TP3-FlowersClassifier

TP3 — Flowers Classifier : Tournoi CNN vs ViT

Objectif

Vous devez implémenter **deux modèles** de classification d'images de fleurs :

1. Un modèle basé sur un **CNN** (ex : ResNet, MobileNet, VGG, ou architecture custom)
2. Un modèle basé sur un **Vision Transformer (ViT)** (ex : ViT, DeiT, Swin, ou architecture custom)

Vos modèles seront évalués dans un **tournoi** : chaque modèle sera testé sur un jeu d'images **non présent dans le dataset d'entraînement**.

Barème (/20)

Composante	Points	Détail
Modèle CNN	/5	Note attribuée selon votre **rang** dans le tournoi CNN
Modèle ViT	/5	Note attribuée selon votre **rang** dans le tournoi ViT
Questions OCR	/5	5 questions sur le cours OCR (1 pt chacune)
Questions ViT	/5	5 questions sur le cours Vision Transformer (1 pt chacune)
Total	**/20**	

La métrique du tournoi est l'**accuracy** sur le jeu de test (images inconnues).

Cas d'évaluation à 0 :

- Le code ne s'exécute pas et il y a des erreurs de syntaxe.
- Il y a trop de similitude avec le code d'autres étudiants (aussi valable pour les questions)

Dataset

Le dataset de fleurs est disponible sur **Moodle**. Il contient des images réparties en sous-dossiers par classe.

ATTENTION — Overfitting

Les images utilisées pour l'évaluation du tournoi ne sont PAS dans le dataset fourni.

Votre modèle doit **généraliser** à des images jamais vues. Conseils :

- Utilisez de la **data augmentation** (rotations, flips, color jitter, etc.)
- Ne sur-entraînez pas trop longtemps (early stopping)
- Préférez le **transfer learning** (modèles pré-entraînés sur ImageNet)
- Validez sur votre split de validation pour surveiller l'overfitting
- Testez avec des images trouvées sur internet pour vérifier la robustesse

---

Consignes de rendu

Vous devez rendre **une archive** (`NOM_PRENOM_flowers_classifier.zip`) qui contient :

1. Un fichier `NOM_PRENOM_flowers_classifier.py` qui contient les fonctions suivantes :
   - `load_cnn_model()`
   - `load_vit_model()`
   - `predict(model, image_path)`
2. Les fichiers `.pth` contenant les poids de vos modèles.
3. Un fichier `NOM_PRENOM_reponses.txt` qui contient les réponses aux questions de cours.

Les poids de vos modèles doivent être sauvegardés dans des fichiers `.pth` et chargés par vos fonctions.

Interface attendue

```python
# NOM_PRENOM_flowers_classifier.py

CLASSES = ['daisy', 'dandelion', 'rose', 'sunflower', 'tulip']

def load_cnn_model(weights_path='cnn_flowers.pth'):
    """Charge et retourne le modèle CNN entraîné."""
    ...

def load_vit_model(weights_path='vit_flowers.pth'):
    """Charge et retourne le modèle ViT entraîné."""
```

```
    ...

def predict(model, image_path):
    """Prédit la classe d'une image. Retourne un str parmi CLASSES."""
    ...
```

Script d'évaluation (utilisé par l'enseignant)

```
from flowers_classifier import load_cnn_model, load_vit_model, predict

cnn = load_cnn_model()
vit = load_vit_model()

for img_path in test_images:
    pred_cnn = predict(cnn, img_path)
    print(pred_cnn)
    pred_vit = predict(vit, img_path)
    print(pred_vit)

```

Partie 1 — Setup et exploration du dataset

Les images sont stockées dans des dossiers nommés selon la classe :

- `daisy/`
- `dandelion/`
- `rose/`
- `sunflower/`
- `tulip/`
- etc.

Seule la reconnaissance de ces 5 classes sera évaluée, veillez à ne pas perdre du temps avec les autres classes.

Partie 2 — Data Augmentation et DataLoaders

La data augmentation est **essentielle** pour éviter l'overfitting, surtout avec un petit dataset.

Exemples de transformations utiles :

- `RandomResizedCrop`, `RandomHorizontalFlip`, `RandomVerticalFlip`
- `ColorJitter` (luminosité, contraste, saturation)
- `RandomRotation`, `RandomAffine`
- `RandAugment` (torchvision >= 0.11)
- Normalisation ImageNet : `mean=[0.485, 0.456, 0.406]`, `std=[0.229, 0.224, 0.225]`

Partie 3 — Modèle CNN (/5 tournoi)

Implémentez votre modèle CNN. Vous pouvez :

- Utiliser un modèle pré-entraîné (ResNet, MobileNet, EfficientNet...) et le **fine-tuner**
- Modifier la tête d'un modèle pré-entraîné

Conseils

- Le **transfer learning** est fortement recommandé (modèle pré-entraîné sur ImageNet)
- Remplacez la dernière couche FC par une couche adaptée à 5 classes
- Essayez différentes stratégies : fine-tuning complet vs gel des premières couches
- Surveillez la courbe de validation pour détecter l'overfitting

---

Partie 4 — Modèle ViT (/5 tournoi)

Implémentez votre modèle Vision Transformer. Vous pouvez :

- Utiliser un ViT pré-entraîné via **timm** (`vit_base_patch16_224`, `vit_small_patch16_224`, `deit_small_patch16_224` ...)
- Utiliser **Swin Transformer** (aussi disponible dans timm)

Conseils

- `pip install timm` pour accéder aux modèles pré-entraînés
- Les modèles `small` ou `tiny` sont plus rapides à fine-tuner
- Pensez au **learning rate** : souvent plus faible pour les ViT (ex : 1e-4 ou 5e-5)
- La data augmentation est encore plus critique pour les ViT (moins de biais inductif)

Partie 5 — Boucle d'entraînement

Fonction d'entraînement générique utilisable pour les deux modèles.

Partie 6 — Préparation du fichier de rendu

Vérifiez que vos fonctions `load_cnn_model`, `load_vit_model` et `predict` fonctionnent correctement.

Partie 7 — Questions de cours (/10)

Répondez dans un fichier `NOM-PRENOM-TP3-Reponses.txt`. Chaque question vaut **1 point**.

Questions OCR (5 pts)

Q1 (OCR) — Quelle est la différence entre le **CER** et le **WER** ? Dans quel cas le WER peut-il être élevé alors que le CER est faible ? Donnez un exemple concret.

Q2 (OCR) — Décrivez les **8 étapes** d'une chaîne OCR complète, de l'acquisition de l'image jusqu'à la validation. Quelle étape a selon vous le plus d'impact sur la performance finale ?

Q3 (OCR) — Expliquez le rôle du **CTC** (Connectionist Temporal Classification) dans un système CRNN. Pourquoi est-il nécessaire ? Illustrez avec l'exemple du mot `BON`.

Q4 (OCR) — Comparez les détecteurs **EAST** et **CRAFT**. Quels sont les avantages et inconvénients de chacun ? Dans quel scénario privilégieriez-vous CRAFT ?

Q5 (OCR) — Pourquoi utilise-t-on un **BiLSTM** (bidirectionnel) plutôt qu'un LSTM simple dans un CRNN ? Donnez un exemple de confusion de caractères que le contexte bidirectionnel permet de résoudre.

Questions Vision Transformer (5 pts)

Q6 (ViT) — Décrivez le pipeline complet d'un ViT pour la classification, de l'image en entrée jusqu'à la prédiction. Précisez les dimensions des tenseurs à chaque étape pour une image 224x224 avec P=16.

Q7 (ViT) — Qu'est-ce qu'un **biais inductif** ? Expliquez pourquoi un CNN en possède davantage qu'un ViT, et quelle conséquence cela a sur le besoin en données d'entraînement.

Q8 (ViT) — Expliquez le mécanisme de **self-attention** (Q, K, V). Quel est le rôle de la division par `sqrt(d)` ? Pourquoi le coût est-il en `O(N^2)` ?

Q9 (ViT) — Comment le **Swin Transformer** réduit-il le coût de l'attention par rapport au ViT classique ? Expliquez le principe des **shifted windows** et du **patch merging**.

Q10 (ViT) — Quel est le rôle du token `[CLS]` dans un ViT ? Pourquoi ne pas utiliser directement la sortie d'un patch spécifique pour la classification ? Quelle alternative existe ?