Reconnaissance efficace d'objets sous-marins

Théophane VALLAEYS

TIPE 2020

Le problème étudié

Le problème

- Reconnaissance d’espèces marines et d’objets sous l’eau
- En temps réel :
  - Classification
  - Apprentissage de nouvelles données
  - Detection d’objets inconnus

Les aspects difficiles

- Complexité des fonds marins
- Diversité des espèces
- Peu de données pour certaines classes

Utilité pratique

- Sous-marins autonomes
- Etudes non-destructive des populations
- Faibles coûts

Données utilisée

Fish4Knowledge species (F4K)	Images depuis Google (G)
23 espèces 12 122 images à 25 par espèce Autour de 100 × 100 pixels	150 espèces 10 images par espèce 128 × 128 pixels

Sous-sets de données

Origine des données et nombre d'images par espèce :

3 espèces	6 espèces	23 espèces	173 espèces
F4K 3200 à 4000	F4K 148 à 4000	F4K 25 à 4000	F4K + G 10

Modèles légers

NumPy

DML

theano

CPU local

Modèles lourds

NumPy

Keras

TensorFlow

colab (GPU)

Classification rapide avec K plus proche voisins

Premières méthodes de classification

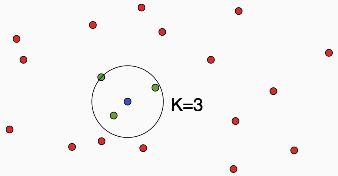
Distances euclidiennes

K plus proches voisins (KNN)

Dimension 30000
Dépend de la position de l'objet sur l'image

Densité de couleurs

Dimension 3
Indépendant de la position de l'objet



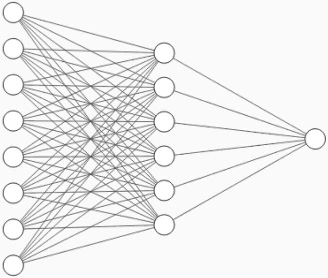
	3 espèces	6 espèces	23 espèces
KNN	91.6%	87.87%	38.84%
Densité	67.50%	39.73%	21.4%
Aléatoire	33.33%	16.67%	4.32%

Construction d’un réseau de neurones (ANN)

Ce qu’est réellement un ANN

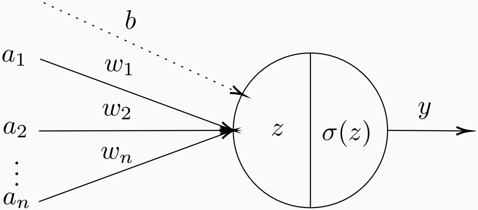
Un réseau de neurones est une manière de représenter certaines fonctions non-linéaires

$$\begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}$$



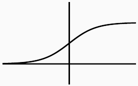
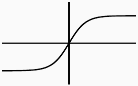
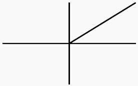
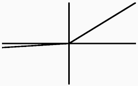
$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

Fonctionnement d’un neurone

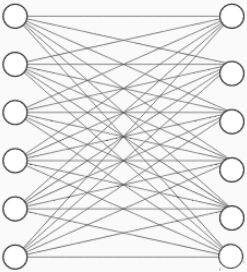


$$z = b + \sum_{i=1}^{n} a_i \times b_i$$

$$y = \sigma(z)$$

Sigmoïde	Tanh
	
$y = \frac{1}{1+e^{-z}}$	$y = tanh(z)$
ReLU	Weak ReLU
	
$y = max(0, z)$	$y = max(\epsilon \times z, z)$

Couche dense de neurones

A^l  Y^l

$$Z^l = W^l \times A^l + B^l$$

$$Y^l = \sigma(Z^l)$$

$$W = \begin{pmatrix} w_{1,1} & \cdots & w_{1,n} \\ w_{2,1} & \cdots & w_{2,n} \\ \vdots & \ddots & \vdots \\ w_{p,1} & \cdots & w_{p,n} \end{pmatrix} \quad A = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} \quad B = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_p \end{pmatrix}$$

W^l et B^l sont appris.

Fonction de coût L_2

$$C = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2$$

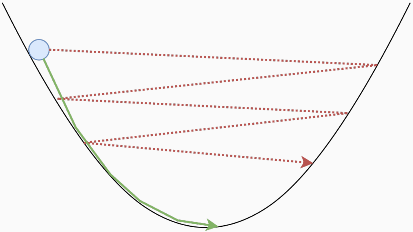
Avec :

- m : nombre d'images
- $\hat{y}$: sortie attendue

Descente de gradient

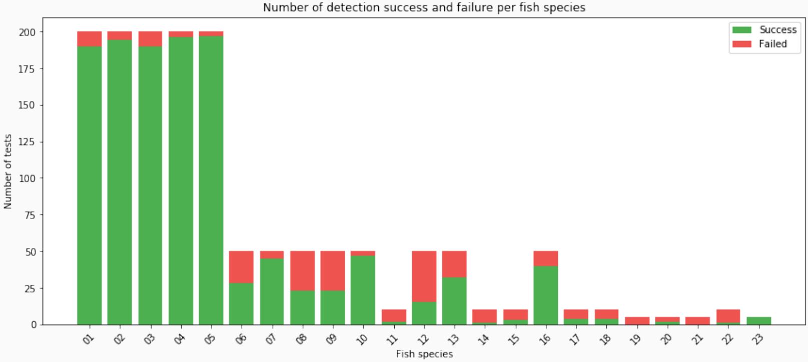
$$W' = W - \lambda\frac{\partial C}{\partial W}$$

$$B' = B - \lambda\frac{\partial C}{\partial B}$$



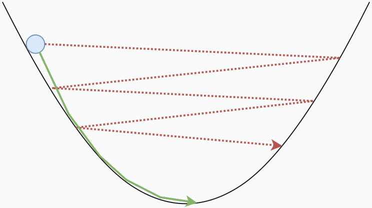
Résultats intermédiaires 1

Modèle	3 espèces	6 espèces	23 espèces
Aléatoire	33.33%	16.67%	4.32%
KNN	91.6%	87.87%	38.84%
ANN dense	96.83%	95.20%	83.58%



Ajouter un moment au gradient [FAUX !!!]

$$M_W = M_W - \lambda \frac{\partial C}{\partial W}$$
$$W = W + M_W$$



Couche softmax

Distribution de probabilité

$$y_j^L = \frac{e^{z_j^L}}{\sum_i e^{z_i^L}}$$

Fonction de coût log-likelihood

$$C = -\log(y_{i_0}^L)$$

On a $\hat{y}$ de la forme $[0, \ldots, 0, 1, 0, \ldots, 0]$, et $\hat{y}_{i_0} = 1$

Normalisation

Entrées

$$\mu = \frac{1}{n}\sum_{i=1}^{n} X^{(i)}$$

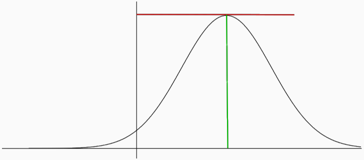
$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X^{(i)} - \mu)^2}$$

Avec le carré élément par élément (au sens du produit de Hadamard).

$$X' = \frac{X - \mu}{\sigma}$$

Activations

μ et σ calculés par batchs
α et β paramètres appris
$\gamma \in]0;1[$ fixe

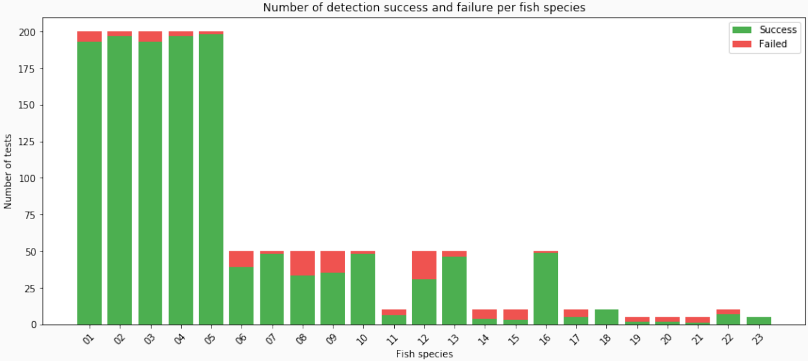


$$\hat{\mu} = \gamma\hat{\mu} + (1-\gamma)\hat{\mu} \quad \text{et} \quad \hat{\sigma} = \gamma\hat{\sigma} + (1-\gamma)\hat{\sigma}$$

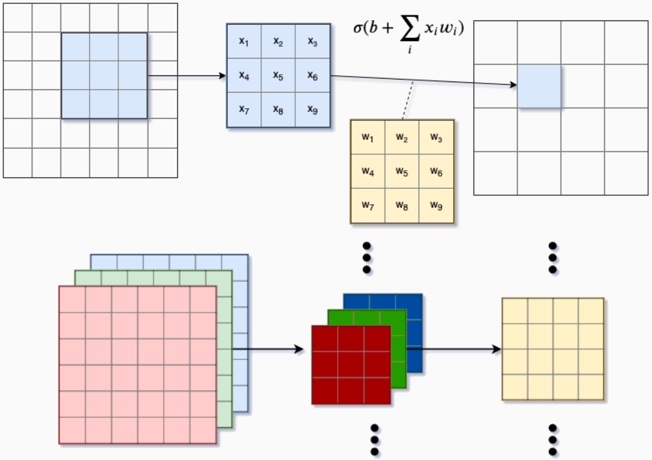
$$a_{\text{norm}} = \alpha\frac{a - \hat{\mu}}{\hat{\sigma}} + \beta$$

Résultats intermédiaires 2

Modèle	3 espèces	6 espèces	23 espèces
Aléatoire	33.33%	16.67%	4.32%
KNN	91.6%	87.87%	38.84%
ANN dense	96.83%	95.20%	83.58%
ANN dense amélioré	99.33%	95.60%	91.35%

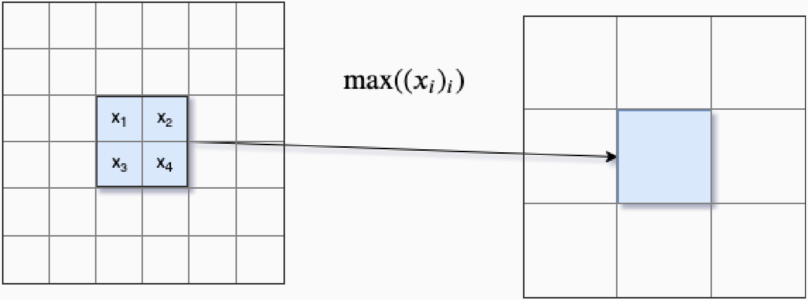


Nouveau type de couche, utilisé pour créer un ConvNet. La matrice des filtres d'une couche sera une matrice 4D.



Pooling

Couche de max-pooling, suit une couche de convolution.



Réduit la dimension spatiale et sélectionne l'information.

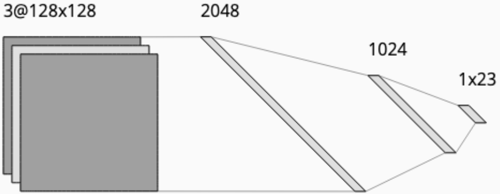
Résultats intermédiaires 3

Modèle	3 espèces	6 espèces	23 espèces
Aléatoire	33.33%	16.67%	4.32%
KNN	91.6%	87.87%	38.84%
ANN dense	96.83%	95.20%	83.58%
ANN dense amélioré	99.33%	95.60%	91.35%
ConvNet	99.50%	97.73%	96.96%

Architectures

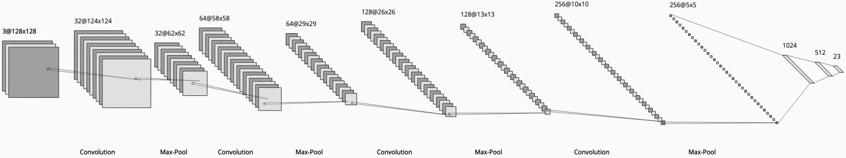
ANN dense

Paramètres : $102,787,095$



ConvNet

Paramètres : $7,801,117$



Complexités

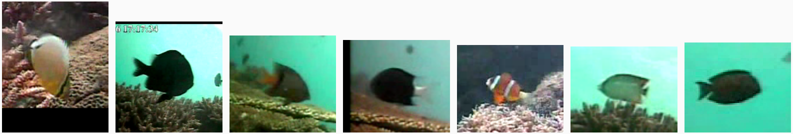
Pour une couche dense, n et m sont les tailles d'entrée et sortie

Pour une couche de convolution :

- $n \times n \times m$ est la taille de l'entrée
- k est la taille du filtre
- l est le nombre de canaux de sortie

Opéation	Complexité
Couche dense	$O(n \times m)$
Convolution	$O(lmk^2n^2)$
Autres	$O(n)$

Limites

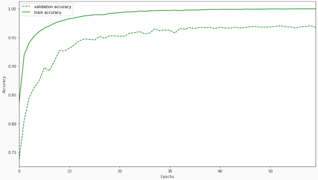


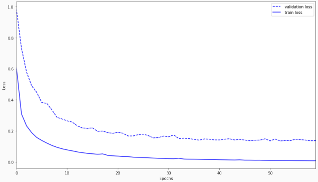
Nombre de classes

Temps d’entrainement :

Ajout de données

Ajout de classes





Transformation de l’image

Ce que l’on cherche

$$\begin{pmatrix} c_{1,1} & c_{1,2} & c_{1,3} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & c_{2,3} & \cdots & c_{2,n} \\ c_{3,1} & c_{3,2} & c_{3,3} & \cdots & c_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{n,1} & c_{n,2} & c_{n,3} & \cdots & c_{n,n} \end{pmatrix}$$

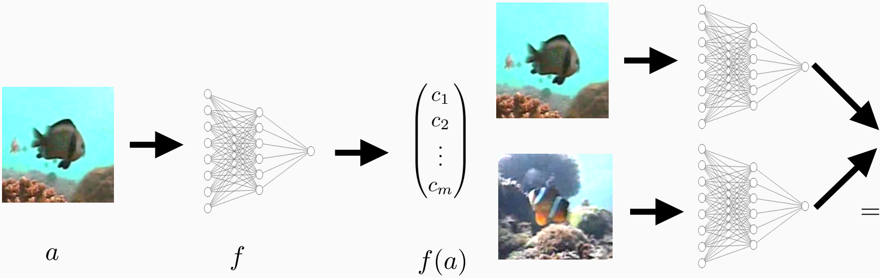
$$(R, V, B)$$

$$f : \mathbb{R}^{n \times n \times 3} \mapsto \mathbb{R}^m$$

$$\begin{pmatrix} Carac\ 1 \\ Carac\ 2 \\ \vdots \\ Carac\ m \end{pmatrix}$$

Invariance selon la
rotation, le fond, etc...

Réseau siamois



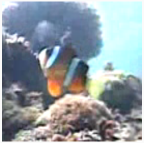
Fonction de coût avec triplets

a_a

$(ancre)$

a_p

$(positif)$



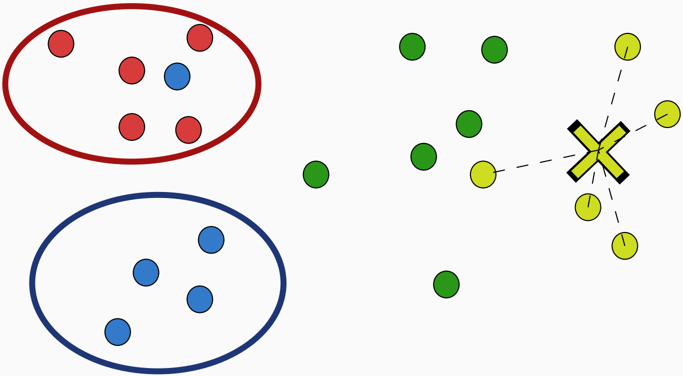
a_n

$(ngatif)$

On fixe la marge $m > 0$.

$$C = max(0, d(a_a, a_p) - d(a_a, a_n))$$

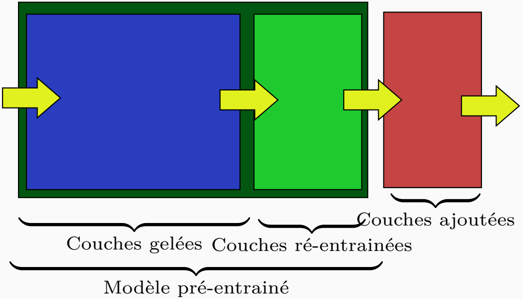
Sélectionner les triplets



Algorithme rapide

- Un représentant par classe
- Distance entre les représentants
- Finalement, les 10 plus éloignés par image

Transfert d'apprentissage



ConvNet

- Entrainé sur F4K
- Spécialisé
- Ré-entrainé

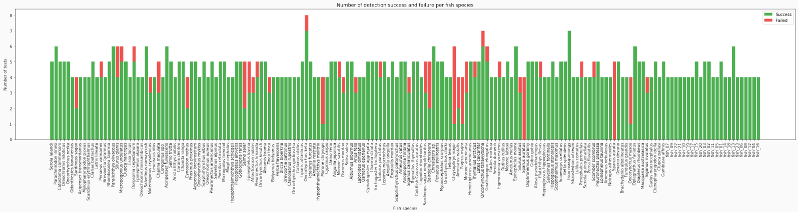
MobileNet V2

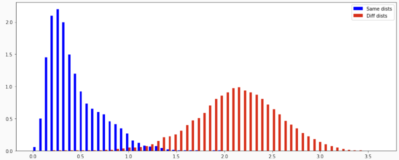
- Entrainé sur ImageNet
- Généraliste
- Utilisé tel quel

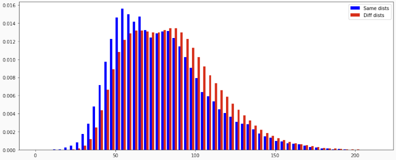
Résultats

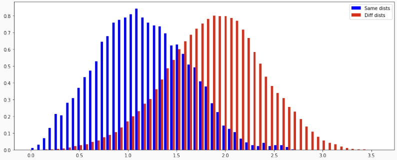
Modèle	173 espèces (entrainement)	23 espèces (test)
Aléatoire	0.58%	4.32%
KNN	8.76%	38.8%
MobileNet + Siamois	26%	67%
ConvNet + Siamois	91.85%	61.29%

Statistiques sur l’entrainement



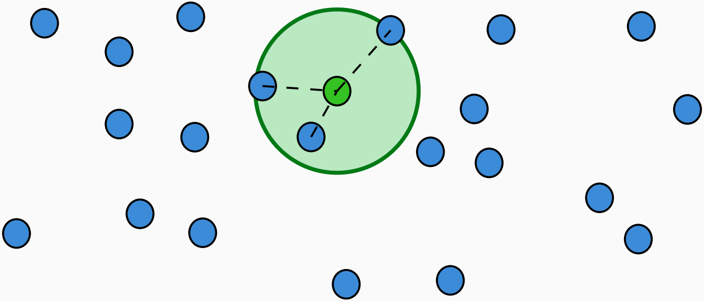






Classification rapide avec K plus proches voisins

Problème des k plus proches en dimension quelconque



Nombre de dimensions : $d \in \mathbb{N}*$ ($d \in \{1, 2, 32\}$) (??)
Nombre de points : $n \in \mathbb{N}*$ ($n \in \{10, 10^2, 10^3, 10^4\}$) (??)
Nombre de voisins : $k \in \mathbb{N}*$ ($k \in \{1, 2, 3, 5\}$) (??)

Approche par force brute

Idée

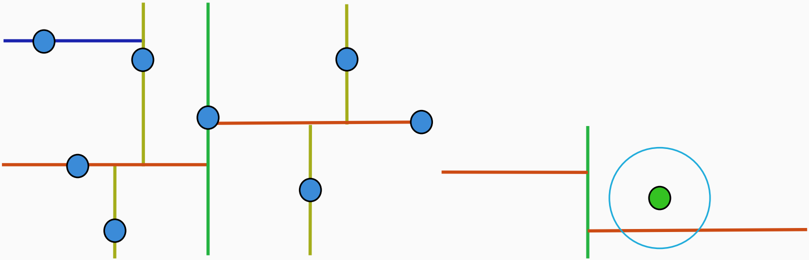
Parcourir tous les points en maintenant les plus proches avec un tas.

Somes stats ???

Complexité

$O(n)$ si $k = 1$, $O(n \ln(k))$ si $k > 1$

K-d tree ($k = 1$ uniquement)



Autres méthodes

TODO : stats de vitesse ET de % de réussite pour toutes les méthodes

Résultats (lesquels ?)

$$k = 1, d = 2$$

Algo	$n = 10^2$	$n = 10^3$
Force brute	100%	100%

Conclusion

Fin