What has my network learned? The rainbow model of deep networks

Florentin Guth

Brice Ménard

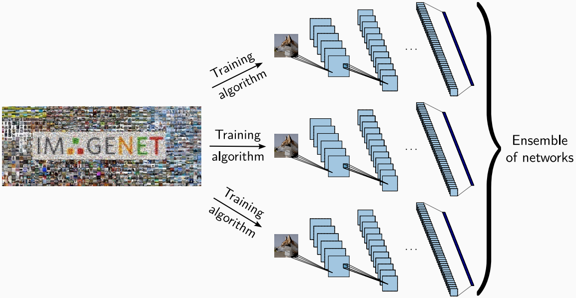
Gaspar Rochette

Stéphane Mallat

ENS | PSL NYU FLATIRON INSTITUTE

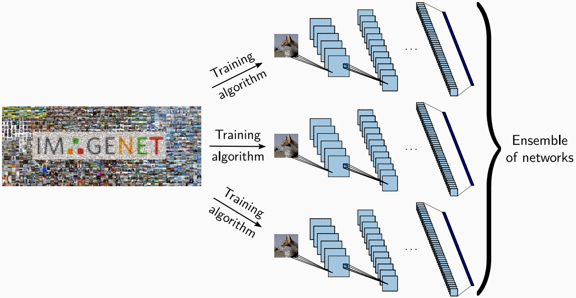
What has the network learned?

Every time we train a network, we get a different set of weights because of the random initialization



What has the network learned?

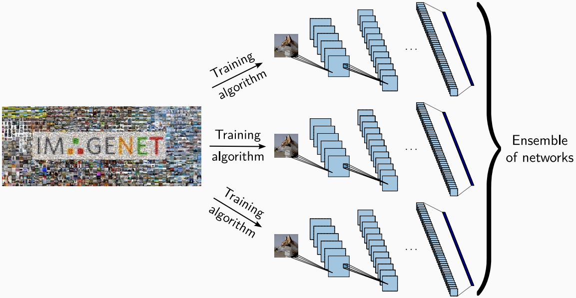
Every time we train a network, we get a different set of weights because of the random initialization



- What is random and what is stable across different training runs?

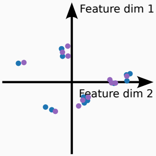
What has the network learned?

Every time we train a network, we get a different set of weights because of the random initialization

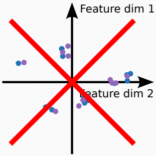


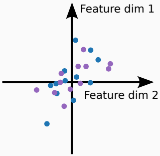
- What is random and what is stable across different training runs?
- **What is the distribution of trained network weights?**

Comparing first layer weights

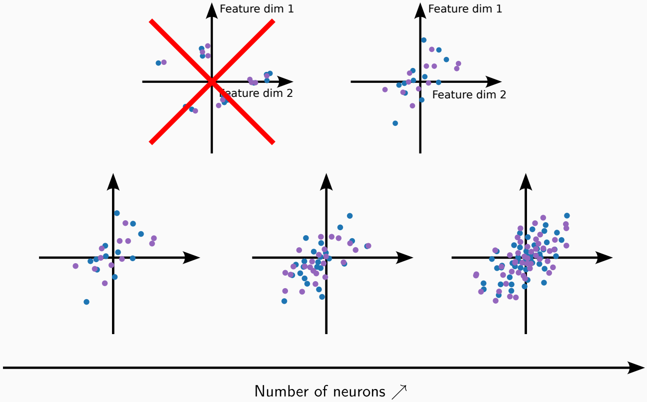


Comparing first layer weights

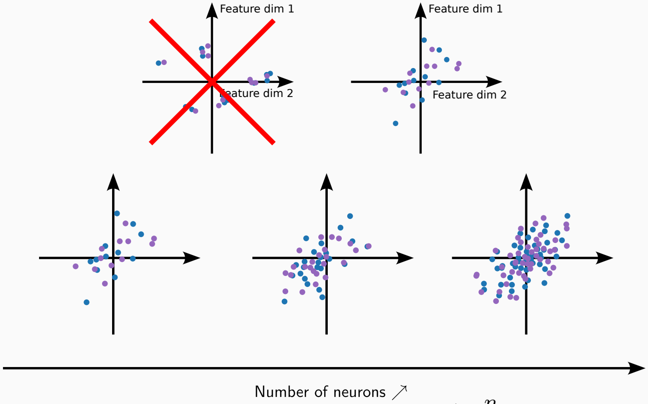




Comparing first layer weights



Comparing first layer weights

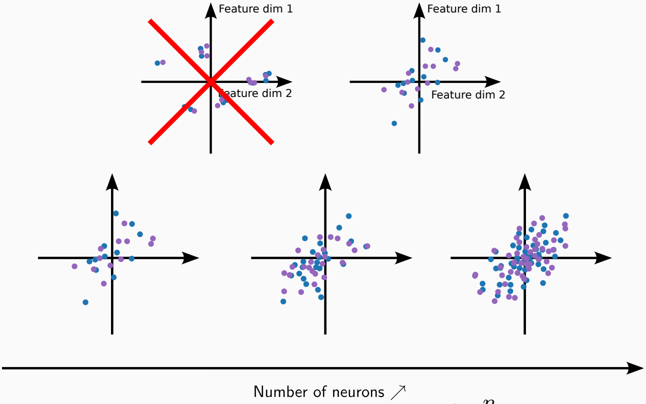


Mean-field (infinite-width) limit

$$\frac{1}{n}\sum_{i=1}^{n}\delta_{w_i} \xrightarrow[n\to\infty]{} \pi$$

(Chizat and Bach, 2018; Mei et al., 2018; Rotskoff and Vanden-Eijnden, 2018; Sirignano and Spiliopoulos, 2020)

Comparing first layer weights



Mean-field (infinite-width) limit

$$\frac{1}{n}\sum_{i=1}^{n}\delta_{w_i} \xrightarrow[n\to\infty]{} \pi$$

(Chizat and Bach, 2018; Mei et al., 2018; Rotskoff and Vanden-Eijnden, 2018; Sirignano and Spiliopoulos, 2020)

Neurons are random samples from some fixed distribution
(random features)

Comparing first layer activations

Random feature activations: $\phi(x) = n^{-1/2}(\sigma(\langle w_i, x\rangle))_{i\leq n}$ with $w_i \sim \pi$ i.i.d.

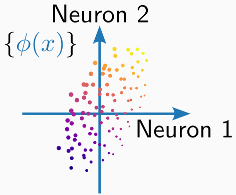
$$\langle \phi(x), \phi(x') \rangle = \frac{1}{n}\sum_{i=1}^{n} \sigma(\langle w_i, x\rangle)\,\sigma(\langle w_i, x'\rangle) \to \mathbb{E}_{w\sim\pi}\Big[\sigma(\langle w, x\rangle)\,\sigma(\langle w, x'\rangle)\Big]$$

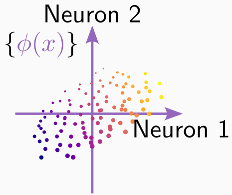
(Rahimi and Recht, 2007; Haxby et al., 2011; Kornblith et al., 2019)

Comparing first layer activations

Random feature activations: $\phi(x) = n^{-1/2}(\sigma(\langle w_i, x\rangle))_{i\leq n}$ with $w_i \sim \pi$ i.i.d.

$$\langle \phi(x), \phi(x')\rangle = \frac{1}{n}\sum_{i=1}^{n} \sigma(\langle w_i, x\rangle)\,\sigma(\langle w_i, x'\rangle) \to \mathbb{E}_{w\sim\pi}\Big[\sigma(\langle w, x\rangle)\,\sigma(\langle w, x'\rangle)\Big]$$



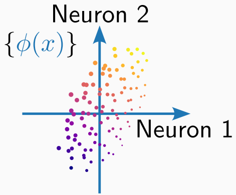


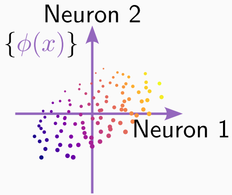
(Rahimi and Recht, 2007; Haxby et al., 2011; Kornblith et al., 2019)

Comparing first layer activations

Random feature activations: $\phi(x) = n^{-1/2}(\sigma(\langle w_i, x\rangle))_{i\leq n}$ with $w_i \sim \pi$ i.i.d.

$$\langle \phi(x), \phi(x')\rangle = \frac{1}{n}\sum_{i=1}^{n} \sigma(\langle w_i, x\rangle)\,\sigma(\langle w_i, x'\rangle) \to \mathbb{E}_{w\sim\pi}\Big[\sigma(\langle w, x\rangle)\,\sigma(\langle w, x'\rangle)\Big]$$

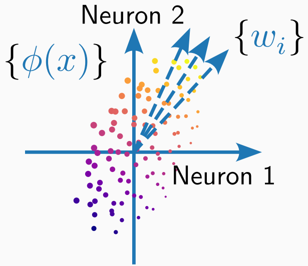


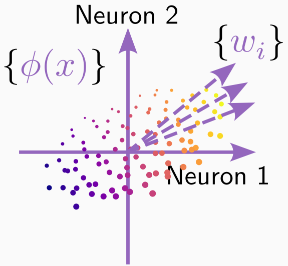


Activations are equal up to rotations: they correspond to a deterministic representation expressed in a random basis

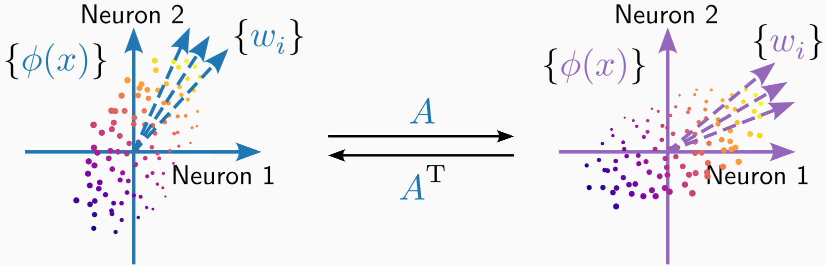
(Rahimi and Recht, 2007; Haxby et al., 2011; Kornblith et al., 2019)

Comparing second layer weights



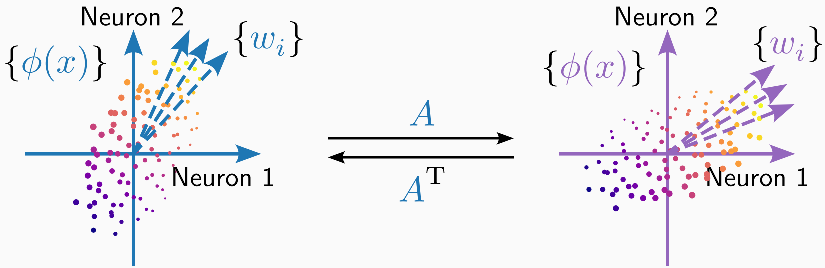


Comparing second layer weights



$$\min_{A^{\mathrm{T}}A=\mathrm{Id}} \; \mathbb{E}_x\left[\|A\,\phi(x)-\phi(x)\|^2\right] \qquad A\,\phi(x)\approx\phi(x)$$

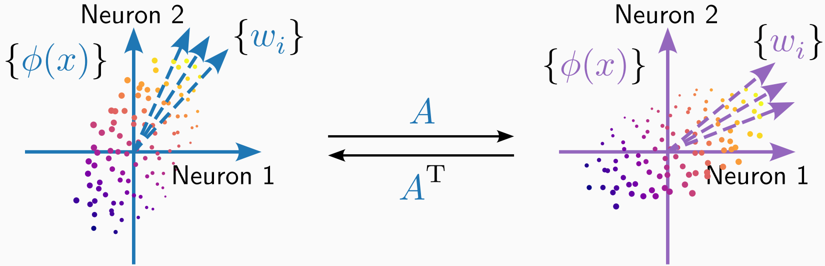
Comparing second layer weights



$$\min_{A^{\mathrm{T}}A=\mathrm{Id}} \; \mathbb{E}_x\left[\|A\,\phi(x) - \phi(x)\|^2\right] \qquad A\,\phi(x) \approx \phi(x)$$

$$\langle w_i, \phi(x)\rangle \approx \langle w_i, A^{\mathrm{T}}\phi(x)\rangle = \langle A\,w_i, \phi(x)\rangle$$

Comparing second layer weights



$$\min_{A^{\mathrm{T}}A=\mathrm{Id}} \mathbb{E}_x\left[\|A\,\phi(x) - \phi(x)\|^2\right] \qquad A\,\phi(x) \approx \phi(x)$$

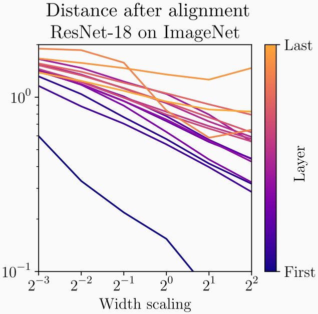
$$\langle w_i, \phi(x)\rangle \approx \langle w_i, A^{\mathrm{T}}\phi(x)\rangle = \langle A\,w_i, \phi(x)\rangle$$

Neurons are random samples from some fixed distribution

expressed in the random basis of its input activations

Comparing aligned activations and weights

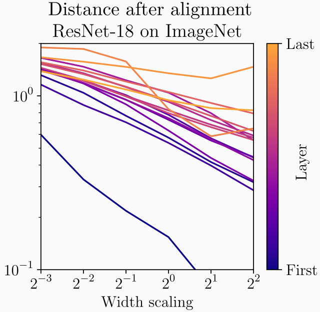
Comparison between activations

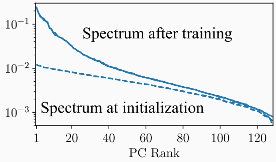


Comparing aligned activations and weights

Comparison between activations

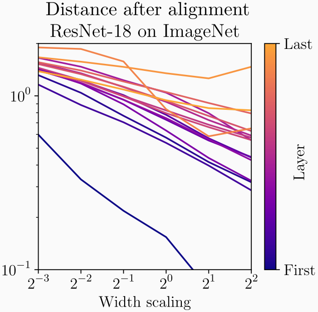
Comparison between weights



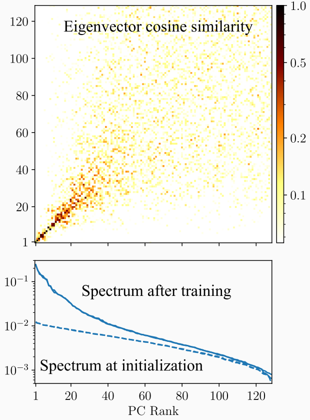


Comparing aligned activations and weights

Comparison between activations

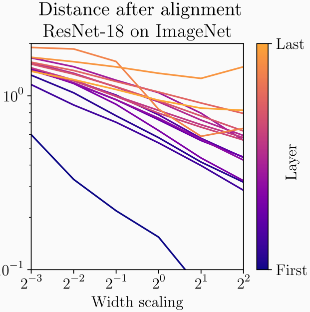


Comparison between weights

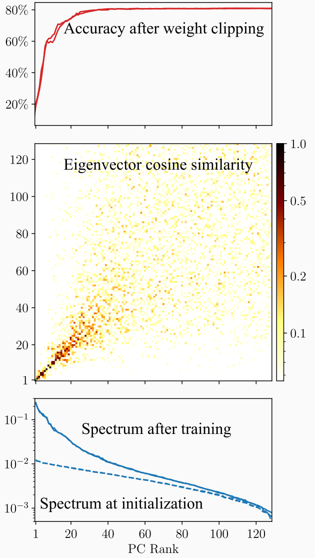


Comparing aligned activations and weights

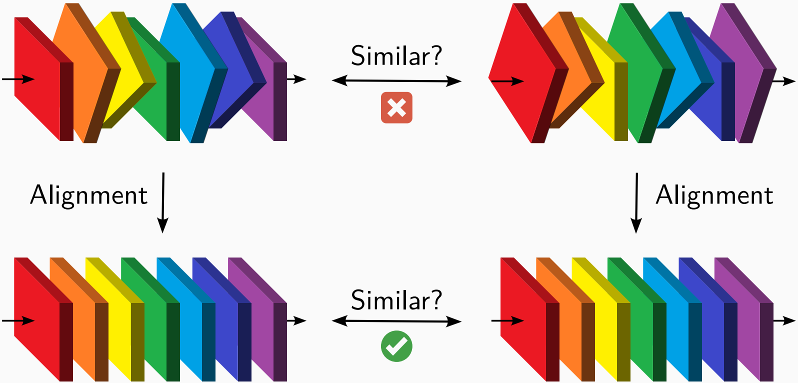
Comparison between activations



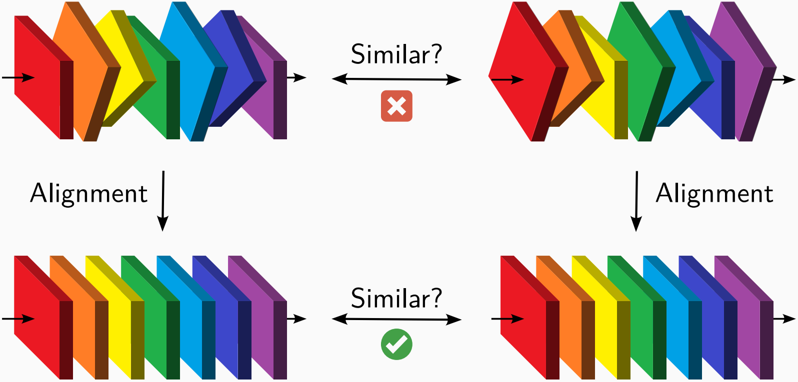
Comparison between weights



Summary



Summary

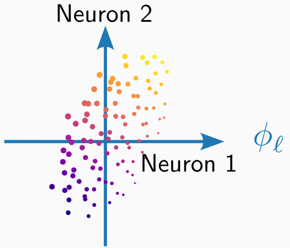


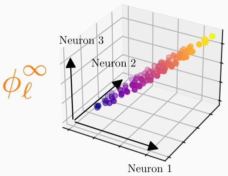
In the infinite-width limit, there is a unique deterministic network and finite-width networks can be seen as random feature discretizations of it.

The rainbow model of trained network weights

Model parameters: weight distributions π_ℓ and representations ϕ_ℓ^∞

Iterative sampling procedure: assume ϕ_ℓ has been defined

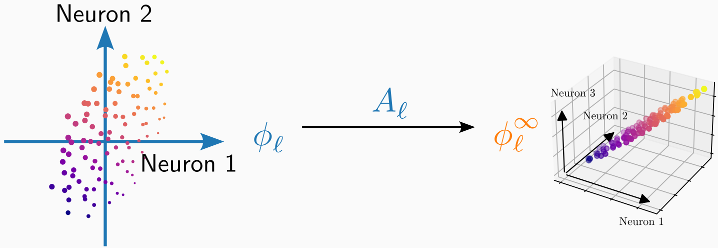




The rainbow model of trained network weights

Model parameters: weight distributions π_ℓ and representations ϕ_ℓ^∞

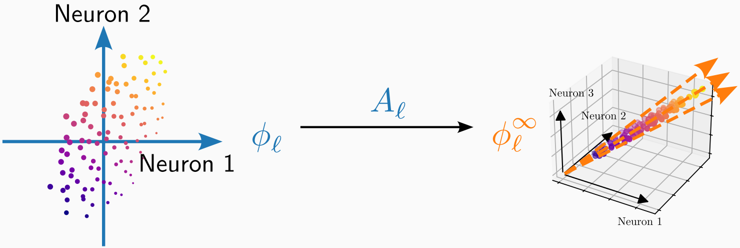
Iterative sampling procedure: assume ϕ_ℓ has been defined



The rainbow model of trained network weights

Model parameters: weight distributions π_ℓ and representations ϕ_ℓ^∞

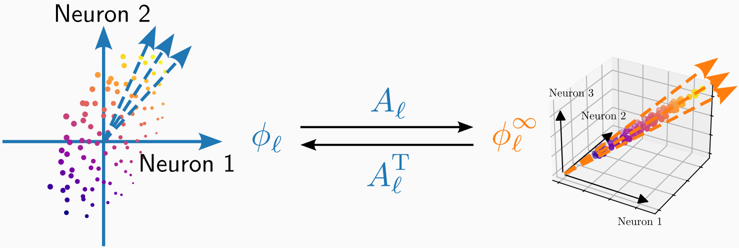
Iterative sampling procedure: assume ϕ_ℓ has been defined



The rainbow model of trained network weights

Model parameters: weight distributions π_ℓ and representations ϕ_ℓ^∞

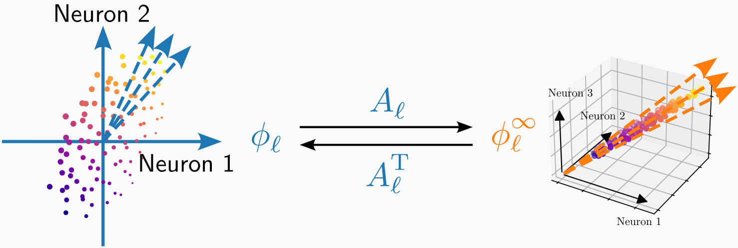
Iterative sampling procedure: assume ϕ_ℓ has been defined



The rainbow model of trained network weights

Model parameters: weight distributions π_ℓ and representations ϕ_ℓ^∞

Iterative sampling procedure: assume ϕ_ℓ has been defined



Theorem: $\forall \ell, A_\ell \, \phi_\ell \to \phi_\ell^\infty$ *polynomially in the widths.*

Assumptions: π_ℓ *has finite fourth-order moments + capacity conditions at each layer.*

A simpler model: Gaussian rainbow networks

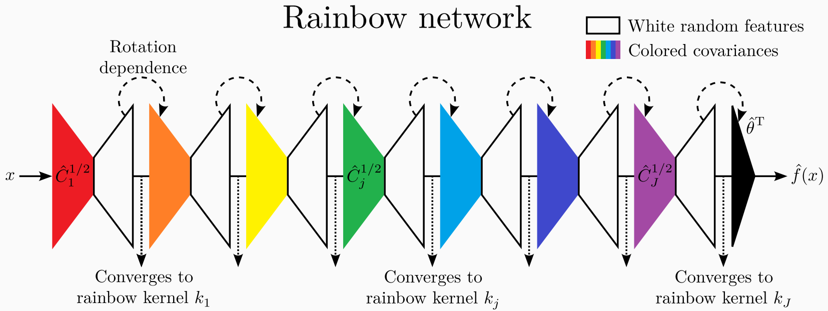
Model fully specified by weight covariances C_ℓ at each layer

- Sample $w_{1,i} \sim \mathcal{N}(0, C_1)$
- Compute A_1 by aligning ϕ_1 to ϕ_1^∞
- Sample $w_{2,i} \sim \mathcal{N}\left(0, A_1^{\mathrm{T}} C_2 A_1\right)$
- ...

A simpler model: Gaussian rainbow networks

Model fully specified by weight covariances C_ℓ at each layer

- Sample $w_{1,i} \sim \mathcal{N}(0, C_1)$
- Compute A_1 by aligning ϕ_1 to ϕ_1^∞
- Sample $w_{2,i} \sim \mathcal{N}\left(0, A_1^{\mathrm{T}} C_2 A_1\right)$
- ...

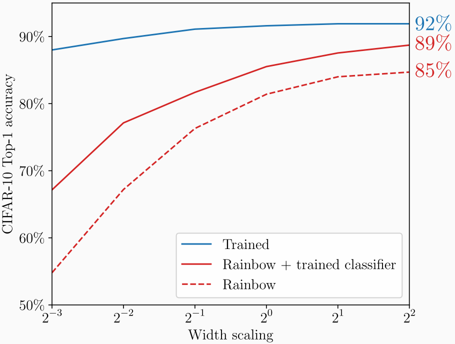


Evaluating the accuracy of rainbow networks

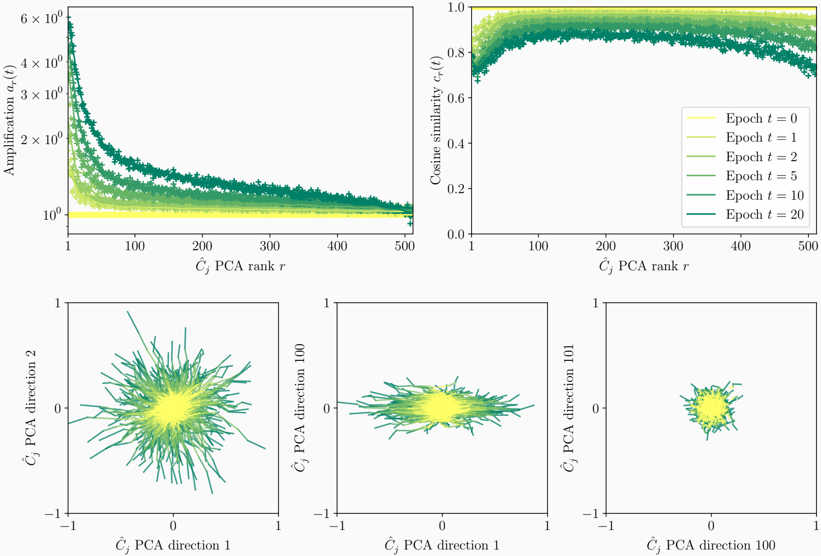
- Train a scattering network on CIFAR-10 (**fixed spatial filters** + learned channel weights) (Guth, Zarka, and Mallat, 2022)
- Extract channel covariances C_ℓ at each layer
- Generate random weights with the same **aligned** covariances
- Evaluate accuracy on test set!

Evaluating the accuracy of rainbow networks

- Train a scattering network on CIFAR-10 (**fixed spatial filters** + learned channel weights) (Guth, Zarka, and Mallat, 2022)
- Extract channel covariances C_ℓ at each layer
- Generate random weights with the same **aligned** covariances
- Evaluate accuracy on test set!



Training dynamics



Conclusion

- What has been learned? Weight distributions, sometimes just covariances
- How do they depend on the training data?
- Trained networks (and real-world datasets) as objects of scientific study
- Opens many questions in optimization (regime of validity of the model?) and generalization (properties of rainbow kernels?)

https://arxiv.org/abs/2305.18512

https://bonnerlab.github.io/ccn-tutorial/pages/analyzing_neural_networks.html