Visualization for machine learning

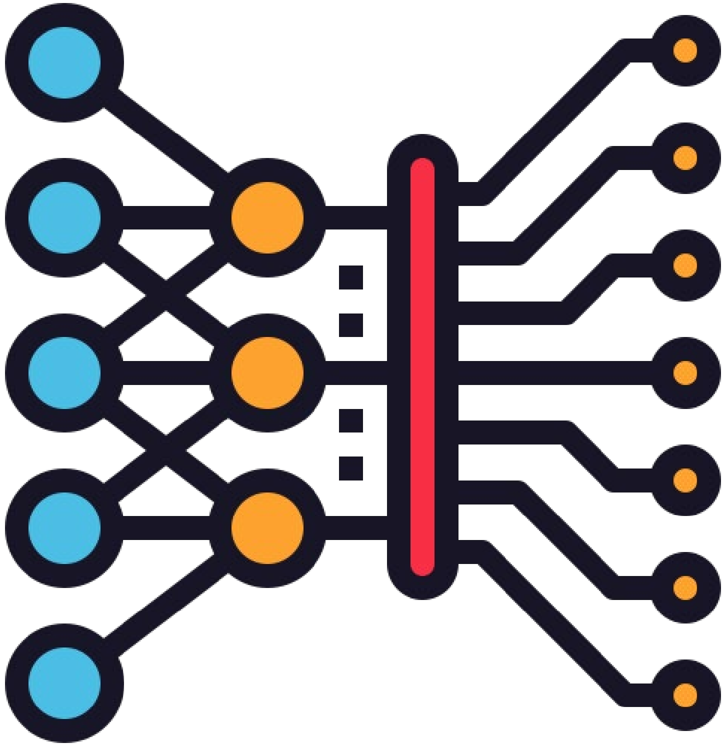
CS524: Big Data Visualization & Visual Analytics

Fabio Miranda

https://fmiranda.me

Slides based on Claudio Silva's ml+vis course

Deep learning is everywhere



But what is <u>*actually*</u> going on in DL models?

Deep learning models are complex

Deep learning models are often complex, application-driven, and hard to investigate.

Visualization is crucial for understanding deep learning models.

What does visualization in deep learning look like?

Visual Analytics in Deep Learning | Interrogative Survey Overview

§4 WHY

Why would one want to use visualization in deep learning?

Interpretability & Explainability
Debugging & Improving Models
Comparing & Selecting Models
Teaching Deep Learning Concepts

§6 WHAT

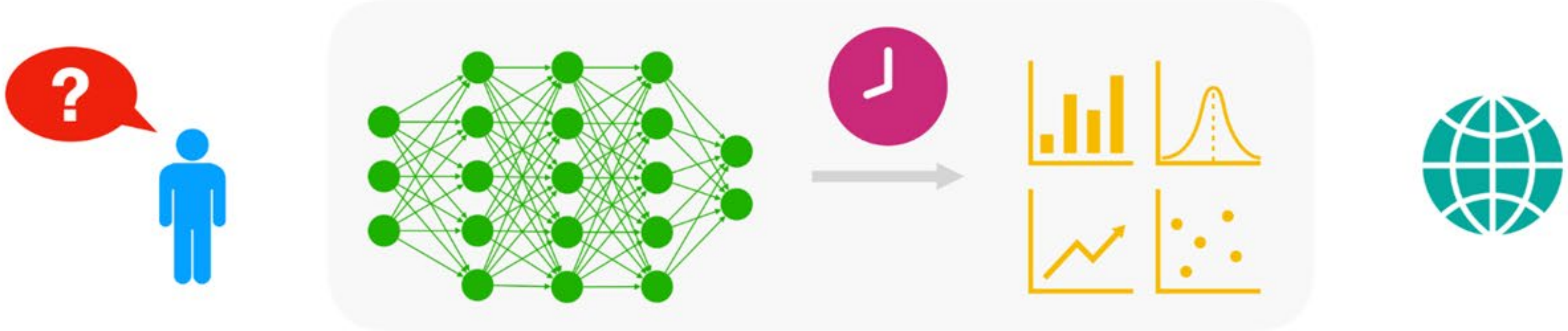
What data, features, and relationships in deep learning can be visualized?

Computational Graph & Network Architecture
Learned Model Parameters
Individual Computational Units
Neurons In High-dimensional Space
Aggregated Information

§8 WHEN

When in the deep learning process is visualization used?

During Training
After Training



§5 WHO

Who would use and benefit from visualizing deep learning?

Model Developers & Builders
Model Users
Non-experts

§7 HOW

How can we visualize deep learning data, features, and relationships?

Node-link Diagrams for Network Architecture
Dimensionality Reduction & Scatter Plots
Line Charts for Temporal Metrics
Instance-based Analysis & Exploration
Interactive Experimentation
Algorithms for Attribution & Feature Visualization

§9 WHERE

Where has deep learning visualization been used?

Application Domains & Models
A Vibrant Research Community

Hohman, F., Kahng, M., Pienta, R., & Chau, D. H. (2018). Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE transactions on visualization and computer graphics*, *25*(8), 2674-2693.

Explainability

Attribution for Pixels

Pixel attribution/Saliency maps are a unique case of feature attribution for image classifiers.

Image classifiers produce $S(I) = [S_1(I), S_2(I), ..., S_{|C|}(I)]$

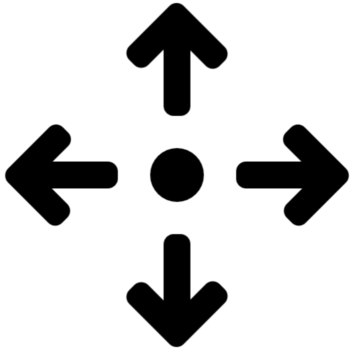
Pixel attribution methods take $x \in \mathbb{R}^p$ as input

And output a relevance score

$$R^c = [R_1^c, R_2^c, ..., R_p^c]$$

Saliency Map Approaches

Gradient-Based



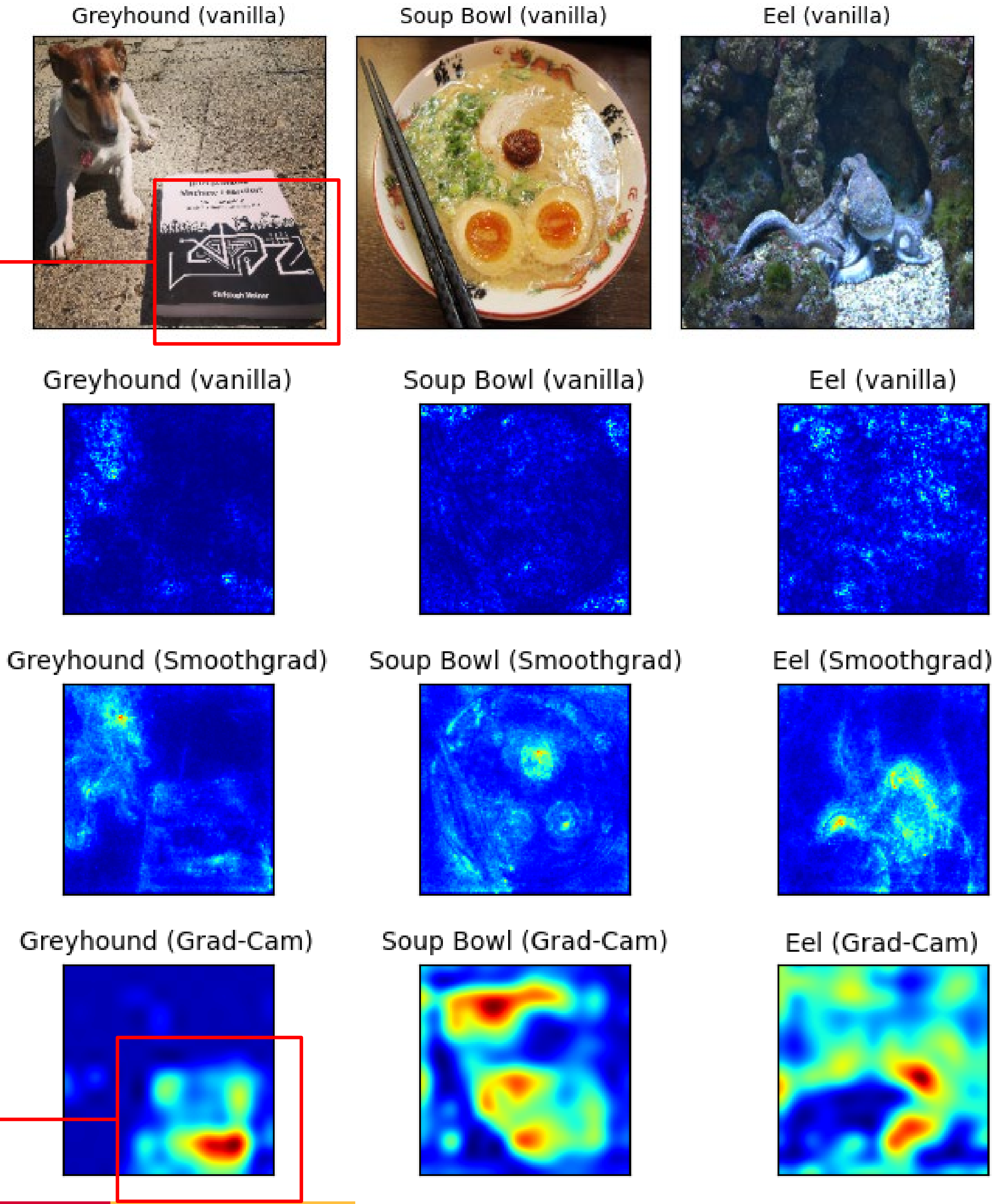
Perturbation-Based

Saliency Map Approach

Step 1: Pass the image through the network.

Step 2: Compute the gradient of the class score w.r.t the input pixels.

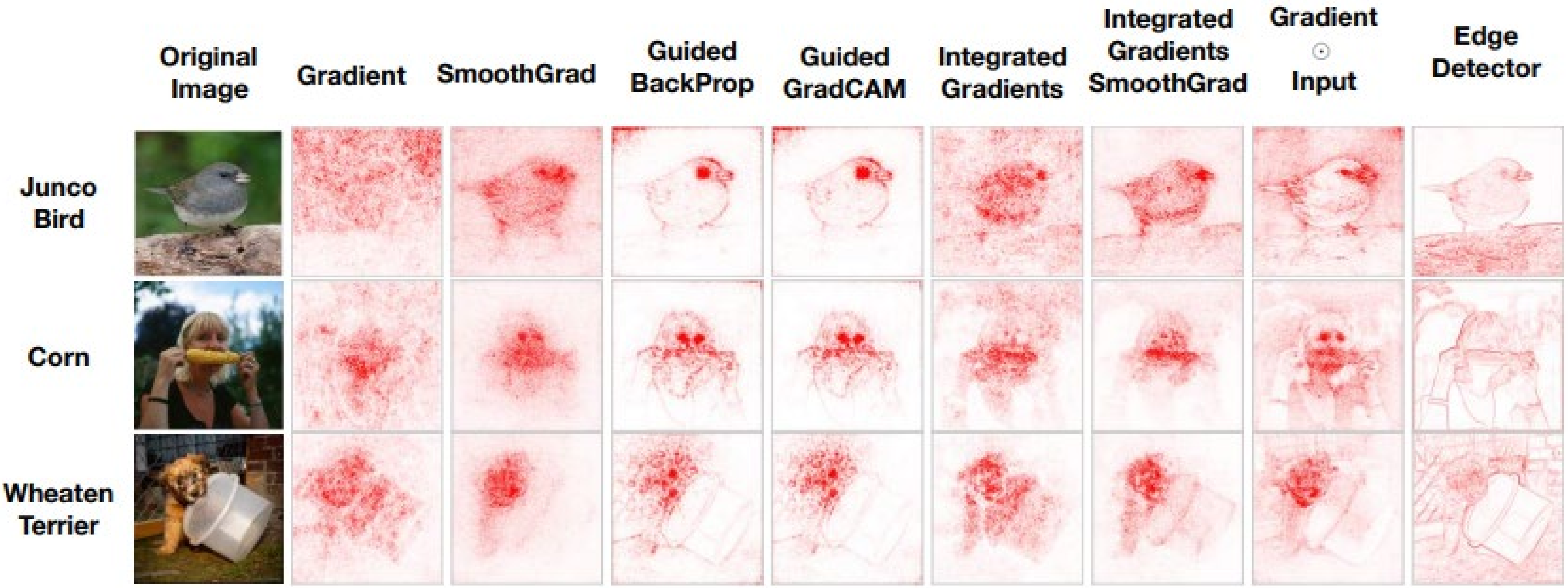
Step 3: Visualize the gradients (either by taking the absolute values or visualizing positive/negative values separately).



Indicating the book, rather than the dog!

Example taken from Christoph Molnar's online "Interpretable Machine Learning" book.

Differences in Pixel Attribution Output



Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., & Kim, B. (2018). Sanity checks for saliency maps. *Advances in neural information processing systems*, *31*.

Pixel Attribution Takeaways

There can be significant variation in the output of different saliency map methods.

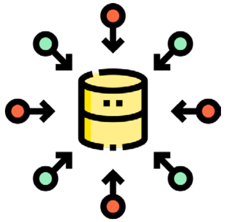
Yet, saliency maps still provide an easy-to-digest local explanation method for images.

Debugging and Summarization

Summit: Scaling Deep Learning Interpretability

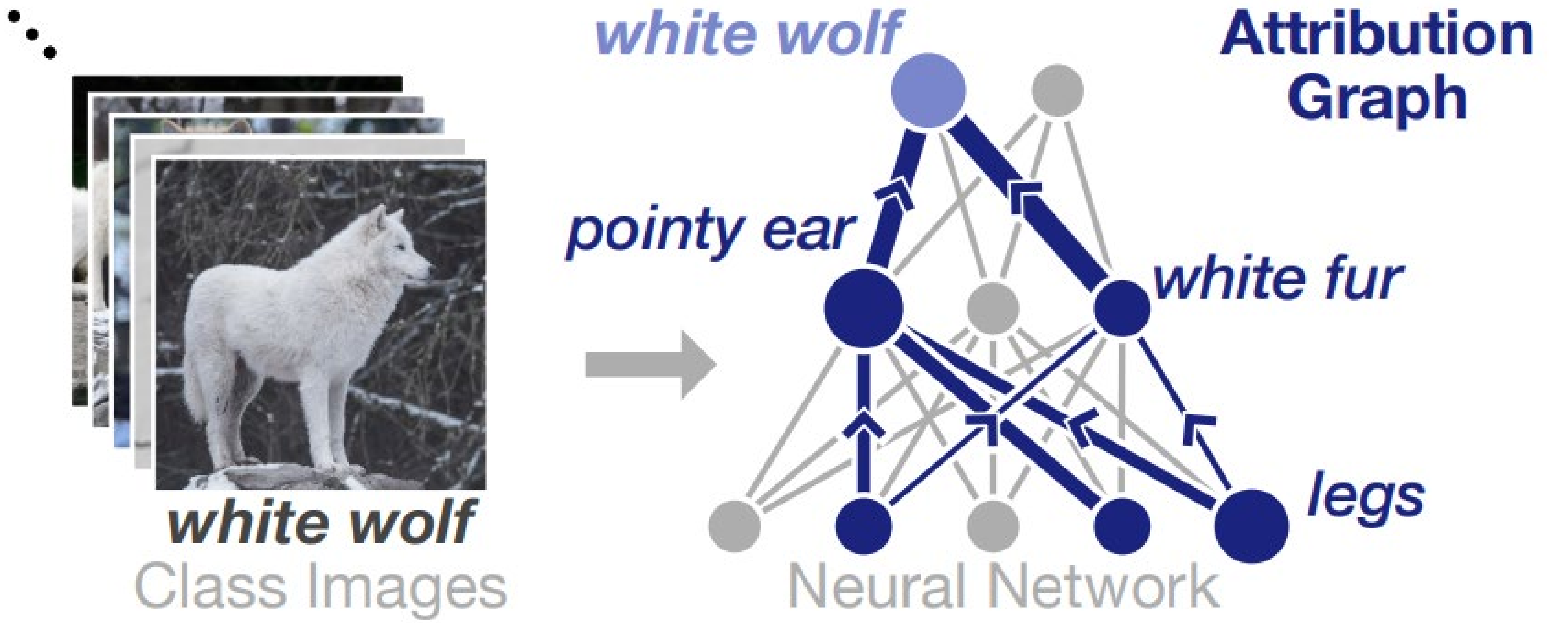
Understanding how a DL model generates a prediction is tricky.

Summit is a visual analytics system to summarize and visualize what features a DL model is learning.

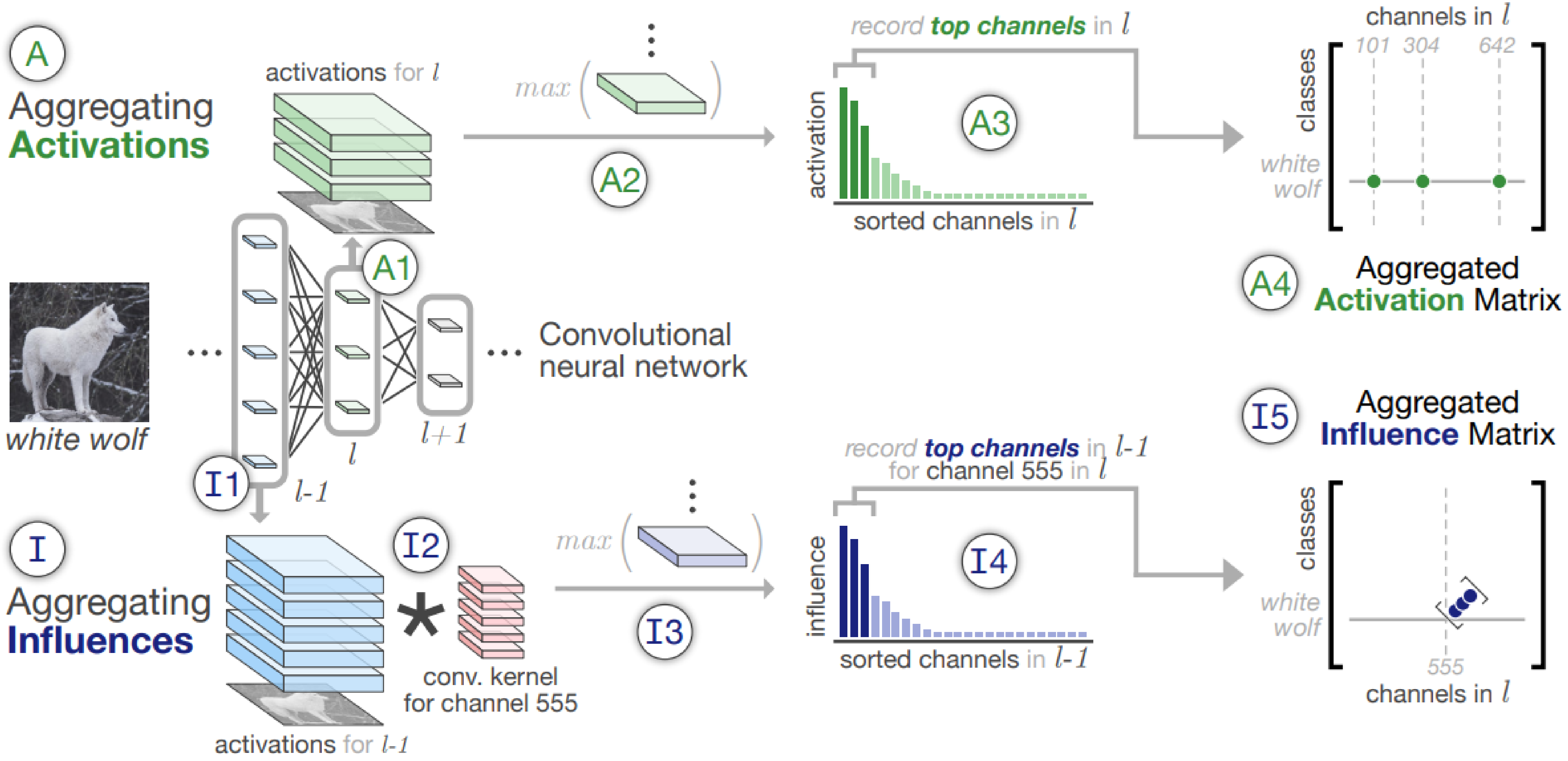


Instead of local attributions, Summit provides a more aggregated view of what a model is learning.

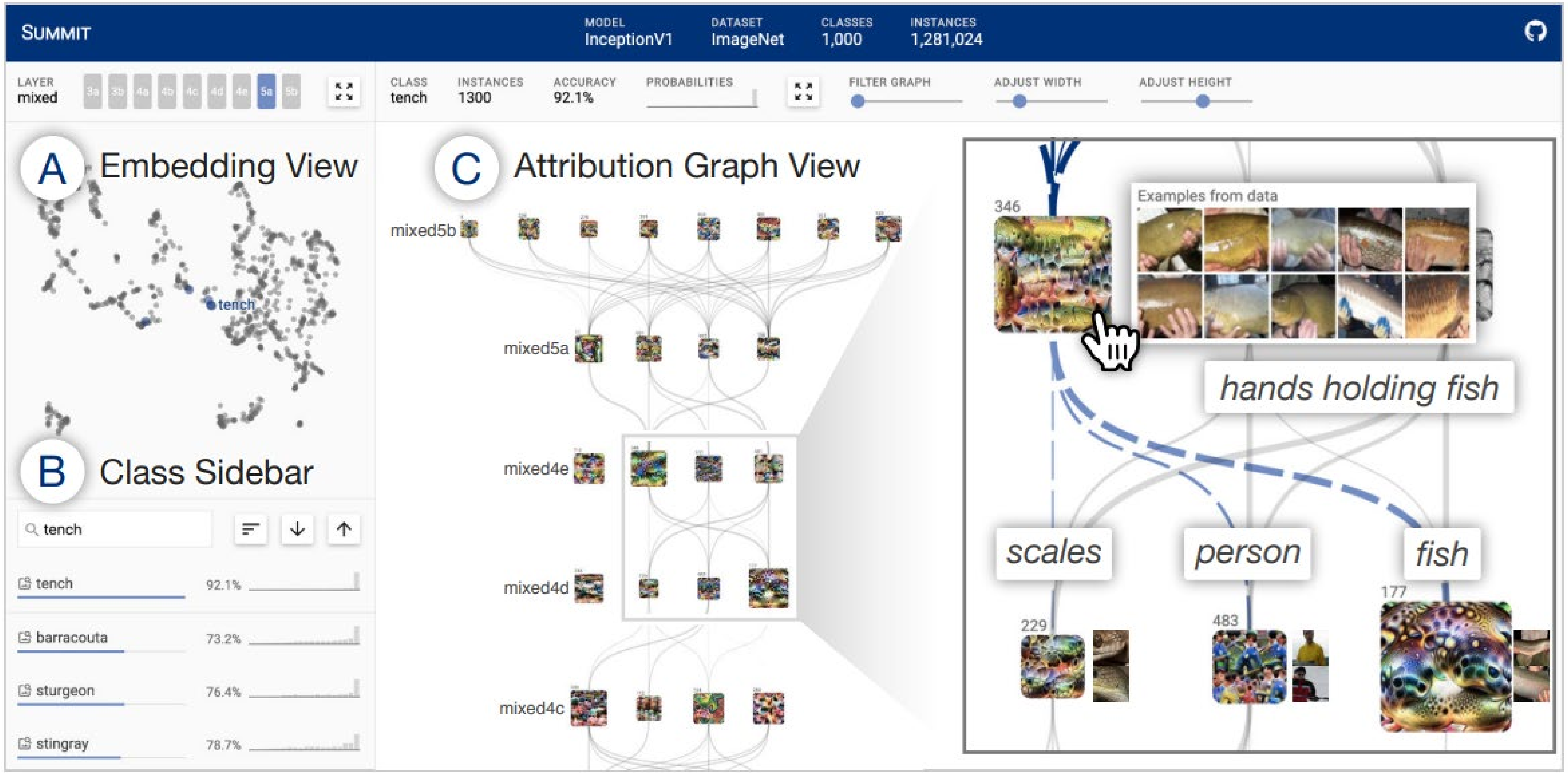
Attribution Graph



Activation and Influence Aggregation

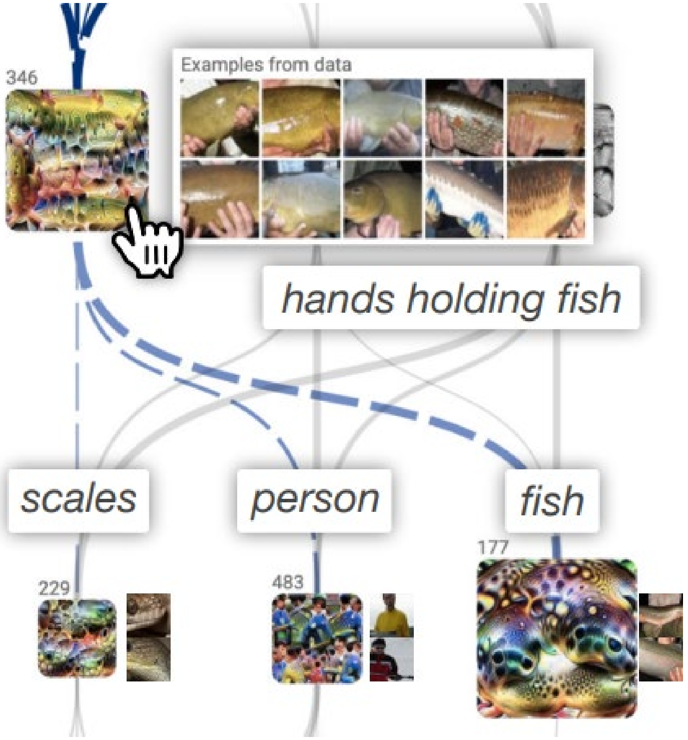


Summit: Scaling Deep Learning Interpretability



Summit Use Cases: Unexpected Semantics

Tench Fish



The model is learning *hands* in early layer but *not* the fish!

Tench is a common fish to be caught for sport.

UIC COMPUTER SCIENCE

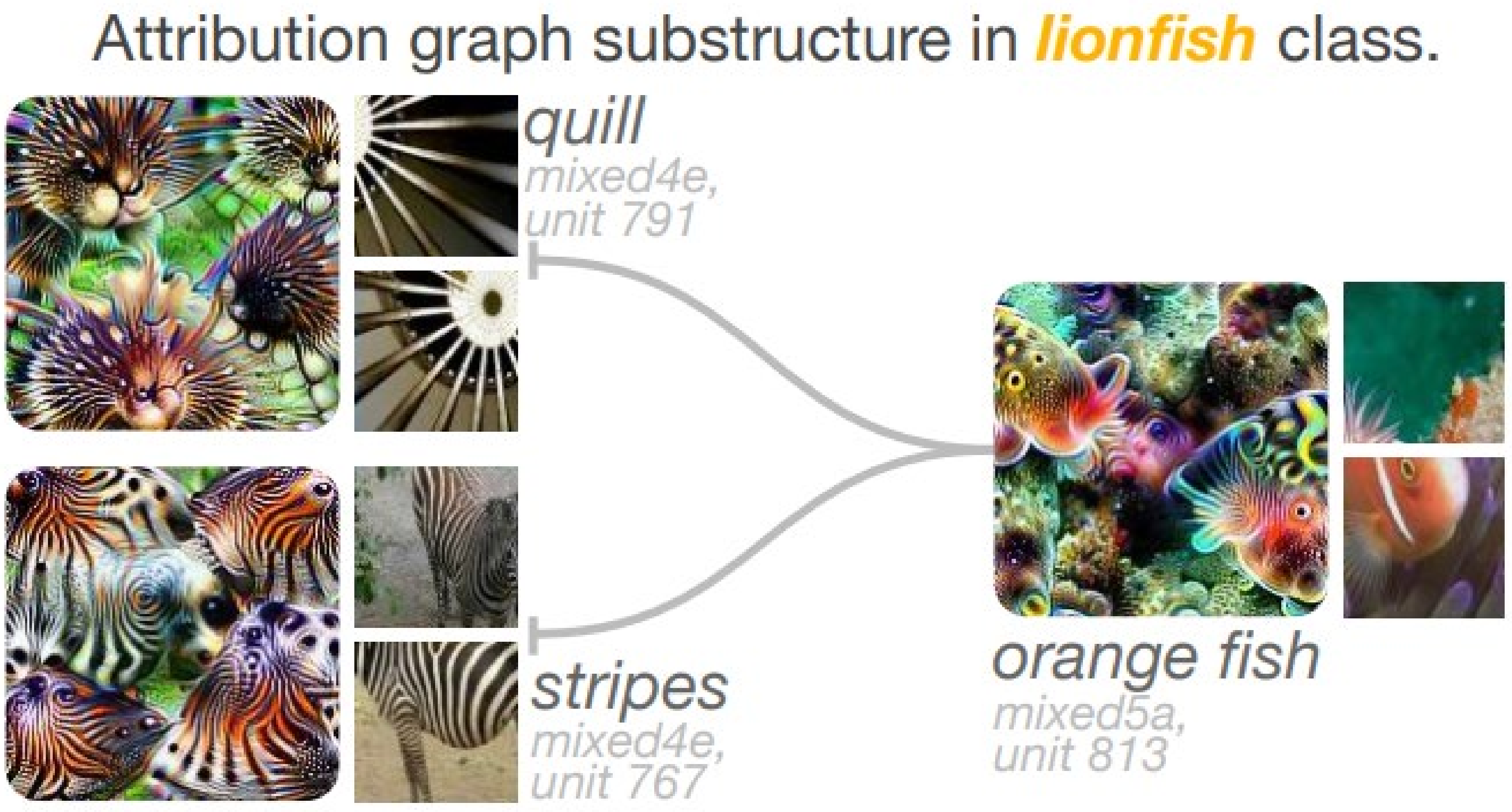
Would you hold this lionfish?

Would you hold this lionfish?

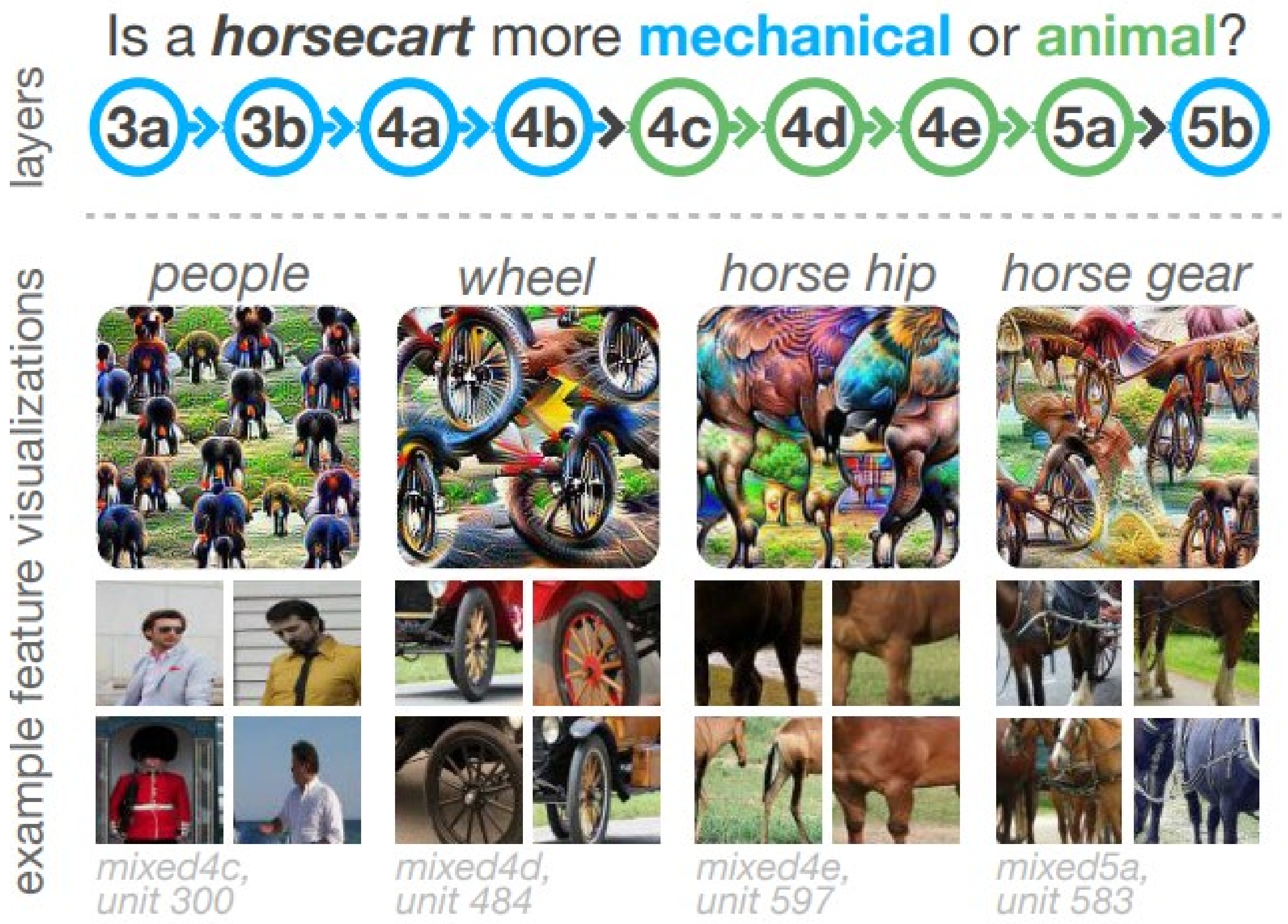
Lionfish have venomous fins and are hazardous to divers and fishermen.

No hands here!

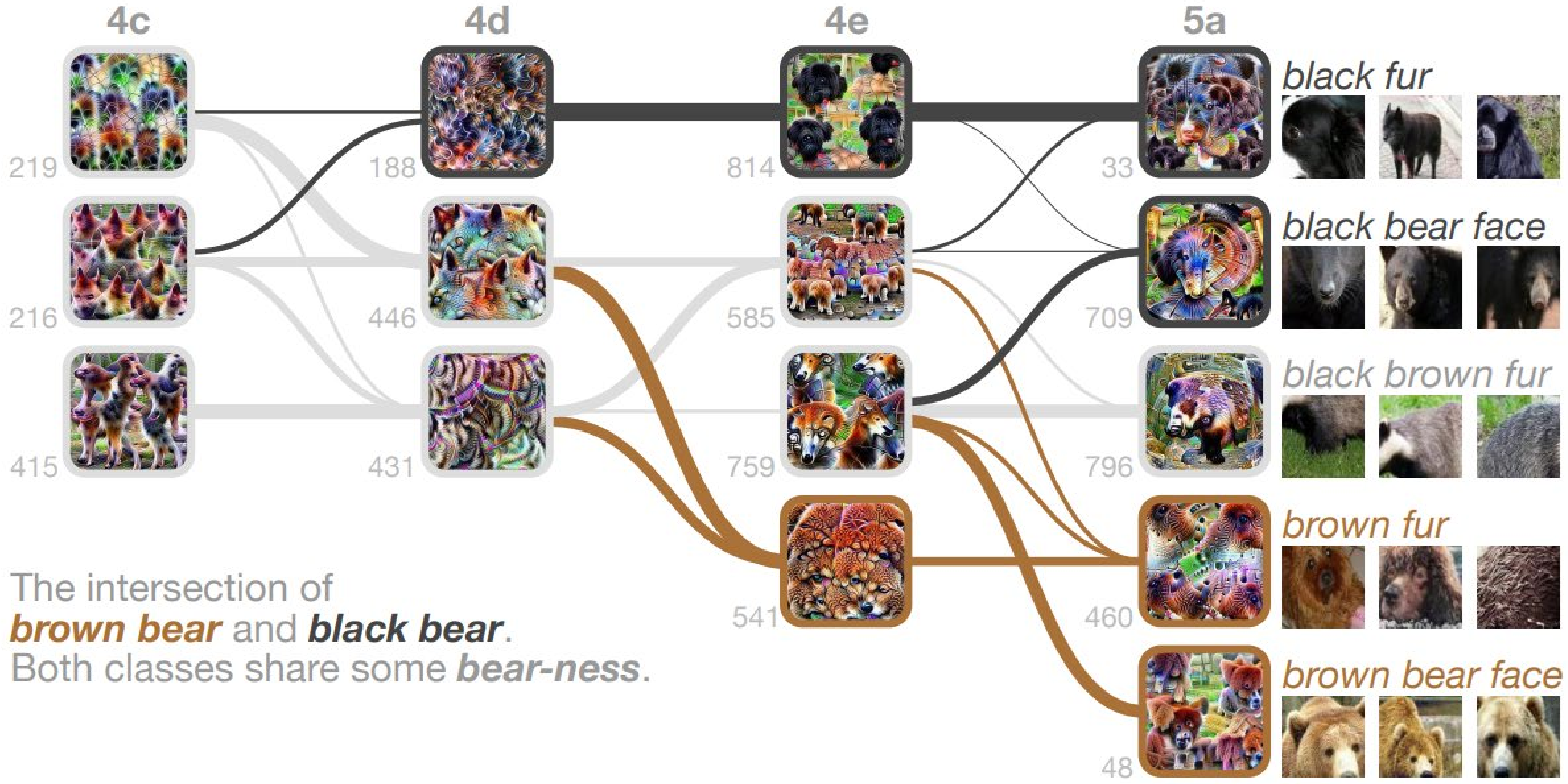
Summit Use Cases: Unexpected Semantics



Summit Use Cases: Mixed Class Association through Layers



Summit Use Cases: Discriminable Features in Similar Classes



Teaching
Deep Learning Concepts

CNN Explainer

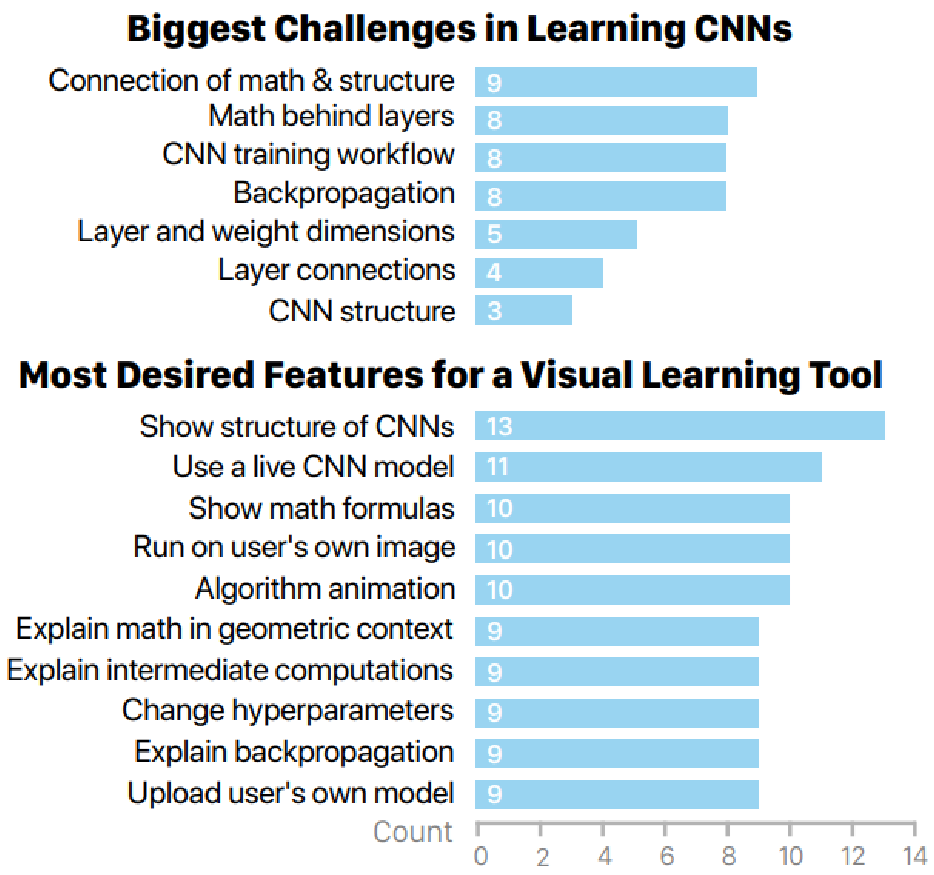
Understanding deep learning is challenging, especially for beginners.

CNN Explainer is a system which helps understand CNNs, which are often taught in intro DL classes and used in practice.

CNN Explainer was designed in conjunction with instructors and students

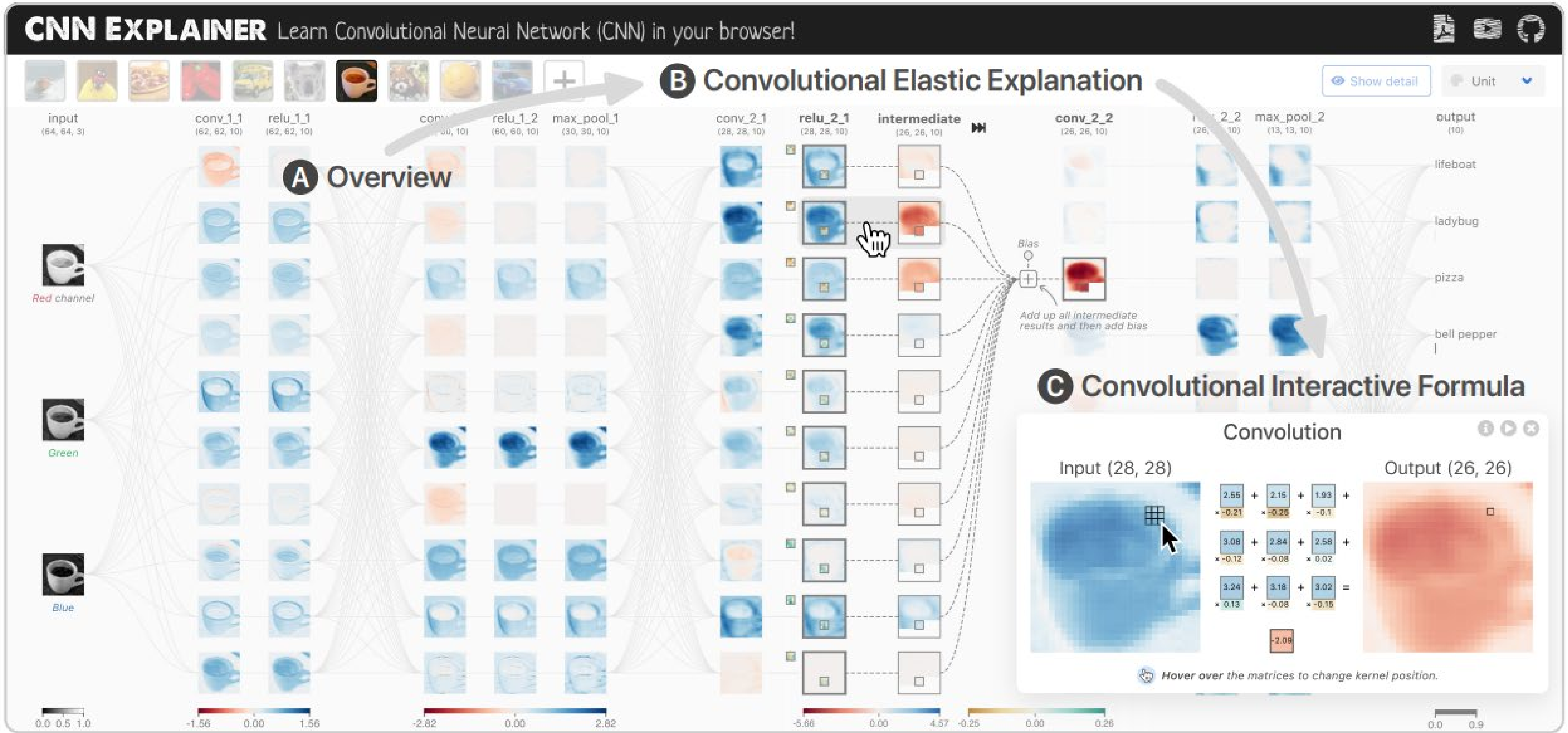
Wang, Z. J., Turko, R., Shaikh, O., Park, H., Das, N., Hohman, F., ... & Chau, D. H. P. (2020). CNN explainer: learning convolutional neural networks with interactive visualization. *IEEE Transactions on Visualization and Computer Graphics*, *27*(2), 1396-1406.

Learning DL is hard!

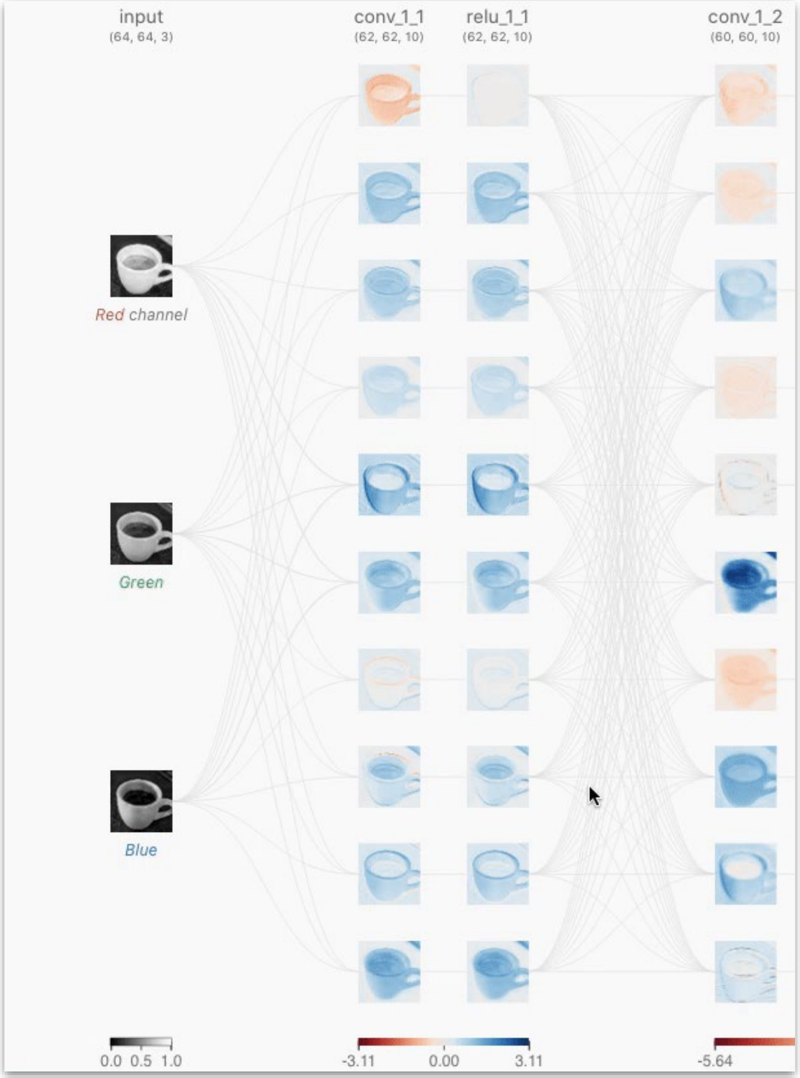


UIC COMPUTER SCIENCE

CNN Explainer

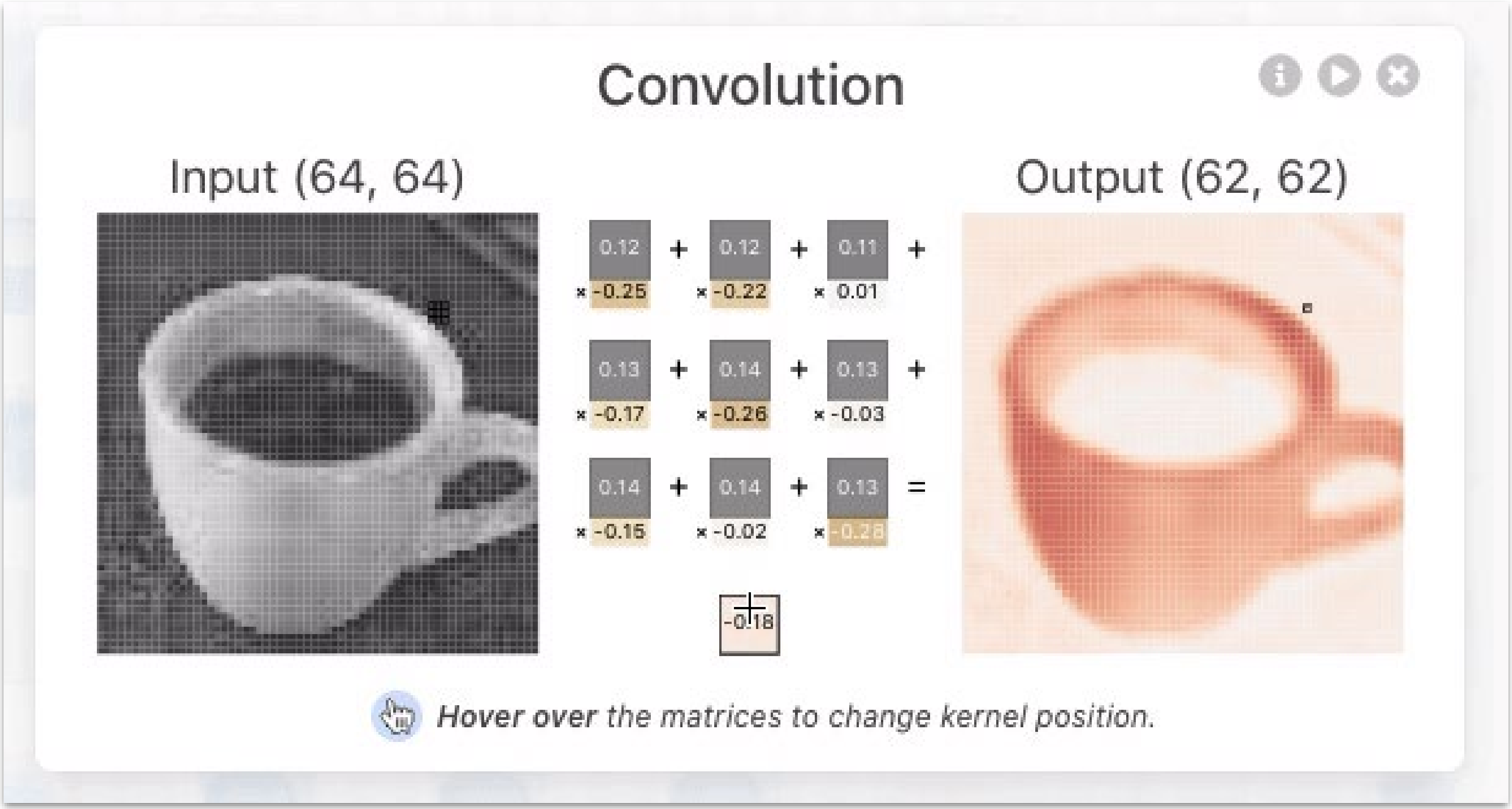


CNN Explainer

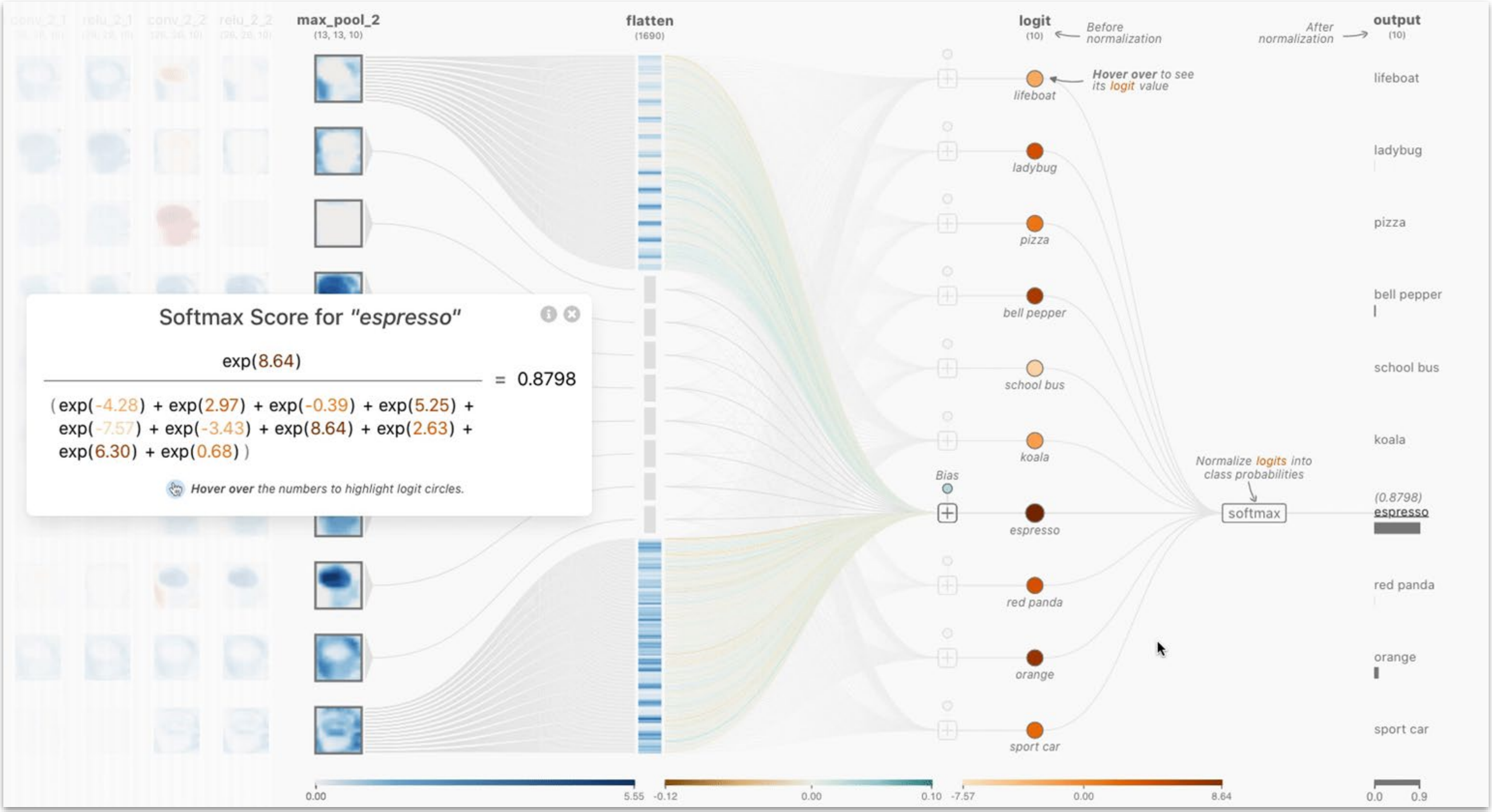


COMPUTER SCIENCE

CNN Explainer



CNN Explainer



CNN Explainer Takeaways

Users found the connection between model structure and low-level mathematical operations helpful.

Animations improve engagement and aid in navigation,

Customization facilitates engagement and hypothesis testing.

Libraries for DL Visualization

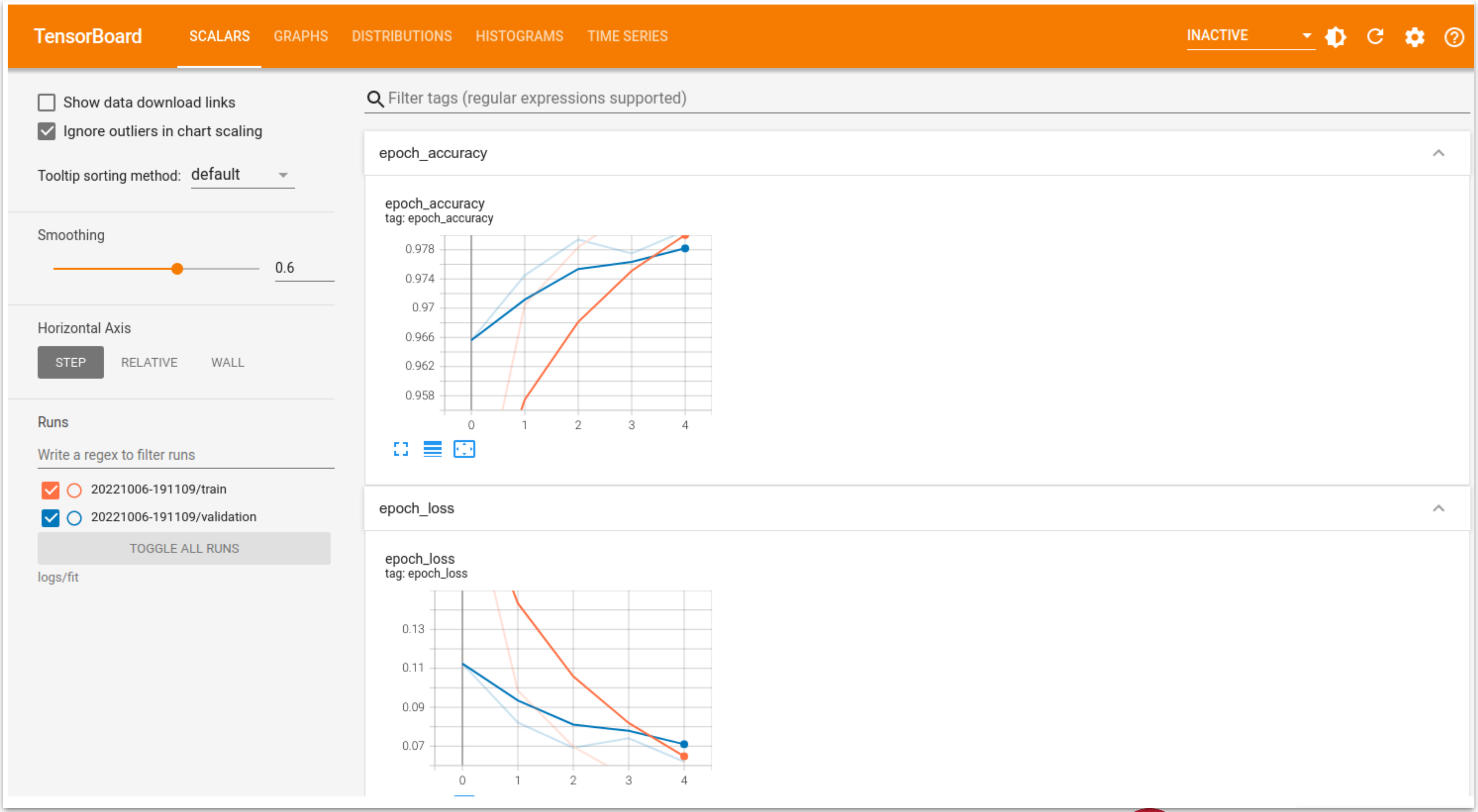
Tensorboard

Improved visualization capabilities are now packaged in many popular DL libraries, like TensorFlow.
TensorBoard is a visual analytics system that allows one to:

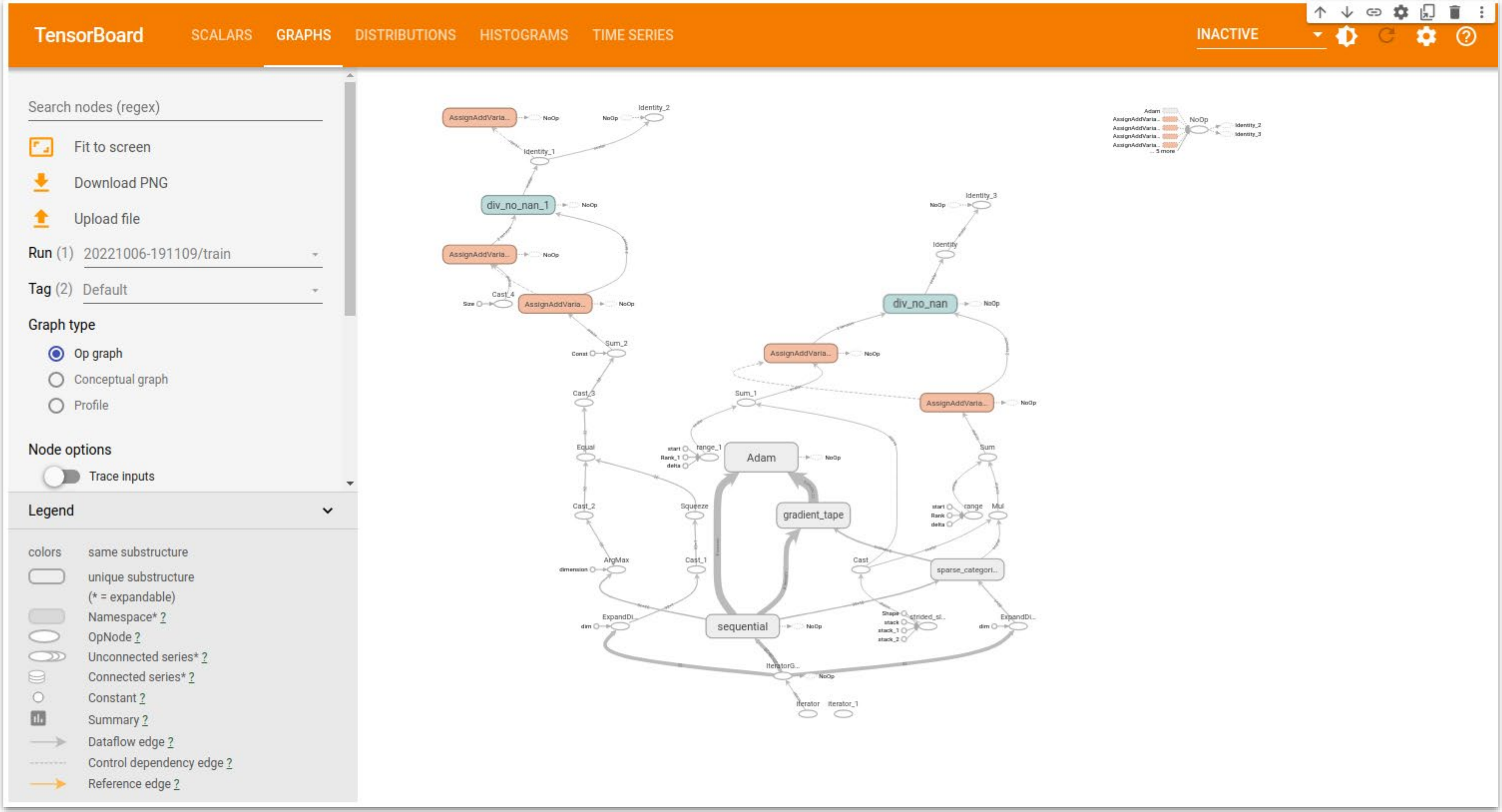
- Track and visualize metrics
- Visualize the operation graph
- Track weights over time
- Project inputs into low dimensions

Tensorboard

```
%tensorboard --logdir logs/fit
```

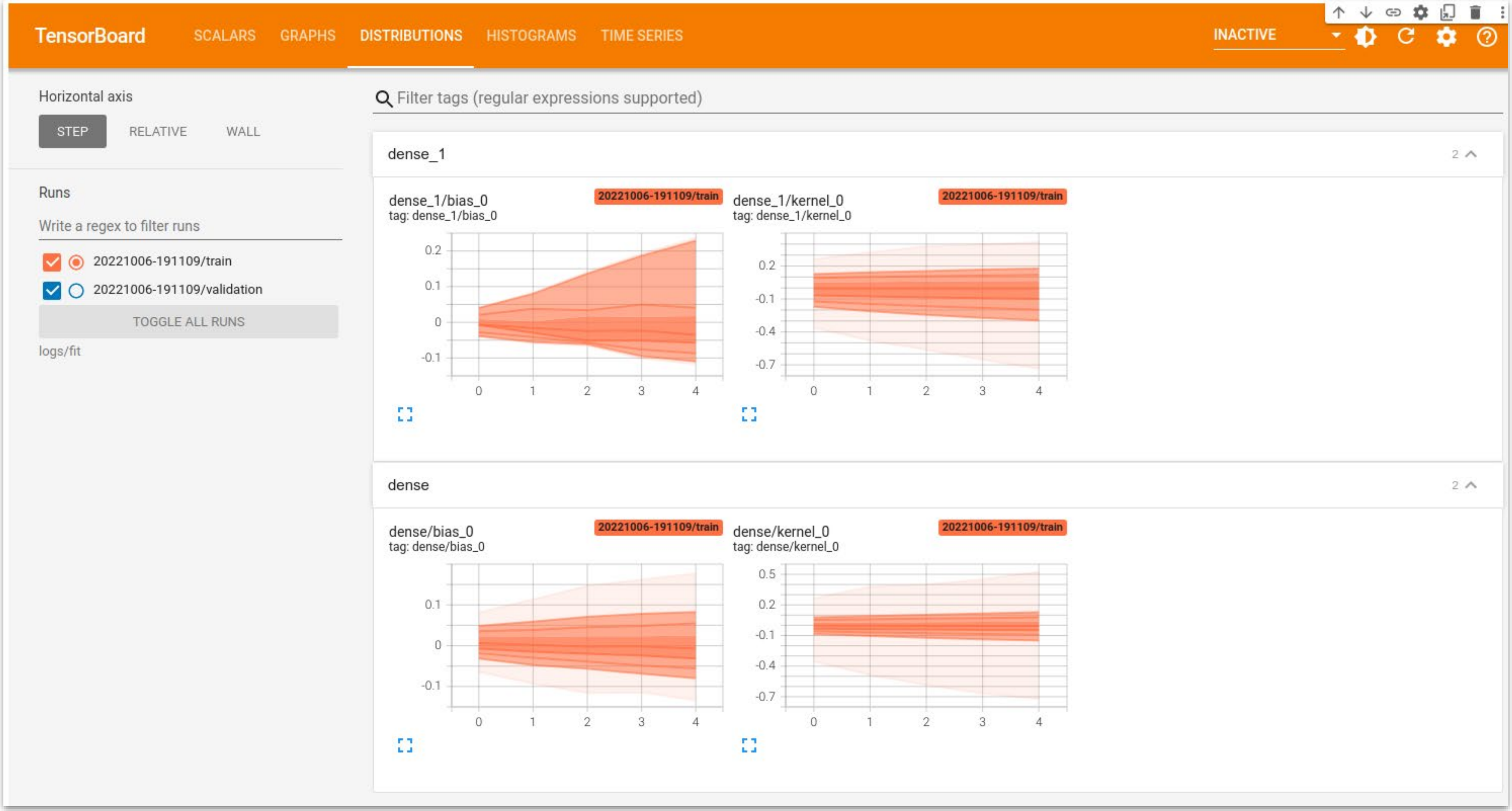


Tensorboard (Operation Graph)

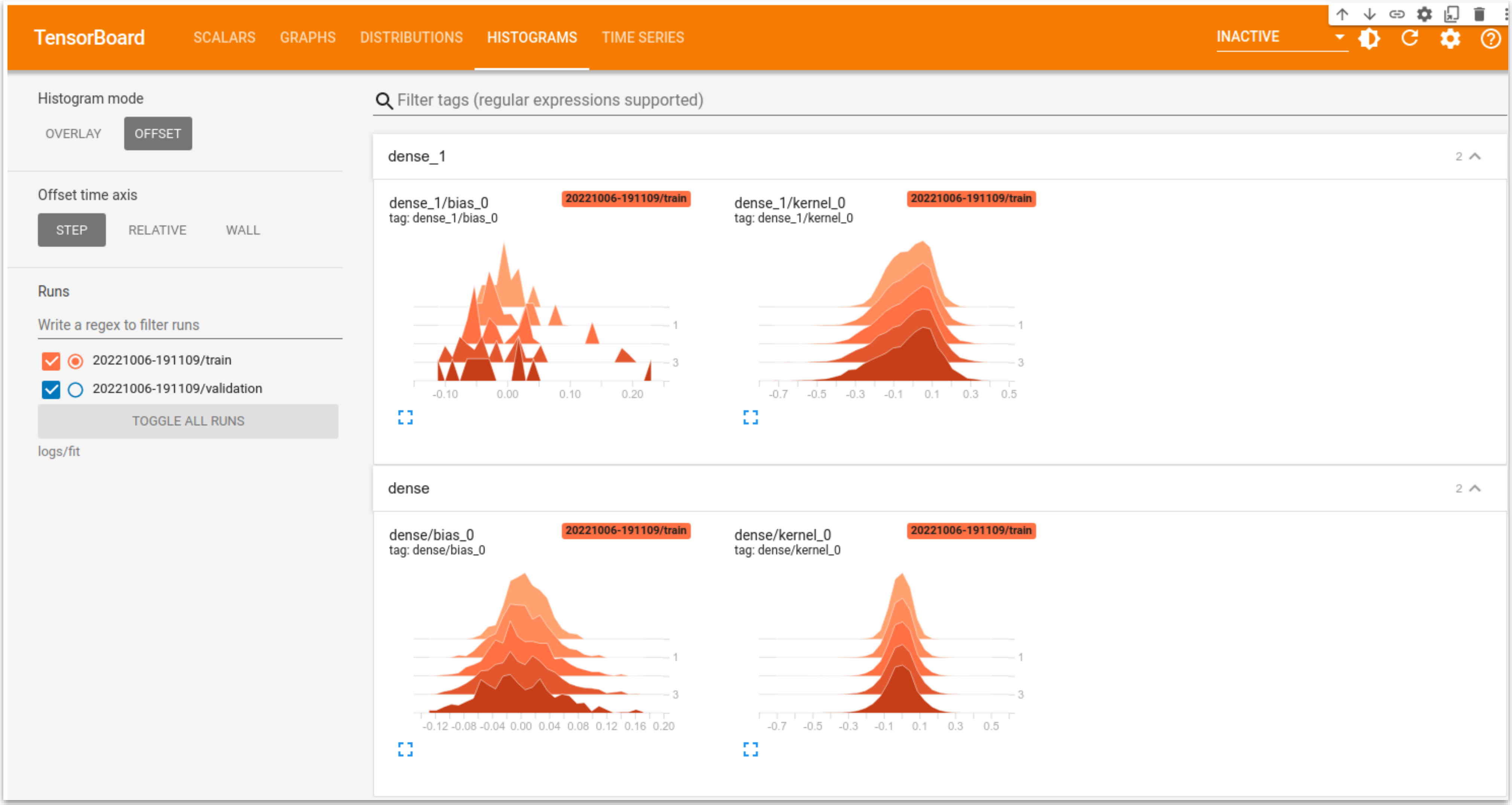


UIC COMPUTER SCIENCE

Tensorboard (Distribution View)



Tensorboard (Histogram View)



Uses of TensorBoard

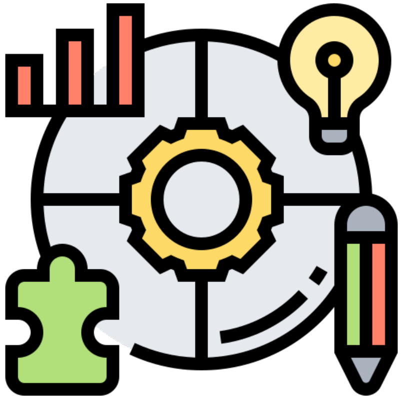
Visualizing the loss or gradients can help adjust the learning rate. Also helpful to see such values live.

Visualizing the computation graph can help identify that the model is doing what the practitioner intends.

Visualizing weights over time can help spot issues such as poor initializations.

Model assessment

We will show methods. We will show systems.



Methods

Confusion matrices

ROC Curves

Calibration

Systems

Squares

Confusion Wheel

Calibrate

Accuracy is simple.

Where does it fail?

Scenario: Disease Prediction

Consider a disease prediction model.

Suppose the hypothetical disease has a 5% prevalence in the population.

The given model converges on the solution of predicting that *nobody* has the disease (i.e., the model predicts “0” for every observation).

Our model is 95% accurate. Yet, public health officials are stumped.

Extended Confusion Matrix

Sources: [20][21][22][23][24][25][26][27] view · talk · edit

		Predicted condition			
	Total population $= P + N$	**Positive (PP)**	**Negative (PN)**	Informedness, bookmaker informedness (BM) $= TPR + TNR - 1$	Prevalence threshold (PT) $= \frac{\sqrt{TPR \times FPR} - FPR}{TPR - FPR}$
Actual condition	**Positive (P)**	**True positive (TP),** **hit**	**False negative (FN),** **type II error, miss, underestimation**	True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power $= \frac{TP}{P} = 1 - FNR$	False negative rate (FNR), miss rate $= \frac{FN}{P} = 1 - TPR$
Actual condition	**Negative (N)**	**False positive (FP),** **type I error, false alarm, overestimation**	**True negative (TN),** **correct rejection**	False positive rate (FPR), probability of false alarm, fall-out $= \frac{FP}{N} = 1 - TNR$	True negative rate (TNR), specificity (SPC), selectivity $= \frac{TN}{N} = 1 - FPR$
	Prevalence $= \frac{P}{P + N}$	Positive predictive value (PPV), precision $= \frac{TP}{PP} = 1 - FDR$	False omission rate (FOR) $= \frac{FN}{PN} = 1 - NPV$	Positive likelihood ratio (LR+) $= \frac{TPR}{FPR}$	Negative likelihood ratio (LR−) $= \frac{FNR}{TNR}$
	Accuracy (ACC) $= \frac{TP + TN}{P + N}$	False discovery rate (FDR) $= \frac{FP}{PP} = 1 - PPV$	Negative predictive value (NPV) $= \frac{TN}{PN} = 1 - FOR$	Markedness (MK), deltaP (Δp) $= PPV + NPV - 1$	Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$
	Balanced accuracy (BA) $= \frac{TPR + TNR}{2}$	F_1 score $= \frac{2PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$	Fowlkes–Mallows index (FM) $= \sqrt{PPV \times TPR}$	Matthews correlation coefficient (MCC) $= \sqrt{TPR \times TNR \times PPV \times NPV} - \sqrt{FNR \times FPR \times FOR \times FDR}$	Threat score (TS), critical success index (CSI), Jaccard index $= \frac{TP}{TP + FN + FP}$

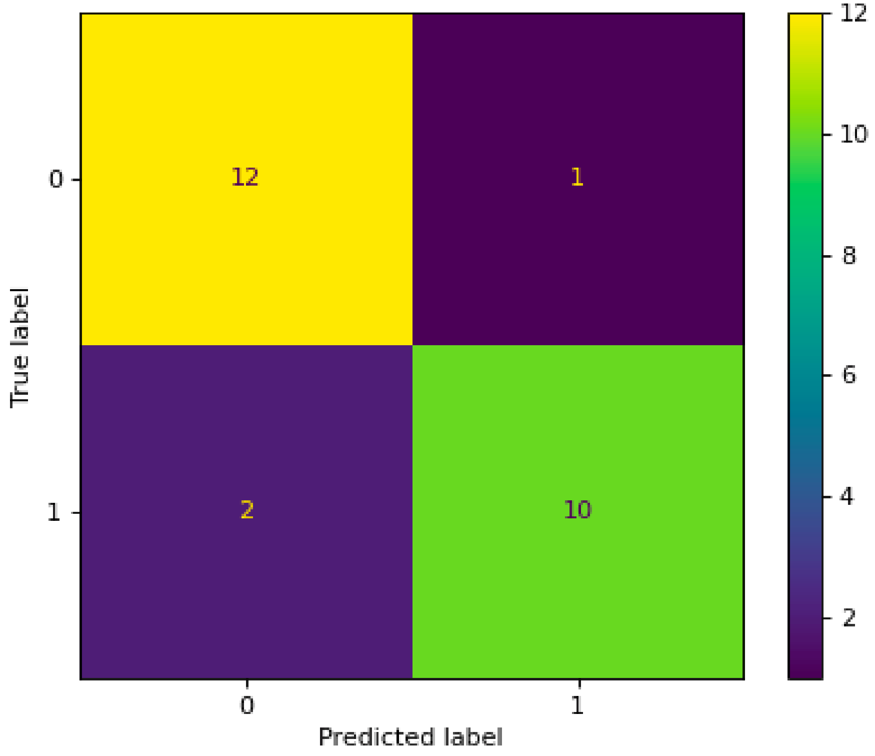
Extended Confusion Matrix

Sources: [20][21][22][23][24][25][26][27] view · talk · edit

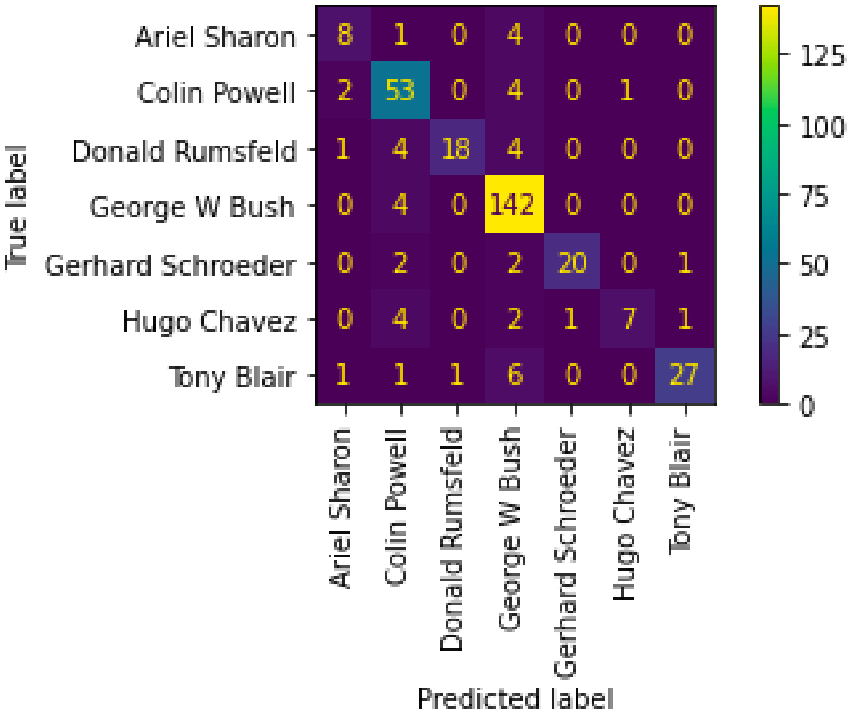
		Predicted condition			
	Total population $= P + N$	**Positive (PP)**	**Negative (PN)**	Informedness, bookmaker informedness (BM) $= TPR + TNR - 1$	Prevalence threshold (PT) $= \frac{\sqrt{TPR \times FPR} - FPR}{TPR - FPR}$
Actual condition	**Positive (P)**	**True positive (TP), hit**	**False negative (FN), type II error, miss, underestimation**	True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power $= \frac{TP}{P} = 1 - FNR$	False negative rate (FNR), miss rate $= \frac{FN}{P} = 1 - TPR$
	Negative (N)	**False positive (FP), type I error, false alarm, overestimation**	**True negative (TN), correct rejection**	False positive rate (FPR), probability of false alarm, fall-out $= \frac{FP}{N} = 1 - TNR$	True negative rate (TNR), specificity (SPC), selectivity $= \frac{TN}{N} = 1 - FPR$
	Prevalence $= \frac{P}{P + N}$	Positive predictive value (PPV), precision $= \frac{TP}{PP} = 1 - FDR$	False omission rate (FOR) $= \frac{FN}{PN} = 1 - NPV$	Positive likelihood ratio (LR+) $= \frac{TPR}{FPR}$	Negative likelihood ratio (LR−) $= \frac{FNR}{TNR}$
	Accuracy (ACC) $= \frac{TP + TN}{P + N}$	False discovery rate (FDR) $= \frac{FP}{PP} = 1 - PPV$	Negative predictive value (NPV) $= \frac{TN}{PN} = 1 - FOR$	Markedness (MK), deltaP (Δp) $= PPV + NPV - 1$	Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$
	Balanced accuracy (BA) $= \frac{TPR + TNR}{2}$	F_1 score $= \frac{2PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$	Fowlkes–Mallows index (FM) $= \sqrt{PPV \times TPR}$	Matthews correlation coefficient (MCC) $= \sqrt{TPR \times TNR \times PPV \times NPV} - \sqrt{FNR \times FPR \times FOR \times FDR}$	Threat score (TS), critical success index (CSI), Jaccard index $= \frac{TP}{TP + FN + FP}$

Confusion Matrices in `sklearn`

```
import matplotlib.pyplot as plt
from sklearn.metrics import plot_confusion_matrix

clf.fit(X_train, y_train)

plot_confusion_matrix(clf, X_test, y_test)
plt.show()
```



Confusion Matrices in `sklearn`



<u>Pros</u>

- Many derived metrics
- Easy to implement
- Summary of model mistakes is clear

<u>Cons</u>

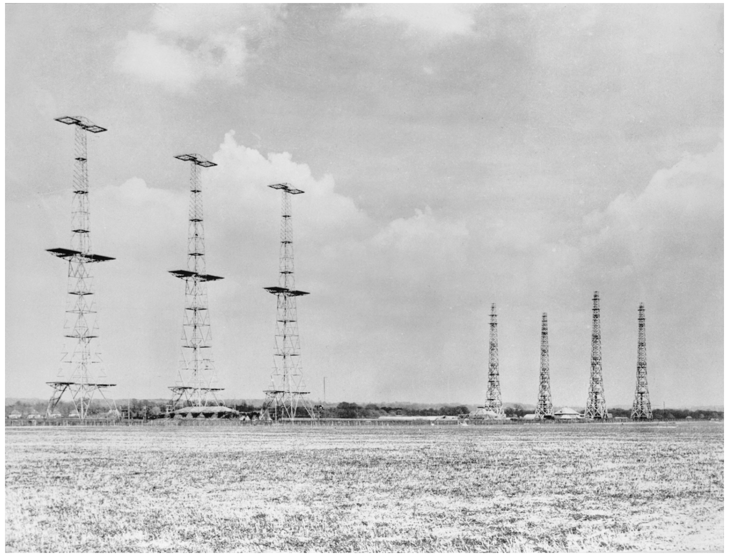
- Hard to scale
- Hard to assess probabilistic output
- Hard to view individual errors

ROC Curves

Classifiers of another age

Classifier

Data



Class 1

Class 0

As radar technology advanced during WW2, the need for a standard system to evaluate detection accuracy became apparent. ROC analysis was developed as a standard methodology to quantify a signal receiver's ability to correctly distinguish objects of interest from the background noise in the system.

Receiver Operating Characteristic (ROC)

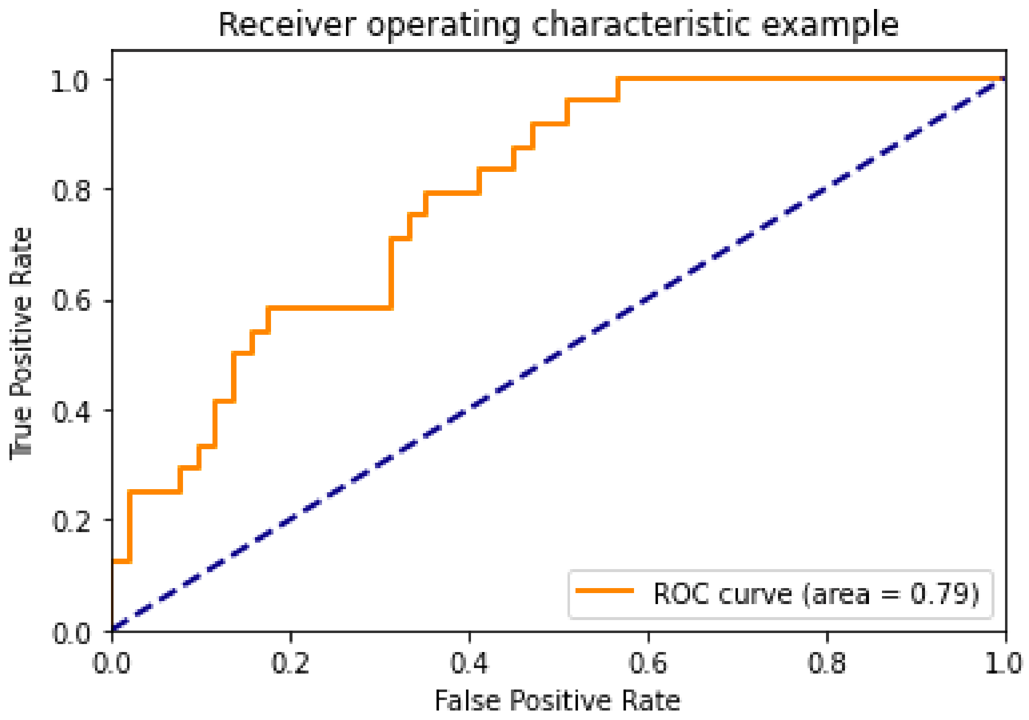
ROC analysis is another way to assess a classifier's output.

ROC analysis developed out of radar operation in the second World War, where operators were interested in detecting signal (enemy aircraft) versus noise. Thereafter, it became popular in medicine and bioinformatics.

We create an ROC curve by plotting the true positive rate (TPR) against the false positive rate (FPR) at various thresholds.

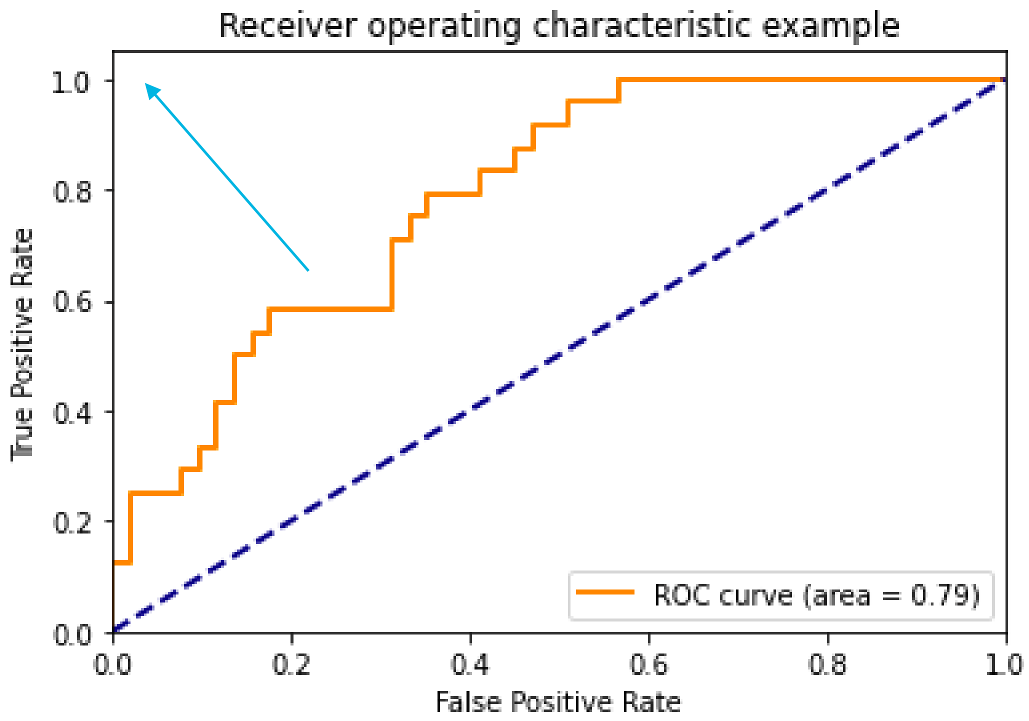
ROC Curve

```python
from sklearn.metrics import roc_curve
from sklearn.metrics import RocCurveDisplay

# Create score
y_score = clf.decision_function(X_test)

# Generate ROC curve
fpr, tpr, _ = roc_curve(y_test, y_score,
pos_label=clf.classes_[1])
# Visualize ROC curve
roc_display = RocCurveDisplay(fpr=fpr, tpr=tpr).plot()
```



ROC Curve

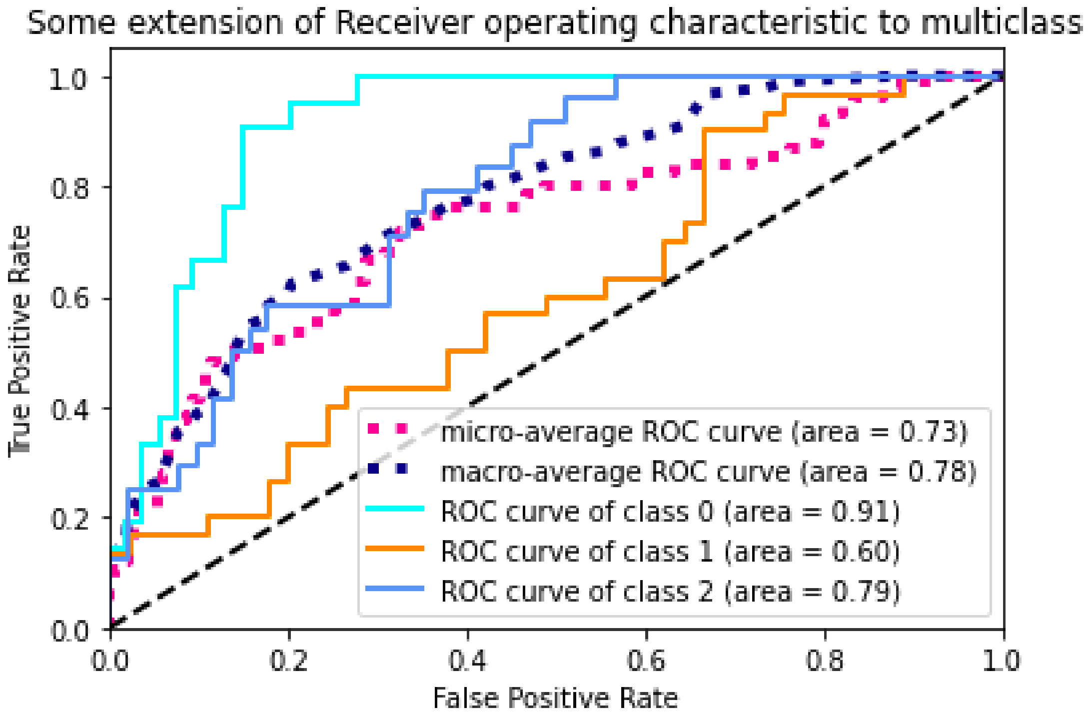
Good classifiers will exhibit ROC curves that are "*up and to the left*".



We can calculate the "*area under the curve*", or AUC, as a measurement of classifier quality.

ROC Curve (multiclass)

In multiclass scenarios, we have to binarize the labels and plot each separately.



Micro-average

Aggregate contributions of all classes to calculate the metric. Useful if there is class imbalance.

Macro-average

Compute the metric for each class separately, then take average (treats all classes equally).

Visual Analytics Systems

Model Understanding

Squares (2016)

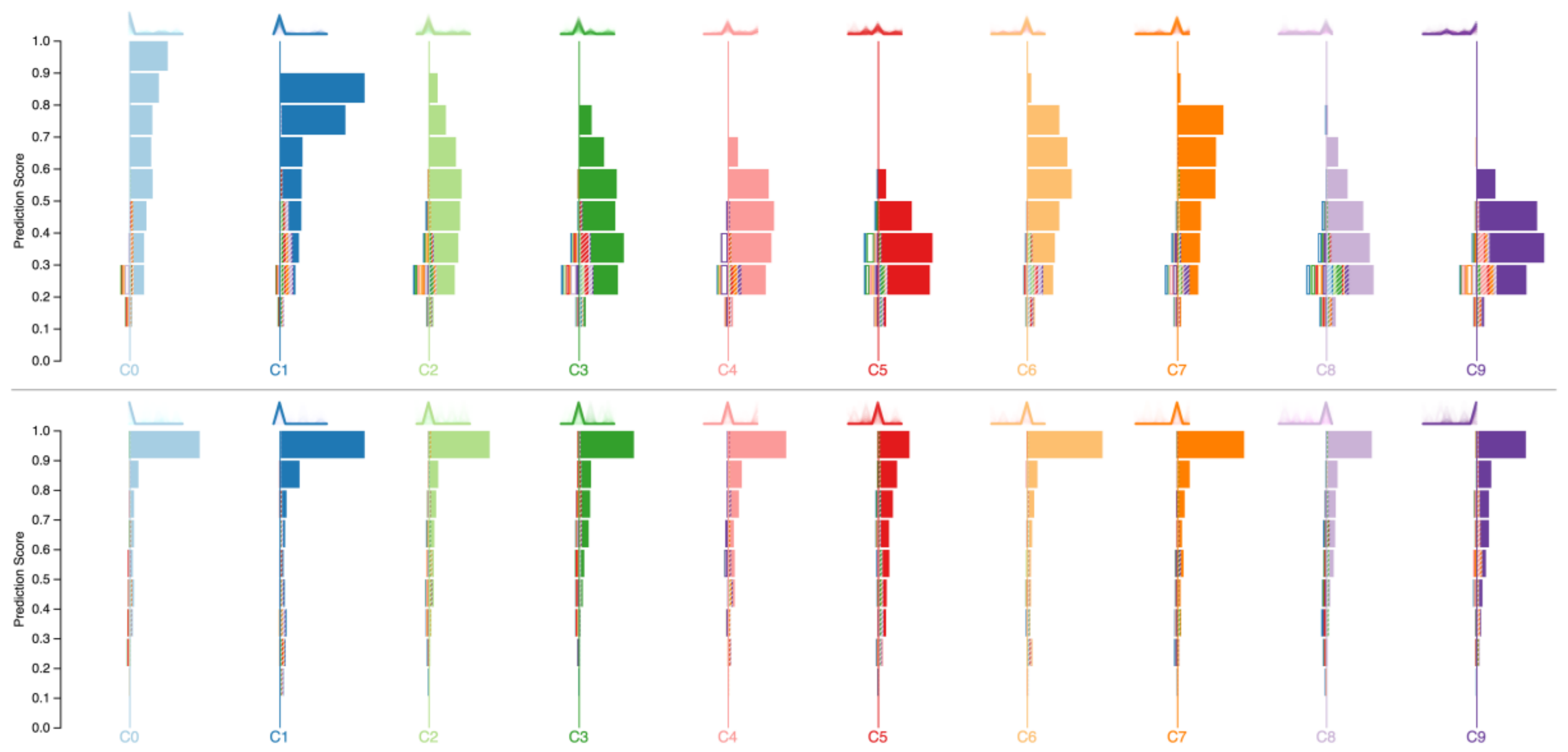


Fig. 1. Squares displaying the performance of two digit recognition classifiers trained on the MNIST handwritten digits dataset [24]. These classifiers yield the same accuracy of 0.87 (top: random forest, bottom: SVM), but show vastly different score distributions.

<u>Challenges</u>

Count-Based Metrics
Score-Based Metrics
Instance Level

Ren, D., Amershi, S., Lee, B., Suh, J., & Williams, J. D. (2016). Squares: Supporting interactive performance analysis for multiclass classifiers. *IEEE transactions on visualization and computer graphics*, *23*(1), 61-70.

Squares

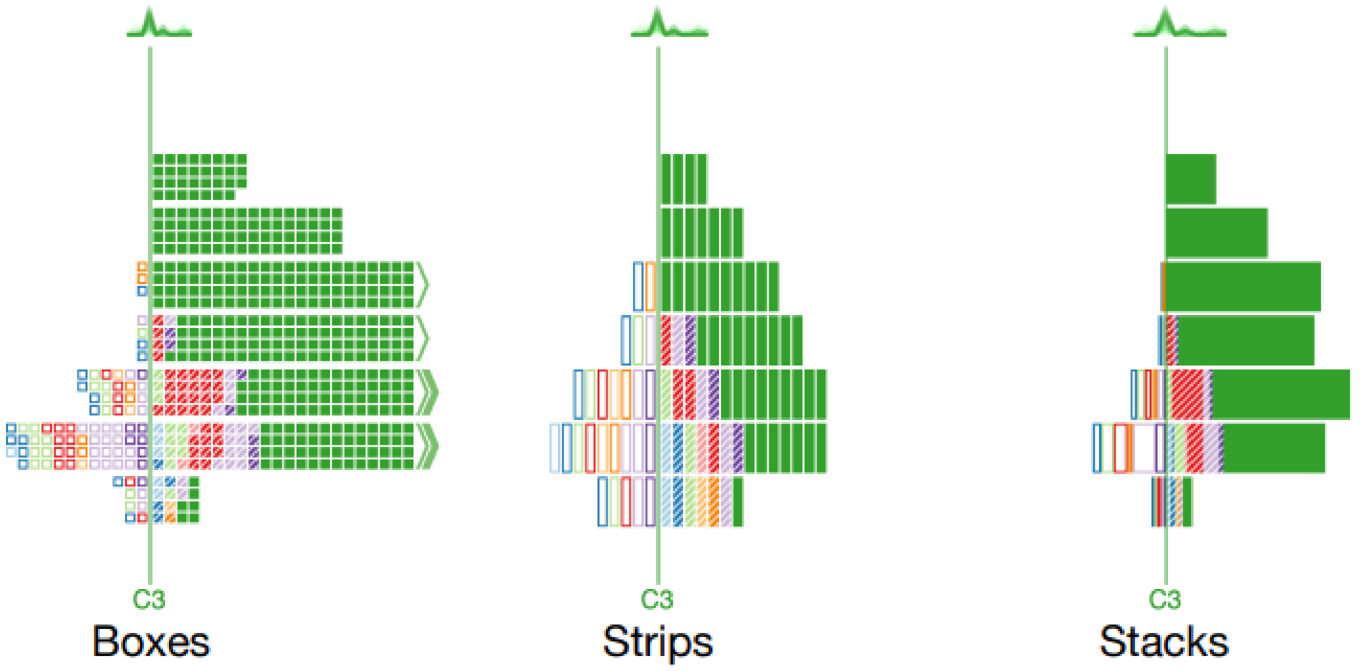


Fig. 5. The strips and stacks view group boxes together when there are a large number of classes or instances. Users can toggle between boxes (left), strips (center), and stacks (right) at the class level.

Ren, D., Amershi, S., Lee, B., Suh, J., & Williams, J. D. (2016). Squares: Supporting interactive performance analysis for multiclass classifiers. *IEEE transactions on visualization and computer graphics*, *23*(1), 61-70.

Squares

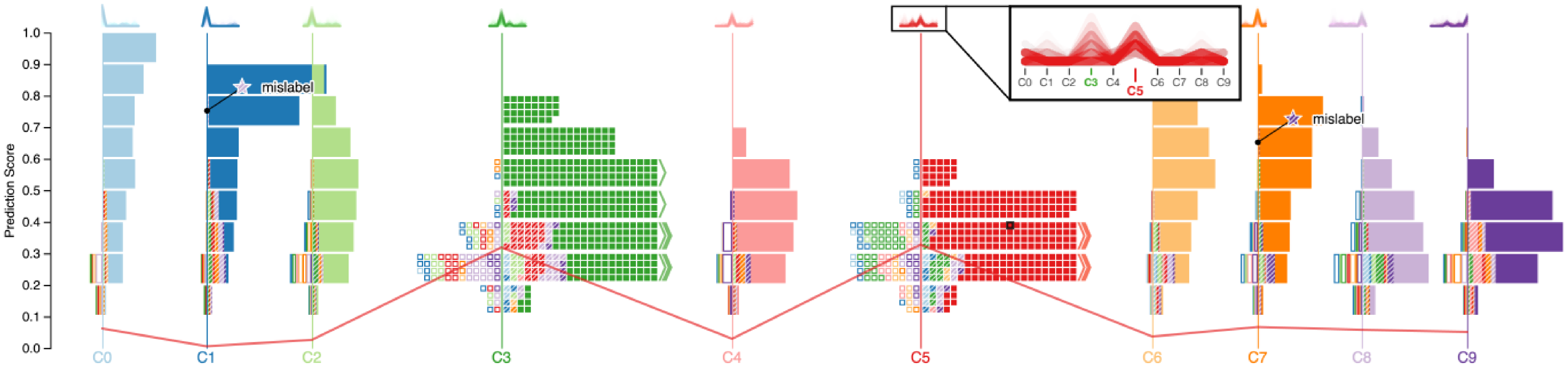


Fig. 4. Squares displaying the performance of a digit recognition classifier trained on the MNIST handwritten digits dataset [24]. All classes are represented with stacks except C3 and C5 which are expanded to boxes for more details. The solid red boxes in C5's column represents instances correctly predicted as C5 while the green-stripped boxes in that column represent instances labeled as C3 but incorrectly predicted as C5.

Ren, D., Amershi, S., Lee, B., Suh, J., & Williams, J. D. (2016). Squares: Supporting interactive performance analysis for multiclass classifiers. *IEEE transactions on visualization and computer graphics*, *23*(1), 61-70.

Squares

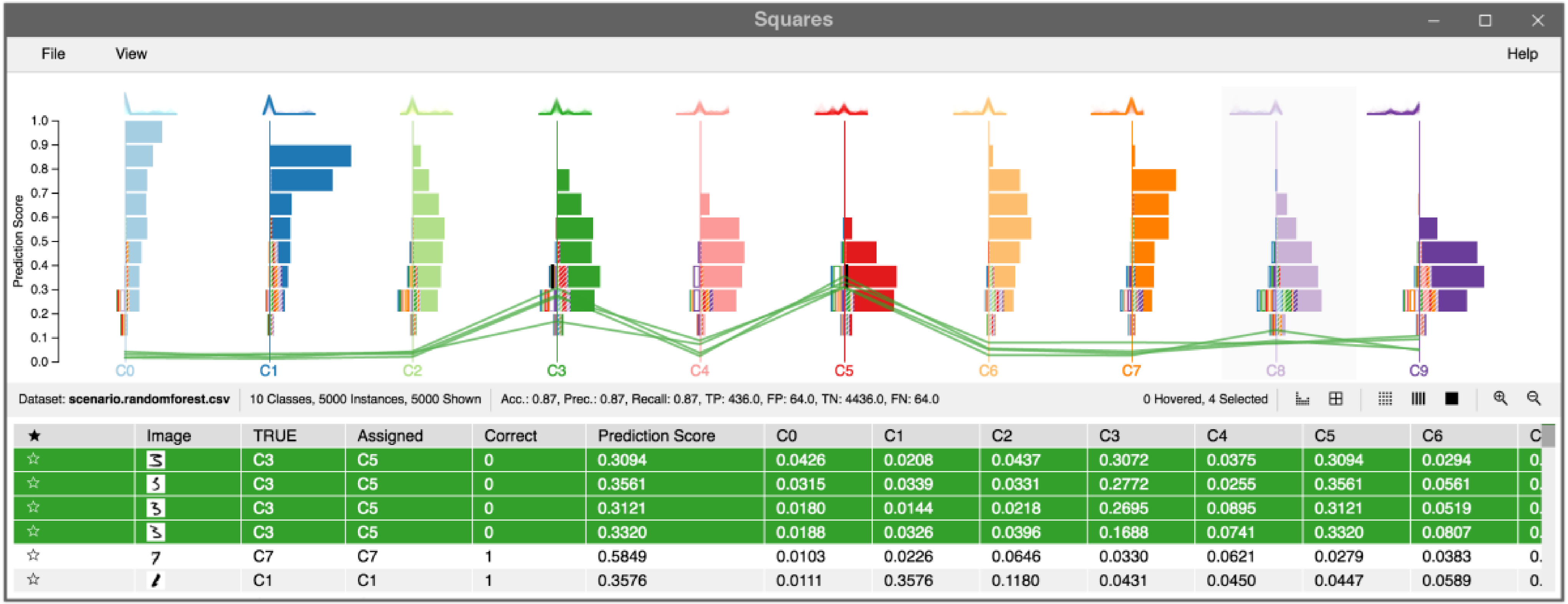


Fig. 6. Bi-directional coupling between the visualization and table allows users to view instance properties in the table by selecting boxes, strips, or stacks from the visualization, or locate interesting instances found in the table in the visualization.

Ren, D., Amershi, S., Lee, B., Suh, J., & Williams, J. D. (2016). Squares: Supporting interactive performance analysis for multiclass classifiers. *IEEE transactions on visualization and computer graphics*, *23*(1), 61-70.

UIC COMPUTER SCIENCE

Evaluating Squares

T1: Select the classifier with the larger number of errors (this required displaying two visualizations side-by-side).

T2: Select one of the two classes with the most errors.

T3: Select an error with a score of .9 or above in the wrong class.

T4: Select the classifier with the worst distribution (this required displaying two visualizations side-by-side).

T5: Select one of the two classes with the worst distribution.

T6: Select the two classes most confused with each other.

Evaluating Squares

Compared against an interactive confusion matrix.

We ran the comparison part of the study as a 2 (Visualization: Squares vs. ConfusionMatrix) x 2 (Class-Size: Small vs. Large) x 3 (Task: T1, T2, and T3) within-subjects design.

Small class size classifiers had 5 classes and Large ones had 15.

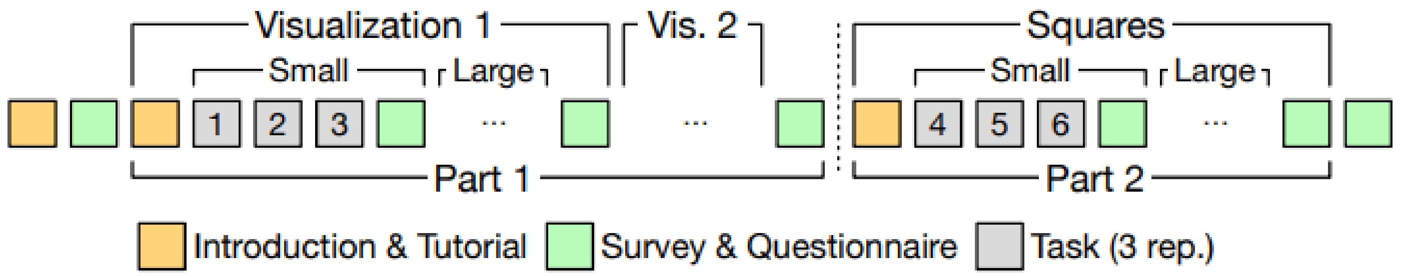
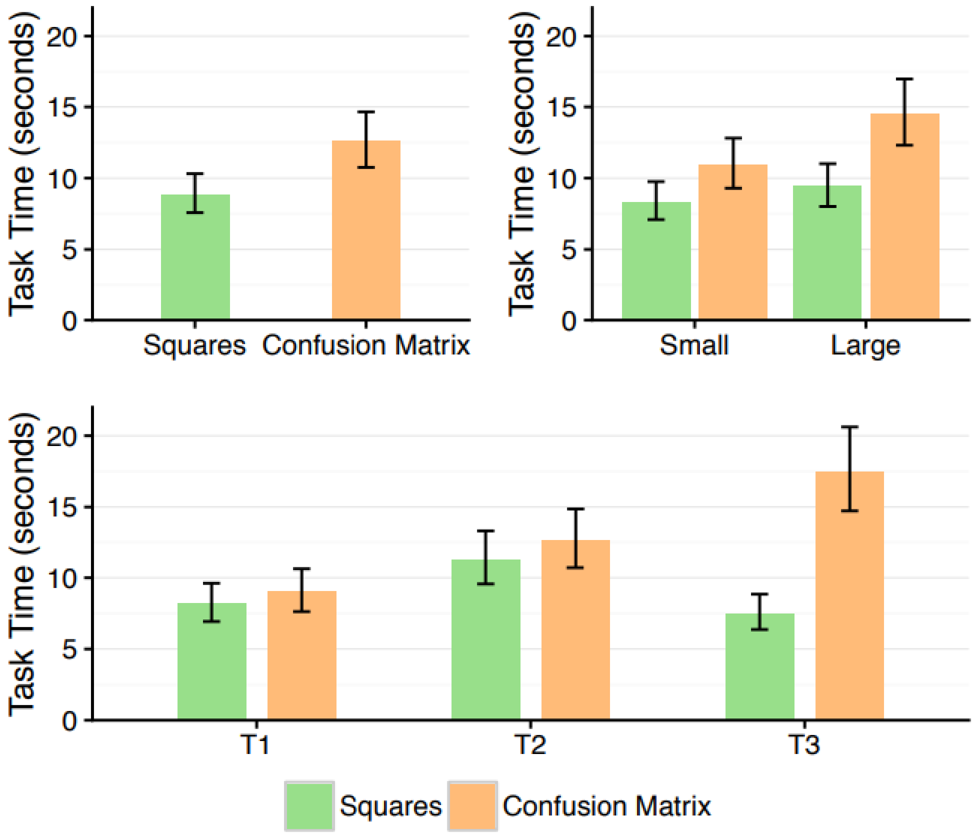


Fig. 8. The study consists of two parts. In the first part, we compared Squares to an interactive confusion matrix. In the second part, we evaluate only Squares for estimating score-based metrics.

Evaluating Squares



Evaluating Squares

Visualization	T1		T2		T3	
	5	15	5	15	5	15
Helpfulness						
Squares	4.3	4.3	4.7	4.1	4.1	4.7
Confusion Matrix	3.7	3.7	3.8	3.5	3.5	3.1
Preference						
Squares	20	17	23	20	23	23
Confusion Matrix	5	7	2	5	2	2

Table 1. Subjective responses: (top) means of participant responses on how helpful (5=Very helpful) the visualization was by task for each class size; (bottom) the numbers of participants who preferred the visualization by task for each class size.

Alsallakh et. al. (2014)

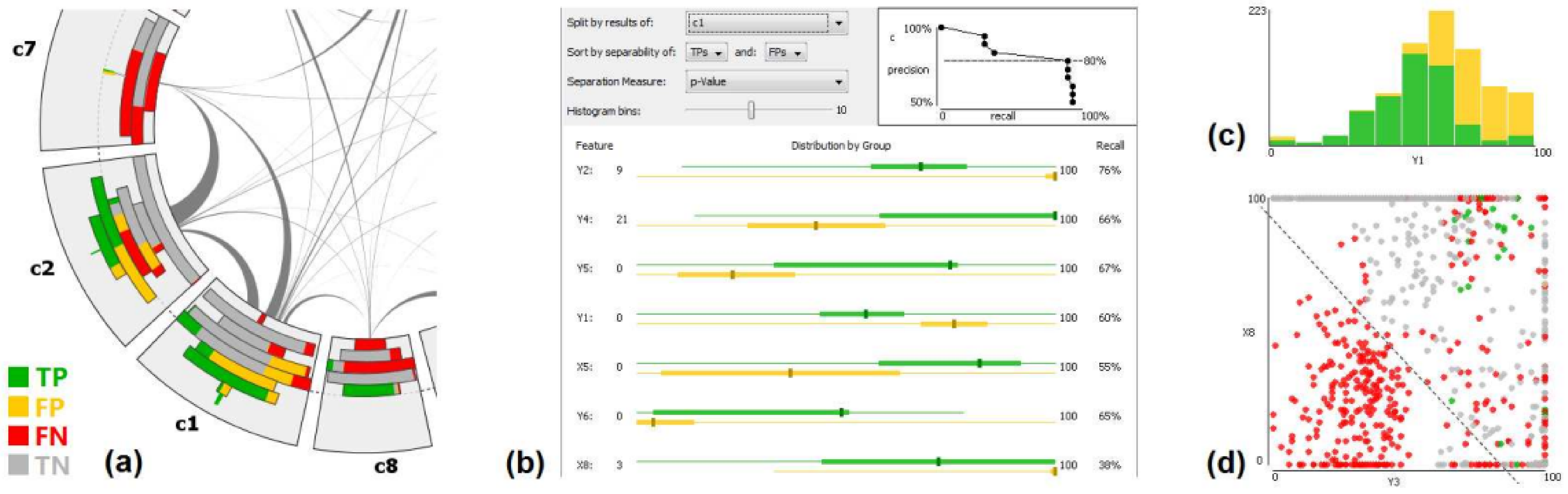


Fig. 1. Our visual analysis tools: (a) the *confusion wheel* shows sample-class probabilities as histograms colored by classification results, (b) the *feature analysis view* depicts feature distributions among selected samples, separated by their results, and ranked by a separation measure, (c, d) histograms and scatterplots reveal the separability of selected true and false classified samples by one or two features.

Alsallakh, B., Hanbury, A., Hauser, H., Miksch, S., & Rauber, A. (2014). Visual methods for analyzing probabilistic classification data. *IEEE transactions on visualization and computer graphics*, *20*(12), 1703-1712.

Alsallakh et. al. (2014)

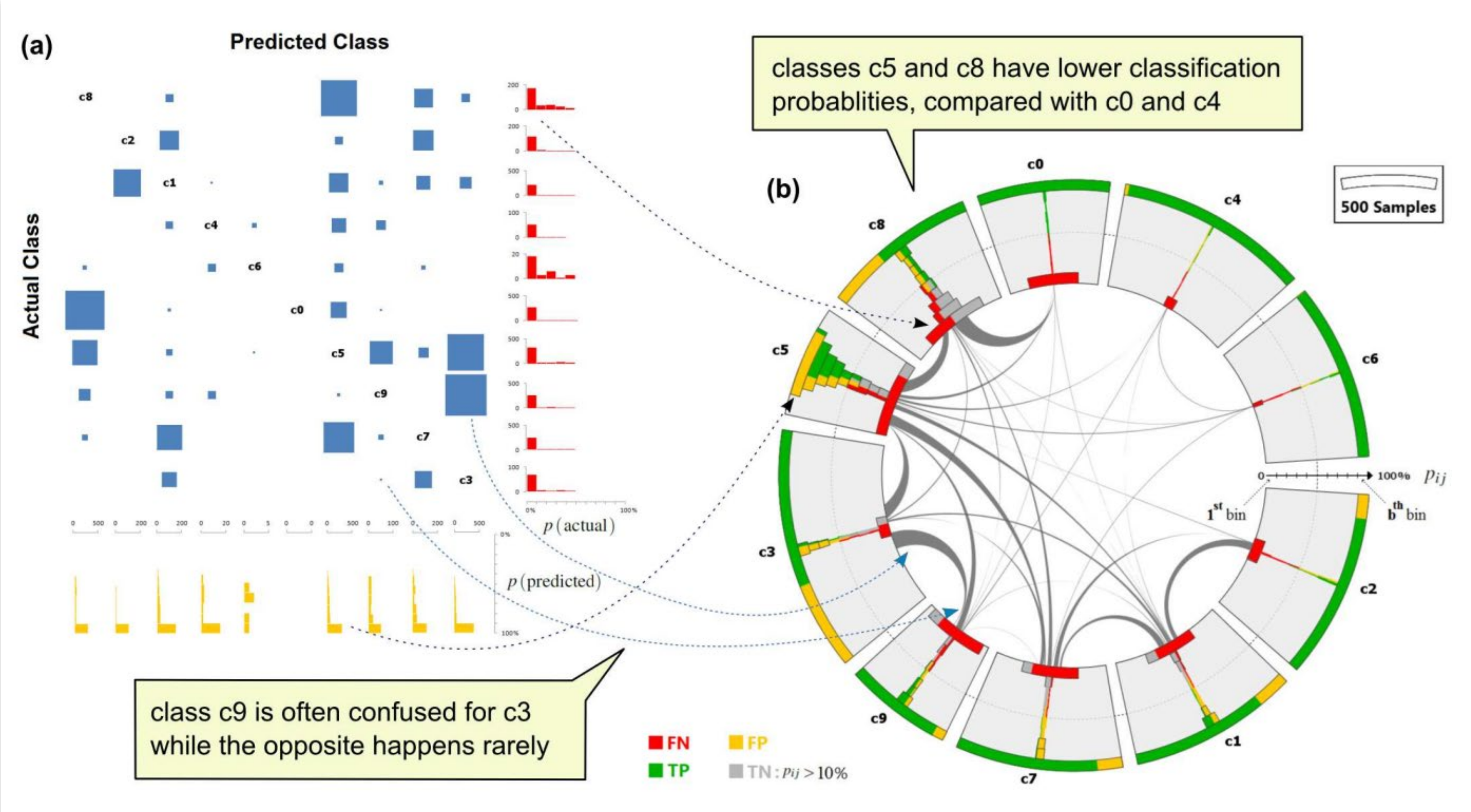


Fig. 3. Visualizing classification results of 10,992 handwritten digits [5] (a) using a confusion matrix augmented with histograms of sample probabilities in the respective rows and columns, (b) using the confusion wheel: Sectors represent digits with chords showing classification confusion between them. Histograms represent the probabilities of the samples in each class according to the color legend.

Alsallakh, B., Hanbury, A., Hauser, H., Miksch, S., & Rauber, A. (2014). Visual methods for analyzing probabilistic classification data. *IEEE transactions on visualization and computer graphics*, *20*(12), 1703-1712.

Beauxis-Aussalet and Hardman (2014)

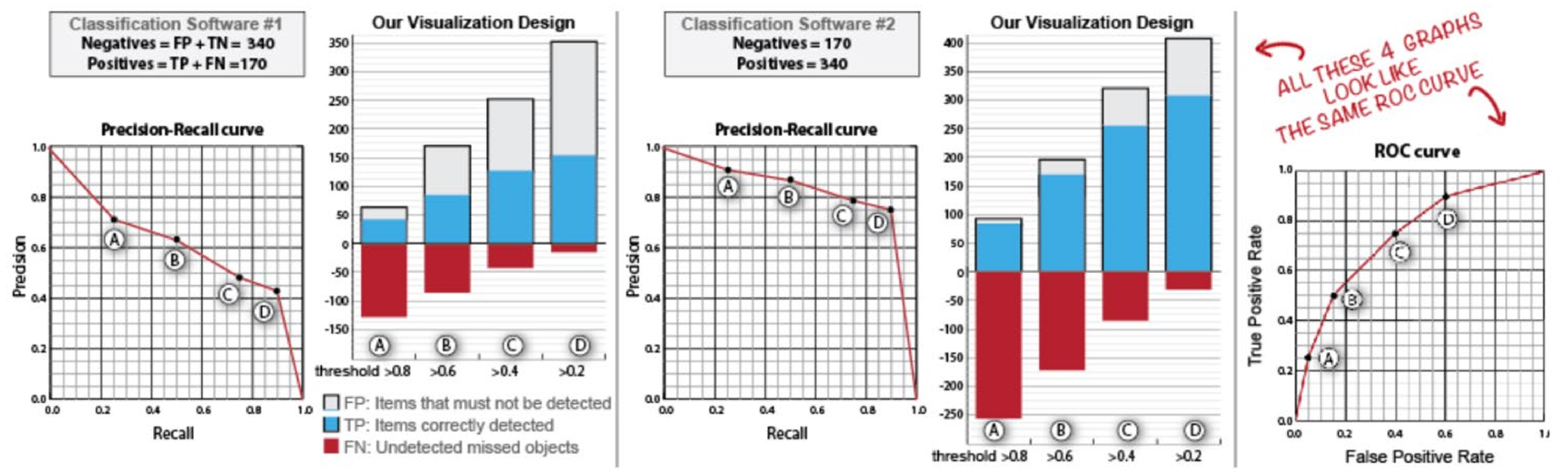


Figure 2: Alternative visualizations of confusion matrices: our design, and equivalent ROC and Precision/Recall curves.

Beauxis-Aussalet, E., & Hardman, L. (2014). Visualization of confusion matrix for non-expert users. In *IEEE Conference on Visual Analytics Science and Technology (VAST)-Poster Proceedings*.

Beauxis-Aussalet and Hardman (2014)

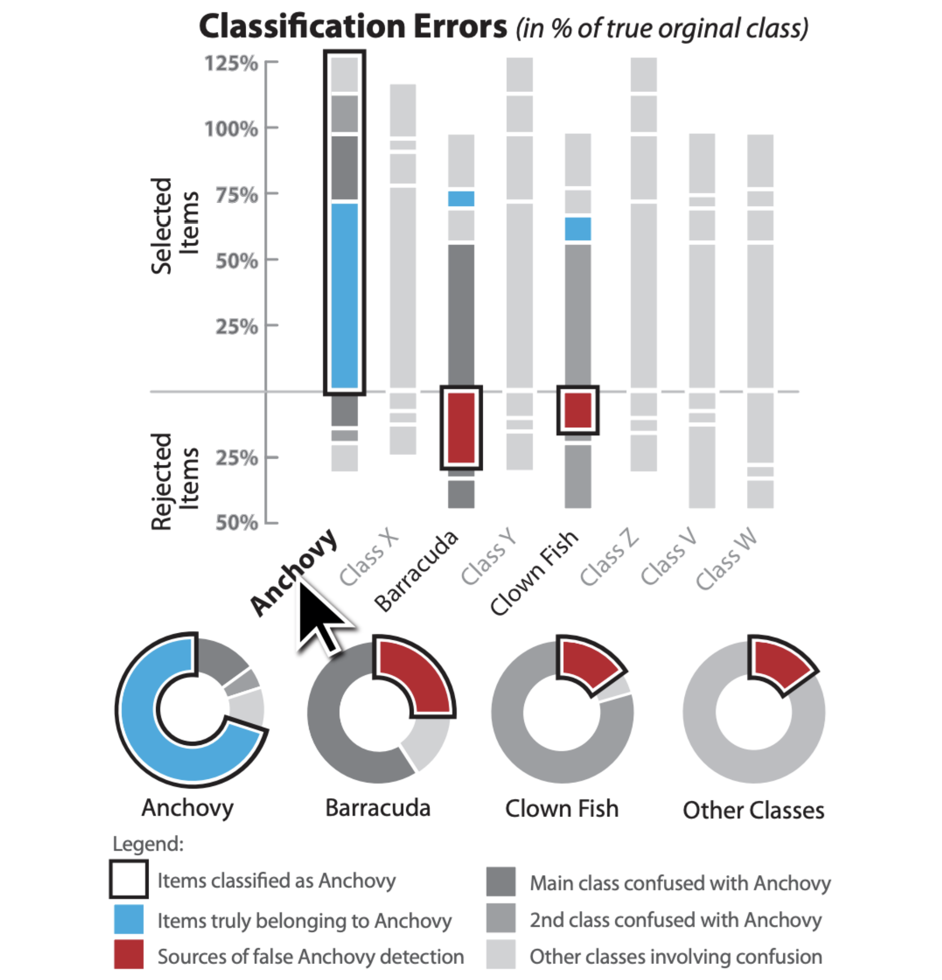


Figure 3: Visualization for analyzing of inter-classes confusions.

Calibration

What if accuracy doesn’t matter?

What if we care about *probabilities* instead of labels?

Scenario: Weather Prediction

Consider a weather channel that sets a chance of rain to its viewers, daily.

The average chance of rain is 30%.

Since the channel doesn't use machine learning, they simply tell their viewers there is an 30% chance of rain every day.

The weather channel claims it is accurate, but all that the viewers have learned is that there is, on average, a 30% chance of rain.

What is calibration?

Typically, we turn to accuracy to help us evaluate models. This makes sense when the model output we care about is the predicted class label.

Oftentimes, however, we are interested in the *probabilistic* output. For example, applications that rely on probabilistic quantities, like betting, require our models not necessarily to be accurate but return *probabilities* that reflect reality.

Modern neural networks can often produce "bad" probabilistic outputs

How do we measure how well a model's probabilistic output aligns with reality?

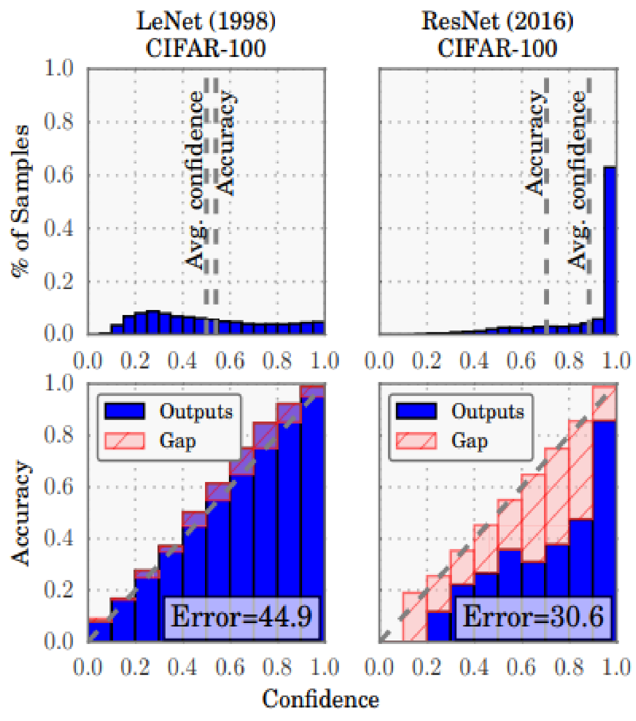
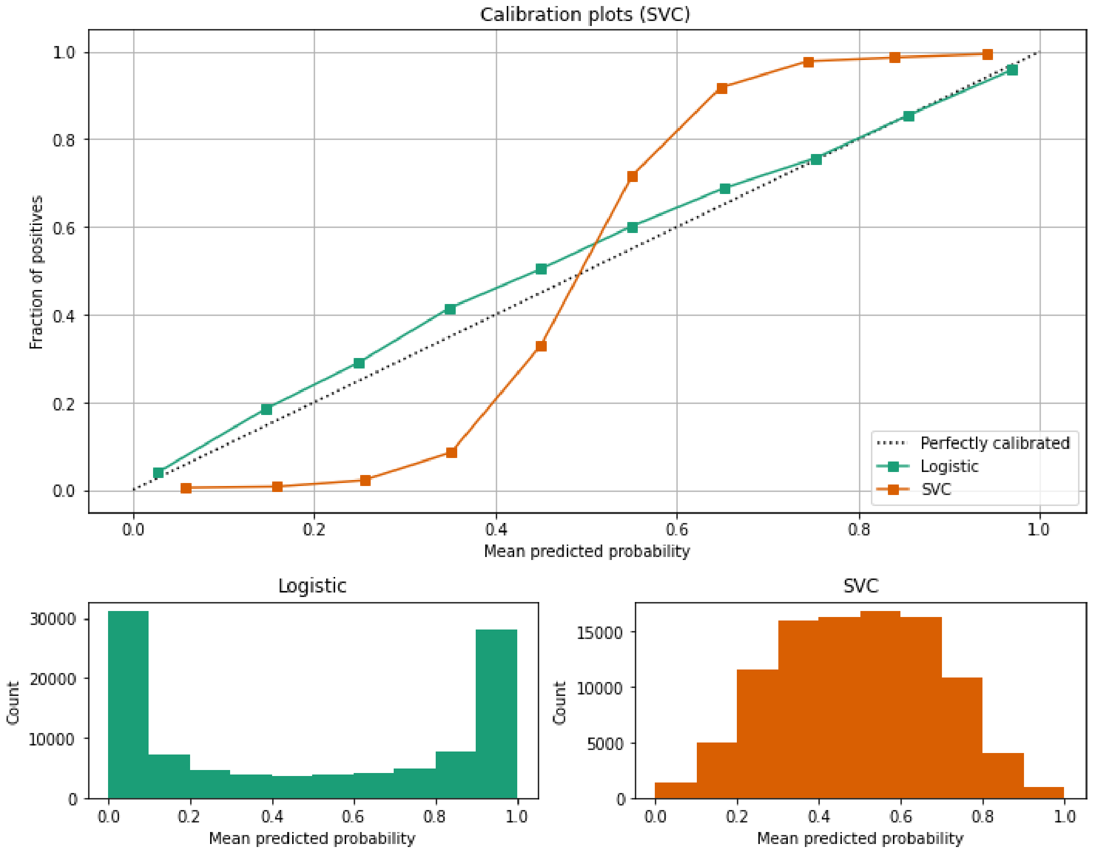


Figure 1. Confidence histograms (top) and reliability diagrams (bottom) for a 5-layer LeNet (left) and a 110-layer ResNet (right) on CIFAR-100. Refer to the text below for detailed illustration.

Image taken from Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017, July). On calibration of modern neural networks. In *International Conference on Machine Learning*. PMLR.

Basic Calibration Plots (Reliability Diagram)



Expected Calibration Error (ECE)

$$ECE = \sum_{b=1}^{B} \frac{n^b}{N} |acc(b) - conf(b)|$$

Expected Calibration Error (ECE)

Total number of bins

Samples in bin b

$$ECE = \sum_{b=1}^{\boxed{B}} \frac{\boxed{n^b}}{\boxed{N}} \left| \boxed{acc(b)} - \boxed{conf(b)} \right|$$

Total samples

Accuracy in bin b

Average predicted probability in bin b

Maximum Calibration Error (MCE)

$$MCE = \max_{m \in \{1,2,\ldots,|B|\}} |acc(B_m) - conf(B_m)|$$

Calibration in `sklearn`

```
from sklearn.calibration import calibration_curve

y_true = np.array([0, 0, 0, 0, 1, 1, 1, 1, 1])
y_pred = np.array([0.1, 0.2, 0.3, 0.4, 0.65, 0.7, 0.8, 0.9,  1.])
prob_true, prob_pred = calibration_curve(y_true, y_pred, n_bins=3)

>>> prob_true
array([0. , 0.5, 1. ])

>>> prob_pred
array([0.2  , 0.525, 0.85 ])
```

Can also pass a parameter strategy as ‘uniform’ or ‘quantile’.
Check out the official sklearn documentation.

Bin 1 Example

- Three observations, predicted 0.1, 0.2, and 0.3.
- All three observations are truly of class “0”.
- Assume a decision boundary of 0.5.
- Thus:

$$acc(b_1) = 1$$

$$conf(b_1) = \frac{0.9 + 0.8 + 0.7}{3}$$

Visualization Calibration for Multi-Class Problems

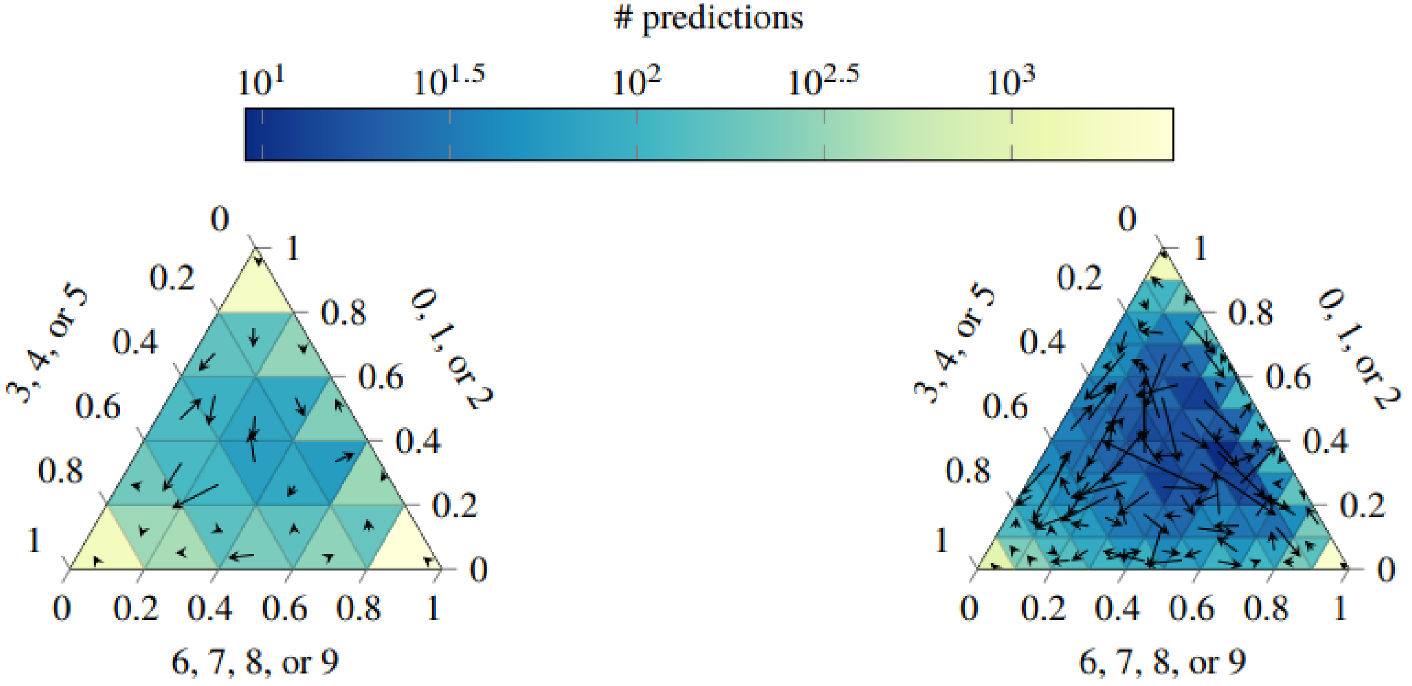
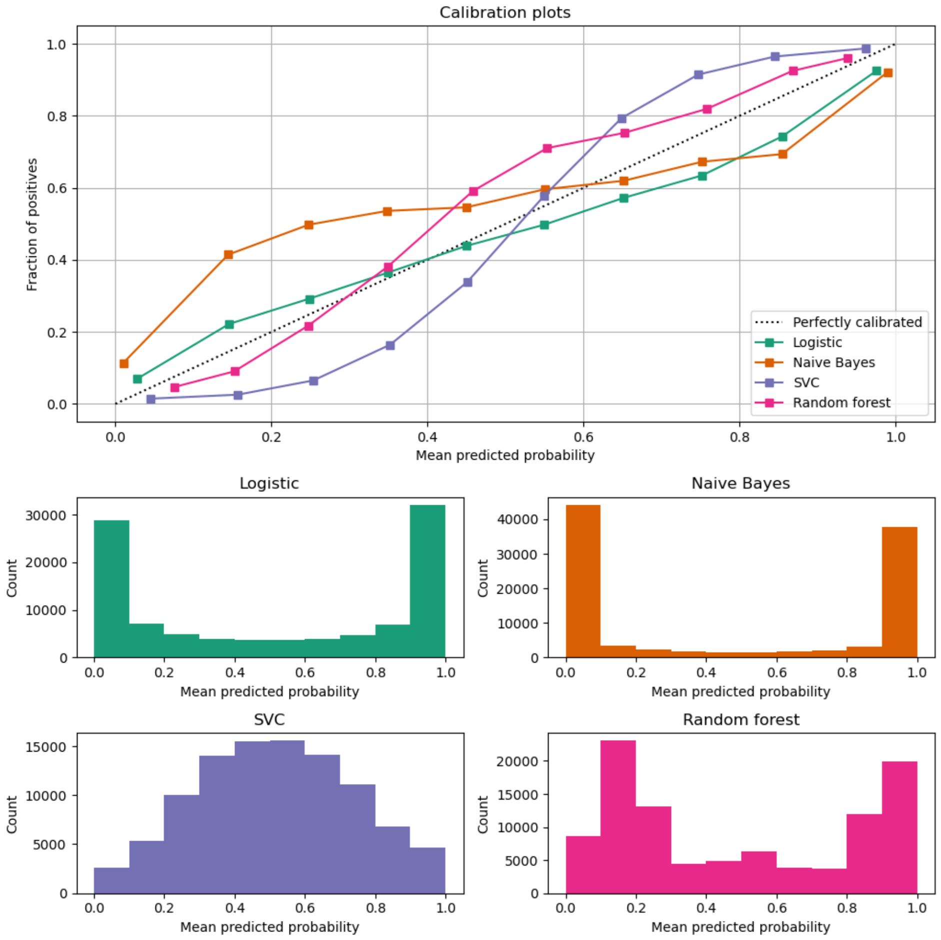


Figure 1: Two-dimensional reliability diagrams for LeNet on the CIFAR-10 test set with 25 and 100 bins of equal size. The predictions are grouped into three groups $\{0,1,2\}$, $\{3,4,5\}$, and $\{6,7,8,9\}$ of the original classes. Arrows represent the deviation of the estimated calibration function value (arrow head) from the group prediction average (arrow tail) in a bin. The empirical distribution of predictions is visualized by color-coding the bins.

Image taken from Vaicenavicius, Juozas, et al. "Evaluating model calibration in classification." *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019.

What to do if your classifier is uncalibrated?



Platt Scaling

True label

Classifier output

$$p(\boxed{y_i} = 1|\boxed{f_i}) = \frac{1}{1 + \exp(\boxed{A} f_i + \boxed{B})}$$

Determined via MLE

Platt Scaling

Assumes the calibration curve can be corrected by applying a sigmoid to the raw predictions.

Works best if the calibration error is symmetrical (classifier output for each binary class is normally distributed with the same variance)

This can be a problem for highly imbalanced classification problems, where outputs do not have equal variance.

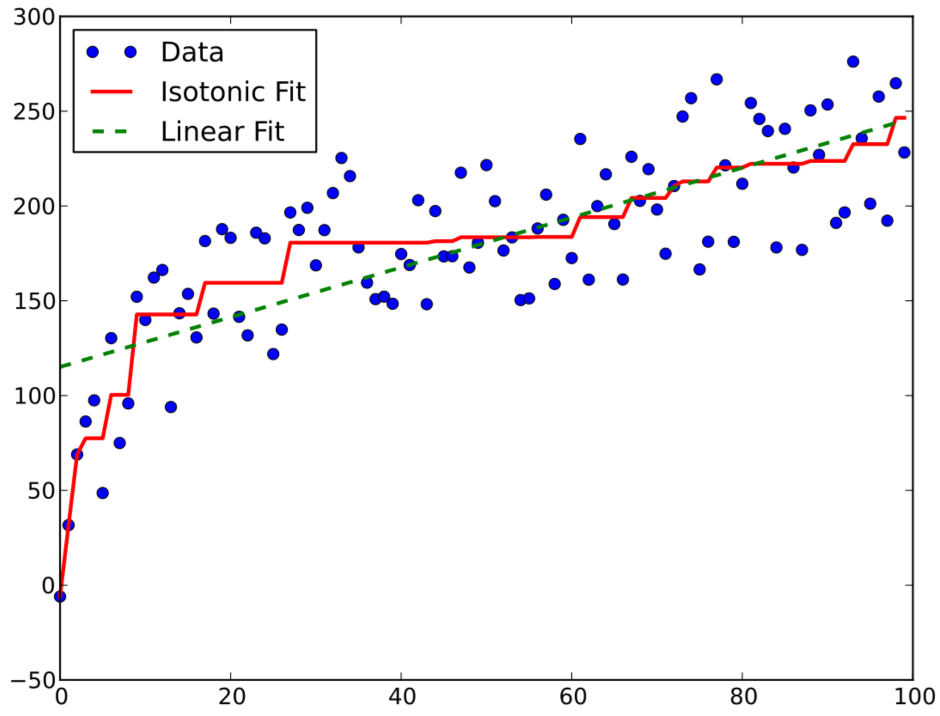
In general this method is most effective when the un-calibrated model is under-confident and has similar calibration errors for both high and low outputs.

Isotonic Regression

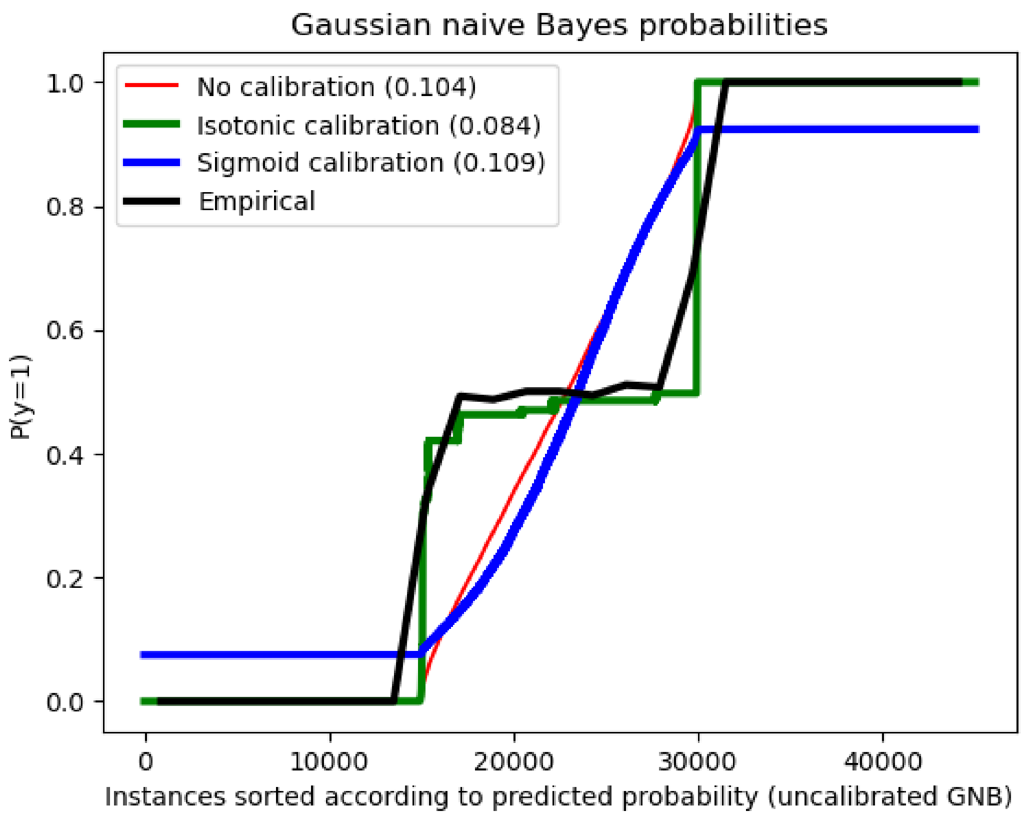
Fits a non-parametric isotonic regressor, which outputs a step-wise non-decreasing function.

Isotonic regression is more general when compared to Platt scaling, as the only restriction is that the mapping function is monotonically increasing.

Thus, isotonic regression is more powerful as it can correct any monotonic distortion of the un-calibrated model. However, it is more prone to overfitting, especially on small datasets.



Isotonic vs. Platt Scaling



Proper Scoring Rules

Proper scoring rules are calculated at the observation level, where as ECE is binned.

Brier Score

$$\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

Log Loss

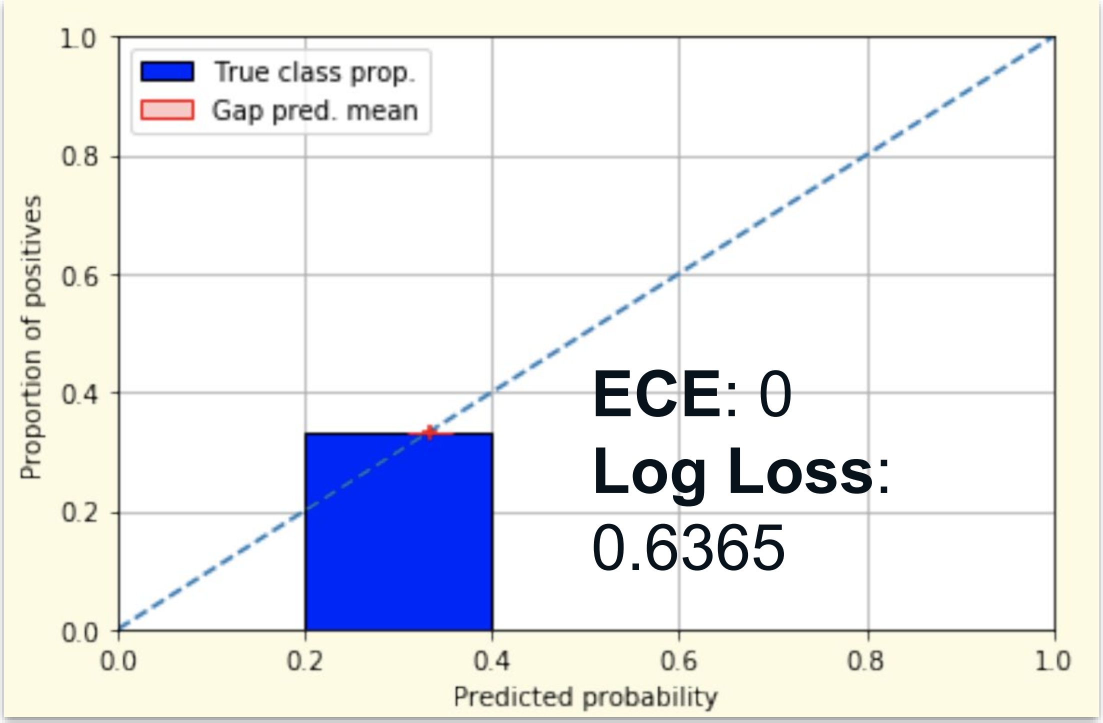
$$-\frac{1}{N}\sum_{i=1}^{N}(y_i \log(\hat{y}_i)+(1-y_i)(\log 1 - \hat{y}_i))$$

```
from sklearn.metrics import brier_score_loss, log_loss
```

Evaluation Trade-Off

$$\mathbf{Q} = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \\ \dots & \dots \\ \frac{2}{3} & \frac{1}{3} \end{bmatrix}$$

$$\mathbf{y} = [1, 1, 1, 0, 0, 0, 0, 0, 0]$$



ECE: 0
Log Loss: 0.6365

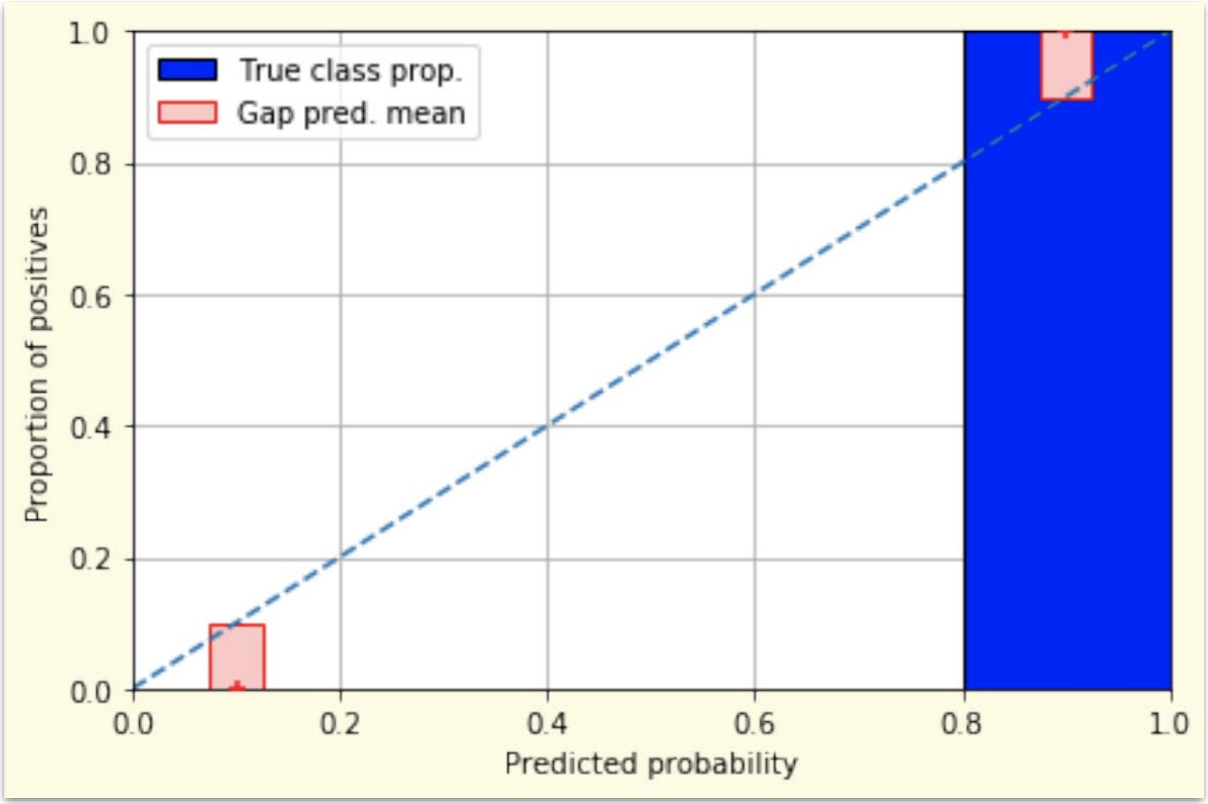
Image taken from ECML/PKDD 2020 Tutorial on Evaluation Metrics and Proper Scoring Rules

UIC COMPUTER SCIENCE

Evaluation Trade-Off

Now, change all of the “1” class to have predicted probabilities of 0.9. Then, we see

Accuracy: 1 (-)
ECE: 0.1 (increased)
Log Loss: 0.1054 (decreased)



Q: Why did log loss decrease?

Image taken from ECML/PKDD 2020 Tutorial on Evaluation Metrics and Proper Scoring Rules

Evaluation Trade-Off

Now, change all of the “1” class to have predicted probabilities of 0.9. Then, we see

Accuracy: 1 (-)
ECE: 0.1 (increased)
Log Loss: 0.1054 (decreased)

Q: Why did log loss decrease?
A: Proper scoring rules can be decomposed into terms with different interpretations (Kull and Flach, 2015).

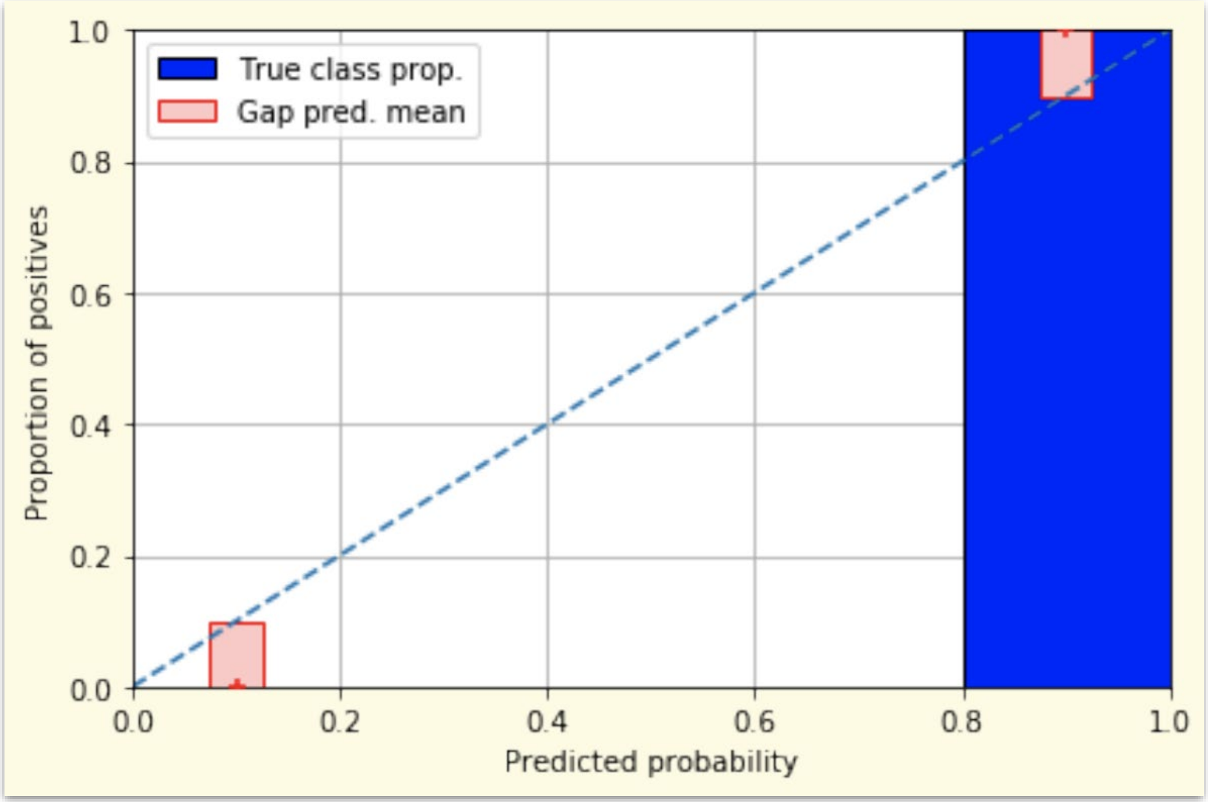


Image taken from ECML/PKDD 2020 Tutorial on Evaluation Metrics and Proper Scoring Rules

Calibration Takeaways

(1) Reliability diagrams are a standard way to visualize calibration.

(2) ECE is a summary of what reliability diagrams show.

(3) Proper scoring rules (Log loss, Brier score) measure different aspects of probability correctness.

(4) However, proper scoring rules cannot tell us *where* a model is miscalibrated.

Suggested Calibration Literature

Niculescu-Mizil, A., & Caruana, R. (2005, August). Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning* (pp. 625-632).

Nixon, J., Dusenberry, M. W., Zhang, L., Jerfel, G., & Tran, D. (2019, June). Measuring Calibration in Deep Learning. In *CVPR Workshops* (Vol. 2, No. 7).

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017, July). On calibration of modern neural networks. In *International Conference on Machine Learning* (pp. 1321-1330). PMLR.

Vaicenavicius, J., Widmann, D., Andersson, C., Lindsten, F., Roll, J., & Schön, T. (2019, April). Evaluating model calibration in classification. In *The 22nd International Conference on Artificial Intelligence and Statistics* (pp. 3459-3467). PMLR.

Kull, M., & Flach, P. (2015, September). Novel decompositions of proper scoring rules for classification: Score adjustment as precursor to calibration. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 68-85). Springer, Cham.

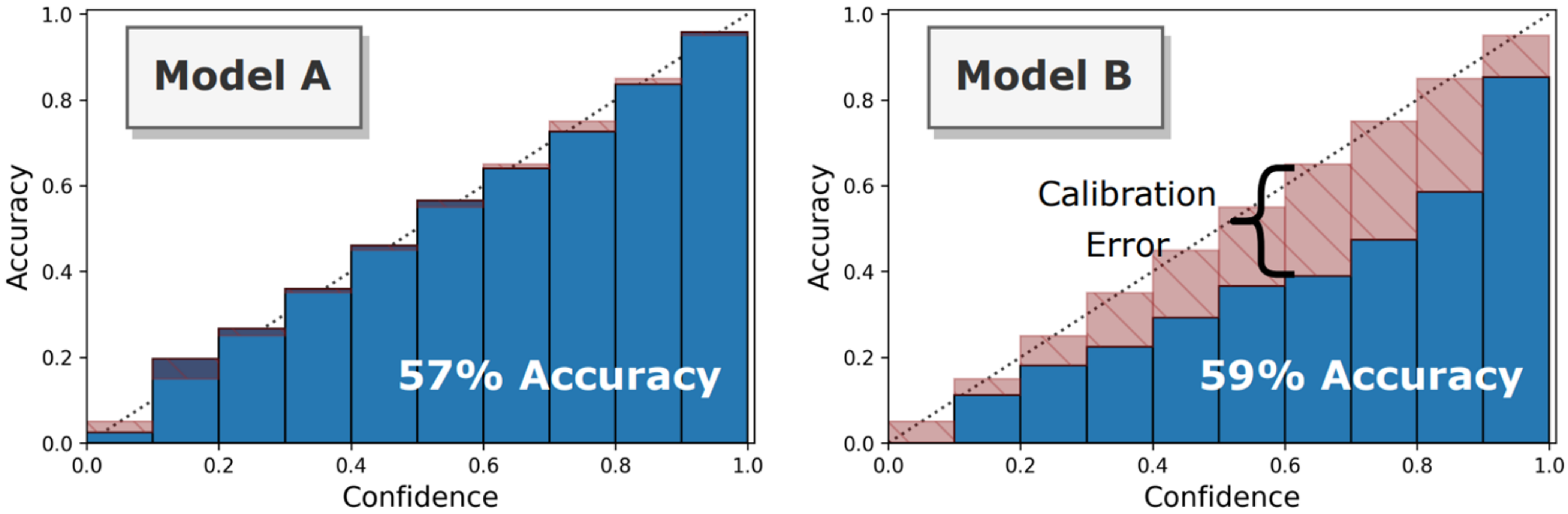
ECML/PKDD 2020 Tutorial: Evaluation metrics and proper scoring rules.

Google Colab notebook for calibration curves.

Visual Analytics Systems

Calibration

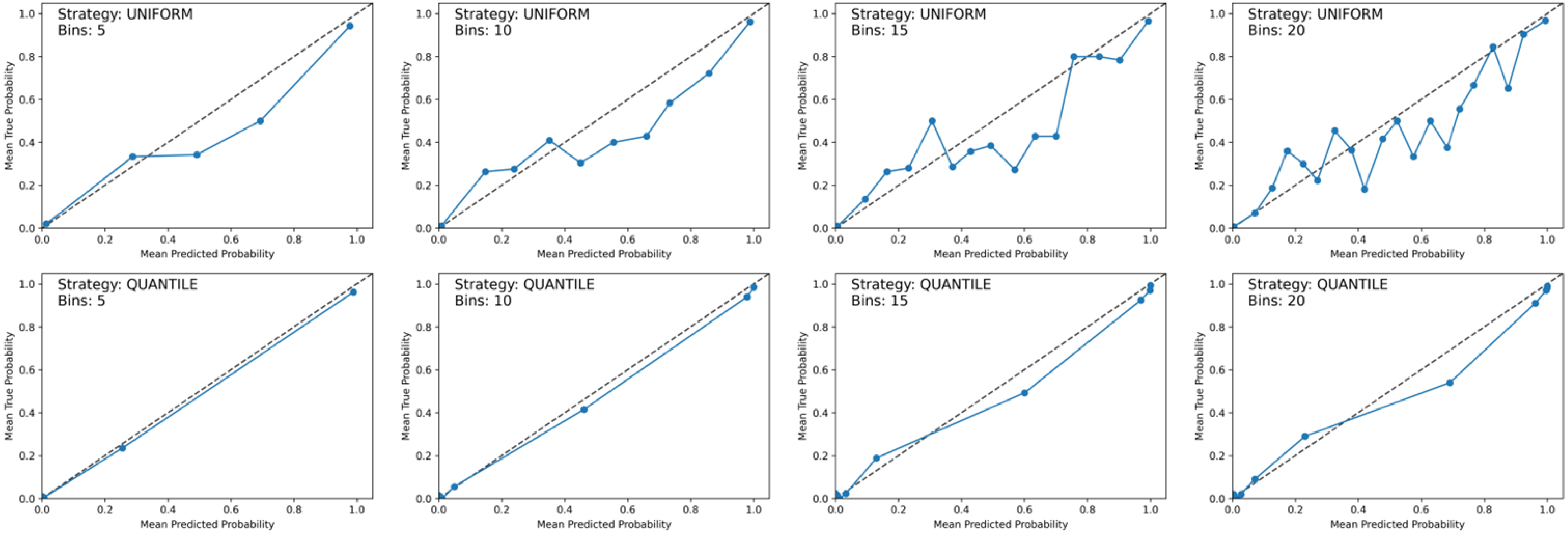
Xenopoulos et. al. (2022)



Parameters Matter!

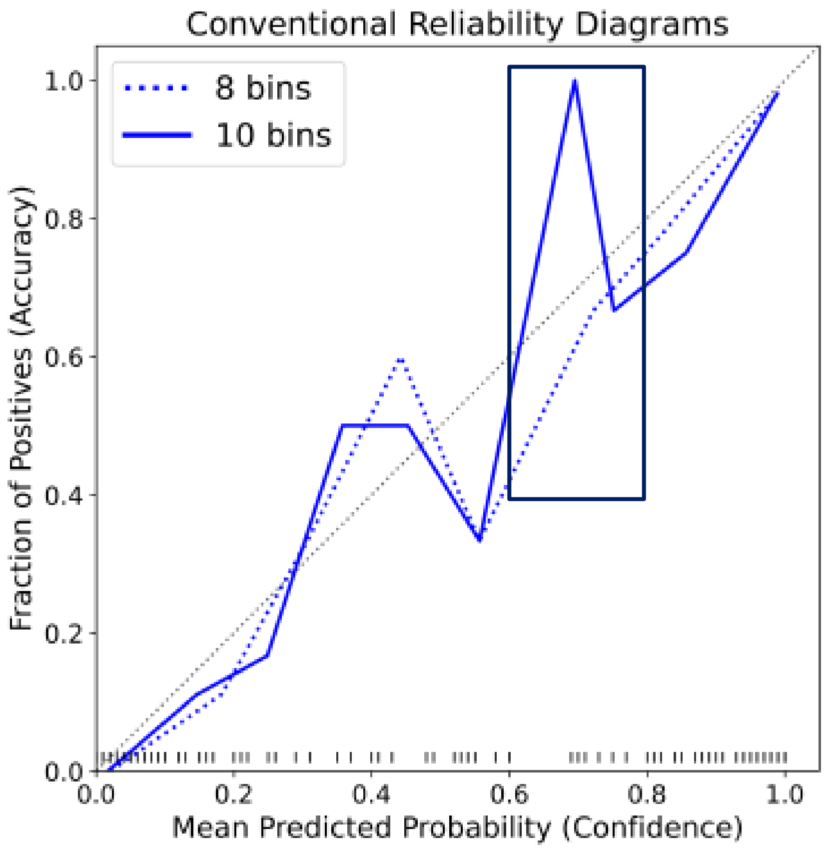
Bins: Number of discrete bins

Strategy: Bins of uniform size (*UNIFORM*) or of equal number of instances (*QUANTILE*)



F

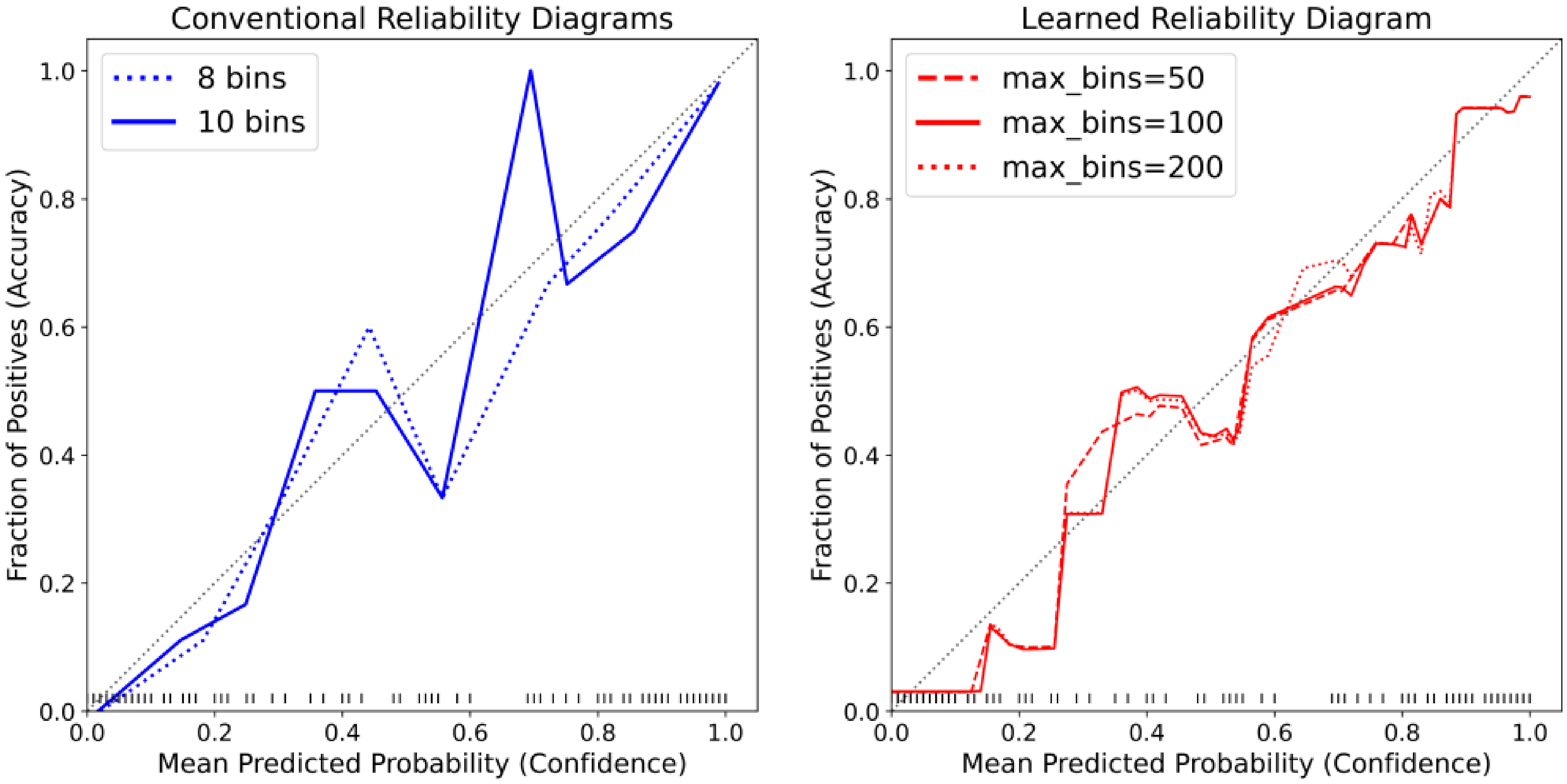
Subtle parameter changes have big impacts



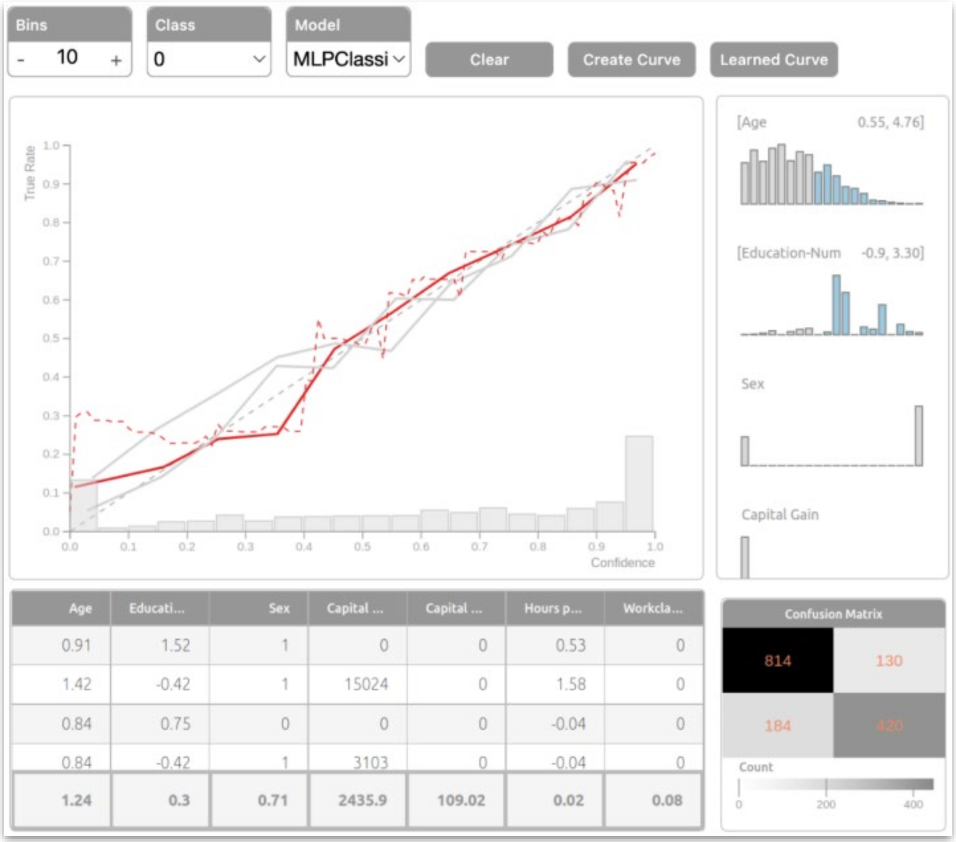
Using 10 bins suggests the model is miscalibrated for predictions in the 0.6-0.8 range.

But, using 8 bins indicates a fairly calibrated model.

Learned reliability diagrams *learn* the relationship between predictions and outcomes



Xenopoulos et. al. (2022)

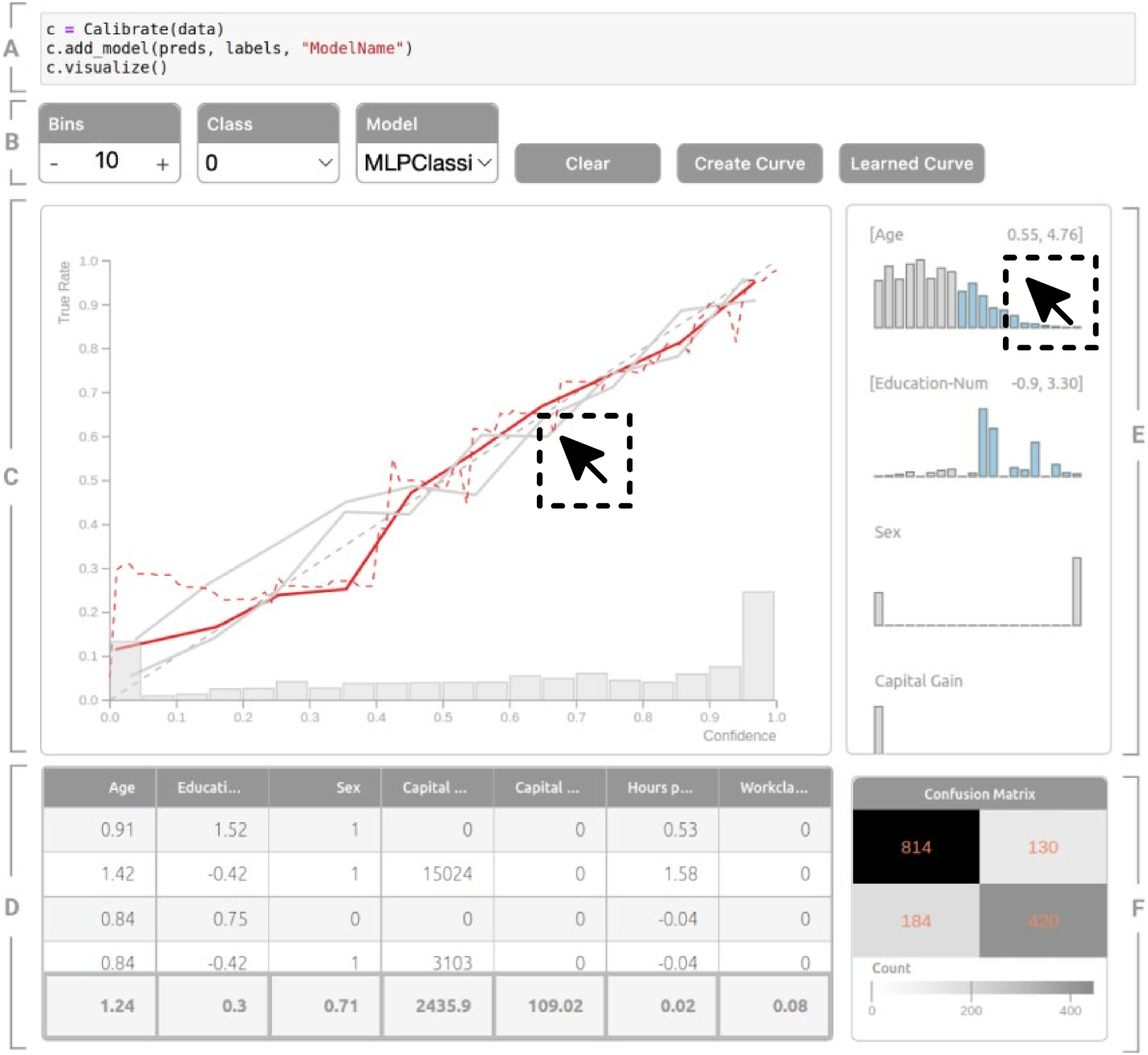


Easy to use

Parameter control

Reliability diagrams

Instance inspection



Interactive component

Subset creation

Performance metrics

Recap: Model Assessment

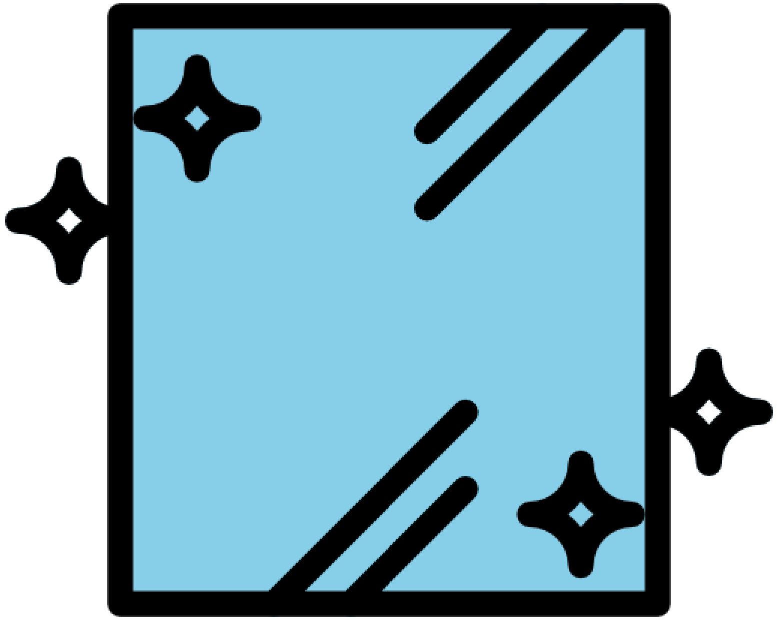
Analyzing model performance is a critical task in a machine learning workflow.

Visualization is useful for understanding context and conveying performance to stakeholders.

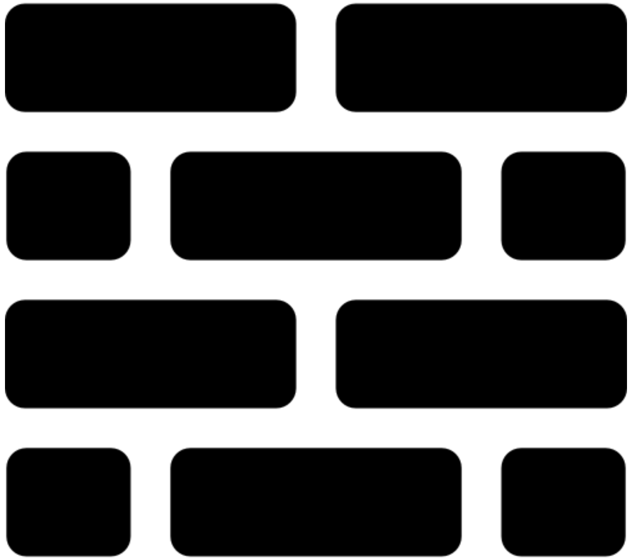
Many classical, static visualization techniques are now benefiting from reworked visual representations and interactivity.

Model understanding

Agenda



White/Glass Box



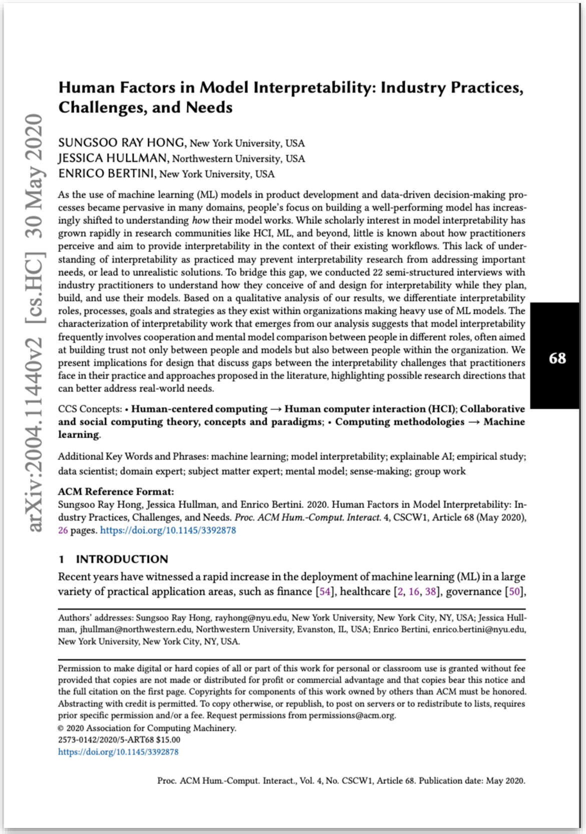
Black Box

Why Model Interpretation & Explanation?

Model Validation and Improvement

Decision-Making and Knowledge Discovery

Gain Confidence and Obtain Trust



arXiv:2004.11440v2 [cs.HC] 30 May 2020

Human Factors in Model Interpretability: Industry Practices, Challenges, and Needs

SUNGSOO RAY HONG, New York University, USA
JESSICA HULLMAN, Northwestern University, USA
ENRICO BERTINI, New York University, USA

As the use of machine learning (ML) models in product development and data-driven decision-making processes became pervasive in many domains, people's focus on building a well-performing model has increasingly shifted to understanding *how* their model works. While scholarly interest in model interpretability has grown rapidly in research communities like HCI, ML, and beyond, little is known about how practitioners perceive and aim to provide interpretability in the context of their existing workflows. This lack of understanding of interpretability as practiced may prevent interpretability research from addressing important needs, or lead to unrealistic solutions. To bridge this gap, we conducted 22 semi-structured interviews with industry practitioners to understand how they conceive of and design for interpretability while they plan, build, and use their models. Based on a qualitative analysis of our results, we differentiate interpretability roles, processes, goals and strategies as they exist within organizations making heavy use of ML models. The characterization of interpretability work that emerges from our analysis suggests that model interpretability frequently involves cooperation and mental model comparison between people in different roles, often aimed at building trust not only between people and models but also between people within the organization. We present implications for design that discuss gaps between the interpretability challenges that practitioners face in their practice and approaches proposed in the literature, highlighting possible research directions that can better address real-world needs.

68

CCS Concepts: • **Human-centered computing → Human computer interaction (HCI); Collaborative and social computing theory, concepts and paradigms**; • **Computing methodologies → Machine learning**.

Additional Key Words and Phrases: machine learning; model interpretability; explainable AI; empirical study; data scientist; domain expert; subject matter expert; mental model; sense-making; group work

ACM Reference Format:
Sungsoo Ray Hong, Jessica Hullman, and Enrico Bertini. 2020. Human Factors in Model Interpretability: Industry Practices, Challenges, and Needs. *Proc. ACM Hum.-Comput. Interact.* 4, CSCW1, Article 68 (May 2020), 26 pages. https://doi.org/10.1145/3392878

1 INTRODUCTION

Recent years have witnessed a rapid increase in the deployment of machine learning (ML) in a large variety of practical application areas, such as finance [54], healthcare [2, 16, 38], governance [50],

Authors' addresses: Sungsoo Ray Hong, rayhong@nyu.edu, New York University, New York City, NY, USA; Jessica Hullman, jhullman@northwestern.edu, Northwestern University, Evanston, IL, USA; Enrico Bertini, enrico.bertini@nyu.edu, New York University, New York City, NY, USA.

2573-0142/2020/5-ART68 $15.00
https://doi.org/10.1145/3392878

UIC COMPUTER SCIENCE

How Do We Interpret Model Behavior?

Methods for machine learning model interpretation can be classified according to various criteria:

White-box / Intrinsic interpretability: Machine learning models that are considered interpretable due to their simple structure, such as *short* decision trees or *sparse* linear models. Interpretability is gained by explaining the internal structure of the model.

Black-box / Post-hoc interpretability: Machine learning models that are hard to gain a comprehensive understanding of their inner working (e.g., deep neural networks) are considered black boxes. Interpretability is gained by explaining the model behavior after training.

Generalized Additive Models

White-Box Model

Generalized Additive Models (GAMs)

Generalized additive models extend standard linear models by allowing non-linear functions of each of the variables.

$$
\begin{aligned}
y_i &= \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}) + \epsilon_i \\
&= \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip}) + \epsilon_i .
\end{aligned}
$$

Generalized Additive Models (GAMs)

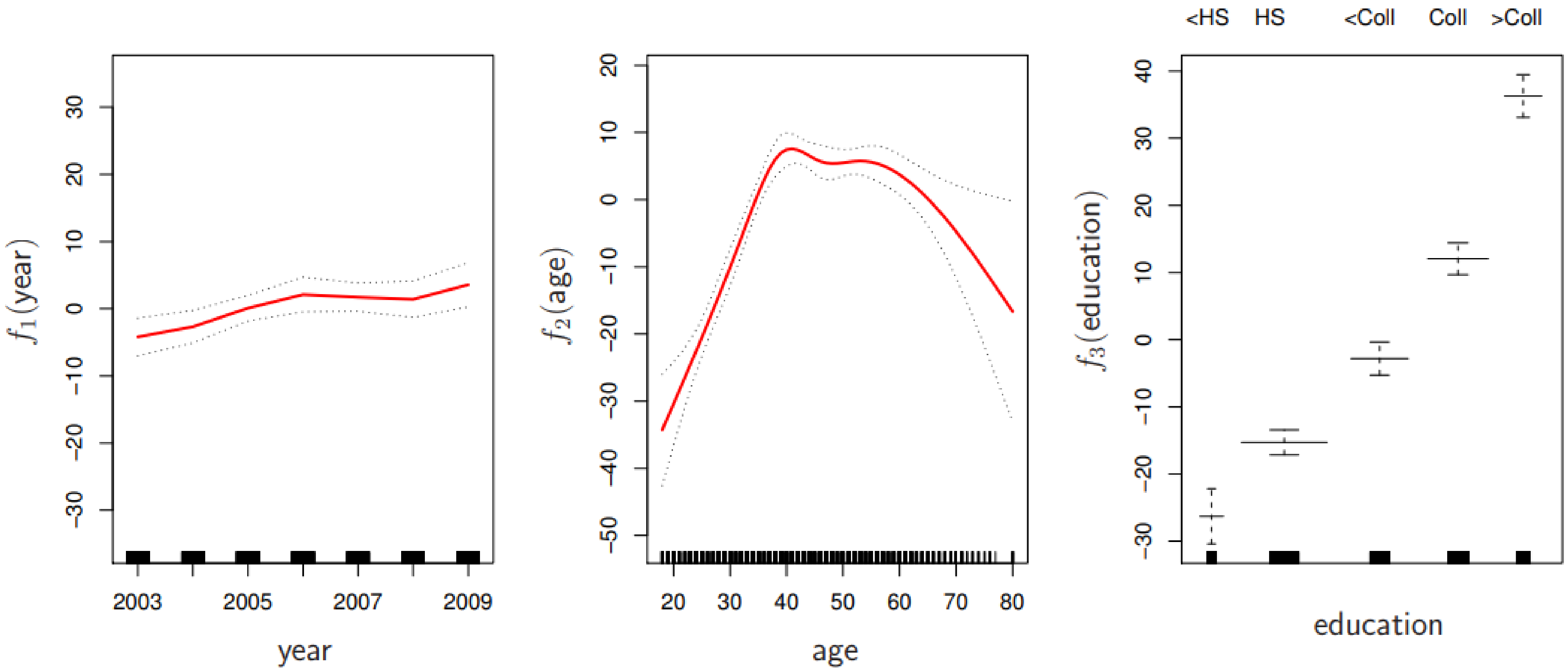
Generalized additive models extend standard linear models by allowing non-linear functions of each of the variables.

$$\begin{aligned} y_i &= \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}) + \epsilon_i \\ &= \beta_0 + \boxed{f_1}(x_{i1}) + \boxed{f_2}(x_{i2}) + \cdots + \boxed{f_p}(x_{ip}) + \epsilon_i. \end{aligned}$$

Shape Functions

Generalized Additive Models (GAMs): An Example

$$\text{Wage} = f(\text{year}, \text{age}, \text{education}) = b_0 + f_1(\text{year}) + f_2(\text{age}) + f_3(\text{education})$$



Generalized Additive Models (GAMs): Pros and Cons

Pros

GAMs allow us to fit a non-linear f_j to each X_j , so that we can model non-linear relationships easily.

The non-linear fits can potentially lead to better predictions.

Because the model is additive, we can examine the effect of each X_j on Y for each observation. This is useful for visualization.

Cons

GAMs are restricted to be additive. With many variables, important interactions can be missed or computationally infeasible to find.

Explainable Boosting Machines

$$g(E[y]) = \beta_0 + \sum f_j(x_j)$$

However, as with linear regression, we can manually add interaction terms to the GAM model by including additional predictors of the form $X_j \times X_k$, which necessitates shape function $f_{jk}(X_j , X_k)$, into the model.

$$g(E[y]) = \beta_0 + \sum f_j(x_j) + \sum f_{i_j}(x_i, x_j)$$

Explainable Boosting Machines in Practice

```
import pandas as pd
from sklearn.model_selection import train_test_split

from interpret.glassbox import ExplainableBoostingClassifier
from interpret import show

# Read data
df = pd.read_csv(...)

# Separate into data/label
label = df.columns[-1]
X = df[train_cols]
y = df[label]

# Generate test/train
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)

# Train EBM
ebm = ExplainableBoostingClassifier()
ebm.fit(X_train, y_train)
```

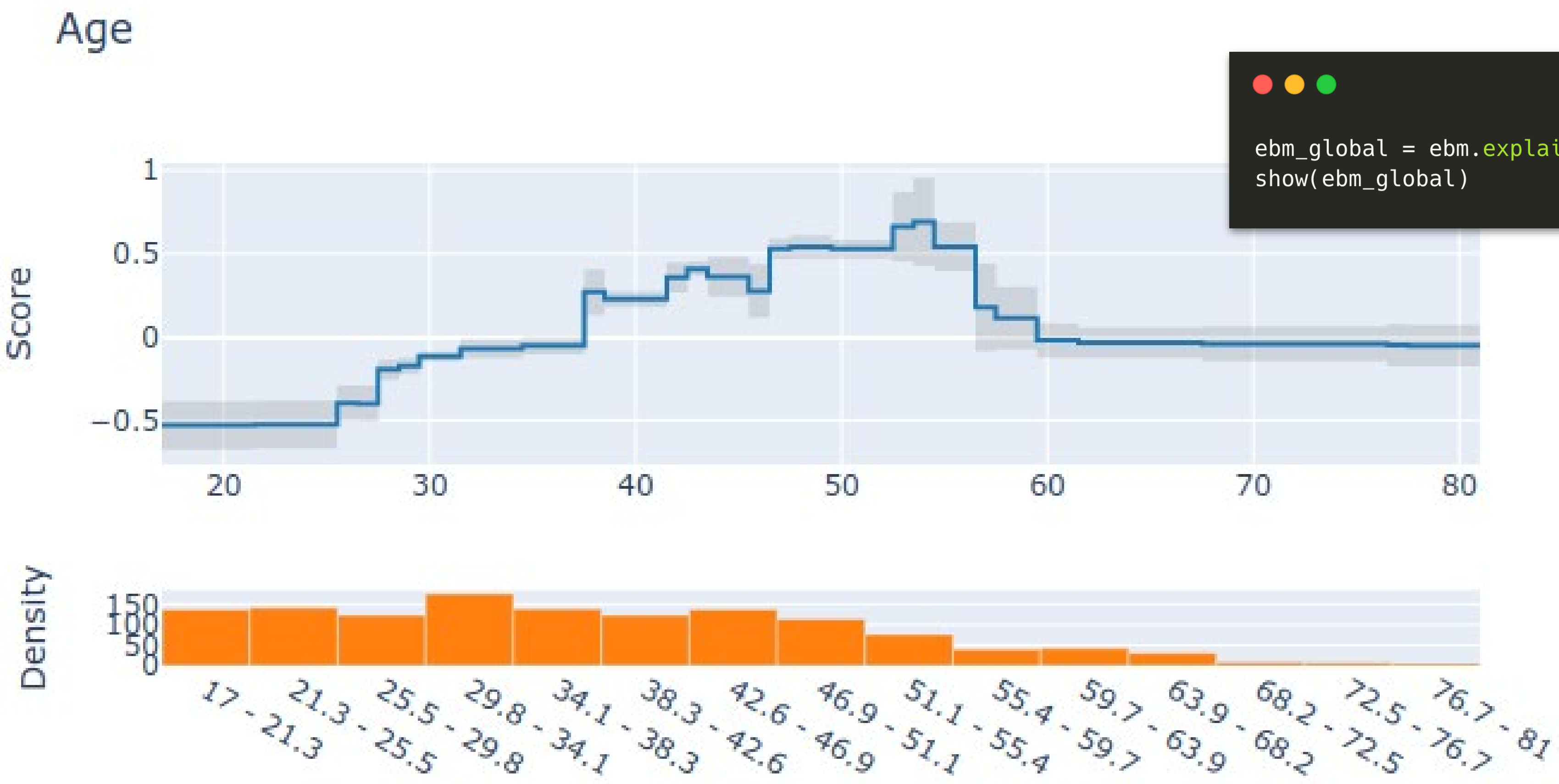
Explainable Boosting Machines in Practice

```python
import pandas as pd
from sklearn.model_selection import train_test_split

from interpret.glassbox import ExplainableBoostingClassifier
from interpret import show

# Read data
df = pd.read_csv(...)

# Separate into data/label
label = df.columns[-1]
X = df[train_cols]
y = df[label]

# Generate test/train
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)

# Train EBM
ebm = ExplainableBoostingClassifier()
ebm.fit(X_train, y_train)
```

EBM Link

interpret.ml

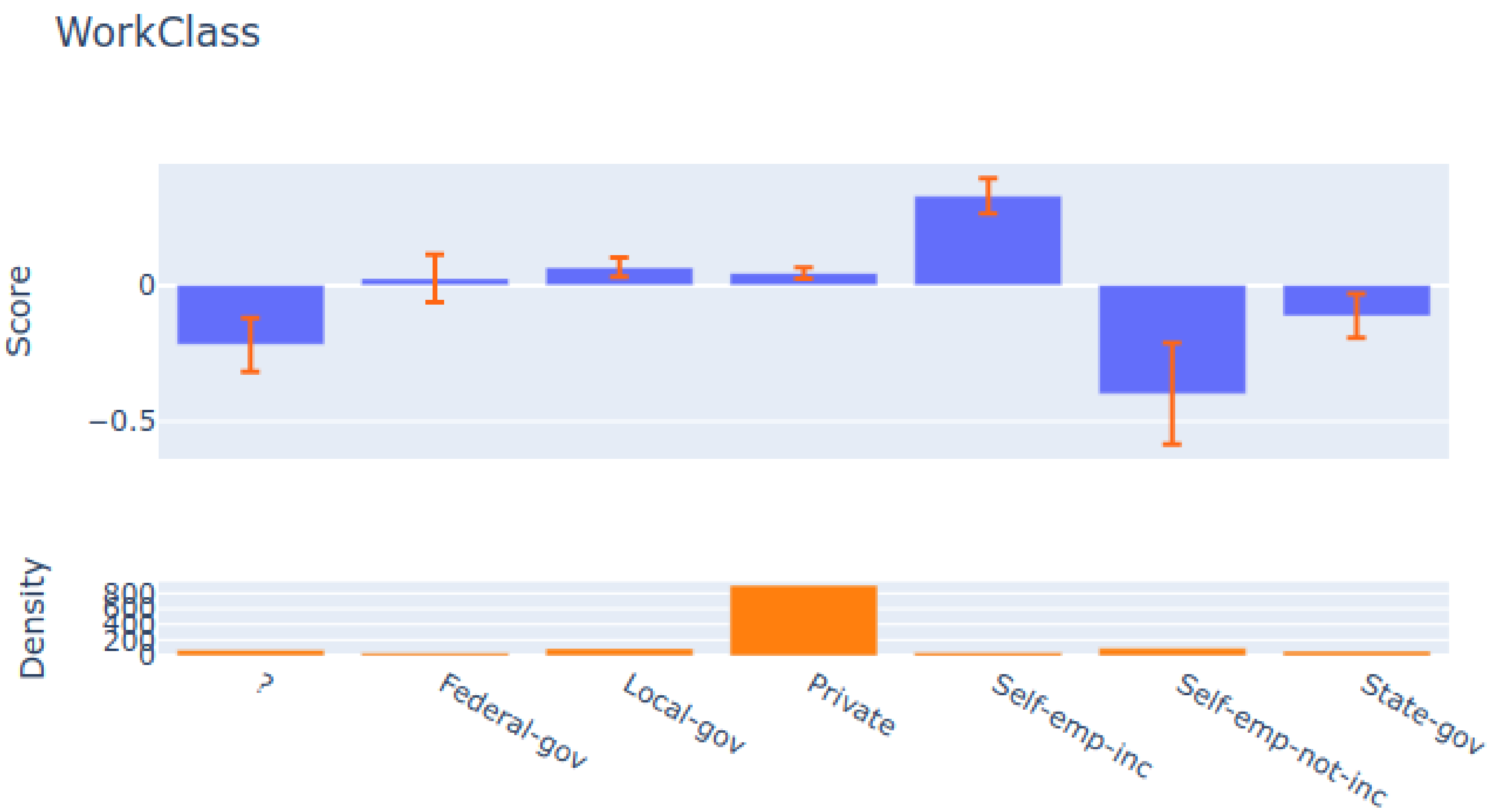
Visualizing EBMs (or GAMs)




```python
ebm_global = ebm.explain_global()
show(ebm_global)
```

Partial Dependency Plot

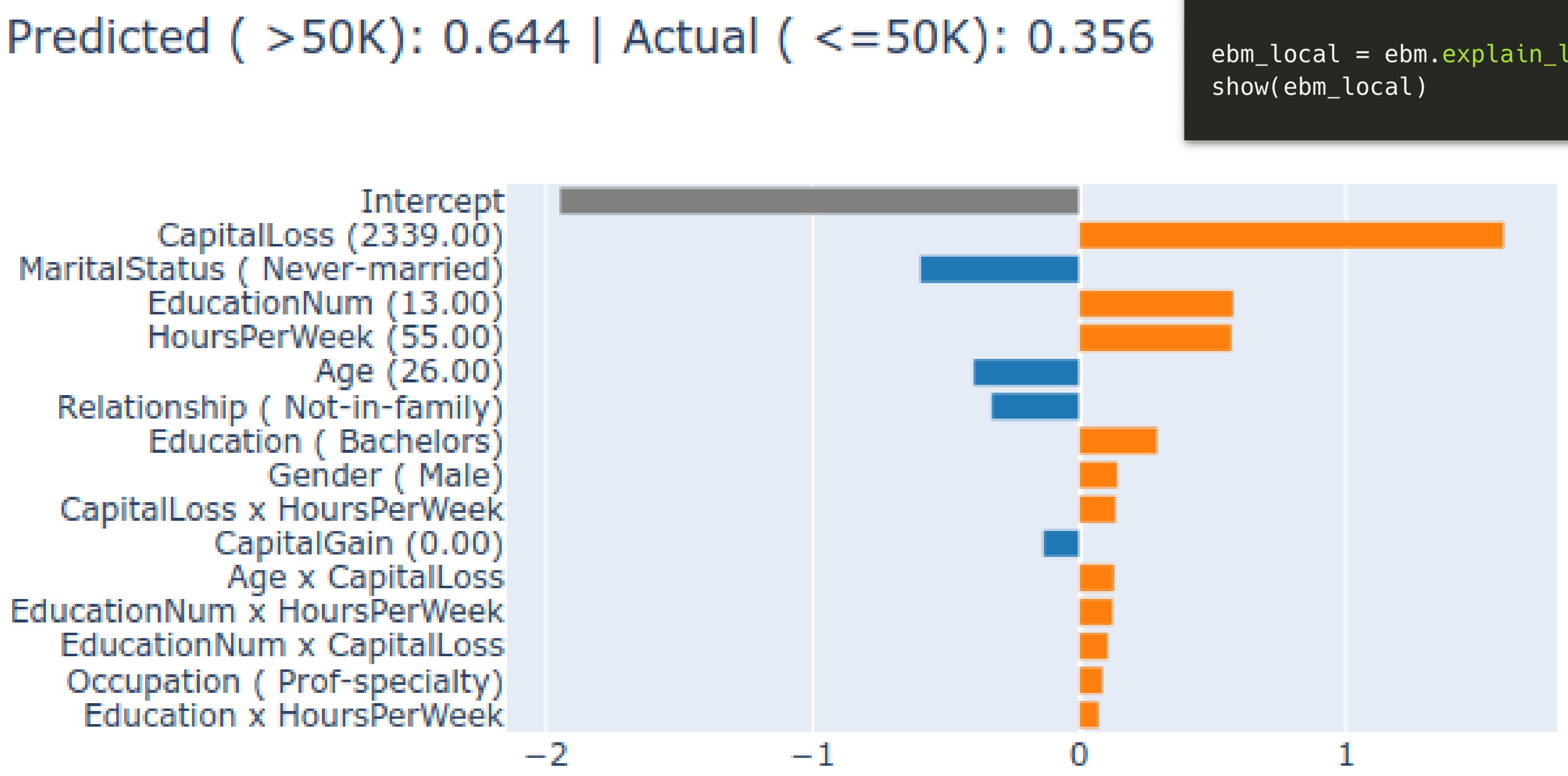
Visualizing EBMs (or GAMs)



Partial Dependency Plot

Visualizing EBMs (or GAMs)

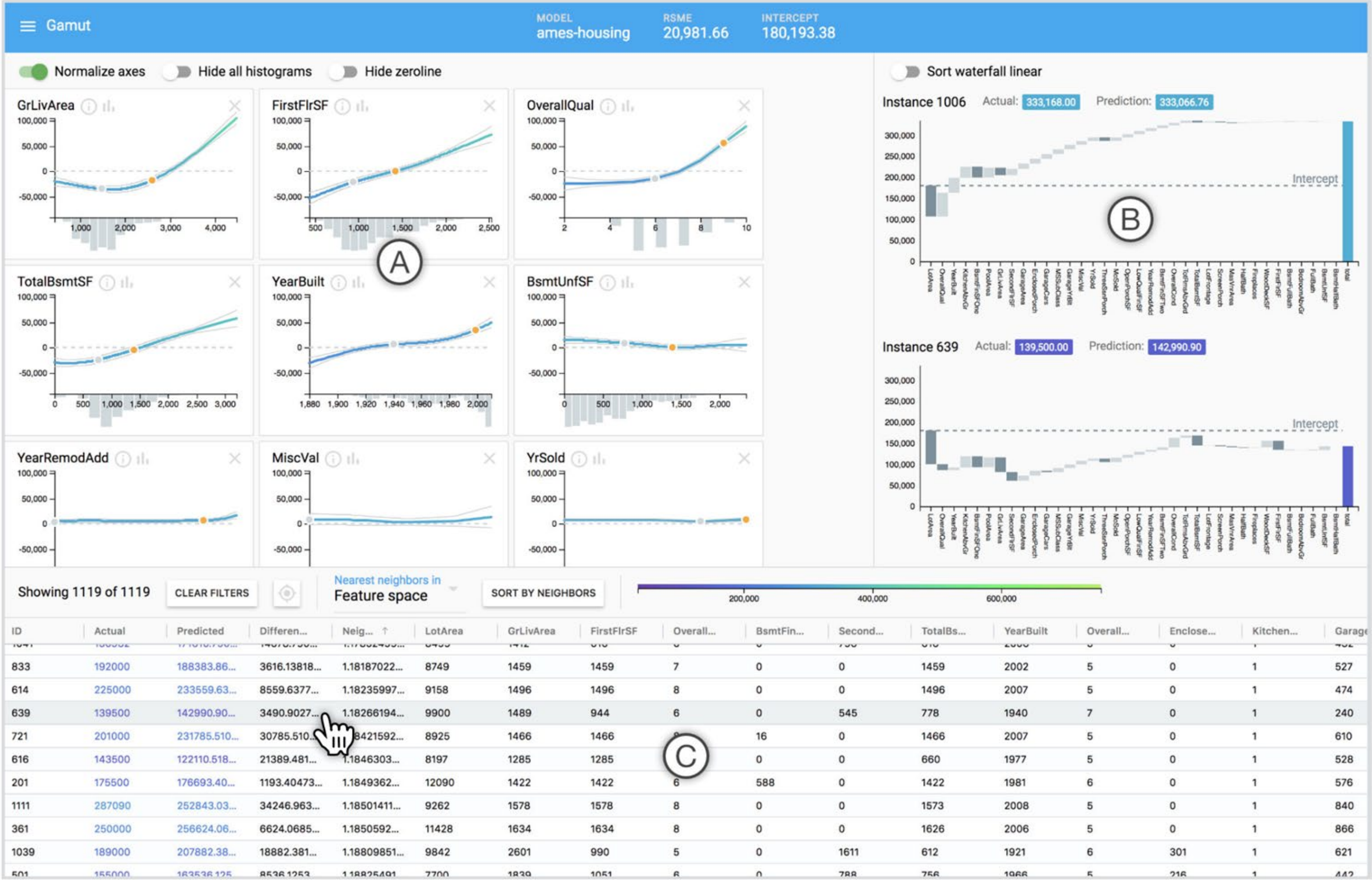
```
ebm_local = ebm.explain_local(X_test[:5], y_test[:5])
show(ebm_local)
```



Feature contributions for an individual observation

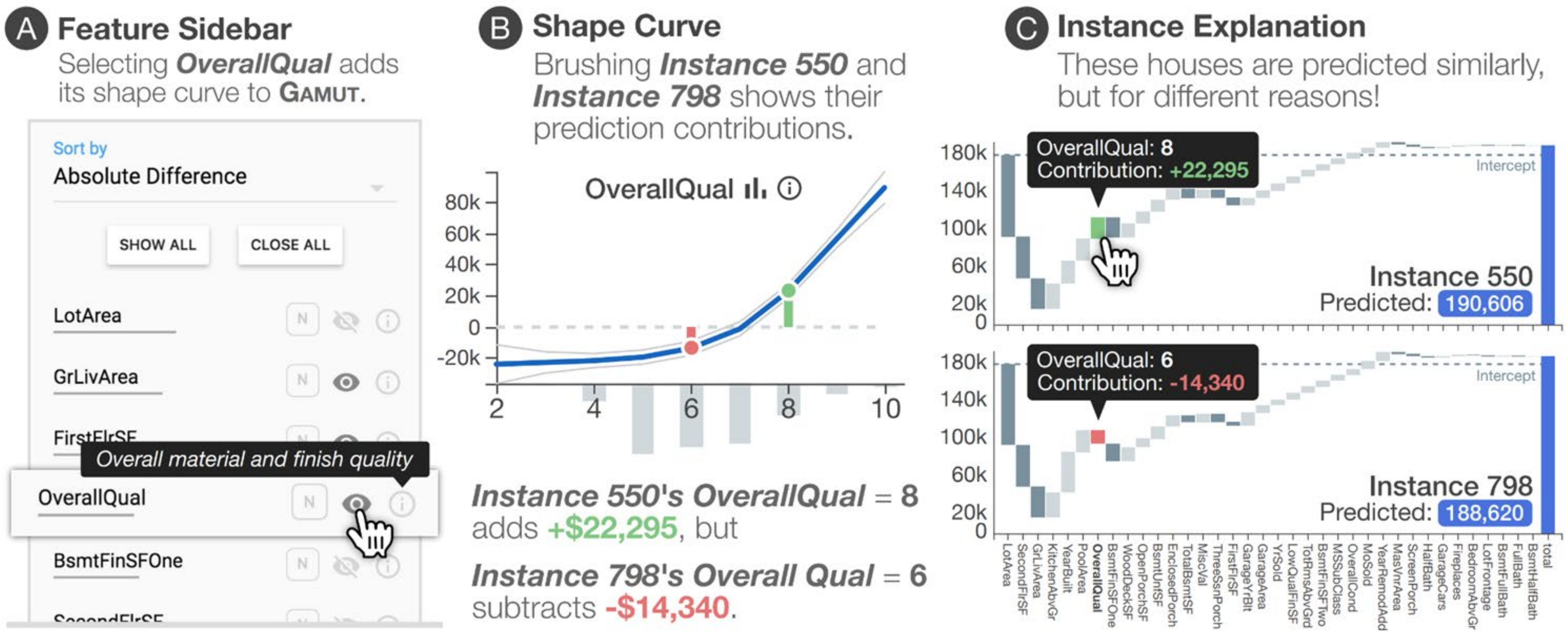
UIC COMPUTER SCIENCE

Gamut



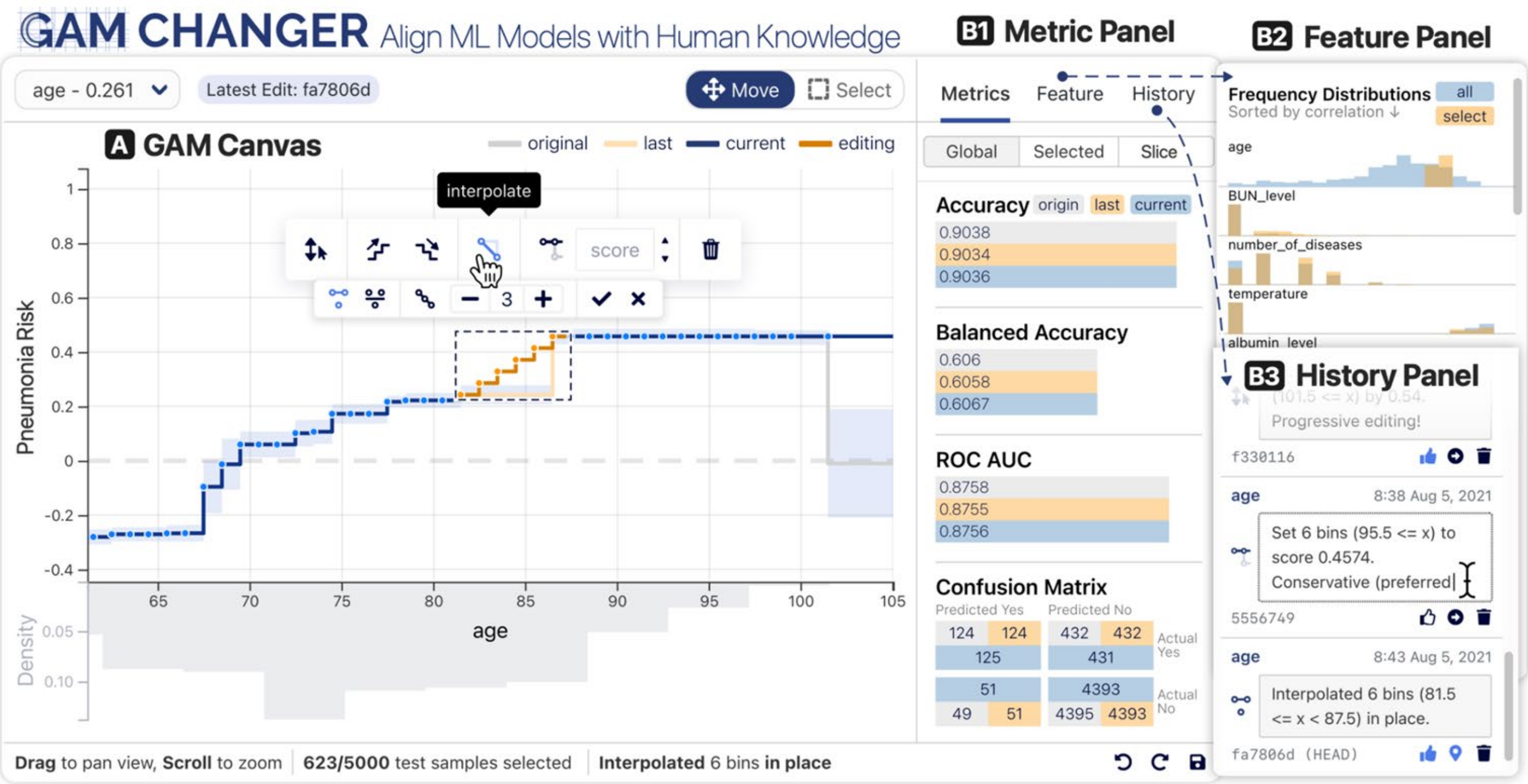
https://www.microsoft.com/en-us/research/publication/gamut-a-design-probe-to-understand-howdata-scientists-understand-machine-learning-models/

Gamut

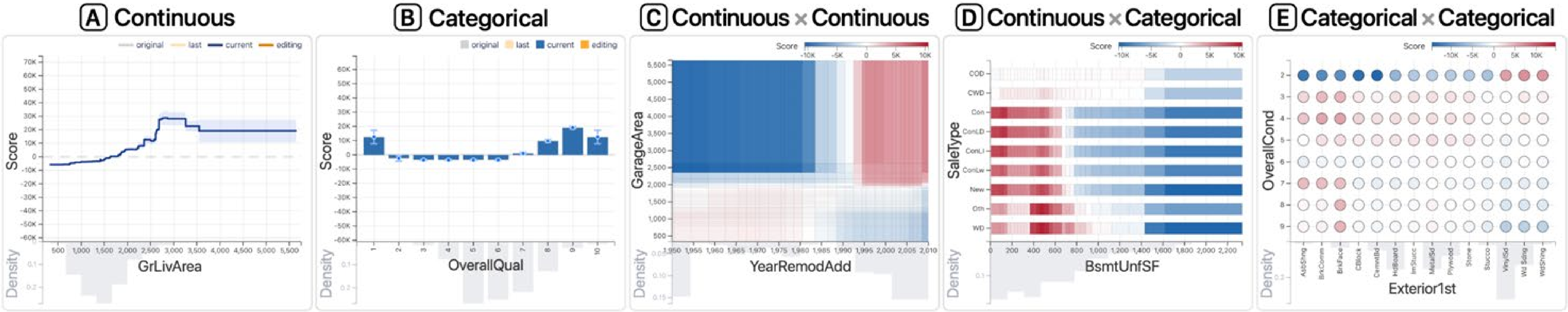


UIC COMPUTER SCIENCE

GAM Changer


GAM CHANGER Align ML Models with Human Knowledge
B1 Metric Panel
B2 Feature Panel
A GAM Canvas
B3 History Panel
age - 0.261
Latest Edit: fa7806d
Move
Select
original
last
current
editing
interpolate
score
Pneumonia Risk
age
Density
Metrics
Feature
History
Global
Selected
Slice
Accuracy
0.9038
0.9034
0.9036
Balanced Accuracy
0.606
0.6058
0.6067
ROC AUC
0.8758
0.8755
0.8756
Confusion Matrix
Frequency Distributions
Sorted by correlation
Progressive editing!
Set 6 bins (95.5 <= x) to score 0.4574. Conservative (preferred)
Interpolated 6 bins (81.5 <= x < 87.5) in place.
Drag to pan view, Scroll to zoom
623/5000 test samples selected
Interpolated 6 bins in place

GAM Changer


A Continuous
B Categorical
C Continuous × Continuous
D Continuous × Categorical
E Categorical × Categorical
original last current editing
Score
Density
GrLivArea
OverallQual
GarageArea
YearRemodAdd
SaleType
BsmtUnfSF
OverallCond
Exterior1st

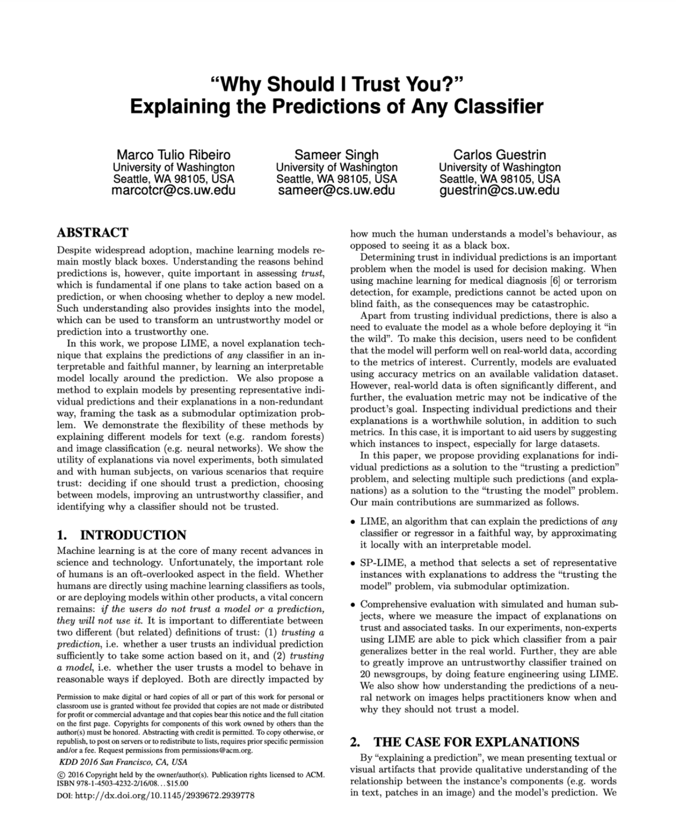
Black Box Methods

When to use black-box explanation methods?

Oftentimes, complex model architectures obfuscate the internal logic of the model (e.g., neural networks, support vector machines, etc.)

In these cases, we turn to “black-box” explanation methods. These methods are gaining significant popularity among machine learning practitioners.

Locally Interpretable Model Explanations (LIME)



"Why Should I Trust You?"
Explaining the Predictions of Any Classifier

Marco Tulio Ribeiro
University of Washington
Seattle, WA 98105, USA
marcotcr@cs.uw.edu

Sameer Singh
University of Washington
Seattle, WA 98105, USA
sameer@cs.uw.edu

Carlos Guestrin
University of Washington
Seattle, WA 98105, USA
guestrin@cs.uw.edu

ABSTRACT

Despite widespread adoption, machine learning models remain mostly black boxes. Understanding the reasons behind predictions is, however, quite important in assessing *trust*, which is fundamental if one plans to take action based on a prediction, or when choosing whether to deploy a new model. Such understanding also provides insights into the model, which can be used to transform an untrustworthy model or prediction into a trustworthy one.

In this work, we propose LIME, a novel explanation technique that explains the predictions of *any* classifier in an interpretable and faithful manner, by learning an interpretable model locally around the prediction. We also propose a method to explain models by presenting representative individual predictions and their explanations in a non-redundant way, framing the task as a submodular optimization problem. We demonstrate the flexibility of these methods by explaining different models for text (e.g. random forests) and image classification (e.g. neural networks). We show the utility of explanations via novel experiments, both simulated and with human subjects, on various scenarios that require trust: deciding if one should trust a prediction, choosing between models, improving an untrustworthy classifier, and identifying why a classifier should not be trusted.

1. INTRODUCTION

Machine learning is at the core of many recent advances in science and technology. Unfortunately, the important role of humans is an oft-overlooked aspect in the field. Whether humans are directly using machine learning classifiers as tools, or are deploying models within other products, a vital concern remains: *if the users do not trust a model or a prediction, they will not use it.* It is important to differentiate between two different (but related) definitions of trust: (1) *trusting a prediction*, i.e. whether a user trusts an individual prediction sufficiently to take some action based on it, and (2) *trusting a model*, i.e. whether the user trusts a model to behave in reasonable ways if deployed. Both are directly impacted by how much the human understands a model's behaviour, as opposed to seeing it as a black box.

KDD 2016 San Francisco, CA, USA

ISBN 978-1-4503-4232-2/16/08...$15.00
DOI: http://dx.doi.org/10.1145/2939672.2939778

Determining trust in individual predictions is an important problem when the model is used for decision making. When using machine learning for medical diagnosis [6] or terrorism detection, for example, predictions cannot be acted upon on blind faith, as the consequences may be catastrophic.

Apart from trusting individual predictions, there is also a need to evaluate the model as a whole before deploying it "in the wild". To make this decision, users need to be confident that the model will perform well on real-world data, according to the metrics of interest. Currently, models are evaluated using accuracy metrics on an available validation dataset. However, real-world data is often significantly different, and further, the evaluation metric may not be indicative of the product's goal. Inspecting individual predictions and their explanations is a worthwhile solution, in addition to such metrics. In this case, it is important to aid users by suggesting which instances to inspect, especially for large datasets.

In this paper, we propose providing explanations for individual predictions as a solution to the "trusting a prediction" problem, and selecting multiple such predictions (and explanations) as a solution to the "trusting the model" problem. Our main contributions are summarized as follows.

- LIME, an algorithm that can explain the predictions of *any* classifier or regressor in a faithful way, by approximating it locally with an interpretable model.
- SP-LIME, a method that selects a set of representative instances with explanations to address the "trusting the model" problem, via submodular optimization.
- Comprehensive evaluation with simulated and human subjects, where we measure the impact of explanations on trust and associated tasks. In our experiments, non-experts using LIME are able to pick which classifier from a pair generalizes better in the real world. Further, they are able to greatly improve an untrustworthy classifier trained on 20 newsgroups, by doing feature engineering using LIME. We also show how understanding the predictions of a neural network on images helps practitioners know when and why they should not trust a model.

2. THE CASE FOR EXPLANATIONS

By "explaining a prediction", we mean presenting textual or visual artifacts that provide qualitative understanding of the relationship between the instance's components (e.g. words in text, patches in an image) and the model's prediction. We

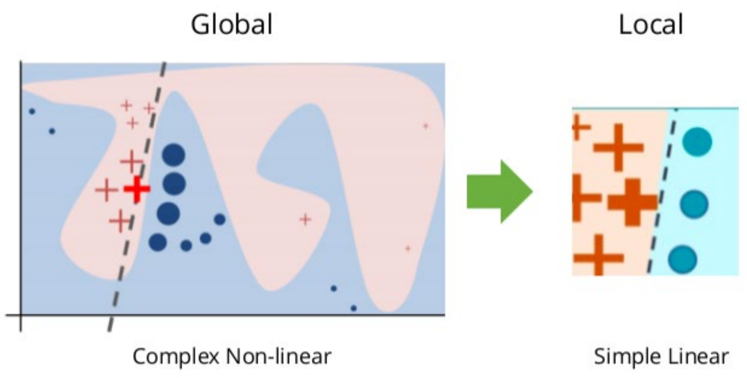
LIME is a popular technique to produce *local explanations*, which produces feature attributions for a given observation.

LIME is applicable to many kinds of data, including tabular, text and image.

LIME

Problem: Complex models lack the ability to generate explanations for individual observations.

Idea: Use a *surrogate model* to estimate the local behavior of a model using an interpretable model.



LIME

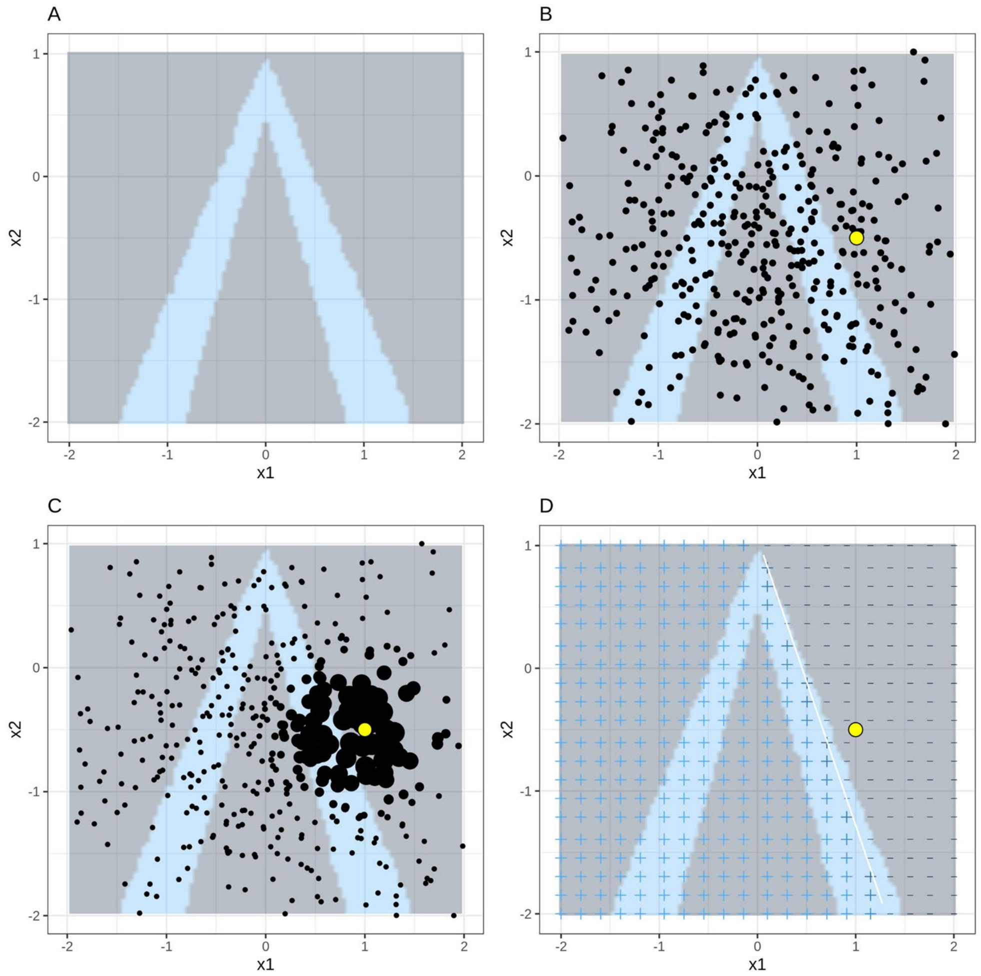
LIME Framework

Select observation of interest.

Perturb dataset and generate predictions using black box model.

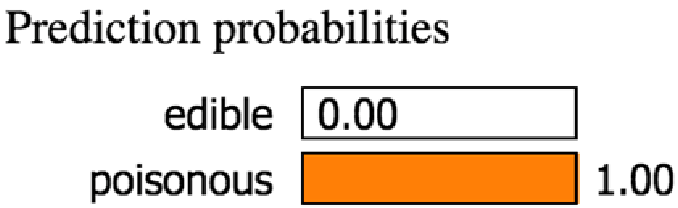
Train an interpretable model (e.g., linear regression) on the perturbed dataset.

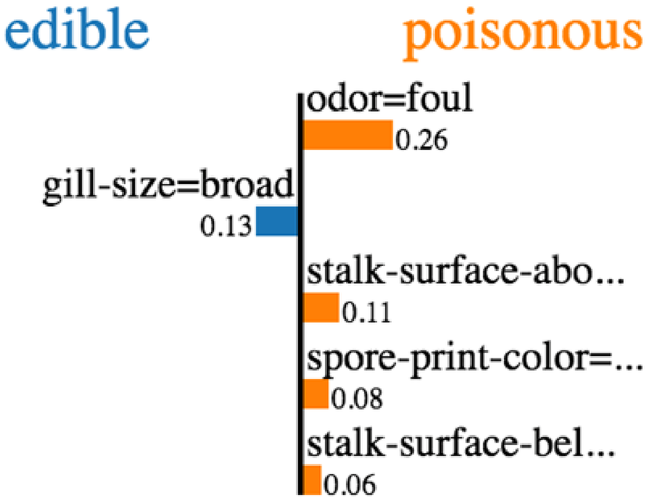
Use weights of interpretable model to explain the prediction.



LIME

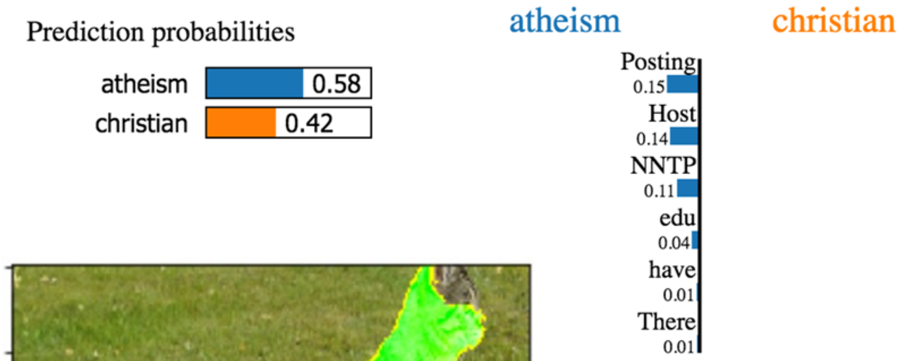
```
import lime
import lime.lime_tabular

explainer = lime.lime_tabular.LimeTabularExplainer(data, ...)

idx = 1
exp = explainer.explain_instance(data[idx], clf.predict_proba, ...)
```

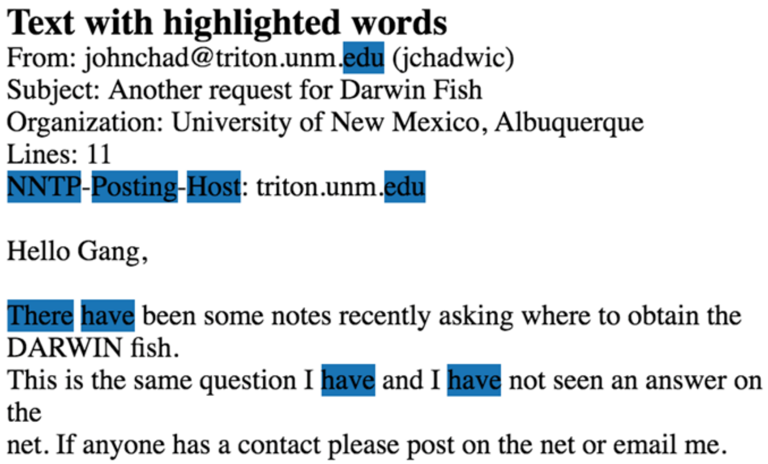




Feature	Value
odor=foul	True
gill-size=broad	True
stalk-surface-above-ring=silky	True
spore-print-color=chocolate	True
stalk-surface-below-ring=silky	True

LIME for Text and Images





LIME: Pros and Cons

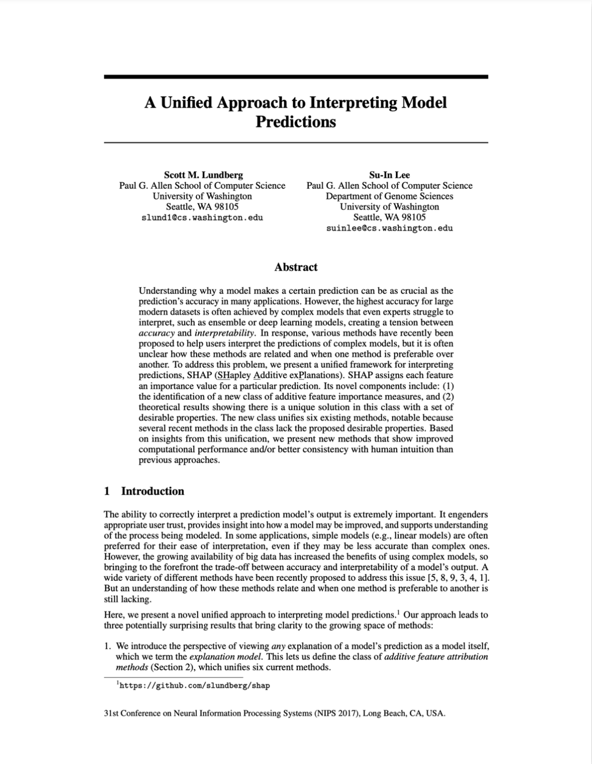
Pros

- Generalized to any underlying black-box model.
- Simple to use and understand.

Cons

- Parameters are hard to tune.
- Results can be unstable.

SHapley Additive exPlanations



A Unified Approach to Interpreting Model Predictions

Scott M. Lundberg
Paul G. Allen School of Computer Science
University of Washington
Seattle, WA 98105
slund1@cs.washington.edu

Su-In Lee
Paul G. Allen School of Computer Science
Department of Genome Sciences
University of Washington
Seattle, WA 98105
suinlee@cs.washington.edu

Abstract

Understanding why a model makes a certain prediction can be as crucial as the prediction's accuracy in many applications. However, the highest accuracy for large modern datasets is often achieved by complex models that even experts struggle to interpret, such as ensemble or deep learning models, creating a tension between *accuracy* and *interpretability*. In response, various methods have recently been proposed to help users interpret the predictions of complex models, but it is often unclear how these methods are related and when one method is preferable over another. To address this problem, we present a unified framework for interpreting predictions, SHAP (SHapley Additive exPlanations). SHAP assigns each feature an importance value for a particular prediction. Its novel components include: (1) the identification of a new class of additive feature importance measures, and (2) theoretical results showing there is a unique solution in this class with a set of desirable properties. The new class unifies six existing methods, notable because several recent methods in the class lack the proposed desirable properties. Based on insights from this unification, we present new methods that show improved computational performance and/or better consistency with human intuition than previous approaches.

1 Introduction

The ability to correctly interpret a prediction model's output is extremely important. It engenders appropriate user trust, provides insight into how a model may be improved, and supports understanding of the process being modeled. In some applications, simple models (e.g., linear models) are often preferred for their ease of interpretation, even if they may be less accurate than complex ones. However, the growing availability of big data has increased the benefits of using complex models, so bringing to the forefront the trade-off between accuracy and interpretability of a model's output. A wide variety of different methods have been recently proposed to address this issue [5, 8, 9, 3, 4, 1]. But an understanding of how these methods relate and when one method is preferable to another is still lacking.

Here, we present a novel unified approach to interpreting model predictions.[1] Our approach leads to three potentially surprising results that bring clarity to the growing space of methods:

1. We introduce the perspective of viewing *any* explanation of a model's prediction as a model itself, which we term the *explanation model*. This lets us define the class of *additive feature attribution methods* (Section 2), which unifies six current methods.

[1] https://github.com/slundberg/shap

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

SHAP is another popular local explanation technique that based on game-theoretic optimal Shapley values.

The SHAP Python library contains rich visualization capabilities.

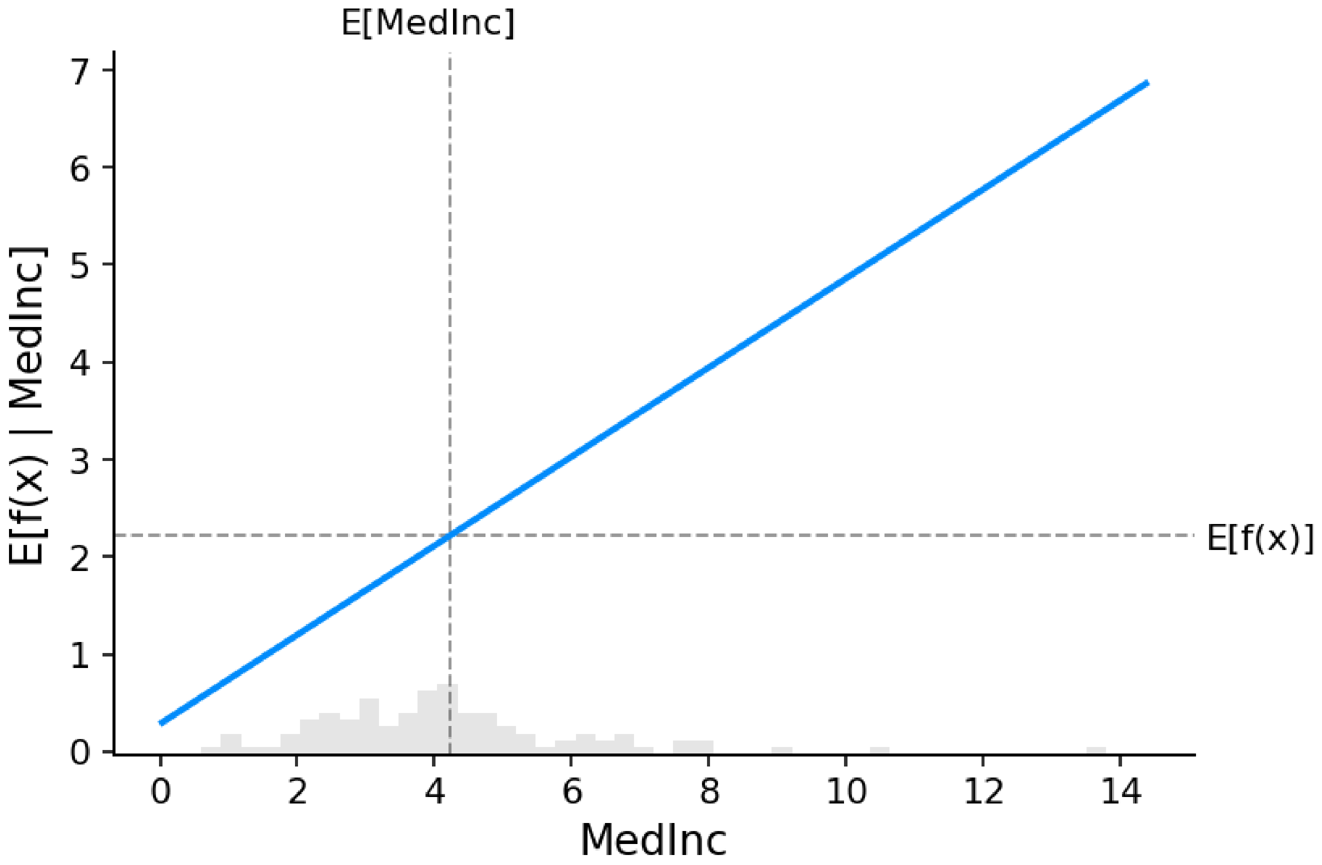
SHAP Basics

SHAP’s goal is to assign contribution for output f(x) for each feature.

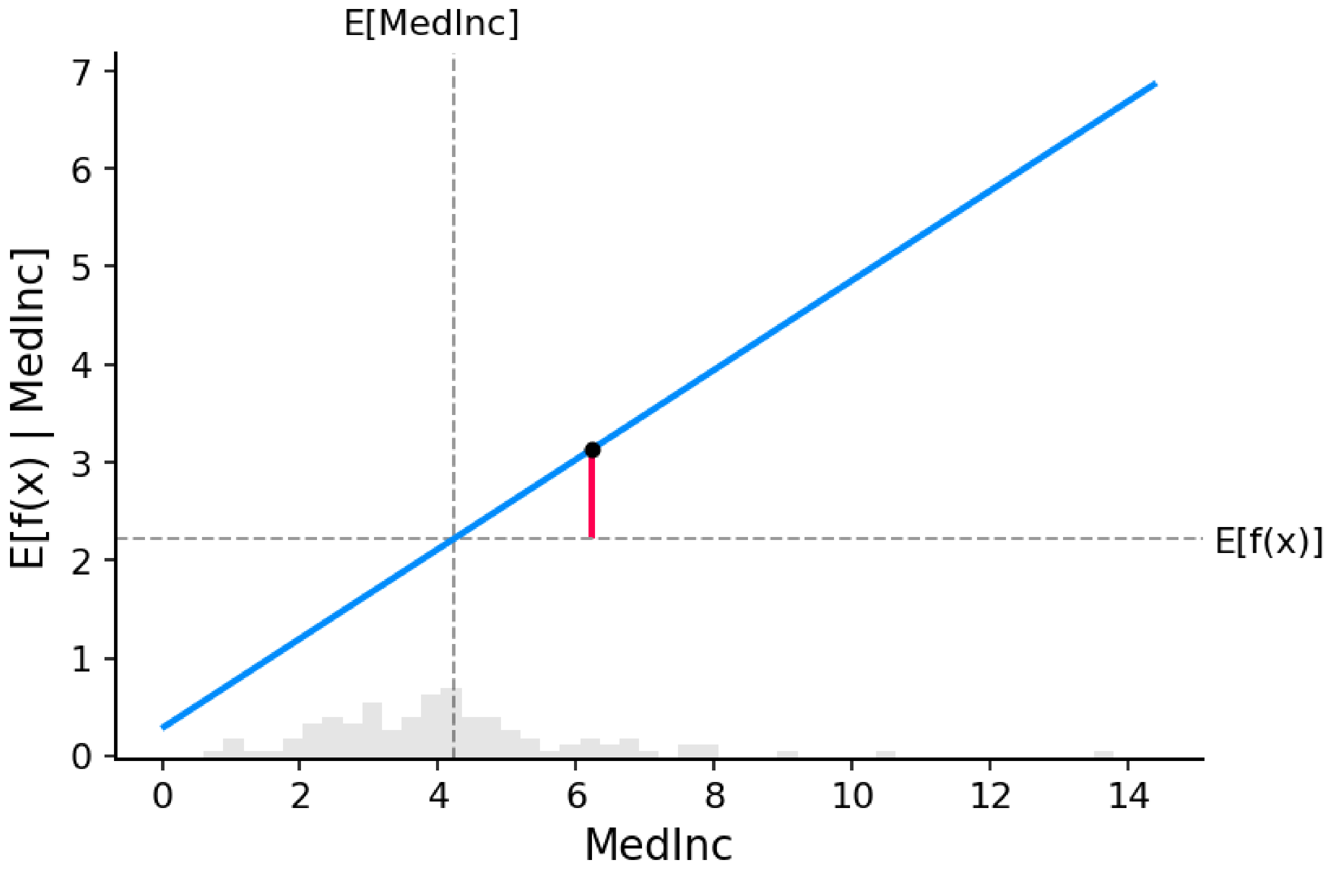
Each feature in the model represents a player in a game.

When a feature has “joined the game”, then we consider the value of that feature known.

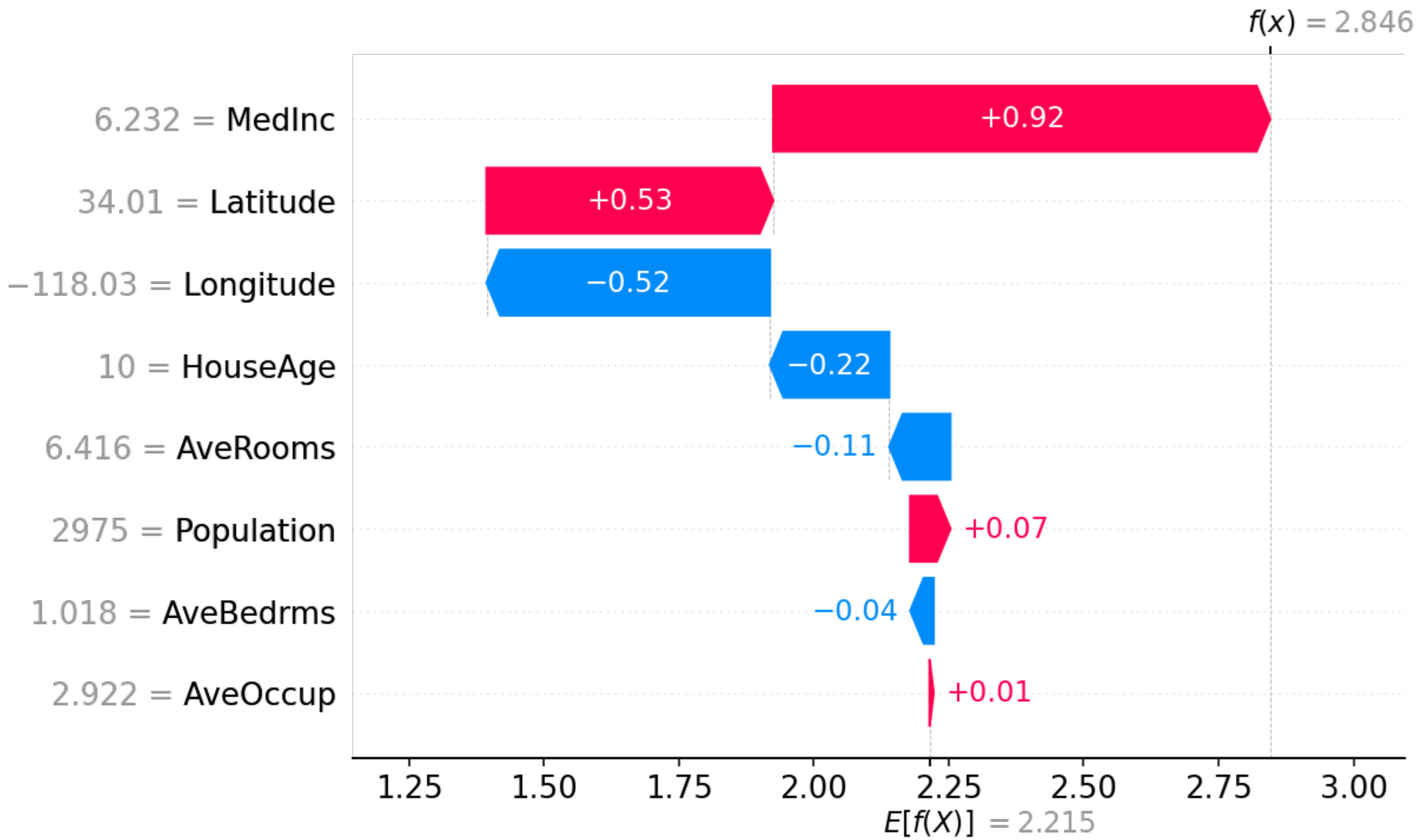
Relationship between SHAP and PDP’s



Relationship between SHAP and PDP's

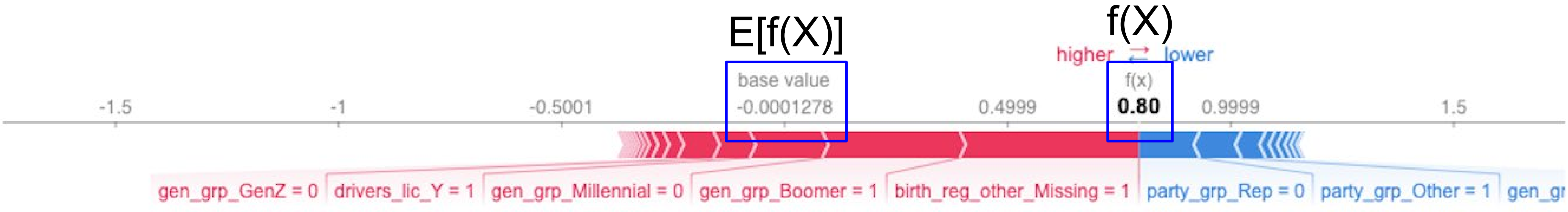


Sum(SHAP Values) = f(x) - E(f(X))

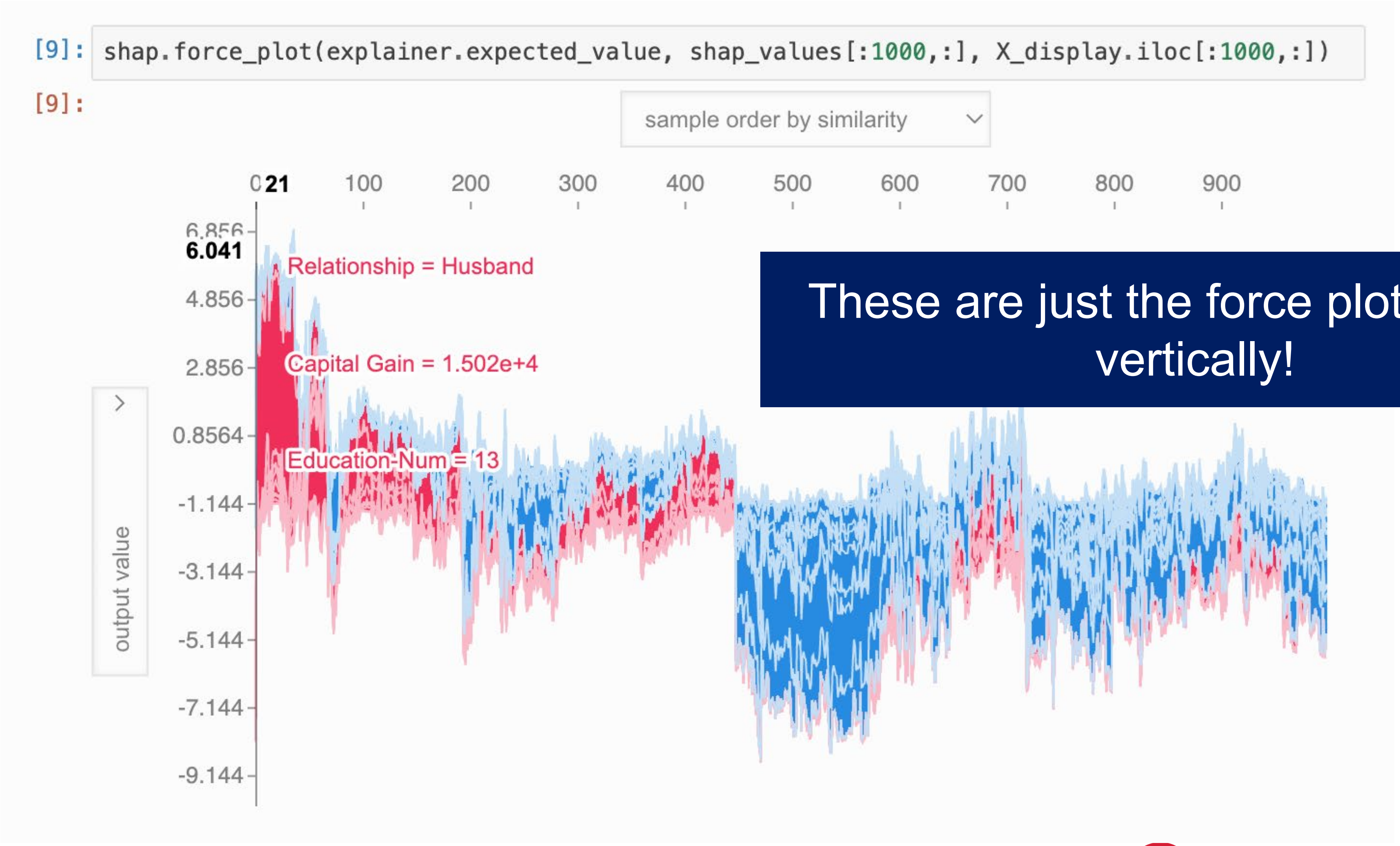


Visualizing SHAP Values

SHAP Force Plot



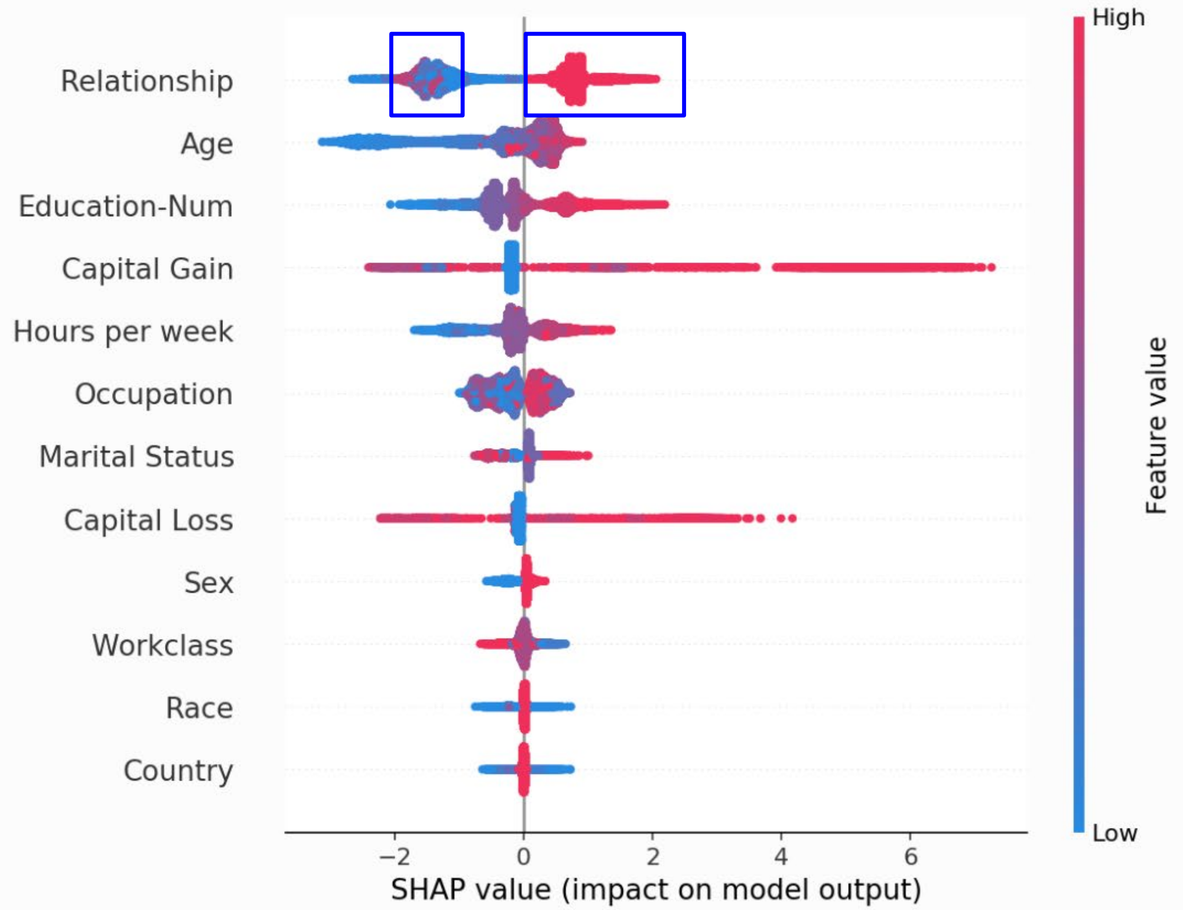
SHAP Force Plot



These are just the force plots flipped vertically!

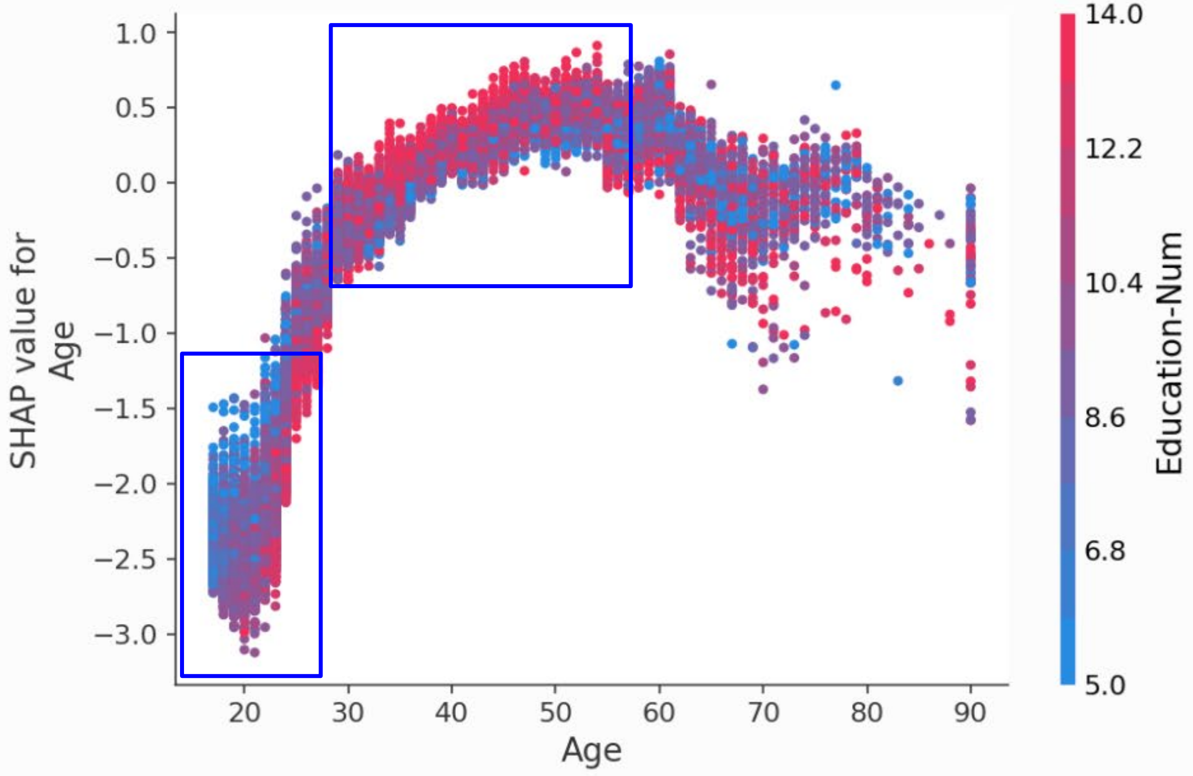
SHAP Summary Plot

```
[9]: shap.summary_plot(shap_values, X)
```



SHAP Dependence Plot

```
[11]: for name in X_train.columns:
          shap.dependence_plot(name, shap_values, X, display_features=X_display)
```



SHAP: Pros and Cons

Pros	Cons
Generalized to any underlying black-box model.	KernelSHAP can be very slow.
Strong visualization capabilities.	Does not account for feature correlation.
Theoretical foundation.	

Recap: Model Understanding

The ability to interpret models and explain their predictions is an increasing need for many practitioners.

Many libraries now contain the ability to visualize many aspects of model interpretability, oftentimes in interactive formats.

Popular methods like LIME & SHAP contain various pitfalls that end users need to consider.