Journée NN2022

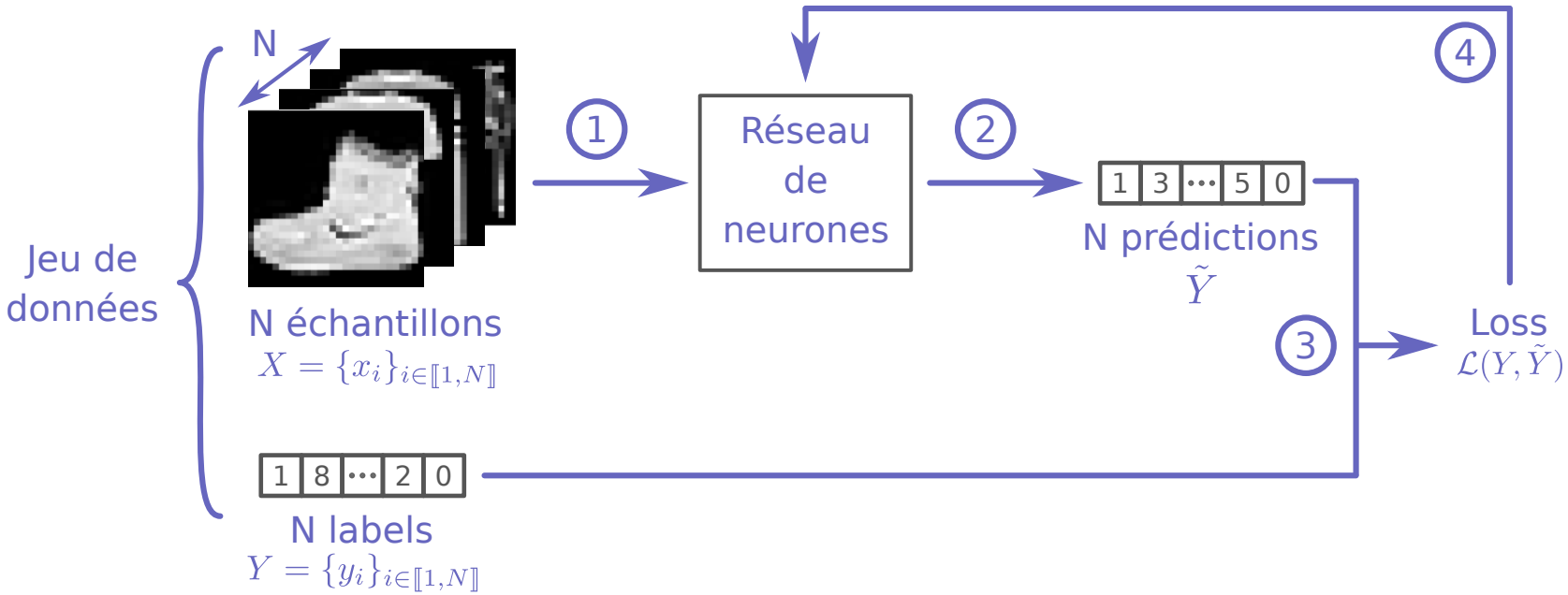
Odyssée Merveille

21/04/2022

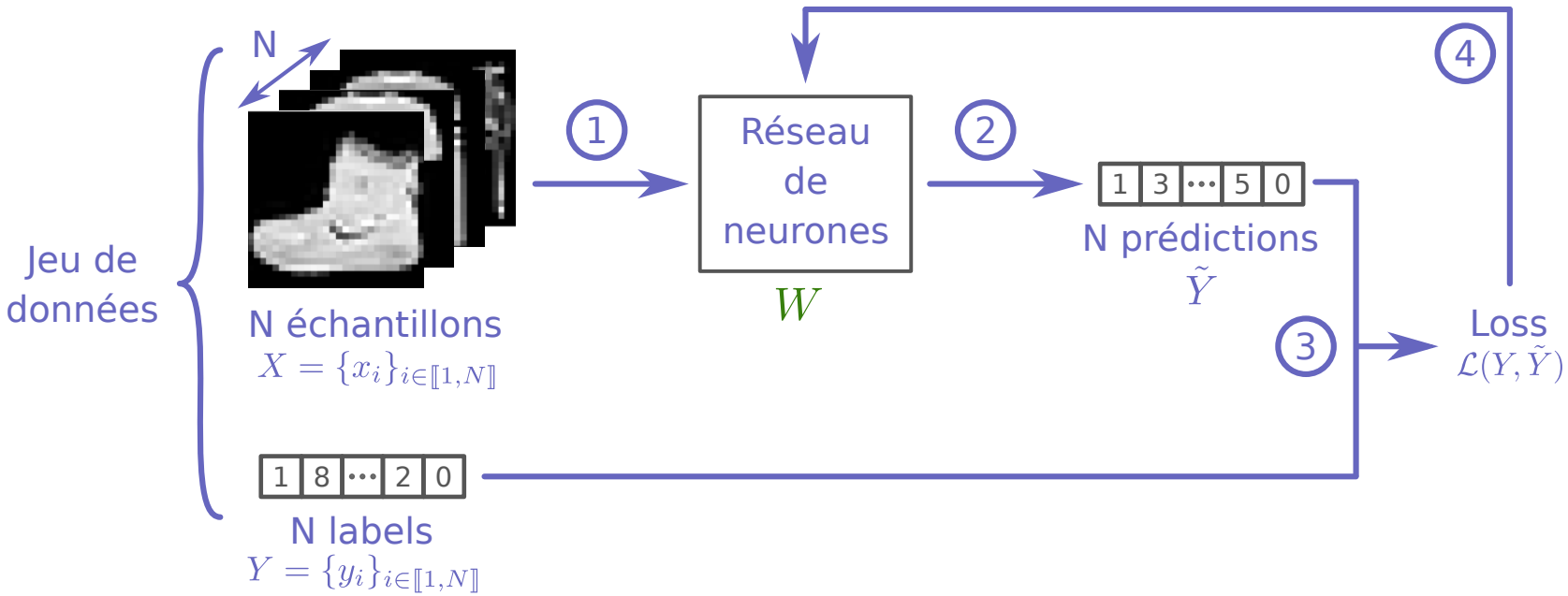
CREATIS

Plan

Retour sur l'entraînement



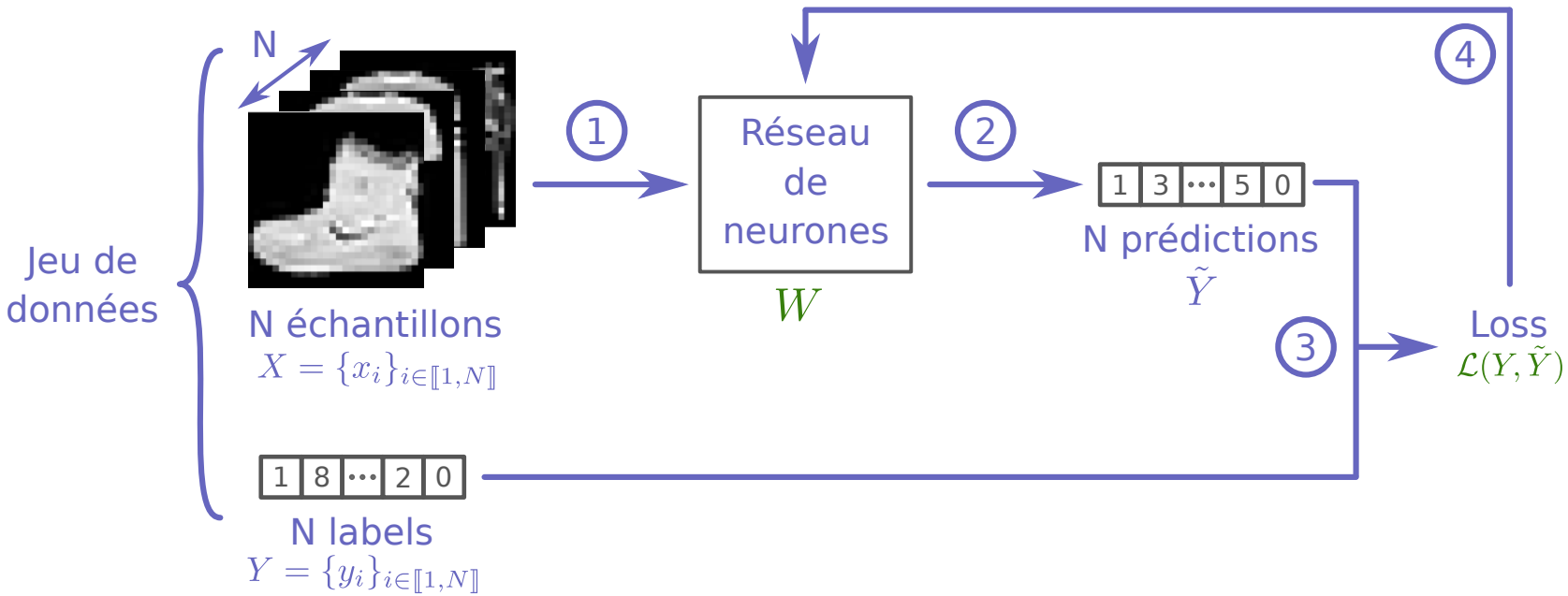
Retour sur l'entraînement



Paramètres :

- Paramètres du réseau W (appris durant l'entraînement)

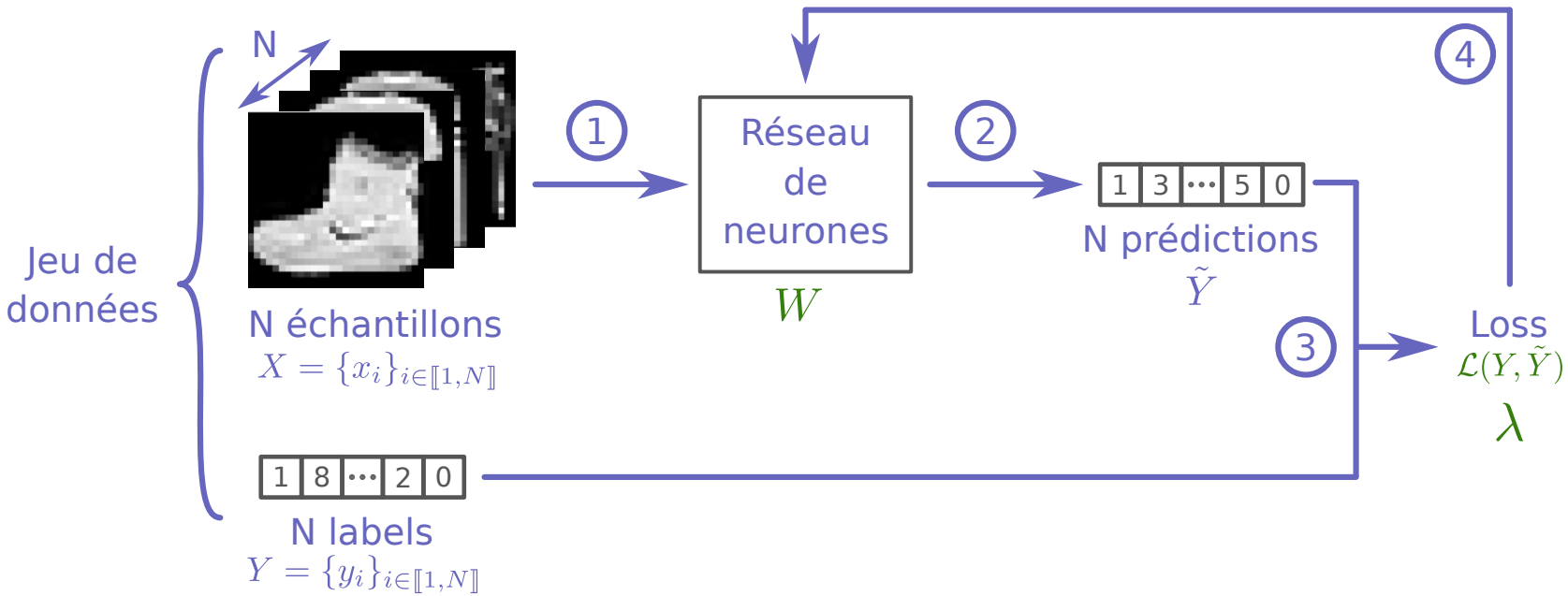
Retour sur l'entraînement



Paramètres :

- Paramètres du réseau W (appris durant l'entraînement)
- Hyperparamètres (à fixer avant l'entraînement)
  - Type de Loss $\mathcal{L}(y_i, \tilde{y}_i)$

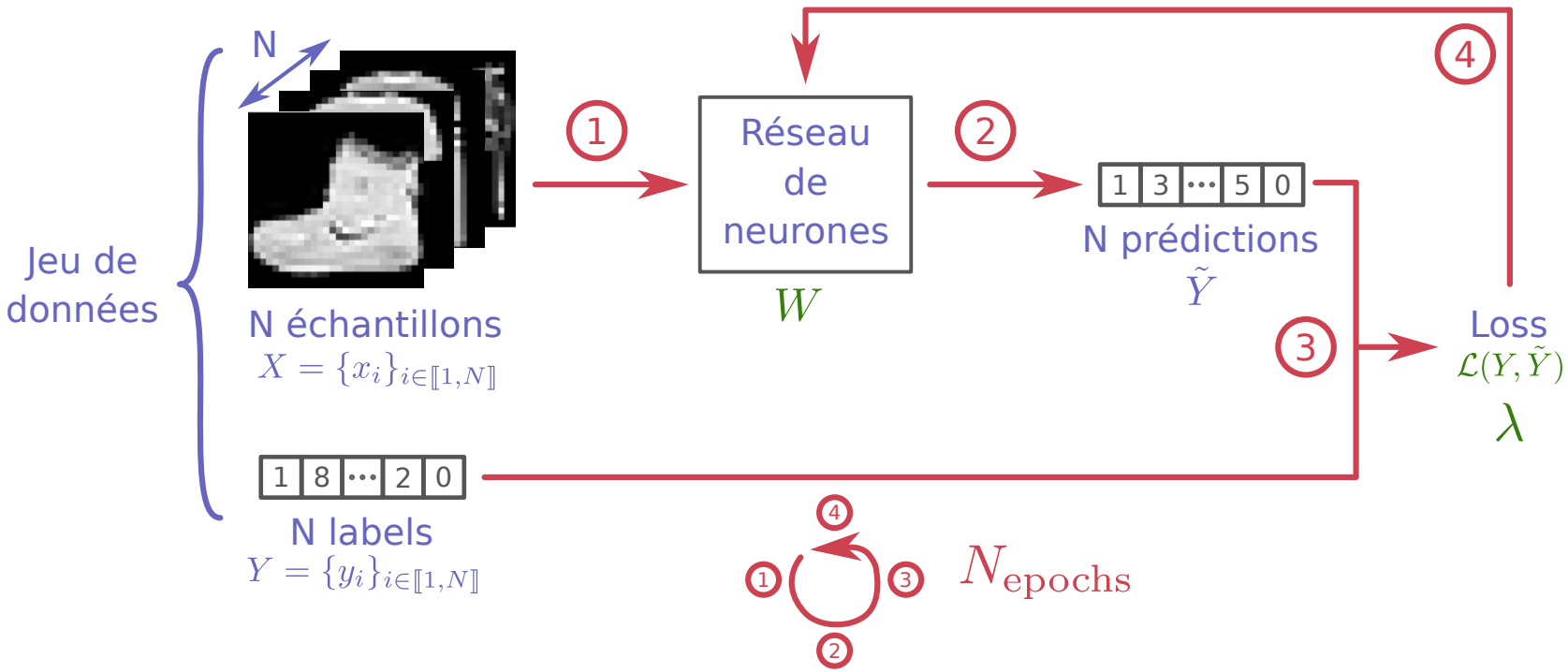
Retour sur l'entraînement



Paramètres :

- Paramètres du réseau W (appris durant l'entraînement)
- Hyperparamètres (à fixer avant l'entraînement)
  - Type de Loss $\mathcal{L}(y_i, \tilde{y}_i)$
  - Learning rate λ

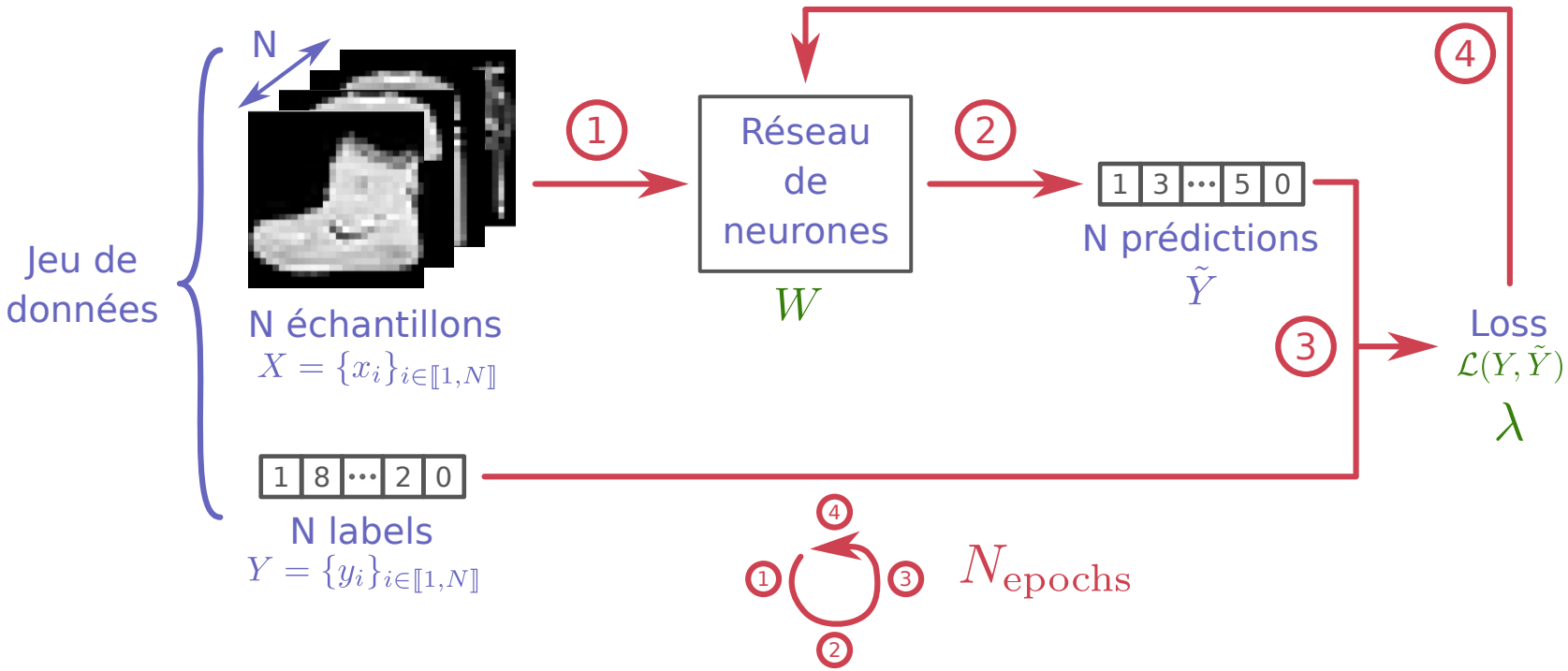
Retour sur l'entraînement



Paramètres :

- Paramètres du réseau W (appris durant l'entraînement)
- Hyperparamètres (à fixer avant l'entraînement)
  - Type de Loss $\mathcal{L}(y_i, \tilde{y}_i)$
  - Learning rate λ
  - Nombre d'epochs N_{epochs}

Retour sur l'entraînement

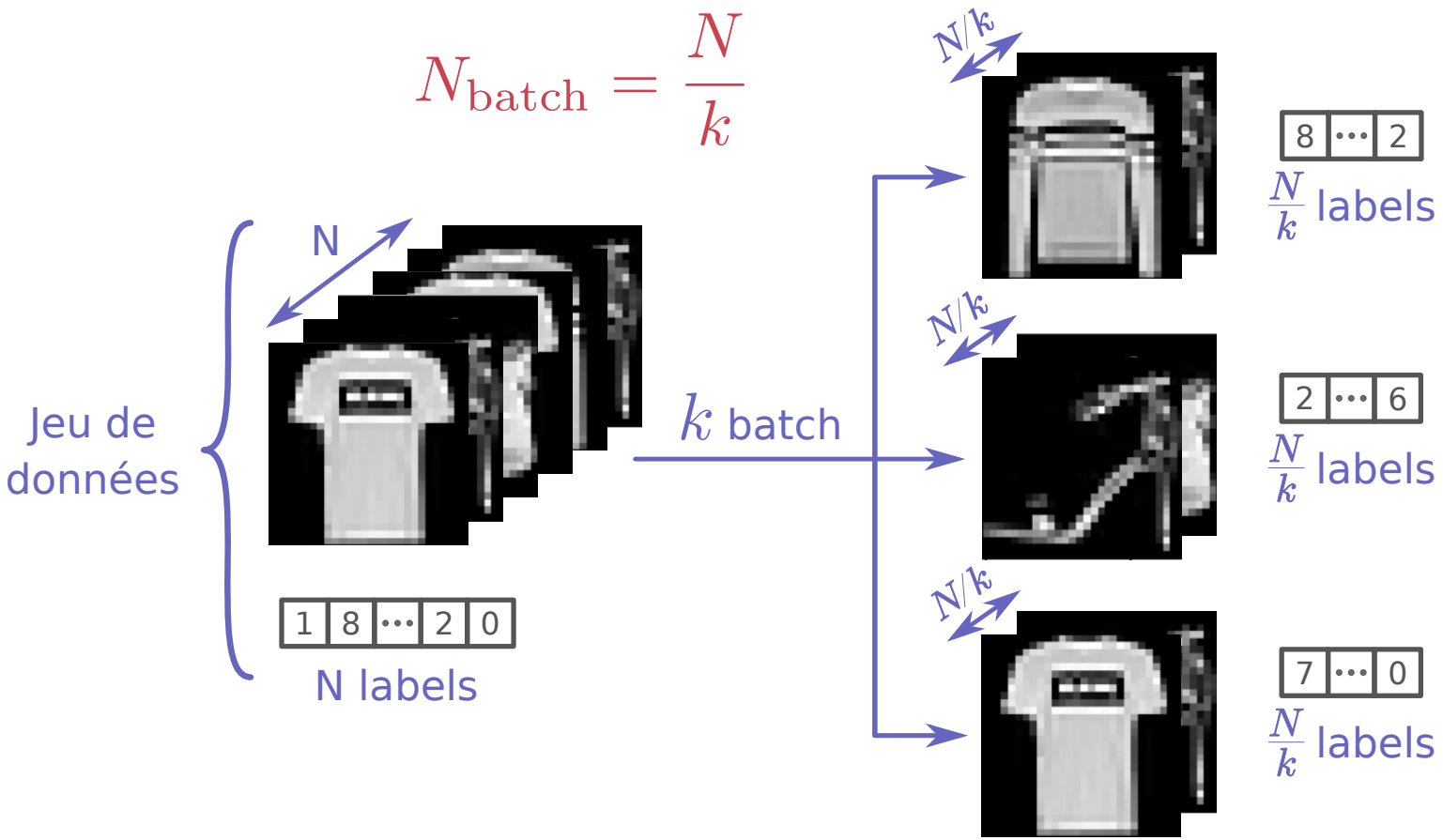


Paramètres :

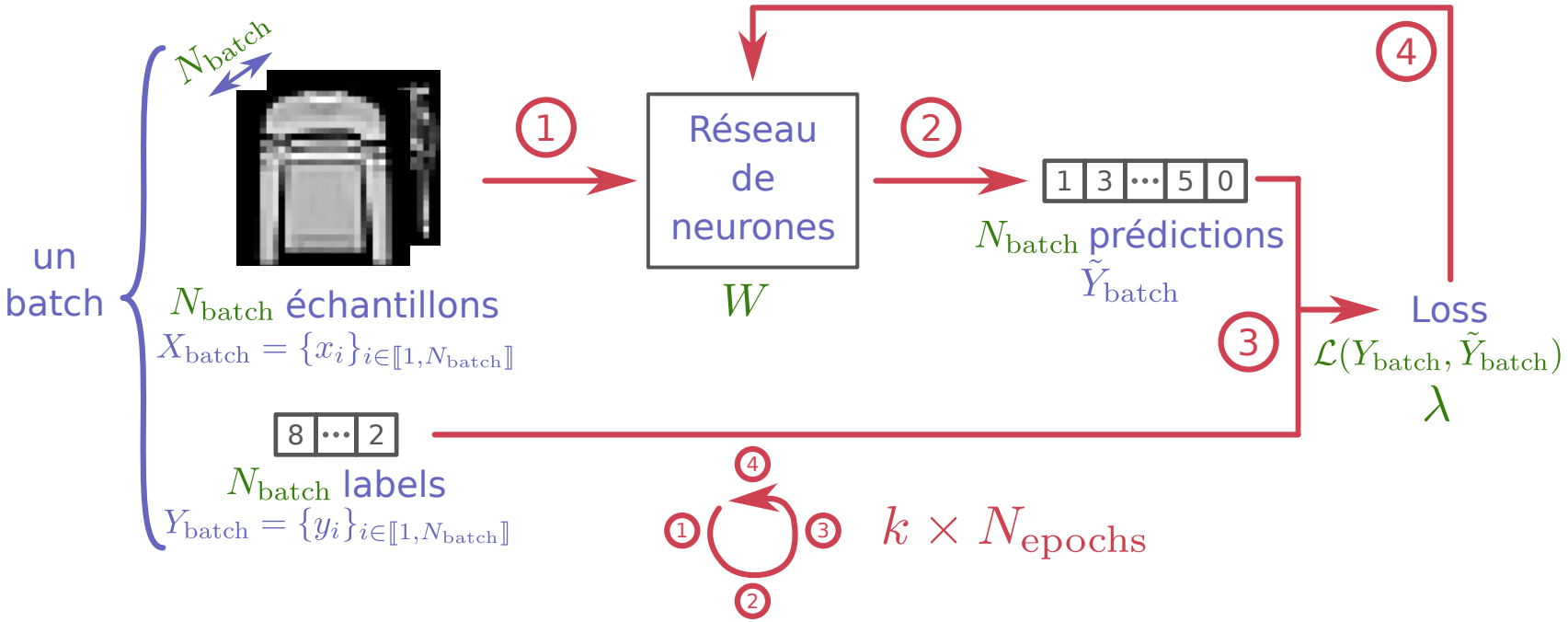
- Paramètres du réseau W (appris durant l'entraînement)
- Hyperparamètres (à fixer avant l'entraînement)
  - Type de Loss $\mathcal{L}(y_i, \tilde{y}_i)$
  - Learning rate λ
  - Nombre d'epochs N_{epochs}
  - Taille de batch N_{batch}

Batch

- **Batch** : Un sous-ensemble d'un jeu de données de taille fixée à l'avance, N_{batch}, sur lequel un réseau est entrainé.



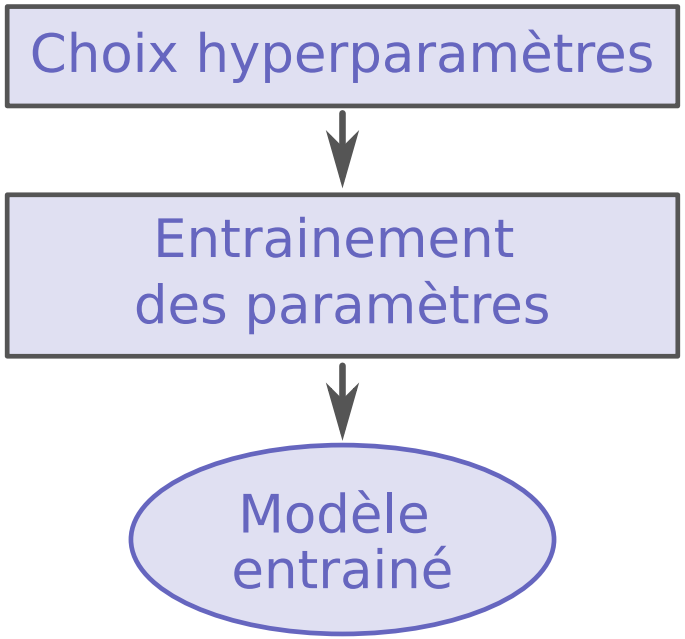
Entraînement par batch



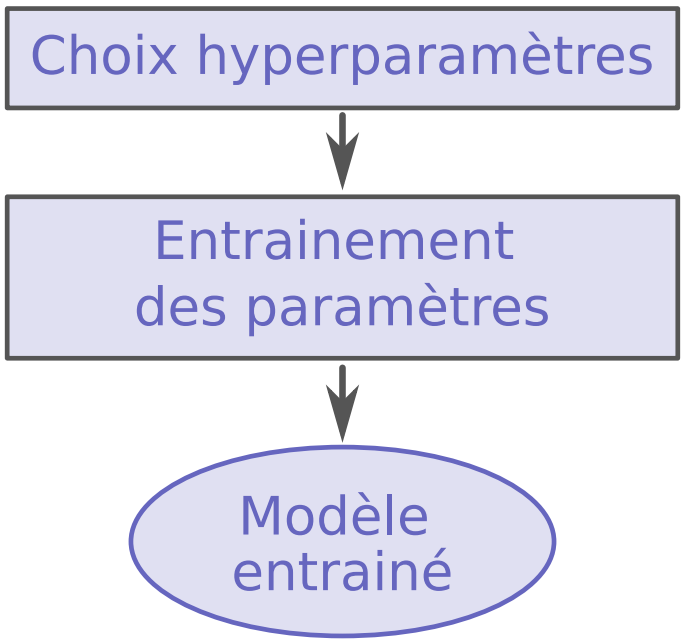
Entraînement par batch :

- Moins couteux en mémoire
- Même nombre de données vues en 1 epoch (*i.e.* N)
  1 epoch = k batch de taille N_{batch}
- Les poids du réseau sont mis à jour k fois plus souvent.
  $\longrightarrow$ La convergence de l'apprentissage est souvent plus rapide.

Étapes d'entraînement



Étapes d'entraînement

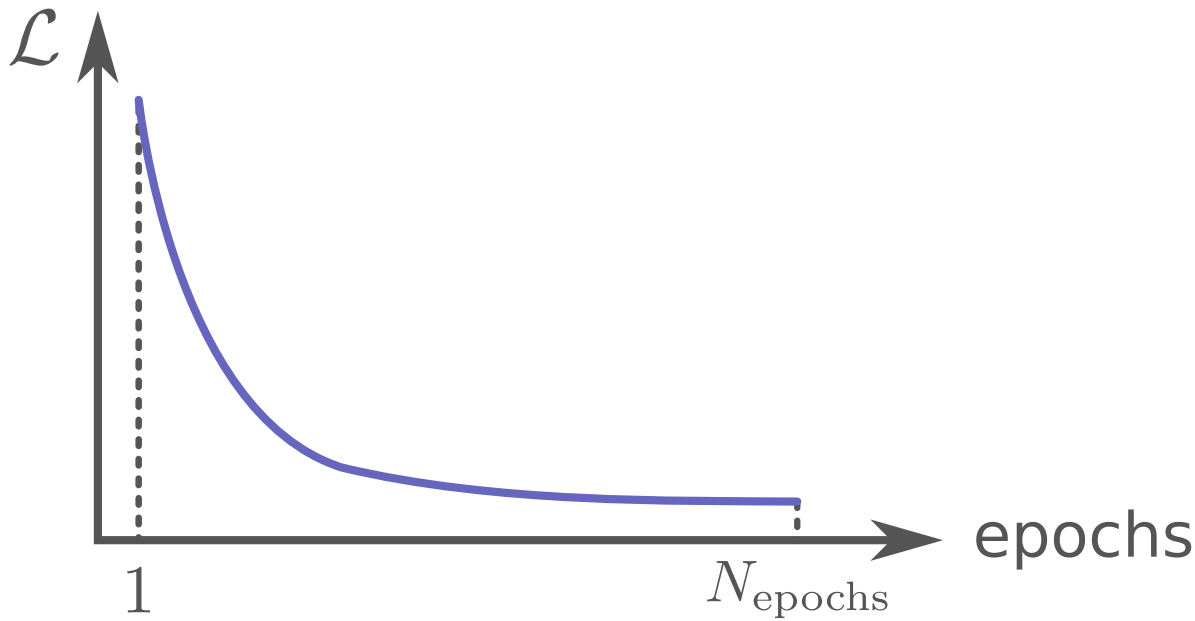


- Comment savoir si mon modèle s'entraîne bien ?

  $\longrightarrow$ **Visualisation de la courbe d'entraînement**

Courbes d'entraînement

Évolution de la loss en fonction du nombre d'epoch d'entraînement



- **Comportement attendu** : Au fil des epochs, la loss doit décroître et converger vers un minimum. Lorsque la loss n'évolue plus, l'entraînement est terminé.

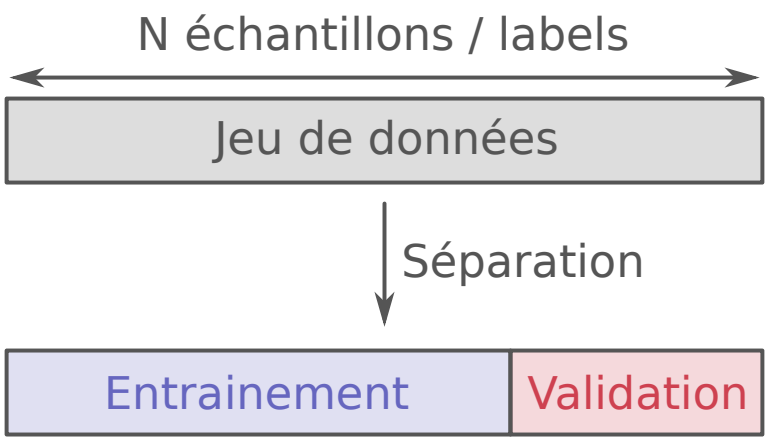
Suivi de l'entraînement - Jeu de validation

- Comment savoir si le modèle entraîné fonctionnera sur des données différentes du jeu d'entraînement ?
  c.-à-d. comment estimer **l'erreur de généralisation** ?

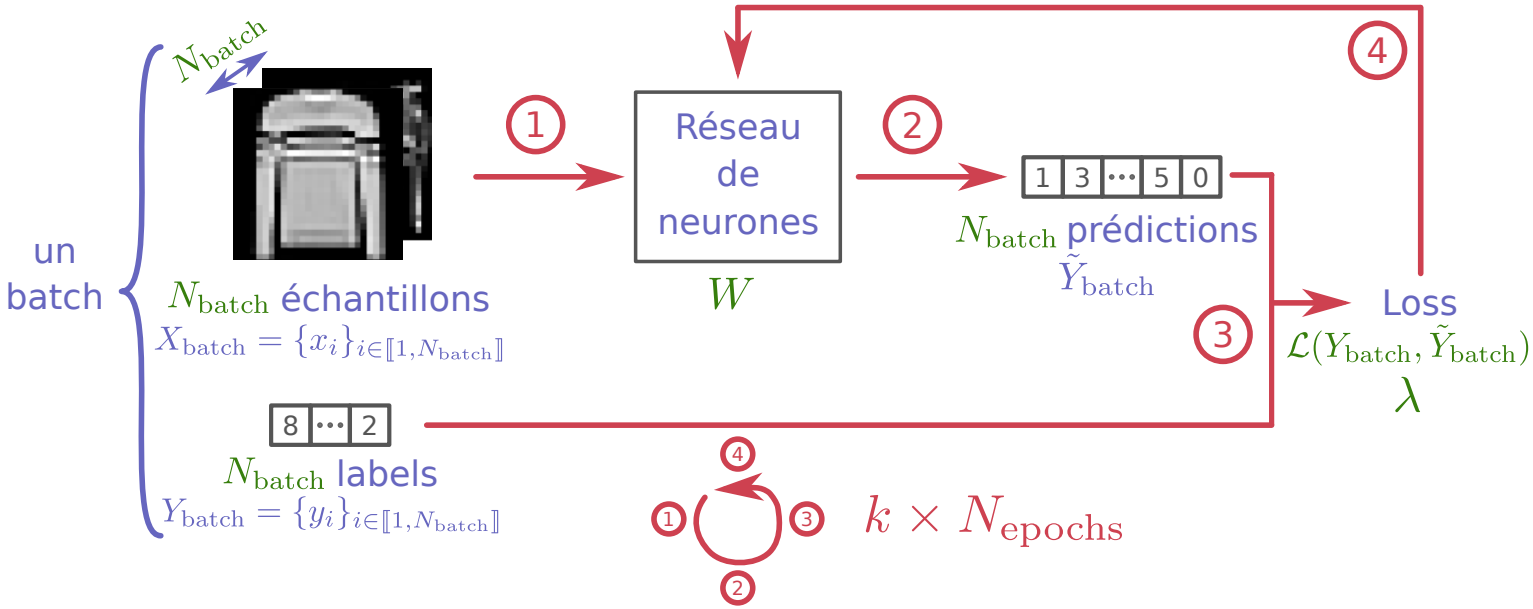
Suivi de l'entraînement - Jeu de validation

- Comment savoir si le modèle entraîné fonctionnera sur des données différentes du jeu d'entraînement ?
  c.-à-d. comment estimer **l'erreur de généralisation** ?

$\longrightarrow$ Appliquer le modèle en cours d'entraînement sur de nouvelles données non utilisées pour l'entraînement : **jeu de validation**.



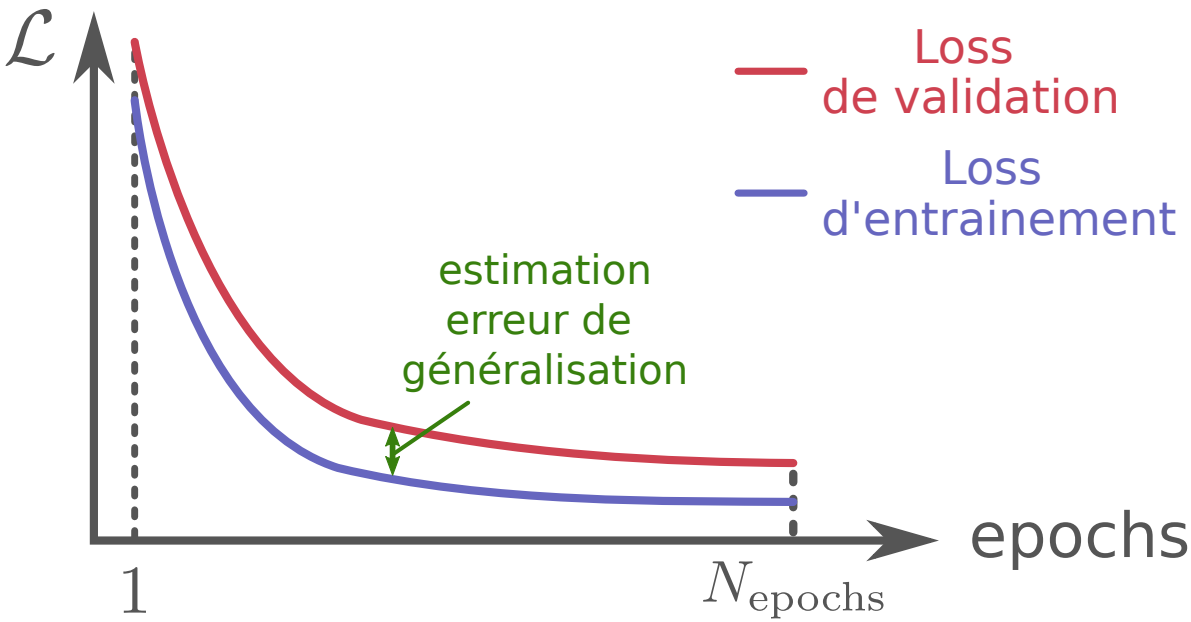
Entraînement avec validation



- Séparation des données en jeu d'entraînement et de validation
- Pour chaque epoch
  - **Pour chaque batch d'entraînement**
    - Prédiction du batch (étapes 1 et 2)
    - Calcul de la loss (étape 3)
    - Mise à jour des poids par rétropropagation (étape 4)
  - **Pour chaque batch de validation**
    - Prédiction du batch (étape 1 et 2)
    - Calcul de la loss (étape 3)

} Pas de rétropropagation !

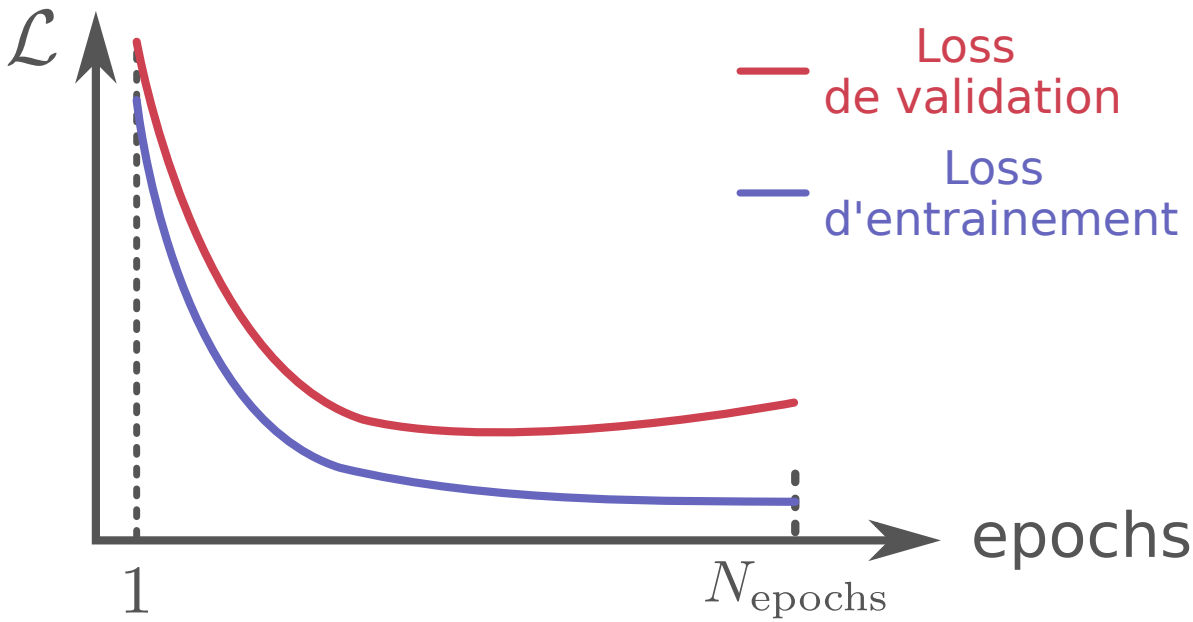
Courbe de validation



- **Comportement attendu** :
  - La loss de validation est plus grande de celle d'entraînement
  - La différence entre les loss d'entraînement et de validation ne doit pas être trop grande pour s'assurer d'avoir un réseau qui généralisera bien.

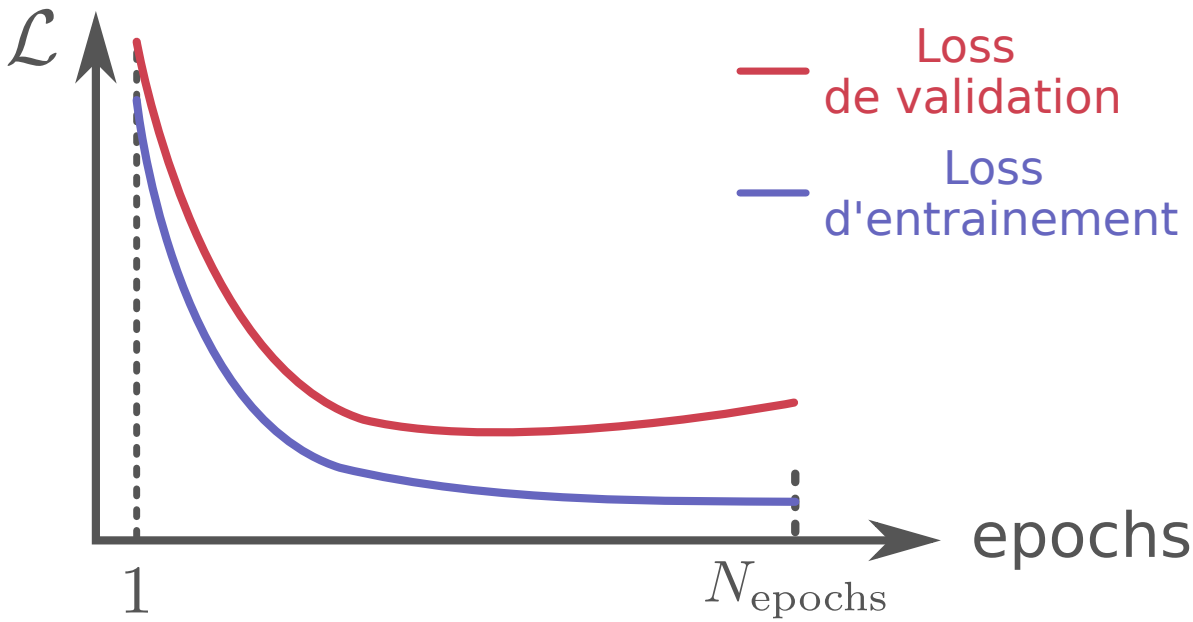
Courbe de validation

- Que se passe-t-il lors de cet apprentissage ?



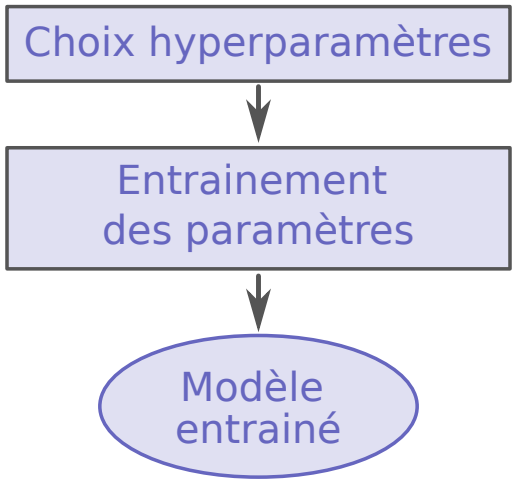
Courbe de validation

- Que se passe-t-il lors de cet apprentissage ?



- La loss de validation augmente à partir d'un certain nombre d'epochs
  $\longrightarrow$ **Surapprentissage** (Overfitting)
- **Surapprentissage** : Le réseau apprend des caractéristiques précises du jeu d'entraînement qui ne se retrouvent pas dans d'autres données (e.g. jeu de validation). Équivalent d'un apprentissage "par coeur".

Étapes d'entraînement



- **Suivi de l'entraînement**

  $\longrightarrow$ Évolution de la loss d'entraînement

- **Suivi de la généralisation du réseau entraîné**

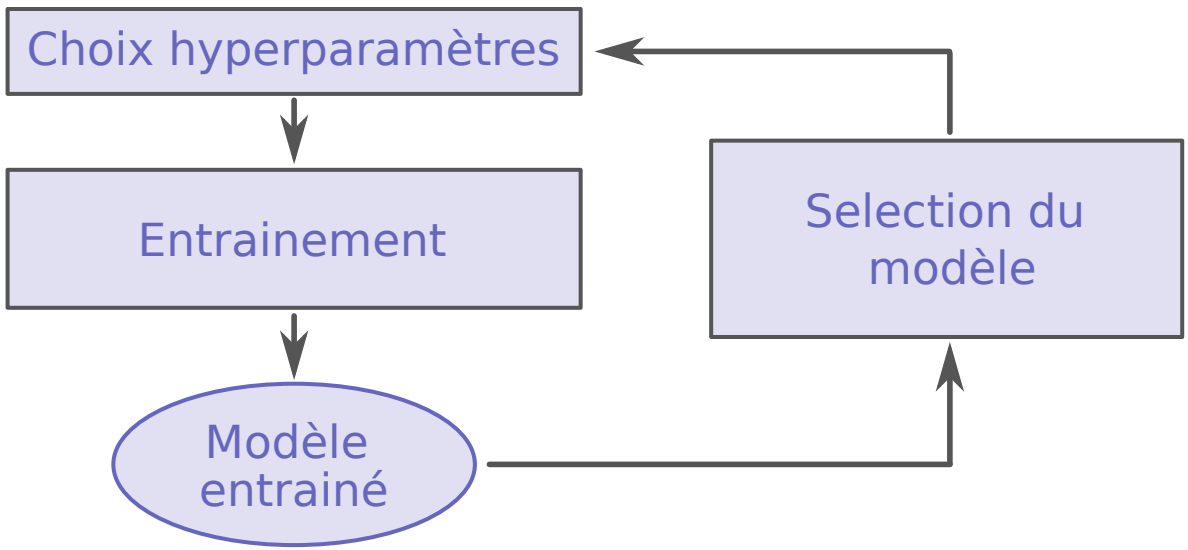
  $\longrightarrow$ Évolution de la loss de validation

Choix d'un modèle

- **Comment obtenir le "meilleur" modèle ?**

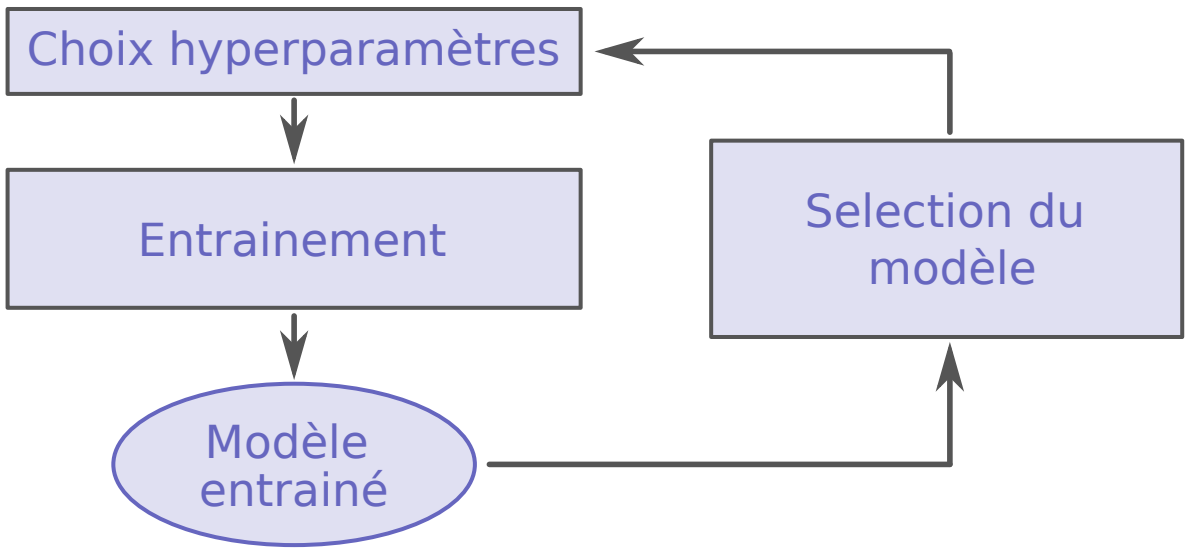
Choix d'un modèle

- **Comment obtenir le "meilleur" modèle ?**
  $\longrightarrow$ Entraînement avec différents jeux d'hyperparamètres et sélection du "meilleur" modèle



Choix d'un modèle

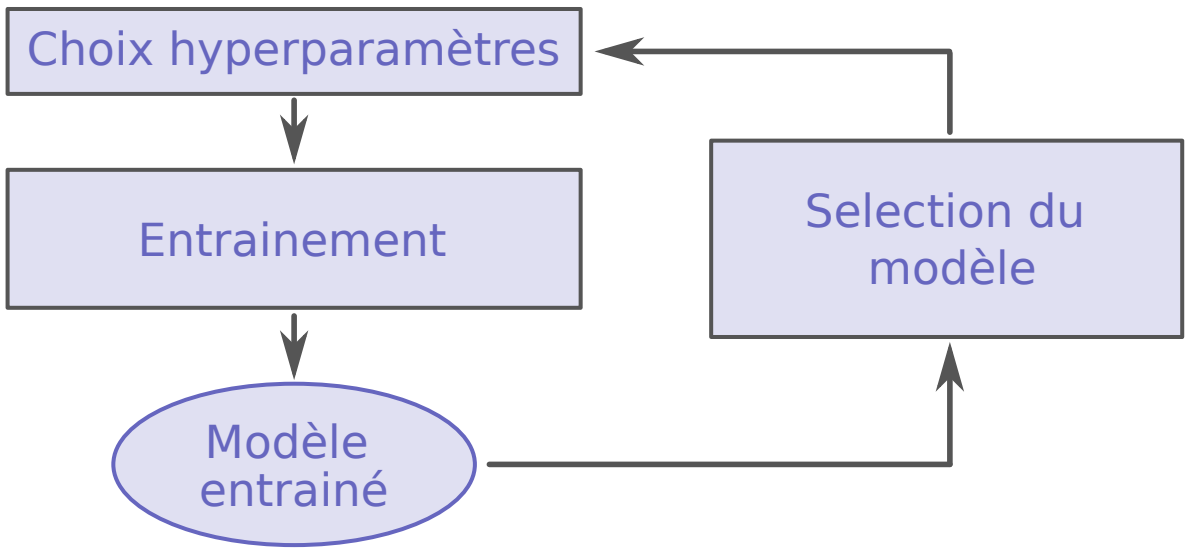
- **Comment obtenir le "meilleur" modèle ?**
  $\longrightarrow$ Entraînement avec différents jeux d'hyperparamètres et sélection du "meilleur" modèle



- **"Meilleur modèle"** : Modèle généralisant le mieux, *i.e.* celui avec la plus faible loss de validation

Choix d'un modèle

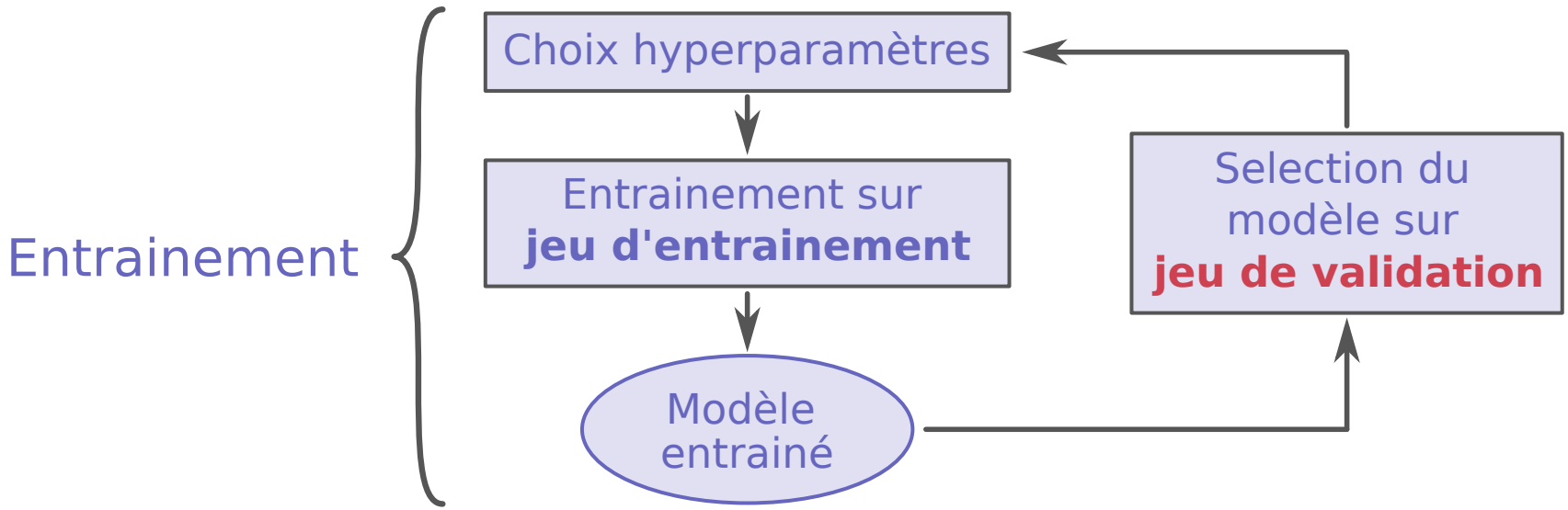
- **Comment obtenir le "meilleur" modèle ?**
  $\longrightarrow$ Entraînement avec différents jeux d'hyperparamètres et sélection du "meilleur" modèle



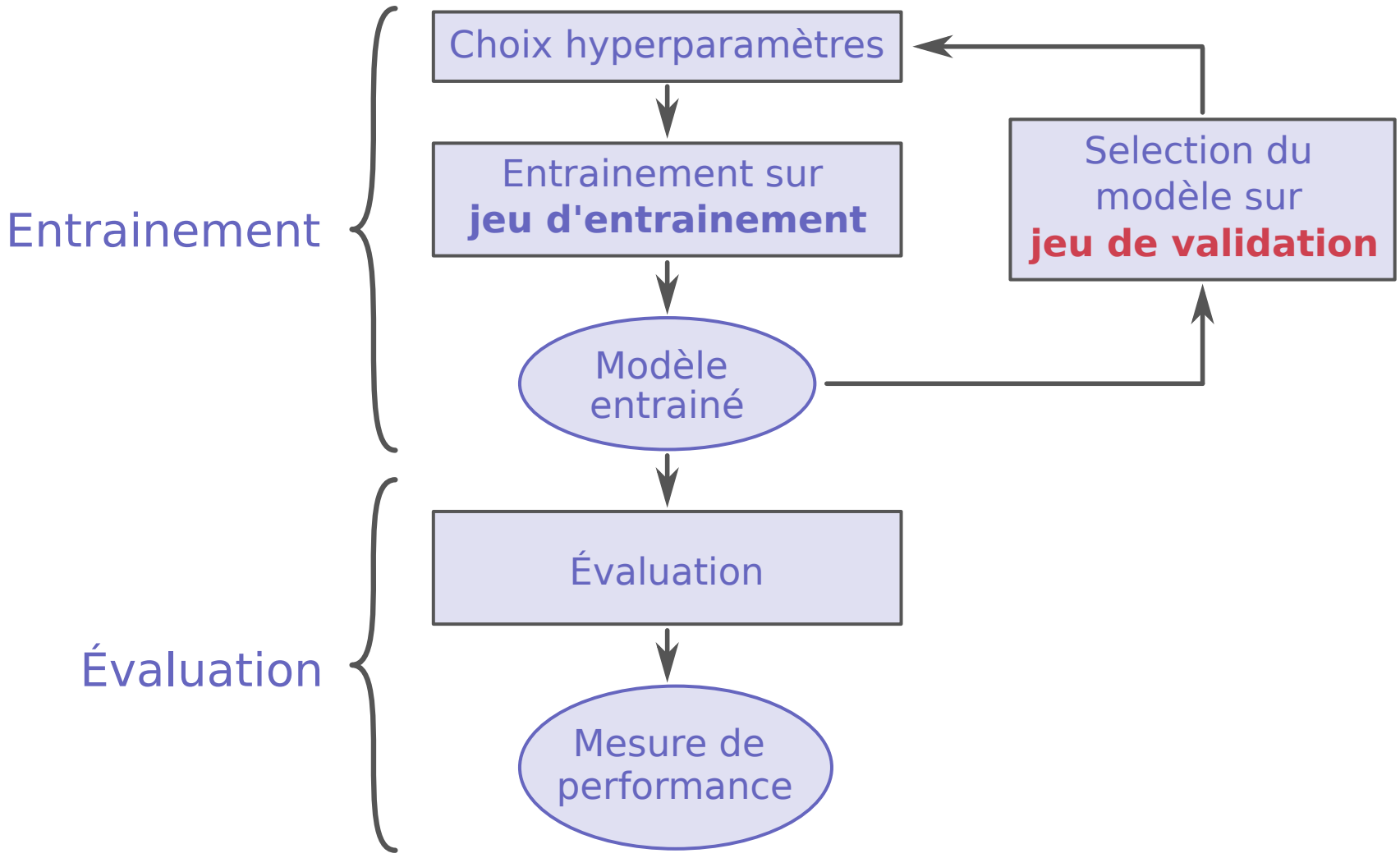
- **"Meilleur modèle"** : Modèle généralisant le mieux, *i.e.* celui avec la plus faible loss de validation
- **Utilisation du jeu de validation** :
  - Suivi de l'entraînement
  - Sélection d'un modèle

Plan

Pipeline d'apprentissage profond



Pipeline d'apprentissage profond



Comment évaluer les performances d'un réseau ?

- Évaluation basée sur des **metriques** choisies en fonction de l'application d'intérêt.

Comment évaluer les performances d'un réseau ?

- Évaluation basée sur des **metriques** choisies en fonction de l'application d'intérêt.

Exemples de métriques classiques :

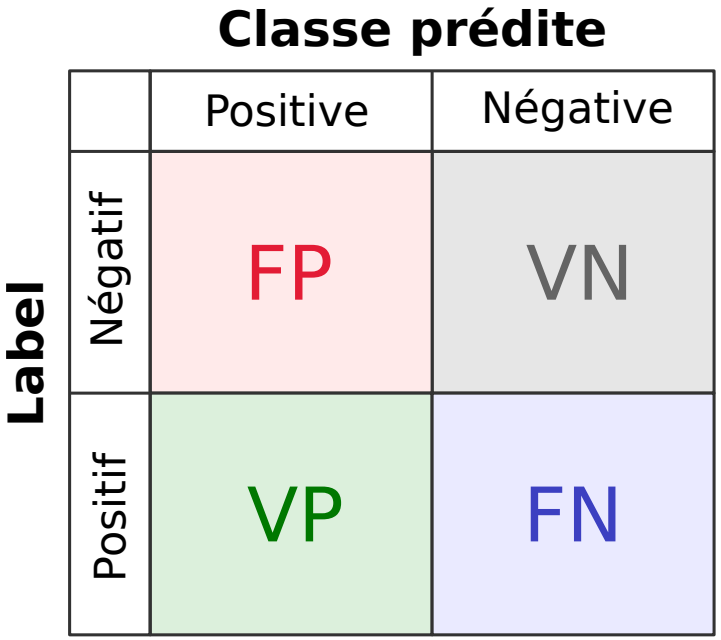
- **Métriques de régression**
  - Erreur quadratique moyenne (MSE)
  - Peak Signal-to-Noise Ratio (PSNR)
  - Structural Similarity (SSIM)

- **Métriques de classification**
  - Accuracy
  - Dice / F1
  - Intersection over union (IoU)
  - Sensibilité
  - Spécificité
  - Précision

  } Basées sur la **Matrice de confusion**

Matrice de confusion binaire

4 types de résultat possibles pour une tâche de **classification binaire** :



- **FP**: Faux positif
- **VN**: Vrai négatif
- **VP**: Vrai positif
- **FN**: Faux négatif

Matrice de confusion binaire

4 types de résultat possibles pour une tâche de **classification binaire** :

		Classe prédite	
		Positive	Négative
Label	Négatif	FP	VN
	Positif	VP	FN

- **FP**: Faux positif
- **VN**: Vrai négatif
- **VP**: Vrai positif
- **FN**: Faux négatif

Exemple de métriques :

$$Acc = \frac{VP + VN}{VP + VN + FP + FN}$$

$$F1 = \frac{2 \times VP}{2VP + FP + FN}$$

$$\text{sensibilité} = \frac{VP}{VP + FN}$$

$$\text{spécificité} = \frac{VN}{VN + FP}$$

Choix et interprétation des métriques

Test 1

		Classe prédite : Positive	Classe prédite : Négative
Label	Négatif	25	25
	Positif	50	0

$$\text{Acc} = \frac{50 + 25}{100} = 75\%$$

$$\text{sensibilité} = \frac{50}{50} = 100\%$$

$$\text{spécificité} = \frac{25}{50} = 50\%$$

Test 2

		Classe prédite : Positive	Classe prédite : Négative
Label	Négatif	0	50
	Positif	25	25

$$\text{Acc} = \frac{25 + 50}{100} = 75\%$$

$$\text{sensibilité} = \frac{25}{50} = 50\%$$

$$\text{spécificité} = \frac{50}{50} = 100\%$$

Choix et interprétation des métriques

Test 1

Classe prédite

Label		Positive	Négative
	Négatif	25	25
	Positif	50	0

$$\text{Acc} = \frac{50+25}{100} = 75\%$$

$$\text{sensibilité} = \frac{50}{50} = 100\%$$

$$\text{spécificité} = \frac{25}{50} = 50\%$$

Test 2

Classe prédite

Label		Positive	Négative
	Négatif	0	50
	Positif	25	25

$$\text{Acc} = \frac{25+50}{100} = 75\%$$

$$\text{sensibilité} = \frac{25}{50} = 50\%$$

$$\text{spécificité} = \frac{50}{50} = 100\%$$

Quel test choisir pour les applications suivantes ?

- **Test de dépistage du COVID**

- **Test traitement chimiothérapie**

Choix et interprétation des métriques

Test 1

Label	Classe prédite : Positive	Classe prédite : Négative
Négatif	25	25
Positif	50	0

$$\text{Acc} = \frac{50+25}{100} = 75\%$$

$$\text{sensibilité} = \frac{50}{50} = 100\%$$

$$\text{spécificité} = \frac{25}{50} = 50\%$$

Test 2

Label	Classe prédite : Positive	Classe prédite : Négative
Négatif	0	50
Positif	25	25

$$\text{Acc} = \frac{25+50}{100} = 75\%$$

$$\text{sensibilité} = \frac{25}{50} = 50\%$$

$$\text{spécificité} = \frac{50}{50} = 100\%$$

Quel test choisir pour les applications suivantes ?

- **Test de dépistage du COVID** $\longrightarrow$ Test 1
  - Important de détecter tous les positifs
  - Moins grave de surdétecter des négatifs
- **Test traitement chimiothérapie** $\longrightarrow$ Test 2
  - Important de ne pas donner le traitement à des négatifs (très nocif)

Matrice de confusion multi-classes

Dans le cas d'une tâche de **classification multi-classes** :

p_{ij} : pourcentage d'échantillons de classe i dont la prédiction est classe j

Classe prédite

Label

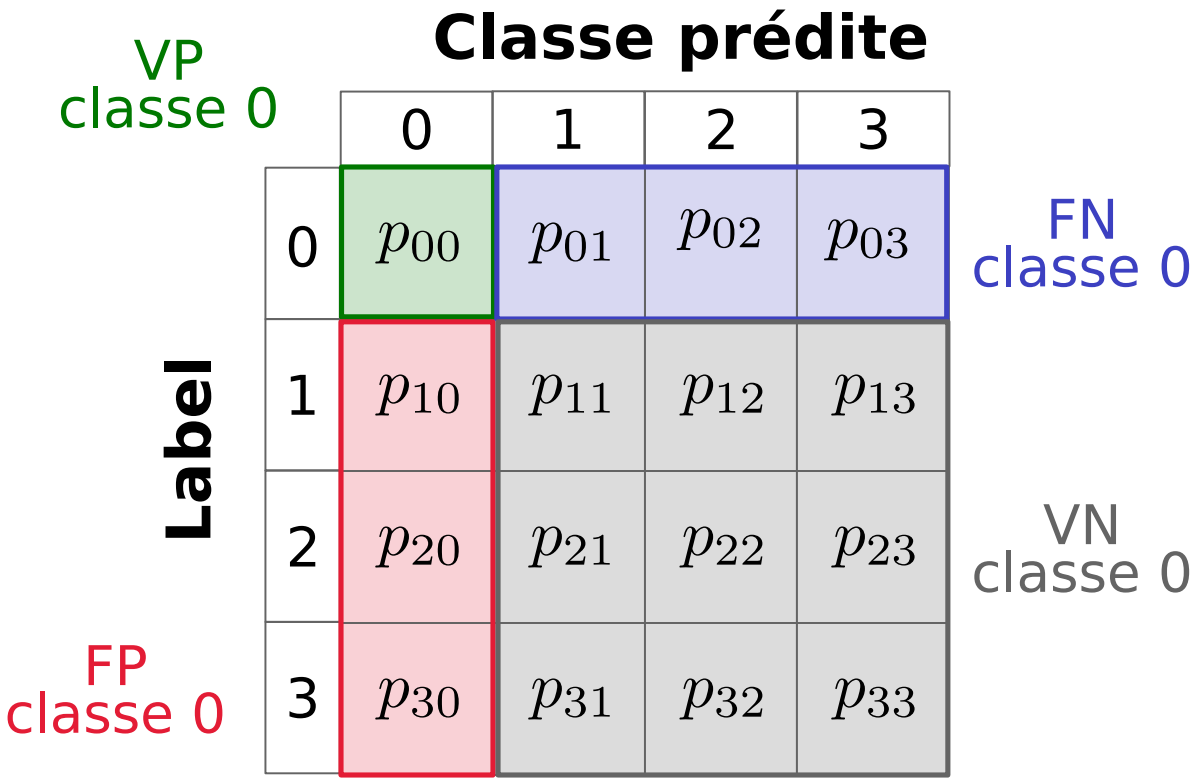
	0	1	2	3
0	p_{00}	p_{01}	p_{02}	p_{03}
1	p_{10}	p_{11}	p_{12}	p_{13}
2	p_{20}	p_{21}	p_{22}	p_{23}
3	p_{30}	p_{31}	p_{32}	p_{33}

$$\sum_i p_{ij} = 1$$

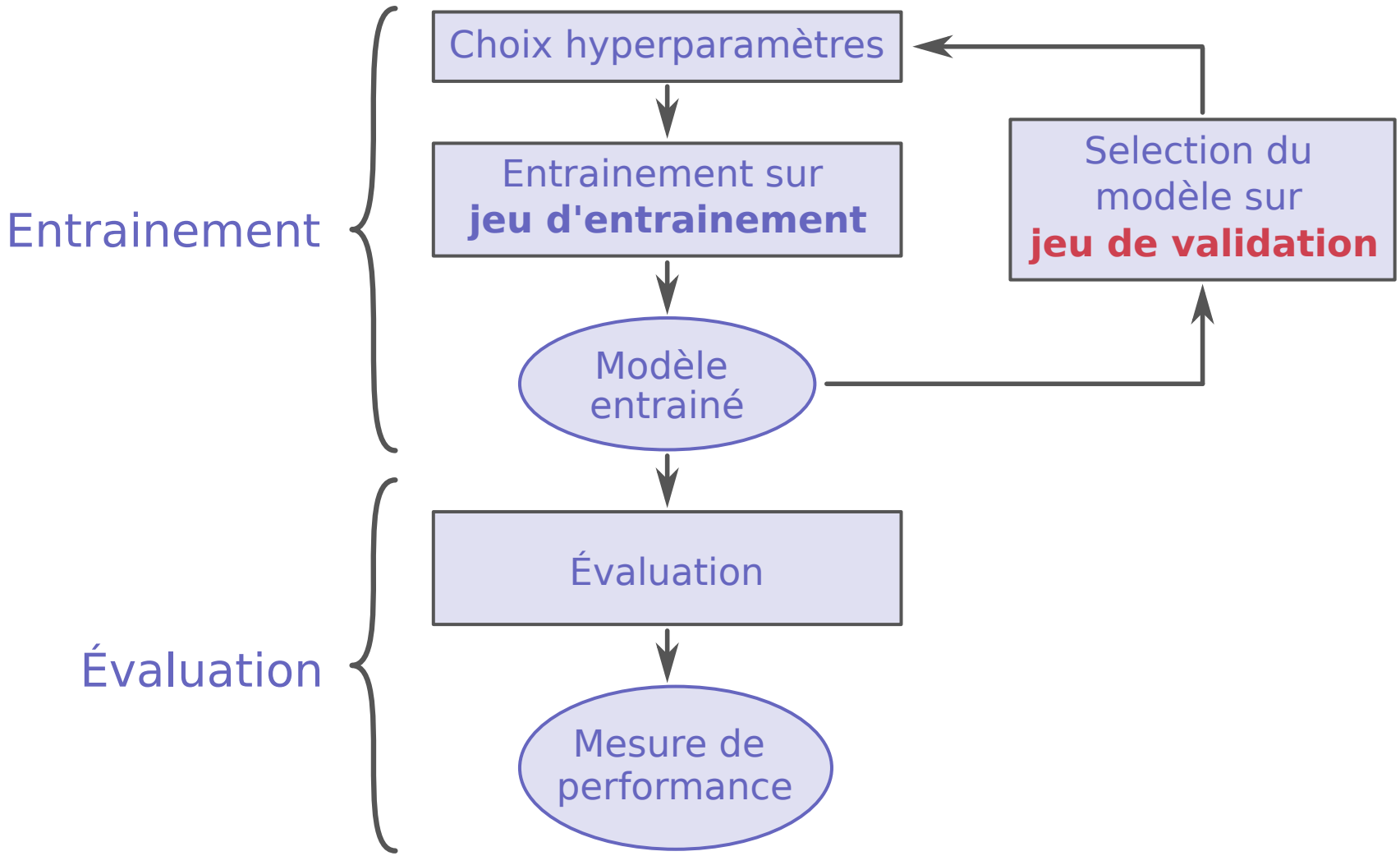
Matrice de confusion multi-classes

Dans le cas d'une tâche de **classification multi-classes** :

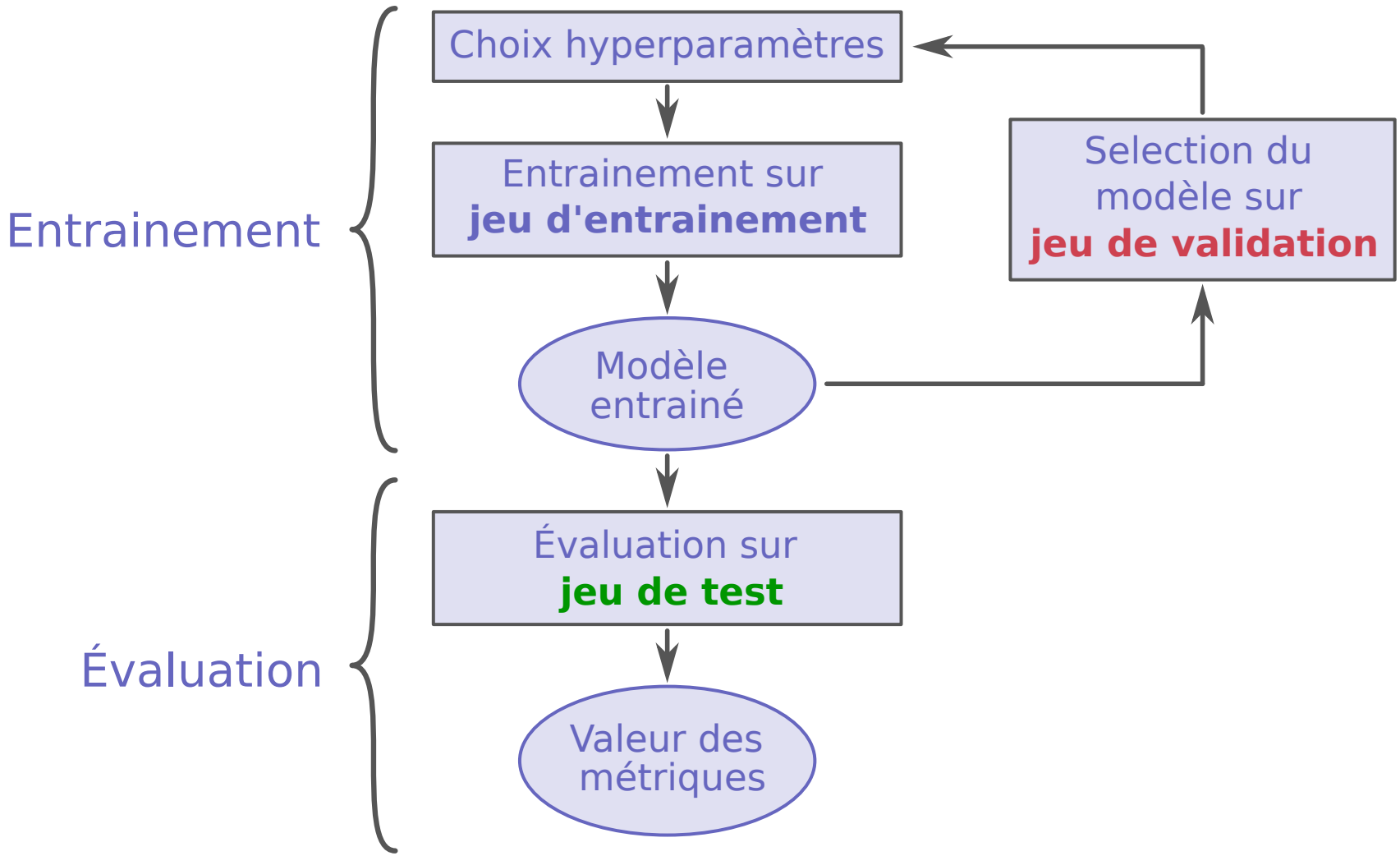
p_{ij} : pourcentage d'échantillons de classe i dont la prédiction est classe j



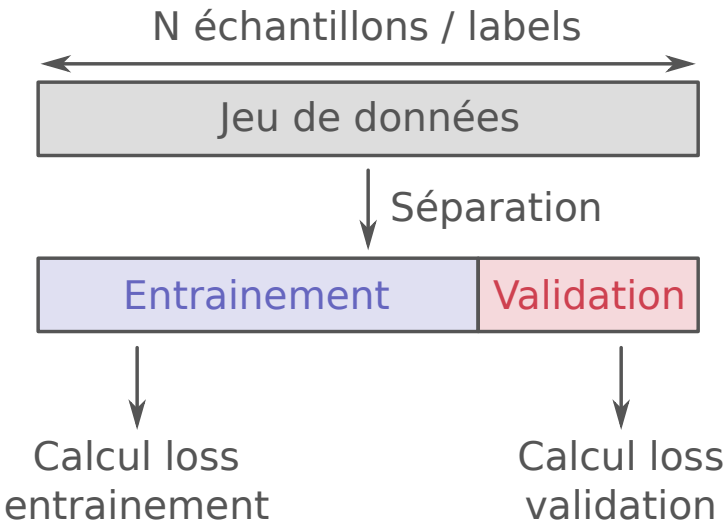
Pipeline d'apprentissage profond

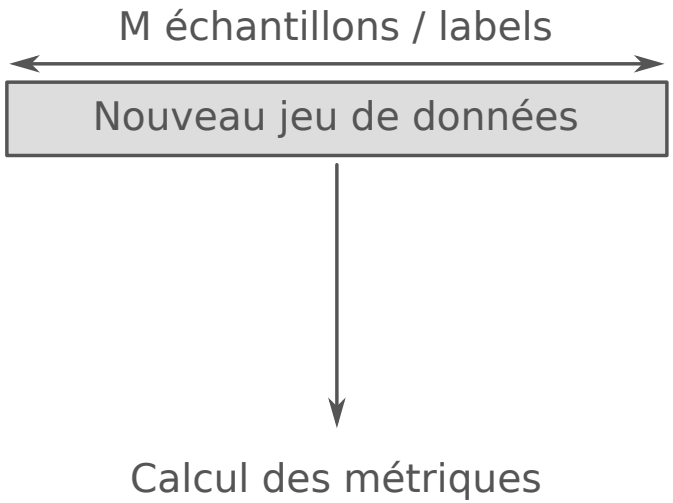


Pipeline d'apprentissage profond

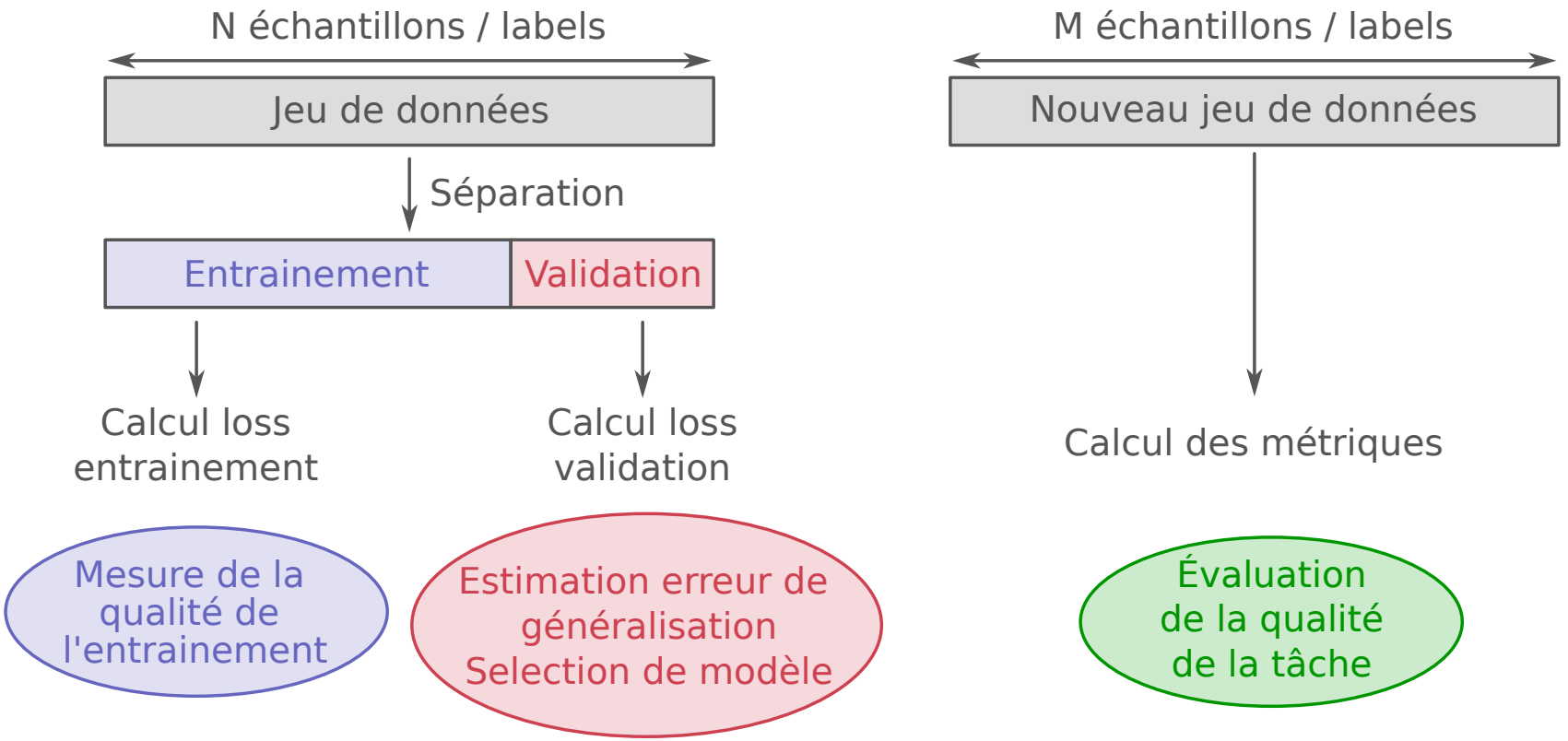


Séparation des données



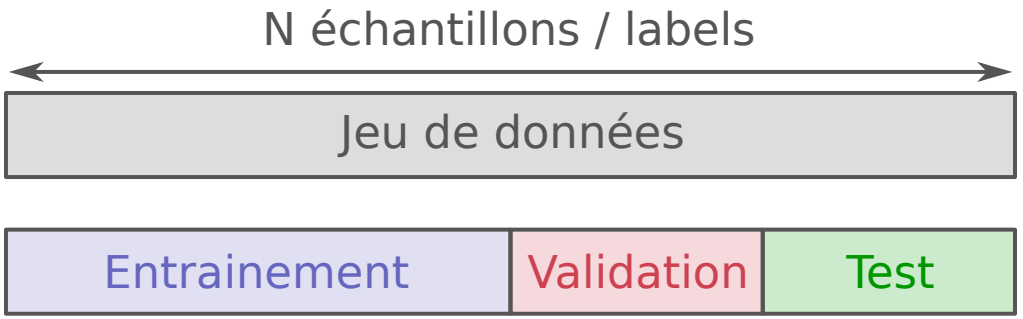


Séparation des données



Séparation des données

- En pratique, on dispose souvent d'un seul jeu de données
- Séparation en 3 selon des proportions fixées. Par exemple :
  - 70% jeu d'entrainement
  - 10% jeu de validation
  - 20% jeu de test

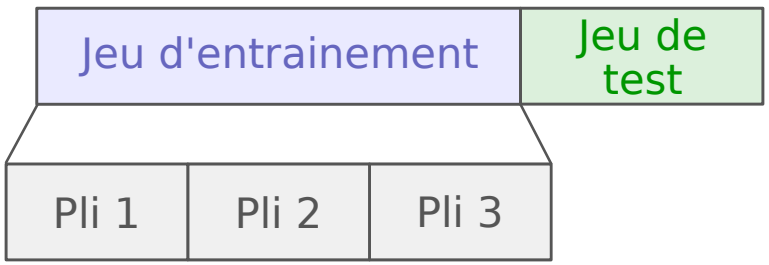


- Problèmes :
  - Réduit la quantité de données disponible pour l'entraînement
  - Pas d'estimation de la variabilité des résultats en fonction de la répartition entraînement/validation

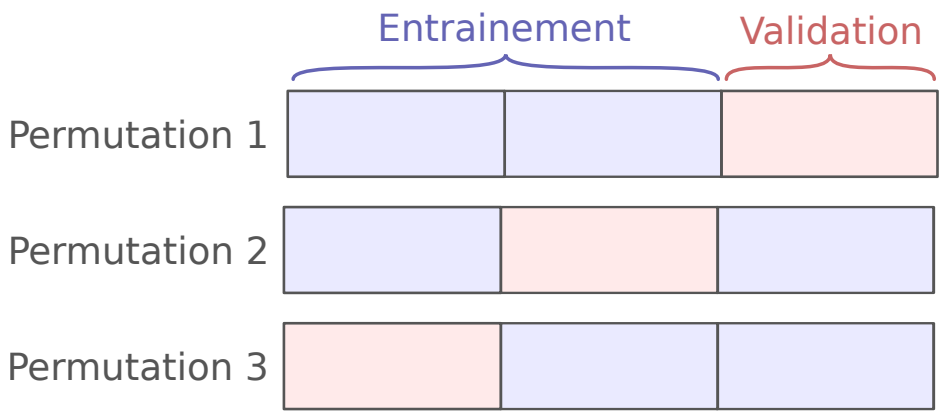
$\longrightarrow$ **Validation croisée**

Validation croisée

- Séparation du jeu de données en 2 : entraînement et test.
  Jeu de test conservé à part
  Séparation du jeu d'entraînement en k plis.

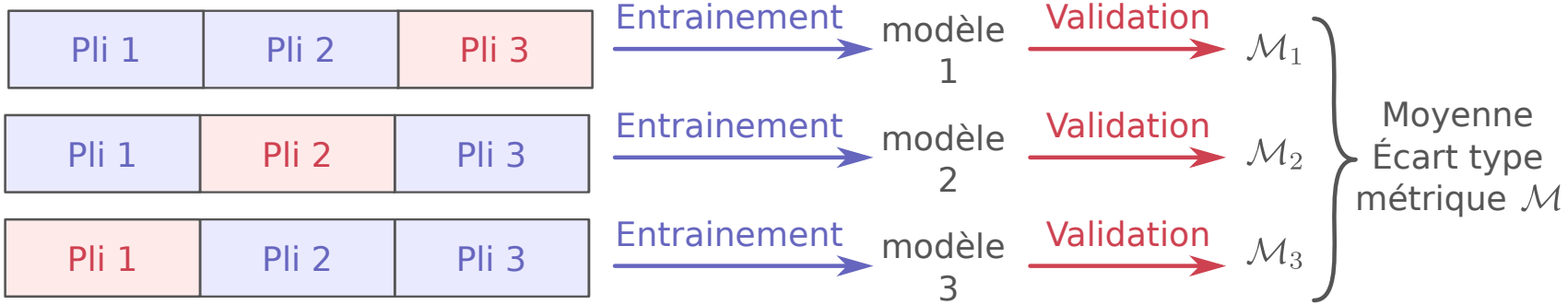


- Génération de k permutations composées d'un pli de validation et de $k-1$ plis d'entraînement.



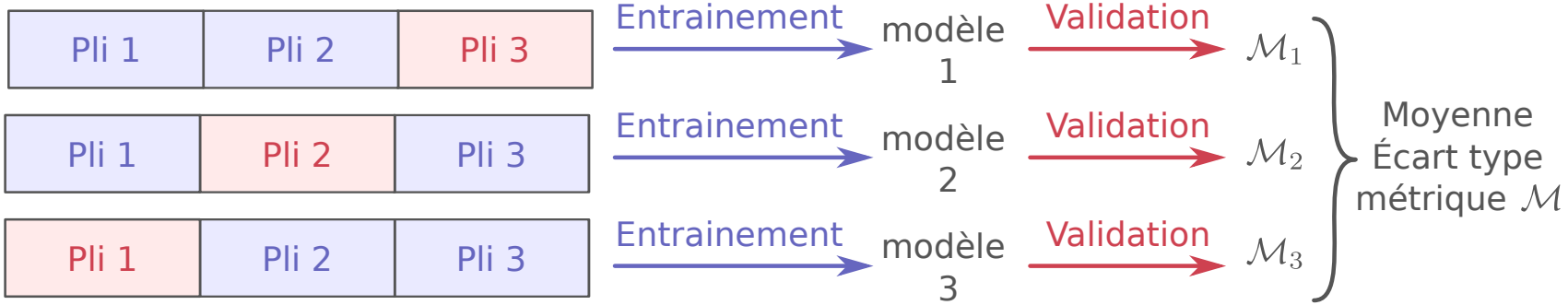
Validation croisée

- Pour chaque permutation
  - Entraînements sur les plis d'entraînements
  - Calcul des métriques sur le pli de validation
- Calcul de la moyenne et de l'écart-type des métriques sur les k permutations



Validation croisée

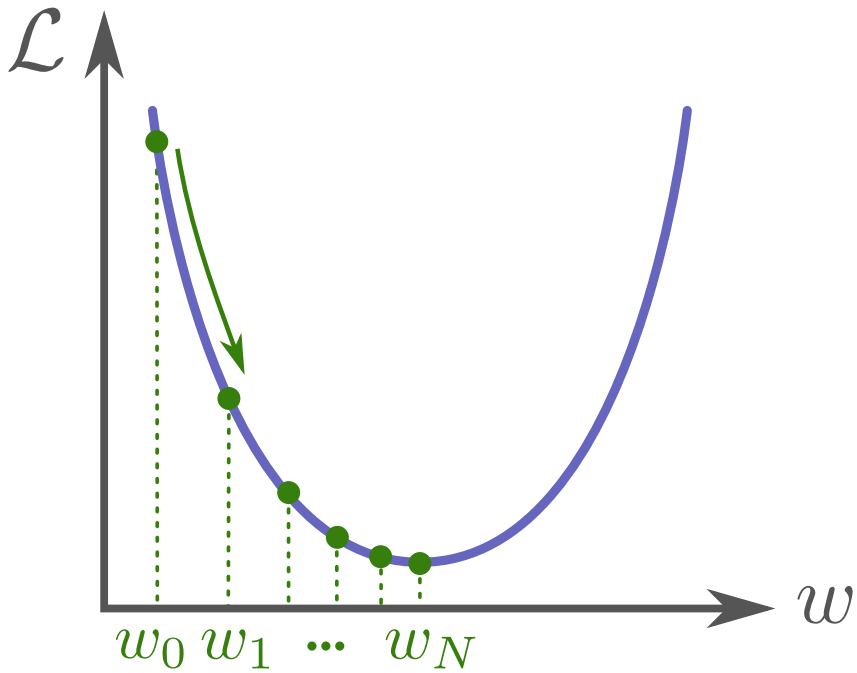
- Pour chaque permutation
  - Entraînements sur les plis d'entraînements
  - Calcul des métriques sur le pli de validation
- Calcul de la moyenne et de l'écart-type des métriques sur les k permutations



- Sélection de modèle par validation croisée :
  - Répétition du processus pour différents jeux d'hyperparamètres
  - Sélection du meilleur jeu d'hyperparamètres
  - Entrainement d'un modèle avec ce jeu d'hyperparamètres sur tout le jeu d'entrainement
- Évaluation finale sur le jeu de test

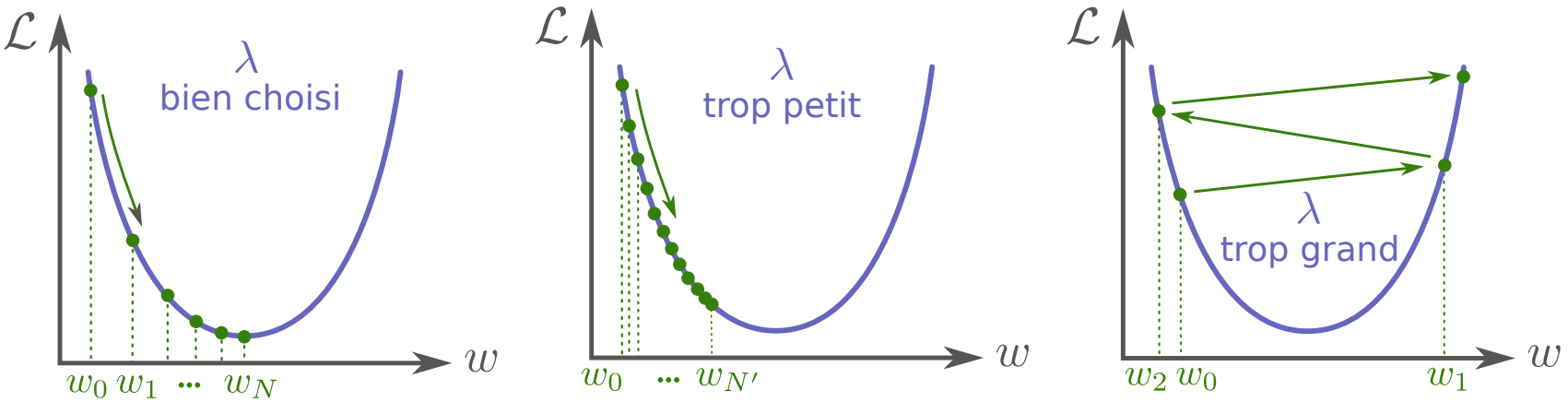
Plan

Influence du learning rate



Learning rate, λ, influe sur la convergence de l'apprentissage.

Influence du learning rate

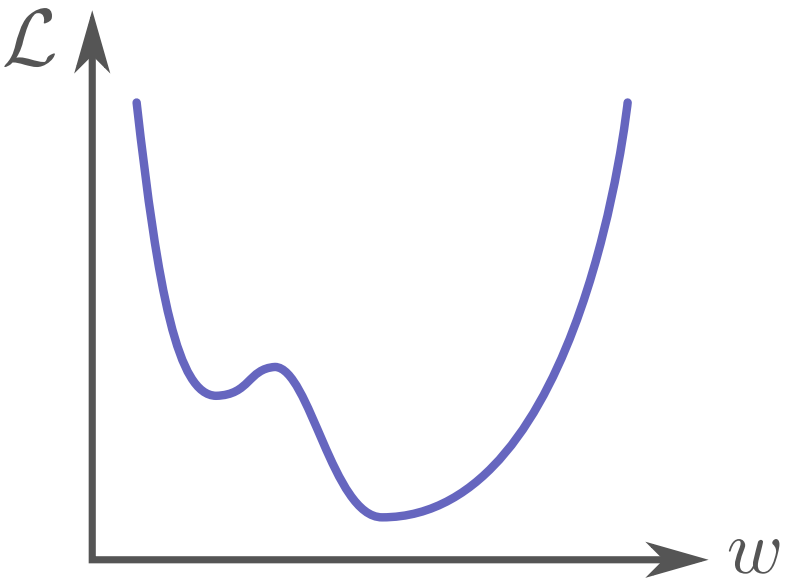


Learning rate, λ, influe sur la convergence de l'apprentissage :

- λ petit $\longrightarrow$ convergence lente
- λ grand $\longrightarrow$ l'apprentissage peut ne pas converger vers une solution

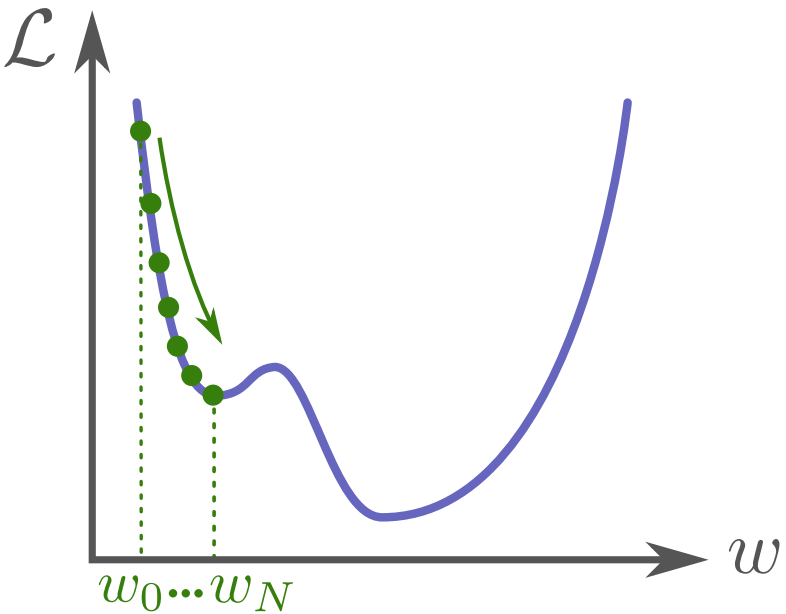
Influence du learning rate - cas non convexe

En pratique, la loss n'est pas une fonction convexe et présente donc plusieurs minima locaux.



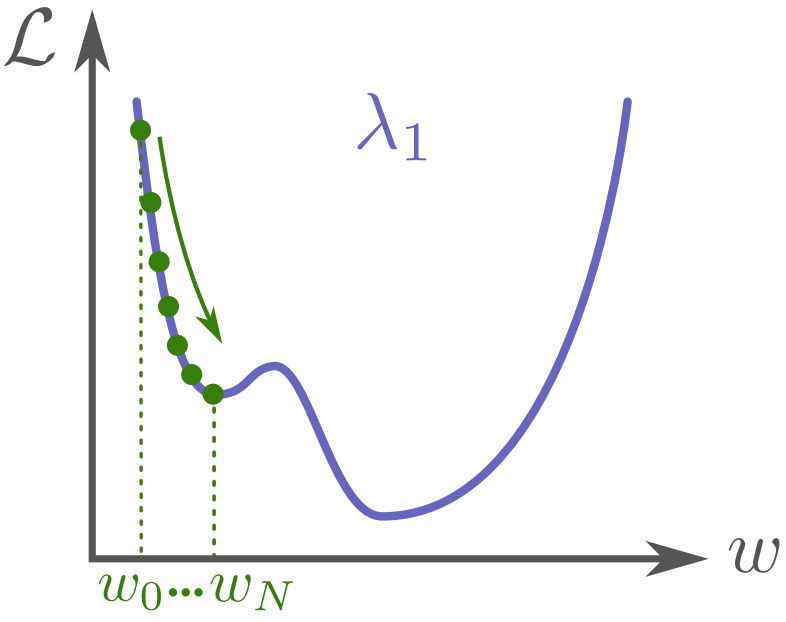
Influence du learning rate - cas non convexe

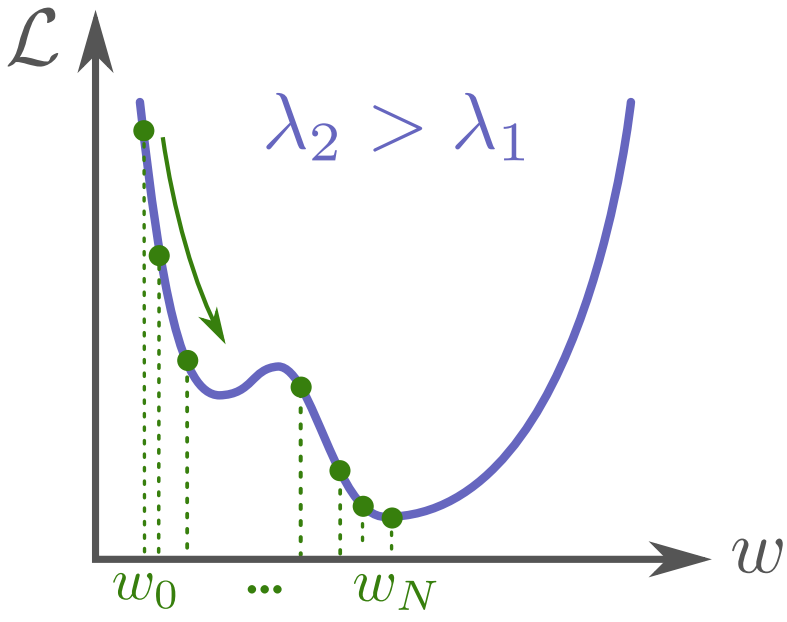
En pratique, la loss n'est pas une fonction convexe et présente donc plusieurs minima locaux.



Influence du learning rate - cas non convexe

En pratique, la loss n'est pas une fonction convexe et présente donc plusieurs minima locaux.





Learning rate, λ, influe sur la convergence de l'apprentissage :

- λ petit $\longrightarrow$ convergence lente, l'apprentissage peut rester coincé dans un minimum local
- λ grand $\longrightarrow$ Apprentissage plus instable, peut ne pas converger
  Exploration d'un espace plus grand de paramètres (et de solutions) possible

Learning rate decay

- **Learning rate decay** : Décroissance du learning rate au cours des epochs.

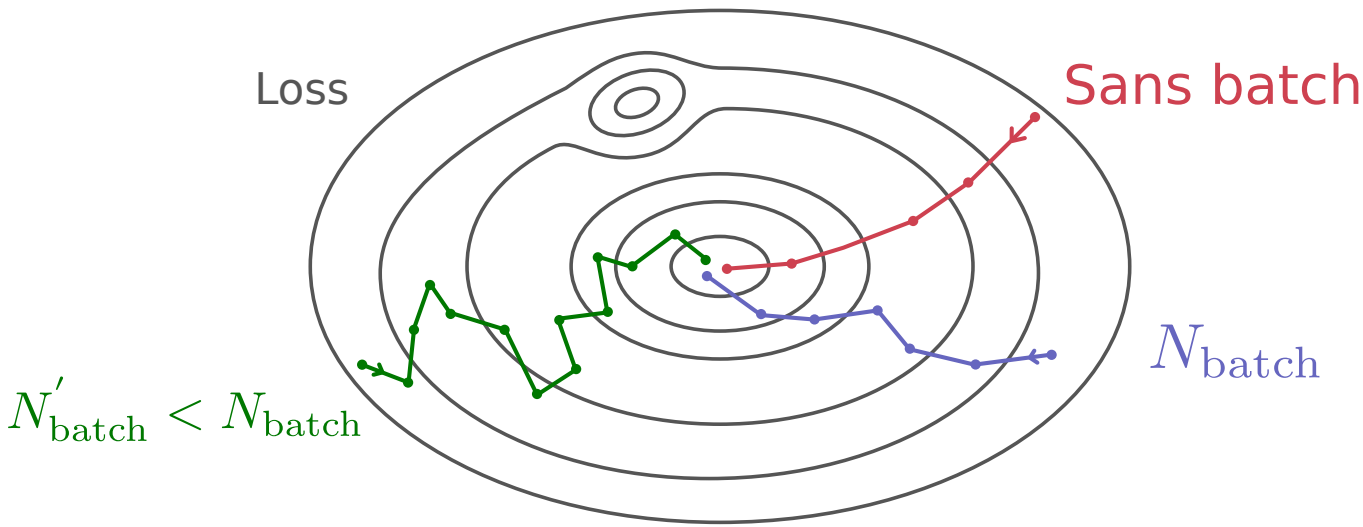
- **Intérêt** :
  - Début d'apprentissage : Convergence plus rapide et potentiellement exploration d'un espace plus grand de paramètres
  - Fin d'apprentissage : Convergence plus fine vers un minimum, stabilisation de l'entraînement.

Learning rate decay

- **Learning rate decay** : Décroissance du learning rate au cours des epochs.
- **Intérêt** :
  - Début d'apprentissage : Convergence plus rapide et potentiellement exploration d'un espace plus grand de paramètres
  - Fin d'apprentissage : Convergence plus fine vers un minimum, stabilisation de l'entraînement.
- **Plusieurs stratégies** :
  - Décroissance en fonction du nombre d'epoch (linéairement, exponentiellement...)
  - Décroissance lorsque la loss de validation stagne
- Remarque : Certains optimiseurs adaptent automatiquement le learning rate durant l'optimisation (*e.g.* Adam)

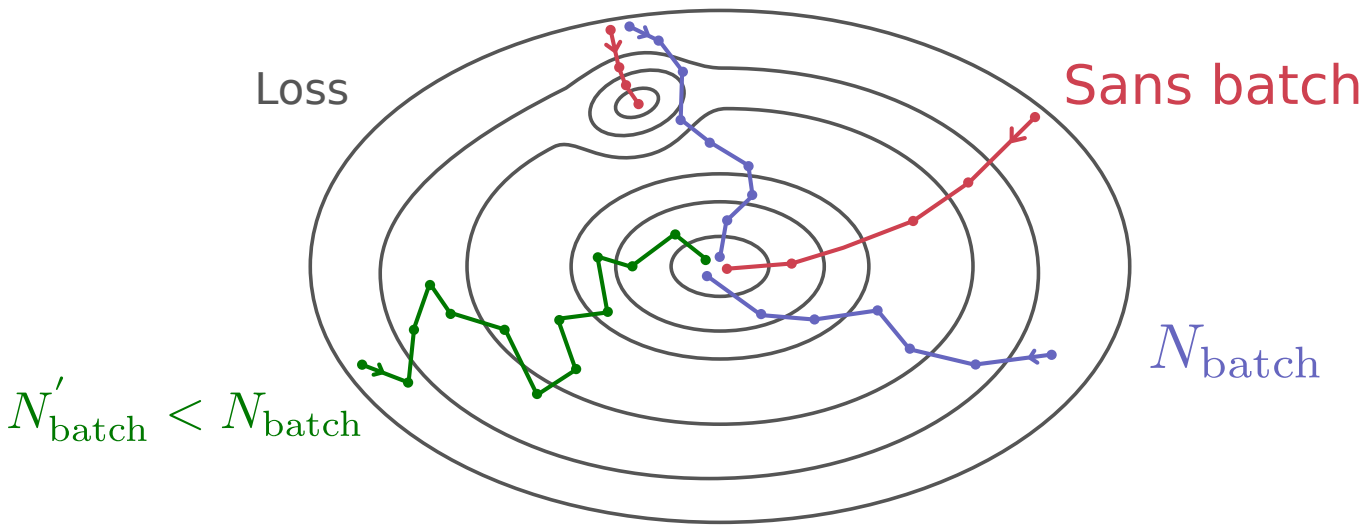
Taille de batch N_{batch}

- Influe principalement sur la quantité de mémoire utilisée et la stabilité de l'apprentissage.
  - Petite taille de batch : Moins de mémoire utilisée, apprentissage plus instable
  - Grande taille de batch : Plus de mémoire utilisée, apprentissage plus stable



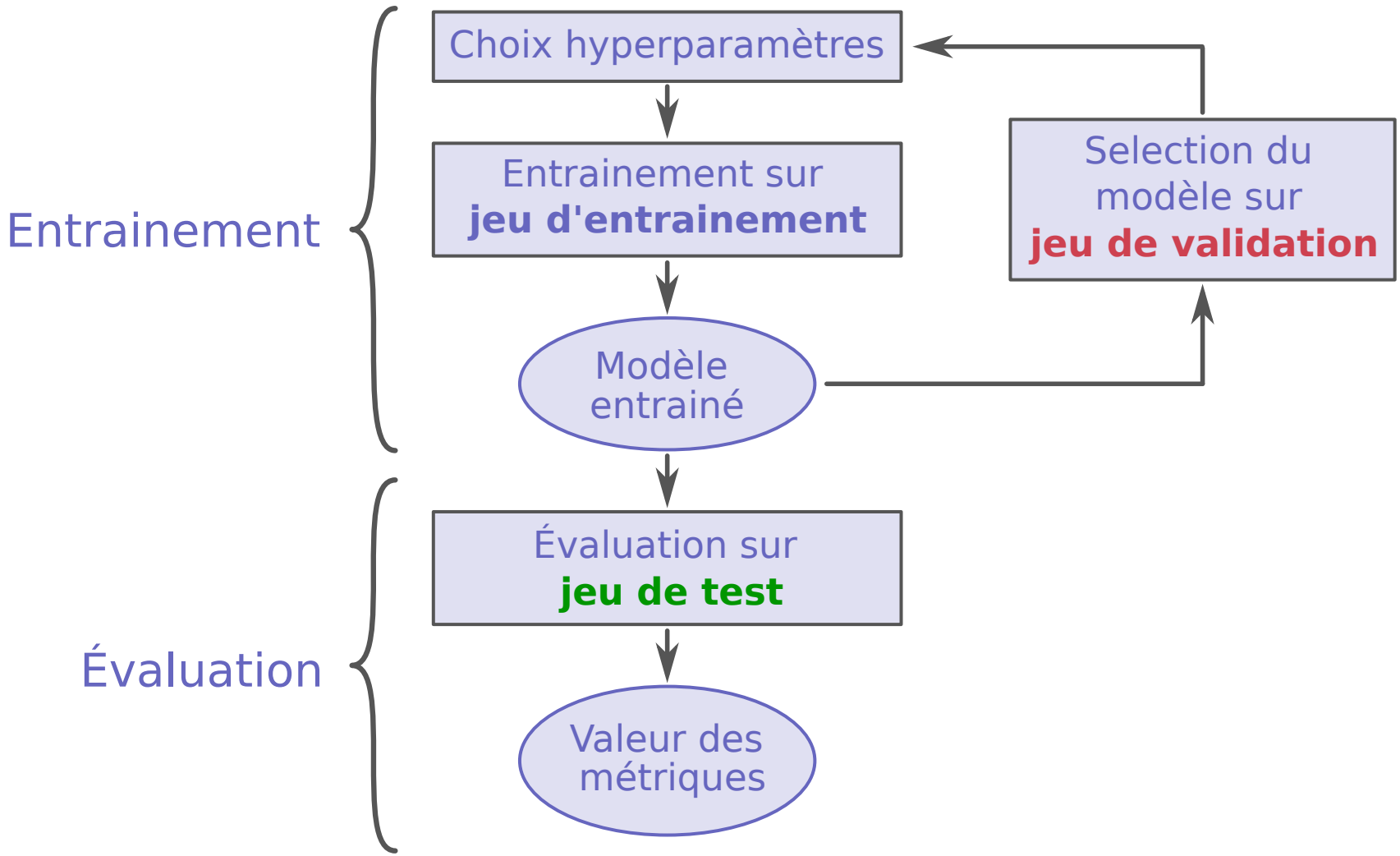
Taille de batch N_{batch}

- Influe principalement sur la quantité de mémoire utilisée et la stabilité de l'apprentissage.
  - Petite taille de batch : Moins de mémoire utilisée, apprentissage plus instable
  - Grande taille de batch : Plus de mémoire utilisée, apprentissage plus stable



- Peut également jouer le rôle de régularisation
  - Réduire la taille de batch peut permettre à l'apprentissage d'éviter des minima locaux

Conclusion


Entrainement
Évaluation
Choix hyperparamètres
Entrainement sur
jeu d'entrainement
Modèle
entrainé
Selection du
modèle sur
jeu de validation
Évaluation sur
jeu de test
Valeur des
métriques

Plan

Sujet du TP

Entraînement de réseaux de neurones pour de la classification d'images

- Sur deux applications
  - Reconnaissance de coupes d'images du cerveau humain
  - Reconnaissance de vêtements et accessoires de mode
- Avec deux types d'architectures
  - MLP
  - CNN

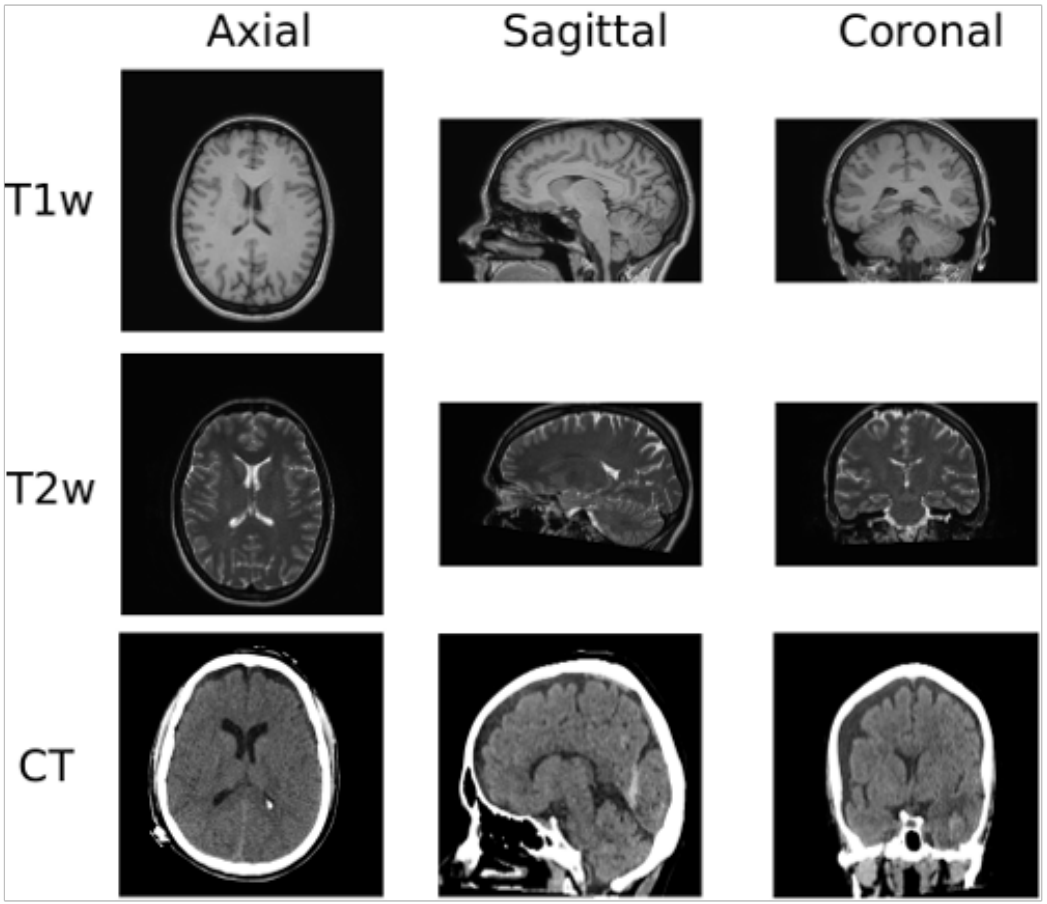
Objectifs :

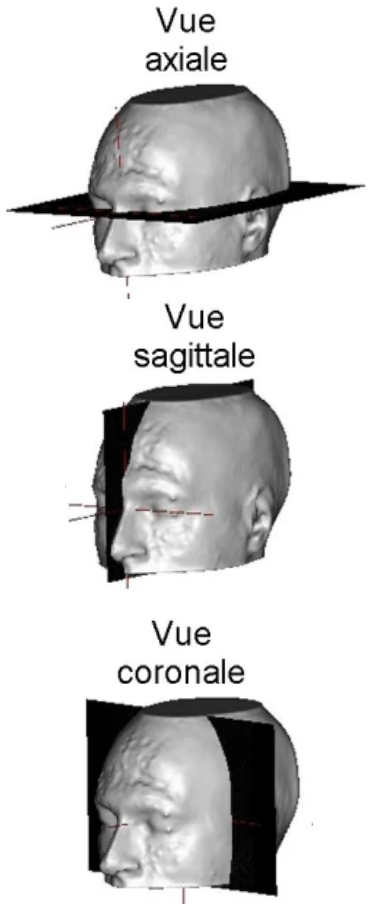
- Manipuler le pipeline complet d'entrainement / évaluation
- Comparer différentes architectures et applications
- Observer et analyser l'influence de plusieurs hyperparamètres

Application 1 : Coupes d'images du cerveau

- Base de données d'images du cerveau acquises selon 3 modalités (IRM T1, IRM T2, scanner) et dans 3 plans de coupe (axial, sagittal, coronal)

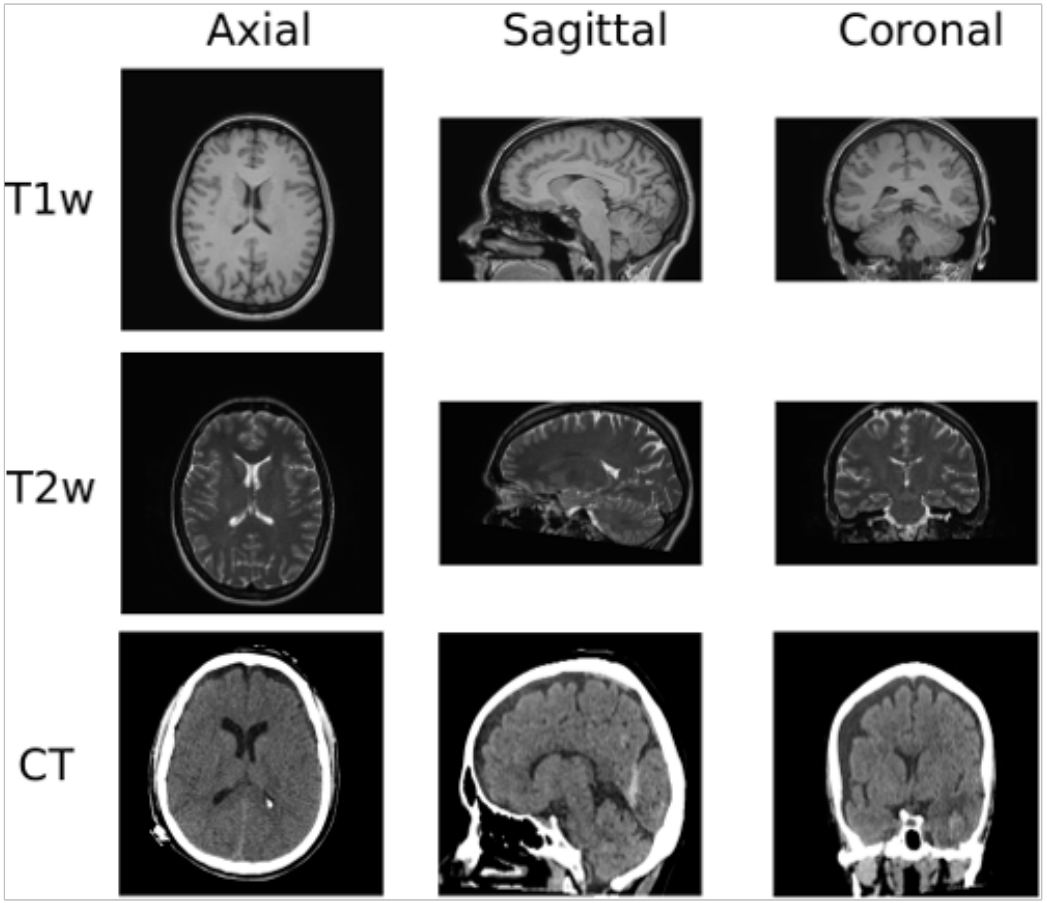
$\longrightarrow$ **9 classes différentes**

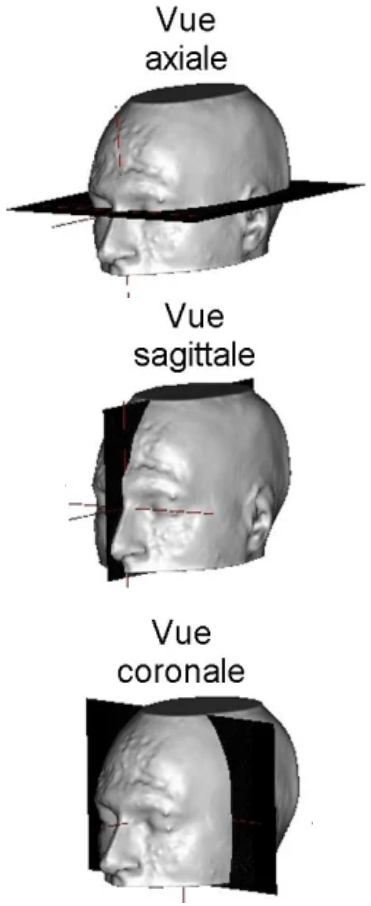




Application 1 : Coupes d'images du cerveau

- Objectif : Reconnaître dans quelle modalité et dans quel plan de coupe une image a été acquise.





Application 2 : Images de vêtements et accessoires de mode

- Base de données libre FashionMNIST contenant des images 2D de vêtements et accessoires de mode réparties en 10 classes :

Application 2 : Images de vêtements et accessoires de mode

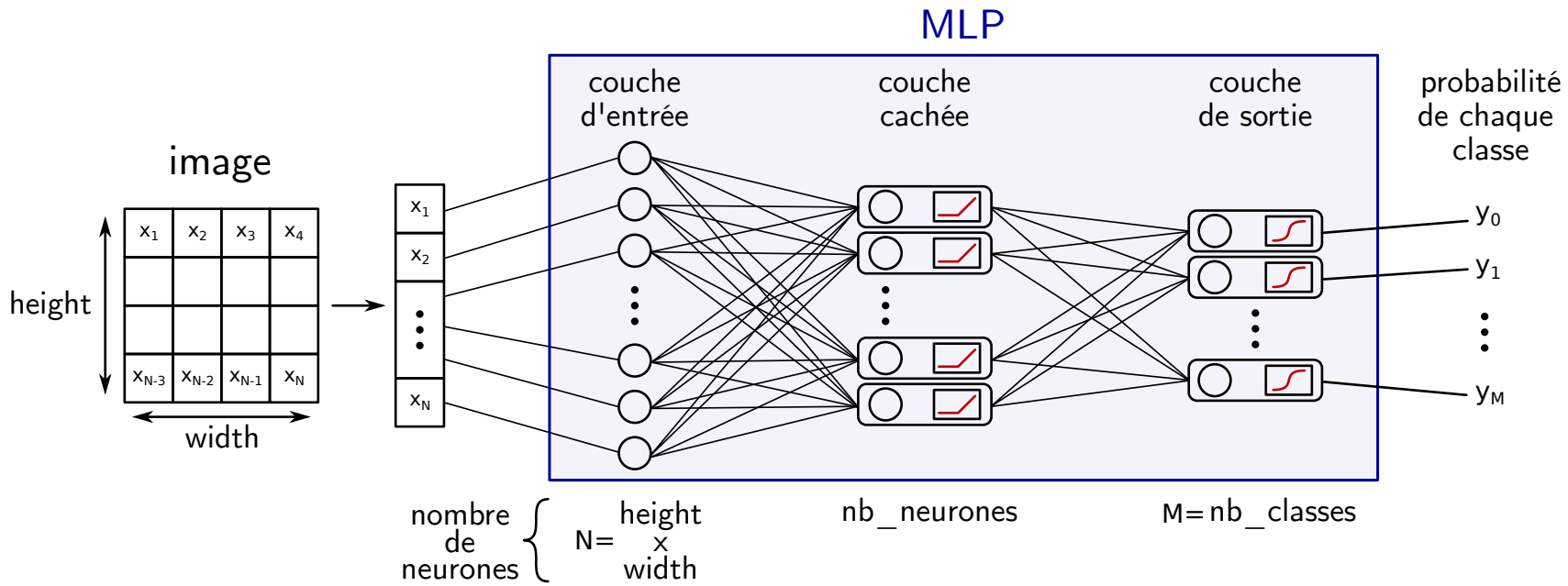
- Objectif : Reconnaître le type de vêtement ou accessoire à partir d'une image

Comparaison des bases de données

	Médicale	FashionMNIST
Nombre d'images	21 314	70 000
Taille des images	64×64 pixels	28×28 pixels
Nombre de classes	9	10

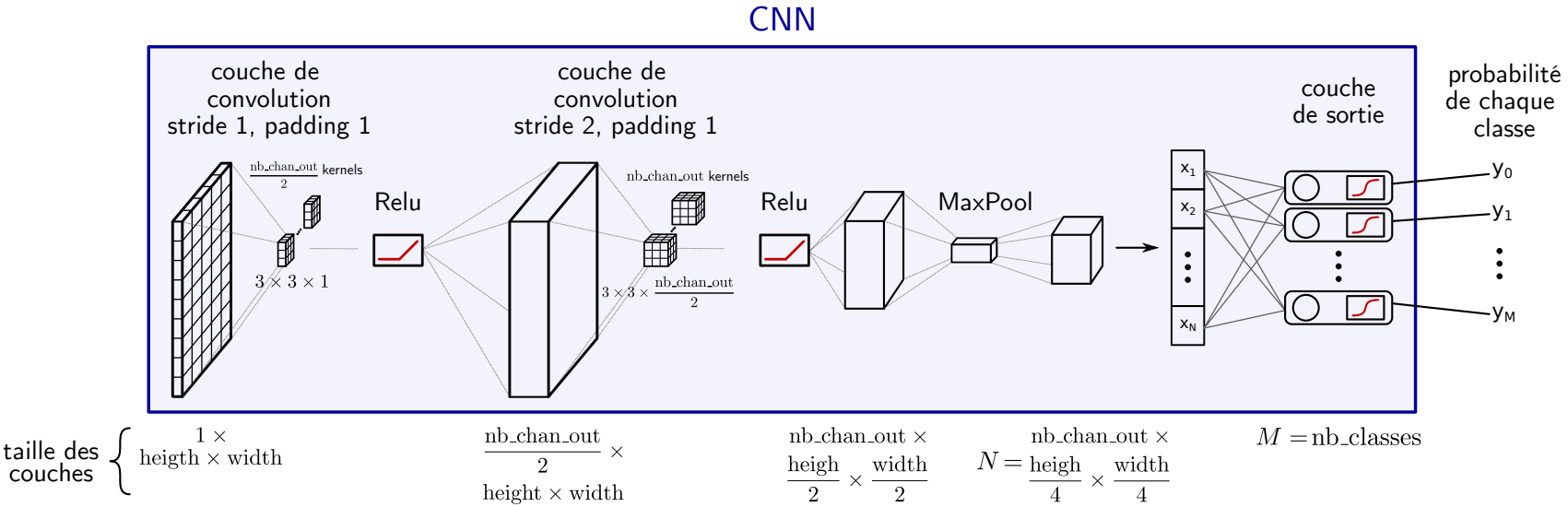
- Bases de données différentes, mais même type d'application
- Possibilité d'utiliser les mêmes architectures aux variations d'entrées et sorties près

Architecture MLP



- Couche cachée : Fully-connected + ReLU
- Couche de sortie : Fully connected + Softmax (logsoftmax)
- Loss : Negative log likelihood
- Sortie : y_i "Probabilité" que l'image d'entrée soit de la classe i
  En pratique $y_i \in]-\infty, 0]$. Le label prédit est celui de la classe associée à la plus grande valeur y_i.

Architecture CNN



- Couche 1 : Convolution + ReLU
- Couche 2 : Convolution + ReLU
- Couche 3 : Max pooling
- Couche de sortie : Fully connected + Softmax (logsoftmax)
- Loss : Negative log likelihood
- Sortie : y_i "Probabilité" que l'image d'entrée soit de la classe i