ReZero is All You Need: Fast Convergence at Large Depth

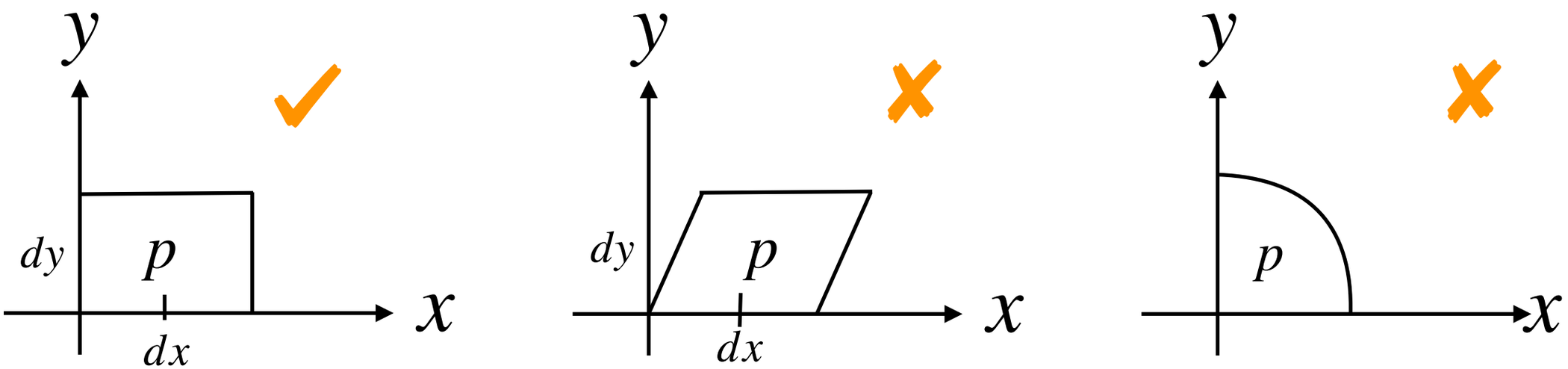
Thomas Bachlechner,* Bodhisattwa Prasad Majumder,* Huanru Henry Mao,*
Garrison W. Cottrell, Julian McAuley
UC San Diego
{tbachlechner@physics, bmajumde@eng, hhmao@eng, gary@eng, jmcauley@eng}
.ucsd.edu

- Jacobian matrix
- Singular Value Decomposition
- ReZero
- Evaluation

Jacobian matrix

(Iterated Integral)

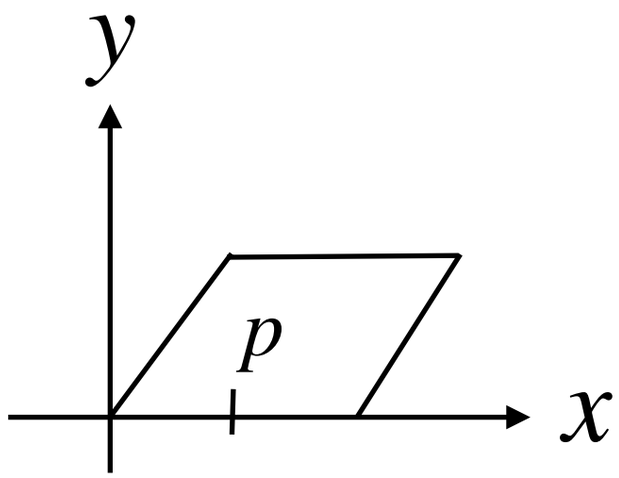
$\iint_p f(x,y)dxdy$: The summation of $f(x,y)$ under area p, where p is produced by dx and dy.



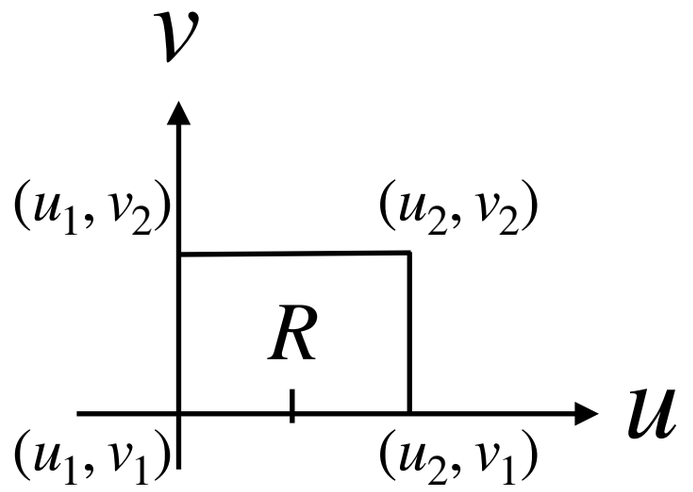
$\int_0^1 \int_0^2 f(x,y)dxdy$ $\int_0^1 \int_?^? f(x,y)dxdy$ $\int_?^? \int_?^? f(x,y)dxdy$

Find a relationship between two coordinate systems.

Jacobian matrix



$f : R^2 \to R^2$



$$\iint_p f(x,y)dxdy$$

$$x = u + v$$
$$y = v$$

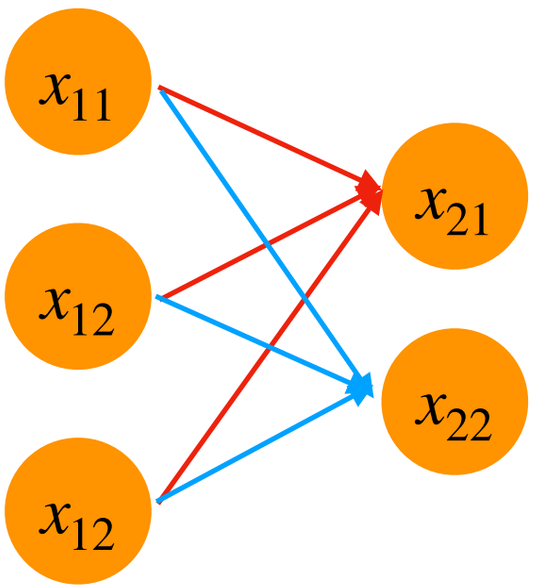
$$\int_{v_1}^{v_2}\int_{u_1}^{u_2} g(u,v)[\frac{\partial(x,y)}{\partial(u,v)}]dudv$$

$$J = [\frac{\partial(x,y)}{\partial(u,v)}] = \begin{bmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} = 1$$

$$\Rightarrow \int_{v_1}^{v_2}\int_{u_1}^{u_2} g(u,v)dudv$$

Jacobian matrix

$$f : R^3 \rightarrow R^2$$



Matrix format :

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \end{bmatrix} = \begin{bmatrix} x_{21} \\ x_{22} \end{bmatrix}$$

Equation format :

$$x_{21} = x_{11}w_{11} + x_{12}w_{12} + x_{13}w_{13}$$
$$x_{22} = x_{11}w_{21} + x_{12}w_{22} + x_{13}w_{23}$$

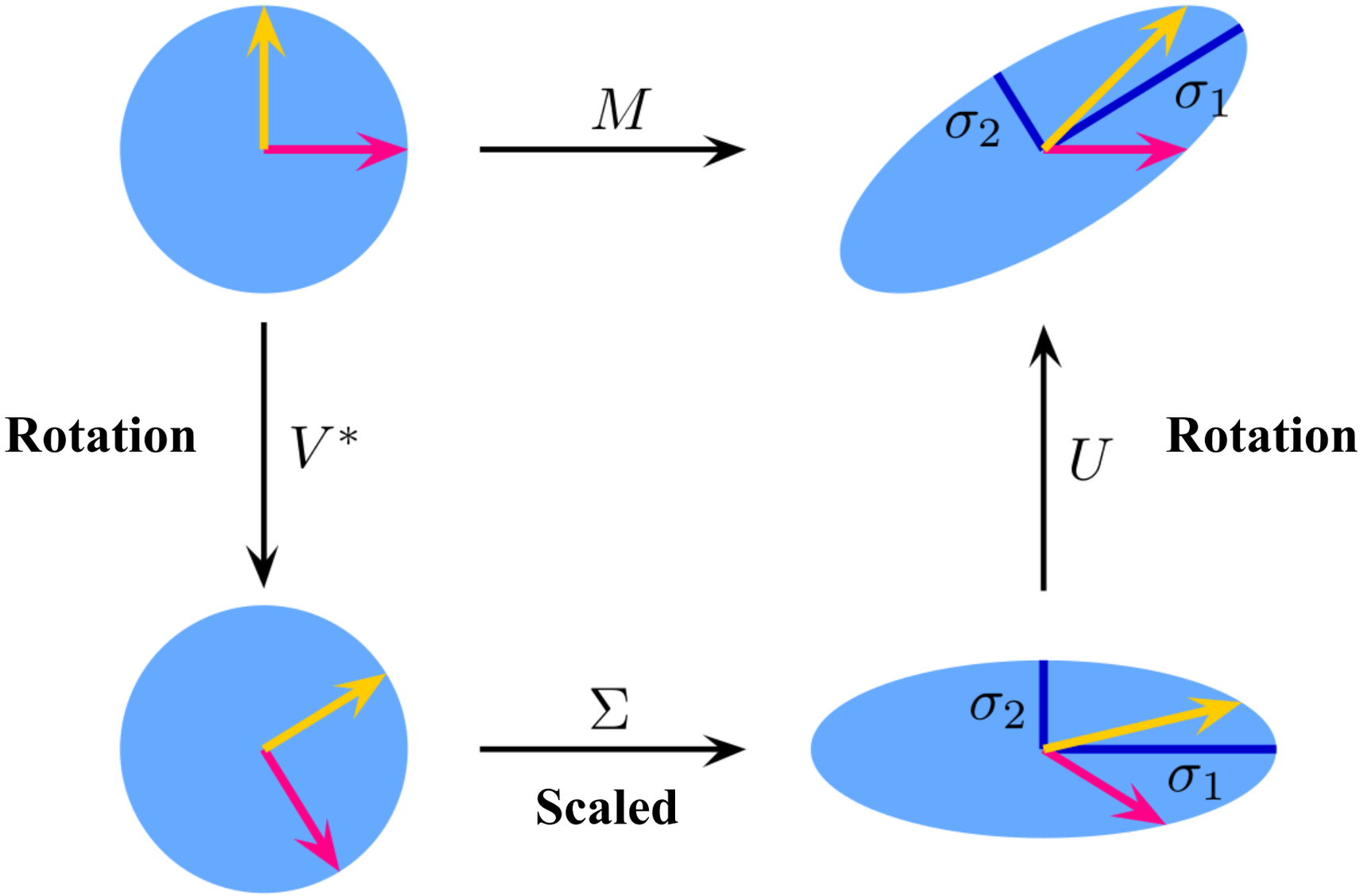
$$J_{io} = \left[\frac{\partial(x_{21}, x_{22})}{\partial(x_{11}, x_{12}, x_{13})}\right] = \begin{bmatrix} \frac{\partial x_{21}}{\partial x_{11}} & \frac{\partial x_{21}}{\partial x_{12}} & \frac{\partial x_{21}}{\partial x_{13}} \\ \frac{\partial x_{22}}{\partial x_{11}} & \frac{\partial x_{22}}{\partial x_{12}} & \frac{\partial x_{22}}{\partial x_{13}} \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix}$$

As same as gradient.

Suppose no activation.

The computation of gradient is the same as Jacobian.

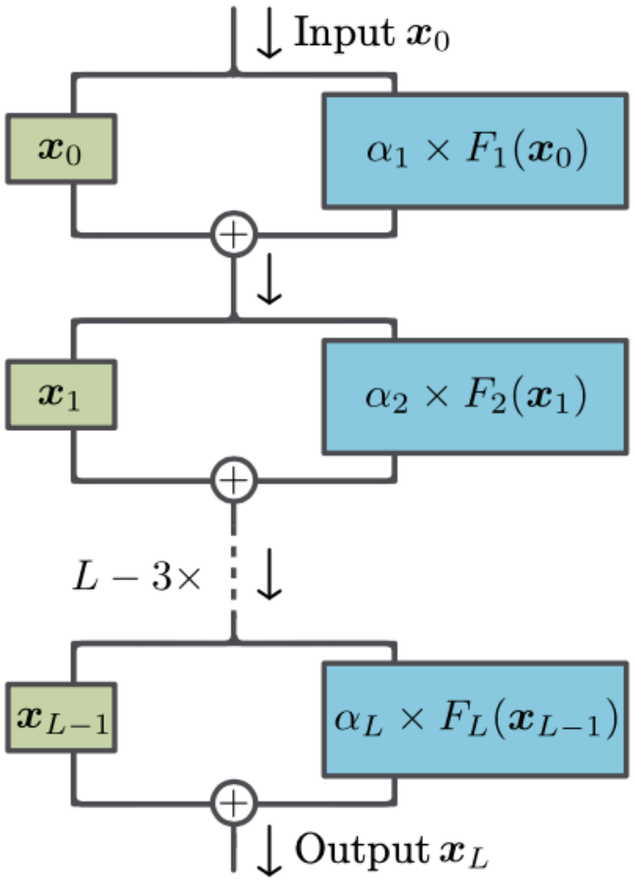
Single Value Decomposition



$$M = U \cdot \Sigma \cdot V^*$$

Source: https://en.wikipedia.org/wiki/Singular_value_decomposition

ReZero



$$x_{i+1} = x_i + \alpha_i F(x_i)$$

Suppose no activation:

$$\begin{aligned} z_1 &= wx_0 + x_0 \\ z_2 &= wz_1 + z_1 \\ z_3 &= wz_2 + z_2 \\ &\cdots \\ z_L &= wz_{L-1} + z_{L-1} \end{aligned}$$

with ReZero →

$$\begin{aligned} z_1 &= \alpha_0 wx_0 + x_0 \\ z_2 &= \alpha_1 wz_1 + z_1 \\ z_3 &= \alpha_2 wz_2 + z_2 \\ &\cdots \\ z_L &= \alpha_{L-1} wz_{L-1} + z_{L-1} \end{aligned}$$

w/o ReZero:

$$\begin{aligned} \frac{\partial z_L}{\partial z_1} &= \frac{\partial z_L}{\partial z_{L-1}} \frac{\partial z_{L-1}}{\partial z_{L-2}} \cdots \frac{\partial z_3}{\partial z_2} \frac{\partial z_2}{\partial z_1} \\ &= \underline{(w+1)(w+1)\cdots(w+1)(w+1)} \end{aligned}$$

With ReZero:

$$\begin{aligned} \frac{\partial z_L}{\partial z_1} &= \frac{\partial z_L}{\partial z_{L-1}} \frac{\partial z_{L-1}}{\partial z_{L-2}} \cdots \frac{\partial z_3}{\partial z_2} \frac{\partial z_2}{\partial z_1} \\ &= \underline{(\alpha_{L-1}w+1)(\alpha_{L-2}w+1)\cdots(\alpha_2 w+1)(\alpha_1 w+1)} \end{aligned}$$

Batch_Normalization BP: https://kratzert.github.io/2016/02/12/understanding-the-gradient-flow-through-the-batch-normalization-layer.html

ReZero

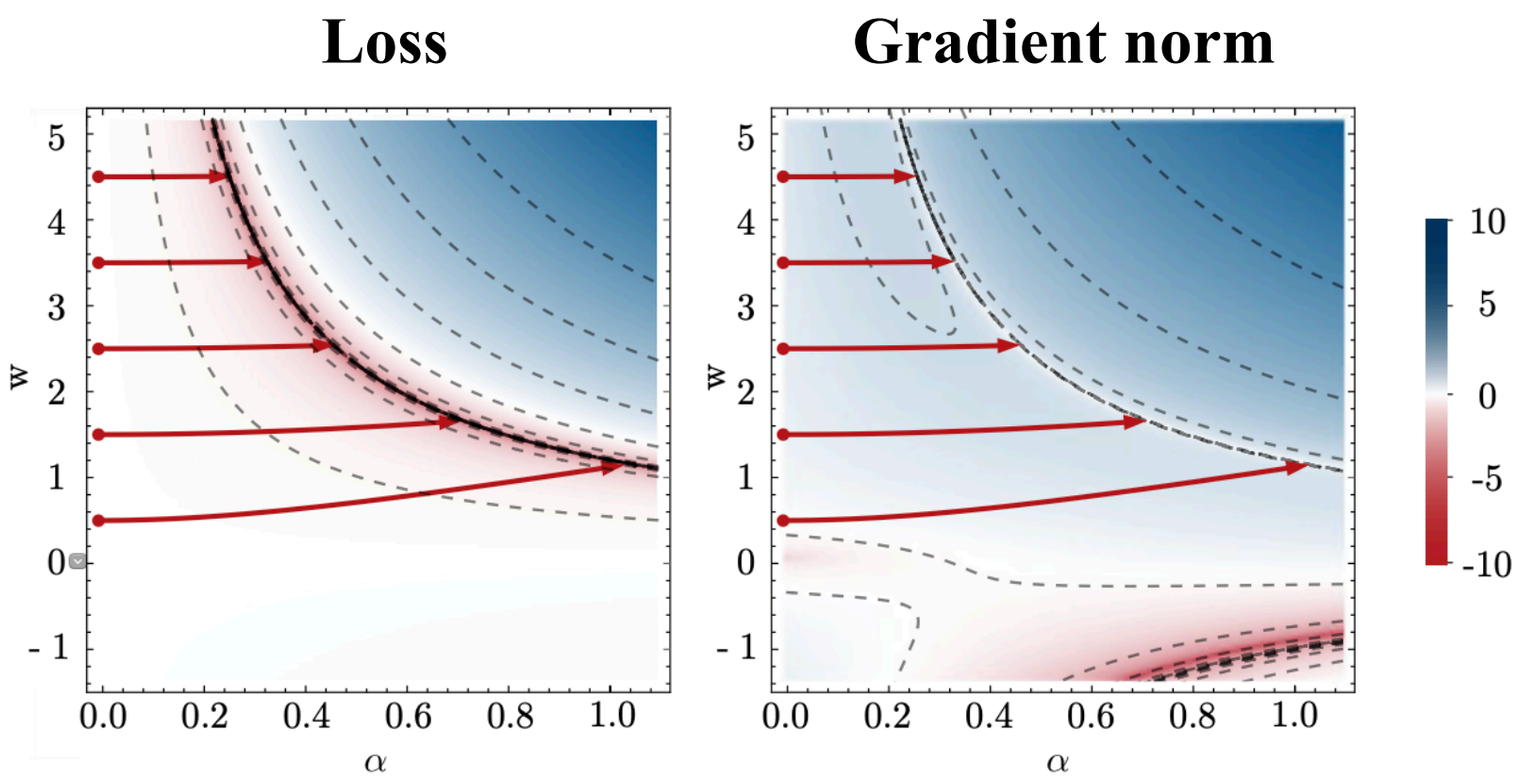
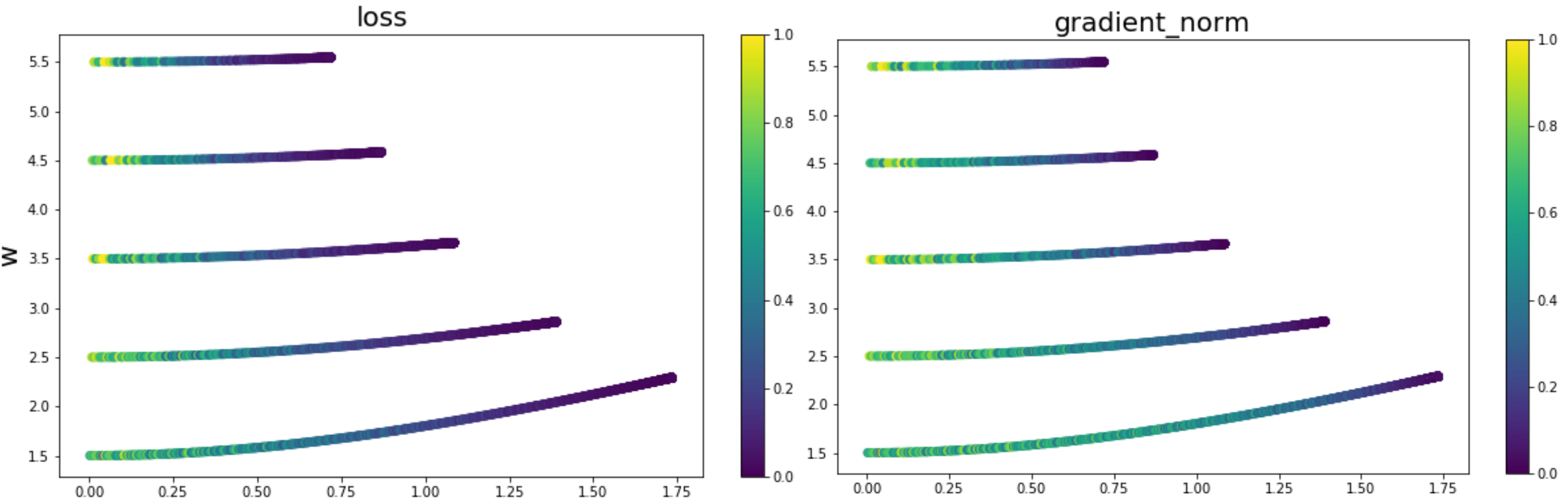


Figure 2: Contour log plots of a quadratic cost function (left) and gradient norm (right) over the network weight w and the residual weight α during the training of the linear function $x_L = 5x_0$ via gradient descent using a training set of $x_0 = \{1, 2, 3\}$. Gradient descent trajectories initialized at $\alpha = 0$ are shown in red for five different initial w's. The trajectory dynamics avoid the poorly conditioned regions around $\alpha \approx 1$.

ReZero




```python
class simple_ReZero(tf.keras.Model):
    def __init__(self, alpha_init=0., w_init=1.):
        super().__init__()

        self.resweight = tf.Variable([alpha_init,], trainable=True)

        self.w = tf.Variable([w_init,], trainable=True)

    def call(self, x, resweight=True):
        if resweight:
            y = x + self.resweight * x * self.w
        else:
            y = x + x * self.w
        return self.resweight, self.w, y
```

Evaluation

Table 1: Various forms of normalization and residual connections. F represents the transformation of an arbitrary layer and "Norm" is a normalization (e.g., LayerNorm or BatchNorm).

(1) Deep Network	$\boldsymbol{x}_{i+1} = F(\boldsymbol{x}_i)$
(2) Residual Network	$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + F(\boldsymbol{x}_i)$
(3) Deep Network + Norm	$\boldsymbol{x}_{i+1} = \text{Norm}\big(F(\boldsymbol{x}_i)\big)$
(4) Residual Network + Pre-Norm	$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + F(\text{Norm}\big(\boldsymbol{x}_i\big))$
(5) Residual Network + Post-Norm	$\boldsymbol{x}_{i+1} = \text{Norm}\big(\boldsymbol{x}_i + F(\boldsymbol{x}_i)\big)$
(6) **ReZero**	$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i F(\boldsymbol{x}_i)$

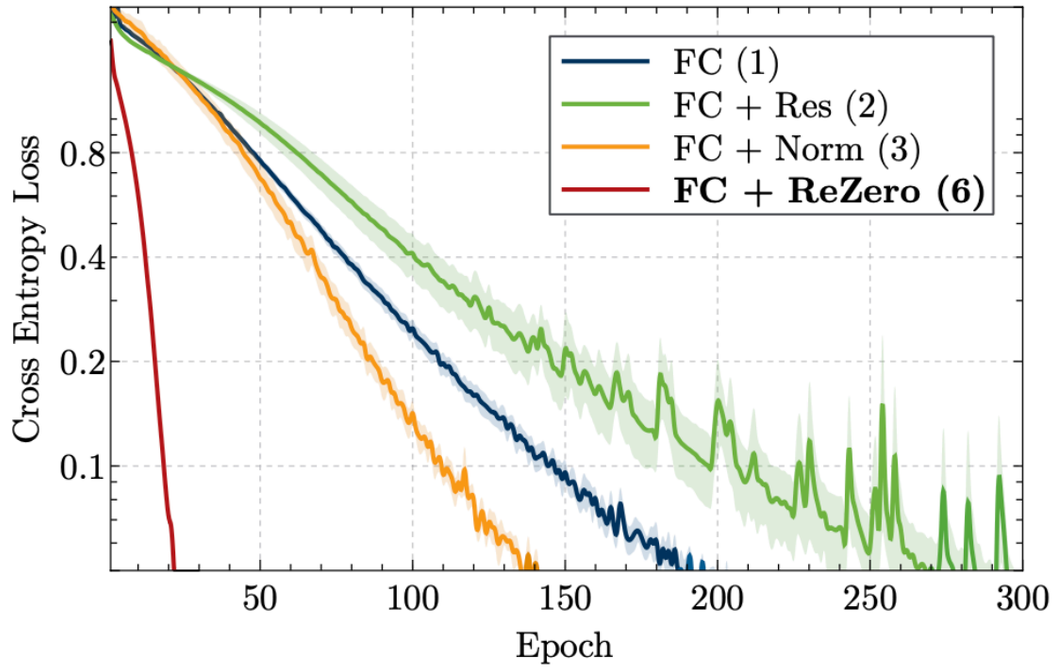
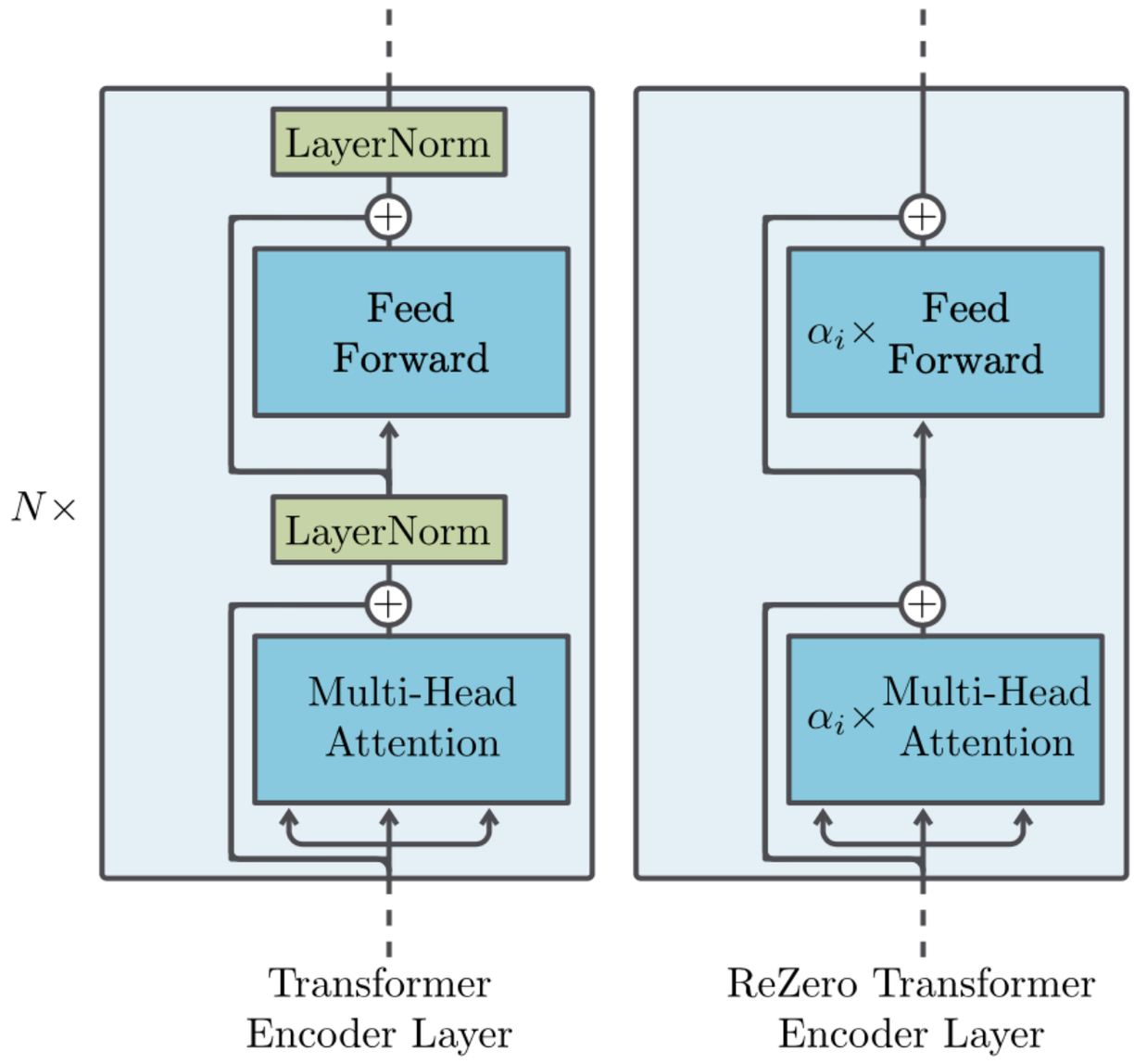


Figure 3: Cross entropy loss during training of four variants of 32 layer fully connected networks with width 256 and ReLU activations. The bracketed numbers refer to the architectures in the corresponding rows of Table 1. We average over five runs each and show 1σ error bands. For all models we use the Adagrad [22] optimizer with learning rate 0.01.

Evaluation



$$x_{i+1} = LayerNorm(x_i + sublayer(x_i))$$

$$sublayer \in \{self - attention, feed - forward\}$$

$$x_{i+1} = x_i + \alpha_i \, sublayer(x_i)$$

Evaluation

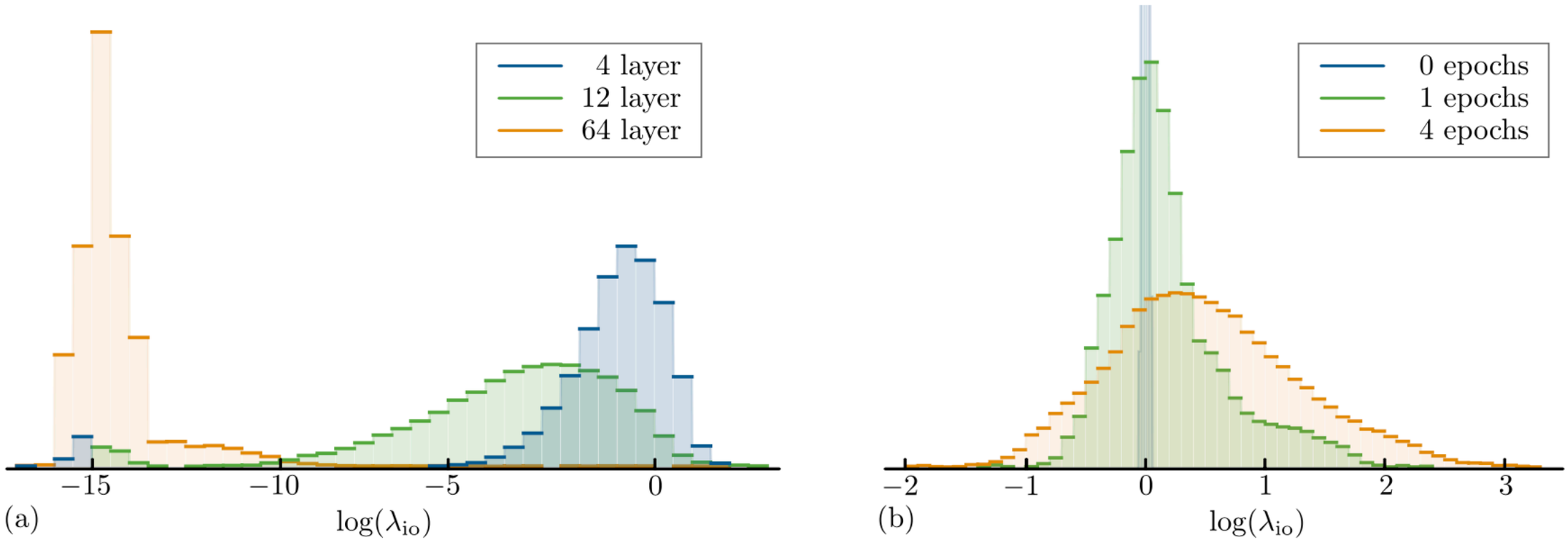


Figure 5: Histograms for log singular values λ_{io} of the input-output Jacobian matrix for: (a) Transformer encoder network at initialization of depths 4, 12 and 64 layers; (b) ReZero Transformer encoder network with 64 layers before and during training. Deep Transformers are far from dynamical isometry, $\lambda_{io} \ll 1$, while ReZero Transformers remain closer to dynamical isometry with mean singular value $\lambda_{io} \approx 1$.

Evaluation

Table 1: Various forms of normalization and residual connections. F represents the transformation of an arbitrary layer and "Norm" is a normalization (e.g., LayerNorm or BatchNorm).

(1) Deep Network	$\boldsymbol{x}_{i+1} = F(\boldsymbol{x}_i)$
(2) Residual Network	$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + F(\boldsymbol{x}_i)$
(3) Deep Network + Norm	$\boldsymbol{x}_{i+1} = \text{Norm}\big(F(\boldsymbol{x}_i)\big)$
(4) Residual Network + Pre-Norm	$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + F(\text{Norm}\big(\boldsymbol{x}_i\big))$
(5) Residual Network + Post-Norm	$\boldsymbol{x}_{i+1} = \text{Norm}\big(\boldsymbol{x}_i + F(\boldsymbol{x}_i)\big)$
(6) **ReZero**	$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i F(\boldsymbol{x}_i)$

Table 2: Comparison of various 12 layer Transformers normalization variants against ReZero and the training iterations required to reach 1.2 BPB on `enwiki8` validation set.

Model	Iterations	Speedup
Post-Norm [21]	Diverged	-
+ Warm-up	13,690	1×
Pre-Norm	17,765	0.77×
GPT2-Norm [4]	21,187	0.65×
ReZero $\alpha = 1$	14,506	0.94×
ReZero $\alpha = 0$	**8,800**	**1.56**×

$$GPT2 - Norm : x_{i+1} = x_i + Norm(F(x_i))$$

Evaluation

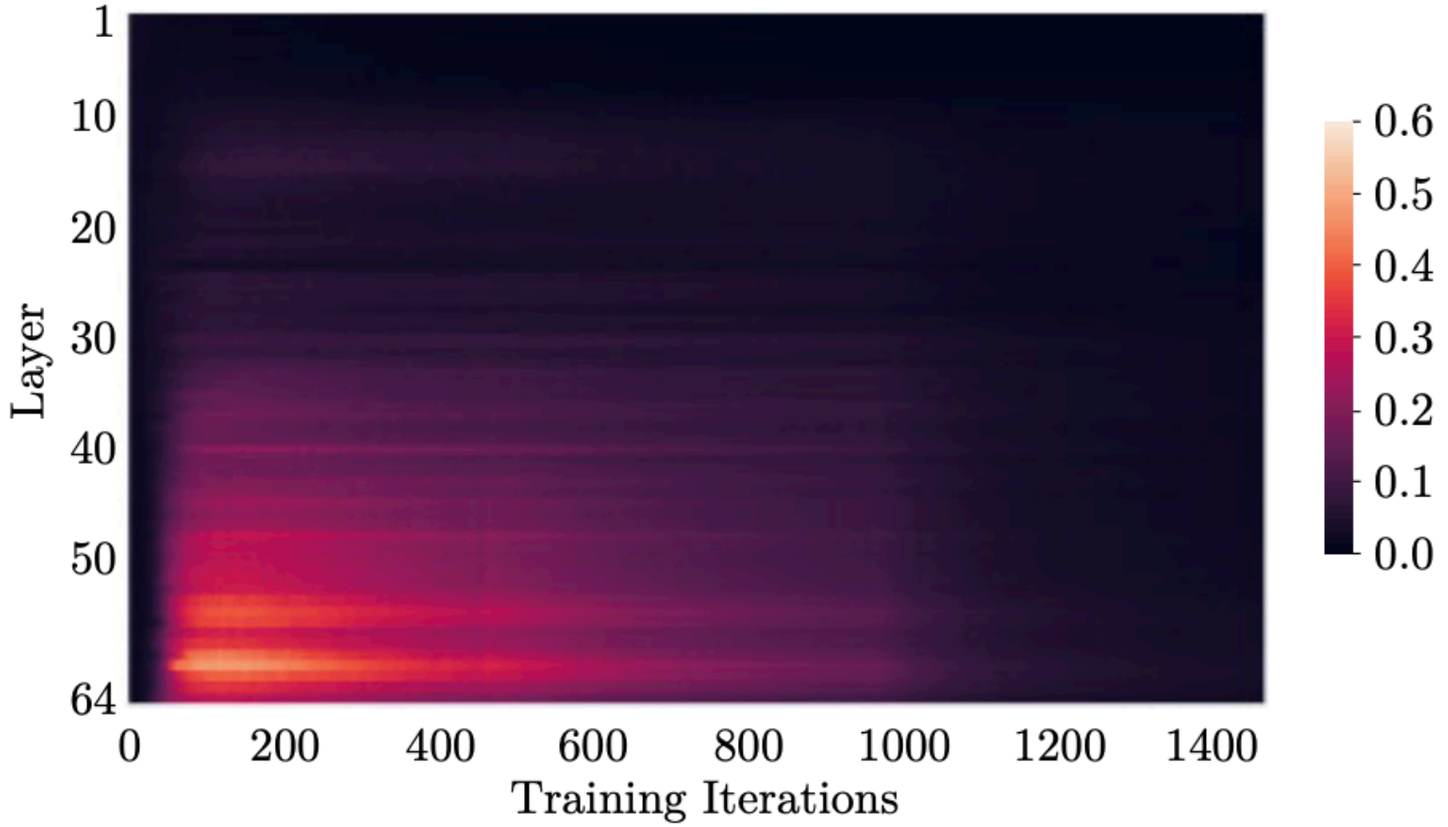


Figure 6: Heat map for residual weight $|\alpha_i|$ evolution during training for 64L ReZero Transformer.

$$x_{i+1} = x_i + \alpha_i F(x_i)$$