

# Projet : Reconnaissance automatique de vidéos avec un processeur photonique neuro-inspiré

Georgin Nicolas

Date : 23/01/2019

# Sommaire




1. Profiling sur de grands réseaux 2025 & 4 096
2. Fonction de sous-échantillonnage
3. Résultats
4. Discussion

# Profiling

- Rappel : la fonction pinv ne pose pas de problème pour un petit réseau ( 1024 )
  - Profiling sur le code amélioré







## Profile Summary

Generated 17-Jan-2020 17:47:48 using performance time.







<u>Function Name</u>	<u>Calls</u>	<u>Total Time</u>	<u>Self Time*</u>	Total Time Plot (dark band = self time)
<u>rc_kth_pca_mix_score</u>	1	330.638 s	228.450 s	
<u>analyse_kth_pca_mix</u>	18	74.089 s	12.346 s	
<u>mode</u>	36	61.665 s	61.665 s	
<u>pinv</u>	18	16.371 s	16.371 s	
<u>...ca_mix_score&gt;@(x)slm_transf_lut(x)</u>	955008	11.026 s	11.026 s	

# Profiling

N = 2025

Line Number	Code	Calls	Total Time	% Time	Time Plot
<a href="#">136</a>	A = mask * inputs;	18	228.867 s	31.7%	
<a href="#">182</a>	weights = P' * pinv(R);	18	115.493 s	16.0%	
<a href="#">180</a>	R = X*X' + reg_te...	18	99.406 s	13.8%	
<a href="#">156</a>	res_in = A(:, t) + w * C;	954990	77.817 s	10.8%	
<a href="#">198</a>	analyse_kth_pca_mix;	18	75.611 s	10.5%	
All other lines			123.790 s	17.2%	
Totals			720.984 s	100%	

N = 4096

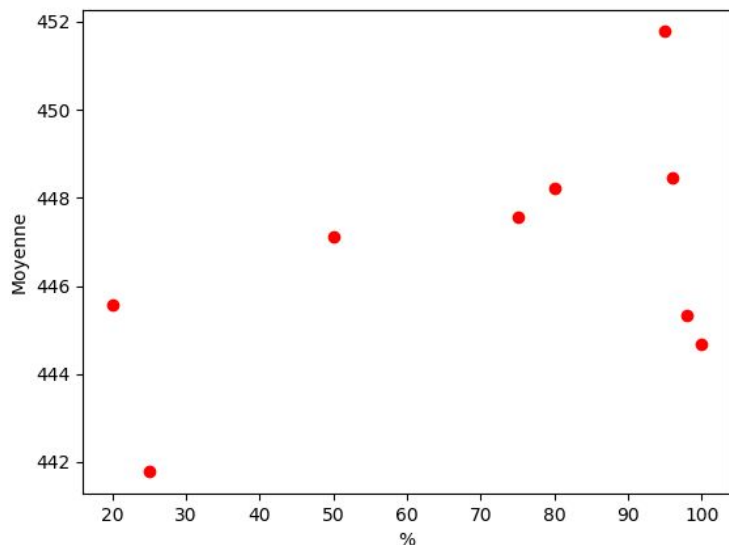
Line Number	Code	Calls	Total Time	% Time	Time Plot
<a href="#">182</a>	weights = P' * pinv(R);	18	1240.714 s	43.8%	
<a href="#">136</a>	A = mask * inputs;	18	476.794 s	16.8%	
<a href="#">180</a>	R = X*X' + reg_te...	18	397.232 s	14.0%	
<a href="#">156</a>	res_in = A(:, t) + w * C;	954990	330.932 s	11.7%	
<a href="#">198</a>	analyse_kth_pca_mix;	18	82.344 s	2.9%	
All other lines			302.368 s	10.7%	
Totals			2830.384 s	100%	

# Fonction de sous échantillonnage

1. Fonction sous-échantillonne à **intervalle régulier**
2. Prends en argument les données et un coefficient correspondant au pourcentage que l'on veut garder.
3. Exemple : 0.5, 50 %, on garde 1 image sur 2
4. 2 cas à distingué
  - a. Si le coefficient est inférieur à 0.5, exemple 0.1, (10 %), l'algorithme retient 1 image sur 10
  - b. Si le coefficient, il est supérieur à 0.5, exemple 0.95 (95 %), l'algorithme rejette 1 image sur 20
5. Coefficient pour le sous-échantillonnage de la base de données
  - a. 0.1 0.2 0.25 0.5 0.75 0.8 0.95 0.96 0.98 1
  - b. D'autres coefficients peuvent marcher.

# Résultats

1. Le sous-échantillonnage de la base de données est testé sur le jeu d'hyper-paramètres du code transmis initialement.
2. Moyenne des scores



Moyenne sur le jeux de paramètres original, pour une taille de réseau de 1024.  
Meilleur résultat en moyenne, excepté une base contenant pour 10% et 25%.

Meilleur **score 520** obtenu avec **25 % de données**

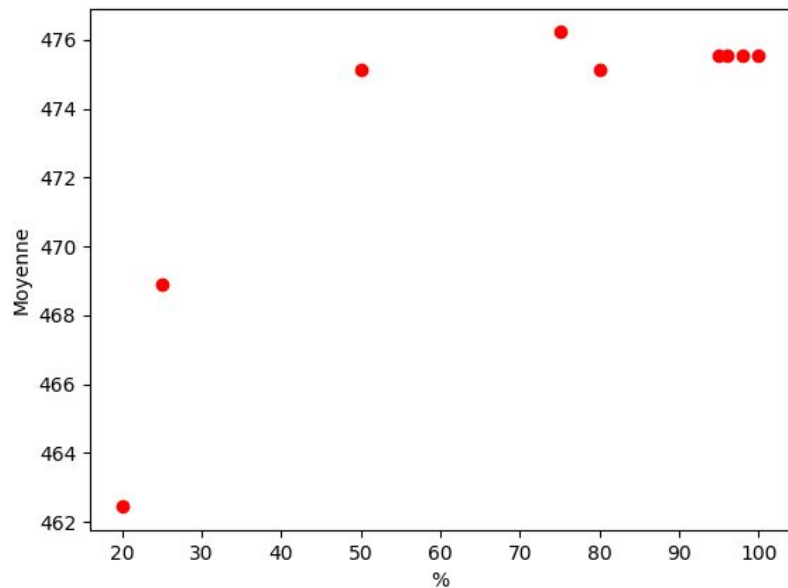
Input gain : 0.01

Feedback gain : 1

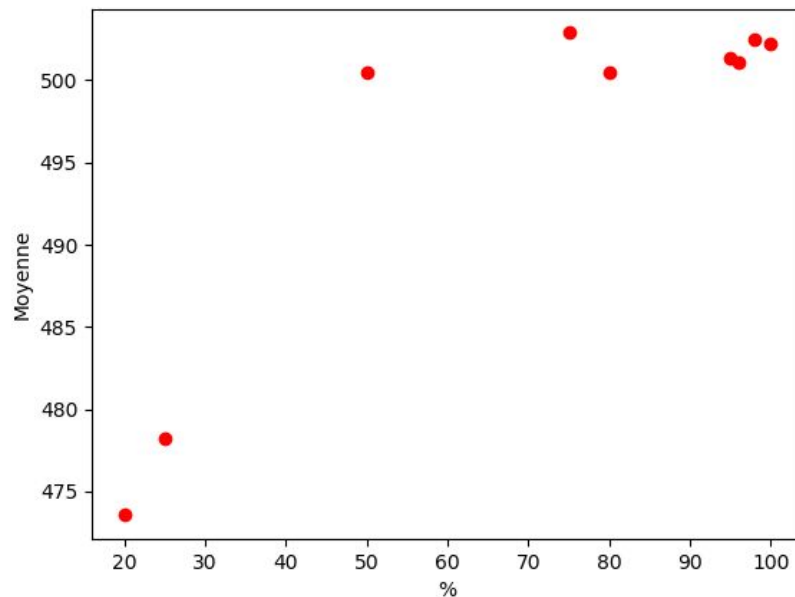
Interconnectivity gain : 0.01

Interconnivity matrix density : 0.01

# Résultats sur de plus grands réseaux



N = 2025



N = 4096

# Discussion

1. Faire d'autres types échantillonnage
  - a. Prendre la moitié de chaque vidéo
2. Impossible échantillonnage random
  - a. Problème de taille des données qui fait planté le logiciel
3. Cross-validation
4. Pour de petits réseaux, le sous-échantillonnage semble marcher, pour de grand réseau c'est moins le cas
  - a. Problème plus complexe, besoin de plus de données.
  - b. Pour des petits réseaux, les images doivent se ressembler, redondance niveau des données.