Optimization in Deep Neural Network

Liang Wang

April 19, 2021

Outline

Neural Network Basics

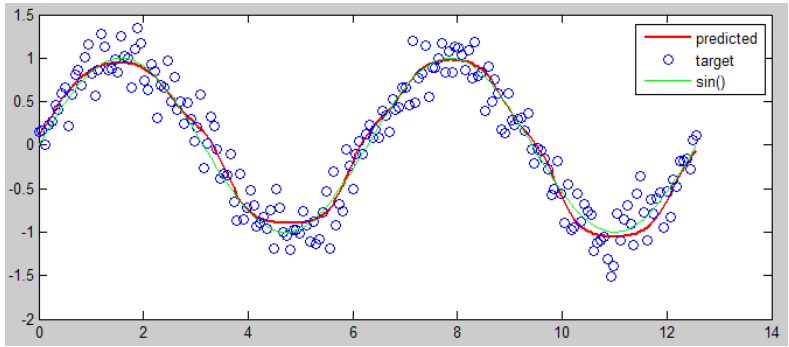
Optimization Problem and Solutions
1. Gradient instability
2. Dimension Varied Oscillation
3. Multi Modal Non-convexity
4. Sensitivity from Bayesian Prospective

CIFAR 10 Example

Reference

Network Motivation

1. Function Approximation



2. Gaussian Process [Blake Bordelon, 2020]

Prior: $f(x) \sim \mathcal{GP}(m(x), k(x,x'))$, meaning $(f(x_1), \ldots, f(x_N)) \sim \mathcal{N}(\boldsymbol{\mu}, K)$, with $\boldsymbol{\mu}_i = m(x_i)$ and $K_{ij} = \mathrm{cov}(f(x_i), f(x_j)) = k(x_i, x_j)$.

$$\overbrace{p(f(x)|\mathcal{D})}^{\text{GP posterior}} \propto \overbrace{p(\mathcal{D}|f(x))}^{\text{Likelihood}} \overbrace{p(f(x))}^{\text{GP prior}}$$

$$\frac{df(\mathbf{x};\boldsymbol{\theta})}{dt} = -\sum_{\mu=1}^{P} K_{NTK}(\mathbf{x}, \mathbf{x}^{\mu}; \boldsymbol{\theta}_0)\left(f(\mathbf{x}^{\mu};\boldsymbol{\theta}) - y^{\mu}\right)$$

Gradient flow to zero error: $f(\mathbf{x}) = \mathbf{y}^{\top}\mathbf{K}_{NTK}^{-1}\mathbf{k}(\mathbf{x})$

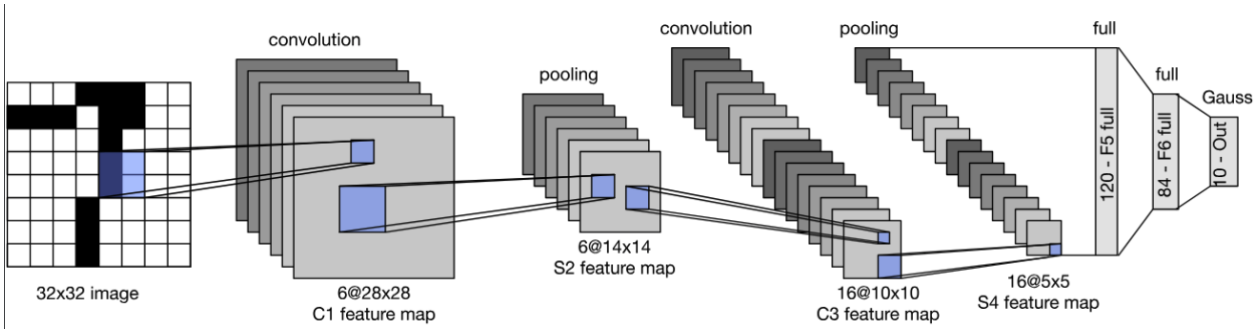
$K_{NTK}(\mathbf{x}, \mathbf{x}') = \nabla_{\boldsymbol{\theta}} f(\mathbf{x};\boldsymbol{\theta}_0) \cdot \nabla_{\boldsymbol{\theta}} f(\mathbf{x}';\boldsymbol{\theta}_0)$ (Neural Tangent Kernel)

$\mathbf{K}_{NTK}$ is the $P \times P$ kernel gram matrix $\mathbf{K}_{NTK,\mu\nu} = K_{NTK}(\mathbf{x}^{\mu}, \mathbf{x}^{\nu})$,

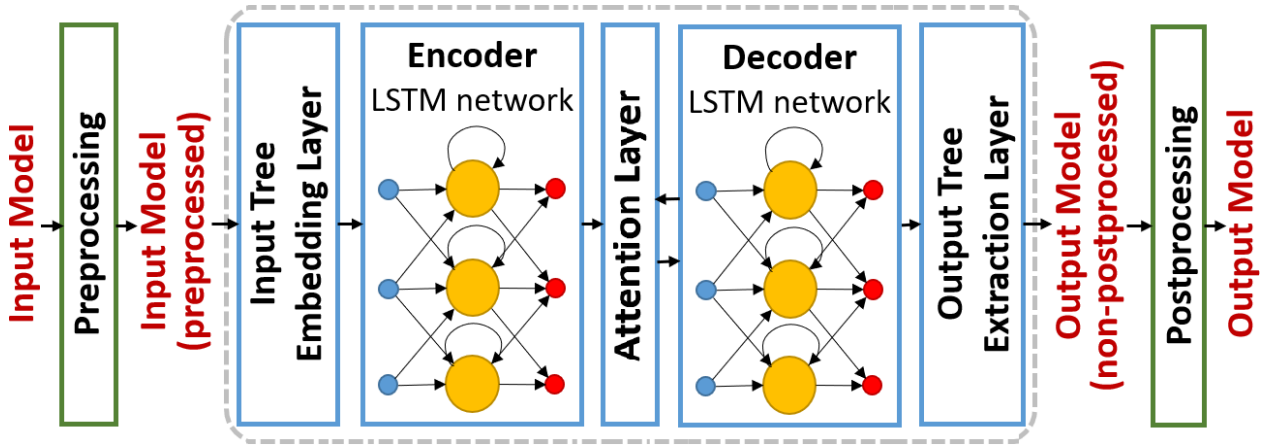
Network Structure

Different Neural Network structure for different dataset.

1. CNN for pattern recognition [Y. Le Cun and Jackel, 1999]



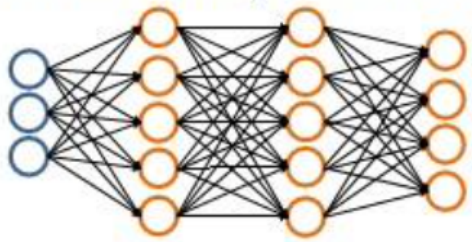
2. LSTM for time series [Felix A. Gers, 1999]



3. ANN for typical regression/classification

Optimization Problems–Setup

Neural Network (Classification)



$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$

$L =$ total no. of layers in network

$s_l =$ no. of units (not counting bias unit) in layer l

Binary classification

$y = 0$ or 1

1 output unit

Multi-class classification (K classes)

$y \in \mathbb{R}^K$ E.g. $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

pedestrian car motorcycle truck

K output units

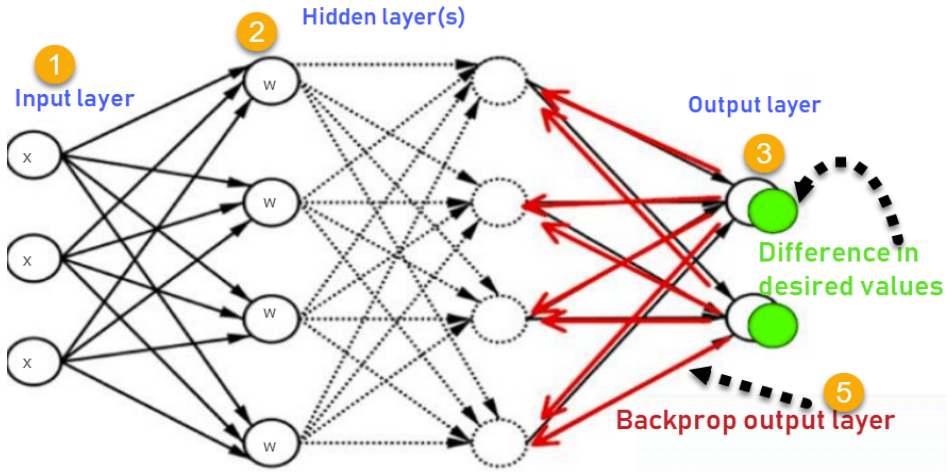
Picture from : *[Ng, 2012]*

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{k} y_k^{(i)} \log\left(h_\Theta\left(x^{(i)}\right)\right)_k + \left(1 - y_k^{(i)}\right)\log\left(1 - \left(h_\Theta\left(x^{(i)}\right)\right)_k\right)\right]$$

$$+ \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}\left(\Theta_{ji}^{(l)}\right)^2$$

Optimization Problems–Gradient Computation

Back propagation



Forward

Backward

$$z = b^2$$

$$b = a - y$$

$$a = \langle \mathbf{x}, \mathbf{w} \rangle$$

$$\frac{\partial z}{\partial b} = 2b$$

$$\frac{\partial z}{\partial a} = \frac{\partial z}{\partial b}\frac{\partial b}{\partial a} = \frac{\partial z}{\partial b}$$

$$\frac{\partial z}{\partial \mathbf{w}} = \frac{\partial z}{\partial a}\frac{\partial a}{\partial \mathbf{x}} = \frac{\partial z}{\partial a}\mathbf{w}^T$$

Picture from : *[Li, 2019]*

Optimization Problems–Gradient Exploding/Vanishing

- Consider a network with d layers

$$\mathbf{h}^t = f_t(\mathbf{h}^{t-1}) \quad \text{and} \quad y = \ell \circ f_d \circ \ldots \circ f_1(\mathbf{x})$$

- Compute the gradient of the loss ℓ w.r.t. $\mathbf{W}_t$

$$\frac{\partial \ell}{\partial \mathbf{W}^t} = \frac{\partial \ell}{\partial \mathbf{h}^d} \underbrace{\frac{\partial \mathbf{h}^d}{\partial \mathbf{h}^{d-1}} \cdots \frac{\partial \mathbf{h}^{t+1}}{\partial \mathbf{h}^t}} \frac{\partial \mathbf{h}^t}{\partial \mathbf{W}^t}$$

Multiplication of d-t matrices

Gradient Exploding

$1.5^{100} \approx 4 \times 10^{17}$

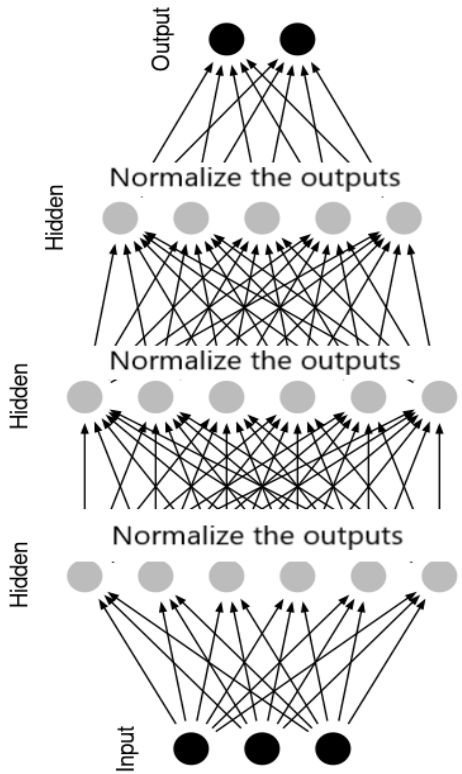
Gradient Vanishing

$0.8^{100} \approx 2 \times 10^{-10}$

Picture from : *[Li, 2019]*

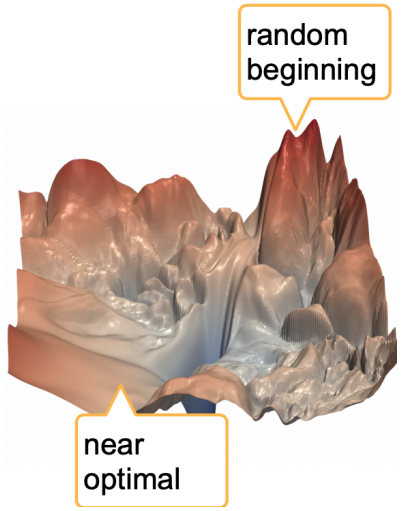
Optimization Problems–Gradient Exploding/Vanishing

1. Batch Normalization



Normalize layer output with $\mu = \beta, \sigma = \gamma$ [Ioffe and Szegedy, 2015]

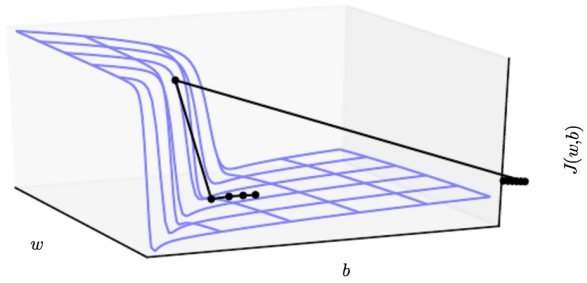
2. Smart Initialization



Xavier initialize to ensure $\frac{\partial L}{\partial W^t}$ is within reasonable range[Glorot and Bengio, 2010]

$$W \sim U[\frac{-\sqrt{6}}{\sqrt{n_j+n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j+n_{j+1}}}]$$

3. Gradient Clipping



Cap the maximum gradient movement [Razvan Pascanu, 2013]

$$\frac{\partial h_t^d}{\partial W_t^d} || \frac{\partial h_t^d}{\partial W_t^d} ||^{-1} c$$

or

$$max(\frac{\partial h_t^d}{\partial W_t^d}, c)$$

Optimization Problems–Dimension Varied Oscillation

1.Adagrad [Duchi and Singer, 2011]

$$
\begin{aligned}
x^{k+1} &= \underset{x\in X}{\operatorname{argmin}}\, f(x) + \frac{1}{2r^k}||x - x^k||_2^2 \\
&\approx \underset{x\in X}{\operatorname{argmin}}\, f(x^k) + \nabla f(x^k)^T(x - x^k) + \frac{1}{2r^k}||x - x^k||_2^2 \\
&= \underset{x\in X}{\operatorname{argmin}}\, \nabla f(x^k)^T x + \frac{1}{2r^k}||x - x^k||_2^2 \\
&= x^k - r^k \nabla f(x^k)
\end{aligned}
\tag{1}
$$

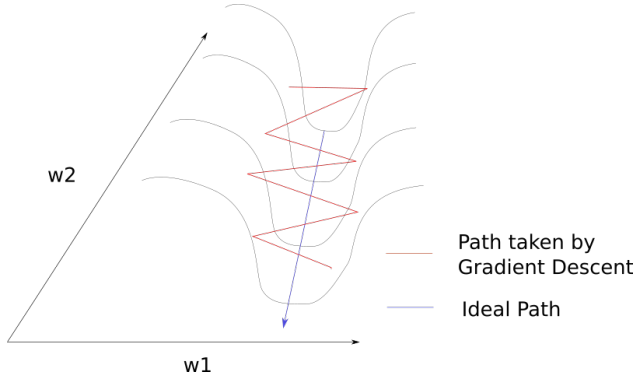
If $||X||_B^2 = x^T Bx \geq 0, B = \sqrt{Diag(\sum_{i=1}^t \nabla_x L(x_i) \odot \nabla_x L(x_i))}$

$$
\begin{aligned}
x^{k+1} &= \underset{x\in X}{\operatorname{argmin}}\, f(x) + \frac{1}{2r^k}||x - x^k||_B^2 \\
&= x^k - r^k B^{-1} \nabla f(x^k) \\
&\equiv x^k - (r^*)^k \nabla f(x^k)
\end{aligned}
\tag{2}
$$

with $(r^*)^k = \frac{r^k}{\sqrt{\epsilon I + diag(G_t)}}$

Optimization Problems–Dimension Varied Oscillation

1.Adagrad [Duchi and Singer, 2011]



Use variation to scale the gradient movement each dimension!

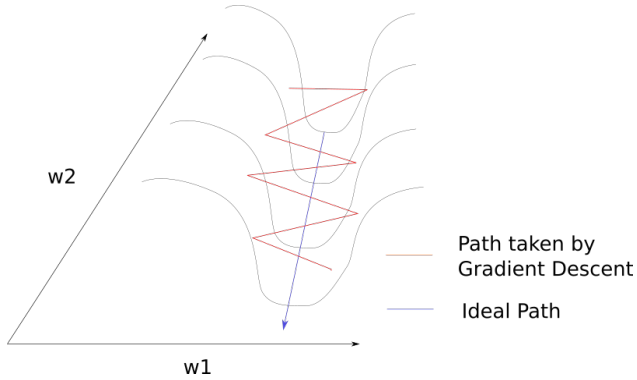
$$S_{dw} = S_{dw} + Diag(\sum_{i=1}^{t} \nabla_w L(w_i) \odot \nabla_x L(w_i))$$

$$w_{t+1} = w_t - \alpha_t \frac{\nabla_x L(w_i)}{\sqrt{S_{dw} + \epsilon}}$$

Observe that $S_{dw} \to \infty$, then we w is not updated anymore.

Optimization Problems–Dimension Varied Oscillation

2.RMSprop [Tieleman, 2012]



Use weighted average of the variation instead of the total variation!

$$S_{dw} = \beta_1 S_{dw} + (1-\beta_1) Diag(\sum_{i=1}^{t} \nabla_w L(w_i) \odot \nabla_x L(w_i))$$

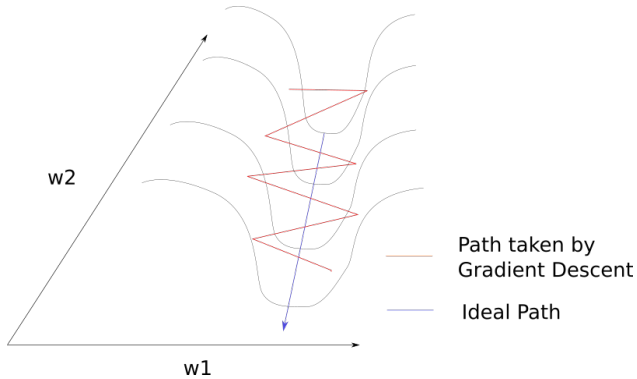
$$w_{t+1} = w_t - \alpha_t \frac{\nabla_x L(w_i)}{\sqrt{S_{dw} + \epsilon}}$$

$\beta_1 \in (0,1)$, its an exponentially weighted average since

$$\frac{1}{1-\beta_1} = 1 + \beta_1 + \beta_1^2 + ...$$

Optimization Problems– Stationary Point

3.Adam [Diederik P. Kingma, 2015]



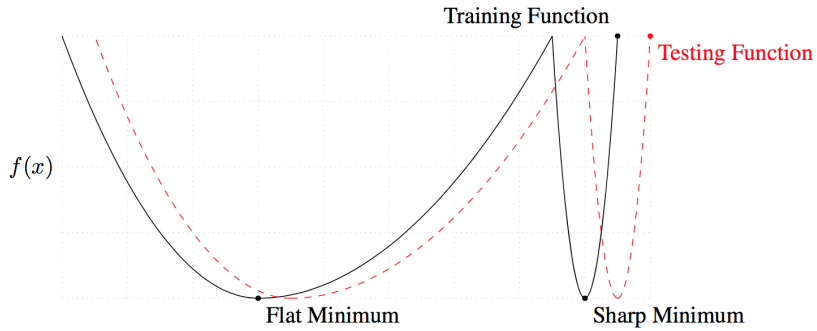
1. Use 1st moment momentum to escape the stationary point
2. Use 2nd moment momentum to scale along each direction

$$
\begin{aligned}
S_{dw} &= \beta_1 S_{dw} + (1-\beta_1) Diag(\nabla_w L(w_i) \odot \nabla_x L(w_i))) \\
V_{dw} &= \beta_2 S_{dw} + (1-\beta_2) \nabla_w L(w_i) \\
w &= w - \alpha_t \frac{V_{dw}}{\sqrt{S_{dw} + \epsilon}}
\end{aligned}
\tag{3}
$$

Usually, $\beta_2 = 0.9, \beta_1 = 0.999, \epsilon = 10^{-8}$

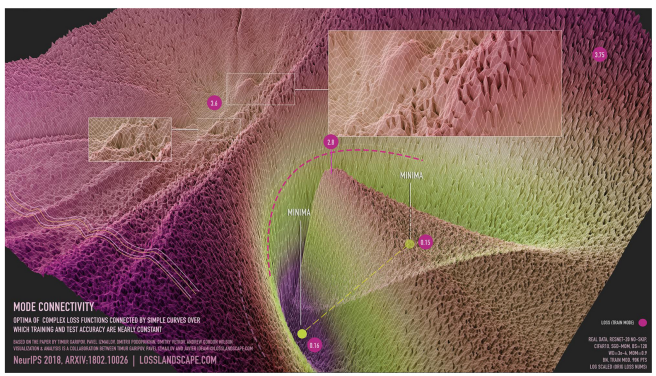
Optimization Problems– Multi-Modal

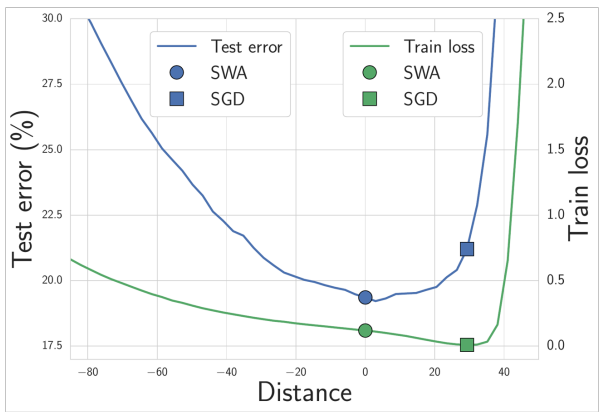
Stochastic Weight Average(SWA)[Timur Garipov and Dmitry Vetrov, 2018b]



Picture from : *[Nitish Shirish Keskar and Tang, 2017]*

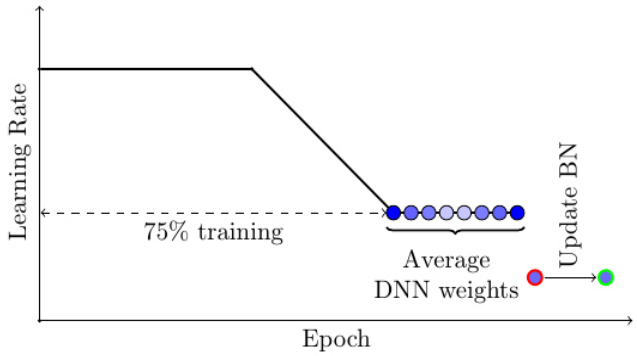
NN loss indeed has flat region[Timur Garipov and Dmitry Vetrov, 2018a] and SGD solution tends to be on the boundary.

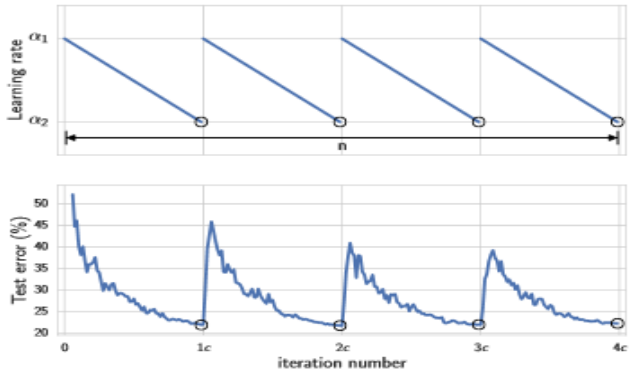




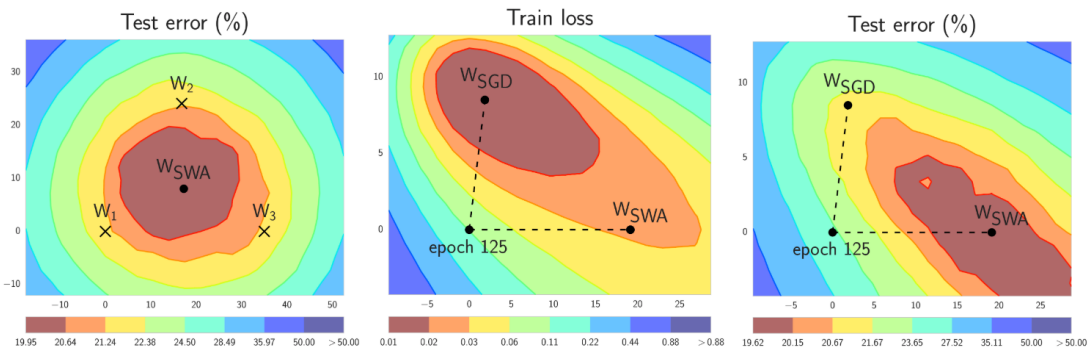
Optimization Problems– Multi-Modal

Stochastic Weight Average(SWA)[Timur Garipov and Dmitry Vetrov, 2018b]





1. Train Neural Network to find $\hat{w}$
2. With $\hat{w}$ as initialization, retrain the model k times with cyclic learning rate to find $\hat{w}_i$
3. Average over $\hat{w}_{flat} = \sum_{i=1}^{k} \frac{1}{k} \hat{w}_i$



Optimization Problems– Uncertainty Quatification

Bayes Marginalization:

$$
\begin{aligned}
p(y|x,D) &= \int p(y|x,D,w)p(w|D)dw \\
p(y|x,D) &\approx \sum_{s=1}^{S} p(y|x,D,w_s), w_s \sim p(w|D)
\end{aligned}
\tag{4}
$$

Under certain conditions[Stephan M, 2017], using SGD:

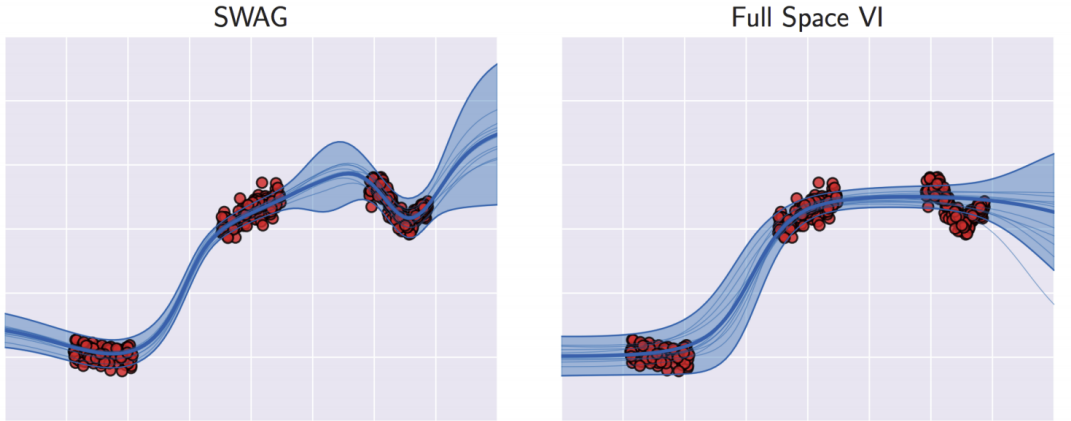
$$
d\theta(t) = -\epsilon g(\theta)dt + \frac{\epsilon}{\sqrt{S}} B dW_t
$$

$d\theta(t)$ is the weight update, $g(\theta)$ is the weight gradient, ϵ is step size,B is kernel, dW_t is Brownian Motion, S is the batch size.

Optimization Problems– Uncertainty Quatification

$\Rightarrow$ Stochastic Weight Average Gaussian(SWAG) [Maddox, 2019]

1. Train Neural Network to find $\hat{w}$
2. With $\hat{w}$ as initialization, retrain the model k times with cyclic learning rate to find $\hat{w}_i$
3. Compute $\hat{w}_{flat} = \sum_{i=1}^{k} \frac{1}{k} \hat{w}_i$,
   $\hat{\Sigma}_w = Diag(\sum_{i=1}^{k} \frac{1}{k} [E[\hat{w}_i^2] - \hat{w}_{flat}^2])$
4. Sample $w \sim N(\hat{w}_{flat}, \hat{\Sigma}_w)$



Picture from : *[Maddox, 2019]*

CIFAR 10 Examples

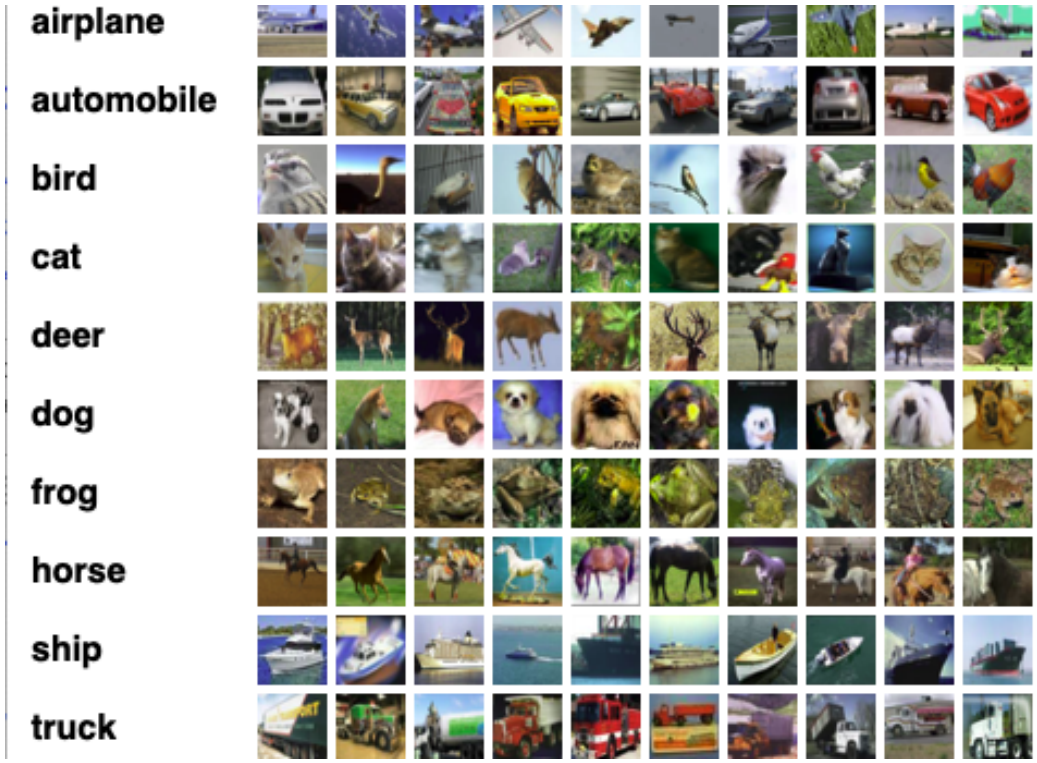


Figure: Cifar 10 Dataset

1. It contains 10 classes, with 6000 images per class.
2. I use 50,000 as training set, 10,000 as testing set

CIFAR 10 Examples

I implemented a multi-layer percetron classifier with

- 256, 128, 10 three layers of network network with 80,000 w_i
- Adam without resorting to built-in pytorch optimizer
- SWA to improve on 'Flat Minimum' to 49.84% testing error
- SWAG to quantify uncertainty and provide learning feedback

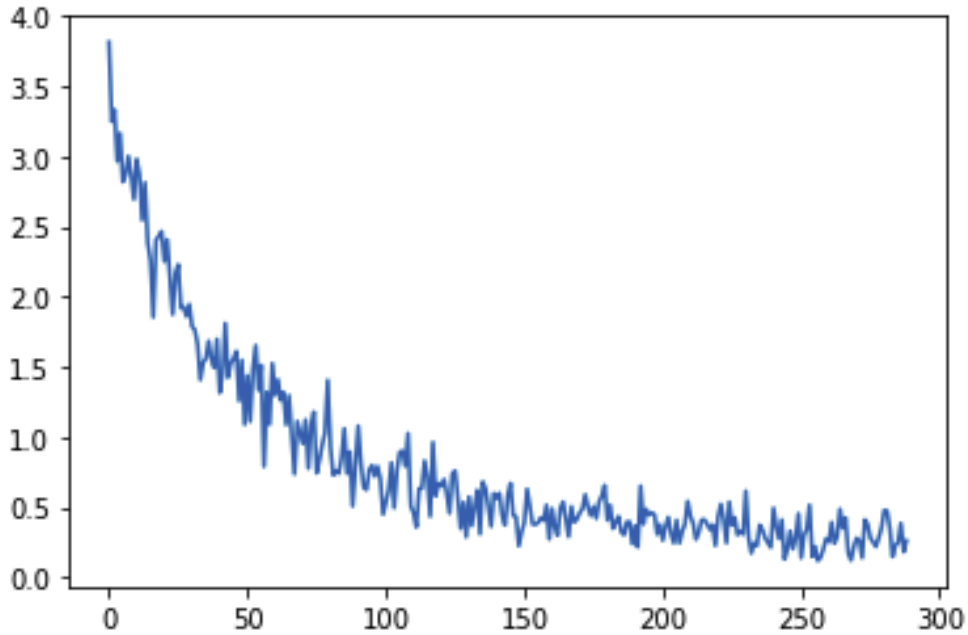


Figure: **In sample Training Loss along iteration ($\alpha = 0.001$)**

CIFAR 10 Examples

I implemented a multi-layer percetron classifier with

- 256, 128, 10 three layers of network network with 80,000 w_i
- Adam without resorting to built-in pytorch optimizer
- SWA to improve on 'Flat Minimum' to 49.84% testing error
- SWAG to quantify uncertainty and provide learning feedback

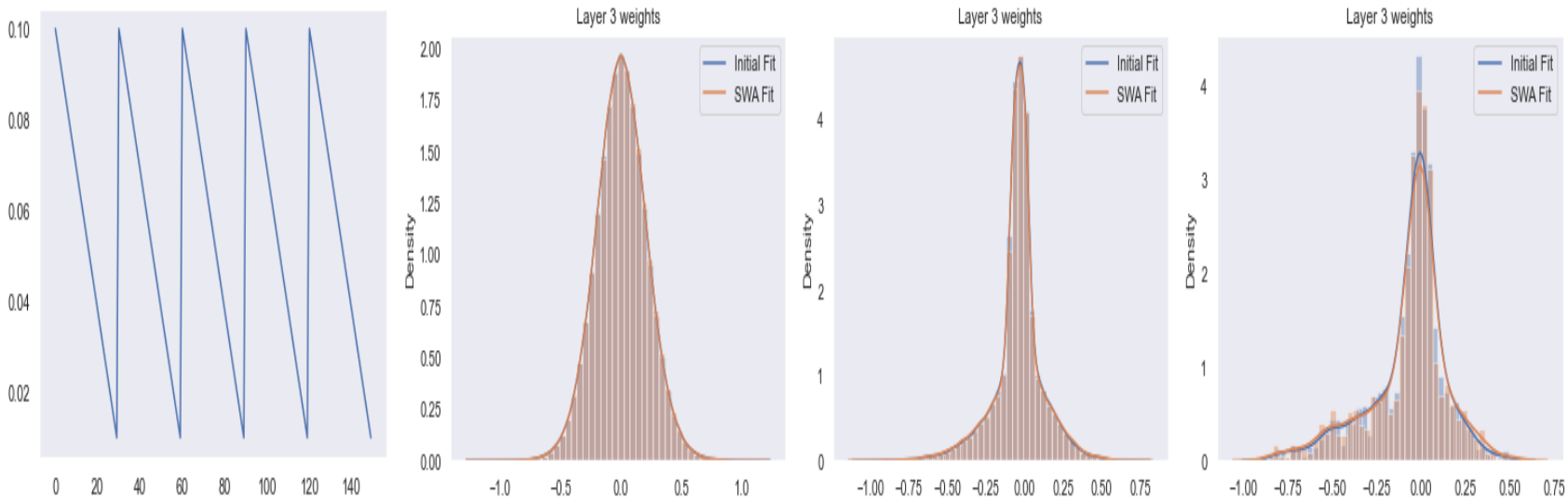


Figure: **SWA Result**

CIFAR 10 Examples

I implemented a multi-layer percetron classifier with

- 256, 128, 10 three layers of network network with 80,000 w_i
- Adam without resorting to built-in pytorch optimizer
- SWA to improve on 'Flat Minimum' to 49.84% testing error
- SWAG to quantify uncertainty and provide learning feedback

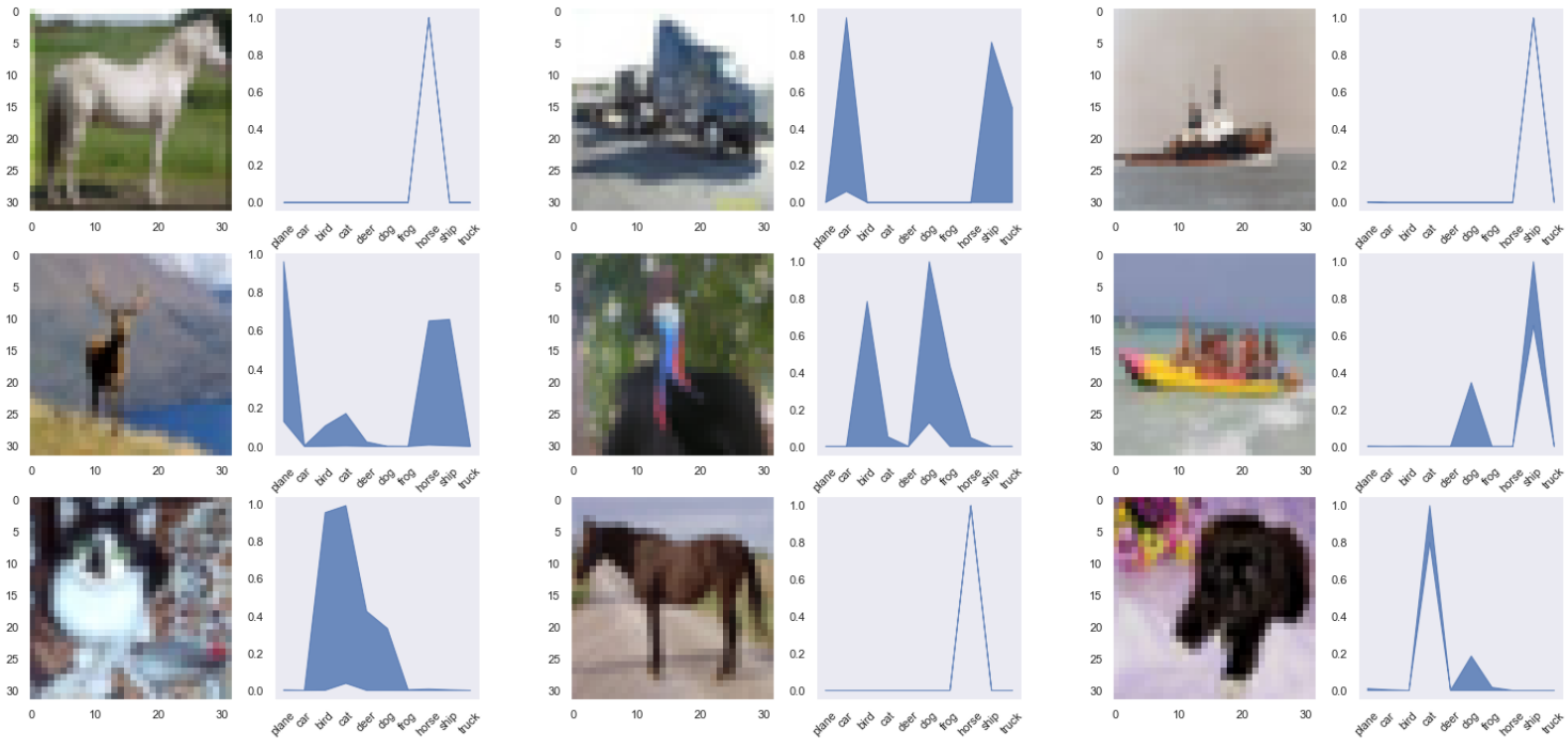


Figure: **SWAG Confidence Interval for CIFAR10**

Reference I

C. P. Blake Bordelon, Abdulkadir Canatar. Spectrum dependent learning curves in kernel regression and wide neural networks. *Journal of Machine Learning Research*, 2020.

. J. B. Diederik P. Kingma. Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

H. E. Duchi, John and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 2011.

J. S. . F. C. Felix A. Gers. Learning to forget: Continual prediction with lstm. *Proc. of the 9th Int. Conf. on Artificial Neural Networks*, 1999.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010.

Reference II

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning, PMLR*, 2015.

A. S. . M. Li. Stat157 lecture notes https://courses.d2l.ai/berkeley-stat-157/index.html. *UC Berkeley*, 2019.

G. T. I. P. V. D. W. A. G. Maddox, Wesley. A simple baseline for bayesian uncertainty in deep learning. *Conference on Neural Information Processing Systems*, 2019.

A. Ng. Cs229 lecture notes http://cs229.stanford.edu/notes/cs229-notes5.pdf. *Stanford*, 2012.

J. N. M. S. Nitish Shirish Keskar, Dheevatsa Mudigere and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *International Conference on Learning Representations*, 2017.

Reference III

Y. B. Razvan Pascanu, Tomas Mikolov. On the difficulty of training recurrent neural networks. *Proceedings of the 30th International Conference on International Conference on Machine Learning*, 2013.

M. D. H. D. M. B. Stephan M, t. Stochastic gradient descent as approximate bayesian inference. *Journal of Machine Learning Research*, 2017.

G. Tieleman, T. Hinton. Neural networks for machine learning. *Technical report*, 2012.

D. P. Timur Garipov, Pavel Izmailov and A. G. W. Dmitry Vetrov. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Conference on Neural Information Processing Systems*, 2018a.

D. P. Timur Garipov, Pavel Izmailov and A. G. W. Dmitry Vetrov. Averaging weights leads to wider optima and better generalization. *Uncertainty in Artificial Intelligence (UAI)*, 2018b.

Reference IV

J. S. D. D. H. R. E. H. W. H. Y. Le Cun, B. Boser and L. D. Jackel. Handwritten digit recognition with a back-propagation network. *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, 1999.