SAISON 24/25

Formation

Introduction au Deep Learning

Séquence n°11

“Méthodologie et optimisation de l'apprentissage”

FIDLE

cnrs UGA Université Grenoble Alpes MIAI Grenoble Alpes EFELIA

GENCI ANITI DEVLOG RÉSEAU Resinfo cnrs IDRIS SIMaP

Listes de diffusion - Newsletters

FIDLE https://fidle.cnrs.fr/listeinfo

Fidle information list

Agoria http://fidle.cnrs.fr/agoria

AI exchange list

DEVLOG RÉSEAU DU DEVELOPPEMENT LOGICIEL

https://listes.services.cnrs.fr/wws/info/devlog

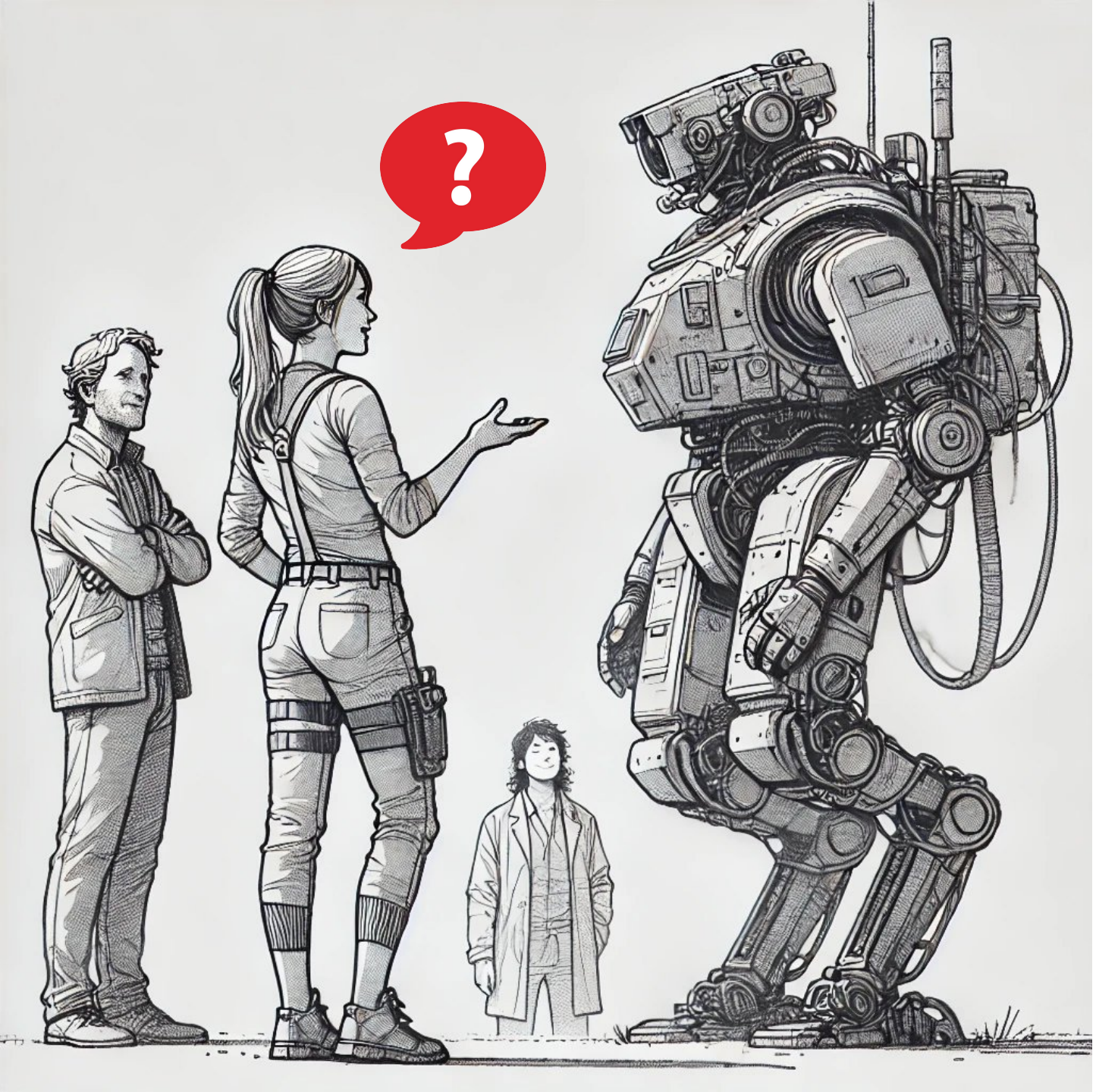
List of ESR* « Software developers » group

GROUPECALCUL

https://listes.math.cnrs.fr/wws/info/calcul

List of ESR* « Calcul » group

(*) ESR is Enseignement Supérieur et Recherche, french universities and public academic research organizations



FIDLE

Pour toutes vos **questions**, n'hésitez pas à utiliser :

- le **chat** pendant le live
- la **liste AGORiA**

Suivez-nous !

FIDLE

youtube.com/@CNRS-FIDLE

linkedin.com/company/fidle-cnrs

twitch.tv/formationfidle_cnrs

FIDLE

Panorama IA

Le magazine de actualité en IA

Vendredi 28 mars, 10h00

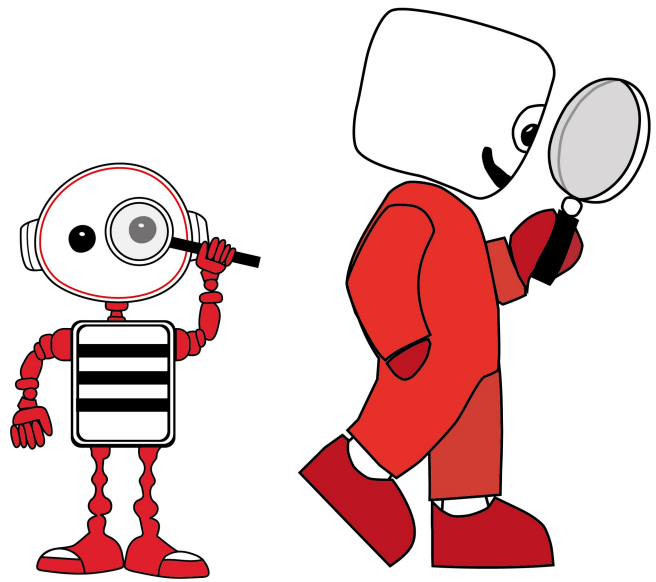
http://www.idris.fr/panoramia.html
https://www.youtube.com/@IDRISCNRS

3e Journée

Deep learning pour la Science

FIDLE

cnrs IDRIS



Jeudi 5 Juin 2025
au CNRS à Paris

3 rue Michel-Ange, Paris 16

Gratuit sur inscription :
jdls-2025.sciencesconf.org

cnrs

Cluster IA
Multidisciplinary Institute In Artificial Intelligence

EFELIA

ANITI

DEVLOG RÉSEAU du développement logiciel

UGA Université Grenoble Alpes

GENCI

SIMaP Science et Ingénierie des Matériaux et des Procédés

FIDLE *PRÉSENTE*

PROCÈS DE L'IA

SOYEZ JURÉ.E DANS LE PROCÈS DE L'ANNÉE !

MIAI Cluster IA
Multidisciplinary Institute In Artificial Intelligence

UGA Université Grenoble Alpes

cnrs

JEUDI 10 AVRIL - 18H

MAISON DU TOURISME
GRENOBLE 14 RUE DE LA RÉPUBLIQUE

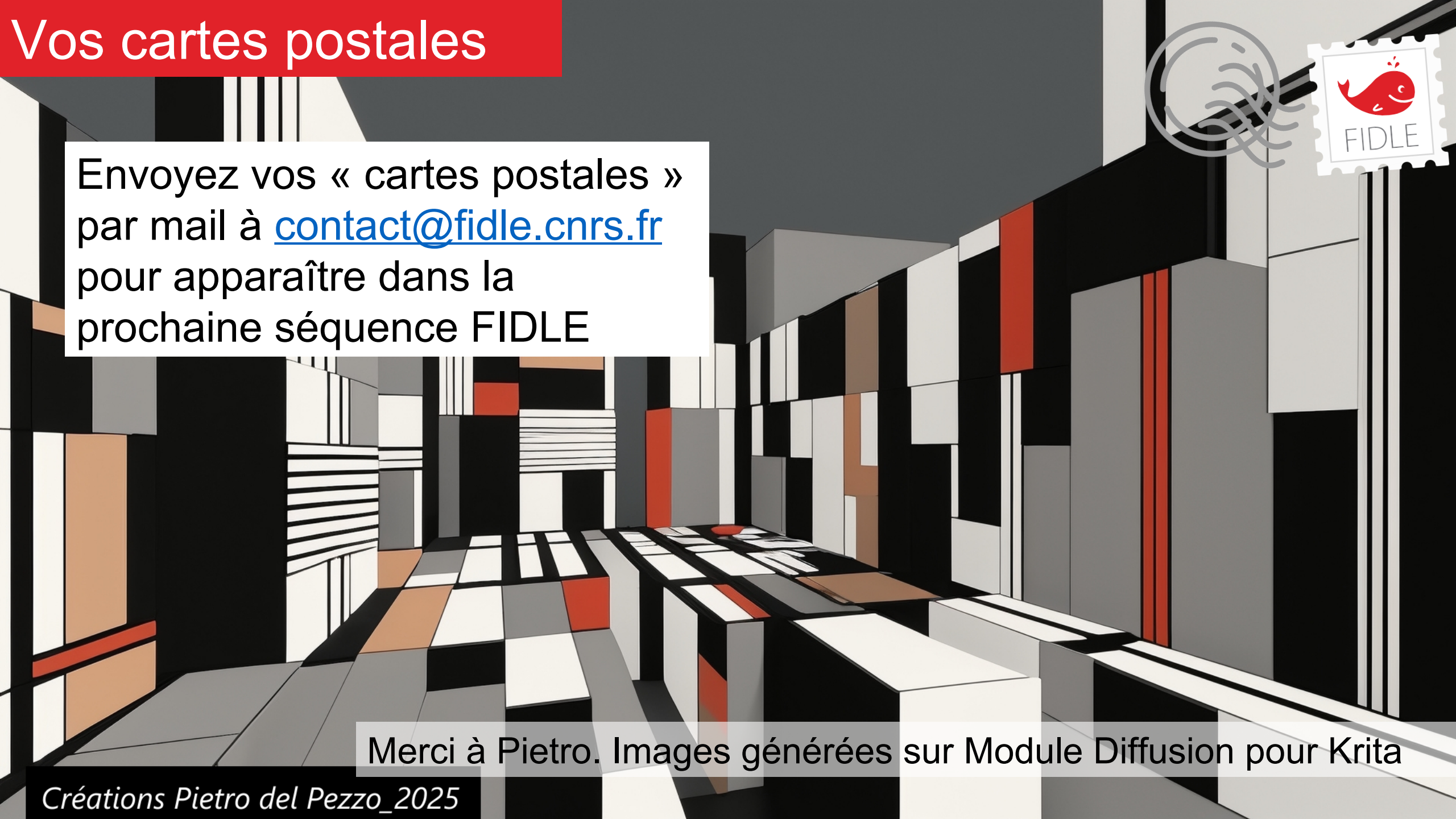
Vos cartes postales

Envoyez vos « cartes postales »
par mail à contact@fidle.cnrs.fr
pour apparaître dans la
prochaine séquence FIDLE

Merci à Franck depuis l'observatoire du Pic du Midi

Vos cartes postales

Envoyez vos « cartes postales »
par mail à contact@fidle.cnrs.fr
pour apparaître dans la
prochaine séquence FIDLE



Merci à Pietro. Images générées sur Module Diffusion pour Krita

Créations Pietro del Pezzo_2025

Vos cartes postales

Envoyez vos « cartes postales »
par mail à contact@fidle.cnrs.fr
pour apparaître dans la
prochaine séquence FIDLE

Merci à Pietro. Images générées sur Module Diffusion pour Krita

Créations Pietro del Pezzo_2025

FIDLE

SAISON 24/25

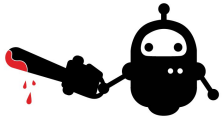
Bases, Concepts et Enjeux

1 ✓

Bases, concepts et histoire...
L'IA, c'est quoi ?

2 ✓

L'IA dans la pratique :-)
Exemples et démos

3 ✓ New !

L'IA est-elle notre amie ?
Le grand procès de l'IA

4 ✓

IA, droit, société et éthique
Vivre avec l'IA

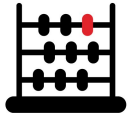
L'IA comme un outil,

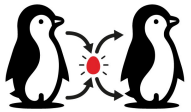
5 ✓

Neural Networks from Zero to Hero
DNN

6 ✓

Convolutional models
CNN

7 ✓

Mathematics, gradients everywhere !

8 ✓

Encoder/Decoder networks
AE/VAE, etc.

9 ✓ New !

Data, embedding and latent spaces
Representing the world

10 ✓

Graph Neural Network
GNN

11

Learning optimisation
Parameters & metrics

12

«Attention is All You Need»
RNN, Transformers

13 New !

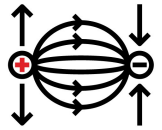
Models that talks !
LLM

14

Models who draw !
Diffusion models

15

Multimodal models
Multimodal LLM

16

Physics-Informed Neural Networks
PINNS

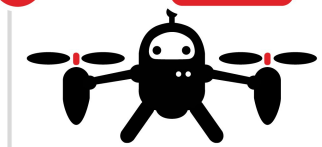
Acteur de l'IA

17 New !

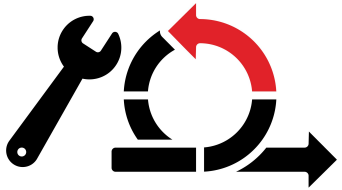
Deep Failed !
The Great AI Blooper Reel

18

Learning optimisation II
Advanced

19 New !

Edge AI / IoT and Inference
Production

11

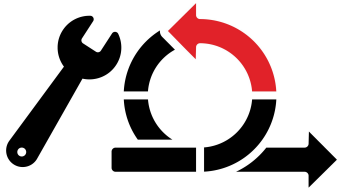


Learning optimisation

Parameters & metrics

1. Data Quality
   Why is is so important ?
2. Loss functions and evaluation metrics
3. Learning optimization

11



Learning optimisation

Parameters & metrics

1 **Data Quality**
Why is is so important ?

2 Loss functions and evaluation metrics

3 Learning optimization

Data quality

Training data → Machine learning model → Predictions

Garbage in

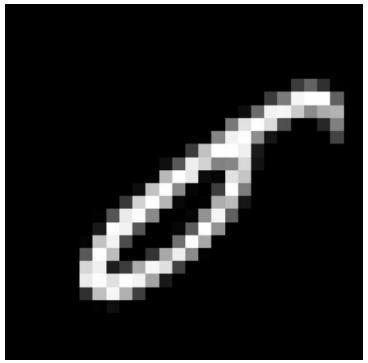
Garbage out

Main data quality dimensions :

1. Consistent representation
2. Completeness
3. Uniqueness
4. Target class balance

These data quality dimensions need to be checked during exploratory data analysis (EDA)

Data quality is an underrated way to improve the quality of the model



Data quality issues even in well known datasets (MNIST index 6651 test set)

Image credits : Supanut Piyakanont (garbage), Smalllike (garbage truck)

Tool : Label Errors in ML Test Sets

Paper : The Effects of Data Quality on Machine Learning Performance on Tabular Data, SEDIR MOHAMMED, LUKAS BUDACH, MORITZ FEUERPFEIL, NINA IHDE, ANDREA NATHANSEN, NELE NOACK, HENDRIK PATZLAFF, FELIX NAUMANN, HAZAR HARMOUCH, 2024

Data quality : consistent representation

Detection of inconsistencies

For tabular data :
Check mixed data types, outliers, multiple representations of the same category, inconsistent units …

For image :
Check image sizes, channels, formats…

For text :
Check encodings, tokenization methods…

	Age	Department	Salary	Location
0	25	HR	50000	Paris
1	Thirty	Human Resources	55000	PA
2	40	hr	Thirty Trillion	New York
3	50	human resources	65000 $	Los Angeles
4	60	HUMAN RESOURCES	70000 €	NYC

pandas Useful methods :

```
df.describe()
df.info()
df.dtypes
```

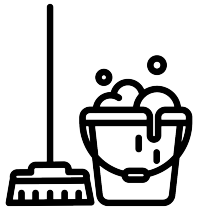
```
df[col].value_counts()
df[col].nunique()
df[col].unique()
df[col].str.extract(r'([\d.]+)')
```

Image credits : Imam Kurniadi (detection)

Code libraries : pandas, numpy, skrub, cleanlab

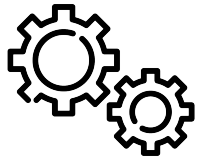
Paper : Data Smells in Public Datasets, Arumoy Shome, Luis Cruz, Arie van Deursen, 2022

Data quality : consistent representation



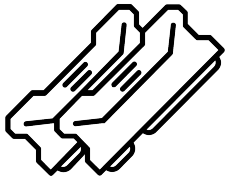
Correction of inconsistencies

Importance of using the right data type :

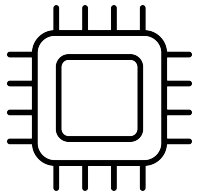


Access to specialized operations

```
df['year'] = df['date'].dt.year
df['weekday'] = df['date'].dt.weekday
df['formatted_date'] = df['date'].dt.strftime('%d-%m-%Y')
```



Memory efficiency



Speed of computation

pandas

Useful methods :

```
df[col] = pd.to_datetime(df[col])
df[col] = pd.Categorical(df[col], categories=['A', 'B', 'C'])
df[col] = df[col].astype('category')
df[col] = df[col].astype('int32')
```

```
replacements = {
    'N/A': 'unknown',
    'NA': 'unknown',
}
df[col] = df[col].replace(replacements)

df = pd.DataFrame({'courses': ['Fidle N°1', 'Fidle N° 2', 'Fidle 4']})
df['courses'] = df['courses'].replace(r'\D', '', regex=True).astype(int)
```

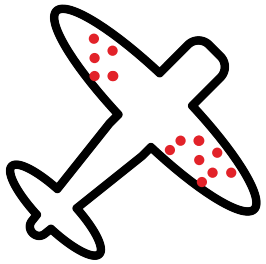
```
df['name'] = df['name'].str.lower()
df['email'] = df['email'].str.replace('@fidle.com', '@fidle.fr')
df[['username', 'domain']] = df['email'].str.split('@', expand=True)
```

pandas

Using the **category** data type instead of **object** for categorical features reduces memory usage, accelerates operations and enhances data consistency

Data quality : completeness

Detection of missing values



Survival biais

	MCAR	MNAR	MAR	
	Exam Score (%)	Time for the Exam (min)	ChatGPT Usage (hrs)	Preferred Study Method
0	85	NaN	5.0	Books
1	78	100.0	2.0	Online
2	72	110.0	3.0	NaN
3	85	90.0	NaN	Books
4	90	120.0	7.0	Videos
5	60	85.0	NaN	NaN
6	50	120.0	6.0	NaN
7	78	95.0	NaN	Books
8	48	115.0	0.0	NaN
9	75	NaN	1.0	Online

Image credits : Imam Kurniadi (detection), zaenul yahya (plane)

Missing Completely At Random (MCAR) :
means missing values occur randomly with no relationship to other variables

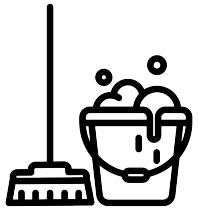
Missing At Random (MAR)
missing values depend on other observed variables but not on the missing data itself

Missing Not At Random (MNAR)
missing values depend on the value itself

Visualization of missing values with the missingno library

Code libraries : Missingno

Data quality : completeness



Correction of missing values

Removing missing values [MCAR] :
Rows or columns

pandas

```
df.dropna(axis=0) # Rows
df.dropna(axis=1, thresh=int(0.5 * len(df))) # Columns
```

Impute missing values (simple methods) [MCAR] :
Central statistical values (median, mean, mode…)

scikit learn

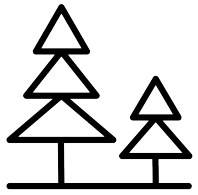
```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='mean')
data_imputed = imputer.fit_transform(data_array)
```

Filling missing values [MCAR] :
Specific values

pandas

```
df.fillna(0)
df['A'] = df['A'].fillna(df['A'].mean())
```

Tree-based models don't require missing values to be corrected



Impute missing values (complex methods) [MAR, NMAR] :
ML models (KNN Imputer, Random Forest, MICE…)

scikit learn

```
from sklearn.impute import KNNImputer
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from sklearn.ensemble import RandomForestRegressor
```

Code libraries : pandas dropna, scikit-learn simple imputer

Image credits : Kiran Shastry (housekeeping), Firda Wahyu Dianti (tree)

Data quality : uniqueness

Detection of duplicated values

```python
import pandas as pd
from difflib import SequenceMatcher

data = {
    "Courses": ["Fidle", "fidle", "Fadol", "Fiddle", "Fedal", "Fodel"],
}
df = pd.DataFrame(data)
def similarity_ratio(a, b):
    return SequenceMatcher(None, a, b).ratio()
```

Tabular data

```python
duplicates = df[df.duplicated(keep='first')]
duplicates_subset = df[df.duplicated(subset=['col_1', 'col_2'])]
```

```
Near-duplicate found:
Fidle ↔ fidle (Similarity: 0.80)
Fidle ↔ Fiddle (Similarity: 0.91)
fidle ↔ Fiddle (Similarity: 0.73)
```

Text data

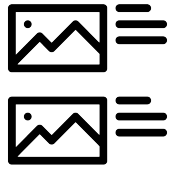
Levenshtein distance

Jaccard similarity with n-grams

Phonetic matching

Cosine similarity with TF-IDF

Cosine similarity with BERT



It is also possible to detect **near duplicated images** with perceptual hashes or deep learning techniques

Image credits : Imam Kurniadi (detection), Setyo Ari Wibowo (images)

Typos

Near-duplicate words

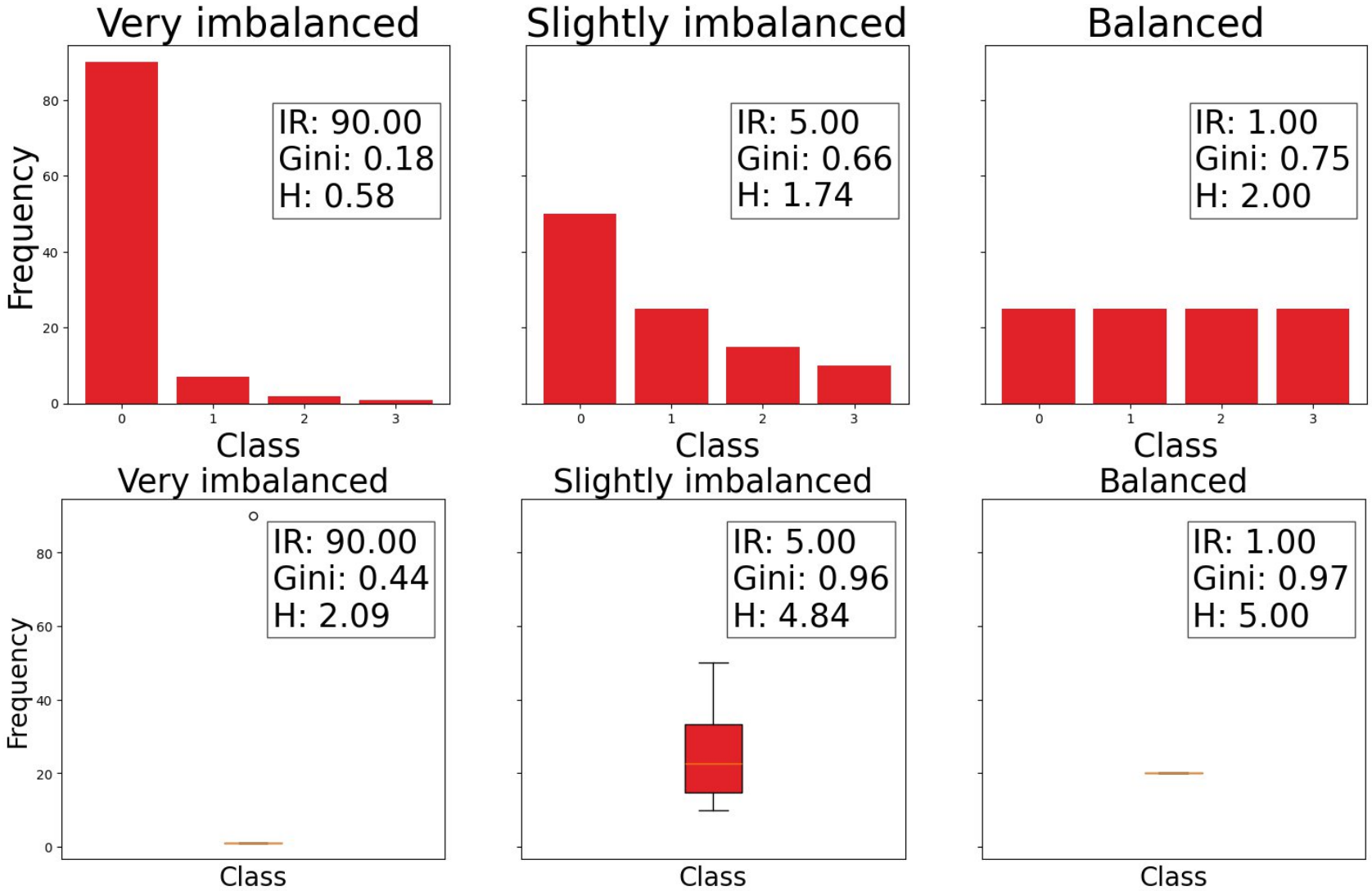
Phrase similarity

Semantic similarity

Code libraries : pandas, difflib (Python standard library), rapidfuzz

Data quality : target class balance

Detection of imbalanced dataset



$$Imbalance\ Ratio = \frac{\max_i N_i}{\min_i N_i}$$

$$Gini = 1 - \sum_{i=1}^{n} p_i^2$$

$$Entropy = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

To ensure comparability across datasets, **normalized metrics** are required

Image credits : Imam Kurniadi (detection)

Data quality : target class balance

Correction of imbalanced dataset

Class weighting :
Class weights in loss function

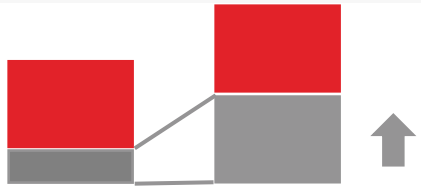
```
from sklearn.utils.class_weight import compute_class_weight
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(class_weight='balanced', random_state=42)
```

```
model.fit(X_train, y_train, epochs=10, class_weight=class_weights_dict)
```

Resampling techniques :
Undersampling, oversampling

```
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
```

Undersampling



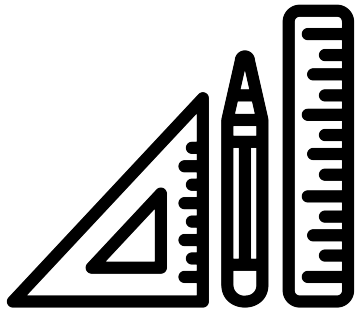
Oversampling

Anomaly detection :
Isolation forest, autoencoder…

Generating synthetic data :
SMOTE, GAN, autoencoder, data augmentation…

Image credits : Kiran Shastry (housekeeping)

Tool : data quality



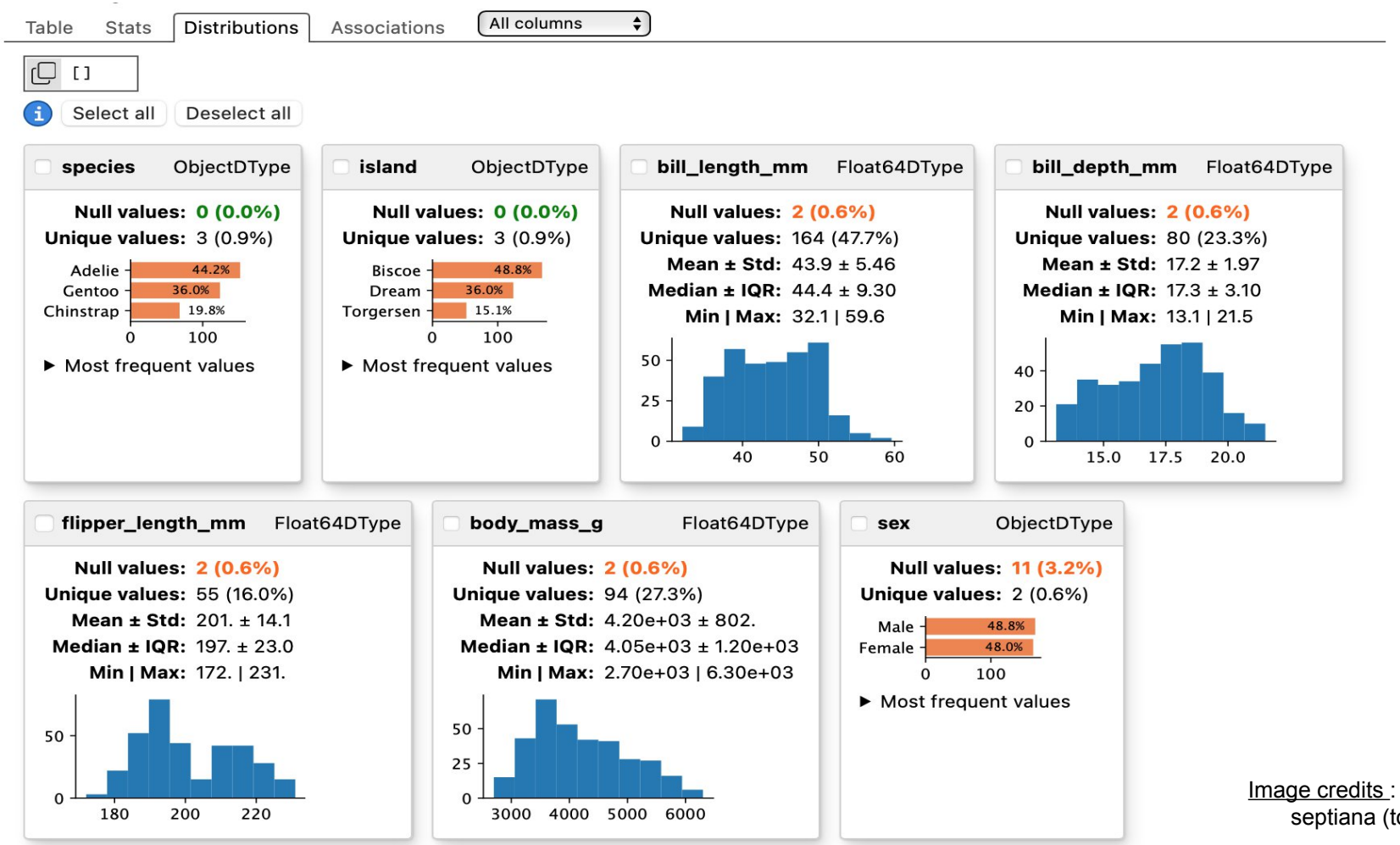


Image credits : metami septiana (tool)

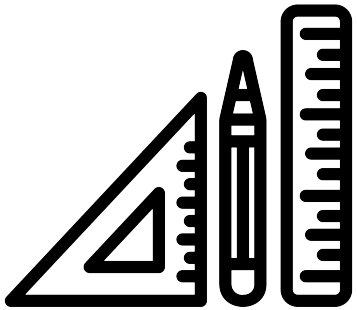
Ease preprocessing and feature engineering for tabular data

Interactive data exploration

GapEncoder to encode categorical data with typos or variations

…

Tool : data quality



UNION PANDERA

Python open source library

```python
import pandas as pd
import pandera as pa

schema = pa.DataFrameSchema({
    "student_id": pa.Column(int, checks=[pa.Check.ge(1)], nullable=False),
    "mathematics": pa.Column(float, checks=[pa.Check.ge(0), pa.Check.le(20)]),
})

data = pd.DataFrame({
    "student_id": [101, 102, 103, 104],
    "mathematics": [15.5, 18.0, 12.0, 24.0],
})

try:
    validated_data = schema.validate(data)
    print("All data is valid")
except pa.errors.SchemaError as e:
    print("Data validation error\n", e)
```

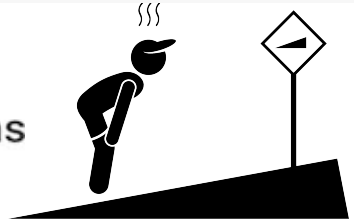
Works with pandas, dask and pyspark

Prevents silent errors

Improves data integrity

Automates data validation

great expectations

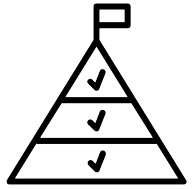


The **Great Expectations** library meets all data quality needs but it has a steeper learning curve

Image credits : metami septiana (tool), Gan Khoon Lay (running uphill)

Paper : A Survey on Data Quality Dimensions and Tools for Machine Learning, Yuhan Zhou, Fengjiao Tu, Kewei Sha, Junhua Ding, Haihua Chen, 2024

Data quality : conclusion



A time-consuming but one of the most essential step

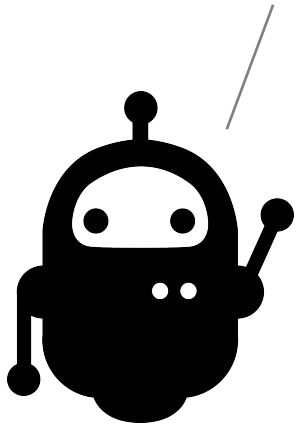
High data quality at the point of creation by enforcing constraints and validation rules upfront

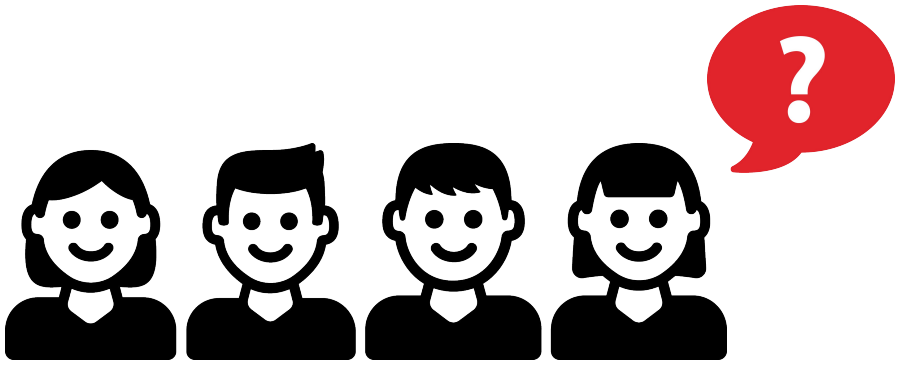
Data quality assessment during exploratory data analysis

Integrating data quality metrics into pipelines

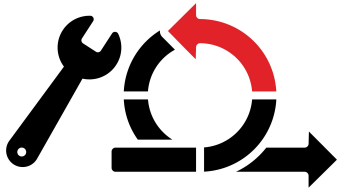
Image credits : Creative Mahira (morning), Andika Cahya Fitriani (assessment), jumiati (dashboard), Cahya Kurniawan (pyramid)

Quelques questions ?





11



Learning optimisation

Parameters & metrics

1. Data Quality
   Why is is so important ?
2. **Loss functions and evaluation metrics**
3. Learning optimization

Loss functions

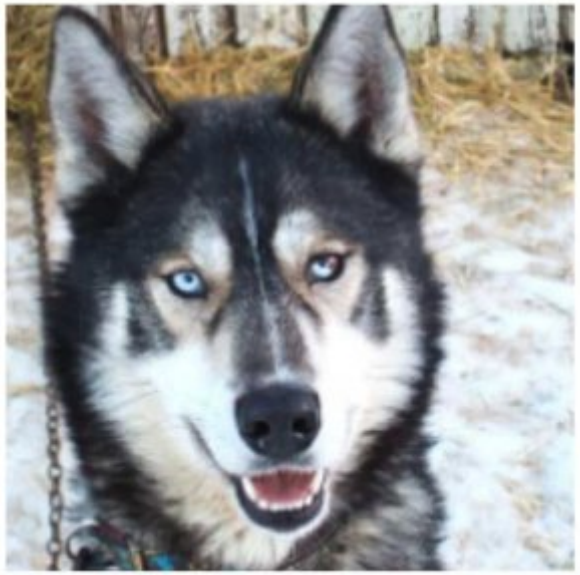
FIDLE

The origin of the alignment problem

A **loss function** quantifies the error between the **model's prediction** and the **true value**

Objectives given to AI models through loss functions are **proxies** for what we really want

« When a measure becomes a target, it ceases to be a good measure »
Goodhart's law



Models classify dogs and wolves by minimizing loss not by reasoning like humans

Book : The alignment problem, Brian Christian, 2020

Paper : Can everyday AI be ethical. Fairness of Machine Learning Algorithms, Philippe Besse, Celine Castets-Renard, Aurelien Garivier, Jean-Michel Loubes, 2018

Difference between loss function and evaluation metrics

1. *Different purposes*
   - **Loss function :** guide the optimization process
   - **Metrics** : evaluate generalisation, detect overfitting or underfitting

2. *Used at different stages*
   - **Loss function :** during training
   - **Metrics :** during and after training, in production

3. *Different constraints*
   - **Loss function :** mathematical constraints driven by optimization algorithms
   - **Metrics :** alignement with the specific goals of the task

Mean Squared Error (MSE) can be used as a loss function or as a evaluation metric

Paper : LOSS FUNCTIONS AND METRICS IN DEEP LEARNING, Juan Terven, Diana M. Cordova-Esparza, Alfonso Ramirez-Pedraza, Edgar A. Chavez-Urbiola, Julio A. Romero-Gonzalez, 2024

Loss fonctions for regression

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

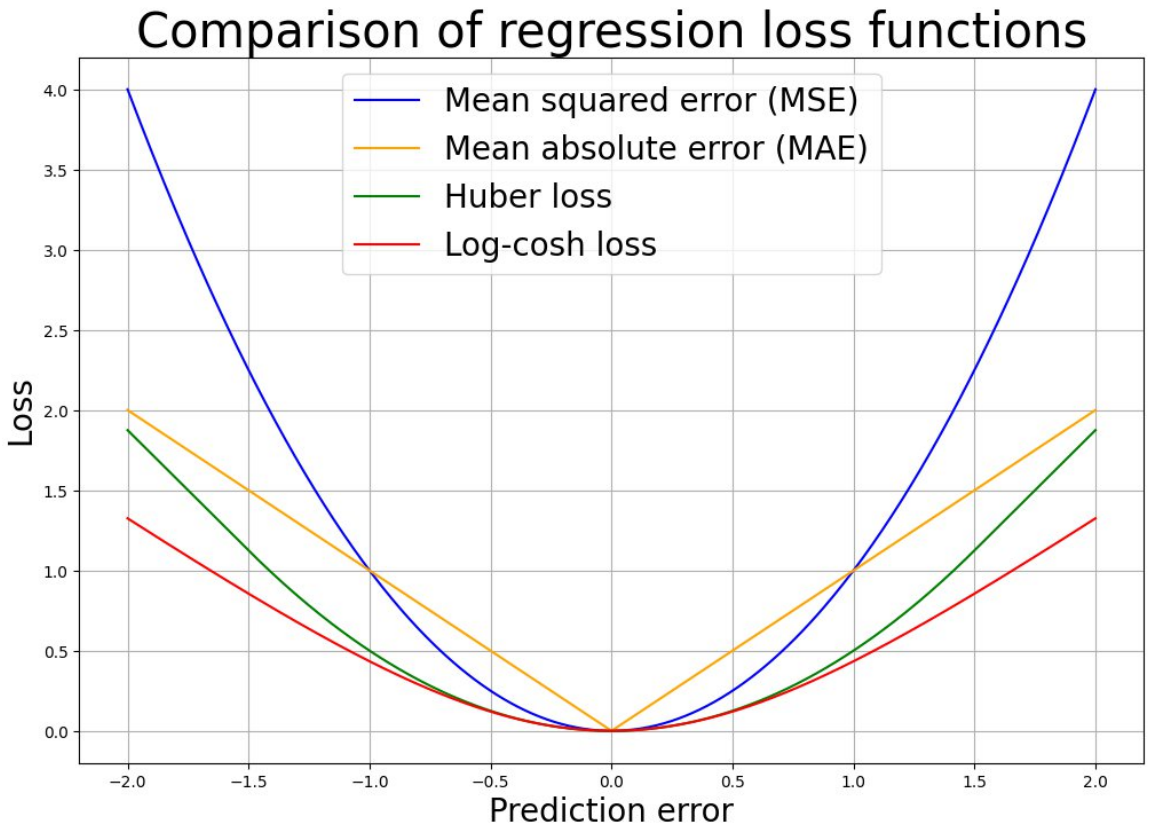
(Mean Squared Error)

1. Continuous and differentiable
2. Sensitive to outliers

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

(Mean Absolute Error)

1. Non differentiable at 0
2. Robust to outliers



Other losses : Huber loss, Log-cosh loss, Quantile loss, Poisson loss…

Code libraries : Pytorch loss functions, Keras regression loss functions

Loss fonctions for binary classification

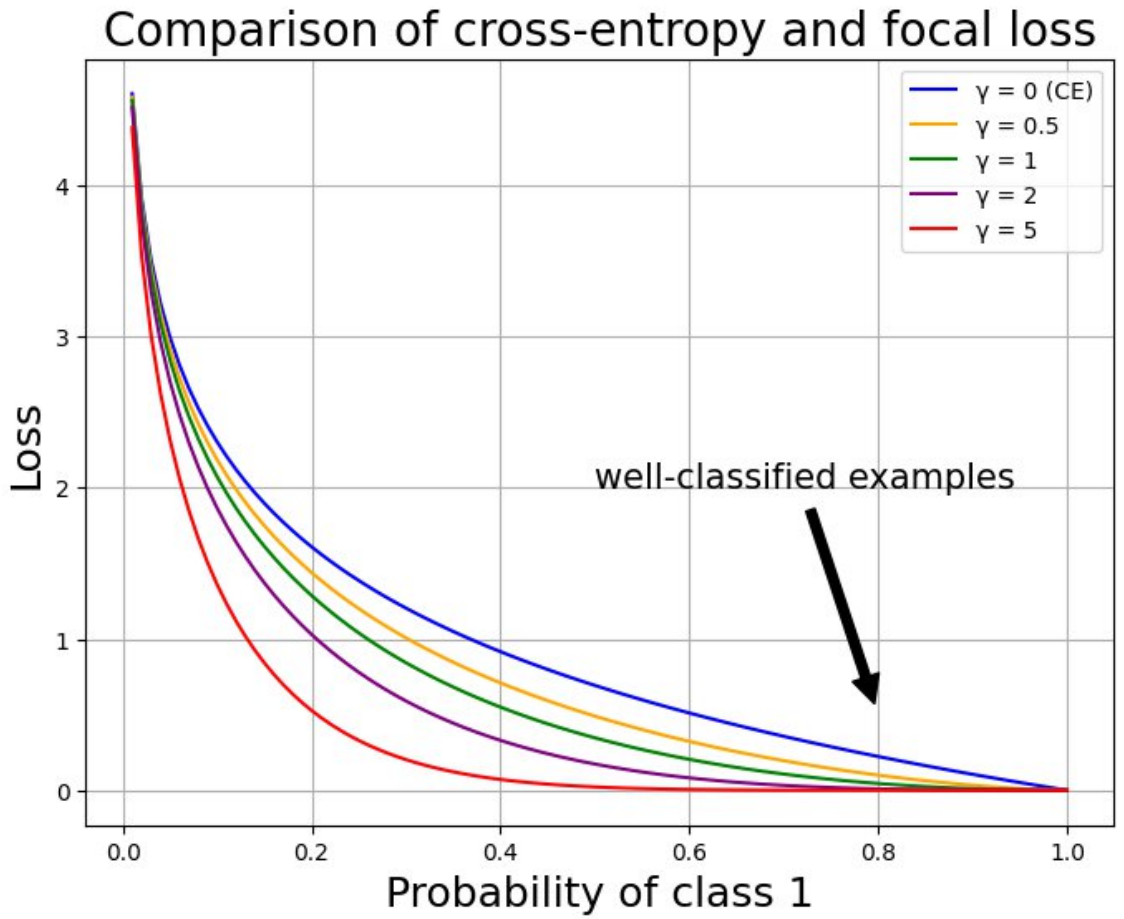
$$BCE = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i.log\left(\hat{y}_i\right) + \left(1-y_i\right).log\left(1-\hat{y}_i\right)\right)$$

(Binary cross entropy)

1. Differentiable
2. Sensitive to class imbalance

$$Focal\ loss(y_i = 1) = -\propto\left(1-\hat{y}_i\right)^{\gamma}.\log\left(\hat{y}_i\right)$$

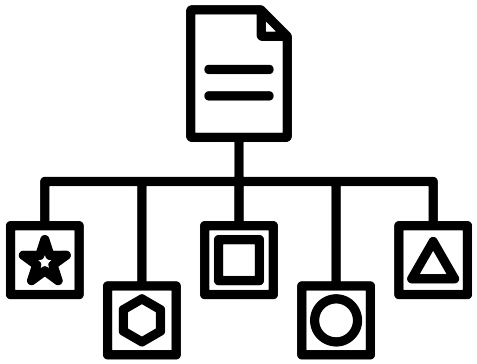
1. γ factor to reduce the loss contribution from easy examples
2. α balances the importance of 0/1 examples



Code libraries : Pytorch loss functions, Keras classification loss functions

Paper : Focal Loss for Dense Object Detection, Tsung-Yi Lin Priya Goyal Ross Girshick Kaiming He Piotr Doll´ar, 2018

Loss fonctions for multiclass classification



$$CCE = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n\_class} y_{ij}.log\left(\hat{y}_{ij}\right)$$

(Categorical Cross Entropy)

Labels

One hot encoded

[0 1 0 0]
[0 0 0 1]
[1 0 0 0]
[0 0 1 0]

↓

Categorical cross entropy

Sparse

[1]
[3]
[0]
[2]

↓

Sparse categorical cross entropy

It is also possible to use **focal loss** with multiclass classification

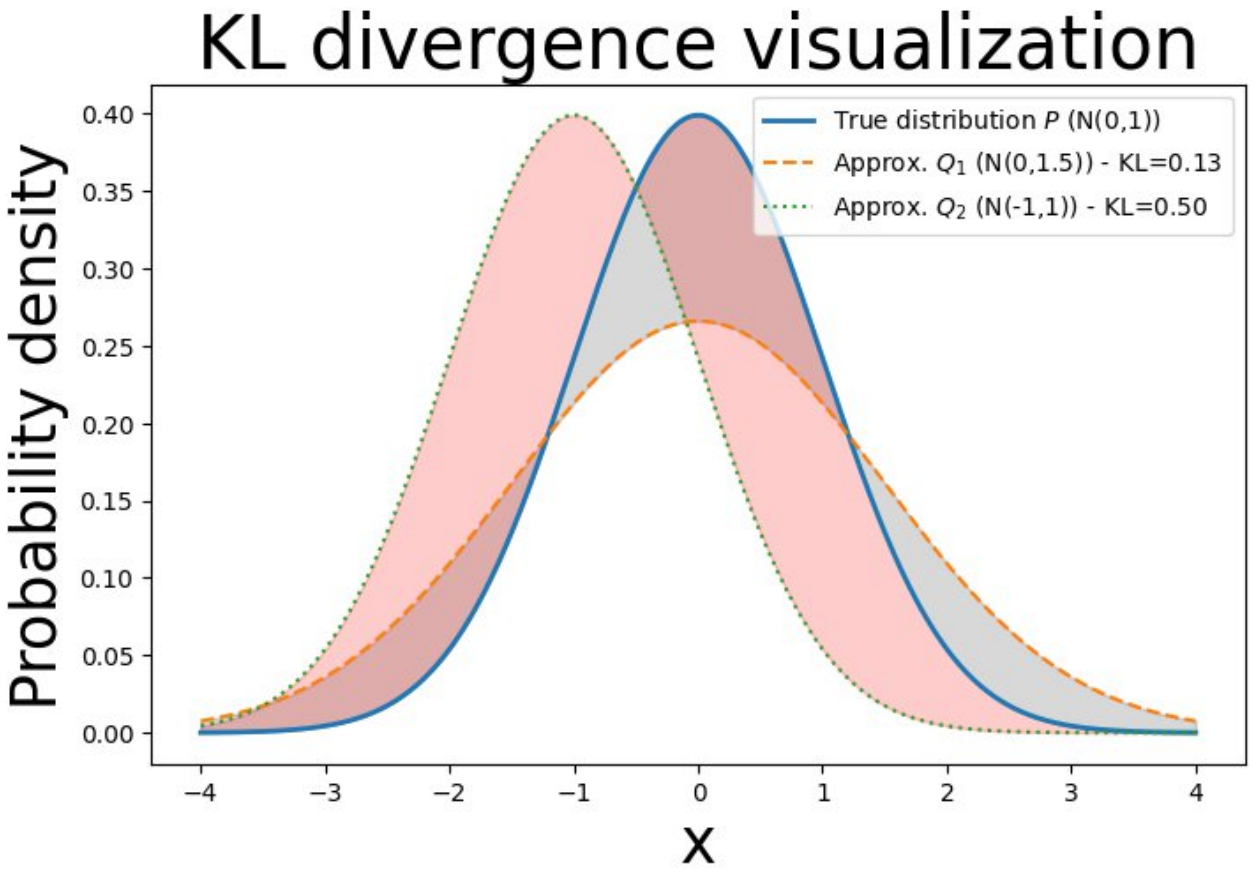
Code libraries : Pytorch loss functions, Keras classification loss functions

Image credits : Nanang A Pratama (classification)

Another useful loss : Kullback-Leibler divergence

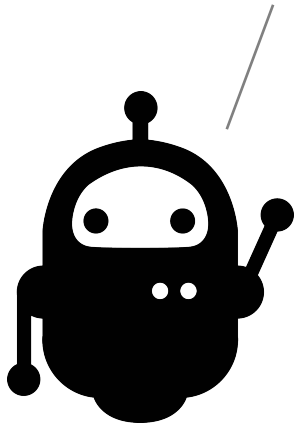
Goal :
Measure the difference between two probability distributions

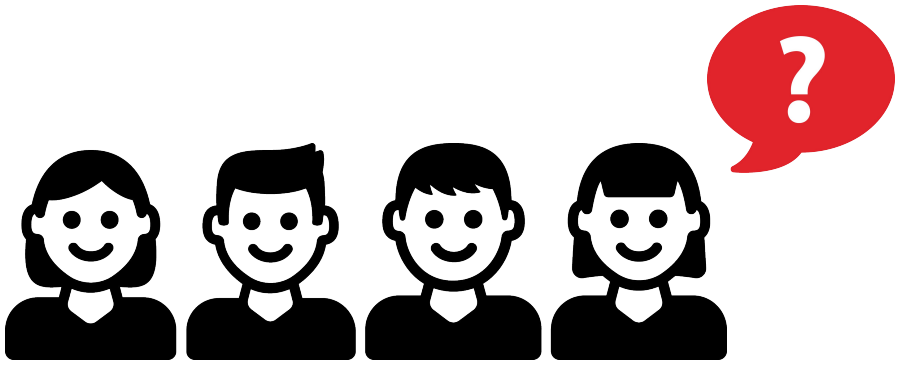
$$D_{KL} = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx$$



Use cases :
Variational autoencoders (VAE)
Knowledge distillation
Reinforcement learning
…

Quelques questions ?





Evaluation metrics

FIDLE

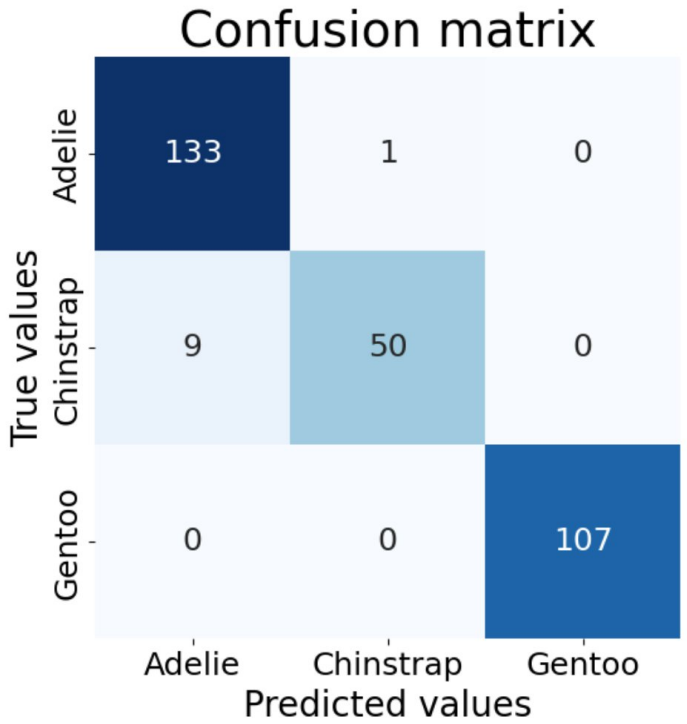
One metric is never enough

Metrics for prediction and for decision

Issue with imbalanced datasets

Global and individual view

	precision	recall	f1-score	support
Adelie	0.94	0.99	0.96	134
Chinstrap	0.98	0.85	0.91	59
Gentoo	1.00	1.00	1.00	107
accuracy			0.97	300
macro avg	0.97	0.95	0.96	300
weighted avg	0.97	0.97	0.97	300



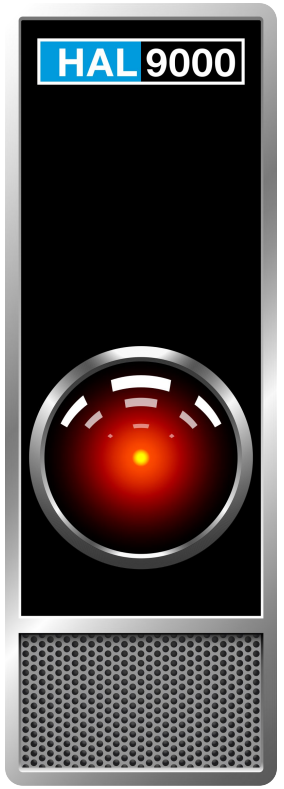


Image credits : wikipedia, Grafiker61

One metric is clearly not enough to understand a model…

Code libraries : Which scoring function should I use ?, Scikit-Learn documentation
Dataset : Palmerpenguins

Regression metrics

1. Absolute error metrics

Other metrics : sMAPE, median absolute error, max error…

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \qquad \text{MAPE} = 100\frac{1}{n}\sum_{i=1}^{n} \left|\frac{y_i - \hat{y}_i}{\max(\varepsilon, |y_i|)}\right|$$

2. Squared error metrics

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \qquad \text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

Other metrics : MSLE, RMSLE, NRMSE…

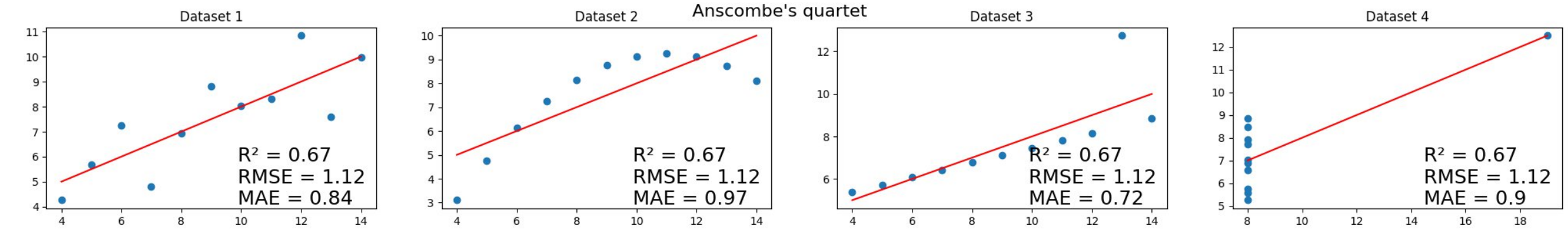
3. Model performance indicators

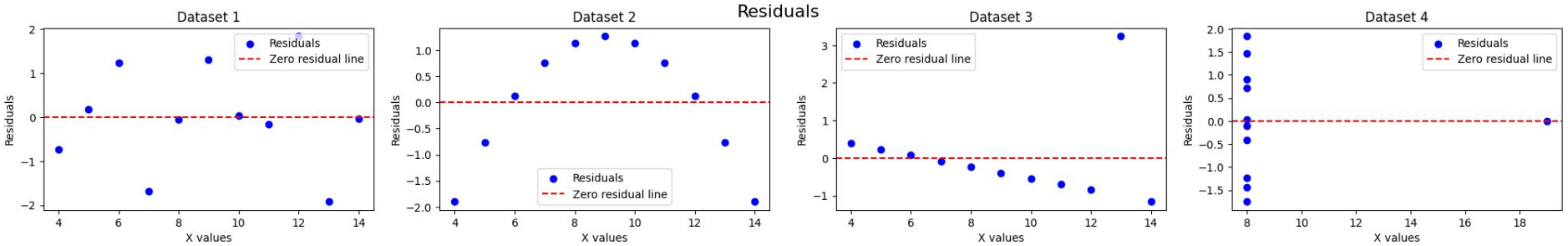
$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \qquad D^2 = 1 - \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{\sum_{i=1}^{n} |y_i - \bar{y}|}$$

Other metrics : *ajusted* R^2, D^2 pinball score, D^2 tweedie score …

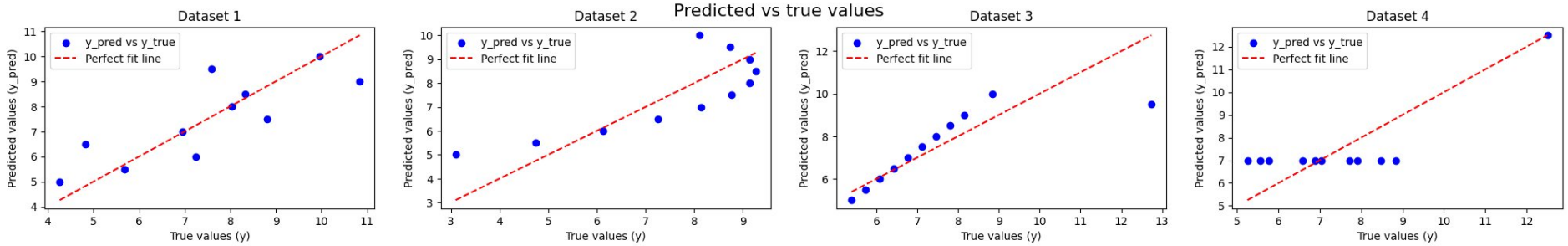
Reference : Torchmetrics, Scikit-learn

Regression metrics



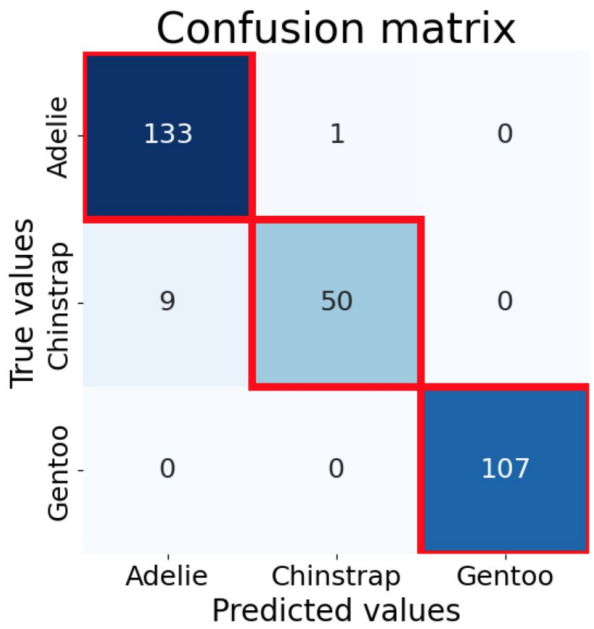


Importance of individual graphical analysis



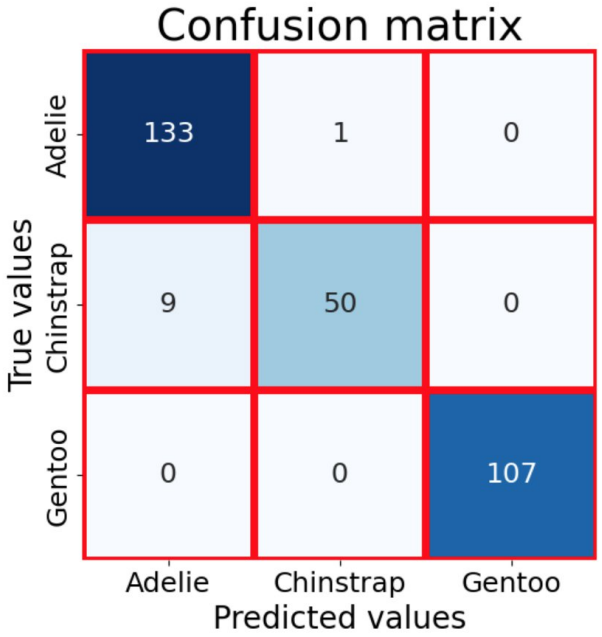
Classification metrics

Accuracy : proportion of correct predictions



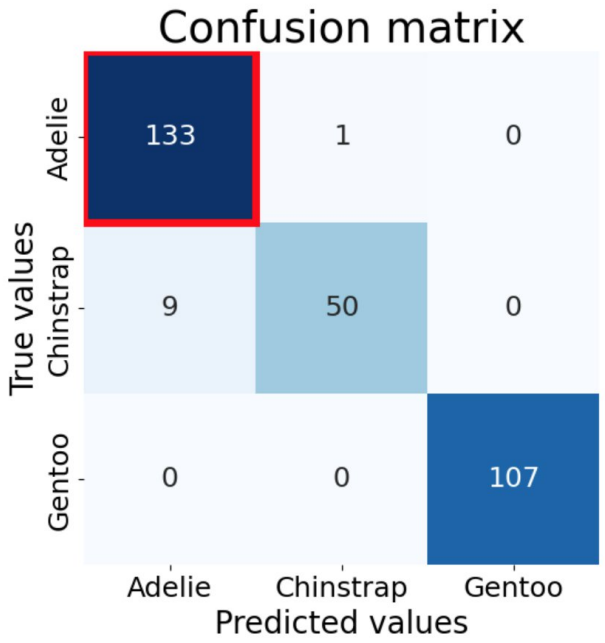
	precision	recall	f1-score	support
Adelie	0.94	0.99	0.96	134
Chinstrap	0.98	0.85	0.91	59
Gentoo	1.00	1.00	1.00	107
accuracy			0.97	300
macro avg	0.97	0.95	0.96	300
weighted avg	0.97	0.97	0.97	300

$$\text{Accuracy} = \frac{\quad}{\quad} = \frac{133 + 50 + 107}{300} = 0.97$$



Classification metrics

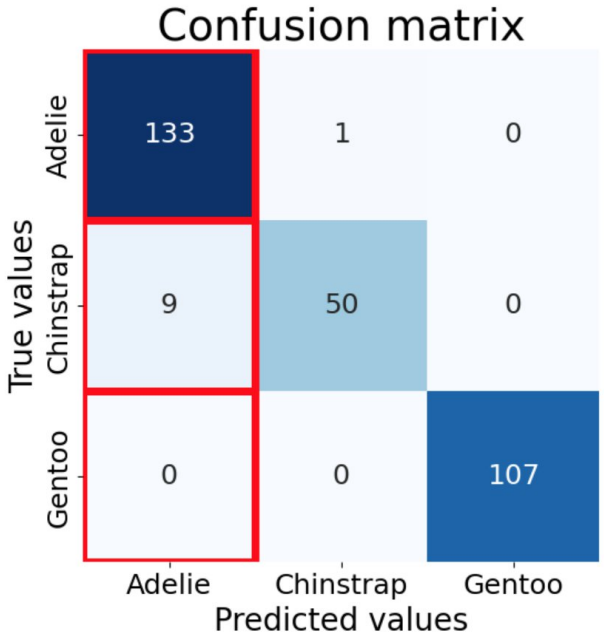
Precision : proportion of true predictions for a class out of all predictions of that class



	precision	recall	f1-score	support
Adelie	0.94	0.99	0.96	134
Chinstrap	0.98	0.85	0.91	59
Gentoo	1.00	1.00	1.00	107
accuracy			0.97	300
macro avg	0.97	0.95	0.96	300
weighted avg	0.97	0.97	0.97	300

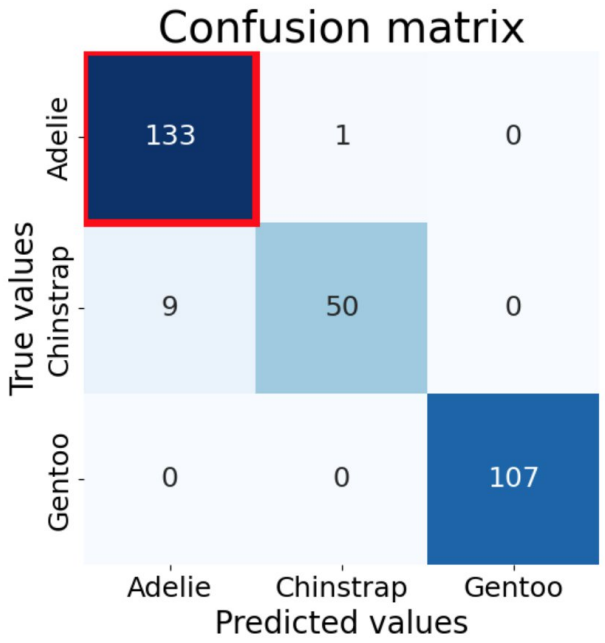
$$\text{Precision} = \frac{}{} = \frac{133}{133 + 9} = 0.94$$

(for Adelie class)



Classification metrics

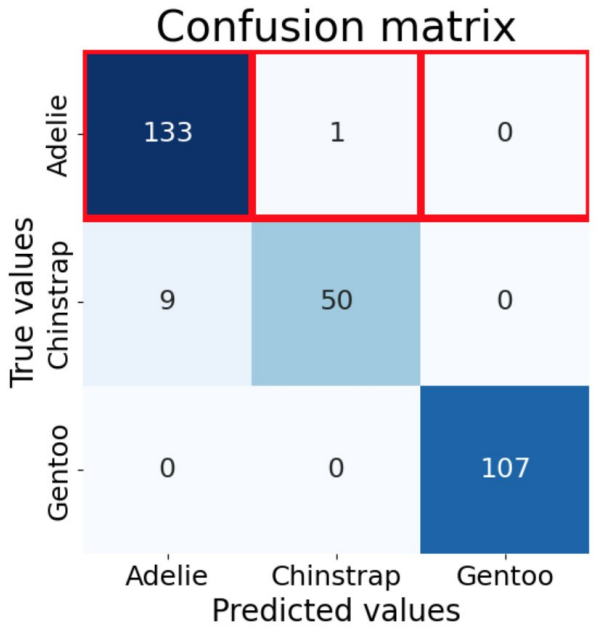
Recall :
proportion of true predictions for a class out of all actual instances of that class



	precision	recall	f1-score	support
Adelie	0.94	0.99	0.96	134
Chinstrap	0.98	0.85	0.91	59
Gentoo	1.00	1.00	1.00	107
accuracy			0.97	300
macro avg	0.97	0.95	0.96	300
weighted avg	0.97	0.97	0.97	300

$$\text{Recall} = \frac{}{} = \frac{133}{133 + 1} = 0.99$$

(for Adelie class)



Precision or recall centric problem ?

A model that predicts diseases ?

Recall : missing a disease diagnosis can have serious consequences

A movie recommendation model ?

Precision : recommending fewer, highly relevant movies

A fraud detection system for banking transactions?

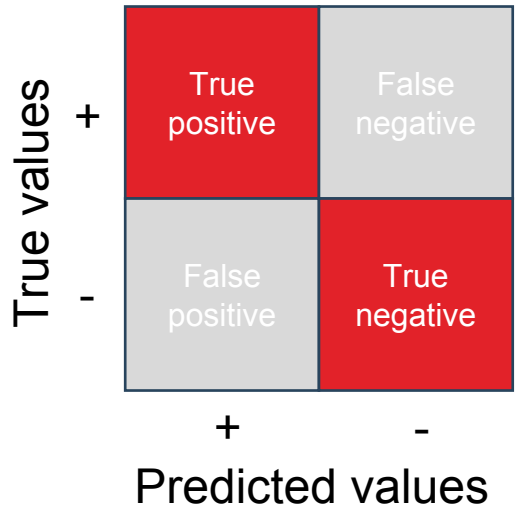
Recall : missing fraudulent transactions is worse than flagging a few legitimate ones

An email spam filter ?

Precision : better to avoid marking important emails as spam

A self-driving car detecting pedestrians ?

Recall : Failing to detect a pedestrian is far more dangerous than stopping unnecessarily



Scikit-Learn convention with True values horizontally

Classification metrics

	precision	recall	f1-score	support
Adelie	0.94	0.99	0.96	134
Chinstrap	0.98	0.85	0.91	59
Gentoo	1.00	1.00	1.00	107
accuracy			0.97	300
macro avg	0.97	0.95	0.96	300
weighted avg	0.97	0.97	0.97	300

F1 score :

harmonic mean of precision and recall

$$F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

$$F\beta\ score = (1 + \beta^2) \times \frac{Precision \times Recall}{(\beta^2 \times Precision) + Recall}$$

$$if\ \beta^2 > 1,\ Recall\ is\ more\ important$$
$$if\ \beta^2 < 1,\ Precision\ is\ more\ important$$

Why harmonic mean ?

It penalizes low values more heavily than the arithmetic or geometric mean. This prevents a model from maximizing just one metric

Precision = 0.1 | Recall = 0.9
Arithmetic mean = 0.5
Geometric mean = 0.3
Harmonic mean = 0.18

```python
from sklearn.metrics import f1_score, fbeta_score

f1_score(y_true, y_pred, average='macro')
fbeta_score(y_true, y_pred, average='macro', beta=0.5)
```

Code libraries : F-1 score Scikit-Learn, F-Beta score Scikit-Learn, F-1 score Torchmetrics, F-Beta score Torchmetrics

Classification metrics

Different ways of computing evaluation metrics

1. **Micro** : aggregate the contributions of all classes before computing the metric

$$Micro\ Precision = \frac{\sum TP}{\sum TP + \sum FP}$$

More weight to larger classes

2. **Macro** : compute the metric individually for each class then take the arithmetic mean

$$Macro\ Precision = \frac{1}{n\_class} \sum_{i=1}^{n\_class} Precision_i$$

Overemphasize minority classes

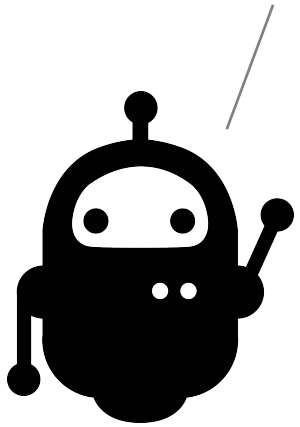
3. **Weighted** : same as macro but each class is weighted by its support

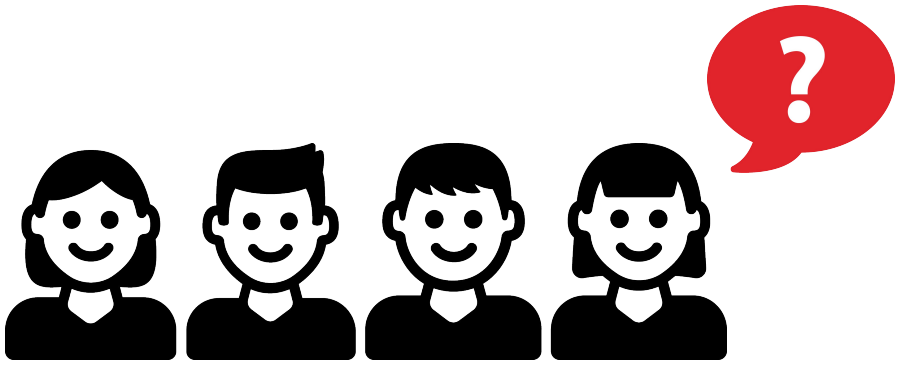
$$Weighted\ Precision = \frac{1}{n\_tot} \sum_{i=1}^{n\_class} n_i.Precision_i$$

Balances both class imbalance and overall performance

4. **Samples** : arithmetic mean of the metric calculated for each sample for multilabel classification

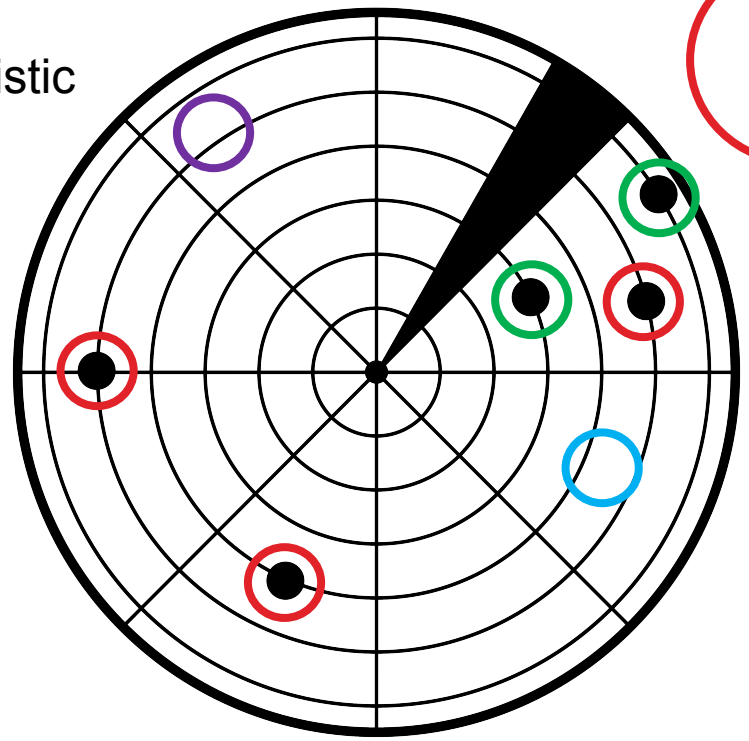
Quelques questions ?





Classification metrics : ROC curve

Receiver Operating Characteristic



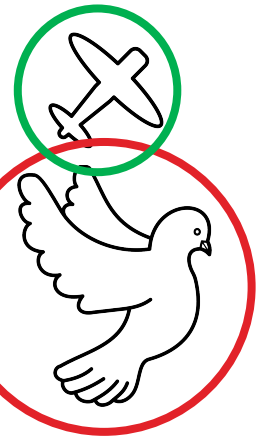


Image credits :
Radar screen : Perkasa Rambe
Radar antenna : ArashDesign
Plane : zaenul yahya
Pigeon : Olena Panasovska

True values \ Predicted values	0	1
0	True negative (TN)	False positive (FP)
1	False negative (FN)	True positive (TP)

$$False\ positive\ rate = \frac{FP}{FP + TN}$$

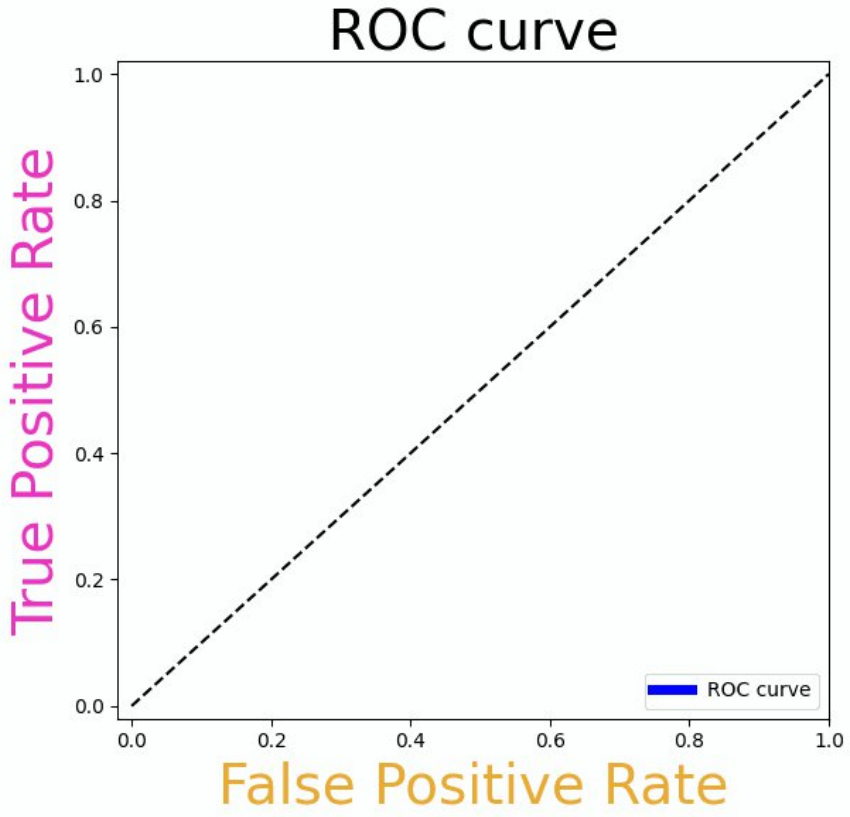
$$True\ positive\ rate = \frac{TP}{TP + FN}$$

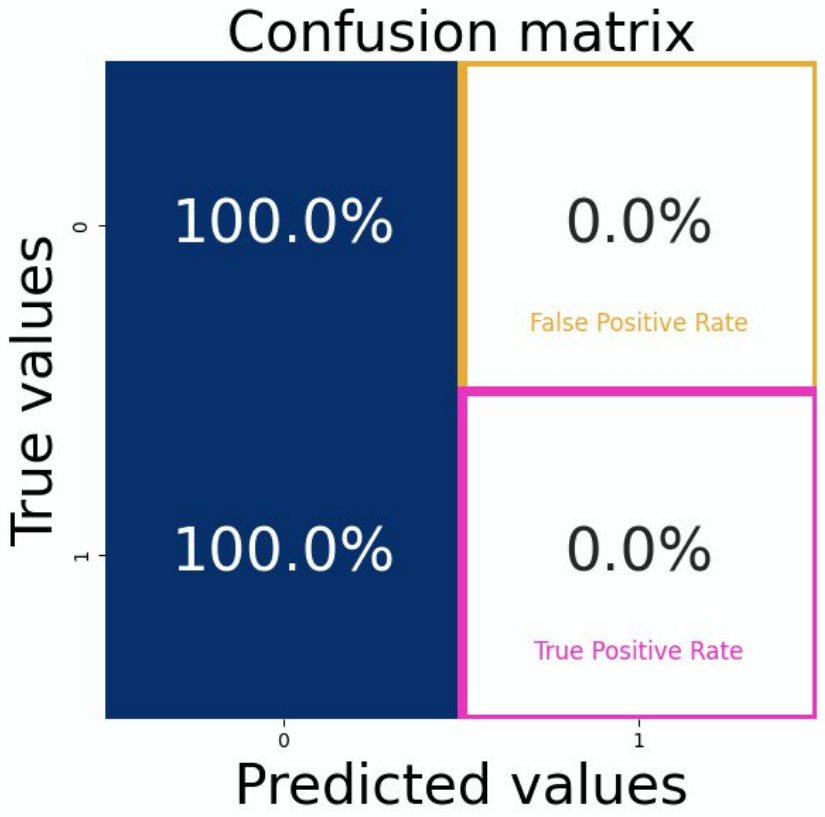
$$(recall\ or\ sensitivity)$$

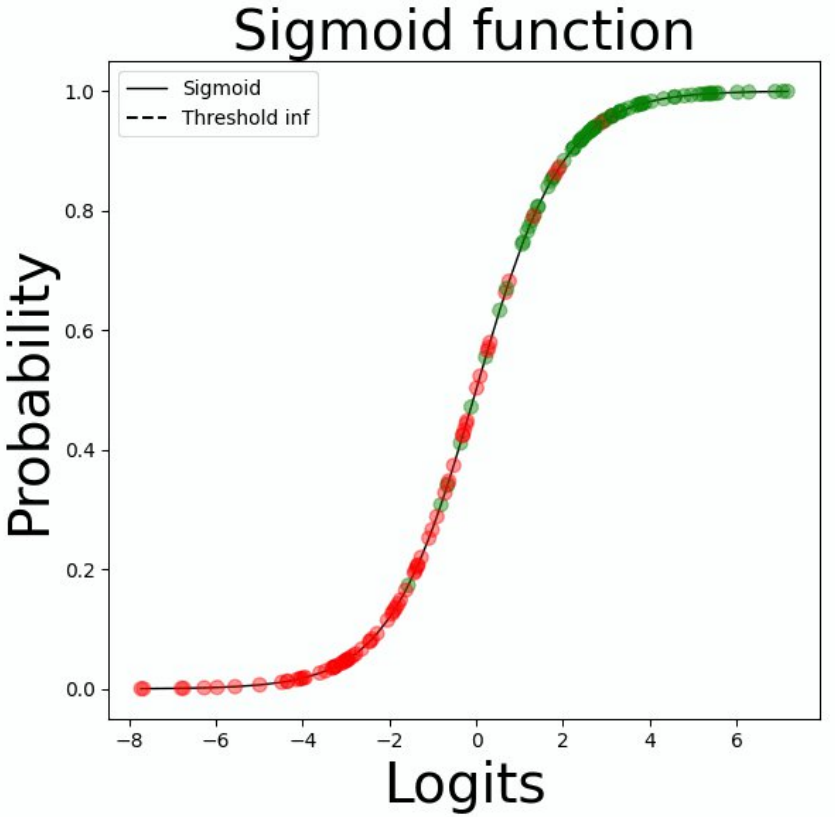
Trade-off between detection and false alarms

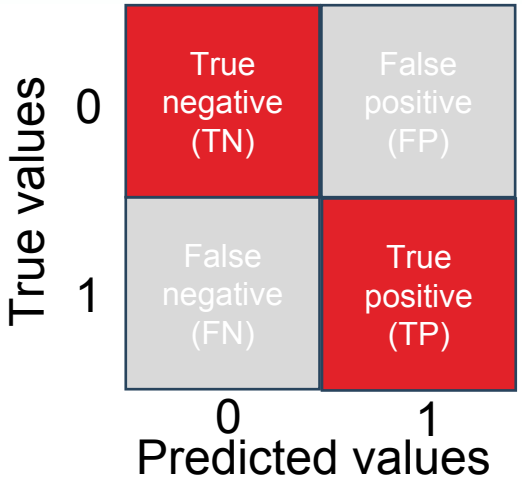
Classification metrics : ROC curve

AUC metric : Area Under the ROC Curve









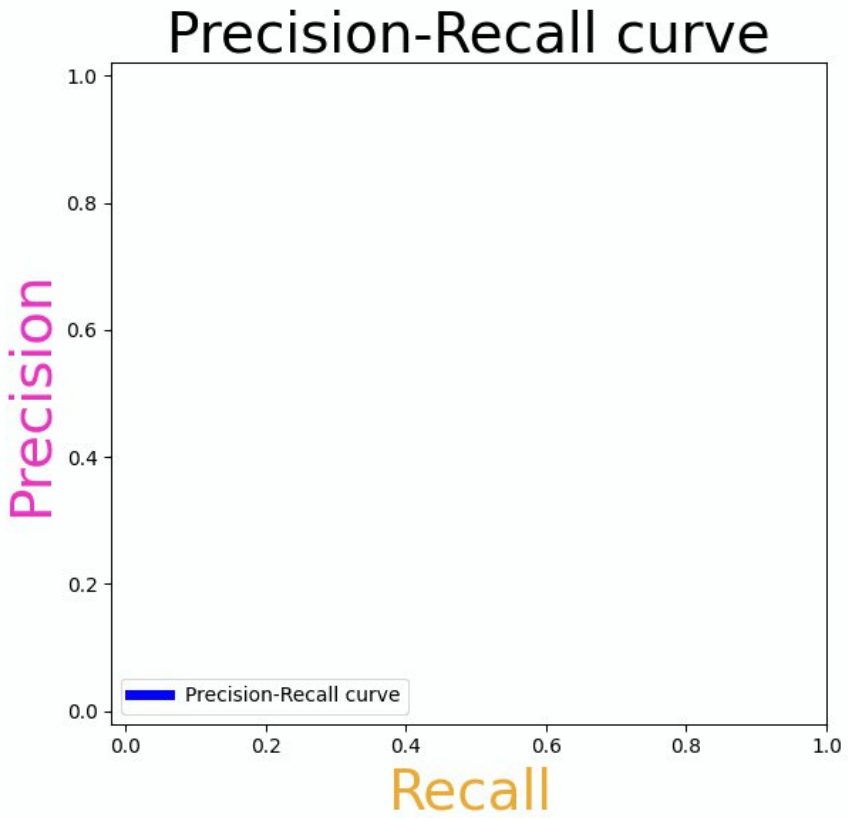
$$False\ positive\ rate = \frac{FP}{FP + TN}$$

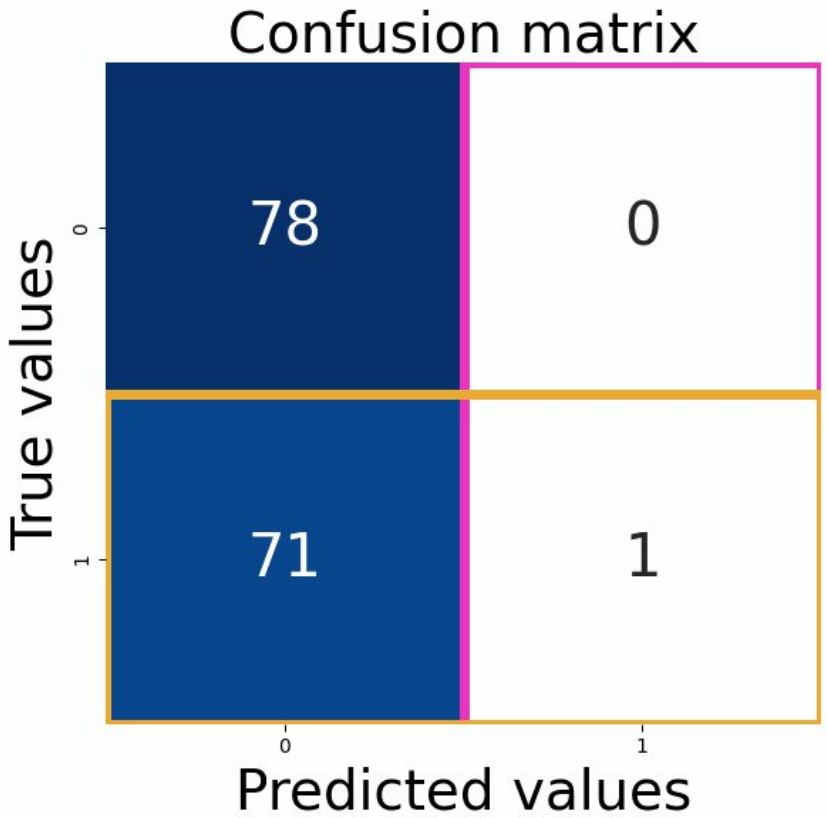
$$True\ positive\ rate = \frac{TP}{TP + FN}$$

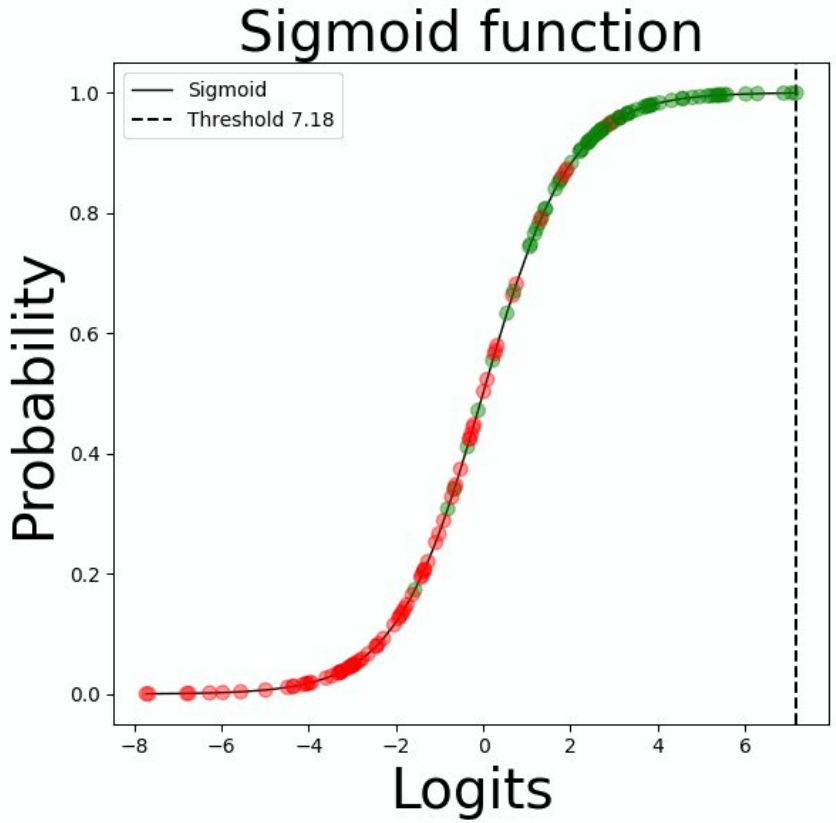
$$(recall\ or\ sensitivity)$$

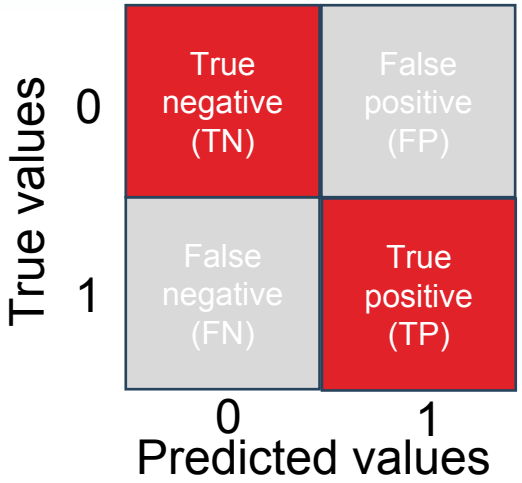
Multiclass classification ROC curve is possible with OvR, OvO, micro-averaged ROC or macro-averaged ROC

Classification metrics : Precision-Recall curve









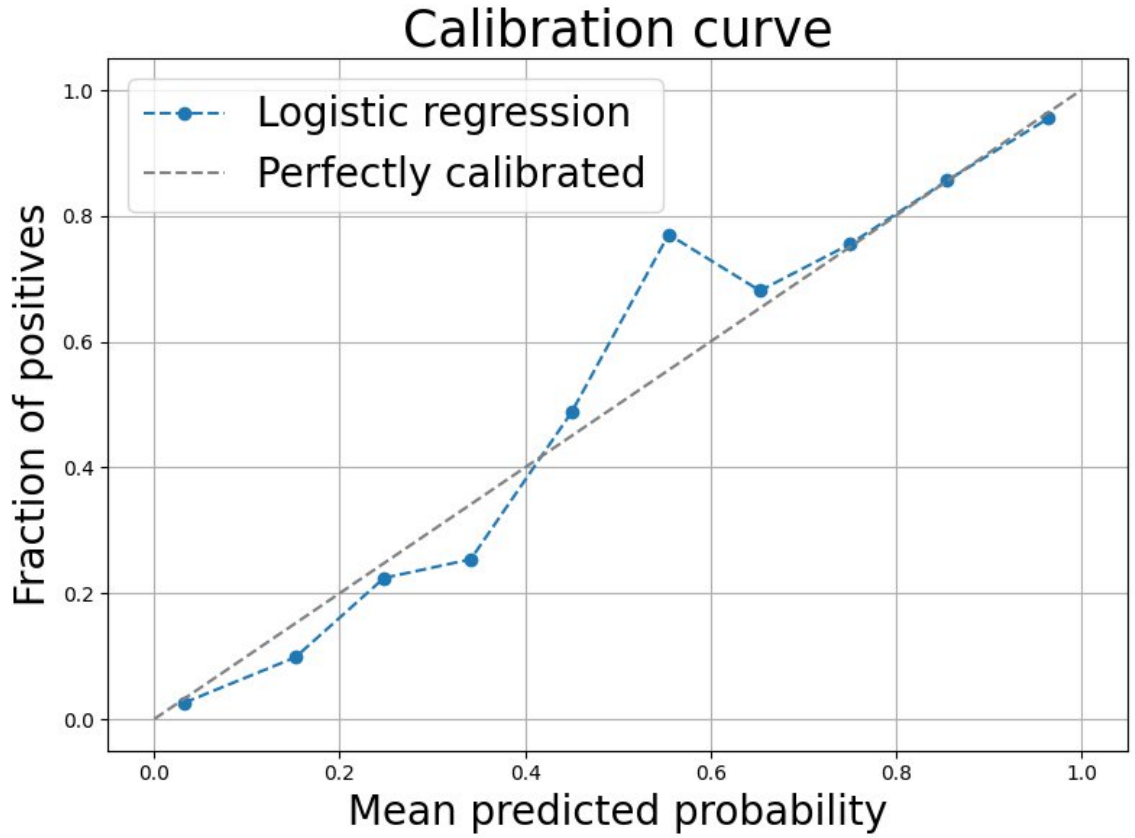
$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

Trade-off between false positive and false negative

Calibration metrics

If a model assigns a probability of P(a)=0.7 to a patient having a disease, does this mean that out of 10 patients with the same predicted probability, exactly 7 will actually have the disease ?

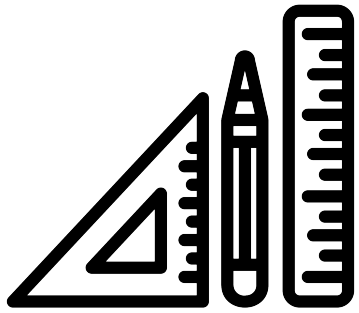


Important when probabilities need to be meaningful (medicine, risky industries, finance…)

Expected Calibration Error (ECE)
Maximum Calibration Error (MCE)

Paper : On Calibration of Modern Neural Networks, Chuan Guo, Geoff Pleiss, Yu Sun, Kilian Q. Weinberger, 2017

Tool : how to monitor evaluation metrics during training ?



TensorBoard

1. Part of Tensorflow/Keras callbacks
2. Usable with Pytorch (torch.utils.tensorboard)
3. Different visualization (metrics and weights)
4. Real-time updates

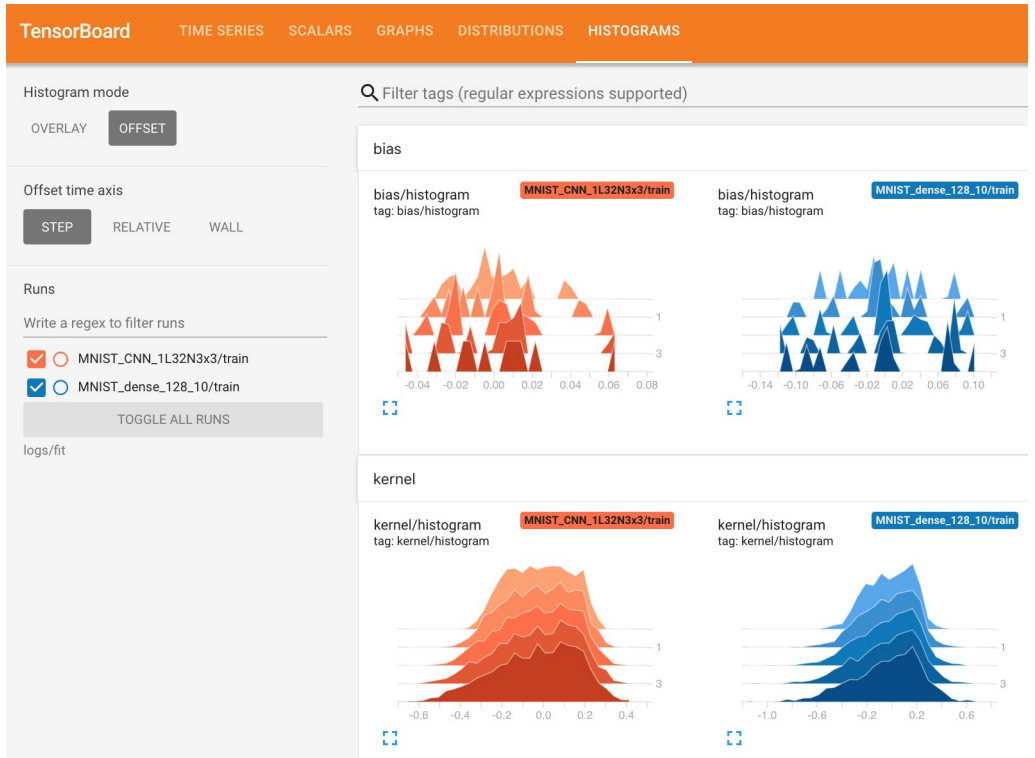


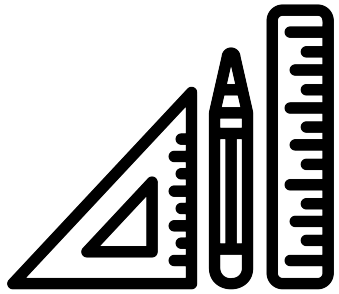
Image credits : metami septiana (tool)

```
from tensorflow.keras.callbacks import TensorBoard

tensorboard_callback = TensorBoard(log_dir="logs/fit", histogram_freq=1)
[...]
model.fit(X_train, y_train, epochs=5, callbacks=[tensorboard_callback])
```

Code library : TensorBoard

Tool : how to monitor evaluation metrics in production ?



MLflow

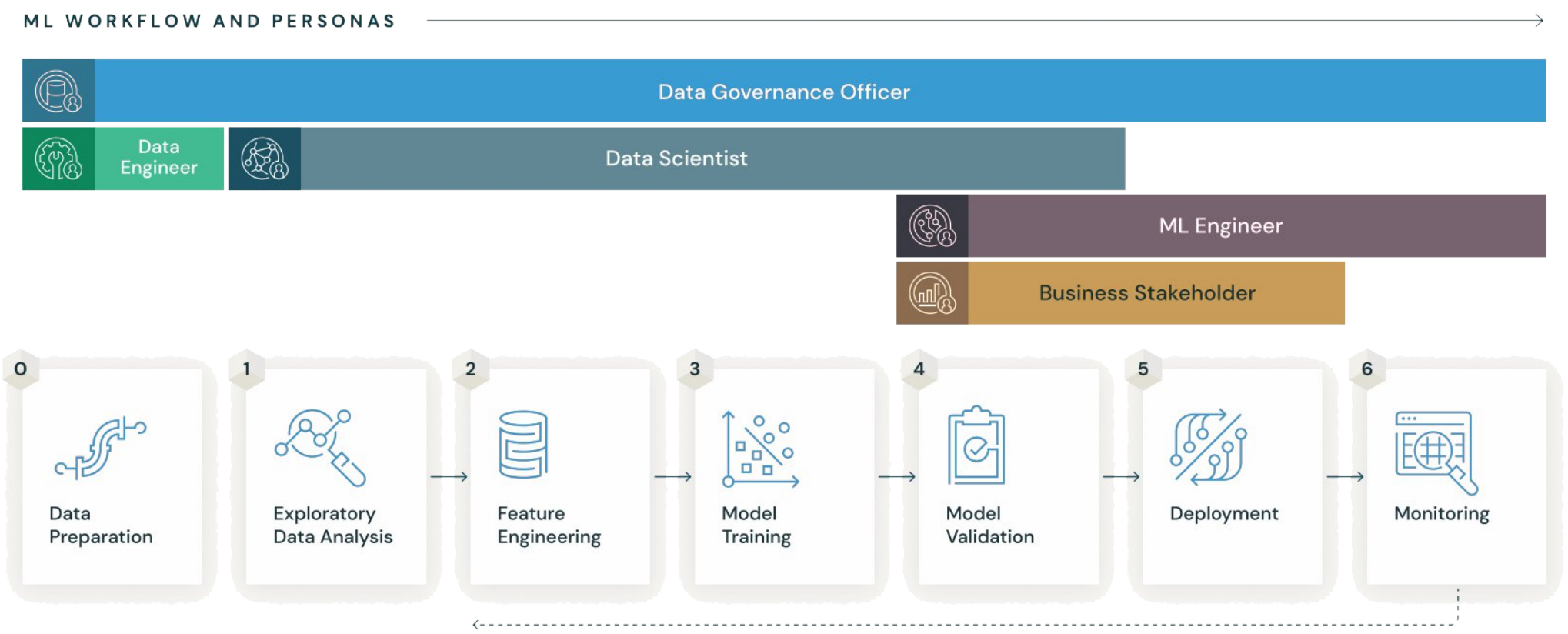
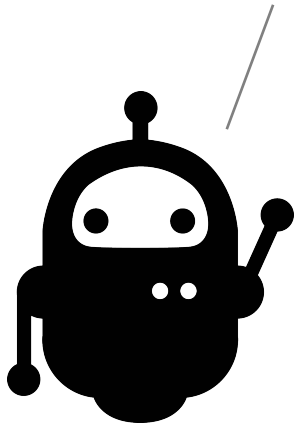


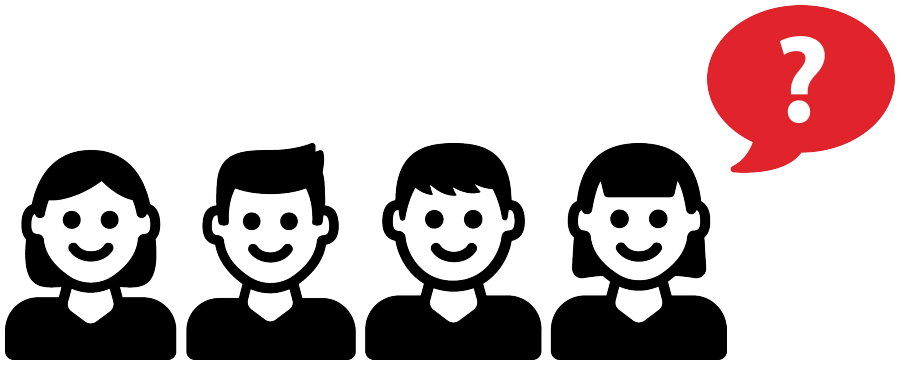
Image credits : metami septiana (tool), MLflow

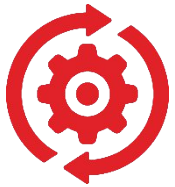
1. Open-source Python library
2. End-to-end ML lifecycle (tracking, projects, models, registry, serving, UI)
3. Framework-agnostic

Code library : MLflow

Quelques questions ?







Improve imbalanced dataset training

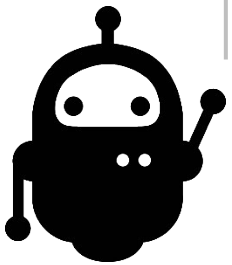
Notebook : Imbalanced.ipynb

Objective :

Improve prediction quality when working with an imbalanced dataset

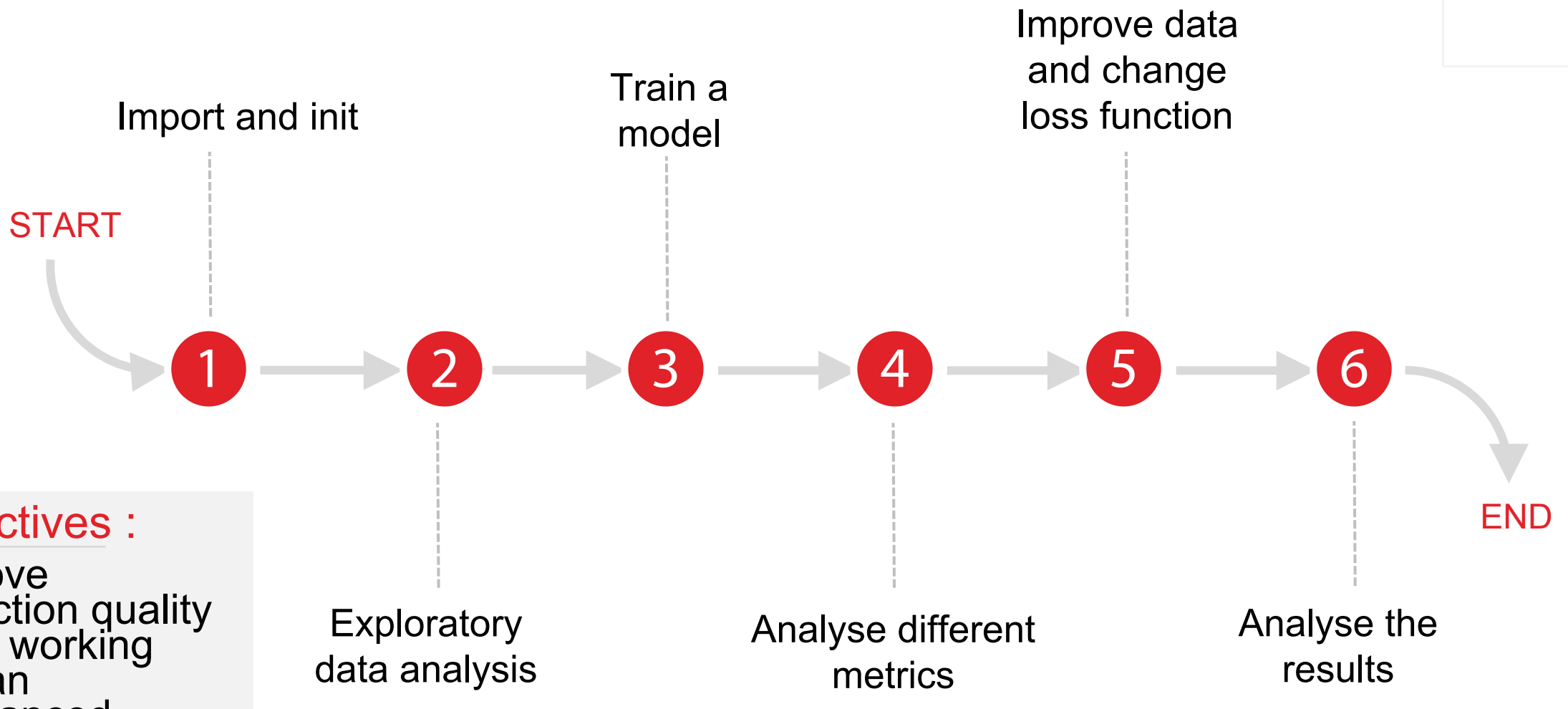
Dataset :

We extract four specific classes (fish, dolphin, shark and whale) from the Quick Draw dataset. To introduce class imbalance, one class is intentionally undersampled while another is oversampled.



Imbalanced.ipynb

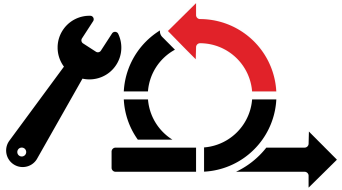
10 minutes



Objectives :

Improve prediction quality when working with an imbalanced dataset

11



Learning optimisation

Parameters & metrics

1. Data Quality
   Why is is so important ?
2. Loss functions and evaluation metrics
3. **Learning optimization**

Optimizers

FIDLE

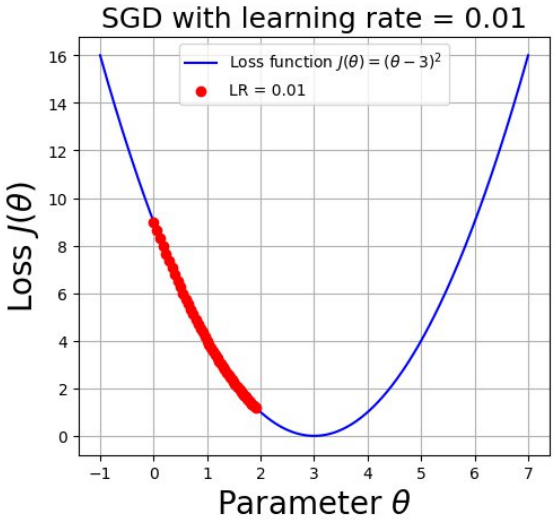
Optimizers

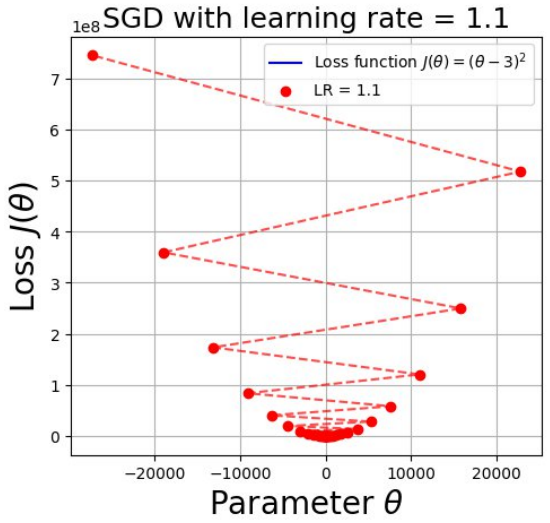
Goal : adjust the model's parameters to minimize the loss function

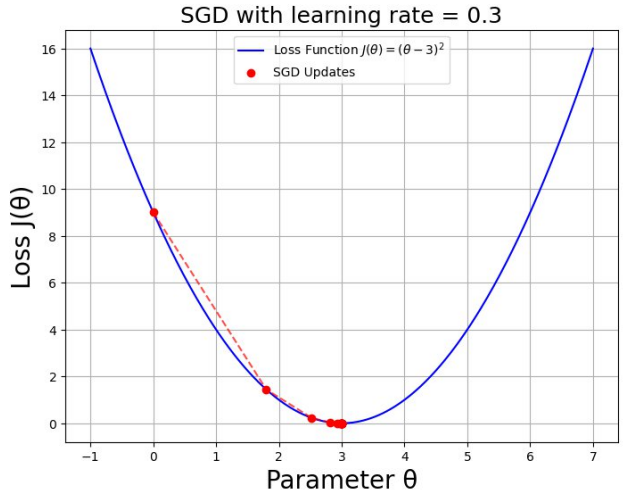
Gradient descent

$$\theta_{t+1} = \theta_t - \eta . \nabla_\theta J(\theta_t)$$

With θ_t and θ_{t+1} model's parameters at batch t and t+1, η learning rate, J loss function and $\nabla_\theta J$ gradient of the loss function







Different kinds of gradient descent :

1. Stochastic gradient descent (SGD)
2. Batch gradient descent
3. Mini-batch gradient descent

```
sgd = tf.keras.optimizers.SGD(learning_rate=0.01)
model.compile(optimizer=sgd, loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_val, y_val)
```

Code libraries : Keras optimizers, Pytorch optimizers

Optimizers

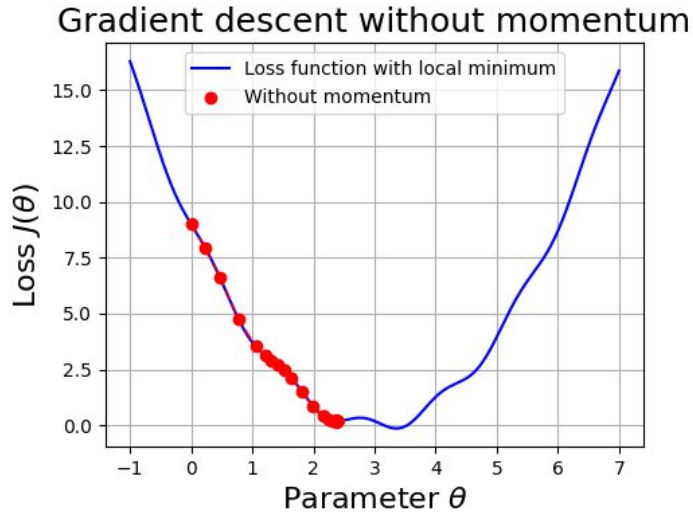
Gradient descent with momentum

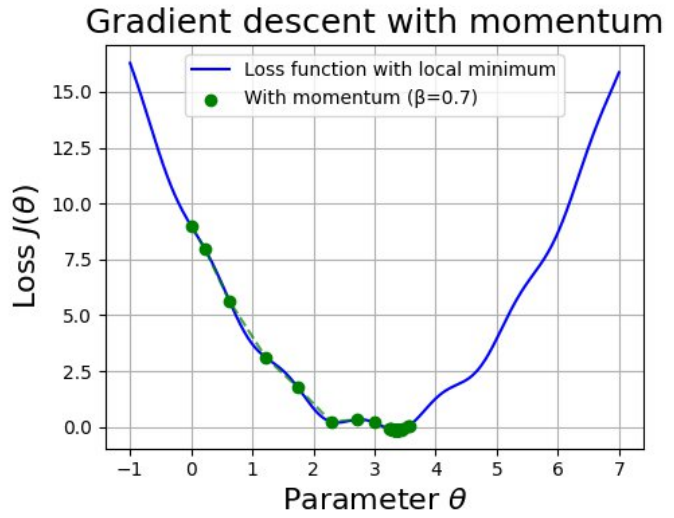
Improvements : faster convergence, reducing oscillations, escaping local minima

$$m_{t+1} = \beta m_t - \eta . \nabla_\theta J(\theta_t)$$

$$\theta_{t+1} = \theta_t + m_{t+1}$$

With θ_t and θ_{t+1} model's parameters at batch t and t+1, m_t and m_{t+1} the momentum vector at batch t and t+1, β the momentum, η learning rate, J loss function and $\nabla_\theta J$ gradient of the loss function






```
sgd = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.5)
model.compile(optimizer=sgd, loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_val, y_val))
```

Code libraries : Keras optimizers, Pytorch optimizers

Paper : SOME METHODS OF SPEEDING UP THE CONVERGENCE OF ITERATIONMETHOD , B. T. Polyak, 1962

Optimizers

Adaptive learning rate methods :

AdaGrad (Adaptive Gradient)

$$G_t = G_{t-1} + \nabla_\theta J(\theta_t)^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \varepsilon}} \nabla_\theta J(\theta_t)$$

Modifies the learning rate per parameter based on past gradients

RMSProp (Root Mean Square Propagation)

$$G_t = \beta G_{t-1} + (1-\beta)\, \nabla_\theta J(\theta_t)^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \varepsilon}} \nabla_\theta J(\theta_t)$$

Prevents adaptive learning rates from decaying too much

```
adagrad = tf.keras.optimizers.Adagrad(learning_rate=0.01)
model.compile(optimizer=adagrad, loss='binary_crossentropy', metrics=['accuracy'])

rmsprop = tf.keras.optimizers.RMSprop(learning_rate=0.01, rho=0.9)
model.compile(optimizer=rmsprop, loss='binary_crossentropy', metrics=['accuracy'])
```

Papers :

Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, Duchi, John and Hazan, Elad and Singer, Yoram, 2011

Lecture 6a Overview of mini--batch gradient descent, Geoffrey Hinton, Nitish Srivastava, Kevin Swersky

Optimizers

Combinaison of gradient descent with momentum and adaptive learning rate : Adam (Adaptive Moment Estimation)

1. Momentum

$$m_t = \beta_1 m_{t-1} - (1-\beta_1)\nabla_\theta J(\theta_t)$$

2. Adaptive learning rate

$$G_t = \beta_2 G_{t-1} + (1-\beta_2)\,\nabla_\theta J(\theta_t)^2$$

3. Biais correction

$$\hat{m}_t = \frac{m_t}{1-\beta_1{}^t} \qquad \hat{G}_t = \frac{G_t}{1-\beta_2{}^t}$$

4. Parameters update

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{G}_t + \varepsilon}}\hat{m}_t$$

```
adam = tf.keras.optimizers.Adam(learning_rate=0.01, beta_1=0.9, beta_2=0.999)
model.compile(optimizer=adam, loss='binary_crossentropy', metrics=['accuracy'])
```

Paper : Adam: A Method for Stochastic Optimization, Diederik P. Kingma, Jimmy Ba, 2014

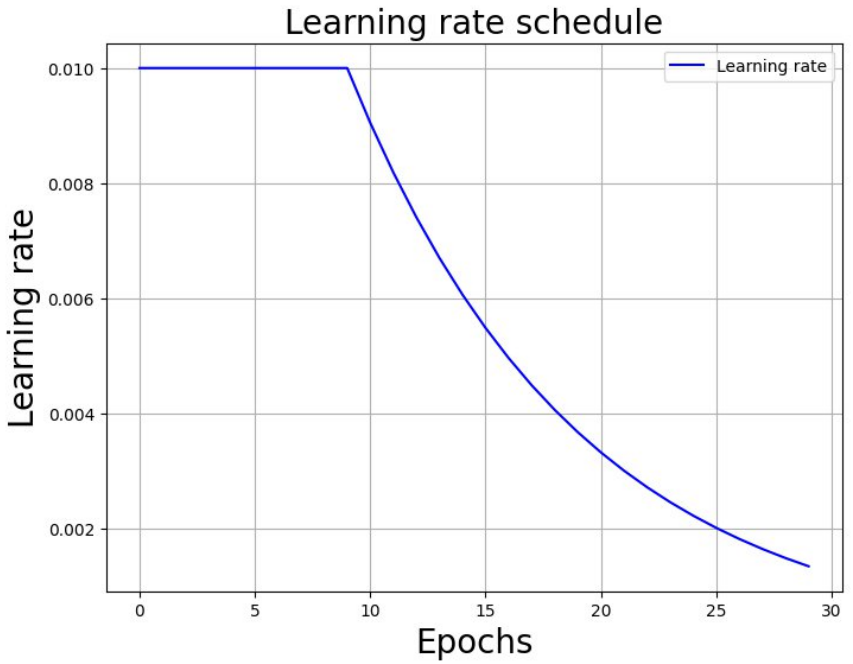
Adapt learning rate during training

Different strategies to update the learning rate during training :

1. Based on epoch number : learning rate scheduler

```
def lr_scheduler(epoch, lr):
  if epoch < 10:
    return lr
  else:
    return lr * ops.exp(-0.1)

lr_scheduler = LearningRateScheduler(lr_schedule)
model.fit(X_train, y_train, epochs=10, batch_size=32, callbacks=[lr_scheduler])
```



2. Based on loss or metrics : reduce LR on plateau

```
reduce_lr = ReduceLROnPlateau(
    monitor='val_loss', factor=0.2, patience=5, min_lr=0.001
)
model.fit(X_train, y_train, epochs=10, batch_size=32, callbacks=[reduce_lr])
```

Code libraries : Keras learning rate scheduler, Keras reduce LR on plateau, Pytorch learning rate scheduler and reduce LR on plateau

Gradient clipping

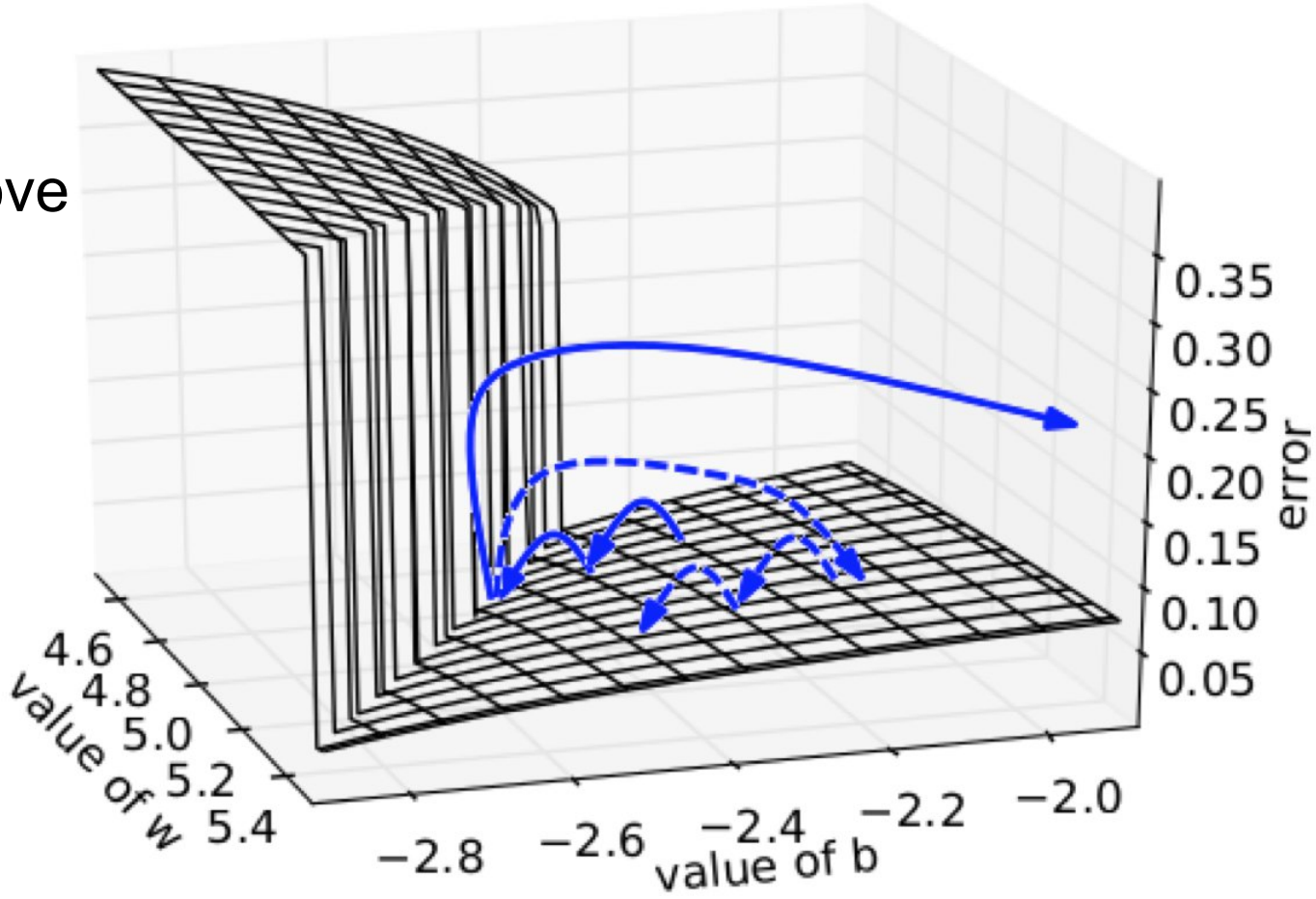
Goals :
prevent exploding gradients and improve training stability

Norm-based gradient clipping

$$\nabla_\theta J_{clip} = \nabla_\theta J \frac{\tau}{\|\nabla_\theta J\|_2}$$
$$if \ \|\nabla_\theta J\|_2 > \tau$$

Value-based gradient clipping

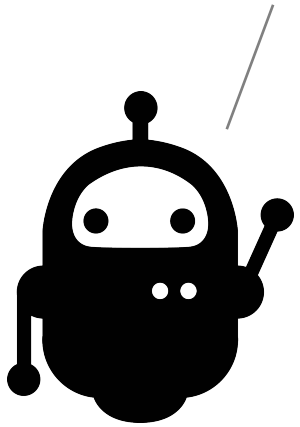
$$\nabla_\theta J_{clip} = max(-\tau, \min(\nabla_\theta J, \tau))$$

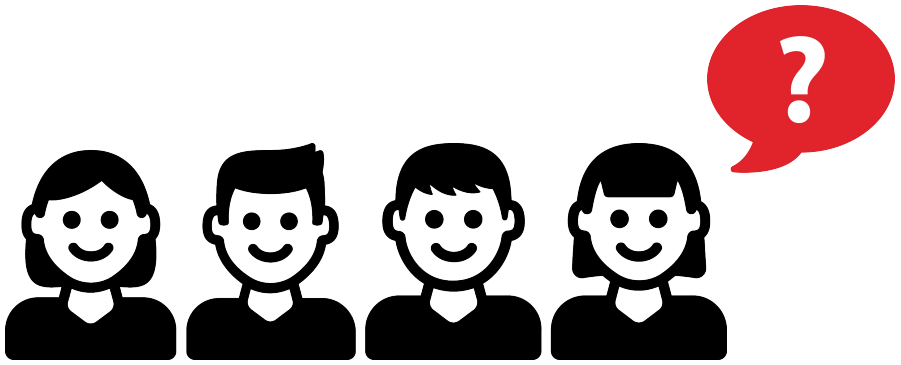


Paper : On the difficulty of training Recurrent Neural Networks, Razvan Pascanu, Tomas Mikolov, Yoshua Bengio, 2012

```
from tensorflow.keras.optimizers import Adam
optimizer = Adam(learning_rate=0.001, clipnorm=1.0, clipvalue=0.5)
model.compile(optimizer=optimizer, loss='categorical_crossentropy')
```

Quelques questions ?

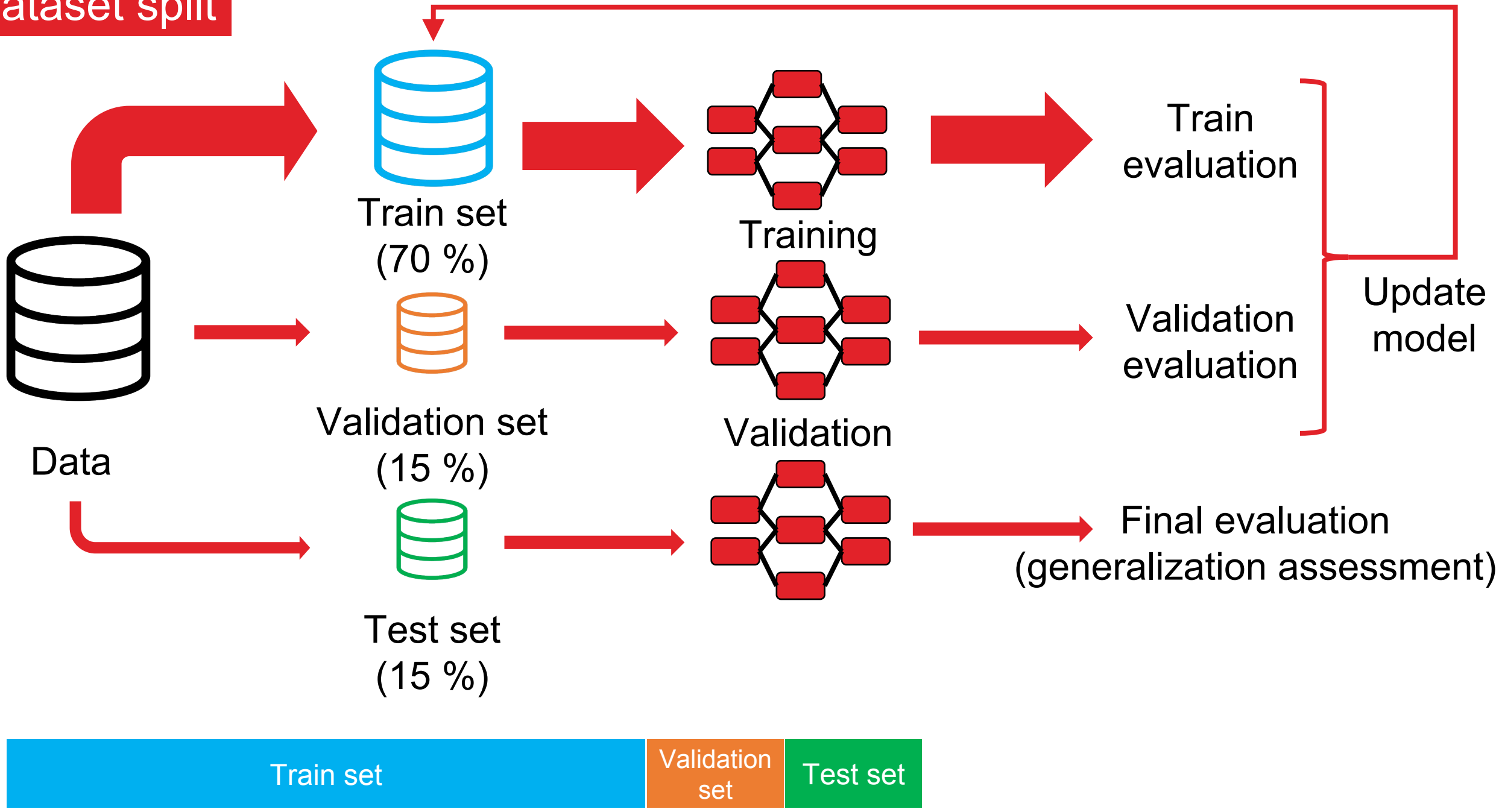




Model evaluation and hyperparameter tuning

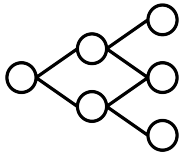
FIDLE

Dataset split

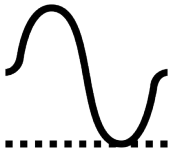


Hyperparameters tuning

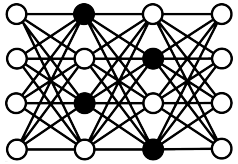
Key hyperparameters in deep learning optimize during training



Model architecture : number of layers, number of neurons, activation functions…

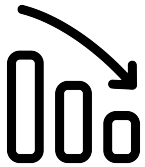


Optimizer : SGD, Adam, RMSprop…



Regularization : L1, L2, dropout…

Loss function : cross entropy, focal loss…

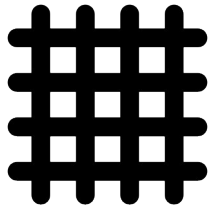


Loss function : cross entropy, focal loss…

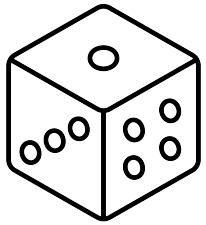
Image credits : Abd Majd (neural network), Md Ahasan Habib (minimum), Lucas Rathgeb (regularization), Nur Achmadi Yusuf (batch), Ilyas Aji Furqon (loss)

Hyperparameters tuning

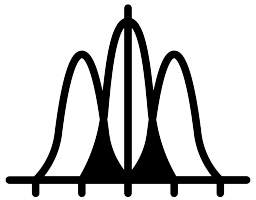
Methods to find the best hyperparameters



Grid search



Random search

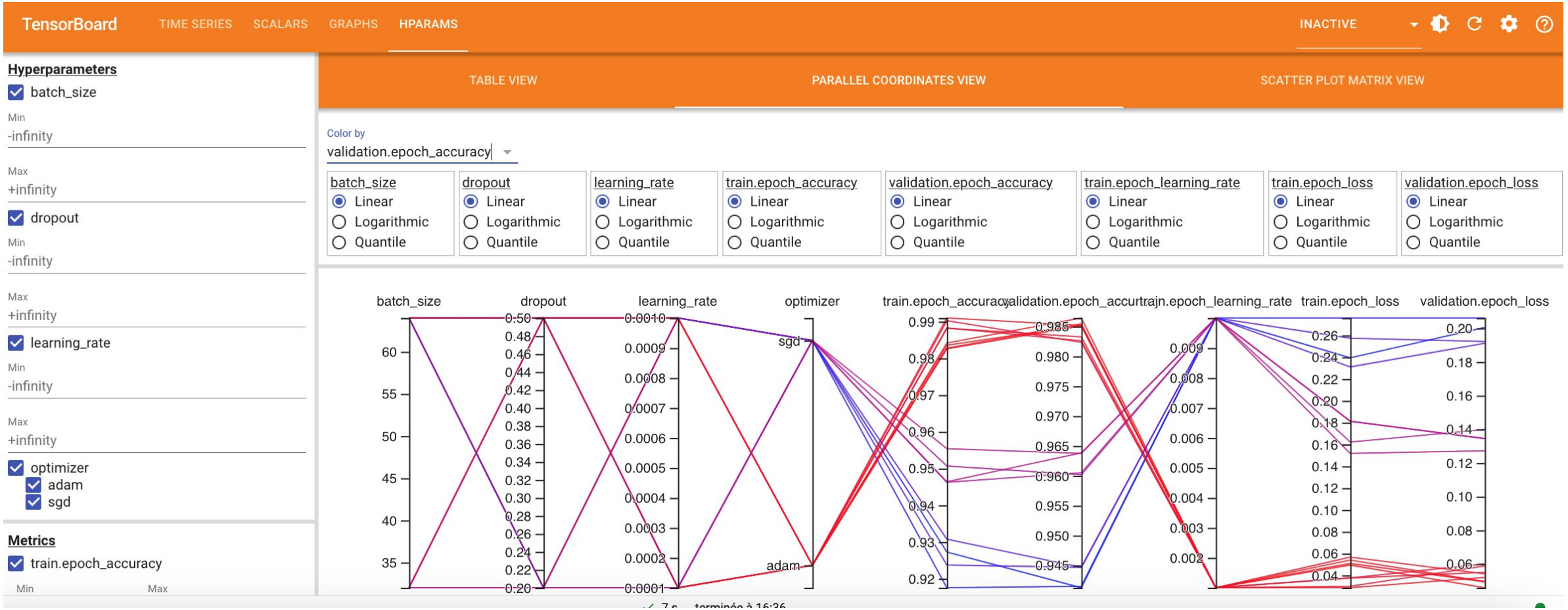


Bayesian search

Image credits : Yoyon Pujiyono (dice), fhr! Studio (bayesian), Larea (grid)

Tool : track experiments

HParams in TensorBoard



Track hyperparameters

Visualize results

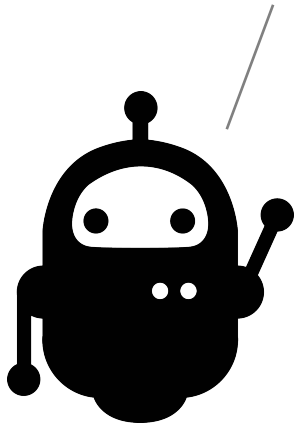
Plot parallel coordinates

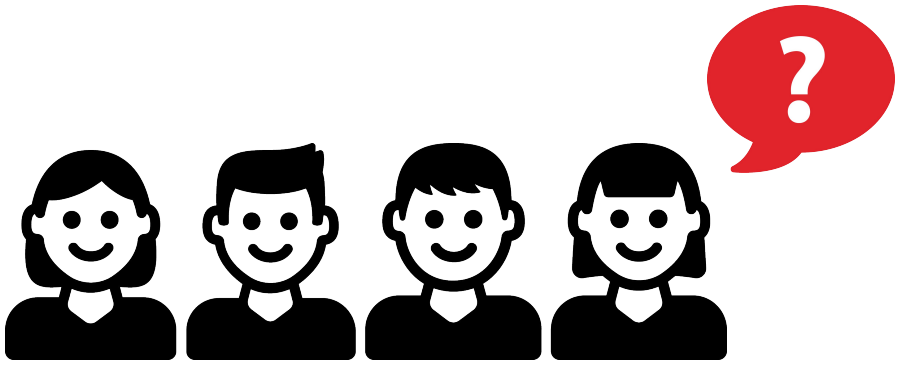
Summary : default choices

Problem type	Loss function	Activation function of the last layer	Optimizer	Evaluation metrics
Regression	MSE	Linear	Adam	MAE/RMSE/R2 Residuals
Binary classification	Binary cross entropy	Sigmoid	Adam	Accuracy/precision/recall/f1-score Confusion matrix
Multi class classification	Categorical cross entropy	Softmax	Adam	Accuracy/precision/recall/f1-score Confusion matrix
Multi label classification	Binary cross entropy	Sigmoid	Adam	Accuracy/precision/recall/f1-score Confusion matrix

Paper : LOSS FUNCTIONS AND METRICS IN DEEP LEARNING, Juan Terven, Diana M. Cordova-Esparza, Alfonso Ramirez-Pedraza, Edgar A. Chavez-Urbiola, Julio A. Romero-Gonzalez, 2024

Quelques questions ?

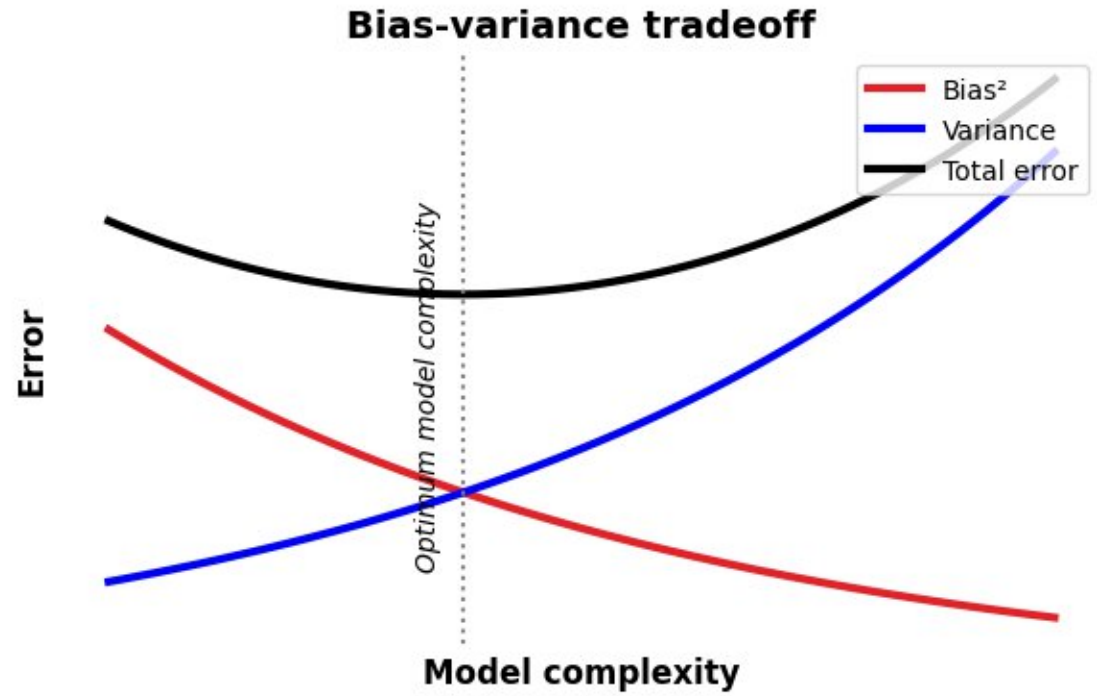




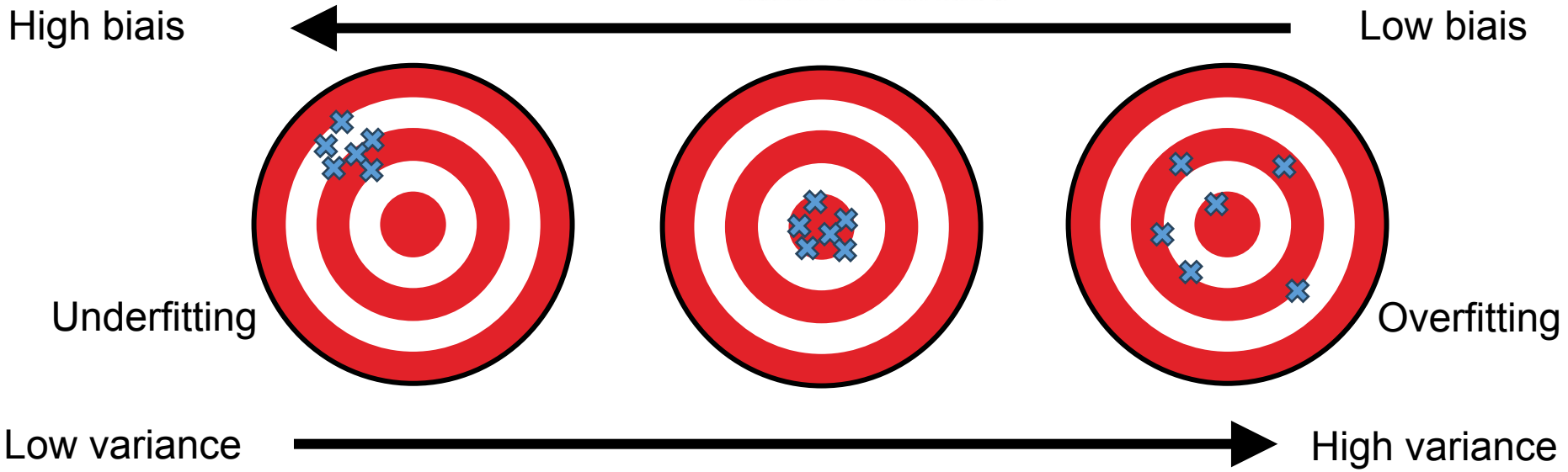
Regularization techniques

FIDLE

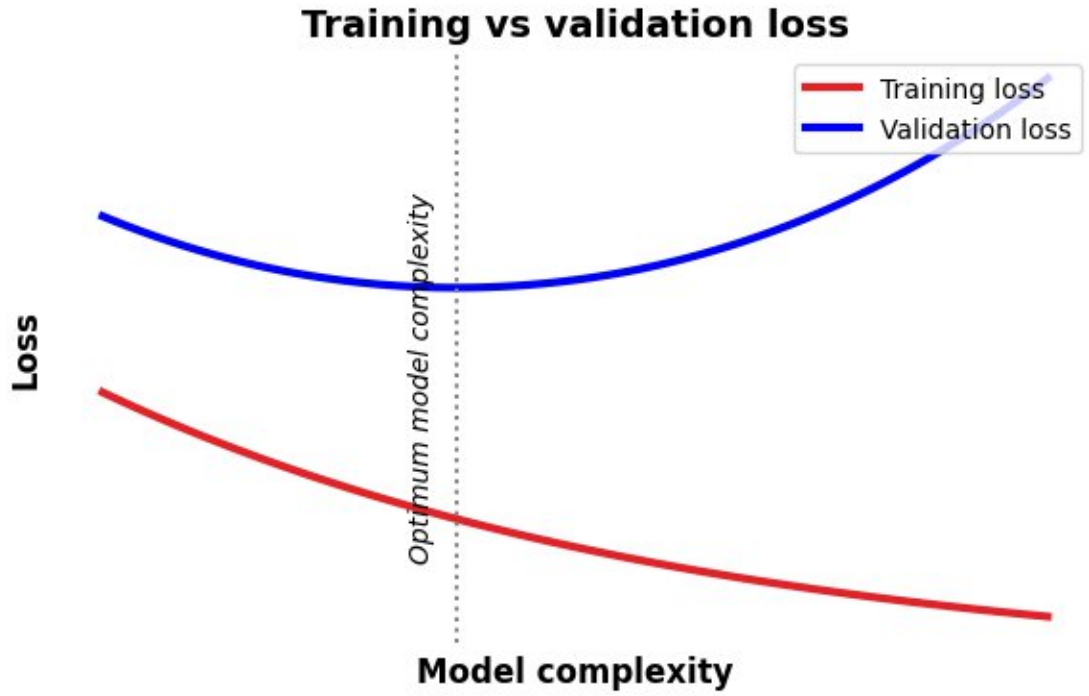
Why regularization ?



Regularization : reducing variance without increasing bias too much

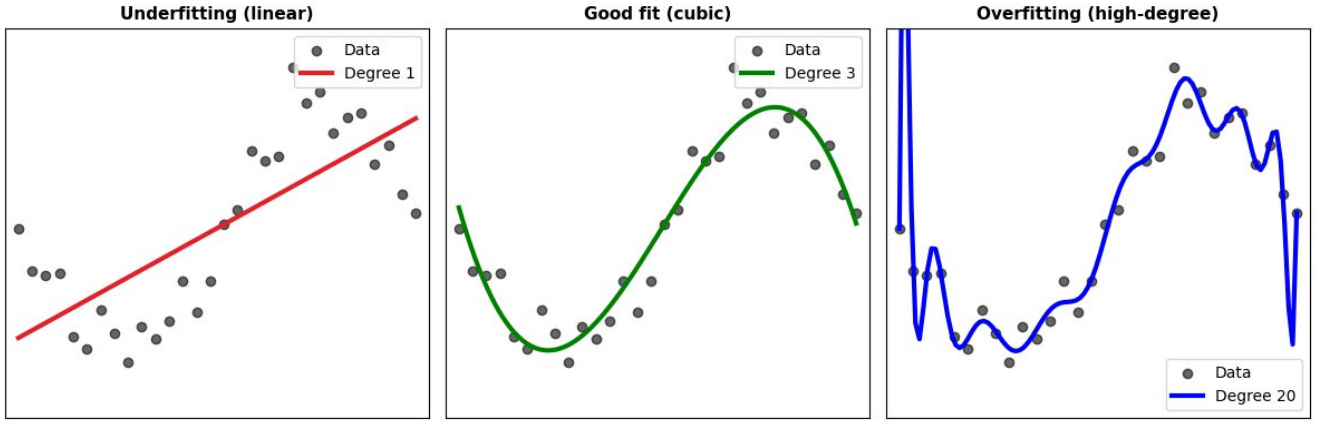


Why regularization ?



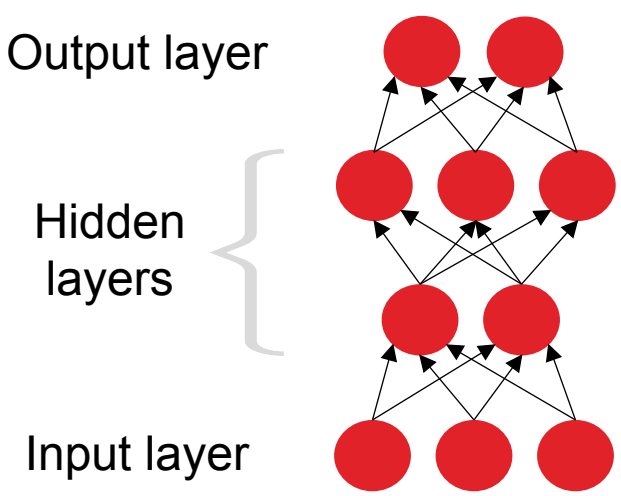
Optimum model complexity at the minimum of the validation loss

High biais ← → Low biais

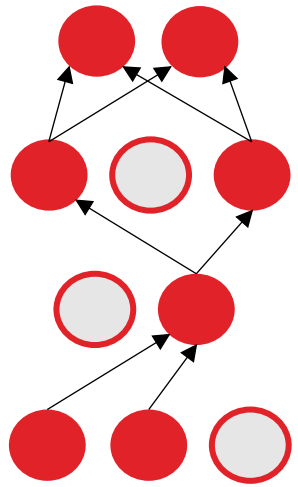


Low variance → High variance

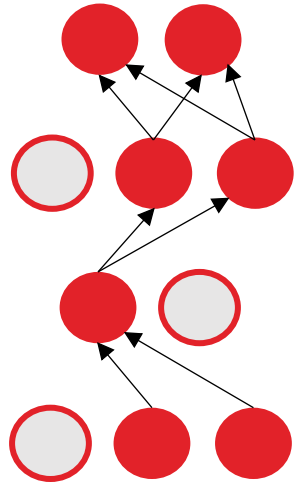
Forcing diversity in learning : dropout layers



Original network



Network with dropout layers at batch n



Network with dropout layers at batch n+1

Randomly setting to zero a percentage of neurons

Avoid becoming too reliant on any specific neuron

Only at training phase

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout

model = Sequential()
model.add(Dropout(0.33, input_shape=(3,)))
model.add(Dense(2, activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(3, activation="relu"))
model.add(Dropout(0.33))
model.add(Dense(2))
```

Code libraries : Keras dropout layer, PyTorch dropout layer

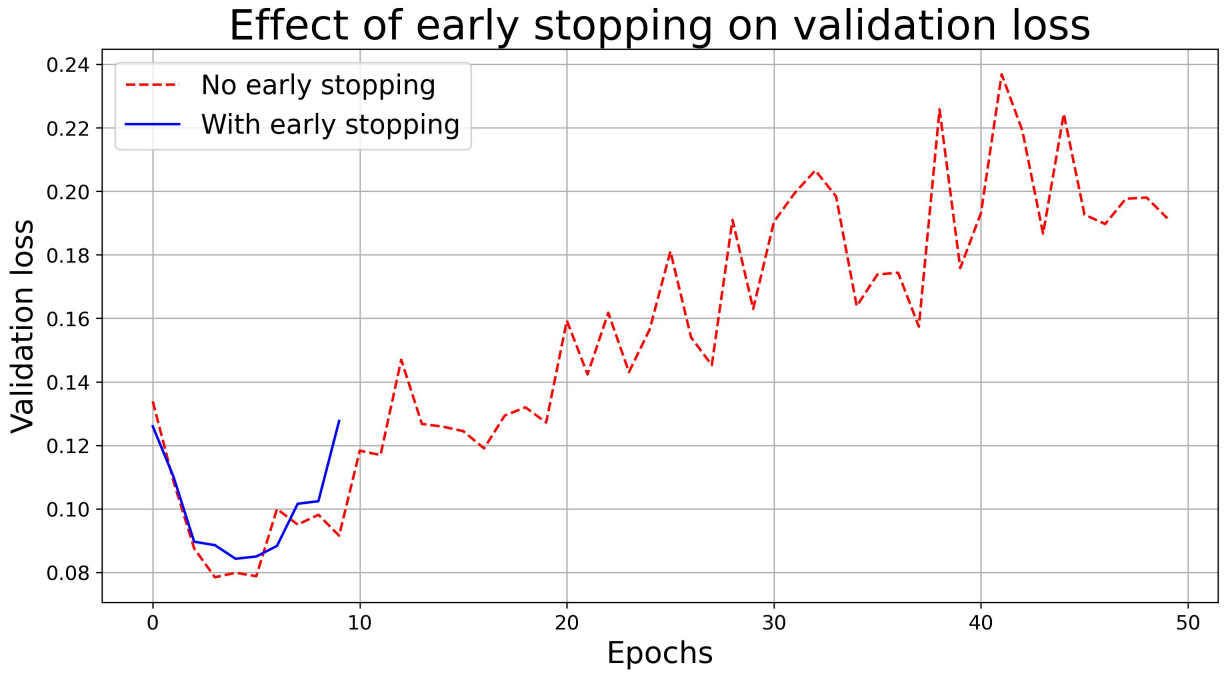
Papers :

Improving neural networks by preventing co-adaptation of feature detectors, G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov, 2012
Dropout: A Simple Way to Prevent Neural Networks from Overfitting, G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov, 2014

Stopping training at the right time : early stopping

Goals :

1. Prevents overfitting
2. Optimizes training efficiency
3. Works with other regularization methods



Stopping criteria :

1. Validation loss or performance metrics
2. Patience

```python
from tensorflow.keras.callbacks import EarlyStopping

early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=5,
    restore_best_weights=True
)
```

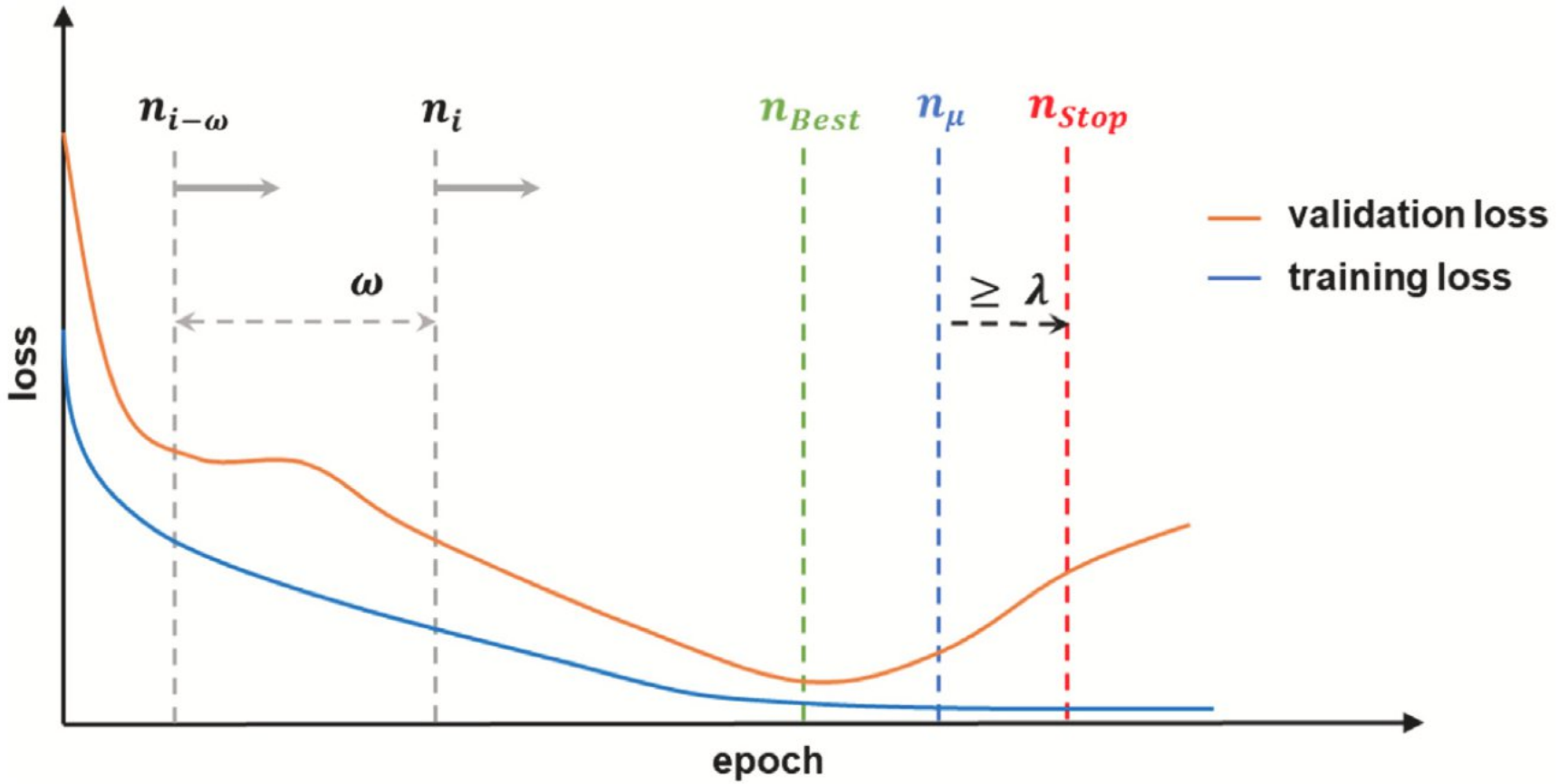
Code library : Keras early stopping

Correlation-Driven Stopping Criterion (CDSC)

Monitor rolling Pearson correlation coefficient of the training and validation losses

Improvements from early stopping :

More robust to noise
Captures overfitting more accurately
Adaptive stopping instead of fixed patience



Paper : Surpassing early stopping: A novel correlation-based stopping criterion for neural networks, Tamás Miseta, Attila Fodor, Ágnes Vathy-Fogarassy, 2023

L1 and L2 regularization

At model level

L1 regularization (Least Absolute Shrinkage and Selection Operator - LASSO)

$$J_{L1} = J + \lambda_1 \sum_j |\theta_j|$$

Encourages sparsity

L2 regularization (Ridge)

$$J_{L2} = J + \lambda_2 \sum_j \theta_j^2$$

Reduces influence but keeps all weights

L1-L2 regularization (Elastic net)

$$J_{L1-L2} = J + \lambda_1 \sum_j |\theta_j| + \lambda_2 \sum_j \theta_j^2$$

Mix of both LASSO and ridge

```
layer_lasso = keras.layers.Dense(32, kernel_regularizer=keras.regularizers.l1(0.01))
layer_ridge = keras.layers.Dense(32, kernel_regularizer=keras.regularizers.l2(0.01))
layer_elasticnet = keras.layers.Dense(32, kernel_regularizer=keras.regularizers.l1_l2(l1=0.01, l2=0.01))
layer_orthogonal = keras.layers.Dense(32, kernel_regularizer=keras.regularizers.orthogonal(factor=0.01))
```

Regularization at 3 levels : kernel_regularizer, biais_regularizer and activity_regularizer

Code library : Keras L1-L2 regularization

Batch normalization

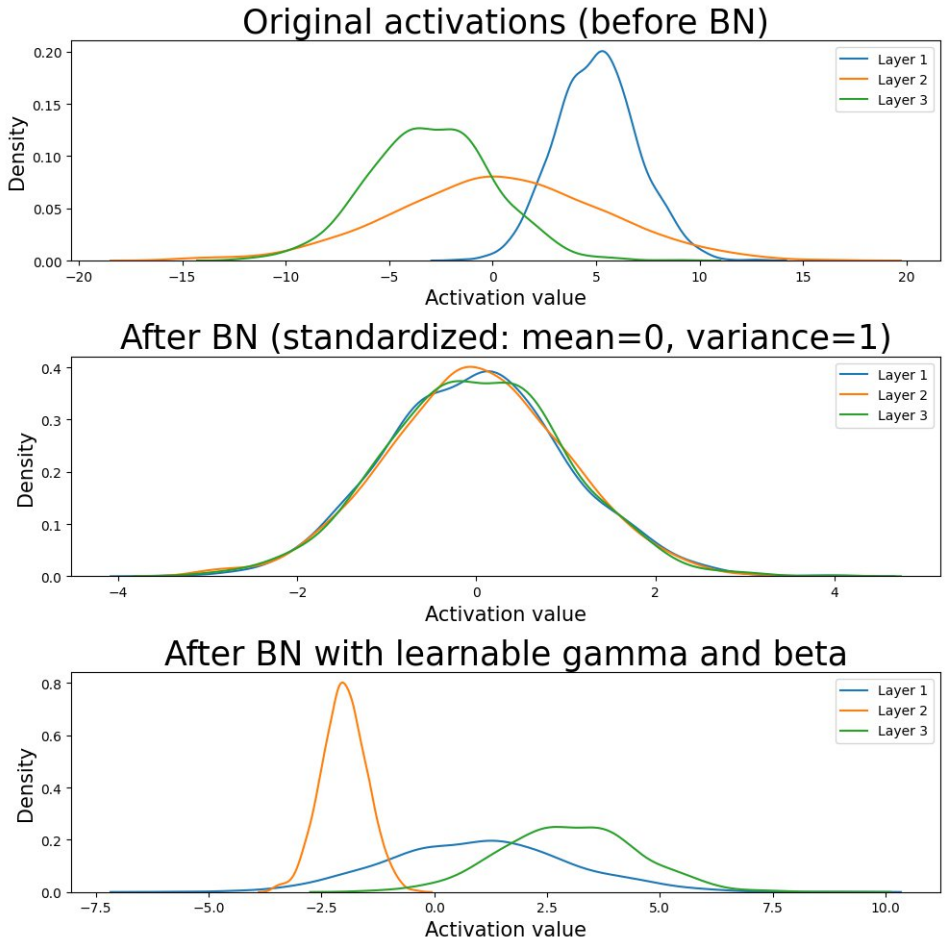
At layer level

1. Batch mean $$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$$

2. Batch variance $$\sigma^2{}_B = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_B)^2$$

3. Normalize $$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B{}^2 + \varepsilon}}$$

4. Scale and shift $$y_i = \gamma \hat{x}_i + \beta$$




```python
from tensorflow.keras.layers import Dense, BatchNormalization, Activation

model = Sequential([
    Dense(64),
    BatchNormalization(),
    Activation('relu'),
])
```

Paper : Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, Sergey Ioffe, Christian Szegedy, 2015
Code libraries : Keras batch normalization, Pytorch batch normalization

Weight normalization

At neuron level

Goals :
prevent large weigh values
and stabilize training

MaxNorm

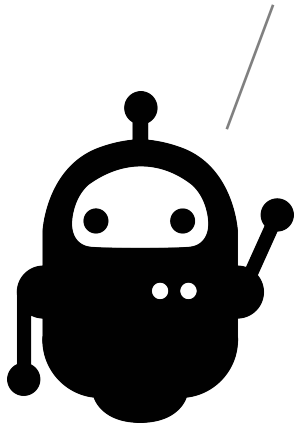
$$\theta_{MaxNorm} = \theta \frac{\tau}{\|\theta\|_2} \quad if \ \|\theta\|_2 > \tau$$

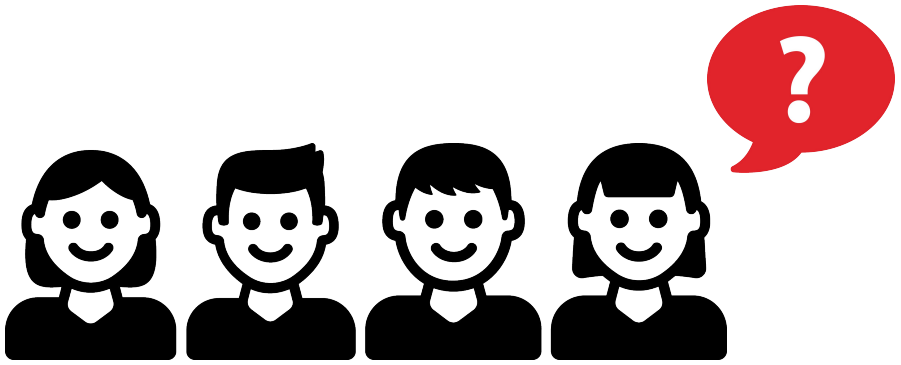
$$\theta_{MaxNorm} = \theta \quad if \ \|\theta\|_2 \leq \tau$$

Other weight constraints : MinMaxNorm and NonNeg

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.constraints import max_norm
model = Sequential([
    Dense(
        128, activation='relu',
        kernel_constraint=max_norm(3.0), input_shape=(64,)
    ),
    Dense(64, activation='relu', kernel_constraint=max_norm(3.0)),
    Dense(10, activation='softmax')
])
```

Quelques questions ?





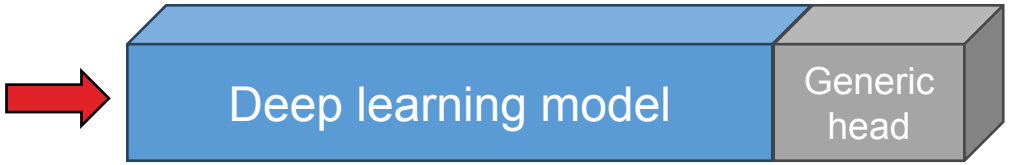
Model compression and efficiency

FIDLE

Transfert learning

Goal : transfer knowledge learned from one task (large dataset) to a specific task (smaller dataset)

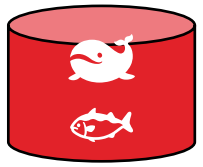
Training dataset (generic)



Rainbow 0.4 - Star 0.2 …
Star 0.6 - Flower 0.1 …
Star 0.8 - Rainbow 0.1 …
…

Knowledge transfert

Training dataset (specific)

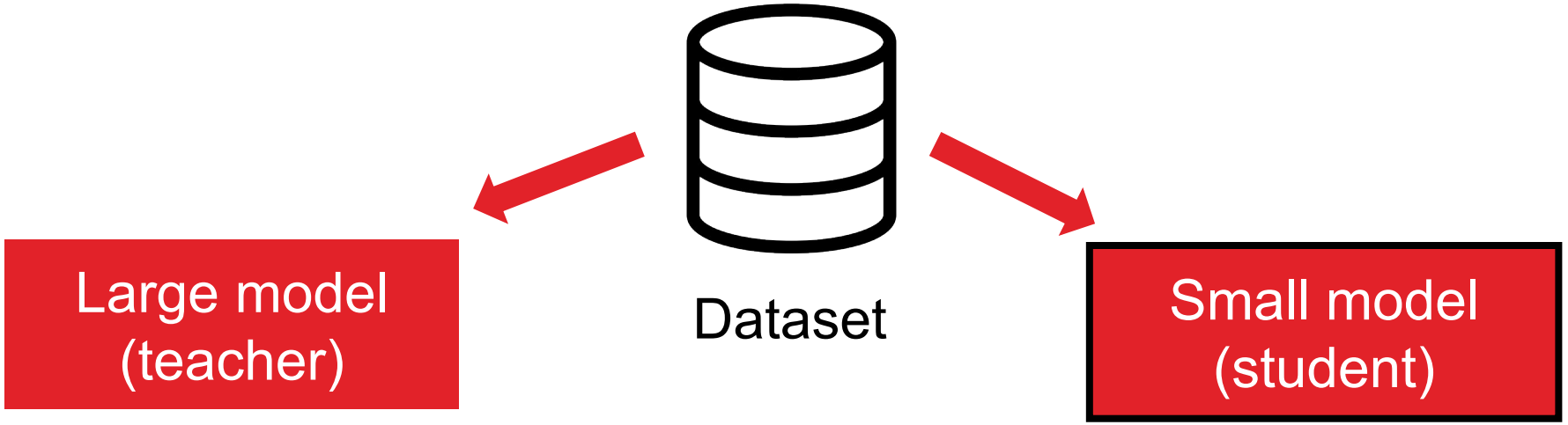


Whale 0.8 - Fish 0.2
Whale 0.6 - Fish 0.4
Whale 0.2 - Fish 0.8
…

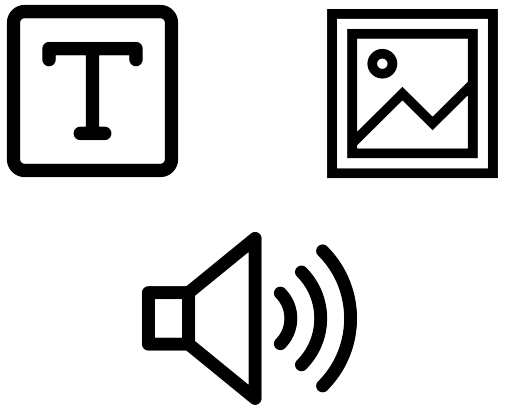
```
base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
base_model.trainable = False
x = GlobalAveragePooling2D()(base_model.output)
output_layer = Dense(1, activation='sigmoid')(x)
```

Knowledge distillation

Goal : transfer knowledge from a large model (teacher) into a smaller model (student)



Use cases :



$$Loss = (1-\alpha)\, L_{distillation} + \alpha L_{true\_labels}$$

Soft loss
(KL divergence)

Hard loss
(Cross entropy)

Temperature is used to soften the probabilities of the teacher model

Image credits : Chenyu Wang (database), SAADI ALA (text), Arthur Shlain (image), abderraouf omara (sound)

Paper : Distilling the Knowledge in a Neural Network, Geoffrey Hinton, Oriol Vinyals, Jeff Dean, 2015

Pruning

Goal : reduce the size of a trained model by removing unimportant weights, neurons or layers

```python
import torch.nn as nn
import torch.nn.utils.prune as prune

linear = nn.Linear(5, 5)
prune.l1_unstructured(linear, name="weight", amount=0.5)
```

```
Original weights:
Parameter containing:
tensor([[ 3.1984e-02, -2.9528e-01,  3.1725e-03,  2.8164e-02, -3.7913e-01],
        [ 3.8295e-01, -8.7382e-02, -2.0893e-01,  1.3495e-01, -1.9247e-01],
        [ 1.5279e-01,  2.2738e-04,  3.3152e-01,  3.6572e-01, -2.9560e-01],
        [-1.9604e-02,  1.0007e-01, -2.1878e-01, -2.0462e-02,  8.8720e-02],
        [-3.6335e-02,  1.6572e-01, -1.7977e-01,  2.0019e-01, -2.4078e-01]],
       requires_grad=True)

Pruned weights:
tensor([[ 0.0000, -0.2953,  0.0000,  0.0000, -0.3791],
        [ 0.3830, -0.0000, -0.2089,  0.0000, -0.1925],
        [ 0.0000,  0.0000,  0.3315,  0.3657, -0.2956],
        [-0.0000,  0.0000, -0.2188, -0.0000,  0.0000],
        [-0.0000,  0.1657, -0.1798,  0.2002, -0.2408]], grad_fn=<MulBackward0>)
```

Different **kinds** of pruning (magnitude pruning, gradient-based pruning…)

At different **levels** (weights, neurons, filters, channels…)

Trade-off between compression and accuracy

Code libraries : pruning with Keras, pruning with Pytorch

Quantization

Goal : reduce the memory size of a trained model by lowering the precision of its parameters

High precision

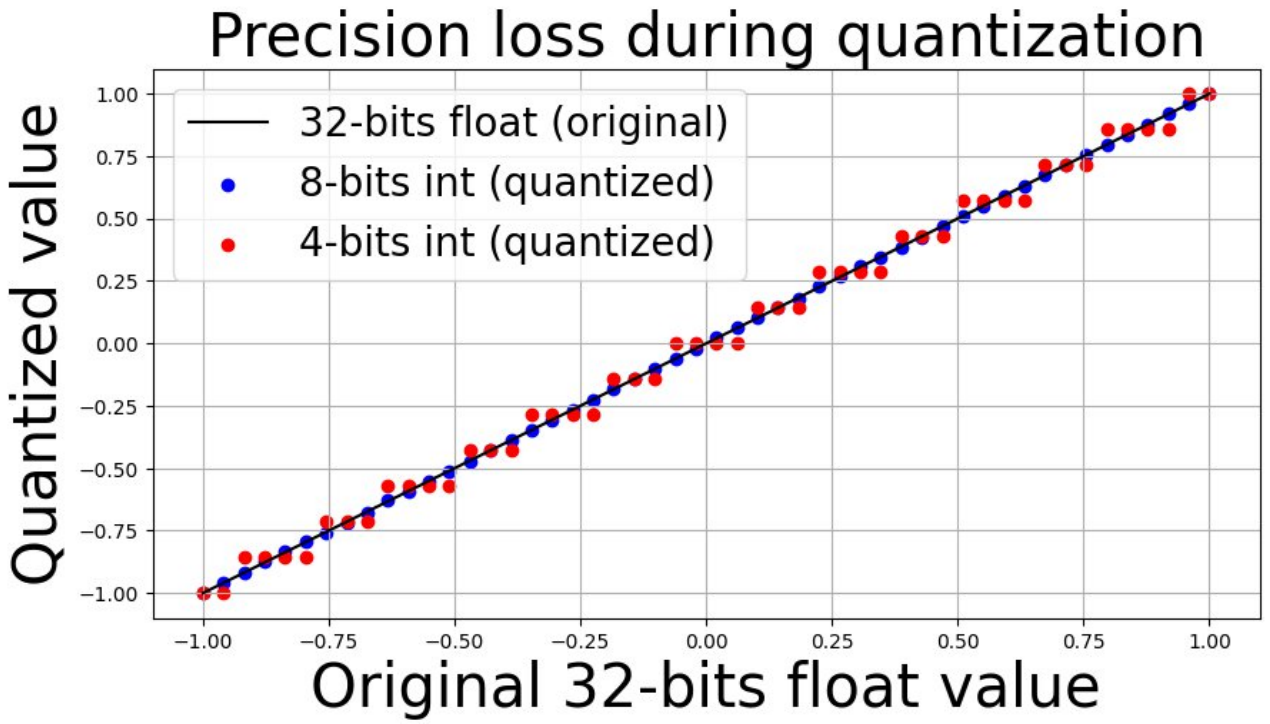
32-bits floats

16-bits floats

8-bits integers

4-bits integers

Low precision



After training (post-training quantization) or **during training** (quantization aware training)

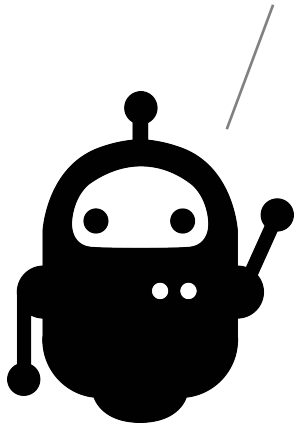
Linked to **hardware** (CPU, GPU, TPU, ASICs…)

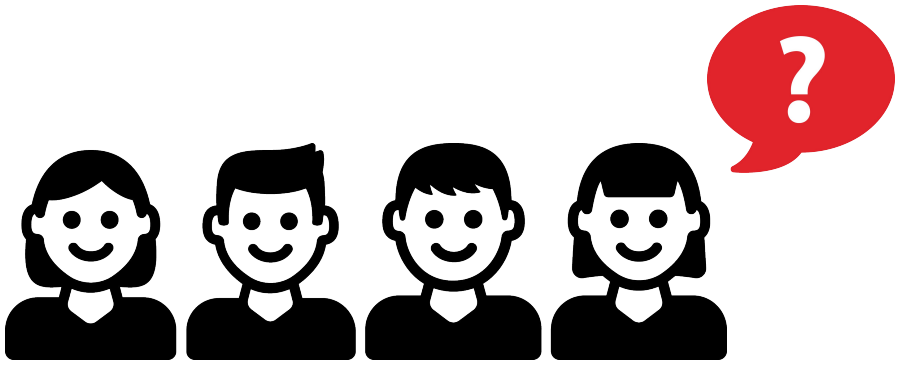
```
import tensorflow as tf

print(tf.keras.backend.floatx())

float32
```

Quelques questions ?

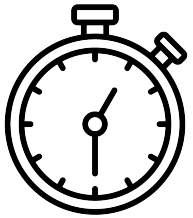




Conclusion

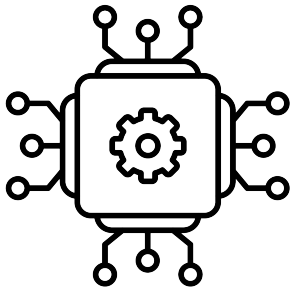
FIDLE

Machine learning : the art of trade-offs

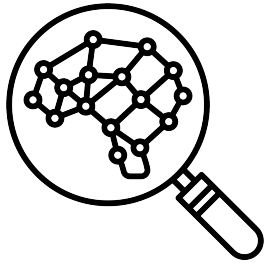


Time : between data and model improvement

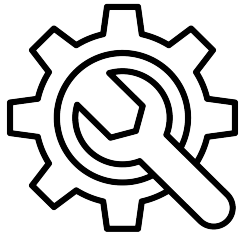
Metrics : between precision and recall or MAE and RMSE



Computational cost : between model complexity and efficiency



Interpretability : between explainability and predictive power



Technical : between bias and variance

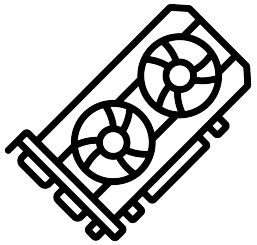
Image credits : Tamada (time), RAGIL HERU FITRIYAN (metrics), Soapi (compute) , iconpro86 (interpretability), dhanis (technical)

Learning optimisation : to infinity and beyond !

Next session about optimisation :

Session N°18 :
learning optimisation II (Advanced)

Wednesday, May 7, 2025, at 2:00 PM



Focused on hardware optimisation

Image credits : Akhmad Zaini (gpu), pojok_icon (advanced)

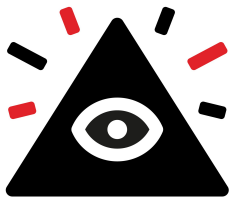
FIDLE

Jeudi 13 Mars 2025, 14h

Prochaine séquence :

12

L'IA comme un outil,



«Attention is All You Need»

RNN, Transformers

cnrs UGA Université Grenoble Alpes MIAI Grenoble Alpes EFELIA

GENCI ANITI DEVLOG RÉSEAU GROUPE CALCUL Resinfo cnrs IDRIS SIMaP