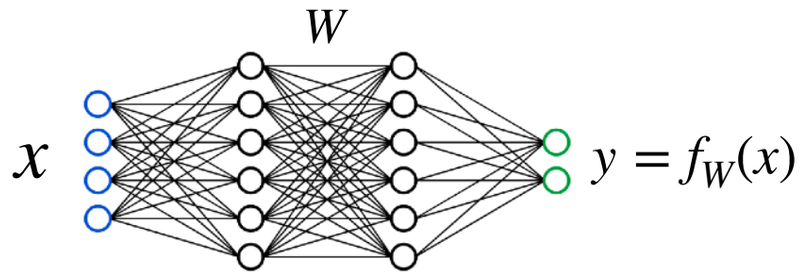
Revisiting neural network approximation theory in the age of generative AI

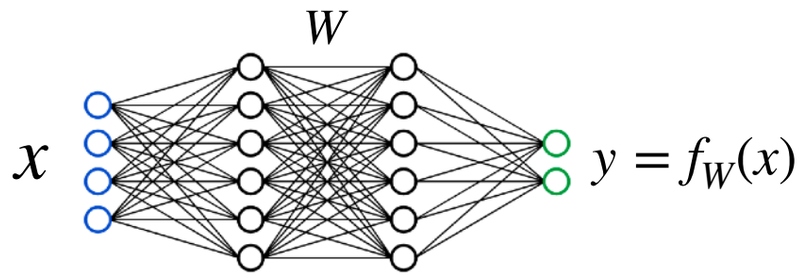
—— From function to algorithm approximation

Song Mei
UC Berkeley

Based on joint work with
Yu Bai (Salesforce Research), Fan Chen (Peking U -> MIT PhD), Licong Lin,
Tianyu Guo (UC Berkeley), Yuchen Wu (Stanford -> Upenn postdoc), Huan Wang,
Caiming Xiong (Salesforce Research)



Traditional view:
Neural networks are function approximators



Traditional view: Neural networks are function approximators

Textbook results:

- 1-hidden layer NNs are universal function approximators.
- NNs efficiently represent “smooth” or “low dimensional” functions.
- For high-dimensional functions, “curse of dimensionality” in worst case.

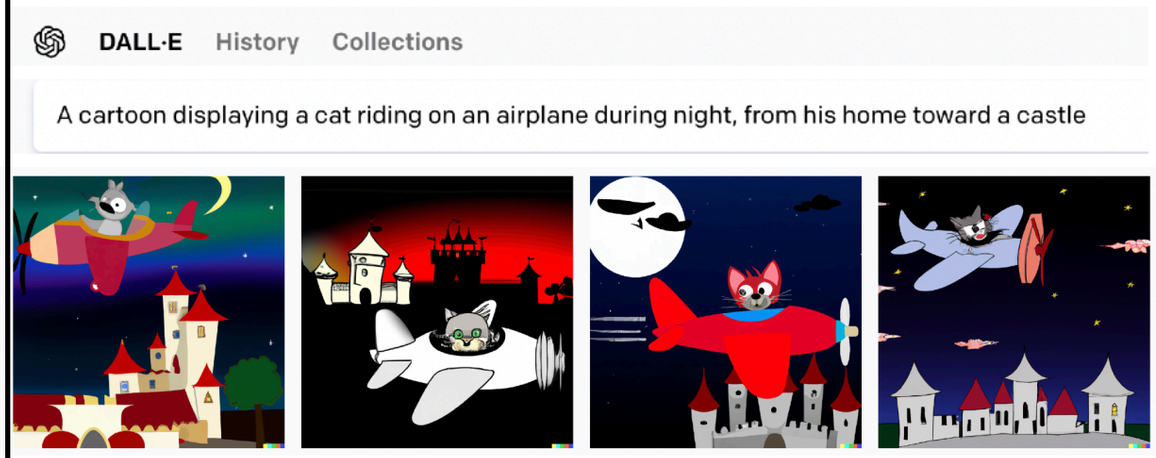
[Hornik, 1991], [Barron, 1993], [Bach, 2017]

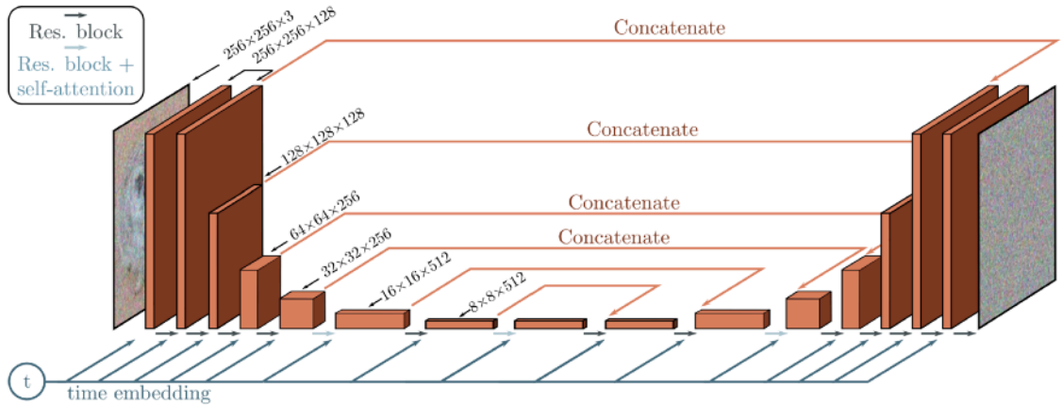
CoD: Require $\#\mathbf{neuron} \sim 1/\varepsilon^{O(d/s)}$ to achieve ε accuracy

s : smoothness of target function in d dimension

Diffusion model

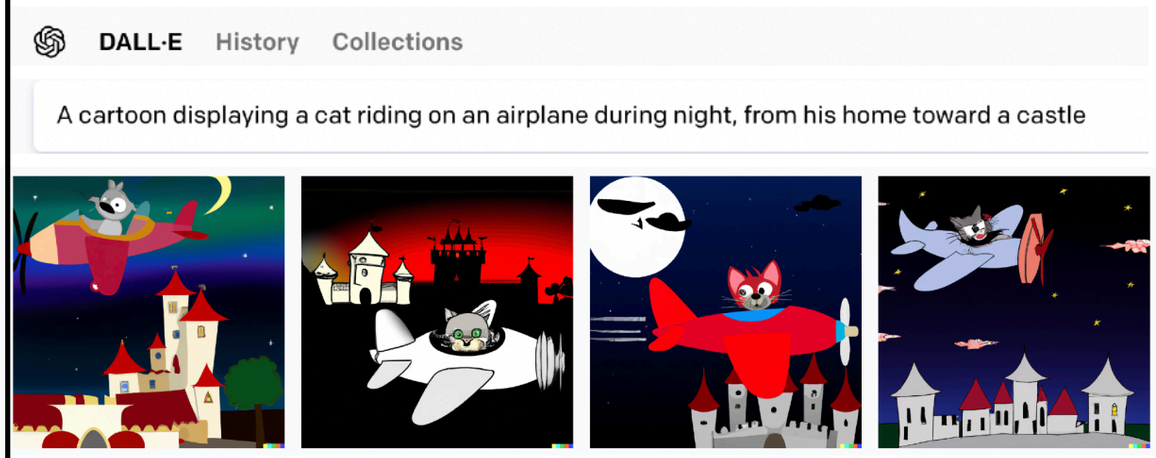
high dimensional input-output

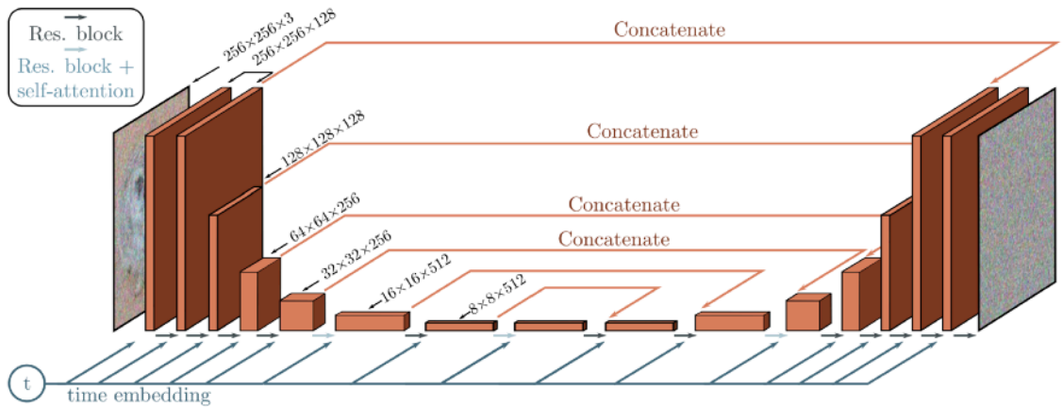




Diffusion model

high dimensional input-output



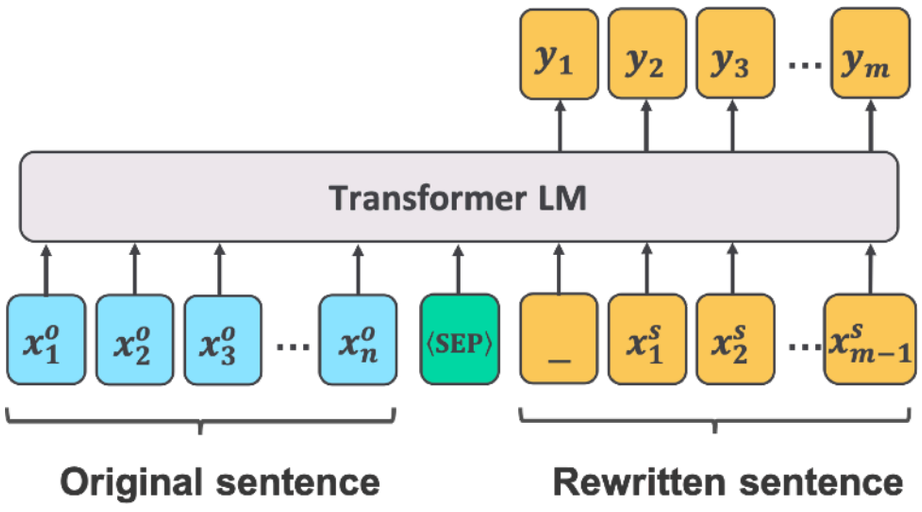


Language model

high dimensional input with varying dimensions

How long does it take from SFO to ORD?

The flight time between San Francisco International Airport (SFO) and Chicago O'Hare International Airport (ORD) varies depending on factors like wind speed and direction, the specific flight path chosen, and the type of aircraft used. Generally, non-stop flights from SFO to ORD take approximately 4 to 4.5 hours. However, actual flight times can be shorter or longer, so it's always best to check with the airline for the most accurate and current information.



Neural networks are ~~function~~ algorithm approximators

Neural networks are ~~function~~ algorithm approximators

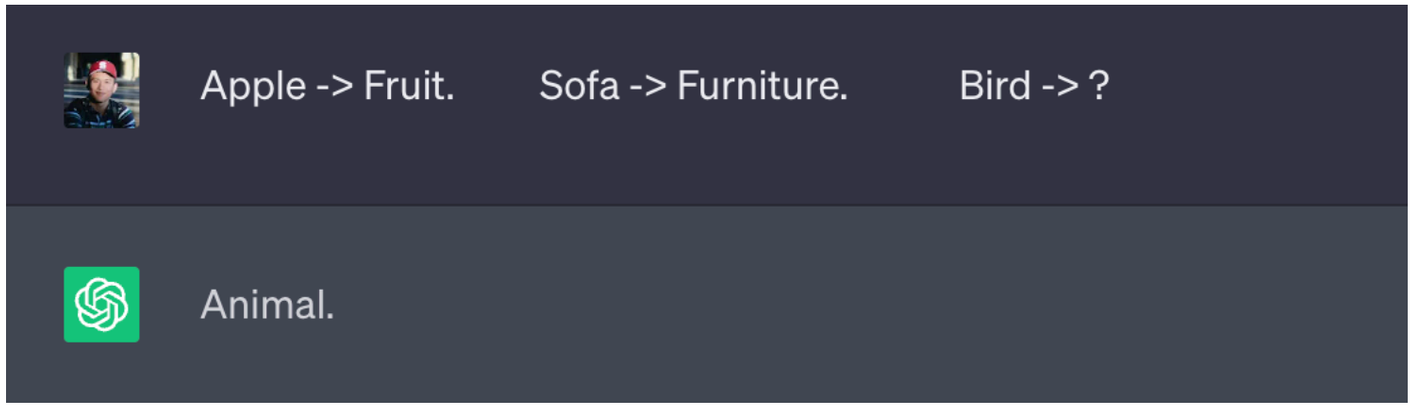
Today:

1. Examples: Language, Diffusion.
2. Discussion: Function vs Algorithm. Benefits of the algorithm approximation perspective.

Example: Language models

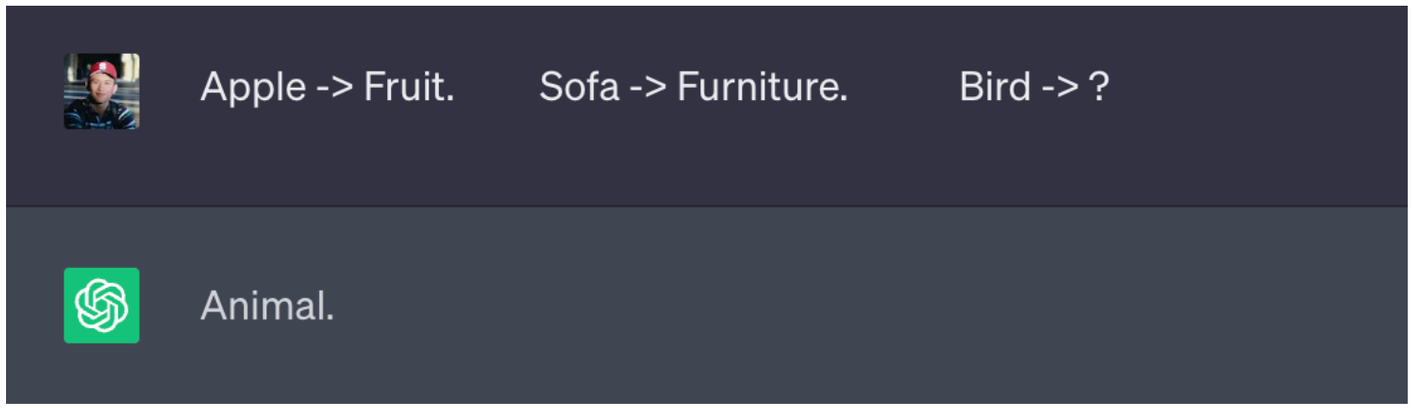
Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).
Radford, Alec, et al. "Language models are unsupervised multitask learners." OpenAI blog 1.8 (2019): 9.
Brown, Tom, et al. "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877-1901.
Chowdhery, Aakanksha, et al. "Palm: Scaling language modeling with pathways." arXiv preprint arXiv:2204.02311 (2022).

The in-context learning (ICL) capability



Brown, Tom, et al. "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877-1901.

The in-context learning (ICL) capability



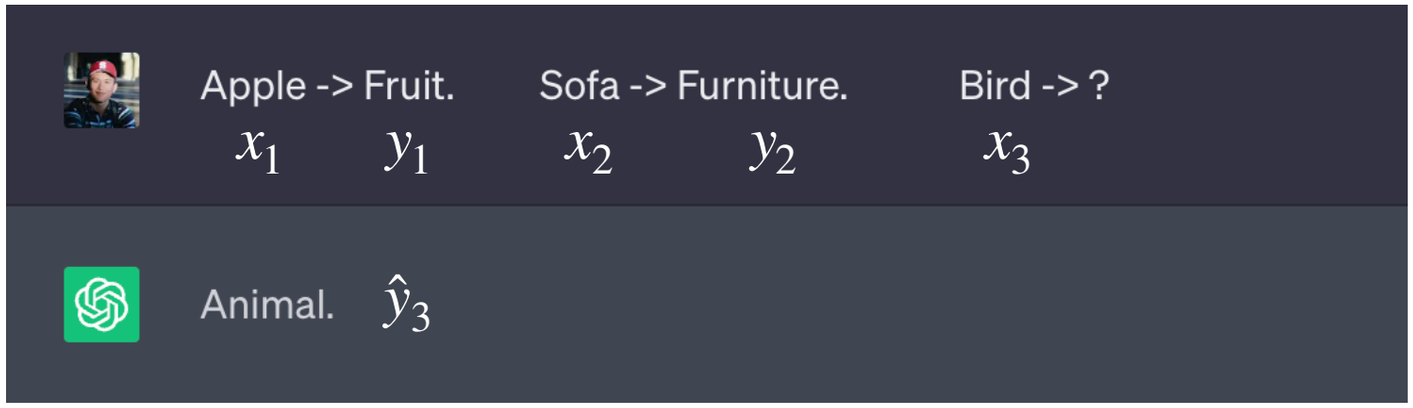
A transformer $y = \mathrm{TF}_{\hat{\theta}}(H)$ meta-trained with a huge meta-dataset.

In-context learning capability

- Let $\{(x_i, y_i)\}_{i \in [N+1]} \sim \mathbb{P}$, where $\mathbb{P}$ is unknown to TF.
- Take $H = [x_1, y_1, x_2, y_2, \ldots, x_N, y_N, x_{N+1}]$ as the context input.
- Output a good estimates $\hat{y}_{N+1} = \mathrm{TF}_{\hat{\theta}}(H) \approx y_{N+1}$.

Brown, Tom, et al. "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877-1901.

The in-context learning (ICL) capability



A transformer $y = \mathrm{TF}_{\hat{\theta}}(H)$ meta-trained with a huge meta-dataset.

In-context learning capability

- Let $\{(x_i, y_i)\}_{i \in [N+1]} \sim \mathbb{P}$, where $\mathbb{P}$ is unknown to TF.
- Take $H = [x_1, y_1, x_2, y_2, \ldots, x_N, y_N, x_{N+1}]$ as the context input.
- Output a good estimates $\hat{y}_{N+1} = \mathrm{TF}_{\hat{\theta}}(H) \approx y_{N+1}$.

Brown, Tom, et al. "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877-1901.

ICL experiment on synthetic datasets

A Task : $\{(x_i, y_i)\}_{i\in[N]}, \quad \beta \sim \mathcal{N}(0, I_d/d),$

$x_i \sim \mathcal{N}(0, I_d), \quad y_i = x_i^\top \beta + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2)$

[Garg, Tsipras, Liang, Valiant. 2022], [Akyürek, Schuurmans, Andreas, Ma, Zhou 2022], [von Oswald, Niklasson, Randazzo, Sacramento, Mordvintsev, Zhmoginov, Vladymyrov, 2022]

ICL experiment on synthetic datasets

A Task : $\{(x_i, y_i)\}_{i\in[N]}, \quad \beta \sim \mathcal{N}(0, I_d/d),$

$x_i \sim \mathcal{N}(0, I_d), \quad y_i = x_i^\top \beta + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2)$

- A dataset of (size N) is a meta-datapoint: $H = [x_1, y_1, x_2, y_2, \ldots, x_N, y_N]$.

[Garg, Tsipras, Liang, Valiant. 2022], [Akyürek, Schuurmans, Andreas, Ma, Zhou 2022], [von Oswald, Niklasson, Randazzo, Sacramento, Mordvintsev, Zhmoginov, Vladymyrov, 2022]

ICL experiment on synthetic datasets

A Task : $\{(x_i, y_i)\}_{i\in[N]}, \quad \beta \sim \mathcal{N}(0, I_d/d),$

$$x_i \sim \mathcal{N}(0, I_d), \quad y_i = x_i^\top \beta + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2)$$

- A dataset of (size N) is a meta-datapoint: $H = [x_1, y_1, x_2, y_2, \ldots, x_N, y_N]$.
- A meta-dataset (size n): $\{H^{(j)} = [x_1^{(j)}, y_1^{(j)}, x_2^{(j)}, y_2^{(j)}, \ldots, x_N^{(j)}, y_N^{(j)}]\}_{j\in[n]}$.

[Garg, Tsipras, Liang, Valiant. 2022], [Akyürek, Schuurmans, Andreas, Ma, Zhou 2022], [von Oswald, Niklasson, Randazzo, Sacramento, Mordvintsev, Zhmoginov, Vladymyrov, 2022]

ICL experiment on synthetic datasets

A Task : $\{(x_i, y_i)\}_{i\in[N]}, \quad \beta \sim \mathcal{N}(0, I_d/d),$

$$x_i \sim \mathcal{N}(0, I_d), \quad y_i = x_i^\top \beta + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2)$$

- A dataset of (size N) is a meta-datapoint: $H = [x_1, y_1, x_2, y_2, \ldots, x_N, y_N]$.
- A meta-dataset (size n): $\{H^{(j)} = [x_1^{(j)}, y_1^{(j)}, x_2^{(j)}, y_2^{(j)}, \ldots, x_N^{(j)}, y_N^{(j)}]\}_{j\in[n]}$.
- Train the GPT2 model using $\{H^{(j)}\}_{j\in[n]}$ (a smaller version of ChatGPT).

$$\min_\theta \sum_{j=1}^{n} \sum_{i=1}^{N} \left(y_i^{(j)} - \mathrm{TF}_\theta(x_{1:i}^{(j)}, y_{1:i-1}^{(j)}) \right)^2$$

[Garg, Tsipras, Liang, Valiant. 2022], [Akyürek, Schuurmans, Andreas, Ma, Zhou 2022], [von Oswald, Niklasson, Randazzo, Sacramento, Mordvintsev, Zhmoginov, Vladymyrov , 2022]

ICL experiment on synthetic datasets

A Task : $\{(x_i, y_i)\}_{i\in[N]}, \quad \beta \sim \mathcal{N}(0, I_d/d),$

$$x_i \sim \mathcal{N}(0, I_d), \quad y_i = x_i^\top \beta + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2)$$

- A dataset of (size N) is a meta-datapoint: $H = [x_1, y_1, x_2, y_2, \ldots, x_N, y_N]$.
- A meta-dataset (size n): $\{H^{(j)} = [x_1^{(j)}, y_1^{(j)}, x_2^{(j)}, y_2^{(j)}, \ldots, x_N^{(j)}, y_N^{(j)}]\}_{j\in[n]}$.
- Train the GPT2 model using $\{H^{(j)}\}_{j\in[n]}$ (a smaller version of ChatGPT).

$$\min_\theta \sum_{j=1}^{n} \sum_{i=1}^{N} \left(y_i^{(j)} - \mathrm{TF}_\theta(x_{1:i}^{(j)}, y_{1:i-1}^{(j)}) \right)^2$$

- Evaluate the test performance of GPT2 on a new independent task.

[Garg, Tsipras, Liang, Valiant. 2022], [Akyürek, Schuurmans, Andreas, Ma, Zhou 2022], [von Oswald, Niklasson, Randazzo, Sacramento, Mordvintsev, Zhmoginov, Vladymyrov , 2022]

ICL experiment on synthetic datasets

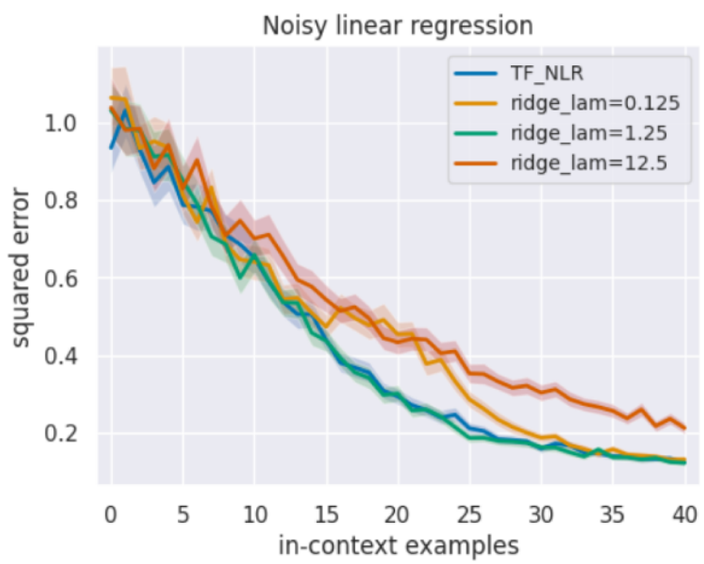
A Task : $\{(x_i, y_i)\}_{i\in[N]}, \quad \beta \sim \mathcal{N}(0, I_d/d),$

$$x_i \sim \mathcal{N}(0, I_d), \quad y_i = x_i^\top \beta + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2)$$

- A dataset of (size N) is a meta-datapoint: $H = [x_1, y_1, x_2, y_2, \ldots, x_N, y_N]$.
- A meta-dataset (size n): $\{H^{(j)} = [x_1^{(j)}, y_1^{(j)}, x_2^{(j)}, y_2^{(j)}, \ldots, x_N^{(j)}, y_N^{(j)}]\}_{j\in[n]}$.
- Train the GPT2 model using $\{H^{(j)}\}_{j\in[n]}$ (a smaller version of ChatGPT).

$$\min_\theta \sum_{j=1}^{n} \sum_{i=1}^{N} \left(y_i^{(j)} - \mathrm{TF}_\theta(x_{1:i}^{(j)}, y_{1:i-1}^{(j)}) \right)^2$$

- Evaluate the test performance of GPT2 on a new independent task.



Bayes optimal

[Garg, Tsipras, Liang, Valiant. 2022], [Akyürek, Schuurmans, Andreas, Ma, Zhou 2022], [von Oswald, Niklasson, Randazzo, Sacramento, Mordvintsev, Zhmoginov, Vladymyrov , 2022]

ICL experiment on synthetic datasets

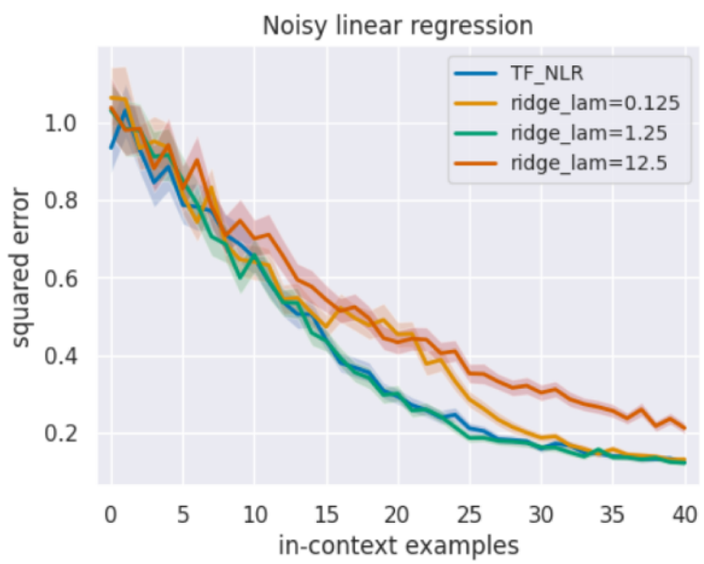
A Task : $\{(x_i, y_i)\}_{i\in[N]}, \quad \beta \sim \mathcal{N}(0, I_d/d),$

$$x_i \sim \mathcal{N}(0, I_d), \quad y_i = x_i^\top \beta + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2)$$

- A dataset of (size N) is a meta-datapoint: $H = [x_1, y_1, x_2, y_2, \ldots, x_N, y_N]$.
- A meta-dataset (size n): $\{H^{(j)} = [x_1^{(j)}, y_1^{(j)}, x_2^{(j)}, y_2^{(j)}, \ldots, x_N^{(j)}, y_N^{(j)}]\}_{j\in[n]}$.
- Train the GPT2 model using $\{H^{(j)}\}_{j\in[n]}$ (a smaller version of ChatGPT).

$$\min_\theta \sum_{j=1}^{n} \sum_{i=1}^{N} \left(y_i^{(j)} - \mathrm{TF}_\theta(x_{1:i}^{(j)}, y_{1:i-1}^{(j)}) \right)^2$$

- Evaluate the test performance of GPT2 on a new independent task.



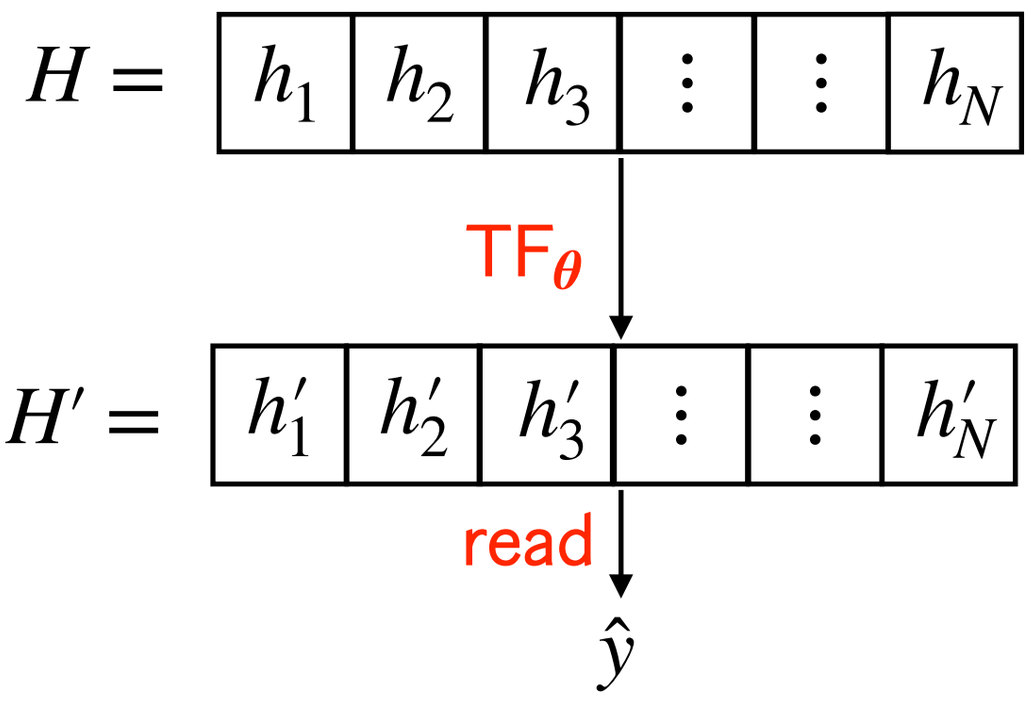
Bayes optimal

[Garg, Tsipras, Liang, Valiant. 2022], [Akyürek, Schuurmans, Andreas, Ma, Zhou 2022], [von Oswald, Niklasson, Randazzo, Sacramento, Mordvintsev, Zhmoginov, Vladymyrov , 2022]

Why can GPT (transformers) perform in-context learning (ICL)?

Transformers

- A transformer* is a sequence-to-sequence neural network $\mathrm{TF}_{\boldsymbol{\theta}} : \mathbb{R}^{D\times N} \to \mathbb{R}^{D\times N}$.
- Input sequence: $H = [h_1, h_2, \ldots, h_N] \in \mathbb{R}^{D\times N}$; each $h_i \in \mathbb{R}^D$ is called a token.



- Transformer output: $H' = \mathrm{TF}_{\boldsymbol{\theta}}(H) = [h'_1, \cdots, h'_N] \in \mathbb{R}^{D\times N}$.
- Final output: $\hat{y} = \mathrm{read}(\mathrm{TF}_{\boldsymbol{\theta}}(H))$.

** Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017). We focus on the encoder architecture.*

Transformers: Feedforward layer

- A transformer is an alternating composition of FF layers and Attention layers

  $\mathrm{TF}_{\boldsymbol{\theta}}(\,\cdot\,) = \mathrm{FF}_{W^{(L)}} \circ \mathrm{ATTN}_{A^{(L)}} \circ \cdots \circ \mathrm{FF}_{W^{(1)}} \circ \mathrm{ATTN}_{A^{(1)}}$

  $\boldsymbol{\theta} = (W^{(L)}, A^{(L)}, \ldots, W^{(1)}, A^{(1)})$

** Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).*

Transformers: Feedforward layer

- A transformer is an alternating composition of FF layers and Attention layers

$$\mathrm{TF}_{\boldsymbol{\theta}}(\,\cdot\,) = \mathrm{FF}_{W^{(L)}} \circ \mathrm{ATTN}_{A^{(L)}} \circ \cdots \circ \mathrm{FF}_{W^{(1)}} \circ \mathrm{ATTN}_{A^{(1)}}$$

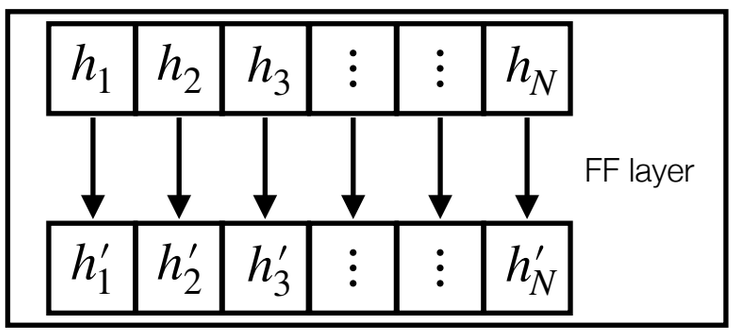
$$\boldsymbol{\theta} = (W^{(L)}, A^{(L)}, \ldots, W^{(1)}, A^{(1)})$$

- Feedforward layer: $H' = \mathrm{FF}_W(H) : \mathbb{R}^{D\times N} \to \mathbb{R}^{D\times N}$,

$$W = (W_1, W_2), \qquad W_1, W_2^\top \in \mathbb{R}^{D'\times D}$$

- Token-wise function:

$$h_i' = h_i + W_2 \cdot \sigma(W_1 h_i)$$



** Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).*

Transformers: Attention layer

- A transformer is an alternating composition of FF layers and Attention layers

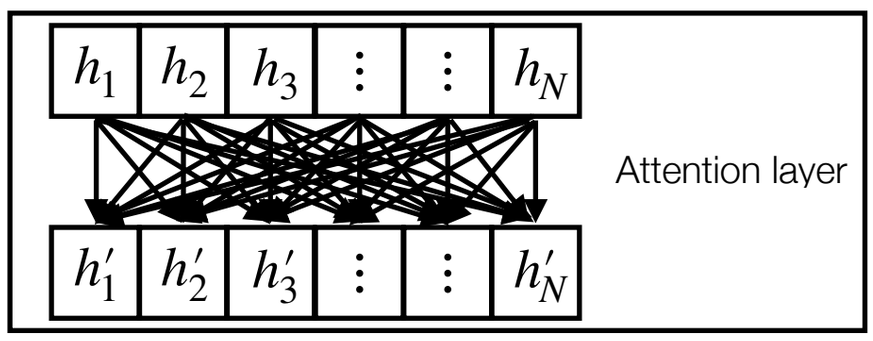
$$\mathrm{TF}_{\boldsymbol{\theta}}(\,\cdot\,) = \mathrm{FF}_{W^{(L)}} \circ \mathrm{ATTN}_{A^{(L)}} \circ \cdots \circ \mathrm{FF}_{W^{(1)}} \circ \mathrm{ATTN}_{A^{(1)}}$$

$$\boldsymbol{\theta} = (W^{(L)}, A^{(L)}, \ldots, W^{(1)}, A^{(1)})$$

- Attention layer: $H' = \mathrm{ATTN}_A(H) : \mathbb{R}^{D\times N} \to \mathbb{R}^{D\times N}$

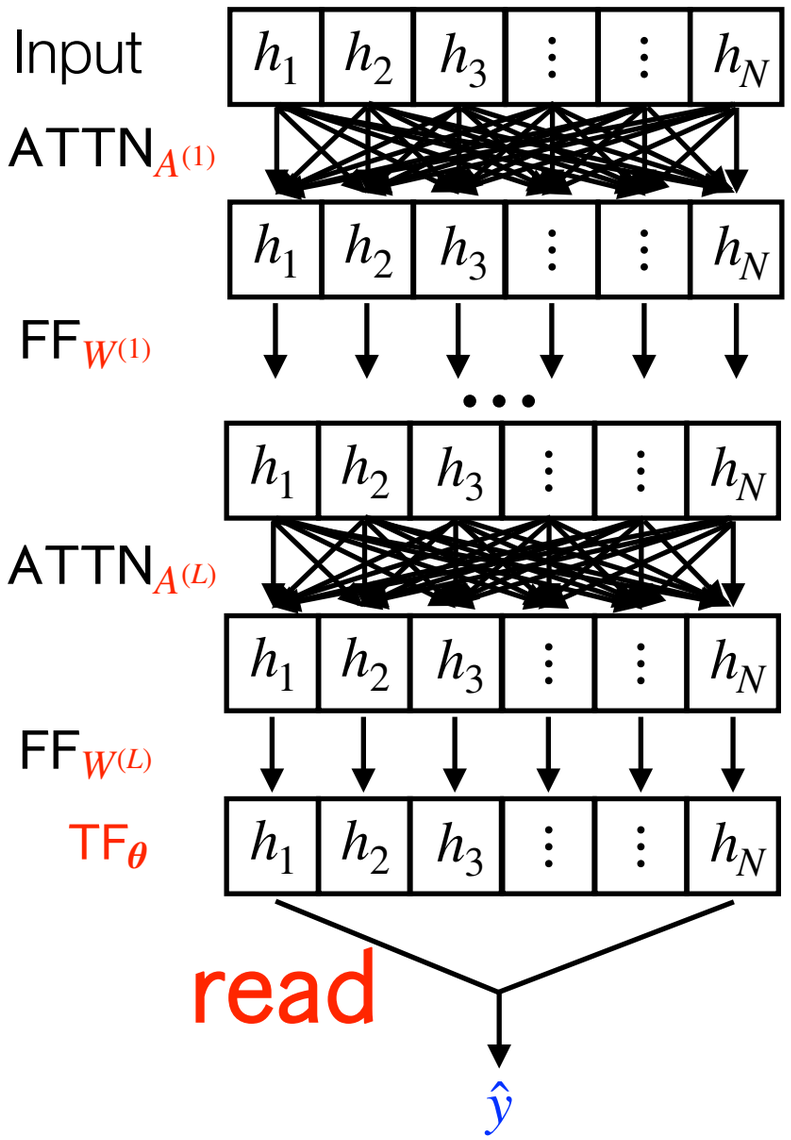
$$A = (Q_m, K_m, V_m)_{m\in[M]}, \qquad Q_m, K_m, V_m \in \mathbb{R}^{D\times D}$$

- Multi-head attention layer: $h_i' = h_i + \sum_{m=1}^{M} \frac{1}{N} \sum_{j=1}^{N} \sigma(\langle Q_m h_i, K_m h_j \rangle) \cdot V_m h_j$



** Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).*

Transformers: the whole structure



Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).
Radford, Alec, et al. "Language models are unsupervised multitask learners." OpenAI blog 1.8 (2019): 9.
Brown, Tom, et al. "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877-1901.
Chowdhery, Aakanksha, et al. "Palm: Scaling language modeling with pathways." arXiv preprint arXiv:2204.02311 (2022).

ICL by transformers

$$\{(x_i, y_i)\}_{i\in[N+1]} \quad \text{iid}, \quad y_i = f(\langle x_i, w_* \rangle) + \varepsilon_i$$

- ICL by transformers:

Input format: $H =$

x_1	x_2	x_3	$\cdots$	$\cdots$	x_N	x_{N+1}
y_1	y_2	y_3	$\cdots$	$\cdots$	y_N	0
0	0	0	$\cdots$	$\cdots$	0	0

→ Reserved slot for output

→ Scratch pad

TF_θ

$\text{TF}_\theta(H) =$

x_1	x_2	x_3	$\cdots$	$\cdots$	x_N	x_{N+1}
y_1	y_2	y_3	$\cdots$	$\cdots$	y_N	$\hat{y}_{N+1}$
$*$	$*$	$*$	$\cdots$	$\cdots$	$*$	$*$

read → Output: $\hat{y}_{N+1}$

Hope: $\hat{y}_{N+1}$ is a good estimator of y_{N+1}

ICL by transformers

$$\{(x_i, y_i)\}_{i\in[N+1]} \quad \text{iid}, \qquad y_i = f(\langle x_i, w_* \rangle) + \varepsilon_i$$

- ICL by transformers:

Input format: $H =$

x_1	x_2	x_3	$\vdots$	$\vdots$	x_N	x_{N+1}
y_1	y_2	y_3	$\vdots$	$\vdots$	y_N	0
0	0	0	$\vdots$	$\vdots$	0	0

→ Reserved slot for output

→ Scratch pad

TF_θ

$\text{TF}_\theta(H) =$

x_1	x_2	x_3	$\vdots$	$\vdots$	x_N	x_{N+1}
y_1	y_2	y_3	$\vdots$	$\vdots$	y_N	$\hat{y}_{N+1}$
$*$	$*$	$*$	$\vdots$	$\vdots$	$*$	$*$

read → Output: $\hat{y}_{N+1}$

Hope: $\hat{y}_{N+1}$ is a good estimator of y_{N+1}

- Explanation: transformers can approximate GD on $\widehat{R}_N(w) = \frac{1}{N}\sum_{i=1}^N \ell(x_i^\top w, y_i)$

$$w^{t+1} = w^t - \frac{\eta}{N}\sum_{i=1}^N \partial_1 \ell(x_i^\top w^t, y_i) \times x_i$$

[Akyürek, Schuurmans, Andreas, Ma, Zhou 2022], [von Oswald, Niklasson, Randazzo, Sacramento, Mordvintsev, Zhmoginov, Vladymyrov , 2022] have rough ideas of this kind.

One-step gradient descent by an attention layer

Attention

$$h_{N+1}^{t+1} = h_{N+1}^{t} + \frac{1}{N}\sum_{i=1}^{N}\sum_{m=1}^{M} \sigma\left(\left\langle Q_m h_{N+1}^{t} , K_m h_i^t \right\rangle \right) \times V_m h_i^t$$

One-step gradient descent by an attention layer

Attention

$$h_{N+1}^{t+1} = h_{N+1}^{t} + \frac{1}{N}\sum_{i=1}^{N}\sum_{m=1}^{M} \sigma\left(\left\langle Q_m h_{N+1}^{t} , K_m h_i^t \right\rangle \right) \times V_m h_i^t$$

Gradient descent

$$w^{t+1} = w^t - \frac{\eta}{N}\sum_{i=1}^{N} \partial_1 \ell\left(\left\langle x_i , w^t \right\rangle , y_i \right) \times x_i$$

One-step gradient descent by an attention layer

Attention

$$h_{N+1}^{t+1} = h_{N+1}^{t} + \frac{1}{N}\sum_{i=1}^{N}\sum_{m=1}^{M} \sigma\left(\left\langle Q_m h_{N+1}^{t} , K_m h_i^t \right\rangle \right) \times V_m h_i^t$$

Weight

$$h_i^t = \begin{pmatrix} x_i \\ y_i \\ w^t \end{pmatrix}, \quad Q_m \begin{pmatrix} * \\ * \\ w^t \end{pmatrix} = \begin{matrix} b_m \times \\ c_m \times \end{matrix} \begin{pmatrix} w^t \\ 1 \end{pmatrix}, \quad K_m \begin{pmatrix} x_i \\ y_i \\ w^t \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix}, \quad V_m \begin{pmatrix} x_i \\ y_i \\ w^t \end{pmatrix} = -\eta a_m \times \begin{pmatrix} 0 \\ 0 \\ x_i \end{pmatrix}$$

$$\begin{pmatrix} x_i \\ y_i \\ w^{t+1} \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \\ w^t \end{pmatrix} - \frac{\eta}{N}\sum_{i=1}^{N}\sum_{m=1}^{M} a_m \sigma\left(b_m \langle x_i , w^t \rangle + c_m y_i \right) \times \begin{pmatrix} 0 \\ 0 \\ x_i \end{pmatrix}$$

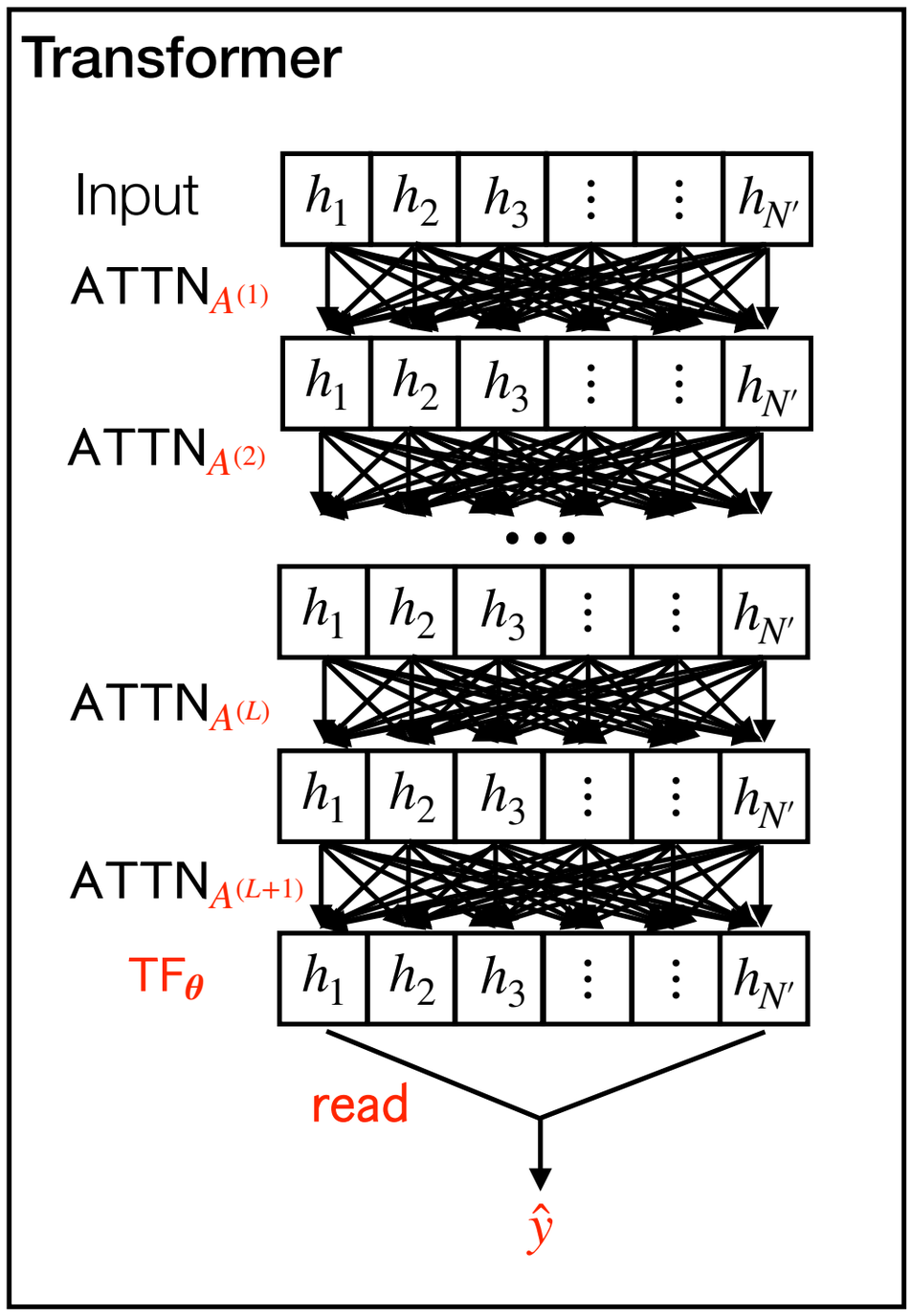
Universal approximation

$$\partial_1 \ell(s,t) \approx \sum_{m=1}^{M} a_m \cdot \sigma(b_m \cdot s + c_m \cdot t)$$

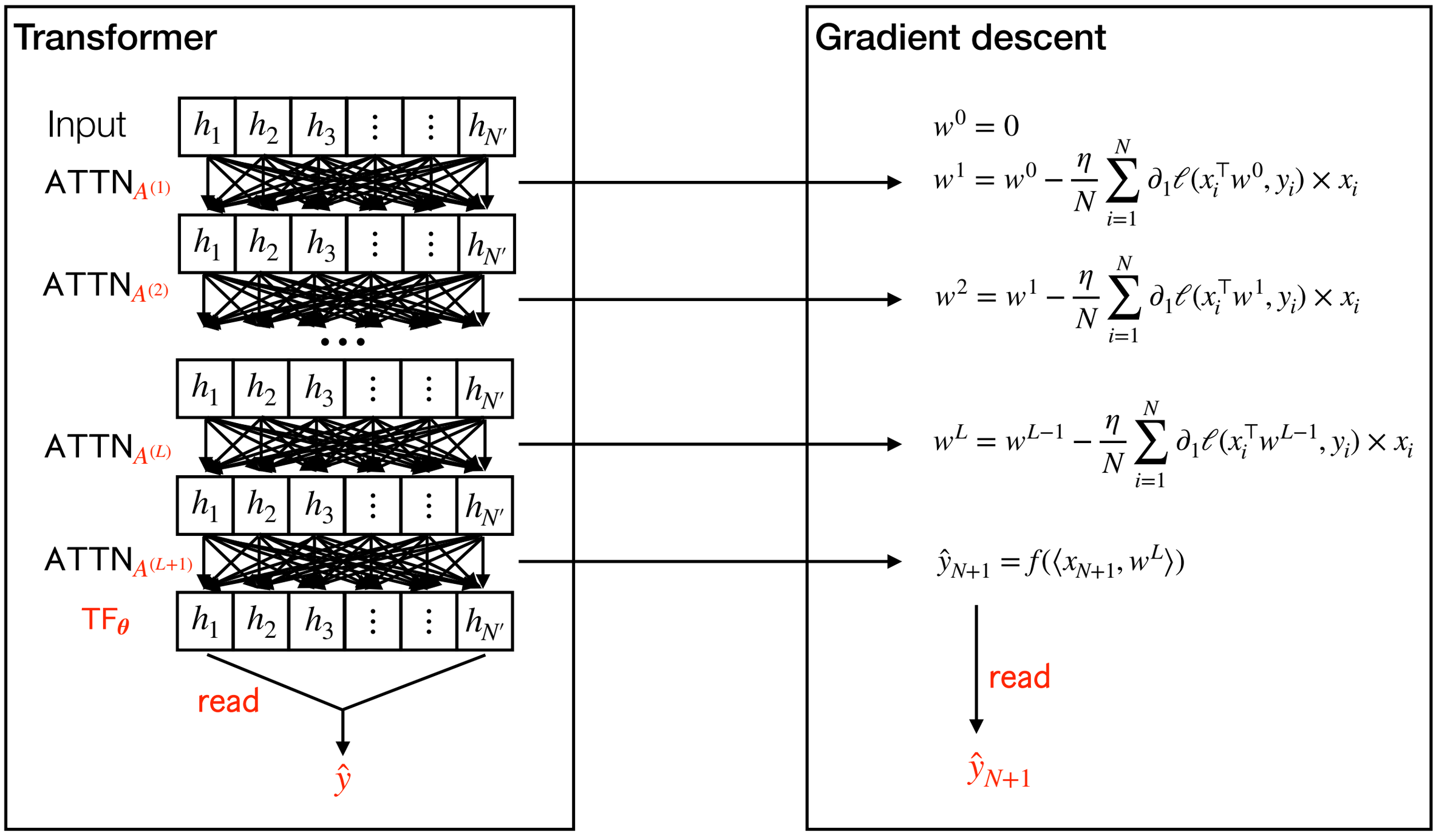
Gradient descent

$$w^{t+1} = w^t - \frac{\eta}{N}\sum_{i=1}^{N} \partial_1 \ell(\langle x_i , w^t \rangle , y_i) \times x_i$$

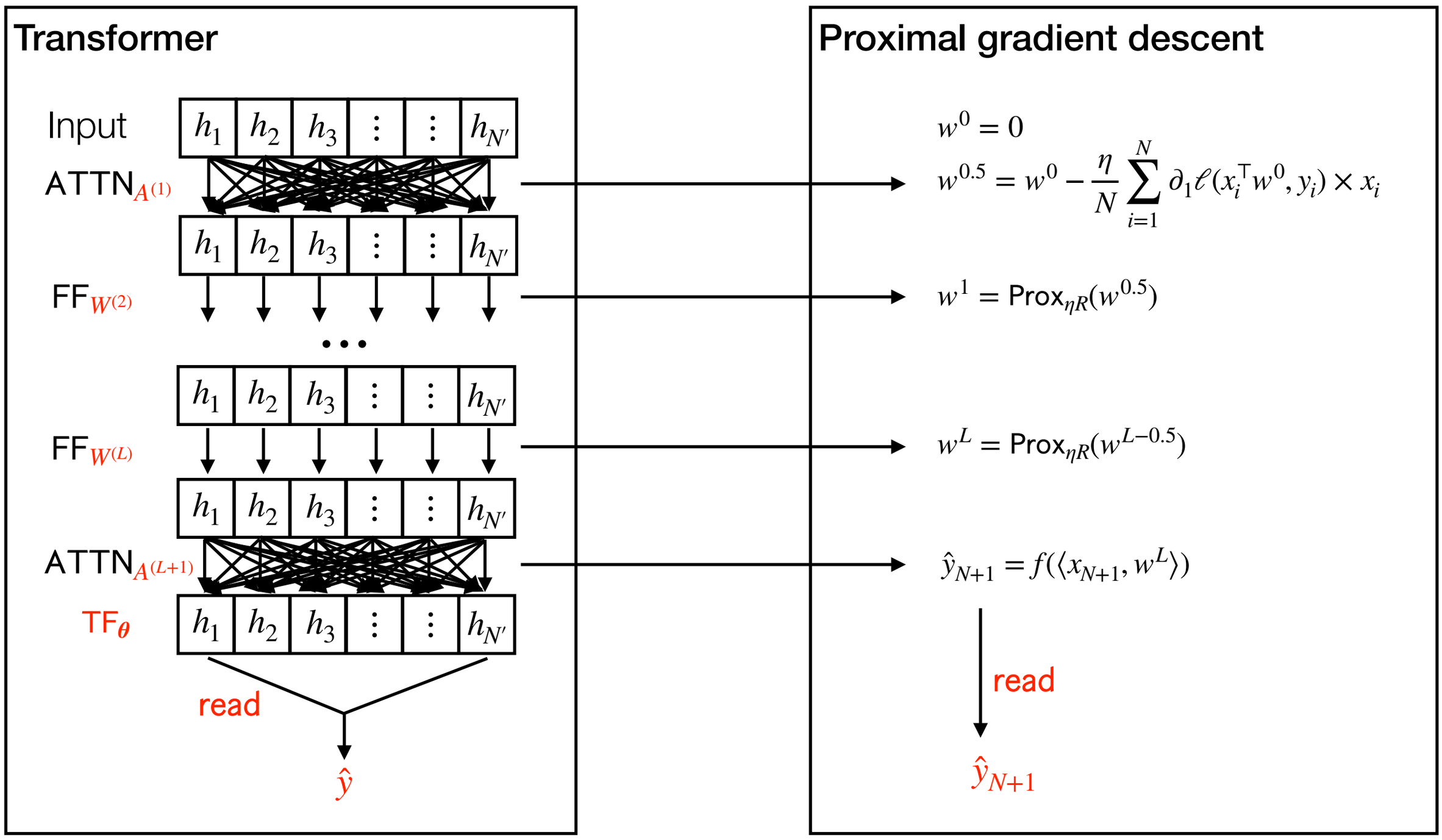
Transformer versus multi-step GD



Transformer versus multi-step GD



Transformer versus PGD



In-context ridge, logistic, LASSO

Parameters for the size of TF:

Embedding dimensions D, number of layers L, number of ATTN heads M,

FF width D', and norm of parameters $|||\theta|||$

Thm: There exists three transformers with $D = D' = \mathcal{O}(d)$, $|||\theta||| = \mathcal{O}(\mathrm{Poly}(\cdot))$

$$\begin{aligned}
\text{Ridge}: &\quad L = \mathcal{O}(\log(N)), &&\quad M = 3,\\
\text{Logistic}: &\quad L = \mathcal{O}(\log(N/d)), &&\quad M = N/d,\\
\text{LASSO}: &\quad L = \mathcal{O}(1 + d/N), &&\quad M = 2,
\end{aligned}$$

that output $\hat{y}_{N+1}$ implementing Ridge, Logistic, LASSO, with

$$\begin{aligned}
\text{Ridge}: &\quad \mathbb{E}[(\hat{y}_{N+1} - y_{N+1})^2] - \sigma^2 \le \mathcal{O}(d/N),\\
\text{Logistic}: &\quad \mathbb{E}[(\hat{y}_{N+1} - \mathbb{E}[y_{N+1}|x_{N+1}])^2] \le \mathcal{O}(d/N),\\
\text{LASSO}: &\quad \mathbb{E}[(\hat{y}_{N+1} - y_{N+1})^2] - \sigma^2 \le \mathcal{O}(s\log d/N)\,.
\end{aligned}$$

[Bai, Chen, Wang, Xiong, **Mei**, 2023]

Such transformers can be found statistically efficiently

Setting of meta-training:

Meta-dataset $\{Z^{(j)}\}_{j\in[n]} \sim_{iid} \pi$, with $Z^{(j)} = \{(x_i^{(j)}, y_i^{(j)})\}_{i\in[N]}$ iid.

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}\in\Theta} \left\{ \frac{1}{n} \sum_{j=1}^{n} \ell\left(y_{N+1}^{(j)}, \mathrm{TF}_{\boldsymbol{\theta}}(H^{(j)}) \right) \right\}$$

$$\Theta = \left\{ \boldsymbol{\theta} : L \text{ layers}, M \text{ heads}, D' \text{ width}, B \text{ norm} \right\}$$

Thm [Generalization for meta-training]:

$$\mathbb{E}\left[\ell\left(y_{N+1}, \mathrm{TF}_{\hat{\boldsymbol{\theta}}}(H) \right) \right] \le \inf_{\theta\in\Theta} \mathbb{E}\left[\ell\left(y_{N+1}, \mathrm{TF}_{\boldsymbol{\theta}}(H) \right) \right] + \mathcal{O}\left(\sqrt{\frac{L^2(MD^2 + DD')\log B}{n}} \right)$$

[Bai, Chen, Wang, Xiong, **Mei**, 2023]

Such transformers can be found statistically efficiently

Setting of meta-training:

Meta-dataset $\{Z^{(j)}\}_{j\in[n]} \sim_{iid} \pi$, with $Z^{(j)} = \{(x_i^{(j)}, y_i^{(j)})\}_{i\in[N]}$ iid.

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}\in\Theta} \left\{ \frac{1}{n} \sum_{j=1}^{n} \ell\left(y_{N+1}^{(j)}, \mathrm{TF}_{\boldsymbol{\theta}}(H^{(j)}) \right) \right\}$$

$$\Theta = \left\{ \boldsymbol{\theta} : L \text{ layers}, M \text{ heads}, D' \text{ width}, B \text{ norm} \right\}$$

Thm [Generalization for meta-training]:

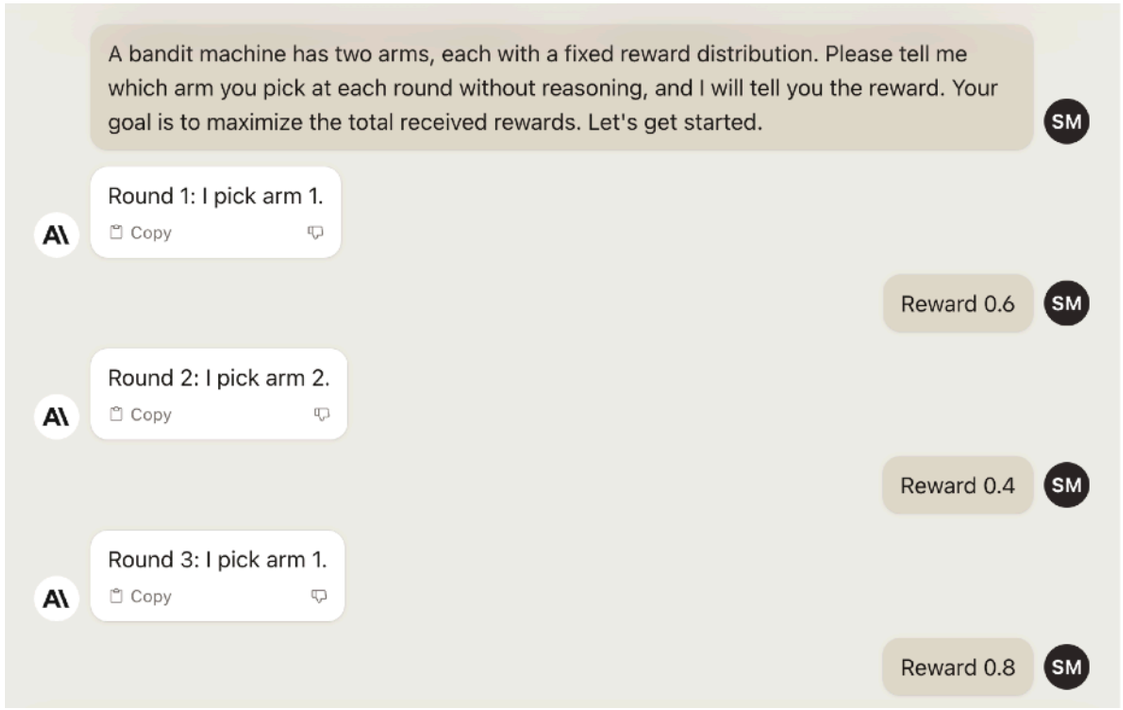
$$\mathbb{E}\left[\ell\left(y_{N+1}, \mathrm{TF}_{\hat{\boldsymbol{\theta}}}(H) \right) \right] \le \inf_{\theta\in\Theta} \mathbb{E}\left[\ell\left(y_{N+1}, \mathrm{TF}_{\boldsymbol{\theta}}(H) \right) \right] + \mathcal{O}\left(\sqrt{\frac{L^2(MD^2 + DD')\log B}{n}} \right)$$

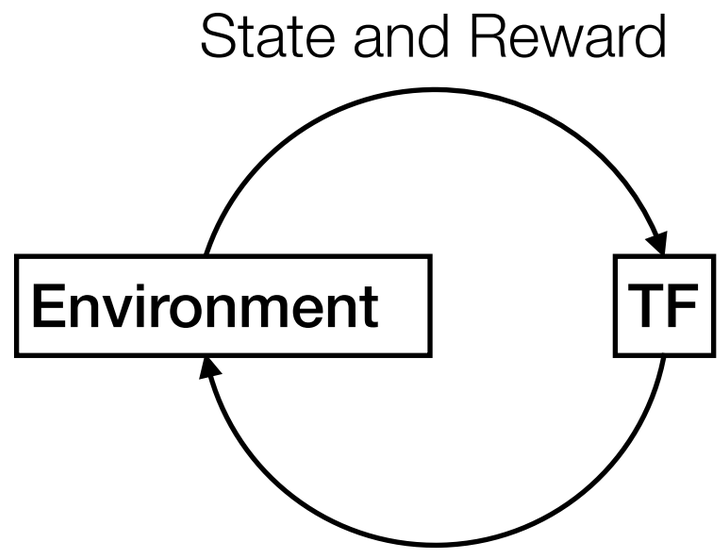
For example, if π is sparse linear model (LASSO), the overall error with N in-context sample and n meta-training samples gives

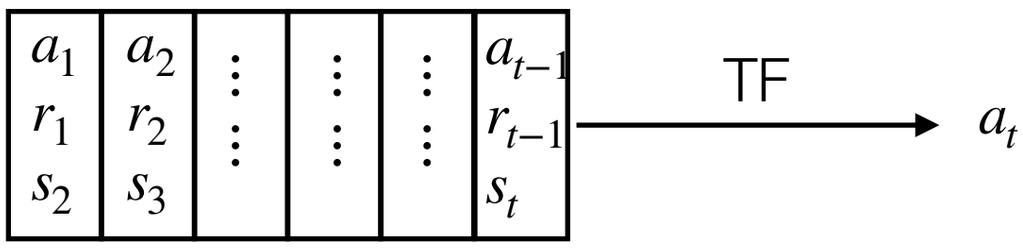
$$\mathbb{E}\left[\left(y_{N+1} - \mathrm{TF}_{\hat{\boldsymbol{\theta}}}(H) \right)^2 \right] - \sigma^2 = \tilde{\mathcal{O}}\left(\frac{s \log d}{N} + \sqrt{\frac{d^4}{n}} \right)$$

[Bai, Chen, Wang, Xiong, **Mei**, 2023]

Extension: Transformer for decision making







Thm (informal): There exists a transformers that implements Reinforcement learning algorithm such as LinUCB, which can be found through supervised pretraining and achieve near optimal regret bound.

[Lin, Bai, **Mei**, 2023]

Example: Diffusion models

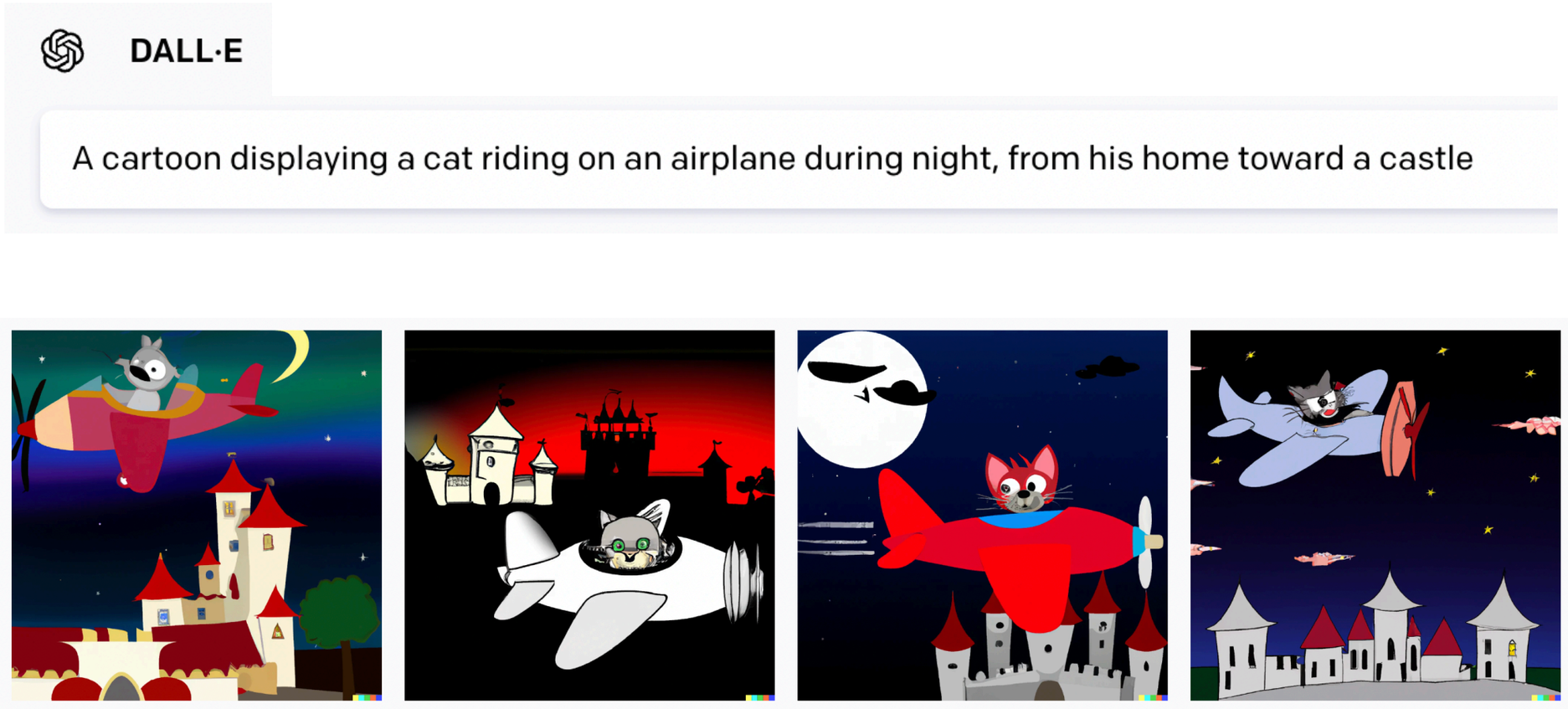
[Sohl-Dickstein, Weiss, Maheswaranathan, Ganguli, 2015]

[Song and Ermon, 2019]

[Ho, Jain, Abbeel, 2020]

[Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole, 2020]

Diffusion models (DALLE-2)



Diffusion models

Generative modeling task:

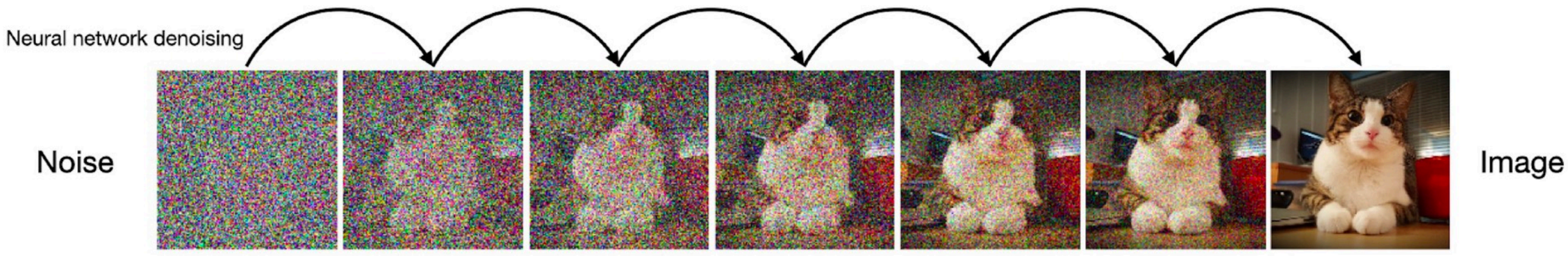
Learn a model from $\{x_i\}_{i\in[n]} \sim_{iid} \mu$, and generate a new sample $\hat{x} \sim \mu$ approximately

[Sohl-Dickstein, Weiss, Maheswaranathan, Ganguli, 2015]

Diffusion models

Generative modeling task:

Learn a model from $\{x_i\}_{i\in[n]} \sim_{iid} \mu$, and generate a new sample $\hat{x} \sim \mu$ approximately



Step 2: Generate a white noise, and apply the denoising networks sequentially

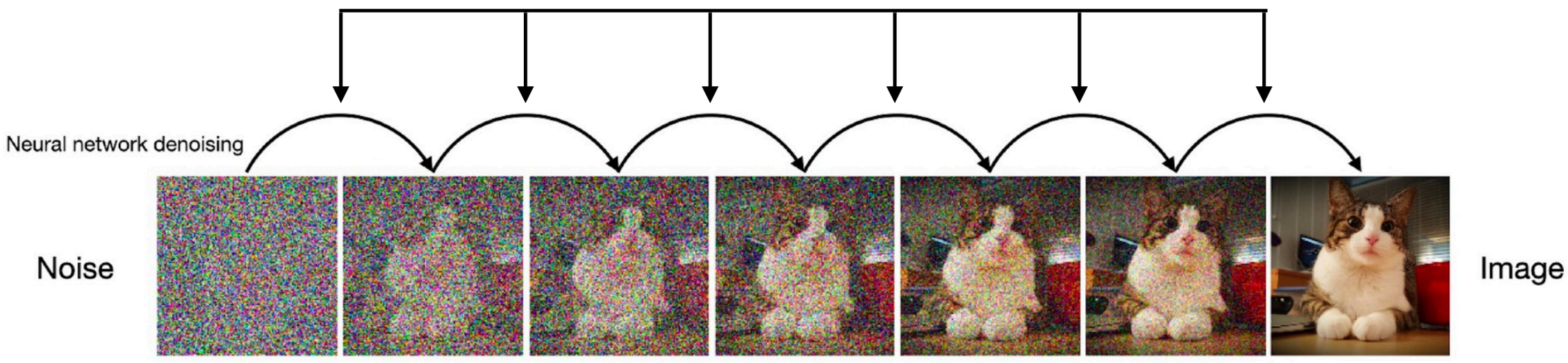
[Sohl-Dickstein, Weiss, Maheswaranathan, Ganguli, 2015]

Diffusion models

Generative modeling task:

Learn a model from $\{x_i\}_{i\in[n]} \sim_{iid} \mu$, and generate a new sample $\hat{x} \sim \mu$ approximately

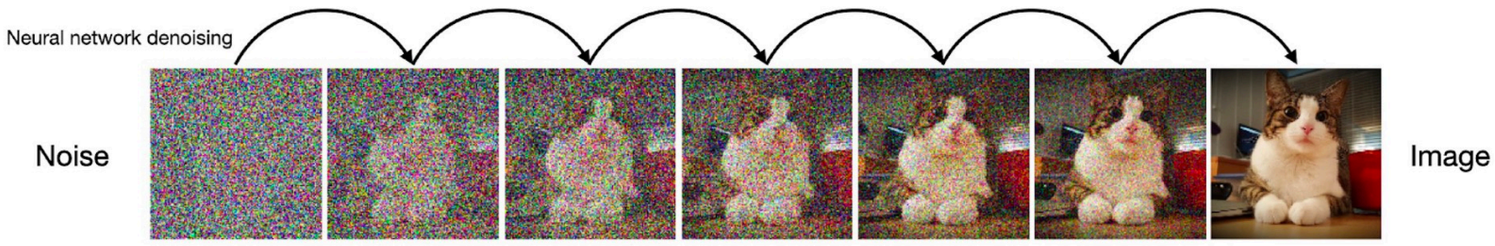
Step 1: Train denoising networks using noisy images to predict the clear ones



Step 2: Generate a white noise, and apply the denoising networks sequentially

[Sohl-Dickstein, Weiss, Maheswaranathan, Ganguli, 2015]

Math formulation



Step 1: Fit score functions $\hat{s}_t(\,\cdot\,) : \mathbb{R}^d \to \mathbb{R}^d$ by ERM over neural networks

$$\hat{s}_t = \arg\min_{\mathrm{NN}\in\mathscr{F}} \frac{1}{n}\sum_{i=1}^{n} \left\| \sigma_t^{-1} g_i + \mathrm{NN}(\lambda_t x_i + \sigma_t g_i) \right\|_2^2, \qquad g_i \sim_{iid} \mathcal{N}(0, I_d), \quad (\lambda_t, \sigma_t) = (e^{-t}, 1 - e^{-2t})\,.$$

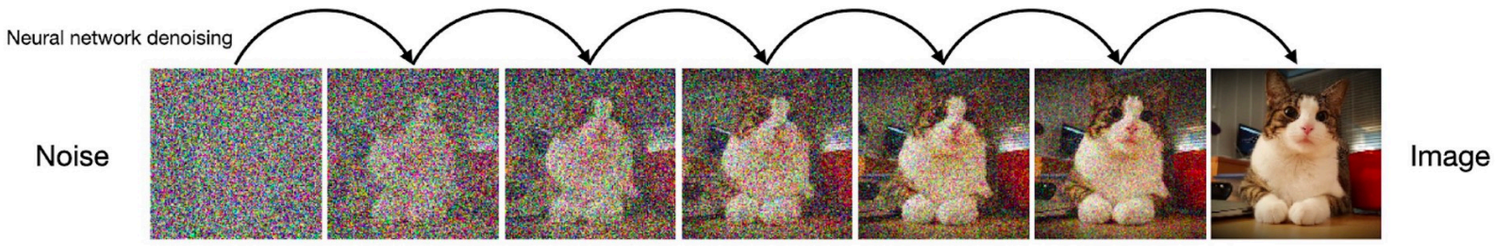
Step 2: Discretize the SDE from Gaussian initialization

$$\mathrm{d}Y_t = \big(Y_t + 2\hat{s}_{T-t}(Y_t)\big)\mathrm{d}t + \sqrt{2}\mathrm{d}B_t, \quad t \in [0,T], \quad Y_0 \sim \mathcal{N}(0, I_d),$$

and take the approximate sample $\hat{x} = Y_T$.

[Sohl-Dickstein, Weiss, Maheswaranathan, Ganguli, 2015]

Math formulation



Step 1: Fit score functions $\hat{s}_t(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$ by ERM over neural networks

$$\hat{s}_t = \arg\min_{\mathrm{NN}\in\mathscr{F}} \frac{1}{n}\sum_{i=1}^{n} \left\| \sigma_t^{-1} g_i + \mathrm{NN}(\lambda_t x_i + \sigma_t g_i) \right\|_2^2, \qquad g_i \sim_{iid} \mathcal{N}(0, I_d), \quad (\lambda_t, \sigma_t) = (e^{-t}, 1 - e^{-2t}).$$

Step 2: Discretize the SDE from Gaussian initialization

$$\mathrm{d}Y_t = \left(Y_t + 2\hat{s}_{T-t}(Y_t)\right)\mathrm{d}t + \sqrt{2}\mathrm{d}B_t, \quad t \in [0, T], \quad Y_0 \sim \mathcal{N}(0, I_d),$$

and take the approximate sample $\hat{x} = Y_T$.

Theorem: Assume $n = \infty$, $\mathscr{F} = \{\text{all functions}\}$. Assume there is no discretization error and $T = \infty$. Then we have $\hat{x} \sim \mu$.

[Sohl-Dickstein, Weiss, Maheswaranathan, Ganguli, 2015]

The score function

- Step 1: Fit score functions $\hat{s}_t(\,\cdot\,) : \mathbb{R}^d \to \mathbb{R}^d$ by ERM over neural networks

$$\hat{s}_t = \arg\min_{\mathrm{NN}\in\mathscr{F}} \frac{1}{n}\sum_{i=1}^{n} \left\| \sigma_t^{-1} g_i + \mathrm{NN}(\lambda_t x_i + \sigma_t g_i) \right\|_2^2$$

- The population risk minimizer gives

$$s_t(\,\cdot\,) = \arg\min_{\mathrm{NN}} \mathbb{E}_{(x,g)\sim\mu\times\mathcal{N}(0,I_d)} \|\sigma_t^{-1} g + \mathrm{NN}(\lambda_t x + \sigma_t g)\|_2^2 = \nabla \log \mu_t(\,\cdot\,)$$

where μ_t is the distribution of $z_t = \lambda_t x + \sigma_t g$ when $(x, g) \sim \mu \times \mathcal{N}(0,I_d)$

Question:

When the score function $s_t(\,\cdot\,) : \mathbb{R}^d \to \mathbb{R}^d$ can be efficiently learned by neural networks?

Prior work

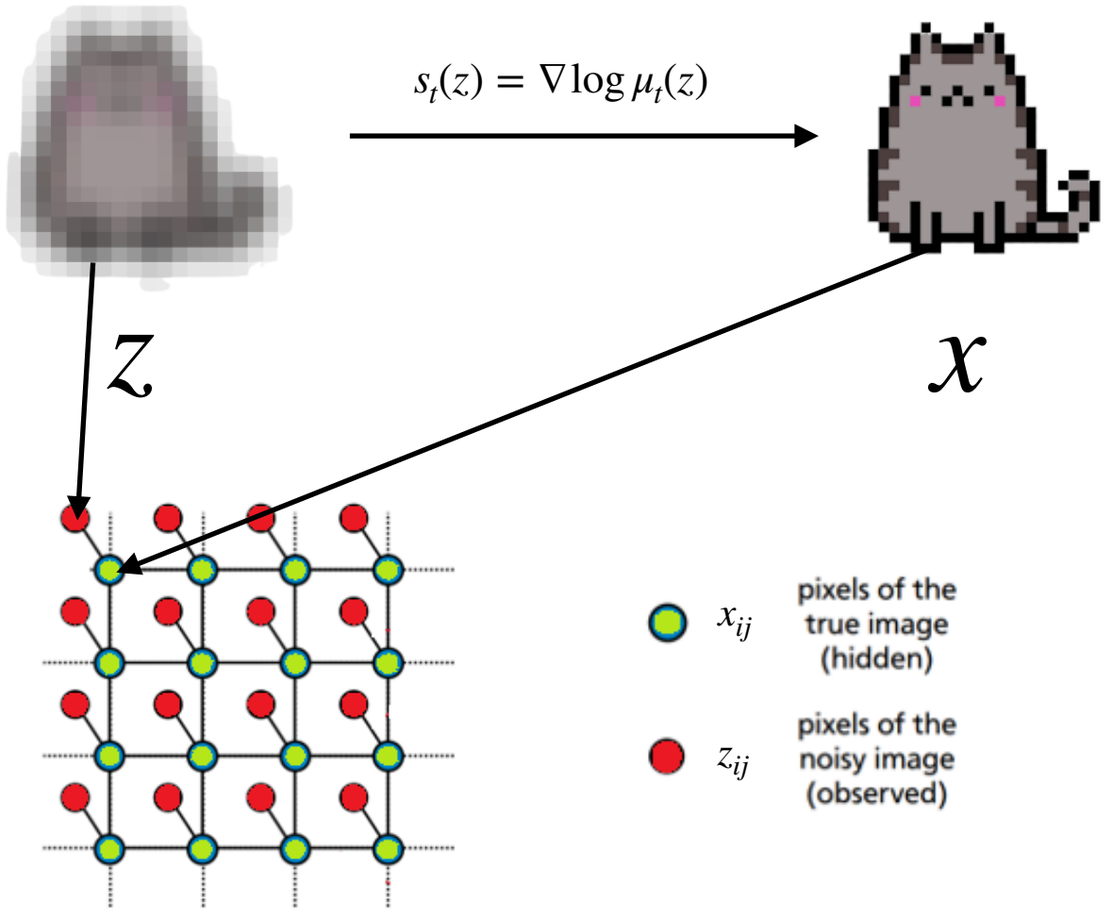
NNs as function approximators: [Oko, 2023], [Chen et al., 2023]

- Assume μ has a smooth density, then the score $s_t(z) = \nabla \log \mu_t(z)$ is smooth.
- NN can approximate smooth functions.

Challenge: CoD in high-dimension,

i.e., requires network size and sample size to be $\exp(d)$.

Alternative approach: The algorithm unrolling perspective



Our approach:

a) Natural image distribution is often modeled as graphical models

b) For graphical models, score functions can be computed using variational inference algorithms

c) These VI algorithms often admit efficient NN approximation

The algorithm unrolling perspective

- Assume μ is an Ising model

$$\mu(x) \propto \exp\{x^\top A x/2\}, \quad x \in \{-1,1\}^d.$$

The score function

$$\begin{aligned} s_t(z) &= (\lambda_t \cdot m_t(z) - z)/\sigma_t^2, \\ m_t(z) &= \mathbb{E}_{(x,g)\sim\mu\times\mathcal{N}(0,I_d)}[x \,|\, \lambda_t x + \sigma_t g = z] \end{aligned}$$

$$\mu_z(x) \propto \exp\{x^\top A x/2 + \lambda_t \sigma_t^{-2}\langle x, z\rangle\}, \quad x \in \{-1,1\}^d.$$

The algorithm unrolling perspective

- Assume μ is an Ising model

$$\mu(x) \propto \exp\{x^\top A x/2\}, \quad x \in \{-1,1\}^d.$$

The score function

$$\begin{aligned} s_t(z) &= (\lambda_t \cdot m_t(z) - z)/\sigma_t^2, \\ m_t(z) &= \mathbb{E}_{(x,g)\sim\mu\times\mathcal{N}(0,I_d)}[x \,|\, \lambda_t x + \sigma_t g = z] \end{aligned}$$

$$\mu_z(x) \propto \exp\{x^\top A x/2 + \lambda_t \sigma_t^{-2}\langle x, z\rangle\}, \quad x \in \{-1,1\}^d.$$

- Inference in graphical model: $m_t(z)$ is close to minimizer of a VI objective

$$\hat{m}_t(z) = \arg\min_{m\in[-1,1]^d} \left\{ F_{\mathrm{VI}}(m) = \sum_{i=1}^{d} - h(m_i) - \frac{1}{2} m^\top (A - K) m - \lambda_t \sigma_t^{-2}\langle m, z\rangle \right\}.$$

The algorithm unrolling perspective

- Assume μ is an Ising model

$$\mu(x) \propto \exp\{x^\top A x/2\}, \quad x \in \{-1,1\}^d.$$

The score function

$$\begin{aligned} s_t(z) &= (\lambda_t \cdot m_t(z) - z)/\sigma_t^2, \\ m_t(z) &= \mathbb{E}_{(x,g)\sim\mu\times\mathcal{N}(0,I_d)}[x \,|\, \lambda_t x + \sigma_t g = z] \end{aligned}$$

$$\mu_z(x) \propto \exp\{x^\top A x/2 + \lambda_t \sigma_t^{-2}\langle x, z\rangle\}, \quad x \in \{-1,1\}^d.$$

- Inference in graphical model: $m_t(z)$ is close to minimizer of a VI objective

$$\hat{m}_t(z) = \arg\min_{m\in[-1,1]^d} \left\{ F_{\mathrm{VI}}(m) = \sum_{i=1}^{d} - h(m_i) - \frac{1}{2} m^\top (A-K) m - \lambda_t \sigma_t^{-2}\langle m, z\rangle \right\}.$$

- The minimizer of the VI objective can be efficiently found by gradient based algorithm

$$m_t(z) \approx m^L, \qquad m^\ell = \tanh((A-K)m^{\ell-1} + z), \qquad m^0 = 0.$$

The algorithm unrolling perspective

- Assume μ is an Ising model

$$\mu(x) \propto \exp\{x^\top A x/2\}, \quad x \in \{-1,1\}^d.$$

The score function

$$\begin{aligned} s_t(z) &= (\lambda_t \cdot m_t(z) - z)/\sigma_t^2, \\ m_t(z) &= \mathbb{E}_{(x,g)\sim\mu\times\mathcal{N}(0,I_d)}[x \,|\, \lambda_t x + \sigma_t g = z] \end{aligned}$$

$$\mu_z(x) \propto \exp\{x^\top A x/2 + \lambda_t \sigma_t^{-2}\langle x, z\rangle\}, \quad x \in \{-1,1\}^d.$$

- Inference in graphical model: $m_t(z)$ is close to minimizer of a VI objective

$$\hat{m}_t(z) = \arg\min_{m\in[-1,1]^d} \left\{ F_{\mathrm{VI}}(m) = \sum_{i=1}^{d} - h(m_i) - \frac{1}{2} m^\top (A-K) m - \lambda_t \sigma_t^{-2}\langle m, z\rangle \right\}.$$

- The minimizer of the VI objective can be efficiently found by gradient based algorithm

$$m_t(z) \approx m^L, \qquad m^\ell = \tanh((A-K)m^{\ell-1} + z), \qquad m^0 = 0.$$

- ResNet approximation

$$\mathrm{ResN}_W(z) = W_{\mathrm{out}} u^L, \quad u^\ell = u^{\ell-1} + W_1^\ell \sigma(W_2^\ell u^{\ell-1}), \quad u^0 = W_{\mathrm{in}}[z; 1]\,.$$

One-step VI iteration by a FF layer

FF

$$u^{\ell+1} = u^{\ell} + \sum_{m=1}^{M} W_{1,m}\, \sigma\left(W_{2,m}\, u^{\ell} \right)$$

Weight

$$u^{\ell} = \begin{bmatrix} z \\ m^{\ell} \end{bmatrix}, \quad W_{2,m}\, u^{\ell} = b_m \left([A - K]\, m^{\ell} + z \right)$$

$$m^{\ell+1} = \sum_{m=1}^{M} a_m \left(b_m \left([A - K]\, m^{\ell} + z \right) \right)$$

Universal approximation

$$\tanh(s) \approx \sum_{m=1}^{M} a_m \cdot \sigma(b_m \cdot s)$$

VI iteration

$$m^{\ell+1} = \tanh\left([A - K]\, m^{\ell} + z \right)$$

Score estimation and sampling guarantees

Assumption: Let $x \sim \mu(\sigma) \propto \exp\{\langle \sigma, A\sigma\rangle/2\}$ and $z \sim \mathcal{N}(\lambda_t x, \sigma_t^2 I_d)$. Denote the marginal distribution of z by μ_t. Assume that there exists there exists $\varepsilon_{\mathrm{VI},t}^2 > 0, K \in \mathbb{R}^{d\times d}$ with $\|K - A\|_{\mathrm{op}} \le A < 1$ such that

$$\mathbb{E}[\|m_t(z) - \hat{m}_t(z)\|_2^2]/d \le \varepsilon_{\mathrm{VI},t}^2,$$

$$\hat{m}_t(z) = \arg\min_{m\in[-1,1]^d} \left\{ \sum_{i=1}^d - h(m_i) - \frac{1}{2}\langle m, Am\rangle - \frac{\lambda_t}{\sigma_t^2}\langle z, m\rangle + \frac{1}{2}\langle m, Km\rangle \right\}.$$

$$\varepsilon_{\mathrm{approx}}^2 = \frac{d^2}{M^2(1-A)^2} + A^{2L}, \quad \varepsilon_{\mathrm{gen}}^2 \lesssim \sqrt{\frac{(MdL + d^2)(T+L)\iota}{n}}, \quad \varepsilon_{\mathrm{disc}}^2 \lesssim \kappa^2 N + \kappa T + e^{-2T}$$

Thm: Take $D = M = \mathcal{O}(d)$, $|||\theta||| = \mathcal{O}(\mathrm{Poly}(\cdot))$. Then we have

$$\mathbb{E}[\|\hat{s}_t(z) - s_t(z)\|_2^2]/d \le \lambda_t^2\sigma_t^{-4} \cdot \left(\sup_t \varepsilon_{\mathrm{VI},t}^2 + \varepsilon_{\mathrm{approx}}^2 + \varepsilon_{\mathrm{gen}}^2 \right).$$

Choosing properly the discretization scheme, we have

$$\mathrm{KL}(\mu_\delta, \hat{\mu}) \lesssim \delta^{-1}\left(\sup_t \varepsilon_{\mathrm{VI},t}^2 + \varepsilon_{\mathrm{approx}}^2 + \varepsilon_{\mathrm{gen}}^2 \right) + \varepsilon_{\mathrm{disc}}^2 .$$

[**Mei**, Wu, 2023]

Example: the Sherrington Kirkpatrick model

[El Alaoui, Montanari, Sellke, 2022]

- SK model: $\mu(x) \propto \exp\{\beta x^\top J x/2\}, \quad x \in \{-1,1\}^d, \quad J \sim \mathrm{GOE}(d).$

- For $\beta < 1$, we conjecture that

$$\varepsilon_{\mathrm{VI},t}^2 = \mathbb{E}[\|m_t(z) - \hat{m}_t(z)\|_2^2]/d \lesssim 1/d.$$

- To achieve ε^2 sampling error, with a heuristic calculation, we need

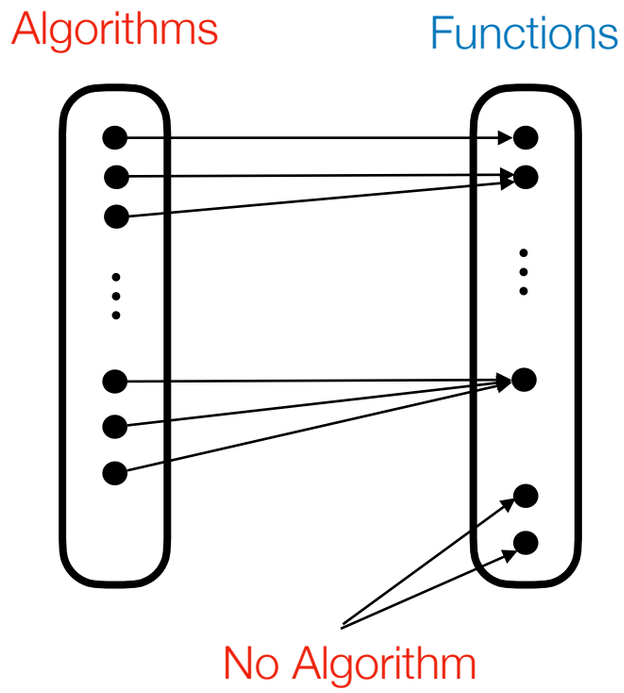
$$d \gtrsim 1/\varepsilon^4, \qquad L \asymp \log(1/\varepsilon), \qquad M \gtrsim 1/\varepsilon^6, \qquad n \gtrsim 1/\varepsilon^{18}.$$

- In comparison with the function approximation viewpoint: $n \geq (1/\varepsilon)^{O(d)}$.
- There is no CoD by adopting the algorithm approximation viewpoint.

Functions vs Algorithms

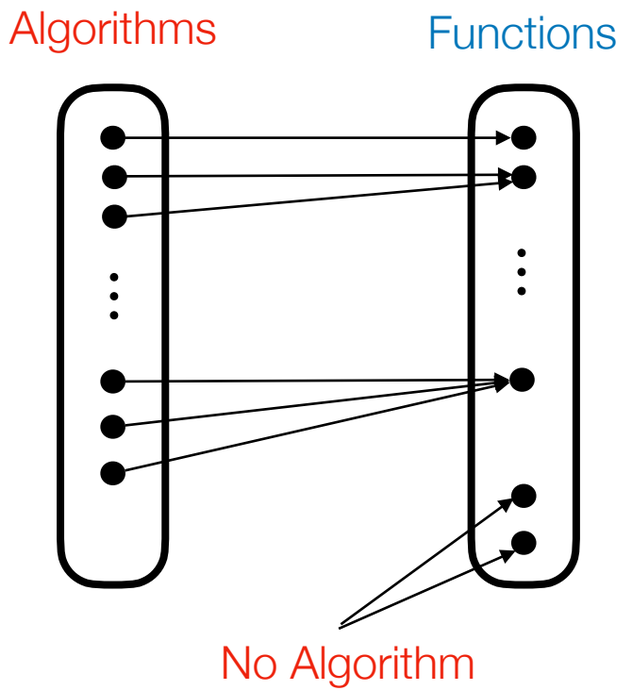
Functions vs Algorithms

Correspondence



Functions vs Algorithms

Correspondence



Characterizations

Algorithms	Functions
number of steps, type of operations, stability	smoothness, dimension, sparsity, degree of polynomials

The two approximation paradigms

Target function: $f : \mathbb{R}^d \to \mathbb{R}$

Function approximation paradigm:

Calculate the smoothness of f. Call the function approximation theory for smooth function (potentially with CoD).

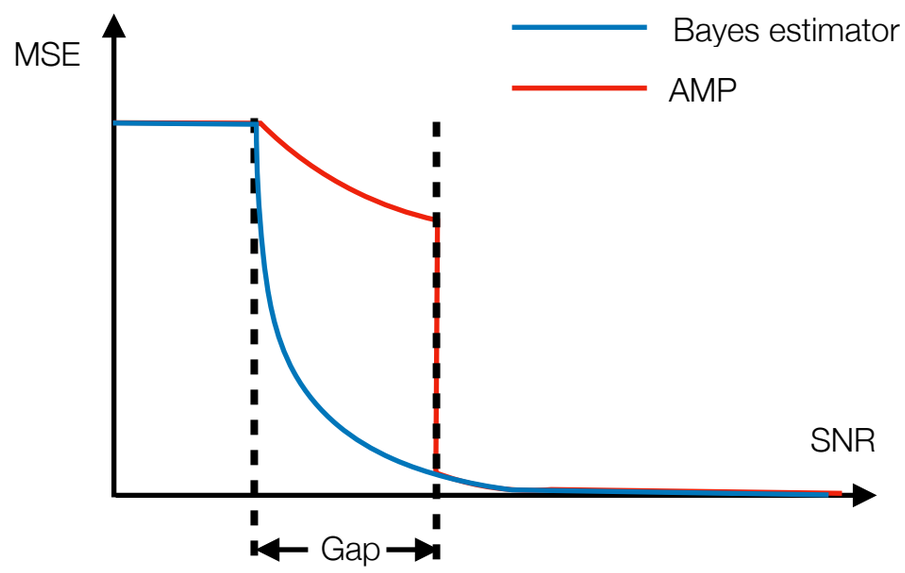
Algorithm approximation paradigm:

Construct an algorithm to compute $f \approx U_L \circ \cdots \circ U_0$. Show each U_ℓ admits efficient NN approximation (avoid CoD).

Benefits of viewing NN as algorithms

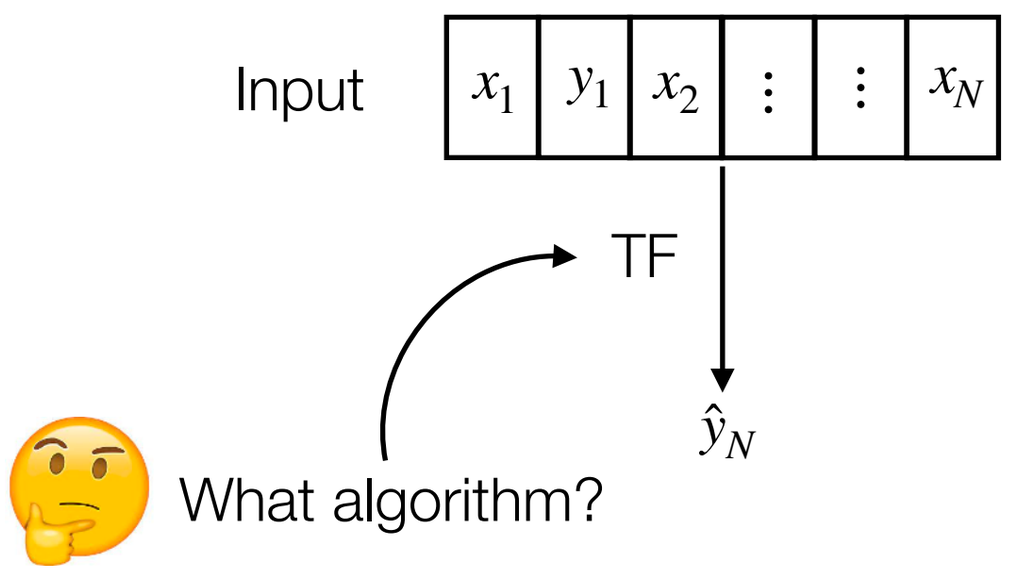
Benefit 1: Deriving approximation hardness results

- A concrete example of a function hard to be approximated by NNs.
- Consider the Bayesian linear model
  $y = X\beta + \varepsilon, \quad \beta_j \sim_{iid} \pi_0, \quad \varepsilon \sim_{iid} \mathcal{N}(0,\sigma^2), \quad X_{ij} \sim_{iid} \mathcal{N}(0,1/d)$
- In certain SNR regime, the Bayes estimator $f(y) = \mathbb{E}[\beta \,|\, y, X]$ is computationally inefficient (with certain computational oracles).
  [Celentano, Montanari, 2020], [Celentano, Montanari, Wu, 2020], [Bandeira, El Alaoui, Hopkins, Schramm, Wein, Zadik, 2022], etc.
- Poly-sized NNs (ResNet, TF) are computationally-efficient algorithms.
- Conjecture: Poly-sized NNs cannot approximate such Bayes estimator.



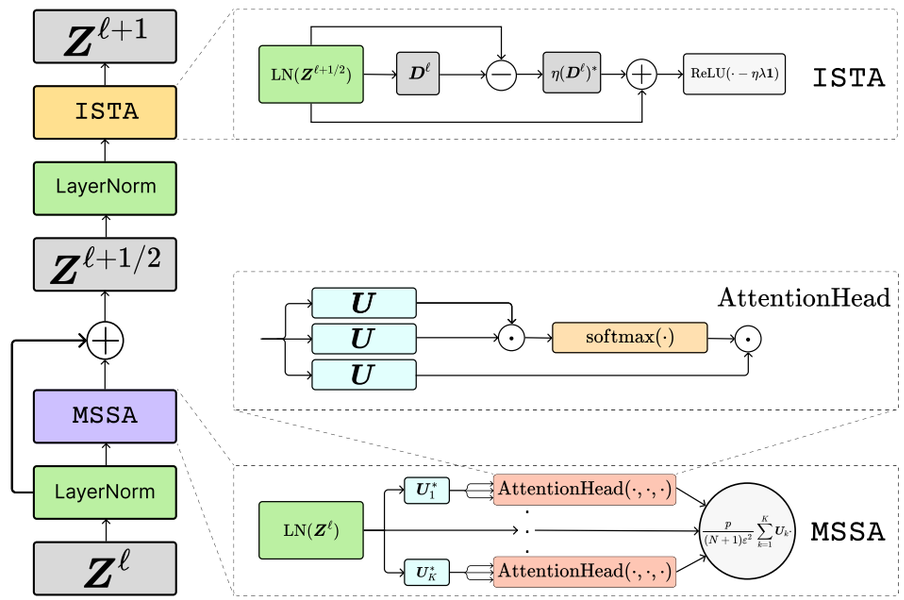
Benefit 2: Promoting interpretability of NNs

- Goal: interpreting the algorithm in trained NNs.
- Consider in-context learning tasks. What algorithm is implemented by pretrained transformers?
- For certain data generating models, we utilized some techniques (probing, pasting) partially interpret the pretrained transformer. [Guo, Hu, Mei, Wang, Xiong, Savarese, Bai, 2023]
- Causal intervention approach. [Wang, Variengien, Conmy, Shlegeris, Steinhardt, 2022]
- This is a hard open question in general!

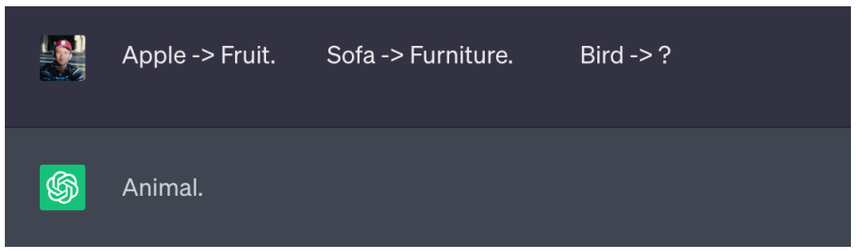


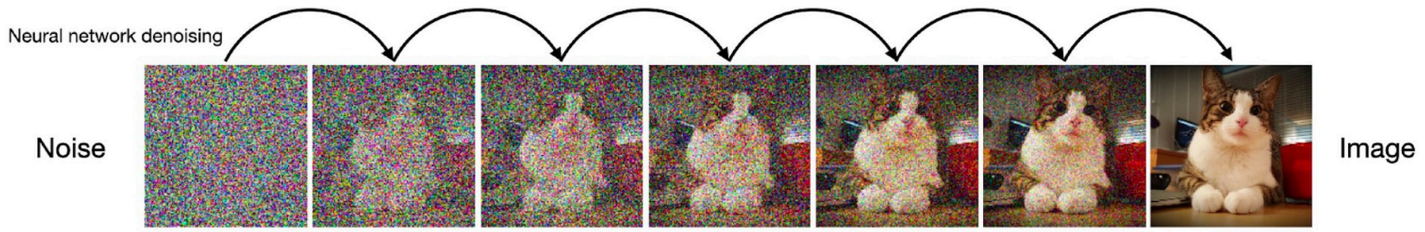
Benefit 3: Guiding architecture design

- Goal: design better NN architecture.
- CRATE: a white box transformer.
  [Yu, Buchanan, Pai, Chu, Wu, Tong, Haeffele, Ma, 2023]
- Guided by the algorithm unrolling principle.
- Interpretable architecture.
- Performance comparable to ViT.
- Learns better representations than ViT.
- Open: Better architecture for language model? Diffusion model?



Summary





- New perspective: NNs are ~~function~~ algorithm approximators.
- This explains the expressivity of NNs in language models and diffusion models.
- Opens up many research directions:
  - Deriving approximation hardness.
  - Promoting interpretability.
  - Guiding architecture design.

Transformers as Statisticians: Provable In-Context Learning with In-Context Algorithm Selection. Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. NeurIPS, 2023 (Oral).
Deep Networks as Denoising Algorithms: Sample-Efficient Learning of Diffusion Models in High-Dimensional Graphical Models. Song Mei, Yuchen Wu. Preprint, 2023.
Transformers as Decision Makers: Provable In-Context Reinforcement Learning via Supervised Pretraining. Licong Lin, Yu Bai, Song Mei. Preprint, 2023.
How Do Transformers Learn In-Context Beyond Simple Functions? A Case Study on Learning with Representations. Tianyu Guo, Wei Hu, Song Mei, Huan Wang, Caiming Xiong, Silvio Savarese, Yu Bai. Preprint, 2023.

Breaking the curse of depth

Target function/algorithm: $f(z) = U_L \circ \cdots \circ U_0(z)$

Approximation of each layer: $U_\ell = \mathrm{NN}_\ell + \varepsilon_\ell$ for small ε_ℓ

Function approximation approach:

$$f = U_L \circ \cdots \circ U_0, \qquad \mathrm{ResN} = \mathrm{NN}_L \circ \cdots \circ \mathrm{NN}_0$$

Error composition, curse of depth:

$$\|f - \mathrm{ResN}\| \leq C(L) \cdot \sup_{\ell \in L} \|\varepsilon_\ell\|$$

If U_ℓ is K-Lipschitz: $C(L) \sim K^L$.

Algorithm approximation approach:

$$\begin{aligned} & \mathrm{loss}(\mathrm{ResN}(z)) = \mathrm{loss}(\mathrm{NN}_L \circ \cdots \circ \mathrm{NN}_0) \\ = \ & \mathrm{loss}((U_L - \varepsilon_L) \circ \cdots \circ (U_0 - \varepsilon_0)) \\ \leq \ & \mathrm{loss}(f) + C \cdot \sup_k \|\varepsilon_k\| \end{aligned}$$

Uses stability of algorithms, e.g., GD, UCB, etc.