Reconnaissance de dessins manuscrits avec jeu de données Quick, Draw!

William Bourget, Samuel Lévesque

Département d'informatique et de génie logiciel, Université Laval

†Authors contributed equally to this work.

Introduction

Nous souhaitons livrer un produit utilisable en temps réel par l'entremise de ce projet. Les projets académiques habituellement réalisés se concentrent sur les l'optimisatione et la comparaison de modèles, mais nous n'avons jamais eu la chance de terminer la chaîne d'un projet et d'utiliser les prédictions des modèles développés en **temps réel**.

Motivations :

- Application des techniques de **traitement d'images** à partir d'un grand corpus de données.
- Déploiement complet d'un modèle prédictif pour usage par des non-experts.

Buts :

- Apprendre à traiter efficacement de grandes quantités de données.
- Se familiariser avec les outils d'**infonuagique** des grands joueurs pour projets à grande échelle.
- Développer une **interface graphique** permettant à des usagers de tester notre modèle en temps réel.
- Couvrir l'ensemble de la chaîne d'un projet en intelligence artificielle du traitement des données à l'utilisation du modèle dans un contexte de vie réelle.

Contraintes

Plusieurs contraintes dans la réalisation de ce projet nous ont empêchées de tester toutes les avenues que nous avions initialement envisagées. **Contraintes computationnelles :**

- Enjeux de mémoire :
  - Incapacité à charger l'ensemble du jeu de données en **mémoire**.
  - Conservation des images en **format vectoriel** pour limiter espace de stockage nécessaire.
- Enjeux de performance :
  - Temps de traitement très important nous limite dans le nombre de modèles et de jeux d'hyperparamètres qu'on peut tester.
  - Impossibilité d'entraîner les modèles sur nos postes personnels. Nous devons utiliser des solutions d'**infonuagique** qui peuvent être coûteuses et complexe à utiliser avec un jeu de données aussi important.

Contraintes de l'interface graphique :

- Perte de l'**information temporelle** des images vectorielles.
- Impossibilité d'utiliser **différents canaux** pour simuler l'évolution temporelle du dessin.
- Perte de la séparation du dessin par trait.

Description des données

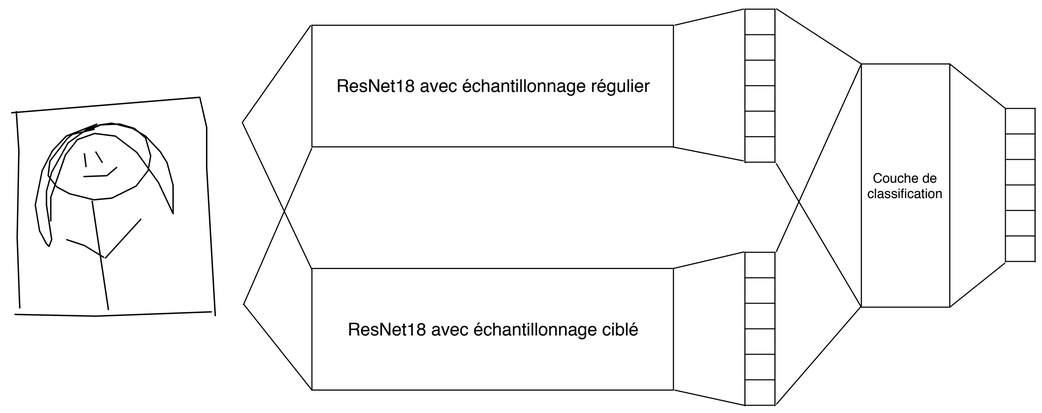
Le jeu de données de Quick, Draw ! est un sous-échantillon du jeu de données de Google AI contenant plus d'un milliard d'images. Voici quelques caractéristiques du jeu de données utilisé pour l'entraînement du modèle :

- Jeux de données bruts et simplifiés disponibles (le jeu de données simplifié élimine des points inutiles qui se trouvent sur une même droite).
- **340 classes**.
- **≈ 150 000 images/classe**.
- Plus de **51 millions d'images**
- **24,4 Gb** de données compressées en format .csv.

Utilisation des données compressées pour le projet pour limiter les enjeux de mémoire autant que possible sous hypothèse que les performances du modèle n'en souffriraient pas.

Structure du réseau et méthodologie

Plusieurs variantes du réseau pré-entraîné *Resnet18*



Étant donné les contraintes, utilisations d'epochs non traditionnelles, chaque epoch consiste en un **échantillonnage d'un nombre fixe d'images par classe** (ex : 500 images aléatoires par classe : epoch de 170 000 données plutôt que le jeu complet).
Similaire à du **Bootstrapping** .
Variantes :

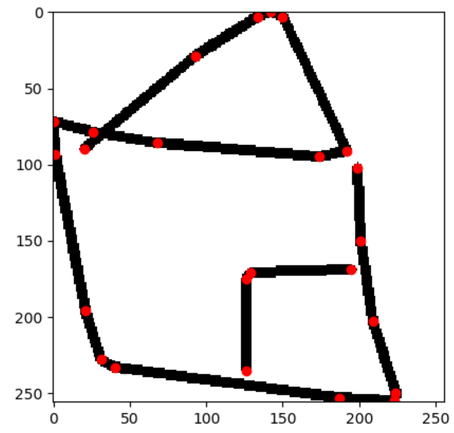
- Échantillonnage variant selon l'exactitude par classe : $N^* = N(1 - accuracy) + 0.25N$
- Méthode par ensemble avec couche de classification
- Méthode par ensemble avec moyenne

Transformations

Transformation de vecteurs de traits de crayon (position x, y) en photos.

x	20	93	134	142	150	192	174	68	26	0	1	21	31	40	187	223	224	209	201	199
y	90	29	3	0	3	91	95	86	79	72	93	196	228	233	253	255	250	203	150	102

x	126	126	129	195
y	235	175	171	169



- Images très différentes pour une même classe
- Certaines images incomplètes

Performances

Modèle	Accuraccy	MAP@3
Resnet18	0.7757	0.8378
Resnet18 données ciblés	**0.7958**	0.8542
Ensemble avec classif	0.7592	0.8264
Ensemble avec moyenne	63	123

Gain d'environ 2% d'accuracy avec la méthode du nombre d'échantillons variables par classe.

Conclusion

Discussion :

- **Morphology** and **context** help predict useful embeddings.
- **The attention mechanism works** : depending on the task, the network will use either more the context or the morphology to generate an embedding.

Future works :

- Apply the **attention mechanism on each character of the OOV word and each word of the context** instead of using the hidden state of the respective elements only.
- Test our attention model in **different languages** and on other NLP tasks, such as **machine translation**.