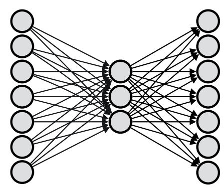
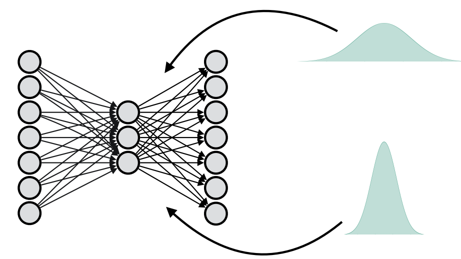
Nadia Chirkova

Bayesian sparsification of neural networks

2021

Slides are partially based on lectures of Dmitry Vetrov, Dmitry Kropotov and Kirill Struminsky, deepbayes.ru/2018

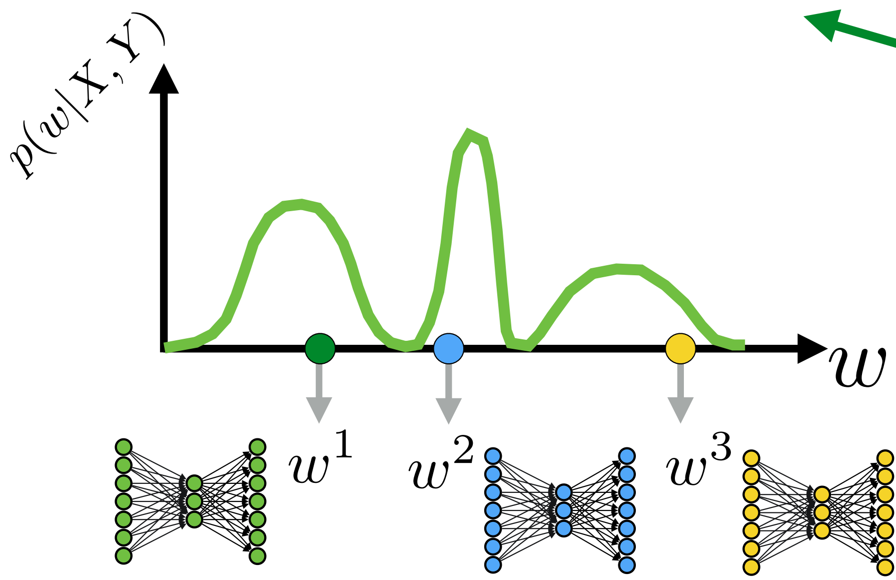
Bayesian neural networks

	Deterministic neural network	Bayesian neural network
Weights:	deterministic weights 	stochastic weights 
Training:	Data X, Y → point estimate w_{MP}	Prior $p(w)$ + Data X, Y → Posterior $p(w \mid X, Y)$
Prediction on new object x_* :	$p(y_* \mid x_*, w_{MP})$	$\mathbb{E}_{p(w \mid X, Y)} p(y_* \mid x_*, w)$

Prediction with Bayesian neural network

Prediction on a new object x_*:

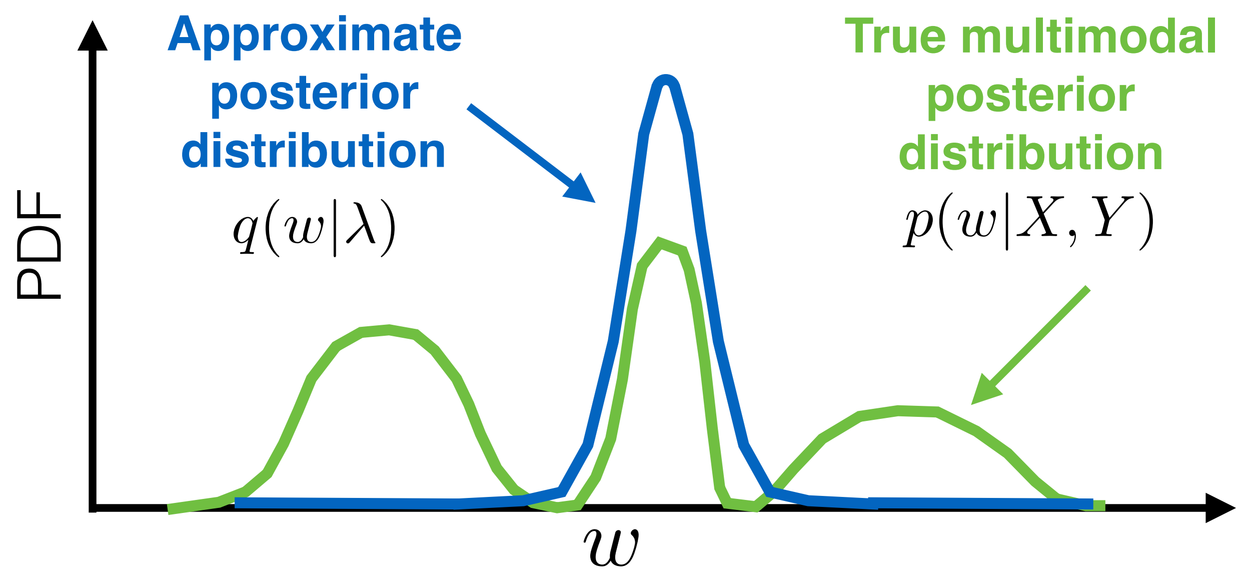
$$\mathbb{E}_{p(w|X,Y)} p(y_*|x_*, w) \approx \frac{1}{K} \sum_{k=1}^{K} p(y_*|x_*, w^k), \quad w^k \sim p(w|X,Y)$$



average net's output across several weight samples

Training Bayesian neural networks

$$\sum_{i=1}^{N} \underbrace{\mathbb{E}_{q(w|\lambda)} \log p(y^i|x^i, w)}_{\text{Data term}} - \underbrace{KL(q(w|\lambda)||p(w))}_{\text{Regularizer}} \to \max_{\lambda}$$

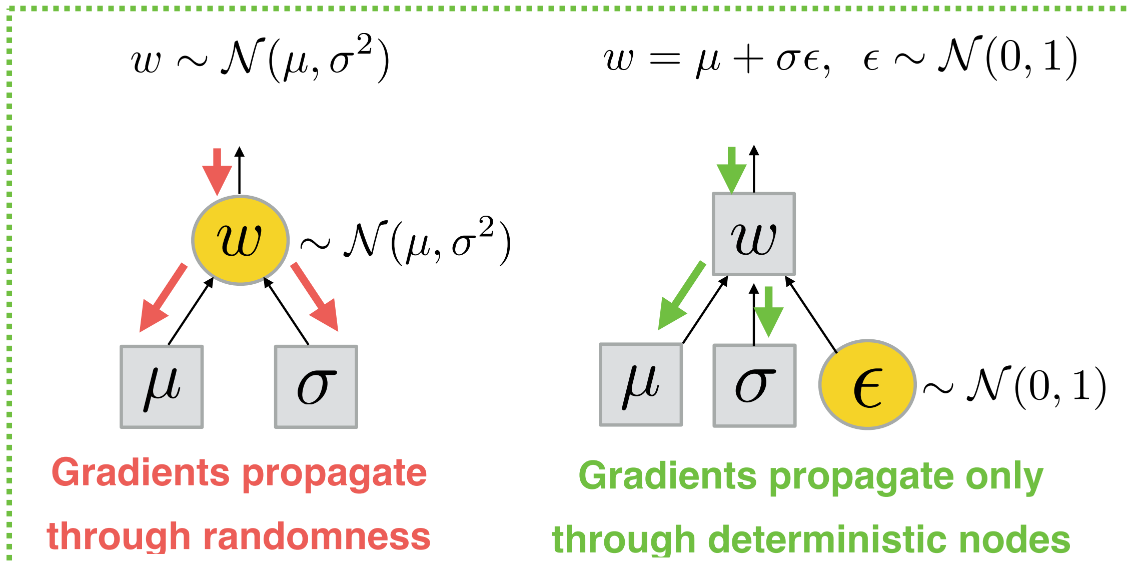


Reparametrization trick

$$\sum_{i=1}^{N} \underbrace{\mathbb{E}_{q(w|\lambda)} \log p(y^i|x^i, w)}_{\text{Data term}} - \underbrace{KL(q(w|\lambda)||p(w))}_{\text{Regularizer}} \to \max_{\lambda}$$

$$\sum_{j=1}^{m} \log p(y^{i_j}|x^{i_j}, w = f(\lambda, \epsilon^j)),$$

$$\epsilon^j \sim p(\epsilon) \qquad i_j \sim \mathrm{Unif}(1, \dots, N)$$



From general framework to particular method

$$\sum_{i=1}^{N} \mathbb{E}_{q(w|\lambda)} \log p(y^i|x^i, w) - KL(q(w|\lambda)||p(w)) \to \max_{\lambda}$$

Model specification:

- Choose particular prior $p(w)$

Training:

- Choose particular family for approximate posterior $q(w|\lambda)$
- How to compute the KL-divergence (regularizer) ?

Compression of neural networks

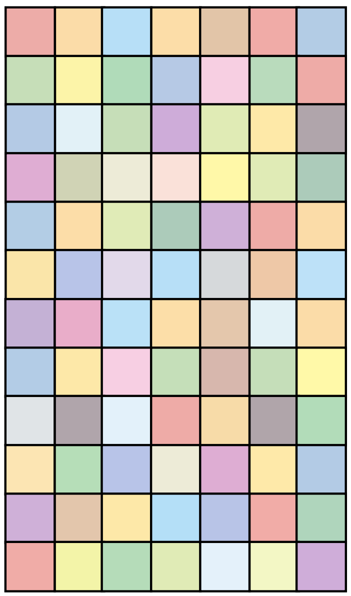
- Deep neural networks achieve state-of-the-art performance in a variety of domains
- Model quality scales with model and dataset size
- State-of-the-art models usually incorporate **tens of millions of parameters**
- But **resources** (memory, processing time) **may be limited**

Compression of neural networks

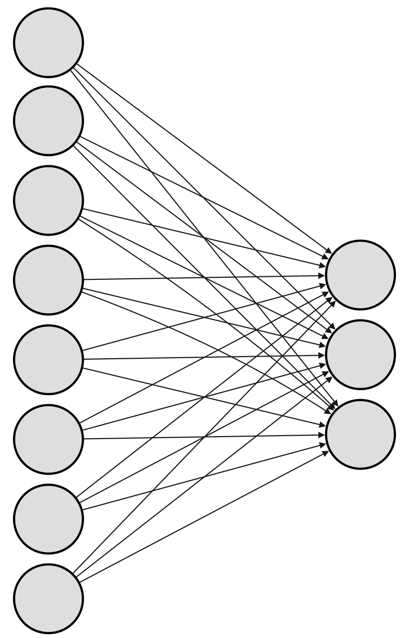
- Deep neural networks achieve state-of-the-art performance in a variety of domains
- Model quality scales with model and dataset size
- State-of-the-art models usually incorporate **tens of millions of parameters**
- But **resources** (memory, processing time) **may be limited**

- One of the solutions — sparsification

Neural network

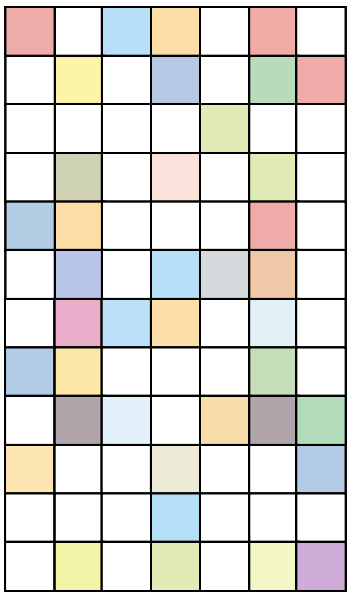


Weight matrix W



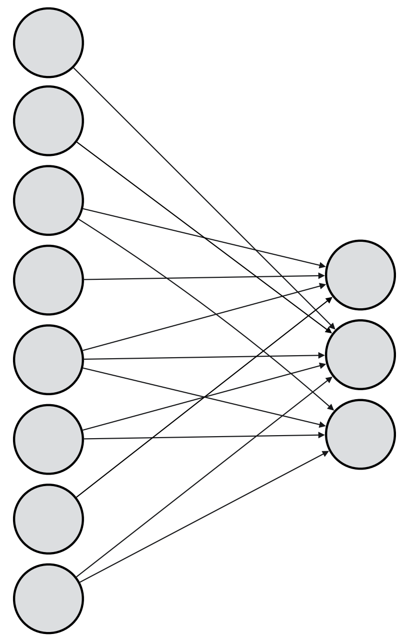
Computational graph

Sparse neural network



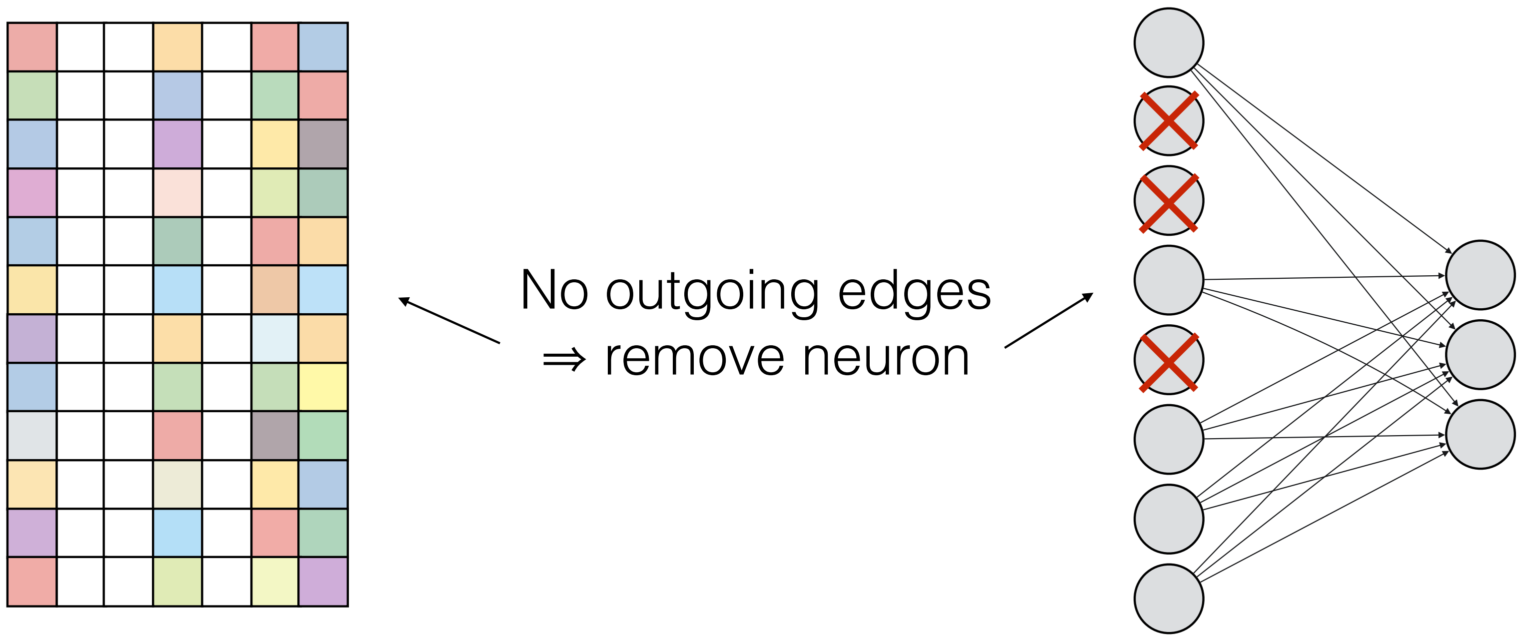
Weight matrix W

A lot of weights
set to zero



Computational graph

Structured sparsity



Weight matrix W

Computational graph

Sparsification of neural networks

Benefits:

- sparse matrices $\Rightarrow$ less memory consumption
- structured sparsification $\Rightarrow$ faster testing stage (prediction)
- regularization
- a bit more interpretable model

Drawbacks:

- sometimes leads to small quality drop

Applications:

- mobile devices, smartphones
- online services (where fast reply is needed)

Sparse variational dropout

Prior: ?

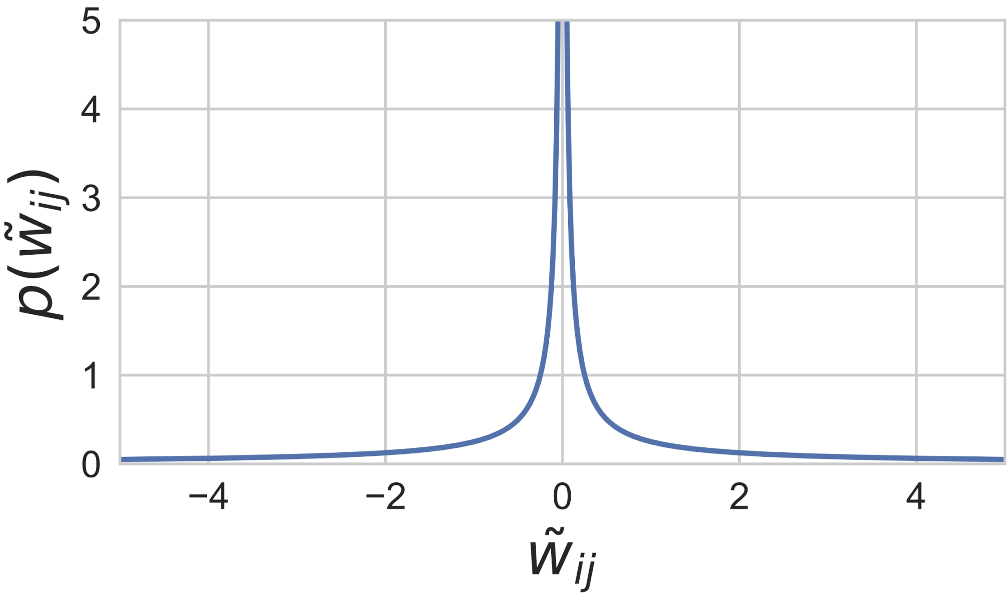
Approximate posterior: ?

Approximate KL-divergence: ?

Diederik P. Kingma et al. Variational dropout and the local reparameterization trick. NIPS 2015
Molchanov, Dmitry et al. Variational dropout sparsifies deep neural networks. ICML 2017

Sparse variational dropout

Prior: $$p(w_{ij}) \propto \frac{1}{|w_{ij}|}$$



Favors removing noisy weights!

Diederik P. Kingma et al. Variational dropout and the local reparameterization trick. NIPS 2015

Molchanov, Dmitry et al. Variational dropout sparsifies deep neural networks. ICML 2017

Sparse variational dropout

Prior: $$p(w_{ij}) \propto \frac{1}{|w_{ij}|}$$

Approximate posterior: ?

Approximate KL-divergence: ?

Diederik P. Kingma et al. Variational dropout and the local reparameterization trick. NIPS 2015
Molchanov, Dmitry et al. Variational dropout sparsifies deep neural networks. ICML 2017

Sparse variational dropout

Prior: $p(w_{ij}) \propto \dfrac{1}{|w_{ij}|}$

Approximate posterior: $q(w_{ij}|\mu_{ij}, \sigma_{ij}) = \mathcal{N}(\mu_{ij}, \sigma_{ij}^2)$

Approximate KL-divergence: ?

Diederik P. Kingma et al. Variational dropout and the local reparameterization trick. NIPS 2015
Molchanov, Dmitry et al. Variational dropout sparsifies deep neural networks. ICML 2017

Sparse variational dropout

Prior: $p(w_{ij}) \propto \dfrac{1}{|w_{ij}|}$

Approximate posterior: $q(w_{ij}|\mu_{ij}, \sigma_{ij}) = \mathcal{N}(\mu_{ij}, \sigma_{ij}^2)$

Reparametrization: $w_{ij} = \mu_{ij} + \epsilon_{ij}\sigma_{ij}, \quad \epsilon_{ij} \sim \mathcal{N}(0, 1)$

Approximate KL-divergence: ?

Diederik P. Kingma et al. Variational dropout and the local reparameterization trick. NIPS 2015
Molchanov, Dmitry et al. Variational dropout sparsifies deep neural networks. ICML 2017

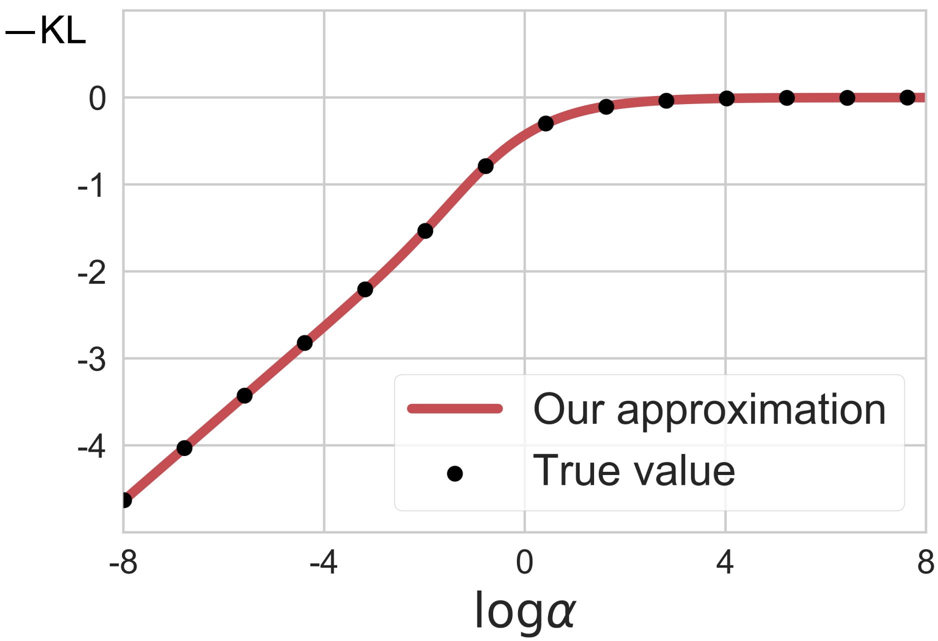
Approximating KL-divergence

Remember: training Bayesian neural networks — optimizing ELBO:

$$\sum_{i=1}^{N} \underbrace{\mathbb{E}_{q(w|\mu,\sigma)} \log p(y^i|x^i, w)}_{\substack{\text{Data term} \\ \text{Sample weights from } q}} - \underbrace{KL(q(w|\mu,\sigma)||p(w))}_{\substack{\text{Regularizer} \\ \text{Analytical approximation}}} \to \max_{\mu, \log\sigma}$$

Diederik P. Kingma et al. Variational dropout and the local reparameterization trick. NIPS 2015
Molchanov, Dmitry et al. Variational dropout sparsifies deep neural networks. ICML 2017

Approximating KL-divergence
(fully factorized)



$$-KL(q(w_{ij}|\mu_{ij},\sigma_{ij}) \,\|\, p(w_{ij})) \approx$$
$$\approx k_1 \sigma(k_2 + k_3 \log \alpha_{ij})) - 0.5 \log(1 + \alpha_{ij}^{-1}) + \mathrm{C}$$
$$k_1 = 0.63576 \qquad k_2 = 1.87320 \qquad k_3 = 1.48695$$
$$\alpha_{ij} = \frac{\sigma_{ij}^2}{\mu_{ij}^2}$$

- KL depends only on α_{ij}
- Favors large α_{ij} $\Rightarrow$ removing noisy weights

Diederik P. Kingma et al. Variational dropout and the local reparameterization trick. NIPS 2015
Molchanov, Dmitry et al. Variational dropout sparsifies deep neural networks. ICML 2017

Sparse variational dropout

Prior: $$p(w_{ij}) \propto \frac{1}{|w_{ij}|}$$

Approximate posterior: $$q(w_{ij}|\mu_{ij}, \sigma_{ij}) = \mathcal{N}\left(\mu_{ij}, \sigma_{ij}^2\right)$$

Approximate KL-divergence: $$-KL(q(w_{ij}|\mu_{ij}, \sigma_{ij}) \,\|\, p(w_{ij})) \approx f_{KL}(\alpha_{ij})$$

$$\alpha_{ij} = \frac{\sigma_{ij}^2}{\mu_{ij}^2}$$

Favors large $\alpha_{ij} \Rightarrow$ removing noisy weights

Diederik P. Kingma et al. Variational dropout and the local reparameterization trick. NIPS 2015
Molchanov, Dmitry et al. Variational dropout sparsifies deep neural networks. ICML 2017

Ok, sparsify weights. What about biases?

$$\sum_{i=1}^{N} \mathbb{E}_{q(w|\mu,\sigma)} \log p(y^i|x^i, w) - KL(q(w|\mu,\sigma)||p(w)) \to \max_{\mu, \log\sigma}$$

Treat biases as deterministic parameters and find a point estimate:

$$\sum_{i=1}^{N} \mathbb{E}_{q(w|\mu,\sigma)} \log p(y^i|x^i, w, b) - KL(q(w|\mu,\sigma)||p(w)) \to \max_{\mu, \log\sigma, b}$$

Final algorithm

Training on a mini-batch X with labels Y:

1. Sample weights: $w_{ij} = \mu_{ij} + \epsilon_{ij}\sigma_{ij}, \quad \epsilon_{ij} \sim \mathcal{N}(0,1)$
2. Forward pass: $Y_{\text{pred}} = NN(X, w, b)$
3. Backward pass + SGD step: compute stochastic gradients of ELBO:

$$\nabla_{\mu, \log\sigma, b}\left(N \cdot \text{Loss}\big(Y, Y_{\text{pred}}\big) + \text{SparseReg}(\sigma/\mu)\right)$$

Final algorithm

Training on a mini-batch X with labels Y:

1. Sample weights: $w_{ij} = \mu_{ij} + \epsilon_{ij}\sigma_{ij}, \quad \epsilon_{ij} \sim \mathcal{N}(0,1)$
2. Forward pass: $Y_{\text{pred}} = NN(X, w, b)$
3. Backward pass + SGD step: compute stochastic gradients of ELBO:

$$\nabla_{\mu,\log\sigma,b}\Big(N \cdot \text{Loss}\big(Y, Y_{\text{pred}}\big) + \text{SparseReg}(\sigma/\mu)\Big)$$

Pruning after training:

If $\mu_{ij}^2/\sigma_{ij}^2 < \text{threshold}$:

$$\mu_{ij} = 0, \;\; \sigma_{ij} = 0$$

signal-to-noise ratio

Final algorithm

Training on a mini-batch X with labels Y:

1. Sample weights: $w_{ij} = \mu_{ij} + \epsilon_{ij}\sigma_{ij}, \quad \epsilon_{ij} \sim \mathcal{N}(0,1)$
2. Forward pass: $Y_{\text{pred}} = NN(X, w, b)$
3. Backward pass + SGD step: compute stochastic gradients of ELBO:

$$\nabla_{\mu, \log\sigma, b} \Big(N \cdot \text{Loss}\big(Y, Y_{\text{pred}}\big) + \text{SparseReg}(\sigma/\mu) \Big)$$

Pruning after training:

If $\mu_{ij}^2/\sigma_{ij}^2 < \text{threshold}$:

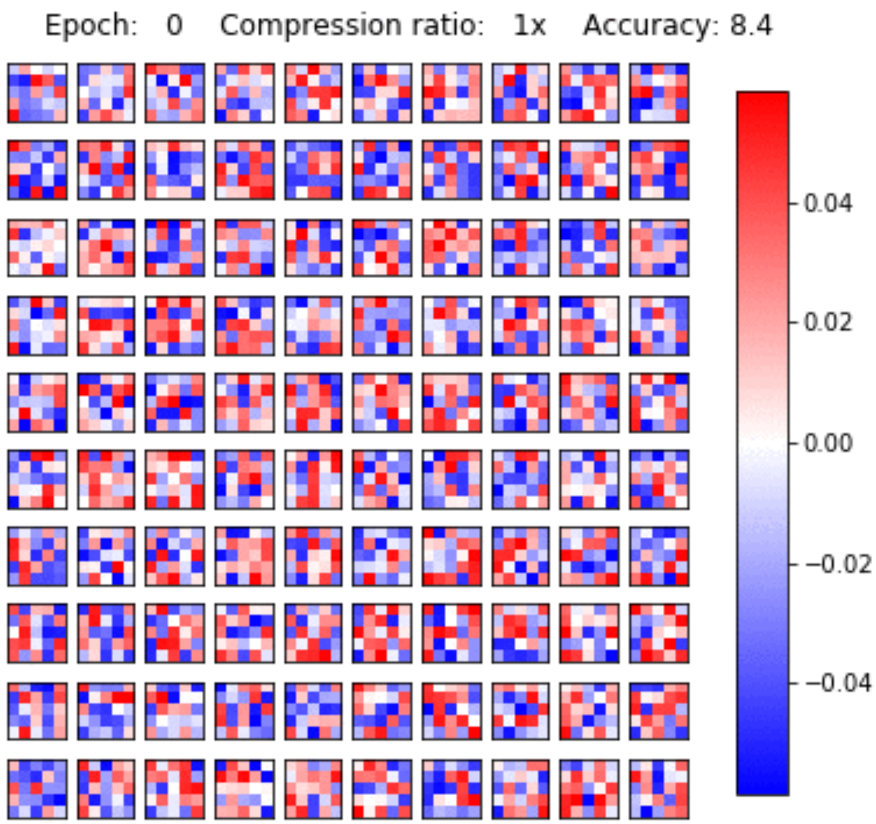
$$\mu_{ij} = 0, \ \sigma_{ij} = 0$$

Prediction for a mini-batch X :

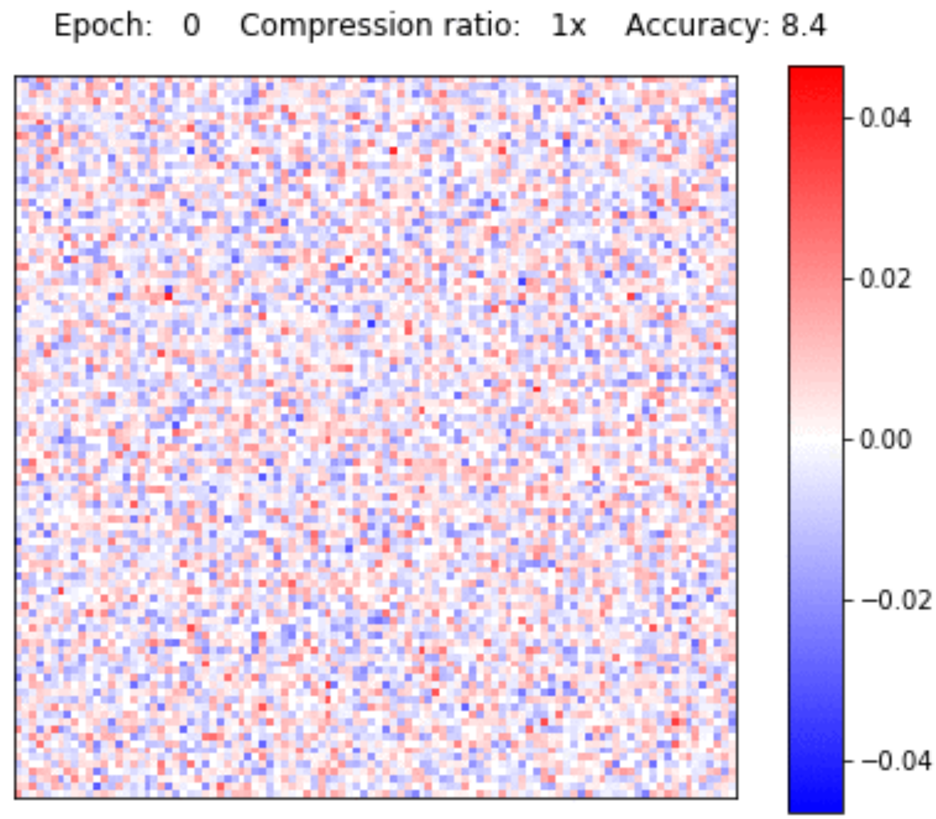
Return $Y_{\text{pred}} = NN(X, \mu, b)$

do not ensemble because we want the most compact and fast network

Sparse variational dropout: visualization

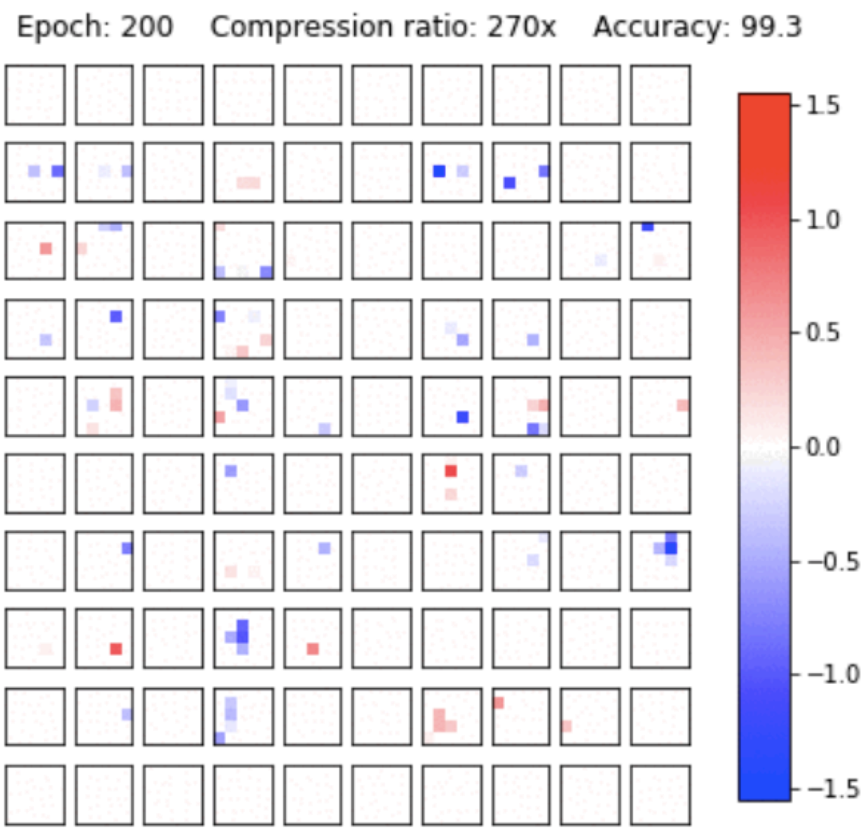


LeNet-5: convolutional layer

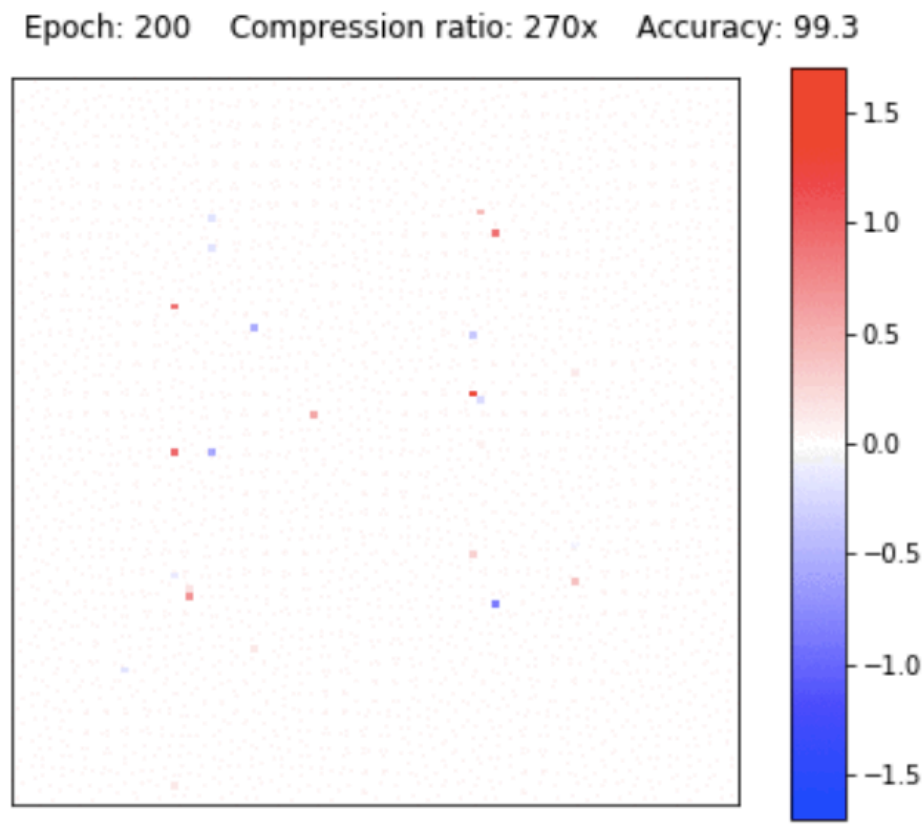


LeNet-5: fully-connected layer
(100 x 100 patch)

Sparse variational dropout: visualization



LeNet-5: convolutional layer

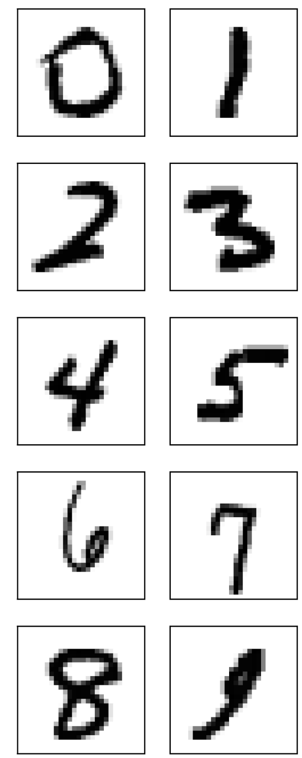


LeNet-5: fully-connected layer
(100 x 100 patch)

Lenet-5-Caffe and Lenet-300-100 on MNIST

Fully Connected network: LeNet-300-100 **Convolutional network**: Lenet-5-Caffe

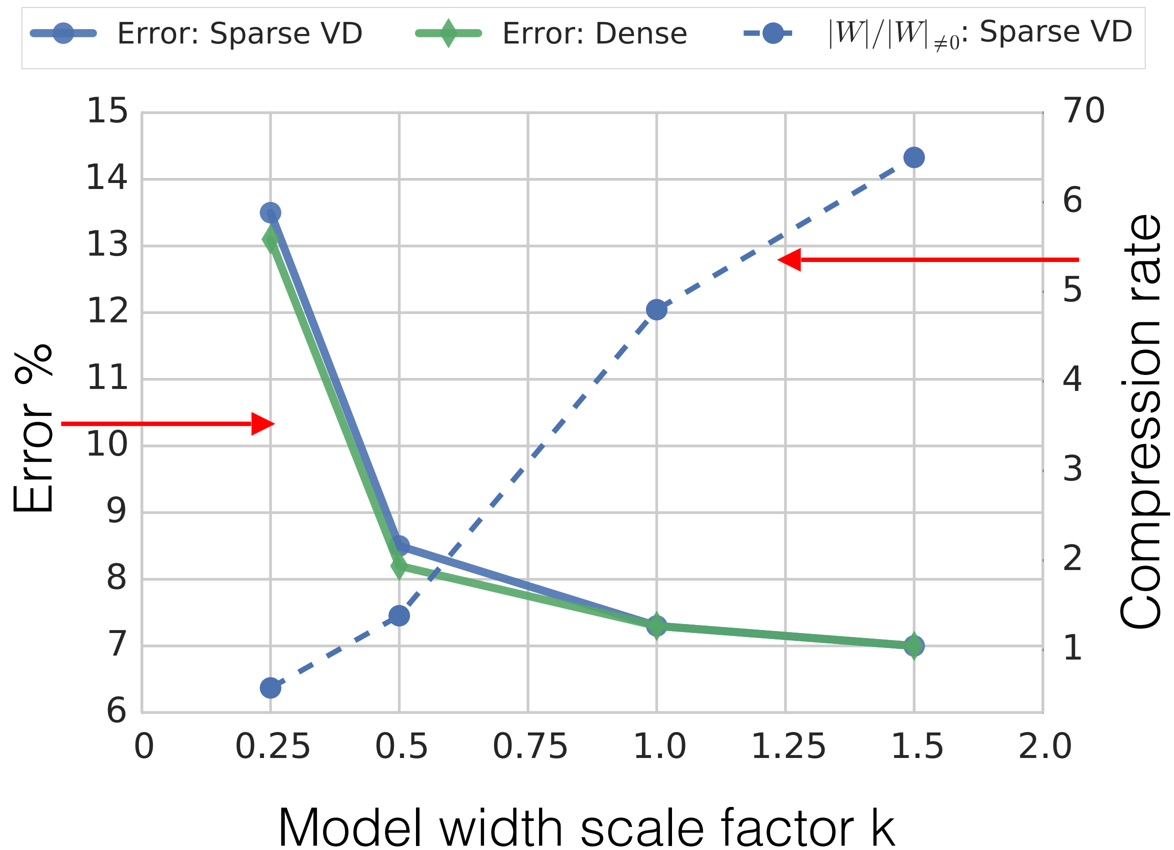
Network	Method	Error %	Sparsity per Layer %	$\frac{\lvert\mathbf{W}\rvert}{\lvert\mathbf{W}_{\neq 0}\rvert}$
LeNet-300-100	Original	1.64		1
	Pruning	1.59	$92.0 - 91.0 - 74.0$	12
	DNS	1.99	$98.2 - 98.2 - 94.5$	56
	SWS	1.94		23
(ours)	Sparse VD	1.92	$98.9 - 97.2 - 62.0$	**68**
LeNet-5-Caffe	Original	0.80		1
	Pruning	0.77	$34 - 88 - 92.0 - 81$	12
	DNS	0.91	$86 - 97 - 99.3 - 96$	111
	SWS	0.97		200
(ours)	Sparse VD	0.75	$67 - 98 - 99.8 - 95$	**280**



Molchanov, Dmitry et al. Variational dropout sparsifies deep neural networks. ICML 2017

VGG-like on CIFAR-10

Number of filters / neurons is linearly scaled by k (the width of the network)



Random Labeling

Dataset	Architecture	Train Acc.	Test Acc.	Sparsity
MNIST	FC + BD	100%	10%	—
MNIST	FC + Sparse VD	10%	10%	100%
CIFAR-10	VGG + BD	100%	10%	—
CIFAR-10	VGG + Sparse VD	10%	10%	100%

No dependency between data and labels ⇒ Sparse VD yields an empty model where conventional models easily overfit.

Zhang, Chiyuan, et al. "Understanding deep learning requires rethinking generalization."

Sparse variational dropout: key messages

- Prior distribution can encode our desirable model properties (e. g. sparse weights)
- Other Bayesian compression techniques:
  - group sparsification (removing neurons / filters)
  - quantization (low-precision weights)