Présentation Deep Learning

12 février 2021

CAI Eddy, MBAE Hakim, SHEIKH Rakib, TARANTO Tom

Sommaire

1) Introduction
2) Perceptron Multi Couches
3) UNet
4) ResNet
5) Convolution G. Sambasivam
6) EfficientNet-B0 - B4 - B7

Introduction

Nous avons choisi de travailler sur le dataset présent sur Kaggle de classification des maladies des feuilles de manioc. Nous avons travaillé avec différentes machines. En local sur nos GPU, avec un GPU type industriel (Tesla), et avec les TPU et GPU Kaggle.

Nous avons fait plusieurs types de modèles. Actuellement, nous sommes classés 3001(12/02/20201:12:41) avec un score de 81.9%.

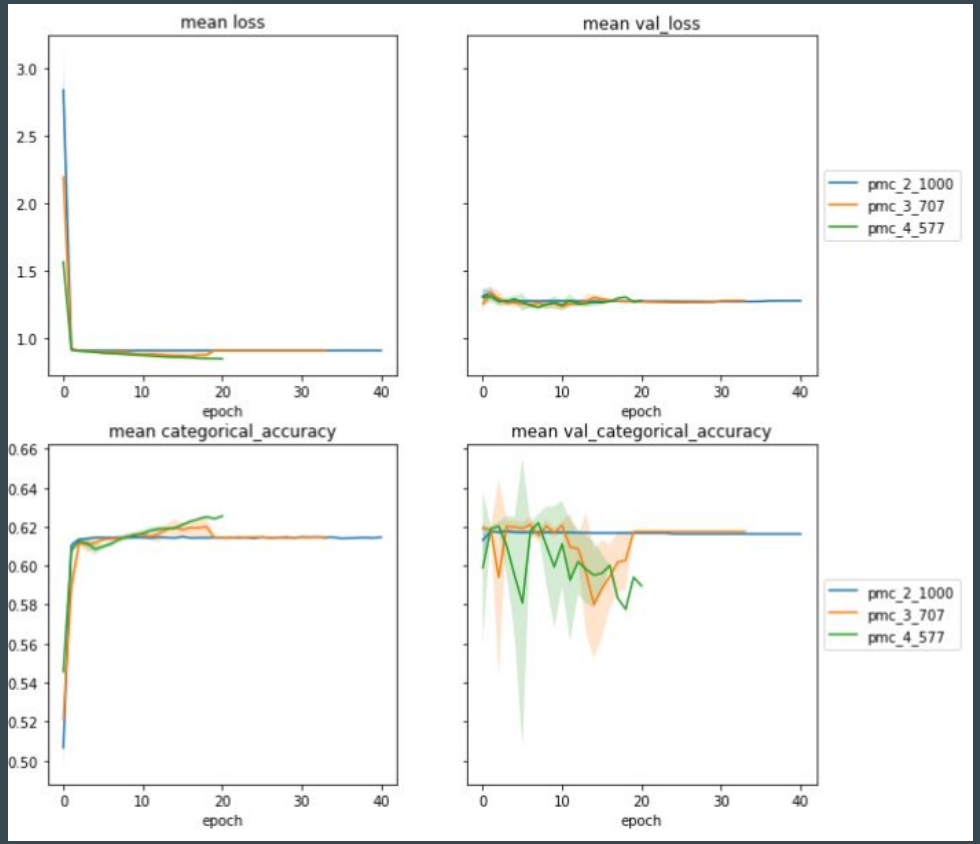
Nous allons vous montrer comment nous sommes parvenus à ce résultat.

Perceptron multicouches

Etudier à nombre de variable équivalent (1 000 000)

	Nombre de couches	Taille des couches	Régularisation
PMC_2_1000	2	1000	Batch Normalisation
PMC_3_707	3	707	Batch Normalisation
PMC_4_577	4	577	Batch Normalisation

Perceptron multicouches



Analyse

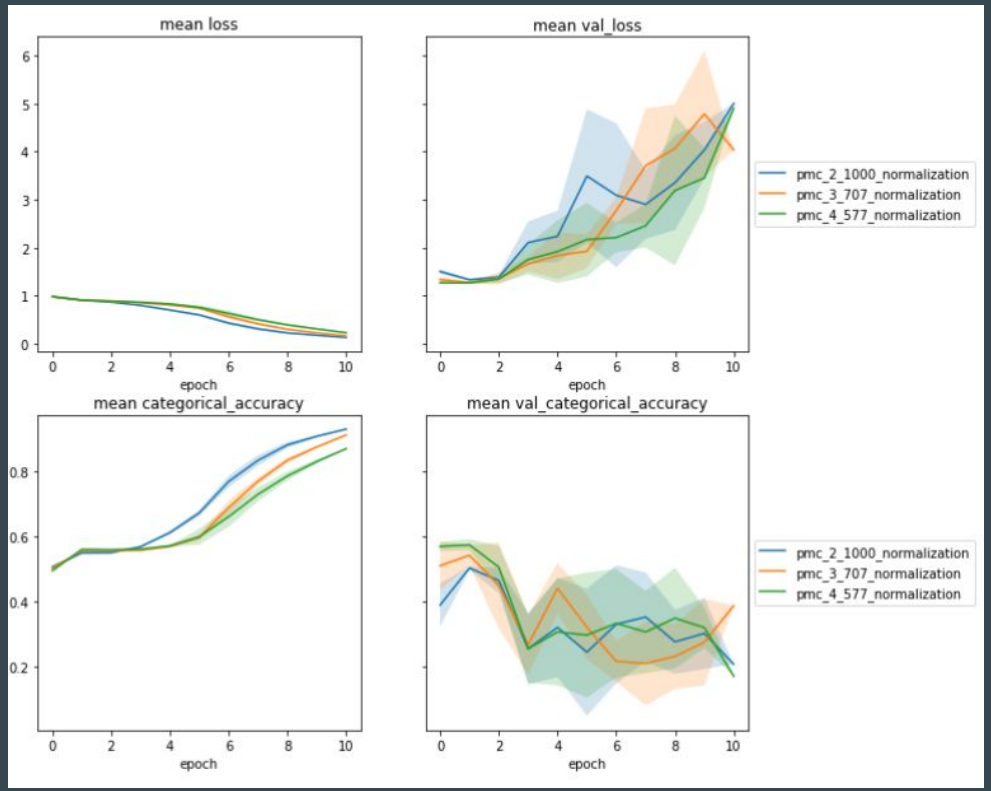
Pour variables équivalent :

++couche --taille

Généralisation : --

Apprentissage : --

Perceptron multicouches



Analyse

Apprentissage : ++

Validation : médiocre

Généralisation : overfitting

Unet

Difficulté Taille en mémoire

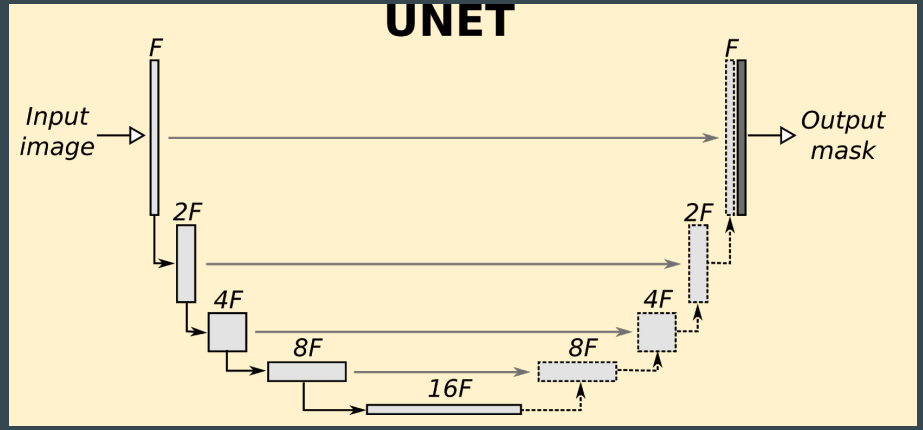
TPU Kaggle

Profondeur 3

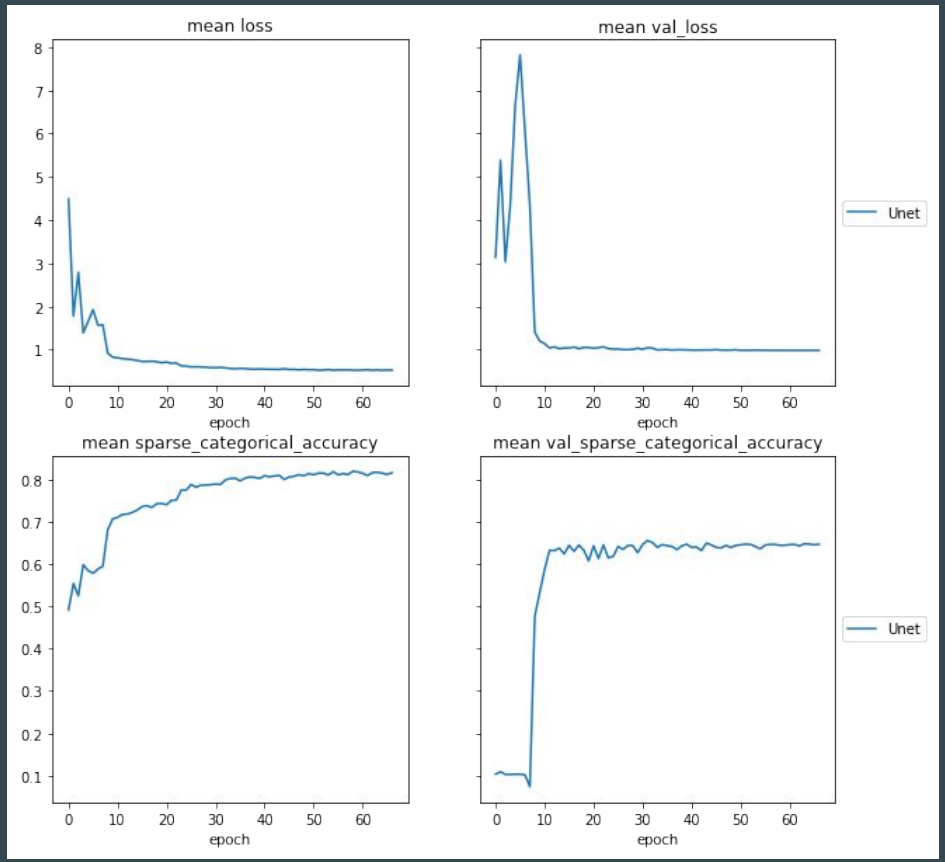
Taille des convolutions : 32 , 64, 128 -> 128, 64, 32

Régularisation : Batch Normalisation

Activation : Relu



Unet



Analyse

Apprentissage par palier

Mauvaise généralisation

Sensiblement meilleur que le perceptron multicouches

Intéressant d'augmenter la profondeur

Machine plus puissante

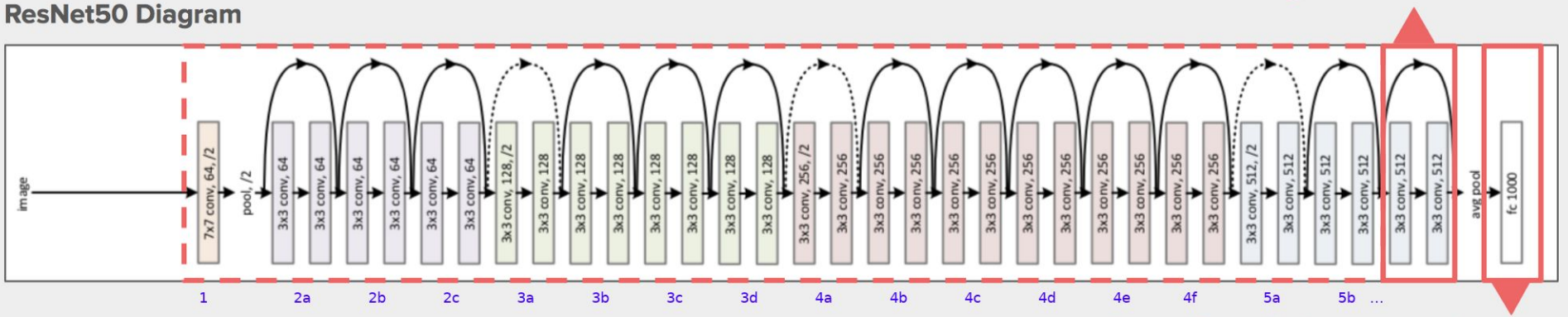
ResNet50

50 couches de convolutions successives

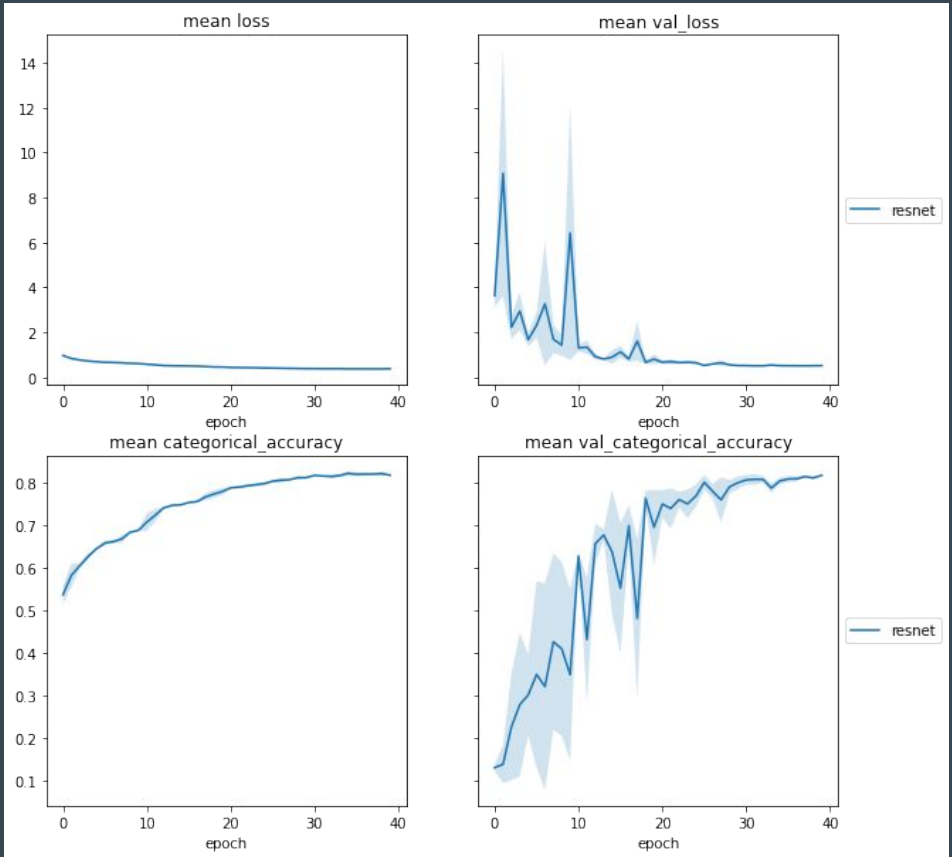
Copie des poids tous les 3 couches

Utilisation GPU Tesla V100, pas de réitération

Peu de data augmentation + pas de cache => temps d'apprentissage très long



ResNet 50



Analyse

Apprentissage en peu d'epoch

Validation un peu plus longue

Très bonne généralisation

Intéressant d'augmenter le nombre de convolutions

Machine plus puissante

Convolution G. Sambasivam

- Article publié en 2020 https://doi.org/10.1016/j.eij.2020.02.007
- Couches de convolution
- Couches Denses

Ré-implémentation de l'article dans le but d'obtenir des résultats similaires avec du fine tuning.

Taille de l'image : 448 x 448 x 3

Résultats obtenus : 93 %

Machine utilisée : CPU only, (3600s par epoch, 5,16 jour d'entrainement)

Augmentation des données : Histogram Equalization (CLAHE), Random Shearing, Random cropping, random scaling, center zoom, height and width shift

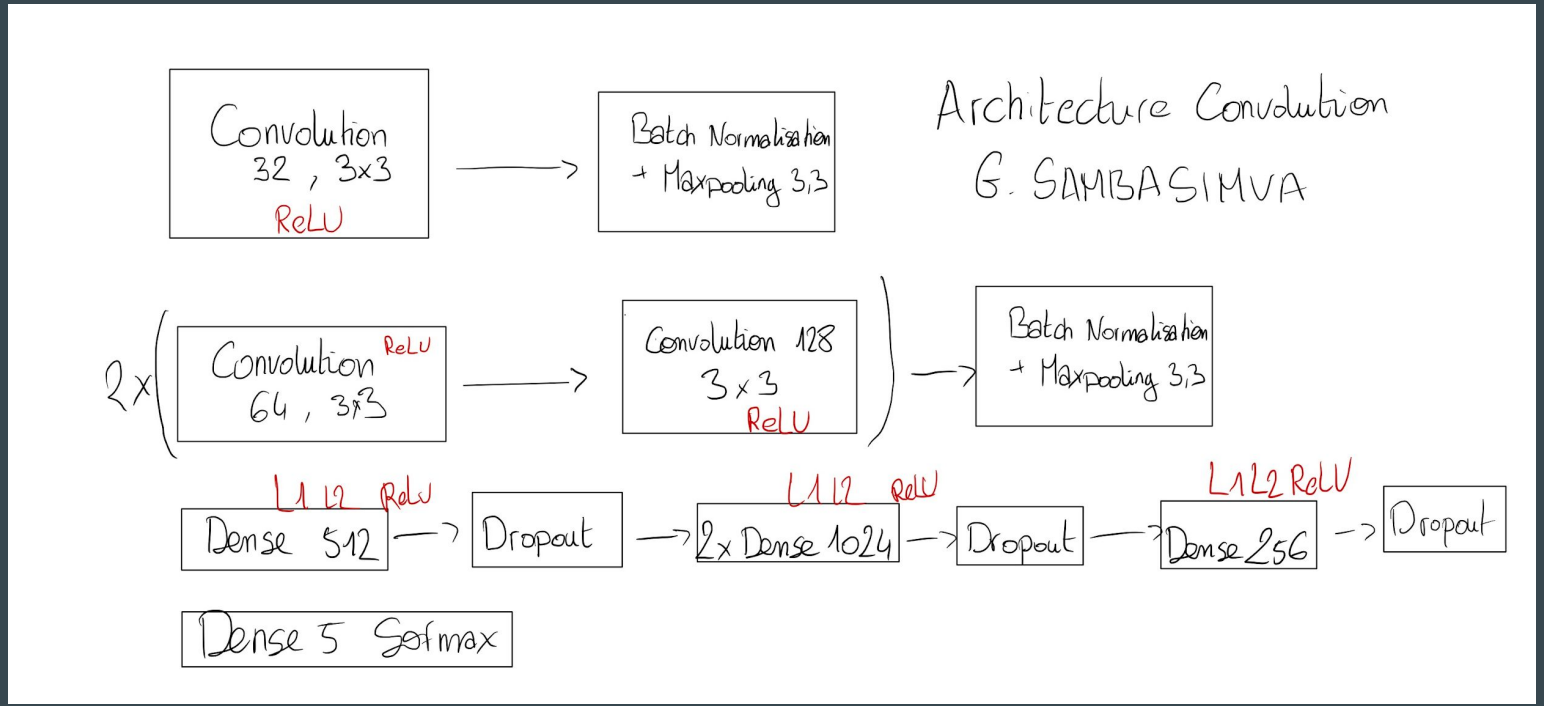
Convolution G. Sambasivam

Augmentation des données utilisés :

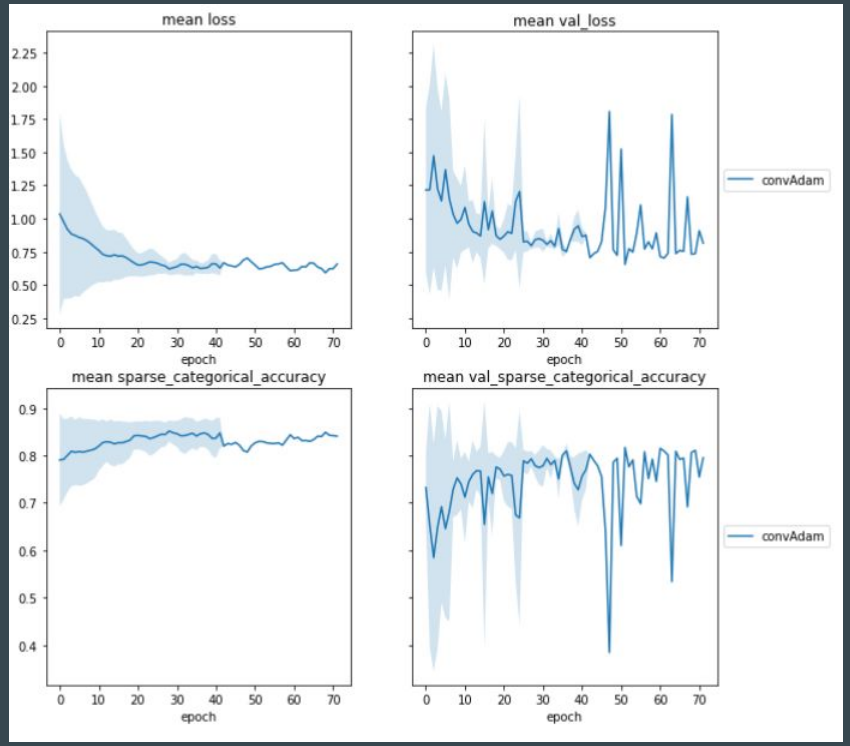
- random_flip_left_right
- ramdom_flip_up_down
- random_saturation
- random_constrast
- random_jpeg quality
- image.rot90 (avec une fonction aléatoire uniforme)

Taille de l'image: 256 x 256 x 3

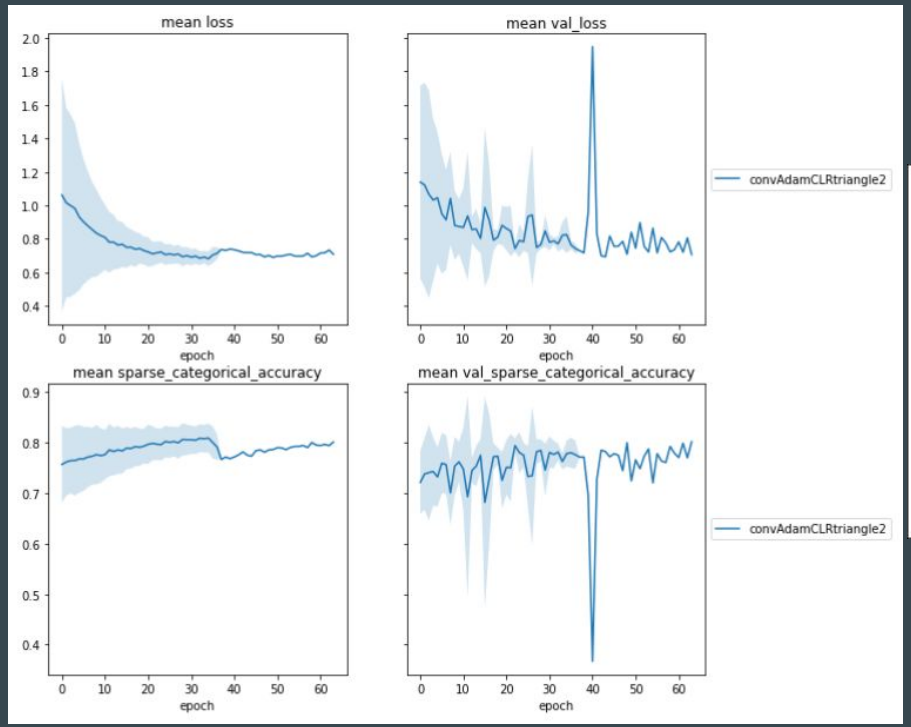
Convolution G. Sambasivam



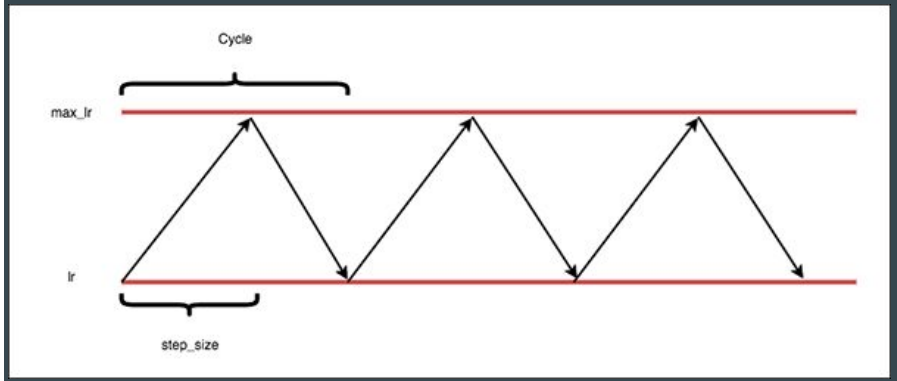
Convolution G. Sambasivam (79.5% Kaggle) Adam Only



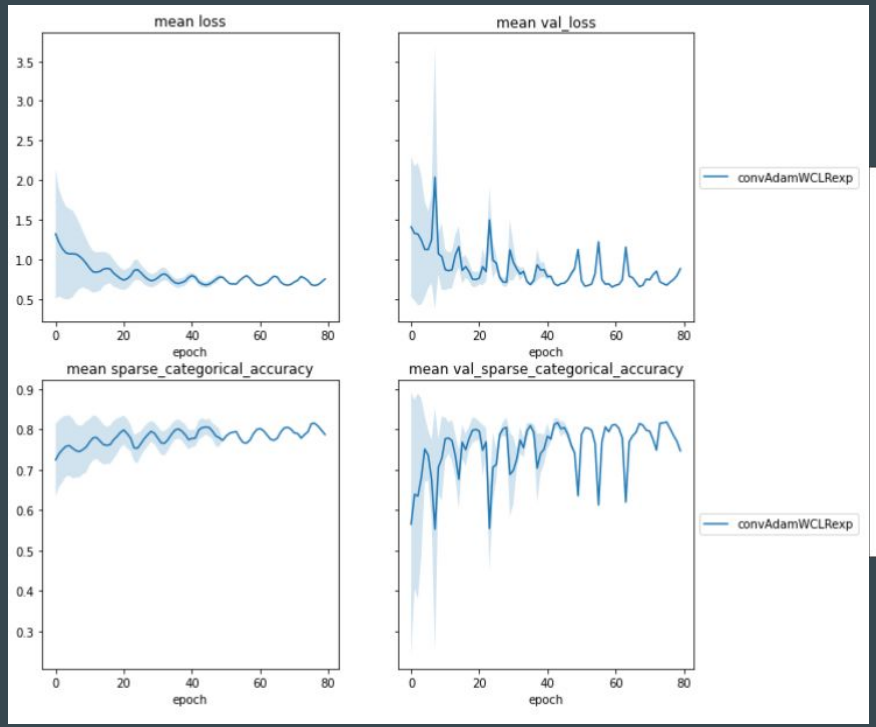
Convolution G. Sambasivam (80.4% Kaggle) Adam + CLR T2



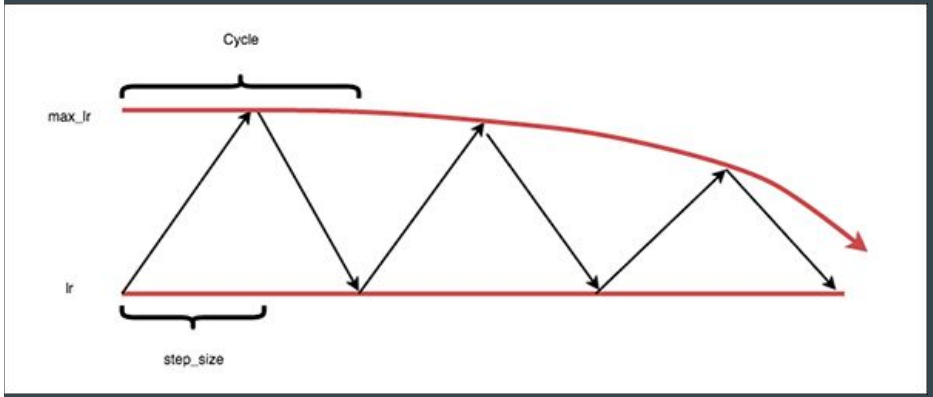
CLR Triangulaire 2 : (1 / (2(x-1))



Convolution G. Sambasivam (81.9% Kaggle) AdamW CLR EXP



CLR EXP = Gamma * x (gamma = 1)



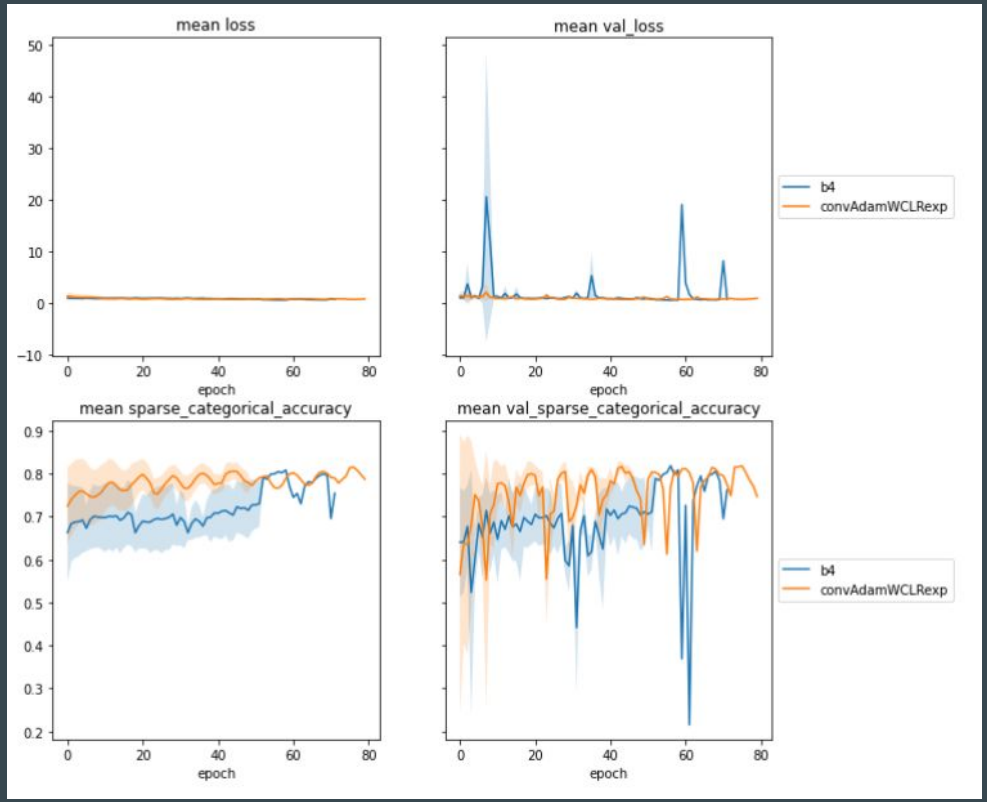
EfficientNet B4 VS Convolution G. Sambasivam

EffnetNet B4 : 81.8 % Kaggle

Convolution SAMBASIVA : 81.9 %

Modèle plus rapide à apprendre et

plus léger (Détails dans le rapport)



Conclusion

Lecture de documents scientifiques

Application de techniques de pointe

Enrichissement personnel

Modèles légers et performants

Article + notebook + présentation