L'écoconception de services numériques

Écoconception et IA

éCOconception

ECO-TIC

design numérique responsable

conception et développement logiciel

logiciel Eco-responsable

Spécificités et enjeux liés à l'Intelligence Artificielle

Green IT

TIC et développement durable

Dans ce document ce sigle signifie qu'il s'agit d'une préconisation « écoconception »

L'écoconception de services numériques
Plan

- **Où se situe l'impact ?**
  - ✓ Centres de calcul
  - ✓ Ordres de grandeur (GWh / Wh)
  - ✓ Données chiffrées entraînement – requête – infrastructure
- **Principe clé : sobriété par l'usage**
  - ✓ Juste assez d'intelligence
- **Objectifs écologiques et leviers**
  - ✓ Où se situent les leviers d'écoconception de l'IA ?
  - ✓ Tableau Objectif → Outils
  - ✓ Recommandations et checklist
- **Optimiser le modèle**
  - ✓ Pruning, quantization, distillation
- **Optimiser le déploiement**
  - ✓ Simplification de graphes et conversion de modèles

L'écoconception de services numériques

Centres de calcul pour l'IA : impacts environnementaux

Rôle central des centres de calcul IA

- Les centres de calcul dédiés à l'intelligence artificielle constituent aujourd'hui le **nœud principal de l'empreinte environnementale de l'IA**.

- Ils concentrent les phases les plus intensives en ressources :
  - ✓ entraînement de modèles de grande taille,
  - ✓ exécution massive de requêtes (inférence),
  - ✓ stockage et circulation de volumes importants de données.

- Contrairement aux datacenters traditionnels, les centres IA s'appuient largement sur des **accélérateurs matériels** (GPU, TPU, NPU), dont la densité de calcul et la consommation énergétique sont très élevées.

L'écoconception de services numériques

Consommation énergétique de l'entraînement de modèle d'IA

- L'entraînement d'un modèle d'IA de grande taille représente une **dette énergétique initiale importante :**

- Des études académiques et industrielles montrent que :
  - ✓ Une étude de l'université du Massachusetts (2019) a montré que l'entraînement d'un seul modèle pouvait générer autant de **CO_2 qu'une voiture sur plus de 500 000 km.**
  - ✓ L'entraînement d'un modèle de type **Transformer de grande taille** peut consommer **de l'ordre de 1 à plusieurs dizaines de GWh** d'électricité.
  - ✓ À titre de repère, **1 GWh** correspond à la consommation annuelle d'environ **200 à 300 foyers européens**.

- Cette consommation dépend fortement :
  - ✓ du nombre de paramètres du modèle,
  - ✓ du nombre d'itérations d'entraînement,
  - ✓ du rendement énergétique du matériel utilisé.

👉 L'entraînement est généralement **ponctuel**, mais son coût environnemental est élevé et doit être **amorti par la durée de vie et l'utilité réelle du modèle**.

L'écoconception de services numériques

Données chiffrées sur l'entraînement de GPT

Phase d'entraînement (training)

- L'entraînement d'un grand modèle comme **GPT-4** a consommé plus de **50 gigawattheures (GWh)** d'électricité, soit l'équivalent de la consommation électrique de **San Francisco pendant ~3 jours**.

- À titre de comparaison, l'entraînement de **GPT-3** avait nécessité environ **1,3 GWh**.

- Ces chiffres montrent que l'entraînement constitue une **dette énergétique initiale importante**, même si elle reste ponctuelle.

👉 Un gigawattheure (GWh) = 1 000 000 kWh

Consommation moyenne d'un foyer français ~ 4 250 kWh/an

1GWh ~ consommation de 235 foyers français

L'écoconception de services numériques

Coût énergétique d'une requête IA

L'inférence – c'est-à-dire l'utilisation du modèle en production – est aujourd'hui la **source dominante de consommation énergétique**, du fait de la massification des usages.

- Les estimations disponibles indiquent que :
  - ✓ Les émissions associées sont de l'ordre de **quelques dixièmes à quelques grammes de CO_2 par requête**, selon le mix électrique du datacenter.
  - ✓ Une requête IA génère **plusieurs grammes de CO_2** (par exemple ~2–4 g CO_2/query), soit **plusieurs fois l'impact d'une recherche web classique** (~0,2 g CO_2).
  - ✓ Une requête de génération de texte par un modèle de type LLM consomme typiquement **entre 0,3 et 1 Wh**.

À titre de comparaison :

- ✓ une requête web classique consomme environ **0,05 à 0,1 Wh**,
- ✓ une requête IA peut donc consommer **5 à 20 fois plus**.

👉 Individuellement, l'impact d'une requête est faible, mais **le cumul de millions ou de milliards de requêtes quotidiennes** devient significatif à l'échelle globale et L'inférence représente de l'ordre de **60 % à 70 % de toute l'énergie consommée par un modèle sur son cycle de vie**.

L'écoconception de services numériques

Infrastructures IA, refroidissement et efficacité énergétique

- Les centres de calcul IA sont confrontés à des contraintes thermiques fortes :
  - ✓ Les GPU/TPU fonctionnent à pleine charge sur de longues durées.
  - ✓ Le refroidissement peut représenter une part importante de la consommation totale.

- L'efficacité globale est souvent mesurée par le **PUE (Power Usage Effectiveness)** :
  - ✓ PUE ≈ 1,0 : efficacité optimale,
  - ✓ PUE > 1,5 : pertes importantes liées au refroidissement et aux infrastructures.

- Les centres IA les plus récents intègrent :
  - ✓ refroidissement liquide,
  - ✓ optimisation de la circulation d'air,
  - ✓ parfois récupération de chaleur,
  - ✓ afin de limiter le surcoût énergétique indirect.

L'écoconception de services numériques

Gradation des solutions : du plus simple au plus intelligent

Le principe de sobriété par l'usage repose sur une hiérarchie de solutions :

1. **Règles métier ou logique déterministe**
   - ✓ Si la tâche est stable, explicite et prévisible.
2. **Algorithmes classiques**
   - ✓ Classification simple, seuils, statistiques.
3. **Modèles d'apprentissage légers**
   - ✓ Régression, arbres de décision, petits réseaux.
4. **IA avancée (deep learning, LLM)**
   - ✓ Uniquement lorsque les niveaux précédents sont insuffisants.

👉 Le recours à une IA complexe doit être **justifié**, et non présumé nécessaire.

L'écoconception de services numériques
Plan

- **Où se situe l'impact ?**
  - ✓ Centres de calcul
  - ✓ Ordres de grandeur (GWh / Wh)
  - ✓ Données chiffrées entraînement – requête – infrastructure
- **Principe clé : sobriété par l'usage**
  - ✓ Juste assez d'intelligence
- **Objectifs écologiques et leviers**
  - ✓ Entraînement vs inférence
  - ✓ Tableau Objectif → Outils
  - ✓ Recommandations et checklist
- **Optimiser le modèle**
  - ✓ Pruning, quantization, distillation
- **Optimiser le déploiement**
  - ✓ Simplification de graphes et conversion de modèles

L'écoconception de services numériques

Sobriété par l'usage

- Une IA écoconçue n'est pas celle qui sait tout faire, mais celle qui **fait exactement ce qu'il faut, quand il le faut, et seulement quand il le faut**.

- La sobriété par l'usage :
  - ✓ Éviter les fonctionnalités IA « gadgets » déclenchées par défaut,
  - ✓ Donner à l'utilisateur le contrôle explicite de l'activation de l'IA,
  - ✓ Ne pas multiplier les suggestions ou reformulations automatiques inutiles,
  - ✓ Désactivation automatique des modèles peu utilisés,
  - ✓ Mise en cache des résultats fréquents,
  - ✓ Seuils de déclenchement (ex. : appel IA seulement si l'incertitude dépasse un seuil),
  - ✓ Réduction de la longueur des entrées/sorties (prompts, tokens générés)…

👉 *La sobriété par l'usage est souvent le levier le plus efficace et le moins coûteux pour réduire l'empreinte environnementale de l'IA.*

L'IA la plus sobre est souvent celle qu'on n'appelle pas.

L'écoconception de services numériques

IA responsable et soutenable

Une IA durable ne se définit pas seulement par sa sobriété énergétique, mais aussi par sa **valeur sociétale** et sa **transparence**.

Les grands axes de soutenabilité sont :

- **Transparence et explicabilité** : comprendre les décisions d'un modèle permet d'éviter le surdimensionnement des architectures "boîtes noires" et de renforcer la confiance, donc la longévité des systèmes.
- **Sobriété des données** : privilégier la qualité à la quantité, réduire la collecte massive, valoriser les jeux de données locaux ou synthétiques, et réutiliser les données existantes.
- **Équité et accessibilité** : une IA plus légère et économe peut être exécutée sur des infrastructures modestes, ouvrant la voie à une **démocratisation technologique** compatible avec des contraintes écologiques.
- **Éthique et gouvernance** : la durabilité de l'IA passe par la prise en compte des externalités sociales et environnementales dans les choix d'architecture, de partenaires cloud ou de modèle d'affaires.

Ainsi, une IA écoconçue doit être envisagée comme **un système socio-technique soutenable,** équilibrant **trois dimensions** :

- la **performance fonctionnelle** (répondre au besoin réel),
- la **sobriété environnementale** (minimiser les ressources),
- et la **responsabilité sociale** (transparence, accessibilité, équité).

L'écoconception de services numériques
Plan

- **Où se situe l'impact ?**
  - ✓ Centres de calcul
  - ✓ Ordres de grandeur (GWh / Wh)
  - ✓ Données chiffrées entraînement – requête – infrastructure
- **Principe clé : sobriété par l'usage**
  - ✓ Juste assez d'intelligence
- **Objectifs écologiques et leviers**
  - ✓ Où se situent les leviers d'écoconception de l'IA ?
  - ✓ Tableau Objectif → Outils
  - ✓ Recommandations et checklist
- **Optimiser le modèle**
  - ✓ Pruning, quantization, distillation
- **Optimiser le déploiement**
  - ✓ Simplification de graphes et conversion de modèles

L'écoconception de services numériques

Où se situent les leviers d'écoconception de l'IA ?

- **Plusieurs types de leviers**, qui n'ont **ni le même poids environnemental, ni la même pertinence selon les phases du cycle de vie** d'un modèle.

- Ces leviers peuvent concerner :
  - ✓ les **choix de conception** (nécessité de l'IA, niveau d'intelligence),
  - ✓ les **données** utilisées,
  - ✓ l'**entraînement** du modèle,
  - ✓ l'**optimisation du modèle**,
  - ✓ le **déploiement** et l'**exploitation**.

- **Les décisions prises en amont (conception, usage)** ont un impact bien plus important que les optimisations techniques réalisées a posteriori.
- Les optimisations bas niveau restent néanmoins nécessaires, notamment lorsque les modèles sont utilisés à grande échelle.

Ces leviers doivent donc être mobilisés différemment selon la phase du projet.

L'écoconception de services numériques

Quand faire quoi, avec quels leviers et outils ?

Apres avoir listé les grands leviers de l'écoconception de l'IA, ce tableau propose une grille de lecture opérationnelle, permettant d'identifier quoi faire concrètement, à quelle étape, et avec quels outils.

Phase	Question clé	Leviers prioritaires	Exemples d'outils
Conception	Ai-je besoin d'IA ?	Sobriété par l'usage	Règles métier
	Quel niveau d'intelligence ?	Modèle simple vs complexe	—
Données	Ai-je trop de données ?	Réduction, qualité	Sampling
Entraînement	Coût acceptable ?	Transfer learning, early stopping	TensorFlow, PyTorch
Modèle	Modèle surdimensionné ?	Pruning, distillation	TF Model Optimization
Déploiement	Coût par requête trop élevé ?	Quantization, simplification de graphe	ONNX, TensorRT
Exploitation	Trop d'appels inutiles ?	Cache, seuils d'activation	UX responsable

L'écoconception de services numériques

Leviers d'écoconception pour l'IA vers une IA frugale

Réduire l'impact environnemental de l'IA implique d'agir sur tout le cycle de vie du modèle : analyse, conception, entraînement, déploiement et usage.

Plusieurs stratégies complémentaires se dégagent :

- **Optimiser la taille et la complexité des modèles** : techniques de ***pruning*** (élagage de neurones inutiles), de ***quantization*** (réduction de la précision numérique) ou de ***distillation de connaissances*** (entraîner un modèle léger à partir d'un modèle plus grand) permettent de diminuer drastiquement les calculs sans perte notable de performance.
- **Déplacer l'intelligence vers la périphérie** : l'***Edge AI*** et le ***TinyML*** privilégient l'exécution locale sur des microcontrôleurs ou appareils embarqués, réduisant la dépendance au cloud et les coûts de transfert de données.
- **Choisir un matériel et un environnement sobres** : certaines architectures matérielles (**GPU basse consommation**, **NPU spécialisés**) et datacenters alimentés en énergies renouvelables réduisent l'impact global.
- **Mesurer et piloter la consommation** : outils comme ***CodeCarbon***, ***Experiment Impact Tracker*** ou ***Carbontracker*** permettent de suivre en temps réel la dépense énergétique d'un modèle et d'en déduire des indicateurs d'efficacité.

👉 Il s'agit de passer d'une logique de "**plus de données, plus de calcul**" à une logique de **"juste assez d'intelligence pour le besoin".**

L'écoconception de services numériques

Recommandations d'écoconception sur les modèles IA

L'écoconception appliquée à l'intelligence artificielle ne consiste pas seulement à "faire consommer moins" : Elle incite à une intelligence plus sobre, distribuée, réutilisable et responsable, alignée sur la maîtrise de son empreinte en considérant la pertinence de ses usages.

Recommandations aux développeurs :

- Ne pas entraîner un modèle si cela ne se justifie pas.
- Tester des modèles plus petits avant de passer au "big model".
- Suivre en temps réel la consommation énergétique.
- Évaluer la performance par rapport au coût énergétique.
- Privilégier les architectures simples et localement pertinentes.
- Utiliser des données justes et utiles, pas massives.
- Réduire la taille et le nombre d'itérations d'apprentissage.
- Déployer sur des serveurs ou clouds verts.
- Surveiller l'impact en production.
- Documenter, expliquer et partager les bonnes pratiques.

L'écoconception de services numériques

Checklist - Avant le développement - Phase d'Analyse

Pertinence et cadrage du besoin :

Objectif	Recommandations
Définir le besoin réel	- Le recours à l'IA est-il réellement justifié ?- Peut-on atteindre le même objectif avec un algorithme plus simple ou des règles métier ?
Cibler la granularité fonctionnelle	- Identifier précisément la tâche automatisée (classification, détection, …).- Éviter les modèles "généralistes" surdimensionnés pour un cas d'usage restreint.
Anticiper les impacts environnementaux et sociétaux	- Évaluer la fréquence d'usage, le volume de données, le matériel nécessaire. Considérer l'impact énergétique, la durée de vie du modèle et son usage social.
Choisir des objectifs multi-critères	- Inclure un critère de sobriété énergétique ou de coût carbone dans les KPI du projet.- Arbitrer entre performance et efficacité environnementale.

L'écoconception de services numériques

Phase de conception du modèle d'IA

Objectif	Recommandations
Favoriser la réutilisation	- Utiliser des modèles pré-entraînés, des datasets existants.- Privilégier le transfer learning ou la fine-tuning ciblée.
Limiter la taille et la complexité	- Choisir une architecture adaptée au problème (pas de modèle "géant" inutile).- Optimiser les modèles d'IA pour réduire les calculs.
Optimiser le pipeline d'apprentissage	- Échantillonnage intelligent des données.- Réduction du nombre d'époques d'entraînement.- Suivi des métriques d'énergie (via CodeCarbon, CarbonTracker…).
Choisir un matériel économe	- Préférer les GPU/NPU sobres ou le calcul sur cloud vert.- Localiser le datacenter dans une région à faible intensité carbone.
Prévoir la modularité et la maintenance	- Développer des modèles pouvant être mis à jour sans réentraîner complètement le système.- Documenter les choix d'architecture et les dépendances.

L'écoconception de services numériques

Phase d'entraînement

Objectif	Recommandations
Mesurer pour maîtriser	- Estimer et consigner la consommation électrique de chaque expérimentation.- Suivre la métrique kWh/epoch ou gCO_2/epoch.
Limiter les itérations inutiles	- Planifier une stratégie expérimentale (design of experiments).- Arrêter l'entraînement dès la convergence (early stopping).
Mutualiser les ressources	- Utiliser des plateformes de calcul partagées.- Planifier les entraînements en heures creuses pour limiter la charge réseau/électrique.
Optimiser les formats de données	- Compresser, normaliser et indexer les jeux de données.- Nettoyer les doublons ou données non pertinentes avant l'apprentissage.

L'écoconception de services numériques

Phase de déploiement

Objectif	Recommandations
Optimiser le modèle en production	- Convertir le modèle pour l'inférence (TensorRT, ONNX, TFLite…).- Supprimer les couches ou paramètres inutiles.
Réduire le coût d'inférence	- Ajuster la fréquence d'appel des modèles.- Mettre en cache les résultats récurrents.- Exécuter en batch plutôt qu'en temps réel si possible.
Utiliser l'Edge ou le cloud vert	- Privilégier le Edge AI ou les datacenters bas-carbone.- Éviter les transferts de données massifs entre serveurs et clients.
Mettre en place un suivi d'usage	- Suivre les métriques d'énergie, de latence et de charge serveur.- Détecter les dérives de performance pour éviter un réentraînement inutile.

L'écoconception de services numériques

Phase d'exploitation

Objectif	Recommandations
Documenter et expliciter	- Fournir une fiche Model Card (origine des données, performances, limites, empreinte carbone estimée).
Garantir la justice et la non-discrimination	- Auditer les biais du jeu de données et des résultats.- Favoriser la diversité et la représentativité des données.
Favoriser la durabilité logicielle	- Rendre le code ouvert, réutilisable et versionné (GitHub, licences libres).- Documenter la méthodologie pour faciliter la reprise et la maintenance.
Sensibiliser les parties prenantes	- Former les développeurs, data scientists et décideurs à la sobriété numérique.- Intégrer les critères RSE et environnementaux dans les revues de projet.

L'écoconception de services numériques
Plan

- **Où se situe l'impact ?**
  - ✓ Centres de calcul
  - ✓ Ordres de grandeur (GWh / Wh)
  - ✓ Données chiffrées entraînement – requête – infrastructure
- **Principe clé : sobriété par l'usage**
  - ✓ Juste assez d'intelligence
- **Objectifs écologiques et leviers**
  - ✓ Où se situent les leviers d'écoconception de l'IA ?
  - ✓ Tableau Objectif → Outils
  - ✓ Recommandations et checklist
- **Optimiser le modèle**
  - ✓ Pruning, quantization, distillation
- **Optimiser le déploiement**
  - ✓ Simplification de graphes et conversion de modèles

L'écoconception de services numériques

Suppression des couches ou paramètres inutiles

Avant ou pendant la conversion, on peut encore simplifier le graphe computationnel :

- ✓ Retirer les couches dédiées à l'entraînement (dropout, batchnorm inutiles, etc.) ;
- ✓ Supprimer les variables intermédiaires non utilisées en inférence ;
- ✓ Figer les paramètres constants pour éviter leur recalcul ;
- ✓ Fusionner les couches redondantes (par exemple convolution + activation) ;
- ✓ Supprimer les paramètres faiblement contributifs.

👉 **La suppression des couches ou paramètres non pertinents** allège non seulement le modèle, mais aussi le fichier exécutable, les dépendances et donc le coût énergétique du chargement et de l'exécution.

L'écoconception de services numériques

Typologie des suppressions possibles

Type d'élément	Fonction d'origine	Pourquoi le supprimer en inférence
Couches de régularisation (Dropout, BatchNormalization)	Réduisent le surapprentissage pendant l'entraînement	En inférence, elles n'apportent plus rien : on peut les geler ou les fusionner avec d'autres couches.
Couches de debug / instrumentation	Suivi, visualisation des activations, logs	Elles augmentent le graphe sans impact sur le résultat final.
Variables non entraînées	Statistiques, compteurs, gradients, caches	Inutiles une fois le modèle convergé.
Opérations de calcul intermédiaires	Étapes de pré/post-traitement internes	Souvent fusionnables ou déléguées à un préprocesseur externe.
Couches mortes ou isolées	Ne participent plus au flot de calcul principal	Résidus d'expérimentations ; à détecter automatiquement.

Exemples concrets (1) : Dropout et BatchNorm

- **Dropout** introduit du hasard pendant l'entraînement pour éviter le surapprentissage. En inférence, il est désactivé ; **il peut donc être retiré du graphe computationnel.**

- **BatchNormalization** peut être fusionnée avec la couche précédente (souvent une convolution) : les poids et biais sont "absorbés" dans la couche convolutive, supprimant une opération complète.

- Exemple (PyTorch)

```
model.eval()  # désactive Dropout et BatchNorm
optimized_model = torch.quantization.fuse_modules(model, [['conv1', 'bn1', 'relu']], inplace=False)
```

Exemples concrets (2) : Pruning structurel des poids nuls

- Certains poids deviennent nuls ou insignifiants après l'entraînement.
- On peut les supprimer définitivement pour réduire la taille du réseau.
- Exemple (PyTorch)

torch.nn.utils.prune.remove(module, 'weight')

L'écoconception de services numériques

Optimisation structurelle de modèles d'IA : Principe

L'optimisation de modèles d'IA est particulièrement importante dans le contexte de l'écoconception.

Elles visent toutes à :

- ✓ **réduire la taille du modèle**,
- ✓ **Réduire la consommation de calcul et des modèles,**
- ✓ **tout en préservant la performance.**

Les trois principales sont :

- ✓ **L'élagage (Pruning)**
- ✓ **La quantification (Quantization)**
- ✓ **La distillation de connaissances (Knowledge Distillation)**

L'écoconception de services numériques

Optimisation : Pruning (élagage)

Principe :

Le pruning consiste à **supprimer les parties inutiles du réseau de neurones**, c'est-à-dire les poids ou connexions qui ont peu d'influence sur le résultat final.

Idée de base :

Dans un modèle entraîné, beaucoup de neurones et de connexions ont un **poids proche de zéro** : ils contribuent très peu aux prédictions.

Les supprimer permet d'obtenir un modèle plus petit et plus rapide, sans perte majeure de précision.

Types de pruning :

Weight pruning : suppression des poids faibles individuellement.

Neuron pruning : suppression de neurones ou de filtres entiers (dans un CNN).

Structured pruning : suppression de blocs structurés (ex. une couche ou un canal).

Avantages :

Moins de calculs (donc moins d'énergie consommée). Taille du modèle réduite → stockage et transfert allégés. Inférence plus rapide.

Analogie :

C'est comme élaguer un arbre : on garde les branches essentielles qui nourrissent l'ensemble, on coupe le superflu.

L'écoconception de services numériques

Optimisation : Quantization (quantification)

Principe :

La quantization réduit la **précision numérique** utilisée pour représenter les poids et les activations du réseau.

Par défaut, les modèles utilisent des nombres **en 32 bits flottants (float32)**.

La quantification convertit ces valeurs en **16 bits, 8 bits**, voire moins, selon le matériel.

Exemple :

Avant : un poids = 0.12345678901234 (32 bits)

Après : un poids = 0.12 (8 bits)

Avantages :

Réduction du **poids mémoire** du modèle (jusqu'à 75 %), Accélération du calcul sur du matériel compatible (GPU/NPU). Baisse de la **consommation énergétique**.

Inconvénient :

Légère perte de précision si la quantification est trop agressive — mais souvent négligeable pour les applications pratiques.

Analogie :

C'est comme passer une photo HD en "qualité moyenne" : elle reste lisible, mais pèse moins lourd.

L'écoconception de services numériques

Optimisation : Knowledge Distillation (distillation de connaissances)

La distillation consiste à **transférer les connaissances** d'un grand modèle complexe (*teacher model*) vers un modèle plus petit et plus rapide (*student model*).

Le modèle "élève" apprend non seulement à reproduire les bonnes réponses, mais aussi à **imiter les probabilités** du modèle "professeur", ce qui lui permet de capturer des nuances dans la décision.

Étapes typiques :

On entraîne un grand modèle très performant.

On utilise ce modèle pour générer des prédictions (probabilités) sur un ensemble de données.

On entraîne un modèle plus petit à reproduire ces prédictions.

Avantages :

On obtient un modèle "léger" (moins de paramètres, plus rapide).

Les performances restent proches du modèle initial.

Très utile pour le déploiement sur appareils embarqués ou mobiles.

Analogie :

C'est comme un élève qui apprend auprès d'un professeur expert : il ne retient pas tout, mais l'essentiel, de manière synthétique et efficace.

L'écoconception de services numériques
Plan

- **Où se situe l'impact ?**
  - ✓ Centres de calcul
  - ✓ Ordres de grandeur (GWh / Wh)
  - ✓ Données chiffrées entraînement – requête – infrastructure
- **Principe clé : sobriété par l'usage**
  - ✓ Juste assez d'intelligence
- **Objectifs écologiques et leviers**
  - ✓ Où se situent les leviers d'écoconception de l'IA ?
  - ✓ Tableau Objectif → Outils
  - ✓ Recommandations et checklist
- **Optimiser le modèle**
  - ✓ Pruning, quantization, distillation
- **Optimiser le déploiement**
  - ✓ Simplification de graphes et conversion de modèles

L'écoconception de services numériques

Simplification de graphes

La simplification de graphe consiste à transformer le graphe de calcul issu de l'entraînement en un graphe d'inférence minimal, ne conservant que les opérations strictement nécessaires à la prédiction.

Objectifs

- ✓ Supprimer les nœuds inutiles à l'inférence
- ✓ Réduire le nombre d'opérations
- ✓ Diminuer la latence et la consommation énergétique
- ✓ Faciliter le déploiement sur des environnements contraints (edge, mobile)

Que supprime-t-on ?

Nœuds d'entraînement :

- ✓ calcul de la loss
- ✓ gradients
- ✓ optimizers (Adam, SGD, etc.)

Dépendances de contrôle inutiles

- ✓ Nœuds non connectés aux sorties du modèle

L'écoconception de services numériques

Exemples concrets (3) : Simplification des graphes - TensorFlow

Les outils d'optimisation : permettent de purger les nœuds d'entraînement (gradients, optimizers, loss…).

- **Ex. 1 — TensorFlow : purge explicite des nœuds d'entraînement**
  - ✓ **L'instruction remove_training_nodes**
    - GradientDescent,
    - nœuds liés à loss,
    - nœuds de contrôle inutiles,
  - ✓ **Exemple de codage**

```
import tensorflow as tf
# supprime automatiquement les nœuds spécifiques à l'entraînement,
# transforme le graphe en graphe d'inférence pur.
tf.compat.v1.graph_util.remove_training_nodes(graph_def)
```

  - ✓ **Gain**
    - Graphe plus petit
    - Inférence plus rapide
    - Réduction directe du coût énergétique par requête

L'écoconception de services numériques

Exemples concrets (3) : Simplification des graphes - ONNX

- **Exemple 2 — TensorFlow → ONNX : élimination automatique des nœuds inutiles**

Gain pour la phase de déploiement du modèle TensorFlow sur différents runtimes.

✓ **Exemple**

```
# identifie les nœuds non connectés aux sorties,
# supprime automatiquement tout ce qui est inutile à l'inférence.
python -m tf2onnx.convert \
 --saved-model model_tf \
 --output model.onnx
```

✓ **Gain**

- Portabilité du modèle
- Simplification sans intervention manuelle

L'écoconception de services numériques

Conversion de modèles d'IA : Principe

La conversion de modèles d'IA est un levier majeur d'écoconception en IA qui permet de réduire drastiquement la consommation de ressources en production (CPU, GPU, mémoire, bande passante), sans toucher aux performances fonctionnelles.

- Une fois un modèle d'IA entraîné (par exemple un réseau neuronal sous TensorFlow ou PyTorch), il est souvent trop **lourd et énergivore** pour être directement déployé en production.

- L'étape de conversion vise à adapter le modèle au contexte réel d'exécution (serveur, embarqué, mobile, edge device), afin d'obtenir une inférence plus rapide, moins consommatrice et plus durable.

- La conversion consiste à transformer le modèle entraîné dans un **format optimisé ou spécialisé** pour le matériel d'exécution.

L'écoconception de services numériques

Conversion : Frameworks de conversion

Ce processus s'appuie sur des outils et frameworks standardisés tels que :

Outil / Format	Usage principal	Description synthétique
ONNX (Open Neural Network Exchange)	Portabilité multi-frameworks	Format ouvert compatible avec TensorFlow, PyTorch, scikit-learn, etc. Permet d'exécuter le même modèle sur différentes plateformes sans réentraînement.
TensorRT (NVIDIA)	Optimisation pour GPU	Convertit et optimise les graphes TensorFlow ou PyTorch pour une exécution ultra-rapide sur GPU NVIDIA (fusion de couches, quantification, batch optimal…).
TensorFlow Lite (TFLite)	Exécution sur mobile / embarqué	Allège les modèles TensorFlow pour Android, Raspberry Pi, microcontrôleurs, etc. Compatible avec quantization, pruning et distillation.
CoreML / OpenVINO	Apple / Intel	Conversion et optimisation pour écosystèmes matériels spécifiques (Mac/iOS, processeurs Intel).
TorchScript	PyTorch déployé en production	"Gèle" le graphe computationnel pour accélérer les inférences et éliminer le code Python inutile.

Ces formats ne changent pas la logique du modèle, mais **réécrivent sa représentation interne** pour **exploiter au mieux le matériel** et **réduire le gaspillage computationnel**.

L'écoconception de services numériques

Conversion : Étapes typiques du processus

Étape 1. Exporter le modèle de base

Sauvegarde du modèle entraîné au format natif (.h5, .pt, .pb, etc.).

Nettoyage des couches ou variables inutiles (dropout, debug nodes, training ops).

Étape 2. Conversion dans un format optimisé

Passage via un convertisseur :

```
tensorflowjs_converter --input_format=tf_saved_model mymodel/ mymodel_optimized/
```

ou :

```
torch.onnx.export(model, dummy_input, "model.onnx")
```

Étape 3. Application d'optimisations structurelles

Fusion de couches (par ex. convolution + batch normalization).

Quantization post-training (réduction des poids en int8 ou float16).

Suppression des gradients et opérateurs liés à l'apprentissage.

Étape 4. Vérification de l'intégrité et des performances

Tests sur un jeu d'évaluation pour vérifier que la précision reste stable.

Mesure du temps d'inférence et de la consommation énergétique (ex. via CodeCarbon).

L'écoconception de services numériques

Optimisation : Gains typiques

Exemple concret : un modèle TensorFlow de 300 Mo converti en TFLite quantifié peut descendre à 25 Mo, tout en gardant 99 % de sa précision et en s'exécutant 5 à 8 fois plus vite sur un smartphone.

Critère	Amélioration moyenne après conversion
Taille du modèle	↓ 50 à 90 %
Temps d'inférence	↓ 2 à 10×
Consommation mémoire	↓ 30 à 80 %
Consommation énergétique	↓ 20 à 70 %
Empreinte carbone par requête	↓ 40 à 90 %