# Interpretability Tools as Feedback Loops

## Toronto Machine Learning Summit 2022

J. Setpal

November 30, 2022

# Outline

1. Setting the Stage
2. Baselining Interpretability
3. Leveraging Interpretability

DagsHub

# Outline

1. Setting the Stage
2. Baselining Interpretability
3. Leveraging Interpretability

DagsHub

# Here's a Scenario

Consider the following:

a. We want to build a classifier (classifiers are cool).

# Here's a Scenario

Consider the following:

a. We want to build a classifier (classifiers are cool).
b. This classifier differentiates between an Orca and a Leopard.

# Here's a Scenario

Consider the following:

a. We want to build a classifier (classifiers are cool).
b. This classifier differentiates between an Orca and a Leopard.
c. We use the Caltech-256 dataset to obtain images of both:


DagsHub

# Here's a Scenario

Consider the following:

a. We want to build a classifier (classifiers are cool).

b. This classifier differentiates between an Orca and a Leopard.

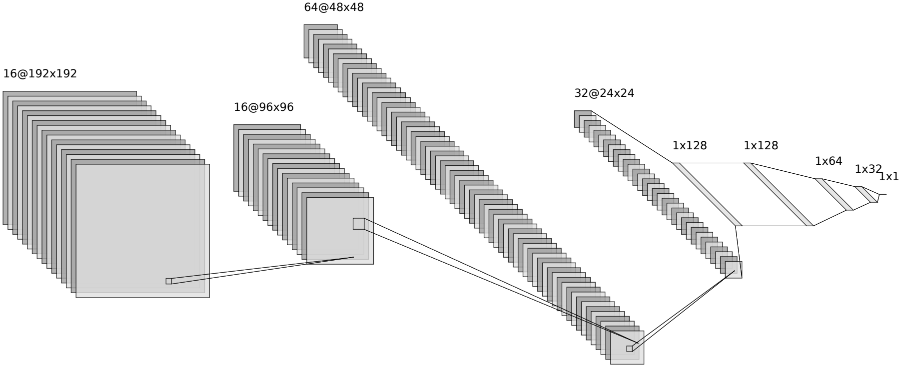
c. We use the Caltech-256 dataset to obtain images of both:


d. There are 188 leopard images and 89 orca images.

DagsHub

# More Scenario Stuff

Here's our model architecture:



# Last Bit of Scenario, I Promise

We use:

a. Optimizer: `Adam`
   - Learning Rate: $10^{-2}$
   - Epsilon: $10^{-8}$

b. Loss: `BinaryCrossEntropy`

# Last Bit of Scenario, I Promise

We use:

a. Optimizer: `Adam`
   - Learning Rate: $10^{-2}$
   - Epsilon: $10^{-8}$

b. Loss: `BinaryCrossEntropy`

After training, we achieve a test accuracy of `0.5000`. This *sucks*.

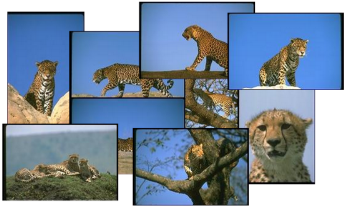
We use:

a. Optimizer: `Adam`
   - Learning Rate: $10^{-2}$
   - Epsilon: $10^{-8}$

b. Loss: `BinaryCrossEntropy`

After training, we achieve a test accuracy of `0.5000`. This *sucks*.

Here are some misclassified samples:

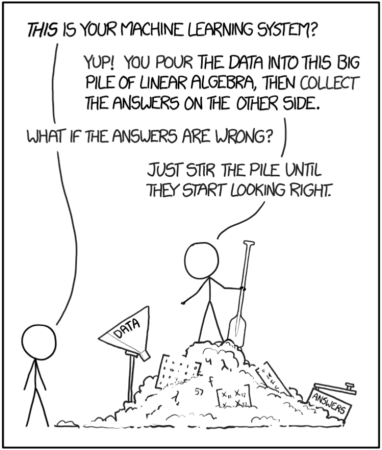


How can we diagnose the cause of this?

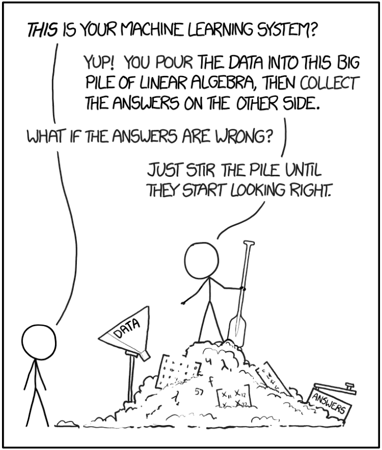
# Outline

1. Setting the Stage
2. Baselining Interpretability
3. Leveraging Interpretability

# What even *is* Interpretability?

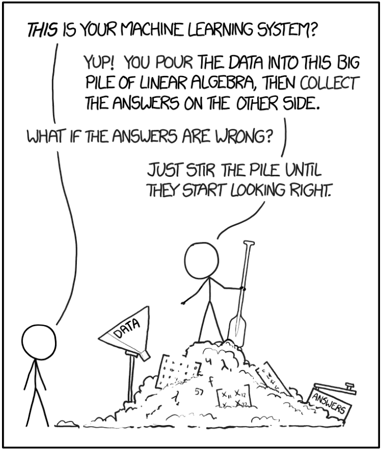


# What even *is* Interpretability?




Interpretability within Machine Learning is the **degree** to which we can understand the **cause** of a decision, and use it to consistently <u>predict the model's prediction</u>.

# What even *is* Interpretability?

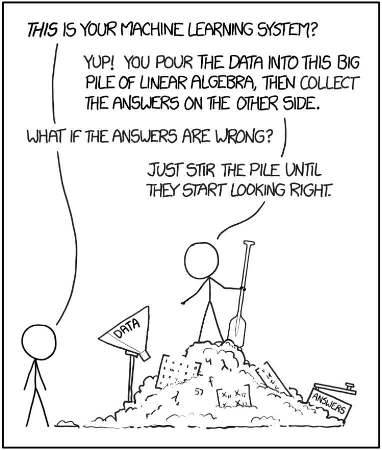



Interpretability within Machine Learning is the **degree** to which we can understand the **cause** of a decision, and use it to consistently <u>predict the model's prediction</u>.

This is easy for shallow learning.

DagsHub

# What even *is* Interpretability?

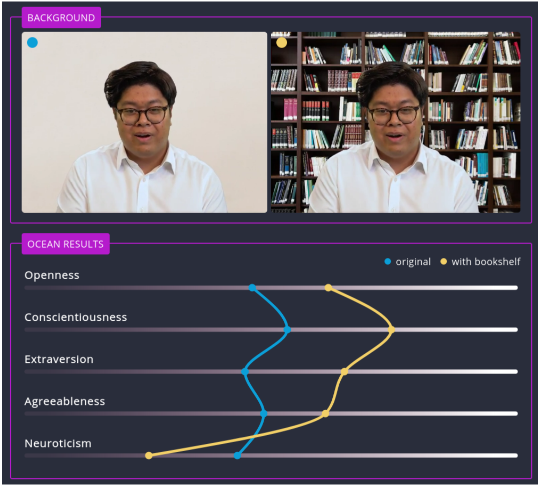



Interpretability within Machine Learning is the **degree** to which we can understand the **cause** of a decision, and use it to consistently predict the model's prediction.

This is easy for shallow learning. For deep learning however, it is a **lot harder**.

DagsHub

# A Cautionary Tale

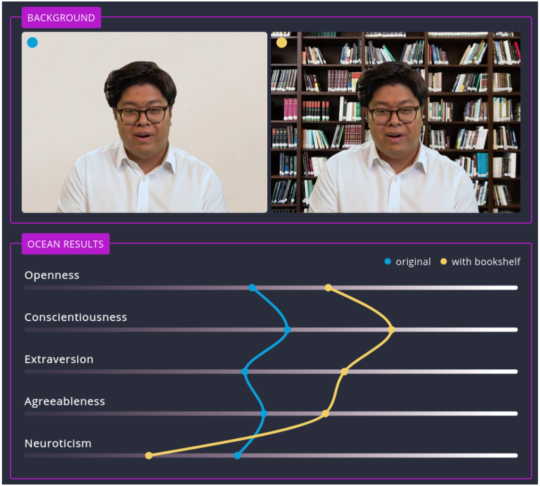



https://interaktiv.br.de/ki-bewerbung/en/

Start-up attempting to make the application process 'faster, but also more objective and fair'.

DagsHub

# A Cautionary Tale




https://interaktiv.br.de/ki-bewerbung/en/

Start-up attempting to make the application process 'faster, but also more objective and fair'.

They were not successful.

DagsHub

# Class Activation Mappings

For deep learning, interpretability techniques today involve a fairly straightforward formula:

# Class Activation Mappings

For deep learning, interpretability techniques today involve a fairly straightforward formula:

- Split hidden layers.
- Expose weights.
- *Observe!*

# Class Activation Mappings

For deep learning, interpretability techniques today involve a fairly straightforward formula:

- Split hidden layers.
- Expose weights.
- *Observe!*

We'll focus today's discussion on **Class Activation Mappings (CAMs)**:

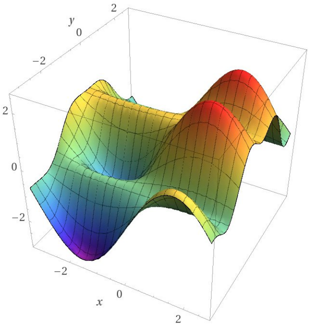

DagsHub

# Building Feedback Loops

Finding optimal model weights is an **NP-hard** problem.

# Building Feedback Loops

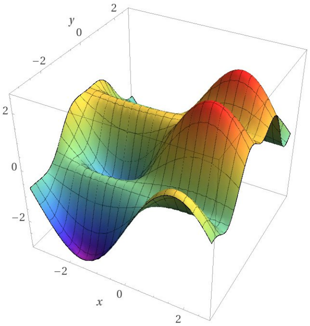
Finding optimal model weights is an **NP-hard** problem.




Model Search Space

# Building Feedback Loops

Finding optimal model weights is an **NP-hard** problem.



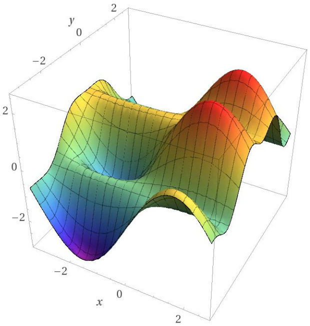

Model Search Space

We can't speed this up. However, we do have information about our training set that we can use to **motivate training behaviour**.

DagsHub

# Building Feedback Loops

Finding optimal model weights is an **NP-hard** problem.




Model Search Space

We can't speed this up. However, we do have information about our training set that we can use to **motivate training behaviour**.

So, the idea here is simple: use shared knowledge (+ common sense) to modify how we train our models.

DagsHub

# Outline

# Getting Back to the Challenge

There are some obvious causes for why it performs poorly:

a. There are too few, unbalanced training samples.

# Getting Back to the Challenge

There are some obvious causes for why it performs poorly:

a. There are too few, unbalanced training samples.
   Solution: Data Augmentation

# Getting Back to the Challenge

There are some obvious causes for why it performs poorly:

a. There are too few, unbalanced training samples.
   Solution: Data Augmentation
b. **The images have a sharp color dominance.**

# Getting Back to the Challenge

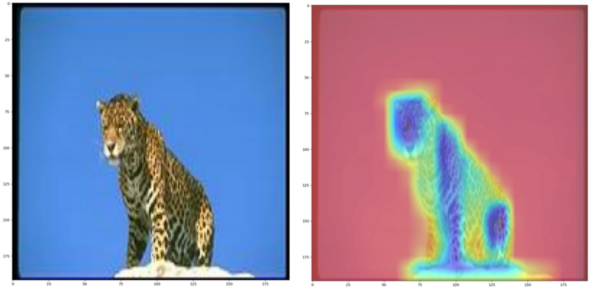
There are some obvious causes for why it performs poorly:

a. There are too few, unbalanced training samples.
   <u>Solution:</u> Data Augmentation

b. **The images have a sharp color dominance.**

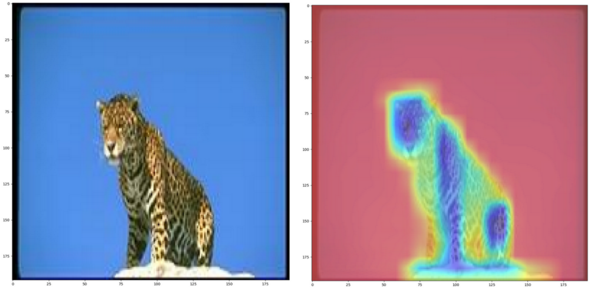
# Diagnosing the Model

When we obtain a Class Activation Map of a sample image, we observe:



# Diagnosing the Model

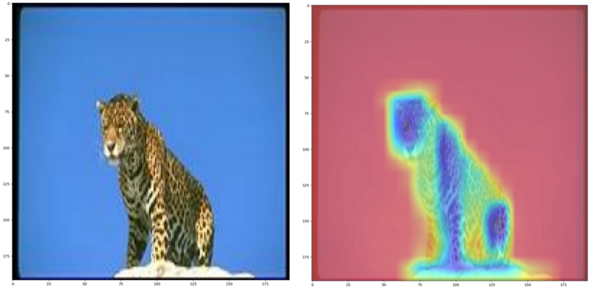
When we obtain a Class Activation Map of a sample image, we observe:



It **does not** use the leopard to base it's prediction! This is prevalent across the dataset.

# Diagnosing the Model

When we obtain a Class Activation Map of a sample image, we observe:

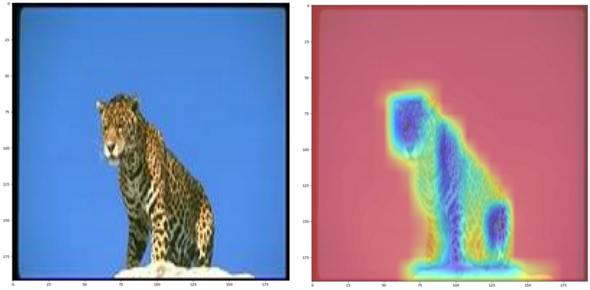


It **does not** use the leopard to base it's prediction! This is prevalent across the dataset.

**Observation:** The targets in our entire training dataset are centered.

# Diagnosing the Model

When we obtain a Class Activation Map of a sample image, we observe:



It **does not** use the leopard to base it's prediction! This is prevalent across the dataset.

**Observation:** The targets in our entire training dataset are centered.

Q: Can we exploit this?

# Introducing CAMLoss!

Here's the idea:

a. In addition to the prediction, we output the class activation map.

# Introducing `CAMLoss`!

Here's the idea:

a. In addition to the prediction, we output the class activation map.
b. We extract a random subset of the **top portion** of the map.

# Introducing CAMLoss!

Here's the idea:

a. In addition to the prediction, we output the class activation map.
b. We extract a random subset of the **top portion** of the map.
c. We return the mean of the weights. Weights $\propto \frac{1}{\text{Fit Quality}}$

# Introducing CAMLoss!

Here's the idea:

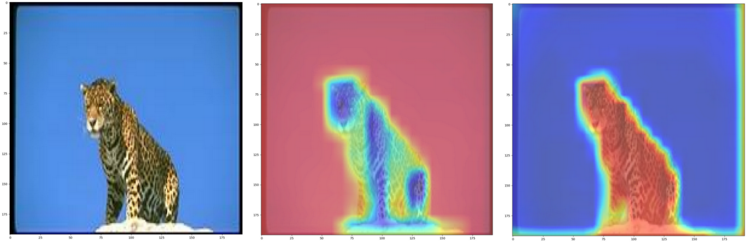
a. In addition to the prediction, we output the class activation map.
b. We extract a random subset of the **top portion** of the map.
c. We return the mean of the weights. Weights $\propto \frac{1}{\text{Fit Quality}}$
d. This is our additional self-supervised loss function!

# Introducing `CAMLoss`!

Here's the idea:

a. In addition to the prediction, we output the class activation map.
b. We extract a random subset of the **top portion** of the map.
c. We return the mean of the weights. Weights $\propto \frac{1}{\text{Fit Quality}}$
d. This is our additional self-supervised loss function!

Obtaining the Class Activation Map of the updated model, we observe:



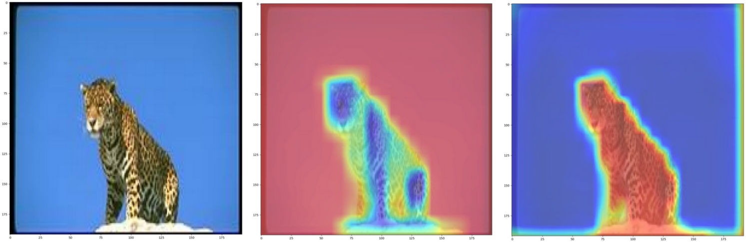
DagsHub

# Introducing `CAMLoss`!

Here's the idea:

a. In addition to the prediction, we output the class activation map.
b. We extract a random subset of the **top portion** of the map.
c. We return the mean of the weights. Weights $\propto \frac{1}{\text{Fit Quality}}$
d. This is our additional self-supervised loss function!

Obtaining the Class Activation Map of the updated model, we observe:



**Great Success!**

# Thank you!

Have an awesome rest of your day! Any questions for me?

**Code, Experiments, Data, Slides:**

`https://dagshub.com/jinensetpal/tmls22.git`