Data Selection

Quick Recap: The ML System Lifecycle

Lifecycle is a loop, not a linear process.

- Data Preprocessing.
- Model Training.
- Model Evaluation.
- Model Deployment.
- Monitoring and Maintenance.

Today's question

- Q: If you had unlimited GPUs but limited high-quality data, what would you optimize first?

Today's question

- Q: If you had unlimited GPUs but limited high-quality data, what would you optimize first?
- ~~A: Get more data.~~
- A: Get more value from existing data.

The Data Wall

The dominant strategy was straightforward: *more data, better models*.

A critical asymmetry has emerged:

Resource	**Growth**	**Rate**
GPU Compute	Exponential	10×/3 yr
Training Data	Sub-linear	2×/5 yr
Labeled Data	Linear	1.5×/5 yr
Synthetic Data	Unbounded	(quality-limited)

The field is now compute-rich and data-poor.

The Data Wall:Compute can process 10T tokens; only 5T high-quality tokens exist.

Consequence:Priority inverts from “get more data” to “get more from existing data.”

Today's Questions

- How can we select the most informative examples for training a model?
- How can we leverage unlabeled data to improve model performance?
- How can we adapt the training data to the model's current state of learning?

Is More Data Always Better?

Key Question:
Can we achieve similar model performance while training on fewer, more informative examples?

Data Selection

Data selection is the process of selecting a subset of the available data for training a model. The goal is to select the most informative examples that will lead to better model performance.

Data Selection Vs Data Engineering

	Data Engineering	**Data Selection**
Focus	Cleanliness, consistency	Informativeness, diversity
Question	Is the data correct?	Is correct data worth the compute?
Metric	Error rate, completeness	Learning per FLOP

Data Selection and the Iron Law

The basic tradeoff is

$$\text{Performance} \approx F(\text{Compute}, \text{Model Size}, \text{Data})$$

and, for a fixed model family and compute budget,

$$\text{Learning} \propto D_{\text{effective}}.$$

If we define the full dataset as D, and a selected subset as $S \subset D$, then

$$|S| < |D|$$

whenever we remove redundant or noisy examples.
Data selection aims to keep the most informative examples so that

$$D_{\text{effective}} = \sum_{x_i \in S} w_i \approx D_{\text{full}} \quad \text{even when} \quad |S| \ll |D|,$$

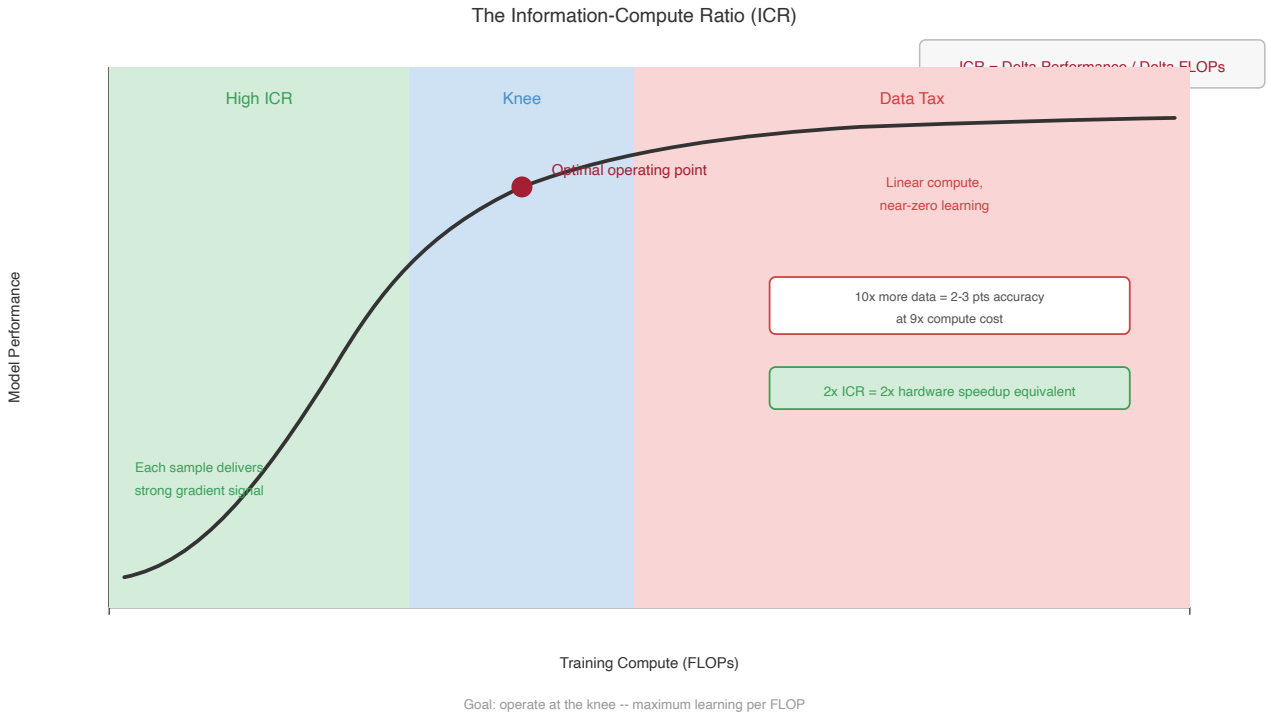
where w_i measures how much each example contributes to learning.
Key idea: we can get similar performance with fewer examples by increasing the average value of each selected example.

How to remove waste?

You have 1 million training samples.
You can only keep 10%.

Which samples do you throw away?

The Information-Compute Ratio (ICR)



The Data Tax:10× more data yields only 2–3 points accuracy at 9× the compute cost.

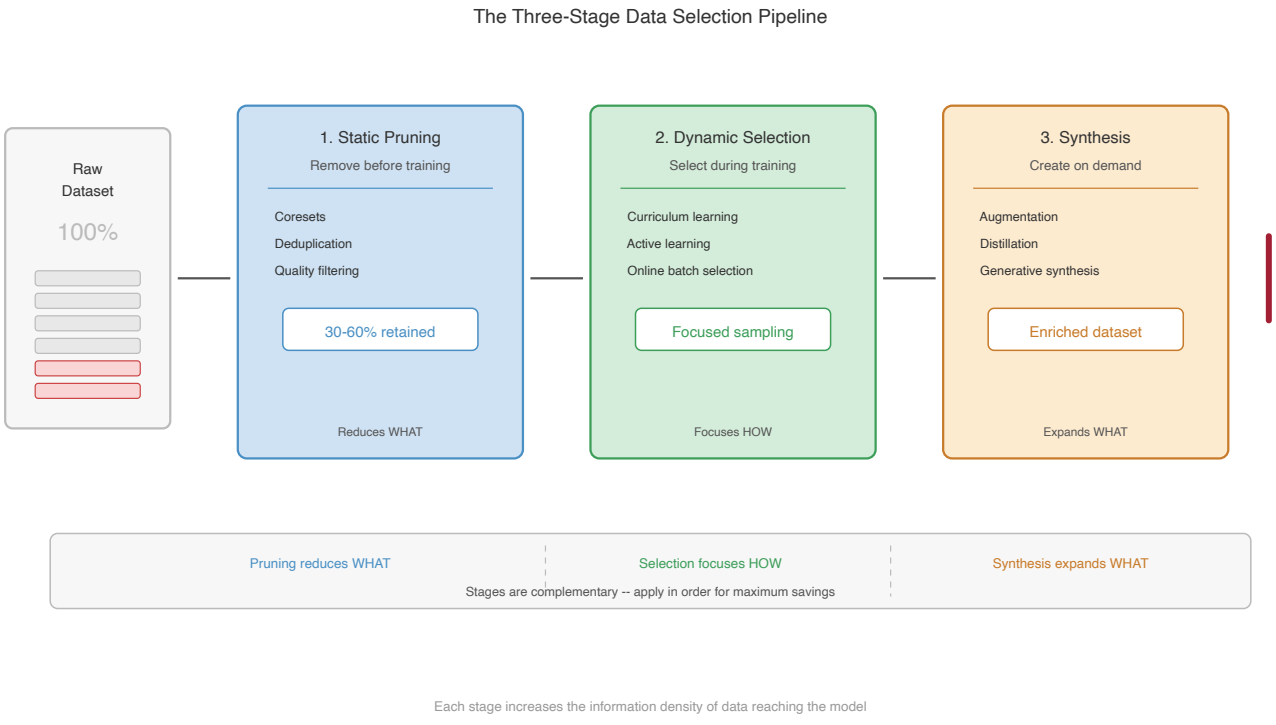
How to determine which examples are most informative?

Option 1

Training Examples	Validation Accuracy
10,000	75%
50,000	85%
100,000	89%
500,000	92%
1,000,000	93%

Observation:
The first 500,000 examples provide the most significant improvement in validation accuracy.
The last 500,000 examples provide only a small improvement in validation accuracy.
Additional learning benefit must match Additional computational cost
Define ICR = Improvement in validation accuracy / Computational cost

The Three-Stage Pipeline



Pipeline:Pruning reduces *what*, selection focuses *how*, synthesis expands *what*.

How to determine which examples are most informative? Option 2

- Static Data Pruning: Remove examples that are deemed uninformative before training begins.
  Which existing examples should we keep?
- Dynamic Data Selection: Prioritize examples as training progresses.
  Which examples should the model learn from next?
- Synthetic Data Generation: Create useful examples when existing data is insufficient.

Note: Static selection changes the training dataset before training.

Quiz Time

Your team has two possible training strategies.

Metric	Strategy A	Strategy B
Dataset size	1 million images	500,000 images
Training time	10 hours	5 hours
Validation accuracy	92%	92%
Training compute	100%	50%

Which strategy would you choose? Why?

Quiz Time

Your team has two possible training strategies.

Metric	Strategy A	Strategy B
Dataset size	1 million images	500,000 images
Training time	10 hours	5 hours
Validation accuracy	92%	92%
Training compute	100%	50%

Which strategy would you choose? Why?
Do you need additional information to make a decision? If so, what information would you like to have?

Static Data Pruning: How Do We Choose Representative Examples?

- Problem Statement: Select 20,000 frames from 100,000

Static Data Pruning: How Do We Choose Representative Examples?

- Problem Statement: Select 20,000 frames from 100,000
- Random selection is simple, but may not be optimal. Why?

Static Data Pruning: How Do We Choose Representative Examples?

- Problem Statement: Select 20,000 frames from 100,000
- Random selection is simple, but may not be optimal. Why?
- Preserve diversity and coverage
- Instead of selecting arbitrary examples, choose examples that represent different regions of the dataset. How?

Static Data Pruning: How Do We Choose Representative Examples?

- Problem Statement: Select 20,000 frames from 100,000
- Random selection is simple, but may not be optimal. Why?
- Preserve diversity and coverage
- Instead of selecting arbitrary examples, choose examples that represent different regions of the dataset. How?
  - Use clustering to group similar examples together, then select representative examples from each cluster.
  - Use a distance metric to measure the similarity between examples, then select examples that are far apart from each other.

k-center Greedy Selection

- Convert each image into a feature vector using a pretrained neural network.

$$z_i = f(x_i)$$

- Select an initial representative example.
- Find the example farthest from all previously selected examples.

$$x^* = \arg\max_{x_i \in D} \min_{x_j \in S} d(z_i, z_j)$$

- Add that example to the selected subset S.
- Repeat until all examples have been selected.

Example: k-center Greedy Selection

Image	Embedding	Scene
A	0.0	Empty driveway
B	0.1	Empty driveway
C	0.2	Empty driveway
D	4.8	Person walking
E	5.0	Person walking
F	5.2	Person walking
G	9.8	Nighttime shadow

Run k-Center Greedy

1. Select B (initial example)

Run k-Center Greedy

1. Select B (initial example)
2. Select G (farthest from B)

Run k-Center Greedy

1. Select B (initial example)
2. Select G (farthest from B)
3. Select E (farthest from B and G)

Result of k-Center Greedy

Final Selected Subset
- B: Empty driveway
- E: Person walking
- G: Nighttime shadow

Result: Three examples covering three distinct regions of the embedding space.

Key Insight: Select examples that cover different regions of the data distribution instead of repeatedly selecting similar examples.
Issues with this approach?

Result of k-Center Greedy

Final Selected Subset
- B: Empty driveway
- E: Person walking
- G: Nighttime shadow

Result: Three examples covering three distinct regions of the embedding space.

Key Insight: Select examples that cover different regions of the data distribution instead of repeatedly selecting similar examples.
Issues with this approach?
Geometric coverage does not guarantee balanced classes or preservation of rare safety-critical cases.
In practice, we should enforce minimum representation of important groups before or during selection.

Static Selection — What If We Want the Most Informative Examples?

k-Center selects examples based on geometric coverage.
But another possibility is to select examples based on model learning difficulty.

Algorithm: EL2N (Error L2-Norm)

- Train a model for a few epochs on the full dataset.
- Use the proxy model to predict each training example.
- For each example, compute the L2 norm of the error between the model's prediction and the true label.

$$EL2N(x_i) = ||f(x_i) - y_i||_2$$

- Rank examples by their EL2N score.
- Select the top-k examples with the highest EL2N scores.

EL2N Example

Image	EL2N Score	Difficulty
A	0.1	Easy
B	0.2	Easy
C	0.3	Easy
D	0.8	Medium
E	0.9	Medium
F	1.0	Hard
G	1.5	Hard

The proxy model already understands the first image. The second and third images may contain important information about what the model has not yet learned. But a high EL2N score can also indicate an incorrect label.

The Case for Smaller Datasets

A carefully selected **10%** of data can match the accuracy of 100%.

Why? Most samples are redundant:

- Sit deep within class clusters
- Contribute near-zero gradient signal

Valuable samples live near the decision boundary where uncertainty is highest.

50% reduction in dataset size saves 50% of forward/backward passes and gradient updates.

Metric	**Value**
Full dataset	1M samples
Training cost	100 GPU-hrs
Coreset (10%)	100K samples
Coreset cost	10 GPU-hrs
Accuracy retained	$\sim$95%
Savings	**90 GPU-hrs**

Data Deduplication

The only "free lunch" in data selection:

- Guaranteed zero accuracy penalty
- Immediate compute savings
- Web-scraped datasets: 30–50% near-duplicates

Two-pass approach:

1. **Exact hash**: MD5/SHA-256 (fast, cheap)
2. **Fuzzy hash**: MinHash/SimHash (near-dupes)

Always deduplicate first — precede all other selection methods.

Why duplicates hurt:Model memorizes instead of generalizing. Gradient wasted on redundant updates.

Dynamic Data Selection

Dynamic Data Selection

- Instead of selecting a static subset of examples before training, we can dynamically select examples during training.
- But the usefulness of an example changes as the model learns.
- $D_t = S(D, \theta_t)$ where D is the full dataset, θ_t are the model parameters at time t, and S is a selection function that chooses the most informative examples based on the current model state.
- Key Idea: The training data should adapt to what the model has already learned and what it still needs to learn.

Curriculum Learning

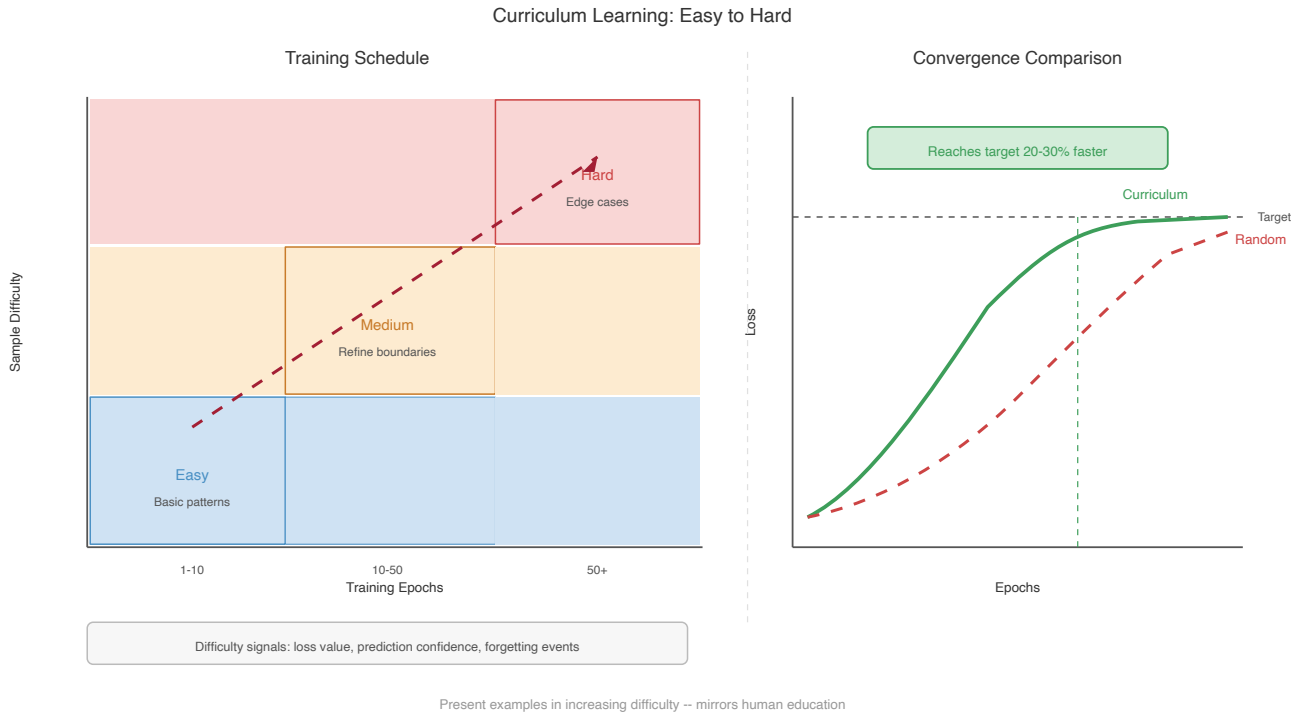
Motivation: How do students learn?

Curriculum Learning

Motivation: How do students learn?

- Start with easy examples and gradually increase the difficulty of the examples as the model learns.
- The difficulty of an example can be measured by its loss, its EL2N score, or its distance from the decision boundary.
- The curriculum can be designed manually or learned automatically.

Curriculum Learning: Easy to Hard



Like human education:Start with easy examples, gradually increase difficulty. 20–30% faster convergence.

Algorithm 1: Loss-Based Curriculum Learning

- Step 1: Assign a Difficulty Score
  For a labeled example (x_i, y_i), use cross-entropy loss:
  $$\boxed{d_i = -\log p_\theta(y_i \mid x_i)}$$
  Where:
  - $p_\theta(y_i \mid x_i)$: Probability assigned to the correct label.
  - d_i: Difficulty score.

  A low loss indicates that the model predicts the correct label confidently.

Algorithm 1: Loss-Based Curriculum Learning

- Step 2: Sort Examples by Difficulty
  Sort the training examples in ascending order of their difficulty scores d_i.
  This creates a curriculum from easy to hard examples.
  Suppose the true label for each image is Person Present.

Image	Probability Assigned to Correct L
A: Clear pedestrian	0.95
B: Pedestrian in shadows	0.70
C: Partially occluded pedestrian	0.40
D: Pedestrian at night	0.10

$A < B < C < D$

The model finds image A easiest and image D hardest.

- Begin with the easiest examples and progressively expand the training set.

Curriculum Learning Variants

- Curriculum Pacing: Once the examples have been ranked, the curriculum can be presented in different ways, such as gradually increasing the difficulty or using a more complex pacing strategy.
- Self-Paced Learning: The model selects examples based on its current state, allowing it to focus on examples that are most informative for its learning progress.

Dynamic Data Selection: Active Learning

- Motivation: Which examples should we label?
- Active learning is a dynamic data selection strategy where the model identifies unlabeled examples that are potentially informative and asks a human annotator to provide their labels.

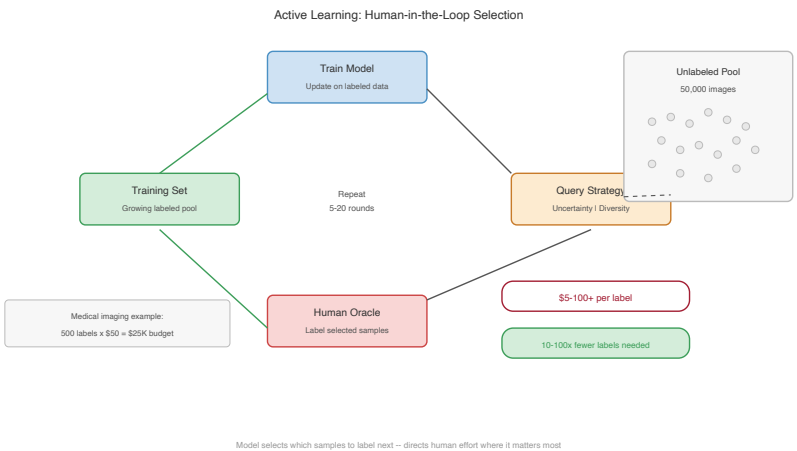
Example: Suppose our smart doorbell company has:
- 10,000 labeled frames.
- 90,000 unlabeled frames.
- A limited annotation budget.

Human labeling is expensive.

Should we randomly choose the next 1,000 frames for annotation?

Or should the model help decide which labels would be most useful?
Active Learning Loop: Train; Evaluate; Select; Annotate; Repeat.

Active Learning: Human-in-the-Loop



The model directs labeling effort:

1. Train on current labeled data
2. Query: select uncertain/diverse samples
3. Human oracle labels selected batch
4. Add to training set, repeat

Labels cost \$5–100+ per sample.

Medical: 500 labels $\times$ \$50 = \$25K.

Savings:10–100$\times$ fewer labels needed.

Active Learning

- Let $D_L = \{(x_i, y_i)\}$ be the labeled dataset.
- Let $D_U = \{x_j\}$ be the unlabeled dataset.
- Train a model on D_L.
- At iteration t, the model selects a batch:
  $B_t = argmax_{B \in D_U, |B|=b} U(B, \theta_t)$,
  where $U(B; \theta_t)$ is a measure of how useful it would be to obtain labels for batch B.

Identifying U: Uncertainty Sampling

- Uncertainty sampling is a common active learning strategy where the model selects examples for which it is least confident in its predictions.
- For a classification model, uncertainty can be measured using the predicted probability distribution over classes.
- Example: Suppose the model predicts the following probabilities for an unlabeled image: - Class A: 0.4 - Class B: 0.35 - Class C: 0.25 The model is uncertain about this example because the predicted probabilities are relatively close together.
- $U(x) = 1 - max(p, 1 - p)$

Alternative Active Learning Query Strategies

- Margin Sampling: Select examples where the difference between the top two predicted class probabilities is smallest.
- Entropy-Based Sampling: Select examples with the highest entropy in their predicted class probabilities.
- Query by Committee: Maintain a committee of models and select examples where the committee members disagree the most.

Can we learn from unlabeled datasets?

- Data Selection: Which examples should we use for training, and when?
- Semi-supervised learning: How can we learn from both labeled and unlabeled examples?

Semi-Supervised Learning

- Semi-supervised learning is a machine learning paradigm that leverages both labeled and unlabeled data for training.
- The goal is to improve model performance by utilizing the abundant unlabeled data, which can provide additional information about the underlying data distribution.
- Common semi-supervised learning techniques include: - Pseudo-Labeling: The model generates pseudo-labels for unlabeled data and uses them for further training. - Consistency regularization: The model is encouraged to produce consistent predictions for unlabeled examples under different perturbations or augmentations. - Label Propagation: Construct a graph connecting labeled and unlabeled examples, and propagate label information through the graph.

Pseudo-Labeling

If a model is highly confident about an unlabeled example, use that prediction as a training target.

- Train a model on the labeled dataset D_L.
- Predict pseudo-labels for the unlabeled dataset D_U using the trained model.
- Select examples from D_U with high-confidence predictions and add them to the labeled dataset D_L.
- Retrain the model on the updated labeled dataset D_L.
- Repeat the process until convergence or a stopping criterion is met.

Q: Are we doing data selection in pseudo-labeling?

Consistency Regularization

A classifier should ideally produce the same prediction for two label-preserving transformations of an image.
Example: Suppose we have an unlabeled image of a cat. We can apply different augmentations to the image, such as rotation, flipping, or color jittering. The model should produce consistent predictions for all augmented versions of the image, even though they look different.

- Train a model on the labeled dataset D_L.
- For each unlabeled example in D_U, generate multiple augmented versions of the example.
  $x_1 = T_1(x)$; $x_2 = T_2(x)$;
- Encourage the model to produce consistent predictions for all augmented versions of the same example.
  So minimize the consistency loss:
  $L_{consistency} = \sum_{i=1}^{n} ||f_\theta(T_i(x)) - f_\theta(T_j(x))||^2$ for all pairs of augmentations (T_i, T_j).
- Combine the consistency loss with the labeled data loss to update the model parameters.

FixMatch: A Semi-Supervised Learning Algorithm

- FixMatch is a semi-supervised learning algorithm that combines pseudo-labeling and consistency regularization.
- The algorithm generates pseudo-labels for weakly augmented unlabeled examples and enforces consistency between the predictions of weakly and strongly augmented versions of the same example.
- FixMatch has been shown to achieve state-of-the-art performance on several benchmark datasets, such as CIFAR-10, CIFAR-100, and SVHN.

Algorithm FixMatch

- Step 1: Generate a Weakly Augmented Image
  Given an unlabeled image x, apply a relatively mild transformation:
  $x_w = T_w(x)$ Examples include a small translation or horizontal flip where appropriate.
- Step 2: Predict a Pseudo-Label
  Compute: $p_w = p_\theta(y \mid x_w)$
  Select $\hat{y} = \arg\max_c p_w(c)$
- Step 3: Apply Confidence Filtering If $\max_c p_w(c) < \tau$, discard the example.
  Otherwise, keep the example and its pseudo-label $\hat{y}$.
- Step 4: Generate a Strongly Augmented Image
  Apply a more aggressive transformation to the original image:
  $x_s = T_s(x)$ Examples include RandAugment or Cutout.
- Step 5: Enforce Consistency Regularization
  Compute the model's prediction for the strongly augmented image:
  $p_s = p_\theta(y \mid x_s)$

Worked example

Say: An unlabeled image contains a pedestrian.

- The weakly augmented image produces:
  $P(\text{Person} \mid x_w) = 0.97$
- Because the prediction is sufficiently confident, assign:
  $\hat{y} = \text{Person}$
- Now apply a stronger brightness and occlusion transformation.
- The model predicts: $P(\text{Person} \mid x_s) = 0.60$
- The loss becomes: $\mathcal{L}_u = -\log(0.60) \approx 0.511$
  The training update encourages the model to become more confident about the pedestrian despite the stronger transformation.
- Complete Training Objective:
  $\mathcal{L} = \mathcal{L}_s + \lambda \mathcal{L}_u$ where $\mathcal{L}_s$ is the supervised loss on labeled examples and L_u is the unsupervised loss on unlabeled examples and λ is a hyperparameter that balances the two loss terms.

Three approaches to Dynamic Data Selection

- Curriculum Learning: Start with easy examples and gradually increase the difficulty of the examples as the model learns.
- Active Learning: Select examples that are most informative for the model based on its current state, and query an oracle (e.g., a human annotator) for labels for those examples.
- Semi-supervised Learning: Training on both labeled and unlabeled datasets

Summary

Algorithm	Selection Signal	Selection Decision	Extra Computation
Curriculum learning	Difficulty score	Which examples become available	Initial difficulty scorin
Active learning	Uncertainty, disagreement, diversity	Which samples humans annotate	Pool inference and r training
FixMatch	Confidence and prediction consistency	Which unlabeled examples contribute to consistency loss	Weak/strong augm and extra model comp

Which algorithm will you use?

- Scenario1: Model learns simple cases well but converges slowly on difficult ones?

Which algorithm will you use?

- Scenario1: Model learns simple cases well but converges slowly on difficult ones?

Which algorithm will you use?

- Scenario1: Model learns simple cases well but converges slowly on difficult ones?
- Scenario2: Model is trained on a small labeled dataset and a large unlabeled dataset?

Which algorithm will you use?

- Scenario1: Model learns simple cases well but converges slowly on difficult ones?
- Scenario2: Model is trained on a small labeled dataset and a large unlabeled dataset?
- Scenario3: Model is trained on a large labeled dataset but has limited annotation budget?

Which algorithm will you use?

- Scenario1: Model learns simple cases well but converges slowly on difficult ones?
- Scenario2: Model is trained on a small labeled dataset and a large unlabeled dataset?
- Scenario3: Model is trained on a large labeled dataset but has limited annotation budget?