



# Machine Learning in Production

# Hello!

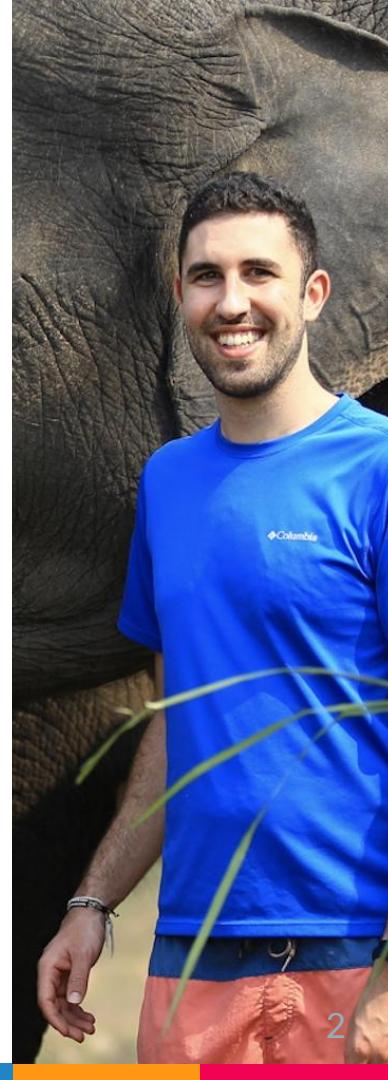
## I am Albert Jiménez

Electrical Engineer @ UPC - Barcelona / CSIRO Australia

4+ years experience with Computer Vision, Deep Learning and Python

2+ years working @ [Triage](#) (Healthcare Startup)

Github: [@jsalbert](#)



# Some of the projects I've done...

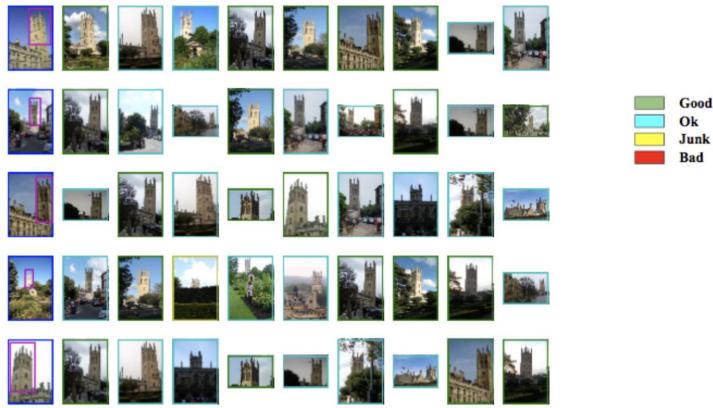
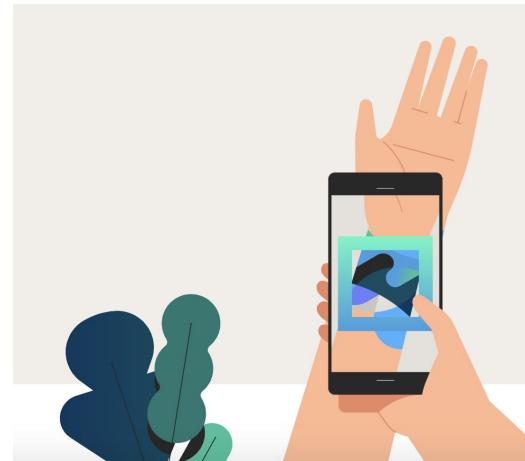
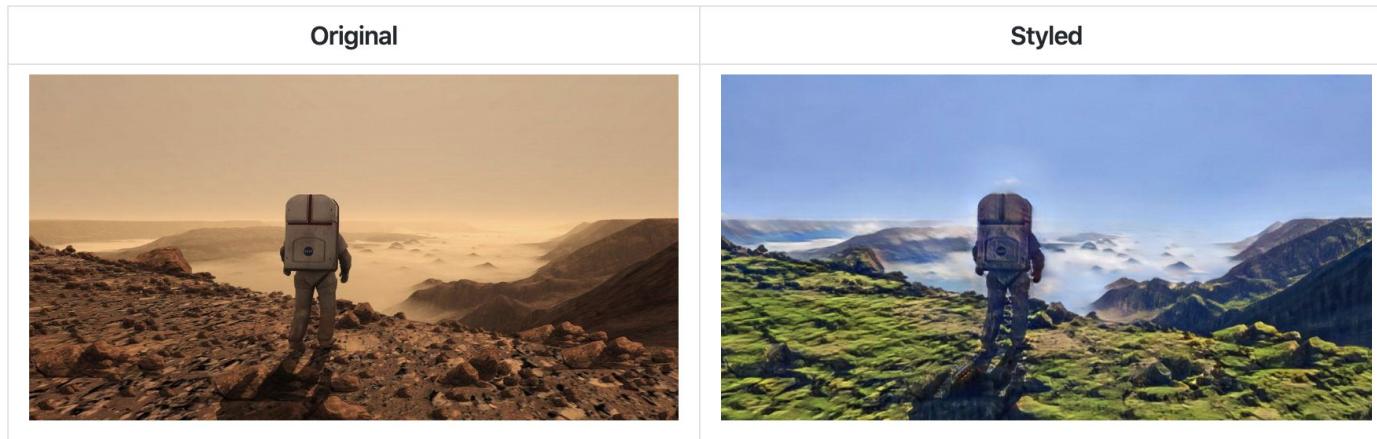


Image Retrieval



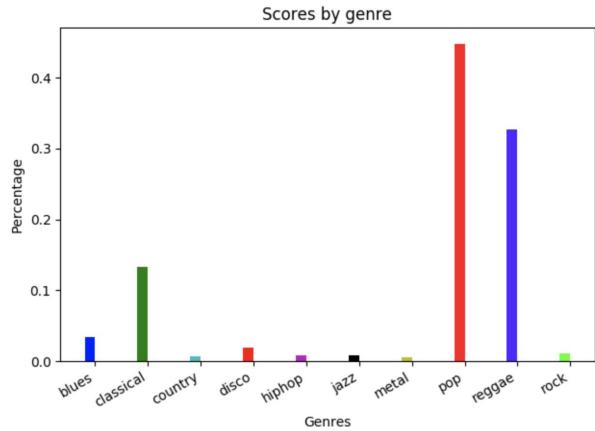
Skin Lesions Recognition

# Some of the projects I've done...



Generative Adversarial Networks

# Some of the projects I've done...



Music Genre Recognition

*In my search of inspiration I build a robot  
Hoping it would give me the perfect love song  
It told me that she was the one I'd imagined  
She smiled sweetly  
She said that I couldn't ask for a better lover  
But it was hard to love you  
I recall her fondly  
The day we met she led me off  
As our passion blew apart  
We split apart  
My robot passion  
She said that we were going to have to fight  
I said "No you don't have to fight  
I know we can't afford to lose your love"  
But she said "You don't have to argue  
I know you can't afford to lose your love"  
  
It wasn't hard to love you  
It wasn't hard to understand  
All that was necessary to be my lover  
Was my love for you  
  
Days that I used to laugh I don't know how  
They all came and fell  
I miss your touch so*

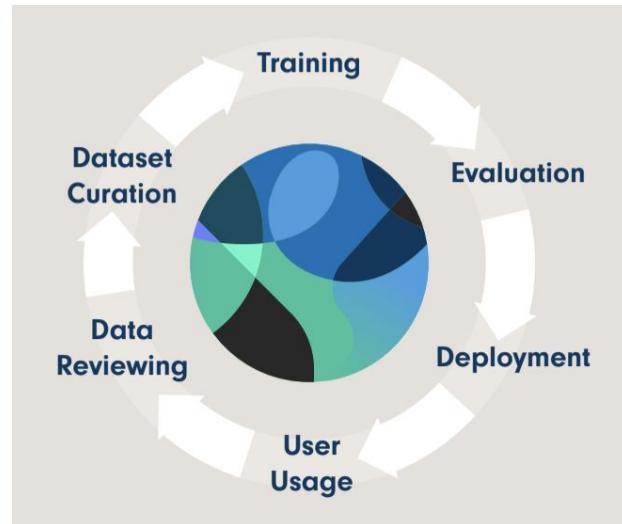
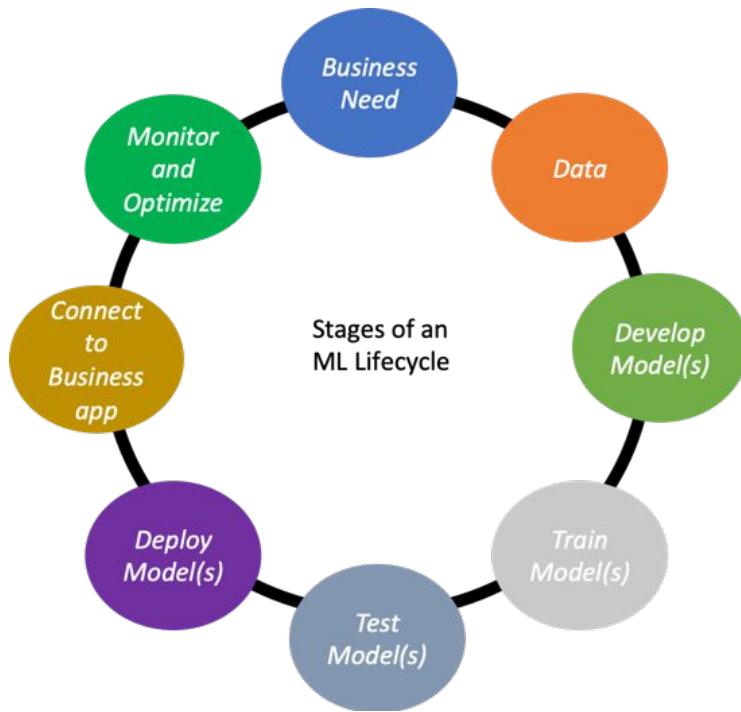
Lyrics Generation

# Agenda

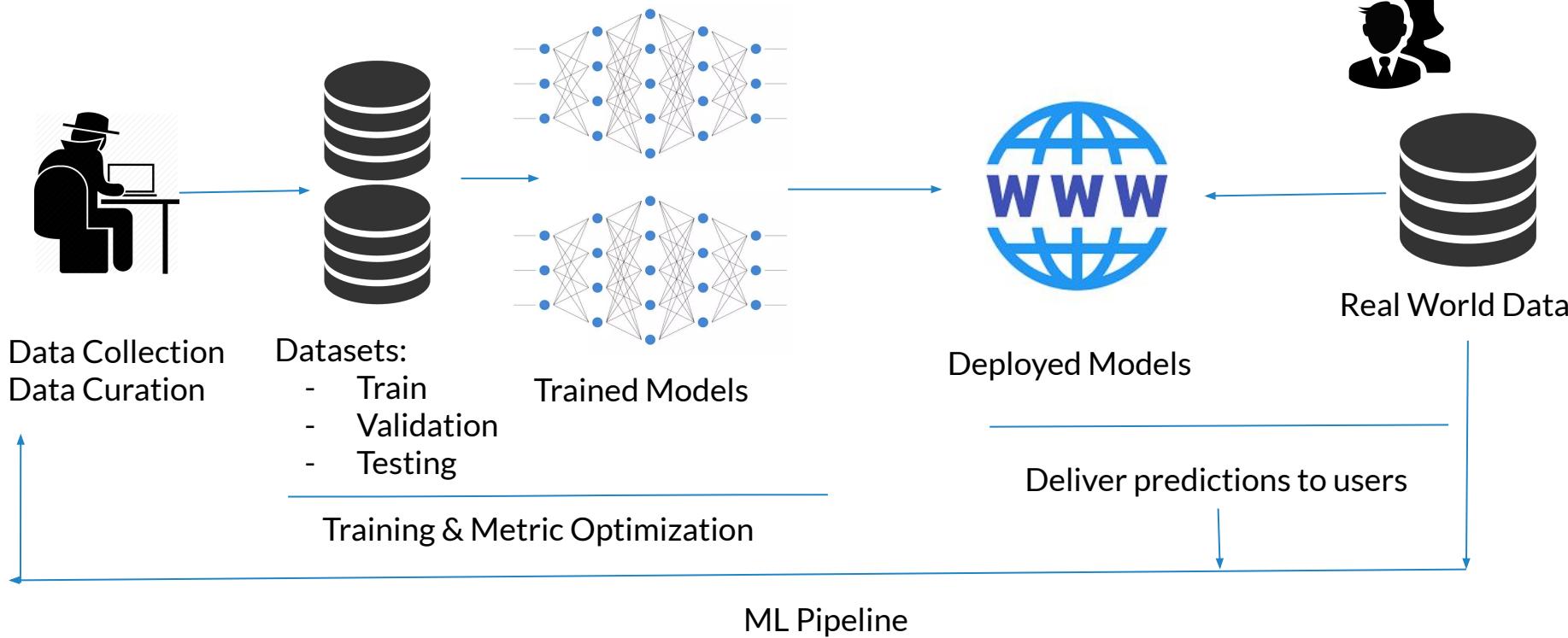
1. Introduction
2. Best Practices for ML in Production
3. Exercises

# 1. Introduction

# ML Life Cycle



# Machine Learning Pipeline



# 2. Best Practices for ML in Production

# When starting a new product...

- Limited access to:
  - Data sources
  - Computational resources
  - User base
  - Working time
- **Planning ahead and ability to iterate fast is the key to success**

# Best Practices of ML

To make great products:

**“Do machine learning like the great engineer you are, not like the great machine learning expert you aren’t”**

<https://developers.google.com/machine-learning/guides/rules-of-ml/> by Martin Zinkevich

# Basic Approach

1. Have a solid pipeline (end to end)
  - a. Data Collection - Train - Test - Deployment
2. Start with a reasonable objective
  - a. Choose a metric that reflects well performance
3. Add features in a simple way
4. Make sure the pipeline stays solid and allows for quick iterations

# Before ML

## 1. Design and implement metrics

- Metrics will help you to:
  - Measure and track the performance of your system over time
  - Allow you to compare different models and make decisions
  - Indicate in which areas of your pipeline you should allocate more resources
    - For instance, data collection for particular cases

# Before ML

## 1. Design and implement metrics

- Metric example
  - You have built a skin cancer binary image classifier and you have a testset containing:
    - 10000 cases of benign lesions (healthy moles)
    - 100 cases of melanomas (malignant skin cancer)
  - **Which metric would you choose to test your model performance?**

# Before ML

## 1. Design and implement metrics

- Accuracy is the most used metric for image classifiers but in our case it has little utility as our dataset is very unbalanced.
- If our model classifies all the images as benign we would have an impressive accuracy of  $10000/10100 = 99\%$

**Accuracy** computes the proportion of correct predictions divided by the total number of samples:

$$\text{Accuracy} = \frac{\text{number of correct predictions}}{\text{number of samples}}$$

# Before ML

## 1. Design and implement metrics

- Sensitivity for the melanoma class would be our perfect metric
- For the previous case we would get a sensitivity value of 0! Which would make obvious that our model is not really doing a good job.

**Sensitivity**, also known as recall, computes the ratio of cases correctly identified as a particular skin condition (true positives) to the total number of images with that condition:

$$\text{Sensitivity} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false negatives}} \quad (2)$$

# Before ML

## 1. Design and implement metrics

- For the same example, which metric would you use to compare 2 different models?

# Before ML

## 1. Design and implement metrics

		True condition			
		Condition positive	Condition negative	Prevalence = $\frac{\sum \text{Condition positive}}{\sum \text{Total population}}$	Accuracy (ACC) = $\frac{\sum \text{True positive} + \sum \text{True negative}}{\sum \text{Total population}}$
Predicted condition	Predicted condition positive	True positive	False positive, Type I error	Positive predictive value (PPV), Precision = $\frac{\sum \text{True positive}}{\sum \text{Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\sum \text{False positive}}{\sum \text{Predicted condition positive}}$
	Predicted condition negative	False negative, Type II error	True negative	False omission rate (FOR) = $\frac{\sum \text{False negative}}{\sum \text{Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\sum \text{True negative}}{\sum \text{Predicted condition negative}}$
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power $= \frac{\sum \text{True positive}}{\sum \text{Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\sum \text{False positive}}{\sum \text{Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR-}}$
		False negative rate (FNR), Miss rate $= \frac{\sum \text{False negative}}{\sum \text{Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\sum \text{True negative}}{\sum \text{Condition negative}}$	Negative likelihood ratio (LR-) = $\frac{\text{FNR}}{\text{TNR}}$	

[https://en.wikipedia.org/wiki/Confusion\\_matrix](https://en.wikipedia.org/wiki/Confusion_matrix)

# Before ML

## 1. Design and implement metrics

		True condition			
		Condition positive	Condition negative	Prevalence = $\frac{\sum \text{Condition positive}}{\sum \text{Total population}}$	Accuracy (ACC) = $\frac{\sum \text{True positive} + \sum \text{True negative}}{\sum \text{Total population}}$
Predicted condition	Predicted condition positive	True positive	False positive, Type I error	Positive predictive value (PPV), Precision = $\frac{\sum \text{True positive}}{\sum \text{Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\sum \text{False positive}}{\sum \text{Predicted condition positive}}$
	Predicted condition negative	False negative, Type II error	True negative	False omission rate (FOR) = $\frac{\sum \text{False negative}}{\sum \text{Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\sum \text{True negative}}{\sum \text{Predicted condition negative}}$
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power $= \frac{\sum \text{True positive}}{\sum \text{Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\sum \text{False positive}}{\sum \text{Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR-}}$
		False negative rate (FNR), Miss rate $= \frac{\sum \text{False negative}}{\sum \text{Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\sum \text{True negative}}{\sum \text{Condition negative}}$	Negative likelihood ratio (LR-) = $\frac{\text{FNR}}{\text{TNR}}$	

[https://en.wikipedia.org/wiki/Confusion\\_matrix](https://en.wikipedia.org/wiki/Confusion_matrix)

# Before ML

## 1. Design and implement metrics

- Important to add them sooner than later:
  - Easier to gain user's permission early on
  - Getting historical data for possible concerning areas in the future
  - Storing metrics on database will save you to look and parse long logs
  - You will notice trends over time

# Before ML

## 1. Design and implement metrics

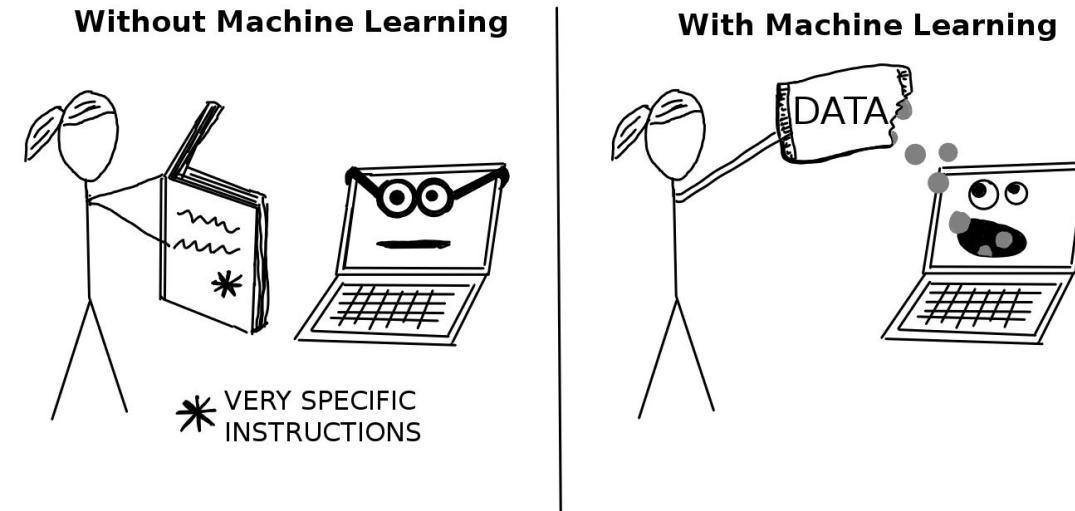
- Sometimes it is very hard to find the perfect metric for a product



- Tracking a good variety of them is a good idea to see how they interoperate

# Before ML

## 2. Don't be afraid to launch a product without machine learning



<https://christophm.github.io/interpretable-ml-book/terminology.html>

# Before ML

## 2. Don't be afraid to launch a product without machine learning

- Use simple heuristics first
  - Very easy to see if correlation with your metrics exist

### Ranking / Suggesting Apps in Google Play or App Store

- Use number of downloads
- Use user scores



# Before ML

## 2. Don't be afraid to launch a product without machine learning

- Use simple heuristics first
  - Very easy to see if correlation with your metrics exist
- Football vs Futbol

Image Metadata  
Geographical location



# Before ML

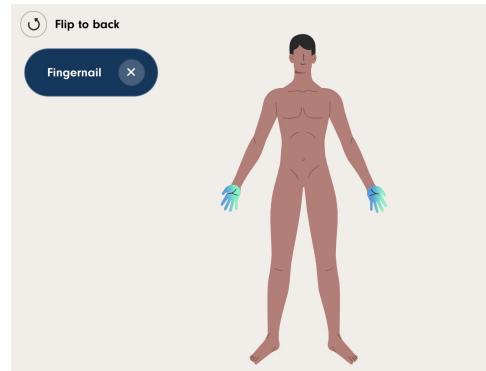
## 2. Don't be afraid to launch a product without machine learning

- Use a rule based system
  - System that by definition must work that way

### Predicting skin diseases



User input



Select Body Location



Filter Output

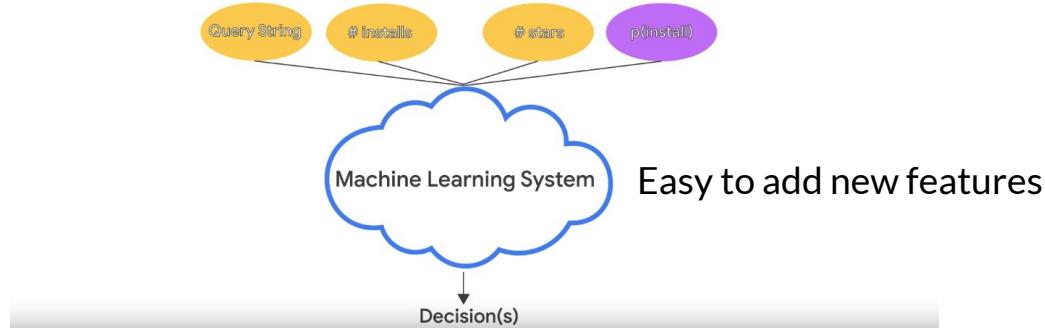
# First steps with ML

## 3. Choose Machine Learning over a complex heuristic

Heuristic 1	# of views
Heuristic 2	# of stars

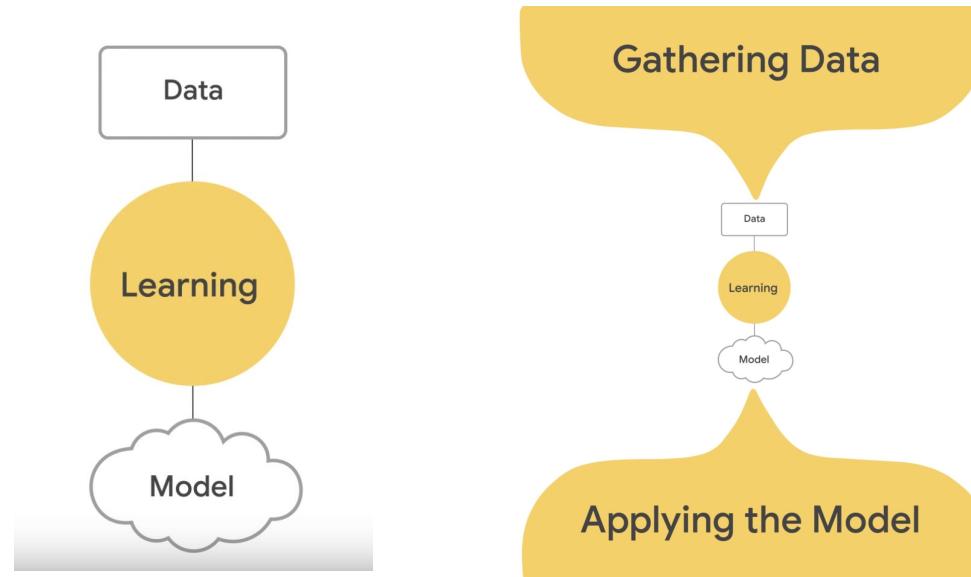
$$(\# \text{ stars})^{2/3} + 2(\# \text{ installs})$$

Hard to maintain & iterate



# First steps with ML

## 4. Keep the first model simple and the infrastructure of your pipeline right



“Perfect is the enemy of the good”

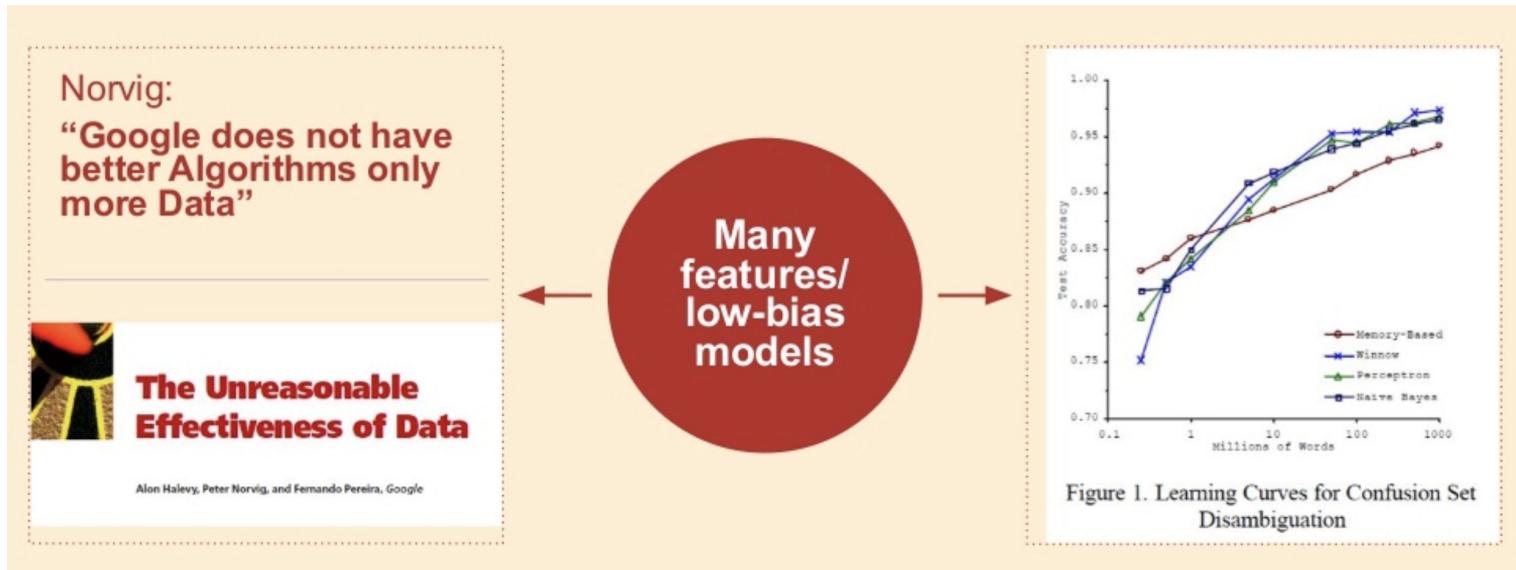
# First steps with ML

## 4. Keep the first model simple and the infrastructure of your pipeline right

- Focus on how are you going to collect and curate the data
  - What is better **More** data or **Better** data?
  - Reviewing and labeling data

# First steps with ML

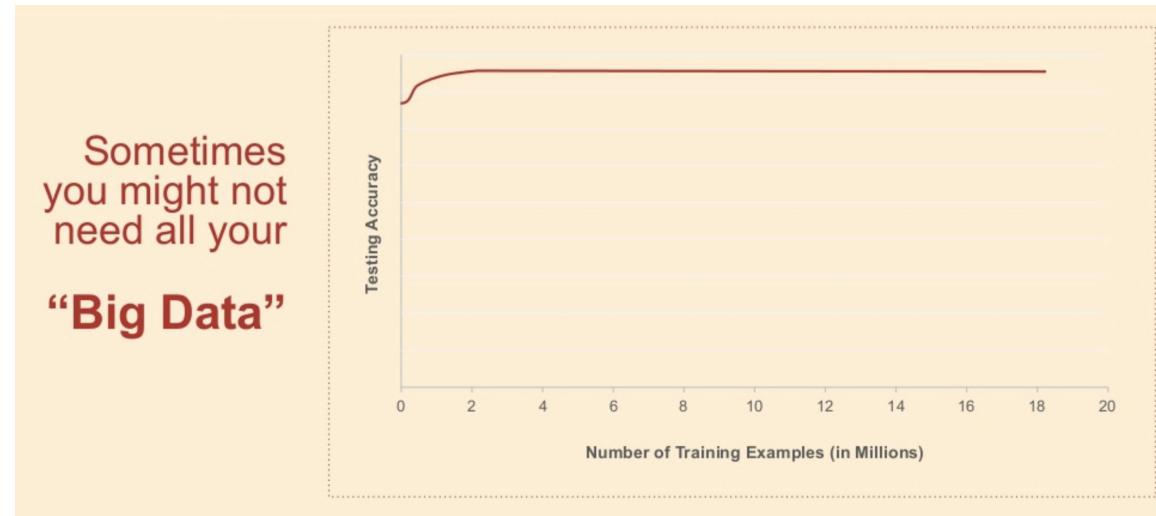
## 4. Keep the first model simple and the infrastructure of your pipeline right



<https://www.slideshare.net/xamat/lessons-learned-from-building-practical-deep-learning-systems>

# First steps with ML

## 4. Keep the first model simple and the infrastructure of your pipeline right



<https://www.slideshare.net/xamat/lessons-learned-from-building-practical-deep-learning-systems>

# First steps with ML

## 4. Keep the first model simple and the infrastructure of your pipeline right

The collage includes:

- Top left: Icons for "THE TRANSFORMER" (flame), "ULM-FIT" (blue circle), "OpenAI Transformer" (pink flame), and "ELMo" (red circle).
- Bottom left: A "WIKIPEDIA" logo featuring a globe and the text "The Free Encyclopedia".
- Middle center: A diagram of the Inception v3 architecture, showing a "stem" leading to multiple "inception modules" and ending at an "output classifier".
- Bottom center: A collage of images forming the word "IMAGENET".
- Bottom left sidebar:
  - NMT**: An open-source neural machine translation system.
  - Models and Recipes**
  - Pretrained
- Bottom middle sidebar:
  - Available models trained using OpenNMT**
    - English → German
    - German → English
    - English Summarization
    - Multi-way – FR,ES,PT,IT,RO <> FR,ES,PT,IT,RO
  - More models coming soon:**
    - Ubuntu Dialog Dataset
    - Syntactic Parsing
    - Image-to-Text

Fine-Tune with domain specific better data

<https://www.slideshare.net/xamat/lessons-learned-from-building-practical-deep-learning-systems>

# First steps with ML

## 4. Keep the first model simple and the infrastructure of your pipeline right

Note: You can add max 3 diagnoses

Diagnosis	Confidence
melanoma	75%
benign melanocytic neoplasm	15%
None of the above	10%

image\_id:1075880

Is the image dermoscopic (taken with a dermatoscope)?

Yes  No

[Add diagnosis](#) [Submit](#)

[Instructions](#) [Poor quality](#) [Stop reviewing](#)

Labeling



# First steps with ML

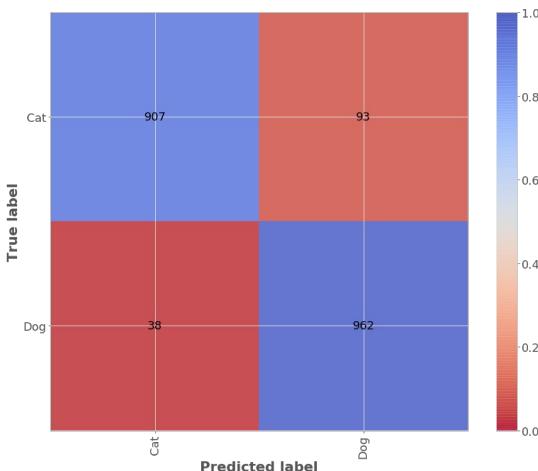
## 4. Keep the first model simple and the infrastructure of your pipeline right

Year	Breakthrough in AI	Datasets (First Available)	Algorithms (First Proposal)
1994	Human-level spontaneous speech recognition	Spoken Wall Street Journal articles and other texts (1991)	Hidden Markov Model (1984)
1997	IBM Deep Blue defeated Garry Kasparov	700,000 Grandmaster chess games, aka "The Extended Book" (1991)	Negascout planning algorithm (1983)
2005	Google's Arabic- and Chinese-to-English translation	1.8 trillion tokens from Google Web and News pages (collected in 2005)	Statistical machine translation algorithm (1988)
2011	IBM Watson became the world Jeopardy! Champion	8.6 million documents from Wikipedia, Wiktionary, Wikiquote, and Project Gutenberg (updated in 2005)	Mixture-of-Experts algorithm (1991)
2014	Google's GoogLeNet object classification at near-human performance	ImageNet corpus of 1.5 million labeled images and 1,000 object categories (2010)	Convolution neural network algorithm (1989)
2015	Google's DeepMind achieved human parity in playing 29 Atari games by learning general control from video	Arcade Learning Environment dataset of over 50 Atari games (2013)	Q-learning algorithm (1992)
Average No. Of Years to Breakthrough		3 years	18 years

# First steps with ML

## 4. Keep the first model simple and the infrastructure of your pipeline right

- Model validation: reliable test set + metrics you trust + evaluation pipeline



class	sensitivity	precision	f1_score	specificity	FDR	AUROC	TP	FP	FN	% of samples
cats	0.907	0.960	0.933	0.962	0.040	1.0	907	38	93	50.0
dogs	0.962	0.912	0.936	0.907	0.088	1.0	962	93	38	50.0



# First steps with ML

## 4. Keep the first model simple and the infrastructure of your pipeline right

- Software Infrastructure

### Option 1 - Researchy

Favor experimentation and only send to production when we have nice results.

Have ML researchers coding in R, Python and then have developers/engineers implementing/optimizing things for production.

### Option 2 - Full production

Favor production and have a team struggle to carry out quick experiments and iterate fast.

Implement highly optimized C code.

# First steps with ML

## 4. Keep the first model simple and the infrastructure of your pipeline right

### Software Infrastructure

Balance between option 1 and 2

- Use software tools like Tensorflow, PyTorch, Scikit-Learn to experiment and do research. Try to use the optimizations that these tools provide for production. Implement optimizations when needed.
- Implement abstractions / API on top of optimized implementations so they can be accessed from friendly interfaced experimentation tools

# First steps with ML

## 4. Keep the first model simple and the infrastructure of your pipeline right

- Write complete integration tests, unit tests
- You have model versioning, data versioning and code documentation



# First steps with ML

## 4. Keep the first model simple and the infrastructure of your pipeline right

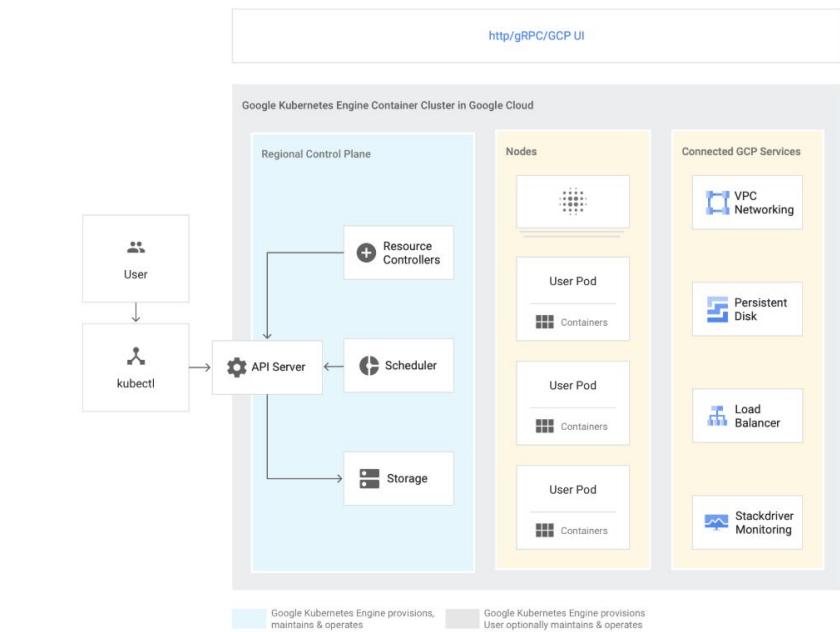
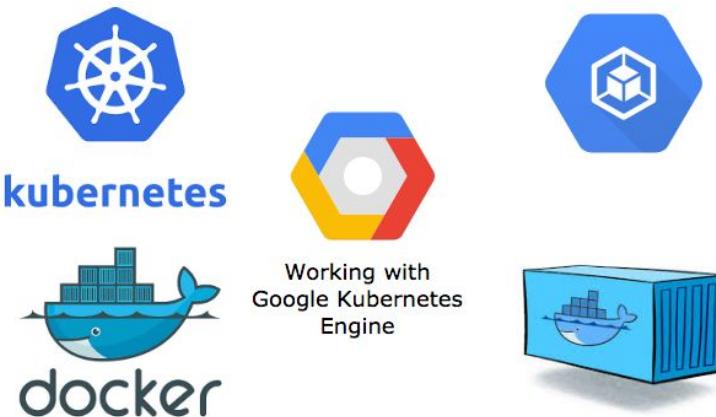
- How are you going to integrate the model within your application
  - Model deployment - (Docker, Kubernetes, GCloud, AWS)
  - How are the predictions going to be computed (Offline/Online)
    - If Online - Will you be able to have enough bandwidth?
  - How are you going to store metrics, user inputs, predictions



# First steps with ML

## 4. Keep the first model simple and the infrastructure of your pipeline right

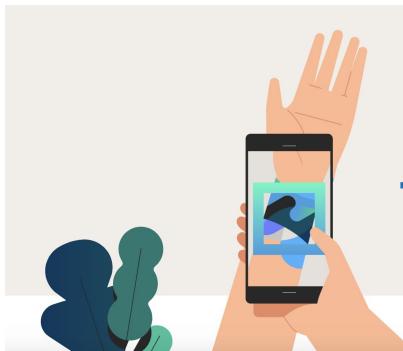
- Kubernetes deployment



# First steps with ML

## 5. Know the refresh rate requirements of your system -- Training Time

- Static vs Dynamic



Skin Classification



News Recommendation System

# First steps with ML

## 5. Know the refresh rate requirements of your system -- Training Time

- Static Model - Offline Training
  - Easy to build and test -- use batch train & test, iterate until good.
  - Still requires monitoring of inputs
  - Easy to let this grow stale

# First steps with ML

## 5. Know the refresh rate requirements of your system -- Training Time

- Dynamic Model - Online Training
  - Feed in training data over time.
  - Use progressive validation rather than batch training & test
  - Needs monitoring, model rollback & data quarantine capabilities
  - Will adapt to changes, staleness issues avoided

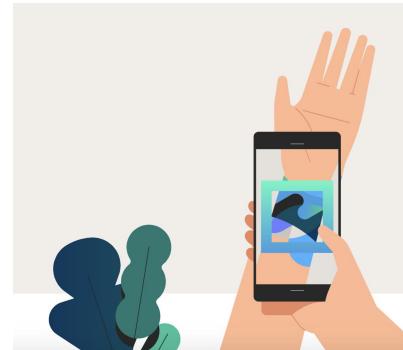
# First steps with ML

## 5. Know the refresh rate requirements of your system -- Inference Time

- Offline vs Online



Weekly Movie Recommendation



Skin Classification

# First steps with ML

## 5. Know the refresh rate requirements of your system -- Inference Time

- Offline
  - Make all possible predictions in a batch.
  - Write to a table, then feed these to a cache/lookup table.
  - **Upside:** don't need to worry much about cost of inference.
  - **Upside:** can do post-verification on predictions on data before pushing.
  - **Downside:** can only predict things we know about.
  - **Downside:** update latency likely measured in hours or days.

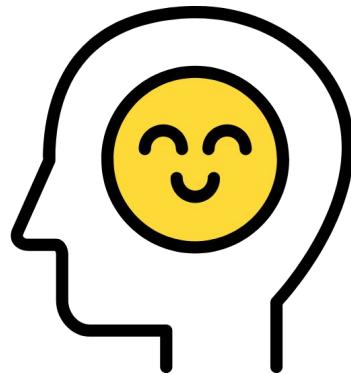
# First steps with ML

## 5. Know the refresh rate requirements of your system -- Inference Time

- Online
  - Predict on demand, using a server.
  - **Upside:** can predict any new item as it comes in.
  - **Downside:** compute intensive, latency sensitive -- may limit model complexity.
  - **Downside:** monitoring needs are more intensive.

# First steps with ML

6. Choose a simple, observable and attributable metric for your first objective



User Happiness

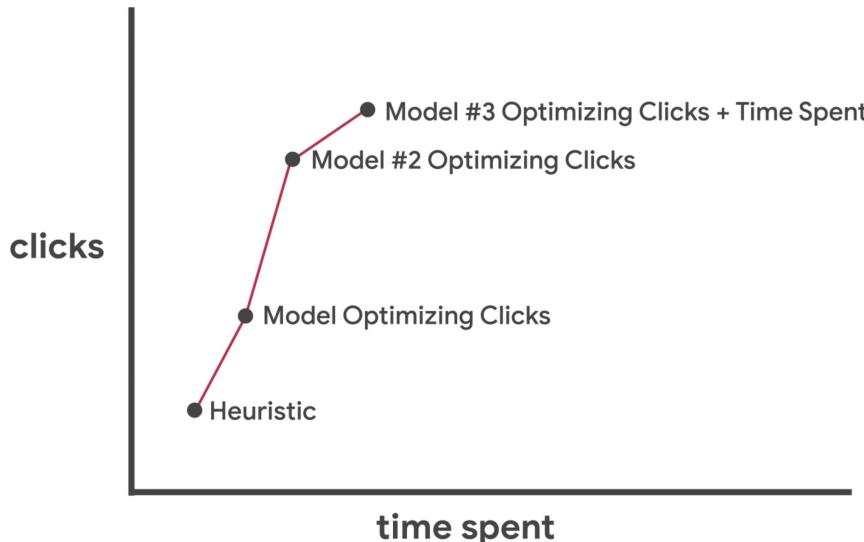
VS



Number of Clicks

# First steps with ML

## 7. Don't overthink the machine learning objective that you choose to optimize



- An objective or cost function is what your machine learning model is trying to optimize for during training
- Probably optimizing for one metric will provide an increase on the rest (correlation)

# First steps with ML

## 8. Do prior research on the topic! don't try to reinvent the wheel

When you get a Deep Learning project idea and you can't find any papers on it



You'll find that people probably have worked on that before!

# First steps with ML

## 8. Do prior research on the topic! don't try to reinvent the wheel

Datasets:

- <https://toolbox.google.com/datasetsearch>, <https://www.kaggle.com/>

Dataset Search

🔍

Try [boston education data](#) or [weather site:noaa.gov](#)

[Find out more](#) about including your datasets in Dataset Search.



# First steps with ML

## 8. Do prior research on the topic! don't try to reinvent the wheel

Papers, algorithms, metrics, objective functions, common failure scenarios

- <https://arxiv.org/>, <http://www.arxiv-sanity.com/>



### Arxiv Sanity Preserver

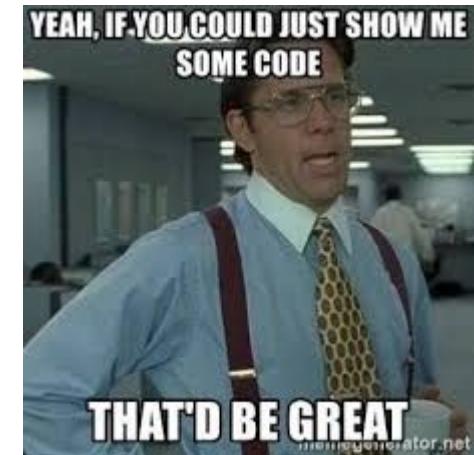
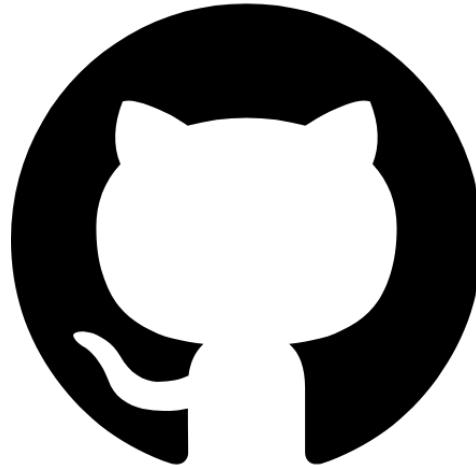
Built in spare time by @karpathy to accelerate research.  
Serving last 95810 papers from cs.[CV|CL|LG|AI|NE]/stat.ML

# First steps with ML

## 8. Do prior research on the topic! don't try to reinvent the wheel

Code implementations

- <https://github.com>



# First steps with ML

## 8. Do prior research on the topic! don't try to reinvent the wheel

### Code implementations

- <https://paperswithcode.com/>

The screenshot shows a detailed view of a research paper on the PapersWithCode platform. The title is "Reformer: The Efficient Transformer" by "google/trax". It was published on "13 Jan 2020". The paper has "1,497 stars/hour" and "0.71 stars/hour". There are two versions of the paper shown, one in English and one in Chinese. A "LANGUAGE MODELLING" button is visible below the abstract. To the right, there are green "Paper" and blue "Code" buttons.

### Browse State-of-the-Art

1673 leaderboards • 1355 tasks • 1447 datasets • 19441 papers with code

Follow on Twitter for updates

### Computer Vision



Semantic Segmentation

45 leaderboards

754 papers with code

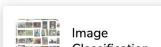


Image Classification

69 leaderboards

630 papers with code



Object Detection

54 leaderboards

529 papers with code



Image Generation

62 leaderboards

258 papers with code



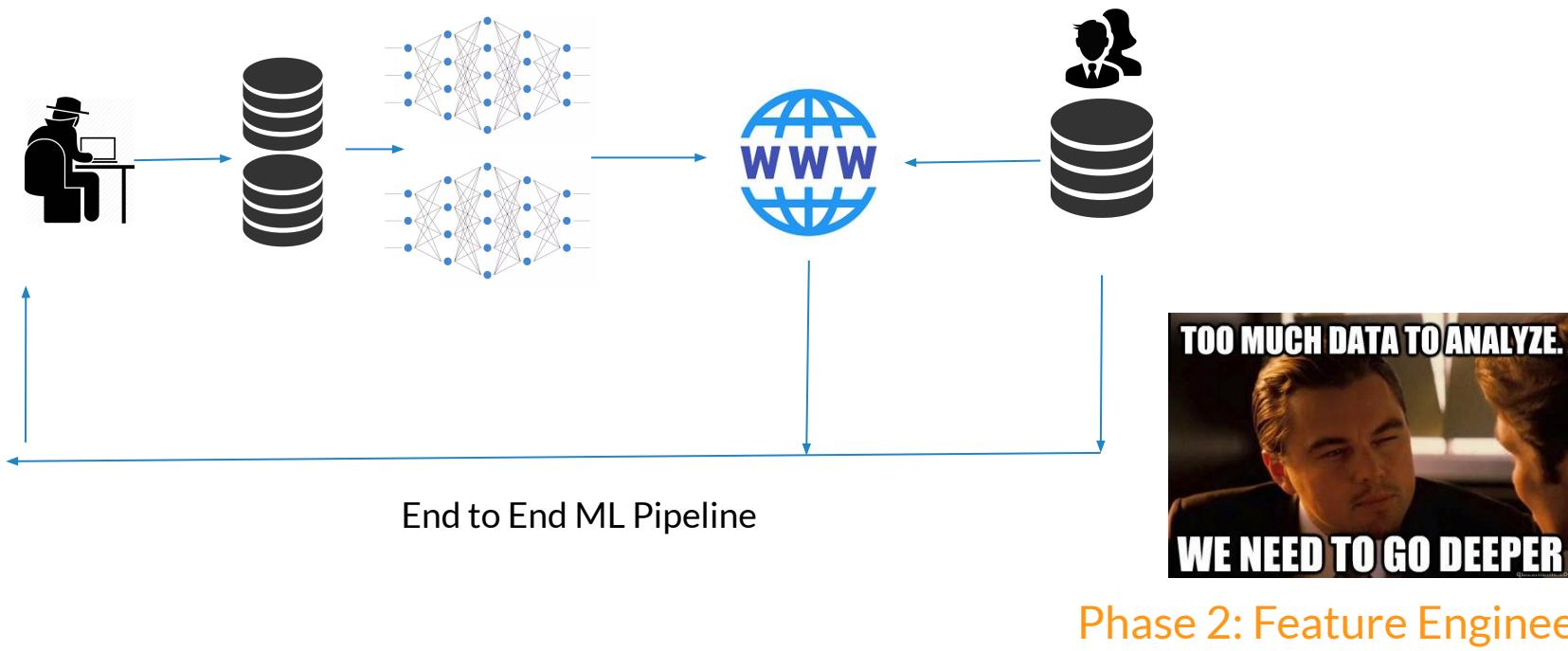
Pose Estimation

49 leaderboards

251 papers with code

See all 719 tasks

# After First Steps



# Deep Learning and Feature Engineering

## 9. Aim for the low-hanging fruit

- Pull as many features as you can and combine them in intuitive ways (if possible)
- Is there any new state of the art model, new feature transformations, new encodings you can implement?
- It is almost always a good approach to spend resources on more data collection

# Deep Learning and Feature Engineering

## 10. Combine features and modify existing features to create new features

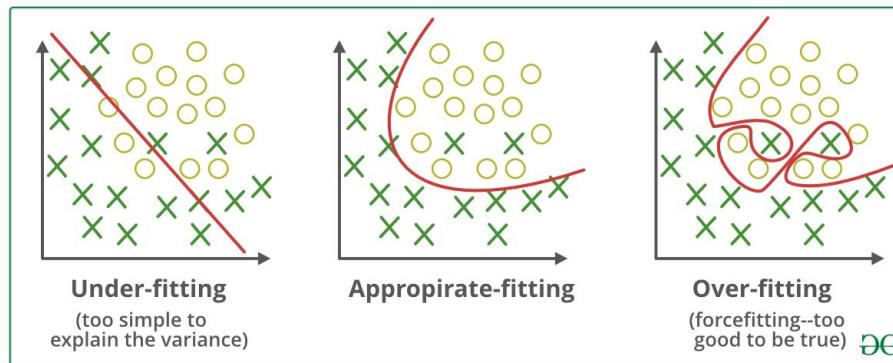
Use prior knowledge / expert knowledge or data analysis statistics

- Discretization of continuous variables
  - E.g. Age in age groups -- Feature 1: Age < 2 years old, Feature 2: age > 65 years old
- Crosses / Combine features into groups
  - E.g. Specific population group: {Male, Canadian}, {Female, Spanish, Older than 35}

# Deep Learning and Feature Engineering

## 11. Use regularization techniques

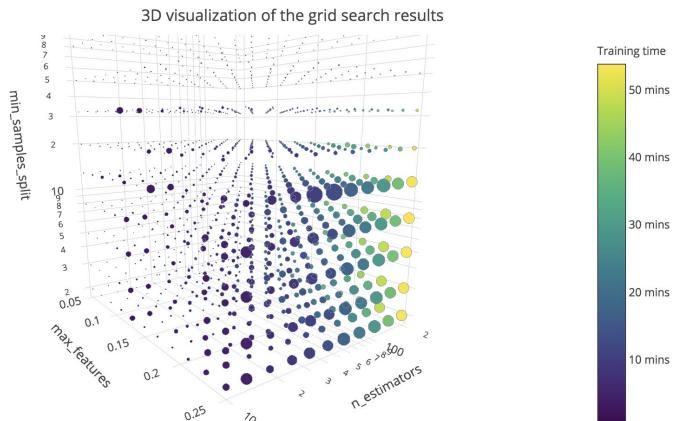
- Regularization can be a great help against overfitting
  - Dropout, Batch Normalization, L1, L2



# Deep Learning and Feature Engineering

## 12. Use hyperparameter search

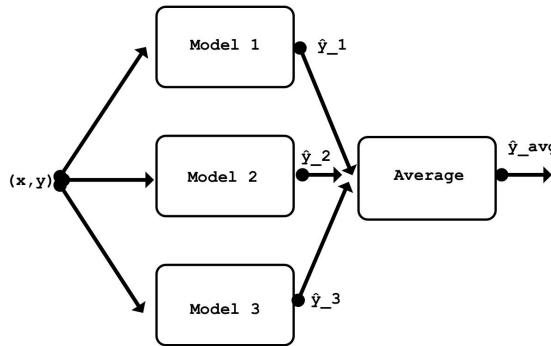
- If you have the computational resources using a search to look for the best parameters for your model can help getting improvements



# Deep Learning and Feature Engineering

## 13. Use model ensembles

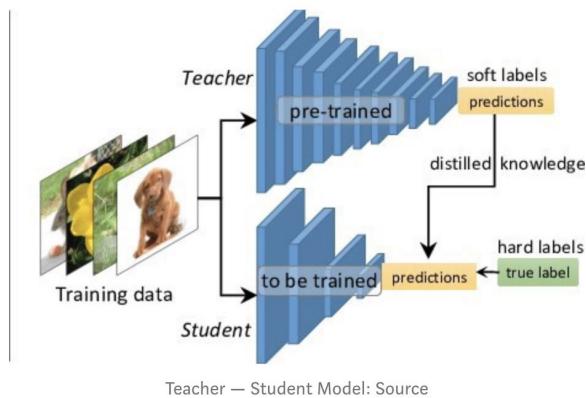
- If you have the computational resources using ensembles is always a good idea to get more accurate predictions.



# Deep Learning and Feature Engineering

## 14. Knowledge Distillation

- You trained a huge model / ensemble but want to deploy a lighter version



Teacher — Student Model: [Source](#)

[Distilling the Knowledge in a Neural Network](#) - Geoffrey Hinton, Oriol Vinyals, Jeff Dean

# Deep Learning and Feature Engineering

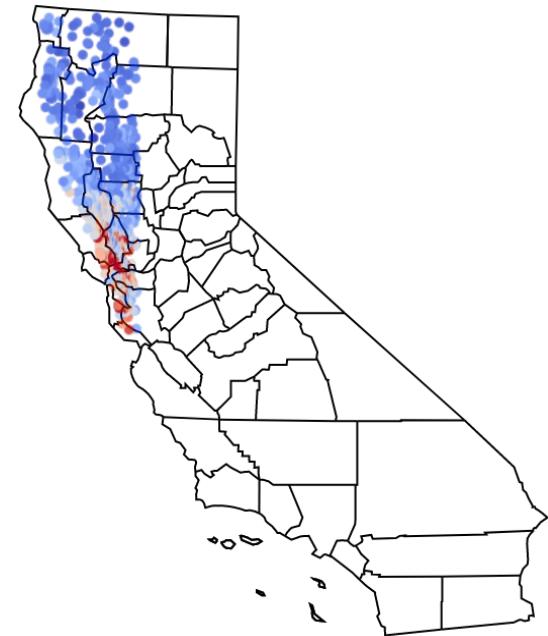
## 15. Look for patterns and analyze your data

- Perform error analysis to find:
  - Biases
    - **Reporting Bias** → Dataset does not reflect real world frequency
    - **Selection Bias** → How dataset samples are chosen does not reflect real-world distribution of data
    - **Group Attribution Bias** → Generalize from individuals to groups
    - **Implicit Bias** → Assumptions based on our own mental models

# Deep Learning and Feature Engineering

## 15. Look for patterns and analyze your data

- Perform error analysis to find:
  - Biases
  - If this unrepresentative sample were used to train a model to predict California housing prices statewide, the lack of housing data from southern portions of California would be problematic.



California State Map

# Deep Learning and Feature Engineering

## 15. Look for patterns and analyze your data

- Perform error analysis to find:
  - Biases

True Positives (TPs): 16	False Positives (FPs): 4
False Negatives (FNs): 6	True Negatives (TNs): 974

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{16}{16 + 4} = 0.800$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{16}{16 + 6} = 0.727$$

Female Patient Results

True Positives (TPs): 10	False Positives (FPs): 1
False Negatives (FNs): 1	True Negatives (TNs): 488

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{10}{10 + 1} = 0.909$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{10}{10 + 1} = 0.909$$

Male Patient Results

True Positives (TPs): 6	False Positives (FPs): 3
False Negatives (FNs): 5	True Negatives (TNs): 486

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{6}{6 + 3} = 0.667$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{6}{6 + 5} = 0.545$$

# Deep Learning and Feature Engineering

## 15. Look for patterns and analyze your data

- Perform error analysis to find:
  - Errors in your own datasets
    - Wrong data labels
    - Users are making mistakes when inputting data
    - Reviewers are making mistakes when labeling data
  - Features that are not contributing to optimize the objective function

# Deep Learning and Feature Engineering

## 16. Monitor the Train / Serving skew

- There will be always difference between training and test data distribution
- You want to be able to detect drops in metric performances
- If you train a model with data until 24 September 2019, using test data from 25 September 2019 will help predicting production behaviour

# After Feature Engineering Phase

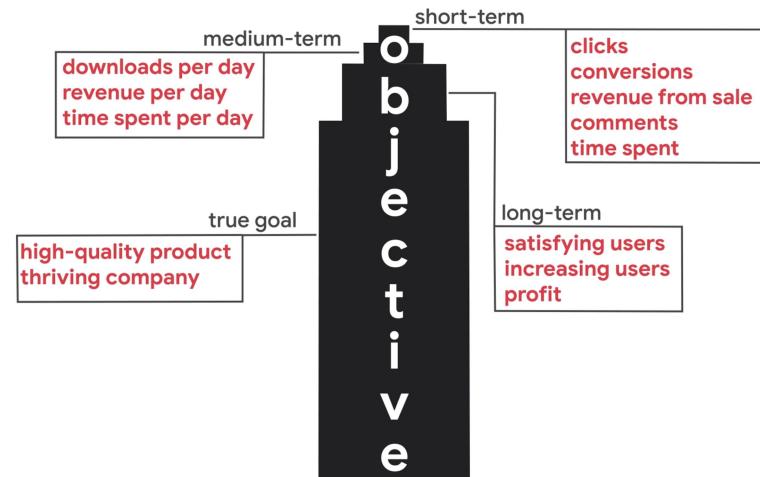
You will know you are here if

- You have a really good pipeline in place
- Gains in performance start to diminish
- You see trade-offs between metrics

# After Feature Engineering Phase

## To get to the next level

- You will need a more sophisticated machine learning system
- Redefine new goals, focus on long term objectives



# Useful Resources

- [Rules of Machine Learning: | ML Universal Guides](#)
- [Production ML](#)
- [A Recipe for Training Neural Networks](#)
- [Lessons Learned from Building Practical ML Systems](#)

# 3. Exercises

# Virtual Environment (virtualenv)

- Tool that helps to keep dependencies required by different projects separate by creating isolated python virtual environments for them.
- 1 environment / project
- Makes easy to
  - Share projects -- Dependencies installation
  - Keep track / fix libraries versions
    - Avoid errors if code in libraries change

<https://virtualenv.pypa.io/en/latest/>

# Palladium



These are the requirements that Palladium fulfills:

- Smooth transition from prototypes to machine learning models in production
- Avoid boilerplate in ML projects
- High scalability
- Avoid license costs (OSS)

## Why Palladium?

- We will see the development cycle and putting models into production scalably
- Covers common tasks in machine learning projects

Go to

[https://github.com/jsalbert/ml\\_production\\_ai\\_deep\\_dive](https://github.com/jsalbert/ml_production_ai_deep_dive)