Stage de Seconde chez Orange Innovation

Nom du Stagiaire: Solal Vazeille **Classe:** Seconde **Lycée:** Jean-Jaurès St Clément de Rivière (34) **Période de Stage:** [du 23 au 27 juin 2025] **Entreprise d'Accueil:** Orange **Tuteur de Stage:** [Vincent Frey - Ingénieur-chercheur]

Table des Matières

---

1. Introduction

Ce rapport détaille les activités menées lors de mon stage de seconde au sein de l'entreprise Orange. Mon objectif principal durant cette période était de m'immerger dans le domaine de l'intelligence artificielle (IA) et de la programmation, en particulier avec le langage Python. J'ai eu l'opportunité de travailler sur **deux projets distincts et complémentaires**, à savoir la création d'une IA capable de reconnaître les chiffres manuscrits et le développement d'un jeu de voiture conçu pour l'entraînement d'une IA à la conduite autonome. Ces projets m'ont permis d'appréhender concrètement les étapes de conception, de

développement et d'optimisation d'une IA, tout en bénéficiant de l'assistance précieuse d'intelligences artificielles génératives telles que ChatGPT et Claude.

Ce rapport a été généré avec l'IA Notebook LM dans sa première version à partir de mes notes personelles et des scripts python que j'ai développé. Je l'ai ensuite amendé à la main pour en faire mon rapport. L'IA n'a pas écrit le rapport, elle m'a aidé à organisé mes notes et à reformuler dans un langage plus professionnel.

2. Le Contexte du Stage : Découverte de Python et de l'IA

Mon stage a débuté par une phase de familiarisation avec l'environnement de développement Python. Bien que j'aie rencontré des difficultés initiales pour faire fonctionner Python sur mon propre ordinateur, l'assistance de mon tuteur et l'envoi des scripts par e-mail m'ont permis de surmonter ces obstacles. Cette étape a été cruciale pour me lancer dans les projets pratiques.

L'approche choisie pour ces projets a été d'utiliser des IA génératives comme assistants de codage. Plutôt que de partir de zéro, j'ai formulé des requêtes précises, appelées "prompts", à ChatGPT et Claude pour qu'ils génèrent des bases de code et m'expliquent leur fonctionnement. Cette méthode m'a permis d'accélérer le développement et d'acquérir rapidement des connaissances techniques malgré mon niveau en seconde. Je me suis servi de ChatGPT pour m'aider à reformuler ma demande sous forme d'un prompt adapé aux IA génératives que j'ai ensuite donné à Claude pour qu'il fasse le développement logiciel.

3. Projet 1 : Création d'une IA de Reconnaissance de Chiffres Manuscrites

3.1. Genèse du Projet et Interactions avec les IA Généralistes

Le premier projet consistait à créer une IA en Python capable de reconnaître les chiffres manuscrits de 0 à 9. J'ai commencé par demander à ChatGPT de me générer un prompt détaillé pour Claude. Ce prompt spécifiait le besoin d'un code Python complet (avec des bibliothèques comme TensorFlow ou Keras) et des explications simplifiées, adaptées à mon niveau (sans les concepts avancés en IA ni les maths de lycée).

Claude a initialement fourni un code utilisant la bibliothèque **TensorFlow**. Cependant, cette version s'est avérée incompatible avec ma version de Python. Aprés intervention de mon tuteur, j'ai demandé à Claude de réécrire le code en utilisant la bibliothèque **PyTorch**. Le code PyTorch a fonctionné correctement, mais il manquait une fonctionnalité essentielle : la possibilité de tester l'IA avec nos propres chiffres manuscrits. Après cette demande, Claude a fourni un nouveau code intégrant une interface pour cette fonctionnalité.

Cependant, les premiers tests de l'IA ont montré que ses performances n'étaient pas à la hauteur de mes attentes. J'ai alors décidé de demander à Claude de concevoir une IA plus performante, capable d'être entraînée plus longtemps.

L'idée était de modifier le processus d'entraînement pour qu'il ne crée pas un nouveau fichier à chaque fois, mais réutilise et mette à jour le même fichier, permettant ainsi un entraînement **incrémental et prolongé**.

3.2. Architecture et Fonctionnement de l'IA (PyTorch)

L'IA de reconnaissance de chiffres est basée sur un **réseau de neurones** développé avec la bibliothèque PyTorch. Son fonctionnement repose sur le principe d'une **fonction mathématique y=f(x)**. * **L'entrée (x)** est une image d'un chiffre manuscrit, représentée sous forme d'une matrice de pixels en niveaux de gris (28x28 pixels). * **La sortie (y)** est un vecteur de taille 10, où chaque élément représente la probabilité que l'image corresponde à un chiffre donné (de 0 à 9).

J'ai demandé à l'IA Notebook LM l'expliquer le code. Voici ce qu'elle écrit :

L'architecture du réseau de neurones, nommée `ReseauChiffres`, se compose de plusieurs couches linéaires (`nn.Linear`) et de couches de régularisation (`nn.Dropout`). * `self.fc1 = nn.Linear(28*28, 128)`: La première couche prend les 28x28 pixels de l'image (784 pixels) et les transforme en 128 neurones. * `self.fc2 = nn.Linear(128, 64)`: Une couche intermédiaire qui réduit les 128 neurones à 64. * `self.fc3 = nn.Linear(64, 10)`: La dernière couche produit 10 sorties, correspondant aux 10 chiffres possibles (0-9). * Des **fonctions d'activation ReLU** (`F.relu`) sont appliquées après les couches `fc1` et `fc2`. Ces fonctions introduisent de la non-linéarité, permettant au réseau d'apprendre des motifs complexes. * Des **couches `Dropout`** (`self.dropout = nn.Dropout(0.2)`) sont utilisées avec un taux de 0.2. Le dropout aide à prévenir le surapprentissage en désactivant aléatoirement 20% des neurones pendant l'entraînement, forçant le réseau à ne pas trop dépendre d'un seul chemin d'information. * Enfin, une fonction `F.log_softmax` est appliquée à la sortie de la dernière couche. Le LogSoftmax est couramment utilisé pour les problèmes de classification, car il normalise les sorties en probabilités log-linéaires, ce qui est compatible avec la fonction de perte `nn.NLLLoss` utilisée pour l'entraînement.

Mon tuteur m'a expliqué le fonctionnement avec des ceoncepts adaptés à un élève de seconde : voici ce que j'ai retenu :

Le réseau de neurones est une **très grande fonction paramétrique** composée de nombreux neurones connectés en couches. Tous les poids (w) et les biais (b) de ce réseau sont des paramètres qui sont **ajustés au fur et à mesure de l'apprentissage automatique**. Le code est configuré pour vérifier si un GPU est disponible (`torch.cuda.is_available()`) et l'utiliser si c'est le cas, car l'entraînement est beaucoup plus rapide sur GPU.

3.3. Entraînement Incrémental et Optimisation des Performances

Pour améliorer les performances de l'IA, le script `IA_nombres_perso3.py` a été conçu pour permettre un **entraînement incrémental**. Cela signifie que l'IA

peut être entraînée sur plusieurs sessions, reprenant là où elle s'est arrêtée, plutôt que de tout recommencer à chaque fois.

- **Chargement des données MNIST**: Les données d'entraînement et de test proviennent du célèbre dataset MNIST, qui contient des milliers d'images de chiffres manuscrits.
- **Fichiers de sauvegarde**: Trois fichiers sont utilisés pour la persistance des données:
  - `modele_chiffres_pytorch.pth`: Sauvegarde l'état (les poids et les biais) du modèle entraîné.
  - `optimiseur_pytorch.pth`: Permet de reprendre l'entraînement exactement là où il s'est arrêté, avec la même dynamique d'apprentissage.
  - `historique_entrainement.json`: Contient les métriques d'entraînement (pertes, précisions), l'époque actuelle, la date de la dernière sauvegarde et la précision finale, pour un suivi complet des progrès.
- **Fonctions de sauvegarde et de chargement (`sauvegarder_modele_complet`, `charger_modele_existant`)**: Ces fonctions gèrent la persistance du modèle et de son historique. Lors du démarrage du script, il tente de charger un modèle existant et son historique. Si aucun modèle n'est trouvé ou si une erreur survient, il initialise un nouvel entraînement.
- **Processus d'entraînement**: L'entraînement se déroule par "époques". Une époque correspond à un passage complet de toutes les données d'entraînement à travers le réseau. Pour chaque époque:
  - **Phase d'entraînement (`entrainer_une_epoque`)**: Le modèle est mis en mode entraînement (`model.train()`). Les données sont passées à travers le réseau, et les poids du modèle sont mis à jour. Les métriques d'entrainement sur l'ensemble du jeu d'entraînement sont enregistrées.
  - **Phase de test (`tester`)**: Le modèle est mis en mode évaluation (`model.eval()`). Les données de test sont utilisées pour évaluer la précision du modèle sur des données qu'il n'a jamais vues, sans aucune mise à jour des poids. Cela permet de mesurer la capacité de généralisation de l'IA.
- **Suivi des progrès**: Après chaque époque, la précision d'entraînement et de test est affichée. Le modèle, l'optimiseur et l'historique sont sauvegardés automatiquement, garantissant que les progrès ne sont pas perdus. Une fois l'entraînement terminé, la précision finale est affichée.
- **Visualisation**: Le script génère des graphiques de l'évolution de la précision (entraînement et test) et de la perte au fil des époques. Ces graphiques sont essentiels pour comprendre comment l'IA apprend et pour détecter des problèmes comme le surapprentissage. Ils incluent même un zoom sur les 10 dernières époques si l'entraînement est long.

3.4. Interface Utilisateur et Capacités de Test

Le script `IA_nombres_perso_interface2.py` fournit une interface interactive pour tester l'IA entraînée. Cette interface charge le modèle et l'historique existants.

Les fonctionnalités principales incluent : * **`predire_image_perso(chemin_image)`**: C'est la fonctionnalité clé pour laquelle j'ai demandé des améliorations. Elle permet de fournir une image personnalisée (par exemple un chiffre dessiné par l'utilisateur) et d'obtenir la prédiction de l'IA. * **Préparation de l'image**: L'image fournie est d'abord convertie en niveaux de gris si nécessaire, redimensionnée à 28x28 pixels (le format attendu par le modèle MNIST), et les couleurs peuvent être inversées si le fond est blanc (pour s'adapter aux données MNIST qui ont des chiffres clairs sur fond sombre). Les valeurs des pixels sont ensuite normalisées. * **Prédiction et affichage**: L'IA effectue la prédiction, et le résultat principal (le chiffre détecté et sa confiance) est affiché. L'interface montre l'image originale et l'image telle que l'IA la voit (après prétraitement). Elle affiche également le **top 3 des prédictions avec leurs confiances**. Un indicateur visuel coloré en fonction du niveau de confiance est également présent pour faciliter l'interprétation. * **`tester_chiffre(index)`**: Cette fonction permet de tester l'IA sur un chiffre spécifique tiré directement du jeu de données de test MNIST, affichant l'image, le vrai label, la prédiction de l'IA et la confiance. * **`predire_plusieurs_images(dossier_path)`**: Cette option permet de traiter un dossier entier d'images et d'obtenir un résumé des prédictions pour toutes les images trouvées. * **`creer_image_test(chiffre)`**: Une fonction utilitaire pour générer une image simple d'un chiffre donné, utile pour des tests rapides. * **`afficher_historique_complet()`**: Affiche toutes les statistiques d'entraînement sauvegardées, y compris des graphiques détaillés de l'évolution de la précision et de la perte au fil du temps. Cela inclut l'amélioration totale et la meilleure époque atteinte. * **`reinitialiser_modele()`**: Offre la possibilité de supprimer toutes les sauvegardes du modèle et de l'historique, permettant de recommencer l'entraînement à zéro.

L'interface est conçue pour être conviviale, avec un menu interactif qui guide l'utilisateur à travers les différentes options. Elle fournit également des informations sur le modèle chargé, comme le nombre total de paramètres et l'utilisation du GPU.

3.5. Concepts d'IA Appliqués

Ce projet m'a permis de découvrir plusieurs concepts fondamentaux de l'IA : * **Apprentissage Supervisé**: L'IA apprend à partir de données d'entraînement labellisées (images de chiffres avec leur valeur correcte). * **Réseaux de Neurones**: Comprendre que les réseaux de neurones sont des fonctions paramétriques complexes qui apprennent à mapper des entrées à des sorties en ajustant des poids et des biais. * **Entraînement et Optimisation**: Le processus itératif de minimisation d'une fonction de perte (erreur) via des "optimiseurs". * **Per-**

sistance de Modèle: L'importance de sauvegarder et de charger les modèles pour ne pas perdre l'entraînement et permettre l'entraînement incrémental.

4. Projet 2 : Développement d'un Jeu de Voiture pour l'Entraînement d'une IA

Le deuxième projet visait à créer un jeu de voiture simple qui servirait de plateforme pour entraîner une IA à la conduite autonome, en utilisant la méthode de l'apprentissage par renforcement (Q-learning). L'objectif à long terme est d'entraîner une IA à rester sur la route.

4.1. Développement Itératif du Jeu avec l'Aide de Claude

Le développement du jeu a été un excellent exemple de **développement itératif avec l'aide d'une IA** (Claude) et de la capacité de l'IA à corriger ses propres erreurs. 1. **Première version - Route Statique**: J'ai d'abord demandé à Claude un jeu de voiture très simple où l'utilisateur pourrait déplacer une voiture de gauche à droite sur une route unie. Claude a fourni un code Pygame basique avec une voiture rouge et un fond gris représentant la route. 2. **Deuxième version - Route et Environnement**: Pour l'entraînement de l'IA, il était nécessaire d'ajouter autre chose que la route. J'ai donc demandé à Claude d'inclure des zones "hors route" (comme l'herbe) et une fonction pour vérifier si la voiture était sur la route (`is_car_on_road()`). Le jeu a alors affiché un fond vert pour l'herbe et un rectangle gris pour la route, avec un texte "On Road" ou "Off Road!" et un score basé sur le temps passé sur la route. 3. **Troisième version - Route Défilante**: J'ai remarqué que la route ne défilait pas, ce qui ne donnait pas l'impression de mouvement. Claude a corrigé cela en ajoutant un effet de défilement des lignes de la route, créant ainsi une illusion de mouvement vers l'avant. 4. **Quatrième version - Route Qui Tourne**: Pour rendre le jeu plus difficile et réaliste pour l'IA, j'ai demandé que la route puisse tourner. Cette version a introduit un système de "segments de route" dynamiques qui changent de direction progressivement grâce à un mouvement sinusoïdal. 5. **Correction d'erreurs visuelles**: Lors de cette dernière étape, j'ai rencontré des problèmes de rendu ("bizarre") que j'ai soumis à Claude via des captures d'écran. **C'est un point notable de ce stage : Claude a démontré sa capacité à comprendre et corriger des erreurs visuelles complexes à partir d'images, ce qui est une fonctionnalité très avancée**. Les corrections apportées ont permis une courbe plus fluide, des lignes centrales cohérentes et une meilleure gestion des segments de route.

Avec mon tuteur, nous avons été surpris par la rapidité de Claude à développer ce jeu. En particulier, pour la troisième version, nous avons fait deux captures d'écran que nous avons joint à la conversation pour expliquer le problème à Claude, et c'était impressionnant de voir comment il a compris les corrections à apporter.

4.2. L'Environnement de Jeu : Route Aléatoire et Capteurs

La version avancée du jeu (script `learn4.py`) a été spécifiquement développée pour l'entraînement d'une IA, en rendant l'environnement plus dynamique et en équipant l'IA de "capteurs".

- **Génération de Route Aléatoire**: La route ne suit plus une courbe sinusoïdale fixe. Elle est désormais générée de manière procédurale et aléatoire. Le jeu contrôle une `current_direction` et une `target_direction` pour les virages, avec un taux de changement progressif (`direction_change_rate`). La route alterne entre des segments droits et des virages dont l'intensité et la longueur varient, tout en veillant à ce qu'elle reste dans les limites de l'écran.
- **Système de Capteurs pour l'IA (`get_sensor_readings`)**: Pour que l'IA puisse "voir" l'environnement sans avoir un accès direct aux données internes du jeu, un système de capteurs a été implémenté.
  - L'IA dispose de **cinq capteurs** simulant des "radars" ou des "yeux".
  - Ces capteurs sont orientés à différents angles par rapport à l'avant de la voiture : -60° (gauche extrême), -30° (gauche), 0° (centre), 30° (droite), 60° (droite extrême).
  - Chaque capteur envoie un "rayon" et mesure la distance jusqu'à ce qu'il rencontre le bord de l'écran ou sorte de la route (en touchant l'herbe).
  - Les distances mesurées sont ensuite **normalisées entre 0 et 1** (0 = très proche, 1 = distance maximale du capteur). Ce sont ces valeurs normalisées que l'IA perçoit comme son "environnement".
  - La visualisation des capteurs peut être activée pour voir ce que l'IA "voit" (lignes jaunes partant de la voiture).
- **Représentation de l'État pour l'IA (`get_simple_state`)**: L'IA ne reçoit pas les distances brutes des capteurs. À la place, chaque distance est **discrétisée en trois zones**:
  - 0: Obstacle proche (< 30% de la portée maximale)
  - 1: Obstacle moyen (entre 30% et 70%)
  - 2: Obstacle loin (> 70%) Ces cinq valeurs discrètes (une pour chaque capteur, allant de 0 à 2) sont ensuite combinées en un seul **état numérique unique**. Étant donné que chaque capteur a 3 valeurs possibles et qu'il y a 5 capteurs, il y a `3^5 = 243` états possibles pour l'IA. Cet état est l'entrée que l'IA utilise pour prendre ses décisions.
- **Actions Possibles (`step`)**: L'IA peut choisir parmi **trois actions**:
  - 0: Tourner à gauche
  - 1: Aller tout droit (pas de mouvement latéral)
  - 2: Tourner à droite
- **Système de Récompense (`reward`)**: Le système de récompense est crucial pour guider l'apprentissage de l'IA.
  - L'IA reçoit une petite récompense positive (0.1) à chaque pas de temps où elle reste sur la route.

- Un bonus plus important (5) est accordé si l'IA reste sur la route pendant une longue période (tous les 100 pas).
- Une **pénalité très importante (-50)** est appliquée si l'IA sort de la route, ce qui met fin à l'épisode (le jeu est "terminé" pour cet essai).
- Un épisode peut également se terminer si trop de temps passe sans progrès (plus de 10 000 pas), avec un bonus lié au score accumulé.

4.3. L'Agent d'Apprentissage par Renforcement (Q-Learning)

L'IA qui apprend à conduire utilise un algorithme d'apprentissage par renforcement appelé **Q-Learning**.

- `QLearningAgent`: Cette classe implémente l'agent qui apprend.
  - **Q-Table**: Le cœur du Q-Learning est la **Q-table** (`self.q_table`). C'est un tableau où l'IA stocke ses "connaissances" sur l'efficacité de chaque action dans chaque état. Au début, la table est initialisée à zéro, ce qui signifie que l'IA "ne sait RIEN".
  - **Paramètres d'apprentissage**:
    - `learning_rate` (0.3): Détermine l'ampleur de la mise à jour de la Q-table à chaque apprentissage. Un taux élevé permet d'apprendre plus vite, mais peut rendre l'apprentissage instable.
    - `discount_factor` (0.9): Détermine l'importance des récompenses futures par rapport aux récompenses immédiates. Une valeur proche de 1 signifie que l'IA prend en compte les conséquences à long terme de ses actions.
    - `epsilon` (1.0 au début, `epsilon_min=0.05`): C'est le paramètre d'**exploration vs. exploitation**.
      - Au début, `epsilon` est à 1.0, ce qui signifie que l'IA choisit des actions **complètement aléatoires** (`np.random.random() <= self.epsilon`) pour explorer l'environnement.
      - Progressivement, `epsilon` diminue (`epsilon_decay=0.999`), et l'IA commence à "exploiter" ses connaissances en choisissant l'action qui a la meilleure Q-value dans l'état actuel (`np.argmax(self.q_table[state])`). Un `epsilon_min` garantit qu'il y a toujours un minimum d'exploration pour éviter de rester bloqué dans des sous-optimaux.
- **Méthode `get_action(state)`**: Cette méthode est responsable du choix de l'action, soit par exploration aléatoire, soit par exploitation de la Q-table.
- **Méthode `learn(state, action, reward, next_state, done)`**: C'est le cœur de l'apprentissage. Elle met à jour la Q-table en utilisant la **formule de Bellman**, qui estime la "valeur future" d'une action. La mise à jour est basée sur la différence entre la valeur Q actuelle et une "cible" qui inclut la récompense immédiate et la meilleure récompense future possible du prochain état.
- **Statistiques d'entraînement**: L'agent suit son propre score, le nombre d'épisodes joués, le nombre de crashs, et son meilleur score, permettant de

visualiser les progrès.

- **Processus d'entraînement (`train_ai_slowly`)**:
  - L'entraînement se déroule sur un nombre défini d'épisodes (par exemple, 2000).
  - À chaque épisode, la voiture est réinitialisée. L'IA interagit avec le jeu en choisissant des actions, observant les nouveaux états et récompenses, puis mettant à jour sa Q-table.
  - La visualisation (`show_visual`) permet de voir l'IA apprendre en temps réel. Au début, ses actions sont très aléatoires, mais progressivement, elle commence à naviguer la route avec plus de succès.
  - Des raccourcis clavier sont inclus pour accélérer la simulation (`K_SPACE`) ou masquer/afficher la visualisation (`K_s`).
  - Les progrès sont affichés en temps réel, montrant le score moyen, le taux de crash et le niveau d'exploration.

4.4. Le Concept du Jeu Inversé : Le Joueur Contrôle la Route

En plus du jeu où l'IA conduit, une version alternative a été conceptualisée et partiellement codée dans `road_curve_game_concept3.py`. Dans ce "**Jeu de Voiture Inversé**", **le joueur ne contrôle pas la voiture, mais la route elle-même !**. L'objectif du joueur est de manipuler la route (direction et largeur) pour aider des voitures contrôlées par l'IA à rester sur le chemin.

- **Voitures IA (AICar)**: Dans ce mode, chaque voiture IA est elle-même un agent Q-Learning indépendant avec ses propres capteurs et sa capacité à prendre des décisions. Pour éviter un comportement complètement aléatoire au début, ces agents sont dotés d'un **pré-entraînement basique** (`pre_train`) qui leur donne quelques règles de base (ex: si obstacle à gauche, favoriser la droite).
- **Interaction**: Ce concept est intéressant car il inverse la dynamique habituelle des jeux de conduite et permet d'explorer l'apprentissage automatique sous un angle différent. Le joueur guide indirectement l'IA, ce qui peut être utilisé pour générer des données d'entraînement ou pour évaluer la robustesse des agents IA face à une route dynamique contrôlée par un humain.

4.5. Les Concepts Fondamentaux du Q-Learning

Ce projet a été l'occasion d'approfondir ma compréhension des concepts d'apprentissage par renforcement et d'optimisation. * **Problème d'optimisation**: L'objectif de l'IA (rester sur la route) est un exemple de problème d'optimisation, où il faut trouver la solution la plus efficace parmi plusieurs options (aller à gauche, droite ou tout droit). * **Algorithmes**: J'ai pu saisir les bases de certains algorithmes, comme la **dichotomie** (recherche par division), et même des concepts plus ésotériques comme le "biais de choix". * **Le couple (état, action)**: Un concept central du Q-learning est le couple (état, action). L'état représente la situation actuelle de l'agent (par exemple, la

position de la voiture par rapport à la route telle que perçue par les capteurs). L'action est la décision que l'agent prend dans cet état (par exemple, aller à droite ou à gauche). La fonction Q-learning estime la qualité d'une action donnée dans un état donné (`Q(s, a)`).

5. Les Outils et Compétences Développées

Ce stage m'a permis d'acquérir ou de renforcer de nombreuses compétences techniques et méthodologiques.

5.1. Programmation Python et Bibliothèques Spécifiques

- **Python**: Une pratique intensive du langage Python, incluant la manipulation de variables, de fonctions, de classes et de boucles.
- **Pygame**: Découverte de Pygame pour la création d'interfaces graphiques simples et de jeux, incluant la gestion des événements, le dessin d'objets, et l'animation.
- **PyTorch**: Première utilisation de PyTorch, une bibliothèque majeure pour l'apprentissage automatique, notamment pour la définition de réseaux de neurones, la gestion des tenseurs, l'entraînement et l'évaluation de modèles. J'ai également appris l'importance de choisir la bonne version de Python et les bibliothèques compatibles.
- **`os` et `json`**: Gestion des fichiers et des données JSON pour la sauvegarde et le chargement des modèles et historiques.

5.2. Interaction et Ingénierie des Prompts avec les IA Génératives

- J'ai appris à formuler des **prompts clairs et précis** pour obtenir le code et les explications souhaitées de la part de ChatGPT et Claude.
- J'ai expérimenté le processus de **dialogue itératif** avec l'IA, en demandant des modifications et des améliorations basées sur les résultats obtenus (ex: passer de TensorFlow à PyTorch, ajouter une interface, améliorer les performances, faire défiler la route, ajouter des virages).
- La capacité de Claude à **comprendre et corriger des erreurs visuelles à partir de captures d'écran** a été une révélation et un gain de temps considérable, démontrant le potentiel de ces outils.

5.3. Debugging et Résolution de Problèmes

- Le stage m'a confronté à des problèmes techniques (incompatibilité de bibliothèques, erreurs de rendu graphique). Cela m'a appris l'importance de l'identification des problèmes, de la recherche de solutions et de la validation des corrections.
- La collaboration avec l'IA pour le debugging, comme le problème de rendu de la route, a été une approche innovante de la résolution de problèmes.

5.4. Compréhension des Concepts d'Intelligence Artificielle

- **Concepts fondamentaux**: J'ai pu assimiler des notions clés comme les réseaux de neurones, les poids et les biais.
- **Apprentissage par renforcement (Q-Learning)**: J'ai acquis une compréhension pratique du Q-learning, incluant les états, les actions, les récompenses, la Q-table, et les paramètres d'exploration/exploitation.
- **Cycle de vie d'un projet IA**: De la définition de l'objectif à l'entraînement, l'évaluation et l'amélioration continue du modèle.
- **Importance des données**: J'ai compris le rôle crucial des données d'entraînement (MNIST) et de leur prétraitement.
- **Types d'IA**: J'ai eu un aperçu des différentes applications de l'IA, de la reconnaissance de formes aux diagnostics médicaux (projet de Thomas).

5.5. Maîtrise des Commandes PowerShell

En plus du développement, j'ai appris des commandes essentielles pour naviguer et gérer les fichiers sur mon ordinateur, ce qui est fondamental en développement: * `ls`: Lister le contenu d'un dossier. * `cd + ***`: Changer de répertoire. * `cd+..`: Remonter d'un niveau. * `cd+~`: Retourner au répertoire de départ. * `history`: Afficher l'historique des commandes. * `pwd`: Afficher le chemin d'accès actuel. * `py + nom doc.py`: Lancer un script Python. * `py -m pip install nom_paquet`: Installer des bibliothèques Python. * `.\.venv\Scripts\Activate.ps1`: Activer un environnement virtuel Python. * `Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass`: Contourner temporairement les restrictions d'exécution de scripts. * `Mv path path`: Déplacer un fichier.

Ces compétences m'ont permis de gérer mon environnement de travail de manière autonome.

6. Conclusion et Perspectives

Mon stage de seconde chez Orange a été une expérience incroyablement enrichissante et formatrice. J'ai eu l'opportunité unique de plonger au cœur de l'intelligence artificielle et de la programmation Python, en abordant des projets concrets de reconnaissance de chiffres et de jeu de conduite. L'utilisation des IA génératives comme assistants de codage a été particulièrement instructive, me montrant comment ces outils peuvent accélérer l'apprentissage et le développement.

J'ai appris que le développement d'une IA est un processus itératif, qui implique non seulement l'écriture de code, mais aussi l'analyse des résultats, l'identification des problèmes et la recherche de solutions créatives. La persévérance est essentielle, surtout lorsqu'on travaille sur des concepts complexes comme le Q-learning, où l'IA commence littéralement de "zéro" et apprend progressivement.

Ce stage a confirmé mon intérêt pour l'informatique et l'intelligence artificielle.

Les connaissances acquises en Python, PyTorch, Pygame et en apprentissage par renforcement constituent une base solide pour de futurs projets. Je suis particulièrement enthousiasmé par la possibilité de continuer à entraîner l'IA de conduite pour qu'elle devienne de plus en plus performante sur des routes complexes et imprévisibles.

Je tiens à remercier Orange et mon tuteur de stage pour m'avoir offert cette opportunité exceptionnelle et pour leur soutien tout au long de cette expérience.
