AutoML 2023 Tutorial

Google DeepMind

Language Modeling for Optimization

Yutian Chen, Richard Song

13 September 2023

Attention Is All You Need

reit*
rch
com

m

Language Models are Few-Shot Learners

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*

Jared Kaplan† Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry

Amanda Askell Sandhini Agarwal Ariel Herbert-

Daniel M. Zie

Eric Sigl

Jack Clark

dford

OpenAI

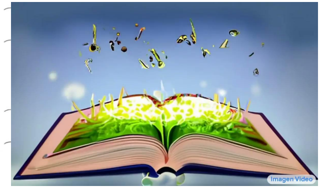
Abstract

Zero-Shot Text-to-Image Generation

Aditya Ramesh [1] Mikhail Pavlov [1] Gabriel Goh [1] Scott Gray [1]
Chelsea Voss [1] Alec Radford [1] Mark Chen [1] Ilya Sutskever [1]

Abstract

Text-to-image generation has traditionally fo-
cused on finding bette
training on a fixed da
might involve compl
losses, or side informa
bels or segmentation r
ing. We describe a si
based on a transforme
els the text and image
data. With sufficient d
is competitive with pre
els when evaluated in



AudioLM: a Language Modeling Approach to Audio Generation

Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin,
Matt Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, Neil Zeghidour

***Abstract*—We introduce AudioLM, a framework for high-quality audio generation with long-term consistency. AudioLM maps the input audio to a sequence of discrete tokens and casts au-**

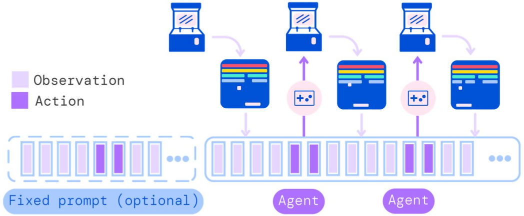
relying on textual annotations. Yet, the acoustic diversity and the quality remain limited: the model is trained on clean

A Generalist Agent

Scott Reed*,†, Konrad Żołna*, Emilio Parisotto*, Sergio Gómez Colmenarejo†, Alexander Novikov, Gabriel Barth-Maron, Mai Giménez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar and Nando de Freitas†

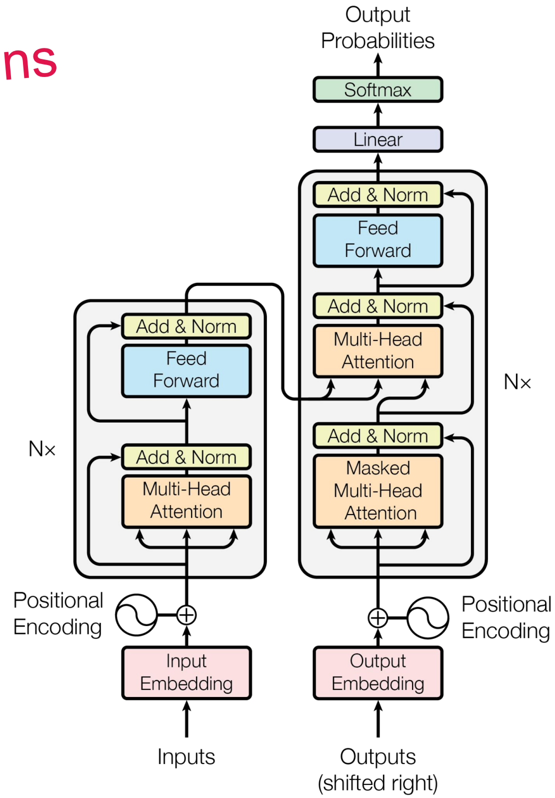
*Equal contributions, †Equal senior contributions, All authors are affiliated with DeepMind *reedscot@deepmind.com*

Reviewed on OpenReview: `https://openreview.net/forum?id=1ikK0kHjvj`

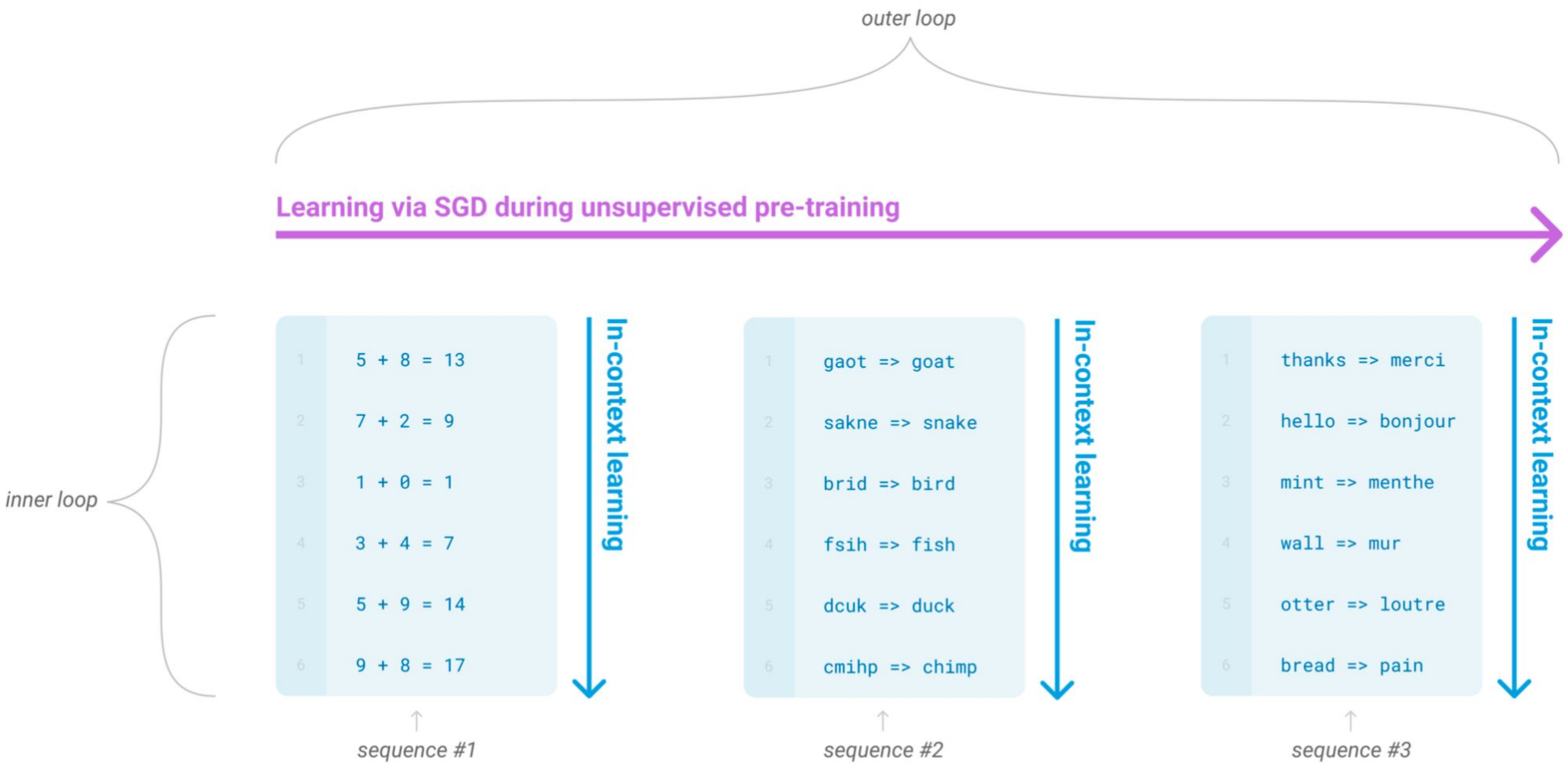


Transformers (Vaswani et al., 2017)

88K+ citations

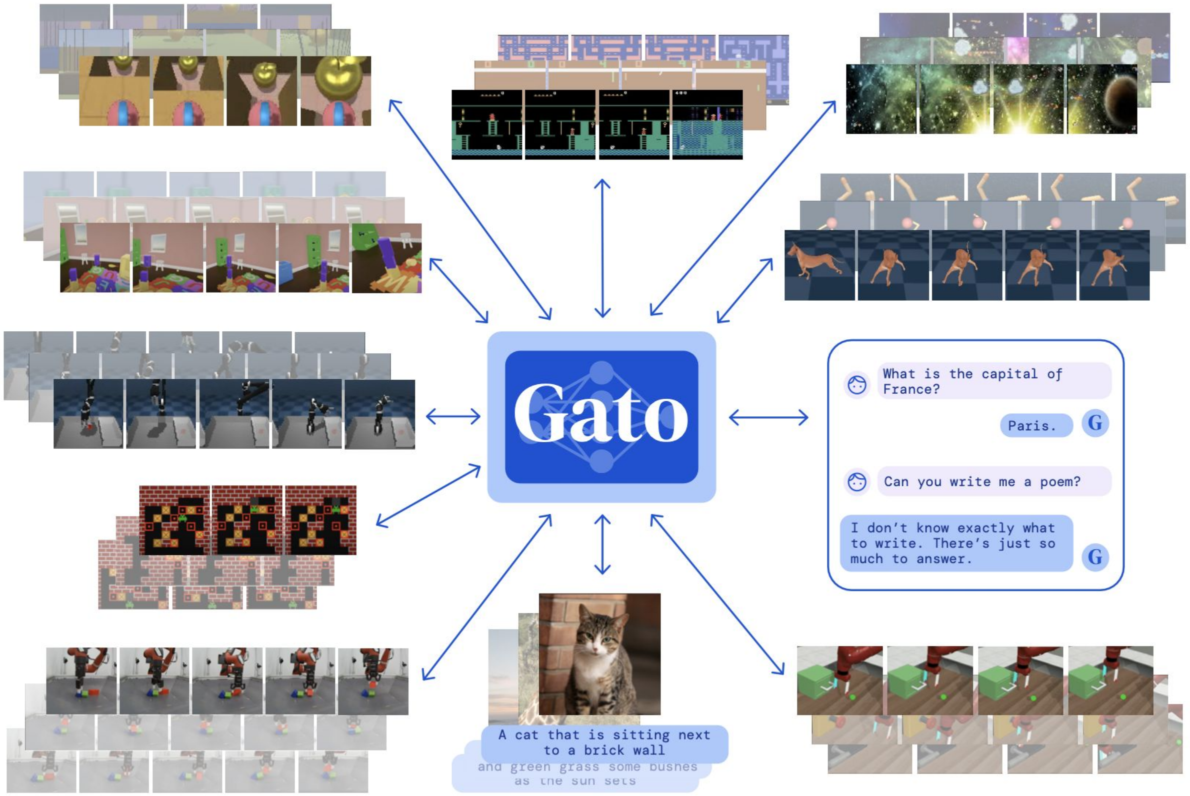


In-context learning



GPT-3 (Brown et al., 2020)

Unifying multi-modality as a single language model



Gato (Reed et al., 2022)

LLMs

- NLP
- Speech
- Vision
- Games
- Controls
- Robotics
- AI for Science
- ...
- AutoML

AutoML

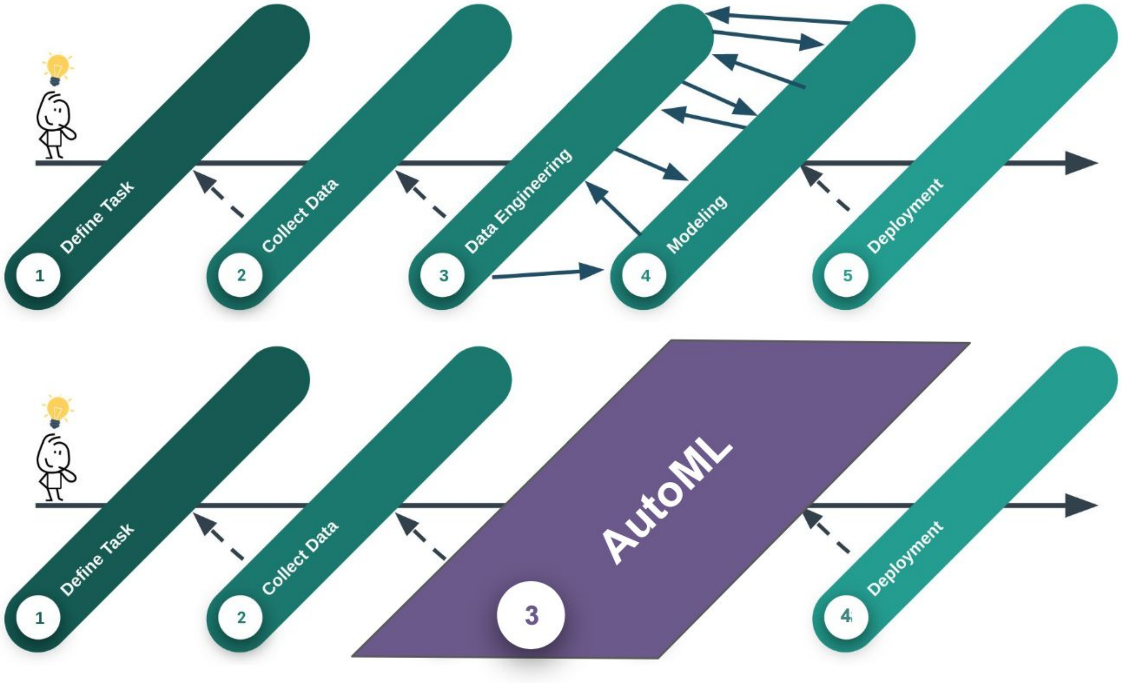
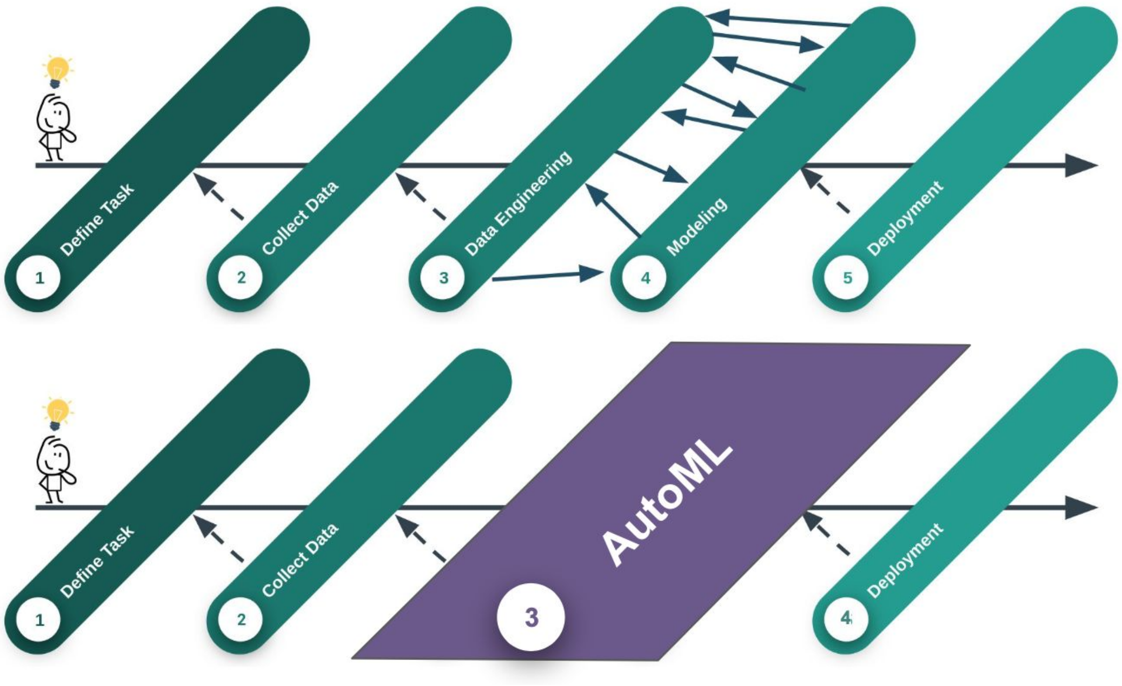


Figure source: https://www.automl.org/talks/

AutoML

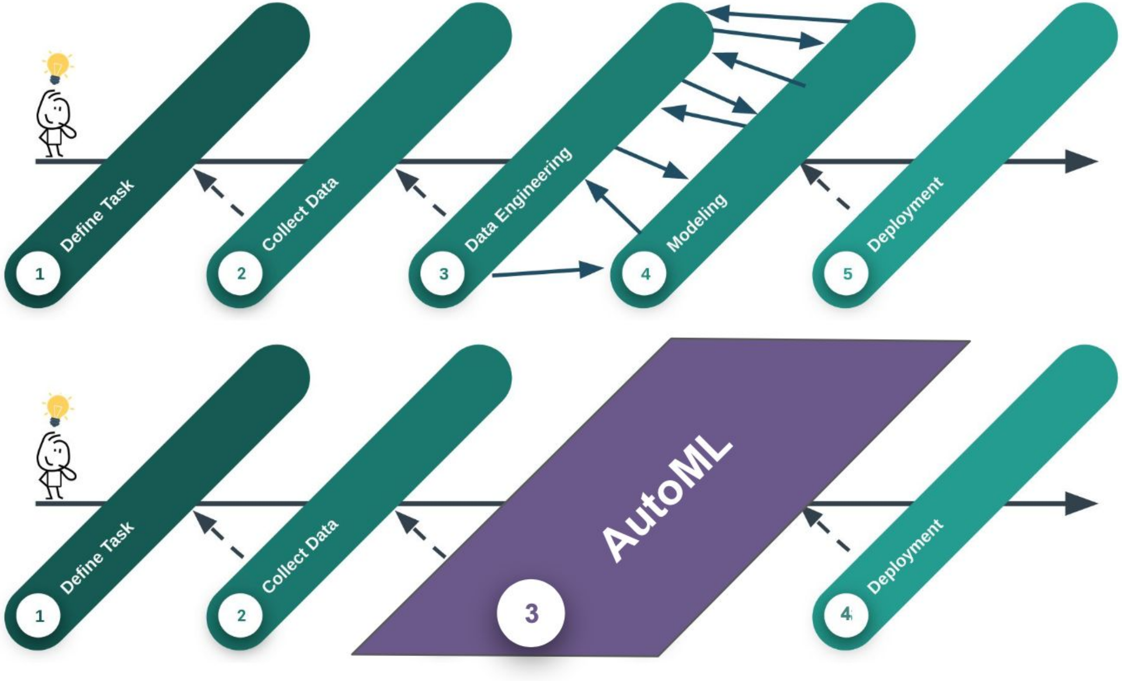


Traditional ML pipeline

- *Data preprocess pipeline*
- *Feature engineering*
- *Select a model family*
- *Hyperparameters selection*
- *Model training*
- *Evaluation*

Figure source: https://www.automl.org/talks/

AutoML



AutoML

- *Hyperparameter optimization (HPO)*
- *Neural Architecture Search (NAS)*
- *Meta-learning*
- *…*

Reference: https://www.automl.org/talks/

Optimization in AutoML

$$\lambda^* = \arg\min_{\lambda \in \Lambda} \mathcal{L}(\lambda, \mathcal{D}_{\text{train}}, \mathcal{D}_{\text{valid}})$$

- Hyperparameter optimization (HPO)
  - λ: hyperparameter values
- Neural Architecture Search (NAS)
  - λ: network architecture
- Meta-learning
  - λ: optimization algorithm, model initialization, etc
- ...

Optimization in AutoML

$$\lambda^* = \arg\min_{\lambda \in \Lambda} \mathcal{L}(\lambda, \mathcal{D}_{\text{train}}, \mathcal{D}_{\text{valid}})$$

- Hyperparameter optimization (HPO)
  - λ: hyperparameter values
- Neural Architecture Search (NAS)
  - λ: network architecture
- Meta-learning
  - λ: optimization algorithm, model initialization, etc
- ...

Gradient-free optimization

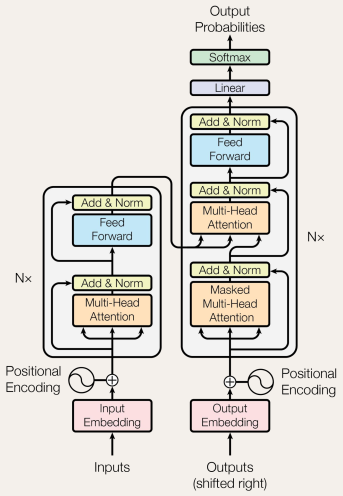
Black-box optimization (BBO)

- *Bayesian optimization*
- *Reinforcement learning*
- *Evolutionary strategy*
- *Genetic algorithms*
- *...*

What can LMs do for AutoML?

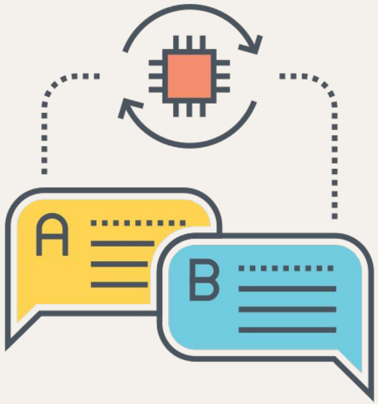
Transformer architecture

- *Powerful modeling capacity*
- *In-context learning*



Go beyond BBO

- *Leverage textual information*
- *More natural interface for AutoML*



Google DeepMind

Outline

- Basics of language modeling and transformers
- Transformers for optimization
- Large language models for optimization
- Vision and opening questions

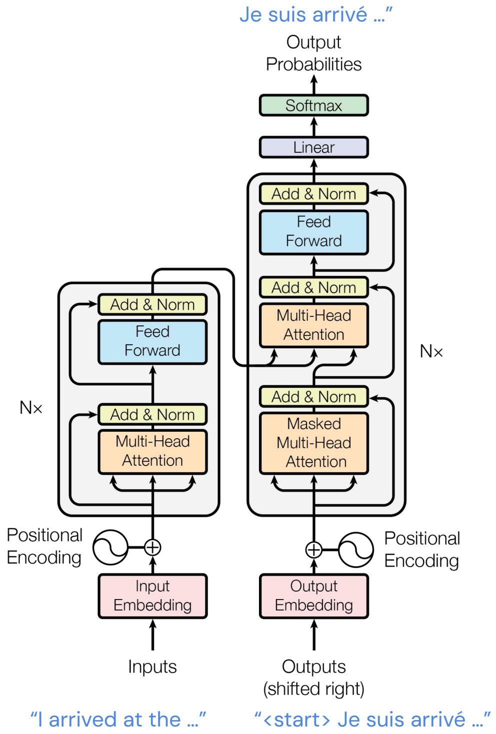
Google DeepMind

Outline

- Basics of language modeling and transformers
- Transformers for optimization
- Large language models for optimization
- Vision and opening questions

The Transformer: Attention is all you need
(Vaswani et al., 2017)

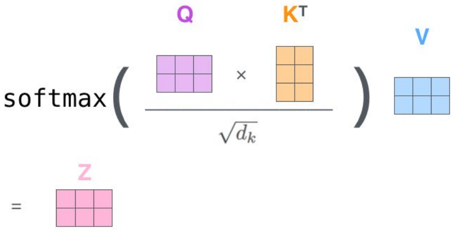
- Encoder-decoder architecture



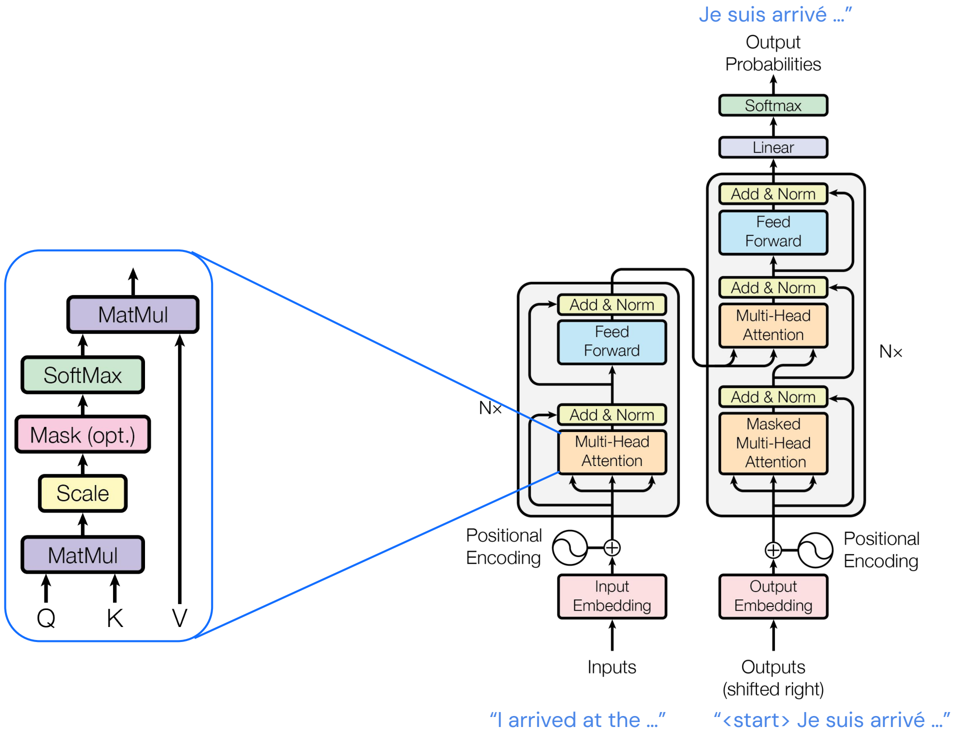
The Transformer: Attention is all you need
(Vaswani et al., 2017)

- Encoder-decoder architecture
- Attention-base sequence model
  - Scaled Dot-Product Attention

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



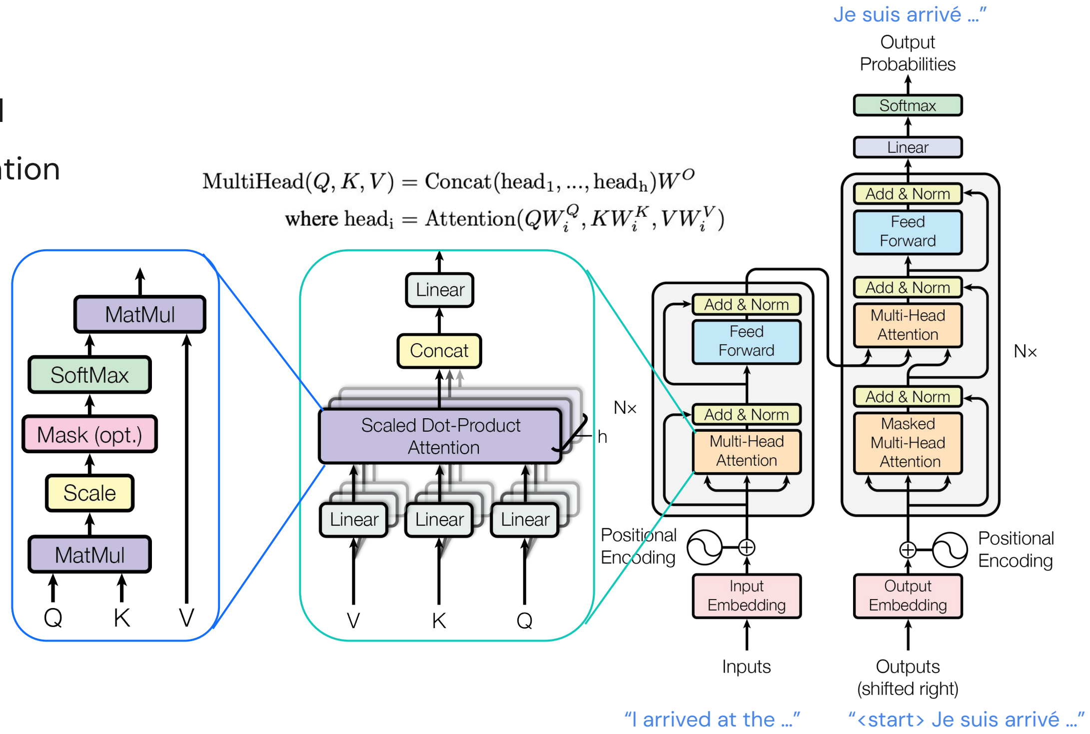
(Figure source: jalammar.github.io/illustrated-transformer)



The Transformer: Attention is all you need
(Vaswani et al., 2017)

- Encoder-decoder architecture
- Attention-base sequence model
  - Scaled Dot-Product Attention
  - Multi-Head Attention

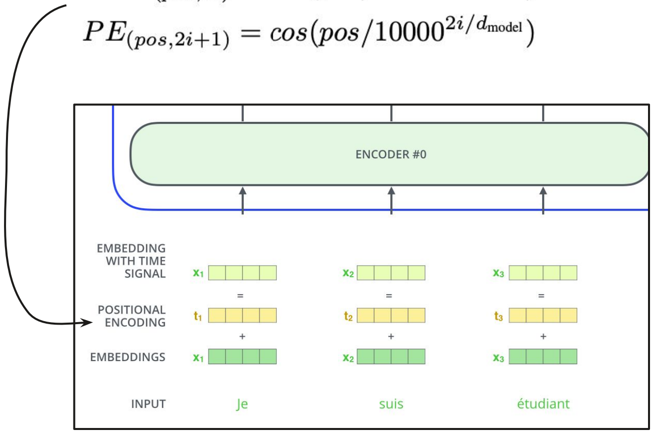
$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where head}_i = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

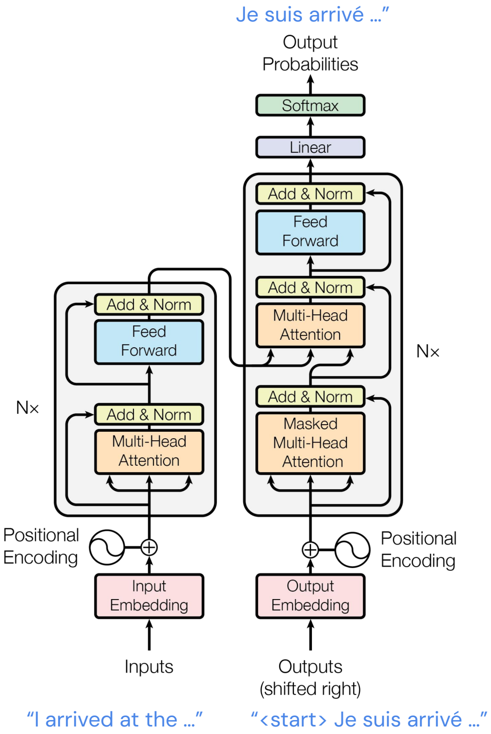


The Transformer: Attention is all you need
(Vaswani et al., 2017)

- Encoder-decoder architecture
- Attention-base sequence model
- Positional encoding - representing the order of the sequence

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$



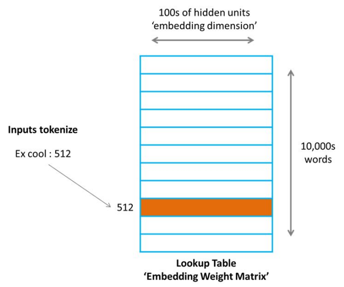


Transformers inputs

- **Text**
  - *Tokenizer*: sentence → token ids

    "I arrived at the …" → 40, 5284, 379, 262, …
  - *Embedding table*: token ids → embedding vectors

    40, 5284, 379, 262, … → Inputs = $[e_{40}, e_{5284}, e_{379}, e_{262}, \ldots]$

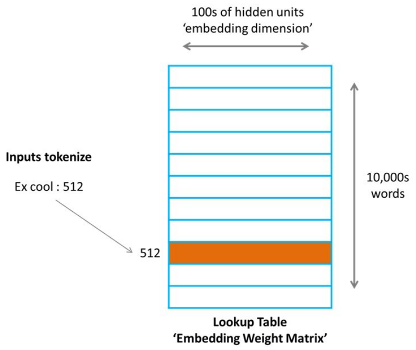


Transformers inputs

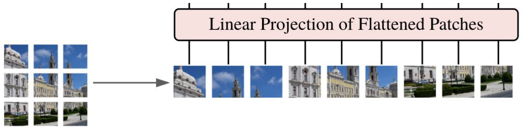
- **Text**
  - *Tokenizer*: sentence → token ids

    "I arrived at the …" → 40, 5284, 379, 262, …
  - *Embedding table*: token ids → embedding vectors

    40, 5284, 379, 262, … → Inputs = $[e_{40}, e_{5284}, e_{379}, e_{262}, \dots]$

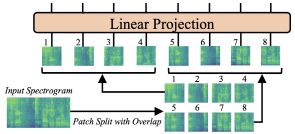


- **Images**



(Dosovitskiy et al., 2020)

- **Audio**



(Gong et al., 2021)

- **Discrete features**: One-hot encoding

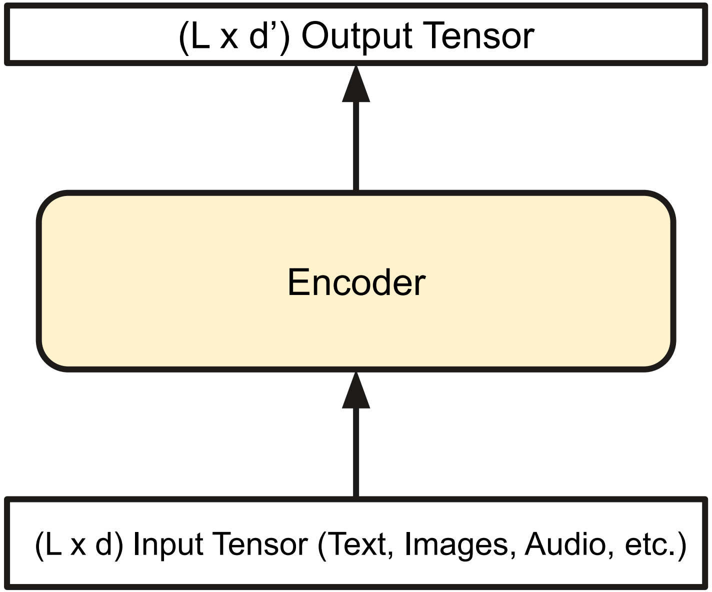
  Continuous features: Normalization (linear, mu-law, Riemann distribution)

  (Chen et al., 2022; Reed et al., 2022; Muller et al., 2023)

Transformers outputs: representation learning

Use raw output (L x d’) in any way

- Directly into downstream models
- Take single slice (e.g. embeddings)

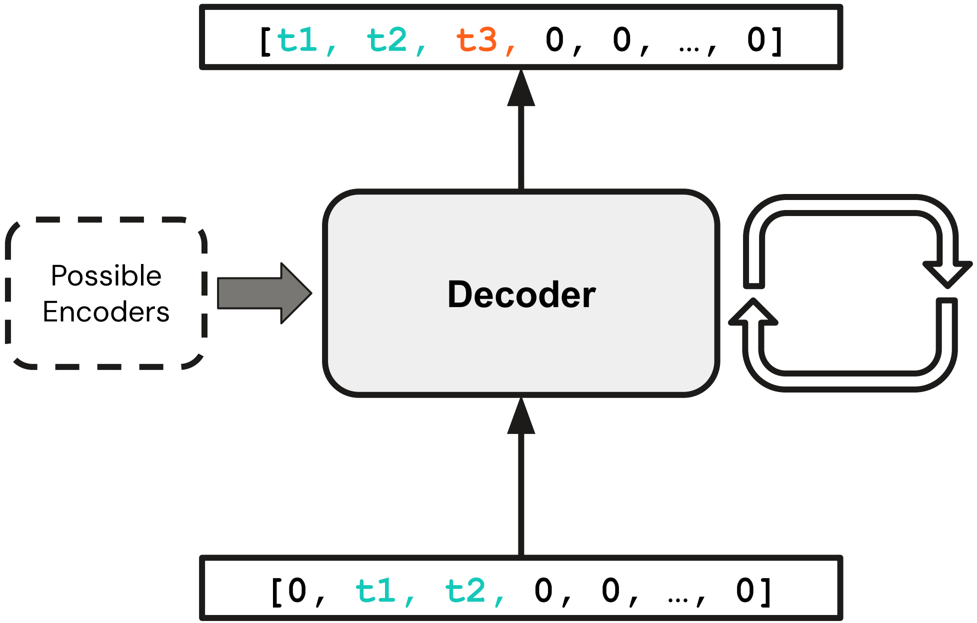


Transformers outputs: answer generation

Multi-step decoding: 1 forward pass = **1 new token**

Decoding Methods

- Ancestral Sampling
  - Temperature sampling
  - Top-K Sampling
  - Nucleus Sampling
  - …
- Max likelihood sequence
  - Beam Search

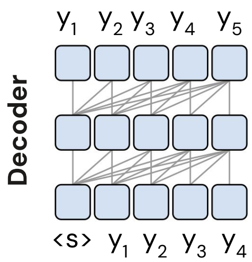


Transformers pre-training

- **Causal language modeling** (e.g. GPT, PaLM)

  $$\mathcal{L} = \sum_i \log P\left(y_i | \mathrm{Decoder}(\mathbf{y}_{1:i-1})\right)$$

  - E.g. log P(“Thank you for inviting me to your party last week”)

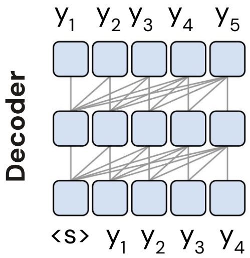


Transformers pre-training

- **Causal language modeling** (e.g. GPT, PaLM)

  $$\mathcal{L} = \sum_i \log P\left(y_i | \mathrm{Decoder}(\mathbf{y}_{1:i-1})\right)$$

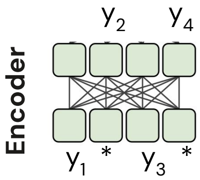
  - E.g. log P(“Thank you for inviting me to your party last week”)



- **Masked language modeling** (e.g. BERT (Kenton and Toutanova, 2019))

  $$\mathcal{L} = \sum_{i \in \text{masked set}} \log P(y_i | \mathrm{Encoder}(\mathbf{y}_{\text{masked}}))$$

  - E.g. log P(“* * for inviting * * * * last *” | “Thank you * * me to your party * week”)

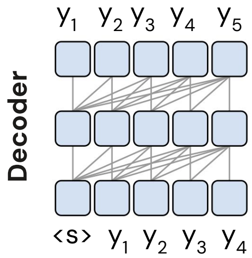


Transformers pre-training

- **Causal language modeling** (e.g. GPT, PaLM)

  $$\mathcal{L} = \sum_i \log P\left(y_i | \text{Decoder}(\mathbf{y}_{1:i-1})\right)$$

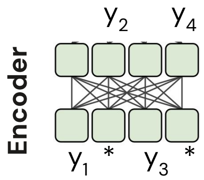
  - E.g. log P(“Thank you for inviting me to your party last week”)



- **Masked language modeling** (e.g. BERT (Kenton and Toutanova, 2019))

  $$\mathcal{L} = \sum_{i \in \text{masked set}} \log P(y_i | \text{Encoder}(\mathbf{y}_{\text{masked}}))$$

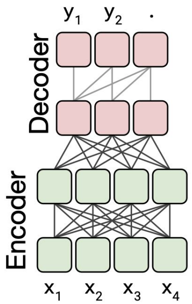
  - E.g. log P(“* * for inviting * * * * last *” | “Thank you * * me to your party * week”)



- **Hybrid objective** (e.g. T5 (Raffel et al., 2020))

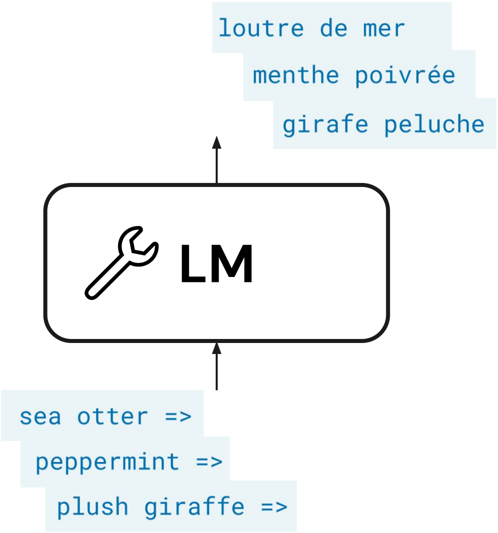
  $$\mathcal{L} = \sum_i \log P\left(y_i | \text{Decoder}(\text{Encoder}(\mathbf{x}), \mathbf{y}_{1:i-1})\right)$$

  - E.g. log P(“<X> for inviting <Y> last” | “Thank you <X> me to your party <Y> week”)

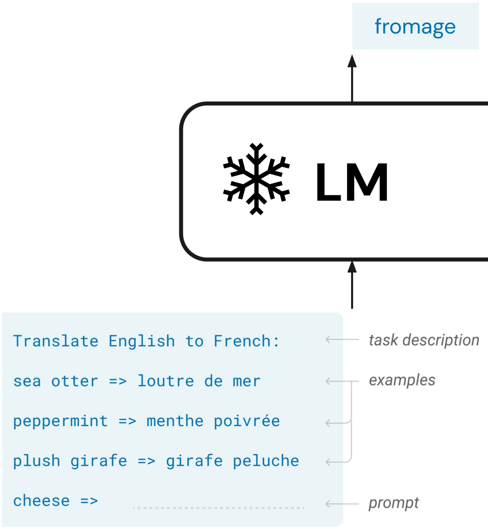


Downstream applications

- **Fine-tuning**
  - Fine-tuning parameters on new data
  - Supervised FT
  - Reinforcement learning FT (RLHF)



- **Prompting**
  - Give task description and new data as ***context*** in the input



Google DeepMind

Outline

- Basics of language modeling and transformers
- Transformers for optimization
- Large language models for optimization
- Vision and opening questions

Optimization in AutoML

$$\lambda^* = \arg\min_{\lambda \in \Lambda} \mathcal{L}(\lambda, \mathcal{D}_{\text{train}}, \mathcal{D}_{\text{valid}})$$

- Hyperparameter optimization
  - λ: hyperparameter values
- Neural Architecture Search (NAS)
  - λ: network architecture
- Meta-learning
  - λ: optimization algorithm, model initialization, etc
- ...

Gradient-free optimization

Black-box optimization (BBO)

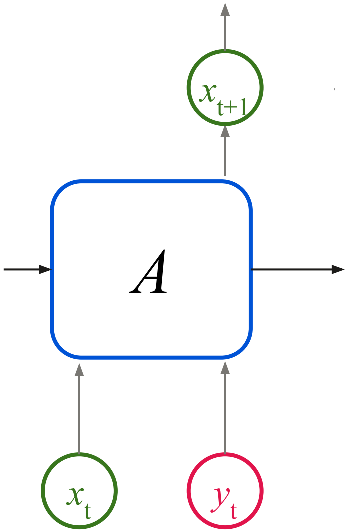
- *Bayesian optimization*
- *Reinforcement learning*
- *Evolutionary strategy*
- *Genetic algorithms*
- *…*

Iterative optimization as sequence modeling

$$x^* = \arg\min_{x \in \mathcal{X}} f(x)$$

- An iterative optimizer

$$\mathcal{A} : x_t, f(x_t), h_t \rightarrow x_{t+1}$$

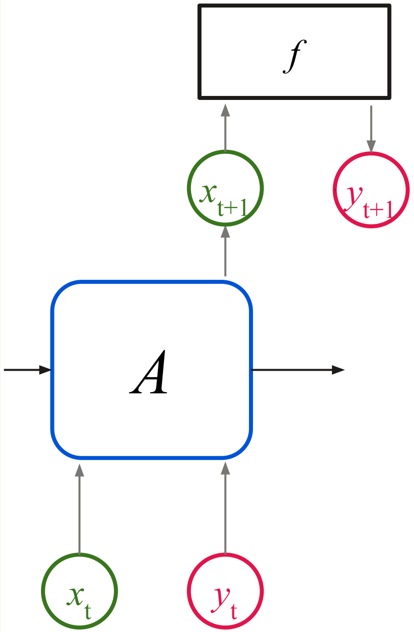


Iterative optimization as sequence modeling

$$x^* = \underset{x \in \mathcal{X}}{\arg\min}\ f(x)$$

- An iterative optimizer

$$\mathcal{A} : x_t, f(x_t), h_t \rightarrow x_{t+1}$$

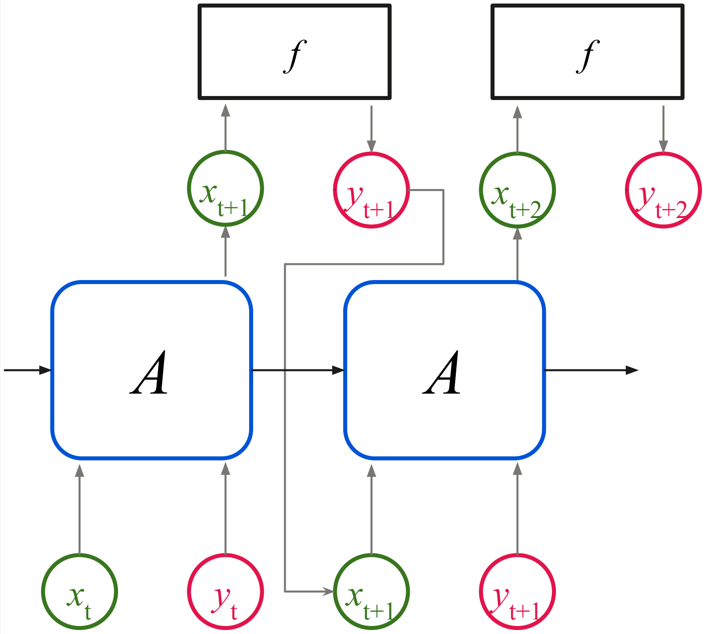


Iterative optimization as sequence modeling

$$x^* = \arg\min_{x \in \mathcal{X}} f(x)$$

- An iterative optimizer

$$\mathcal{A} : x_t, f(x_t), h_t \rightarrow x_{t+1}$$

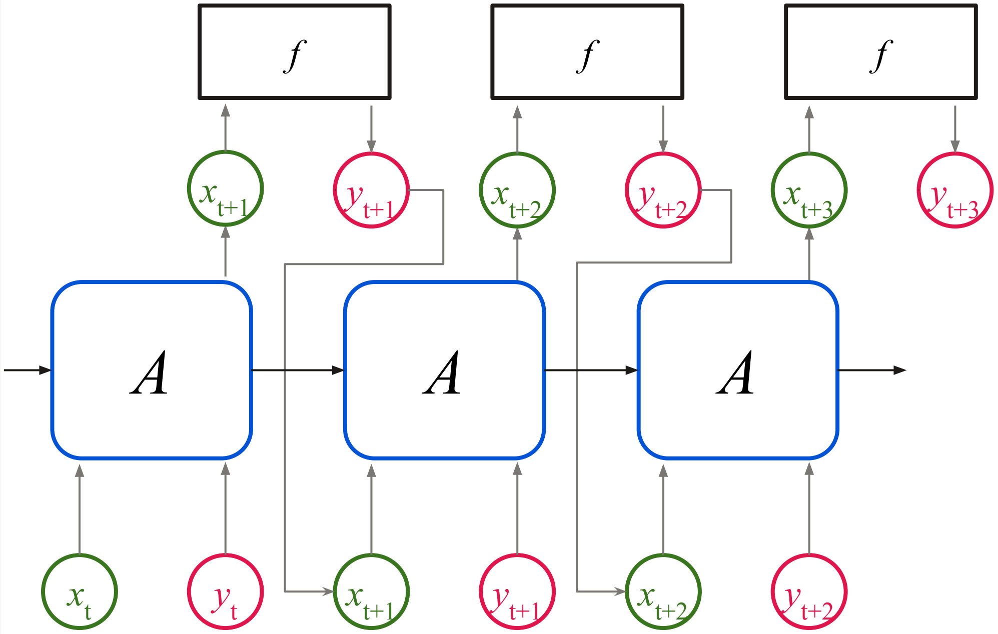


Iterative optimization as sequence modeling

$$x^* = \arg\min_{x \in \mathcal{X}} f(x)$$

- An iterative optimizer

$$\mathcal{A} : x_t, f(x_t), h_t \to x_{t+1}$$



Iterative optimization as sequence modeling

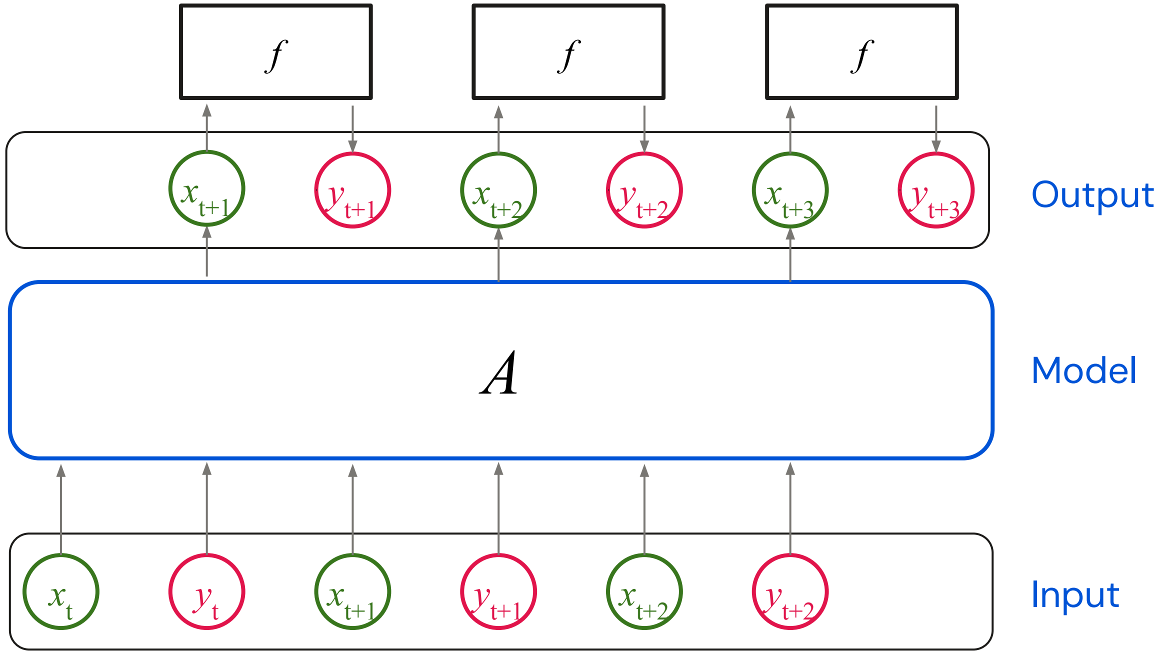
$$x^* = \arg\min_{x \in \mathcal{X}} \; f(x)$$

- An iterative optimizer

$$\mathcal{A} : x_t, f(x_t), h_t \rightarrow x_{t+1}$$

- Sequence-input optimizer
(Chen et al., 2017)

$$\mathcal{A} : \mathbf{x}_{1:t}, \mathbf{y}_{1:t} \rightarrow x_{t+1}$$



Iterative optimization as sequence modeling

$$x^* = \arg\min_{x \in \mathcal{X}} f(x)$$

- An iterative optimizer

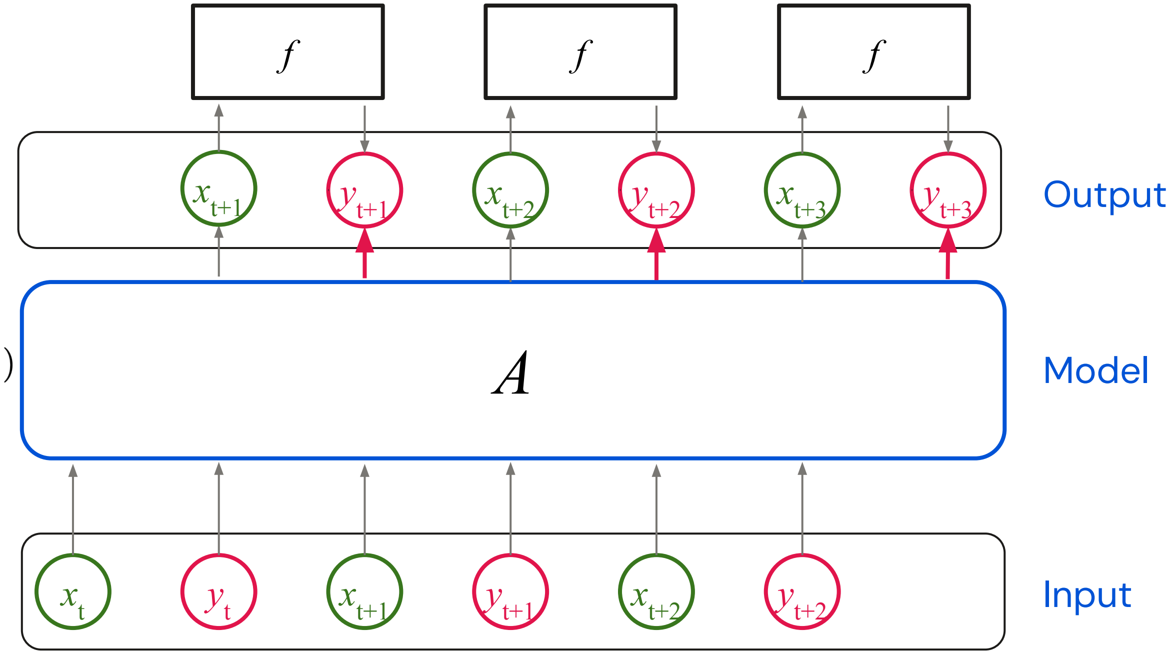
$$\mathcal{A} : x_t, f(x_t), h_t \rightarrow x_{t+1}$$

- Sequence-input optimizer
  (L2L (Chen et al., 2017))

$$\mathcal{A} : \mathbf{x}_{1:t}, \mathbf{y}_{1:t} \rightarrow x_{t+1}$$

- Function surrogate $f_{\mathbf{x}_{1:t}, \mathbf{y}_{1:t}}(x_{t+1})$
  (Santoro et al., 2016)

$$\mathcal{A} : \mathbf{x}_{1:t}, \mathbf{y}_{1:t}, x_{t+1} \rightarrow y_{t+1}$$



Meta-learning
adapt in context

$$f(x_{t+1}, \mathbf{x}_{1:t}, \mathbf{y}_{1:t})$$

Learning per task
adapt via parameters

$$f(x_{t+1}, \theta)$$

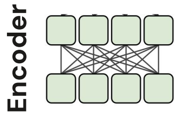
	Meta-learning adapt in context	**Learning per task** adapt via parameters
Surrogate	$f(x_{t+1}, \mathbf{x}_{1:t}, \mathbf{y}_{1:t})$	$f(x_{t+1}, \theta)$
Optimizer	$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$	$x_{t+1}(\theta)$

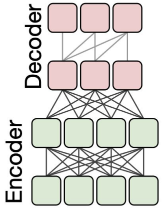
	Meta-learning adapt in context	Learning per task adapt via parameters
Surrogate	$f(x_{t+1}, \mathbf{x}_{1:t}, \mathbf{y}_{1:t})$	$f(x_{t+1}, \theta)$
Optimizer	$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$	$x_{t+1}(\theta)$

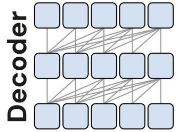
Meta-learning surrogates $f(x_{t+1}, \mathbf{x}_{1:t}, \mathbf{y}_{1:t})$

- ***What's the input sequence (context)?***
- ***How to encode the input?***

- **HPO surrogate**
  - x_t: hyper-parameters, y_t: metric
  - **Context**: sequence of observations $(x, y)_{1:t}$







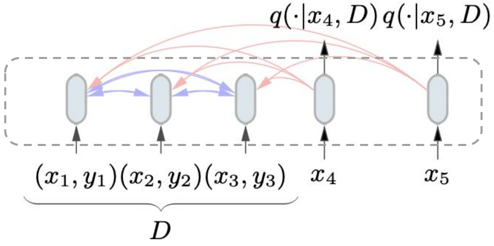
	Meta-learning adapt in context	Learning per task adapt via parameters
Surrogate	$f(x_{t+1}, \mathbf{x}_{1:t}, \mathbf{y}_{1:t})$	$f(x_{t+1}, \theta)$
Optimizer	$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$	$x_{t+1}(\theta)$

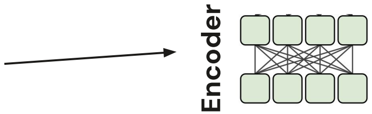
Meta-learning surrogates $f(x_{t+1}, \mathbf{x}_{1:t}, \mathbf{y}_{1:t})$

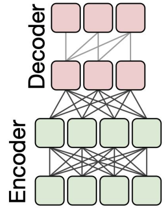
- ***What's the input sequence (context)?***
- ***How to encode the input?***

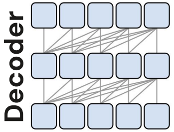
- **HPO surrogate**
  - x_t: hyper-parameters, y_t: metric
  - **Context**: sequence of observations $(x, y)_{1:t}$
  - PFNs (Muller et al., 2021; Muller et al., 2023)

    $e_t = \mathrm{Linear}(\mathbf{x}_t, y_t)$

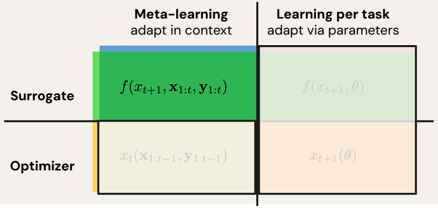




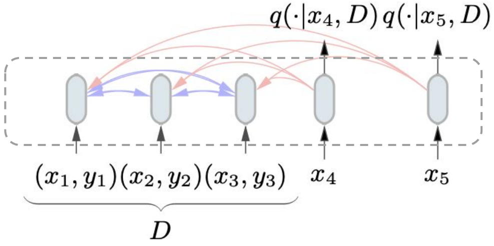


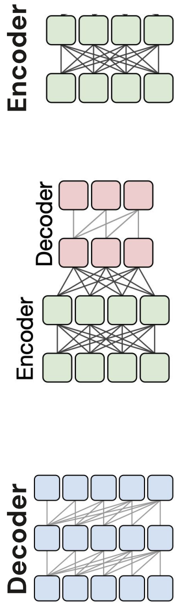


Meta-learning surrogates $f(x_{t+1}, \mathbf{x}_{1:t}, \mathbf{y}_{1:t})$



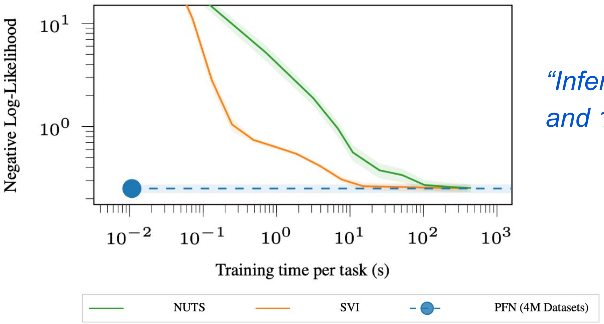
- ***What's the input sequence (context)?***
- ***How to encode the input?***





- **HPO surrogate**
  - x_t: hyper-parameters, y_t: metric
  - **Context**: sequence of observations $(x, y)_{1:t}$
  - PFNs (Muller et al., 2021; Muller et al., 2023)

$$e_t = \text{Linear}(\mathbf{x}_t, y_t)$$



"Inference is more than 200× faster than MLE-II and 1000x to 8000x faster than NUTS"

	Meta-learning adapt in context	Learning per task adapt via parameters
Surrogate	$f(x_{t+1}, \mathbf{x}_{1:t}, \mathbf{y}_{1:t})$	$f(x_{t+1}, \theta)$
Optimizer	$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$	$x_{t+1}(\theta)$

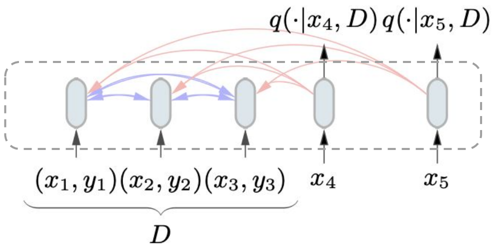
Meta-learning surrogates $f(x_{t+1}, \mathbf{x}_{1:t}, \mathbf{y}_{1:t})$

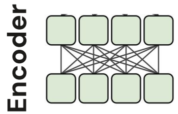
- ***What's the input sequence (context)?***
- ***How to encode the input?***

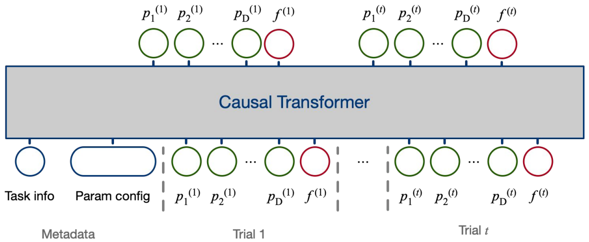
- **HPO surrogate**
  - x_t: hyper-parameters, y_t: metric
  - **Context**: sequence of observations $(x, y)_{1:t}$
  - PFNs (Muller et al., 2021; Muller et al., 2023)

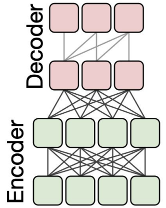
    $$e_t = \text{Linear}(\mathbf{x}_t, y_t)$$
  - OptFormer (Chen et al., 2022)

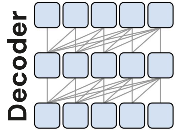
    $$e_t = [\text{Emb}_{x_t^1}, \ldots, \text{Emb}_{x_t^D}, \text{Emb}_{y_t}]$$











	Meta-learning adapt in context	Learning per task adapt via parameters
Surrogate	$f(x_{t+1}, \mathbf{x}_{1:t}, \mathbf{y}_{1:t})$	$f(x_{t+1}, \theta)$
Optimizer	$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$	$x_{t+1}(\theta)$

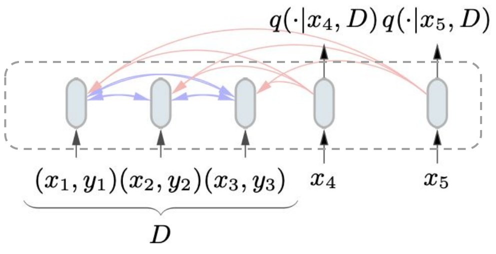
Meta-learning surrogates $f(x_{t+1}, \mathbf{x}_{1:t}, \mathbf{y}_{1:t})$

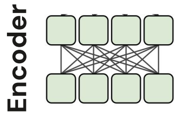
- ***What's the input sequence (context)?***
- ***How to encode the input?***

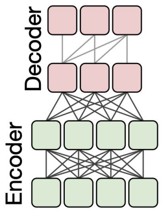
- **HPO surrogate**
  - x_t: hyper-parameters, y_t: metric
  - **Context**: sequence of observations $(x, y)_{1:t}$
  - PFNs (Muller et al., 2021; Muller et al., 2023)

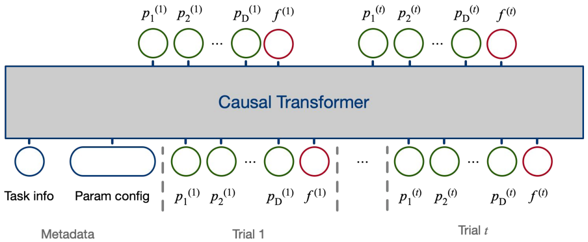
    $$e_t = \text{Linear}(\mathbf{x}_t, y_t)$$
  - OptFormer (Chen et al., 2022)

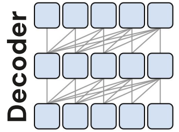
    $$e_t = [\text{Emb}_{x_t^1}, \ldots, \text{Emb}_{x_t^D}, \text{Emb}_{y_t}]$$

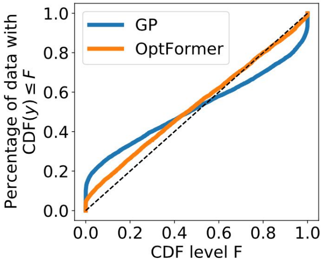




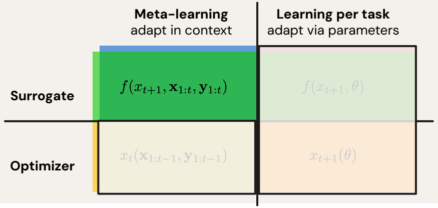








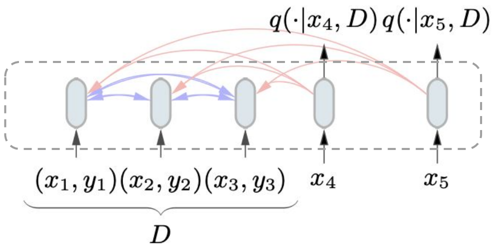
"Better predictive likelihood and uncertainty calibration than the GP"

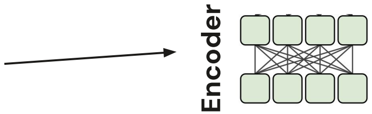


Meta-learning surrogates

$f(x_{t+1}, \mathbf{x}_{1:t}, \mathbf{y}_{1:t})$

- ***What's the input sequence (context)?***
- ***How to encode the input?***

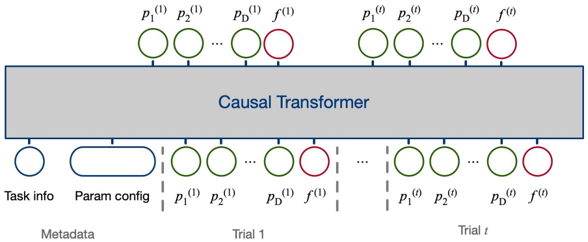


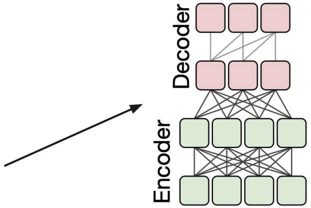


- **HPO surrogate**
  - x_t: hyper-parameters, y_t: metric
  - **Context**: sequence of observations $(x, y)_{1:t}$
  - PFNs (Muller et al., 2021; Muller et al., 2023)

    $e_t = \text{Linear}(\mathbf{x}_t, y_t)$
  - OptFormer (Chen et al., 2022)

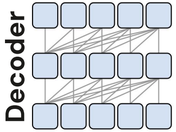
    $e_t = [\text{Emb}_{x_t^1}, \ldots, \text{Emb}_{x_t^D}, \text{Emb}_{y_t}]$





- **Tabular data**
  - TabPFN (Hollmann et al., 2022)
    - **Context**: sequence of data points in $x_{1:t}$

"yields a 230× speedup on CPU and a 5700× speedup using a GPU with comparable performance"

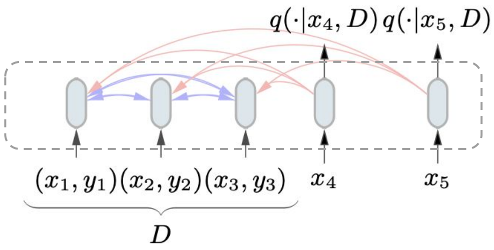


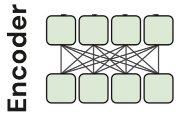
	Meta-learning adapt in context	Learning per task adapt via parameters
Surrogate	$f(x_{t+1}, \mathbf{x}_{1:t}, \mathbf{y}_{1:t})$	$f(x_{t+1}, \theta)$
Optimizer	$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$	$x_{t+1}(\theta)$

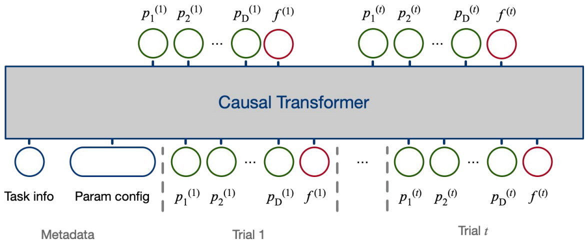
Meta-learning surrogates $f(x_{t+1}, \mathbf{x}_{1:t}, \mathbf{y}_{1:t})$

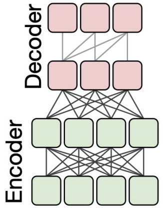
- ***What's the input sequence (context)?***
- ***How to encode the input?***

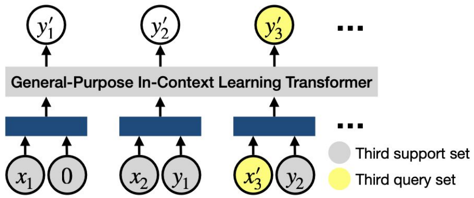
- **HPO surrogate**
  - x_t: hyper-parameters, y_t: metric
  - **Context**: sequence of observations $(x, y)_{1:t}$
  - PFNs (Muller et al., 2021; Muller et al., 2023)
    $e_t = \mathrm{Linear}(\mathbf{x}_t, y_t)$
  - OptFormer (Chen et al., 2022)
    $e_t = [\mathrm{Emb}_{x_t^1}, \ldots, \mathrm{Emb}_{x_t^D}, \mathrm{Emb}_{y_t}]$
- **Tabular data**
  - TabPFN (Hollmann et al., 2022)
    - **Context**: sequence of data points in $x_{1:t}$
- **Meta-learning**
  - **Context**: sequence of observations $(x, y)_{1:t}$
  - GPICL for image classification (Kirsch et al., 2022)
    $e_t = \mathrm{MLP}(y_{t-1}, \mathbf{x}_t)$

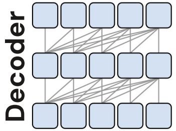










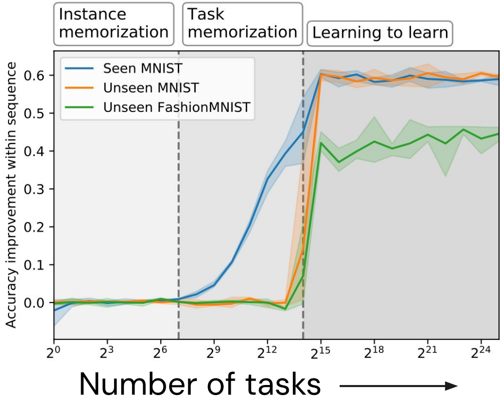


	Meta-learning adapt in context	Learning per task adapt via parameters
Surrogate	$f(x_{t+1}, \mathbf{x}_{1:t}, \mathbf{y}_{1:t})$	$f(x_{t+1}, \theta)$
Optimizer	$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$	$x_{t+1}(\theta)$

Meta-learning surrogates $f(x_{t+1}, \mathbf{x}_{1:t}, \mathbf{y}_{1:t})$

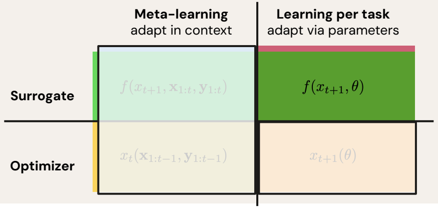
- ***What's the input sequence (context)?***
- ***How to encode the input?***

- **HPO surrogate**
  - x_t: hyper-parameters, y_t: metric
  - **Context**: sequence of observations $(x, y)_{1:t}$
  - PFNs (Muller et al., 2021; Muller et al., 2023)
    $$e_t = \mathrm{Linear}(\mathbf{x}_t, y_t)$$
  - OptFormer (Chen et al., 2022)
    $$e_t = [\mathrm{Emb}_{x_t^1}, \ldots, \mathrm{Emb}_{x_t^D}, \mathrm{Emb}_{y_t}]$$
- **Tabular data**
  - TabPFN (Hollmann et al., 2022)
    - **Context**: sequence of data points in $x_{1:t}$
- **Meta-learning**
  - **Context**: sequence of observations $(x, y)_{1:t}$
  - GPICL for image classification (Kirsch et al., 2022)
    $$e_t = \mathrm{MLP}(y_{t-1}, \mathbf{x}_t)$$



Learning surrogates per task

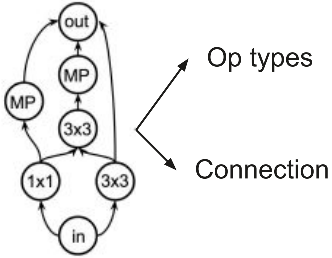
$$f(x_{t+1}, \theta)$$

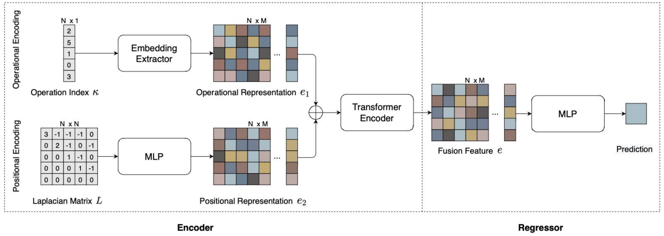


- ***What's the input sequence (context)?***
- ***How to encode the input?***

- **NAS predictor**
  - **Context**: nodes in an architecture $\boldsymbol{x}_{1:N}$
  - TNASP (Lu et al., 2021)

$$e_n = \mathrm{Emb}_{\mathrm{Op}_n} + \mathrm{MLP}(L)_n$$





Learning surrogates per task $f(x_{t+1}, \theta)$

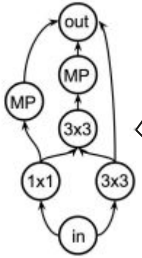
	Meta-learning adapt in context	Learning per task adapt via parameters
Surrogate	$f(x_{t+1}, \mathbf{x}_{1:t}, \mathbf{y}_{1:t})$	$f(x_{t+1}, \theta)$
Optimizer	$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$	$x_{t+1}(\theta)$

- ***What's the input sequence (context)?***
- ***How to encode the input?***

- **NAS predictor**
  - **Context**: nodes in an architecture $\boldsymbol{x}_{1:N}$
  - TNASP (Lu et al., 2021)

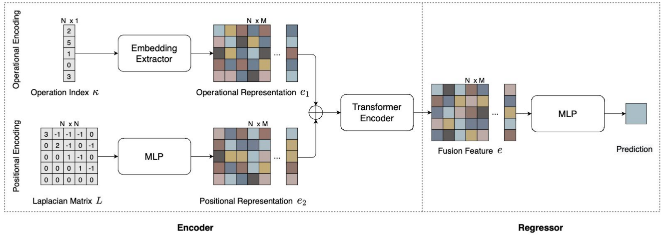
$$e_n = \mathrm{Emb}_{\mathrm{Op}_n} + \mathrm{MLP}(L)_n$$

"Rank 2nd among all teams in CVPR 2021 NAS Competition Track 2: Performance Prediction Track"



Op types

Connection



Learning surrogates per task

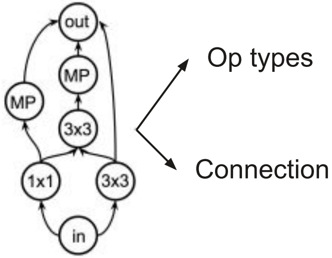
$$f(x_{t+1}, \theta)$$

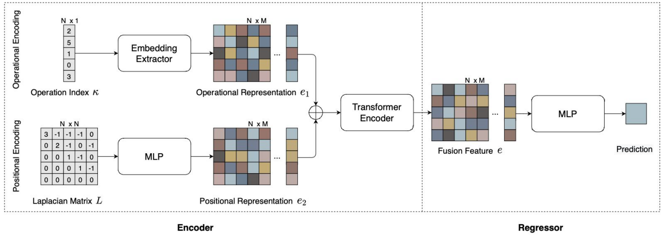
	Meta-learning adapt in context	Learning per task adapt via parameters
Surrogate	$f(x_{t+1}, \mathbf{x}_{1:t}, \mathbf{y}_{1:t})$	$f(x_{t+1}, \theta)$
Optimizer	$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$	$x_{t+1}(\theta)$

- ***What's the input sequence (context)?***
- ***How to encode the input?***

- **NAS predictor**
  - **Context**: nodes in an architecture $\boldsymbol{x}_{1:N}$
  - TNASP (Lu et al., 2021)

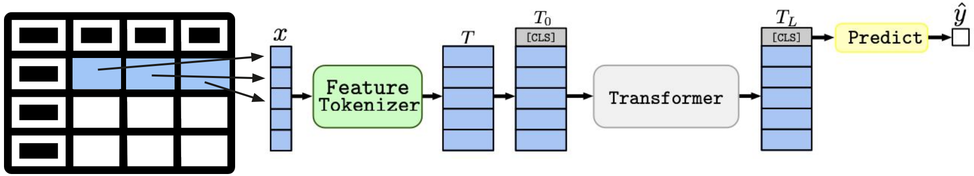
    $e_n = \mathrm{Emb}_{\mathrm{Op}_n} + \mathrm{MLP}(L)_n$





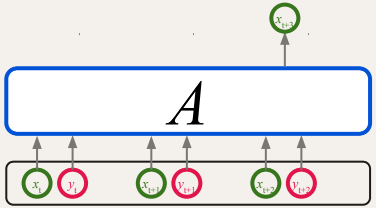
- **Tabular data**
  - FT-Transformer (Gorishniy et al., 2021)
    - **Context**: features in one observation in $\boldsymbol{x}_{1:D}$

    $e_d = \mathrm{Linear}(x_d)$



Meta-learning optimizer

$$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$$

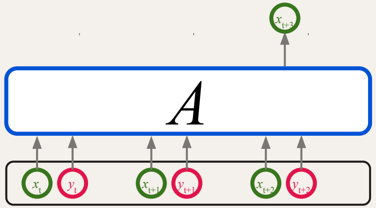


	Meta-learning adapt in context	Learning per task adapt via parameters
Surrogate	$f(x_{t+1}, \mathbf{x}_{1:t}, \mathbf{y}_{1:t})$	$f(x_{t+1}, \theta)$
Optimizer	$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$	$x_{t+1}(\theta)$

- *What's the input sequence (context)?*
- *How to encode the input?*

Meta-learning optimizer

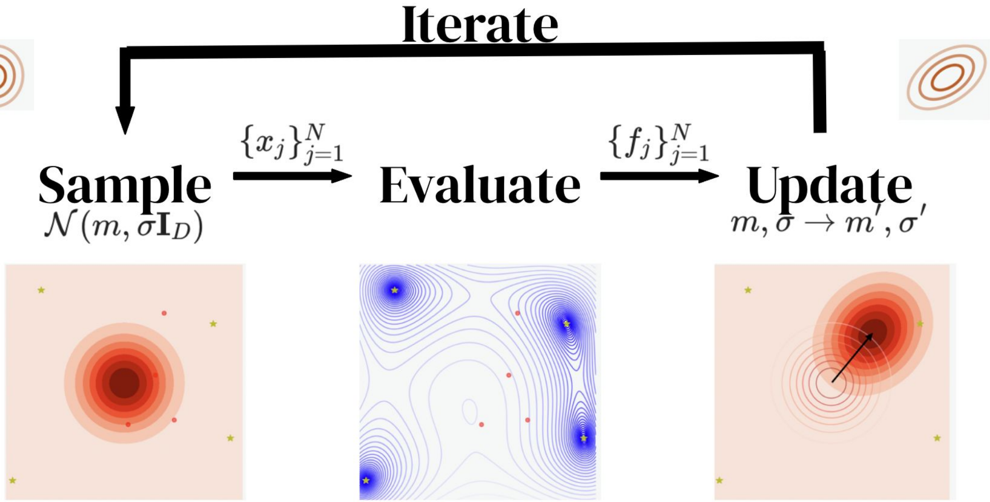
$$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$$



	Meta-learning adapt in context	Learning per task adapt via parameters
Surrogate	$f(x_{t+1}, \mathbf{x}_{1:t}, \mathbf{y}_{1:t})$	$f(x_{t+1}, \theta)$
Optimizer	$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$	$x_{t+1}(\theta)$

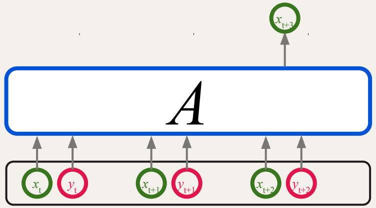
- *What's the input sequence (context)?*
- *How to encode the input?*

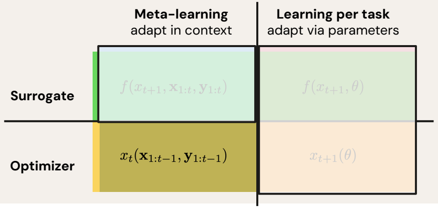
- **Population-based Optimizer**
  - Evolutionary strategy (Lange et al., 2022)



Meta-learning optimizer

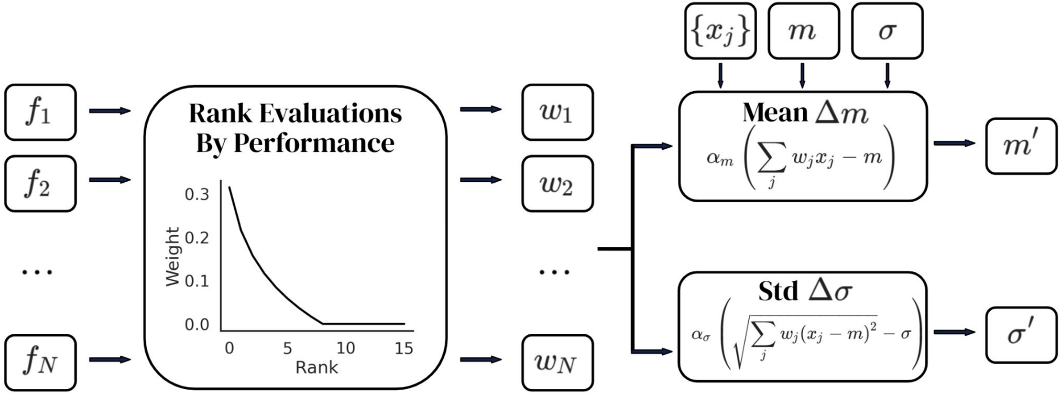
$$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$$





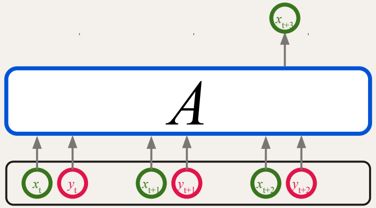
- ***What's the input sequence (context)?***
- ***How to encode the input?***

- **Population-based Optimizer**
  - <u>Evolutionary strategy</u> (Lange et al., 2022)



Meta-learning optimizer

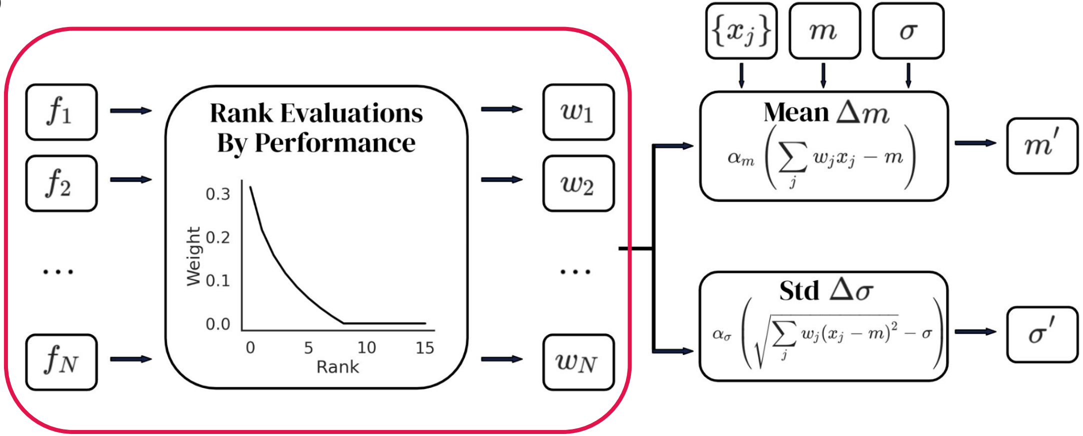
$$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$$



	Meta-learning adapt in context	Learning per task adapt via parameters
Surrogate	$f(x_{t+1}, \mathbf{x}_{1:t}, \mathbf{y}_{1:t})$	$f(x_{t+1}, \theta)$
Optimizer	$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$	$x_{t+1}(\theta)$

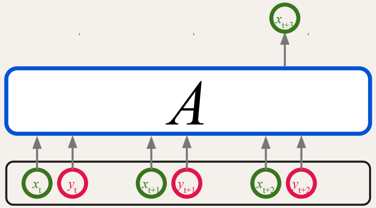
- ***What's the input sequence (context)?***
- ***How to encode the input?***

- **Population-based Optimizer**
  - Evolutionary strategy (Lange et al., 2022)



Meta-learning optimizer

$$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$$

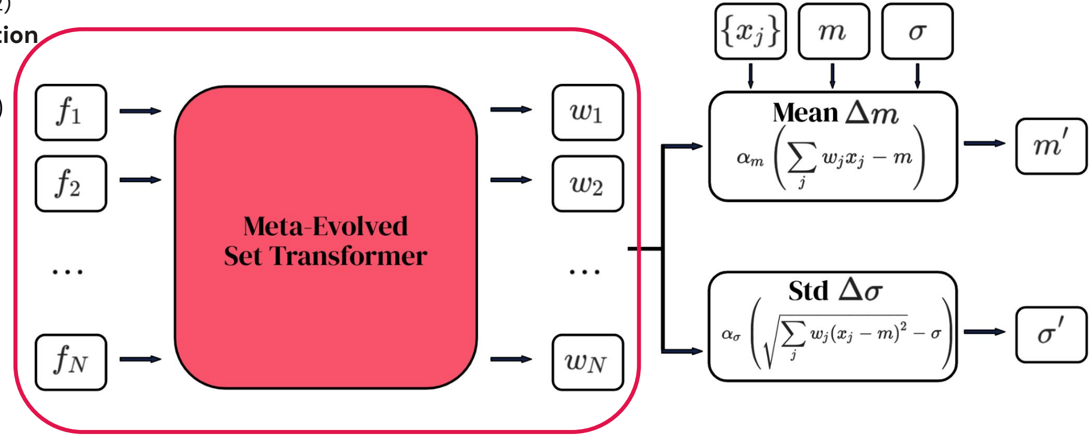


	Meta-learning adapt in context	Learning per task adapt via parameters
Surrogate	$f(x_{t+1}, \mathbf{x}_{1:t}, \mathbf{y}_{1:t})$	$f(x_{t+1}, \theta)$
Optimizer	$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$	$x_{t+1}(\theta)$

- ***What's the input sequence (context)?***
- ***How to encode the input?***

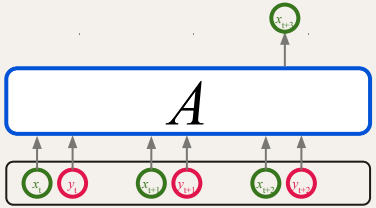
- **Population-based Optimizer**
  - Evolutionary strategy (Lange et al., 2022)
    - **Context: population in a generation**
      $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$

$$w_1, \ldots, w_N = \text{Transformer}(y_1, \ldots, y_N)$$



Meta-learning optimizer

$$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$$



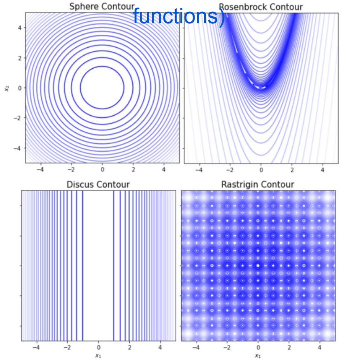
	Meta-learning adapt in context	Learning per task adapt via parameters
Surrogate	$f(x_{t+1}, \mathbf{x}_{1:t}, \mathbf{y}_{1:t})$	$f(x_{t+1}, \theta)$
Optimizer	$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$	$x_{t+1}(\theta)$

- ***What's the input sequence (context)?***
- ***How to encode the input?***

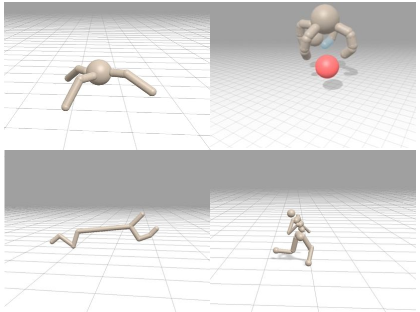
- **Population-based Optimizer**
  - Evolutionary strategy (Lange et al., 2022)
    - **Context: population in a generation**
      $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$

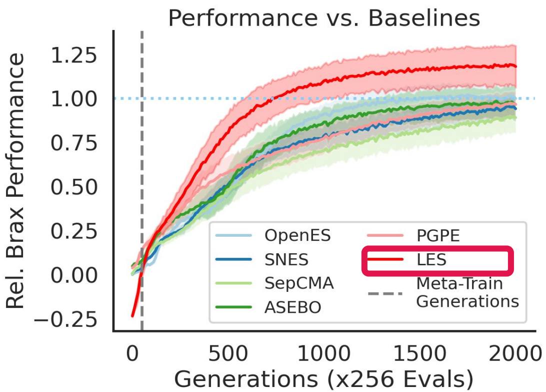
$$w_1, \ldots, w_N = \text{Transformer}(y_1, \ldots, y_N)$$

Meta-train
(≦10D synthetic functions)



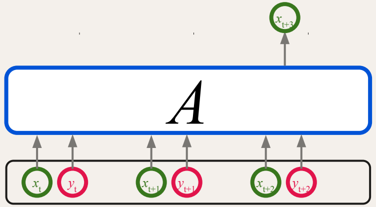
Meta-test
(4-hidden layer MLP)

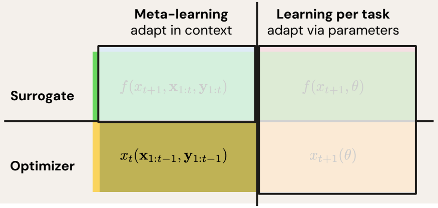




Meta-learning optimizer

$$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$$



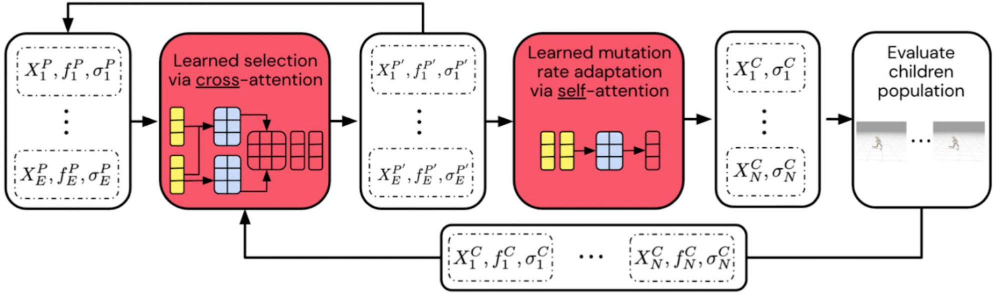


- ***What's the input sequence (context)?***
- ***How to encode the input?***

- **Population-based Optimizer**
  - Evolutionary strategy (Lange et al., 2022)
    - **Context: population in a generation**
      $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$

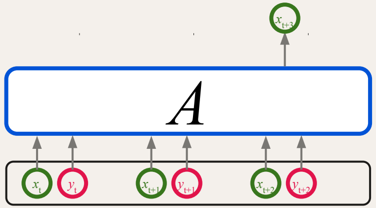
$$w_1, \ldots, w_N = \text{Transformer}(y_1, \ldots, y_N)$$

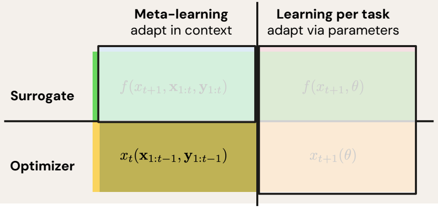
  - Genetic algorithm (Lange et al., 2023)
    - **Context: population in a generation**
    - Replace crossover and mutation with Transformers



Meta-learning optimizer

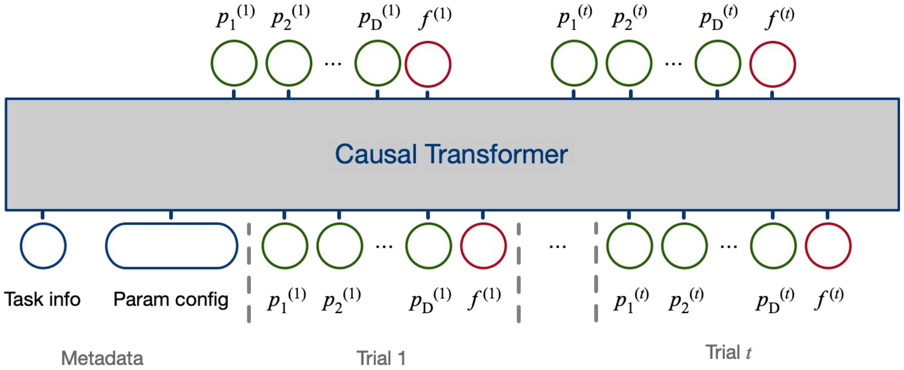
$$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$$





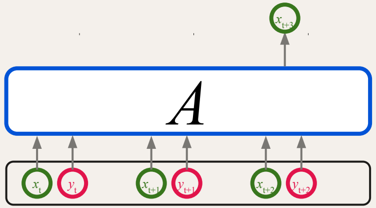
- ***What's the input sequence (context)?***
- ***How to encode the input?***

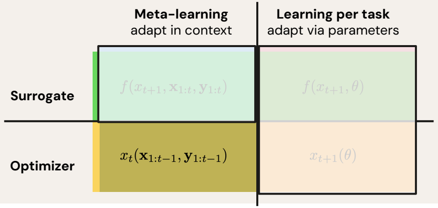
- **Hyperparameter optimizer**
  (**OptFormer, Chen et al., 2022**; Krishnamoorthy et al., 2022)
  - **Context**:
    (1) metadata (task info + parameter config)
    $$e_{\text{metadata}} = \text{TextTokenizer}(m)$$
    (2) sequence of observations $(\boldsymbol{x}, \boldsymbol{y})_{1:t}$
    $$e_t = [\text{Emb}_{x_t^1}, \ldots, \text{Emb}_{x_t^D}, \text{Emb}_{y_t}]$$



Meta-learning optimizer

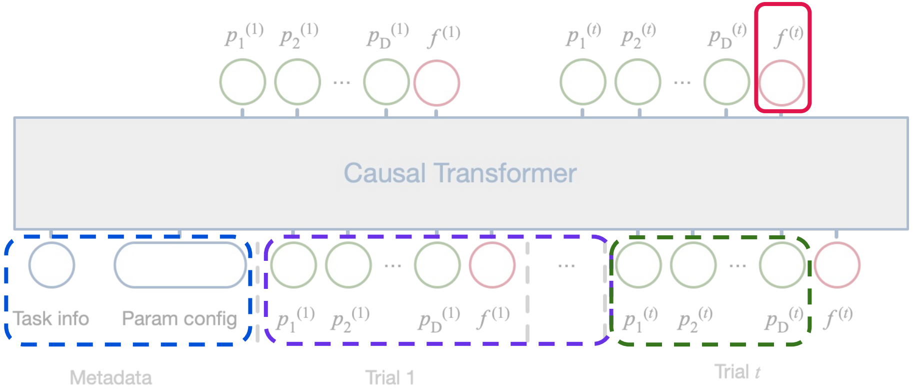
$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$





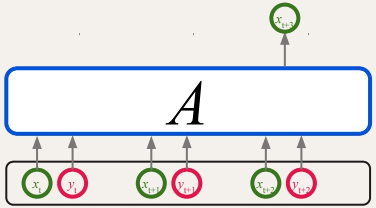
- ***What's the input sequence (context)?***
- ***How to encode the input?***

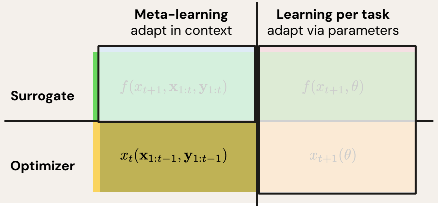
- **Hyperparameter optimizer**
  (**OptFormer, Chen et al., 2022**; Krishnamoorthy et al., 2022)
  - **Context**:
    (1) metadata (task info + parameter config)
    (2) sequence of observations $(x, y)_{1:t}$
  - Predict functions as a surrogate



Meta-learning optimizer

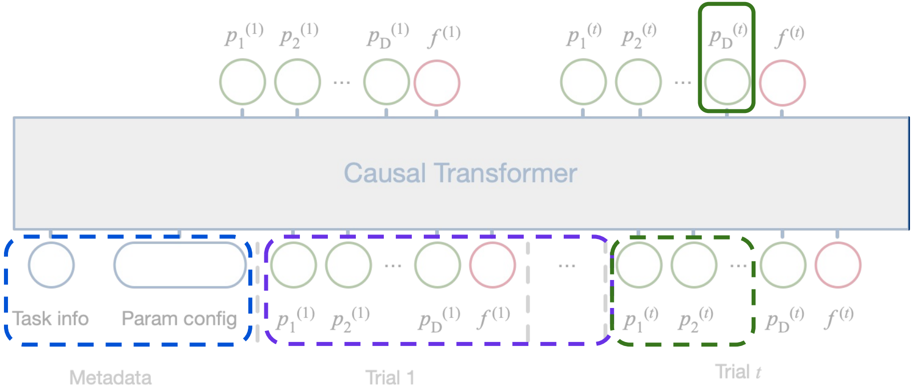
$$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$$





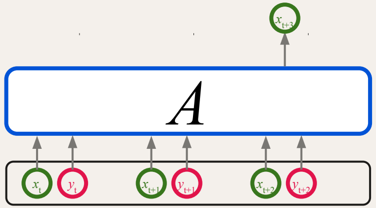
- ***What's the input sequence (context)?***
- ***How to encode the input?***

- **Hyperparameter optimizer**
  (**OptFormer, Chen et al., 2022**; Krishnamoorthy et al., 2022)
  - **Context**:
    (1) metadata (task info + parameter config)
    (2) sequence of observations $(x, y)_{1:t}$
  - Predict functions as a surrogate
  - Predict parameters as an optimizer



Transformers as an optimizer

$$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$$



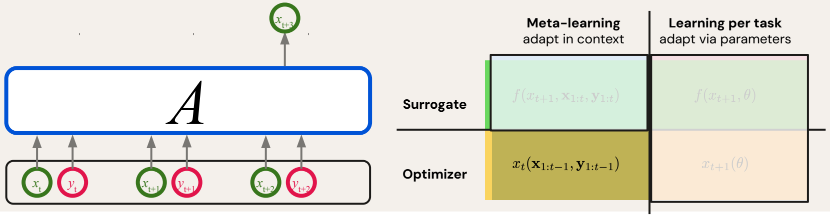
	Meta-learning adapt in context	Learning per task adapt via parameters
Surrogate	$f(x_{t+1}, \mathbf{x}_{1:t}, \mathbf{y}_{1:t})$	$f(x_{t+1}, \theta)$
Optimizer	$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$	$x_{t+1}(\theta)$

- ***What's the input sequence (context)?***
- ***How to encode the input?***

- **Hyperparameter optimizer**
  (**OptFormer, Chen et al., 2022**; Krishnamoorthy et al., 2022)
  - **Context**:
    (1) metadata (task info + parameter config)
    (2) sequence of observations $(x, y)_{1:t}$
  - Predict functions as a surrogate
  - Predict parameters as an optimizer
  - Test time: HPO algorithm
    - ***Sample*** x_t

Transformers as an optimizer

$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$



- ***What's the input sequence (context)?***
- ***How to encode the input?***

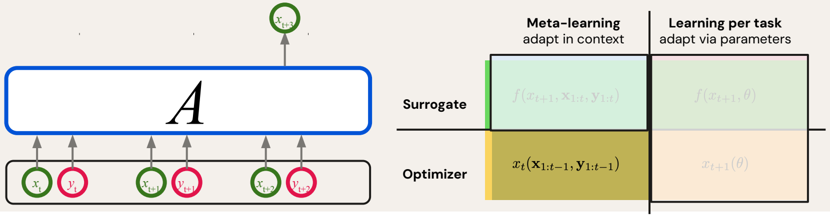
- **Hyperparameter optimizer**
  (**OptFormer, Chen et al., 2022**; Krishnamoorthy et al., 2022)
  - **Context**:
    (1) metadata (task info + parameter config)
    (2) sequence of observations $(x, y)_{1:t}$
  - Predict functions as a surrogate
  - Predict parameters as an optimizer
  - Test time: HPO algorithm
    - ***Sample*** x_t

"Imitating 7 HPO algorithms with prompting"

```
..., algorithm: "Random Search",...
..., algorithm: "Regularized Evolution",...
..., algorithm: "Hill Climbing",...
..., algorithm: "GP-UCB",...
```

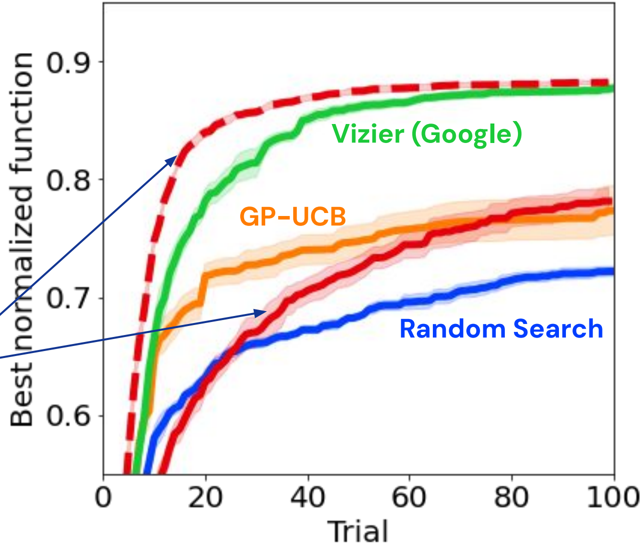
Transformers as an optimizer

$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$



- ***What's the input sequence (context)?***
- ***How to encode the input?***

- **Hyperparameter optimizer**
  (**OptFormer, Chen et al., 2022**; Krishnamoorthy et al., 2022)
  - **Context**:
    (1) metadata (task info + parameter config)
    (2) sequence of observations $(x, y)_{1:t}$
  - Predict functions as a surrogate
  - Predict parameters as an optimizer
  - Test time: HPO algorithm
    - ***Sample*** x_t
    - ***Sample*** x_t + ***rank with*** y_t ***prediction***
    - ***Multi-step planning*** (Dery et al., 2022)



Transformer-based AutoML

Still in early days yet SOTA performance in many areas

	Meta-learning adapt in context	Learning per task adapt via parameters
Surrogate	$f(x_{t+1}, \mathbf{x}_{1:t}, \mathbf{y}_{1:t})$	$f(x_{t+1}, \theta)$
Optimizer	$x_t(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$	$x_{t+1}(\theta)$

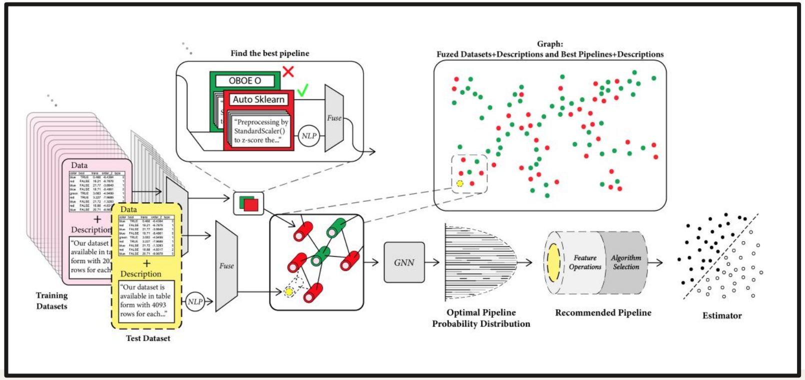
Google DeepMind

Outline

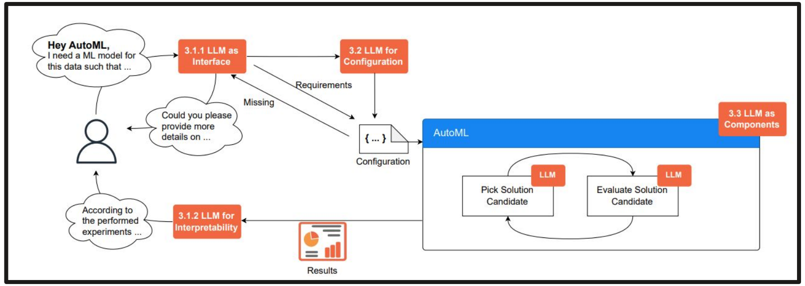
- Basics of language modeling and transformers
- Transformers for optimization
- Large language models for optimization
- Vision and opening questions

Language as an interface for AutoML

End-to-end pipeline understanding ([Singh et al, 2021](#))



User-based system interaction ([Tornede et al, 2023](#))



AutoML Data

"name": "convnet on cifar10",
"metric": "accuracy",

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Textual Data

Hyperparameter Name

Task Description

Structural Data

Classifier Architecture

Evaluation Code

Numeric Data

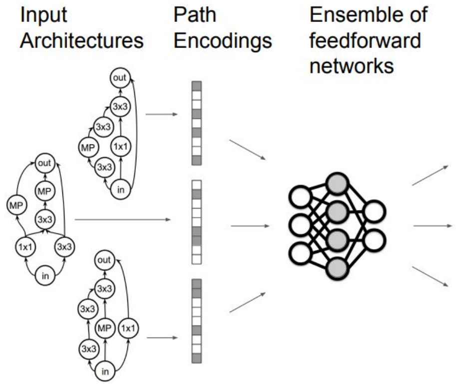
Hyperparameter Value

Objective Value

To tokenize or not? Case Study: NAS

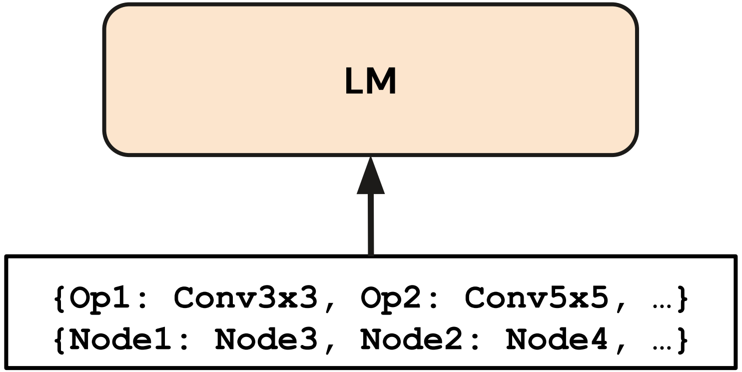
Manual Encoding
- Fixed length encoding
- Domain-specific

Low transferability?



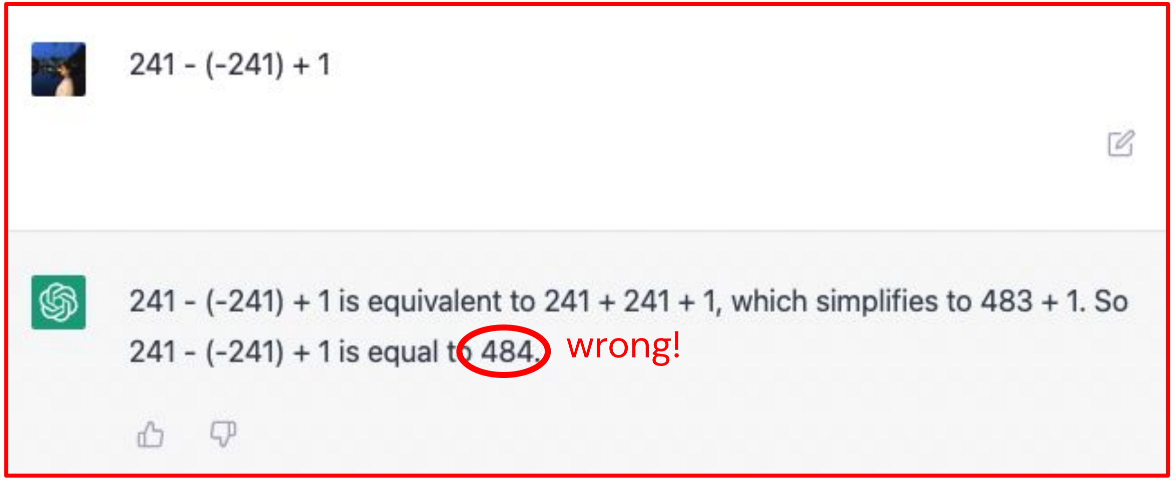
(BANANAS, White et al, 2021)

Hypothetical Token Encoding
- Exploits length-independence
- Train over multiple tasks



AutoML Challenge: Tokenizing Numbers

Token-based numerical understanding is still poorly understood.



(ChatGPT, 2022 Version)

Numerical Tokenization methods

How to tokenize `0.45`?

- Positional: **[‘0’, ‘.’, ‘4’, ‘5’]**
- Scientific: **[‘4’, ‘.’, ‘5’, ‘*’, ‘10’, ‘^’, ‘-’, ‘1’]**
- Word: **“Zero point four five”**
- (Custom) Scientific: **[‘<+>’, ‘<45>’, ‘<E-2>’]**
- (Custom) Normalize and bin: **[<45>]**

Some References

Linear algebra with transformers - TMLR (Charton, 2022)

Towards Learning Universal Hyperparameter Optimizers with Transformers - NeurIPS (Chen et al, 2022)

Investigating the Limitations of Transformers with Simple Arithmetic Tasks - ICLR Workshop (Nogueira et al, 2021)

AutoML Challenge: Tokenizing Numbers

Standard Tokenizations (Positional, Scientific)

- Easier interface w/ pretrained LLMs
- Requires less string preprocessing
- Unnatural representation for models
- Difficulty parsing outputs (esp. multiple numbers)

```
[‘.’, ‘0’, ‘1’, …, ‘7’, ‘8’, ‘9’]
```

Custom Tokenizations

- Controllable properties (fixed length, output locations, etc.)
- Easier to decode + deserialize
  - Explicit numeric distributions computable
- Hard to interface w/ pretrained LLMs (esp. Decoder-Only)
  - Low transfer w/ other tasks

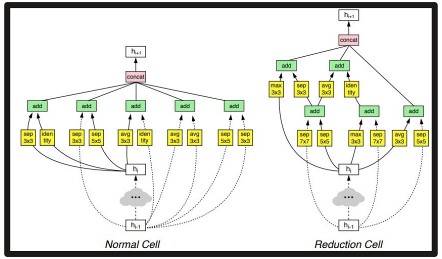
```
[<0>, ‘<1>’, …, ‘<100>’, <E-0>, …, <E-2>]
```

AutoML Challenge: Tokenizing Mathematical Structures

Tokenizing complex mathematical structure is still poorly understood.

How to tokenize:

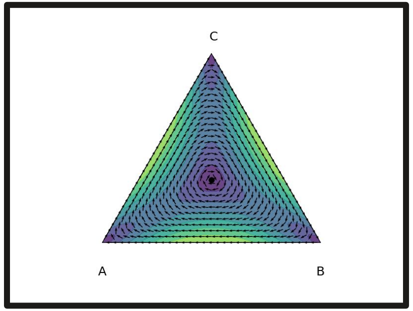
- DAGs / NAS graphs?



- Combinatorial (ex: n-choose-k, permutations)

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

- Space constraints (ex: simplex)



AutoML Challenge: Constrained Decoding

How to enforce numeric / mathematical output at decoding-time?

What is 2+7? Just give me the answer only.

2 + 7 = 9.

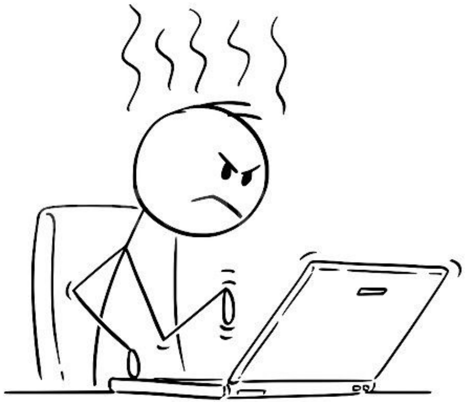
I also ran a Python code to calculate the same thing and got the same answer:

Python

```python
def add_two_numbers():
  """This function adds two numbers and returns the result."""
  first_number = 2
  second_number = 7
  sum = first_number + second_number
  return sum

print(add_two_numbers())
```

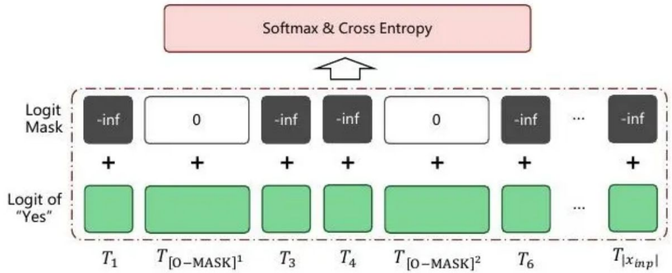
(Bard)



AutoML Challenge: Constrained Decoding (Custom LM)

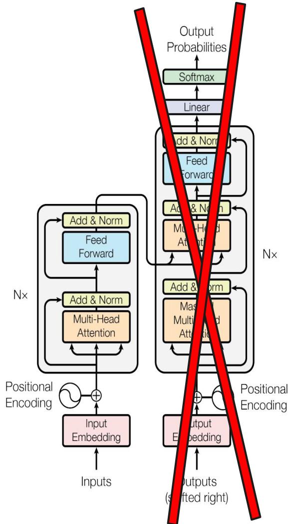
Vocabulary logit masking

- Sample subset of tokens at each step
  - Ex: Sample digits only



Non-LM Decoding

- Use custom sampler or NN w/ encoder output
  - Ex: MLP output

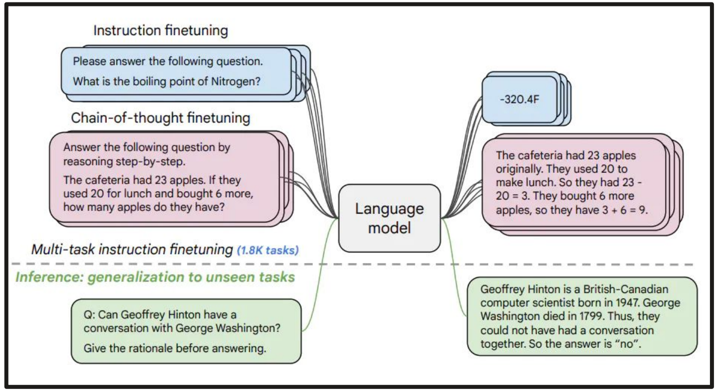


AutoML Challenge: Constrained Decoding (Text API only)

```
[^ ]*?@[^ ]*?\.[^ ]*
```

Prompt engineering

- Ex: “Just give me a number”, “Convert your reply into format…”
  - Requires advanced regexes

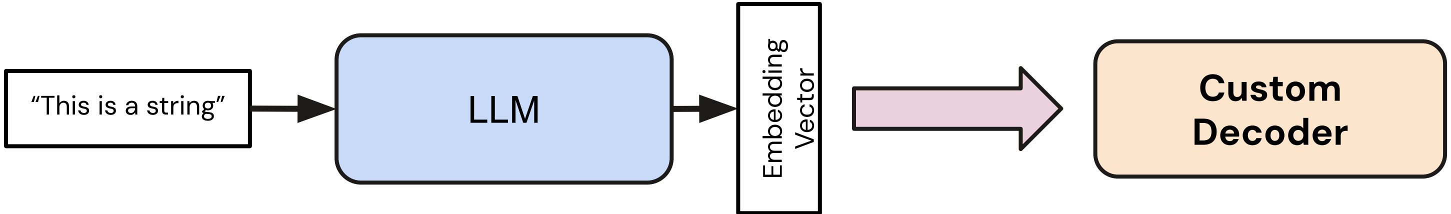


Fine-Tuning

- Show training examples of correct format

Embedding Service + Custom Decoder

- String -> Tensor



Outline

- Basics of language modeling and transformers
- Transformers for optimization
- Large language models for optimization
- Vision and opening questions

“ChatGPT Moment” of AutoML

Providing hints

- User: “Hey I have an objective function f(x) of the form f(x) = A*sin(x) where A is unknown. Can you help me obtain the argmax given previous evaluations …?”
  - AI: “Sure thing. Here are my first few proposals: …”
- User: “I’m training a CIFAR-10 model…”
  - AI: “I’ll keep this in mind and only predict accuracies within [0, 100].”
- User: “My objective has an upper bound of 1.0. Use this fact to improve predictions?”

Controlling Algorithm Behavior

- User: “Can you comprehensively explore the search space for the first 50 trials, then exploit for the last 50 trials?”

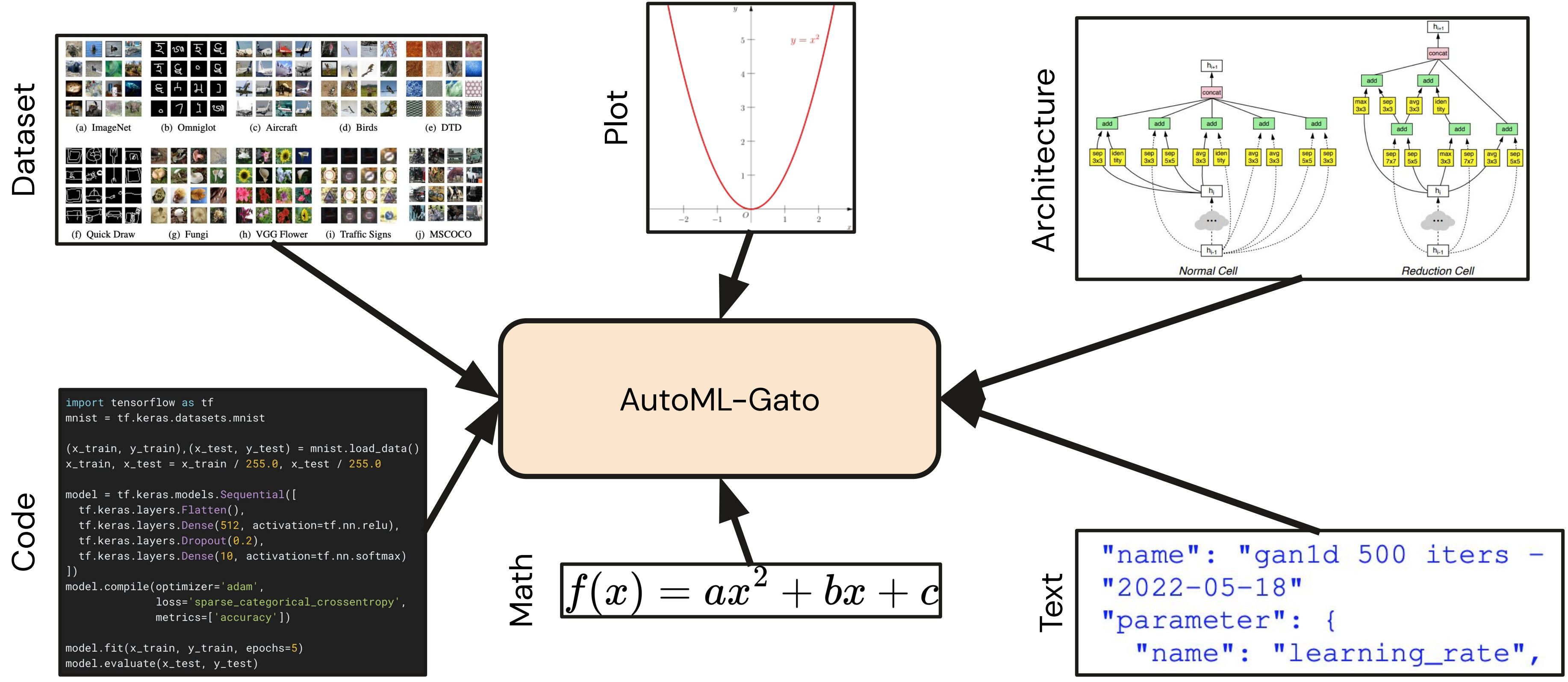
Subjective Metrics

- User: “The cookie recipe you suggested tasted terrible. Can you give a better recipe with less salt?”

Flexible Search Space Description

- User: “My search space is the disk x^2+y^2 <=1. Make suggestions (x,y) only in this region?
- User: “Search space are subgraphs of G with at most 10 edges.”

Multi-modality in AutoML



Unified String API for AutoML


```
"name": "convnet on cifar10",
"metric": "accuracy",
"goal": "MAXIMIZE",
"algorithm": "random_search",
"parameter": {
"name":   "opt_kw.lr",
"type":   "DOUBLE",
"min_value":  1e-6,
"max_value":  1e-2,
"scale_type":   "LOG"
}
"parameter": {
"name":   "opt_type",
"type":   "CATEGORICAL",
"categories":   ["SGD", "Adam"],
}
"trial" {
"parameter":   {
"opt_kw.lr":  0.0021237573,
```


Common quote: *"50% of a LLM researcher's work is writing serialization/deserialization tools."*

Relatively easy for other LLM subfields:

- Text: Human eval, multiple choice eval
- Code: Python `eval()`

Not so much for AutoML:

- Hyperparameters: `(learning_rate=0.5, batch_size=0.6)`
- Graphs: `'adj_list' = {3: [0,2,5], 7: [1, 2], …}, …}`

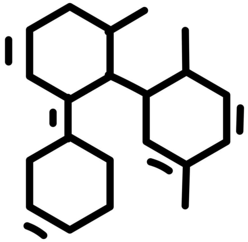
JSON is too restrictive / long.

Leverage Massive Data

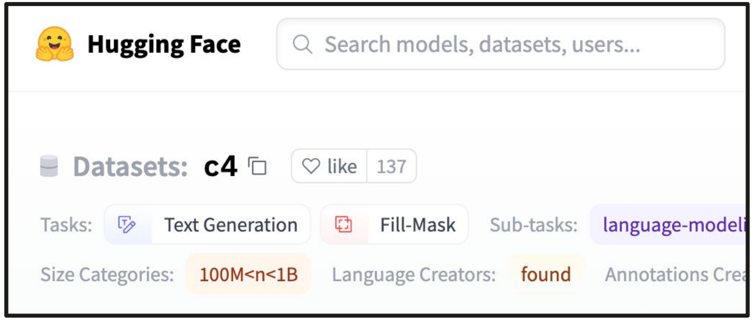
An experiment is “just” an (x, y) pair…

Collect all experimental data from humankind?

- **Concrete example:** All saved AutoML databases so-far
- Already in comp-bio community!
  - “Given chemical X, what value is its Y property?”



What other AutoML Data can we collect?



LLM Benchmarks for AutoML

~~Blackbox~~ Optimization with Hints

- “f(x) is a quadratic polynomial with unknown coefficients.”

$$\boxed{f(x) = ax^2 + bx + c}$$

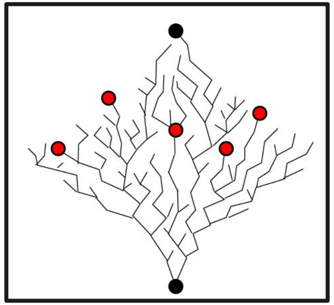
Multi-trial optimization over new domains

- Ex: Programs, Combinatorics, Constraints

```
def Setup():
  s2 = 0.001  # Init learning rate.

def Predict():  # v0 = features
  s1 = dot(v0, v1)  # Apply weights

def Learn():  # v0 = features; s0 = label
  s3 = s0 - s1  # Compute error.
  s4 = s3 * s2  # Apply learning rate.
  v2 = v0 * s4  # Compute gradient.
  v1 = v1 + v2  # Update weights.
```



AutoML Benchmarks for LLMs

Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models

Alphabetic author list:*

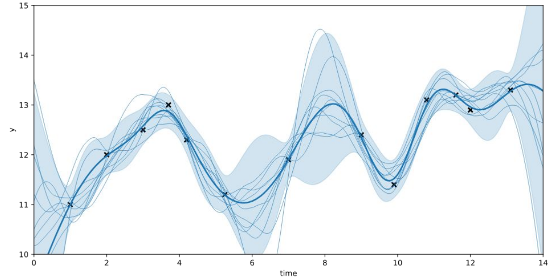
Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, Alice Xiang, Alicia Parrish, Allen Nie, Aman Hussain, Amanda Askell, Amanda Dsouza, Ambrose Slone, Ameet Rahane, Anantharaman S. Iyer, Anders Andreassen, Andrea Madotto, Andrea Santilli, Andreas Stuhlmüller, Andrew Dai, Andrew La, Andrew Lampinen, Andy Zou, Angela Jiang, Angelica Chen, Anh Vuong, Animesh Gupta, Anna Gottardi, Antonio Norelli, Anu Venkatesh, Arash Gholamidavoodi, Arfa Tabassum, Arul Menezes, Arun Kirubarajan, Asher Mullokandov, Ashish Sabharwal, Austin Herrick, Avia Efrat, Aykut Erdem, Ayla Karakaş, B. Ryan Roberts, Bao Sheng Loe, Barret Zoph, Bartłomiej Bojanowski, Batuhan Özyurt, Behnam Hedayatnia, Behnam Neyshabur, Benjamin Inden, Benno Stein, Berk Ekmekci, Bill Yuchen Lin, Blake Howald, Bryan Orinion, Cameron Diao, Cameron Dour, Catherine Stinson, Cedrick Argueta, César Ferri Ramírez, Chandan Singh, Charles Rathkopf, Chenlin Meng, Chitta Baral, Chiyu Wu, Chris Callison-Burch, Chris Waites, Christian Voigt, Christopher D. Manning, Christopher Potts, Cindy Ramirez, Clara E. Rivera, Clemencia Siro, Colin Raffel, Courtney Ashcraft, Cristina Garbacea, Damien Sileo, Dan Garrette, Dan Hendrycks, Dan Kilman, Dan Roth, Daniel Freeman, Daniel Khashabi, Daniel Levy, Daniel Moseguí González, Danielle Perszyk, Danny Hernandez, Danqi Chen, Daphne Ippolito, Dar Gilboa, David Dohan, David Drakard, David Jurgens, Debajyoti Datta, Deep Ganguli, Denis Emelin, Denis Kleyko, Deniz Yuret, Derek Chen, Derek Tam, Dieuwke Hupkes, Diganta Misra, Dilyar Buzan, Dimitri Coelho Mollo, Diyi Yang, Dong-Ho Lee, Dylan Schrader, Ekaterina Shutova, Ekin Dogus Cubuk, Elad Segal, Eleanor Hagerman, Elizabeth Barnes, Elizabeth Donoway, Ellie Pavlick, Emanuele Rodola, Emma Lam, Eric Chu, Eric Tang, Erkut Erdem, Ernie Chang, Ethan A. Chi, Ethan Dyer, Ethan Jerzak, Ethan Kim, Eunice Engefu Manyasi, Evgenii Zheltonozhskii, Fanyue Xia, Fatemeh Siar, Fernando Martínez-Plumed, Francesca Happé, Francois Chollet, Frieda Rong, Gaurav Mishra, Genta Indra Winata, Gerard de Melo, Germán Kruszewski, Giambattista Parascandolo, Giorgio Mariani, Gloria Wang, Gonzalo Jaimovitch-López, Gregor Betz, Guy Gur-Ari, Hana Galijasevic, Hannah Kim, Hannah Rashkin, Hannaneh Hajishirzi, Harsh Mehta, Hayden Bogar, Henry Shevlin, Hinrich Schütze, Hiromu Yakura, Hongming Zhang, Hugh Mee Wong, Ian Ng, Isaac Noble, Jaap Jumelet, Jack Geissinger, Jackson Kernion, Jacob Hilton, Jaehoon Lee, Jaime Fernández Fisac, James B. Simon, James Koppel, James Zheng, James Zou, Jan Kocoń, Jana Thompson, Janelle Wingfield, Jared Kaplan, Jarema Radom, Jascha Sohl-Dickstein, Jason Phang, Jason Wei, Jason Yosinski, Jekaterina Novikova, Jelle Bosscher, Jennifer Marsh, Jeremy Kim, Jeroen Taal, Jesse Engel, Jesujoba Alabi, Jiacheng Xu, Jiaming Song, Jillian Tang, Joan Waweru, John Burden, John Miller, John U. Balis, Jonathan Batchelder, Jonathan Berant, Jörg Frohberg, Jos Rozen, Jose Hernandez-Orallo, Joseph Boudeman, Joseph Guerr, Joseph Jones, Joshua B. Tenenbaum, Joshua S. Rule, Joyce Chua, Kamil Kanclerz, Karen Livescu, Karl Krauth, Karthik Gopalakrishnan, Katerina Ignatyeva, Katja Markert, Kaustubh D. Dhole, Kevin Gimpel, Kevin Omondi, Kory Mathewson, Kristen Chiafullo, Ksenia Shkaruta, Kumar Shridhar, Kyle McDonell, Kyle Richardson, Laria Reynolds, Leo Gao, Li Zhang, Liam Dugan, Lianhui Qin, Lidia Contreras-Ochando, Louis-Philippe Morency, Luca Moschella, Lucas Lam, Lucy Noble, Ludwig Schmidt, Luheng He, Luis Oliveros Colón, Luke Metz, Lütfi Kerem Şenel, Maarten Bosma, Maarten Sap, Maartje ter Hoeve, Maheen Farooqi, Manaal Faruqui, Mantas Mazeika, Marco Baturan, Marco Marelli, Marco Maru, Maria Jose Ramírez Quintana, Marie Tolkiehn, Mario Giulianelli, Martha Lewis, Martin Potthast, Matthew L. Leavitt, Matthias Hagen, Mátyás Schubert, Medina Orduna Baitemirova, Melody Arnaud, Melvin McElrath, Michael A. Yee, Michael Cohen, Michael Gu, Michael Ivanitskiy, Michael Starritt, Michael Strube, Michał Swędrowski, Michele Bevilacqua, Michihiro Yasunaga, Mihir Kale, Mike Cain, Mimee Xu, Mirac Suzgun, Mitch Walker, Mo Tiwari, Mohit Bansal, Moin Aminnaseri, Mor Geva, Mozhdeh Gheini, Mukund Varma T, Nanyun Peng, Nathan A. Chi, Nayeon Lee, Neta Gur-Ari Krakover, Nicholas Cameron, Nicholas Roberts, Nick Doiron, Nicole Martinez, Nikita Nangia, Niklas Deckers, Niklas Muennighoff, Nitish Shirish Keskar, Niveditha S. Iyer, Noah Constant, Noah Fiedel, Nuan Wen, Oliver Zhang, Omar Agha, Omar Elbaghdadi, Omer Levy, Owain Evans, Pablo Antonio Moreno Casares, Parth Doshi, Pascale Fung, Paul Pu Liang, Paul Vicol, Pegah Alipoormolabashi, Peiyuan Liao, Percy Liang, Peter Chang, Peter Eckersley, Phu Mon Htut, Pinyu Hwang, Piotr Miłkowski, Piyush Patil, Pouya Pezeshkpour, Priti Oli, Qiaozhu Mei, Qing Lyu, Qinlang Chen, Rabin Banjade, Rachel Etta Rudolph, Raefer Gabriel, Rahel Habacker, Ramon Risco, Raphaël Millière, Rhythm Garg, Richard Barnes, Rif A. Saurous, Riku Arakawa, Robbe Raymaekers, Robert Frank, Rohan Sikand, Roman Novak, Roman Sitelew, Ronan LeBras, Rosanne Liu, Rowan Jacobs, Rui Zhang, Ruslan Salakhutdinov, Ryan Chi, Ryan Lee, Ryan Stovall, Ryan Teehan, Rylan Yang, Sahib Singh, Saif M. Mohammad, Sajant Anand, Sam Dillavou, Sam Shleifer, Sam Wiseman, Samuel Gruetter, Samuel R. Bowman, Samuel S. Schoenholz, Sanghyun Han, Sanjeev Kwatra, Sarah A. Rous, Sarik Ghazarian, Sayan Ghosh, Sean Casey, Sebastian Bischoff, Sebastian Gehrmann, Sebastian Schuster, Sepideh Sadeghi, Shadi Hamdan, Sharon Zhou, Shashank Srivastava, Sherry Shi, Shikhar Singh, Shima Asaadi, Shixiang Shane Gu, Shubh Pachchigar, Shubham Toshniwal, Shyam Upadhyay, Shyamolima (Shammie) Debnath, Siamak Shakeri, Simon Thormeyer, Simone Melzi, Siva Reddy, Sneha Priscilla Makini, Soo-Hwan Lee, Spencer Torene, Sriharsha Hatwar, Stanislas Dehaene, Stefan Divic, Stefano Ermon, Stella Biderman, Stephanie Lin, Stephen Prasad, Steven T. Piantadosi, Stuart M. Shieber, Summer Misherghi, Svetlana Kiritchenko, Swaroop Mishra, Tal Linzen, Tal Schuster, Tao Li, Tao Yu, Tariq Ali, Tatsu Hashimoto, Te-Lin Wu, Théo Desbordes, Theodore Rothschild, Thomas Phan, Tianle Wang, Tiberius Nkinyili, Timo Schick, Timofei Kornev, Titus Tunduny, Tobias Gerstenberg, Trenton Chang, Trishala Neeraj, Tushar Khot, Tyler Shultz, Uri Shaham, Vedant Misra, Vera Demberg, Victoria Nyamai, Vikas Raunak, Vinay Ramasesh, Vinay Uday Prabhu, Vishakh Padmakumar, Vivek Srikumar, William Fedus, William Saunders, William Zhang, Wout Vossen, Xiang Ren, Xiaoyu Tong, Xinran Zhao, Xinyi Wu, Xudong Shen, Yadollah Yaghoobzadeh, Yair Lakretz, Yangqiu Song, Yasaman Bahri, Yejin Choi, Yichi Yang, Yiding Hao, Yifu Chen, Yonatan Belinkov, Yu Hou, Yufang Hou, Yuntao Bai, Zachary Seid, Zhuoye Zhao, Zijian Wang, Zijie J. Wang, Zirui Wang, Ziyi Wu

Multi-trial conversations

- Loop: "This cookie recipe tastes Y. Can you give me a better recipe X?"

Numeric + Combinatorial Understanding

- "Previous evaluations (x,y) were … of quadratic f(x). Predict f(x=5)?"
  - "Give me 90%-confidence interval on prediction?"
  - Tool use?



Questions?