Introduction aux réseaux de neurones

julie.rabette@nokia.com

Agenda

1. Introduction aux réseaux de neurones
2. Principe de fonctionnement
3. Limitations
4. Les réseaux de neurones au quotidien
5. Frameworks et outils
6. Les réseaux de neurones et la sécurité

Les réseaux de neurones

Format et but du cours

CM 1h + TD 4h

TD :

- Utilisation d'un réseau de neurone pré-entrainé pour faire de la classification d'images

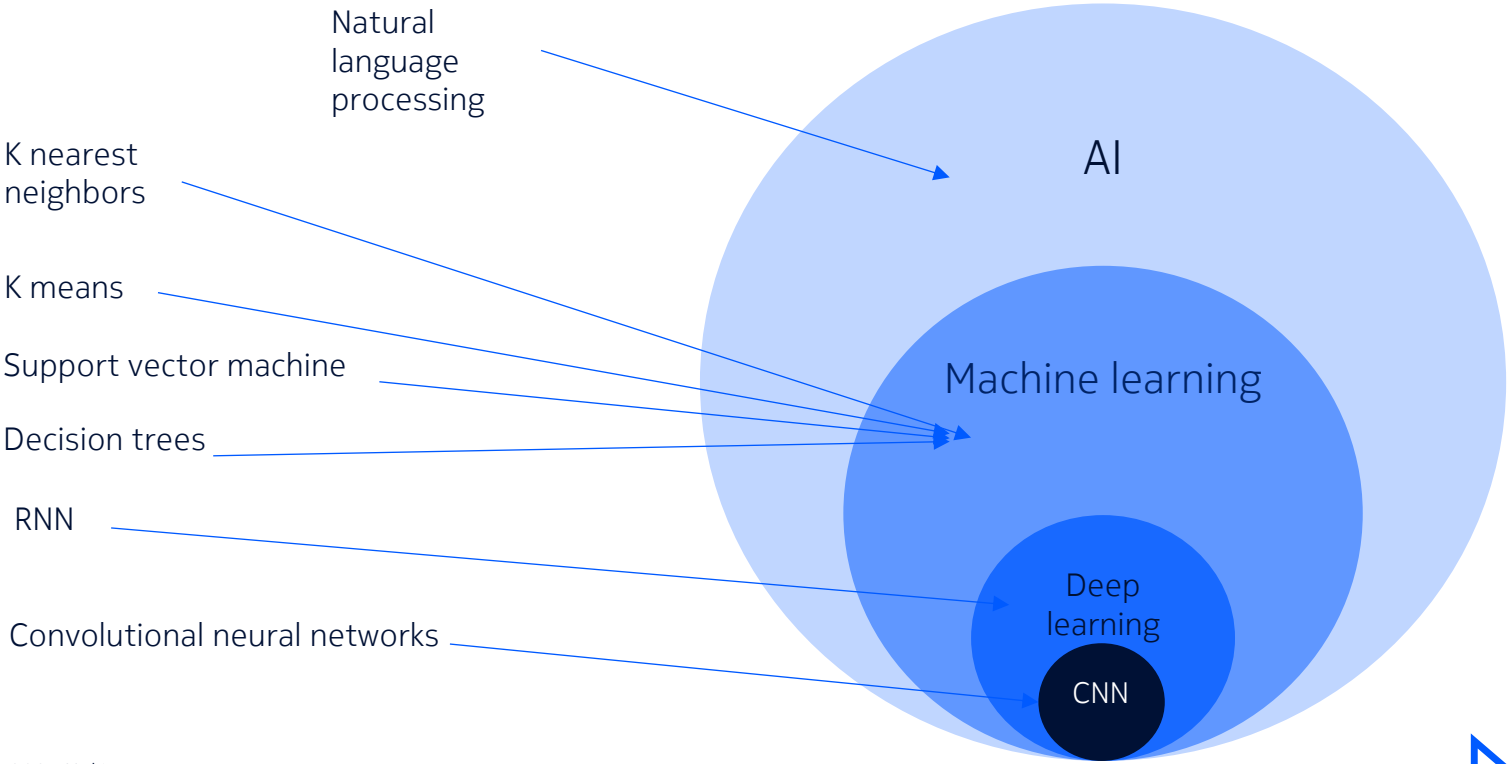
But :

- Manipuler un des framework python qui permet d'utiliser des réseaux de neurones : pytorch
- Voir comment on peut utiliser les modèles pré-entrainés
- Comprendre les différentes étapes pour utiliser un réseau de neurones

Introduction aux réseaux de neurones

les réseaux de neurones et l'IA

La place des réseaux de neurones dans le monde de l'IA



Training / Inférence

Les NN font parties des techniques d'apprentissage supervisé

forward
backward

Training

Réseau de neurones

Prédiction du NN

Calcul de la distance entre la prédiction et le label

Prise en compte de la distance pour calculer les nouveaux poids

Mise a jour des params du NN

Inférence

Réseau de neurones

Prédiction du NN

Classe = chat

Dans quel cas peut-on utiliser les réseaux de neurones ?

Taille du dataset

Les NN ont besoin de grand dataset pour un entrainement efficace

Ressources

Le training d'un NN est couteux en terme de calcul => GPU/énergie

Type de données

Les NN sont efficaces sur les données non structures telles que les images / les textes / le son

Cas où les reseaux de neurones ne sont pas les plus adaptés pour résoudre le pb :

Traitement d'images avec un petit dataset => pas assez de données => voir plutôt les méthodes de descentes de gradient (SGD)

Traitement de données (pas image/pas son/pas texte) avec des catégories => arbre de décisions

Exemple matlab

Quel algo pour quel problème à résoudre ?

Matlab propose d'utiliser les critères suivants pour choisir un algo :

- Taille du dataset
- Temps d'apprentissage
- Facilité a voir si l'algo converge
- Facilité du tuning des param de l'algo

Which Machine Learning Algorithm Is Right for You?

Algorithm	Dataset *What is the ideal dataset size for each algorithm?*	Training Speed *How quickly will the algorithm train without acceleration hardware?*	Interpretability *How hard is it to see how the algorithm arrived at a decision?*	Tuning *How much tuning does the algorithm allow?*	Comments
Linear models	Small	Very fast	Easy	Minimal	Widely used basic algorithm Linear SVM handles high-dimensional data well
Decision trees	Small	Very fast	Easy	Some	Good generalist algorithm, check for overfitting
(Nonlinear) Support vector machine	Medium sized	Moderately slow	Difficult	Some	Good accuracy
Nearest neighbor	Medium sized	Moderately fast	Moderately easy	Minimal	Lower accuracy, but easy to use and interpret
Naïve Bayes	Medium sized	Very fast	Moderately easy	Some	Widely used for text analytics (e.g., spam filtering); kernel Bayes will run slower
Ensembles	Large	Moderately fast	Difficult	Some	Higher accuracy with a tradeoff of lower interpretability
Neural network (shallow)	Medium sized	Moderately fast	Moderately easy	Some	Still used for signal classification, compression, and forecasting
Deep nets	Large	Very slow	Difficult	A lot	A standard algorithm for image, video, signals, and text

A quoi servent les réseaux de neurones ?

- Initialement les réseaux de neurones ont été utilisés avec le traitement d'image
- Ils sont maintenant utilisés dans de nombreux domaines

Domaines

Reconnaissance images / voix / signaux

Diagnostic medical

Base des LLM

Détection d'anomalies (fraude, attaque ...)

Systèmes de recommendations

Prévisions météo, eco, ventes ...

Classification

Binaire (chat/ non chat)

Mutiple (plus de 2 classes en sortie)

Attention il faut connaître le nombre de classes que l'on veut en sortie

Régression

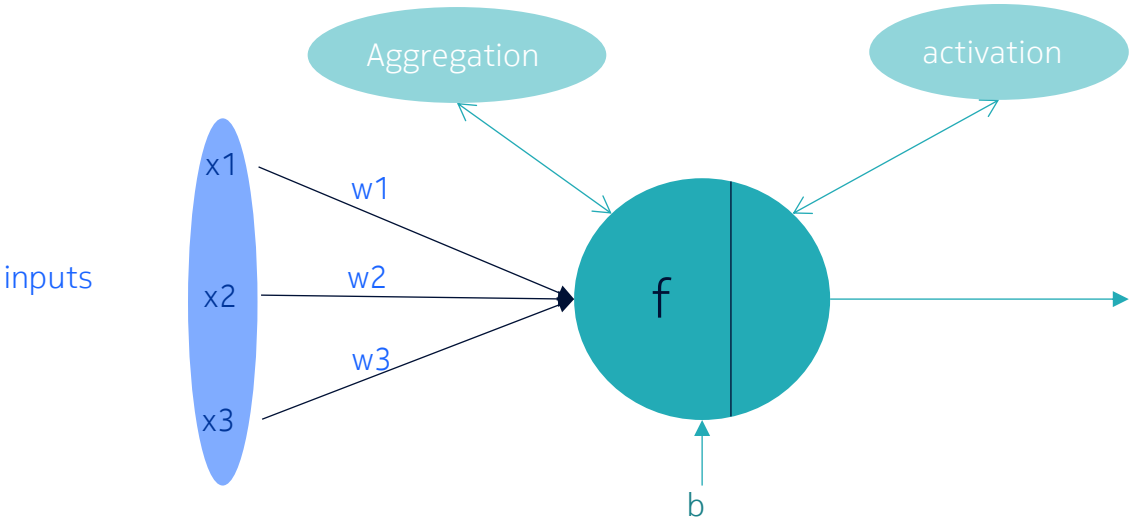
Prédiction d'une valeur de sortie à partir de valeurs d'entrées

Exemple prediction de precipitation, de temperature, ...

Principes de fonctionnement

Principe de fonctionnement d'un neurone

1 neurone : fonction d'aggrégation + activation

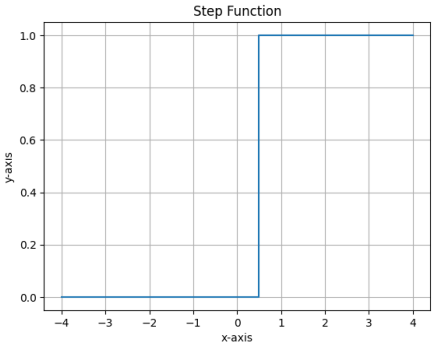


Paramètres du neurone qui sont à optimiser :

w => poids (weight)
b => biais (bias)

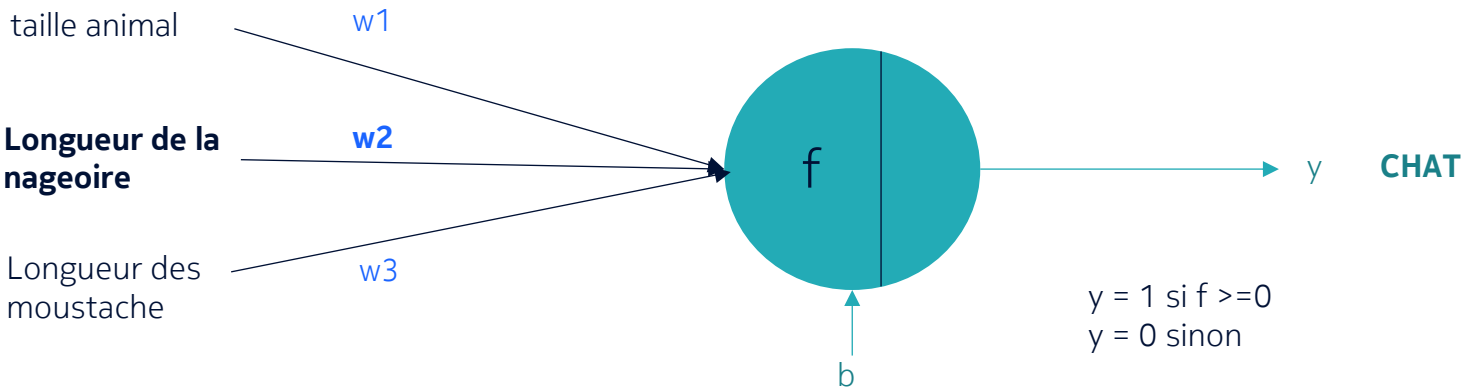
f(xi) = w1x1 + w2x2 +w3x3 +b

y = 1 si f >=0
y = 0 sinon



Principe de fonctionnement d'un neurone

1 neurone : apprentissage renforce les poids qui amènent a la bonne prediction



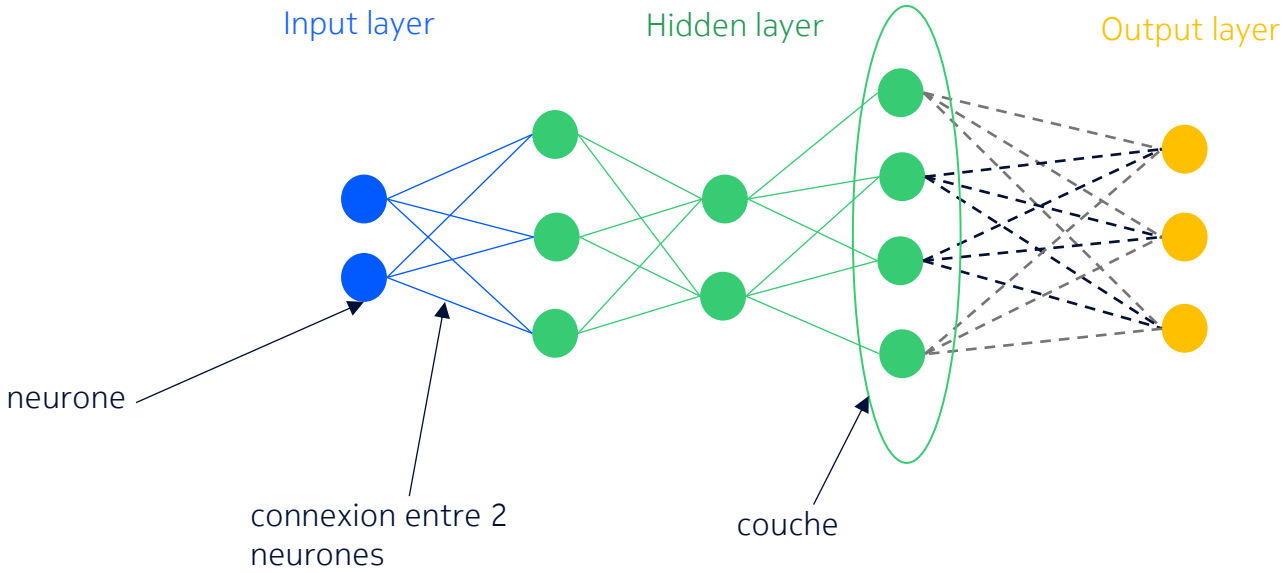
y = 1 si f >=0
y = 0 sinon

Si y =1 classe = chat
sinon classe = NON chat

Durant le training le poids w2 va diminuer au point que la participation dans l'aggregation de 'longueur de la nageoire' soit negligeable

Structure d’un réseau de neurones

Organisation en layers (couche) d’un réseau de neurones



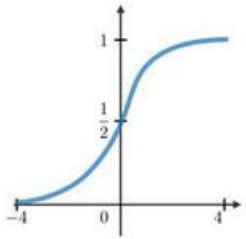
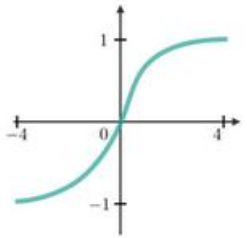
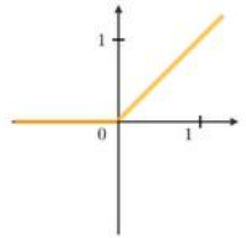
Types de layers pour les convolutional neural network (CNN)

- **Layers**

- Convolution layer : opération de convolution (contient des weights qui sont appris pendant le training)
- Pooling layer : réduction de la taille
- Fully Connected layer : connecte les neurones à toutes les activations de la couche précédente
- Batch norm layer : opération de normalisation
- Flatten layer : mise a plat des dimensions (tensor d'une seule dimension en sortie)

- **Layers d'activation**

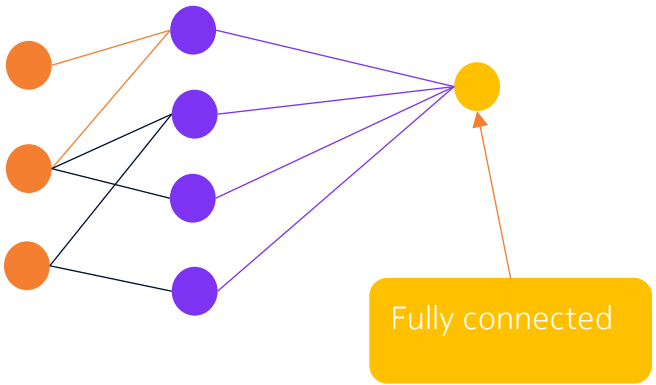
- Permet d'insérer de la non-linéarité dans le CNN
- Différentes fonctions d'activation possibles

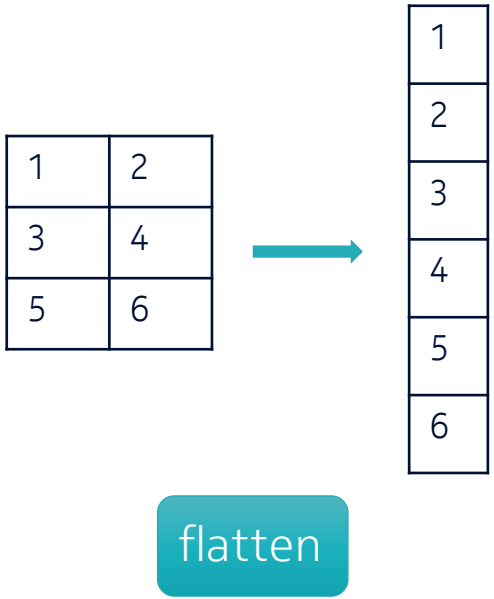
Sigmoid	Tanh	ReLU
$g(z) = \frac{1}{1+e^{-z}}$	$g(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$	$g(z) = \max(0, z)$
		

- Softmax : transforme la sortie en probabilité (pour les classifiation)

Types de layers pour les convolutional neural network (CNN)

Exemple de pattern Classique en CNN





Types de layers pour les convolutional neural network (CNN)

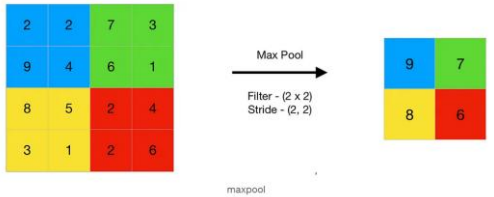
Les layers classiques

Conv

Operation de convolution

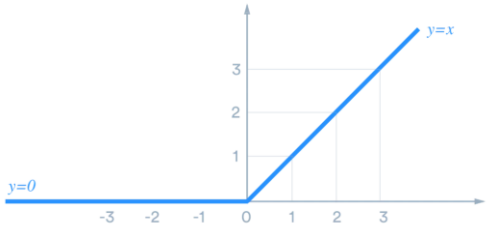
Pool

Reduction de la taille de la matrice d'entrée



FC

Fully connected

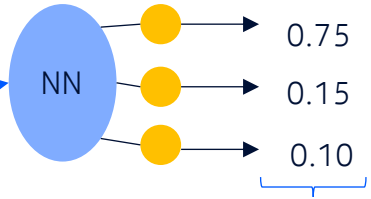


Relu

Activation

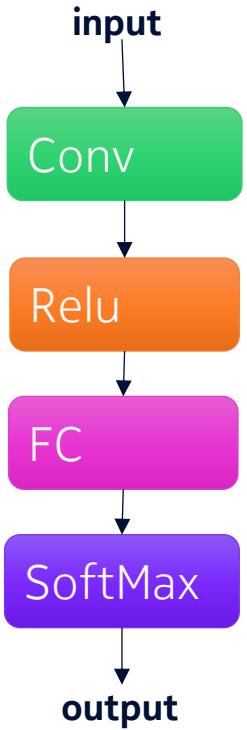
SoftMax

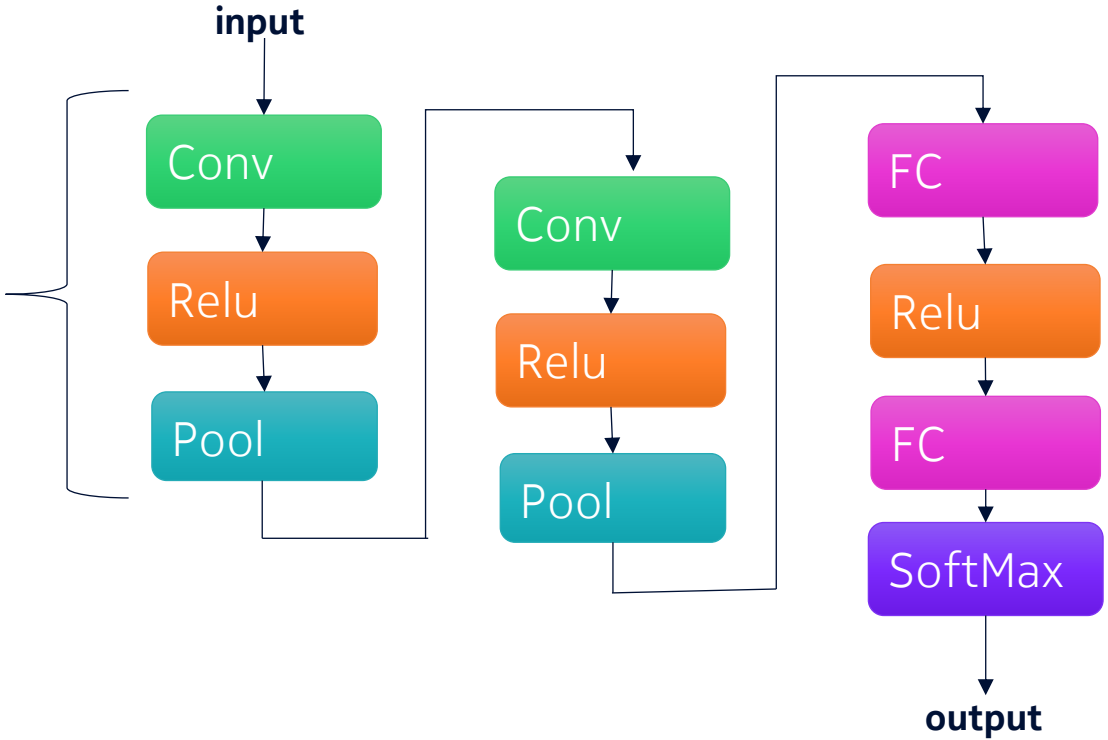
Permet d'avoir des probabilités sur la sortie du NN



Types de layers pour les convolutional neural network (CNN)

Exemple de pattern Classique en CNN





Les architecture de réseau de neurones convolutif

Architectures utilisées dans le monde du ML en utilisant des patterns connus

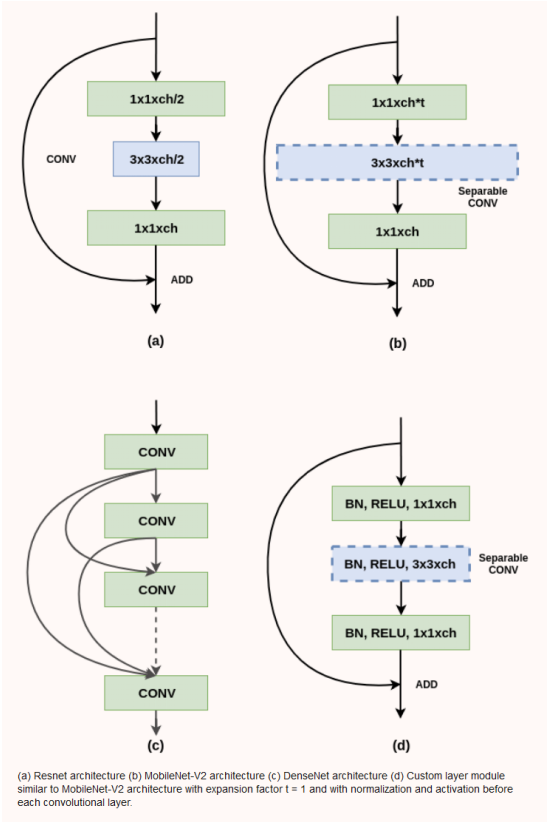
- Restnet V1, V2
- mobilenet V1, V2
- densenet
- ....

En TP on va utiliser un Resnet

resnet

mobilenet

densenet



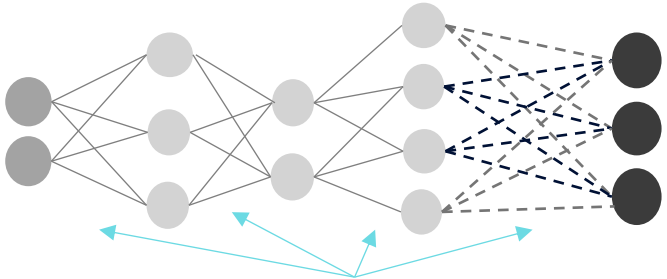
https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8

Training

Grandes phases du training

Training

Matrice de pixel + label = ‘chat’



Prédiction du NN

Calcul de la distance entre la prédiction et le label

Prise en compte de la distance pour calculer les nouveaux poids

Mise a jour des poids

Forward propagation

Calcul de la loss function

backward propagation

Loss function ou function de coût

Distance entre la prédiction du modèle et le résultat attendu (label)

BUT du training :

⇒ **Optimisation** de function de coût (loss function)

⇒ Modifier les poids du modèle pour **minimiser le loss**

⇒ Modifier les poids des neurones de sortie et propager cette optimisation aux neurones de la couche précédente (retropropagation / backward)

- Loss pour de la régression (valeurs continues)
  - Erreur quadratique moyenne (MSE)
  - Erreur absolue moyenne (MAE)
  - L1 lisse

- Loss pour de la classification (valeurs discrètes)
  - Cross entropy binaire
  - Cross entropy multiple
  - Hinge embedding loss

https://www.geeksforgeeks.org/ml-common-loss-functions/
https://www.geeksforgeeks.org/ml-common-loss-functions/

Limitations

Importance des données de training

Qualité des données => capital

Shit in shit out

Garbage in, Garbage out
(George E. P. Box statistician)

Principe de l'apprentissage supervisé :

Entrainer un modèle avec un dataset de training (**représentatif** du **problème** que l'on veut résoudre)

On attend du modèle qu'après le training il soit capable de **généraliser** et de pouvoir **prédire** correctement sur des données **inconnues**. (pas dans les données de training)

Si les données de training ne sont pas correctes, ne représentent pas le problème a résoudre, le model apprendra des choses incorrecte et donnera de mauvaises predictions

⇒ "biais" des LLM selon les données d'apprentissage
⇒ "biais" de l'IA (pour l'obtension de prêt, le recrutement, cibler les controles, ...)
⇒ https://www.numerama.com/tech/426774-amazon-a-du-desactiver-une-ia-qui-discriminait-les-candidatures-de-femmes-a-lembauche.html

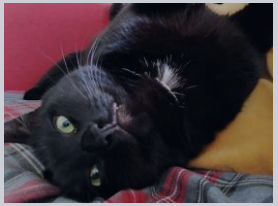
Importance des données de training

Données erronnées : mauvaise labellisation

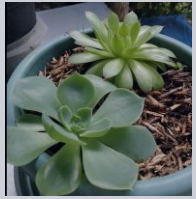
Shit in shit out

Dataset training

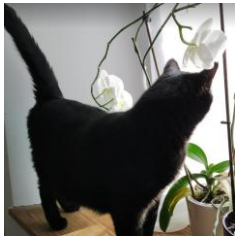
mouton



panthère



plante



Model

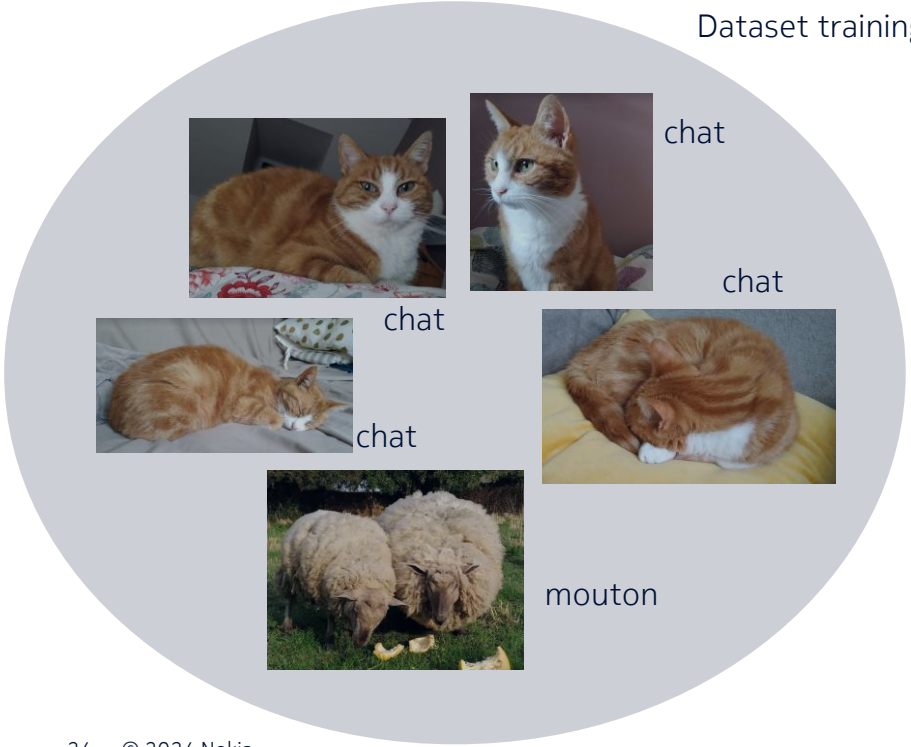
Prediction

?

Importance des données de training

Représentativité du dataset training : données biaisées

Shit in shit out

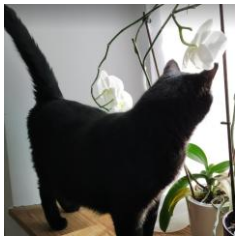


Model

Prediction

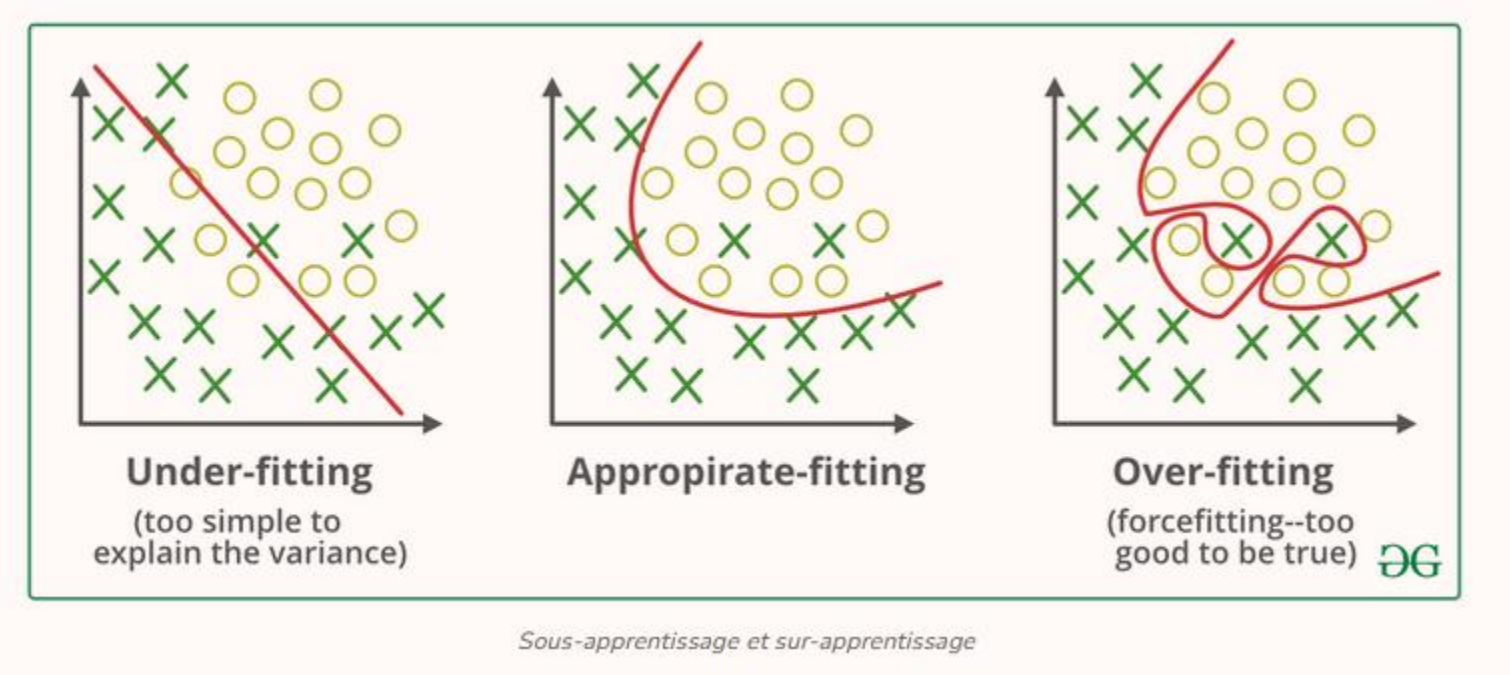
chat

?



Les problèmes que l'on peut avoir en apprentissage supervisé

Overfitting & underfitting

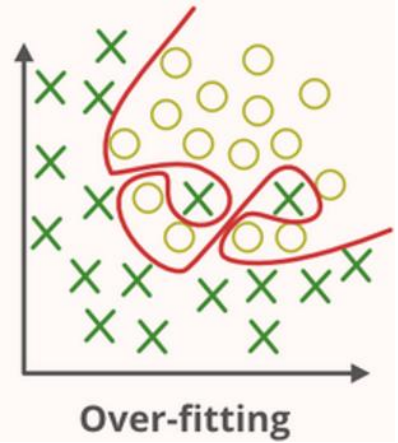


Sous-apprentissage et sur-apprentissage

Les problèmes que l'on peut avoir en apprentissage supervisé

Overfitting

- Le modèle apprend "par Coeur" les données de training
- Il prend en compte notamment des points qui sont du "bruit" (non représentatif, point a la marge).
- Il a de très bonnes perf sur le dataset de training
- Il n'arrive pas a généraliser sur des données inconnues
- Arrive quand on utilise un modèle très complexe pour gérer un pb simple mais bruité



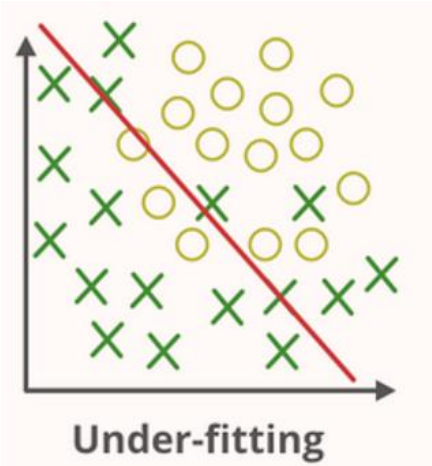
Solutions:

- **Split du dataset** en train/test et vérifier les perf sur des données inconnues (il faut un dataset assez grand)
- **Cross validation**
- **Normalisation**
- **Early stopping**
- Réduire la complexité du modèle

Les problèmes que l'on peut avoir en apprentissage supervisé

Underfitting

- Le model n'arrive pas à apprendre
- Peut arriver si on a prévu un modèle trop simple pour un pb complexe
- (ex : une regression linéaire pour un pb polynomial)
- Peut arriver si les données sont très bruitées où avec des valeurs erronnées
- Taille du dataset de training insuffisante



Solutions:

- Utiliser un **modèle plus complexe**
- Supprimer le **bruit** des données d'entrée
- **Augmenter le nombre d'epoch** pour le training

Les réseaux de neurones au quotidien

Les différentes façons d'utiliser des NN

From scratch

Init des poids de façon aléatoire

Définition de l'architecture et de la profondeur du reseau de neurones

Contraintes:

- Dataset très **important**
- Nombreuses GPU
- Temps de training important
- Connaissances poussées en NN

Transfer learning

On se base sur un **modèle pré-entrainé** qui va etre ré-entrainé.

Remplacer la dernière couche du modèle par **une couche spécifique** a notre problème

Les **poids** de toutes les couches sauf la dernières vont être **figés**
Seules les couches spécifiques vont être entrainées

=> Nécéssite un dataset de training plus petit que pour un training from scratch et le fine tuning

Reconnaissance large éventail de photos d'animaux / reconnaissance de races de chien

Fine tuning

On se base sur un **modèle pré-entrainé** qui va etre ré-entrainé.

Remplacer la dernière couche du modèle par **une couche spècifique**

Pas de freeze sur les poids du modèle : **tous les poids vont être mis a jour**

=> Nécessite moins de data que le from scratch mais plus que le transfert learning

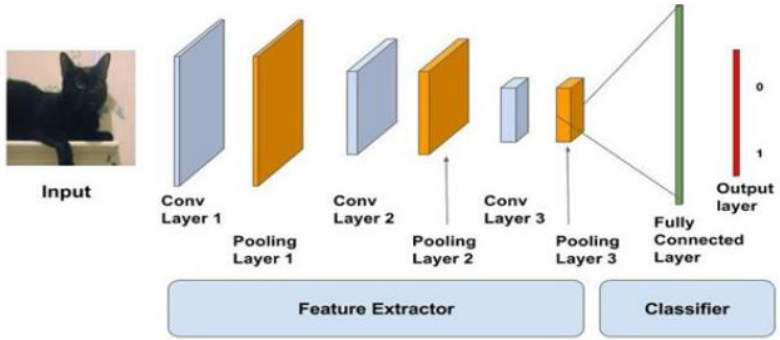
Reconnaissance générale d'object sur des images / reconnaissance de fractures sur des radios

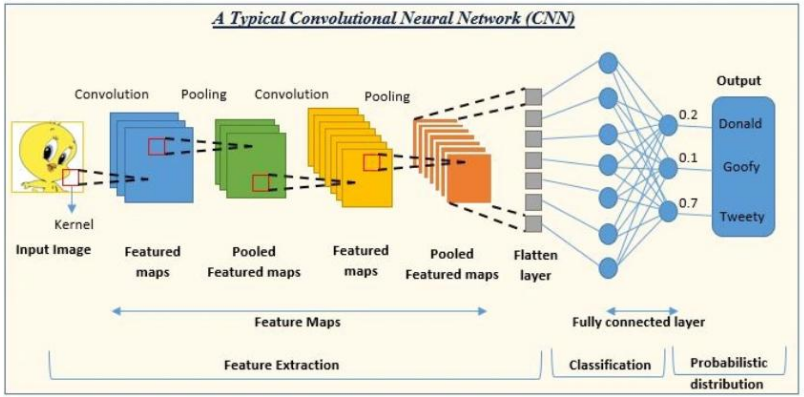
Utilisation d'un réseau de neurone pour de la classification

Feature extraction / classifier

- Les derniers layer du NN (flatten / FC /softmax) servent pour la classification

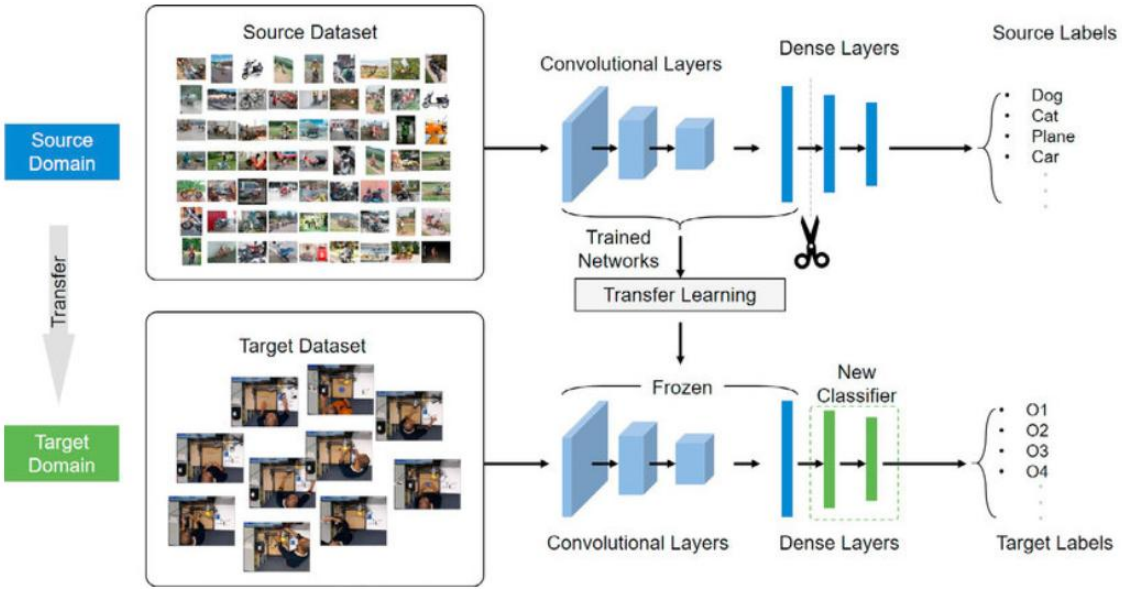
- Les layers d'avant font du feature learning :
  - Extraction d'infos à partir de l'image





Transfert learning :

Utiliser la partie extraction feature / adapter le classifier a notre use case



The architecture of our transfer learning model.

Frameworks et outils

Open-source libraries pour le deepLearning en python

https://builtin-com.translate.goog/data-science/pytorch-vs-tensorflow?_x_tr_sl=en&_x_tr_tl=fr&_x_tr_hl=fr&_x_tr_pto=rq

Keras

- API de haut niveau
- Backend tensorflow
- Utilisé pour faire des PoC rapidement
- Support CNN et RNN et même un mixte
- Le plus user friendly
- API haut niveau de tensorflow

Tensorflow (google)

- API de haut et bas niveau
- Static computation graph
- Visualisation tensorboard
- Utilisation CPU / GPU
- Utilisé en recherche et en prod
- Pour un dev plus compliqué a prendre en main que pytorch
- Plusieurs niveau d'abstraction possibles

Pytorch (facebook)

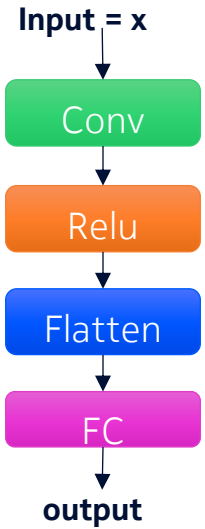
- API de bas niveau
- Dynamic graph
- Basé sur la librairie torch
- GPU HW acceleration
- Tres proche du dev python
- Etait un peu moins utilisé que tensorflow mais maitenant c'est équivalent
- Inclus des lib comme torchVision
- DDP pour le //

Un réseau de neurone en pytorch

C'est une classe qui hérite de torch.nn.Module

Init :
On définie les layers du NN

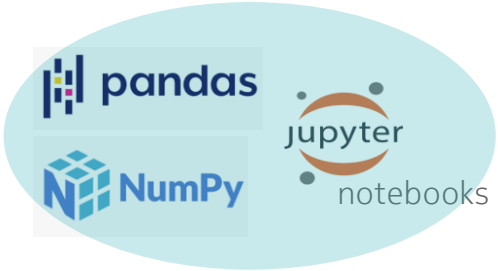
Forward :
On explique comment le forward est calculé




```python
class mon_model(nn.Module):
    def __init__(self):
        super(mon_model, self).__init__()
        # conv layer
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=16, kernel_size=3, padding=1)
        # shape de la sortie de la conv : (batch_size, 16, 50, 50)
        # relu layer
        self.relu1 = nn.ReLU()
        # Flatten layer
        self.flatten = nn.Flatten()
        # Fully connected layer
        self.fc1 = nn.Linear(16 * 50 * 50, 2)  # => on a 2 classes en sortie donc dimension sortie = 2

    def forward(self, x):
        x = self.conv1(x)
        x = self.relu1(x)
        x = self.flatten(x)
        x = self.fc1(x)

        return x


input = torch.randn(1, 3, 50, 50) # 1 = batch size / 3 = number of channels / 50 = height / 50 = width
model = mon_model() # instanciation du modèle
result = model(input) # forward pass : on passe l'image dans le modèle : la prediction = result
```

L’ecosystéme pour le ML en python

Les librairies utilisées pour le ML python



utilitaires

mlflow

Sauvegarde des model .pth ou .onnx et des plots

Outils pour la visualisation

Sauvegarde des dataset

Librairie ML

Format pour l’échange des models

Visualisation pendant le training des poids, biais ...

Hyperparameters tuning

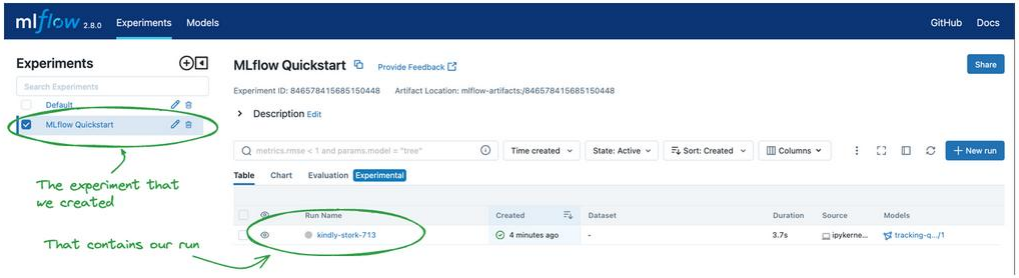
Mlflow

Pour le tracking des modèles

https://mlflow.org/docs/latest/index.html

Experiment :

- Contient plusieurs runs
- 1 run = un lancement de training



The main MLflow Tracking page, showing Experiments that have been created

Pour un run on enregistre :

- Le model qui a été généré par le training (format .th / .onnx / tensorflow)
- Les params du training
- Les metrics : loss / accuracy mais on peut aussi avoir des plots (qui peuvent être générés pendant le training)
- Les checkpoints

On peut aussi comparer plusieurs (très pratique)

Les réseaux de neurones et la sécurité

Attaques sur les réseaux de neurones

Types d'attaques

Par manipulation

Détourner le comportement du modele en prod

Requêtes malveillantes

DoD, réponses inatendues

Par infection

Pousser a la mauvaise prediction

Infecter le dataset de training ou insertion de malware

Mauvaises réponses du modèle ou utilisation du malware

Par exfiltration

Obtenir des informations du modèle

Requetes malveillantes

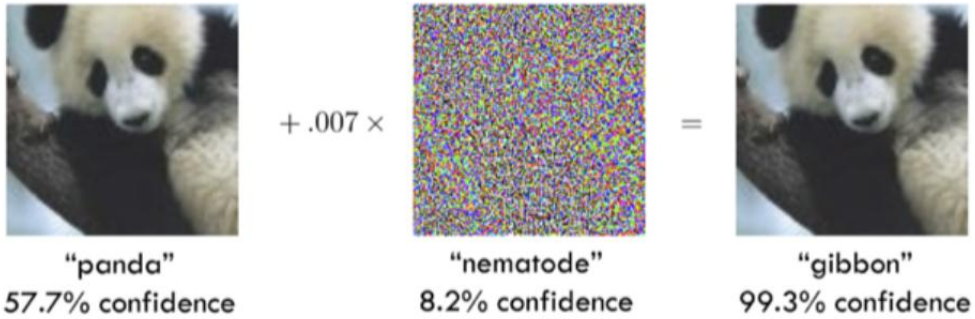
Fuite de données d'entrainement, des poids du modèle ...

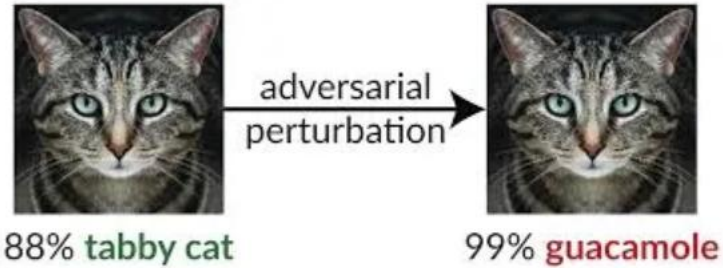
Exemple attaque par manipulation

Adversarial attack

Attaque sur un modèle en inférence

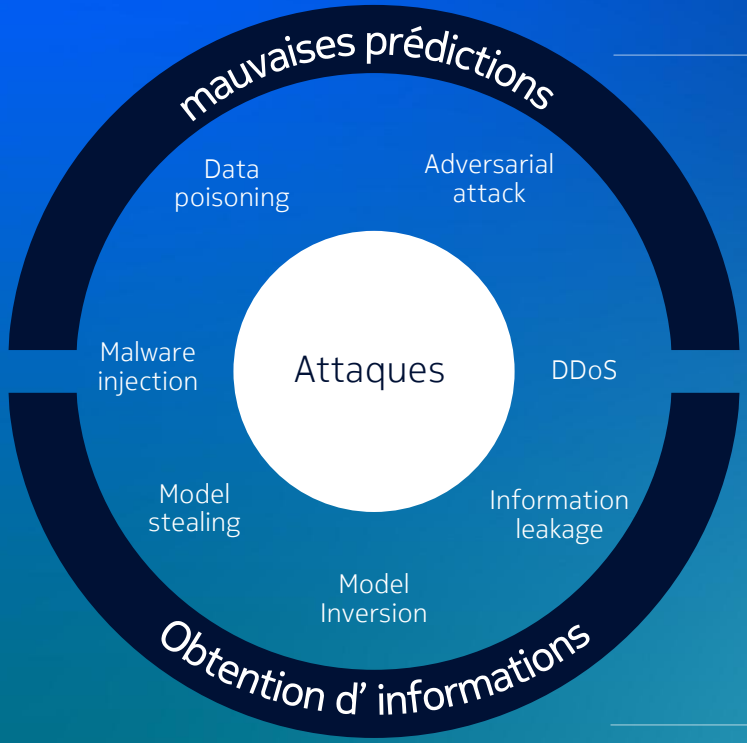
On ajoute du bruit a la requête invisible pour les humain => le bruit va provoquer la mauvaise prediction du modèle





https://www.futura-sciences.com/tech/actualites/intelligence-artificielle-voici-pulls-anti-reconnaissance-faciale-103008/

Sécurité : les types d’attaques sur un réseau de neurone



Accès aux données d’entrainement
Data poisoning / Adversarial attack

Accès au modèle
Malware injection, backdooring attack, trojaning attack

Accès a l’API du modèle en inférence
Model extraction=stealing / inversion / information leakage

Accès a l’API du modèle en inférence + but = tuer le service
DDoD

IA Act

Réglement européen qui encadre l'utilization de l'IA

- **But** :
- Protéger les **droits fondamentaux**, la santé et la sécurité des citoyens européens face aux risques de l'IA

- Classification par **risque** des systèmes d'IA:
  - Risque minimal
  - Risque élevé (exigences strictes pour santé, justice ..)
  - Pratiques interdites

- Les trucs **interdits** :
  - Techniques de manipulatrices ou trompeuses
  - Catégorisation biométrique déduisant des attributs sensibles (vie sexuel, religion, opinions politique, etat de santé ...)
  - Notation sociale
  - Evaluation du risque qu'une personne commette des infractions pénales
  - Faire des bases de données de reconnaissance faciale
  - Déduction des émotions sur le lieu de travail

Infos pratiques

Les rôles dans une équipe de dev ML

Il y a plusieurs rôles avec des compétences différentes dans une équipe qui fait du ML

Data scientist

Archi du model

Hyperparam du training

Analyse du training

Dev Python ML

Dev python pour :

- Les outils
- les différentes chaines
- La connexion aux bases
- La connexion a mlflow

MLOps

DevOps (CI/CD,gitlab, serveurs, ...)

Gestion des serveur de GPU

Gestion des pipeline special ML

Informations utiles

Tuto / cheat sheet

https://pytorch.org/tutorials/beginner/ptcheat.html

https://www.learnpytorch.io/pytorch_cheatsheet/

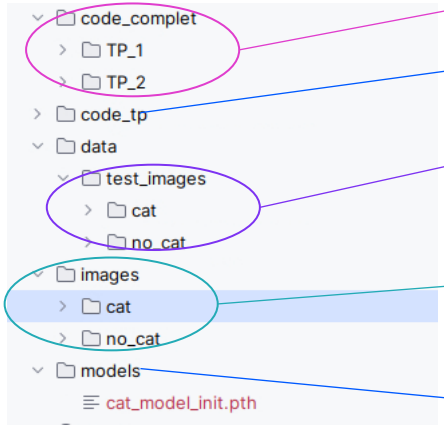
Ce dont vous aurez besoin pour le TD : **Env python**

pytorch /
matplotlib / torchvision /
tqdm /
PIL / numpy /
seaborn / sklearn

le code du TP est a aller chercher sur github : **cats_detector**

Les TPs

détecteur d'images de chats



Le code complet du TP

Le repo ou il y a du code a trou

Le repo ou se trouvent les images de test : ne pas les utiliser dans le dataset de training

Le repo ou se trouvent les images du dataset de training

Le repo pour sauvegarder les modèles entrainés

Les TPs

Détecteur d'images de chats

- TP1

- On prend un resnet pytorch pre-entrainé
- On va utiliser ce model en inference sur un dataset de test
- But = prise en main de pytorch
- Comprendre comment on interprete une prediction du model

- Utilisation du repo **test_images**

- TP2

- On utilise le model du TP1 et on va faire du transfert learning
  - Changer le model et refaire une etape de training
- En sortie du modèle uniquement 2 classes (cat/no_cat)
- Ne pas utiliser les images de test dans le dataset de training
- But : faire plusieurs training en changeant les param et trouver un modele qui a la meilleure prediction sur le dataset de test (test_images)

- Utilisation du repo **images** comme dataset de training

TP 2
training

Training avec un optimizer

=> module qui fait la mise a jour des parametres du modele. Son but est de minimiser la function de cout

=> 2 actions : calcul des gradients + mise a jours des parametres

=> l'optimizer a un learning rate

Training sans scheduler :

=> il permet de changer dynamiquement pendant le training le learning rate (de l'optimizer)

:

TP 2

Les paramètres qui influent sur le training

Les paramètres que l on peut changer :

```
35  #------------------------------
36  # parametres utilisés dans ce fichier
37  batch_size = 4 #taille du batch pour l'entrainement
38  ratio = 0.4 #ratio pour la taille du dataset de validation par rapport a la taille du dataset
39  # d'entrainement
40  optimizer_lr = 1e-4 #learning rate de l'optimiseur
41  optimizer_weight_decay = 0.001 #weight decay de l'optimiseur
42  nb_epoch = 4  #nombre d'epoch pour l'entrainement
```

- changer le ratio dataset training/test
- changer le learning rate
- changer le weight_decay
- changer le nombre d'epoch
- changer les images qui sont dans le dataset de training (en ajouter sans ajouter les photo de test bien sur !)
- changer l'optimizer => prendre un optimizer pytorch différent d'Adam
- changer la loss function => prendre une loss function differente de CrossEntropy
- ajouter un scheduler

Liens utilisés pour le cours

https://datascientest.com/convolutional-neural-network
https://www.researchgate.net/figure/The-comparison-between-the-general-connections-and-the-residual-connections-used-in_fig2_352408937
https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
https://www.linkedin.com/pulse/understanding-activation-functions-neural-networks-guide-davis-joseph-aswpe/
https://www.researchgate.net/figure/Artificial-neural-network-activation-functions-In-this-figure-the-most-common_fig8_344331692
https://pyimagesearch.com/2021/05/14/convolutional-neural-networks-cnns-and-layer-types/
https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/
https://www.linkedin.com/pulse/techniques-reduce-overfitting-andreas-aristidou-phd/
https://www.researchgate.net/figure/Architecture-of-normal-residual-block-a-and-pre-activation-residual-block-b_fig2_337691625
https://www.researchgate.net/figure/a-Resnet-architecture-b-MobileNet-V2-architecture-c-DenseNet-architecture-d_fig1_328652206
https://www.researchgate.net/publication/342400905_Real-Time_Assembly_Operation_Recognition_with_Fog_Computing_and_Transfer_Learning_for_Human-Centered_Intelligent_Manufacturing
https://www.linkedin.com/pulse/transfer-learning-fine-tuning-pre-trained-deep-models-g-her-w-adhane/
https://medium.com/@danushidk507/max-pooling-ef545993b6e4
https://datascientest.com/reseaux-de-neurones-densenet
https://www.analyticsvidhya.com/blog/2022/01/convolutional-neural-network-an-overview/
https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall
https://towardsdatascience.com/https-medium-com-dashingaditya-rakhecha-understanding-learning-rate-dd5da26bb6de
https://spotintelligence.com/2024/02/19/learning-rate-machine-learning/
https://cyber.gouv.fr/sites/default/files/document/Recommandations_de_s%C3%A9curit%C3%A9_pour_un_syst%C3%A8me_d_IA_g%C3%A9n%C3%A9rative.pdf


NOKIA

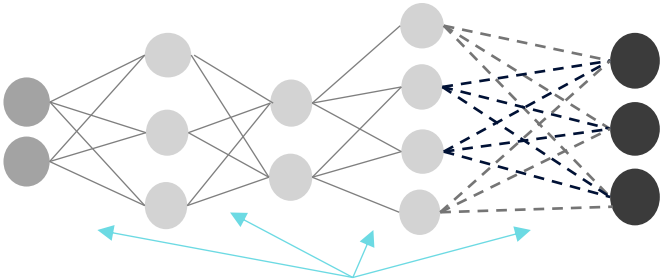
Training / inference

Les CNN font parties des techniques d'apprentissage supervisé

forward
backward

Training

Matrice de pixel +
label = 'chat'



Prédiction
du NN

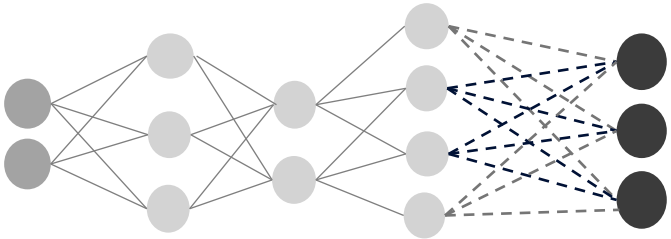
Calcul de la distance entre
la prédiction et le label

Prise en compte de la
distance pour calculer les
nouveaux poids

Mise a jour
des poids

Inférence

Matrice de pixel



Prediction
du NN

Classe = chat

Metrics

Estimation de la performance d'un modèle

Accuracy (ou precision) : pourcentage de predictions correctes pour un modèle par rapport au nombre total de prédictions

$$\text{Accuracy} = \frac{\text{Correct predictions}}{\text{All predictions}}$$

Loss function (ou fonction de cout) : function qui évalue la difference entre la prediction et la Valeur cible réelle (label)

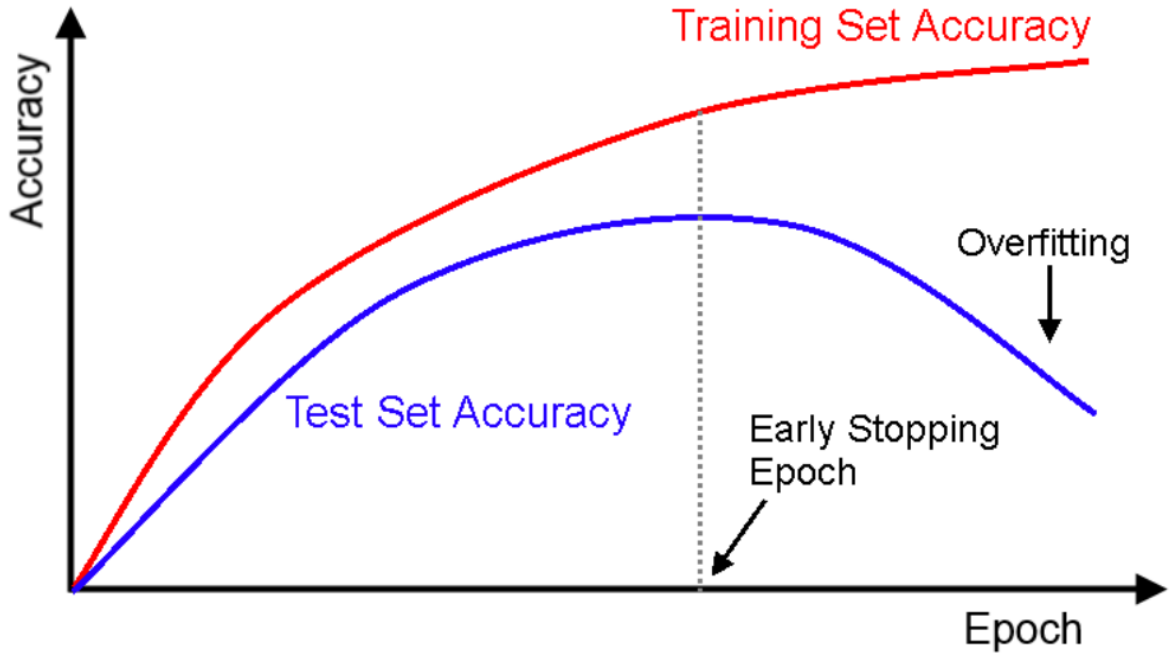
matrice de confusion : pour les classifications. Permet de comparer les classes prédites par le modéles aux classes réelles

	Prediction Chat	Prediction Pas chat
Label chat	15	4
Label pas chat	2	60

Courbe de l'accuracy

Comment voir l'overfitting

- Accuracy Curve



Learning rate

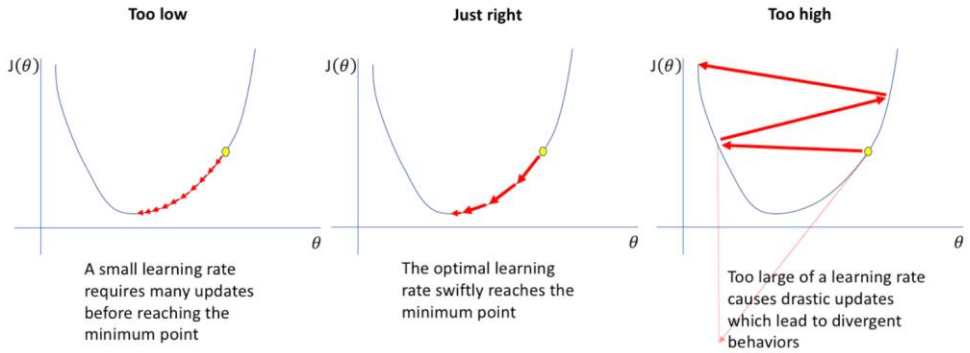
Taux d'apprentissage

Un learning rate adapté permet de converger efficacement vers le minimum global et permet d'éviter de tomber dans un minimum local

- un hyperparamètre
- Il contrôle la taille des pas effectués par l'algorithme d'optimisation qui mets à jour les poids et biais

Il a un impact sur :

- La vitesse d'apprentissage
- La précision de la convergence

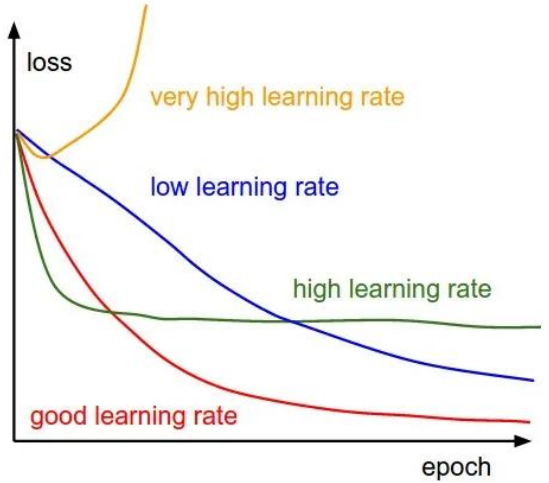


Learning rate

Impact sur l’apprentissage

•**Learning rate élevé (pas élevé)**:
•accélère l’entraitenement, mais peut provoquer des oscillations ou un échec de convergence.
•Les mises à jour peuvent devenir instables, et l'algorithme risque de diverger ou de ne jamais atteindre un minimum.

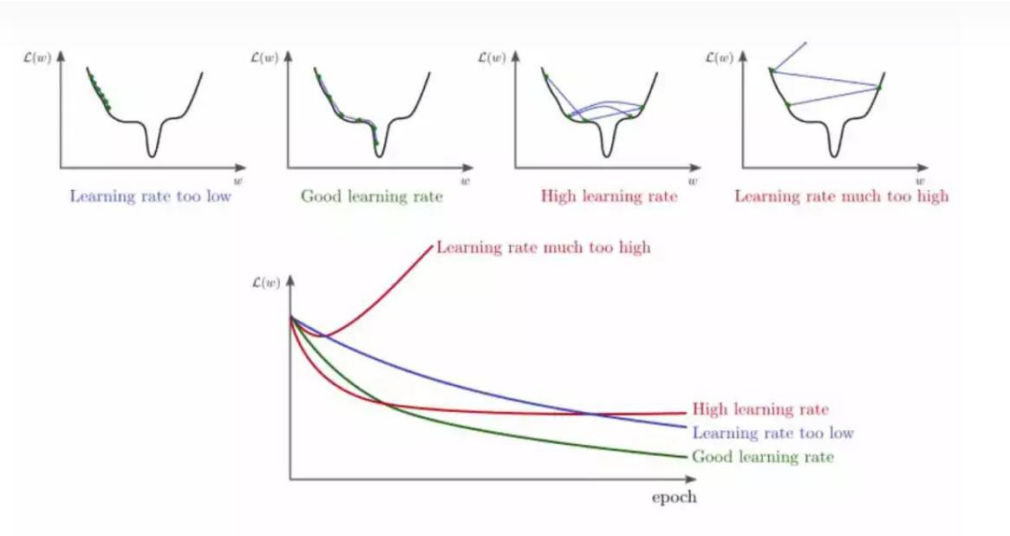
•**Learning petit (pas faible)** :
• L'entraînement devient lent, et le modèle peut rester bloqué dans un minimum local.
•.



Effect of various learning rates on convergence (Img Credit: cs231n)

Learning rate

Min local et global de la function de cout



optimizer

Rôle :
Gèrer la mise à jour des paramètres (poids et biais) d'un modèle pendant l'entraînement

But :
Minimiser la fonction de coût

Utilisation :
Il met en œuvre un algorithme d'optimisation (descente de gradient et ses différentes version améliorées)
Il utilise le learning rate

$$w_{\text{nouveau}} = w_{\text{ancien}} - \eta \cdot \nabla J(w)$$

Où :

- η : le **learning rate** (taux d'apprentissage),
- $\nabla J(w)$: le gradient de la fonction de coût $J(w)$ par rapport à w.

Types :
- SGD
- Adam
- Adagrad

https://pytorch.org/docs/stable/optim.html#algorithms

optimizer

Utilisation pratique pytorch

Dans la boucle des epoch :

- Réinit des gradients
  - optimizer.zero_grad()
- Calcul de la perte
  - loss = criterion(output, target)
- Rétropropagation
  - loss.backward()
- Mise a jour des poids
  - optimizer.step()

scheduler

Un bon ajustement du learning rate peut réduire le nombre d'itérations nécessaires pour atteindre un modèle performant.

Rôle :
Ajuster le learning rate durant le training

But :
Couplé à un optimizer permet **d'améliorer la convergence, la stabilité** et les perfs du modèle.

Utilisation :
Modification du learning rate durant le training
Peut diminuer le learning rate progressivement pour éviter de sauter au-dessus du min global vers la fin du training
Peut augmenter le learning rate pour éviter de tomber dans un minimum local

Types :
- stepLR
- multiStepLR

https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate

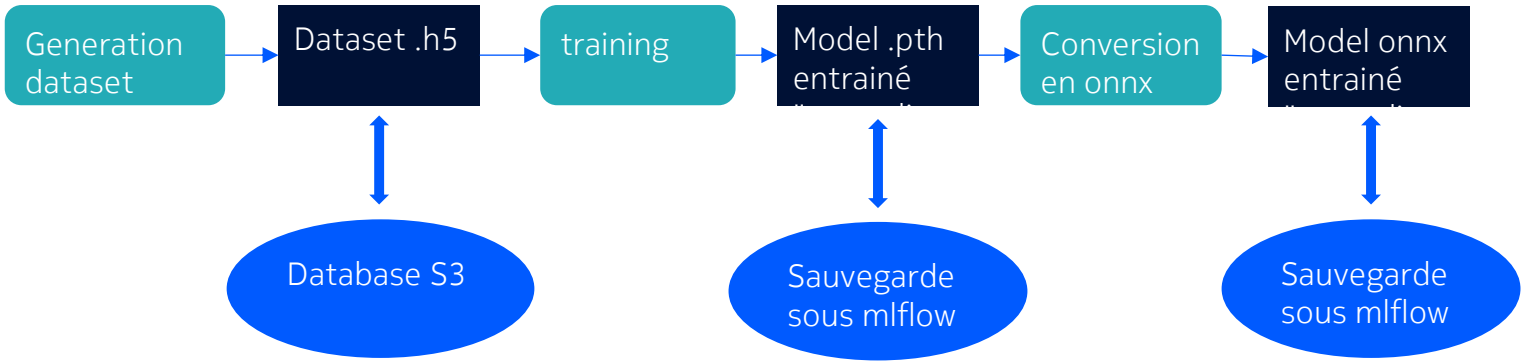
Workflow de training

Exemple d'un workflow de training

BUT = avoir en sortie un modele entrainé pret a aller en inference (production) et echangeable facilement donc au format onnx

En entrée on a les uses cases choisis pour l'entrainement
En sortie on a un model entrainé pret a etre envoyé en inference

On utilise le format .h5 pour les dataset
On utilise le standard onnx (permet d'echanger des CNN entre differentes appli qui n'utilisent pas forcement le meme framework pour les NN)



Performances lors de l’infèrence : optimisation

Utilisation mémoire

Les poids et l’activation sont stocké en mémoire en 8-bits au lieu de 32-bits => gain de place

Consommation electrique

Réduction de la consummation élec a la fois pour l’accès mémoire et pour les calculs

Latence

Les calculs et les accès mémoire sont plus simple en 8-bits qu’en 32-bits => gain en latence

Surface de Silicium sur le composant

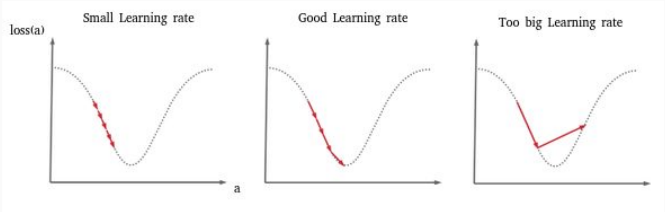
Utilisation de moins de silicium sur la puce pour faire des calculs en int8 qu’en float => on gagne de la place sur le composant => moins de dissipation de chaleur

Learning rate

Le learning rate est un des hyperparamètres les plus importants à régler, car il a un impact direct sur la vitesse et la stabilité de l'apprentissage, ainsi que sur la qualité des résultats finaux.

Choice of Learning rate

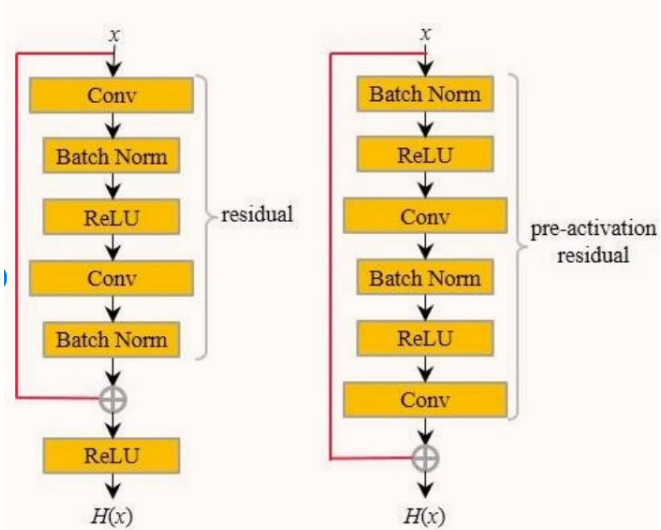
This example actually illustrates an extreme case that can occur when the Learning rate is too high. During gradient descent, between two steps we then skip the minimum and sometimes we can even totally diverge from the result to arrive at something totally false. The diagram below (especially the 3rd image) illustrates this phenomenon:



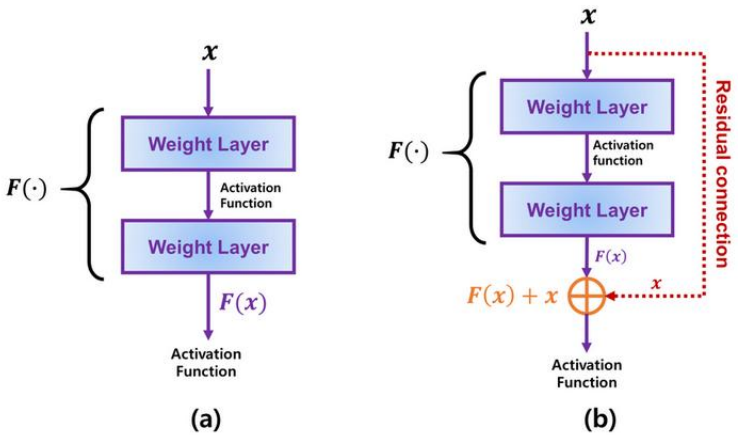
So we chose a learning rate that was too high.

Learning rate (λ) is one such **hyper-parameter** that defines the **adjustment in the weights of our network with respect to the loss gradient descent**. It determines how fast or slow we will move towards the optimal weights
The Gradient Descent Algorithm estimates the weights of the model in many iterations by minimizing a cost function at every step.

RESNET pattern



Architecture of normal residual block (a) and pre-activation residual block (b).



The comparison between the general connections and the residual connections used in ResNet. (a) describes how the input data x is passed into the weight layers in general CNN models with no residual connections. (b) shows how x is passed to both the weight layers and the residual connections in ResNet.