Optimization of Target Values in an Artificial Neural Network

Andrassik, Bhuiyan, Yakar, Zauner

Recap: Definitions

- Target values → expected output a neural Network is trained to produce for a specific given input
- Class values → values assigned to the correct class position in the target vector
- Non-class values → values used for all incorrect class positions in the target vector

Bi-directional σ-Adaptation

λ ← 1 (default), tunable in [0, ∞)

Steps:

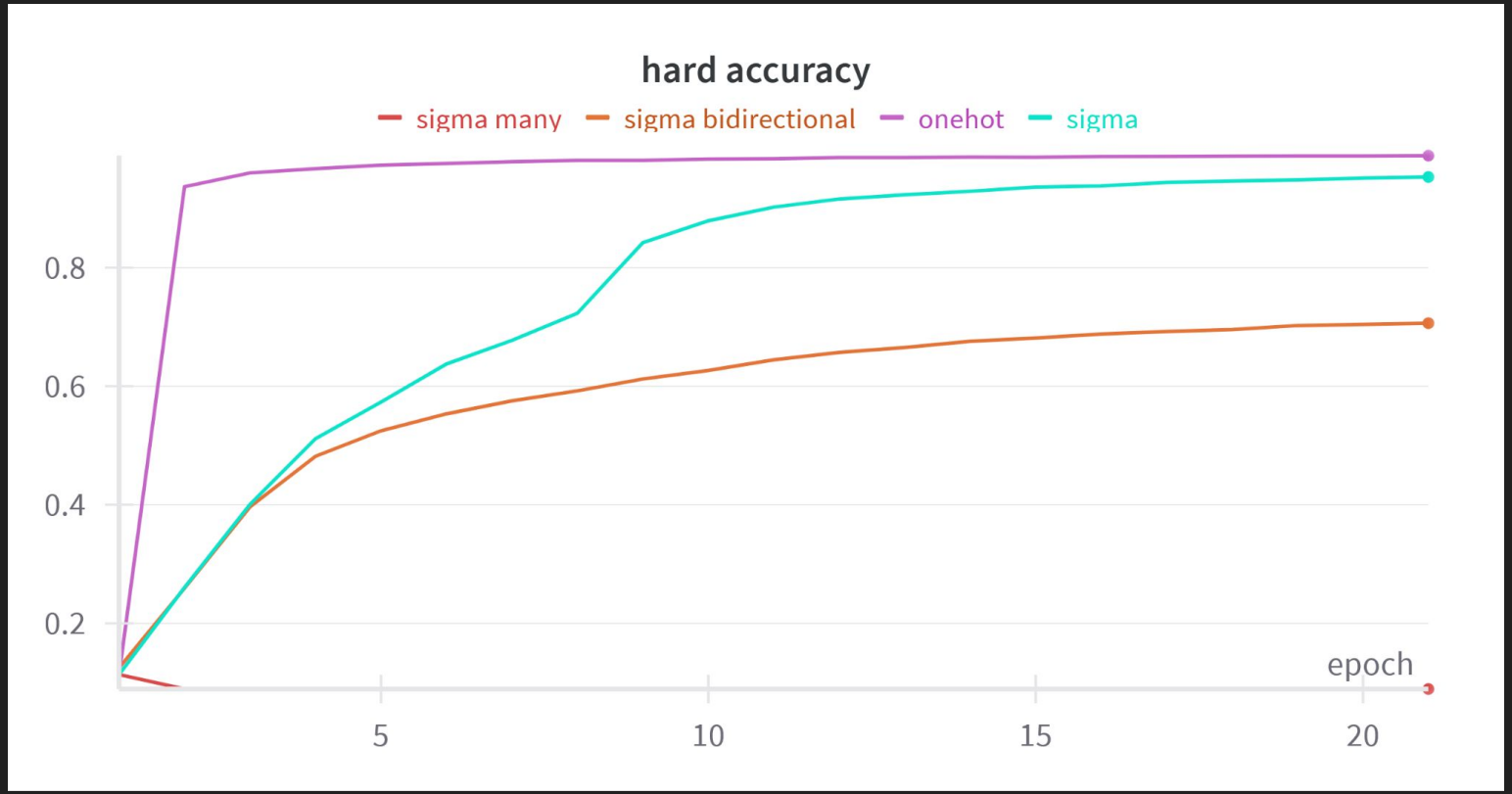
1. Compute the sum of standard deviations for current outputs: σ_{sum}
2. For every class ensure:
   - All values remain within `[0, 1]`
   - `|class - non-class|` ≥ σ_{sum}
3. If too close:
   - Adjust non-class output by $\pm\lambda\sigma_{sum}$ (**increase** if *non-class* ≥ *class*, otherwise **decrease**)
4. If this would exceed bounds clip to 0 or 1

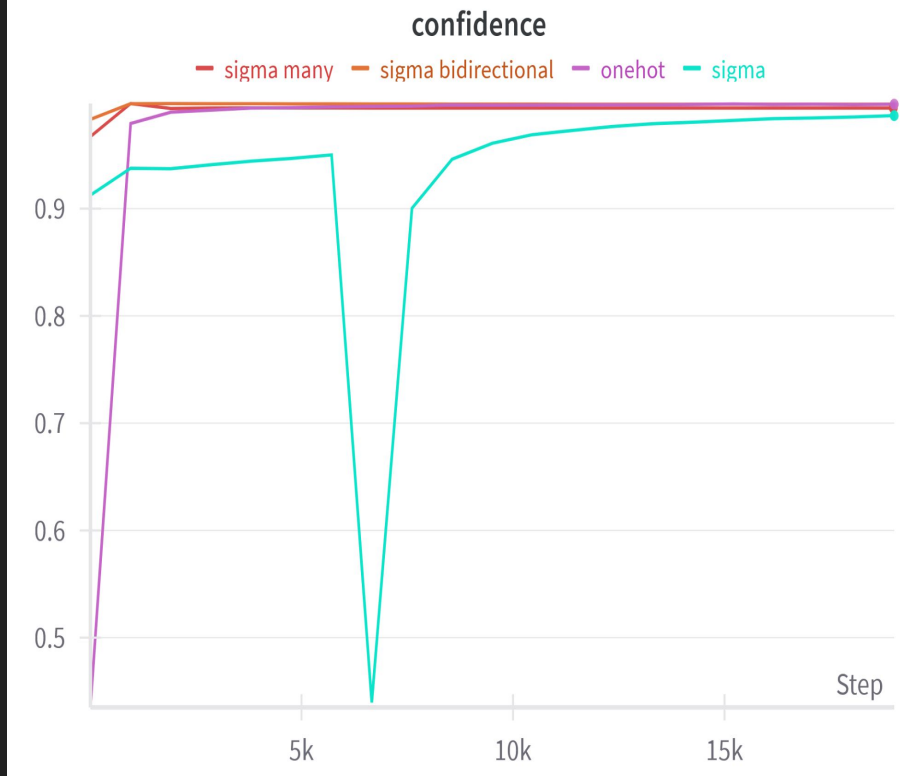
We can define the scaling factor λ separately for the initial pre-training run.

Changes and Challenges

- Each class now has its own non-class value
  - Less values near any non-class value which can cause a shift
    → too few shifts to allow for training
- Switched to Sigmoid from ReLU
- Disable λ to simplify shifting
  - Shift distance is always σ_{sum} → shifts that do happen are too small

Results


hard accuracy
sigma many
sigma bidirectional
onehot
sigma
0.8
0.6
0.4
0.2
epoch
5
10
15
20


confidence
sigma many
sigma bidirectional
onehot
sigma
0.9
0.8
0.7
0.6
0.5
Step
5k
10k
15k


Train Loss and Test Loss
Loss
3
2
1
Epoch
5
10
15
20
sigma many loss
sigma many test loss
sigma bidirectional loss
sigma bidirectional test loss
onehot loss
onehot test loss
sigma loss
sigma test loss

Next Steps

- Create more space between class and non-class values
  - Push apart initially by some amount?
  - Find a better metric than σ_{sum}?

- More readable logging for class/non-class values
  - Visualize all class / non-class value pairs + their σ_{sum} individually
  - Plot all distances in one graph

- Force distance to always be exactly σ_{sum} when pushing (?)