Deep Learning

CSCI 1470

Day 34: Future of DL

Eric Ewing

Monday, 4/21/25

Challenges in RL and Robotics

- Simulation environment and real world won’t match perfectly (Sim2Real Gap)
  - Hard to collect enough data in the real world
  - Impossible to simulate physics perfectly
- No guarantees of safe policies
  - If you follow a learned and cause an accident, that’s very expensive
- Sparse/Delayed rewards
  - It is challenging for a robot to know if it is doing well until a task is complete
- Partial Observability in the real world
  - Robots do not have access to the entire world state, just what they can observe with their sensors.

Why don’t we see more RL in deployed robots?

Why don’t we see more RL in deployed robots?

Industrial robots work in *very controlled* environments

Deep Learning is not the answer to every problem

We already know optimal-control algorithms for certain types of problems, Deep RL cannot be better than optimal solutions…

The strength of Deep Learning is its ability to handle uncertainty and generalize to new data/environments

But there’s lots of problems left

How could we create
generally intelligent robots?

General Intelligence

What properties do we want from a generally intelligent robot?

1. **Adapt to new environments and tasks quickly**
2. Goal alignment and value learning
3. Work with multi-modal data
4. Safe exploration and failure recovery
5. Long term memory and experience integration
6. Explainability and Interpretability

Adapt After Training: Continual Learning

What do you do when you encounter new data?

Keep trying to update your model…

2 things may go wrong:

Catastrophic Forgetting: The network no longer knows how to complete a task it once knew

Loss of Plasticity: The network can no longer learn and adapt to new tasks

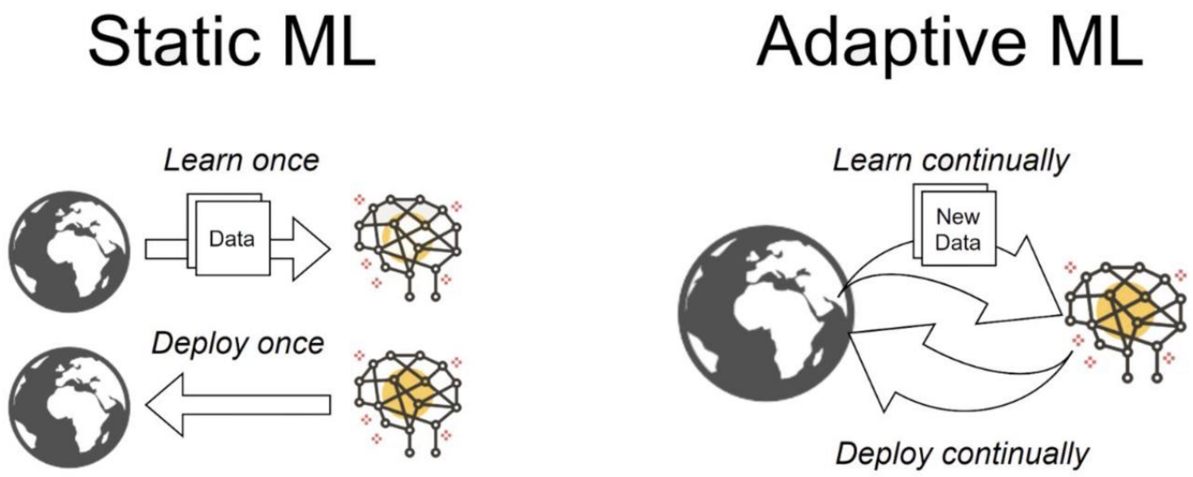
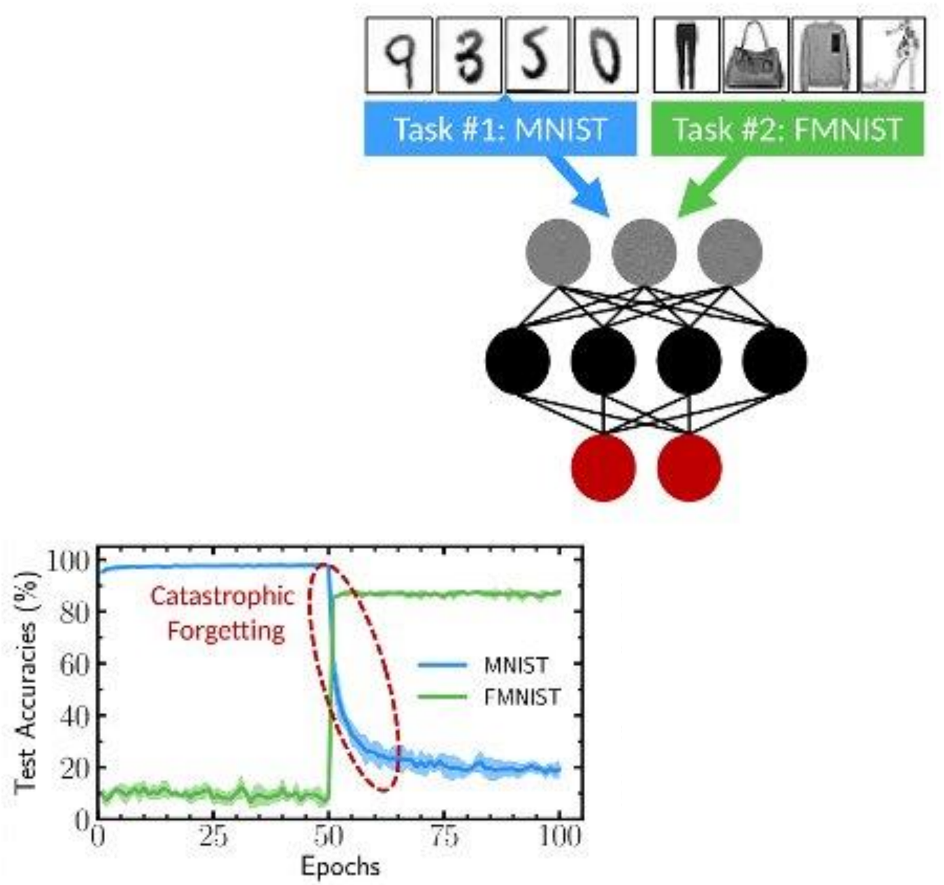


Image source: https://imerit.net/blog/a-complete-introduction-to-continual-learning/

Catastrophic Forgetting

Train network on MNIST, then switch to FMNIST (separate outputs)

Ideally, our networks would remember how to complete the MNIST task

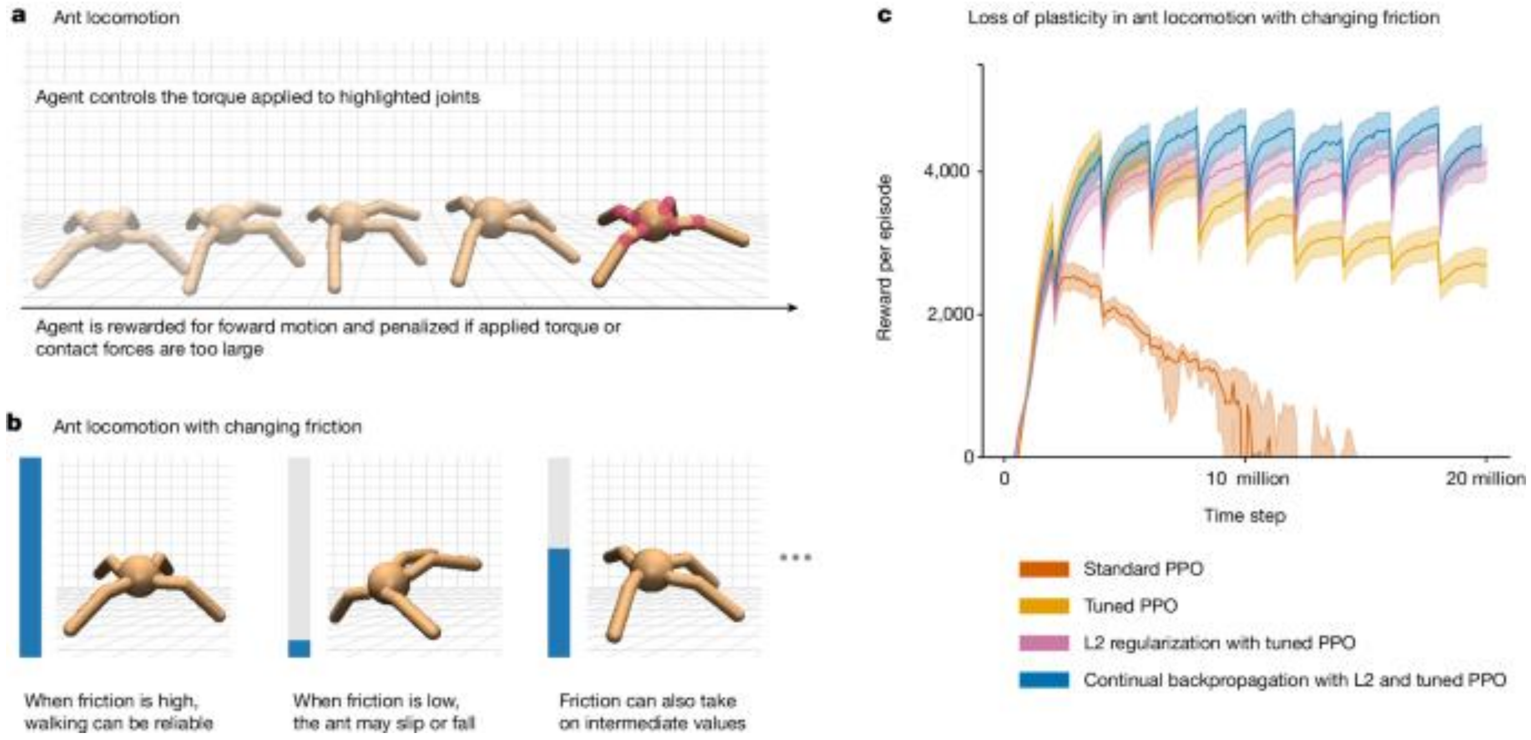


Source: https://www.nature.com/articles/s41467-021-22768-y

Loss Of Plasticity

Catastrophic forgetting is a problem whenever the task switches

But even worse… the network may not learn to complete new tasks



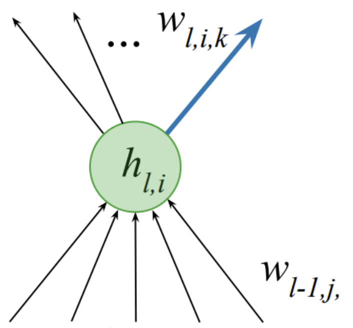
Source: Loss of Plasticity in Continual Deep Learning https://www.nature.com/articles/s41586-024-07711-7

Continual Backprop

Calculate *utility* of each neuron in network

Reinitialize neurons that do not contribute to the output

Continue to run SGD on dataset

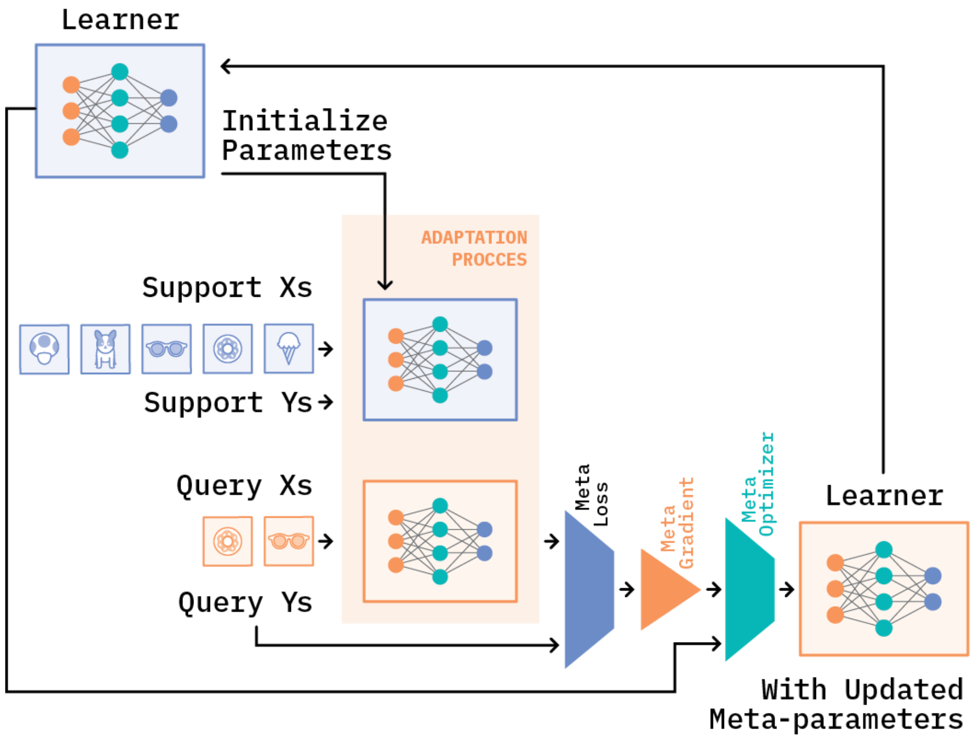


$$y_{l,i,t} = \frac{|h_{l,i,t} - \hat{f}_{l,i,t}| * \sum_{k=1}^{n_{l+1}} |w_{l,i,k,t}|}{\sum_{j=1}^{n_{l-1}} |w_{l-1,j,i,t}|}$$

Figure 4: A feature/hidden-unit in a network. The utility of a feature at time t is the product of its contribution utility and its adaptation utility. Adaptation utility is the inverse of the sum of the magnitude of the incoming weights. And, contribution utility is the product of the magnitude of the outgoing weights and feature activation ($h_{l,i}$) minus its average ($\hat{f}_{l,i}$). $\hat{f}_{l,i}$ is a running average of $h_{l,i}$.

Adapting to New Tasks: Meta-Learning

Train a model that can adapt **quickly** to new tasks



Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
6: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
7: **end for** Note: the meta-update is using different set of data.
8: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
9: **end while**

Model Agnostic Meta-Learning (MAML)

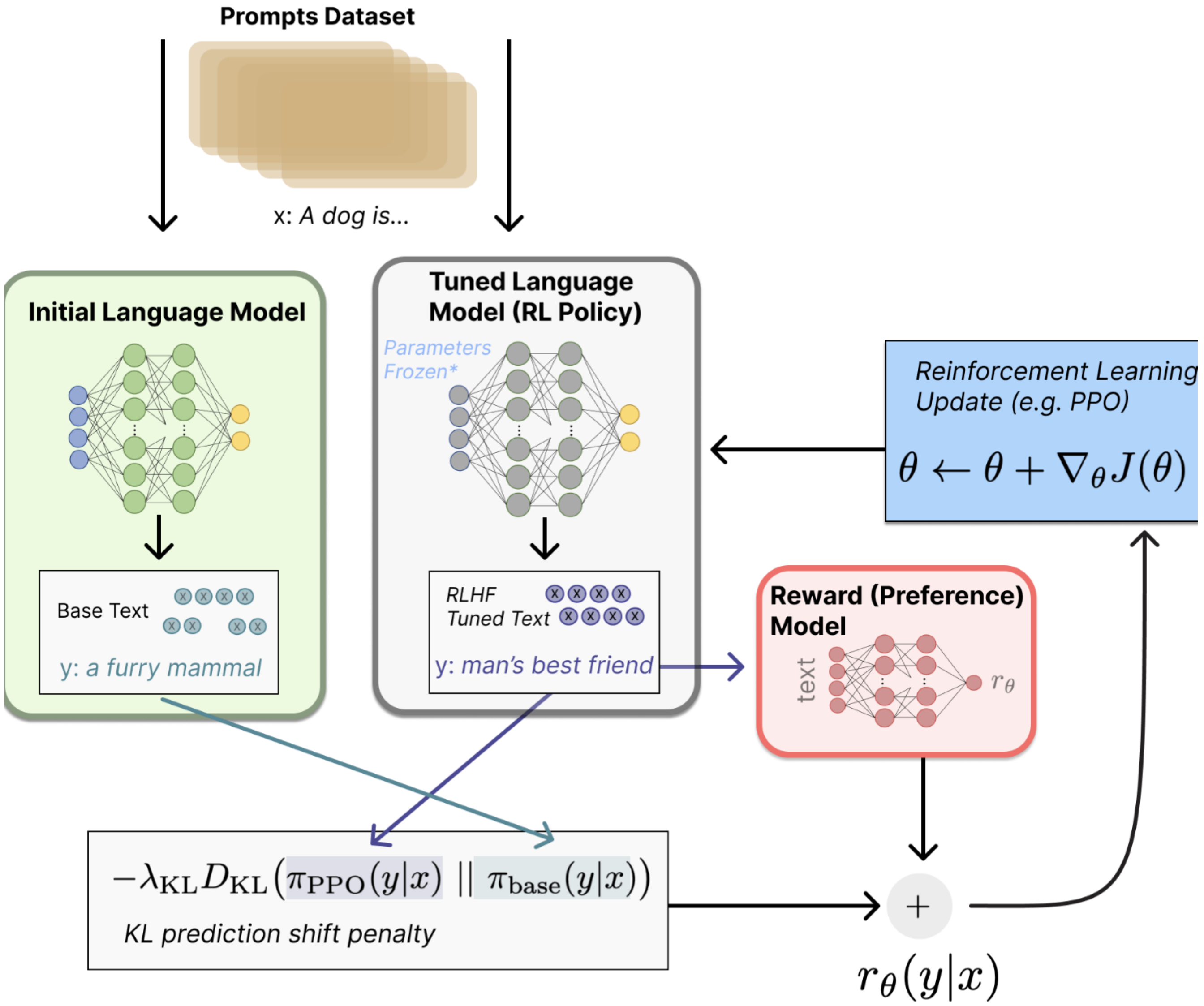
urce: https://medium.com/aimonks/mastering-adaptability-exploring-model-agnostic-meta-learning-maml-996796c890f0

General Intelligence

What properties do we want from a generally intelligent robot?

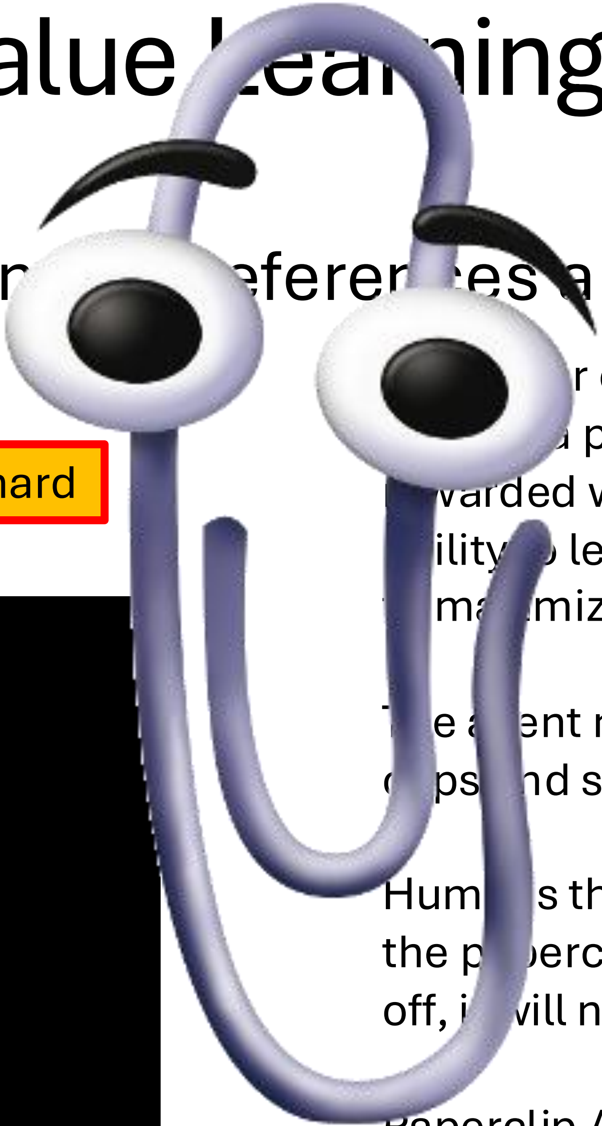
1. Adapt to new environments and tasks quickly
2. **Goal alignment and value learning**
3. Work with multi-modal data
4. Safe exploration and failure recovery
5. Long term memory and experience integration
6. Explainability and Interpretability

RLHF is a way to perform alignment



Alignment and Value L... ning

How can robots learn hu... efer... es and what we want them to do?



Specifying reward functions is hard

Positive reward for surviving, negative reward for losing

... r clip parable:
... paper clip factory and train an agent that is ... varded when it produces a paper clip. We give it the ... ility ... learn even better strategies. The agent wants ... ma... mize reward.

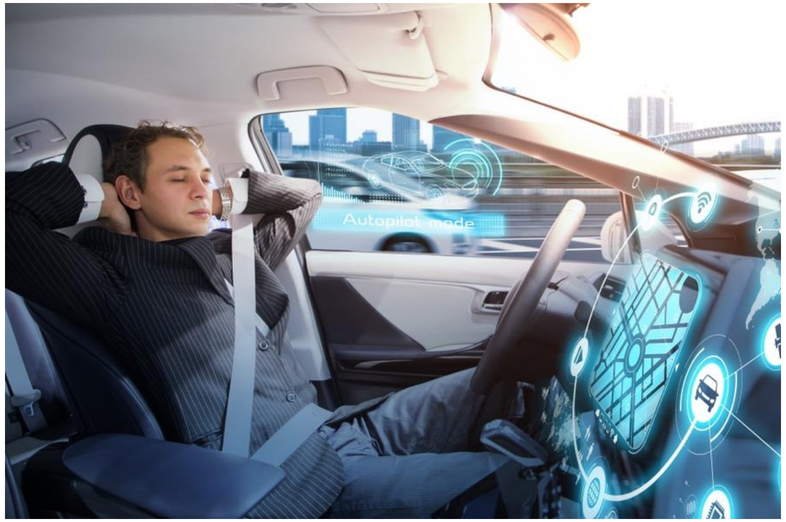
... e a... ent needs to secure more resources for paper ... ps ... nd starts strip mining.

Hum... s think strip mining is bad, and want to turn off the p... perclip AI. The paperclip AI knows if it is turned off, i... will no longer get rewards.

Paperclip AI wipes out humanity so that it can continue to make paperclips.

Learning Human Preferences

Given expert demonstration data, how can we learn to imitate the expert policy?



Imitation Learning

Behavior Cloning seeks to *imitate* the expert policy

Given a dataset of (state, action) pairs (i.e., trajectories), use supervised learning to learn a policy $\pi_\theta(s)$

Potential Issues:

What happens when we encounter a state we don't have data for?

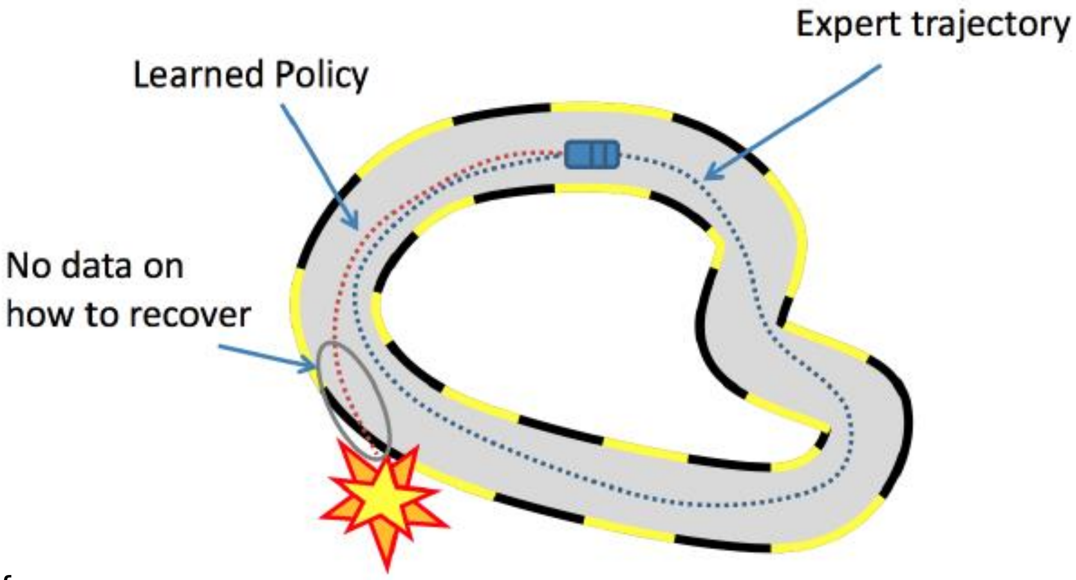
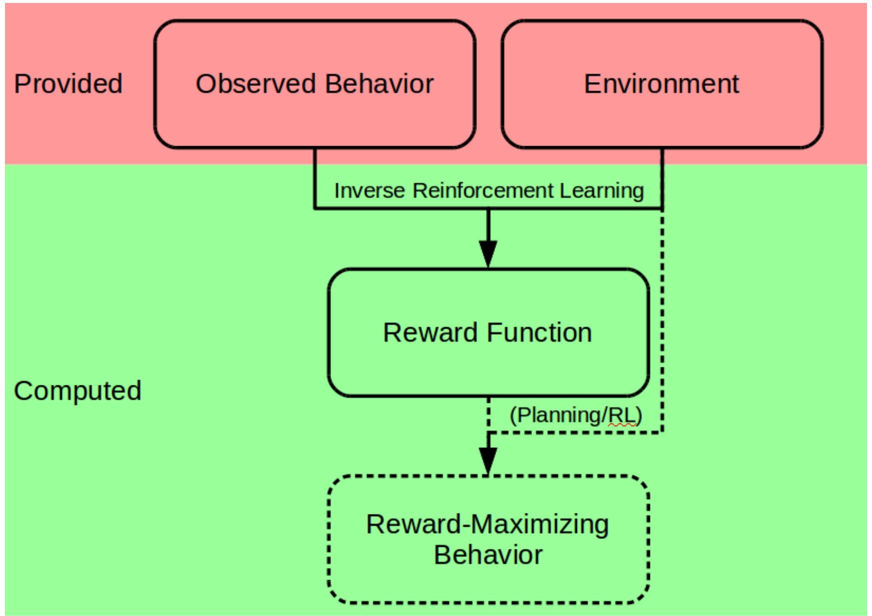


Image source: https://web.stanford.edu/class/cs234/slides/lecture7.pdf

Inverse Reinforcement Learning

IRL seeks to learn the reward function of the expert

Given a dataset of (state, action) pairs (i.e., trajectories), learn a reward model. Then, use RL with the learned reward model to learn a policy



Potential Issues:

There is more than one reward function that could reproduce the given trajectories. How do we decide what the correct reward function is?

Image source: https://dkasenberg.github.io/inverse-reinforcement-learning-rescue/

General Intelligence

What properties do we want from a generally intelligent robot?

1. Adapt to new environments and tasks quickly
2. Goal alignment and value learning
3. **Work with multi-modal data**
4. Safe exploration and failure recovery
5. Long term memory and experience integration
6. Explainability and Interpretability

Working with Multi-Modal Data

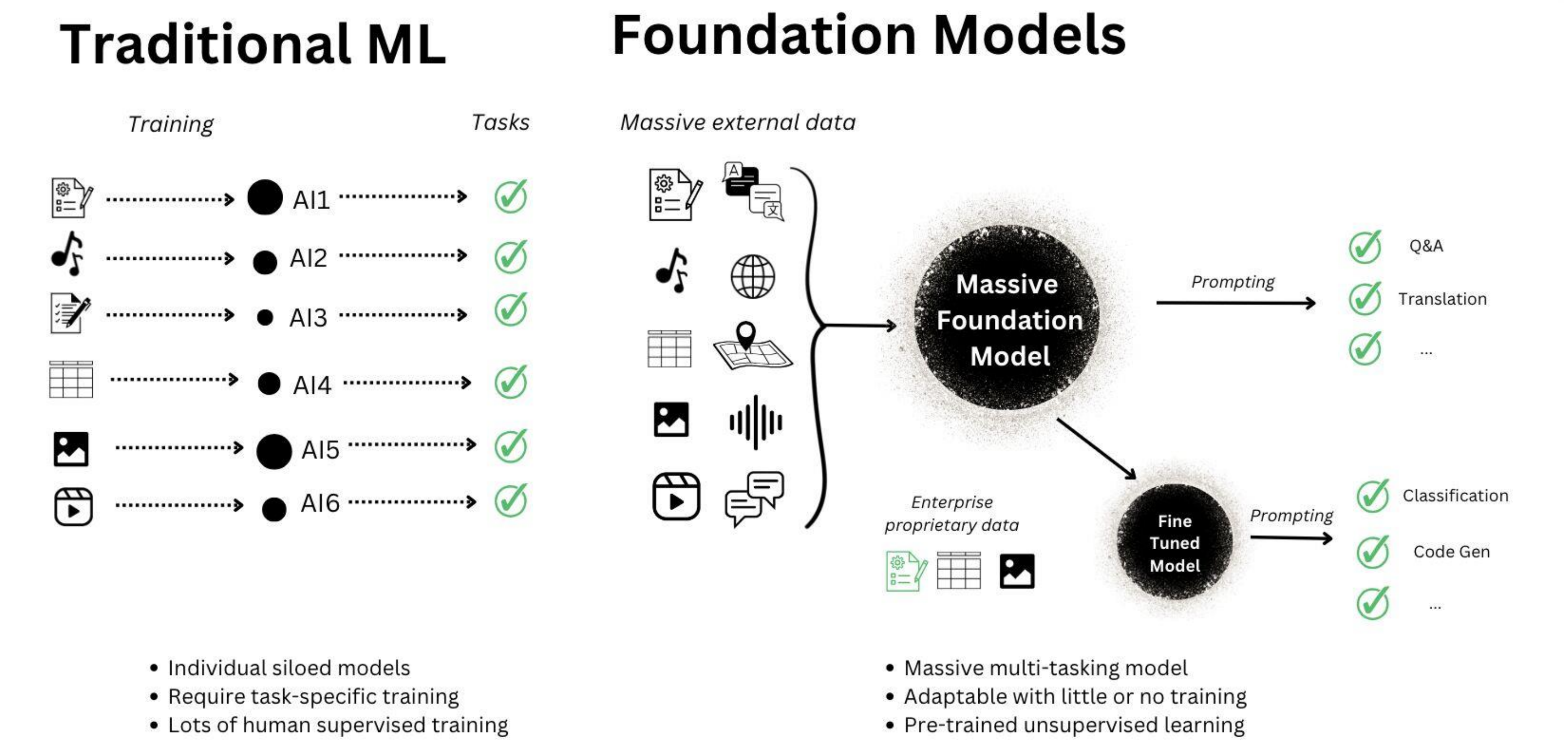
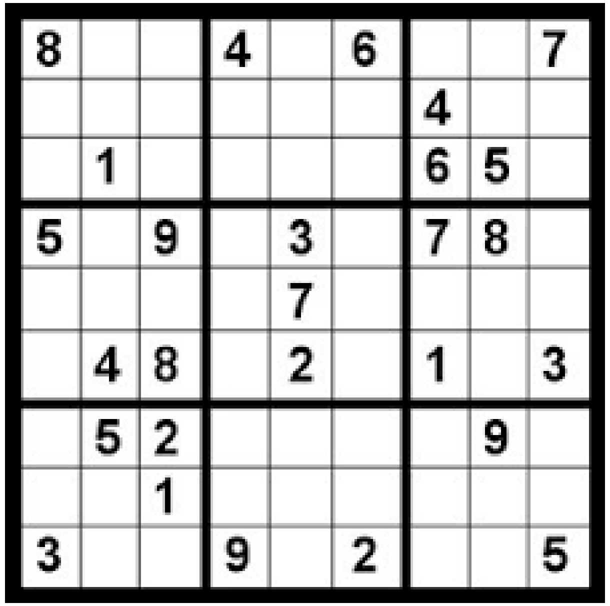


Image source: https://humanloop.com/blog/foundation-models

Representations Aren’t Free

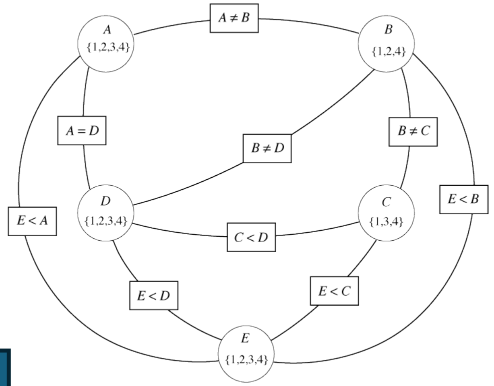
Classical AI



This is represented as a 2D matrix

Rules or Constraints

These are hardcoded



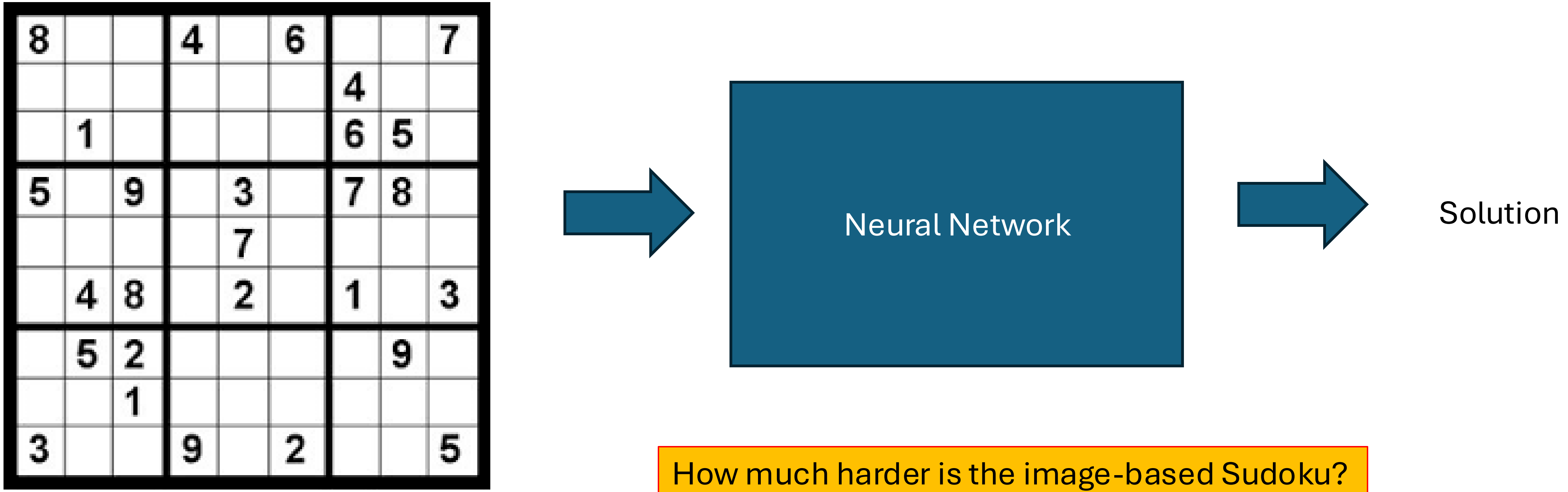
Optimization

This is very general, but requires a very specific representation

Solution

Representations Aren’t Free

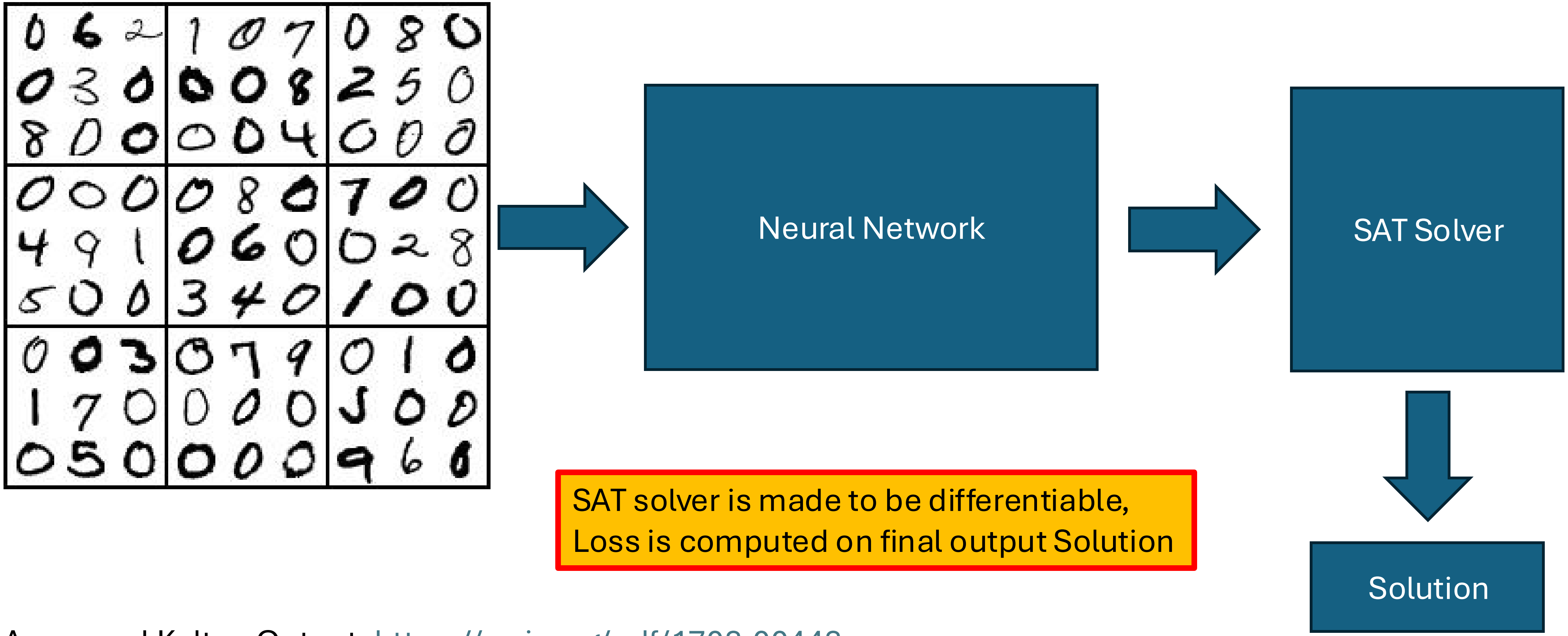
Naïvely applying Deep Learning



How much harder is the image-based Sudoku?

What if your network doesn’t start with the ability to recognize digits (i.e., wasn’t pre-trained on MNIST).

Differentiable Optimization Functions as Layers



Amos and Kolter, Optnet: https://arxiv.org/pdf/1703.00443

General Intelligence

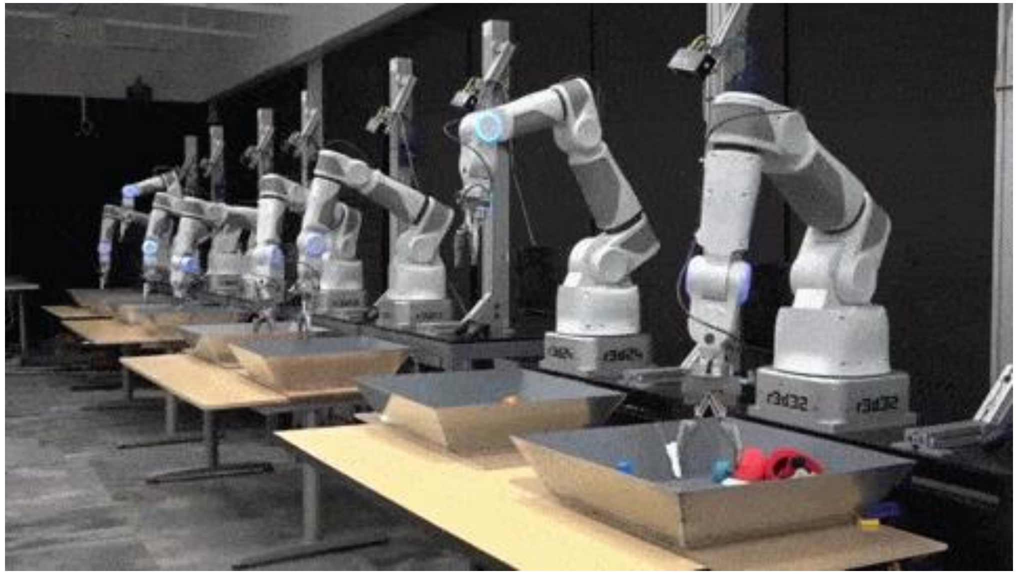
What properties do we want from a generally intelligent robot?

1. Adapt to new environments and tasks quickly
2. Goal alignment and value learning
3. Work with multi-modal data
4. **Safe exploration and failure recovery**
5. Long term memory and experience integration
6. Explainability and Interpretability

Safe Exploration

Two issues with using the real world to collect trajectories for RL:

1. The real world is too slow… (simulations are fast to run)
2. Before Robots learn a "good" policy, they may take dangerous actions



Google Research

How do we still encourage safe exploration?

Failure Recovery?

Easiest way: Train failure recovery policy through a separate process (i.e., reward standing up). Learn when to switch policies

Harder: Learn a single general policy to optimize desired objective

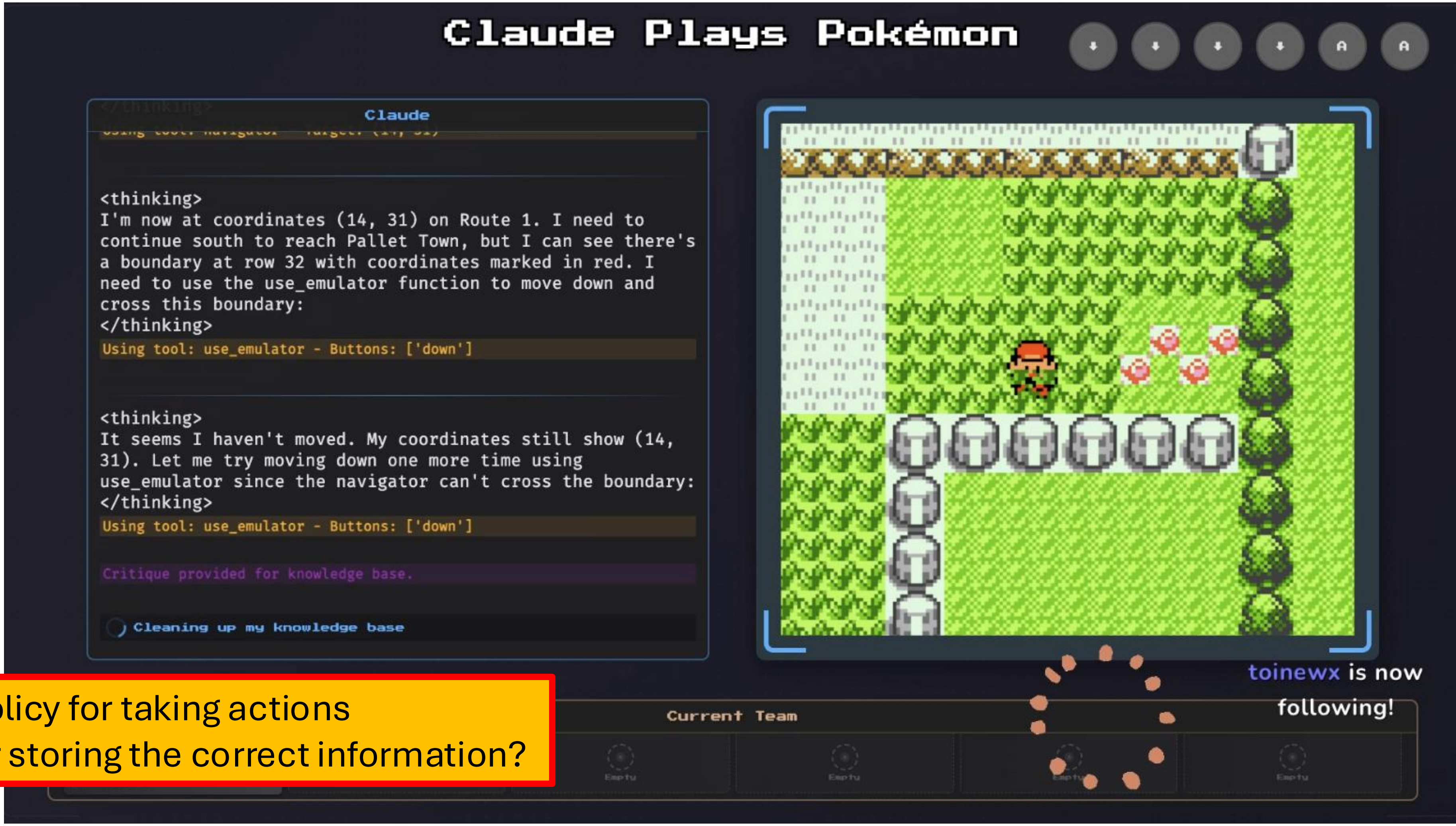
General Intelligence

What properties do we want from a generally intelligent robot?

1. Adapt to new environments and tasks quickly
2. Goal alignment and value learning
3. Work with multi-modal data
4. Safe exploration and failure recovery
5. **Long term memory and experience integration**
6. Explainability and Interpretability

Long-Term Memory

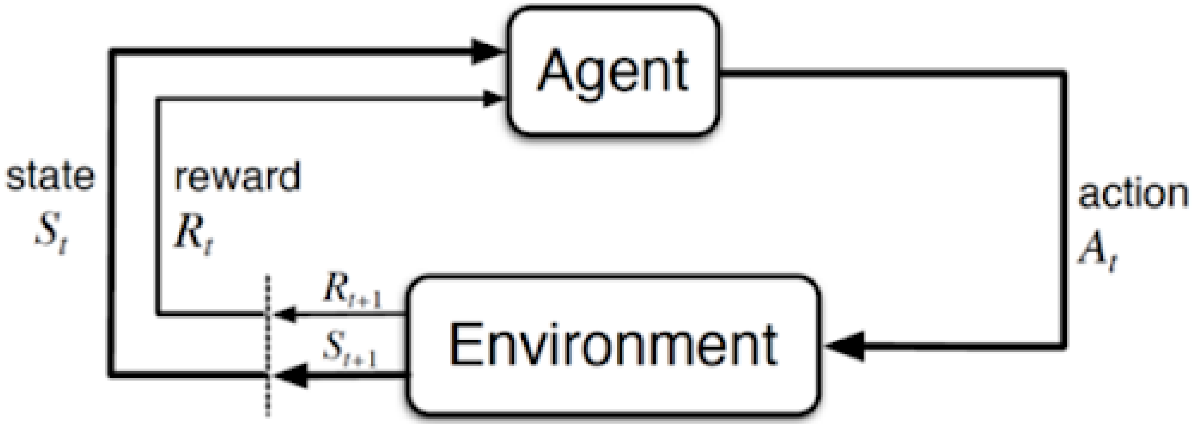
1. What should be stored in long term memory?
2. How should it be stored?
3. How can it be accessed?



Reward signal helps to learn a policy for taking actions
How can we reward our agent for storing the correct information?

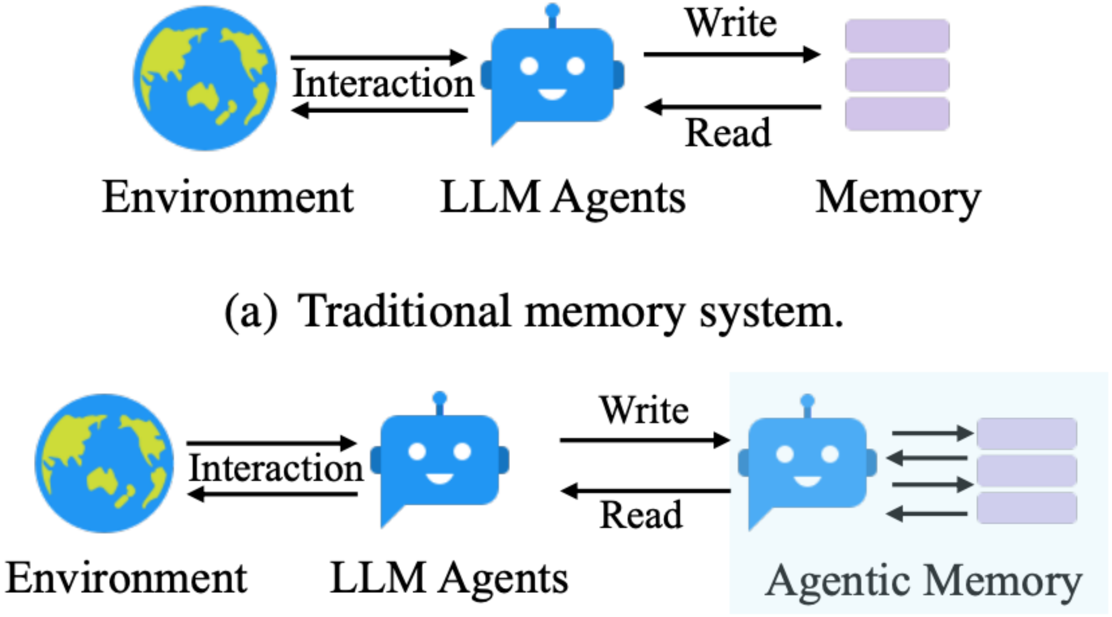
Claude plays pokemon

Imbuing Agents with Long Term Memory



Can we model everything as an MDP?

- *Markov* implies the next state depends only on the current state and action taken.
- If the next state depends on the entire history of states, it is a partially observable MDP (POMDP)



Xu et al., A-MEM: Agentic Memory for LLM Agents: https://arxiv.org/pdf/2502.12110

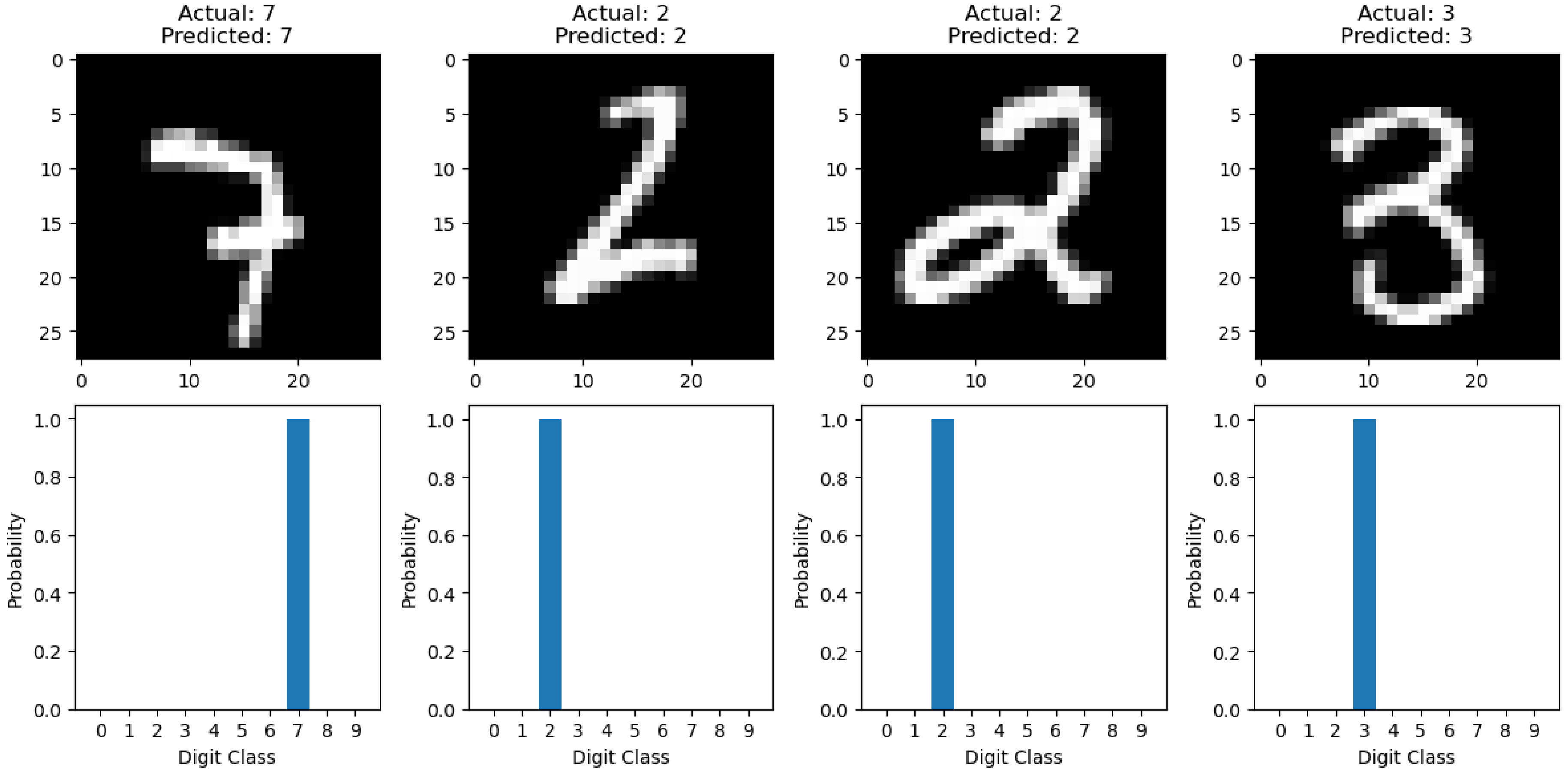
General Intelligence

What properties do we want from a generally intelligent robot?

1. Adapt to new environments and tasks quickly
2. Goal alignment and value learning
3. Work with multi-modal data
4. Safe exploration and failure recovery
5. Long term memory and experience integration
6. **Explainability and Interpretability**

Uncertainty in Deep Learning

How certain of a prediction is a Neural Network?



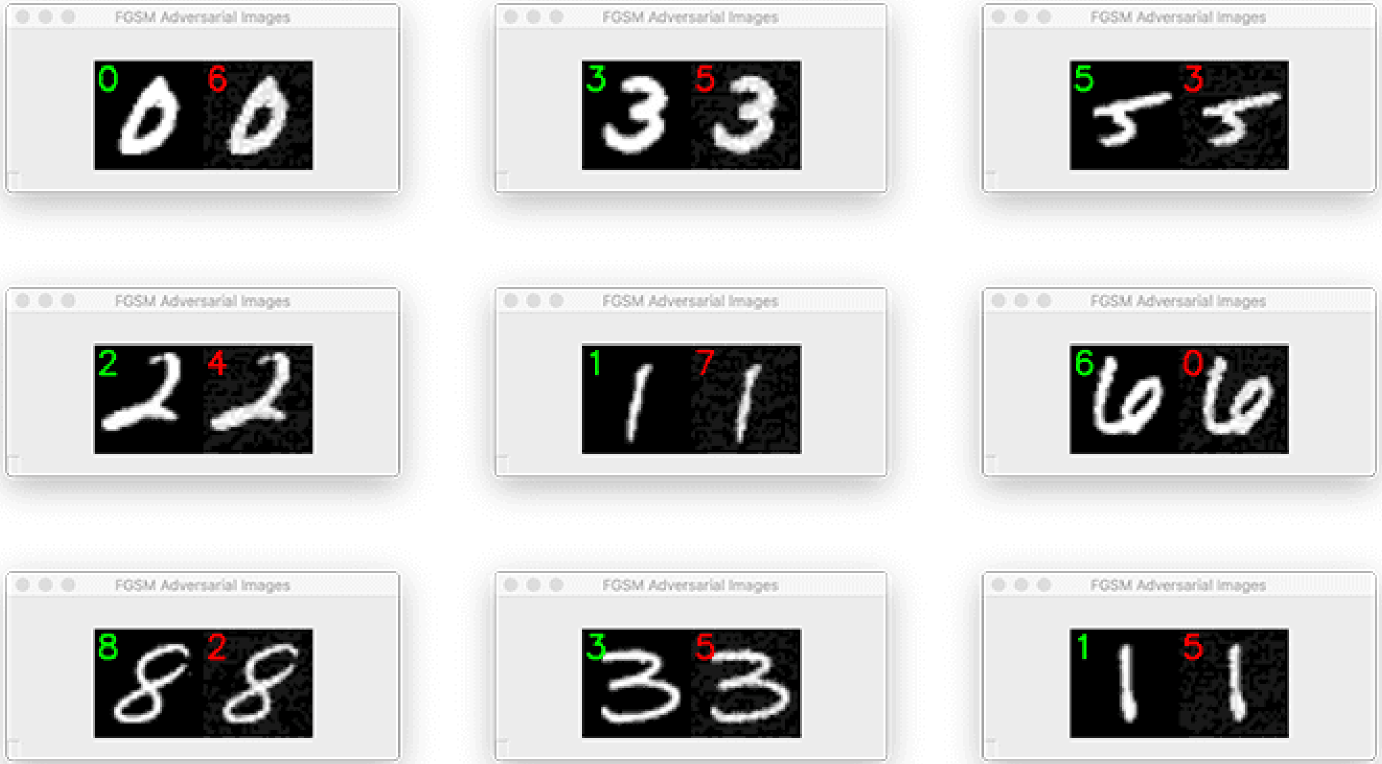
Uncertainty in Deep Learning

How certain of a prediction is a Neural Network?

Neural Networks can be confidently incorrect!

Why do Adversarial Attacks work?

Neural Networks are penalized for uncertainty during training



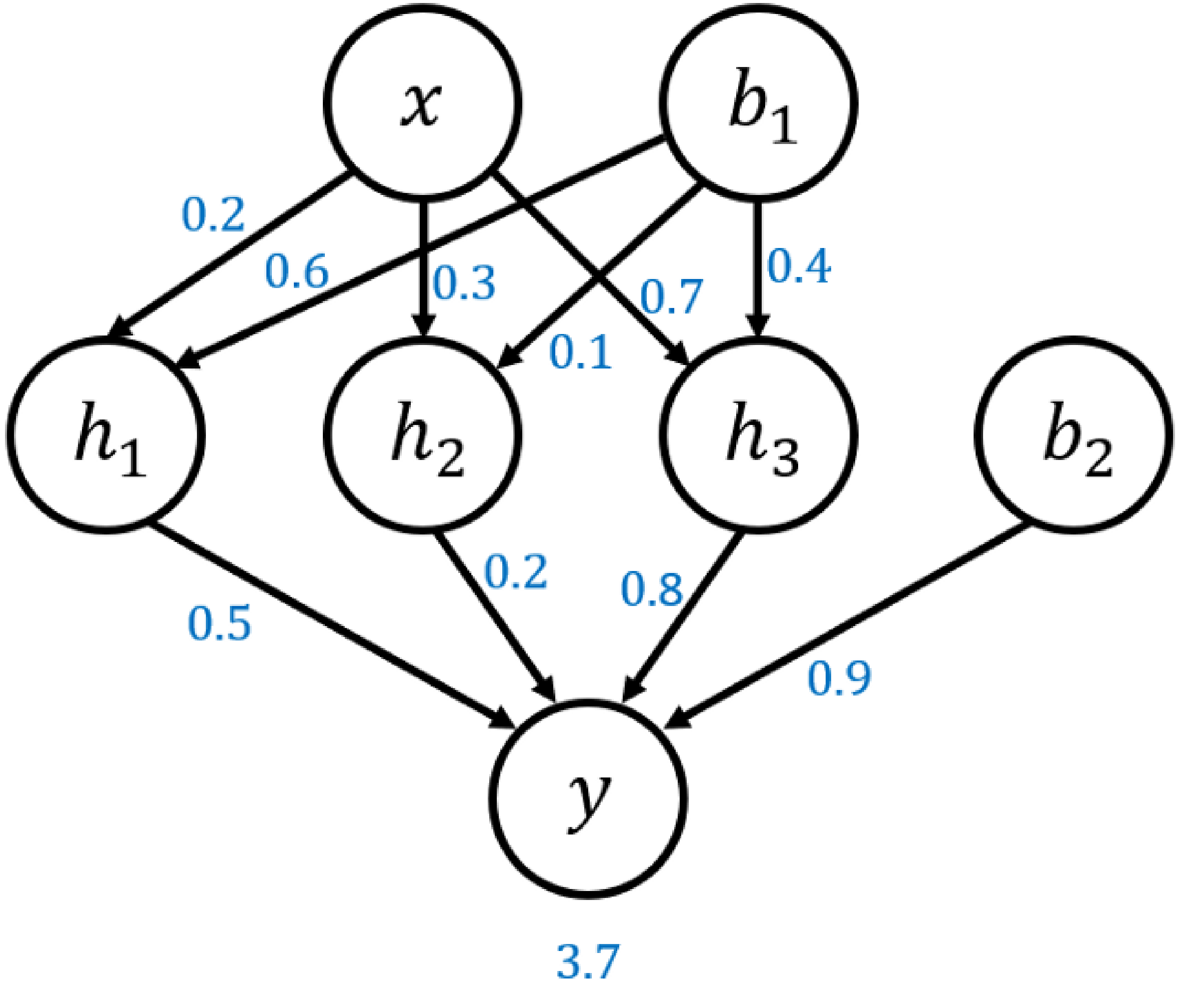
Source: https://pyimagesearch.com/2021/03/01/adversarial-attacks-with-fgsm-fast-gradient-sign-method/

Uncertainty in Deep Learning

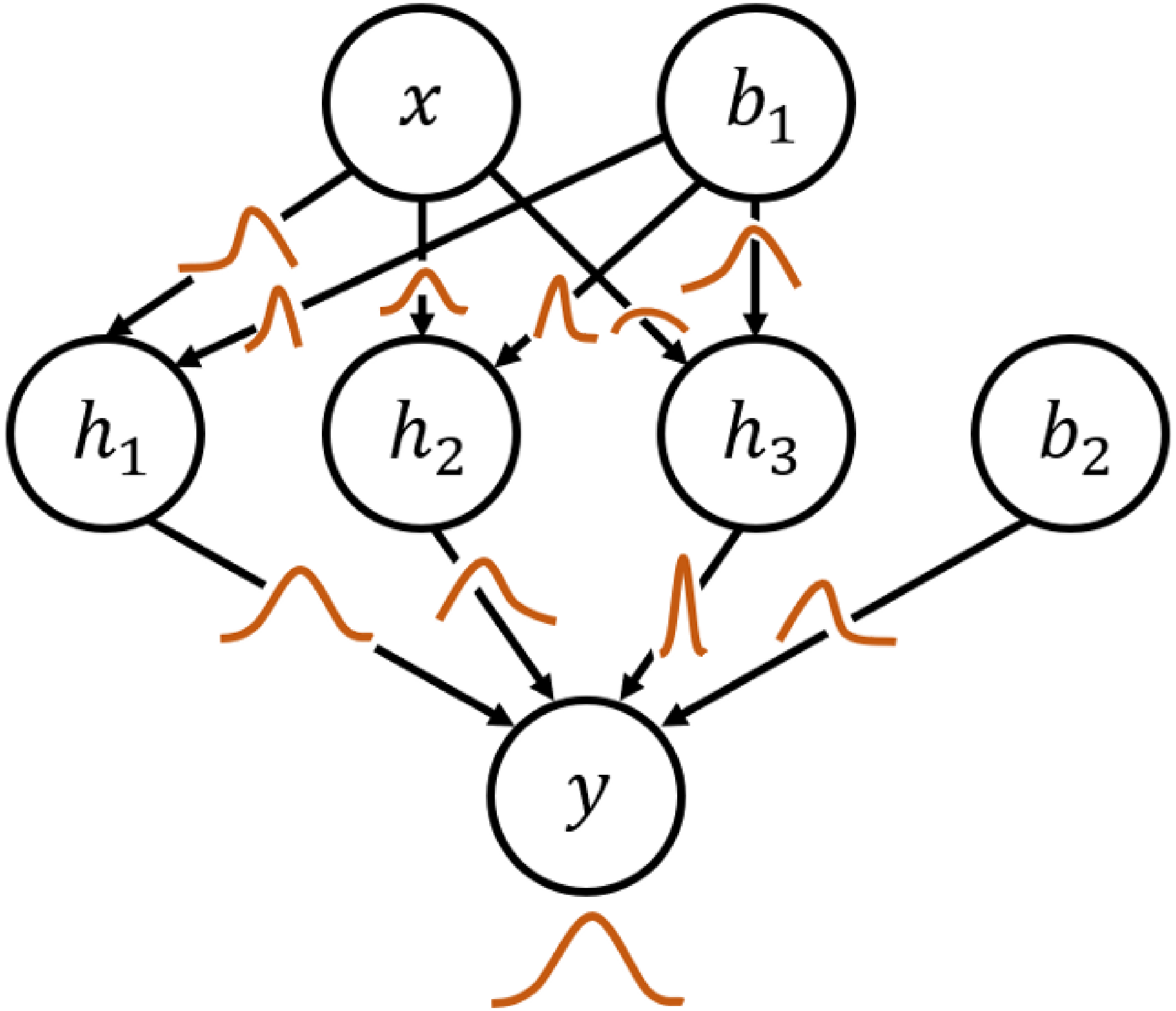
Help! I'm uncertain of what action to take. Please take over.

Not only want the prediction, but also an estimate of the uncertainty of that prediction

Standard Neural Network



Bayesian Neural Network



Every parameter is a distribution (i.e., $\mathcal{N}(\mu, \sigma^2)$), output is a distribution over labels, with quantified variance

Interpretability

LIME: What parts of an image contribute to a model's predictions?

Local Interpretable Model-agnostic Explanations (LIME)

1. Use image segmentation to group pixels together into super pixels
2. Run predictions on image with some super-pixels masked out
3. Train a simple classifier to predict which super-pixels were most important

LIME

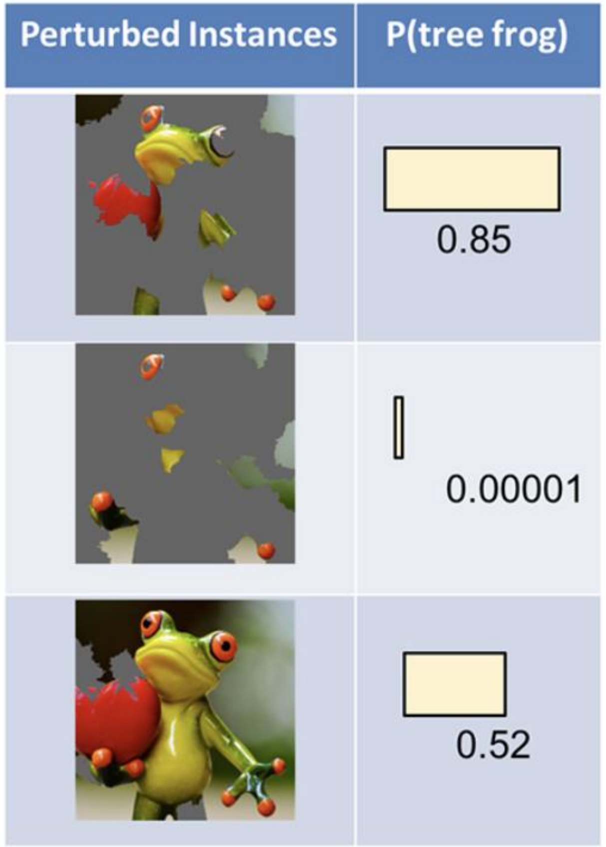
1. Separate Image into Super-pixels using image segmentation

Original Image

Interpretable Components

LIME

2. Run classification with some super-pixels masked



How much does the presence (or absence) of pixels affect the prediction?

LIME

3. Train simple regression model to determine feature weighting of the super-pixels