Enhancing Simple Models by Interpretable Knowledge Transfer

Presenter:

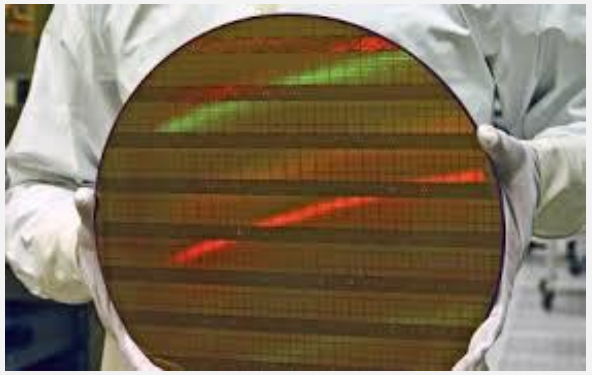
Amit Dhurandhar,
Research Staff Member,
IBM T.J. Watson Research,
Yorktown Heights, New York

Collaborators

Karthikeyan Shanmugam,
Research Staff Member,
IBM T.J. Watson Research,
Yorktown Heights, New York

Ronny Luss,
Research Staff Member,
IBM T.J. Watson Research,
Yorktown Heights, New York

Motivation



Use Case: Fabrication engineers need to measure the amount of metal etched on each manufactured wafer (collection of chips).

Modeling: The engineers are comfortable using decision trees (e.g. CART) to model their problem because they understand the tool and know how to derive insights from the solution.

Challenge: Transfer information from complex models with higher accuracy to the decision trees in a manner that enhances performance and adds valuable insight to the engineers that is easily consumable by them.

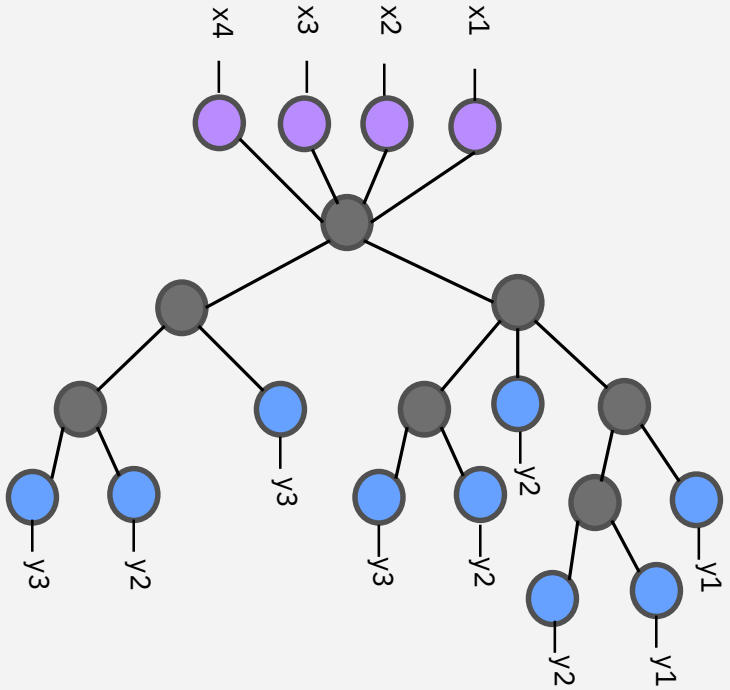
Knowledge Transfer: General Idea

Low performing

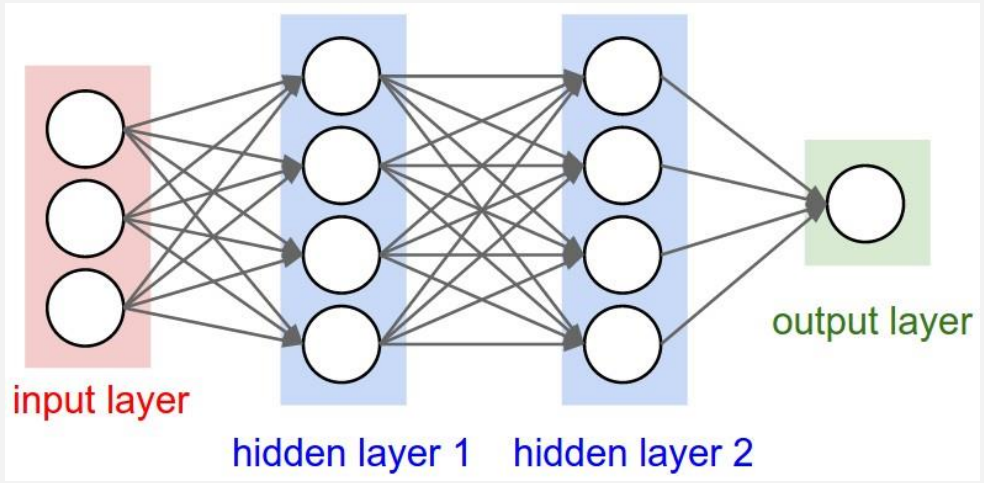
Simple Model
(Decision Tree, SVM, smaller neural network)

Training Data set

Complex Model
(Deep Neural Network, Boosted Trees)



Can you transfer information from a pre-trained neural network to this simple model ?



Applications

Domain Experts Preference: SMEs many times want to use a model they understand and hence trust. Our methods try to get the most out of these models by significantly enhancing their performance.

Small Data settings: Small client data where complex neural networks might overfit so simple models are preferred. However, possible to improve performance if we have a pretrained network on a large public or private corpora belonging to the same feature space.

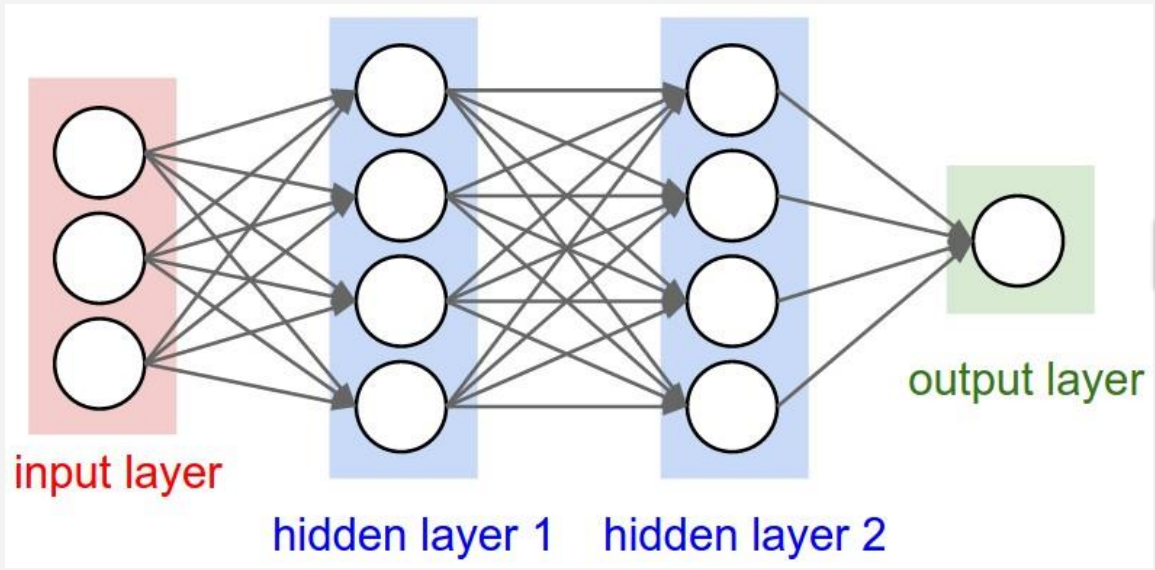
Resource Constrained Settings: In memory and power constrained settings only small models can be deployed. Here improving small neural networks with the help of larger neural network can be useful.

Types of Knowledge Transfer (to Simple Models)

Fit to Black-Box Models Predictions

Knowledge Distillation

- Obtain soft predictions from a complex neural network (for different temperature scalings).
- Use them to regress a simpler neural network with cross-entropy loss.
- Model compression (Bucila et. al, 2006) where you fit to the logits or hard targets is a special case.



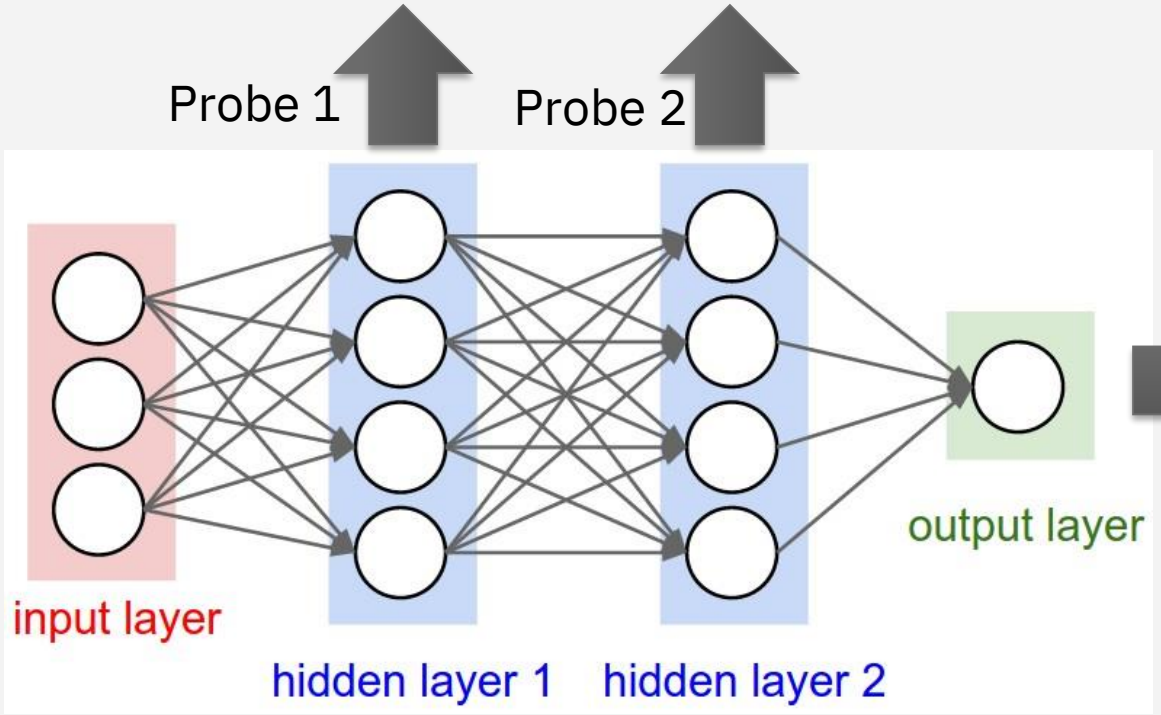
Soft Prediction

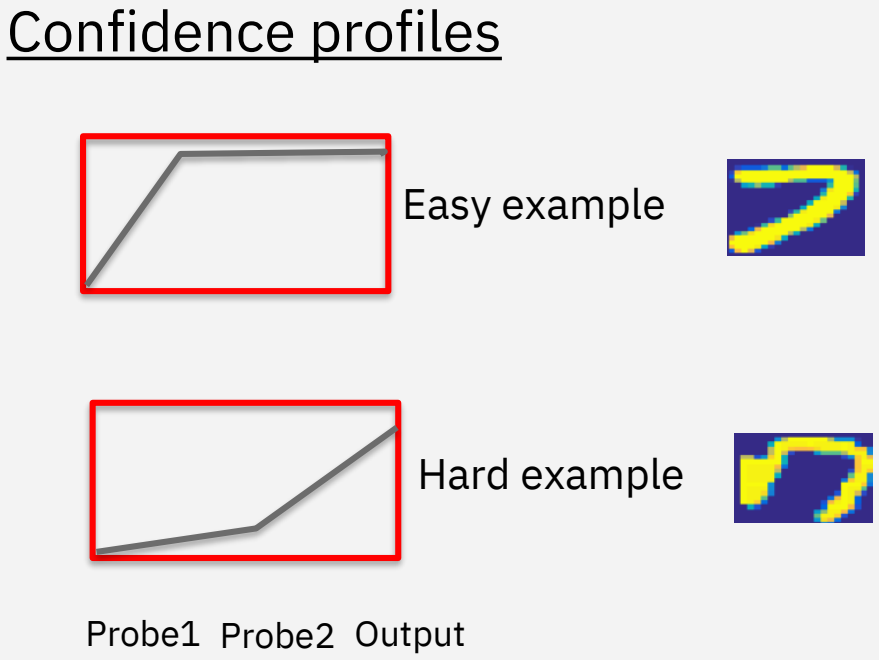
Distilling the Knowledge in a Neural Network. Hinton et. al. NeurIPS 2014
Interpreting Blackbox Models via Model Extraction. Bastani et. al. 2017

Sample Reweighting Methods

ProfWeight

- Find area under curve (AUC) based on confidence scores at each probe whose accuracy > α of the simple model for each example based on complex neural network.
- Weight each training example by corresponding AUC and retrain simple model such as a decision tree.



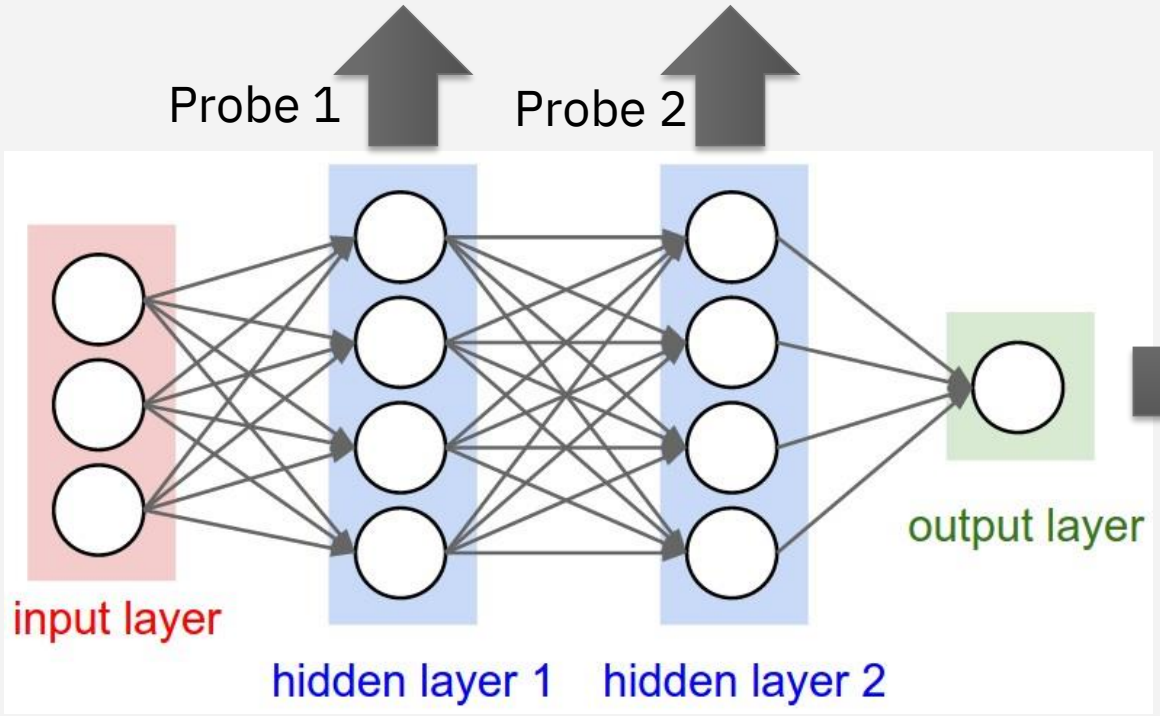


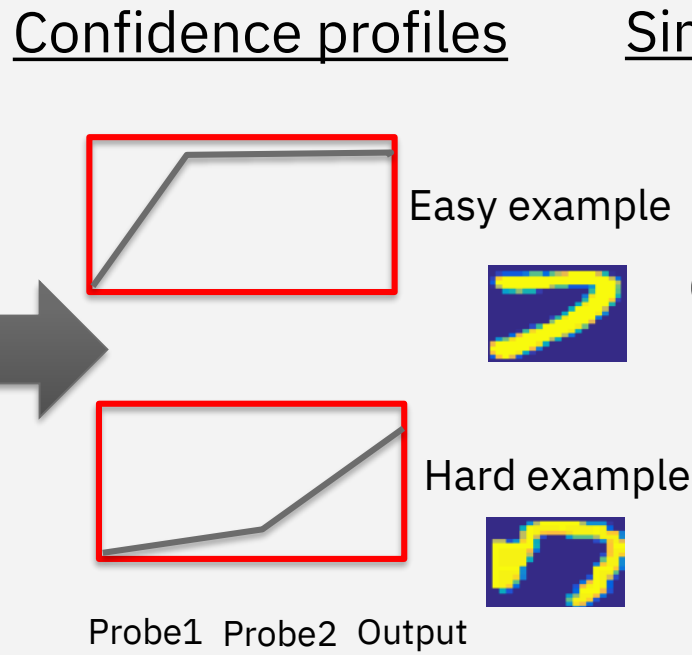
Improving Simple Models with Confidence Profiles. Dhurandhar et. al. NeurIPS 2018

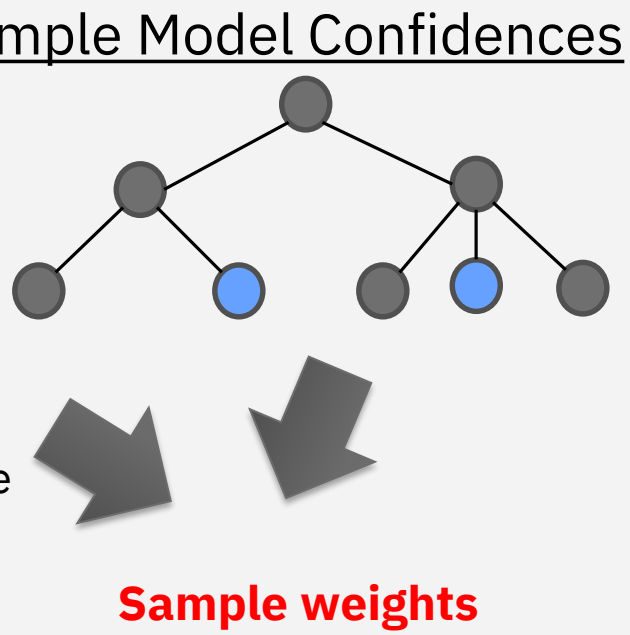
Sample Reweighting Methods

SRatio

- Compute AUC like Profweight (also applicable to other classifiers than NNs).
- Divide by simple models confidence for that sample where the ratio is upper bounded by a tunable parameter (β) which is its weight for retraining the simple model.



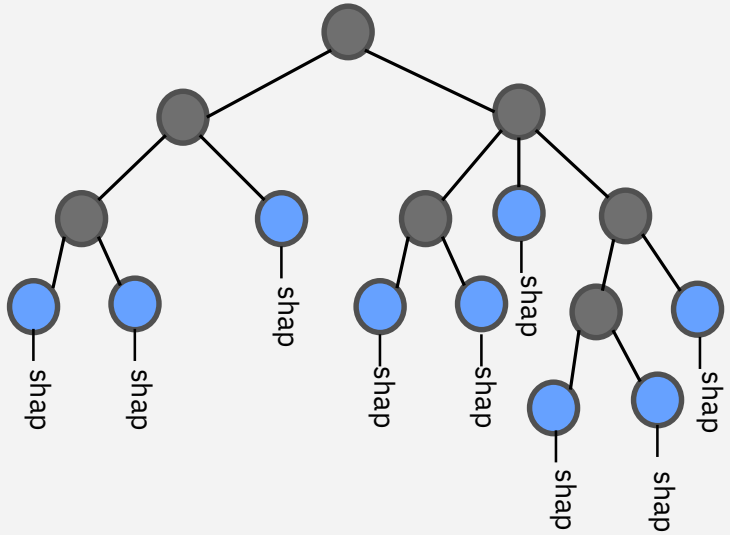




Enhancing Simple Models by Exploiting What They Already Know. Dhurandhar et. al. ICML 2020

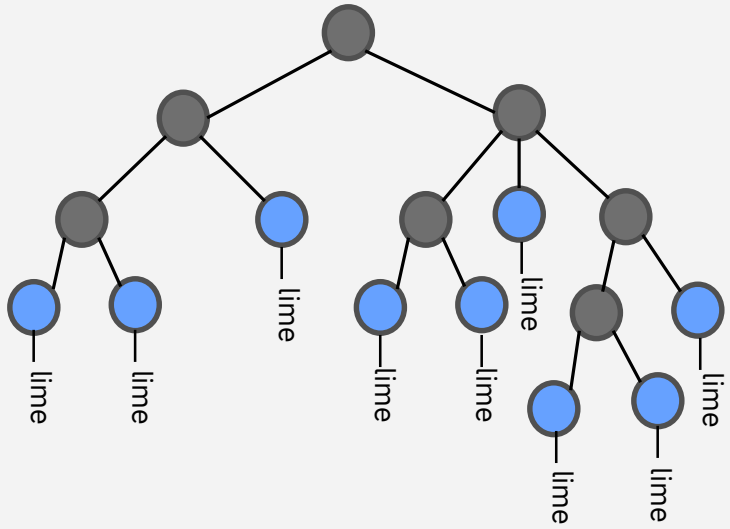
Local Explanations to Global Understanding

TreeExplainer
Combining SHAP based models



From local explanations to global understanding with explainable AI for trees.
Lundberg et. al, Nat. Mach. Intl. 2020

MAME
Jointly learning LIME-like models



Model Agnostic Multilevel Explanations.
Ramamurthy et. al, NeurIPS 2020

GBFL
Learning from Contrastive Explanations

Salary =< 60K

AND

Debt >= 10K

Loan Denied

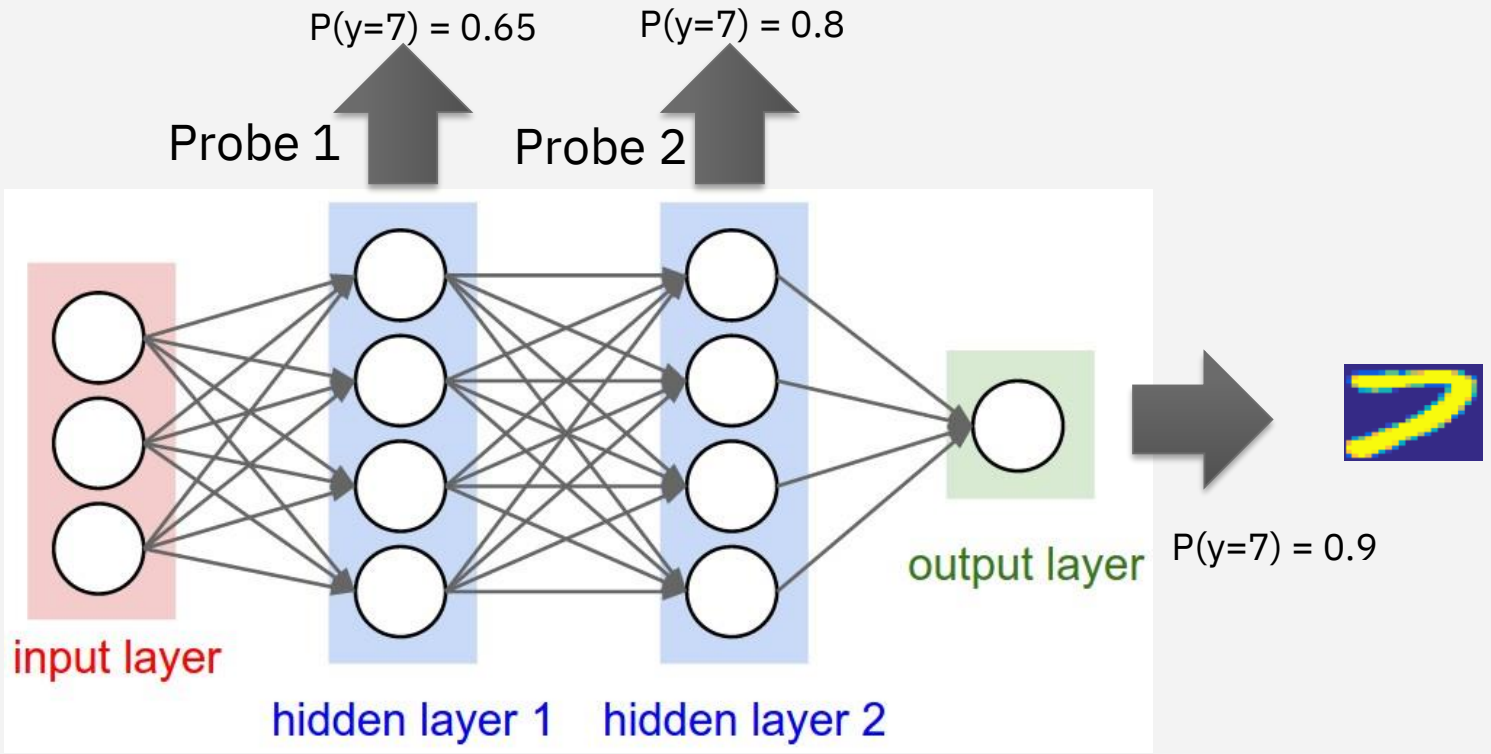
Learning Global Transparent Models Consistent with Local Contrastive Explanations.
Pedapati et. al, NeurIPS 2020

Sample Reweighting Methods

Sample Reweighting Methods

What are Probes?

- Probes are logistic classifiers i.e. a linear model with bias followed by softmax.
- We train them on intermediate representations of a neural network (NN) keeping the weights of the NN fixed.

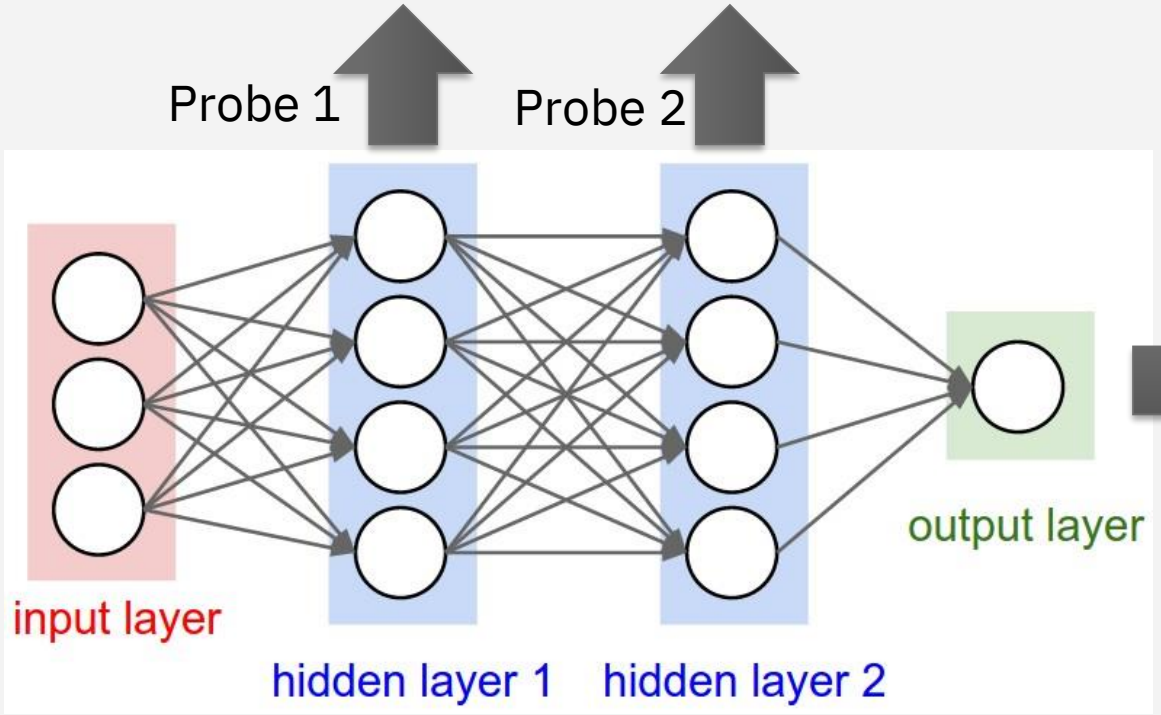


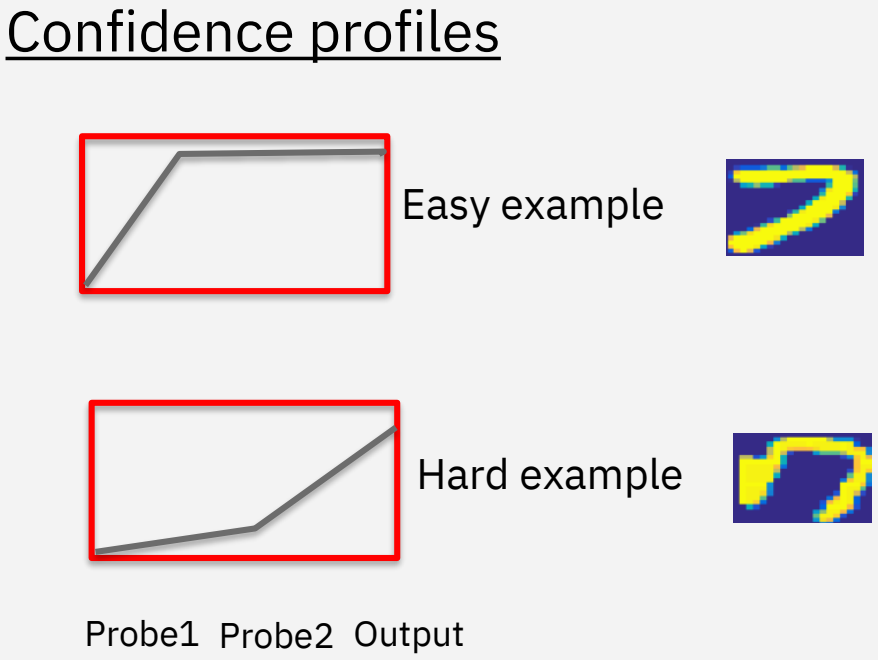
Understanding intermediate layers using linear classifier probes. G. Alain and Y. Bengio, arXiv preprint arXiv:1610.01644, 2016

Sample Reweighting Methods

ProfWeight

- Find area under curve (AUC) based on confidence scores at each probe whose accuracy > α of the simple model for each example based on complex neural network.
- Weight each training example by corresponding AUC and retrain simple model such as a decision tree.





Improving Simple Models with Confidence Profiles. Dhurandhar et. al. NeurIPS 2018
https://www.kdnuggets.com/2020/09/deep-learning-dream-accuracy-interpretability-single-model.html

Algorithm 1 ProfWeight

Input: k unit neural network $\mathcal{N}$, learning algorithm for simple model $\mathcal{L}_{\mathcal{S}}$, dataset D_N used to train $\mathcal{N}$, dataset $D_S = \{(x_1, y_1), ..., (x_m, y_m)\}$ to train a simple model and margin parameter α.

1) Attach probes $P_1, ..., P_k$ to the k units of $\mathcal{N}$.
2) Train probes based on D_N and obtain errors $e_1, ..., e_k$ on D_S. {There is no backpropagation of gradients here to the hidden units/layers of $\mathcal{N}$.}
3) Train the simple model $\mathcal{S} \leftarrow \mathcal{L}_{\mathcal{S}}(D_S, \beta, \vec{1}_m)$ and obtain its error e_S.{$\mathcal{S}$ is obtained by unweighted training. $\vec{1}_m$ denotes a m dimensional vector of 1s.}
4) Let $I \leftarrow \{u \mid e_u \leq e_S - \alpha\}$ {I contains indices of all probes that are more accurate than the simple model $\mathcal{S}$ by a margin α on D_S.}
5) Compute weights w using Algorithm 2 or 3 for AUC or neural network, respectively.
6) Obtain simple model $\mathcal{S}_{\mathbf{w},\beta} \leftarrow \mathcal{L}_{\mathcal{S},\beta}(D_S, \beta, w)$ {Train the simple model on D_S along with the weights w associated with each input.}
return $\mathcal{S}_{w,\beta}$

Ideally (ProfWeight Version 2),

$$S^* = S_{\mathbf{w}^*,\beta^*} = \min_{\mathbf{w}\in\mathcal{C}} \min_{\beta\in\mathcal{B}} E\left[\lambda\left(S_{\mathbf{w},\beta}(x), y\right)\right]$$

Function (eg. NN) of probes weights

Simple model parameters

1. Minimize loss of simple model (βs) for current weights.
2. For given parameters of the simple model minimize over the weights (w).
3. Repeat steps 1 and 2 until convergence or max iterations.

In practice,

Problem: Weights may all go to zero

Solution: Add regularization $\hat{\mathcal{R}}(\mathbf{w}) = (\frac{1}{m}\sum_{i=1}^{m} w_i - 1)^2$

$$S^* = S_{\mathbf{w}^*,\beta^*} = \min_{\mathbf{w}\in\mathcal{C}} \min_{\beta\in\mathcal{B}} E\left[\lambda\left(S_{\mathbf{w},\beta}(x), y\right)\right]$$

Subject to: $\hat{\mathcal{R}}(\mathbf{w}) = (\frac{1}{m}\sum_{i=1}^{m} w_i - 1)^2$

Theoretical Justification

Theorem 1. $\mathbb{E}_{P_M(\mathbf{x}|y)}[w_{M'}(\mathbf{x}, y)] = 1$ *implies that* $w_{M'}(\mathbf{x}, y) = \frac{P(\mathbf{x}|y)}{P_M(\mathbf{x}|y)}$.

It is well-known that the error of the performance of the best classifier (Bayes optimal) on a distribution of class mixtures is the total variance distance between them. That is:

Lemma 2. *[8] The error of the Bayes optimal classifier trained on a uniform mixture of two class distributions is given by:* $\min_\theta \sum \mathbb{D}[L_\theta(x, y)] = \frac{1}{2} - \frac{1}{2} D_{\mathrm{TV}}(P(\mathbf{x}|y = 1), P(\mathbf{x}|y = 0))$ *where* $L(\cdot)$ *is the* $0, 1$ *loss function and* θ *is parameterization over a class of classifiers that includes the Bayes optimal classifier.* D_{TV} *is the total variation distance between two distributions.* $P(\mathbf{x}|y)$ *are the class distributions in dataset* D.

From Lemma 2 and Theorem 1, it is clear that:

$$\min_{M', \theta \text{ s.t. } \mathbb{E}_{\tilde{D}_{\text{train}}}[w'_M]=1} \mathbb{E}_{\tilde{D}}[w_{M'}(\mathbf{x}, \mathbf{y}) L_\theta(\mathbf{x}, y)] = \frac{1}{2} - \frac{1}{2} D_{\mathrm{TV}}(P(\mathbf{x}|y = 1), P(\mathbf{x}|y = 0)) \quad (2)$$

Experiments

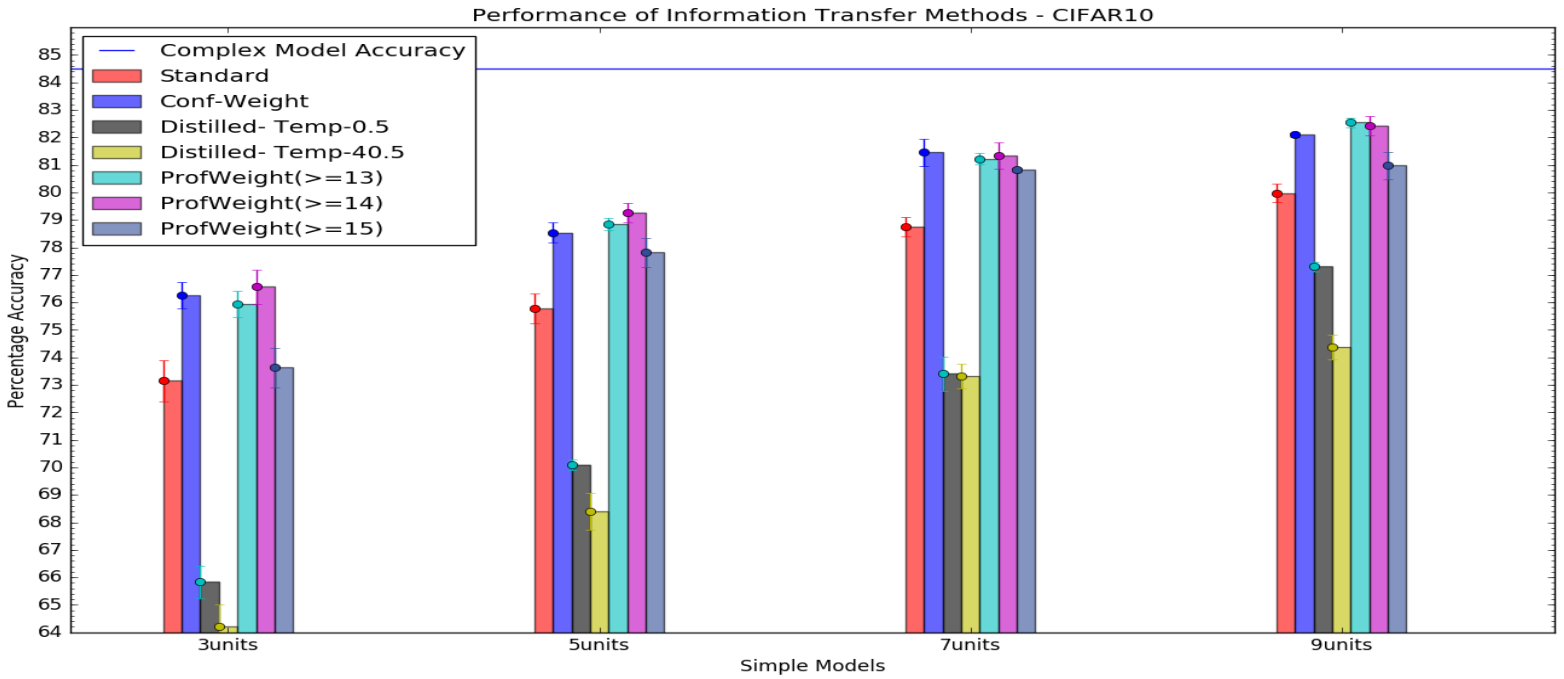
Dataset:
CIFAR-10

Complex Model:
18 Unit Resnet

Simple Models:
3 Unit Resnet
5 Unit Resnet
7 Unit Resnet
9 Unit Resnet

Performance Improvement:
3-4%

	SM-3	SM-5	SM-7	SM-9
Standard	73.15(± 0.7)	75.78(±0.5)	78.76(±0.35)	79.9(±0.34)
ConfWeight	76.27 (±0.48)	78.54 (±0.36)	**81.46**(±0.50)	82.09 (±0.08)
Distillation	65.84(±0.60)	70.09 (±0.19)	73.4(±0.64)	77.30 (±0.16)
ProfWeight^{ReLU}	**77.52** (±0.01)	78.24(±0.01)	80.16(±0.01)	81.65 (±0.01)
ProfWeight^{AUC}	76.56 (±0.62)	**79.25**(±0.36)	**81.34**(±0.49)	**82.42** (±0.36)



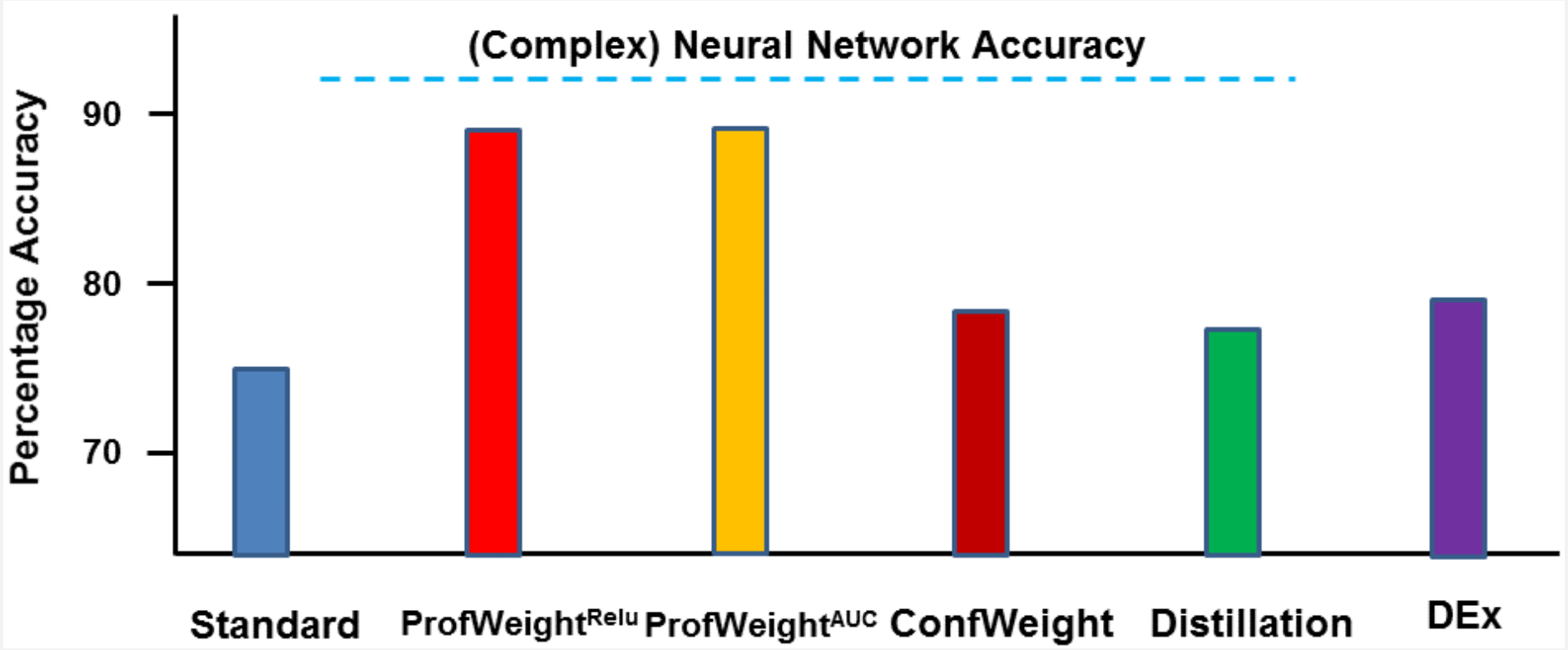
Experiments

Dataset:
Semi-conductor manufacturing

Complex Model:
6 layer Neural Network

Simple Models:
Decision Tree (CART)

Performance Improvement:
13%



Takeaways from ProfWeight

Conceptual Contribution: Upweighting simpler examples (w.r.t. the complex model) helps (somewhat opposite intuition to boosting, but not really …).

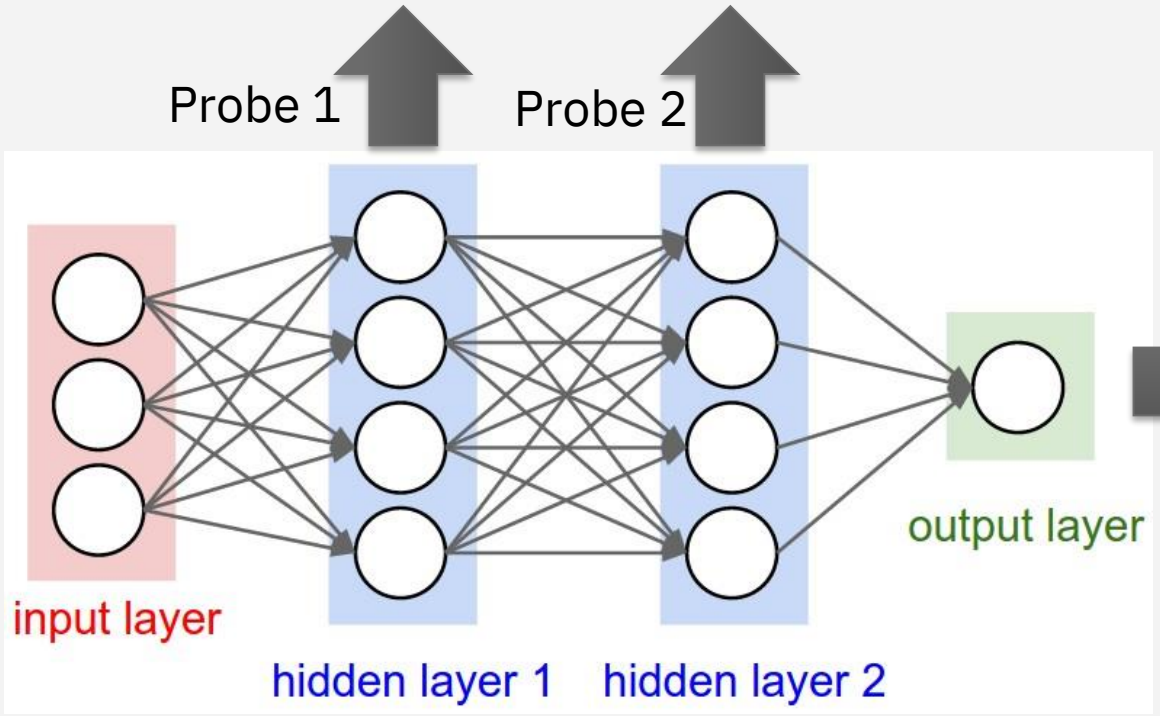
Algorithmic Contribution: Proposed a method, called ProfWeight, based on the above conceptual idea that actually accomplishes improvement using Probes.

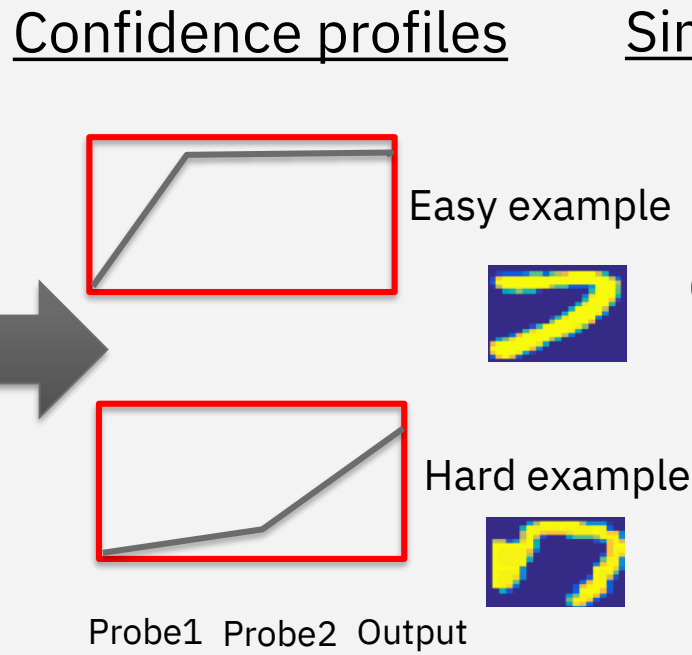
Theoretical Contribution: We show that the alternating optimization scheme has justification (although hard to control in practice).

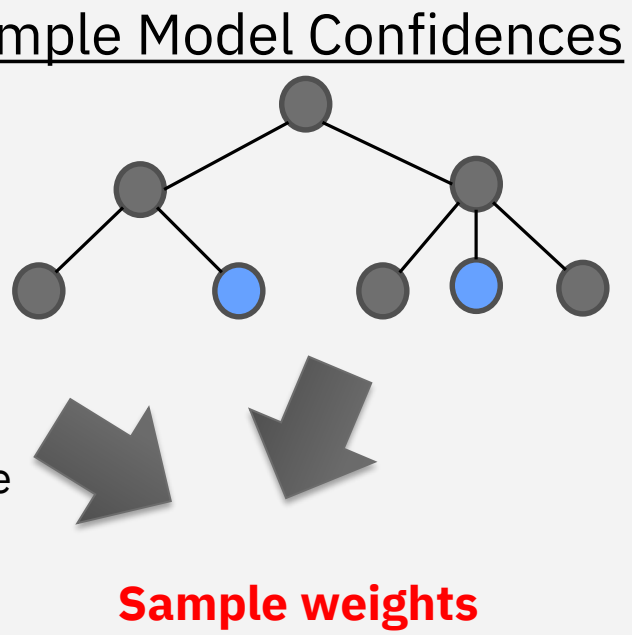
Sample Reweighting Methods

SRatio

- Compute AUC like Profweight (also applicable to other classifiers than NNs).
- Divide by simple models confidence for that sample where the ratio is upper bounded by a tunable parameter (β) which is its weight for retraining the simple model.







Enhancing Simple Models by Exploiting What They Already Know. Dhurandhar et. al. ICML 2020

Intuition for SRatio

Covariate Shift

p is source, q is target

$p(y|x) = q(y|x)$

But $p(x) \neq q(x)$

Typical Soln:
Weight by $q(x)/p(x)$

Classifiers

p is simple, q is complex

$p(x) = q(x)$

But $p(y|x) \neq q(y|x)$

Our (analogous) Soln:
Weight by $q(y|x)/p(y|x)$

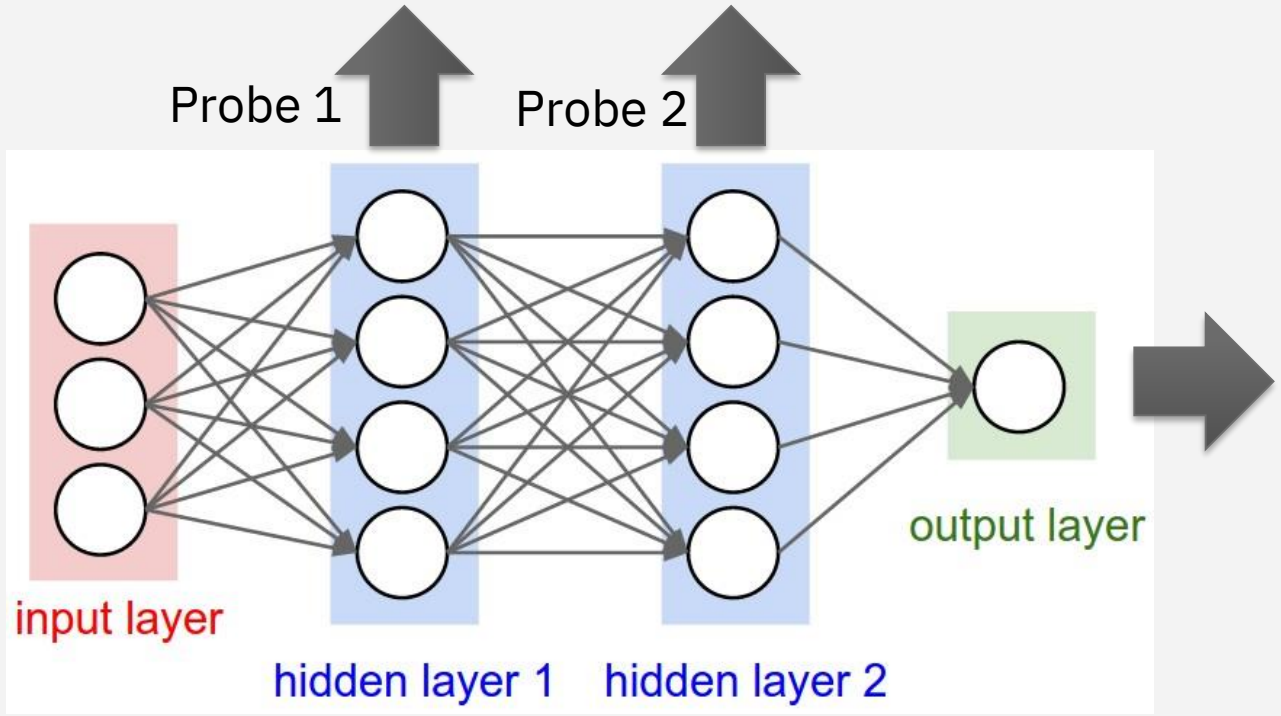
(Inverse covariate shift?)

Theory

Lemma 3.1. *Let $p_\theta(y|x)$ be the softmax scores on a specific model θ from simple model space Θ. Let $\theta^* \in \Theta$ be the set of simple model parameters that is obtained from a given learning algorithm for the simple model on a training dataset. Let $p_c(y|x)$ be a pre-trained complex classifier whose loss is smaller than θ^* on the training distribution. Let $\beta \geq 1$ be a scalar clip level for the ratio $p_c(y|x)/p_{\theta^*}(y|x)$. Then we have:*

$$\mathbb{E}[-\log p_\theta(y|x)] \leq \mathbb{E}\left[\max\left(1, \min\left(\frac{p_c(y|x)}{p_{\theta^*}(y|x)}, \beta\right)\right) \log\left(\frac{1}{p_\theta(y|x)}\right)\right] - \mathbb{E}\left[\log\left(\min\left(\frac{p_c(y|x)}{p_{\theta^*}(y|x)}, \beta\right)\right)\right] + \log(\beta). \quad (1)$$

Probes: To judge hardness of examples in NNs



Improving Simple Models with Confidence Profiles. Dhurandhar et. al. NeurIPS 2018

Generalization of Probes: Graded Classifiers

Definition (δ**-graded**) Let $X \times Y$ denote the input-output space and $p(x, y)$ the joint distribution over this space. Let $\zeta_1, \zeta_2, ..., \zeta_n$ denote classifiers that output the prediction probabilities for a given input $x \in X$ for the most probable (or true) class $y \in Y$ determined by $p(y|x)$. We then say that classifiers $\zeta_1, \zeta_2, ..., \zeta_n$ are δ-graded for some $\delta \in (0, 1]$ and a (measurable) set $Z \subseteq X$ if $\forall x \in Z$, $\zeta_1(x) \leq \zeta_2(x) \leq \cdots \leq \zeta_n(x)$, where $\int_{x \in Z} p(x) \geq \delta$.

Deep Neural Networks: (Linear) probes to intermediate representations.

Boosted Trees: Average predictions of first k trees, first 2k trees, ...

Random Forests: Order trees by accuracy and average predictions of first k trees, first 2k trees, ...

Other Models: Taking increasing order Taylor approximations or do functional decomposition.

SRatio Algorithm

Algorithm 1 Our proposed method SRatio.

Input: n (graded) classifiers $\zeta_1, ..., \zeta_n$, learning algorithm for simple model $\mathcal{L}_\mathcal{S}$, dataset D_S of cardinality N, performance gap parameter γ and maximum allowed ratio parameter β.

1) Train simple model on D_S, $\mathcal{S} \leftarrow \mathcal{L}_\mathcal{S}(D_S, \vec{1}_N)$ and compute its (average) prediction error ϵ_S.{Obtain initial simple model where each input is given a unit weight.}
2) Compute (average) prediction errors $\epsilon_1, ..., \epsilon_n$ for the n graded classifiers and store the ones that are at least γ more accurate than the simple model i.e. $I \leftarrow \{i \in \{1, ..., n\} \mid \epsilon_S - \epsilon_i \geq \gamma\}$
3) Compute weights for all inputs x as follows: $w(x) = \frac{\sum_{i \in I} \zeta_i(x)}{m\mathcal{S}(x)}$, where m is the cardinality of set I and $\mathcal{S}(x)$ is the prediction probability/score for the true class of the simple model.
4) Set $w(x) \leftarrow 0$, if $w(x) > \beta$. {Limit the importance of extremely hard examples for the simple model.}
5) Retrain the simple model on the dataset D_S with the corresponding learned weights $\mathbf{w}$, $\mathcal{S}_w \leftarrow \mathcal{L}_\mathcal{S}(D_S, \mathbf{w})$
6) **Return** $\mathcal{S}_w$

Comparison with other transfer methods

Method	Complex Model	Simple Model	Models Leveraged
Distillation	NN	NN	Complex
ProfWeight	NN	Any	Complex
SRatio	Any	Any	Complex and Simple

Results

Complex Models

- Boosted Trees
- Random Forests
- ResNet 18

Simple Models

- Decision tree
- SVM
- ResNet 3, 5, 7

Table 1. Dataset characteristics, where N denotes dataset size and d is the dimensionality.

Dataset	N	d	# of Classes
Ionosphere	351	34	2
Ovarian Cancer	216	4000	2
Heart Disease	303	13	2
Waveform	5000	40	3
Human Activity	10299	561	6
Musk	6598	166	2
CIFAR-10	60000	32×32	10

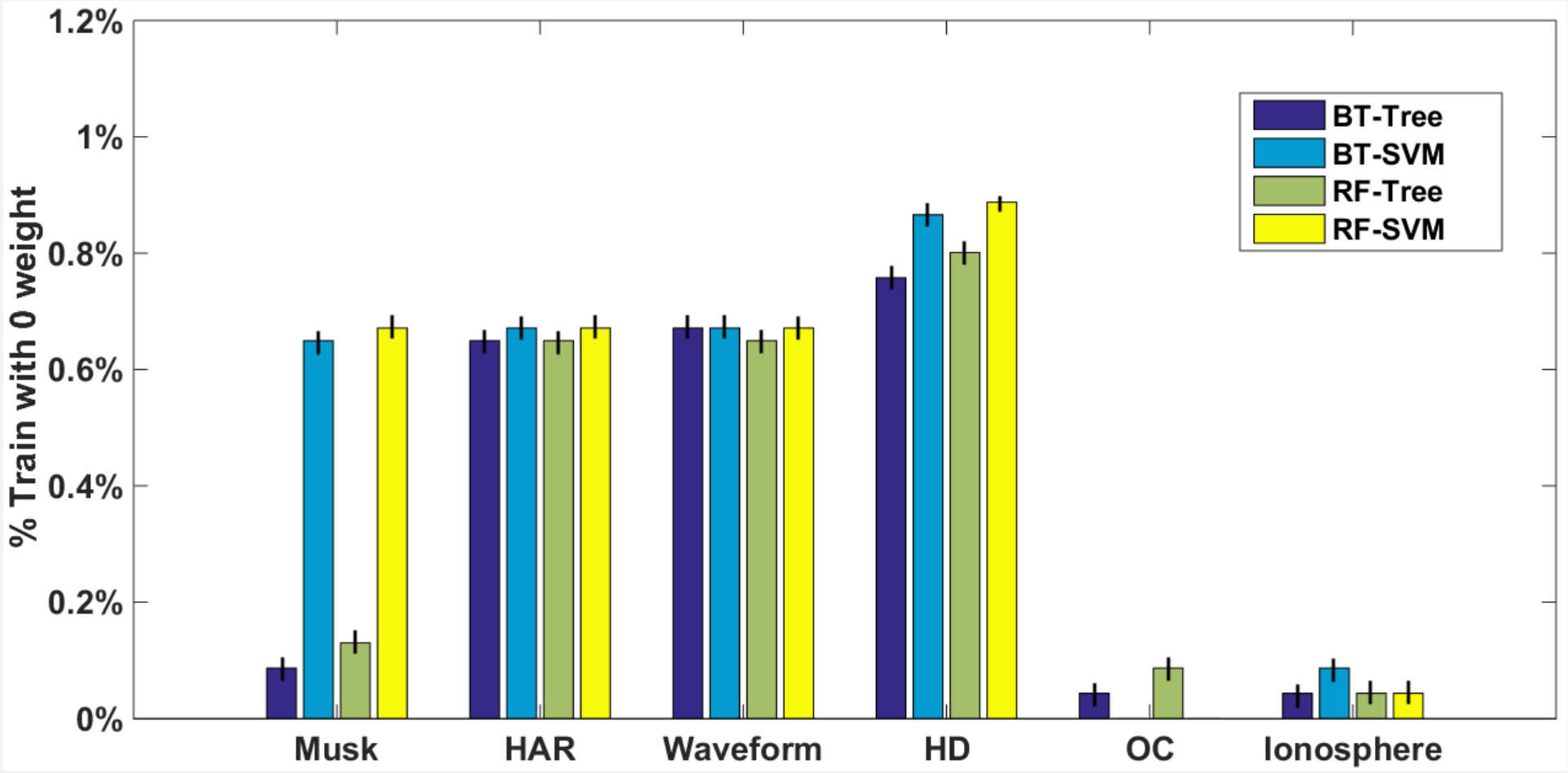
Results

Dataset	Complex Model	CM Error	Simple Model	SM Error	Distill-proxy 1 Error (SM)	ConfWeight Error (SM)	SRatio Error (SM)
Ionosphere	Boosted Trees	8.10 ±0.4	Tree	10.95 ±0.4	10.95 ±0.4	11.42 ±0.8	**8.57* ±0.5**
			SVM	12.38 ±0.6	11.90 ±0.6	11.90 ±0.6	**10.47 ±0.5**
	Random Forest	6.19 ±0.4	Tree	10.95 ±0.4	10.95 ±0.4	11.42 ±0.4	**10.42 ±0.1**
			SVM	12.38 ±0.6	12.38 ±0.6	12.38 ±0.6	**11.42 ±0.3**
Ovarian Cancer	Boosted Trees	4.68 ±0.4	Tree	**15.62 ±0.8**	**15.62 ±0.8**	**15.62 ±1.0**	**15.62 ±0.5**
			SVM	**1.56 ±0.4**	**1.56 ±0.4**	**1.56 ±0.4**	**1.56 ±0.4**
	Random Forest	6.25 ±0.8	Tree	15.62 ±0.8	15.62 ±0.8	**14.06 ±0.1**	**14.04 ±0.1**
			SVM	**1.56 ±0.4**	**1.56 ±0.4**	**1.56 ±0.4**	**1.56 ±0.4**
Heart Disease	Boosted Trees	15.55 ±0.6	Tree	23.88 ±0.7	**22.77 ±0.1**	23.33 ±0.3	**22.77 ±0.2**
			SVM	17.22 ±0.2	**16.67 ±0.3**	17.22 ±0.2	**16.77 ±0.2**
	Random Forest	15.88 ±0.6	Tree	23.88 ±0.7	23.88 ±0.7	25.55 ±0.5	**22.77 ±0.3**
			SVM	17.22 ±0.2	17.22 ±0.2	**16.67 ±0.3**	**16.67 ±0.2**

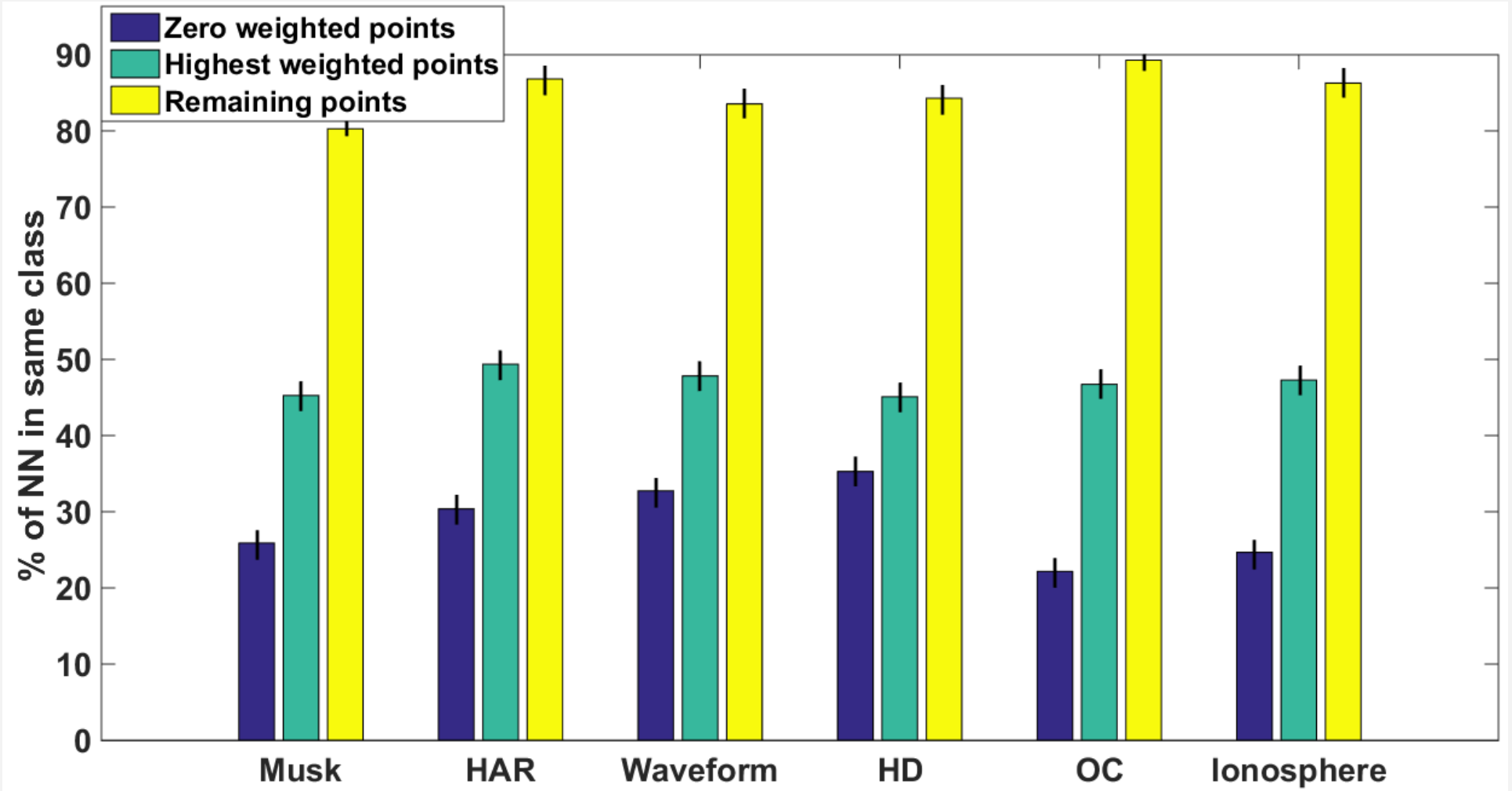
Results

Waveform	Boosted Trees	12.96 ±0.1	Tree	25.43 ±0.2	**25.06 ±0.1**	**25.10 ±0.1**	**25.06 ±0.1**
			SVM	14.70 ±0.2	15.33 ±0.0	14.70 ±0.2	**13.72 ±0.2**
	Random Forest	10.90 ±0.1	Tree	25.43 ±0.2	25.43 ±0.2	25.43 ±0.2	**25.06 ±0.1**
			SVM	14.70 ±0.2	14.33 ±0.0	14.30 ±0.2	**12.72 ±0.5**
Human Activity Recognition	Boosted Trees	6.32 ±0.0	Tree	7.93 ±0.2	7.93 ±0.1	7.86 ±0.2	**7.15 ±0.1**
			SVM	14.56 ±0.1	15.85 ±0.1	**13.92 ±0.1**	**13.92 ±0.2**
	Random Forest	2.34 ±0.0	Tree	7.93 ±0.2	7.23 ±0.1	7.21 ±0.1	**6.67 ±0.0**
			SVM	14.56 ±0.1	**13.92 ±0.1**	14.24 ±0.1	**13.92 ±0.1**
Musk	Boosted Trees	4.06 ±0.1	Tree	4.49 ±0.1	6.11 ±0.1	4.45 ±0.1	**4.06* ±0.1**
			SVM	6.11 ±0.1	6.29 ±0.1	6.41 ±0.1	**5.48 ±0.1**
	Random Forest	2.45 ±0.1	Tree	4.49 ±0.1	4.49 ±0.1	4.47 ±0.1	**3.89 ±0.1**
			SVM	6.11 ±0.1	6.16 ±0.1	5.96 ±0.1	**5.53 ±0.1**

Results


1.2%
1%
0.8%
0.6%
0.4%
0.2%
0%
% Train with 0 weight
BT-Tree
BT-SVM
RF-Tree
RF-SVM
Musk
HAR
Waveform
HD
OC
Ionosphere

Results


Zero weighted points
Highest weighted points
Remaining points
% of NN in same class
0
10
20
30
40
50
60
70
80
90
Musk
HAR
Waveform
HD
OC
Ionosphere

Results

Table 3. Below we see the % of training points whose weights based on SRatio have changed by $> 1\%$ compared to just considering the complex model, i.e., ignoring simple model confidences in Step 3 of algorithm 1. This depicts the impact of considering the simple models confidences in the weighting, which is our main conceptual contribution. Results are averaged over both complex models.

Ionosphere		OC		HD		Waveform		HAR		Musk	
Tree	SVM	Tree	SVM	Tree	SVM	Tree	SVM	Tree	SVM	Tree	SVM
37.40	90.65	8.55	6.57	92.02	99.06	44.9	99.7	5.9	29.45	7.5	13.93
($\pm$ 2.4)	($\pm$ 1.1)	($\pm$ 0.2)	($\pm$ 1)	($\pm$ 1.8)	($\pm$ 0.1)	($\pm$ 2.4)	($\pm$ 0.1)	($\pm$ 0.9)	($\pm$ 1.3)	($\pm$ 0.5)	($\pm$ 0.7)

Results on CIFAR-10

Table 4. Below we observe the averaged accuracies (%) of simple models SM-3 (3 Res units), SM-5 (5 Res units) and SM-7 (7 Res units) trained with various weighting methods and distillation. The complex model achieved 84.5% accuracy. Statistically significant best results are indicated in bold.

	SM-3	SM-5	SM-7
Standard	73.15 (± 0.7)	75.78 (±0.5)	78.76 (±0.35)
ConfWeight	76.27 (±0.48)	78.54 (±0.36)	**81.46** (±0.50)
Distillation	65.84 (±0.60)	70.09 (±0.19)	73.4 (±0.64)
ProfWeight	76.56 (±0.51)	79.25 (±0.36)	**81.34** (±0.49)
SRatio	**77.23** (±0.14)	**80.14** (±0.22)	**81.89** (±0.28)

Takeaways SRatio

Conceptual Contribution: Simple models can be useful to improve themselves (without increasing complexity).

Algorithmic Contribution: Proposed a method, called SRatio, based on the above conceptual idea that actually accomplishes that.

Theoretical Contribution: We show that our weighting scheme is principled as it is a (tight) upper bound on the simple models expected loss.

Formalization: We propose a notion of delta-graded classifiers which is a formalization and generalization of the idea of probes.

Future Work

1. Experiments on more domains and datasets.

2. Try to more deeply understand why the ratio weighting works.

3. Develop other adaptive methods that perform better (viz. multihop methods (blog to come out Oct. end in kdnuggets)).

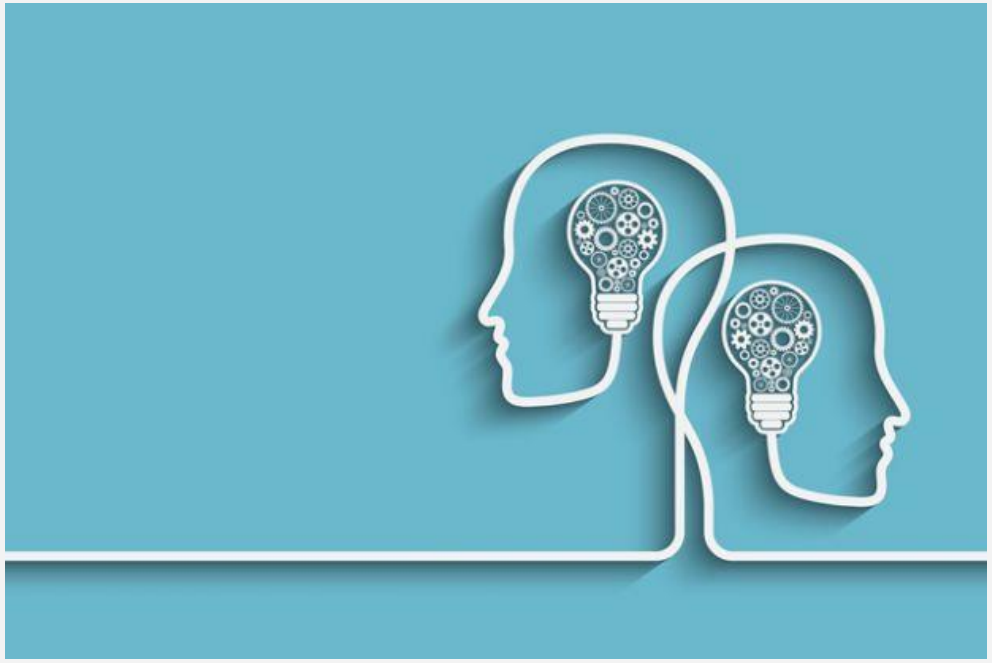
4. Develop methods that are also resilient to hardware faults.

Concluding Remark

Even if you desire just a simple model...

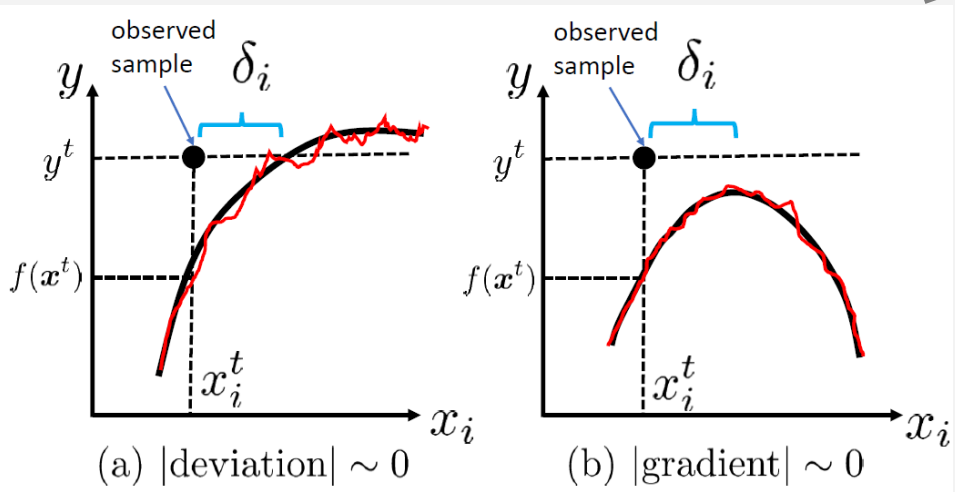
build the most accurate model you can
and
transfer information to the simple model

Thank you

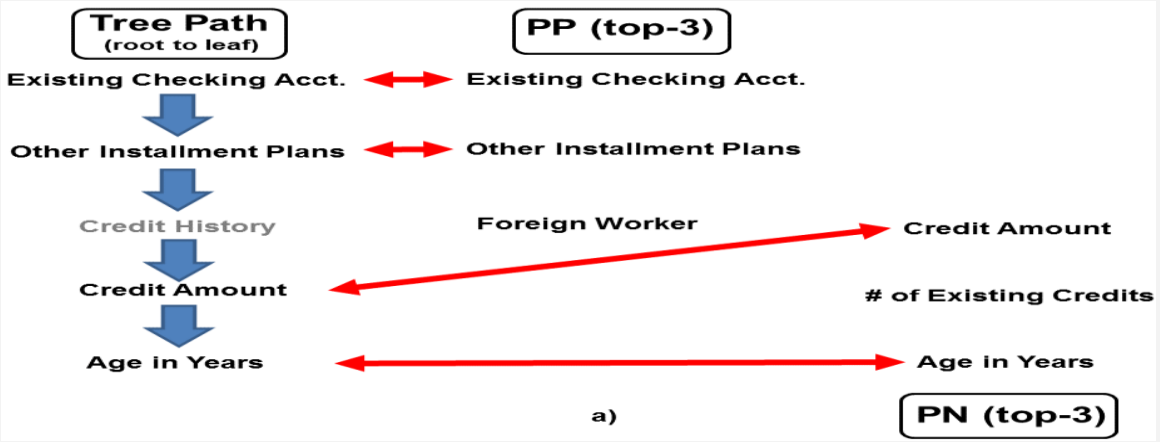


Other Directions: Contrastive Explanations

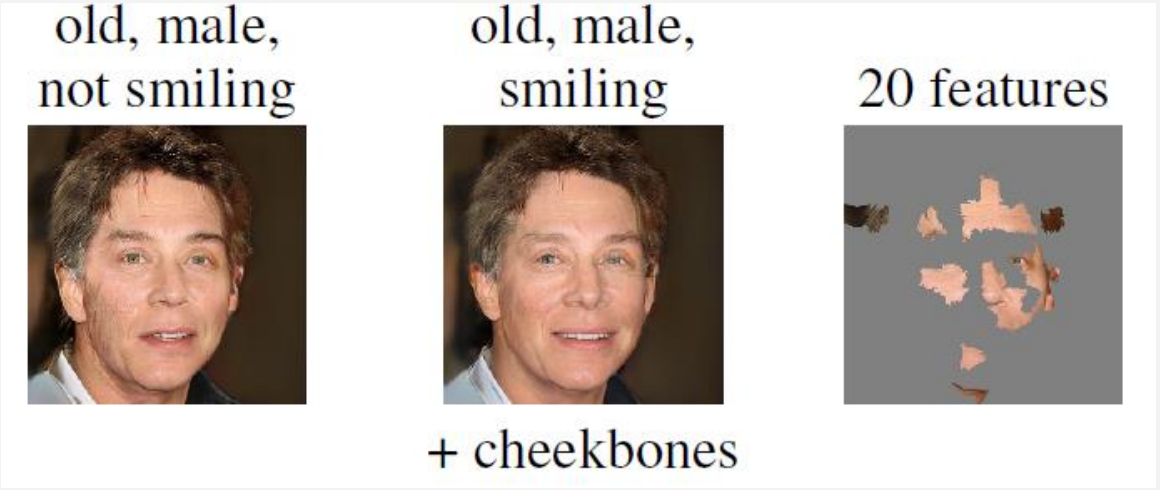
Explanations based on the Missing: Towards Contrastive Explanations with Pertinent Negatives. Dhurandhar et. al. NeurIPS 2018



Anomaly Attribution with Likelihood Compensation. Ide et. al. 2020



Model Agnostic Contrastive Explanations for Structured Data. Dhurandhar et. al. arxiv 2019



Generating Contrastive Explanations with Monotonic Attribute Functions. Russ et. al. arxiv 2019

Other Directions

1. **Meta-Methods Local to Global:** a) From LIME-like techniques can build a tree of explanations and b) From contrastive/counterfactual explanations can generate global Boolean features which can be used to train simple models.

2. **Modeling and Handling Cognitive Biases:** We have done some recent work on modeling cognitive biases and user tested an adaptive time based strategy to debias.

3. **Explaining Similarity Based Classifiers:** Feature based and analogy based (new type) explanations.

4. **Causal Disentanglement:** Uncovering latents with particular causal structures.

5. **Generating Neighborhoods with Business Rules:** For local explainability techniques how do we generate good neighborhoods that represent business rules well.

6. **Interpretable Strategies:** Given a complicated strategy by an agent can we disentangle it into understandable, but accurate sub-goals.

7. **Explaining Representations:** Given an arbitrary representation can I explain it in terms of an understandable representation.