DADS7305: MLOPs

Northeastern University

Instructor: Ramin Mohammadi

February 10, 2026

These materials have been prepared and sourced for the course **MLOPs** at Northeastern University. Every effort has been made to provide proper citations and credit for all referenced works.

If you believe any material has been inadequately cited or requires correction, please contact me at:

`r.mohammadi@northeastern.edu`

Thank you for your understanding and collaboration.

Data Iteration

Data-centric AI development

Data-centric AI development

Model-centric view

Collect what data you can, and develop a model good enough to deal with the noise in the data.

Hold the data fixed and iteratively improve the code/ model.

Data-centric view

The consistency of the data is paramount. Use tools to improve the data quality; this will allow multiple models to do well.

Hold the code fixed and iteratively improve the data.

Data Iteration

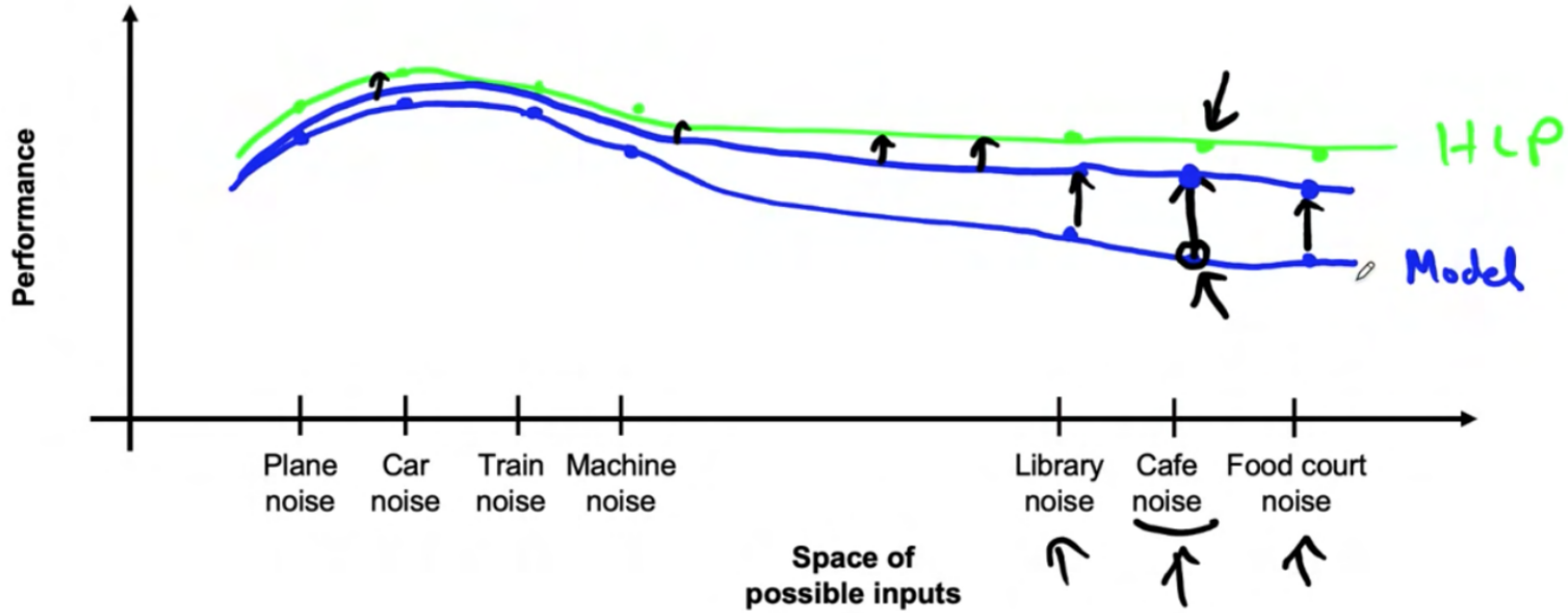
A useful picture of data augmentation

Speech recognition example

Different types of speech input:

- Car noise
- Plane noise
- Train noise
- Machine noise
- Cafe noise
- Library noise
- Food court noise

Speech recognition example



Data Iteration

Data augmentation

Data augmentation

Goal:

Create realistic examples that (i) the algorithm does poorly on, but (ii) humans (or other baseline) do well on

Checklist:

- Does it sound realistic?
- Is the X $\rightarrow$ Y mapping clear? (e.g., can humans recognize speech?)
- Is the algorithm currently doing poorly on it?

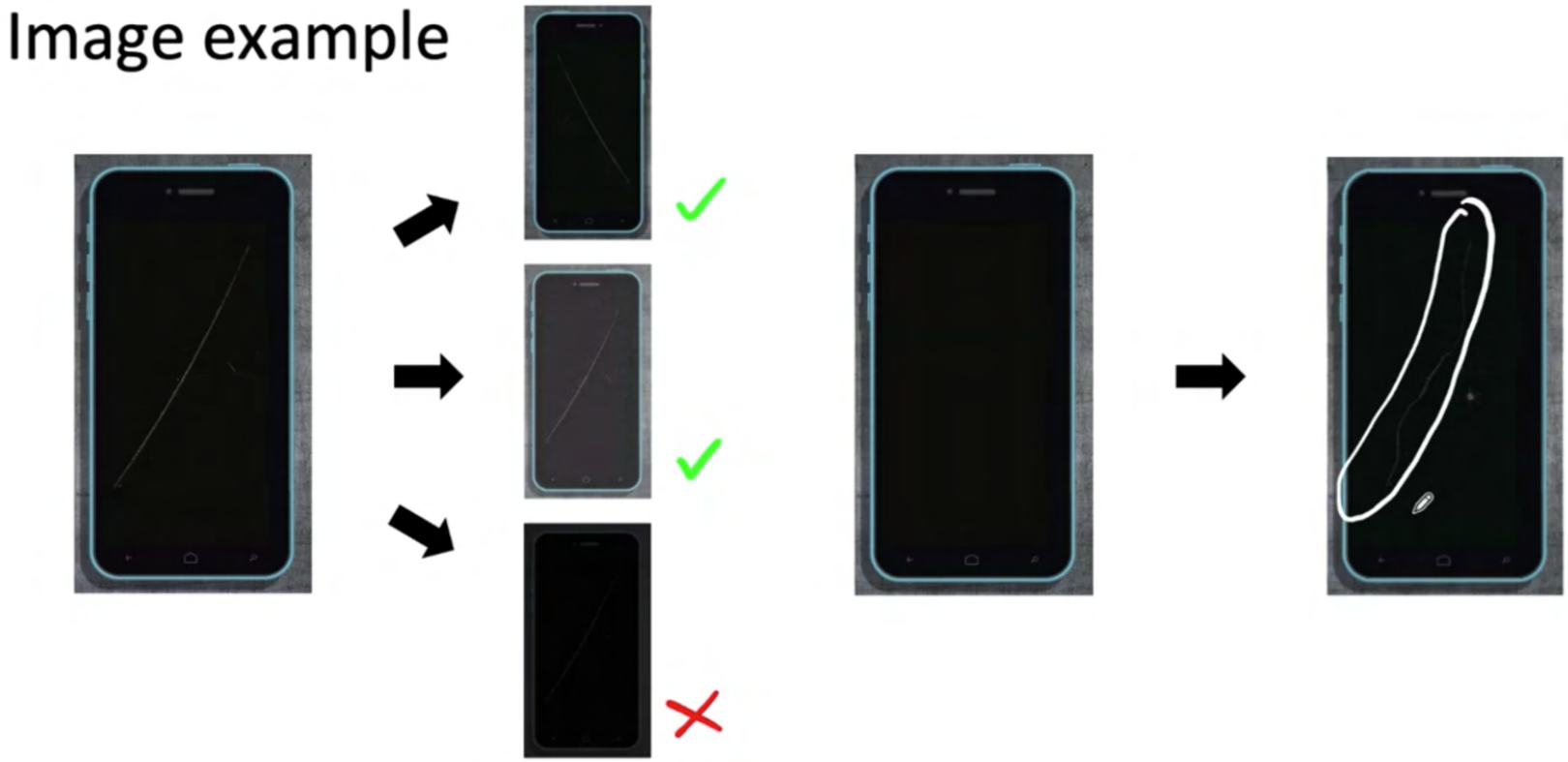
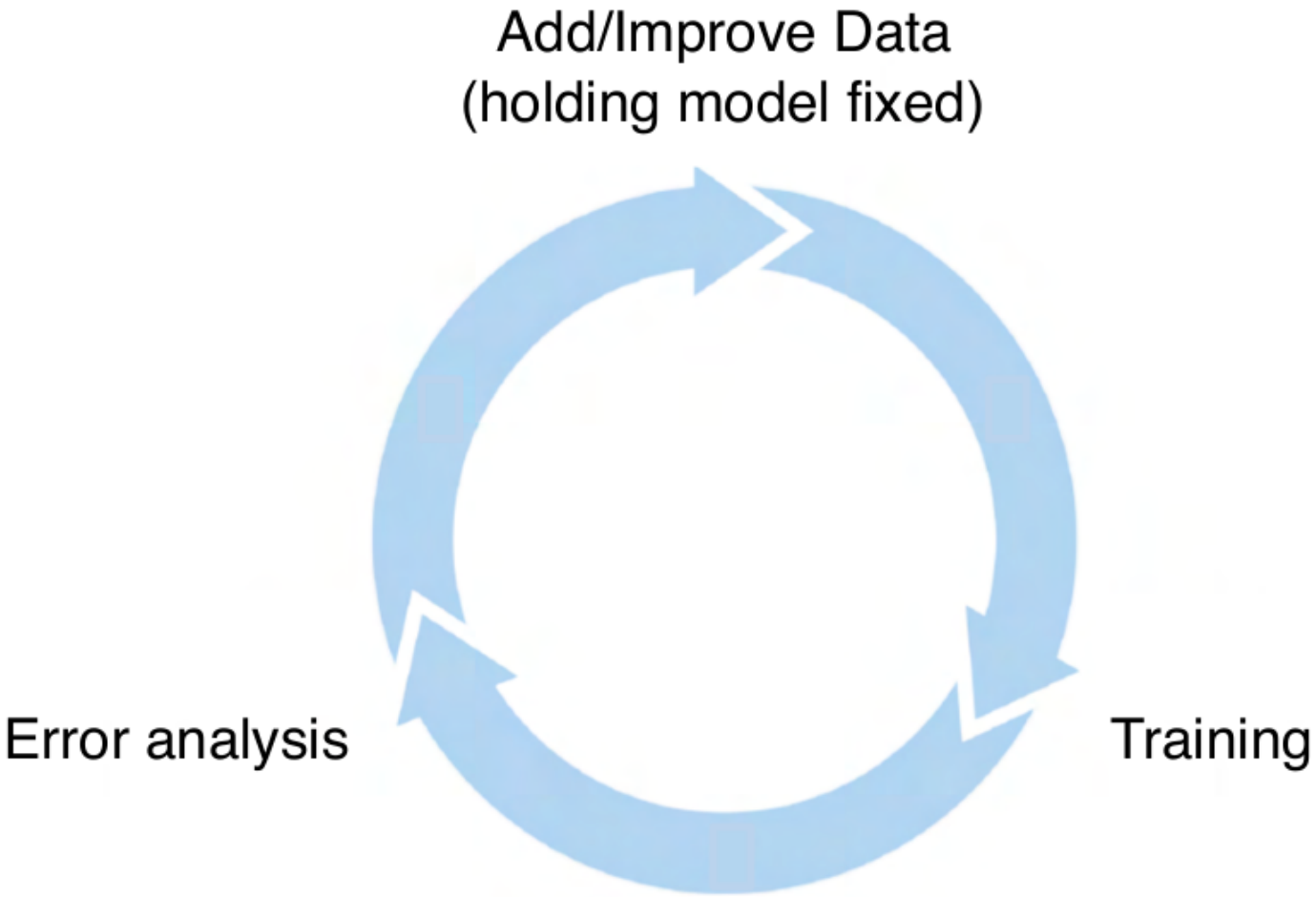


Image example

Data iteration loop



Data Iteration

Can adding data hurt?

Can adding data hurt performance?

For unstructured data problems, if:

- The model is large (low bias).
- The mapping X $\rightarrow$ Y is clear (e.g., humans can make accurate predictions).

Then, **adding data rarely hurts accuracy.**

Photo OCR counterexample

1

high accuracy

I

421

low accuracy

1? I?

Adding a lot of new "I"s may skew the dataset and hurt performance

Data Iteration

Adding features

Structured data

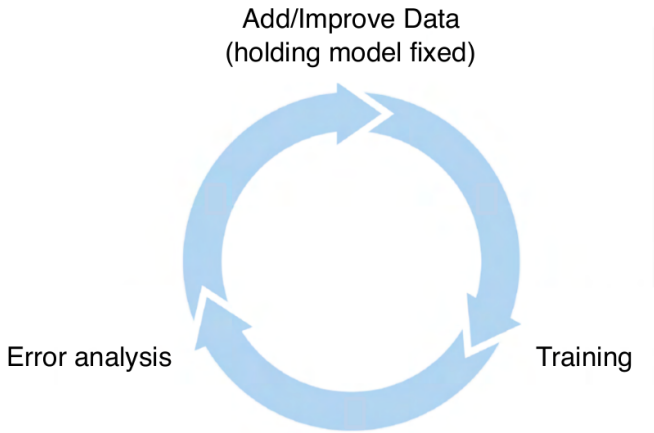
Restaurant recommendation example

Vegetarians are frequently recommended restaurants with only meat options.

Possible features to add?

- Is person vegetarian (based on past orders)?
- Does restaurant have vegetarian options (based on menu)?

Data iteration



Cycle: Error Analysis → Model Architecture (add features) → Training

- Error analysis can be harder if there is no good baseline (such as HLP) to compare to.
- Error analysis, user feedback, and benchmarking to competitors can all provide inspiration for features to add.

Data Iteration

Experiment tracking

Experiment tracking

What to track?

- Algorithm/code versioning
- Dataset used
- Hyperparameters
- Results

Tracking tools

- Text files
- Spreadsheet
- Experiment tracking system

Desirable features

- Data needed to replicate results
- In-depth analysis of experiment results
- Perhaps also: Resource monitoring, visualization, model error analysis

Data Iteration

From big data to good data

From Big Data to Good Data

Try to ensure consistently high-quality data in all phases of the ML project lifecycle.

Good data is:

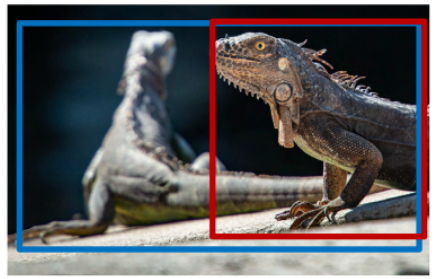
- Cover of important cases (good coverage of inputs x)
- Defined consistently (definition of labels y is unambiguous)
- Has timely feedback from production data (distribution covers data drift and concept drift)
- Sized appropriately

Define data and establish baseline

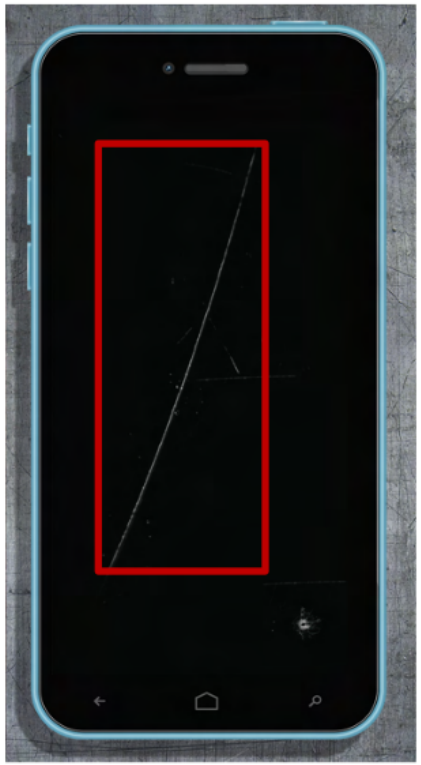
Why is data definition hard?

Iguana detection example

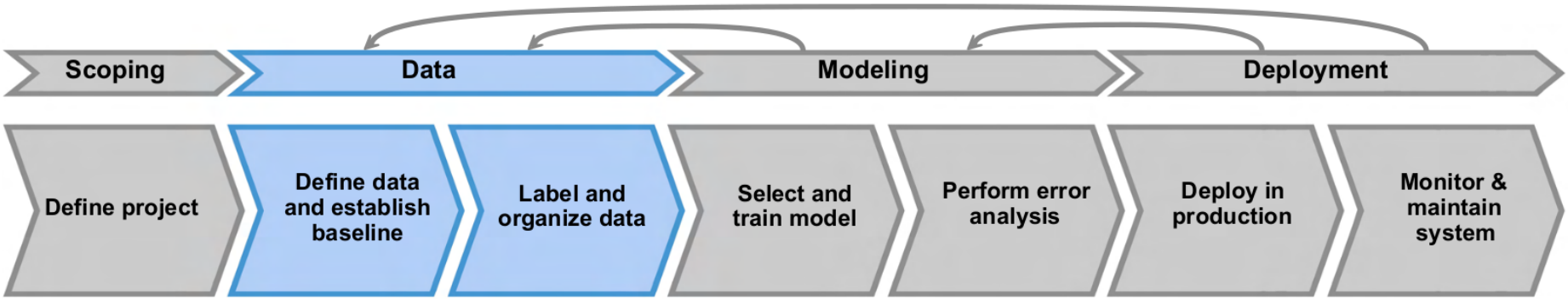
Labeling instructions: "Use bounding boxes to indicate the position of iguanas"



Phone defect detection



Data stage


Scoping
Data
Modeling
Deployment
Define project
Define data and establish baseline
Label and organize data
Select and train model
Perform error analysis
Deploy in production
Monitor & maintain system

Define data and establish baseline

More label ambiguity examples

Speech recognition example

"Um, nearest gas station"

"Umm, nearest gas station"

"Nearest gas station [unintelligible]"

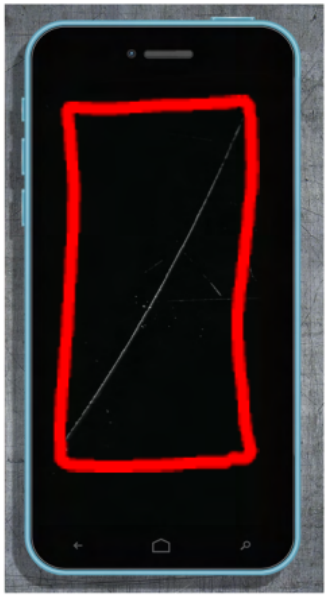
User ID merge example

	Job Board (website)	**Resume chat (app)**
Email	nova@deeplearning.ai	nova@chatapp.com
First Name	Nova	Nova
Last Name	Ng	Ng
Address	1234 Jane Way	?
State	CA	?
Zip	94304	94304

Binary features: 1 if same, 0 if different

Data definition questions

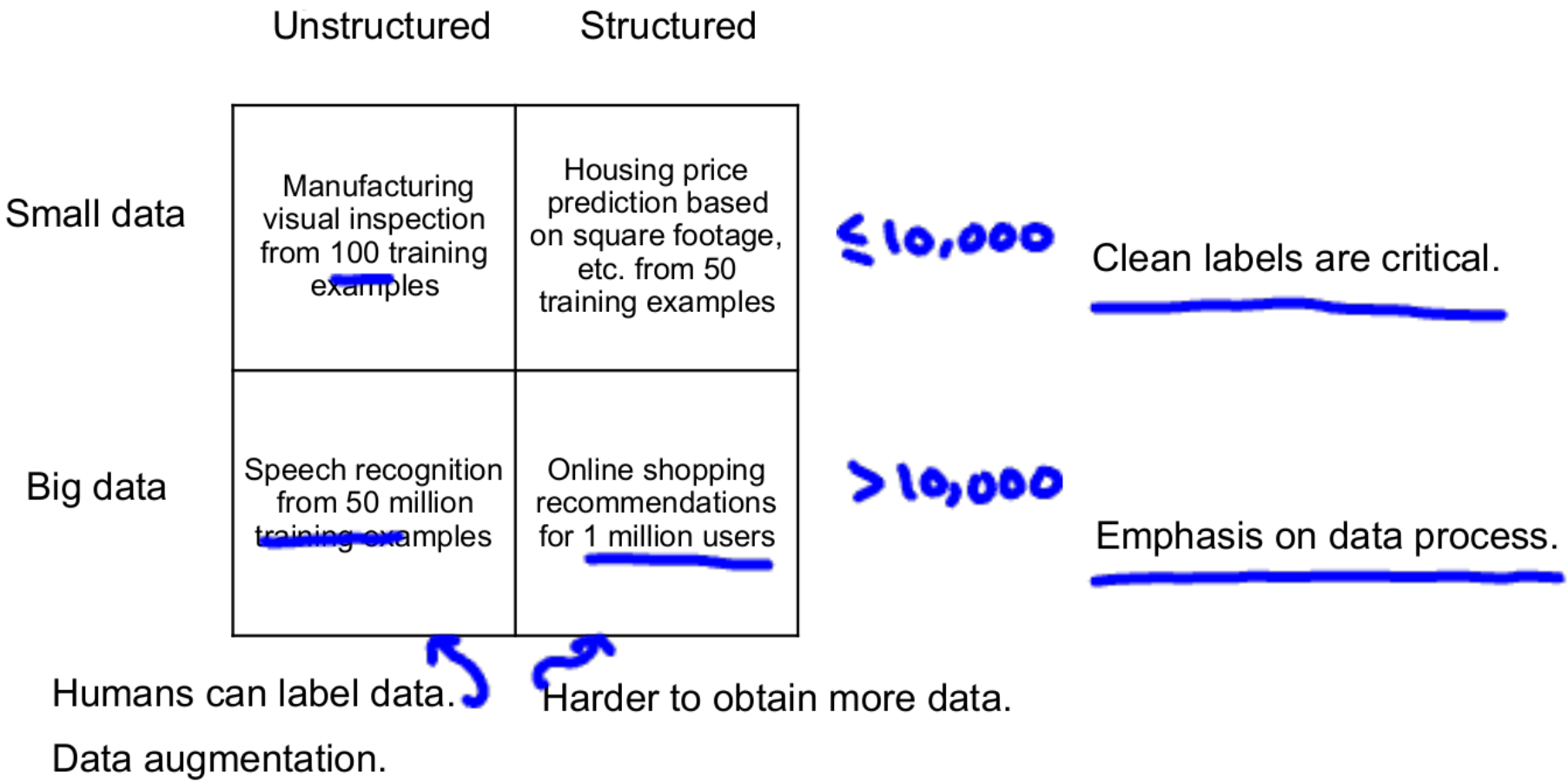
- What is the input x?
  - Lighting? Contrast? Resolution?
- What features need to be included?
- What is the target label y?
  - How can we ensure labelers give consistent labels?



Define data and establish baseline

Major types of data problems

Major types of data problems



Unstructured vs. structured data

Unstructured data

- May or may not have huge collection of unlabeled examples x.
- Humans can label more data.
- Data augmentation more likely to be helpful.

Structured data

- May be more difficult to obtain more data.
- Human labeling may not be possible (with some exceptions).

Small data vs. big data

Small data

- Clean labels are critical.
- Can manually look through dataset and fix labels.
- Can get all the labelers to talk to each other.

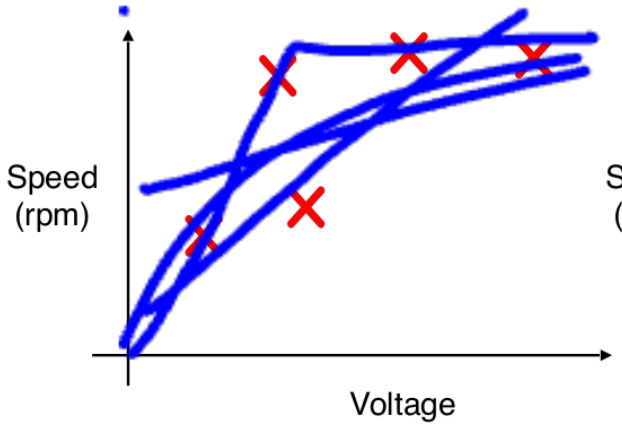
Big data

- Emphasis data process.

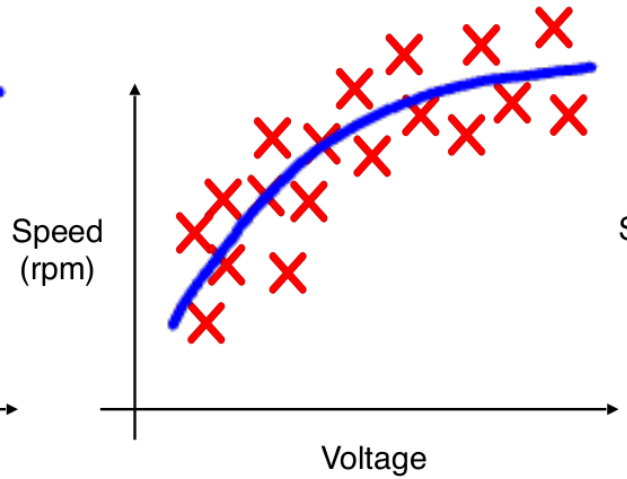
Define data and establish baseline

Small data and label consistency

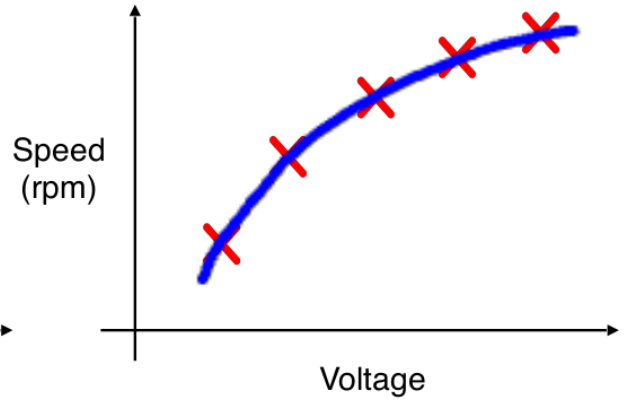
Why label consistency is important



- Small data
- Noisy labels



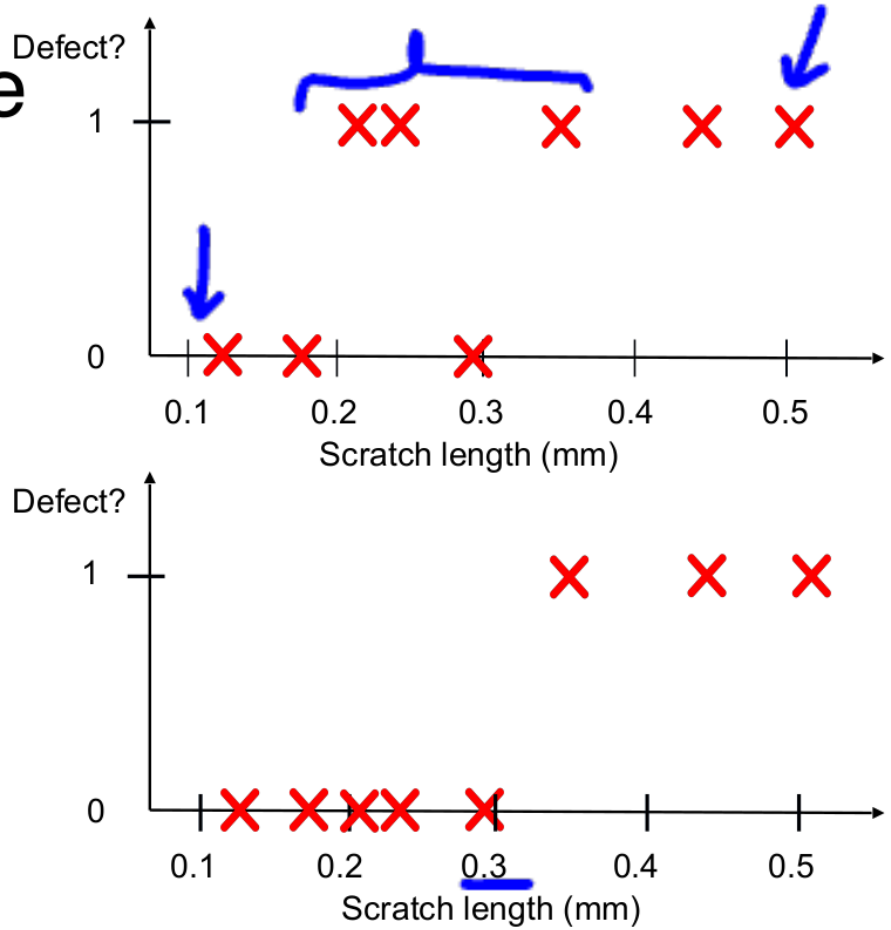
- Big data
- Noisy labels



- Small data
- Clean (consistent) labels

Why label consistency is important

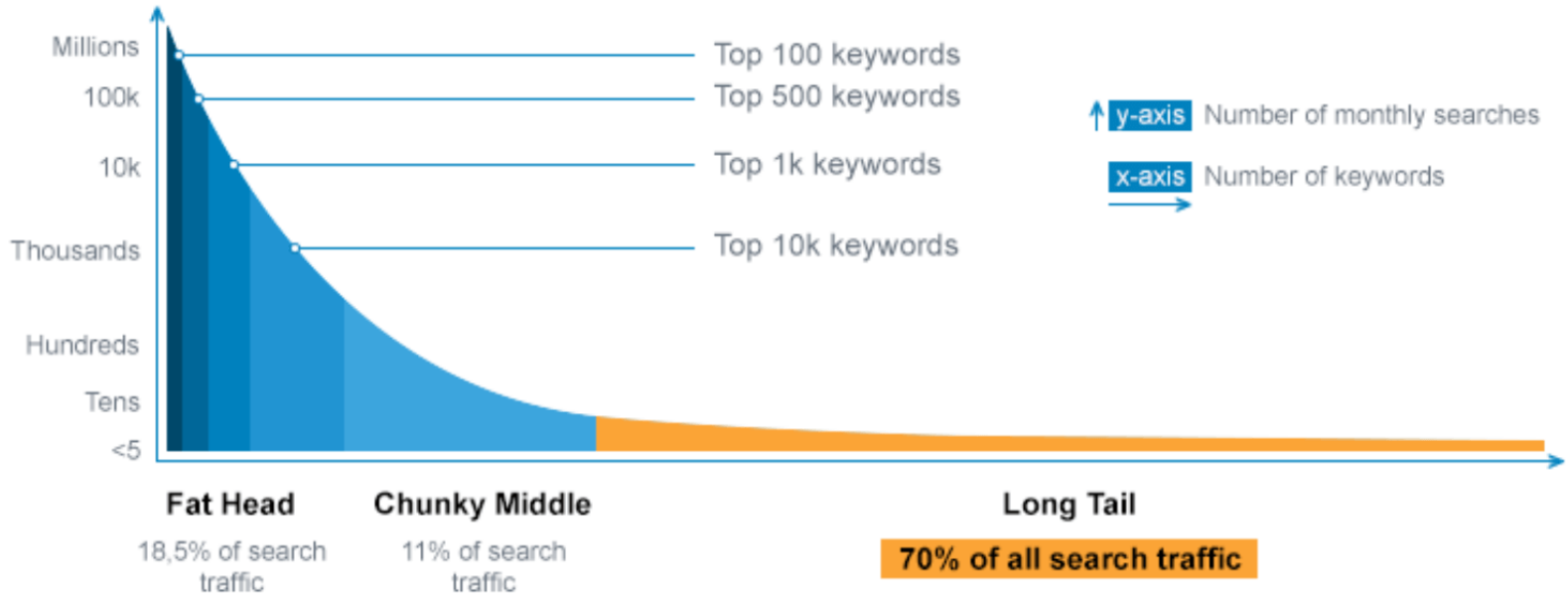
Phone defect example



Big data problems can have small data challenges too

Problems with a large dataset but where there's a long tail of rare events in the input will have small data challenges too.

- Web search
- Self-driving cars
- Product recommendation systems



Source: Bill Tancer via Hittail

Define data and establish baseline

Improving label consistency

Improving label consistency

- Have multiple labelers label same example.
- When there is disagreement, have MLE, subject matter expert (SME) and/or labelers discuss definition of y to reach agreement.
- If labelers believe that x doesn't contain enough information, consider changing x.
- Iterate until it is hard to significantly increase agreement.

Examples

- Standardize labels

"Um, nearest gas station"
"Umm, nearest gas station"
"Nearest gas station [unintelligible]"

➡ "Um, nearest gas station"

- Merge classes


Deep scratch


Shallow scratch

➡ Scratch

Have a class/label to capture uncertainty

- Defect: 0 or 1

Alternative: 0, Borderline, 1

- Unintelligible audio

"nearest go"

"nearest grocery"

"nearest [unintelligible]"

Small data vs. big data (unstructured data)

Small data

- Usually small number of labelers.
- Can ask labelers to discuss specific labels.

Big data

- Get to consistent definition with a small group.
- Then send labeling instructions to labelers.
- Can consider having multiple labelers label every example and using voting or consensus labels to increase accuracy.

Why measure HLP?

- Estimate Bayes error / irreducible error to help with error analysis and prioritization.

99%

Ground Truth Label	Inspector
1	1 ✓
1	0 ✗
1	1 ✓
0	0 ✓
0	0 ✓
0	1 ✗

↑ Human?

66.7% accuracy

Raising HLP

- When the label y comes from a human label, HLP $< 100\%$ may indicate ambiguous labeling instructions.
- Improving label consistency will raise HLP.
- This makes it harder for ML to beat HLP. But the more consistent labels will raise ML performance, which is ultimately likely to benefit the actual application performance.

HLP on structured data

Structured data problems are less likely to involve human labelers, thus HLP is less frequently used.

Some exceptions:

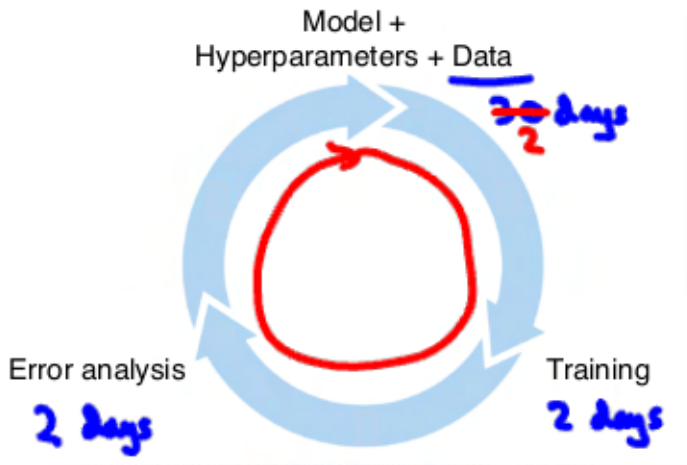
- User ID merging: Same person?
- Based on network traffic, is the computer hacked?
- Is the transaction fraudulent?
- Spam account? Bot?
- From GPS, what is the mode of transportation – on foot, bike, car, bus?

Label and organize data

Obtaining data

How long should you spend obtaining data?

- Get into this iteration loop as quickly as possible.
- Instead of asking: How long would it take to obtain m examples?
  Ask: How much data can we obtain in k days.
- Exception: If you have worked on the problem before and from experience you know you need m examples.



Inventory data

Brainstorm list of data sources (speech recognition)

Source	Amount	Cost
Owned	100h	$0
Crowdsourced – Reading	1000h	$10000
Pay for labels	100h	$6000
Purchase data	1000h	$10000

Other factors: Data quality, privacy, regulatory constraints

Labeling data

- Options: In-house vs. outsourced vs. crowdsourced
- Having MLEs label data is expensive. But doing this for just a few days is usually fine.
- Who is qualified to label?
  - Speech recognition – any reasonably fluent speaker
  - Factory inspection, medical image diagnosis – SME (subject matter expert)
  - Recommender systems – maybe impossible to label well
- Don't increase data by more than 10x at a time

Label and organize data

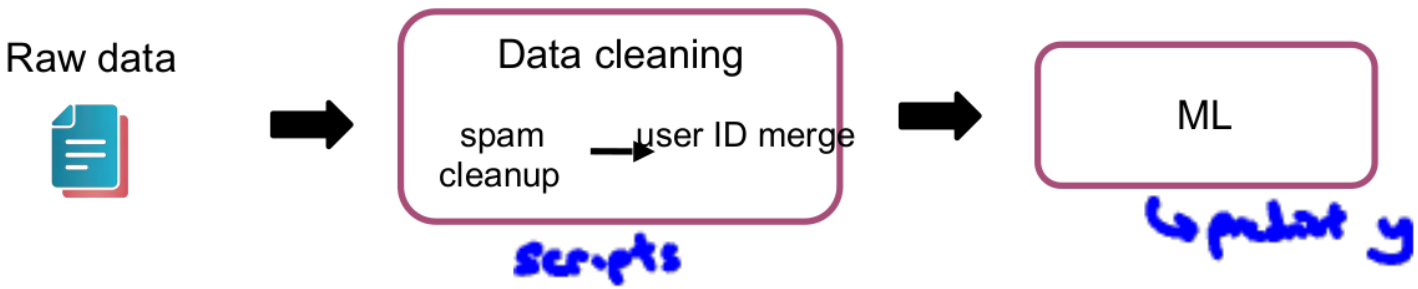
Data pipeline

Data pipeline example

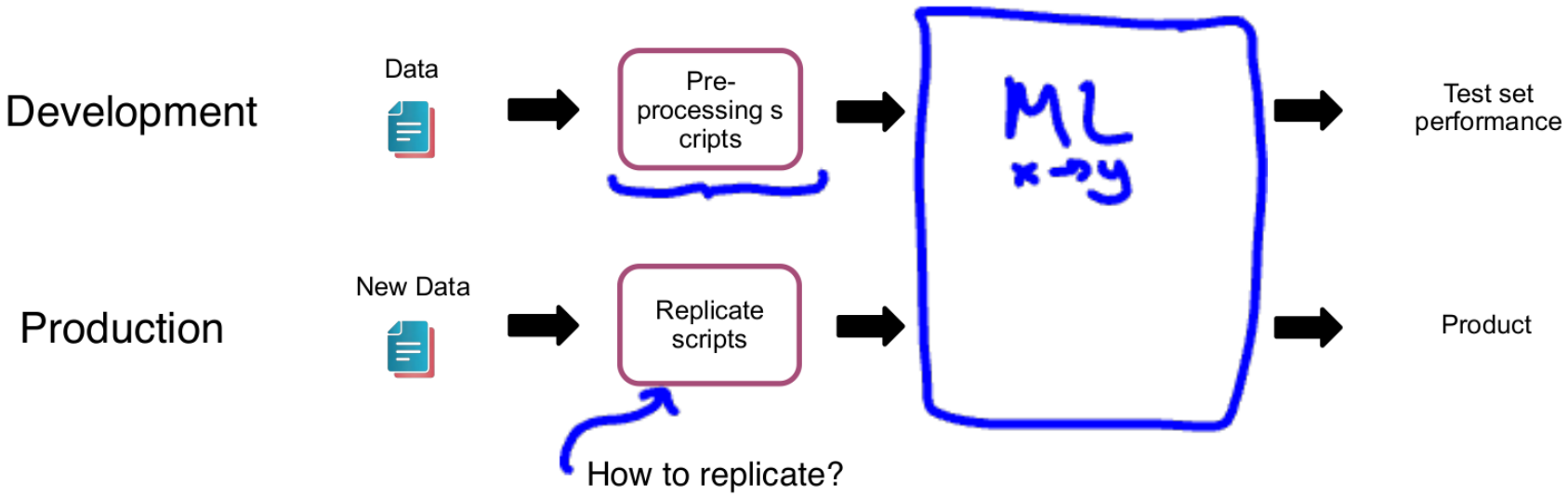
	Job Board (website)	Resume chat (app)
Email	nova@deeplearning.ai	nova@chatapp.com
First Name	Nova	Nova
Last Name	Ng	Ng
Address	1234 Jane Way	?
State	CA	?
Zip	94304	94304

x = user info

y = looking for job



Data pipeline example



POC and Production phases

POC (proof-of-concept):

- Goal is to decide if the application is workable and worth deploying.
- Focus on getting the prototype to work!
- It's ok if data pre-processing is manual. But take extensive notes/comments.

Production phase

- After project utility is established, use more sophisticated tools to make sure the data pipeline is replicable.
- E.g., TensorFlow Transform, Apache Beam, Airflow,...

LLM Data Pipeline Construction

- Critical but often overlooked part of LLMOps.
- Moves raw text data through transformations for model training or inference.
- LLMs are extremely data-hungry (GB–TB scale).
- A well-designed pipeline:
  - Handles ingestion, cleaning, tokenization, batching, sharding, streaming.
  - Is memory-efficient and scalable.
  - Keeps GPUs fully utilized.
- Enables reproducibility and data versioning.

Role of Data Pipelines in LLMOps

- Traditional ML: Simple CSV read, feature normalization, small batches.
- LLMOps: Must handle massive, unstructured text efficiently.
- Challenges:
  - Tokenization for model vocabulary.
  - Efficient batching to minimize padding.
  - Sharding for distributed training.
  - Streaming for large or real-time datasets.

Key Concept 1: Tokenization

- Converts text to integer token IDs (BPE, WordPiece, SentencePiece).
- Use the exact tokenizer tied to the pre-trained model.
- Pre-tokenize large datasets for speed.
- Key decisions:
  - Sequence length and splitting.
  - Handling special tokens (e.g., $< EOS >$, padding).
- Balances vocabulary size and sequence length.

Key Concept 2: Batching

- Groups sequences for parallel GPU processing.
- Naive batching wastes memory due to padding.
- Use bucketing by length to reduce padding.
- Dynamic padding with tools like Hugging Face DataCollator.
- Tune batch size for maximum GPU usage without OOM errors.

Key Concept 3: Sharding

- Splits dataset across multiple workers or machines.
- Ensures each sample is seen once per epoch across all GPUs.
- Improves scalability and training speed.
- Can also split huge datasets into smaller files for easier handling.
- Example: `dataset.shard(num_shards=N, index=i)` in Hugging Face.

Key Concept 4: Streaming

- Processes data on-the-fly without loading all into memory.
- Essential for web-scale or continuous data.
- Hugging Face: `load_dataset(..., streaming=True)` returns iterable dataset.
- Can combine with sharding for distributed streaming.
- Also useful for real-time inference pipelines.

Putting it All Together

- Robust LLM data pipelines combine:
  - Tokenization → Sharding → Streaming → Smart batching.
- Goals:
  - Keep GPUs fully fed with data.
  - Avoid preprocessing bottlenecks.
  - Maintain reproducibility and version control.
- In LLMOps, **data quality and pipeline design** often determine model success.

Practical Tools for LLM Data Pipelines

- **Hugging Face Datasets:** Loading, sharding, streaming.
- **Transformers:** Model-specific tokenizers.
- **PyTorch DataLoader:** Efficient batch loading and collation.
- Integrate in containerized or cloud-based workflows.
- Always log dataset versions and preprocessing configs.

Putting It All Together: LLM Data Pipelines

- **Pipeline Steps:**
  - Stream or load dataset from disk/remote source.
  - Tokenize (optionally distributed across CPUs for speed).
  - Batch & pad, then send to GPU for training.
  - Encapsulate in classes/scripts for easy swapping of datasets or tokenizers.
- **Industry-Scale Practices:**
  - Distributed storage: data lakes, Apache Spark, Ray.
  - Streaming libraries: TensorFlow `tf.data`, PyTorch `torchdata`.
  - Monitor & profile: avoid GPU idle time (e.g., TensorBoard).

- **Data Quality:**
  - Cleaning: remove problematic or undesirable content.
  - Deduplication to avoid overfitting/waste (e.g., MinHash, LSH).

- **Versioning & Traceability:**
  - Tools: DVC or dataset versioning in storage.
  - Enables tracing model behavior issues back to specific training data.

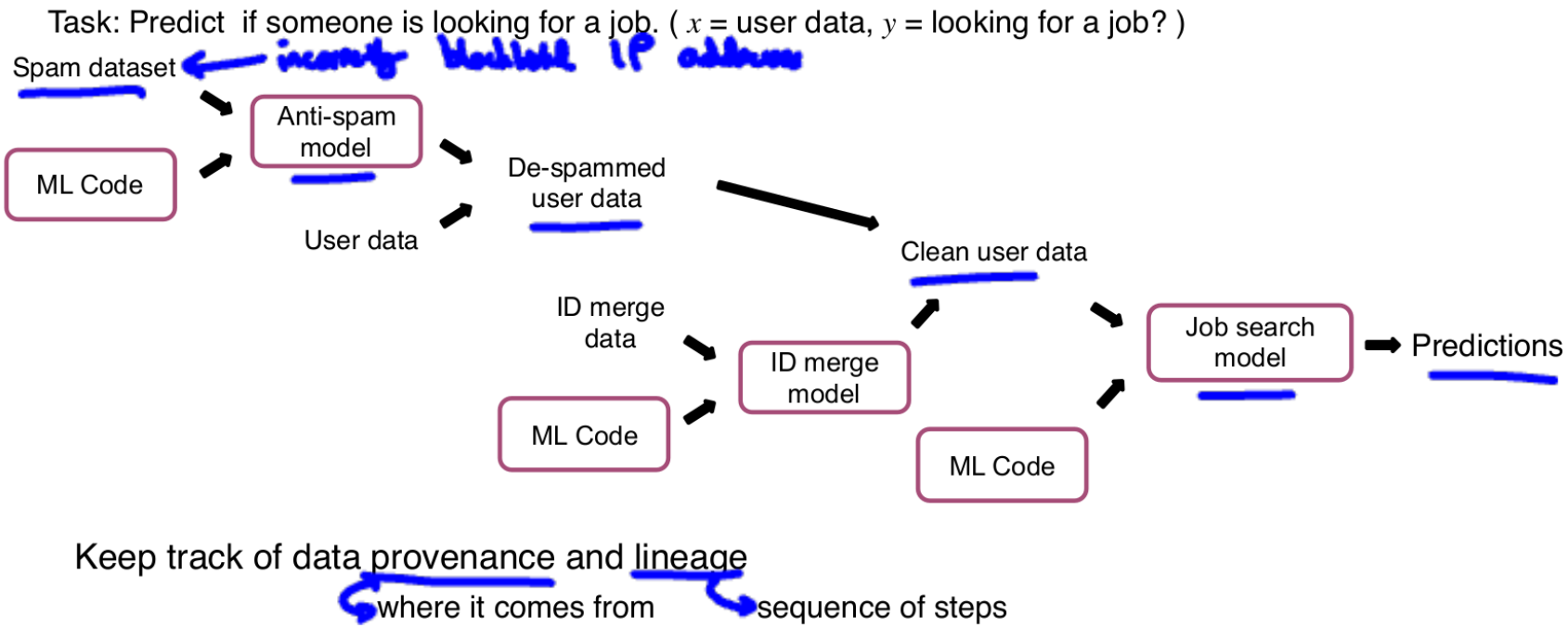
Modern Approaches and Tools

- **Hugging Face Streaming:** Host large datasets and stream directly into training (e.g., OpenLLaMA, RedPajama) without full download.
- **WebDataset Format:** Store data in many sharded tar files; workers read different shards in parallel. Libraries: `webdataset` (Python).
- **DataOps for LLMs:** Automated orchestration for continual fine-tuning , pulls fresh data, updates datasets, triggers pipelines.
- **Efficient Shuffling:** Use external memory shuffles or database-backed sampling for large datasets to maintain randomness.
- **Monitoring Pipelines:** Detect data drift , monitor length, language distribution, and topical changes in incoming data.
- **Iterative Optimization:** Identify bottlenecks (e.g., tokenization, IO) and optimize , parallelize, cache, or use compression for faster reads.

Label and organize data

Meta-data, data provenance and lineage

Data pipeline example



Meta-data

Examples:

- Manufacturing visual inspection: Time, factory, line #, camera settings, phone model, inspector ID, ...
- Speech recognition: Device type, labeler ID, VAD model ID, ...

Useful for:

- Error analysis. Spotting unexpected effects.
- Keeping track of data provenance.

Label and organize data

Balanced train/dev/test splits

Balanced train/dev/test splits in small data problems

Visual inspection example: 100 examples, 30 positive (defective)

Train/dev/test: 60%/20%/20%

Random split: 21/2/7 positive example

35% 10% 35%

Want: 18/6/6 } balanced split

30%/30%/30%

No need to worry about this with large datasets – a random split will be representative.

Labs for This Week

Objective

Briefly describe the learning goal for this week's lab(s).

Lab Activities:

- Lab 1: [API] , [FastAPI, Flask and Streamlit Tutorial]
- Lab 2: [Docker] , [Docker Tutorial]
- Lab 3: [Cloud Run] , [Cloud-Run Tutorial]

Submission Deadline: [Refer to Canvas]

- Assignment 4: [API] , [Create a Airflow pipeline of your choice eithe locally or via Cloud-Run]

Reading Materials

This Week's Theme

Topic focus: [People + AI Guidebook - Data Collection + Evaluation.pdf]
You should use the worksheet related to this pdf to your project and submit it when its requested.

Required Readings:

- [TECHNOLOGY READINESS LEVELS FOR MACHINE LEARNING SYSTEMS]

Be prepared to discuss highlights and open questions in class.

Sources

 DeepLearning.AI

 The People + AI Guidebook