Advanced Machine Learning: Auto-Regressive Models

Week 3

So Far, Discriminative Models

- Mature field
- Defining the learning task was straight forward
- Mainly served us for motivating neural architecture
- Main challenges are:
  - architecture selection
  - hyparameter-tuning
  - But mostly, data collection and labeling

Next: Generative Models

- Previously minor goal: put the world into buckets
- Grand goal: generating the world
- Much harder task!
- Learn a model that can generate the world
- Reconsider ImageNet
- Discrimanative model: how like is this image to contain a dog?
- Generative model: how likely is this image to be a dog image?

Generative Modeling Tasks

- Generative models: P(x)
  - Estimation: find density function Q that approximates P
  - Sampling: draw samples from P
  - Point estimation: compute the probabilty density of sample x

Example: ImageNet without generalization

- Estimation: Q(x) that assigns:
  - 1/N to every imagenet training image
  - 0 otherwise
- Sampling: sample a random ImageNet training image
- Point estimation: query Q for image x

Example: Non-Gen Results on Test Set

- Sampling: cannot sample test set images
- Point estimation: test set images give 0 probability

Example: ImageNet with generalization

- Estimation: Q(x) - true distribution that generated ImageNet
- Sampling: from the true distribution, i.e. new ImageNet images
- Point sampling: How likely is x in the true distribution of ImageNet

Conditional Generative Models

- Model P(x|y) rather than P(x)
- Conditionining can be class labels, text prompt etc.

Example: Class-conditioned ImageNet

- Sampling:
  - sample a new image, given its class is Dalmatian
- Point estimation:
  - how likely is this image to come from the Dalmatian dog distribution

Why Do We Care About This?

- World models e.g., GPT (this week)
- Human-guided creativity e.g., Text-to-image (in a few weeks)

Several Paradigms for Probability Estimation

- Auto-regressive / Non-AR (this week)
- Variational
- Adversarial
- Flow-based
- Score-based / Diffusion

Why Learn So Many Paradigms

- All have pros and cons
- Sometimes a combination works best
- Need to choose different paradigms for situation

Tower Rule of Probability

- P(X, Y) = P(X|Y)P(Y)
  - This is always true. Independence is when P(X|Y) = P(X)
- In the case of N variables, this implies the following decomposition:

$$\begin{aligned} P(X_1, X_2, \ldots X_n) &= P(X_1)P(X_2|X_1)P(X_3|X_1, X_2)\ldots \\ &= \prod_{i=1}^{n} P(X_i|X_1, \ldots, X_{i-1}) \end{aligned}$$

Turning This into an ML Task

- We would like to estimate $P(x_{i+1}|x_1, x_2..x_i)$

- Assume that X_i is a categorical variable
- Learn $Q(x_{i+1}|x_1, x_2..x_i)$
- Estimate Q via cross-entropy:

$$\ell(P(x), Q(x)) = -\sum_j P(j|x_1, x_2..x_i) \log(Q(j|x_1, x_2..x_i))$$

Example: Language Modeling

- Input: a set of sentences decomposed into tokens
  - Here: token = word
- Estimation task: learn model that predict next token, given previous
- We **do not** observe the true $P(j|x_1, x_2..x_i)$
- We **do** observe finite samples from $P(j|x_1, x_2..x_i)$

Example: Language Modeling (2)

- Sample: “the”,”cat”,”sat”,”on”,”the”,”mat”
- Q(“on”| “the”,”cat”,”sat”) – **high** likelihood
- Q(“mat”| “the”,”cat”,”sat”,”on”,”the”) – **high** likelihood
- Q(“ate”| “the”,”cat”,”sat”,”on”,”the”) – **low** likelihood

Learning in Practice

- Given a set S containing sequences (($x_1, x_2..x_i$), y)
- Learn classifer:

$$L = -\sum_{(x_1,x_2..x_i),y \in S} \sum_j 1_{y=j} \log(Q(j|x_1, x_2..x_i))$$

- Or more simply:

$$L = -\sum_{(x_1,x_2..x_i),y \in S} \log(Q(y|x_1, x_2..x_i))$$

Language Modeling (2)

- In our example, we had a large set of (sentence, next word)
- Define each word as a class
- Learn a classifier that takes a sequence of words and returns class
- It should handle different sequence lengths 0<i<=N

$$L = -\sum_{(x_1, x_2..x_i), y \in S} \log(Q(y|x_1, x_2..x_i))$$

Running Example

- The training sample “the cat sat on the mat”
- Transform to Q(y|”the”,”cat”,”sat”,”on”,”the”)
- Many possible classes: “the”,”cat”,”sat”,”on”,”ate”,”apple”,”dog” etc.
- Objective: loss cross entropy with (0,1,0,0,0,0,0,0,…)

$$L = -\sum_{(x_1, x_2 .. x_i), y \in S} \sum_j 1_{y=j} \log(Q(j|x_1, x_2 .. x_i))$$

Sampling from an AR Model

- Assume we trained an AR model $Q(j|x_1, x_2..x_i)$
- We want to generate a new token sequence $x_1..x_N$
- Objective: generate the **most likely** sequences
- Option 1: select the most likely token
  - Lacks diversity

Sampling New Data (2)

- To sample multiple likely sequence, we need stochasticity
- Option 2: sample according to the distribution of $Q(j|x_1, x_2..x_i)$
  - high diversity, but maybe too much
- Option 3: sample from the top-K tokens
  - K can tune the amount of diversity

Point Estimation

- Provided a model, point estimation is quite easy with AR models
- The point probability is simply:

$$Q(X) = \prod_{i=1}^{N} P(x_i | x_1, x_2 .. x_{i-1})$$

- Evaluating log probability is more numerically stable

Extension to Conditional AR Models

- The framework is easy extended to conditional probs.
- The tower rule in this case:

$$Q(X|c) = \prod_{i=1}^{N} P(x_i|x_1, x_2..x_{i-1}, c)$$

- Model estimation, sampling, point estimation: virtually unchanged

Perplexity: Measure of Model Quality

- Test perplexity: the geometric mean of Q on the test set

$$\prod_{m=1}^{M} Q(X^m)^{-\frac{1}{M}}$$

- Intuition: samples of real data should have high likelihood
- In our example:
  - Test sentences: high probability
  - Garbled sentences: low probability

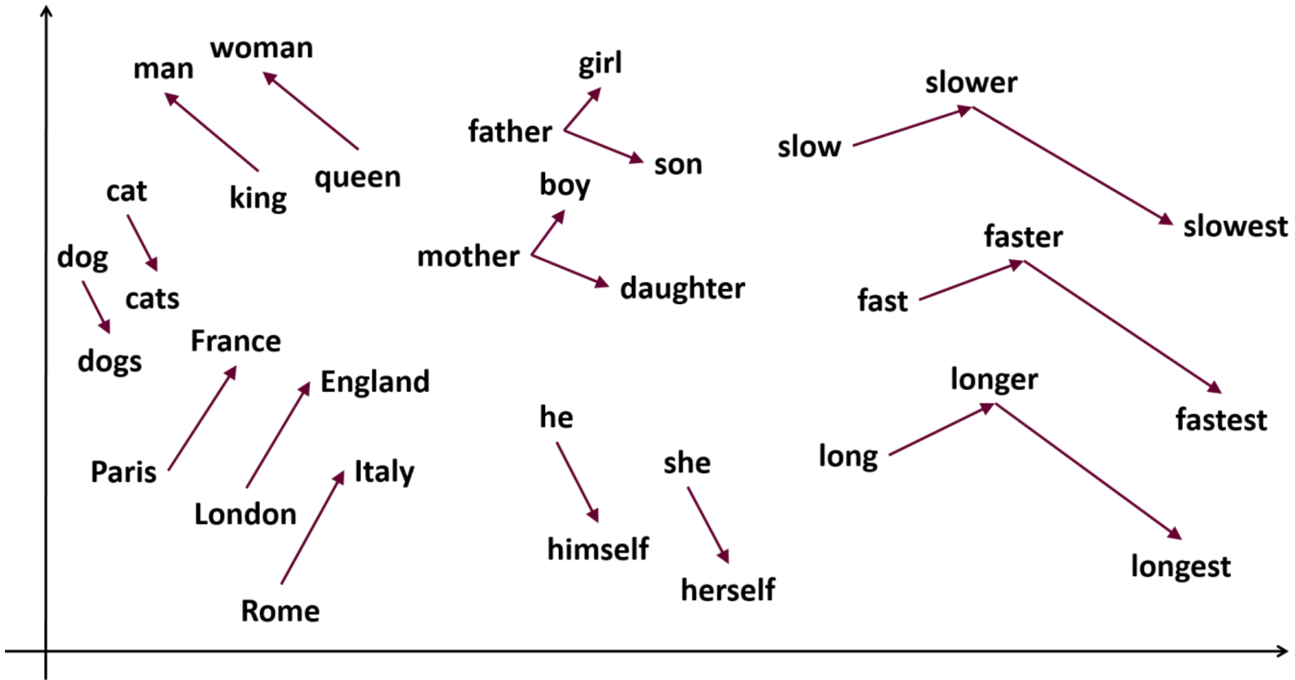
Simple Example: Linear Language Model

- Every token *t* is described by a one-hot vector $\mathbf{e}_t$
- A linear layer **W** transforms each token into a dense vector $\mathbf{W}\mathbf{e}_t$
- Features of all tokens are average pooled into a fixed representation
- The average vector is linearly classified to the next word matrix **V**

$$soft\max(V^T W \sum_{t=1}^{N} \frac{e_t}{N})$$

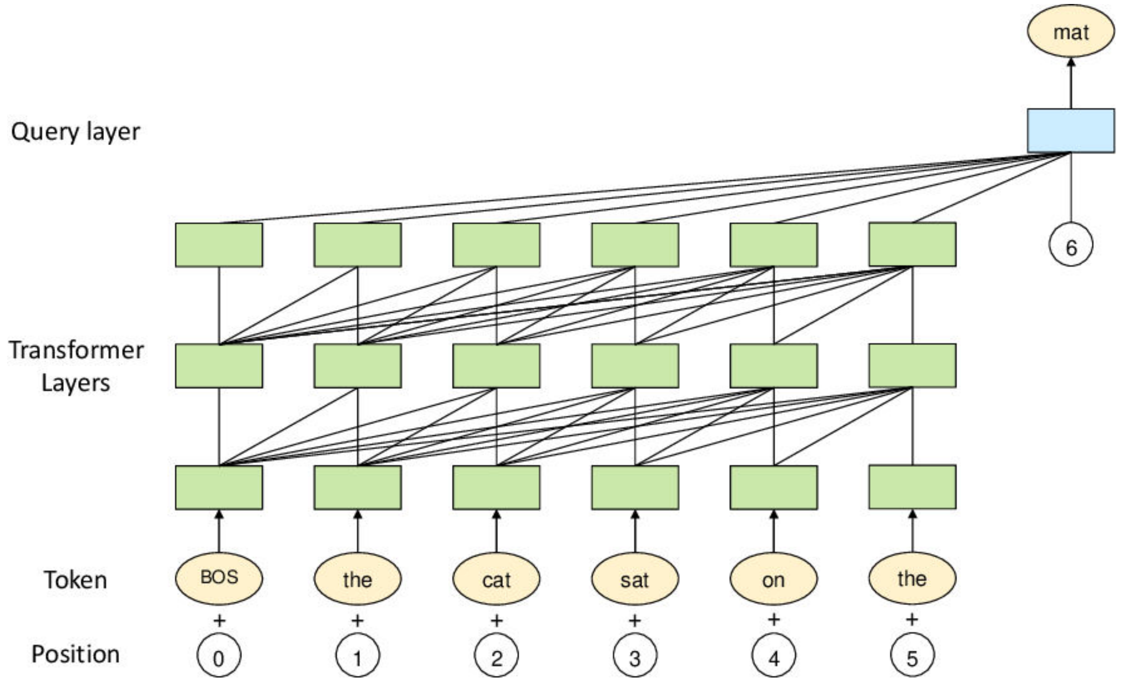
Intuition: Word Representations

- Matrix W embeds words into a common vector space
- Matrix V maps representations into words
- The columns of W correspond to the embeddings of each word
- These can be semantic



Transformer Language Models

- Modern language models use huge transformers
- Much more expressive that the simple linear model
- Can look at huge context sizes (next word given last 4000 words)
- Take word order into account

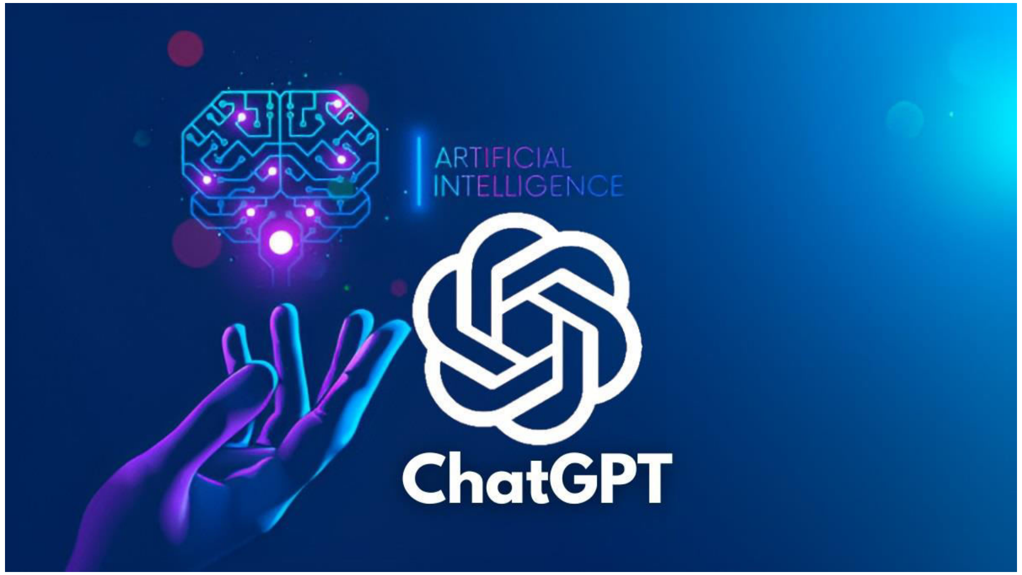


GPT3 (OpenAI)

- Similar to what is used in ChatGPT
- Context: 2048 tokens
- Parameter number: 175B
- Transformer block number: 96
- Training data: 0.5 Tn words crawled from the internet

Are LMs the key to Sentient Machines?

- ”Once LMs can predict all sentences, we can ask them virtually any question and get the correct answer, making them practically sentient” – what do you think?



Conditional LMs: Machine Translation

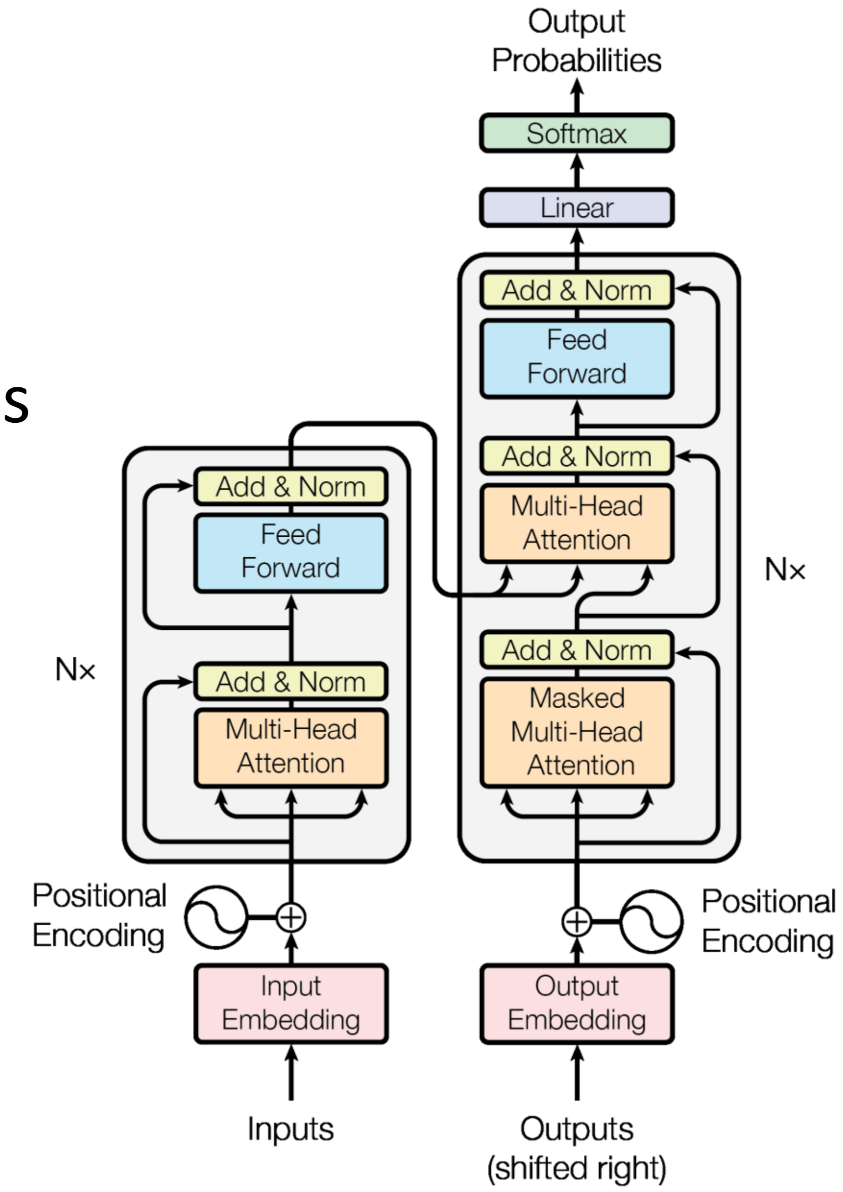
- Predict next word given previous words + source language sentence

$$Q(X|c) = \prod_{i=1}^{N} P(x_i|x_1, x_2..x_{i-1}, c)$$

- About the same solution as before
- The paper that started transformers:
  - "Attention is all you need", Google 2017

Transformer Implementation

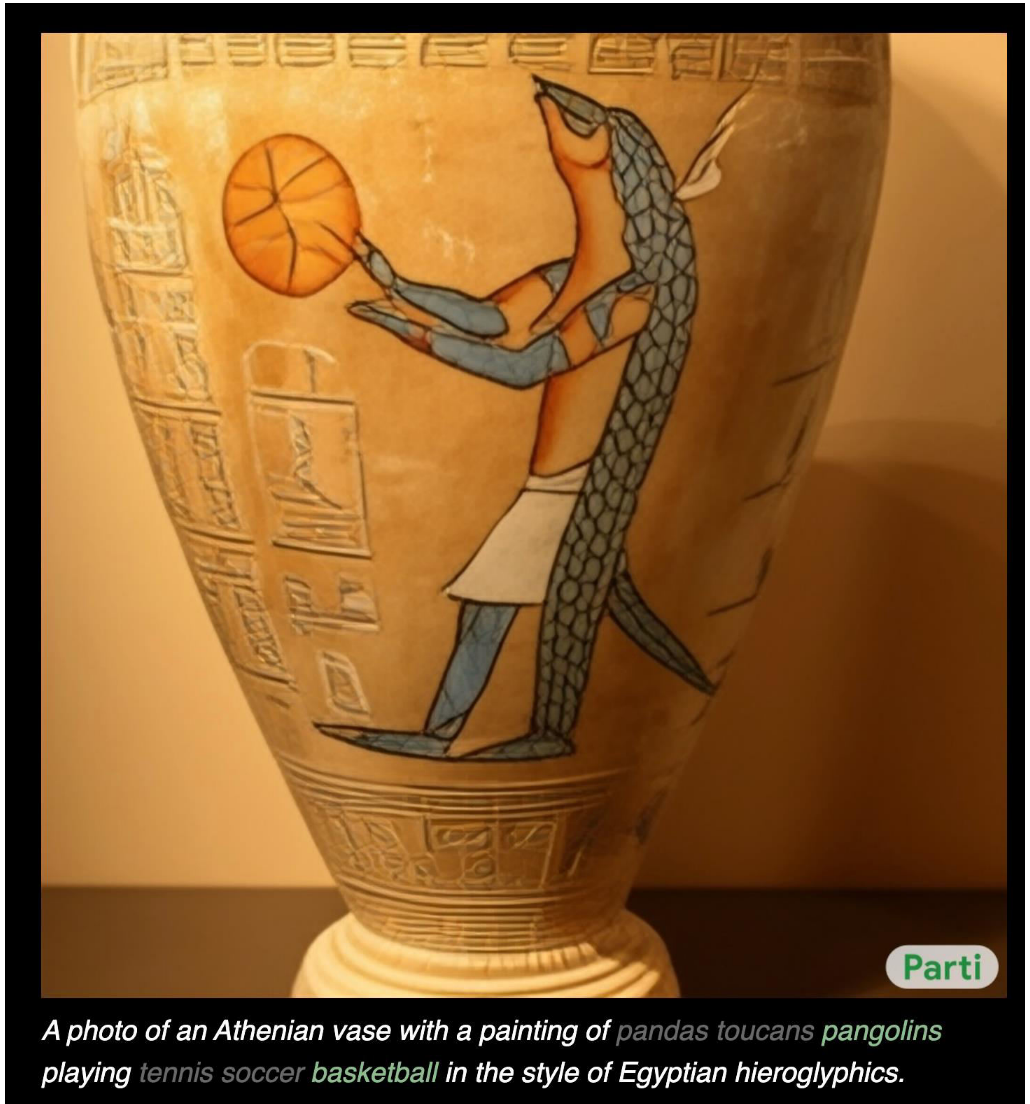
- Transformer encoder of source sentence
- Transformer decoder over encoding + previous words in target sentence
- Why not a decoder over source sentence + previous target sentence words?



Applying the Same Idea to Continuous Data

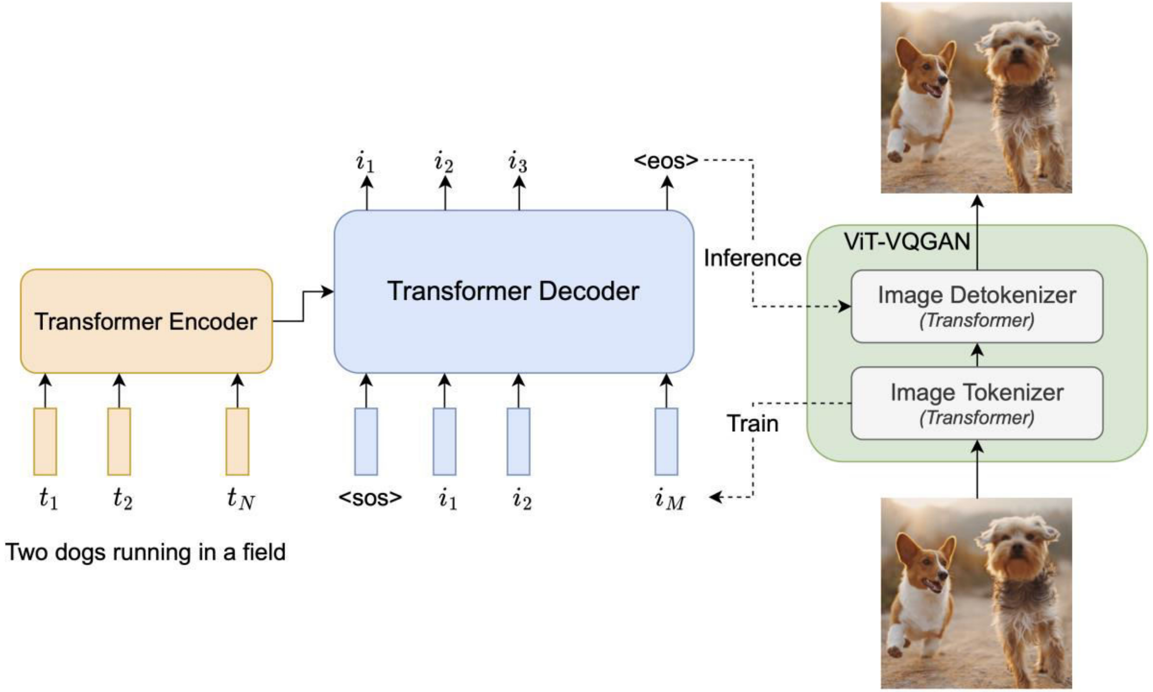
- LM presented so far apply to discrete multivariate data
- How can we apply this to continuous data? E.g. images
- Idea: quantization
  - Learn encoder that maps image to a small number of discrete variables
  - Need to also have a decoder that maps discrete tokens back to image
  - Then everything else applies as before

Parti: LM for Text-to-Image Translation



A photo of an Athenian vase with a painting of pandas toucans pangolins playing tennis soccer basketball in the style of Egyptian hieroglyphics.

Parti (Google)



Effect of Number of Parameters

- Current models have poor sample and compute complexity

A portrait photo of a kangaroo wearing an orange hoodie and blue sunglasses standing on the grass in front of the Sydney Opera House holding a sign on the chest that says Welcome Friends!

Code Example

- https://github.com/karpathy/minGPT

Neural Scaling Laws

- Imagine you are a manager in a large company developing LMs
- You want: best LM in the world
- Have resources but they are finite
- Where would you invest them?
  - Architecture and optimization research?
  - More data?
  - More compute?
  - More memory?

Power Law

- Power law: simple way of representating a relation between variable
- Product of powers of the variables
- Scale invariant

$$f(cx) = a(cx)^{-k} = c^{-k} f(x) \propto f(x),$$

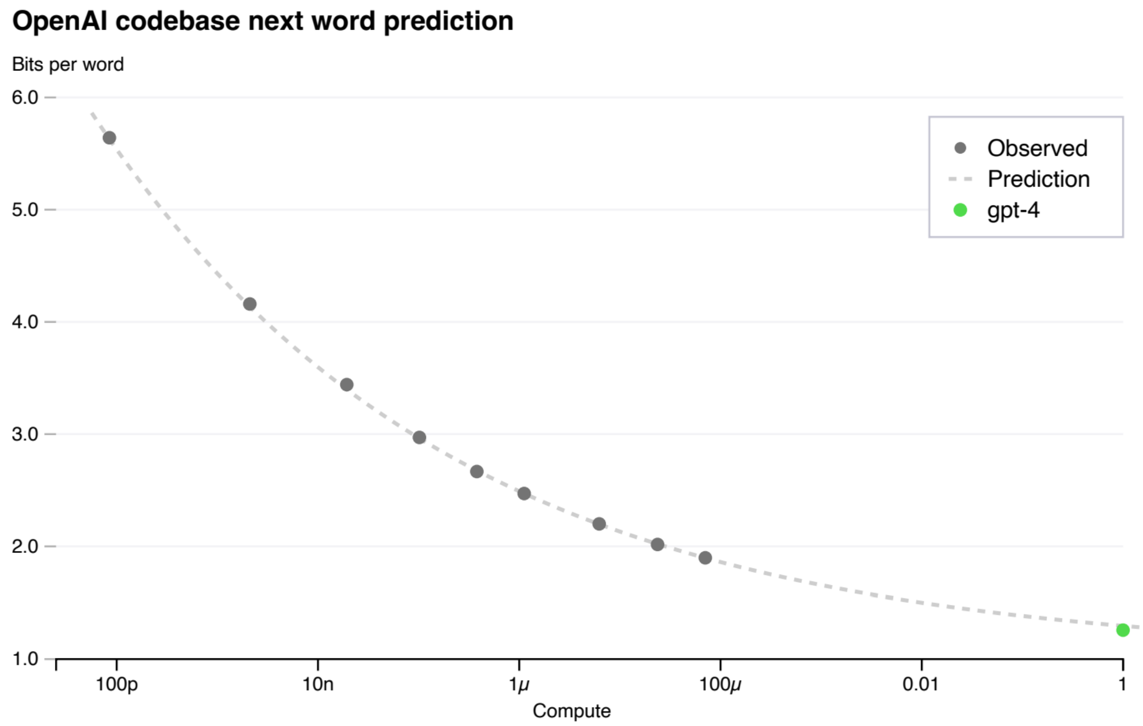
Example: GPT4

- How can we know the accuracy of GPT4 on a small budget?
- Idea: train a set of smaller models, estimate the power law

$$L(C) = aC^b + c,$$

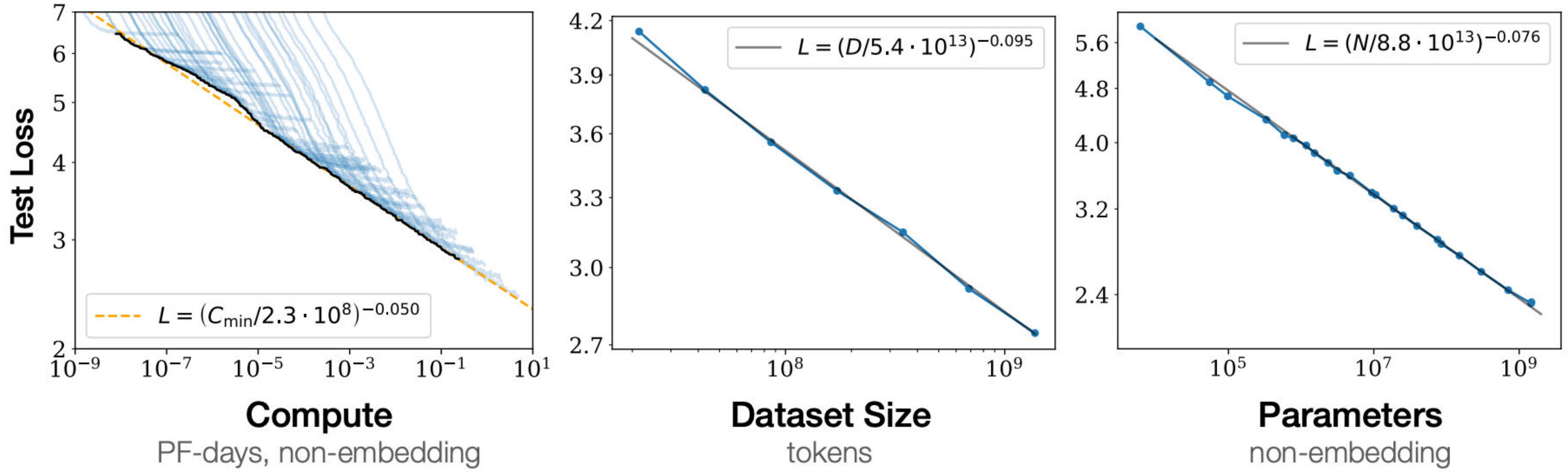
GPT4 Results

- Power law predicted true accuracy very accurately
- Prediction extrpolating 10000X



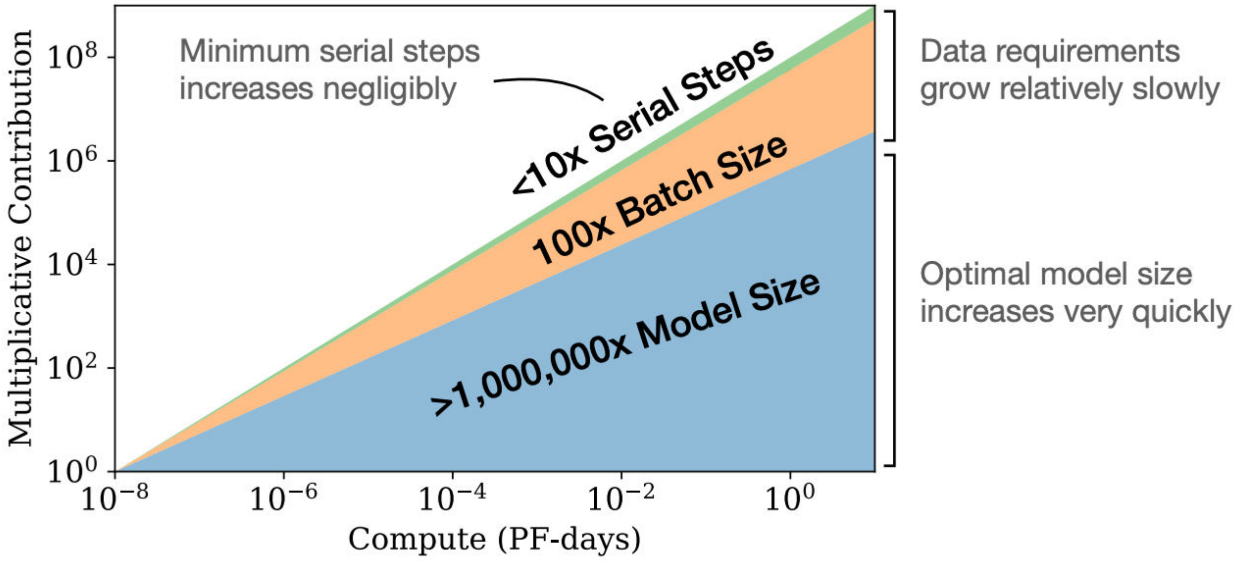
Other Findings in Previous Papers

- Only total compute matters – doesn't matter if it is in depth or width
- Larger models require fewer training step to reach given loss



What Should You Invest In?

- Given a compute budget:
  - Scale model a lot
  - Increase dataset size a little
  - Moderately increase batch



Conclusion

- Language models are a very simple but effective density estimator
- Can be conditional or unconditional
- Given massive compute, can achieve amazing resultsNot suitable for all types of data
- Scaling laws predict their performance very well