Projet : Accélération de l'inférence sur CIFAR-10
Recueil des exigences et périmètre

1 Contexte et objectif

Le projet consiste à entraîner un modèle de classification d'images pour **CIFAR-10** et à le rendre **aussi rapide que possible en inférence**, en évaluant prioritairement la **latence sur GPU** (batch = 1), tout en respectant une contrainte de performance en précision (accuracy).

2 Hypothèses et contraintes

— **Données d'entraînement autorisées :** uniquement **CIFAR-10 train**. Aucune donnée externe ne doit être utilisée pour l'apprentissage.

— **Objectif de précision :** atteindre $\geq 85\%$ d'accuracy sur le **jeu de test CIFAR-10**.

— **Matériel de mesure : GPU** (conditions de benchmark fixées et reproductibles).

3 Exigences fonctionnelles (RF)

RF1 — Entraînement du modèle

— Le système doit permettre d'entraîner un modèle de classification sur CIFAR-10 en utilisant **uniquement** CIFAR-10 train (augmentations autorisées sur ces images).

— Le modèle final doit produire une sortie **à 10 classes** (logits) correspondant aux classes CIFAR-10.

RF2 — Évaluation de la précision

— Le système doit calculer l'accuracy sur CIFAR-10 test.

— Le système doit permettre de comparer l'accuracy entre plusieurs variantes de modèles/optimisations.

RF3 — Mesure de latence en inférence (GPU)

— Le système doit mesurer la **latence d'inférence** sur GPU pour **batch = 1**.

— Le protocole doit inclure :

  — un **warm-up** avant mesure,

  — un grand nombre d'itérations de mesure,

  — une **synchronisation GPU** afin d'obtenir un temps réel (ex. `torch.cuda.synchronize()` ou événements CUDA).

— Le système doit produire au minimum : **latence moyenne** et **latence p95** (95e percentile).

RF4 — Suivi des expérimentations

— Le système doit enregistrer, pour chaque expérience : modèle/variante, accuracy, latence moyenne, latence p95, et (optionnel) taille du modèle et nombre de paramètres.

— Le système doit permettre une comparaison synthétique (tableau) entre les variantes.

4 Exigences non fonctionnelles (RNF)

RNF1 — Reproductibilité

— Les mesures doivent être réalisées **dans des conditions fixes** : même GPU, mêmes réglages de batch, même prétraitement, même mode d'inférence (`eval()` et `no_grad()`).

— Les résultats doivent être reproductibles à l'exécution (seed, versions loggées si possible).

RNF2 — Conformité aux contraintes de données

— Aucune dépendance à un pré-entraînement externe (ex. ImageNet) ne doit être utilisée pour l'apprentissage si l'interprétation du sujet est stricte.

— En pratique, les modèles de type *MobileNet* devront être instanciés avec `weights=None` et entraînés sur CIFAR-10.

RNF3 — Performance

— Le modèle final doit respecter **accuracy** $\geq 85\%$ et viser la **plus faible latence** possible sur GPU (batch = 1).

— Les optimisations doivent être évaluées **après chaque modification** (démarche incrémentale).

RNF4 — Qualité de la démarche

— Le livrable doit expliciter la **méthodologie** : baseline → modification → mesure → conclusion, répétée sur plusieurs itérations.

5 Périmètre (ce qui est inclus)

5.1 Périmètre recommandé (MVP robuste)

P1. Benchmark de latence GPU : implémentation du protocole (warm-up, synchronisation, moyenne + p95).

P2. Baseline "vitesse" : entraîner depuis zéro un modèle léger (**MobileNetV3-Small** ou **ShuffleNetV2**) avec `weights=None` et une tête de sortie à 10 classes.

P3. Baseline "précision" / teacher : entraîner **ResNet-18 (adaptée CIFAR)** depuis zéro pour obtenir une référence robuste ($\geq 85\%$) et servir de teacher.

P4. Optimisations GPU prioritaires (mesurées une à une) :

- inférence en **FP16** (autocast),
- `torch.compile` (si disponible et stable),
- format mémoire `channels_last` (si bénéfique).

P5. Tableau comparatif final : accuracy, latence moyenne, latence p95, (optionnel) taille & paramètres.

5.2 Extensions (si nécessaire pour atteindre $\geq 85\%$ avec un modèle léger)

E1. Knowledge Distillation : teacher ResNet-18 $\rightarrow$ student MobileNet/ShuffleNet pour conserver l'accuracy tout en réduisant la latence.

E2. Quantification : à considérer uniquement si le support GPU et la chaîne d'export le rendent pertinent (sinon priorité moindre en GPU).

6 Hors périmètre (pour éviter la dispersion)

— Mixture of Experts (MoE) : complexité élevée pour un gain de latence incertain dans ce contexte.

— Approches lourdes de *feature engineering* + XGBoost si elles compromettent l'objectif d'accuracy ou la latence globale.

7 Livrables attendus

— Code d'entraînement et d'évaluation (accuracy + latence GPU).

— Rapport synthétique des expériences (méthode incrémentale) et tableau de résultats.

— Modèle final (poids) et configuration utilisée pour reproduire les mesures.