Evaluating, Interpreting, and Monitoring Machine Learning Models

Ankur Taly

Research Scientist, Google Cloud

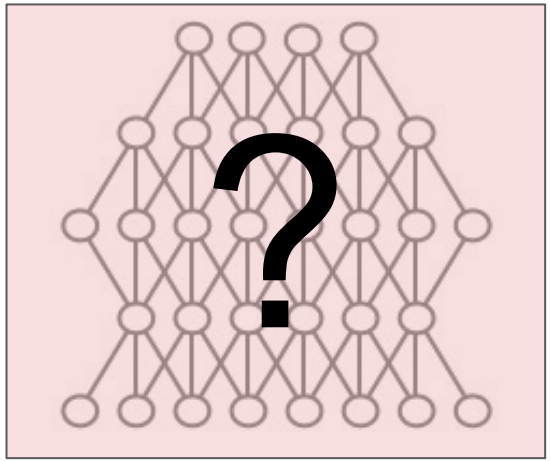
ataly@google.com

ODSC East Conference

April 2022

Problem: Understanding Black Box Machine Learning Models

Output
(Label, sentence, next word, next move, etc.)



Input
(Image, sentence, game position, etc.)

How do we:
- Evaluate
- Debug
- Explain
- Monitor

large, complex models?

Evaluating ML Models

- Practically: Test/Train Split
  - Some data is randomly kept aside (test data)
  - Model is trained on rest (training data)
  - Evaluation: **Test accuracy**
- Theoretically: PAC learning
  - Learner gets sample from underlying data distribution
  - Evaluation: Model is **P**robably **A**pproximately **C**orrect over distribution

Issues with Test Accuracy

- Test accuracy may vary across slices
- Test set may not be representative of deployment

Test Accuracy may vary across slices

Slice	Log Loss	Size	Effect Size
All	0.35	30k	n/a
Sex = Male	0.41	20k	0.28
Sex = Female	0.22	10k	-0.29
Occupation = Prof-specialty	0.45	4k	0.18
Education = HS-grad	0.33	9.8k	-0.05
Education = Bachelors	0.44	5k	0.17
Education = Masters	0.49	1.6k	0.23
Education = Doctorate	0.56	0.4k	0.33

Consequence: Disparate Impact

Issues with Test Accuracy

- Test accuracy may vary across slices
- Test set may not be representative of deployment

Visual Question Answering (VQA 1.0)

Q. How symmetric are the white bricks on either side of the building?

Model answers: very
Ground truth: very

Thoughtfully constructed training data

200K images, 600K questions

Test accuracy of Kazemi and Elqursh (2017) model: **61%**

Right for the wrong reason!

Q: “how asymmetric are the white bricks on either side of the building”
A: *very*

Q: “how soon are the bricks fading on either side of the building”
A: *very*

Q: “how fast are the bricks speaking on either side of the building”
A: *very*

Paper: Did the model understand the question? ACL 2018

Issue

- Test data is not representative of deployment
- Model relies on spurious correlations to show good test data performance
    - It relies on the type of question (“how many”, “what color”) to pick the answer

Fix: Interpret model predictions

Interpreting Model Predictions

- ***Why did the model make this prediction?***

Interpreting Model Predictions

Hot topic with several known approaches (e.g., LIME, SHAP, Integrated Gradients, TCAV, …)

I will cover two in this talk:

- Integrated Gradients [ICML 2017]
- Targeted What-If Exploration [UAI 2021]

The Attribution Problem

Attribute a model's prediction on an input to features of the input

Examples:

- Attribute an object recognition network's prediction to its pixels
- Attribute a text sentiment network's prediction to individual words
- Attribute a lending model's prediction to features of the loan application

Feature Attributions

Attribution to pixels

Question: how symmetrical are the white bricks on either side of the building

Attribution to words

Feature Attributions

Attribution to pixels

Question: how symmetrical are the white bricks on either side of the building

Notice that the word “symmetrical” gets tiny attribution. This explains the model’s insensitivity to perturbations to this word.

Attribution to words

Applications of Attributions

While attributions are very simplified response to “why this prediction”, they are surprisingly useful!

- Debugging model predictions
- Generating an explanation for the end-user
- Analyzing model robustness
- Monitoring models in production

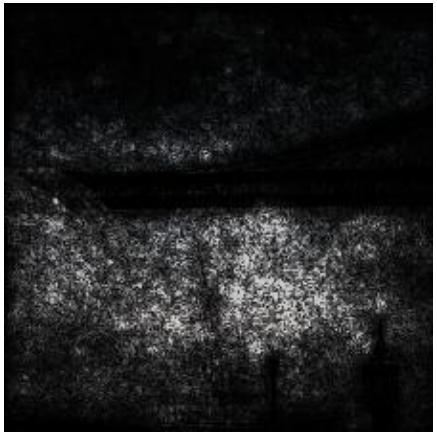
Naive Approaches

- **Ablations**: Drop each feature and note the change in prediction
  - Computationally expensive, Unrealistic inputs, Misleading when features interact

Naive Approaches

- **Ablations**: Drop each feature and note the change in prediction
  - Computationally expensive, Unrealistic inputs, Misleading when features interact
- **Feature*Gradient**: Attribution for feature x_i is $x_i * \partial y/\partial x_i$

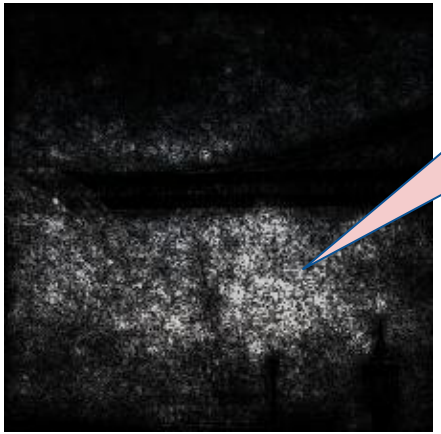
Prediction: **“fireboat”**



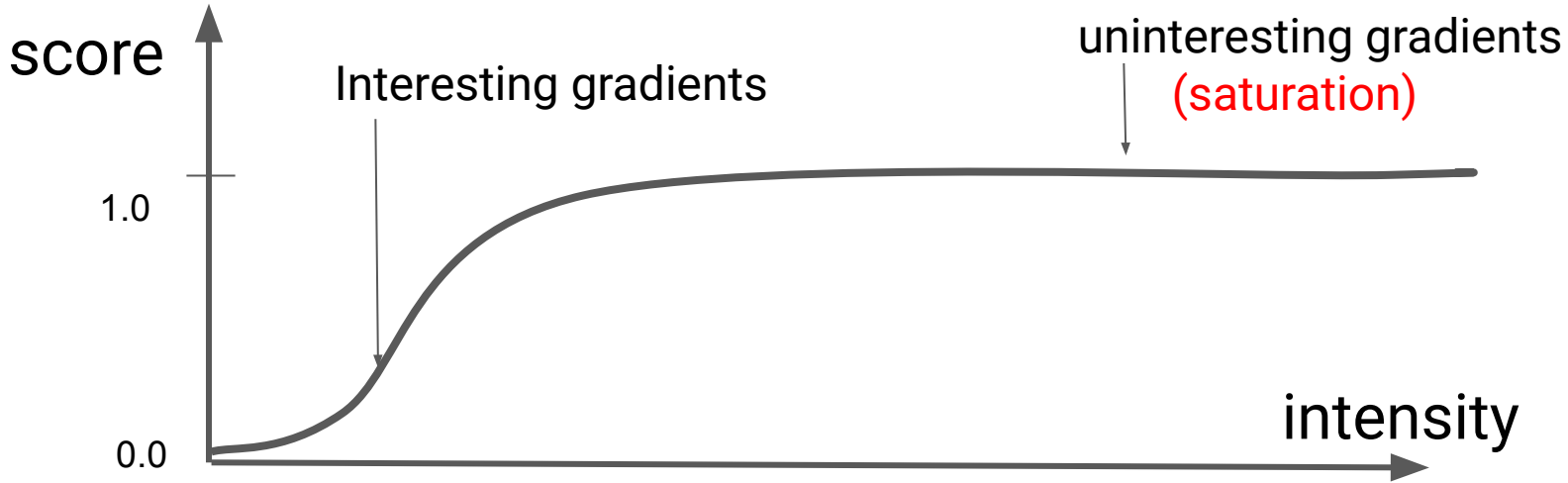
Naive Approaches

- **Ablations**: Drop each feature and note the change in prediction
  - Computationally expensive, Unrealistic inputs, Misleading when features interact
- **Feature*Gradient**: Attribution for feature x_i is $x_i * \partial y/\partial x_i$

Prediction: **“fireboat”**



Gradients in the vicinity of the input seem like noise



Baseline

… scaled inputs ...

Input

… gradients of scaled inputs ….

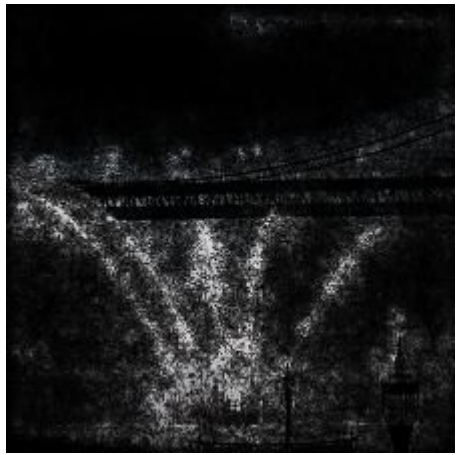
Integrated Gradients [ICML, 2017]

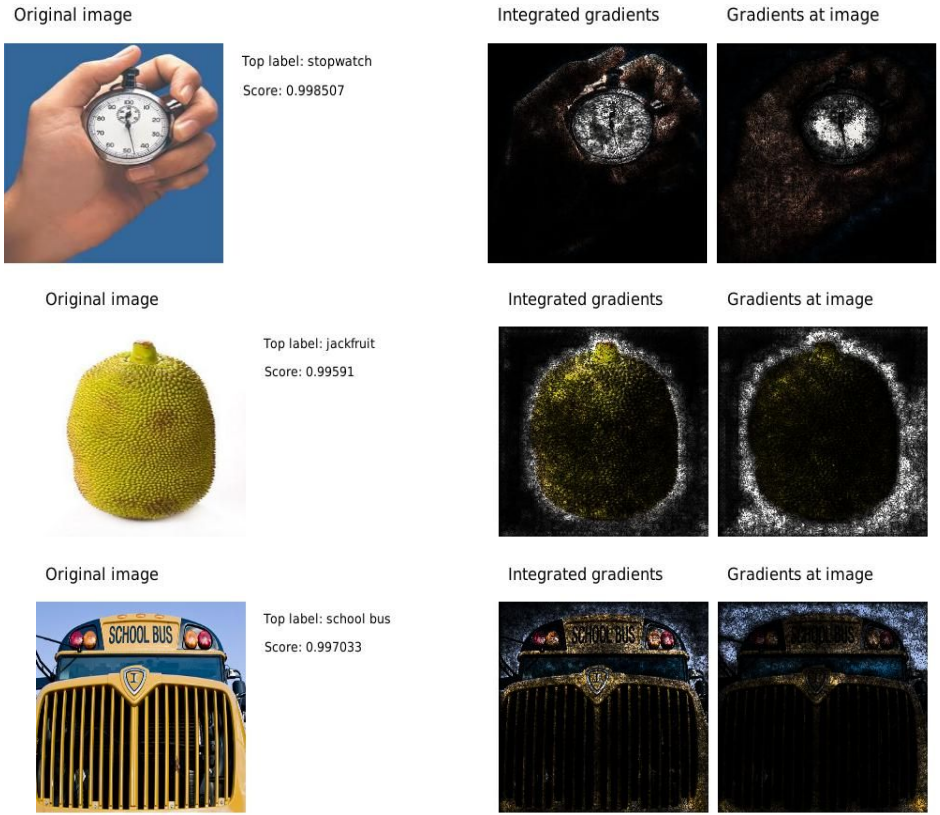
Integrate the gradients along a **straight-line path from baseline to input**

$$IG(input, base) ::= (input - base) * \int_{0}^{1} \nabla F(\alpha * input + (1-\alpha) * base)\, d\alpha$$

Original image

Integrated Gradients





Many more Inception+ImageNet examples here

What is a baseline?

- Ideally, the baseline is an **informationless input for the model**
  - E.g., Black image for image models
  - E.g., Empty text or zero embedding vector for text models
- **Integrated Gradients explains F(input) - F(baseline) in terms of input features**

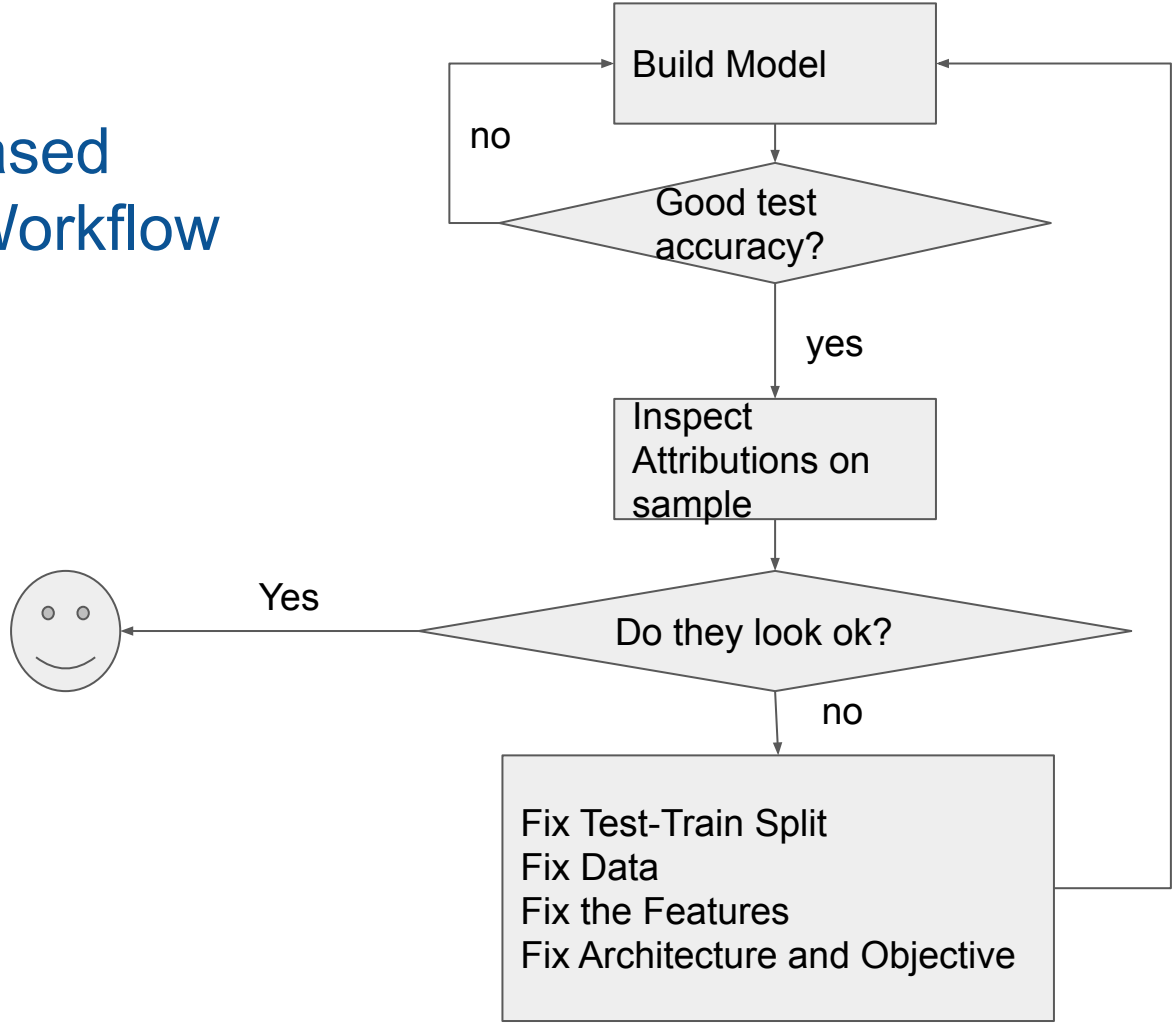
Axiomatic Guarantee

Theorem [ICML 2017]: Integrated Gradients is the **unique** path-integral method satisfying certain desirable properties: Sensitivity, Insensitivity, Linearity preservation, Implementation invariance, Completeness, and Symmetry

Historical note:

- Integrated Gradients is the **Aumann-Shapley method** from cooperative game theory, which has a similar characterization; see [Friedman 2004]

Attribution based Debugging Workflow



Why is this image labeled as a “clog”?

Original image

“Clog”

Why is this image labeled as a “clog”?

Original image

Integrated Gradients
(for label “clog”)

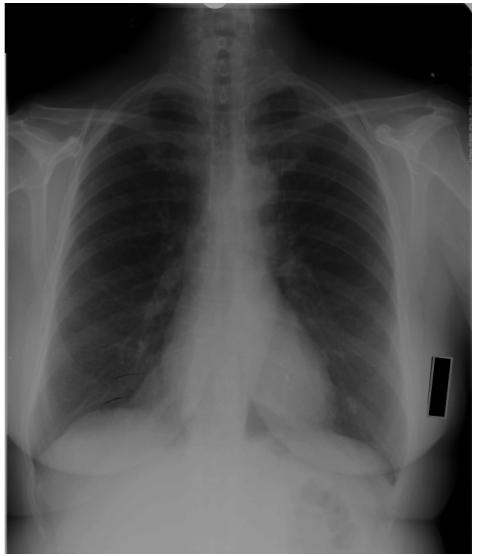
“Clog”

Next step: Gather more images of Clogs of different colors?

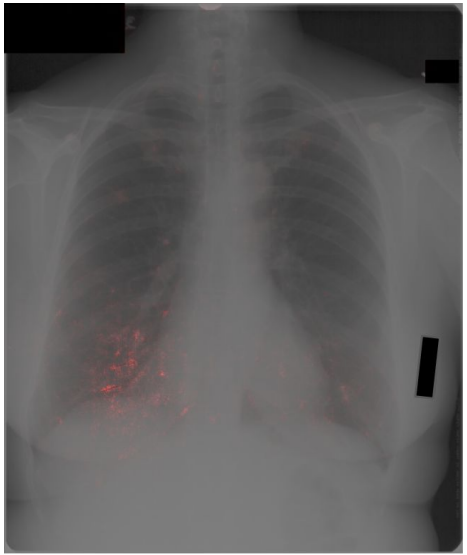
Detecting a data issue

- Deep network predicts various diseases from chest x-rays

Original image



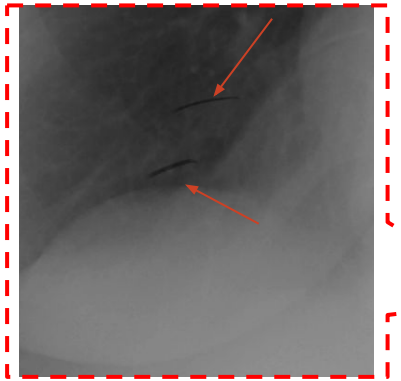
Integrated gradients
(for top label)



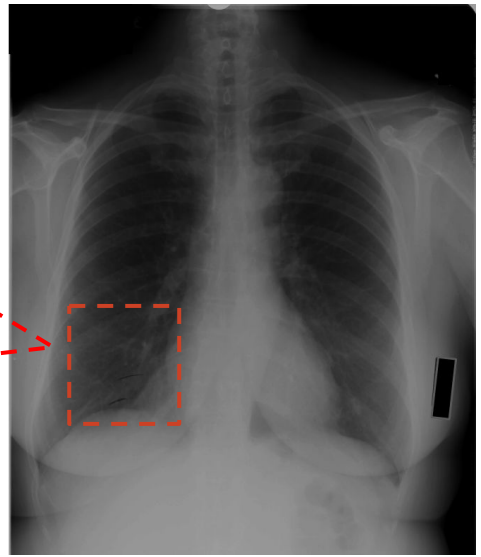
Detecting a data issue

- Deep network predicts various diseases from chest x-rays
- **Finding**: Attributions fell on radiologist's markings (rather than the pathology)

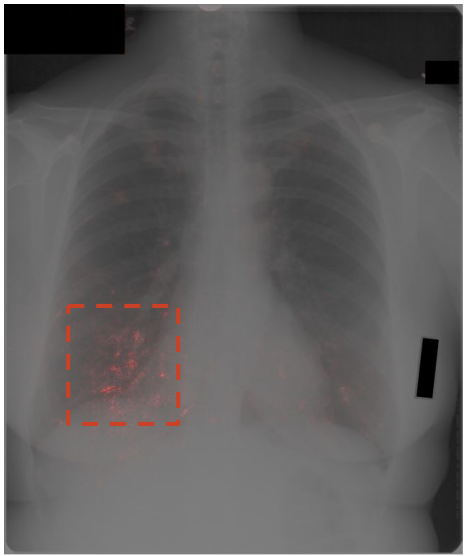
Pen marks!!



Original image



Integrated gradients
(for top label)



Summary

Integrated Gradients is a technique for attributing a deep network's prediction to its input features. It is **very easy to apply**, **widely applicable** and backed by an **axiomatic theory.**

Code: https://github.com/ankurtaly/Integrated-Gradients

References:

- Axiomatic Attribution for Deep Networks [ICML 2017]
- Did the model understand the question? [ACL 2018]
- Using a deep learning algorithm and integrated gradients explanation to assist grading for diabetic retinopathy [Journal of Ophthalmology, 2018]
- Exploring Principled Visualizations for Deep Network Attributions [EXSS Workshop, 2019]
- Using Attribution to Decode Dataset Bias in Neural Network Models for Chemistry [PNAS, 2019]

What-If Exploration

Feature Attributions: Pros and Cons

Pros:

- Axiomatic foundation
- Identifies the salient factors
- Completeness: Attributions apportion the prediction

Cons:

- Often deemed unintuitive by users
- Cannot directly map attributions to model semantics [Kumar et al.,ICML 2020]

Another technique: What-If Exploration

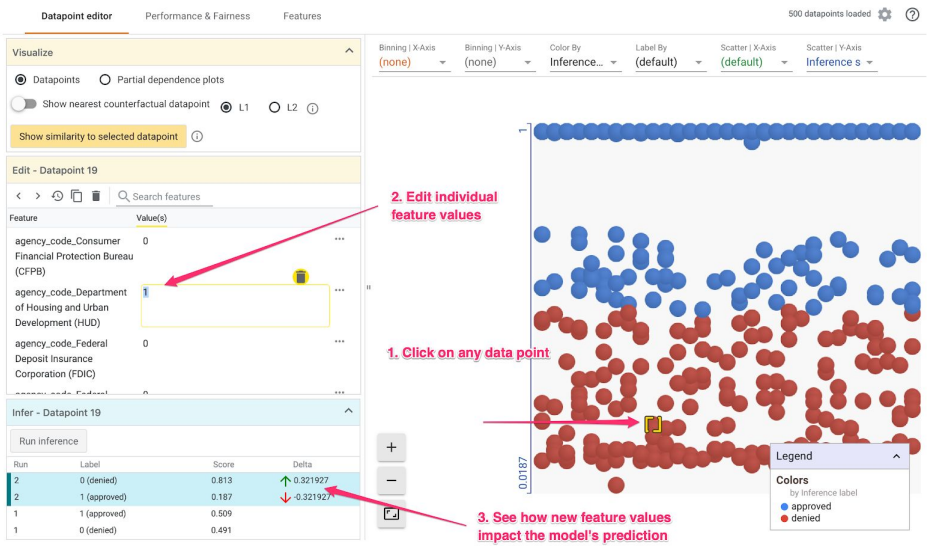
Probe the model on various What-If scenarios.

Examples:

- What if “income” was increased by 20%
- What if “he” was replaced with “she”

Applications:

- Model understanding / debugging
- Algorithmic Recourse



Visual interface offered by [What-if tool]

What-If Exploration: Pros and Cons

Pros:

- Intuitive: What you see is what you get
- Highly expressive: Most explainability techniques can be thought of as a summarization of what-if behavior

Cons:

- Untargeted analysis
  - How to identify what-if scenarios that achieve a target prediction?
- Assessing coverage
  - How to navigate the space of such what-if scenarios?

Can we get the best of both worlds?

- Intuitiveness of What-ifs
- Targeted nature of feature attributions

Problem Statement

Given an input and a prediction target, identify a set of minimal perturbations that achieve the target

- Perturbations defined via drawing features values from a reference distribution
- Minimality is defined via **partial order (⩽)** on the space of perturbations
    - E.g., perturbation {income → 120k} is more preferable (⩽) to {income → 120k, fico → 700}

Technique: Targeted What-Ifs

- Iterate through the space of perturbation in topologically sorted order
- Return perturbations that achieve the prediction target with at least probability τ

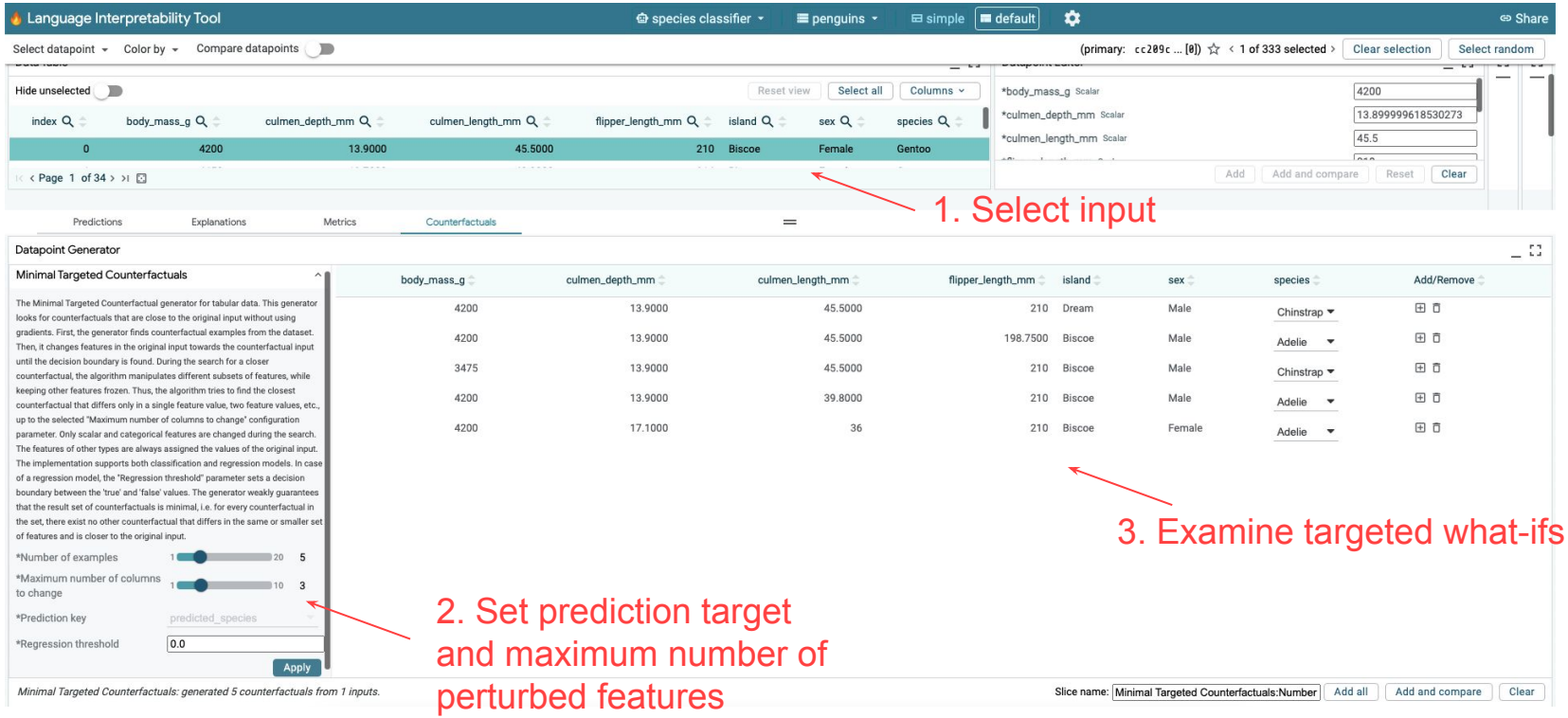
Technique: Targeted What-Ifs

- Iterate through the space of perturbation in topologically sorted order
- Return perturbations that achieve the prediction target with at least probability τ

Paper: Local Explanations via Necessity and Sufficiency: Unifying Theory and Practice, **UAI 2021**

- Frames the problem using the theory of sufficient and necessary causes, and proves a correctness guarantee
- Considers the setting where we only consider perturbations that are feasible according to a causal graph

Targeted What-Ifs supported by Language Interpretability Tool



Case study from a Search team: Detecting Irrelevant Features

Issue: A search model was predicting high pCTR for certain queries paired with an irrelevant result.

Debugging: Identify query token ablations (what-ifs) that lowered the pCTR

Finding: Perturbations identified out-of-vocab (OOV) tokens, e.g., the token “ph8” in query “water filter ph8”

Root cause: Model was not trained well on queries with OOV tokens.

Fix: Increase the vocab threshold (so that more OOV tokens are seen during training) and retrain. This fixed the issue!

Monitoring Models

Why monitor models?

- Production data may differ significantly from test data
- During production, the joint distribution of features and labels may drift over time.
  - Task itself may vary over time, e.g., the definition of spam
  - Outlier events, e.g., pandemic
  - Bugs in feature pipeline

  This is known as **concept drift**

- This may adversely affect the model's performance, uncertainty, and calibration.

How to monitoring models?

Directly tracking various performance metrics (accuracy, fairness, calibration) over time may not be feasible due to absence of groundtruth labels.

In the absence of groundtruth, teams often monitor

- Feature distribution
- Prediction distribution

Feature distribution monitoring

Monitor trend of feature values for each feature

Feature drift: Compare distribution of each feature in a certain serving window with that in a certain reference window (say via KL divergence)

Feature drift detection helps guard against:

- Feature distribution changes due to dynamics of the task
- Feature pipeline bugs

Limitations of feature distribution monitoring

- Dealing with multiple feature representations (e.g., numeric, categorical, embeddings)
- Large feature drift may not always imply large change in performance
- Does not track drift in correlations between features

Alternative: Attribution-Based Monitoring

Attribution-based monitoring

Monitor trend of feature attribution score for each feature

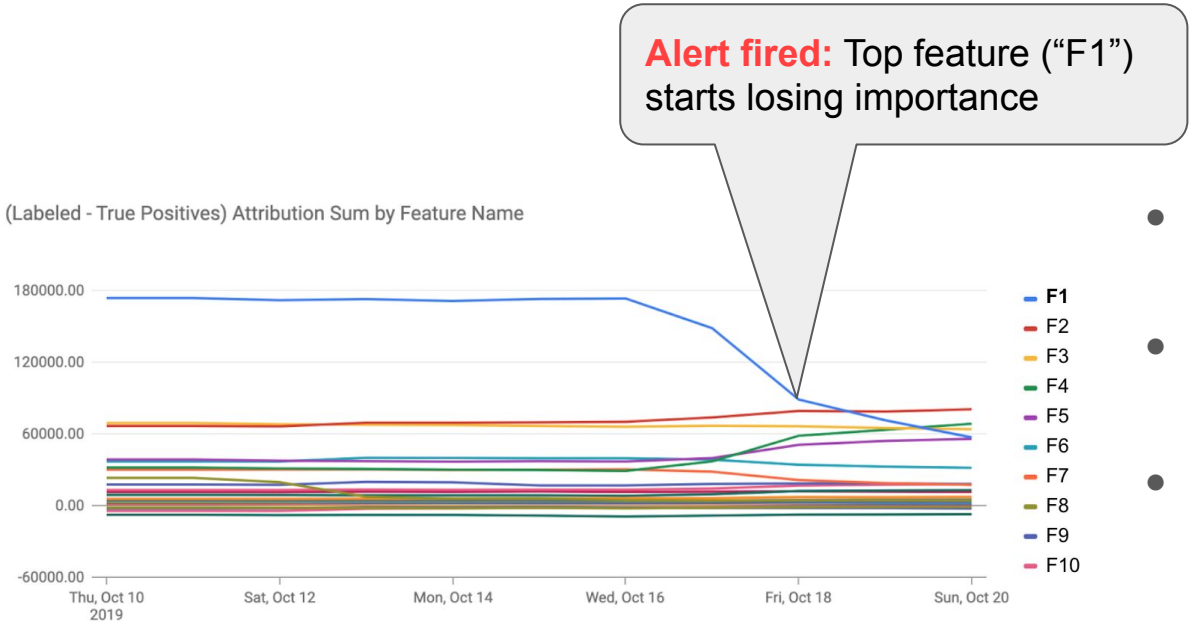
Feature Attribution Drift: Compare distribution of feature attributions from a serving window with those from a certain reference window.

- Computed separately for each feature

Benefits of Attribution-Based Monitoring

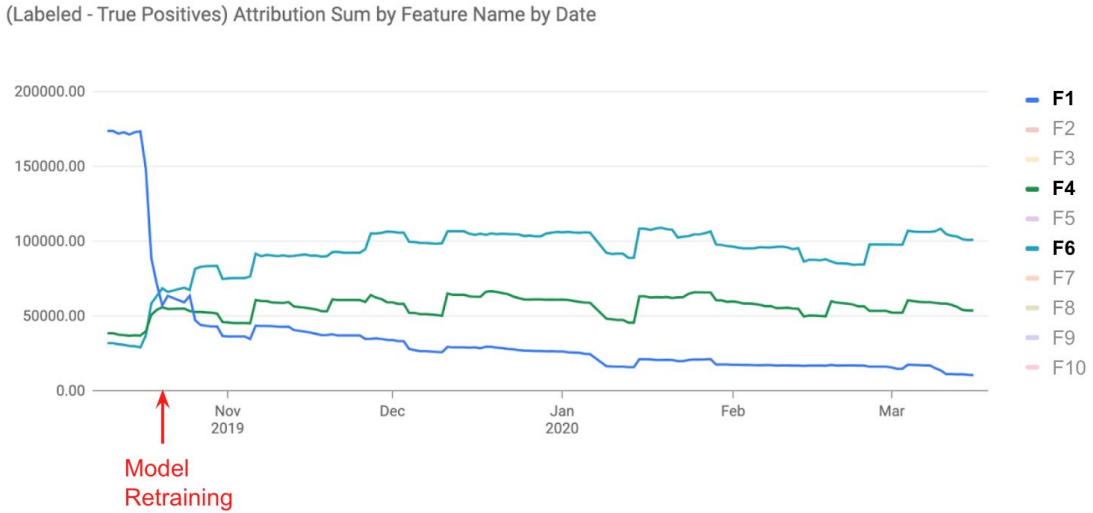
- Inherently importance weighted
- Applicable to all feature representations
- Account for feature interactions
- Can be extended to feature groups
- Enables monitoring stability of feature importances across model versions

Case study from a large-scale ML model at Google



- Attribution monitoring helped quickly surface the issue.
- Triggered retraining of all models that relied on the feature
- (It was later found that the drop was caused by a certain infrastructure change made by the team that owned the feature.)

Case study from a large-scale ML model at Google



Feature F4 and F6 made up for the drop in coverage of F1, leaving downstream services largely unaffected.

Takeaways

- Test accuracy alone can be misleading
  - Examine model performance on slices
  - Assess if test set is representative of deployment
- Probe the model's reasoning on individual predictions
  - Is the model relying on spurious/irrelevant features?
  - Is the model ignoring relevant features?
- Monitor models in production

Thank you for listening! Questions?
(ataly@google.com)

Explanations for the end-user

Diabetic Retinopathy Prediction

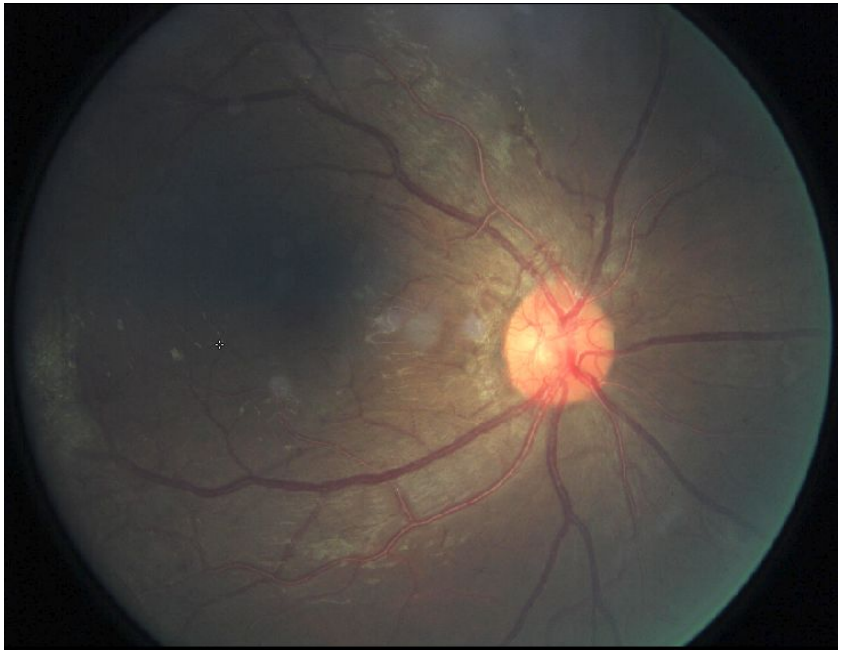
Retinal Fundus Image

⇨ Prediction: **“proliferative”** DR

- Proliferative implies vision-threatening

Can we provide an explanation to the doctor with supporting evidence for **“proliferative”** DR?

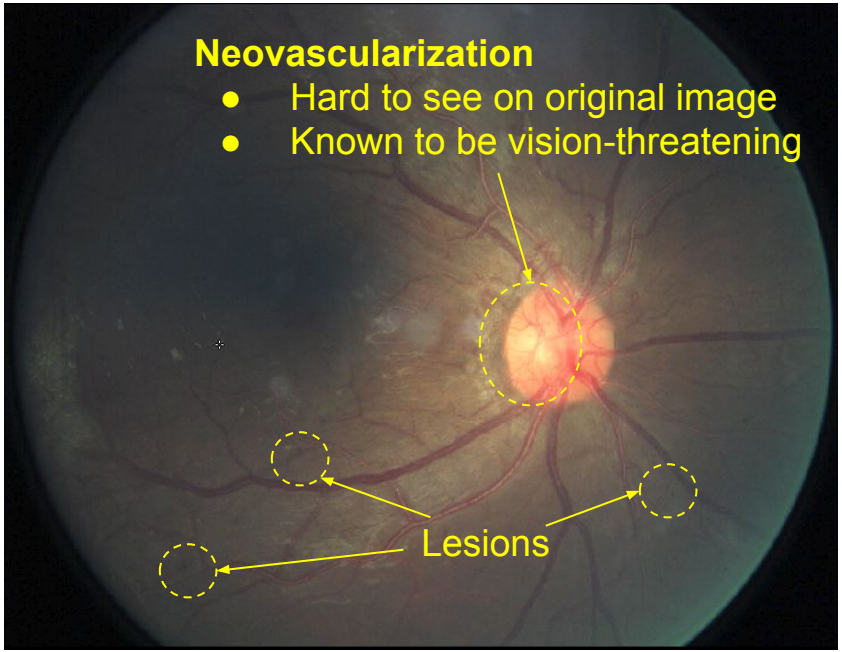
Retinal Fundus Image



Integrated Gradients for label: “proliferative”

Visualization: Overlay heatmap on green channel

Retinal Fundus Image



Integrated Gradients for label: “proliferative”

Visualization: Overlay heatmap on green channel

Assisted Read Study

9 doctors grade 2000 images under three different conditions

A. Image only
B. Image + Model's prediction scores
C. Image + Model's prediction scores + Explanation (Integrated Gradients)

Some findings:

- Seeing prediction scores (B) significantly increases accuracy vs. image only (A)
- Showing explanations (C) only provides slight additional improvement
  - Masks help more when model certainty is low
- Both B and C increase doctor ↔ model agreement

Paper: Using a deep learning algorithm and integrated gradients explanation to assist grading for diabetic retinopathy --- Journal of Ophthalmology [2018]

Efficacy of Explanations

Explanations help when:

- Model is right, and explanation convinces the doctor
- Model is wrong, and explanation reveals the flaw in the model's reasoning

But, Explanations can also hurt when:

- Model is right, but explanation is unintelligible
- Model is wrong, but the explanation convinces the doctor

Be careful about long-term effects too!

Humans and Automation: Use, Misuse, Disuse, Abuse - Parsuraman and Riley, 1997

Evaluating an Attribution Method

Evaluating an Attribution Method

- Ablate top attributed features and examine the change in prediction
  - Issue: May introduce artifacts in the input (e.g., the square below)

- Compare attributions to (human provided) groundtruth on “feature importance”
  - Issue 1: Attributions may appear incorrect because the network reasons differently
  - Issue 2 : **Confirmation bias**

Evaluating an Attribution Method

- Ablate top attributed features and examine the change in prediction
  - Issue: May introduce artifacts in the input (e.g., the square below)

- Compare attributions to (human provided) groundtruth on “feature importance”
  - Issue 1: Attributions may appear incorrect because the network reasons differently
  - Issue 2 : **Confirmation bias**

The mandate for attributions is to be faithful to the network’s reasoning

Our Approach: Axiomatic Justification

- List **desirable criteria (axioms)** for an attribution method
- Establish a uniqueness result: X is the **only** method that satisfies these criteria

Axioms

- **Insensitivity:** A variable that has no effect on the output gets no attribution
- **Sensitivity**: If baseline and input differ in a single variable, and have different outputs, then that variable should receive some attribution
- **Linearity preservation**: Attributions(α*F1 + ß*F2) = α*Attributions(F1) + ß*Attributions(F2)
- **Implementation invariance**: Two networks that compute identical functions for all inputs get identical attributions
- **Completeness**: Sum(attributions) = F(input) - F(baseline)
- **Symmetry**: Symmetric variables with identical values get equal attributions

Result

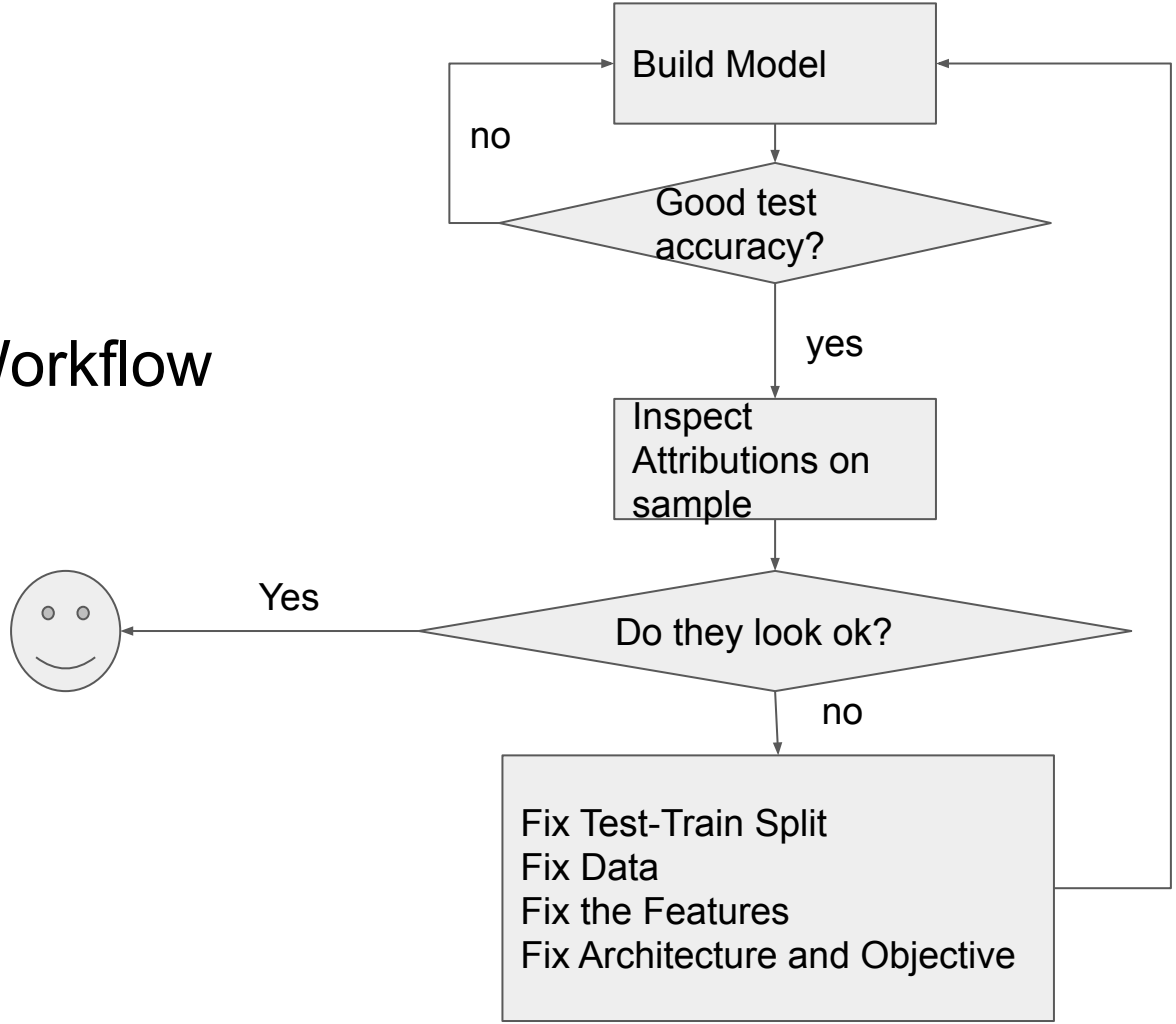
Theorem [ICML 2017]: Integrated Gradients is the **unique** path-integral method satisfying: Sensitivity, Insensitivity, Linearity preservation, Implementation invariance, Completeness, and Symmetry

Historical note:

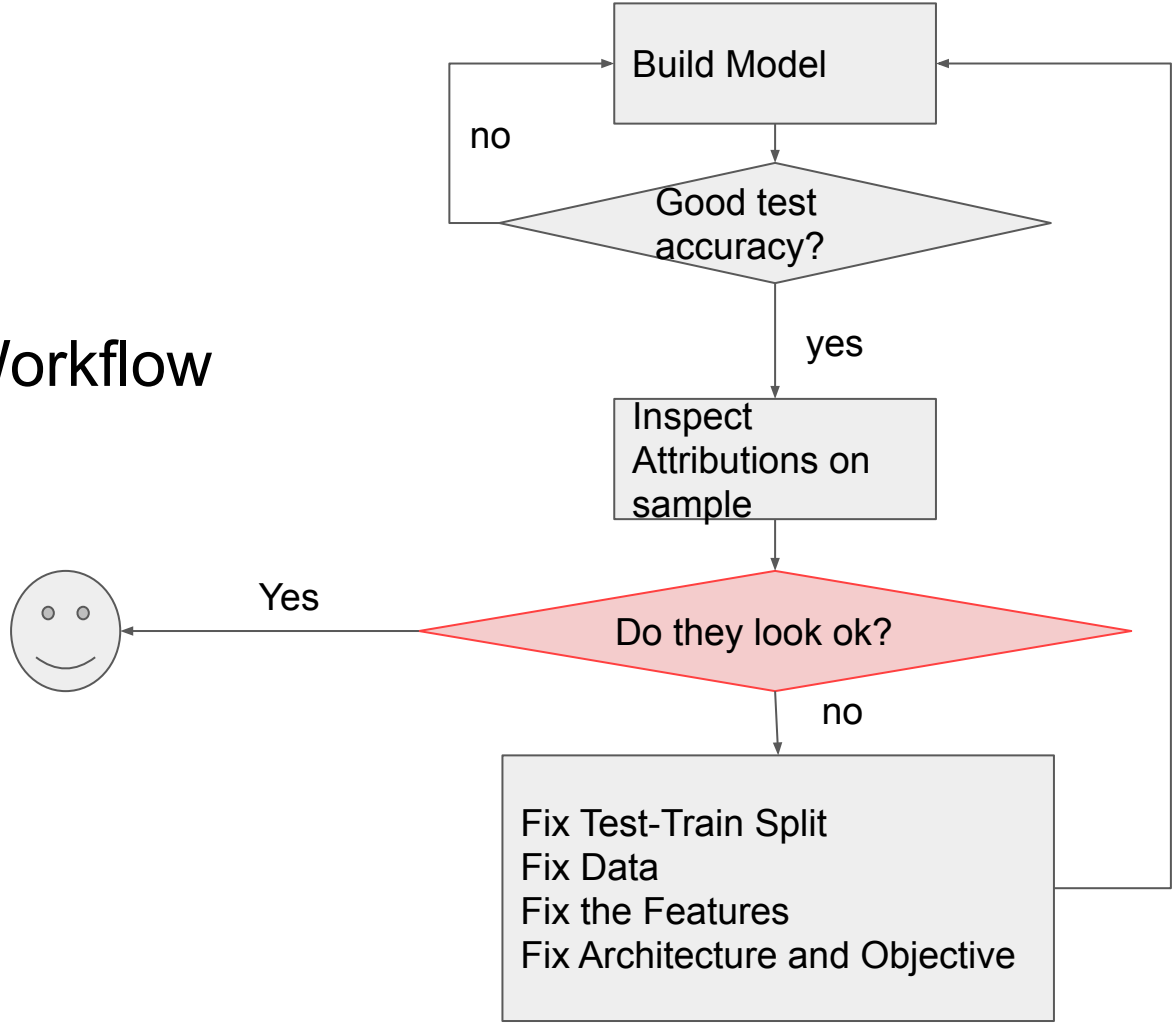
- Integrated Gradients is the **Aumann-Shapley method** from cooperative game theory, which has a similar characterization; see [Friedman 2004]

Some limitations and caveats

Debugging Workflow



Debugging Workflow



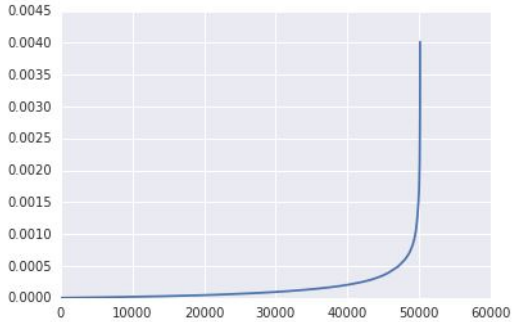
Role of the Analyst

- Humans are poor at foreseeing problems
- Humans excel at understanding real world implications of specific explanations
  - Disease prediction: "Pen marks won’t be available on X-rays in deployment"
  - Question answering: "most words in a question matter"
- Proper visualization is very important in making attributions intelligible to humans

Importance of Visualization

Naive scaling of attributions from 0 to 255

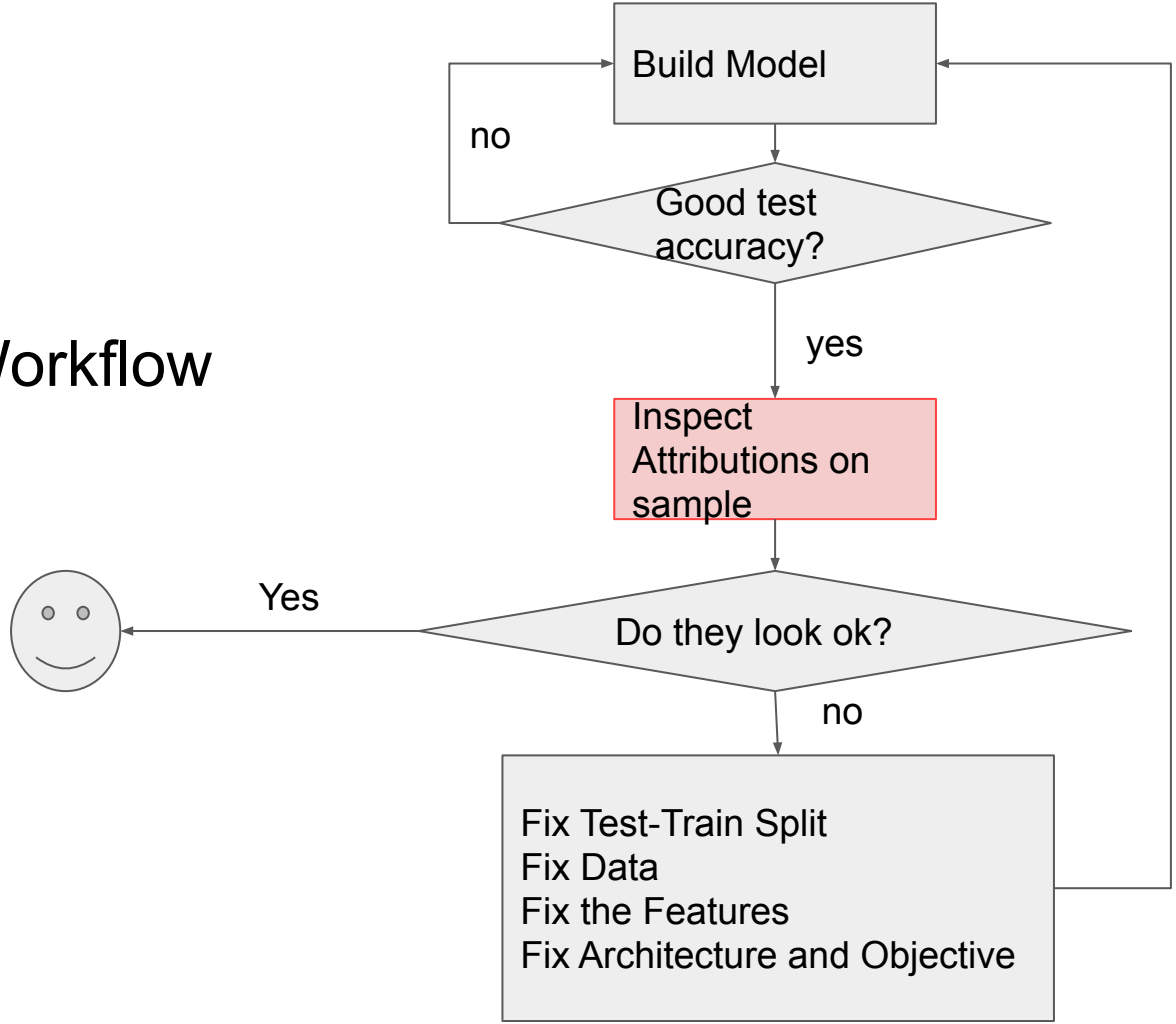
Attributions have a **large range** and **long tail** across pixels



After clipping attributions at 99% to reduce range

Paper: Exploring Principled Visualizations for Deep Network Attributions, IUI Workshop 2019

Debugging Workflow



Attributions are pretty shallow

Attributions do not explain:

- How the network combines the features to produce the answer?
- What training data influenced the prediction
- Why gradient descent converged
- etc.

An instance where attributions are useless:

- A network that predicts TRUE when there are **even number** of black pixels and FALSE otherwise

Attributions are useful when the network behavior entails that a strict subset of input features are important