
Collecte de Données et Data Visualization

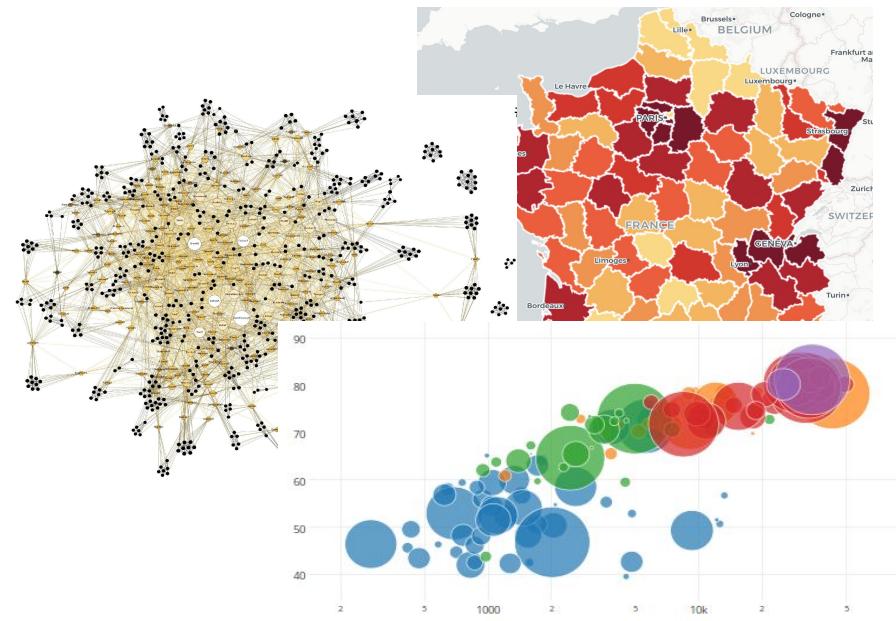
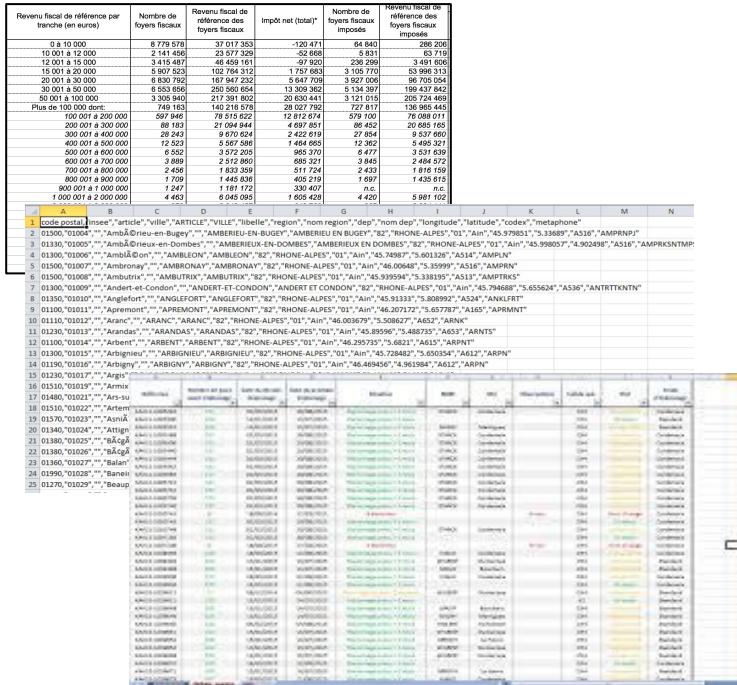
*Reporting de mi-projet
Avancement*

Plan

- Contexte & Objectifs
- Etat de l'art
- Les Réseaux de Neurones Convolutionnels
- Construction du dataset
- Utilisation de TensorFlow et entraînement
- Premiers Résultats
- Prochaines étapes

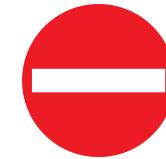
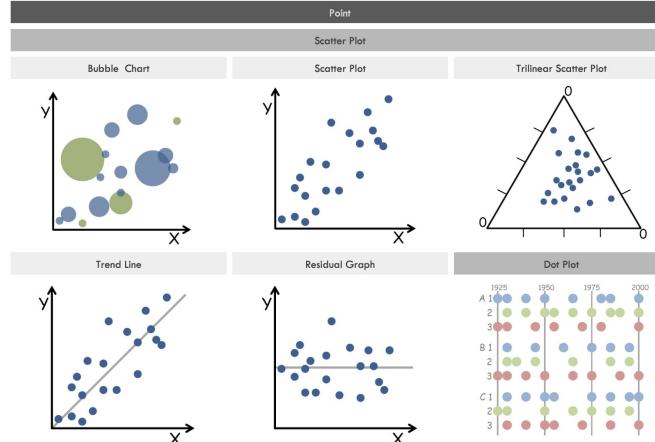
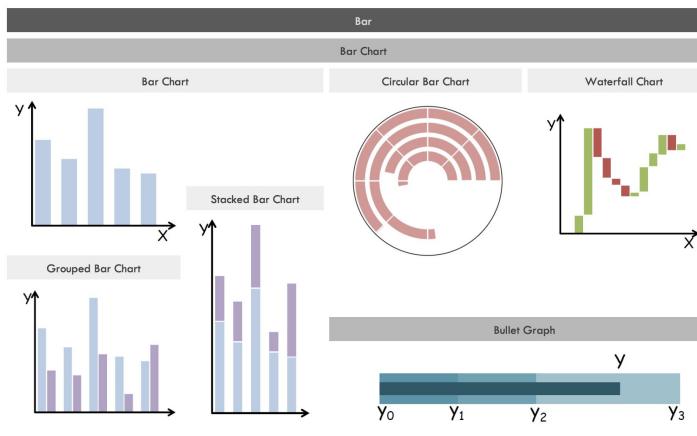
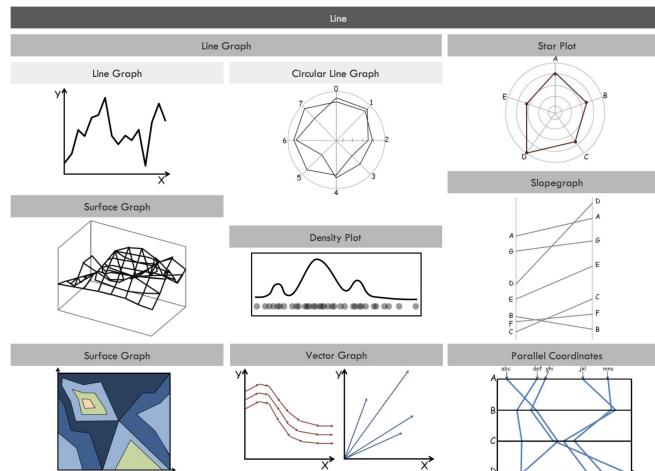
Contexte & Objectifs

Data Visualization : représentation visuelle des données pour une meilleure compréhension



Objectif : labéliser et trier automatiquement les visuels de data-visualisation et construire une base de données complète de visuels

Contexte & Objectifs



- Pas de labélisation / nomination arrêtée
- Ressemblance de forme
- Variété au sein d'une même classification

Etat de l'art des algorithmes de classification

Différents types d'algorithmes de machine learning :

Classification vs Régression

Supervisé vs non supervisé

Multiclasse vs binomial

Paramétrique vs non paramétrique

Linéaire vs non linéaire

Géométrique vs probabiliste

Etat de l'art des algorithmes de classification

Différents types de classification

Classification naïve bayésienne : efficace et simple mais ne fonctionne que dans des conditions précises (indépendance proba conditionnelles)

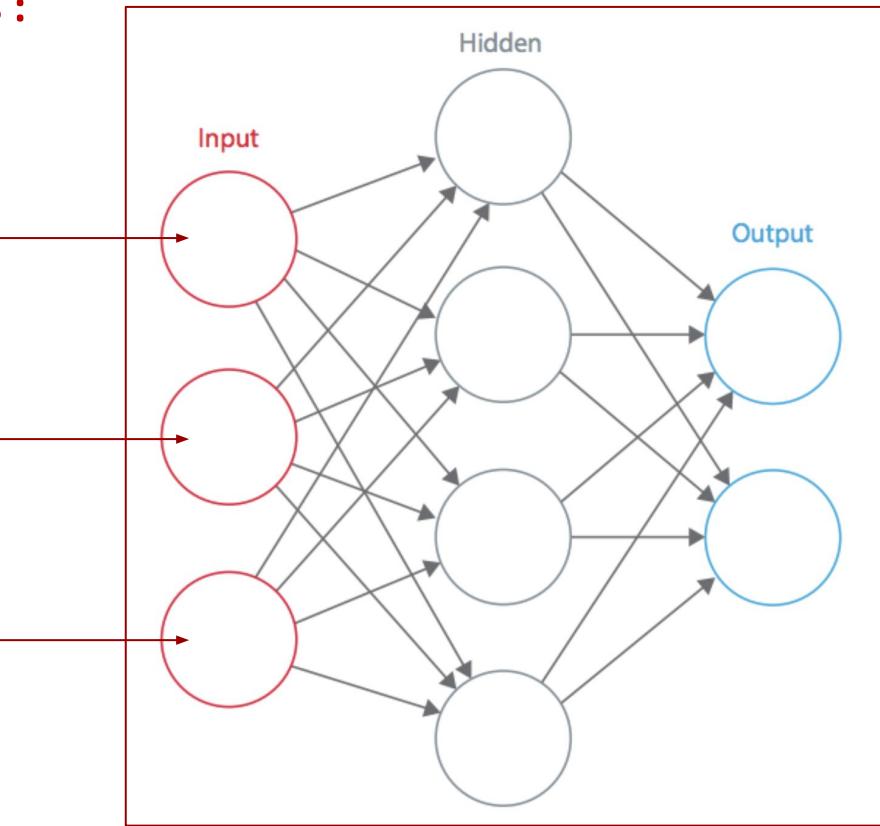
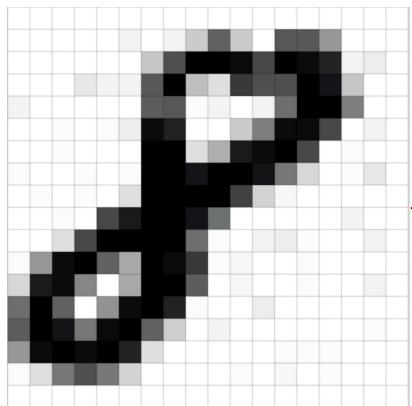
Machine à vecteurs supports : Fonctionne sur beaucoup de problème mais difficulté au niveau de la performance et du passage à l'échelle.

Arbre de décision/Forêts aléatoire : Fonctionne sur beaucoup de problème avec une bonne performance mais risque de sur-apprentissage et de déséquilibre (premier noeuds de l'arbre + décisifs que les suivants)

Réseau de neurones : Fonctionne de manière efficace sur beaucoup de problème mais choix de la structure complexe et parfois peu intelligible

Les Réseaux de Neurones Convolutionnels

Fonctionnement d'un réseau de neurones :



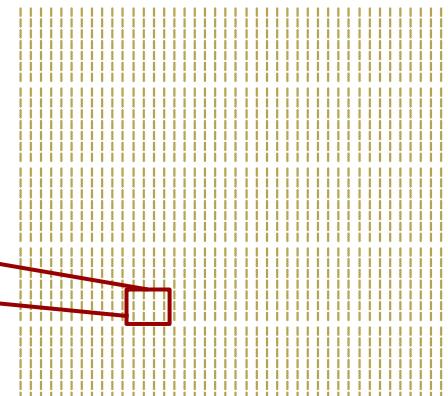
- chaque objet dans l'image est considéré différemment
 - Apprend indépendamment à reconnaître des formes

Les Réseaux de Neurones Convolutionnels

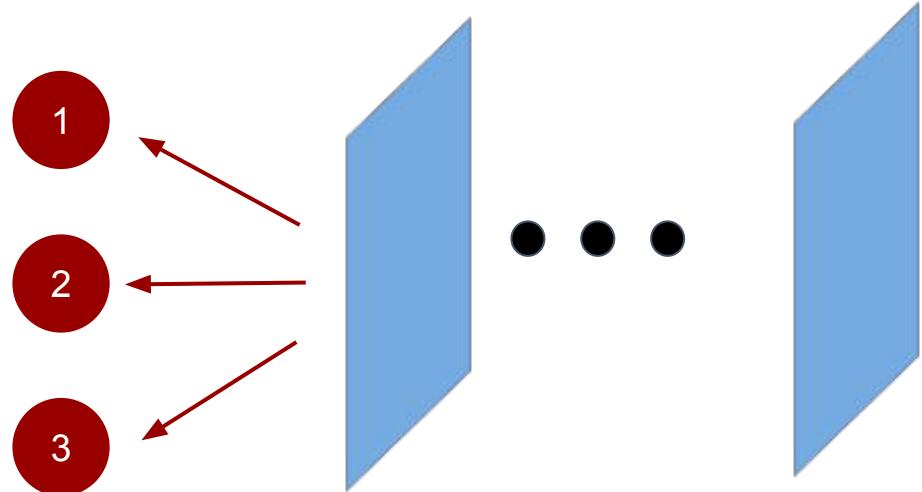
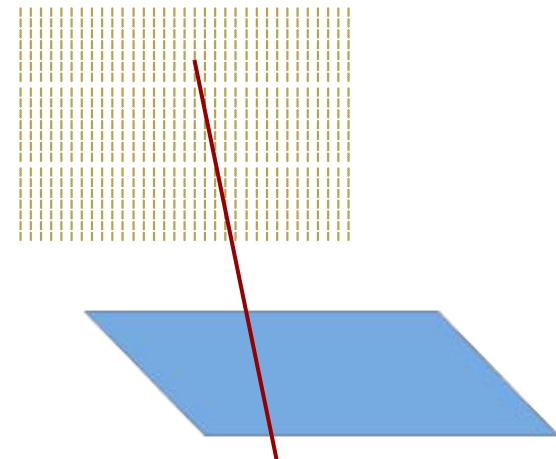
Partitionnement de l'image en tuile avec chevauchement



Passage de chaque tuile dans un réseau de neurones



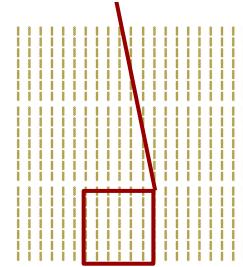
Sous-échantillonnage



Sous-échantillonnage



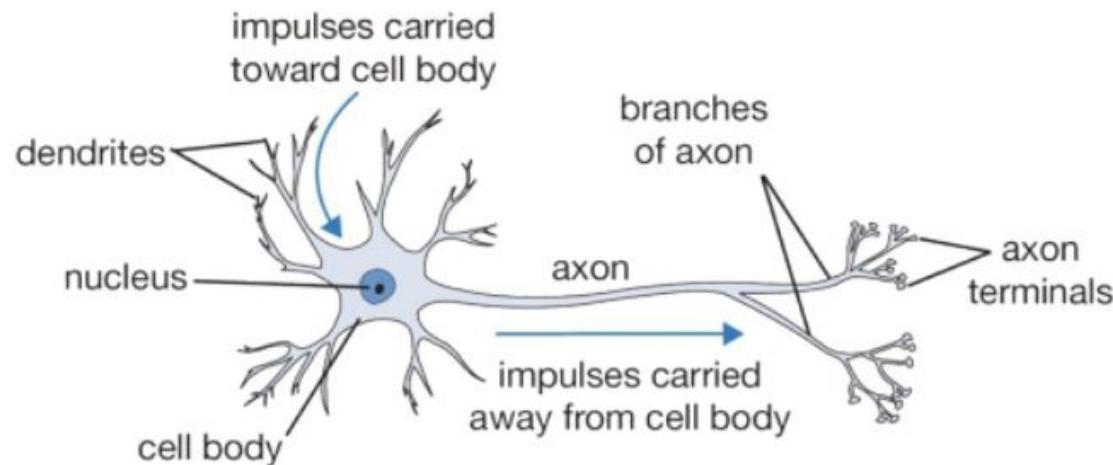
Passage de chaque tuile dans un réseau de neurones



Les Réseaux de Neurones Convolutionnels

Avantages

- Demande moins de mémoire
- Demande moins de puissance de calcul
- Invariants sur les décalage d'image car tous les pixels ont le même poids dans le réseau neuronal
- Les réseaux de neurones sont adaptés via des boucles retour ce qui permet d'optimiser le résultat



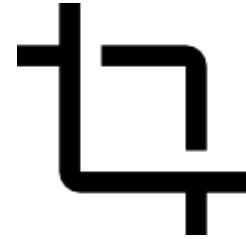
Construction de la base de données exploitable

Scraping



Python
Google Images

Formattege



Compléton des
images

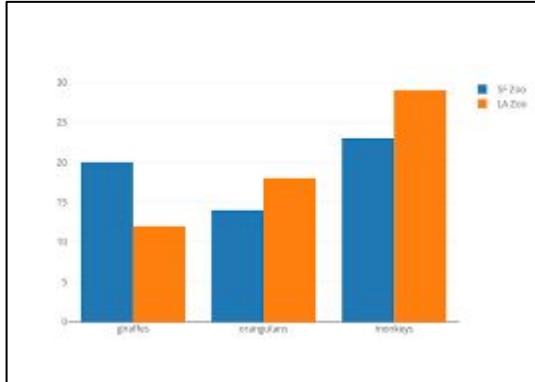
Sélection



Type de fichier
Contraste important

Résultat: 135 images exploitable

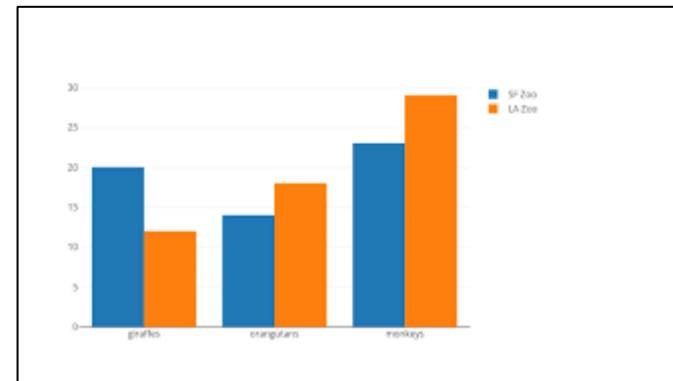
Construction de la base de données exploitable



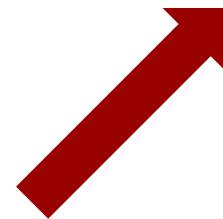
Exemple de transformation
pour un format 64 x 36



Mise au ratio



Changement
d'échelle



Construction de la base de données exploitables

Caractéristiques de la BDD :

- 135 images en format JPG, de trois catégories différentes : Line Chart, Bar Chart et Scatter Plot.
- Divisées en deux aléatoirement : 120 images pour l'entraînement - cross-validation du modèle et 15 images pour le test du modèle
- Nommage des fichiers images : nom_class_id.jpg

Constitution et sérialisation de nos 2 tableaux d'images (entraînement, validation) associés à 2 tableaux pour les classes respectives des images à partir du script Python build_dataset.py :

```
#Create a list of resized image array for the training dataset
for addr in train_addrs:
    img = loadData.resize_image(addr,64,64)
    X.append(img)

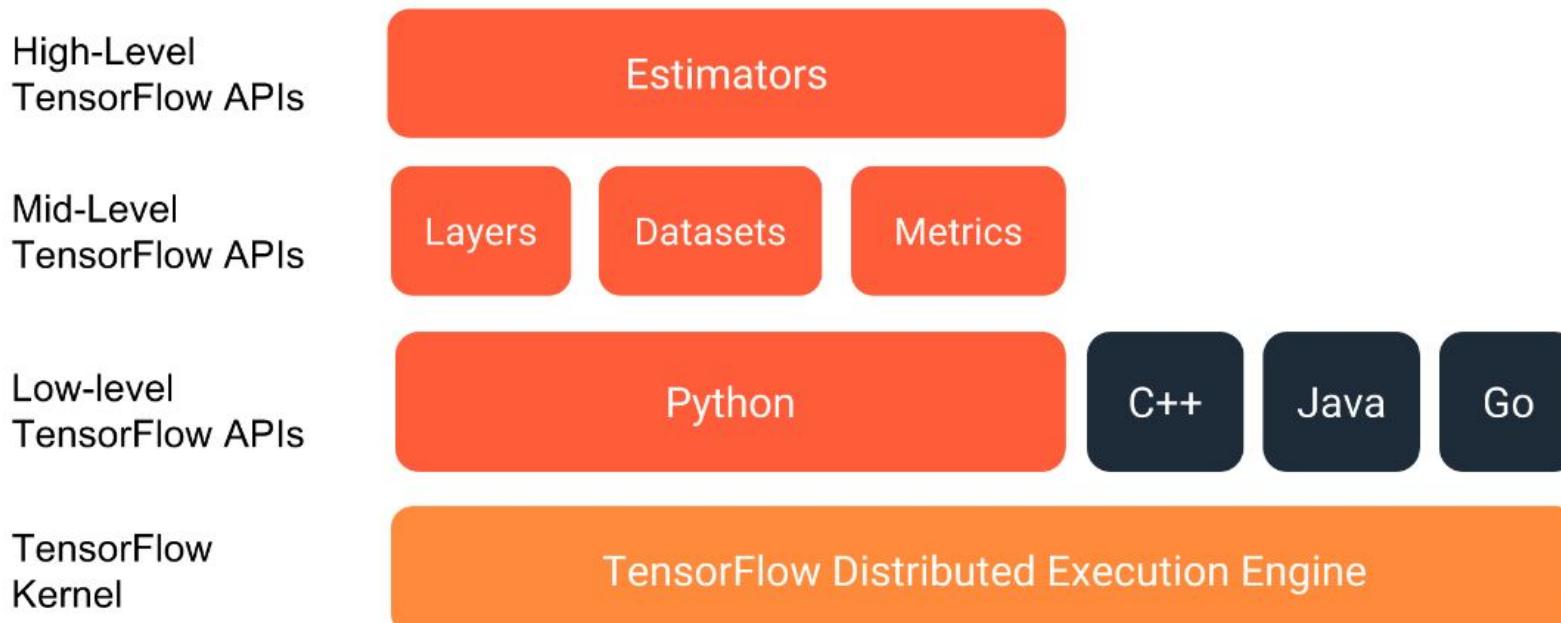
with open('dataset.pkl', 'wb') as f:
    #train_set, valid_set with images array and label array
    pickle.dump((X_train,Y_train,X_val_resized,Y_val_resized),f)
```

→ Fichier dataset.pkl prêt à être utilisé pour entraîner notre modèle de réseaux de neurones.

Présentation de Tensor Flow

Framework open-source de Machine Learning

- Développé par Google et publié en novembre 2015
- Dispose de plusieurs APIs de plus en plus complexes permettant de faire évoluer notre algorithme
- Forte communauté et Framework reconnu (~ 90000 étoiles sur Github)



Entraînement du réseau de neurones

Pour bien assimiler Tensor Flow, premier test avec un modèle de CNN inspiré d'un article scientifique :

Taille des images en entrée 32x32x3, structure : conv / max-pooling / conv / conv / max-pooling / fully-connected / drop-out / fully-connected

```
# Wrap the network in a model object
model = tflearn.DNN(network, tensorboard_verbose=0, checkpoint_path='dataviz-classifier.tfl.ckpt')

# Train it! We'll do 100 training passes and monitor it as it goes.
model.fit(X, Y, n_epoch=80, shuffle=True, validation_set=(X_val, Y_val),
           show_metric=True, batch_size=1,
           snapshot_epoch=True,
           run_id='dataviz-classifier')
```

Premiers résultats

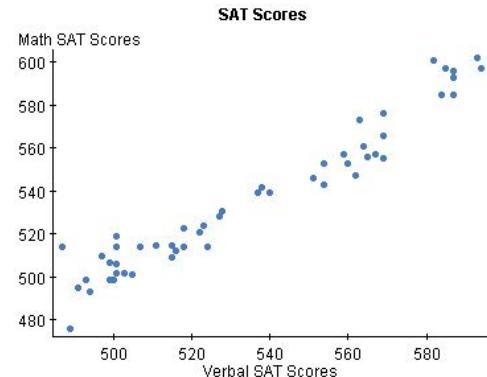
On sauvegarde la dernière itération de notre modèle afin de pouvoir utiliser notre classificateur sur le dossier test de 15 images dont les classes sont à déterminer

Résultat :

```
The confusion matrix is :  
[[ 5.  0.  0.]  
 [ 0.  5.  0.]  
 [ 4.  0.  1.]]
```

Analyse : 11 images sur 15 bien classées

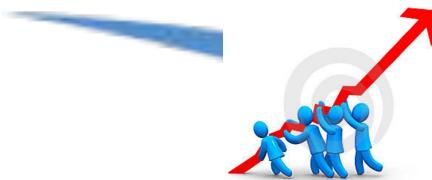
L'algorithme a du mal à reconnaître un Scatter Plot et lui assigne quasi systématiquement un line chart



Amélioration

- Résoudre le problème de distinction scatterplot / linechart
- Modification de l'algorithme pour de meilleurs résultats (nombre de couches, modification de l'ensemble d'apprentissage)

Prochaines étapes



Extension à une plus grande base de données

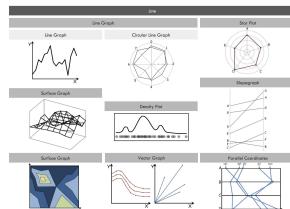
Objectif

25 Février



Extension à un plus grand nombre de catégories

02 Mars



Construire une base de données de visuels de classification

20 Mars



Rendre le jeu de données public

26 Mars

Des questions ?
