Lipschitz-bounded Feed-through Networks (TBD)

Anonymous ECCV 2024 Submission

Paper ID #*****

Abstract. TBD

Keywords: Lipschitz networks · NeRF · Third keyword

1 Introduction

2 Lipschitz-bounded Feed-through Networks

2.1 Feed-through Networks

We consider an L-layer feed-through networks of the form

$$z_k = \sigma(W_k z_{k-1} + U_k x + b_k), \quad y = \mu x + \sum_{k=1}^{L} Y_k z_k + b_y \tag{1}$$

where $z_k \in \mathbb{R}^{m_k}$ with $z_0 = 0$ are the hidden variables, $x, y \in \mathbb{R}^n$ are the input and output variables, respectively. Here U_k, W_k, Y_k and b_k, b_y are the learnable weights and biases, respectively. The above model has a compact representation

$$z = \sigma(Wz + Ux + b), \quad y = \mu x + Yz + b_y \tag{2}$$

where $z = \begin{bmatrix} z_1^\top & \cdots & z_L^\top \end{bmatrix}^\top$, $b = \begin{bmatrix} b_1^\top & \cdots & b_L^\top \end{bmatrix}^\top$, and

$$W = \begin{bmatrix} 0 & & & \\ W_2 & 0 & & \\ & \ddots & \ddots & \\ & & W_L & 0 \end{bmatrix}, \quad U = \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_L \end{bmatrix}, \quad Y = \begin{bmatrix} Y_1 \; Y_2 \cdots \; Y_L \end{bmatrix}.$$

TODO: explain why this architecture does not suffer from vanishing gradient issues

2.2 Lipschitz Model Parameterization

Theorem 1. *The neural network* (2) *is* (μ, ν)*-Lipschitz if there exists a* $\Lambda \in \mathbb{D}_+^m$*, where* $\mathbb{D}_+^m$ *is the set of positive diagonal matrices, such that the following condition holds with* $\gamma = \nu - \mu$*:*

$$Y = U^\top \Lambda, \quad 2\Lambda - \Lambda W - W^\top \Lambda \succeq \frac{2}{\gamma} Y^\top Y. \tag{3}$$

Let Θ be the set of all $\theta = \{\Lambda, U, W, Y\}$ such that Condition (3) holds. Since it is generally not scalable to train a model with SDP constraints, we instead construct a smooth *direct parameterization* $\mathcal{M} : \mathbb{R}^N \to \Theta$ such that $\mathcal{M}(\mathbb{R}^N) = \Theta$. With such parameterization, we can use standard unconstrained optimization algorithms to train the free parameter $\phi \in \mathbb{R}^N$.

To construct $\mathcal{M}$, we first introduce the free parameter

$$\phi = \{d, F_k^a, F_k^b, F^q, F^\star\}, \quad k = 1, \ldots, L \tag{4}$$

where $d \in \mathbb{R}^m$, $F_k^a \in \mathbb{R}^{m_k \times m_k}$, $F_k^b \in \mathbb{R}^{m_{k-1} \times m_k}$, $F^q \in \mathbb{R}^{m \times n}$ and $F^\star \in \mathbb{R}^{n \times n}$ with $m_0 = 0$. Then, we compute some intermediate variables $\Psi = \operatorname{diag}\big(e^\psi\big)$ and

$$\begin{bmatrix} A_k^\top \\ B_k^\top \end{bmatrix} = \operatorname{Cayley}\left(\begin{bmatrix} F_k^a \\ F_k^b \end{bmatrix}\right), \quad \begin{bmatrix} Q \\ \star \end{bmatrix} = \operatorname{Cayley}\left(\begin{bmatrix} F^q \\ F^\star \end{bmatrix}\right)$$

where $\operatorname{Cayley} : \mathbb{R}^{n \times p} \to \mathbb{R}^{n \times p}$ with $n \geq p$ is defined by

$$J = \operatorname{Cayley}\left(\begin{bmatrix} G \\ H \end{bmatrix}\right) := \begin{bmatrix} (I+Z)(I-Z)^{-1} \\ -2V(I-Z)^{-1} \end{bmatrix}$$

where $Z = G^\top - G + H^\top H$. It is easy to verify that $J^\top J = I$ for any $G \in \mathbb{R}^{p \times p}$ and $H \in \mathbb{R}^{(n-p) \times p}$. We finally construct $\theta = \mathcal{M}(\phi)$ as follows:

$$\begin{aligned} \Lambda_k &= 1/2\Psi_k^2, \quad W_k = 2\Psi_k^{-1} B_k A_{k-1}^\top \Psi_{k-1}, \\ U_k &= \sqrt{2\gamma}\Psi_k^{-1}(A_k Q_k - B_k Q_{k-1}), \\ Y_k &= \sqrt{\gamma/2}(A_k Q_k - B_k Q_{k-1})^\top \Psi_k \end{aligned} \tag{5}$$

where $B_1 = 0$ and $Q_0 = 0$. The following proposition shows that we can learn the free parameter ϕ without any loss of model expressivity.

Proposition 1. *Let $\mathcal{M}$ be defined by (4) and (5). We have $\mathcal{M}(\mathbb{R}^N) = \Theta$.*

2.3 Modular forward computation

We rewrite (1) as follows

$$\begin{aligned} z_k &= \sigma(2\Psi_k^{-1} B_k A_{k-1}^\top \Psi_{k-1} z_{k-1} + \sqrt{2\gamma}\Psi_k^{-1}(A_k Q_k - B_k Q_{k-1})x + b_k) \\ y &= \mu x + \sum_{k=1}^{L} \sqrt{\gamma/2}(A_k Q_k - B_k Q_{k-1})^\top \Psi_k z_k + b_y \end{aligned}$$

Note that the model parameters are shared by neighborhood layers. To make the implementation in a modular way, we introduce

$$\hat{b} = \Psi b, \; \hat{x} = \sqrt{\gamma} Q x, \; h_k = \sqrt{2} A_k^\top \Psi_k z_k - \hat{x}_k, \; g_k = \sqrt{2} B_k^\top \Psi_k z_k, \; \hat{y}_k = h_k - g_{k+1}$$

with $g_{L+1} = 0$. Then, the proposed network (1) can be rewritten as

$$\begin{aligned} \begin{bmatrix} h_k \\ g_k \end{bmatrix} &= \sqrt{2} R_k^\top \hat{\sigma} \left(\sqrt{2} R_k \begin{bmatrix} \hat{x}_k \\ h_{k-1} \end{bmatrix} + \hat{b}_k \right) - \begin{bmatrix} \hat{x}_k \\ 0 \end{bmatrix} \\ y &= \mu x + \frac{\sqrt{\gamma}}{2} Q^\top (\hat{x} + \hat{y}) + b_y = \frac{\nu + \mu}{2} x + \frac{\sqrt{\gamma}}{2} Q^\top \hat{y} + b_y \end{aligned} \tag{6}$$

where $R_k = \begin{bmatrix} A_k & B_k \end{bmatrix}$. Here $\hat{\sigma}(x) := \Psi \sigma(\Psi^{-1} x)$ is a monotone and 1-Lipschitz activation with learnable scaling Ψ. Note that for ReLU activation we have $\hat{\sigma}(\cdot) = \sigma(\cdot)$, *i.e.* no need to learn the scaling factor Ψ.

3 Related Work

4 Conclusion

References