Titre cool

William Bourget et Samuel Lévesque

Université Laval

May 2, 2019

ABSTRACT

1 Introduction

C'est l'intro

das d as da sd as gsd h hhhhhhfghfgh tdh fgj fdj fg fg

2 Description des données

Google AI rend disponible une partie du jeux de données de *Quick, Draw*[1] qui est constitué de milliards de dessins fait à la main par différents utilisateurs sur leur plateforme web.

Les données sont fournies sous forme de fichiers csv (1 par classe) contenant principalement des vecteurs de positions ayant permis de réaliser les traits du dessin, le pays de l'utilisateur et la classe du dessin.

Il s'agit d'un jeu de données extrêmement volumineux, on note certaines caractéristiques :

- **340 classes**.
- $\approx$ **150 000 images/classe**.
- Plus de **51 millions d'images** au total
- **24,4 Gb** de données sous format .csv.

3 Contraintes et enjeux

Une quantité de données aussi importante crée des enjeux et problèmes importants dans l'entraînement du réseau. On présente ici les enjeux principaux qui nous amènent à tester différentes manières d'entraînement.

[1] Lien

3.1 Enjeux de mémoire

Pour pouvoir utiliser des réseaux de neurones à convolution, il faut utiliser des données étant sous forme d'image. Étant donné la quantité massive de données, il ne semble pas optimal de transformer toutes les données du format csv en images directement avant l'entraînement. Pour pallier à ce problème et pour pouvoir utiliser des réseaux pré-entraîné , on transforme le format csv des traits de crayon au moment où on charge les données dans le modèle.

3.2 Enjeux de performance

La quantité massive de données ne nous permet pas de faire des epochs traditionnelles. En effet, avec les moyens de calculs que nous utilisons (*Google Colab*) , une seule epoch peut prendre environ 1 mois en calculant 24h/24.

Pour cette même raison, on ne peut pas tester une grande quantité d'hyperparamètres. Il faut se limiter un ensemble d'hyperparamètres pour chacun des modèles que l'on veut tester.

3.3 Enjeux d'une application réelle

La mise en place d'une application concrète nous amène également quelques contraintes supplémentaires. L'application créée nous fait perdre l'information temporelle des traits de crayons puisque la sortie de l'application est une image de format png. On ne peut donc pas utiliser l'ordre dans lequel les traits ont été dessinés par l'utilisateur. Il nous est donc impossible d'utiliser différents canaux pour simuler l'évolution temporelle du dessin

4 Transformations des données

La figure 1 nous montre un exemple de transformation d'une image de maison réalisé en 2 traits de crayons.

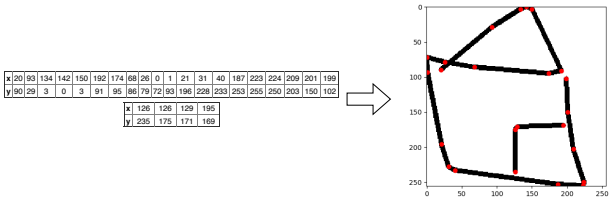

Figure 1: *Transformation de vecteur à image*

La plupart des images sont loin d'être facile à apprendre pour un ordinateur, puisque que ces données ne sont pas filtrées par un humain qui peut valider si l'image est bonne. N'importe quel individu peut contribuer à cette base de données. Pour cette raison, on peut observer des images qui peuvent varier énormément pour une même classe. La figures 2 nous permet de voir différentes images pour la classe *frog*.

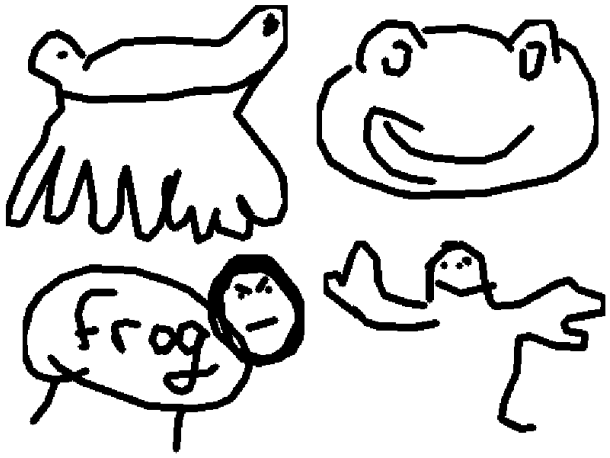

Figure 2: *Différentes images de la classe frog*

On peut voir que les utilisateurs n'adoptent pas tous les mêmes techniques pour dessiner un même objet. Également puisque les utilisateurs ont seulement 20 secondes pour faire leur dessin, certains d'entre eux sont parfois incomplets. Comme aucun filtrage n'a été fait, on peut constater également que certaines images ne sont pas réalisés dans le but exact du projet (certains utilisateurs ne dessinent pas la bonne catégorie ou bien ne font qu'écrire le nom de la classe)

5 Structure du modèle et méthodologie

En gardant nos contraintes et enjeux en tête, il fallait trouver une façon efficace d'entraîner un modèle dans un temps raisonnable. Le premier modèle testé est un modèle pré-entraîné avec architecture *Resnet18*, mais avec un seul canal de couleur au lieu de 3.

6 Performances

C'est l'intro

das d as da sd as gsd h hhhhhhfghfgh tdh fgj fdj fg fg

7 Conclusion

C'est l'intro

das d as da sd as gsd h hhhhhhfghfgh tdh fgj fdj fg fg