Compressing Neural Networks

3rd IML Machine Learning Workshop, CERN, Apr. 2019

Tim Genewein

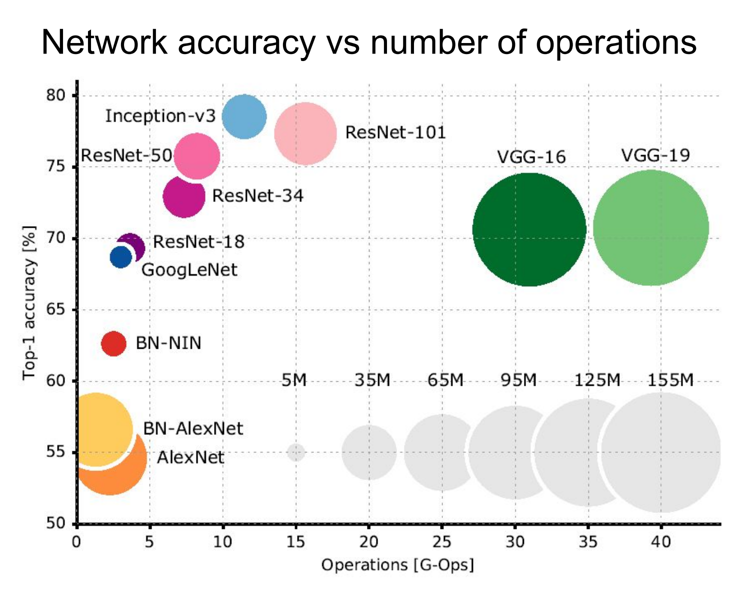
DeepMind / Bosch Center for Artificial Intelligence

DeepMind

BOSCH

Why network compression?

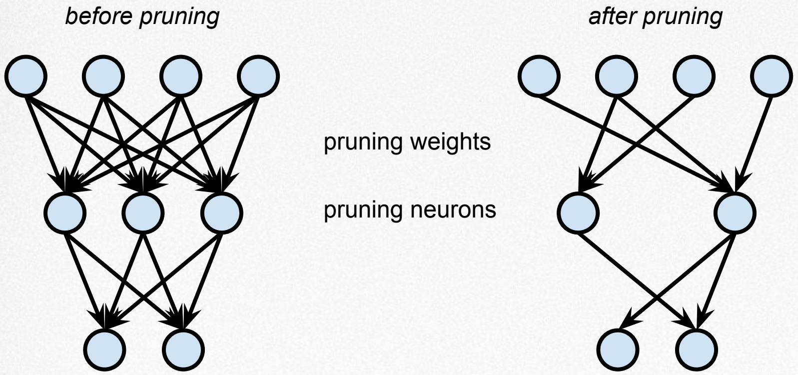
- Networks are typically over-parameterized
  - Clever architectures (SqueezeNet, MobileNets)
  - Low-rank factorization, hashing trick, …
  - *“Compression before training”*
- Weights of trained networks are redundant
  - Pruning
  - Reducing numerical precision
  - Distillation
  - *“Compression after/during training”*



Source: Canziani, A., Paszke, A., & Culurciello, E. (2016). An Analysis of Deep Neural Network Models for Practical Applications. arXiv preprint arXiv:1605.07678v2.

Pruning

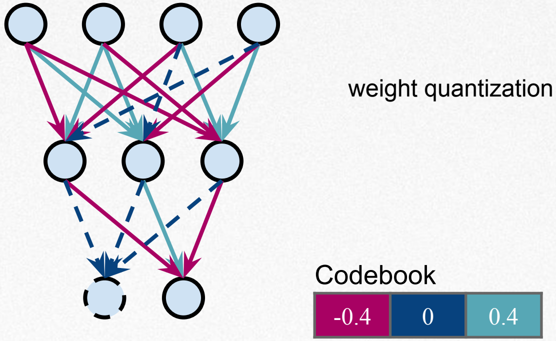
"Removal of unnecessary weights, neurons, or filters"



Which weights/neurons are important?

Quantization

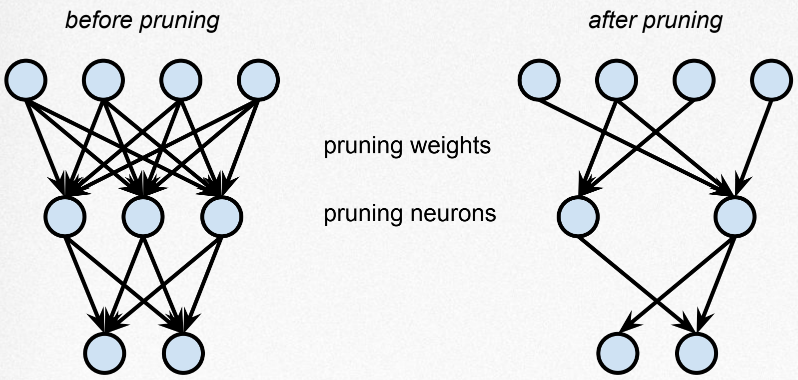
"Reduction of bit-precision of weights and activations"



Can you train networks to be robust against quantization noise?

Pruning

"Removal of unnecessary weights, neurons, or filters"



Which weights/neurons are important?

Prune based on:

- Weight-magnitude
- Activation statistics
- Second-order derivatives
- Sparse Bayesian Learning
- ...

- Fine-tuning, iterative pruning
- Weight- or neuron-pruning?

Often: >80% sparsity w/o accuracy loss

Quantization

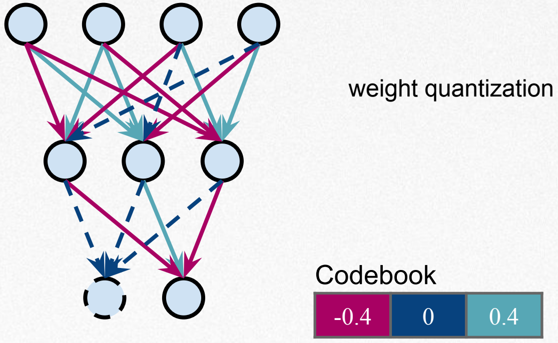
"Reduction of bit-precision of weights and activations"

Quantization methods:

- Fixed-point arithmetics
- Trained-quantization
- Few-bit quantization
- Binarization/Ternarization
- ...

- Quantize weights and/or activations?
- Post-training or during training?

Often: <16-bit easily achievable
ImageNet-scale **binary** networks possible

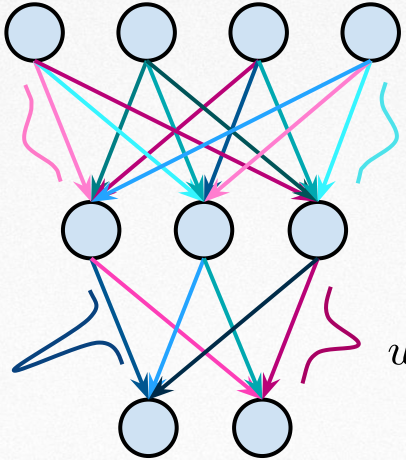


Can you train networks to be robust against quantization noise?

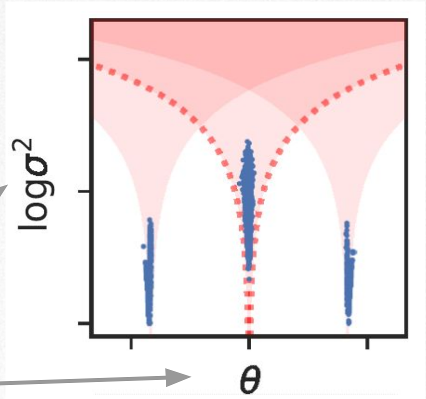
Pruning and Quantization as Inference

Bayesian neural networks

Learn a posterior distribution over weights $p(w|D) = \frac{p(D|w)p(w)}{p(D)}$

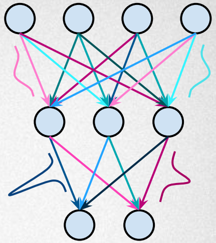


$w_{ij} \sim \mathcal{N}(\theta_{ij}, \sigma_{ij}^2)$



Gaussian mean field approximation

Bayesian neural networks



Model uncertainty

$$p(w|D) = \frac{p(D|w)p(w)}{p(D)}$$

Predictive uncertainty

$$p(y|x, D) = \int p(y|w, x)p(w|D)dw$$

Training via ELBO maximization (variational inference)

Prior

$$\mathcal{L}(\phi) = \underbrace{\sum_{n=1}^{N} \mathbb{E}_{q_\phi(w)}[\log p(y_n|x_n, w)]}_{\text{(neg.) Cross Entropy}} - \underbrace{D_{\mathrm{KL}}(q_\phi(w)||p(w))}_{\text{Regularizer}}$$

Training via ELBO maximization (variational inference)

$$\mathcal{L}(\phi) = \underbrace{\sum_{n=1}^{N} \mathbb{E}_{q_\phi(w)}[\log p(y_n|x_n, w)]}_{\text{(neg.) Cross Entropy}} - \underbrace{D_{\mathrm{KL}}(q_\phi(w)||p(w))}_{\text{Regularizer}}$$

maximize: **Data-fit** (neg. error) **-** **Model Complexity**

Automatic regularization - penalize overly complex models:

- Minimum description-length principle, Bayesian Occam's Razor

Relations to rate-distortion, Info Bottleneck

The usual "bells and whistles": (local) reparametrization trick, "warm-up", etc.

One (simple, but somewhat crude) way to implement this:

Dropout Training -> *Variational Dropout* (Kingma 2015)

$$\mathcal{L}(\phi) = \underbrace{\sum_{n=1}^{N} \mathbb{E}_{q_\phi(w)}[\log p(y_n|x_n, w)]}_{\text{(neg.) Cross Entropy}} - \underbrace{D_{\mathrm{KL}}(q_\phi(w)||p(w))}_{\text{Regularizer}}$$

Dropout noise approx corresponds to independent, Gaussian noise per weight

(mean-field approx.)

Log-uniform prior is implicit

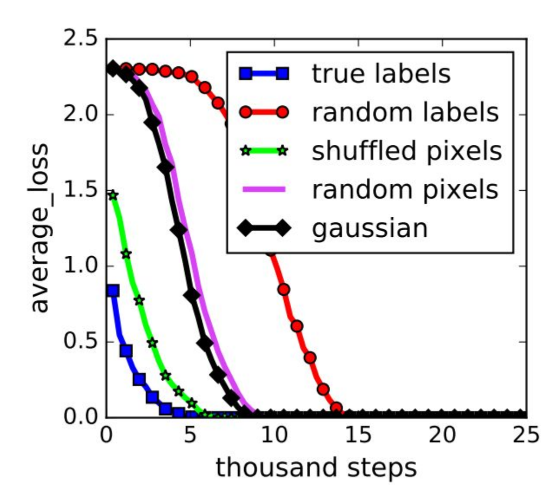
… and sparsity-inducing

(see Kingma 2015)

Effective regularization of model capacity

Effective regularization?

- Fitting random labels or pixels works really well
  - Surprisingly large memorization capacity (standard networks, standard training)



(a) learning curves

Source: Zhang et al., Understanding Deep Learning Requires Rethinking Generalization, ICLR 2017

CIFAR-10, Inception network
Training accuracy goes to 1

Test-accuracy goes to 0.1
(random guessing, not shown in plot)

Source: Achille, Soatto, Emergence of Invariance and Disentanglement in Deep Representations, JMLR 2018

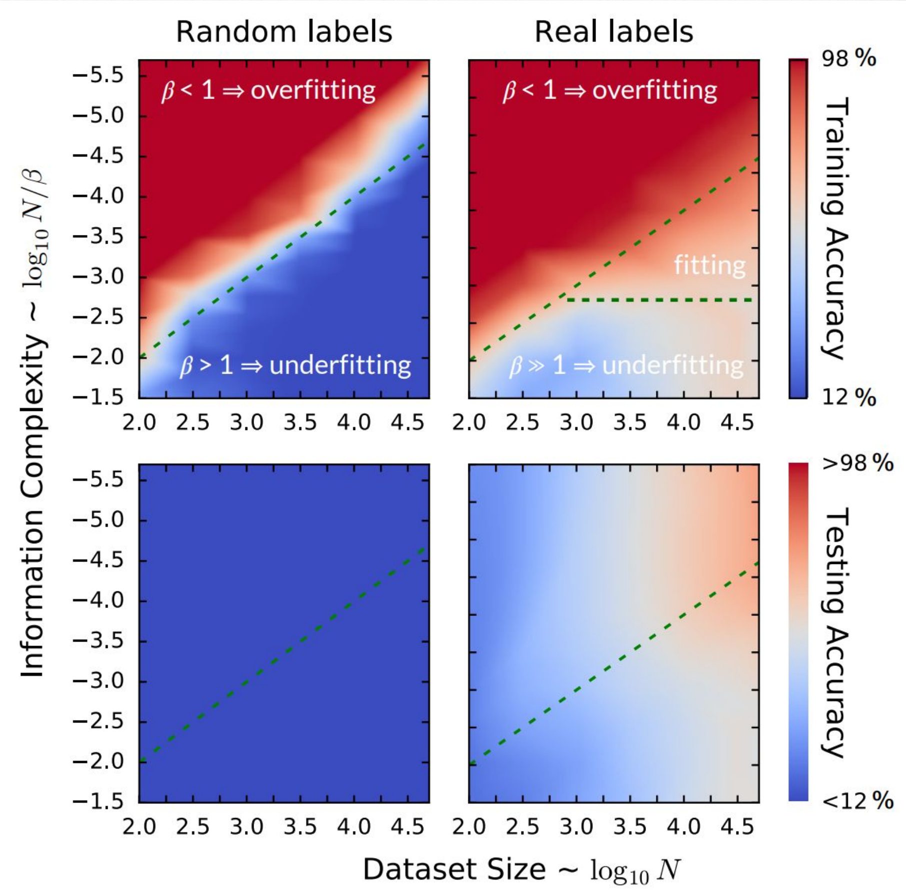
Effective regularization

How to explicitly regularize model capacity?

- Take the IB as an explicit objective function

$$\mathcal{L}(q(w|\mathcal{D})) = H_{p,q}(\mathbf{y}|\mathbf{x}, w) + \beta I(w; \mathcal{D})$$

- I(w;D) as a measure of model complexity
- Relations to Bayesian Deep Learning (similar regularizers)



Sparsity via Log-Uniform prior

$$p(w_i) \propto \frac{1}{|w_i|}$$

Kingma 2015:

Training with (Gaussian) Dropout is equivalent to **approximate variational inference** under:

- Gaussian mean-field approx.
- log-uniform prior over weights
    - see Sparse Bayesian Learning, Tipping 2001

Molchanov 2017: **Variational Dropout**

Prune weights by learning "Dropout-rates" per weight (instead of rate per layer)

-> Prior favors high Dropout rates.

Successfully prevents fitting random labels

Later: extensions for group-sparsity (pruning of whole neurons/feature-maps)

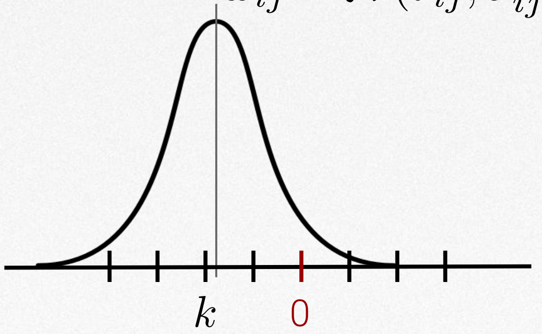
Bayes for Quantization?

… use posterior-uncertainty for determining required bit-precision

Quantization noise and posterior likelihood

3 bits

$$w_{ij} \sim \mathcal{N}(\theta_{ij}, \sigma_{ij}^2)$$



Pruning

Quantization

$$\mathcal{N}(0|\theta_{ij}, \sigma_{ij}^2)$$

$$\mathcal{N}(k|\theta_{ij}, \sigma_{ij}^2)$$

How likely is it to draw the quantized value under the posterior?

(not the same as squared error from mean, variance is also taken into account)

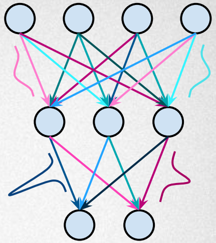
Icons by Robbe de Clerck from the Noun Project

Compressibility as a secondary objective

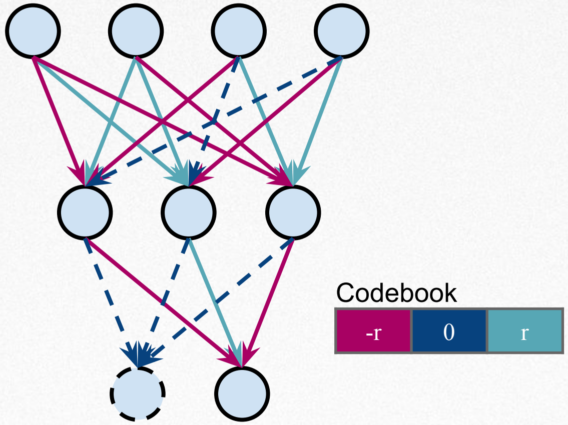
There are (surprisingly) many weight-configurations that give good task performance. Design network training such that out of all well-performing weight-configurations a ***well-compressible*** *one is favored.*

Variational Network Quantization

Variational Network Quantization, Achterhold et al., ICLR 2018

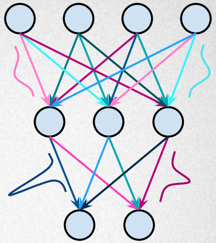


Simultaneous pruning and quantization of weights - ternary codebook

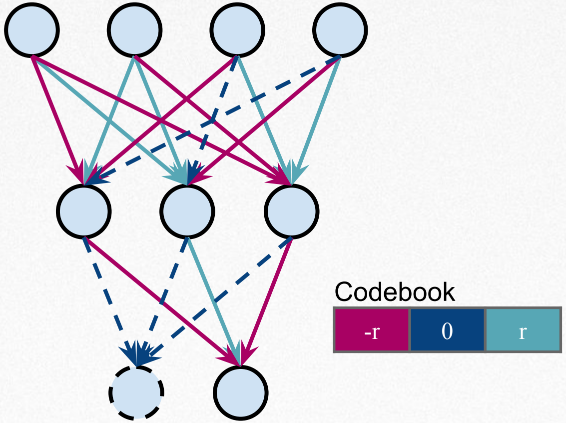


Variational Network Quantization

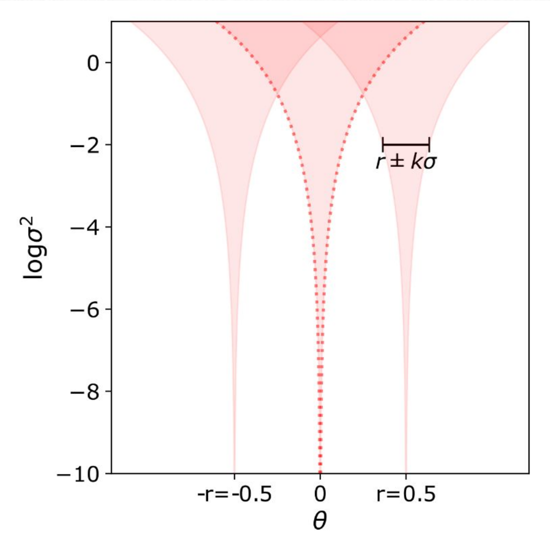
Variational Network Quantization, Achterhold et al., ICLR 2018



Simultaneous pruning and quantization of weights - ternary codebook

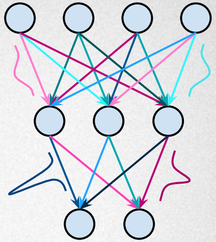


Quantizing prior: “multi-spike-and-slab”



Variational Network Quantization

Variational Network Quantization, Achterhold et al., ICLR 2018



Mixture of shifted log-uniforms

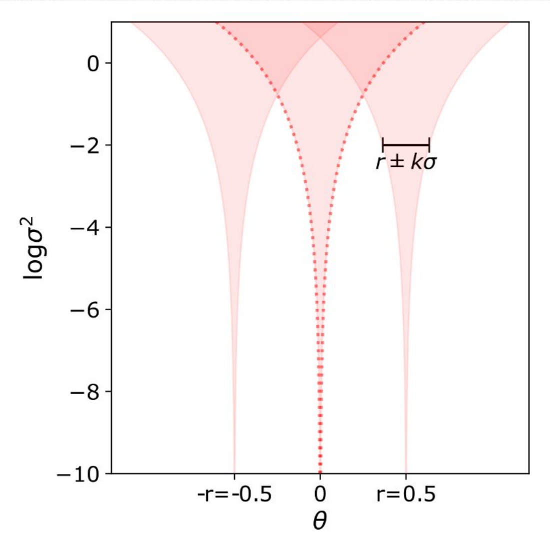
$$p(w_{ij}) \propto \sum_{k=1}^{K} a_k \int \frac{1}{|z|} \mathcal{N}(w_{ij}|c_k, z^2) dz =$$

$$= \sum_{k=1}^{K} a_k \frac{1}{|w_{ij} - c_k|}$$

Compare: sparsity-inducing ARD-style prior

$$p(w_i) \propto \frac{1}{|w_i|}$$

Quantizing prior: "multi-spike-and-slab"

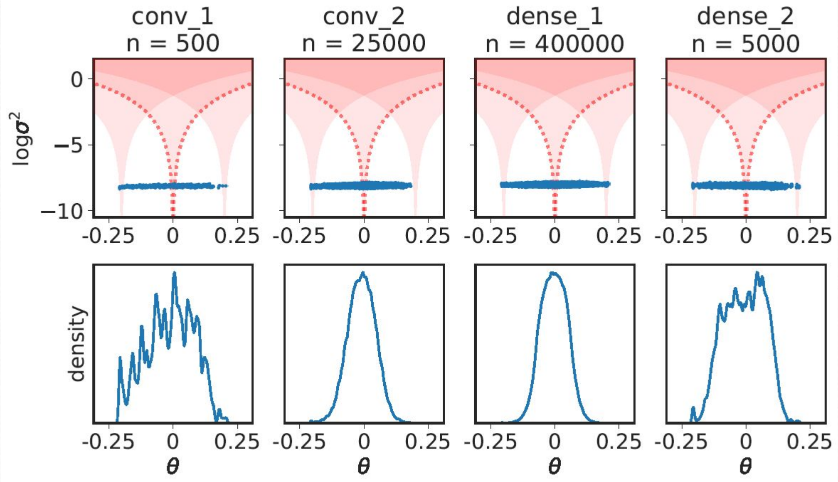


Simultaneous pruning and binarization of weights

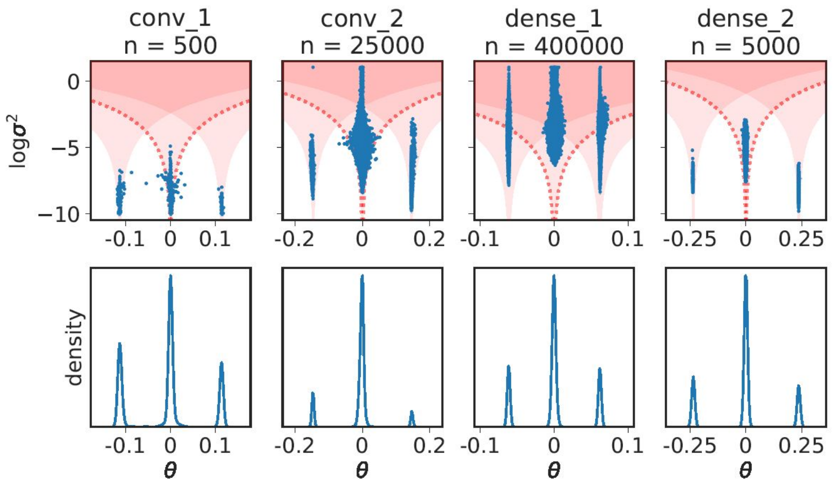
Variational Network Quantization, Achterhold et al., ICLR 2018

Same accuracy (LeNet-5 on MNIST),
qualitatively very **different weight-configuration**

Standard Training



VNQ Training

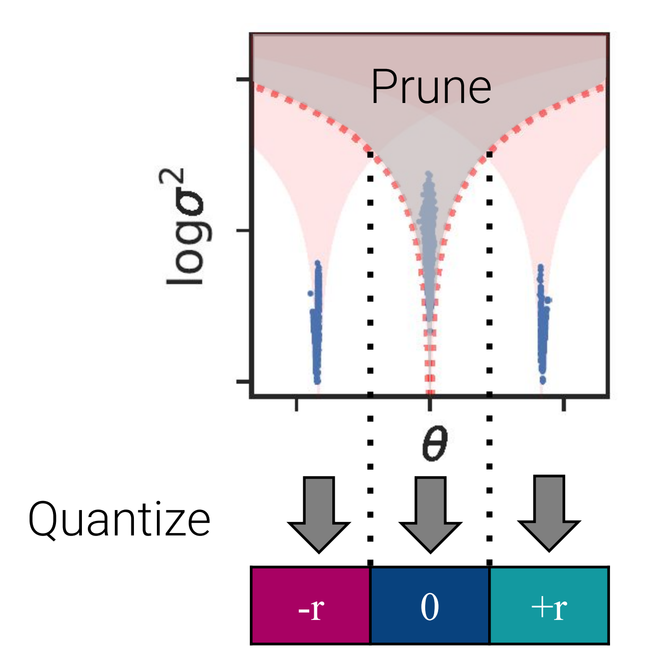


Pruning and Quantization becomes trivial after training

Variational Network Quantization, Achterhold et al., ICLR 2018

Prune weights with $\boxed{\frac{\sigma_{ij}^2}{\theta_{ij}^2} \geq T}$
(small expected value or large variance)

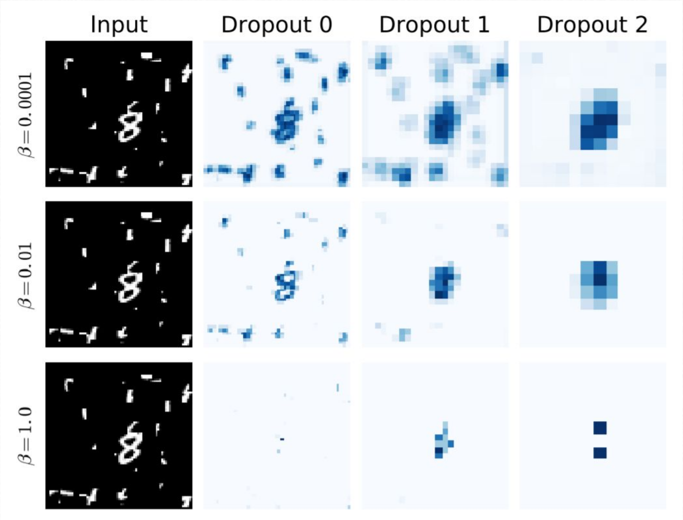
Quantize by assigning weights to closest quantization level

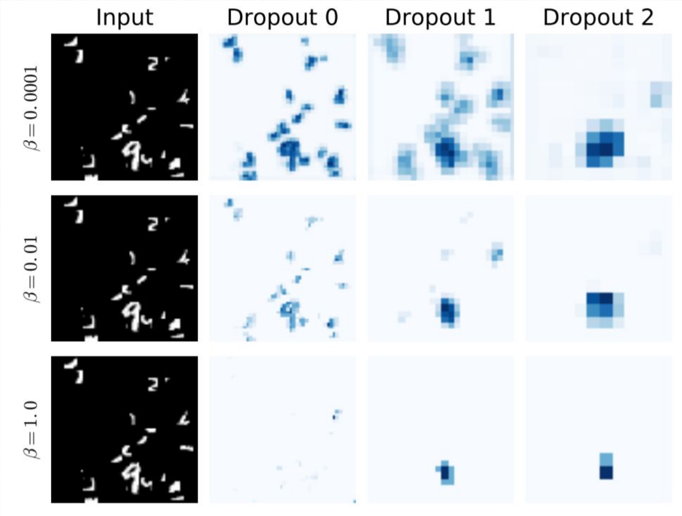


Towards bandwidth reduction: Information Dropout

Dropout noise is input-dependent (lighter blue = more noise)

Only salient information reaches deeper layers





Source: Achille and Soatto, Information Dropout: Learning Optimal Representations Through Noisy Computation, 2017

Network compression in the wild

Which compression algorithm should I pick?

There is no single compression method *"to rule them all"*.

Strong interplay between compression algorithm and:

- **Network architecture** (dense, wide, deep, skip-connections, recurrent?)
- **Task** (classification, regression, segmentation, ...)
- **HW capabilities** (memory bandwidth, accelerators for nxn convolutions, ...)

Icons by sachin modgekar, priyanka, fiaze, farias, Noura Mbarki, Chameleon Design, shashank singh, flaticondesign.com from the Noun Project

“Smaller” networks - what does that mean?

- **Offline storage space**
  - Download via mobile network, app-store limitations, ...
- **Online memory** requirements
  - Memory-bandwidth, S-RAM vs. D-RAM, bit-width of arithmetic units, ...
- **Energy consumption**
  - Battery life, heat dissipation, low-power devices
- **Forward-pass speed**
  - Real-time applications, faster training

?

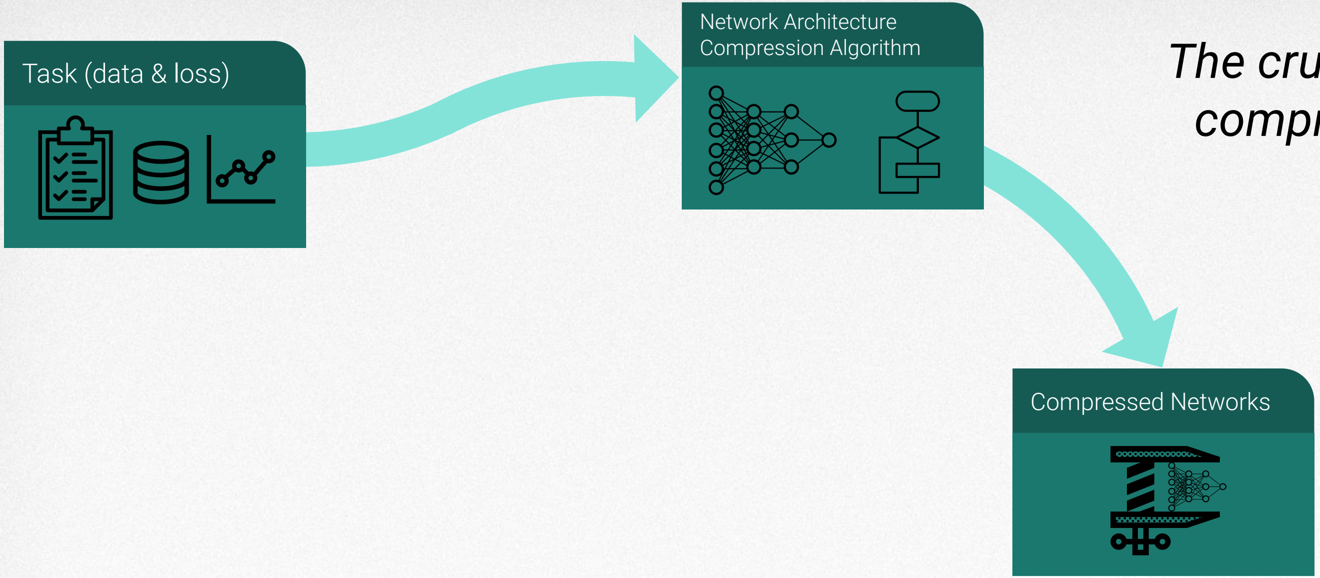
Accuracy loss acceptable?

Pre-trained network vs. training from scratch

Non-standard hardware required?

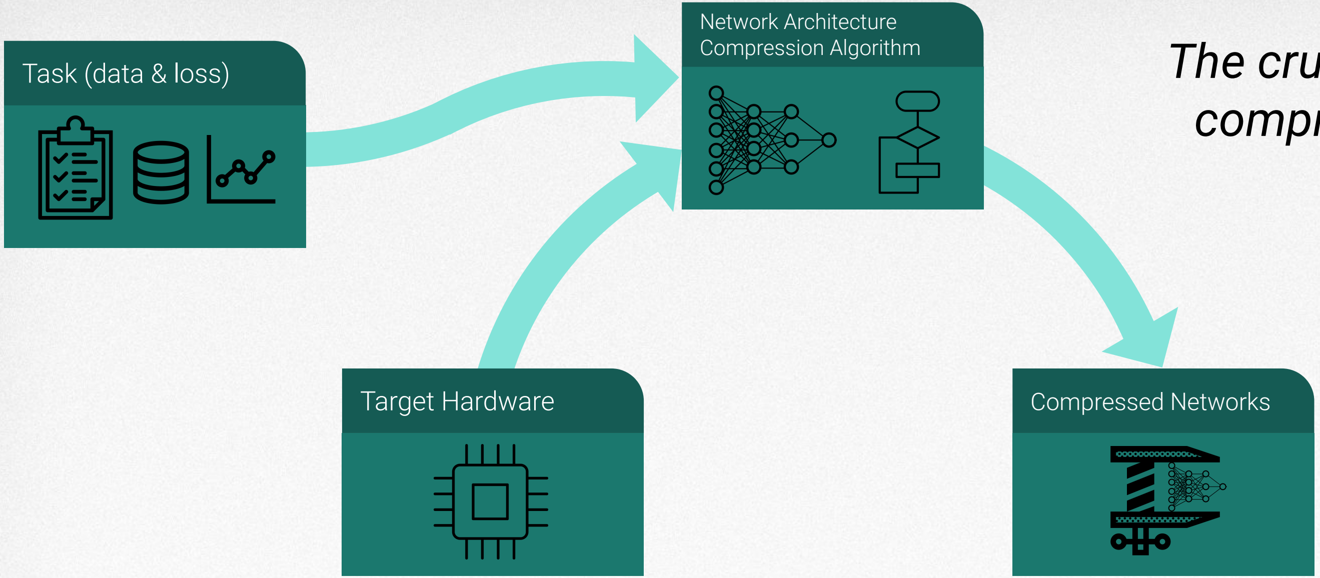
HW-capabilities (linalg accelerators, execution order, etc.)

The crux of neural network compression in practice

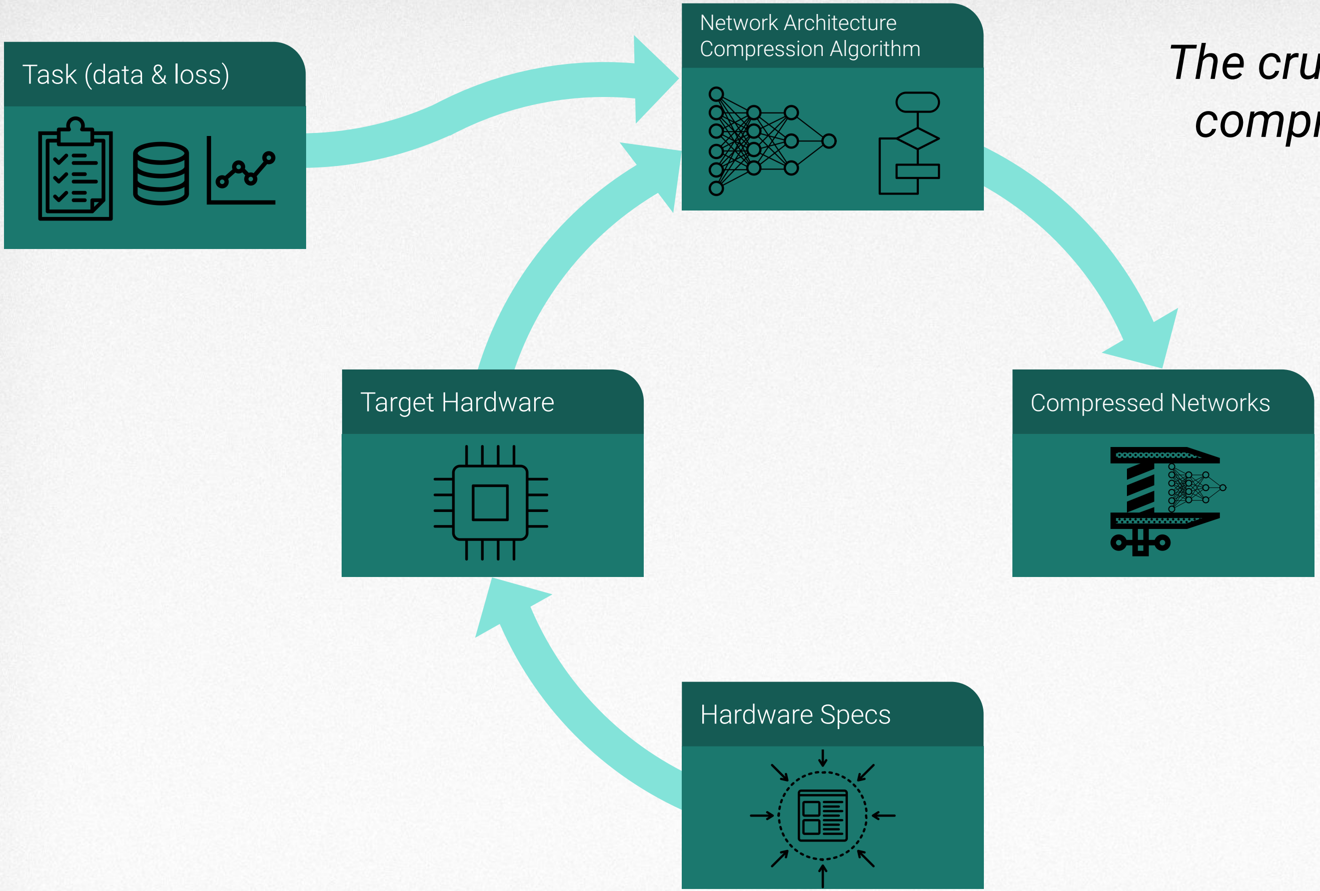


Icons by sachin modgekar, priyanka, fiaze, farias, Noura Mbarki, Chameleon Design, shashank singh, flaticondesign.com from the Noun Project

The crux of neural network compression in practice

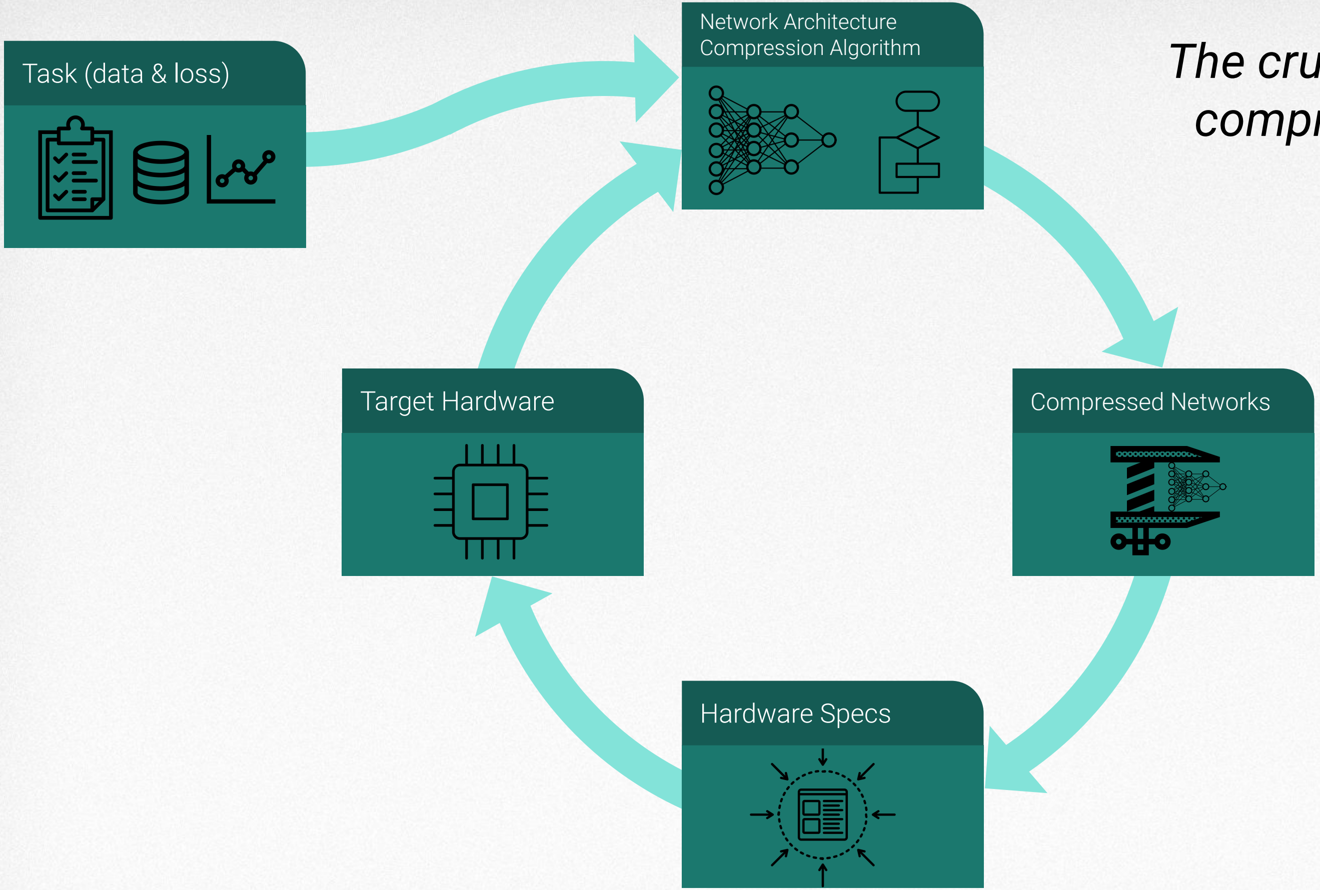


The crux of neural network compression in practice

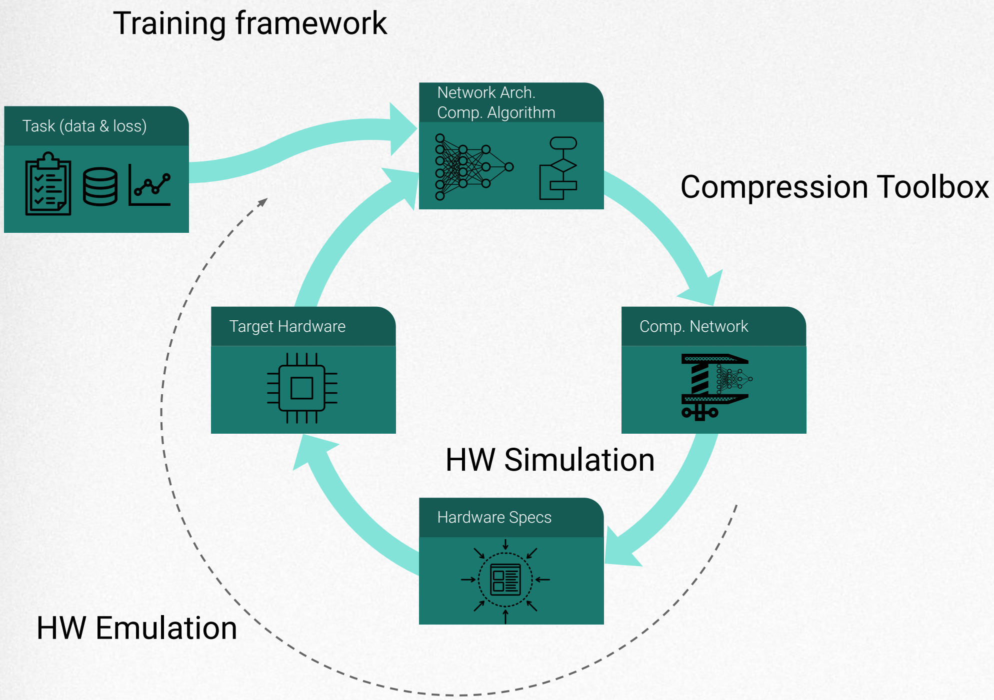


Icons by sachin modgekar, priyanka, fiaze, farias, Noura Mbarki, Chameleon Design, shashank singh, flaticondesign.com from the Noun Project

The crux of neural network compression in practice



Icons by sachin modgekar, priyanka, fiaze, farias, Noura Mbarki, Chameleon Design, shashank singh, flaticondesign.com from the Noun Project



Remedies / Mitigation

- Close the circle as fast as possible
  - Pipeline for rapid iterations
  - HW-simulation/-emulation
  - Vendor-tools vs. In-house (compression as a service, compilers)
- Scalable compression algorithms that allow for easy trade-off between resources and task performance
- Experience
- Automate the pipeline via ML(!)

Icons by sachin modgekar, priyanka, fiaze, farias, Noura Mbarki, Chameleon Design, shashank singh, flaticondesign.com from the Noun Project

Ambitious questions

- Exploiting **redundancy over time**
    - Video- and audio-data, RADAR, LIDAR
    - RNN activations
- Exploiting redundancy across inputs / sensory modalities

- Relationship between **uncertainty and compression**
    - Quantization (or setting something to 0) is another form of noise
        - Uncertainty (or posterior likelihood) quantifies whether you can afford to do that
    - E.g. semantic segmentation - do you need pixel-precise boundaries between sky and foliage (on every single input frame)?

- Input-dependent regulation of capacity? Information Dropout

Summary

- Intro to neural network compression: pruning and quantization
- One interesting family of methods: (sparse) Bayesian methods
    - Automatic regularization of model capacity
    - Flexible framework for “designing” priors
    - c.f. simultaneous pruning and quantization

- Network compression in the wild
    - Strong interplay between target hardware, task and network architecture
    - Often: classic hardware/software co-design problem

Thanks to my collaborators!

Jan
Achterhold

Bill
Beluch

Thomas
Pfeil

Jan-Mathias
Köhler

Jorn
Peters

Max
Welling

Anke
Schmeink

References

- MacKay, Bayesian methods for backpropagation networks. 1994
- **Neal, Bayesian Learning for Neural Networks. 1996**
- **Tipping, Sparse Bayesian Learning and the Relevance Vector Machine. 2001**

- Gal, Ghahramani, Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. 2015
- Kingma et al., Variational dropout and the local reparameterization trick. 2015
- Molchanov et al., Variational Dropout Sparsifies Deep Neural Networks. 2017
- Ullrich et al., Soft Weight-Sharing for Neural Network Compression. 2017
- **Louizos et al., Bayesian Compression for Deep Learning. 2017**
- Federici et al., Improved Bayesian Compression. 2017
- Neklyudov et al., Structured Bayesian Pruning via Log-Normal Multiplicative Noise. 2017
- Achterhold et al., Variational Network Quantization. 2018
- **Achille and Soatto, Information Dropout: Learning Optimal Representations Through Noisy Computation, 2017**
- Frankle and Carbin, The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. 2018
- Gale et al., The State of Sparsity in Deep Neural Networks. 2019
- **Havasi et al. Minimal random code learning: getting bits back from compressed model parameters. 2019**

Sparse Bayesian Learning

...use a sparsity-inducing prior

Automatic Relevance Determination

Originally (MacKay 1994, Neal 1996): determine relevant inputs to NN

Idea: Weights of relevant inputs should have broad range of values

For a single input: $\mathcal{N}(w|0, \alpha^{-1})$

Shared scale prior

$$p(\alpha) = \mathrm{Gamma}(\alpha|a, b)$$

Posterior:

Relevant inputs' weights: low precision α

Irrelevant inputs' weights: high precision α

Sparse Bayesian Learning

Tipping 2001: ARD idea for individual parameters (of an SVM)

$$p(w|\alpha) = \prod_{i=1}^{N} \mathcal{N}(w_i|0, \alpha_i^{-1}) \qquad p(\alpha) = \prod_{i=1}^{N} \mathrm{Gamma}(\alpha_i|a, b)$$

Irrelevant parameters -> high precision, expected value close to 0

-> **Prune** by thresholding α_i

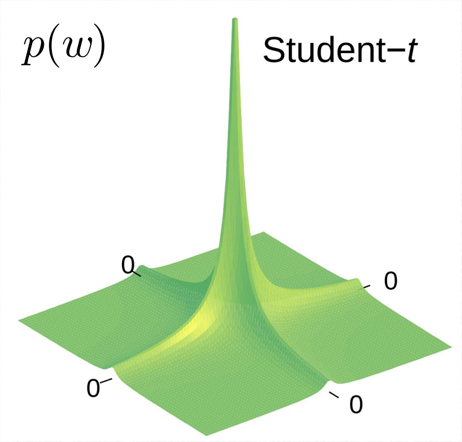
Sparse Bayesian Learning

Tipping 2001: ARD idea for individual parameters (of an SVM)

$$p(w|\alpha) = \prod_{i=1}^{N} \mathcal{N}(w_i|0, \alpha_i^{-1}) \qquad\qquad p(\alpha) = \prod_{i=1}^{N} \mathrm{Gamma}(\alpha_i|a, b)$$

Observation: ARD-style prior induces sparsity

$$p(w_i) = \int p(w_i|\alpha_i)p(\alpha_i)\ d\alpha_i$$



Source: M Tipping (2001). Sparse Bayesian Learning and the Relevance Vector Machine. JMLR.

Sparse Bayesian Learning

Tipping 2001: ARD idea for individual parameters (of an SVM)

$$p(w|\alpha) = \prod_{i=1}^{N} \mathcal{N}(w_i|0, \alpha_i^{-1}) \qquad\qquad p(\alpha) = \prod_{i=1}^{N} \mathrm{Gamma}(\alpha_i|a, b)$$

Observation: ARD-style prior induces sparsity

$$p(w_i) = \int p(w_i|\alpha_i)p(\alpha_i)\ d\alpha_i$$

For $a = b = 0$

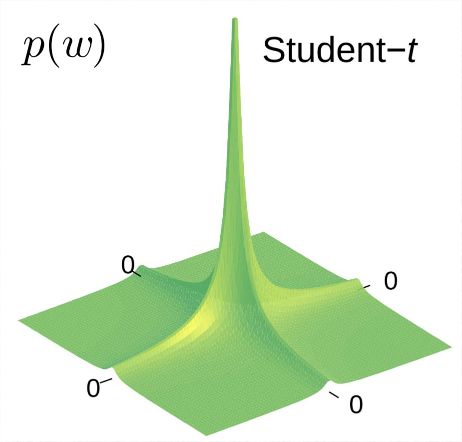
$$p(w_i) \propto \frac{1}{|w_i|}$$

Log-Uniform prior
(scale-invariant, improper)

ARD / Sparse Bayesian Learning

Large body of literature on other choices of sparsity-inducing priors (Log-Uniform, Half-Cauchy, “spike-and-slab”, …)

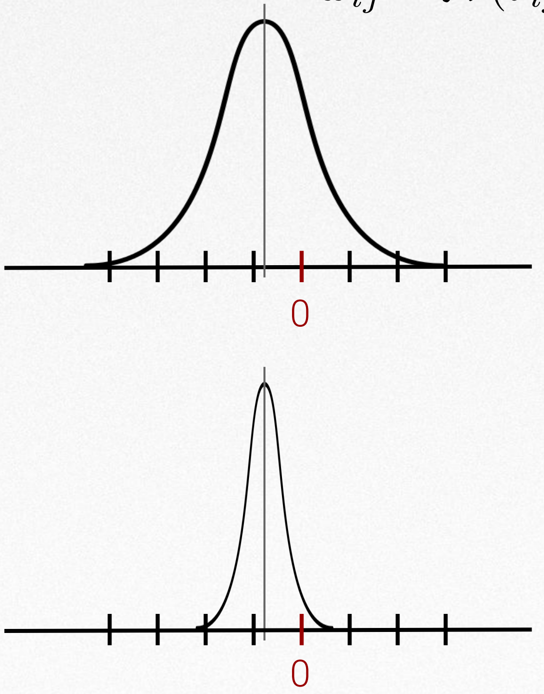
Formal relations to well-known sparsity-inducing regularizers (Dropout, L1-regularization, LASSO, …)



Source: M Tipping (2001). Sparse Bayesian Learning and the Relevance Vector Machine. JMLR.

3 bits

$$w_{ij} \sim \mathcal{N}(\theta_{ij}, \sigma_{ij}^2)$$



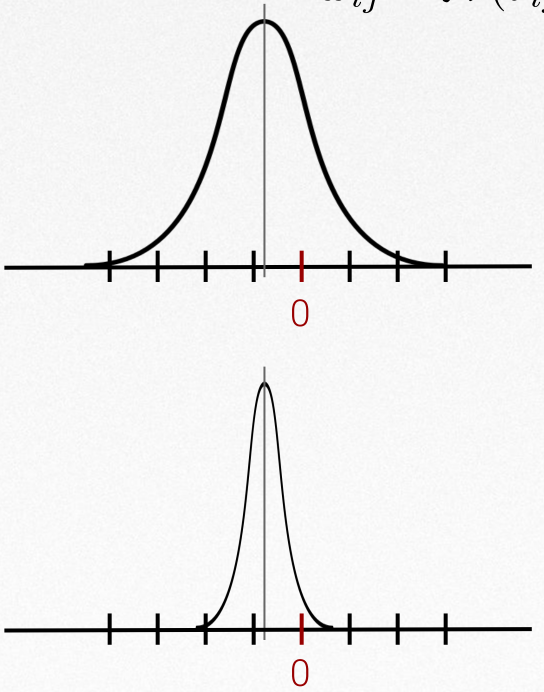
How likely is it to draw the quantized value under the posterior?

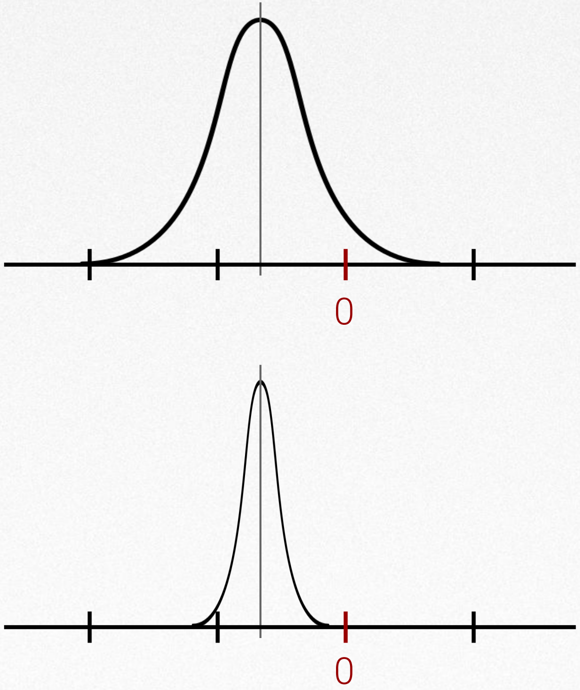
(not the same as squared error from mean, variance is also taken into account)

3 bits

2 bits

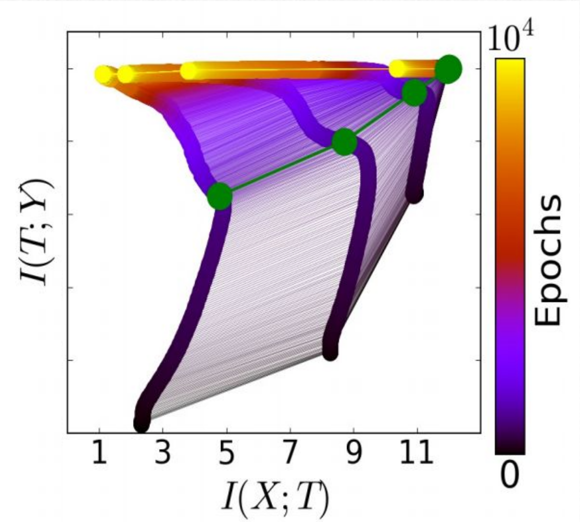
$$w_{ij} \sim \mathcal{N}(\theta_{ij}, \sigma_{ij}^2)$$



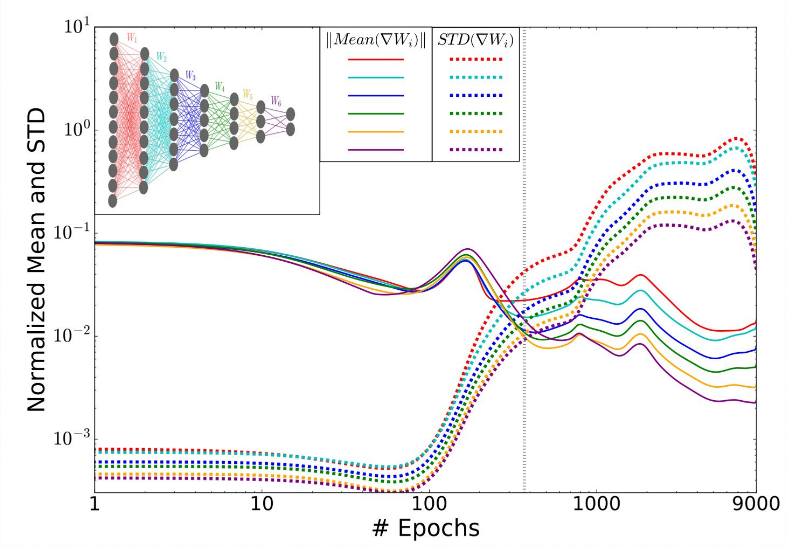


- Why do neural networks generalize at all, given genuine data?
  - SGD might have strong regularizing effects (Info-Bottleneck analysis)

Layers during training and their mutual info with inputs (x-axis) or labels (y-axis)



Weight-gradient-mean and -variance (two distinct phases)



VNQ Details

Variational Network Quantization, Achterhold et al., ICLR 2018

$$\max_{\phi=\{\theta,\sigma,r\}} \mathcal{L}(\phi) = \sum_{n=1}^{N} \mathbb{E}_{q_\phi(w)}[\log p(y_n|x_n, w)] - D_{\mathrm{KL}}(q_\phi(w)||p_r(w))$$

Mixture of shifted log-uniforms

$$p(w_{ij}) \propto \sum_{k=1}^{K} a_k \frac{1}{|w_{ij} - c_k|}$$