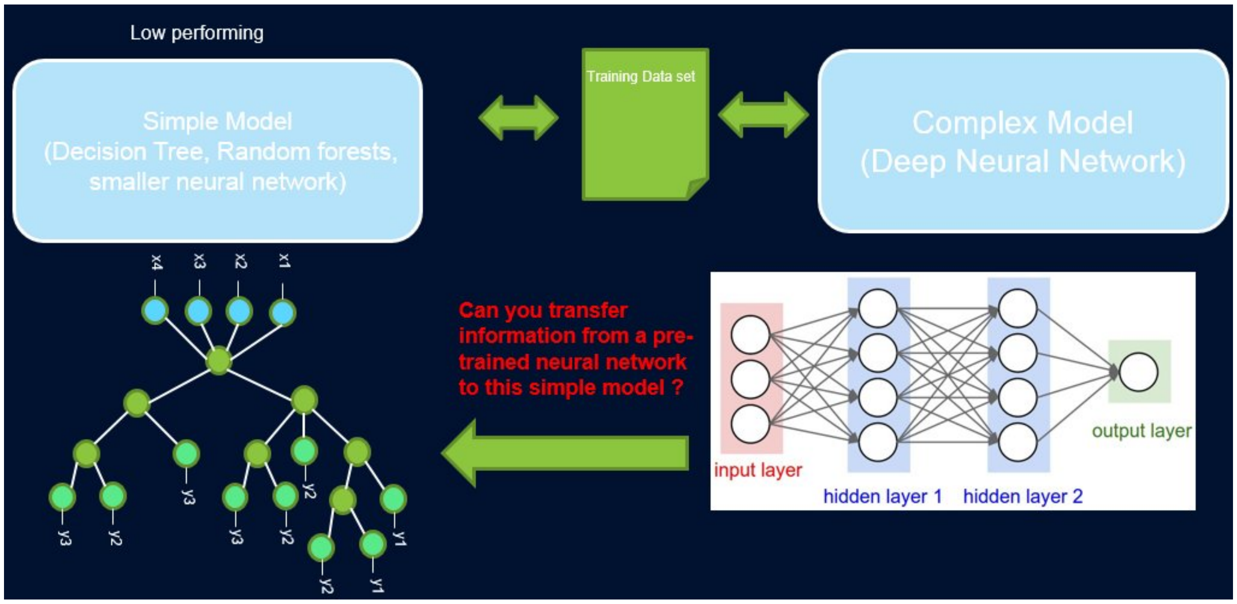
Interpretability and performance: Can the same model achieve both?

December 3 2018, by Amit Dhurandhar


Low performing
Simple Model (Decision Tree, Random forests, smaller neural network)
Training Data set
Complex Model (Deep Neural Network)
Can you transfer information from a pre-trained neural network to this simple model ?
input layer
hidden layer 1
hidden layer 2
output layer
x4
x3
x2
x1
y1
y2
y3

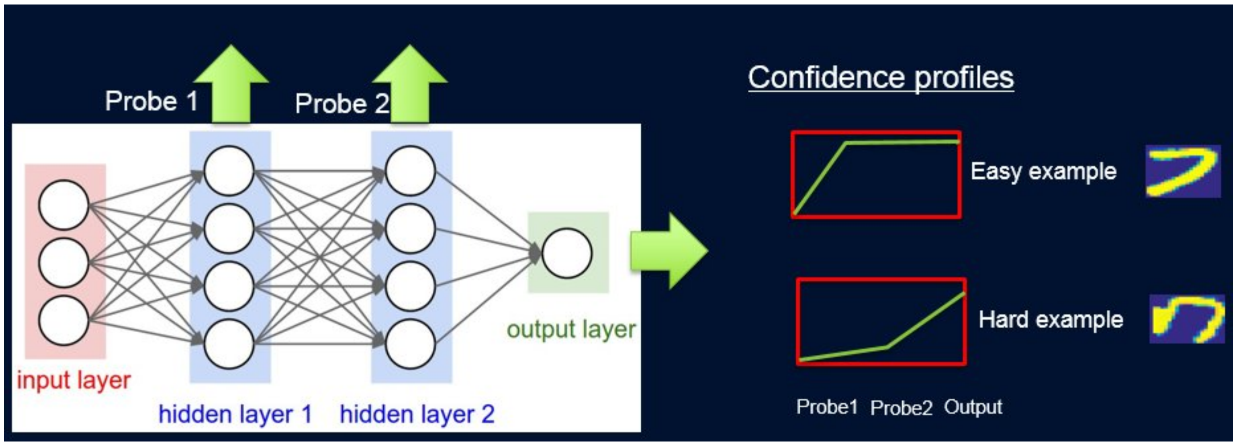
Credit: IBM

Interpretability and performance of a system are usually at odds with each other, as many of the best-performing models (viz. deep neural networks) are black box in nature. In our work, Improving Simple Models with Confidence Profiles, we try to bridge this gap by proposing a method to transfer information from a high-performing neural network to another model that the domain expert or the application may demand. For example, in computational biology and economics, sparse linear models are often preferred, while in complex instrumented domains such as semi-conductor manufacturing, the engineers might prefer using decision trees. Such simpler interpretable models can build trust with the expert and provide useful insight leading to discovery of novel and previously unknown facts. Our goal is pictorially depicted below, for a specific case in which we are trying to improve performance of a decision tree.

The assumption is that our network is a high-performing teacher, and we can use some of its information to teach the simple, interpretable, but generally low-performing student model. Weighting samples by their difficulty can help the simple model in focusing on easier samples that it can successfully model when training, and thus achieve better overall performance. Our setup is different from boosting: in that approach, difficult examples with respect to a previous 'weak' learner are highlighted for subsequent training to create diversity. Here, difficult examples are with respect to an accurate complex model. This means that these labels are near random. Moreover, if a complex model cannot resolve these, there is little hope for the simple model of fixed complexity. Hence, it is important in our setup to highlight easy examples that the simple model can resolve.

To do this, we assign weights to samples according to the difficulty of the network to classify them, and we do this by introducing probes. Each probe takes its input from one of the hidden layers. Each probe has a single fully connected layer with a softmax layer in the size of the network output attached to it. The probe in layer i serves as a classifier that only uses the prefix of the network up to layer i. The assumption is that easy instances will be classified correctly with high confidence even with first layer probes, and so we get confidence levels p_i from all probes for each of the instances. We use all p_i to calculate instance difficulty w_i, e.g. as the area under curve (AUC) of p_i's.

Now we can use the weights to retrain the simple model on the final weighted dataset. We call this pipeline of probing, obtaining confidence weights, and re-training ProfWeight.



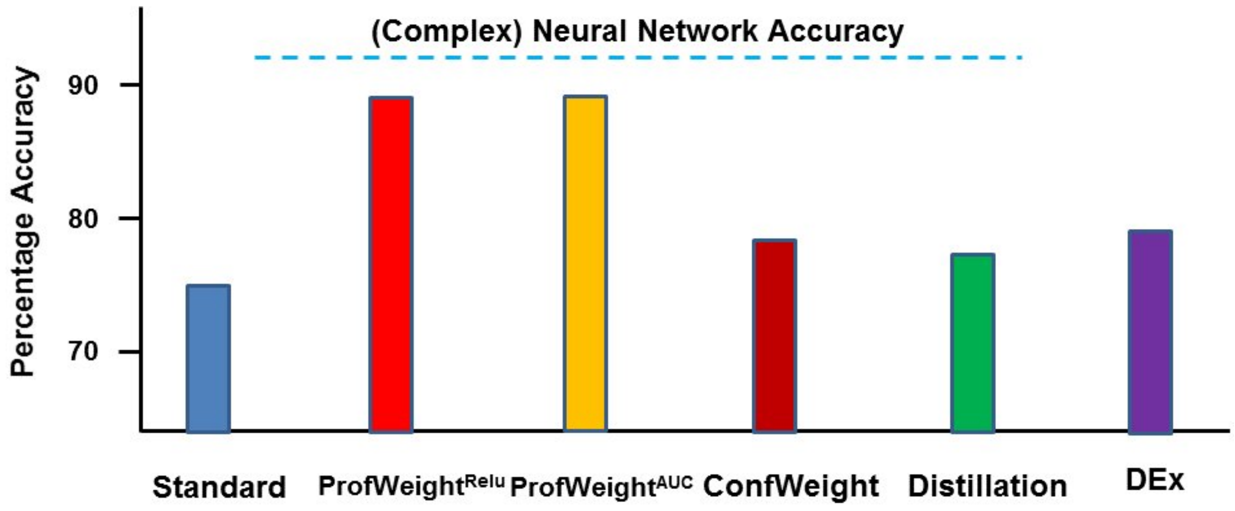
Credit: IBM

We present two alternatives as to how we compute weights for examples in the dataset. In the AUC approach mentioned above, we note the validation error/accuracy of the simple model when trained on the original training set. We select probes that have an accuracy at least α (> 0) greater than the simple model. Each example is weighted based on the average confidence score for the true label that is computed using the individual soft predictions from the probes.

A second alternative involves optimization using a neural network. Here we learn optimal weights for the training set by optimizing the following objective:

$S^*=\min_w \min_\beta E[\lambda(Sw\beta\ (x),y)]$, sub. to. $E[w]=1$

where w are the weights to be found for each instance, β denotes the parameter space of the simple model S, and λ is its loss function. We need to constrain the weights, since otherwise the trivial solution of all the weights going to zero will be optimal for the above objective. We show in the paper that our constraint of E[w]=1 has a connection to finding the optimal importance sampling.



Credit: IBM

More generally ProfWeight can be used to transfer to even simpler but opaque models such as smaller neural networks, which may be useful in domains with severe memory and power constraints. Such constraints are experienced when deploying models on edge devices in IoT systems or on mobile devices or on unmanned aerial vehicles.

We tested our method on two domains: a public image dataset CIFAR-10 and a proprietary manufacturing dataset. On the first dataset, our simple models were smaller neural networks that would comply to strict memory and power constraints and where we saw 3-4 percent improvement. On the second dataset, our simple model was a decision tree and we significantly improved it by ~13 percent, which led to actionable results by the engineer. Below we depict ProfWeight in comparison with the other methods on this dataset. We observe here that we outperform the other methods by quite some margin.

In the future we would like to find necessary/sufficient conditions when transfer by our strategy would result in improving simple models. We would also like to develop more sophisticated methods for information transfer than what we have already accomplished.

We will present this work in a paper titled "Improving Simple Models with Confidence Profiles" at the 2018 Conference on Neural Information Processing Systems, on Wednesday, December 5, during the evening poster session from 5:00 – 7:00 pm in Room 210 & 230 AB (#90).

This story is republished courtesy of IBM Research. Read the original story here.

Provided by IBM

Citation: Interpretability and performance: Can the same model achieve both? (2018, December 3) retrieved 3 May 2024 from
https://phys.org/news/2018-12-interpretability-and-performance-can-the.html