Octave Convolution

Thomas Weber

Introduction

- En classification d'images, les CNNs sont très performants

- Comment améliorer cette performance ?
  - Amélioration de la topologie du réseau avec des connexions raccourcies (ResNet, DenseNet)
  - Utiliser des convolutions « depth-wise » pour réduire le coût de calcul des convolutions classiques (Xception, MobileNet)
  - Recherche automatique de la meilleure architecture (NAS, PNAS, AmoebaNet)

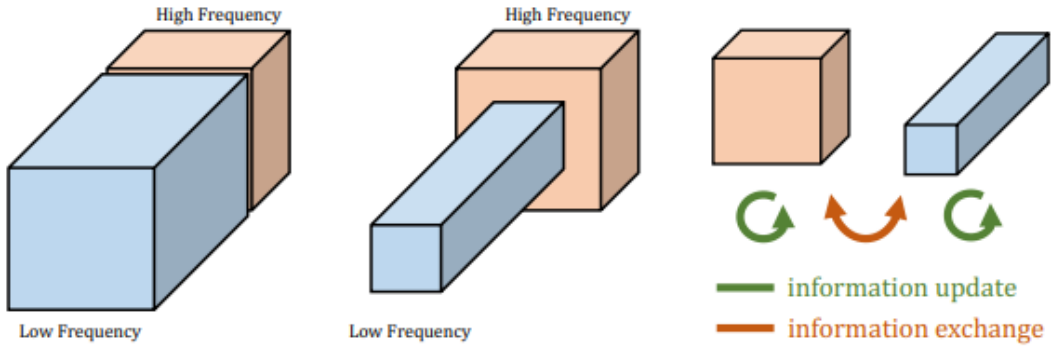
- Idée retenue: exploiter la redondance spatiale des feature maps avec un nouvel opérateur de convolution: Octave Convolution

Redondance spatiale

- Décomposition de l’image en deux parties:
  - Haute fréquence: représente les détails qui changent rapidement
  - Basse fréquence: représente la structure globale qui évolue lentement

Principe

- Séparation des feature maps en 2 groupes
- Réduction spatiale de la partie basse fréquence d'une octave (par 2)
- Intérêt ? :
  - Gain en mémoire et coût de calcul
  - Agrandissement du champ réceptif pour les basse fréquences
  - Amélioration des performances
- Difficultés:
  - Communication efficace entre les groupes
  - Intégration « plug-and-play »



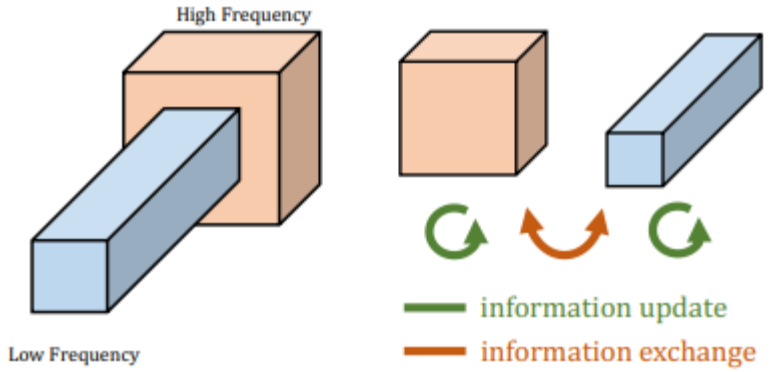
Objectif

- Implémenter le nouvel opérateur de convolution
- L’utiliser sur des nouvelles conditions:
  - Architectures
  - Datasets
- Vérifier que les gains de performances sont généralisables

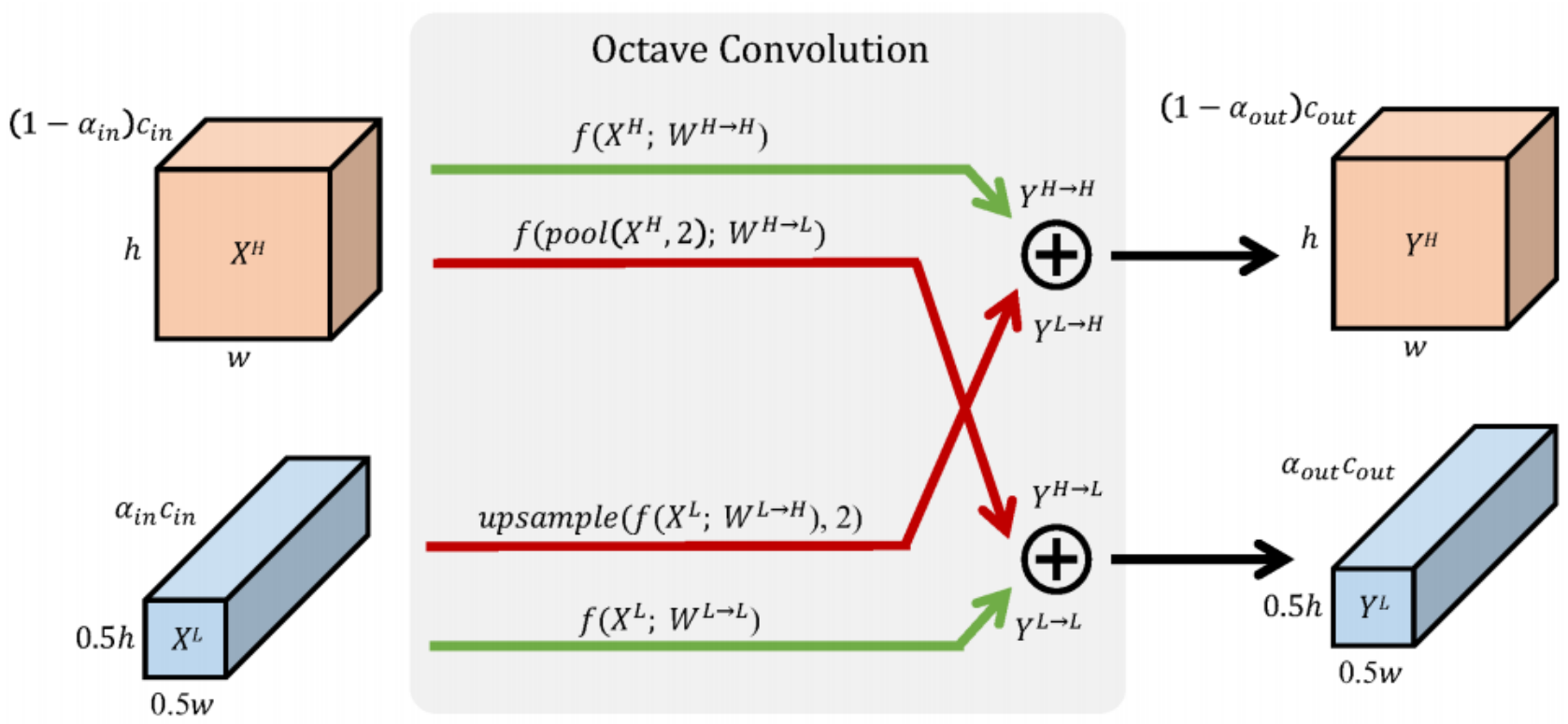
Représentation multi-échelles

- Avant: $X \in \mathbb{R}^{c \times h \times w}$
- Après:
  - $X = \{X^H; X^L\}$
  - $X^H \in \mathbb{R}^{(1-\alpha)c \times h \times w}$
  - $X^L \in \mathbb{R}^{\alpha c \times \frac{h}{2} \times \frac{w}{2}}$
- Apparition d'un paramètre alpha entre 0 et 1: proportion de features basse fréquence

- En sortie:
  - $Y = \{Y^H, Y^L\}$
  - $Y^H = Y^{H \to H} + Y^{L \to H}$
  - $Y^L = Y^{L \to L} + Y^{H \to L}$

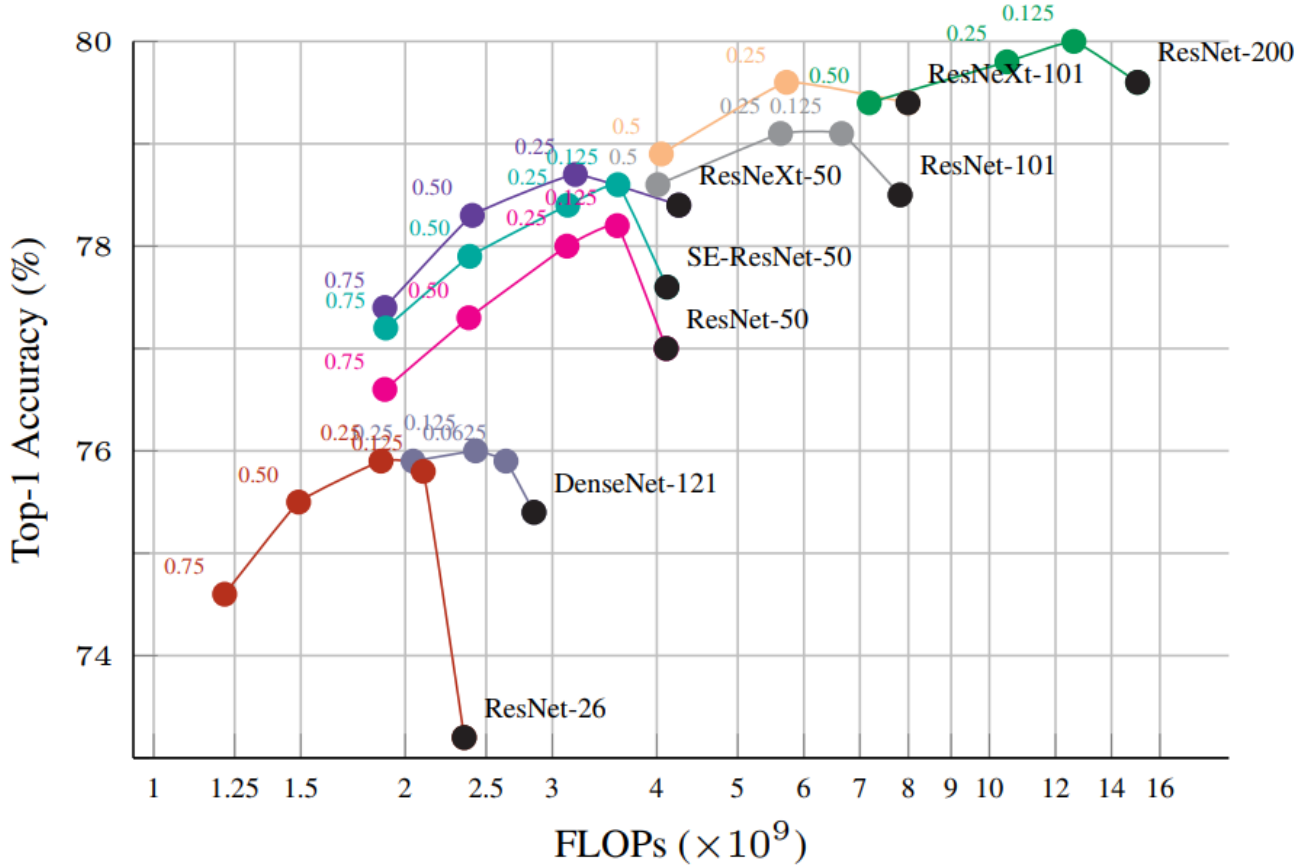


Octave Convolution



$$Y^H = f(X^H, W^{H\to H}) + upsample(f(X^H, W^{L\to H}), 2)$$
$$Y^L = f(X^L, W^{L\to L}) + f(pool(X^H, 2), W^{H\to L})$$

Résultats


Top-1 Accuracy (%)
80
78
76
74
1
1.25
1.5
2
2.5
3
4
5
6
7
8
9
10
12
14
16
FLOPs ($\times 10^9$)
ResNet-200
ResNeXt-101
ResNet-101
ResNeXt-50
SE-ResNet-50
ResNet-50
DenseNet-121
ResNet-26

Preuve de concept

- Choix d'une implémentation existante:

https://github.com/koshian2/OctConv-TFKeras

- Test sur réseau Wide ResNet et dataset CIFAR-10

Alpha	Test accuracy (on repository)	Test accuracy (ours)
0	88.47 %	88.22 %
0.125		94.64 %
0.25	94.83 %	94.53 %
0.5	94.40 %	93.64 %
0.75	93.54 %	92.50 %

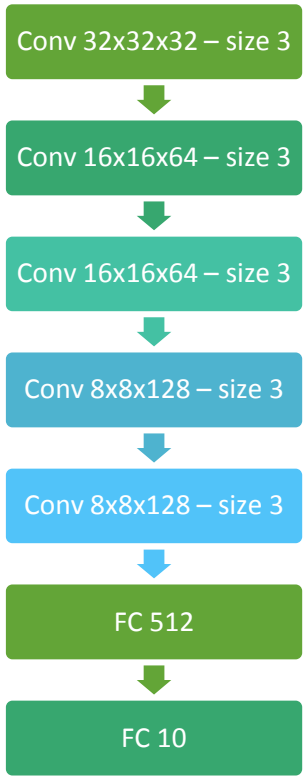
- Choix des datasets:
  - CIFAR-10: 60000 images / 10 classes
  - Stanford Dogs Dataset: 20000 images / 120 classes

- Choix des architectures:
  - CNN 7 couches
  - CNN 9 couches
  - Network-in-Network

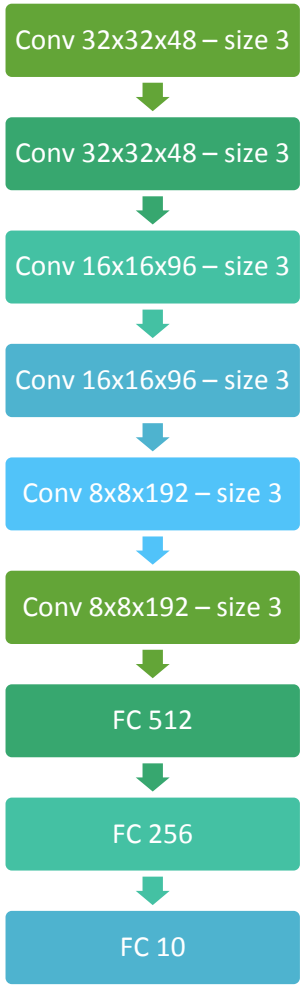
- Choix d'alpha:
  - 0, 0.125, 0.25, 0.5, 0.75

Architectures

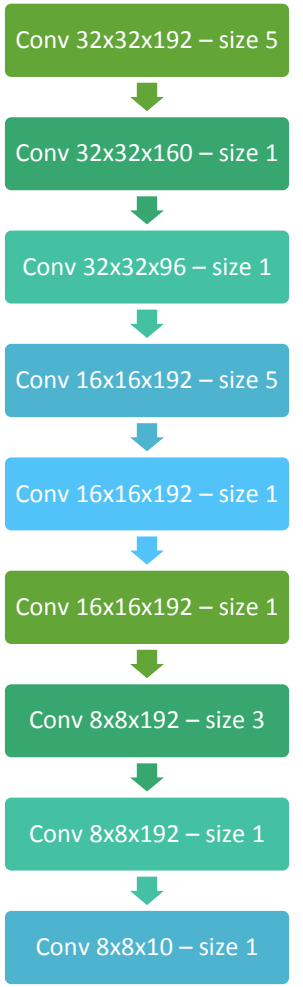
CNN 7 couches



CNN 9 couches



Network in Network



Résultats CIFAR-10

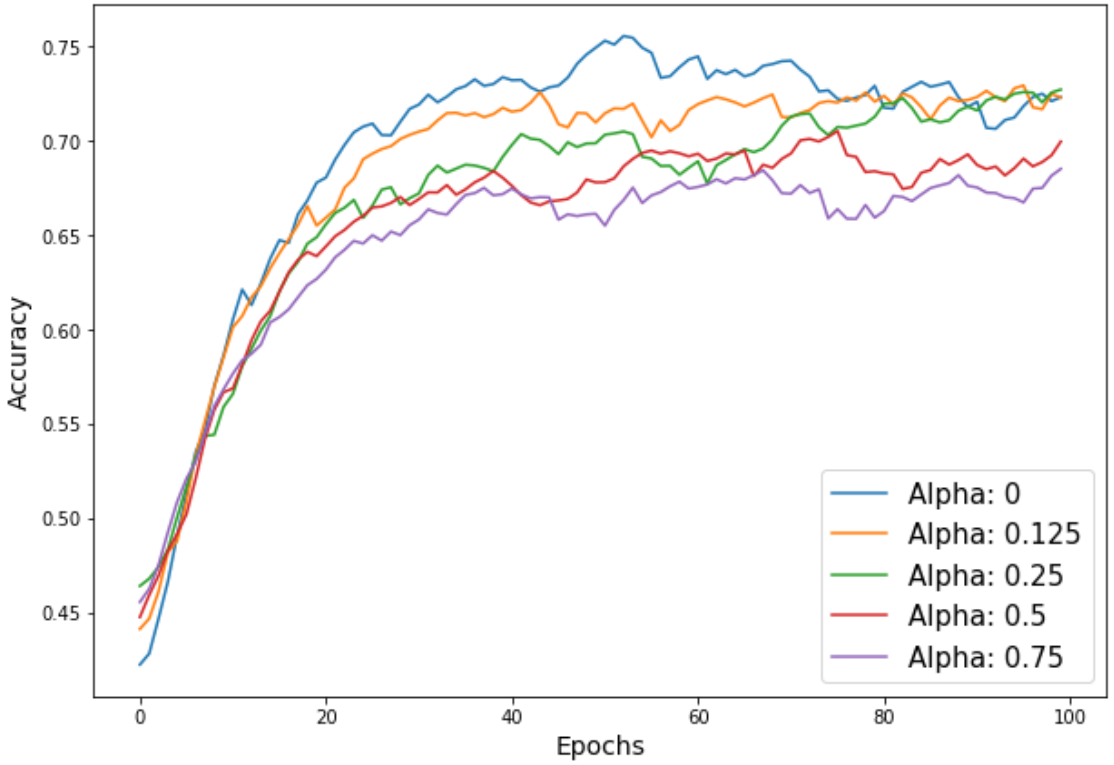


Figure 1: Test accuracy with 7-layers CNN (smoothed curve)

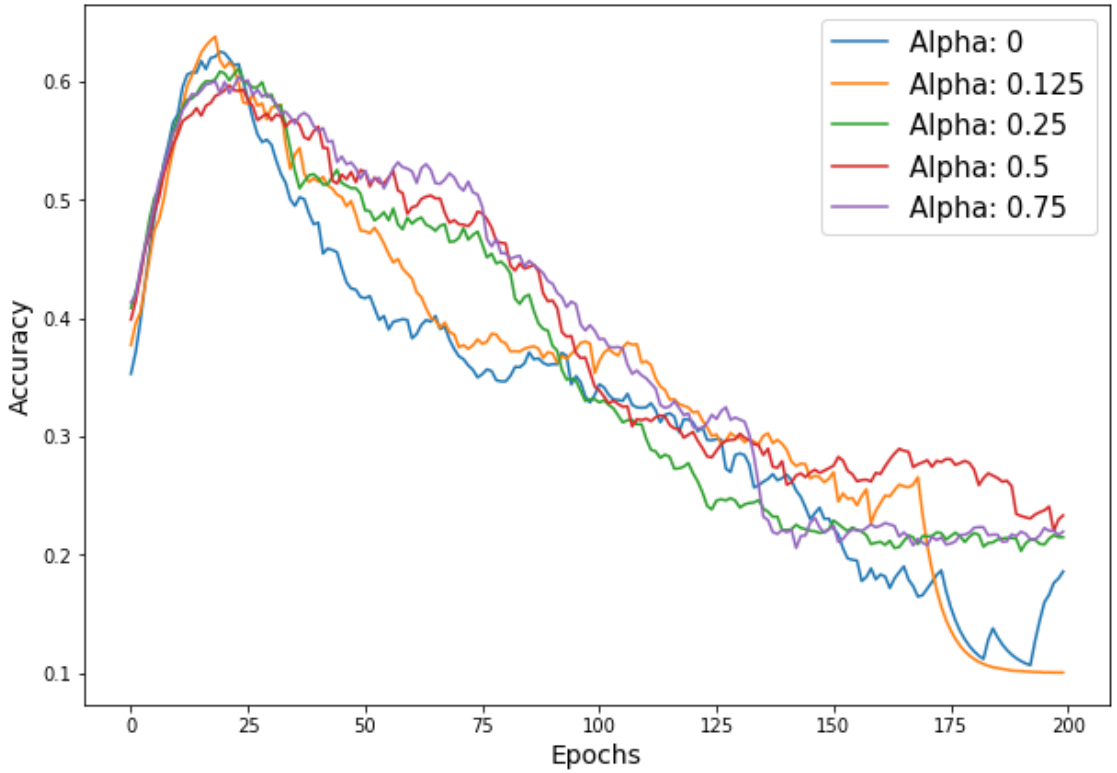


Figure 2: Test accuracy with 9-layers CNN (smoothed curve)

Résultats CIFAR-10

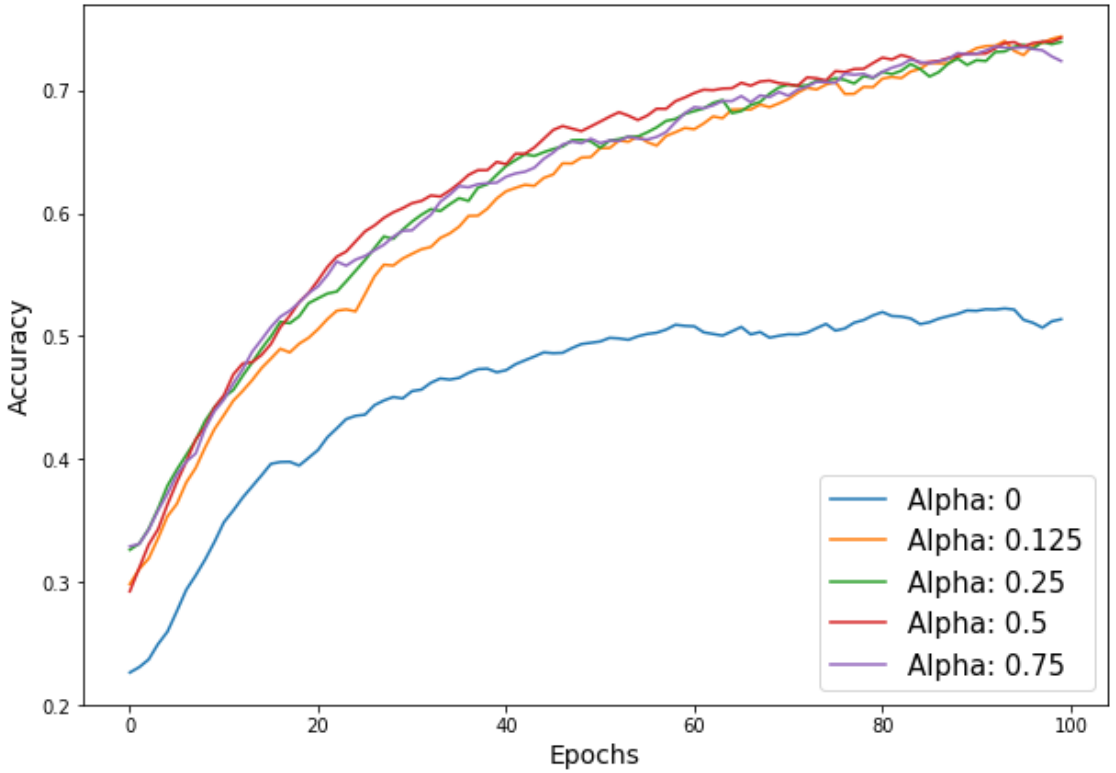


Figure 3: Test accuracy with NiN (smoothed curve)

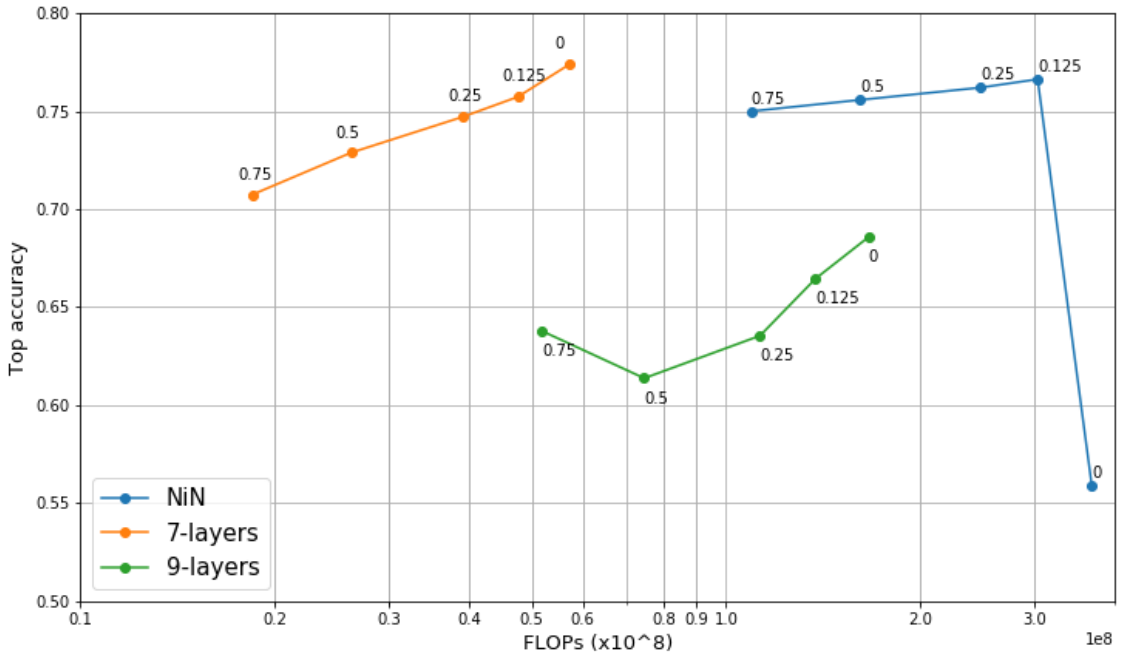


Figure 4: Accuracy-FLOPs trade-off curve on CIFAR-10

Résultats Stanford Dogs Dataset

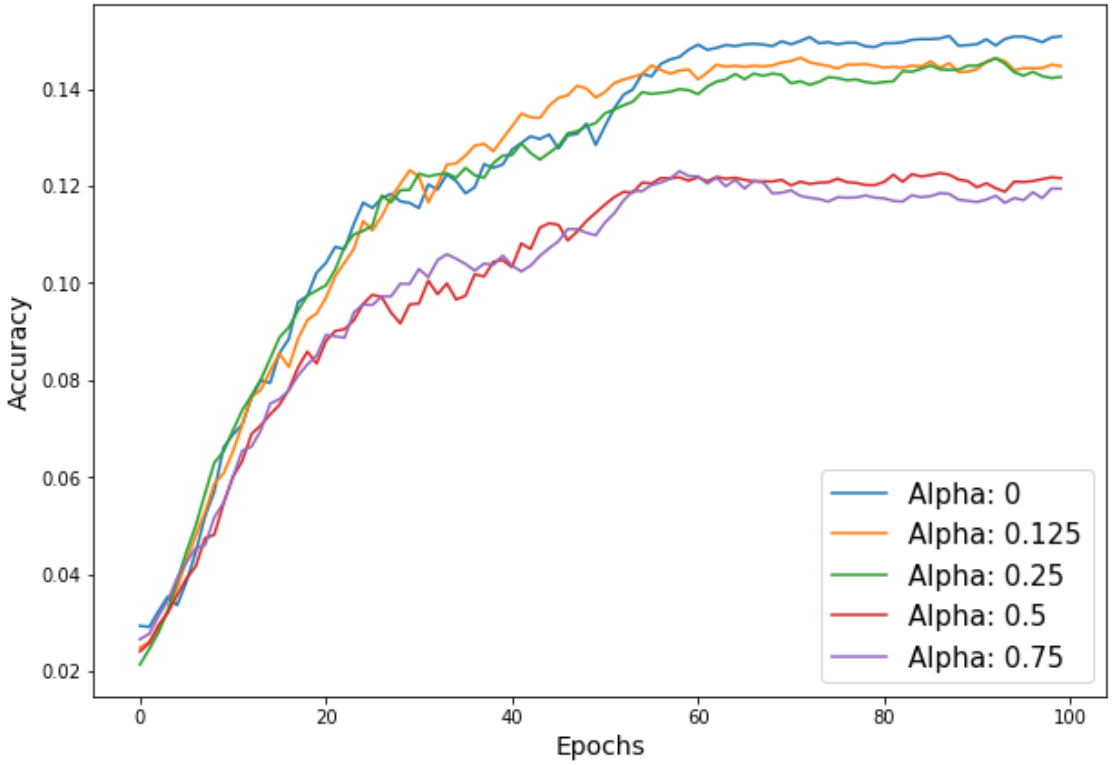


Figure 5: Test accuracy with 7-layers CNN (smoothed curve)

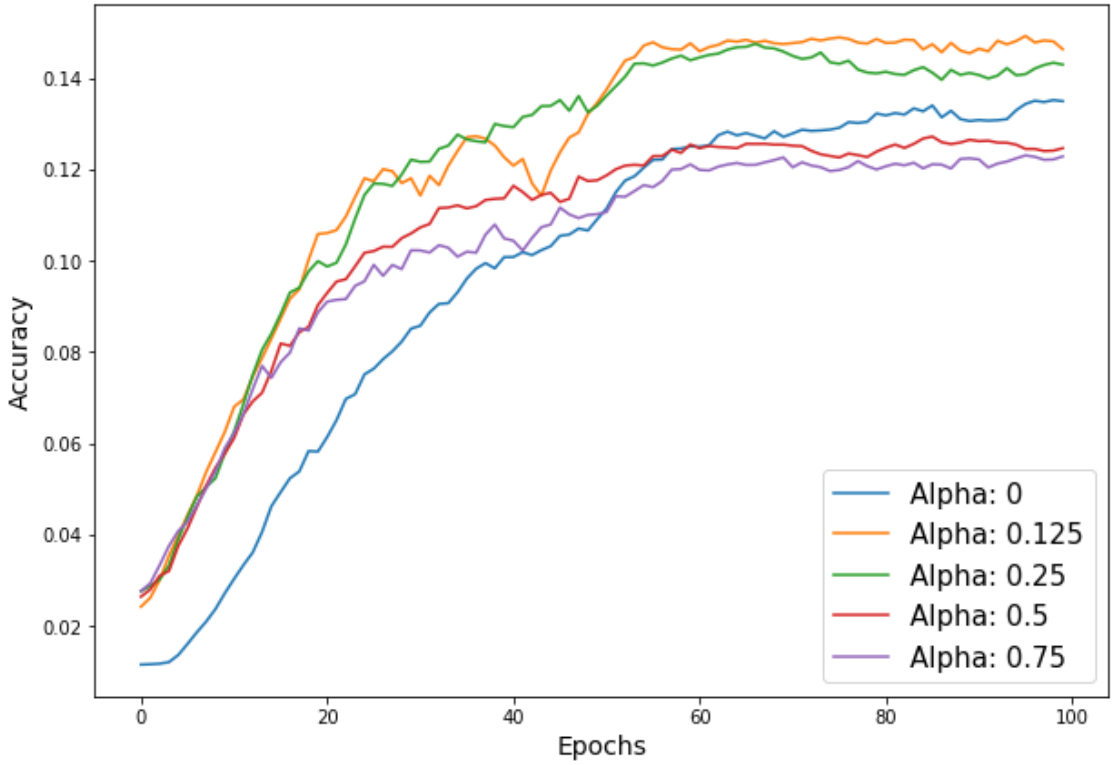


Figure 6: Test accuracy with 9-layers CNN (smoothed curve)

Résultats Stanford Dogs Dataset

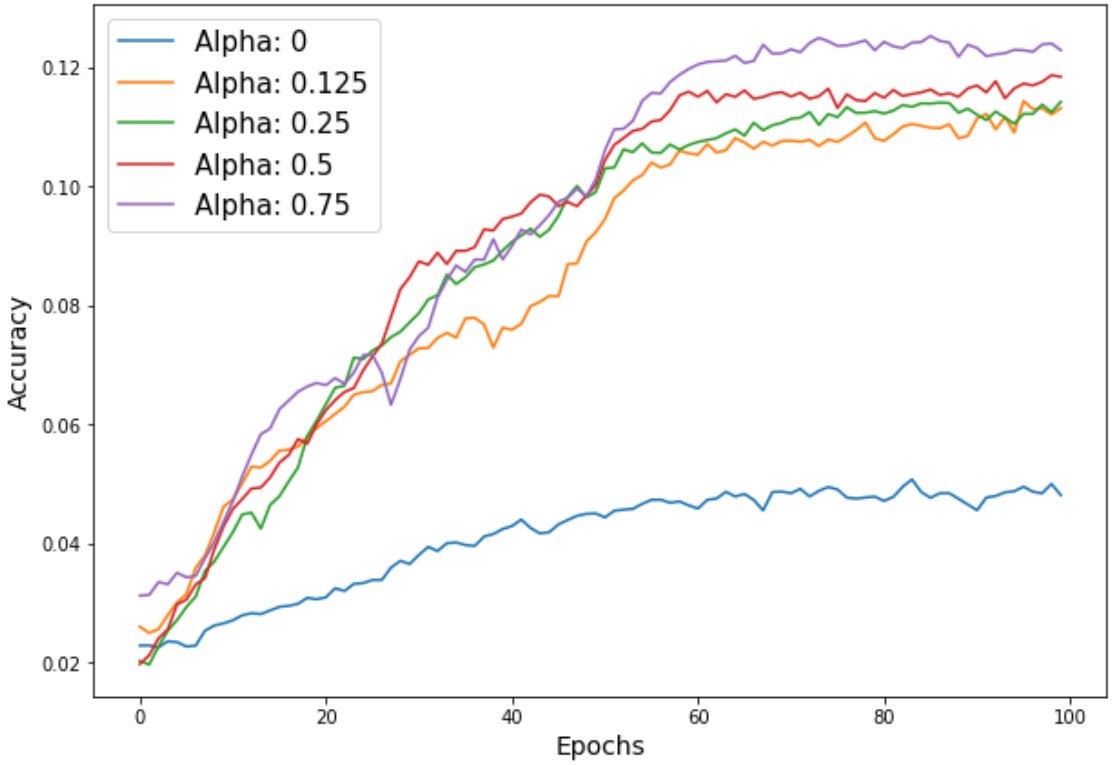


Figure 7: Test accuracy with NiN (smoothed curve)

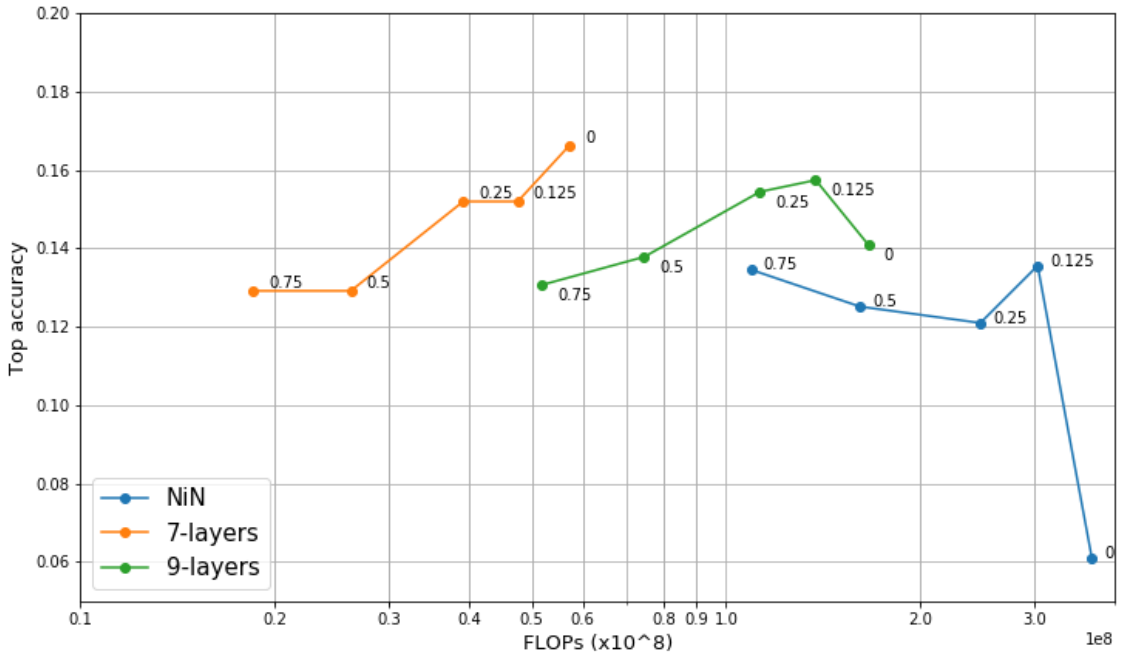


Figure 8: Accuracy-FLOPs trade-off curve on Stanford Dogs Dataset

Observations

Influence de l'architecture:

- Plus il y a de couches de convolution, plus la nouvelle convolution est efficace:
  - Avec NiN (9 couches de convolution): gain important de performances
  - Avec le CNN 9 couches (dont 6 de convolution): gain modéré sur le Dogs dataset
  - Avec le CNN 7 couches (dont 5 de convolution): pas de gain

Temps d'exécution:

- Pas de différences avec le CNN 7 ou 9 couches
- Avec NiN:

Alpha	Execution time (s) 1 epoch - CIFAR-10	Execution time (s) 1 epoch – Dogs Dataset
0	102	12
0.125	129	21
0.25	115	19
0.5	84	14
0.75	63	10

Conclusions

- Réduction mémoire et coût de calcul intéressante
- Résultats sur la performance mitigés:
  - Dépendant de l'architecture et/ou du dataset
- Avantages:
  - Côté générique, peut être testé dans toutes les architectures
- Perspectives:
  - Aller plus loin dans le multi-échelles avec plus que 2 octaves