Méthodologie générale d'un projet d'Apprentissage Profond

Apprentissage Profond

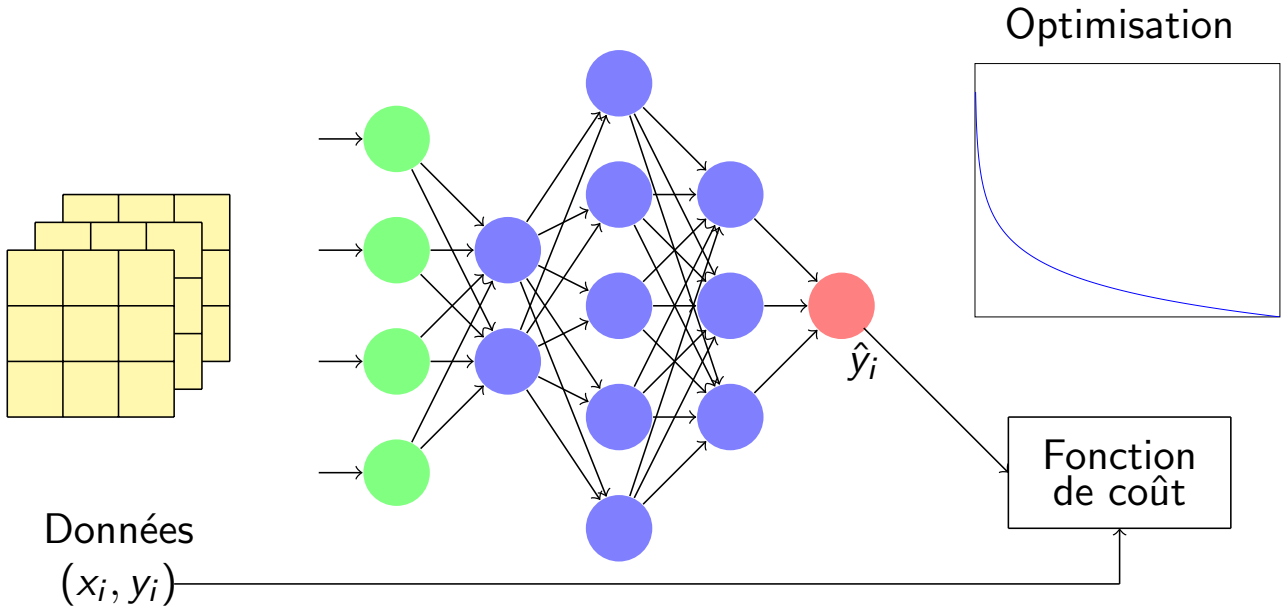
A. Carlier

2026

Difficulté de l'apprentissage profond

Vous connaissez maintenant tous les éléments constitutifs d'un algorithme basé sur l'apprentissage profond.

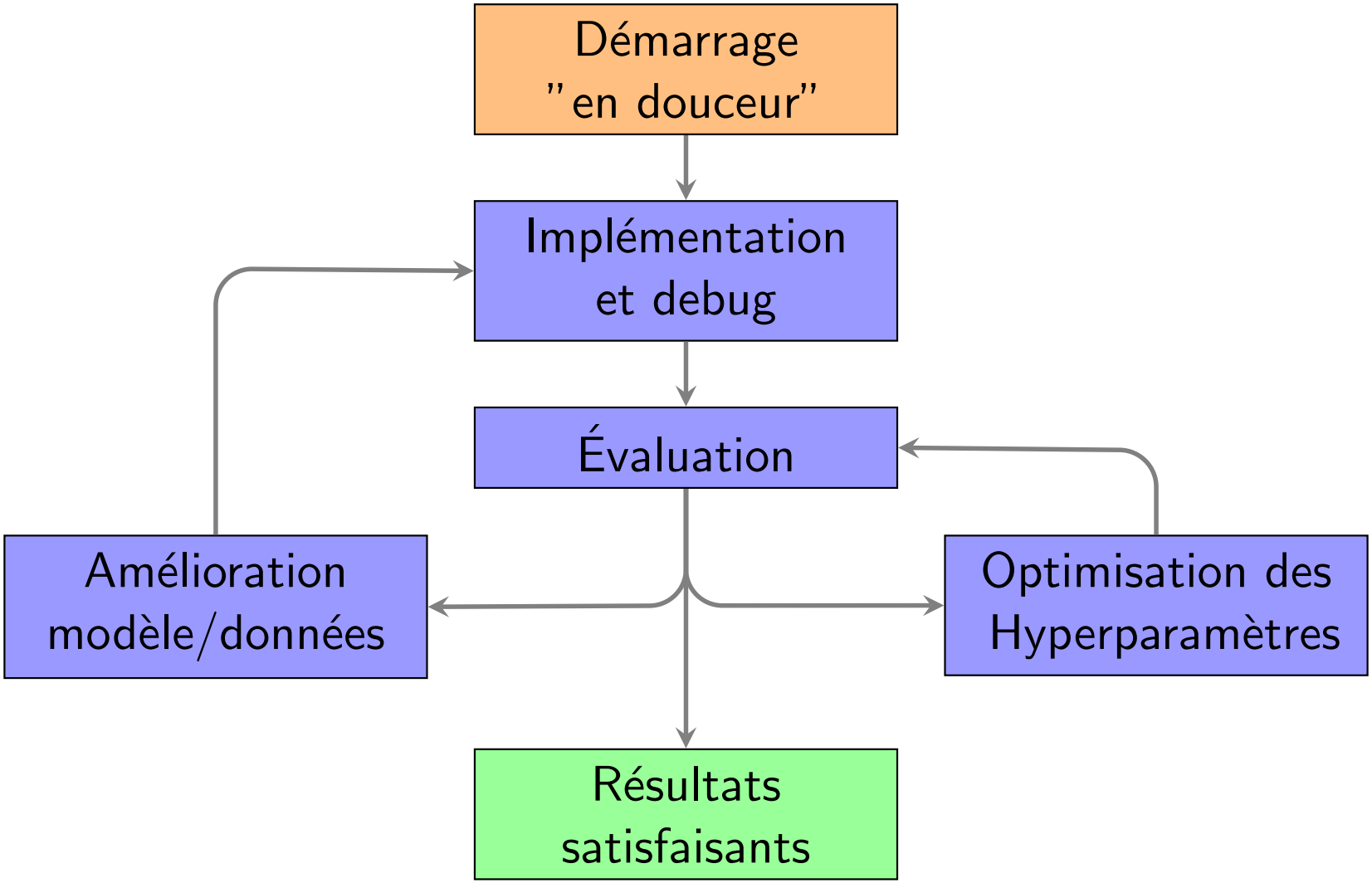
Comment s'y retrouver parmi toutes les combinaisons possibles d'architectures, d'hyperparamètres, de type de régularisation, etc. ? Tout n'est pas toujours nécessaire, ni désirable, ni aussi efficace en fonction des problèmes, des données, et du contexte.



profondeur, #neurones, activation α, momentum

$w_{i,j}, b_k$

Méthodologie générale



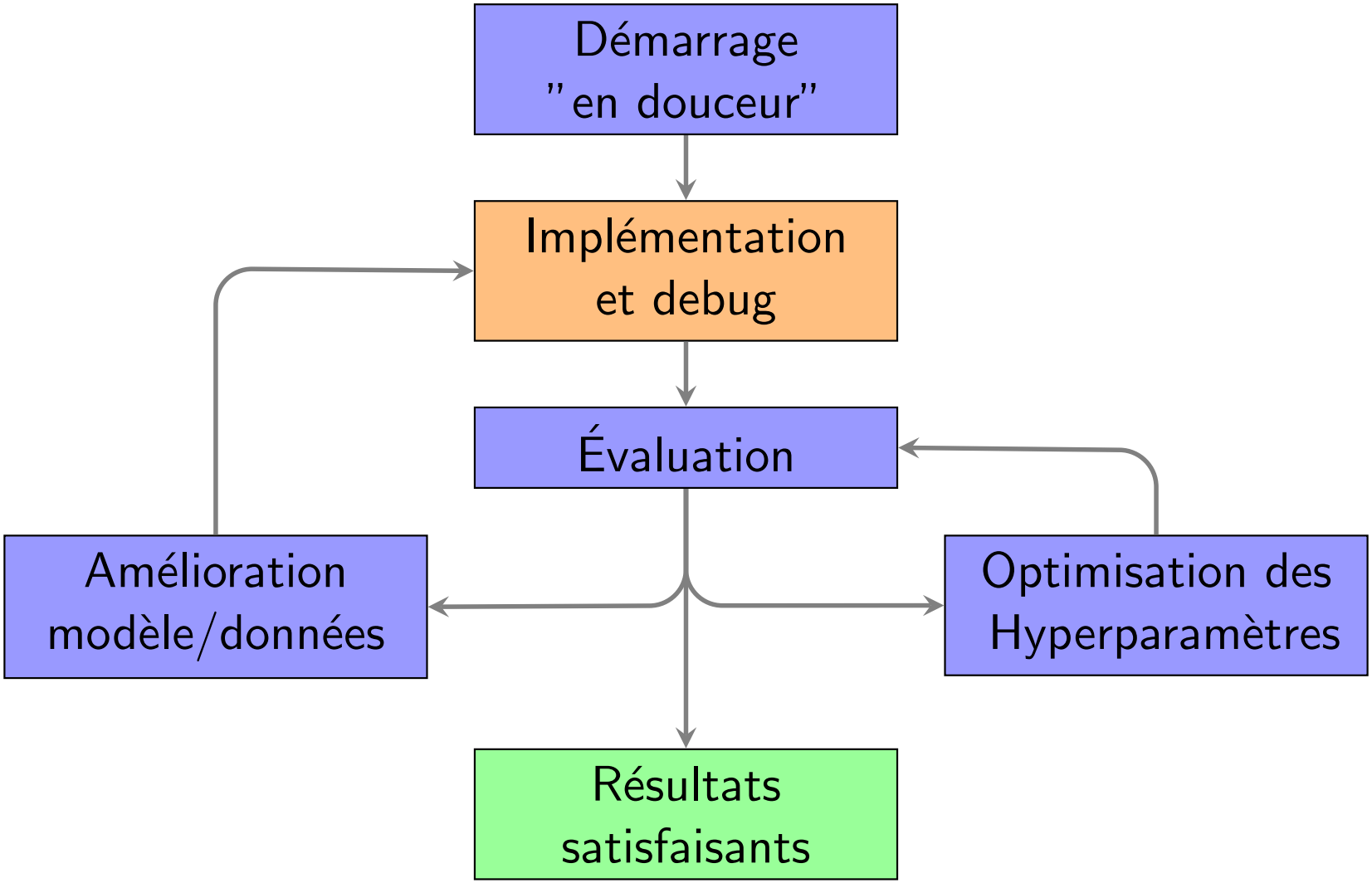
Adapté de *Troubleshooting Deep Neural Networks* de Josh Tobin

Démarrer simplement

Pour commencer sur un nouveau problème, l'objectif est de mettre au point une première version simpliste, mais qui fonctionne, à partir de laquelle on va itérer. Pour cela :

- Choisir une **architecture simple**
  Perceptron multi-couche, LeNet, LSTM
- Utiliser des **paramètres par défauts** efficaces
  Adam (avec taux d'apprentissage magique !), activation `reLU`, pas de régularisation
- **Normalisation** des données
  centrer-réduire, ramener entre 0 et 1
- Éventuellement **simplifier** le problème
  ensemble d'apprentissage réduit, diminuer le nombre de classes, la dimension des images

Méthodologie générale



Adapté de *Troubleshooting Deep Neural Networks* de Josh Tobin

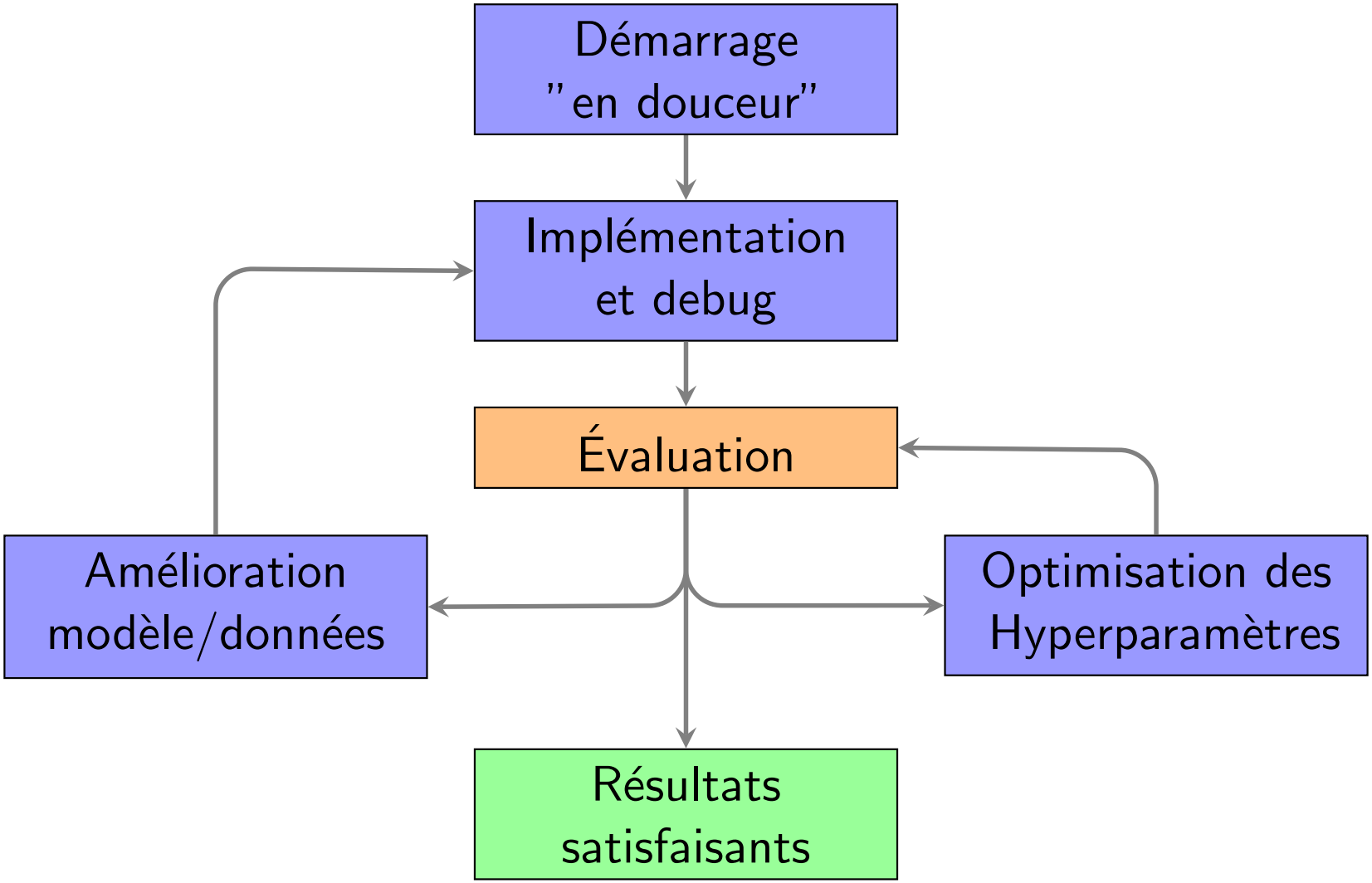
Débugger

Quelques erreurs et problèmes courants :

- Taille ou type des tenseurs incorrects
- Pré-traitement des données oublié ou fait plusieurs fois
- Sortie non adaptée à la fonction de perte
- Instabilité numérique (`NaN`, `Inf`) liée aux exponentielles, logarithmes, divisions, etc.
- Déficit de RAM GPU

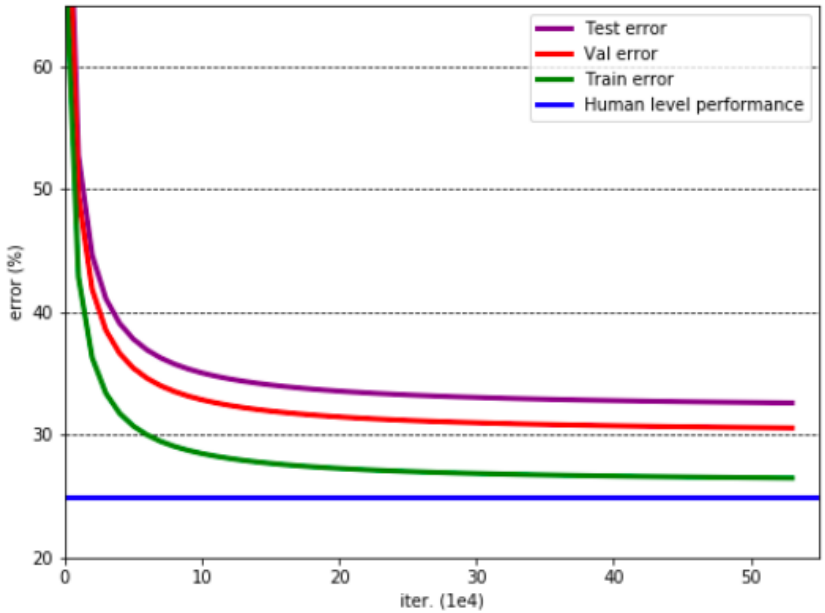
Conseil important : commencer par essayer de surapprendre un *batch* de données (voire une seule donnée)

Méthodologie générale



Adapté de *Troubleshooting Deep Neural Networks* de Josh Tobin

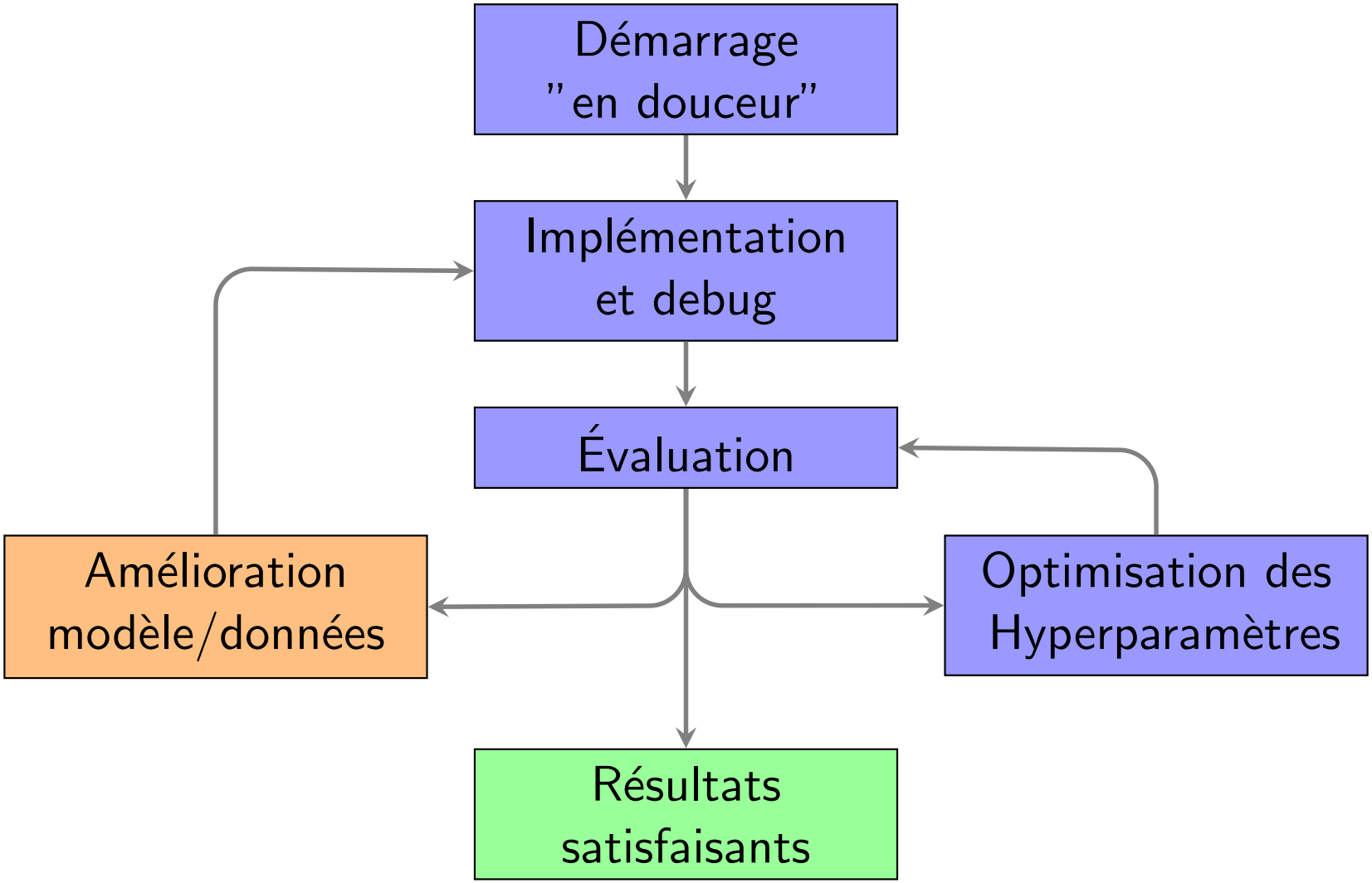
Évaluer la source des erreurs



L'erreur sur l'ensemble de test peut provenir de différentes sources :

- **Erreur irréductible** : meilleure performance objectivement atteignable
- **Sous-apprentissage**
- **Sur-apprentissage**
- **Sur-apprentissage de l'ensemble de validation**

Méthodologie générale



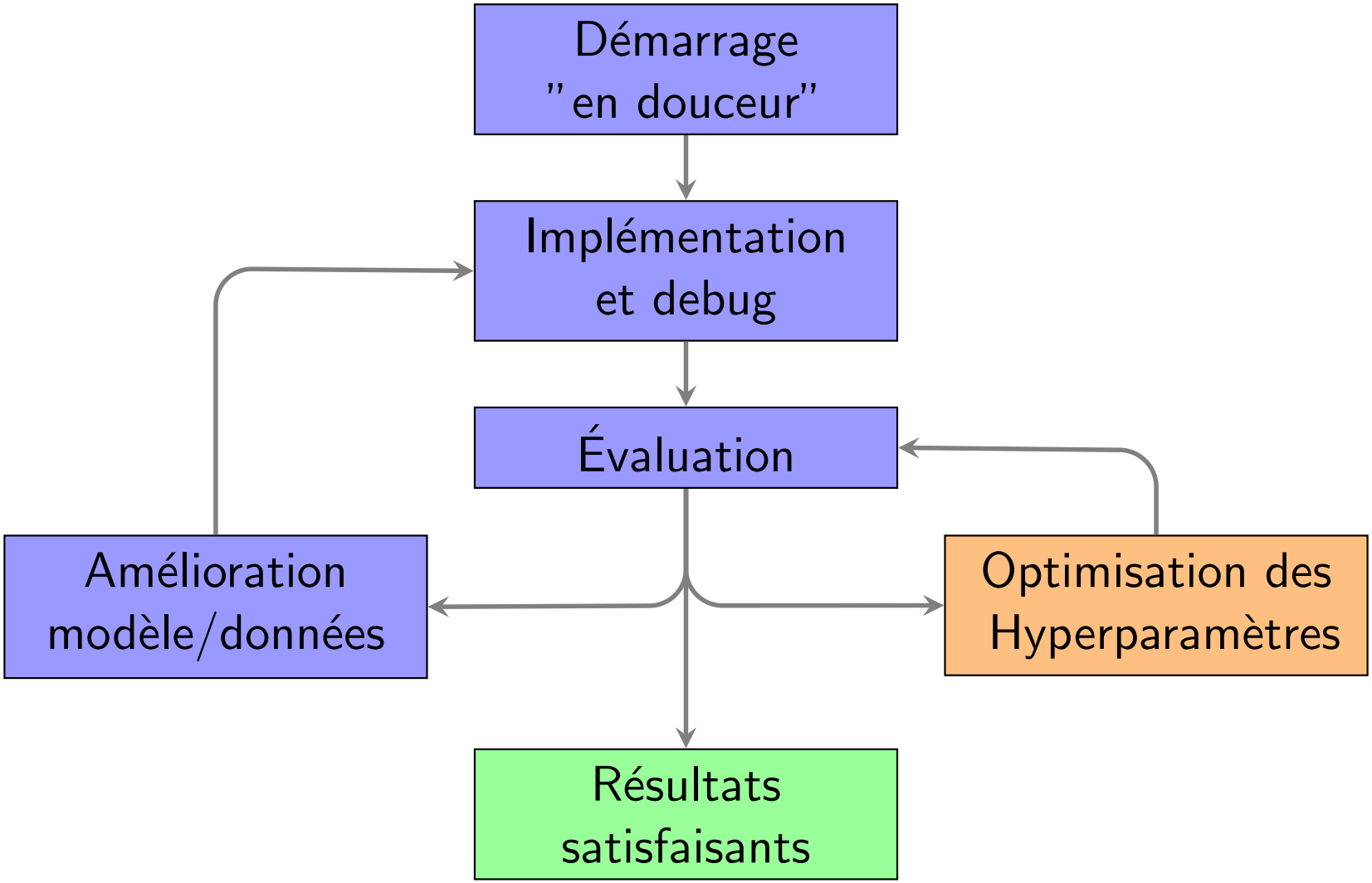
Adapté de *Troubleshooting Deep Neural Networks* de Josh Tobin

Améliorer le modèle et/ou les données

Pour corriger les problèmes mis à jour lors de l'évaluation on va privilégier l'ordre suivant :

1. Correction du sous-apprentissage
   Augmentation de la taille du modèle, changement d'architecture
   Réduction de la régularisation
2. Correction du sur-apprentissage
   Ajout de données (quand c'est possible !), ou augmentation des données
   Régularisation
3. Correction des problèmes de bases de donnée
   Déséquilibre de classes, distributions train/val/test différentes

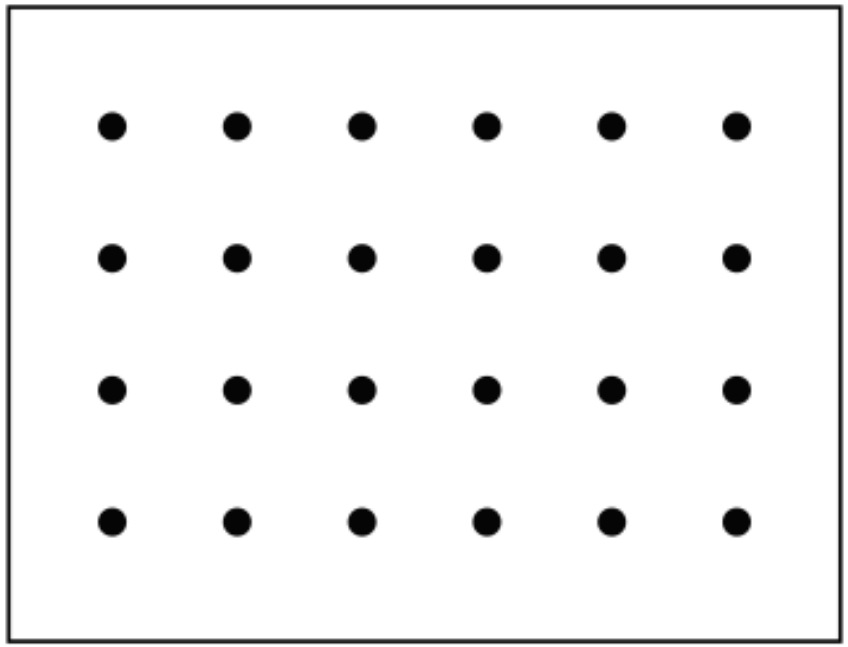
Méthodologie générale



Adapté de *Troubleshooting Deep Neural Networks* de Josh Tobin

Hyperparamètres

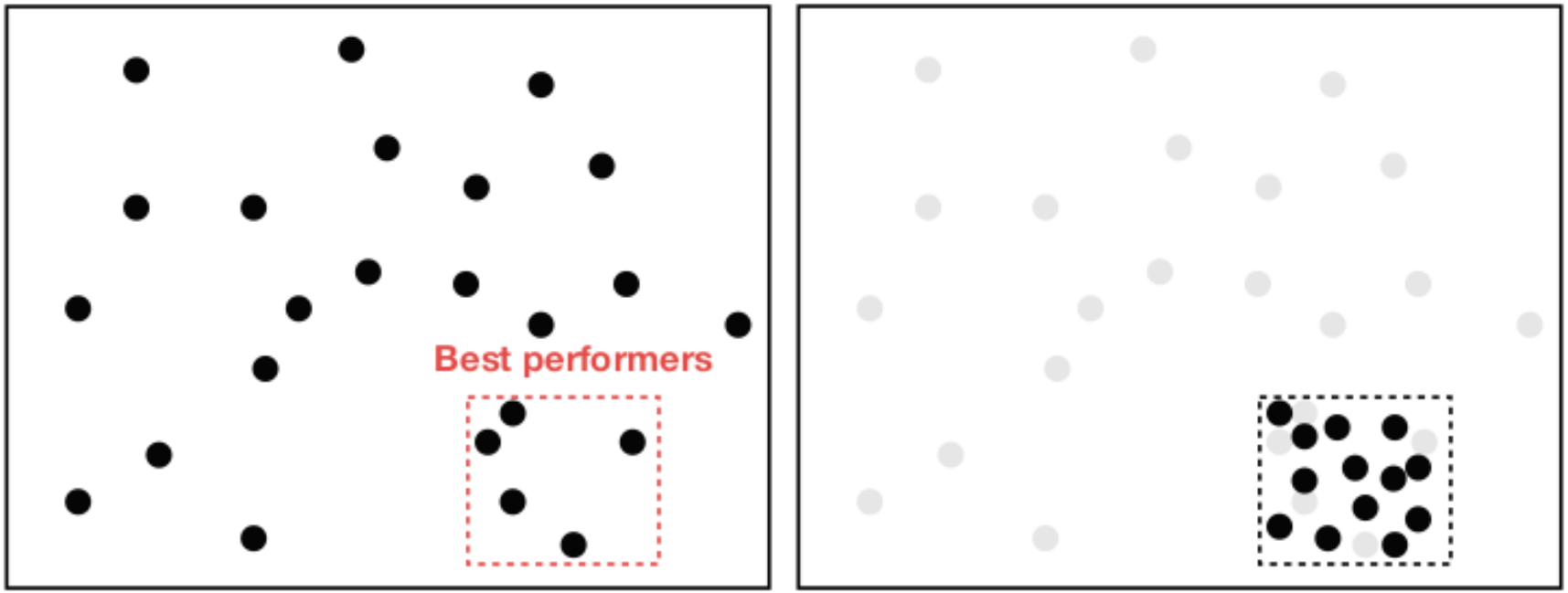
Pour mettre au point les valeurs optimales d'hyperparamètres (taux d'apprentissage, momentum, nombre de neurones par couche, etc.), une pratique commune est de déterminer un ensemble de valeurs possibles pour chaque hyperparamètre et de tester toutes les combinaisons. On conserve la combinaison qui minimise l'erreur sur l'ensemble de validation.



Chaque point correspond à l'entraînement complet d'un réseau de neurones (peut prendre plusieurs heures/jours/semaines...)

Hyperparamètres

D'autres variantes plus courantes consistent à implémenter une recherche aléatoire *coarse-to-fine* ou encore une optimisation bayésienne de la recherche d'hyperparamètres (e.g. Optuna)



Chaque point correspond à l'entraînement complet d'un réseau de neurones (peut prendre plusieurs heures/jours/semaines...)