CS2109S Tutorial 7

Convolutional Neural Networks

(AY 25/26 Semester 2)

March 26, 2026

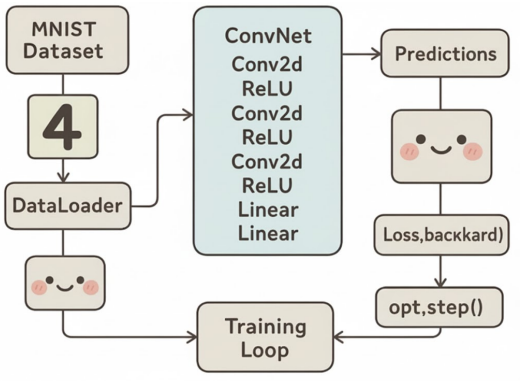
(Prepared by Benson and Suhail)

Admin Info

- No tutorials next week! (NUS Well-Being Day)
  - I will upload a (short) prerecorded video on RNNs – plus a tiny bit of extras (token embeddings and probabilistic decoding).
  - Please watch it before Tutorial 9 (transformers and LLMs).
- For your well-being in Week 13, please work on the capstone project.
  - You should be able to work on the entire project now (after learning how to implement CNNs).

Contents

PyTorch: Training Loop




```python
class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.c1 = nn.Conv2d(1, 16, 3, 1)
        self.c2 = nn.Conv2d(16, 32, 3, 1)
        self.c3 = nn.Conv2d(32, 64, 3, 1)
        self.fc1 = nn.Linear(64*2*2, 128)
        self.fc2 = nn.Linear(128, 10)
        self.pool = nn.MaxPool2d(2)

    def forward(self, x):
        x = self.pool(F.relu(self.c1(x)))
        x = self.pool(F.relu(self.c2(x)))
        x = self.pool(F.relu(self.c3(x)))
        x = torch.flatten(x, 1)
        x = F.relu(self.fc1(x))
        return self.fc2(x)

# Training
opt = torch.optim.Adam(model.parameters())
loss_fn = nn.CrossEntropyLoss()

for epoch in range(1):
    for xb, yb in train:
        pred = model(xb)
        loss = loss_fn(pred, yb)
        opt.zero_grad()
        loss.backward()
        opt.step()
    print(f"Loss: {loss.item():.4f}")
```

Why CNNs/RNNs?

We use Neural Networks to solve many types of problems:

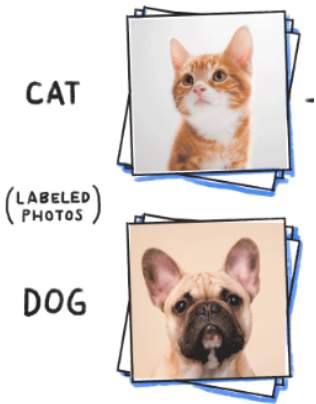
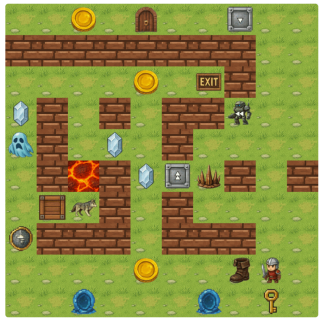


Image Recognition



Playing Grid Adventure

You

Write a code to play Grid Adventure

ChatGPT

Sorry, I'm as screwed as you.

(in code)

Why CNNs/RNNs?

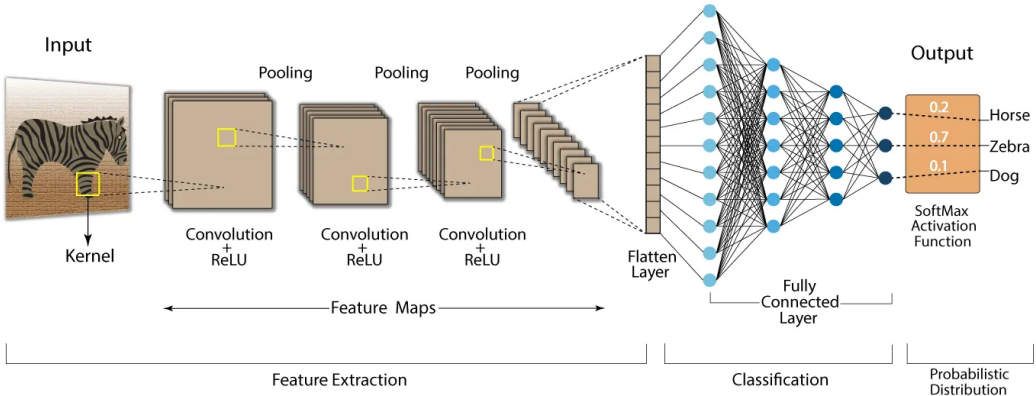
Should we simply whack a *very very very* deep Neural Network?

Problem context is important.

Idea: Exploit characteristics in the problem domain.

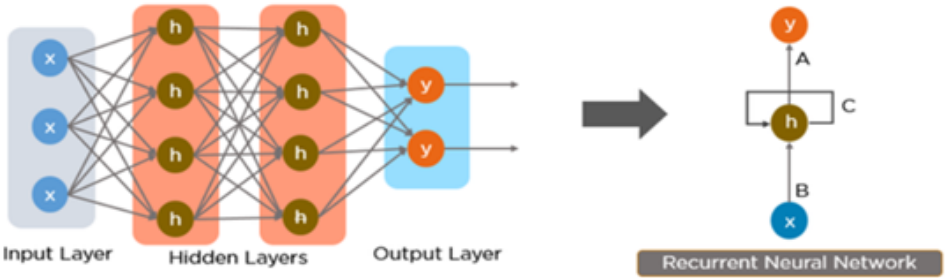
Why CNNs/RNNs?

Convolutional Neural Networks: Exploiting **locality**.



Why CNNs/RNNs?

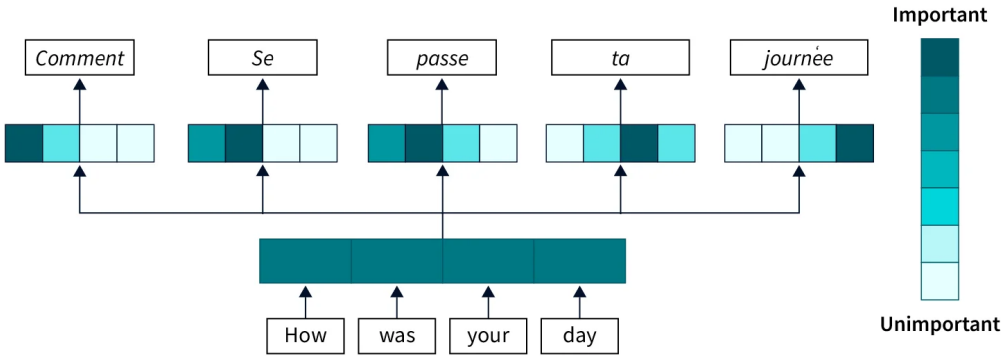
Recurrent Neural Networks: Exploiting **sequential nature**.



Why CNNs/RNNs?

Attention Mechanism and Transformers:

- How to be less forgetful?
- Focusing on the important bits.



How good is an architecture?

Extra Slide

On Limitations of the Transformer Architecture

Prompt: If Amy is to the southwest of Ben, Cindy is to the northeast of Amy and directly north of Ben, is Amy further from Ben or Cindy?

GPT 3.5: Ben

GPT 4: Ben

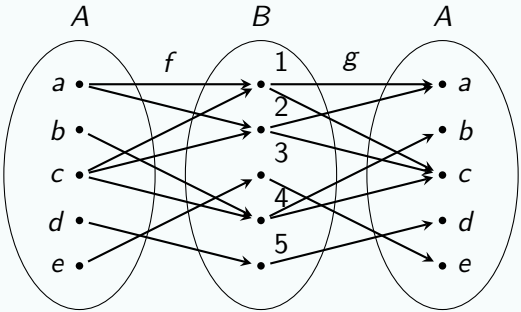
Bard: Ben

Correct answer: Cindy

How good is an architecture?

Extra Slide

Functional composition: Given two functions f and g, find the function $g \circ f$ (that is, find $g(f(x))$ given x.).



How good is an architecture?

Extra Slide

Theorem

Consider a function composition problem with input domain size $|A| = |B| = |C| = n$, and an H-headed transformer layer $\mathcal{L}$ with embedding dimension d and computation precision p, and assume that $H(d+1)p < n \log n$. Then L cannot solve correctly the function composition problem. In particular, if $R = n \log n - H(d+1)p > 0$, then the probability, over all possible functions and queries, that $\mathcal{L}$ answers the query incorrectly is at least $\frac{R}{3n \log n}$.

The proof relies on **communication complexity**, an algorithmic tool.

Q1. ConvNets

(a) Get the output feature map from this convolution and write down its output dimensions if the kernel moved with a stride of 1×1. What do you think is the purpose of this filter?

0.1	0.2	0.1	0.1	0	0
0.8	0.9	1	1	0.9	0
1	1	1	1	1	1
0.9	1	1	0.8	1	0.6
0	0.1	0.1	0.2	0	0.1
0	0.8	1	0.9	0.6	0.2

1	0	-1
1	0	-1
1	0	-1

-0.2	0	0.2	1.1
-0.3	0.1	0.1	1.2
-0.2	0.1	0.1	0.3
-1.2	0	0.5	1

Purpose: Vertical edge detection

Q1. ConvNets

(b) Apply 2×2 pooling kernel with a stride of 2. Compute the resulting output feature maps for both (a) Max pooling and (b) Average pooling.

-0.2	0	0.2	1.1
-0.3	0.1	0.1	1.2
-0.2	0.1	0.1	0.3
-1.2	0	0.5	1

0.1	1.2
0.1	1

Max Pooling

-0.1	0.65
-0.325	0.475

Average Pooling

Q1. ConvNets

(c) We feed a single image to the network. Each image has 3(C) channels (RGB) with a height and width of 224×224 ($H = 224, W = 224$). Here our input is a 3-dimensional tensor ($H \times W \times C$). The first layer of the big CNN is a convolutional Layer with 96(C_1) kernels which has a height and width of 11, and each kernel has the same number of channels as the input channel. The stride is 4 and no padding is used. Calculate the output size $H_1 \times W_1 \times C_1$ after the first convolutional Layer.

- Height/Width: $0, 4, 8, 12, \ldots, 212$ ($\leq 224 - 11$)
  $$\left\lfloor \frac{H - K}{S} \right\rfloor + 1 = \left\lfloor \frac{224 - 11}{4} \right\rfloor + 1 = 54$$
- Channels: $C_1 = 96$

Q1. ConvNets

(d) In most of Deep Learning libraries such as PyTorch, images are fed together as a batch B. B can take values such as 8, 16, 32, 64. Comment on the output shape if we feed the large CNN in part (b) with a batch of images. What are the advantages of using a batch of images rather than a single image?

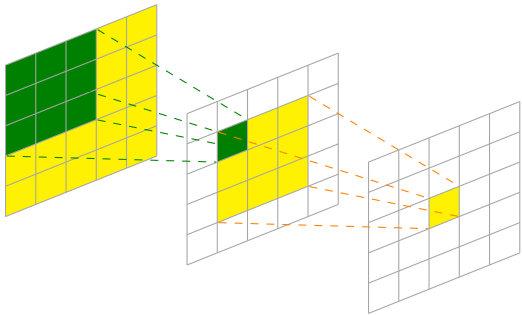
- Output shape: $B \times H_1 \times W_1 \times C_1$
- More parallelism (GPU) friendly.

Q2. Receptive Fields

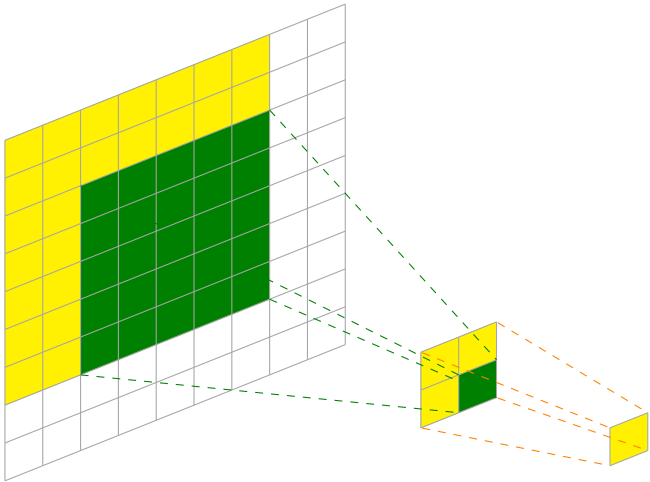
(a) Consider a 2-layer convolutional neural network (CNN) with the following configuration:

- An initial input matrix with size 9×9
- Layer 1: Uses a kernel of size 5×5 and a stride of 2×2.
- Layer 2: Uses a kernel of size 2×2 and a stride of 1×1.

Calculate the receptive field sizes of neurons in the first and second layers.



Q2. Receptive Fields



- Layer 1: $5x5$
- Layer 2: $7x7$

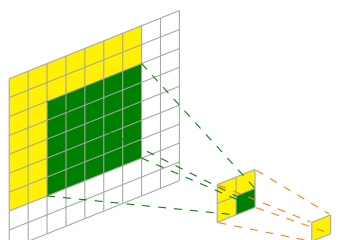
Q2. Receptive Fields

More generally, we can derive the following recursive formulas to calculate the receptive field for each layer in the CNN:

$$r_i = r_{i-1} + (\text{kernel}_i - 1) \times j_{i-1}$$

$$j_i = j_{i-1} \times \text{stride}_i$$

- For $i = 0$
  - $r_0 = 1$, $j_0 = 1$
- For $i = 1$
  - $r_1 = 1 + [(5 - 1) \times 1] = 5$
  - $j_1 = 1 \times 2 = 2$
- For $i = 2$
  - $r_2 = 5 + [(2 - 1) \times 2] = 7$
  - $j_2 = 2 \times 1 = 2$



Q2. Receptive Fields

(b) How does an increase in the receptive field affect the performance of a Convolutional Neural Network (CNN)?

- Each neutron "sees" a larger portion of the input image.
- Require larger kernel sizes or more layers $\Rightarrow$ expensive in compute.

Q3. Parameter Efficiency and Translation Invariance

Models

- **Model A**: One convolutional layer with $f^{[1]} = 5$, $s^{[1]} = 1$, $p^{[1]} = 0$, $n_c^{[1]} = 8$.
- **Model B**: Flatten the input and use one fully connected layer with $28 \times 28 \times 8 = 6272$ output units.

(a) Compute the total number of parameters (including biases) in both models. Input dimension: $32 \times 32 \times 3$, Output dimension: $28 \times 28 \times 8$.

- Model A: $5 \times 5 \times 3 \times 8 + 8 = \boxed{608}$.
- Model B: $(32 \times 32 \times 3) \times (28 \times 28 \times 8) + (28 \times 28 \times 8) = \boxed{19,273,856}$.

Q3. Parameter Efficiency and Translation Invariance

Models

- **Model A**: One convolutional layer with $f^{[1]} = 5$, $s^{[1]} = 1$, $p^{[1]} = 0$, $n_c^{[1]} = 8$.
- **Model B**: Flatten the input and use one fully connected layer with $28 \times 28 \times 8 = 6272$ output units.

(b) Consider a feature such as a vertical edge that may appear at different spatial locations in an image.
Which model exhibits **translation invariance**?

- **Translation invariance** means that the same feature can be detected regardless of where it appears in the image.
- Model A: Due to **parameter sharing**, the kernel is applied at all spatial locations $\Rightarrow$ the edge must be detected regardless of where it appears.
- Model B: The model uses a separate set of weights in each location, leading to **location-specific** representations.

Q4. Transfer Learning using Pretrained CNNs

ImageNet pretraining: early layers – edges, corners, textures; intermediate layers – shapes and object parts; later layers – task-specific features.
Transfer learning:

- **Feature extraction**: Freeze pretrained layers, remove the original last layer, i.e. the predictor, then add a new predictor, potentially with different number of output classes; train only the new predictor.
- **Fine-tuning**: Replace the predictor and allow some or all earlier layers to update on the new task.

(a) How is a pretrained CNN useful?

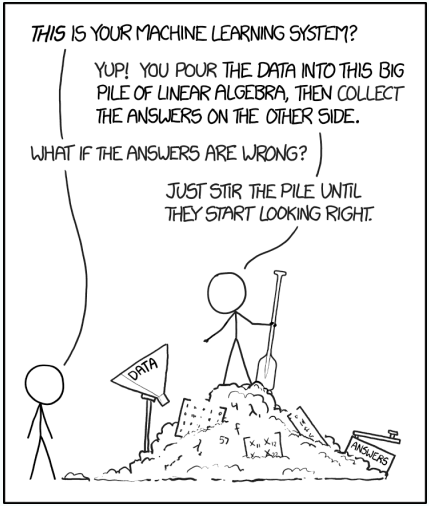
- It has already learned a hierarchy of general visual structure such as edges, corners, textures, contours, spatial hierarchies. They are **transferrable**.

(b) When is feature extraction or fine-tuning preferable?

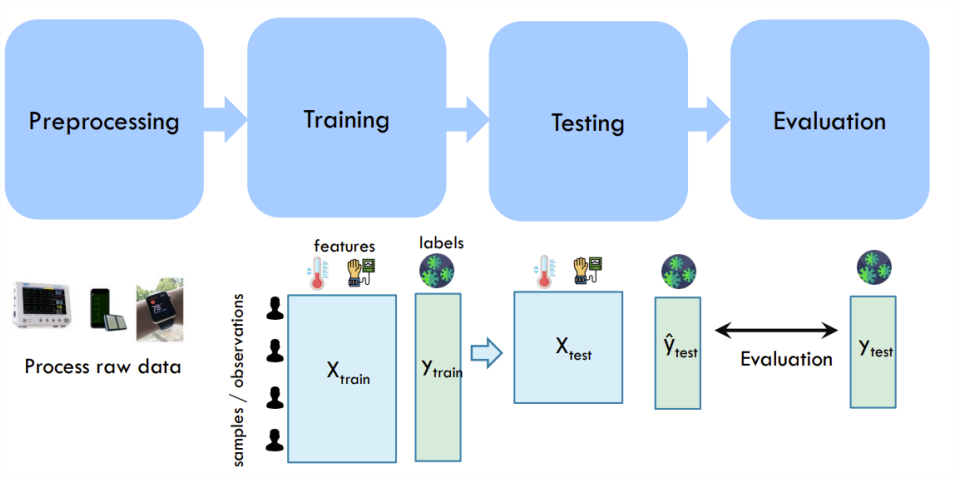
- Feature extraction: When the task is similar (lower training time).
- Fine-tuning: When the task differs significantly.

Extra. ML Training Pipeline

The Reality


THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

ML Training Pipeline



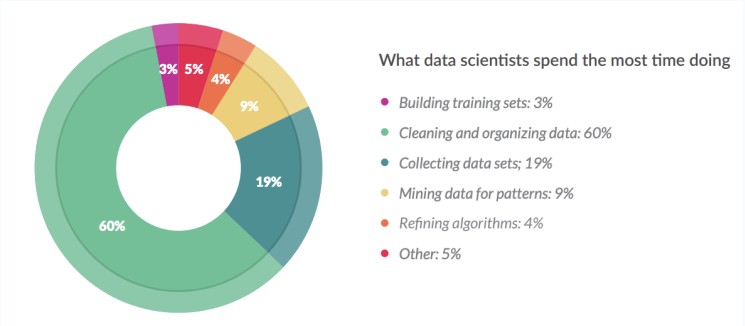
(Adapted from CS4225 Materials)

ML Training Pipeline: Preprocessing

Extra Slide

NY Times: data scientists spend 50-80% of their time on data preparation.

- "Garbage in, garbage out"
- Google AI (SIGCHI 2021): 92% of the analysed high-stakes AI projects practitioners report negative downstream effects from data issues



(Adapted from CS4225 Materials)

ML Training Pipeline: Preprocessing

Extra Slide

Handling Missing Values:

- Information was not collected or lost at random
- **Imputation**: E.g. based on the mean / median of that attribute; Fitting a regression model to predict that attribute given other attributes (Sklearn libraries)
- **Dummy variables**: optionally insert a column which is 1 if the variable was missing, and 0 otherwise

	A	B
0	1.0	1.0
1	2.0	7.0
2	NaN	7.0
3	4.0	7.0
4	5.0	NaN

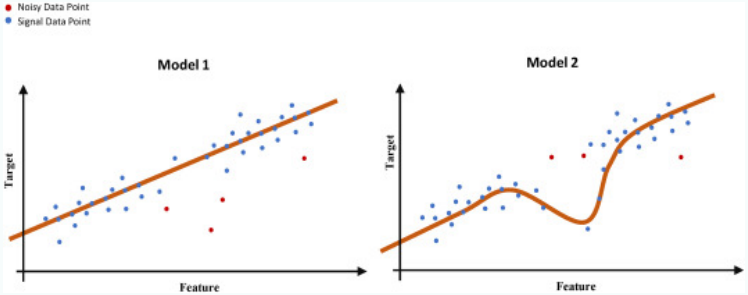
	A	B
0	1.0	1.0
1	2.0	7.0
2	3.0	7.0
3	4.0	7.0
4	5.0	7.0

ML Training Pipeline: Preprocessing

Extra Slide

Detecting Outliers / Understanding Relationships:

- Unreliable data sources ⇒ "Unreasonable data" / Some features are simply noise.



ML Training Pipeline: Preprocessing

Encoding: Convert categorical feature to numerical features.

Many choices out there!

ML Training Pipeline: Preprocessing

Extra Slide

Feature Selection / Engineering:

- Is there class imbalance? (for classification problems)
- Are there duplicate samples?
- Removing uninformative features: Linearly dependent features (makes training unstable), Low or no correlation with the target.
- Principal component analysis: Retain most of the explained variance, optimizing training time.
- Combining features / Creating new features.

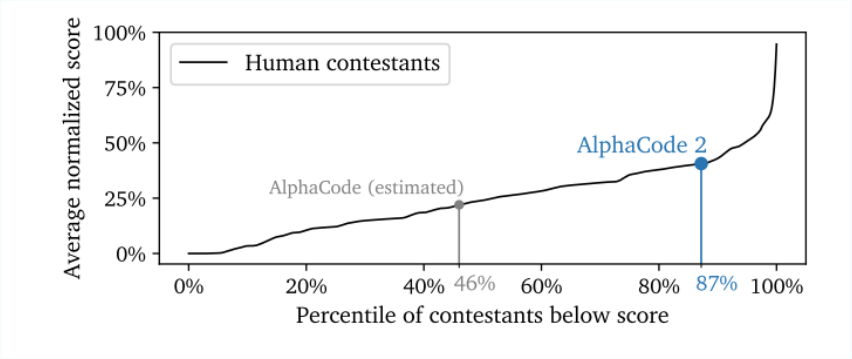
ML Training Pipeline: Preprocessing

Extra Slide

Important to carefully perform **train-test-split**!

Test Leakage: AlphaCode 2 (2023) (A relevant discussion online)

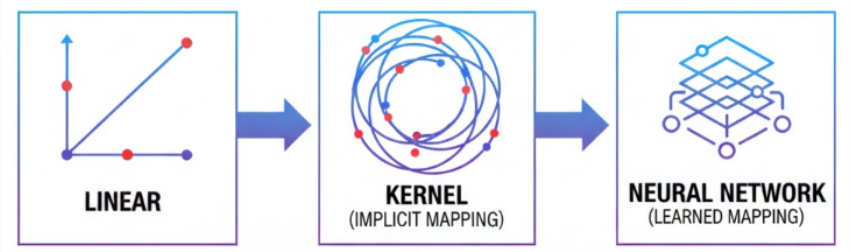
- Happens very easily! E.g. preprocessing before train-test-split, time leakage (some features are not supposed to be known before prediction)



ML Training Pipeline: Training

Extra Slide

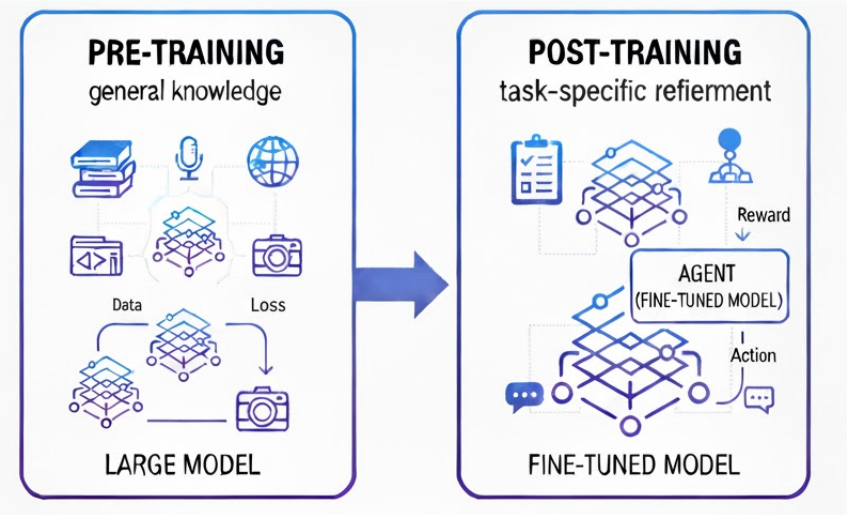
Decide the **hypothesis class** $\mathcal{H}$.



- **Inductive Learning Assumption**: Any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples.

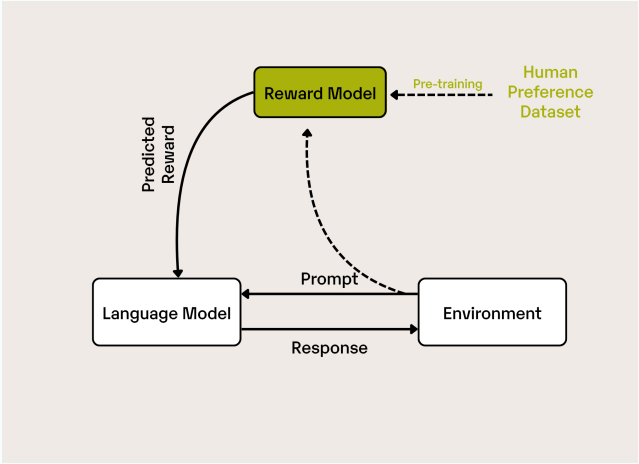
ML Training Pipeline: Training

The **pre-training** + **post-training** paradigm (esp. modern LLMs):



ML Training Pipeline: Training

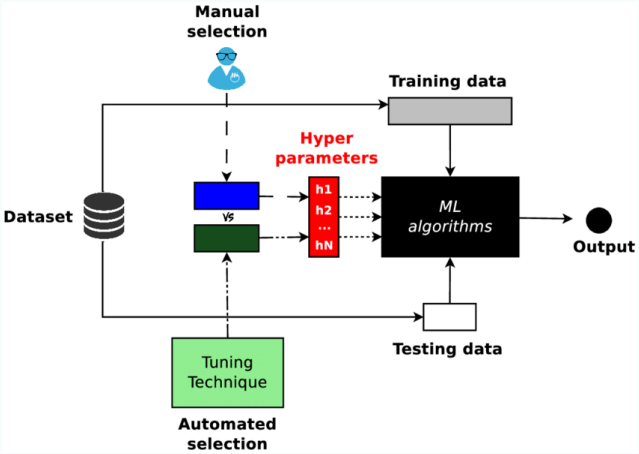
Post-training using **Reinforcement Learning** (RL):



ML Training Pipeline: Evaluation

Extra Slide

Hyperparameter Tuning: Many methods, e.g. grid search, random search, Bayesian optimization. (Useful link)



Conclusion

- **Spend time on data preparation and feature engineering**
- **Don't ignore domain specific knowledge**
- **Simple models can work well** (Occam's Razor)