Distillation & Ray

Model Distillation

- Definition: process where a smaller, simpler model (student) is trained to mimic the behavior of a larger, more complex model (teacher)
- Goal: maintain high level of accuracy while reducing inference computational resources

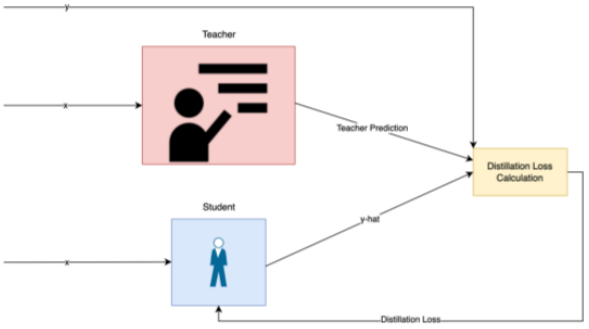
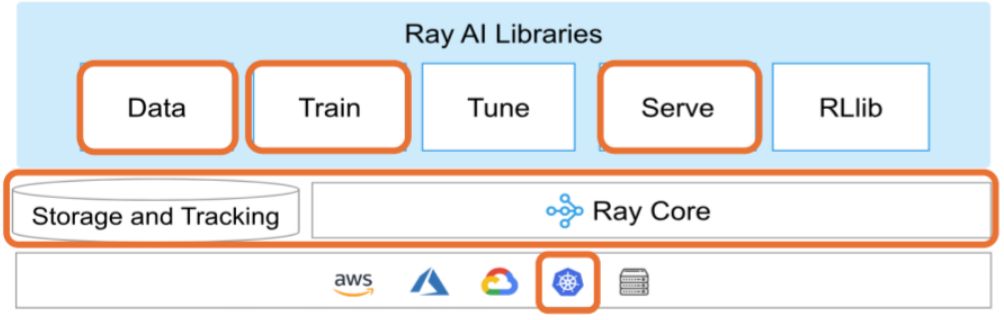


Illustration of one training iteration using distillation

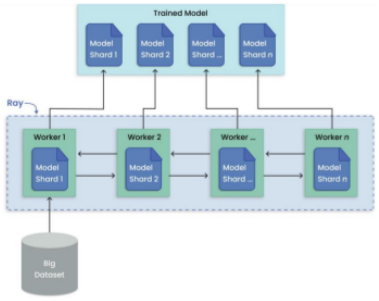
Ray

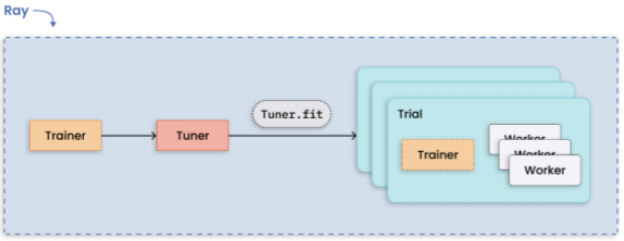
- Unified framework for scaling AI and Python applications
- Consists of a core distributed runtime and a set of AI libraries for simplifying ML compute



Ray

- Scalable
  - Allows developers to scale workloads from a single machine to a distributed cluster without changing code
- Distributed
  - Distributed architecture that manages efficiently resources and tasks -> ideal for large scale machine learning and data processing

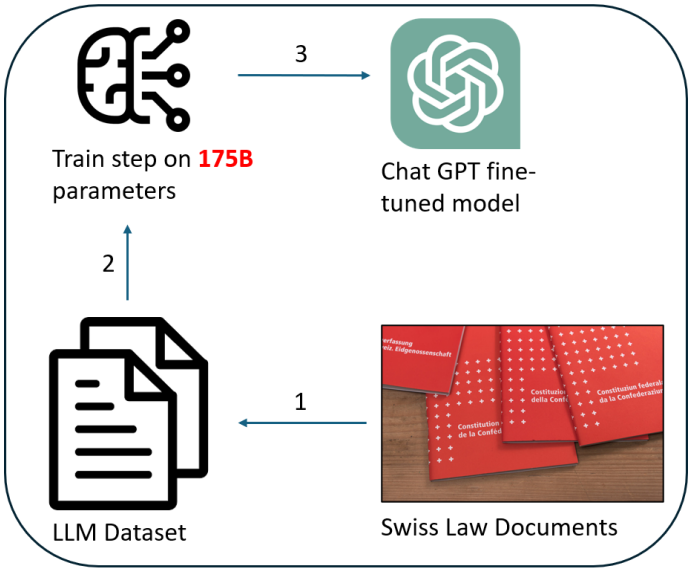




https://docs.ray.io/en/latest/ray-overview/use-cases.

LORA vs Adaptor vs RAG

1. Use case - LLMs problematic (1/2)



Without LoRA

- When fine-tune to a particular a task /domain

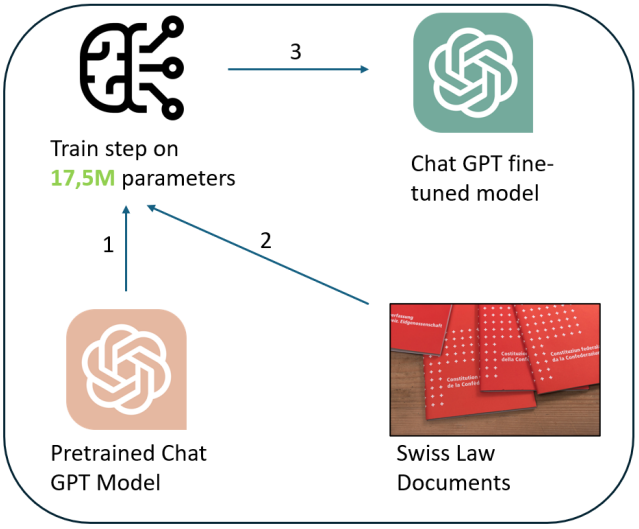
→ All parameters trained

→ Very expensive

1. Use case - LLMs problematic (2/2)

- Solutions → LLMs fine-tuning techniques
  - **LoRA**
  - Adapters
  - Retrieval Augmented Generation

- **LoRA**
  - Reduces **parameters** by up to **10,000x**
  - Requires **3x** less **GPU** resources



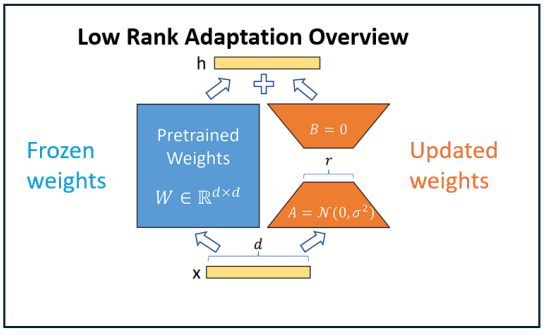
With LoRA

2. What's LoRA

- Low Rank Adaptation

- Inspired by Meta's 2020 research, introduced by Microsoft in 2021

- **Reduces** trainable **parameters** by adding **low-rank matrices to represent weight updates**
- while **keeping** the **original weight parameters frozen**, making **fine-tuning more efficient**..



3. LoRA vs. RAG & Adapters

FEATURE	LoRA	ADAPTERS	RAG
Benefits	- **Resource-efficient**: Reduces trainable parameters by up to **10,000x** - Uses **3x less GPU memory** - **Preserves core knowledge** by keeping main weights frozen during adaptation	**Flexible** and designed for **multitasking**, perfect for models that frequently switch between tasks	Provides **real-time access** by **integrating up-to-date data** and external data retrieval
Disadvantages	- **Complex to Implement** - Best suited for **small adjustments**, making it less ideal for major changes	Requires **more memory than LoRA** and may **lack real-time data** capabilities	**Complex integration**, as it relies on APIs or databases and is dependent on external sources
Best use case	**High-performance** adaptation in scenarios with limited resources	Versatile, **multi-task models** that require frequent task switching	Tasks requiring **current information** or **real-time updates**

Quantization, Canary Testing

Quantization

MSE | MASTER O IN ENGINE

Post-Training Quantization (PTQ)

Quantization-Aware Training (QAT)

Quantization

MASTER OF SCIENCE IN ENGINEERING

PTQ / QAT

-Dynamic (Weight in int8, activations in float32)
Offers a good balance between performance and precision.

-Float16 (Weight and activations in float16)
Optimised for GPUs.

-Integer (Weight and activations in int8)
Less precise in accuracy but more efficient in resource usage.

Canary Testing

MASTER OF SCIENCE IN ENGINEERING

What is Canary Testing?

- A strategy for releasing a new model version incrementally, starting with a small subset of traffic.
- Helps identify issues early, ensuring that the model performs as expected before full deployment.

Canary Testing

Why Canary Testing Matters ?

- Risk Reduction
- Performance Validation
- Control and Flexibility

Data storage & version control

LAKEFS

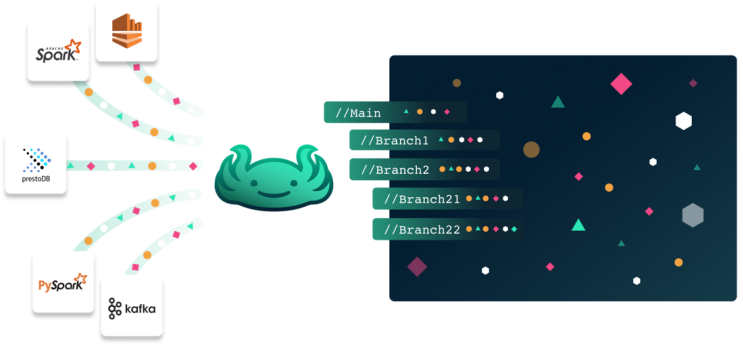
GIT-LIKE OPERATIONS FOR DATA LAKES

Key Features

- Atomic commits and rollbacks.
- Branching and merging.
- Scalable storage.

Challenge Addressed

- Volume & Variety
- Structure & Scaling
- Storage Optimization (deduplication)



DELTA LAKE

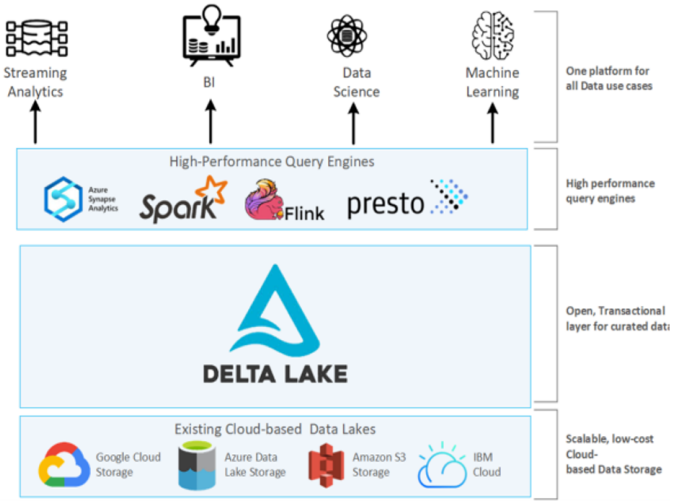
ACID TRANSACTIONS FOR DATA LAKES

Key Features

- ACID data transactions
- Batch and streaming with high-performance reads
- Schema enforcement and evolution
- Snapshots & Time Travel for historical data

Challenge Addressed

- Volume & Velocity
- Structure & Scaling
- Storage Optimization
- Security & Acces



DVC

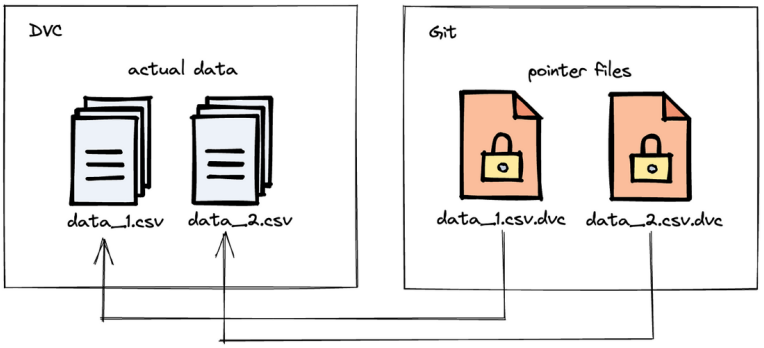
DATA VERSION CONTROL FOR ML PROJECTS

Key Features

- Tracks large files and datasets
- Reproducibility and collaboration

Challenge Addressed

- Structure
- Efficient storage management.



PACHYDERM

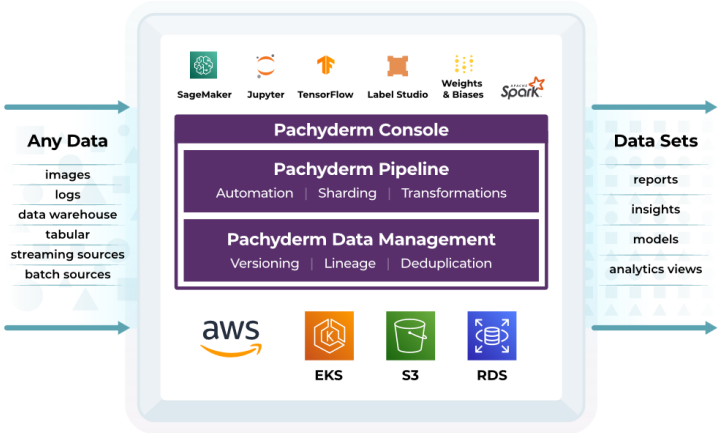
DATA LINEAGE WITH PIPELINE

Key Features

- Lineage tracking.
- Incremental processing.
- Role-based access control.

Challenge Addressed

- Veracity & Security
- Optimized storage.



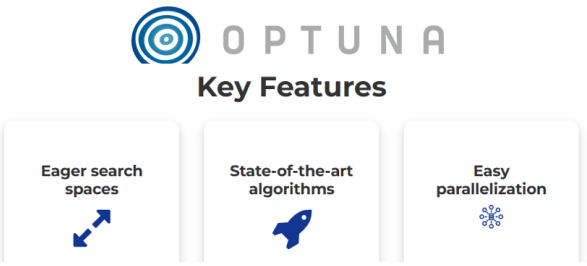
COMPARISON

Feature	LakeFS	Delta Lake	DVC	Pachyderm
Data Versioning	yes	yes	yes	yes
Scalability	high	high	medium	high
Git Integration	partial	no	full	no
Schema Enforcement	no	yes	no	no
Pipeline Support	no	limited	limited	yes
Access Control	yes	limited	no	yes
Github Stars	4.5k	7.5k	13.9k	6.2k

Hyper parameter optimization

TRAINING AND OPTIMIZATION

- Optuna
- Vector Support Machine (SVM)
  - **C** (e.g. 0.1, 10, …)
  - **Gamma** (e.g. scale, auto, …)
  - **Kernel** (e.g. linear, rbf, …)
  - …
- From scratch
- Training set grows overtime



Continuous learning, stability, plasticity

II. Concepts théoriques

A. Définition :

- Continuous Learning
- Inspirer sur l'apprentissage humaine
- Développer un système intelligent

B. Différence :

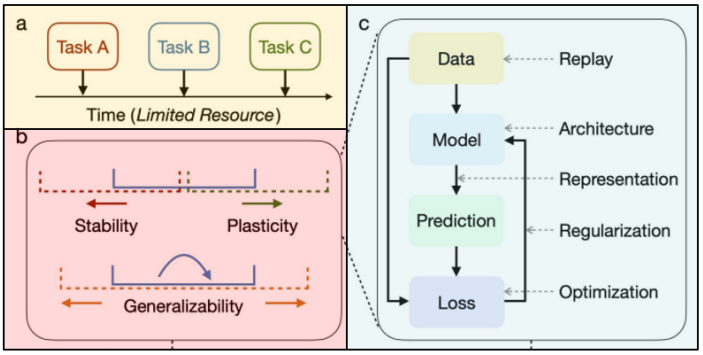
- Méthode traditionnelle -> difficile d'intégrer des nouvelles données
- Méthode continue -> adaptation dynamique de nouvelle donnée

C. Objectif :

- **Gérer efficacement les ressources**
- **Maintenir** un équilibre entre **stabilité** et **plasticité**

II. Concepts théoriques

Figure 2 : A conceptual framework of continual learning **[1]**



[1] Art. A Comprehensive Survey of Continual Learning: Theory, Method and Application

II. Contexte Data

A. Spécification par rapport à notre application

- Classification d'image
- Les données sont obtenues par les développeurs

B. Scénario

- Donnée limitée pour l'instant -> stratégie d'augmentation de donnée

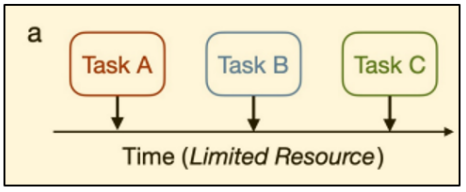


Figure 3 : Data

II. Méthodes

A. Replay

- **l'objectif** du replay est d'éviter l'oubli catastrophique
- **Collaboration :** 1 membre = 1 lot de 10 Hiragana

Contexte	Mémoire d'échantillons réels	Rejeu génératif
Avantages	▪ Simple et efficace	▪ Résout les problèmes de stockage
Inconvénients	▪ Stockage coûteux	▪ Générer des données réalistes pour certaines tâches peut être difficile.

Tableau 1: Méthodes replay

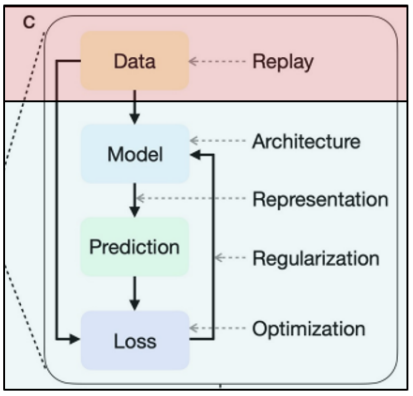


Figure 4 : Approches

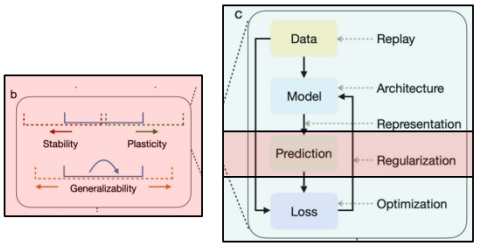
II. Régularisation

A. Problème central :

- Placidités excessive -> oublie des anciennes tâches
- Stabilité excessive -> limite l'apprentissage de nouvelle tâche

B. régularisation d'EWC :

- Une fonction de perte de régularisation qui aide à éviter que le modèle oublie les caractères déjà appris lorsqu'il en apprend de nouveaux.



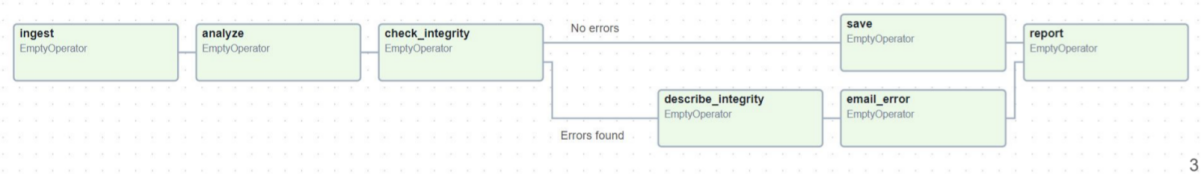
$$LEWC(\theta) = \boxed{Lk(\theta)} + \boxed{\frac{1}{2}\lambda \sum Fi(\theta i - \theta i^{*})}$$

Eq.1 : Fonction perte EWC

DAG & Workflow

Workflows

- Represented as a DAG - Directed Acyclic Graph
- Contains task
  - Individual pieces of work
  - Dependencies between tasks
  - Can be anything from a Python function to a SQL query
  - Possible to chain tasks together
  - Make decisions based on the result of a task - Branching
  - Run on dataset update - Data driven scheduling

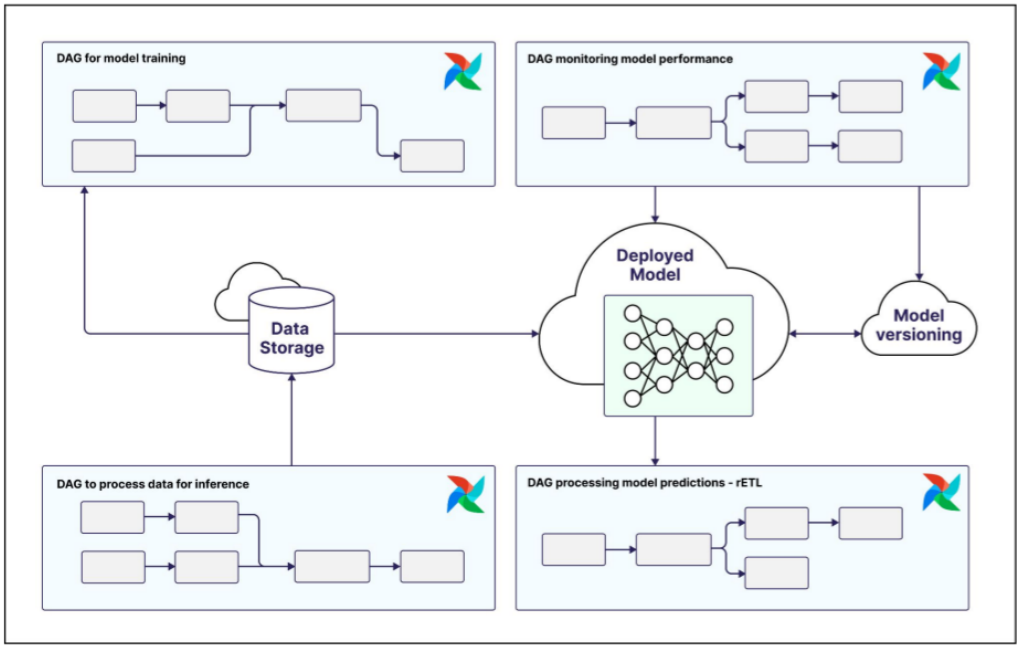


3

Airflow and Github Actions

	Apache Airflow	GitHub Actions
Workflows definition	DAGs	YAML configuration
Workflows triggering	Time-based, event-based, manually triggered, etc.	Code repository events (push, pull request, manually)
Execution environment	Customizable (docker, kubernetes, VMs, etc.)	Github runners (hosted or self-hosted)
Monitoring & UI	Rich web UI for DAGs, logs, task states, etc.	Simple web UI for job statuses and logs

Airflow pipeline example



Active Learning, Reinforcement Learning, Contrastive Learning

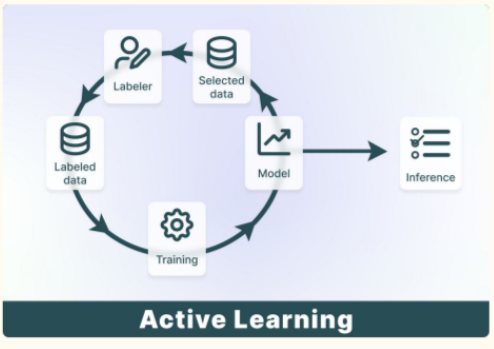
Advanced labeling - Theoretical concepts

Active Learning (Human in the Loop):

Reduce the number of manually labeled data entries by selecting samples with the highest “uncertainty”.

Process:

- **Initial Training**: Start with a small set of labeled data.
- **Uncertainty Sampling**: For new, unlabeled data, the model identifies the most uncertain examples—those with the least confident predictions.
- **Human-in-the-Loop**: These uncertain examples are passed to human for review, who provide the correct labels.
- **Retraining:** The model is re-trained using the newly labeled examples to improve its accuracy.



Source: https://dagshub.com/blog/active-learning-with-domain-experts-a-case-study/

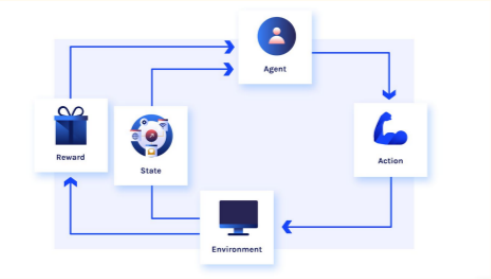
Advanced labeling - Theoretical concepts

Reinforcement Learning from human feedback:

An agent learns from human-provided feedback, improving its decisions and enhancing model performance with minimal manual effort.

Process:

- **Agent Initialization**: Set up the agent to work with unlabeled data.
- **Define Actions**:
  a. Suggest a label or prediction.
  b. Request human feedback for clarification.
  c. Skip label if uncertain.
- **Engage with Data**: The agent decides how to act based on its confidence or learned policy.
- **Gather Feedback**: Humans validate or correct the agent's decisions.
- **Update Policy**: Use feedback to improve decision-making strategies.
- **Reward System**: Actions improving performance earn rewards, reinforcing better behavior.
- **Iteration**: Repeat until the agent meets human-defined performance goals.



Source: https://datatonic.com/insights/reinforcement-learning-identifying-opportunities-use-cases/

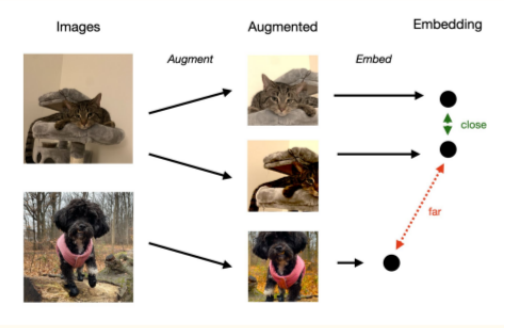
Advanced labeling - Theoretical concepts

Contrastive Learning:

Learn the general features of a dataset without labels by teaching the model which data points are similar or different..

Process:

- **Data Augmentation**: creating multiple views of the same input (crop, flip, color modifications, noise...)
- **Encoder Network**: input data (and augmented data) into a CNN to get vectors
- **Contrastive Loss Function**: show similarity between samples
- **Negative Sampling**: include "negative" examples



Source: https://www.rtealwitter.com/deeplearning2023/contrastive.html

FAAS & Limitations

What is FaaS?

Function as a Service (FaaS)
Cloud service model that allows developers to write functions and deploy them in a serverless environment, where the cloud provider handles scaling, execution, and management.

Limitation

01
Stateless
Model and data need to be stored in external buckets
Pay per access, read and upload

02
Cold start
Starting container : seconds

03
Resource constraint
Time limit : ms to seconds
Ram : 1 - 10 Go