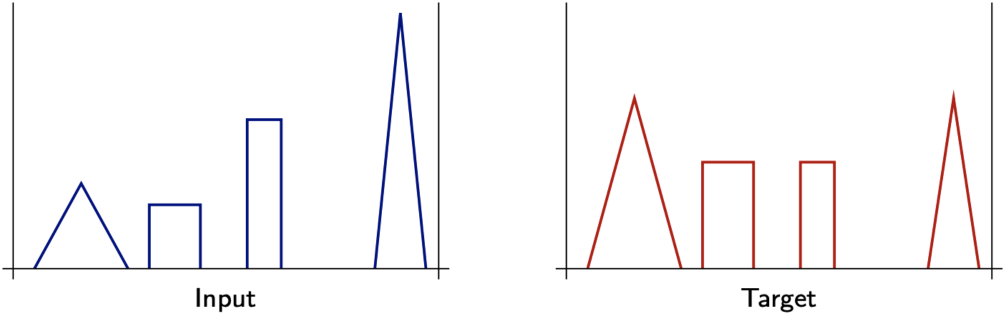


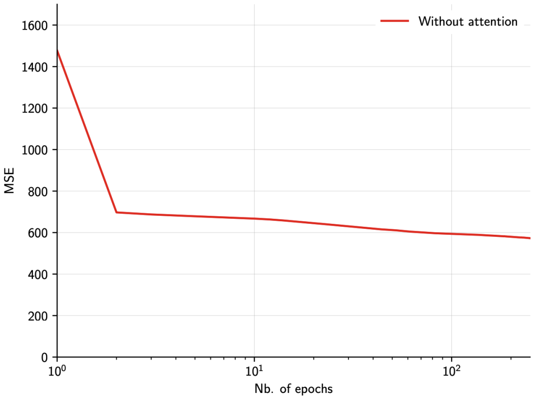
image credit: Francois Fleuret

Convolutional Nets

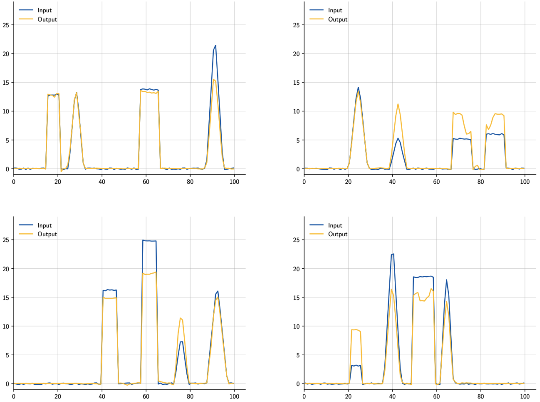
```
Sequential(
  (0): Conv1d(1, 64, kernel_size=(5,), stride=(1,), padding=(2,))
  (1): ReLU()
  (2): Conv1d(64, 64, kernel_size=(5,), stride=(1,), padding=(2,))
  (3): ReLU()
  (4): Conv1d(64, 64, kernel_size=(5,), stride=(1,), padding=(2,))
  (5): ReLU()
  (6): Conv1d(64, 64, kernel_size=(5,), stride=(1,), padding=(2,))
  (7): ReLU()
  (8): Conv1d(64, 1, kernel_size=(5,), stride=(1,), padding=(2,))
)

nb_parameters 62337
```

image credit: Francois Fleuret

Convolutional Nets



Convolutional Nets



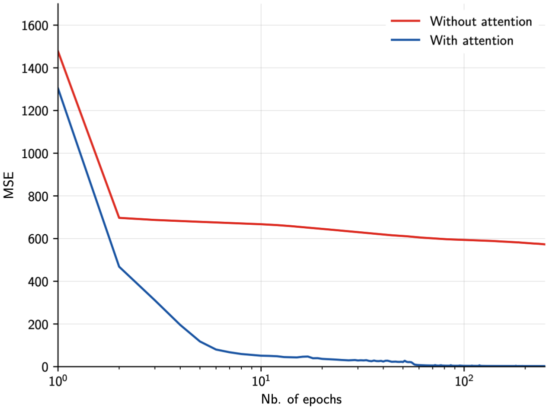
Attention Layer in a Convolutional Network

```
Sequential(
  (0): Conv1d(1, 64, kernel_size=(5,), stride=(1,), padding=(2,))
  (1): ReLU()
  (2): Conv1d(64, 64, kernel_size=(5,), stride=(1,), padding=(2,))
  (3): ReLU()
  (4): SelfAttentionLayer(in_dim=64, out_dim=64, key_dim=64)
  (5): Conv1d(64, 64, kernel_size=(5,), stride=(1,), padding=(2,))
  (6): ReLU()
  (7): Conv1d(64, 1, kernel_size=(5,), stride=(1,), padding=(2,))
)

nb_parameters 54081
```

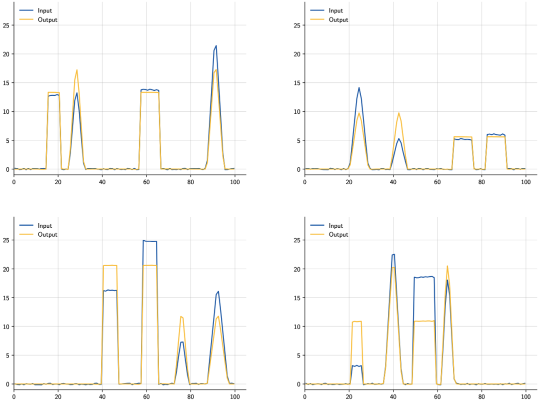
Attention Layer

```
class SelfAttentionLayer(nn.Module):
    def __init__(self, in_dim, out_dim, key_dim):
        super().__init__()
        self.conv_Q = nn.Conv1d(in_dim, key_dim, kernel_size = 1, bias = False)
        self.conv_K = nn.Conv1d(in_dim, key_dim, kernel_size = 1, bias = False)
        self.conv_V = nn.Conv1d(in_dim, out_dim, kernel_size = 1, bias = False)

    def forward(self, x):
        Q = self.conv_Q(x)
        K = self.conv_K(x)
        V = self.conv_V(x)
        A = Q.transpose(1, 2).matmul(K).softmax(2)
        y = A.matmul(V.transpose(1, 2)).transpose(1, 2)
        return y
```

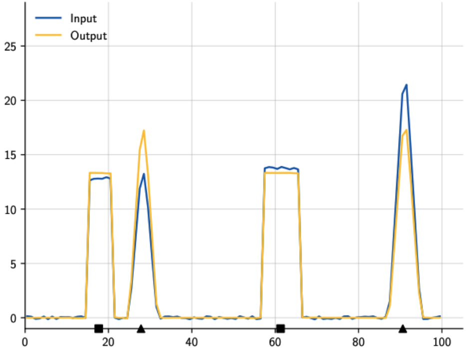
Training Loss

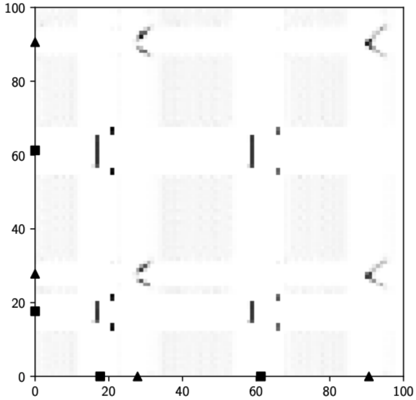


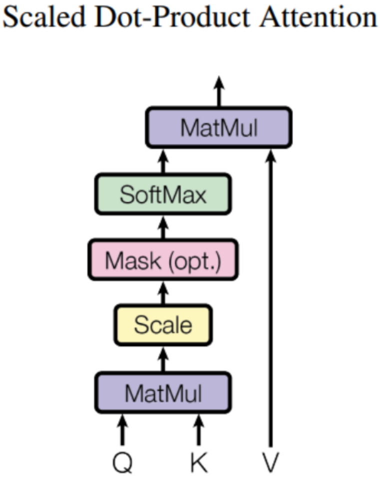
Predictions via Attention

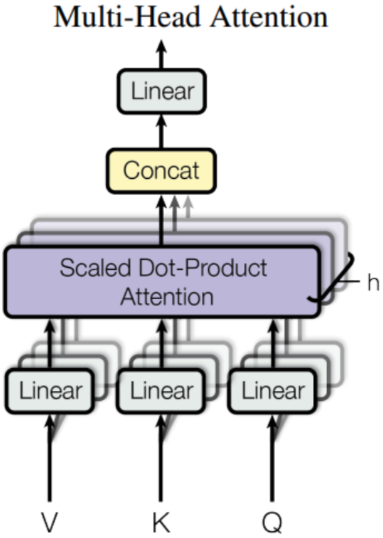


Attention Patterns

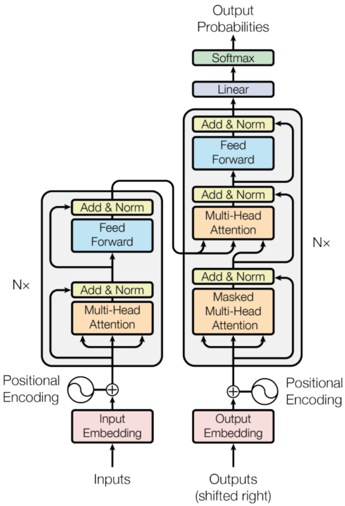





Scaled Dot-Product Attention
MatMul
SoftMax
Mask (opt.)
Scale
MatMul
Q
K
V


Multi-Head Attention
Linear
Concat
Scaled Dot-Product Attention
h
Linear
Linear
Linear
V
K
Q

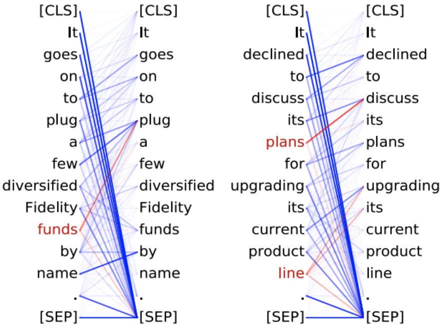
Transformer Networks



Attention Is All You Need, Vaswani et al., 2017

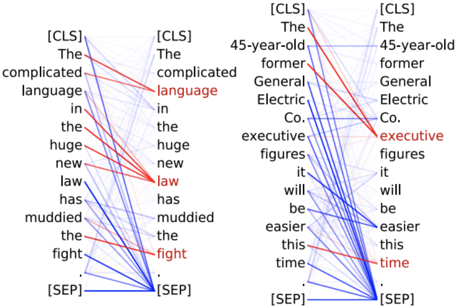
Head 8-10

- **Direct objects** attend to their verbs
- 86.8% accuracy at the `dobj` relation



Head 8-11

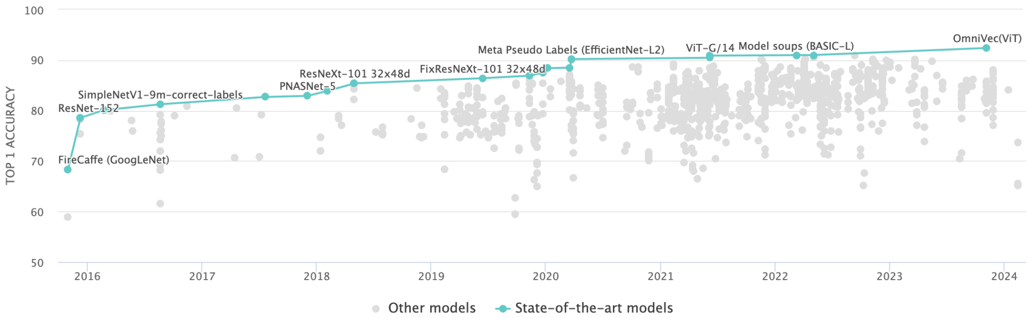
- **Noun modifiers** (e.g., determiners) attend to their noun
- 94.3% accuracy at the `det` relation



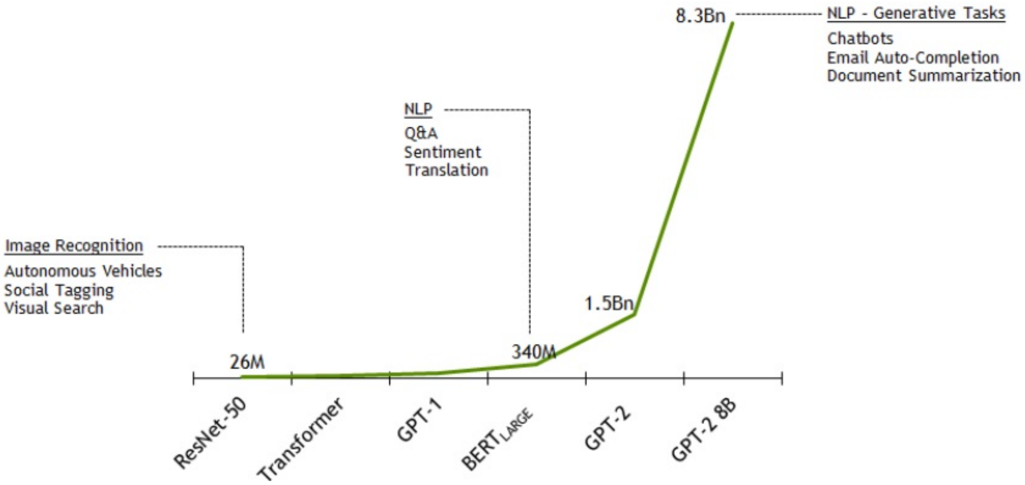
(Clark et al., 2019)

Imagenet Challenge

- Vision Transformers (ViT)

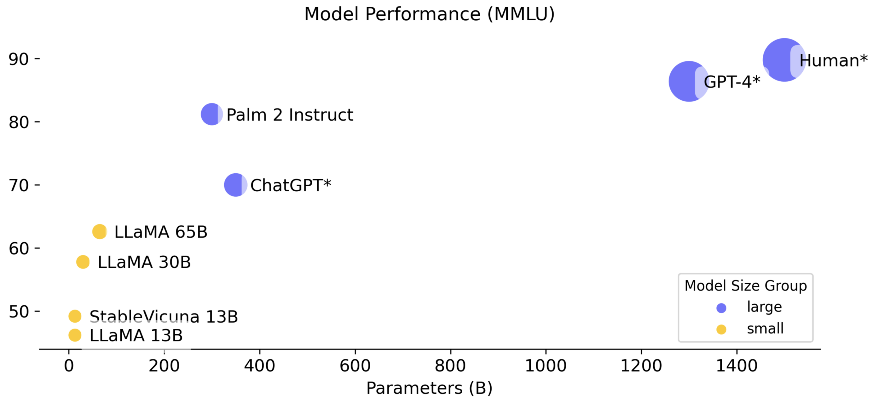


Sizes of Transformer Networks

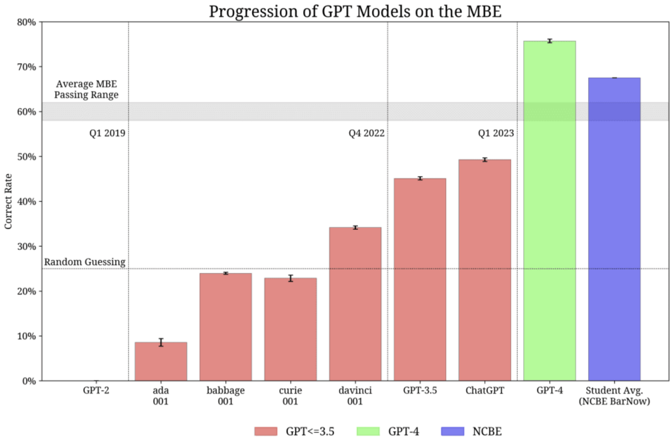


MMLU Benchmark

- Measuring Massive Multitask Language Understanding
- covers 57 tasks including elementary mathematics, US history, computer science, law, and more

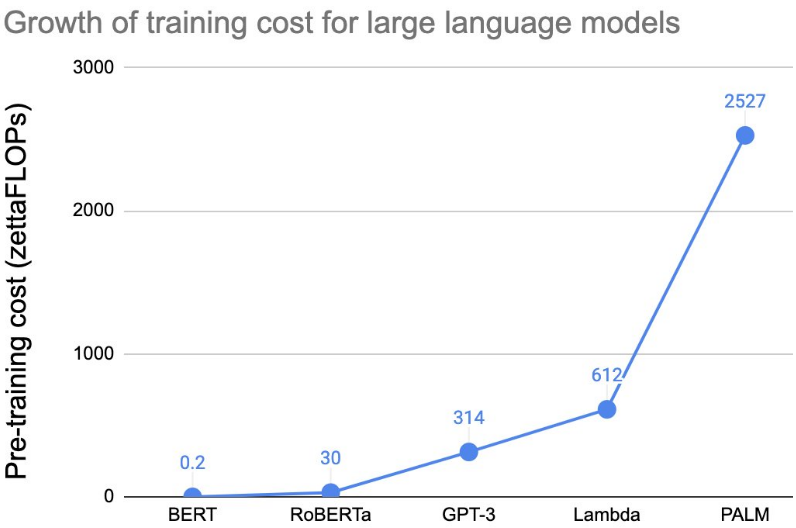


The Multistate Bar Exam



The Multistate Bar Exam (MBE) is a challenging battery of tests designed to evaluate an applicant's legal knowledge and skills, and is a precondition to practice law in the US.

Compute Requirements



prompt: A photograph of an astronaut riding a horse

Text-Conditioned Image Diffusion

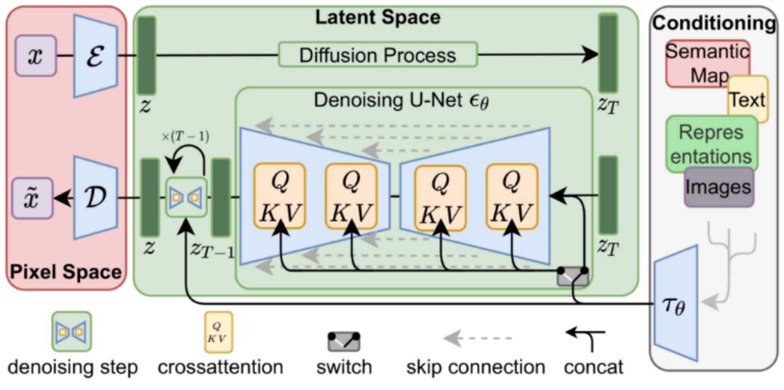


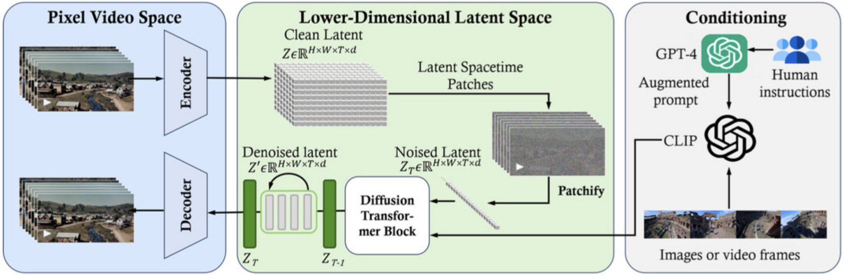
Figure 3. We condition LDMs either via concatenation or by a more general cross-attention mechanism.

Sora (OpenAI), 2024

Prompt: Animated scene features a close-up of a short fluffy monster kneeling beside a melting red candle. The art style is 3D and realistic, with a focus on lighting and texture. Th...
more

0:04 / 0:08

Text-Conditioned Video Diffusion



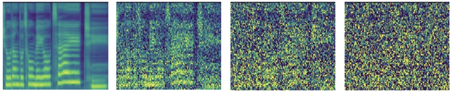
Spectrogram diffusion

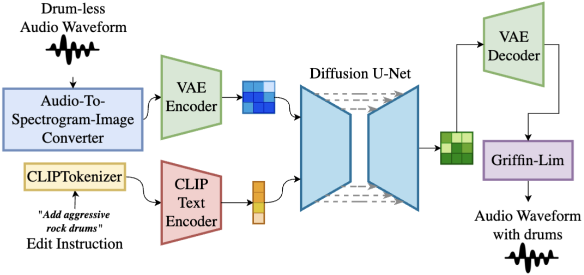
convert audio to spectrograms and apply the diffusion model

separate the spectrograms into components (e.g., vocals, drums, bass, guitar etc.)

train a diffusion model to generate a component given the rest using a text prompt

generate audio by inverting the spectrograms (e.g., via Griffin-Lim algorithm)





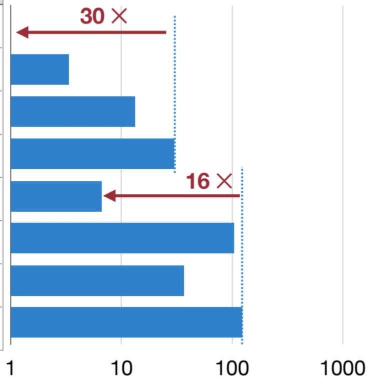
- **prompt:** "add jazz drums"

Villa-Renteria, Wang, Pilanci, "Subtractive Training for Music Stem Insertion using Latent Diffusion Models", 2024

Quantization and Energy Costs

Operation	Energy [pJ]
8 bit int ADD	0.03
32 bit int ADD	0.1
16 bit float ADD	0.4
32 bit float ADD	0.9
8 bit int MULT	0.2
32 bit int MULT	3.1
16 bit float MULT	1.1
32 bit float MULT	3.7

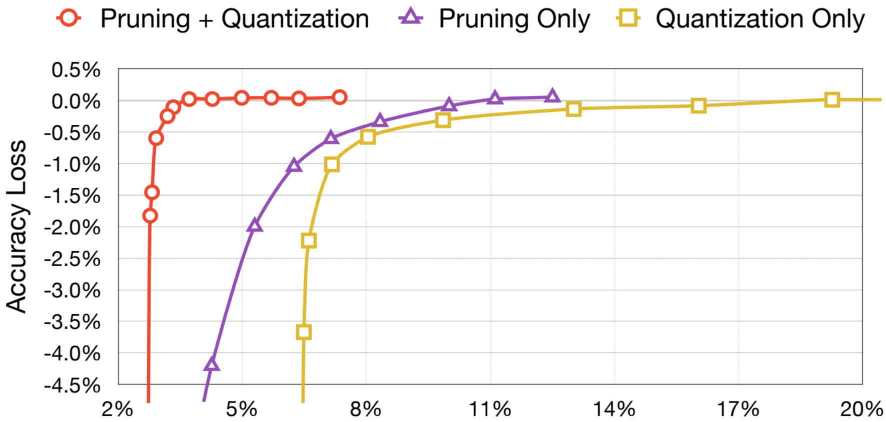
Rough Energy Cost For Various Operations in 45nm 0.9V



M. Horowitz, Computing's Energy Problem (and what we can do about it), 2014

Quantizing Deep Neural Networks

- Han, Mao, Dally, Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding, 2015
  convolutional network: AlexNet (240MB) trained on the ImageNet dataset

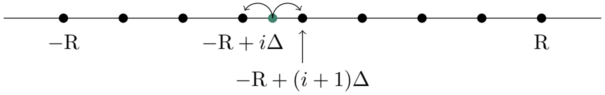


Uniform Quantization

B-bit quantizer $\mathrm{Q}(x)$. Dynamic range R, i.e., $-\mathrm{R} \leq x \leq \mathrm{R}$.

Quantization points: $q_1 = -\mathrm{R}, q_2 = -\mathrm{R} + \Delta, \ldots, q_{2^\mathrm{B}} = -\mathrm{R} + (2^\mathrm{B} - 1)\Delta$.

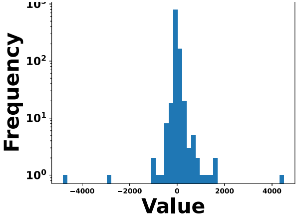
Do nearest-neighbor quantization.



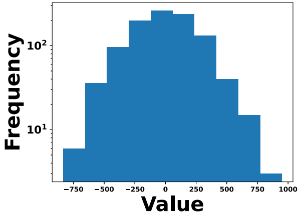
Resolution is $\Delta = \frac{\mathrm{R}}{2^\mathrm{B} - 1}$. Do coordinate wise for vectors. Error $\|\mathrm{Q}(\mathbf{x}) - \mathbf{x}\|_2 = \mathrm{O}(\sqrt{d})$.

Random Embeddings Compress the Dynamic Range

- generate a random matrix S and compute Sx
- randomized embeddings equalize the coordinate magnitudes
- the entries of Sx are (approximately) Gaussian-distributed



vector with large outliers



after random matrix multiplication

Lyubarskii & Vershynin, 2006

Saha, Pilanci, Goldsmith, Low Precision Representations for High-Dimensional Models, 2023

Improved Uniform Quantization via Random Embeddings

Choice of $\mathbf{S}$:

- Gaussian: $S_{ij} \sim \mathcal{N}\left(0, \frac{1}{m}\right)$. (Dense: $\mathrm{O}(d^2)$ multiplications)
- Randomized Hadamard: $\mathbf{S} = \frac{1}{\sqrt{d}}\mathbf{H}\mathbf{D}$, where $\mathbf{H} \in \{-1,+1\}^{d\times d}$ is the Hadamard matrix, $\mathbf{D}$ is a diagonal ± 1 matrix. (Recursive: $\mathrm{O}(d \log d)$ additions and sign-flips)

$$\|\mathbf{S}\mathbf{x}\|_\infty \leq \mathrm{O}\left(\frac{\|\mathbf{x}\|_2}{\sqrt{d}}\right) \quad \text{or,} \quad \|\mathbf{S}\mathbf{x}\|_\infty \leq \mathrm{O}\left(\sqrt{\frac{\log d}{d}}\|\mathbf{x}\|_2\right) \qquad \text{w.h.p.}$$

Quantization error (w.h.p.):

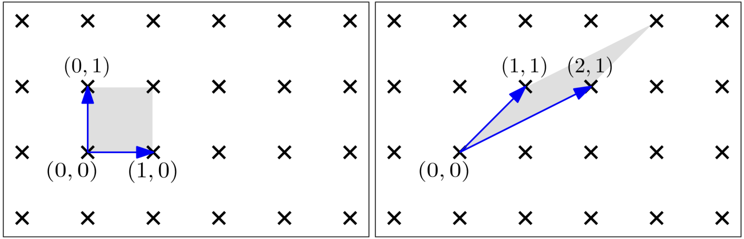
$$\|\mathrm{Q}(\mathbf{S}\mathbf{x}) - \mathbf{S}\mathbf{x}\|_2 = \begin{cases} \mathrm{O}(1) & \text{for } \mathbf{S} \text{ Gaussian,} \\ \mathrm{O}(\sqrt{\log d}) & \text{for } \mathbf{S} \text{ Randomized Hadamard.} \end{cases}$$

Lyubarskii & Vershynin, 2006

Further Improvement: Lattice Quantization

- **definition:** Given n linearly independent vectors $b_1, b_2, ..., b_n \in \mathbb{R}^d$ the lattice generated by them is defined as

$$\left\{ \sum_{i=1}^{n} b_i x_i \mid x_i \in \mathbb{Z} \right\} = \left\{ Bx \mid x \in \mathbb{Z}^n \right\}$$



(a) A basis of $\mathbb{Z}^2$ (b) Another basis of $\mathbb{Z}^2$

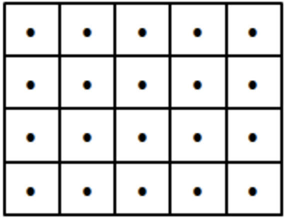
- we round x to its nearest point in the lattice

Lattices

- **definition:** Given n linearly independent vectors $b_1, b_2, ..., b_n \in \mathbb{R}^d$ the lattice generated by them is defined as

$$\left\{ \sum_{i=1}^{n} x_i b_i \mid x_i \in \mathbb{Z} \right\} = \left\{ Bx \mid x \in \mathbb{Z}^n \right\}$$

- the square lattice is the product of uniform scalar quantizers

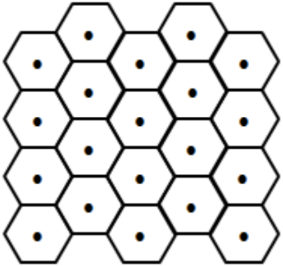


Lattices

- **definition:** Given n linearly independent vectors $b_1, b_2, ..., b_n \in \mathbb{R}^d$ the lattice generated by them is defined as

$$\left\{ \sum_{i=1}^{n} x_i b_i \mid x_i \in \mathbb{Z} \right\} = \left\{ Bx \mid x \in \mathbb{Z}^n \right\}$$

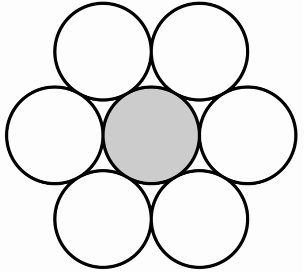
- the hexagonal lattice has lower MSE for a given number of quantization points in a given area with uniform density. MSE improvement is $\frac{5\sqrt{3}}{9} = 0.962$

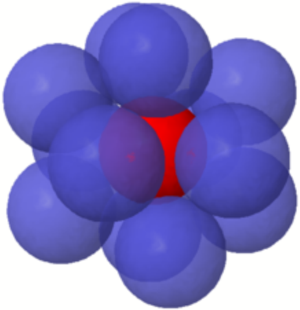


- E_8 lattice achieves the 8 dimensional **kissing number**

$$E_8 = \left\{ x \mid x \in \left(\mathbb{Z}^8 \cup \left(\mathbb{Z} + \frac{1}{2} \right)^8 \right) \wedge \sum_i x_i \equiv 0 \pmod{2} \right\}$$

- **kissing number:** the maximum number of unit balls touching a central unit ball
- Maryna Viazovska won the Fields Medal in 2022 for the proof that E_8 provides the densest packing of identical spheres in 8 dimensions





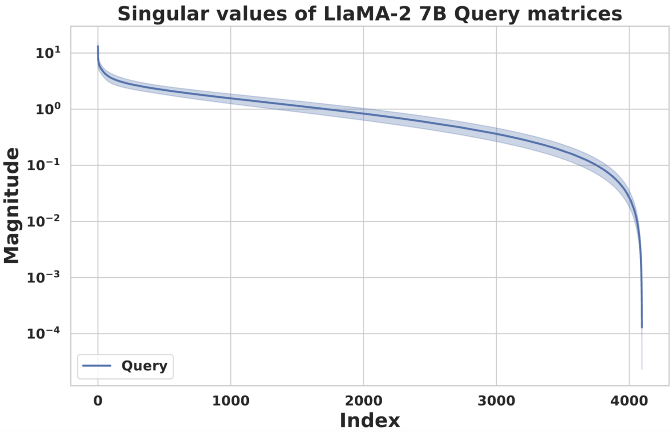
- kissing number is 6 in $\mathbb{R}^2$, 12 in $\mathbb{R}^3$ and 240 in $\mathbb{R}^8$

- E_8 lattice gives the densest packing in 8 dimensions
- a basis is given by

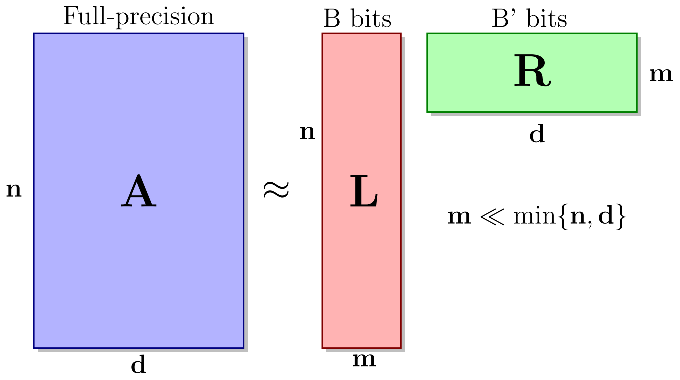
$$\begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \frac{1}{2} \end{bmatrix}$$

- E_8 has 2^{16} elements but only requires looking up into a size codebook 2^8
- finding the nearest neighbor can be done efficiently
- Viazovska and co-authors later proved that the Leech lattice is optimal in 24 dimensions
- E_8 is used in compressing large language models (QuIP$\#$ Tseng et al., Feb 2024)
  2-2.5 bits per weight on average with reasonable fidelity

Singular Value Decay



Low Precision Low-Rank (LPLR) Decomposition



Saha, Srivastava, Pilanci, Matrix Compression via Randomized Low Rank and Low Precision Factorization, NeurIPS 2023

Quantizing Large Language Models via Low-Rank Low-Precision Decomposition

- low-rank low-precision decomposition with lattice quantization
- **CALDERA**: **C**alibration **A**ware **L**ow-Precision **DE**composition with Low-**R**ank **A**daptation (Saha, Sagan, Srivastava, Pilanci, May 2024)

Table 1: Zero-shot perplexities (denoted by ↓) and accuracies (↑) for LLaMa-2. $B_Q = 2$ bits throughout.

Method	Rank	$B_L (= B_R)$	Avg Bits	Wiki2 ↓	C4 ↓	Wino ↑	RTE ↑	PiQA ↑	ArcE ↑	ArcC ↑
CALDERA (7B)	64	16	2.4	7.36	9.47	64.6	66.4	73.7	60.8	31.7
CALDERA (7B)	64	4	2.1	7.37	9.74	63.7	62.1	72.3	60.9	31.7
CALDERA (7B)	128	4	2.2	6.76	8.83	63.8	59.9	75.1	**65.1**	**34.6**
CALDERA (7B)	256	4	2.4	**6.19**	**8.14**	**66.0**	60.6	**75.6**	63.6	34.0
QuIP# (7B, No FT)	64	16	2.4	7.73	10.0	63.1	**66.8**	71.7	63.2	31.7
QuIP# (7B, No FT)	0	—	2	8.23	10.8	61.7	57.8	69.6	61.2	29.9
CALDERA (70B)	64	16	2.2	**4.50**	**6.38**	**75.4**	**71.7**	**79.2**	71.8	**43.9**
CALDERA (70B)	128	4	2.1	5.07	7.10	72.9	62.1	78.0	**73.2**	**43.9**
QuIP# (70B, No FT)	0	—	2	5.37	7.51	72.3	47.6	77.7	68.8	40.9
Unquantized (7B)	0	—	16	5.12	6.63	67.3	63.2	78.5	69.3	40.0
Unquantized (70B)	0	—	16	3.12	4.97	77.0	67.9	81.1	77.7	51.1

QuIP# was the state-of-art quantization method as of April 2024