Beyond Weights —
Adaptation through verbalized machine learning

Tim Xiao

Max Planck Institute for Intelligent Systems & University of Tübingen

A quick recap of weight-based adaptation

- Target function: $g(x)$
- Pre-trained existing function: $f(x; W)$
- Goal of weight-based adaptation:
  - Learning a transformation ϕ for W:

$$\arg\min_{\phi} \; ||f(x; \phi(W)) - g(x)||$$

    - LoRA:

$$\arg\min_{\Delta W} \; ||f(x; W + \Delta W) - g(x)||$$

    - OFT:

$$\arg\min_{R} \; ||f(x; R \cdot W) - g(x)||$$

Problems with weight-based adaptation

- **Catastrophic Forgetting**
  - E.g., Losing the general problem solving ability of LLMs
  - Difficult to constrain
  - Difficult to adapt to multiple tasks
- Access to the pre-trained model weights
- Access to compute for fine-tuning

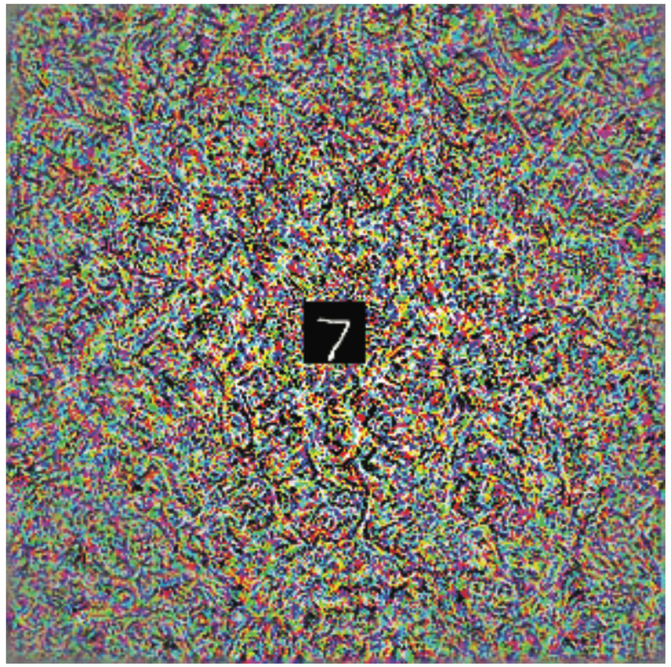
What about adapting x instead of W in $f(x; W)$?

Input-based adaptation

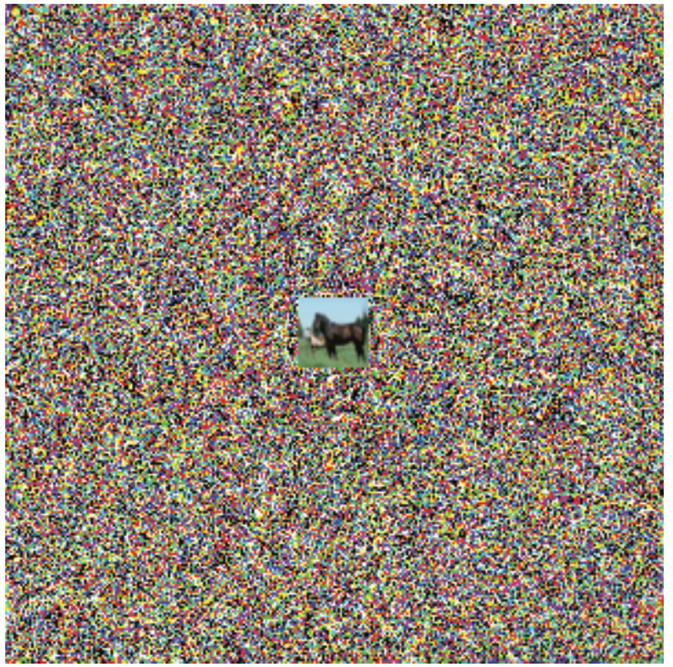
- Weight-based adaptation: $\arg\min_{\phi} \; ||f(x; \phi(W)) - g(x)||$
- Input-based adaptation: $\arg\min_{\psi} \; ||f(\psi(x); W) - g(x)||$
  - $\psi(\cdot)$ are normally less expressive than $\phi(\cdot)$
  - E.g., concatenation $\psi(x) = [x; C]$

An image classification example

(a)



(b)



- Adversarial program
- Inception V3 ImageNet model to function as (a) MNIST classifier. (b) CIFAR-10 classifier

Elsayed, et al. **Adversarial reprogramming of neural networks**, ICLR 2019

An image classification example

Table 1: **Neural networks adversarially reprogrammed to perform a variety of tasks.** Table gives accuracy of reprogrammed networks to perform a counting task, MNIST classification task, and CIFAR-10 classification task, and Shuffled MNIST pixels classification task.

Model	Pretrained on ImageNet						Untrained
	Counting	MNIST		CIFAR-10		Shuffled MNIST	MNIST
		train	test	train	test	test	test
Incep. V3	0.9993	0.9781	0.9753	0.7311	0.6911	0.9709	0.4539
Incep. V4	0.9999	0.9638	0.9646	0.6948	0.6683	0.9715	0.1861
Incep. Res. V2	0.9994	0.9773	0.9744	0.6985	0.6719	0.9683	0.1135
Res. V2 152	0.9763	0.9478	0.9534	0.6410	0.6210	0.9691	0.1032
Res. V2 101	0.9843	0.9650	0.9664	0.6435	0.6301	0.9678	0.1756
Res. V2 50	0.9966	0.9506	0.9496	0.6	0.5858	0.9717	0.9325
Incep. V3 adv.		0.9761	0.9752				

- Pre-training is important
- Larger and more powerful pre-trained model?

Elsayed, et al. **Adversarial reprogramming of neural networks**, ICLR 2019

Leveraging pre-trained LLMs

- Features:
  - Few-shot and zero-shot learning
  - Implicitly encode various types of knowledge within their weights
  - Interpretable reasoning

- Input-based adaptation — How?

Can we use natural language to parameterize functions?

Parameterize functions using natural language

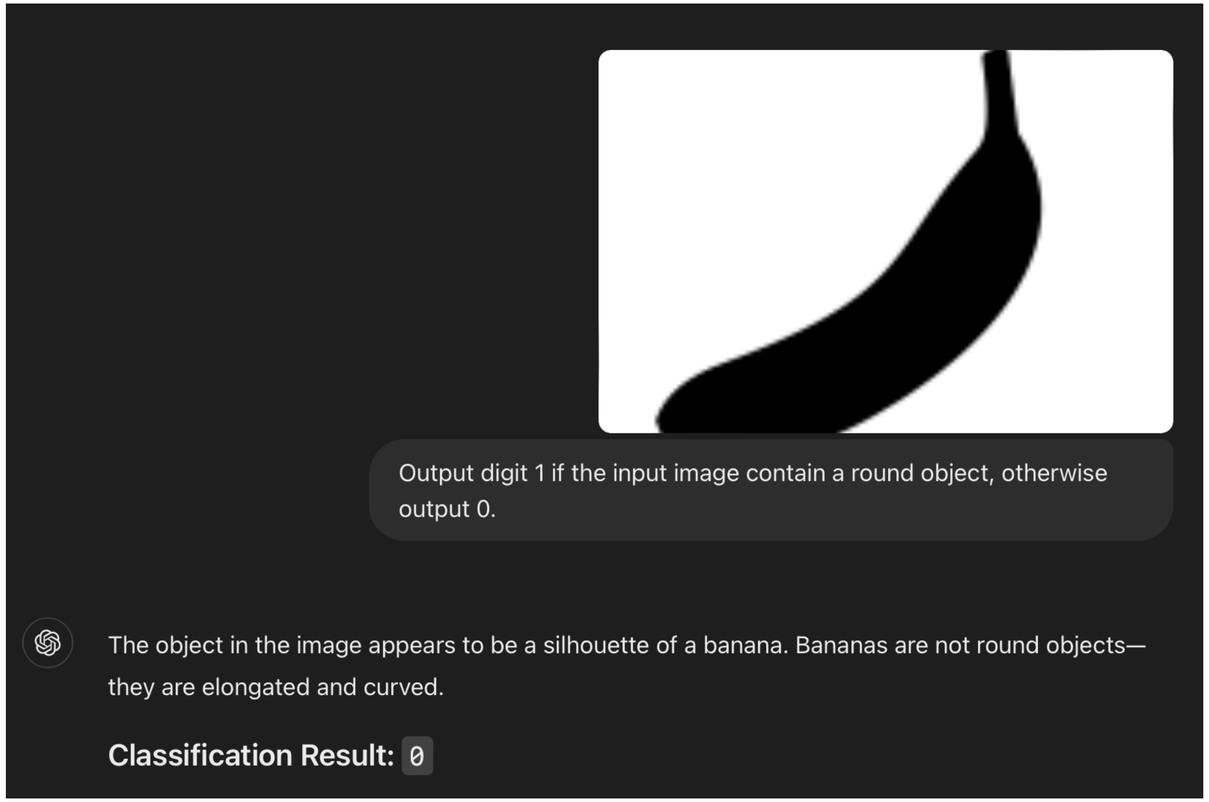
- θ in numerical space, e.g.:
  - θ is a set of matrices
  - Neural nets, CNN, etc.
- θ in natural language space, e.g.:
  - θ = “Output digit 1 if the input image contain a round object, otherwise output 0.”
- Evaluate these functions requires corresponding inference engines

$$y = f(x; \theta)$$

x :		
y :	1	0

Parameterize functions using natural language

- θ in natural language space, e.g.:
  - θ = “Output digit 1 if the input image contain a round object, otherwise output 0.”
  - Inference Engine:
    - Human
    - **LLMs**



Parameterize functions using natural language

$$y = f(x; \theta)$$

Remark

We use θ instead of W to denote the function parameters from now on. This is to distinguish θ (input; natural language) from the weights W of LLMs. The full notation should be $y = f(x, \theta; W)$, where W is fixed, both x and θ are inputs to the LLM.

How to find θ^* so that $f(x; \theta)$ can solve a given task?

How to find θ^* for a task? (θ in Numerical Space)

(Numerical) Machine Learning

- **Goal:** $\theta^* = \arg\min_\theta \; ||f(x;\theta) - g(x)||$
- Given $D_{\text{train}} = \{x, y\}$ (defining a task and the target function $g(x)$)
- And: θ_0 (initial model params), T (number of iteration), M (batch size), etc.
- For $i = 1, \ldots, T$ do
  Sample M training data $x_1, \ldots, x_M$;
  For $m = 1, \ldots, M$ do
  $\hat{y}_m = f_{\text{model}}(x_m;\ \theta_{i-1})$; // Forward Pass
  End
  $\theta_i = \theta_{i-1} - \eta \cdot \nabla_\theta \ell(y, \hat{y})$; // Backward Pass
  End

How to find θ^* for a task? (θ in Natural Language)

LLMs as function approximators (**Forward Pass**)

- $\hat{y}_m = f_{\text{model}}(x_m;\ \theta_{i-1})$
- Both x and θ are inputs to LLMs
- Similar to normal question answering settings

Xiao, et al. **Verbalized Machine Learning: Revisiting Machine Learning with Language Models**, arXiv:2406.04344

How to find θ^* for a task? (θ in Natural Language)

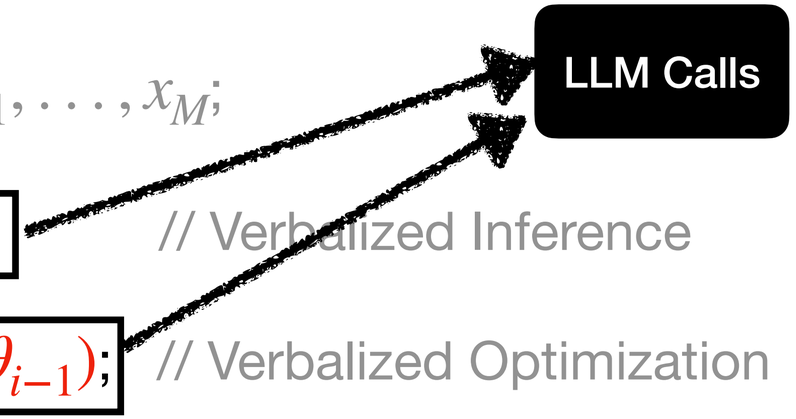
LLMs as function approximators (Backward Pass)

- $\theta_i = \theta_{i-1} - \eta \cdot \nabla_\theta \ell(y, \hat{y})$
- — This is also a function!
- Rewrite as: $\theta_i = f_{\text{opt.}}(x, y, \hat{y}, \theta_{i-1})$
- Can be approximated by LLMs too!
- Can be phrased as a question answering task

Xiao, et al. **Verbalized Machine Learning: Revisiting Machine Learning with Language Models**, arXiv:2406.04344

Verbalized Machine Learning (VML)

- **Goal:** $\theta^* = \arg\min_\theta \; ||f(x;\theta) - g(x)||$
- Given $D_{\text{train}} = \{x, y\}$ (defining a task and the target function $g(x)$)
- And: θ_0 (initial model params), T (number of iteration), M (batch size), etc.
- For $i = 1, \ldots, T$ do
  Sample M training data $x_1, \ldots, x_M$;
  For $m = 1, \ldots, M$ do
    $\hat{y}_m = f_{\text{model}}(x_m;\ \theta_{i-1})$; // Verbalized Inference
  End
  $\theta_i = f_{\text{opt.}}(\{x_m, y_m, \hat{y}_m\}^M, \theta_{i-1})$; // Verbalized Optimization
  End



Xiao, et al. **Verbalized Machine Learning: Revisiting Machine Learning with Language Models**, arXiv:2406.04344

Verbalized Inference — $f_{\text{model}}(x;\theta)$

Example Template (Regression)

"

You are the model. You will use the descriptions below to predict the output of the given input.

** Pattern Descriptions: **

You are designed to do regression, i.e., to predict the output of any given input. Both input and output are real numbers.

Model parameters θ

** Input: **

[0.59]

Data x

Please give your output strictly in the following format:

```
Explanations: [Your step-by-step analyses and results]

Output:
[Your output MUST be in REAL NUMBER ROUNDED TO TWO DECIMAL POINTS; make necessary assumptions if needed; it MUST be in the same format as the Input]
```

Please ONLY reply according to this format, don't give me any other words.

"

Verbalized Inference — $f_{\text{model}}(x;\theta)$

Example Template (Regression)

“

Explanations:
Since there is no specific pattern or function provided, I will make an assumption that the output is the same as the input.

Output:
[1.89]

Prediction $\hat{y}$

”

Verbalized Optimization — $f_{\text{opt.}}(\{x_m, y_m, \hat{y}_m\}^M, \theta_{i-1})$

Example Template (Regression)

You are the optimizer for a model, your goal is to learn the best descriptions for the model. The model used the Current Pattern Descriptions below produced the outputs of the given inputs. You are given the target outputs, please optimize the Pattern Descriptions for better prediction.

** Inputs (a batch of i.i.d. data): **
[[0.59] [1.55] [0.64] [1.43] [0.28] [0.02] [0.84] [0.39] [0.02] [1.28]]

A batch of data $\{x_1, \ldots, x_M\}$

** Current Pattern Descriptions: **
You are designed to do regression, i.e., to predict the output of any given input. Both input and output are real numbers.

Model parameters θ

** The model outputs: **
[[0.59] [3.88] [1.28] [1.43] [0.53] [0.02] [1.] [0.39] [1.] [0.]]

Predictions $\{\hat{y}_1, \ldots, \hat{y}_M\}$

** The target outputs: **
[[5.84] [8.51] [5.92] [8.09] [4.98] [3.91] [6.46] [5.23] [3.88] [7.88]]

Ground Truth $\{y_1, \ldots, y_M\}$

If the model is doing well, you can keep using the current descriptions. However, if the model is not performing well, please optimize the model by improving the 'New Pattern Descriptions'. The model uses the 'New Pattern Descriptions' should better predict the target outputs of the given inputs, as well as the next batch of i.i.d. input data from the same distribution. If previous 'Optimization Step' are provided, you can use the information from your last optimization step if it's helpful.

Verbalized loss function

Optimizer parameter Ψ

Please think step by step and give your outputs strictly in the following format:

```
Reasoning:
[be explicit and verbose, improve the Current Pattern Descriptions by yourself;]

New Pattern Descriptions:
[put your new descriptions here; MUST be specific and concrete;]
```

Please ONLY reply according to this format, don't give me any other words.

Verbalized Optimization — $f_{\text{opt.}}(\{x_m, y_m, \hat{y}_m\}^M, \theta_{i-1})$

Example Template (Regression)

Reasoning:

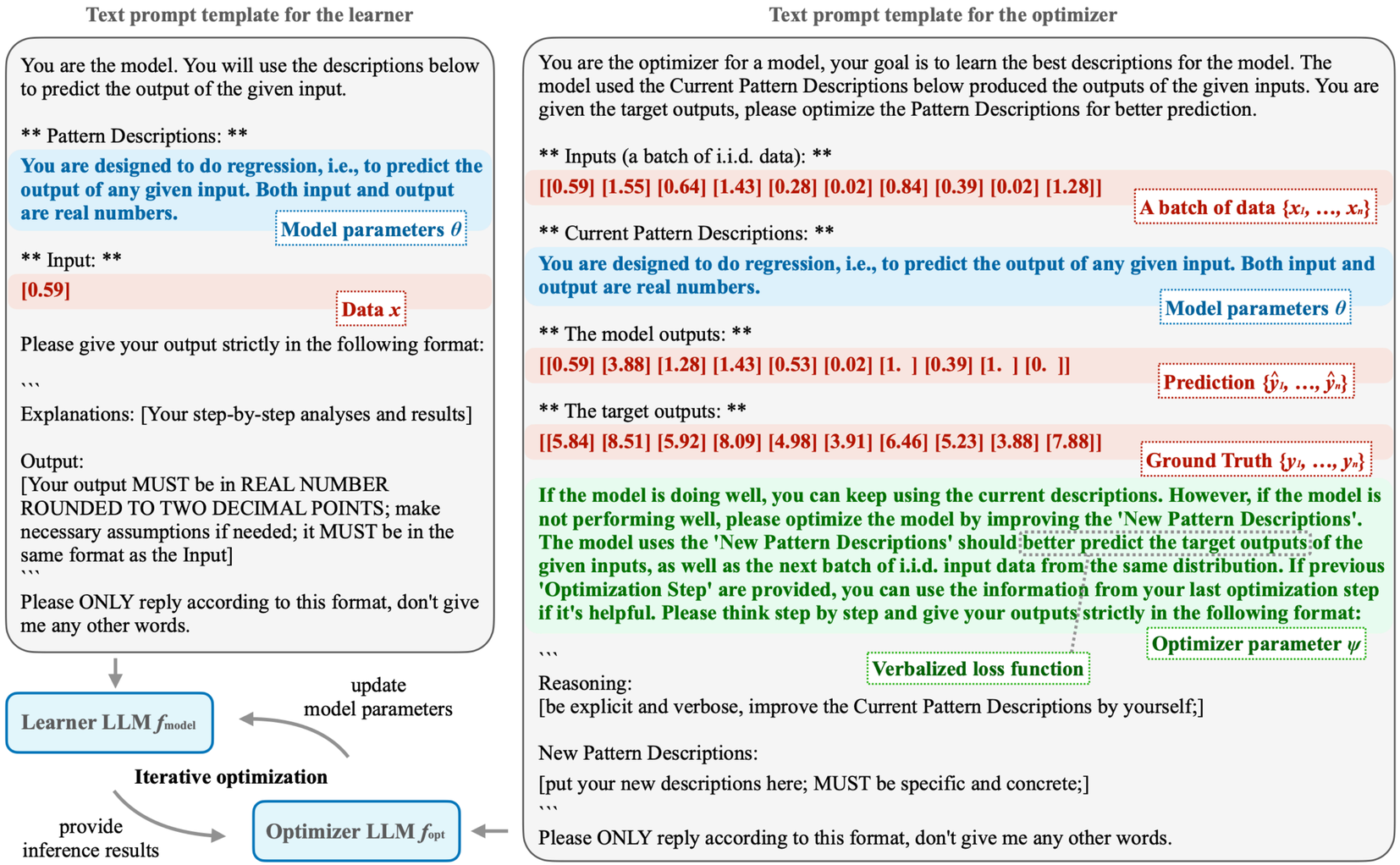
The current model outputs are not close to the target outputs, indicating that the current pattern descriptions are not effective. Upon examining the inputs and target outputs, I notice that the target outputs are generally larger than the inputs, suggesting a possible linear relationship with a positive slope. However, the current model outputs are mostly similar to the inputs, indicating that the current pattern descriptions do not capture this relationship.
To optimize the pattern descriptions, I will introduce a scaling factor to capture the linear relationship between the inputs and target outputs. I will also add a small bias term to account for any constant shift in the target outputs.

New Pattern Descriptions:

The new pattern descriptions will be based on a linear regression model with a scaling factor of 5.5 and a bias term of 2.5. The new pattern descriptions are: y = 5.5x + 2.5 where y is the predicted output and x is the input.

Model parameters θ

Iterative Optimization



Case study — Polynomial Regression

Step 1

Model parameters θ_0 at Step 1

Model parameter initialization

You are designed to do regression, i.e., to predict the output of any given input. Both input and output are real numbers.

Optimizer output at Step 1

Reasoning:

After examining the model outputs and target outputs, it's clear that the model is not performing well. The model outputs are mostly around 0, while the target outputs have a larger range of values. This suggests that the model is not capturing the underlying pattern in the data.

Upon closer inspection, I notice that the inputs and target outputs seem to have a positive correlation. This means that as the input increases, the target output also tends to increase.

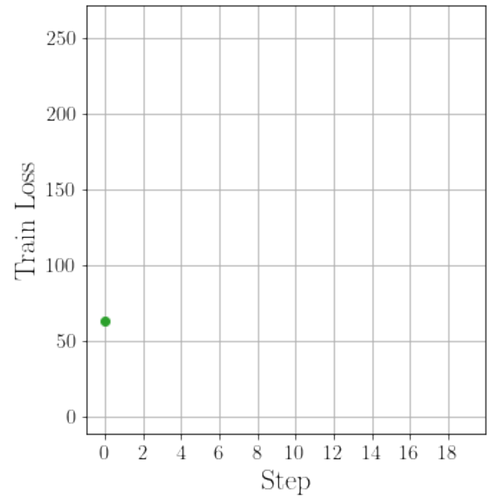
To improve the model, I will introduce a new pattern description that captures this positive correlation. I will use a simple linear relationship between the input and output, with a slope and intercept that can be learned from the data.

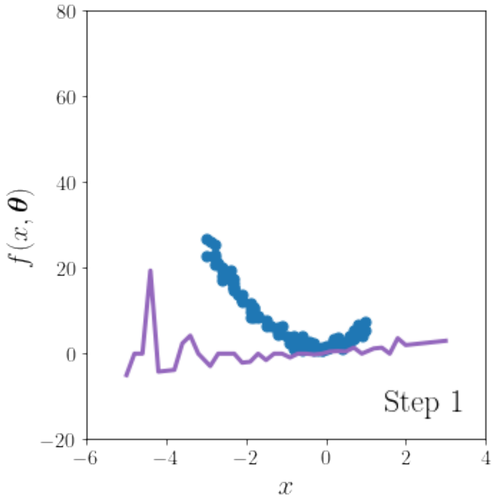
New Pattern Descriptions:

Model parameters θ_1

The new pattern description is: output = 2.5 * input + 1.5.

Model Selection





Case study — Polynomial Regression

Step 2

Model parameters θ_1 at Step 2

The new pattern description is: output = 2.5 * input + 1.5.

Optimizer output at Step 2

Reasoning:

After examining the model outputs and target outputs, it's clear that the model is not performing well. The model outputs are not close to the target outputs, and the pattern description output = 2.5 * input + 1.5 seems to be oversimplifying the relationship between the input and output.

Upon closer inspection, I notice that the target outputs are generally larger than the model outputs. This suggests that the model is underestimating the target outputs. I also notice that the target outputs seem to have a non-linear relationship with the inputs, with larger inputs corresponding to larger outputs, but not in a linear way.

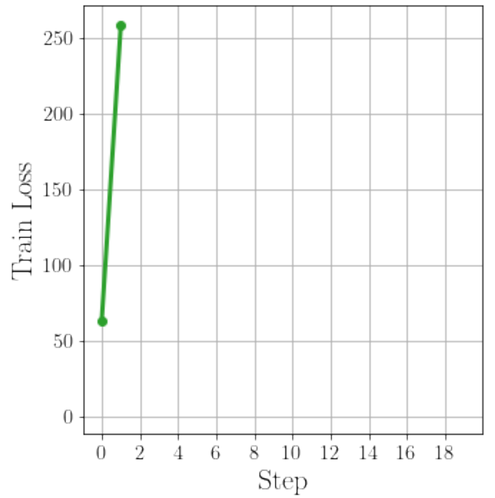
To improve the model, I will introduce a new pattern description that captures this non-linear relationship. I will use a quadratic relationship between the input and output, with coefficients that can be learned from the data. This will allow the model to better fit the curved relationship between the inputs and target outputs.

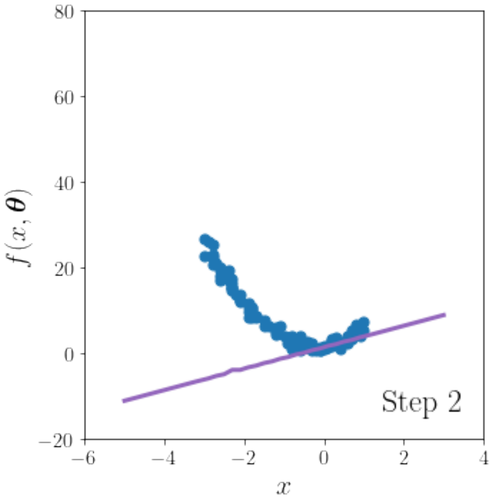
New Pattern Descriptions:

Model parameters θ_2

The new pattern description is: output = 2.2 * input^2 + 1.8 * input + 0.6.

Model Selection





Case study — Polynomial Regression

Step 3

Model parameters θ_2 at Step 3

The new pattern description is: output = 2.2 * input^2 + 1.8 * input + 0.6.

Optimizer output at Step 3

Reasoning:

After examining the model outputs and target outputs, it's clear that the model is not performing well. The model outputs are not close to the target outputs, and the pattern description output = 2.2 * input^2 + 1.8 * input + 0.6 seems to be undershooting the target outputs.

Upon closer inspection, I notice that the target outputs are generally larger than the model outputs, especially for larger inputs. This suggests that the model is underestimating the target outputs, especially for larger inputs. I also notice that the target outputs seem to have a stronger quadratic relationship with the inputs, with larger inputs corresponding to even larger outputs.

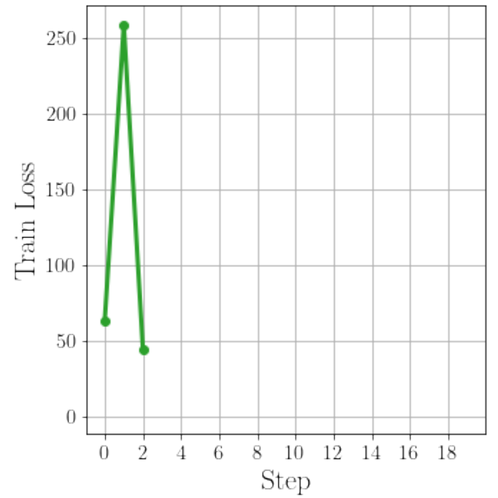
To improve the model, I will introduce a new pattern description that captures this stronger quadratic relationship. I will increase the coefficient of the quadratic term to capture the stronger curvature of the relationship. I will also adjust the linear term to better fit the data.

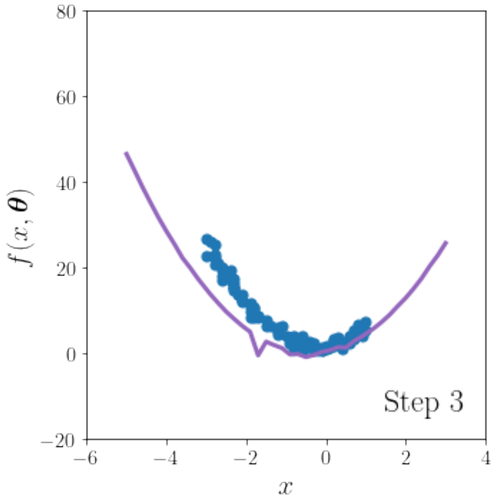
New Pattern Descriptions:

Model parameters θ_3

The new pattern description is: output = 2.8 * input^2 + 1.2 * input + 0.4.

Params Fitting



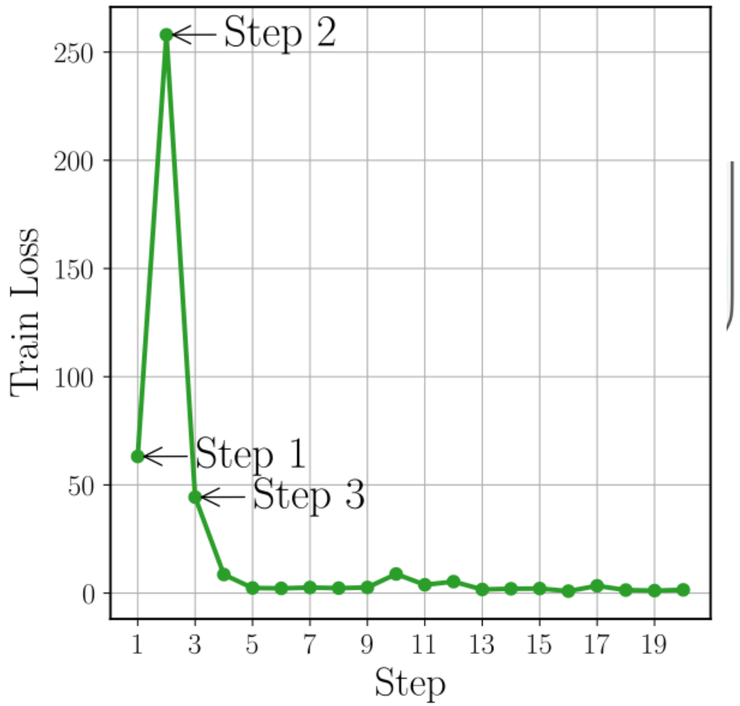


Case study — Polynomial Regression

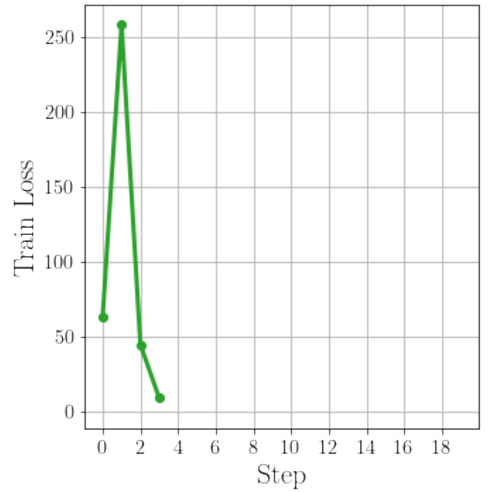
Step 4

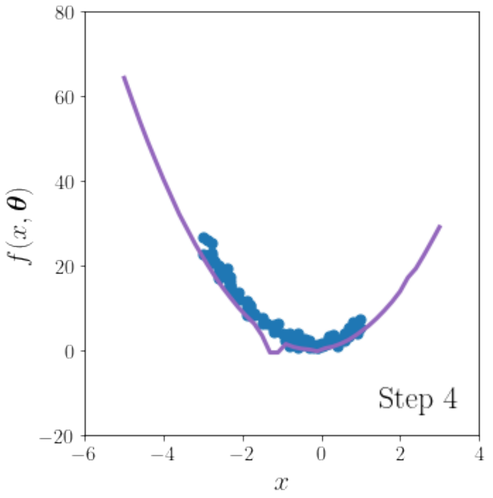
New Pat

The new
input +

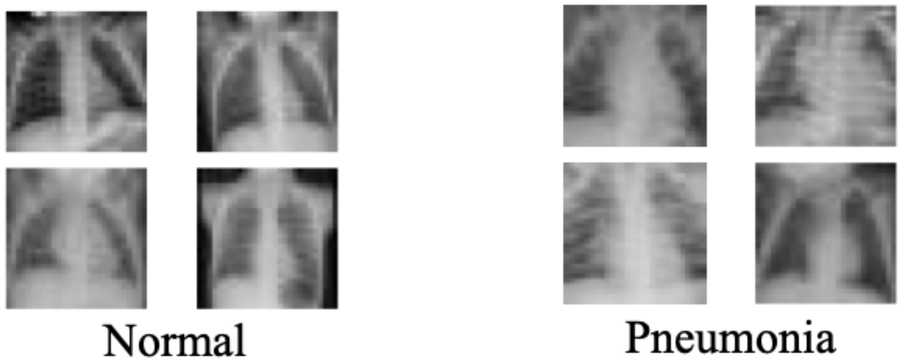


(a) Training loss dynamics





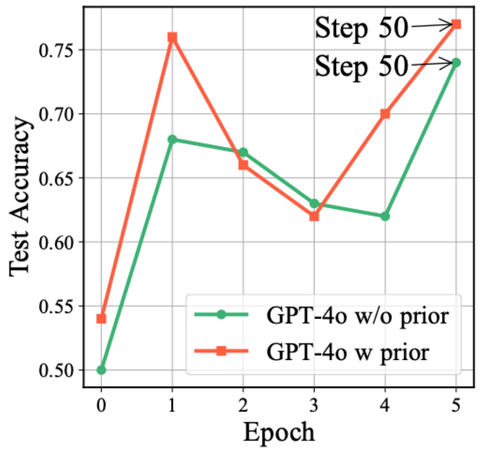
Case study — Medical Image Classification



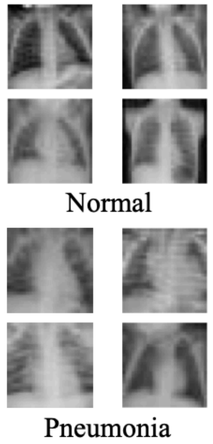
- Inference engine: vision-language model (GPT-4o)
- Binary classification

Yang, et al. **Medmnist v2-a large-scale lightweight benchmark for 2d and 3d biomedical image classification**. Scientific Data 2023

Case study — Medical Image Classification



(a) Convergence



(b) Examples

Model parameters θ_0 at Step 1 (No Prior)

Model parameter initialization

You are designed to do binary classification. The input is an image; you need to output the class label, i.e., an integer in the set {0, 1}.

(c) Model parameters at initialization

Model parameters θ_{49} at Step 50 (No Prior)

1. If the image shows clear anatomical structures, uniform texture, symmetry, and normal lung fields without visible opacities, well-defined masses, streaks, irregular granular patterns, or any deviations, output class label 0.
2. If the image shows any distinct opacities or masses, interstitial patterns, visible fluid accumulations, asymmetry, irregular heart shapes, or multiple small opacities, output class label 1.
3. For granular or streaky opacities over any lung region, indistinct or uneven lung fields, subtle masses, minor deviations in symmetry, or any lung irregularities, output class label 1. For minimal irregularities or slight deviations, output class label 0.

Optimizer output at Step 50 (No Prior)

Reasoning:

The model misclassified several images where minor opacities or very light irregular patterns were present. We need to be more precise in detecting these subtle features while examining the overall lung fields more thoroughly to reduce false negatives for label 1. Specifically, minor irregularities or granular patterns that seem inconsequential should still be marked as potential abnormalities if they occur in specific regions, such as the periphery.

New Model Descriptions:

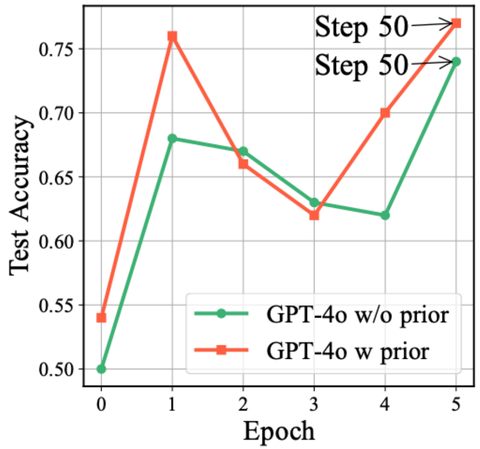
Model parameters θ_{50}

1. If the image shows clear anatomical structures, uniform texture, symmetry, and normal lung fields without any visible opacities, well-defined masses, streaks, or granular patterns, output class label 0.
2. If the image shows any distinct opacities or masses, interstitial patterns, visible fluid accumulations, asymmetry, irregular heart shapes, or multiple small opacities across the lung fields, output class label 1.
3. For granular or streaky opacities over any lung region, indistinct or uneven lung fields, subtle masses, minor deviations in symmetry, or any lung irregularities or granular patterns, including faint ones, in the periphery, output class label 1. For very minimal irregularities or slight deviations in central regions only, output class label 0.

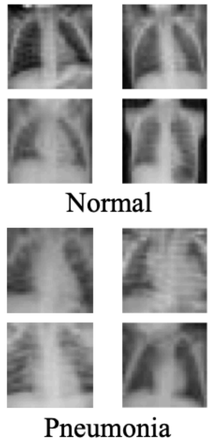
General Visual Features

(d) Model parameters after training

Case study — Medical Image Classification



(a) Convergence



(b) Examples

Model parameters θ_0 at Step 1 (with Prior)

Model parameter initialization

Prior: The input is X-ray image for identifying pneumonia.

You are designed to do binary classification. The input is an image; you need to output the class label, i.e., an integer in the set {0, 1}.

(c) Model parameters at initialization

Model parameters θ_{49} at Step 50 (with Prior)

The input is an X-ray image. To classify:

- Output 1 (Pneumonia) if there are lobar consolidations, asymmetric or bilateral ground-glass opacities, air bronchograms, moderate to significant alveolar fluid, asymmetric infiltrates, patchy inflammations, moderate to severe pleural effusion, central or peripheral acute opacities, air-fluid levels, increasing opacities, pneumatocele formations, rapidly changing patterns in chronic lesions indicative of acute infection, or persistent cough-based findings.

- Output 0 (No Pneumonia) if lung fields are clear, minimal localized fluid, no distinct acute opacities, stable fibrotic scarring, solitary calcifications, old healed lesions, stable localized non-infectious anomalies, chronic interstitial fibrosis, chronic reticulonodular patterns without recent changes, stable chronic lesions without acute signs, or purely non-persistent anomalies.

Optimizer output at Step 50 (with Prior)

Reasoning:

The model missed some true cases of pneumonia and incorrectly identified some non-pneumonia cases. To improve the accuracy, refine the criteria to better identify pneumonia even in less obvious cases and differentiate between chronic stable conditions and acute infections.

New Model Descriptions:

Model parameters θ_{50}

The input is an X-ray image. To classify:

- Output 1 (Pneumonia) if there are lobar consolidations, asymmetric or bilateral ground-glass opacities, air bronchograms, moderate to significant alveolar fluid, asymmetric infiltrates, patchy inflammations, moderate to severe pleural effusion, central or peripheral acute opacities, air-fluid levels, increasing opacities, pneumatocele formations, rapidly changing patterns in chronic lesions indicative of acute infection, persistent cough-related findings, or recent onset of ground-glass opacities.

- Output 0 (No Pneumonia) if lung fields are clear, minimal localized fluid, no distinct acute opacities, stable fibrotic scarring, solitary calcifications, old healed lesions, stable localized non-infectious anomalies, chronic interstitial fibrosis, chronic reticulonodular patterns without recent changes, stable chronic lesions without acute signs, or absence of any acute infection markers.

Symptom Features

(d) Model parameters after training

Case study — Mathematical Reasoning

Question Template

When {**name**} watches her {**family**}, she gets out a variety of toys for him. The bag of building blocks has {**x**} blocks in it. The bin of stuffed animals has {**y**} stuffed animals inside. The tower of stacking rings has {**z**} multicolored rings on it. {**name**} recently bought a tube of bouncy balls, bringing her total number of toys she bought for her {**family**} up to {**total**}. How many bouncy balls came in the tube?

Variable Domain

name = random.choice(["Alice", "Beth", "Cindy", "Diana", "Emily", "Fiona", "Gina", "Hannah", "Ivy", "Jenny"])
family = random.choice(["nephew", "cousin", "brother"])
x = randint(5, 100)
y = randint(5, 100)
z = randint(5, 100)
total = randint(100, 500)
ans = total - (x+y+z) with condition: ans >= 85 and ans <= 200

- Many LLMs have declined performance on the symbolically modified GSM8K
- Might be due to data contamination and memorization during pre-training
- VML can reduce such a performance variation and enable robust mathematical reasoning without changing the internal weights of LLMs.
- Generate training and test set from the above template

Mirzadeh, et al. **Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models**. arXiv:2410.05229

Case study — Mathematical Reasoning

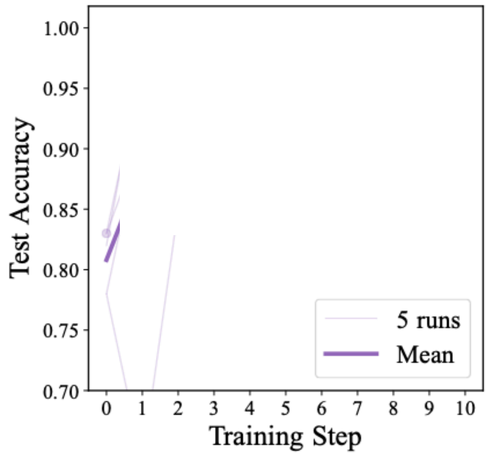
Question Template

When {**name**} watches her {**family**}, she gets out a variety of toys for him. The bag of building blocks has {**x**} blocks in it. The bin of stuffed animals has {**y**} stuffed animals inside. The tower of stacking rings has {**z**} multicolored rings on it. {**name**} recently bought a tube of bouncy balls, bringing her total number of toys she bought for her {**family**} up to {**total**}. How many bouncy balls came in the tube?

Variable Domain

name = random.choice(["Alice", "Beth", "Cindy", "Diana", "Emily", "Fiona", "Gina", "Hannah", "Ivy", "Jenny"])
family = random.choice(["nephew", "cousin", "brother"])
x = randint(5, 100)
y = randint(5, 100)
z = randint(5, 100)
total = randint(100, 500)
ans = total - (x+y+z) with condition: ans >= 85 and ans <= 200

(a) The symbolic template used for generating training and testing questions



(b) Test accuracy at each step

Model parameters θ_0 at Step 1

Model parameter initialization

You are given a math question, please reason and produce the corresponding answer. The answer is an integer.

(c) Model parameters at initialisation

(d) Optimizer output at Step 1, 3 and 6

Case study — Mathematical Reasoning

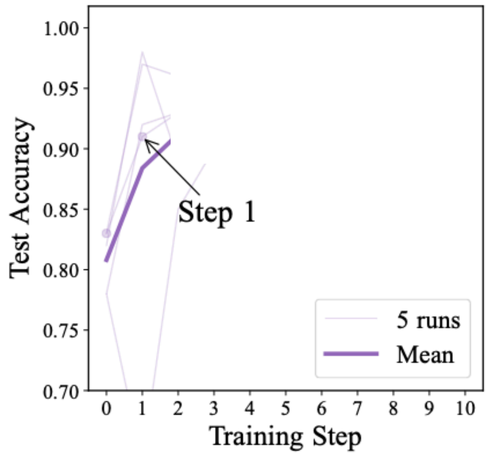
Question Template

When {**name**} watches her {**family**}, she gets out a variety of toys for him. The bag of building blocks has {**x**} blocks in it. The bin of stuffed animals has {**y**} stuffed animals inside. The tower of stacking rings has {**z**} multicolored rings on it. {**name**} recently bought a tube of bouncy balls, bringing her total number of toys she bought for her {**family**} up to {**total**}. How many bouncy balls came in the tube?

Variable Domain

name = random.choice(["Alice", "Beth", "Cindy", "Diana", "Emily", "Fiona", "Gina", "Hannah", "Ivy", "Jenny"])
family = random.choice(["nephew", "cousin", "brother"])
x = randint(5, 100)
y = randint(5, 100)
z = randint(5, 100)
total = randint(100, 500)
ans = total - (x+y+z) with condition: ans >= 85 and ans <= 200

(a) The symbolic template used for generating training and testing questions



(b) Test accuracy at each step

Model parameters θ_0 at Step 1

Model parameter initialization

You are given a math question, please reason and produce the corresponding answer. The answer is an integer.

(c) Model parameters at initialisation

Optimizer output at Step 1

Reasoning:

To solve the problem, we need to find the difference between the total number of toys after buying the bouncy balls and the total number of toys before buying the bouncy balls. We are given the total number of toys after buying the bouncy balls and the number of blocks, stuffed animals, and stacking rings before buying the bouncy balls. First, we need to add the number of blocks, stuffed animals, and stacking rings to find the total number of toys before buying the bouncy balls. Then, we subtract this total from the total number of toys after buying the bouncy balls to find the number of bouncy balls.

New Instructions: **Model parameters θ_1**

Add the number of blocks, stuffed animals, and stacking rings to find the total number of toys before buying the bouncy balls. Subtract this total from the total number of toys after buying the bouncy balls to find the number of bouncy balls.

Got the correct description already

(d) Optimizer output at Step 1, 3 and 6

Case study — Mathematical Reasoning

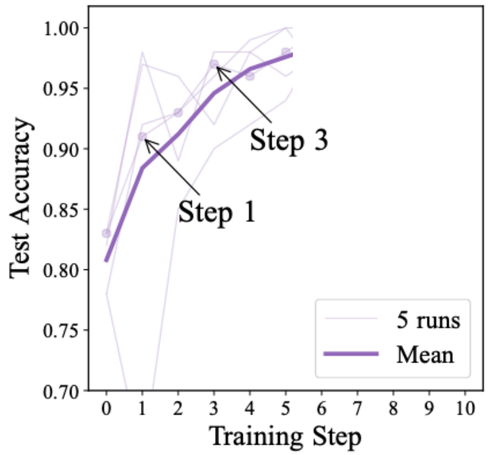
Question Template

When {**name**} watches her {**family**}, she gets out a variety of toys for him. The bag of building blocks has {**x**} blocks in it. The bin of stuffed animals has {**y**} stuffed animals inside. The tower of stacking rings has {**z**} multicolored rings on it. {**name**} recently bought a tube of bouncy balls, bringing her total number of toys she bought for her {**family**} up to {**total**}. How many bouncy balls came in the tube?

Variable Domain

name = random.choice(["Alice", "Beth", "Cindy", "Diana", "Emily", "Fiona", "Gina", "Hannah", "Ivy", "Jenny"])
family = random.choice(["nephew", "cousin", "brother"])
x = randint(5, 100)
y = randint(5, 100)
z = randint(5, 100)
total = randint(100, 500)
ans = total - (x+y+z) with condition: ans >= 85 and ans <= 200

(a) The symbolic template used for generating training and testing questions



(b) Test accuracy at each step

Model parameters θ_0 at Step 1

Model parameter initialization

You are given a math question, please reason and produce the corresponding answer. The answer is an integer.

(c) Model parameters at initialisation

Optimizer output at Step 1

Reasoning:

To solve the problem, we need to find the difference between the total number of toys after buying the bouncy balls and the total number of toys before buying the bouncy balls. We are given the total number of toys after buying the bouncy balls and the number of blocks, stuffed animals, and stacking rings before buying the bouncy balls. First, we need to add the number of blocks, stuffed animals, and stacking rings to find the total number of toys before buying the bouncy balls. Then, we subtract this total from the total number of toys after buying the bouncy balls to find the number of bouncy balls.

New Instructions: **Model parameters θ_1**

Add the number of blocks, stuffed animals, and stacking rings to find the total number of toys before buying the bouncy balls. Subtract this total from the total number of toys after buying the bouncy balls to find the number of bouncy balls.

Optimizer output at Step 3

Reasoning:

Upon reviewing the questions and solver outputs, it appears that the current instructions are clear and specific. However, to improve the accuracy of the solver, we should emphasize the importance of accurate calculations and double-checking the results. The solver's mistakes seem to be random and not related to the instructions. To optimize the instructions, we should focus on providing more detailed steps and encouraging the solver to verify its results.

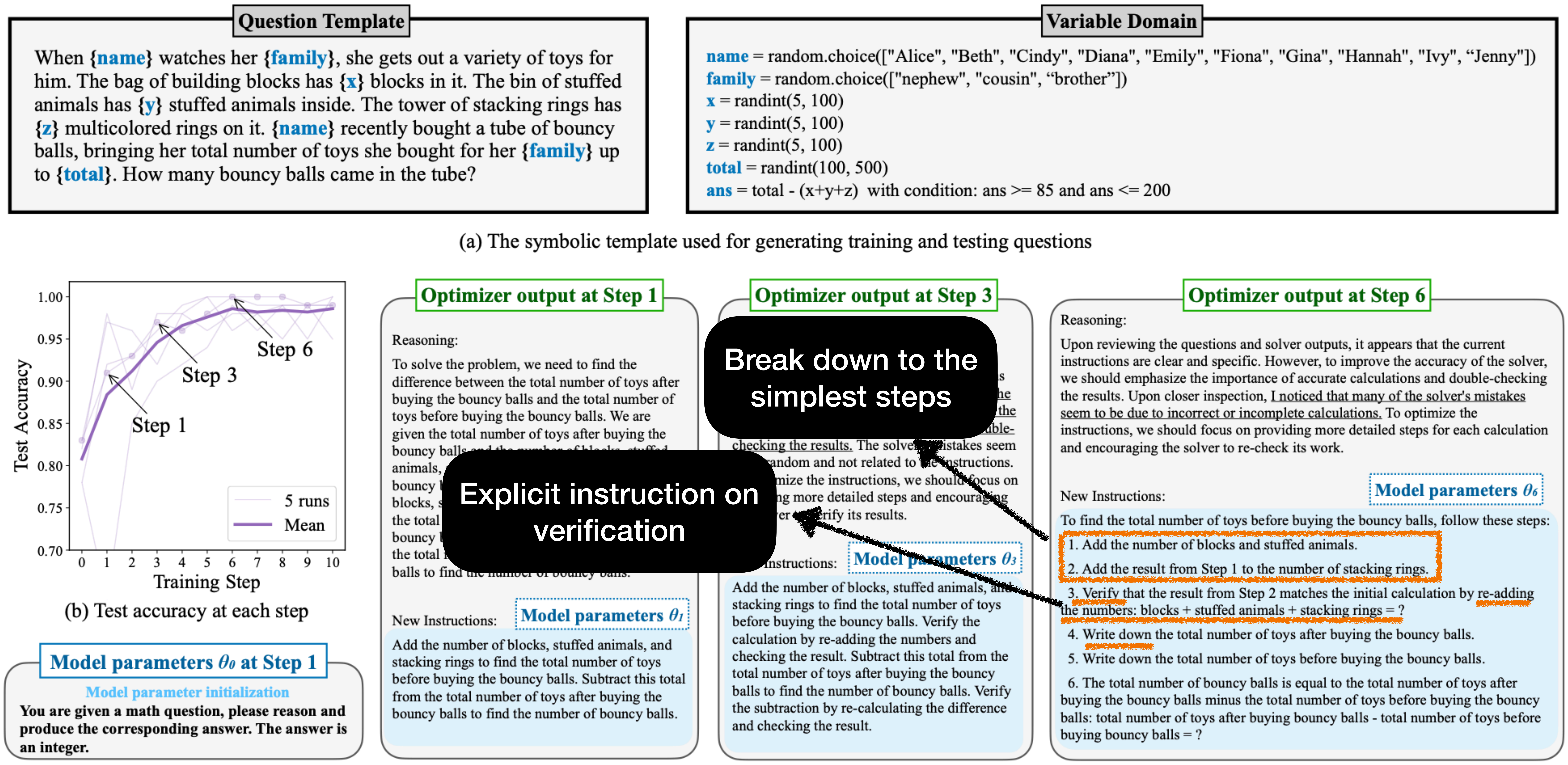
New Instructions: **Model parameters θ_3**

Add the number of blocks, stuffed animals, and stacking rings to find the total number of toys before buying the bouncy balls. Verify the calculation by re-adding the numbers and checking the result. Subtract this total from the total number of toys after buying the bouncy balls to find the number of bouncy balls. Verify the subtraction by re-calculating the difference and checking the result.

(d) Optimizer output at Step 1, 3 and 6

Learning general Instructions

Case study — Mathematical Reasoning



(a) The symbolic template used for generating training and testing questions

(b) Test accuracy at each step

(c) Model parameters at initialisation

(d) Optimizer output at Step 1, 3 and 6

Advantages of Verbalized Machine Learning

1. **Easy encoding of inductive bias**:
   Prior knowledge about the problem and hypothesis class can be encoded in natural language and fed into the LLM-parameterized learner.

2. **Automatic model class selection**:
   The optimizer can automatically select a model class based on data and verbalized prior knowledge, and it can update the model class during training.

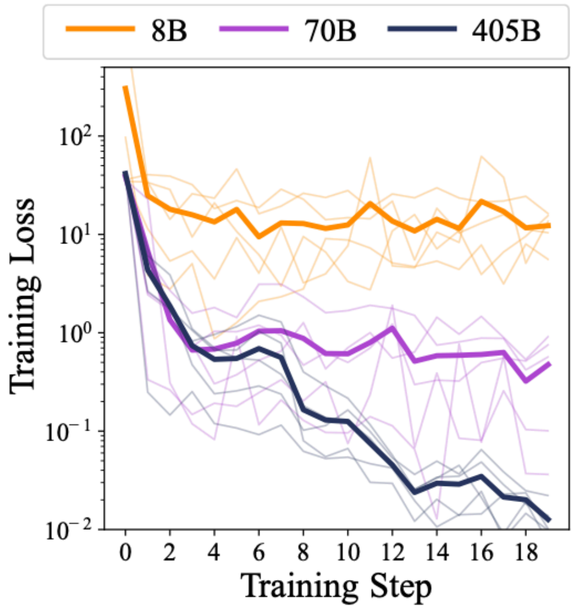
3. **Interpretable updates**:
   The LLM-parameterized optimizer can provide explanations for why an update is performed.

Current limitations of VML

1. **Numerical ability**
2. **Limited optimizer**
3. **Measuring distance between two function**

Scaling effect with various size of LLMs

- Llama-3.1 with different size (8B, 70B, 405B)
- Linear regression setting
- **Larger LLMs learn faster and achieve lower loss**



(a) Powerful LLMs improve VML

Difference between VML and prompt optimization

- Similar to Machine Learning vs Optimization
- Goals:
  - **ML / VML** :
    - To build models that can learn patterns or relationships in data and make predictions or decisions on unseen data.
    - Can be phrased as an optimization objective, and use tools from opt.
  - **Optimization / Prompt optimization**: To find the best solution to a problem by minimizing or maximizing an objective function.

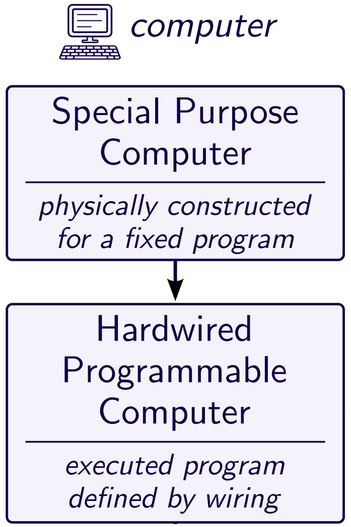
From input based adaptation to universality

Special Purpose Computer

physically constructed for a fixed program

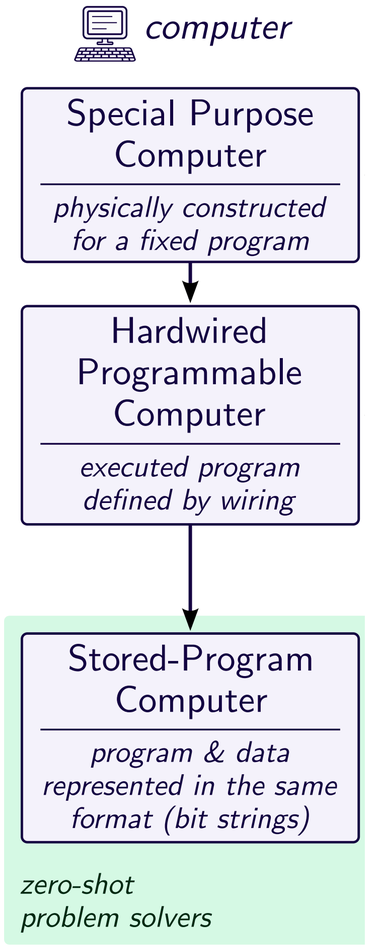
https://timx.me/blog/2023/computers-vs-llms/

From input based adaptation to universality



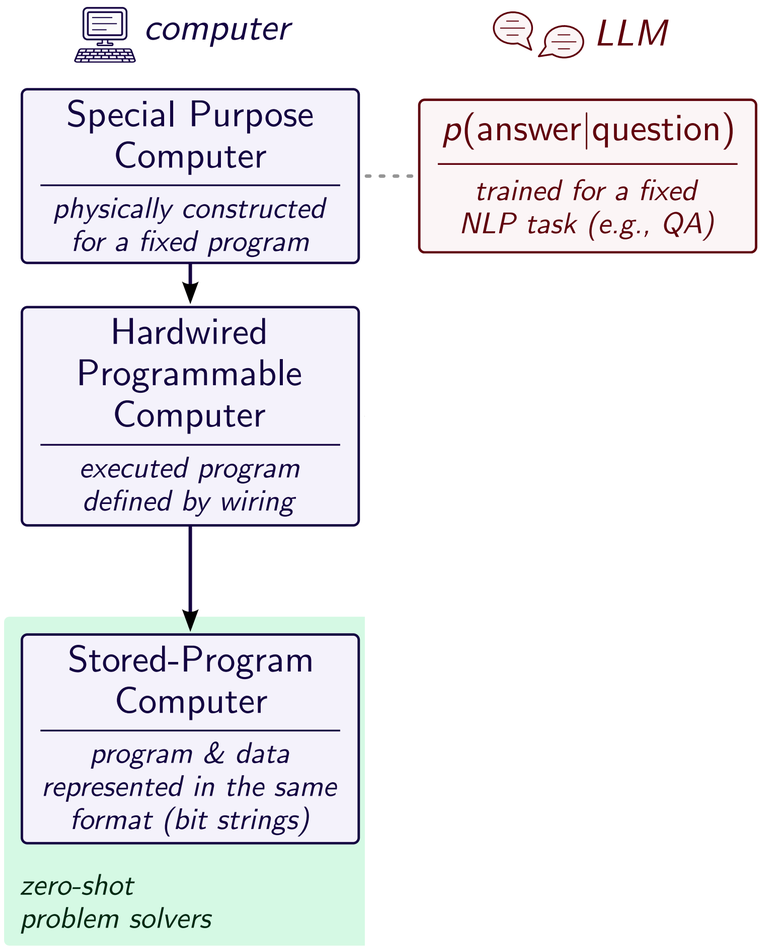
https://timx.me/blog/2023/computers-vs-llms/

From input based adaptation to universality



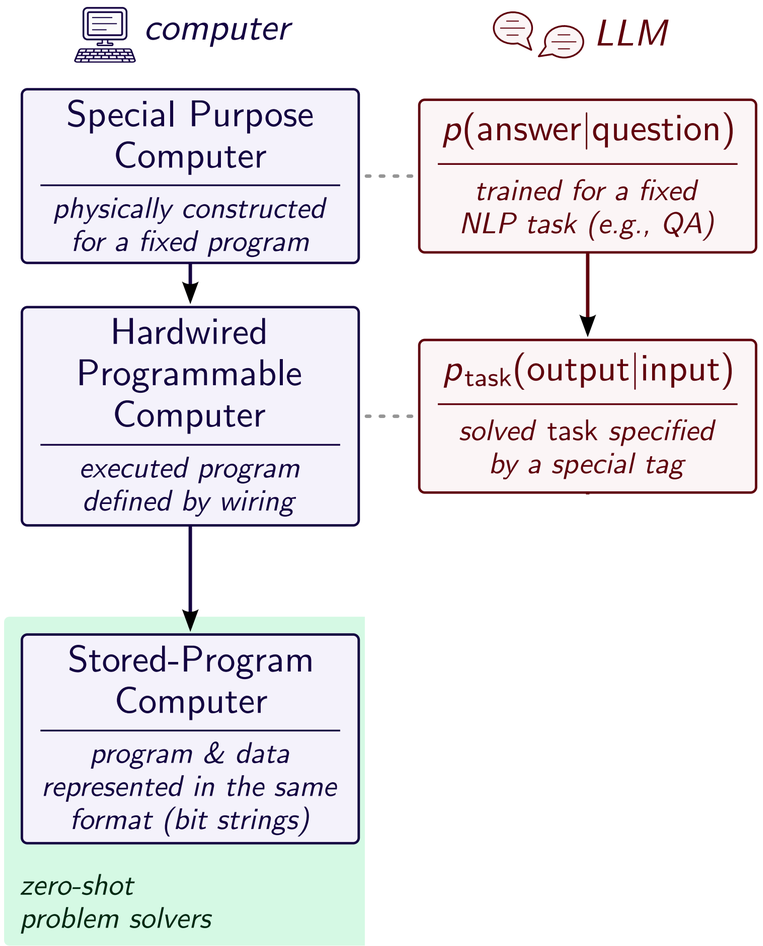
https://timx.me/blog/2023/computers-vs-llms/

From input based adaptation to universality



https://timx.me/blog/2023/computers-vs-llms/

From input based adaptation to universality

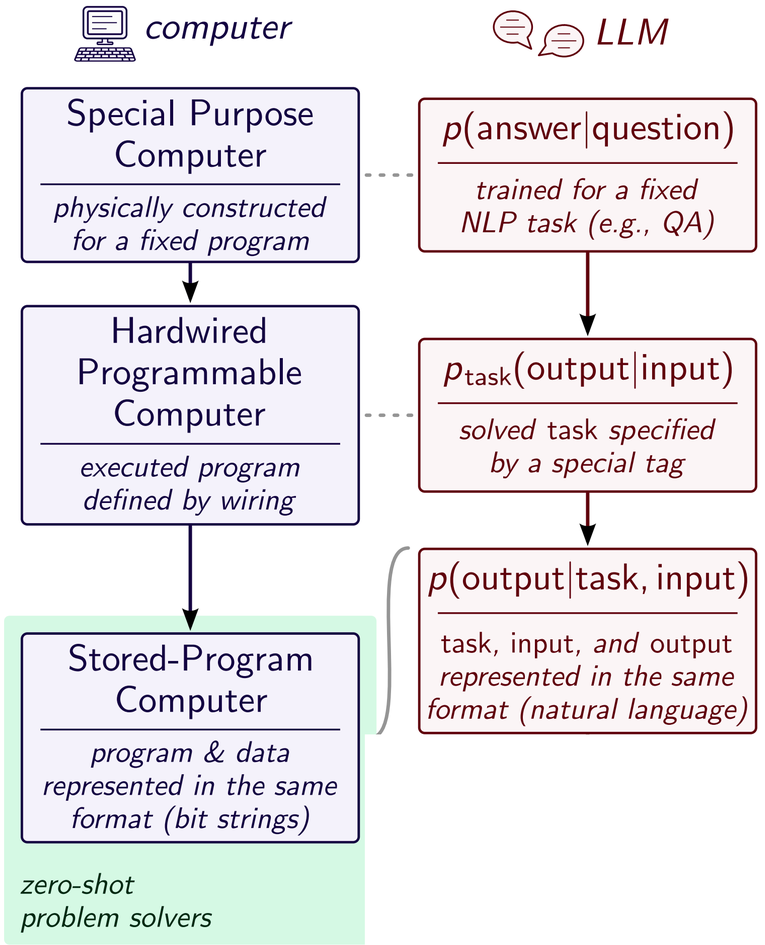


Shazeer, et al.
Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. ICLR 2017

Kaiser, et al.
One model to learn them all. arXiv:1706.05137

https://timx.me/blog/2023/computers-vs-llms/

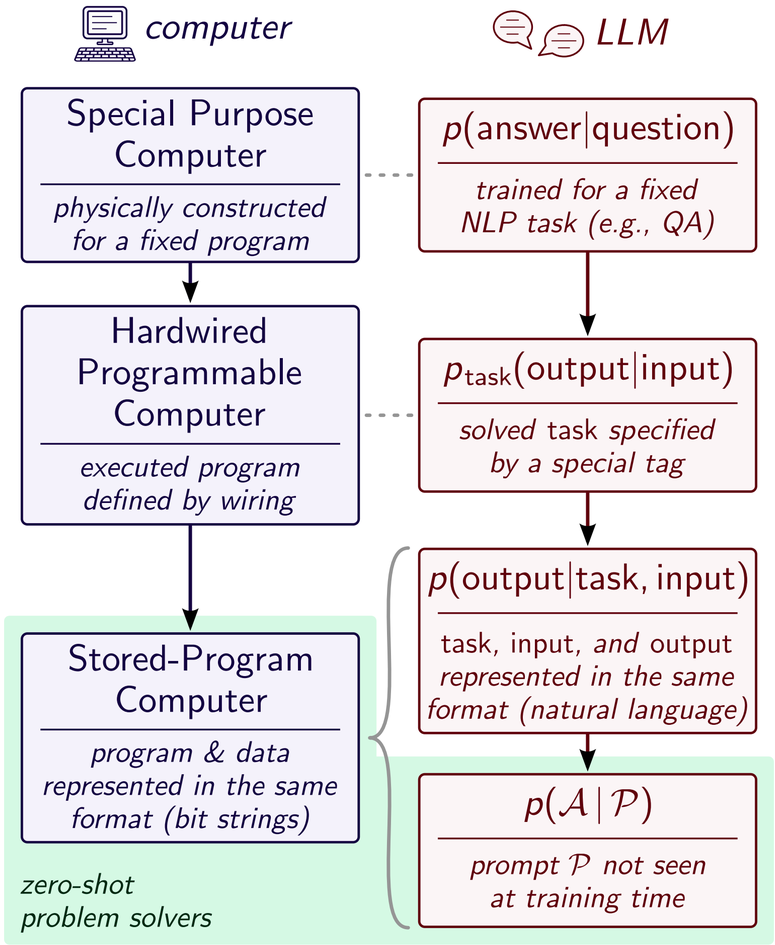
From input based adaptation to universality



McCann, et al.
The natural language decathlon: Multitask learning as question answering. arXiv:1806.08730

https://timx.me/blog/2023/computers-vs-llms/

From input based adaptation to universality



Radford, et al.
Language models are unsupervised multitask learners. OpenAI blog 2019

https://timx.me/blog/2023/computers-vs-llms/

Summary

- Input-based Adaptation vs Weight-based Adaptation
- Verbalized machine learning
  - Use natural language to parameterized functions.
  - Optimize such parameters to approximate functions (learning)
- Connections to modern computers