A THEORETICAL PERSPECTIVE ON THE

OPTIMISATION AND GENERALIZATION

OF DEEP NEURAL NETWORKS

WHAT IS OPTIMIZATION ?

Given **a Model Class *H***, **a dataset S of n points**, **a loss function L**, optimization is the problem of finding a **specific model** in H that produces the minimum **loss** (defined by L) on **average on the n points** in S.

WHAT IS OPTIMIZATION ?

Given **a Model Class *H***, **a dataset S of n points**, **a loss function L**, optimization is the problem of finding a **specific model** in H that produces the minimum **loss** (defined by L) on **average on the n points** in S.

$$f_S^* = \underset{f \in H}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L(f(x_i), y_i)$$

WHAT IS OPTIMIZATION ?

Given **a Model Class *H***, **a dataset S of n points**, **a loss function L**, optimization is the problem of finding a **specific model** in H that produces the minimum **loss** (defined by L) on **average on the n points** in S.

$$f_S^* = \underset{f \in H}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L(f(x_i), y_i)$$

- For a sufficiently large H, if the data and loss function behaves nicely, there will exist a minimizer.
- The purpose of the optimization algorithm is to find the minimiser with reasonable resources.
- However, not all optimization algorithms can guarantee this unless f (eg. convexity), S(eg. conditioning), and L (eg. differentiability) behave nicely.

WHAT IS GENERALIZATION ?

Given **a Model Class *H***, **a data distribution D**, **a loss function L**, generalisation is the problem of finding a **specific model** in H that produces the minimum **loss** (defined by L) on **average on the points** in D.

Statistically, this problem is more commonly referred to as risk minimisation.

WHAT IS GENERALIZATION ?

Given **a Model Class *H***, **a data distribution D**, **a loss function L**, generalisation is the problem of finding a **specific model** in H that produces the minimum **loss** (defined by L) on **average on the points** in D.

$$f_S^* = \underset{f \in \mathcal{H}}{\operatorname{argmin}} \; \mathbb{E}_{(x,y)\sim D} \, L(f(x), y)$$

Statistically, this problem is more commonly referred to as risk minimisation.

GENERALIZATION ERROR

$$\text{Risk}(f; D) = \mathbb{E}_{x,y \sim D}\left[(f(x) - y)^2\right]$$

$\tilde{f}_{S_n}$: Minimizes optimization error

$\hat{f}_{A,S_n}$: Solution returned by A on S_n

$f^*_{\mathcal{F}}$: $\underset{f \in \mathcal{F}}{\operatorname{argmin}}\ \text{Risk}(f; D)$

f^* : $\operatorname{argmin}\ \text{Risk}(f; D)$

GENERALIZATION ERROR

$$\text{Risk}(f;D) = \mathbb{E}_{x,y\sim D}\left[(f(x)-y)^2\right]$$

$$\text{Excess Risk}(f;D) = \text{Risk}(f;D) - \text{Risk}(f^*;D)$$

Learning objective :-

$$\boxed{\text{Minimize Excess Risk}}$$

$\tilde{f}_{S_n}$: Minimizes optimization error

$\hat{f}_{A,S_n}$: Solution returned by A on S_n

f_F^* : $\underset{f\in F}{\operatorname{argmin}}\ \text{Risk}(f;D)$

f^* : $\operatorname{argmin}\ \text{Risk}(f;D)$

$\longrightarrow$ Bayes optimal classifier

EXCESS RISK DECOMPOSITION

$$\text{Excess Risk}(\hat{f}_{A,S_n}) = \text{Risk}(\hat{f}_{A,S_n}) - \text{Risk}(f^*)$$

$$= \underbrace{\text{Risk}(\hat{f}_{A,S_n}) - \text{Risk}(f^*_F)}_{\text{Estimation Error}}$$

$$+ \underbrace{\text{Risk}(f^*_F) - \text{Risk}(f^*)}_{\text{Approximation Error}}$$

$$\approx 0$$

EXCESS RISK DECOMPOSITION

$$\text{Excess Risk}(\hat{f}_{A,S_n}) = \text{Risk}(\hat{f}_{A,S_n}) - \text{Risk}(f^*)$$

$$= \underbrace{\text{Risk}(\hat{f}_{A,S_n}) - \text{Risk}(f^*_F)}_{\text{Estimation Error}}$$

$$= \text{Risk}(\hat{f}_{A,S_n}) - \text{Risk}(\tilde{f}_{S_n}) \rightarrow \text{Optimization error}$$

$$+ \text{Risk}(\tilde{f}_{S_n}) - \text{Risk}(f_F^*) \rightarrow \text{Estimation error}$$

APPROACH I

$\text{Risk}(\tilde{f}_{S_n}) - \text{Risk}(f_F^*)$ → Estimation error

→ Is positive (Cannot be controlled)

Either choose S_n properly or

choose F to be very small

APPROACH I

$$\mathrm{Risk}(\tilde{f}_{S_n}) - \mathrm{Risk}(f_F^*) = \mathrm{Risk}(\hat{f}_{S_n}; D) - \mathrm{Risk}(\hat{f}_{S_n}; S)$$

$$+ \underbrace{\mathrm{Risk}(\hat{f}_{S_n}; S) - \mathrm{Risk}(f_F^*; S)}_{\leq 0}$$

$$+ \underbrace{\mathrm{Risk}(f_F^*; S) - \mathrm{Risk}(f_F^*)}_{O\left(\sqrt{1/n}\right)}$$

Control this

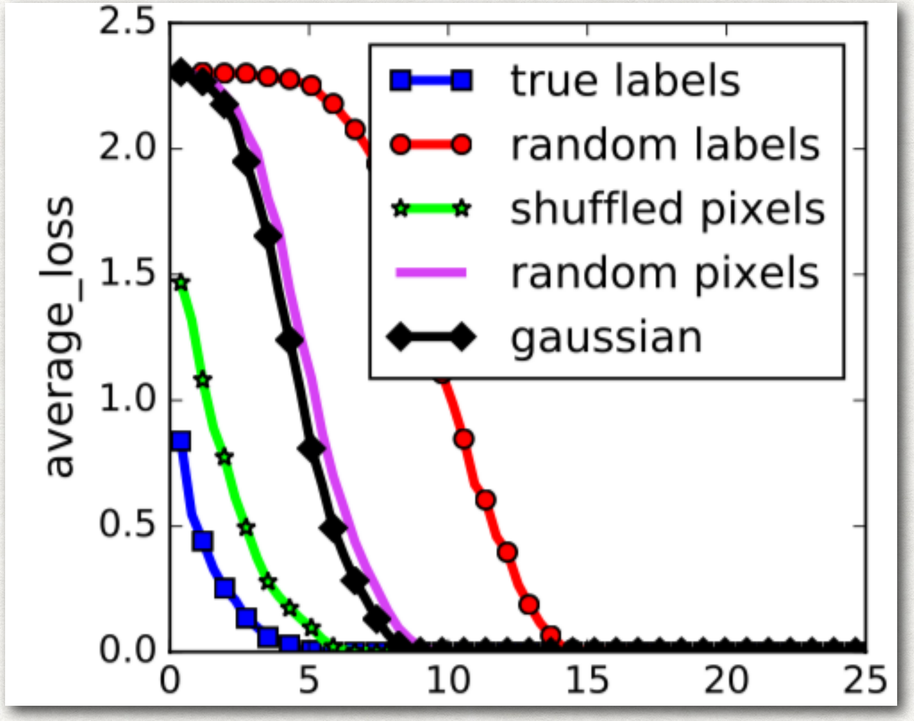
APPROACH I (REGULARIZATION)

$$\mathrm{Risk}(\hat{f}_{S_n}; D) - \mathrm{Risk}(\hat{f}_{S_n}; S) \leq \sup_{f \in \mathcal{F}} \Big(\mathrm{Risk}(f; D) - \mathrm{Risk}(f; S) \Big)$$

$$\Rightarrow \mathrm{Risk}(\hat{f}_{S_n}; D) \leq \underbrace{\mathrm{Risk}(\hat{f}_{S_n}; S)}_{\approx 0} + \sup_{f \in \mathcal{F}} \Big(\mathrm{Risk}(f; D) - \mathrm{Risk}(f; S) \Big)$$

APPROACH I (REGULARIZATION)

$$\text{Risk}(\hat{f}_{S_n}; D) \leq \text{Risk}(\hat{f}_{S_n}; S) + \sup_{f \in F} \underbrace{\left(\text{Risk}(f; D) - \text{Risk}(f; S)\right)}_{\text{Term 2}}$$



Zhang et. al. 2017

Problem: Term 2 can be very large for the same architecture and algorithm.

Theory question: Can we bound Term 2 given a trained model on the dataset ?

Experimental Question: Can we try to measure the maximum value of Term 2 by designing experiments ?

Why: We can then guarantee how the architecture + algorithm will perform on unseen data.

TRADITIONAL METHODS

$$\sup_{f \in \mathcal{F}} \left(\text{Risk}(f; D) - \text{Risk}(f; S) \right) = \tilde{O}\left(\sqrt{\frac{\text{Complexity}}{(\text{Margin})^2}} \right)$$

Traditionally people have measured this bounds using Complexity Measures like

- VC Dimension
- Rademacher Complexity
- Covering Number
- Fat Shatterim Dimension
- Pseudo Dimension etc.

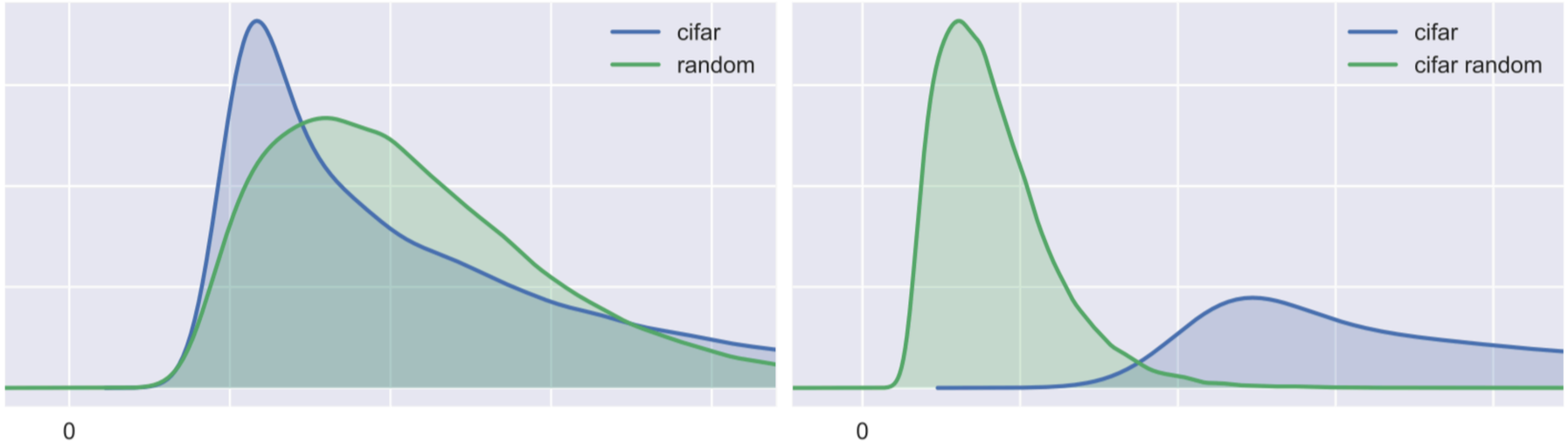
But it is not trivial to measure them without being too pessimistic for neural networks.

MARGIN BOUNDS

$$\sup_{f \in F} \left(\text{Risk}(f; D) - \text{Risk}(f; S) \right) \leq O\left(\frac{\prod_i \|W_i\|_2^2 \sum \frac{\|W_i\|_F^2}{\|W_i\|_2^2}}{(\text{margin})^2} \right)$$

Q: Can the margin predict the gap ?

Ans: No! But if properly normalised with weight parameters, it can. Why ?



(a) Margins. (b) Normalized margins.

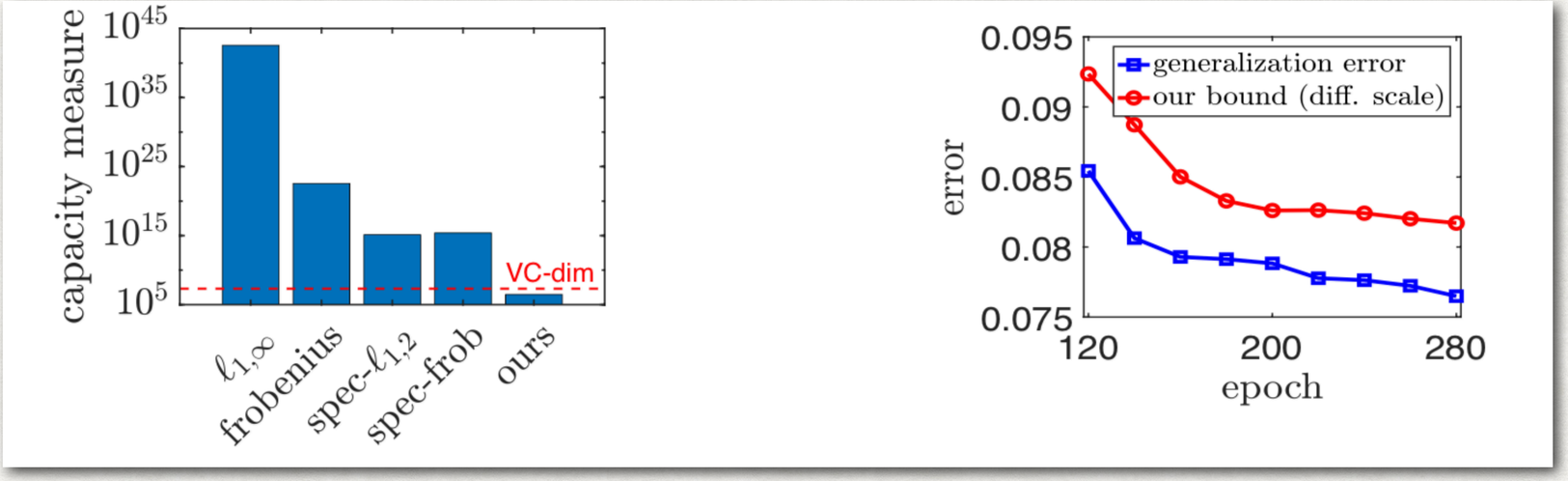
Bartlett et. al. 2017

Neyshabur et. al. 2017

COMPRESSION BASED BOUNDS

$$\sup_{f \in \mathcal{F}} \left(\text{Risk}(f; D) - \text{Risk}(f; S) \right) \leq O\left(\sqrt{\frac{\text{Complexity}}{(\text{margin})^2}} \right)$$

Q: But networks seem to have a very large complexity!
Ans: Maybe develop a smaller network which behaves exactly the same way as the larger one and then report the complexity of the smaller network.



Arora et. al. 2018

APPROACH II (IMPLICIT REGULARIZATION)

$\text{Risk}(\tilde{f}_{S_n}) - \text{Risk}(f_F^*) \rightarrow$ Estimation error

$\hookrightarrow$ Is positive (Cannot be controlled)

$\text{Risk}(\hat{f}_{A,S_n}) - \text{Risk}(\tilde{f}_{S_n}) \rightarrow$ Optimization error

$\hookrightarrow$ • Can be negative

• Can be controlled using A.

APPROACH II (IMPLICIT REGULARIZATION)

Questions:

- What inductive bias does Gradient Descent impart to the model ?
- Can we bound **Optimization Error** to be very negative if we take into account that gradient Descent is being used ?
- Is it possible to say what the generelization error will be without even training the network ?

APPROACH II (IMPLICIT REGULARIZATION)

Questions:

- What inductive bias does Gradient Descent impart to the model ? Ans: Learn signals over noise
- Can we bound **Optimization Error** to be very negative if we take into account that gradient Descent is being used ? Ans: Yes!
- Is it possible to say what the generelization error will be without even training the network ? Ans: Yes!

NEURAL TANGENT KERNEL

Main Idea:

- Neural Networks behave almost like a **Linear Model (more precisely like a Kernel).**
- The Kernel is called the **Neural Tangent Kernel (NTK).**
- To evaluate the NTK you need the **Data** and the **Network Architecture.**

$$NTK = \mathbb{E}_{\omega \sim N(0,I)} \left[\left(\frac{\partial F(\omega, X)}{\partial X} \right) \left(\frac{\partial F(\omega, X)}{\partial X} \right)^T \right]$$

NEURAL TANGENT KERNEL

Main Idea:

- Neural Networks behave almost like a **Linear Model (more precisely like a Kernel).**
- The Kernel is called the **Neural Tangent Kernel (NTK).**
- To evaluate the NTK you need the **Data** and the **Network Architecture.**

$$\mathrm{NTK} = H^{\infty} = \mathbb{E}_{\omega \sim N(0,I)} \left[\left(\frac{\partial F(\omega, x)}{\partial x} \right) \left(\frac{\partial F(\omega, x)}{\partial x} \right)^{T} \right]$$

i.e. the **Expected Jacobean Outer Product under random initialisation**

NTK BASED BOUNDS

$$\text{Loss at } K^{th} \text{ time step} \approx \frac{1}{2} \left\| (I - \eta H^{\infty})^{K} y \right\|_2^2$$

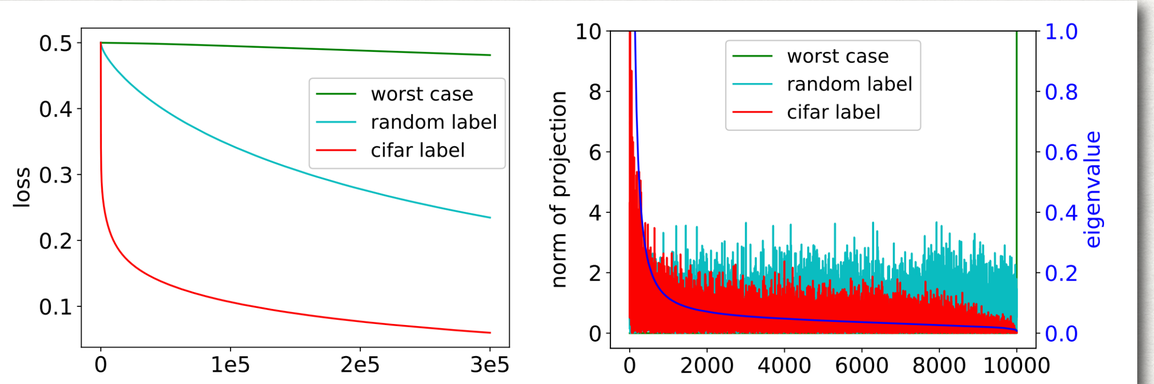
Depending on the labels y, the convergence can be fast or slow.

$$\text{Generalization Error} \approx O\left(\sqrt{\frac{y^T (H^{\infty})^{-1} y}{n}} \right)$$

You can guess how the generelization will be without training the network.

NTK BASED BOUNDS

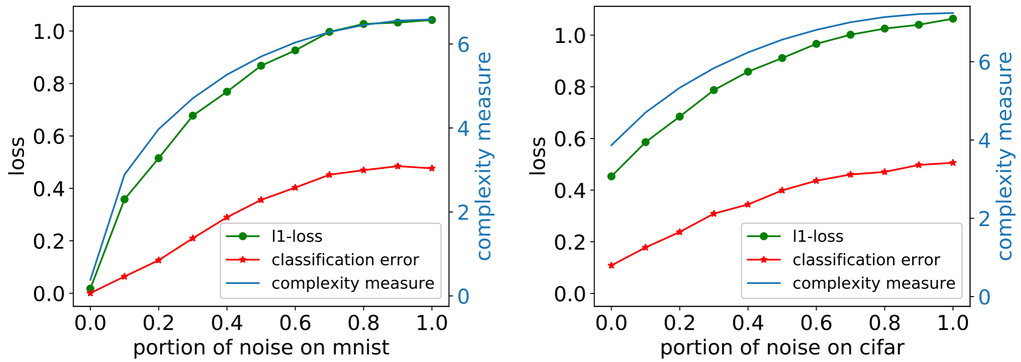
Loss at K^{th} time step $\approx \frac{1}{2} \| (I - \eta H^{\infty})^{K} y \|_2^2$



(c) Convergence Rate, CIFAR. (d) Eigenvalues & Projections, CIFAR.

Arora et. al. 2019

Generalization Error $\approx O\left(\sqrt{\frac{y^T (H^{\infty})^{-1} y}{n}}\right)$



(a) MNIST Data. (b) CIFAR Data.

Arora et. al. 2019

LOW RANK JACOBIAN

Hypothesis (Somewhat proven): Signals lie on the top eigenvectors of the NTK and noise lies on the bottom eigenvectors.

Known: Optimization happens fast on the space covered by the top eigenvectors.

Known: Noise cannot be generalised. Thus generelization also happens only when the model only learns the top eigenvectors.

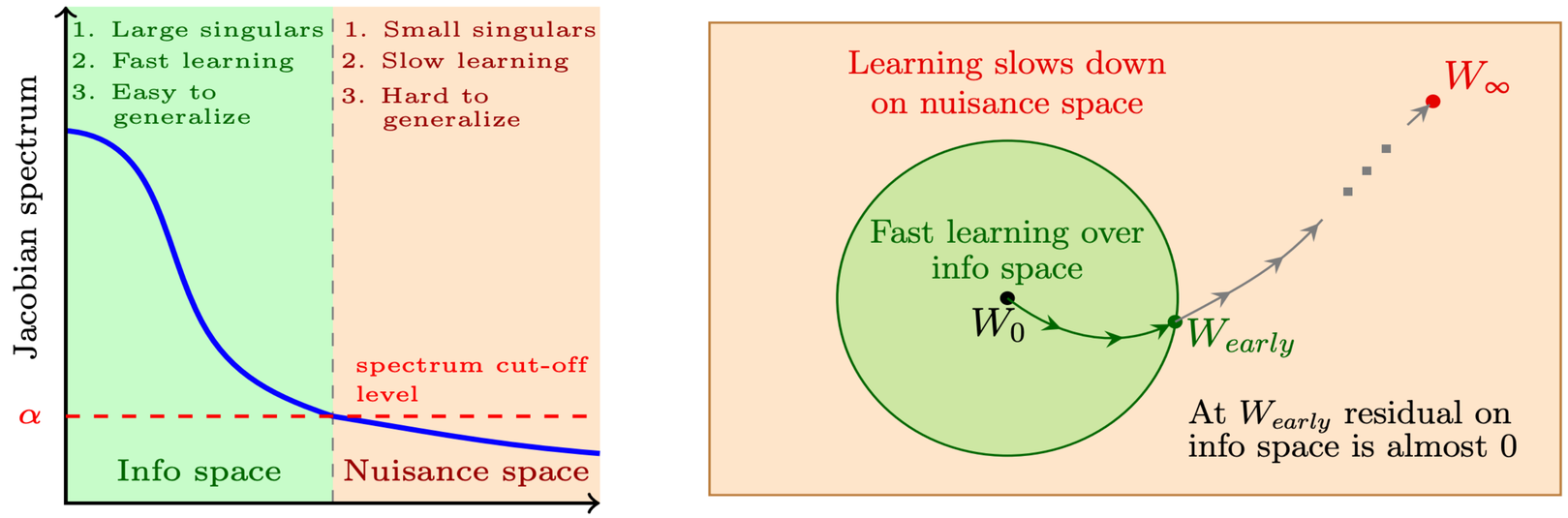
Takeaway 1: So, for a model to generalise better **it should learn fast**. If it takes more time, it is memorising noise.

Takeaway 2: If it takes more time, the model parameters will move further away from initialisation. So, to learn noise it needs to **move further away from initialisation**.

LOW RANK JACOBIAN

Takeaway 1: So, for a model to generalise better **it should learn fast**. If it takes more time, it is memorising noise.

Takeaway 2: If it takes more time, the model parameters will move further away from initialisation. So, to learn noise it needs to **move further away from initialisation**.



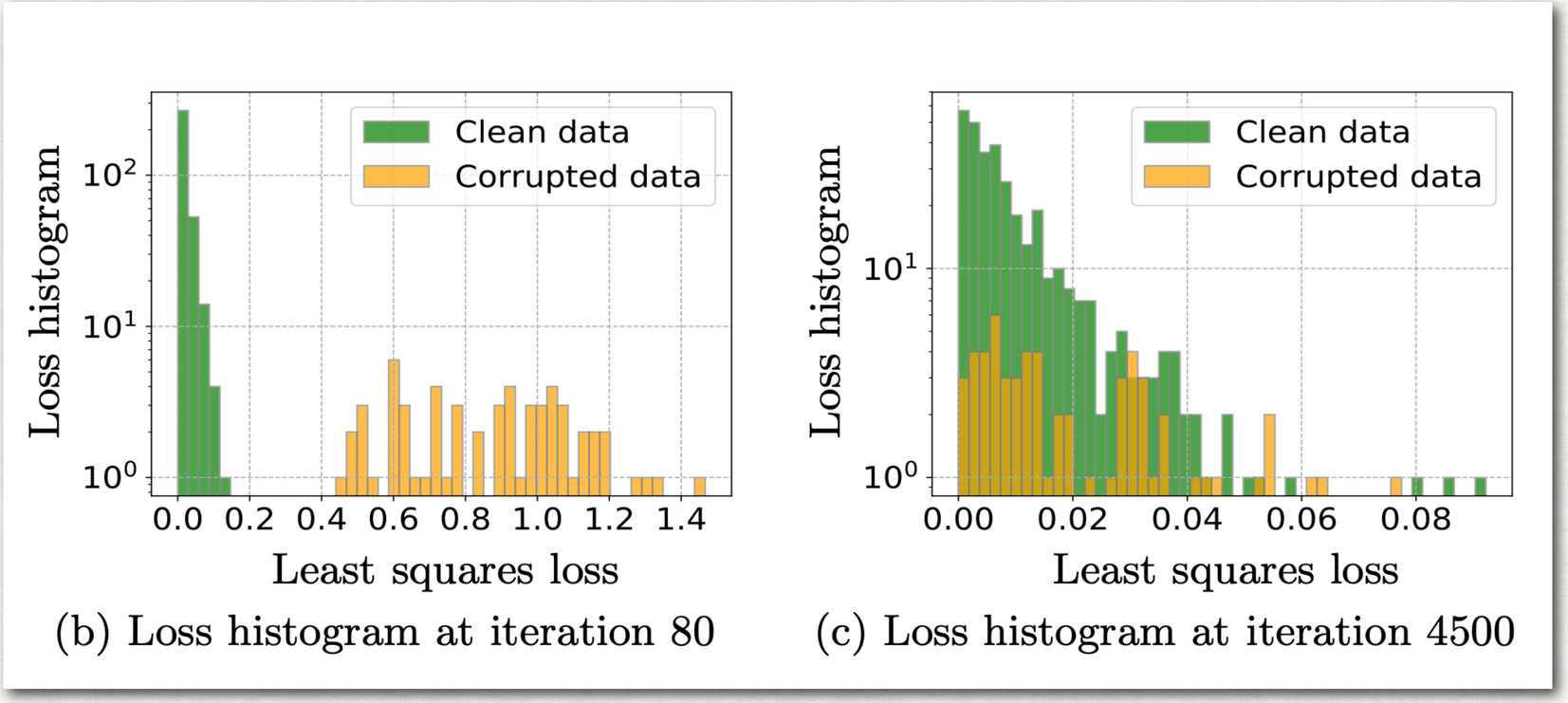
(a) Depiction via the Jacobian spectrum

(b) Depiction in parameter space

Oymak et. al. 2019

LOW RANK JACOBIAN

Takeaway 1: So, for a model to generalise better **it should learn fast.** If it takes more time, it is memorising noise.

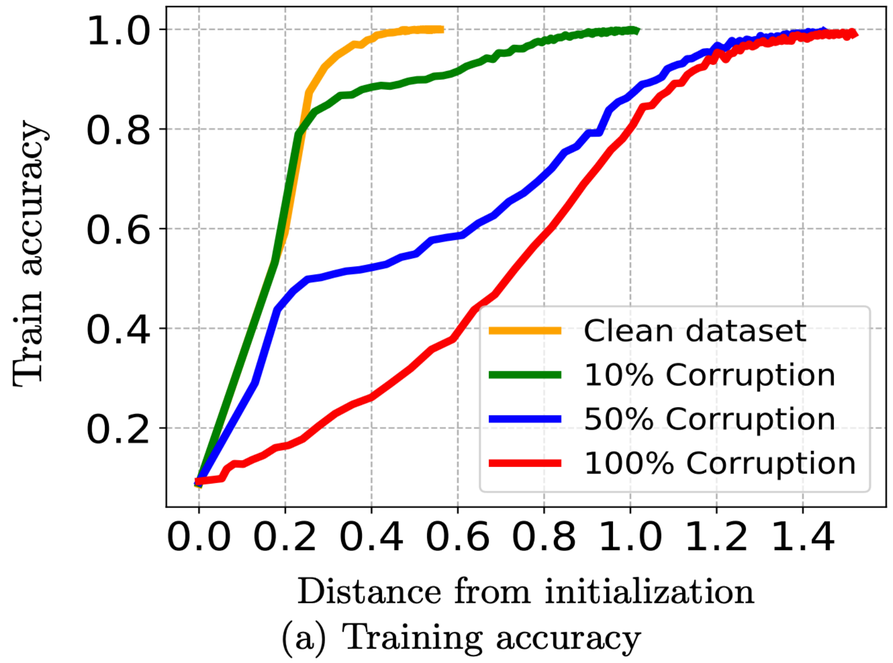


(b) Loss histogram at iteration 80 (c) Loss histogram at iteration 4500

Li et. al. 2019

LOW RANK JACOBIAN

Takeaway 2: If it takes more time, the model parameters will move further away from initialisation. So, to learn noise it needs to **move further away from initialisation**.



(a) Training accuracy

Li et. al. 2019