Introduction to Machine Learning Engineering

Chicago ML
February 27, 2019
Garrett Smith

https://chicago.ml

Chicago AI Days

PROGRAM SPEAKERS REGISTRATION ORGANIZERS TRAVEL CONTACT

Super!

AI Ideas, Careers and Impact March 12th | Midwest Applied AI Conference May 20-21st

The Chicago Artificial Intelligence community invites you to:

Discover

Learn about AI applications

Learn

Dig into data & algorithms

Network

Meet other AI professionals

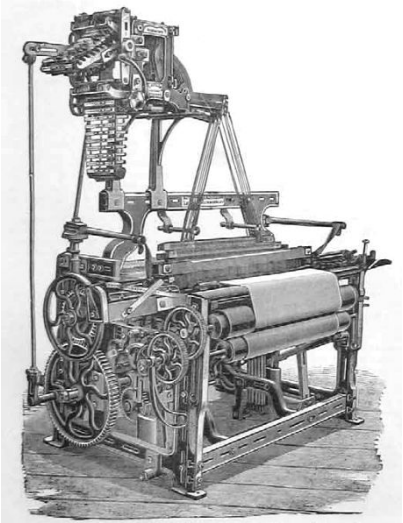
Start an AI Business

Attend AI startup boot camp

@guildai

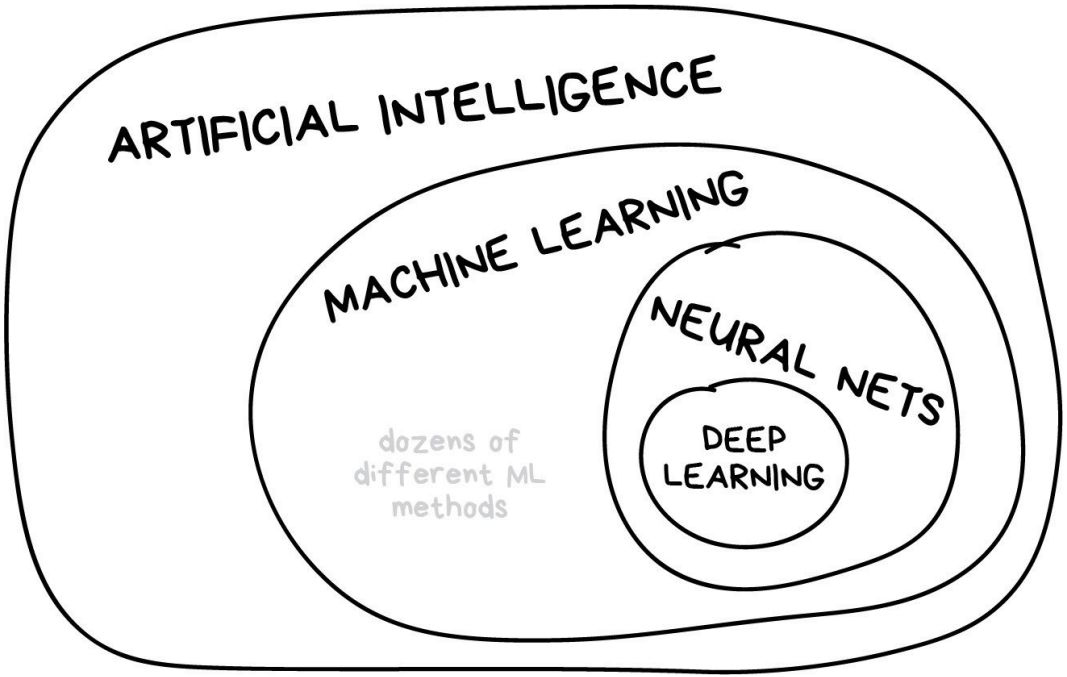
What is machine learning?

Theory



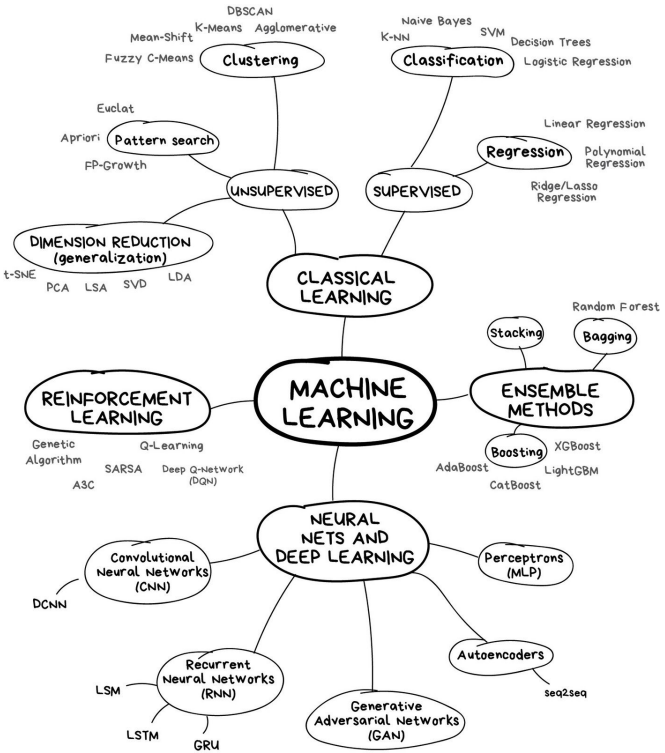
Tools

What is machine learning?



Credit: vas3k.com

What is machine learning?



Credit: vas3k.com

What is machine learning engineering?

Infrastructure

Facilities and tools for research and engineering

Continuous integration and continuous development

Research

Data analysis

Data processing and preparation

Model selection

Training a model

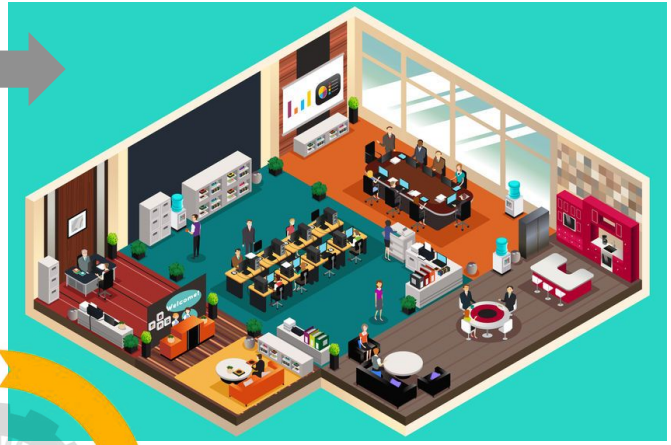
Production

Model inference

Model optimization

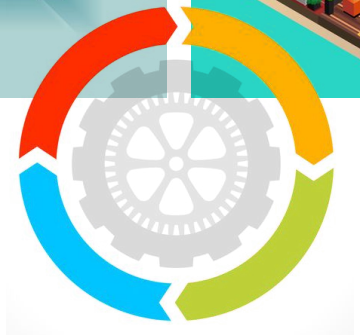
Deployment

Why machine learning engineering?



Data

Business Value



Reproducibility

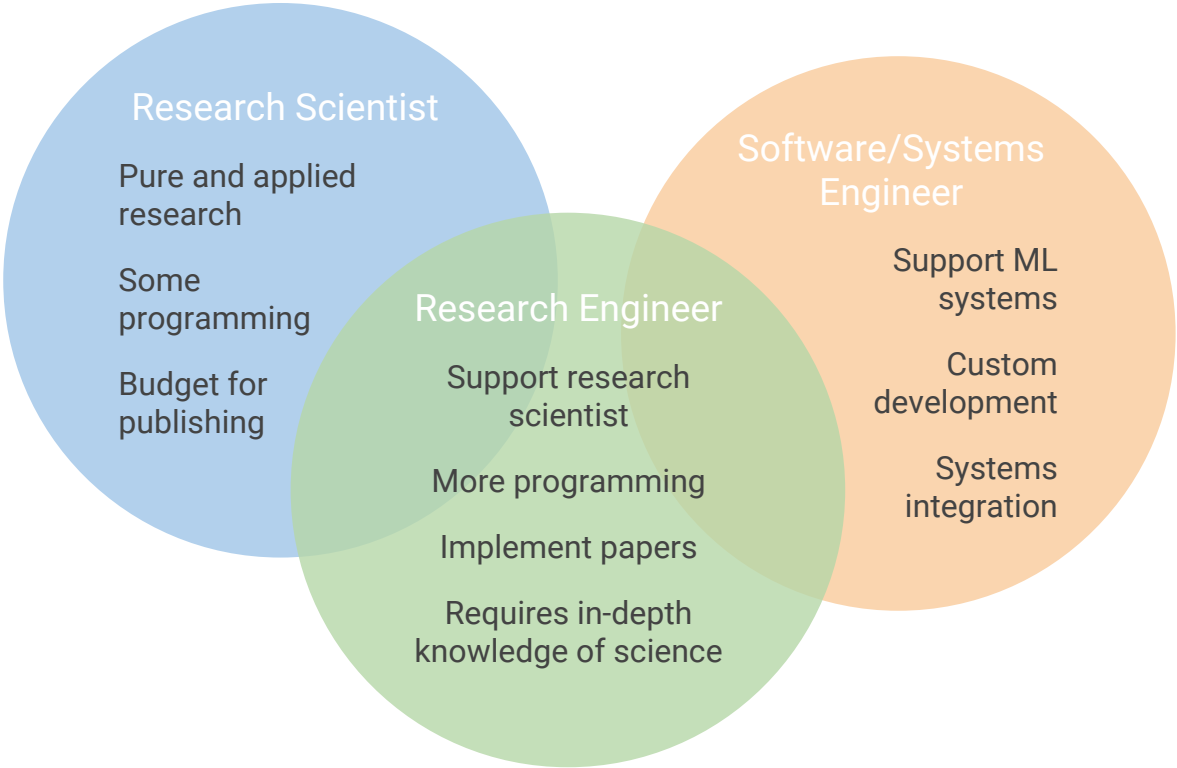
Use Cases

- Anomaly detection (e.g. fraud)
- Optimization (e.g. minimize cost, maximize yield)
- Market analysis
- Risk analysis
- Prediction

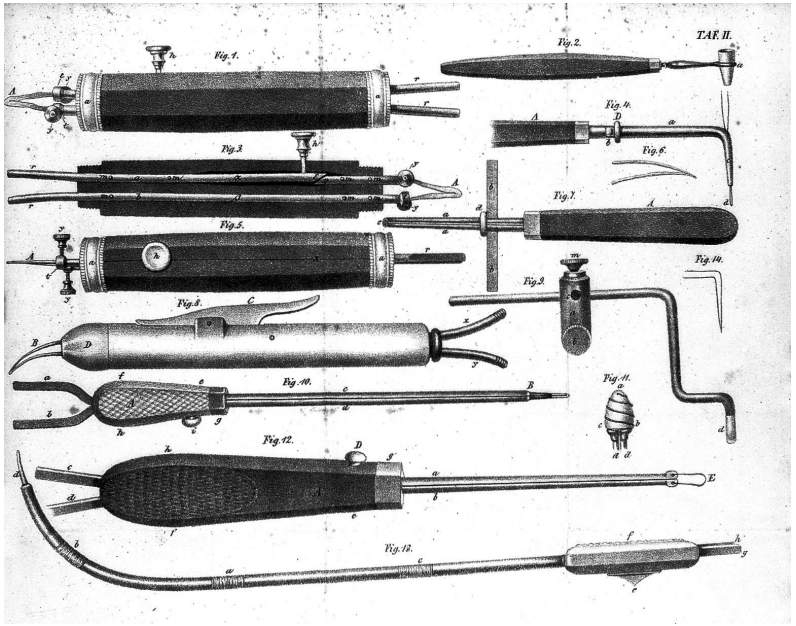
Machine learning vs traditional data analytics

	Traditional Data Analytics / BI	Machine Learning
Data suited for	Structured	Structured and unstructured
Typical application	Summary/reports, some prediction	Prediction, some summary/reports
Artifacts	Reports, graphs	Trained models, applications
Used by	Human decision makers	Application developers

What are the roles in an ML engineering team?



Tools of the trade



First instruments for galvanocautery introduced by Albrecht Middeldorpf in 1854 (source)

Tools of the trade

Programming languages

Language	When to Use
Python	General ML, data processing, systems integration
R	Stats, general data science
C/C++	System software, HPC
JavaScript	Web based applications
Java/Scala	Enterprise integration
bash	Systems integration

Tools of the trade

Computational libraries and frameworks

Library	Sweet Spot	When to Look Elsewhere
TensorFlow	Deep learning, production systems including mobile	New to ML, no production requirements
PyTorch	Ease of use, popular among researchers	Production requirements beyond simple serving
Keras	Ease of use, production backend with TensorFlow	Affinity with another library (e.g. colleagues use something else),
MXNet	Performance, scalability, stability	Seeking larger community or features not available in MXNet
Caffe 2	Computer vision heritage	Seeking larger community or need features not available in Caffe
scikit-learn	General purpose ML	Deep learning, need GPU

Tools of the trade

Modules and toolkits - Prepackaged models

Name	Application	Language and Libraries Used
spaCy	Natural language processing	Python, TensorFlow, PyTorch
TF-Slim	Image classification	TensorFlow
TF-object detection	Object detection	TensorFlow
TensorFlow Hub	Various	TensorFlow
Caffe Model Zoo	Various	Caffee
TensorFlow models	Various	TensorFlow
Keras applications	Various	Keras

Tools of the trade

Scripting tools

Tool	When to Use
Python + argparse	Create reusable scripts with well defined interfaces
Guild AI	Capture script output as ML experiments
Paver	Python make-like tool
Traditional build tools (make, cmake, ninja)	General purpose build automation

Tools of the trade

Workflow automation

Tool	When to Use
MLFlow	Enterprise wide machine learning workflow
Guild AI	Ad hoc workflows, integration with other automation systems
Polyaxon	Kubernetes based job scheduling
Airflow	General workflow automation
Traditional scripting	Ad hoc automation

Data analysis

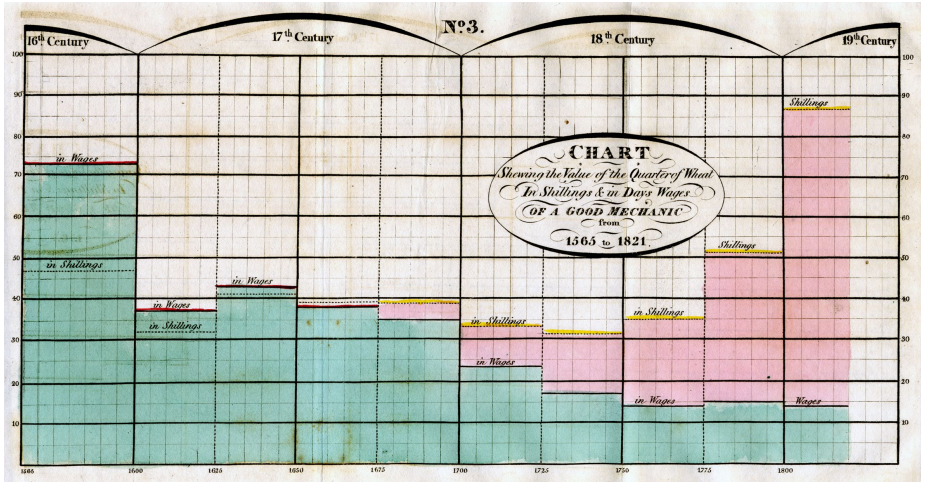


Chart showing quarterly value of wheat, 1821 (source)

Data analysis

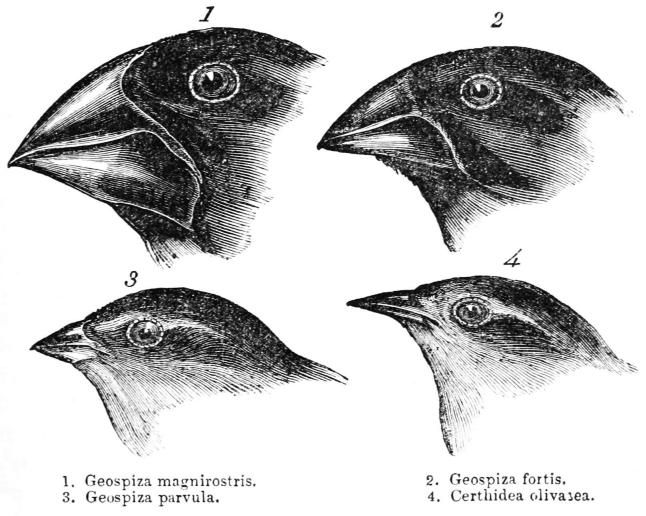
Structured vs unstructured data

FACTORY INDIRECT EXPENSES

Name of Account	A No.	B No.	C No.	D No.	E No.	F No.	G No.	H No.	J No.	Power Plant No.	Cost Office No.	Storeroom No.
Supervision	360	380	400	420	440	460	480	500	520	540	560	580
Clerks' Salaries	361	381	401	421	441	461	481	501	521	.....	561	581
Non-Productive Labor	362	382	402	422	442	462	482	502	522	542	.....	582
Repairs	363	383	403	423	443	463	483	503	523	543	563	583
Depreciation	364	384	404	424	444	464	484	504	524	544	564	584
Insurance—Fire	365	385	405	425	445	465	485	505	525	545	.....	585
Insurance—Liability	366	386	406	426	446	466	486	506	526	546	566	586
Rent	367	387	407	427	447	467	487	507	527	547	567	587
Taxes	368	388	408	428	448	468	488	508	528	548	.....	588
Supplies	369	389	409	429	449	469	489	509	529	549	569	589
Experimental Work	370	390	410	430	450	470	490	510	530	.....	.....	.....
Defective Work	371	391	411	431	451	471	491	511	531	.....	.....	.....
Light, Heat, and Power	372	392	412	432	452	472	492	512	532	.....	572	592
Sundry Expenses	373	393	413	433	453	473	493	513	533	553	573	593
General Operating	374	394	414	434	454	474	494	514	534	554	.....	.....

Structured Data

Classification chart of Factory Ledger Accounts, 1919 (source)



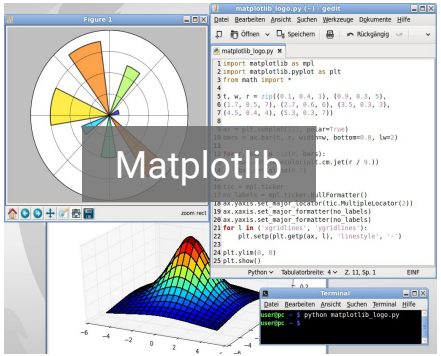
1. Geospiza magnirostris.
2. Geospiza fortis.
3. Geospiza parvula.
4. Certhidea olivacea.

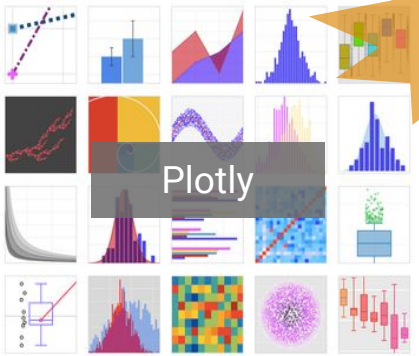
Unstructured Data

Darwin's Finches, 1837 (source)

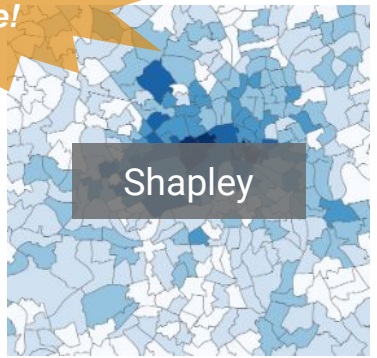
Data analysis

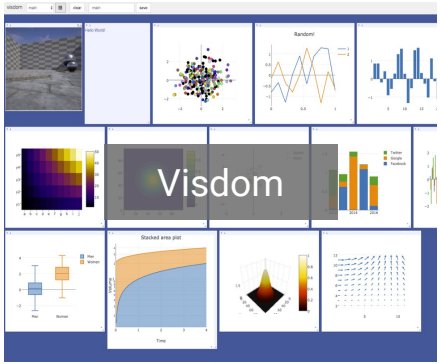
Visualization

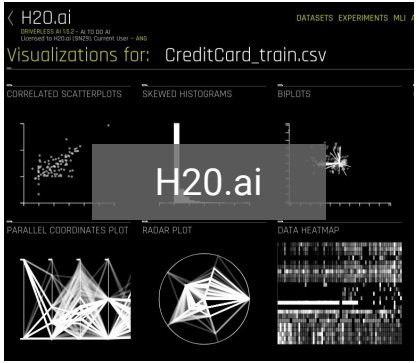


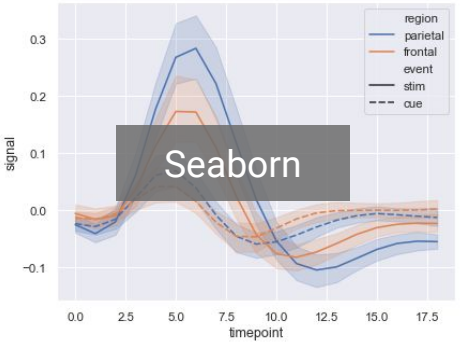


Many, many more!

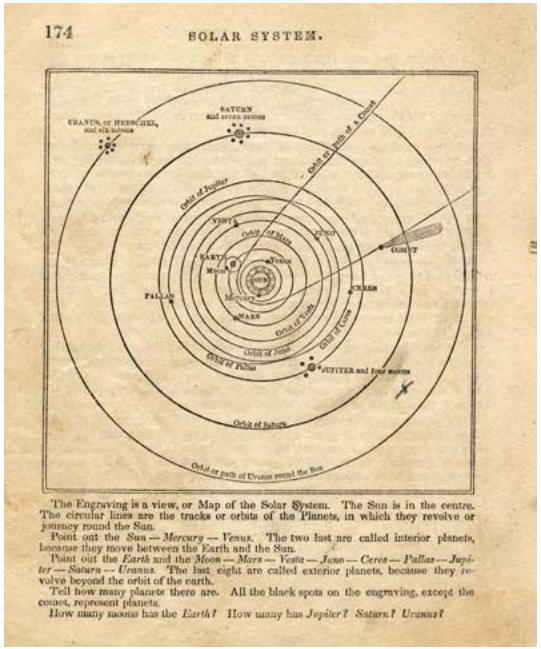








Model selection
(Representation)



174 SOLAR SYSTEM.

The Engraving is a view, or Map of the Solar System. The Sun is in the centre. The circular lines are the tracks or orbits of the Planets, in which they revolve or journey round the Sun.

Point out the *Sun* — *Mercury* — *Venus*. The two last are called interior planets, because they move between the Earth and the Sun.

Point out the *Earth* and the *Moon* — *Mars* — *Vesta* — *Juno* — *Ceres* — *Pallas* — *Jupiter* — *Saturn* — *Uranus*. The last eight are called exterior planets, because they revolve beyond the orbit of the earth.

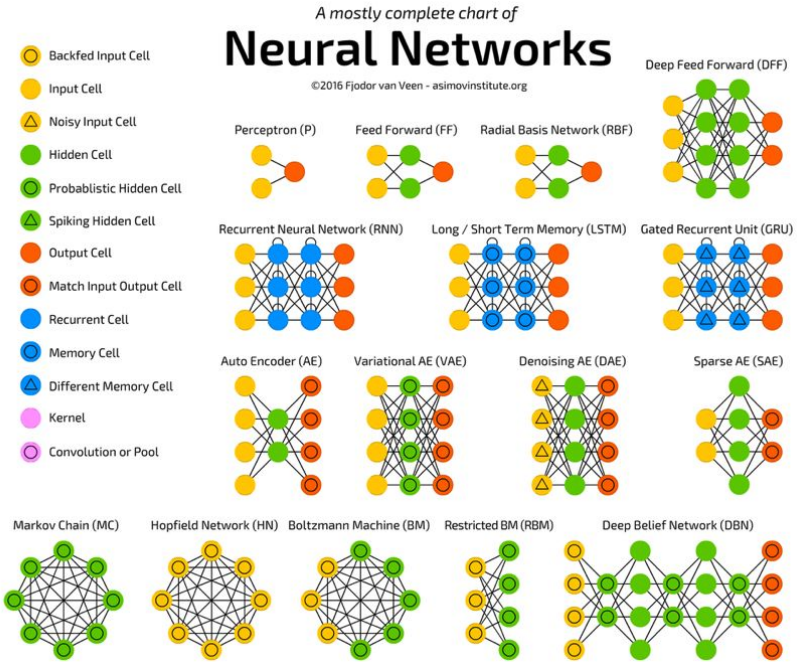
Tell how many planets there are. All the black spots on the engraving, except the comet, represent planets.

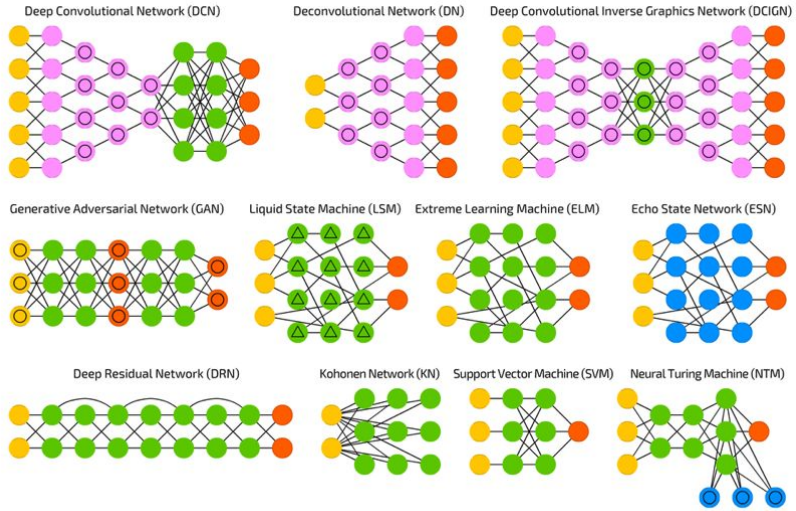
How many moons has the *Earth*? How many has *Jupiter*? *Saturn*? *Uranus*?

Mitchels Solar System, 1846 (source)

Model selection

Standard architectures





CNN, RNN, LSTM, GAN, NAT, AutoML, SVM etc...

Model selection

Hand engineered or learned?

Hand Engineered

Rely on experience and recommendation of experts

Experiment with novel changes to hyperparameters and architecture

Best place to start

Learned

AutoML for hyperparameter and simple architectural optimization

Neural architecture search to learn entire architecture on data

Advanced technique

Model selection

Runtime performance criteria

Accuracy/Precision

Various measurements (e.g. accuracy, precision, recall)

Metrics depend on prediction task

Speed/Latency

Inference time per example

Inference time per batch

Model and runtime environment interaction

Resource Constraints

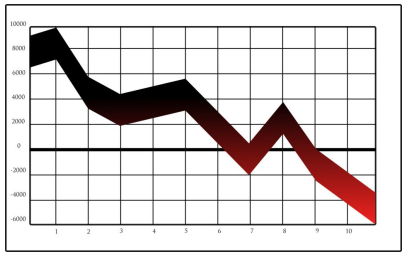
Required memory and power

Model/runtime environment interaction

Mobile and embedded devices severely constrained

Model selection

Training performance criteria



Training Progress

Training and validation loss/accuracy

Time/epochs to convergence

Vanishing/exploding gradient

Time to Train

Model training time can vary by order of magnitude

Longer runs mean fewer trials

Direct impact on time-to-market

Cost

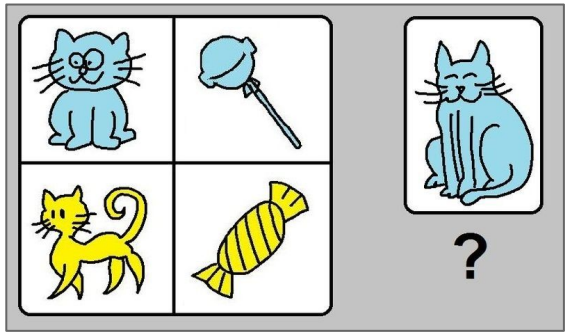
GPU / HPC time is expensive

Opportunity cost of not training other models

Model selection

Sample trade off comparison

Task: image classification



	Logistic Regression	3 Layer CNN	ResNet-50	NASNET
Accuracy	Low	Medium	High	Very High
Inference Memory	Very Low	Low	High	Very High
Inference Latency	Very Low	Low	High	Very High
Training Time	Very Low	Low	High	Very High
Training Cost	Very Low	Very Low	Medium	Medium

Training

Wanderer above the Sea of Fog,
Caspar David Friedrich, 1818 (source)

Primary training patterns

- Train from scratch
- Transfer learn
- Fine tune
- Retrain

Training

Train from Scratch

Wooden frame construction in Sabah, Malaysia (source)

Training

Transfer Learn

“The Barge” at PolarTrec Northeast Scientific Station, Siberia Russia (source)

Training

Fine Tune

WTC under construction, April 2012 (source)

Training

Retrain

Framing for new addition to home (source)

Training techniques

	Train from Scratch	Transfer Learn	Fine Tune	Retrain
When	No pretrained models	Pretrained models for different task	Pretrained model for same task	Pretrained model same task, different number of output classes
Data Requirements	Highest	Reduced	Reduced	Reduced
Training Time	Highest	Reduced	Reduced to Unchanged	Reduced
Domains/tasks involved	1	2	1	1
When Used	No pretrained model, lots of data and compute resources, highest accuracy required	Pretrained model, limited data and compute resources	Pretrained model, additional data or compute resources to improve accuracy	Pretrained model for same task, need to remove or add classes

TF Slim transfer learn example

```
$ python train_image_classifier.py
   --model_name resnet-50
   --dataset_dir ./prepared-data
   --train_dir train
   --checkpoint_path checkpoint/resnet_v1_50.ckpt
   --checkpoint_exclude_scopes resnet_v1_50/logits
   --trainable_scopes resnet_v1_50/logits
```

https://github.com/tensorflow/models/tree/master/research/slim

TF Slim transfer learn example

```
$ python train_image_classifier.py
    --model_name resnet-50
    --dataset_dir ./prepared-data
    --train_dir train
    --checkpoint_path checkpoint/resnet_v1_50.ckpt
    --checkpoint_exclude_scopes resnet_v1_50/logits
    --trainable_scopes resnet_v1_50/logits
```

Model architecture (network)

TF Slim transfer learn example

```
$ python train_image_classifier.py
    --model_name resnet-50
    --dataset_dir ./prepared-data
    --train_dir train
    --checkpoint_path checkpoint/resnet_v1_50.ckpt
    --checkpoint_exclude_scopes resnet_v1_50/logits
    --trainable_scopes resnet_v1_50/logits
```

New data for new task

TF Slim transfer learn example

```
$ python train_image_classifier.py
    --model_name resnet-50
    --dataset_dir ./prepared-data
    --train_dir train
    --checkpoint_path checkpoint/resnet_v1_50.ckpt
    --checkpoint_exclude_scopes resnet_v1_50/logits
    --trainable_scopes resnet_v1_50/logits
```

Model weights from source task (ImageNet)

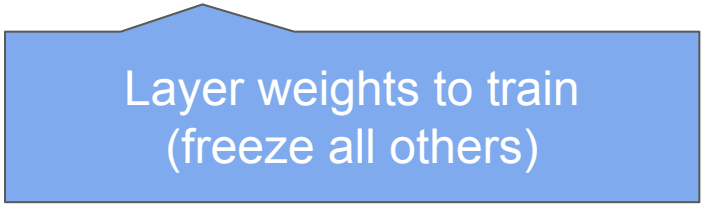
TF Slim transfer learn example

```
$ python train_image_classifier.py
    --model_name resnet-50
    --dataset_dir ./prepared-data
    --train_dir train
    --checkpoint_path checkpoint/resnet_v1_50.ckpt
    --checkpoint_exclude_scopes resnet_v1_50/logits
    --trainable_scopes resnet_v1_50/logits
```

Layer weights to not initialize from checkpoint (unfrozen)

TF Slim transfer learn example

```
$ python train_image_classifier.py
    --model_name resnet-50
    --dataset_dir ./prepared-data
    --train_dir train
    --checkpoint_path checkpoint/resnet_v1_50.ckpt
    --checkpoint_exclude_scopes resnet_v1_50/logits
    --trainable_scopes resnet_v1_50/logits
```

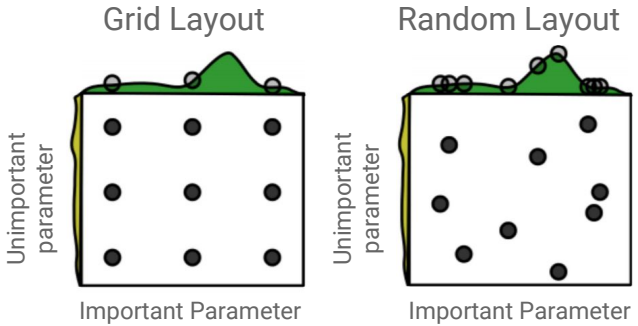


Hyperparameters and tuning

```
$ python train.py
  --learning-rate=0.01
  --activation=relu
  --dropout=0.2
```

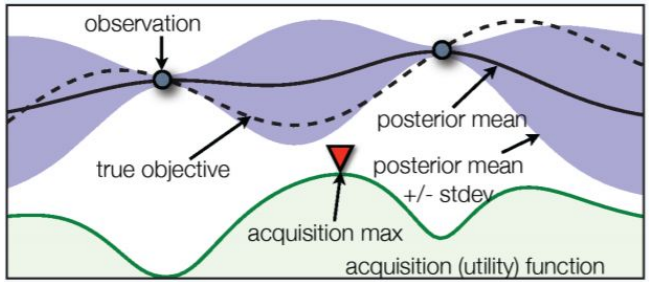
What combination of hyperparameters will train the best model on our data?

Hyperparameter	Search Space
learning-rate	uniform from 1e-4 to 1e-1
activation	choice of “relu” or “sigmoid”
dropout	uniform from 0.1 to 0.9



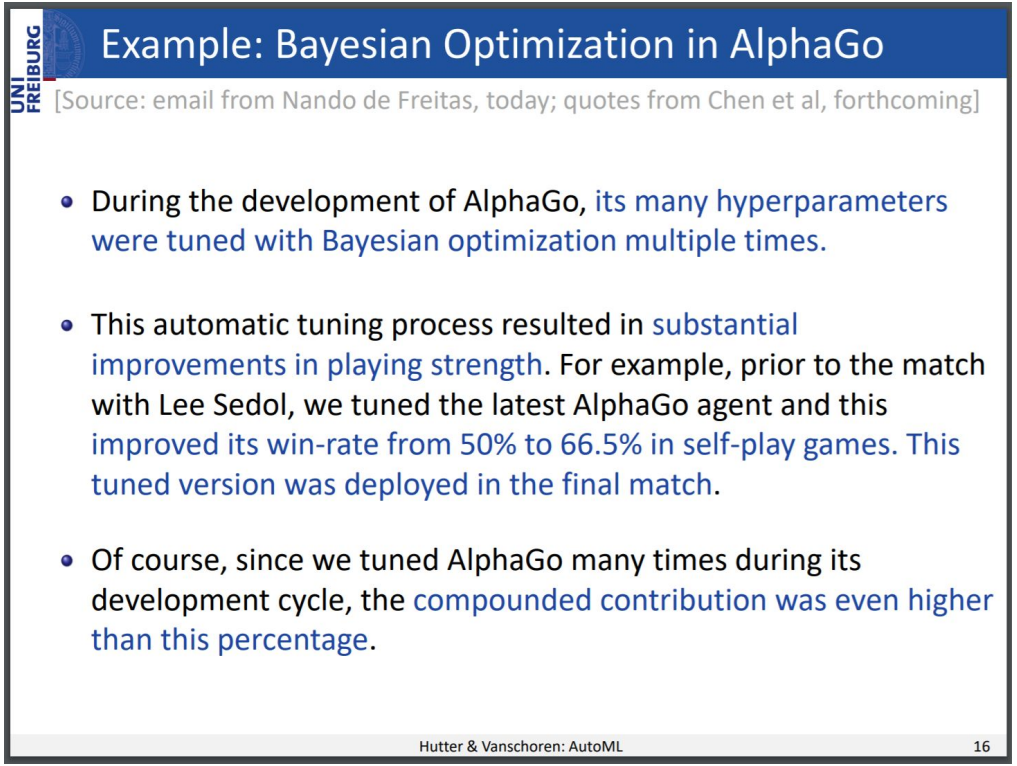
Credit: James Bergstra, Yoshua Bengio (source)

Bayesian Optimization



Credit: Hutter & Vanschoren (source)

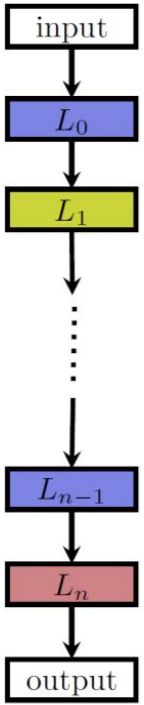
Hyperparameter tuning example



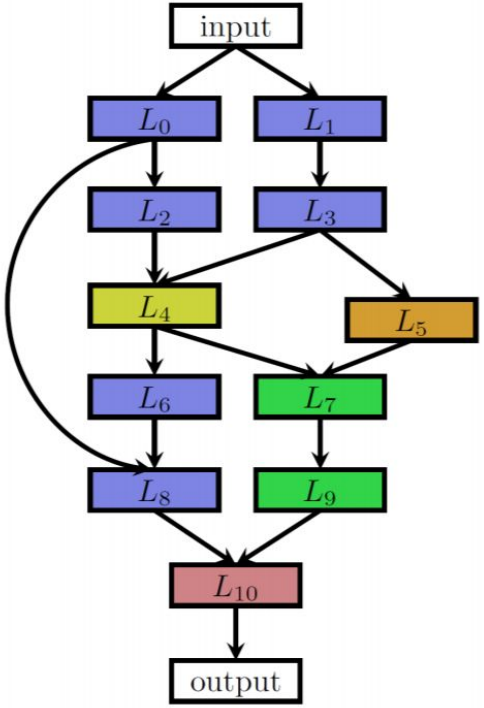
Fom Automatic Machine Learning (AutoML): A Tutorial at NeurIPS 2018 (source)

Architecture search (advanced topic)

Typical Model Layout (layers)

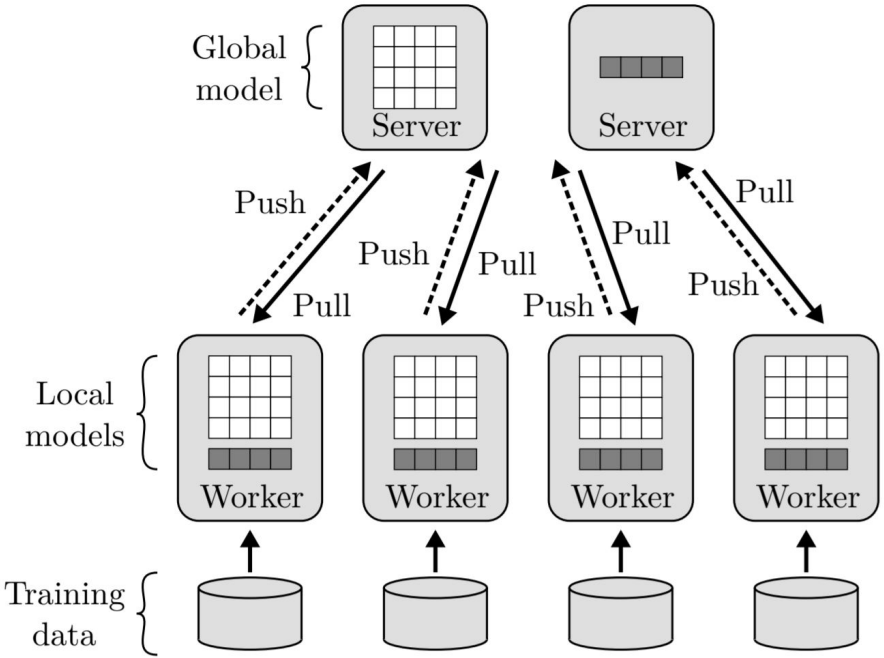


Layers with branches and skip connections



Fom Automatic Machine Learning (AutoML): A Tutorial at NeurIPS 2018 (source)

Distributed training



Credit: Lim, Andersen, and Kaminsky (source)

Motivations for distribution

Too Much Data

Large model (e.g. ResNet-200)

Large batch size (effects accuracy and total training time)

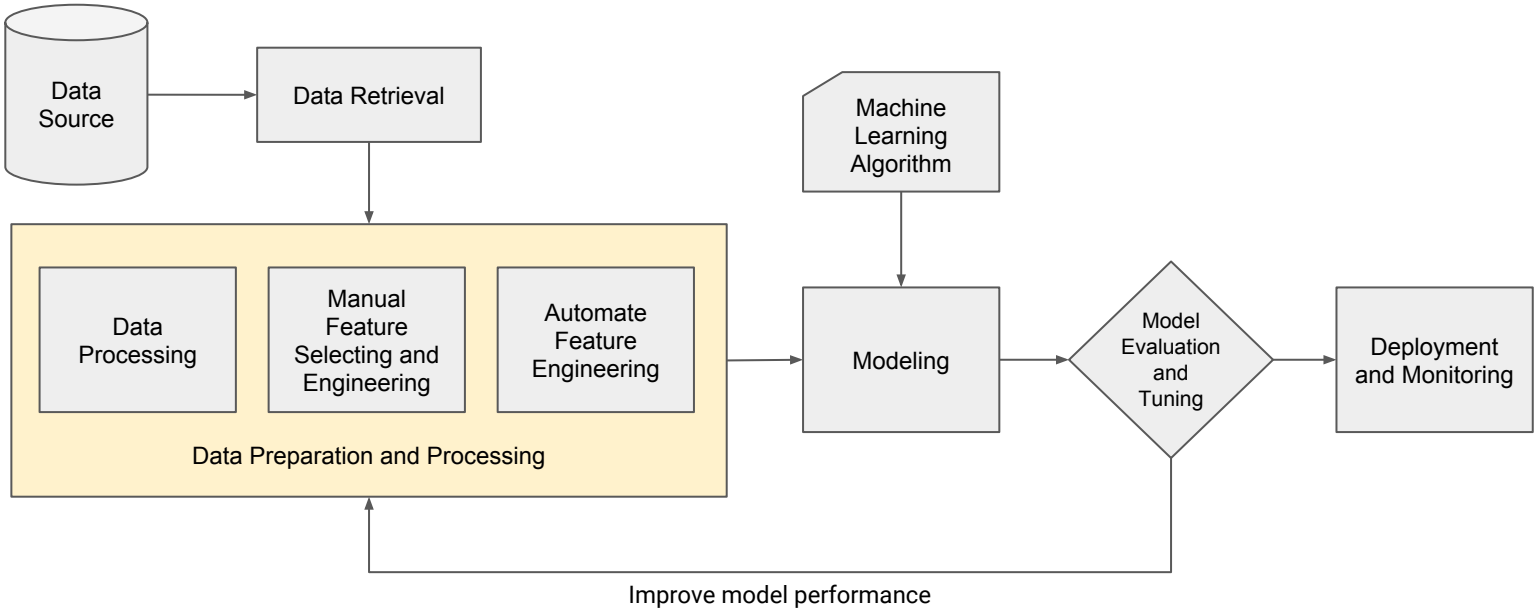
Not Enough Wall Time

Use data or task parallelism to distribute training over multiple GPUs

Data preparation and processing

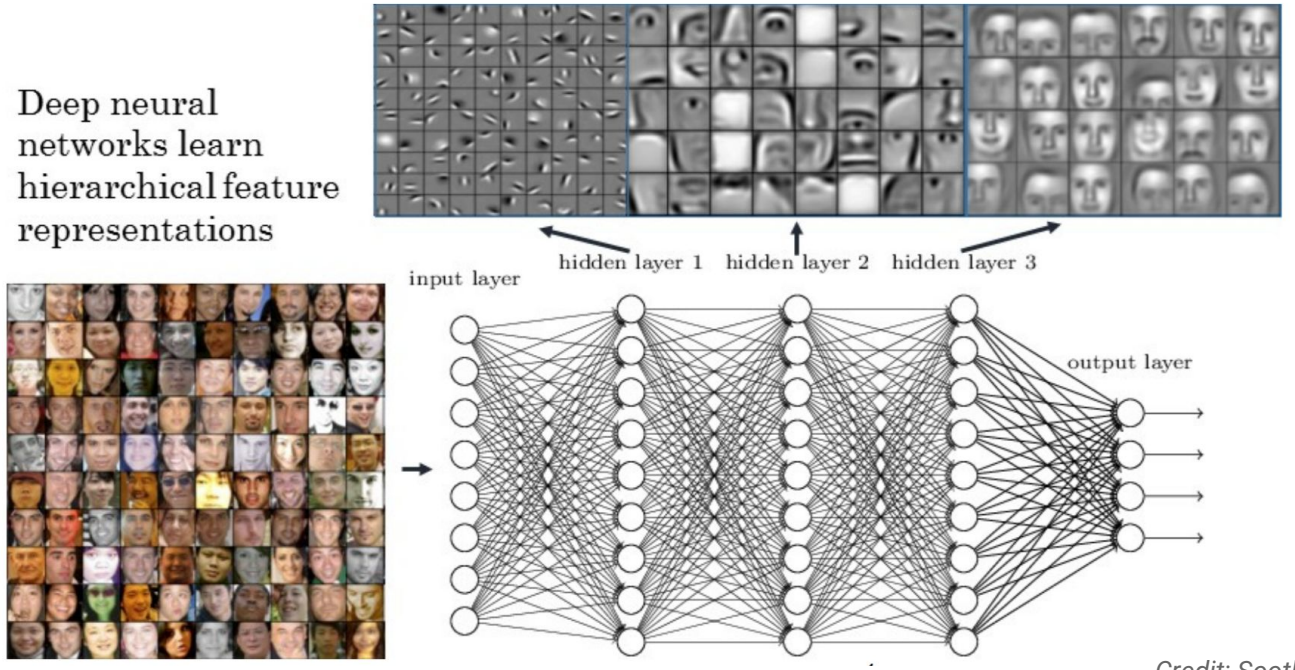
Women lumberjacks at Pityoulish lumber camp, 1941 (source)

Role of data preparation and processing



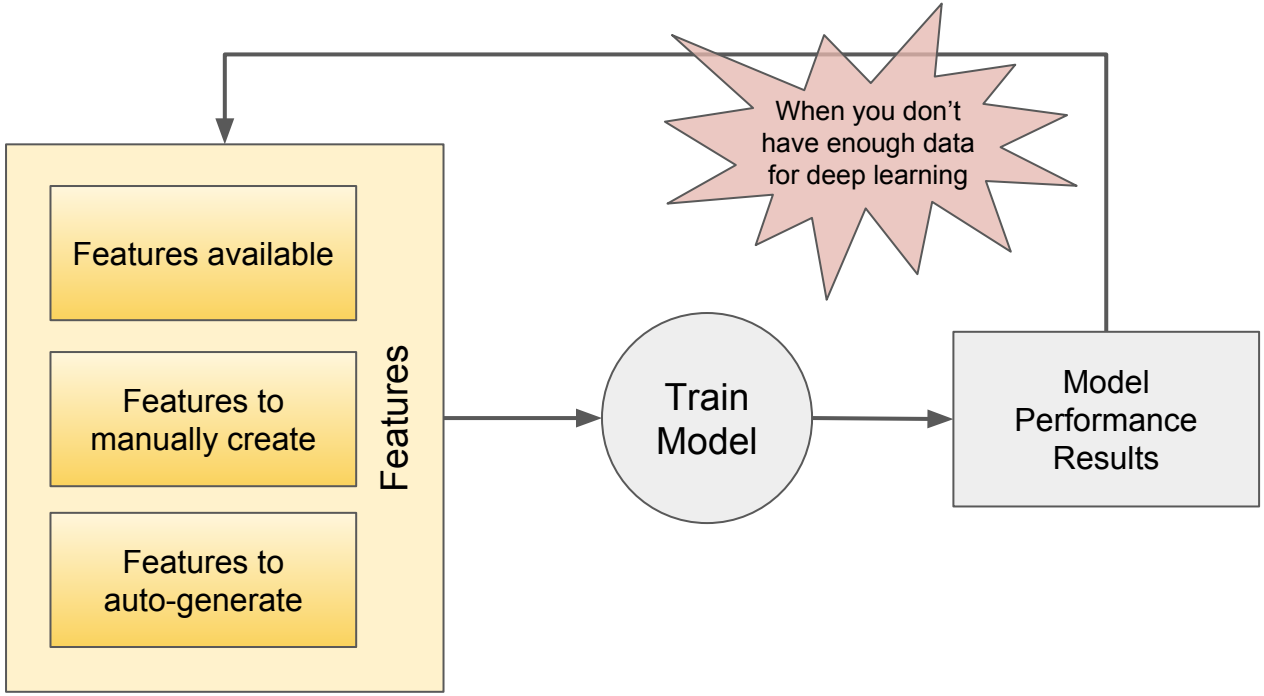
A standard machine learning pipeline (source: Practical Machine Learning with Python, Apress/Springer)

Feature detection in neural networks



Credit: Sootla (source)

Feature selection and engineering



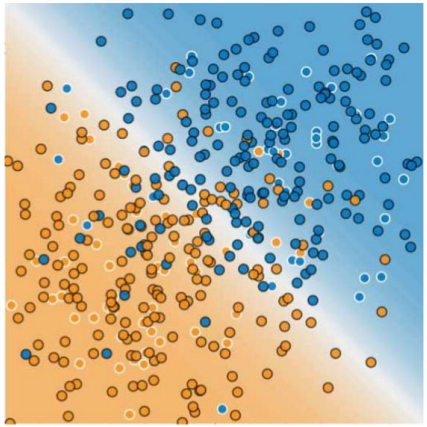
Data splitting and test rules

Training algorithm never, ever sees *validation* and *test* data

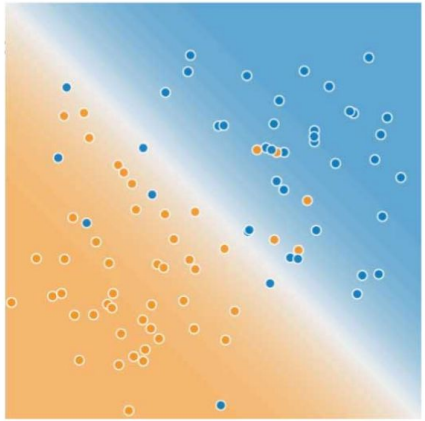
Training orchestrator never, ever sees *test* data

Test data is used for *final scoring* and once used becomes validation data

Validation and test data much have the same distribution as training data

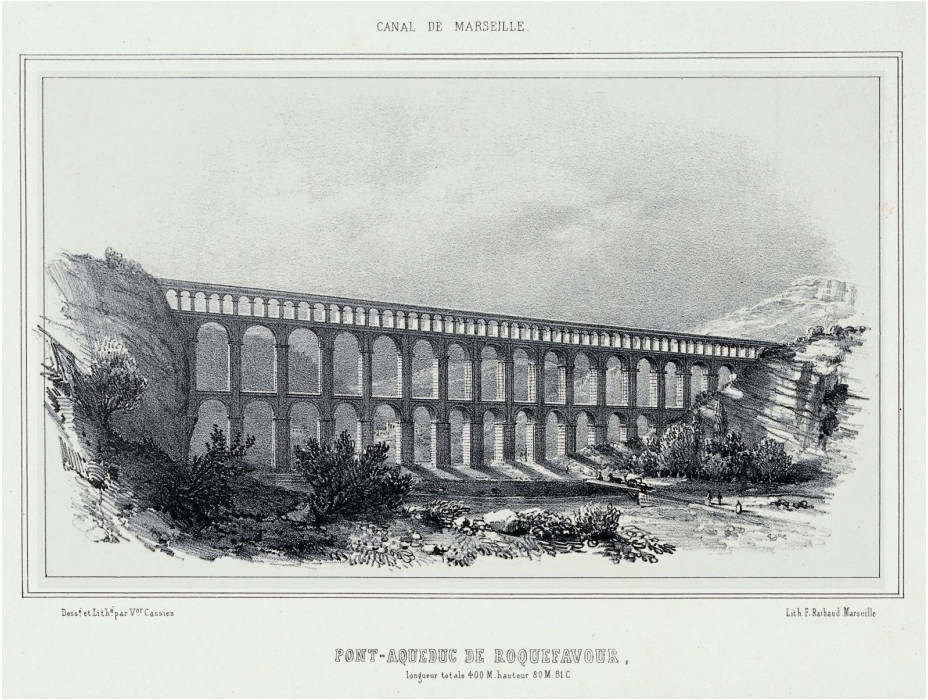


Training Data



Validation/Test Data

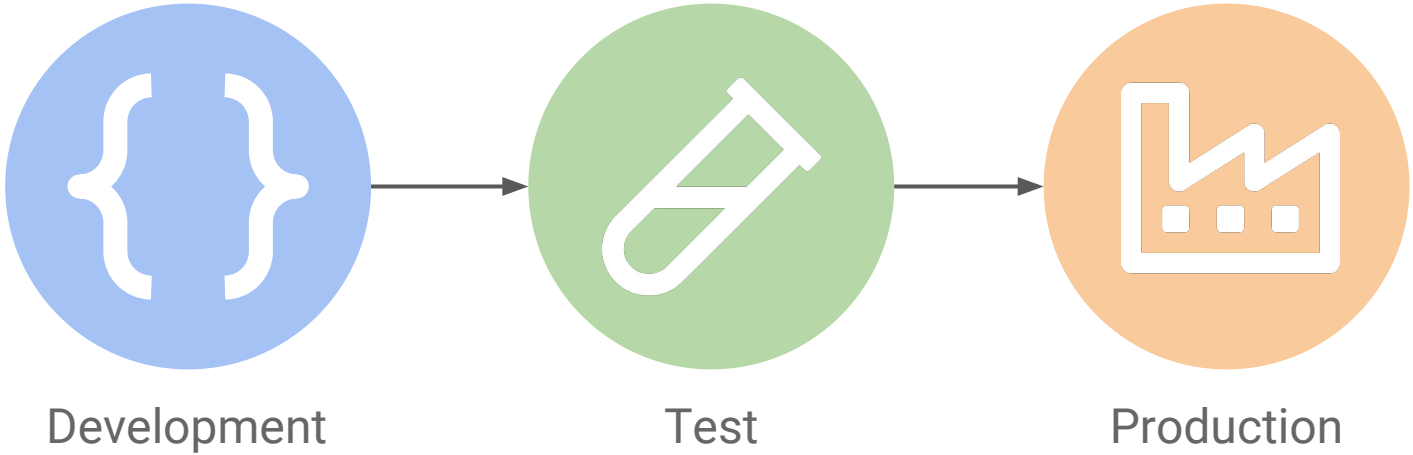
Infrastructure



The Roquefavour bridge-aqueduct over the Canal de Marseille (source)

Infrastructure

Environment isolation



Infrastructure

Workflow management and job scheduling

Apache Airflow

- Automate data pipelines
- ETL
- General workflow

Jenkins

- Continuous integration
- Automate software production pipelines
- General workflow

Kubernetes

- Container orchestration
- General purpose application platform

Infrastructure

Cloud services and accelerators

AWS

- General purpose IaaS
- Standard GPU options
- Track record of improving performance while lowering prices

GCP

- General purpose IaaS
- Standard GPU options and TPUs
- Complement to TensorFlow ecosystem

Azure

- General purpose IaaS
- Standard GPU options

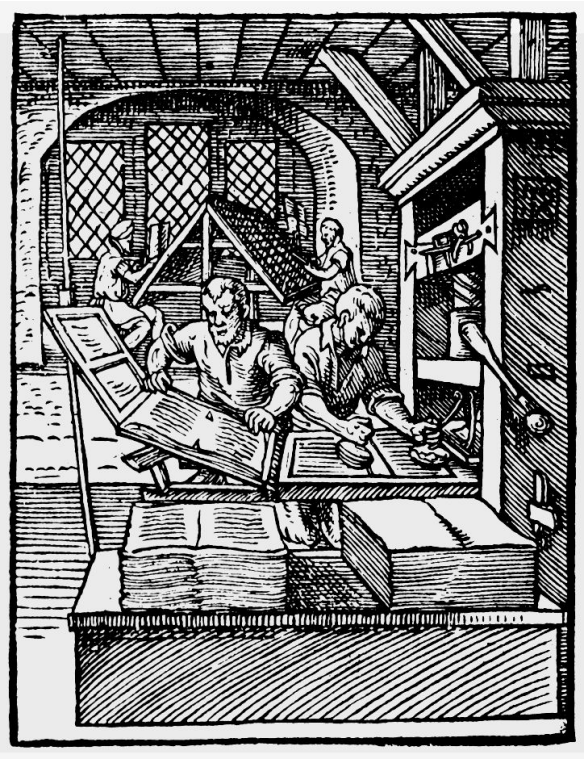
Kubernetes

- Container orchestration
- General purpose application platform

Other GPU

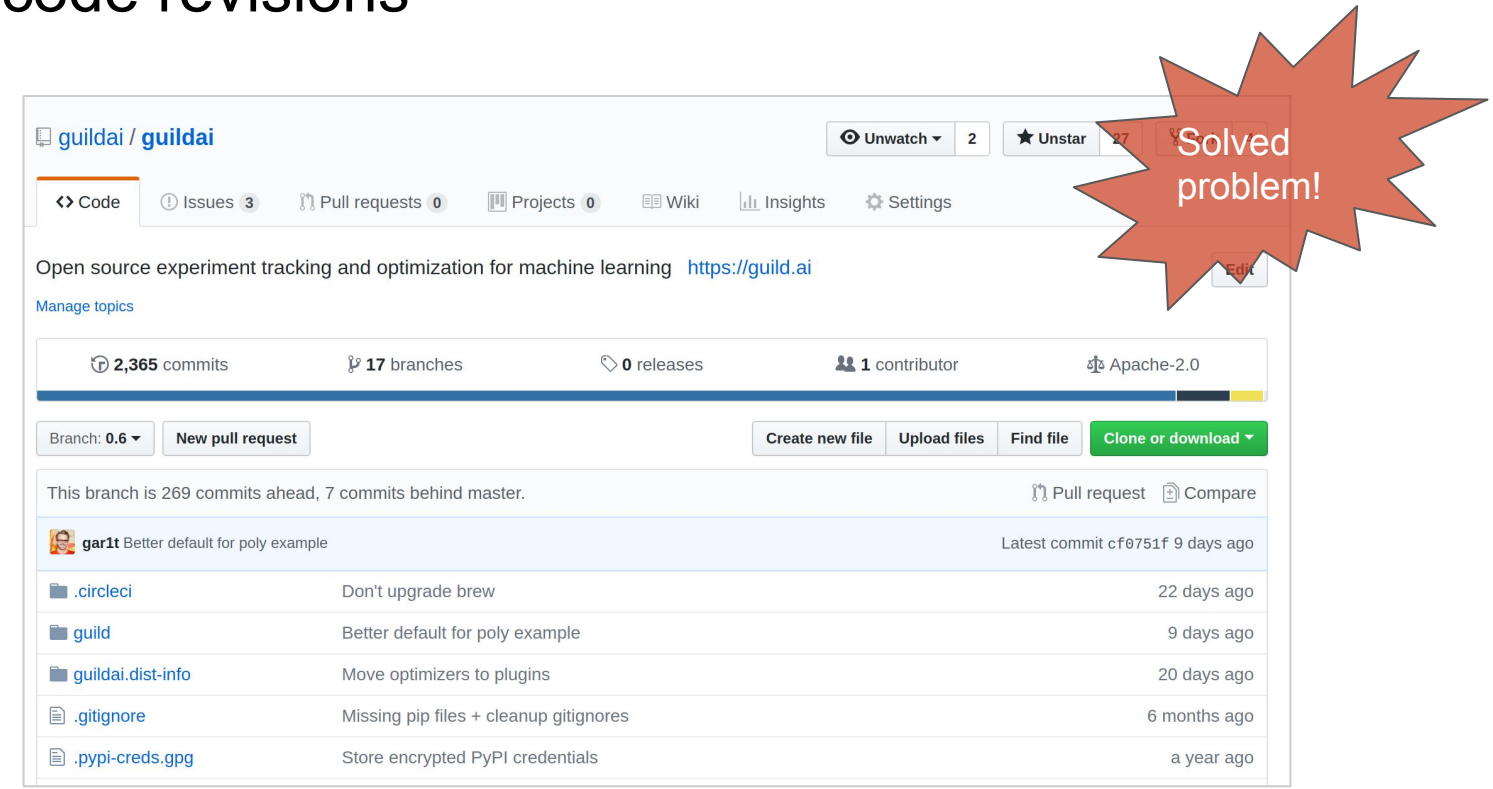
- Dedicated GPU servers - on prem or hosted in datacenter
- Paperspace
- FloydHub

Reproducibility



Early wooden printing press, 1568 (source)

Source code revisions



Data versioning and auditability

File system

+	-
Simple, universal interface	Batch oriented
Wide range of tooling	Highly latent
Secure	
Easily auditable	

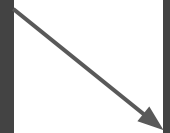
Database

+	-
Real time oriented	Complex
Low latency	
Checkpointing depending on DBMS	

- Track data type
- Identify data that contains private/confidential information
- Anonymize data
- Implement access control and auditability

Experiment automation and management

```
$ guild run train.py lr=0.1
Refreshing project info...
You are about to run train.py
  batch_size: 100
  epochs: 10
  lr: 0.1
Continue? (Y/n)
```



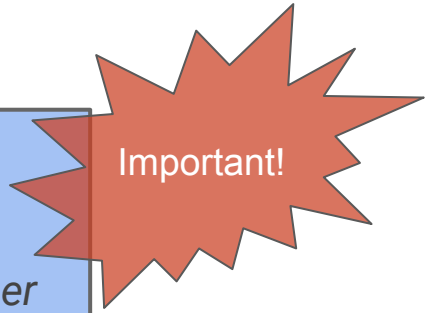
```
$ guild ls
~/.guild/runs/072817ee348d11e98c6cc85b764bbf34:
  data/
  data/t10k-images-idx3-ubyte.gz
  data/t10k-labels-idx1-ubyte.gz
  data/train-images-idx3-ubyte.gz
  data/train-labels-idx1-ubyte.gz
  model/
  model/checkpoint
  model/export.data-00000-of-00001
  model/export.index
  model/export.meta
  train/
  train/events.out.tfevents.1550611600.omaha
  validate/
  validate/events.out.tfevents.1550611600.omaha
```

Experiment

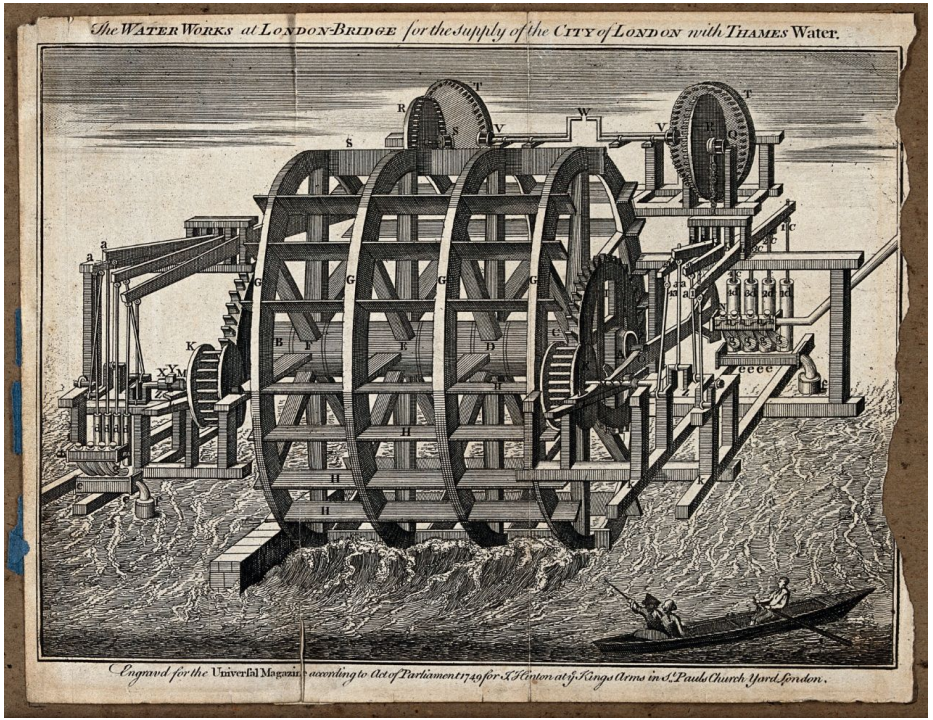
- Metadata (unique ID, model, operation, hyperparameters, time)
- Source code snapshot
- Output (stdio)
- Logs
- Metrics (e.g. loss, accuracy)
- Generated files (e.g. checkpoints)

Experiment automation and management

- No matter how good the result, if it's not reproducible, it's not ready to ship.
- Code review equivalent: *can another engineer easily reproduce this result*? It's a pass/fail grade.
- Without reproducibility, organization is exposed to enormous risk.
- Runs counter to traditional data science tendencies to keep results, tools, and knowledge private.



Production



Water-wheel at London Bridge, 1749 (source)

Production

Serving systems

Batch Inference

Accumulate examples in a file system directory or other similar container (e.g. S3 bucket)

Run batch job to process examples and perform inference (e.g. predict image class)

Simple and effective but highly latent - not suitable for low latency applications

Start here if possible

Online Inference

Requires a serving system (e.g. TF Serving, Deep Detect, or cloud server like Google Cloud ML) - or a Python based system like Flask

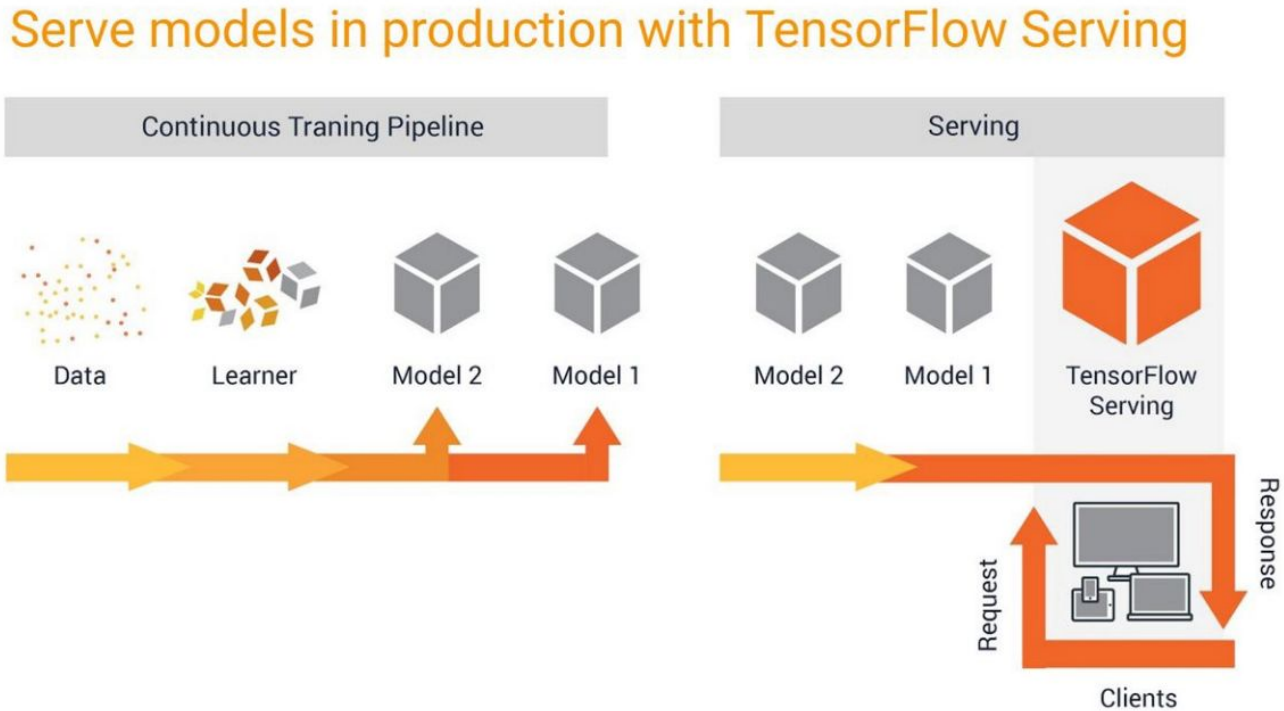
Process examples as they are submitted or accumulate a batch of min size (efficiency)

Python not suitable for performance critical applications - triggers need for native execution

Complex at scale

Production

TensorFlow Serving



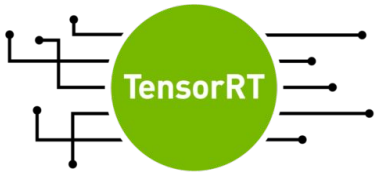
Source

Production

Mobile and embedded platforms

TensorFlow Lite
(ML Kit)

- By Google
- Closely tied to TensorFlow ecosystem
- Android and iOS
- 8 bit quantization



TensorRT

- By NVIDIA
- Embedded and datacenter
- Support for
- 8 bit quantization

CoreML

- By Apple
- iOS only
- 8, 4, 2, or 1 bit quantization

Embedded

- By Intel, ARM, Samsung, lots more
- Varied applications and platform support

Production

Monitoring model and application performance

Open Source


Prometheus


Kibana


Sensu


Nagios


Zabbix

Hosted

Data Dog


New Relic


App Dynamics

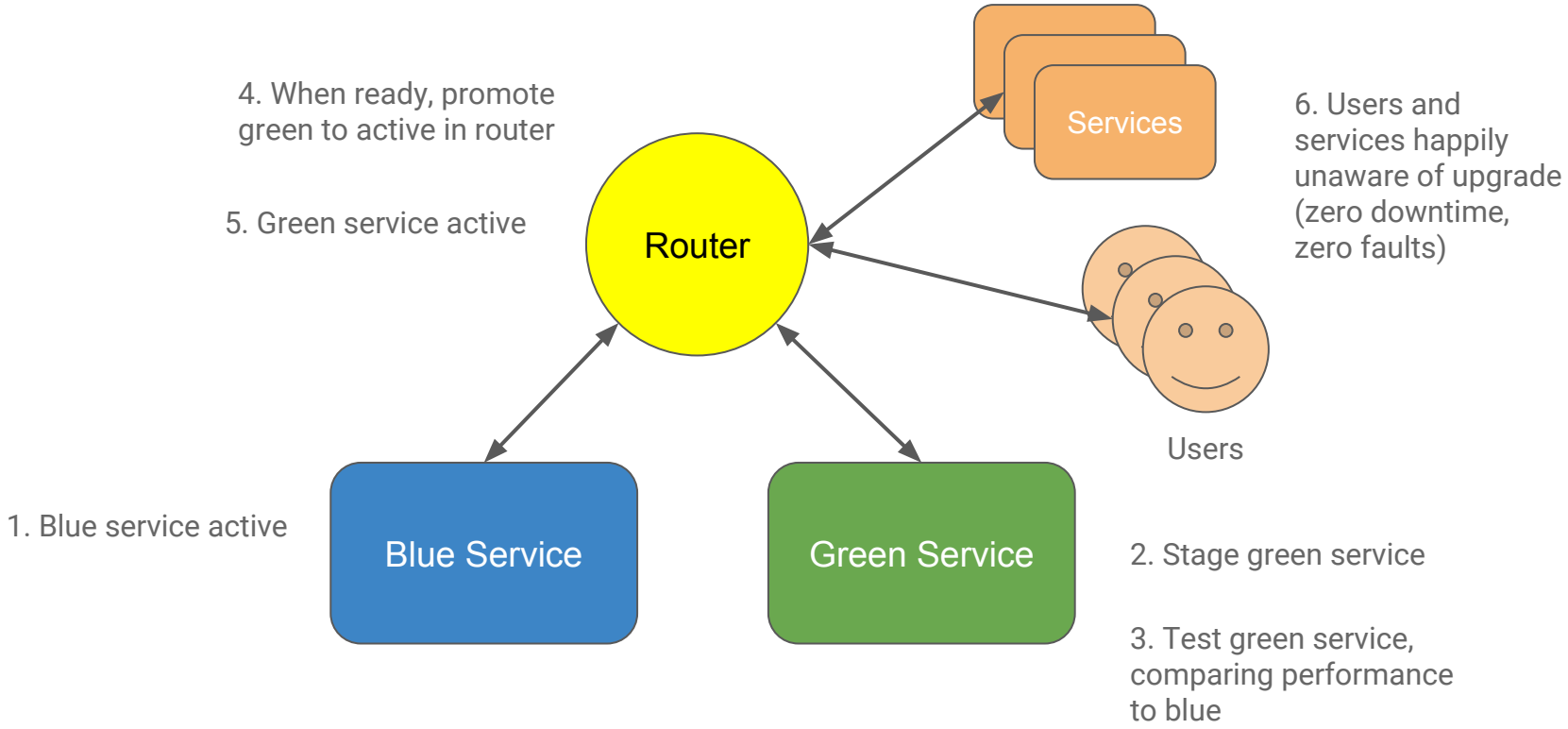

AWS Cloud Watch


Google Stack Driver

Ongoing development

Chicago in 1820 (source)

Upgrading production systems



Ongoing development

Acquiring more data

- Build data acquisition into the application
- Run as a long running, continual process
- Look for new applications to collect new data
- Can bootstrap an application with limited data provided application can collect more

General guidelines

- Commonly revisit: model architecture, data acquisition, data processing, and retraining
- Production systems rely heavily on traditional systems engineering practices that cannot be short-circuited
- Again, without measuring, you’re guessing - even a nominal data collection facility is better than nothing
- Stress collaboration between researchers and engineers

The Story of a Little Gray Mouse, 1945 (source)