GANs & Statistical Tests

Intro to ML

Fall 2022

Administrivia

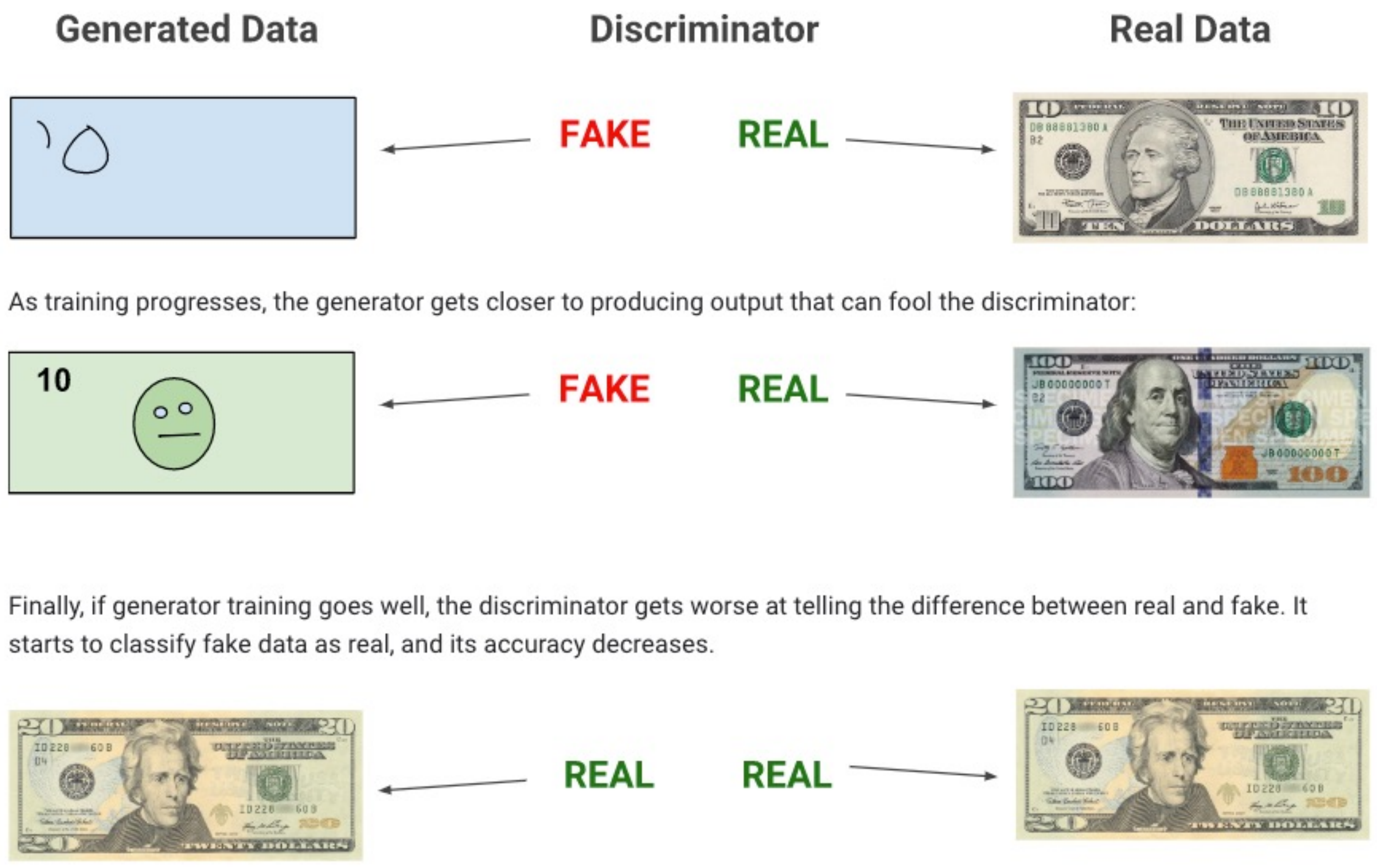
- You may have 1 cheat sheet
  - 1-sided
  - 8.5x11 inches (normal letter page size)
- I will release some example exam questions on eclass
- No class Thursday (I am at a conference)

GAN

- Generative Adversarial Network

- Two dueling neural networks
  - One trained to generate images
  - One trained to distinguish generated images from true images

GAN

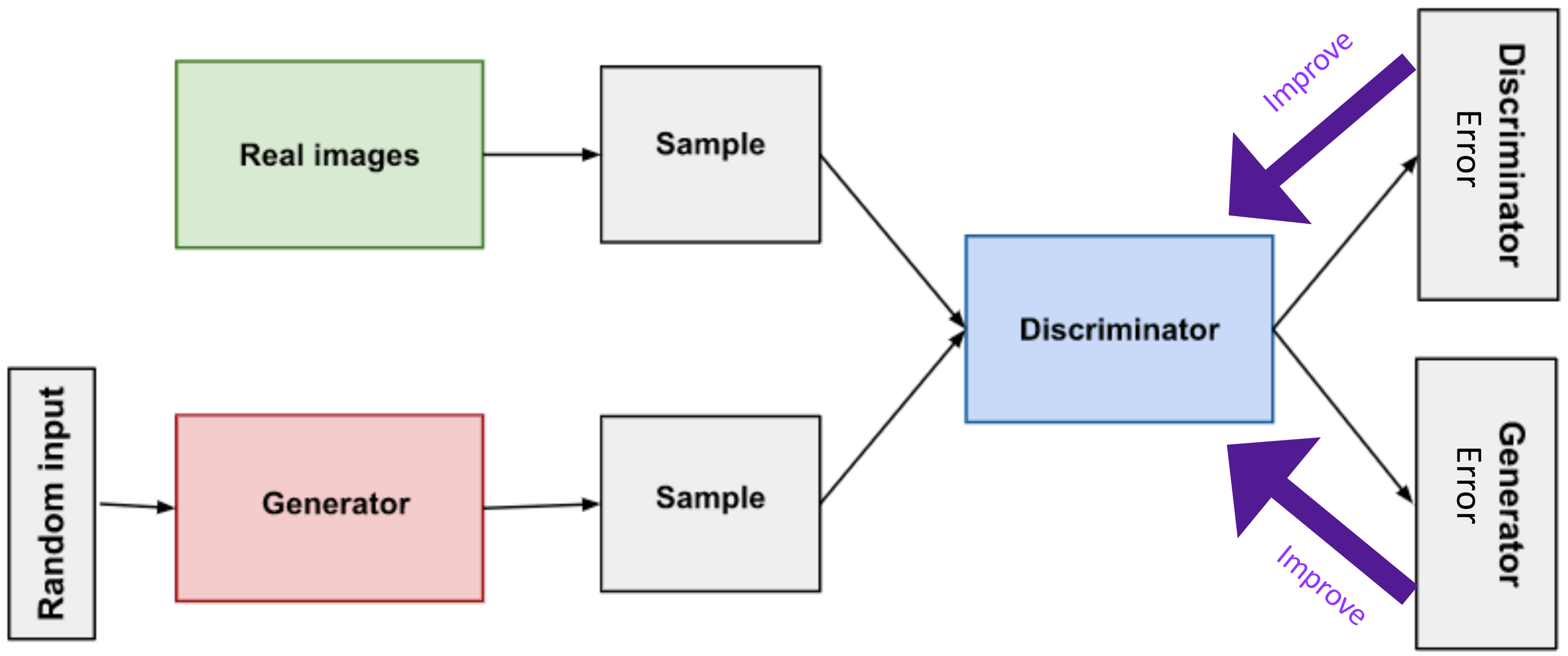


As training progresses, the generator gets closer to producing output that can fool the discriminator:

Finally, if generator training goes well, the discriminator gets worse at telling the difference between real and fake. It starts to classify fake data as real, and its accuracy decreases.

https://developers.google.com/machine-learning/gan/gan_structure

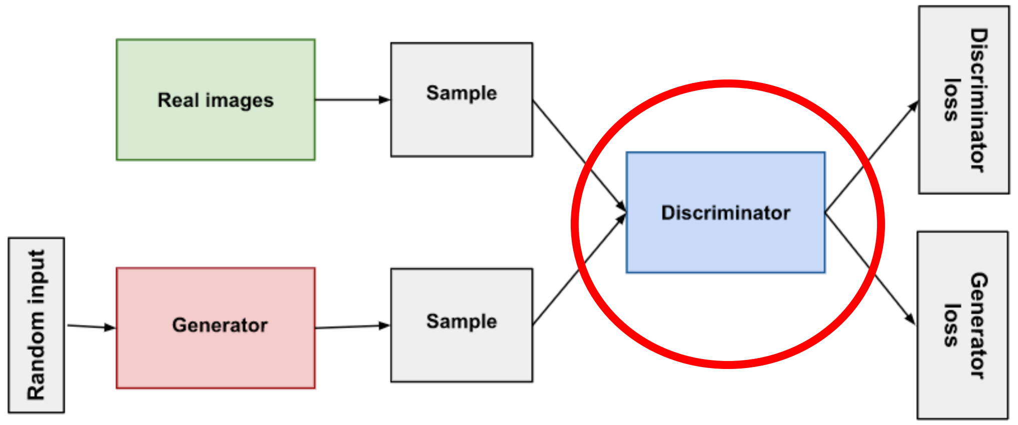
GAN Learning



https://developers.google.com/machine-learning/gan/gan_structure

What is the Discriminator trying to do?

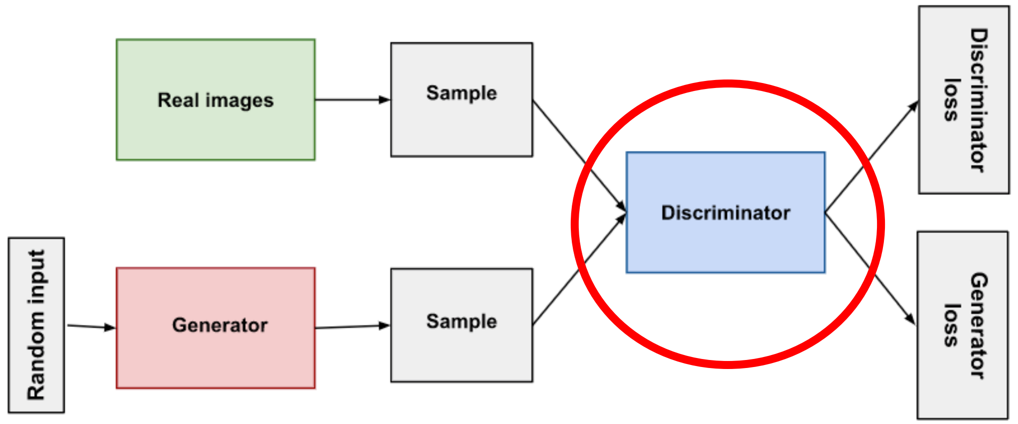
- Let's say
  - x is a true sample
  - z is the generated sample
  - Discriminator is a function D(.)
  - D(.) is a probability [0,1]



- The Discriminator wants to learn D such that
  - D(x) is high
  - D(z) is low

What is the Generator trying to do?

- Let’s say
  - x is a true sample
  - z is the generated sample
  - Discriminator is a function D(.) in [0,1]
  - Generator is a function z = G(random)



- The Generator wants to learn D such that
  - D(x) is low
  - D(z) = D(G(random)) is high

Put it together

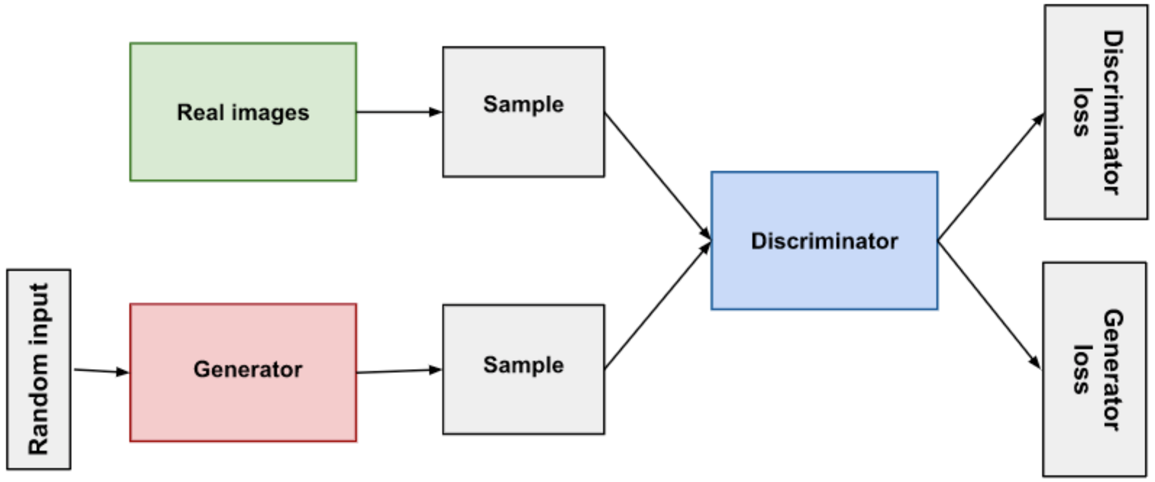
- The Discriminator wants to learn D such that
  - D(x) is high
  - D(z) is low

  - Maximize D(x) + (1 - D(G(random)))

- The Generator wants to learn D such that
  - D(x) is low
  - D(z) = D(G(random)) is high

  - Minimize D(x) + (1 - D(G(random)))

Minimax game



- Generator creates some random sample z = G(random)
- Discriminator computes D(x), D(z)

$$\min_{\theta_G} \max_{\theta_D} D(x) + (1 - D(G(\text{random})))$$

- Updates to θ_D depend on θ_G via z
- Updates to θ_G depend on θ_D because of chain rule in derivative

Under the hood

- Discriminator network takes as input an image and produces as output the probability that it came from the real dataset
  - Basically, this is a classifier
  - Can use a simple MLP, or CNN structures

Under the hood

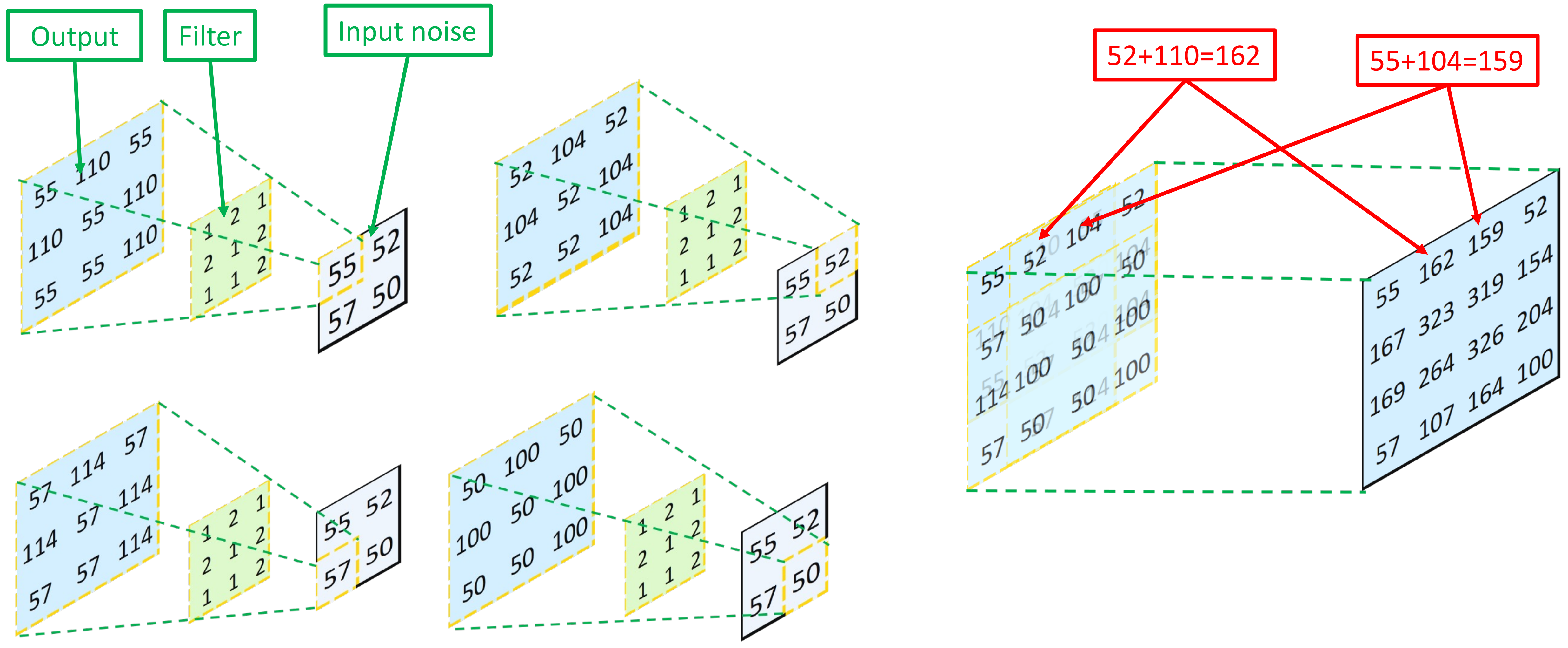
- Generator network is a bit more complex
  - Takes as input a noise vector
  - Produces as output an image i.e. an m*m matrix

- How can we go noise → image?

Under the hood: Generator

- Generators use deconvolution (Transposed convolution)
- Use learned filters to essentially invert the convolutional process

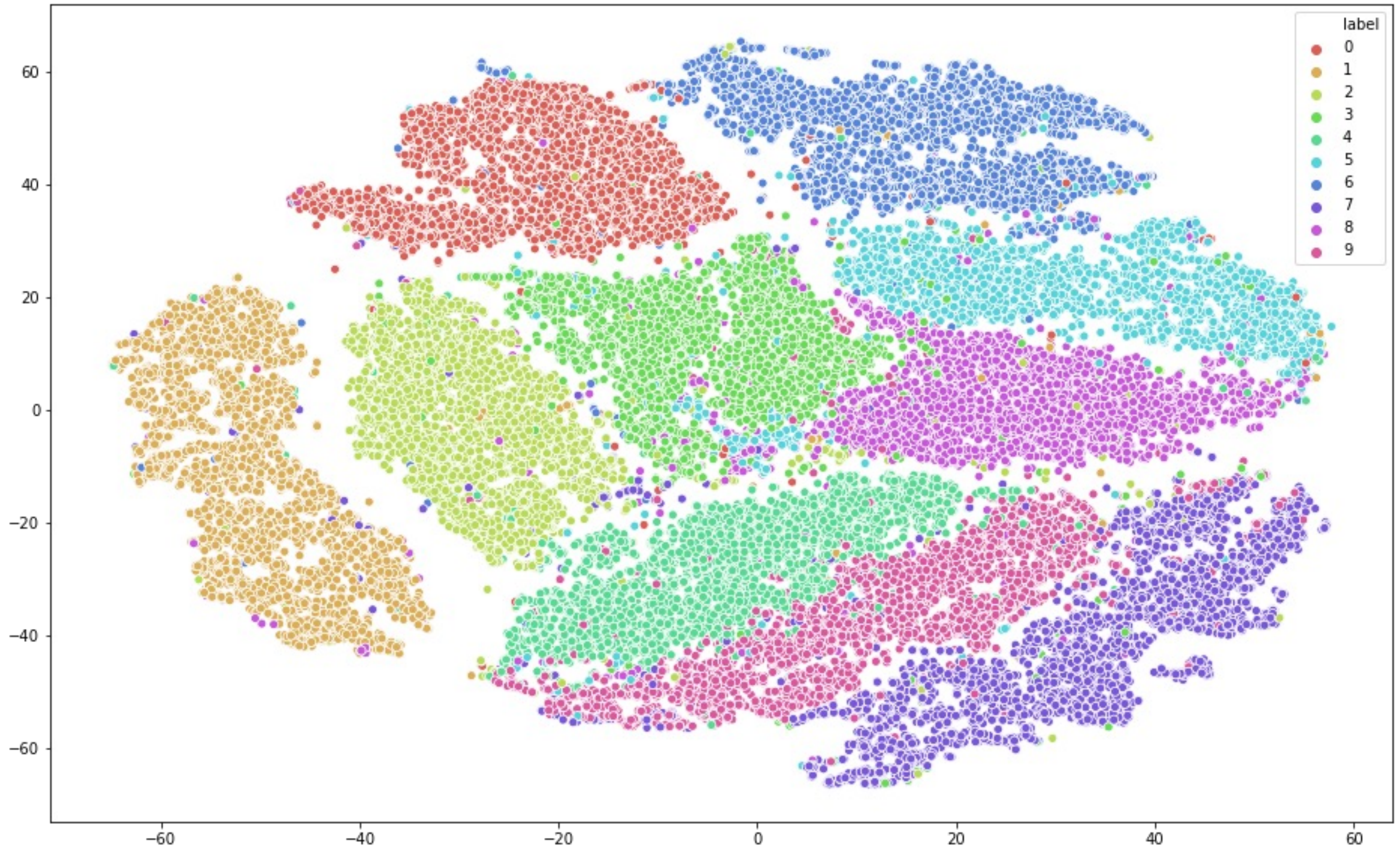
Under the hood: Generator



https://towardsdatascience.com/understand-transposed-convolutions-and-build-your-own-transposed-convolution-layer-from-scratch-4f5d97b2967

Problems in practice

- Mode collapse
  - Networks can miss entire “areas” (modes) of the generative space



From https://towardsdatascience.com/dimensionality-reduction-using-t-distributed-stochastic-neighbor-embedding-t-sne-on-the-mnist-9d36a3dd4521

Two Time-scale Update Rule (TTUR)

- Convergence can be better if the generator learns slower than the discriminator
  - Give the discriminator time to learn in each segment of input space

Discriminator too strong

- Conversely, if the discriminator is too strong, the generator can lack signal about which direction to move in parameter space (all directions look bad)

Statistical Tests

So, you’ve trained and tested a model… now what?

- After you’ve measured performance, it’s time to test for significance
    - Kinds of statistical tests
        - Testing if a model performs better than chance
        - Testing if a model performs better than another model
        - Regimes where training is very computationally expensive
    - Correcting statistical tests
        - Bonferroni
        - BHY corrections

Is my model better than chance?

Performance

- Comparing performance of classifiers/regressors
- How do you know if your accuracy number is “high” or error is “low”?

- Majority Class classifier is a good starting point for classification
  - Predicts the majority class (most common class in the training data) for every test sample
- E.g. if our “plays soccer” data has 10 days where they play and 5 days where they do not, the majority class would always predict play, and would be correct 10/15=66.7% of the time.

Performance

- E.g. if our “plays soccer” data has 10 days where they play and 5 days where they do not, the majority class would always predict play, and would be correct 10/15=66.7% of the time.

- Great! Let’s say your classifier gives 71.2% accuracy. Is that different enough from 66.7?

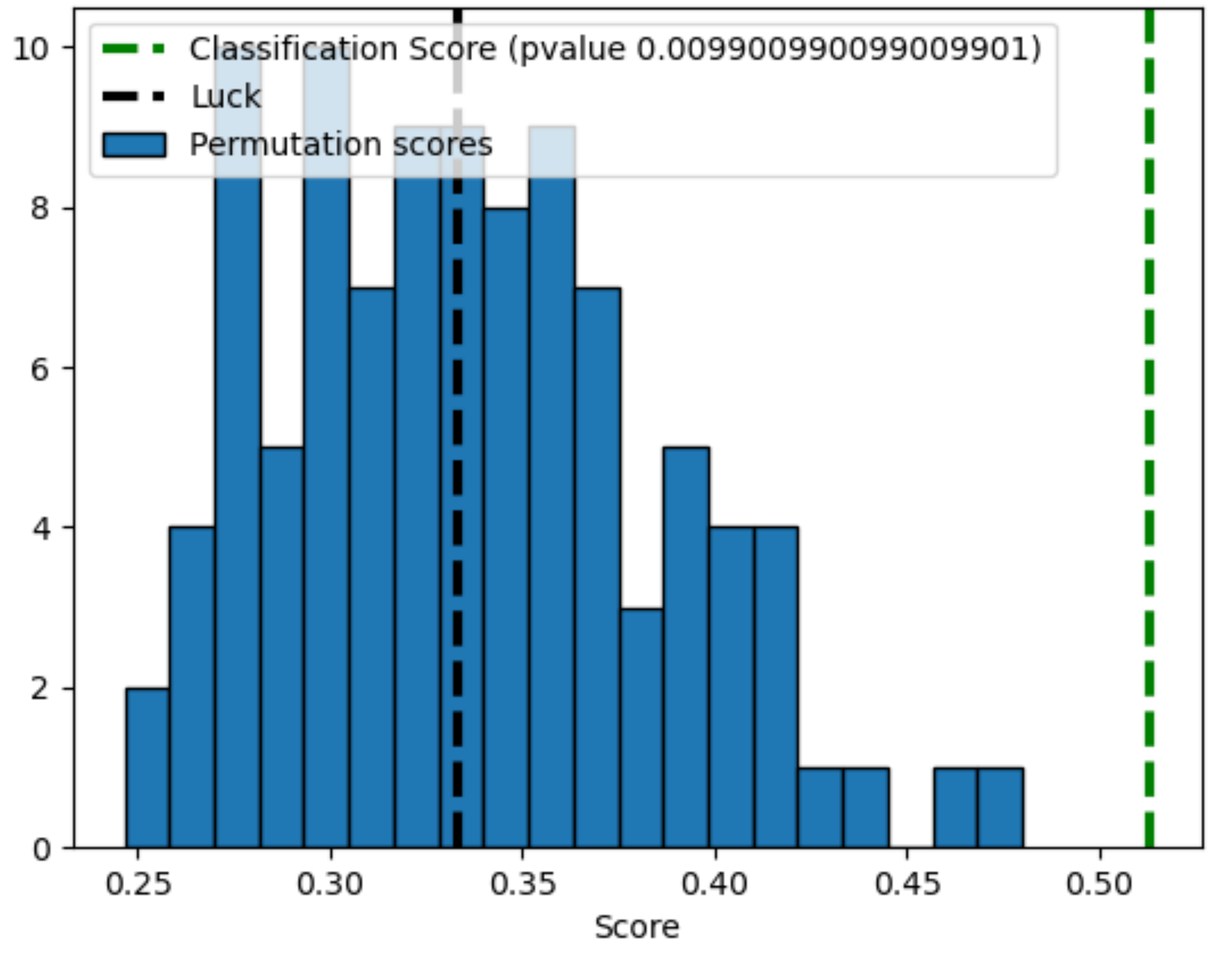
- We don’t have enough information to tell!

Performance

- The performance of the majority classifier does not give you a confidence interval (CI)

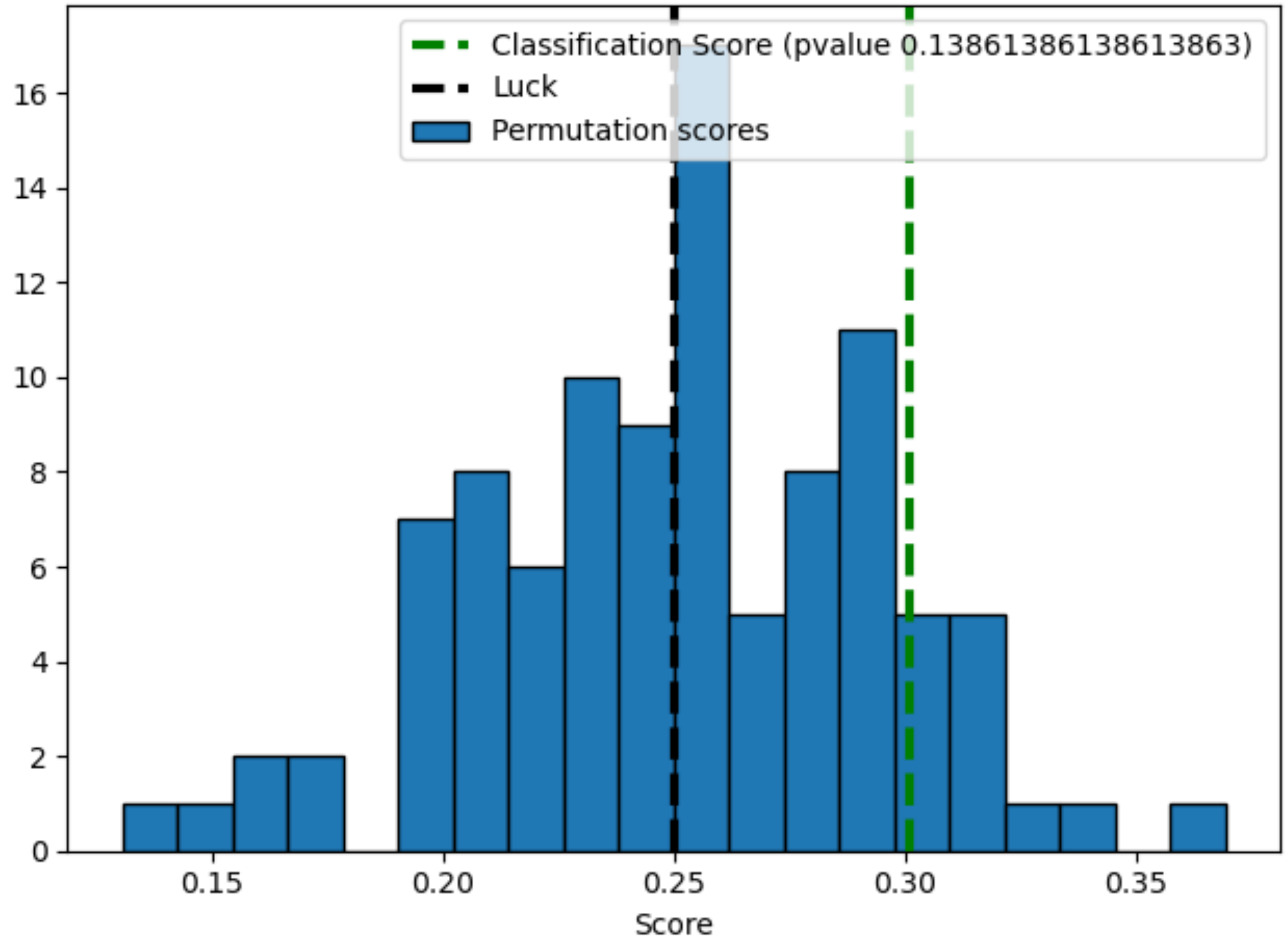
- To get a CI, you can use a permutation test
  - Randomly re-assign the labels to the rows of your data matrix
    - I.e. shuffle the elements (rows) in the label vector Y
  - Re-run your entire pipeline on the shuffled assignment, compute performance (accuracy/MSE)
  - Repeat many times (100-1000), recording performance each time
  - These repeated performance measurements on shuffled data create a null distribution
  - We can calculate a p-value from that distribution
    - % of shuffled-data-performance measures that outperform your true-data-performance is your p-value

Permutation test



https://scikit-learn.org/stable/auto_examples/feature_selection/plot_permutation_test_for_classification.html

Permutation test



Permutation test

- Fun animation https://www.jwilber.me/permutationtest/

Testing for a difference in performance between two models

Difference between performance

- Permutation test tells you if a model is doing better than chance
  - Does **not** tell you if there is a difference between models/algorithms/conditions
- E.g. if I use SVM I get 0.68 accuracy, with decision trees I get 0.71 accuracy. Are decision trees *really* better, or is it just due to random chance?

Difference between performance

- We need confidence intervals around our estimates of accuracy.
- Monte-Carlo or Repeated holdout validation
  - Like cross validation, but test set is randomly sampled
  - Can do a larger number of samples than with k-fold (which is limited to k samples)
- Then use a paired t-test
  - Note: some of the assumptions of the t-test are violated

Monte Carlo Cross Validation

Typical cross validation
Test data is always disjoint

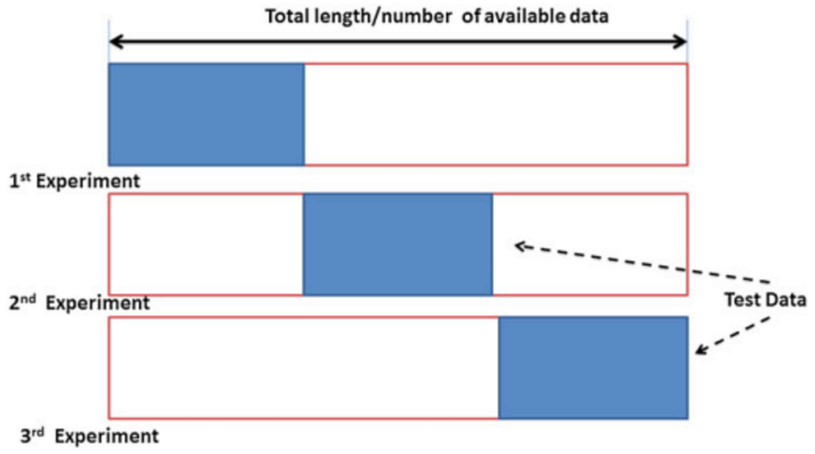


Fig. 3.8 Data splitting in *K*-fold cross-validation

MC cross validation
Test data is randomly sampled, can be re-used in different folds

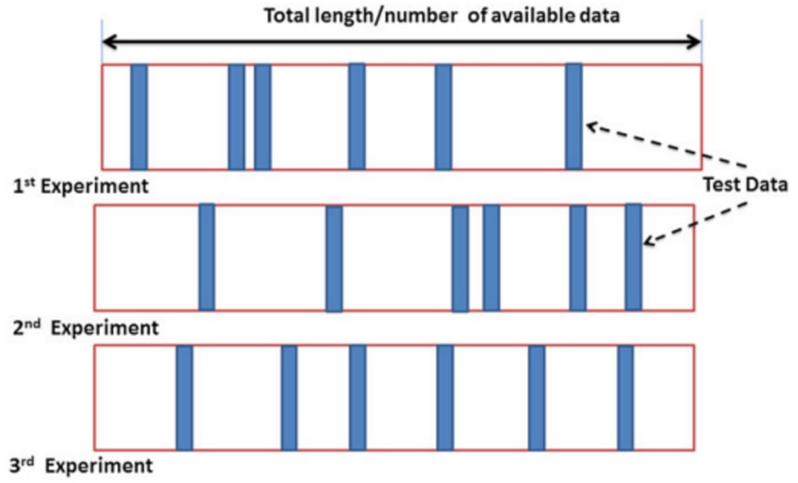


Fig. 3.7 Data splitting in the random sub-sampling approach

https://stats.stackexchange.com/questions/51416/k-fold-vs-monte-carlo-cross-validation

Difference between performance

- Better: McNemar’s test
  - See https://arxiv.org/pdf/1811.12808.pdf
  - See Sec 3.1 https://www.mitpressjournals.org/doi/pdf/10.1162/089976698300017197

What if it’s super expensive to run CV?

- Quite honestly, people do not always run statistical tests in DL
- You should at the very least run with several (e.g. 5) random initializations and report the standard deviation across those random initializations to be sure the effect you are seeing is real
- Could also use McNemar test here too

Correcting for multiple comparisons

- This you probably know something about because of your statistics classes
- Let’s say my algorithms are buggy or my data is bogus. My performance should not be above chance, no matter what I do.
- However, if I test for significance 20 times with $p=0.05$, what are the chances at least one of my tests will show significance ($p<0.05$) even though it should not?
    - P(at least one test passes) = 1-(P(no tests pass)) = $1-0.95^{20} = 0.64$

Correcting for multiple comparisons

- Thus, we need to adjust our p-value threshold to compensate
- Bonferoni - divide p-value threshold (alpha) by the total number of tests (e.g. 0.05/20)
    - P(at least one test passes) = 1-(P(no tests pass)) = $1-0.9975^{20} = 0.048$

- Benjamini-Hochberg or Benjamini-Yekutieli FDR correction
    - https://en.wikipedia.org/wiki/False_discovery_rate#Benjamini%E2%80%93Hochberg_procedure (independent or positively correlated dependence assumption)
    - https://en.wikipedia.org/wiki/False_discovery_rate#Benjamini%E2%80%93Yekutieli_procedure (arbitrary dependence)

Taking into account time/space

- If you calculate results over many samples in time or space, you can use the cluster permutation test (Maris & Oostenveld, 2007)
    - https://pubmed.ncbi.nlm.nih.gov/17517438/
- This corrects for multiple comparisons, while also taking into account space/time correlations

Themes

- Overfitting
  - What is it?
  - Why does it happen?
  - How can we detect it?
  - How can we avoid it?
- Measures of performance
- Cross validation
- Hypothesis testing

Other Resources

- Permutation Tests for Studying Classifier Performance
  - https://www.jmlr.org/papers/volume11/ojala10a/ojala10a.pdf
- Nested Cross Validation
  - https://www.youtube.com/watch?v=LpOsxBeggM0
- T-test video
  - https://www.youtube.com/watch?v=pTmLQvMM-1M
- Accuracy/Precision/F1
  - https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

Interpreting AI Models

Why interpretability?

- If you don’t know why your model predicts something, you’re flying blind
  - There could be errors in your data
    - E.g. duplicates of a house listed for a certain price in a housing price database causes skewed results and incorrect models.

These examples care of Rich Caruana, EMNLP 2020 keynote

Why interpretability?

- Your model could be leveraging spurious correlation
  - Pneumonia patients have ***better*** outcomes if they are asthmatic
    - Doctors will tell you this isn't true, but the effect appears because asthmatics are more conscious and tuned-in to difficulties in breathing
  - Patients over 100 years old have ***better*** outcomes than those 85-99 years old
    - Due to differences in care given to patients in different age groups
- Users may not know when to trust your predictions!!

These examples care of Rich Caruana, EMNLP 2020 keynote

What does interpretability enable?

- If we can interpret a model, we may be able to fix it
  - E.g. undo the strange shifts in pneumonia outcome in different age brackets
- Allows you to reason about the problem, better understand it
- Auditing of the decision process
  - Necessary in some fields (e.g. law., healthcare)

How to get Interpretability?

- Some models are easier to interpret than others
  - E.g. linear regression: many regression libraries will return p-values for the coefficients associated with each feature to tell you which ones are indicative
  - https://scikit-learn.org/stable/auto_examples/inspection/plot_linear_model_coefficient_interpretation.html
  - Neural networks: good luck!
- Can we interpret a model in a way that is not model-specific?
  - Model agnostic

LIME :
Local Interpretable Model-agnostic Explanations

- “The key intuition behind LIME is that it is much easier to approximate a black-box model by a simple model locally (in the neighborhood of the prediction we want to explain), as opposed to trying to approximate a model globally.”

https://www.oreilly.com/content/introduction-to-local-interpretable-model-agnostic-explanations-lime/

Intuition

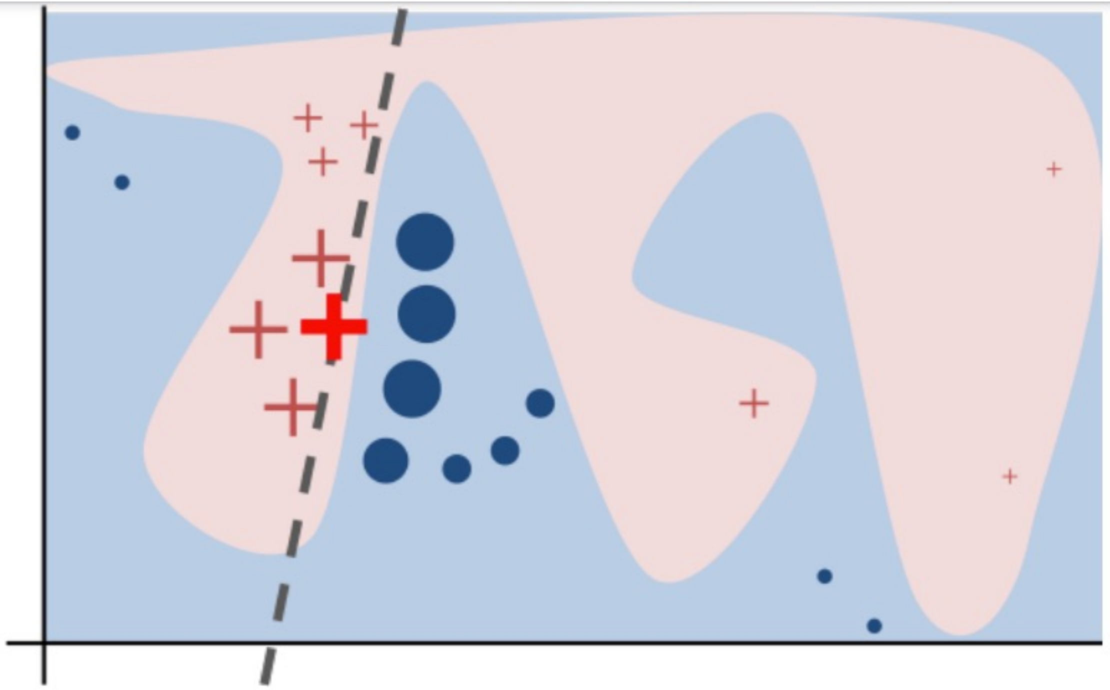


Figure 3: Toy example to present intuition for LIME. The black-box model's complex decision function f (unknown to LIME) is represented by the blue/pink background, which cannot be approximated well by a linear model. The bold red cross is the instance being explained. LIME samples instances, gets predictions using f, and weighs them by the proximity to the instance being explained (represented here by size). The dashed line is the learned explanation that is locally (but not globally) faithful.

https://arxiv.org/pdf/1602.04938.pdf

Lime Algorithm

- Preterb data
- Calculate similarity between perturbed and original observations
- Make predictions on new perturbed data using complex model
- Pick m features best describing the complex model’s prediction on the perturbed data
- Fit a simple model on the perturbed data with m features and similarity scores as weights
- Feature weights from the simple model make explanations for the complex model

https://www.youtube.com/watch?v=CY3t11vuuOM

Permute data

- Create a new dataset for the observation to be explained
- Do this by creating “nearby” data samples
    - E.g. turning on or off the features of the example

Intuition

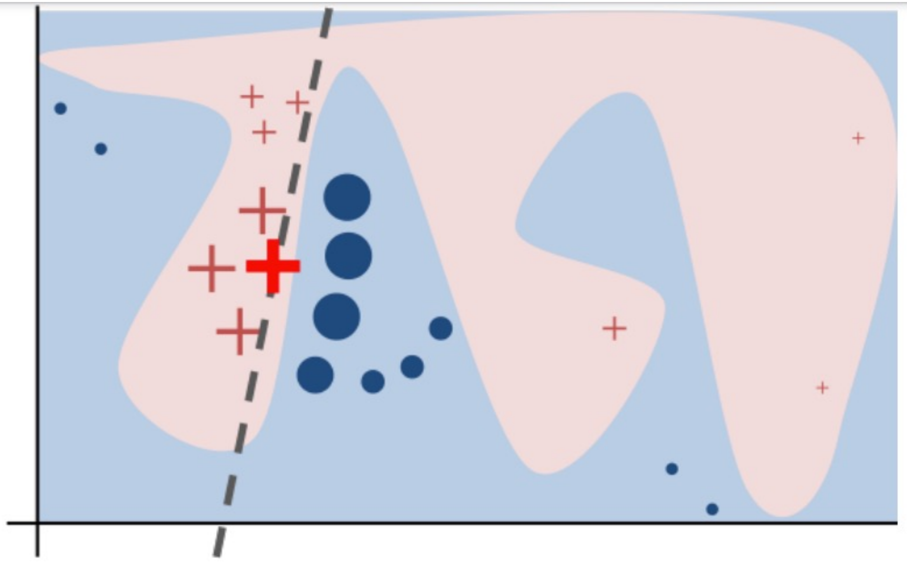


Figure 3: Toy example to present intuition for LIME. The black-box model's complex decision function f (unknown to LIME) is represented by the blue/pink background, which cannot be approximated well by a linear model. The bold red cross is the instance being explained. LIME samples instances, gets predictions using f, and weighs them by the proximity to the instance being explained (represented here by size). The dashed line is the learned explanation that is locally (but not globally) faithful.

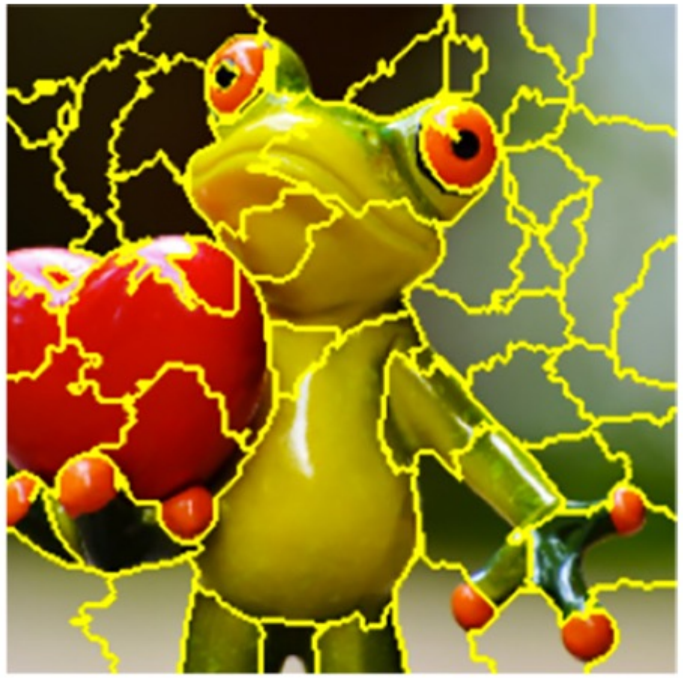
https://arxiv.org/pdf/1602.04938.pdf

3.5 Example 1: Text classification with SVMs

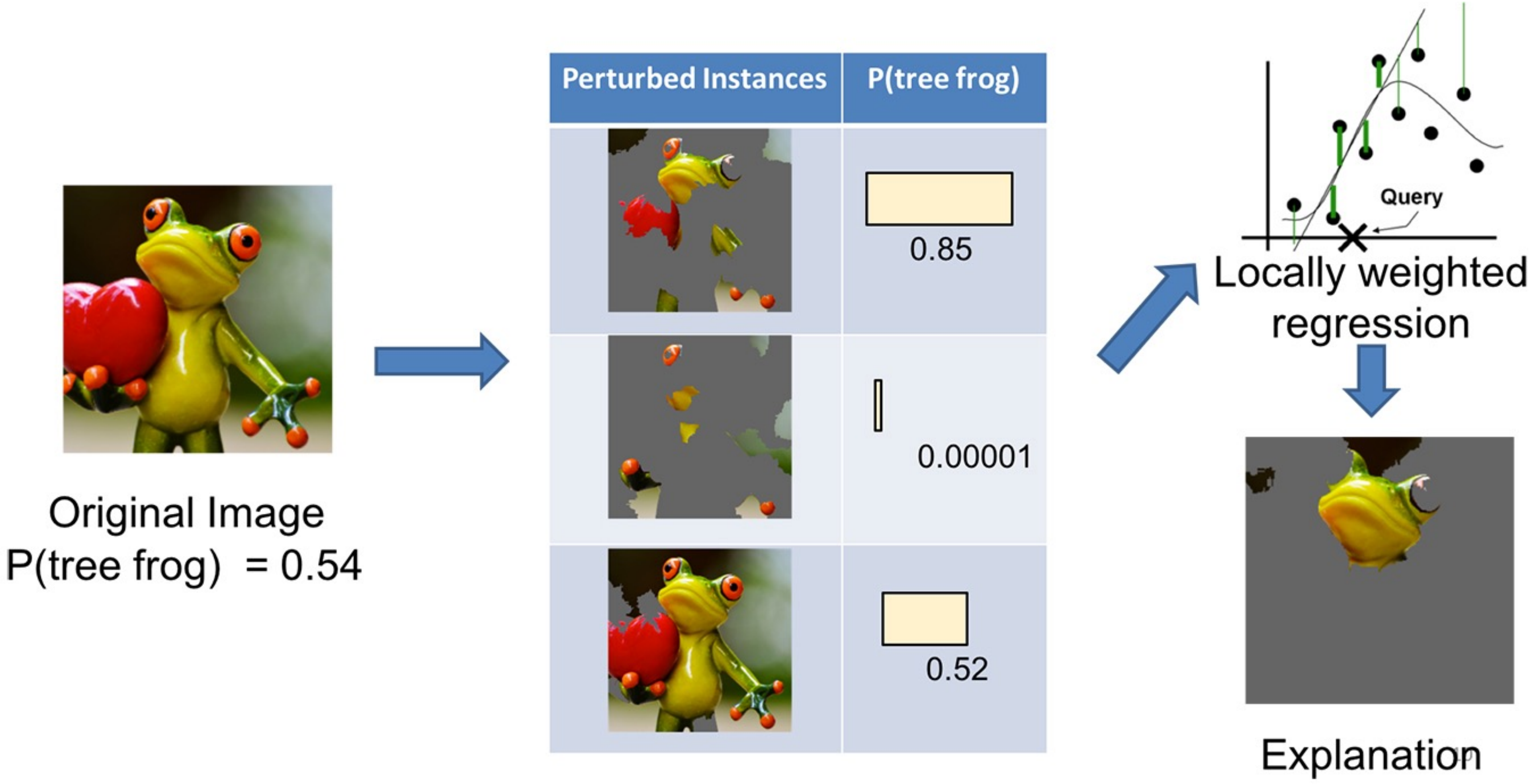
In Figure 2 (right side), we explain the predictions of a support vector machine with RBF kernel trained on unigrams to differentiate “Christianity” from “Atheism” (on a subset of the 20 newsgroup dataset). Although this classifier achieves 94% held-out accuracy, and one would be tempted to trust it based on this, the explanation for an instance shows that predictions are made for quite arbitrary reasons (words “Posting”, “Host”, and “Re” have no connection to either Christianity or Atheism). The word “Posting” appears in 22% of examples in the training set, 99% of them in the class “Atheism”. Even if headers are removed, proper names of prolific posters in the original newsgroups are selected by the classifier, which would also not generalize.

After getting such insights from explanations, it is clear that this dataset has serious issues (which are not evident just by studying the raw data or predictions), and that this classifier, or held-out evaluation, cannot be trusted. It is also clear what the problems are, and the steps that can be taken to fix these issues and train a more trustworthy classifier.

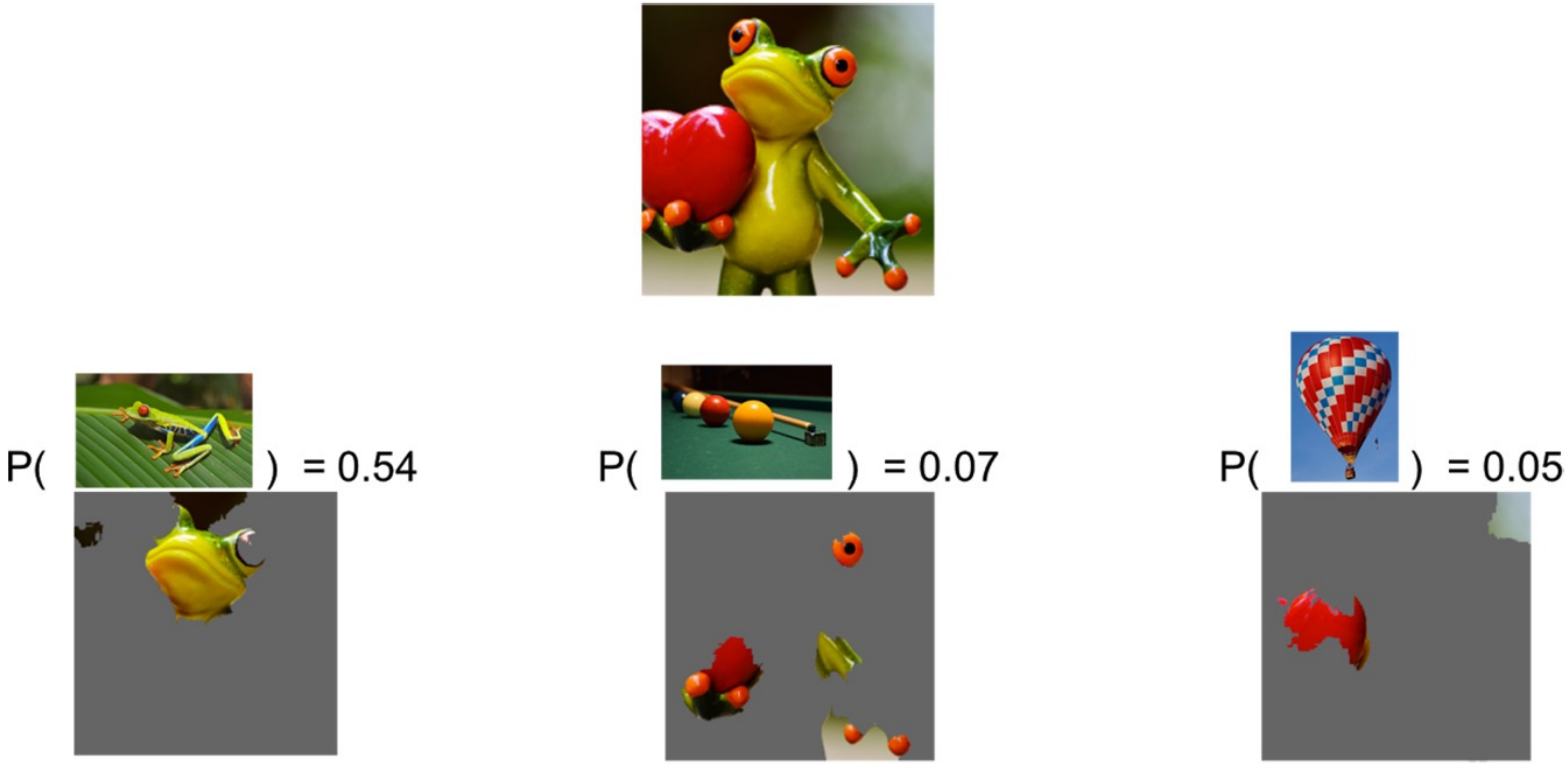
Original Image


Interpretable Components

https://www.oreilly.com/content/introduction-to-local-interpretable-model-agnostic-explanations-lime/



https://www.oreilly.com/content/introduction-to-local-interpretable-model-agnostic-explanations-lime/



https://www.oreilly.com/content/introduction-to-local-interpretable-model-agnostic-explanations-lime/

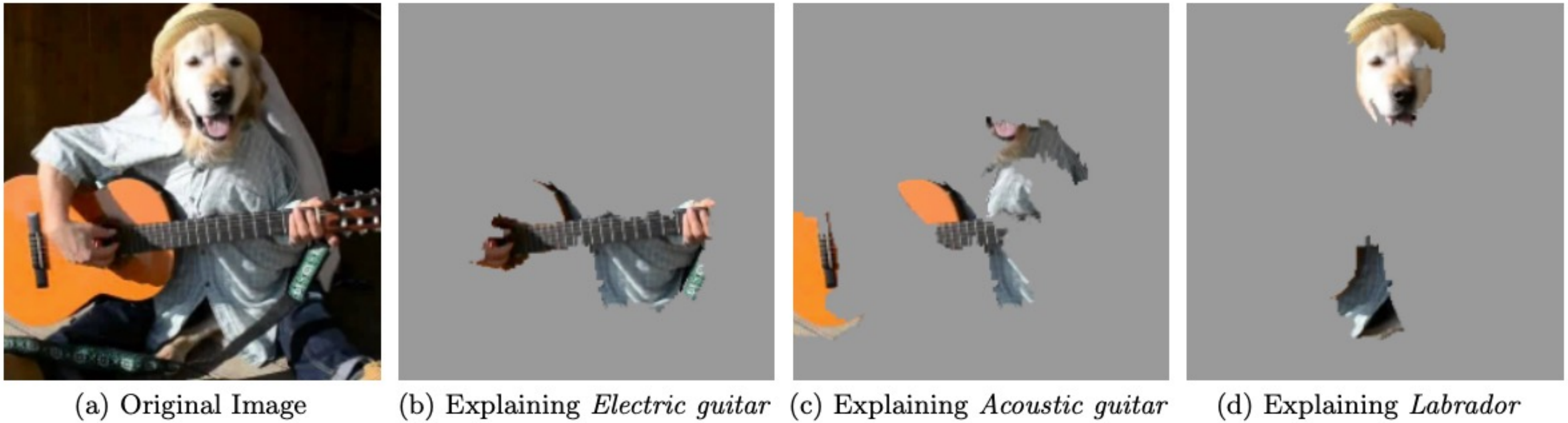
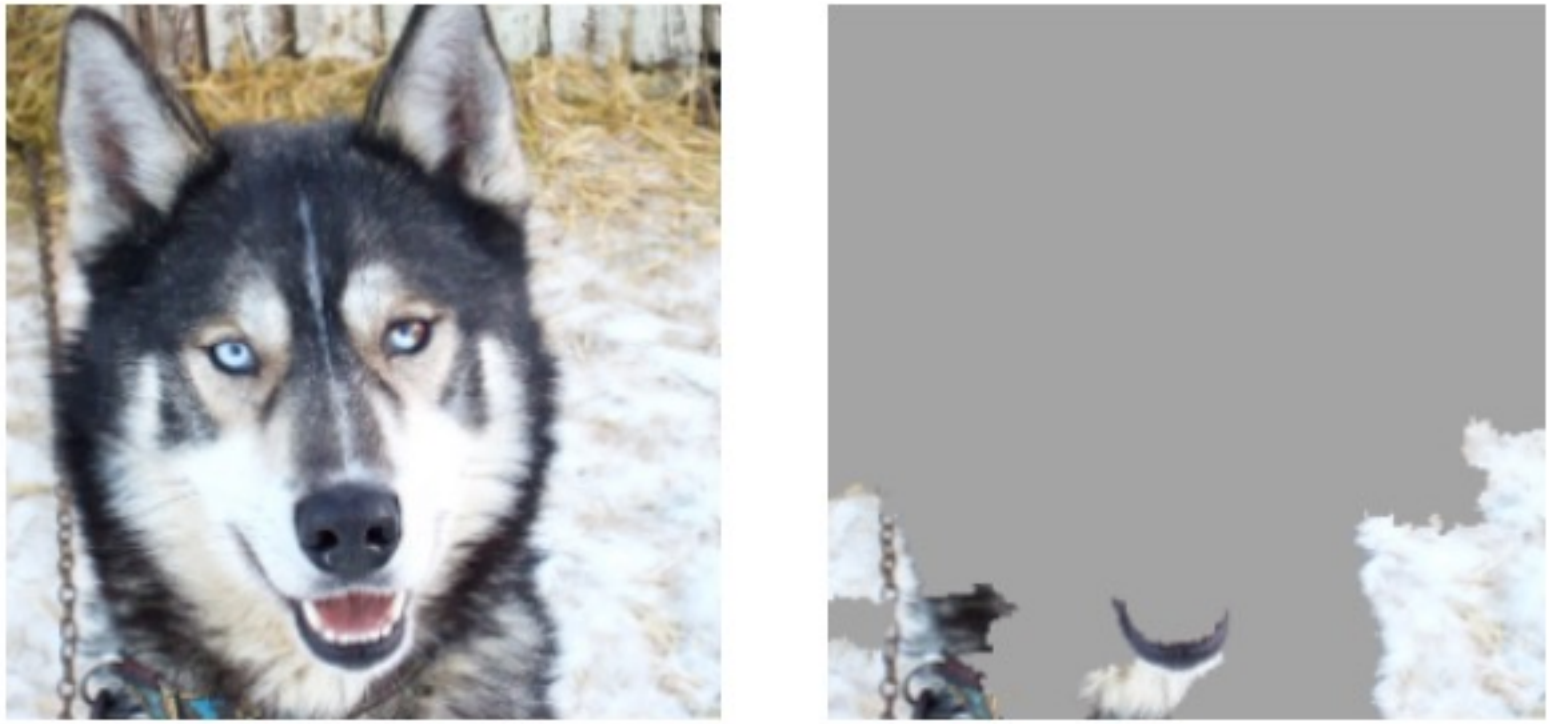

(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

Figure 4: Explaining an image classification prediction made by Google's Inception neural network. The top 3 classes predicted are "Electric Guitar" ($p = 0.32$), **"Acoustic guitar"** ($p = 0.24$) **and "Labrador"** ($p = 0.21$)

https://arxiv.org/pdf/1602.04938.pdf

Husky vs Wolf classification

- Example dataset to differentiate Huskys from Wolves
- Buuuut… in the training data, all of the hukies were inside and all of the wolves were outside!
  - Test data had huskies outside.
- Performance was very good



(a) Husky classified as wolf (b) Explanation

Figure 11: Raw data and explanation of a bad model's prediction in the "Husky vs Wolf" task.

Resources

- Lime tutorials
  - https://github.com/marcotcr/lime
- I loved this set of 3 talks on interpretable ML – some technical parts, but also a lot of intuition.
  - https://www.youtube.com/watch?v=vO7h2-_VwWE