Bienvenue!

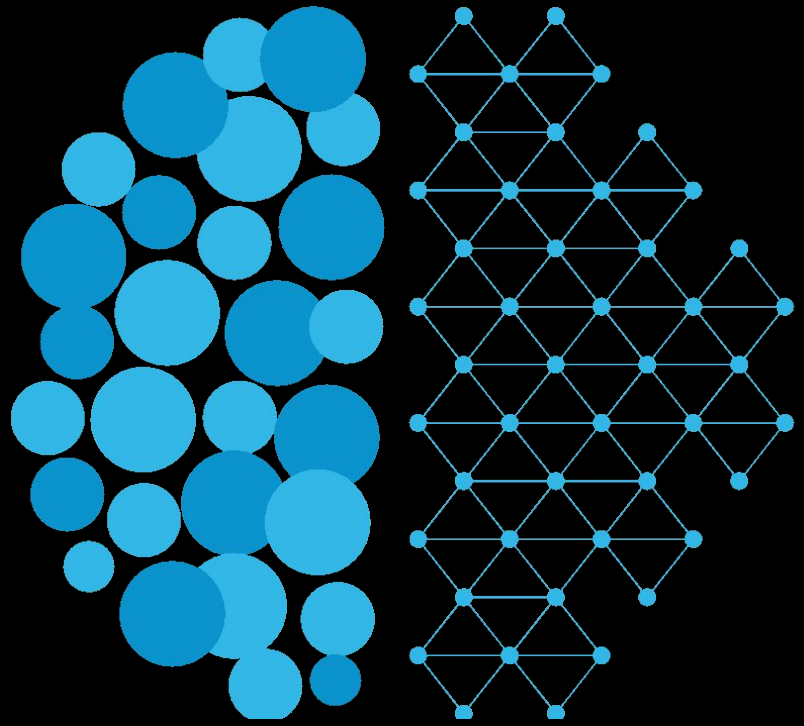
ÉCOLE D'HIVER FRANCOPHONE EN APPRENTISSAGE PROFOND

5 - 9 mars 2018

IVADO
HEC Montréal
Polytechnique Montréal
Université de Montréal

MILA

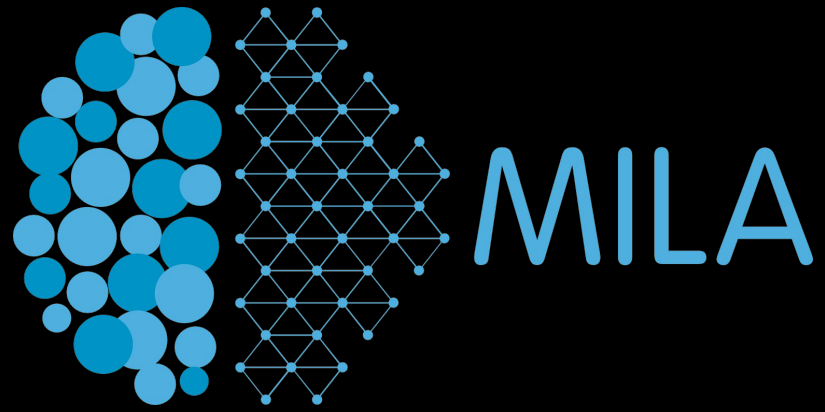
Institut
québécois
d'intelligence
artificielle



MILA

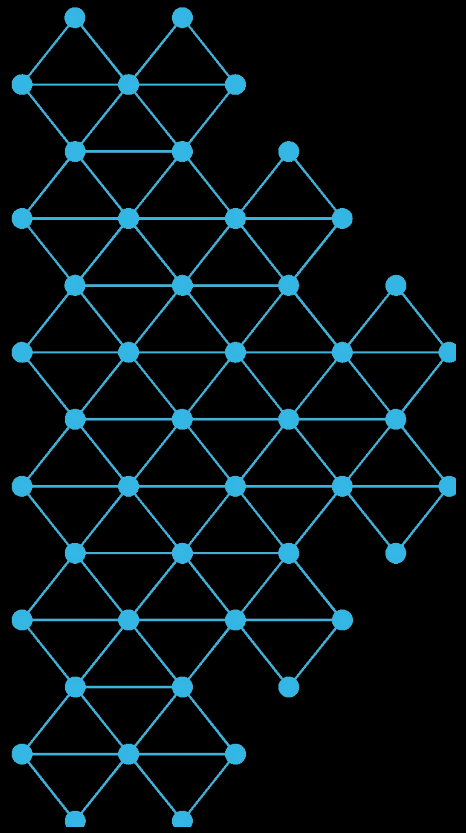
Université de Montréal

McGill



Entraînement: Graphe computationnel & backpropagation

Gaétan Marceau Caron
gaetan.marceau.caron@rd.mila.quebec



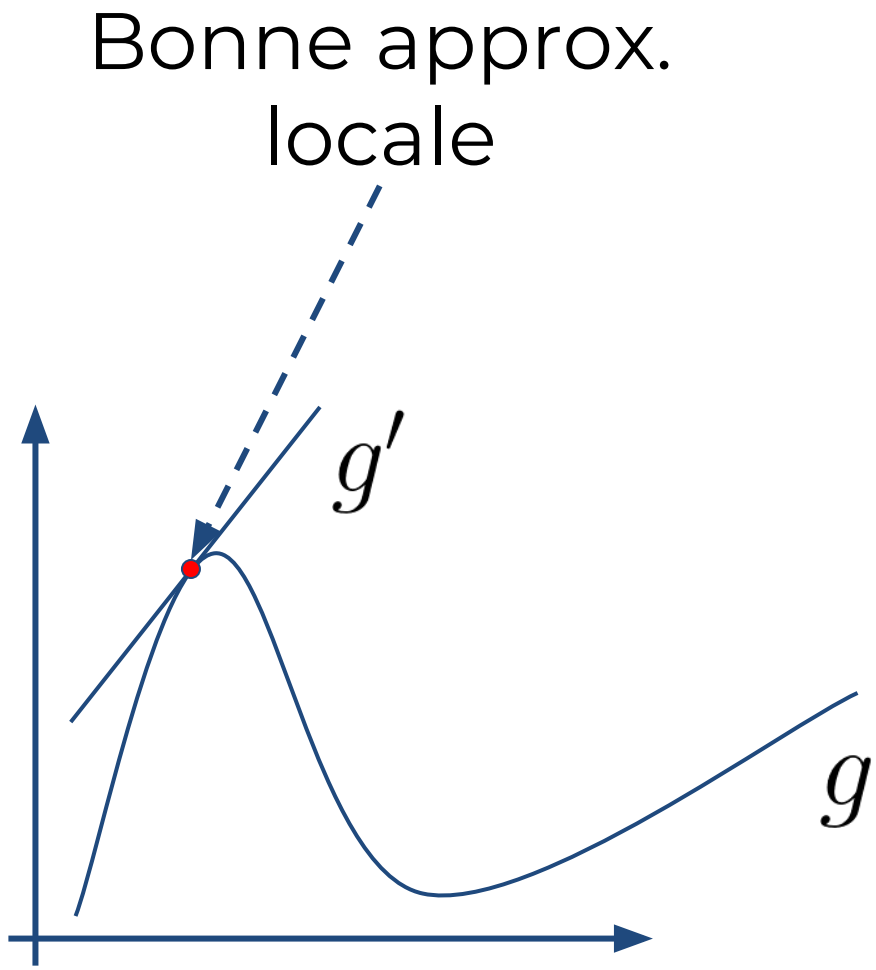
Rappel
Calcul différentiel

Rappel de Calcul 1

Dérivée d'une fonction

$$g : \mathbb{R} \to \mathbb{R}$$

$$g'(x) = \lim_{\Delta x \to 0} \frac{g(x+\Delta x) - g(x)}{\Delta x}$$



Exemples de dérivées

Linéaire	Sigmoid	ReLU
$g(x) = Wx + b$	$g(x) = \frac{1}{1+e^{-x}}$	$g(x) = \max(x, 0)$
$g'(x) = W$	$g'(x) = g(x)(1 - g(x))$	$g'(x) = \begin{cases} 1 \text{ si } x > 0 \\ 0 \end{cases}$

Rappel de Calcul 1

Fonction différentiable au point x

$$\lim_{\Delta x \to 0} \frac{g(x+\Delta x) - g(x) - g'(x)\Delta x}{\Delta x} = 0$$

Approx. linéaire

$$g(x + \Delta x) - g(x) \approx g'(x)\Delta x$$

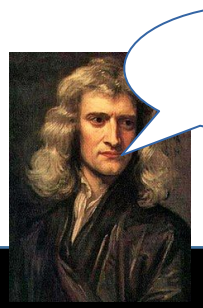
Rappel de Calcul 1

Composition de fonctions

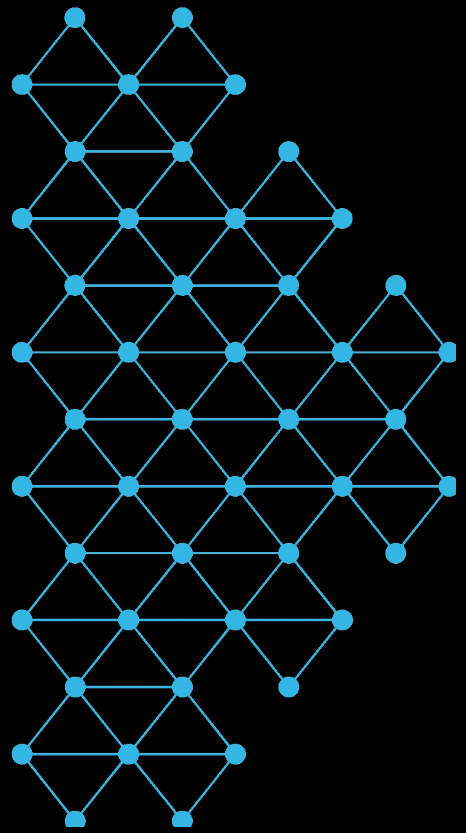
$$g \circ f(x) = g(f(x))$$

$$(g \circ f)'(x) = g'(f(x))f'(x)$$

Gottfried Wilhelm Leibniz
(1676)



MILA

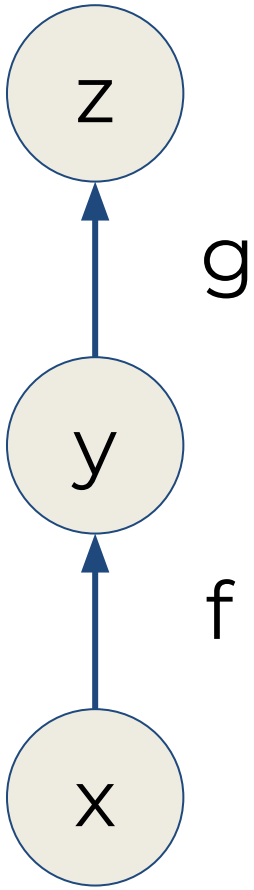


Graphe Computationnel

Graphe computationnel: Ex. 1

$$z = g(y)$$

$$y = f(x)$$



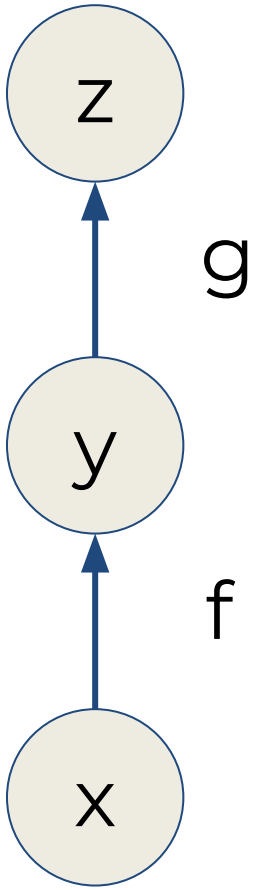

```
def g(y):
        ... let's do something
        return z
def f(x):
        ... let's do something else
        return y
float x = 3.1415
z = g(f(x))
```


MILA

Graphe computationnel: Ex. 1

$$z = g(y)$$

$$y = f(x)$$



$$\Delta z \approx g'(f(x))\Delta y$$

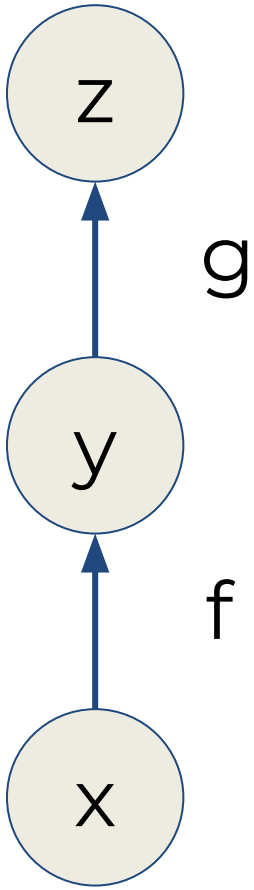
$$\Delta y \approx f'(x)\Delta x$$

$$\Delta x$$

Graphe computationnel: Ex. 1

$$z = g(y)$$

$$y = f(x)$$

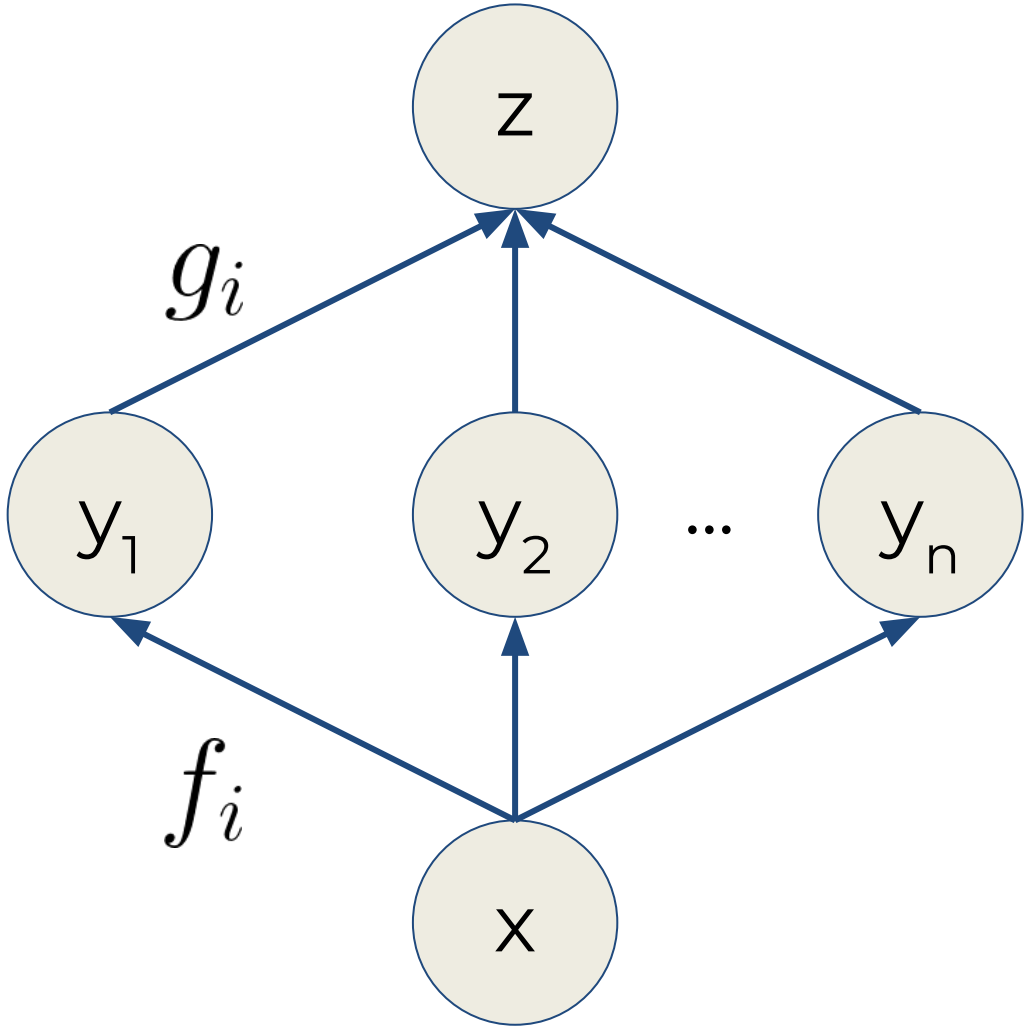


$$\Delta z \approx g'(f(x))f'(x)\Delta x$$

$$\Delta y \approx f'(x)\Delta x$$

$$\Delta x$$

Graphe computationnel: Ex. 2

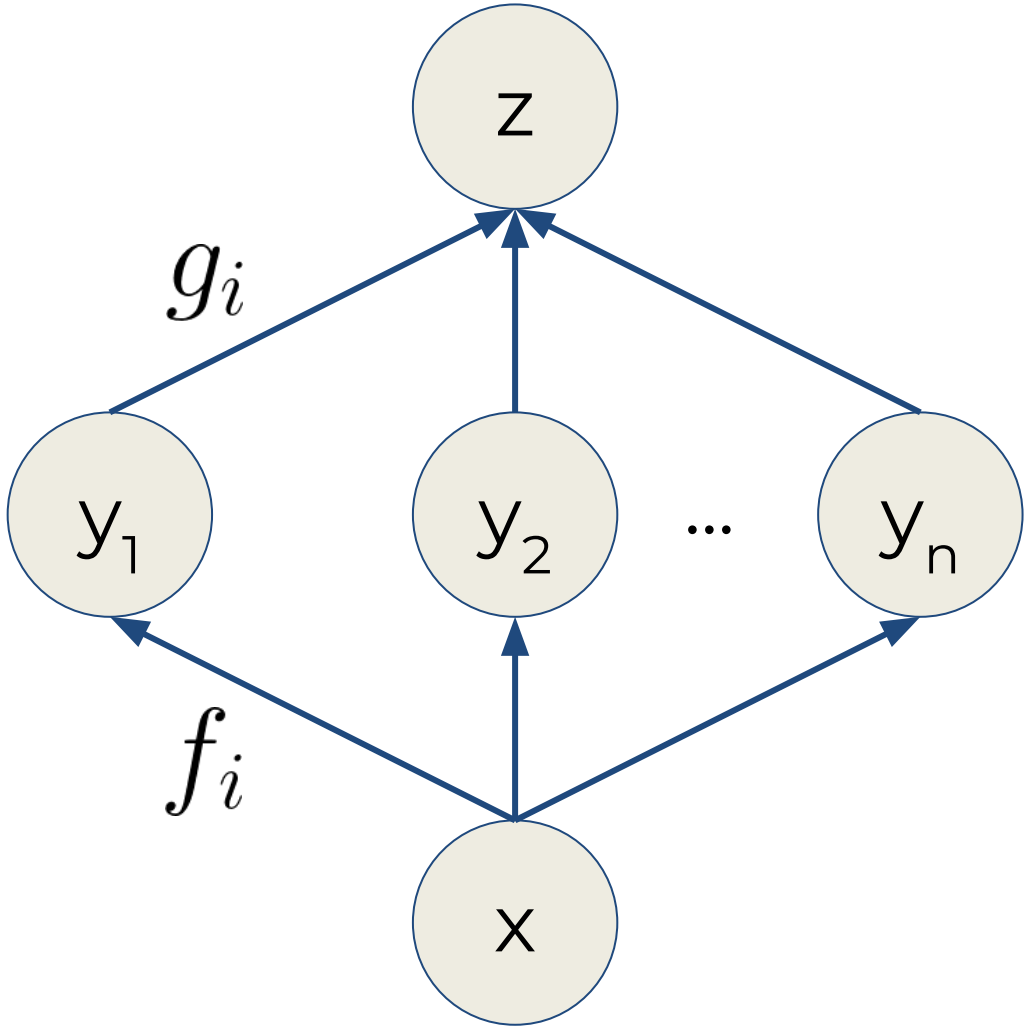


$$z = \sum_i g_i(y_i)$$

$$y_i = f_i(x)$$

MILA

Graphe computationnel: Ex. 2

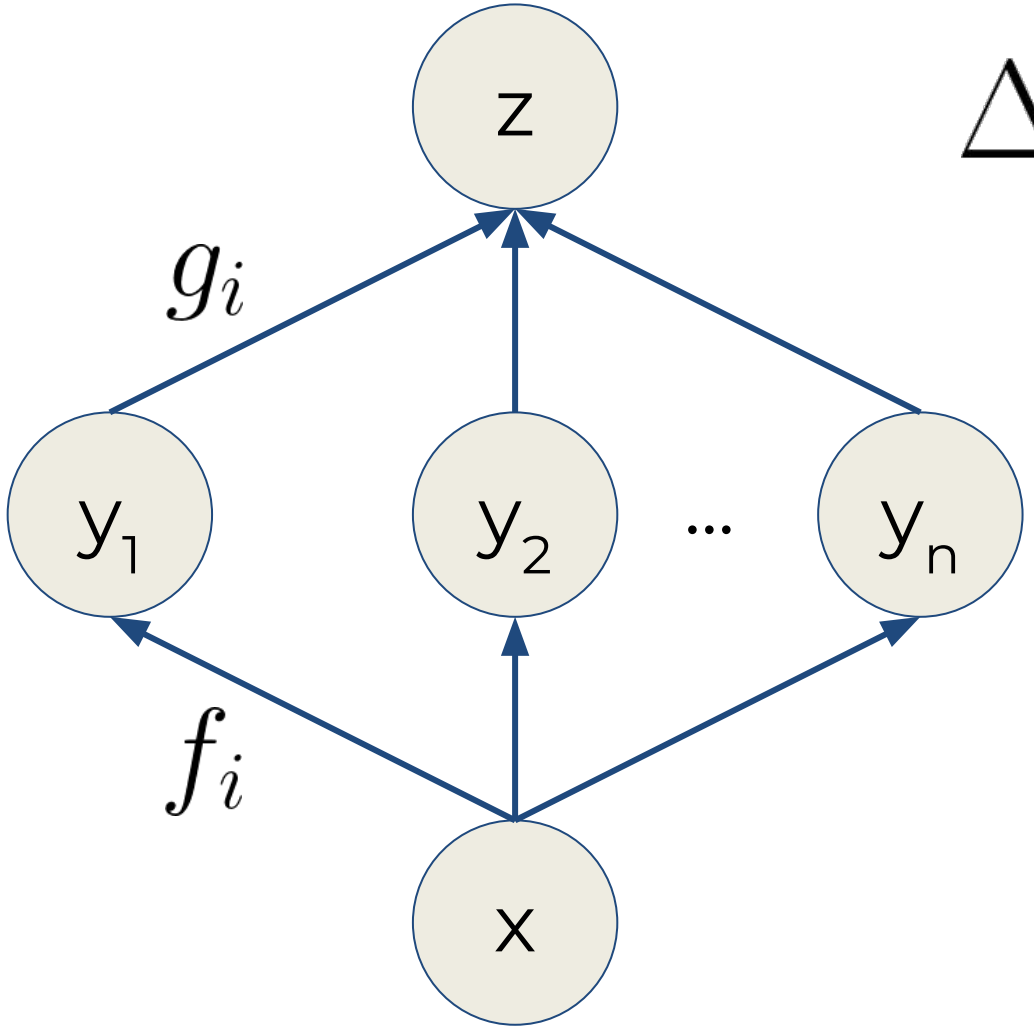


$$\Delta z \approx \sum_i g_i'(f_i(x))\Delta y_i$$

$$\Delta y_i \approx f_i'(x)\Delta x$$

$$\Delta x$$

Graphe computationnel: Ex. 2

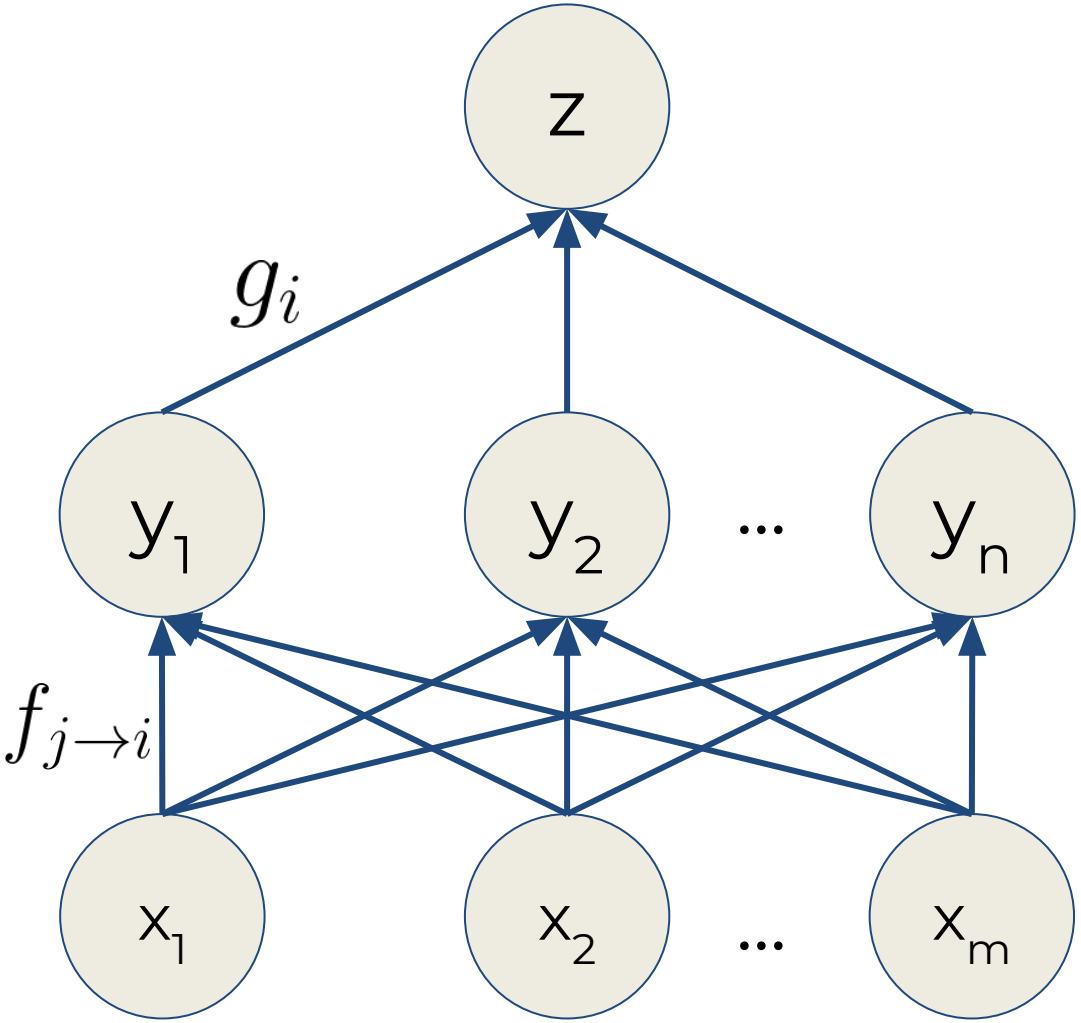


$$\Delta z \approx \sum_i g_i'(f_i(x)) f_i'(x) \Delta x$$

$$\Delta y_i \approx f_i'(x) \Delta x$$

$$\Delta x$$

Graphe computationnel: Ex. 3

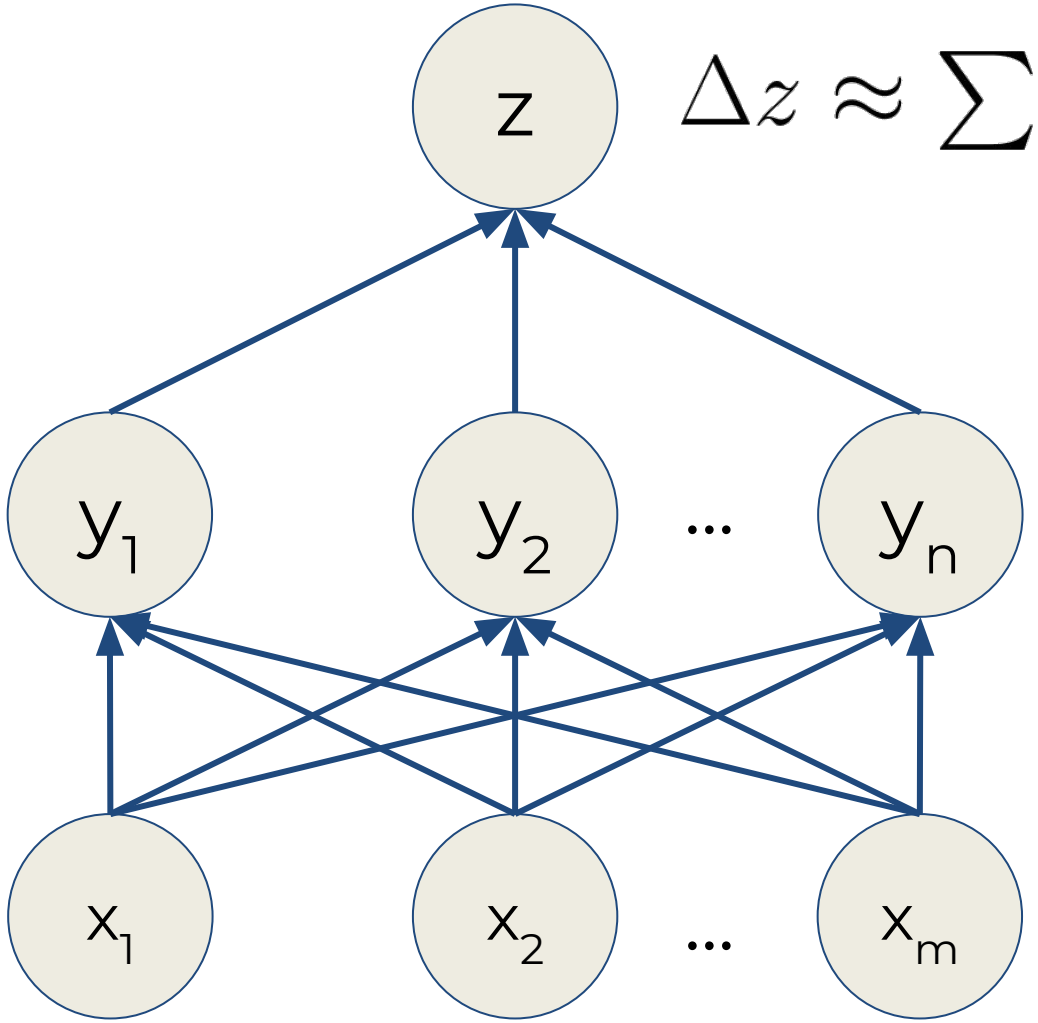


$$z = \sum_i g_i(y_i)$$

$$y_i = \sum_j f_{j\to i}(x_j)$$

$$\Delta x_1, \ldots \Delta x_m$$

Graphe computationnel: Ex. 3

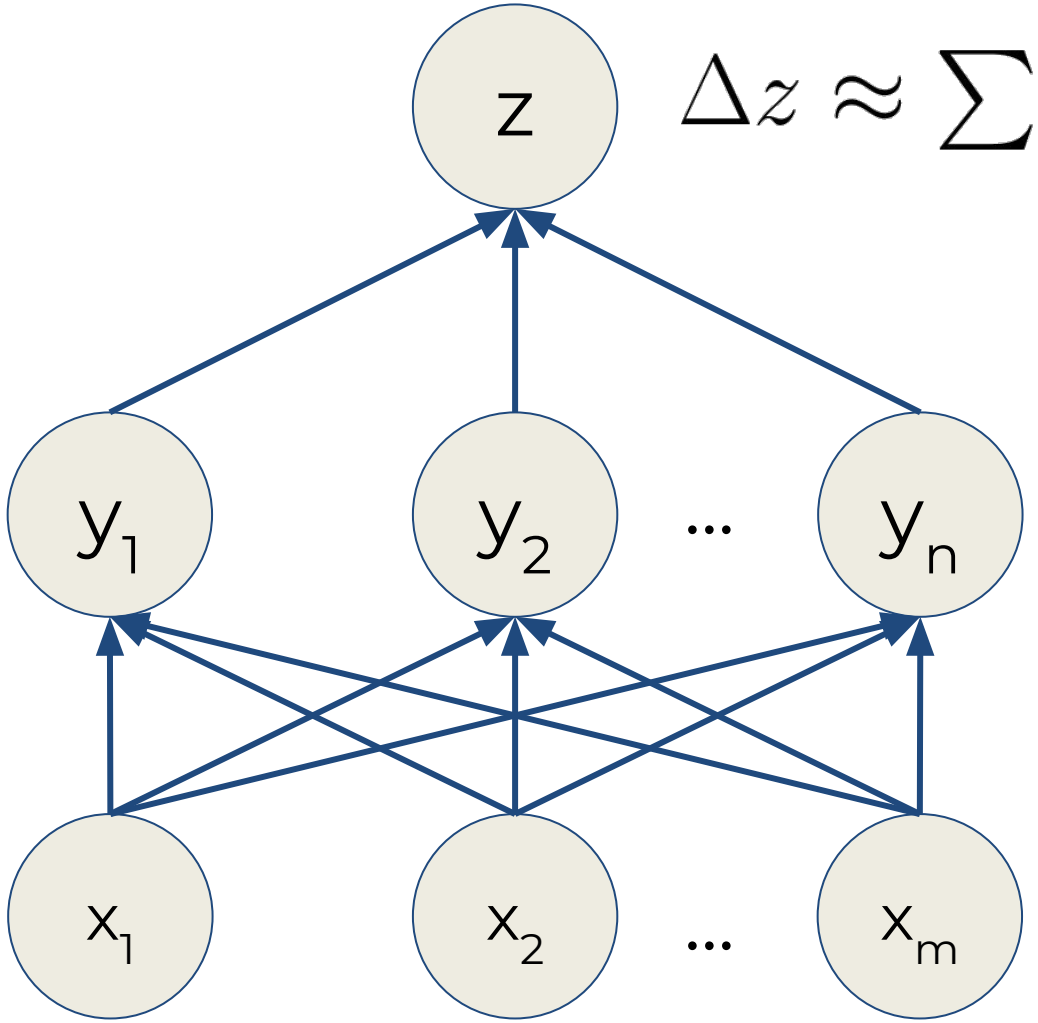


$$\Delta z \approx \sum_i g_i'(\sum_j f_{j \to i}(x_j))\Delta y_i$$

$$\Delta y_i \approx \sum_j f_{j \to i}'(x_j)\Delta x_j$$

$$\Delta x_1, \ldots \Delta x_m$$

Graphe computationnel: Ex. 3



$$\Delta z \approx \sum_i g_i'(\sum_j f_{j\to i}(x_j)) \sum_j f_{j\to i}'(x_j)\Delta x_j$$

$$\Delta y_i \approx \sum_j f_{j\to i}'(x_j)\Delta x_j$$

$$\Delta x_1, \ldots \Delta x_m$$

Graphe computationnel: Ex. 3

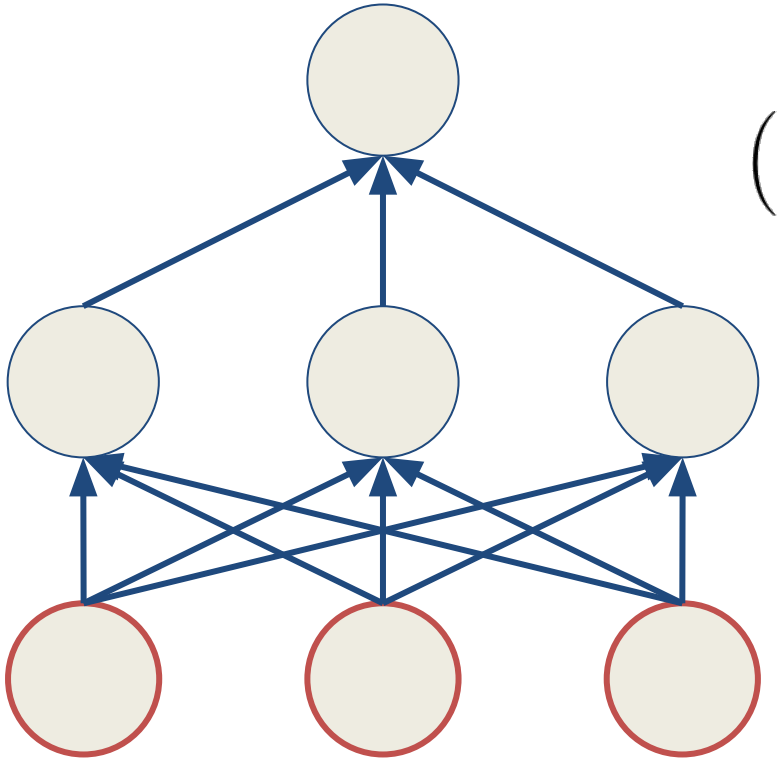
$$\Delta z = \sum_i g_i'(\sum_j f_{j\to i}(x_j)) \sum_j f_{j\to i}'(x_j)\Delta x_j$$

$$= \sum_j \sum_i g_i'(\sum_j f_{j\to i}(x_j)) f_{j\to i}'(x_j)\Delta x_j$$

$$= < \nabla z, \Delta x >$$

$$(\nabla z)_j = \sum_i g_i'(\sum_j f_{j\to i}(x_j)) f_{j\to i}'(x_j)$$

Backpropagation



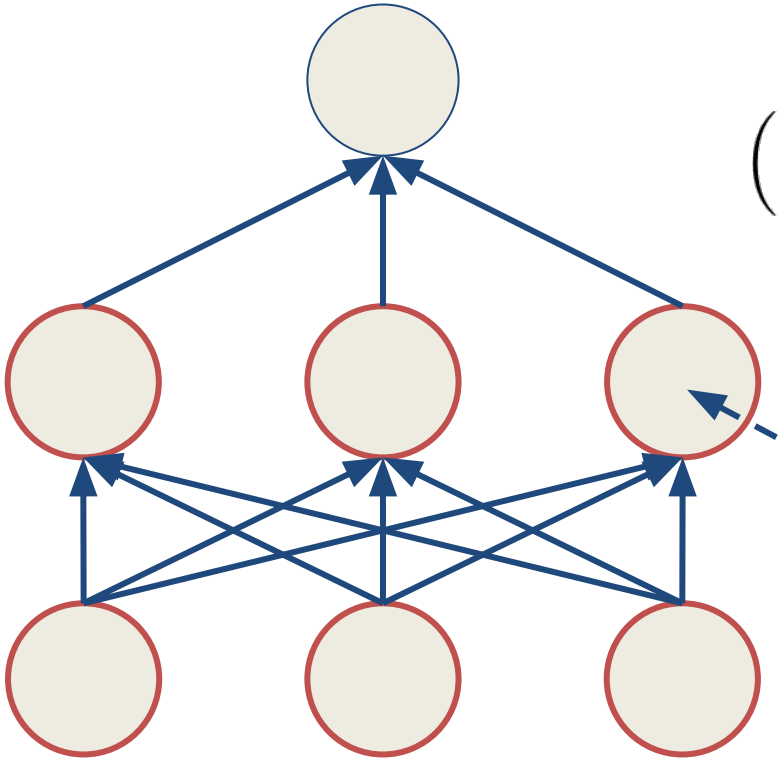
Approx. linéaire

$$(\nabla z)_j = \sum_i g_i'\left(\sum_j f_{j\to i}(x_j)\right) f_{j\to i}'(x_j)$$

Forward

MILA

Backpropagation



Approx. linéaire

$$(\nabla z)_j = \sum_i g_i'\left(\sum_j f_{j\to i}(x_j)\right) f_{j\to i}'(x_j)$$

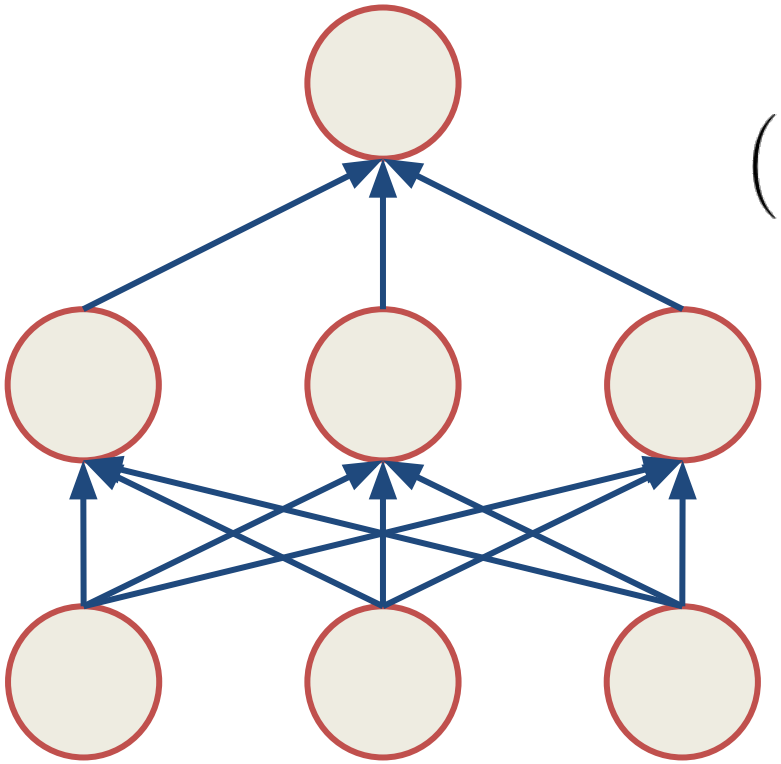
Forward

$$\sum_j f_{j\to i}(x_j)$$

Stocke le résultat intermédiaire

MILA

Backpropagation



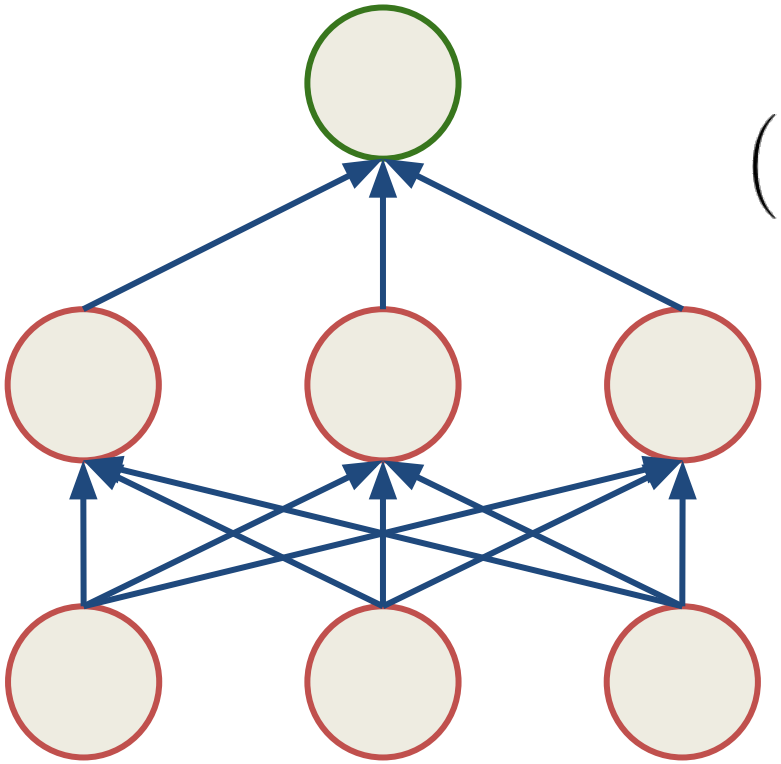
Approx. linéaire

$$(\nabla z)_j = \sum_i g_i'\left(\sum_j f_{j\to i}(x_j)\right) f_{j\to i}'(x_j)$$

Forward

MILA

Backpropagation



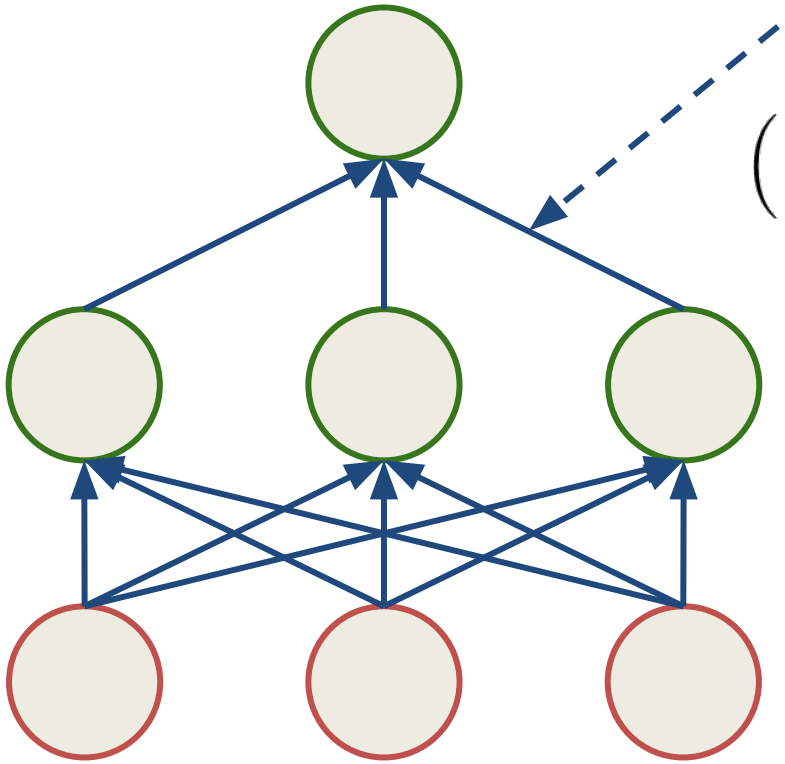
Approx. linéaire

$$(\nabla z)_j = \sum_i g_i'\left(\sum_j f_{j\to i}(x_j)\right) f_{j\to i}'(x_j)$$

Backpropagation

$g_i'(\sum_j f_{j\to i}(x_j))$

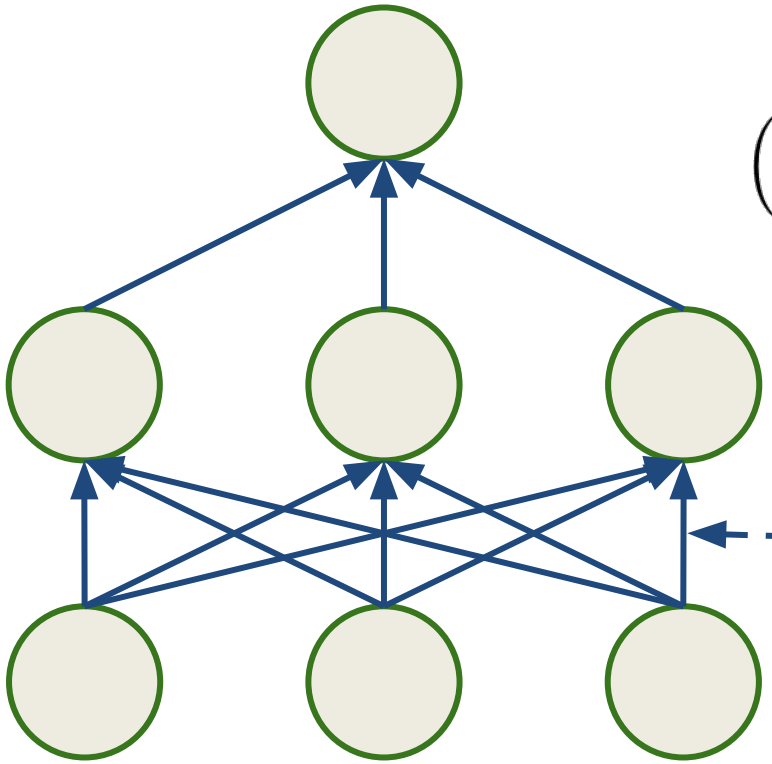
Stocke le résultat intermédiaire



Approx. linéaire

$$(\nabla z)_j = \sum_i \underbrace{g_i'(\sum_j f_{j\to i}(x_j))}_{\text{Backward}} f_{j\to i}'(x_j)$$

Backpropagation



Approx. linéaire

$$(\nabla z)_j = \sum_i g_i'\left(\sum_j f_{j\to i}(x_j)\right) f_{j\to i}'(x_j)$$

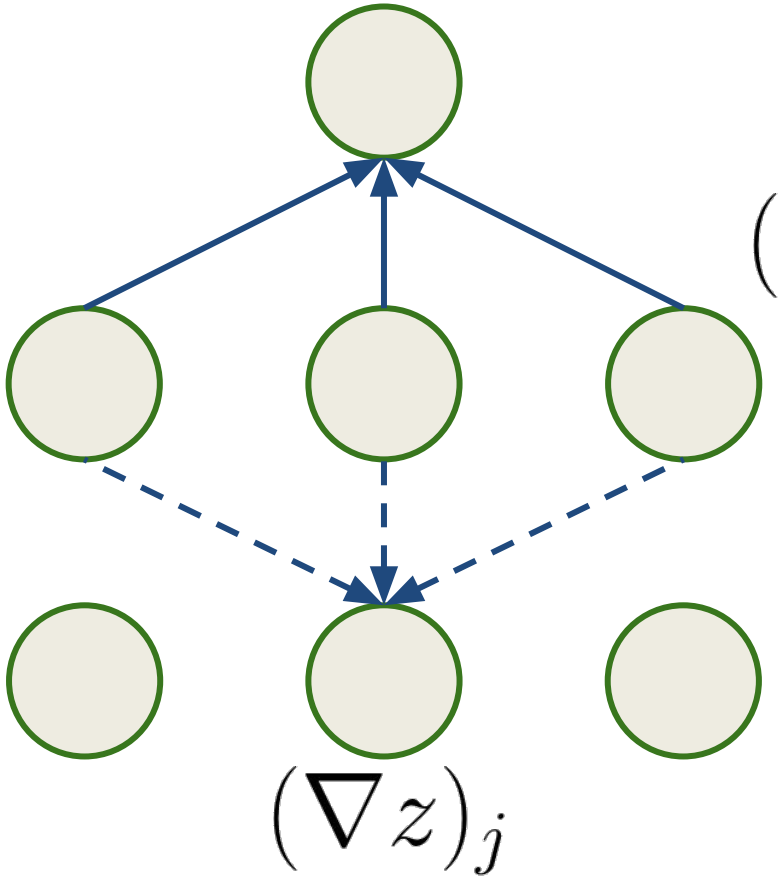
Backward

$$g_i'\left(\sum_j f_{j\to i}(x_j)\right) f_{j\to i}'(x_j)$$

Stocke le résultat intermédiaire

MILA

Backpropagation



Approx. linéaire

$$(\nabla z)_j = \overbrace{\sum_i g_i'\left(\sum_j f_{j\to i}(x_j)\right) f_{j\to i}'(x_j)}$$

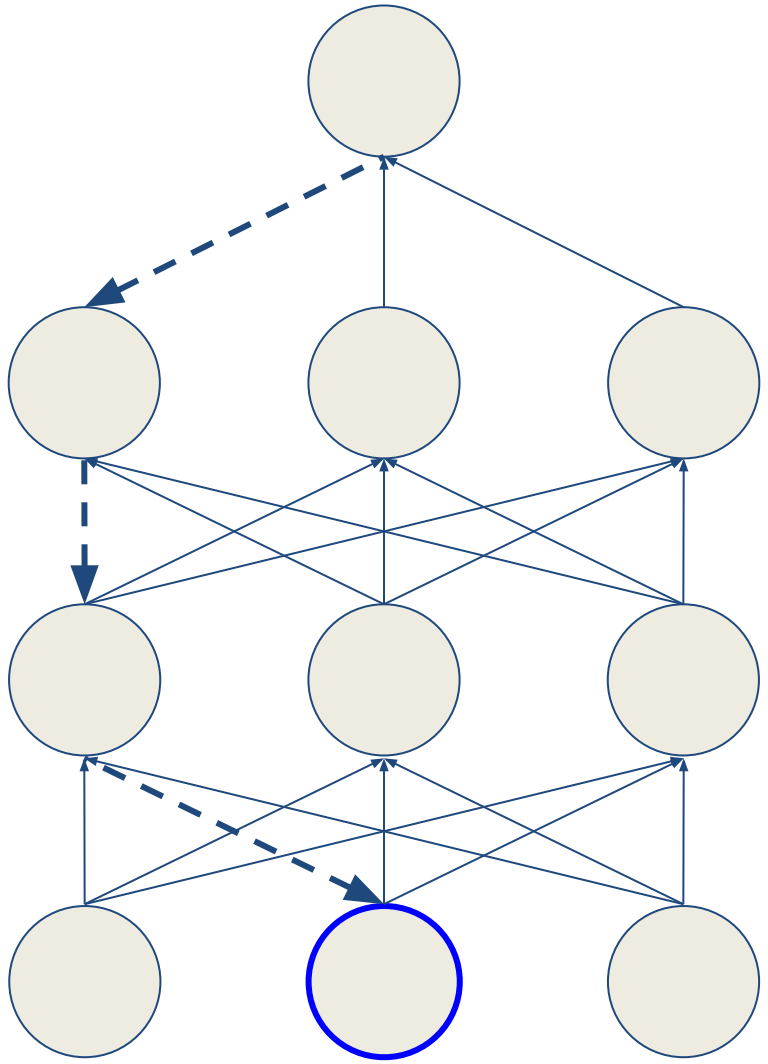
Backward

On inverse l'ordre de la propagation!

MILA

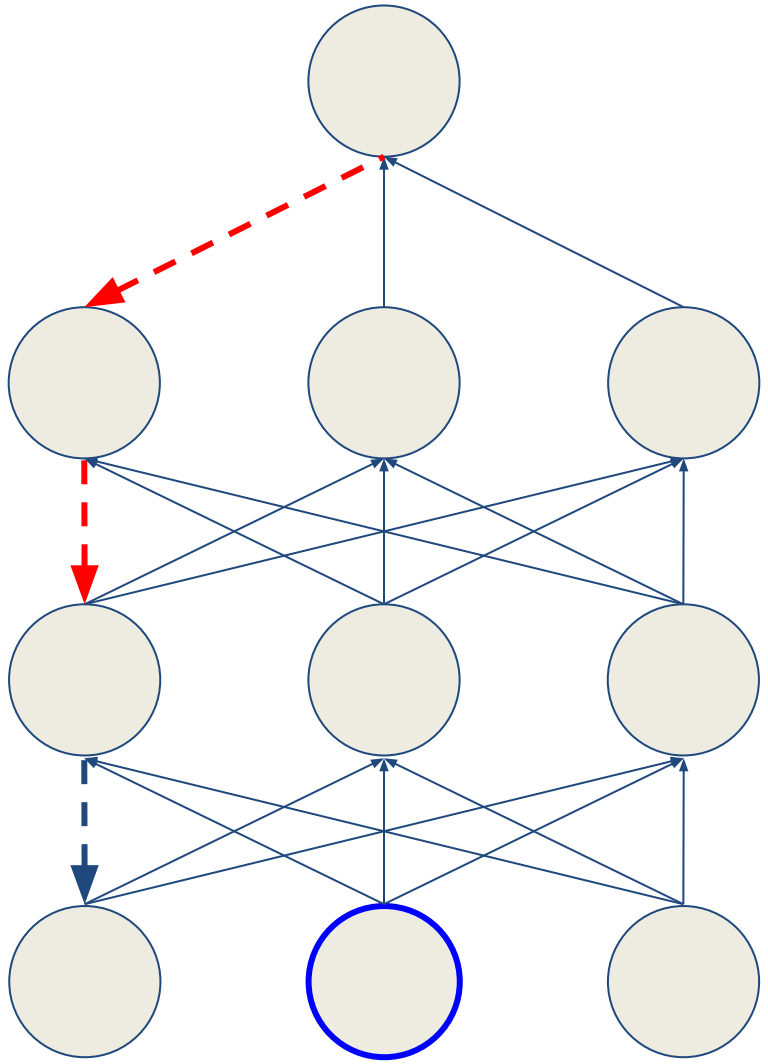
Backpropagation

Problème de
parcours de graphe



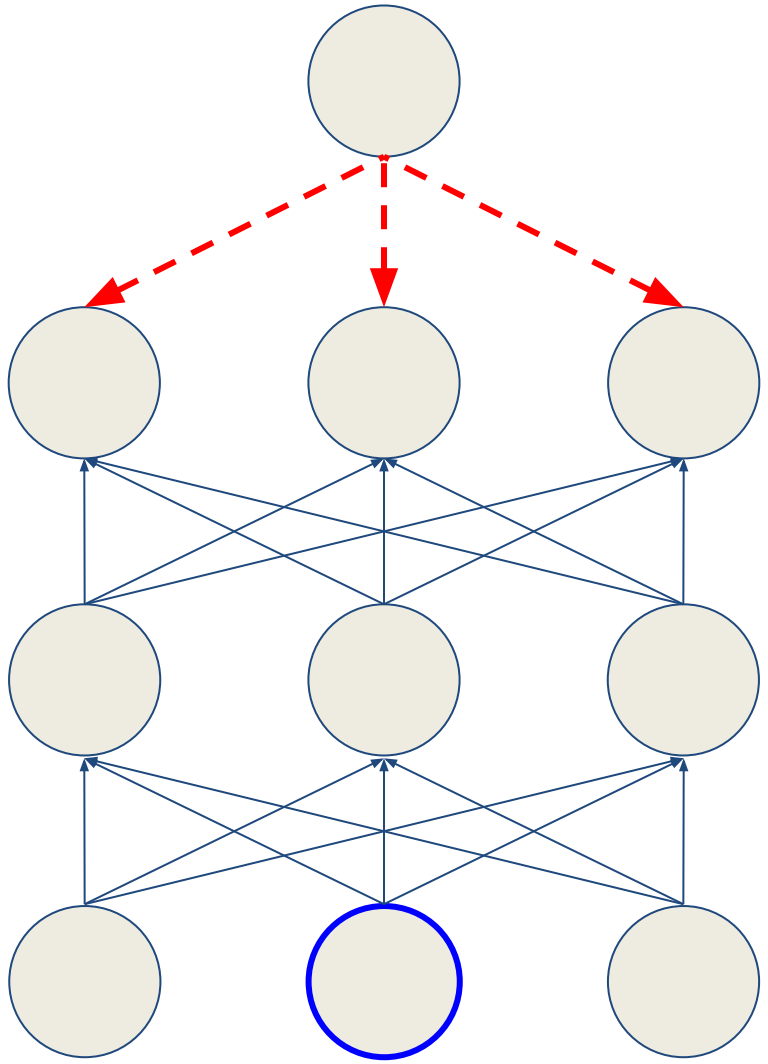
Backpropagation

Le produit sur ce chemin partiel a déjà été calculé!



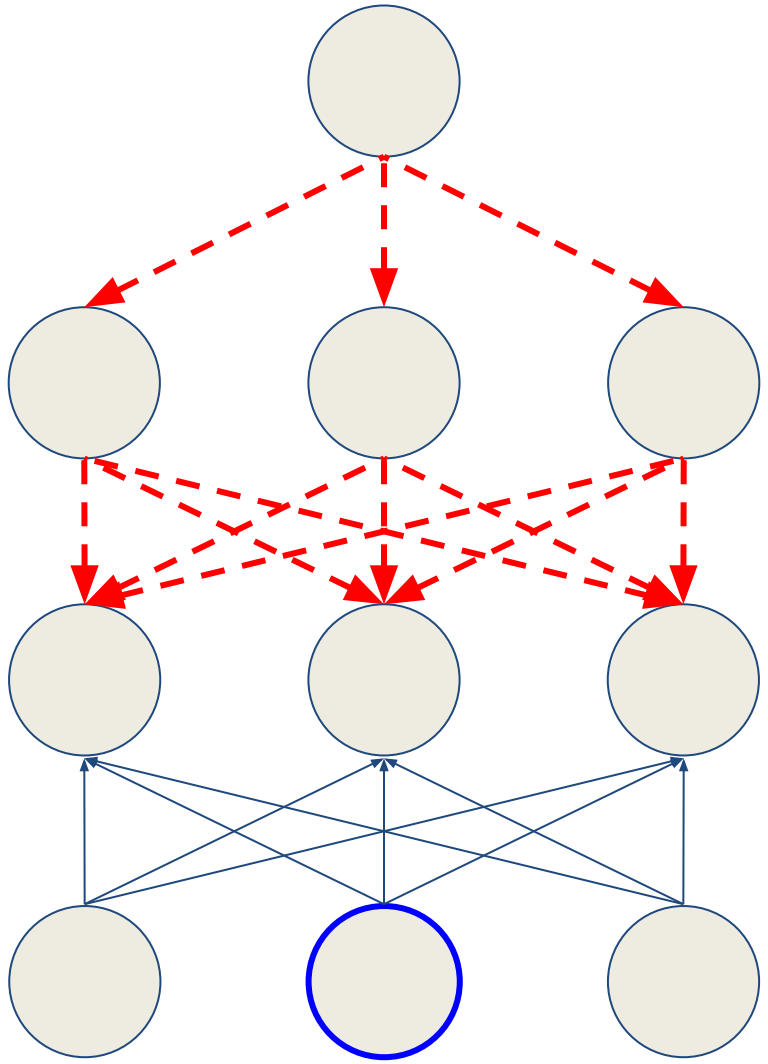
Backpropagation

Programmation dynamique



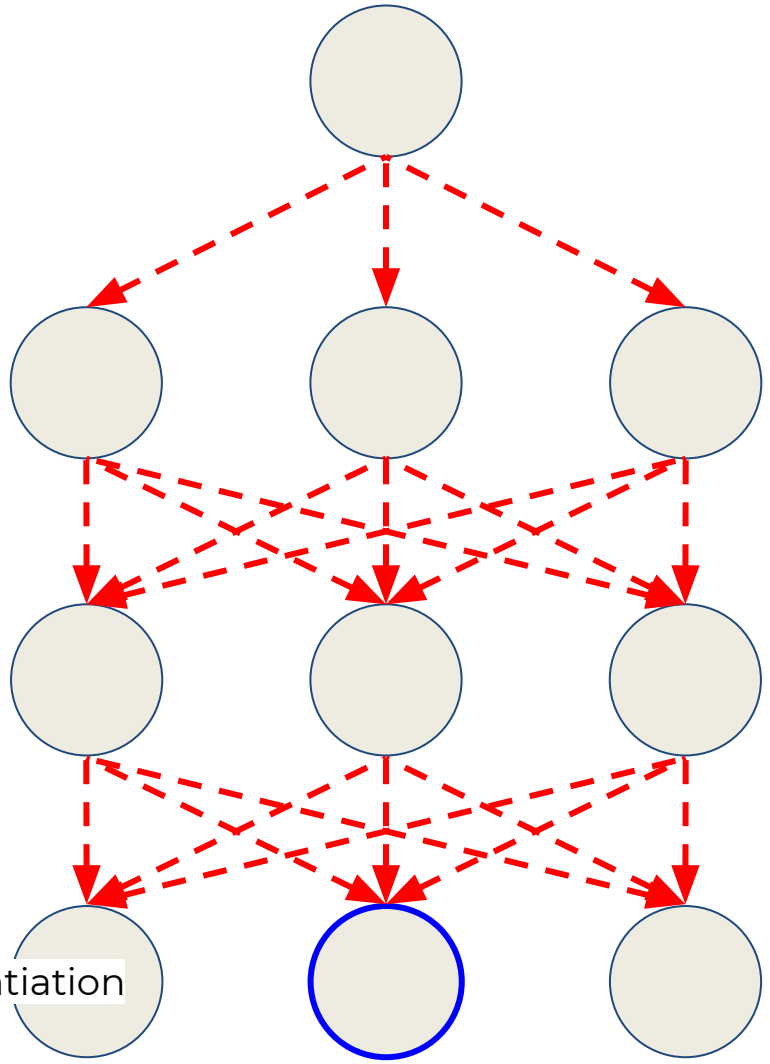
Backpropagation

Programmation dynamique

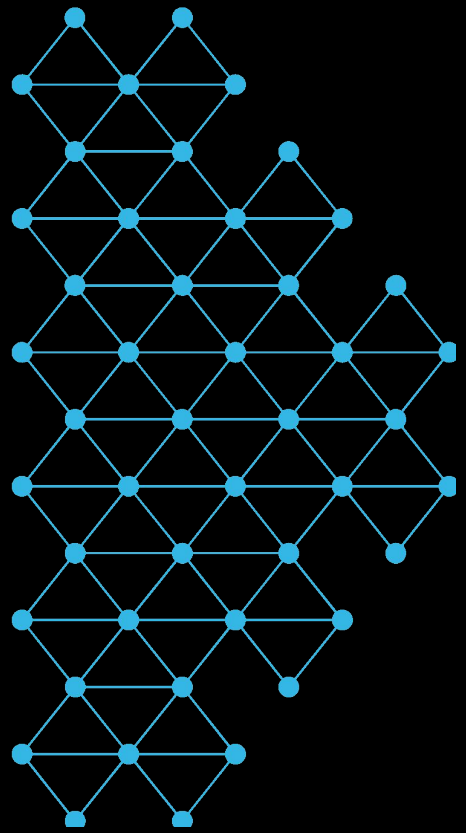


Backpropagation

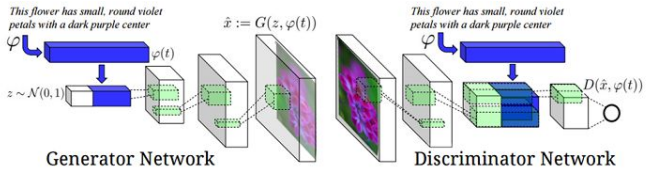
Programmation dynamique

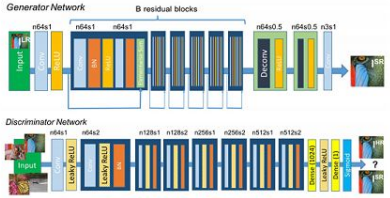


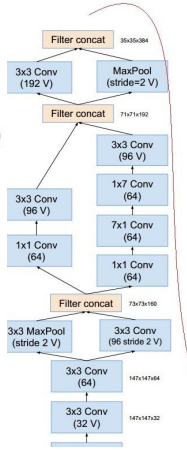
Seppo Linnainmaa (1970): automatic differentiation
Rumelhart et al., (1986): neural networks

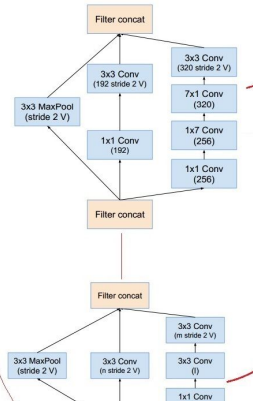


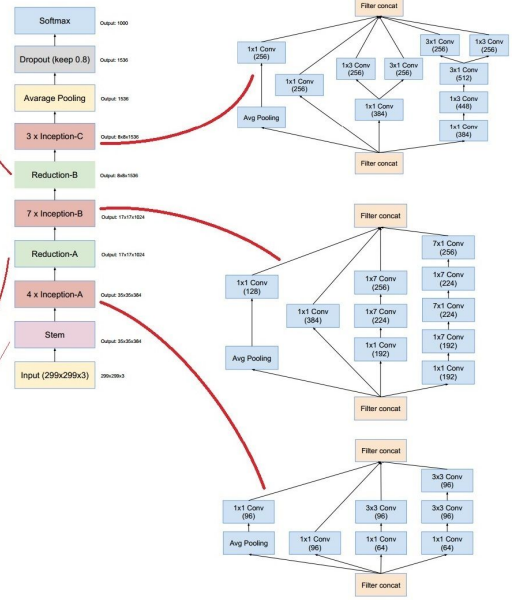
Retour en 2018...

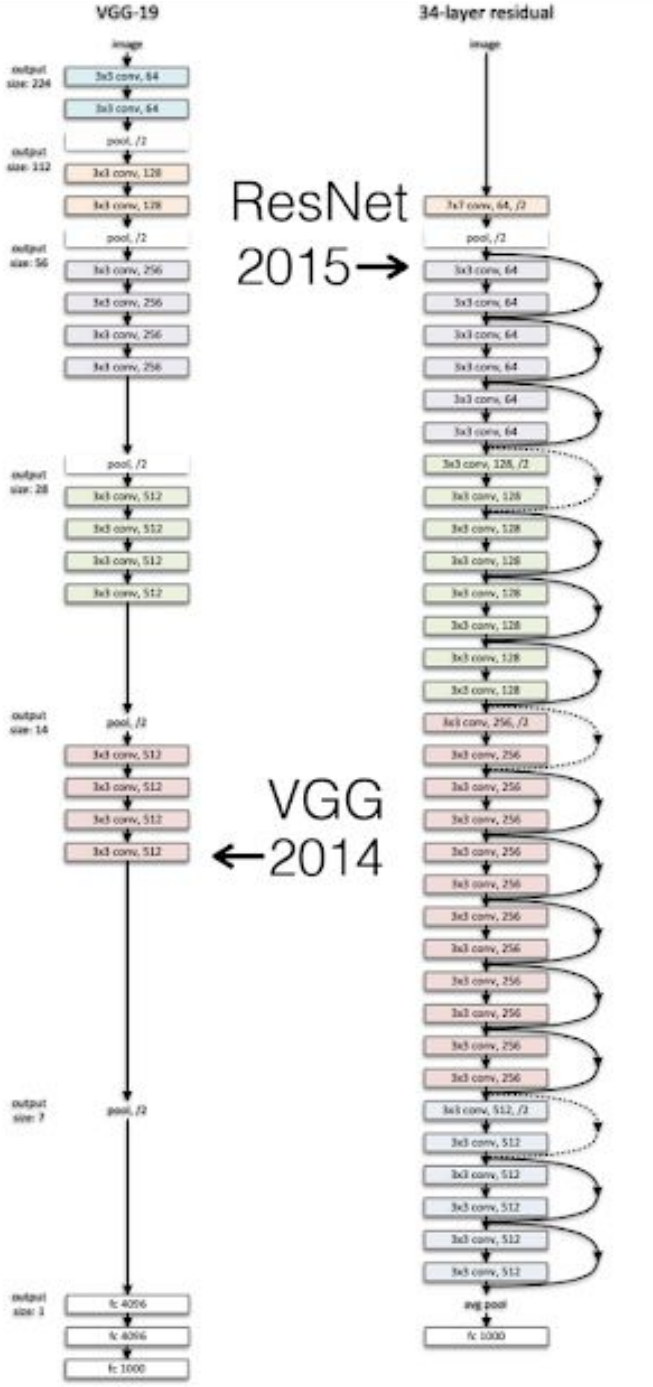


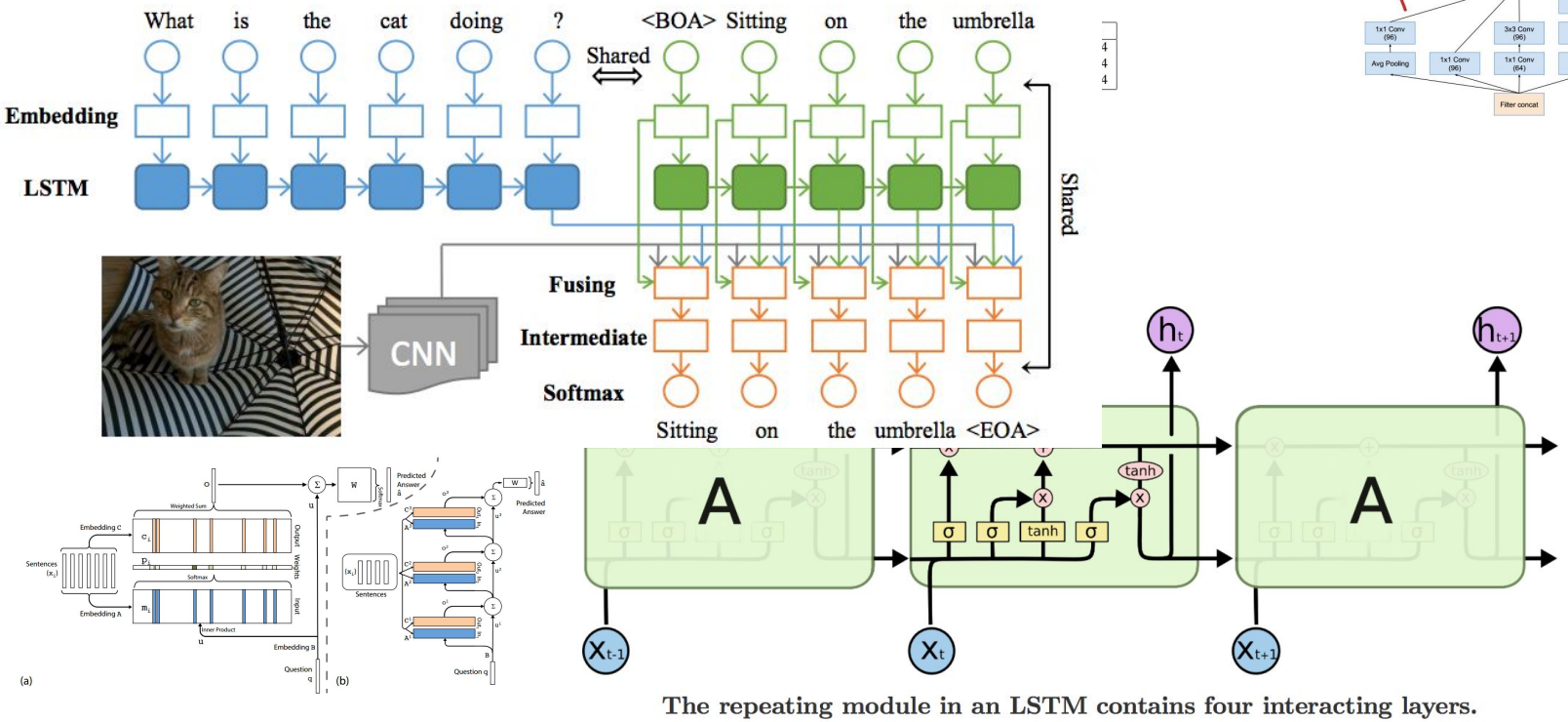




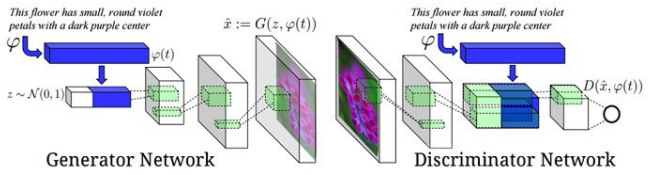


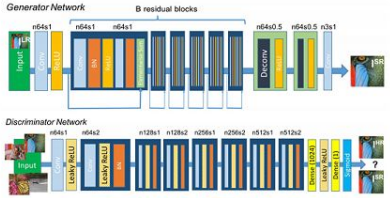


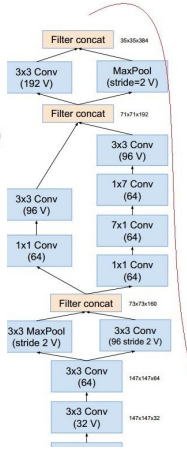


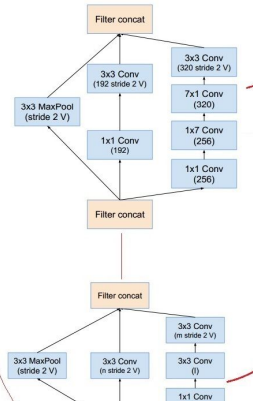


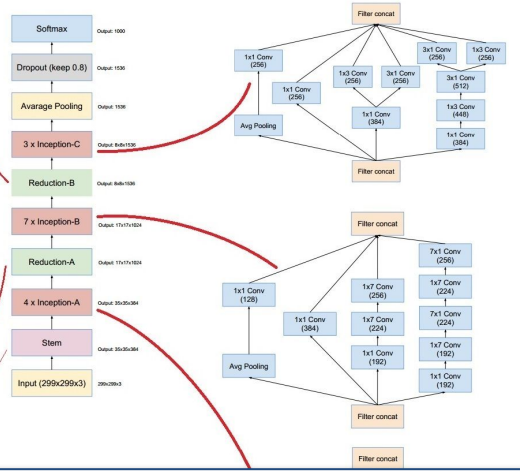
The repeating module in an LSTM contains four interacting layers.

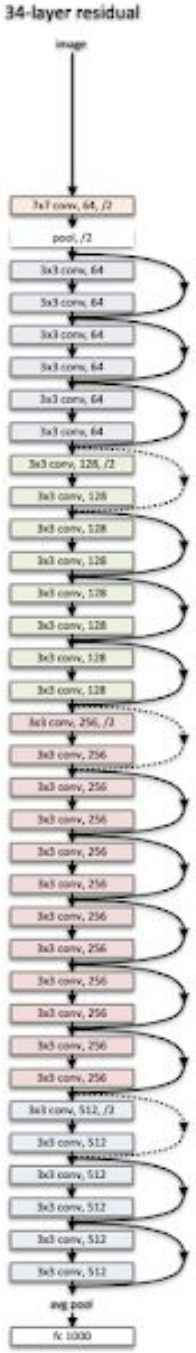






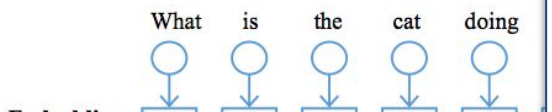


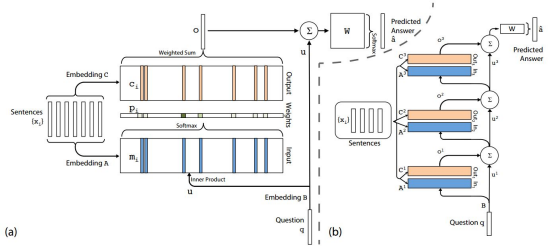


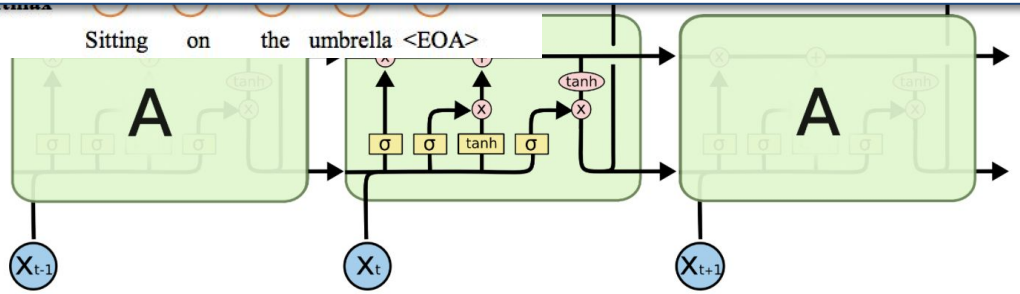


ResNet 2015→

VGG -2014







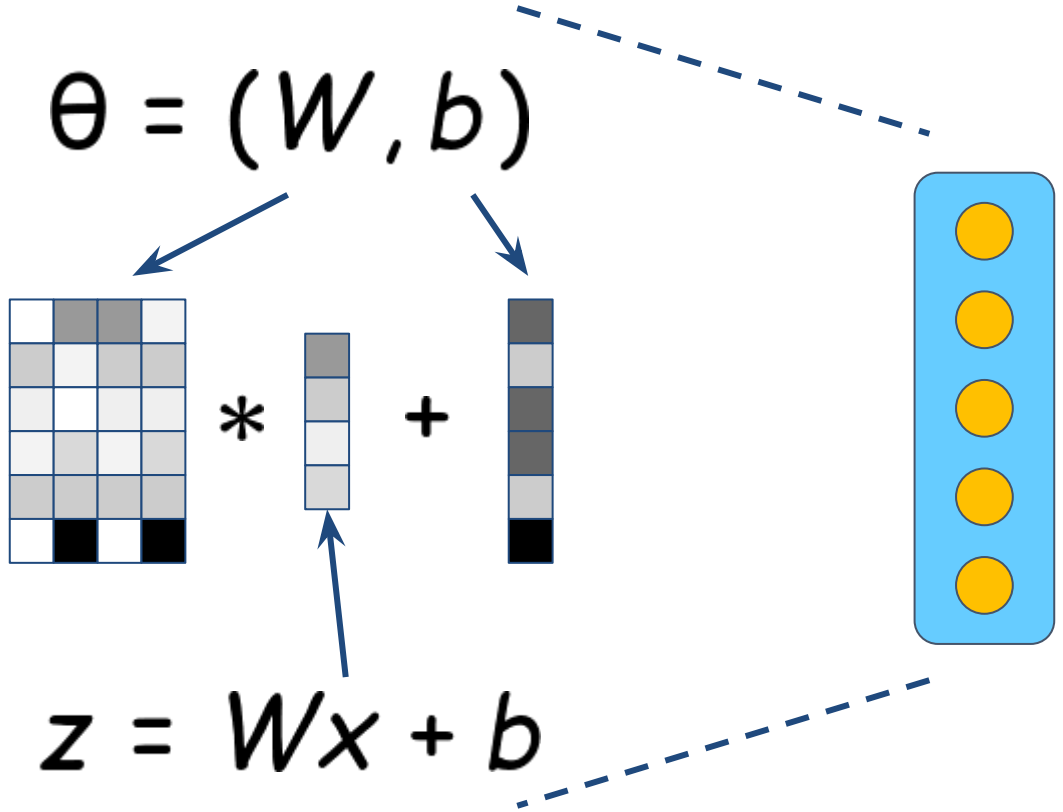
The repeating module in an LSTM contains four interacting layers.

Programmation différentiable

Définition d'un module

Paramètres

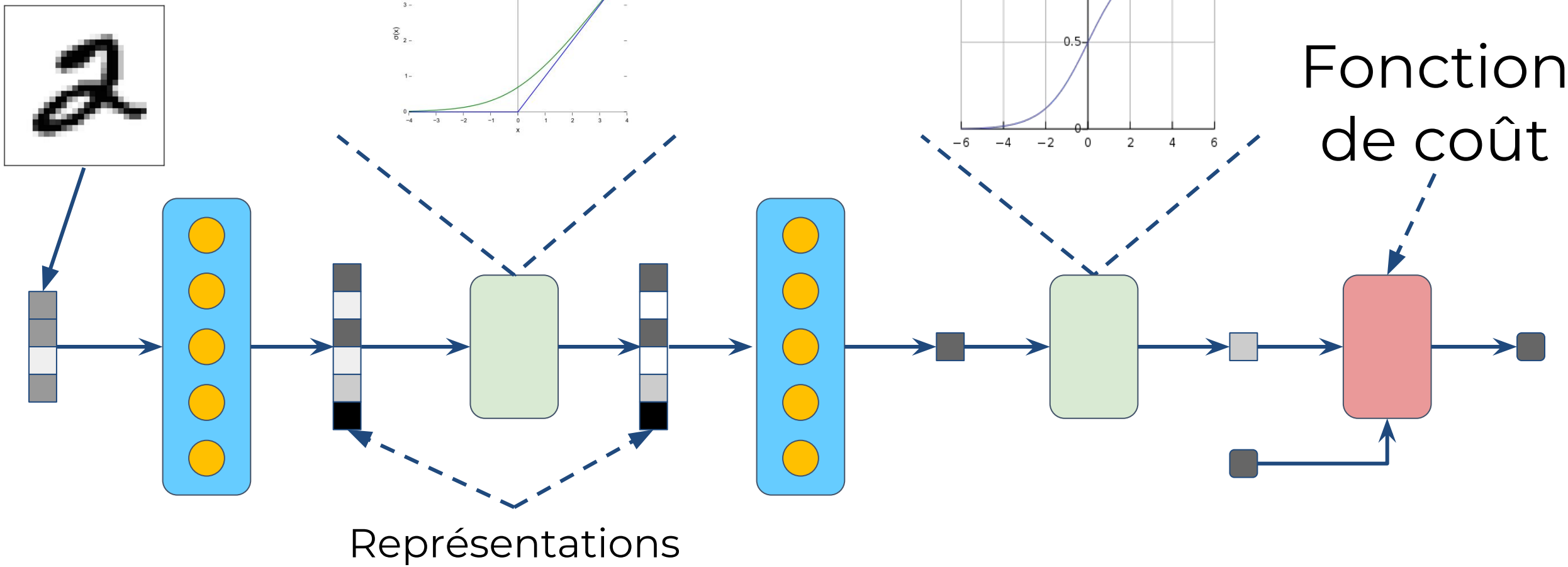
$$\theta = (W, b)$$



Transformations

$$z = Wx + b$$

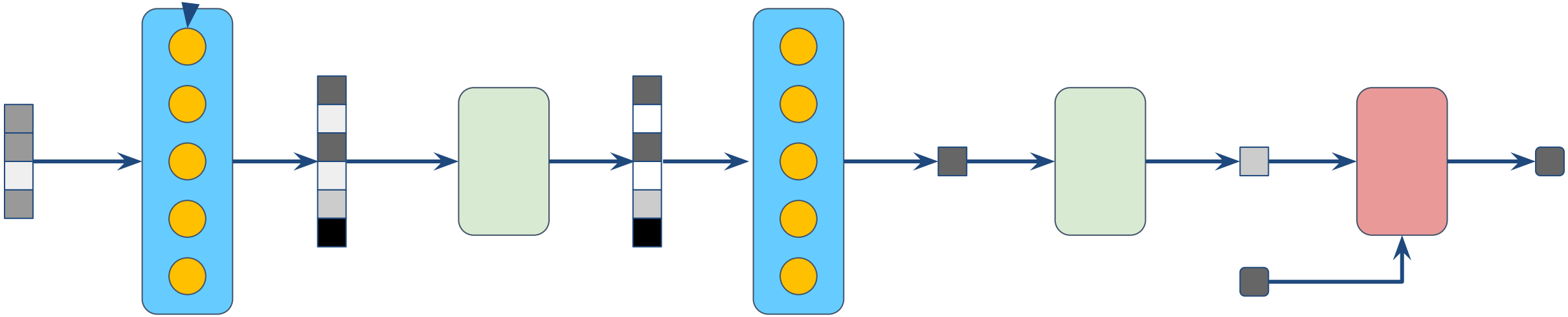
Exemple



MILA

Impact d’un paramètre

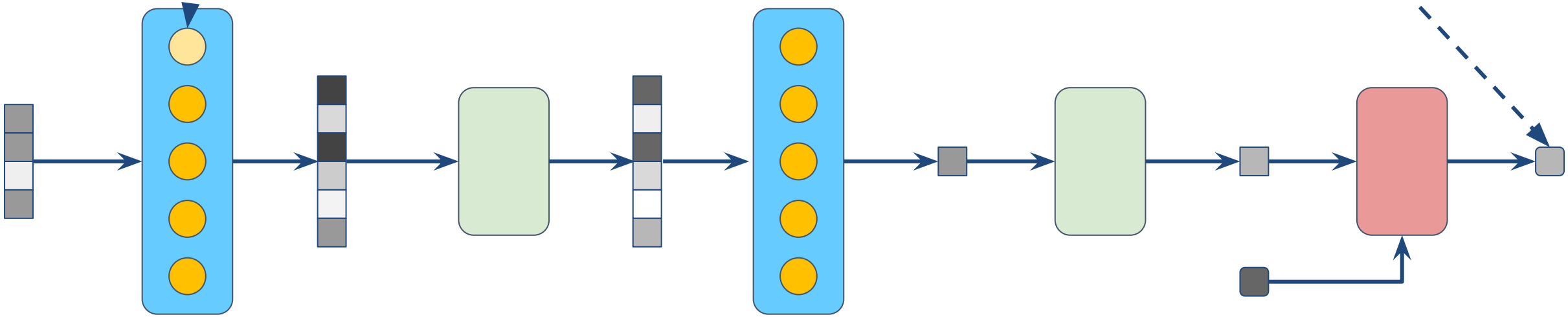
Perturbation d’un paramètre



Impact d’un paramètre

Perturbation d’un paramètre

Impact positif ou négatif sur le coût?



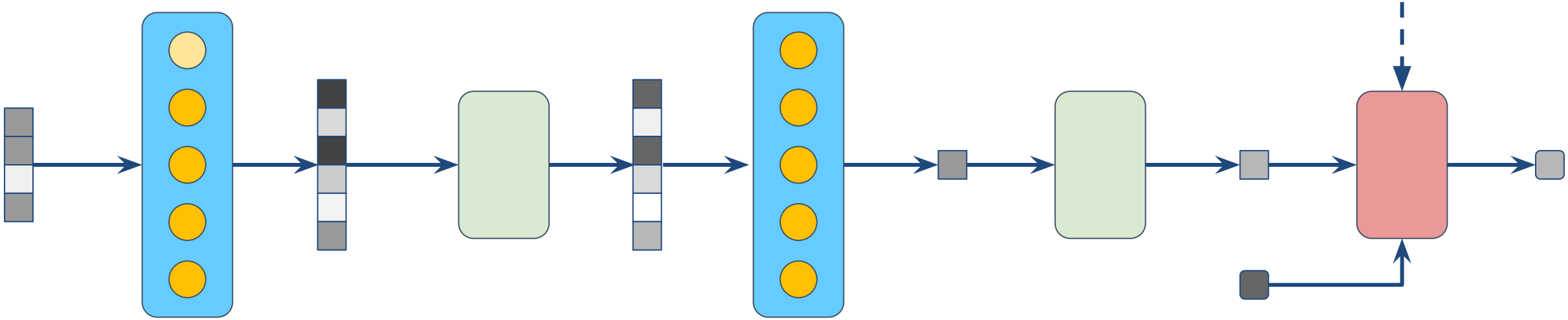
Application en DL

$$l : \mathbb{R}^n \to \mathbb{R}$$

$$\theta \mapsto c$$

N : Nombre d'exemples
n : Nombre de paramètres

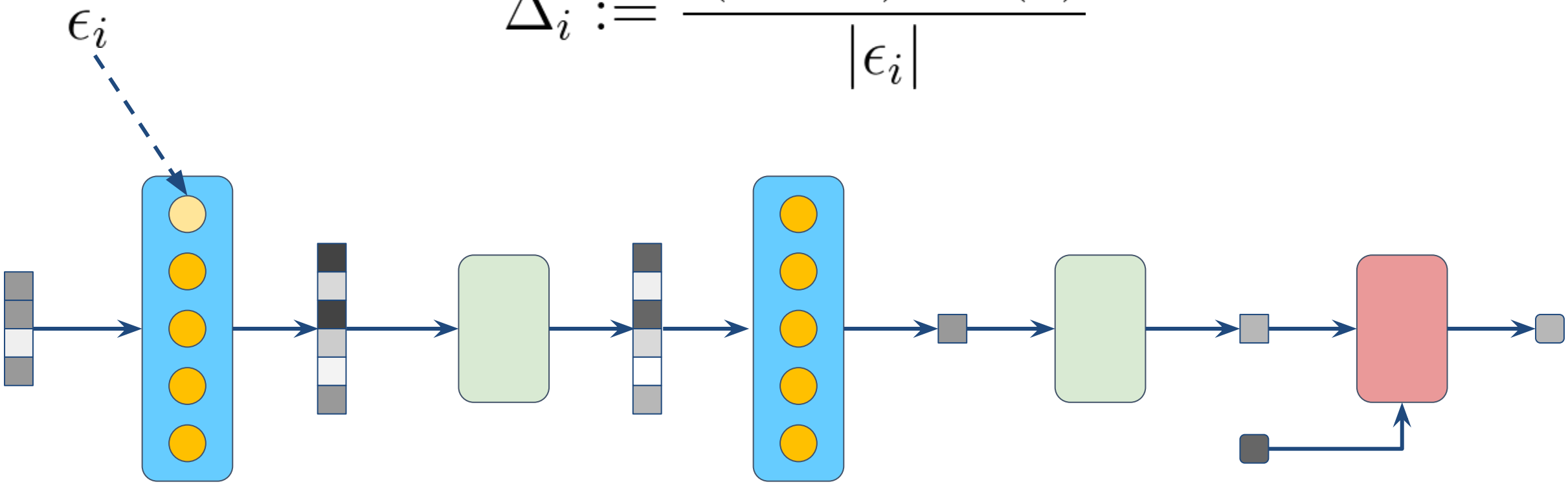
$$l(\theta) := \frac{1}{N} \sum_{k=1}^{N} l_k(\theta)$$



MILA

Impact d'un paramètre

$$\Delta_i := \frac{l(\theta + \epsilon_i) - l(\theta)}{|\epsilon_i|}$$

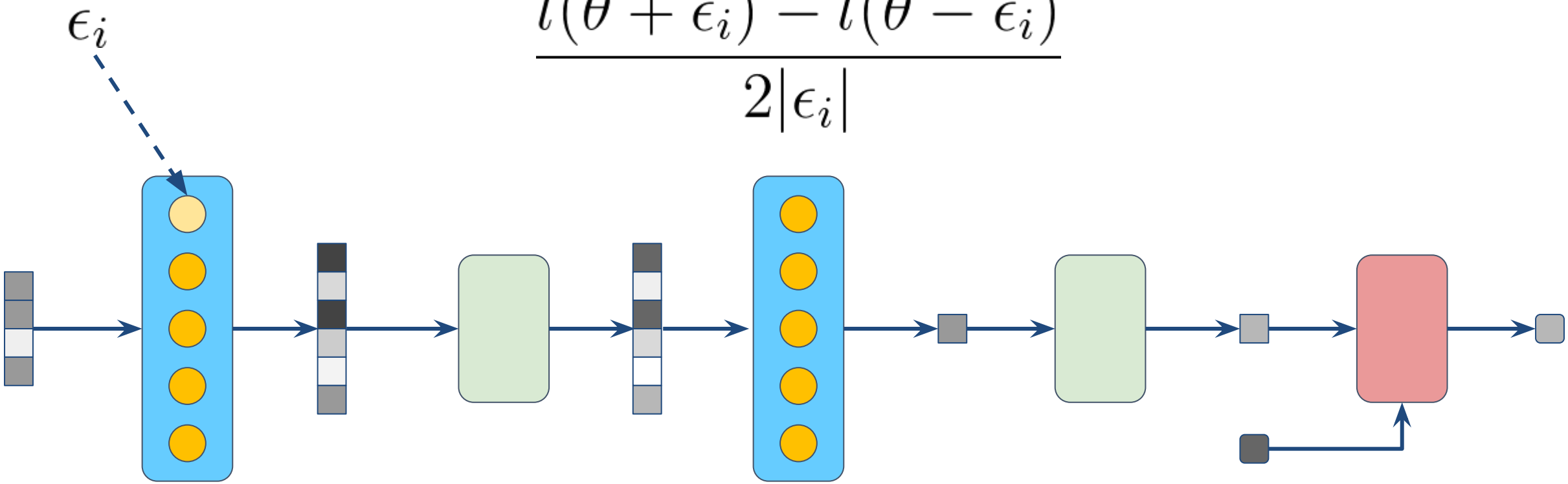


MILA

Impact d'un paramètre

Algorithme des différences finies

$$\frac{l(\theta + \epsilon_i) - l(\theta - \epsilon_i)}{2|\epsilon_i|}$$



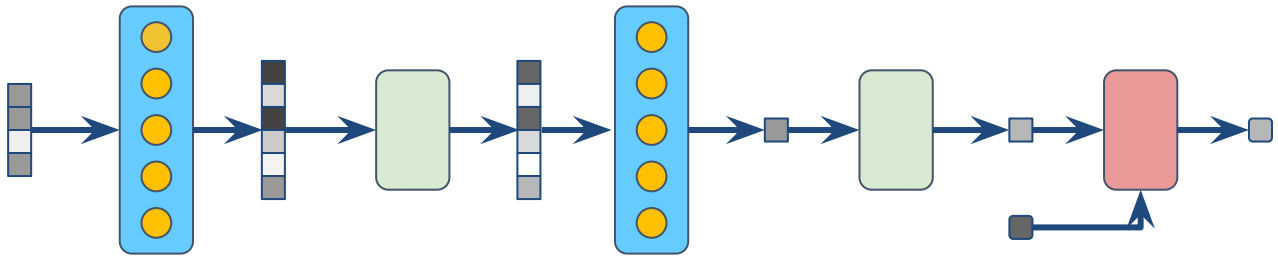
MILA

Impact de tous les paramètres

Complexité de l'algorithme des différences finies:

2*N*n propagations (séquentiel)

N : Nombre d'exemples
n : Nombre de paramètres



Réduire le nombre d'exemples

- Stochastic Gradient Descent

$$l(\theta) := \frac{1}{N}\sum_{k=1}^{N} l_k(\theta) \quad \text{Approximation} \quad \hat{l}(\theta) := \frac{1}{K}\sum_{k=1}^{K} l_{\sigma(k)}(\theta)$$

K<<N : Nombre d'exemples

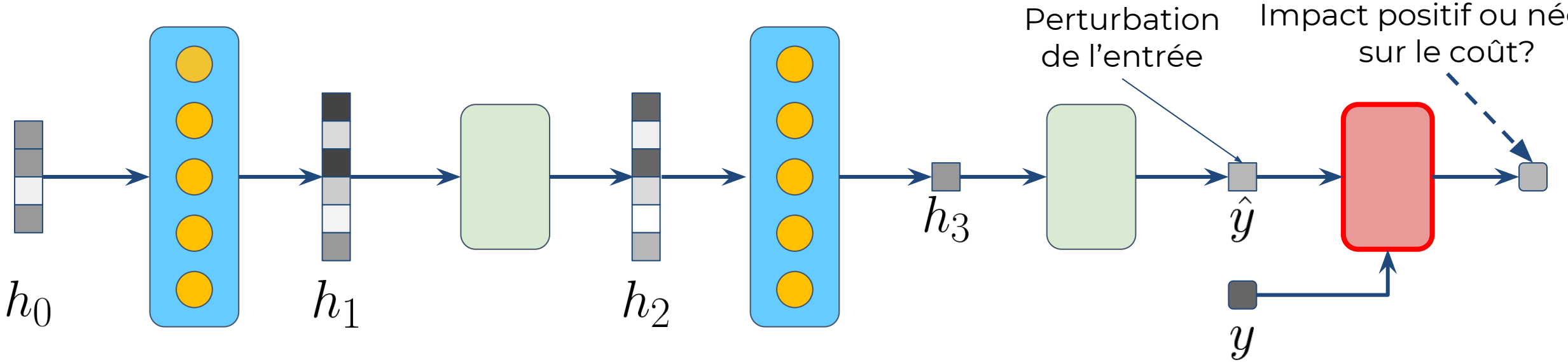
n : Nombre de paramètres

Implémentation en programmation différentiable

- Forward(x): transformation de la donnée
- Backward(g): transformation du gradient (sortie->entrée)
- Update(): calcul du gradient des paramètres

Exemple

Forward $l(\hat{y}, y) = \overbrace{-y \ln \hat{y} - (1-y) \ln(1-\hat{y})}^{\text{Binary Cross-Entropy}}$

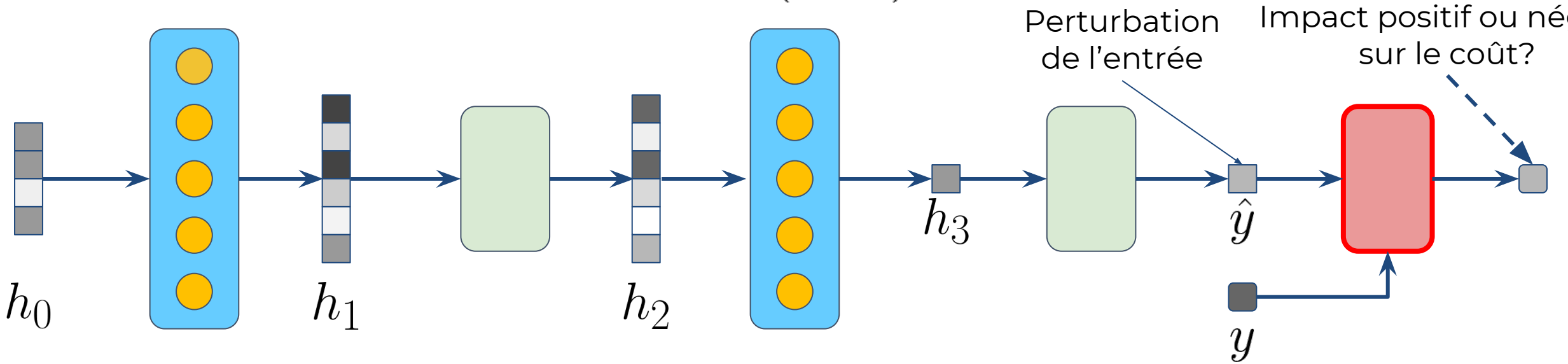


Exemple

Binary Cross-Entropy

Forward $$l(\hat{y}, y) = -y \ln \hat{y} - (1 - y) \ln(1 - \hat{y})$$

Backward $$\partial_{\hat{y}} l(\hat{y}, y) = \frac{\hat{y} - y}{\hat{y}(1 - \hat{y})}$$



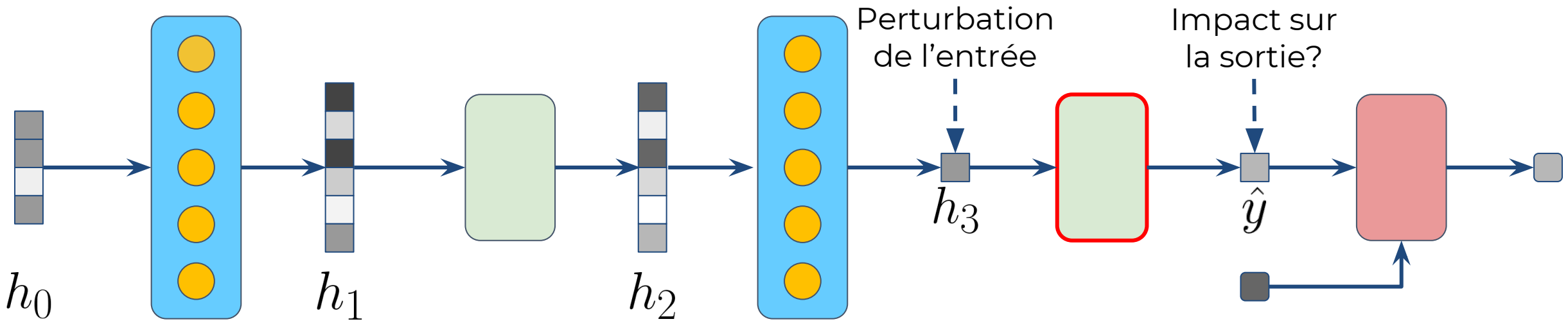
MILA

Exemple

Sigmoid

Forward $g(x) = \frac{1}{1+e^{-x}}$

Backward $g'(x) = g(x)(1 - g(x))$



Exemple

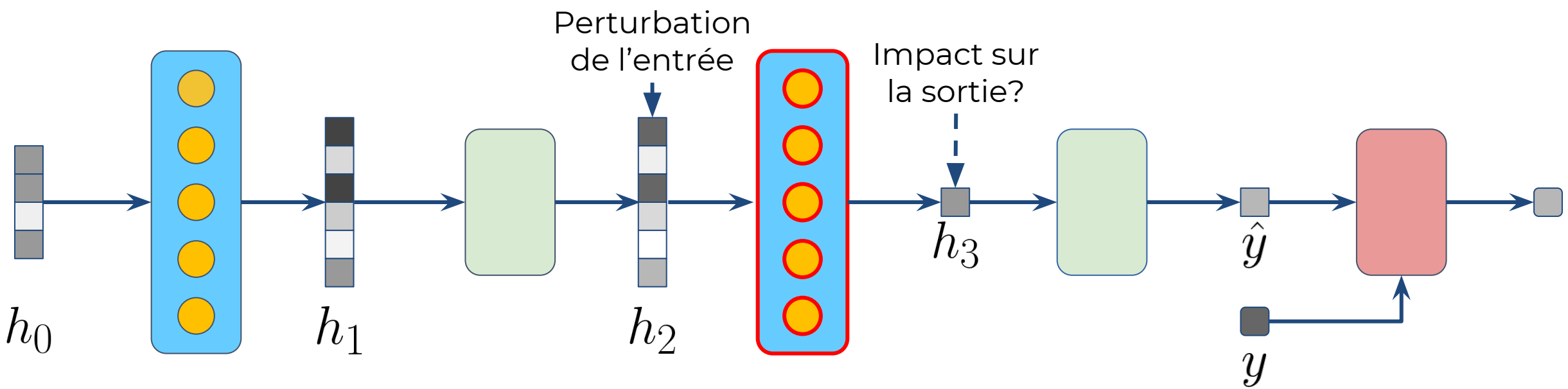
Linéaire

Forward $g(x; W, b) = Wx + b$

Update $\partial_W g(x; W, b) = x^\top$

Backward $\partial_x g(x; W, b) = W^\top$

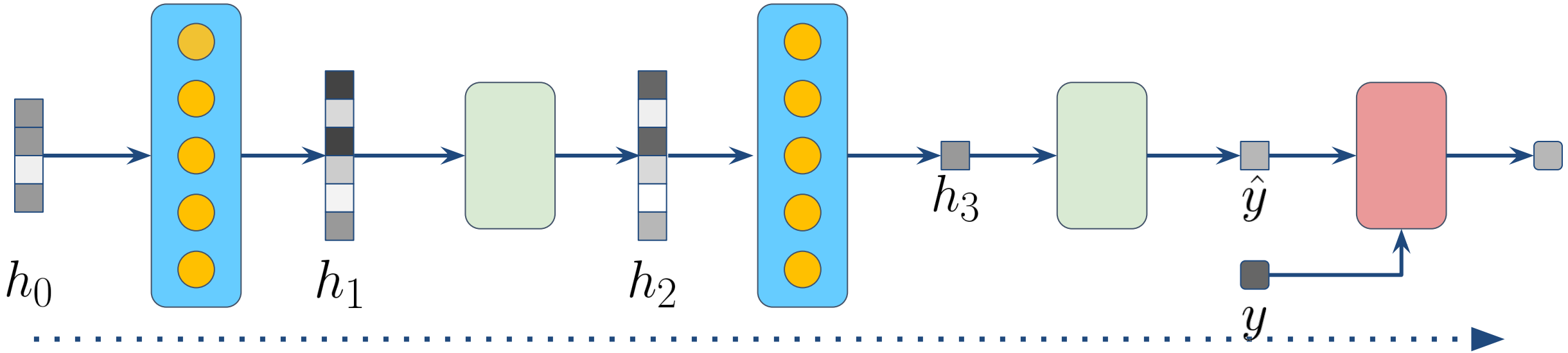
$\partial_b g(x; W, b) = \mathbb{I}$



MILA

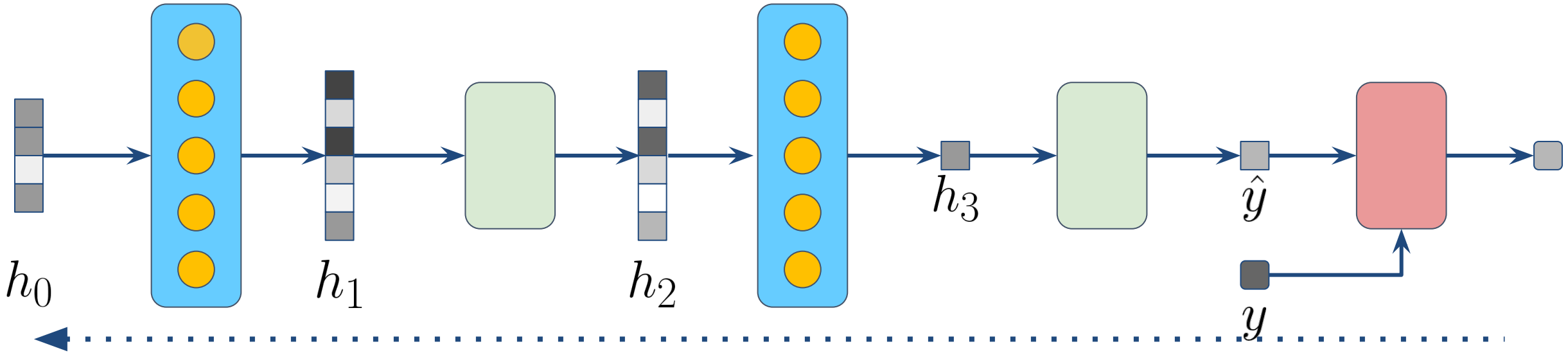
Exemple

- Forward: calcul du point actuel de la différentielle



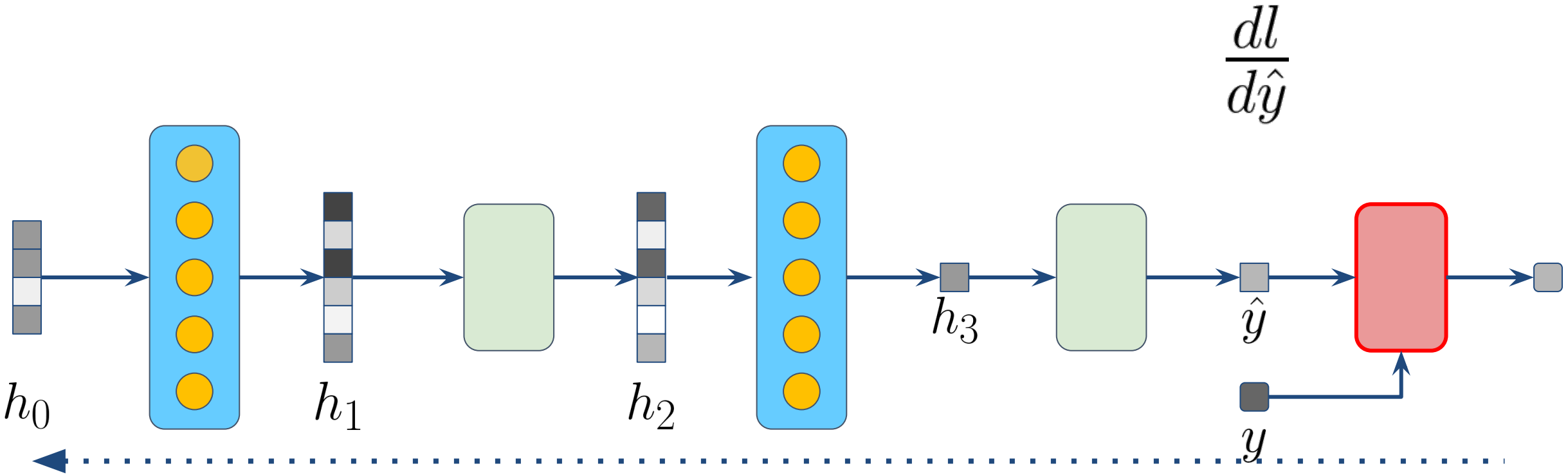
Exemple

- Backward: calcul efficace de la règle de la chaîne (version matricielle)
- **multiplication matricielle: Jacobienne*gradient**



Exemple: backward

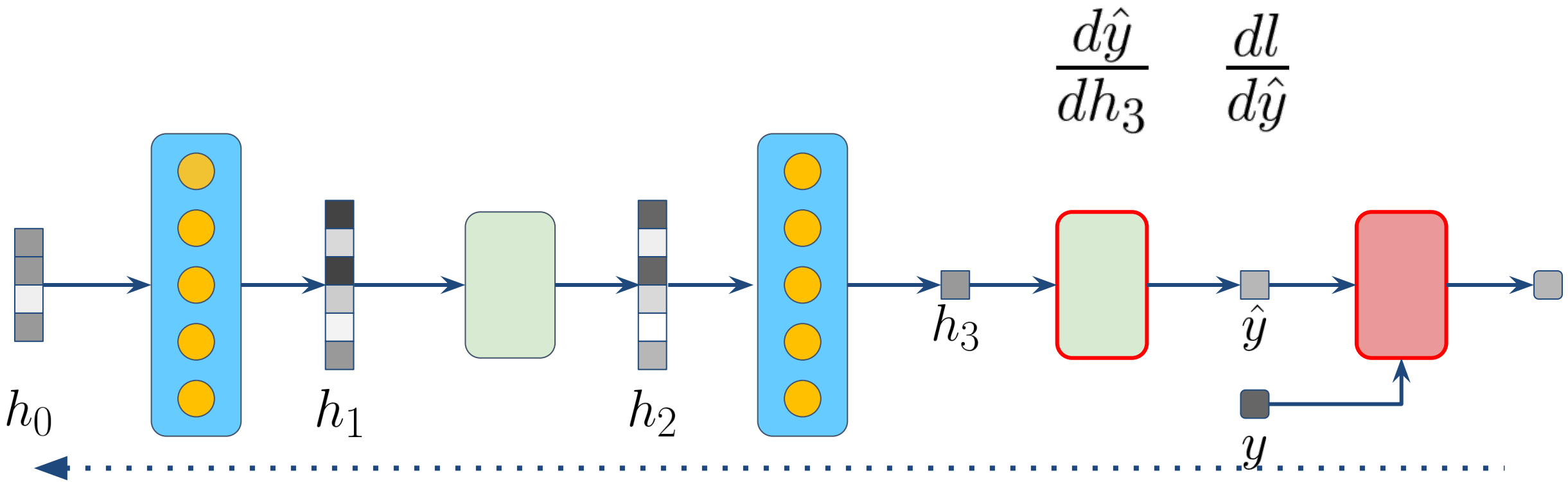
$$\frac{dh_1}{dh_0}\frac{dh_2}{dh_1}\frac{dh_3}{dh_2}\frac{d\hat{y}}{dh_3}\underbrace{\frac{dl}{d\hat{y}}}_{\square}$$



MILA

Exemple: backward

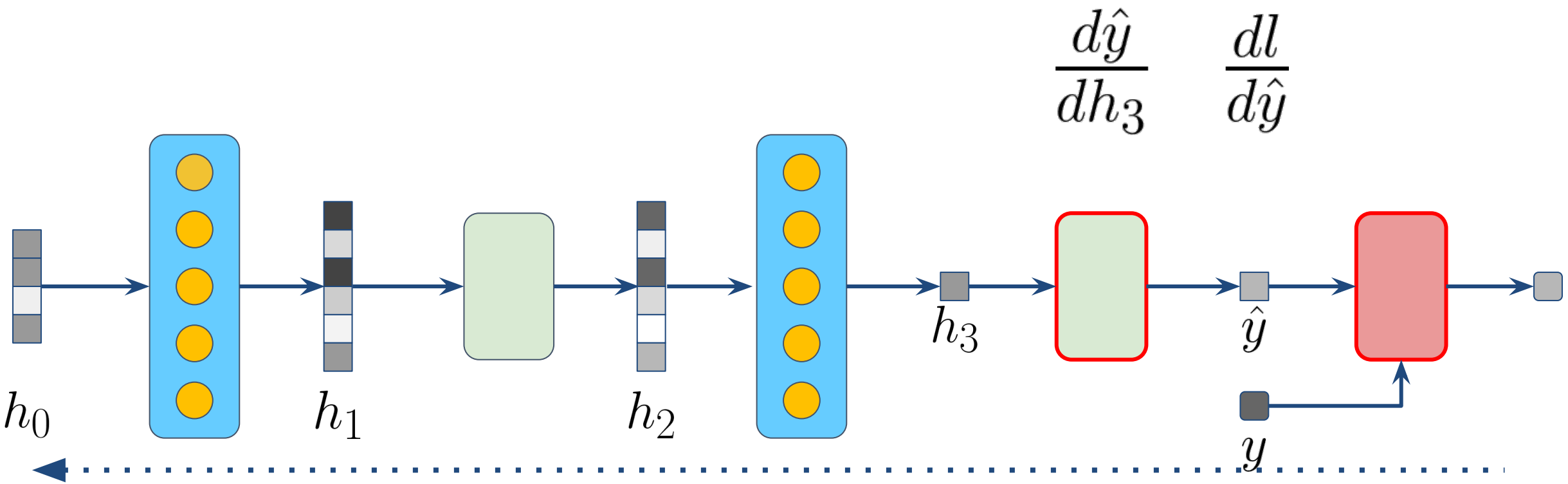
$$\frac{dh_1}{dh_0}\frac{dh_2}{dh_1}\frac{dh_3}{dh_2}\underbrace{\frac{d\hat{y}}{dh_3}}_{\square}\underbrace{\frac{dl}{d\hat{y}}}_{\square}$$



MILA

Exemple: backward

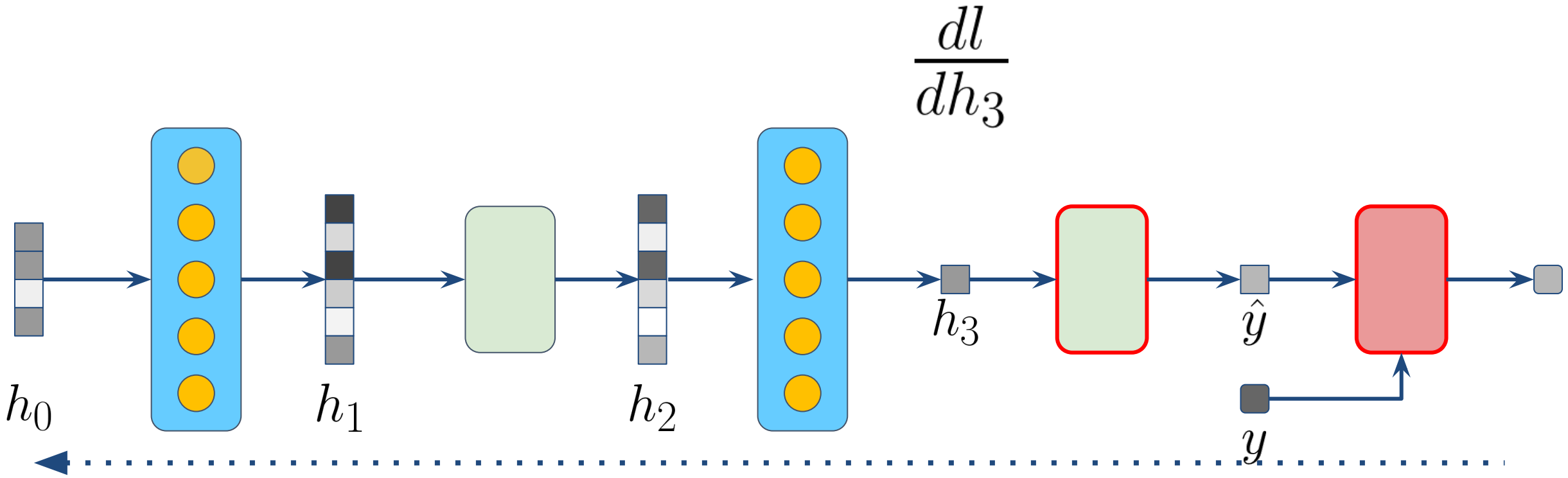
$$\frac{dh_1}{dh_0}\frac{dh_2}{dh_1}\frac{dh_3}{dh_2}\underbrace{\frac{d\hat{y}}{dh_3}\frac{dl}{d\hat{y}}}_{\square}$$



MILA

Exemple: backward

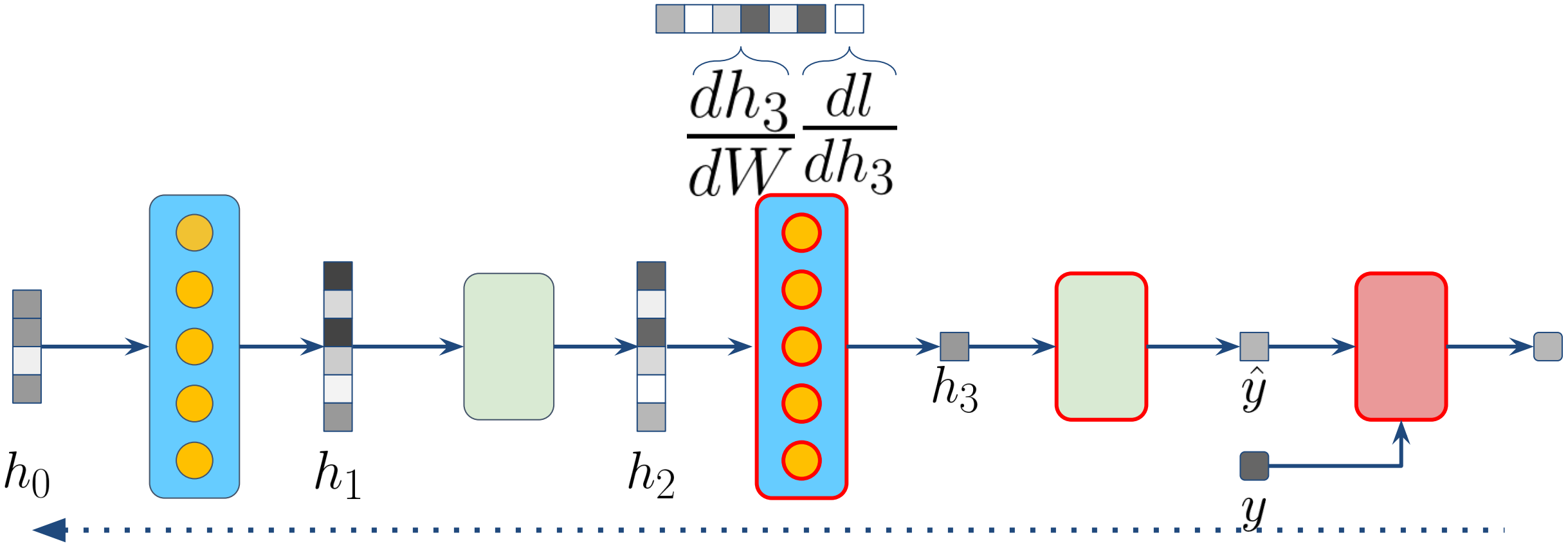
$$\frac{dh_1}{dh_0}\frac{dh_2}{dh_1}\frac{dh_3}{dh_2}\underbrace{\frac{dl}{dh_3}}_{\square}$$



MILA

Exemple: update

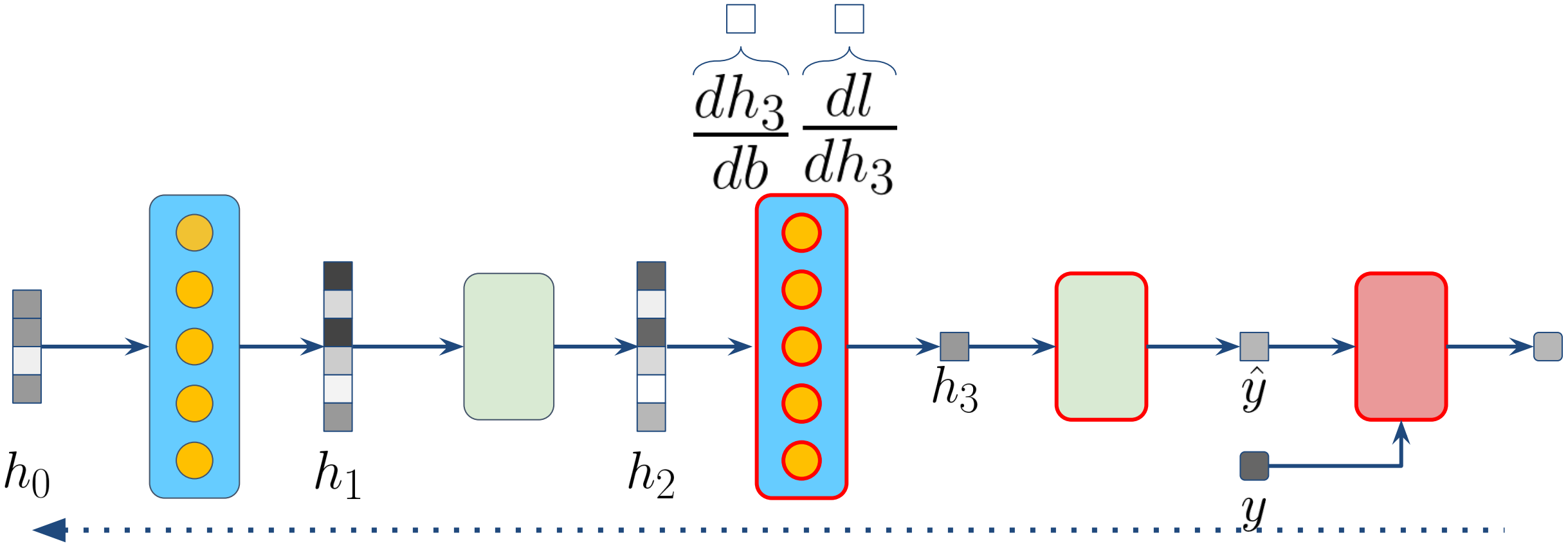
$$\frac{dh_1}{dh_0}\frac{dh_2}{dh_1}\frac{dh_3}{dh_2}\underbrace{\frac{dl}{dh_3}}_{\square}$$



MILA

Exemple: update

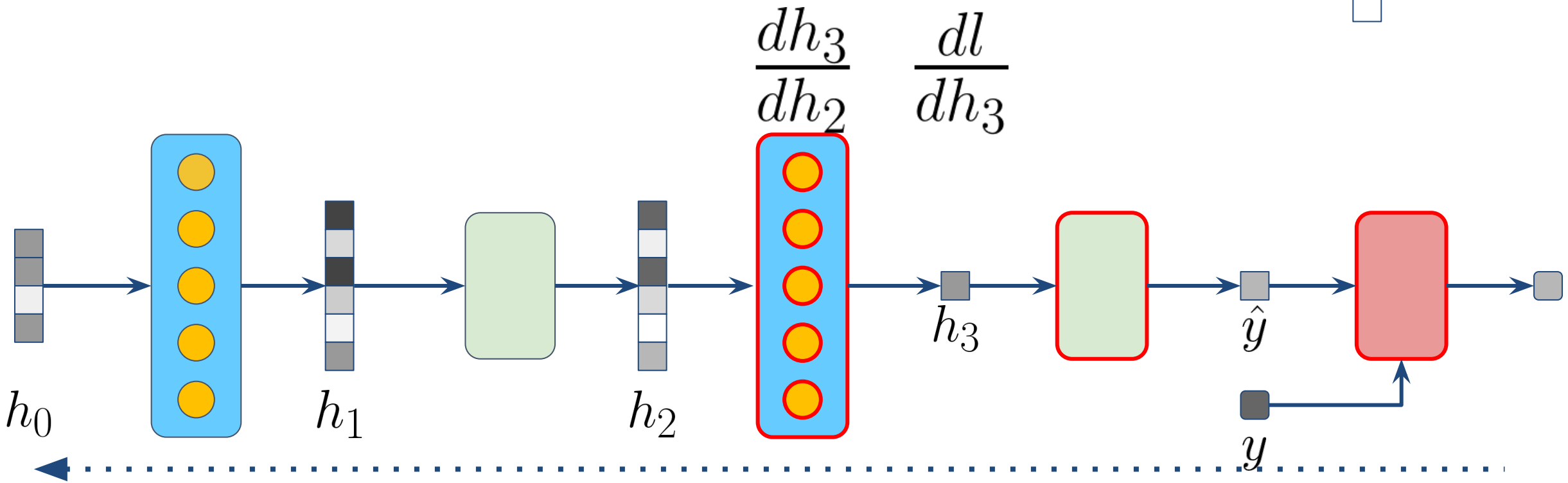
$$\frac{dh_1}{dh_0}\frac{dh_2}{dh_1}\frac{dh_3}{dh_2}\frac{dl}{dh_3}$$



MILA

Exemple: backward

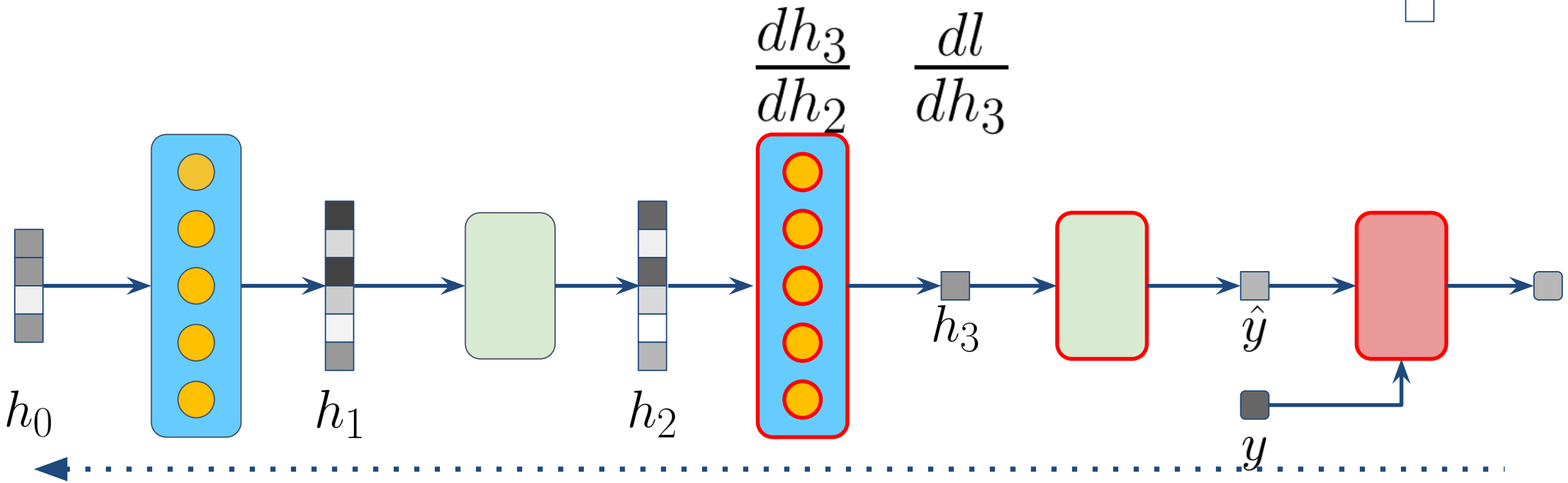
$$\frac{dh_1}{dh_0}\frac{dh_2}{dh_1}\underbrace{\frac{dh_3}{dh_2}}\underbrace{\frac{dl}{dh_3}}$$



MILA

Exemple: backward

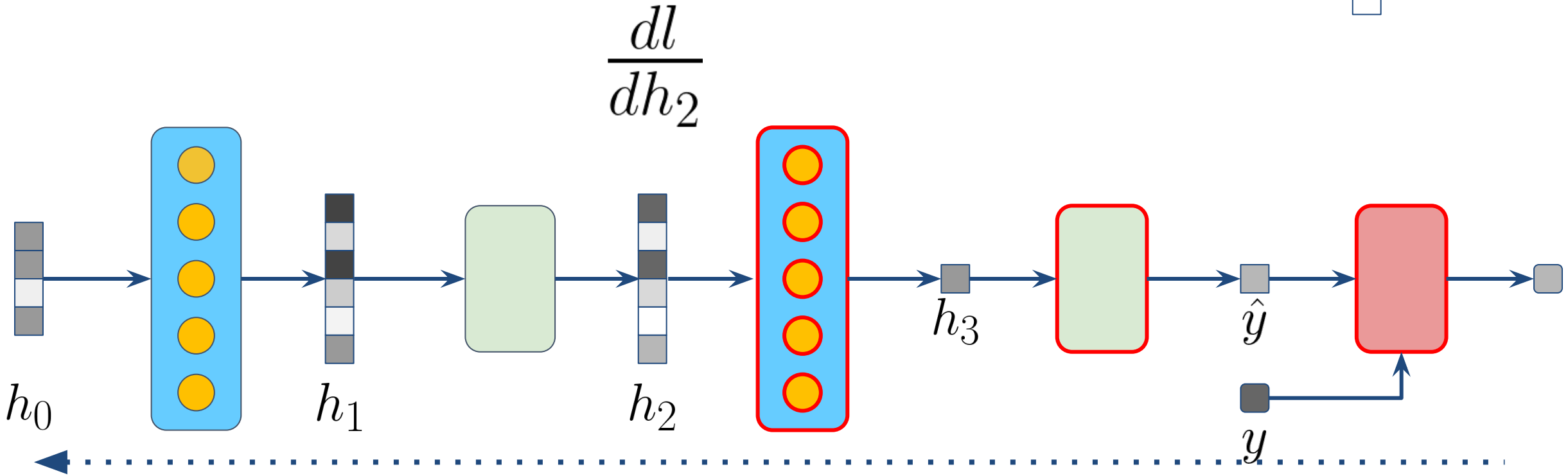
$$\frac{dh_1}{dh_0}\frac{dh_2}{dh_1}\underbrace{\frac{dh_3}{dh_2}\frac{dl}{dh_3}}$$



MILA

Exemple: backward

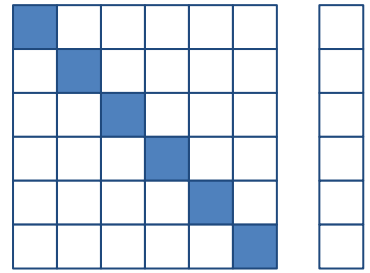
$$\frac{dh_1}{dh_0}\frac{dh_2}{dh_1}\frac{dl}{dh_2}$$



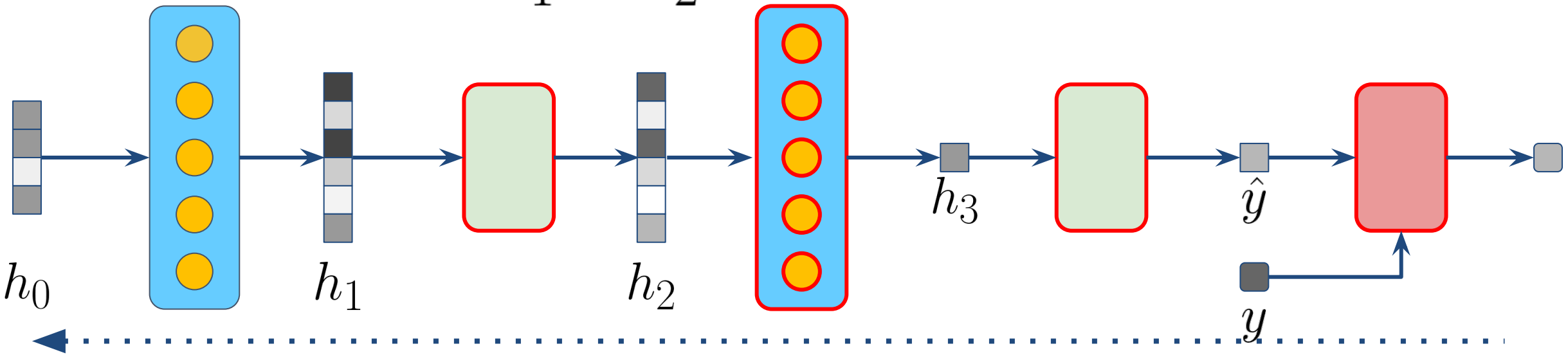
MILA

Exemple: backward

$$\frac{dh_1}{dh_0}\frac{dh_2}{dh_1}\frac{dl}{dh_2}$$

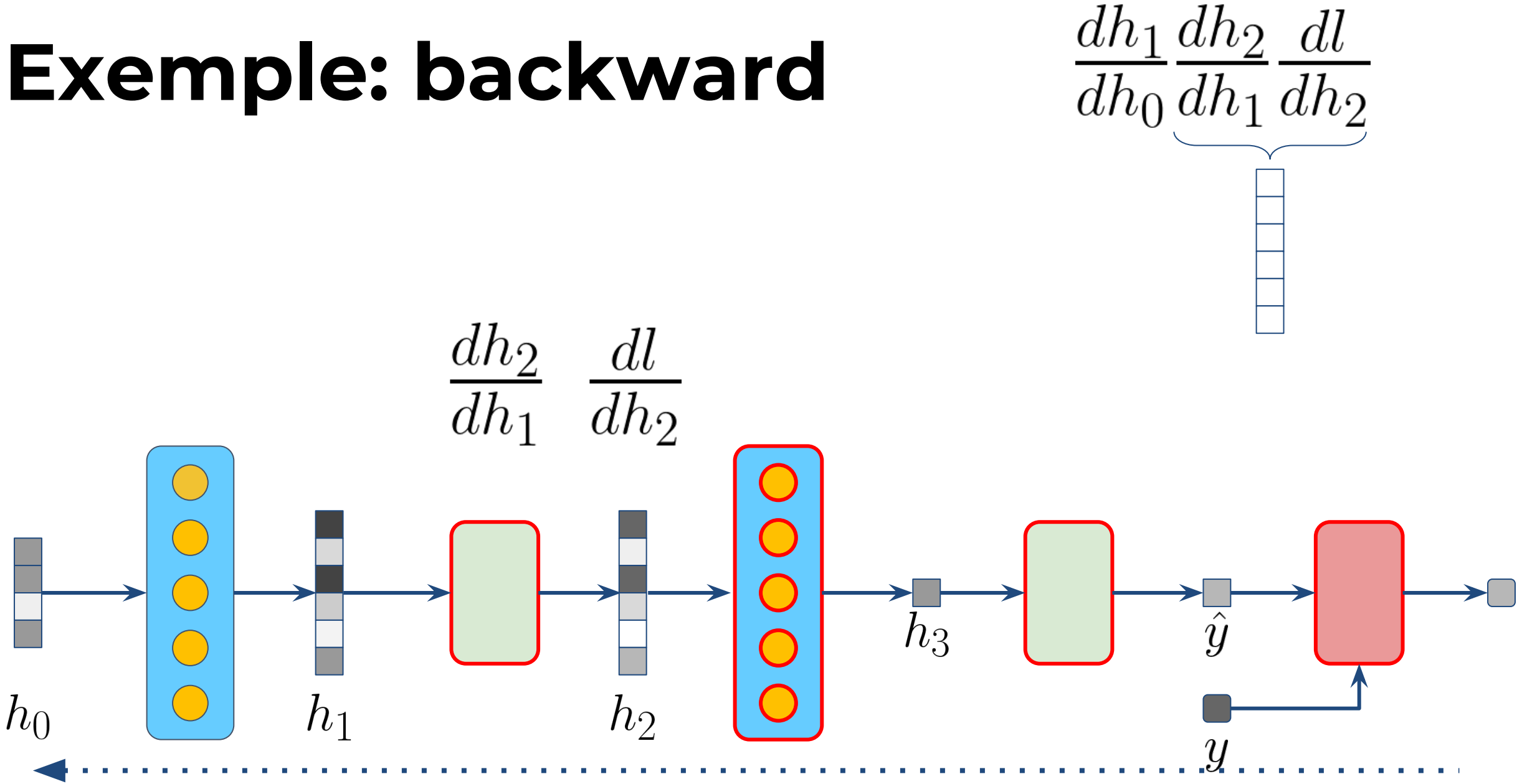


$$\frac{dh_2}{dh_1}\quad\frac{dl}{dh_2}$$



MILA

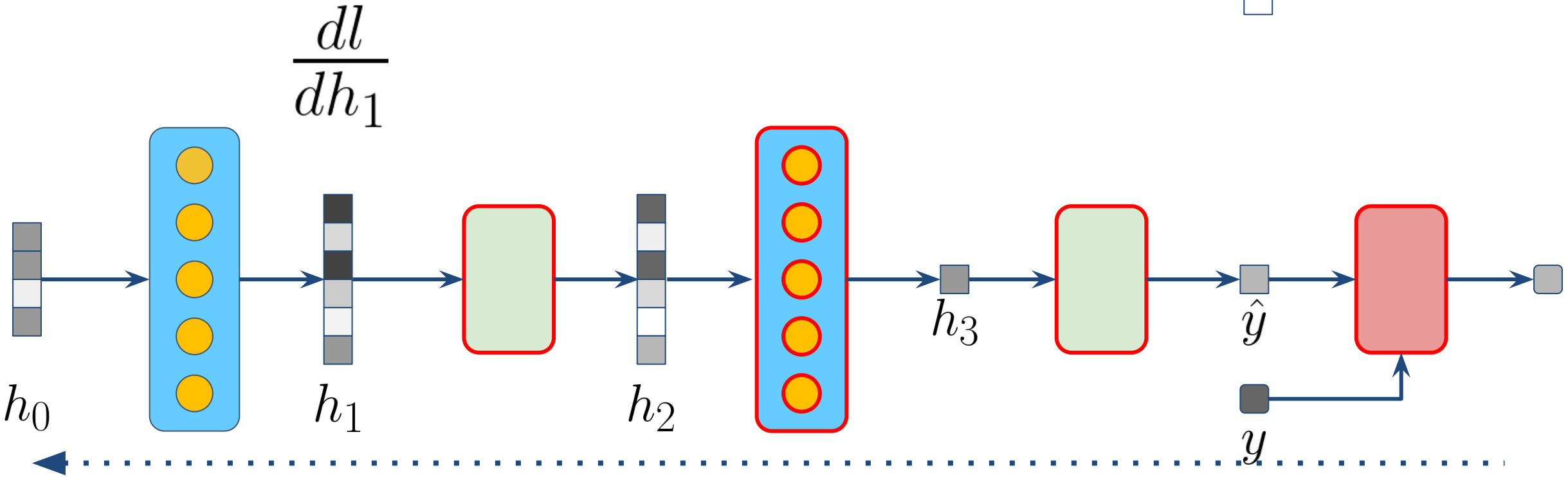
Exemple: backward



MILA

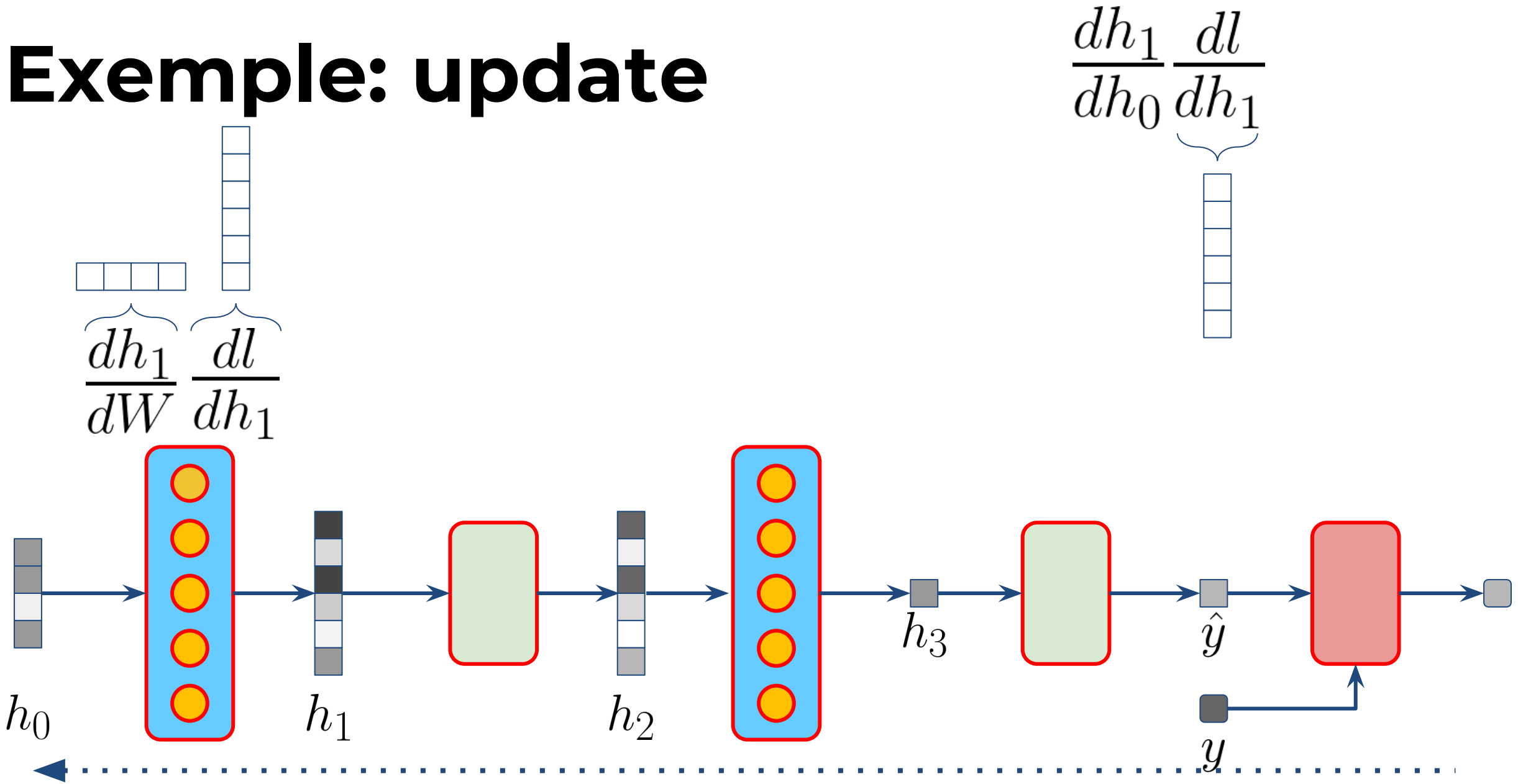
Exemple: backward

$$\frac{dh_1}{dh_0}\frac{dl}{dh_1}$$



MILA

Exemple: update

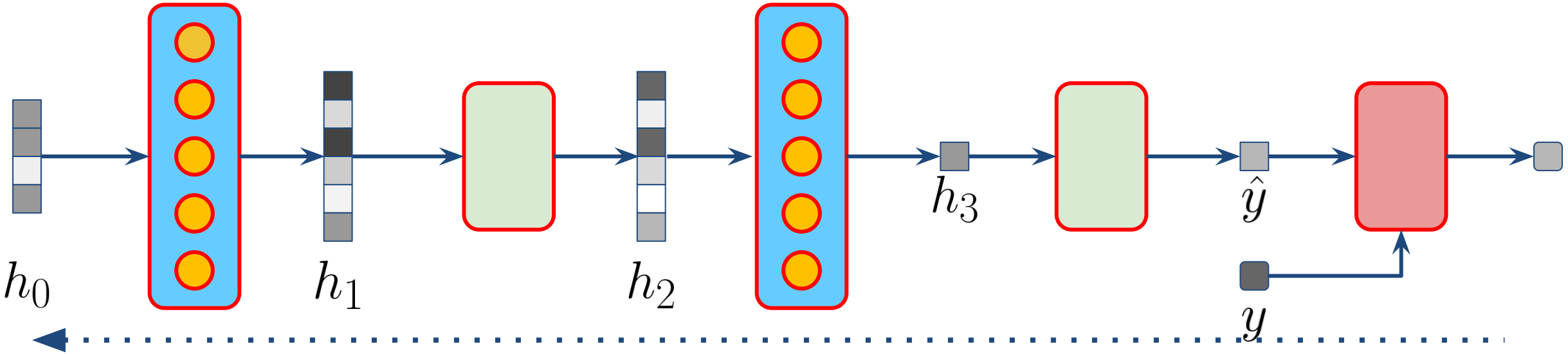


Exemple: update

$$\frac{dh_1}{dh_0}\frac{dl}{dh_1}$$

$$\frac{dh_1}{dW}\frac{dl}{dh_1}$$

Bug! On ne peut pas garder cette notation dans le cas de différentielle d'une fonction de matrice!



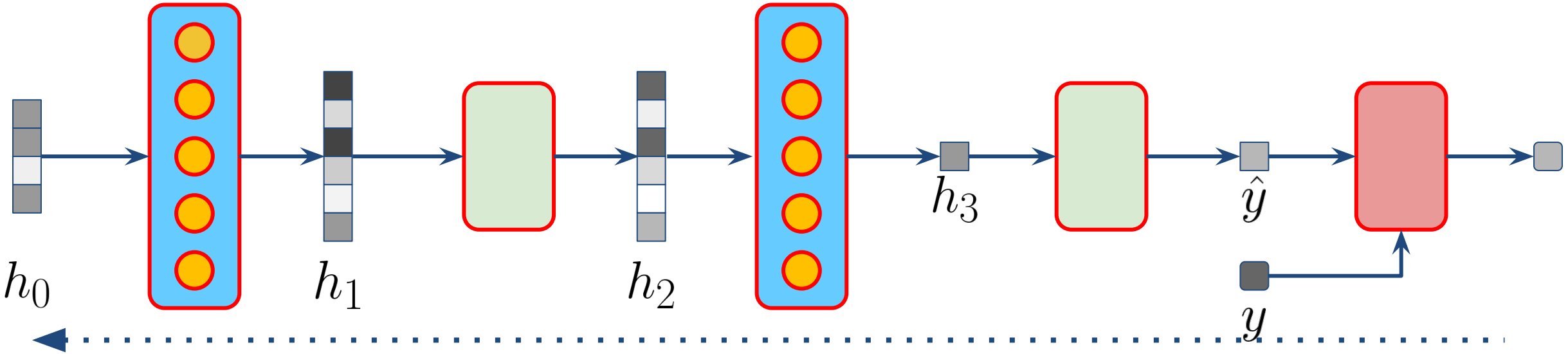
MILA

Exemple: update

$$\frac{dh_1}{dh_0}\frac{dl}{dh_1}$$

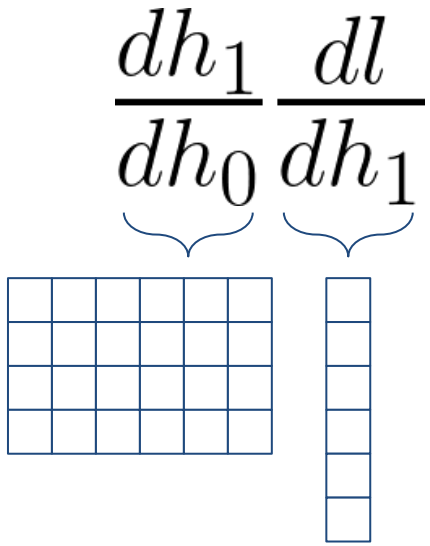
Bug! On ne peut pas garder cette notation dans le cas de différentielle d'une fonction de matrice!

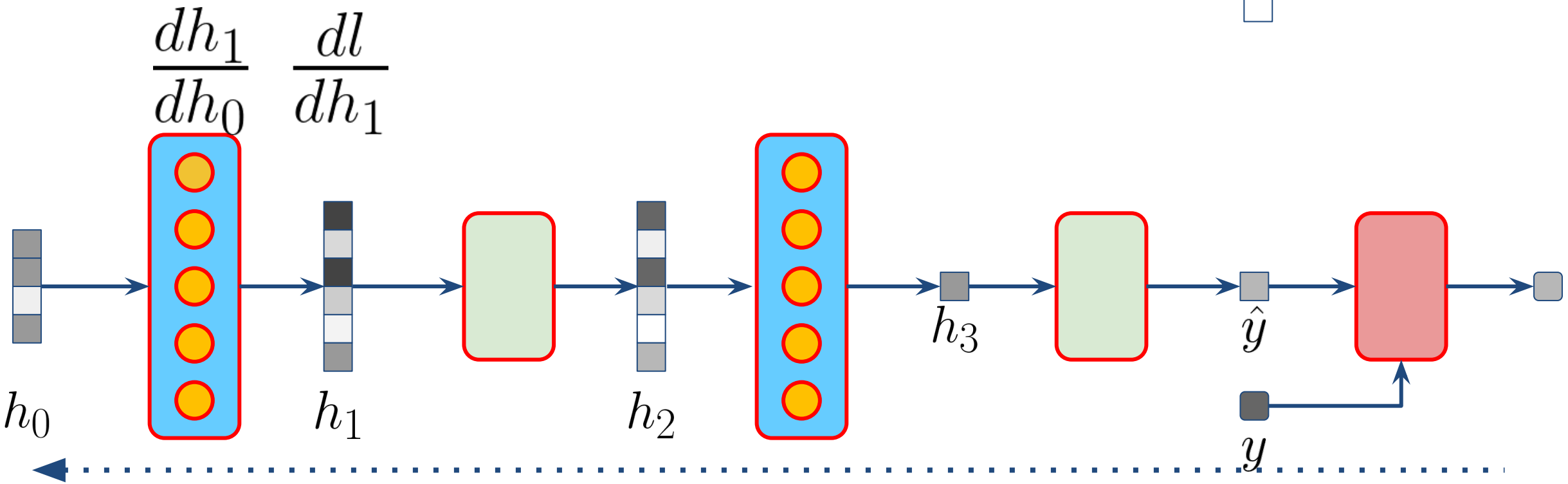
$$\frac{dl}{dh_1}\frac{dh_1}{dW}$$



MILA

Exemple: backward

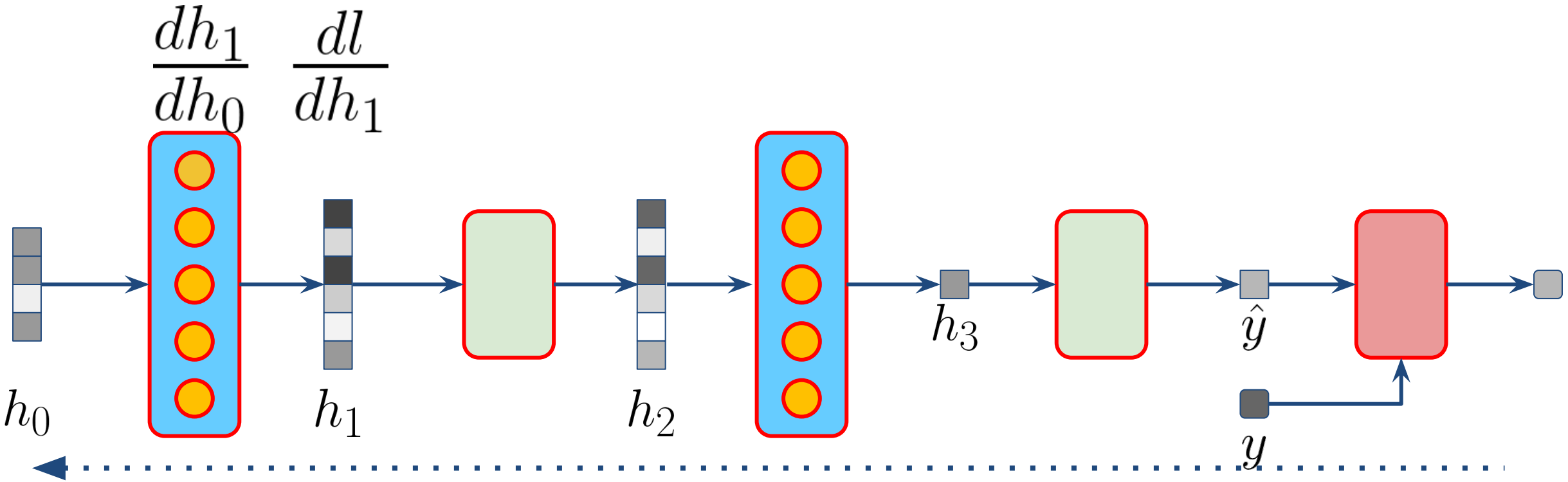




MILA

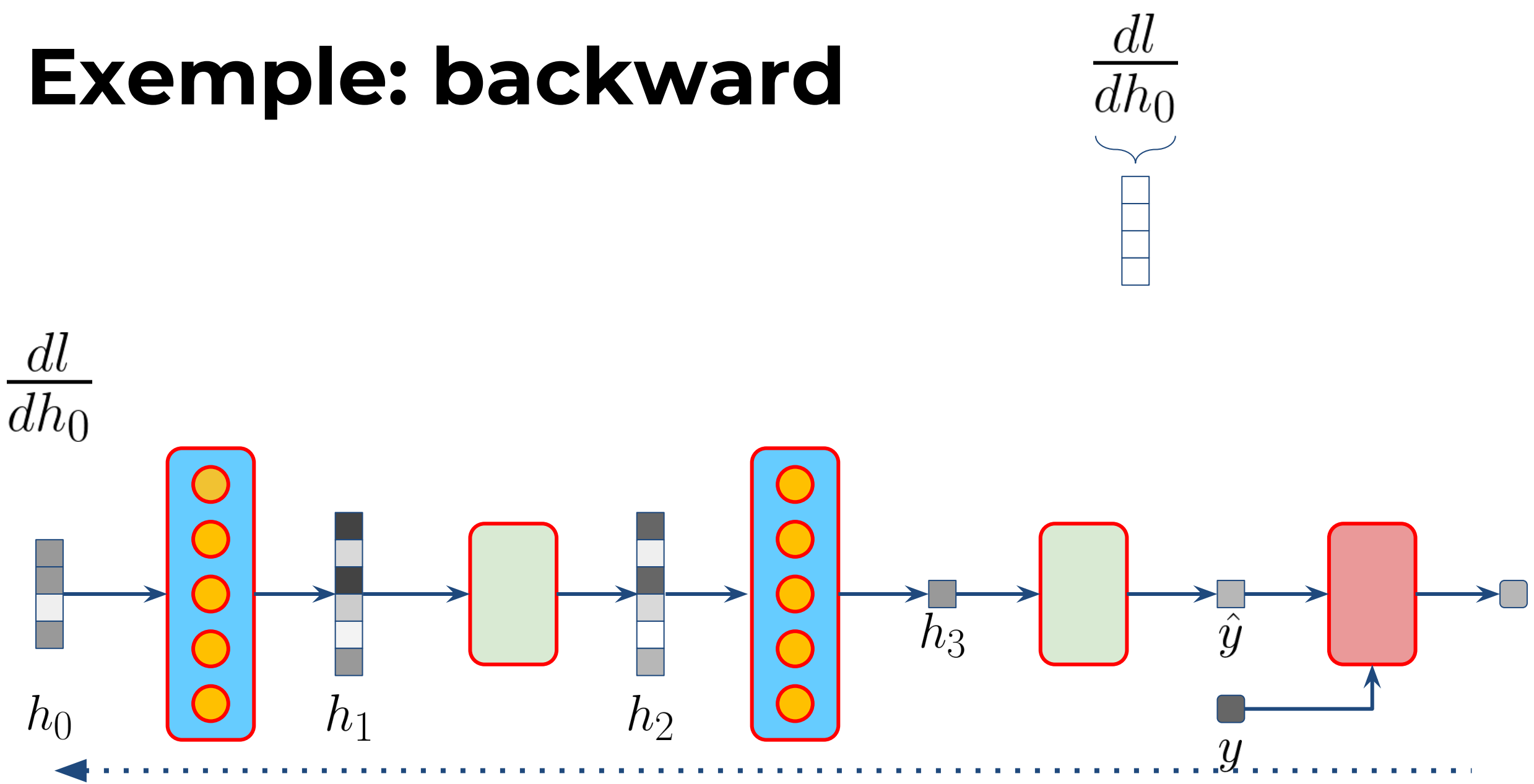
Exemple: backward

$$\frac{dh_1}{dh_0}\frac{dl}{dh_1}$$



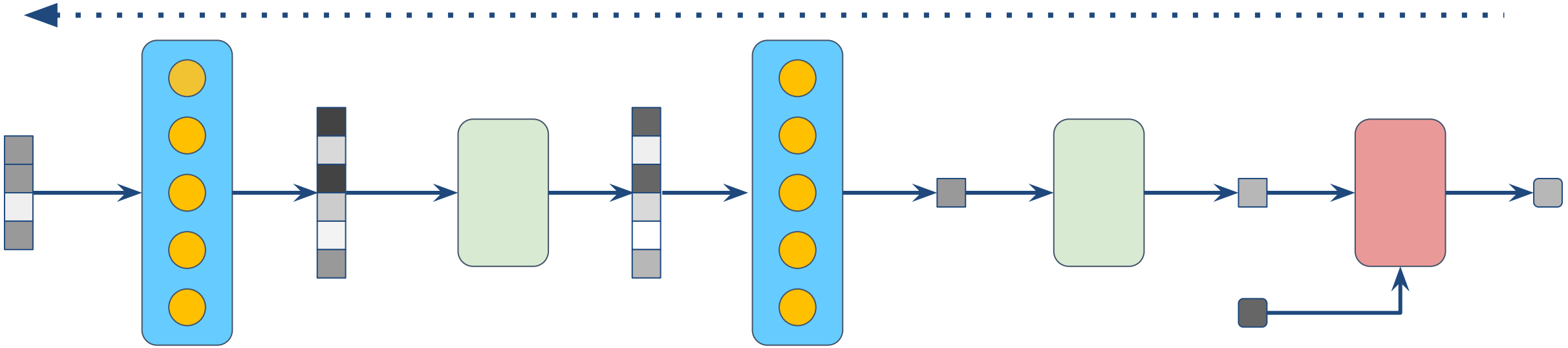
MILA

Exemple: backward



Calcul efficace: Backpropagation

- Backward: calcul efficace de la règle de la chaîne (version matricielle)
- **Optimisation: Jacobienne*gradients**
- On peut empiler plusieurs exemples

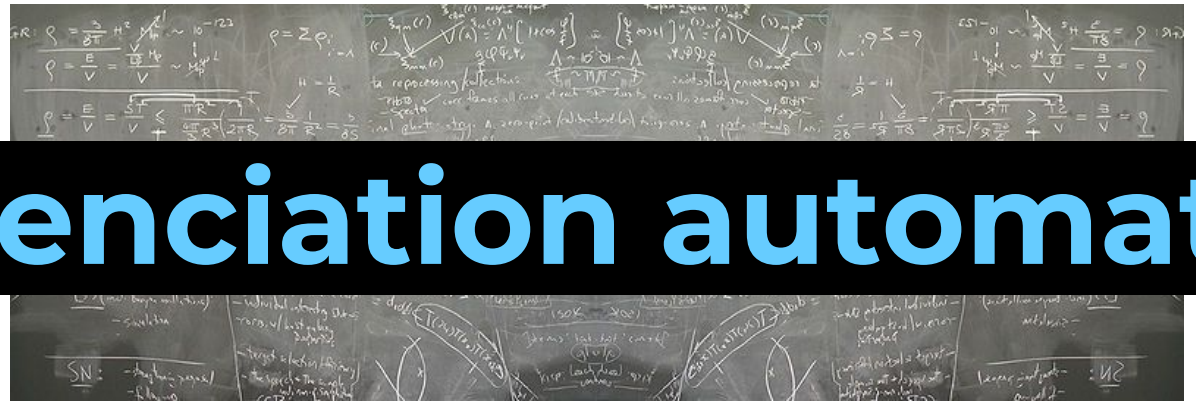


Implémentation

- Forward(x): transformation de la donnée

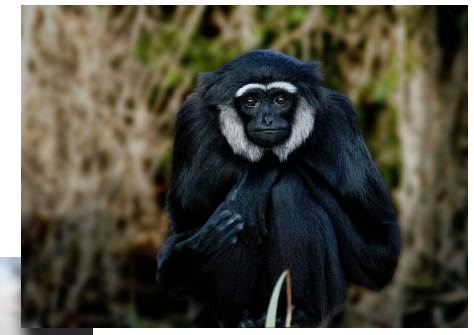
- **Backward**(g): transformation du gradient (sortie->entrée)

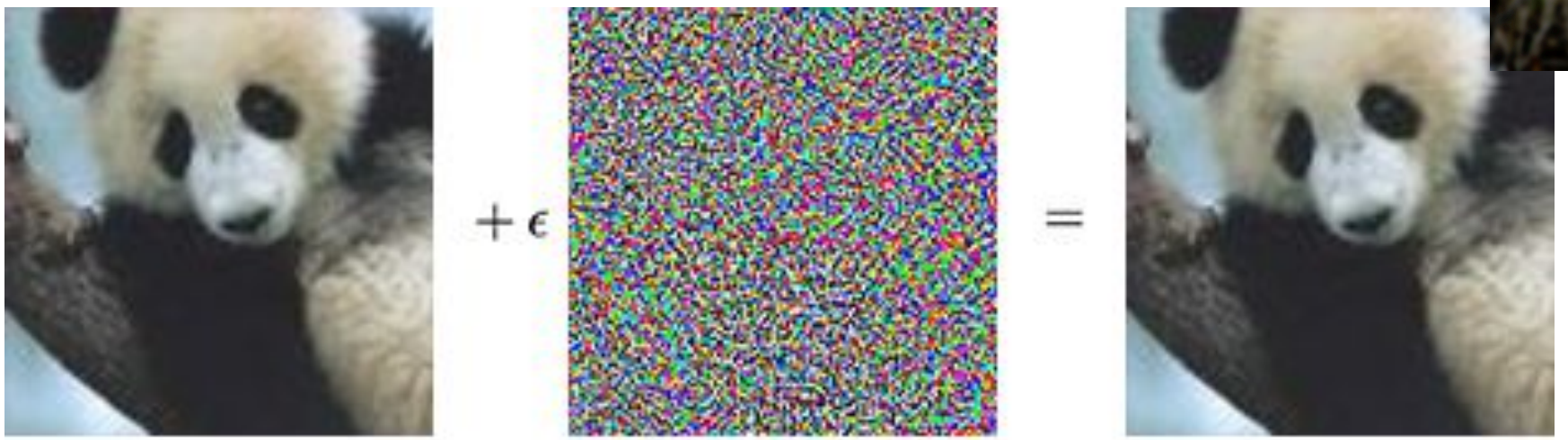
- **Update**(): calcul du gradient des paramètres



Différenciation automatique

Exemple adversarial





"panda"

57.7% confidence

"gibbon"

99.3% confidence

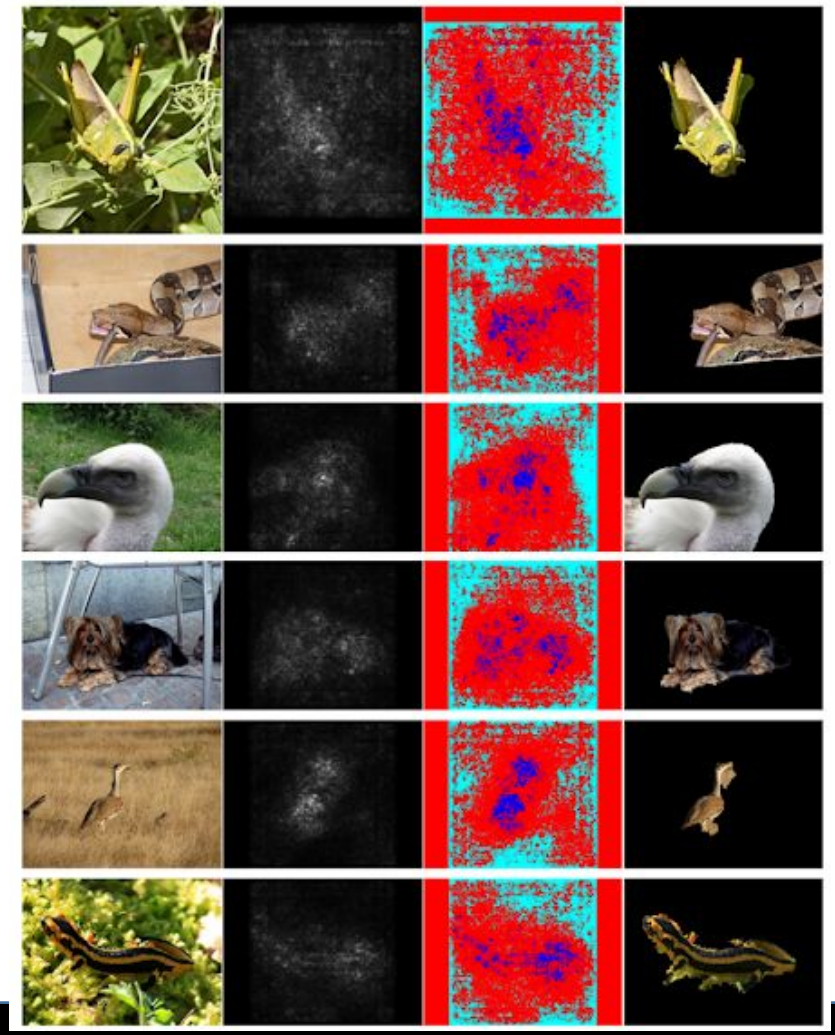
MILA

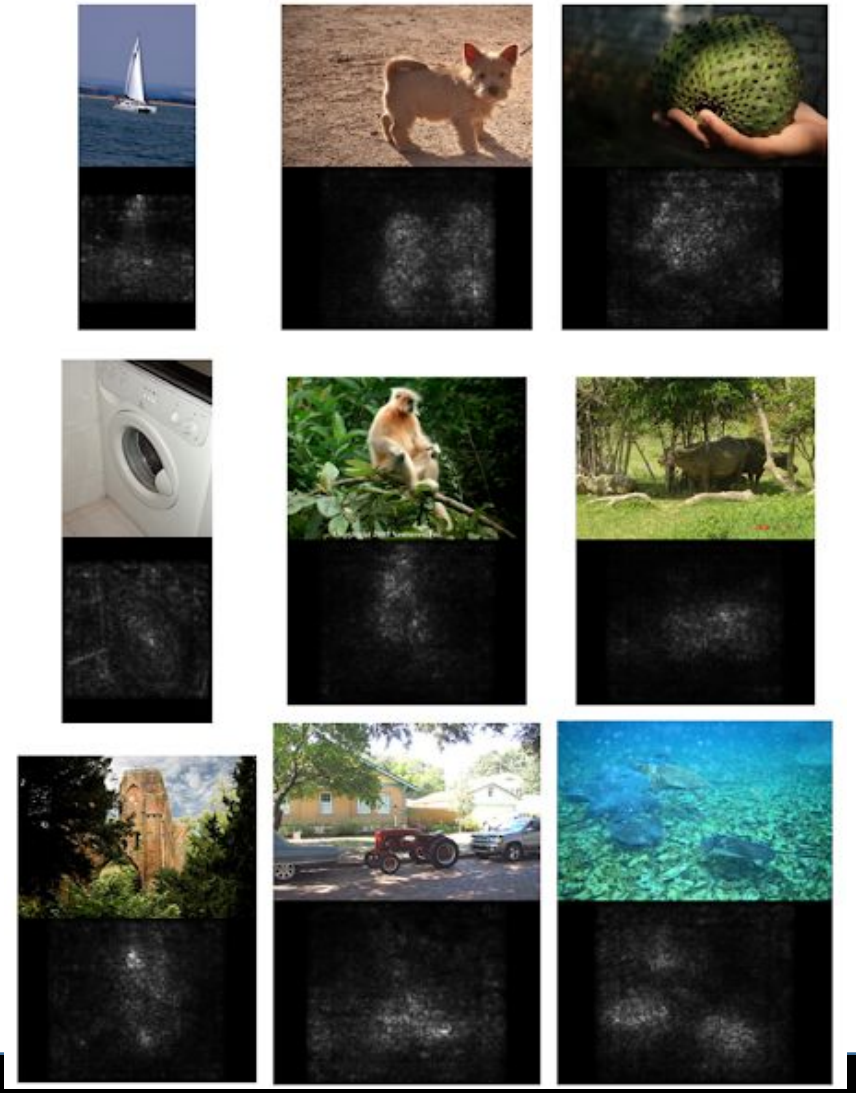
DeepDream

https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

MILA

Saliency map

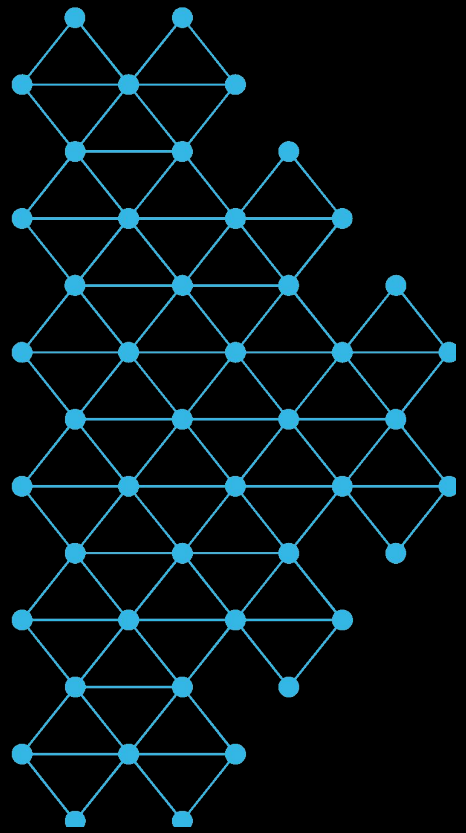




Simonyan, Karen, Andrea Vedaldi, and Andrew Zisserman. "Deep inside convolutional networks: Visualising image classification models and saliency maps." *arXiv:1312.6034* (2013).

Erhan, Dumitru, et al. "Visualizing higher-layer features of a deep network." *University of Montreal* 1341.3 (2009): 1.

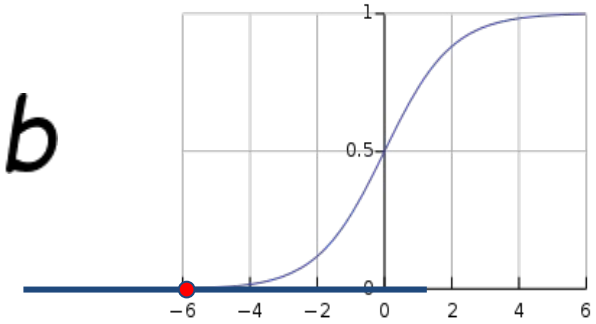
MILA

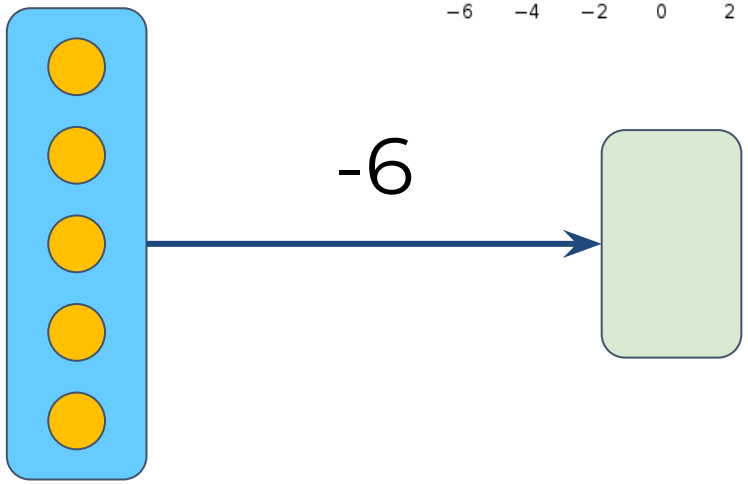


Initialisation des paramètres

$$z = Wx + b$$

Sigmoid





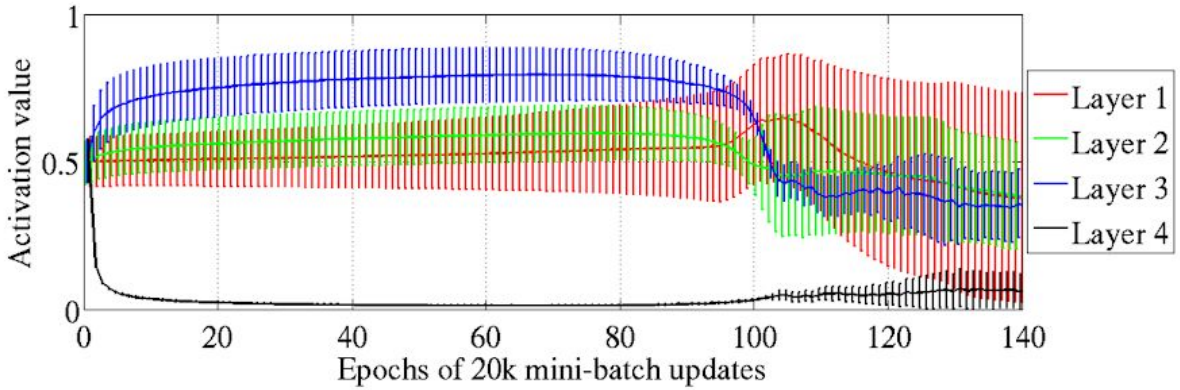


Figure 2: *Mean and standard deviation (vertical bars) of the activation values (output of the sigmoid) during supervised learning, for the different hidden layers of a deep architecture. The top hidden layer quickly saturates at 0 (slowing down all learning), but then slowly desaturates around epoch 100.*

Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 2010.

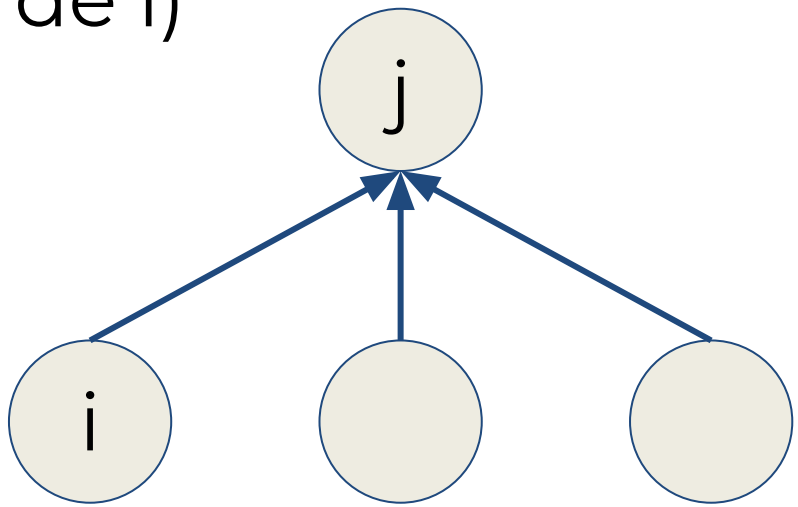
MILA

Initialisation des paramètres

- S’assurer que le gradient ne sature pas à l’initialisation avec la distribution des données

- L’initialisation devrait être invariantes au nombre d’entrées (Variance de j = Variance de i)

$$w_{i \to j} = N\left(0, \frac{1}{\sqrt{N_{in}(j)}}\right)$$



Normalisation Batchnorm

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

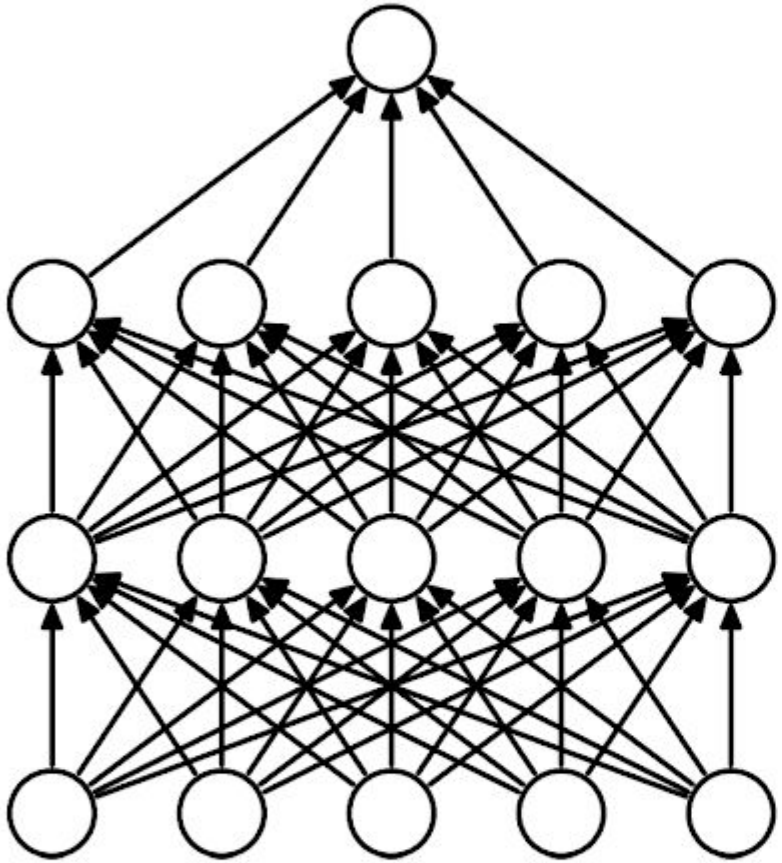
$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

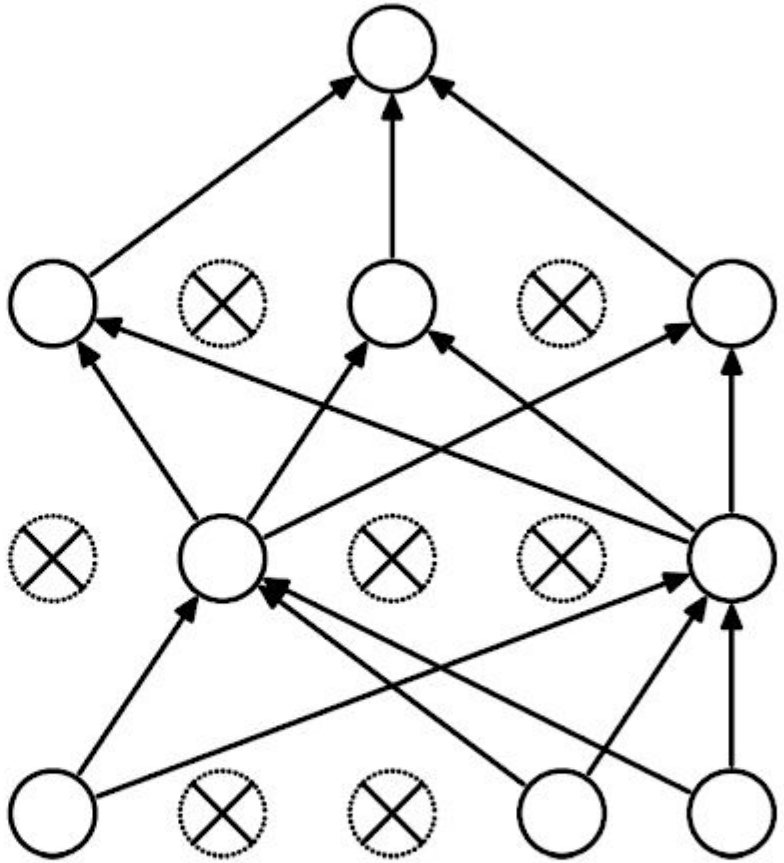
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *International conference on machine learning*. 2015.

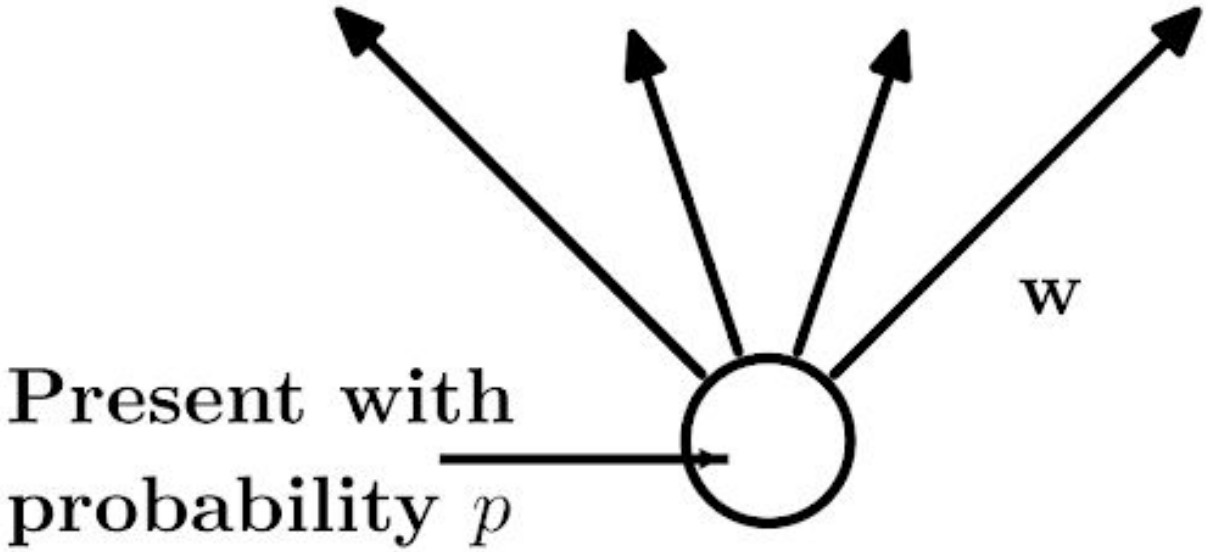
Régularisation Dropout



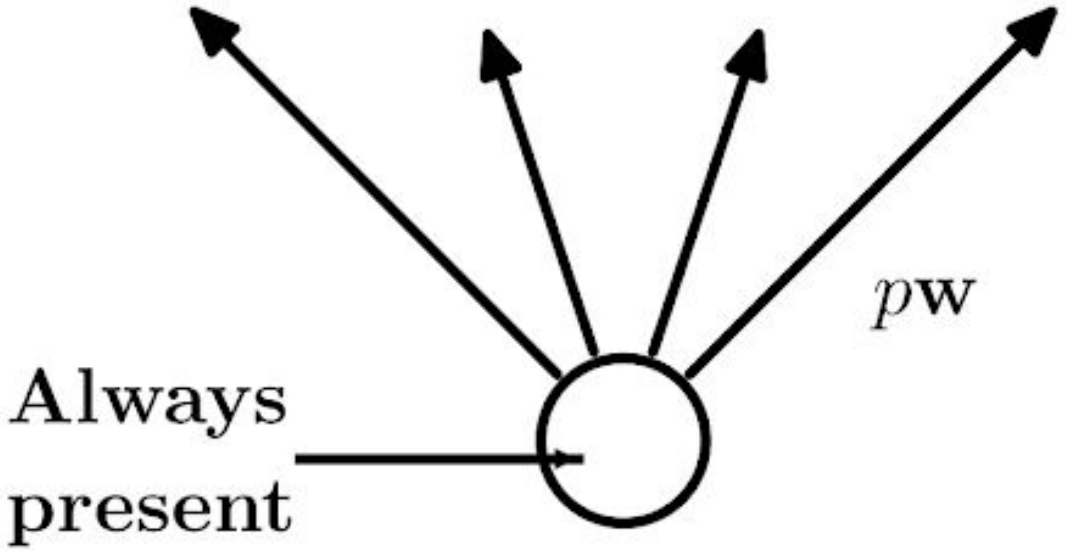


Srivastava, Nitish, et al. "Dropout: A simple way to prevent neural networks from overfitting." The Journal of Machine Learning Research 15.1 (2014): 1929-1958.

Régularisation Dropout



(a) At training time



(b) At test time

Srivastava, Nitish, et al. "Dropout: A simple way to prevent neural networks from overfitting." The Journal of Machine Learning Research 15.1 (2014): 1929-1958.

MILA

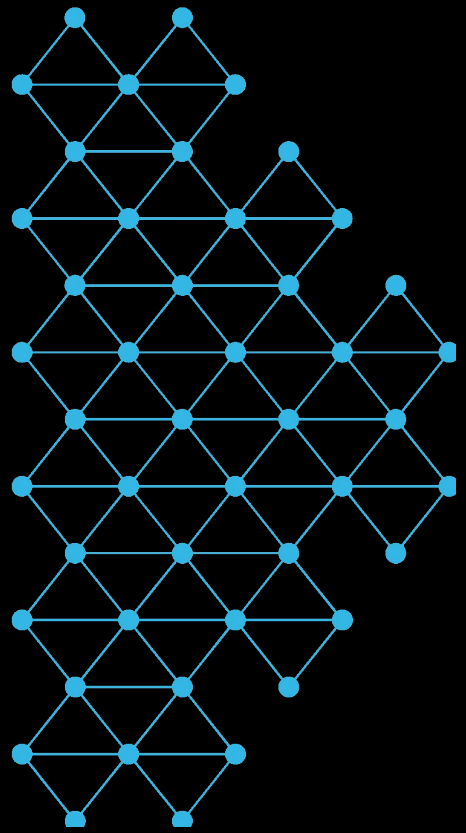
Références

- Cours de Hugo Larochelle
  https://goo.gl/XfekK1

- Colah's blog on calculus on Computational graph:
  https://goo.gl/tW1S9W

- Michael Nielsen's chapter on backpropagation:
  https://goo.gl/uMVAeG

- Feature Visualization in Distill Journal:
  https://goo.gl/ZeaKU7



Merci pour votre attention!

Contacts

https://mila.quebec/

Gaétan Marceau Caron, Membre MILA R&D,
gaetan.marceau.caron@rd.mila.quebec

Myriam Côté, Directrice équipe MILA R&D,
myriam.cote@rd.mila.quebec

MILA