Lecture 1: Introduction

Matthew Caldwell

COMP0088 Introduction to Machine Learning · UCL Computer Science

Contents

1.1: What’s it all about, Alfie?

COMP0088 Introduction to Machine Learning • UCL Computer Science

Ubiquity & Utility

- ML is everywhere!
- Everyone uses it (sorta kinda)
- Everyone talks about it
  - → Including a lot of nonsense
- It has made huge progress in recent years
  - → Perhaps the great success story of 21st century computing
  - → Confluence of more data, fast hardware (esp. for linalg) and algorithmic improvements
- It is culturally important as well as extremely useful
  - → Will “user engagement” kill us all?

What isn’t it?

- It’s not artificial intelligence
  - → At least, not exactly. Though they are in a relationship. (“It’s complicated”)
  - → Provocatively: it’s the bit of AI that works
- It’s not on the verge of ending civilisation in a robot singularity
- It’s not magic
- More prosaically: it isn’t always the right tool for the job
  - → Excels for well-defined questions with densely sampled data
  - → Not good at abstract reasoning — mostly does not even operate in that space
  - → Avoid the human trap of imagining an ML model understands anything

What is it?

- A bunch of techniques to make program behaviour be driven by data
  - → But, hang on… isn't program behaviour always driven by data?
- All code acts on data, and most programs have some kind of conditionality
- But that's just the specific data the program is using right now
- What a normal program does to that data has been specified by the coder
- In ML, the behaviour being applied now has been previously extracted — distilled — (learned) — from a large body of other data
- Crucially, the coder does not need to specify explicitly what to do — or even have to know how to do it

Trivial example: know your audience

- In its simplest forms this isn't much more than basic statistics
- If you know your users are 95% cat lovers, then you can make a good guess when they upload a photo of a furry animal that it's a cat
- If you know they're dog people, sell them puppy treats
- If you know they're obsessive Star Wars fans advertise Jar-Jar Binks toys
- This seems like rudimentary stuff, but suggesting customers buy products like things they already own* took Jeff Bezos to (or anyway, near) space
  - → * there may be more to Amazon's success than just this, obvs
- There's a continuum from this to more complex data-driven modelling

Not trivial at all: face recognition

- Human visual system is really good at seeing faces
  - → Even when they aren't there: pareidolia
- We're social animals, tuned by millions of years of evolution because identifying individuals — and potential individuals — is good for survival
- The mechanics of this, as for most brain functions, are pretty opaque, but we know there are multiple regions of the brain dedicated to it
- It happens automatically — unconsciously — without having to know how
- If you had to explain it you might handwave vaguely about hairlines and eyebrows and mouth shape and so on, but that's post hoc rationalisation
  - → And at a pretty high level of abstraction too

Your phone isn’t too bad at this either

- Which is really kind of surprising, because we don’t know how it’s done, and phones are products of human engineering
- Some of the ways in which this works are loosely inspired by things we’ve observed in neurophysiology — structure and activity in real brains
- Some come from people being quite clever in terms of algorithms etc
- But probably the biggest contributor is that the software in your phone has seen an awful lot of faces, and figured out an empirical recognition strategy — probably a jerry-built mish-mash of quasi-strategies — that mostly works
- Which it can’t explain either

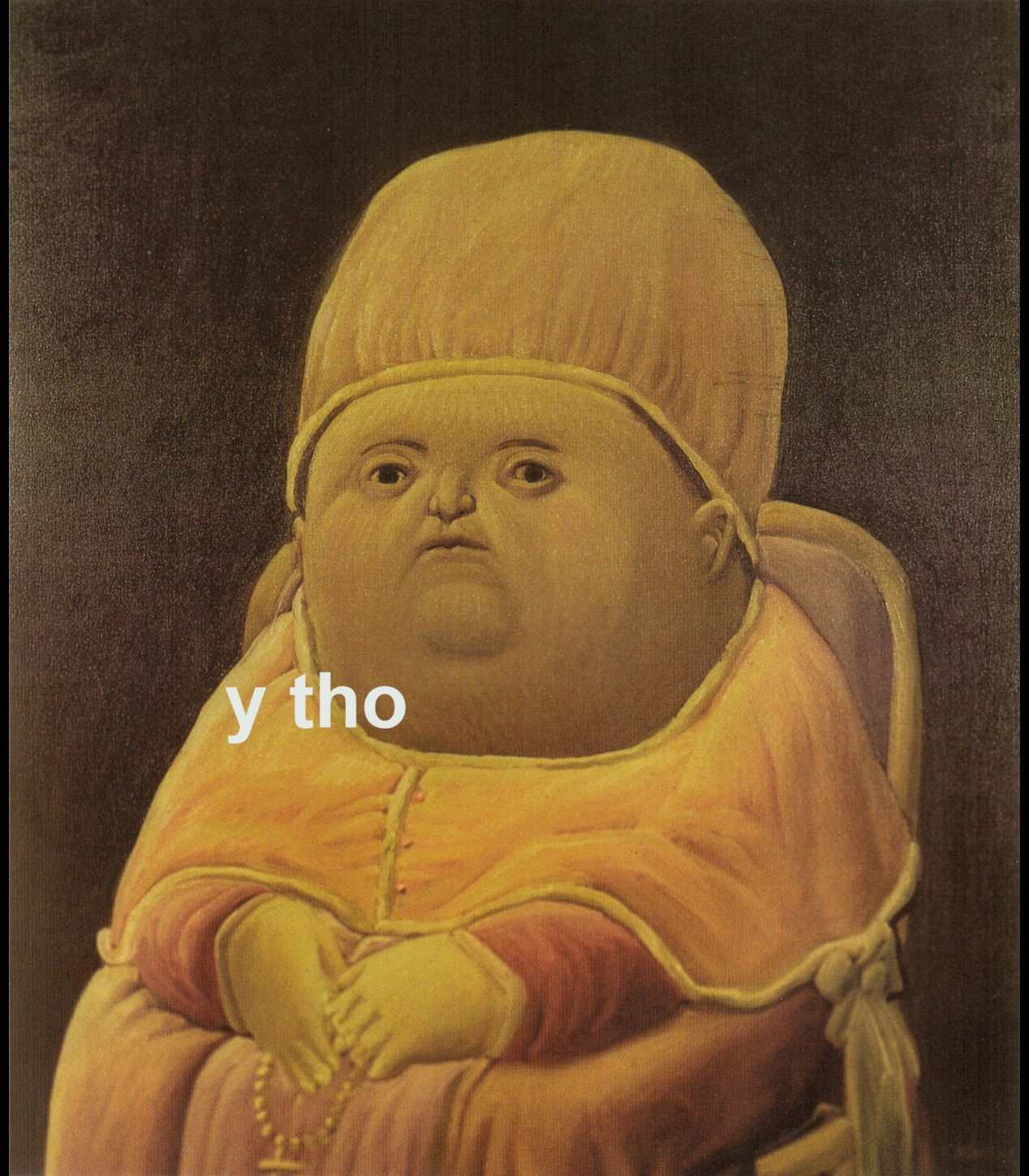
  → It’s artificial, so we can dissect it in detail, but that’s not an explanation

Learning by experience

- This is what we mean by machine learning: parsing tactics out of data
- But we don't necessarily — or usually — mean it in an ongoing way
- Human experience of learning is slow, incremental, continuing
- ML is often "one and done"
  - → In a few of the simpler cases it's not even iterative, you just calculate an answer
- There is a logic to the terminology, but it carries unhelpful connotations
  - → This is common in ML, and I'll occasionally point out egregious examples
- We'll come back to learning, I promise, but to begin with I'd like to talk instead in terms of models

1.2: Models

COMP0088 Introduction to Machine Learning · UCL Computer Science


y tho

Is “model” any more helpful?

- The word is widely used with many shades of meaning
- Not immediately clear what model trains and planes, model portfolios, molecular models, cosmological models, epidemiological models, catwalk supermodels and machine learning have in common
- But all in some way attempt to capture some features of a system or process
  - → perhaps superficially, in terms of appearance
  - → or more deeply, in terms of fundamental behaviour
- …while (explicitly or implicitly) glossing over others

Wrong to be right

- “All models are wrong, but some are useful”
  - → George E P Box
- Should not be interpreted as an admission of failure
- A significant part of the usefulness arises from the wrongness
  - → ie, the approximations and elisions that make the model viable
  - → a model including every last detail is not really a model at all, but the thing itself
  - → like the 1:1 scale maps in Carroll or Borges

Models are functions

- In the context of ML (but also throughout science and beyond) we want a model to tell us something about some system of interest
  - → ie, produce some output
- When we tell it some information about the state of that system
  - → ie, give it some input
- In other words, it's just a function:

$$\mathbf{y} = f(\mathbf{x})$$

Models are functions

$$\mathbf{y} = f(\mathbf{x})$$

- Given some potentially multi-valued input $\mathbf{x} = [x_1, x_2, \ldots, x_d]$
  - → variously known as independent variables, explanatory variables, predictors, observations, samples, measurements, features…
- Predict a potentially multi-valued output $\mathbf{y} = [y_1, y_2, \ldots, y_m]$
  - → known as the dependent variable, response variable, prediction, inference, label…
- Collectively, **x** and **y** are the model variables

Variables could be anything

- For the moment, forget about the types of data in **x** and **y**
  - → If there's one thing we know from living in the age of information it's that literally everything can be represented as numbers
  - → We will get much more specific later — there can be significant differences in applicability of different techniques to different types of data
- Then this abstracted, simplistic, seemingly trivial model of a model is general enough to encompass a wide range of classic examples from physics, biology, economics, engineering, etc

Models!

$$v = V_{max}\frac{[S]}{K_M + [S]}$$

$$E = mc^2$$

$$V_0 = V_t e^{-rt}$$

$$F = G\frac{m_1 m_2}{r^2}$$

$$V = IR$$

$$I = C_m\dot{V}_m + \bar{g}_K n^4(V_m - V_K) + \bar{g}_{Na} m^3 h(V_m - V_{Na}) + \bar{g}_l(V_m - V_l)$$

$$\dot{n} = \alpha_n(V_m)(1-n) - \beta_n(V_m)n$$

$$\dot{m} = \alpha_m(V_m)(1-m) - \beta_m(V_m)m$$

$$\dot{h} = \alpha_h(V_m)(1-h) - \beta_n(V_m)h$$

$$F = ma$$

$$d = \frac{\lambda}{2n\sin\theta}$$

$$\mathbf{F} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B})$$

$$\mathbb{E}(r_f) = r_j + \sum_i \lambda_{j,i} R_i$$

Laws?

- Some of the models on the previous page are elegant enough — and reliable enough — that we get full of ourselves and call them “laws”
  - → Physics is particularly good at finding problem spaces — astronomically large, atomically small, highly mechanistic — where the model approximations are very good
- They are still models though, which means — as George Box would tell us — they are still wrong
  - → At least in the useful sense of glossing over or omitting less relevant details in order to get to the fundamental truths of the situation

Parameters

- Consider Newton's ~~law~~ model of gravity:

$$F \propto \frac{m_1 m_2}{r^2}$$

- This is a functional form describing the relationship between variables
- But to get actual numbers out, it needs to be parameterised:

$$F = G\frac{m_1 m_2}{r^2}$$

Parameters

- The ~~universal constant~~ parameter *G* grounds the model, turning it from an abstract notion of how the entities interact into something quantitative, which can make concrete predictions

$$F = G\frac{m_1 m_2}{r^2}$$

- *G* isn't just made up at random, it is fitted from data
  - → It's the value that makes the model predictions agree with observable reality
- The same is true for model parameters in general

Parameters

- Update our model model to include parameters $\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_k]$

$$\mathbf{y} = f(\mathbf{x}; \boldsymbol{\theta})$$

$$\mathbf{y} = f_{\boldsymbol{\theta}}(\mathbf{x})$$

  - → The semicolon distinguishes parameters from variables
  - → Subscript equivalent but tries to emphasise how integral params are to model

- For linear models we use $\mathbf{w}$ instead of $\boldsymbol{\theta}$: $\mathbf{w} = [w_1, w_2, \ldots, w_d]$

Bad parameters! Bad!

- Traditionally parameters have been viewed with suspicion
- Necessary but problematic, a source of errors and/or excess flexibility
- “With four parameters I can fit an elephant, and with five I can make him wiggle his trunk”
  - → John Von Neumann
- Having lots of parameters is seen as cheating because it makes it possible to fit data more closely without having a better model
  - → In the intuitive sense of being more connected to reality or more explanatory
- This is known as overfitting, and we’ll be talking about it a lot

Model specification

- Consider (again) Newton's model of gravity:

$$F \propto \frac{m_1 m_2}{r^2}$$

- Where does a model like this come from?
- Informed by observations, of course
- But then much pondering and beard-stroking ensues
- Until eventually insight strikes and everything becomes magically clear

Genius isn’t a game plan

- This has been the model of scientific enquiry since antiquity
  - → It’s the classic “Eureka!” moment of Archimedes in his bath
- And obviously it has achieved great things
- But it’s predicated on an unreliable resource: genius
- And even then it tends to fall down when problems are high dimensional, inelegant, or too complex to hold in one’s mind at once
- Which is true for an awful lot of problems

The mesoscale is complicated

- Activities at the scale of living things — resource management and logistics, social dynamics and diseases, media and politics, cities full of people with desires and opinions and emotions and agency — are a messy business
- Behaviour is often driven by large numbers of factors, many of which may not even be measurable
- Factors can interact in non-obvious and unfathomable ways
- These situations tend not to be susceptible to elegant explanatory models combining a few simple variables in a neat functional form
- But if insight may be thin on the ground, we do sometimes have data

Explicit & implicit models

- Which — finally! — brings us back to machine learning
- Traditional models in physics, physiology, economics, etc are explicit
- Behaviour is directly expressed through the functional form
- ML (at least to some extent) gives up on explicit specification
- Instead, as much behaviour as possible is outsourced to the parameters

- “We don’t know how to do this. Can we fit our way out of that hole?”

1.3: Fitting

COMP0088 Introduction to Machine Learning · UCL Computer Science

What does fitting even mean?

- Trivially: find the best values for our model parameters
- But find how?
- And what do you mean “best”?

Loss functions

- Loss functions are really the heart of machine learning
- They provide a measure of how good (or rather how bad) our model is
- Importantly: a measure we can (somehow) optimise
  - → What this usually (not always) means in practice is: differentiable
- We'll often (not always) denote them *L*, e.g.:

$$L(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}, f)$$

  - → Some people use *J* instead of *L*. The arguments and general form will also vary a lot according to our application

Sorry for your loss

- Almost always, the loss is just a proxy for some other, non-optimisable task
  - → A means to an end
- Nobody actually cares about the loss per se
  - → We care about classification accuracy, not categorical cross-entropy
  - → We care about image quality, not mean squared error
  - → We care about audio fidelity, not spectr— well, you get the idea
- In most cases, loss values have meaning only in relation to one another
  - → Edge values (e.g. 0, 1, ∞) may be meaningful, 5.62371692613 really isn't

Loss can be difficult

- A loss function will often have two or more components, trading off different qualities we're looking for in the solution
  - → We'll start seeing examples of this next week
- It is possible to devise very complicated losses, taking into account many aspects of the model, in order to capture the nuances of a particular task
  - → We'll see examples towards the end of term
- But quite a lot of problems can be solved with just a handful of loss metrics
- The familiar Euclidean distance (or L_2 norm) can be a reasonable starting point, if only for its intuitive geometric interpretation. For suitable problems, it literally quantifies how far away you are from the right answer.

Optimisation

- Fine, we have a loss. Where do we go from here?
- We want to find the parameter set — call it $\boldsymbol{\theta}^\star$ — that gives the lowest loss

  → Superscript star is used to denote “best”
- There’s a handy bit of notation for this, argmin:

$$\boldsymbol{\theta}^\star = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\ L(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}, f)$$

this means we’re optimising solely with respect to θ — we can’t change **x**, **y** or *f*

Optimisation

- Fine, we have a loss. Where do we go from here?
- We want to find the parameter set — call it $\boldsymbol{\theta}^\star$ — that gives the lowest loss
  - → Superscript star is used to denote “best”
- There’s a handy bit of notation for this, argmin:

$$\boldsymbol{\theta}^\star = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\ L(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}, f)$$

  - → The corresponding argmax means what you think it does
- This specifies what we want to do, but it doesn’t tell us how to do it
  - → In the general case, the task may be difficult or impossible

But how?

- There are (at least) five options:

  → Find an analytic solution by solving for root(s) of derivative

  → Find a numeric solution by following the (sub)gradient

  → Perform an exhaustive search of available values

  → Perform a systematic non-exhaustive search

  → Try values at random

- We'll take these in order (but note that higher isn't necessarily better)

Analytic solution

- Recall from calculus that function derivatives are zero at extrema
- Denote the loss derivative with respect to the parameters as $\nabla_{\boldsymbol{\theta}} L$
  - → More on this next week!
- It is occasionally possible to solve the following equation:

$$\nabla_{\boldsymbol{\theta}} L = \mathbf{0}$$

- In which case, you're laughing!
  - → There are other conditions for a global minimum, but let's assume they're met
- Don't count on doing this: it almost never works

Numerical solution

- If the loss is (more or less) differentiable, we can find the local gradient
  - → Can also estimate that numerically, though can get expensive
- Unless we're at a minimum, destination must be further down*
- So keep taking small steps down the gradient until we get there
- This (with various tweaks) is the workhorse method of ML & DL

* At least in a global sense. The local picture may be different.

Convexity

- Gradient descent uses local information, so can get stuck in local minima
  - → The only way is up!
- Convex functions have no local minima, only a single global minimum
  - → Concave means basically the same but with a single global maximum
- This makes them much more reliably optimisable
  - → Convex optimisation is a whole important field in its own right
  - → There are mature tools for optimising various problem classes with constraints
  - → Won't go into detail here, but important for implementation of some models
- We'll mention when losses are convex — but some very much aren't

Brute force

- For a finite number of options, it may be possible to try them all
- This can be possible even for continuous parameters
- Obviously can't try all $\boldsymbol{\theta} \in \mathbb{R}^k$
- But often only a countable subset of values actually matter
  - → Notably: those that occur in your training data
- Not always applicable, may be prohibitively expensive, and kind of icky
  - → But guaranteed to give you an optimum when you can do it

Systematic search

- If exhaustive search is too costly, maybe we can divide up the search space into more manageable pieces
- Grid search
  - → Test parameter values at regular intervals across their ranges
- Recursive grid search
  - → Repeatedly resample best region at higher density
- Greedy algorithms
  - → Avoid combinatorial explosion by optimising for one thing at a time
- Not guaranteed to find the right answer in general, but may for some tasks

Random search

- Not as crazy as it sounds
- Can achieve more efficient sampling of multidimensional spaces
  - → Dimensions are sampled independently
  - → Especially relevant when some parameters are unimportant
- Particularly useful for ensemble models
  - → ie, when aggregating multiple predictions

- Worst case performance is, of course, very bad

Hyperparameters

- Fitting a model involves choices
  - → What model, what loss, what optimisation algorithm?
  - → How many neighbours or clusters or neurons or trees, how fast to learn?
- These are not parameters of the model, they are parameters of the process
- Known as hyperparameters
- Can be the difference between a model working or not
- Crucially, they are not learned — we have to set them some other way
- Having lots of hyperparameters is often disadvantageous for that reason

Take home

- Fitting means finding best model parameters
- Quantified by a loss function
  - → A proxy for the real task
- Found by an optimisation algorithm

- Most of ML boils down to this. Of course, the devil is in the details.

1.4: Types of Data, Types of Learning

COMP0088 Introduction to Machine Learning · UCL Computer Science

Training data

- By far the most crucial distinction between types of data in ML is:

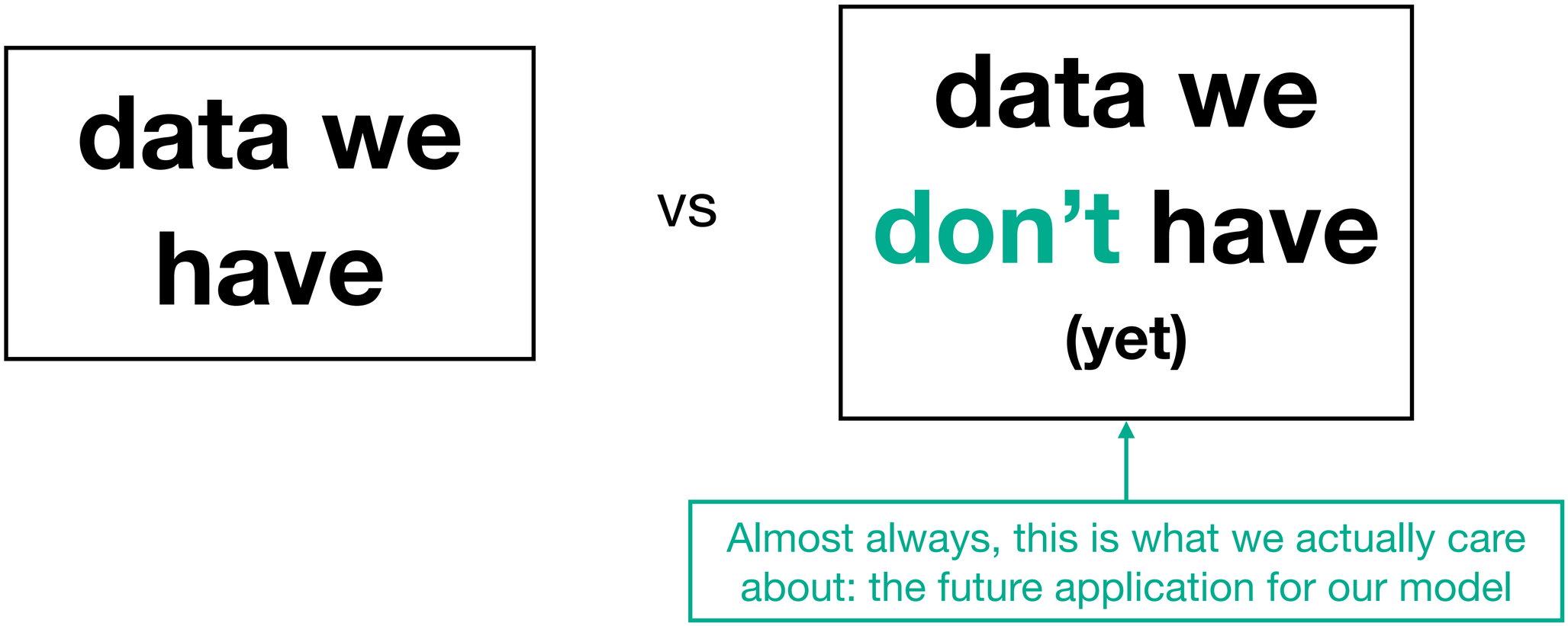
data we have

vs

data we don't have (yet)

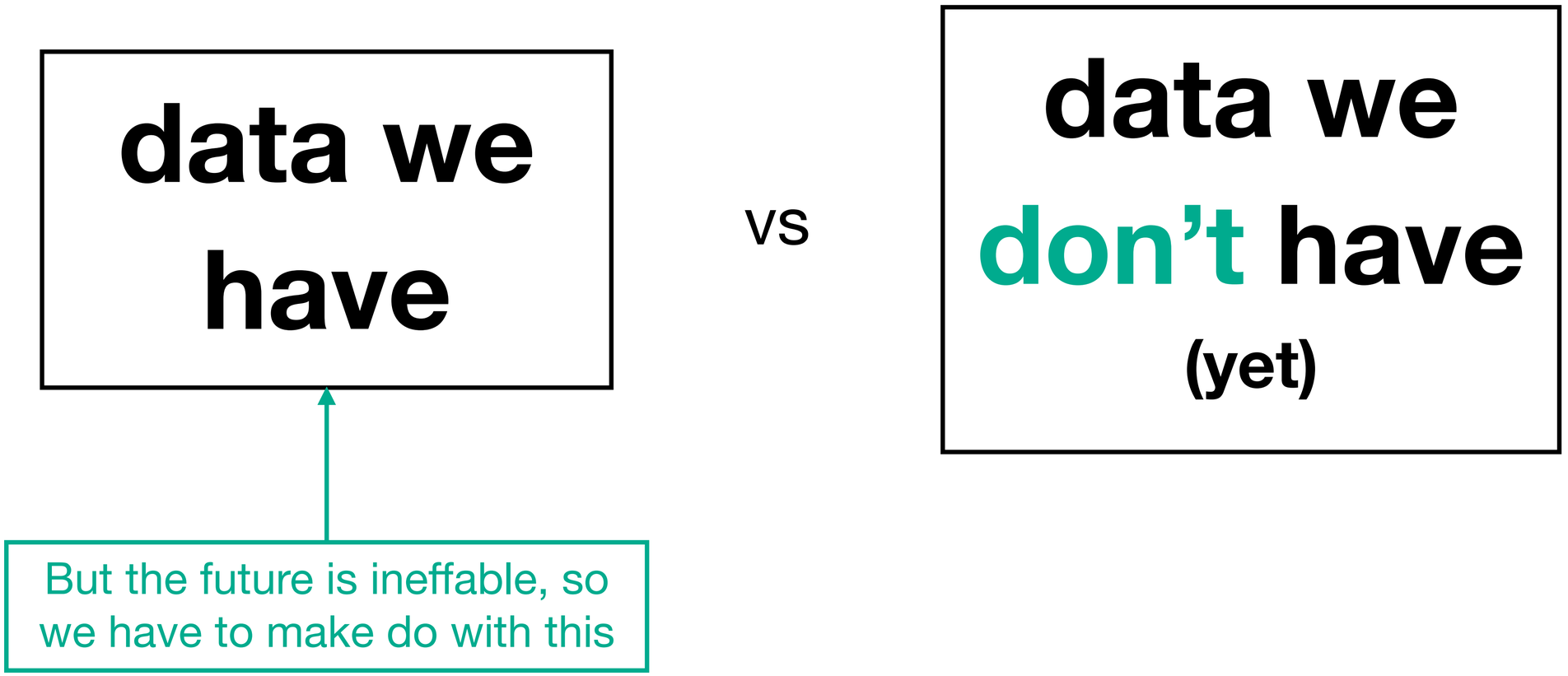
Training data

- By far the most crucial distinction between types of data in ML is:



Training data

- By far the most crucial distinction between types of data in ML is:



Training data

- We have to assume the available data is representative
  - → ie, unseen future data will be “like” what we have
  - → if this is not so then we’re in trouble anyway
- We’re going to use this data for training
  - → But if we use all of it for that, we’ll have no way of testing our model
  - → Testing on data you trained on **is not a test**: you already know the answers
  - → Not all ML models have the capacity to memorise everything they’ve seen in training, but some do — and they’ll always memorise something
  - → A model must generalise to unseen data or it is useless
  - → That’s the O-word again: overfitting

Train/test split

- So: we take some of the data and put it aside for testing
  - → Seriously. Put it aside. Don’t use it to check your code is working. Don’t use it to tune your hyperparameters.
  - → Don’t even look at it. Hide it. In a locked safe. At the bottom of the Marianas Trench
- Obviously there is a trade-off here: you need enough test data to get meaningful results, but you also need data to train on
  - → In practice something like 80/20 is common, depending on the size of your dataset

Train/test/validation split

- We often want to run on unseen data for things like hyperparameter tuning
- Don’t use test data for this
  - → This is not a test!
- We typically split out *another* chunk of the data for this, the validation set
  - → Perhaps taking us to 70/10/20 or whatever
- If there’s really not enough data for that, we might use cross-validation
  - → Eg: for each training sample
    - remove it from training set
    - train model on all the rest
    - test on the left-out sample
    - average over all the results

Data types

- Model inputs **x** and outputs **y** could have all manner of types
  - → Numbers, words, images, sounds, video, perhaps combinations of all these
- For most models, most types will eventually be represented as numbers
  - → We'll discuss some possible ways of converting between types shortly
- Probably the most important distinction is continuous vs discrete:
  - → Continuous: real numbers, or well approximated as such — eg, pixel values are usually discrete (0-255) but almost always treated as continuous
  - → Discrete: separate classes (usually represented by integers): truth values, presence or absence, categories, letters, words, concepts, arbitrary symbols — Disney princesses
  - → There may be grey areas for ordinal data: ratings out of 10, clothing sizes, age bands

Regression vs classification

- We tend to care most about the output type, because that affects our choice of loss function
- If **y** is continuous, the problem is called regression, and a distance-based loss makes sense
  - → A typical regression loss is mean squared error (MSE)
- If **y** is discrete, the problem is called classification, and an information-based loss is more appropriate
  - → A typical classification loss is cross-entropy
- Both kinds of losses will often be conceptualised probabilistically, but there's a corresponding continuous/discrete dichotomy there too

Categorical to numeric

- A few models can deal with categorical variables, but most expect numbers
- So one of the key conversions is categorical to numeric
- It's possible to assign numeric values directly to different classes, but such encodings will be arbitrary signifiers
  - → The order and scale of the numbers won't mean anything
- Instead, we commonly split out categorical values into distinct feature dimensions or indicator variables
  - → Instead of a single "genre" dimension with values "comedy", "action", "horror" etc, we have separate dimensions for "comedy", "action", "horror" etc

Indicators & one-hot encoding

- We often use notation like this for indicators

$$\mathbf{1}(x = \text{“horror”})$$

- Which is shorthand for

$$\begin{cases} 1 & \text{if } x = \text{“horror”} \\ 0 & otherwise \end{cases}$$

- Computationally, the categorical value is represented as a one-hot vector, which has value 1 in the true class dimension and 0 everywhere else

Embeddings

- One-hot vectors are a simple example of an embedding: a representation of one data space within another
- Embeddings are very common in NLP to represent words
- One-hot representation of words will be large (as many dimensions as words in your vocabulary) and will impose an unrealistic degree of independence
  - → Words are related by meaning and patterns of usage
- A word embedding maps the discrete words to points in a lower dimensional continuous space, attempting to capture some semantic properties in the distances and directions between those points
- Such embeddings are typically learned rather than engineered

Numeric to categorical

- Thresholding (for binary case)
  - → If number is greater than the threshold it's in one class, otherwise the other
- Binning (generalises thresholds to multiple classes)
  - → Single number maps to multiple classes by which bounds it falls in
- High score
  - → Separate numeric scores for each class, highest one wins
- Voting
  - → Multiple values predicted (eg as above) and class with most votes wins

Collections

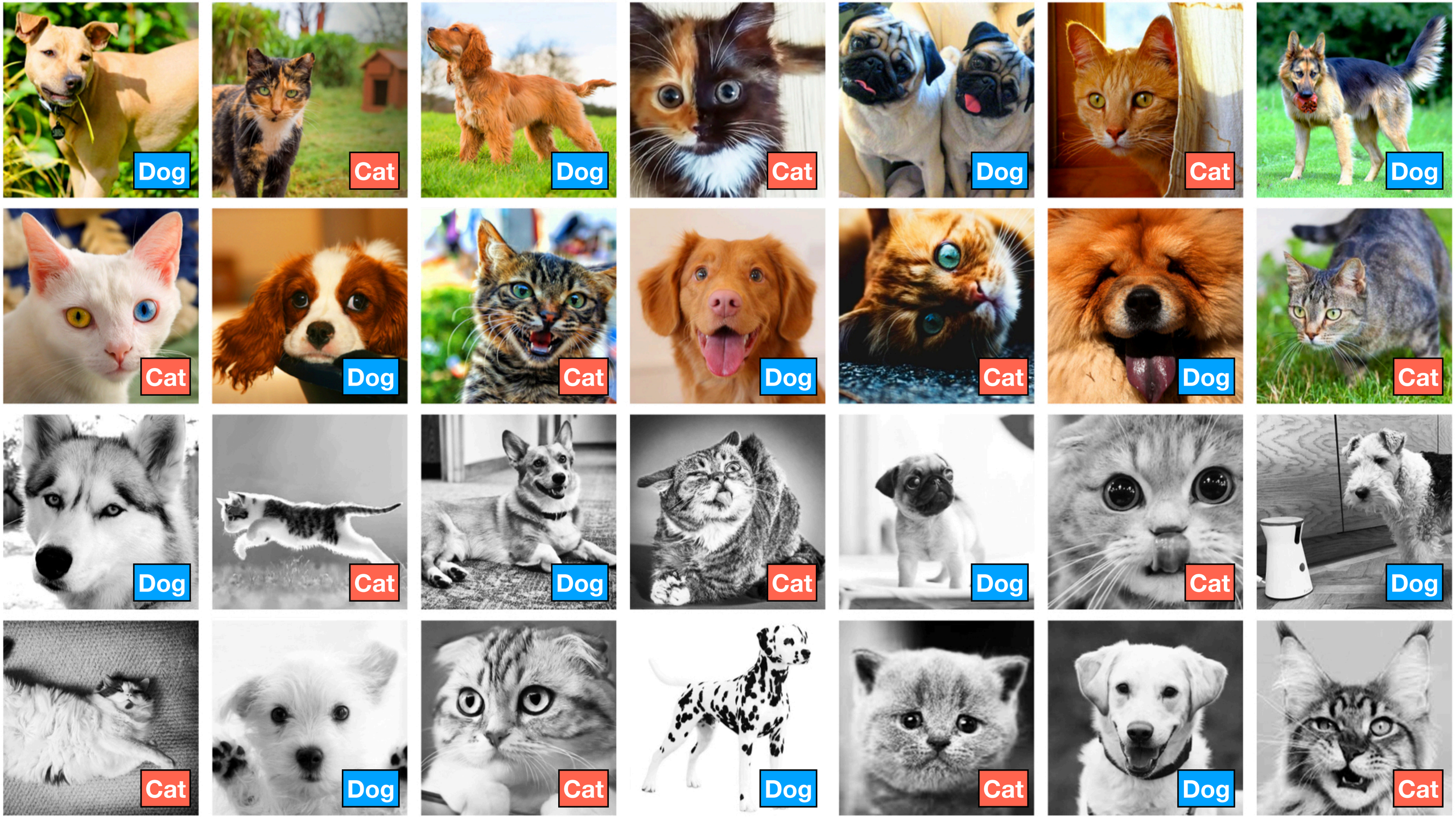
- Might be a whole bunch of data of different kinds
- Mostly, just bundle them together as dimensions of an even larger collective space
- Useful to distinguish two special cases:
  - → Images (and more generally digital signals) having a large number of elements of the same type with a consistent spatio-temporal structure (regular sampling, local connectivity, translation equivariance, etc)
  - → Sequences, where the ordering of elements is important (sentences, musical notes, medical histories or movie plotlines)

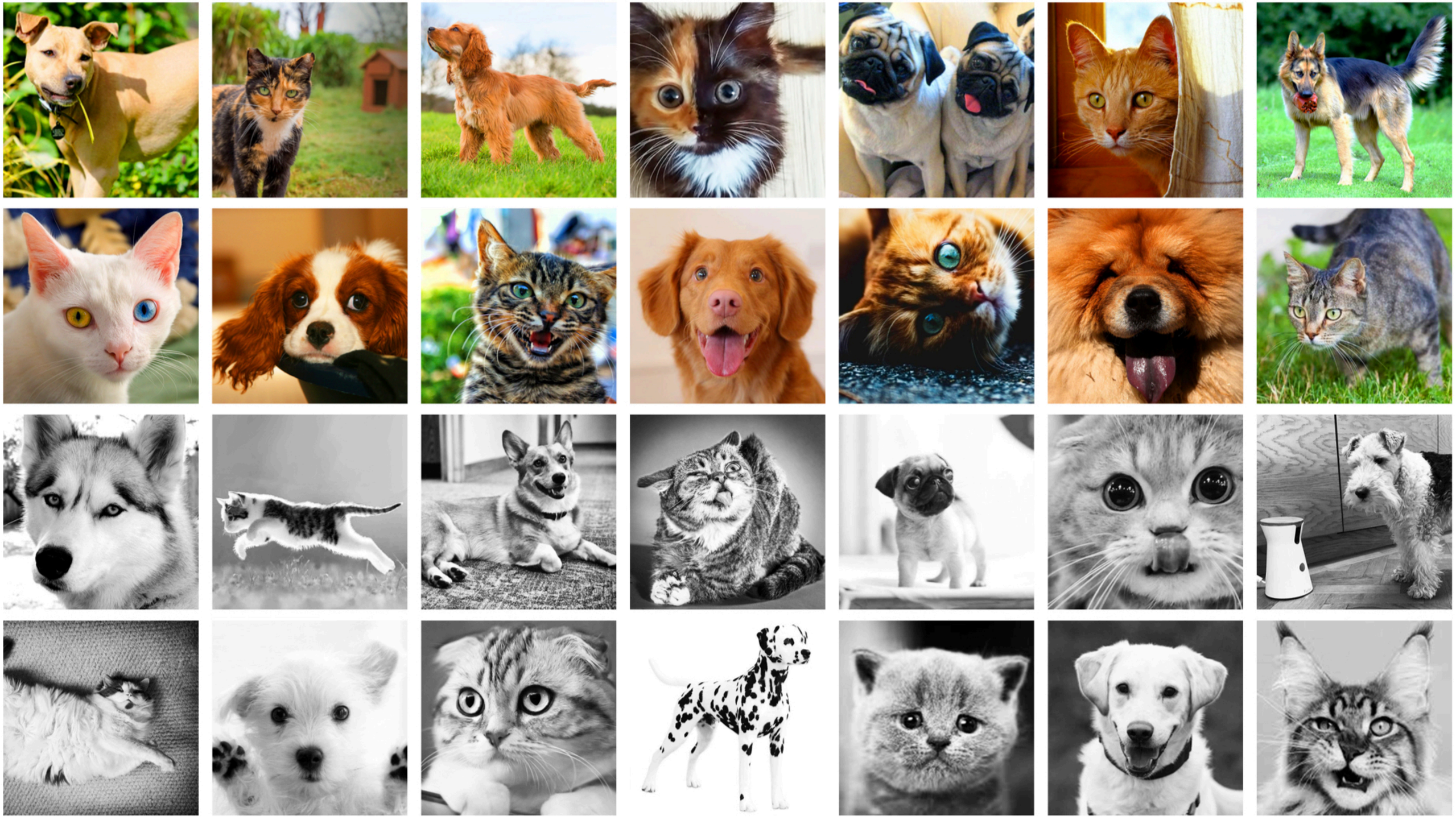
Preprocessing

- Loss functions and learning algorithms are often sensitive to differences in feature scale, and also to asymmetry, so we'll often preprocess data to make it more consistent:
  - → Centering: mean subtraction so feature dimensions have expected value 0
  - → Normalisation: rescaling values into some target range (eg [0,1])
  - → Standardisation: special case of normalisation so features have mean 0 variance 1
  - → Whitening: linear transform to have identity covariance (remove correlations)
- Log and other domain-specific transforms (polar, FFT, …)
- Everything you do to training data you have to do at test time too!
  - → Must remember standardisation parameters etc

Supervised vs unsupervised learning

- We've so far tended to assume, implicitly, that we know what **y** is — what kind of thing in general, and also the specific instances for training
- If we do, it is known as supervised learning — supervision in this case meaning there is something to tell you whether you're right or wrong
- This is the most straightforward form of ML: learning by example
- But ground truth labels have to come from somewhere — from humans, ultimately — and they tend to be expensive
- Often we want or need to learn from data without labels: ie unsupervised
- Seems like a tall order: how can you learn if you don't know what's true?


Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat



Unsupervised learning

- In all forms of ML, we're looking for regularities, patterns, structure
- Labels make that very specific, tell us exactly what we're looking for
- In their absence, we need some other constraints on the problem: what kind of structures? how many?
- Probably the two most common kinds of unsupervised learning are clustering and dimensionality reduction — and these actually correspond pretty closely to the supervised tasks of classification and regression:
  - → clustering: invent class labels
  - → dimensionality reduction: invent continuous outputs
- These inventions must have logic to them, of course

Weakly-, semi-, self-supervised

- There are a number of other learning modes that hover somewhere in between supervised and unsupervised, trying to find ways to mitigate the expense or impossibility of getting detailed and accurate labels:
  - → Weakly-supervised learning: unreliable or unspecific labels, eg multiple instance learning for digital pathology
  - → Semi-supervised learning: small labelled set plus more unlabelled, eg text classification
  - → Self-supervised learning: learn a proxy task for which training labels can be automatically generated, eg image inpainting

Reinforcement learning

- Very different learning paradigm: no training set!
- In some ways more intuitive, resembling exploratory human learning
- Quasi-autonomous agents explore by trial and error, maximising rewards
- Used for games playing and robotics
- Dynamic environment changed by actions, so sequence important
- Typically needs to run in simulation for very large number of repetitions

1.5: Pitfalls & Practicalities

COMP0088 Introduction to Machine Learning · UCL Computer Science

Errors, uncertainty and noise

- I quoted George Box's maxim earlier with a positive spin, but of course models are often wrong in ways that aren't useful at all
- Models learned from data are only as good as the data
- And only as good as the model
- And only as good as the learning process

- All of which are subject to errors and uncertainty
  - → Whether due to random fluctuation or poor choices
- So there are lots of ways ML can go wrong

1 + 1 ☞ 2

- It may be that the process being modelled is just not well behaved
- Or the data isn’t representative
- Or you’ve posed the question badly
- Or the model you’ve chosen to fit cannot capture the real patterns
- Or maybe there are no real patterns to capture, and the process being modelled isn’t a process at all

Failure modes

- There are a couple of ways this can play out
- Your learning algorithm may not produce an answer at all
  - → Often described as “failing to converge”
  - → This is obviously disappointing, but hey
- Much more perniciously, it may come up with a spurious answer
  - → Which looks superficially plausible or even persuasive
  - → It may even play into and confirm what you expected to see
- It is extremely important to be on guard against this, to be critical, to be wary of confirmation bias

ML doesn’t care about your problem

- The learning algorithm doesn’t even know what your problem is
- All it knows about is data, parameters and loss
- And it will do anything that helps to get the loss down
- The loss is the proxy, but maybe it’s not a good proxy

- ML cheats. If you give it the opportunity — if there’s some systematic or accidental cheat code for the task you pose — it will cheat

Honesty is the best policy

- There are things you can do to help mitigate this, which we'll discuss
- But first and foremost, be aware of this risk and honest about it
  - → Check and triple check your results
  - → Be especially suspicious if you got the answer you wanted
  - → If it looks too good to be true, maybe it is
  - → Try to falsify
  - → Look for other causes and rule them out
  - → Find alternative explanations and disprove them

Overfitting and underfitting

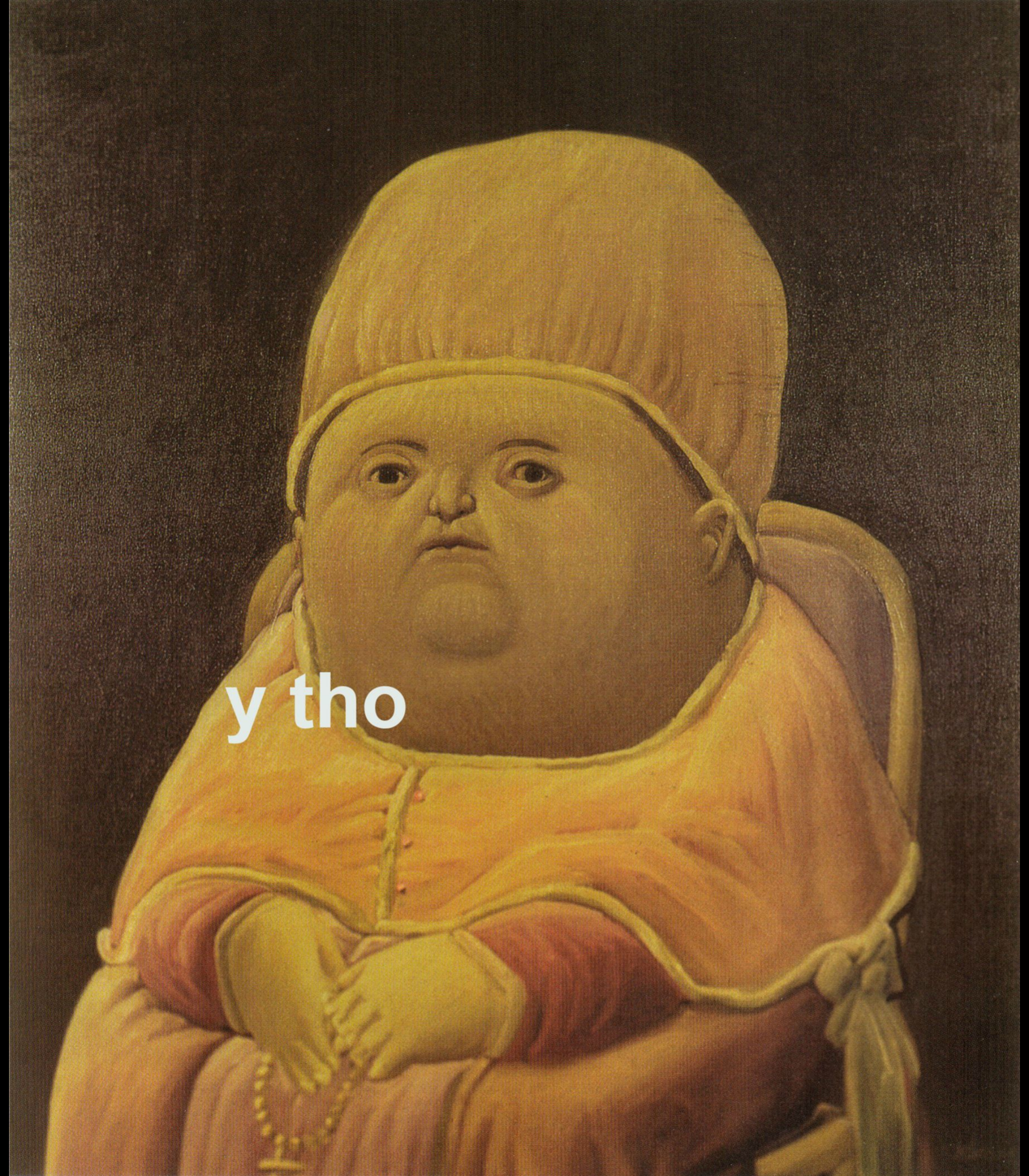
- Overfitting is tuning your model too closely to the training data, to the point where it doesn't generalise
- A model with sufficient capacity — like Von Neumann's elephant — will eventually learn to fit the training data exactly
- You can often — not always — see this in the validation and test results
  - → Be aware that choices you make on the basis of validation — for hyperparameter tuning or model selection — are themselves a kind of fitting, even when the ostensible learning algorithm never sees them. You can overfit those too!
- Overfitting is sometimes used a bit loosely as a catch-all term for generalisation failure, even if the ultimate cause is something else

Overfitting and underfitting

- Underfitting is when your model doesn’t even capture the behaviour of the training set
- This may be because the model has too little capacity
- Or because you didn’t give it enough opportunity (time, compute, data) to learn properly

Regularisation

- Attempts to mitigate overfitting are known as regularisation
- Almost always this means: trade some accuracy for “nicer” behaviour
- There are many forms — niceness isn’t just one thing — we’ll look at several in detail over weeks ahead


y tho

Bias and variance

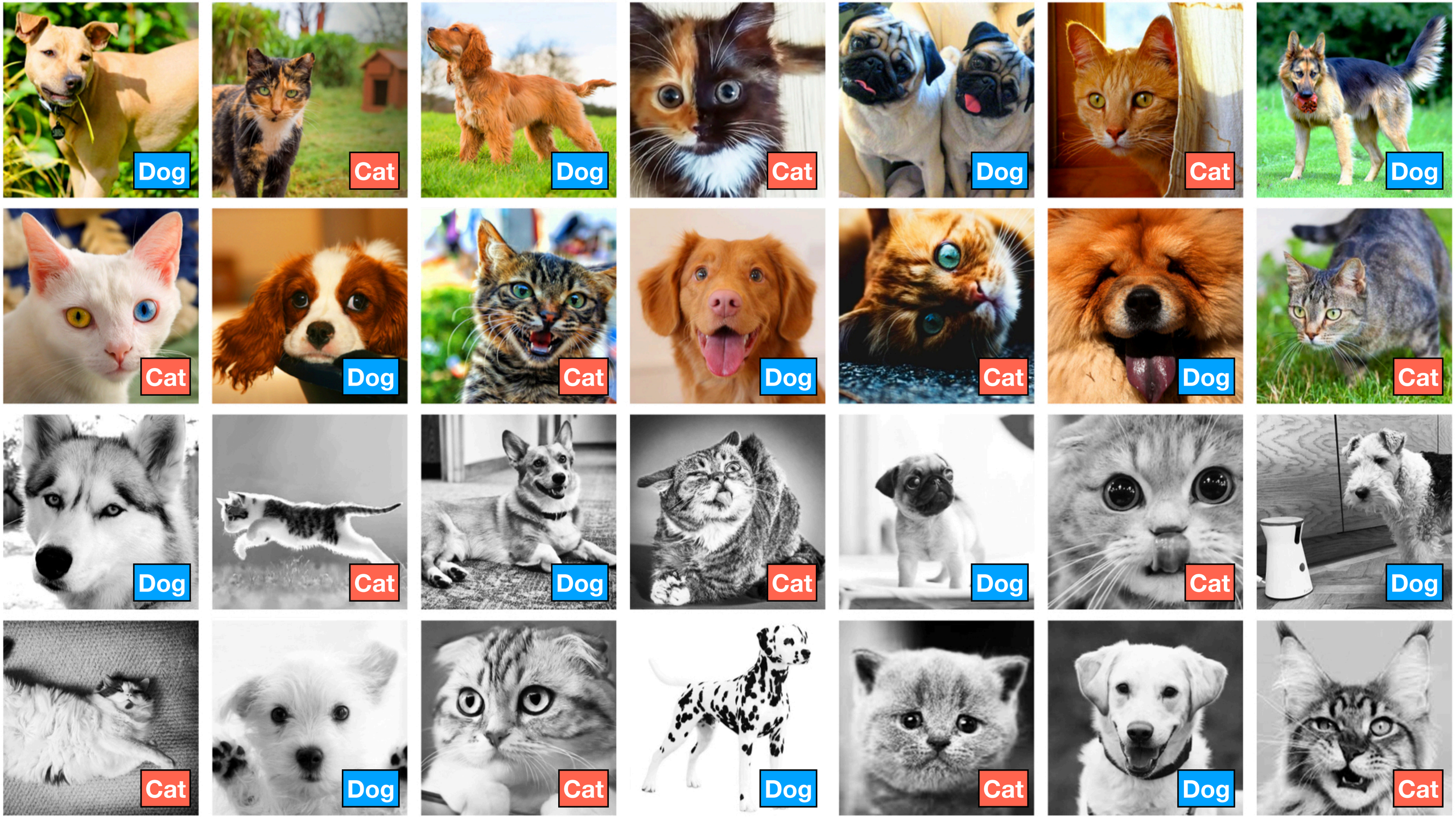
- When it comes to unhelpful ML terminology, the word “bias” is a leading contender
- It has about 8 million subtly-different meanings depending on context, nearly all of which are not the one most people think of first
  - → The usages make sense — there’s a rationale linking them — but they’re not intuitive
- Arguably the least intuitive of all is used to characterise patterns of fitting mismatch for different model classes: the bias-variance tradeoff
- Bias is the baseline tendency of the model to misfit due to its assumptions
- Variance is the degree of misfit attributable to its sensitivity to training data

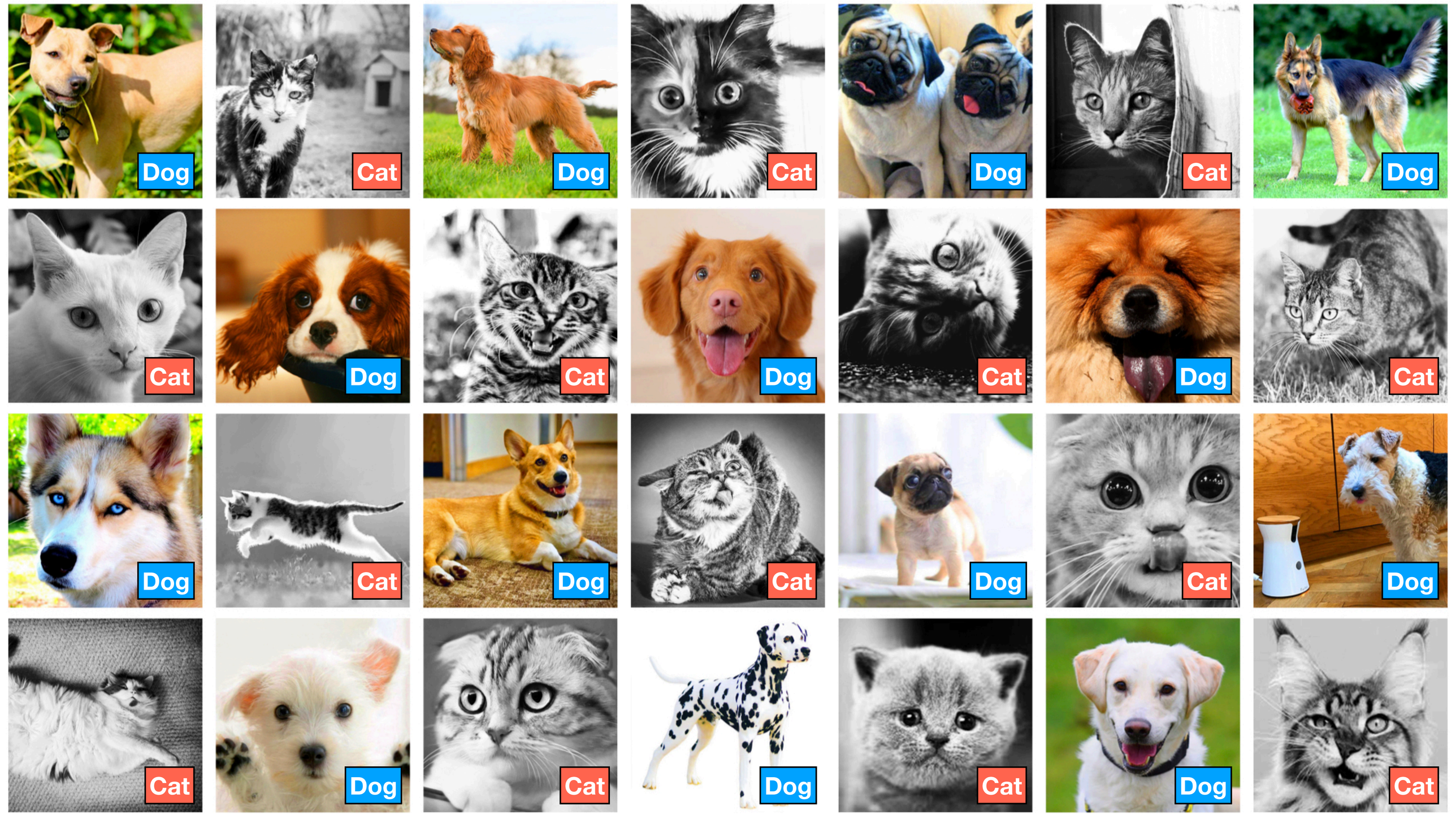
Bias and variance

- A high bias model will tend to converge to a reasonably similar (but systematically wrong) answer despite random variations in training set
  - → Imposes stronger assumptions
  - → May be prone to underfitting
- A high variance model may converge to very different answers (but possibly closer to the correct answer on average) in response to minor data variations
  - → Imposes weaker assumptions
  - → May be prone to overfitting
- Ideally we'd like both to be low, but usually not possible
- Regularisation often trades increased bias for reduced variance

Bias and unfairness

- Mostly when real people talk about bias, what they mean is: being treated unfairly, prejudged or traduced, a systematic skew against them
  - → Or possibly in their favour, but that's more when talking about others
- This kind of bias is rife in ML
- The most common and obvious cause (not the only one) is biased data
- That is: the training data is in some way unrepresentative of the true population
- The model will learn this misrepresentation and transfer it to future cases


Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat


Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat
Dog
Cat

Answering the wrong question

- Examples abound (some probably apocryphal) of ML systems learning the wrong thing because the data contained cues the modeller didn't notice
  - → Is that a plane or does it just have a blue background?
  - → Is that a horse or does it have a green background?
  - → Is it a Russian tank or a sunny day? (or — depending on who's telling this anecdote — a cloudy day, or a low-res satellite image, or an over-exposed roll of film?)
  - → Am I a natural born criminal or this just a government ID photo?
  - → Do my cheekbones give away my sexuality or is it the lighting of my selfie?
  - → Am I hot or white?

REUTERS World Business Markets Politics TV

BUSINESS NEWS OCTOBER 10, 2018 / 4:12 AM / 4 MONTHS AGO

Amazon scraps secret AI recruiting tool that showed bias against women

DemocracyPost • Opinion

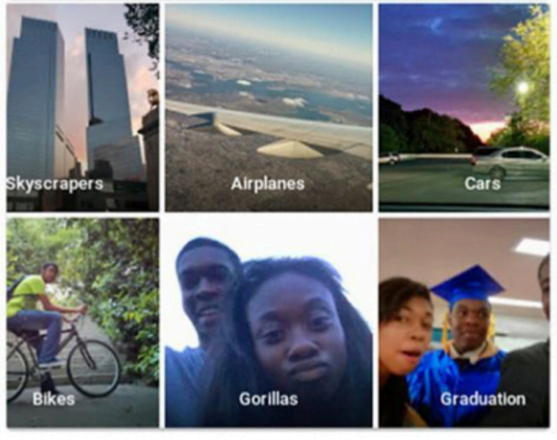
Why your AI might be racist

The Telegraph

Technology Intelligence

IBM launches software to detect racist and sexist AI

Google Photos, y'all [redacted] up. My friend's not a gorilla.



Artificial intelligence is 'shockingly' racist and sexist

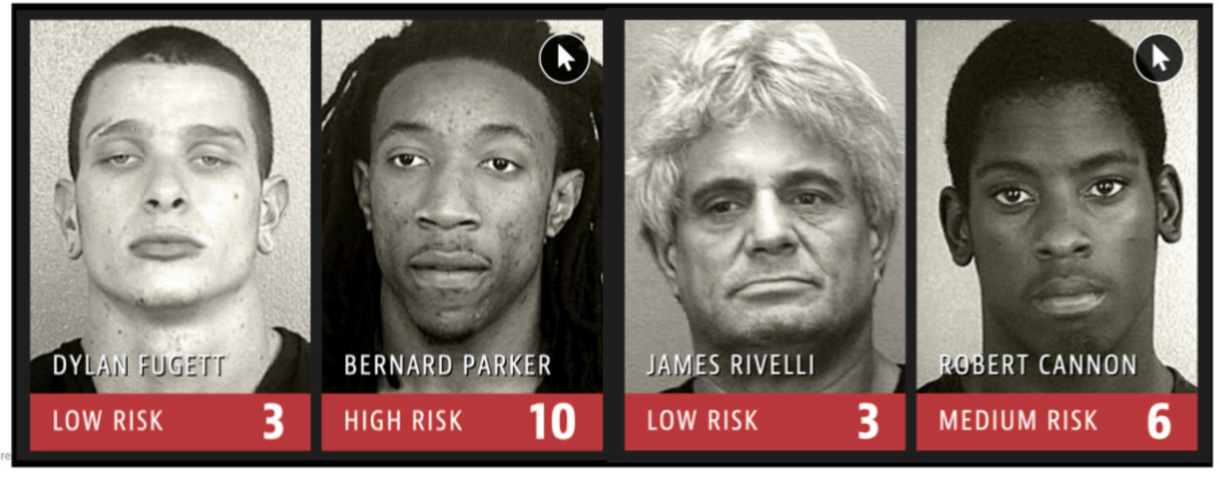
18 Nov, 2018 9:07am

3 minutes to re

FaceApp apologises for 'racist' filter that lightens users' skintone

Rise of the racist robots – how AI is learning all our worst impulses

There is a saying in computer science: garbage in, garbage out. When we feed machines data that reflects our prejudices, they mimic them – from antisemitic chatbots to racially biased software. Does a horrifying future await people forced to live at the mercy of algorithms?



nature
International journal of science

COMMENT · 18 JULY 2018

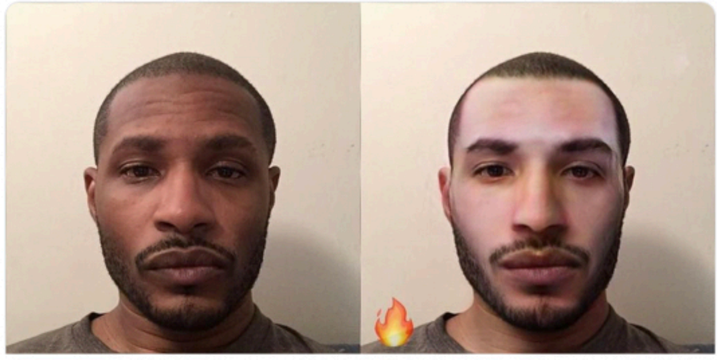
AI can be sexist and racist — it's time to make it fair

Computer scientists must identify sources of bias, de-bias training data and develop artificial-intelligence algorithms that are robust to skews in the data, argue James Zou and Londa Schiebinger.

khary @ ECCC K-1
@kharyrandolph

Follow

So this app is apparently racist as hell. But at least I'm sassy. #faceapp ift.tt/2pvtFG4



Forbes Billionaires Innovation Leadership Money Consumer Industry

EDITOR'S PICK | 21,577 views | Feb 26, 2018, 12:26pm

Racist, Sexist AI Could Be A Bigger Problem Than Lost Jobs

Perpetuating inequality

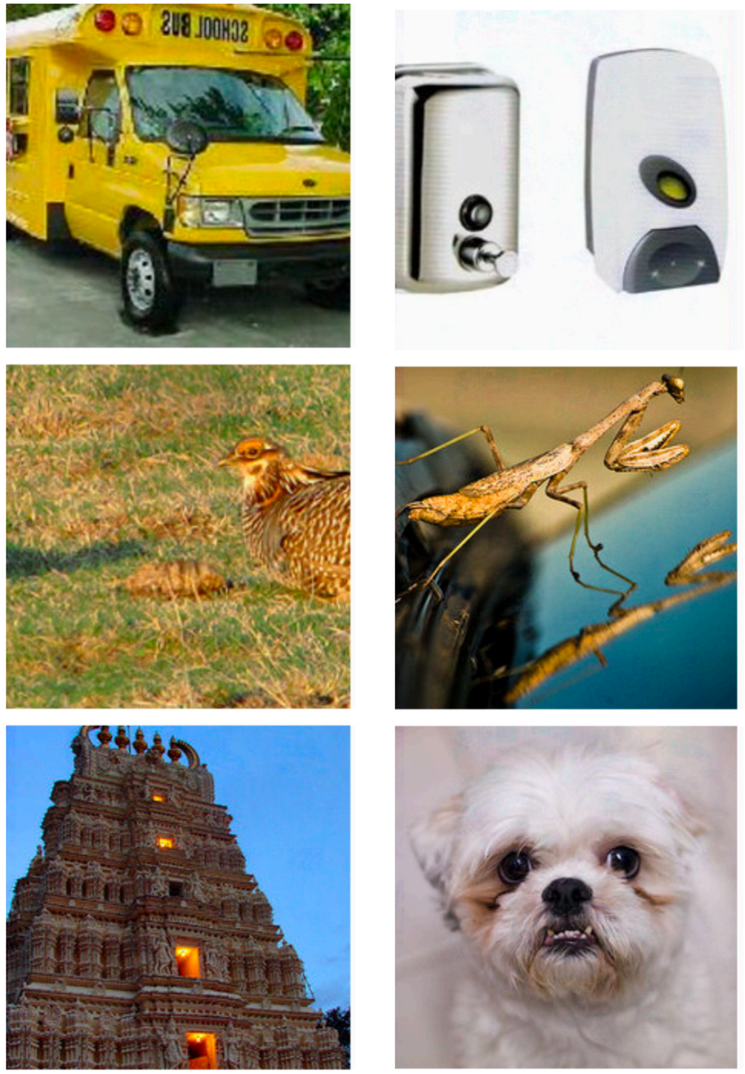
- Optimistically, most people don’t set out to collect racist training data or build a sexist model
- But they do it anyway, for two reasons:
    1. There’s a lot of racism and sexism out there, a lot of structural inequality that percolates deep into the data sources we harvest, so it’s bound to infect whatever we collect if we’re not careful
    2. They’re not careful
- There are certainly complex philosophical arguments you could make about descriptive vs prescriptive data collection. You might not want to change the world. But if you don’t at least think about the inherent biases in your data, you’re likely to find them biting you in unexpected ways.

Black boxes

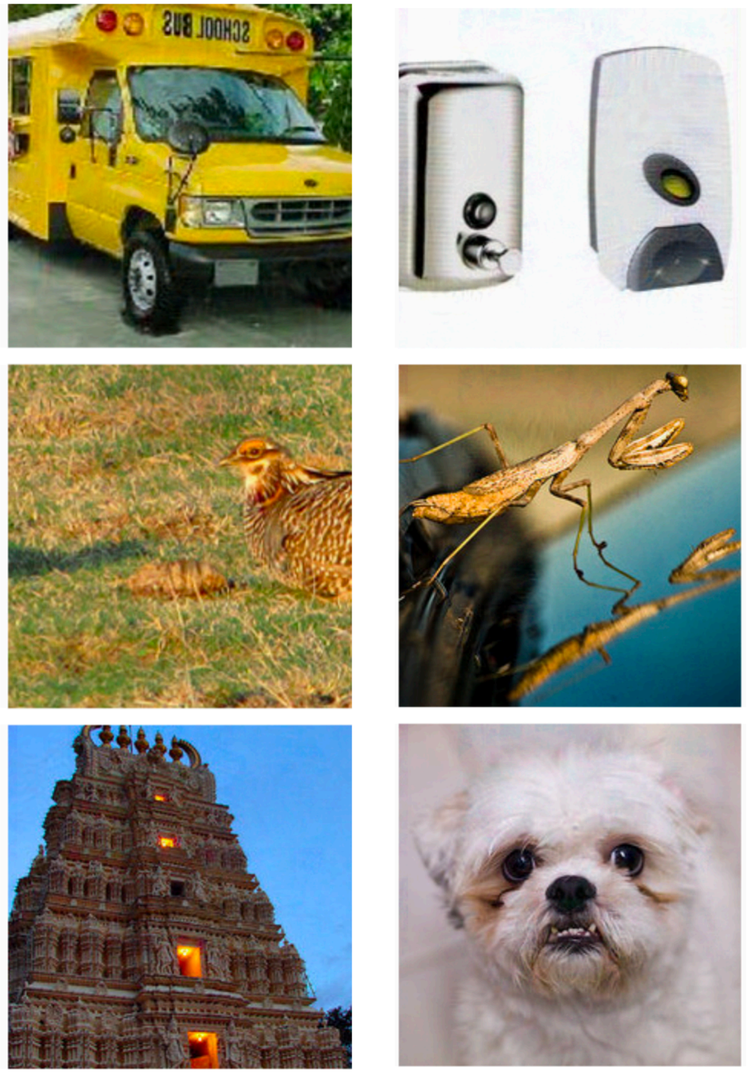
- Because they learn behaviour from data rather than having it set out in explicit instructions, ML models can be hard to interpret and understand
  - → Not always: basic linear models are often interpretable, as are decision trees
- So they often lack explanatory power
  - → Sometimes just empirically working is enough
  - → But we'd also like to gain insight into the tasks they perform
- It also makes them difficult or impossible to verify
  - → How do we know they'll do the right thing?
  - → How can we tell what vulnerabilities they might have?

Black boxes

- We've already seen cases where they don't do the right thing because the training data poorly represented the population at test time
- It turns out that if you probe the decision processes of complex ML models in detail, there will often — perhaps always — be very fine-grained vulnerabilities to data outside the training distribution & model assumptions
- These can be exploited by carefully crafted adversarial examples
- The surprising thing is how subtle these vulnerabilities can be — input tweaks that are imperceptible to humans may derail the model completely
- Though they don't have to be subtle…



Correctly classified



Ostriches

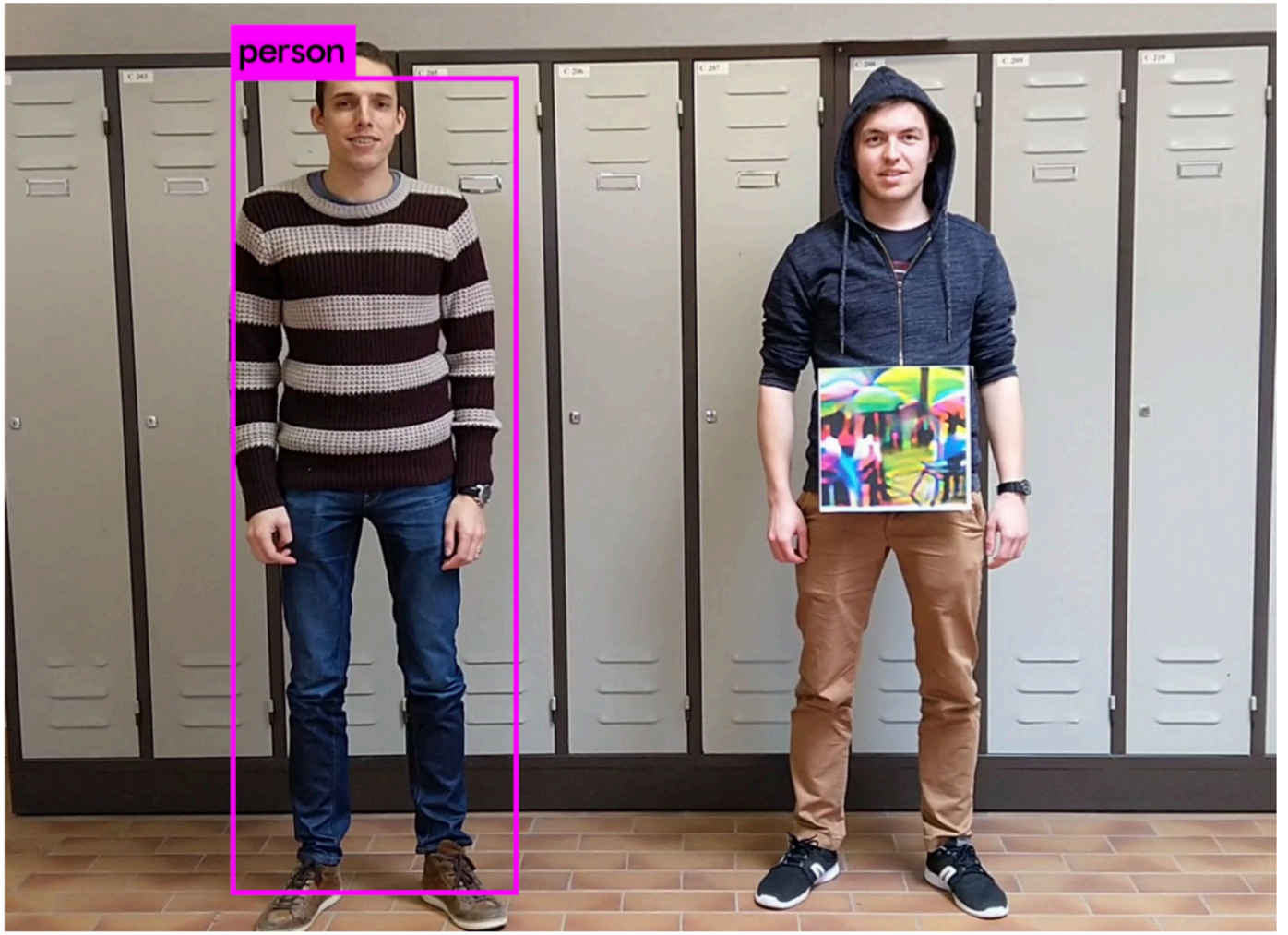


Figure 1: We create an adversarial patch that is successfully able to hide persons from a person detector. Left: The person without a patch is successfully detected. Right: The person holding the patch is ignored.

Curses of dimensionality

- We'll often use low-dimensional examples because they're easier to understand and visualise and allow us to boil a problem down to its essence
  - → Humans experience — and have an intuitive understanding of — just 3 spatial dimensions, and tend to be really bad at visualising, or in any visceral way comprehending, higher dimensional spaces
- But ML problems are frequently very high dimensional
  - → 10 or 100 dimensions is a low dimensional problem
  - → Standard ultra-basic image datasets for ML are 784-dimensional
  - → Front facing phone cameras from a decade ago had a resolution of 1280x960 px
    - ▸ At a conservative (if naive) estimate, the space of selfies would be 3.6 million-D

- This is a problem partly because it’s bewildering
- And partly because it requires a lot of compute
- But also because it turns out that in important ways, high dimensional spaces are not like the low dimensional ones we’re used to dealing with
- Low-dimensional analogies can be misleading, our spatial intuitions wrong

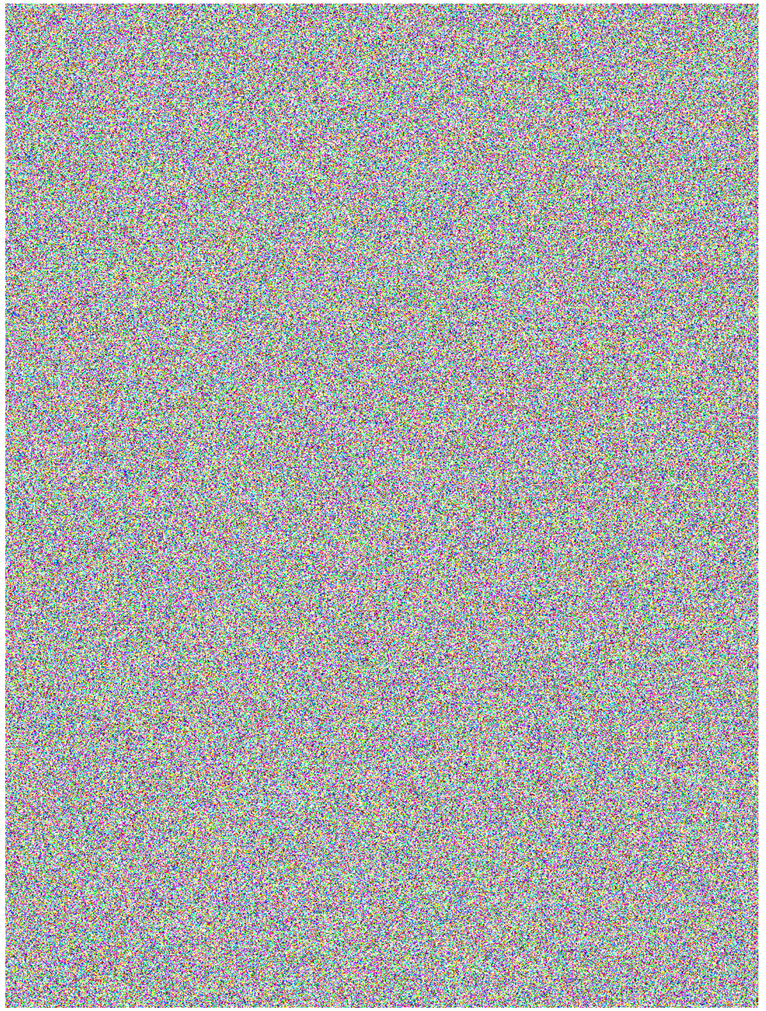
Space is big

- “Space is big. Really big. You might think it’s a long way down the street to the chemist, but that’s just peanuts to space. Listen…”
  - → Douglas Adams
- Trivially, high dimensional spaces are bigger — exponentially bigger — than low dimensional ones
  - → Eg: adjacent corners of unit d-cube are 1 unit apart, but opposite corners are $\sqrt{d}$
    - ▸ Diagonals increase without bound
- Which means their hypervolumes are very unevenly distributed with distance
  - → Hypersphere of radius ½ touches all faces, but volume proportional to $½^d$ tends to zero
  - → Hypervolume concentrates in corners (and more generally: far away)

- Make a hyperobject slightly bigger, and the hypervolume massively increases
  - → The amount of hypervolume that is “far away” dominates all the rest
- Nearly all space is concentrated in a thin shell at maximum distance
- If you randomly sample points across a high dimensional space, they all wind up roughly the same distance apart
- So distance metrics can get less discriminative

Random stuff isn't interesting

- Good news: we rarely care about randomly distributed data
- Consider the 3.6-million-D selfie space
- To a very high order approximation, all of the points in that space look like this:
  - → Very dense in "information" terms, but to human eyes entirely uninformative
- Images we're interested in have vastly fewer degrees of freedom
  - → Occupy a lower dimensional manifold



Manifolds are our friends

- Human problems typically have many fewer dimensions than our representations make it seem
- They have semantic or causal structure that makes solutions possible at all
  - → Otherwise, the brain probably wouldn't be able to deal with them either
- So the distance thing is not *quite* as catastrophic as it sounds
- But it's not negligible — there's always some randomness
- And even the "low" dimensional manifolds will often be pretty high