Training Neural Networks with Local Error Signals

Arild Nøkland

Lars H. Eidnes

Local learning

- Typically we train neural networks by backpropagating errors from the loss function and back through the layers.
  - Hard to explain how the brain could do this.
    - Backward locking, weight symmetry, other problems
- Massive practical benefits if you could avoid this.
  - Don't have to keep activations in memory
  - Can parallelize easily. Put each layer on its own GPU, train all at the same time.

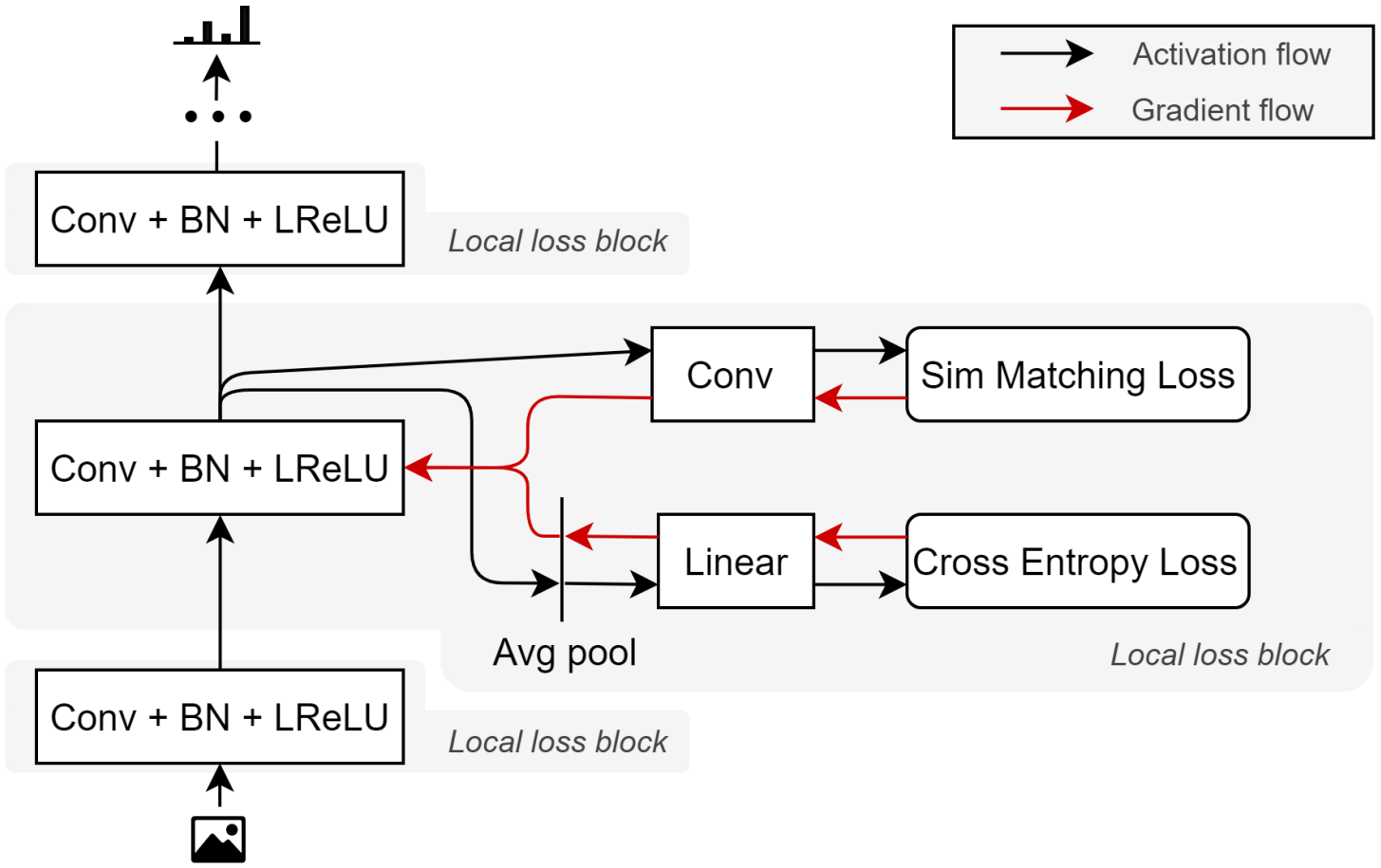
Training each layer on its own works!

Table 4. CIFAR10 with standard data augmentation. Test error in percent.

Model	#par	glob	Local loss functions pred	sim	predsim
3x3000 MLP	27M	33.6	32.3	33.5	**30.9**
VGG8B	8.9M	5.99	8.40	7.16	**5.58**
VGG11B	12M	5.56	8.39	6.70	**5.30**
VGG11B(2x)	42M	4.91	7.30	6.66	**4.42**
VGG11B(3x)	91M	5.02	7.37	9.34[3]	**3.97**
11B(3x)+CO	91M	-	-	-	3.60
WRN	56M	3.87	-	-	-
WRN+CO	56M	3.08	-	-	-

Results on more datasets later.

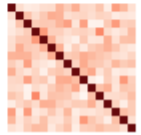
The approach



Train each layer with two sub-networks, each with its own loss function

Similarity matching loss

hidden activations labels

$$L_{sim} = \|S(NeuralNet(H)) - S(Y)\|_F^2 = \text{L2\_loss}(\ \text{  }\ ,\ \text{  }\ )$$

Intuition: Want things from the same class to have similar representations.
Measure similarity with a matrix of cosine similarities.

Results

Table 1. MNIST with 2 pixel jittering. Test error in percent.

Model	#par	glob	Local loss functions pred	sim	predsim
3x1024 MLP	2.9M	0.75	0.68	0.80	**0.62**
VGG8B	7.3M	**0.26**	0.40	0.65	0.31
VGG8B+CO	7.3M	-	-	-	0.26
Ladder	-	0.57	-	-	-
CapsNet	8.2M	0.25	-	-	-

Table 2. Fashion-MNIST with 2 pixel jittering and horizontal flipping. Test error in percent.

Model	#par	glob	Local loss functions pred	sim	predsim
3x1024 MLP	2.9M	**8.37**	8.60	9.70	8.54
VGG8B	7.3M	**4.53**	5.66	5.12	4.65
VGG8B(2x)	28M	4.55	5.11	4.92	**4.33**
8B(2x)+CO	28M	-	-	-	4.14
WRN	37M	4.63	-	-	-
WRN+RE	37M	4.16	-	-	-

Table 3. Kuzushiji-MNIST with no data augmentation. Test error in percent.

Model	#par	glob	Local loss functions pred	sim	predsim
3x1024 MLP	2.9M	**5.99**	7.26	9.80	7.33
VGG8B	7.3M	1.53	2.22	2.19	**1.36**
VGG8B+CO	7.3M	-	-	-	0.99
PARN	11M	2.18	-	-	-
PARN+MM	11M	1.17	-	-	-

Results

Table 4. CIFAR10 with standard data augmentation. Test error in percent.

Model	#par	glob	Local loss functions pred	sim	predsim
3x3000 MLP	27M	33.6	32.3	33.5	**30.9**
VGG8B	8.9M	5.99	8.40	7.16	**5.58**
VGG11B	12M	5.56	8.39	6.70	**5.30**
VGG11B(2x)	42M	4.91	7.30	6.66	**4.42**
VGG11B(3x)	91M	5.02	7.37	9.34[3]	**3.97**
11B(3x)+CO	91M	-	-	-	3.60
WRN	56M	3.87	-	-	-
WRN+CO	56M	3.08	-	-	-

Table 6. CIFAR100 with standard data augmentation. Test error in percent.

Model	#par	glob	Local loss functions pred	sim	predsim
3x3000 MLP	27M	62.6	58.9	62.5	**56.9**
VGG8B	9.0M	26.2	29.3	32.6	**24.1**
VGG11B	12M	25.2	29.6	30.8	**24.1**
VGG11B(2x)	42M	23.4	26.9	28.0	**21.2**
VGG11B(3x)	91M	23.7	25.9	28.0	**20.1**
WRN	56M	18.8	-	-	-
WRN+CO	56M	18.4	-	-	-

Results

Table 8. STL-10 with no data augmentation. Test error in percent.

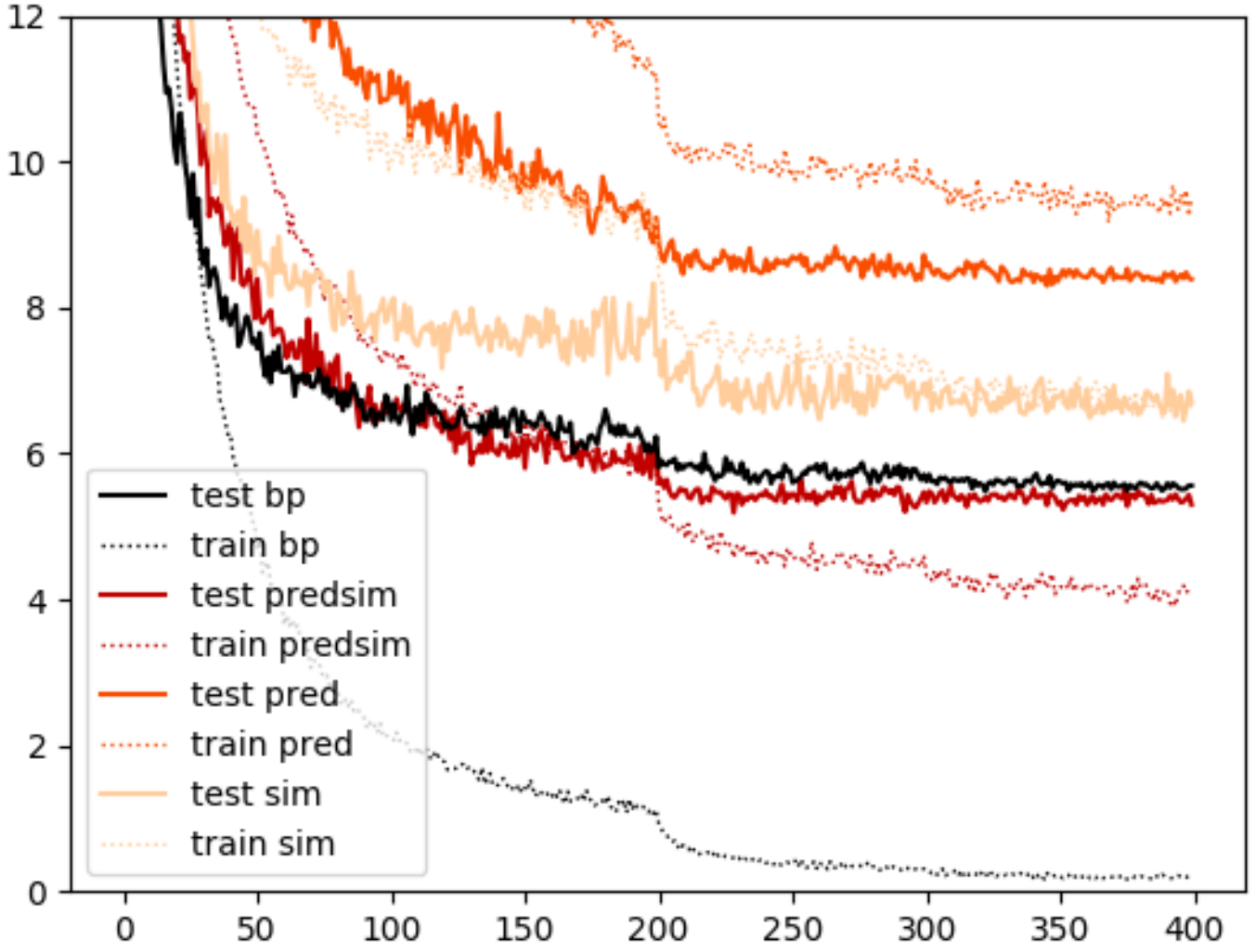
Model	#par	glob	Local loss functions pred	sim	predsim
VGG8B	12M	33.08	26.83	23.15	**20.51**
VGG8B+CO	12M	-	-	-	19.25
WRN	11M	23.48	-	-	-
WRN+CO	11M	20.77	-	-	-

Table 7. SVHN with extra training data, but no data augmentation. Test error in percent.

Model	#par	glob	Local loss functions pred	sim	predsim
VGG8B	8.9M	2.29	2.12	1.89	**1.74**
VGG8B+CO	8.9M	-	-	-	1.65
WRN	11M	1.60	-	-	-
WRN+CO	11M	1.30	-	-	-

Optimization vs generalization

- Back-prop has fastest & lowest drop in training error
- Local learning is competitive with back-prop in terms of test error
- Local learning is a good regularizer
- But: Both **pred** and **sim-** losses help *optimization* in a complementary way.



Sim-loss + global backprop

Table 9. Similarity matching as a complementary objective. Test error in percent.

Dataset	Model	#par	glob	predsim	glob+sim
MNIST	VGG8B	7.3M	0.26	0.31	**0.24**
Fashion-MNIST	VGG8B	7.3M	4.53	4.65	**4.16**
Kuzushiji-MNIST	VGG8B	7.3M	1.53	1.36	**1.13**
CIFAR-10	VGG11B	12M	5.56	5.30	**4.33**
CIFAR-100	VGG11B	12M	25.2	24.1	**22.2**
SVHN	VGG8B	8.9M	2.29	**1.74**	1.95
STL-10	VGG8B	12M	33.1	**20.5**	25.6

Results, back-prop free version

- Still have 1-step backprop. To remove it:
  - Remove the conv2d before the sim-loss
  - Use Feedback Alignment [Lillicrap et al, 2014] through linear before the pred-loss
- Also: Use a random projection of the labels

Table 5. CIFAR10 with standard data augmentation. No back-propagation. Test error in percent.

Model	#par	pred-bpf	sim-bpf	predsim-bpf
VGG8B	8.9M	9.80	13.39	**9.02**
VGG8B(2x)	31M	-	-	7.80

Summary

- We train each layer on its own, without global backprop
- We use two loss functions
  - Standard cross entropy loss
  - A similarity matching loss
    - Squared error on similarity matrices
    - Wants similar activations for things of the same class
- Works well on VGG-like networks

Intriguing questions

- We've just prodded the space of local loss functions, and stumbled across something that helps a lot. Is there more to be found in this space?
- Can we better understand how layers interact when they are trained on their own? I.e. why does this work?
- Does something like this happen in the brain?