TP 4: Estimation de l'orientation d'une pipeline par réseaux de neurones convolutifs

Master TSI, 2024-2025

M.M. Nawaf

Objectif

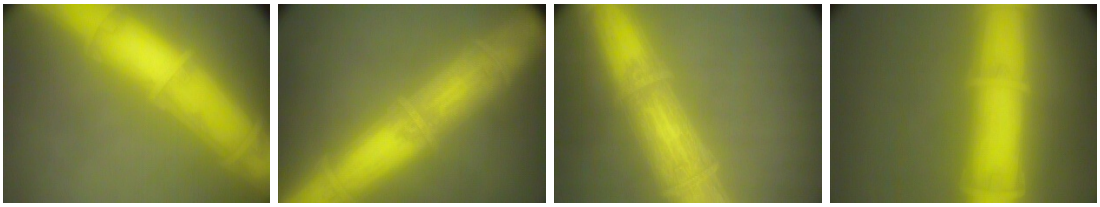
Le but de ce TP est de mobiliser les notions apprises en cours sur les réseaux de neurones convolutifs pour estimer l'orientation d'une pipeline avec un système de bout en bout.

Ce TP est noté et compte deux TP , la remise se fait sur Ametice/Education Hub, la date limite est le dimanche 20/10 à 23h.

1 Jeu de données

Le jeu de données peut être consulté sur Ametice ou à partir de l'un des liens suivants :
`https://share.nawaf.fr//s/deep3`
`https://amubox.univ-amu.fr/s/CX7rjBdkafsFeg4`

Il y a deux fichiers, chacun contenant un tableau NumPy enregistré avec np.save. Le premier tableau contient les images. Le second tableau contient les informations de position et d'orientation du pipeline, qui seront détaillées pendant la session.



2 Rendu

Il faut soumettre un fichier .py et un tableau comparatif illustrant l'effet de la variation des hyperparamètres et le vecteur d'entrée sur les résultats.

3 Évaluation

Le TP sera évalué sur quatre critères : les résultats (accuracy du modèle), la compacité du code, l'analyse des résultats (le tableau), et le respect des consignes données en section 5.

4 Travail à effectuer

Le principal problème est que la nature périodique de l'orientation ne peut pas être correctement gérée par les réseaux de neurones. Par exemple, il n'est techniquement pas possible pour le modèle d'apprendre que 360 et 0 degrés représentent le même angle. De plus, comme notre objet d'intérêt est une pipeline, qui n'a pas d'orientation directionnelle, une orientation de 135 degrés est identique à 45 degrés.

Une méthodologie pour aborder ce problème sera discutée pendant la session. Mais vous êtes totalement libre de choisir une approche différente.

Pour ce TP, vous devrez peut-être écrire vos propres fonctions de perte et de métrique, qui peuvent être définies en tant que fonction Python procédé par un décorateur. Attention, il faut exclusivement utiliser des fonctions de TensorFlow (pourquoi ?). Voici des exemples de fonction de perte et de métrique.

```
@keras.utils.register_keras_serializable(package="Custom")
def abs_difference_loss(y_true, y_pred):
        return tf.abs(y_true, y_pred)

@keras.utils.register_keras_serializable(package="Custom")
def ratio_metric(y_true, y_pred):
        return tf.abs(y_true/y_pred)

# When you load the model, don't forget to add
custom_objects={
        "abs_difference_loss": abs_difference_loss,
        "ratio_metric": ratio_metric
}
```

5 Consignes à Respecter

— Respectez le style de code Python.
— Programmez en anglais (une norme de l'industrie).
— Pas de commentaires.
— Évitez les paramètres codés en dur.
— Pré-allouez les matrices (l'utilisation de *append* est interdite).
— L'utilisation des boucles (for, while, etc) est interdite.
— Utilisez uniquement les notions abordées en cours.