UNIVERSITÉ DE LORRAINE | IUT Metz Informatique

Département Informatique

SAÉ 5 — Développement avancé

Rapport SAÉ 5 — Sprint 4

Diego TRIVINO

Mathias FRIES

Alexandre KESSELER

Joshua BROCHOT

Table des matières

1 Introduction

Durant ce sprint, nous avons poursuivi le développement de l'application en corrigeant les problèmes identifiés lors du sprint précédent. Nous avons commencé à implémenter l'authentification des utilisateurs, ainsi qu'à intégrer les modèles de reconnaissance d'images. Nous av

2 Réflexion sur le modèle de reconnaissance

Cette section présente à la fois les choix déjà retenus et les aspects qui devront être testés pour le futur modèle de reconnaissance d'images destiné à l'identification des salles. Ces orientations résultent d'une phase de recherche et d'analyse ayant permis de comparer plusieurs approches. Elles couvrent les décisions prises ainsi que les pistes d'expérimentation à explorer afin de déterminer la solution la plus adaptée.

2.1 Paramètres de données et estimation de stockage

Les images seront standardisées à une résolution de **384** $\times$ **384** pixels. L'objectif minimal retenu est de disposer d'au moins **300 images par salle**.

Estimation de stockage (ordre de grandeur) :
Supposant une taille moyenne compressée par image en JPEG d'environ **120 Ko** :

$$300\ images/salle \times 120\ Ko \approx 36\ Mo/salle.$$

Pour 20 salles :

$$36\ Mo \times 20 \approx 720\ Mo.$$

2.2 Paramètres dynamiques exposés via l'admin

Les paramètres suivants seront modifiables dynamiquement depuis le panneau d'administration afin de faciliter les tests et l'optimisation :

- `input_size` : résolution d'entrée du modèle (ex. 224, 320, 384).
- `confidence_threshold` : seuil minimal de confiance pour valider une prédiction.
- `top_k` : nombre de classes (salles) renvoyées en sortie.
- `model_version` : sélection de la version du modèle (ex. stable / expérimental).

2.3 Fonctions essentielles du pipeline de traitement d'images

Le pipeline de prétraitement et validation comprendra au minimum les fonctions suivantes (signature indicative) :

```
def load_image(bytes_or_path)               # chargement et conversion en RGB
def resize_and_center_crop(img, size)       # redimensionne et centre l'image
def normalize(tensor, mean, std)            # normalise les valeurs des pixels
def augment_for_training(img)               # flips, rotations, brightness, noise, blur
def image_batcher(list_images, batch_size)  # regroupe les images en lots
def compress_image(img, max_kb)             # compresse JPEG pour réduire latence upload
def validate_image(img)                     # vérifie corruption, dimensions, ratio
```

2.4 Stratégie d'inférence

Deux modes d'inférence seront testés afin d'évaluer leurs gains réels selon les performances du VPS et le profil d'utilisation :

- **Inférence par image (mode simple)** : chaque image générée est envoyée et traitée immédiatement. Avantage : simplicité et latence par image minimale. Inconvénient : overhead réseau et moindre efficacité serveur.
- **Batching côté serveur (mode optimisé)** : plusieurs images sont regroupées et traitées en un seul appel au modèle (batch tensoriel). Avantage : meilleur throughput et utilisation efficace du CPU/GPU. Inconvénient : complexité d'implémentation (buffering, timers) et gestion mémoire.

L'approche retenue pour les expérimentations est de **comparer les deux modes** sur des métriques mesurables : temps moyen d'inférence par image, latence perçue, consommation mémoire et précision des prédictions. Le choix final dépendra des résultats en conditions réelles sur le VPS.

2.5 Prochaines étapes

Les décisions prises cette semaine structurent le plan d'expérimentation : implémentation du pipeline de prétraitement, création d'un script d'essais pour inference en batch et en synchronisé, puis mesure des performances pour déterminer la stratégie optimale (basculement éventuel vers un batching dynamique en production). Le panneau d'administration permettra d'ajuster les paramètres sans redéploiement afin d'itérer rapidement sur les configurations.

3 Mise en place de l'authentification et du profil utilisateur

Afin de sécuriser l'accès à l'application et de permettre une identification fiable des utilisateurs, on a mis en place un système d'authentification complet basé sur FastAPI, PostgreSQL et les JSON Web Tokens (JWT). L'objectif était de fournir un mécanisme d'inscription, de connexion et de consultation du profil, tout en restant compatible avec l'architecture existante du projet.

3.1 Système d'authentification avec JWT

L'authentification repose sur un échange de jetons JWT, permettant aux utilisateurs de s'identifier de manière sécurisée. Lors de l'inscription, les informations de l'utilisateur sont enregistrées en base de données, et le mot de passe est préalablement haché afin de ne jamais être stocké en clair.

Lors de la connexion, si les informations fournies sont valides, un jeton JWT signé est généré et retourné au client. Ce jeton contient l'identifiant de l'utilisateur et permet ensuite d'accéder aux routes protégées de l'API. Grâce à ce fonctionnement, aucune session n'est conservée côté serveur, ce qui garantit une architecture légère et facilement scalable.

3.2 Route protégée et récupération du profil

Une route sécurisée a été mise en place afin de permettre à l'utilisateur connecté de consulter ses informations de profil. Cette route nécessite obligatoirement la présence d'un jeton valide dans l'en-tête de la requête, conformément au standard `Authorization: Bearer <token>`.

Si le jeton est valide, l'API identifie automatiquement l'utilisateur et renvoie ses informations. Ce mécanisme permet au front-end de savoir à tout moment si un utilisateur est connecté et d'afficher dynamiquement son profil dans l'application.

3.3 Intégration avec l'interface utilisateur

Côté application mobile, une interface a été développée pour gérer la connexion, la création de compte et l'affichage du profil sur une seule et même page. L'application communique directement avec les routes d'authentification du backend, en stockant le jeton JWT sur l'appareil afin de maintenir une connexion persistante. Lorsque le jeton est valide, les informations de l'utilisateur sont automatiquement chargées et affichées dans l'onglet Profil.

Cette intégration offre une expérience fluide pour l'utilisateur, tout en s'appuyant sur une architecture sécurisée et conforme aux bonnes pratiques du développement mobile et web moderne. Il s'agit d'une première version de cette fonctionnalité, destinée à évoluer et à être enrichie lors des prochains sprint du projet.

4 Système de gestion modulaire des modèles de reconnaissance

L'objectif de cette fonctionnalité est de mettre en place un système de gestion modulaire des modèles de reconnaissance. L'idée est de pouvoir disposer de plusieurs modèles interchangeables, sélectionnables directement depuis un panneau d'administration de l'application. Cela permettrait de changer dynamiquement le modèle utilisé pour l'inférence, sans modification du code source et sans redémarrage de l'application.

Ce mécanisme offrira une grande flexibilité lors de nos phases de tests, notamment pour comparer différentes versions de nos modèles de reconnaissance et basculer de l'une à l'autre sans friction.

Nous avons appelé ce système **MME** (Modular Model Engine). Pour le moment, son fonctionnement inclut :

— la récupération automatique des modèles déposés dans un dossier dédié ;

— l'enregistrement des informations associées dans la base de données.

À l'avenir, cette fonctionnalité devra permettre d'ajouter ou de supprimer des modèles directement en production, toujours sans nécessiter de redémarrage de l'application.

Bien que le MME soit encore en cours de développement, il constitue un élément essentiel pour faciliter les tests liés à nos expérimentations sur les modèles de reconnaissance.

5 Corrections et améliorations réalisées durant le sprint

Une partie importante de ce sprint a été consacrée à la correction des erreurs rencontrées lors du sprint précédent, ainsi qu'à l'amélioration générale de la stabilité de notre environnement de développement et de test. Ces interventions ont principalement concerné la gestion des dépendances, la cohérence des versions React/Expo, la configuration Docker et divers ajustements nécessaires à la fiabilité globale du projet.

5.1 Résolution des conflits de dépendances et alignement des versions React

Nous avons mis à jour l'ensemble des dépendances du frontend afin de résoudre les conflits qui bloquaient l'installation et l'exécution du projet. Cette intervention a permis d'aligner les versions de React et de l'écosystème Expo, améliorant ainsi la stabilité générale du frontend. Le package a également été marqué comme `private` pour éviter toute publication accidentelle.

5.2 Améliorations de la configuration Docker et prévention des conflits de ports

Nous avons apporté plusieurs corrections à la configuration Docker afin de permettre l'exécution simultanée des environnements de développement, de test et de production. Cela inclut le renommage du conteneur backend de `api-prod` vers `api-test` dans l'environnement de test, ainsi que la réorganisation des ports exposés dans les fichiers `docker-compose.dev.yml` et `docker-compose.test.yml`. Ces ajustements évitent désormais toute collision de ports entre les différents environnements et garantissent un workflow plus fluide, en particulier lors du développement parallèle et de l'exécution des tests.

5.3 Améliorations fonctionnelles

Enfin, nous avons apporté des correctifs liés à l'importation de photos depuis la galerie. Le bouton d'import a été ajusté pour fonctionner également sur les appareils ne disposant pas de caméra, améliorant ainsi l'accessibilité et la compatibilité globale de la fonctionnalité.

6 Conclusion

Durant ce sprint, nous avons apporté de nombreux correctifs afin de garantir la viabilité et la fiabilité du projet. Nous avons également mené une réflexion sur les modèles de reconnaissance, établissant des orientations et des objectifs pour leur intégration future. Par ailleurs, l'implémentation de l'authentification et du système MME (Modular Model Engine) a été initiée, posant les bases pour les prochaines fonctionnalités. Certaines fonctionnalités évoquées durant ce sprint ne sont toutefois pas encore déployées en production et feront l'objet de travaux lors des prochains sprints.

6.1 Objectifs pour le prochain sprint

Pour le prochain sprint, nous prévoyons de :

— Finaliser les fonctionnalités encore inachevées.

— Mettre en place un premier modèle de reconnaissance léger.

— Commencer les tests sur le modèle afin d'affiner les choix techniques.

— Poursuivre l'amélioration générale de l'application.

7 Liens utiles

— Repository GitHub du projet

https://github.com/invader237/sae5

— Tableau Trello du projet

https://trello.com/b/GN5yM7u6/sae-5

— L'application en ligne

http://neuroom.fr