**Projet Final:**

**Rapport de Sprint #2**

Michael Morin

Département d’Informatique et de Mathématique

Université du Québec à Chicoutimi

8INF892 – Apprentissage Profond

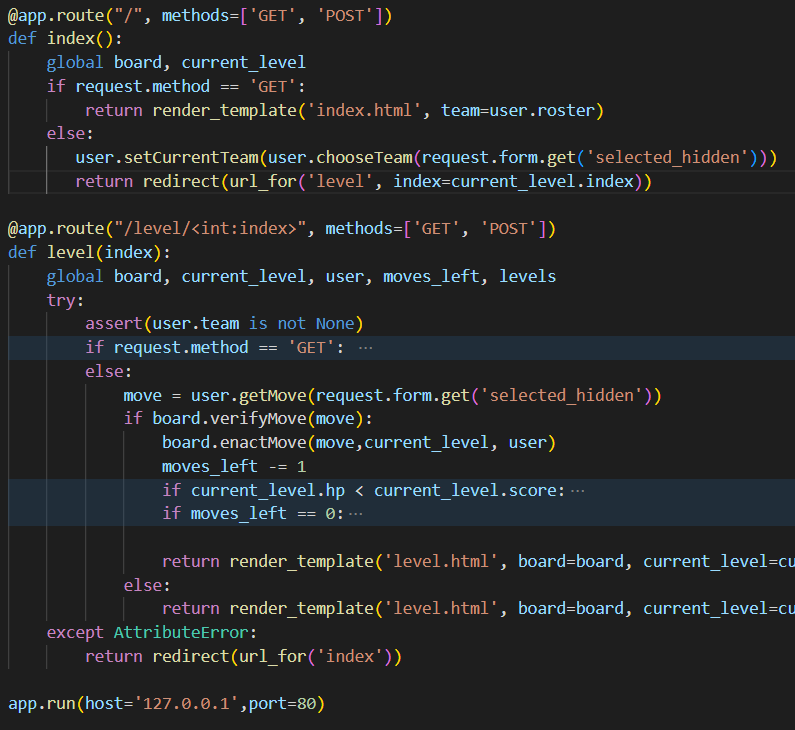
Dr. Kevin Bouchard

7 Avril 2023

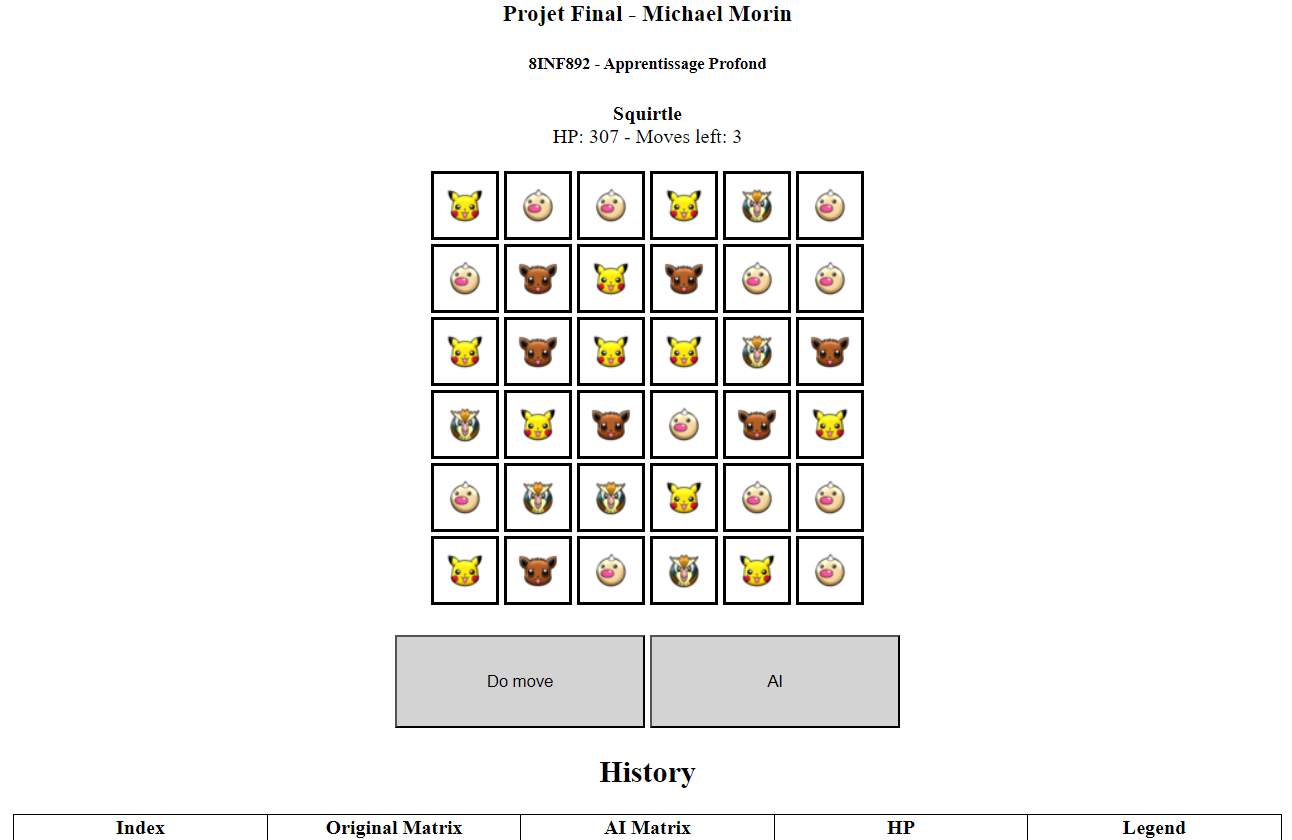
**Rapport de Sprint #2**

**Problématique à adresser**

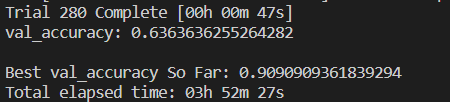
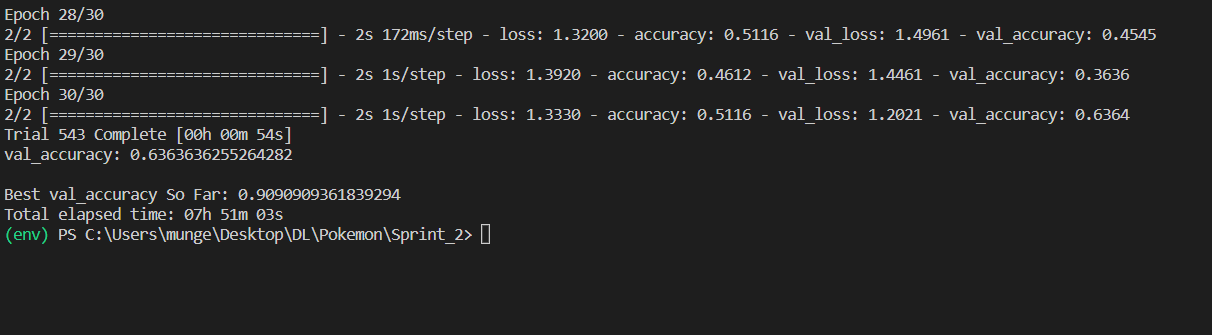
La solution logicielle jusqu’à présent recrée une mini version de *Pokémon Shuffle*. Alors que l’affichage jusqu’à présent s’est fait sur la console, la prochaine étape inclus une transition vers une affichage graphique. Bien que cette partie ne soit pas obligatoire, elle permet de rendre la solution plus facile d’utilisation et répond mieux à sa fonction initiale de mini-jeu. Une partie de cette transition inclut l’ajout d’icônes représentant les pokémons et les obstacles. En fait, cette représentation graphique est nécessaire pour la deuxième grande partie de ce sprint, la création et l’optimisation un CNN de classification d’image. Le but de ce réseau de neurones est de prendre la matrice crée précédemment sous forme graphique afin de recréer la matrice formée par les images pour ensuite l’envoyer à l’agent de renforcement À la fin de ce sprint, le CNN pourrait être réutiliser pour agir directement sur la version originale de *Pokémon Shuffle*.

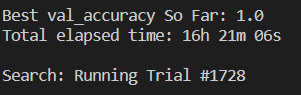
**Travail réalisé**

La transition graphique à une importance moyenne, telle que discuté précédemment. Cependant, elle constitue une valeur ajouté intéressante donc elle sera décrite brièvement. L’interface utilisateur consiste en un serveur Flask pour un affichage web sur 127.0.01:80 (localhost). Deux routes sont disponibles, soient l’index et ‘level/<int>’, pour les méthodes ‘Get’ et ‘Post’. Alors que l’index permet de choisir l’équipe voulu, la route du level permet l’affichage du niveau ainsi que le choix des mouvements. Pour ce qui est des fonctionnalités internes, les fonctions du sprint #1 ont été réutilisé dans leur entièretés. La principale nouveauté provient de l’affichage de l’historique. À ce point-ci, le bouton AI ne fonctionne pas car il va faire appel à l’agent de renforcement.

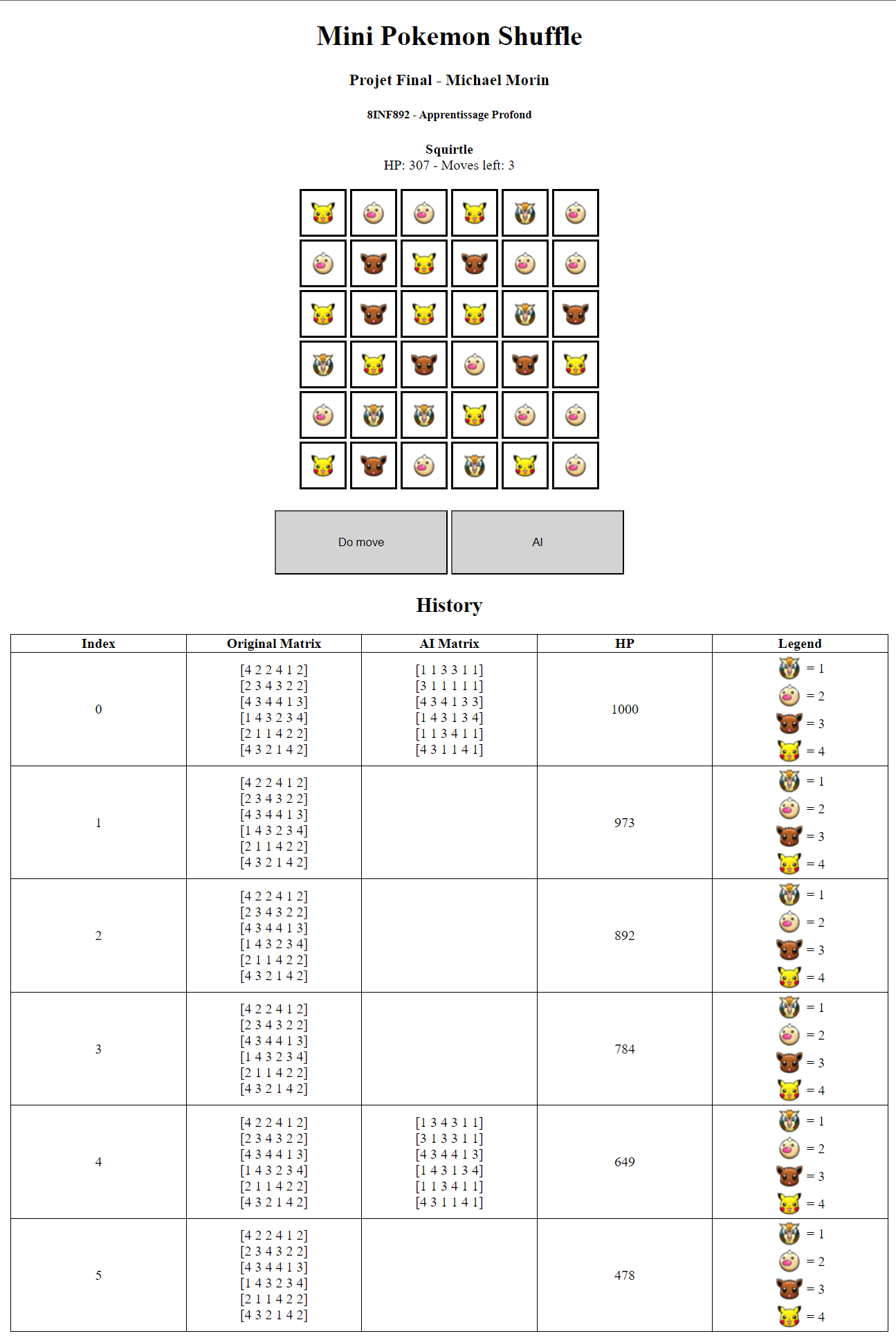
Page d’accueil

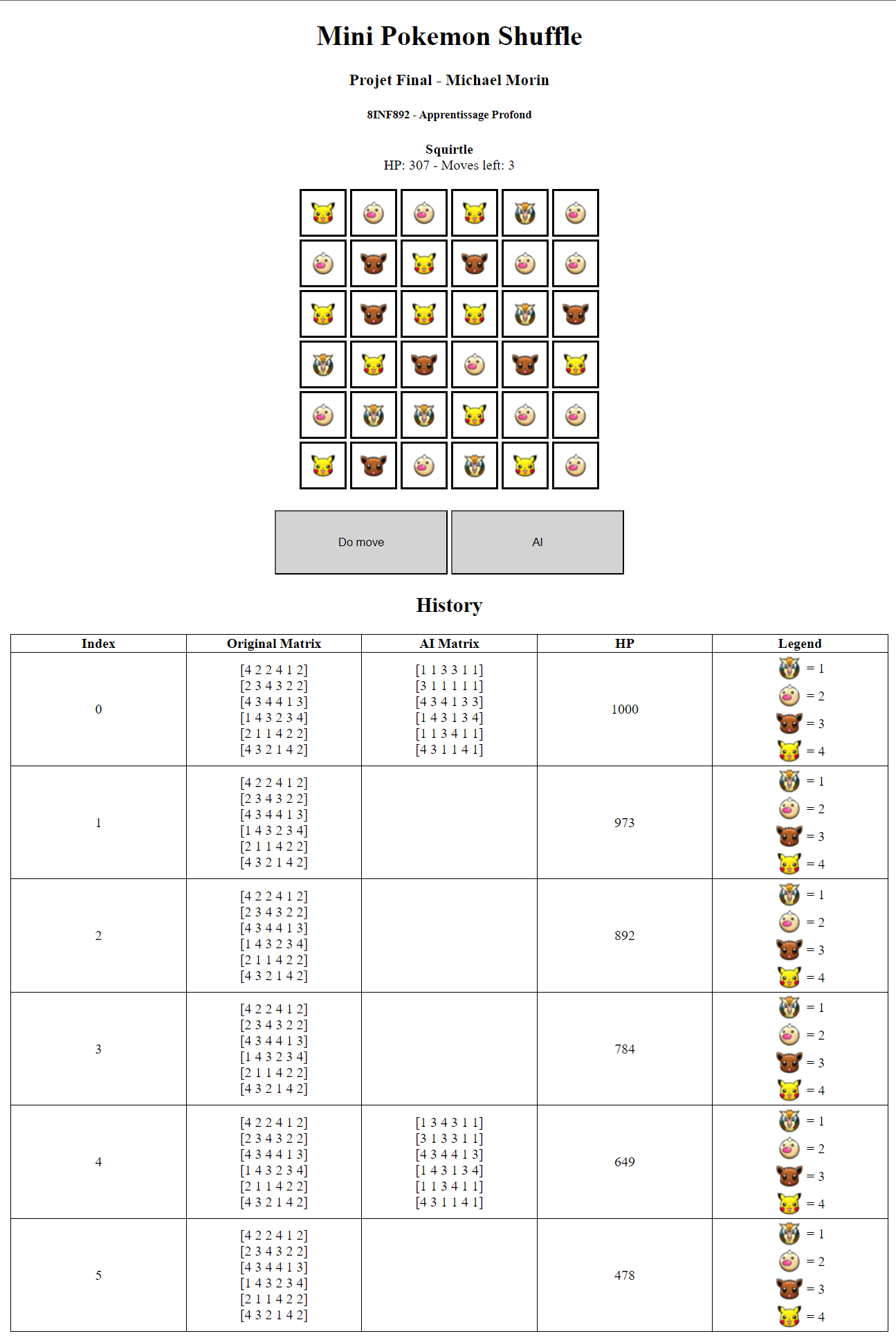
Page de jeu

Bien que l’interface graphique permette d’accéder aux fonctionnalités du jeu, elle ne permet pas de demander un nouvel entrainement du CNN. Le model utilisé par le jeu est sauver dans les fichiers model.h5/.json et ils sont importé au démarrage. Le CNN a été choisi au lieu du MLP car, en général, les performances de ce dernier sont plus basses que sa contrepartie et il tend à être plus long à mettre en place. Pour ce qui est du choix des couches, je suis partie du travail précédant utilisant le MINST puis j’ai fait des ajustements manuels jusqu’à l’obtention d’un modèle avec une exactitude au-delà de 80% au moins une fois sur six. Ensuite, j’ai utilisé Keras Tuner avec des paramètres entre 32 et 128 (step=16) pour les trois filtres et le filtre dense, entre 3 ou 5 pour la taille des noyaux, entre .01 ou .001 pour le taux d’apprentissage et entre 1 et 237 (step=2) pour la taille des lots. Après 348 essaies, les noyaux ont tous été placé à 3, le taux d’apprentissage a été placé à .001 et la taille des lots à été réduit à 3-150 (step=1). En utilisant ces paramètres sur 543 essaies, les valeurs du filtre 2 ont été réduit à 112 ou 128, les filtre 3 et dense ont vu leur minimum rehaussé à 64 et les tailles de lots ne comprennent que 8 valeurs possibles. Avec 280 essaies, les valeurs pour le filtre 3 sont tombés à 112 et 128 alors que le filtre dense 1 a eu ces valeurs en plus de 96. Le premier filtre, de son côté a vu ses valeurs réduites à 64, 80 et 112 alors que la tailles des lots ont perdues les valeurs 5, 7 et 15.

Après ces essaies et quelques autres, les modèles n’était toujours pas consistant. C’est à ce moment que j’ai réalisé que le dataset était dynamique. En fait, pour ce CNN, le dataset a été assemblé avec 258 images de 11 catégories (8 pokemons, 3 obstacles). Chaque image a été coupé, a vu son arrière-plan disparaitre avent d’être redimensionné en 32x32px. Puisque 258 est un nombre d’image beaucoup trop petits, ils ont été transformés en vecteur et passé à un générateur d’image augmentées. Le problème d’inconsistance des modèles provenait du fait que je passais ce générateur au Keras Tuner au lieu d’un dataset fixe. Pour régler ce problème, le générateur a été utilisé pour générer un fichier csv qui est ensuite passer au Tuner. Après près d’une semaine de ‘tuning’, les meilleurs modèles ont enfin été trouvé.

**Bilan et Prochain Sprint**

 Énormément de temps lors de ce sprint a été utilisé pour l’optimisation du modèle. En fait, un modèle avec une exactitude de >90% a été trouvé rapidement, mais mes efforts ont été principalement pour trouver les paramètres donnant les meilleur modèles et le meilleur modèle possible. Un effort important à aussi été donné lors de la création du dataset puisqu’aucun de ce genre n’existait encore. Le jeu permet maintenant de sélectionner l’équipe et de faire les movement graphiquement en cliquant sur les cases avant de confirmer avec un bouton. De plus, l’historique des mouvements, des combos et des interprétations du CNN sont disponible facilement et graphiquement. Pour confirmer le bon fonctionnement du programme, voici une trace d’exécution :



Lors du dernier sprint, l’agent de renforcement sera créé et optimisé. Bien que ce type d’agent va me demander une bonne recherche pour bien comprendre le concept, son implémentation risque de demander un temps moindre comparé à son optimisation. Cependant, une fois l’agent entrainé sauvegardé, il sera ajouté à l’interface graphique, ce qui permettre de compléter le mini-jeu avec un robot pour résoudre les casse-têtes trop difficiles à l’aide d’un bouton.