UNDERSTANDING GENERALIZATION IN DEEP NEURAL NETWORKS

The role of the optimization method, metrics for measuring generalization, and an analytic approach for gaining insight

Anastasia Borovykh
Based on joint work with Sander Bohte and Cornelis Oosterlee

Woudschoten, 2019

STRUCTURE

- A brief introduction to neural networks and their generalization capabilities
- Previous observations on generalization
    - The role of the optimization algorithm
    - Measuring the generalization capabilities
- Understanding the network output through analytic expressions
    - A linear approximation
    - A Taylor expansion approach

NEURAL NETWORKS

- Given an input $x \in \mathbb{R}^{n_0}$ and some target $y \in \mathbb{R}^{n_L}$ such that $(x, y) \sim \mathcal{D}$, a neural network computes

  $$\hat{y}_t(x, w) = W^L f(W^{L-1} ... f(W^1 x))...),$$

  where $f(\cdot)$ is the non-linear activation function and $W^{(l)} \in \mathbb{R}_{n_l \times n_{l-1}}$ are the weights.
- It is a compositional mapping from input x to output y with parameters $\theta := (W^1, ... W^L)$.

TRAINING THE NETWORK

- We train the network to minimize some loss function $\mathcal{L}$, e.g. the mean squared error for regression problems.
- Weights are initialized from some distribution $\theta_0 \sim P_{\text{init}}$.
- In a first-order method the weights are updated according to,

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(x, y),$$

where $\mathcal{L}(x, y)$ can be the gradient computed over the full training set (X, Y) (gradient descent), or the gradient computed over a subset of the training data (X^S, Y^S) (stochastic gradient descent).

TRAINING THE NETWORK

Gradient descent

- By a continuous approximation of gradient descent, parameters evolve as

$$d\theta_t = -\eta(\nabla_\theta \hat{y}_t)^\top \nabla_{\hat{y}} \mathcal{L}(\hat{y}_t),$$

where $\mathcal{L}$ is the loss function and η is the learning rate.

Stochastic training

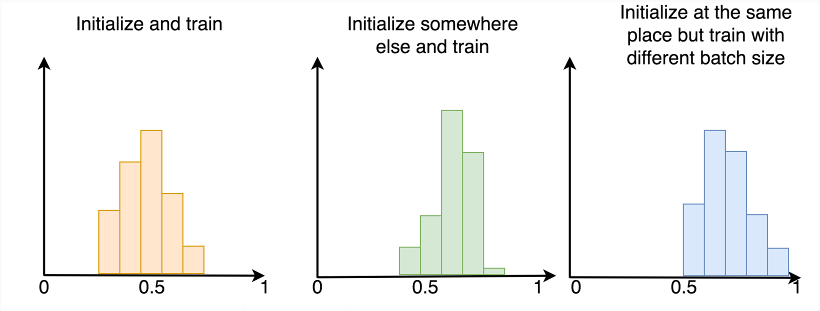
- Under stochastic training we have,

$$d\theta_t = -\eta(\nabla_\theta \hat{y}_t)^\top \nabla_{\hat{y}} \mathcal{L}(\hat{y}_t) + \sigma dW_t,$$

where W_t is a Brownian motion.

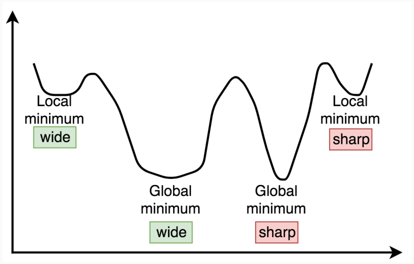
AN OBSERVATION...

Let's consider the S&P500 data, and try to train a neural network on it...



A histogram of the hit rate (=how many times up/down movement predicted correctly)

THE LOSS SURFACE



- When we are training the neural network, we are traversing the loss surface.
- The loss surface is a non-convex function of the high-dimensional weight vector θ.
- The loss surface has many local minima, global minima, wide minima, sharp minima. Which minimum is good?

THE GOAL

- We would like to gain insight into which minimum is good
- Is there a metric we could define to measure the ‘goodness’ of a minimum?

GENERALIZATION

- The ability of a network trained on dataset (x, y) to perform well on dataset $(\hat{\mathrm{x}}, \hat{\mathrm{y}})$, where $(\mathrm{x}, \mathrm{y}), (\hat{\mathrm{x}}, \hat{\mathrm{y}}) \sim P_{X,Y}$.
- We want for datasets (x, y) and $(\hat{\mathrm{x}}, \hat{\mathrm{y}})$,

$$\mathcal{L}(\mathrm{x}, \theta, \mathrm{y}) - \mathcal{L}(\hat{\mathrm{x}}, \theta, \hat{\mathrm{y}}).$$

  to be small.

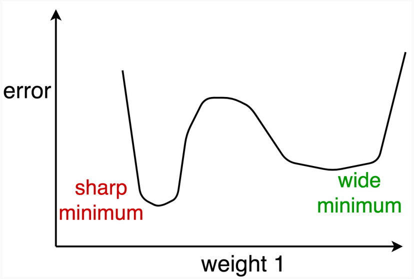
GENERALIZATION

- Overparametrized networks can achieve zero train loss, however this says nothing about their performance on unseen data (they might overfit on the noise in the train data!)
- Good generalization involves finding a trade-off between the complexity of the learned function and its ability to represent the train data.
- Generalization capabilities are influenced by the network architecture, dataset properties and the optimization algorithm used to train the network.

PREVIOUS WORK ON GENERALIZATION

- Different ways of measuring these generalization capabilities exist:
  - The flatness of a minimum can be a good indicator of its ability to generalize [Hochreiter, Schmidhuber (1997)], [Keskar et al (2016)], [Srebro et al (2017)]
  - The smoothness of the learned function can also be an indicator [Novak et al (2018)]
  - Various norms of the network parameters can be used to measure the capacity of neural networks [Bartlett, (1998)] [Srebro et al (2015)], [Bartlett et al (2017)] [Srebro et al (2017)]
- Noise in the neural network training algorithm can regularize the solution resulting in better generalization [Srebro et al (2015)] [Jastrzkebski et al (2017)] [Borovykh et al (2019)]

MEASURING GENERALIZATION: WEIGHT HESSIAN



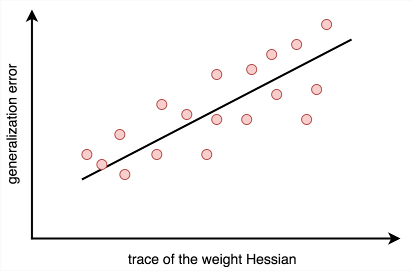
- An empirical way of measuring generalization is the smoothness with respect to inputs or weights.
- Flatness in weight space, i.e. the size of the trace of the Hessian with respect to the weights $\Delta^{\theta}\mathcal{L}(\hat{y}_t)$ with elements $\partial_{\theta_i}\partial_{\theta_j}\mathcal{L}(\hat{y}_t)$ has been proposed as a metric.

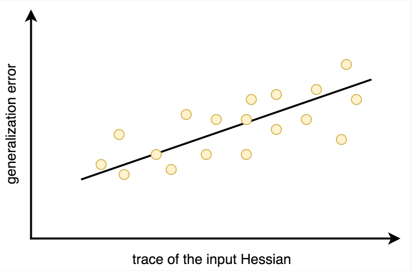
MEASURING GENERALIZATION: INPUT HESSIAN

- Smoothness in input space [Novak et al (2018)], e.g. the trace of $\Delta_x \mathcal{L}(\hat{y}_t)$ with elements $\partial_{x_i} \partial_{x_j} \mathcal{L}(\hat{y}_t)$ averaged over the input samples x^i, $i = 1, \ldots, N$.
- Smoothness in input space can be related to noise robustness, i.e. the output function should be robust to different noise to the input variables
- Smoothness in weight and input space are related.

MEASURING GENERALIZATION

Let's go back to our finance example and see if the metrics for the minima can provide any insight into when we will be able to perform well...





WHAT'S NEXT?

- We have seen that there exist minima which are ‘better’ than others
- Wide minima, with a small weight or input Hessian, are more resistant to noise and seem to result in better generalization
- Can we use the optimization algorithm to bias into these ‘good’ minima?

THE ROLE OF THE OPTIMIZATION ALGORITHM

- It was observed [Zhang et al, 2017] that deep networks can memorize noise, due to overparametrization, and thus perform bad out of sample
- The same network can however learn well on real data and perform well out of sample
- How can this be?
    - Generalization is not implicit to the network architecture, but is related to the way we train a neural network
    - Ways of controlling the complexity of the learned function exist: early stopping, learning rate, batch size

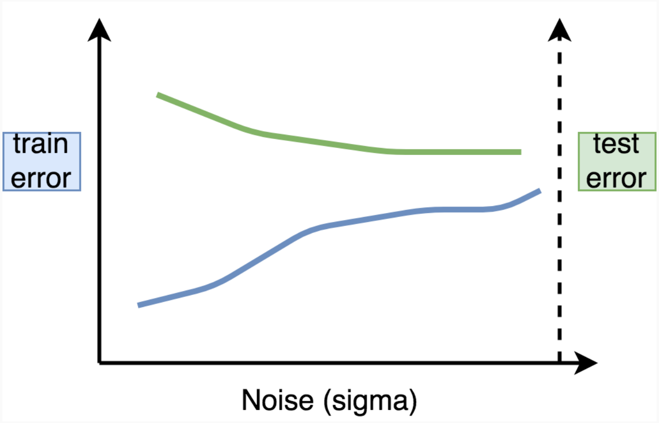
THE ROLE OF THE OPTIMIZATION ALGORITHM

It could be that that there exists a so-called implicit bias [Srebro et al (2015), (2017)]:

- the real-world data can be explained by a function with low complexity;
- the optimization algorithm biases towards low-complexity.

THE EFFECTS OF NOISE IN THE OPTIMIZATION ALGORITHM

The following can be observed... Let $d\theta_t = -\eta(\nabla_\theta \hat{y}_t)^\top \nabla_{\hat{y}} \mathcal{L}(\hat{y}_t) + \sigma dW_t$.



Increasing noise results in better test performance... Can we explain this?

UNDERSTANDING THE NETWORK OUTPUT

- Understanding the network output could help in understanding the generalization capabilities of the network
- Deep neural networks are models with many parameters and complex compositional mappings.
- The output is thus not an easy-to-analyze object.

UNDERSTANDING THE NETWORK OUTPUT: TWO APPROACHES

The large parameter limit

- under a particular scaling in the neural network and under the limit of network width $n_l \to \infty$, the network output during training becomes equivalent to a linear model

An Taylor expansion approximation

- higher order approximation terms can be obtained by using a Taylor expansion of the output

THE LINEAR APPROXIMATION UNDER GRADIENT DESCENT

- The first-order approximation of the network output is given by,

$$\hat{y}_t^{\text{lin}} := \hat{y}_0 + \nabla_\theta \hat{y}_0(\theta_t - \theta_0),$$

  where θ_0 are the parameters at initialization and $\hat{y}_0$ is the network output at initialization.
- Now, suppose $w^l \sim \frac{1}{\sqrt{n_{l-1}}}$.
- Under gradient descent as $n_1, ..., n_L \to \infty$, sequentially, [Jacot et al (2018)] [Lee et al (2019)]

$$\sup_{t \in [0,T]} ||\hat{y}_t - \hat{y}_t^{\text{lin}}||_2 \to 0.$$

THE LINEAR APPROXIMATION UNDER GRADIENT DESCENT

- If the linear approximation is sufficient we can directly solve for the output:

$$\hat{y}_t(X) = \left(I - e^{-\frac{\eta}{N}\nabla_\theta \hat{y}_0(X)\nabla_\theta \hat{y}_0(X)^T t}\right) Y + e^{-\frac{\eta}{N}\nabla_\theta \hat{y}_0(X)\nabla_\theta \hat{y}_0(X)^T t}\hat{y}_0.$$

THE LINEAR APPROXIMATION: THE EFFECTS OF GRADIENT DESCENT

Theorem

Gradient descent converges to the minimum norm solution, i.e.

$$\theta_t \to \arg \min_{\hat{y}_0 + \nabla_\theta \hat{y}_0 (\theta - \theta_0) = Y} ||\theta - \theta_0||_2.$$

- Gradient descent biases the algorithm into weights which are closest to the initial weights among all weights that satisfy $\lim_{t \to \infty} \hat{y}_t = Y$.

THE LINEAR APPROXIMATION: THE EFFECTS OF REGULARIZATION

- Suppose we train the model with regularization i.e. add a term $\lambda||\theta_t - \theta_0||_2^2$ to the loss function.
- Solving for the continuous form

$$\begin{aligned}\hat{y}_t(X) =& e^{-\frac{\eta}{N}(\nabla_\theta \hat{y}_0(X)\nabla_\theta \hat{y}_0(X)^T+\lambda)t}\hat{y}_0(X) \\ &+ (\nabla_\theta \hat{y}_0(X)\nabla_\theta \hat{y}_0(X)^T Y + \lambda \hat{y}_0)(\nabla_\theta \hat{y}_0(X)\nabla_\theta \hat{y}_0(X)^T + \lambda)^{-1} \\ &\cdot \left(I - e^{-\frac{\eta}{N}\nabla_\theta \hat{y}_0(X)\nabla_\theta \hat{y}_0(X)^T+\lambda)t}\right).\end{aligned}$$

- As $t \to \infty$ this is similar to the posterior of a Gaussian process where the likelihood is Gaussian with variance λ, i.e. λ is the noise in the observations Y.

THE LINEAR APPROXIMATION: THE EFFECTS OF NOISE

- If the number of training iterations is small, also under noisy gradient descent the linear model can be a approximation. Suppose this is the case.
- Then we can solve for the network output explicitly,

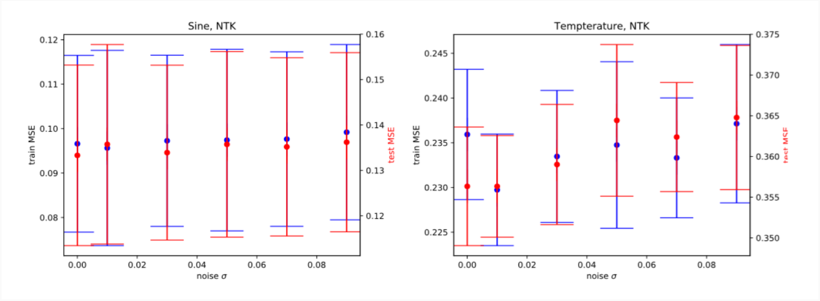
$$\hat{y}_t(X) = \left(I - e^{-\frac{\eta}{N}\Theta_0 t}\right) Y + e^{-\frac{\eta}{N}\Theta_0 t}\hat{y}_0(X) - \sigma\frac{\eta}{N}\int_0^t e^{-\frac{\eta}{N}\Theta_0(t-s)}\Theta_0 dW_s.$$

- Noise thus does not explicitly regularize or smooth the solution.
- The Hessian of the loss function:

$$\mathbb{E}\left[\Delta\mathcal{L}(\hat{y}_t)\right] = \nabla_\theta \hat{y}_0(X)(\mathbb{E}[\hat{y}_t(X)] - Y),$$

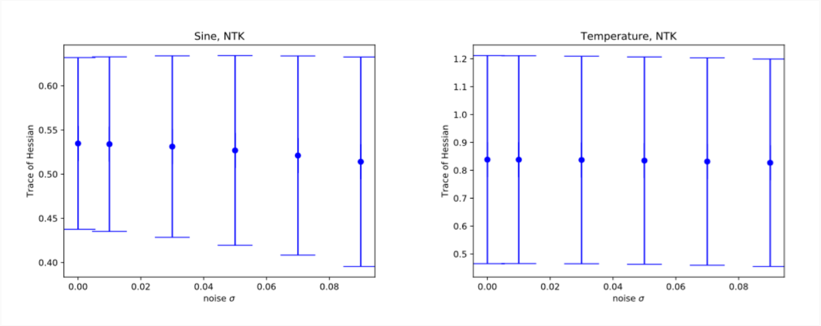
is not a function of σ.

THE LINEAR APPROXIMATION: THE EFFECTS OF NOISE



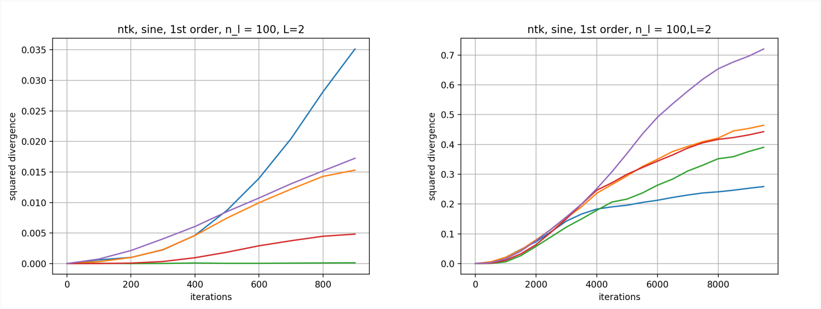
The network training MSE is an increasing function of noise, but test performance does not improve with more noise.

THE LINEAR APPROXIMATION: THE EFFECTS OF NOISE



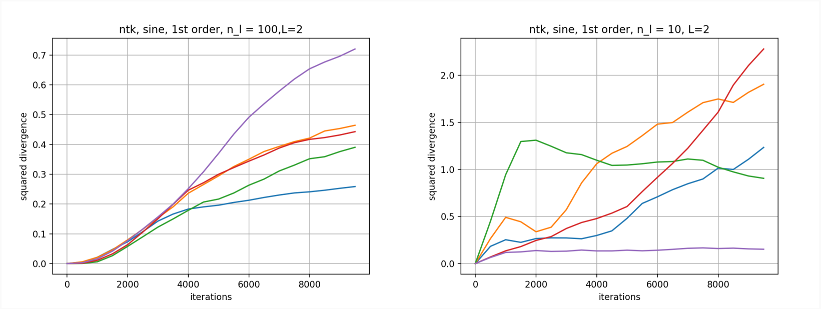
The weight Hessian, the metric for generalization, is not affected by noise.

WHEN IS THE LINEAR MODEL APPROXIMATION SUFFICIENT?



As the number of training iterations increases divergence grows.

WHEN IS THE LINEAR MODEL APPROXIMATION SUFFICIENT?



For narrow neural network layers the divergence is high.

HIGHER ORDER APPROXIMATION

- Higher order approximations of the network output (for notational simplicity $\theta \in \mathbb{R}$)

$$\hat{y}_t^{(N)}(X) := \sum_{n=0}^{N} \frac{\partial_\theta^n \hat{y}_t(X)}{n!}\bigg|_{\bar{\theta}} (\theta_t - \bar{\theta})^n.$$

- Under noisy training, the expected value of some function $h(\cdot)$ of the weights $u(t_0) = \mathbb{E}[h(\theta_t)|\theta_0]$ solves the following Cauchy problem, (Feynman-Kac)

$$(\partial_t + \mathcal{A}(\theta))u(t, \theta) = 0,$$

where

$$\mathcal{A} := -\eta \nabla_\theta \hat{y}_t(X)^\top (\nabla_{\hat{y}} \mathcal{L}(\hat{y}_t))\partial_\theta + \frac{1}{2}\sigma^2 \partial_\theta^2.$$

SOLVING THE CAUCHY PROBLEM

- Suppose we are interested in solving the following Cauchy problem

$$(\partial_t + \mathcal{A})u(t,\theta) = 0, \quad u(T,\theta) = h(\theta),$$

  where $\mathcal{A}$ depends on θ.
- This is not directly solvable!
- The idea is to choose an expansion $(\mathcal{A}_n(t))_{n\in\mathbb{N}}$ that closely approximates $\mathcal{A}(t)$, i.e.

$$\mathcal{A}(t,\theta) = \sum_{n=0}^{\infty} \mathcal{A}_n(t,\theta).$$

- We will use a Taylor expansion of this generator.

SOLVING THE CAUCHY PROBLEM

- Following the classical perturbation approach, the solution u can be expanded as an infinite sum,

$$u = \sum_{n=0}^{\infty} u^n.$$

- The N-th order approximation of u is then given by,

$$u^{(N)}(t,\theta) = \sum_{n=0}^{N} u^n(t,\theta).$$

- We then obtain the following sequence of nested Cauchy problems, for $x \in \mathbb{R}$,

$$(\partial_t + \mathcal{A}_0)u^0(t,\theta) = 0, \quad u^0(T,\theta) = h(\theta),$$

$$(\partial_t + \mathcal{A}_0)u^n(t,\theta) = -\sum_{k=1}^{n} \mathcal{A}_k(t,\theta)u^{n-k}(t,\theta), \quad u^n(T,\theta) = 0,\ n > 0.$$

- These are solvable!

HIGHER ORDER APPROXIMATION: ANALYTIC SOLUTIONS

We can solve the above Cauchy problem by using a Taylor expansion of the generator $\mathcal{A}$ [Lorig, Pascucci, Pagliarani (2015)]

Theorem

Consider the N-th order approximation of $\hat{y}_t$. The expected value of $\hat{y}_t^{(N)}$ is then given by

$$\mathbb{E}\left[\hat{y}_t^{(N)}(X)|\hat{y}_0\right] = \sum_{n=0}^{N} \frac{\partial_\theta^n \hat{y}_t(X)}{n!}\Big|_{\bar{\theta}} \sum_{n=0}^{2N-1} u_m^n(t_0,\theta),$$

with

$$u_m^0(t_0,\theta) = \partial_s^k \exp\left(\left(-\frac{\eta}{N}\mu_0(t-t_0)+\theta-\bar{\theta}\right)s+\frac{1}{2}\sigma^2(t-t_0)s^2\right)\Big|_{s=0},$$
$$u_m^n(t_0,\theta) = \mathcal{L}_n(0,t)u_m^0(t_0,\theta).$$

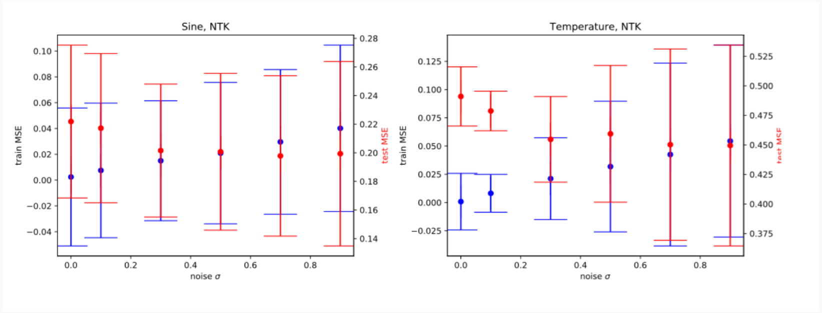
HIGHER ORDER APPROXIMATION: GENERALIZATION

- Having obtained the expected value of the weights, we can compute the Hessian of the loss function $\mathcal{L}$ with respect to the weights.
- We obtain,

$$\mathbb{E}\left[\Delta\mathcal{L}(\hat{y}_t)\right] = f\left(\partial_\theta^n \hat{y}_0(X), \sigma^2\right).$$

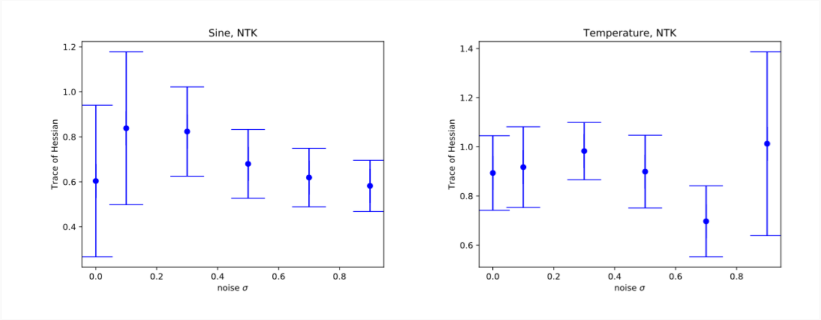
- The Hessian in case of higher order approximations is influenced by σ, so that the noise has an explicit effect on the network output.

HIGHER ORDER APPROXIMATION: THE EFFECTS OF NOISE



Increasing noise in the linear regime has little effect; in the non-linear regime we observe an effect.

HIGHER ORDER APPROXIMATION: THE EFFECTS OF NOISE



The weight Hessian is also affected: the more noise the smaller the weight Hessian, i.e. the better the generalization should be (and it is!)

CONCLUSION

- Understanding the network output is crucial to gain insight into generalization capabilities of the network but it is complicated due to the complex function map.
- We considered two approximations:
- Under the layer width tending to infinity, the model output can be approximated by a linear model.
- Here, noise does not explicitly regularize solution, noise only keeps model from fully converging
- Using a Taylor expansion approximation we obtained insight into higher order approximations of the network output
- In this case, i.e. when the network is non-linear in the weights, the noise during training has an explicit effect on the weight Hessian.