UNIVERSITÉ DE LORRAINE | IUT Metz Informatique

Département Informatique

SAÉ 5 — Développement avancé

Rapport SAÉ 5 — Sprint 6

Diego TRIVINO

Mathias FRIES

Alexandre KESSELER

Joshua BROCHOT

Table des matières

1 Introduction

Durant ce sprint, nous avons poursuivi le développement des fonctionnalités liées à la reconnaissance d'images, à la gestion des modèles et au *Modular Model Engine* (MME). Les travaux ont principalement porté sur l'amélioration du pipeline d'inférence, l'ajout du prétraitement, la mise en place d'un système de hot reload des modèles, ainsi que l'intégration du stockage externe.

Nous avons également finalisé plusieurs aspects transverses : la refonte de l'authentification, l'intégration complète du flux d'envoi d'images et la sécurisation du déploiement web via un reverse proxy. Ce sprint renforce ainsi la stabilité et la modularité du système.

2 Implémentation du système d'inférence du modèle de reconnaissance

2.1 Amélioration du pipeline d'inférence

Une étape importante a été franchie dans la mise en place du pipeline de reconnaissance d'images. Le système d'inférence est désormais capable de charger un modèle de classification, d'exécuter une prédiction à partir d'une image et de gérer des opérations complémentaires telles que la sauvegarde ou le traitement post-prédiction. Cette base solide permettra prochainement de mener des tests comparatifs sur les performances et la précision.

2.2 Ajout du prétraitement d'images

Les fonctions dédiées au traitement des images avant leur passage dans le modèle ont été centralisées. Elles assurent notamment la conversion, le redimensionnement, la normalisation et les autres transformations nécessaires à un résultat cohérent et reproductible. Cette étape garantit une meilleure stabilité du modèle et prépare le terrain pour la future augmentation de données lors de l'entraînement.

2.3 Évolution de la gestion des modèles

La structure de gestion des modèles IA a été enrichie afin d'intégrer un paramètre configurable déterminant la taille d'entrée attendue par le modèle. Ce paramètre peut désormais être défini pour chaque modèle stocké, avec une valeur par défaut fixée à 384. Cette fonctionnalité offre une plus grande flexibilité lors de l'utilisation de modèles de tailles différentes et permettra des ajustements dynamiques via l'administration.

3 Gestion du stockage des modèles & MME

Dans le cadre du développement du **MME (Modular Model Engine)** — présenté plus en détail au sprint 5 — nous avons souhaité intégrer un système de *hot reload* des modèles. L'objectif de ce mécanisme est de permettre l'amélioration continue du modèle de reconnaissance sans nécessiter un redéploiement complet de l'application.

Grâce au *hot reload*, il devient possible d'entraîner un nouveau modèle à tout moment et de le rendre immédiatement disponible en production. Cette approche permet ainsi de **décorréler complètement le cycle de déploiement de l'application de celui des modèles**, réduisant la friction lors des phases de test et facilitant l'expérimentation.

3.1 Besoin d'un stockage externe

Pour mettre en place ce système, il est nécessaire de disposer d'un stockage **indépendant de l'application**. Deux solutions ont été envisagées dès le début du projet :

— un stockage des modèles dans une branche dédiée du dépôt GitHub ;

— un stockage dans le cloud Microsoft Azure.

Nous avons étudié les avantages et inconvénients de chacune de ces options.

3.2 Comparaison des solutions

Azure

— Aucun coût grâce à l'offre *Azure for Students* (100 € de crédits).

— Bonne fiabilité et disponibilité de la plateforme cloud.

— Récupération des modèles simple via une API officielle.

— En revanche, l'envoi des modèles nécessite également l'utilisation d'une API, ce qui complique l'intégration.

— Mise en place plus lourde pour automatiser la synchronisation des modèles.

GitHub

— Aucun coût d'utilisation.

— Ajout d'un nouveau modèle facilité grâce à un simple `git push`.

— Intégration naturelle avec des workflows CI/CD.

— En revanche, l'application ne peut pas communiquer directement et de manière sécurisée avec le dépôt (nécessite un *pull* ou un *clone*).

— Taille maximale des fichiers limitée selon le type de dépôt.

— Débit parfois réduit en fonction de la taille des modèles.

3.3 Choix retenu

À l'issue de cette analyse, nous avons choisi d'utiliser **GitHub comme solution de stockage des modèles**. En mettant en place un **workflow GitHub Actions**, nous pouvons :

— mettre automatiquement à jour la version locale des modèles en production dès qu'un nouveau modèle est ajouté au dépôt ;

— déclencher automatiquement la requête permettant au MME de recharger ce nouveau modèle.

Nous avons commencé la mise en place de ce système et poursuivrons son intégration au cours des prochains sprints.

4 Mises à jour suite à la Pull Request sur l'authentification

La Pull Request dédiée à l'authentification a permis d'aligner cette fonctionnalité avec l'architecture du projet et d'améliorer la qualité générale du code.

4.1 Refactorisation backend

Le module d'authentification a été restructuré pour utiliser les composants du domaine existants (Catalog, Adapter, Mapper). Les conversions entre entités et DTO sont maintenant centralisées, et la gestion du JWT a été clarifiée : extraction du *user_id*, validation du token et unification des messages d'erreur.

4.2 Gestion d'erreurs améliorée

La plupart des erreurs (email déjà utilisé, identifiants invalides, token incorrect, format du header, etc.) sont désormais gérées proprement dans des handlers dédiés. Les endpoints renvoient des réponses cohérentes, facilitant le traitement côté mobile. Le code a aussi été normalisé pour respecter les règles `flake8`.

4.3 Intégration frontend

La page *Profil* et le composant `AuthForm` ont été ajustés pour consommer correctement les endpoints `/auth/login`, `/auth/register` et `/auth/me`. Une bannière d'erreur informe désormais l'utilisateur en cas d'email invalide, mot de passe incorrect ou problème serveur. La persistance et la validation du token sont assurées via `AsyncStorage`.

5 Flux image

5.1 Mise en place de la méthode POST

L'intégration entre le front-end et le back-end de l'application a été finalisée au moyen de la création d'une méthode `POST` au sein de l'API. Cette méthode permet au front-end de transmettre efficacement des données, notamment des images, vers le serveur. Lors de chaque transmission, une information supplémentaire est envoyée afin de préciser le type d'envoi : soit `analyse`, destinée au traitement par le modèle d'intelligence artificielle pour déterminer la salle correspondante, soit `database`, utilisée pour enrichir la base d'images et améliorer les performances du modèle.

5.2 Gestion de l'envoi d'images

Du côté de l'interface utilisateur, une fonctionnalité d'upload d'image a été intégrée, permettant l'envoi ponctuel de fichiers visuels. Le front-end prend également en charge l'envoi automatique d'une image par seconde lorsqu'un flux continu provenant de la caméra est activé. Cette approche permet de capturer et de transférer régulièrement des images vers le back-end, simulant un flux vidéo léger tout en préservant de bonnes performances.

6 Déploiement et Hébergement

6.1 Reverse Proxy et Sécurisation du site

Le site `neuroom.fr`, qui héberge la version web du projet, a été relié à l'application au moyen du reverse proxy Docker `SWAG`. Cette solution permet la génération et la gestion d'un certificat SSL, assurant ainsi la disponibilité du site en `HTTPS`. Grâce à cette configuration sécurisée, les navigateurs considèrent le domaine comme fiable et autorisent les requêtes d'accès à la caméra, élément essentiel pour le fonctionnement de l'application.

7 Conclusion

Durant ce sprint, nous avons pu implémenter plusieurs fonctionnalités essentielles au bon fonctionnement de l'application tout en poursuivant celles pour lesquelles nous avions pris du retard. Ces avancées nous rapprochent désormais d'une première version fonctionnelle capable de réaliser la reconnaissance des salles de classe à l'aide d'un modèle de test.

7.1 Objectifs pour le prochain sprint

Pour le prochain sprint, nous prévoyons de :

— finaliser la gestion de la reconnaissance d'image ;

— terminer l'implémentation de l'authentification ;

— mettre en place le système d'historique.

8 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

https://neuroom.fr