Aggregation of Dropout Subnetworks

Final Bachelor Thesis Presentation

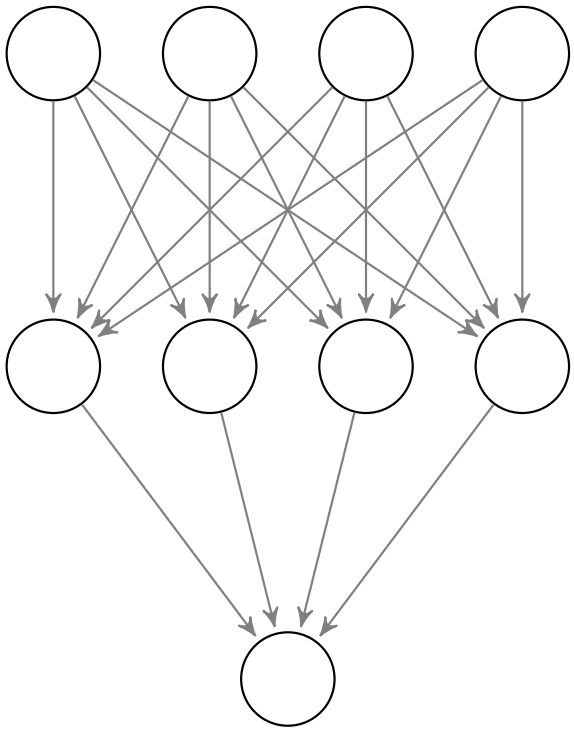
Paul Miller

University of Fribourg

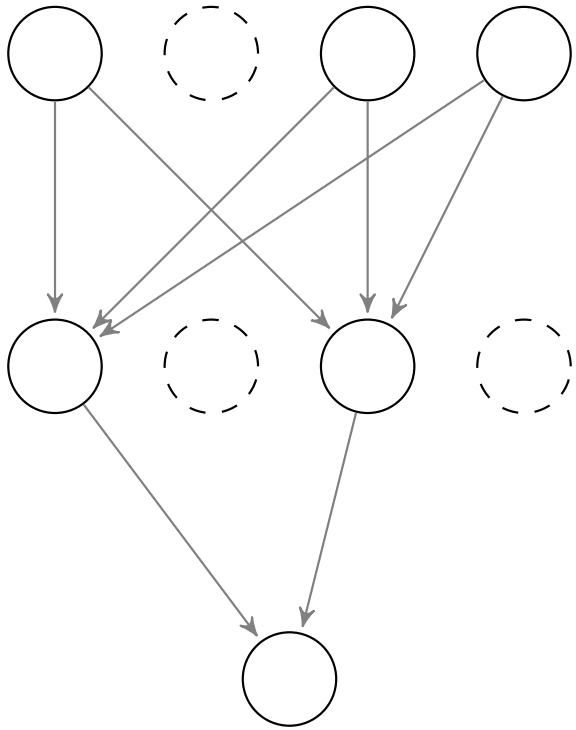
January 31, 2018

Dropout training

Dropout: A Simple Way to Prevent Neural Networks from Overfitting, 2014.

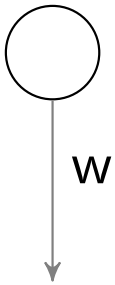


(a) Training without Dropout

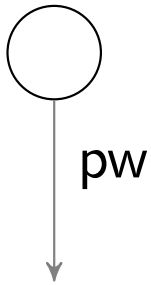


(b) Training with Dropout

Dropout testing



(a) Dropout training phase of a neuron present with probability p

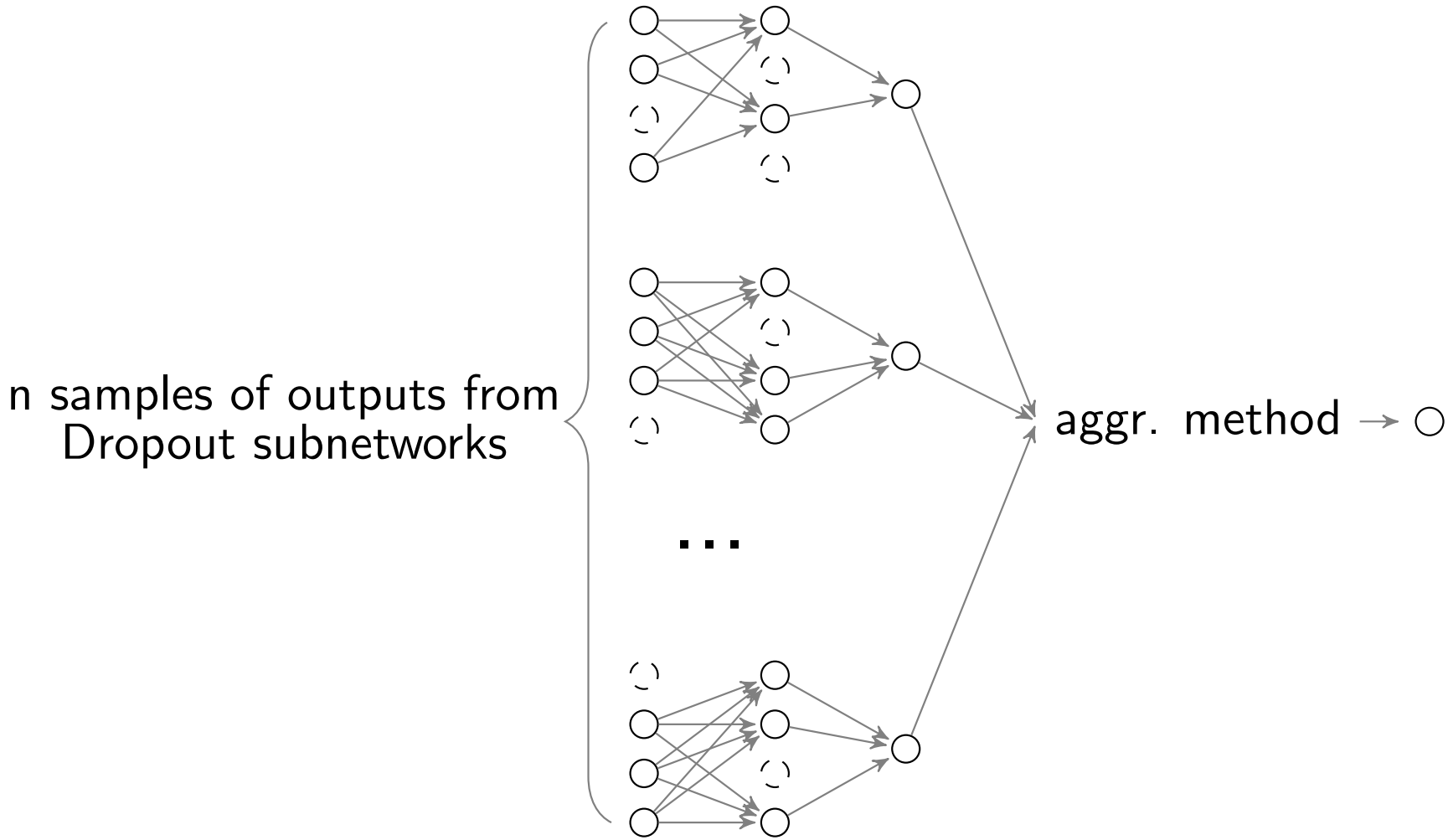


(b) Dropout testing phase

Problem statement

Finding a better way of using Dropout subnetworks at test time

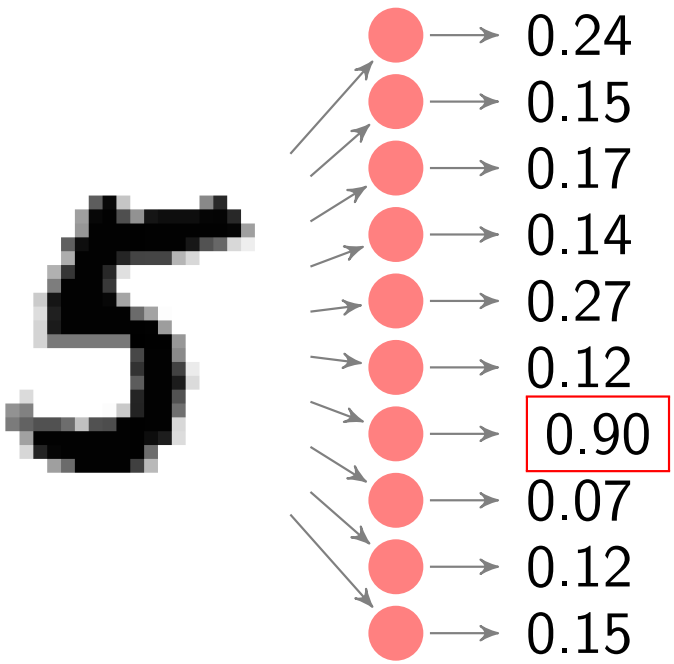
My project



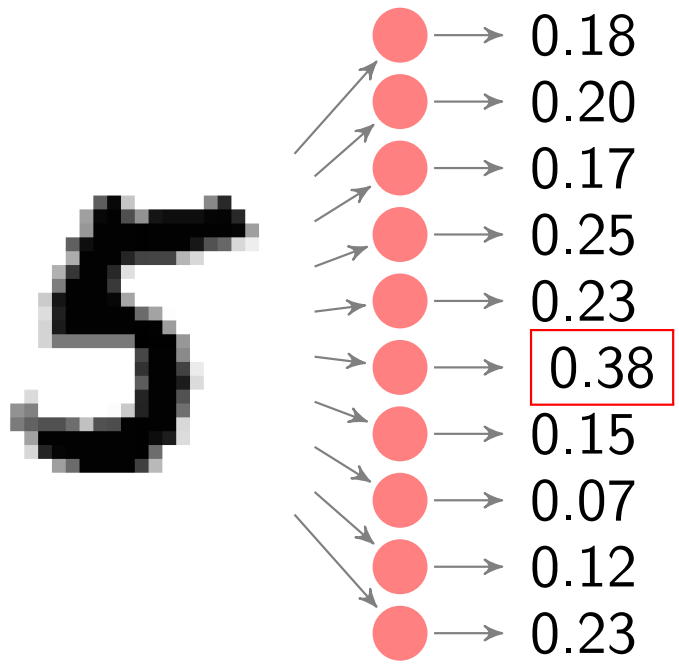
Tools

- DeepDIVA
- AlexNet model for final results
- MNIST: database of handwritten digits
- CIFAR-10: database for object classification

Example of two subnetwork outputs

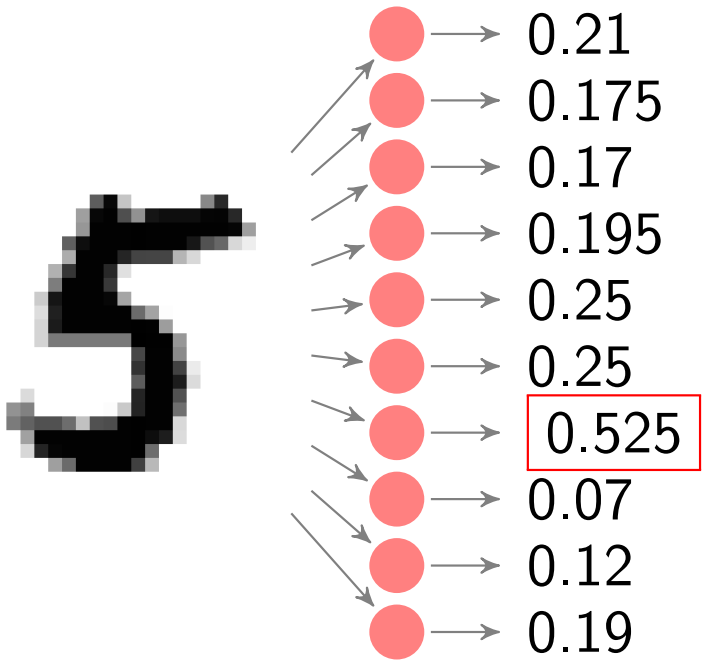


(a) Output with different values



(b) Output with similar values

Average of both outputs



Subnetwork (a) has a lot more effect on the average and its prediction is treated as more valuable

Voting system for subnetworks

Idea: normalize subnetwork outputs by only giving one vote for an output class

Experimental results on CIFAR-10

Accuracy loss/gain on CIFAR-10[1]

nb samples	Dropout	median	average	voting system
50	80.0125%	+0.172%	**+0.2535%**	+0.2185%
100	80.0905%	+0.115%	+0.129%	**+0.1405%**
200	79.9995%	+0.1825%	+0.2025%	**+0.2035%**

[1] AlexNet model, 20 epochs, average of 20 runs

Experimental results on MNIST

Accuracy loss/gain on MNIST[2]

nb samples	Dropout	median	average	voting system
50	**99.439%**	–0.0075%	–0.009%	–0.0135%
100	99.466%	+0.0015%	**+0.009%**	+0.0015%
200	99.4725%	+0%	+0.0025%	**+0.009%**

[2] AlexNet model, 20 epochs, average of 20 runs

Scoring system for subnetworks

Idea: compare the output of each subnetwork with mean output for the predicted class during validation to measure how reliable it is and give it the appropriate weight in the final prediction

Implementation

Consider the matrix $Y = (y_{ij})$ of size NxN, where N is the number of output classes, such as each line $[y_{i1}, ..., y_{iN}]$ contains the mean output of inputs from class $i \in [1, ..., N]$. For every forward pass $k \in [1, ..., K]$, we get an output $x^k = [x_1^k, ..., x_N^k]$, which predicts a class $h \in [1, ..., N]$. We compute:

$$\alpha_k = \sum_{i=1}^{N} (x_i^k - y_{hi})^2$$

$$\omega = \max(\alpha_k)$$

$$z = \sum_{k=1}^{K} x^k \frac{(a\omega - b\alpha_k)}{\omega}$$

with a and b constant parameters.

Experimental results on CIFAR-10

Accuracy loss/gain on CIFAR-10[3]

nb samples	Dropout	median	average	scoring system[4]
10	**79.5805%**	–0.286%	–0.121%	–0.0825%
25	80.1255%	–0.029%	**+0.082%**	+0.0815%
50	80.0295%	+0.092%	+0.159%	**+0.1975%**
100	79.9675%	+0.1465%	+0.195%	**+0.2045%**
200	79.9145%	+0.065%	+0.0795%	**+0.118%**
500	79.8585%	+0.141%	+0.148%	**+0.1935%**

[3] AlexNet model, 20 epochs, average of 20 runs

[4] Using a=1 and b=1

Experimental results on MNIST

Accuracy loss/gain on MNIST[5]

nb samples	Dropout	median	average	scoring system[6]
10	99.472%	–0.019%	–0.01%	**+0.015%**
25	99.474%	–0.0025%	–0.00325%	**+0.0065%**
50	99.4905%	+0.0045%	+0.0025%	**+0.009%**
100	99.467%	–0.007%	–0.001%	**+0.015%**
200	99.464%	+0.006%	+0.0065%	**+0.012%**

[5] AlexNet model, 20 epochs, average of 20 runs

[6] Using a=1 and b=1

Conclusion

Voting system: up to +0.2% accuracy on CIFAR-10 and +0.009% on MNIST using 200 forward passes

Scoring system: up to +0.2% accuracy on CIFAR-10 and +0.015% on MNIST using 100 forward passes

Further work

- Try out different values for a and b
- Calculate the computational time added by both systems
- Different datasets, models and parameters
- Test these methods on DropConnect models

Thank you for your attention!