Mini challenge sur la classification de kanjis

Résultat obtenu avec kanji_test_predictions : 0.0578833693 kanji_test_predictions.zip

Il se peut que le fichier ait été généré par une méthode aléatoire, qui attribue des étiquettes de classe sans prendre en compte les entrées. Cela se traduirait par une précision proche de la probabilité de deviner correctement un caractère parmi le nombre total de classes (20).
Cette supposition est plausible car *1/20 = 0,05. On est donc proche du score obtenu.*

Visualisation t-SNE des caractères Kanji

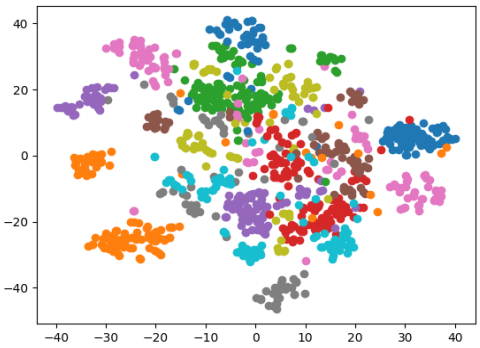
Objectif

- Utiliser l'algorithme t-SNE (t-distributed Stochastic Neighbor Embedding) pour visualiser la distribution spatiale des caractères kanji dans un espace à deux dimensions.

Méthodologie

1. Chargement des Données : Les données sont chargées à partir de fichiers CSV, séparant les caractéristiques des cibles (kanji_train_data.csv et kanji_train_target.csv).
2. Application de t-SNE : L'algorithme t-SNE est appliqué aux données pour réduire leur dimensionnalité de l'espace original à un espace 2D.
3. Visualisation : Les points sont visualisés dans un graphique, où chaque point représente un caractère kanji. Les points sont colorés différemment selon leur catégorie (cible) pour distinguer les différents groupes de kanji.

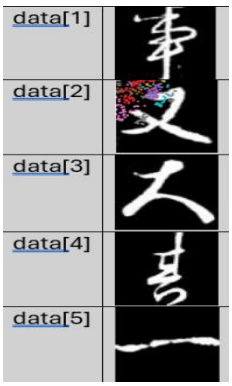
Résultats et Discussion



Le script génère un graphique montrant la distribution spatiale des premiers 1000 kanji dans un espace 2D après l'application de t-SNE.

Les clusters visibles dans le graphique peuvent indiquer des similitudes structurelles entre certains caractères kanji, permettant de mieux comprendre les relations entre eux.

display_kanji() :



La fonction display_kanji est conçue pour visualiser les caractères kanji, transformant des données numériques en images interprétables. Cette fonction prend un vecteur aplati représentant un kanji et le reforme en une matrice 64x64, correspondant aux dimensions d'une image en niveaux de gris. L'image est ensuite affichée sans axes pour une meilleure clarté visuelle. Cette méthode est particulièrement utile pour vérifier les données pendant le prétraitement ou pour examiner les résultats d'une classification ou d'une réduction de dimensionnalité, comme celle effectuée par t-SNE. Pour utiliser cette fonction, il suffit de fournir un vecteur de caractéristiques de la taille appropriée, et la fonction s'occupe de la représentation graphique.

Classification KNN des Caractères Kanji : Détermination et Justification du k Optimal

Objectif

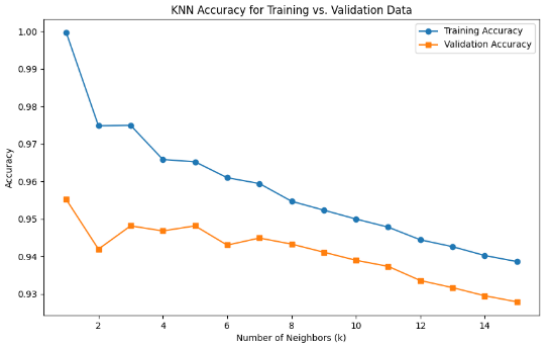
- Le but est de classifier les caractères kanji avec le classificateur K-Nearest Neighbors (KNN) et de déterminer le k qui offre la meilleure performance.

Méthodologie

- La procédure a suivi la séparation des données en ensembles d'entraînement et de validation pour ajuster et évaluer le modèle KNN. Des valeurs de k de 1 à 15 ont été testées pour observer leur impact sur la précision du modèle, cherchant à éviter le surapprentissage et le sous-apprentissage.

Visualisation et Sélection de k

- Un graphique a permis de visualiser la précision sur les deux ensembles. La convergence des précisions suggérait un modèle équilibré, avec un choix de k=5 résultants d'une haute précision de validation (0.9482) et une précision d'entraînement suffisante (0.9652), indiquant une bonne généralisation sans biais excessif envers les données d'entraînement.



Résultats

Le choix de k=5 représente un compromis optimal pour la classification KNN des kanji, équilibrant précision et généralisation. Ce k a été ensuite utilisé pour prédire des caractères sur l'ensemble de test, avec des résultats sauvegardés pour des évaluations futures.

```
k = 1: Training Accuracy = 0.9997, Validation Accuracy = 0.9552
k = 2: Training Accuracy = 0.9748, Validation Accuracy = 0.9420
k = 3: Training Accuracy = 0.9750, Validation Accuracy = 0.9482
k = 4: Training Accuracy = 0.9658, Validation Accuracy = 0.9468
k = 5: Training Accuracy = 0.9652, Validation Accuracy = 0.9482
k = 6: Training Accuracy = 0.9610, Validation Accuracy = 0.9430
k = 7: Training Accuracy = 0.9594, Validation Accuracy = 0.9449
k = 8: Training Accuracy = 0.9547, Validation Accuracy = 0.9433
k = 9: Training Accuracy = 0.9523, Validation Accuracy = 0.9411
k = 10: Training Accuracy = 0.9500, Validation Accuracy = 0.9390
k = 11: Training Accuracy = 0.9478, Validation Accuracy = 0.9374
k = 12: Training Accuracy = 0.9444, Validation Accuracy = 0.9336
k = 13: Training Accuracy = 0.9426, Validation Accuracy = 0.9317
k = 14: Training Accuracy = 0.9403, Validation Accuracy = 0.9295
k = 15: Training Accuracy = 0.9386, Validation Accuracy = 0.9279
```

En somme, la détermination du k optimal s'est basé sur la recherche d'une précision élevée et constante à travers les données d'entraînement et de validation, tout en assurant une capacité de généralisation robuste. La méthode choisie pour le KNN, avec k=5, montre une performance solide et constitue une base fiable pour de futures applications de reconnaissance des kanjis.

Score sur la phase de test (Codalab) :

0.9552915767	kanji_test_predictions.zip

Régression Logistique pour la Classification des Caractères Kanji

Objectif

- Appliquer la régression logistique pour classifier les caractères kanji et identifier la meilleure valeur du paramètre de régularisation C.

Méthodologie

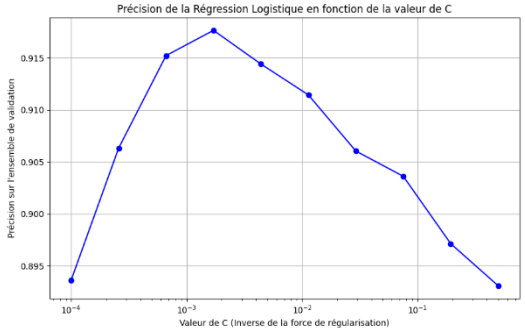
- Prétraitement des Données : Standardisation des caractéristiques des données d'entraînement.
- Recherche de la Meilleure Valeur de **C** : Entraînement de modèles de régression logistique avec différentes valeurs de C et évaluation de leur précision et du score F1 sur l'ensemble de validation.
- Visualisation des Performances : Deux graphiques sont générés pour montrer l'impact de la valeur de C sur la précision et le score F1 Macro.

Résultat sans étude paramétrique

0.8807775378	kanji_test_predictions.zip

Voyons si l'on peut trouver les paramètres optimaux (valeur de C & F1 Macros) :

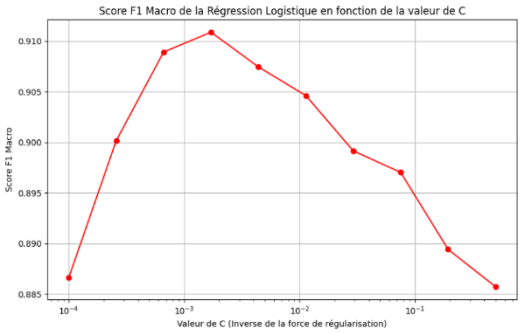
```
C=0.0001, Validation Accuracy: 0.89
C=0.0003, Validation Accuracy: 0.91
C=0.0007, Validation Accuracy: 0.92
C=0.0017, Validation Accuracy: 0.92
C=0.0044, Validation Accuracy: 0.91
C=0.0113, Validation Accuracy: 0.91
C=0.0292, Validation Accuracy: 0.91
C=0.0753, Validation Accuracy: 0.90
C=0.1941, Validation Accuracy: 0.90
C=0.5000, Validation Accuracy: 0.89
```



Si plusieurs valeurs de C donnent des précisions similaires, on peut choisir la plus petite de ces valeurs. Ceci est conforme au principe du rasoir d'Ockham (ou principe de parcimonie) qui suggère qu'entre plusieurs modèles ayant des performances similaires, il est préférable de choisir le plus simple. En termes de régularisation, une valeur plus petite de C (sans être trop petite pour éviter le sous-apprentissage) implique une régularisation plus forte, ce qui peut aider à prévenir le sur-apprentissage.

D'après les résultats, les meilleures précisions sont obtenues pour C = 0.0007 et C=0.0017, toutes deux avec une précision de 0.92. Parmi ces options, choisir la valeur la plus petite (soit C = 0.0007) peut être judicieux pour favoriser la simplicité et la généralisation.

```
C=0.0001, F1 Macro Score: 0.89
C=0.0003, F1 Macro Score: 0.90
C=0.0007, F1 Macro Score: 0.91
C=0.0017, F1 Macro Score: 0.91
C=0.0044, F1 Macro Score: 0.91
C=0.0113, F1 Macro Score: 0.90
C=0.0292, F1 Macro Score: 0.90
C=0.0753, F1 Macro Score: 0.90
C=0.1941, F1 Macro Score: 0.89
C=0.5000, F1 Macro Score: 0.89
```



Concernant le score F1, il est particulièrement sensible aux performances dans les classes minoritaires, mais si aucune classe n'est significativement plus difficile à prédire que les autres, les deux métriques pourraient montrer des tendances similaires. Le score F1 est une moyenne harmonique de la précision et du rappel. Sachant que les deux métriques suivent bien une tendance similaire cela veut dire que le variation du rappel n'est pas très significative.

Résultats et Discussion

L'analyse révèle l'importance de la sélection du paramètre de régularisation dans la performance du modèle de régression logistique. La valeur optimale de C est choisie en fonction de la précision et du score F1 Macro sur l'ensemble de validation.

Le modèle final est entraîné avec la valeur optimale de C et utilisé pour prédire les étiquettes des caractères kanji dans l'ensemble de test. Les prédictions sont enregistrées dans un fichier CSV.

Lorsque l'on compare les performances de la régression logistique et du classificateur KNN, il est important de noter que malgré les améliorations apportées par la régression logistique, le KNN peut toujours surpasser ce dernier en termes de précision globale pour cette tâche spécifique. La raison pour laquelle le KNN peut s'avérer plus efficace tient à sa capacité à capter des motifs complexes dans les données sans supposer de frontière de décision linéaire, contrairement à la régression logistique qui, malgré la régularisation, impose un modèle linéaire. De plus, les caractéristiques uniques des caractères kanji pourraient mieux se prêter à l'approche basée sur la proximité du KNN plutôt qu'aux frontières de décision rigides de la régression logistique.

Réseau de neurones avec des couches linéaires

Objectif

- Utiliser un réseau de neurones multicouches pour classifier les caractères kanji et déterminer la configuration optimale des couches cachées.

Méthodologie

- Prétraitement des Données : Les caractéristiques des données d'entraînement sont standardisées.
- Exploration des Configurations : Différentes configurations de couches cachées sont testées pour évaluer leur impact sur la précision et le score F1.
- Évaluation des Performances : Des graphiques sont générés pour illustrer l'exactitude et le score F1 de chaque configuration testée.

L'algorithme utilisé, MLPClassifier de la bibliothèque scikit-learn, crée un modèle de réseau de neurones multicouche perceptron (MLP) avec des couches entièrement connectées (aussi connues sous le nom de couches linéaires ou denses dans d'autres frameworks).

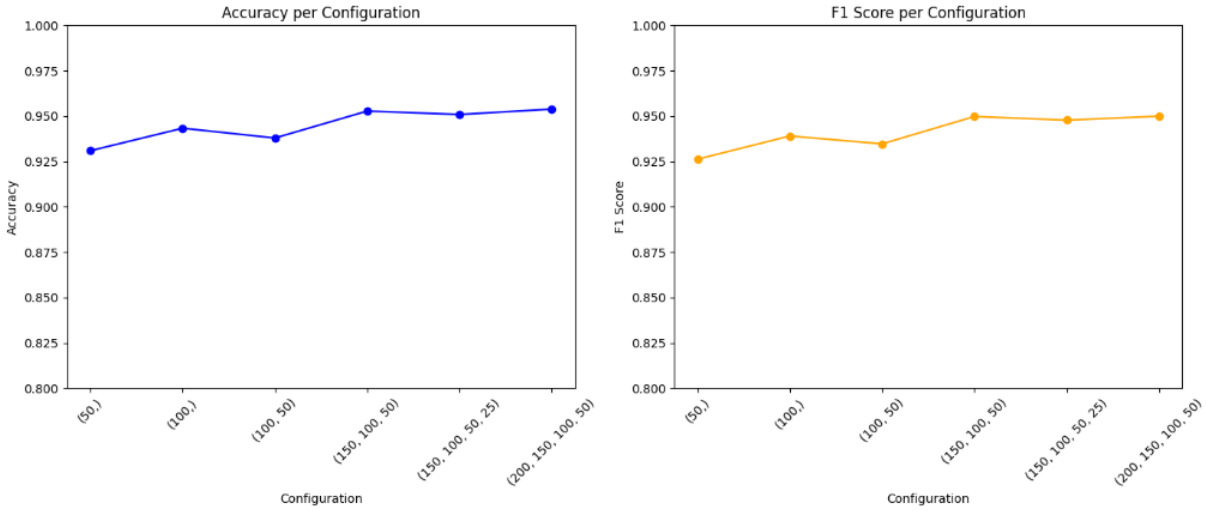
Chaque neurone dans une couche est connecté à tous les neurones de la couche précédente et de la couche suivante. Ce sont des structures de réseau de neurones de base sans couches convolutives, récurrentes ou d'autres types de couches plus spécialisées que l'on pourrait trouver dans des architectures plus complexes pour des tâches comme la reconnaissance d'images ou le traitement du langage naturel.

score du modèle de réseau de neurones avec des couches linéaires sur l'ensemble de test

0.9483801296	kanji_test_predictions.zip

Voyons si on peut améliorer ce résultat. On va tester les différentes configurations de couches cachées pour trouver la plus optimal en termes de performance, de complexité, de temps d'entrainements & d'économie de ressources.

```
Configuration (50,), Accuracy: 93.09%, F1 Score: 0.93
Configuration (100,), Accuracy: 94.33%, F1 Score: 0.94
Configuration (100, 50), Accuracy: 93.79%, F1 Score: 0.93
Configuration (150, 100, 50), Accuracy: 95.28%, F1 Score: 0.95
Configuration (150, 100, 50, 25), Accuracy: 95.09%, F1 Score: 0.95
Configuration (200, 150, 100, 50), Accuracy: 95.38%, F1 Score: 0.95
```



Résultats et Discussion

Les performances du modèle de réseau de neurones sont évaluées en utilisant différentes structures de couches cachées. Alors que la configuration la plus complexe (200, 150, 100, 50) offre une légère amélioration des scores, il a été observé que la configuration (150, 100, 50) présente un bon équilibre en termes de performance et de complexité du modèle.

L'ajout de couches et de neurones supplémentaires a montré des gains de performance décroissants, ce qui soulève des questions sur la nécessité de complexifier davantage le modèle en considérant le coût computationnel. Cela reflète la loi des rendements décroissants en apprentissage machine : au-delà d'un certain point, les améliorations deviennent marginales par rapport à l'augmentation de la complexité.

En sélectionnant la configuration (150, 100, 50), nous bénéficions d'une efficacité computationnelle et d'un risque moindre de surajustement, tout en maintenant de solides performances. Cette configuration a été utilisée pour former le modèle final et effectuer des prédictions sur les données de test, avec des résultats sauvegardés pour un usage ultérieur.

Résultat sur les données du Challenge après amélioration :

0.9514038877	kanji_test_predictions.zip

Réseau de Neurones Convolutionnels

Objectif

- L'objectif est de développer un modèle de réseau de neurones convolutionnels (CNN) pour la classification des caractères kanji, optimisé pour la précision et l'évitement du surapprentissage.

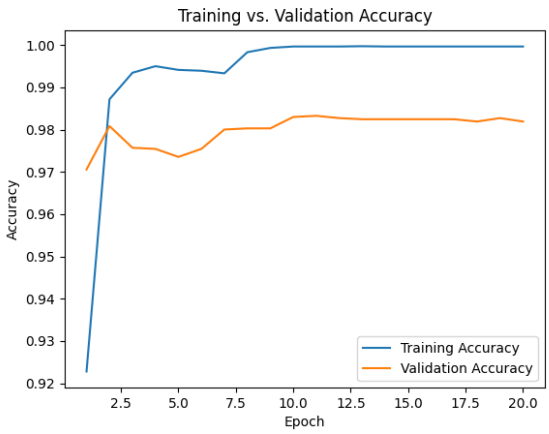
Méthodologie

- Prétraitement des Données : Normalisation des données d'entraînement et reformulation en un format approprié pour le traitement CNN.
- Conception du Modèle : Création d'un modèle CNN avec deux couches convolutionnelles suivies de couches entièrement connectées, en utilisant la bibliothèque PyTorch et en exécutant sur un GPU via CUDA pour l'accélération du calcul.
- Entraînement et Validation : Séparation des données en ensembles d'entraînement et de validation, suivi d'un entraînement supervisé avec évaluation des performances sur les deux ensembles.

Résultats et Discussion

```
Epoch 1, Loss: 0.2597, Train Accuracy: 0.9228, Validation Accuracy: 0.9706
Epoch 2, Loss: 0.0275, Train Accuracy: 0.9872, Validation Accuracy: 0.9808
Epoch 3, Loss: 0.0060, Train Accuracy: 0.9935, Validation Accuracy: 0.9757
Epoch 4, Loss: 0.0208, Train Accuracy: 0.9950, Validation Accuracy: 0.9754
Epoch 5, Loss: 0.0067, Train Accuracy: 0.9941, Validation Accuracy: 0.9735
Epoch 6, Loss: 0.0039, Train Accuracy: 0.9939, Validation Accuracy: 0.9754
Epoch 7, Loss: 0.0207, Train Accuracy: 0.9933, Validation Accuracy: 0.9800
Epoch 8, Loss: 0.0000, Train Accuracy: 0.9983, Validation Accuracy: 0.9803
Epoch 9, Loss: 0.0005, Train Accuracy: 0.9993, Validation Accuracy: 0.9803
Epoch 10, Loss: 0.0000, Train Accuracy: 0.9997, Validation Accuracy: 0.9830
Epoch 11, Loss: 0.0000, Train Accuracy: 0.9997, Validation Accuracy: 0.9833
Epoch 12, Loss: 0.0000, Train Accuracy: 0.9997, Validation Accuracy: 0.9827
Epoch 13, Loss: 0.0000, Train Accuracy: 0.9997, Validation Accuracy: 0.9825
Epoch 14, Loss: 0.0000, Train Accuracy: 0.9997, Validation Accuracy: 0.9825
Epoch 15, Loss: 0.0000, Train Accuracy: 0.9997, Validation Accuracy: 0.9825
Epoch 16, Loss: 0.0000, Train Accuracy: 0.9997, Validation Accuracy: 0.9825
Epoch 17, Loss: 0.0000, Train Accuracy: 0.9997, Validation Accuracy: 0.9825
Epoch 18, Loss: 0.0000, Train Accuracy: 0.9997, Validation Accuracy: 0.9819
Epoch 19, Loss: 0.0000, Train Accuracy: 0.9997, Validation Accuracy: 0.9827
```

Le CNN conçu a été entraîné sur plusieurs époques, montrant une amélioration constante de la précision d'entraînement et de validation, indiquée par la courbe de précision. L'utilisation d'un ensemble de validation distinct aide à surveiller et à prévenir le surapprentissage.



Le graphique montre que, durant les premières époques, la précision de validation suit de près la précision d'entraînement, mais à partir d'une certaine époque, la précision de validation cesse de s'améliorer et commence même à diminuer légèrement, tandis que la précision d'entraînement reste presque parfaite. Ce phénomène est un signe classique d'overfitting, où le modèle apprend par cœur les données d'entraînement mais ne parvient pas à généraliser aussi efficacement sur les données qu'il n'a jamais vues.

Résultat pour le modèle qui utilise deux ensembles (Entrainement & Validation)

0.9799136069	kanji_test_predictions.zip

Résultat dont le modèle utilise l'intégralité du dataset pour l'entrainement

0.9844492441	kanji_test_predictions.zip

Cela pourrait indiquer que le modèle a réussi à capturer des caractéristiques essentielles pour la classification des kanjis, et que l'utilisation de l'ensemble de données complet pour l'entraînement a probablement permis au modèle d'apprendre des détails fins qui ont contribué à la réussite des prédictions sur l'ensemble de test. Cependant, il convient de noter que bien que l'entraînement sur

l'ensemble complet de données puisse entraîner de meilleurs résultats sur un ensemble de test spécifique, il peut y avoir des risques de surajustement, comme l'indique l'augmentation de la précision d'entraînement par rapport à la précision de validation. En l'absence de validation, il est difficile de garantir que les résultats obtenus se généraliseront à d'autres ensembles de données de kanji non vus. Une approche prudente consisterait à utiliser des techniques comme la validation croisée pour évaluer la capacité de généralisation du modèle.

Conclusion

Les résultats mettent en évidence la supériorité des CNN pour cette application spécifique, justifiant leur réputation dans le traitement d'images. Cependant, il est crucial de maintenir une stratégie rigoureuse pour éviter le surapprentissage, en utilisant des ensembles de validation et éventuellement des techniques comme la validation croisée ou la régularisation. La haute précision obtenue suggère que les CNN sont bien adaptés pour la reconnaissance de caractères kanji et pourraient potentiellement être utilisés pour des applications en temps réel, avec une prudence quant à la généralisation des résultats.