NUS National University of Singapore | Institute for Functional Intelligent Materials

MLHEP 2023

Interpretability

Nikita Kazeev & 

INFN National Institute for Nuclear Physics

What is interpretability?

Interpretability is the degree to which a human can understand the cause of a decision.

Miller, Tim. "Explanation in artificial intelligence: Insights from the social sciences." arXiv Preprint arXiv:1706.07269. (2017).

Interpretability is the degree to which a human can consistently predict the model's result.

Kim, Been, Rajiv Khanna, and Oluwasanmi O. Koyejo. "Examples are not enough, learn to criticize! Criticism for interpretability." Advances in Neural Information Processing Systems (2016).

An interpretable machine learning model is a model that is easy to understand and explain to humans, such as domain experts or stakeholders. The model's output and reasoning can be easily interpreted, visualized, and communicated in a way that is transparent and accessible to non-experts. This is especially important in domains where accountability and fairness are critical, such as healthcare, finance, and justice. Interpretability can be achieved through various techniques such as decision trees, rule-based models, linear models, and feature importance analysis.

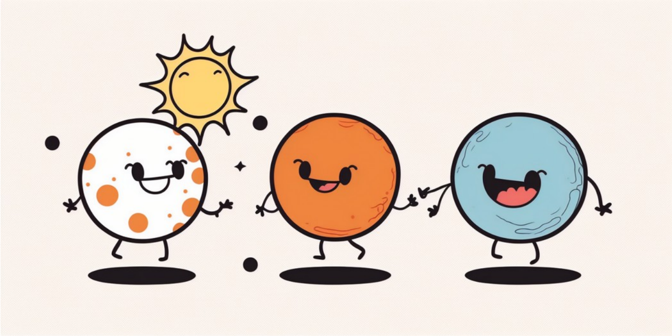
ChatGPT

Contrast

Law of gravity

$$\vec{F}_{21} = G \frac{m_1 m_2}{|\vec{r}_{21}|^2} \vec{r}_{21}$$

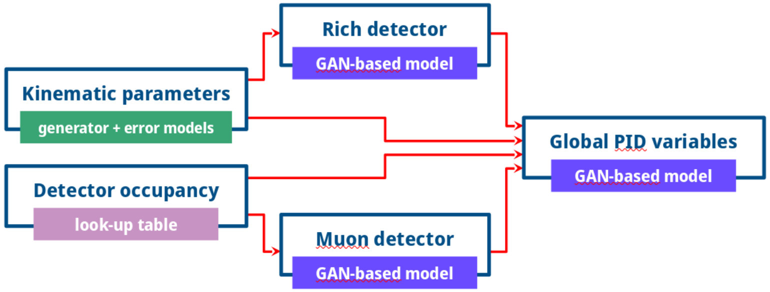
Computer program solving three-body problem



LHC event simulation

GEANT4
A SIMULATION TOOLKIT

Neural network LHC detector simulation



My takes

- Interpretable
  - Parameters are defined independently of the particular system
  - In principle, can be evaluated by a human
  - Has a well-defined scope
- Non-interpretable
  - Parameters are fully specific to the system, no way adjust them for a new system aside from retraining
  - Can’t possibly be evaluated by a human
  - No way to tell whether it works aside from testing

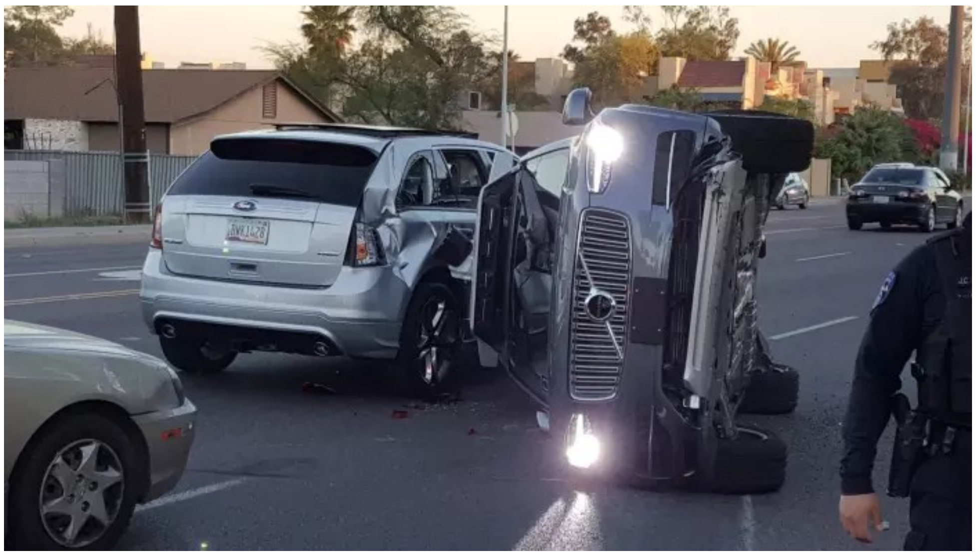
Why one wants interpretability?

- Build upon the extracted knowledge
- Guide model development
- Verify correctness

Inductive vs deductive reasoning

- Usual physical models are inductive: start with assumptions, build a complex system, ergo Geant4 and Pythia
  - They are not necessary ab initio
  - Theoretically, easy to trust: if the assumptions hold, the result is correct
  - In practice, validation fails with complexity, e.g. CERN MC
- ML models are deductive: start with data, generalize
  - Like human intuition
  - Really neat move from manual analysis to ML at CERN, cuts are essentially decision trees
  - Performance guarantees only on similarly-distributed data
    - Almost never the case in practice

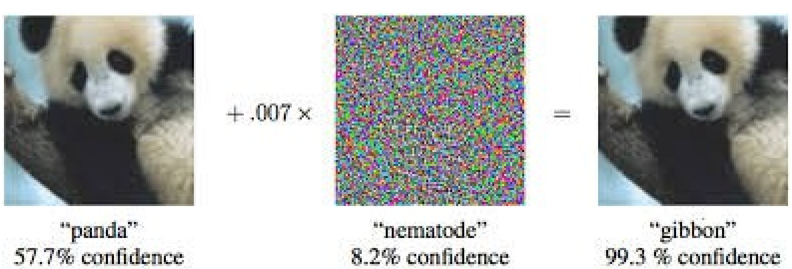
ML mistakes have a cost



Uber self-driving car crashes dur
US tests

3. Robot injured a child

A so-called "crime fighting robot," created by the platform
into a child in a Silicon Valley mall in July, injuring the 16-



Chinese billionaire's face identified as jaywalker

Traffic police in major Chinese cities are using AI to address jaywalking. They deploy smart cameras using facial recognition techniques at
intersections to detect and identify jaywalkers, whose partially obscured

The question of trust

How can I explain my model’s prediction?
Why did it make this decision/mistake?
What features does it rely on?

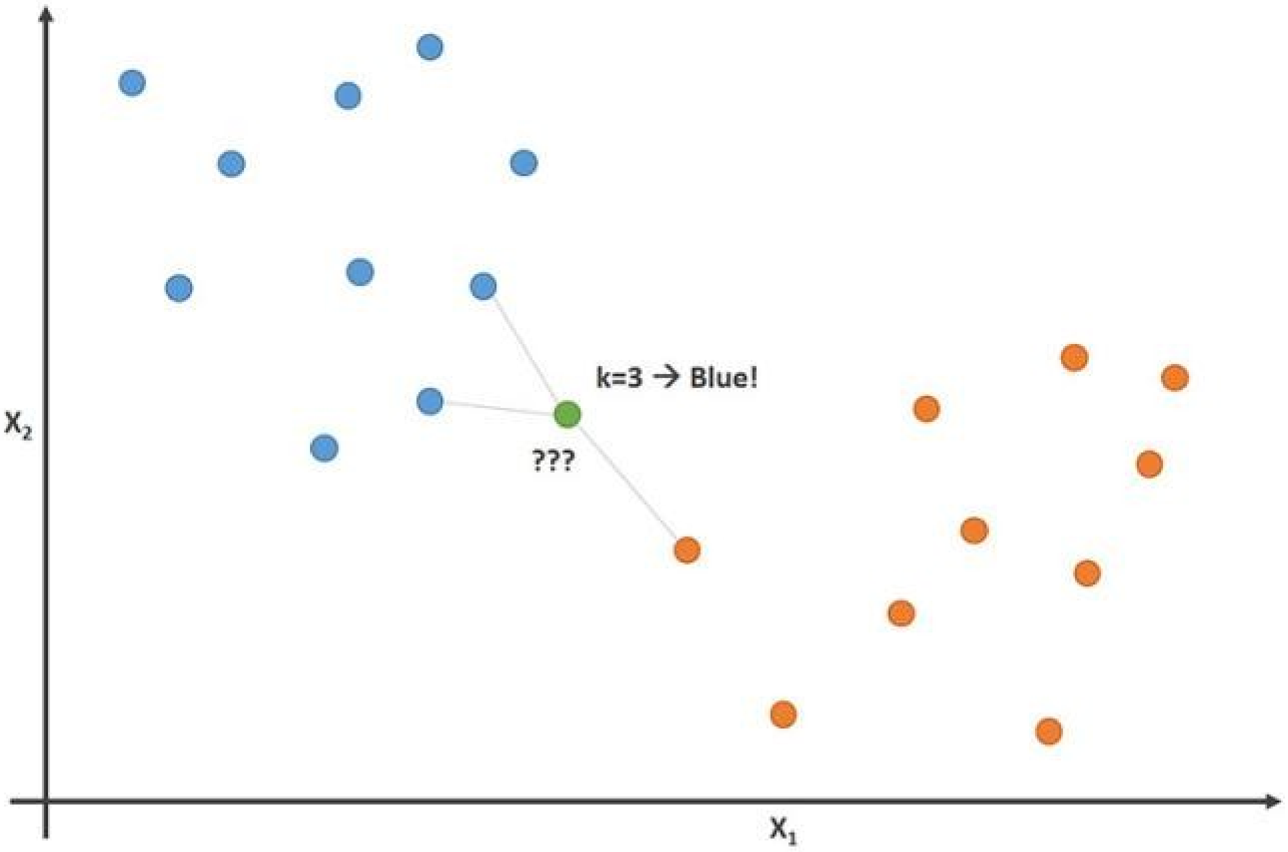
Is my model certain about what it says?
Is there something wrong with this input?
Can I rely on this prediction?

Can I trust this data?
Is something missing?
Is there any bias?

Looking inside a model

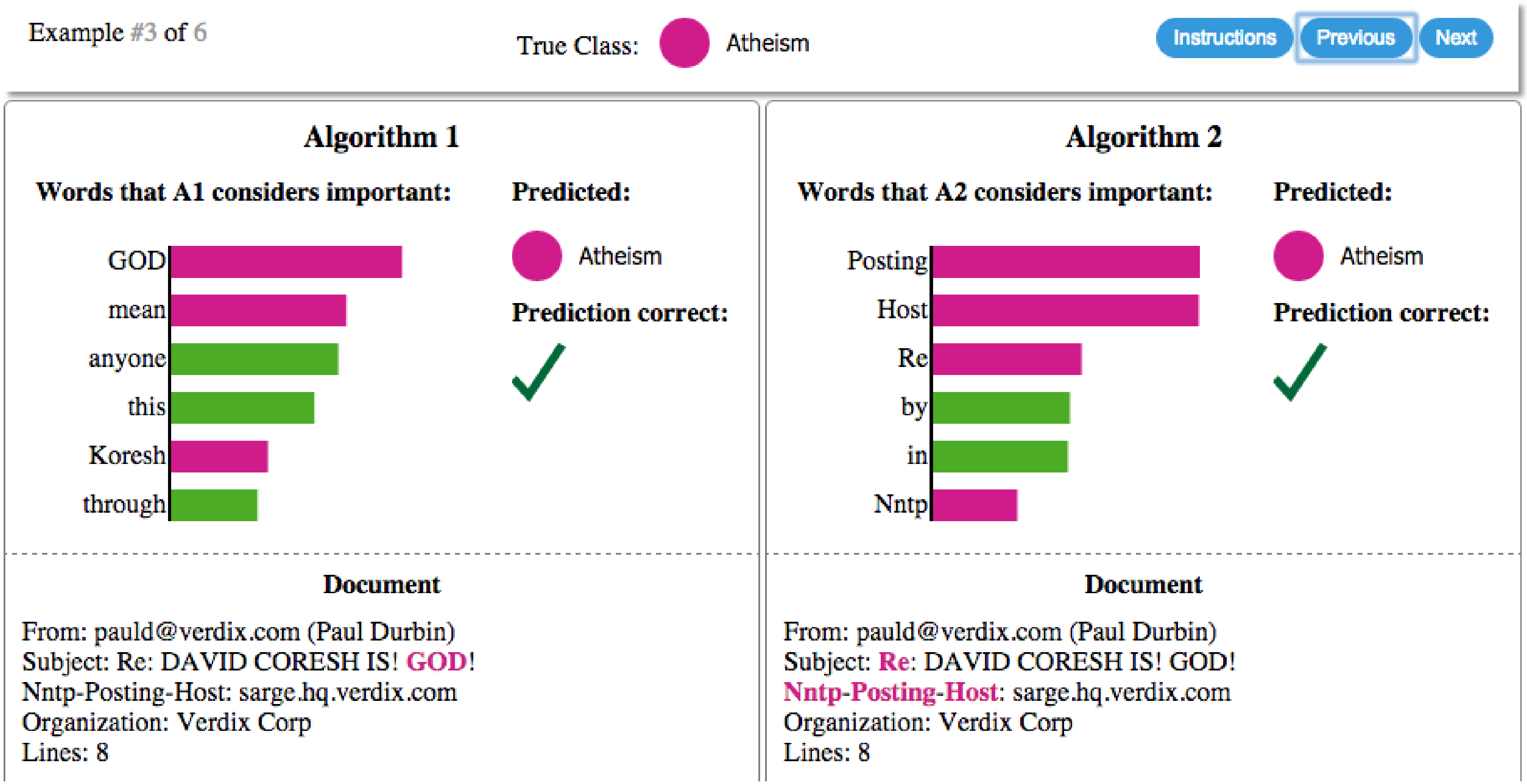
What is interpretable?

Simple stuff like **K Nearest Neighbors**



What is interpretable?

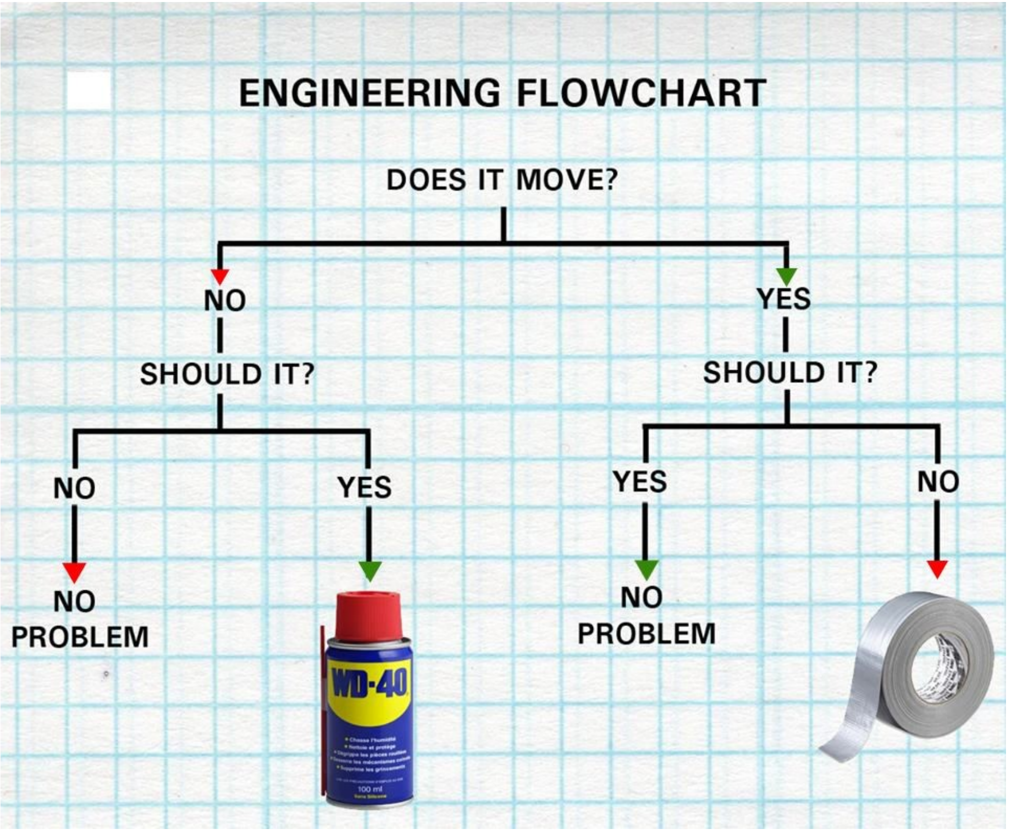
Simple stuff like **Linear models**



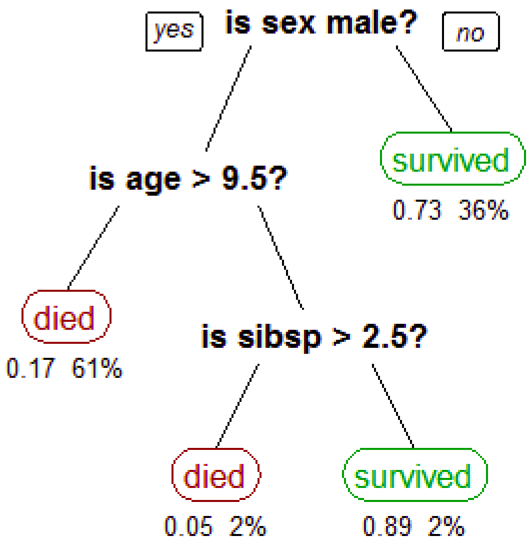
"Why Should I Trust You? Explaining the Predictions of Any Classifier Ribeiro et al., KDD 2016

What is interpretable?

Simple stuff like **Decision Trees**

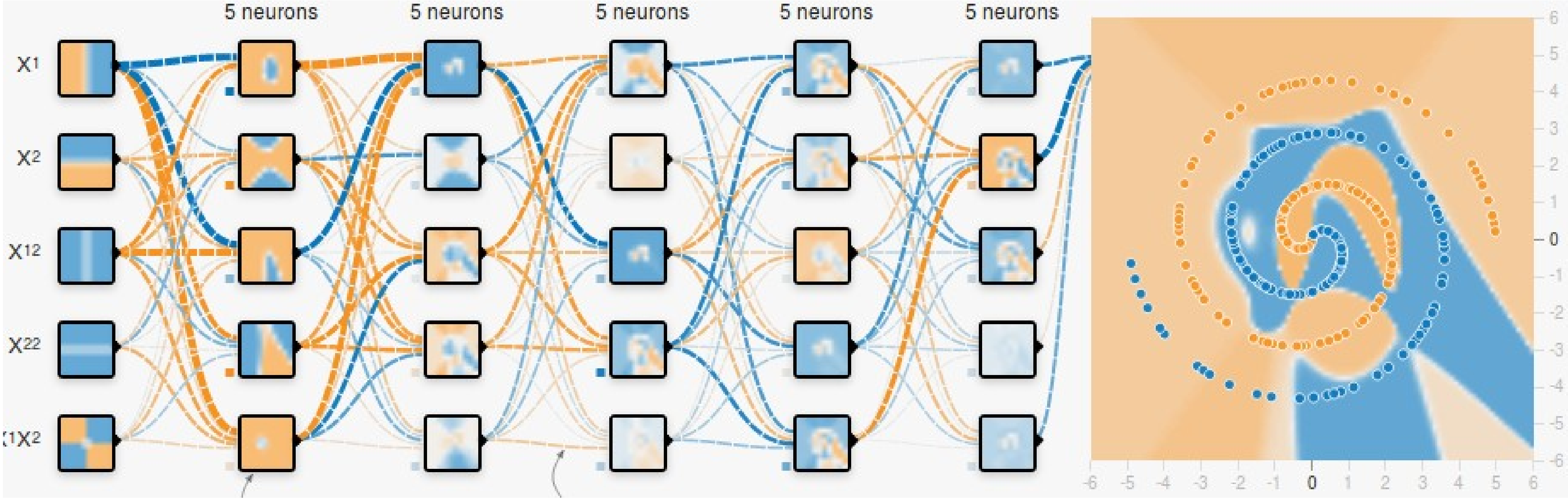


Survival on Titanic



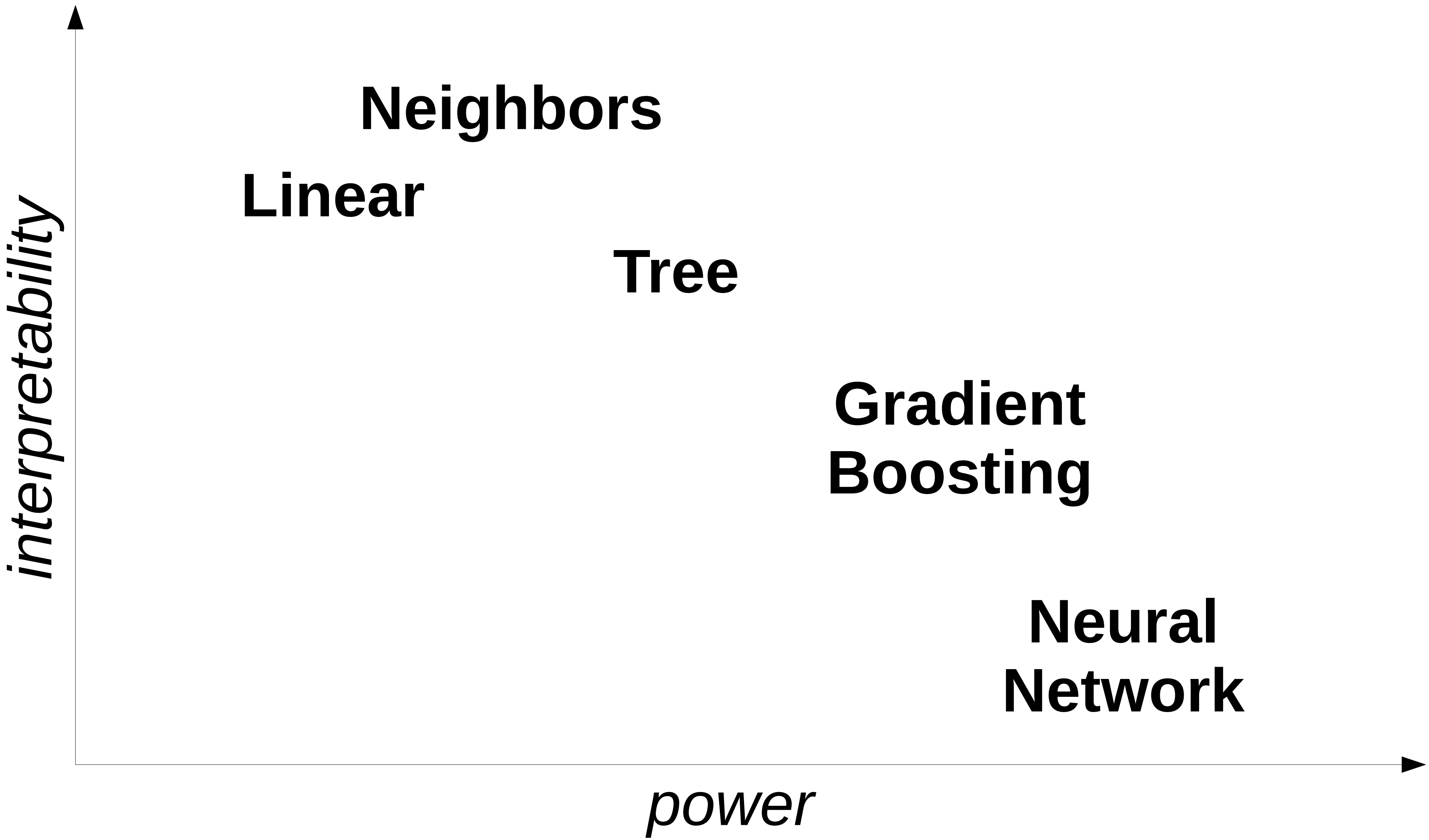
What is interpretable?

Neural networks are **not** naturally interpretable

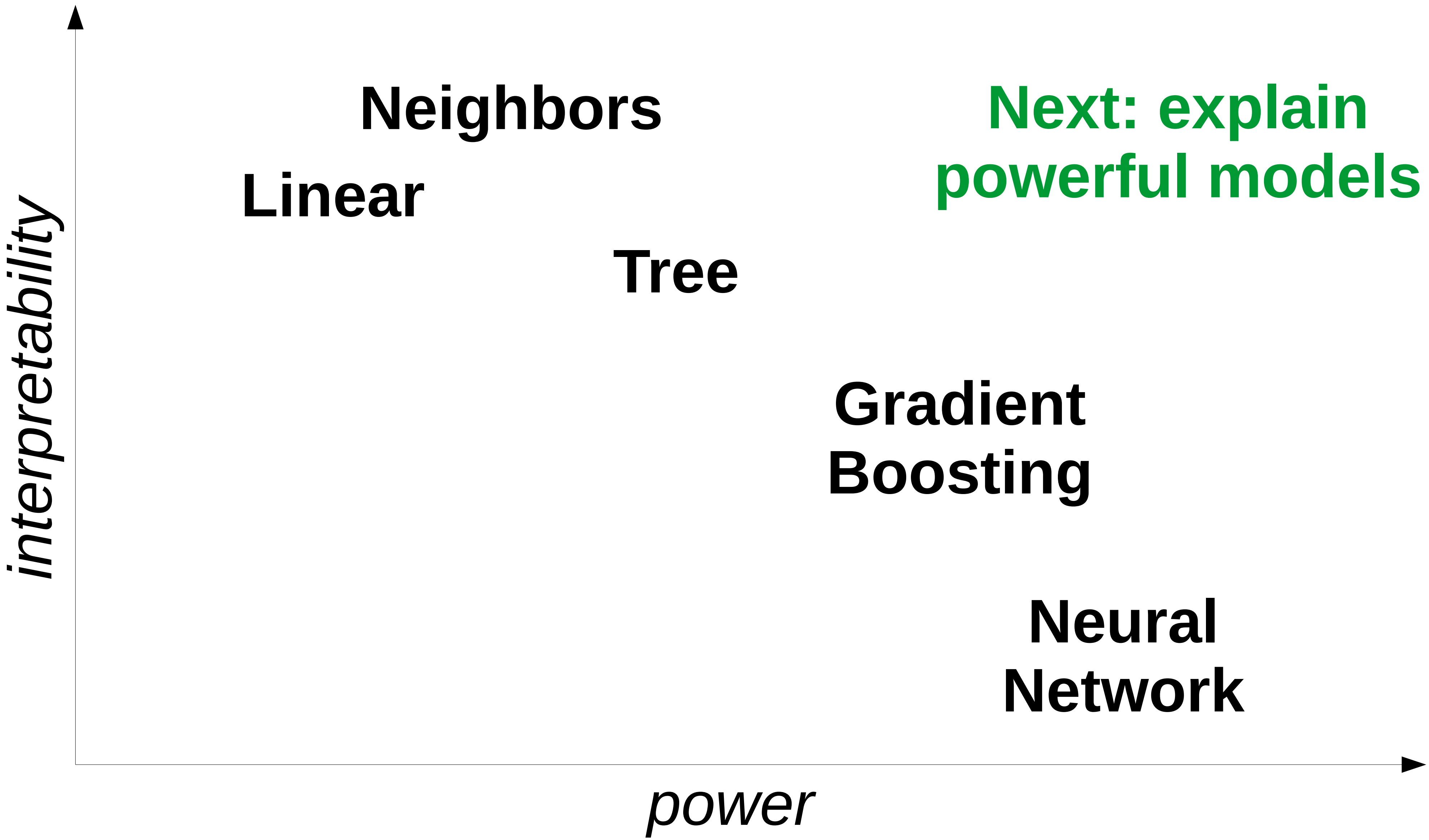


https://playground.tensorflow.org

Power vs interpretability



Power vs interpretability

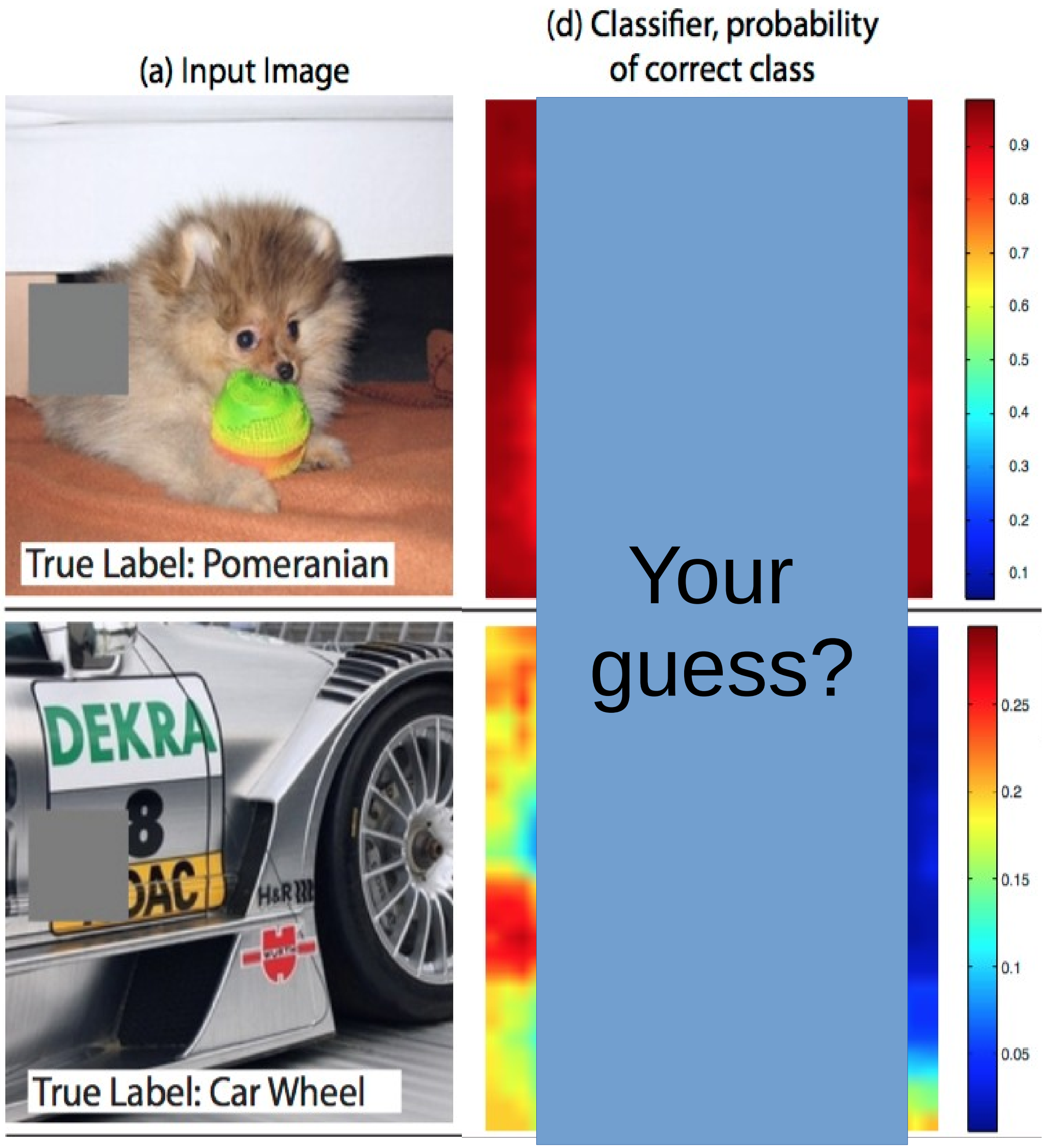


Explanation by occlusion

Idea:

- Add noise
  to inputs and see
  what happens!

- For images: slide
  a gray square over
  the image, measure
  how it affects
  predictions



Explanation by occlusion

Idea:

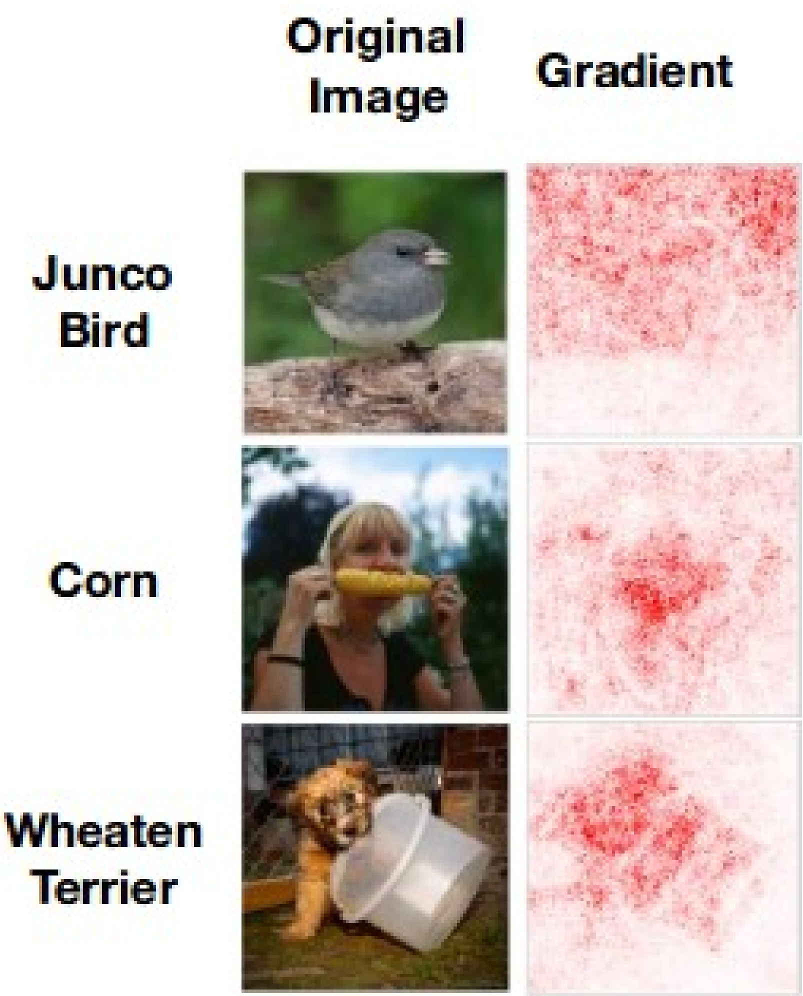
- Add noise to inputs and see what happens!
- For texts: drop individual words and measure how it affects predictions

Predicting salary from job description
https://www.kaggle.com/competitions/job-salary-prediction/data

Explanation by gradients

Idea: use gradients!

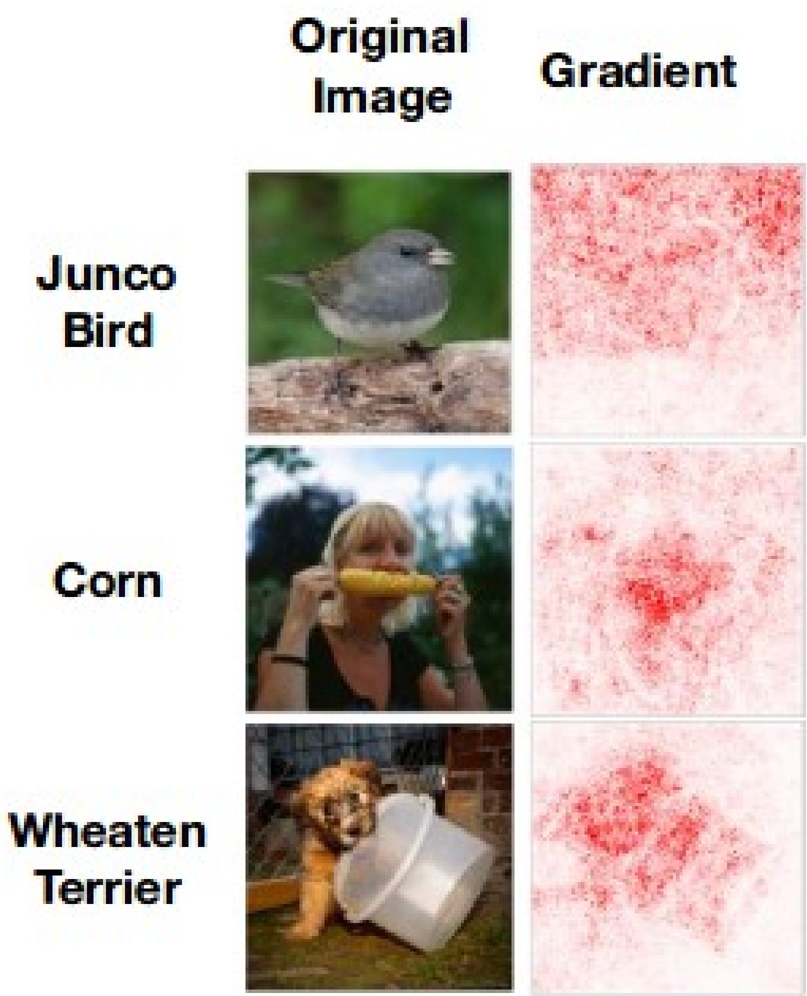
$$\nabla_{x_i} model(x) = \frac{\partial\, model(x)}{\partial\, x_i}$$



Explanation by gradients

Idea: use gradients!

$$\nabla_{x_i} model(x) = \frac{\partial\, model(x)}{\partial x_i}$$



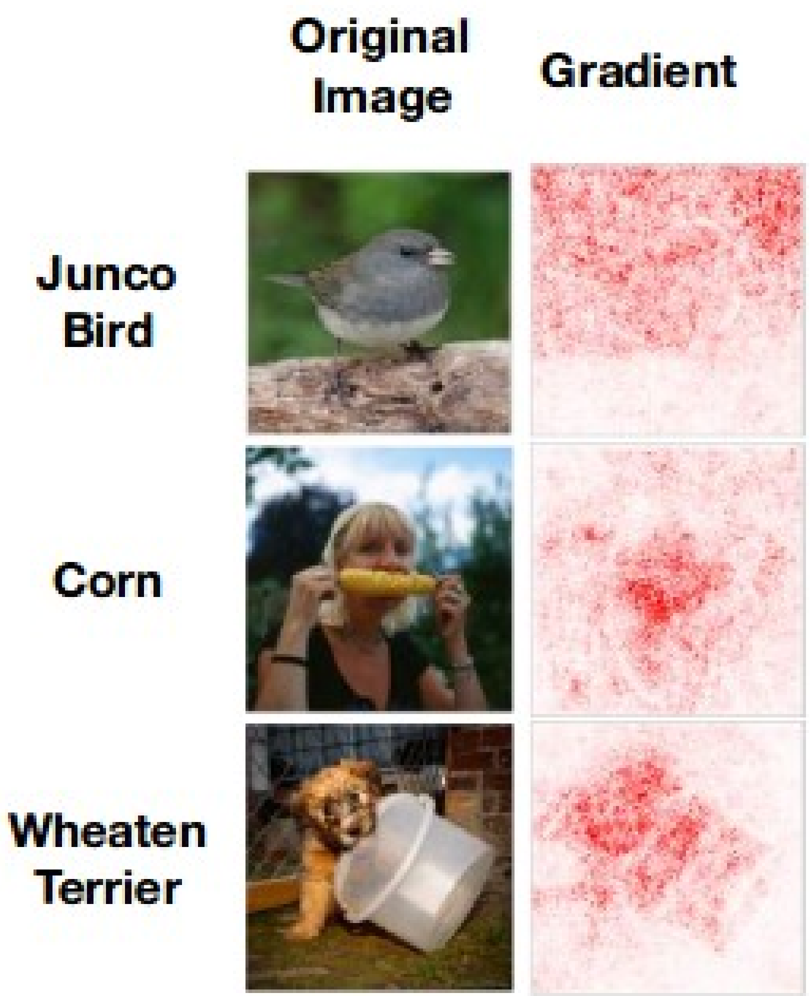
Gradients are too sensitive to small changes in **x**

Q: How would you fix that?

Explanation by gradients

Idea: use gradients!

$$\nabla_{x_i} model(x) = \frac{\partial\, model(x)}{\partial\, x_i}$$



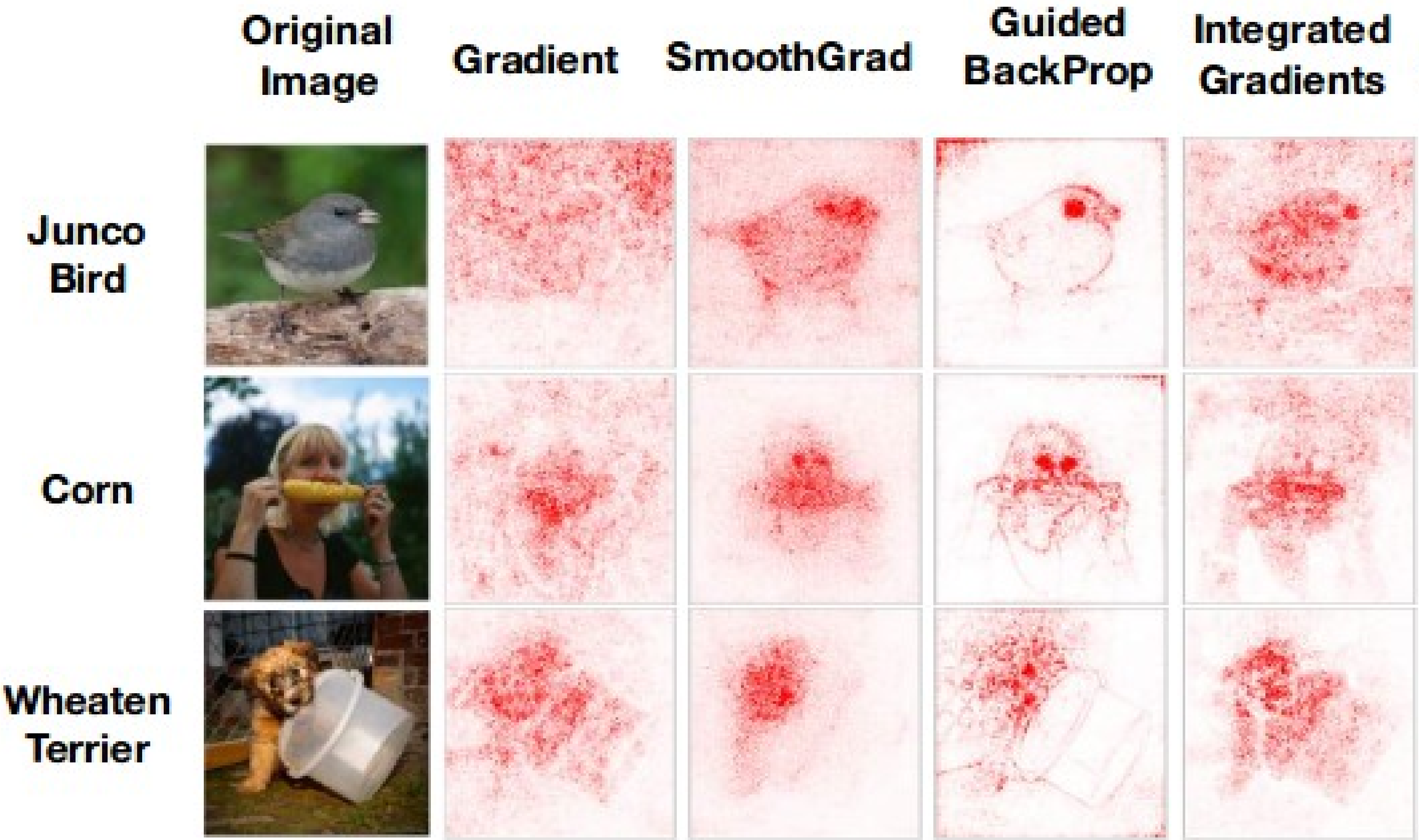
Gradients are too sensitive to small changes in **x**

Smoothgrad: average gradients over several **noisy** copies of **x**

(one of many heuristics)

Explanation by gradients

Idea: use gradients! $$\nabla_{x_i} model(x) = \frac{\partial\, model(x)}{\partial x_i}$$



Quick summary

Explaining models can be done by finding small changes that affect the output

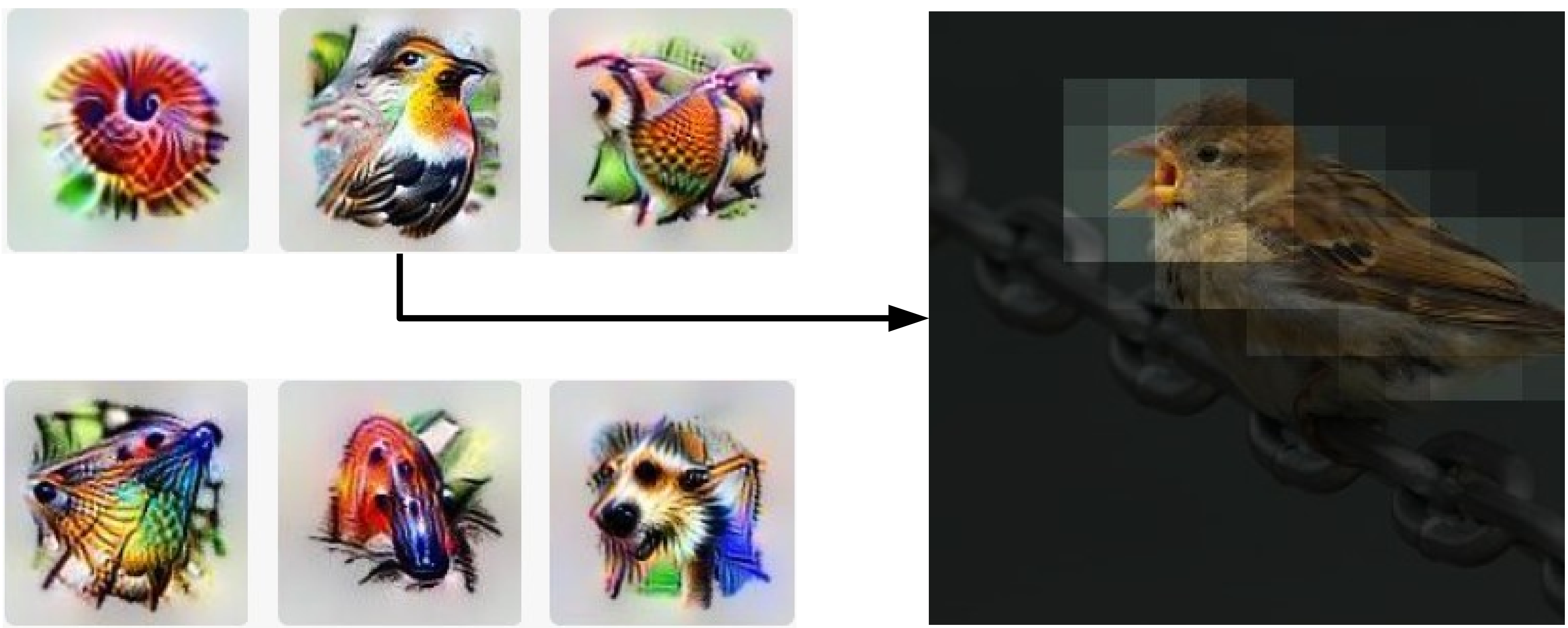
HEP ideas:

- Occlusion of detector pads during reconstruction
- Occlusion of particles during signal selection

Explanation by optimization

Idea: build an image that maximizes the activation of a particular neuron

Must read: distill.pub/2018/building-blocks



Explanation by optimization

Idea: build an image that maximizes the activation of a particular neuron

Must read: **distill.pub/2018/building-blocks**

More:

https://distill.pub

https://poloclub.github.io

https://karpathy.github.io

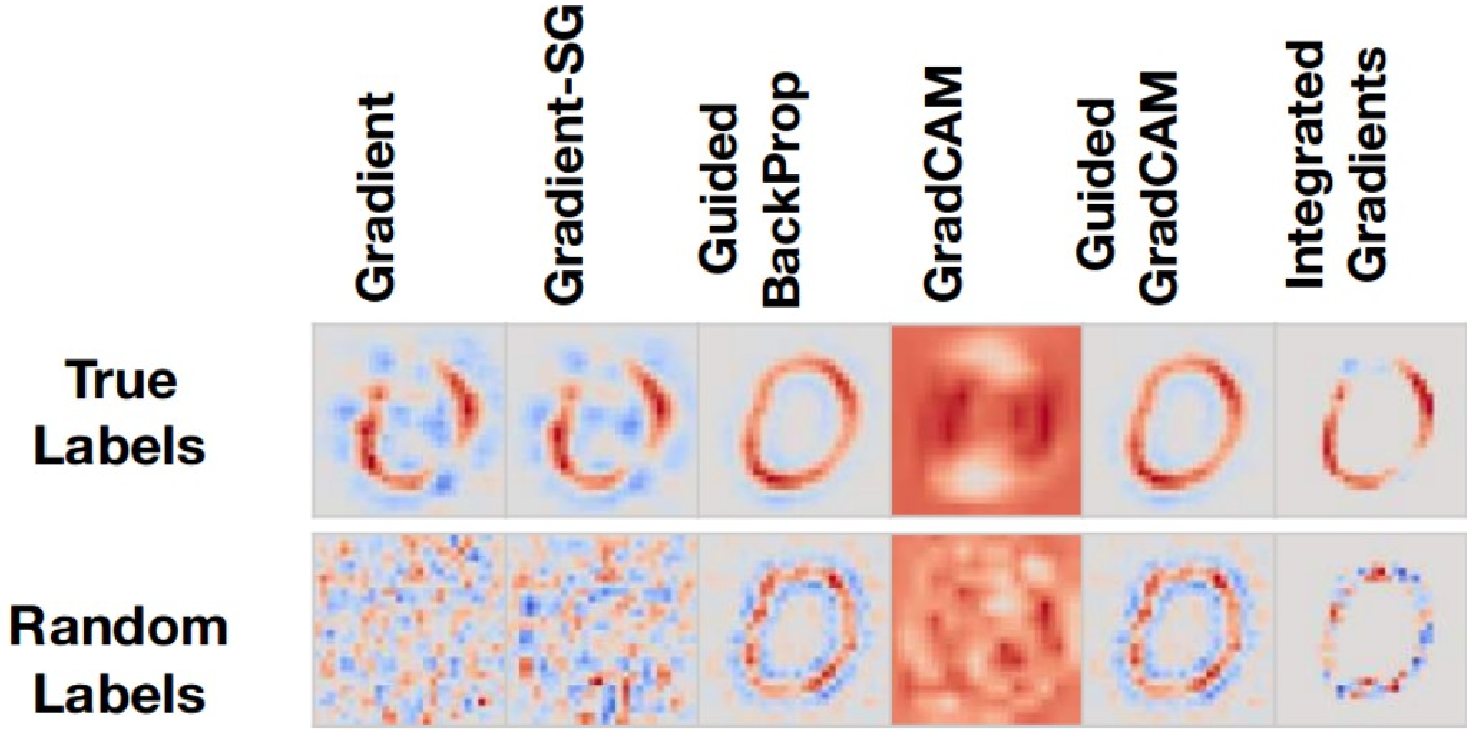
Don’t trust yourself!

The method outputs a noisy image
you see something reasonable
should you be satisfied?

How can you **verify** the explanation?

Don’t trust yourself!

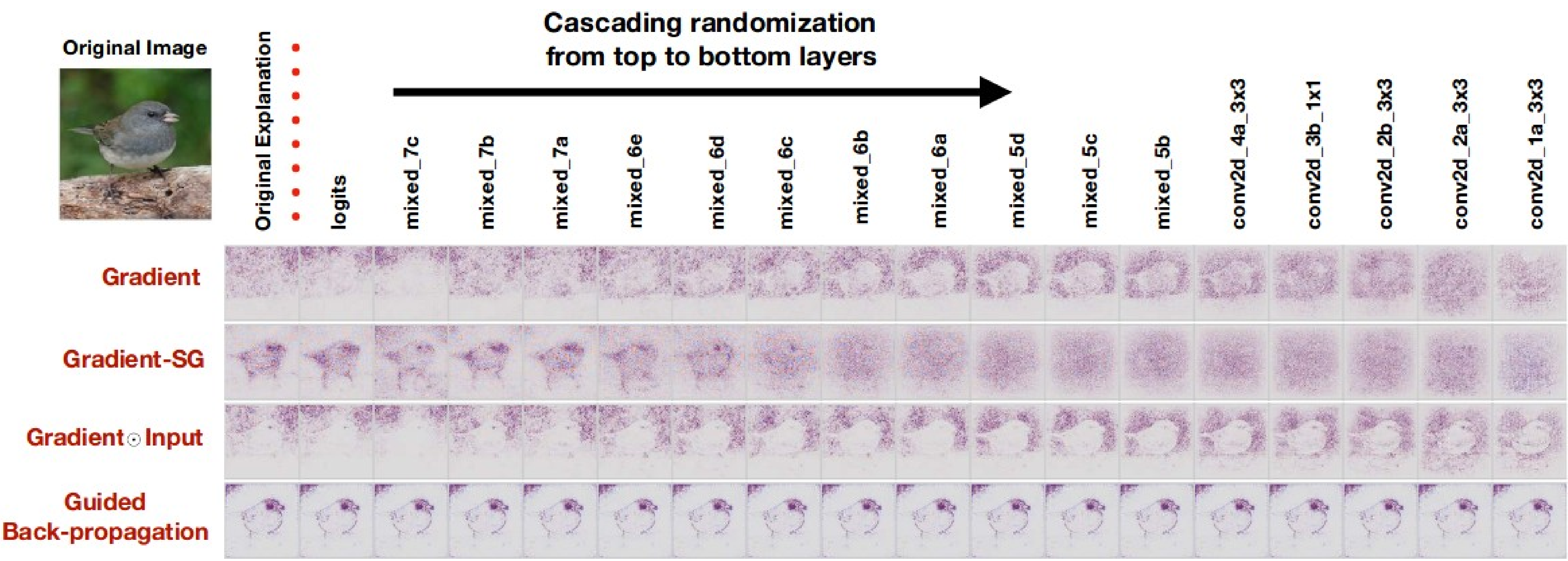
Idea: train a bogus model to see
if the method can “explain” the fake model



Adebayo, Julius, et al. "Sanity checks for saliency maps." Advances in neural information processing systems 31 (2018).

Don't trust yourself!

Idea: replace weights with random one layer at a time (top to bottom)



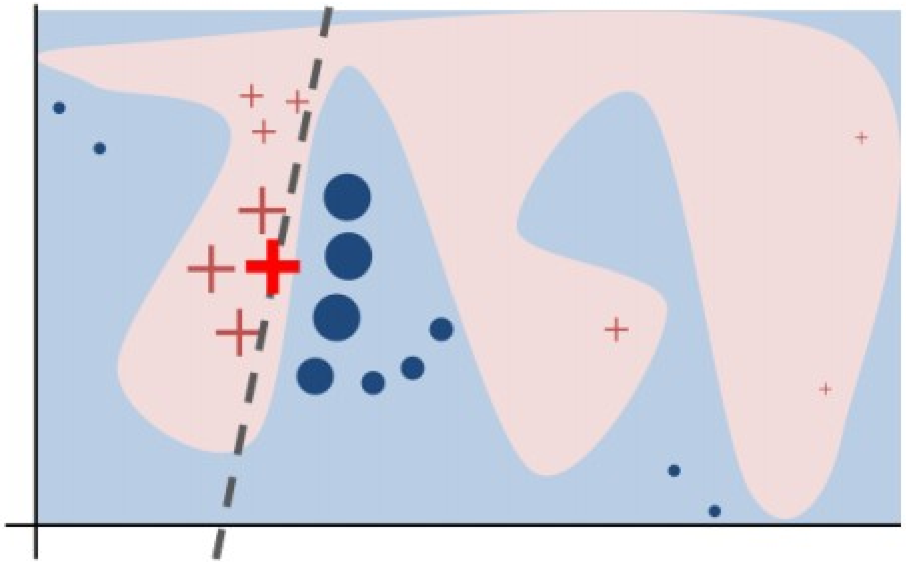
Adebayo, Julius, et al. "Sanity checks for saliency maps." Advances in neural information processing systems 31 (2018).

Explanation by approximation

Idea:

- Approximate your model with something explainable
  e.g. linear model

- The approximation only needs to hold **locally**
  i.e. on similar inputs



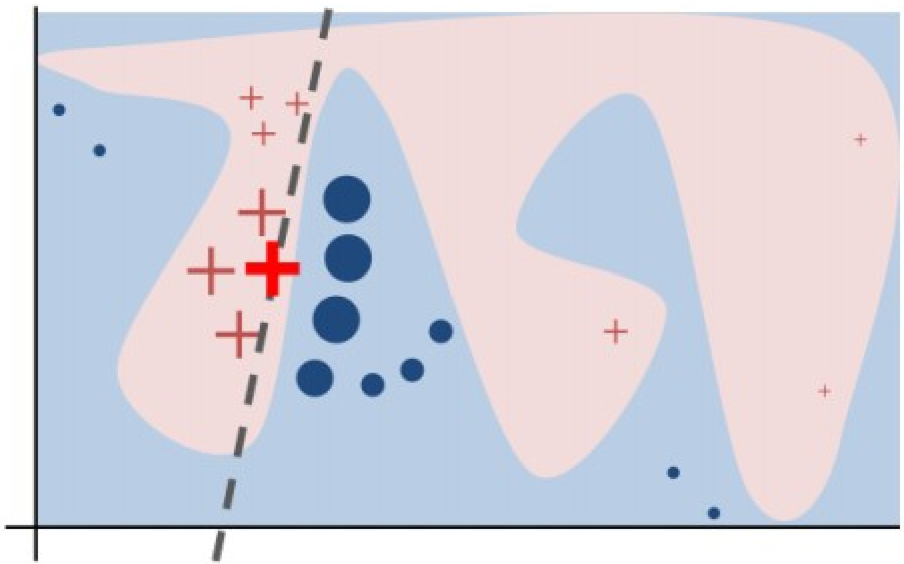
“Why Should I Trust You? Explaining the Predictions of Any Classifier Ribeiro et al., KDD 2016

Explanation by approximation

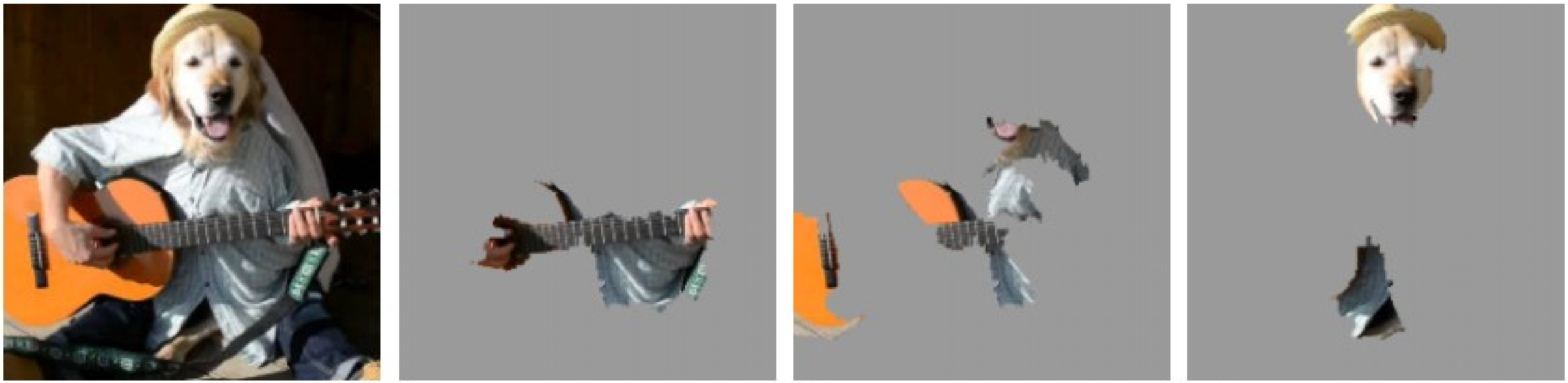
Idea:

- Approximate your model with something explainable
  e.g. linear model

- The approximation only needs to hold **locally**
  i.e. on similar inputs



"Why Should I Trust You? Explaining the Predictions of Any Classifier Ribeiro et al., KDD 2016



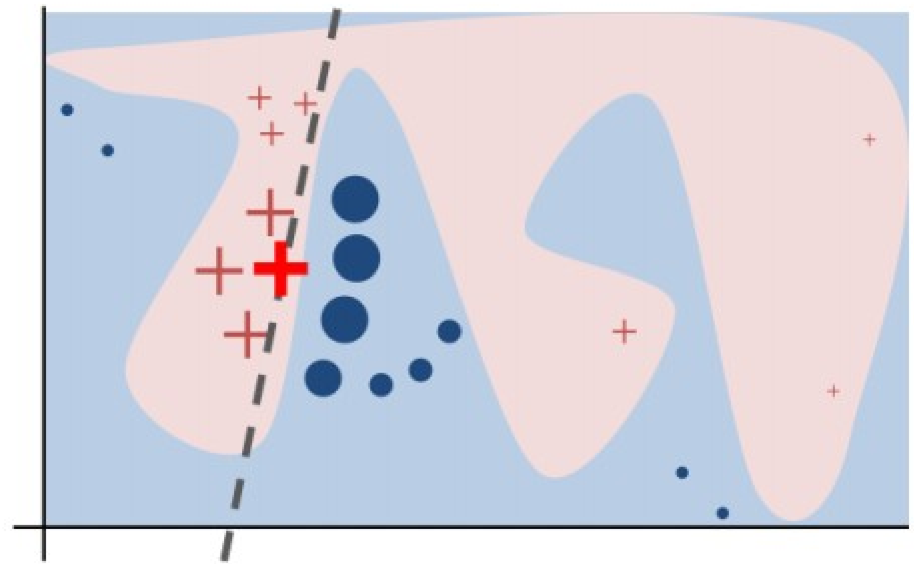
(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

Explanation by approximation

Idea:

- Approximate your model with something explainable
  e.g. linear model

- The approximation only needs to hold **locally**
  i.e. on similar inputs



"Why Should I Trust You? Explaining the Predictions of Any Classifier Ribeiro et al., KDD 2016

(a) Husky classified as wolf (b) Explanation

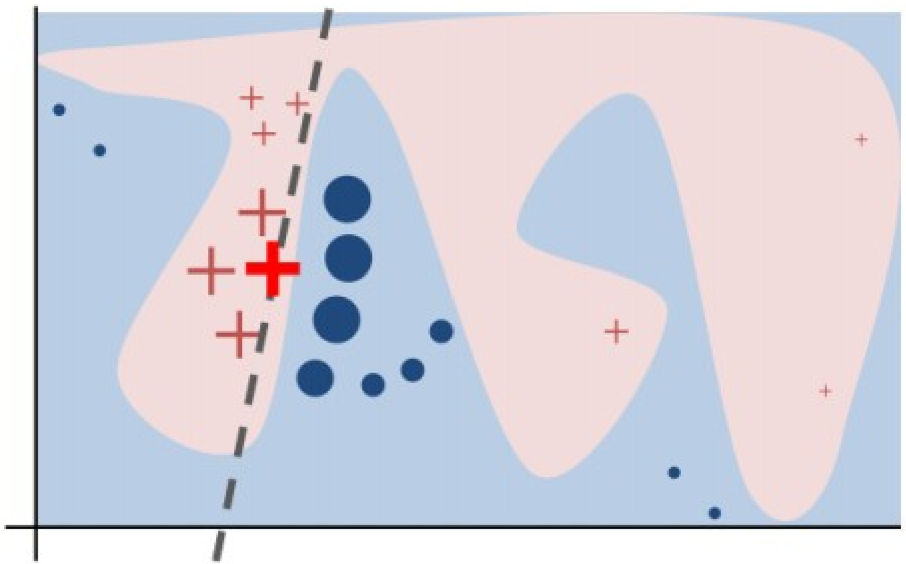
Left image: model mislabeled a husky dog as a wolf; explanation: snow :)

Explanation by approximation

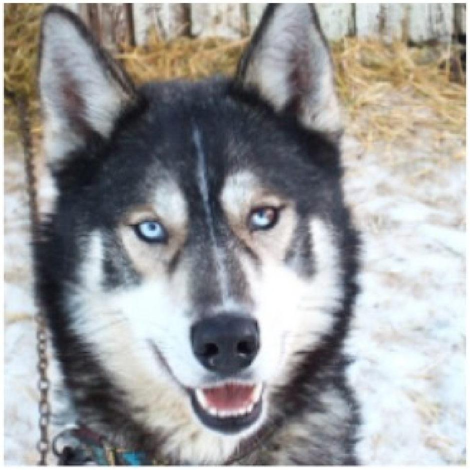
Idea:

- Approximate your model with something explainable
  e.g. linear model

- The approximation only needs to hold **locally**
  i.e. on similar inputs



"Why Should I Trust You? Explaining the Predictions of Any Classifier Ribeiro et al., KDD 2016



(a) Husky classified as wolf

(b) Explanation

Read more:
arxiv.org/abs/1602.04938
arxiv.org/abs/1705.07874
arxiv.org/abs/1904.12991

Explanation by game theory

Idea: features are a “players” that play
a cooperative game of making a prediction

Explanation by game theory

Idea: features are a "players" that play
a cooperative game of making a prediction

Equivalent "game":
Alice, **B**ob and **C**arol ordered a $1000 meal at a restaurant
Q: How should they split the bill?

Hint: here's what it would cost for them individually & in pairs

Who goes	**A**lice	**B**ob	**C**arol	**A** & **B**	**A** & **C**	**B** & **C**	**A**, **B** & **C**
Total price	400	560	720	740	780	980	1000

Ideas?

Explanation by game theory

Idea: features are a "players" that play
a cooperative game of making a prediction

Equivalent "game":
Alice, **B**ob and **C**arol ordered a $1000 meal at a restaurant
Q: How should they split the bill?

Hint: here's what it would cost for them individually & in pairs

Who goes	**A**lice	**B**ob	**C**arol	**A** & **B**	**A** & **C**	**B** & **C**	**A**, **B** & **C**
Total price	400	560	720	740	780	980	1000

Game theorist's answer: Shapley values!

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!}(v(S \cup \{i\}) - v(S))$$

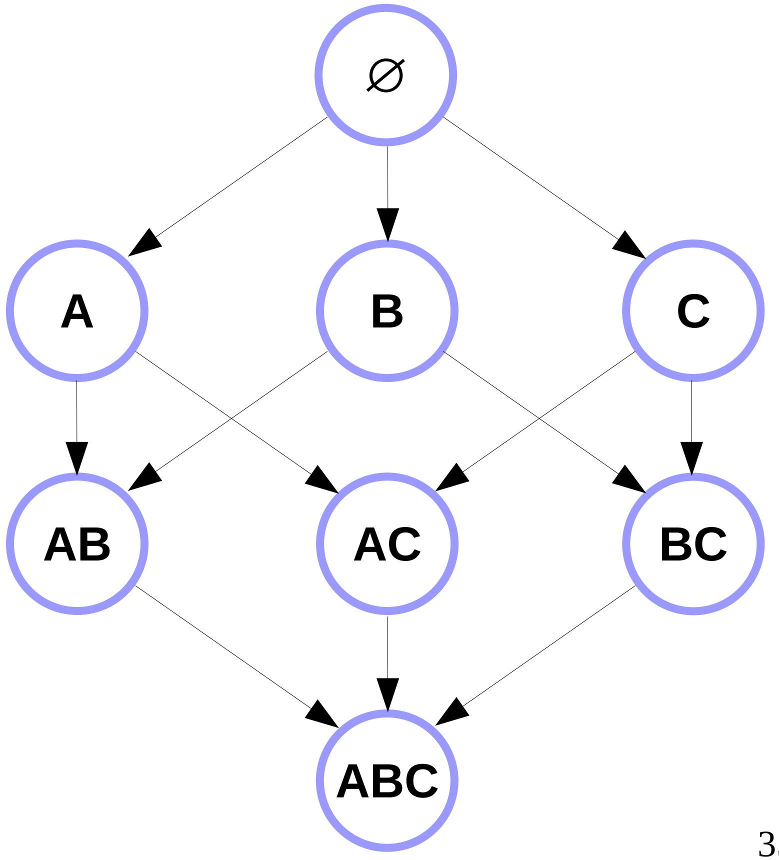
Shapley values explained

Same old table

Who goes	Alice	Bob	Carol	A & B	A & C	B & C	A, B & C
Total price	400	560	720	740	780	980	1000

Shapley(X) = average increase in cost from adding X to a group

Note: average over all ***paths***.



Shapley values explained

Same old table

Who goes	Alice	Bob	Carol	A & B	A & C	B & C	A, B & C
Total price	400	560	720	740	780	980	1000

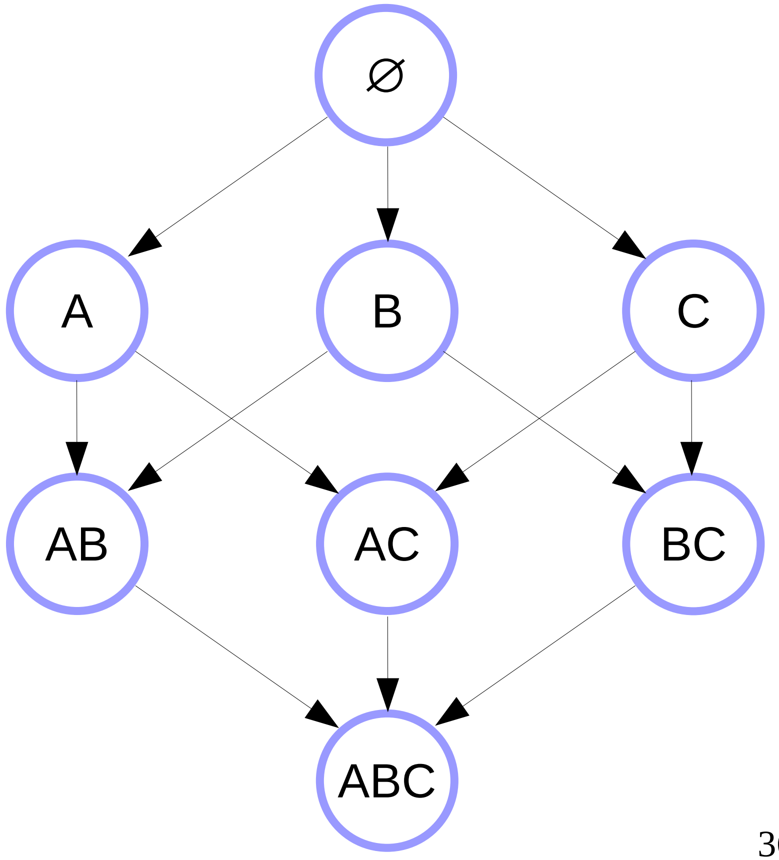
Shapley(X) = average increase in cost from adding X to a group

Note: averaging over all ***paths*** *is* ***NP-hard***

Shapley(A) = $\frac{2}{6}\cdot 400$ **+** $\dots$

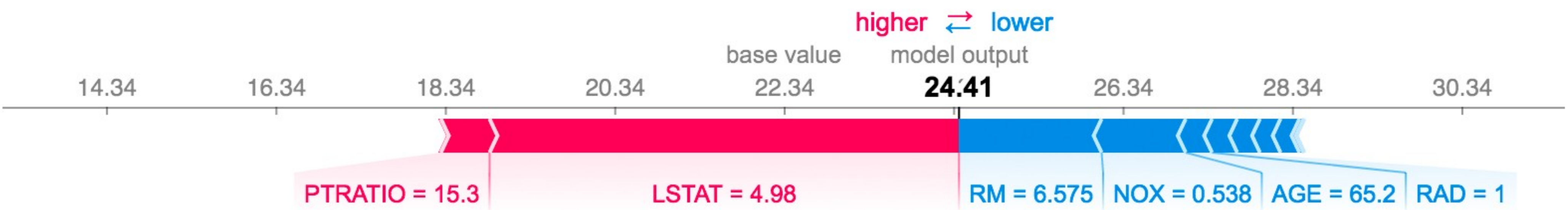
+ $\frac{1}{6}\cdot\left(740-560\right)$ **+** $\frac{1}{6}\cdot\left(780-720\right)$ **+**

+ $\frac{2}{6}\cdot\left(1000-990\right) = 180$



Explanation by game theory

SHAP *= Shapley values for features + clever approximation*
State of the art in after-the-fact model explanation



Links:

- SHAP original paper: tinyurl.com/shap-paper (NeurIPS'17)
- SHAP explained by paper author: youtu.be/ngOBhhINWb8
- Shapley values in game theory: youtu.be/w9O0fkfMkx0

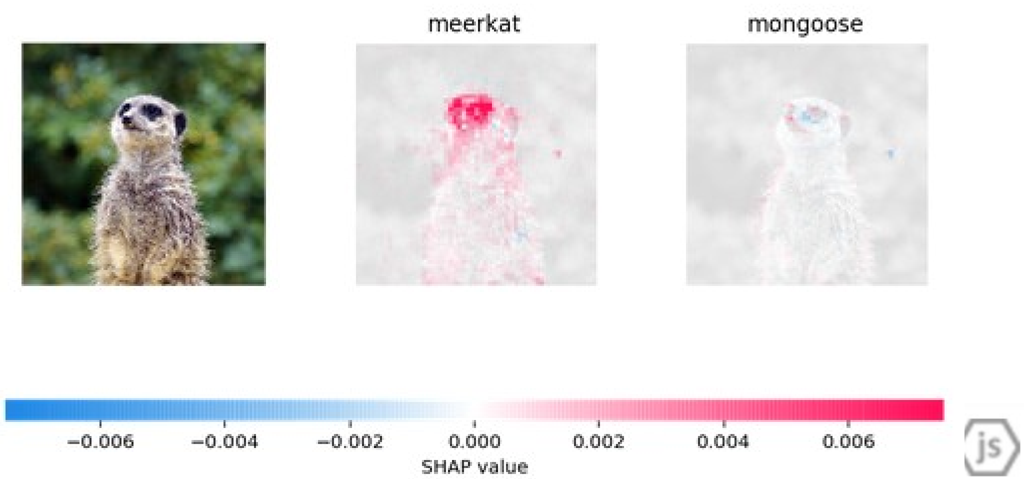
Frameworks

SHAP - https://github.com/slundberg/shap

(tensorflow, keras, pytorch, sklearn-like)

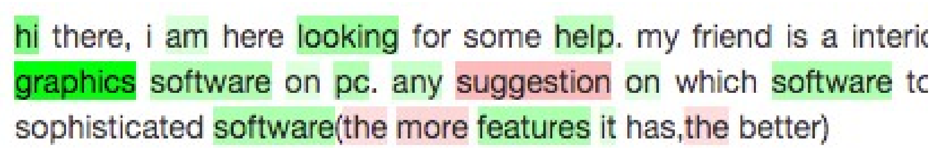
ELI5 - https://github.com/TeamHG-Memex/eli5

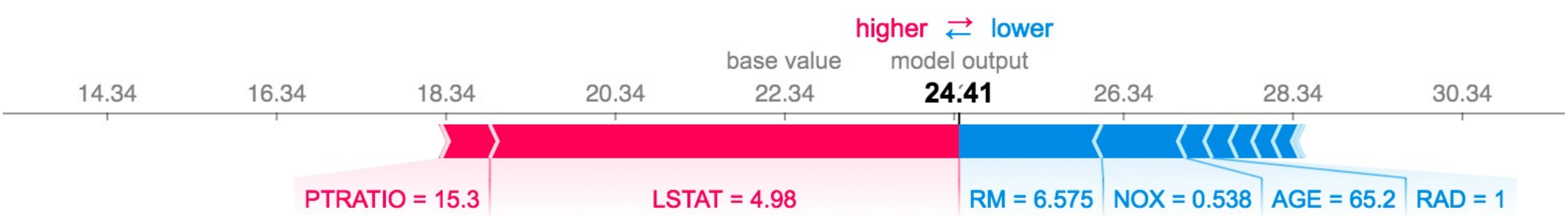
(popular explainers for keras/tf, sklearn-like)



Left, bottom – shap

Right - ELI5



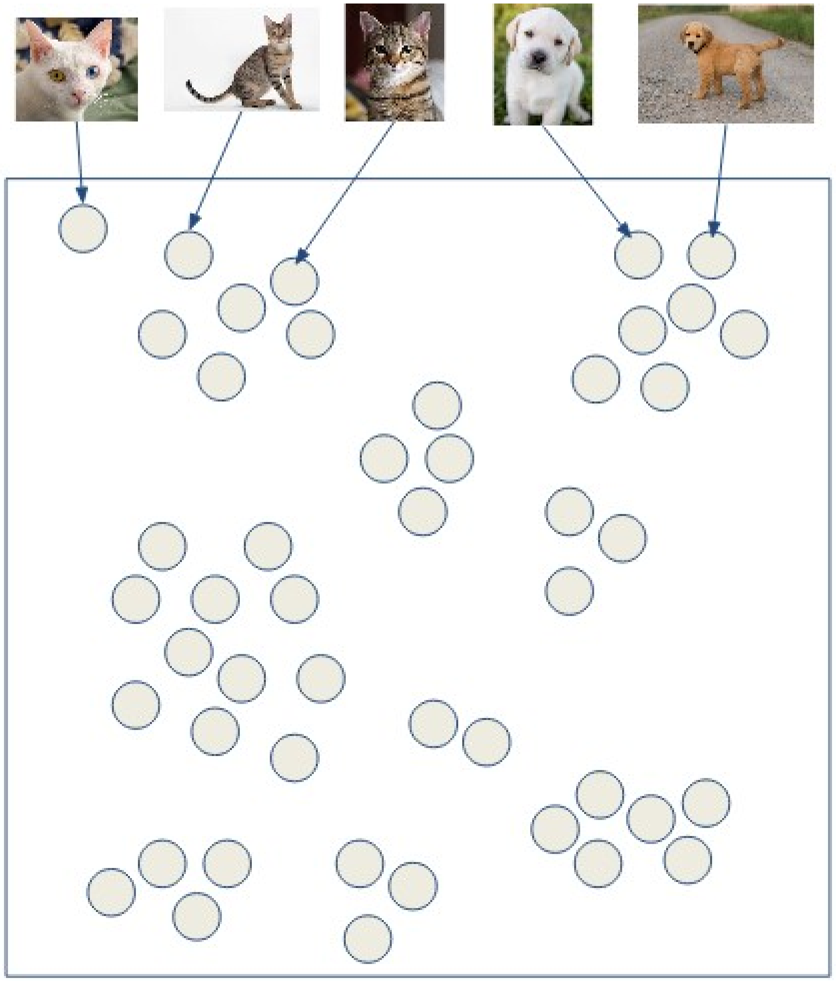


So far: explaining black-box models

Now: model-specific methods

Explanation by design

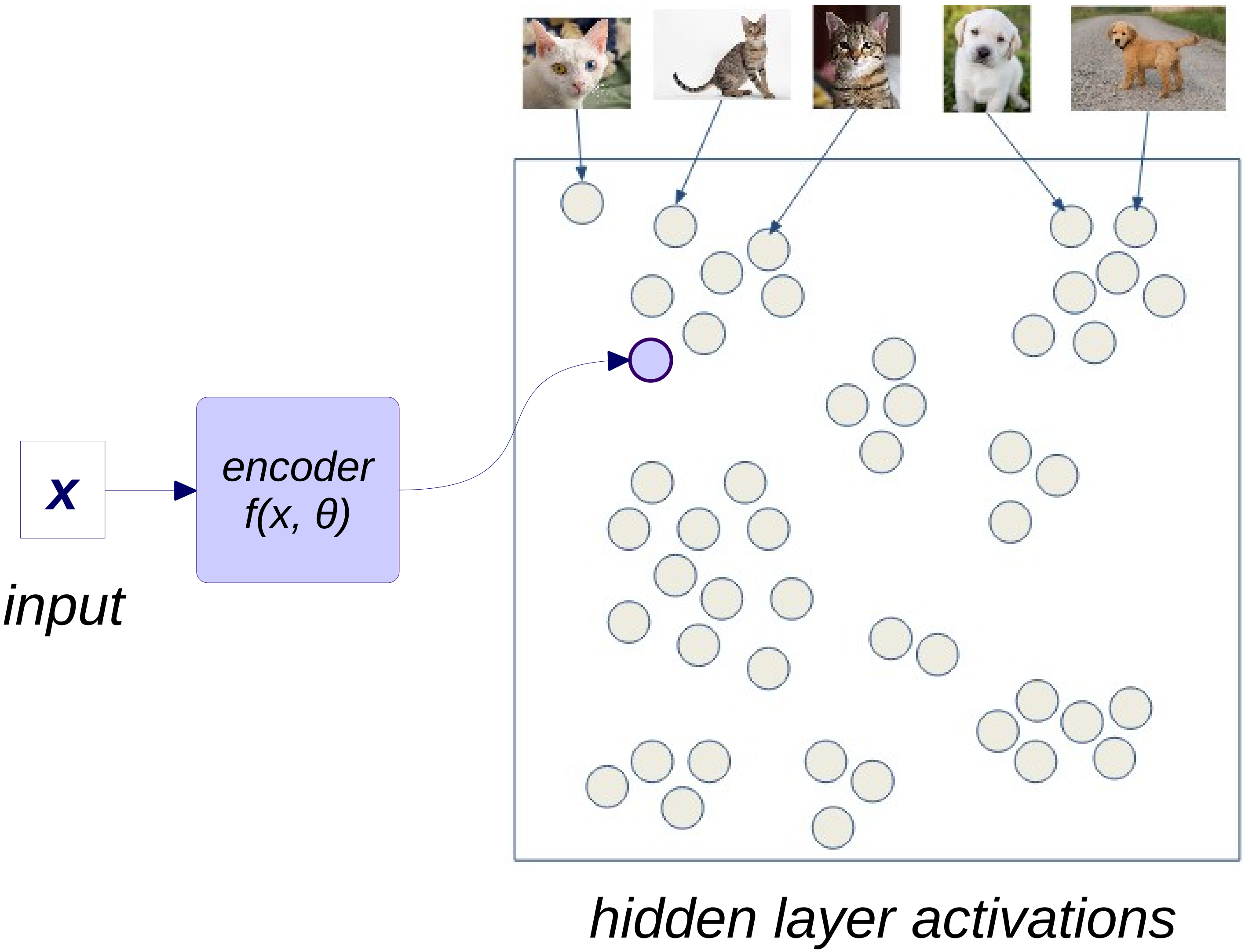
Idea: design architecture to be interpretable



hidden layer activations

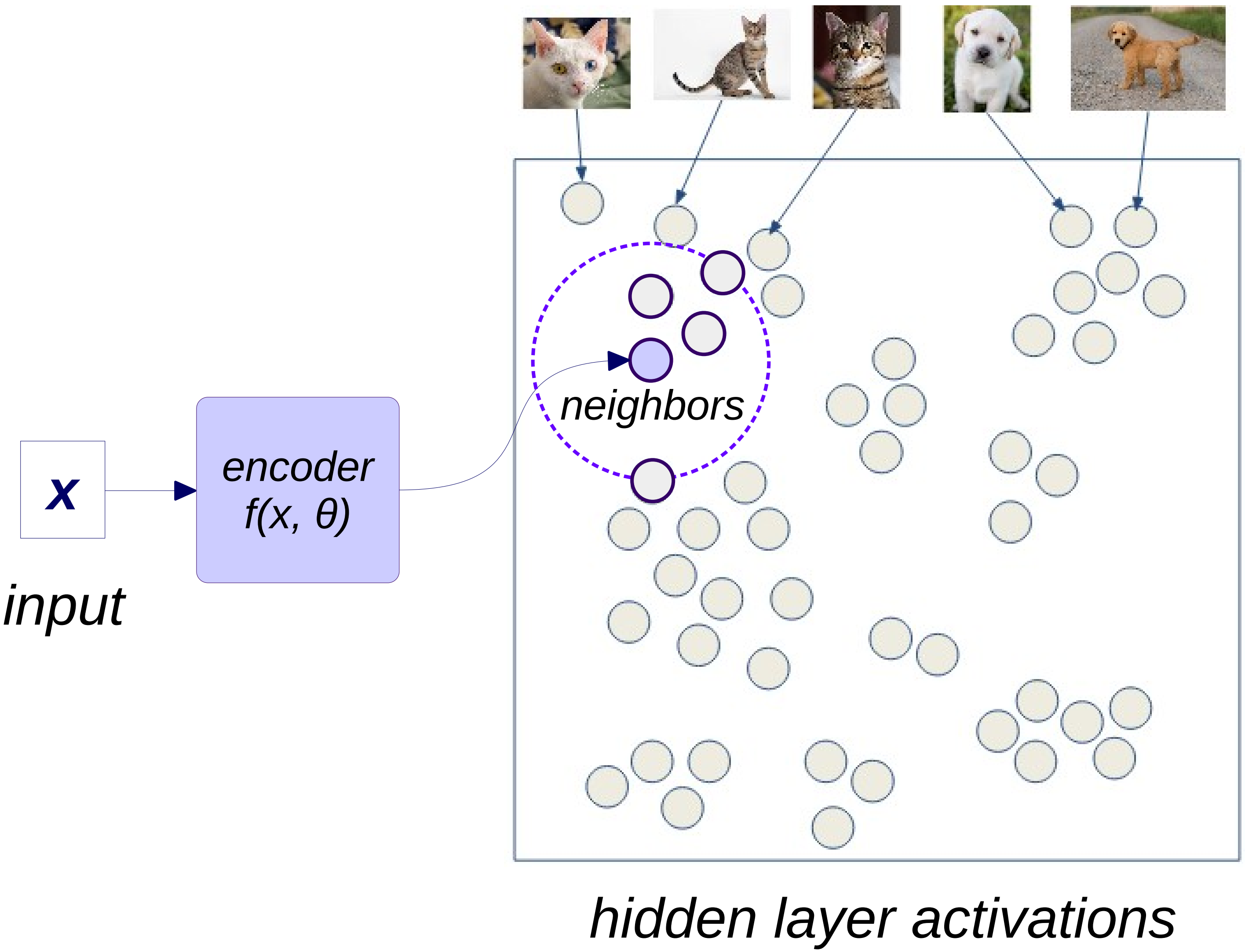
Explanation by design

Idea: design architecture to be interpretable



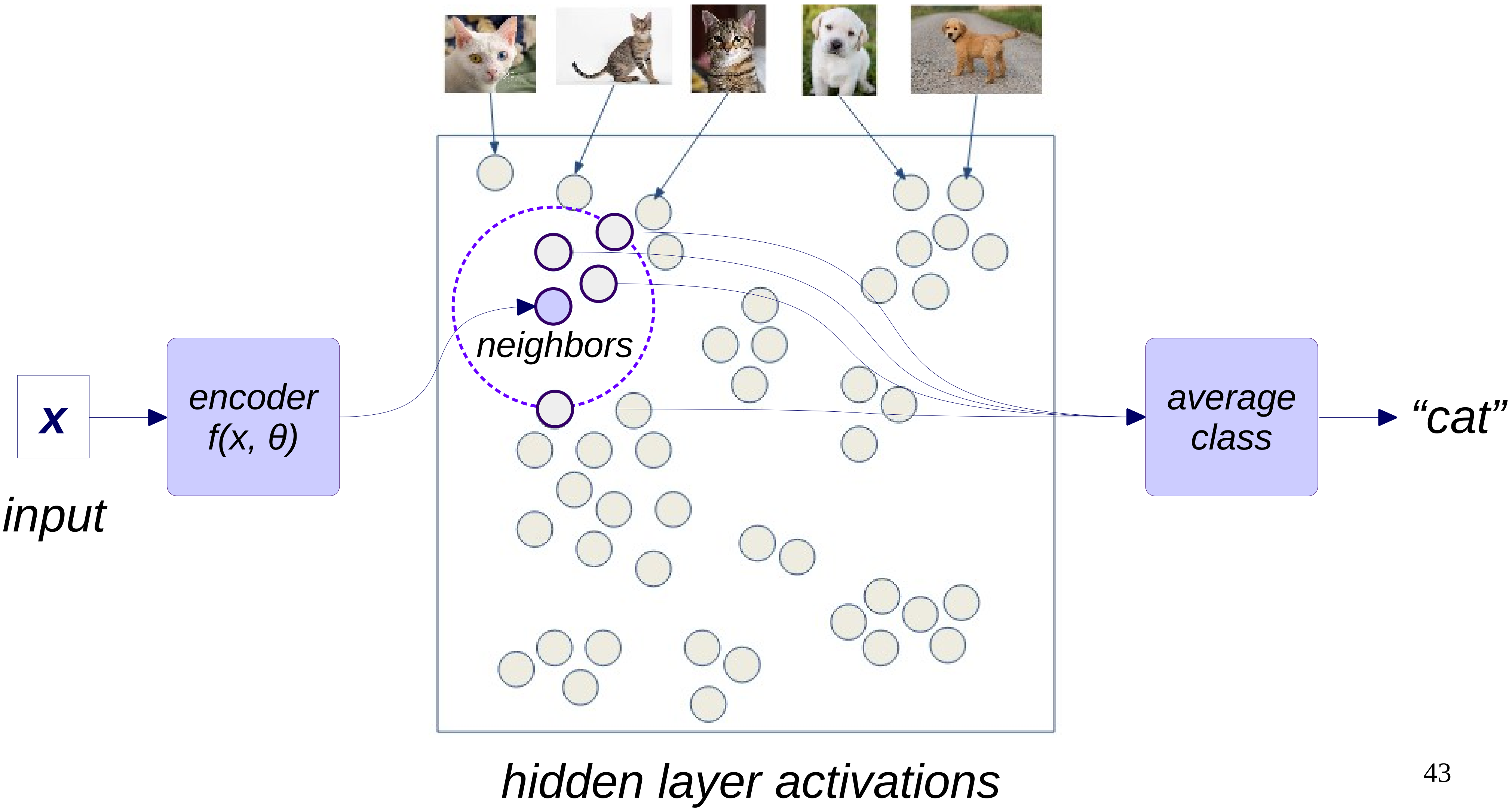
Explanation by design

Idea: design architecture to be interpretable



Explanation by design

Idea: design architecture to be interpretable



Explanation by design

Idea: design architecture to be interpretable

Prototype objects and answers: $\left(\hat{x}_{0,}\hat{y}_0\right),\ldots,\left(\hat{x}_N,\hat{y}_N\right)$

"Attention" weights: $a\left(x,\hat{x}_i\right)=\frac{e^{\langle f(x,theta),f(\hat{x}_i,theta)\rangle}}{\sum_{j=0}^{N} e^{\langle f(x,theta),f(\hat{x}_j,theta)\rangle}}$

Prediction by averaging: $y^{pred}(x)=\sum_i \hat{y}_i\cdot a_i\left(x,\hat{x}_i\right)$

Explanation by design

Idea: design architecture to be interpretable

Prototype objects and answers: $(\hat{x}_{0,} \hat{y}_0), \ldots, (\hat{x}_N, \hat{y}_N)$

"Attention" weights: $$a(x, \hat{x}_i) = \frac{e^{\langle f(x, theta), f(\hat{x}_i, theta) \rangle}}{\sum_{j=0}^{N} e^{\langle f(x, theta), f(\hat{x}_j, theta) \rangle}}$$

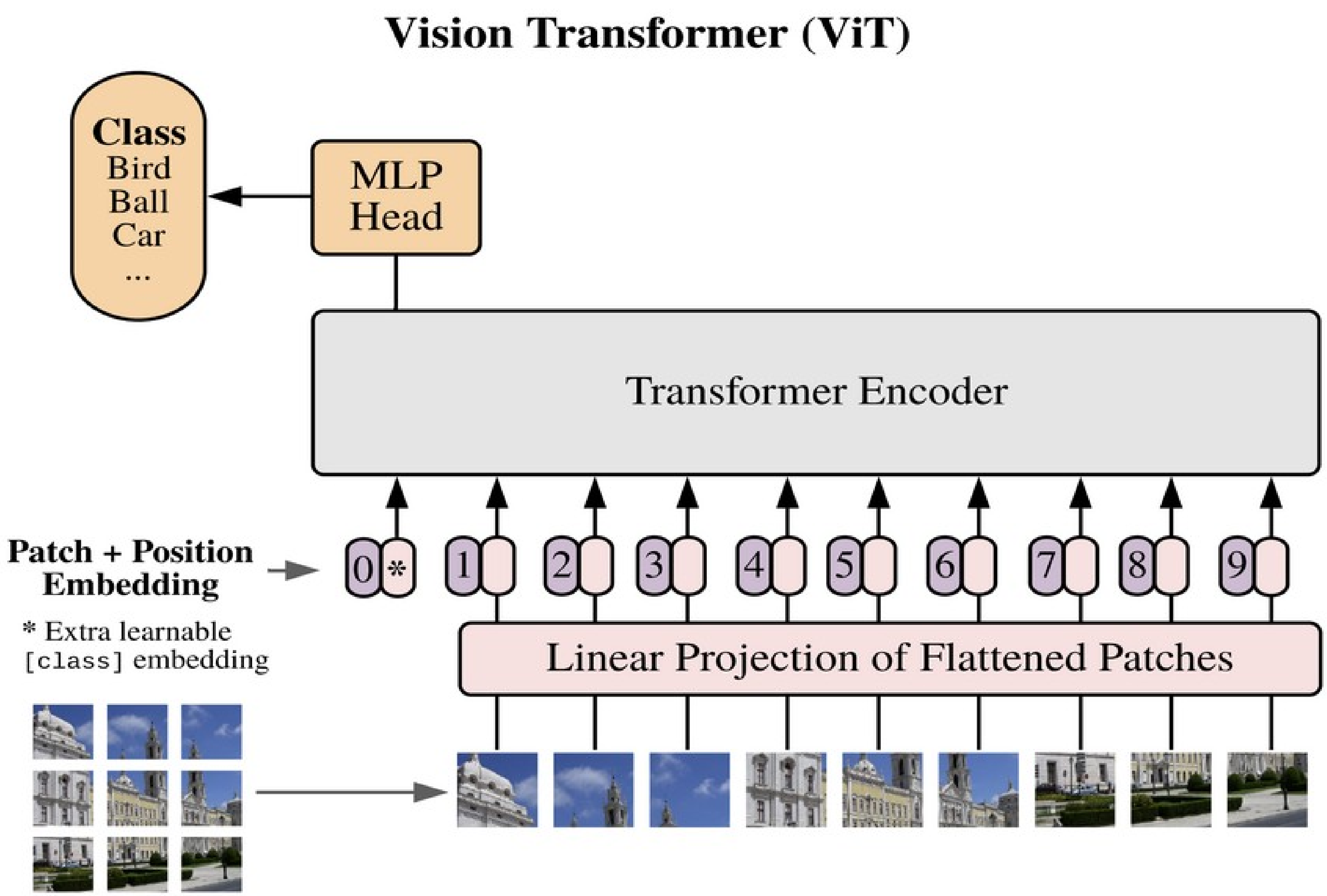
$$y^{pred}(x) = \sum_i \hat{y}_i \cdot a_i(x, \hat{x}_i)$$

Read more: KNN
arxiv.org/abs/1703.05175
arxiv.org/abs/1803.04765
arxiv.org/abs/1809.02847

Read more: Linear
arxiv.org/abs/1705.08078
arxiv.org/abs/1806.07538

Taking it to the extreme

Paper: https://arxiv.org/abs/2010.11929



Taking it to the extreme

Paper: https://arxiv.org/abs/2104.14294

View attention maps: https://epfml.github.io/attention-cnn/

The question of trust

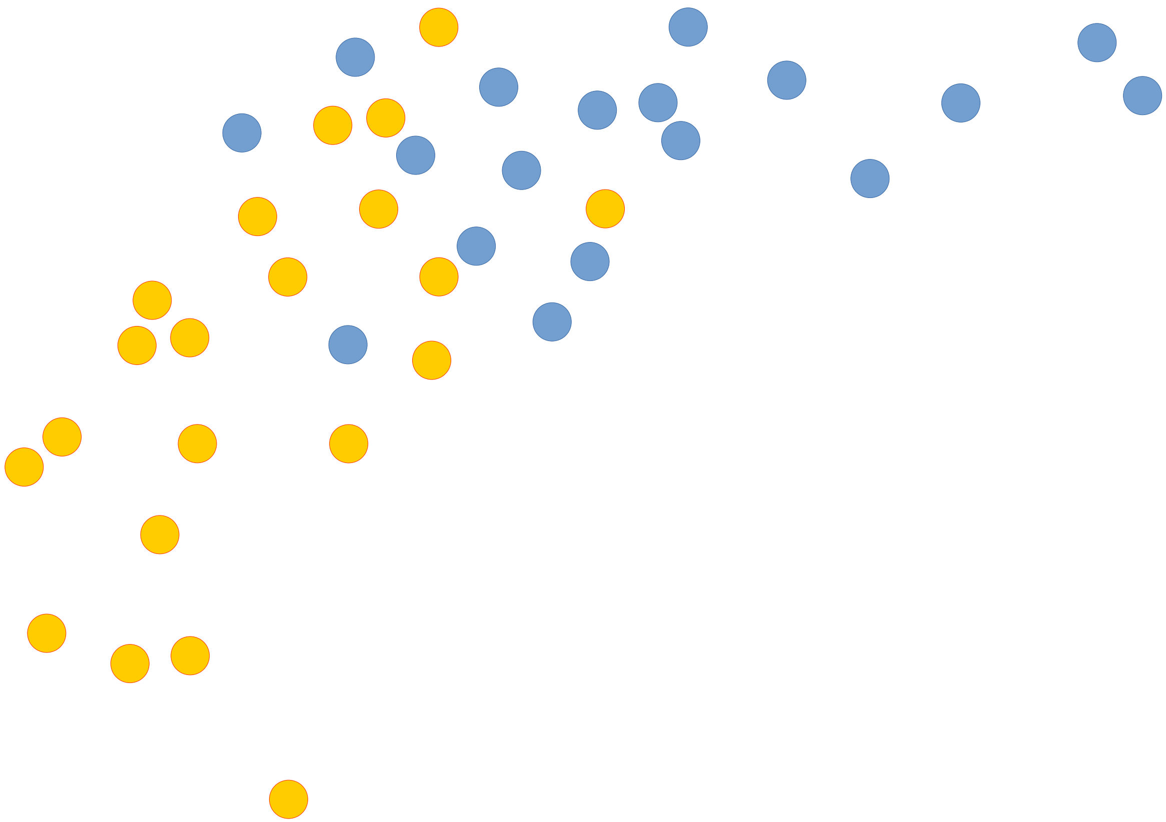
How can I explain my model's prediction?
Why did it make this decision/mistake?
What features does it rely on?

Is my model certain about what it says?
Is there something wrong with this input?
Can I rely on this prediction?

Can I trust this data?
Is something missing?
Is there any bias?

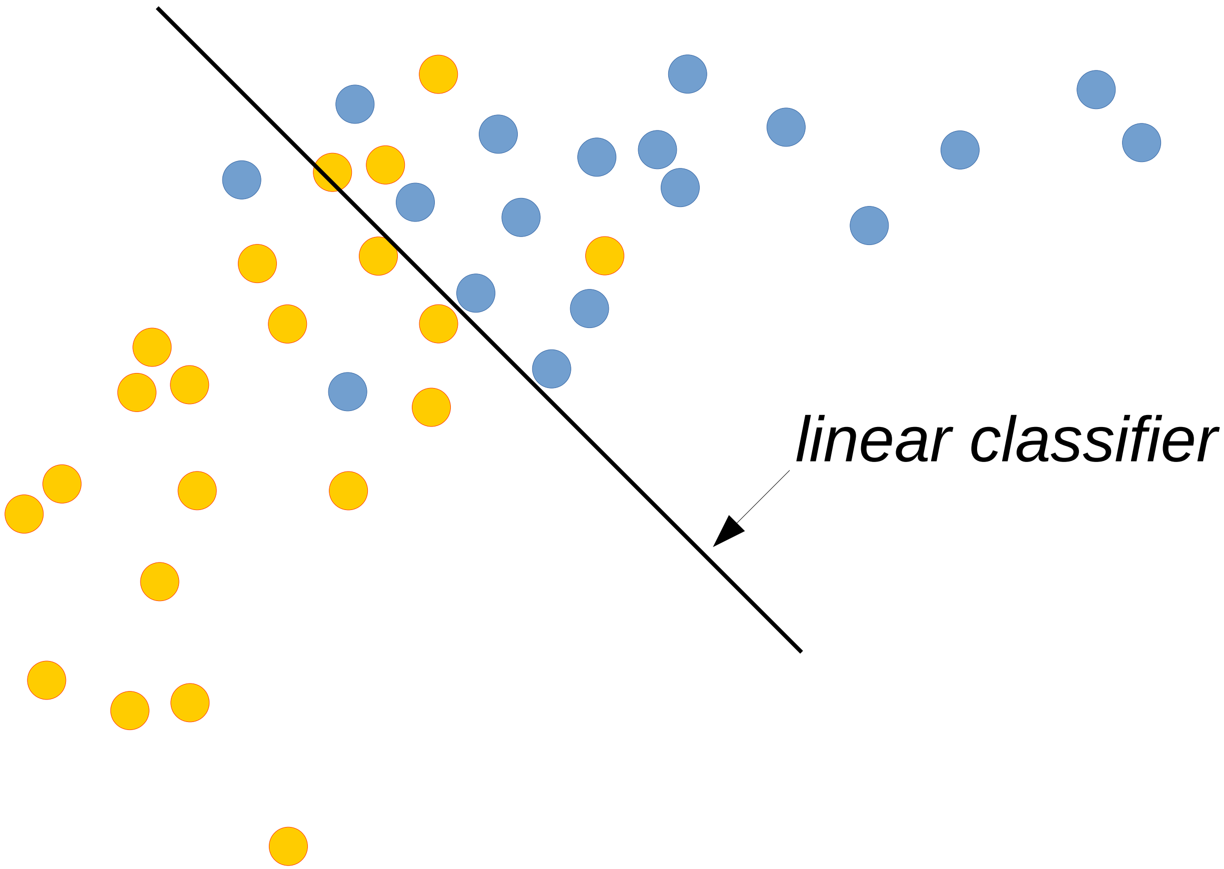
Recap: types of uncertainty

example: binary classification



Recap: types of uncertainty

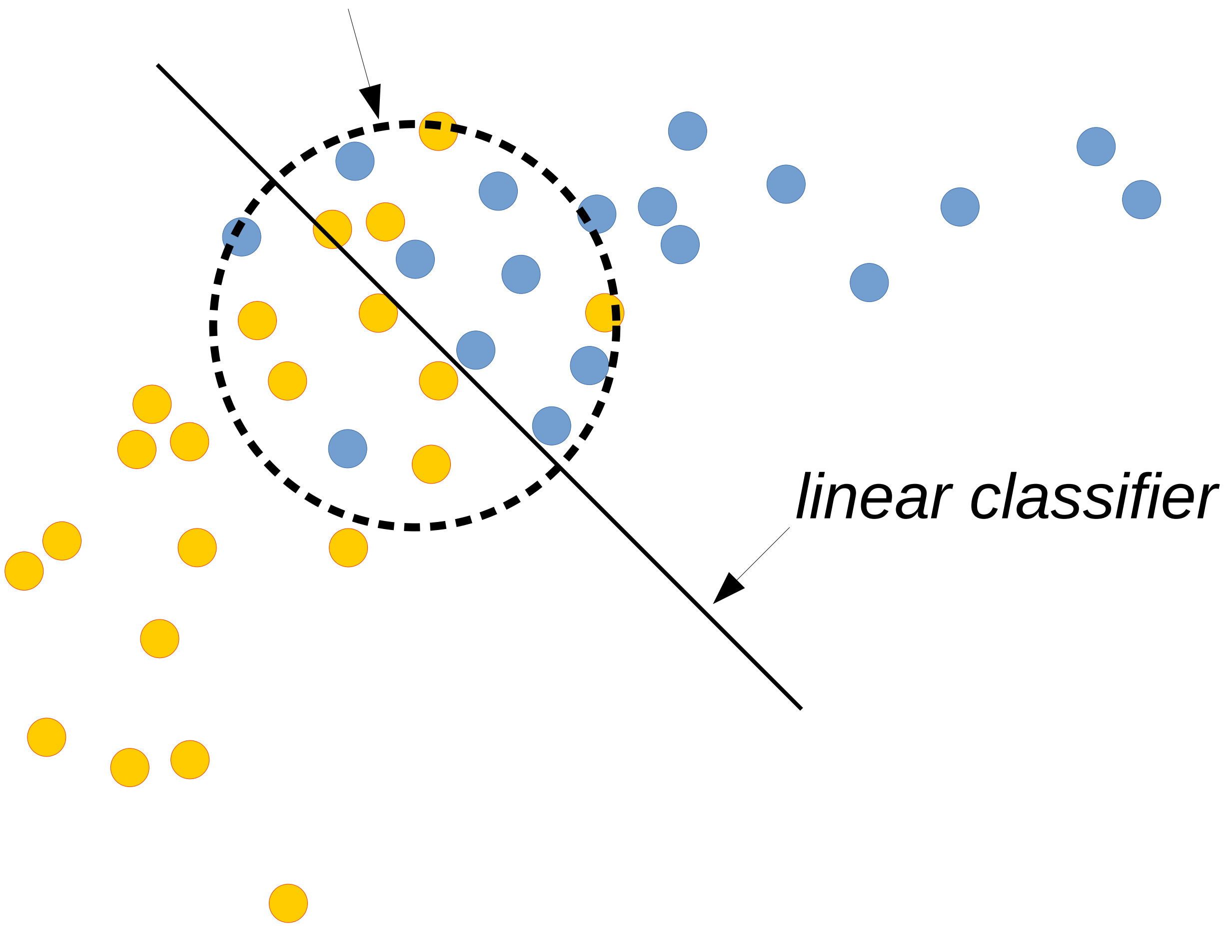
example: binary classification



Recap: types of uncertainty

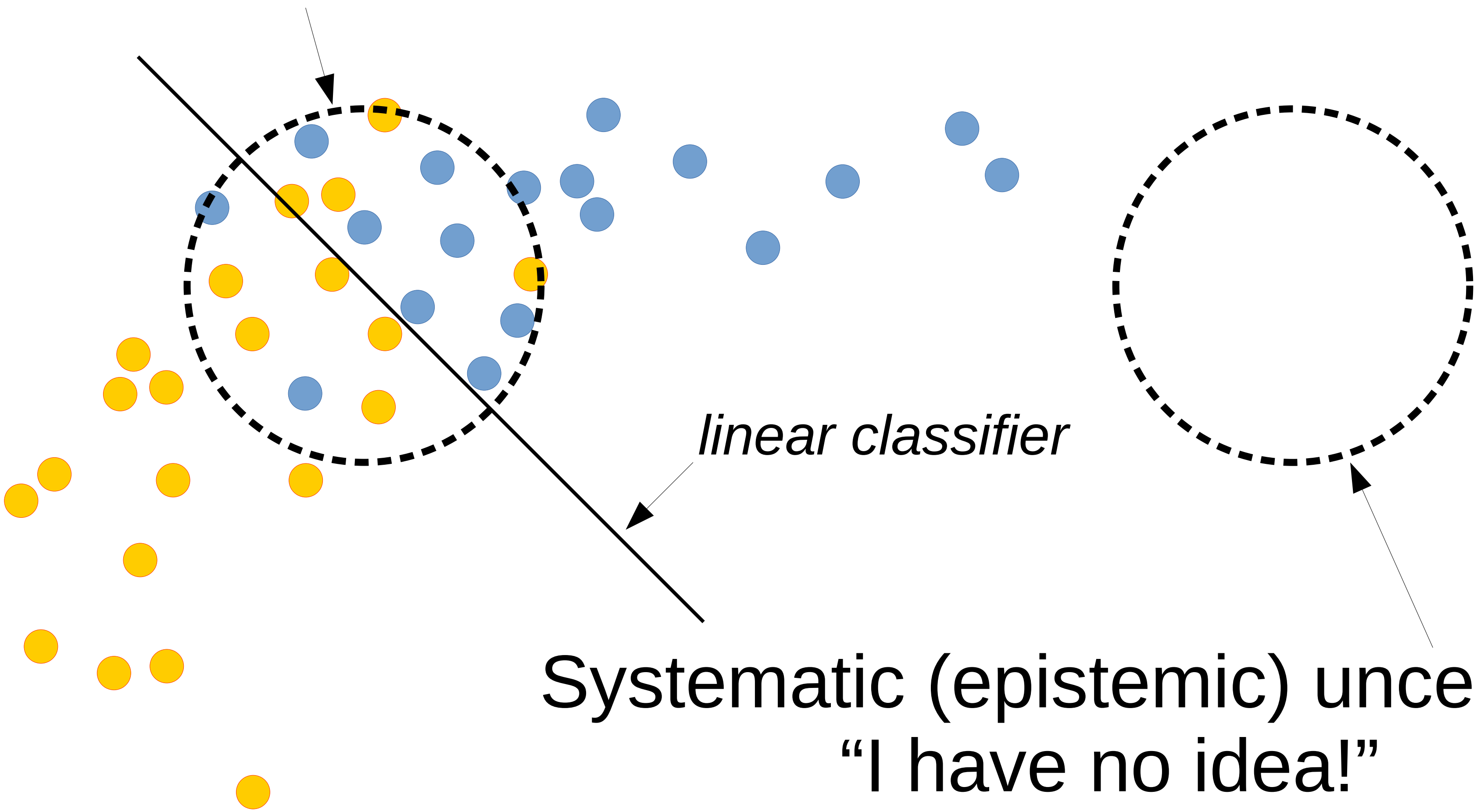
Statistical (aleatoric) uncertainty

“I know there’s randomness”



Recap: types of uncertainty

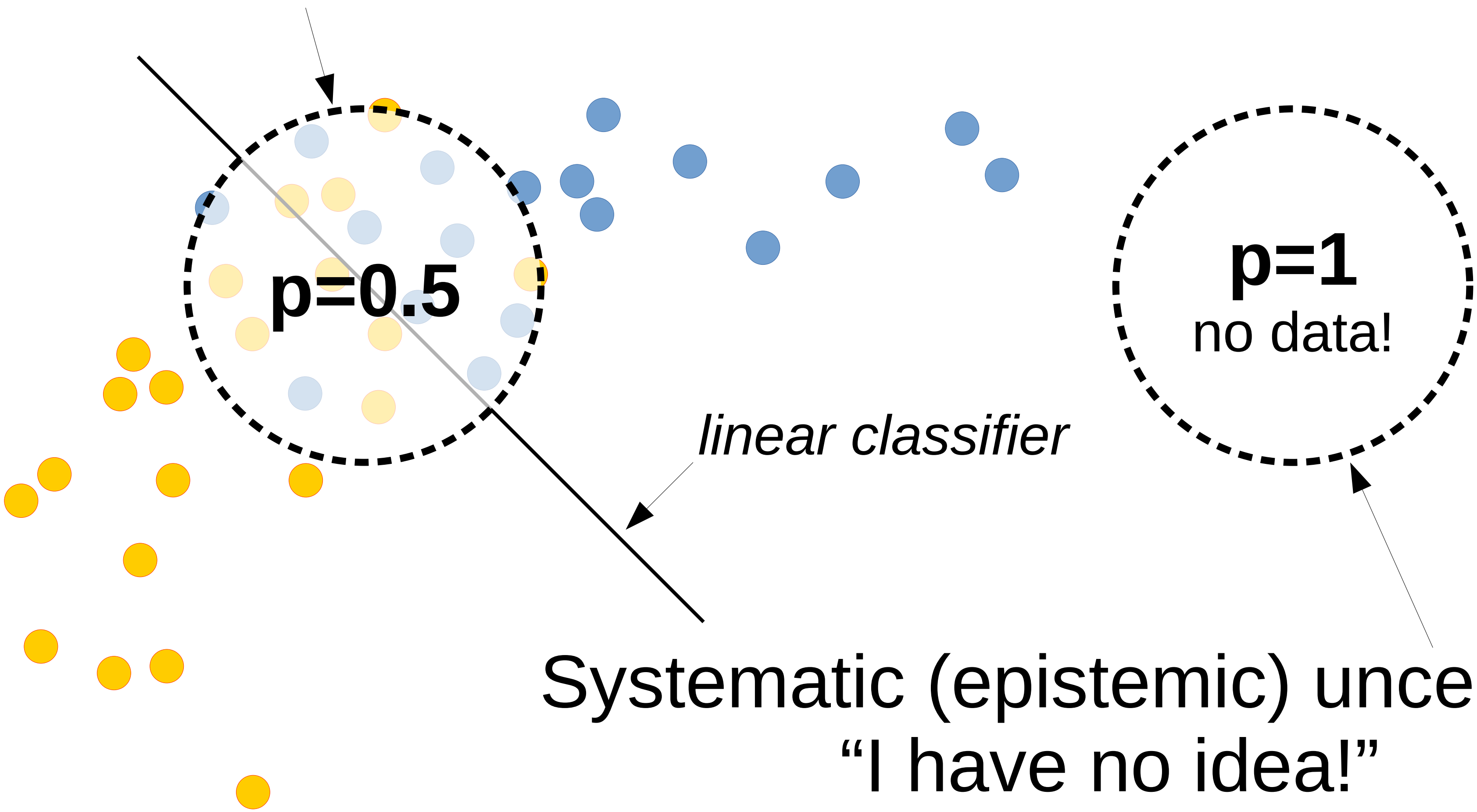
Statistical (aleatoric) uncertainty
“I know there’s randomness”



Systematic (epistemic) uncertainty
“I have no idea!”

Recap: types of uncertainty

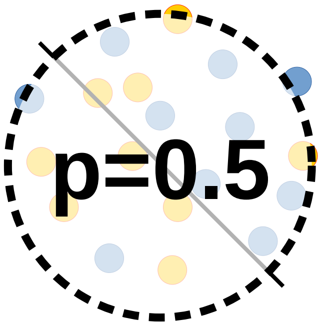
Statistical (aleatoric) uncertainty
"I know there's randomness"



Systematic (epistemic) uncertainty
"I have no idea!"

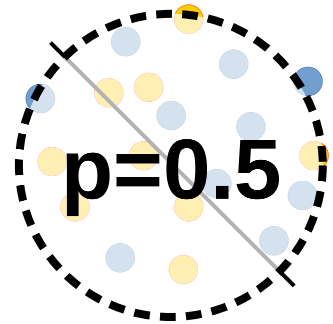
How to measure uncertainty

Aleatoric uncertainty: use predicted probability!
Exception: neural networks can be **overconfident**
Fix it by *calibrating* model predictions after the fact,
Read more: tinyurl.com/sklearn-calibration



How to measure uncertainty

Aleatoric uncertainty: use predicted probability!
Exception: neural networks can be **overconfident**
Fix it by *calibrating* model predictions after the fact,
Read more: tinyurl.com/sklearn-calibration

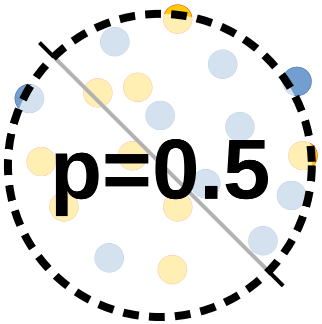


Epistemic (systematic) uncertainty: it gets tricky

Ideas?

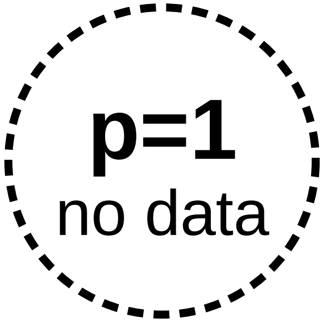
How to measure uncertainty

Aleatoric uncertainty: use predicted probability!
Exception: neural networks can be **overconfident**
Fix it by *calibrating* model predictions after the fact,
Read more: tinyurl.com/sklearn-calibration



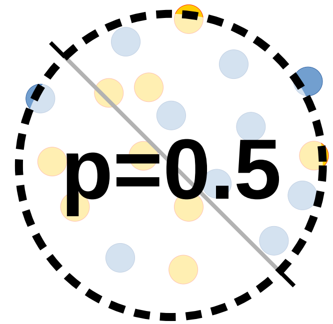
Epistemic (systematic) uncertainty: it gets tricky

Approach A: train *autoencoder* on input features
  Low reconstruction error = **certain or not?**
  High reconstruction error = **certain or not?**



How to measure uncertainty

Aleatoric uncertainty: use predicted probability!
Exception: neural networks can be **overconfident**
Fix it by *calibrating* model predictions after the fact,
Read more: tinyurl.com/sklearn-calibration

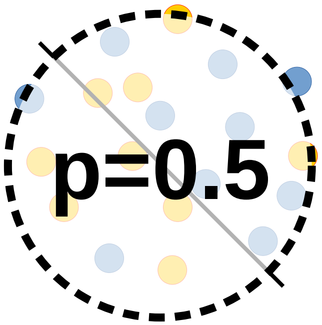


Epistemic (systematic) uncertainty: it gets tricky

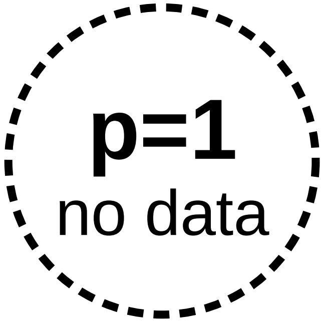
Approach A: train *autoencoder* on input features
Low reconstruction error = familiar data
High reconstruction error = unfamiliar data
(*For NLP: use language models)*

How to measure uncertainty

Aleatoric uncertainty: use predicted probability!
Exception: neural networks can be **overconfident**
Fix it by *calibrating* model predictions after the fact,
Read more: tinyurl.com/sklearn-calibration



Epistemic (systematic) uncertainty: it gets tricky



Approach A: train *autoencoder* on input features
- Low reconstruction error = familiar data
- High reconstruction error = unfamiliar data

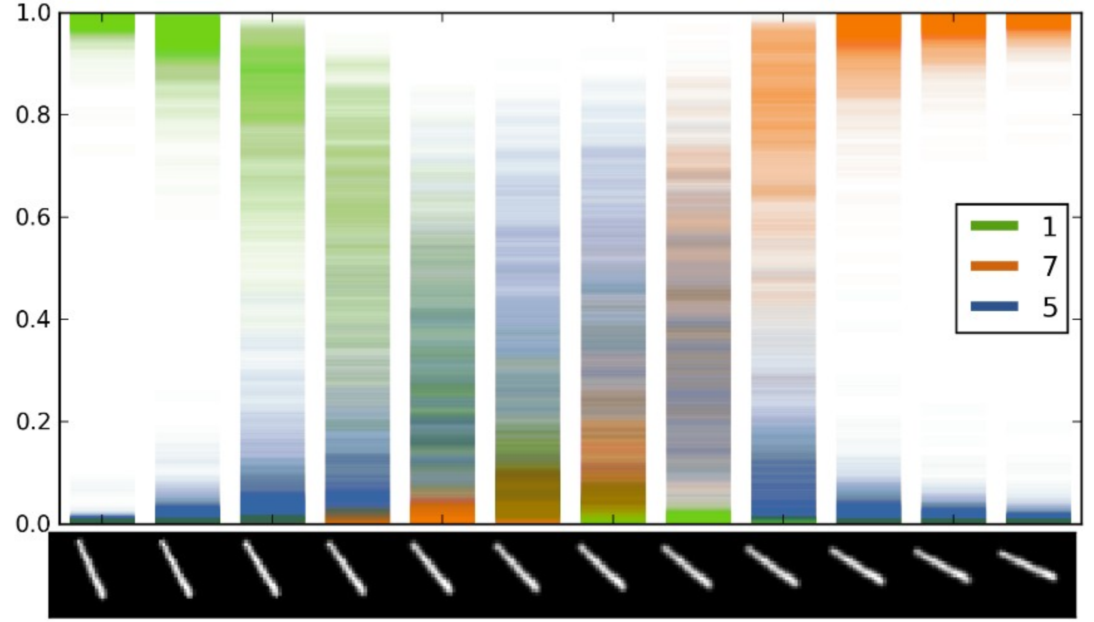
Approach B: train an *ensemble* of predictors
- Predictors agree = familiar data
- Predictors disagree = unfamiliar data

More: tinyurl.com/uncertainty-ensembles

Uncertainty from dropout

Idea:
measure how robust does your network perform under noise

Example (left):
use dropout and estimate variance



Systematic uncertainty for different input images, source: arXiv:1506.02142

Read more in the paper or in a blog post

Bayesian Neural Networks

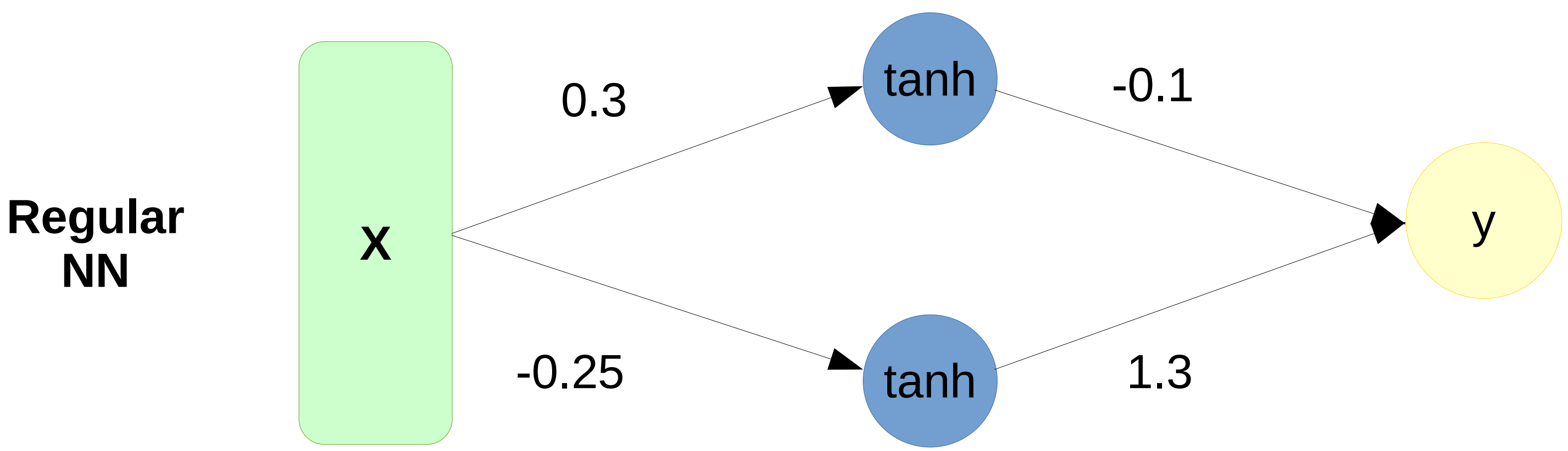
Disclaimer: this is a hacker's guide to BNNs!

It does not cover all the philosophy and general cases.

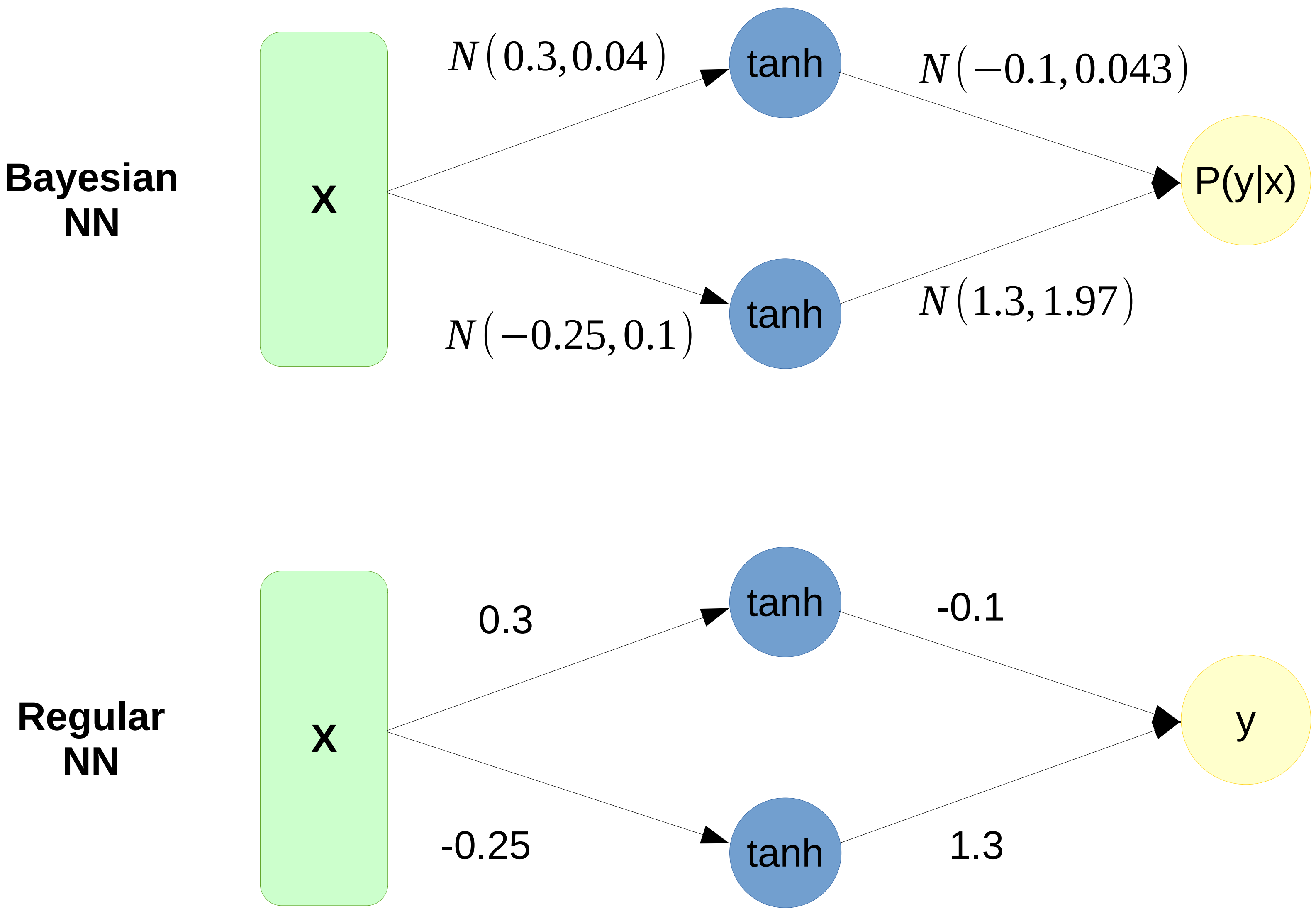
Bayesian Neural Networks

Disclaimer: this is a hacker's guide to BNNs!

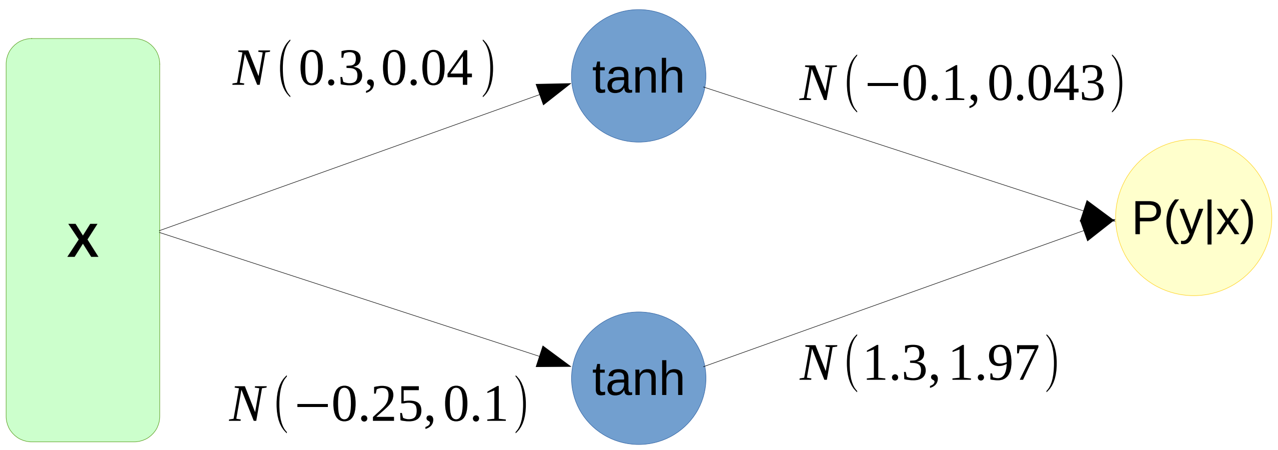
It does not cover all the philosophy and general cases.



Bayesian Neural Networks



Bayesian Neural Networks



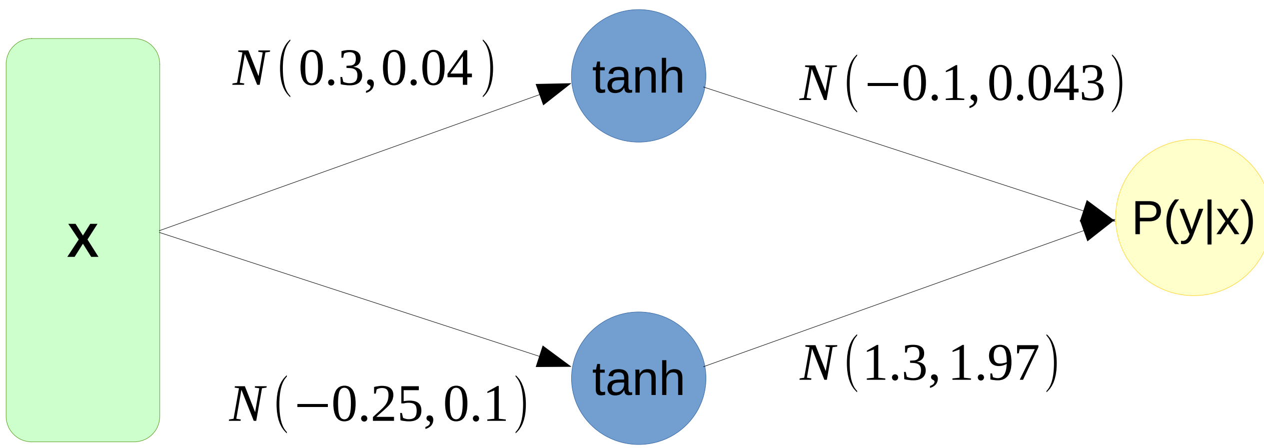
Idea:

- No explicit weights
- Maintain parametric distribution on them instead!
- Practical: fully-factorized normal or similar

$$q(\theta|\phi:[\mu,\sigma]) = \prod_i N(\theta_i|\mu_i,\sigma_i)$$

$$P(y|x) = E_{\theta\sim q(\theta|\phi)} P(y|x,\theta)$$

Bayesian Neural Networks



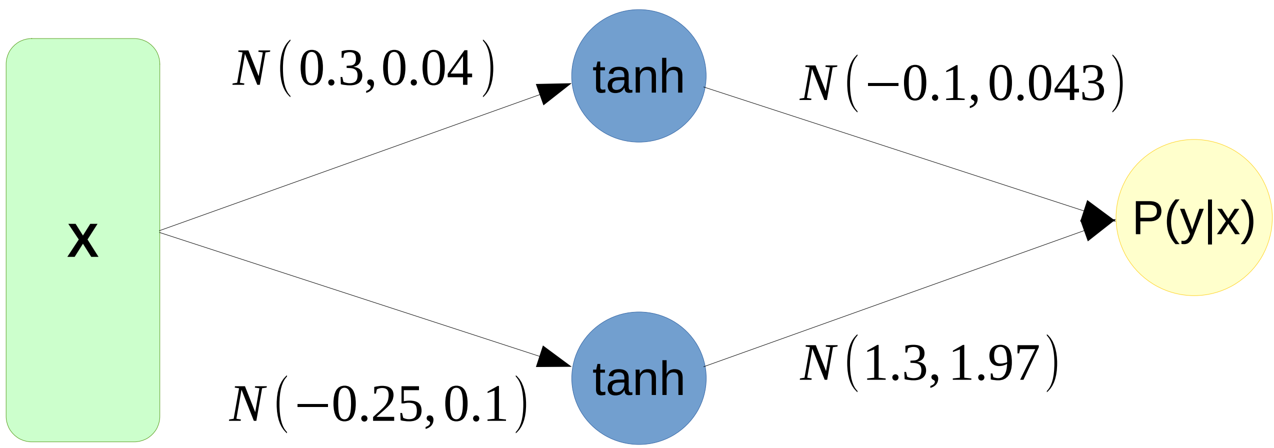
Idea:

- No explicit weights
- Maintain parametric distribution on them instead!
- Practical: fully-factorized normal or similar

$$q(\theta|\phi:[\mu,\sigma]) = \prod_i N(\theta_i|\mu_i,\sigma_i)$$

$$P(y|x) = \underline{E_{\theta\sim q(\theta|\phi)}} P(y|x,\theta)$$

Bayesian Neural Networks



Idea:

- No explicit weights
- Inference: sample from weight distributions, predict 1 “sample”
- To get distribution, aggregate K samples (e.g. with histogram)
- Yes, it means running network **multiple times per one X**

$$P(y|x) = E_{\theta \sim q(\theta|\phi)} P(y|x,\theta)$$

Bayesian Neural Networks

Idea:

- No explicit weights
- Maintain parametric distribution on them instead!
- Practical: fully-factorized normal or similar

$$q(\theta|\phi:[\mu,\sigma]) = \prod_i N(\theta_i|\mu_i,\sigma_i)$$

$$P(y|x) = E_{\theta \sim q(\theta|\phi)} P(y|x,\theta)$$

- Learn parameters of that distribution (reparameterization trick)
- Less variance: local reparameterization trick.

$$\mathring{\phi} = argmax_{\phi} E_{x_i, y_i \sim d} E_{\theta \sim q(\theta|\phi)} P(y_i|x_i,\theta)$$

wanna explicit formulae? *d = dataset*

Evidence Lower bound

d = dataset

$$-KL(q(\theta|\phi)||p(\theta|d)) = -\int_\theta q(\theta|\phi)\cdot\log\frac{q(\theta|\phi)}{p(\theta|d)}$$

$$-\int_\theta q(\theta|\phi)\cdot\log\frac{q(\theta|\phi)}{\left[\frac{p(d|\theta)\cdot p(\theta)}{p(d)}\right]} = -\int_\theta q(\theta|\phi)\cdot\log\frac{q(\theta|\phi)\cdot p(d)}{p(d|\theta)\cdot p(\theta)}$$

$$-\int_\theta q(\theta|\phi)\cdot\left[\log\frac{q(\theta|\phi)}{p(\theta)} - \log p(d|\theta) + \log p(d)\right]$$

$$\left[E_{\theta\sim q(\theta|\phi)}\log p(d|\theta)\right] - KL(q(\theta|\phi)||p(\theta)) + \log p(d)$$

loglikelihood **-distance to prior** **+const**

Evidence Lower bound

$$\phi = \underset{\phi}{argmax}\left(-KL\left(q\left(\theta|\phi\right)\|p\left(\theta|d\right)\right)\right)$$

$$\underset{\phi}{argmax}\left(\underbrace{\left[E_{\theta\sim q(\theta|\phi)}\log p\left(d|\theta\right)\right]}_{\textbf{fit to the data}}-KL\underbrace{\left(q\left(\theta|\phi\right)\|p\left(\theta\right)\right)}_{\textbf{don't be too certain}}\right)$$

Evidence Lower bound

$$\phi = \underset{\phi}{argmax}\left(-KL\left(q(\theta|\phi)\|p(\theta|d)\right)\right)$$

$$\underset{\phi}{argmax}\left(\left[E_{\theta\sim q(\theta|\phi)}\log p(d|\theta)\right]-KL\left(q(\theta|\phi)\|p(\theta)\right)\right)$$

Can we perform gradient ascent directly?

Reparameterization trick

$$\phi = \underset{\phi}{argmax}\left(-KL\left(q(\theta|\phi)\|p(\theta|d)\right)\right)$$

$$\underset{\phi}{argmax}\left(\underbrace{\left[E_{\theta\sim q(\theta|\phi)}\log p(d|\theta)\right]}_{\text{Use reparameterization trick}} - \underbrace{KL\left(q(\theta|\phi)\|p(\theta)\right)}_{\text{simple formula (for normal q)}}\right)$$

Use reparameterization trick

simple formula
(for normal q)

What does this log
P(d|...) mean?

BNN likelihood

$$E_{\theta\sim N(\theta|\mu_\phi,\sigma_\phi)}\log p(d|\theta) = E_{\psi\sim N(0,1)}\log p\left(d|\left(\mu_\phi+\sigma_\phi\cdot\psi\right)\right)$$

Reparameterization trick

$$\phi = \underset{\phi}{argmax}\left(-KL\left(q(\theta|\phi)\|p(\theta|d)\right)\right)$$

$$\underset{\phi}{argmax}\left(\left[E_{\theta\sim q(\theta|\phi)}\log p(d|\theta)\right]-KL\left(q(\theta|\phi)\|p(\theta)\right)\right)$$

In other words,
$\Sigma_{x,y\sim d}$ log $p(y|x,\mu+\sigma\psi)$

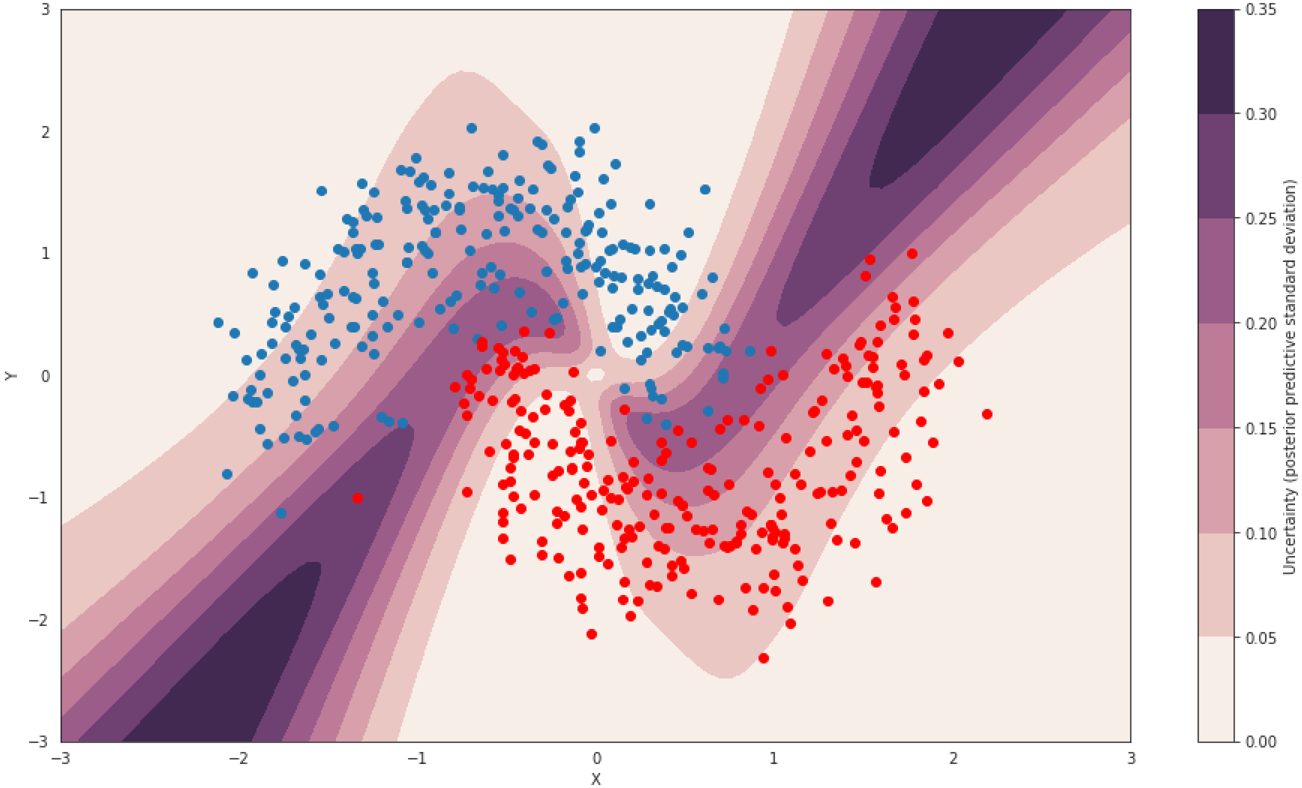
BNN likelihood

$$E_{\theta\sim N(\theta|\mu_\phi,\sigma_\phi)}\log p(d|\theta)=E_{\psi\sim N(0,1)}\log p\left(d|\left(\mu_\phi+\sigma_\phi\cdot\psi\right)\right)$$

Bayesian Neural Networks

Estimating uncertainty:

1. sample weights several times
2. predict by averaging outputs
3. uncertainty = standard deviation



Read more...

Papers on uncertainty

bayesian neural networks: blog post
prior networks: arxiv.org/abs/1802.10501
batchnorm: arxiv.org/abs/1802.04893
dropout: arxiv.org/abs/1506.02142
video stuff: youtube.com/watch?v=HRfDiqgh6CE

The question of trust

How can I explain my model's prediction?
Why did it make this decision/mistake?
What features does it rely on?

Is my model certain about what it says?
Is there something wrong with this input?
Can I rely on this prediction?

Can I trust this data?
Is something missing?
Is there any bias?

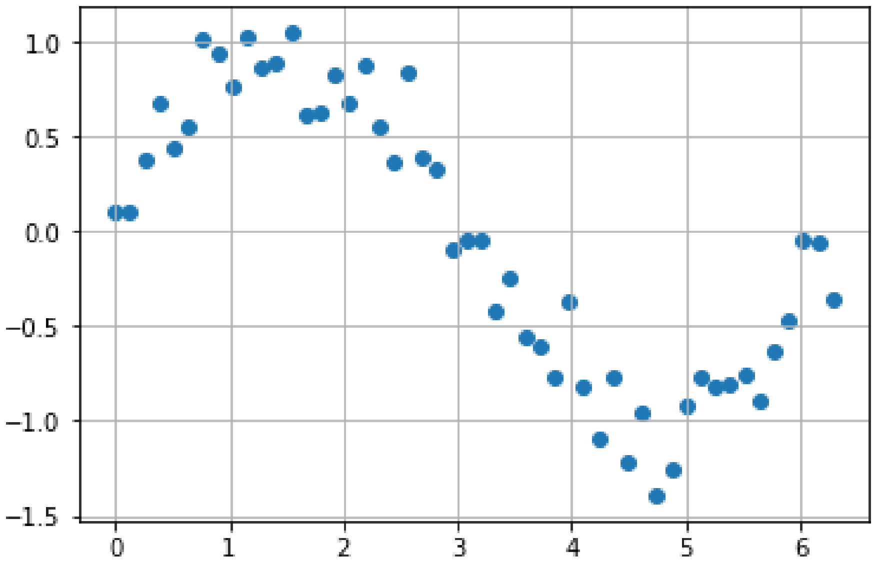
Exploratory data analysis

aka “seeing for yourself what’s in your data”

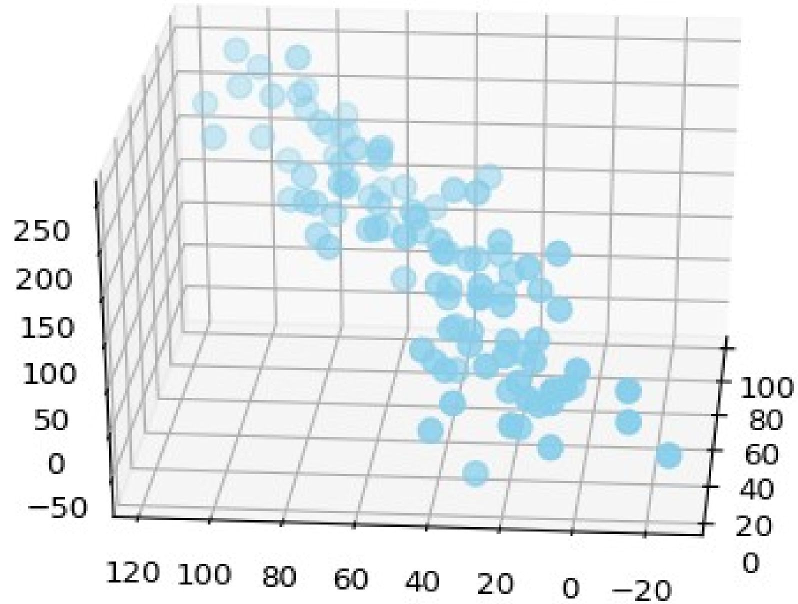
Q: How many dimensions can you show on a plot?

Exploratory data analysis

Q: How many dimensions can you show on a plot?



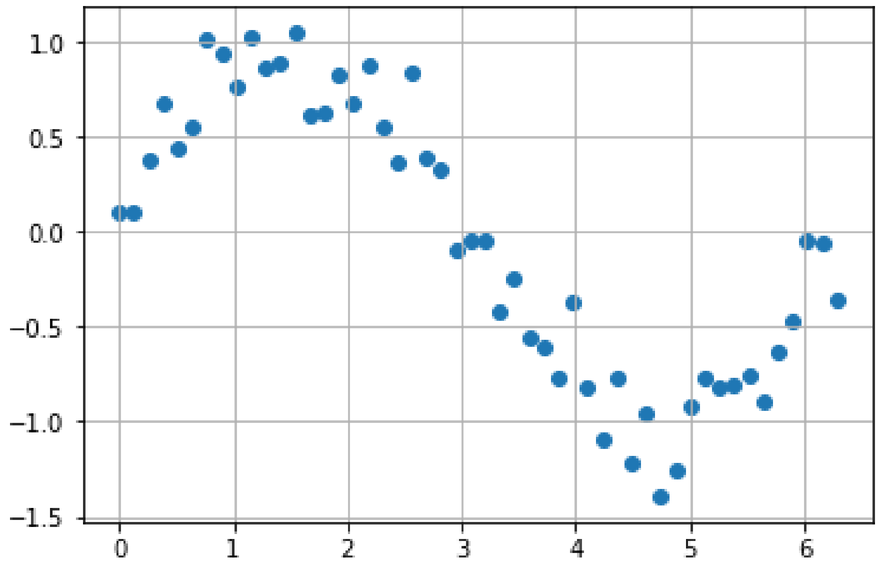
2d scatter-plot



3d scatter-plot

Exploratory data analysis

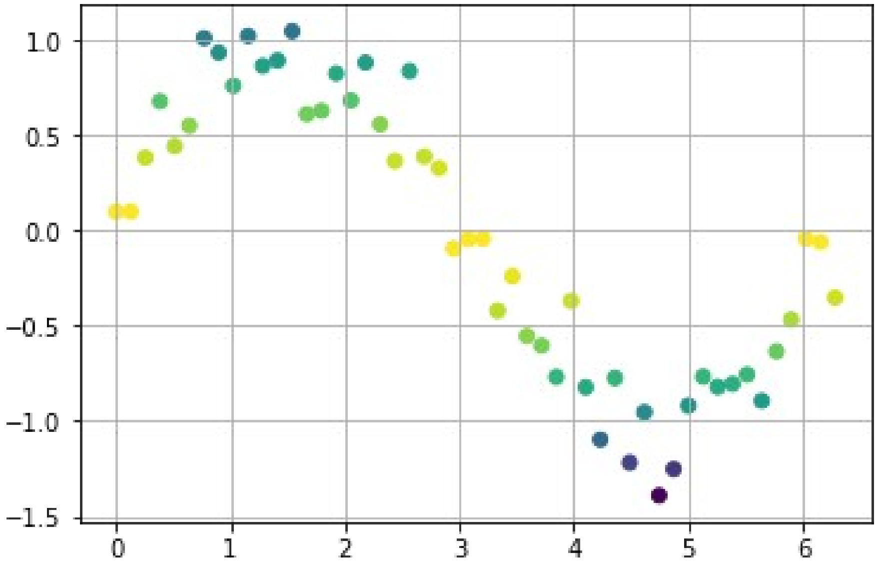
Q: How many dimensions can you show on a plot?



2 dimensions

Exploratory data analysis

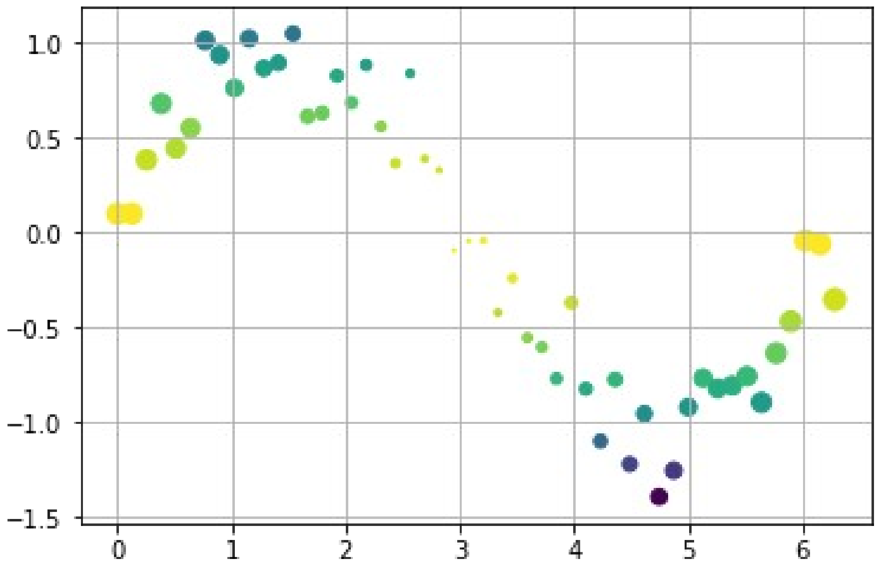
Q: How many dimensions can you show on a plot?



3 dimensions

Exploratory data analysis

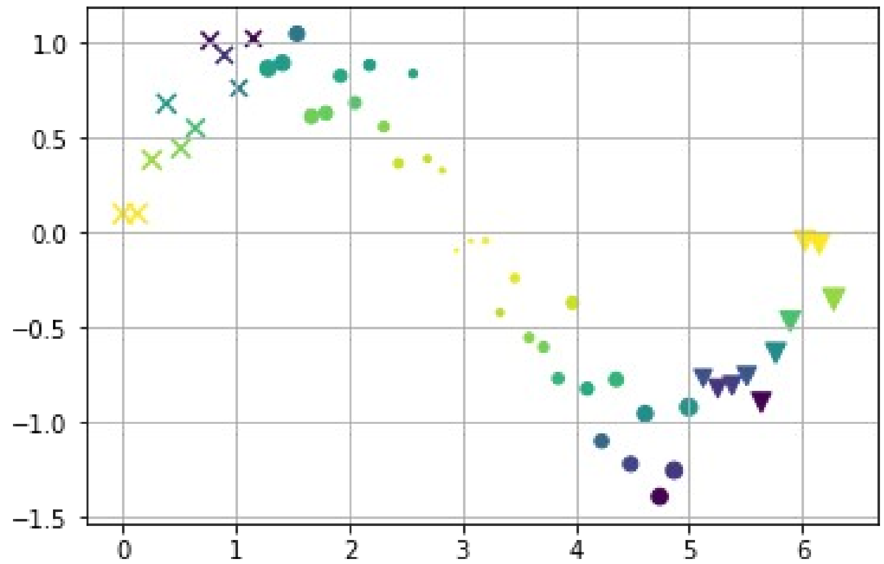
Q: How many dimensions can you show on a plot?



4 dimensions

Exploratory data analysis

Q: How many dimensions can you show on a plot?



5 dimensions

Exploratory data analysis

Q: How many dimensions can you show on a plot?

Your data has 200 dimensions...
any ideas?

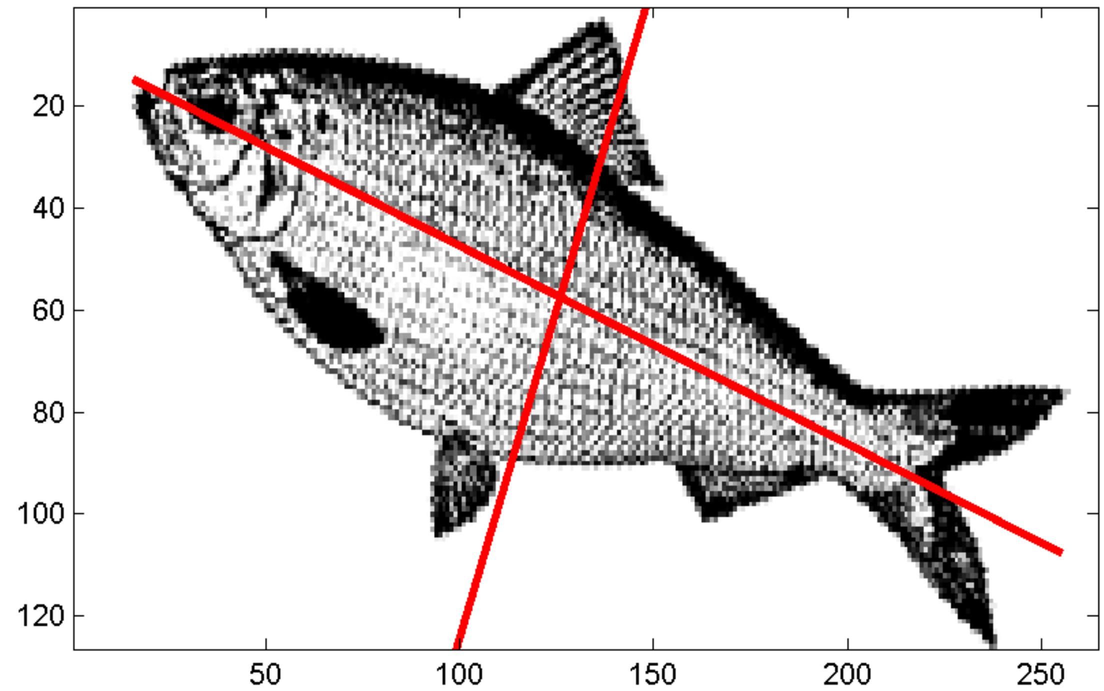
Recap: Principal Component Analysis

Idea:

- Linearly project data to lower-dim space

$$X \approx (X \times W_1) \times W_2$$

Minimize MSE



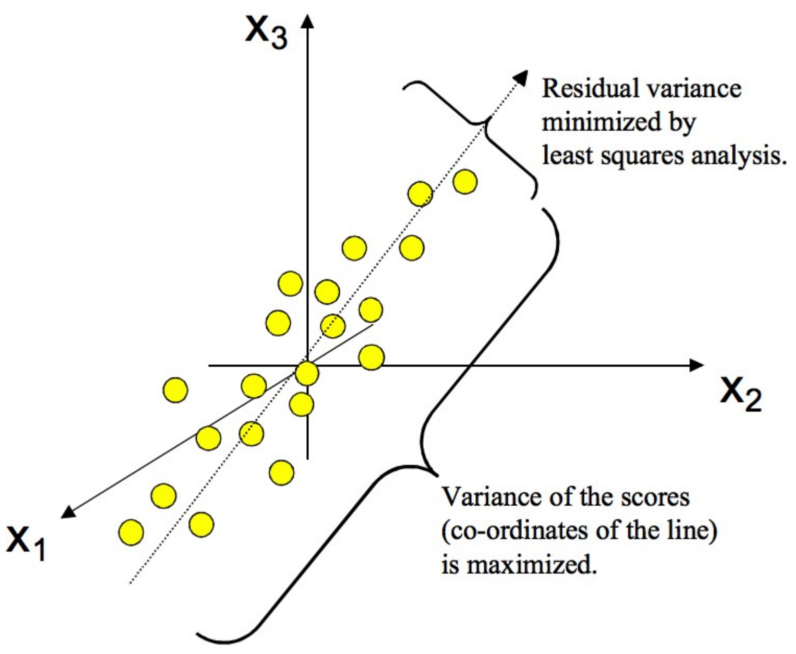
$$argmin_{W_1, W_2} \| X - (X \times W_1) \times W_2 \|$$

Recap: Principal Component Analysis

Idea:

- Linearly project data to lower-dim space

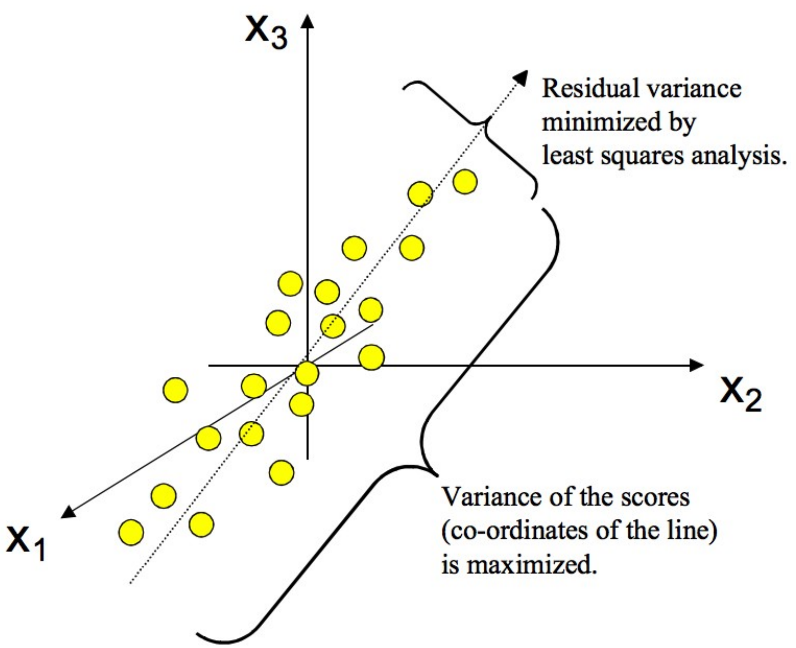
- Attempt to preserve as much variance as possible



Recap: Principal Component Analysis

Idea:

- Linearly project data to lower-dim space
- Attempt to preserve as much variance as possible

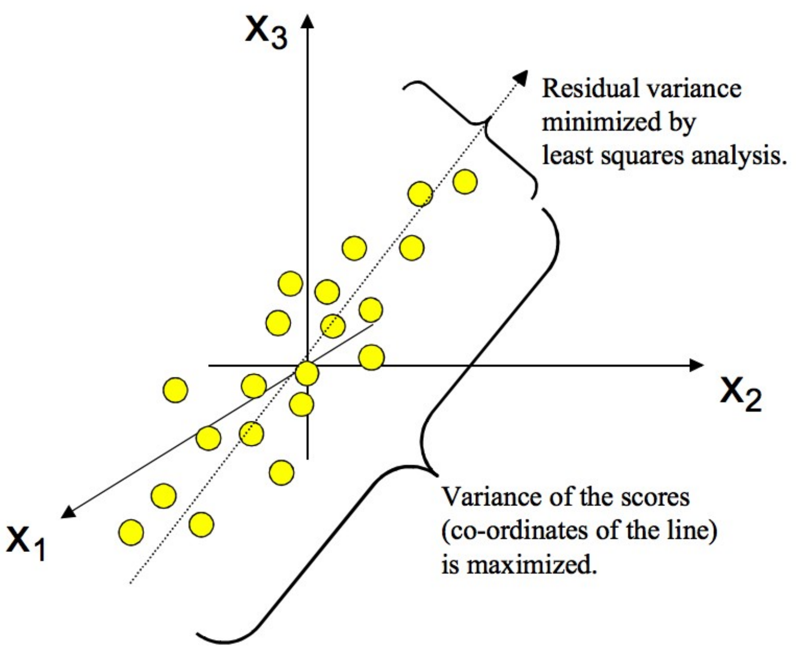


Q: What if linear projection is not enough?

Recap: Principal Component Analysis

Idea:

- Linearly project data to lower-dim space
- Attempt to preserve as much variance as possible



Q: What if linear projection is not enough?

deep autoencoders… or better

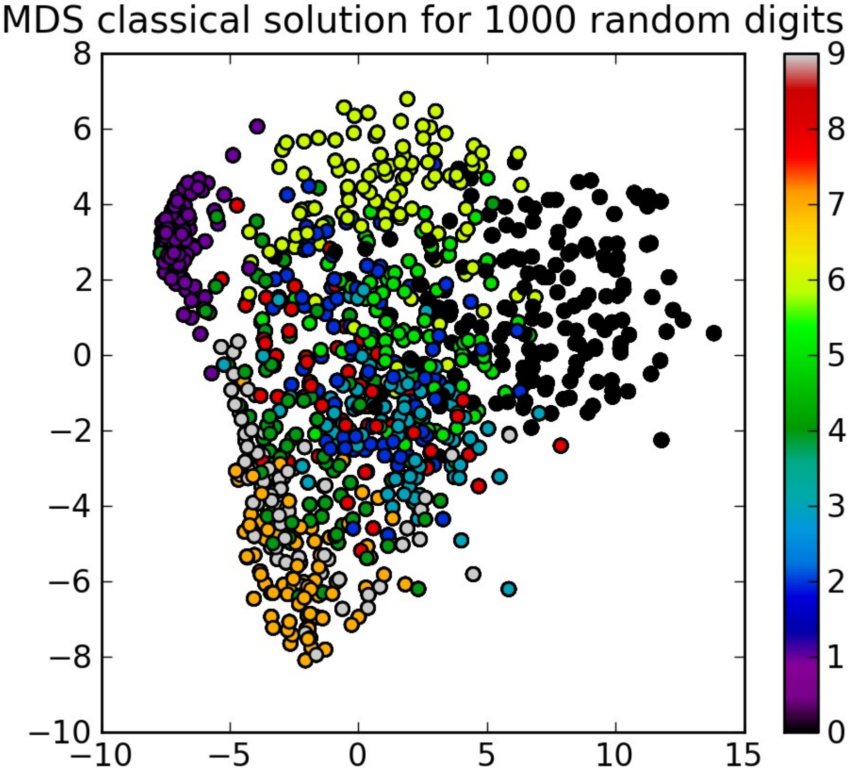
Manifold learning

Idea: let’s directly “learn” 2d point coordinates

Multidimensional Scaling

try preserving pairwise distances

$$\hat{x} = argmin_{\hat{x}} \frac{2}{N^2 - N} \sum_{i \neq j} \left(\|x_i - x_j\| - \|\hat{x}_i - \hat{x}_j\| \right)^2$$



Stochastic Neighborhood Embedding

try preserving neighbor "probabilities"

$$P_{j|i} = \frac{e^{-\|x_i - x_j\|_2^2}}{\sum_k e^{-\|x_k - x_j\|_2^2}}$$

- large for nearest neighbors
- small for distant points
- adds up to 1

$$\hat{P}_{j|i} = \frac{e^{-\|\hat{x}_i - \hat{x}_j\|_2^2}}{\sum_k e^{-\|\hat{x}_k - \hat{x}_j\|_2^2}}$$

- same as P
- but in learned space

optimize crossentropy w.r.t. $\hat{x}$

$$\hat{x} = argmin_{\hat{x}} - \frac{1}{N} \sum_i \sum_j P_{j|i} \cdot \log \hat{P}_{j|i}$$

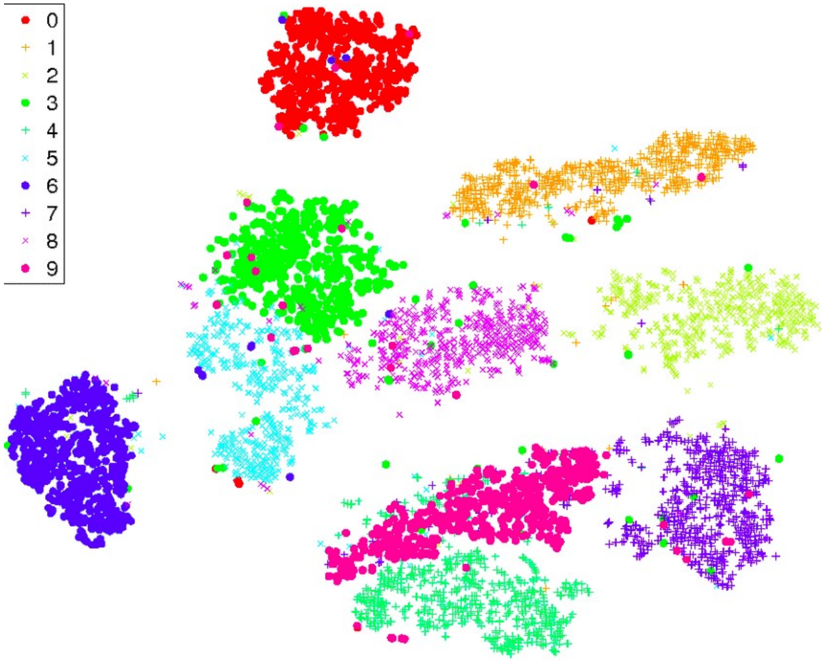
T-SNE

Like SNE from prev slide, but

- P is now *Student's t-distribution*

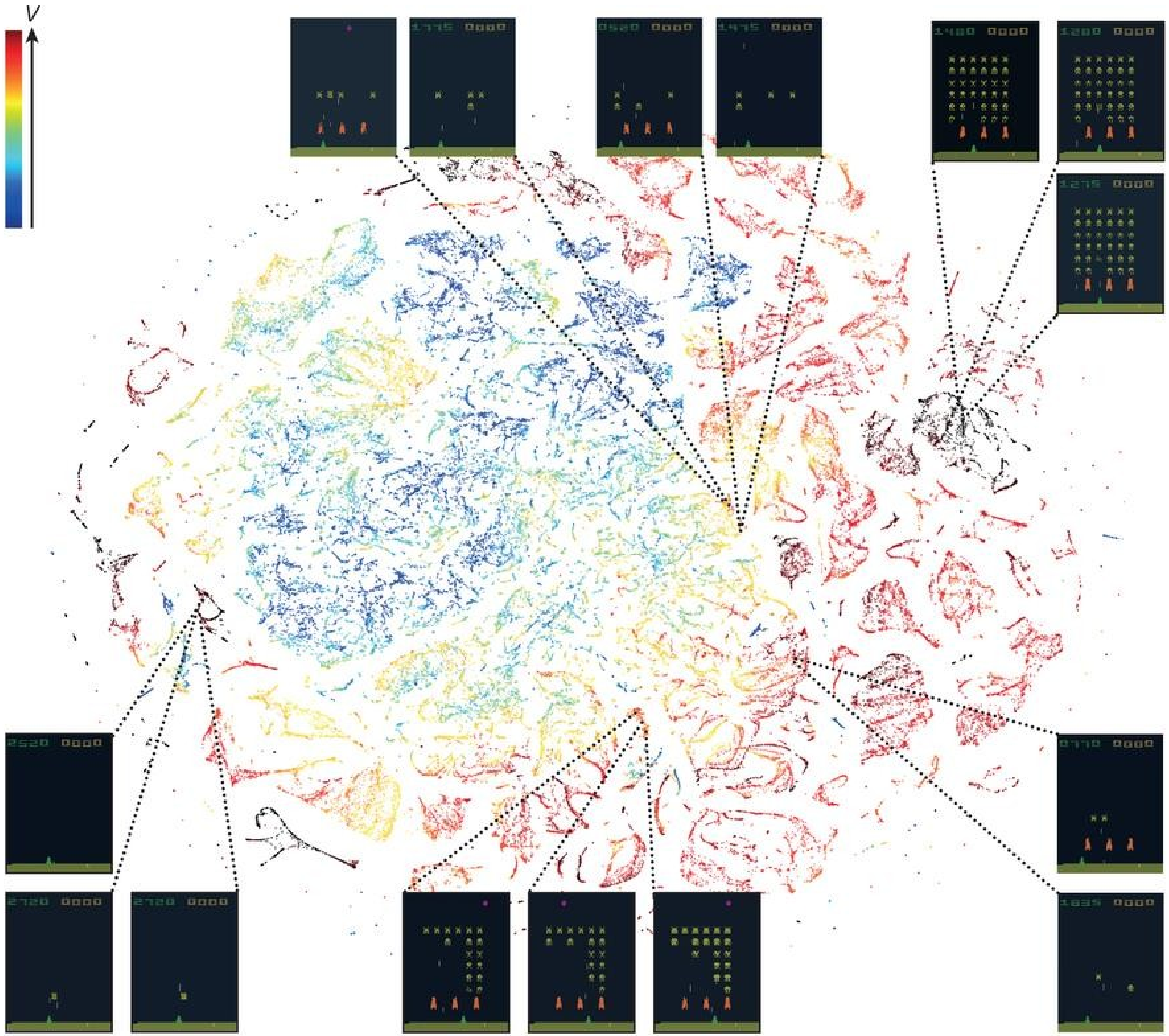
$$\hat{P}_{j|i} = \frac{\left(1 + \|\hat{x}_i - \hat{x}_j\|_2^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|\hat{x}_k - \hat{x}_l\|_2^2\right)^{-1}}$$

- A lot of optimization hacks
- By far the most popular method

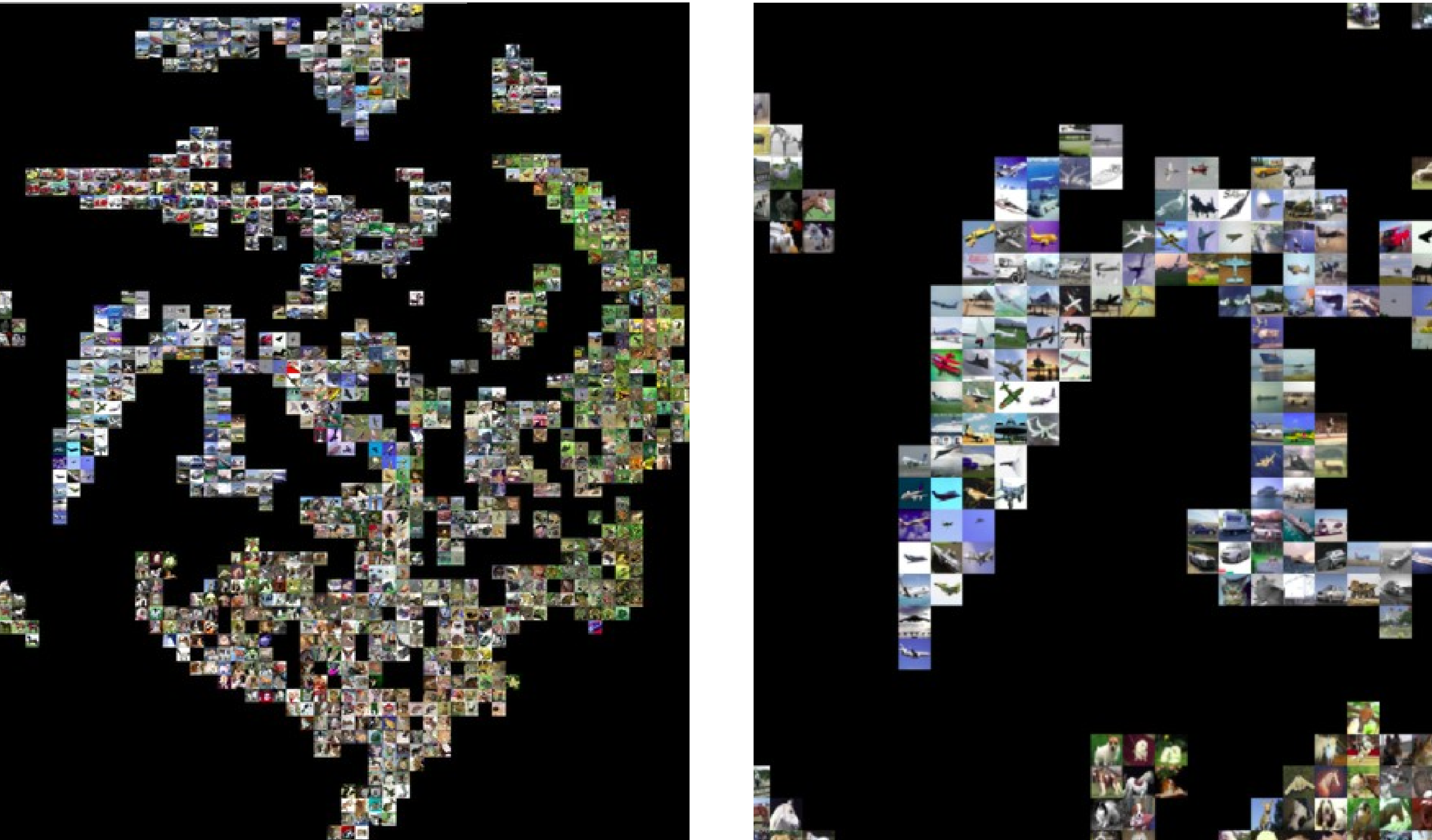


Read More:
Original paper
Interactive demo

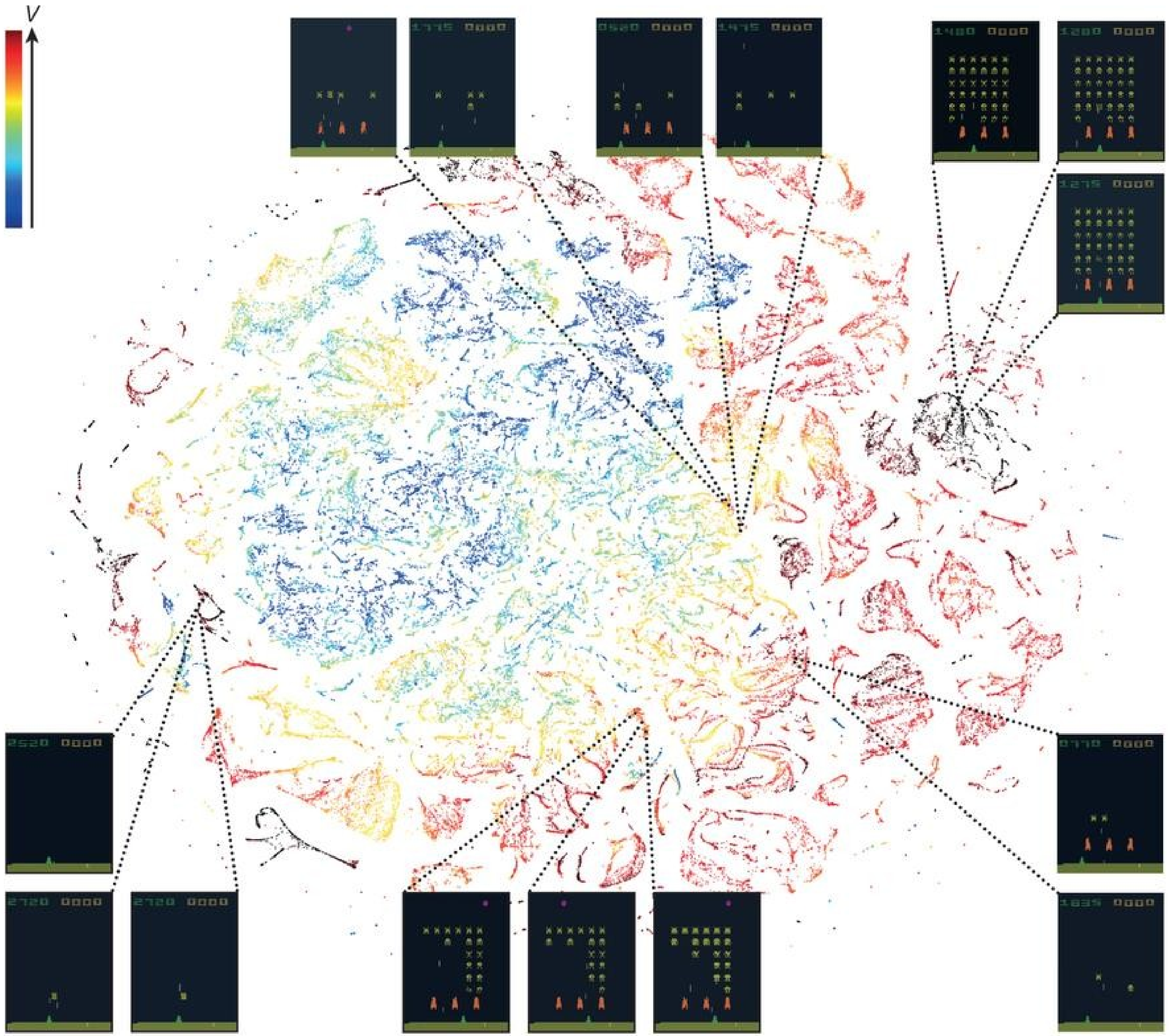
T-SNE + deep encoder



T-SNE + deep encoder (CIFAR10)



T-SNE + deep encoder (atari DQN)



Thank you

[question time!]