Input relevance analysis driven by spectral learning

Lorenzo Chicchi

UNIVERSITÀ DEGLI STUDI FIRENZE

Spectral Learning

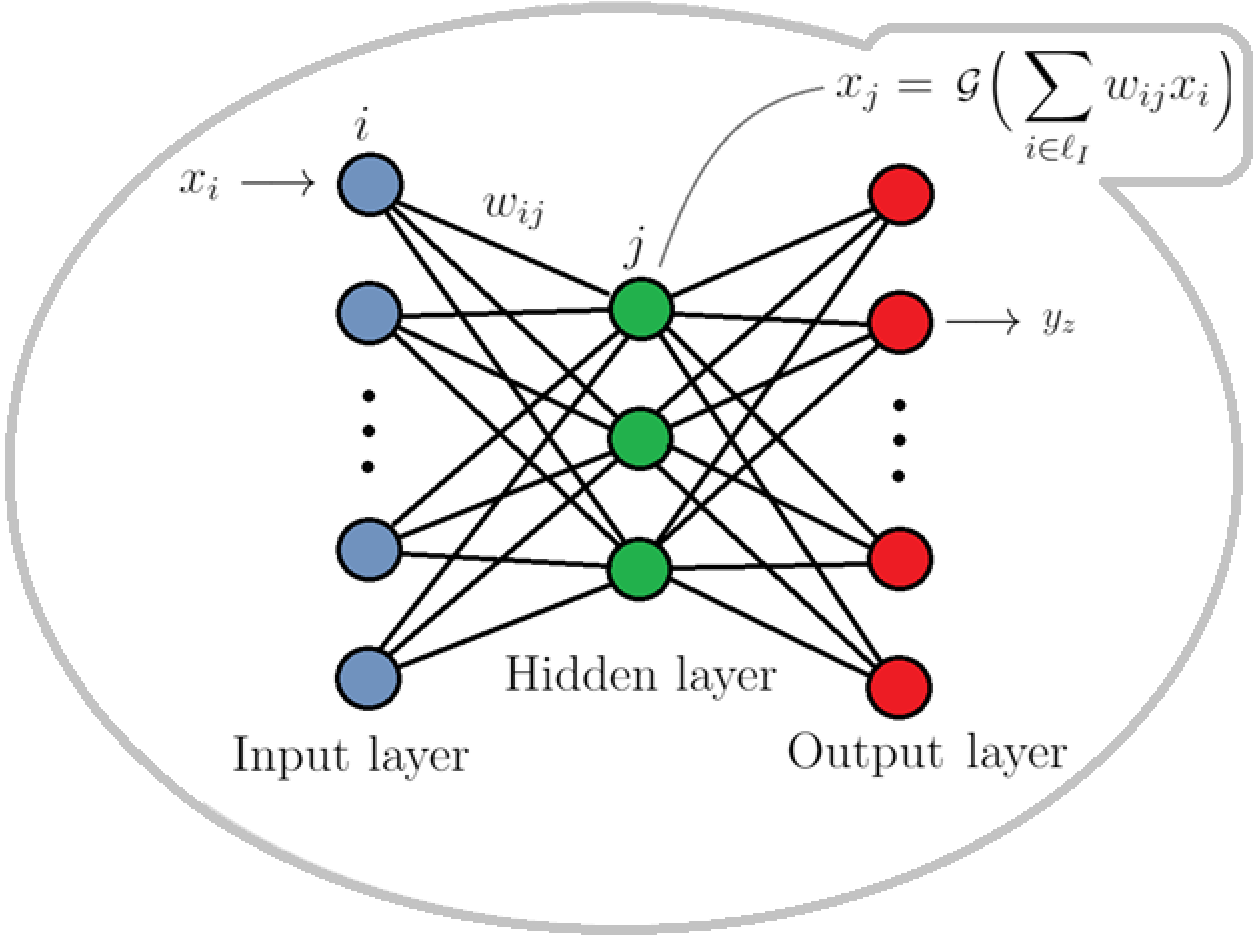
CSDC CENTRO STUDI DINAMICHE COMPLESSE

INFN Istituto Nazionale di Fisica Nucleare

Neural Networks

Neural networks (NNs) are models that can efficiently **approximate complex target functions** (Universal approximation theorem[1,2])

$$\overrightarrow{y} = N(\overrightarrow{x}) \approx \mathcal{F}(\overrightarrow{x})$$

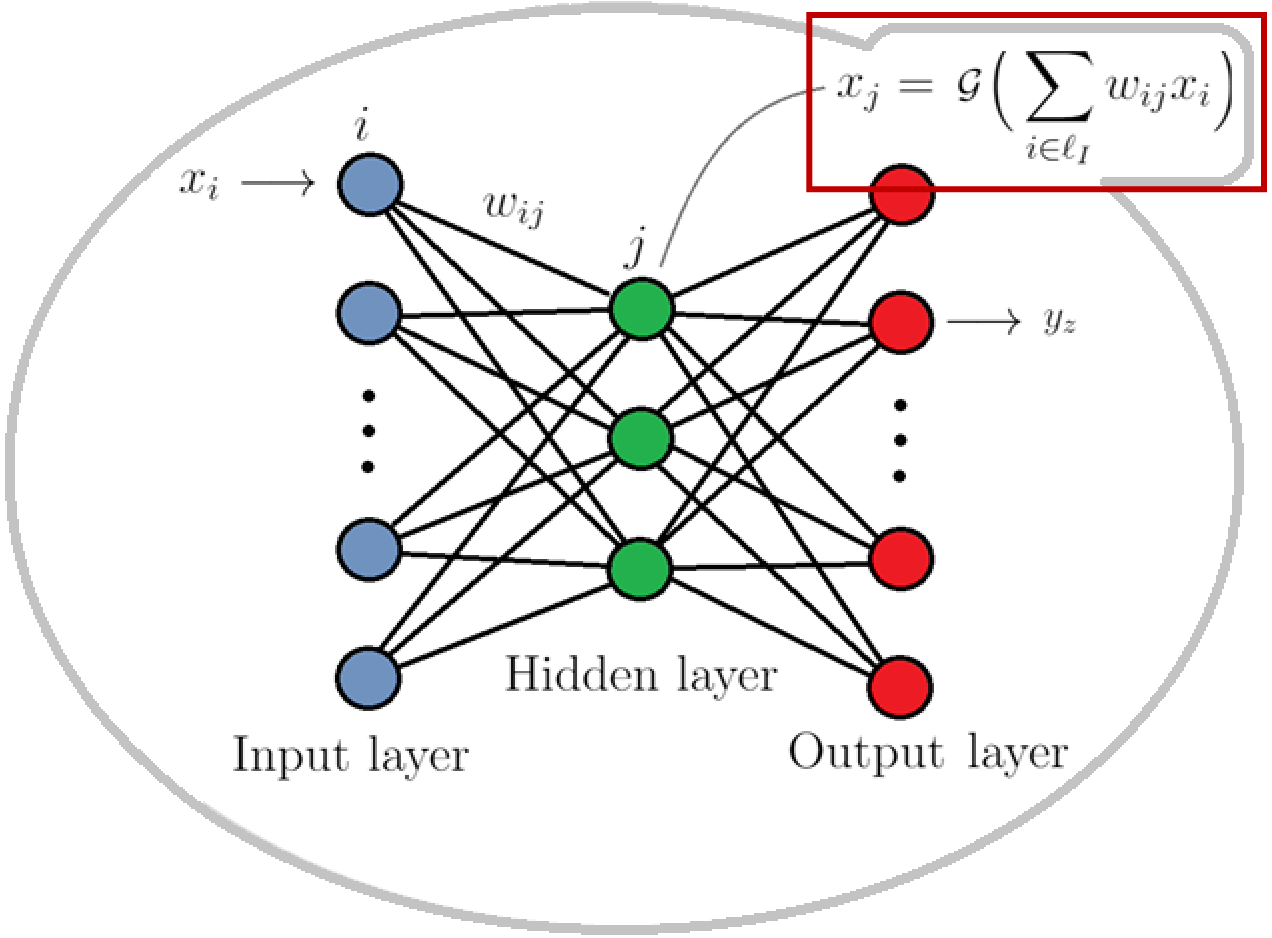


- $\boldsymbol{G_l}$ is a non-linear function (Tanh, Sigmoid,..)
- w_{ij} are the network (trainable) parameters

Neural Networks

Neural networks (NNs) are models that can efficiently **approximate complex target functions** (Universal approximation theorem[1,2])

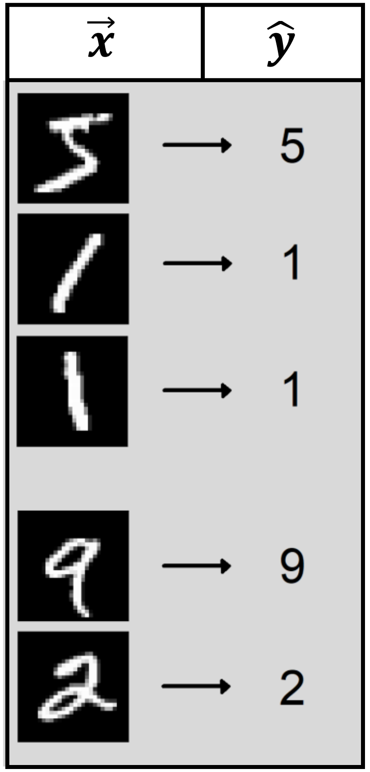
$$\overrightarrow{y} = N(\overrightarrow{x}) \approx \mathcal{F}(\overrightarrow{x})$$



- $\boldsymbol{G_l}$ is a non-linear function (Tanh, Sigmoid,..)
- w_{ij} are the network (trainable) parameters

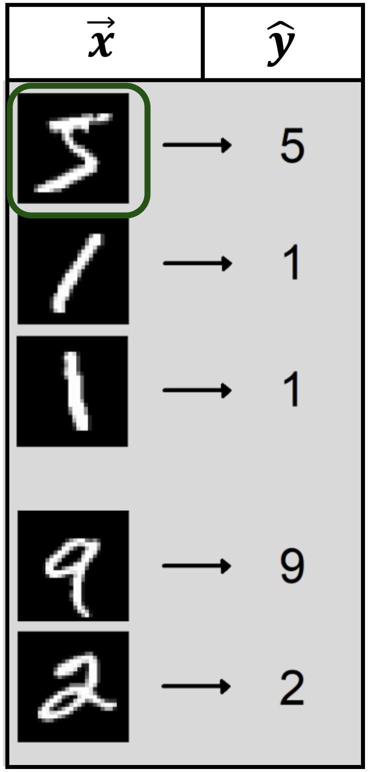
Neural Networks: how to use them

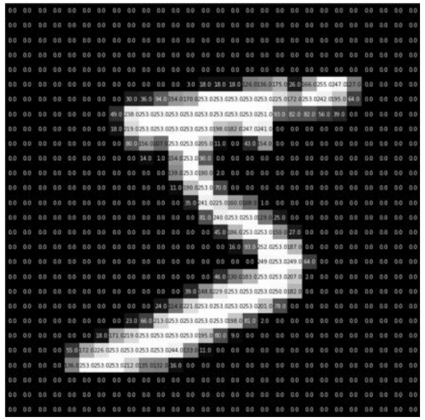
Starting from a DATA SET, that is, a set of examples input-target: $(\vec{x}, \hat{y})_i \ i = 1, .. N$



Neural Networks: how to use them

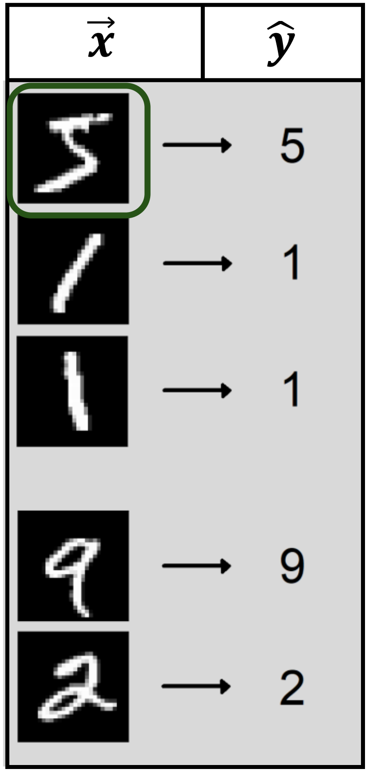
Starting from a DATA SET, that is, a set of examples input-target: $(\vec{x}, \hat{y})_i \; i = 1, .. N$

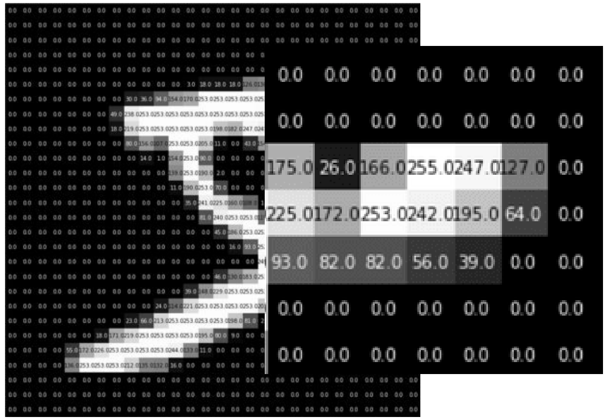




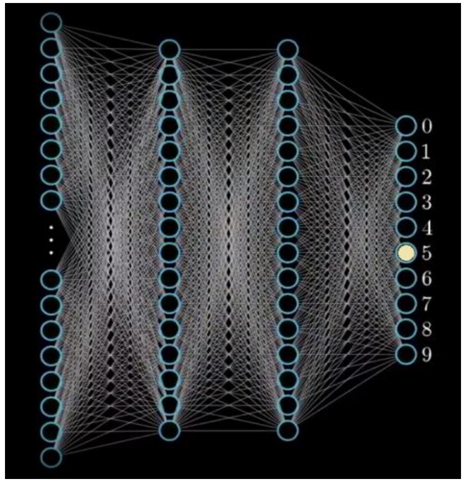
Neural Networks: how to use them

Starting from a DATA SET, that is, a set of examples input-target: $(\vec{\boldsymbol{x}}, \hat{\boldsymbol{y}})_i \ \boldsymbol{i} = \mathbf{1}, .. \boldsymbol{N}$





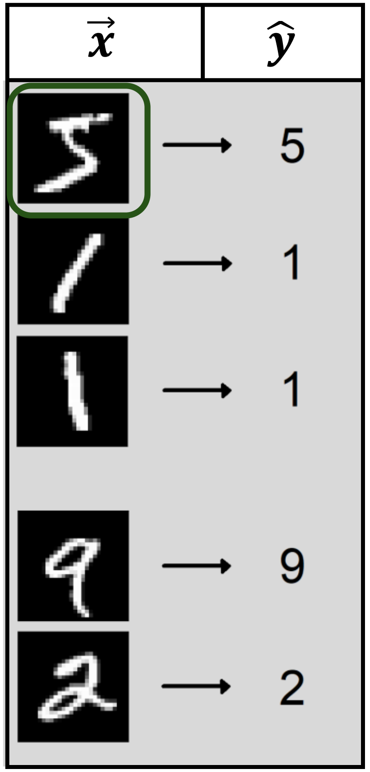
→ $\vec{x}_i$ →

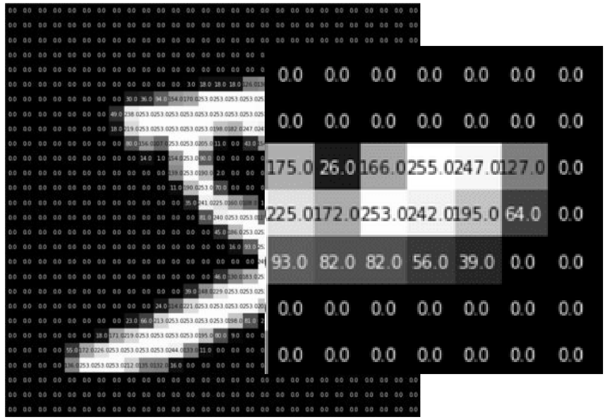


→ $N(\vec{x}_i; \vec{w})$

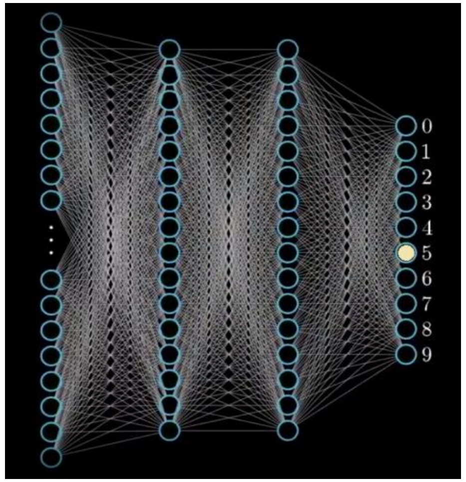
Neural Networks: how to use them

Starting from a DATA SET, that is, a set of examples input-target: $(\vec{\boldsymbol{x}}, \hat{\boldsymbol{y}})_i \ \boldsymbol{i} = \mathbf{1}, .. \boldsymbol{N}$





→ $\vec{x}_i$ →



→ $N(\vec{x}_i; \vec{w})$

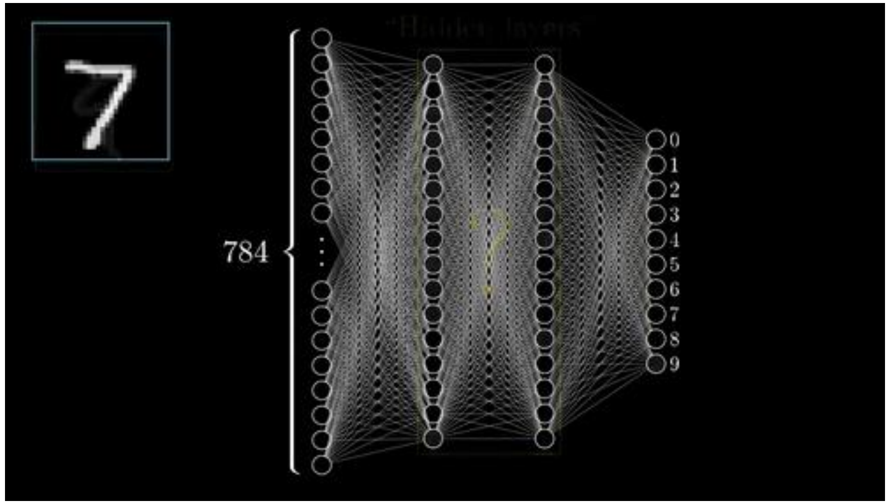
Train the network: find the parameters $\mathbf{w}_{ij}$
that minimize a *Loss function:*

$$\mathcal{L} \propto \sum_i |\hat{y}_i - y_i| = \sum_i |\hat{y}_i - N(\vec{x}_i; \vec{w})|$$

Back to Neural Networks

Feed forward architecture:

The process consists in applying linear transformations W_l alternating with non-linear functions g_l



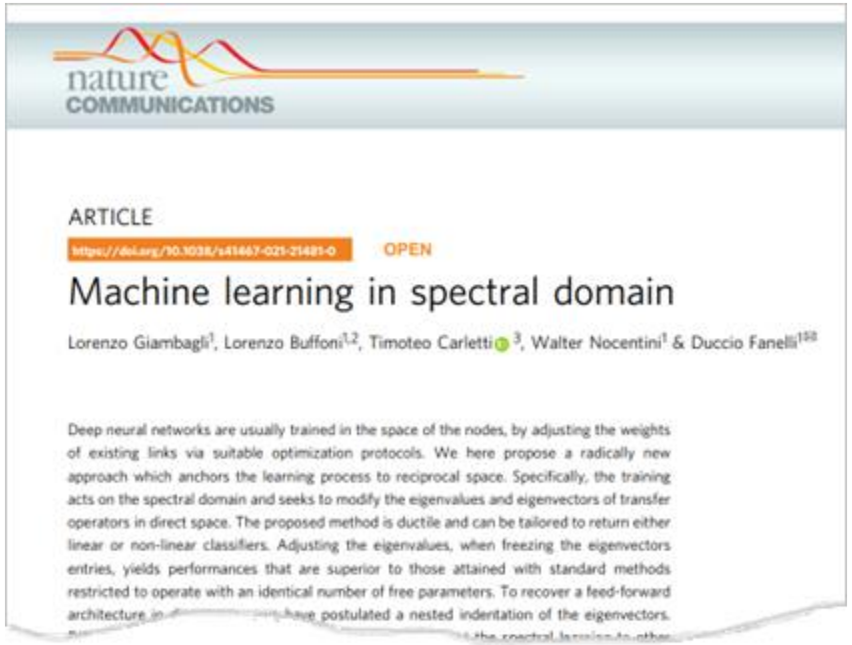
$$\vec{x}_i \rightarrow \boxed{W_1(\vec{x}_i) \rightarrow g_1\left(W_1(\vec{x}_i)\right)} \rightarrow \boxed{W_2\left(g_1\left(W_1(\vec{x}_i)\right)\right) \rightarrow g_2\left(W_2\left(g_1\left(W_1(\vec{x}_i)\right)\right)\right)} \rightarrow \cdots$$

Layer 1 — Layer 2

Deep Learning in spectral domain

Giambagli et al. [1] recently introduced the idea of training neural networks by acting in the **reciprocal space**.

The main idea is to tune the spectral parameters instead of the weights of the matrix W.


nature COMMUNICATIONS

ARTICLE

https://doi.org/10.1038/s41467-021-21481-0 OPEN

Machine learning in spectral domain

Lorenzo Giambagli[1], Lorenzo Buffoni[1,2], Timoteo Carletti[3], Walter Nocentini[1] & Duccio Fanelli[1]

Deep neural networks are usually trained in the space of the nodes, by adjusting the weights of existing links via suitable optimization protocols. We here propose a radically new approach which anchors the learning process to reciprocal space. Specifically, the training acts on the spectral domain and seeks to modify the eigenvalues and eigenvectors of transfer operators in direct space. The proposed method is ductile and can be tailored to return either linear or non-linear classifiers. Adjusting the eigenvalues, when freezing the eigenvectors entries, yields performances that are superior to those attained with standard methods restricted to operate with an identical number of free parameters. To recover a feed-forward architecture in ... have postulated a nested indentation of the eigenvectors.

$$W = \Phi \Lambda \Phi^{-1}$$

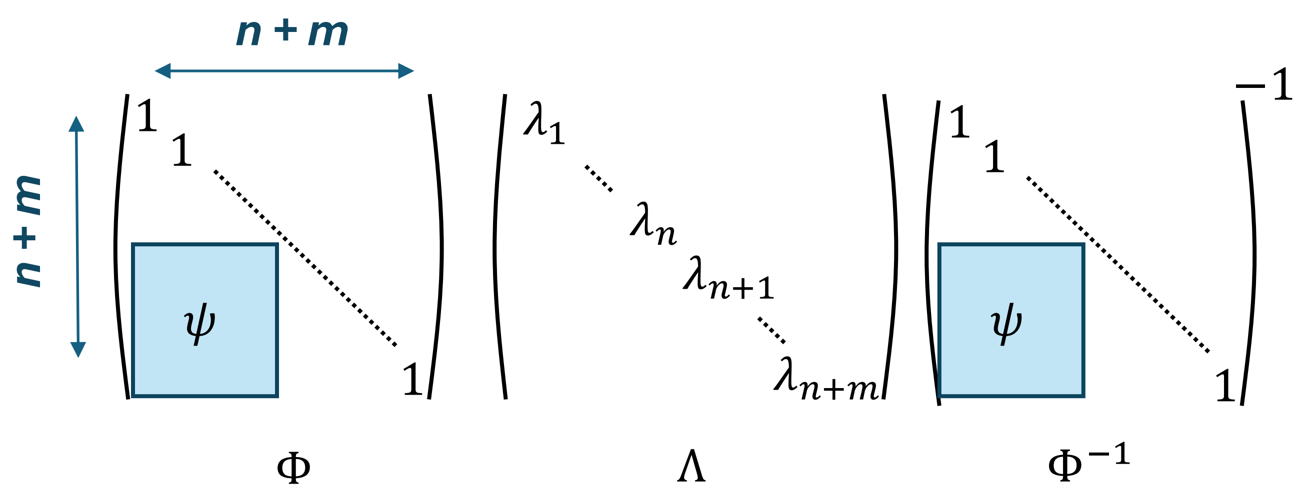
↓

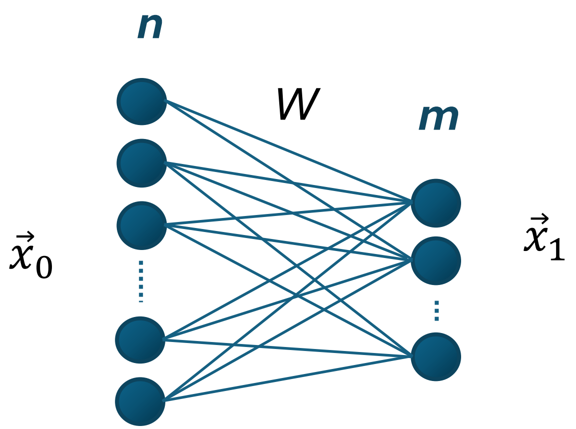
Spectral decomposition

[1] L. Giambagli, L. Buffoni, T. Carletti, W. Nocentini, and D.Fanelli, Machine learning in spectral domain, Nat. Commun. 12, 1330 (2021).

Deep Learning in spectral domain

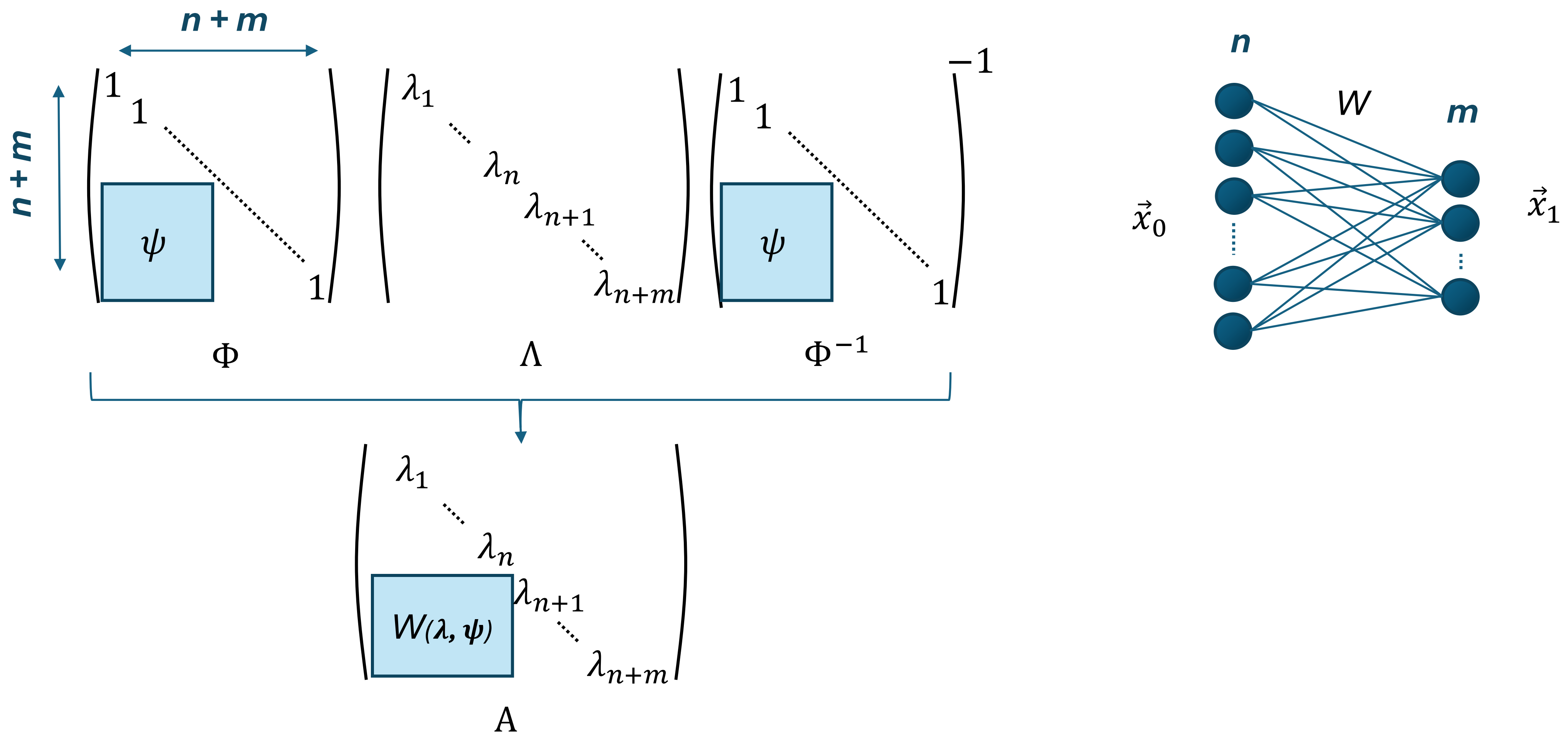
Starting from the reciprocal space, a suitable set of eigenvectors Φ can be chosen so as to obtain in the direct space a matrix **A** that describes the linear transfer between two consecutive layers.





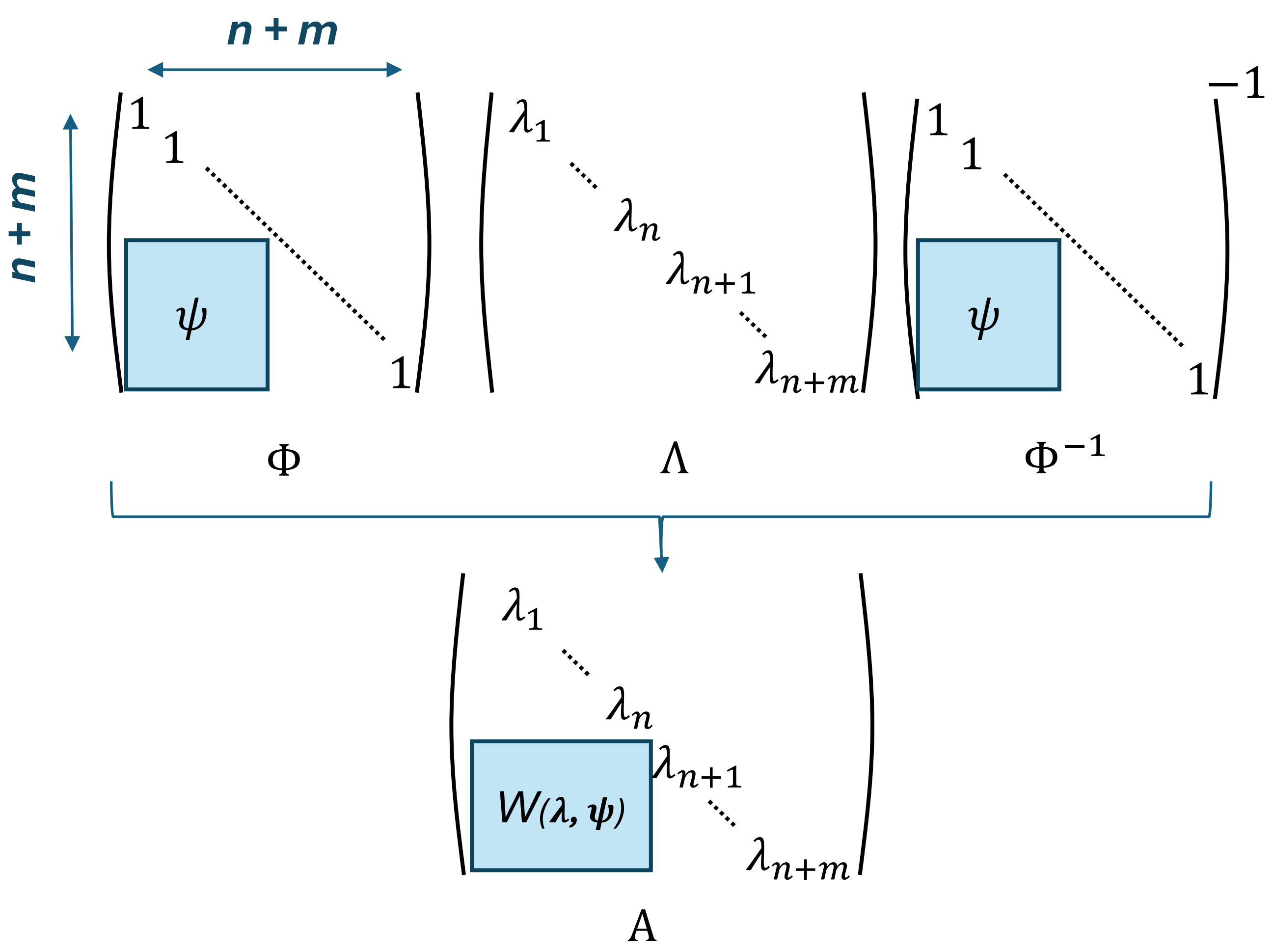
Deep Learning in spectral domain

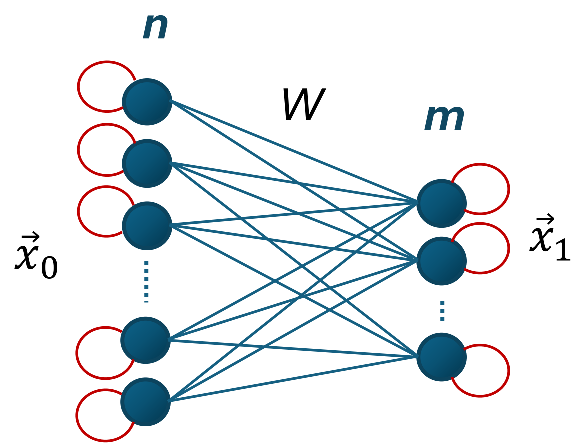
Starting from the reciprocal space, a suitable set of eigenvectors Φ can be chosen so as to obtain in the direct space a matrix A that describes the linear transfer between two consecutive layers.



Deep Learning in spectral domain

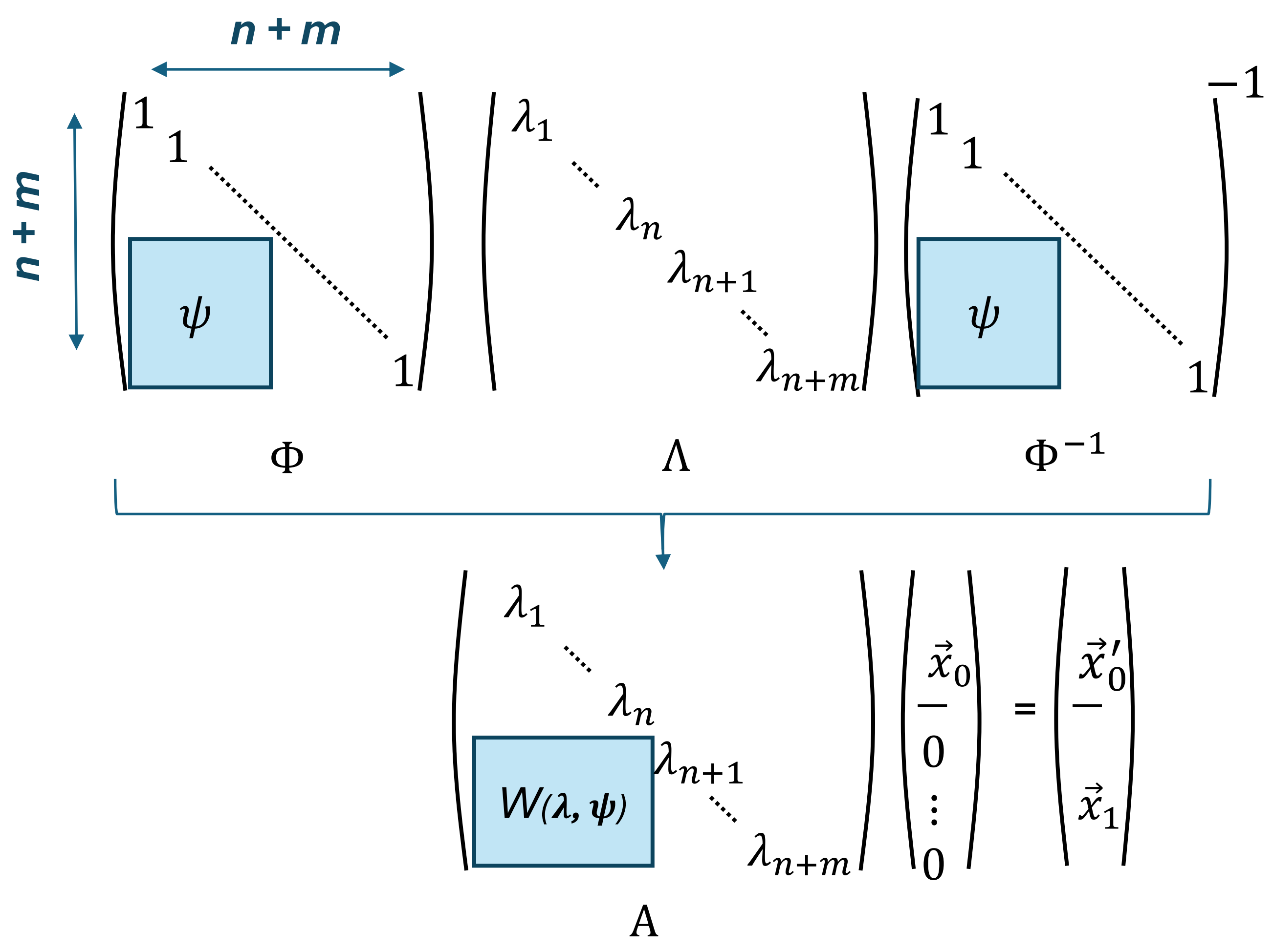
Starting from the reciprocal space, a suitable set of eigenvectors Φ can be chosen so as to obtain in the direct space a matrix A that describes the linear transfer between two consecutive layers.

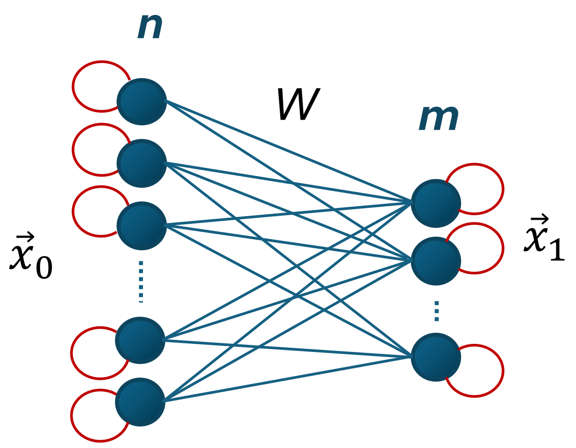




Deep Learning in spectral domain

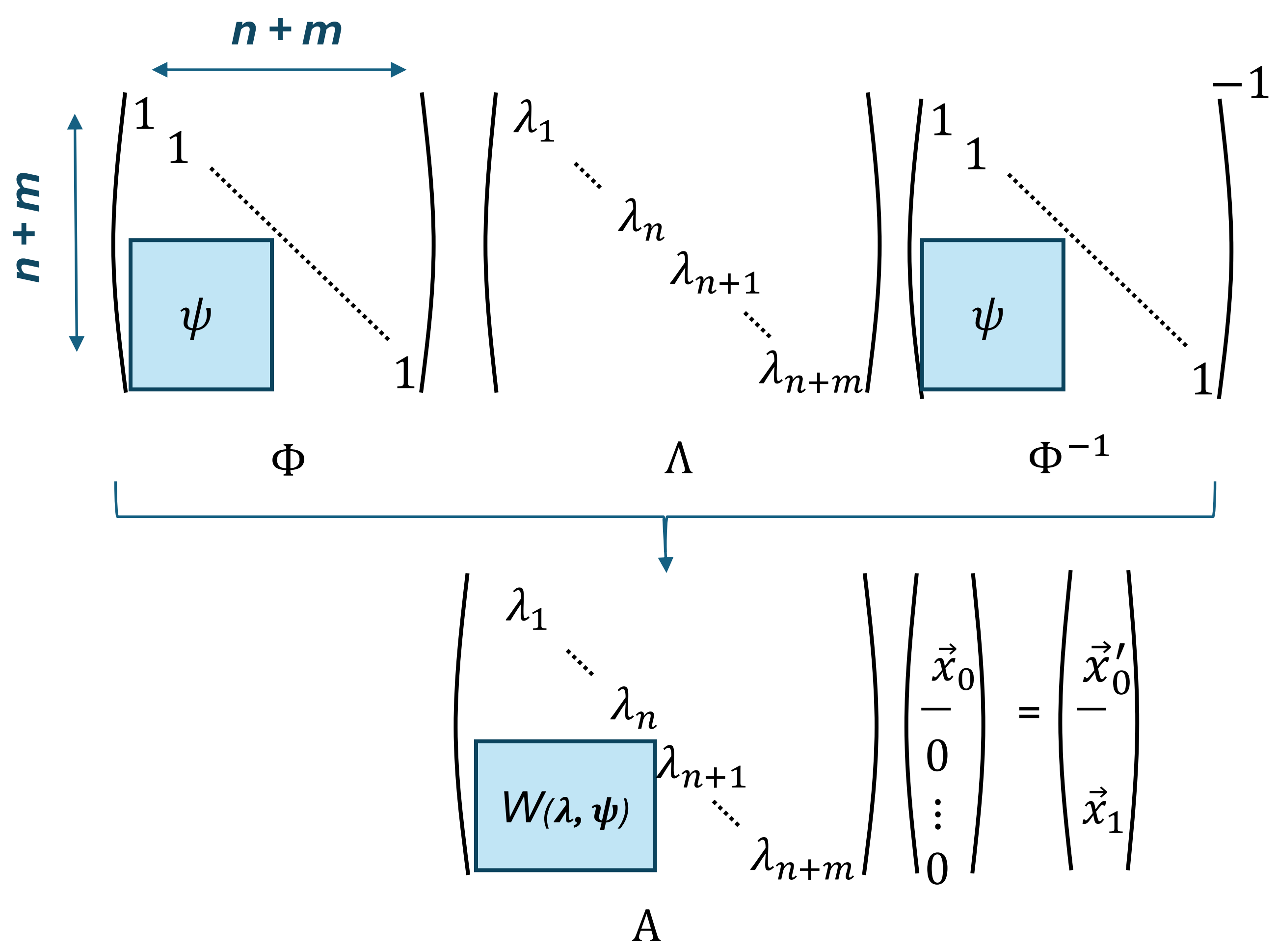
Starting from the reciprocal space, a suitable set of eigenvectors Φ can be chosen so as to obtain in the direct space a matrix A that describes the linear transfer between two consecutive layers.

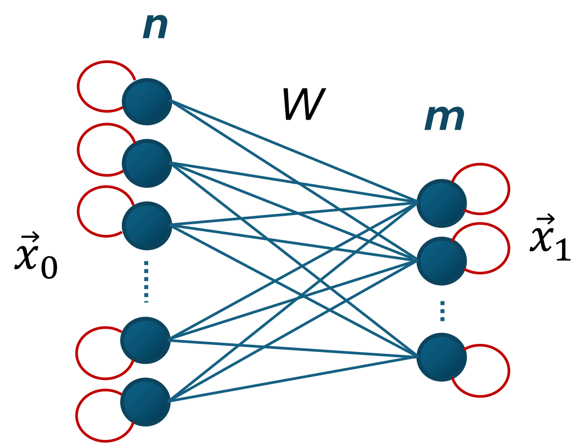




Deep Learning in spectral domain

Starting from the reciprocal space, a suitable set of eigenvectors Φ can be chosen so as to obtain in the direct space a matrix A that describes the linear transfer between two consecutive layers.





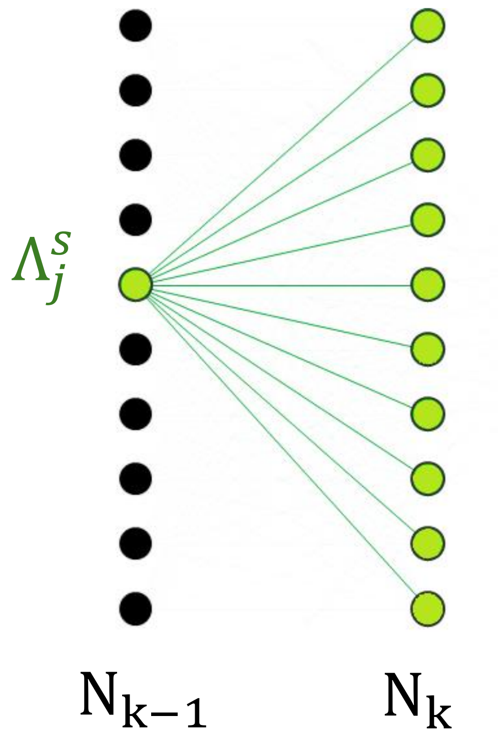
This leads to a new set of trainable parameters

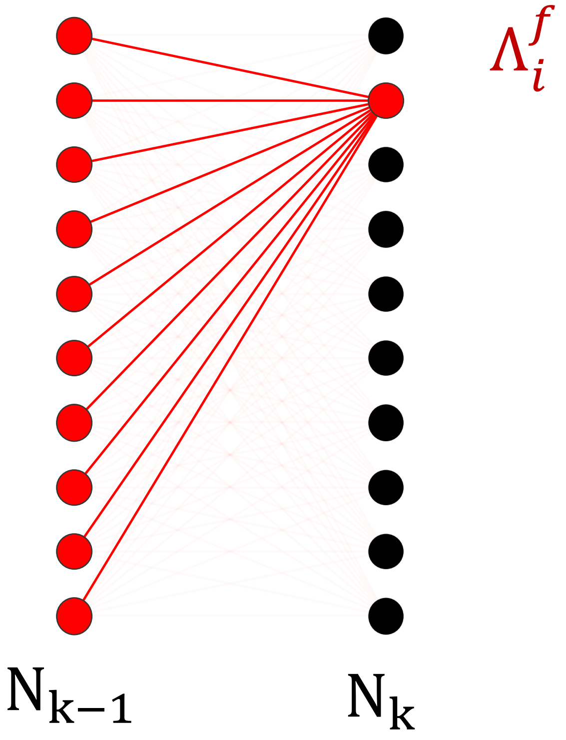
$$w_{ij} = (\lambda_j - \lambda_i)\psi_{ij}$$

Deep Learning in spectral domain

$$\sum_j w_{ij}^{(k)} x_j^{(k-1)} = \sum_j \psi_{\mathrm{ij}} \Lambda_j^s x_j^{(k-1)} - \Lambda_i^f \psi_{\mathrm{ij}} x_j^{(k-1)}$$

The eigenvalues identify the way information flows in the linear transfer

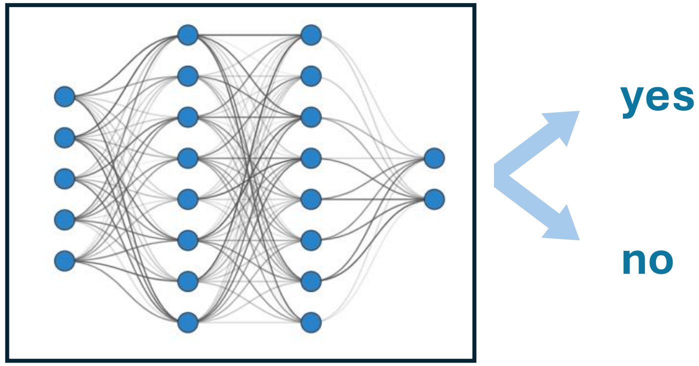




Input feature relevance

Explainability: what does the network is looking at to take its decisions?

Input vector $\begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ \vdots \end{pmatrix}$



It is crucial to define methods to identify what features are relevant!

Input feature relevance

$$\Lambda_i^f = 0$$

$$\sum_j w_{ij}^{(k)} x_j^{(k-1)} = \sum_j \psi_{\text{ij}} \Lambda_j^s x_j^{(k-1)} - \Lambda_i^f \psi_{\text{ij}} x_j^{(k-1)}$$

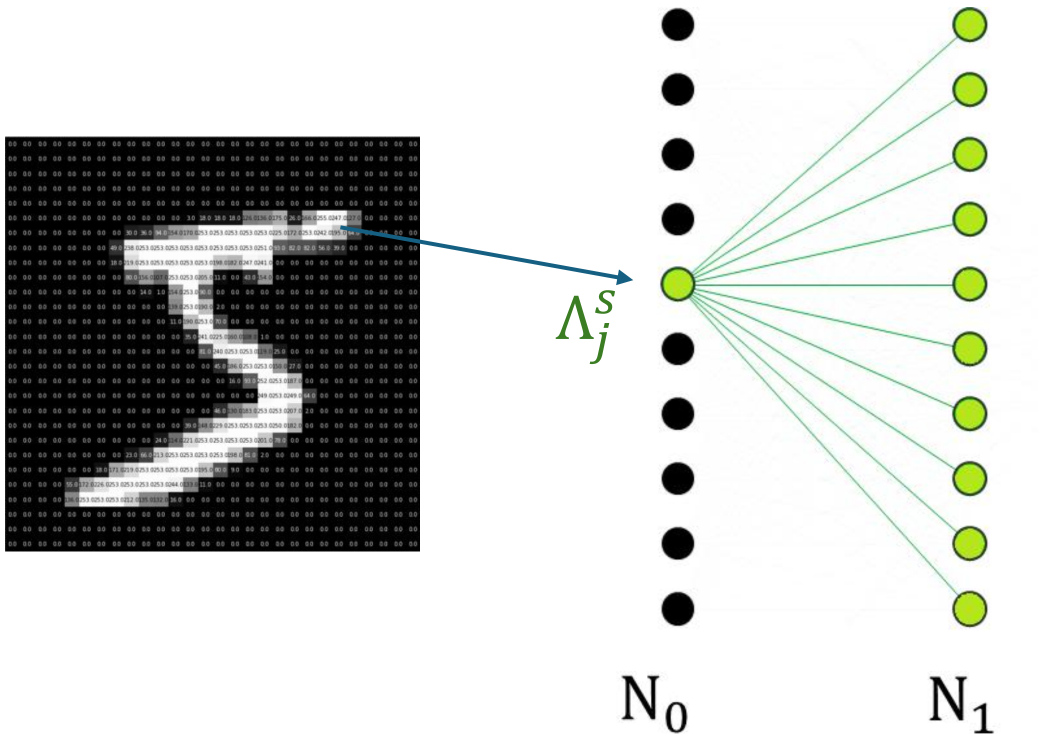
→

$$w_{ij} = \lambda_j \psi_{ij}$$

Input feature relevance

$$\Lambda_i^f = 0$$

$$\sum_j w_{ij}^{(k)} x_j^{(k-1)} = \sum_j \psi_{ij} \Lambda_j^s x_j^{(k-1)} - \cancel{\Lambda_i^f \psi_{ij} x_j^{(k-1)}} \quad \rightarrow \quad w_{ij} = \lambda_j \psi_{ij}$$



Not all the components of the input are important to solve the task

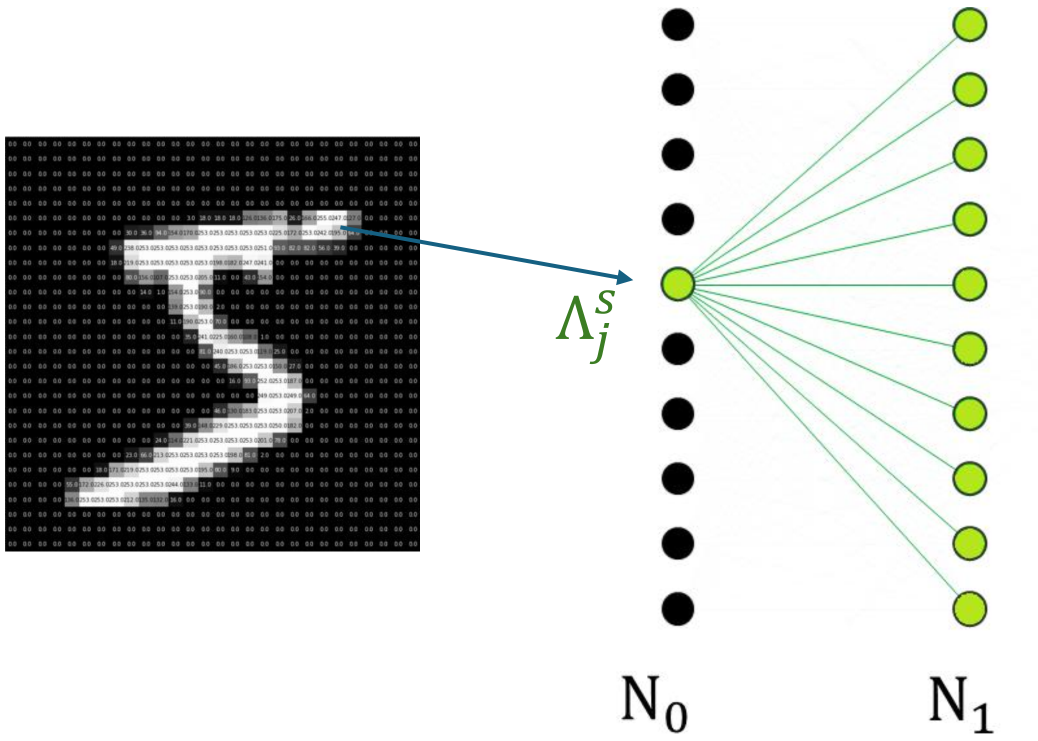
Can eigenvalues be a proxy of the relevance?

Input feature relevance

$$\Lambda_i^f = 0$$

$$\sum_j w_{ij}^{(k)} x_j^{(k-1)} = \sum_j \psi_{ij} \Lambda_j^s x_j^{(k-1)} - \cancel{\Lambda_i^f \psi_{ij} x_j^{(k-1)}}$$

$\rightarrow$ $w_{ij} = \lambda_j \psi_{ij}$

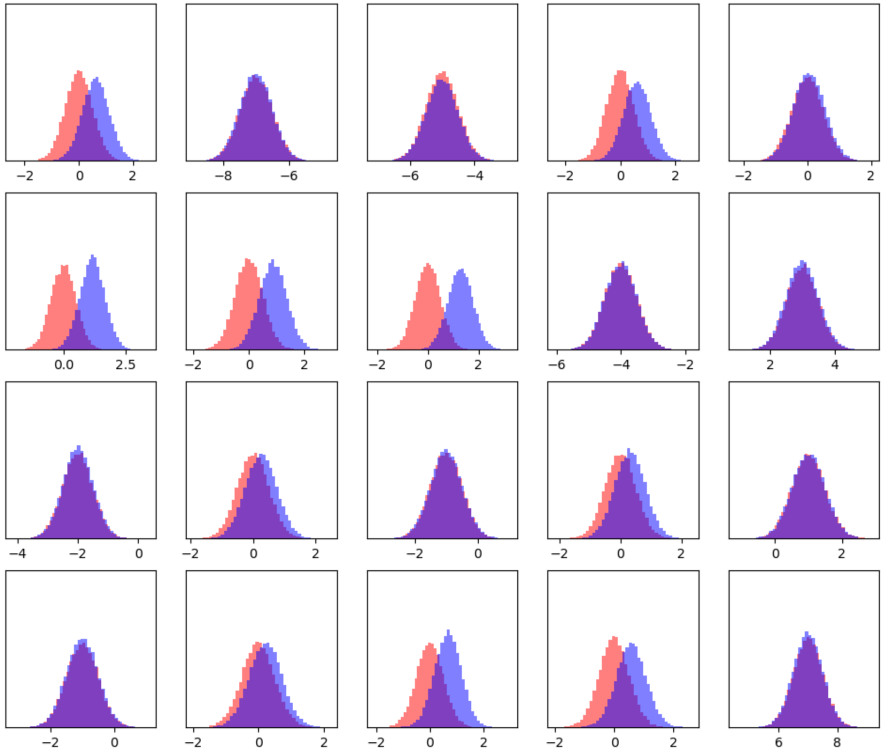


Not all the components of the input are important to solve the task

Can eigenvalues be a proxy of the relevance?

A simple handmade dataset

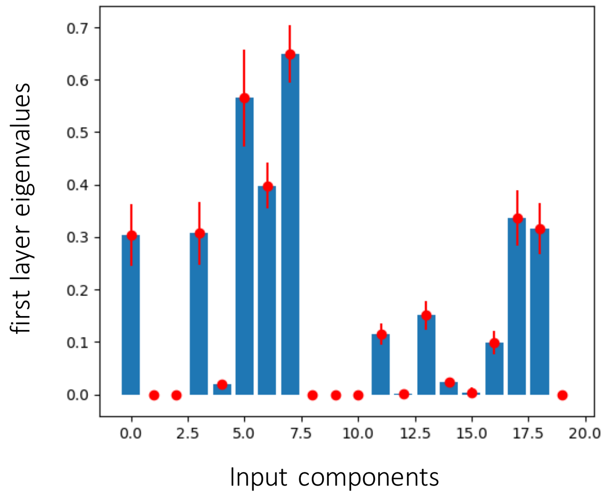
- Classification problem with two classes $c \in [0,1]$
- $\vec{x}_0 \in R^{20}$
- x_i from independent Gaussian distributions $(\mu_i^c\ ,\ \sigma_i^c)$
- Some features are irrelevant: $\mu_i^0 = \mu_i^1$ and $\sigma_i^0 = \sigma_i^1$
- For some components, the two classes are partially separated. We define the relevance of the features as the *distance* between the two distributions.

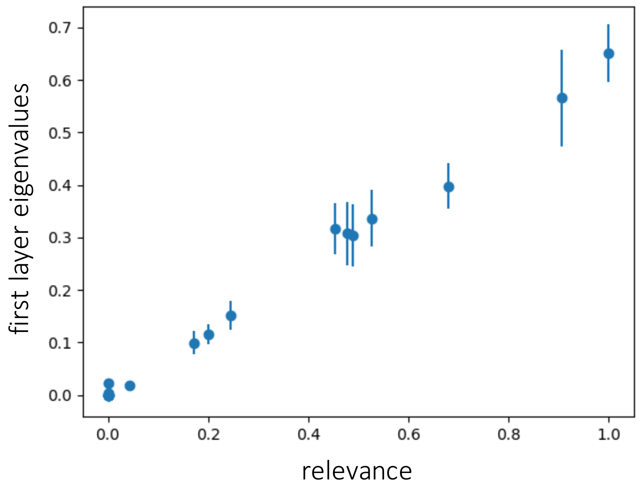


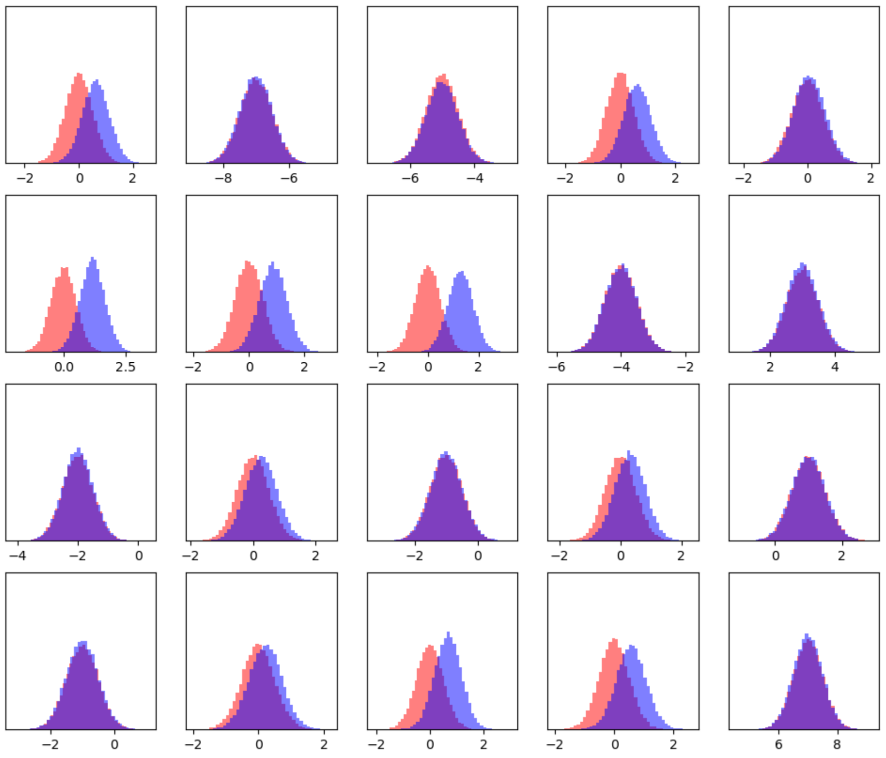
distributions of the input features

A simple handmade dataset

- Classification problem with two classes $c \in [0,1]$
- $\vec{x}_0 \in R^{20}$
- x_i from independent Gaussian distributions $(\mu_i^c\ ,\ \sigma_i^c)$
- Some features are irrelevant: $\mu_i^0 = \mu_i^1$ and $\sigma_i^0 = \sigma_i^1$
- For some components, the two classes are partially separated. We define the relevance of the features as the *distance* between the two distributions.







distributions of the input features

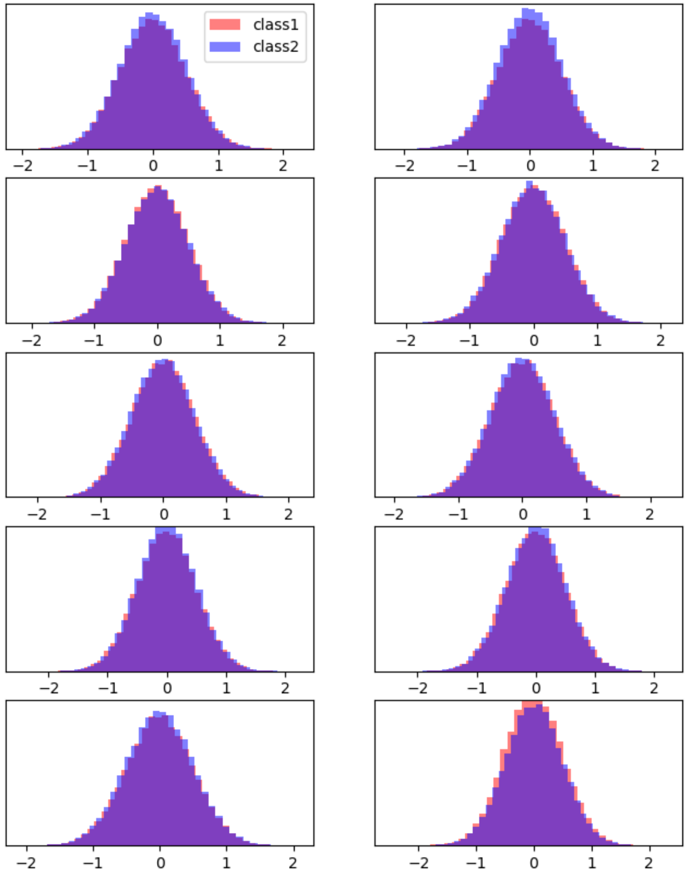
Finding correlations

- Classification problem with two classes $c \in [0,1]$
- $\vec{x}_0 \in R^{10}$
- if $c(\vec{x}) = \ 0$:

$$x_{2n+1} = x_{2n}$$

- if $c(\vec{x}) = \ 1$:

$$x_{2n+1} = \begin{cases} -x_{2n} & \text{with prob. } p \\ x_{2n} & \text{with prob. } 1-p \end{cases}$$



distributions of the input features

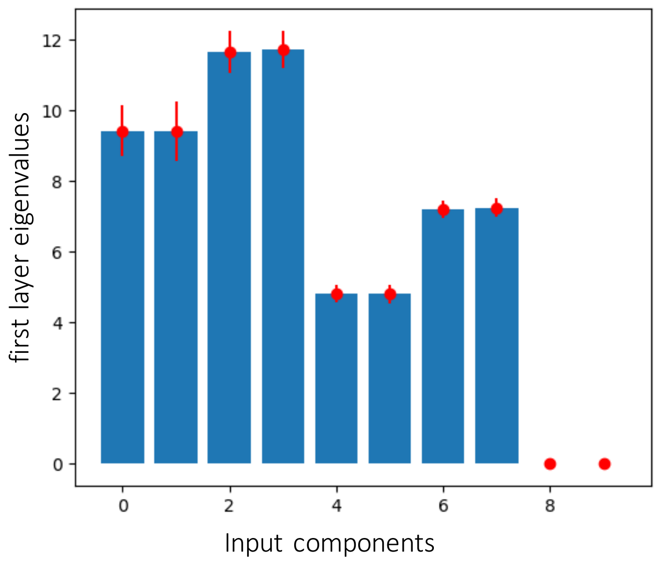
Finding correlations

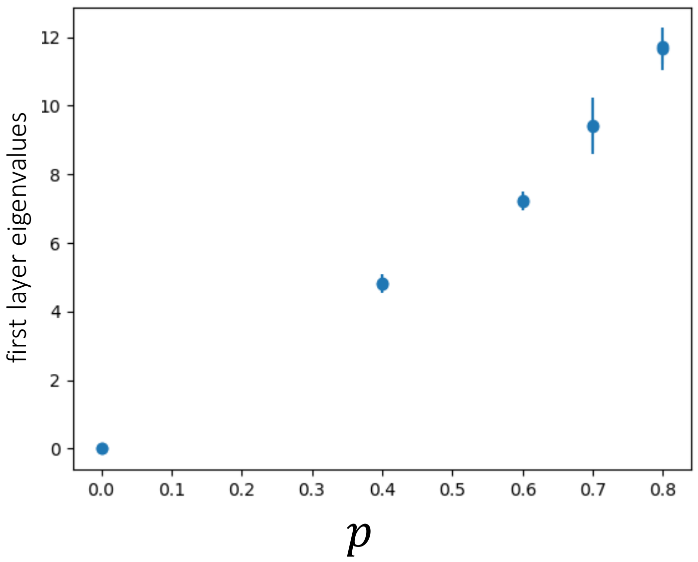
- Classification problem with two classes $c \in [0,1]$
- $\vec{x}_0 \in R^{10}$
- if $c(\vec{x}) = \; 0$:

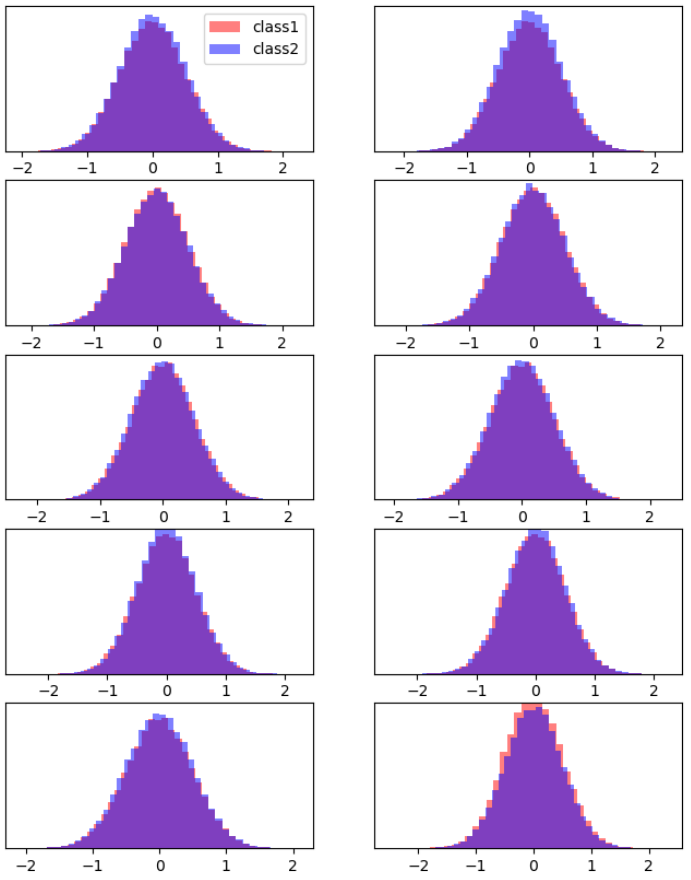
$$x_{2n+1} = x_{2n}$$

- if $c(\vec{x}) = \; 1$:

$$x_{2n+1} = \begin{cases} -x_{2n} & \text{with prob.}\, p \\ x_{2n} & \text{with prob.}\, 1-p \end{cases}$$



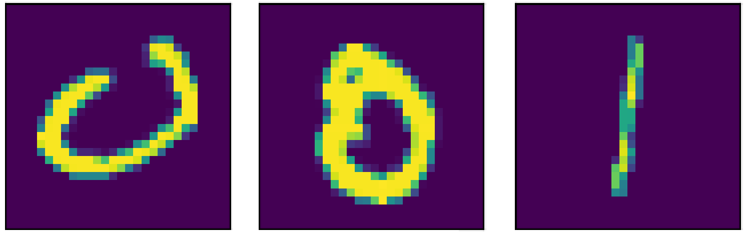




distributions of the input features

The MNIST dataset

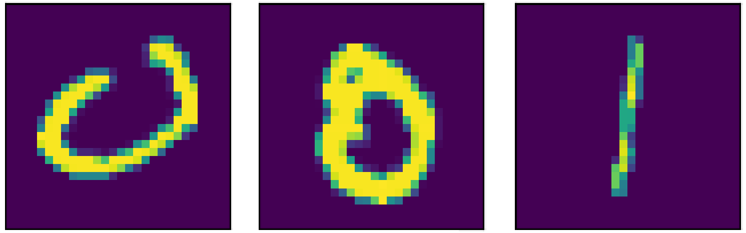
A simple real dataset: images of zeros and ones → $\vec{x}_0 \in R^{784}$



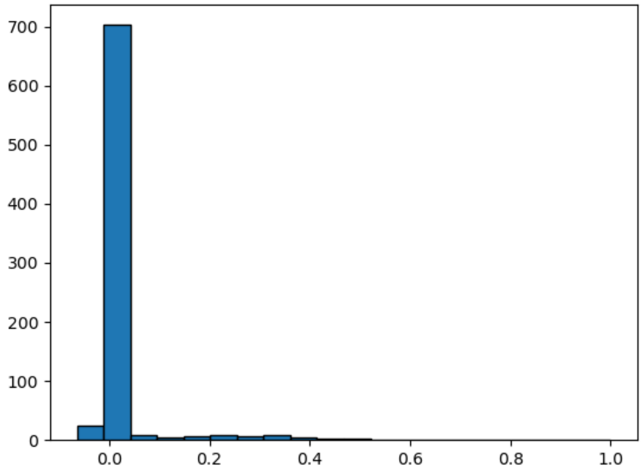
....

The MNIST dataset

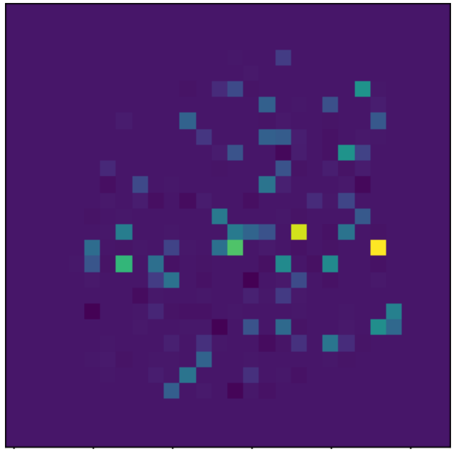
A simple real dataset: images of zeros and ones → $\vec{x}_0 \in R^{784}$



....



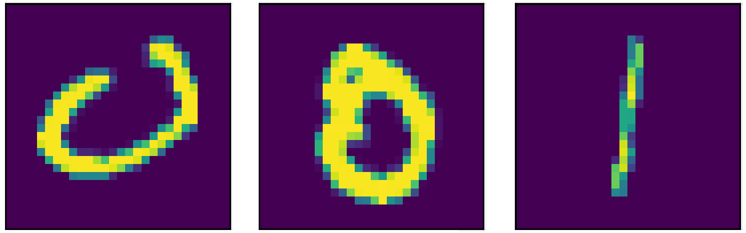
eigenvalues



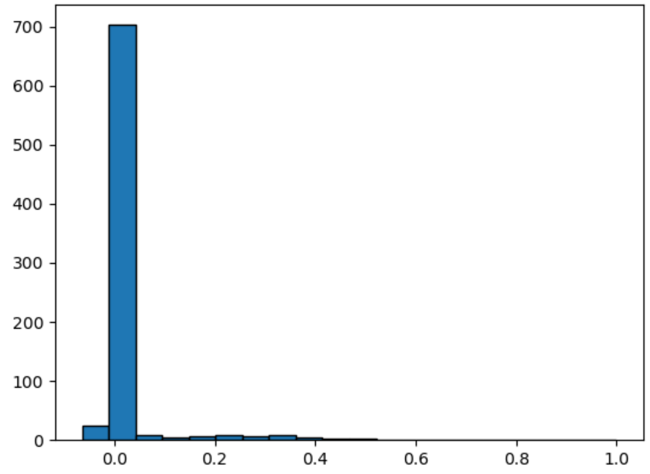
Colormap: eigenvalues

The MNIST dataset

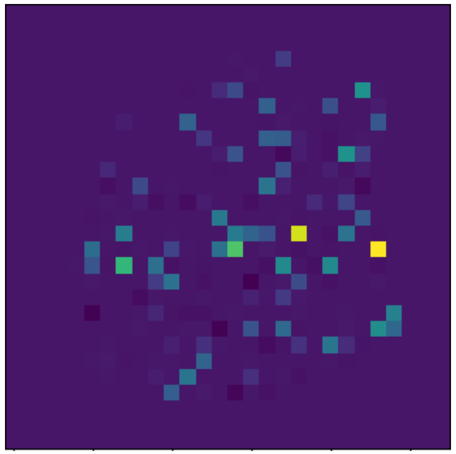
A simple real dataset: images of zeros and ones $\longrightarrow \vec{x}_0 \in R^{784}$



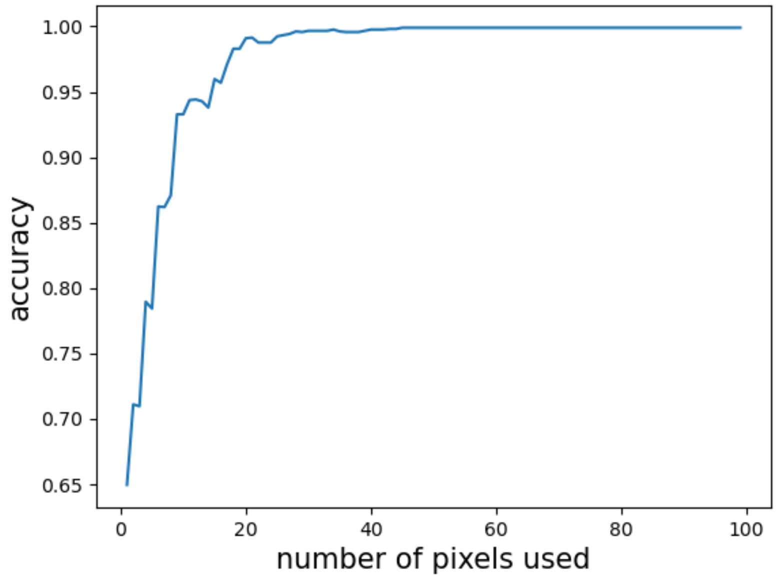
....



eigenvalues

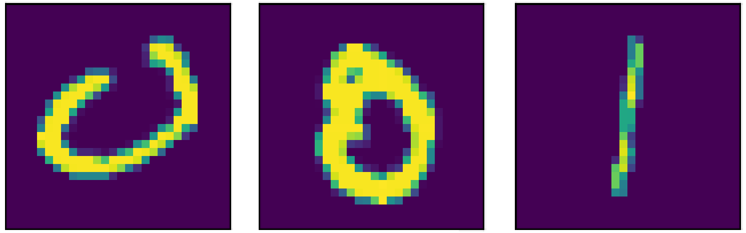


Colormap: eigenvalues

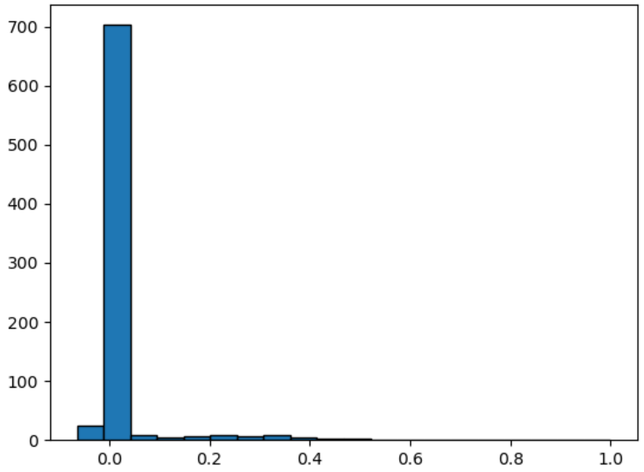


The MNIST dataset

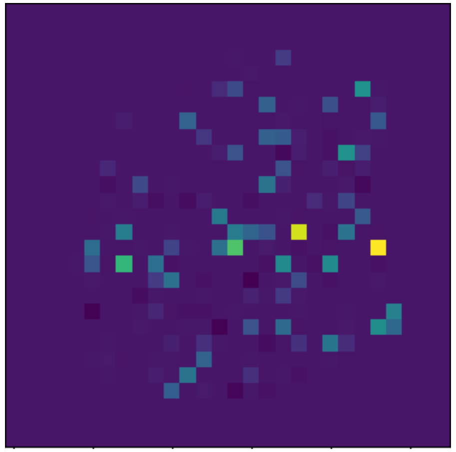
A simple real dataset: images of zeros and ones $\longrightarrow \vec{x}_0 \in R^{784}$



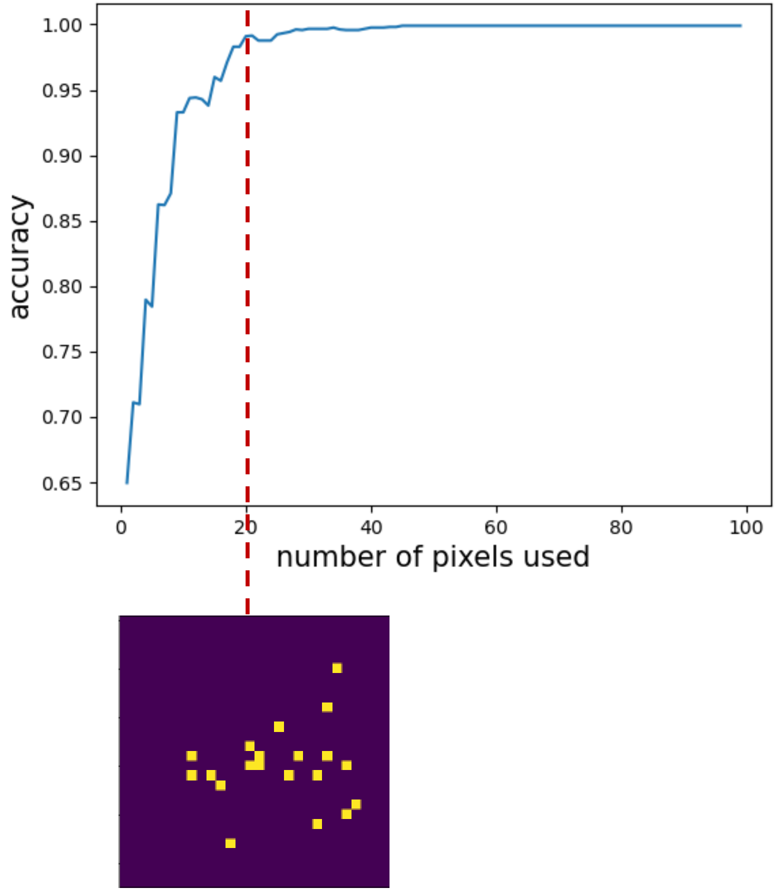
....



eigenvalues

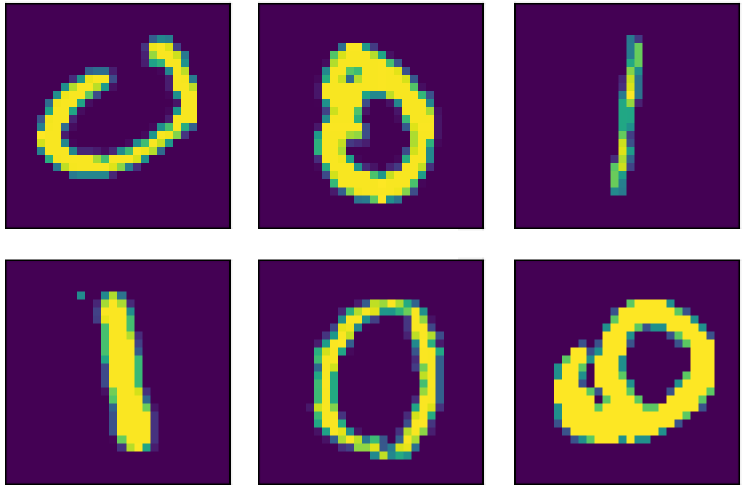


Colormap: eigenvalues



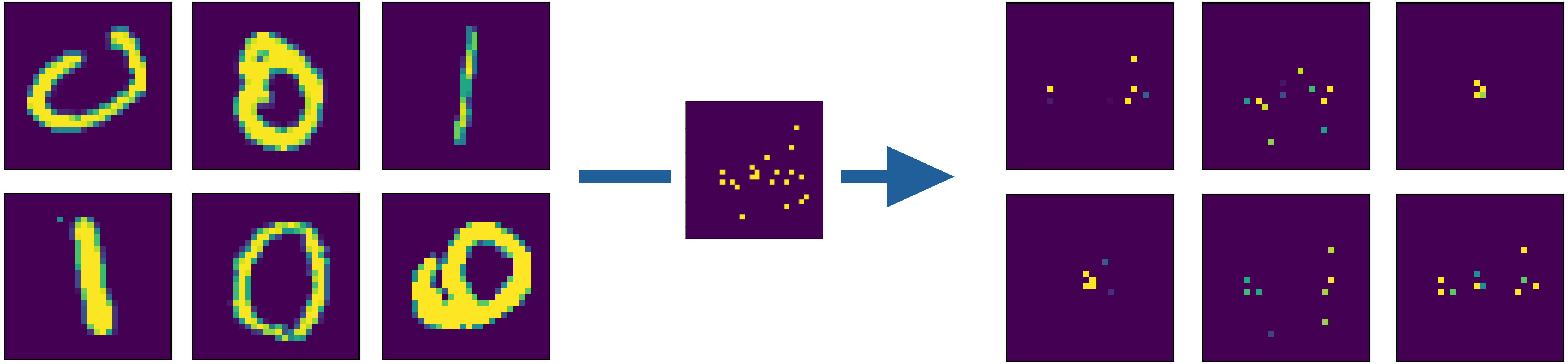
The MNIST dataset

By applying the mask to the input images, we see what the network is looking at:



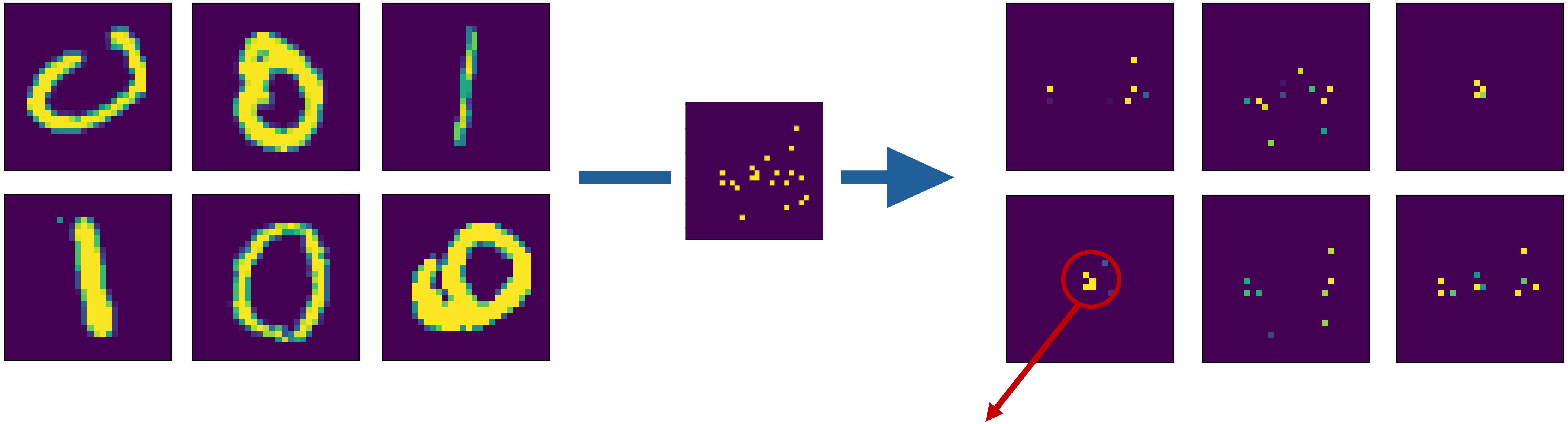
The MNIST dataset

By applying the mask to the input images, we see what the network is looking at:



The MNIST dataset

By applying the mask to the input images, we see what the network is looking at:



To understand if the input is a “one”, the network looks if this central part is active

Conclusion

- **Spectral formulation** gives us node-related parameters by which we can rank nodes (pruning strategy)
- By applying this idea to **first-layer nodes** it is possible to identify relevant features: we tested on two handmade dataset and one very simple real dataset.
- Real dataset are way more complicated, but this analysis is a first attempt to **connect post-training eigenvalues to input dimensions relevance**

References:

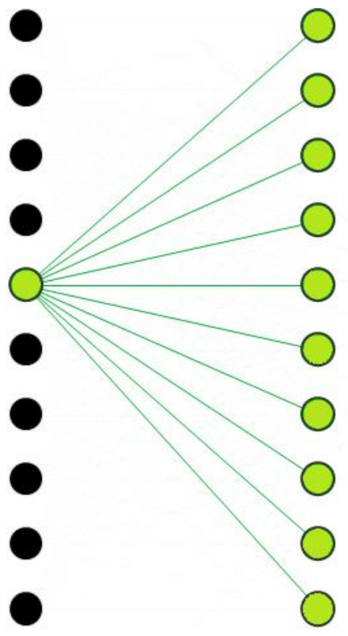
- Less parameters same performance (Chicchi, Lorenzo, et al. *Physical Review E* 104.5 (2021): 054312.)
- Spectral Pruning (Buffoni, Lorenzo, et al. *Scientific reports* 12.1 (2022): 1-9.)
- Recurrent Spectral Learning (Chicchi, Lorenzo, et al., *Chaos, Solitons & Fractals* 168 (2023): 113128.)
- Complex Recurrent Spectral Network (Chicchi, Lorenzo, et al., *arXiv preprint arXiv:2312.07296* (2023).)
- How a student becomes a teacher: learning and forgetting through Spectral methods (Giambagli, Lorenzo, et al, *NeurIPS* 2023)

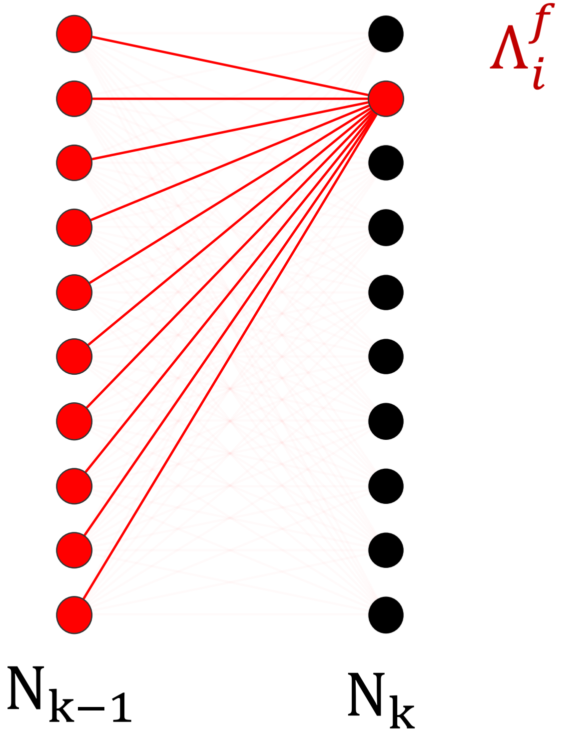
Thank you!

Deep Learning in spectral domain

$$\sum_j w_{ij}^{(k)} x_j^{(k-1)} = \sum_j \psi_{ij} \Lambda_j^s x_j^{(k-1)} - \Lambda_i^f \psi_{ij} x_j^{(k-1)}$$

$\Lambda_j^s = 0$





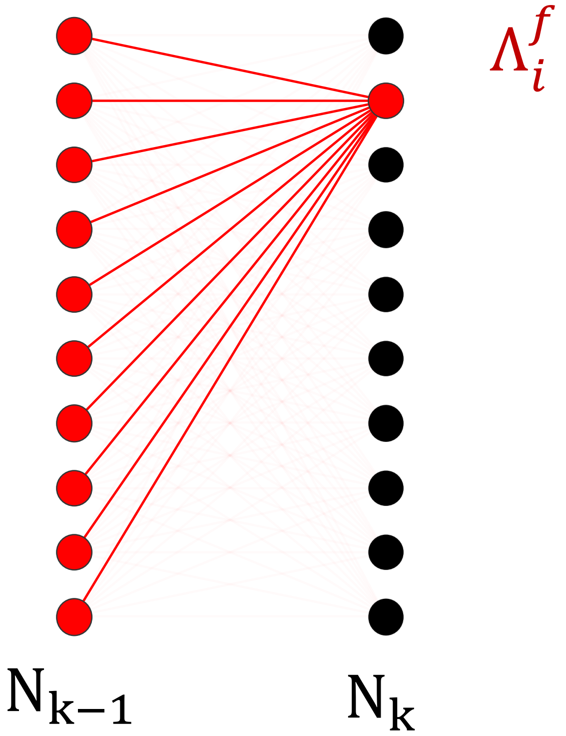
Buffoni, Lorenzo, et al. "Spectral pruning of fully connected layers." *Scientific Reports* 12.1 (2022): 11201.

Deep Learning in spectral domain

$$\sum_j w_{ij}^{(k)} x_j^{(k-1)} = \sum_j \psi_{ij} \Lambda_j^s x_j^{(k-1)} - \Lambda_i^f \psi_{ij} x_j^{(k-1)}$$

Λ_j^s=0

Every eigenvalues is now associated with one node : **Pruning strategy**



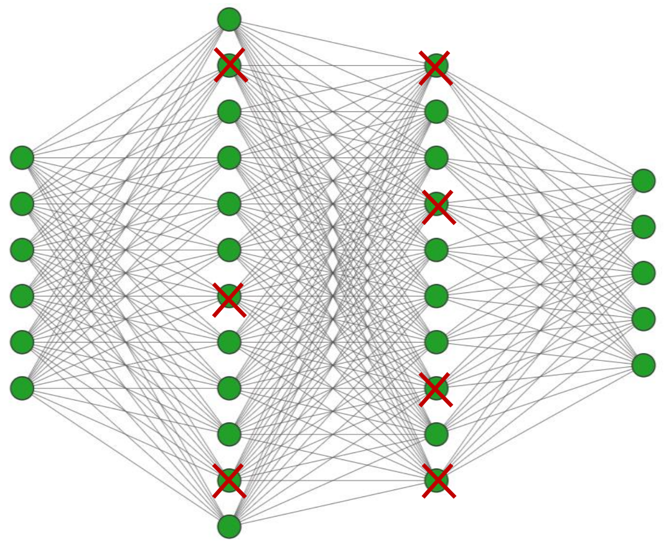
Buffoni, Lorenzo, et al. "Spectral pruning of fully connected layers." *Scientific Reports* 12.1 (2022): 11201.

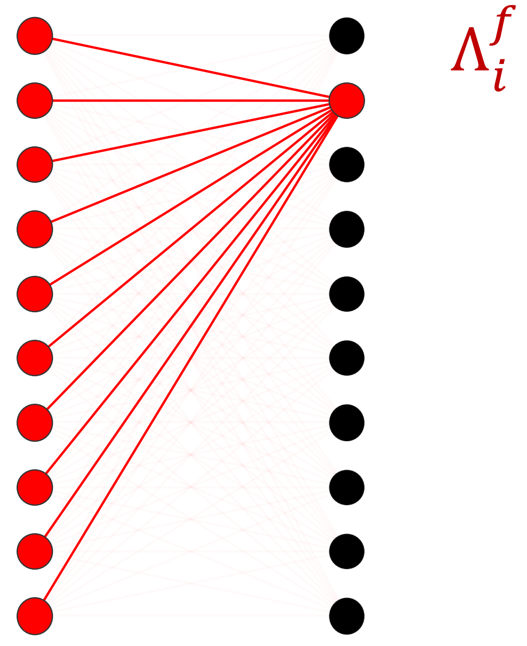
Deep Learning in spectral domain

$$\sum_j w_{ij}^{(k)} x_j^{(k-1)} = \sum_j \psi_{ij} \Lambda_j^s x_j^{(k-1)} - \Lambda_i^f \psi_{ij} x_j^{(k-1)}$$

Λ_j^s=0

Every eigenvalues is now associated with one node : **Pruning strategy**





Buffoni, Lorenzo, et al. "Spectral pruning of fully connected layers." *Scientific Reports* 12.1 (2022): 11201.