



Utilisation d'un réseau neuronal convolutif pour la reconnaissance de Pokémon

EL KHALFIOUI Nadir

EL KHAMLICH Badreddine

LEHLALI Hèdi

MAM 4A – Polytech Lyon
2022 - 2023

Table des matières

| | |
|---|----|
| I - Présentation de la licence | 5 |
| II - Objectif | 6 |
| 1 - Idée du Projet..... | 6 |
| 2 - Réseaux de neurones convolutifs (CNN) | 7 |
| A - Applications | 7 |
| B - Convolution | 7 |
| C - Fonction d'activation | 9 |
| D - Pooling | 9 |
| E - Classification | 10 |
| 3 - Mise en pratique..... | 10 |
| III - Méthodologie..... | 11 |
| 1 - Description du Dataset..... | 11 |
| A - Choix des Pokémon..... | 11 |
| B - Entraînement, validation et test | 12 |
| 2 - Librairies utilisées..... | 12 |
| 3 - Traitement des données | 13 |
| 4 - Entraînement du modèle | 14 |
| A - Construction de l'architecture du CNN : | 14 |
| B - Techniques de régularisation : | 15 |
| C - Augmentation des données : | 15 |
| D - Évaluation du modèle pendant l'entraînement : | 16 |
| E - Évaluation finale sur l'ensemble de test : | 16 |
| IV - Résultats | 17 |
| 1 - Précision du modèle..... | 17 |
| 2 - Overfitting | 18 |
| 3 - Matrice de confusion..... | 19 |
| 4 - Test du modèle | 20 |
| V - Discussion des résultats..... | 21 |
| 1 - Points forts du modèle..... | 21 |
| 2 - Points faibles du modèle | 21 |
| 3 - Pistes d'amélioration..... | 22 |
| VI - Conclusion | 23 |
| Références..... | 24 |

Table des figures

| | |
|---|----|
| Figure 1 - Fiche Pokédex de Pikachu | 5 |
| Figure 2 - Personnages principaux des jeux Pokémon | 5 |
| Figure 3 - Pokédex..... | 6 |
| Figure 4 - Pokédex rempli avec des informations sur des Pokémon capturés | 6 |
| Figure 5 - Réseau neuronal convolutif | 7 |
| Figure 6 - Convolution d'une image et d'un kernel..... | 8 |
| Figure 7 - Effet d'une matrice de flou..... | 8 |
| Figure 8 - Effet d'une matrice de netteté | 8 |
| Figure 9 - Utilisation du produit de convolution dans le traitement d'image | 9 |
| Figure 10 - Bulbizarre, Salamèche et Carapuce..... | 10 |
| Figure 11 - Pokémon sélectionnés | 11 |
| Figure 12 – TensorFlow | 12 |
| Figure 13 – NumPy | 12 |
| Figure 14 - Précision du modèle | 17 |
| Figure 15 - Perte du modèle..... | 18 |
| Figure 16 - Matrice de confusion | 19 |
| Figure 17 - Test du modèle sur une image de Bulbizarre..... | 20 |
| Figure 18 - Sacha Ketchum devenu meilleur dresseur du monde | 23 |

I - Présentation de la licence

La licence Pokémon est l'une des franchises de divertissement les plus populaires et les plus appréciées dans le monde entier. Elle a été créée en 1996 par Satoshi Tajiri et Ken Sugimori, et depuis lors, elle a conquis le cœur de millions de fans de tous âges à travers ses jeux vidéo, sa série animée, ses films ou encore ses jeux de cartes à collectionner.

Cet univers est rempli de créatures fantastiques appelées Pokémon (contraction de "Pocket" et "Monster", des monstres de poche). Ces créatures variées possèdent des pouvoirs et des capacités spéciales et elles coexistent avec les humains dans un environnement harmonieux. Les Pokémon ont différentes caractéristiques comme la taille, le poids, le sexe ou encore le type.


| Généralités | | | | |
|--|---|---------|---------------------|------------|
|  | Nom français | Pikachu | Numéro National | #025 |
| | Nom anglais | Pikachu | Numéro Sinnoh | #104 |
| | Nom japonais | Pikachu | Type 1 | ELECTRIQUE |
| | Taille | 0.4 m | Type 2 | AUCUN |
| | Poids | 6.0 kg | Taux de capture [%] | 190 |
| | Expérience max | 1000000 | Genre | M : 50.0% |
| | Espèce | Souris | | F : 50% |
| | Les jours d'orages, on peut apercevoir des Pikachu regroupés en haut des arbres ou des poteaux. Ces Pokémon attendent patiemment que la foudre leur tombe dessus, pour pouvoir recharger au maximum leur réserve d'électricité. Quand il attaque, des milliers de volts sortent par ses joues rouges. | | | |

Figure 1 - Fiche Pokédex de Pikachu

Les joueurs endossent le rôle d'un Dresseur Pokémon, dont le but principal est de capturer, d'entraîner et de combattre les Pokémon pour devenir le meilleur dresseur d'entre tous, le Maître Pokémon. Lorsque l'on capture un Pokémon, toutes les informations relatives à celui-ci sont enregistrées dans le Pokédex, un ordinateur de poche qui enregistre toutes les informations des Pokémon.



Figure 2 - Personnages principaux des jeux Pokémon

II - Objectif

1 - Idée du Projet

Dans les jeux Pokémon, le personnage principal se voit offrir un outil appelé Pokédex. Il s'agit d'un outil qui permet d'enregistrer et collecter des informations sur les Pokémon. Le joueur doit capturer un Pokémon pour obtenir ses informations.



Figure 3 - Pokédex

Un des objectifs des jeux est de remplir le Pokédex en capturant tous les Pokémon, soit 151 pour les premiers jeux et plusieurs centaines pour les plus récents. Ces informations sont ensuite au Professeur Chen, chercheur spécialiste en Pokémon et créateur du Pokédex.

| LISTE POKÉMON | | | |
|------------------|------------|--------|--------|
| N°001 | BULBIZARRE | PLANTE | POISON |
| N°002 | HERBIZARRE | PLANTE | POISON |
| N°003 | FLORIZARRE | PLANTE | POISON |
| N°004 | SALAMECHE | FEU | |
| N°005 | REPTINCEL | FEU | |
| N°006 | DRACAUFEU | FEU | VOL |
| N°007 | CARAPUCE | EAU | |
| N°008 | CARABAFFE | EAU | |
| N°009 | TORTANK | EAU | |
| ←CHOIX OK ANNUL. | | | |

Figure 4 - Pokédex rempli avec des informations sur des Pokémon capturés

Notre objectif est de créer un modèle d'intelligence artificielle reprenant ce principe. En utilisant l'image d'un Pokémon, on pourrait déterminer son ou ses types. On va utiliser pour cela un modèle de réseau neuronal convolutif pour reconnaître les images de Pokémon et déduire le type le plus probable du Pokémon.

2 - Réseaux de neurones convolutifs (CNN)

A - Applications

Les réseaux neuronaux convolutifs (CNN) sont utilisés dans le domaine du Computer Vision pour des tâches telles que la classification d'images, la détection d'objets et la reconnaissance faciale. Leur utilisation repose sur l'exploitation de la structure et des motifs présents dans les données visuelles.

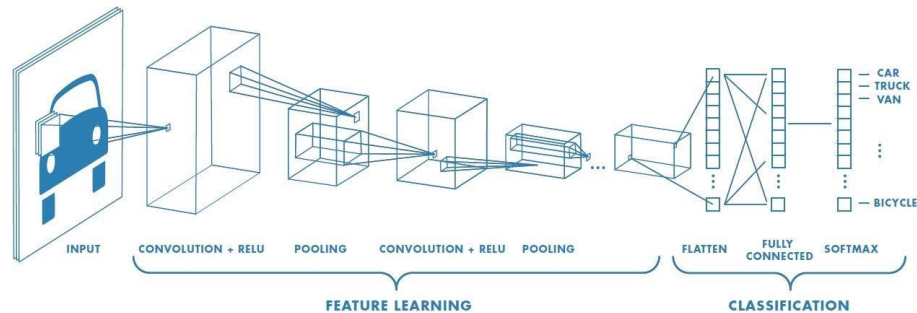


Figure 5 - Réseau neuronal convolutif

Les CNN sont utilisés pour la détection d'objets. Cela est particulièrement utile dans des domaines tels que la sécurité, la surveillance et la réalité augmentée où la détection précise des objets est essentielle.

Les CNN sont également utilisés pour la reconnaissance faciale. En entraînant un modèle sur un grand nombre d'images de visages, les CNN peuvent extraire des caractéristiques uniques des visages et les utiliser pour identifier et reconnaître des personnes. Cette technologie est largement utilisée dans les systèmes de sécurité, les applications de détection de fraude et les outils de gestion d'identité.

B - Convolution

En entraînant un réseau neuronal convolutif sur un ensemble d'images étiquetées, le modèle est capable de reconnaître et de classer automatiquement de nouvelles images dans des catégories prédéfinies. Les couches de convolution permettent de capturer les caractéristiques visuelles importantes, comme les bords, les textures, les couleurs et les formes, tandis que les couches de classification finales attribuent une probabilité d'appartenance de l'image à chaque catégorie.

La convolution est une opération fondamentale dans les réseaux de neurones convolutifs (CNN) utilisés pour le traitement d'images. Elle consiste à appliquer un kernel (aussi appelé filtre ou matrice de convolution) sur une région locale de l'image d'entrée pour calculer une nouvelle valeur correspondant à une caractéristique spécifique.

La convolution est réalisée en glissant le filtre sur l'image, généralement de gauche à droite et de haut en bas, en effectuant une opération mathématique appelée produit scalaire entre les valeurs du filtre et les valeurs de l'image dans la région couverte. Le résultat de cette opération est attribué à une nouvelle position dans la carte de caractéristiques.

La phase de convolution permet de détecter et mettre en avant les caractéristiques importantes de l'image.

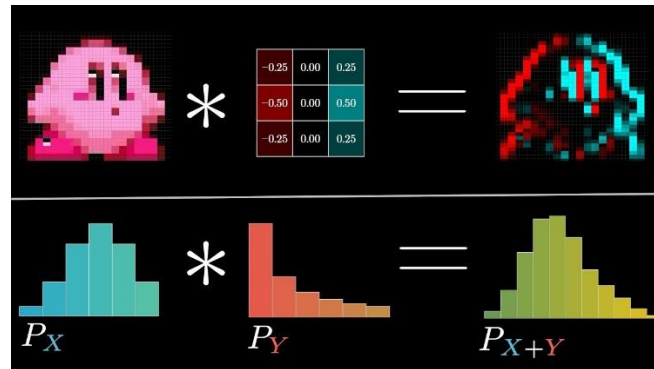


Figure 9 - Utilisation du produit de convolution dans le traitement d'image

C - Fonction d'activation

Après l'opération de convolution, les sorties obtenues sont généralement soumises à une fonction d'activation pour introduire une non-linéarité dans le réseau. Une fonction d'activation couramment utilisée dans les CNN est la fonction ReLU (Rectified Linear Unit). Une activation est la sortie d'un neurone ou d'une couche de neurones après avoir appliqué une fonction d'activation à la somme pondérée des entrées.

La fonction d'activation détermine si le neurone doit être activé ou non. Elle introduit de la non-linéarité dans le réseau, permettant ainsi au modèle d'apprendre et de représenter des relations complexes entre les entrées et les sorties.

La fonction ReLU attribue la valeur zéro à toutes les activations négatives, tandis que les activations positives sont laissées intactes. Cela signifie que si une activation est négative, elle est supprimée, tandis que si elle est positive, elle est conservée sans modification. Il s'agit d'une forme de valeur absolue.

La fonction ReLU est la plus utilisée en raison de sa simplicité et de son efficacité. Elle aide à introduire de la non-linéarité dans le réseau, ce qui lui permet de capturer des relations complexes entre les caractéristiques extraites.

D - Pooling

Les couches de pooling sont utilisées dans les réseaux de neurones pour réduire la dimension spatiale des données tout en préservant les caractéristiques importantes.

L'une des techniques les plus utilisées est le Max Pooling, qui consiste à extraire la valeur maximale dans chaque zone d'une grille. Pendant l'opération de pooling, une grille est superposée sur les données de sortie de la couche précédente (généralement la sortie de la couche de convolution). Chaque région de la grille, appelée fenêtre de pooling, regroupe un ensemble de valeurs. Dans le cas du Max Pooling, la valeur maximale de chaque fenêtre est extraite et utilisée comme représentation réduite de cette région.

Le Max Pooling présente plusieurs avantages. Il permet de réduire la dimension des données en conservant uniquement les valeurs maximales. Cela conduit à une réduction de la complexité du modèle et à des calculs plus efficaces.

Le Max Pooling conserve aussi les caractéristiques les plus saillantes de chaque région, ce qui permet de préserver les informations importantes tout en réduisant les détails inutiles.

E - Classification

Après l'extraction des caractéristiques par les couches de convolution et de pooling, la dernière étape d'un réseau de neurones convolutif (CNN) est la classification. Dans cette étape, les caractéristiques extraites sont utilisées pour prédire les classes ou les étiquettes correspondantes aux données d'entrée. Les couches de classification, qui sont généralement des couches entièrement connectées, prennent les caractéristiques extraites en entrée et les transforment en probabilités pour chaque classe possible.

Chaque neurone dans la couche de classification est connecté à tous les neurones de la couche précédente, de manière à apprendre les relations complexes entre les caractéristiques et les classes. La fonction d'activation finale utilisée dans la couche de classification est généralement la fonction softmax.

Cette fonction attribue des probabilités à chaque classe, indiquant la confiance du modèle dans sa prédiction. La somme des probabilités pour toutes les classes est égale à 1, ce qui permet d'interpréter les sorties du modèle comme des distributions de probabilités.

En utilisant la fonction softmax, le CNN est en mesure de prédire la classe la plus probable pour les données d'entrée, en se basant sur les caractéristiques extraites pendant les étapes précédentes. Cette étape de classification permet donc de prendre une décision finale sur la classe ou l'étiquette correspondant aux données analysées.

3 – Mise en pratique

Ces étapes de convolution, de mise en commun et de classification sont généralement répétées plusieurs fois dans un CNN, formant ainsi une architecture en couches. Les poids des neurones sont ajustés lors de l'apprentissage du modèle en utilisant des algorithmes d'optimisation tels que la rétropropagation de l'erreur, permettant au réseau de s'ajuster aux données d'entraînement et d'améliorer ses performances de prédiction.

Ensuite, le modèle est compilé avec l'optimiseur Adam, qui ajuste les poids du modèle pour minimiser la perte lors de l'entraînement. La perte utilisée est la perte de catégorie croisée creuse, adaptée à la classification multi-classes. La métrique d'exactitude est également spécifiée pour évaluer les performances du modèle.

On récupèrera des images de ces Pokémon afin de créer un ensemble d'images qui serviront à entraîner notre modèle.



Figure 10 - Bulbizarre, Salamèche et Carapuce

III - Méthodologie

1 - Description du Dataset

A – Choix des Pokémon

On dispose d'une banque d'images de Pokémon trouvé sur Kaggle (ainsi que d'autres images récupérées sur Google Images grâce à une extension Chrome). Chaque Pokémon a un dossier contenant des images de lui qui serviront à entraîner le modèle. On possède également un fichier csv contenant pour chaque Pokémon de Kanto :

- **Nom**
- **Type**
- **Type secondaire**
- **Total**
- **HP** (Health Points – Points de vie)
- **ATK** (Capacité d'attaque)
- **ATK SPE** (Capacité d'attaque à distance)
- **DEF** (Défense)
- **DEF SPE** (Défense sur les attaques à distance)
- **Speed** (Vitesse)
- **Génération** (1 pour les Pokémon de la 1^{ère} génération ici)
- **Legendary** (Si le Pokémon est un Pokémon légendaire ou non)

On décide d'entraîner le modèle à reconnaître un nombre restreint de Pokémon pour des raisons pratiques (stockage et temps de calcul). Les Pokémon choisis sont :

- **Bulbizarre**
- **Salamèche**
- **Carapuce**
- **Pikachu**
- **Rondoudou**
- **Ronflex**
- **Evoli**
- **Lokhlass**
- **Onix**
- **Mewtwo**

Nous avons choisi ces Pokémon car ils sont les plus populaires et possèdent chacun des caractéristiques physiques bien distinctives. Voici leurs designs respectifs :



Figure 11 - Pokémon sélectionnés

B – Entraînement, validation et test

On part d'un total de plus de 2000 images pour les 10 Pokémon. On répartit ensuite les images comme suit :

- **70%** pour entraîner le modèle – environ 1400
- **20%** pour la validation – environ 400
- **10%** pour tester le modèle – environ 200

Cette division nous a permis d'entraîner le modèle, de l'optimiser et d'évaluer ses performances de manière rigoureuse.

2 - Librairies utilisées

Pour ce projet, plusieurs bibliothèques populaires ont été utilisées pour faciliter le développement et l'analyse des données. Les principales librairies utilisées sont TensorFlow, NumPy, OpenCV (cv2), Matplotlib et Pandas. TensorFlow est une bibliothèque d'apprentissage automatique très populaire, largement utilisée pour construire et entraîner des réseaux de neurones, y compris les réseaux de neurones convolutifs.

- **TensorFlow** fournit une interface conviviale pour créer des modèles d'apprentissage en profondeur, ainsi que des outils avancés pour l'optimisation, la visualisation et le déploiement des modèles.



Figure 12 – TensorFlow

- **NumPy** est une librairie fondamentale du calcul scientifique.

Elle fournit des structures de données avancées pour la manipulation de tableaux multidimensionnels et offre une large gamme de fonctions mathématiques. NumPy est largement utilisé pour la préparation des données, le traitement des images et la manipulation des tenseurs, ce qui en fait une librairie essentielle dans les projets d'apprentissage en profondeur.



Figure 13 – NumPy

- **OpenCV (cv2)** est une librairie spécialisée dans le traitement d'images et la vision par ordinateur.

Elle offre un large éventail de fonctions et d'algorithmes pour lire, manipuler et analyser des images. OpenCV est couramment utilisé dans les projets d'apprentissage en profondeur pour le prétraitement des images, la gestion des jeux de données et l'analyse des résultats.

- **Matplotlib** est une librairie de visualisation.

Elle permet de créer des graphiques et des visualisations de données, ce qui est essentiel pour l'analyse et la présentation des résultats dans un projet d'apprentissage en profondeur.

- **Pandas** est une bibliothèque de manipulation et d'analyse de données.

Elle offre des structures de données efficaces pour le traitement et l'analyse des données tabulaires. Pandas est souvent utilisé pour la préparation des données, le chargement des jeux de données, la gestion des étiquettes et des métadonnées, ainsi que pour la génération de statistiques et la manipulation des tableaux de données.

3 - Traitement des données

Pour garantir la cohérence des normes et éviter tout problème de format, nous avons veillé à ce que toutes les images soient au format JPG ou PNG. Pour celles qui n'étaient pas dans le bon format, nous avons créé des convertisseurs d'images en utilisant des scripts Python. Ces convertisseurs nous ont permis de les renommer correctement et de les convertir au format PNG, afin de standardiser leur format et faciliter l'apprentissage de notre modèle. De cette manière, nous nous sommes assurés que toutes les images utilisées étaient compatibles avec notre processus de traitement des données.

On commence donc par charger les images à partir d'un dossier spécifique appelé "dataset". On spécifie le chemin du dossier et on compte le nombre total d'images pour évaluer la taille de notre dataset.

Ensuite, on organise nos données en les répartissant dans des dossiers distincts pour les ensembles d'entraînement, de validation et de test. On crée les dossiers "train", "test" et "val", ainsi que les sous-dossiers correspondant à chaque classe de Pokémon. Cette façon de structurer nos données nous permet de mieux gérer leur utilisation future. Ensuite, on répartit les images de manière aléatoire dans ces dossiers, en respectant les proportions spécifiées (généralement 70% pour l'entraînement, 20% pour les tests et 10% pour la validation). Cette répartition aléatoire nous assure que les images représentent équitablement chaque classe de Pokémon dans chaque ensemble.

Pour enrichir notre dataset, on génère des données d'entraînement et de validation en utilisant des générateurs de données. On utilise la classe ImageDataGenerator de

Keras pour cela. Ces générateurs nous permettent d'appliquer des transformations et des augmentations sur nos images, comme la rotation, le zoom, le décalage horizontal/vertical et le retournement horizontal. Ces transformations ajoutent de la diversité à nos données, ce qui renforce la robustesse de notre modèle. De plus, on redimensionne nos données pour qu'elles aient une taille spécifique de 200x200 pixels, afin de s'adapter à l'architecture de notre modèle.

Enfin, on normalise les valeurs des pixels de nos images. Cette étape consiste à ramener les valeurs des pixels dans une plage allant de 0 à 1. Cette normalisation facilite la convergence de l'entraînement de notre modèle en réduisant les écarts entre les valeurs des pixels. Pour cela, on divise chaque valeur de pixel par la valeur maximale possible (généralement 255 pour une image RVB). Ainsi, toutes les valeurs des pixels sont ramenées dans l'intervalle $[0, 1]$, ce qui permet à notre modèle d'apprendre de manière plus efficace.

Cette méthodologie classique de traitement des données dans le domaine de la classification d'images nous permet de préparer nos données pour l'entraînement d'un modèle CNN. Elle facilite l'apprentissage et l'évaluation de notre modèle en nous assurant que nos données sont organisées de manière équilibrée et en enrichissant notre dataset avec des transformations et des augmentations.

4 – Entraînement du modèle

A - Construction de l'architecture du CNN :

Dans notre processus d'entraînement, nous avons utilisé TensorFlow pour construire un réseau de neurones convolutif (CNN) spécialement adapté à la vision par ordinateur. Les couches de convolution jouent un rôle essentiel dans l'extraction des caractéristiques des images en appliquant des filtres qui détectent les contours, les textures et les formes. En ajustant la taille des filtres dans ces couches, nous avons pu capturer les informations pertinentes pour notre tâche spécifique de classification des Pokémon. Ensuite, les couches de pooling ont été utilisées pour réduire la dimension des caractéristiques extraites, tout en préservant les informations essentielles. En définissant le taux de pooling, qui détermine la taille de la fenêtre de regroupement, nous avons pu contrôler la réduction de dimension pour une meilleure efficacité de traitement.

De plus, les couches de classification, également connues sous le nom de couches entièrement connectées, ont été utilisées pour la classification finale des images. Ces couches ont transformé les caractéristiques extraites par les couches précédentes en une probabilité pour chaque classe de Pokémon. En ajustant le nombre de neurones dans ces couches, nous avons pu influencer la complexité du modèle et sa capacité à apprendre des relations plus sophistiquées entre les caractéristiques. En somme, en utilisant TensorFlow, nous avons construit un CNN en définissant les couches de

convolution pour l'extraction des caractéristiques, les couches de pooling pour la réduction de dimension et les couches de classification pour la classification finale. Ces différentes couches travaillent en tandem pour permettre au modèle d'extraire les caractéristiques discriminantes des images de Pokémon et de les utiliser pour une classification précise.

B - Techniques de régularisation :

Dans notre processus d'entraînement, nous avons mis en œuvre la régularisation pour éviter le surapprentissage et améliorer la capacité de généralisation de notre modèle. La régularisation consiste à ajouter des termes de pénalité à la fonction de perte du modèle, ce qui décourage les poids excessivement grands. En limitant la complexité du modèle, nous réduisons sa sensibilité aux variations mineures présentes dans les données d'entraînement. Grâce à l'utilisation de la régularisation, notre modèle a pu se concentrer sur les caractéristiques les plus importantes et généraliser plus efficacement, même lorsqu'il était confronté à des variations dans les données.

En intégrant la régularisation dans notre processus d'entraînement, nous avons réussi à prévenir le surapprentissage, où le modèle s'adapte excessivement aux données d'entraînement spécifiques et ne parvient pas à généraliser correctement. En contrôlant la complexité du modèle à l'aide de termes de pénalité, nous avons favorisé une meilleure généralisation et renforcé la capacité du modèle à produire des prédictions précises sur de nouvelles données. La régularisation s'est avérée être une technique efficace pour améliorer la robustesse et la performance de notre modèle d'apprentissage automatique.

C - Augmentation des données :

Nous avons utilisé l'augmentation des données comme une technique essentielle pour améliorer la capacité de généralisation de notre modèle. En appliquant des transformations aléatoires telles que la rotation, le zoom ou le décalage aux images d'entraînement, nous avons créé de nouvelles variations réalistes. Cette approche a enrichi notre jeu de données d'entraînement en introduisant des variations que le modèle pourrait rencontrer dans des situations réelles.

L'objectif de l'augmentation des données était de renforcer la capacité du modèle à reconnaître et à généraliser des motifs importants, même en présence de variations telles que des rotations ou des zooms. En exposant le modèle à ces variations contrôlées, nous l'avons rendu plus robuste et capable de généraliser efficacement sur de nouvelles données. L'augmentation des données a joué un rôle crucial en permettant au modèle de saisir les caractéristiques discriminantes des images et de les utiliser pour une classification précise des différentes classes de Pokémon.

D - Évaluation du modèle pendant l'entraînement :

Tout au long de l'entraînement, nous avons effectué une évaluation régulière des performances du modèle en utilisant un ensemble de validation dédié. Cette approche nous a permis de surveiller attentivement des métriques clés telles que l'exactitude et la perte du modèle. En examinant ces métriques, nous étions en mesure de détecter tout signe de surapprentissage ou de performances insatisfaisantes.

Si nous avons observé une stagnation des performances ou une dégradation de l'exactitude sur l'ensemble de validation, nous avons pris des mesures correctives. Cela pouvait inclure l'ajustement des hyperparamètres, tels que la taille des filtres ou le nombre de neurones, ou même une révision de l'architecture du modèle. L'objectif était d'améliorer les performances du modèle en tenant compte des observations et des retours fournis par l'ensemble de validation.

En somme, l'évaluation régulière sur l'ensemble de validation a joué un rôle crucial dans l'amélioration des performances du modèle. Cela nous a permis d'identifier les problèmes potentiels et de prendre des mesures pour les résoudre, garantissant ainsi que le modèle soit capable de faire des prédictions précises et généralisables sur de nouvelles données.

E - Évaluation finale sur l'ensemble de test :

Après avoir obtenu des performances satisfaisantes sur l'ensemble de validation, nous avons procédé à une évaluation finale du modèle en utilisant un ensemble de test indépendant. Cet ensemble de test était composé de données entièrement nouvelles qui n'avaient pas été utilisées lors de l'entraînement ni de la validation. L'objectif de cette évaluation était d'obtenir une estimation impartiale de la capacité du modèle à généraliser et à effectuer des prédictions précises sur de nouvelles données.

L'évaluation sur l'ensemble de test nous a permis de mesurer la performance réelle du modèle, en simulant des conditions réelles où il est confronté à des données inconnues. Si le modèle obtenait de bons résultats sur cet ensemble de test, cela indiquait sa capacité à reconnaître et à classer correctement de nouvelles instances de Pokémon. Cette étape d'évaluation finale nous a conféré une confiance supplémentaire dans les capacités du modèle et nous a assuré qu'il était prêt à être utilisé dans des situations pratiques.

En résumé, l'évaluation sur un ensemble de test indépendant nous a permis d'obtenir une estimation objective de la capacité du modèle à généraliser sur de nouvelles données. Cette étape finale d'évaluation nous a fourni une mesure fiable de la performance du modèle et nous a donné l'assurance que le modèle était apte à être utilisé avec précision dans des scénarios réels.

IV - Résultats

1 - Précision du modèle

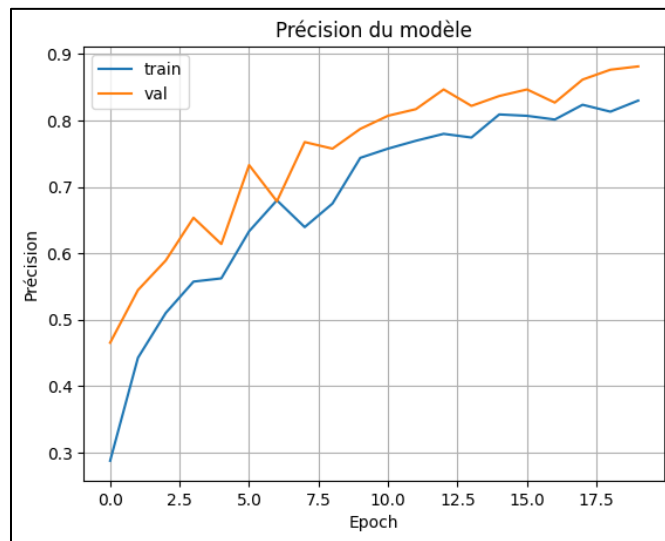


Figure 14 - Précision du modèle

Le graphique représentant la précision du modèle est un indicateur important pour évaluer les performances de l'algorithme d'apprentissage. Dans notre cas, le graphique montre l'évolution de la précision à la fois sur l'ensemble d'entraînement ("train") et sur l'ensemble de validation ("val") au fil des itérations.

Au début de l'entraînement, la précision du modèle sur l'ensemble d'entraînement augmente rapidement. Cela signifie que le modèle apprend progressivement à reconnaître les motifs et les caractéristiques des images de Pokémon présentes dans l'ensemble d'entraînement. Parallèlement, la précision sur l'ensemble de validation augmente également, mais à un rythme légèrement plus lent. Cela suggère que le modèle est capable de généraliser et de reconnaître des motifs similaires dans de nouvelles données.

Cependant, il est important de noter qu'à un certain stade de l'entraînement, la précision sur l'ensemble d'entraînement continue d'augmenter tandis que la précision sur l'ensemble de validation atteint un plateau. Cela peut indiquer un potentiel de surapprentissage, où le modèle commence à mémoriser les données d'entraînement spécifiques au lieu de généraliser les motifs. Lorsqu'un modèle est surapprenant, il peut avoir du mal à faire des prédictions précises sur de nouvelles données qui ne ressemblent pas exactement à celles sur lesquelles il a été entraîné.

Le graphe où on peut apercevoir la précision est issu d'un modèle configuré à 20 Epoch. Ainsi, nous trouvons un bon équilibre entre un nombre correct d'itérations mais également un modèle protégé de l'overfitting.

Dans notre cas, il est encourageant de constater que la précision du modèle sur l'ensemble de validation converge vers une valeur élevée, proche de 0,9. Cela suggère que le modèle a réussi à apprendre des motifs et des caractéristiques générales des

images de Pokémon, lui permettant de faire des prédictions précises sur de nouvelles données. Toutefois, il convient d'être vigilant quant à tout signe de surapprentissage éventuel et de surveiller les performances du modèle sur des ensembles de test indépendants pour confirmer sa capacité à généraliser correctement.

2 - Overfitting

Pour remarquer si ce phénomène est présent ou non dans notre modèle, nous pouvons regarder le graphe des pertes ci-dessous :

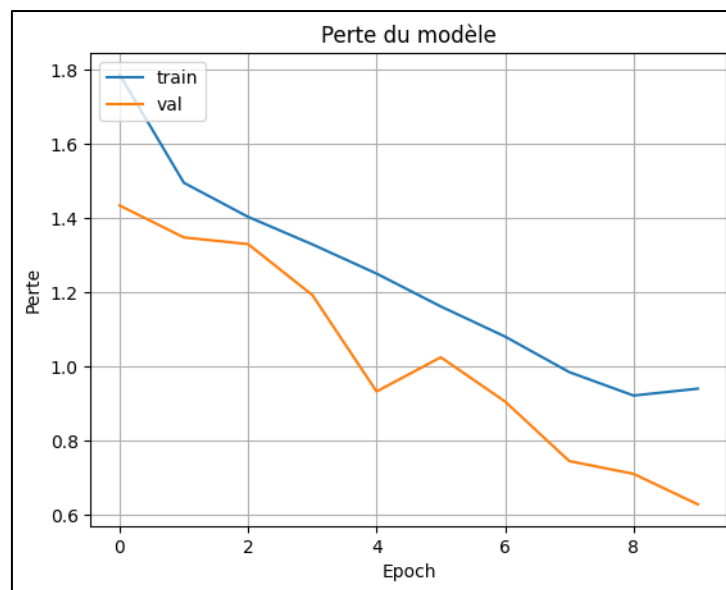


Figure 15 - Perte du modèle

L'overfitting, ou surapprentissage, est un problème courant dans l'apprentissage automatique où un modèle s'adapte trop étroitement aux données d'entraînement spécifiques au détriment de sa capacité à généraliser sur de nouvelles données. Pour évaluer la présence d'overfitting, nous avons également examiné le graphe des pertes.

Le graphe des pertes est un outil utile pour comprendre comment la performance du modèle évolue au cours de l'entraînement. Dans notre cas, nous avons observé que les courbes de perte pour l'ensemble d'entraînement et l'ensemble de validation évoluent de manière similaire. Au fur et à mesure de l'entraînement, les pertes diminuent à la fois pour l'ensemble d'entraînement et l'ensemble de validation, ce qui indique que le modèle est en train d'apprendre à prédire avec précision.

En analysant ce graphe des pertes, nous constatons qu'il n'y a pas d'écart significatif entre les courbes de perte de l'ensemble d'entraînement et de l'ensemble de validation. Cela suggère que le modèle n'est pas en train de surapprendre les données d'entraînement et qu'il pourrait bien se généraliser à de nouvelles images qui ne font pas partie de l'ensemble de données initial.

D'après le graphique ci-dessus, les courbes se suivent et ne présentent pas de réelles divergences, de bonnes nouvelles pour éviter le surapprentissage.

3 - Matrice de confusion

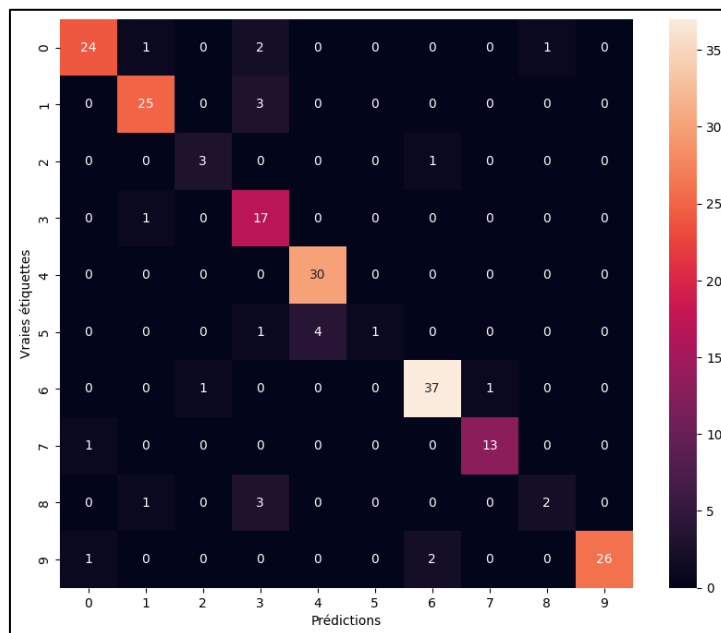


Figure 16 - Matrice de confusion

Nous avons établi une matrice de confusion à la suite de l'entraînement. La matrice de confusion joue un rôle essentiel lors de l'évaluation des performances d'un réseau de neurones ou d'un modèle de classification. Elle permet d'analyser les résultats des prédictions en comparant les étiquettes réelles des données avec les prédictions du modèle. Elle offre une vision approfondie des performances du modèle en identifiant les erreurs de classification spécifiques et les tendances associées. En utilisant la matrice de confusion, on peut calculer des métriques telles que la précision, le rappel, le taux de faux positifs et le score F1, fournissant des informations quantitatives sur les performances du modèle.

Sur cette matrice de confusion, nous retrouvons dans la diagonale les valeurs de photos prédites qui s'avèrent (ou non). Nous remarquons que pour la plupart des 10 catégories de pokémon choisies, nous avons globalement un bon score entre les données et les prédictions. La catégorie 2, par exemple, offre un scoring très bas contrairement à ses consœurs. Et pour cause : c'est une catégorie de pokémon dans laquelle seules 28 images sont utilisées pour l'entraînement, contrairement à la catégorie 6 dans laquelle 277 images sont utilisées afin d'entraîner le modèle.

Nous remarquons ainsi l'importance de la taille de l'échantillon d'entraînement dans la performance du modèle. Une quantité insuffisante de données d'entraînement dans une catégorie spécifique peut conduire à des prédictions moins précises pour cette catégorie. Cela met en évidence le besoin d'un équilibre dans la répartition des données d'entraînement pour chaque catégorie, afin d'obtenir des performances homogènes sur l'ensemble du modèle. De plus, la matrice de confusion nous permet d'identifier les erreurs de classification spécifiques, telles que les faux positifs et les faux négatifs, qui peuvent être des indications importantes pour affiner le modèle. En analysant ces erreurs, il est possible de comprendre les caractéristiques particulières

des catégories de pokémon qui posent problème et d'apporter des améliorations ciblées au modèle, que ce soit en ajustant les hyperparamètres, en augmentant la quantité de données d'entraînement ou en utilisant des techniques de prétraitement des images. En résumé, la matrice de confusion offre une vue détaillée des performances du modèle, met en évidence les points forts et les points faibles, et permet de guider les actions correctives pour améliorer la précision des prédictions.

4 - Test du modèle

Après avoir effectué plusieurs tests sur notre modèle CNN entraîné, nous avons soumis une image provenant de notre galerie de test pour évaluer ses performances. Dans ce cas précis, le modèle a identifié le pokémon présent sur l'image comme étant un "bulbizarre" avec une probabilité de 60%. Cette probabilité représente le degré de similarité entre l'image soumise et les images utilisées lors de l'entraînement du modèle.

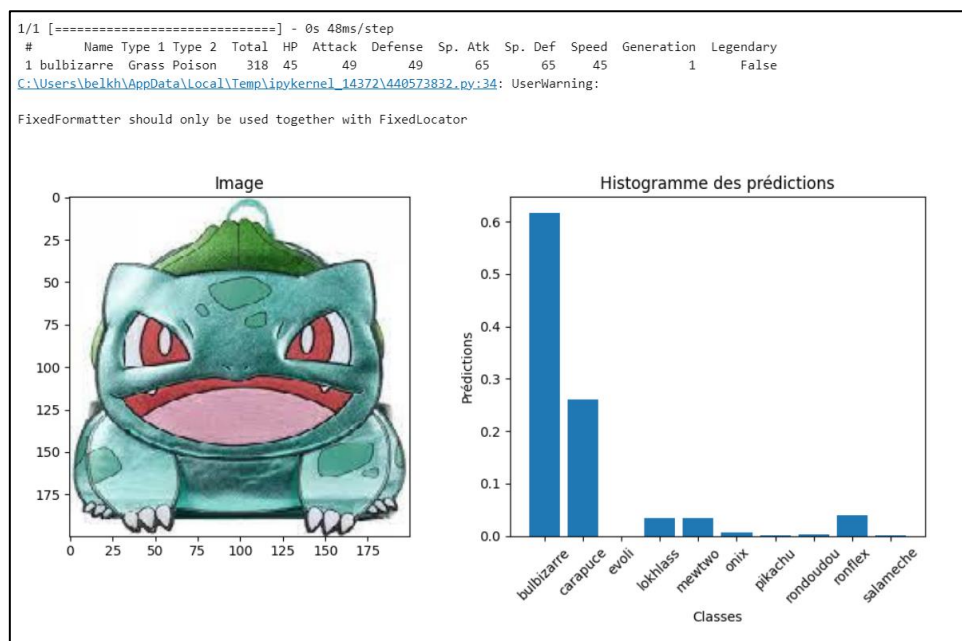


Figure 17 - Test du modèle sur une image de Bulbizarre

Il est important de souligner que le modèle recherche la ressemblance la plus forte entre l'image soumise et les exemples qu'il a été entraîné à reconnaître. Cette approche repose sur l'apprentissage des caractéristiques visuelles spécifiques à chaque classe de pokémon à partir des données d'entraînement.

Le fait que le modèle ait identifié correctement le pokémon comme étant un "bulbizarre" avec une probabilité de 60% indique que certaines caractéristiques visuelles de l'image soumise correspondent à celles des "bulbizarres" présents dans l'ensemble d'entraînement. Cependant, la probabilité de 60% suggère également que le modèle n'est pas entièrement certain de sa prédiction et qu'il existe une certaine incertitude quant à l'exactitude de cette classification.

V - Discussion des résultats

1 – Points forts du modèle

En analysant les résultats de précision de notre modèle CNN, nous pouvons constater qu'il a obtenu une précision globale de 88% sur l'ensemble des classes de pokémon. Cela signifie que dans près de 9 cas sur 10, le modèle a prédit correctement la classe à laquelle appartient un pokémon donné.

En examinant les mesures de précision pour chaque classe, nous pouvons identifier les points forts et les points faibles du modèle. Les classes "bulbizarre", "carapuce", "mewtwo", "pikachu", "rondoudou" et "ronflex" ont toutes une précision supérieure à 90%, ce qui indique que le modèle les identifie avec une grande précision.

En termes de points forts, le modèle a obtenu une précision élevée pour plusieurs classes, montrant ainsi sa capacité à bien généraliser et à prédire avec précision différentes catégories de pokémon.

Quant aux points faibles, Pour améliorer les performances du modèle, il serait recommandé de recueillir davantage de données d'entraînement pour les classes sous-représentées ou difficiles à prédire. Cela permettrait au modèle d'apprendre à mieux discriminer entre les caractéristiques des pokémons de chaque classe. En outre, des techniques de prétraitement des images ou des ajustements des hyperparamètres du modèle pourraient également être explorés pour améliorer les résultats.

En résumé, notre modèle a montré de bons résultats de précision dans l'ensemble.

2 – Points faibles du modèle

Cependant, certaines classes présentent des performances moins bonnes. Par exemple, la classe "onix" a une précision de seulement 17%, ce qui suggère que le modèle a du mal à identifier correctement les pokémons de cette catégorie. De même, la classe "salameche" a une précision de 33%, ce qui indique que le modèle fait également des erreurs significatives dans la prédiction des pokémons de cette classe.

Une autre observation intéressante concerne la classe "evoli", pour laquelle la précision est de 75%. Bien que cela soit supérieur à la moyenne, il est important de noter que cette classe présente une plus faible précision par rapport à certaines autres classes.

Les classes "onix" et "salameche" sont clairement des défis pour le modèle, ce qui est fortement corrélés avec le manque d'image dans le dataset. De plus, le peu d'image présente dans chacun des deux datasets n'est pas d'une grande qualité, augmentant ainsi le manque de fiabilité du modèle pour ces catégories précises.

3 – Pistes d'amélioration

Une première piste d'amélioration consisterait à optimiser le modèle en ajustant les hyperparamètres, en explorant différentes architectures et en testant différentes fonctions d'activation. Cela permettrait d'obtenir un modèle plus performant et plus adapté aux spécificités du problème.

De plus, l'utilisation de techniques de data augmentation pourrait contribuer à augmenter la taille de l'ensemble de données et à diversifier les exemples d'apprentissage. Cela permettrait d'améliorer la capacité du modèle à généraliser et à traiter des variations dans les images des Pokémon.

Une autre approche intéressante serait d'explorer le transfert d'apprentissage en utilisant des modèles pré-entraînés. En exploitant les connaissances préalables acquises sur de vastes bases de données d'images, il serait possible de bénéficier de caractéristiques visuelles déjà apprises. Cela permettrait de réduire le temps d'entraînement du modèle et d'améliorer ses performances.

Enfin, il serait également pertinent d'effectuer une validation croisée pour évaluer les performances du modèle de manière robuste. Cela permettrait de s'assurer que le modèle est capable de généraliser correctement et d'obtenir des résultats cohérents sur des données inconnues.

VI – Conclusion

En conclusion, ce projet de classification des Pokémon en utilisant des réseaux neuronaux convolutifs présente des résultats prometteurs. Cependant, il reste encore des possibilités d'amélioration et d'optimisation du modèle pour obtenir des performances encore meilleures.

Une des pistes d'amélioration serait d'explorer davantage les techniques d'augmentation des données. En appliquant des transformations telles que la rotation, le zoom ou le décalage sur les images d'entraînement, on pourrait augmenter la variabilité des données et améliorer la capacité du modèle à généraliser sur de nouvelles images.

Une autre approche intéressante serait d'utiliser le transfert d'apprentissage. Plutôt que d'entraîner le modèle à partir de zéro, on pourrait exploiter les connaissances acquises par des modèles pré-entraînés sur de grandes bases de données d'images. En adaptant ces modèles pré-entraînés aux spécificités des Pokémon, on pourrait bénéficier de leur capacité à extraire des caractéristiques pertinentes.

En outre, la validation croisée pourrait être utilisée pour évaluer plus rigoureusement les performances du modèle. En divisant l'ensemble de données en plusieurs plis (folds) et en effectuant plusieurs cycles d'entraînement et de validation, on obtiendrait une estimation plus fiable de la capacité du modèle à généraliser.

En poursuivant ces pistes d'amélioration, il serait possible de développer un modèle encore plus performant et précis dans la reconnaissance des Pokémon à partir d'images. Cela aurait des applications pratiques dans le domaine des jeux Pokémon, de l'analyse de données liées aux Pokémon, et même dans d'autres domaines où la classification d'images est nécessaire.

Vous pouvez maintenant devenir le meilleur dresseur du monde.



Figure 18 - Sacha Ketchum devenu meilleur dresseur du monde

Références

Noyau dans le traitement d'images : [Noyau \(traitement d'image\) – Wikipédia \(wikipedia.org\)](#)

Convolution : [S06_convolution.pdf \(cornell.edu\)](#)

Réseau de neurones convolutifs : [Convolutional Neural Network : Tout ce qu'il y a à savoir \(datascientest.com\)](#)

Projet de CNN avec Pikachu et Rondoudou : [Coder un réseau de neurones convolutifs de classification d'image avec Python et Tensorflow. - YouTube](#)

Banque d'images de Pokémon : [Pokemon Generation One | Kaggle](#)

Cf notebook pour plus de précisions sur les étapes réalisées et les résultats déjà générés.