

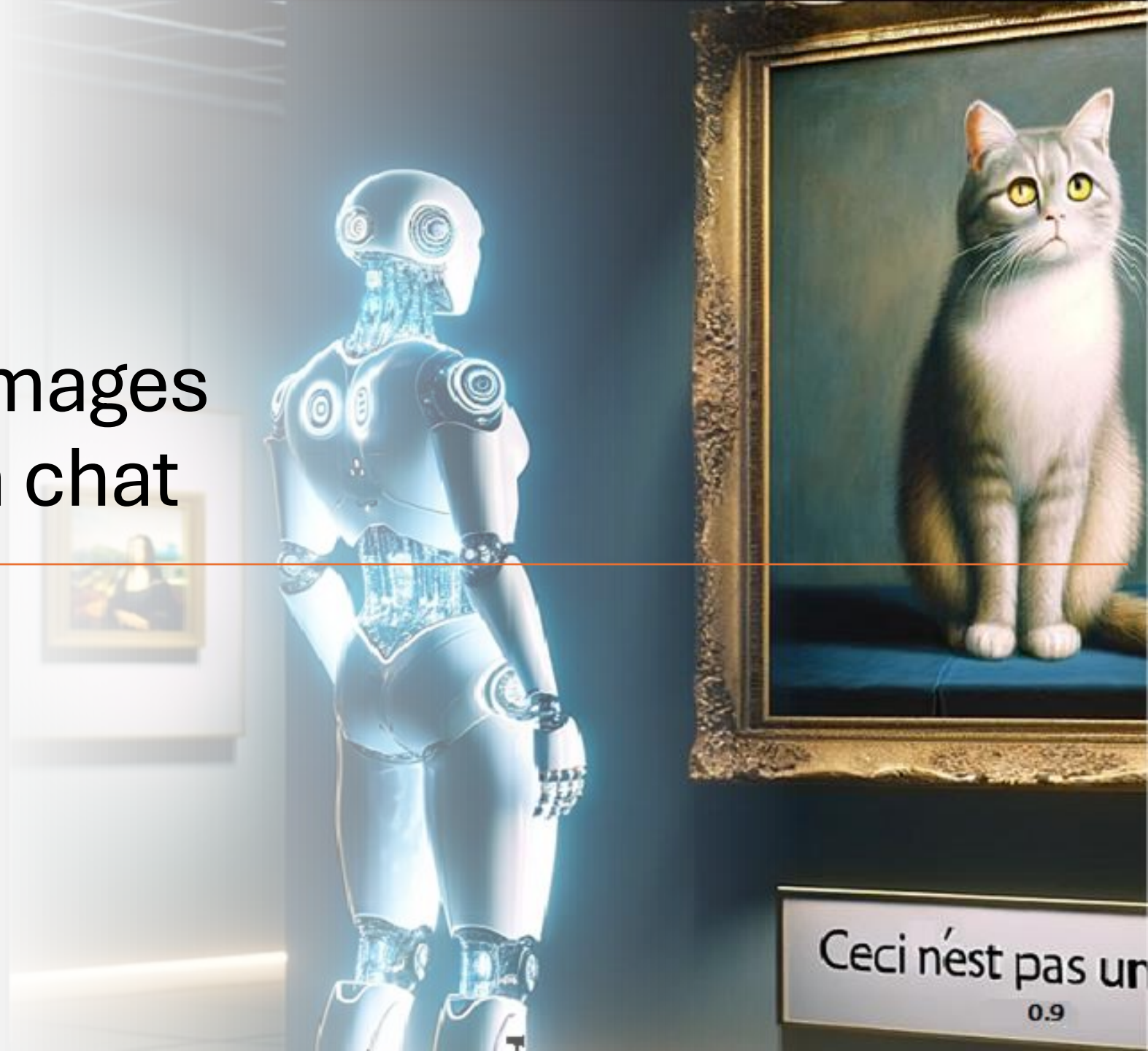
La trahison des images Ceci n'est pas un chat

Meetup Python Grenoble

La Turbine.coop

Jeudi 23 mai 2024 – 19h

Pierre-Loïc Bayart



Présentation en quelques emojis



Pierre-Loïc
Bayart



2010

2012

2016

2017

2023



EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES

Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy
Google Inc., Mountain View, CA
{goodfellow, shlens, szegedy}@google.com

ABSTRACT


Several machine learning models, including neural networks, consistently misclassify *adversarial examples*—inputs formed by applying small but intentionally worst-case perturbations to examples from the dataset, such that the perturbed input results in the model outputting an incorrect answer with high confidence. Early attempts at explaining this phenomenon focused on nonlinearity and overfitting. We argue instead that the primary cause of neural networks' vulnerability to adversarial perturbation is their linear nature. This explanation is supported by new quantitative results while giving the first explanation of the most intriguing fact about them: their generalization across architectures and training sets. Moreover, this view yields a simple and fast method of generating adversarial examples. Using this approach to provide examples for adversarial training, we reduce the test set error of a maxout network on the MNIST dataset.

Adversarial example using FGSM

On this page

- What is an adversarial example?
- Fast gradient sign method
- Original image
- Create the adversarial image
 - Implementing fast gradient sign method
- Next steps

 Run in Google Colab

 View source on GitHub

 Download notebook

This tutorial creates an *adversarial example* using the Fast Gradient Signed Method (FGSM) attack as described in [Explaining and Harnessing Adversarial Examples](#) by Goodfellow *et al.* This was one of the first and most popular attacks to fool a neural network.

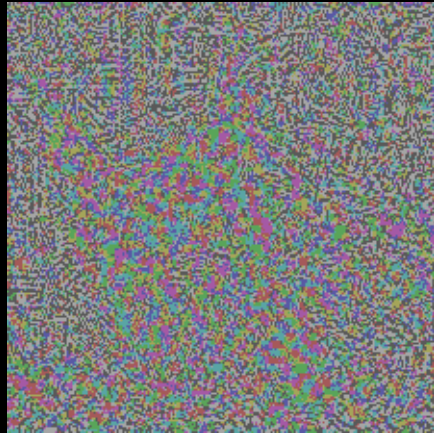
Sources d'inspiration

Exemple d'image adverse

Source : Pixabay



+



=



Prédictions modèle MobileNetV2 :

1. Chat égyptien : 0.64
2. Chat domestique : 0.17
3. Chat tigré : 0.13

Bruit construit pour tromper le modèle

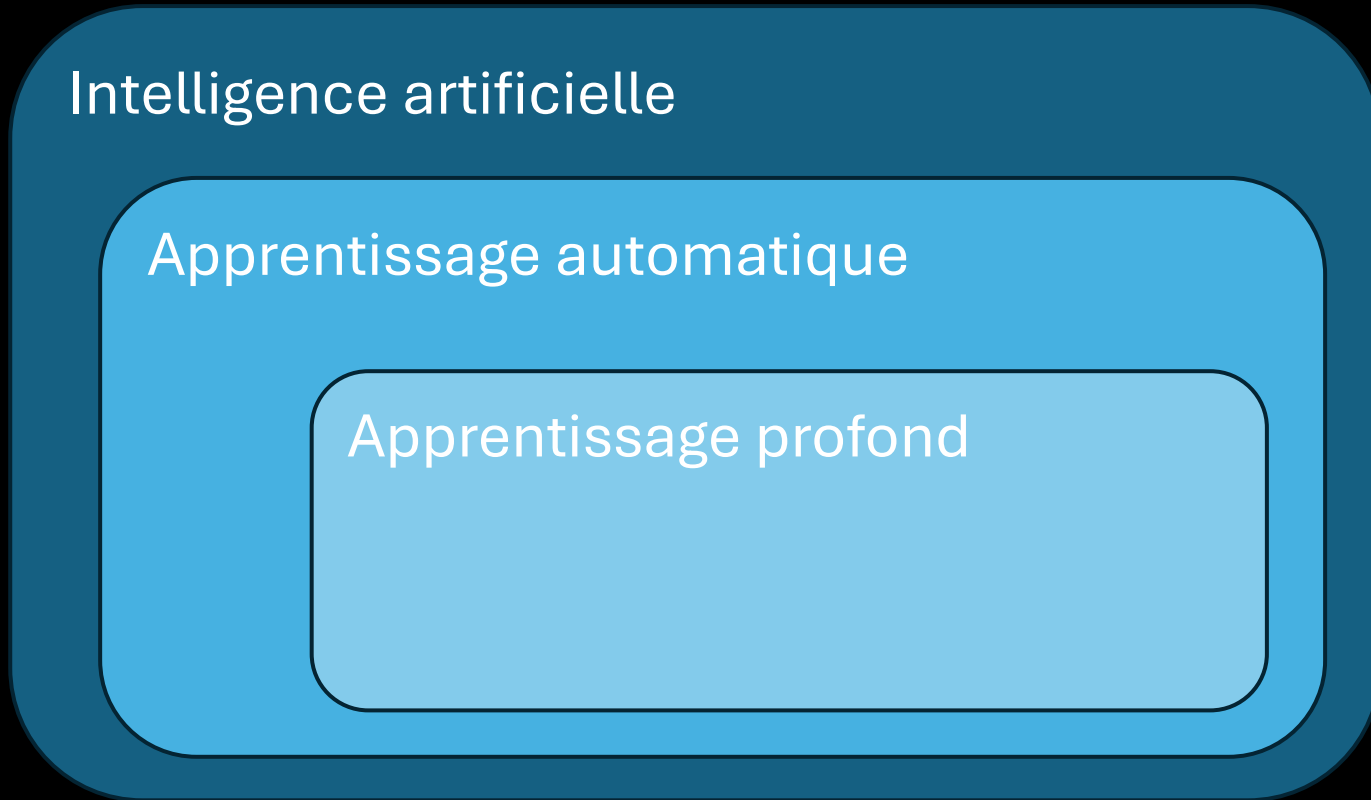
Prédictions modèle MobileNetV2 :

1. Récif corallien : 0.58
2. Corail cerveau : 0.17
3. Paon : 0.05

Plan de la présentation

- Intelligence artificielle
- Apprentissage automatique
- Apprentissage supervisé
 - ✓ Régression et descente de gradient
- Classification
- Réseaux de neurones
 - ✓ Images et filtres de convolution
- Classification d'images avec réseaux de convolution
 - ✓ Rétro-propagation du gradient
- Images adverses

Définition de l'intelligence artificielle



Actuellement :
Intelligence artificielle
=
Apprentissage
automatique
=
Apprentissage à partir
des données

L'intelligence artificielle en Python (la sagesse)

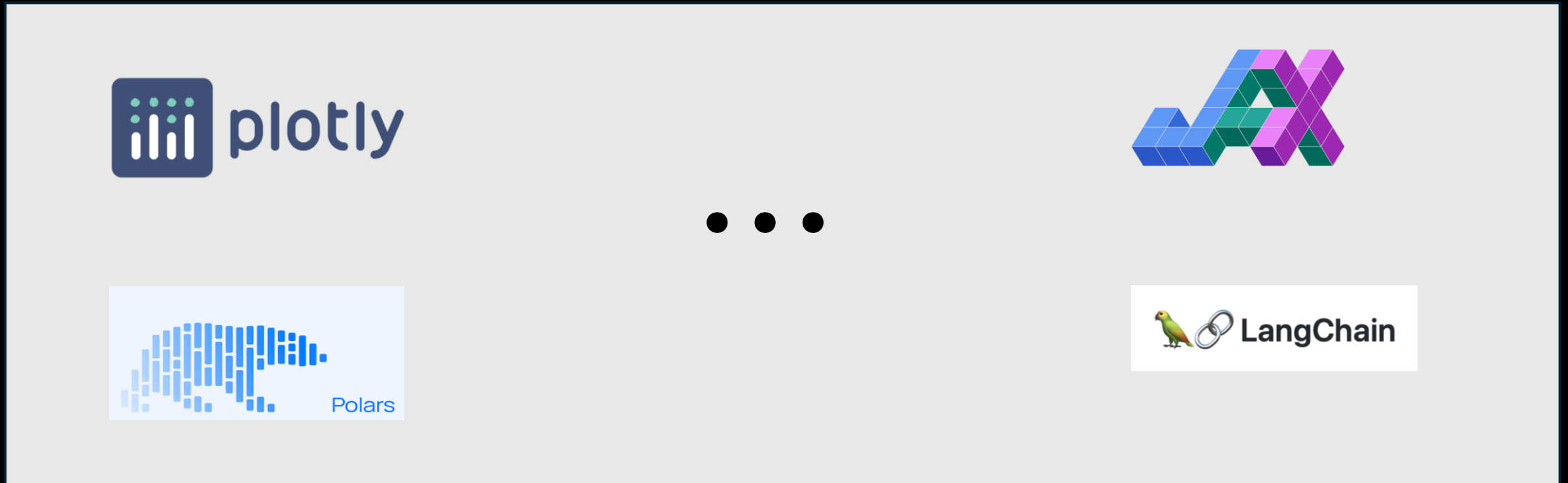


Les données



L'intelligence artificielle

L'intelligence artificielle en Python (la jeunesse)



Les données



L'intelligence artificielle

Définition de l'intelligence artificielle

Intelligence artificielle

Apprentissage automatique

Apprentissage profond

Apprentissage à partir
des données




Quelles
données ?

Intelligence artificielle avec quelles données ?

- Données tabulaires / structurées
- Données non structurées :
 - Texte
 - Images
 - Audio
 - Vidéos
 - ...

Exemples de sous-domaines de l'IA

Suivant le type de données d'entraînement :

-  Apprentissage automatique classique : modèles statistiques, modèles ensemblistes...
-  Traitement automatique des langues : modèles de transformeurs (GPT, BERT...)
-  Vision par ordinateur : réseaux de neurones de convolutions

Vision par ordinateur

- **Classification d'images**
- Détection d'objets
- Segmentation sémantique
- Segmentation d'instances



Chat



Hérisson



Vélo

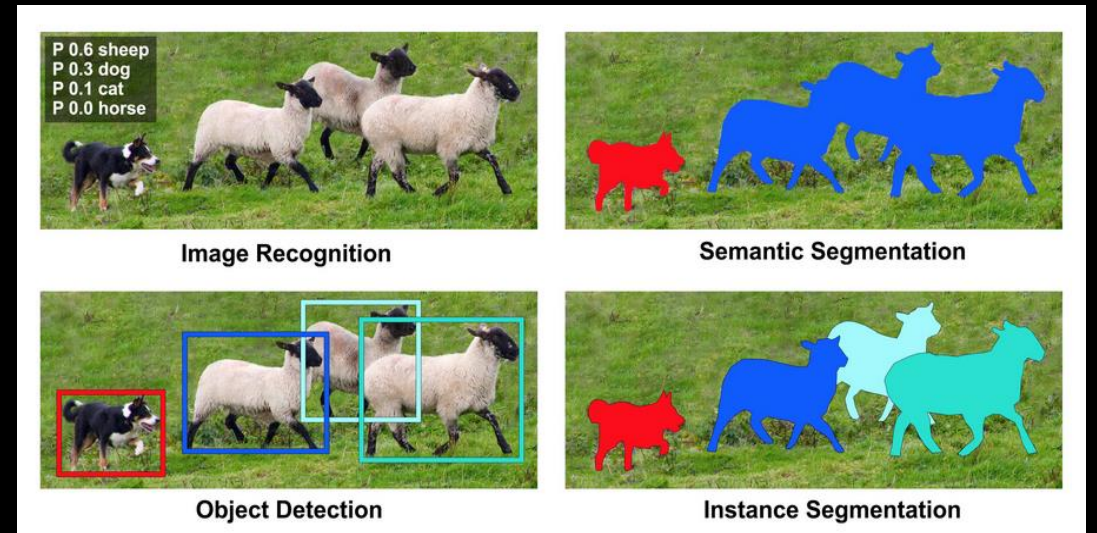
Vision par ordinateur

- Classification d'images
- **Détection d'objets**
- Segmentation sémantique
- Segmentation d'instances



Vision par ordinateur

- Classification d'images
- Détection d'objets
- **Segmentation sémantique**
- **Segmentation d'instances**



Définition de l'intelligence artificielle

Intelligence artificielle

Apprentissage automatique

Apprentissage supervisé

Régression

Classification

Apprentissage
non supervisé

Apprentissage
par
renforcement

Apprentissage profond

Partie un peu plus technique...

Apprentissage automatique

"classique" supervisé

Définition de l'intelligence artificielle

Intelligence artificielle

Apprentissage automatique

Apprentissage supervisé

Régression

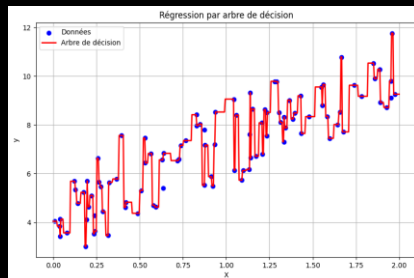
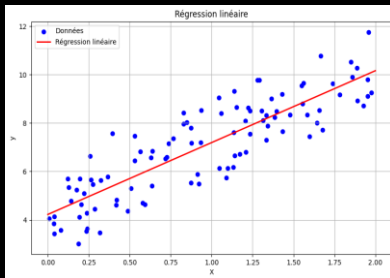
Classification

Apprentissage
non supervisé

Apprentissage
par
renforcement

Apprentissage automatique supervisé

Régression

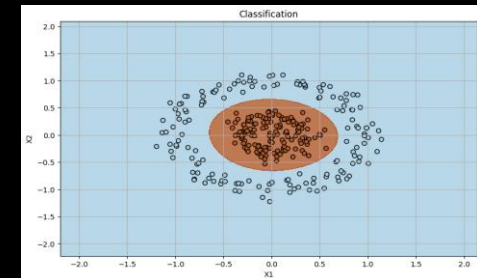


Exemples : prédire le prix d'une maison, la température, la durée de vie d'un produit...

Mathématiquement : $f : \mathbb{R}^n \rightarrow \mathbb{R}$

Valeurs cibles continues

Classification



Exemples : Diagnostiquer une maladie (oui/non), reconnaître des chiffres manuscrits, classier des emails en spam ou non-spam

Mathématiquement : $f : \mathbb{R}^n \rightarrow \{1, 2, \dots, K\}$

Valeurs cibles discrètes

Apprentissage automatique supervisé

Phase d'entrainement

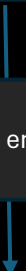
Jeu de données
d'entrainement



Modèle entrainé

Phase de test

Jeu de données
de test



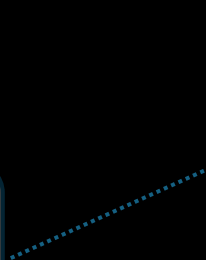
Modèle testé

Phase d'inférence

Jeu de données
à prédire

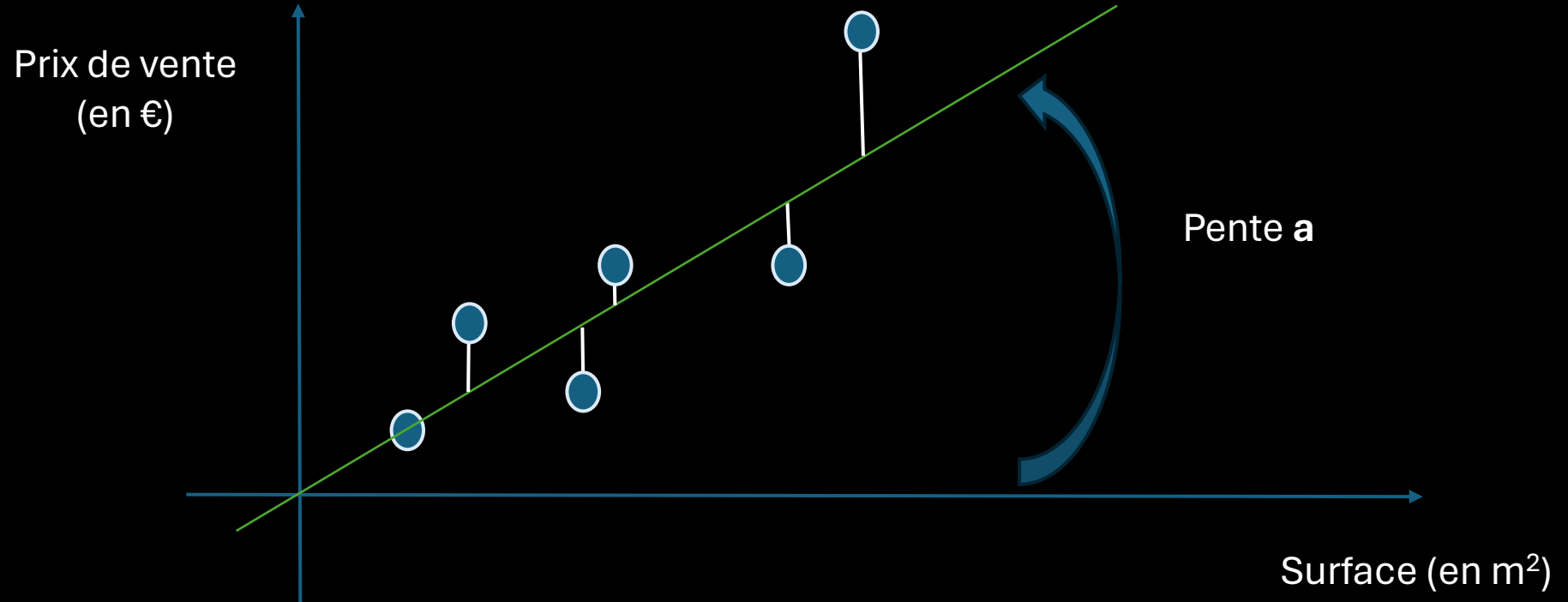


Prédictions



Apprentissage : cas le plus simple

Objectif : trouver le paramètre a de la pente qui minimise la fonction de perte



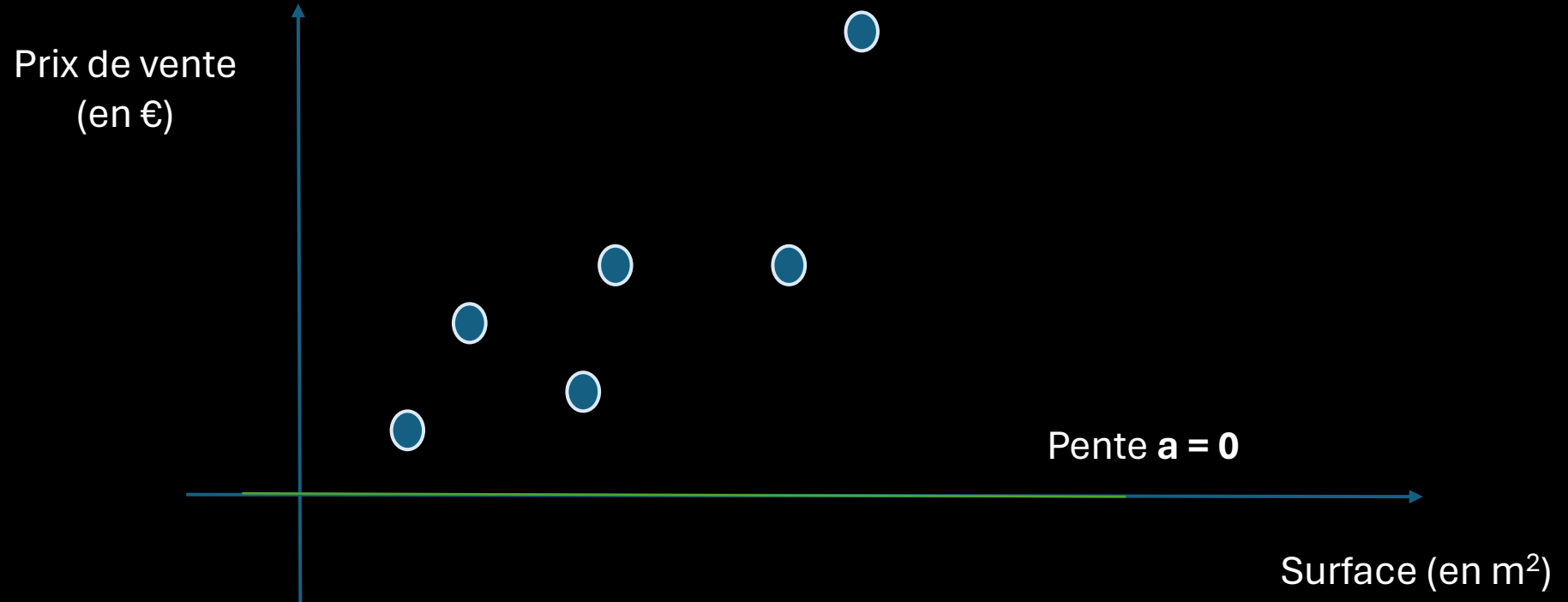
Apprentissage : cas le plus simple

Processus itératif de descente de gradient :

1. Initialisation du paramètre de la pente à 0 (notée a)

Apprentissage : cas le plus simple

1. Initialisation du paramètre de la pente à 0 (notée a)



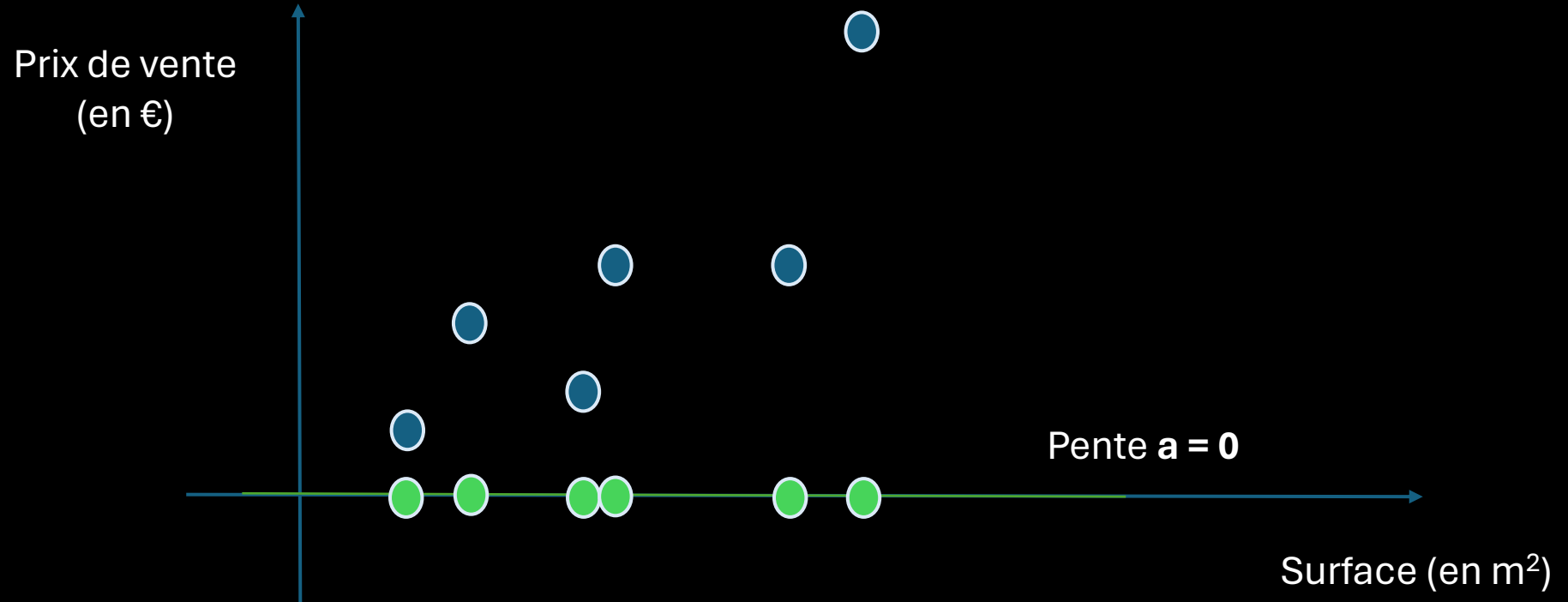
Apprentissage : cas le plus simple

Processus itératif de descente de gradient :

1. Initialisation du paramètre de la pente à 0 (notée a)
2. Calcul de la prédiction du prix de vente

Apprentissage : cas le plus simple

2. Calcul de la prédiction du prix de vente



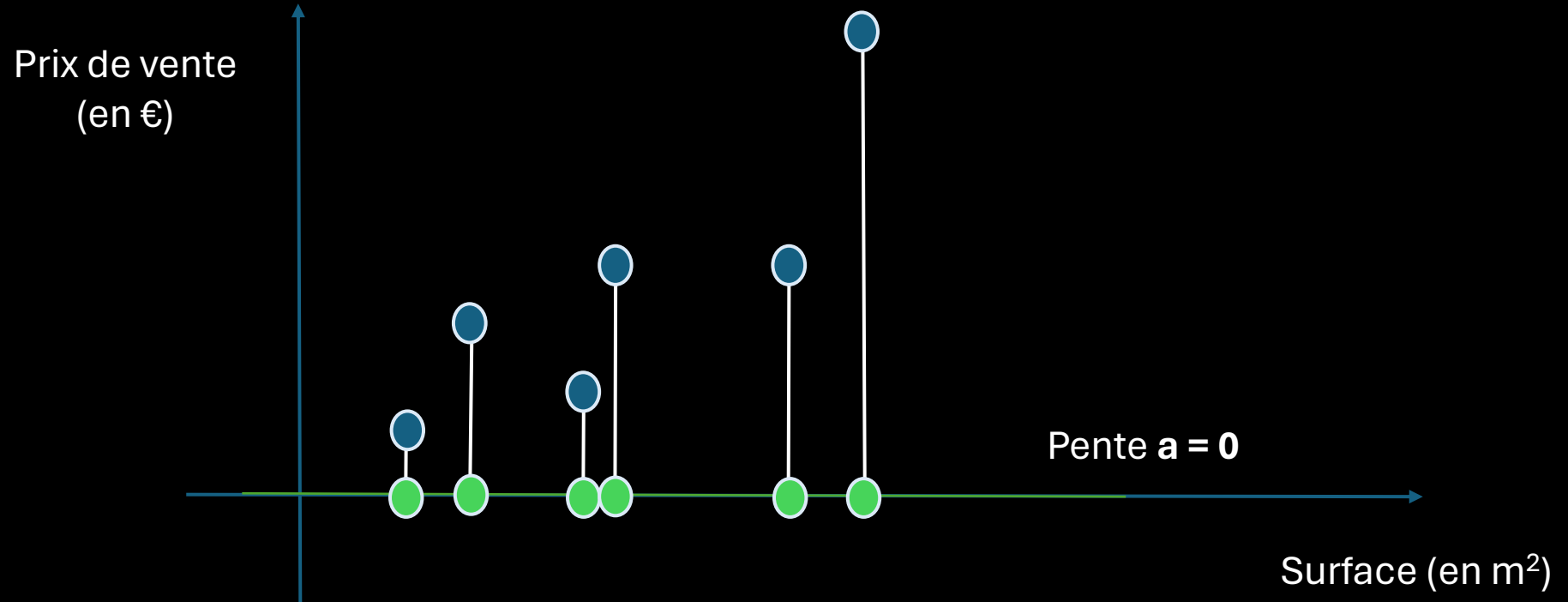
Apprentissage : cas le plus simple

Processus itératif de descente de gradient :

1. Initialisation du paramètre de la pente à 0 (notée a)
2. Calcul de la prédiction du prix de vente
3. Calcul de la fonction de perte qui mesure la différence les prédictions du modèle et les valeurs réelles

Apprentissage : cas le plus simple

3. Calcul de la fonction de perte



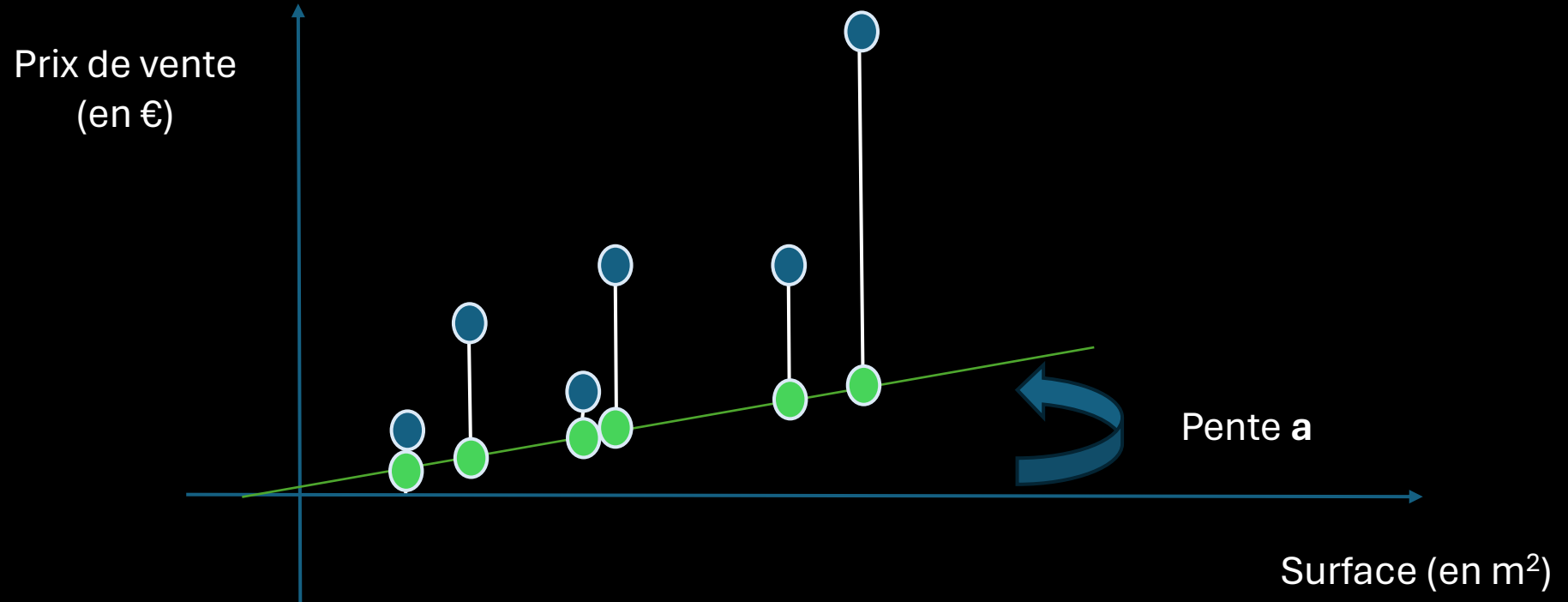
Apprentissage : cas le plus simple

Processus itératif de descente de gradient :

1. Initialisation du paramètre de la pente à 0 (notée a)
2. Calcul de la prédiction du prix de vente
3. Calcul de la fonction de perte qui mesure la différence les prédictions du modèle et les valeurs réelles
4. Calcul du gradient qui indique la pente de la fonction de perte
5. Mise à jour de la pente a dans la direction inverse au gradient proportionnellement au taux d'apprentissage

Apprentissage : cas le plus simple

- Mise à jour de la pente a dans la direction inverse au gradient proportionnellement au taux d'apprentissage



Apprentissage : cas le plus simple

Processus itératif de descente de gradient :

1. Initialisation du paramètre de la pente à 0 (notée a)
2. Calcul de la prédiction du prix de vente
3. Calcul de la fonction de perte qui mesure la différence les prédictions du modèle et les valeurs réelles
4. Calcul du gradient qui indique la pente de la fonction de perte
5. Mise à jour de la pente a dans la direction inverse au gradient proportionnellement au taux d'apprentissage

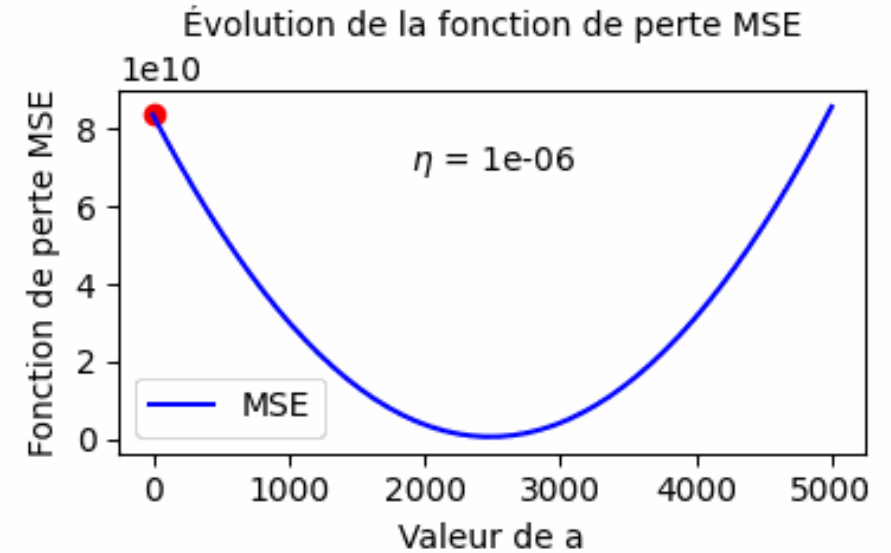
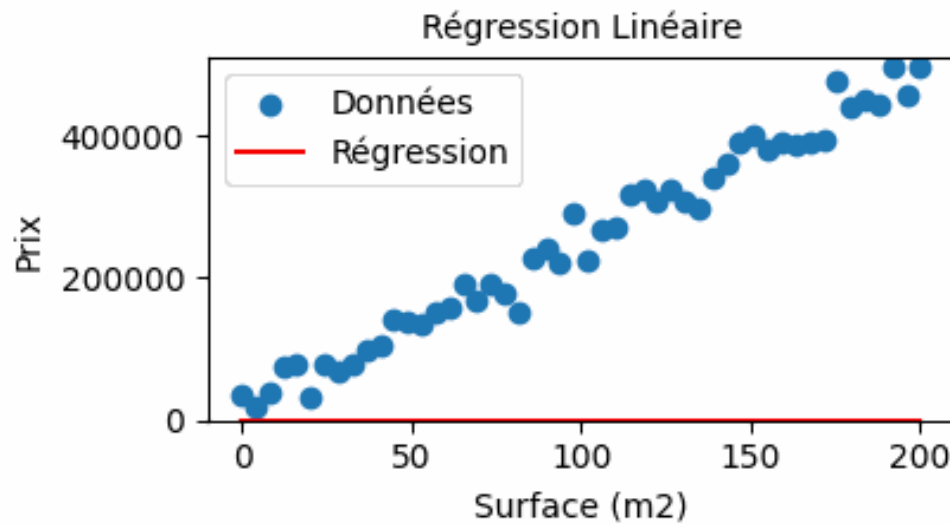


Processus
itératif

Apprentissage : cas le plus simple

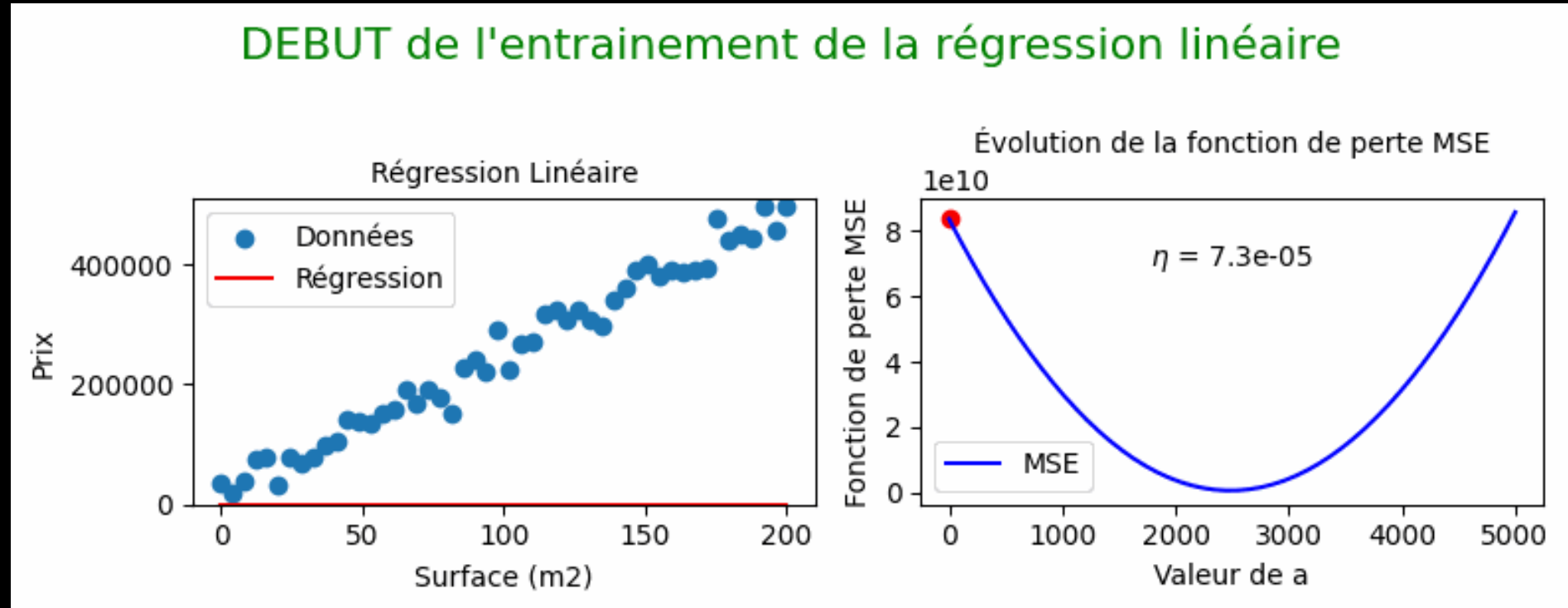
Avec un taux
d'apprentissage
trop faible

DEBUT de l'entrainement de la régression linéaire



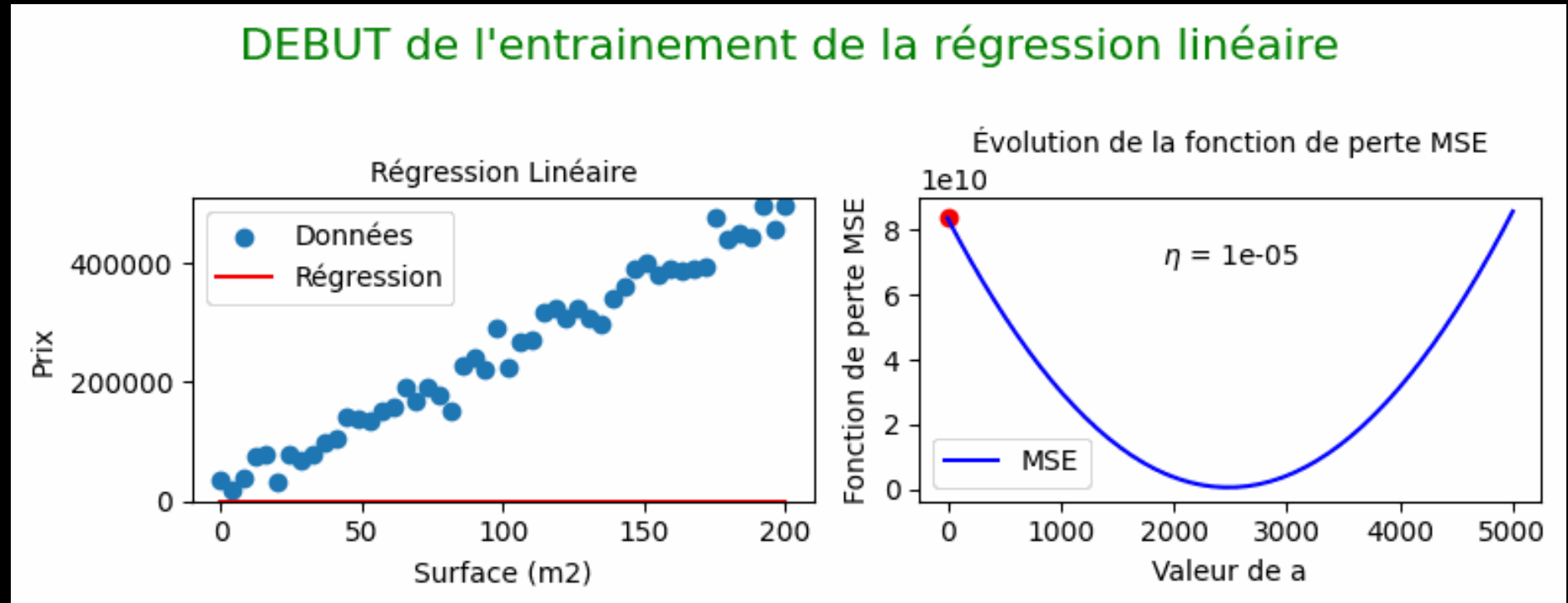
Apprentissage : cas le plus simple

Avec un taux
d'apprentissage
trop fort



Apprentissage : cas le plus simple

Avec un taux
d'apprentissage
correct



Autres types d'apprentissage

Beaucoup d'optimiseurs dérivent de la descente de gradient :

- **Momentum** : Ajoute une "inertie" lors de la descente du gradient, permettant de surmonter les obstacles et d'accélérer vers le minimum.
- **AdaGrad (Adaptive Gradient Algorithm)** : Adapte le taux d'apprentissage pour chaque paramètre en fonction de l'historique des mises à jour, réduisant le taux pour les paramètres fréquemment ajustés.
- **RMSProp (Root Mean Square Propagation)** : Ajuste le taux d'apprentissage en utilisant une moyenne mobile décroissante des carrés des gradients récents, ce qui permet une réponse plus flexible aux changements dans les gradients.
- **Adam (Adaptive Moment Estimation)** : Combine le meilleur du Momentum (utilisation de l'inertie pour maintenir la direction des mises à jour) et de RMSProp (ajustement adaptatif du taux d'apprentissage basé sur les gradients récents), pour une optimisation efficace et rapide.

Partie un peu plus technique...
Apprentissage profond supervisé

Définition de l'intelligence artificielle

Intelligence artificielle

Apprentissage automatique

Apprentissage supervisé

Régression

Classification

Apprentissage profond

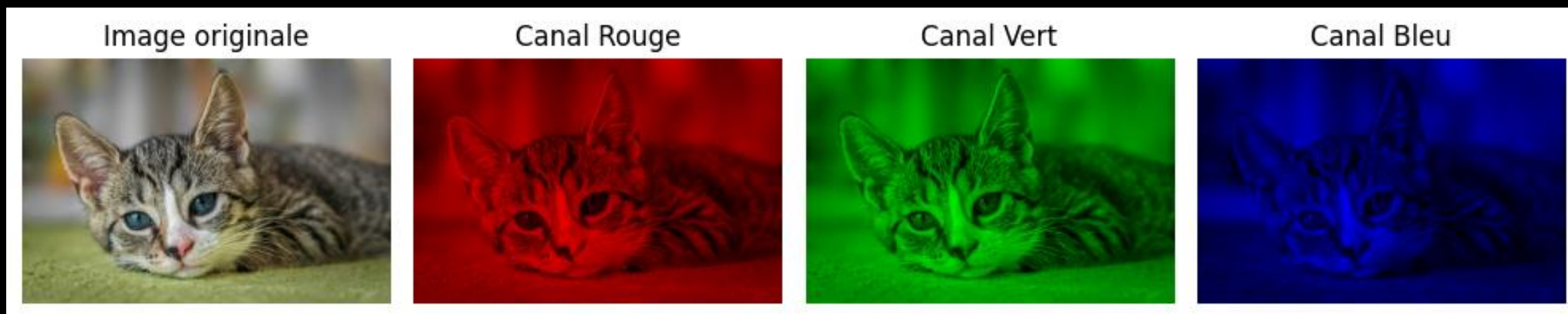
Apprentissage non supervisé

Apprentissage par renforcement

Apprentissage profond

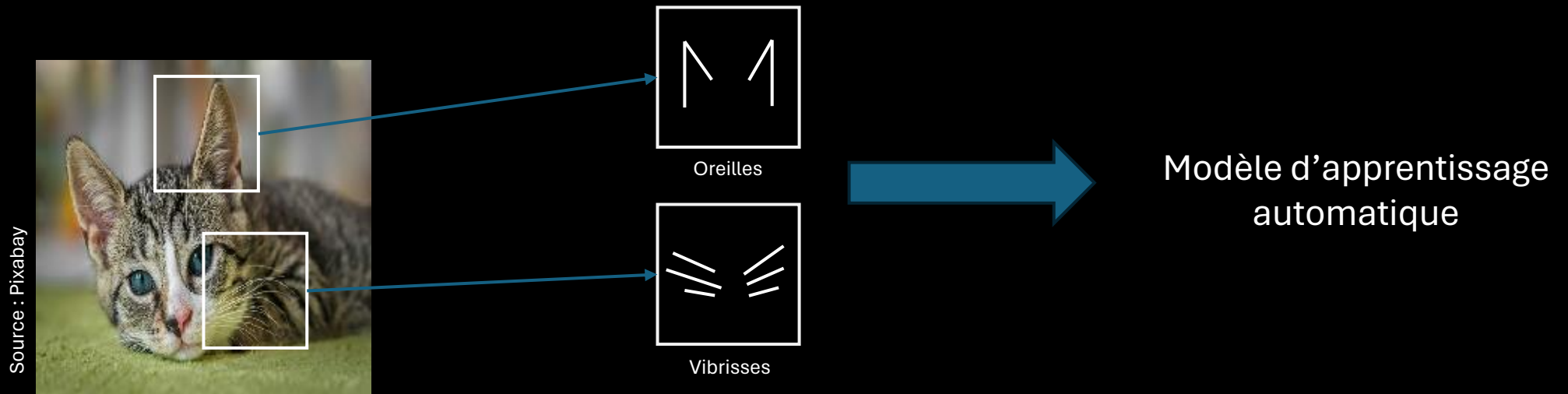
Qu'est-ce qu'une image ?

Image en couleurs = 3 tableaux de nombres pour l'intensités des 3 couleurs primaires (rouge, vert et bleu)



Détection de caractéristiques

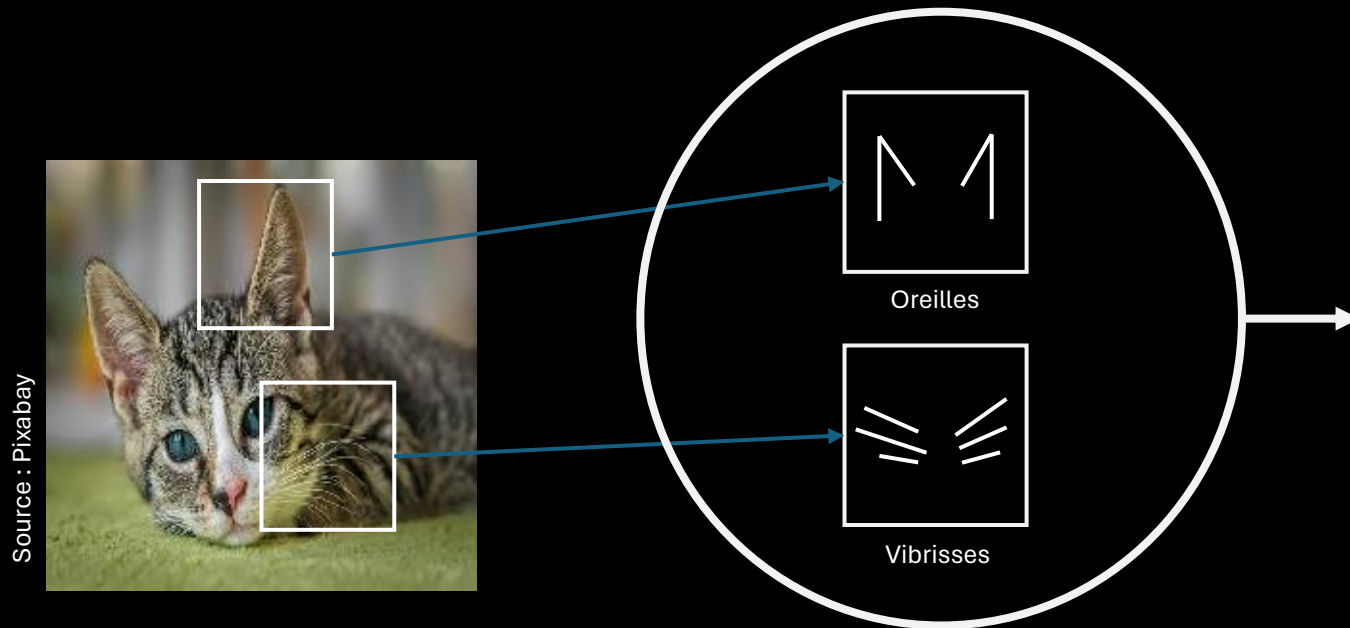
A l'ancienne : créer les caractéristiques à la main



Quelles sont les problématiques de cette solution ?

Détection de caractéristiques

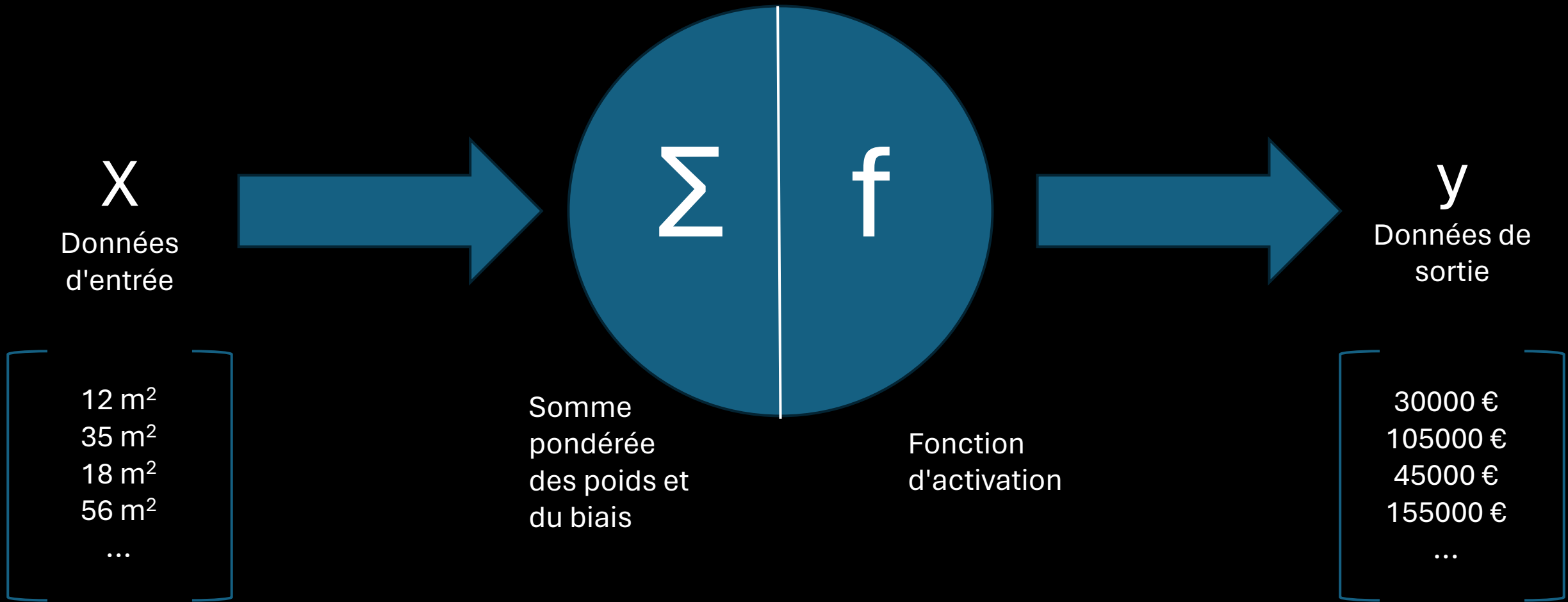
Avec des réseaux de neurones de convolution :



Caractéristiques
**appries à partir des
données**
d'entraînement par le
réseau de neurones
grâce aux noyaux de
convolution

Tout d'abord, qu'est qu'un réseau
de neurones ?

Un neurone

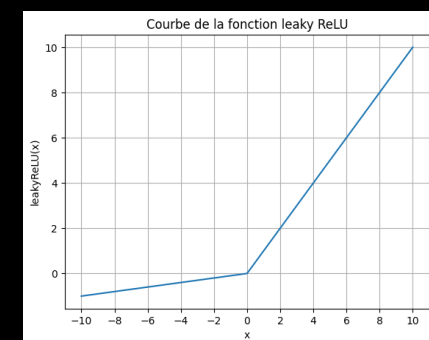
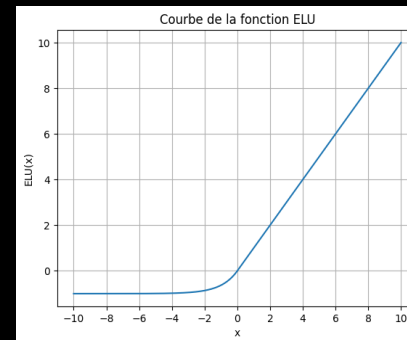
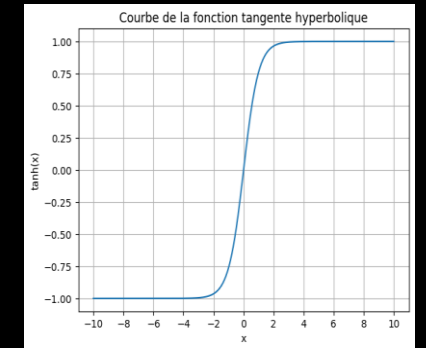
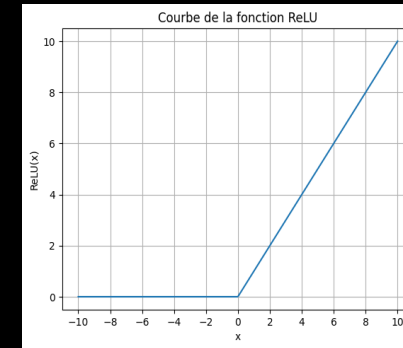
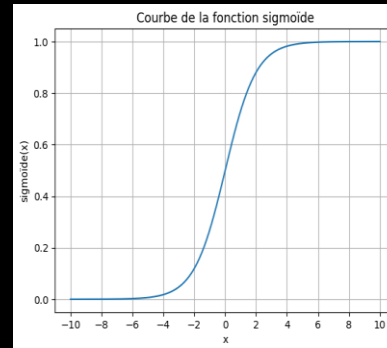


Un neurone : mathématiquement

Fonctions d'activation g

Sortie du neurone

$$\hat{y} = g\left(\sum_{i=1}^n w_i x_i + w_0\right)$$



Réseaux de neurones

Illustration d'un réseau de neurones simple avec l'application [Tensorflow Playground](#)

Tinker With a **Neural Network** Right Here in Your Browser.
Don't Worry, You Can't Break It. We Promise.

Epoch: 000,000 | Learning rate: 0.03 | Activation: Tanh | Regularization: None | Regularization rate: 0 | Problem type: Classification

DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE

FEATURES
Which properties do you want to feed in?
X1
X2
X1²
X2²
X1*X2
sin(X1)
sin(X2)

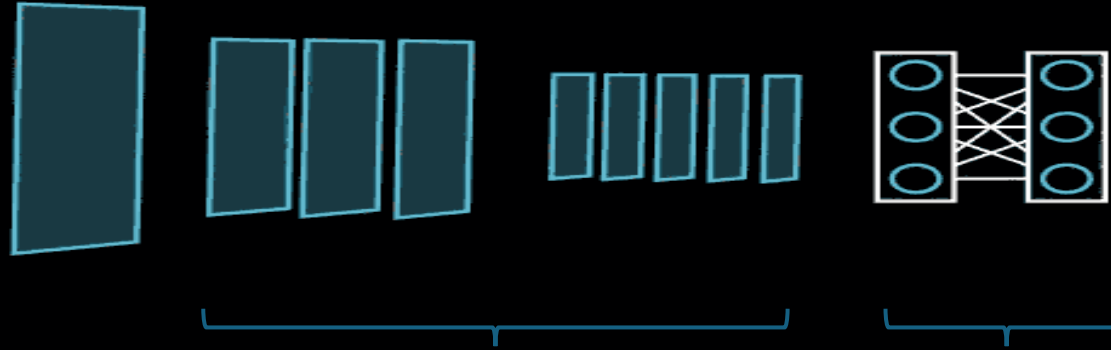
1 HIDDEN LAYER
1 neuron
This is the output from one neuron. Hover to see it larger.

OUTPUT
Test loss 0.372
Training loss 0.365

Colors shows data, neuron and weight values.
 Show test data Discretize output

Et maintenant, les réseaux de
convolution...

Convolutions



Alternance de couches convolution et de pooling

Couche de neurones denses pour calculer les probabilités de sortie

Caractéristiques bas niveau



Caractéristiques niveau moyen



Caractéristiques haut niveau



Architecture des réseaux de neurones

LeNet-5 :

- Proposé par Yann LeCun et ses collègues dans les années 1990
- Alternance de couches convolution et de pooling
- Reconnaissance des chiffres manuscrit MNIST

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 6)	156
average_pooling2d (Average Pooling2D)	(None, 14, 14, 6)	0
conv2d_1 (Conv2D)	(None, 10, 10, 16)	2416
average_pooling2d_1 (Average Pooling2D)	(None, 5, 5, 16)	0
conv2d_2 (Conv2D)	(None, 1, 1, 120)	48120
flatten (Flatten)	(None, 120)	0
dense (Dense)	(None, 120)	14520
dense_1 (Dense)	(None, 84)	10164
dense_2 (Dense)	(None, 10)	850

=====
Total params: 76226 (297.76 KB)
Trainable params: 76226 (297.76 KB)
Non-trainable params: 0 (0.00 Byte)

Réseaux de neurones de convolution

```
model = tf.keras.applications.MobileNetV2(weights='imagenet', include_top=True)
```

Architecture du réseau de neurones de convolution

MobileNetV2 :

- Architecture légère (3.5 millions de paramètres)
- Conçue pour fonctionner sur des appareils à ressources limitées (exemple : smartphones)
- Utilisée dans beaucoup d'applications mobile de reconnaissance d'images

```
=====
Total params: 3538984 (13.50 MB)
Trainable params: 3504872 (13.37 MB)
Non-trainable params: 34112 (133.25 KB)
=====
```

Différentes versions de MobileNet existent dans Keras :

- MobileNet
- MobileNetV2
- MobileNetV3

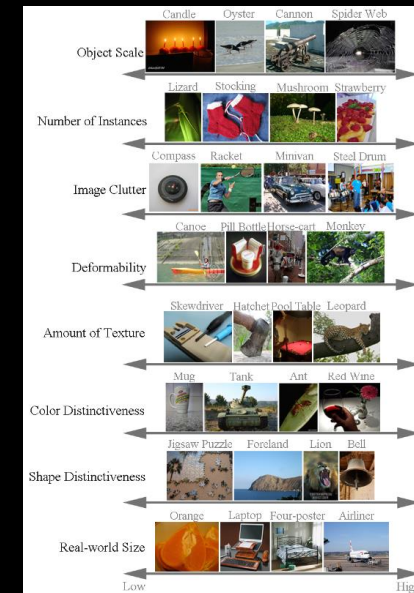
Réseaux de neurones de convolution

```
model = tf.keras.applications.MobileNetV2(weights='imagenet', include_top=True)
```

Jeu de données d'entraînement utilisé pour fixer les paramètres du modèle

Jeu de données ImageNet :

- 1000 catégories différentes (objets, animaux...) et 10 millions d'images annotées à la main
- Utilisé comme benchmark de modèles
- Base pour le concours ImageNet Large Scale Visual Recognition Challenge (ILSVRC)



Source : ImageNet Large Scale Visual Recognition Challenge

Réseaux de neurones de convolution

```
model = tf.keras.applications.MobileNetV2(weights='imagenet', include_top=True)
```

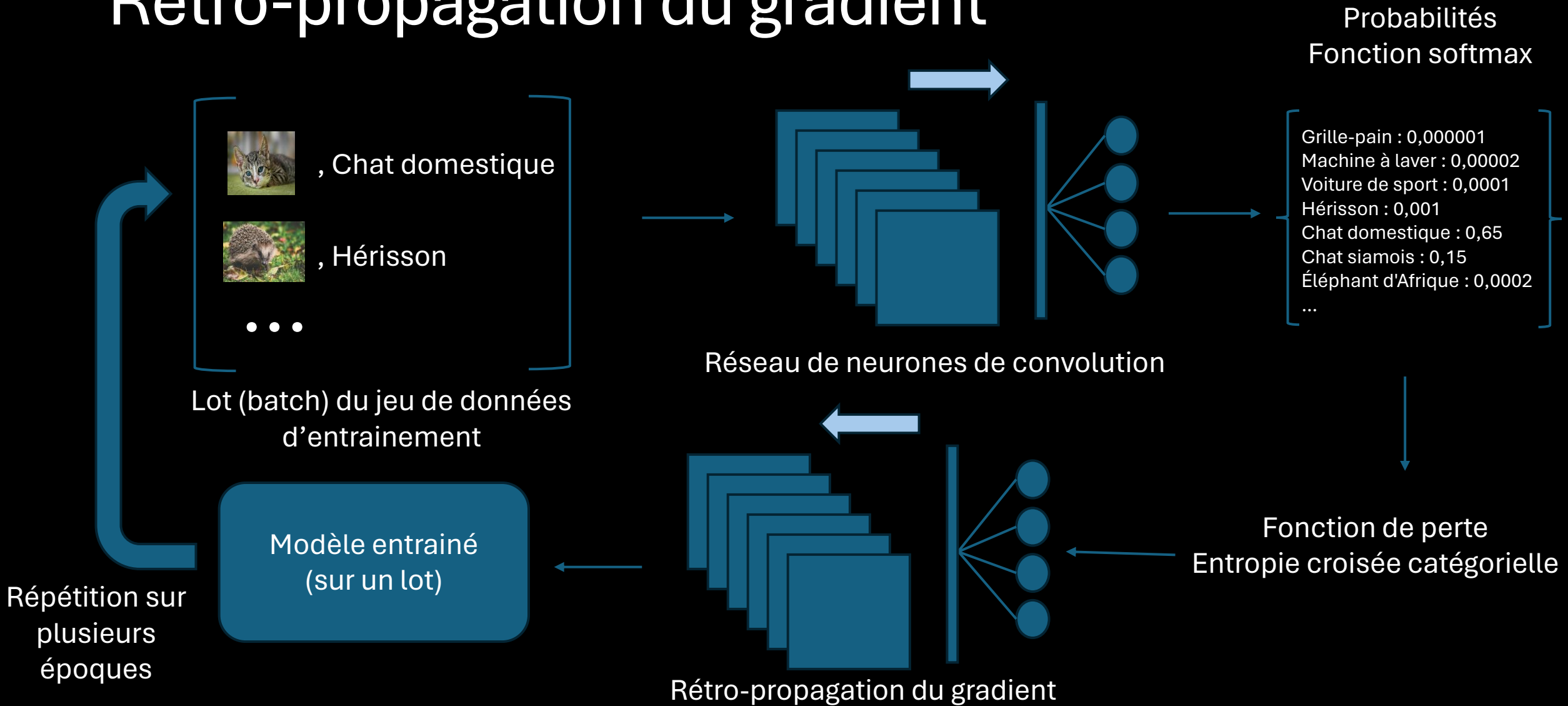
Prise en compte de la couche
de classification finale

Apprentissage par transfert avec `include_top=False` :

- Permet le réentraînement sur une tâche spécifique différente de la classification d'origine
- Supprime la dernière couche de Global Average Pooling et toutes les couches complètement connectées finales

Et enfin, la création d'images
adverse

Rétro-propagation du gradient



Fast gradient signed method (FGSM)

Source de l'image de chat : Pixabay



, Chat domestique

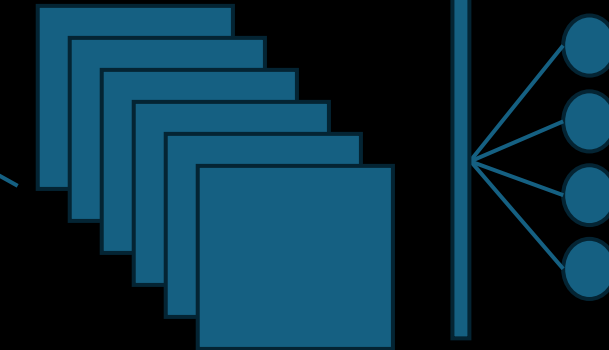


Epsilon x Signe
du gradient



Gradient sur l'entrée

Réseau de neurones de convolution



Rétro-propagation du gradient

Probabilités
Fonction softmax

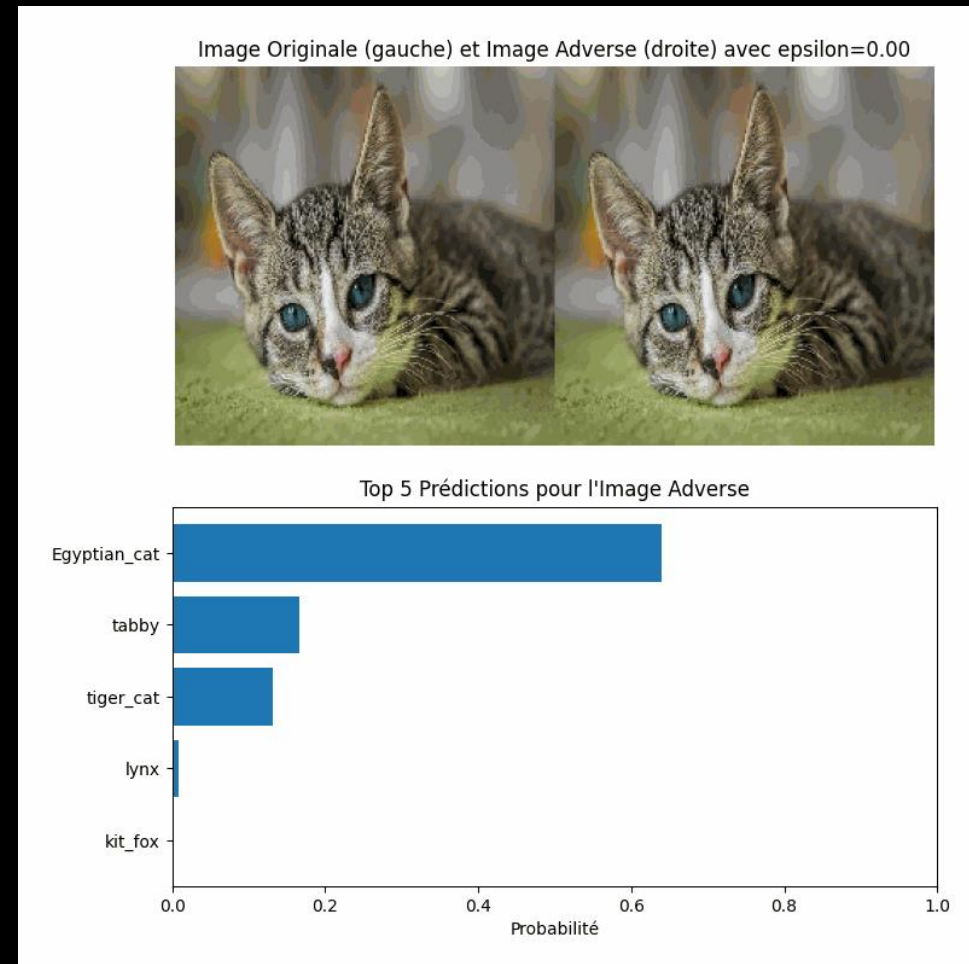
Grille-pain : 0,000001
Machine à laver : 0,00002
Voiture de sport : 0,0001
Hérisson : 0,001
Chat domestique : 0,65
Chat siamois : 0,15
Éléphant d'Afrique : 0,0002
...

- Log(0,65)

Fonction de perte
Entropie croisée catégorielle

Fast gradient signed method (FGSM)

En modifiant la valeur de epsilon entre 0 et 0.5



Signe du gradient

Pourquoi utiliser le **signe du gradient** uniquement et **pas son amplitude** ?

Rappel de l'objectif : créer une perturbation minimale mais efficace qui, ajoutée à l'image d'entrée, maximise la perte du modèle pour la classe de l'image

Raison principale : générer une perturbation uniforme

Autres méthodes plus complexes

- Projected Gradient Descent (PGD)
- Carlini & Wagner Attacks (C&W)
- DeepFool
- Jacobien-based Saliency Map Attack (JSMA)
- Boundary Attack
- Generative Adversarial Networks (GANs)
- ...

Méthodes de protection

- Entraînement Adversarial : inclure des images adverses dans le jeu de données d'entraînement
- Régularisation : utiliser des méthodes de régularisation (L1, L2, dropout...) pour mieux généraliser
- Diversification des modèles : entraîner différents modèles pour mieux généraliser
- Distillation de défense : entraîner un modèle "étudiant" à partir des sorties d'un modèle "enseignant"

Conclusion

Conclusion principale

L'intelligence artificielle **n'apprend pas les informations**
des données de la **même façon que les humains**

Ceci est vrai pour les **images** mais aussi pour le **langage** et les
autres données

L'humanisation de ces modèles amène à avoir une
confiance aveugle dans les prédictions

Conclusion principale

Donc :

- Analysez les métriques des modèles
- Privilégiez les modèles facilement interprétables
- Utilisez des outils pour interpréter les prédictions des modèles
 - Testez les limites des modèles

Conclusion complémentaire

- Ecosystème vaste de **bibliothèques Python** pour faire de l'apprentissage automatique et profond
- Méthode d'apprentissage par **descente de gradient** = base de l'apprentissage des réseaux de neurones
- Risques de **cybersécurité** à prendre en compte

Bibliographie

Bibliographie

Vidéos :

- [MIT 6.S191: Convolutional Neural Networks, 2024](#)
- [Source of confusion! Neural Nets vs Image Processing Convolution, 2023](#)

Tutoriels :

- [Adversarial example using FGSM, Keras](#)
- [Cours sur l'image, PHELMA Grenoble-INP](#)

Bibliographie

Papiers de recherche :

- [Explaining and Harnessing Adversarial Examples](#), Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy, 2015
- [Adversarial Attacks and Defences: A Survey](#), Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, Debdeep Mukhopadhyay, 2018
- [ImageNet Large Scale Visual Recognition Challenge](#), Li Fei-Fei et al., 2015
- [Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks](#), Nicolas Papernot et al., 2016