Bayesian neural networks

Dmitry Molchanov

Samsung AI Center

Samsung-HSE Laboratory

Lecture outline

- What Bayesian Neural Networks (BNNs) are?
- Why go Bayesian?
- How to train BNNs?
- Sparse Variational Dropout

What you already know

- Stochastic optimization
- Bayesian modelling
- Latent variable models
- Variational inference
  - Bayesian inference ↔ (stochastic) optimization
  - (Doubly) Stochastic variational inference
  - Reparameterization Trick

⇒ **Bayesian neural networks**

Ensemble learning

- Deep Ensembles (Lakshminarayanan et al. 2017)
  - Learn independent models from different initializations
- Snapshot-based methods
  - Save several snapshots during one training procedure
  - Stochastic Gradient MCMC (Welling et al. 2011; Zhang et al. 2019)
  - SnapShot Ensembles (Huang et al. 2017)
- Stochastic neural networks
  - Pack a bunch of models into one stochastic computation graph
  - Variational inference
  - Dropout, data augmentation, BatchNorm, …

Stochastic neural networks

- Deterministic neural network:

$$p(t|x,w) = F(x,w)$$

- Stochastic neural network:

$$p(t|x,w) = \int p(t|x,w,\epsilon)p(\epsilon)d\epsilon = \mathbb{E}_{p(\epsilon)}F(x,w,\epsilon)$$

- Expected log-likelihood:

$$\mathbb{E}_{p(\epsilon)} \log p(t|X,w,\epsilon) \to \max_w$$

- “Weight scaling rule” heuristic (deterministic prediction):

$$p(t|x,w) \approx F(x,w,\mathbb{E}\epsilon)$$

Generative models vs discriminative models

Bayesian Discriminative Model:

Likelihood $p(\boldsymbol{t}|X,\boldsymbol{w}) = \prod_{i=1}^{N} p(t_i|\boldsymbol{x}_i,\boldsymbol{w})$

Can be a neural network with weights W!

Prior $p(\boldsymbol{w})$

Posterior $p(\boldsymbol{w}|X,\boldsymbol{t}) = \frac{p(\boldsymbol{t}|X,\boldsymbol{w})p(\boldsymbol{w})}{\int p(\boldsymbol{t}|X,\boldsymbol{w})p(\boldsymbol{w})d\boldsymbol{w}} \approx ?$

- Set of snapshots
- Stochastic NN

- No local latent variables; we want the posterior over the weights instead
- Much higher dimensionality
  - 10^2-10^3 for generative models, 10^5-10^8 and more for discriminative models

Why go Bayesian?

A principled framework with many useful applications

- Regularization
- Ensembling
- Uncertainty estimation
- On-line / continual learning
- Quantization
- Compression
- …

Ensembling

A Bayesian neural network is **an infinite ensemble** of neural networks

$\boldsymbol{w} \sim p(\boldsymbol{w}|X, \boldsymbol{t})$ One sample from the posterior

One element of the ensemble

Predictive distribution $p(t^*|\boldsymbol{x}^*, X, \boldsymbol{t}) = \int p(t^*|\boldsymbol{x}^*, \boldsymbol{w}) p(\boldsymbol{w}|X, \boldsymbol{t}) d\boldsymbol{w}$

And its unbiased estimate

$$\mathbb{E}_{p(\boldsymbol{w}|X,\boldsymbol{t})} p(t^*|\boldsymbol{x}^*, \boldsymbol{w}) \simeq \frac{1}{K} \sum_{i=1}^{K} p\left(t^* \middle| \boldsymbol{x}^*, \boldsymbol{w}^k\right); \qquad \boldsymbol{w}^k \sim p(\boldsymbol{w}|X, \boldsymbol{t})$$

- Higher accuracy
- More robust

Average SoftMax outputs across several samples

Uncertainty estimation

Deterministic NNs: **a point estimate** of the output, overconfident

Bayesian framework allows us to obtain **a distribution** over the outputs

- Detection of adversarial attacks and out-of-domain data
- Rejection of classification
- Active learning
- …
- Stay tuned for the next lecture!

On-line / incremental learning

Assume that the dataset arrives in independent parts.

$$\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \cdots \cup \mathcal{D}_M$$

We can train on the first dataset as usual…

$$p(\boldsymbol{w}|\mathcal{D}_1) = \frac{p(\mathcal{D}_1|\boldsymbol{w})p(\boldsymbol{w})}{\int p(\mathcal{D}_1|\boldsymbol{w})p(\boldsymbol{w})d\boldsymbol{w}}$$

… And then use the obtained posterior as the prior for the next step!

$$p(\boldsymbol{w}|\mathcal{D}_2, \mathcal{D}_1) = \frac{p(\mathcal{D}_2|\boldsymbol{w})p(\mathcal{D}_1|\boldsymbol{w})p(\boldsymbol{w})}{\int p(\mathcal{D}_2|\boldsymbol{w})p(\mathcal{D}_1|\boldsymbol{w})p(\boldsymbol{w})d\boldsymbol{w}} =$$

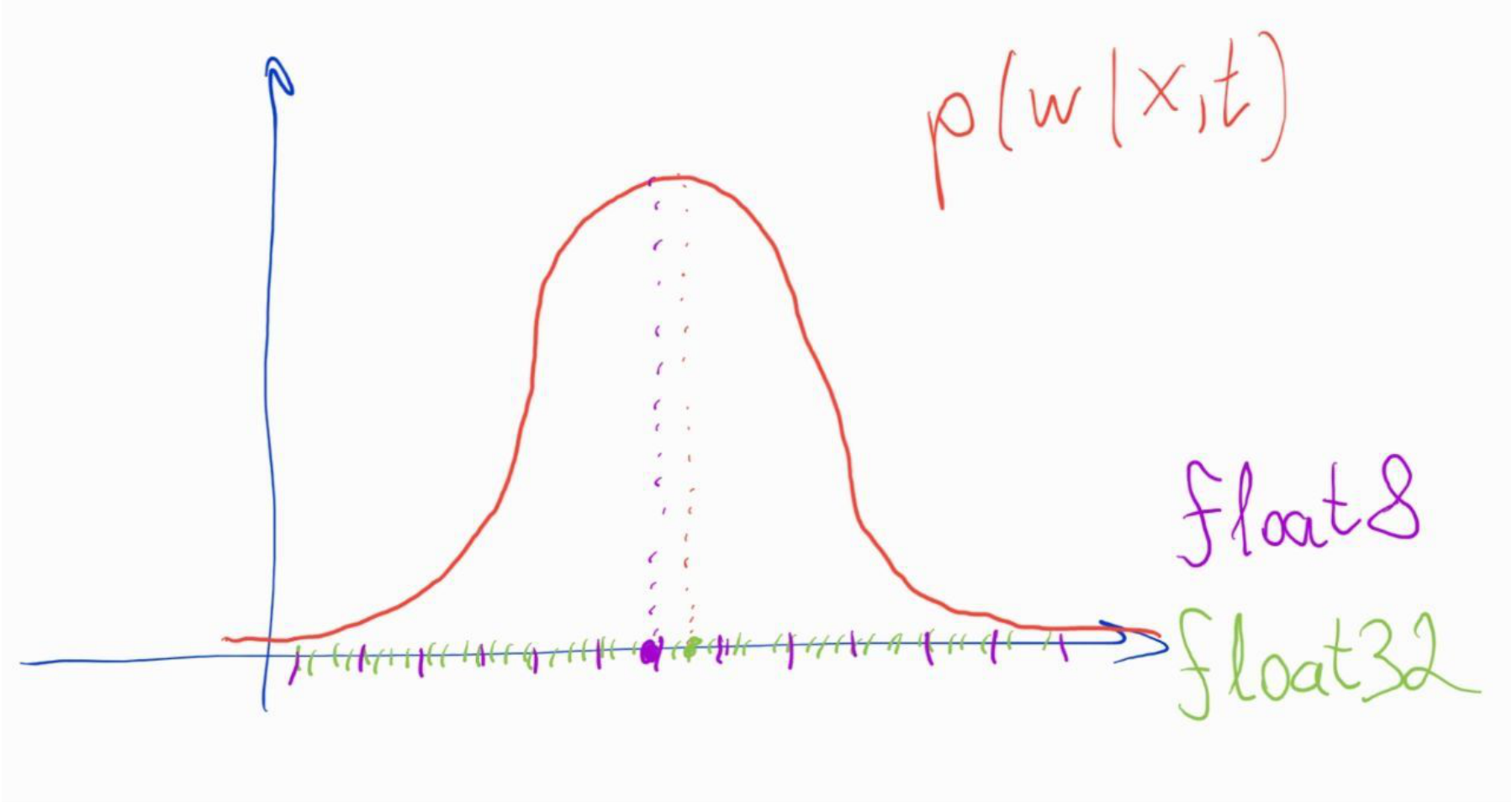
$$= \frac{p(\mathcal{D}_2|\boldsymbol{w})p(\boldsymbol{w}|\mathcal{D}_1)}{\int p(\mathcal{D}_2|\boldsymbol{w})p(\boldsymbol{w}|\mathcal{D}_1)d\boldsymbol{w}}$$

Using these sequential updates, we can find $p(\boldsymbol{w}|\mathcal{D})$!

Kirkpatrick et al. "Overcoming catastrophic forgetting in neural networks." *PNAS 2017*
Ritter et al. "Online structured laplace approximations for overcoming catastrophic forgetting." *NeurIPS 2018*

Quantization

- Automatically determine the necessary bit precision!



Variational inference for Bayesian NNs

The posterior distribution $p(\boldsymbol{w}|X,\boldsymbol{t}) = \frac{p(\boldsymbol{t}|X,\boldsymbol{w})p(\boldsymbol{w})}{\int p(\boldsymbol{t}|X,\boldsymbol{w})p(\boldsymbol{w})d\boldsymbol{w}}$

How to find it? Use (doubly stochastic) variational inference!

$$q_{\boldsymbol{\phi}}(\boldsymbol{w}) \approx p(\boldsymbol{w}|X,\boldsymbol{t})$$

$$\mathrm{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{w})\| \, p(\boldsymbol{w}|X,\boldsymbol{t})\right) \to \min_{\boldsymbol{\phi}}$$

Bayesian NN ELBO: $\mathcal{L}(\boldsymbol{\phi}) = \sum_{i=1}^{N} \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{w})} \log p(\boldsymbol{t}|X,\boldsymbol{w}) - \mathrm{KL}\big(q_{\boldsymbol{\phi}}(\boldsymbol{w})\|p(\boldsymbol{w})\big) \to \max_{\boldsymbol{\phi}}$

VAE ELBO: $\mathcal{L}(\boldsymbol{\phi}) = \sum_{i=1}^{N} \Big[\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i)} \log p(\boldsymbol{x}_i|\boldsymbol{z},\boldsymbol{\theta}) - \mathrm{KL}\big(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i)\|p(\boldsymbol{z})\big)\Big]$

Only two differences from LVMs:

- KL-term is global
- Extremely high-dimensional posterior

Reparameterization trick for Bayesian NNs

Reparameterize $q(\boldsymbol{w}|\boldsymbol{\phi})$ and plug the sample into the ELBO

$$\boldsymbol{w} \sim q(\boldsymbol{w}|\boldsymbol{\phi}) \quad \Leftrightarrow \quad \boldsymbol{w} = g(\boldsymbol{\epsilon}, \boldsymbol{\phi}); \qquad \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$$

$$\mathcal{L}(\boldsymbol{\phi}) = \mathbb{E}_{p(\boldsymbol{\epsilon})} \log p(\boldsymbol{t}|X, \boldsymbol{w} = g(\boldsymbol{\epsilon}, \boldsymbol{\phi})) - \mathrm{KL}(q \parallel p) \to \max_{\boldsymbol{\phi}}$$

Obtain an unbiased differentiable mini-batch estimator

$$\mathcal{L}(\boldsymbol{\phi}) \simeq \frac{N}{M} \sum_i \log p\left(\boldsymbol{t}_{m_i} \middle| \boldsymbol{x}_{m_i}, \boldsymbol{w} = g(\boldsymbol{\epsilon}, \boldsymbol{\phi})\right) - \mathrm{KL}(q \parallel p); \qquad \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$$

Very similar to conventional loss functions

Basically, using any kind of noise during training is close to being Bayesian

Usually just **1 sample per iteration is enough!**

Ex: dropout training as variational inference

Binary dropout results in a binary dropout posterior

$$\boldsymbol{w} = \boldsymbol{\phi} \cdot \mathrm{diag}(\boldsymbol{\epsilon}); \qquad \epsilon_i \sim \mathrm{Bernoulli}(p)$$

It can be shown that a Gaussian prior leads to L2 regularization here

ELBO for binary dropout training:

$$\mathcal{L}(\boldsymbol{\phi}) = \mathbb{E}_{p(\boldsymbol{\epsilon})} \log p(\boldsymbol{t}|X, \boldsymbol{w} = \boldsymbol{\phi} \cdot \mathrm{diag}(\boldsymbol{\epsilon})) - \lambda \|\phi\|_2^2 \to \max_{\phi}$$

- Using binary dropout means being Bayesian ☺
- There are other uses beyond regularization!
  - Ensembling
  - Uncertainty estimation
  - We can tune the dropout rate p using REINFORCE / Gumbel trick / …

Gal, Yarin, and Zoubin Ghahramani. "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning." *ICML* 2016.

Ex: Fully-Factorized Gaussians

Approximate posterior

$$q(\boldsymbol{w}) = \prod_i \mathcal{N}\left(w_i \middle| \mu_i, \sigma_i^2\right)$$

Reparameterization

$$\boldsymbol{w} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}; \qquad \epsilon_i \sim \mathcal{N}(0, 1)$$

The prior here is, e.g. a zero-centered FF Gaussian prior with variance σ_{prior}^2

ELBO:

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\sigma}) = \mathbb{E}_{p(\boldsymbol{\epsilon})} \log p(\boldsymbol{t}|X, \boldsymbol{w} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}) - \underbrace{\frac{\|\mu\|_2^2 + \|\sigma\|_2^2}{2\sigma_{prior}^2} + \sum_i \log \frac{\sigma_i^2}{\sigma_{prior}^2}}_{\text{KL between two } \mathcal{N}} \to \max_{\mu,\sigma}$$

- More tractable
- Richer approximation
- Twice as many parameters
- Start with small σ, optimize w.r.t. $\log \sigma$ to avoid constrained optimization

Titsias, Michalis, and Miguel Lázaro-Gredilla. "Doubly stochastic variational Bayes for non-conjugate inference." *ICML* 2014.

Ex: Fully-Factorized Gaussians

Approximate posterior

$$q(\boldsymbol{w}) = \prod_i \mathcal{N}\left(w_i \middle| \mu_i, \sigma_i^2\right)$$

Reparameterization

$$\boldsymbol{w} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}; \qquad \epsilon_i \sim \mathcal{N}(0, 1)$$

The prior here is, e.g. a zero-centered FF Gaussian prior with variance σ_{prior}^2

ELBO:

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\sigma}) = \mathbb{E}_{p(\boldsymbol{\epsilon})} \log p(\boldsymbol{t}|X, \boldsymbol{w} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}) - \underbrace{\frac{\|\mu\|_2^2 + \|\sigma\|_2^2}{2\sigma_{prior}^2} - \sum_i \log \frac{\sigma_i^2}{\sigma_{prior}^2}}_{R(\boldsymbol{\mu}, \boldsymbol{\sigma})} \to \max_{\mu, \sigma}$$

KL between two $\mathcal{N}$

- More tractable
- Richer approximation
- Twice as many parameters
- Start with small σ, optimize w.r.t. $\log \sigma$ to avoid constrained optimization

Titsias, Michalis, and Miguel Lázaro-Gredilla. "Doubly stochastic variational Bayes for non-conjugate inference." *ICML* 2014.

The local reparameterization trick

ELBO estimator may have high variance:

Shared noise sample!

$$\mathcal{L}(\phi) \simeq \hat{\mathcal{L}}(\phi) = \frac{N}{M}\sum_{i=1}^{M} L_i(\phi, \epsilon)$$

$$\mathrm{Var}\left[\hat{\mathcal{L}}\right] = \frac{N^2}{M^2}\left(\sum_{i=1}^{M}\mathrm{Var}[L_i] + 2\sum_{i}^{M}\sum_{i}^{M}\mathrm{Cov}\left[L_i, L_j\right]\right)$$

$$= N^2\left(\frac{1}{M}\mathrm{Var}[L_i] + \frac{M-1}{M}\mathrm{Cov}\left[L_i, L_j\right]\right)$$

Kingma, Diederik P., Tim Salimans, and Max Welling. "Variational dropout and the local reparameterization trick." *Advances in NIPS* 2015.

WARNING: NOTATION ABUSE

Green means element-wise

The local reparameterization trick

Consider a linear layer with weight matrix W, input A and output B.

$$w_{ij} \sim \mathcal{N}(\mu_{ij}, \sigma_{ij}^2)$$
$$B = AW$$

Predictions have **high** correlation because there is **one weight sample per batch**

$$\mathbb{E}B = A\mu \qquad \mathrm{Var}B = A^2\sigma^2$$
$$B \sim \mathcal{N}(A\mu, A^2\sigma^2)$$
$$B = A\mu + \sqrt{A^2\sigma^2} \odot \epsilon$$

Predictions have **zero** correlation because there is **one weight sample per object**

Kingma, Diederik P., Tim Salimans, and Max Welling. "Variational dropout and the local reparameterization trick." *Advances in NIPS* 2015.

The local reparameterization trick

LRT also reduces the variance of the stochastic gradient for **one** object

$\frac{\partial L}{\partial \mu_i}$ is the same for both RT and LRT, but

$$\frac{\partial L}{\partial \sigma_i} = \frac{\partial L}{\partial b} \cdot \frac{\partial b}{\partial \sigma_i} = \frac{\partial L}{\partial b} \cdot a_i \epsilon_i$$

RT, 1 sample per weight
A lot of redundant stochasticity

$$\frac{\partial L}{\partial \sigma_i} = \frac{\partial L}{\partial b} \cdot \frac{\partial b}{\partial \sigma_i} = \frac{\partial L}{\partial b} \cdot \frac{a_i^2 \sigma_i \epsilon}{\sqrt{a^{2\top} \sigma^2}}$$

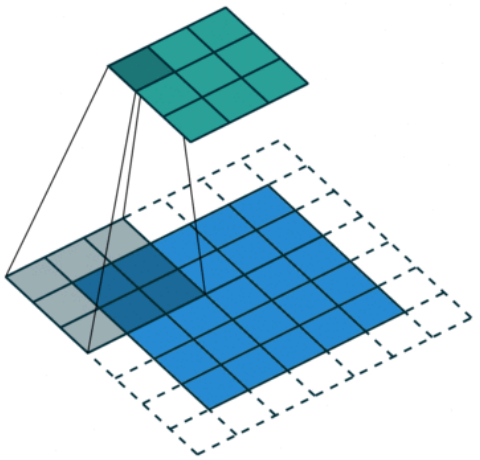
LRT, 1 sample per neuron
No redundant stochasticity

Kingma, Diederik P., Tim Salimans, and Max Welling. "Variational dropout and the local reparameterization trick." *Advances in NIPS* 2015.

WARNING: NOTATION ABUSE

Green means element-wise

LRT for convolutions



- B no longer factorizes in convolutional layers
  - Same weight samples should be used for different spatial positions
- Exact local reparameterization is too complex
  - We need to calculate the full covariance matrix for each activation
- We can use the mean-field local reparameterization as an approximation
  - Not justified *(yet)*
  - Performs much better than plain reparameterization

$$\mathbb{E}B = A \star \mu \qquad \mathrm{Var}B = A^2 \star \sigma^2$$

$$B \sim \mathcal{N}(A \star \mu, A^2 \star \sigma^2)$$

$$B = A \star \mu + \sqrt{A^2 \star \sigma^2} \odot \epsilon$$

What next?

- A few technical details
  - Treating deterministic parameters
  - How to choose prior
  - Faster test-time inference
- Sparse Variational Dropout

Treating deterministic parameters

What about other parameters, not just weight matrices?

- Biases
- Linear transformation in BatchNorm
- Any other “non-expressive” parameters

1) We can put priors over them, and treat them as random variables
2) We can treat them as deterministic parameters
   - Assume a flat prior and a delta-peak posterior
   - OR: bounding the marginal likelihood of the data given these parameters

$$\log p(\boldsymbol{t}|X, \boldsymbol{\theta}) \geq \mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\theta}) = \mathbb{E}_{q(\boldsymbol{w}|\boldsymbol{\phi})} \log p(\boldsymbol{t}|X, \boldsymbol{w}, \boldsymbol{\theta}) - \mathrm{KL}(q(\boldsymbol{w}|\boldsymbol{\phi}) \,\|\, p(\boldsymbol{w})) \to \max_{\boldsymbol{\phi}, \boldsymbol{\theta}}$$

Empirical Bayes for Bayesian NNs

- How to choose the prior distribution?
- Type-II maximum likelihood (maximum evidence):

$$\log p(\boldsymbol{t}|X,\boldsymbol{\theta}) \to \max_{\boldsymbol{\theta}}$$

$$\log p(\boldsymbol{t}|X,\boldsymbol{\theta}) \geq$$

$$\geq \mathcal{L}(\boldsymbol{\phi},\boldsymbol{\theta}) = \mathbb{E}_{q(\boldsymbol{w}|\boldsymbol{\phi})} \log p(\boldsymbol{t}|X,\boldsymbol{w}) - \mathrm{KL}\big(q(\boldsymbol{w}|\boldsymbol{\phi}) \big\| p(\boldsymbol{w}|\boldsymbol{\theta})\big) \to \max_{\boldsymbol{\phi},\boldsymbol{\theta}}$$

- It is okay when $\dim \boldsymbol{\theta}$ is small
- May overfit if $\dim \boldsymbol{\theta}$ is large
  - Ideally we would have $p(w|\boldsymbol{\theta}) = q(w|\boldsymbol{\theta}) = \delta(\boldsymbol{w}_{ML})$
  - You never know whether you can overfit with a particular parameterization
  - Add a hyperprior $p(\theta)$?
- Usually used to induce sparsity / quantization (RVM, SWS, …)

Distillation

Test-time averaging is expensive

Imaging we have a good sampler $q_t(w_t)$ for $\boldsymbol{w}$

- SG MCMC
- Variational approximate posterior
- Any other ensemble

We can train a separate deterministic neural network (student) to "mimic" the ensemble (teacher):

$$\mathcal{L}(\boldsymbol{w}_{st}) = \mathbb{E}_{q_t(\boldsymbol{w}_t)} \mathbb{E}_{p(\boldsymbol{t}|X, \boldsymbol{w}_t)} \log p(\boldsymbol{t}|X, \boldsymbol{w}_{st})$$

- Worse than the ensemble
- Better than a single network

Balan, Anoop Korattikara, et al. "Bayesian dark knowledge." *NIPS* 2015.

Bayesian neural networks: takeaways

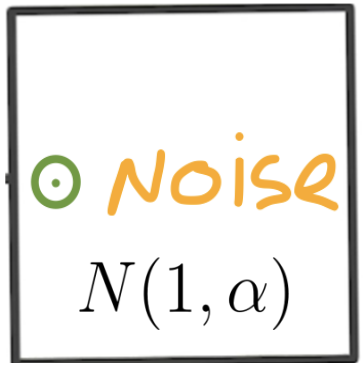
- Stochastic neural networks
- Local reparameterization
- Empirical Bayes

- Other techniques except VI:
  - MCMC
  - Laplace approximation
  - Stein variational gradient descent
  - Variational information bottleneck
  - Deep GPs and deep kernel learning
  - …

Gaussian Dropout

input

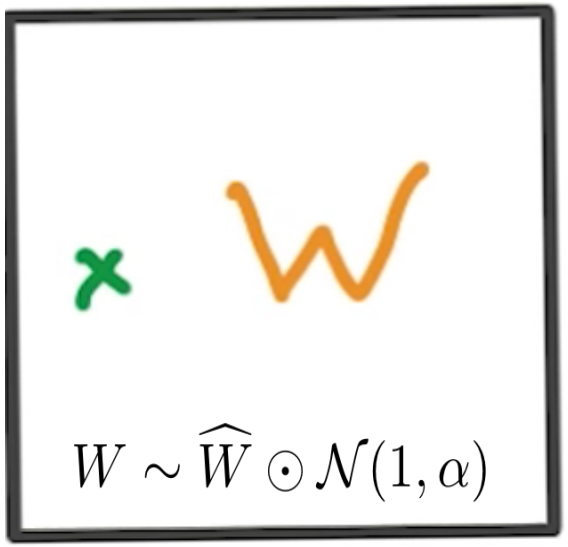
weights



noise

activations

nonlinearity



it does not change the mean value

adds multiplicative noise

$$W \sim \mathcal{N}(\widehat{W}, \alpha \odot \widehat{W}^2)$$

Variational Dropout

$$\mathbb{E}_{q(W\,|\,\phi)} \log p(y\,|\,x, W) - D_{KL}(q(W\,|\,\phi)\,||\,p(W)) \to \max_{\phi}$$

- Posterior distribution

Gaussian Dropout with noise $\sim \mathcal{N}(1, \alpha_{ij})$

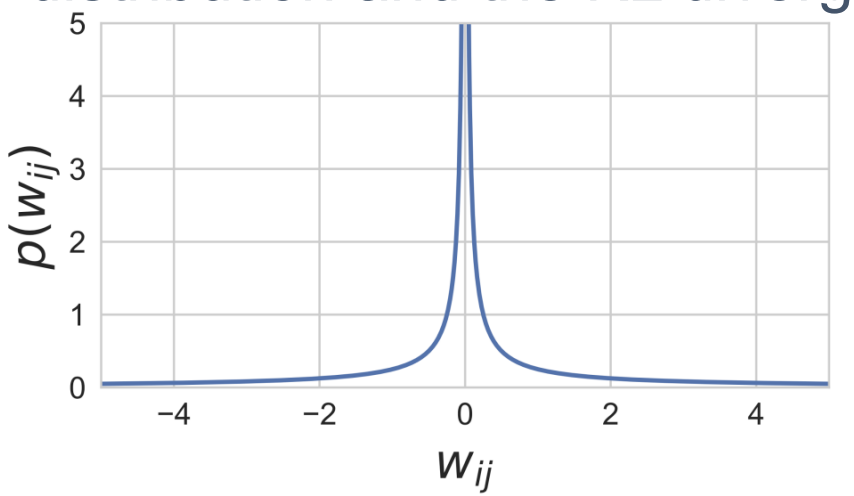
$$w_{ij} = \hat{w}_{ij} \cdot (1 + \sqrt{\alpha_{ij}} \cdot \varepsilon_{ij})$$

$$\varepsilon_{ij} \sim \mathcal{N}(0, 1)$$

$$q(w_{ij}\,|\,\phi_{ij}) = \mathcal{N}(w_{ij}\,|\,\hat{w}_{ij}, \alpha_{ij}\hat{w}_{ij}^2)$$

Prior distribution and the KL divergence term

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



$$-D_{KL}(q(w_{ij}\,|\,\hat{w}_{ij}, \alpha_{ij})\,\|\,p(w_{ij})) =$$

$$= 0.5 \log \alpha_{ij} - \mathbb{E}_{\epsilon \sim \mathcal{N}(1, \alpha_{ij})} \log |\epsilon| + \mathrm{C}$$

Does not depend on w_{ij}

Diederik Kingma, Tim Salimans, and Max Welling, Variational dropout and the local reparameterization trick

Variance Reduction

The variance of the gradients goes out of control when α are large

$$\frac{\partial \mathcal{L}}{\partial \hat{w}_{ij}} = \frac{\partial \mathcal{L}}{\partial w_{ij}} \cdot \boxed{\frac{\partial w_{ij}}{\partial \hat{w}_{ij}}}$$

$$w_{ij} = \hat{w}_{ij} \cdot (1 + \sqrt{\alpha_{ij}} \cdot \varepsilon_{ij})$$

$$\frac{\partial w_{ij}}{\partial \hat{w}_{ij}} = 1 + \sqrt{\alpha_{ij}}\varepsilon_{ij}$$ **Very noisy!**

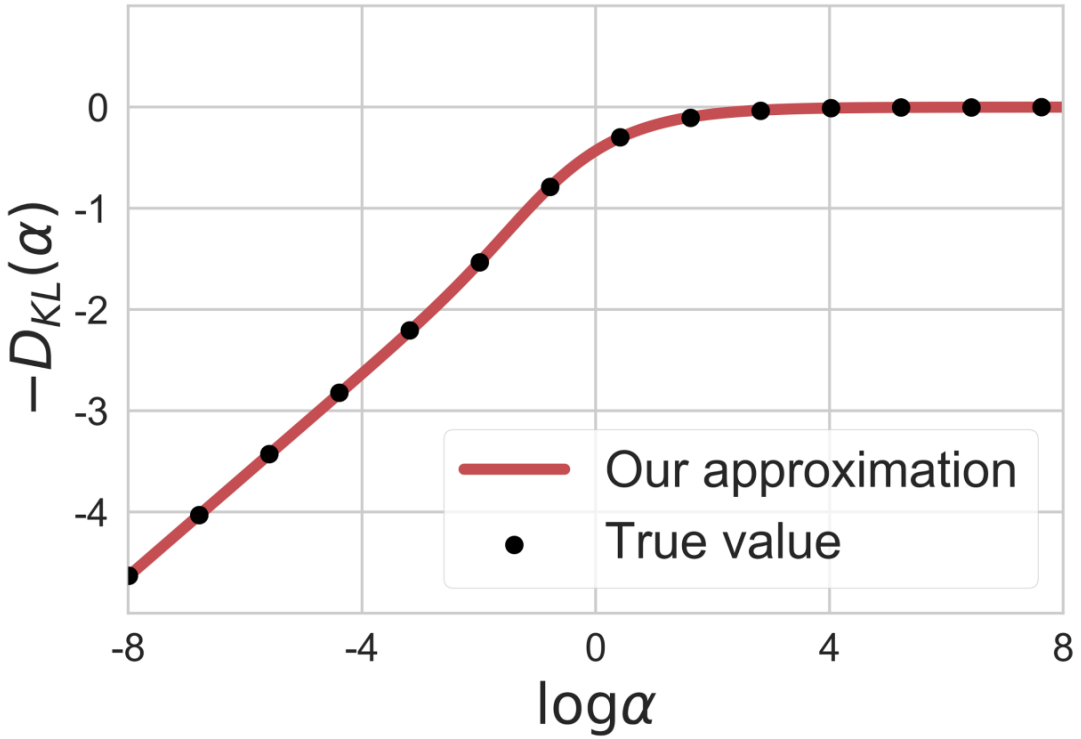
Solution: restrict $0 < \alpha < 1$ (Kingma, et. al.)

It prohibits to use large alphas!

Why do we need large alphas?

$$\mathbb{E}_{q(W|\widehat{W},\alpha)} \log p(t|X,W) - \mathrm{KL}(\alpha) \to \max_{\widehat{W},\alpha}$$

The KL term favors large dropout rates α



Large α_{ij} $(\alpha_{ij} \to +\infty)$ means:

- Infinitely large noise that corrupts the data term

$$w_{ij} = \hat{w}_{ij} \cdot (1 + \alpha_{ij} \cdot \varepsilon_{ij})$$
$$\Rightarrow \hat{w}_{ij} \to 0$$

- Equivalent binary dropout rate goes to 1

$$w_{ij} = \hat{w}_{ij}\theta_{ij} \qquad p_{ij} = \frac{\alpha_{ij}}{1+\alpha_{ij}} \to 1$$
$$\theta_{ij} \sim \mathrm{Bernoulli}(1 - p_{ij})$$

Variance Reduction

The variance of the gradients goes out of control when α are large

$$\frac{\partial \mathcal{L}}{\partial \hat{w}_{ij}} = \frac{\partial \mathcal{L}}{\partial w_{ij}} \cdot \boxed{\frac{\partial w_{ij}}{\partial \hat{w}_{ij}}}$$

Before $w_{ij} = \hat{w}_{ij} \cdot (1 + \sqrt{\alpha_{ij}} \cdot \varepsilon_{ij})$

$$\frac{\partial w_{ij}}{\partial \hat{w}_{ij}} = 1 + \sqrt{\alpha_{ij}}\varepsilon_{ij}$$ **Very noisy!**

Solution: restrict $0 < \alpha < 1$ (Kingma, et. al.) or …

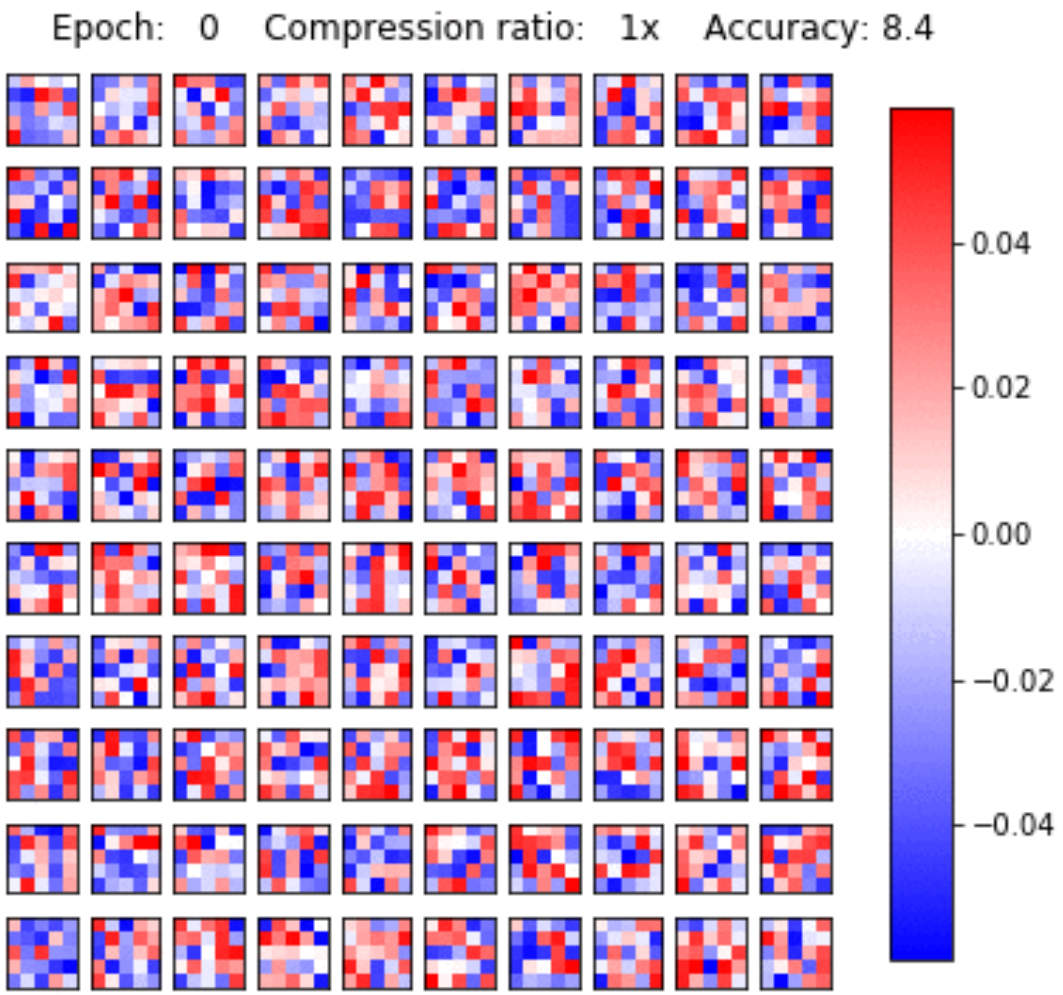
… use Additive Noise Parameterization!

After: $w_{ij} = \hat{w}_{ij} + \sigma_{ij}\varepsilon_{ij}$

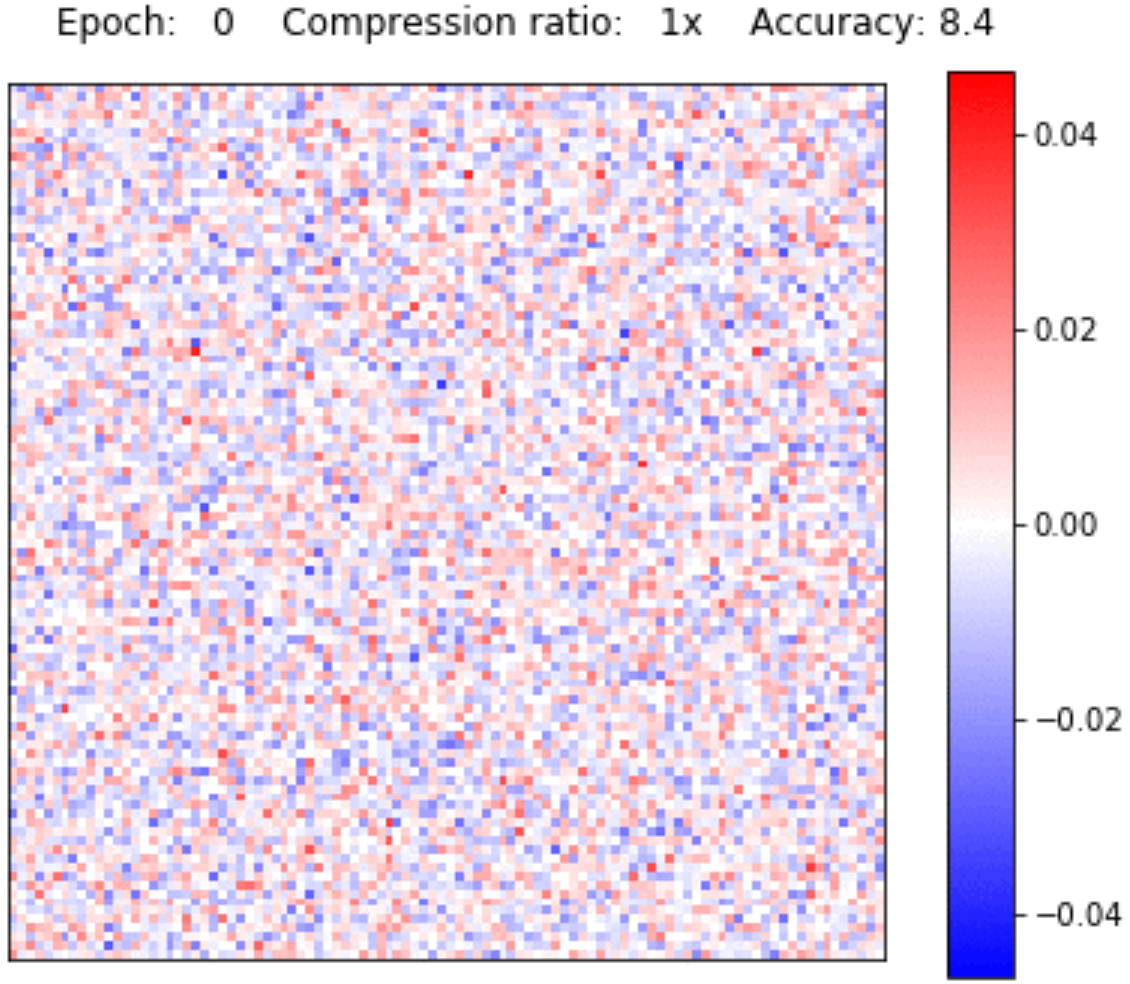
$$\frac{\partial \tilde{w}_{ij}}{\partial w_{ij}} = 1$$ **No noise!**

Optimize the VLB w.r.t. $(\hat{W}, \sigma)$ σ is a new independent variable

Visualization



LeNet-5: convolutional layer



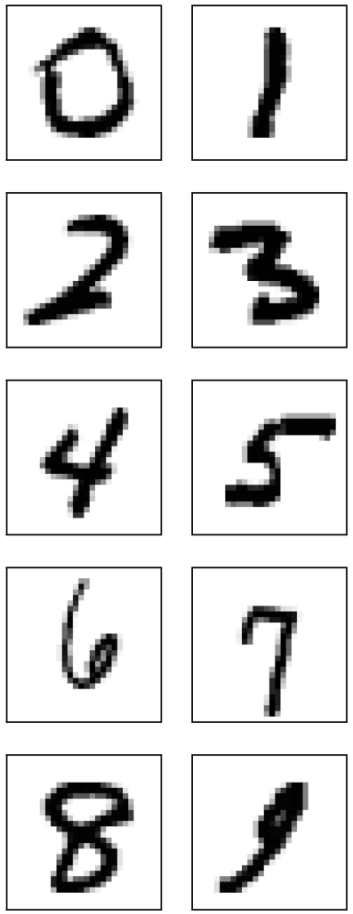
LeNet-5: fully-connected layer
(100 x 100 patch)

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

Fully Connected network: LeNet-300-100

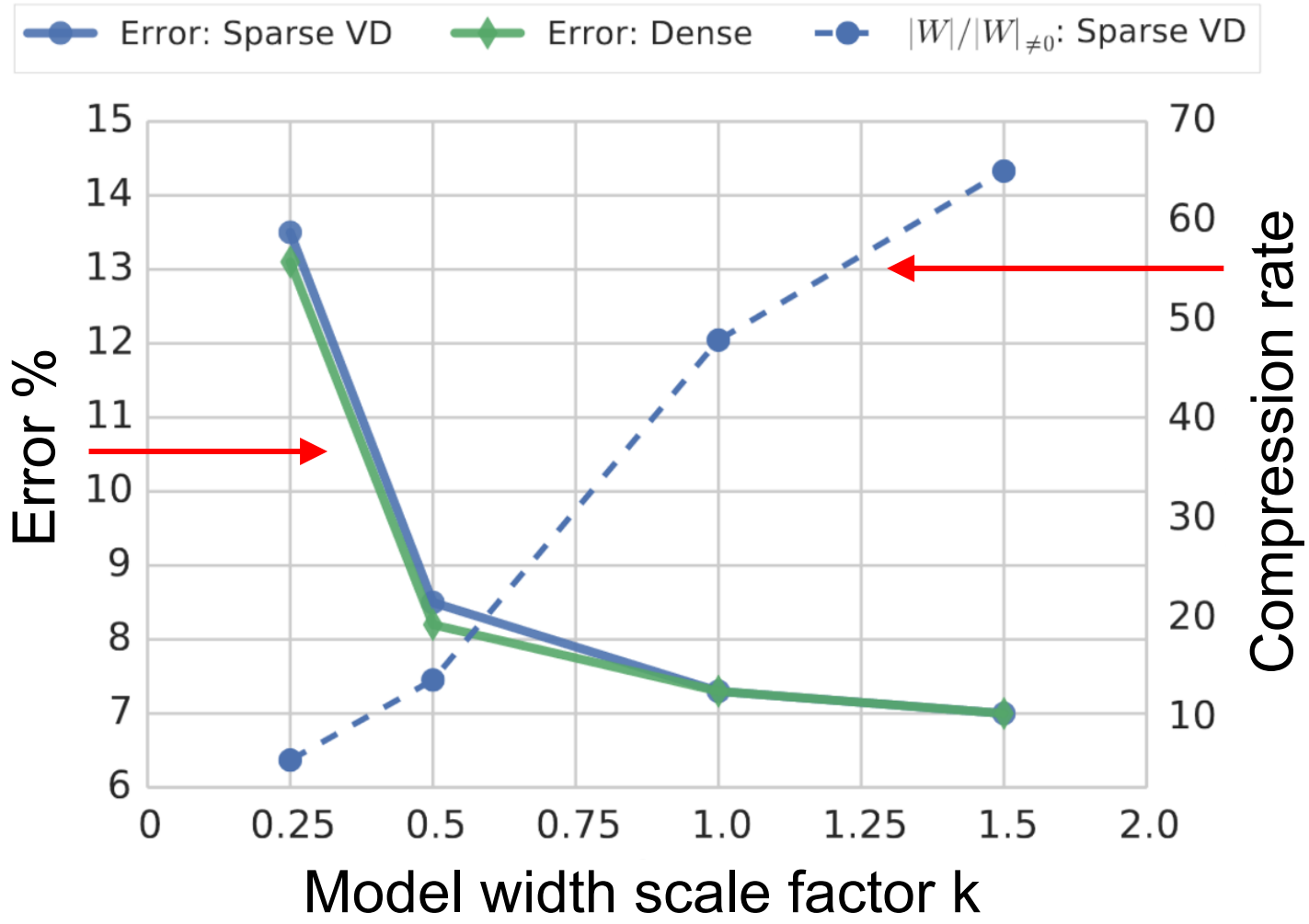
Convolutional network: Lenet-5-Caffe

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**

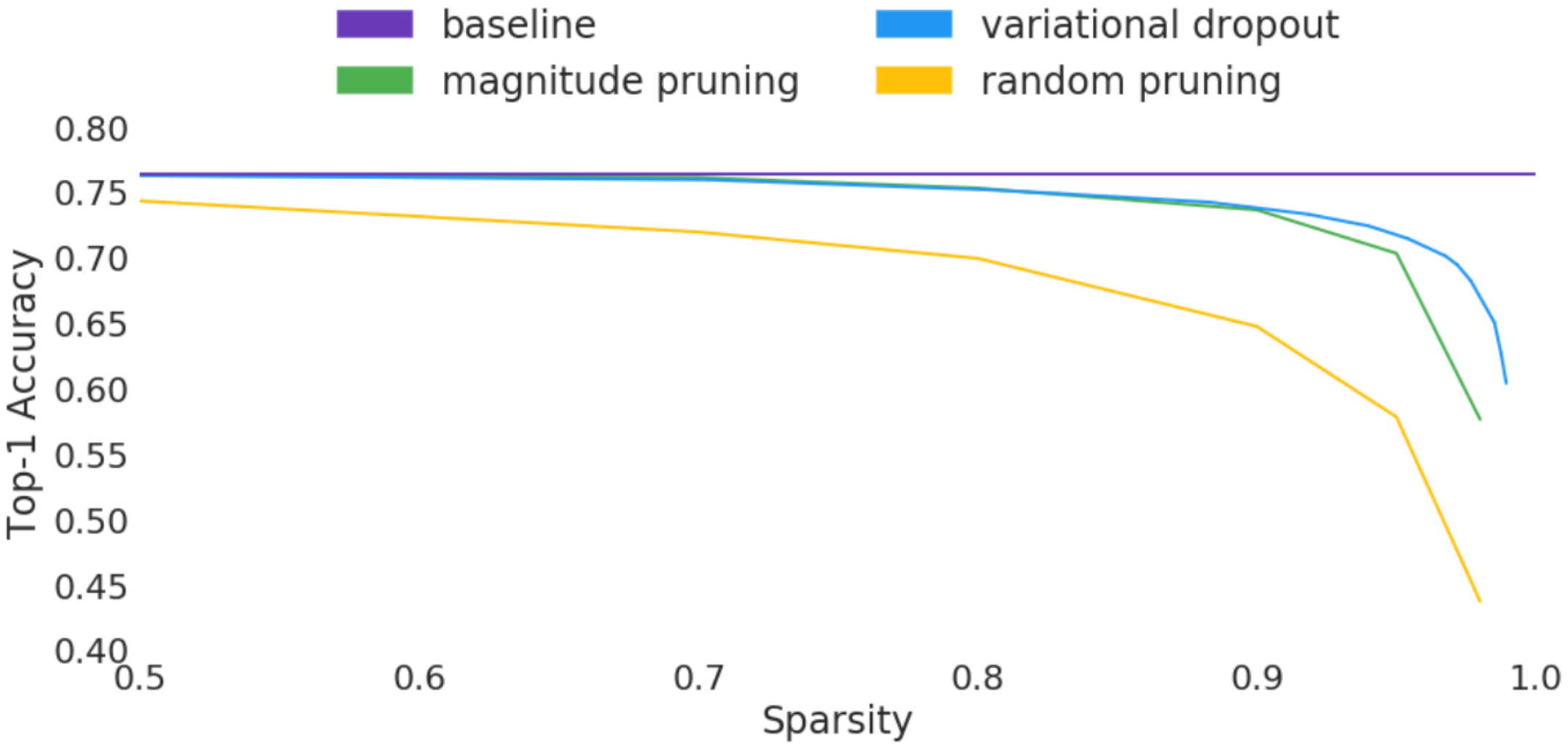


VGG-like on CIFAR-10

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



ResNet-50 on ImageNet



Gale et al. "The state of sparsity in deep neural networks." *arXiv:1902.09574* (2019).

Random Labeling

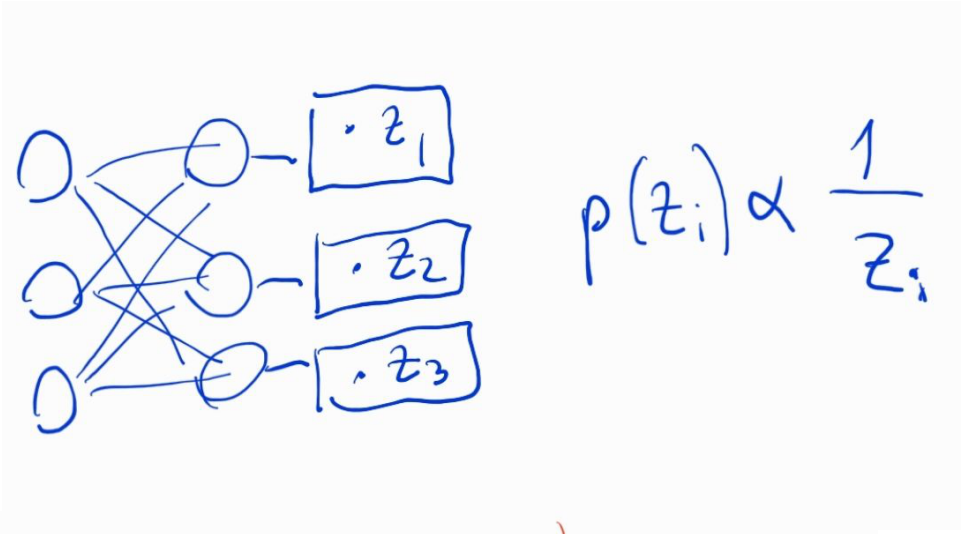
Dataset	Architecture	Train Acc.	Test Acc.	Sparsity
MNIST	FC + BD	100%	10%	—
MNIST	FC + Sparse VD	10%	10%	100%
CIFAR-10	VGG + BD	100%	10%	—
CIFAR-10	VGG + Sparse VD	10%	10%	100%

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

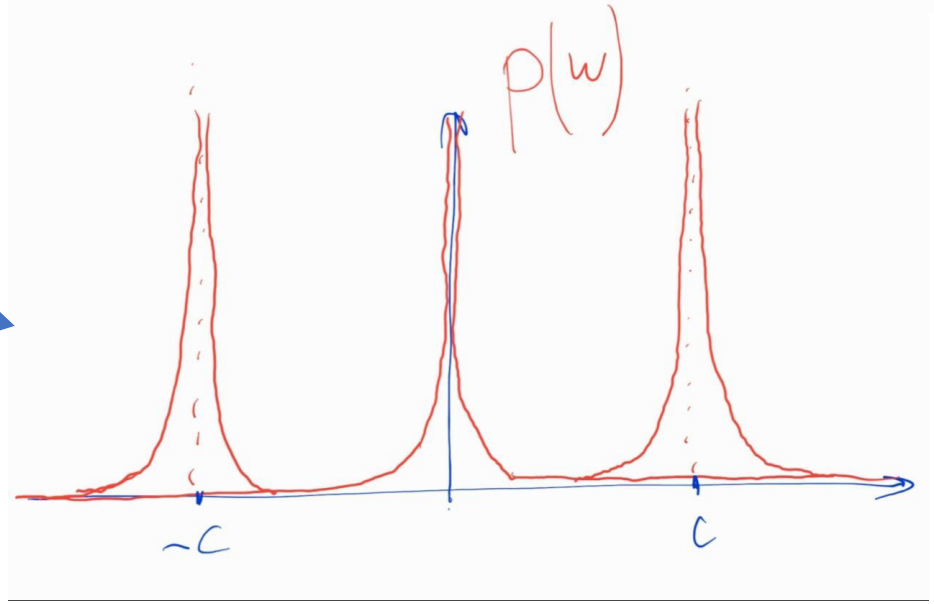
Zhang, Chiyuan, et al. "Understanding deep learning requires rethinking generalization."

Extensions

- Recurrent neural networks
- Structured sparsity



- Quantization



- Variance networks