National Data Science Bowl

Florent Buisson

One net to rule them all

4 mai 2015

1. La compétition
   - Kaggle
   - Le plancton
   - Objectif
   - Les données
   - L'évaluation
2. Mon parcours
   - Plan initial
   - Analyse SWOT
   - C'est parti !
   - Training
   - Essais
   - Bilan

Kaggle est un système d'échanges de bons procédés autour du machine learning par le biais de compétitions sponsorisées :

- Accès aux meilleurs spécialistes pour les détenteurs de données,
- Accès à des données réelles pour les académiques,
- Accès aux deux pour les étudiants cherchant autre chose que les données MNIST.

+ Communauté active motivée par des problématiques réelles et gratifiantes (médical, environnemental) et potentiellement lucratives.

Le plancton

Il en existe deux types :

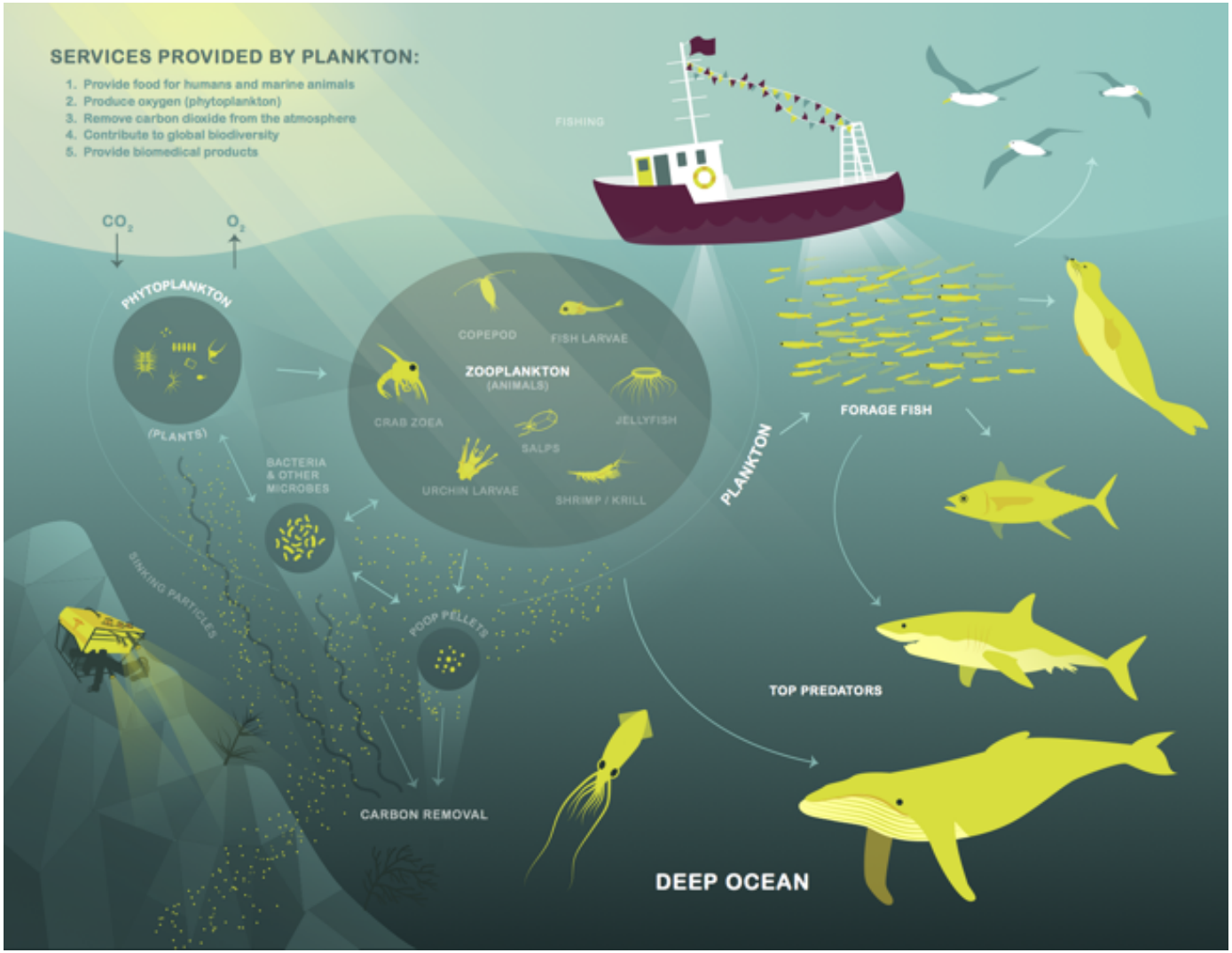
- Le phytoplancton (plantes)
- Le zooplancton (animaux)

Principaux services rendus :

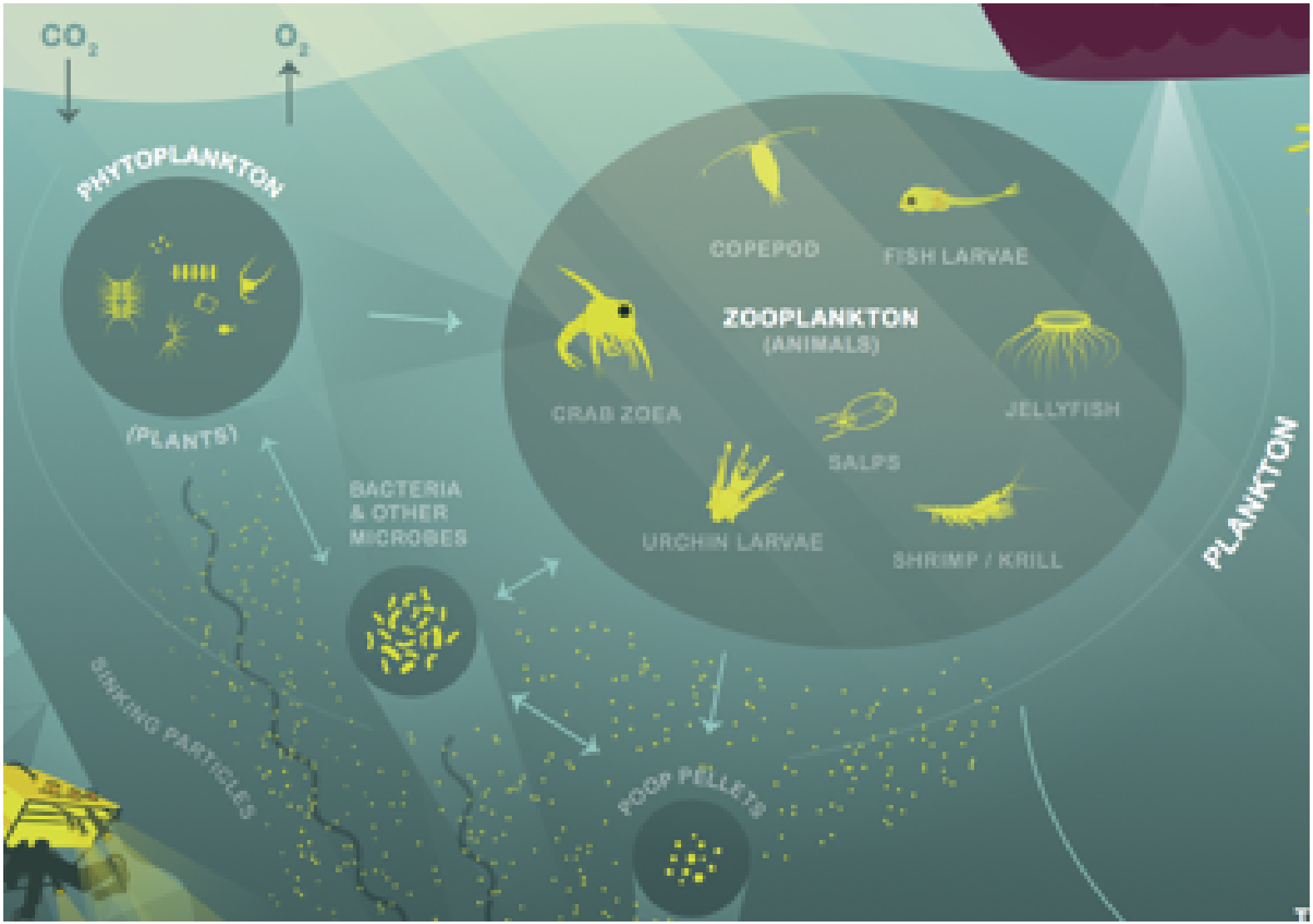
- Nourriture pour les humains (crevettes) et les animaux marins (baleines, coquillages filtreurs)
- Production d'oxygène (approximativement 50%)
- Fixation du carbone de l'atmosphère (approximativement 50%)

$\rightarrow$ Indicateur de la santé de l'écosystème océanique

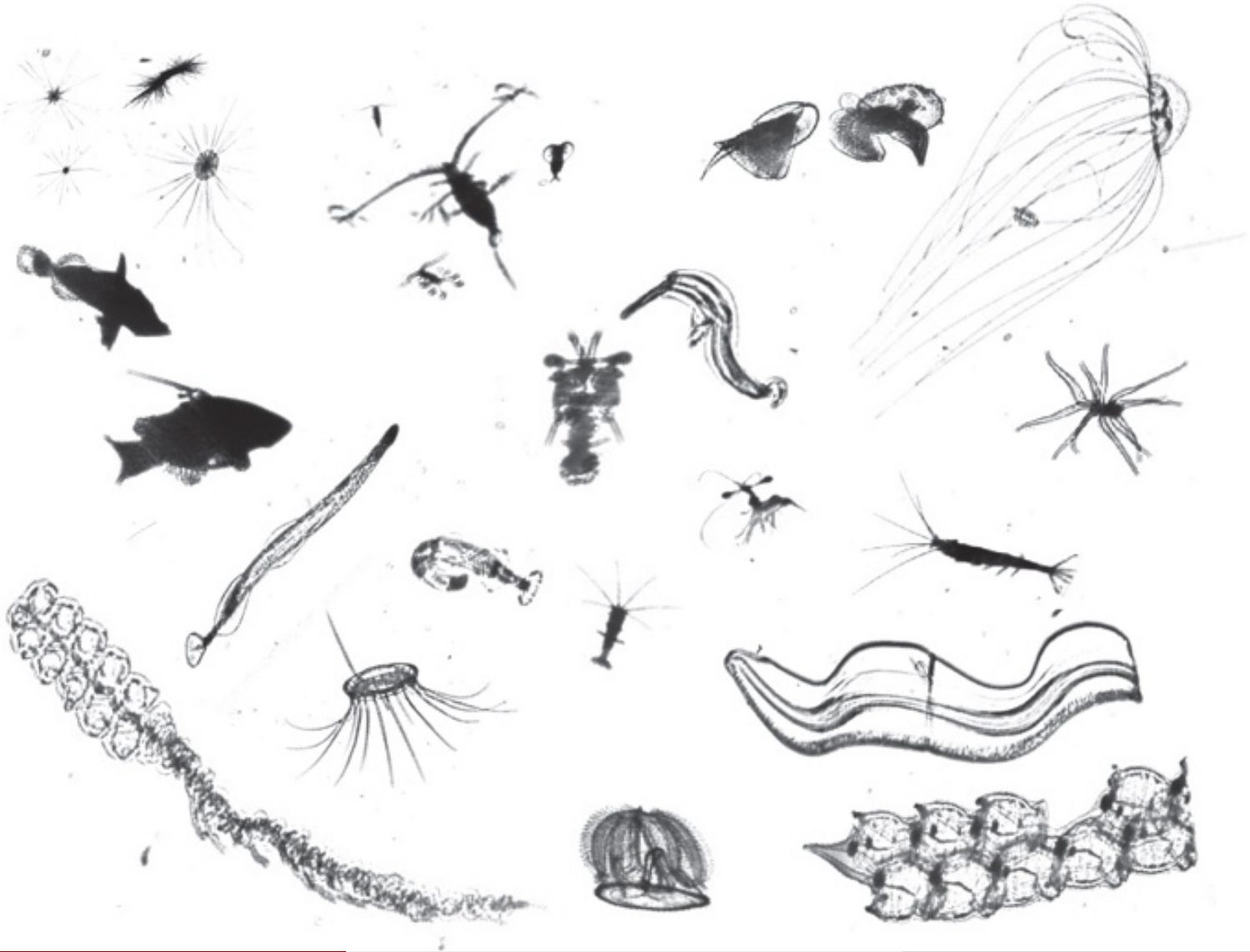
Objectif : État des lieux des populations de plancton



Objectif : État des lieux des populations de plancton



Les données



Les données

Collecte d'images sous-marines à haute résolution pendant 18 jours :

- 50 millions d'images de planctons (segmentation automatique)
- $>$ 80 TB de données
- Beaucoup d'espèces différentes + détritus
- Dans toutes les positions 3D
- Mise au point à l'infini, zoom constant (respect de l'échelle)

$\rightarrow$ Problématique idéale pour du machine learning.

Données fournies par le Hatfield Marine Science Center : 30K images labellisées manuellement en 121 classes pour l'apprentissage.

L'évaluation

Pour participer on soumet un fichier csv de probabilités permettant d'évaluer sur serveur un score de performance :

- N = 140000 images
- M = 121 classes
- p_{ij} : 121 probabilités de classe j par image i
- (sur serveur) y_{ij} : 1 si i la classe de l'image i est j, 0 sinon
- ≪Multi-class logarithmic loss≫

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} ln(p_{ij})$$

- Pour appliquer ln on remplace p par

$$max(min(p, 1 - 10^{-}15), 10^{-}15)$$

Plan initial et mes objectifs

- Phase 1 : Collecter les données
- Phase 2 : ?
- Phase 3 : PROFIT

Mes objectifs :

- Avoir un score pas trop ridicule (déjà engagé à vous présenter)
- Profiter de l'arrêt de mon contrat pour enfin toucher à du ML
- Ne pas complètement arrêter le boulot pour ne pas perdre la main
- Tester de nouvelles choses : GPU, Python, SIMD, etc

Analyse SWOT

- Forces
  - Plein de temps libre à y consacrer
  - Bon pc, CUDA un atout
  - Pas peur d'essayer n'importe quoi, de mettre les mains dans le code
- Faiblesses
  - Démarrage en retard, sur un nouvel environnement de travail
  - Départ de 0 : deep learning pour la reconnaissance d'images
- Opportunités
  - Les bibliothèques existantes (mais qui ont l'air un peu lourdes)
  - Communauté + forum très actif
- Menaces
  - Adversaires formés, avec du matériel, attirés par un prix de 100000$

Analyse SWOT

- Forces
  - Plein de temps libre à y consacrer
  - Bon pc, CUDA un atout
  - Pas peur d'essayer n'importe quoi, de mettre les mains dans le code
- Faiblesses
  - Démarrage en retard, sur un nouvel environnement de travail
  - Départ de 0 : deep learning pour la reconnaissance d'images
- Opportunités
  - Les bibliothèques existantes (mais qui ont l'air un peu lourdes)
  - Communauté + forum très actif
- Menaces
  - Adversaires formés, avec du matériel, attirés par un prix de 100000$
  - La procrastination me guette

6 mars : c'est parti !

- Après avoir lu le forum :
  - 6 tutoriels à essayer
  - Une vingtaine de pages à lire (rmb, semi-supervisé)
- Préliminaires :
  - Lasagne, Theanorc pour cuda tutoriel de dnouri pour un kaggle précédent (reconnaissance de visages)
  - CXXNET & OpenCV (+CUDA & BLAS) pour un score prétendu de 1,38 $\rightarrow$ fait en une aprèm, optimisé pour un score de 1,20 (jeu avec les paramètres)
- CNN (convolutional neural networks) sont très bien adaptés à ce challenge. Choix d'une lib
  - Convnet : aucune idée de sur quel réseau partir
  - Convnet v2 : ne fonctionne pas sur ma gtx 560 ti
  - Cxxnet : tutoriel très bien, SIMD, gpu, retenue.

Moins d'outils sur Cxxnet que sur convnet.

Déroulement d'un essai de prédiction avec cxxnet : initial

- Redimensionner les images (train et test)
- Générer des fichiers textes listant les images et leur classe (0-120)
- Créer à partir de cette liste et des images un fichier binaire
- Ajuster la configuration (cf exemple fichier de configuration)
- Lancer l'apprentissage
- Lancer la prédiction avec le dernier modèle
- Créer un fichier résultat formaté pour le concours
- Le soumettre sur le site et se faire noter (5/jour)

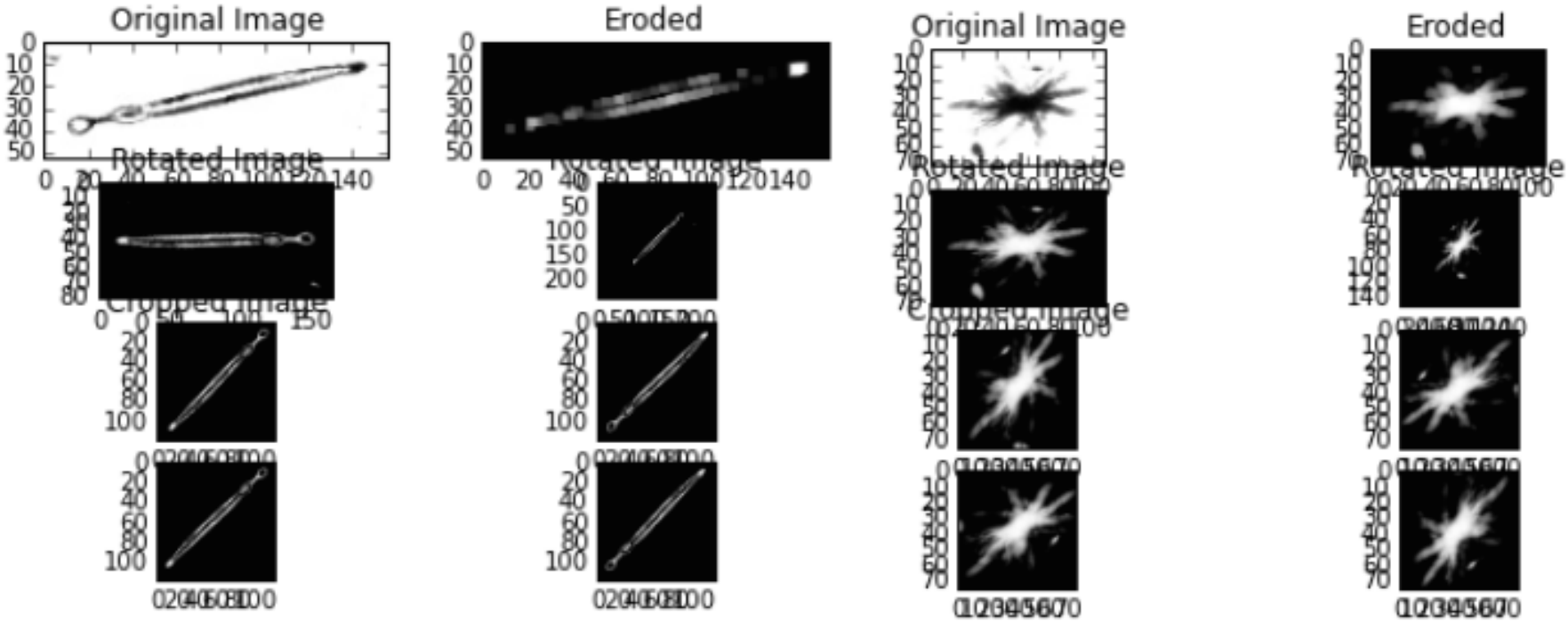
- Beaucoup de paramètres, de layers, donc de choses à tester
- Essais très longs

Premiers essais : environ 2 semaines

Après quelques itérations pour jouer avec les paramètres, j'arrive à un score de 1,20. Les itérations sont lentes, chaque apprentissage met plusieurs heures (jusqu'à 10h) et les pré-traitements en python sont longs

- Première fois : réseau trop gros pour mon gpu donc je diminue la taille du réseau
- Jeu avec les paramètres : learning rate, momentum, j'arrive à 1,12
- Observation des données : on m'a prémâché le travail!
- Nettoyage et augmentation des données : Résultats -0,10 à 1,02
- Beaucoup de planctons longilignes, maximisation de la place occupée : recentrage et alignement sur la verticale

Premiers essais : environ 2 semaines



Version 1, 20 jours restants

Industrialisation du code pour itérer plus vite :

- Gestionnaire de versions : git (fichiers de conf, scripts)
- Scripts de génération du workflow complet en IPython
- Pré traitement pour essayer de diminuer la dimentionalité du problème en alignant les planctons
- Objectif : ne pas gâcher les capacités d'apprentissage du réseau
- Première version alignée sur la diagonale succès auparavant : 1,02
- Mais antennes coupées précédemment, recentrage sans couper d'informations
- Augmentation des données précédentes 600k images, trop long pour les itérations, ramenées à 300k

Version 1, 20 jours restants

Conclusions :

- Score 1,17 mais certaines images encore non vues
- Le pré traitement semble meilleur
- Toutes les images devraient être vues lors de l'apprentissage
- Plus d'augmentation des classes sous-représentées

Version 2

À faire :

- Plus petit set d'apprentissage, 100k serait mieux
- Plus d'augmentation des classes sous-représentées

Conclusions :

- Apprentissage beaucoup plus rapide : 500 epoch en 5h au lieu de 200 en 10
- Mais malgré 500 itérations, score moins bon : 1,215
- Meilleure précision sur les classes sous-représentées qui étaient à 0 auparavant
- Bis 1 : +200 itérations 1,215 $\rightarrow$ 1,214
- Bis 2 : Essai d'apprentissage uniquement sur les classes à mauvais score, mauvaise idée

Version 3

À faire :

- Conservation des set d'apprentissage existants (pas de nouveau pré-traitement)
- Essai de remplacer les layers relu par des sigmoid
  $\rightarrow$ Échec, pas d'apprentissage, ou alors pas attendu assez longtemps.
- Essai avec la configuration du repo cxxnet pour ImageNet (énorme réseau)

Conclusions :

- Overfitting immédiat avec les 100k, on reprend les 600k
- Même avec l'énorme set d'apprentissage, overfitting patent

Version 4

À faire :

- Conservation des set d'apprentissage existants (par de nouveau pré-traitement)
  (très long et voyage le lendemain matin)
- Réduction de la taille du réseau pour réduire l'overfitting

Conclusions :

- 1,37 à l'itération 21 et augmente ensuite
- ≪Early stopping≫ : score 1,13 à l'itération 8

Version 5

À faire :

- Idem avec réseau encore plus petit

Conclusions :

- Avec arrêt prématuré, score de 1,0 environ

Moyenne de 4 scores obtenus à environ 1,0 : 0,82!!!!!!!! (+130 places)

Version 6 -7

- Version 6 : script plus complexe d'analyse des données prédites
- Version 7 : Plus gros set d'apprentissage et cross-validation avec données non présentées
- Nouveau prétraitement sans scaling des images.

Conclusions :

- Beaucoup d'overfitting

Version 8

- Nouveau prétraitement, sans alignement, avec rotations multiples (Je reviens sur la décision de réduction de dimentionalité)

Conclusions :

- Score 0,89 ! ! ! ! !
- Moyenne avec les précédents donne 0,74 +33 places
- 45 itérations supplémentaires car pas d'impression d'overfitting, mais pas de progrès (0,92)

Version 9

- Même process
- 100 itérations
- Hasard différent (random seed) : présentation des données à l'apprentissage dans un ordre différent

Conclusions :

- Overfit davantage

Version 10-16

- Scripts pour étude des candidats avec incertitudes, nettoyage du set de test.
- Itérations sur le prétraitement : petites images moins petites.

Conclusions :

- 10@87 : 0,97, 10@46 : 0,88, 8@17 : 0,84,
- 12@34 : 1,02, 12@22 : 0,92,
- REUSSITE : augmenter le set de test et moyenner les scores : 0,72 ! ! !
- Version 16 : Essais ratés d'optimisation du logloss à la main autre que moyenne : moyenne pondérée, moyenne tenant compte des f1-score sur le set d'apprentissage.

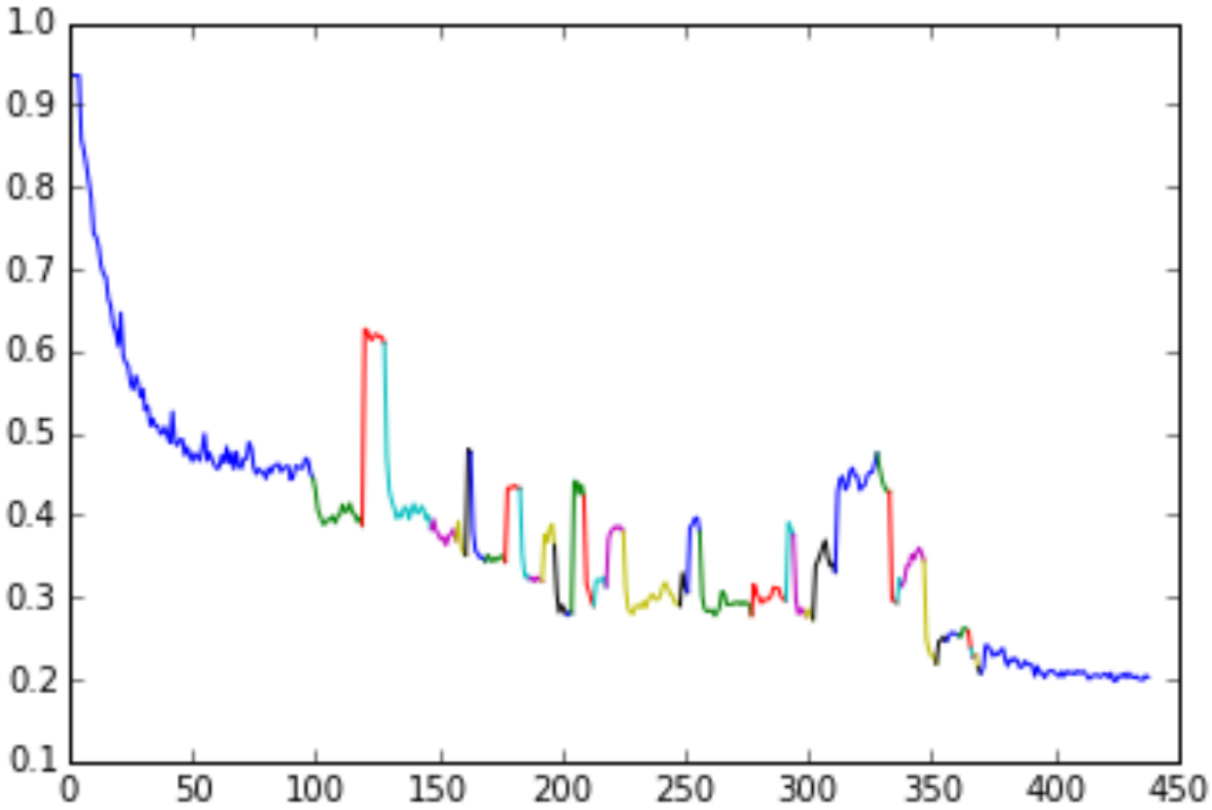
Version 17-18

- Pas de temps pour passer sur convent v2
- Séparation des sets d'apprentissage et cycles tournants pour essayer d'éviter l'overfitting : multisets
- 4 sous sets - 3 premiers * 4 avec mirroirs, apprentissage tournant

Conclusions :

- cf apprentissage
- Score 0,87, bien mais franchement moins bien que ce à quoi je m'attendait
- Plus de temps ni d'essais, en faisant la moyenne des meilleurs scores j'arrive à 0,71 (+10 places, c'est très serré), je remonte 65e pour retomber à 70 lorsque que tout le monde publie ses derniers résultats.

Version 17-18



Logloss sur le set d'apprentissage en fonction des epochs. Les changements de couleurs indiquent une rotation de set d'apprentissage.

Bilan

- Très content de ma place finale : 70e
- Je me suis laissé emporter par le démon de la compétition : manque de rigueur scientifique
- Cxxnet ne proposait pas suffisamment d'augmentation à la volée :
  - Seulement miroir et crop
  - rotation
  - crop min max
  - aspect ratio
  - shear
- J'ai modifié rapidement Cxxnet mais les parties gpu et simd trop drues pour que je comprenne sous pression du temps
- Pas eu le temps de parallèliser le python