

Introduction

Nous souhaitons livrer un produit utilisable en temps r  el par l'entremise de ce projet. Les projets acad  miques habituellement r  alis  s se concentrent sur les l'optimisation et la comparaison de mod  les, mais nous n'avons jamais eu la chance de terminer la cha  ne d'un projet et d'utiliser les pr  dictions des mod  les d  velopp  s en **temps r  el**.

Motivations :

- Application des techniques de **traitement d'images**    partir d'un grand corpus de donn  es.
- D  ploiement complet d'un mod  le pr  dictif pour usage par des non-experts.

Buts :

- Apprendre    traiter efficacement de grandes quantit  s de donn  es.
- Se familiariser avec les outils d'**infonuagique** des grands joueurs pour projets    grande   chelle.
- D  velopper une **interface graphique** permettant    des usagers de tester notre mod  le en temps r  el.
- Couvrir l'ensemble de la cha  ne d'un projet en intelligence artificielle du traitement des donn  es    l'utilisation du mod  le dans un contexte de vie r  elle.

Contraintes

Plusieurs contraintes dans la r  alisation de ce projet nous ont emp  ch  es de tester toutes les avenues que nous avons initialement envisag  es. **Contraintes computationnelles :**

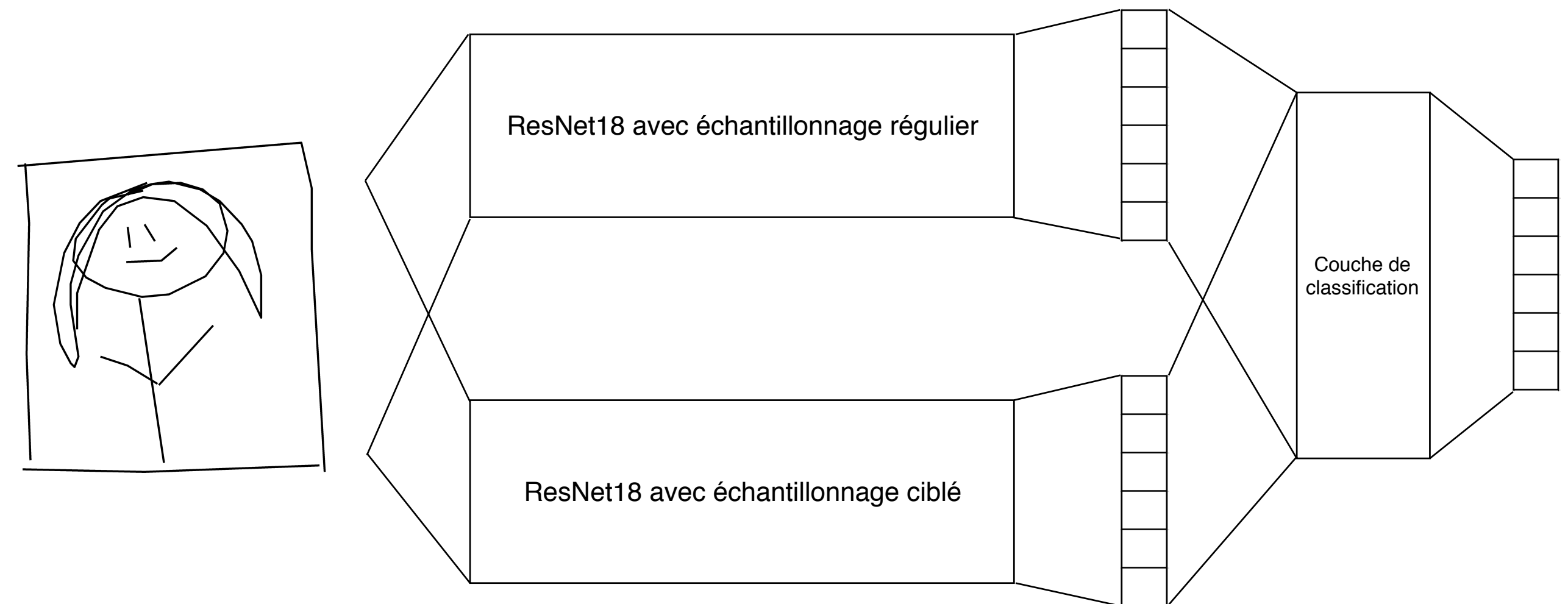
- Enjeux de m  moire :
 - Incapacit      charger l'ensemble du jeu de donn  es en **m  moire**.
 - Conservation des images en **format vectoriel** pour limiter espace de stockage n  cessaire.
- Enjeux de performance :
 - Temps de traitement tr  s important nous limite dans le nombre de mod  les et de jeux d'hyperparam  tres qu'on peut tester.
 - Impossibilit   d'entra  ner les mod  les sur nos postes personnels. Nous devons utiliser des solutions d'**infonuagique** qui peuvent   tre co  teuses et complexe    utiliser avec un jeu de donn  es aussi important.

Contraintes de l'interface graphique :

- Perte de l'**information temporelle** des images vectorielles.
- Impossibilit   d'utiliser **diff  rents canaux** pour simuler l'  volution temporelle du dessin.
- Perte de la s  paration du dessin par trait.

Structure du r  seau et m  thodologie

Plusieurs variantes du r  seau pr  -entra  n   *Resnet18*



  tant donn   les contraintes, utilisations d'epochs non traditionnelles, chaque epoch consiste en un **  chantillonnage d'un nombre fixe d'images par classe** (ex : 500 images al  atoires par classe : epoch de 170 000 donn  es plut  t que le jeu complet).

Similaire    du **Bootstrapping**.

Variantes :

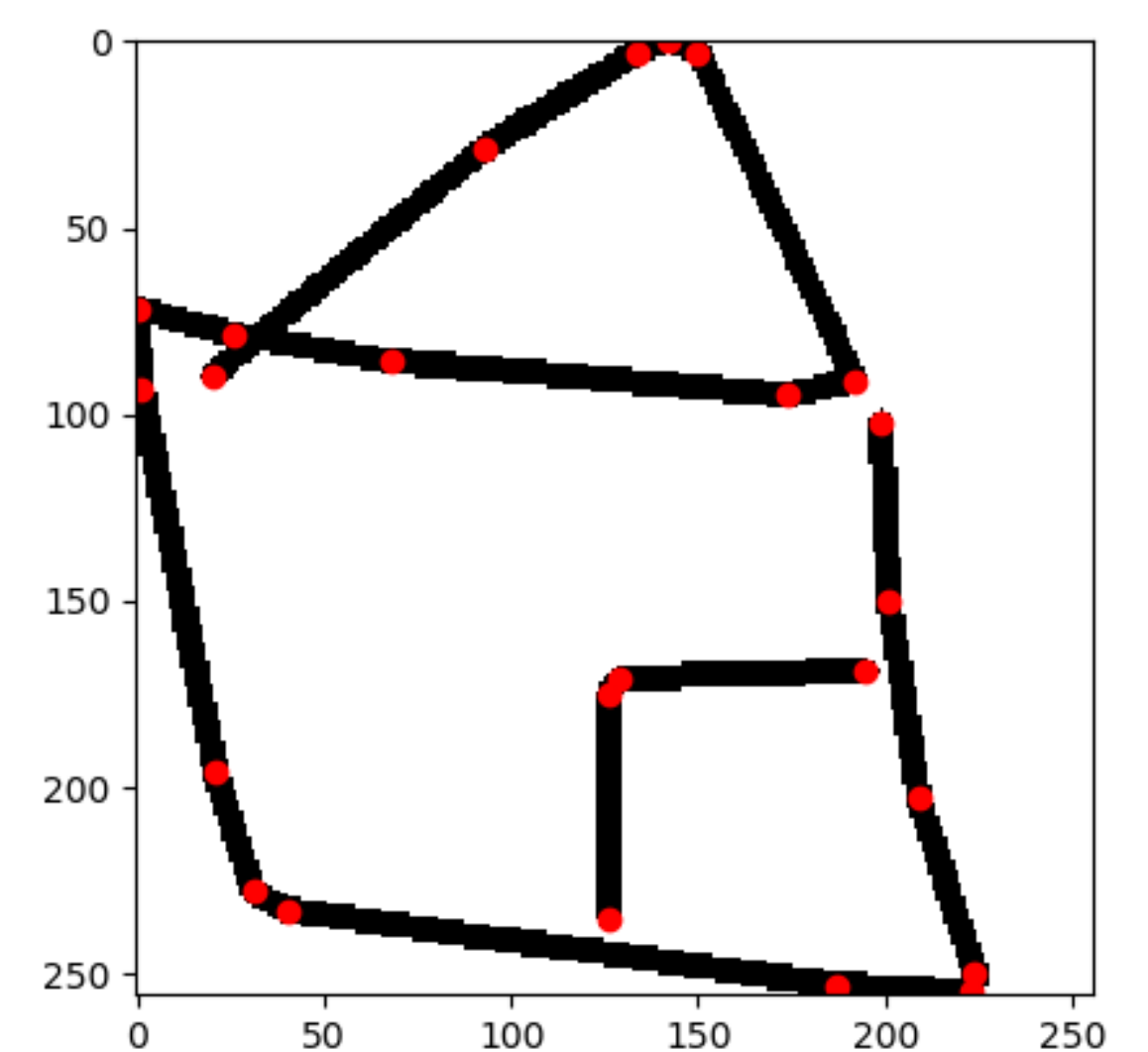
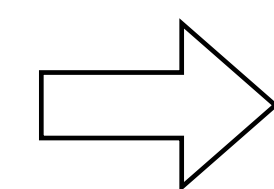
-   chantillonnage variant selon l'exactitude par classe : $N^* = N(1 - accuracy) + 0.25N$
- M  thode par ensemble avec couche de classification
- M  thode par ensemble avec moyenne

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 tr  s diff  rentes pour une m  me classe
- Certaines images incompl  tes



Description des donn  es

Le jeu de donn  es de Quick, Draw! est un sous-  chantillon du jeu de donn  es de Google AI contenant plus d'un milliard d'images. Voici quelques caract  ristiques du jeu de donn  es utilis   pour l'entra  nement du mod  le :

- Jeux de donn  es bruts et simplifi  s disponibles (le jeu de donn  es simplifi     limine des points inutiles qui se trouvent sur une m  me droite).
- **340 classes**.
- **   150 000 images/classe**.
- Plus de **51 millions d'images**
- **24,4 Gb** de donn  es compress  es en format .csv.

Utilisation des donn  es compress  es pour le projet pour limiter les enjeux de m  moire autant que possible sous hypoth  se que les performances du mod  le n'en souffriraient pas.

Performances

Mod��le	Performance en test
<i>Resnet18</i>	123
<i>Resnet18</i> avec p��nalit��	123
Ensemble avec classif	142
Ensemble avec moyenne	63

texte de support, gain de x

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**.