Mars 2019

Stanford Dogs dataset

Projet OpenClassrooms

Par Xavier Montamat

Problématique

Réaliser un classificateur automatique d’images de chiens.

Développer et entraîner un algorithme capable de prédire la race d’un chien à partir d’une seule image.

Axes d’approche envisagés

Approche ‘classique’

- Pré traiter nos images
- Possible classification par couleur
- Utilisation de features SIFT

Approche réseau de neurones

- S’inspirer de modèle existants
- Entraîner le modèle sur un GPU
- Utiliser un modèle pré entraîné

Plan de réalisation

Pré traitement

- Découverte du dataset
- Cropping transformation

Approche classique

- Masquage de l'arrière plan
- Analyse des couleurs
- Résultats
- Features SIFT
- PCA & Kmeans
- Résultats

Réseau de neurones

- Augmentation des données
- Création d'un modèle Keras
- Résultats
- Modèles pré entraînés
- Résultats

Pré traitement

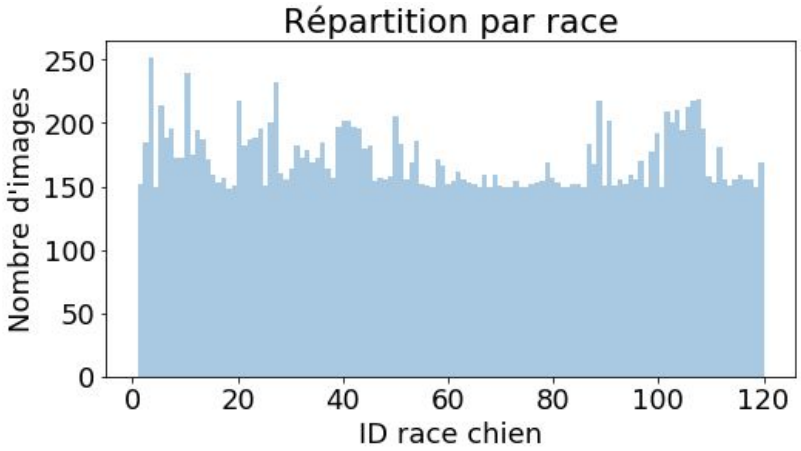
Avant de nous lancer dans la création de modèles, nous allons rapidement décrire notre dataset

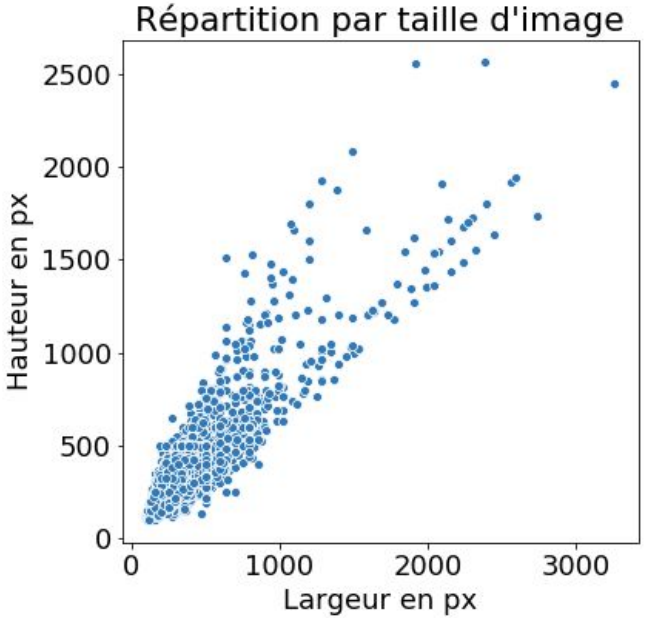
Puis exposer les transformations appliquées à nos images

Dataset d'images

20K images

- 120 races de chiens
- Répartition équilibrée





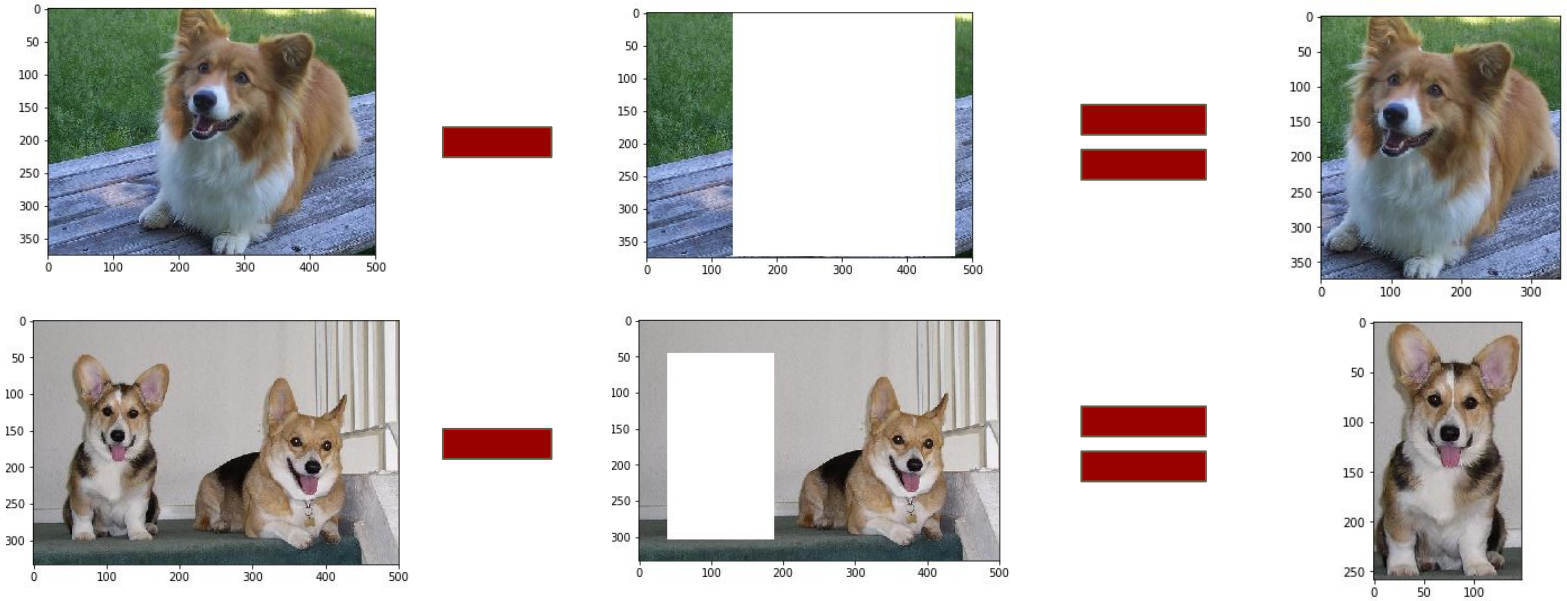
Tailles d'images

- Median 500*453
- Median 97*100

Transformations

Cropping

Abstraction du background (Méta données du dataset)



Approches classiques

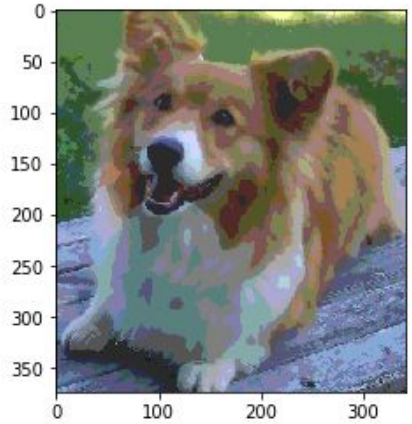
Créons un algorithme de classification à partir des méthodes ‘classiques’ de machine learning

Nous allons tester une classification basée sur les couleurs dominantes, et une sur les features SIFTS

Approche des couleurs dominantes

Déterminer les couleurs dominantes

Flouter l’image pour réduire les variations

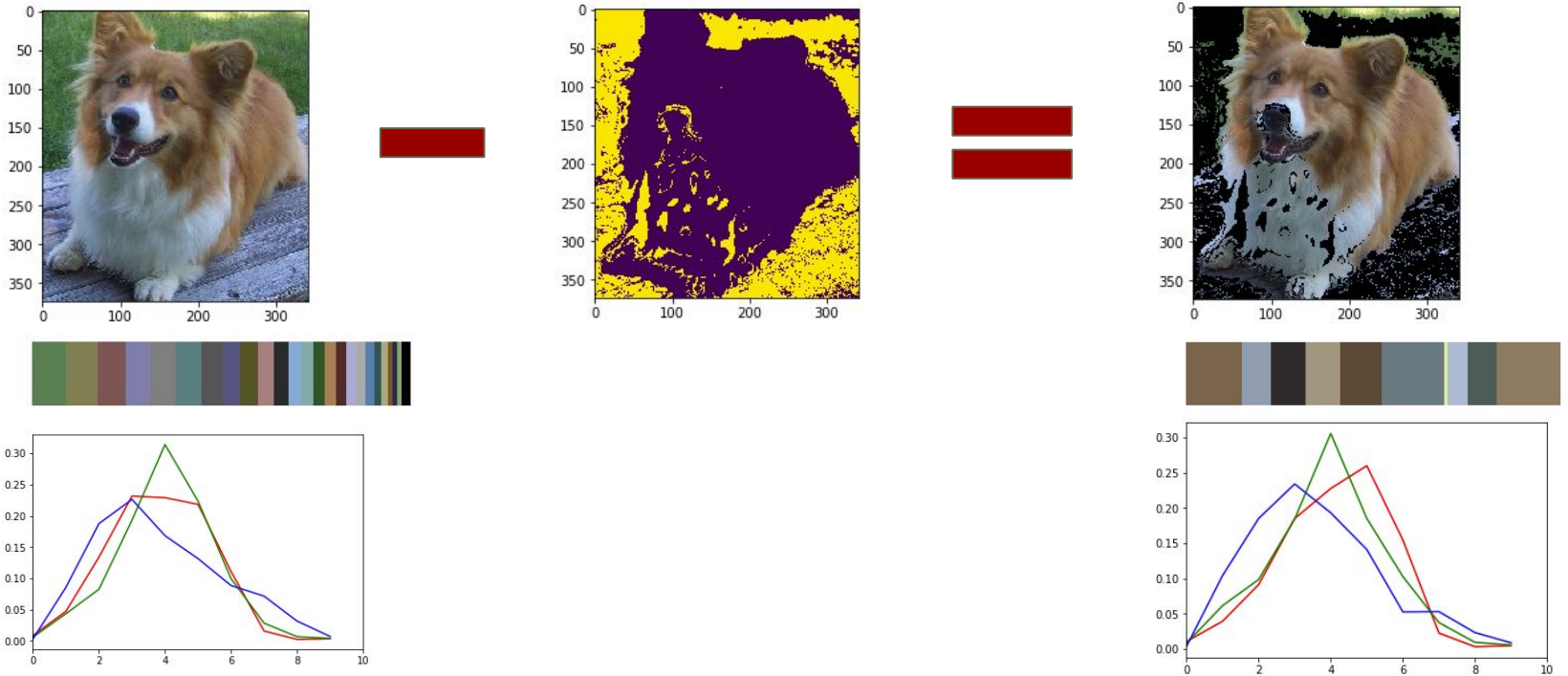


	colorR	colorG	colorB	0	color_percent
18	85	127	85	11770	9.202
25	127	127	85	10565	8.260
22	127	85	85	9353	7.312
27	127	127	170	8613	6.734
26	127	127	127	8521	6.662
19	85	127	127	8362	6.538

Couleurs dominantes fortement influencées par le background

Approche des couleurs dominantes

Masquage de l'arrière plan



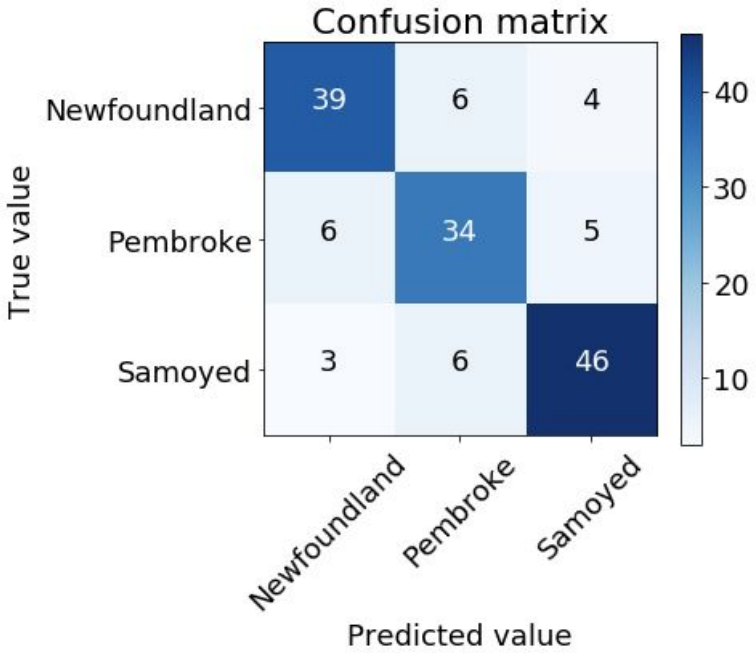
Classification & résultats

RandomForestClassifier

Sur **3** races - précision de 0.80

Sur **12** races - précision de 0.31

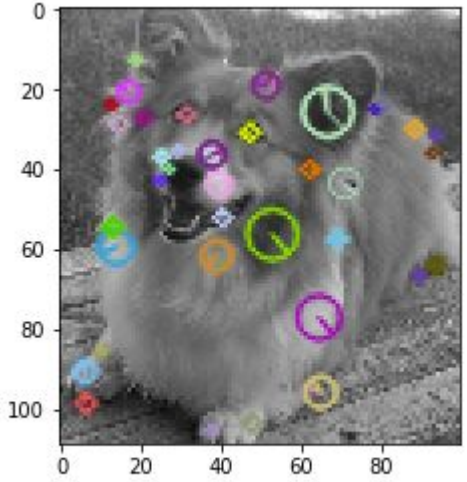
Sur **120** races - précision de 0.06



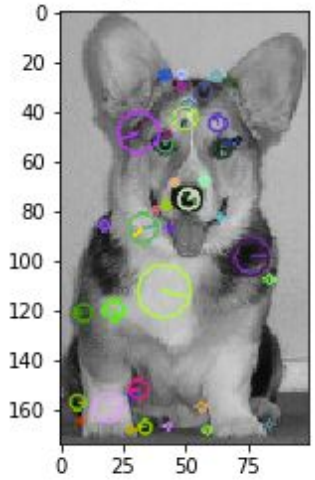
Approche des features SIFT

SIFT (Scale-Invariant Feature Transform)

Détection des features

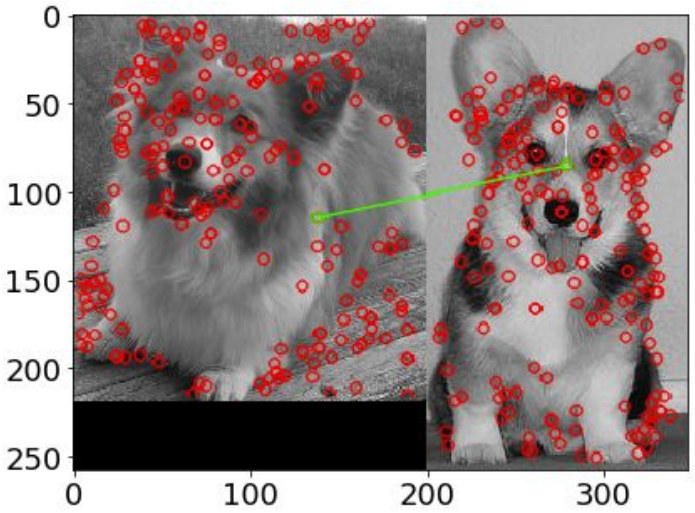


50 descriptors found



50 descriptors found

Matching direct peu probant



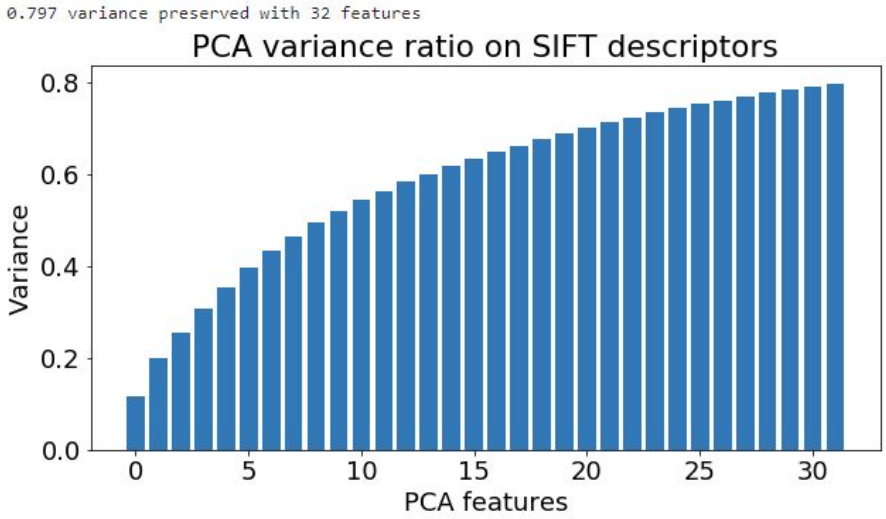
Approche des features SIFT

PCA reduction

Descripteurs 128 dimensions

PCA de 32 composants

80% variance préservée

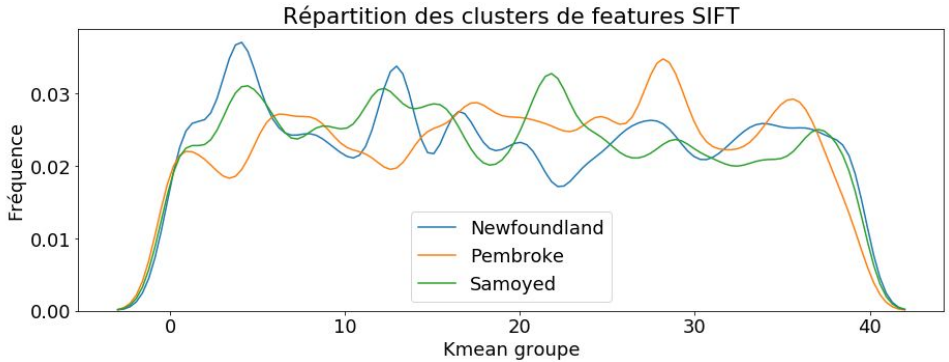


Approche des features SIFT

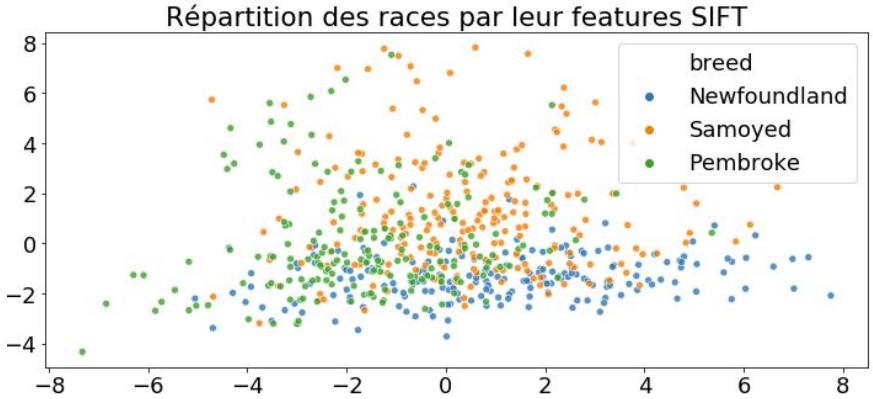
Classification des features SIFT

40 groupes Kmeans

Observation de leur répartition



Visualisation PCA 2D



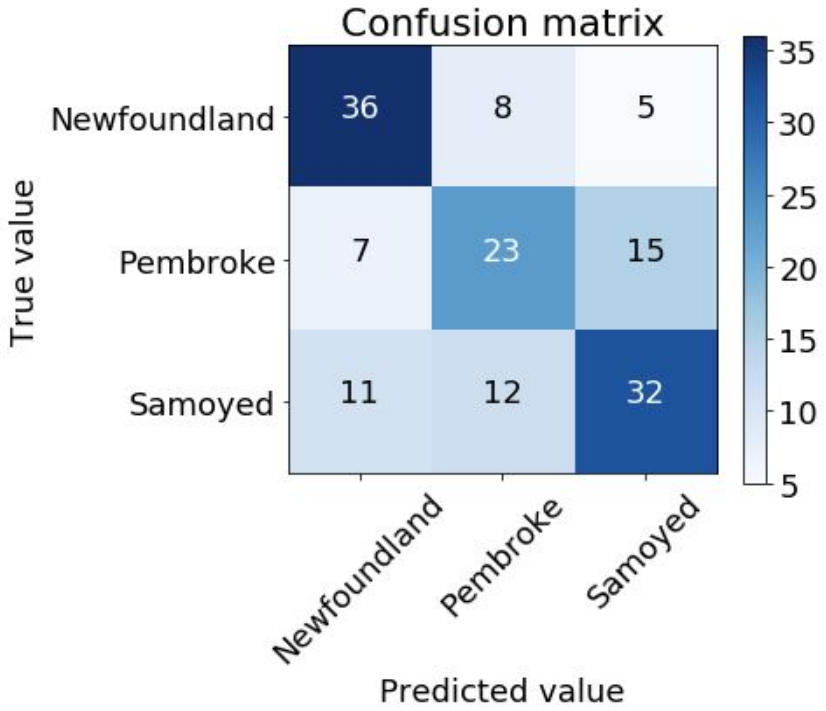
Classification & résultats

RandomForestClassifier

Sur **3** races - précision de 0.61

Sur **12** races - précision de 0.25

Sur **120** races - Trop long à calculer



Réseau de neurones

Malgré des performances honorables pour distinguer quelques races de chien, l'approche classique n'est pas efficace pour 120 races.

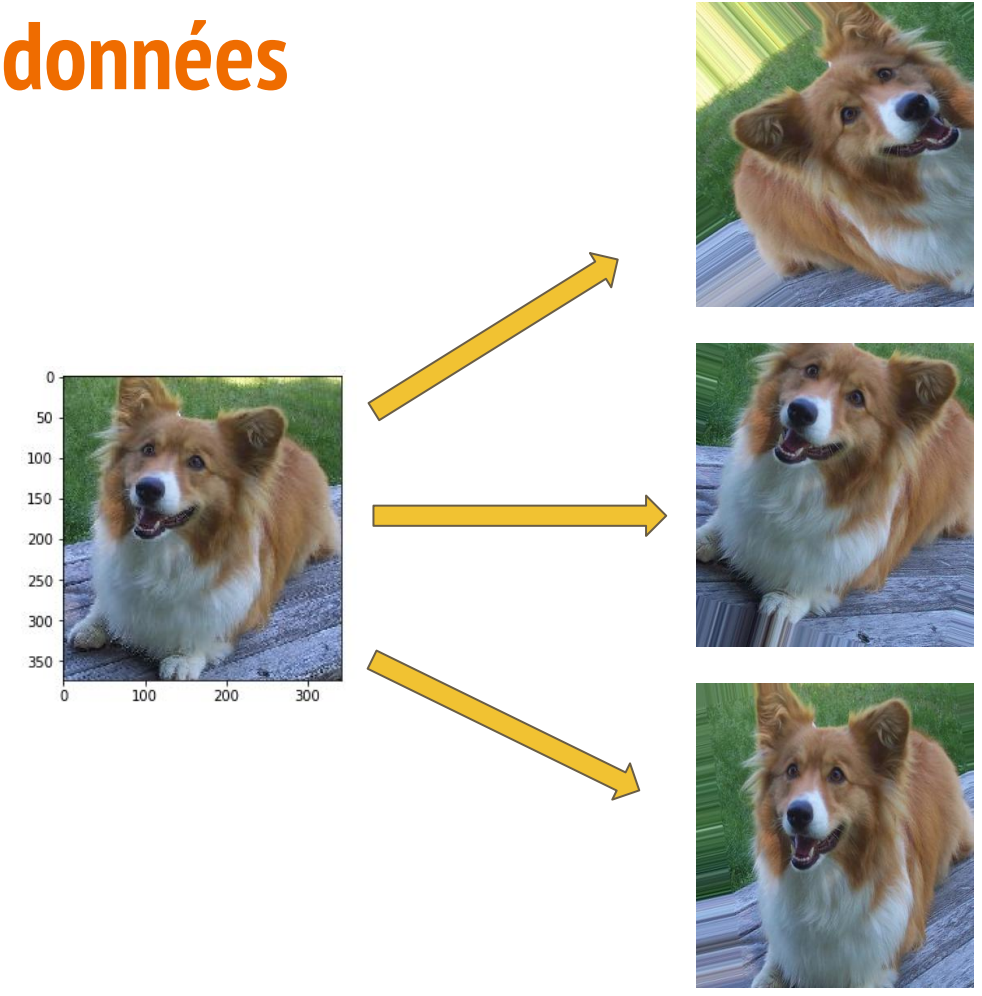
L'état de l'art de la classification d'images étant les réseaux de neurones, nous devrions théoriquement obtenir de meilleurs performances avec ces algorithmes

Augmentation des données

Pré processing aléatoire

- Redimensionnement
- Rotation
- Zoom
- Flip
- Shift

Augmenter données entrainement



Modèle CNN avec Keras

Construction du modèle CNN

- Successions de couches
  - Convolution
  - MaxPooling
  - Dense
- Inspiré de modèles existants
  - VGG

Layer (type)	Output Shape	Param #
conv2d_89 (Conv2D)	(None, 32, 224, 224)	896
max_pooling2d_81 (MaxPooling	(None, 32, 112, 112)	0
conv2d_90 (Conv2D)	(None, 32, 112, 112)	9248
max_pooling2d_82 (MaxPooling	(None, 32, 56, 56)	0
conv2d_91 (Conv2D)	(None, 64, 56, 56)	18496
max_pooling2d_83 (MaxPooling	(None, 64, 28, 28)	0
conv2d_92 (Conv2D)	(None, 64, 28, 28)	36928
max_pooling2d_84 (MaxPooling	(None, 64, 14, 14)	0
conv2d_93 (Conv2D)	(None, 128, 14, 14)	73856
max_pooling2d_85 (MaxPooling	(None, 128, 7, 7)	0
conv2d_94 (Conv2D)	(None, 128, 7, 7)	147584
max_pooling2d_86 (MaxPooling	(None, 128, 3, 3)	0
flatten_11 (Flatten)	(None, 1152)	0
dense_27 (Dense)	(None, 128)	147584
dense_28 (Dense)	(None, 12)	1548

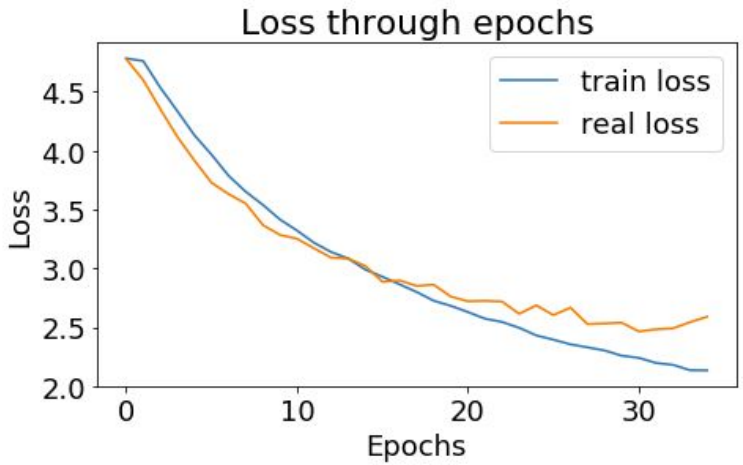
Total params: 436,140
Trainable params: 436,140
Non-trainable params: 0

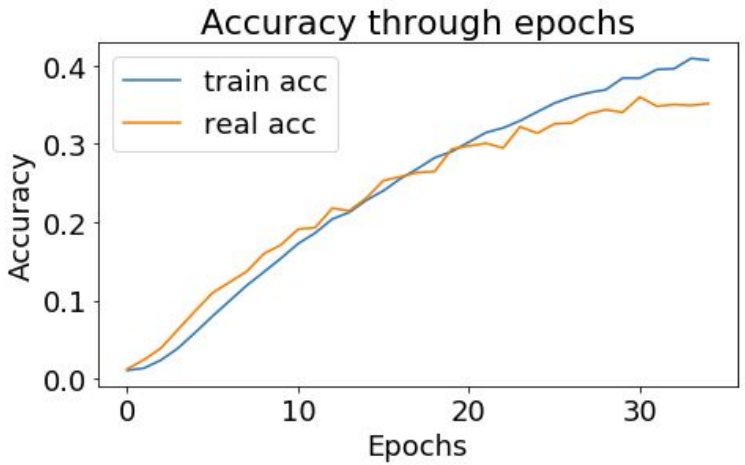
Entrainement du modèle

Séparation train/test (0.25)

Métriques de perte et précision

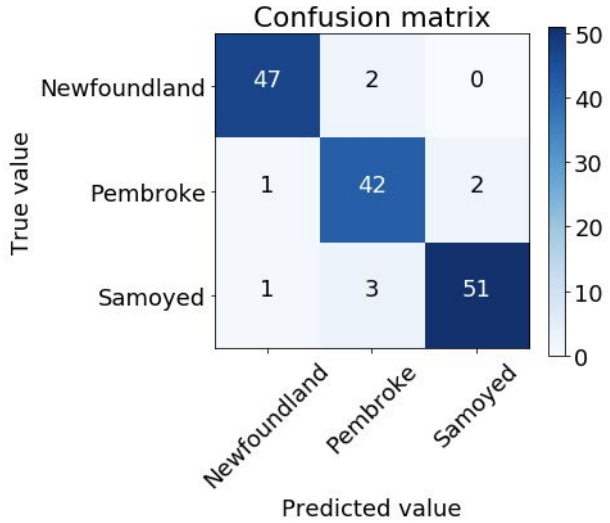
Epoques





Résultats

- Sur **3** races - précision de 0.93
- Sur **12** races - précision de 0.41
- Sur **120** races - précision de 0.35



Déjà de très bon résultats !

Modèles CNN pré entraînés

Choix du modèle

Parmis plusieurs modèles possibles

- VGG
- ResNet
- Xception

Remplacement des dernières couches

- GlobalAveragePooling
- Dropout
- Dense

Entraînement des dernières couches

Exemple des paramètres du modèle Xception

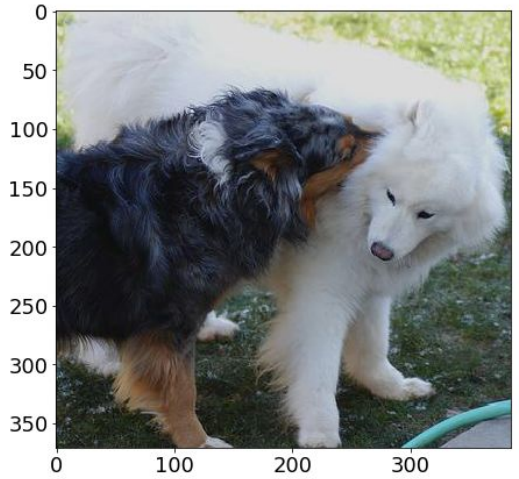
```
Total params: 21,107,360
Trainable params: 245,880
Non-trainable params: 20,861,480
```

Résultats des modèles pré entrainés

Sur 120 races

- ➔ **VGG16 (FastAI)**
  - ◆ Perte : 0.88
  - ◆ Précision : 0.68
- ➔ **Resnet34 (FastAI)**
  - ◆ Perte : 0.61
  - ◆ Précision : 0.75
- ➔ **Xception (Keras)**
  - ◆ Perte : 0.58
  - ◆ Précision : 0.86

Exemple d'échec



Conclusion

Les CNN pré entraînés permettent une précision surprenantes même sur 120 races !

Les résultats obtenus surpassent de très loin les approches ‘classiques’

Ils sont en outre relativement rapides à mettre en place

Améliorations possibles

Optimisation learning rate du CNN
Identifier races compliquées et trouver plus d'images
Réentraîner le modèle une fois en production

Questions

?