May 29th, 2019

Paper Sharing

Dinghuai Zhang

All Neural Networks are Created Equal

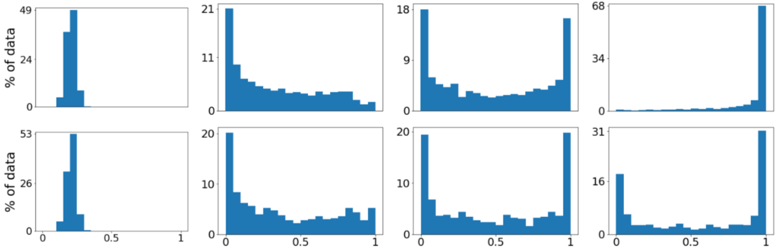
Many networks at epoch e: $f_1^e, \ldots, f_N^e$
Define consistency score of an example

$$c^e(\boldsymbol{x}, y) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}_{[f_i^e(\boldsymbol{x}) = y]}$$

Define consensus score of an example

$$s^e(\boldsymbol{x}, y) = \max_{k \in [\boldsymbol{K}]} \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}_{[f_i^*(\boldsymbol{x}) = k]}$$

Learning dynamics: 4 phases



(a) Initialization (b) Beginning of learning (c) Bi-modality (d) End of learning

Figure 1: The distribution of consistency scores in the 4 phases of learning, training st-VGG on the small-mammals dataset. Top: train data. Bottom: test data.

Figure 1: Consistency score

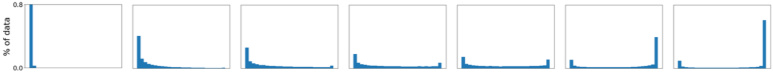
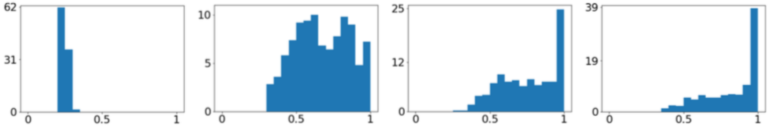


Figure 2: Consistency score



(a) Initialization (b) Beginning of learning (c) During learning (d) End of learning

Figure 3: The distribution of consensus scores during the 4 phases of learning, in corresponding epochs as in Fig. 1, for st-VGG trained on the small-mammals dataset.

Figure 3: Consensus score

Architectures Diversity

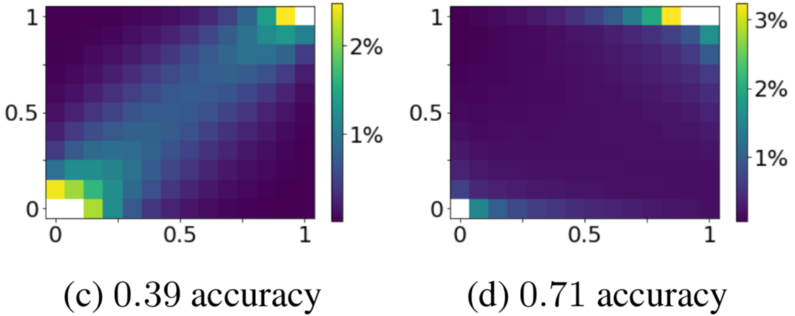
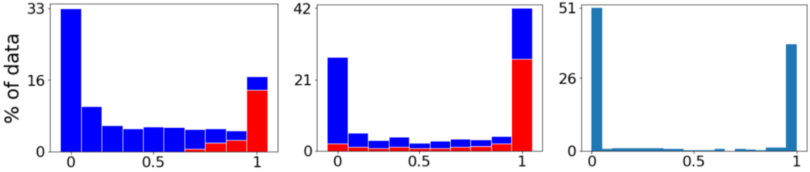


Figure 4: Consistency score for ResNet50 & AlexNet, same acc

Linear NN



(a) Beginning of learning (b) End of learning (c) Linear network consistency

Figure 5: Linear networks. (a-b) Comparing linear to non-linear networks. In blue, the distribution of consistency score for st-VGG trained on the small-mammals dataset. In red, the distribution of a subset of the examples which are classified correctly by a linear version of st-VGG at the given epoch: (a) epoch 1, (b) epoch 140. (c) Distribution of consistency scores for linear st-VGG trained on the small-mammals dataset.

Figure 5

Compared to Other Learning Paradigms

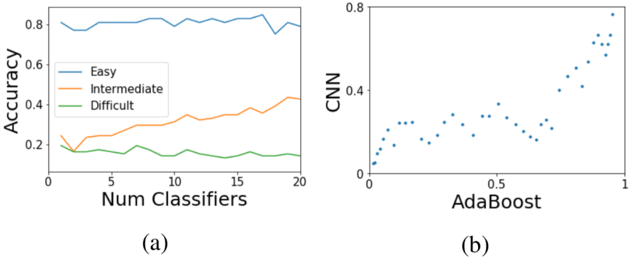


Figure 6: (a) Adaboost accuracy as a function of the number of classifiers, for easy, intermediate, and hard examples as grouped by the consistency score of a CNN. (b) Correlation between the measured difficulty based on Adaboost (X-axis) and CNN (Y-axis), with $r = 0.83,\ p \leq 10^{-10}$.

Disappear of Learning Phases

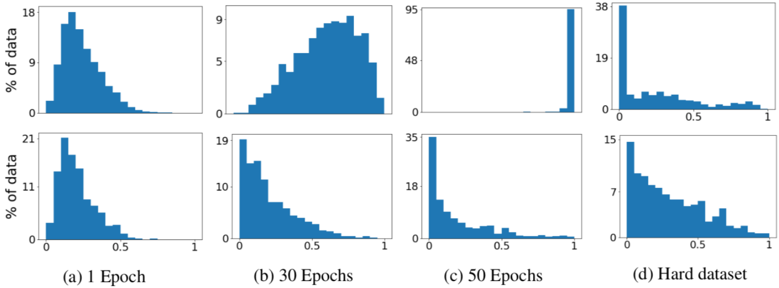


Figure 8: Distribution of consistency scores during the learning process using: (a)-(c) st-VGG trained on a randomized small-mammals dataset; top - train data, bottom - test data. (d) st-VGG trained on an artificially generated hard dataset; top - beginning of training, bottom - end of training.

Figure 6: Random labels & Hard dataset

Bias Also Matters: Bias Attribution for Deep Neural Network Explanation

- Popular network:

$$
\begin{aligned}
x_\ell &= \psi_{\ell-1}\left(W_{\ell-1}x_{\ell-1} + b_{\ell-1}\right) \\
&= \psi_{\ell-1}\left(W_{\ell-1}\psi_{\ell-2}\left(\ldots\psi_1\left(W_1x + b_1\right)\ldots\right) + b_{\ell-1}\right)
\end{aligned}
$$

- Use piecewise linear nonlinear activation functions (such as ReLU & PReLU)
- As a result, the whole network is piecewise linear
- $f(x) = \frac{\partial f(x)}{\partial x}x + b^x$ (at each x)

Bias Also Matters: Bias Attribution for Deep Neural Network Explanation

Then for each feature map x_ℓ

$$f(x) = \left(\prod_{i=\ell}^{m} W_i^x \right) x_\ell + \left(\sum_{j=\ell+1}^{m} \prod_{i=j}^{m} W_i^x b_{j-1}^x + b_m \right)$$

On the other hand

$$f(x) = \sum_{p=1}^{d_\ell} \left[\left(\prod_{i=\ell}^{m} W_i^x \right) [p] \cdot x_\ell[p] + \beta_\ell[p] \right]$$

Want to study properties of β_ℓ

Figure 3: MNIST digit flip test: boxplots of increase in log-odds scores of target vs. source class after the features removed. “Integrated grads-n” refers to the integrated gradient method with n step approximations. "ba1, ba2 and ba3" refer to our 3 options of bias attribution.

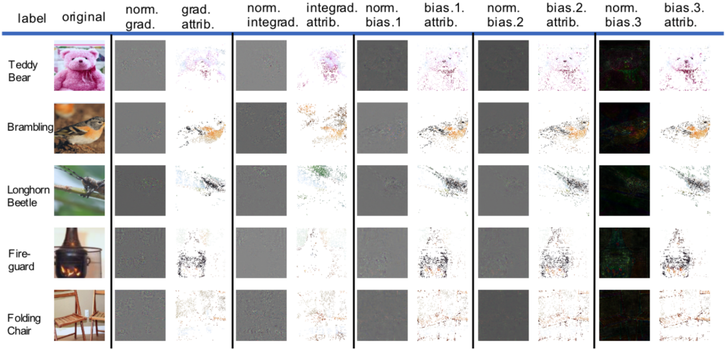


Figure 7: Bias attribution on the ImageNet

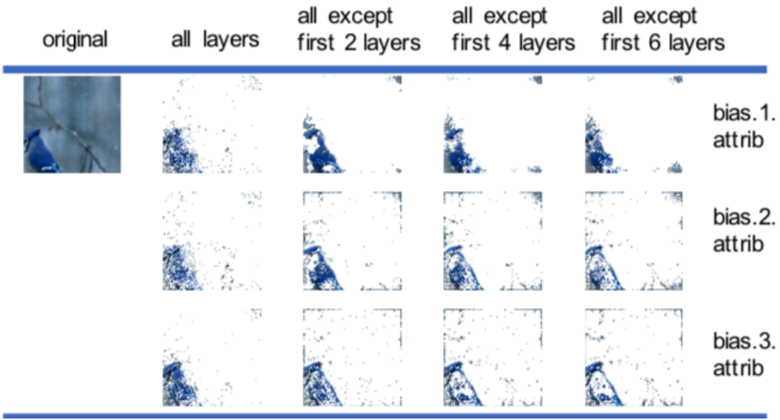


Figure 8: Bias attribution on different layers

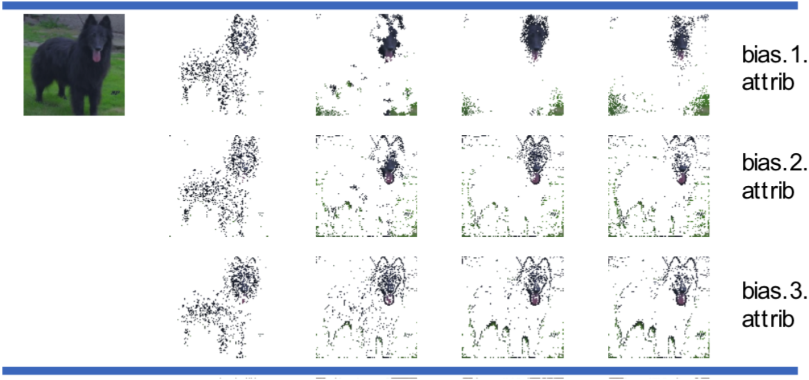


Figure 9: Bias attribution on different layers

Jumpout : Improved Dropout for Deep Neural Networks with ReLUs

Traditional view on the success of dropout

- Prevents the co-adaptation of the neurons
- Train a large number of smaller networks, and during test, the network prediction can be treated as an ensembling

This paper proposed several improvements.

Improvement 1

Encourage the use of smaller dropout rate

- sample $p \sim \mathcal{N}(0, \sigma)$
- truncate through $\min\left(p_{\text{min}} + |p|, p_{\text{max}}\right)$ as dropout rate (instead of treating the rate as a hyperparameter)

Interpretation: Different weights denote different polyhedra. Use a small dropout rate may boost the local smoothness for nearby polyhedra.

Improvement 2

After ReLU, some neurons are set to 0

- The fraction of active neurons is $q^+ = \left(\sum_{i=1:d} \mathbf{1}_{h[i]>0}\right) / |h|$
- The effective dropout rate of every layer is pq^+
- The dropout rate should be adapted as $p' = p/q^+$