

test.md

# Projet d'Intelligence Artificielle : Classification d'Images de Tournesols et Choux-fleurs

---



## Compte rendu de développement du Projet

---

### Introduction

Ce projet avait pour objectif d'adapter un modèle de réseau de neurones existant afin de reconnaître et classer des images de tournesols, de choux-fleurs, ou d'autres catégories. Puis de réaliser une petite application, pour tester notre modèle avec des images.

L'équipe de développement était composée :

- [Antoine](#)
- [Lucas](#)

- [Gaëtan](#)

## Spécifications

Le projet visait à développer une application capable d'identifier si une image contient un tournesol, un chou-fleur, ou autre. Nous avons utilisé **MobileNet**, une architecture de réseau neuronal optimisée pour les appareils mobiles, comme base pour notre modèle de détection d'images de choux-fleurs, de tournesols et d'autres catégories. Nous nous sommes également appuyés sur la bibliothèque Python **Keras**.

## Jeu de Données

La deuxième étape, après avoir récupéré notre modèle initial, nous avons constitué un jeu de données (datasets) avec environ 5000 images.

1. Des images de Tournesol, données par notre professeur.
2. Des images de Choufleur, récupérées sur :  
<https://data.mendeley.com/datasets/t5sssfgn2v/3>
3. Des images random, dites "autres", constituées d'images de paysages, bâtiments, humains, animaux... , récupérées sur :  
<https://www.kaggle.com/datasets/pankajkumar2002/random-image-sample-dataset/>

Ce dataset nous a permis d'entraîner notre modèle à notre guise sur un large choix de photos.

Nous avons chargé nos images de datasets à partir d'un dossier à l'aide de cette fonction, à partir de la librairie Keras.

```
dataset = ut.image_dataset_from_directory(  
    directory,  
    labels='inferred',  
    label_mode='int',  
    color_mode='rgb', # ou "grayscale"  
    class_names=['Choux-fleurs', 'Tournesols', 'Autres'],  
    image_size=(224, 224),  
    batch_size=32, # nombre d'images à lire à chaque fois,  
    shuffle=True,  
    interpolation='nearest'  
)
```

## Apprentissage du modèle

Pour l'apprentissage du modèle, nous avons donc utilisé Keras, voici la fonction utilisée :

```
# Entraînement du modèle
history = model_mobilenet.fit(train_images, train_labels,
                              batch_size=32,
                              epochs=20,
                              validation_data=(test_images, test_labels))
```

Pour entraîner notre modèle avec Keras, nous avons utilisé 20 epochs et un batch\_size de 32. Ce choix est le résultat d'expérimentations visant à trouver un équilibre entre vitesse d'apprentissage, performances du modèle et prévention du surapprentissage. Certains des tests nous ont montré que le modèle présentait une dégradation des performances au-delà de 20 epochs, probablement en raison d'une sur-adaptation aux données d'entraînement. Ces paramètres ont été ajustés pour obtenir des performances optimales tout en évitant le surapprentissage.

## Test du modèle, après entraînement

Après avoir entraîné notre modèle, nous l'avons testé directement en Python sur différentes images pour évaluer sa performance. Nous avons fait un affichage rapide avec l'image passée au modèle et en titre la prédiction de celui-ci à l'aide de la librairie matplotlib :

```
def predict_image(image_path):
    img = keras_image.load_img(image_path, target_size=(224, 224)) # Redi
    img_array = keras_image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)

    # Faire une prédiction avec votre modèle
    predictions = model_mobilenet.predict(img_array)

    # Afficher les prédictions, avec matplotlib
    class_names = ['Choux-fleurs', 'Tournesols', 'Autres'] # Liste des no
    predicted_class = np.argmax(predictions)
    predicted_class_name = class_names[predicted_class]
    print(f'Classe prédite : {predicted_class_name}')
    plt.imshow(img)
    plt.title(predicted_class_name)
    plt.show()

predict_image('./image_test/chou.jpg')
predict_image('./image_test/tournesol.png')
predict_image('./image_test/elephant.png')
predict_image('./image_test/madame.png')
predict_image('./image_test/vroum.jpg')
```

Resultat :

1. image de chou-fleur : le modele prédit "chou-fleur"

2. image de tournesol : le modele prédit "tournesol"
3. image d'éléphant : le modele prédit "autres"
4. image de femme : le modele prédit "autres"
5. image de voiture : le modele prédit "autres"

## Sauvegarde du modèle

Enfin, nous avons sauvegardé notre modèle pour pouvoir l'utiliser dans une application Java.

## Outils et Technologies

Pour le développement, nous avons utilisé :

- Python avec la librairie Keras pour la construction et l'entraînement du modèle de reconnaissance d'images.
- Le modèle était basé sur MobileNetV3.
- Adaptation du modèle pour la classification des images de tournesols et de choux-fleurs.

## Déploiement

Pour le déploiement, nous avons utilisé Java avec un fichier JAR pour développer une application qui permettait de tester l'inférence du modèle entraîné. Le modèle préalablement sauvegardé était chargé dans cette application Java pour réaliser des prédictions sur de nouvelles images.

## Conclusion

En résumé, ce projet a réussi à adapter un modèle existant pour identifier des tournesols, des choux-fleurs et d'autres objets dans des images. L'application Java développée a été utilisée avec succès pour évaluer l'efficacité du modèle pré-entraîné.