Topics in ML, Lecture 12
ResNets (I)

Lénaïc Chizat*

December 8, 2025

Contents

1 Residual Neural Networks: definition

1.1 Forward pass

A generic Residual Neural Network (ResNets) of embedding dimension d, hidden width M and depth L is defined as follows. For an input $x \in \mathbb{R}^d$,

$$h_\theta^0(x) = x, \qquad h_\theta^\ell(x) = h_\theta^{\ell-1}(x) + \alpha \sum_{j=1}^{M} \phi(h_\theta^{\ell-1}(x), z^{j,\ell}), \qquad \ell \in [1:L] \tag{1}$$

where $\theta = (z^{j,\ell})_{j,\ell} \in (\mathbb{R}^p)^{M\times L}$ are the parameters, $\phi : \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R}^d$ is a nonlinear map and $\alpha = \alpha(M, L) > 0$ a scalar multiplier.

In the particular case of two-layer perceptron (2LP) blocks, we have $\phi(x, z) = v\rho(u^\top x)$ where $\rho : \mathbb{R} \to \mathbb{R}$ and $z = (u, v) \in \mathbb{R}^d \times \mathbb{R}^d$ (hence $p = 2d$).

Transformer The Transformer architecture, which is state-of-the-art in many AI applications (text, image, sound processing, etc) is of the form (1) but with the following modifications:

*EPFL lenaic.chizat@epfl.ch

- the nonlinear maps ϕ alternate between standard 2LP blocks (for ℓ odd) and attention blocks (for ℓ even). We will not define attention blocks in this course; they are nonlinear maps that allow interactions between several inputs processed in parallel (such as several words/tokens in a sentence or patches in an image).
- there are two additional linear maps: the input is replaced by $h_\theta^0(x) = W_E x$ with an embedding matrix W_E and the output is $W_R h_\theta^L(x)$ with a readout (or unembedding) matrix W_R. Both W_E an W_R are "trainable" weight matrices. Therefore, the embedding dimension d is an architecture parameter which can also be scaled-up (like depth L and hidden width M). However, in this course, we'll keep d fixed.

Our analyses of ResNets therefore apply to Transformers as well, up to minor adjustments.

1.2 Gradients and backward pass

Consider a loss function loss and a training set $(x_i, y_i)_{i=1}^n \in (\mathbb{R}^d \times \mathbb{R}^d)$. Define the loss

$$\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^n \mathcal{L}_i(\theta), \qquad\qquad \mathcal{L}_i = \mathrm{loss}(y_i, h_\theta^L(x_i)).$$

We now derive the formula for the gradients of $\mathcal{L}$. By the chain rule, it holds for $(j,\ell) \in [M]\times[L]$

$$\frac{\partial \mathcal{L}_i}{\partial z^{j,\ell}} = \frac{\partial \mathcal{L}_i}{\partial h^L}\frac{\partial h^L}{\partial h^\ell}\frac{\partial h^\ell}{\partial z^{j,\ell}} \qquad \Longrightarrow \qquad \nabla_{z^{j,\ell}}\mathcal{L}_i = \left(\frac{\partial h^\ell}{\partial z^{j,\ell}}\right)^\top \left(\frac{\partial h^L}{\partial h^\ell}\right)^\top \nabla \mathrm{loss}(y_i, h^L(x_i)).$$

where the physics notation $\frac{\partial \cdot}{\partial \cdot}$ represent Jacobian matrices evaluated at the appropriate input value. Moreover

$$\frac{\partial h^\ell}{\partial h^{\ell-1}} = \mathrm{Id} + \alpha\sum_{j=1}^M D_1\phi(\cdot, z^{j,\ell}) \qquad \Longrightarrow \qquad \left(\frac{\partial h^\ell}{\partial h^{\ell-1}}\right)^\top = \mathrm{Id} + \alpha\sum_{j=1}^M D_1\phi(\cdot, z^{j,\ell})^\top.$$

Combining these equations leads to a nice recursive formula for the gradients, which is a particular case of the *backpropagation algorithm.*

For an input $x \in \mathbb{R}^d$ and a loss gradient $w = \nabla\mathcal{L}_i \in \mathbb{R}^d(\theta)$, define the *backward pass*

$$b_\theta^\ell(x, w) \;:=\; \Big(\frac{\partial h^L}{\partial h^\ell}(x)\Big)^\top w \quad \in \mathbb{R}^d$$

which is linear in w. By the chain rules above, we have $\forall j \in [1:M]$, $\forall \ell \in [1:L]$,

$$\nabla_{z^{j,\ell}}\mathcal{L}_i(\theta) = \alpha\, D_2\phi\big(h_\theta^{\ell-1}(x_i),\, z^{j,\ell}\big)^\top b_\theta^\ell\big(x_i,\, \nabla\mathrm{loss}(y_i, h_\theta^L(x_i))\big), \tag{2}$$

and $(b_\theta^\ell)_{\ell\in[1:L]}$ can be obtained from the *backward pass* recursion

$$b_\theta^L(x,w) = w, \qquad b_\theta^{\ell-1}(x,w) = b_\theta^\ell(x,w) + \alpha\sum_{j=1}^M D_1\phi\big(h_\theta^{\ell-1}(x),\, z^{j,\ell}\big)^\top b_\theta^\ell(x,w). \tag{3}$$

In these expressions, $D_1\phi$ and $D_2\phi$ denote the Jacobians of ϕ in its first and second argument, respectively.

Remark 1.1 (Automatic differentiation)**.** *In practice, one does not need to derive the expression of the gradients in deep learning by hand : the automatic differentiation algorithms can do this reliably and with the same computational complexity as evaluating the expressions we've just derived. The reason we have derived these expressions here is because they are needed in our theoretical analysis.*

1.3 Gradient descent equations

Consider an initial probability distribution $\mu_0 \in \mathcal{P}(\mathbb{R}^p)$ (with sufficiently light tails) and a learning-rate $\eta = \eta(L, M) > 0$. The gradient descent (GD) dynamics $(\theta_k)_{k\geq 0} = (Z_k^{j,\ell})_{j,\ell,k}$ is defined by

$$Z_0^{j,\ell} \overset{\text{iid}}{\sim} \mu_0, \qquad Z_{k+1}^{j,\ell} = Z_k^{j,\ell} - \eta \nabla_{z^{j,\ell}} \mathcal{L}(\theta_k), \qquad \forall j \in [1:M],\ \forall \ell \in [1:L],\ \forall k \in \mathbb{N}.$$
$$= \hat{Z}_k^{j,\ell} - \frac{\eta}{n} \sum_{i=1}^{n} \nabla_{z^{j,\ell}} \mathcal{L}_i(\theta_k)$$

We have switched to capital letters to emphasize that we are dealing with random variables. In what follows, we will often consider $n = 1$ to shorten expressions.

2 Signal Propagation in a Random ResNet

In our 2LP analysis of last week, we saw that the scale of the first output (at random initialization) was given by a CLT. Let us now discuss the scale of the first output for ResNets. This is more subtle because the forward pass involves a recursion.

2.1 Forward pass: general case

Let us fix an input $x \in \mathbb{R}^d$. Dropping all subscripts, the first recursion is given by

$$h^0 = x, \qquad h^\ell = h^{\ell-1} + \alpha \sum_{j=1}^{M} \phi(h^{\ell-1}, Z_0^{j,\ell}), \qquad \ell \in [1:L].$$

We assume that the $(Z_0^{j,\ell})$ are iid samples from Z_0, a $\mathbb{R}^p$-valued random variable such that there exists a constant $c > 0$ such that $\forall h \in \mathbb{R}^d$

$$\mathbf{E}\big[\phi(h, Z_0) \mid h\big] = 0, \qquad \mathbf{E}\big[\|\phi(h, Z_0)\|_2^2 \mid h\big] = c\|h\|_2^2. \tag{4}$$

The second assumption is related to positive 1-homogeneity of ϕ wrt to x and is quite restrictive. However, if it holds with an inequality instead $\leq$ (which is very common) then the result that follows holds with inequalities.

Proposition 2.1. *For every $\ell \geq 0$, it holds $\mathbf{E}[h^\ell] = x$ and, letting $\Delta_\ell^2 = \mathbf{E}\big[\|h^\ell - x\|_2^2\big]$,*

$$\Delta_{\ell+1}^2 = (1 + cM\alpha^2)\Delta_\ell^2 + c\alpha^2 M\|x\|_2^2.$$

In particular,

$$\Delta_\ell^2 = \big((1 + c\alpha^2 M)^\ell - 1\big)\|x\|_2^2.$$

Proof. The fact that $\mathbf{E}[h^\ell] = x$ is immediate from the expression

$$h^\ell = x + \alpha \sum_{k=1}^{\ell} \sum_{j=1}^{M} \phi(h^{k-1}, Z_0^{j,k}).$$

Next,

$$\|h^{\ell+1} - x\|_2^2 = \|h^\ell - x\|_2^2 + 2\alpha \sum_{j=1}^{M} (h^\ell - x)^\top \phi(h^\ell, Z^{j,\ell+1}) + \alpha^2 \Big\| \sum_{j=1}^{M} \phi(h^\ell, Z^{j,\ell+1}) \Big\|_2^2.$$

Set $\mathcal{F}_\ell := \sigma(Z^1, \dots, Z^\ell)$. Taking conditional expectation with respect to $\mathcal{F}_\ell$, we get

$$\begin{aligned}\mathbf{E}[\|h^{\ell+1} - x\|_2^2 \mid \mathcal{F}_\ell] &= \|h^\ell - x\|_2^2 + 2\alpha \sum_{j=1}^M \mathbf{E}[(h^\ell - x)^\top \phi(h^\ell, Z^{j,\ell+1}) \mid \mathcal{F}_\ell] + \alpha^2 \mathbf{E}\Big[\Big\| \sum_{j=1}^M \phi(h^\ell, Z^{j,\ell+1})\Big\|_2^2 \mid \mathcal{F}_\ell\Big] \\ &= \|h^\ell - x\|_2^2 + \alpha^2 \sum_{j=1}^M \mathbf{E}\Big[\|\phi(h^\ell, Z^{j,\ell+1})\|_2^2 \mid \mathcal{F}_\ell\Big] \\ &= \|h^\ell - x\|_2^2 + cM\alpha^2 \|h^\ell\|^2\end{aligned}$$

Taking total expectations and using $\mathbf{E}\|h^\ell\|_2^2 = \mathbf{E}\|h^\ell - x\|_2^2 + \mathbf{E}\|x\|_2^2$ gives the recursion. Integrating the recursion, we get

$$\Delta_\ell^2 = c\alpha^2 M \sum_{k=0}^{\ell-1} (1 + c\alpha^2 M)^k \|x\|_2^2 = \big((1 + c\alpha^2 M)^\ell - 1\big)\|x\|_2^2. \qquad \square$$

When L diverges, maintaining bounded variance requires $cM\alpha^2 \to 0$, in which case the multiplier is $(1 + cM\alpha^2)^L - 1 = \exp(cML\alpha^2 + o(1)) - 1$. Therefore the variance of the first output is nontrivial iff $\alpha = \Theta(1/\sqrt{ML})$. If $\alpha = o(1/\sqrt{ML})$ then $\Delta_L^2 \sim cML\alpha^2 = o(1)$ so the first forward pass concentrates around the identity map.

Remark 2.2 (Concentration in high probability)**.** *The process $S^\ell = \|h^\ell - x\|_2^2$ is a nonnegative submartingale with respect to the filtration $\mathcal{F}_\ell$ (indeed, by Jensens' inequality, $\mathbf{E}[S^{\ell+1} \mid \mathcal{F}_\ell] = \mathbf{E}[\|h^{\ell+1} - x\|_2^2 \mid \mathcal{F}] \geq \|\mathbf{E}[h^{\ell+1} - x \mid \mathcal{F}_\ell]\|_2^2 = \|h^\ell - x\|_2^2 = S^\ell$). Therefore, by Doob's maximal inequality, we also have uniform high-probability bound of the form:*

$$\mathbf{P}\left(\sup_{\ell \leq L} S^\ell \leq a\right) \leq \frac{\mathbf{E}[S^L]}{a} \implies \mathbf{P}\left(\sup_{\ell \leq L} \|h^\ell - x\|_2 \geq \beta \|x\|_2\right) \leq \frac{(1 + c\alpha^2 M)^L - 1}{\beta^2}.$$

Of course, under stronger tail assumptions on $\phi(\cdot, Z)$ (such as subgaussian) one can obtain improved dependency in β (see e.g. Azuma Hoeffing's inequality).

2.2 The case of ResNets with ReLU 2LP blocks

To illustrate the result on a concrete case, let us compute the value of c for 2LP blocks. Consider

$$\phi(x, z) = v \max\{0, u^\top x\}, \qquad z = (u, v) \in \mathbb{R}^d \times \mathbb{R}^d$$

and take as initial distribution

$$U \overset{iid}{\sim} \mathcal{N}(0, \sigma_u^2 I_d), \qquad V \overset{iid}{\sim} \mathcal{N}(0, \sigma_v^2 I_d).$$

Then clearly, for any $x \in \mathbb{R}^d$, it holds $\mathbf{E}[V \max\{0, U^\top x\}] = 0$. Let us compute the variance. Let $a = \max\{0, U^\top x\}$. Clearly, $U^\top x = \sum_{i=1}^d U_i x_i \sim \mathcal{N}(0, \sigma_u^2 \|x\|_2^2)$. By a direct integral computation, one can check that

$$\mathbf{E}[a^2] = \mathbf{E}[\max\{0, U^\top x\}^2] = \frac{1}{2}\mathbf{E}[(U^\top x)^2] = \frac{1}{2}\sigma_u^2 \|x\|_2^2.$$

Moreover

$$\mathbf{E}[\|V \max\{0, U^\top x\}\|_2^2 \mid a] = a^2 \mathbf{E}\|V\|_2^2 = d\sigma_v^2 a^2.$$

Taking the total expectation we obtain

$$\mathbf{E}[\|V \max\{0, U^\top x\}\|_2^2] = \frac{d}{2}\sigma_v^2 \sigma_u^2 \|x\|_2^2$$

Therefore in this case, the value of c from Proposition 2.1 is $c = \frac{d}{2}\sigma_v^2 \sigma_u^2$.

Remark 2.3. *It is desirable at initialization that a has nontrivial variance, so that initial features are "diverse" which requires to take* $\sigma_u = 1/\sqrt{d}$*, in which case* $c = \sigma_v^2/2$.

3 First-order equivalence with shallow NNs and phase diagram

Let us consider the large depth limit $L \to \infty$ of a ResNet such that $ML\alpha^2 = o(1)$ and a single training sample (x, y). In this case, we have seen that with high probability the first forward pass concentrates around x, i.e. $\|h^\ell - x\|^2 = o(1)$. For simplicity, we will not carry the error terms and assume the identity $h^\ell = x$.

3.1 Backward pass

Under this simplification, the equations for the first backward pass are:

$$b^L = \nabla \text{loss}(y, x), \qquad\qquad b^{\ell-1} = b^\ell + \alpha \sum_{j=1}^{M} D_1\phi(x, Z^{j,\ell})^\top b^\ell$$

Clearly, this iteration has the same form as the forward pass, provided one identifies $D_1\phi(x, Z^{j,\ell})^\top b^\ell$ with $\phi(b^\ell, z)$. Therefore, under the same assumptions, we have with high probability $\sup_\ell \|b^\ell - \nabla \text{loss}(y, x)\|^2 = o(1)$, i.e. the backward pass is asymptotically the identity map initialized with $\nabla \text{loss}(y, x)$.

Case of 2LP In the case of 2LP (same notations as above), we have

$$D_1\phi(x, (u, v))^\top b = \rho'(x^\top u)(b^\top v)u$$

and it is easy to check that the assumptions of the previous sections (replacing $=$ by $\leq$ in the variance identity) hold for some $c > 0$, for instance if ρ' is bounded.

3.2 Gradient updates

Under these assumptions, at $k = 0$, (2) becomes for a single sample (x, y):

$$\nabla_{z^{j,\ell}} \mathcal{L}_i(\theta_0) = \alpha\, D_2\phi\big(x_i,\, z^{j,\ell}\big)^\top \nabla \text{loss}(y_i, x_i). \tag{5}$$

This is exactly the same update than for a shallow neural network with effective scale α, with the minor difference that the first output is x rather than 0 (or a random variable if $\alpha = \Theta(1/\sqrt{M})$).

3.3 Tangent kernel

One deduces as well that, in this range of scaling, the tangent model at initialization is the same as for a shallow NN of width $M \times L$ (provided one corrects the first output), namely

$$K(x, x') = \alpha^2 \sum_{j=1}^{M} \sum_{\ell=1}^{L} D_2\phi\big(x,\, z^{j,\ell}\big) D_2\phi\big(x',\, z^{j,\ell}\big)^\top$$

In other words, this model behaves at initialization and in first order in training time, like a shallow NN with $M \times L$ units. By comparing the relative speed of evolution of the parameters vs the predictor, we deduce the phase diagram shown on Figure 1. Therein, we have in particular the following behaviors:

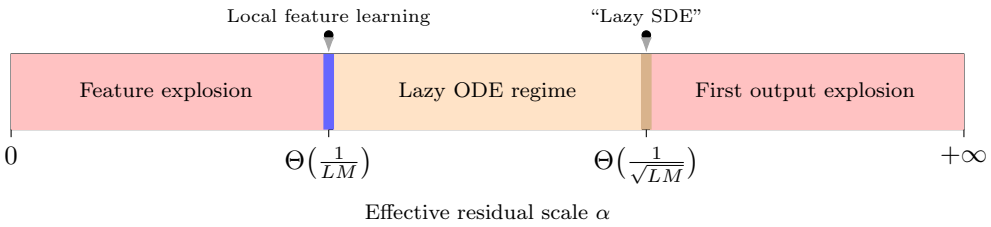

Figure 1

- *Local feature learning*: the $Z^{j,\ell}$ evolve at the same time-scale as the predictor. In particular, **the contribution of the local weight** $Z^{j,\ell}$ to the evolution of the feature $\phi(\cdot, Z^{j,\ell})$ is **nonvanishing**;
- *Lazy ODE regime*: the $Z^{j,\ell}$ evolve at a slower time-scale as the predictor. Hence, **the contribution of the local weight** $Z^{j,\ell}$ to the evolution of the feature $\phi(\cdot, Z^{j,\ell})$ is **vanishing**;

Note that in the lazy ODE regime, the features still evolve significantly. Let's look for instance at $h^{L/2}$: it evolves at the same time-scale as h^L (compare their respective tangent kernels). So all features after $h^{L/2}$ evolve because of the evolution of their first input. In other words, the evolution of the features at layer ℓ are due to the integral effect of all the weights update before $\ell' < \ell$ but not due to the local weight updates. It has been observed empirically that the lazy ODE regime is suboptimal [Dey et al., 2025].

References

Nolan Dey, Bin Claire Zhang, Lorenzo Noci, Mufan Li, Blake Bordelon, Shane Bergsma, Cengiz Pehlevan, Boris Hanin, and Joel Hestness. Don't be lazy: Completep enables compute-efficient deep transformers. *arXiv preprint arXiv:2505.01618*, 2025.