Rapport projet 3A : Détection de gestes dans le jeu de Shifumi

Rebecca HOUITTE
Charles BOUYROU

5 avril 2024

Résumé

Ce document présente notre travail sur l'amélioration de la reconnaissance de gestes pour un jeu de Shifumi joué contre une intelligence artificielle. Nous nous sommes concentrés sur l'amélioration des performances de la détection des gestes du joueur en temps réel, en intégrant les technologies de vision par ordinateur avancées.

Table des matières

1 Introduction

Dans le cadre du projet de troisième année à l'école Centrale Méditerranée, nous avons choisi de travailler sur l'amélioration de la détection des gestes réalisés par un joueur de shifumi (pierre-papier-ciseaux) face à une intelligence artificielle. Ce projet s'inscrit dans le cadre de notre cursus puisqu'il relève de la computer vision.

2 Contexte

En 2018, l'équipe Qarma du laboratoire LIS a mis au point un jeu de Shifumi face à un agent artificiel [1]. Le joueur joue en temps réel face à la webcam d'un ordinateur. L'image collectée est traitée pour détecter le geste et ainsi réaliser un coup contre un des joueurs artificiels. Une partie est constitué de 20, 30 ou 40 coups. Les joueurs artificiels utilisent des algorithmes qui apprennent et s'adaptent en permanence, essayant de déterminer un pattern dans les gestes du joueur pour prédire son prochain coup et le contrer.

Le jeu a été déployé pour la première fois en 2018 et le meilleur joueur artificiel a remporté 75% des parties jouées. Cependant, un problème majeur a été constaté : les conditions de jeu et de détection du geste fait par le joueur réel ne sont pas optimales.

Dans ce système, la reconnaissance du geste se fait avec la webcam sur fond vert. Un set-up de fond vert a été fabriqué avec une boîte en carton pour le déploiement, avec un système d'éclairage. Depuis le flux vidéo en direct, l'image est binarisée puis une boîte englobante est réalisée sur la main du joueur. Ensuite, un Support Vector Machine (SVM) réalise la tâche de classification pour obtenir le geste fait par le joueur.

Ce modèle possède plusieurs limites. Premièrement, il nécessite un fond vert pour pouvoir jouer au jeu, ainsi qu'un bon éclairage pour le bon fonctionnement de la binarisation de l'image. En effet, si l'éclairage n'est pas bon, la différence de contraste entre la main et le fond n'est pas suffisante pour permettre la binarisation, ce qui porte atteinte à la précision de la boîte englobante puis à la classification par le SVM. De plus, les performances du système de reconnaissance de geste du jeu sont très mauvaises. Aucun chiffre précis n'a été communiqué, mais l'une des conclusions retenues de l'expérience a été une mauvaise détection du geste avec beaucoup de gestes "pierre" détectés.

Notre mission est de trouver un meilleur moyen de réaliser cette tâche de classification du geste joué afin d'améliorer le jeu, et aussi de rendre le système moins contraignant vis-à-vis des conditions de détection par flux vidéo. Enfin, nous visons à intégrer notre solution dans le code source qui nous a été fourni.

3 État de l'art

L'état de l'art présenté dans cette section se concentre sur deux aspects pour le développement de notre projet : les données disponibles pour l'entraînement ou le fine-tuning des modèles, et les modèles et technologies existantes utilisés dans le domaine de la reconnaissance de gestes, en particulier ceux relatifs au jeu du Shifumi.

3.1 Données

RPS Images réelles [2] :
Rock-Paper-Scissors (RPS) Images réelles est une base de données est disponible gratuitement sur Kaggle, répertoriant plus de 2100 images de mains de personnes effectuant les trois gestes du shifumi (environ 700 par classe). Les mains varient en taille et en âge des individus, et certaines portent des manches tandis que d'autres n'en portent pas. Cependant, il n'y a pas de bijoux, de vernis à ongles ou de tatouages, et toutes les personnes ont la peau blanche. Les images sont prises sur fond vert, toujours dans le même sens. Bien que cette base de données soit pertinente pour la classification, nous ne pouvons pas baser entièrement notre système sur ces images car cela pourrait introduire un biais. De plus, la contrainte du fond vert est une limitation que nous aimerions surmonter.

FIGURE 1 – RPS images réelles

RPS Images de synthèse [3] :

Similaire à la base de données précédente, cette base contient environ 2900 images générées à partir de technologies CGI (Computer Generated Imagery), cette fois-ci sur fond blanc. L'avantage est la variété des mains (races, âges, sexes, vernis à ongles). Cependant, certaines mains peuvent parfois apparaître déformées.

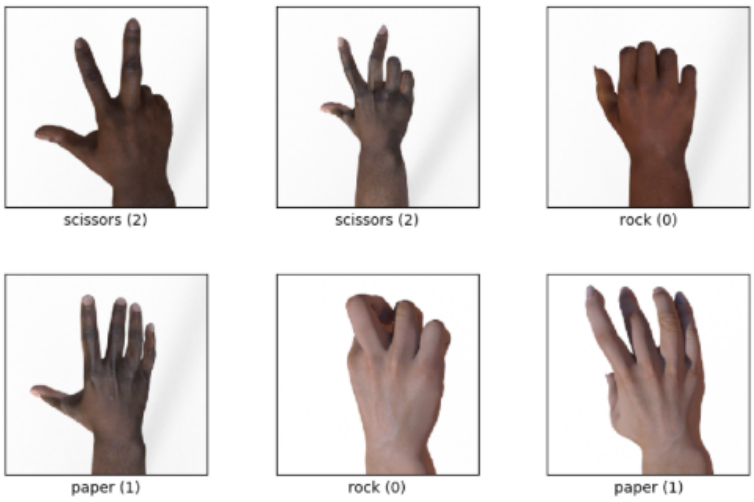

FIGURE 2 – RPS Images de synthèse

RPS Images collectées :
Nous avons constitué notre propre base de données, comprenant environ 110 images par classe, sur des fonds et avec des luminosités variés. L'objectif est d'utiliser ces données pour tester nos modèles dans des conditions réelles.

Pour améliorer la robustesse de nos modèles, nous avons appliqué des techniques d'augmentation de données. En particulier, nous avons effectué des rotations sur nos images afin de diversifier davantage notre ensemble de données et de mieux généraliser notre modèle à différentes orientations des mains.

Nous avons également trouvé d'autres ensembles de données contenant des images de mains effectuant des gestes autres que ceux du shifumi. Cependant, nous n'avons conclu que ces données n'étaient pas nécessaires pour notre projet et ne les avons donc pas utilisées.

3.2 Modèles

Nous avons effectué une revue des modèles et technologies existants en matière de classification d'images. Nous avons jugé pertinent d'explorer les technologies utilisées dans le domaine de la reconnaissance de gestes pour le langage des signes. Parmi les options qui se sont présentées, nous avons examiné Mediapipe [4], un outil largement utilisé pour la reconnaissance de gestes, car il permet une détection efficace des coordonnées du squelette de la main.

En outre, nous nous sommes intéressés à l'utilisation d'un réseau de neurones convolutionnel (CNN) [5] pour la classification multiclasses à partir d'images. Nous avons développé un modèle CNN composé de couches convolutionnelles et de couches denses, de normalisation par Batch et de pooling.

4 Solutions

4.1 CNN

Nous avons entrainé un modèle CNN dont l'architecture est détaillé dans le tableau ci-après. Il a été entraîné sur les données RPS images de synthèse et images réelles, augmentées avec des rotations, avec au total plus de 9000 images. L'entraînement s'est déroulé sur 15 époch, avec une précision de test de 0.94.

Couche (type)	**Forme de sortie**	**Paramètres**
Conv2D	(None, 200, 300, 16)	160
Batch Normalization	(None, 200, 300, 16)	64
MaxPooling2D	(None, 100, 150, 16)	0
Conv2D	(None, 100, 150, 32)	4,640
Batch Normalization	(None, 100, 150, 32)	128
MaxPooling2D	(None, 50, 75, 32)	0
Conv2D	(None, 50, 75, 64)	18,496
Batch Normalization	(None, 50, 75, 64)	256
MaxPooling2D	(None, 25, 37, 64)	0
Flatten	(None, 59,200)	0
Dense	(None, 256)	15,155,456
Dropout	(None, 256)	0
Dense	(None, 3)	771

TABLE 1 – Architecture du CNN

Cependant, lors des tests sur nos propres données (RPS images collectées), les performances du modèle ne sont pas satisfaisantes (voir matrice de confusion). Nos données d'entrée sont trop différentes de nos images de test, avec des variations de fonds et d'éclairages. Par conséquent, nous avons décidé d'écarter cette solution et de nous tourner vers Mediapipe, déjà entraîné à reconnaître les mains même dans des conditions de fond et d'éclairage variables, et quelle que soit la couleur de peau.

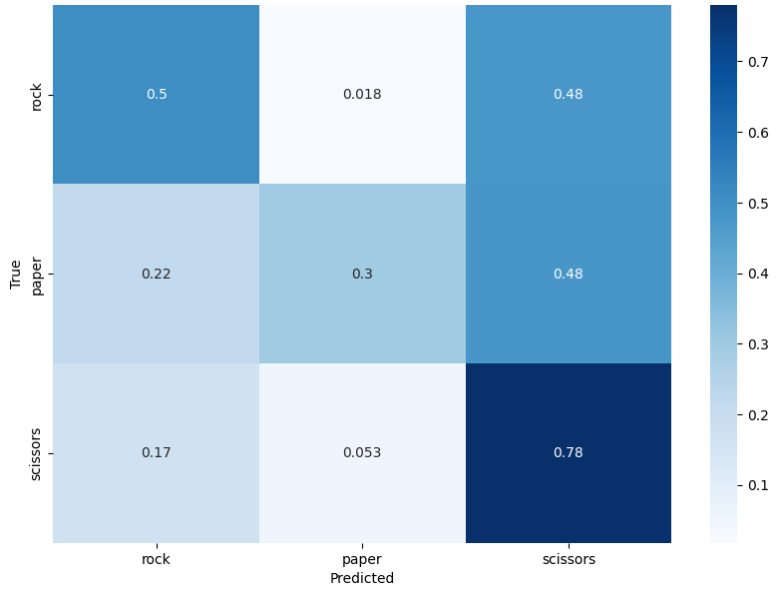

FIGURE 3 – Matrice de confusion du CNN

4.2 Mediapipe

Nous avons exploré les solutions proposées par Google Mediapipe, une plateforme open source offrant des outils et des bibliothèques pour le traitement de signaux multimédias en temps réel. Deux fonctionnalités de Mediapipe ont attiré notre attention :

Hand Landmarks Detection [6] : Il s'agit d'un modèle pré-entrainé qui permet de tracer le squelette de la main. Il est particulièrement performant pour détecter et suivre la main en temps réel sur un flux vidéo, quel que soit l'arrière-plan ou les conditions d'éclairage. Entraîné sur des images de personnes présentant des caractéristiques différentes (âges, sexe, couleur de peau) et dans des conditions variées, il est très robuste. Il prend en entrée une image et fournit en sortie les coordonnées en trois dimensions des 21 points caractéristiques de la main.

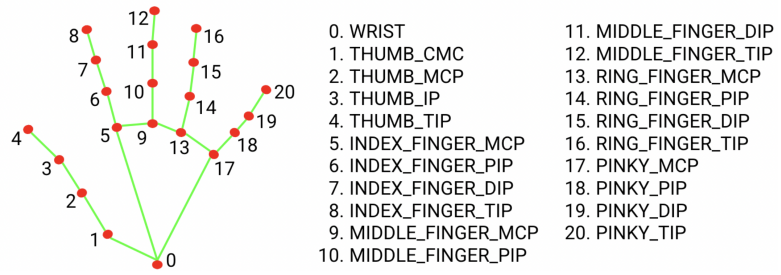

FIGURE 4 – Squelette de la main de Mediapipe

Hand Gesture Recognizer [7] : Ce modèle utilise Hand Landmarks Detection précédent pour classifier les gestes effectués par la main. Il prend en entrée une image traitée par le premier modèle pour extraire les coordonnées de la main, puis un réseau de neurones entièrement connecté classifie le geste. Pour le finetuner à la classification des gestes de notre choix, il faut lui fournir un ensemble d'images labellisées par classe, avec une classe "none" par défaut.

Nous avons utilisé le Reconnaiseur de Gestes de Mediapipe pour apprendre à classifier les gestes du jeu de Shifumi. Nous avons fine-tuné le modèle sur le jeu de données RPS images réelles composé de 2100 images. Nous avons choisi de ne pas utiliser les images de synthèse car la détection du squelette de la main n'était pas très performante sur celles-ci. Pour faire fonctionner le modèle, nous avons ajouté une classe "None" comprenant 100 images représentant des gestes non pertinents pour le jeu de Shifumi trouvés dans une autre base de données. Cependant, les résultats initiaux étaient insatisfaisants, avec plus de la moitié des images "rock" prédites comme "none". Nous avons alors décidé de faire de la classe "rock" la classe par défaut. Les prédictions obtenues sur nos données RPS images collectées sont très bonnes, surpassant les résultats obtenus avec le CNN. Ces résultats sont présentés dans la figure 5 (la colonne "unpredicted" correspond aux images pour lesquelles la main n'a pas été détectée). Cependant, définir une classe par défaut pose un problème conséquent car cela pourrait fausser les prediction.

4.3 Mediapipe avec un KNN

Ensuite, nous avons développé un autre modèle exploitant l'outil de détection du squelette de la main de MediaPipe. Notre approche consiste à utiliser ce modèle pour l'entraînement d'un algorithme de classification des k plus proches voisins (KNN). Pour ce faire, nous avons généré des données d'entraînement en extrayant les coordonnées du squelette de la main sur les images du

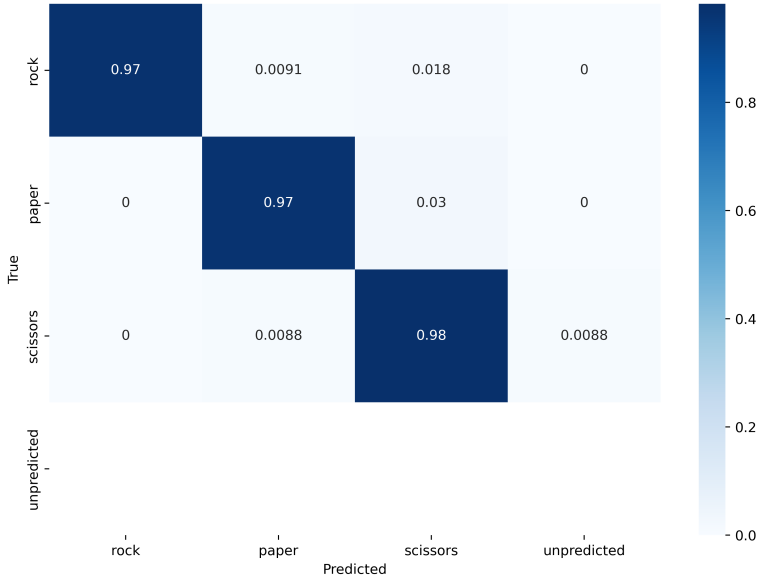

FIGURE 5 – Matrice de confusion de Mediapipe Hand Gesture Recognizer

jeu de données RPS images réelles, soumises à des rotations aléatoires. Nous avons ensuite entraîné un classificateur KNN sur ces coordonnées. Ainsi, la solution que nous proposons est la suivante : une image capturée en temps réel est soumise à MediaPipe pour extraire ses coordonnées du squelette de la main, puis ces coordonnées sont utilisés comme entrée pour le KNN afin de déterminer à quelle classe ils appartiennent. Les résultats obtenus lors des tests nos données RPS images collectées sont les suivant : voir figure 6.

Nous constatons que les résultats obtenus avec les deux modèle utilisant Mediapipe sont assez satisfaisants. Après évaluation, nous avons choisi de conserver le modèle utilisant le KNN en raison de sa simplicité et de son efficacité. En effet, ce modèle nécessite uniquement l'utilisation d'un fichier CSV contenant les coordonnées des mains par classe, et reste très basique par rapport au Gesture Recognizer de MediaPipe, qui utilise des couches denses et nécessite l'installation de modules supplémentaires de MediaPipe.

4.4 Détails de la solution

La solution que nous avons retenue pour réaliser la tâche de détection et de classification du geste est celle qui combine les coordonnées de la main de Mediapipe et le KNN comme classifier. Premièrement, nous utilisons Media-

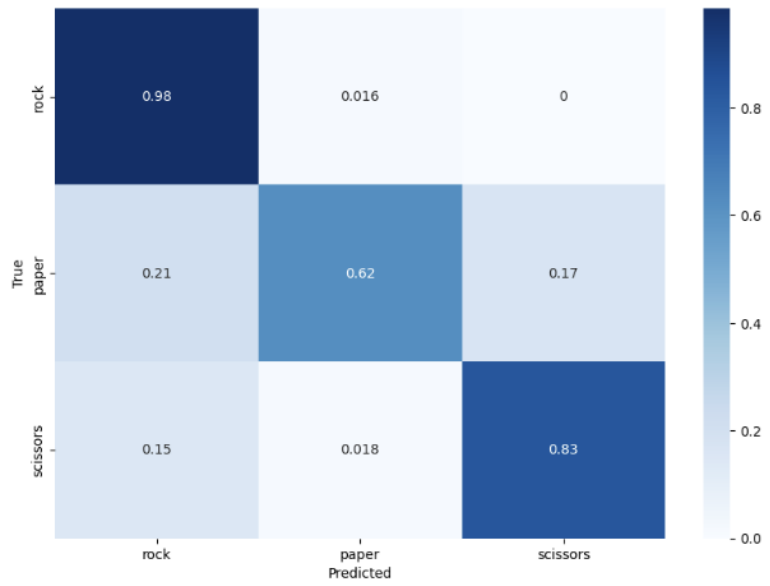

FIGURE 6 – Matrice de confusion de Mediapipe avec un KNN

pipe afin d'obtenir un "squelette" de la main du joueur en temps réel. Ensuite, nous utilisons un KNN entraîné sur nos données. Pour mieux comprendre notre chemin de pensée, expliquons pourquoi nous avons choisi Mediapipe et en quoi cela consiste.

MediaPipe est conçue pour faciliter le développement d'applications de vision par ordinateur en offrant une boîte à outils permettant le traitement d'images et de vidéos et notamment la détection de gestes.

Pour la reconnaissance de gestes, MediaPipe utilise généralement des modèles d'intelligence artificielle basés sur des réseaux de neurones convolutifs (CNN) et parfois des réseaux neuronaux récurrents (RNN) pour analyser les séquences temporelles des mouvements. Dans notre cas, le modèle que nous utilisons est le modèle MediaPipe Hands, qui utilise un modèle CNN pour la détection de la main et des points de repère (landmarks) des doigts. Ce modèle est capable d'identifier la position et l'orientation de la main et des doigts, permettant ainsi de reconnaître différents gestes. Le modèle est composé d'un "Two step neural network pipeline with single-shot detector and following regression model running on the cropped region.". En entrée, le modèle prend un flux vidéo ou une image RGB de taille arbitraire. En sortie, il retourne une liste pour chaque main détectée qui contient 21 landmarks en 3D et un float qui représente la probabilité de détection d'une main.

Concernant l'entraînement de ce modèle, nous ne savons pas sur quel dataset il a été fait, ni même sur la quantité des données qu'il a reçu. Aussi,

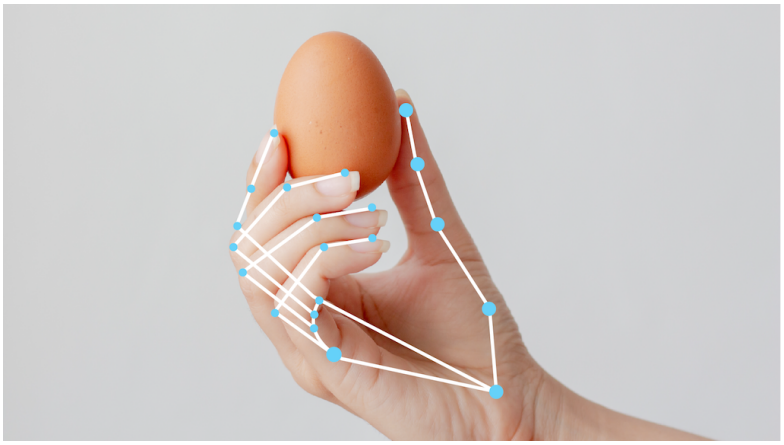

FIGURE 7 – Hand Landmarks detection de Mediapipe

la plupart des images qui ont servi à l'entraînement ont été capturées via les appareils frontaux et arrières de téléphones portables. Elles ont été capturées dans des situations de la vie réelle avec différentes luminosités, bruit, flou de mouvement via une application de réalité augmentée.

Nous avons utilisé les données RPS images réelles (700 images par classe). Ces données sont passées en input du modèle Mediapipe pour obtenir les landmarks en 3D de chaque main présente sur les images. Pour finir notre data processing, nous avons normalisé les landmarks en 3D obtenus pour chaque main. Cette étape consistait à les redimensionner de manière à ce que les coordonnées sur chaque dimenssions soient comprise entre 0 et 1. Cela permet au modèle de mieux généraliser à partir des données, car il n'est pas biaisé par la taille ou la position des mains dans les différentes images. Nous avons également transformé les coordonnées cartésiennes en coordonnées sphériques et ajouté des rotations aléatoires pour rendre le modèle robuste face à la rotation.

Nous avons ensuite divisé notre dataset en trois ensembles distincts : un ensemble d'entraînement, qui comprenait 70% des données, un ensemble de validation, qui en comprenait 15 %, et un ensemble de test, qui constituait les 15 % restants. Cela nous a permis d'entraîner notre modèle sur un grand nombre d'exemples, de choisir le nombre de voisins par validation croisée, et enfin de tester la performance du modèle sur l'ensemble de test.

La validation croisée consiste à explorer une gamme de valeurs possibles pour k afin de trouver un équilibre entre précision et généralisation. Une fois le meilleur k identifié, nous avons utilisé cette valeur pour entraîner notre modèle KNN sur l'intégralité du dataset d'entraînement (entrainement +

validation). L'approche KNN, malgré sa simplicité conceptuelle, a prouvé une grande efficacité avec un score de test de 0.98, notamment grâce à la richesse des features extraites des landmarks en 3D des mains, qui contiennent des informations spatiales permettant la distinction entre les gestes de pierre, papier, et ciseaux.

Les résultats obtenus sur l'ensemble de test ont été très bons. Cependant les tests sur nos données RPS images collectées sont moins bons (voir figure 6). En effet, les images de ce dataset sont prises dans des conditions extrêmes (mauvais éclairage, images floues, etc.). Nous avons aussi pu tester notre KNN avec un flux vidéo en direct dans des conditions réelles. Nous trouvons ces résultats satisfaisants. De profil, avec un éclairage non standardisé, le modèle n'a aucune difficulté à classer les gestes que l'on fait à la webcam.

Pour finir, nous avons décidé de rajouter les données RPS images collectées dans l'entraînement du KNN pour le rendre plus robuste. Nous ne pouvons pas fournir de score du nouveau modèle obtenu car toutes les données ont été utilisées pour l'entraînement. Nous avons testé le modèle en direct et les résultats sont plus que satisfaisants. En effet, lors des tests en temps réel, le modèle a démontré une bonne capacité à classifier les gestes avec précision, même dans des conditions variables telles que des éclairages changeants et des angles de vue différents.

5 Intégration du modèle dans le code source

Maintenant que nous avons un modèle avec de bonnes performances pour la tâche de classification du joueur réel, il nous a fallu l'intégrer dans le code source fourni.

Pour réaliser cette partie, nous avons dû nous intéresser au fonctionnement des fichiers à l'intérieur du code source. Nous nous sommes aperçus que pour faire fonctionner le jeu, il fallait exécuter deux fichiers successivement. Dans la réalité, il faut brancher deux ordinateurs afin d'établir une connexion Ethernet entre les deux. Ensuite, il faut lancer un script sur chaque machine. Chaque script contient une partie de connexion Ethernet afin d'établir la connexion entre les deux machines. Un fichier s'occupe de lancer l'interface de jeu, c'est-à-dire afficher les scores, le prochain coup joué par l'agent artificiel, etc. Tandis que l'autre s'occupe de détecter le geste fait par le joueur réel et de le communiquer à l'autre machine. Ainsi, il nous suffisait juste d'enlever la brique déjà existante relative à la détection du geste du joueur réel et de la remplacer par notre nouvelle brique qui fait cette tâche.

Nous avons identifié le fichier qui permet l'exécution de la détection du geste. Nous l'avons modifié pour intégrer notre solution de détection de la

main et de classification du geste joué grâce à MediaPipe et le KNN. Nous avons respecté les structures du modèle précédent en utilisant la programmation orientée objet et en définissant les classes nécessaires à l'exécution du reste du système. Malheureusement, nous ne sommes pas parvenus à faire fonctionner le système global. En effet, nous ne sommes jamais parvenus à clairement identifier la source du problème. Il semblerait que le jeu ne se déclenche pas à cause des conditions de démarrage qui ne sont pas satisfaites. En effet, dans le système précédent, il fallait notamment détecter un fond vert pour lancer le système. Les fichiers étant nombreux et avec très peu de documentation, nous sommes parvenus à intégrer notre partie de détection de geste du joueur réel dans le code, mais nous ne sommes pas parvenus à rétablir les connexions entre les fichiers qui font que le jeu puisse être fonctionnel.

6 Axes d'amélioration

L'axe d'amélioration majeur consiste à parvenir à réussir l'intégration complète dans l'architecture du jeu de shifumi fournie. Cela aurait été une belle réussite pour nous de présenter une démo du jeu de shifumi en ayant modifié la stratégie de détection du geste du joueur réel.

De plus, nous avons utilisé un module de MediaPipe qui est à ce jour open source mais qui ne le sera peut-être plus dans un futur proche ou lointain. Ainsi, il pourrait être judicieux d'utiliser d'autres modèles de CNN pour réaliser cette tâche de détection afin de se passer de MediaPipe.

7 Conclusion

Nous aimerions conclure en disant que nous avons sûrement sous-estimé cette tâche d'intégration et nous ne pensions pas que cela allait être la partie la plus difficile du projet. En effet, c'est une étape que nous avions placée en fin de projet car cela nous semblait logique de d'abord construire notre brique avant de réfléchir à où et comment la placer. Nous nous sommes laissé impressionner par la tâche de détection qui nous attendait. N'ayant pas beaucoup d'expérience pratique dans ce domaine, nous sommes tout de même fiers des nombreux enseignements que ce projet nous a permis de tirer, que ce soit dans le domaine de l'intelligence artificielle ou de la gestion de projet.

Centrale Méditerranée

Références

[1] Équipe Qarma. *Shifumi-IA : saurez-vous être imprévisible ?* Équipe d'AppRentissage de Marseille, Laboratoire d'Informatique et Systèmes, 2018. https://qarma.lis-lab.fr/shifumi-ia/.

[2] D. Freeman. *Rock Paper Scissors Dataset*, Kaggle, [En ligne]. Disponible sur : https://www.kaggle.com/datasets/drgfreeman/rockpaperscissors?select=rock.

[3] S. Mal. *Rock Paper Scissors Dataset*, Kaggle, [En ligne]. Disponible sur : https://www.kaggle.com/datasets/sanikamal/rock-paper-scissors-dataset. [Accédé le : 5 avril 2024].

[4] MediaPipe Solutions. *Introducing MediaPipe Solutions for On-Device Machine Learning*, Google for Developers, May 11, 2023. [En ligne]. Disponible sur : https://developers.googleblog.com/2023/05/introducing-mediapipe-solutions-fo

[5] Aradbenmenashe. (2024, February 1). *Rock Paper Scissors Classification 97.9%aCC*. Kaggle.[En ligne]. Disponible sur : https://www.kaggle.com/code/aradbenmenashe/rock-paper-scissors-classificatio avril 2024].

[6] MediaPipe Hands. *MediaPipe Hand Landmark Detection*, Google Developers. [En ligne]. Disponible sur : https://developers.google.com/mediapipe/solutions/hands.

[7] MediaPipe Gestures. *MediaPipe Gesture Recognizer*, Google Developers. [En ligne]. Disponible sur : https://developers.google.com/mediapipe/solutions/gestures. [Accédé le : 5 avril 2024].