Réseaux de neurones profonds

Guillaume Bourmaud

PLAN

I. Réseaux de neurones profonds

II. Modèles de fondation

III. « Gradient checkpointing »

IV. Apprentissage multi-GPU

I) Réseaux de neurones profonds

I)

Résumé des ingrédients du « Deep Learning »

1) Grande base de données étiquetées

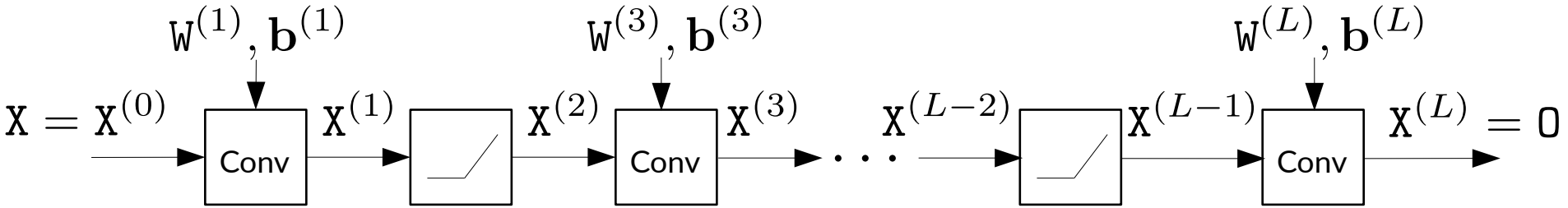
2) **« Bonne »** architecture de réseau de neurones profond

- « Perceptron » multicouche, Réseau de neurones à convolution, Transformer
- Optimisation par descente de gradient stochastisque (AdamW, etc.)

3) Grande capacité de calculs en parallèle (GPUs)

I)

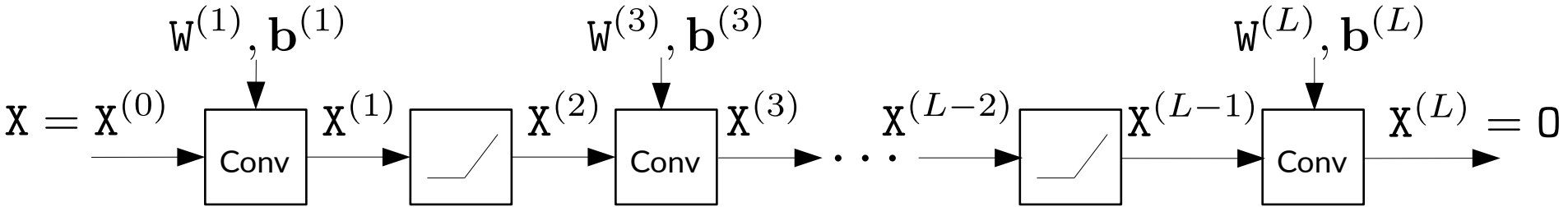
Limites du CNN « classique »

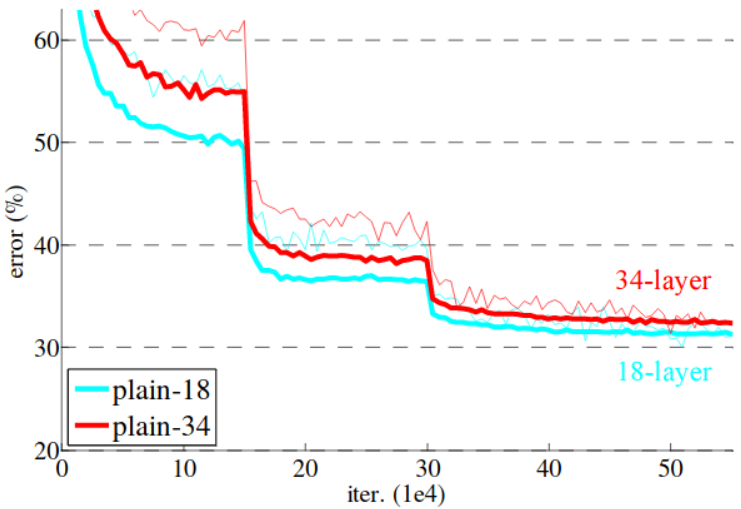


He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

I)

Limites du CNN « classique »

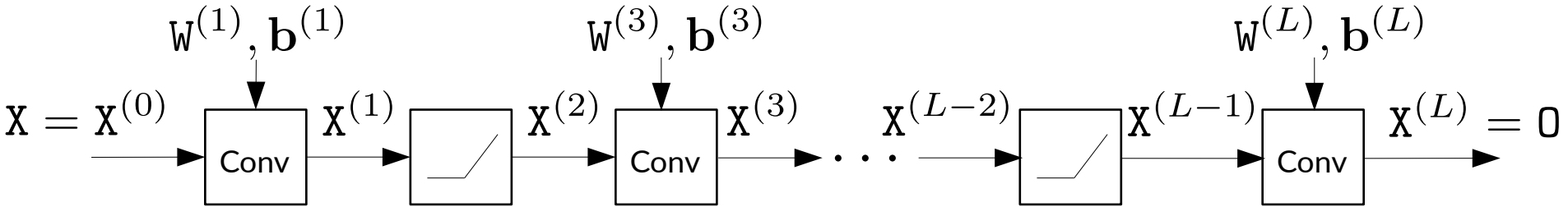


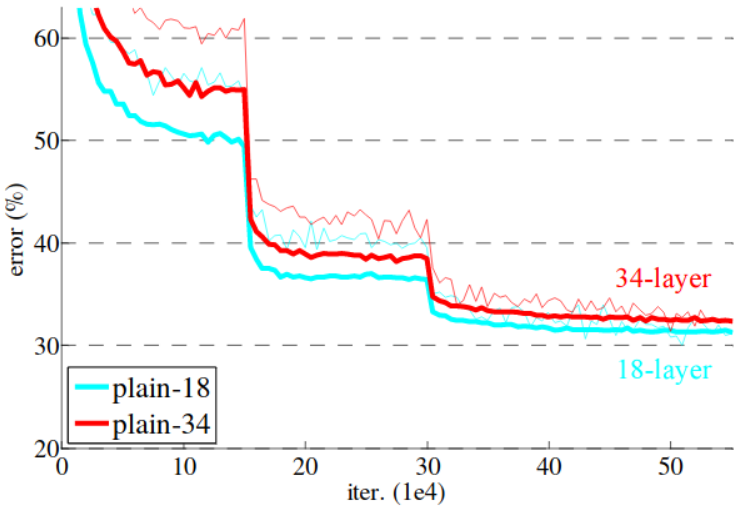


He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

I)

Limites du CNN « classique »





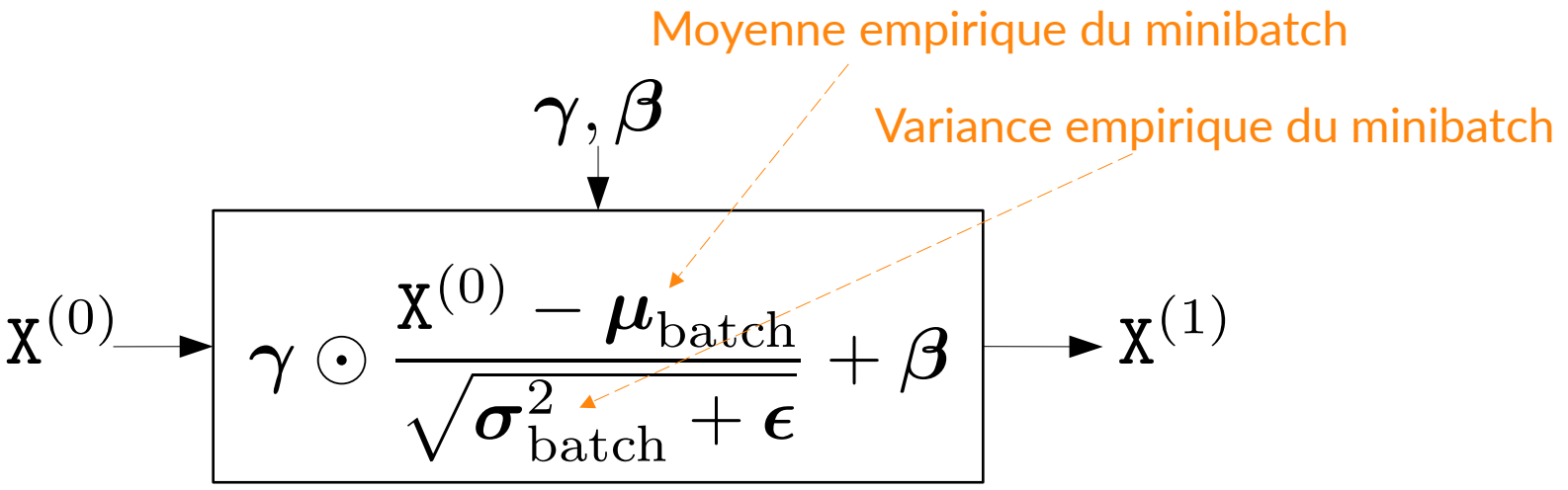
Ingrédient limitant les performances

Architecture du CNN

He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

I)

Ingrédient 1 : Couche de “Batch Normalization”



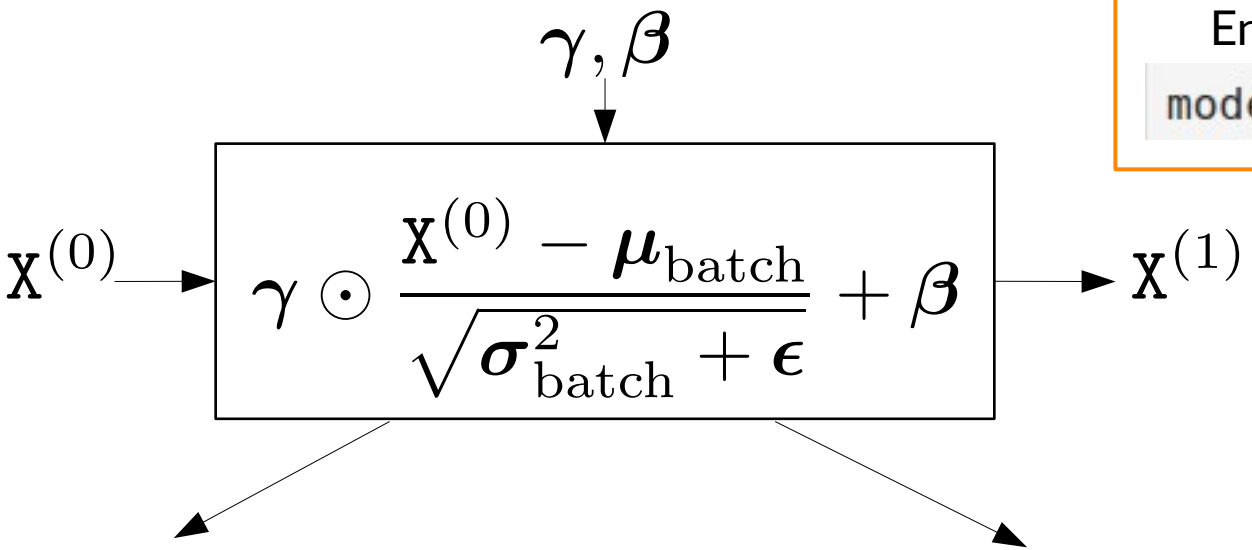
Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.

I)

Ingrédient 1 : Couche de “Batch Normalization”

En PyTorch

```
model.train()
```



$$\hat{\mu} = 0.1 * \mu_{\text{batch}} + 0.9 * \hat{\mu}$$

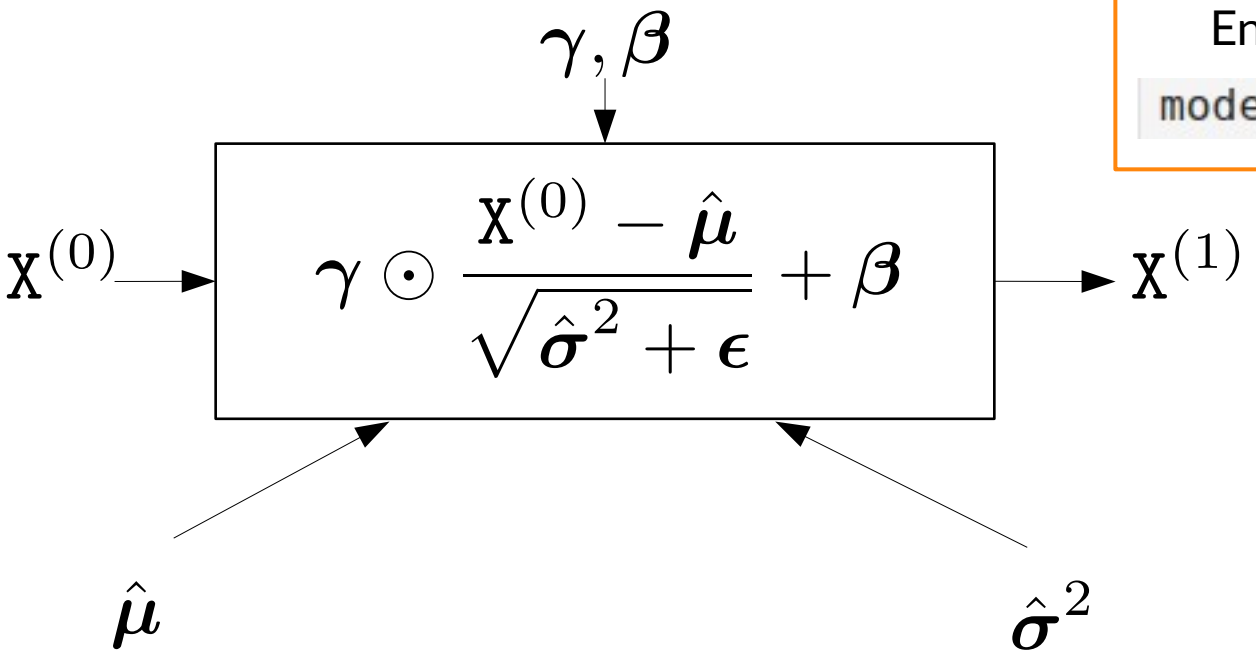
$$\hat{\sigma}^2 = 0.1 * \sigma^2_{\text{batch}} + 0.9 * \hat{\sigma}^2$$

Fonctionnement de la couche BN **à l’entraînement.**

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.

I)

Ingrédient 1 : Couche de “Batch Normalization”



En PyTorch

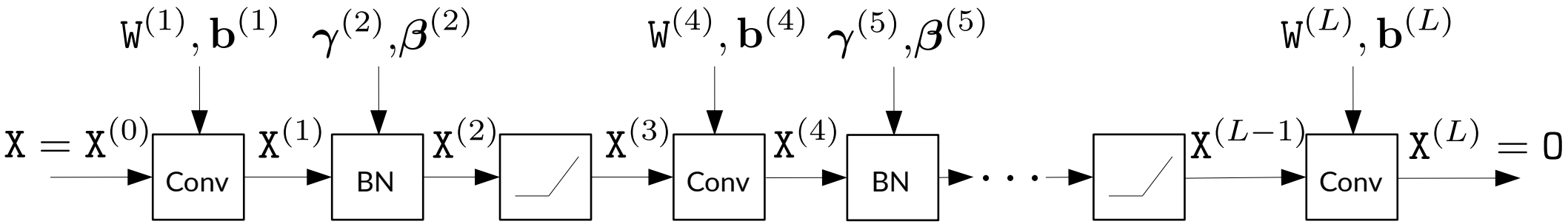
```
model.eval()
```

Fonctionnement de la couche BN **à l’évaluation.**

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.

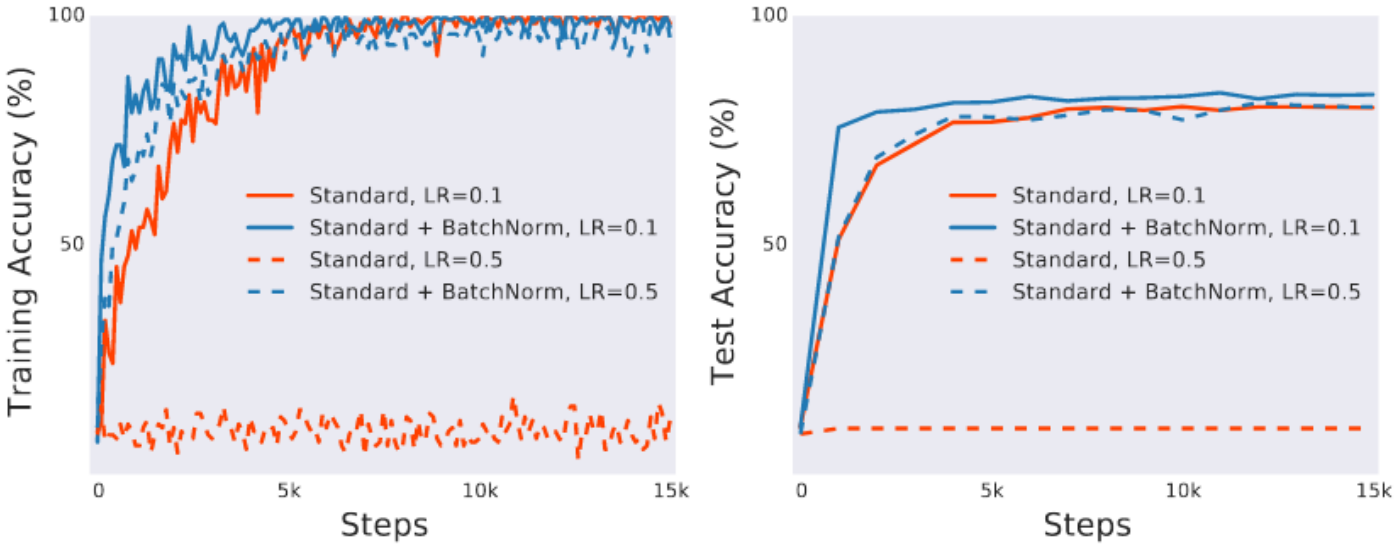
I)

CNN + BN


$\mathbb{W}^{(1)}, \mathbf{b}^{(1)}$
$\boldsymbol{\gamma}^{(2)}, \boldsymbol{\beta}^{(2)}$
$\mathbb{W}^{(4)}, \mathbf{b}^{(4)}$
$\boldsymbol{\gamma}^{(5)}, \boldsymbol{\beta}^{(5)}$
$\mathbb{W}^{(L)}, \mathbf{b}^{(L)}$
$\mathbb{X} = \mathbb{X}^{(0)}$
Conv
$\mathbb{X}^{(1)}$
BN
$\mathbb{X}^{(2)}$
$\mathbb{X}^{(3)}$
Conv
$\mathbb{X}^{(4)}$
BN
$\mathbb{X}^{(L-1)}$
Conv
$\mathbb{X}^{(L)} = \mathbb{O}$

I)

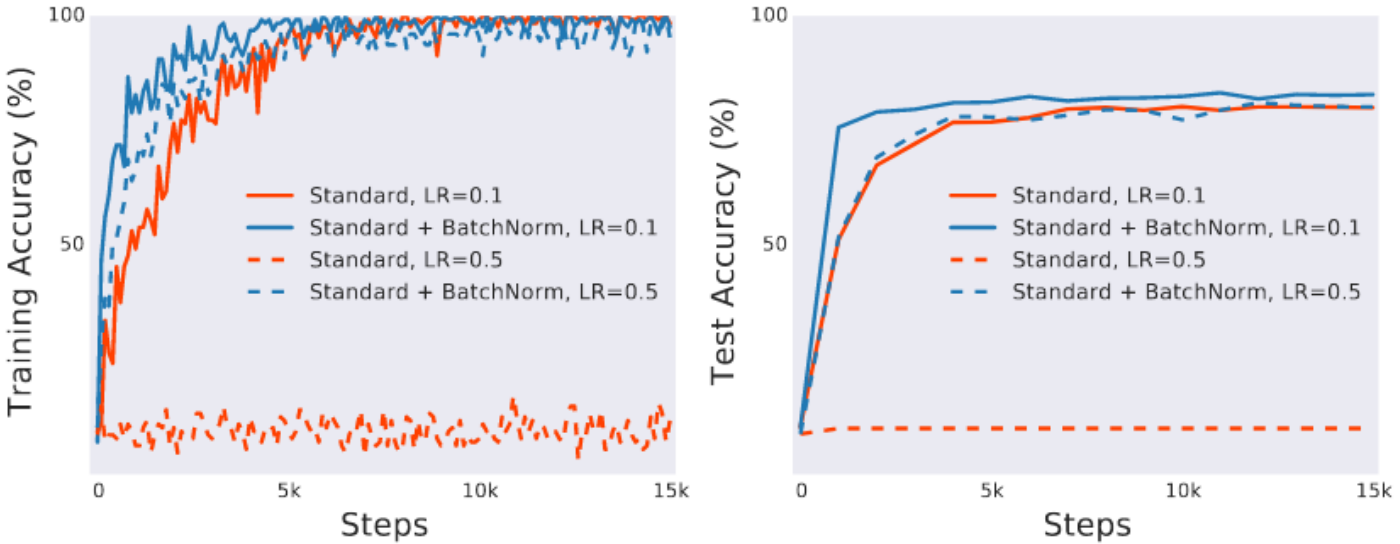
CNN + BN (suite)



Santurkar, Shibani, et al. "How does batch normalization help optimization?." Advances in neural information processing systems. 2018.

I)

CNN + BN (suite)



Rend le problème d'optimisation plus « lisse » :

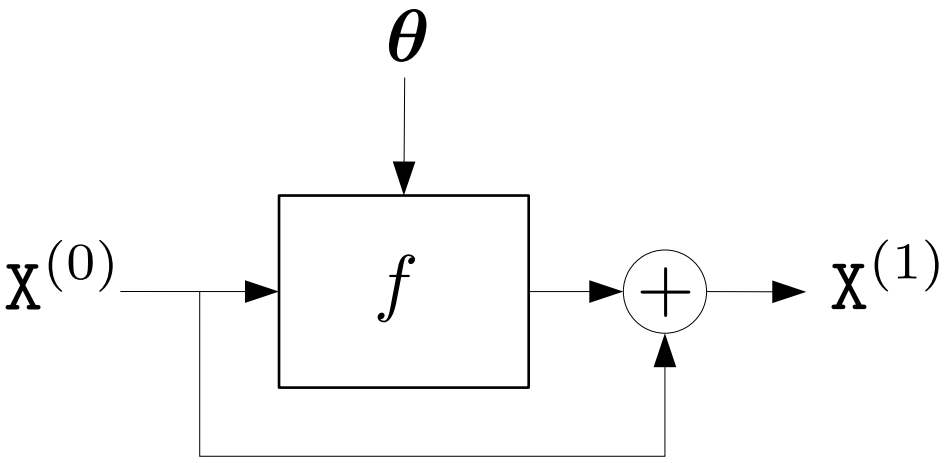
→ Initialisation des paramètres moins critique

→ Possibilité d'utilisation d'un plus grand pas d'apprentissage → accélération de l'entraînement

Santurkar, Shibani, et al. "How does batch normalization help optimization?." Advances in neural information processing systems. 2018.

I)

Ingrédient 2 : Connexion résiduelle

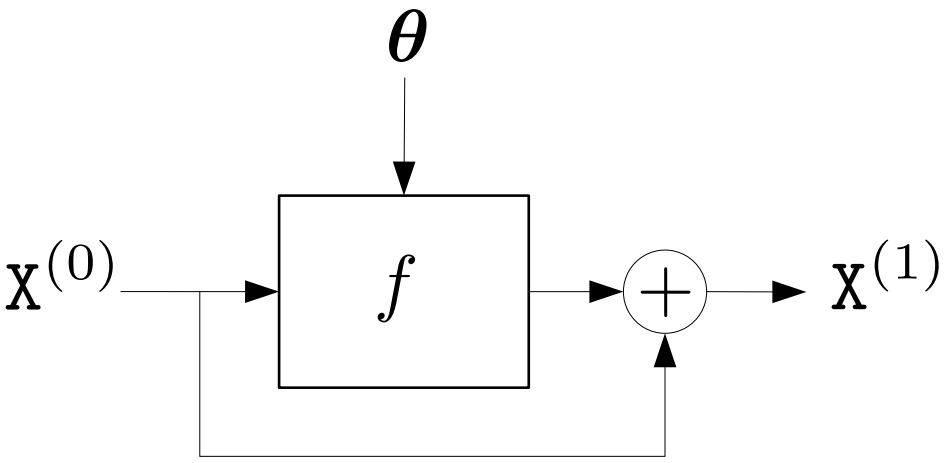


$$\mathrm{x}^{(1)} = \mathrm{x}^{(0)} + f(\mathrm{x}^{(0)}; \boldsymbol{\theta})$$

He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

I)

Ingrédient 2 : Connexion résiduelle



$$\mathrm{x}^{(1)} = \mathrm{x}^{(0)} + f(\mathrm{x}^{(0)}; \boldsymbol{\theta})$$

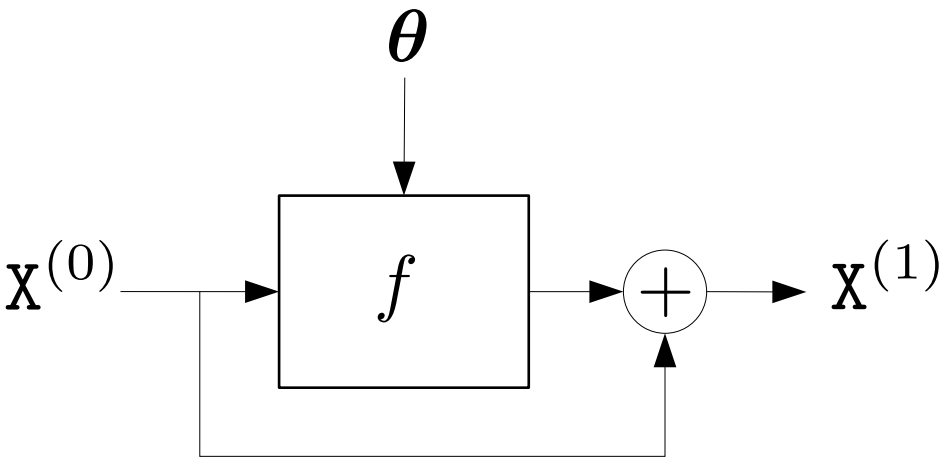
Rend la fonction plus « linéaire » :

→ Réduit sa « capacité » → augmentation du nombre de couches pour un même résultat

He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

I)

Ingrédient 2 : Connexion résiduelle



$$\mathrm{x}^{(1)} = \mathrm{x}^{(0)} + f(\mathrm{x}^{(0)}; \boldsymbol{\theta})$$

$$\frac{\partial \mathrm{x}^{(1)}}{\partial \mathrm{x}^{(0)}} = \mathrm{I} + \frac{\partial f(\mathrm{x}^{(0)}; \boldsymbol{\theta})}{\partial \mathrm{x}^{(0)}}$$

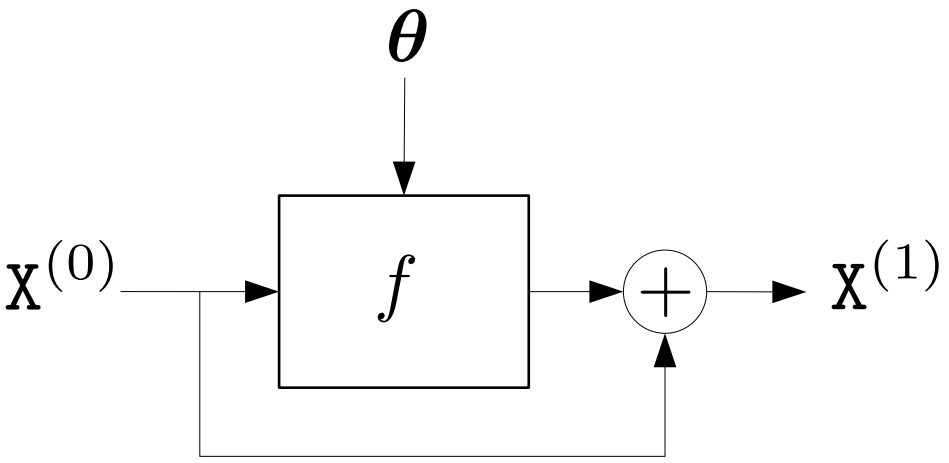
Rend la fonction plus « linéaire » :

→ Réduit sa « capacité » → augmentation du nombre de couches pour un même résultat

He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

I)

Ingrédient 2 : Connexion résiduelle



$$\mathrm{x}^{(1)} = \mathrm{x}^{(0)} + f(\mathrm{x}^{(0)}; \boldsymbol{\theta})$$

$$\frac{\partial \mathrm{x}^{(1)}}{\partial \mathrm{x}^{(0)}} = \mathrm{I} + \frac{\partial f(\mathrm{x}^{(0)}; \boldsymbol{\theta})}{\partial \mathrm{x}^{(0)}}$$

Rend la fonction plus « linéaire » :

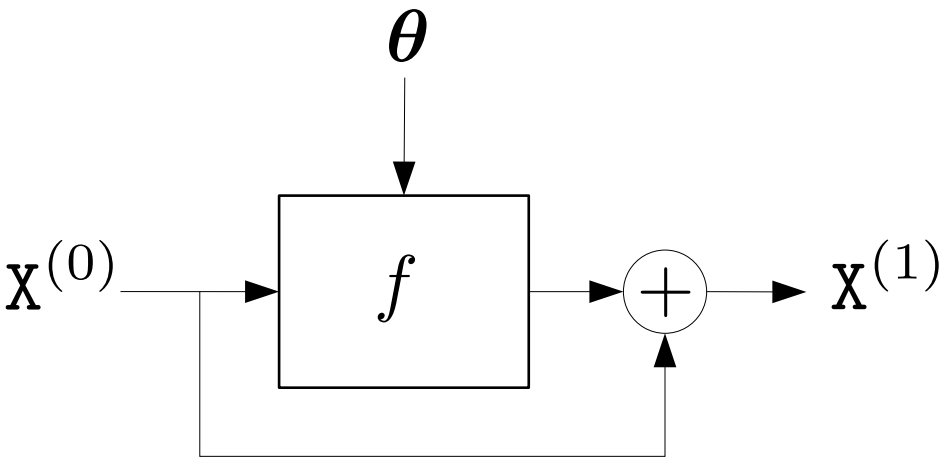
→ Réduit sa « capacité » → augmentation du nombre de couches pour un même résultat

Même si ce gradient « s'évanouit » (vaut presque zéro), il reste l'identité grâce à la connexion résiduelle

He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

I)

Ingrédient 2 : Connexion résiduelle



$$\mathrm{x}^{(1)} = \mathrm{x}^{(0)} + f(\mathrm{x}^{(0)}; \boldsymbol{\theta})$$

$$\frac{\partial \mathrm{x}^{(1)}}{\partial \mathrm{x}^{(0)}} = \mathrm{I} + \frac{\partial f(\mathrm{x}^{(0)}; \boldsymbol{\theta})}{\partial \mathrm{x}^{(0)}}$$

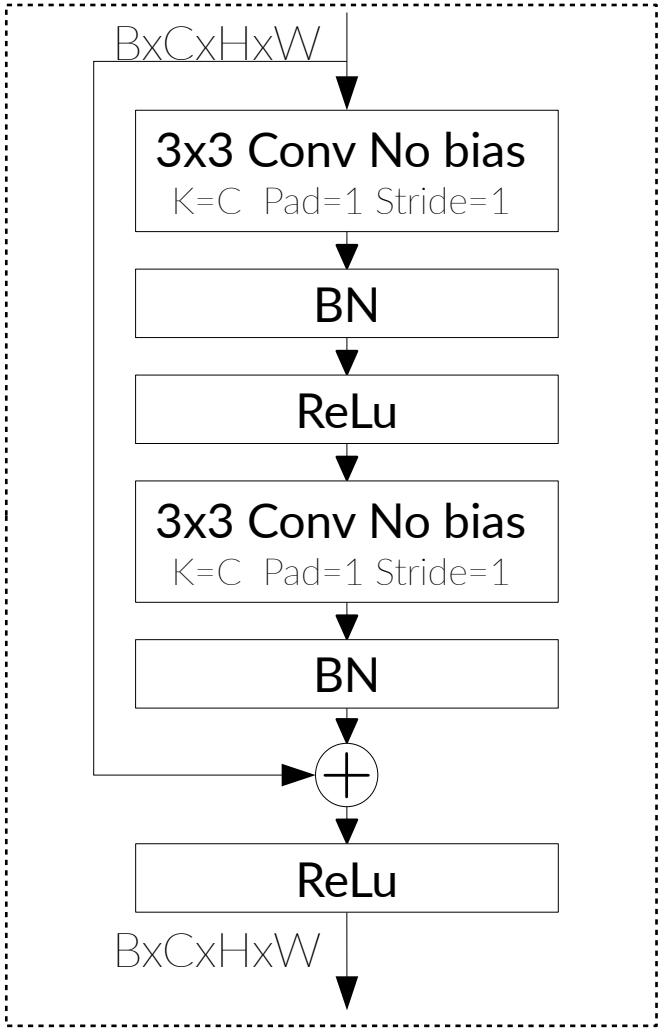
Rend la fonction plus « linéaire » :

→ Réduit sa « capacité » → augmentation du nombre de couches pour un même résultat

→ Facilite la propagation du gradient → plus de couches conduit à de meilleurs résultats (en théorie)

He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

I)



ResBlock A

Transforme le tenseur d'entrée

- en préservant la résolution
- en préservant le nombre de canaux

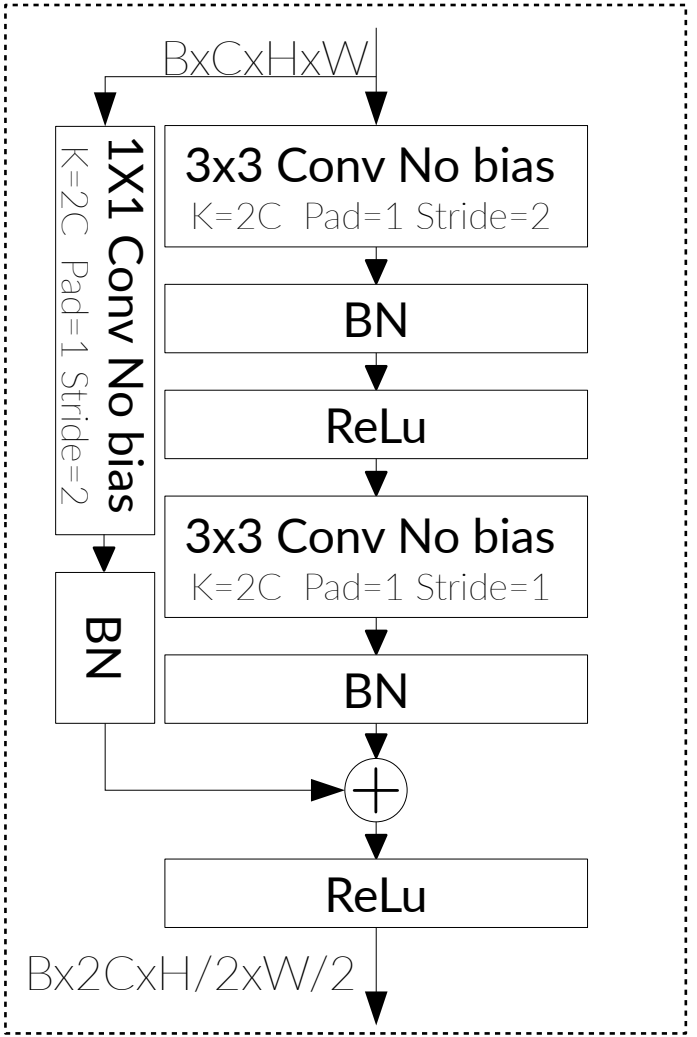
He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

I)

ResBlock B

Transforme le tenseur d'entrée

- en divisant la résolution par 2
- en augmentant le nombre de canaux par 2



He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

I)

ResNet 18

$$\arg\max(\mathbf{s}) = 748 \doteq \text{"raie"}$$

Passage en grande dimension

3x224x224

64x112x112

7x7 Conv No bias
K=64 Pad=3 Stride=2

64x112x112

BN

64x112x112

ReLu

64x112x112

3x3 MaxPool
Pad=1 Stride=2

64x56x56

ResBlock A

64x56x56

ResBlock A

64x56x56

ResBlock B

128x28x28

ResBlock A

128x28x28

ResBlock B

256x14x14

ResBlock A

256x14x14

ResBlock B

512x7x7

ResBlock A

512x7x7

7x7 Average Pool

512x1

1000x512 FC

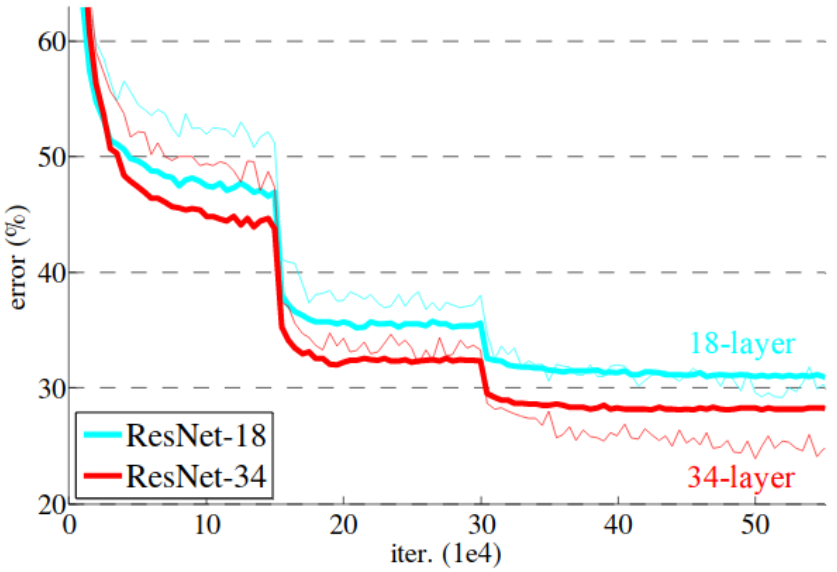
1000x1

$\mathbf{s}$

He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

I)

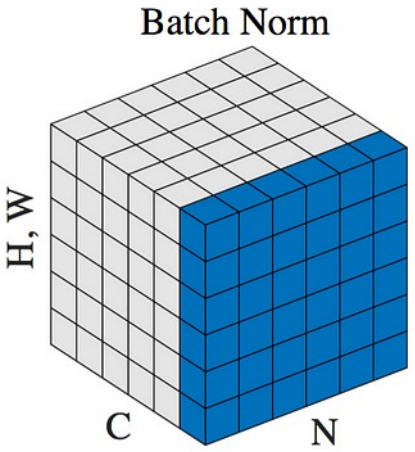
ResNet 18 < ResNet 34



He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

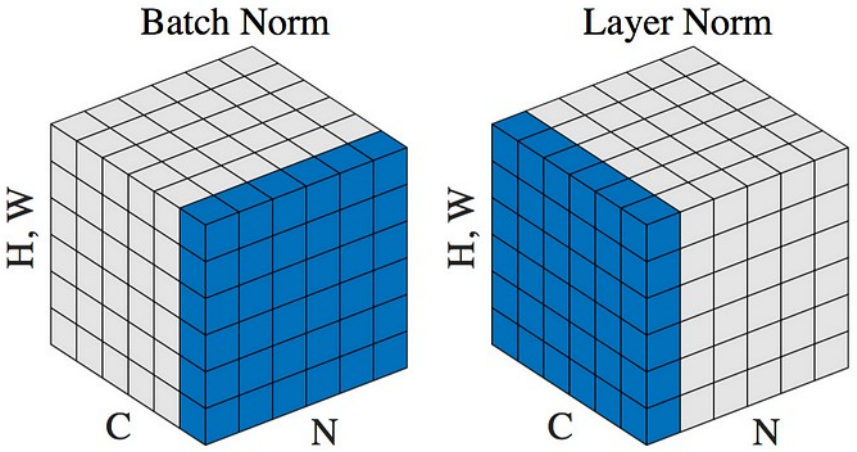
I)

Différentes couches de normalisation



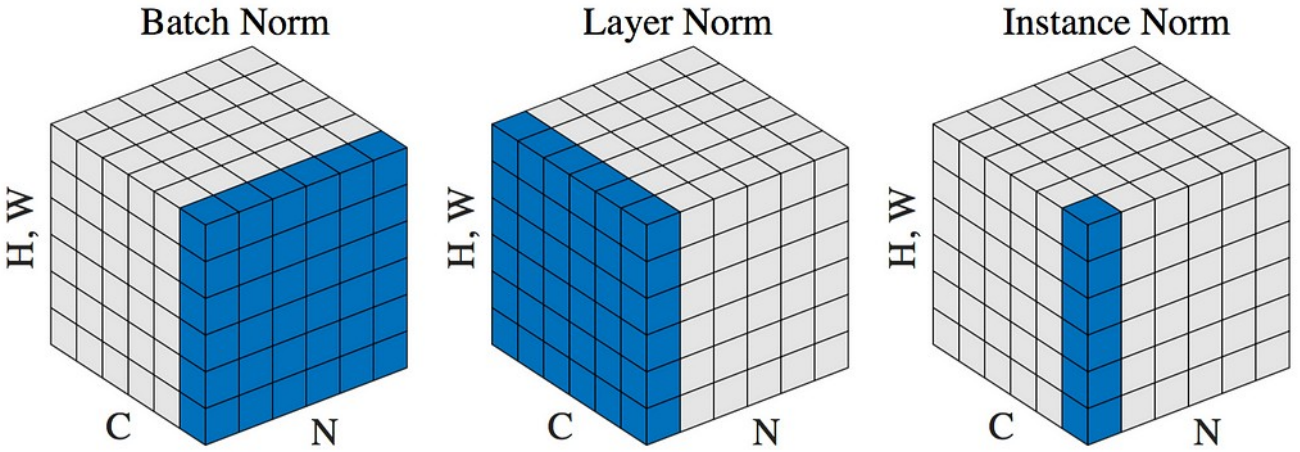
I)

Différentes couches de normalisation



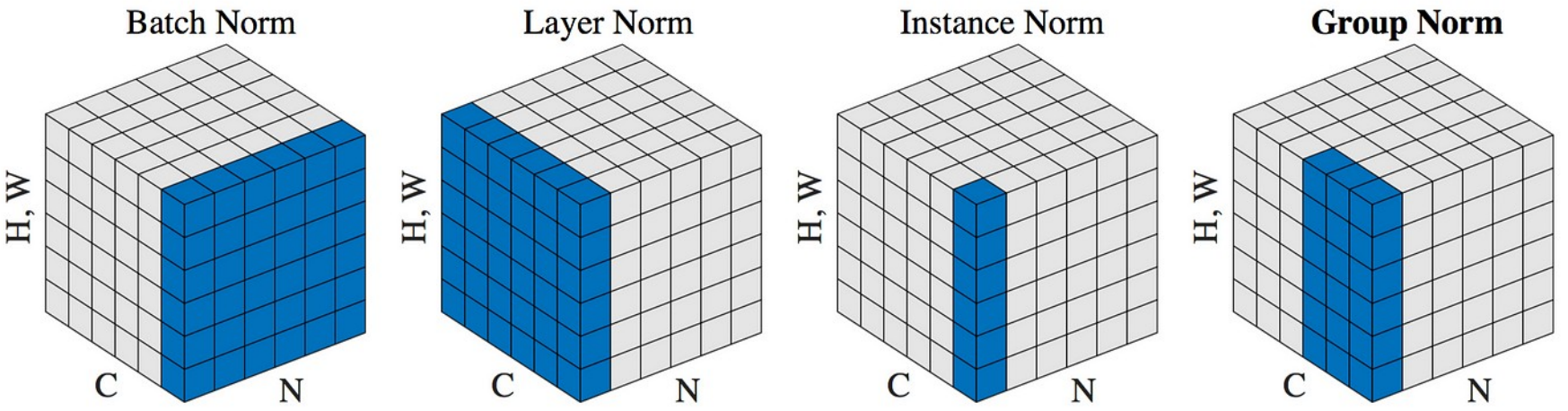
I)

Différentes couches de normalisation



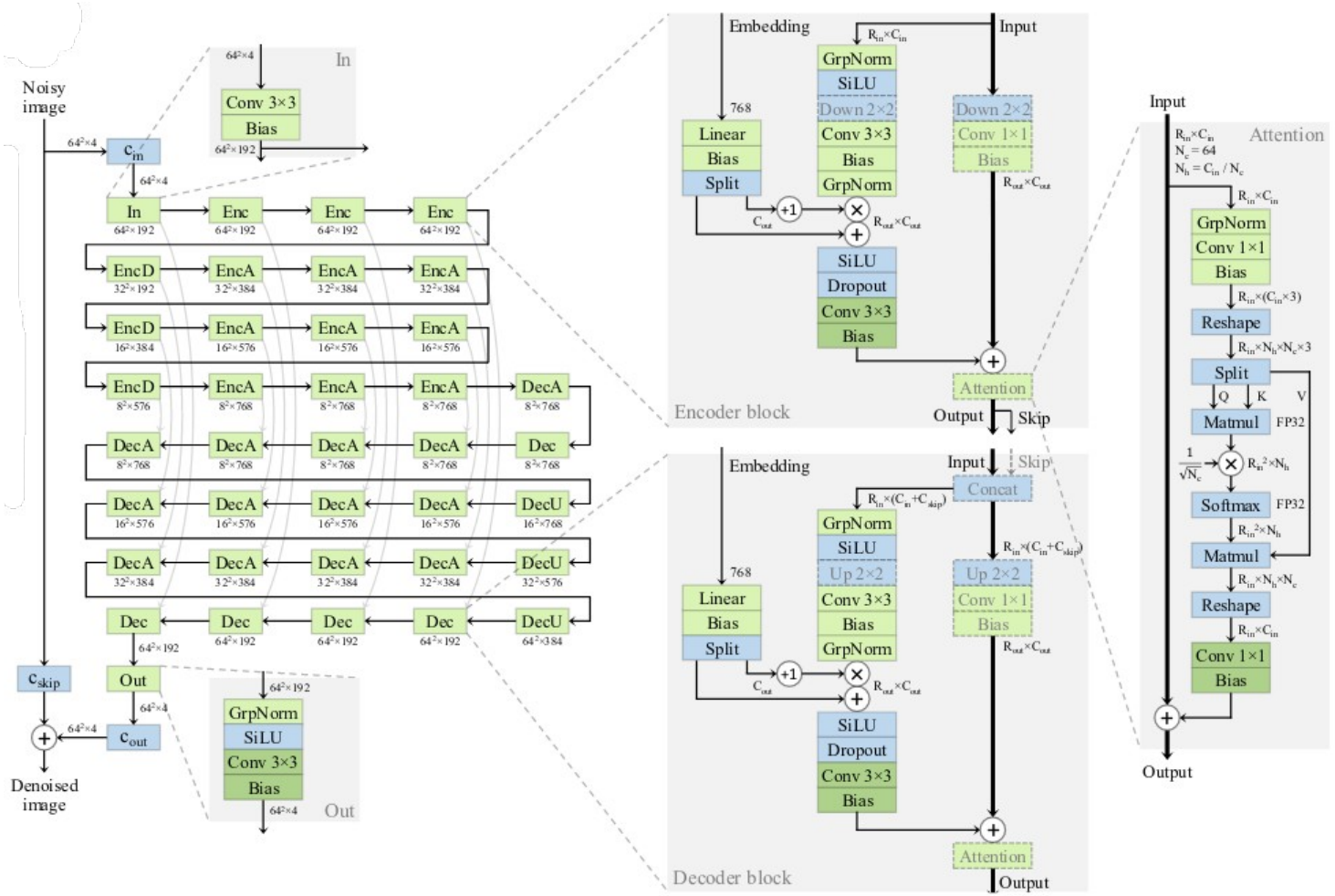
I)

Différentes couches de normalisation



I)

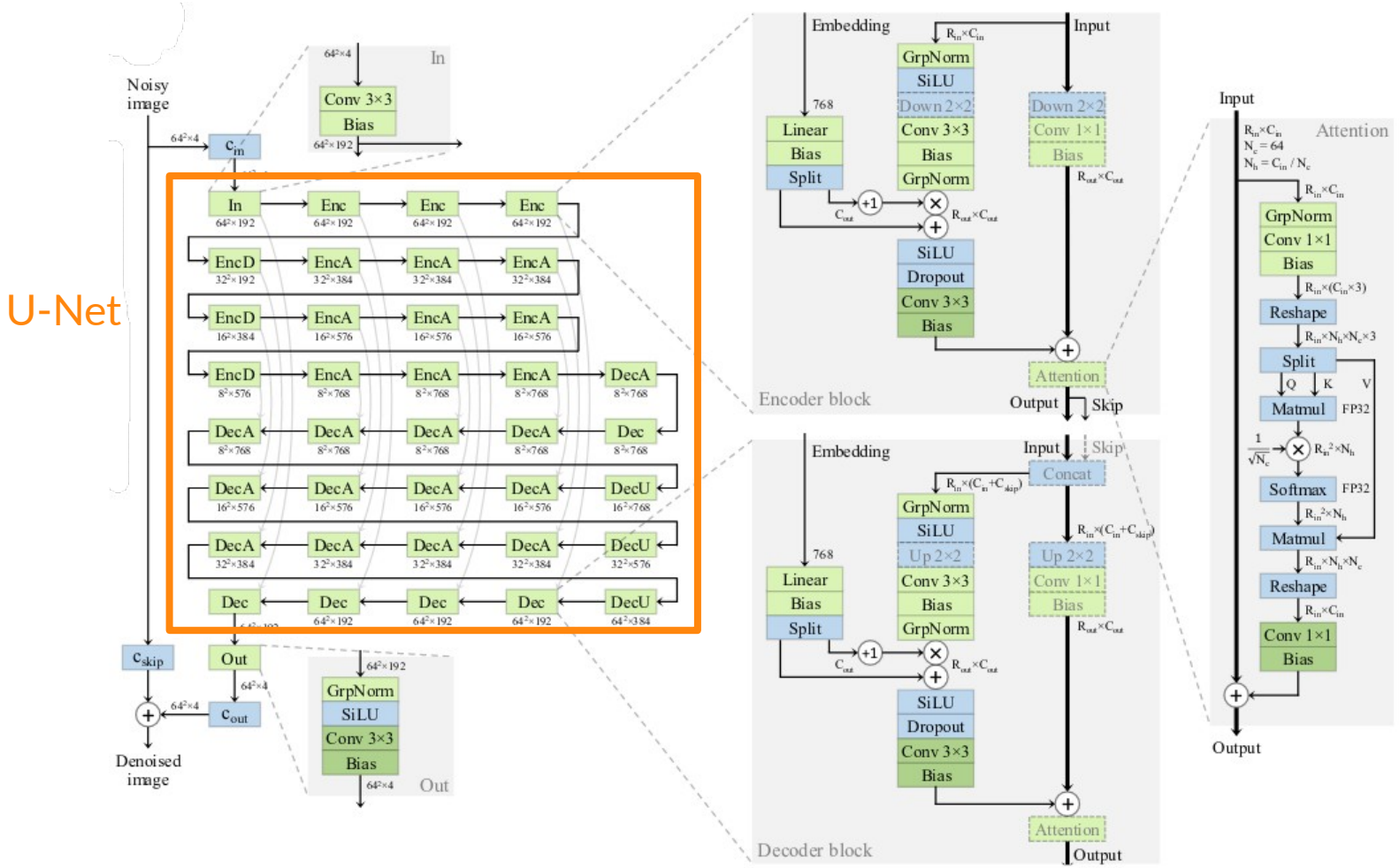
En 2024 : U-Net + ResBlock toujours là !



Karras et al. (2024). Analyzing and Improving the Training Dynamics of Diffusion Models arXiv preprint arXiv:2312.02696.

I)

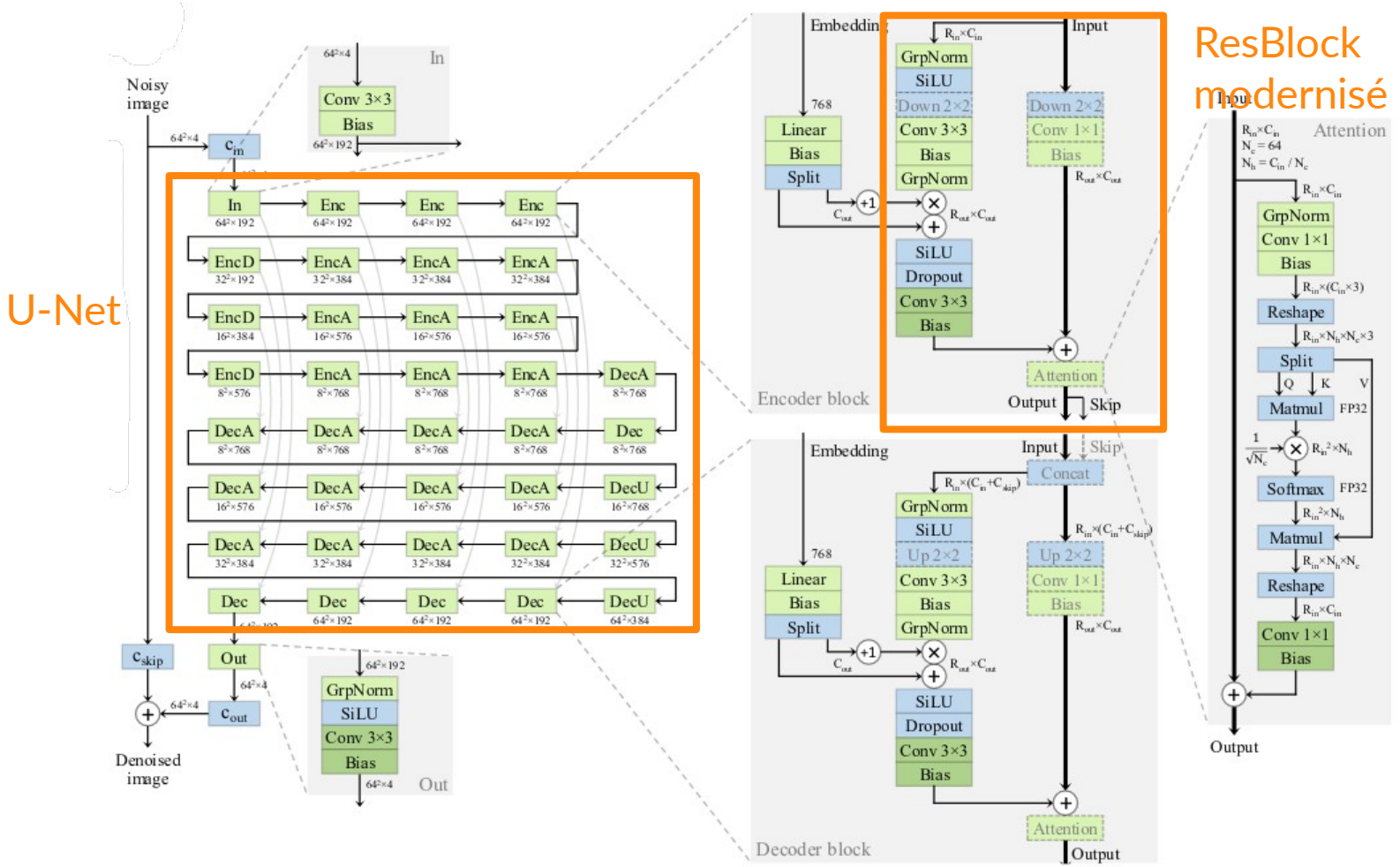
En 2024 : U-Net + ResBlock toujours là !



Karras et al. (2024). Analyzing and Improving the Training Dynamics of Diffusion Models arXiv preprint arXiv:2312.02696.

I)

En 2024 : U-Net + ResBlock toujours là !



Karras et al. (2024). Analyzing and Improving the Training Dynamics of Diffusion Models arXiv preprint arXiv:2312.02696.

II) Modèles de fondation

Résumé des ingrédients du « Deep Learning »

1) **Grande** base de données étiquetées

2) « Bonne » architecture de réseau de neurones profond

- « Perceptron » multicouche, Réseau de neurones à convolution, Transformer
- Optimisation par descente de gradient stochastisque (AdamW, etc.)

3) Grande capacité de calculs en parallèle (GPUs)

II)

Comment faire avec une petite base de données étiquetées ?

Exemple : détection du frelon asiatique

Présence (1042 images)

Absence (1844 images)

Ingrédient limitant les performances

↓

Taille de la base de données étiquetées

II)

Comment faire avec une petite base de données étiquetées ?

Exemple : détection du frelon asiatique

Présence (1042 images)

Absence (1844 images)

Ingrédient limitant les performances

↓

Taille de la base de données étiquetées

Solution

1) Récupérer un modèle de fondation

2) Spécialiser ce modèle de fondation sur sa petite base de données étiquetées

II)

Modèle de fondation « historique »

Historiquement (~ à partir de 2012), un modèle de fondation (le terme « foundation model » n'est introduit qu'en 2021) est :

II)

Modèle de fondation « historique »

Historiquement (~ à partir de 2012), un modèle de fondation (le terme « foundation model » n'est introduit qu'en 2021) est :

- un CNN (e.g ResNet)

II)

Modèle de fondation « historique »

Historiquement (~ à partir de 2012), un modèle de fondation (le terme « foundation model » n'est introduit qu'en 2021) est :

- un CNN (e.g ResNet)
- qui est **pré-entraîné =** entraîné sur ImageNet1k

Rappel : ImageNet1k = 1.2M d'images étiquetées sur 1000 classes qui représentent une grande diversité d'images **issues de notre monde**

II) Exemple de spécialisation (« fine-tuning ») d'un ResNet 18 pré-entraîné sur ImageNet1k

3x224x224

7x7 Conv No bias
K=64 Pad=3 Stride=2

64x112x112

BN

64x112x112

ReLu

64x112x112

3x3 MaxPool
Pad=1 Stride=2

64x56x56

ResBlock A

64x56x56

ResBlock A

64x56x56

ResBlock B

128x28x28

ResBlock A

128x28x28

ResBlock B

256x14x14

ResBlock A

256x14x14

ResBlock B

512x7x7

ResBlock A

512x7x7

7x7 Average Pool

512x1

1000x512 FC

1000x1

II) Exemple de spécialisation (« fine-tuning ») d'un ResNet 18 pré-entraîné sur ImageNet1k

Problème : on veut prédire 2 scores (présence de frélon, absence de frelon), pas 1000 scores...

3x224x224
7x7 Conv No bias
K=64 Pad=3 Stride=2
64x112x112
BN
64x112x112
ReLu
64x112x112
3x3 MaxPool
Pad=1 Stride=2
64x56x56
ResBlock A
64x56x56
ResBlock A
64x56x56
ResBlock B
128x28x28
ResBlock A
128x28x28
ResBlock B
256x14x14
ResBlock A
256x14x14
ResBlock B
512x7x7
ResBlock A
512x7x7
7x7 Average Pool
512x1
1000x512 FC
1000x1

II) Exemple de spécialisation (« fine-tuning ») d'un ResNet 18 pré-entraîné sur ImageNet1k

3x224x224

7x7 Conv No bias
K=64 Pad=3 Stride=2

64x112x112

BN

64x112x112

ReLu

64x112x112

3x3 MaxPool
Pad=1 Stride=2

64x56x56

ResBlock A

64x56x56

ResBlock A

64x56x56

ResBlock B

128x28x28

Nouvelle couche FC !

2x512 FC

2x1

7x7 Average Pool

512x1

ResBlock A

512x7x7

ResBlock B

512x7x7

ResBlock A

256x14x14

ResBlock B

256x14x14

ResBlock A

128x28x28

II) Exemple de spécialisation (« fine-tuning ») d'un ResNet 18 pré-entraîné sur ImageNet1k

$$\arg\max(\mathbf{s}) = 2 \doteq \text{"Présence"}$$

3x224x224	7x7 Conv No bias K=64 Pad=3 Stride=2	
64x112x112	BN	
64x112x112	ReLu	
64x112x112	3x3 MaxPool Pad=1 Stride=2	
64x56x56	ResBlock A	
64x56x56	ResBlock A	
64x56x56	ResBlock B	
128x28x28	ResBlock A	128x28x28
	ResBlock B	256x14x14
	ResBlock A	256x14x14
	ResBlock B	512x7x7
	ResBlock A	512x7x7
	7x7 Average Pool	512x1
	2x512 FC	2x1

Paramètres du ResNet18 pré-entraîné sur ImageNet

Initialisation classique (aléatoire)

II) Exemple de spécialisation (« fine-tuning ») d'un ResNet 18 pré-entraîné sur ImageNet1k

Lors du pré-entraînement sur ImageNet1k :

- Les images sont normalisées (comme on l'a fait en TP sur MNIST)

$$\mathtt{X}_{\mathrm{norm}} = \frac{\mathtt{X} - \boldsymbol{\mu}_{\mathrm{ImageNet1k}}}{\boldsymbol{\sigma}_{\mathrm{ImageNet1k}}}$$

II) Exemple de spécialisation (« fine-tuning ») d'un ResNet 18 pré-entraîné sur ImageNet1k

Lors du pré-entraînement sur ImageNet1k :

- Les images sont normalisées (comme on l'a fait en TP sur MNIST)

$$\mathtt{X}_{\text{norm}} = \frac{\mathtt{X} - \boldsymbol{\mu}_{\text{ImageNet1k}}}{\boldsymbol{\sigma}_{\text{ImageNet1k}}}$$

- Pour utiliser le réseau, il faut appliquer cette **même** normalisation à nos images !

II) Exemple de spécialisation (« fine-tuning ») d'un ResNet 18 pré-entraîné sur ImageNet1k

Image « normalisée »

3x224x224

7x7 Conv No bias
K=64 Pad=3 Stride=2

64x112x112

BN

64x112x112

ReLu

64x112x112

3x3 MaxPool
Pad=1 Stride=2

64x56x56

ResBlock A

64x56x56

ResBlock A

64x56x56

ResBlock B

128x28x28

ResBlock A

128x28x28

ResBlock B

256x14x14

ResBlock A

256x14x14

ResBlock B

512x7x7

ResBlock A

512x7x7

7x7 Average Pool

512x1

2x512 FC

2x1

$$\arg\max(\mathbf{s}) = 2 \doteq \text{"Présence"}$$

Paramètres du ResNet18 pré-entraîné sur ImageNet

Initialisation classique (aléatoire)

II)

Modèles de fondation « modernes »

Pré-entraînement sur une **très** grande base de données.

Exemple : ~2012 ImageNet 1k = 1.2x10^6 images étiquetées
~2022 LAION-5B = 5x10^9 images étiquetées

II)

Modèles de fondation « modernes »

Pré-entraînement sur une **très** grande base de données.

Exemple : ~2012 ImageNet 1k = 1.2x10^6 images étiquetées
~2022 LAION-5B = 5x10^9 images étiquetées

Rendu possible grâce à :

- l'augmentation continue de la puissance des cartes graphiques
- l'apparition des couches d'attention (« Transformer »)

II)

Modèles de fondation « modernes »

Pré-entraînement sur une **très** grande base de données.

Exemple : ~2012 ImageNet 1k = 1.2x10^6 images étiquetées
~2022 LAION-5B = 5x10^9 images étiquetées

Rendu possible grâce à :

- l'augmentation continue de la puissance des cartes graphiques
- l'apparition des couches d'attention (« Transformer »)

Apparition de modèles de fondation dans un grand nombre de domaines :

- Reconstruction 3D, Segmentation d'image, Imagerie médicale etc.

II)

Modèles de fondation « modernes »

Pré-entraînement sur une **très** grande base de données.

Exemple : ~2012 ImageNet 1k = 1.2x10^6 images étiquetées
~2022 LAION-5B = 5x10^9 images étiquetées

Rendu possible grâce à :

- l'augmentation continue de la puissance des cartes graphiques
- l'apparition des couches d'attention (« Transformer »)

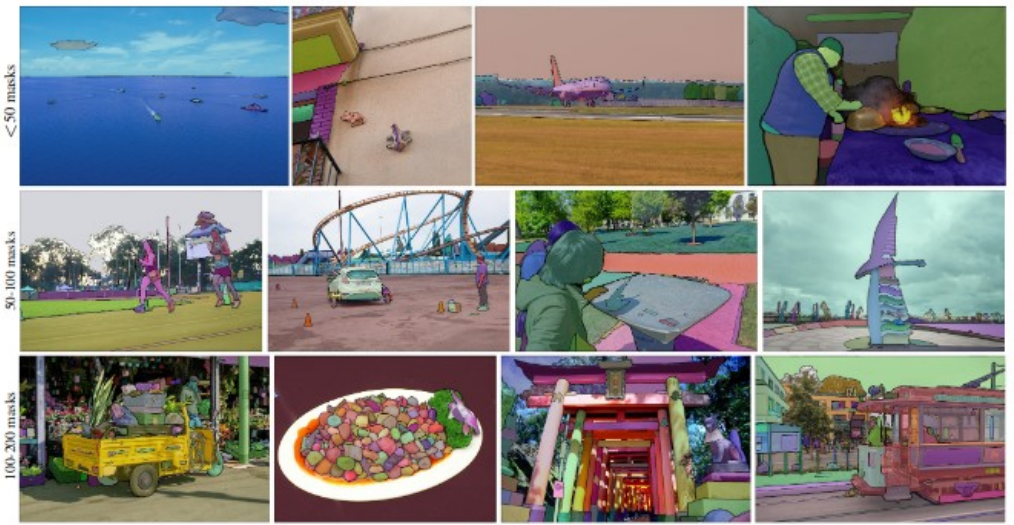
Apparition de modèles de fondation dans un grand nombre de domaines :

- Reconstruction 3D, Segmentation d'image, Imagerie médicale etc.
- Et bien-sûr les LLM (« Large Language Model ») avec GPT !

II) Exemple de modèle de fondation pour la segmentation d'images

-

« Segment Anything »



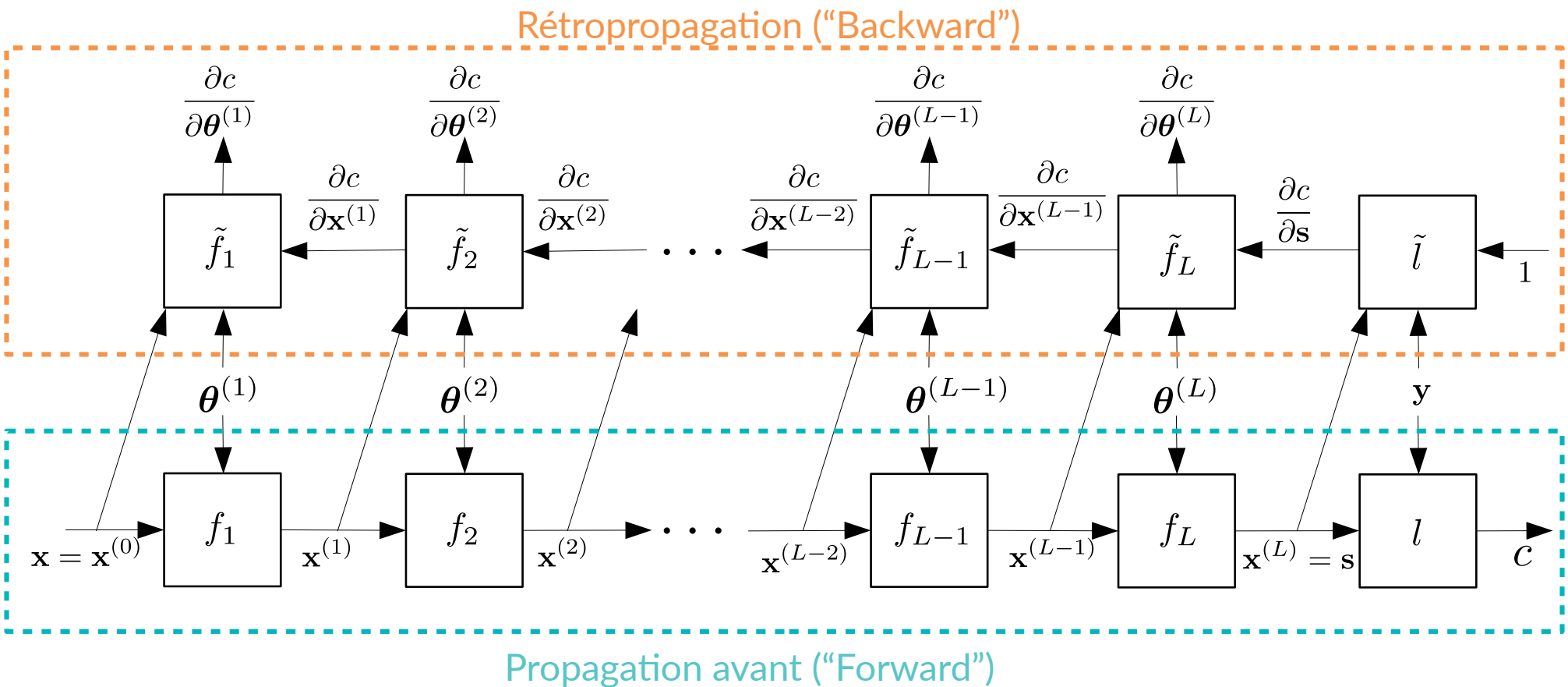
Entraînement sur « SA-1B » : une base de 11×10^6 images avec 1.1×10^9 masques de segmentation

Kirillov, Alexander, et al. "Segment Anything." CVPR 2023

III) « Gradient checkpointing »

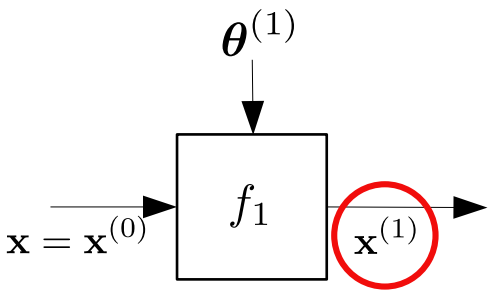
III)

Rétropropagation « classique »



III)

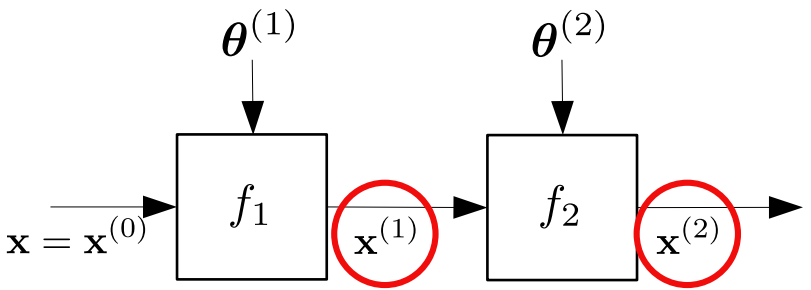
Rétropropagation « classique »



En mémoire

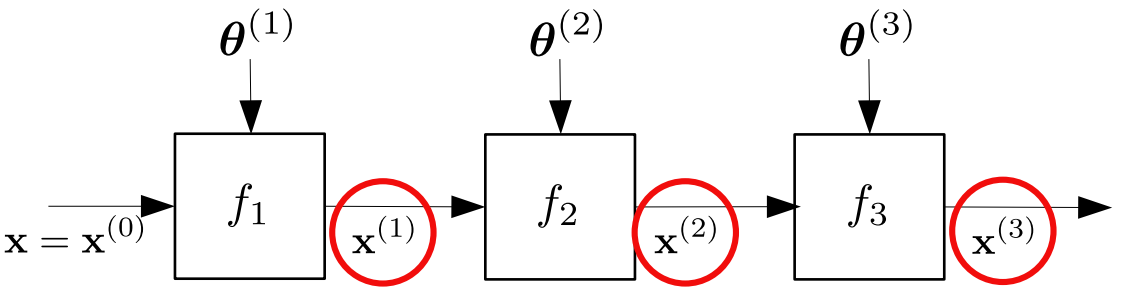
III)

Rétropropagation « classique »



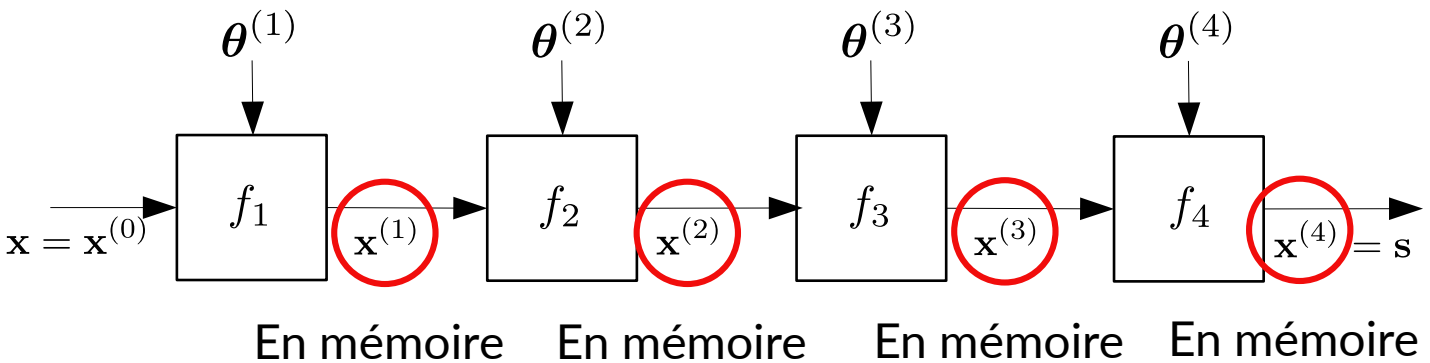
III)

Rétropropagation « classique »



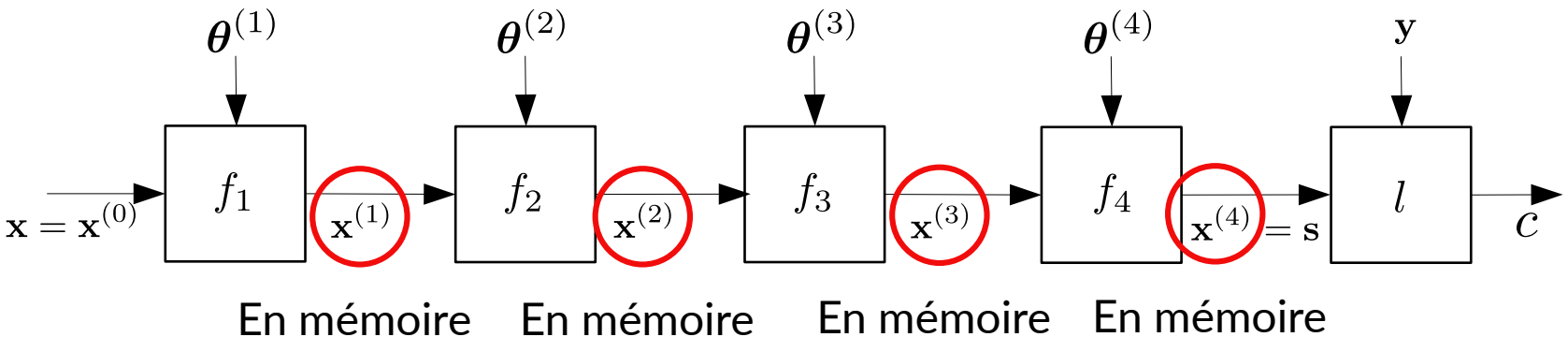
III)

Rétropropagation « classique »



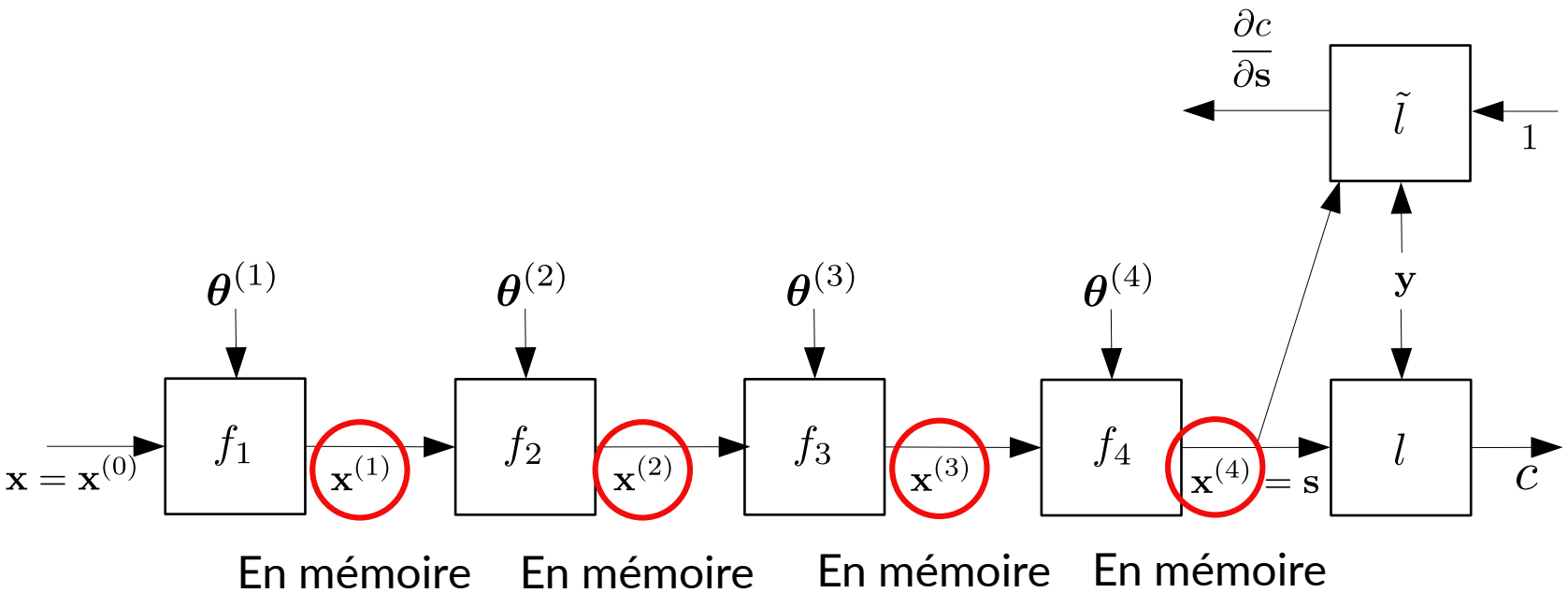
III)

Rétropropagation « classique »



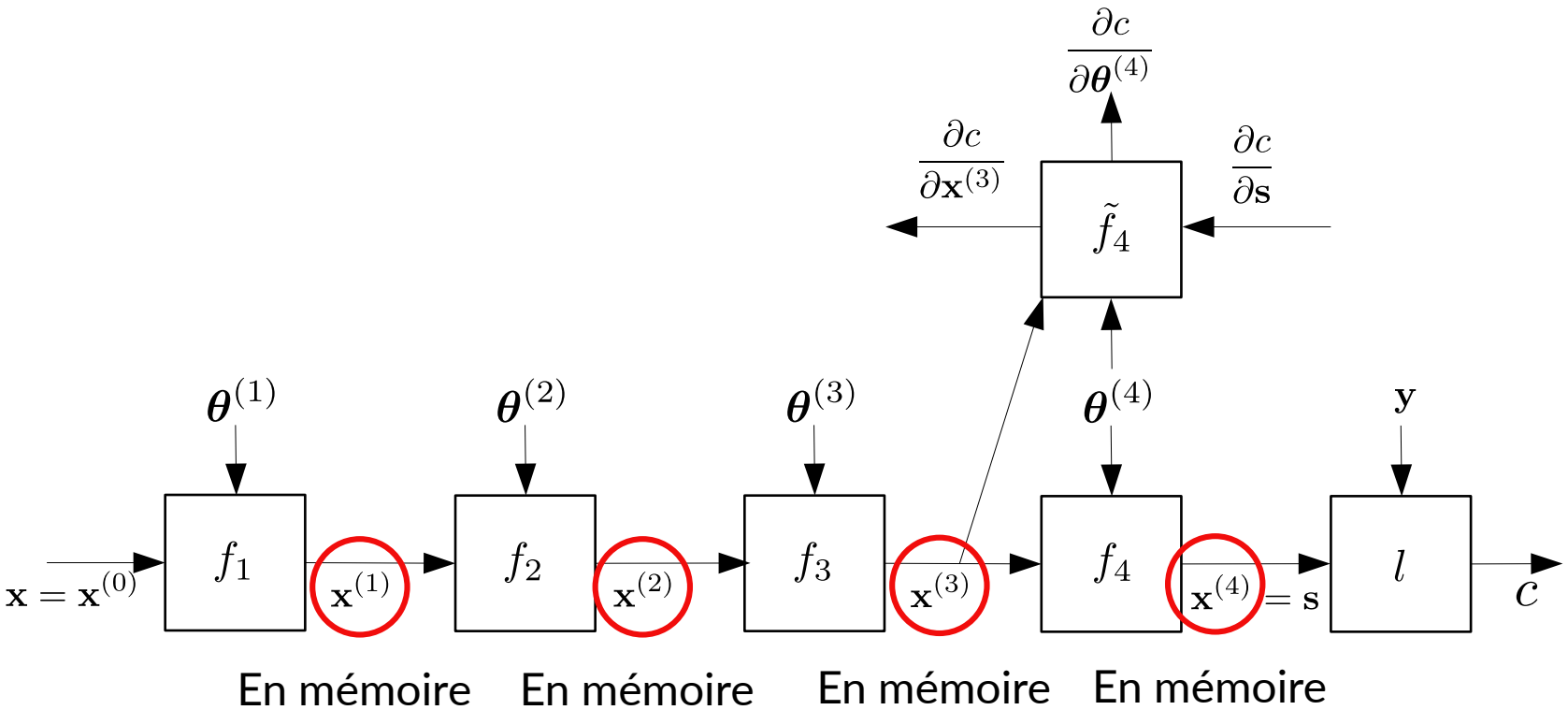
III)

Rétropropagation « classique »



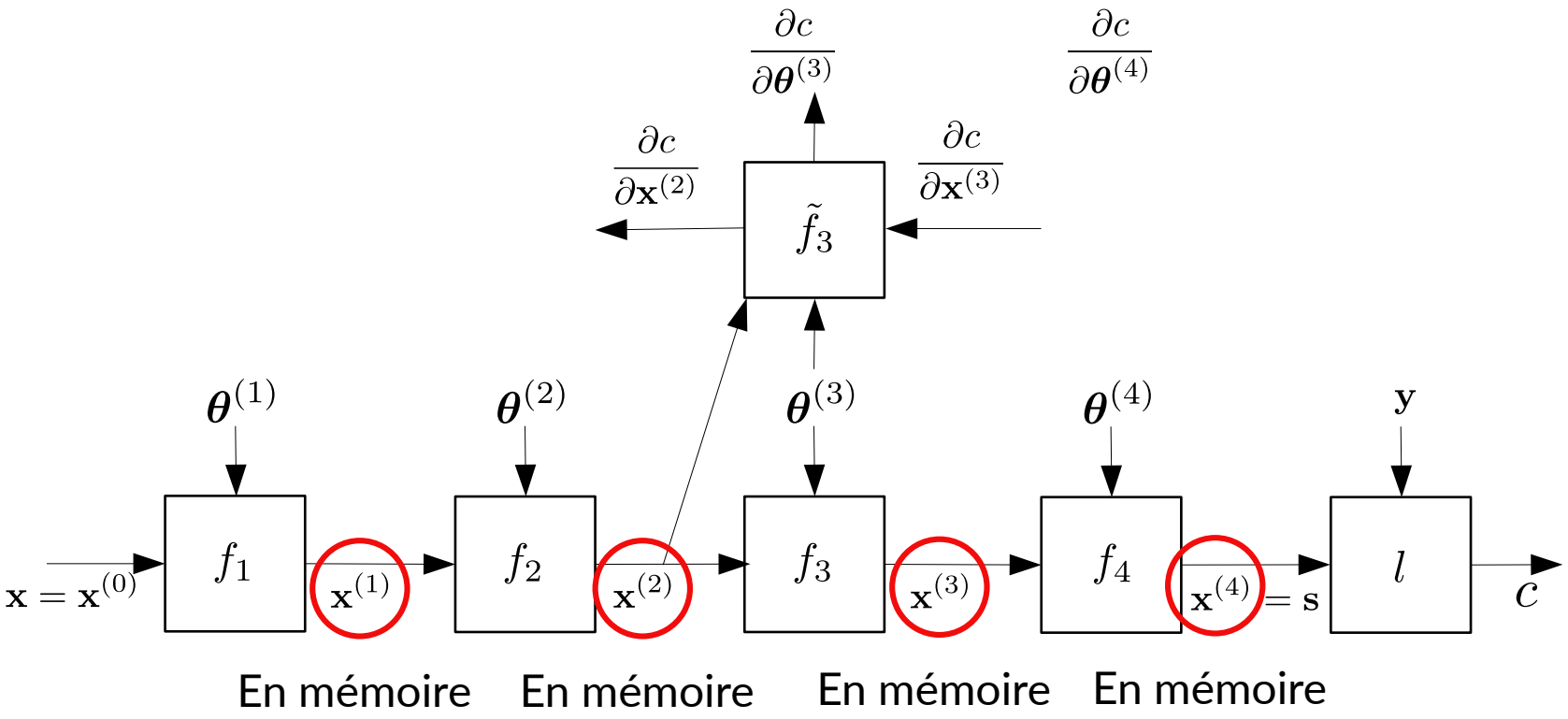
III)

Rétropropagation « classique »



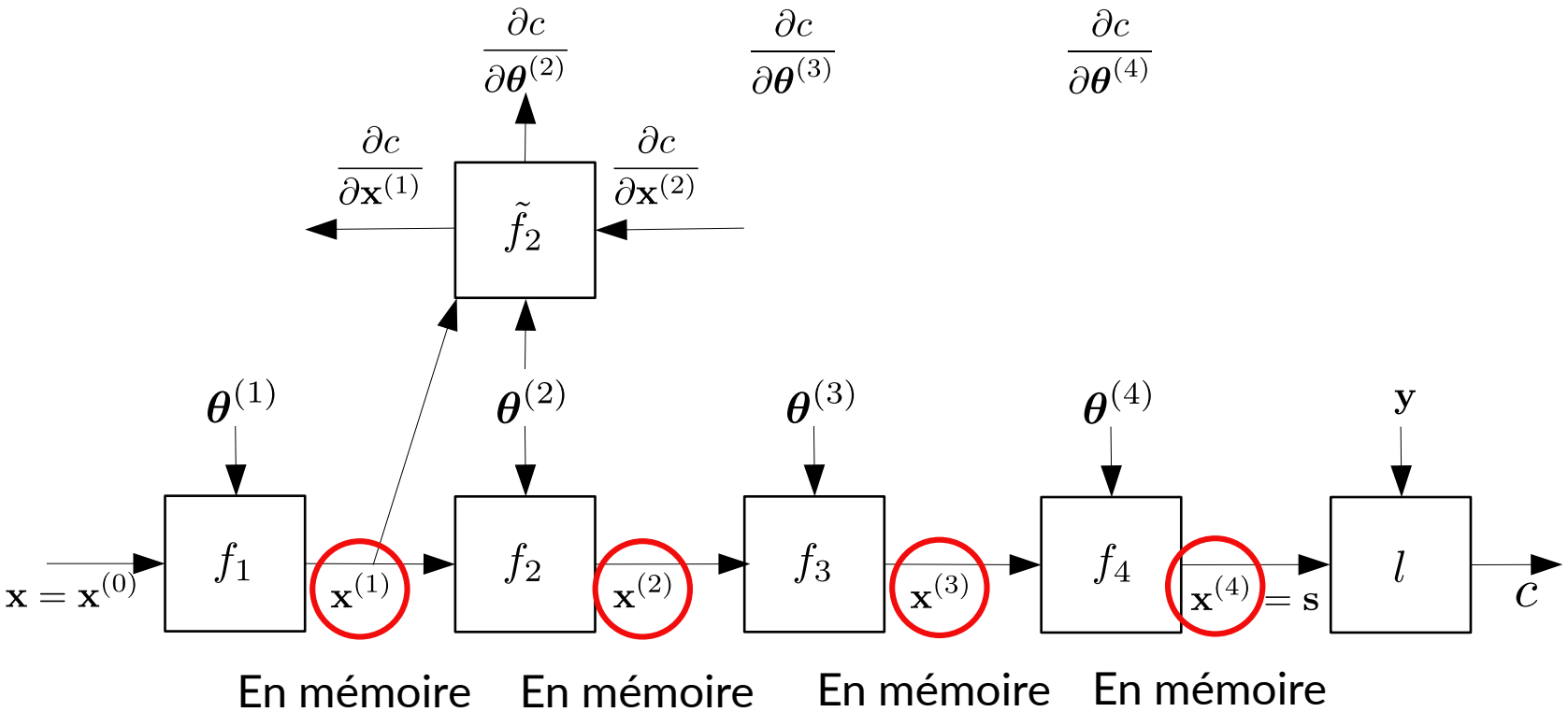
III)

Rétropropagation « classique »



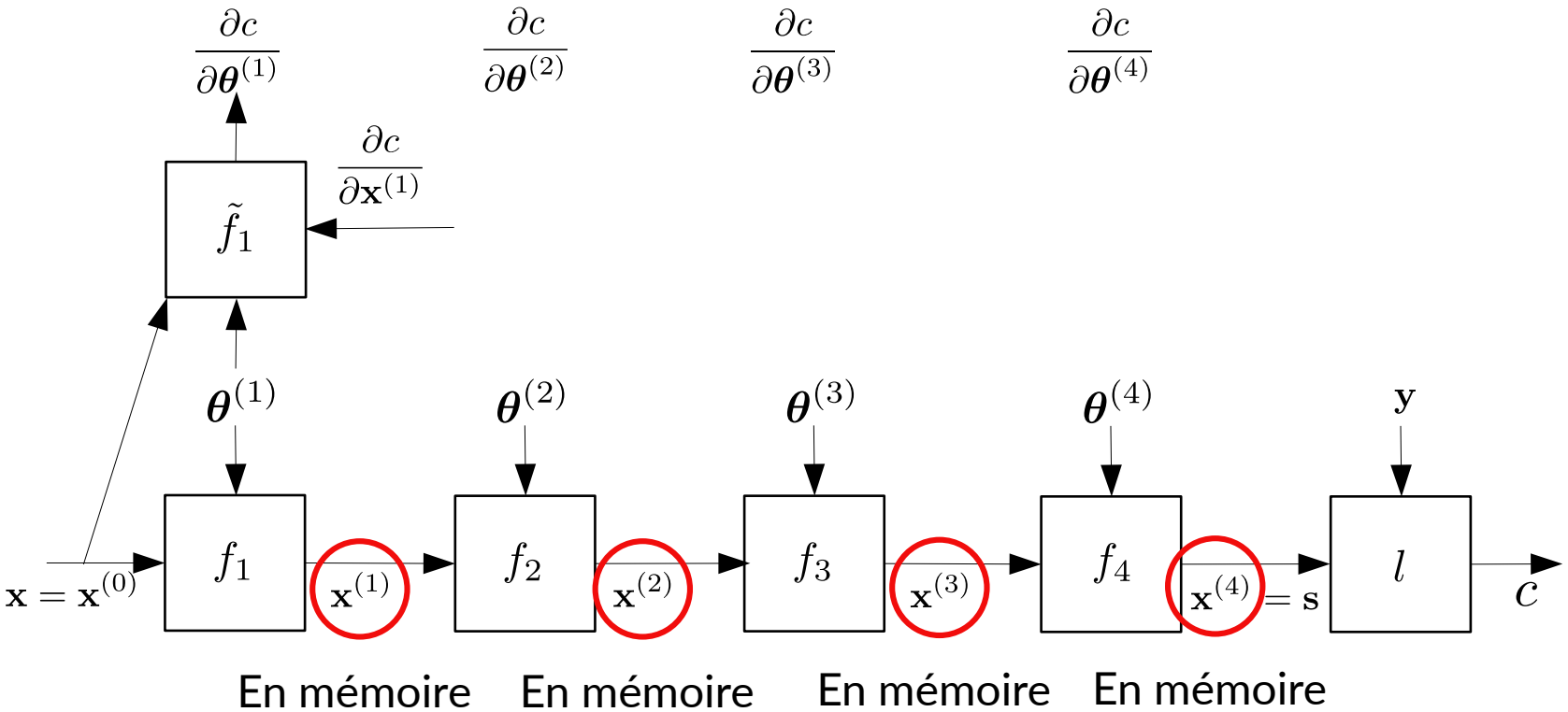
III)

Rétropropagation « classique »



III)

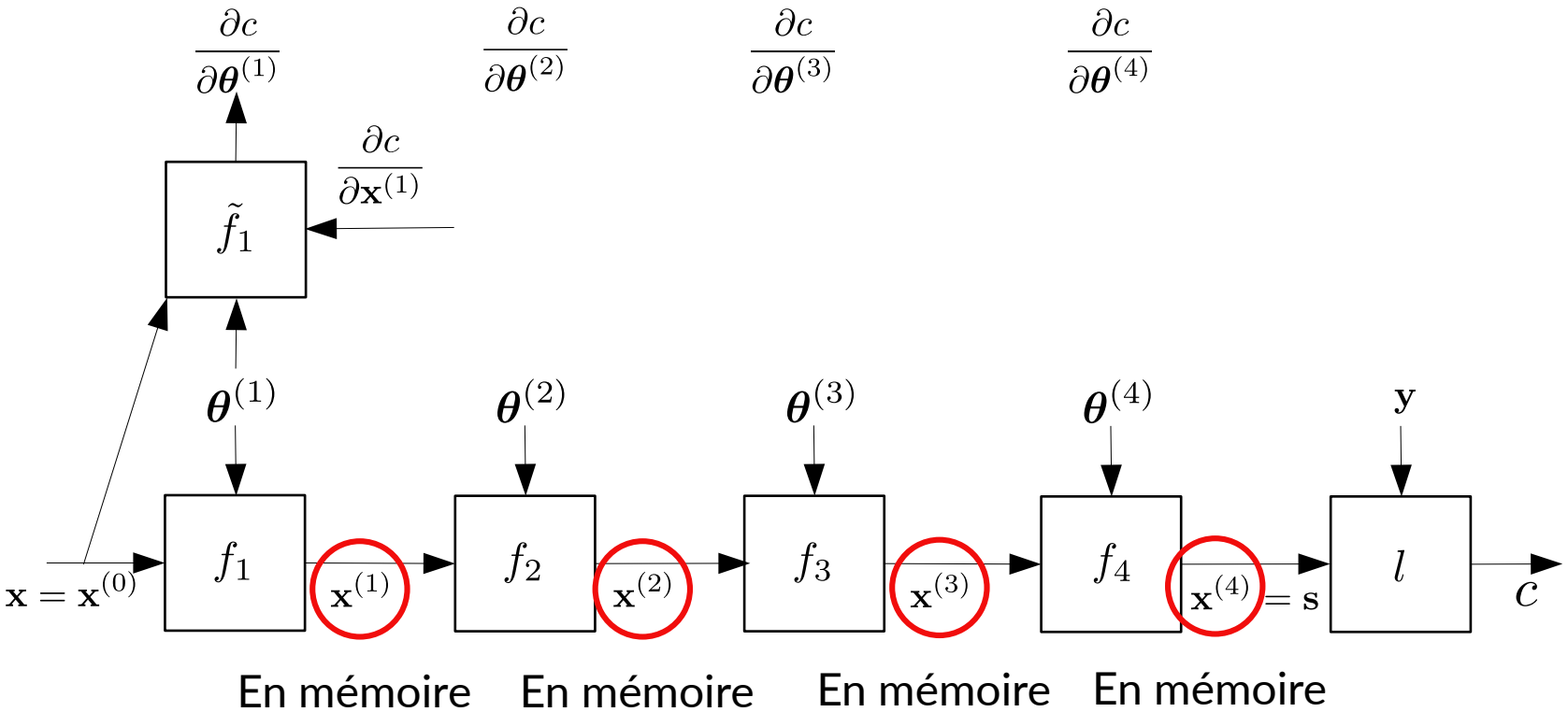
Rétropropagation « classique »



Toutes les activations sont conservées en mémoire pour effectuer la rétropropagation.

III)

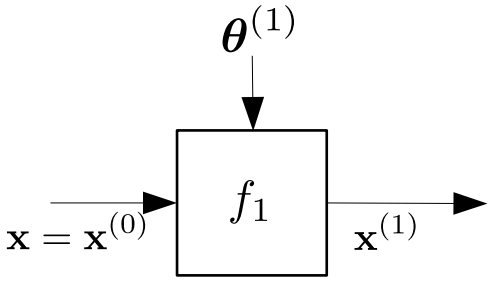
Rétropropagation « classique »



Toutes les activations sont conservées en mémoire pour effectuer la rétropropagation.
« Gradient checkpointing » = conserver uniquement quelques activations en mémoire

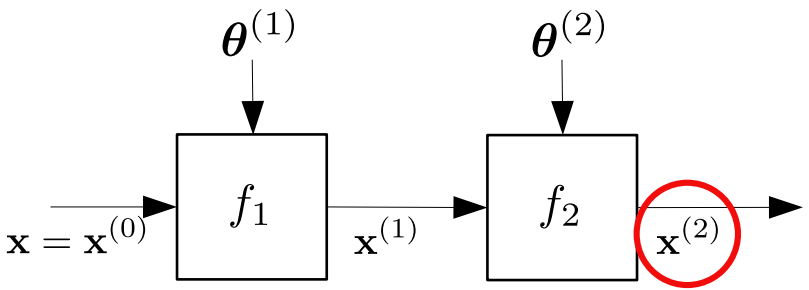
III)

Exemple de « checkpointing » des couches 2 et 4



III)

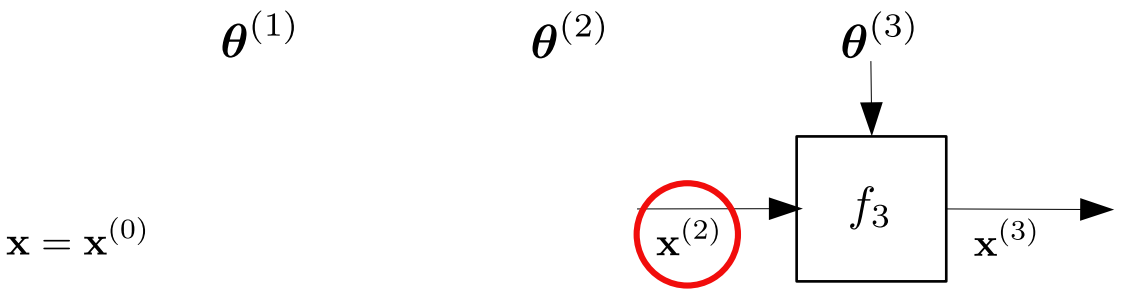
Exemple de « checkpointing » des couches 2 et 4



En mémoire

III)

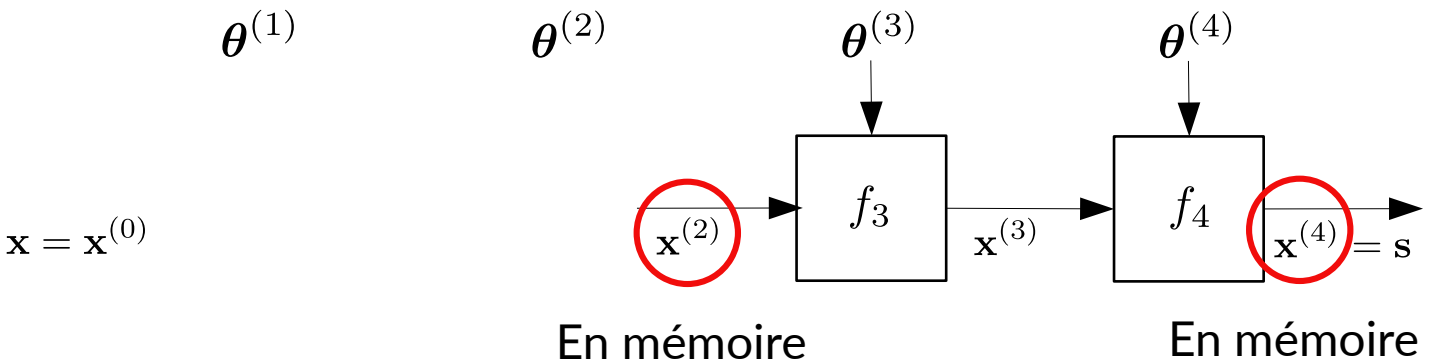
Exemple de « checkpointing » des couches 2 et 4



En mémoire

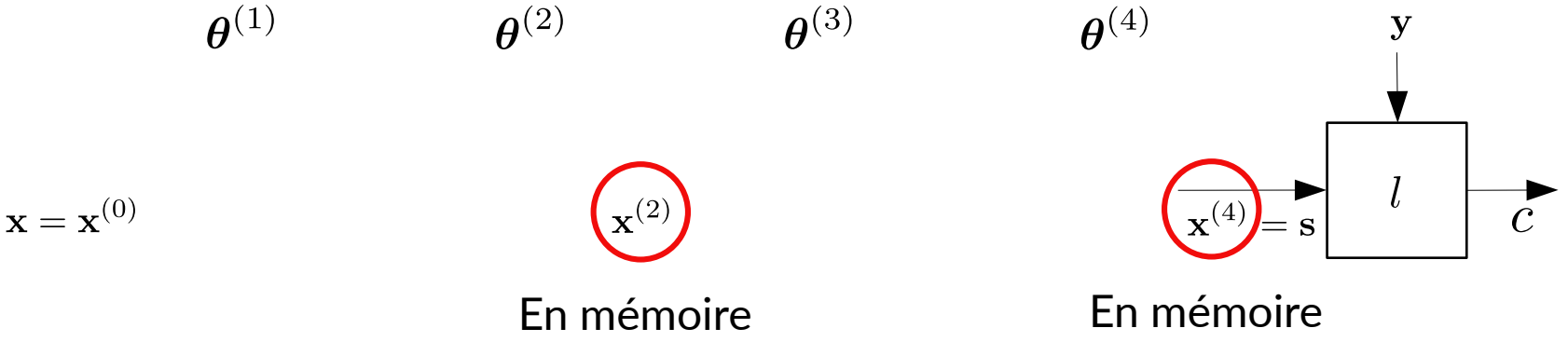
III)

Exemple de « checkpointing » des couches 2 et 4



III)

Exemple de « checkpointing » des couches 2 et 4



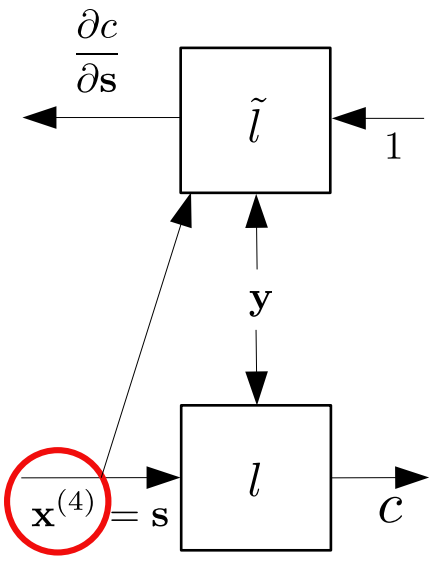
III)

Exemple de « checkpointing » des couches 2 et 4

$\boldsymbol{\theta}^{(1)}$ $\boldsymbol{\theta}^{(2)}$ $\boldsymbol{\theta}^{(3)}$ $\boldsymbol{\theta}^{(4)}$

$\mathbf{x} = \mathbf{x}^{(0)}$ $\mathbf{x}^{(2)}$

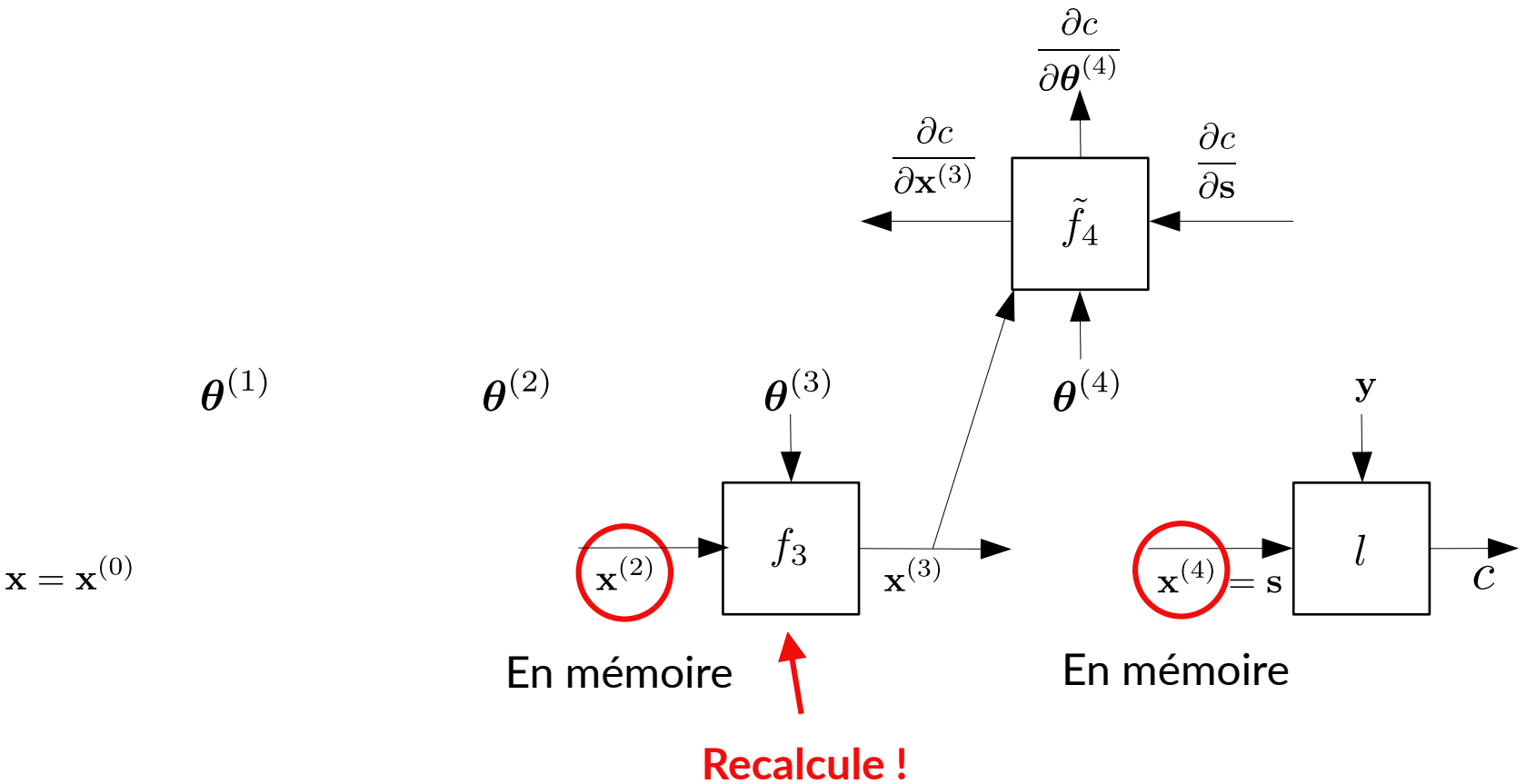
En mémoire



En mémoire

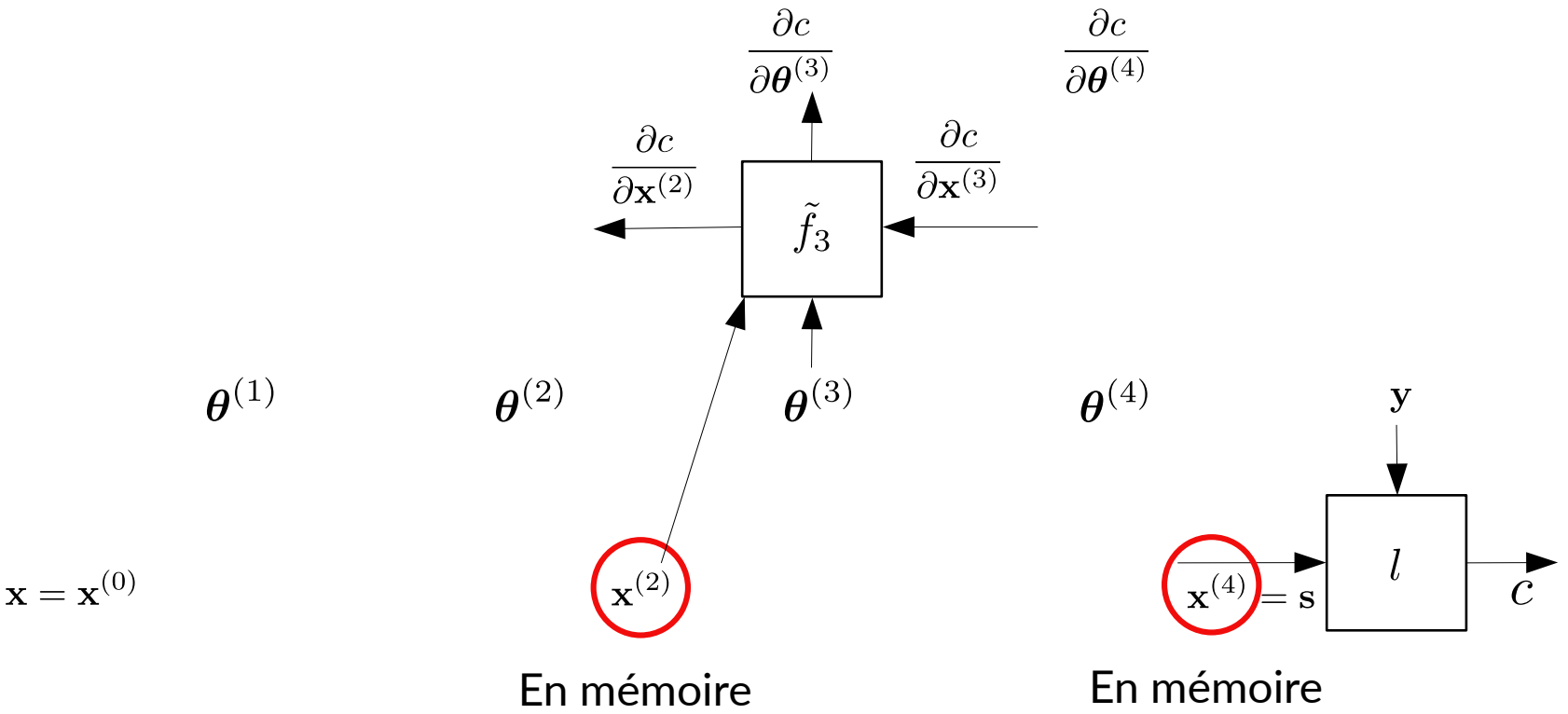
III)

Exemple de « checkpointing » des couches 2 et 4



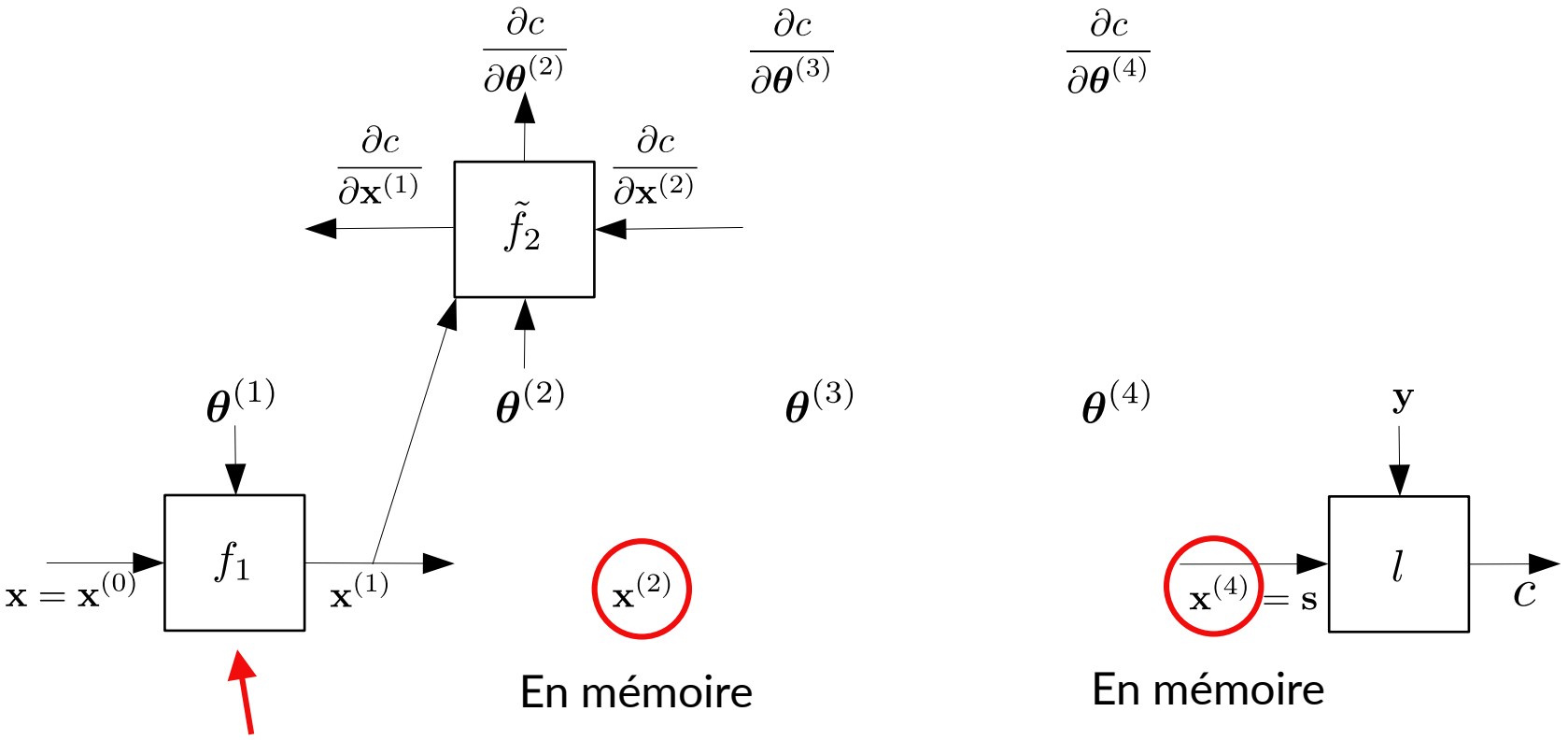
III)

Exemple de « checkpointing » des couches 2 et 4



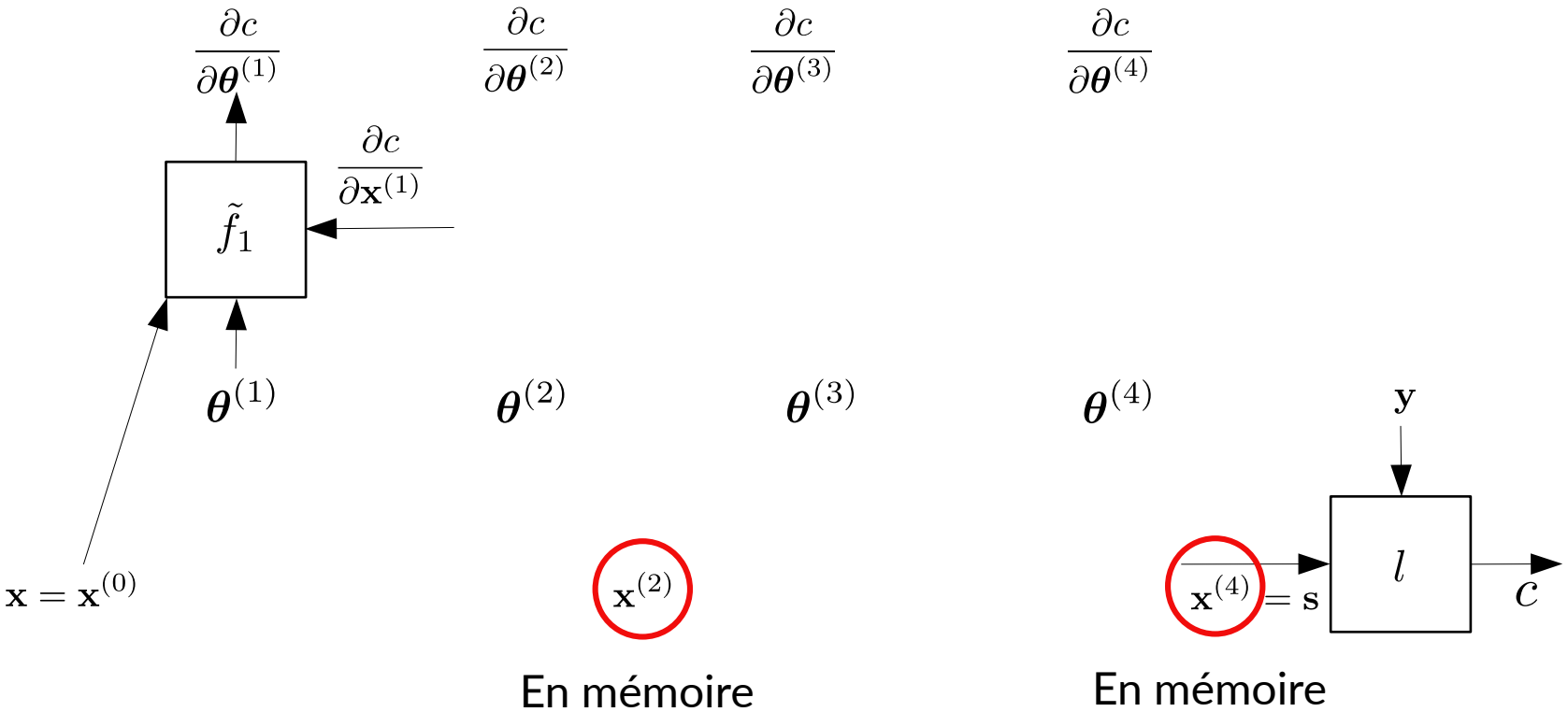
III)

Exemple de « checkpointing » des couches 2 et 4



III)

Exemple de « checkpointing » des couches 2 et 4



En mémoire

En mémoire

Résumé « checkpointing »

Réduit l'empreinte mémoire de la rétroprogation
→ permet d'entraîner de plus gros réseau ou d'augmenter la taille du minibatch

Nécessite de recalculer des activations
→ Accroît le temps de rétropropagation et donc le temps d'entraînement

IV) Apprentissage multi-GPU

Résumé des ingrédients du « Deep Learning »

1) Grande base de données étiquetées

2) « Bonne » architecture de réseau de neurones profond

- « Perceptron » multicouche, Réseau de neurones à convolution, Transformer
- Optimisation par descente de gradient stochastisque (AdamW, etc.)

3) Grande capacité de calculs en parallèle (GPUs)

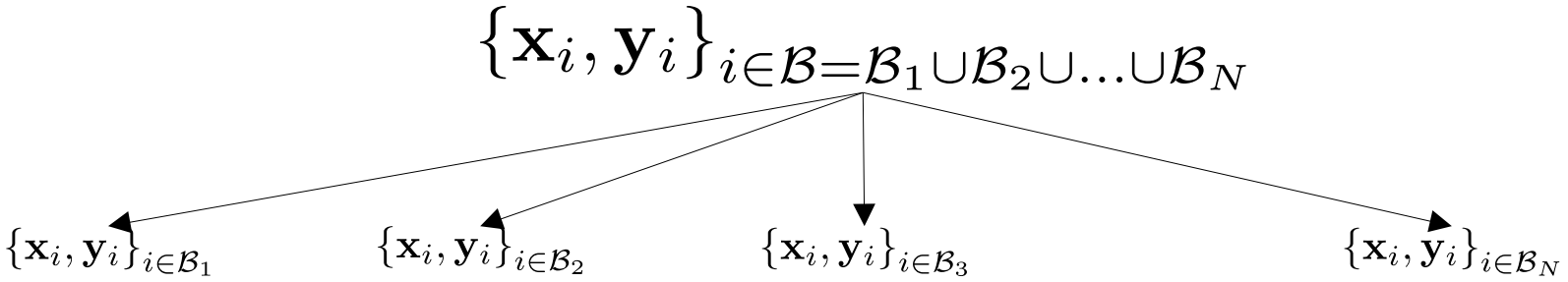
Grande capacité de calculs en parallèle

Hypothèse : On dispose de plusieurs GPUs sur lesquels rentrent de « petits » minibatchs

$$\{\mathbf{x}_i, \mathbf{y}_i\}_{i \in \mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2 \cup \ldots \cup \mathcal{B}_N}$$

Grande capacité de calculs en parallèle

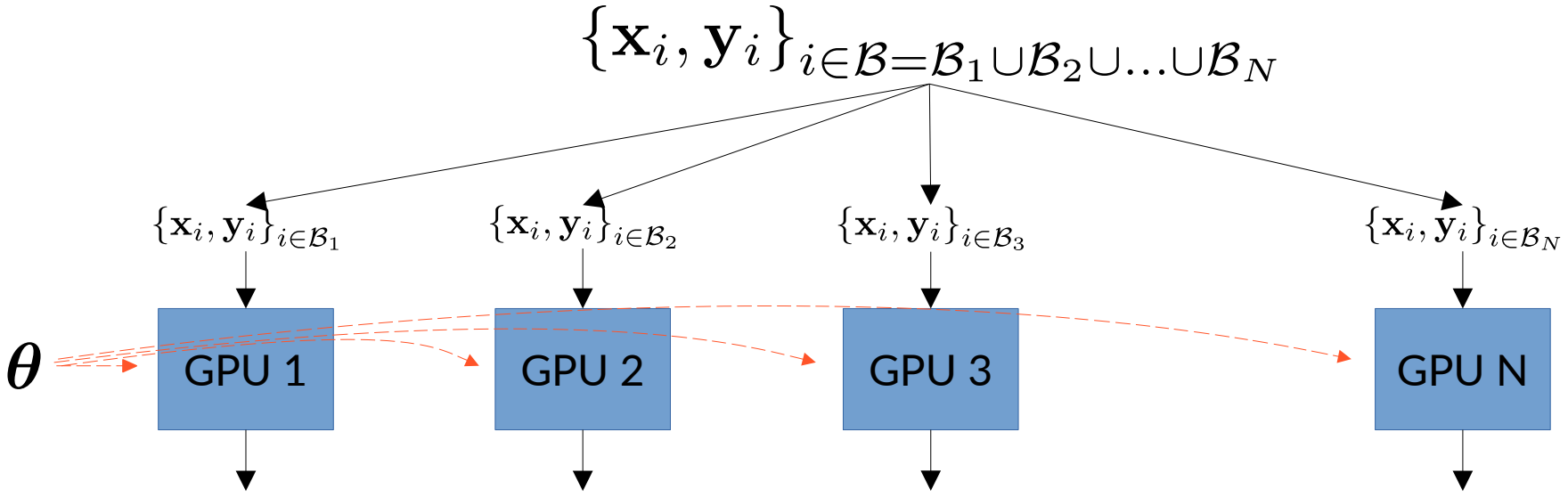
Hypothèse : On dispose de plusieurs GPUs sur lesquels rentrent de « petits » minibatchs



IV)

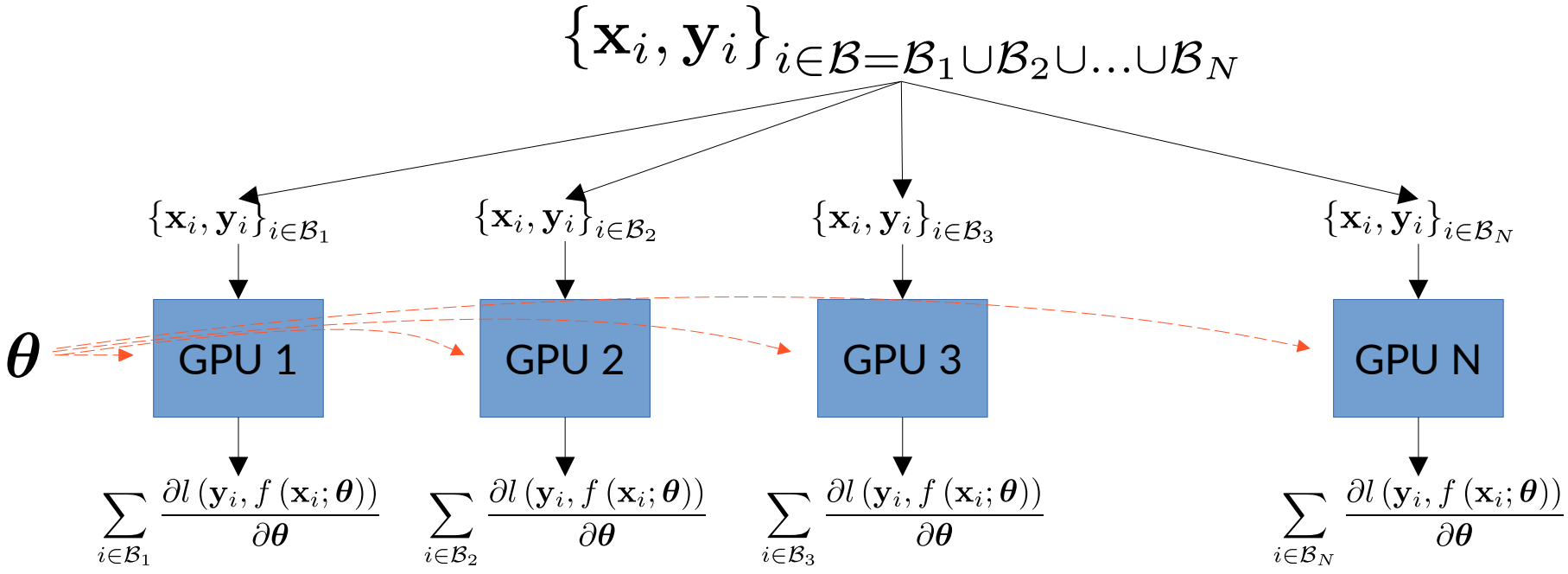
Grande capacité de calculs en parallèle

Hypothèse : On dispose de plusieurs GPUs sur lesquels rentrent de « petits » minibatchs



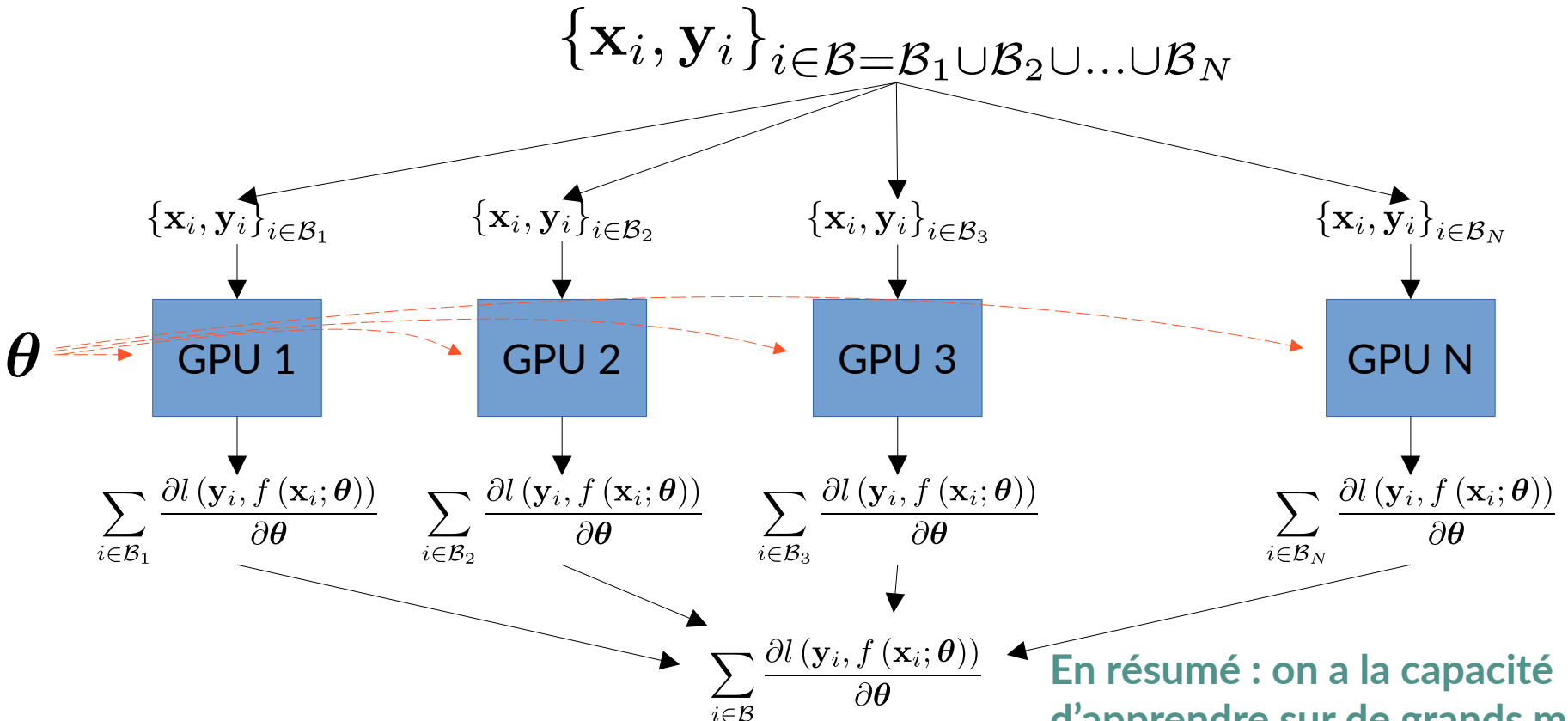
Grande capacité de calculs en parallèle

Hypothèse : On dispose de plusieurs GPUs sur lesquels rentrent de « petits » minibatchs



Grande capacité de calculs en parallèle

Hypothèse : On dispose de plusieurs GPUs sur lesquels rentrent de « petits » minibatchs



En résumé : on a la capacité d'apprendre sur de grands minibatchs

Grande capacité de calculs en parallèle

Hypothèse : On dispose de plusieurs GPUs sur lesquels rentrent de « petits » minibatchs

On a la capacité d'apprendre sur de grands minibatchs

Grande capacité de calculs en parallèle

Hypothèse : On dispose de plusieurs GPUs sur lesquels rentrent de « petits » minibatchs

↓

On a la capacité d'apprendre sur de grands minibatchs

↓

Peut-on exploiter cette capacité pour réduire le temps d'entraînement ?

IV)

Grande capacité de calculs en parallèle

Hypothèse : On dispose de plusieurs GPUs sur lesquels rentrent de « petits » minibatchs

↓

On a la capacité d'apprendre sur de grands minibatchs

↓

Peut-on exploiter cette capacité pour réduire le temps d'entraînement ?

↓

Oui ! Il suffit d'augmenter le pas d'apprentissage (« learning rate ») et de modifier l'évolution du pas d'apprentissage (« scheduling ») !

Goyal et al. (2018). Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour arXiv preprint arXiv.1706.02677

Augmentation du pas d'apprentissage

« Linear Scaling Rule » : Quand la taille du minibatch est multipliée par s, multiplier le pas d'apprentissage par s.

Goyal et al. (2018). Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour arXiv preprint arXiv.1706.02677

IV)

Augmentation du pas d'apprentissage

« Linear Scaling Rule » : Quand la taille du minibatch est multipliée par s, multiplier le pas d'apprentissage par s.

s itérations avec un minibatch de taille n

$$\boldsymbol{\theta}_{k+s} = \boldsymbol{\theta}_k - \alpha \frac{1}{n} \sum_{j=0}^{s-1} \sum_{i \in \mathcal{B}_j} \frac{\partial l\left(\mathtt{Y}_{\text{train},i}, f\left(\mathtt{X}_{\text{train},i}; \boldsymbol{\theta}\right)\right)}{\partial \boldsymbol{\theta}} \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{k+j}}$$

1 itération avec un minibatch de taille sn

$$\hat{\boldsymbol{\theta}}_{k+1} = \boldsymbol{\theta}_k - \hat{\alpha} \frac{1}{sn} \sum_{j=0}^{s-1} \sum_{i \in \mathcal{B}_j} \frac{\partial l\left(\mathtt{Y}_{\text{train},i}, f\left(\mathtt{X}_{\text{train},i}; \boldsymbol{\theta}\right)\right)}{\partial \boldsymbol{\theta}} \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}$$

Goyal et al. (2018). Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour arXiv preprint arXiv.1706.02677

IV)

Augmentation du pas d'apprentissage

« Linear Scaling Rule » : Quand la taille du minibatch est multipliée par s, multiplier le pas d'apprentissage par s.

s itérations avec un minibatch de taille n

$$\boldsymbol{\theta}_{k+s} = \boldsymbol{\theta}_k - \alpha \frac{1}{n} \sum_{j=0}^{s-1} \sum_{i \in \mathcal{B}_j} \frac{\partial l\left(\mathtt{Y}_{\text{train},i}, f\left(\mathtt{X}_{\text{train},i}; \boldsymbol{\theta}\right)\right)}{\partial \boldsymbol{\theta}} \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{k+j}}$$

1 itération avec un minibatch de taille sn

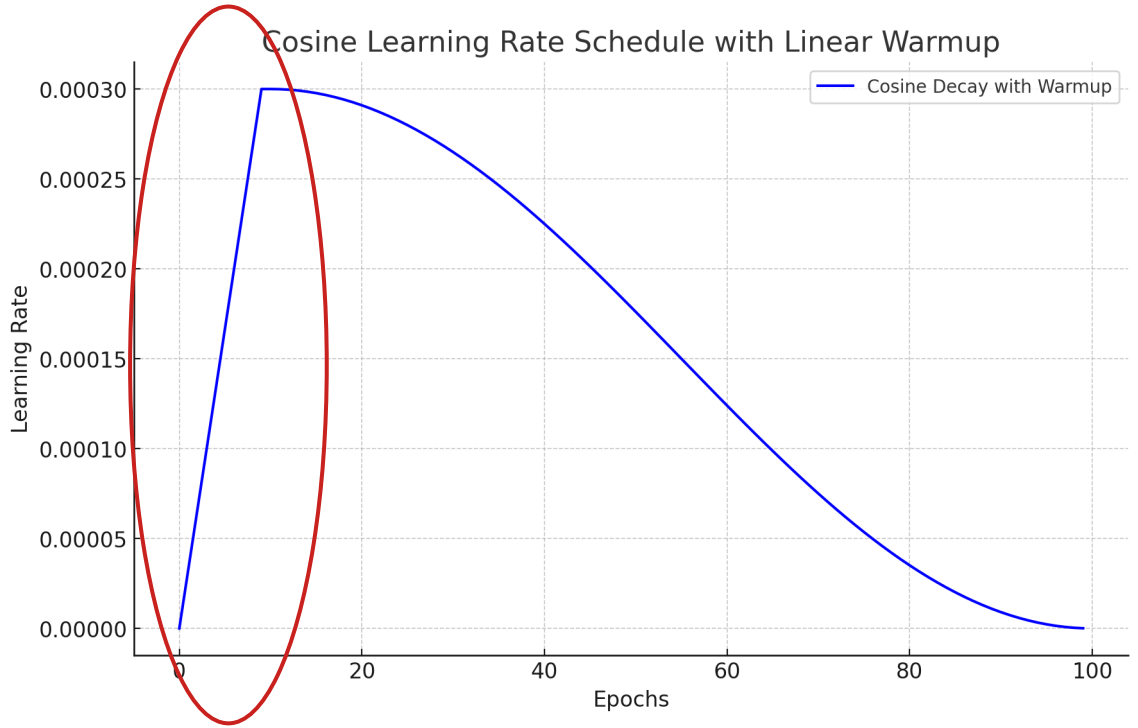
$$\hat{\boldsymbol{\theta}}_{k+1} = \boldsymbol{\theta}_k - \hat{\alpha} \frac{1}{sn} \sum_{j=0}^{s-1} \sum_{i \in \mathcal{B}_j} \frac{\partial l\left(\mathtt{Y}_{\text{train},i}, f\left(\mathtt{X}_{\text{train},i}; \boldsymbol{\theta}\right)\right)}{\partial \boldsymbol{\theta}} \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{k}}$$

Si $\frac{\partial l\left(\mathtt{Y}_{\text{train},i}, f\left(\mathtt{X}_{\text{train},i}; \boldsymbol{\theta}\right)\right)}{\partial \boldsymbol{\theta}} \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{k+j}} \approx \frac{\partial l\left(\mathtt{Y}_{\text{train},i}, f\left(\mathtt{X}_{\text{train},i}; \boldsymbol{\theta}\right)\right)}{\partial \boldsymbol{\theta}} \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{k}}$ et $\hat{\alpha} = s\alpha$ alors $\boldsymbol{\theta}_{k+s} \approx \hat{\boldsymbol{\theta}}_{k+1}$

Goyal et al. (2018). Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour arXiv preprint arXiv.1706.02677

IV)

Évolution du pas d'apprentissage

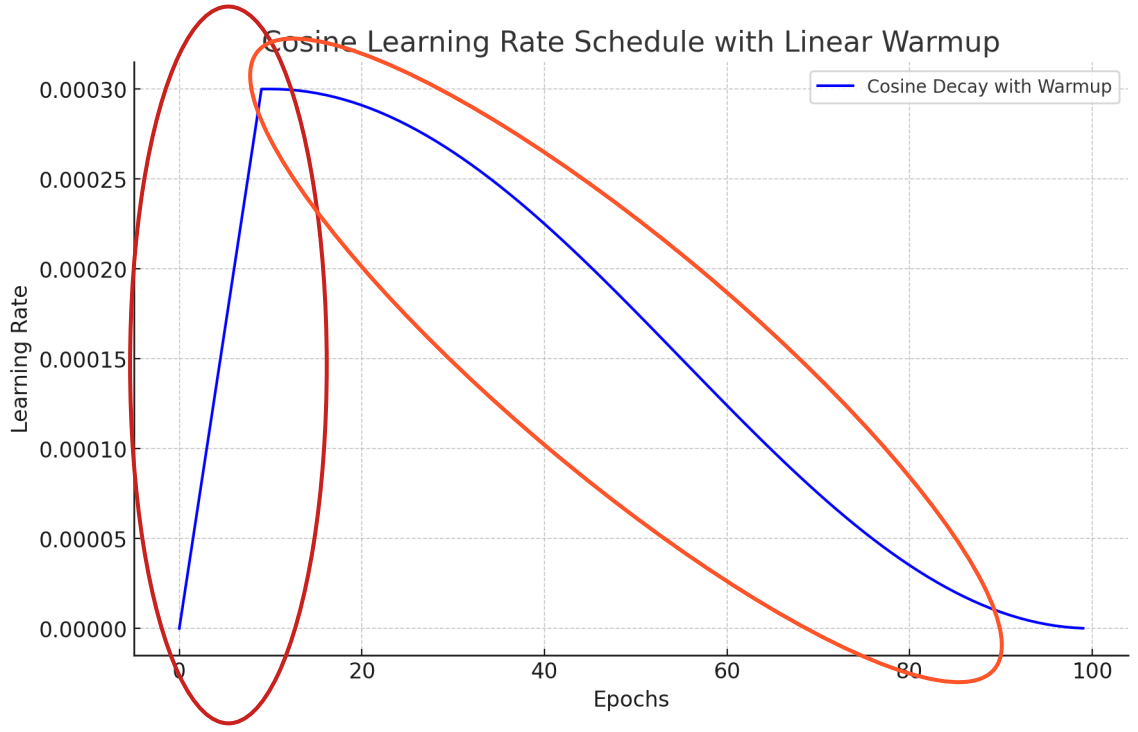


« Linear warmup » : augmentation progressive et **rapide** du pas d'apprentissage

Permet de se placer dans une « bonne » région de l'espace des paramètres

IV)

Évolution du pas d'apprentissage

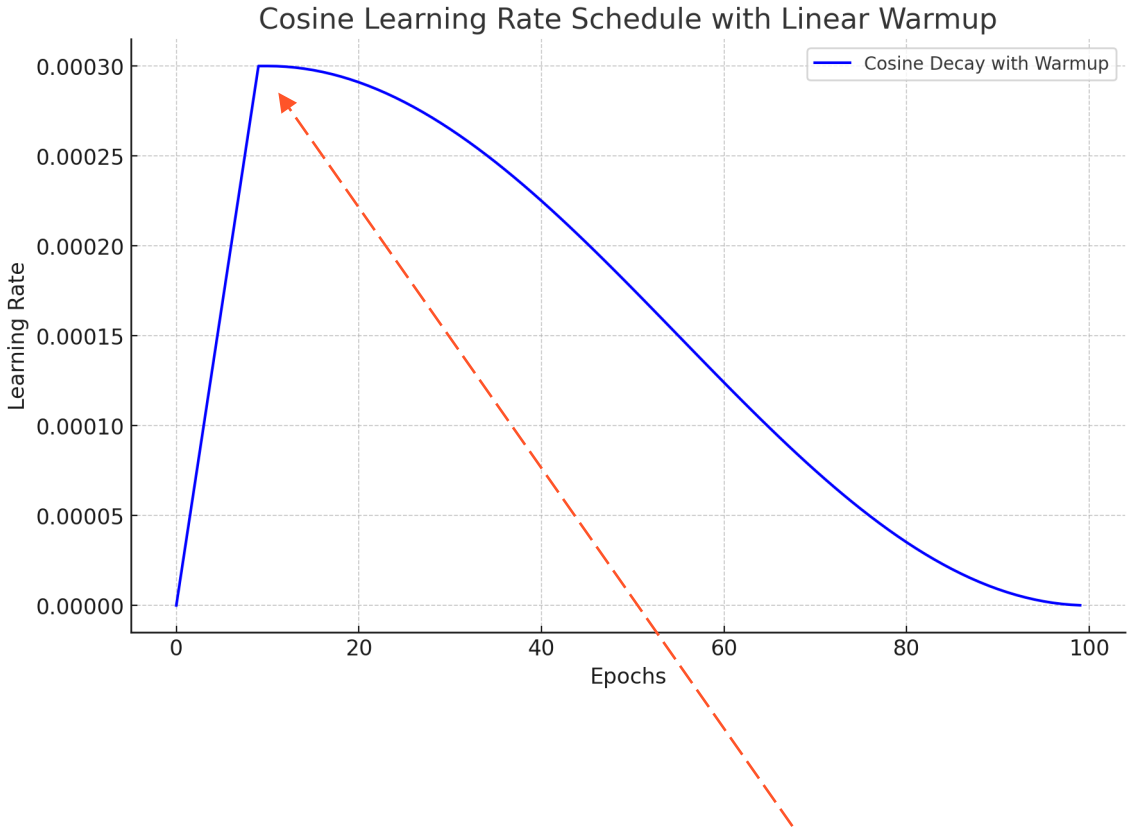


« Linear warmup » : augmentation progressive et **rapide** du pas d'apprentissage
« Decay » : décroissance progressive et **lente** du pas d'apprentissage

Décroissance « classique » du pas d'apprentissage

IV)

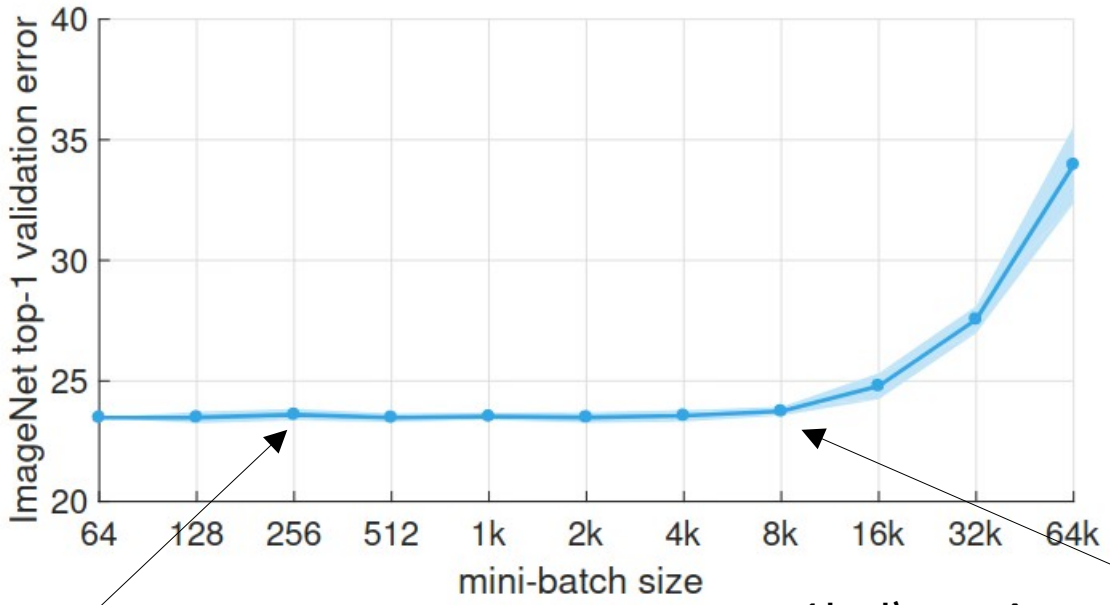
Évolution du pas d'apprentissage



Valeur du pic définie par la « linear scaling rule ».

Grande capacité de calculs en parallèle

Il suffit d'augmenter le pas d'apprentissage (« learning rate ») et de modifier l'évolution du pas d'apprentissage (« scheduling ») !



29h d'entraînement

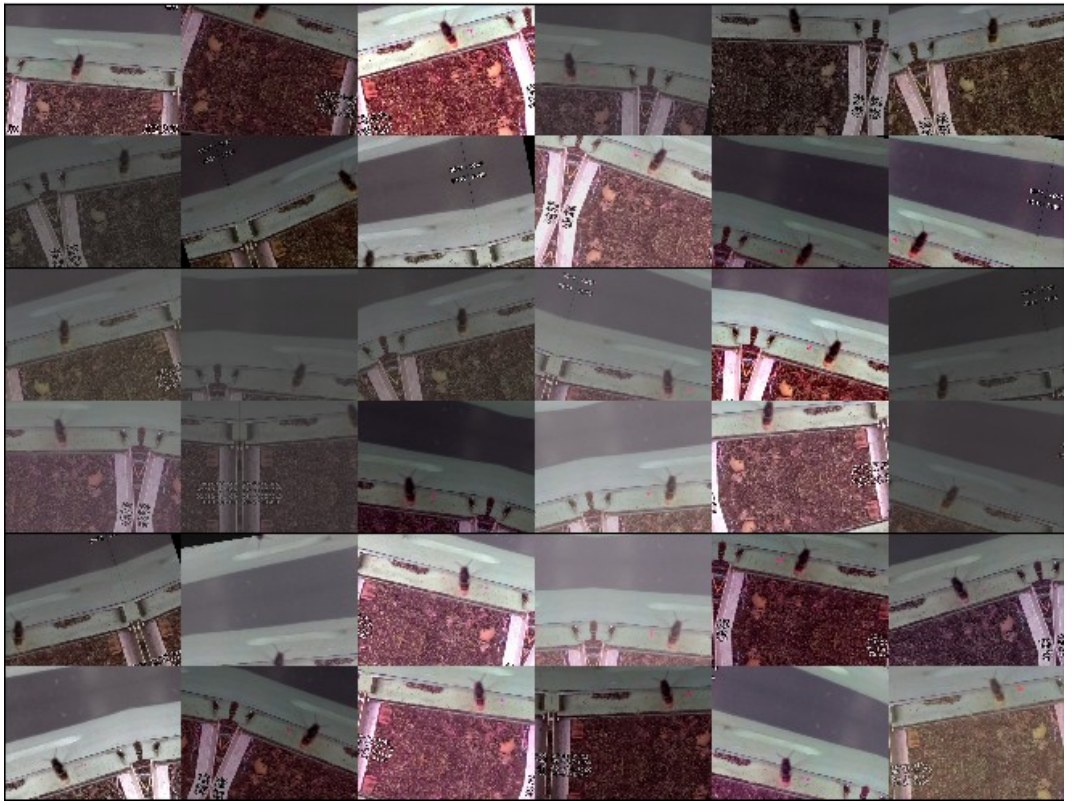
1h d'entraînement sans perte de performance

Goyal et al. (2018). Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour arXiv preprint arXiv.1706.02677

Annexes

Augmentation de données

- Mirroir
- Transformation affine
- Perturbation couleur
- Effacement
- Bruit
- …



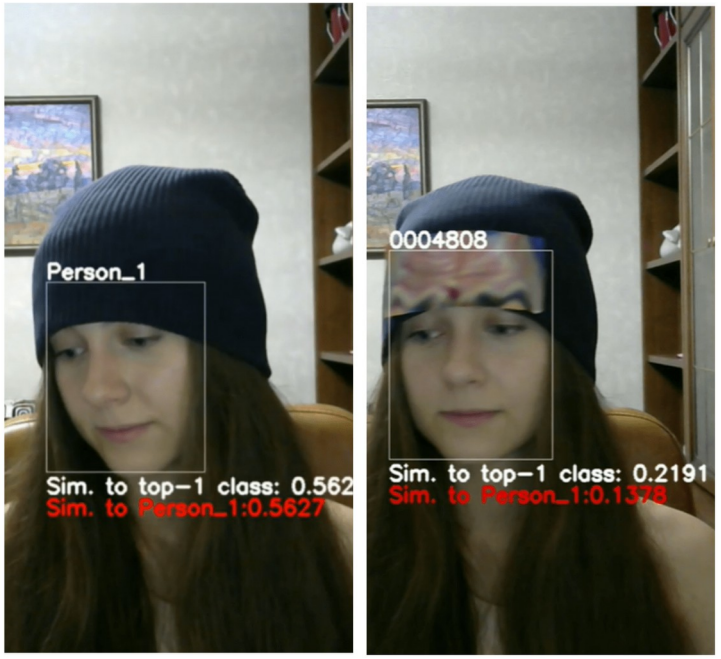
Cubuk et al. (2019). Autoaugment: Learning augmentation strategies from data. CVPR

Cubuk et al. (2020). Randaugment: Practical automated data augmentation with a reduced search space. CVPR

“Adversarial examples”

Source : https://openai.com/blog/adversarial-example-research/

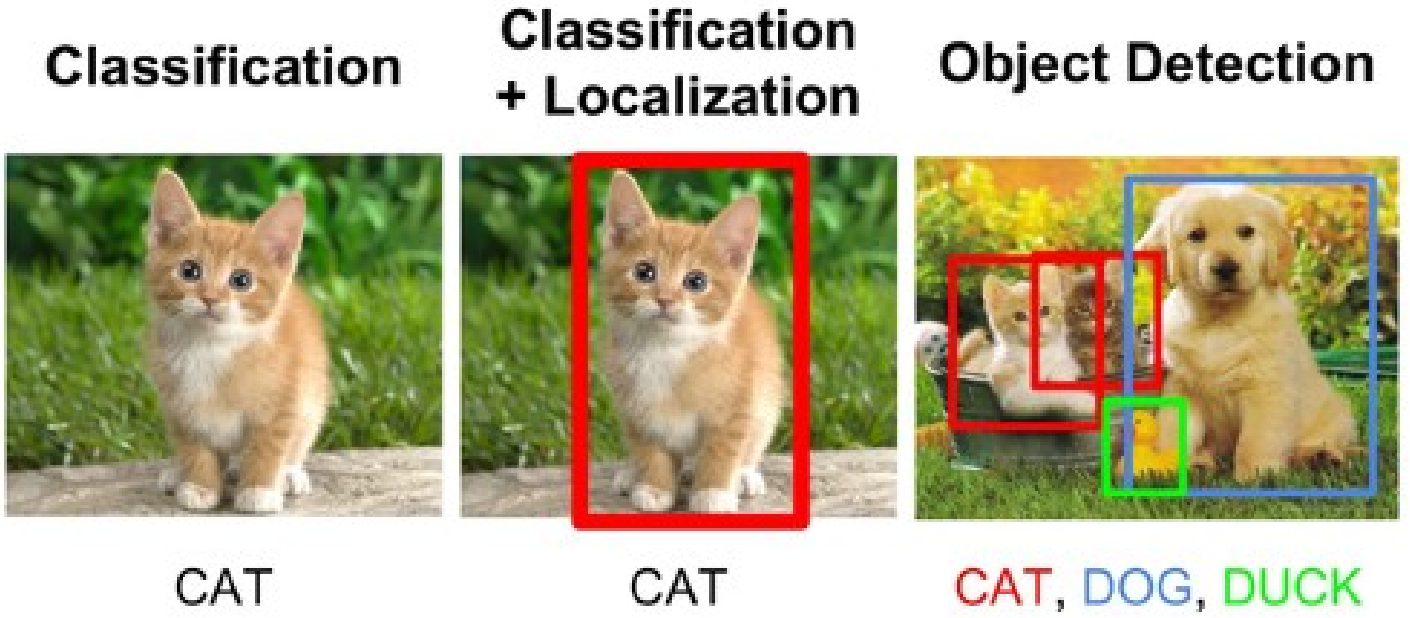
“Adversarial patches”



Sources : “ADVHAT: Real-world adversarial attack on ArcFace face ID system” (https://arxiv.org/pdf/1908.08705.pdf)
”Robust Physical-World Attacks on Deep Learning Visual Classification” (https://arxiv.org/pdf/1707.08945.pdf)

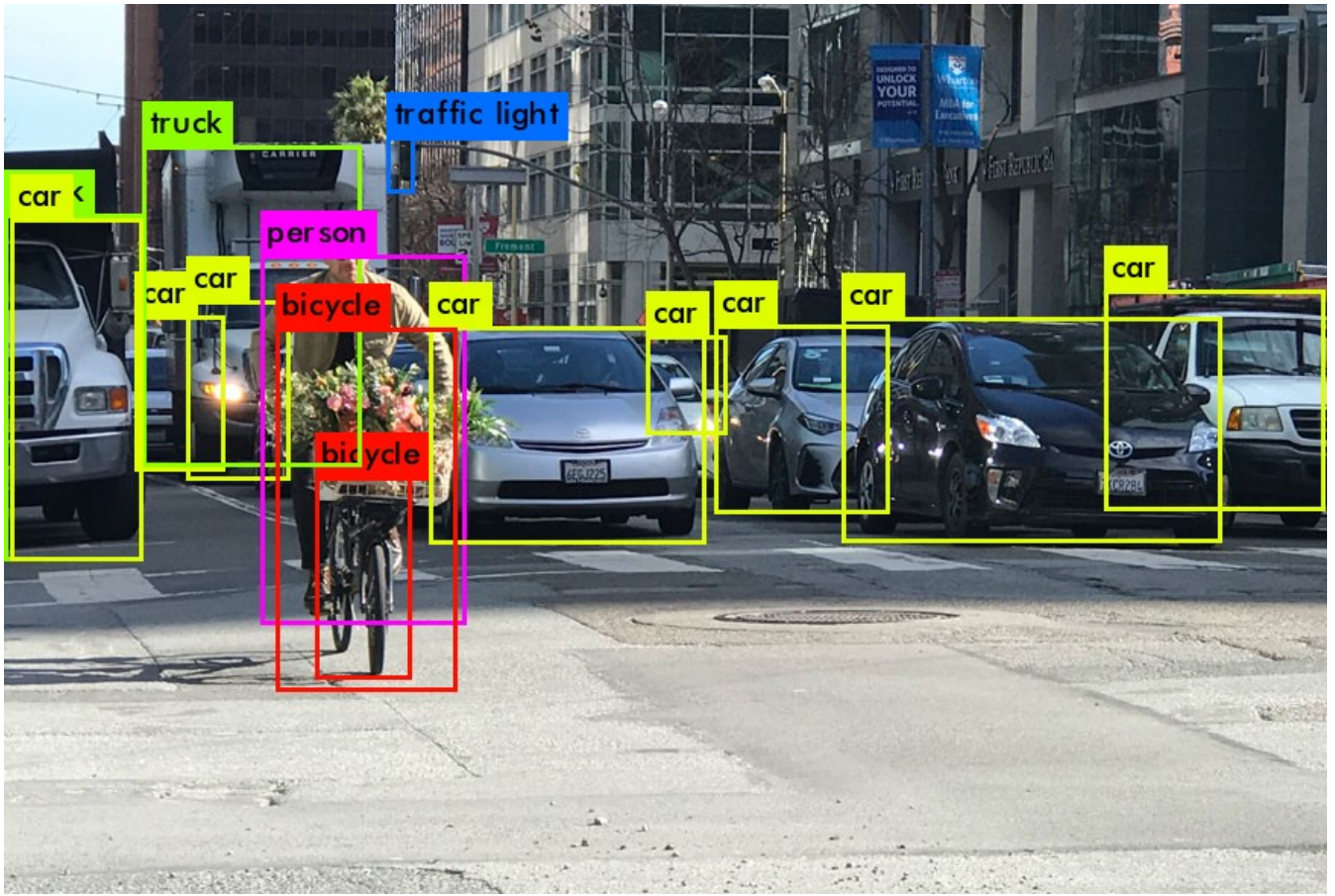
Annexe : Application à la détection d’objets dans une image

Détection d’objets



Source : https://medium.com/analytics-vidhya/yolov3-real-time-object-detection-54e69037b6d0

Exemple de détection d'objets



Source : https://towardsdatascience.com/yolo-you-only-look-once-17f9280a47b0

Formulation du problème

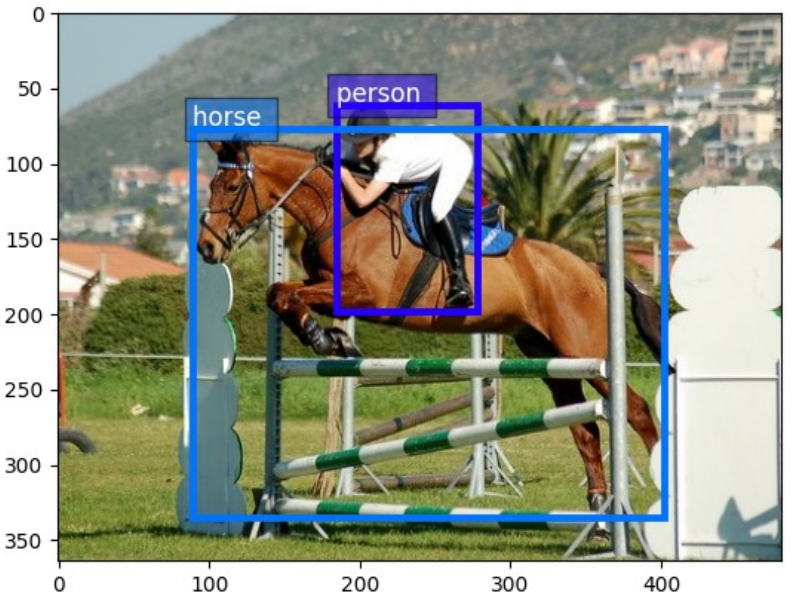
- Objectif
  - Prédire une boîte englobante autour de chaque objet,
  - Prédire la classe de l'objet,
  - En utilisant un réseau de neurones en apprentissage supervisé.

- Questions
  - Quelles données annotées disponibles ?
  - Quelle architecture ?
  - Quelle fonction de coût ?

Quelles données annotées disponibles ?

Microsoft COCO dataset (80 classes)

1: 'person',
2: 'bicycle',
3: 'car',
4: 'motorcycle',
5: 'airplane',
6: 'bs',
7: 'train',
8: 'trck',
9: 'boat',
10: 'traffic light',
11: 'fire hydrant',
12: 'stop sign',
13: 'parking meter',
14: 'bench',
15: 'bird',
16: 'cat',
17: 'dog',
18: 'horse',
19: 'sheep',
20: 'cow',
21: 'elephant',
22: 'bear',
23: 'zebra',
24: 'giraffe',
25: 'backpack',
26: 'mbrella',
27: 'handbag',
28: 'tie',
29: 'sitcase',
30: 'frisbee',
31: 'skis',
32: 'snowboard',
33: 'sports ball',
34: 'kite',
35: 'baseball bat',
36: 'baseball glove',
37: 'skateboard',
38: 'srfboard',
39: 'tennis racket',
40: 'bottle',
41: 'wine glass',
42: 'cp',
43: 'fork',
44: 'knife',
45: 'spoon',
46: 'bowl',
47: 'banana',
48: 'apple',
49: 'sandwich',
50: 'orange',
51: 'broccoli',
52: 'carrot',
53: 'hot dog',
54: 'pizza',
55: 'dont',
56: 'cake',
57: 'chair',
58: 'coch',
59: 'potted plant',
60: 'bed',
61: 'dining table',
62: 'toilet',
63: 'tv',
64: 'laptop',
65: 'mose',
66: 'remote',
67: 'keyboard',
68: 'cell phone',
69: 'microwave',
70: 'oven',
71: 'toaster',
72: 'sink',
73: 'refrigerator',
74: 'book',
75: 'clock',
76: 'vase',
77: 'scissors',
78: 'teddy bear',
79: 'hair drier',
80: 'toothbrsh'



Pascal VOC dataset (20 classes)

Person:
- 1: person

Animal:
- 2: bird
- 3: cat
- 4: cow
- 5: dog
- 6: horse
- 7: sheep

Vehicle:
- 8: aeroplane
- 9: bicycle
- 10: boat
- 11: bus
- 12: car
- 13: motorbike
- 14: train

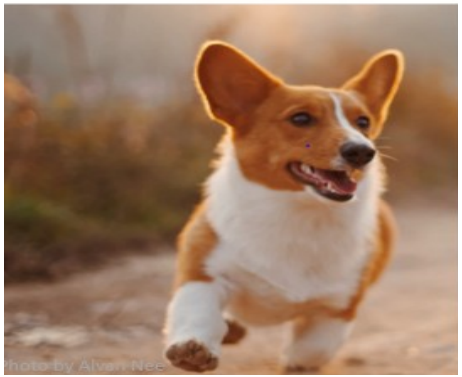
Indoor:
- 15: bottle
- 16: chair
- 17: dining table
- 18: potted plant
- 19: sofa
- 20: tv/monitor

Comment prédire le nombre d'objets et pour chaque objet sa boîte et sa classe ?

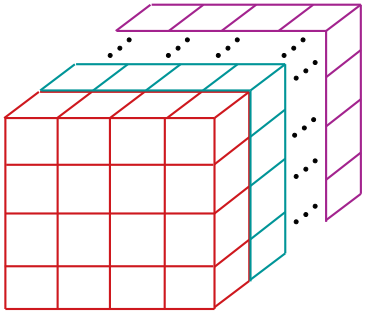
Exemple de solution : CenterNet (Zhou et. al, Objects as points, 2019)

keypoint heatmap [C] local offset [2] object size [2]

CenterNet

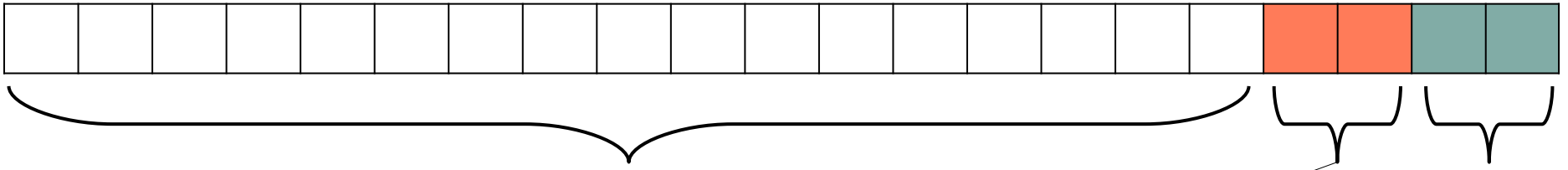


CNN



3 x H x W

(C+2+2) x H/4 x W/4

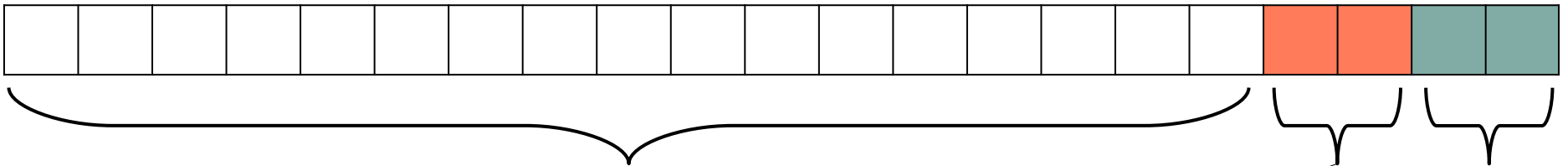


C scores (un pour chaque classe)

Décalage x et y
du centre de la boîte

Largeur et hauteur
de la boîte

CenterNet (suite)



C scores (un pour chaque classe)

Décalage x et y
du centre de la boîte

Largeur et hauteur
de la boîte

Fonction de coût = somme de trois fonctions de coût

- Pour les scores : « pixelwise logistic regression » (i.e. sur chaque case grise)

- Pour le décalage : régression L1 (sur les cases oranges s'il y a une boîte sinon rien)

- Pour la largeur et la hauteur : régression L1 (sur les cases vertes s'il y a une boîte sinon rien)

CenterNet (suite)