



Apprenant : Lepicaut
Objet : Technologies pour l'intelligence artificielle
Période : Centre de formation
Jury Académique : Philippe Besse . Beatrice Laurent-Bonneau . Brendan Guillouet
Intervenant : Brendan Guillouet
Date : 25/03/19

Mots clefs : TF.Keras, GPU, Deep Learning, Computer Vision, Accuracy, RMSE

CADRAGE	
date	20 février 2019
version	22 mars 2019
source des données	images 128 x128, Airbus GEO > prises de vue satellitaires d'éolienne
nombre de classes	k = 2
nombre d'enregistrements	54573 (training) + 13645 (validation) et 17057 (test)
périmètre	Apprentissage statistique > deep learning > computer vision
problème	Classifieur Multi-classes (binaire) sur des données images
type d'analyse	Méthode d'apprentissage machine supervisée
langage de programmation	Langage Python - TensorFlow - Keras
Note destinée au lecteur	résultats clefs



GPU Elastic Amazon EC2

- SOMMAIRE -

(1) Résumé	p. 3
(2) Problème	p. 4
(3) Méthodologie	p. 5
(4) Données	p. 6
(5) Outils	p. 7
(6) Modèle	p. 11
(7) Fine Tuning	p. 13
(8) Résultats	p. 14
(9) Transfert Learning	p. 15
(10) Annexes	p. 19

1. Résumé

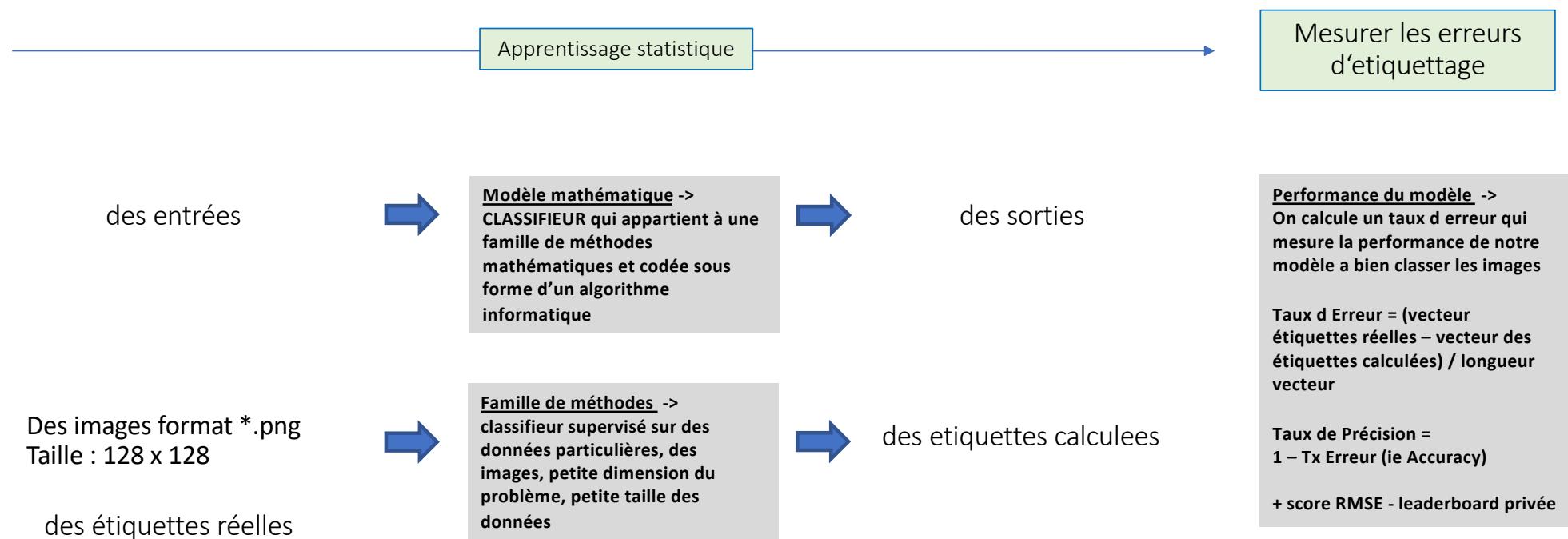
The screenshot shows a web browser window with the following details:

- Tab:** Défi 2019
- Address Bar:** Institut National des Sciences Appliquées de Toulouse [FR] | <https://defi-ia.insa-toulouse.fr/account>
- Toolbar:** applications, Apple, Importés depuis S..., Télécharger toute..., Matrice de corréla..., http://www.career..., https://chrisalbon..., Apprenez à utiliser..., Aperçu des métho...
- Header:** Accueil, FAQ, Compte, Déconnexion, Défi 2018
- Left Sidebar:** 37
- Center Content:** Feb. 19, 2019, 9:04 p.m.
- Right Content:** 97.27219, didier lepicaut
- Text Area (highlighted with a blue box):**

```
#=====
Executive summary :
=====
. Calculs avec Keras-GPU
. Archictecture réseau CNN classique (i.e. Cats and Dogs)
. Fine tuning
. Accuracy Train > Accuracy Validation
. Recherche du modèle qui généralise bien
. Tests avec plusieurs environnements technologiques
. Performance obtenue avec Centrage-Réduction des Entrées (Xi)
```

2. Problème

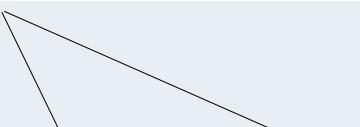
Demande: « ... entraîner et spécifier une machine qui produit des étiquettes ... »



DATASET LOCAL			PLATEFORME https://defi-ia.insa-toulouse.fr	
Training	Validation	Test	Test plateforme	
Entraîne les modèles	Optimise les Modèles	Arbitre les modèles avec avec le biais le plus petit ($Y - \hat{Y}$)	Leaderbord publique	Leaderboard privée
FORME	OPTIMISE	TESTE		
<i>on connaît les labels</i>		<i>on ne connaît pas les labels</i>	<i>on connaît ne pas les labels</i>	
Disponible dès le début			Disponible dès le début, utilisé pour afficher le classement durant la compétition mais peut mener à du sur-apprentissage	Dévoilé à la fin de la compétition et utilisé pour produire le classement final et définitif

Le modèle de fonctionnement d'une compétition en Data Sciences – Défi IA / Kaggle.com

4. Données

TRAIN		TEST	
	target	TOTAL	TOTAL
other	32966	68218	17057
35252			
51,68%	48,32%	100,00%	
			
TRAIN		VALIDATION	TOTAL
other	28201	7051	35252
target	26372	6594	32966
TOT	54573	13645	68218
Poids	80,00%	20,00%	100,00%

- > Type : image au format *.jpg, en couleur,
- > Taille des images : en entrées avec dimension unique 128 x 128 pixels,
- > Procédure apprentissage simple (train vs validation) = (80.00%) / (20.00 %) de la taille du fichier départ (i.e. Train fourni),
- > Qualité des données : ~ « 2% Fake image » et/ou « image de mauvaise qualité »,
- > « preprocessing du type augmentation des données »,
- > Pour garantir la convergence des calculs numériques, une normalisation « 1/255 » appliquée aux entrées (normalisation de la valeur de nos entrées 0-255 en 0-1)

5. Outils

5.1. Mono - GPU

5.1.1. Personal Computer

HARDWARE 1	
Processeur	Intel Core i7-8750 H@ 2.20GHz × 12
Carte graphique	NVIDIA GeForce GTX 1060 Max-Q 6Go mémoire
Mémoire	2×8 Go, DDR4, DDR4-2400 (jusqu'à 32 Go)
Stockage	SSD 512Go M.2 + HDD 1 To @ 5400 RPM)
SOFTWARE	
Ubuntu	Lts 18.04 Bionic Beaver
Anaconda	1.9.6
Jupyter Notebook	5.7.4
Nvidia	NVIDIA-SMI 396.54
	cudatoolkit 9.2
	cudnn 7.2.1
Python	version 3.6
Keras	version 2.2.4
Tensorflow-GPU	version 1.12.0

HARDWARE 2	
Processeur	Intel Core i7-7700K @ 4,2GHz × 4
Carte graphique	NVIDIA GeForce GTX 1080 - 8Go mémoire
Mémoire	2×8 Go, DDR4, DDR4-2400 (jusqu'à 32 Go)
Stockage	SSD 512Go M.2 + HDD 2 To @ 7200 RPM)
SOFTWARE	
Ubuntu	Lts 18.04 Bionic Beaver
Anaconda	1.9.6
Jupyter Notebook	5.7.4
Nvidia	NVIDIA-SMI 396.54
	cudatoolkit 9.2
	cudnn 7.2.1
Python	version 3.6
Keras	version 2.2.4
Tensorflow-GPU	version 1.12.0

5. Outils

5.1. Mono - GPU

5.1.1. Personal Computer

GPU	GTX 1080	GTX 1060
hardware	tour	portable
Mémoire	8Go	6Go
Temps (mn)	30.14	47.01
Temp Max (°)	56.00	65.00
SpeedUP [1060 > 1080] (mn)	35.89%	

A retenir:

- . Importance critique de la performance du hardware pour former et faire converger un réseau de neurones convolutionnel

Réflexe :

1. Je monte mon code sur le GPU de mon laptop
2. Je « run » mon code sur une machine performante (i.e. Cloud GPU as a service)

MODEL VERSION
DATA
% [TRAIN, VALIDATION]
TOPOLOGIE RESEAU
NORMALISATION χ_i
DATA AUGMENTATION
DATASET
FCT ACTIVATION Conv2D
FCT ACTIVATION Sortie
FCT PERTE
FCT OPTIMISEUR
METRIQUE
TX APPRENTISSAGE
FCT REGULARISATION
BATCH NORMALISATION
NOMBRE EPOCHS
TAILLE BATCH
TRAIN 2 / VALIDATION 2
Train accuracy
Validation accuracy
TEMPS EXECUTION (mn)
Score (RMSE)

5_Eol.ipynb

Image = 128 x 128 x 3
3 Conv2D + perceptron
128-3-3> 128-3-3 >256-3-3>512
1/255
shear_range / zoom_range / horizontal_flip
shear_range / zoom_range = 0.2
Train + validation + test
relu
sigmoid
cross_entropy
adam
accuracy
0.5
50 / 100
128
Train accuracy
Validation accuracy
0.9718 / 0.9837
0.9641 / 0.9618
47.01 / 58.74 [gtx1060]
96.83 / 96.82

5_Eol.ipynb

Image = 128 x 128 x 3
3 Conv2D + perceptron
128-3-3> 128-3-3 >256-3-3>512
1/255
shear_range / zoom_range / horizontal_flip
shear_range / zoom_range = 0.3
Train + validation + test
relu
sigmoid
cross_entropy
adam
accuracy
0.5
50
128
0.9737
0.9644
30.14 [gtx1080]
96.89

5. Outils

5.1. Mono - GPU

5.1.2. Computing As A Service



Google Cloud Platform

- Intel® optimized Deep Learning Image: TensorFlow 1.13.1 m23 (with Intel® MKL-DNN/MKL and CUDA 10.0)
A Debian based image with TensorFlow (With CUDA 10.0 and Intel® MKL-DNN, Intel® MKL) plus Intel® optimized NumPy, SciPy, and scikit-learn.
- Debian GNU/Linux 9 Stretch + TF 1-10
A Debian linux image with Tensorflow Version 1-10 pre-installed and optimized for Cloud TPUs.
- Debian GNU/Linux 9 Stretch + TF 1-11
A Debian linux image with Tensorflow Version 1-11 pre-installed and optimized for Cloud TPUs.
- Debian GNU/Linux 9 Stretch + TF 1-12
A Debian linux image with Tensorflow Version 1-12 pre-installed and optimized for Cloud TPUs.
- Debian GNU/Linux 9 Stretch + TF 1-13

Vous ne trouvez pas ce que vous recherchez ? Découvrez des centaines de solutions de VM dans [Marketplace](#)

Type de disque de démarrage

Taille (Go)

Disque persistant SSD

30

GPU Elastic Amazon EC2



Deep Learning AMI (Ubuntu) Version 22.0 -
ami-0174e69c12bae5410

Select

64-bit (x86)

MXNet-1.4, TensorFlow-1.13, PyTorch-1.0, Keras-2.2, Chainer-5.3, Caffe2-0.8, Theano-1.0 & CNTK-2.6, configured with NVIDIA CUDA, cuDNN, NCCL, Intel MKL-DNN, Docker & NVIDIA-Docker. For a fully managed experience, check: <https://aws.amazon.com/sagemaker>

Root device type: ebs Virtualization type: hvm

Filter by: GPU instances Current generation Show/Hide Columns							
Currently selected: p2.xlarge (11.75 ECUs, 4 vCPUs, 2.7 GHz, E5-2686v4, 61 GiB memory, EBS only)							
	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
<input type="checkbox"/>	GPU instances	g3s.xlarge	4	30.5	EBS only	Yes	Up to 10 Gigabit
<input type="checkbox"/>	GPU instances	g3.4xlarge	16	122	EBS only	Yes	Up to 10 Gigabit
<input type="checkbox"/>	GPU instances	g3.8xlarge	32	244	EBS only	Yes	10 Gigabit
<input type="checkbox"/>	GPU instances	g3.16xlarge	64	488	EBS only	Yes	25 Gigabit
<input checked="" type="checkbox"/>	GPU instances	p2.xlarge	4	61	EBS only	Yes	High

```
+-----+-----+-----+
| NVIDIA-SMI 410.104      Driver Version: 410.104      CUDA Version: 10.0 |
| GPU  Name     Persistence-M| Bus-Id     Disp.A  | Volatile Uncorr. ECC |
| Fan  Temp     Perf  Pwr:Usage/Cap| Memory-Usage | GPU-Util Compute M. |
|-----+-----+-----+-----+-----+-----+-----+-----+
| 0   Tesla K80      On           00000000:00:1E.0 Off    0 |
| N/A  41C     P8    27W / 149W |       0MiB / 11441MiB |    0% Default |
+-----+-----+-----+-----+-----+-----+-----+-----+
+
| Processes:
| GPU  PID  Type  Process name          GPU Memory Usage
|-----+-----+-----+-----+
| No running processes found
+-----+-----+-----+
```

5. Outils

5.1. Mono - GPU

5.1.2. Computing As A Service

Google Cloud Platform				
GPU	Tesla K80(4vCPU)	Tesla P100(4vcpu)		
Hardware	GCP		GCP	
Ram	16Go		16Go	
SSD	30GO		30GO	
Type de modèle	MobileNet	VGG16	MobileNet	VGG16
Durée (mn)	50.82	52.79	41.98	43.36
Temp max. (°)	35° -> 48°		40°->53°	
Accuracy (Val)	0.9316	0.9332	0.9341	0.9423
SpeedUP [K80 vs P100] (mn)			+21.05%	+20.43%
durée utilisation service GCP	~ 5H00			
Billing GCP	10,42 USD			

<input type="checkbox"/> Nom ^	Zone	Recommandation	Utilisée par	Adresse IP interne	Adresse IP externe	Connecter
<input type="radio"/> eol-p100	europe-west1-b			10.132.0.7 (nic0)	35.210.39.235 ↗	SSH ⏪ :
<input checked="" type="checkbox"/> instance-k80	europe-west1-d			10.132.0.9 (nic0)	35.187.1.191 ↗	SSH ⏪ :

GPU Elastic Amazon EC2

GPU	GTX 1080	GTX 1060	Tesla K80	Tesla V100
Hardware	tour	portable	AWS	AWS
Ram	8Go	6Go	11.4Go	16Go
Durée (mn)	30.14	47.01	33.47	21.14
Temp max. (°)	56.00	65.00	x	x
Accuracy (Val)	0.9644	0.9641	0.9637	0.9643
SpeedUP [1080 vs 1060] (mn)	35.89%			
SpeedUP [K80 vs 1080] (mn)	- 11.05%			
durée utilisation service AWS	~ 6H00			
Billing AWS	5.33 USD			
SpeedUP [V100 vs 1080] (mn)	29.86%			
durée utilisation service AWS	51 mn			
Billing AWS	3.56 USD			
Stockage s3 per Day AWS	(216.4 MB)	1.03 USD		

vCPU	ECU	Mémoire (Go)	Stockage des instances (Go)	Utilisation de Linux/UNIX
GPU Instances - Current Generation				
p3.2xlarge	8	26	61 GiB	EBS Only \$3.06 per Hour
p3.8xlarge	32	94	244 GiB	EBS Only \$12.24 per Hour

6. Modèle

6.1. Best Modèle

```
# =====
# 3. ARCHITECTURE DU RESEAU DE NEURONES CONVOLUTIONNELLES
# =====

model_conv_eol = km.Sequential()

# Source : https://keras.io/layers/convolutional/

# bloc1=Conv2D
model_conv_eol.add(kl.Conv2D(128, (3, 3),
                            input_shape=(img_width_eol, img_height_eol, 3),
                            data_format="channels_last"))
##model_conv_eol.add(BatchNormalization())
model_conv_eol.add(kl.Activation('relu'))
model_conv_eol.add(kl.MaxPooling2D(pool_size=(2, 2)))

# bloc2=Conv2D
model_conv_eol.add(kl.Conv2D(128, (3, 3)))
model_conv_eol.add(kl.Activation('relu'))
model_conv_eol.add(kl.MaxPooling2D(pool_size=(2, 2)))

# bloc4=Conv2D
model_conv_eol.add(kl.Conv2D(256, (3, 3)))
model_conv_eol.add(kl.Activation('relu'))
model_conv_eol.add(kl.MaxPooling2D(pool_size=(2, 2)))
model_conv_eol.add(kl.Flatten()) # Conversion feature 2D en feature 1D

# sortie perceptron 1 couche "full-connected"
model_conv_eol.add(kl.Dense(512))
model_conv_eol.add(kl.Activation('relu'))
model_conv_eol.add(kl.Dropout(0.5))
model_conv_eol.add(kl.Dense(1))
model_conv_eol.add(kl.Activation('sigmoid')) #binary classification

model_conv_eol.compile(loss='binary_crossentropy',
                      optimizer='Adam',
                      metrics=['accuracy'])

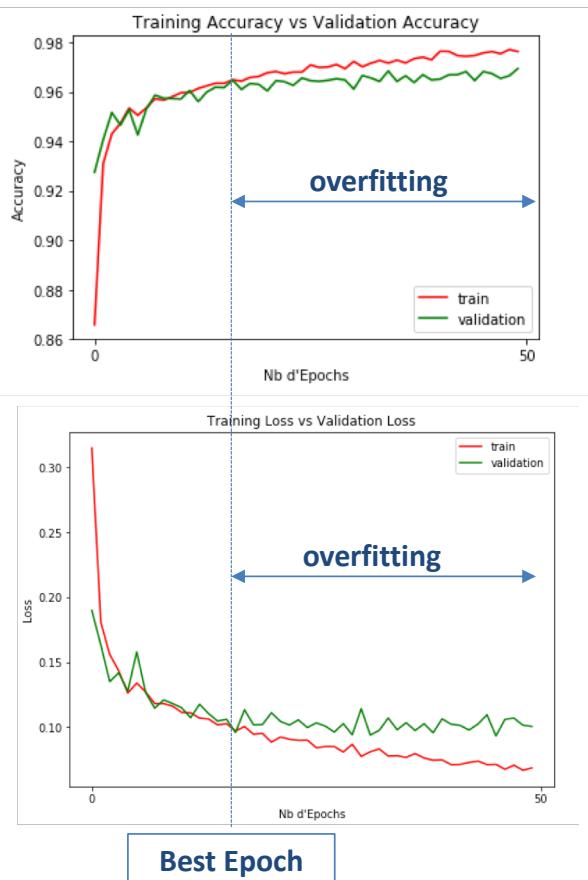
model_conv_eol.summary()
```

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 126, 126, 128)	3584
activation_1 (Activation)	(None, 126, 126, 128)	0
max_pooling2d_1 (MaxPooling2D)	(None, 63, 63, 128)	0
conv2d_2 (Conv2D)	(None, 61, 61, 128)	147584
activation_2 (Activation)	(None, 61, 61, 128)	0
max_pooling2d_2 (MaxPooling2D)	(None, 30, 30, 128)	0
conv2d_3 (Conv2D)	(None, 28, 28, 256)	295168
activation_3 (Activation)	(None, 28, 28, 256)	0
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 256)	0
flatten_1 (Flatten)	(None, 50176)	0
dense_1 (Dense)	(None, 512)	25690624
activation_4 (Activation)	(None, 512)	0
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 1)	513
activation_5 (Activation)	(None, 1)	0
Total params:	26,137,473	
Trainable params:	26,137,473	
Non-trainable params:	0	

> 26 Millions de paramètres à entraîner !

6. Modèle

6.1. Best Modèle



> Choix architecture du modèle apprentissage :

4 critères mobilisés pour sélectionner l'architecture adaptée à notre problème

- . état de l'art des modèles d'apprentissage en computer vision
- . vérification empirique modèle ranking top3 en compétition Kaggle
- . utiliser un modèle qui passe l'échelle sur classif image multilabels
- . utiliser un CNN activable empiriquement sur des datasets de petites tailles

Modèle retenu = CNN à 3 couches convolutionnelles 2D [performant sur Mnist]

> Typage des paramètres du CNN :

- . fct activation Relu (Unité de Rectification Linéaire) dans les Conv2D
- . MaxPooling2D 2 x 2 (couches d'agrégation spatiale, afin de permettre une invariance aux translations locales)

. dans la couche « fully connected »

- une fonction activation de sortie sigmoïde (k=2)

. dropout = 0.5 technique de régularisation pour réduire l'over-fitting

. les métriques de performance du modèle

- fct de perte [loss] = binary_crossentropy
- optimiseur = adam (Published as a conference paper at ICLR 2015)
- métrique = accuracy

> Mise en œuvre technique :

- . prise en compte de la taille de nos entrées et déclinaison dans les couches

- . pas d'utilisation de modèle de transfert learning VGG16

- . epochs = 50 & batch_size = 256 [valeur max pour ce type de technique et de calculs]

. travail de fine-tuning des paramètres à la main

- . Code Py : introduction, d'un compteur de temps pour mesurer l'exécution des Epochs

7. Fine Tuning

```
# =====  
PLAN ETUDE POUR MAXIMISER CRITERE SCORE RMSE  
# =====  
  
axe 1 = data quality  
. visionneuse d image, 2x checking et nettoyage a la main chaque image,  
. sanity check industriel = amazon mechanical turk  
. creation de feature en image ?,  
  
axe 2 = "force" brute grosGPU = batchsize max,  
. strategie competitors kaggle, tester a la volee plein de modeles cnn, pour 1 selection  
  
axe 3 = 1 architecture CNN + optimiser hyperparametre (Gridsearch à la main)
```

Stratégie :
Sur la base du modèle de base,
faire des itérations numériques
pour optimiser les hyper-
paramètres du modèle.

- . **Architecture réseau :** modèle base 32.3.3 > 32.3.3 > 64.3.3-64 => Modele *** : **128-3-3 > 128-3-3 > 256-3-3-512** [best RMSE]
- . **Weight Initialization :** sans transfert learning
- . **Learning Rate :** **Adam** « version base », pas de learning rate adaptatif selon descente du gradient
- . **Activation Function :** **Relu** dans les couches conv2D et sortie en Sigmoïde pour un problème de classification binaire supervisée
- . **Batch size :** batch_size *** = **256** avec meilleur score RMSE = 97.27%, gridsearch a la main, batch_size = [32,128,256,512,1024]
- . **Epochs :** gridsearch a la main, Nb epochs = [50,100] => **50** = nb d'epochs du modèle ***
- . **Regularization :** dropout = **0.5** (itérations non significatives à 0.2 et 0.3)
- . **Optimization and Loss :** optimizers testés SGD, RMSprop et Adam => **Adam "version base"** = optimiseur du modèle ***
- . **Early Stopping :** à la main, en regardant l'écart de rupture acceptable entre les courbes loss training et loss validation
- . **Data augmentation :** stratégie minimalistique shear_range / zoom_range = **0.3** pour le modèle ***
en revanche, l'itération **Centrage et reduction des entrées** ==> meilleur modèle *** (RMSE : 97.27 > 97.22)
- . **Poids (%) du split [train ; validation] :** best modèle *** - RMSE = 97.27 avec **[0.8;0.2]** – testés [0.9;0.1] / [0.7;0.3] / [0.85;0.15] / [0.75;0.25]

8. Résultats Best Model

MODEL VERSION	5_Eol.ipynb ***
DATA	Image = 128 x 128 x 3
TOPOLOGIE RESEAU	3 Conv2D + perceptron 128-3-3>128-3-3>256-3-3>512
NORMALISATION Xi	1/255
DATA AUGMENTATION	shear_range / zoom_range = 0.3 /horizontal_flip featurewise_center=True + featurewise_std_normalization=True,
DATASET	Train + validation + test
FCT ACTIVATION Conv2D	relu
FCT ACTIVATION Sortie	sigmoid
FCT PERTE	cross_entropy
FCT OPTIMISEUR	adam
METRIQUE	accuracy
TX APPRENTISSAGE	
FCT REGULARISATION	0.5
BATCH NORMALISATION	
NOMBRE EPOCHS	50
TAILLE BATCH	256
TRAIN 2 / VALIDATION 2	
Train accuracy	0.9802
Validation accuracy	0.9675
TEMPS EXECUTION (mn)	56.35 [gtx1080]
Score (RMSE)	97.27
classe des meilleurs RMSE	importance centrage et reduction entre es



Clou

InceptionV3

```
[[('n03804744', 'nail', 0.54710066),  
 ('n02690373', 'airliner', 0.02919579),  
 ('n04355338', 'sundial', 0.025510244),  
 ('n03532672', 'hook', 0.022000339),  
 ('n04258138', 'solar_dish', 0.01931357)]]
```

MobileNet

```
[[('n03804744', 'nail', 0.68149525),  
 ('n02690373', 'airliner', 0.047769096),  
 ('n01494475', 'hammerhead', 0.042381134),  
 ('n04266014', 'space_shuttle', 0.03447685),  
 ('n02074367', 'dugong', 0.025757257)]]
```

tête de marteau

VGG16

```
[[('n01494475', 'hammerhead', 0.4624241),  
 ('n01484850', 'great_white_shark', 0.06637151),  
 ('n01491361', 'tiger_shark', 0.04668102),  
 ('n02640242', 'sturgeon', 0.030736005),  
 ('n01498041', 'stingray', 0.02506855)]]
```

ResNet50

```
[[('n04355338', 'sundial', 0.4939556),  
 ('n01494475', 'hammerhead', 0.1804506),  
 ('n02690373', 'airliner', 0.09421142),  
 ('n04552348', 'warplane', 0.018284412),  
 ('n02906734', 'broom', 0.016259592)]]
```

Clou

cadran solaire

9.1 Modèles Transfert Learning

Model	Image size	Weights size	Top-1 accuracy	Top-5 accuracy	Parameters	Depth
VGG16	224 x 224	528 MB	0.715	0.901	138,357,544	23
MobileNet	224 x 224	17 MB	0.665	0.871	4,253,864	88

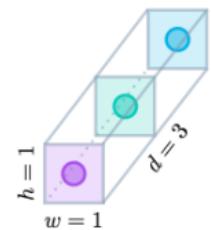
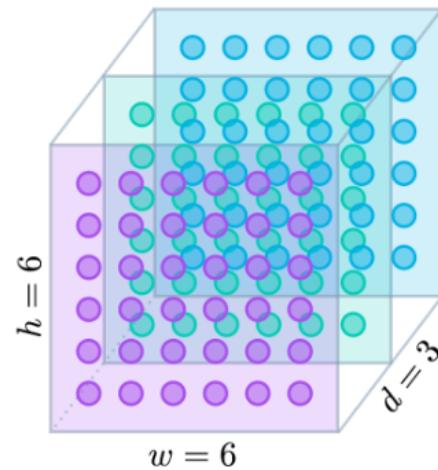
Total params: 16,814,913
 Trainable params: 2,100,225
 Non-trainable params: 14,714,688

Gèle des 20 premières couches

Total params: 5,853,377
 Trainable params: 5,817,473
 Non-trainable params: 35,904

global_average_pooling2d_3	(None, 1024)	0
dense_9	(None, 1024)	1049600
dense_10	(None, 1024)	1049600
dense_11	(None, 512)	524800
dense_12	(None, 1)	513

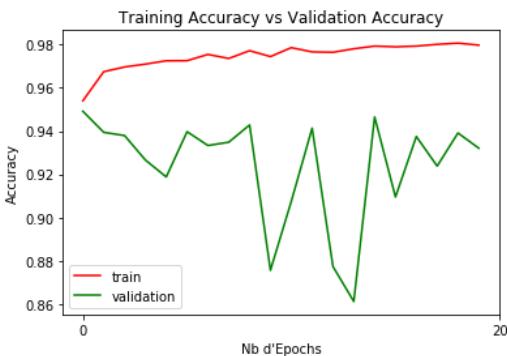
- Nombre d'epochs = **20**
- Batch-Size = **256**
- Loss = **Binary_crossentropy**
- Optimizer = '**Adam**',
- Metrics = '**accuracy**'



9.2 Résultats

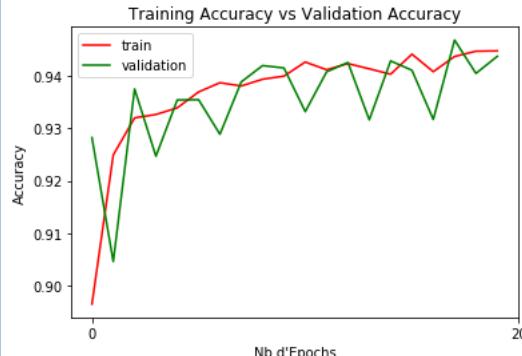
P100 (~120s/epochs)

MobileNet



Train accuracy: 0.939
Validation accuracy: 0.9341

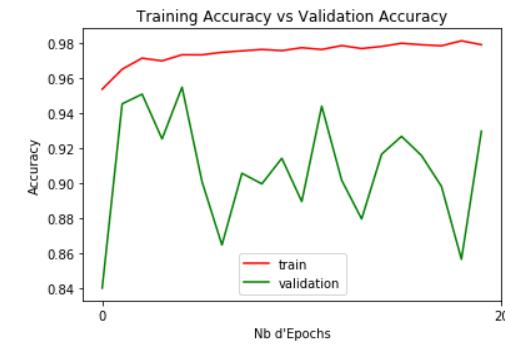
VGG16



Train accuracy: 0.9483
Validation accuracy: 0.9423 mars 2019 – 17/22

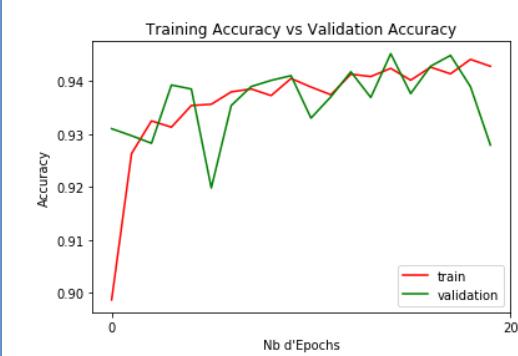
K80 (~147s/ epochs)

MobileNet



Train accuracy: 0.9306
Validation accuracy: 0.9316

VGG16



Train accuracy: 0.9305
Validation accuracy: 0.9332

9.3. Difficultés

- Nombres de GPU(s) disponibles dans une région limité (impossible de créer une instance VM)
- Problème de quota.
- Choix de la configuration optimale par rapport au travail envisagé sur la VM. (contrainte de crédit)
- Actualisation du coût de facturation très long .
- Réglage réseau pour run le notebook à distance localement (règle parefeu, adressage statique...)
- Chargement des données si ma VM est dans une région europe-west1b (~25min)

Échec du démarrage de l'instance de VM "instance-eol-dl". Erreur : The zone 'projects/centering-badge-224319/zones/europe-west1-b' does not have enough resources available to fulfill the request. Try a different zone, or try again later. X

Cette machine AWS propose différents Stack techniques CUDA pré-installés et activables

```
=====
 _|_(_/_\_) Deep Learning AMI (Ubuntu) Version 22.0
 ___\_\_|_|
=====

Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-1077-aws x86_64v)

Please use one of the following commands to start the required environment with the framework of your choice:
for MXNet(+Keras2) with Python3 (CUDA 9.0 and Intel MKL-DNN) _____ source activate mxnet_p36
for MXNet(+Keras2) with Python2 (CUDA 9.0 and Intel MKL-DNN) _____ source activate mxnet_p27
for MXNet(+Amazon Elastic Inference) with Python3 _____ source activate amazonei_mxnet_p36
for MXNet(+Amazon Elastic Inference) with Python2 _____ source activate amazonei_mxnet_p27
for TensorFlow(+Keras2) with Python3 (CUDA 10.0 and Intel MKL-DNN) _____ source activate tensorflow_p36
for TensorFlow(+Keras2) with Python2 (CUDA 10.0 and Intel MKL-DNN) _____ source activate tensorflow_p27
for Tensorflow(+Amazon Elastic Inference) with Python2 _____ source activate amazonei_tensorflow_p27
for Tensorflow(+Amazon Elastic Inference) with Python3 _____ source activate amazonei_tensorflow_p36
for Theano(+Keras2) with Python3 (CUDA 9.0) _____ source activate theano_p36
for Theano(+Keras2) with Python2 (CUDA 9.0) _____ source activate theano_p27
for PyTorch with Python3 (CUDA 10.0 and Intel MKL) _____ source activate pytorch_p36
for PyTorch with Python2 (CUDA 10.0 and Intel MKL) _____ source activate pytorch_p27
for CNTK(+Keras2) with Python3 (CUDA 9.0 and Intel MKL-DNN) _____ source activate cntk_p36
for CNTK(+Keras2) with Python2 (CUDA 9.0 and Intel MKL-DNN) _____ source activate cntk_p27
for Caffe2 with Python2 (CUDA 9.0) _____ source activate caffe2_p27
for Caffe with Python2 (CUDA 8.0) _____ source activate caffe_p27
for Caffe with Python3 (CUDA 8.0) _____ source activate caffe_p35
for Chainer with Python2 (CUDA 9.0 and Intel iDeep) _____ source activate chainer_p27
for Chainer with Python3 (CUDA 9.0 and Intel iDeep) _____ source activate chainer_p36
for base Python2 (CUDA 9.0) _____ source activate python2
for base Python3 (CUDA 9.0) _____ source activate python3

Official Conda User Guide: https://docs.conda.io/projects/conda/en/latest/user-guide/
AWS Deep Learning AMI Homepage: https://aws.amazon.com/machine-learning/amis/
```

→ C https://aws.amazon.com/fr/ec2/instance-types/p3/ 

Applications Apple Importés depuis S... Télécharger toute... Matrice de corréla... http://www.career... https://chrisalbon... Apprenez à utiliser... Autres fai

aws Contacter l'équipe commerciale Support Français Mon compte Crée un compte AWS

Produits Solutions Tarification Documentation Apprendre Réseau de partenaires AWS Marketplace Découvrir davantage Q

Instance Amazon EC2 P3 - Détails du produit

Taille d'instance	GPU - Tesla V100	Pair à pair GPU	Mémoire de GPU (Go)	Processeurs virtuels	Mémoire (Go)	Bande passante réseau	Bande passante EBS	Prix/heure pour les instances à la demande*	Tarif horaire effectif des instances réservées sur 1 an*	Tarif horaire effectif des instances réservées sur 3 ans*
p3.2xlarge	1	N/A	16	8	61	Jusqu'à 10 Go/s	1,5 Go/s	3,06 USD	1,99 USD	1,05 USD
p3.8xlarge	4	NVLink	64	32	244	10 Go/s	7 Go/s	12,24 USD	7,96 USD	4,19 USD
p3.16xlarge	8	NVLink	128	64	488	25 Go/s	14 Go/s	24,48 USD	15,91 USD	8,39 USD
p3dn.24xlarge	8	NVLink	256	96	768	100 Go/s	14 Gb/s	31,218 USD	18,30 USD	9,64 USD

Modèle GPU NVIDIA	Performances en virgule flottante double précision (64 bits)
GeForce GTX Titan X Maxwell	jusqu'à 0,206 TFLOPS
GeForce GTX 1080 Ti	jusqu'à 0,355 TFLOPS
GeForce Titan Xp	jusqu'à 0.380 TFLOPS
GeForce Titan V	jusqu'à 6,875 TFLOPS
GeForce RTX 2080 Ti	environ ~ 0.44 TFLOPS
Tesla K80	1.87+ TFLOPS
Tesla P100 *	4.7 ~ 5.3 TFLOPS
Quadro GP100	5.2 TFLOPS
Tesla V100 *	7 ~ 7.8 TFLOPS
Quadro GV100	7.4 TFLOPS
Quadro RTX 6000 et 8000	~ 0.5 TFLOPS
Tesla T4	estimé à environ 0,25 TFLOPS

* La valeur exacte dépend des références SKU PCI-Express ou SXM2

<https://www.microway.com/knowledge-center-articles/comparison-of-nvidia-geforce-gpus-and-nvidia-tesla-gpus/>

```
omar.attaf@instance-k80:~/data_airbus_defi$ cat /proc/version
Linux version 4.9.0-8-amd64 (debian-kernel@lists.debian.org) (gcc version 6.3.0 20170516 (Debian 6.3.0-18+deb9u1) )
#1 SMP Debian 4.9.130-2 (2018-10-27)
```