Loss surfaces

Dmitry Molchanov

Samsung AI Center

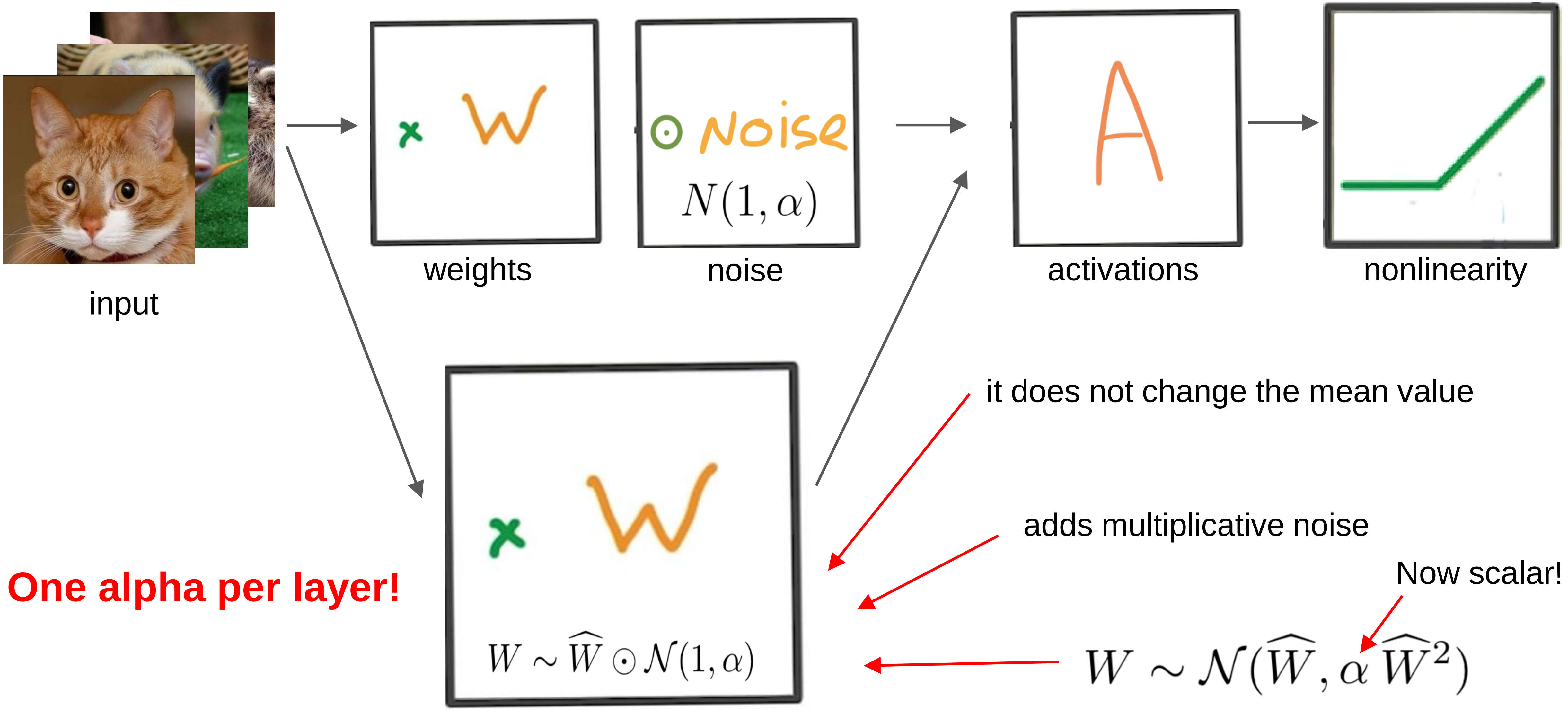
Samsung-HSE Laboratory

DeepBayes

Agenda

- Variance networks
- Mode connectivity
- Stochastic weight averaging

Gaussian Dropout



One alpha per layer!

Variational Dropout

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

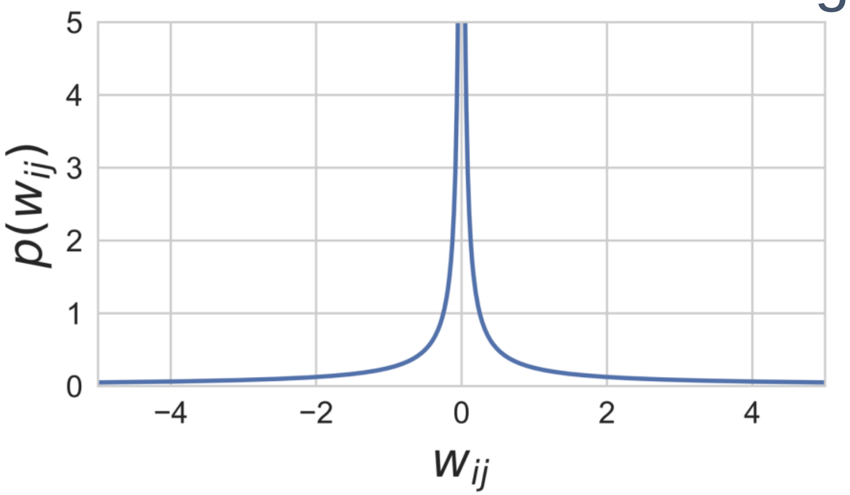
- Posterior distribution

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

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

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

Prior distribution and the KL divergence term



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

$$-D_{KL}(q(w_{ij}\,|\,\hat{w}_{ij}, \alpha)\,|\,p(w_{ij})) =$$
$$0.5 \log \alpha - \mathbb{E}_{\varepsilon \sim \mathcal{N}(1,\alpha)} \log|\epsilon| + \mathrm{C}$$

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

Different prediction schemes

Three prediction schemes:

- Ensemble

$$p(t^*|x^*, x_{train}, t_{train}) \approx \mathbb{E}_{q(w)} p(t^*|x^*, w) \simeq \frac{1}{K}\sum_k p\left(t^*\middle|x^*, w^k\right), w^k \sim q(w)$$

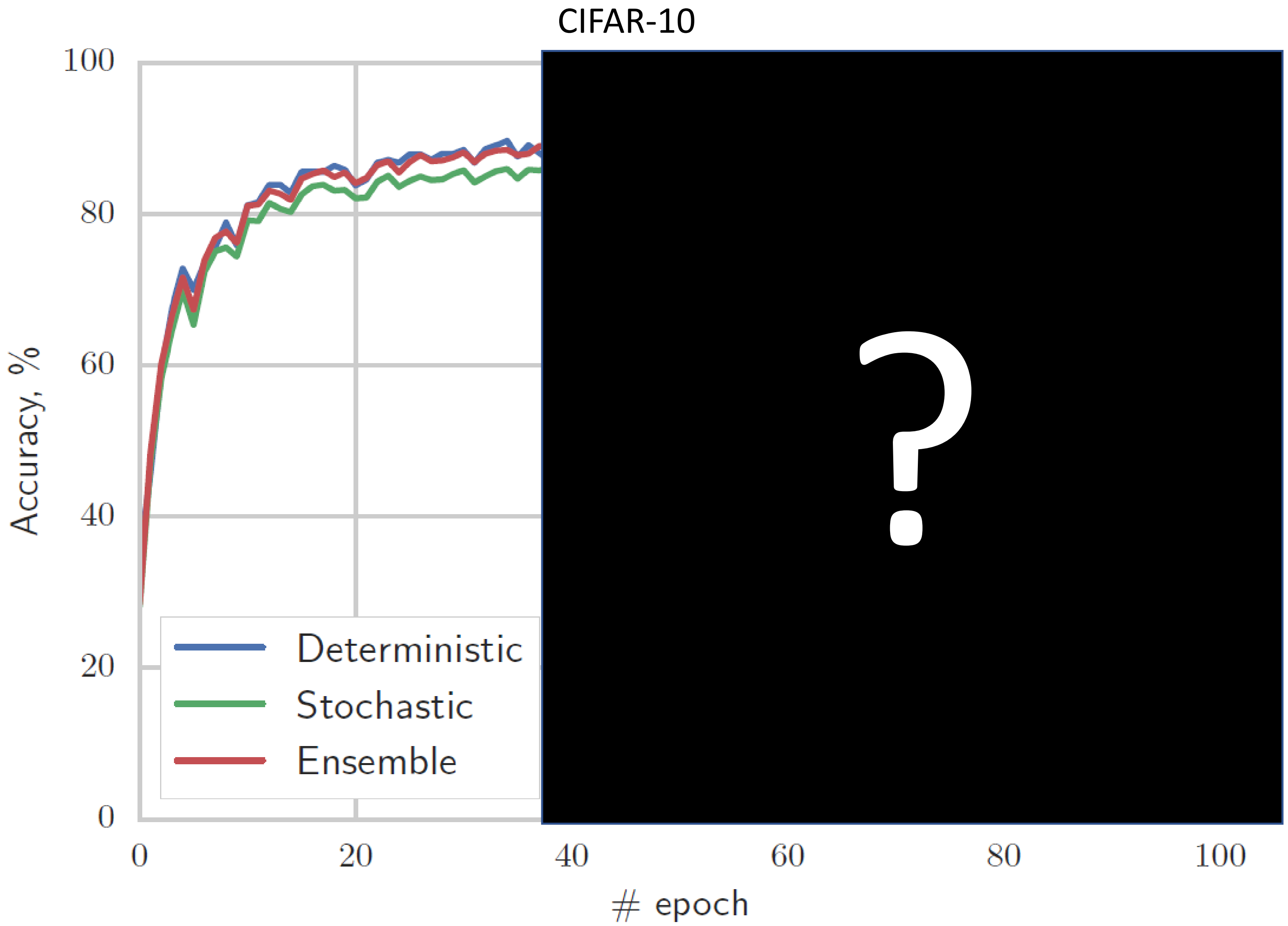
- Stochastic

$$p(t^*|x^*, x_{train}, t_{train}) \approx p(t^*|x^*, \widehat{w}), \widehat{w} \sim q(w)$$

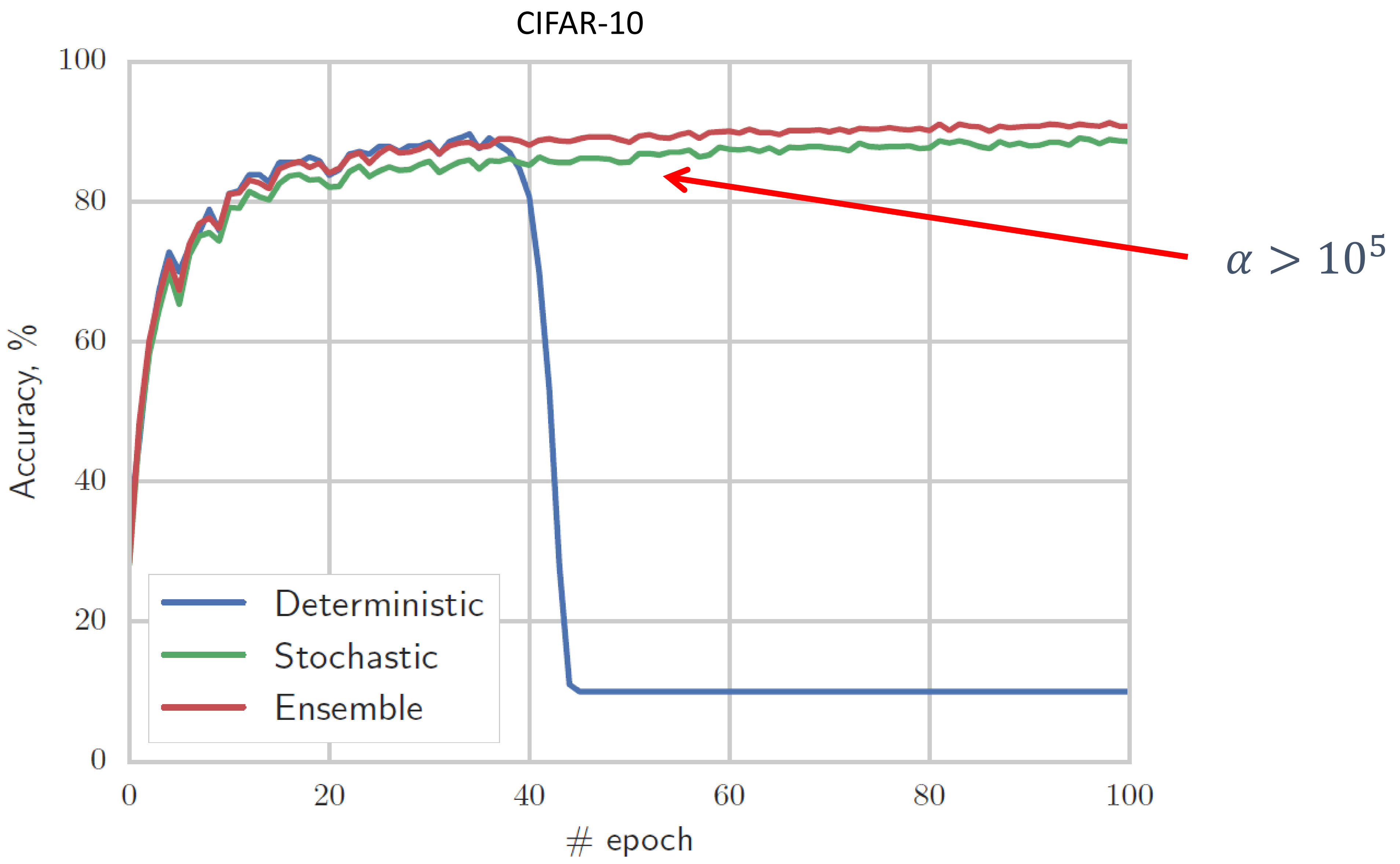
- Deterministic

$$p(t^*|x^*, x_{train}, t_{train}) \approx p(t^*|x^*, \mathbb{E}w)$$

What will happen?



Something weird is happening



Variational dropout converges to a variance network

Let's take a look at alpha $\sigma_{ij}^2 = \alpha \hat{w}_{ij}^2$

$$\frac{1}{\alpha} = \frac{\hat{w}_{ij}^2}{\sigma_{ij}^2} = \text{SNR} = 10^{-5}$$

$$\mathcal{N}(\hat{w}_{ij}, \alpha \hat{w}_{ij}^2) \to \mathcal{N}(0, \sigma_{ij}^2)$$

- We have almost zero Signal-to-Noise Ratio (SNR)
- Means are negligible compared to variances

$$\text{MMD}\left(\mathcal{N}(\hat{w}_{ij}, \alpha \hat{w}_{ij}^2) \middle\| \mathcal{N}(0, \alpha \hat{w}_{ij}^2)\right) \to 0 \quad \text{as} \quad \alpha \to \infty$$

Variance Netrworks

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

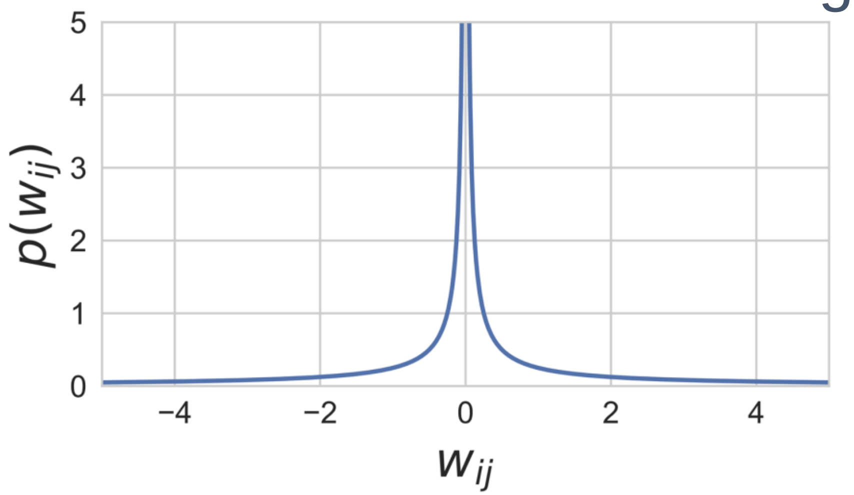
- Posterior distribution

$$w_{ij} = \sigma_{ij} \cdot \varepsilon_{ij}$$

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

$$q(w_{ij}\,|\,\phi_{ij}) = \mathcal{N}(w_{ij}\,|\,0, \sigma_{ij}^2)$$

Prior distribution and the KL divergence term



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

$$-D_{KL}(q(w_{ij}\,|\,0, \sigma_{ij}^2)\,|\,p(w_{ij})) = \text{const}$$

Different parameterizations

- Variance network is actually the best possible variational dropout network!
- Sparse Variational Dropout is just a poor local optimum 😔

	Layer	Neuron	Weight	Additive
ELBO	$\mathbf{-5.9 \cdot 10^2}$	$-7.7 \cdot 10^2$	$-6.4 \cdot 10^4$	$-2.3 \cdot 10^4$
Det. accuracy	11.3	11.3	81.3	96.3
Ens. accuracy	99.2	99.2	99.2	99.2

$$q(w_{ij}) = \mathcal{N}(w_{ij} \,|\, \mu_{ij}, \alpha\mu_{ij}^2) \qquad \text{layer-wise}$$

$$q(w_{ij}) = \mathcal{N}(w_{ij} \,|\, \mu_{ij}, \alpha_j\mu_{ij}^2) \qquad \text{neuron-wise}$$

$$q(w_{ij}) = \mathcal{N}(w_{ij} \,|\, \mu_{ij}, \alpha_{ij}\mu_{ij}^2) \qquad \text{weight-wise}$$

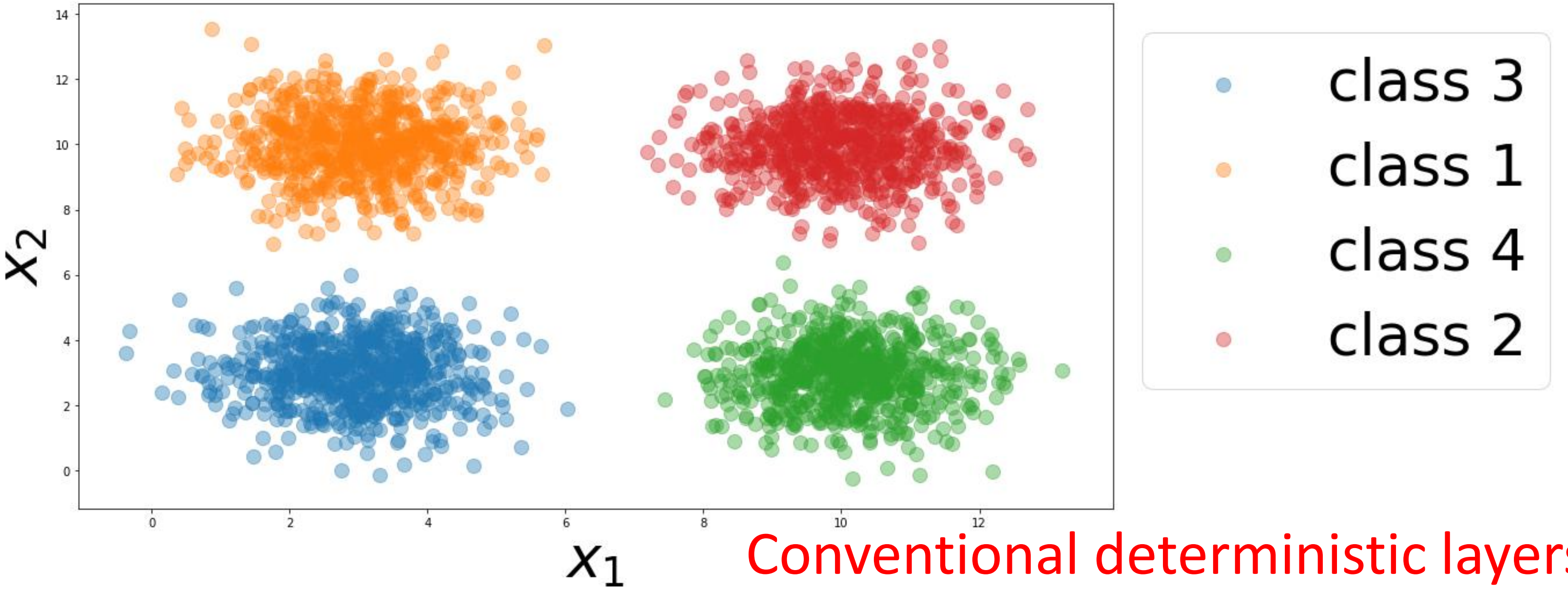
$$q(w_{ij}) = \mathcal{N}(w_{ij} \,|\, \mu_{ij}, \sigma_{ij}^2) \qquad \text{additive}$$

Experiments: classification

Architecture	Dataset	Network	Accuracy (%)		
			Stoch.	Det.	Ens.
LeNet5	MNIST	Dropout	99.1	99.4	99.4
		Variance	95.9	10.1	99.3
VGG-like	CIFAR10	Dropout	91.0	93.1	93.4
		Variance	91.3	10.0	93.4
VGG-like	CIFAR100	Dropout	77.5	79.8	81.7
		Variance	76.9	5.0	82.2

Intuition for variance networks: a toy problem

Dataset:



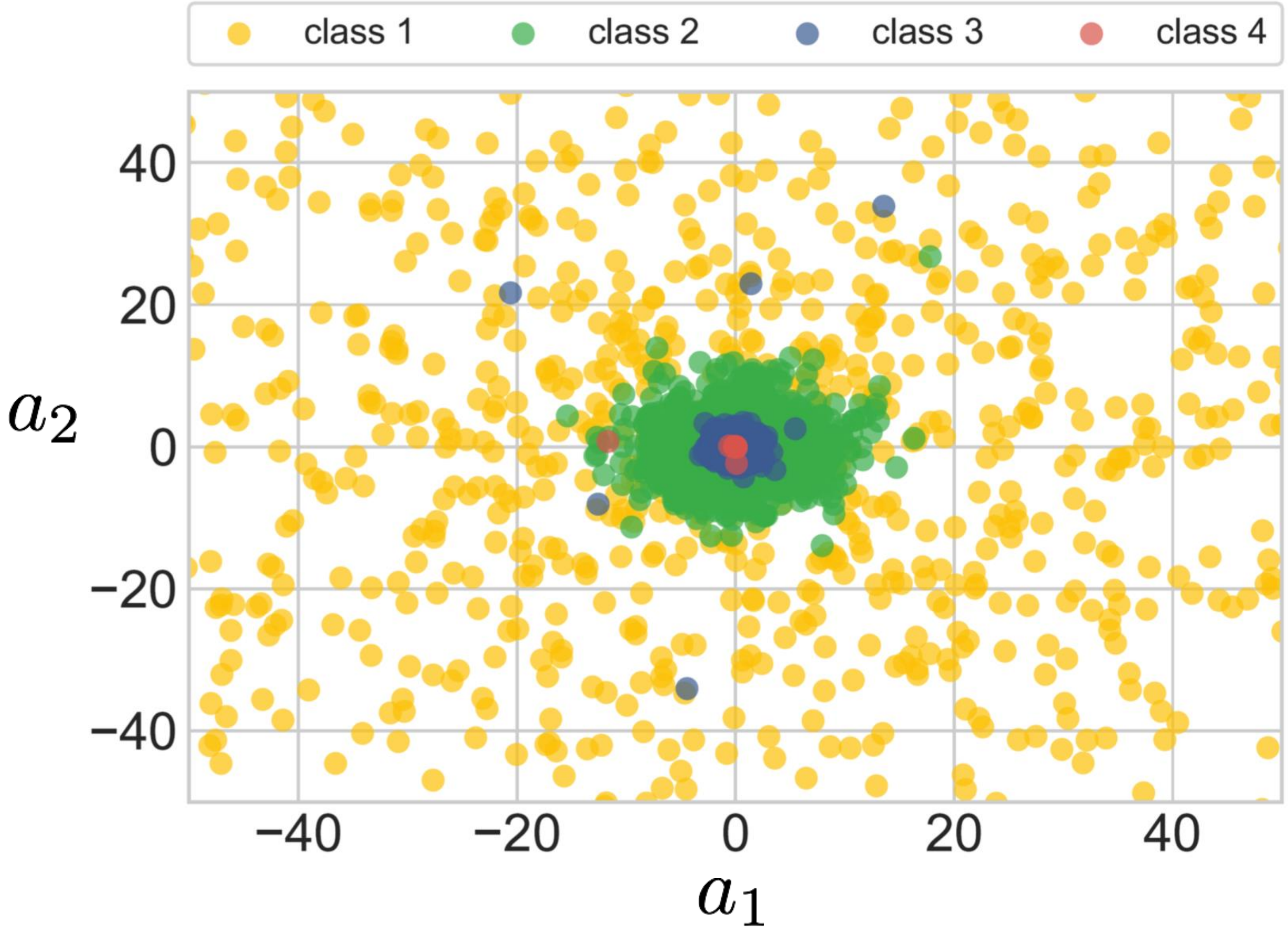
Model:

Conventional deterministic layers

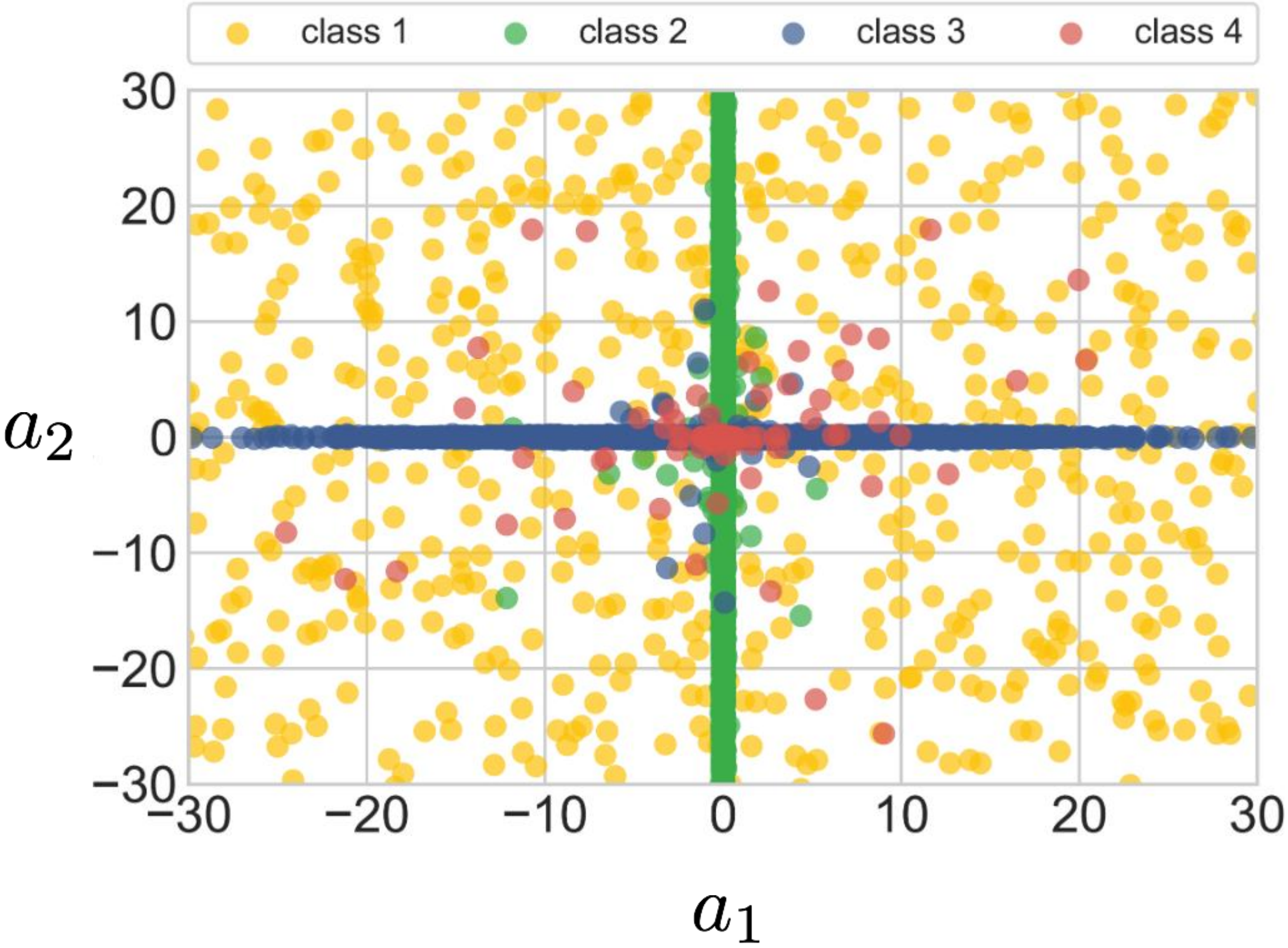
$$x \to \boxed{2 \times d_1} \to \boxed{d_1 \times d_2} \cdots \boxed{d_i \times 2} \to \boxed{2 \times 4} \to y$$

Variance layer with output $\begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$

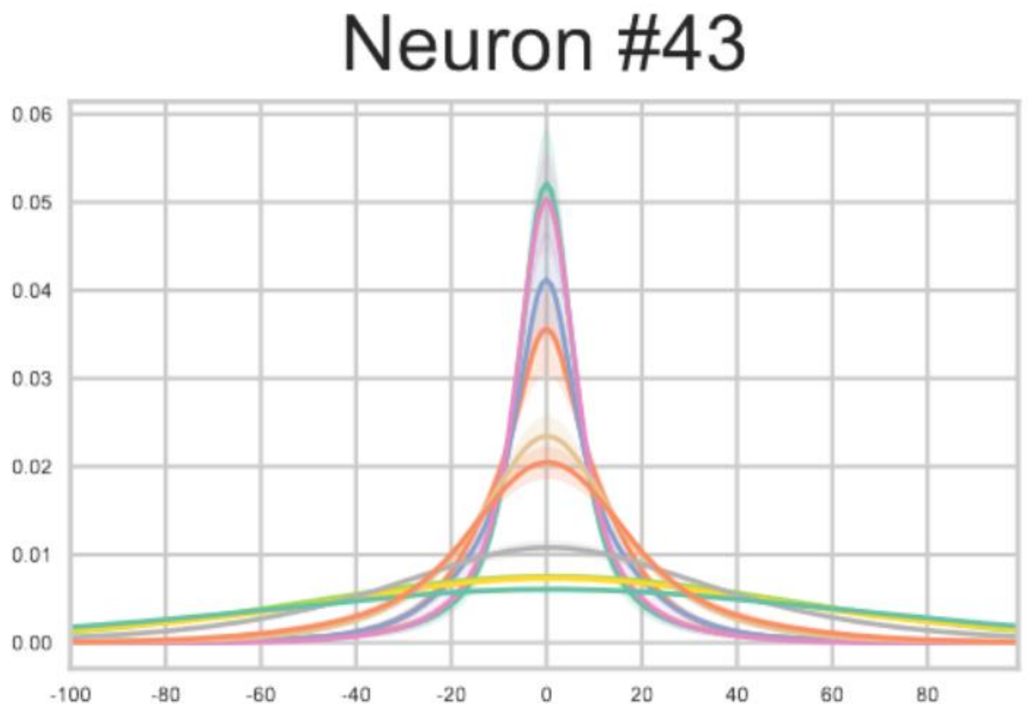
Intuition for variance networks: a toy problem

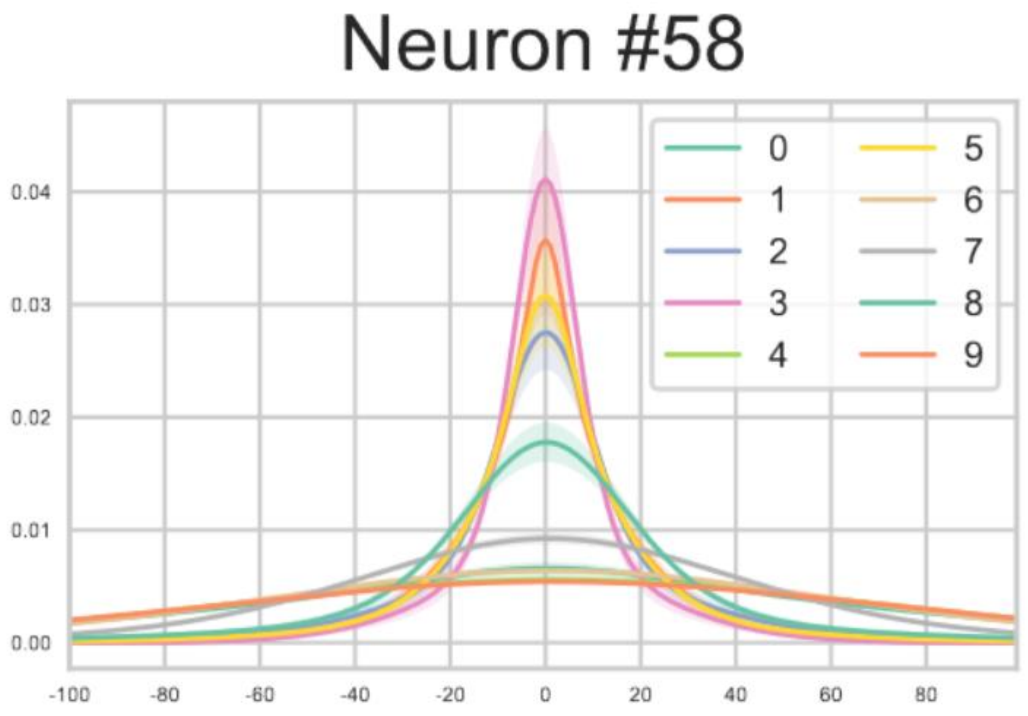


Intuition for variance networks: a toy problem

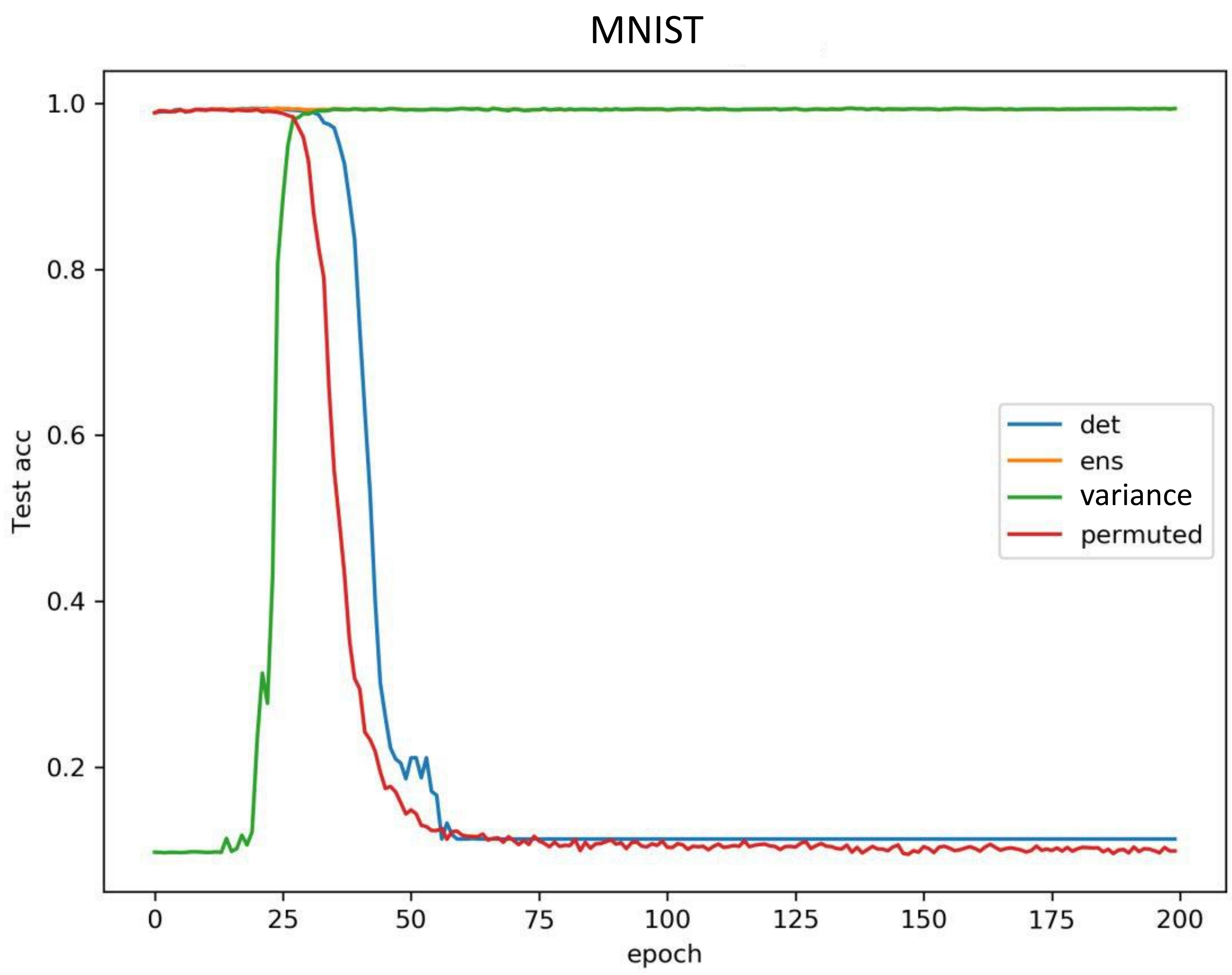


Intuition for variance networks: LeNet-5

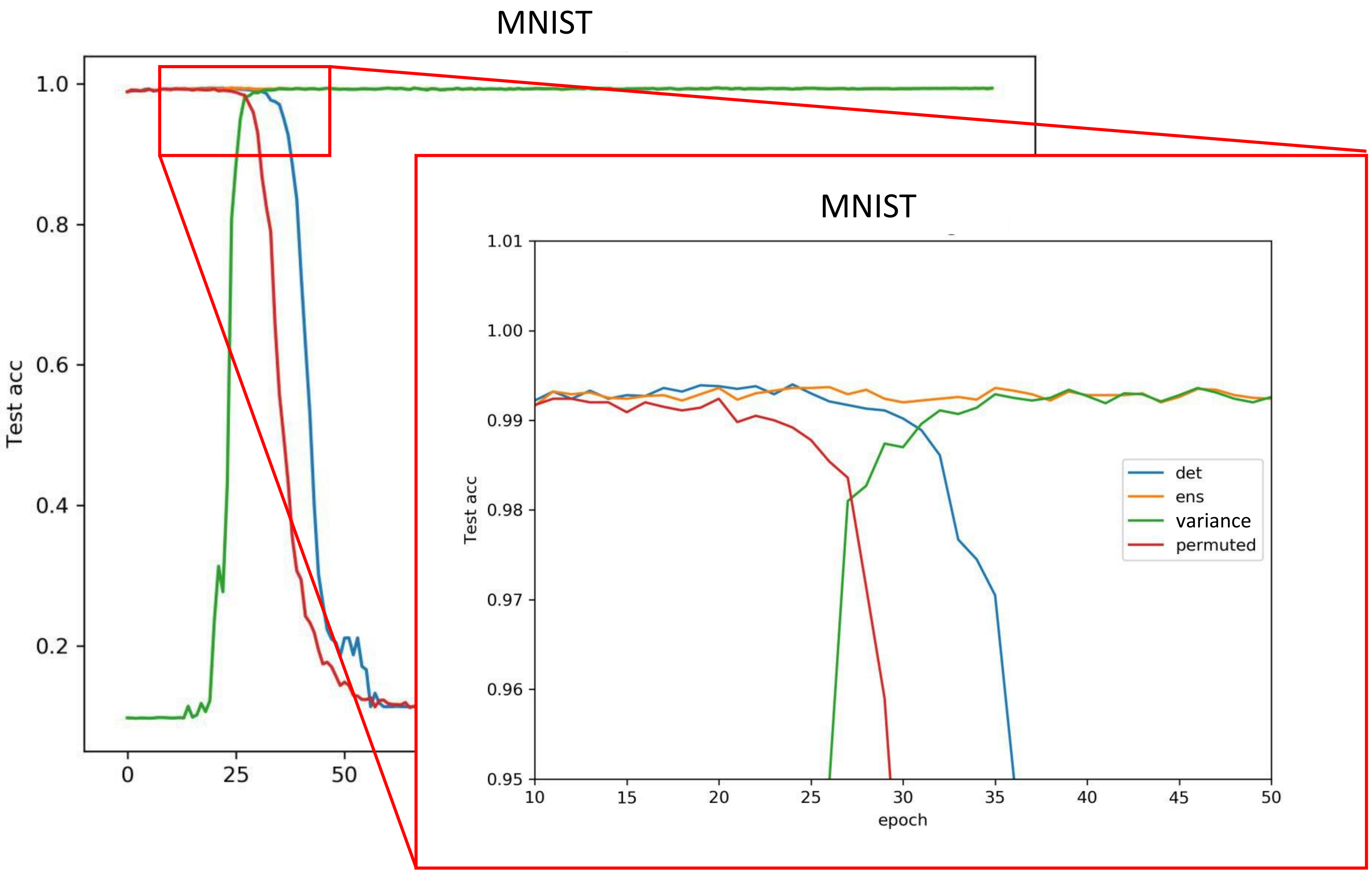




Information flow from means to variances



Information flow from means to variances



Takeaways

- “Weight scaling rule” or deterministic prediction can catastrophically fail
- Random guess quality at test time $\neq$ your model is bad
- More flexible posterior approximation $\neq$ higher ELBO
- Neural networks can withstand seemingly absurd amounts of noise

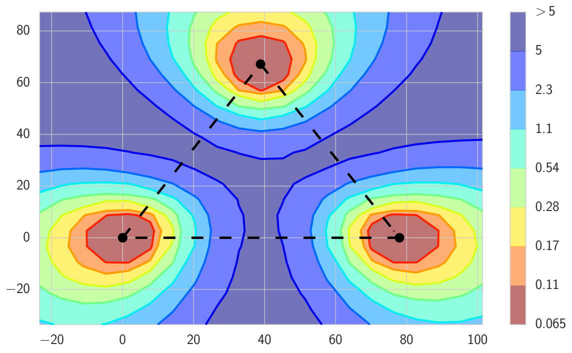
Loss surfaces of neural networks

$$\mathcal{L}(w; X, Y) \to \min_{w}$$

- The loss surfaces of DNNs are highly non-convex and depend on millions of parameters.
- The geometric properties of these loss surfaces are not well understood.
- Even for simple networks, the number of local optima and saddle points is large and can grow exponentially in the number of parameters (Auer et al., 1996; Dauphin et al, 2014).

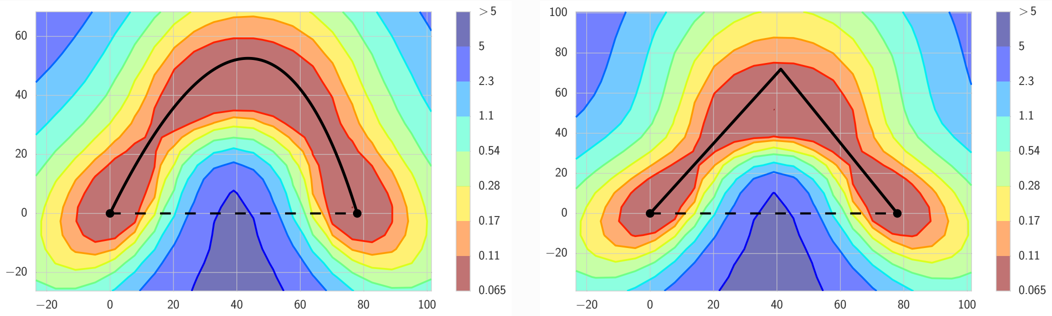
Connecting local minima

The loss is high along a line segment connecting two optima (Goodfellow et al., 2015; Keskar et al., 2017).



The cross-entropy train loss surface in the plane containing weights of three independently trained networks (ResNet-164, CIFAR-100).

Connecting local minima (Garipov et al., 2018)



The cross-entropy train loss surface along the curves connecting two optima (ResNet-164, CIFAR-100).

- Empirically **any pair of “optima” can be connected by a simple path!**
- No "local optima"; we have a manifold of solutions instead.

Connection procedure

Notation

- $\mathcal{L}(w)$ — DNN loss function (e.g. cross-entropy loss)
- $\hat{w}_1, \hat{w}_2 \in \mathbb{R}^{|net|}$ — sets of weights corresponding to two local minima

Parametric curve ϕ_θ with parameters θ:

$$\phi_\theta : [0,1] \to \mathbb{R}^{|net|}, \quad \phi_\theta(0) = \hat{w}_1, \quad \phi_\theta(1) = \hat{w}_2$$

Minimization of the loss along the curve:

$$\hat{\ell}(\theta) = \frac{\int \mathcal{L}(\phi) d\phi}{\int d\phi} = \frac{\int\limits_0^1 \mathcal{L}(\phi(t)) \|\phi'(t)\| dt}{\int\limits_0^1 \|\phi'(t)\| dt} \to \min_\theta$$

Connection procedure

The curve loss could be represented as an expectation:

$$\hat{\ell}(\theta) = \int\limits_0^1 \mathcal{L}(\phi(t)) \underbrace{\left(\frac{\|\phi'(t)\|}{\int\limits_0^1 \|\phi'(s)\| ds} \right)}_{q_\theta(t)} dt = \mathbb{E}_{t \sim q_\theta(t)} \Big[\mathcal{L}(\phi_\theta(t)) \Big] \to \min_\theta$$

- \+ Stochastic optimization could be applied
- – The stochastic gradient w.r.t. θ is intractable in general

A simple heuristic:

$$\ell(\theta) = \int_0^1 \mathcal{L}(\phi_\theta(t)) dt = \mathbb{E}_{t \sim U(0,1)} \Big[\mathcal{L}(\phi_\theta(t)) \Big] \to \min_\theta$$

$$\nabla_\theta \ell(\theta) = \nabla_\theta \mathbb{E}_{t \sim U(0,1)} \mathcal{L}(\phi_\theta(t)) = \mathbb{E}_{t \sim U(0,1)} \nabla_\theta \mathcal{L}(\phi_\theta(t))$$

Example Parameterizations

The trained networks $\hat{w}_1$ and $\hat{w}_2$ serve as the endpoints of the curve.

The parameters θ are trainable parameters of the curve.

Polygonal chain

$$\phi_\theta(t) = \begin{cases} 2\left(t\theta + (0.5 - t)\hat{w}_1\right), & 0 \leq t \leq 0.5 \\ 2\left((t - 0.5)\hat{w}_2 + (1 - t)\theta\right), & 0.5 \leq t \leq 1. \end{cases}$$

Bezier curve

$$\phi_\theta(t) = (1 - t)^2\hat{w}_1 + 2t(1 - t)\theta + t^2\hat{w}_2, \quad 0 \leq t \leq 1.$$

Can add more bends if needed: $\theta = \{\theta^1, \theta^2, \ldots, \theta^n\}$

Batch normalization

Training phase:

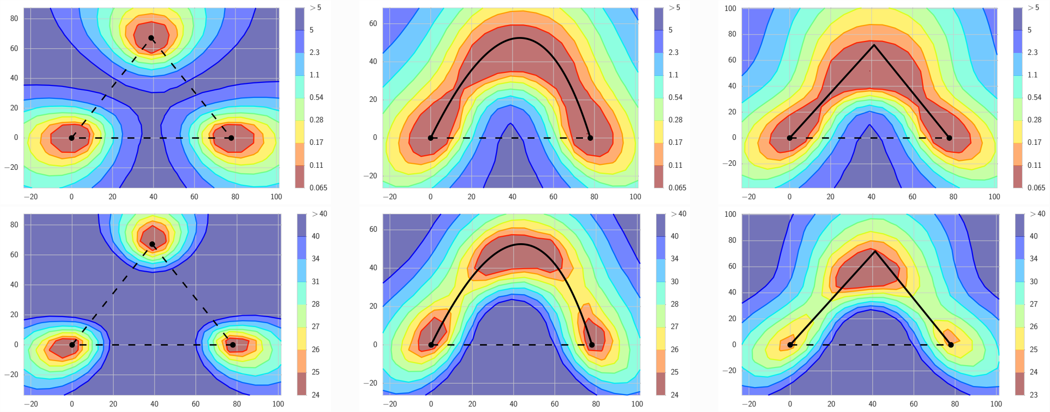
$$\hat{x} = \gamma \frac{x - \mu(x)}{\sigma(x) + \epsilon} + \beta$$

Testing phase:

$$\hat{x} = \gamma \frac{x - \widetilde{\mu}}{\widetilde{\sigma} + \epsilon} + \beta$$

- **During training** for any given t and weights $w = \phi(t)$, we compute $\mu(x)$ and $\sigma(x)$ over mini-batches as usual.
- **During testing** for any given t and weights $w = \phi(t)$ we compute $\widetilde{\mu}$ and $\widetilde{\sigma}$ with one additional pass over the data with the fixed weights, as running averages for such networks are not collected during training.

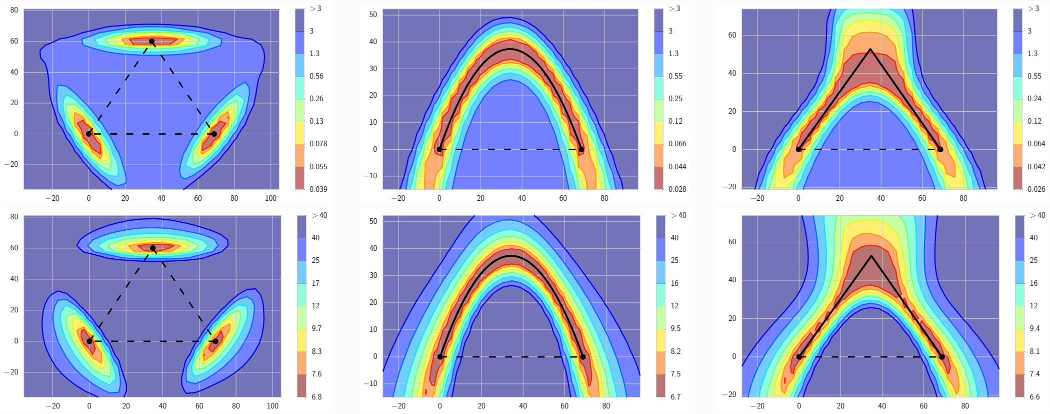
Trained curves (ResNet-164, CIFAR-100)



- Top row: **Train loss**
- Bottom row: **Test error,%**

- Left col: independent optima
- Middle col: Bezier curve
- Right col: polygonal chain

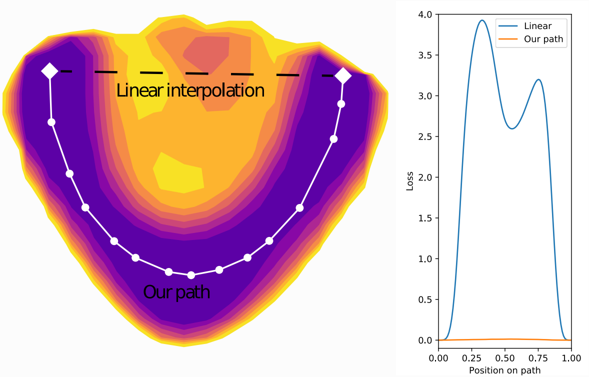
Trained curves (VGG-16, CIFAR-10)



- Top row: **Train loss**
- Bottom row: **Test error,%**
- Left col: independent optima
- Middle col: Bezier curve
- Right col: polygonal chain

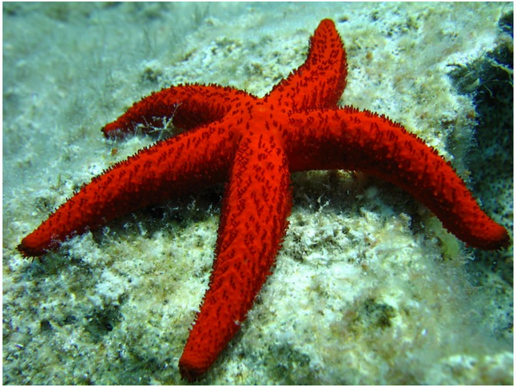
Independent similar results

Essentially No Barriers in Neural Network Energy Landscape
(Draxler et al., 2018)



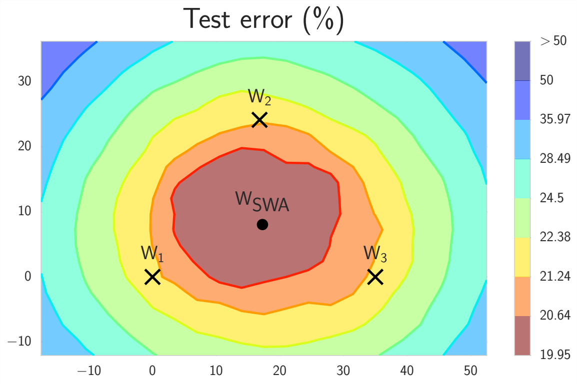
Left: Training loss function surface of DenseNet-40-12 on CIFAR-10 and the minimum energy path. **Right**: Loss along the linear line segment between minima, and along found path.

Vicinity of a "local optima"



- Any local optimum has different functions in its vicinity.
- SGD with a cyclic or a constant learning rate traverses these functions!

Weight averaging (Garipov et al., 2018; Izmailov et al., 2018)



Test error surface for three ensemble elements and their average (ResNet-110, CIFAR-100).

- One could ensemble these points... (FGE, fast geometric ensembling)
- ...or just average the weights! (SWA, stochastic weight averaging)

Stochastic Weight Averaging (SWA)

Run starting from "good enough" model $\hat{w}$

Stochastic Weight Averaging

Require: weights $\hat{w}$, number of iterations n,
cycle length c, LR schedule $\alpha(i)$
Ensure: w_{SWA}
$w \leftarrow \hat{w}$ {Initialize weights with $\hat{w}$}
$w_{\text{SWA}} \leftarrow w$, $n_{\text{models}} \leftarrow 1$
for $i \leftarrow 1, 2, \ldots, n$ **do**
  $\alpha \leftarrow \alpha(i)$ {Calculate LR for the iteration}
  $w \leftarrow w - \alpha \nabla \mathcal{L}_i(w)$ {Stochastic gradient update}
  if $\mathrm{mod}(i, c) = 0$ **then**
    $n_{\text{models}} \leftarrow i/c$ {Number of models}
    $w_{\text{SWA}} \leftarrow \frac{w_{\text{SWA}} \cdot n_{\text{models}} + w}{n_{\text{models}} + 1}$ {Update average}
  end if
end for
Update BatchNorm statistics

Learning rate:

- Cyclical
- Constant

SWA experiments (CIFAR)

Accuracies (%) of SWA, SGD and FGE methods on CIFAR-100 and CIFAR-10 datasets for different training budgets.

			SWA		
DNN (Budget)	SGD	FGE	1 Budget	1.25 Budgets	1.5 Budgets
CIFAR-100					
VGG-16 (200)	72.55 ± 0.10	74.26	73.91 ± 0.12	74.17 ± 0.15	74.27 ± 0.25
ResNet-164 (150)	78.49 ± 0.36	79.84	79.77 ± 0.17	80.18 ± 0.23	80.35 ± 0.16
WRN-28-10 (200)	80.82 ± 0.23	82.27	81.46 ± 0.23	81.91 ± 0.27	82.15 ± 0.27
PyramidNet-272 (300)	83.41 ± 0.21	–	–	83.93 ± 0.18	84.16 ± 0.15
CIFAR-10					
VGG-16 (200)	93.25 ± 0.16	93.52	93.59 ± 0.16	93.70 ± 0.22	93.64 ± 0.18
ResNet-164 (150)	95.28 ± 0.10	95.45	95.56 ± 0.11	95.77 ± 0.04	95.83 ± 0.03
WRN-28-10 (200)	96.18 ± 0.11	96.36	96.45 ± 0.11	96.64 ± 0.08	96.79 ± 0.05
ShakeShake-2x64d (1800)	96.93 ± 0.10	–	–	97.16 ± 0.10	97.12 ± 0.06

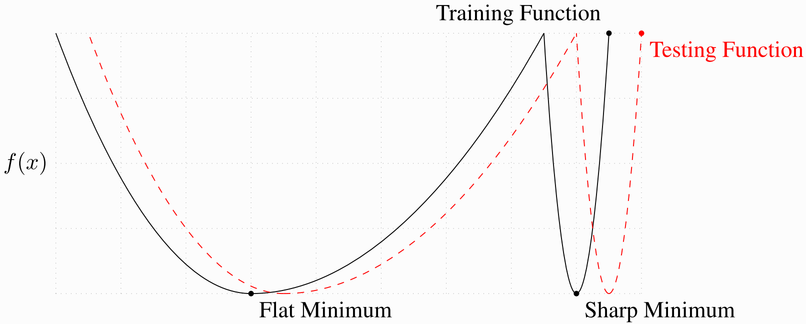
SWA experiments (ImageNet)

Accuracies (%) on ImageNet dataset for SWA and SGD with different architectures.

		SWA	
DNN	SGD	5 epochs	10 epochs
ResNet-50	76.15	76.83 ± 0.01	76.97 ± 0.05
ResNet-152	78.31	78.82 ± 0.01	78.94 ± 0.07
DenseNet-161	77.65	78.26 ± 0.09	78.44 ± 0.06

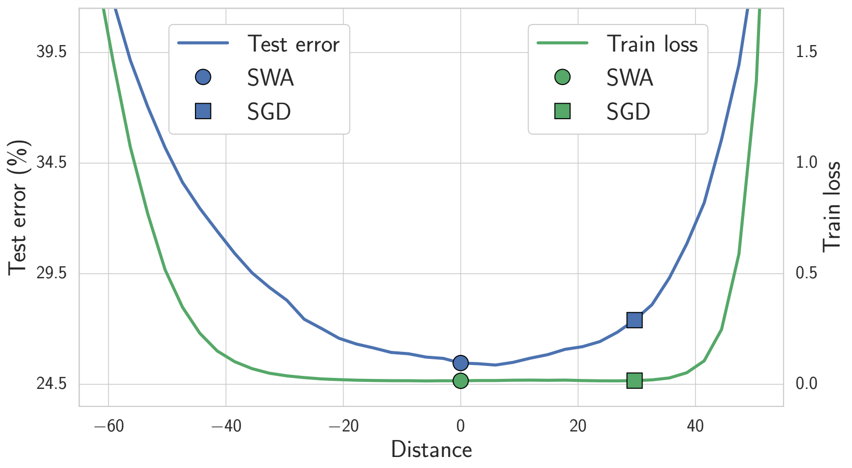
Optima Width

- (Keskar et al., 2017): flat minima lead to strong generalization, while sharp minima generalize poorly.



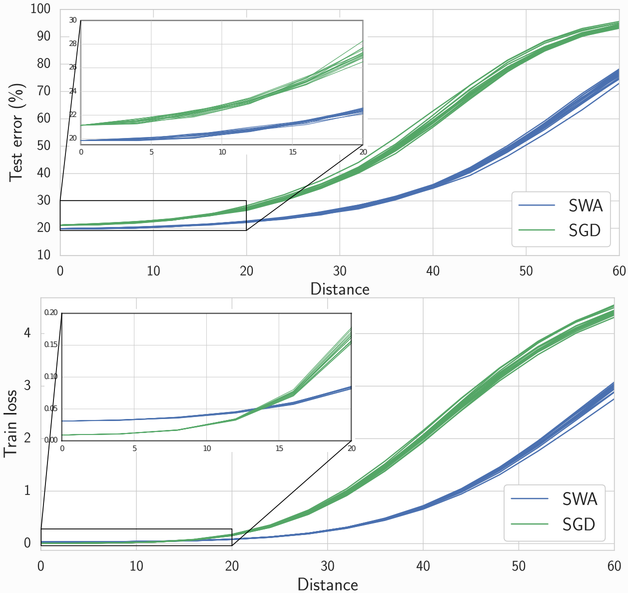
- Generally accepted intuition
  (but might be misleading (Dinh et al., 2017)).

Optima Width (CIFAR-100, VGG-16)



L_2-regularized cross-entropy train loss and test error as a function of a point on the line connecting w_{SWA} and w_{SGD}.

Optima Width along random directions



Applications of stochastic weight averaging

SWA turns out to be useful in a lot of different problems:

- Classification (Izmailov et al., 2018)
- GAN training (Yazici et al., 2018)
- Reinfocement learning (Nikishin et al., 2018)
- Semi-supervised learning (Athiwaratkun et al., 2018)
- Uncertainty estimation (Maddox et al., 2019)
- Low-precision training (Yang et al., 2019)

Links

- (Neklyudov et al., 2018) Variance Networks: When Expectation Does Not Meet Your Expectations
  arxiv.org/abs/1803.03764
- (Garipov et al., 2018) Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs
  arxiv.org/abs/1802.10026
- (Izmailov et al., 2018) Averaging Weights Leads to Wider Optima and Better Generalization
  arxiv.org/abs/1803.05407
- (Yang et al., 2019) SWALP : Stochastic Weight Averaging in Low-Precision Training
  https://arxiv.org/abs/1904.11943
- (Athiwaratkun et al., 2019) There Are Many Consistent Explanations of Unlabeled Data: Why You Should Average
  https://arxiv.org/abs/1806.05594
- (Nikishin et al., 2018) Improving Stability in Deep Reinforcement Learning with Weight Averaging
  https://izmailovpavel.github.io/files/swa_rl/paper.pdf
- (Maddox et al., 2019) A Simple Baseline for Bayesian Uncertainty in Deep Learning
  https://arxiv.org/abs/1902.02476