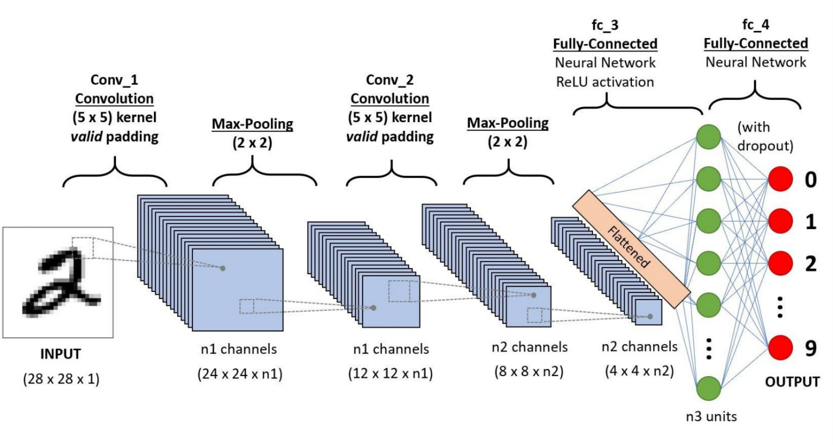
Mathematical foundations in deep learning
Part VIII: Unfolded algorithms

Nelly Pustelnik

CNRS, Laboratoire de Physique de l'ENS de Lyon, France

Deep learning: generalities



(extracted from: datasciencepr.com)

- Deep networks are composed of a stack of layers.
- Each layer is composed with linear transforms (e.g. convolution, pooling), nonlinear transforms (i.e. activation functions).

Feedforward neural network model

- Database: $\mathcal{S} = \left\{(\overline{u}_\ell, z_\ell) \in \mathcal{H} \times \mathcal{G} \;\middle|\; \ell \in \{1, \dots, L\}\right\}$

- Goal: Learn a prediction function d_Θ

$$\widehat{\Theta} \in \underset{\Theta}{\mathrm{Argmin}}\; \mathrm{E}(\Theta) := \frac{1}{L}\sum_{\ell=1}^{L} f\big(z_\ell, \mathrm{d}_\Theta(u_\ell)\big) \tag{1}$$

- Feedforward neural network model:

$$\boxed{\mathrm{d}_\Theta(u_\ell) = \eta^{[K]}\big(W^{[K]} \dots \eta^{[1]}(W^{[1]} u_\ell + b^{[1]}) \dots + b^{[K]}\big)}$$

  where for every $k \in \{1, \dots, K\}$,
  - $W^{[k]}$ denotes a weight matrix,
  - $b^{[k]}$ is a bias vector,
  - $\eta^{[k]}$ is the nonlinear activation function.

Standard activation functions

- Most basic: $\eta = \mathrm{Id} = \mathrm{prox}_0$,
- Saturated linear activation function: $\eta = \mathrm{prox}_{\iota_C}$ with $C = [-1, 1]$,
- Rectified linear unit (ReLU): $\eta = \mathrm{prox}_{\iota_C}$ with $C = [0, +\infty[$,
- Parametric ReLU,
- Bent identity,
- Inverse square root,
- Unimodal sigmoid
- Elliot function
- Softmax

  $\to$ Most of activation functions are proximity operators.

  $\to$ **Exhaustive list** in P. L. Combettes and J.-C. Pesquet, Deep neural network structures solving variational inequalities, Set-Valued and Variational Analysis, vol. 28, pp. 491–518, September 2020. [PDF]

Feedforward neural network model

- **Database**: $\mathcal{S} = \big\{(\overline{u}_\ell, z_\ell) \in \mathcal{H} \times \mathcal{G} \;\big|\; \ell \in \{1,\ldots,L\}\big\}$

- **Goal**: Learn a prediction function d_Θ

$$\widehat{\Theta} \in \operatorname*{Argmin}_{\Theta} \mathrm{E}(\Theta) := \frac{1}{L}\sum_{\ell=1}^{l} f\big(z_\ell, \mathrm{d}_\Theta(u_\ell)\big)$$

- **Feedforward neural network model**:

$$\boxed{\mathrm{d}_\Theta(u_\ell) = \eta^{[K]}\big(W^{[K]} \ldots \eta^{[1]}(W^{[1]}u_\ell + b^{[1]}) \ldots + b^{[K]}\big)}$$

  where for every $k \in \{1,\ldots,K\}$,
  - $W^{[k]}$ denotes a weight matrix,
  - $b^{[k]}$ is a bias vector,
  - $\eta^{[k]}$ is the nonlinear activation function.

Feedforward neural network model

- **Database**: $\mathcal{S} = \left\{ (\overline{u}_\ell, z_\ell) \in \mathcal{H}_0 \times \mathcal{H}_K \;\middle|\; \ell \in \{1, \ldots, L\} \right\}$

- **Goal**: Learn a prediction function d_Θ

$$\widehat{\Theta} \in \underset{\Theta}{\mathrm{Argmin}}\; \mathrm{E}(\Theta) := \frac{1}{L} \sum_{\ell=1}^{L} f\big(z_\ell, \mathrm{d}_\Theta(u_\ell)\big)$$

- **Feedforward neural network model**:

$$\boxed{\mathrm{d}_\Theta(u_\ell) = \eta^{[K]}\big(W^{[K]} \ldots \eta^{[1]}(W^{[1]} u_\ell + b^{[1]}) \ldots + b^{[K]}\big)}$$

where for every $k \in \{1, \ldots, K\}$,

- $W^{[k]}$ denotes a weight matrix,
- $b^{[k]}$ is a bias vector,
- $\eta^{[k]}$ is the nonlinear activation function.

Feedforward neural network model

- **Database**: $\mathcal{S} = \left\{(\overline{u}_\ell, z_\ell) \in \mathcal{H}_0 \times \mathcal{H}_K \;\middle|\; \ell \in \{1, \ldots, L\}\right\}$

- **Goal**: Learn a prediction function d_Θ

$$\widehat{\Theta} \in \underset{\Theta}{\mathrm{Argmin}}\; \mathrm{E}(\Theta) := \frac{1}{L} \sum_{\ell=1}^{l} f\big(z_\ell, \mathrm{d}_\Theta(u_\ell)\big)$$

- **Feedforward neural network model**:

$$\boxed{\mathrm{d}_\Theta(u_\ell) = \mathrm{prox}_{f^{[K]}}\big(W^{[K]} \ldots \mathrm{prox}^{[1]}_{f^{[1]}}(W^{[1]} u_\ell + b^{[1]}) \ldots + b^{[K]}\big)}$$

where for every $k \in \{1, \ldots, K\}$,

- $W^{[k]}$ denotes a weight matrix,
- $b^{[k]}$ is a bias vector,
- $\eta^{[k]}$ is the nonlinear activation function.

Feedforward neural network model

- **Database**: $\mathcal{S} = \left\{ (\overline{u}_\ell, z_\ell) \in \mathcal{H}_0 \times \mathcal{H}_K \;\middle|\; \ell \in \{1, \ldots, L\} \right\}$

- **Goal**: Learn a prediction function d_Θ

$$\widehat{\Theta} \in \underset{\Theta}{\operatorname{Argmin}}\, \mathrm{E}(\Theta) := \frac{1}{L} \sum_{\ell=1}^{I} f\big(z_\ell, \mathrm{d}_\Theta(u_\ell)\big)$$

- **Feedforward neural network model**:

$$\boxed{\mathrm{d}_\Theta(u_\ell) = \mathrm{prox}_{f^{[K]}}\big(W^{[K]} \ldots \mathrm{prox}_{f^{[1]}}^{[1]}(W^{[1]} u_\ell + b^{[1]}) \ldots + b^{[K]}\big)}$$

where for every $k \in \{1, \ldots, K\}$,

- $W^{[k]} \colon \mathcal{H}_{k-1} \to \mathcal{H}_k$ denotes a bounded linear operator,
- $b^{[k]}$ is a bias vector,
- $\eta^{[k]}$ is the nonlinear activation function.

Feedforward neural network model

- **Database**: $\mathcal{S} = \big\{(\overline{u}_\ell, z_\ell) \in \mathcal{H}_0 \times \mathcal{H}_K \;\big|\; \ell \in \{1, \ldots, L\}\big\}$

- **Goal**: Learn a prediction function d_Θ

$$\widehat{\Theta} \in \underset{\Theta}{\operatorname{Argmin}}\ \mathrm{E}(\Theta) := \frac{1}{L} \sum_{\ell=1}^{I} f\big(z_\ell, \mathrm{d}_\Theta(u_\ell)\big)$$

- **Feedforward neural network model**:

$$\boxed{\mathrm{d}_\Theta(u_\ell) = \mathrm{prox}_{f^{[K]}}\big(W^{[K]} \ldots \mathrm{prox}^{[1]}_{f^{[1]}}(W^{[1]} u_\ell + b^{[1]}) \ldots + b^{[K]}\big)}$$

where for every $k \in \{1, \ldots, K\}$,

- $W^{[k]} \colon \mathcal{H}_{k-1} \to \mathcal{H}_k$ denotes a bounded linear operator,
- $b^{[k]} \in \mathcal{H}_k$ is a bias vector,
- $\eta^{[k]}$ is the nonlinear activation function.

Feedforward neural network model

- **Database**: $\mathcal{S} = \big\{(\overline{u}_\ell, z_\ell) \in \mathcal{H}_0 \times \mathcal{H}_K \;\big|\; \ell \in \{1,\ldots,L\}\big\}$

- **Goal**: Learn a prediction function d_Θ

$$\widehat{\Theta} \in \underset{\Theta}{\operatorname{Argmin}}\ \mathrm{E}(\Theta) := \frac{1}{L}\sum_{\ell=1}^{l} f\big(z_\ell, \mathrm{d}_\Theta(u_\ell)\big)$$

- **Feedforward neural network model**:

$$\boxed{\mathrm{d}_\Theta(u_\ell) = \mathrm{prox}_{f^{[K]}}\big(W^{[K]} \ldots \mathrm{prox}^{[1]}_{f^{[1]}}(W^{[1]}u_\ell + b^{[1]}) \ldots + b^{[K]}\big)}$$

where for every $k \in \{1,\ldots,K\}$,

  - $W^{[k]} \colon \mathcal{H}_{k-1} \to \mathcal{H}_k$ denotes a bounded linear operator,
  - $b^{[k]} \in \mathcal{H}_k$ is a bias vector,
  - $f^{[k]} \in \Gamma_0(\mathcal{H}_k)$.

Feedforward neural network model

- **Database**: $\mathcal{S} = \big\{(\overline{u}_\ell, z_\ell) \in \mathcal{H}_0 \times \mathcal{H}_K \;\big|\; \ell \in \{1,\ldots,L\}\big\}$

- **Goal**: Learn a prediction function d_Θ

$$\widehat{\Theta} \in \underset{\Theta}{\mathrm{Argmin}}\ \mathrm{E}(\Theta) := \frac{1}{L}\sum_{\ell=1}^{l} f\big(z_\ell, \mathrm{d}_\Theta(u_\ell)\big)$$

- **Feedforward neural network model**:

$$\boxed{\mathrm{d}_\Theta(u_\ell) = \mathrm{prox}_{f^{[K]}}\big(W^{[K]} \ldots \mathrm{prox}^{[1]}_{f^{[1]}}(W^{[1]}u_\ell + b^{[1]}) \ldots + b^{[K]}\big)}$$

where for every $k \in \{1,\ldots,K\}$,
  - $W^{[k]} \colon \mathcal{H}_{k-1} \to \mathcal{H}_k$ denotes a bounded linear operator,
  - $b^{[k]} \in \mathcal{H}_k$ is a bias vector,
  - $f^{[k]} \in \Gamma_0(\mathcal{H}_k)$.

$\to$ This model allows to derive tight Lipschitz bounds for feedforward neural networks in order to evaluate their stability.

More details in P. L. Combettes and J.-C. Pesquet, Deep neural network structures solving variational inequalities, Set-Valued and Variat. Anal., vol. 28, pp. 491–518, 2020. **[PDF]**

Unfolded Forward-Backward

- Reminder: One iteration of Forward-Backward to solve

$$\underset{x}{\text{minimize}}\ f(x) + g(x)$$

is, for some $\gamma > 0$, $\boxed{\mathrm{x}_{k+1} = \mathrm{prox}_{\gamma g}(x_k - \gamma \nabla f(x_k))}$

- Specific case: Considering the specific minimization problem

$$\underset{x}{\text{minimize}} \frac{1}{2}\|Ax - z\|_2^2 + \lambda\|x\|_1$$

the iteration are

$$\begin{aligned} x_{k+1} &= \mathrm{prox}_{\gamma\lambda\|\cdot\|_1}(x_k - \gamma A^* A x_k + \gamma A^* z) \\ &= \mathrm{prox}_{\gamma\lambda\|\cdot\|_1}((\mathrm{I} - \gamma A^* A) x_k + \gamma A^* z) \end{aligned}$$

which can be equivalently written

$$x_{k+1} = \eta^{[k]}(W^{[k]} x_k + b^{[k]}) \quad \text{where} \quad \begin{cases} W^{[k]} & = \mathrm{I} - \gamma A^* A \\ b^{[k]} & = \gamma A^* z \\ \eta^{[k]} & = \mathrm{prox}_{\gamma\lambda\|\cdot\|_1} \end{cases}$$

Unfolded Condat-Vũ splitting algorithm

- Reminder: One iteration of Condat-Vũ splitting to solve

$$\underset{x}{\text{minimize}}\ f(x) + g(Lx)$$

is, for some $\sigma, \tau > 0$,

$$x_{k+1} = x_k - \tau \nabla f(x_k) - \tau L^* y_k$$
$$y_{k+1} = \text{prox}_{\sigma g^*}\big(y_k + \sigma L(2x_{k+1} - x_k)\big)$$

- Specific case: Considering the specific minimization problem

$$\underset{x}{\text{minimize}} \frac{1}{2}\|Ax - z\|_2^2 + \lambda \|Lx\|_1$$

the iteration are

$$x_{k+1} = x_k - \tau A^*(Ax_k - \mathrm{z}) - \tau L^* y_k$$
$$y_{k+1} = \text{prox}_{\sigma g^*}\big(y_k + \sigma L(2x_{k+1} - x_k)\big)$$

Unfolded Condat-Vũ splitting algorithm

- Specific case: Considering the specific minimization problem

$$\underset{x}{\mathrm{minimize}} \frac{1}{2}\|Ax - z\|_2^2 + \lambda\|Lx\|_1$$

the iteration are

$$\begin{aligned} x_{k+1} &= x_k - \tau A^*(Ax_k - \mathrm{z}) - \tau L^* y_k \\ y_{k+1} &= \mathrm{prox}_{\sigma g^*}\big(y_k + \sigma L(2x_{k+1} - x_k)\big) \end{aligned}$$

or equivalently

$$\begin{aligned} x_{k+1} &= (\mathrm{Id} - \tau A^*A)x_k - \tau L^* y_k + \tau A^*\mathrm{z} \\ y_{k+1} &= \mathrm{prox}_{\sigma\|\cdot\|^*}\big(\sigma L(\mathrm{Id} - 2\tau A^*A)x_k + (\mathrm{Id} - 2\tau\sigma LL^*)y_k + 2\tau\sigma LA^*\mathrm{z}\big). \end{aligned}$$

which can be equivalently written

$$u_{k+1} = \eta^{[k]}(W^{[k]}u_k + b^{[k]}) \quad \text{where} \quad \begin{cases} u_k = (x_k, y_k) \\ D^{[k]} = \begin{pmatrix} \mathrm{Id} - \tau A^*A & -\tau L^* \\ \sigma L(\mathrm{Id} - 2\tau A^*A) & \mathrm{Id} - 2\tau\sigma LL^* \end{pmatrix} \\ b^{[k]} = \begin{pmatrix} \tau A^*\mathrm{z} \\ 2\tau\sigma LA^*\mathrm{z} \end{pmatrix} \\ \eta^{[k]} = \begin{pmatrix} \mathrm{Id} \\ \mathrm{prox}_{\sigma\|\cdot\|^*} \end{pmatrix} \end{cases}$$

Unfolded Condat-Vũ splitting algorithm

- Reminder: Predictor designed from proximal algorithm

$$\boxed{d_\Theta(u_\ell) = \eta^{[K]}\big(D^{[K]} \dots \eta^{[1]}(D^{[1]} u_\ell + b^{[1]}) \dots + b^{[K]}\big)}$$

- Learn the parameters Θ: The parameter to learn can be the algorithmix step-size γ_k but also $D^{[k]}$ or $b^{[k]}$.

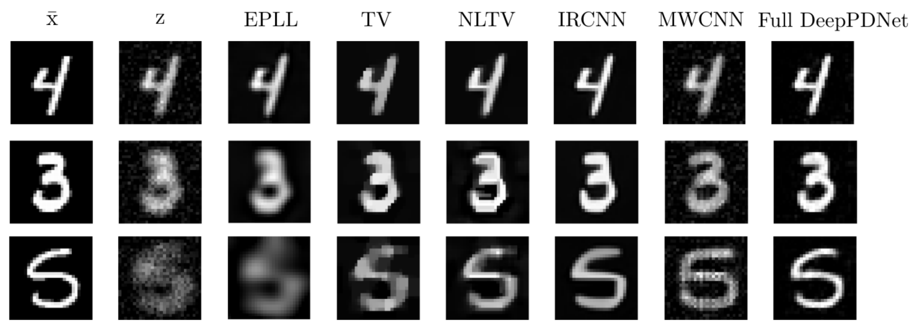
- Algorithmic strategy: based on stochastic gradient descent to estimate

$$\min_\Theta \frac{1}{L} \sum_{\ell=1}^{I} f\big(z_\ell, d_\Theta(u_\ell)\big)$$

$\rightarrow$ require the computation of the gradient of f_Θ (backpropagation strategy, automatic differentiation)

Unfolded Condat-Vũ splitting algorithm

- **Image restoration on MNIST database**: original $\bar{\mathrm{x}}$, degraded z, restored ones by EPLL, TV, NLTV, IRCNN, MWCC, and the proposed full DeepPDNet ($K = 6$).
  (first row) uniform 3×3 blur and Gaussian noise with $\alpha = 20$,
  (second row) uniform 5×5 blur and Gaussian noise with $\alpha = 20$,
  (third row) uniform 7×7 blur and Gaussian noise with $\alpha = 20$.



→ Results extracted from M. Jiu and N. Pustelnik, A deep primal-dual proximal network for image restoration, accepted to IEEE JSTSP, 2021. [**PDF**]

Unfolded Condat-Vũ splitting algorithm:

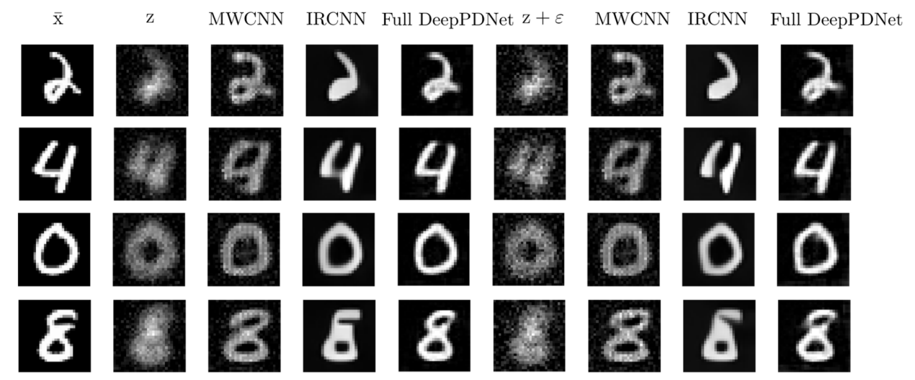
- **Image restoration on MNIST database**: Performance PSNR/SSIM for different configurations.

Data	Method	3×3 Blur			5×5 Blur	7×7 Blur
		$\alpha = 10$	$\alpha = 20$	$\alpha = 30$	$\alpha = 20$	$\alpha = 20$
		PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM
MNIST	EPLL [18]	24.02/0.8564	20.99/0.7628	19.05/0.6871	16.42/0.5629	13.97/0.3265
	TV [8]	25.07/0.8583	19.58/0.7004	18.86/0.6681	18.86/0.6681	16.31/0.5665
	NLTV [57]	25.49/0.8697	21.98/0.7738	20.73/0.7353	20.73/0.7353	16.79/0.6228
	MWCNN [58]	19.16/0.7219	18.53/0.6782	17.78/0.6499	15.83/0.5343	13.04/0.3175
	IRCNN [59]	**28.52**/0.8904	25.00/0.8193	22.63/0.7723	21.46/0.7698	18.29/0.6546
	Partial DeepPDNet	23.67/0.8366	22.03/0.7983	20.93/0.7750	17.96/0.6534	16.21/0.5505
	Full DeepPDNet	27.40/**0.9410**	**25.09/0.9254**	**23.61/0.9097**	**22.43/0.8738**	**20.43/0.8157**

→ Results extracted from M. Jiu and N. Pustelnik, A deep primal-dual proximal network for image restoration, accepted to IEEE JSTSP, 2021.

Unfolded Condat-Vũ splitting algorithm: robustness

- **Image restoration on MNIST database**: Robustness to additional noise.



→ Results extracted from M. Jiu and N. Pustelnik, A deep primal-dual proximal network for image restoration, accepted to IEEE JSTSP, 2021.