Loss surfaces, fast ensembling, and weight averaging of DNNs

Timur Garipov* Pavel Izmailov* Dmitrii Podoprikhin*
Dmitry Vetrov Andrew Gordon Wilson

*equal contirbution

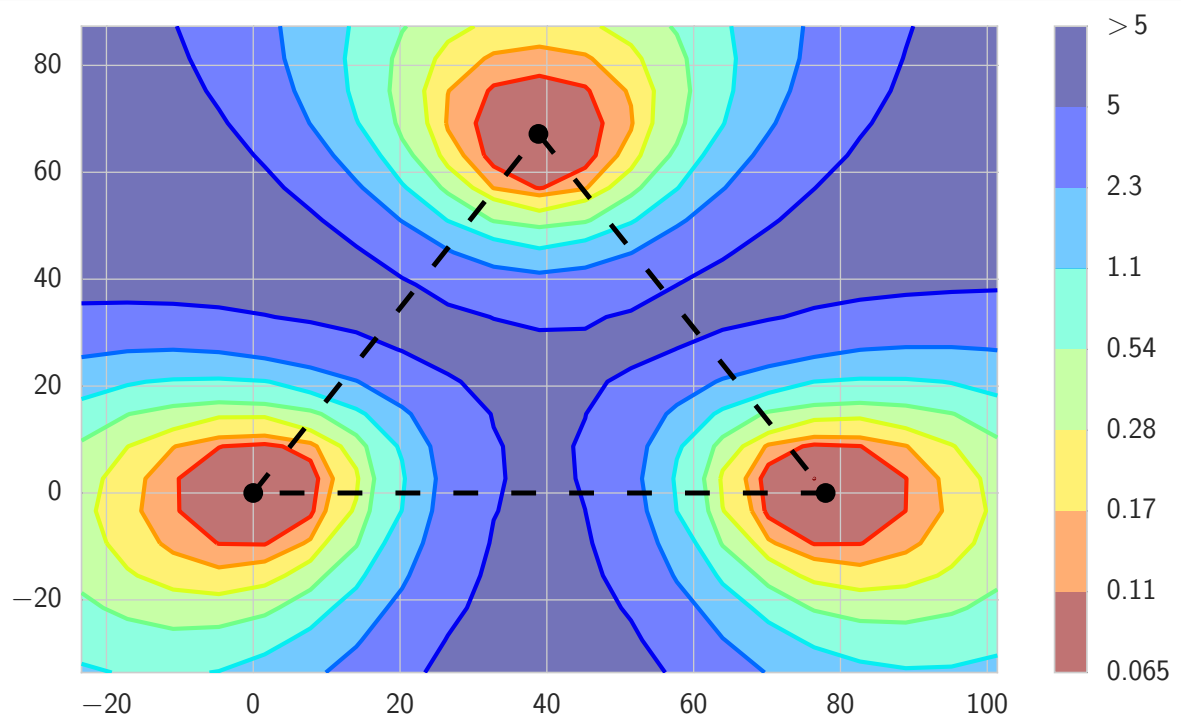
March 23, 2018

Loss surfaces of neural networks

- 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).

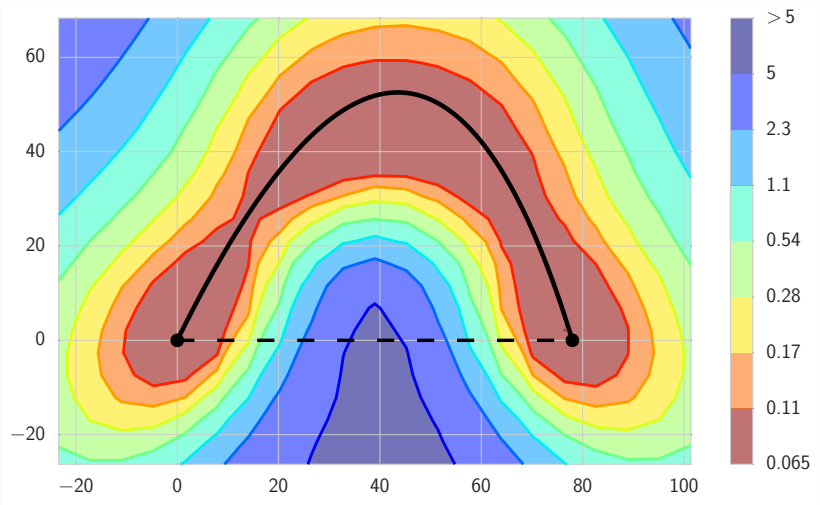
Connecting local minima

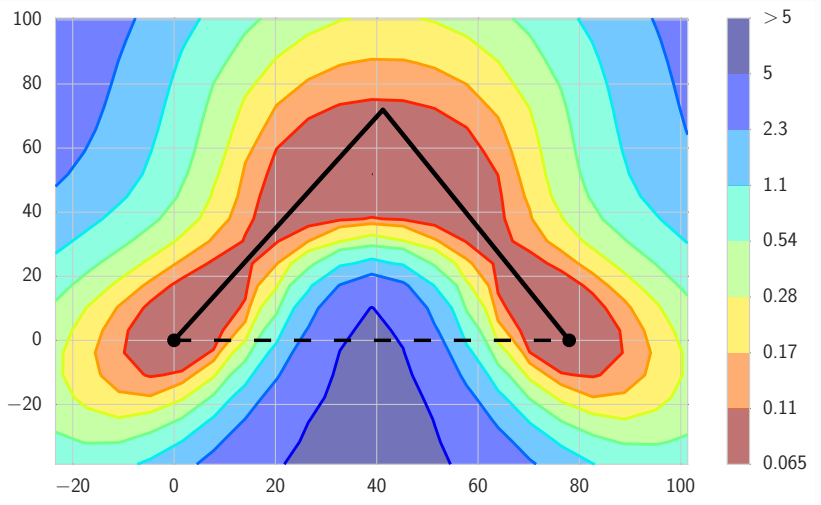
The loss is high along a line segment connecting two optima (Goodfellow et al., 2015; Keskar et al., 2017).



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

Connecting local minima





The cross-entropy train loss surface in the planes containing Bezier curve and polygonal chain connecting the two optima from the previous slide.

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:

$$\hat{\ell}(\theta) = \frac{\int \mathcal{L}(\phi) d\phi}{\int d\phi} = \frac{\int_0^1 \mathcal{L}(\phi(t)) \|\phi'(t)\| dt}{\int_0^1 \|\phi'(t)\| dt} \to \min_\theta$$

Connection procedure

The curve loss could be represented as an expectation:

$$\hat{\ell}(\theta) = \int_0^1 \mathcal{L}(\phi(t)) \underbrace{\left(\frac{\|\phi'(t)\|}{\int_0^1 \|\phi'(s)\| ds} \right)}_{q_\theta(t)} dt = \mathbb{E}_{t \sim q_\theta(t)} \Big[\mathcal{L}(\phi_\theta(t)) \Big] \to \min_\theta$$

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

We use simplified curve loss:

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

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

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

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

Bezier curve

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

Remark: these formulas naturally generalize for n bends
$\theta \;=\; \{w_1, w_2, \ldots, w_n\}$

Batch normalization

Training phase:

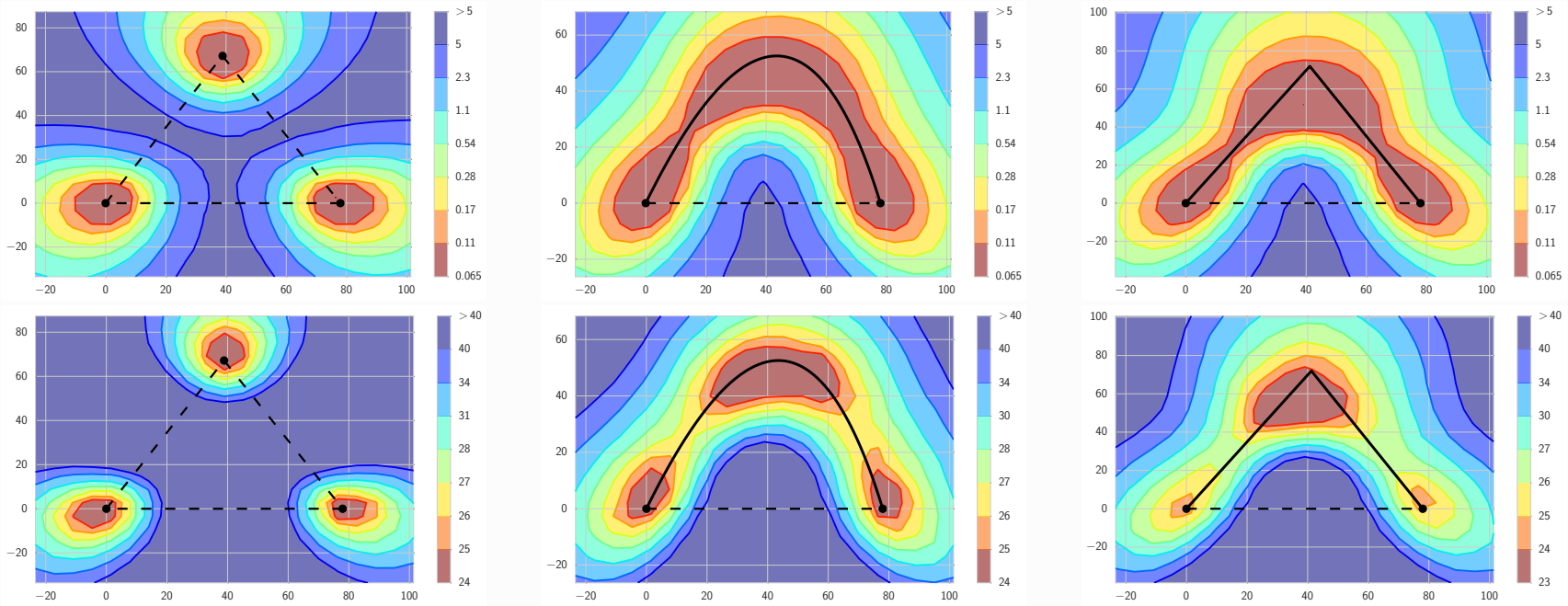
$$\hat{x} = \gamma \frac{x - \mu(x)}{\sigma(x) + \epsilon} + \beta$$

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.

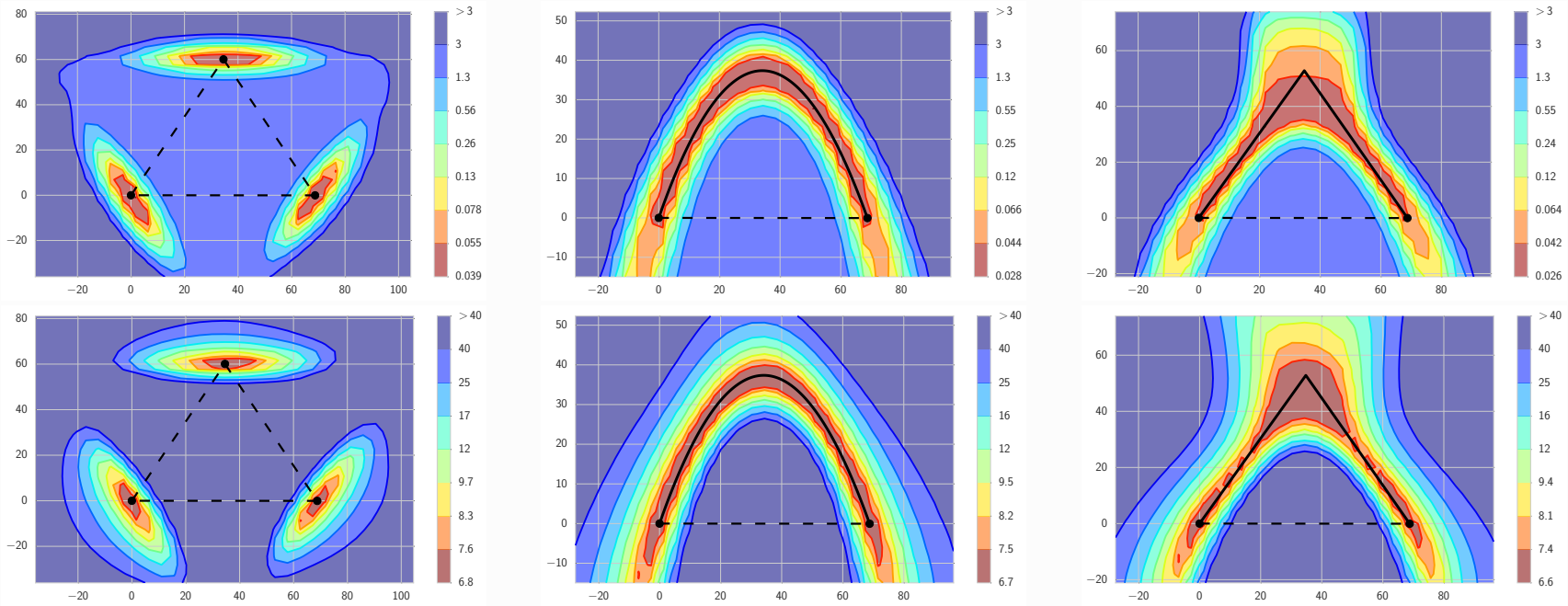
Trained curves (ResNet-164, CIFAR-100)



- 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)

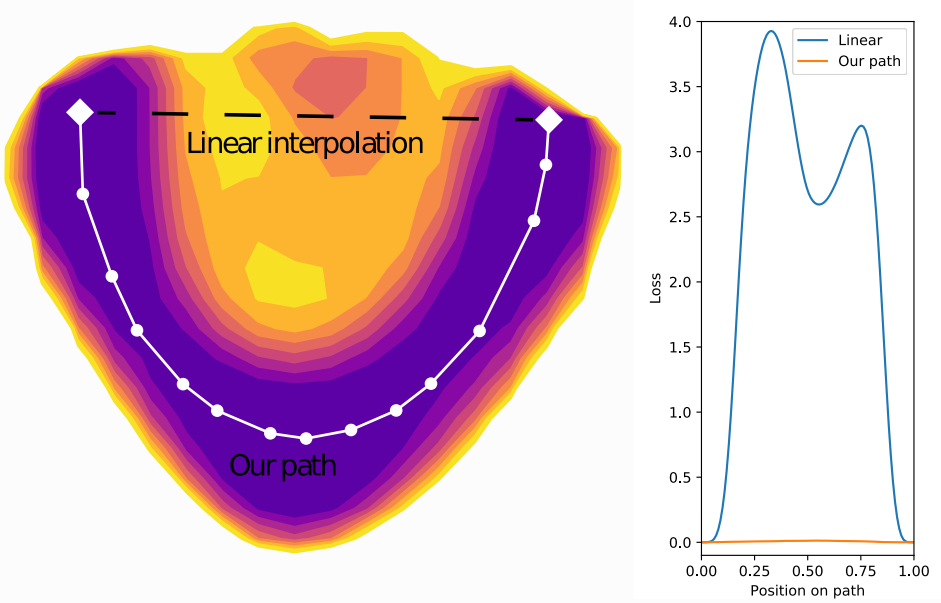


- Top row: **Train loss**
- Bottom row: **Test error,%**

- 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)



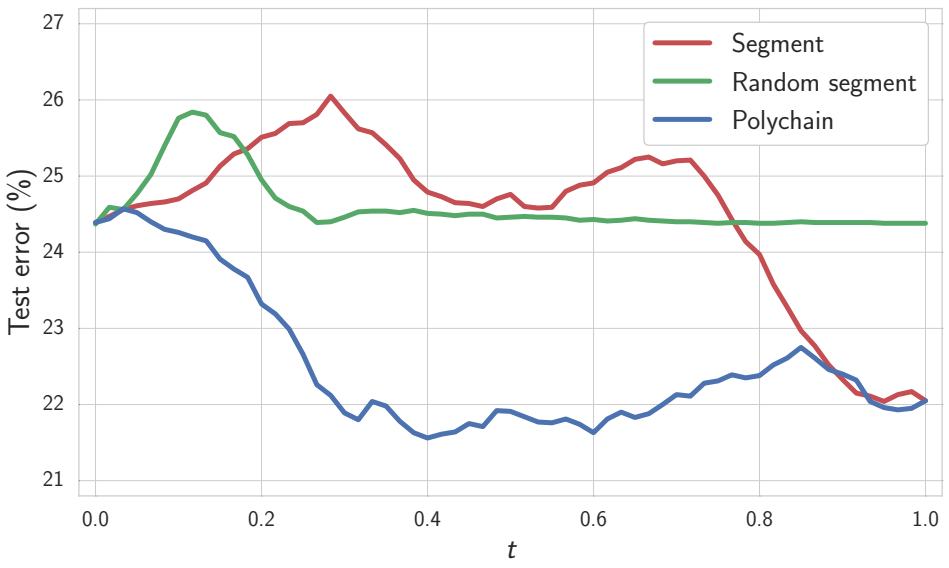
Left: Training loss function surface of DenseNet-40-12 on CIFAR-10 and the minimum energy path. The plane is spanned by the two minima and the mean of the nodes of the path. **Right**: Loss along the linear line segment between minima, and along found path.

Curve Ensembling

- For curves with one bend the total number of parameters is equal to the number of parameters of three independent networks
- Sample several times $t \sim U[0, 1]$ and construct ensemble from networks with parameters $\phi_\theta(t)$.
- The accuracy of such an ensemble is close to accuracy of the ensemble of three independent networks.
- ResNet-164, CIFAR-100. Three independent networks: 21.01%, ensemble along the curve (50 points): 21.07%

Diversity of points on a curve

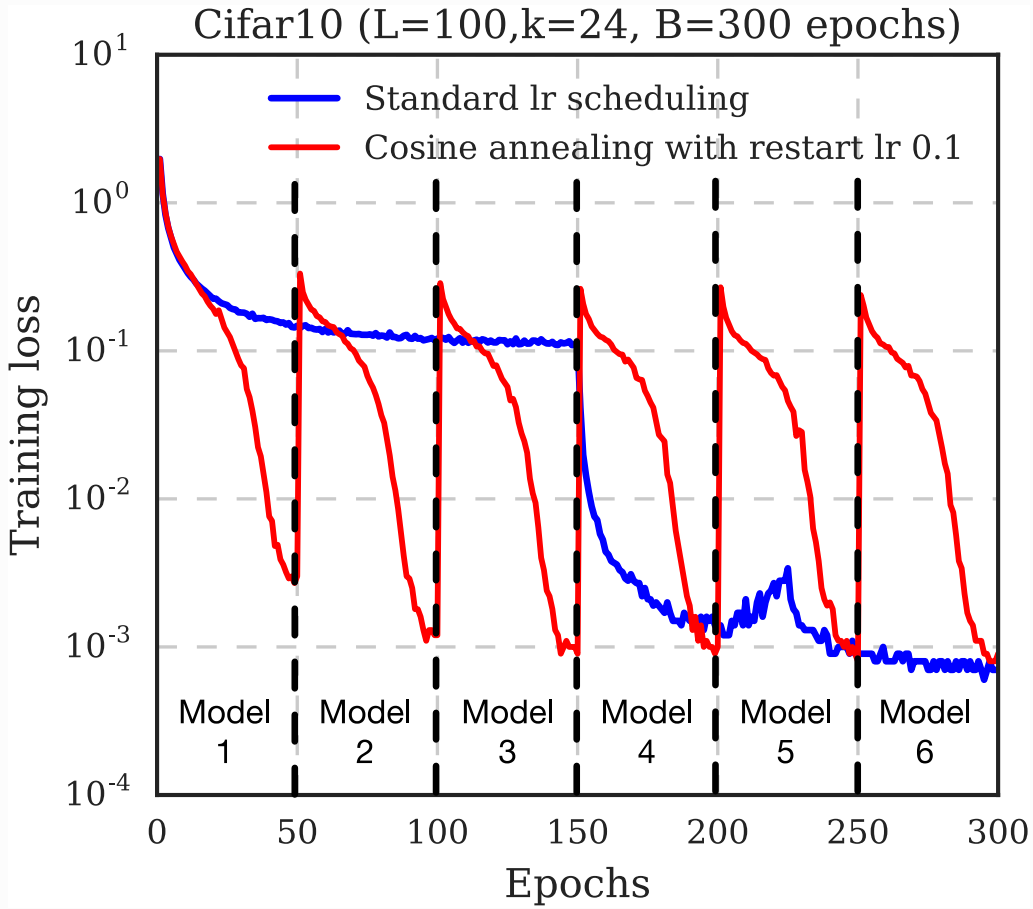
Error of the two-network ensemble (w_0, w_t) consisting of $w_0 = \phi_\theta(0)$ and $w_t = \phi_\theta(t)$ points on the curve (CIFAR-100, ResNet-164).



Segment: $\phi_\theta(\cdot)$ is a line segment connecting two modes found by SGD.
Polychain: $\phi_\theta(\cdot)$ is a polygonal chain connecting the same endpoints.
Random segment: $\phi_\theta(\cdot)$ is a straight line following from the endpoint of the curve in a random direction.

SnapShot Ensembles (SSE) (Huang et al., 2017)

Training an ensemble in time needed for training a single model:

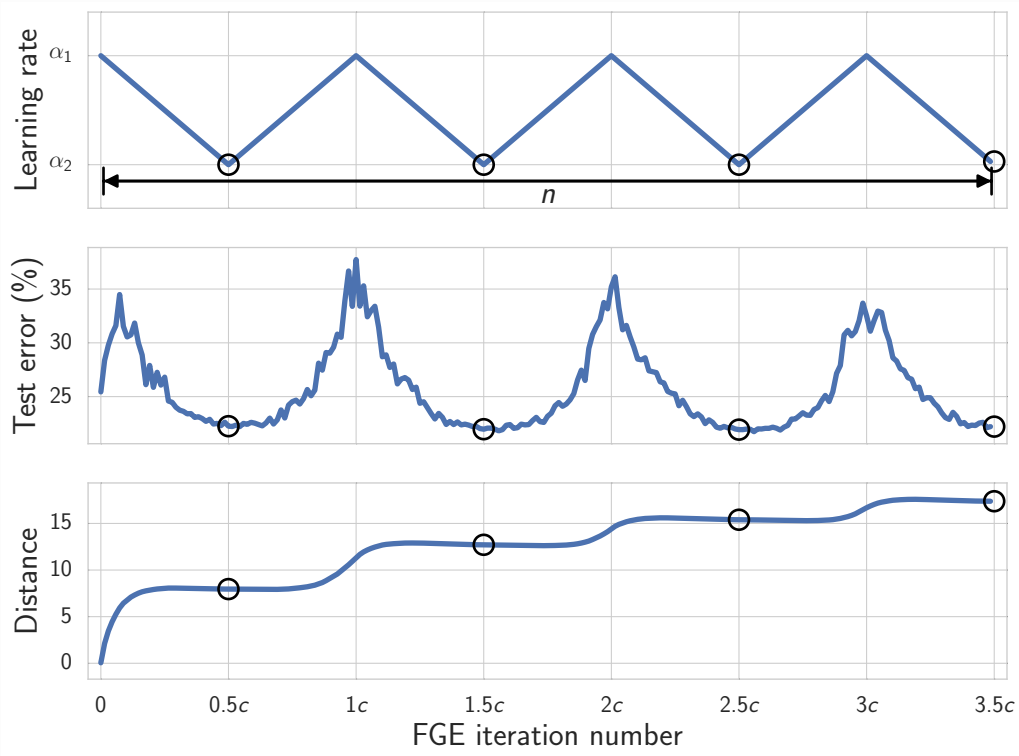


- Use cyclical learning rate schedule
- Collect a snapshot of the network at the end of each cycle
- Form the ensemble of M last snapshots

Fast Geometric Ensembling (FGE)

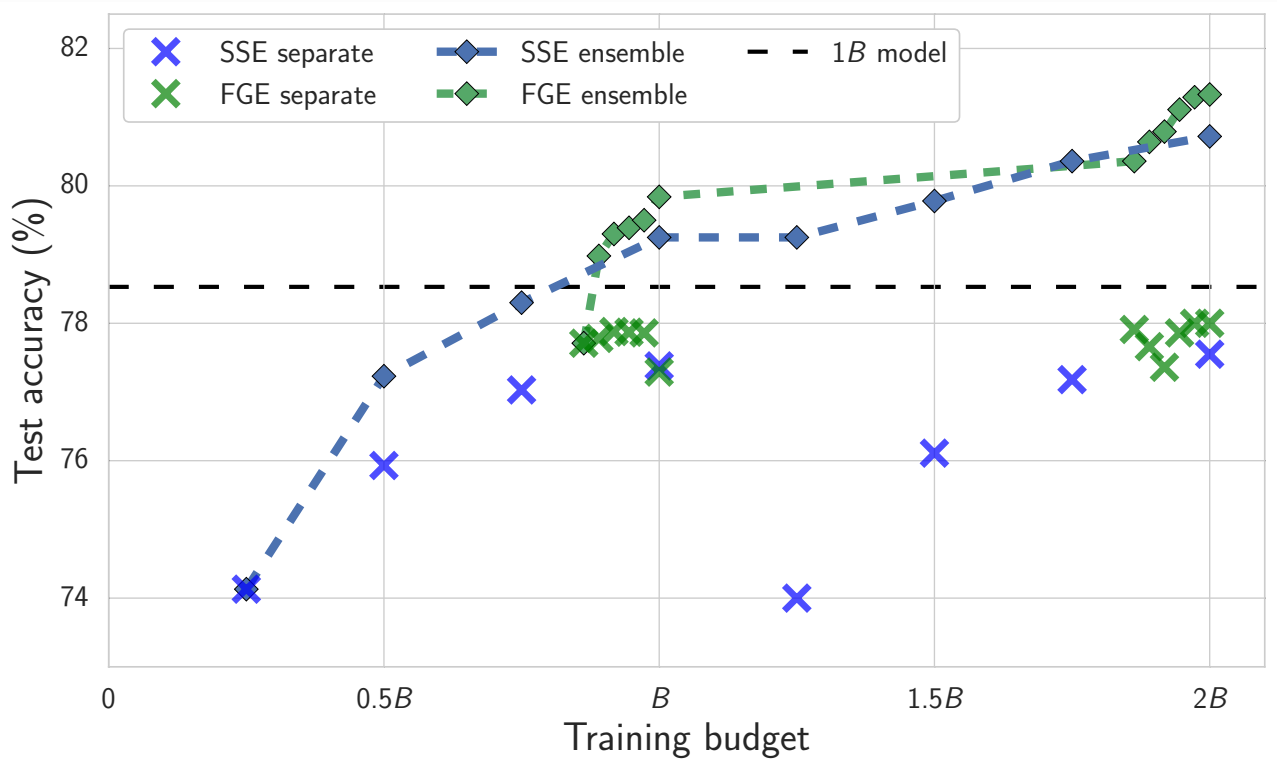
Proposed changes:

- smaller LR (0.005 instead of 0.1)
- shorter cycles (4 epochs instead of 30+)



Top: learning rate, **Middle**: test error, **Bottom:** distance from the initial point as a functions of iteration (ResNet-110, CIFAR-100). Circles indicate the times when we save snapshots.

SSE and FGE comparison



Ensemble performance of FGE and SSE as a function of training time (ResNet-110, CIFAR-100, $B = 150$ epochs).

Crosses represent the performance of separate "snapshot" models, and **diamonds** show the performance of the ensembles constructed of all models available by the given time.

Experiments

		CIFAR-100			CIFAR-10		
DNN (Budget)	method	1 Budget	2 Budgets	3 Budgets	1 Budget	2 Budgets	3 Budgets
VGG-16 (200)	Ind	27.4	25.28	24.45	6.81	5.89	5.9
	SSE	26.4	25.16	24.69	6.5	6.19	5.95
	FGE	**25.74**	**24.11**	**23.54**	**6.48**	**5.82**	**5.66**
ResNet-110 (150)	Ind	21.37	19.04	18.59	4.7	**4.1**	**3.77**
	SSE	20.75	19.28	18.91	4.66	4.37	4.3
	FGE	**20.16**	**18.67**	**18.21**	**4.55**	4.21	3.98
WRN-28-10 (200)	Ind	19.1	17.48	17.01	3.74	3.4	**3.31**
	SSE	17.78	17.3	16.97	3.74	3.54	3.55
	FGE	**17.73**	**16.95**	**16.88**	**3.64**	**3.38**	3.52

Error rates (%) for different ensembling techniques and training budgets.

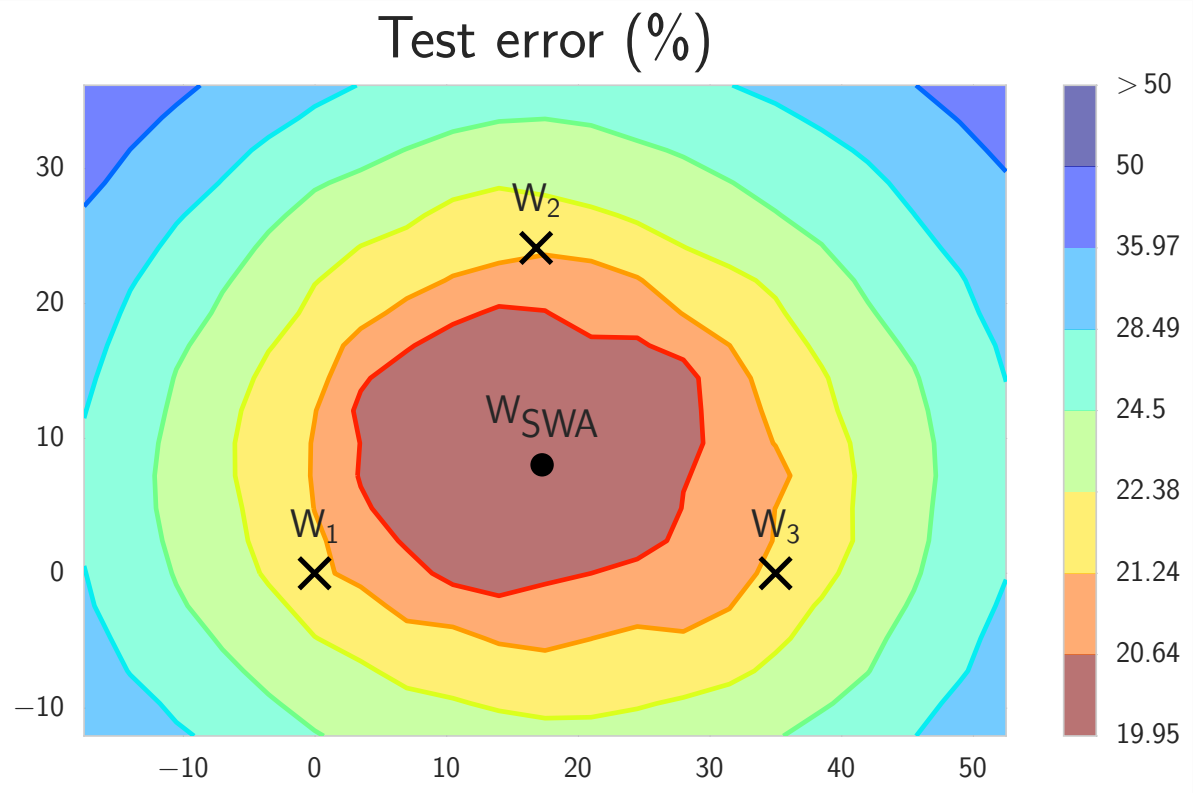
Budget is the number of epochs required to train a single model

SSE – SnapShot Ensemble (Huang et al., 2017)

Ind – independent networks

ImageNet ResNet-50: 23.87, FGE (add. 10 epochs): 23.31, SSE: 23.33.

Weight averaging



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

Stochastic Weight Averaging (SWA)

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

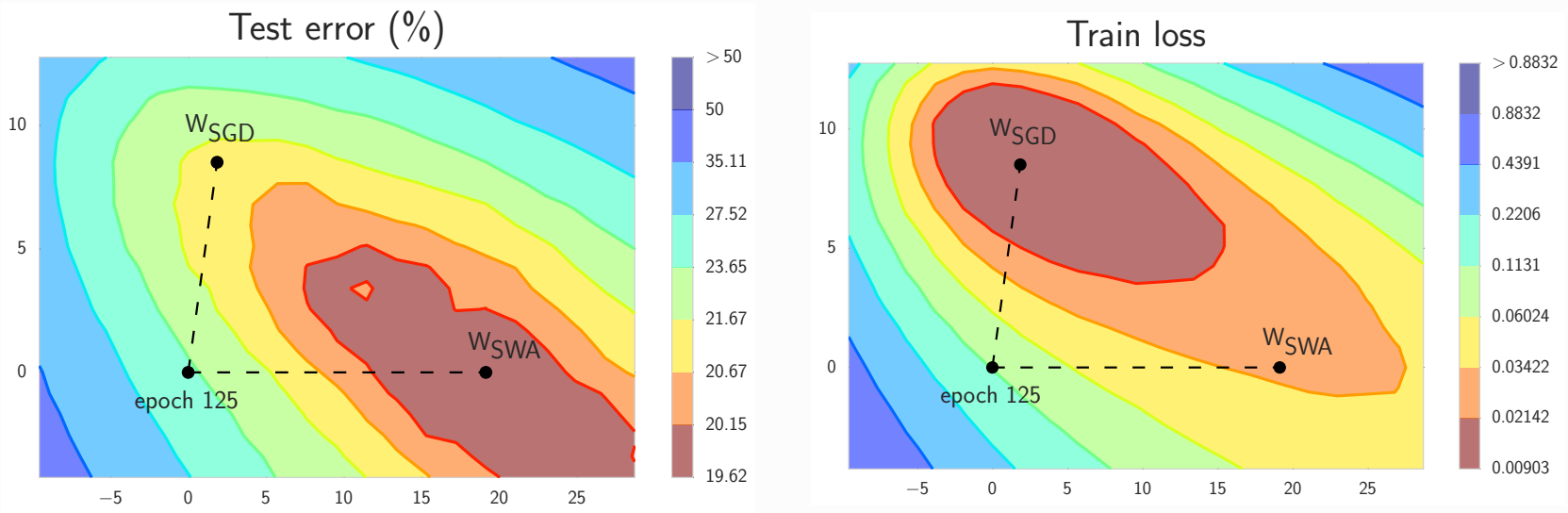
Stochastic Weight Averaging

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

Learning rate:

- Cyclical
- Constant

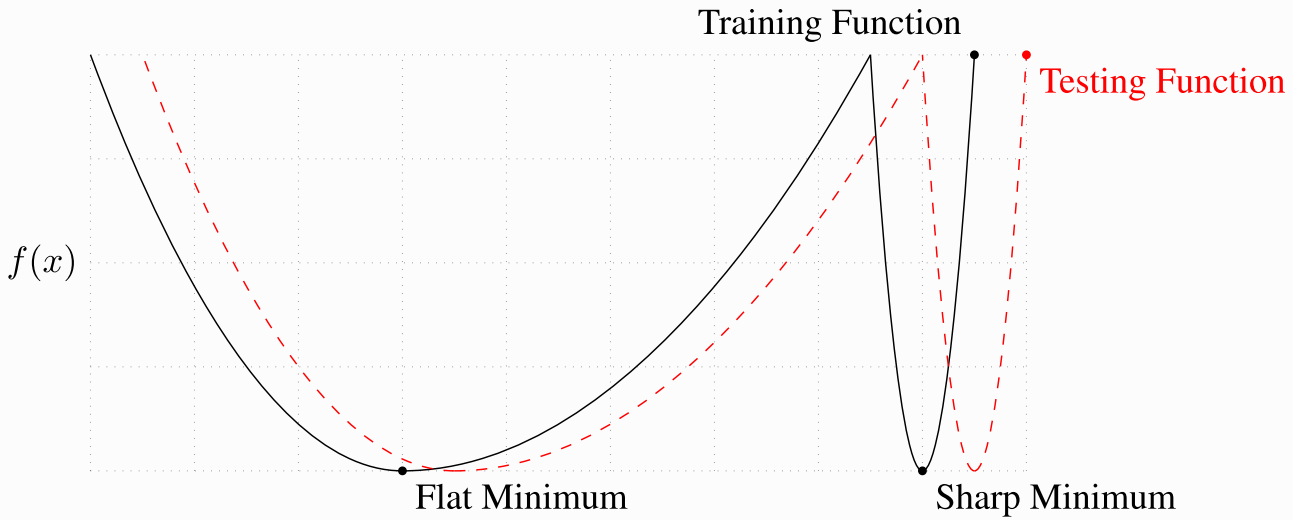
SWA and SGD



Test error and train loss surfaces for SGD and SWA ran from the same initial point (ResNet-110, CIFAR-100).

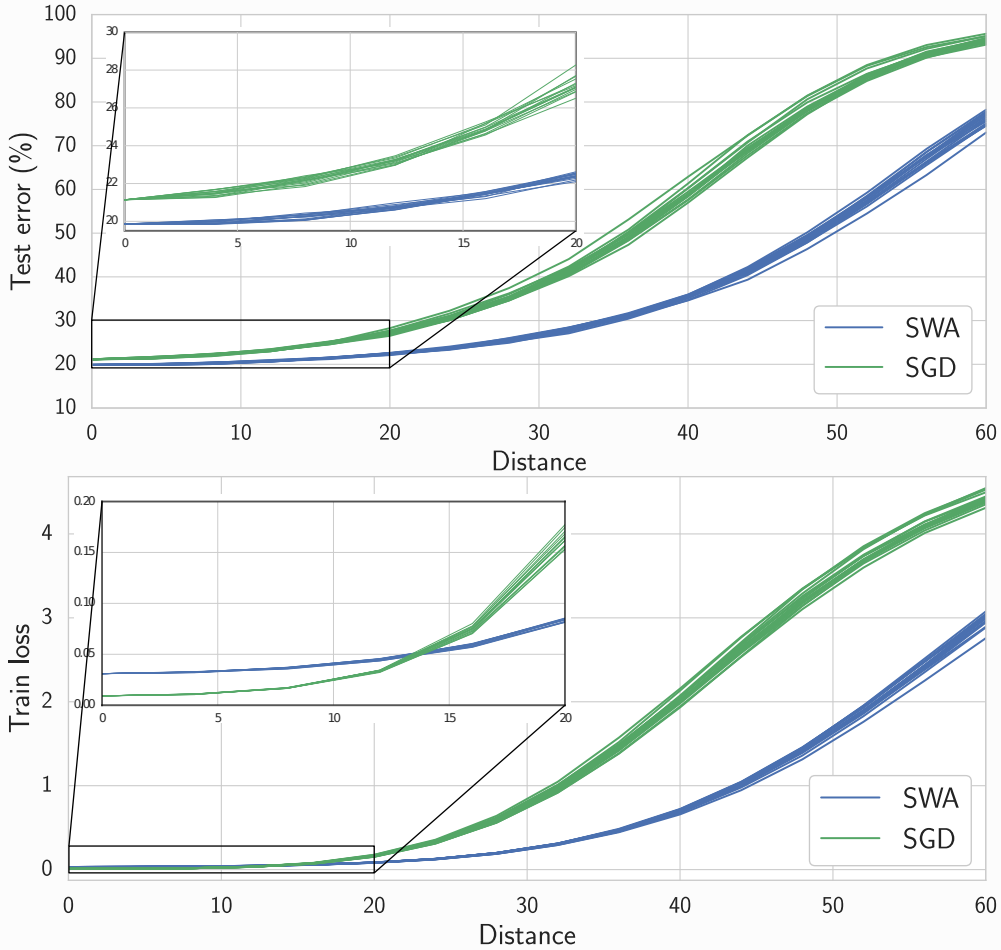
Optima Width

- It has been observed in practice that using large batch SGD leads to a drop in generalization performance.
- (Keskar et al., 2017) claim that there are different types of minima: flat minima lead to strong generalization, while sharp minima generalize poorly.



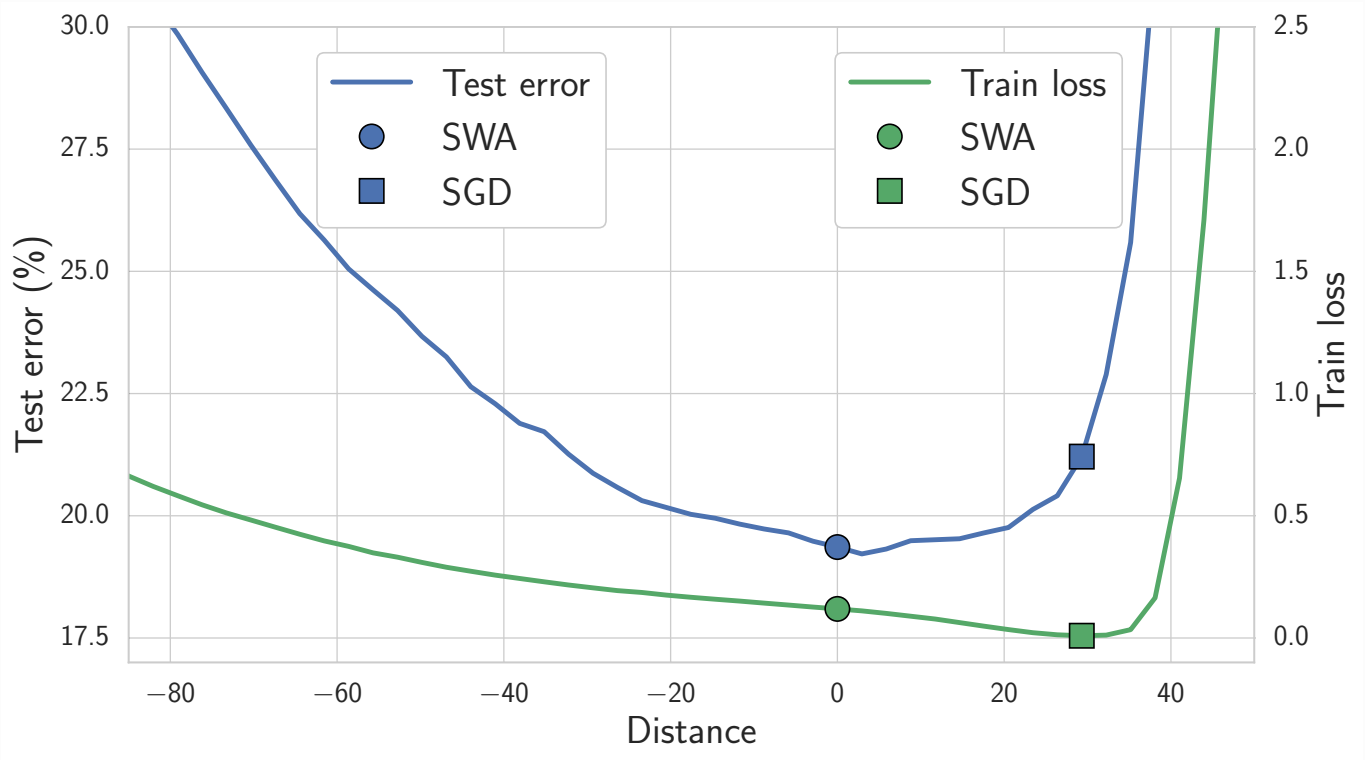
- At the same time, (Dinh et al., 2017) argue that all the known definitions of sharpness are unsatisfactory and cannot on their own explain generalization.

Optima Width



(**Top**) Test error and (**Bottom**) L_2-regularized cross-entropy train loss (ResNet-110, CIFAR-100) as a function of a point on a random ray starting at SWA (blue) and SGD (green). Each line corresponds to a different random ray.

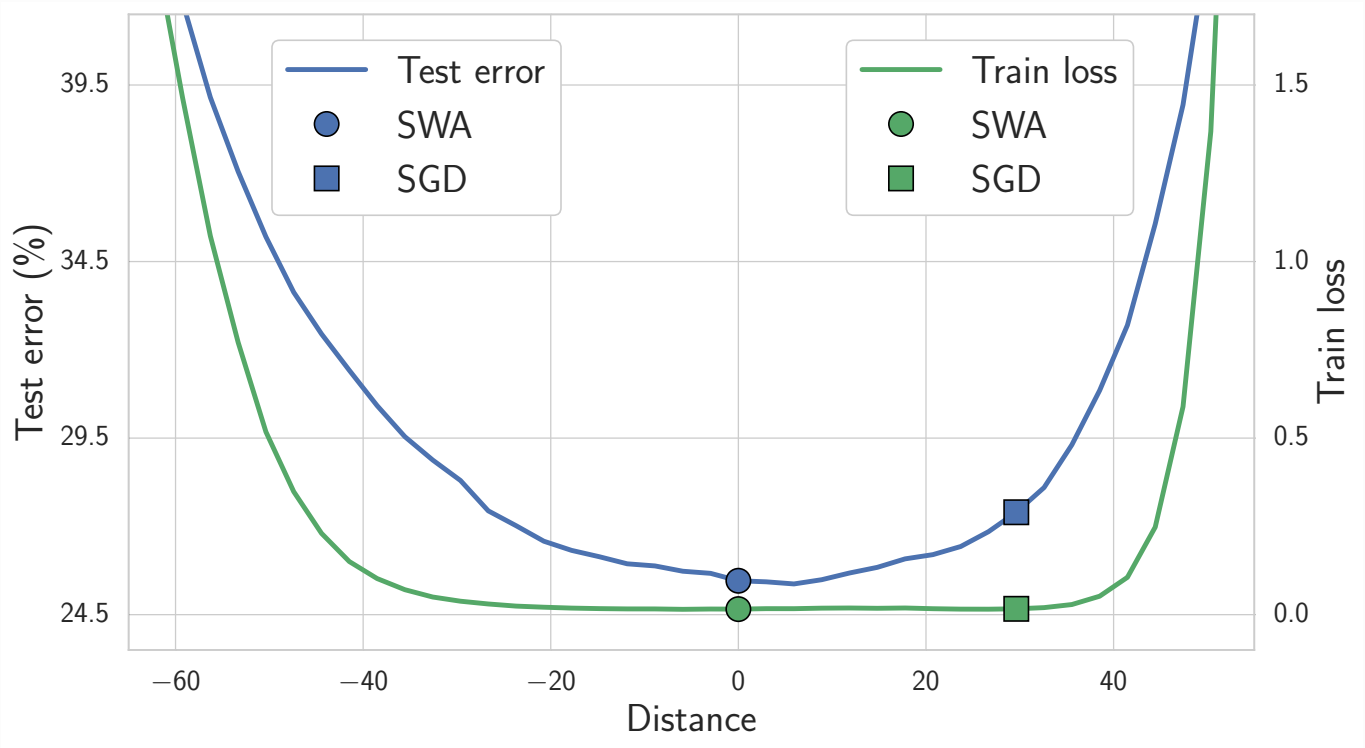
Optima Width (CIFAR-100, ResNet-110)



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}.

$$w(t) = t w_{\text{SWA}} + (1 - t) w_{\text{SGD}}$$

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}.

$$w(t) = tw_{\text{SWA}} + (1-t)w_{\text{SGD}}$$

Connection to Ensembling

Let $f(\cdot)$ – prediction of DDN parametrized by weights w.

Assumptions:

1. f is a scalar (e.g. the probability for a particular class) and $f \in C^2(w)$.
2. Points w_i proposed by FGE are close in the weight space and concentrated around their average $w_{\text{SWA}} = \frac{1}{n}\sum_{i=1}^{n} w_i$.

We denote $\Delta_i = w_i - w_{\text{SWA}}$. Note that $\sum_{i=1}^{n} \Delta_i = 0$.

Ensembling the networks w_i corresponds to averaging the function values

$$\bar{f} = \frac{1}{n}\sum_{i=1}^{n} f(w_i).$$

Consider the linearization of f at w_{SWA}.

$$f(w_j) = f(w_{\text{SWA}}) + \langle \nabla f(w_{\text{SWA}}), \Delta_j \rangle + O(\|\Delta_j\|^2).$$

Connection to Ensembling

Estimate the difference between $\bar{f}$ and $f(w_{\mathsf{SWA}})$

$$\begin{aligned}\bar{f} - f(w_{\mathsf{SWA}}) &= \frac{1}{n}\sum_{i=1}^{n}\left(\langle \nabla f(w_{\mathsf{SWA}}), \Delta_i\rangle + O(\|\Delta_i\|^2)\right) \\ &= \left\langle \nabla f(w_{\mathsf{SWA}}), \frac{1}{n}\sum_{i=1}^{n}\Delta_i \right\rangle + O(\Delta^2) = O(\Delta^2),\end{aligned}$$

where $\Delta = \max_{i=1}^{n} \|\Delta_i\|$.

Note that the difference between the predictions of different perturbed networks is

$$f(w_i) - f(w_j) = \langle \nabla f(w_{\mathsf{SWA}}), \Delta_i - \Delta_j\rangle + O(\Delta^2),$$

SWA experiments (CIFAR)

Accuracies (%) of SWA, SGD and FGE methods for different training budgets.

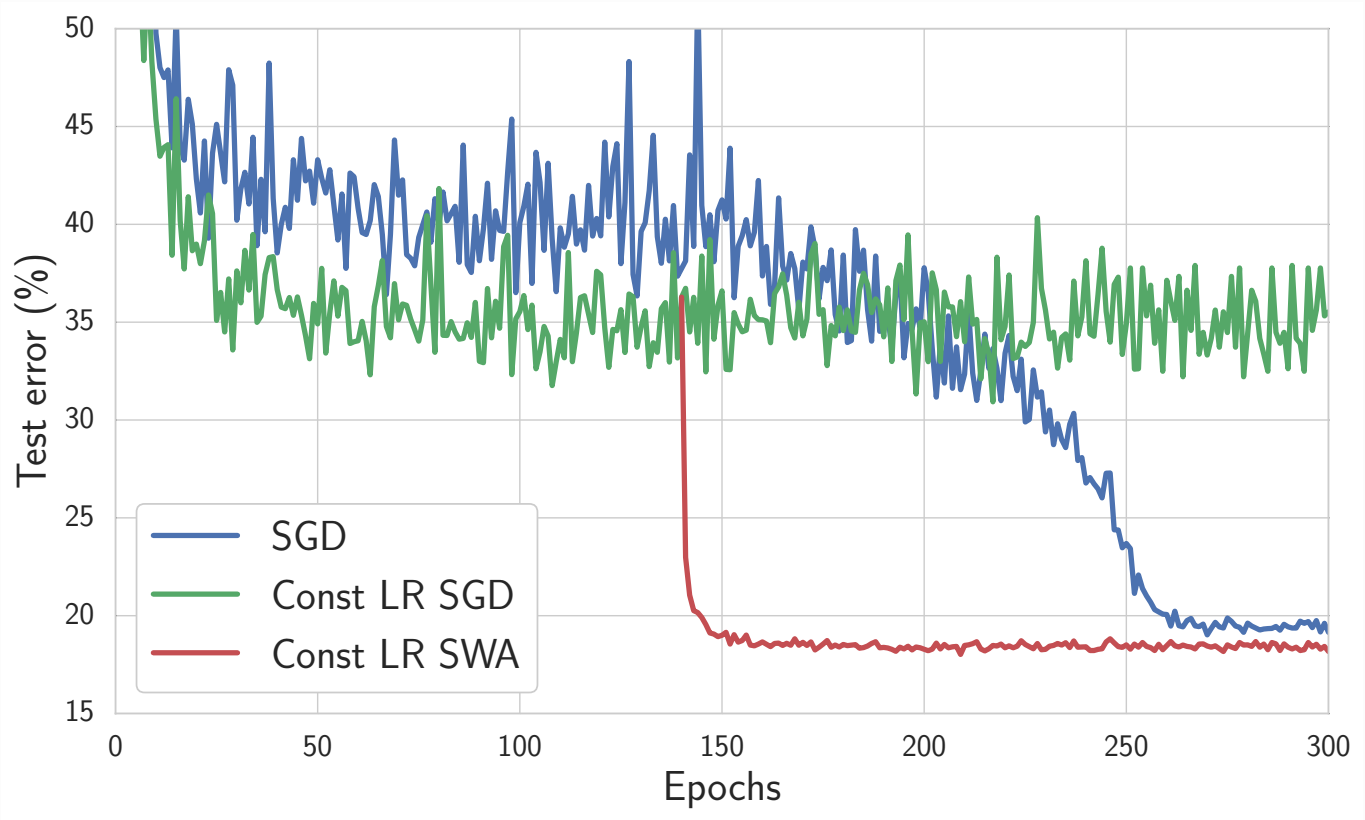
			SWA		
DNN (Budget)	SGD	FGE (1 Budget)	1 Budget	1.25 Budgets	1.5 Budgets
		CIFAR-100			
VGG-16 (200)	72.55 ± 0.10	74.26	73.91 ± 0.12	74.17 ± 0.15	74.27 ± 0.25
ResNet-110 (150)	78.49 ± 0.36	79.84	79.77 ± 0.17	80.18 ± 0.23	80.35 ± 0.16
WRN-28-10 (200)	80.82 ± 0.23	82.27	81.46 ± 0.23	81.91 ± 0.27	82.15 ± 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-110 (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

DNN training with a fixed learning rate



Test error as a function of training epoch for constant (green) and decaying (blue) learning rate schedules (Wide ResNet-28-10, CIFAR-100).

The red curve corresponds to averaging the points along the trajectory of SGD with constant learning rate starting at epoch 140.

Future work

We believe that these insights of geometric properties of DNN loss surfaces will have valuable implications for deep learning research directions, including:

- Improving the efficiency, reliability, and accuracy of training
- Creating better ensembles
- Deriving more effective posterior approximation families in Bayesian deep learning
- Developing efficient stochastic MCMC approaches which could now jump along bridges between modes, rather than getting stuck exploring a single mode
- Constructing methods which are more robust to adversarial attacks

Links

Papers:

- Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs
  arxiv.org/abs/1802.10026
- Averaging Weights Leads to Wider Optima and Better Generalization
  arxiv.org/abs/1803.05407

Code:

- Stochastic Weight Averaging in PyTorch
  github.com/timgaripov/swa