BARKAN Ines, Alpha Ba

Sujet du projet :
Estimation de l'âge à partir d'images de visages en utilisant des réseaux de neurones profonds

Introduction

Le dépôt est forké depuis `yu4u/age-estimation-pytorch`, qui est une implémentation PyTorch d'un CNN pour l'estimation d'âge à partir d'images de visage, entraînée sur le dataset APPA-REAL, un dataset avec véritables âges annotés.

Les hyperparamètres, options d'optimisation et architectures peuvent être modifiés via les fichiers `defaults.py` et les arguments de ligne de commande (ex. choix d'un backbone ResNet ou autre).

Dataset : il utilise APPA-REAL, qui contient environ 7 591 images avec de nombreux votes humains par image (ce qui rend la moyenne des âges assez stable).

Que fait chaque script (juste un grand aperçu)

- ➢ **train.py** : entraîne le modèle sur les données d'entrainement, en sauvegardant checkpoints et métriques.
- ➢ **test.py** : évalue le modèle entraîné et calcule un MAE (mean absolute error, erreur moyenne absolue) sur le test set.
- ➢ **demo.py** : permet d'appliquer le modèle sur des images ou webcam pour prédire l'âge, souvent pour une démonstration rapide.
- ➢ **model.py** : contient la définition du réseau CNN utilisé (peut être modifiable).

Ce qu'on peut faire

Implementation Section

1. Backbone et architectures

- Tester différents backbones : ResNet18 vs ResNet50, EfficientNet, MobileNet.
- Multi-task learning avec attributs supplémentaires (ethnie, maquillage, genre, expression).
- Ensembling : combiner plusieurs modèles pour moyenne des prédictions.

2. Data et prétraitement

- Ajouter / enlever data augmentation pour améliorer généralisation.
- Ajouter données synthétiques via GAN (StyleGAN), comparer réel vs synthétique vs mixte.
- Tester influence du fond, qualité de l'image et éclairage.
- Entraîner avec moins de données pour tester robustesse.
- Stratégies pour gérer données déséquilibrées (à trouver faut se renseigner).

3. Formulation du problème et fonctions de perte

- Poser problème en régression vs classification et comparer (DEX, residualDEX, label smoothing c'est déjà de la classification faut les utiliser pour comparer les performances).
- Tester Ordinal Regression, DEX et Residual DEX.
- Implémenter label smoothing pour classification.
- Loss aleatoric : Gaussian / Laplace likelihood pour régression (homoscedastic / heteroscedastic) => Comparer différentes fonctions de perte : L1 vs L2.

4. Régularisation et incertitude

- Ajouter Dropout et utiliser MC-Dropout pour estimation d'incertitude.
- Améliorer performance avec Test-Time-Augmentation (TTA) et ensembling.

Pour analyser :

(parfois c’est déjà écrit dans la section précédente mais c’est juste pour etre clair sur là ou on doit analyser)

Analysis Section

1. Performance et robustesse

- Comparer performance et coût de calcul selon architecture/backbone.
- Évaluer effet de la data augmentation et des données synthétiques.
- Étudier l’effet de l’entraînement sur moins de données et des stratégies pour données déséquilibrées.

2. Étude des erreurs

- Analyse des erreurs selon âge (enfants / adultes / seniors).
- Étudier influence du fond, qualité d’image et éclairage.

3. Généralisation

- Tester généralisation d’un modèle APPA-REAL sur d’autres datasets.
- Explorer méthodes simples de domain adaptation, par ex. BatchNorm.

4. Incertitude et fiabilité

- Évaluer qualité des prédictions avec MC-Dropout, TTA et loss aleatoric.