UNIVERSITÉ LAVAL

GLO-4030/7030
APPRENTISSAGE PAR RÉSEAUX DE NEURONES PROFONDS

CNN (Partie II) :
Exemples d'architecture

Hiver 2020

Philippe Giguère

Admin

- Présentations orales ce vendredi

Alex Sirois	Taskonomy: Disentangling Task Transfer Learning
Xavier-Charles Lebel	A Unified Theory of Early Visual Representations from Retina to Cortex through Anatomically Constrained Deep CNNs
Léandre Gagnon-Lewis	Interpreting CNNs via Decision Trees
Marc-André Ménard	ACTIVE LEARNING WITH PARTIAL FEEDBACK
Tianyang Deng	A convolutional neural network regression for quantifying cyanobacteria using hyperspectral imagery

Local : PLT-1120

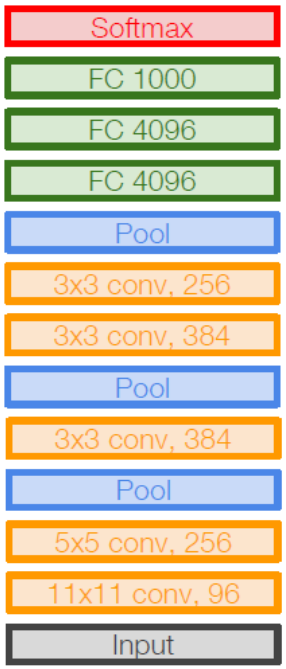
9 AM

- Quelques questions de pratique (surtout CNN)
- Projet de reconnaissance de cerfs (biologie)

Résumé CNN I

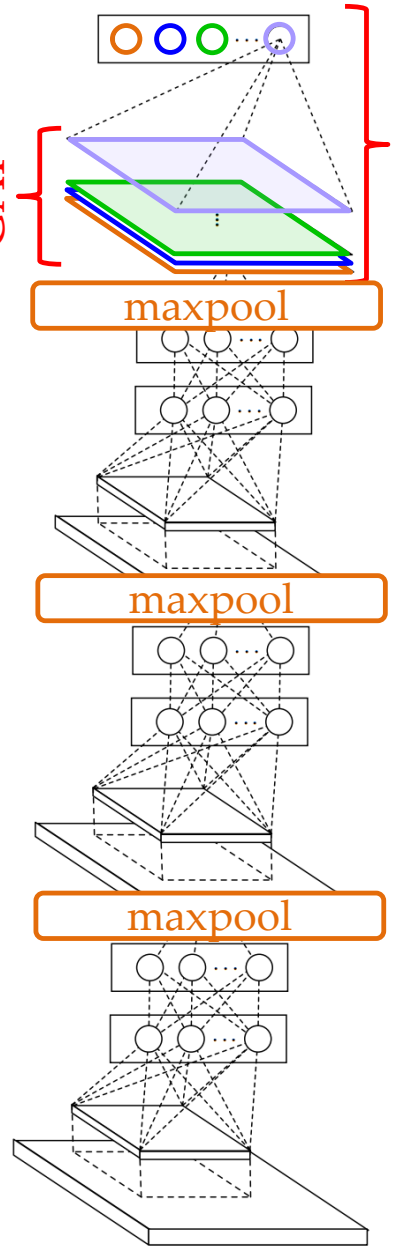
- Plus grande profondeur
- Disparition graduelle des têtes fully-connected
  - remplacé par Global Average Pooling + 1 layer de fully-connected
- conv 3x3 est la taille dominante
- Parfois conv plus grande que 3x3 à la base
  - 5x5 ou 7x7

Résumé des architectures CNN I

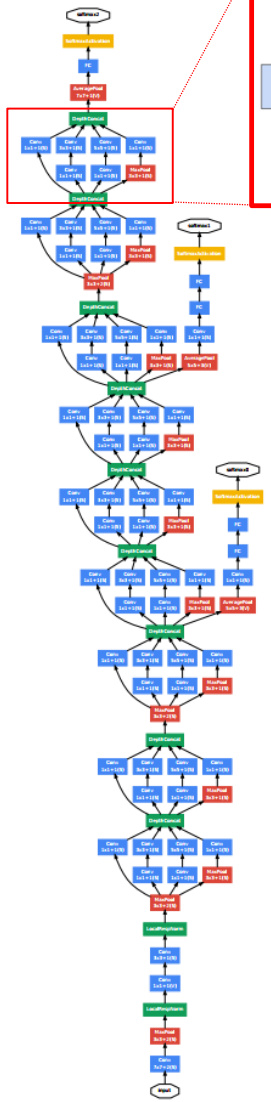


AlexNet

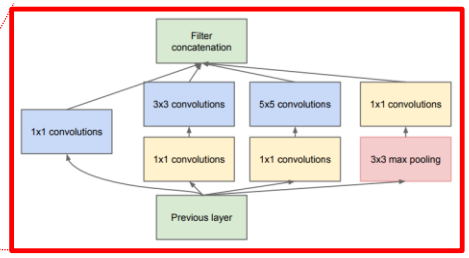
VGG



NiN



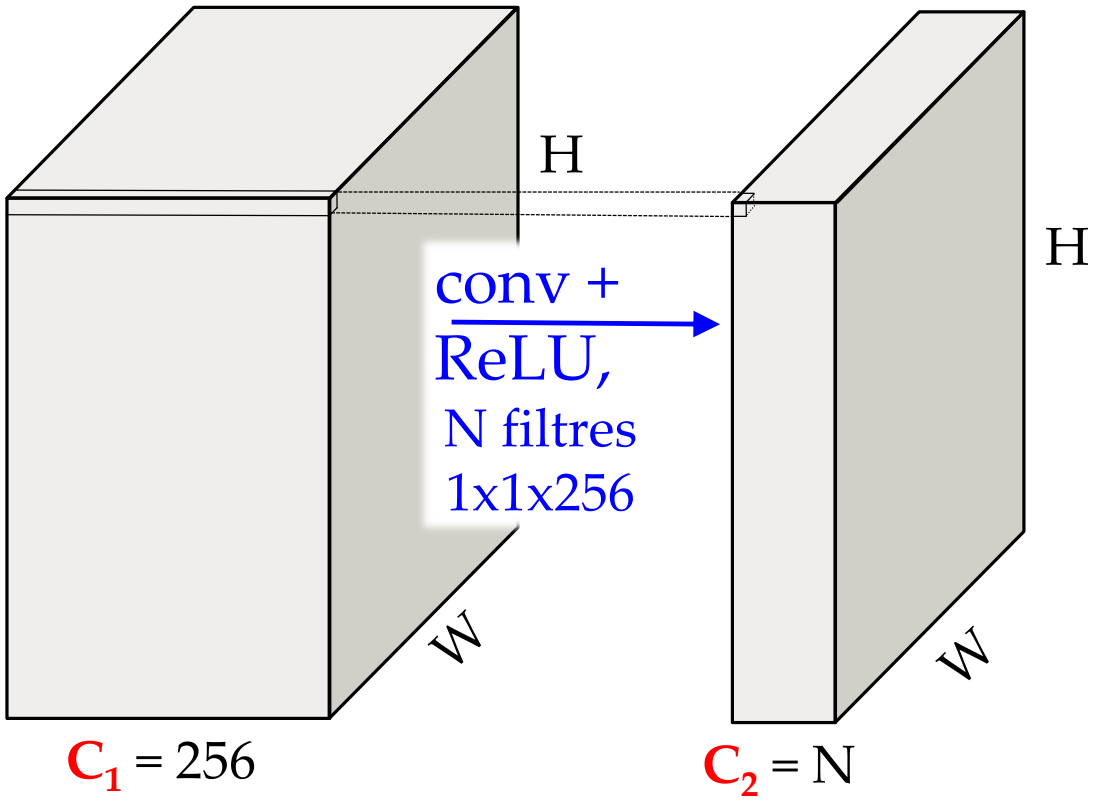
GoogLeNet



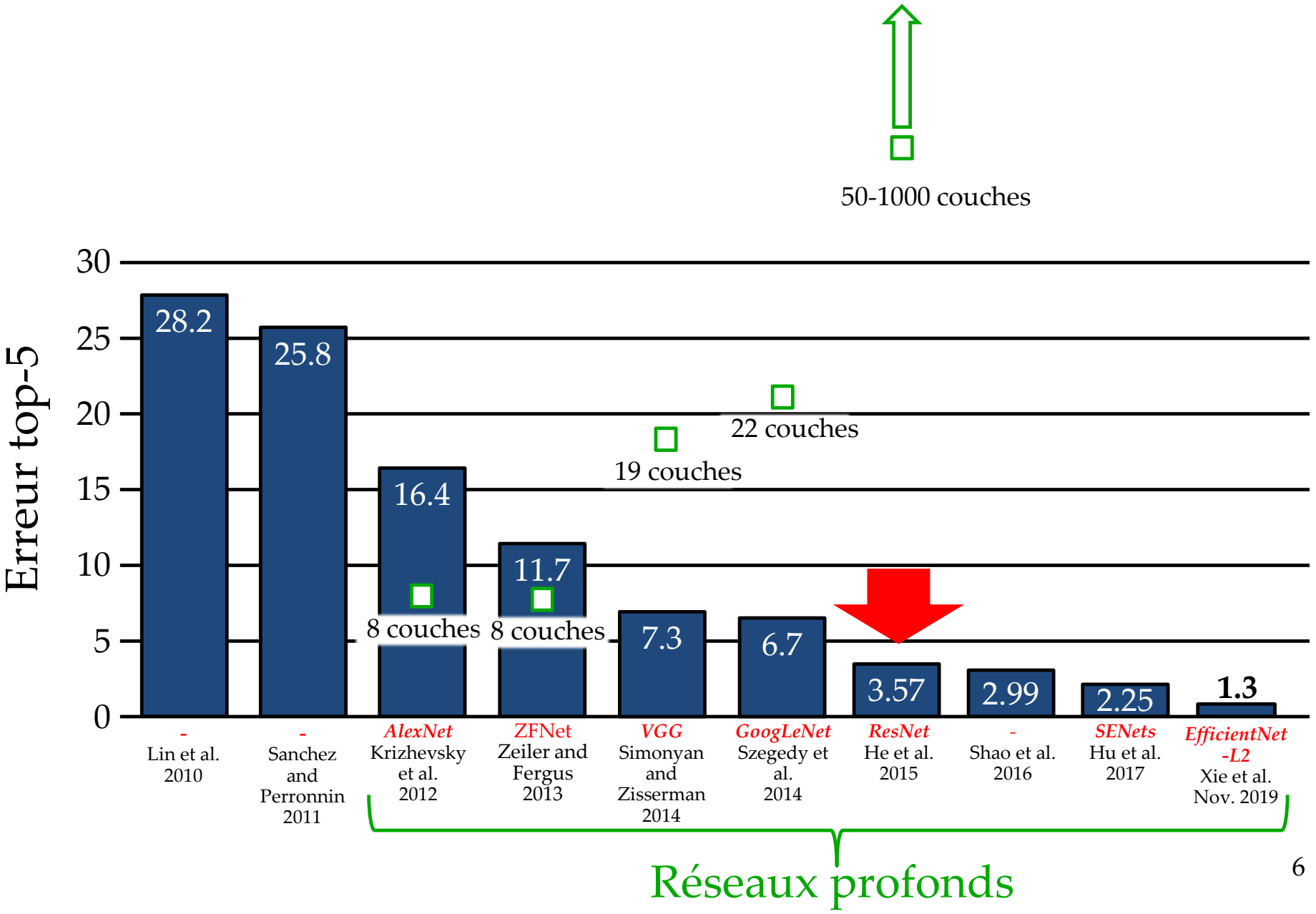
Module Inception (multi-branche)

Résumé CNN I

- conv 1x1 sont des réseaux *fully-connected*
- Servent à réduire la dimensionnalité des *features*

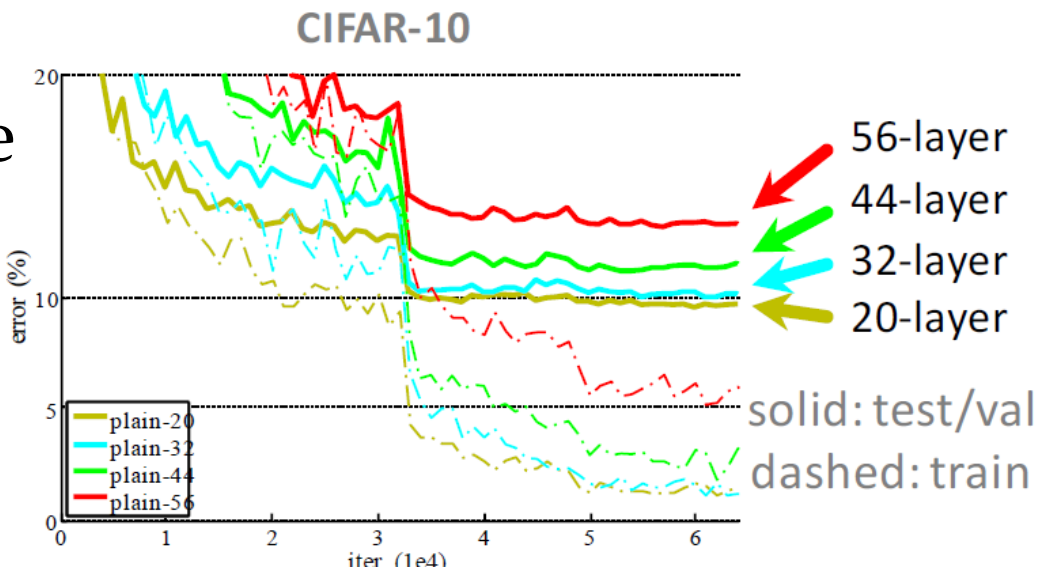


Large Scale Visual Recognition Challenge



ResNet

- Vise l'entraînement de réseaux très profonds (30+ couches)
- Problème du *vanishing gradient* en partie réglé par la Batch Norm
- Constat : dégradation des résultats passé une certaine profondeur



ICML tutorial 2016, He.

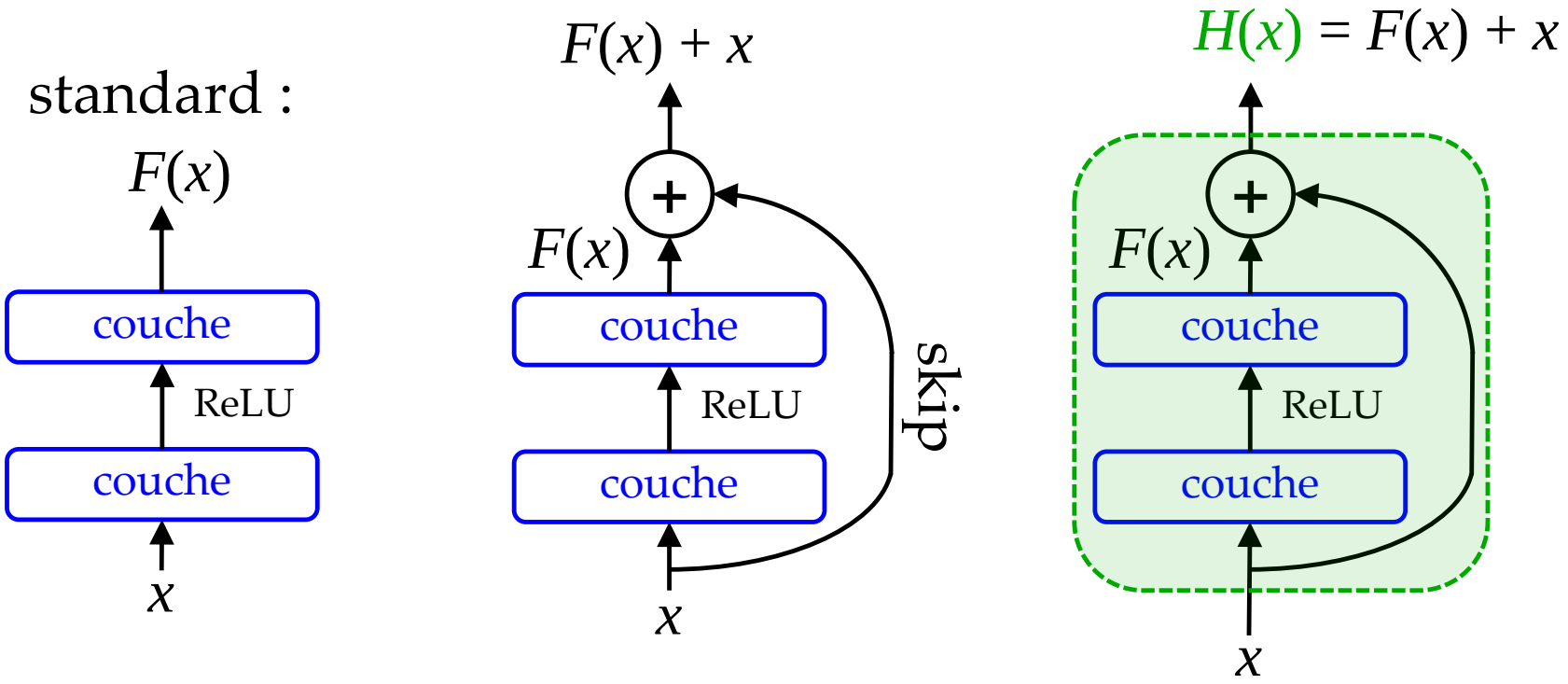
- Intuition : un réseau devrait simplement apprendre la fonction identité
  - mais l'optimisation n'y arrive pas

ResNet

- Dilution de l'information de correction (gradient)
- Difficile pour une couche de réutiliser des *features* des couches précédentes
  - perte de l'*information flow* [1]
- Difficulté d'apprendre la fonction identitaire

[1] Srivastava, R.K., Greff, K., Schmidhuber, J.: Highway networks. arXiv preprint arXiv:1505.00387 (2015)

ResNet : idée de base

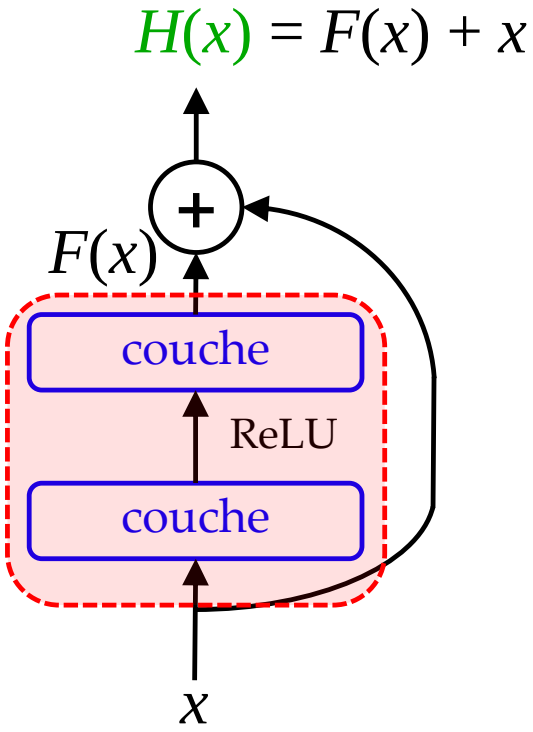


$F(x)$ est le résiduel entre la fonction $H(x)$ désirée et la fonction identitaire : $F(x) = H(x) - x$

- N'ajoute aucun paramètre au réseau, très peu de calcul
- Doit avoir au moins deux couches internes

ResNet : Question

- Quelle doit-être la taille du tenseur de sortie du **bloc résiduel** $F(x)$, si la taille du tenseur d'entrée x est de HxWxC ?



- Réponse : HxWxC

(les deux tenseurs doivent avoir la même taille, pour pouvoir s'additionner)

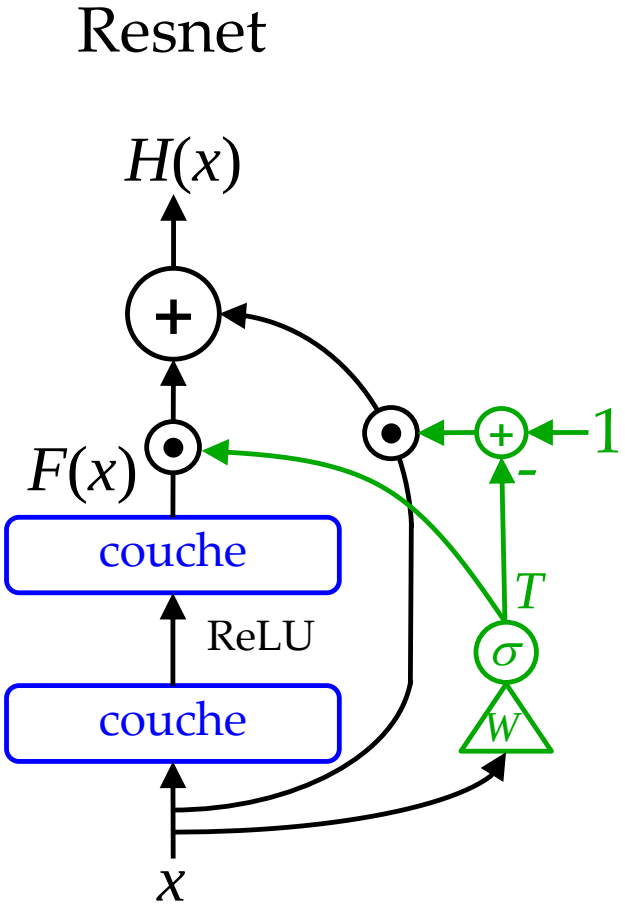
Highway network

- Compétiteur contemporain des ResNet
- Va utiliser du *gating* pour choisir le mélange résiduel vs. identité

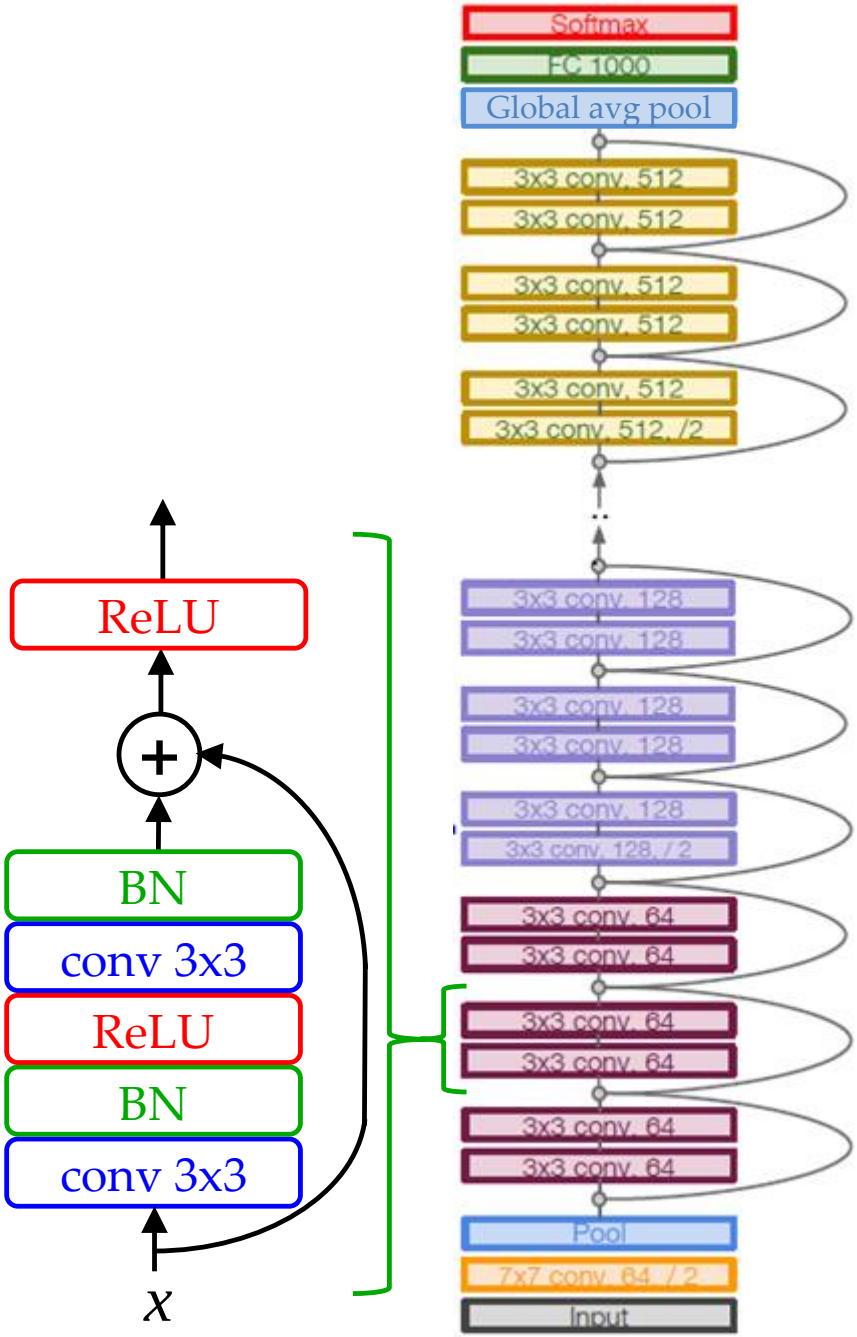
$$H(x) = F(x)T + x(1-T)$$

$$T(\mathbf{x}) = \sigma(\mathbf{W_T}^T\mathbf{x} + \mathbf{b_T})$$

- ResNet a fait le pari qu'il est toujours mieux de faire la somme des deux : architecture plus simple



ResNet



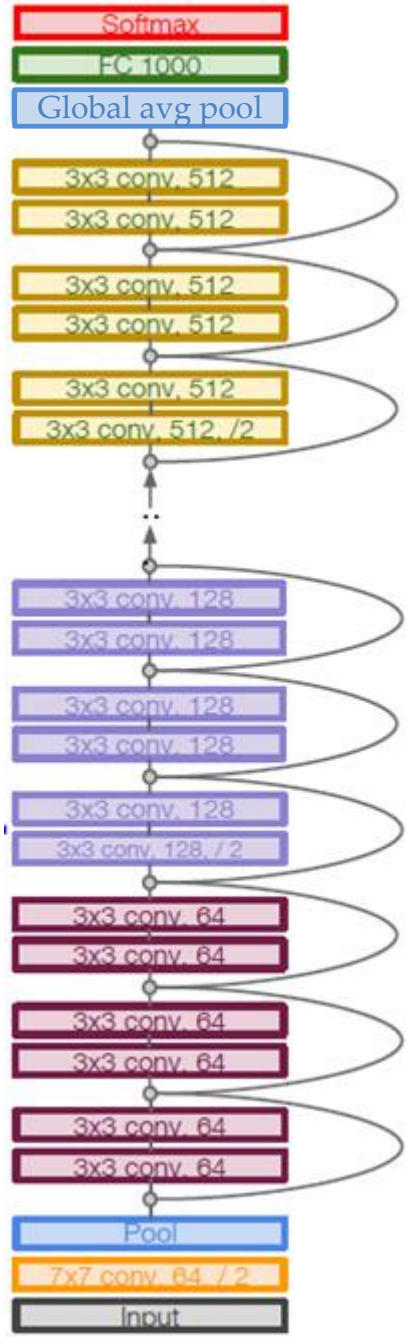
adapté de : cs231n, Université Stanford

ResNet

- Apprentissage du résiduel :
  - plus facile (car cela peut être que des petits ajustements)
  - $F(x)$ initialisé avec des petites valeurs
  - pourra facilement apprendre des mapping identitaires

Notes :

- architecture simple de conv 3x3, style VGG
- convolution 7x7 avec stride 2 à la base
- double le nombre de filtre après chaque réduction de taille du feature map
- position des ReLU : sera différente pour version « identity mapping »
- pas de dropout



adapté de : cs231n, Université Stanford

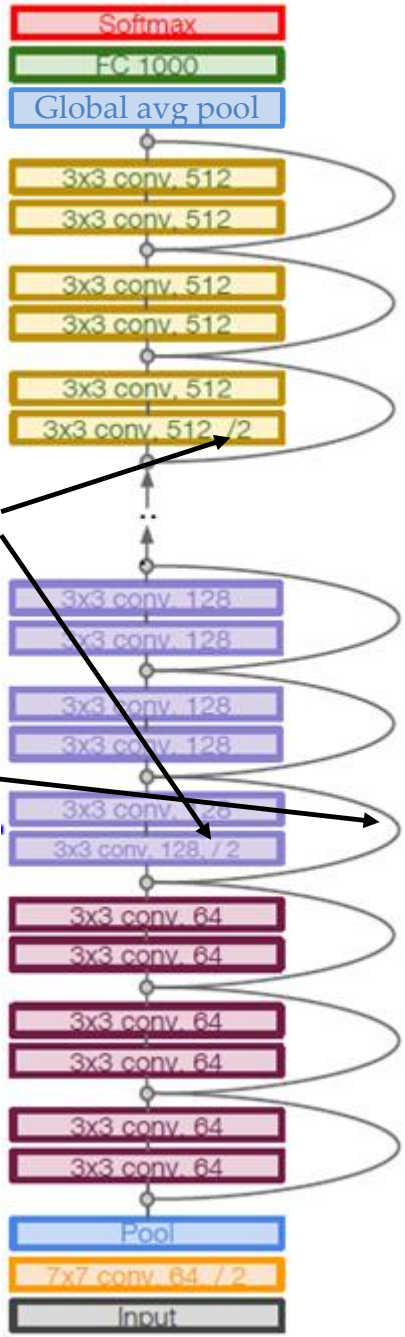
ResNet

Global Average Pooling

Downsampling se fait par des conv avec pas=2

(1x1 conv)

Si besoin d'ajuster les dimensions : $y = F(x) + W_s x$
des tenseurs



Dans certains cas, besoin de warm up

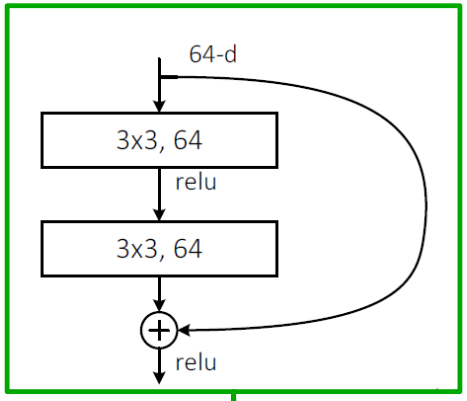
"…we find that the initial learning rate of 0.1 is slightly too large to start converging. So we use 0.01 to warm up the training until the training error is below 80% (about 400 iterations), and then go back to 0.1 and continue training."

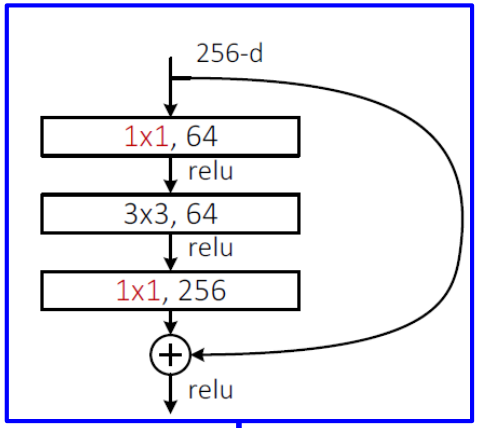
adapté de : cs231n, Université Stanford

ResNet

Bottleneck

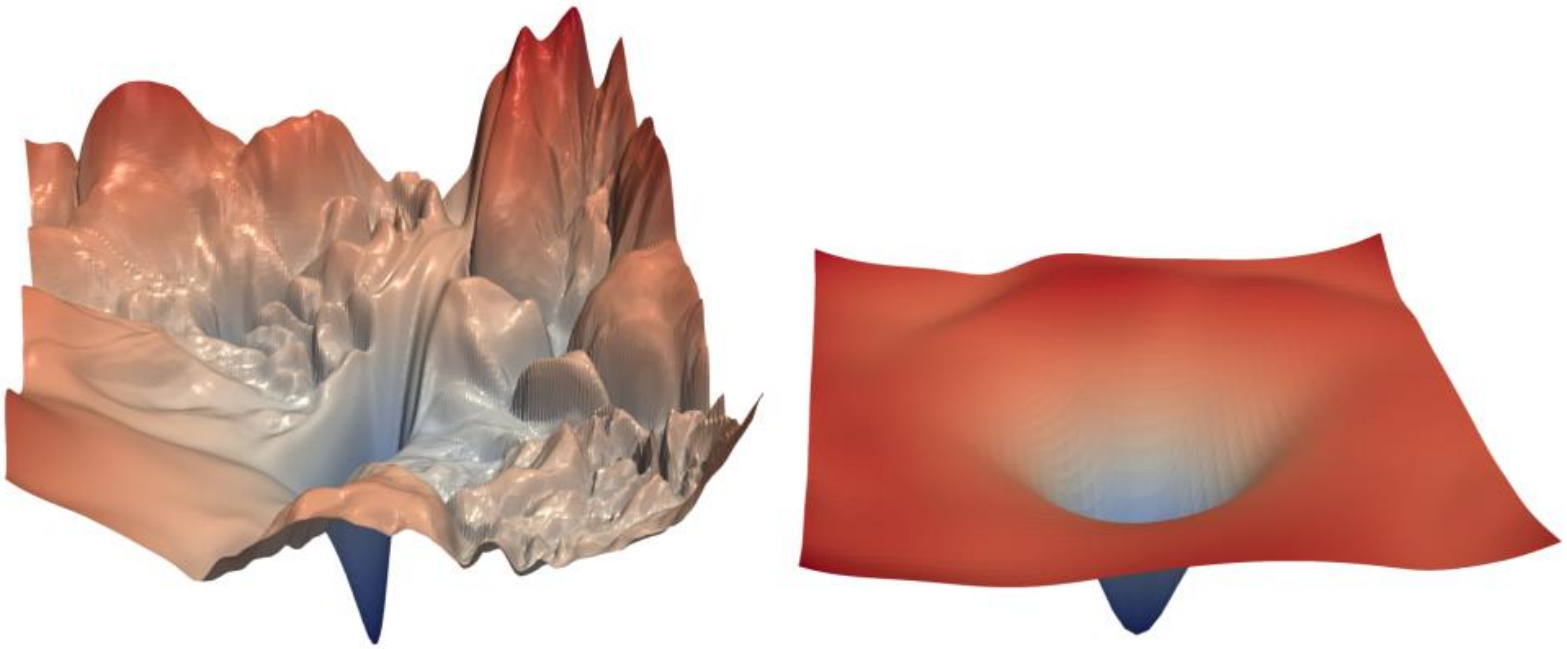
(pour efficacité, pas pour régularisation)





layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

ResNet : lissage de la fonction de coût



(a) without skip connections

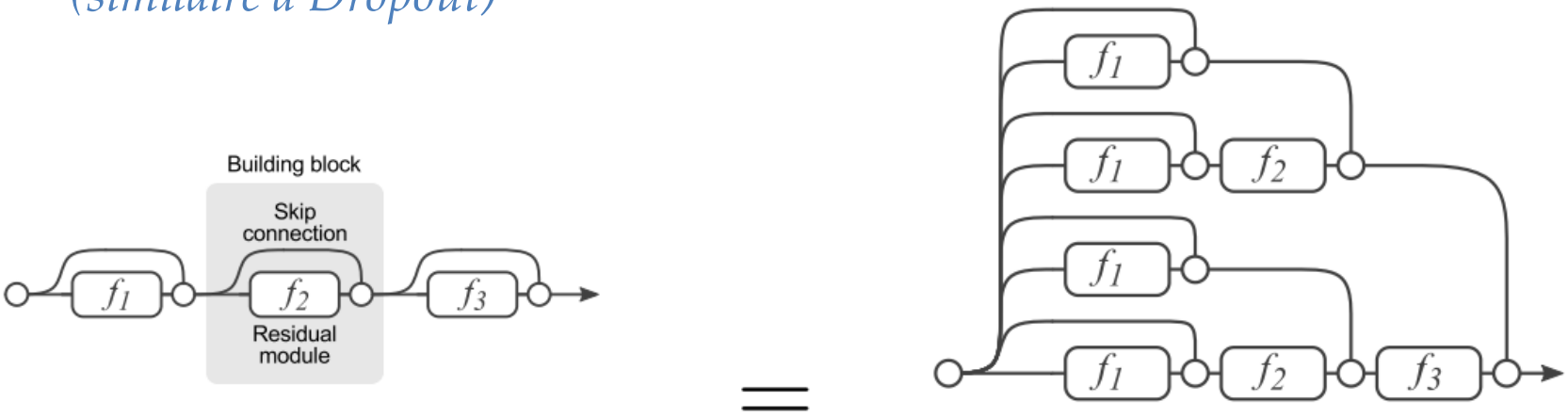
(b) with skip connections

Figure 2.10: The difference between the loss function landscape of a network without skip connections (a) and a network with skip connections (b). The use of skip connections makes the loss function smoother, which makes the search for a good minimizer easier. Figure taken from Li et al. (2017).

H. Li, Z. Xu, G. Taylor, and T. Goldstein. Visualizing the loss landscape of neural nets. http://arxiv.org/abs/1712.09913, 2017.

ResNet : ensemble implicite

- Ensemble exponentiel 2^L de réseaux
  (similaire à Dropout)



(a) Conventional 3-block residual network

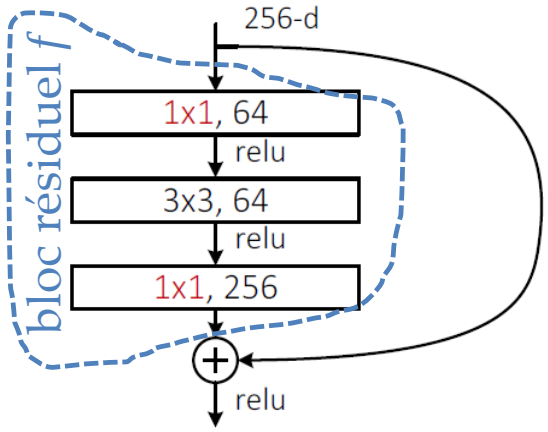
(b) Unraveled view of (a)

- Gradient est atténué dans le résiduel :
  - profondeur effective du gradient est de 10-34 couches (sur 110)

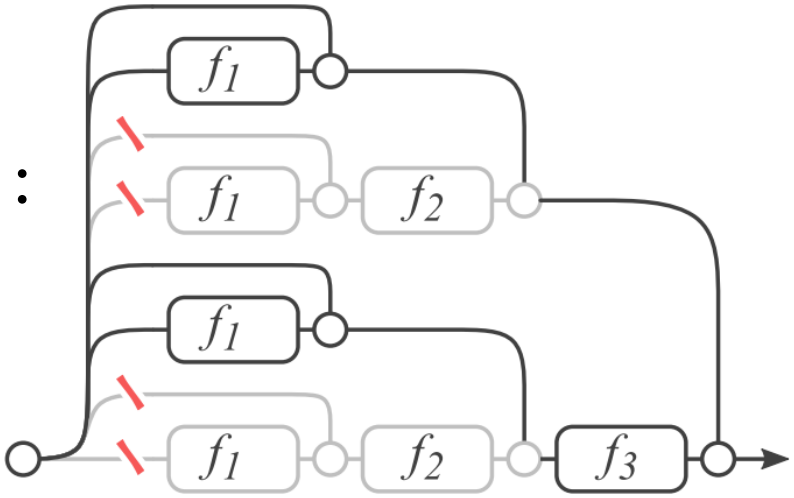
A. Veit, M. Wilber and S. Belongie. Residual Networks Behave Like Ensembles of Relatively Shallow Networks. arXiv:1605.06431v2,2016.

ResNet : ensemble implicite

- Si l'on retire un bloc résiduel ResNet, on élimine un nombre de sous-réseaux
- Il reste encore 2^{L-1} sous-réseaux



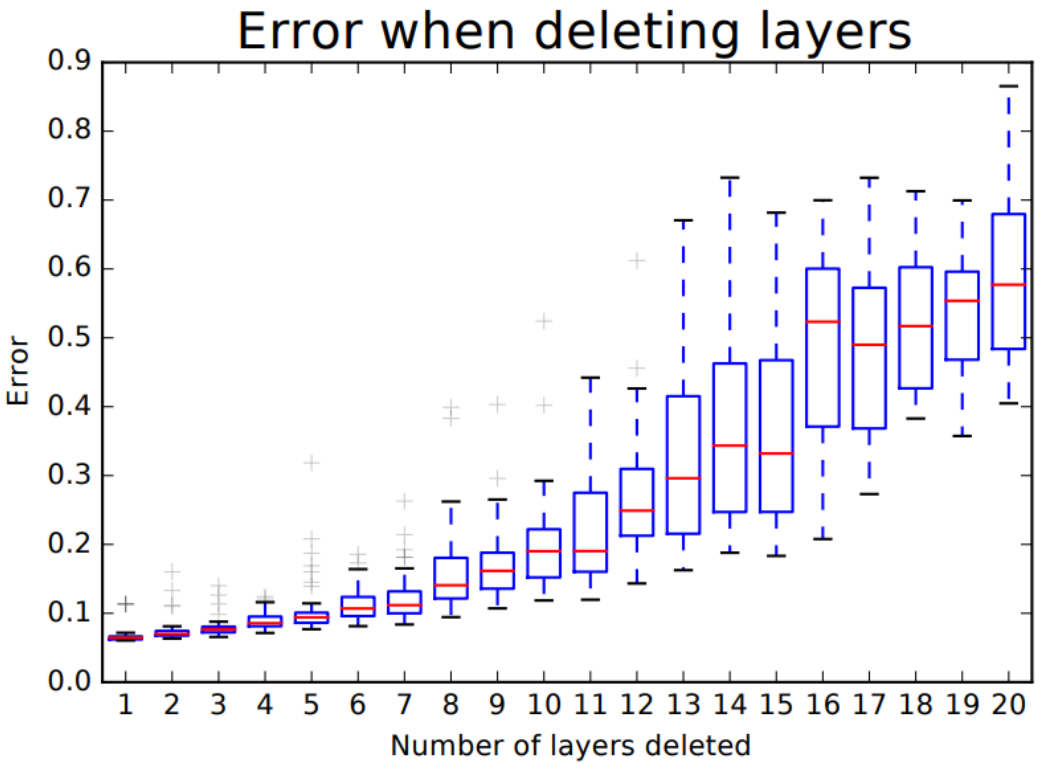
On retire f_2 :



A. Veit, M. Wilber and S. Belongie. Residual Networks Behave Like Ensembles of Relatively Shallow Networks. arXiv:1605.06431v2,2016.

ResNet : ensemble implicite

- Retirer des couches en testing pour ResNet
- Ensemble implicite semble jouer un rôle plus important que la profondeur absolue

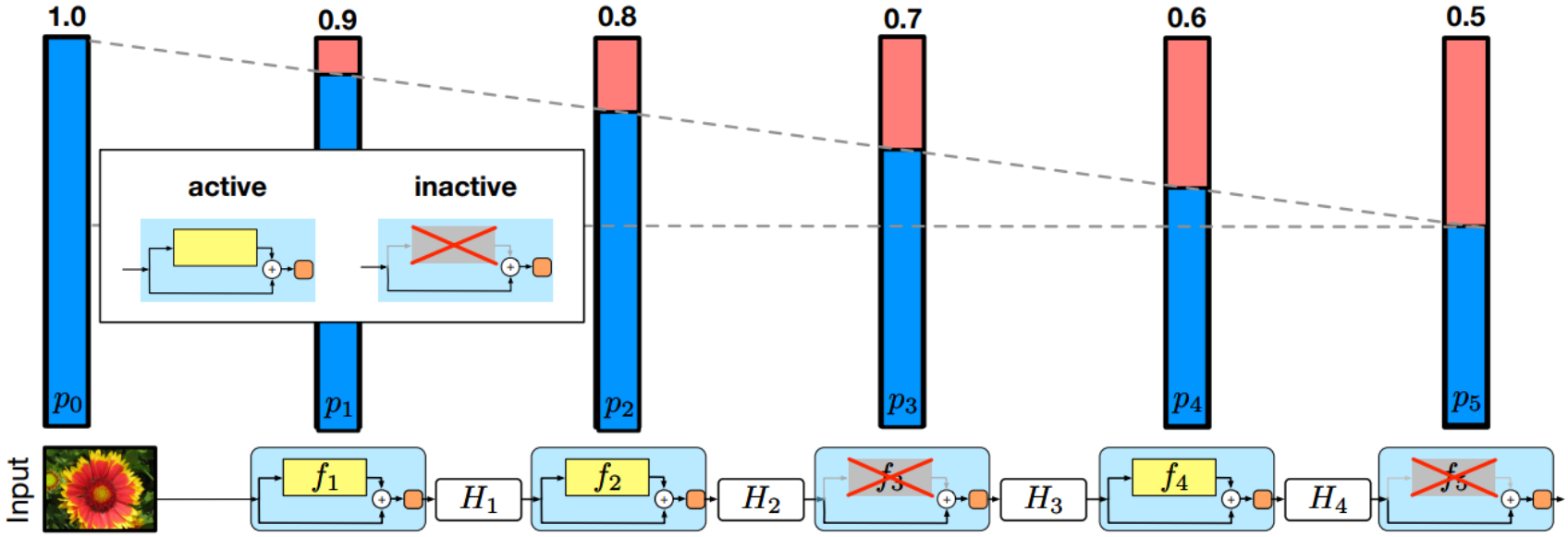


- Retrait de couche : catastrophique pour VGG!

A. Veit, M. Wilber and S. Belongie. Residual Networks Behave Like Ensembles of Relatively Shallow Networks. arXiv:1605.06431v2,2016.

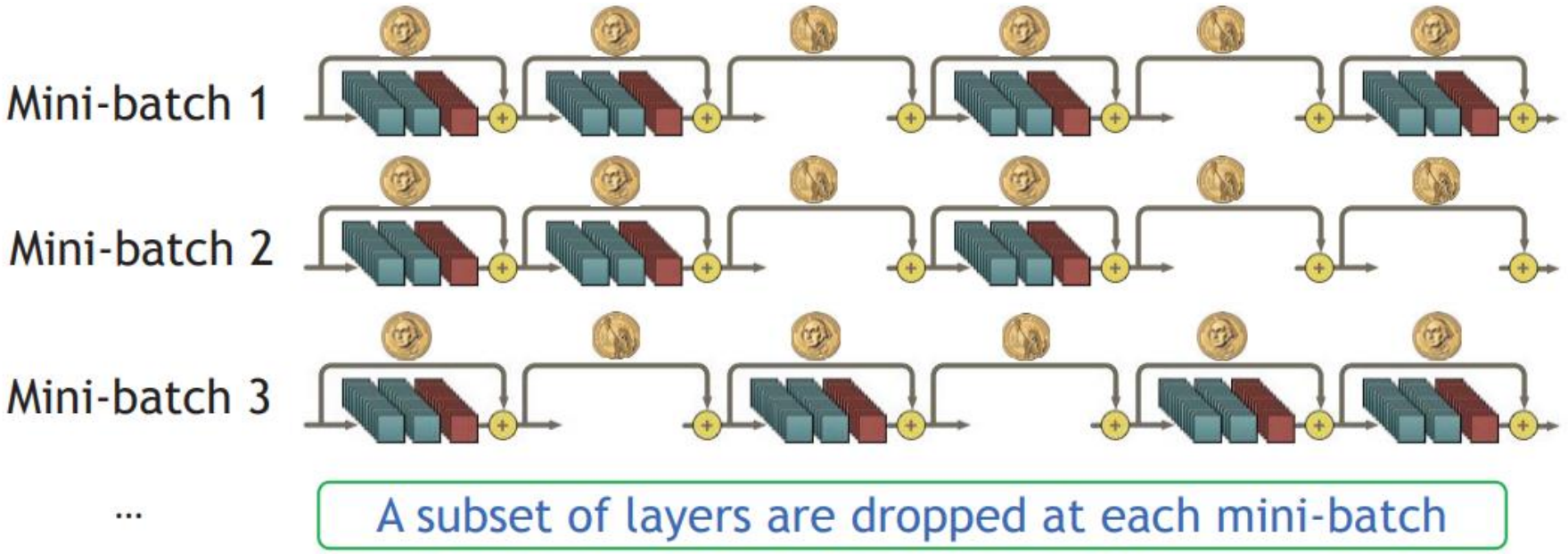
Stochastic Depth

- Aléatoirement retirer des blocs résiduels lors du training
  - dropout empiriquement inutile selon eux
- Conserver plus souvent les blocs résiduels de bas niveau



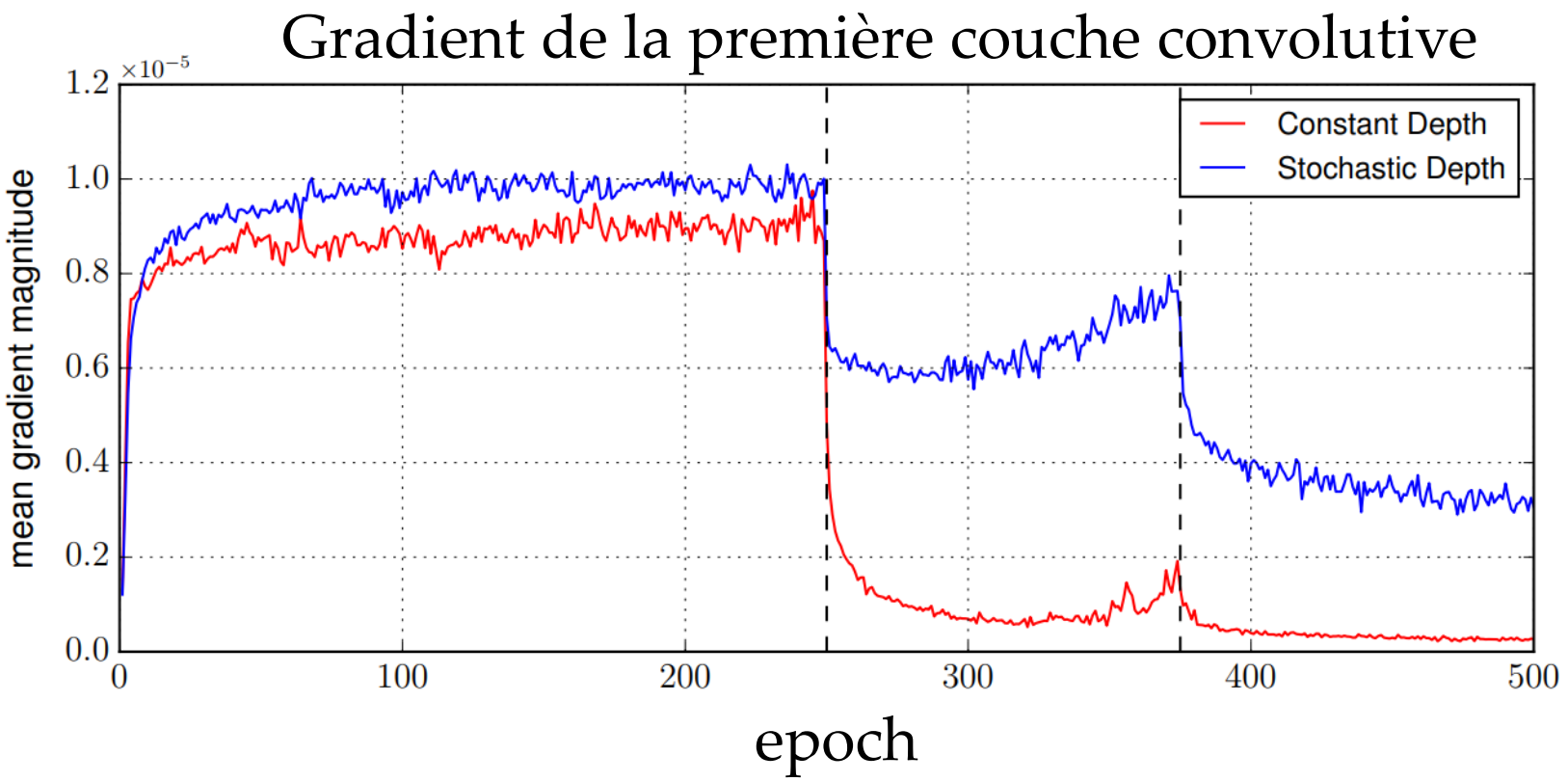
Huang et al. Deep Network with Stochastic Depth, 2016.

Stochastic Depth



Stochastic Depth

- Améliore le flot du gradient, en réduisant le nombre de couches



Huang et al. Deep Network with Stochastic Depth, 2016.

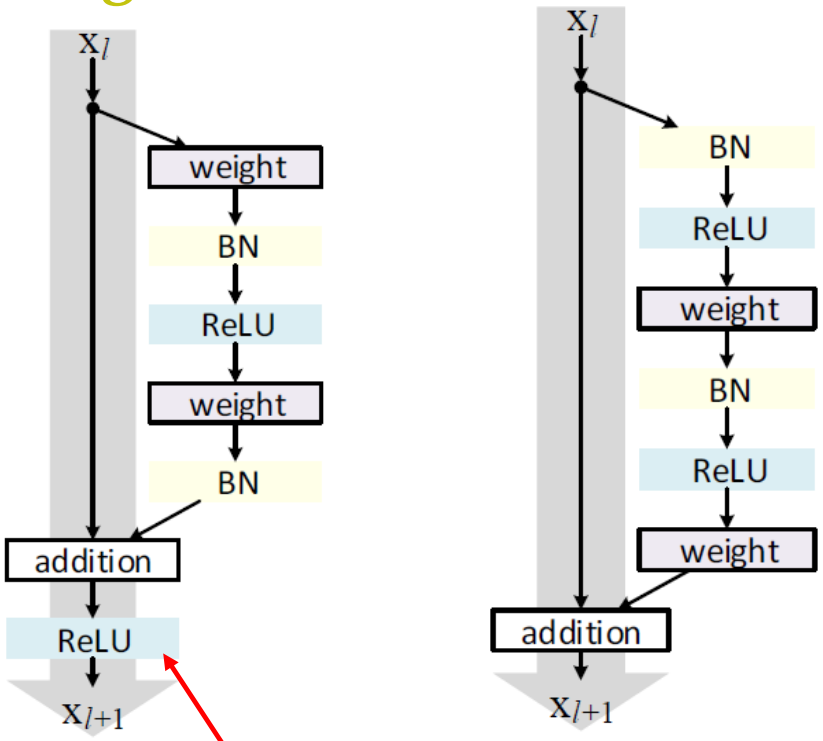
Stochastic Depth

- Accélère l'entraînement :
  - réseaux moins profonds sont plus rapides à entraîner
  - 25% moins de calcul (si décroissance linéaire 1→0.5 pour probabilité p_l droper le bloc l)
- Forme de régularisation
- Utilise le plein réseau en test
  - comme avec dropout
  - calibrer les forces des features en fonction de p_l

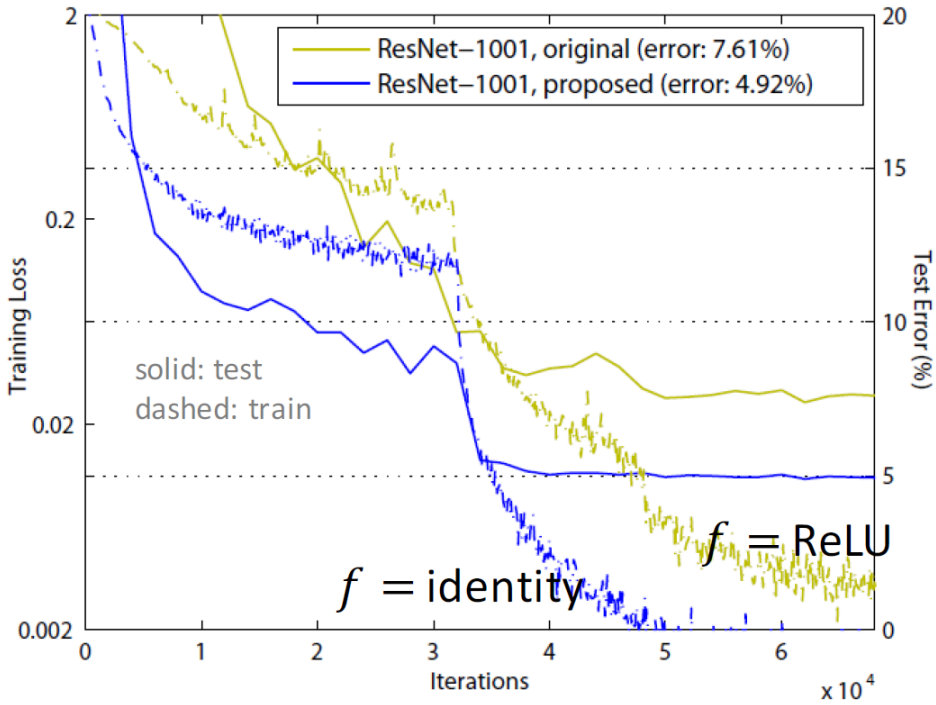
Huang et al. Deep Network with Stochastic Depth, 2016.

ResNet version preactivation

Original **«pre-activation»**



ReLU dans le chemin du gradient



tiré : ICML workshop 2016, He.

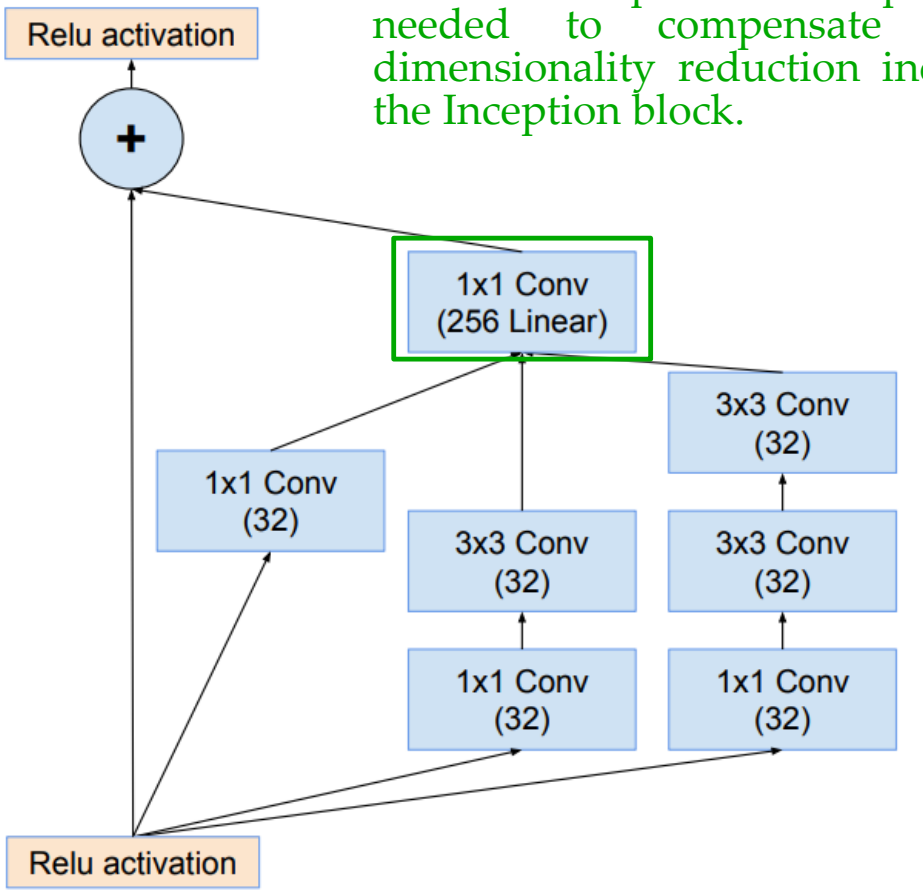
- Bloc résiduel amélioré
- Meilleur flot du gradient
- Améliore les résultats
  - 6.7% → 4.8% top-5

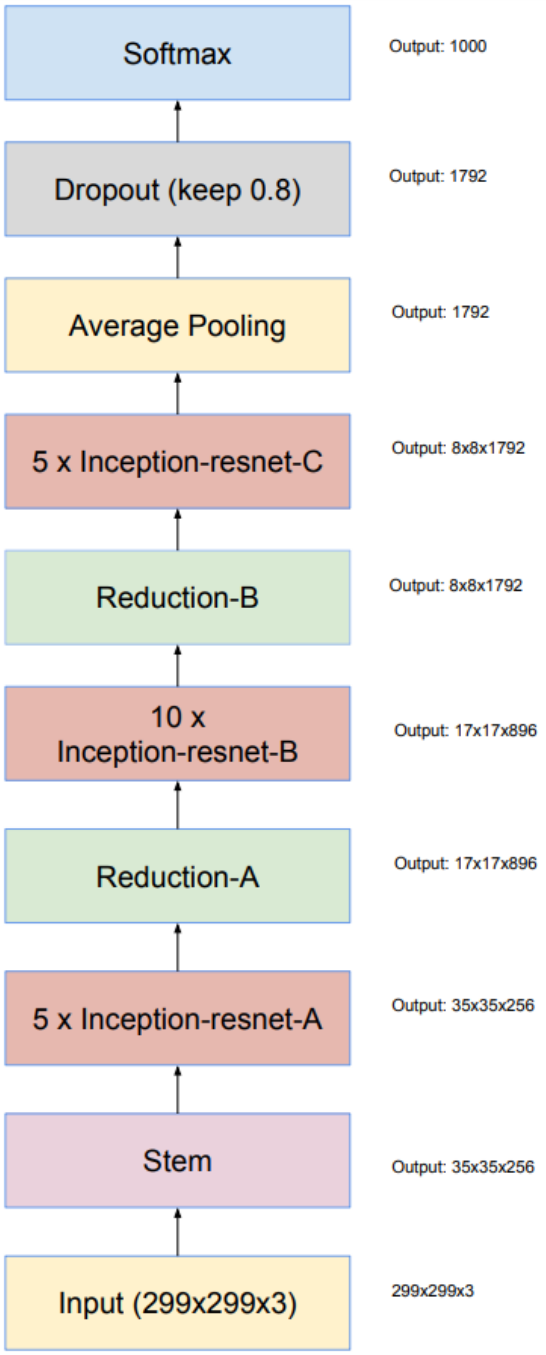
He et al., Identity Mappings in Deep Residual Networks, ECCV 2016.

Variations de ResNet

Inception-ResNet v1

Each Inception block is followed by filter-expansion layer (1×1 convolution without activation) which is used for scaling up the dimensionality of the filter bank before the addition to match the depth of the input. This is needed to compensate for the dimensionality reduction induced by the Inception block.





ResNeXt

- Architecture multi-branche (comme Inception), mais répète la même topologie (contrairement à Inception)
- Le nombre de branches == *cardinality*
- Prétend qu'il est mieux d'augmenter la *cardinality* que la profondeur[1]/largeur[2]
- Cherche à améliorer la performance sous un budget fixe de FLOPS et de paramètres
- (Visitera ce sujet avec EfficientNet)

[1]nombre couches

[2]nombre canaux/filtres

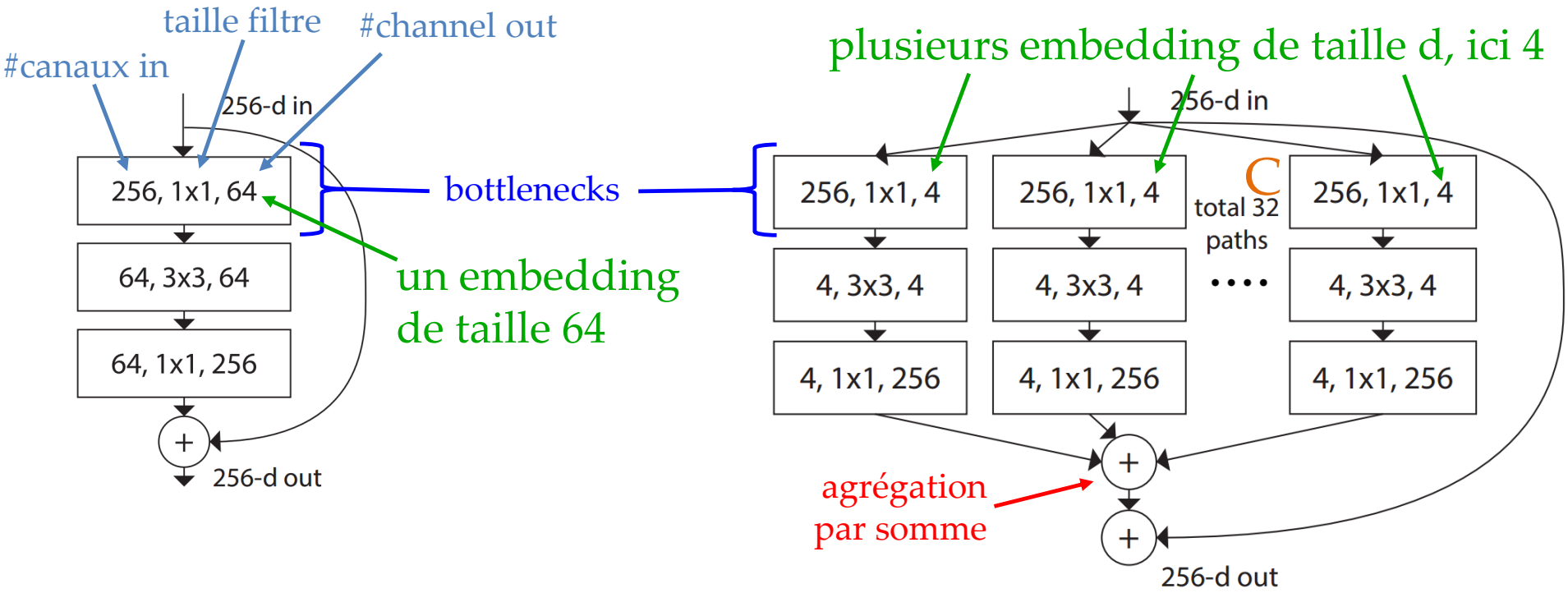
S. Xie, R. Girshick, P. Dollar, Z. Tu and K. He. Aggregated Residual Transformations for Deep Neural Networks. arXiv preprint arXiv:1611.05431v1,2016.

ResNeXt

- ResNeXt va empruter la philosophie *split-transform-merge* d'Inception
- Le nombre de filtres 1x1, 3x3, 5x5 dans ResNeXt varie d'une couche à l'autre, difficile à tuner ☹
- On peut voir Inception comme étant un sous-espace d'une couche ne contenant que des 5x5

Cardinality C=1

Cardinality C=32



cardinality C	1	2	4	8	32
width of bottleneck d	64	40	24	14	4

Table 2. Relations between cardinality and width (for the template of conv2), with roughly preserved complexity on a residual block.

ResNeXt

diminution/ augmentation au même rythme

stage	output	ResNet-50	**ResNeXt-50 (32×4d)**
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2 $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$ (répétitions)	3×3 max pool, stride 2 $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C{=}32 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C{=}32 \\ 1\times1, 512 \end{bmatrix} \times 4$
conv4	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C{=}32 \\ 1\times1, 1024 \end{bmatrix} \times 6$
conv5	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C{=}32 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		**25.5**×10^6	**25.0**×10^6
FLOPs		**4.1**×10^9	**4.2**×10^9

ResNeXt : Résultats

cardinality

bottleneck

		setting	top-1 error (%)
capacité similaire	ResNet-50	1 × 64d	23.9
	ResNeXt-50	2 × 40d	23.0
	ResNeXt-50	4 × 24d	22.6
	ResNeXt-50	8 × 14d	22.3
	ResNeXt-50	32 × 4d	**22.2**
capacité similaire	ResNet-101	1 × 64d	22.0
	ResNeXt-101	2 × 40d	21.7
	ResNeXt-101	4 × 24d	21.4
	ResNeXt-101	8 × 14d	21.3
	ResNeXt-101	32 × 4d	**21.2**

Table 3. Ablation experiments on ImageNet-1K. (**Top**): ResNet-50 with preserved complexity (∼4.1 billion FLOPs); (**Bottom**): ResNet-101 with preserved complexity (∼7.8 billion FLOPs). The error rate is evaluated on the single crop of 224×224 pixels.

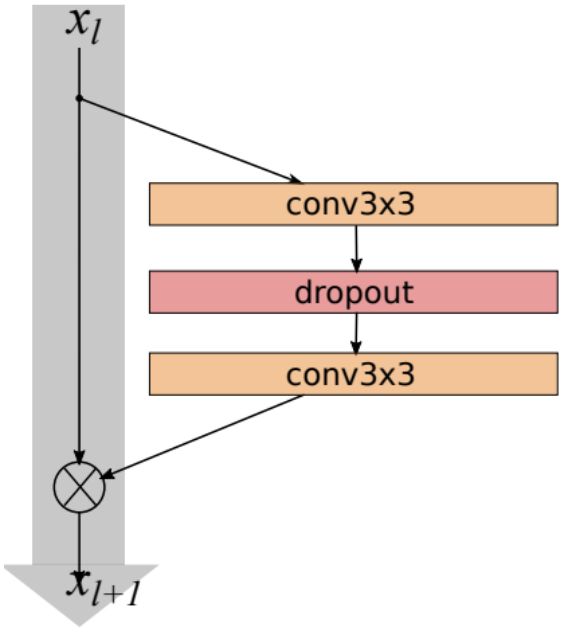
Si on double la capacité

	setting	top-1 err (%)	top-5 err (%)
1× complexity references:			
ResNet-101	1 × 64d	22.0	6.0
ResNeXt-101	32 × 4d	21.2	5.6
2× complexity models follow:			
ResNet-**200** [15]	1 × 64d	21.7	5.8
ResNet-101, wider	1 × **100**d	21.3	5.7
ResNeXt-101	**2** × 64d	20.7	5.5
ResNeXt-101	**64** × 4d	**20.4**	**5.3**

Table 4. Comparisons on ImageNet-1K when the number of FLOPs is increased to 2× of ResNet-101's. The error rate is evaluated on the single crop of 224×224 pixels. The highlighted factors are the factors that increase complexity.

Wide Residual Network

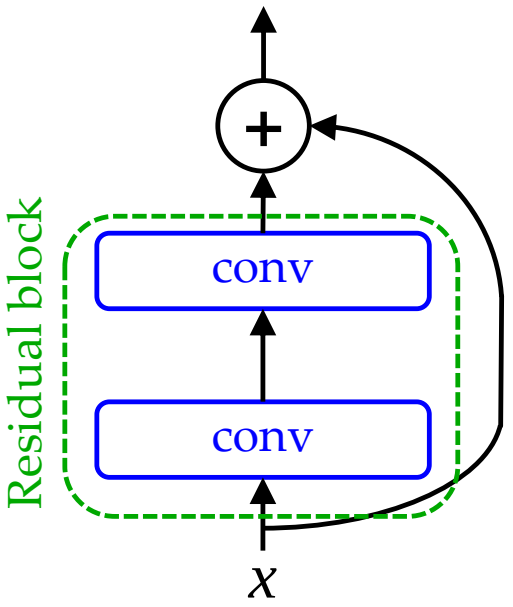
- Prétend qu'il vaut mieux avoir des réseaux plus large (# filtres par couche) que profond
- WRN de 16 couches > ResNet 1000 couches
- Plus rapide à entraîner
  - large est plus facile pour GPU que profond (séquentiel)
- Trouve où ajouter le dropout
  - entre les couches de convolution



[Zagoruyko et al., Wide Residual Network, BMVC, 2016]

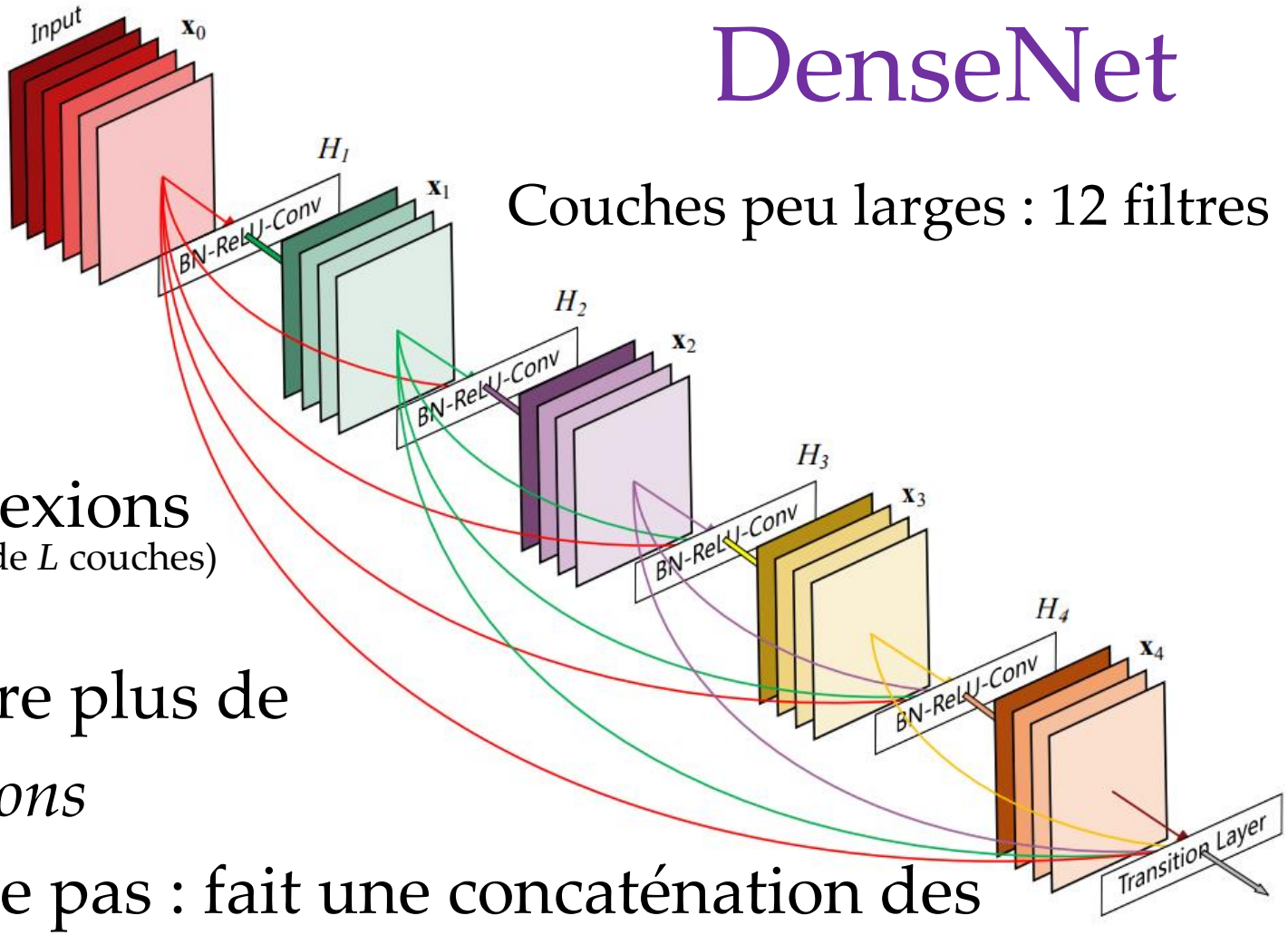
Wide Residual Network

- Manières d'augmenter la capacité des *residual blocks* :
  - ajouter plus de couches de convolution par bloc
  - ajouter plus de filtres dans les couches de convolution
  - augmenter la taille des filtres
- Un réseau WRN de 50 couches bat ResNet-152



[Zagoruyko et al., Wide Residual Network, BMVC, 2016]

DenseNet



Couches peu larges : 12 filtres

$\frac{L(L+1)}{2}$ connexions
(par bloc de L couches)

- Ajoute encore plus de *skip connections*
- N'additionne pas : fait une concaténation des *features* des couches précédentes

ResNet : $\mathbf{x}_\ell = H_\ell(\mathbf{x}_{\ell-1}) + \mathbf{x}_{\ell-1}$

DenseNet : $\mathbf{x}_\ell = H_\ell([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}])$

concaténation

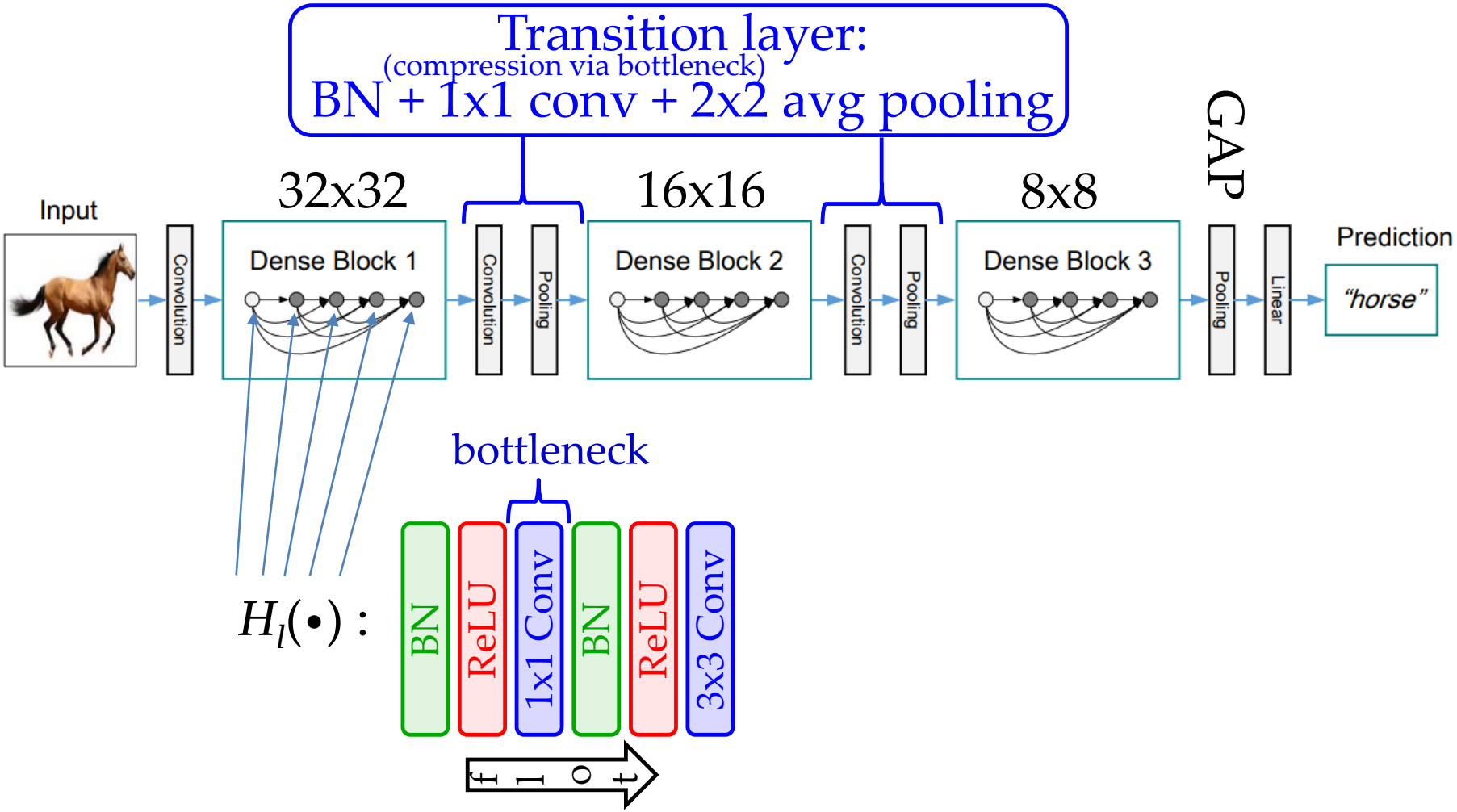
Huang et al., Densely Connected Convolutional Networks, 2016.

DenseNet

- Diminue *vanishing gradient*
- Augmente la propagation et réutilisation de *features*
- Semble régulariser
- Réduit le nombre de paramètres :
  - Conv standard : doit apprendre **implicitement** quels *features* laisser passer
  - ResNet : **explicite**, mais certaines couches contribuent peu (voir Stoch. Depth), donc des poids inutiles

DenseNet

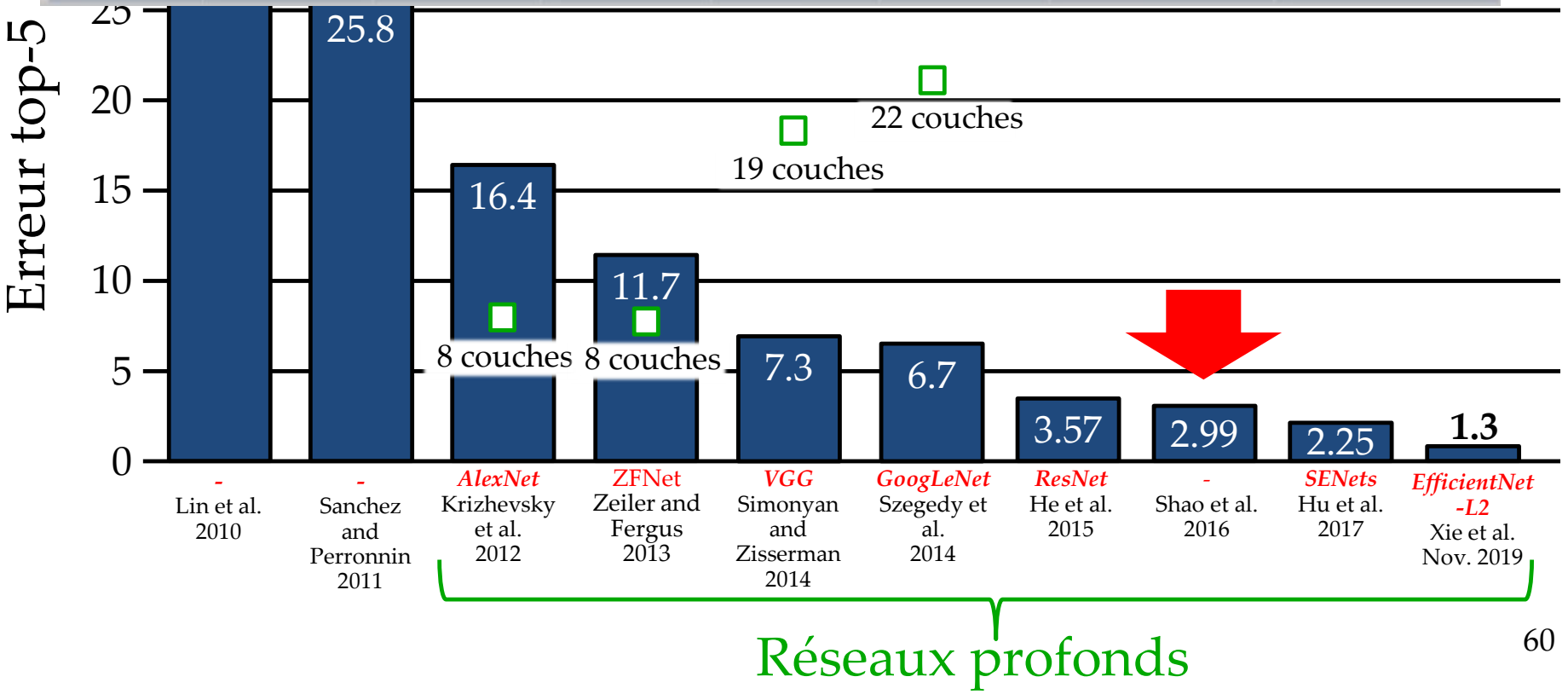
- Connection vers toutes les couches subséquentes qui ont la même taille de feature map (Dense Block)



Large Scale Visual Recognition Challenge

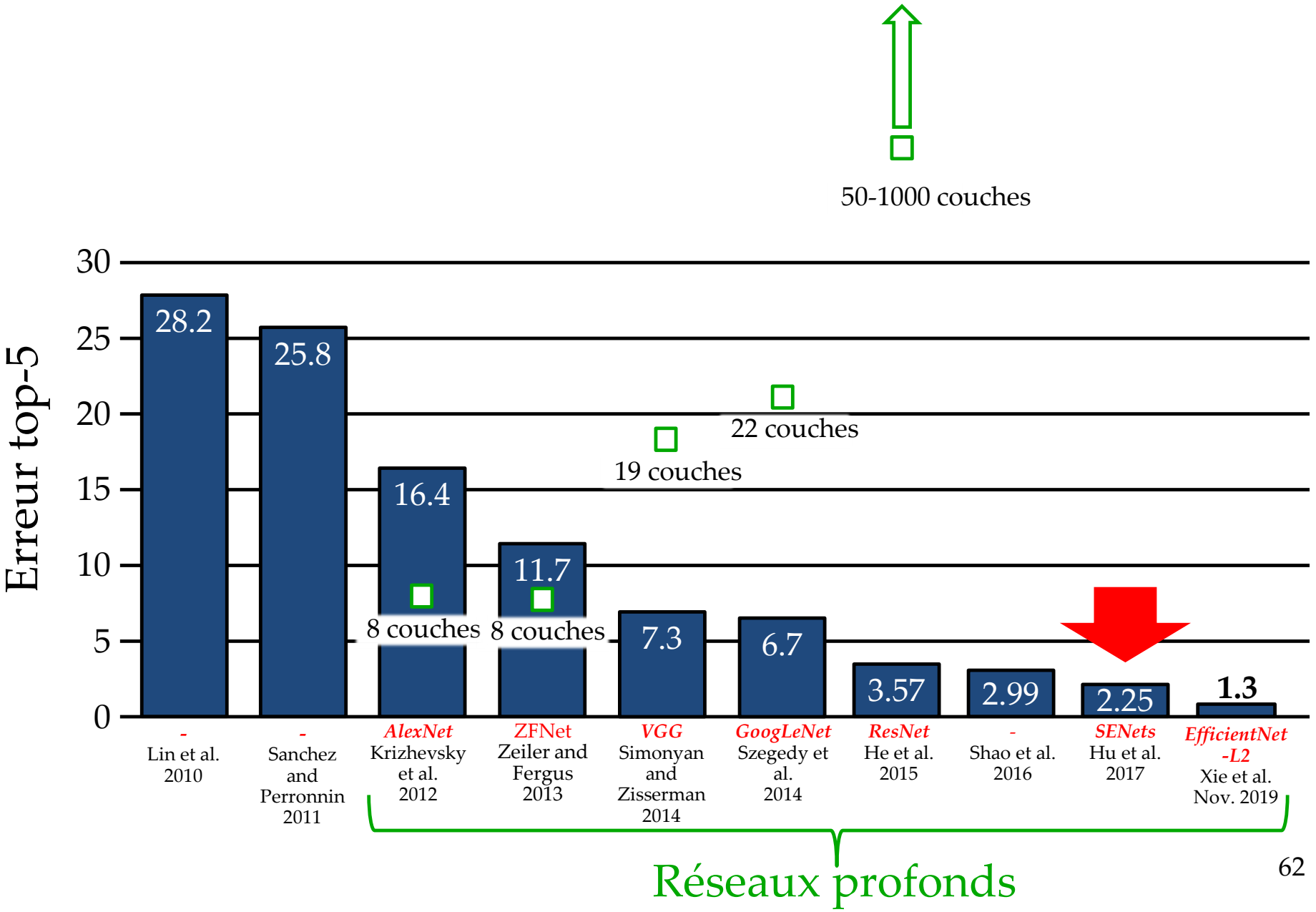
Shao et al. : méthode ensemble, peu intéressant

	Inception-v3	Inception-v4	Inception-Resnet-v2	Resnet-200	Wrn-68-3	Fusion (Val.)	Fusion (Test)
Err. (%)	4.20	4.01	3.52	4.26	4.65	2.92 (-0.6)	2.99



SENets :
Squeeze-and-Excite Nets

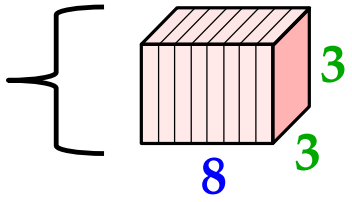
Large Scale Visual Recognition Challenge



Squeeze and Excite Net : SENet

- Exemple de **microarchitecture**
- convnet standard entremêle l'information **spatiale** et du domaine des features

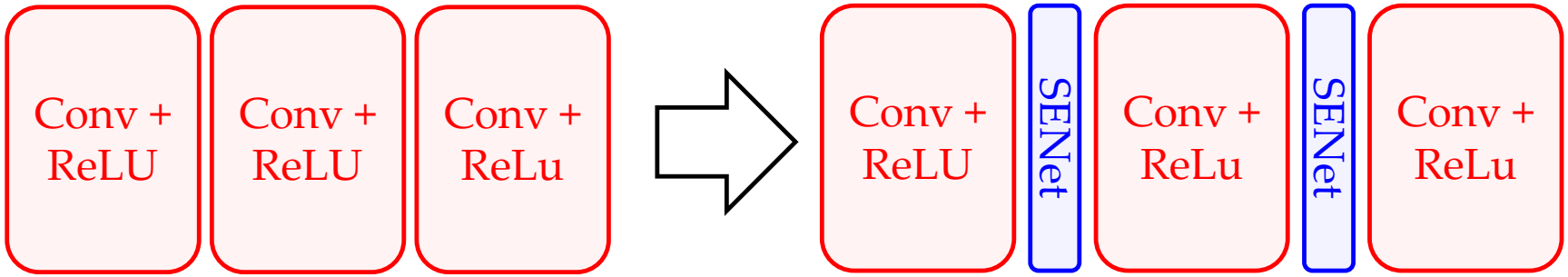
Dans une région spatiale donnée (localité)



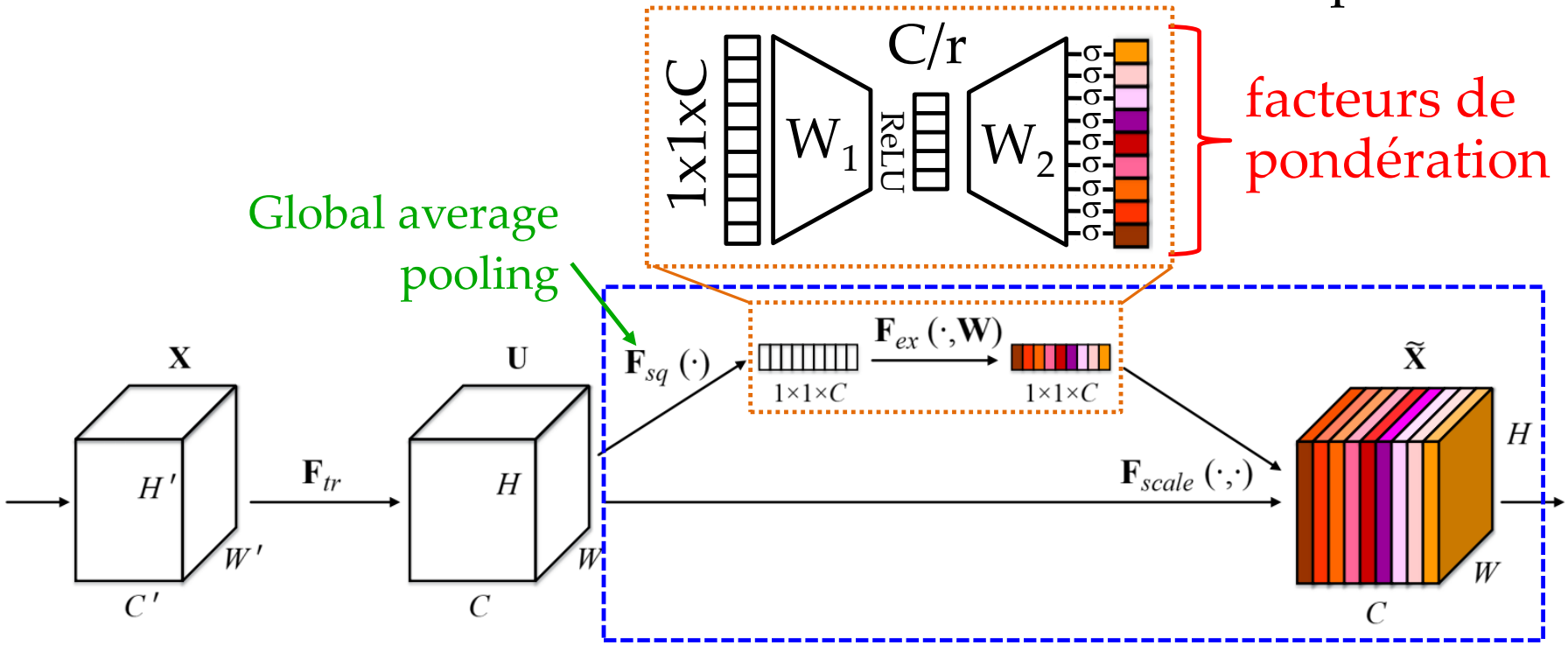
- SENet: va pondérer les *feature maps* **globalement** avec un micro-réseau
- Ajoute très peu de paramètres et de calcul
  - architectures sont déjà larges
- Idée du *bottleneck* qui revient

SENet

- Ajout tel-que-tel (*drop-in*)



r : ratio de compression

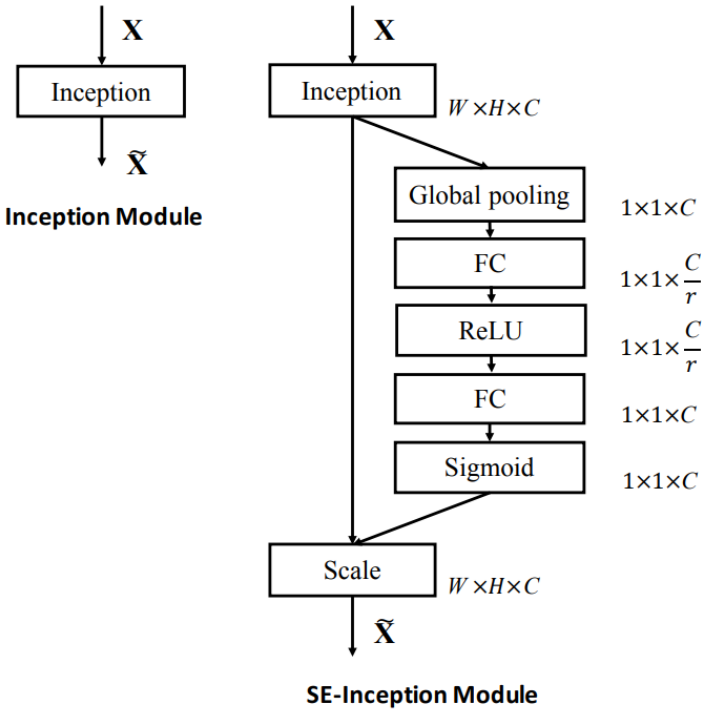


SENet

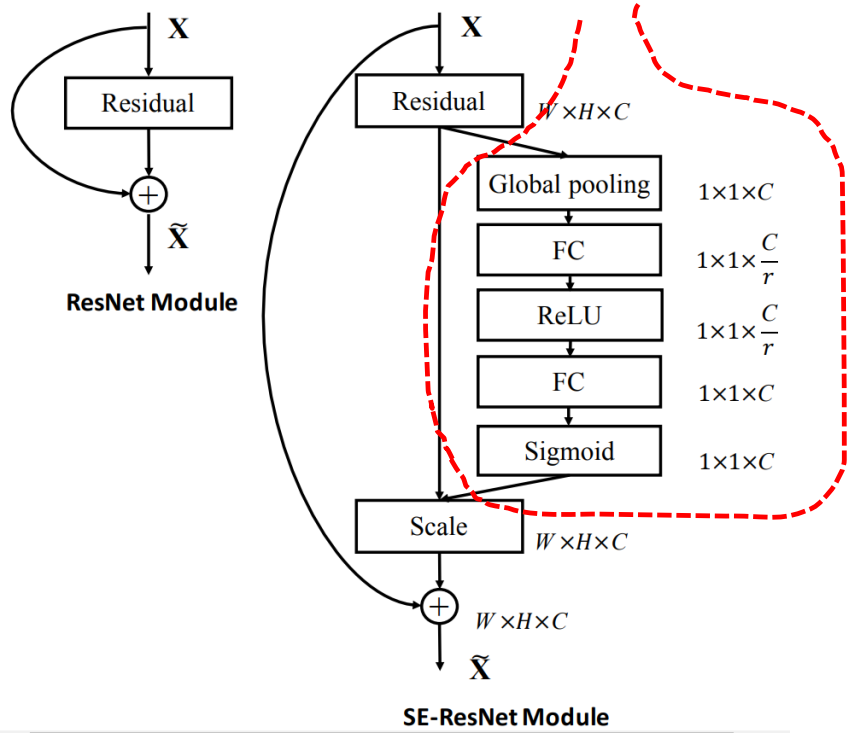
- Modéliser les interdépendances entre *feature maps* avec un micro-réseau
- Appelle ce mécanisme *feature recalibration*
- Forme d'**attention**, mais selon les *feature map*
- Utilise l'information globale (via GAP) pour rehausser des *features* utiles et réprimer les moins utiles

SENet

- Testés sur différentes architectures
  - ResNet
  - Inception et Inception-ResNet
  - ResNeXt



ajoute 4-10% + de paramètres



SENet : Résultats ImageNet

Différence négligeable en calcul

	original		re-implementation			SENet		
	top-1 err.	top-5 err.	top-1 err.	top-5 err.	GFLOPs	top-1 err.	top-5 err.	GFLOPs
ResNet-50 [9]	24.7	7.8	24.80	7.48	3.86	$23.29_{(1.51)}$	$6.62_{(0.86)}$	3.87
ResNet-101 [9]	23.6	7.1	23.17	6.52	7.58	$22.38_{(0.79)}$	$6.07_{(0.45)}$	7.60
ResNet-152 [9]	23.0	6.7	22.42	6.34	11.30	$21.57_{(0.85)}$	$5.73_{(0.61)}$	11.32
ResNeXt-50 [43]	22.2	-	22.11	5.90	4.24	$21.10_{(1.01)}$	$5.49_{(0.41)}$	4.25
ResNeXt-101 [43]	21.2	5.6	21.18	5.57	7.99	$20.70_{(0.48)}$	$5.01_{(0.56)}$	8.00
BN-Inception [14]	25.2	7.82	25.38	7.89	2.03	$24.23_{(1.15)}$	$7.14_{(0.75)}$	2.04
Inception-ResNet-v2 [38]	$19.9^{\dagger}$	$4.9^{\dagger}$	20.37	5.21	11.75	$19.80_{(0.57)}$	$4.79_{(0.42)}$	11.76

Gains partout

r : facteur de réduction de dimensionnalité

Ratio *r*	top-1 err.	top-5 err.	model size (MB)
4	23.21	6.63	137
8	23.19	6.64	117
16	23.29	6.62	108
32	23.40	6.77	103
original	24.80	7.48	98

valeur utilisée ⇩

SENet est amélioration « orthogonale » : il suffit simplement de l'ajouter à n'importe quel réseau !

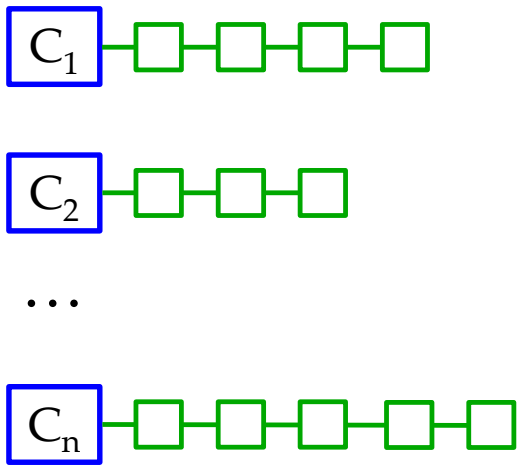
(2300 citations)

SENet

- Rôle joué semble varier en fonction de la profondeur
  - en bas, les excitations des *features* sont peu corrélées avec la classe
  - en haut, les excitations des *features* sont plus spécialisés (en fonction de la classe)

SENet

- Utilise une méthode dite « class-aware sampling », pour contrer débalancement du nombre d'exemples par classe
- Liste de classes C
- Listes d'exemples par classe
- Sampling :
  - pige la classe
  - prend une image dans la liste
    - si fin de la liste, reshuffle



Relay Backpropagation for Effective Learning of Deep Convolutional Neural Networks, https://arxiv.org/abs/1512.05830, 2016.

Convolution
depthwise separable

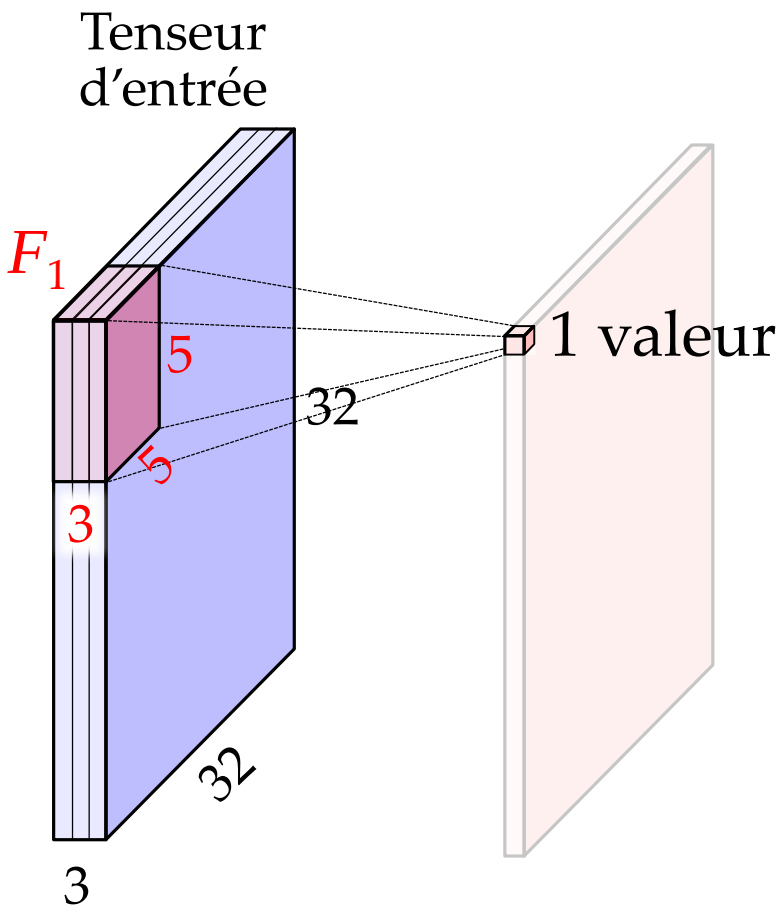
(en route vers EfficientNet…)

Convolution *depthwise separable*

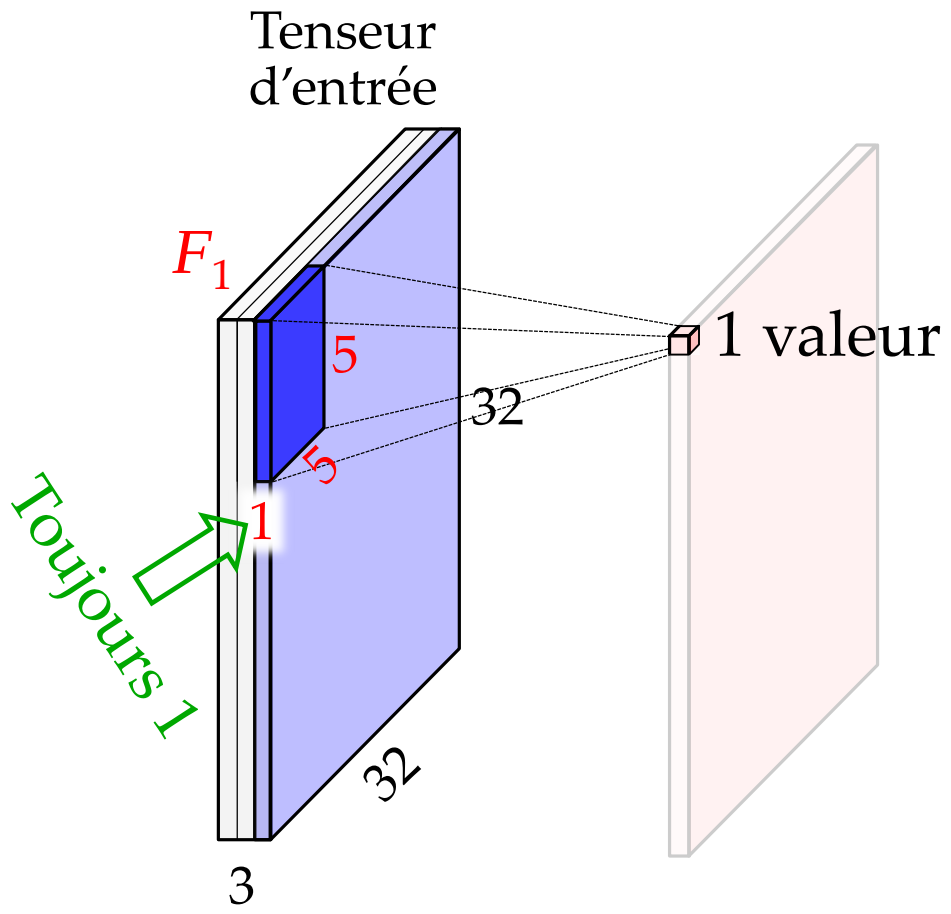
- Remplacement direct des convolutions standards
- Réduction
  - temps de calcul
  - nombre de paramètres
- Se fait en deux étapes subséquentes
  - Convolutions ***depthwise*** (FxFx1)
  - Convolutions ***pointwise*** (1x1xC)
- Utilisés dans de nombreuses architectures
  - MobileNets
  - EfficientNets

Filtres convolutifs : comparaison

Standard

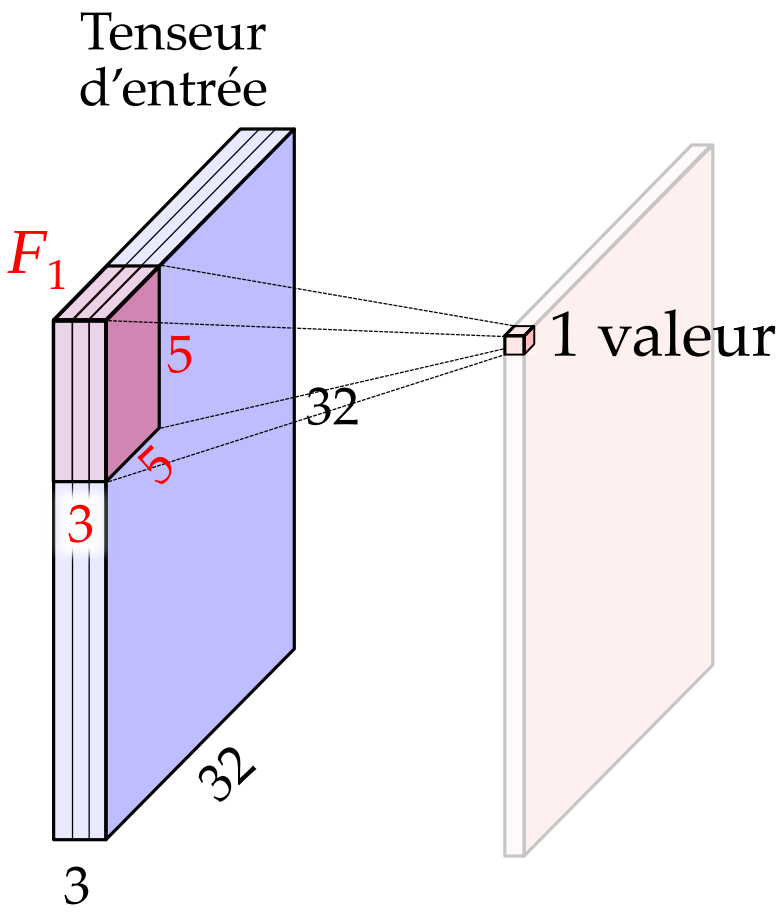


depthwise

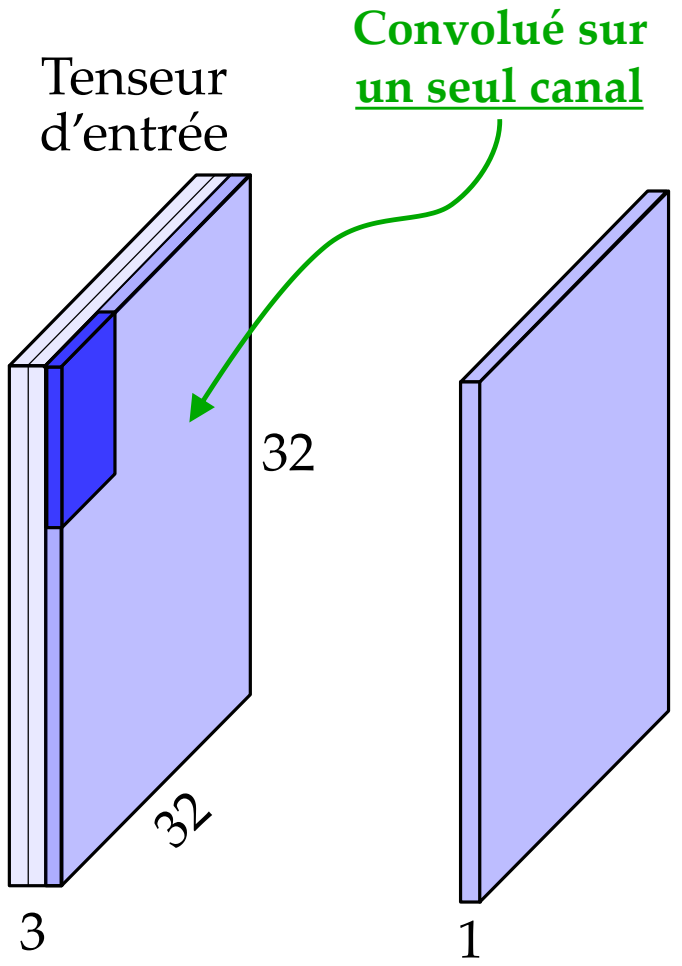


Filtres convolutifs : comparaison

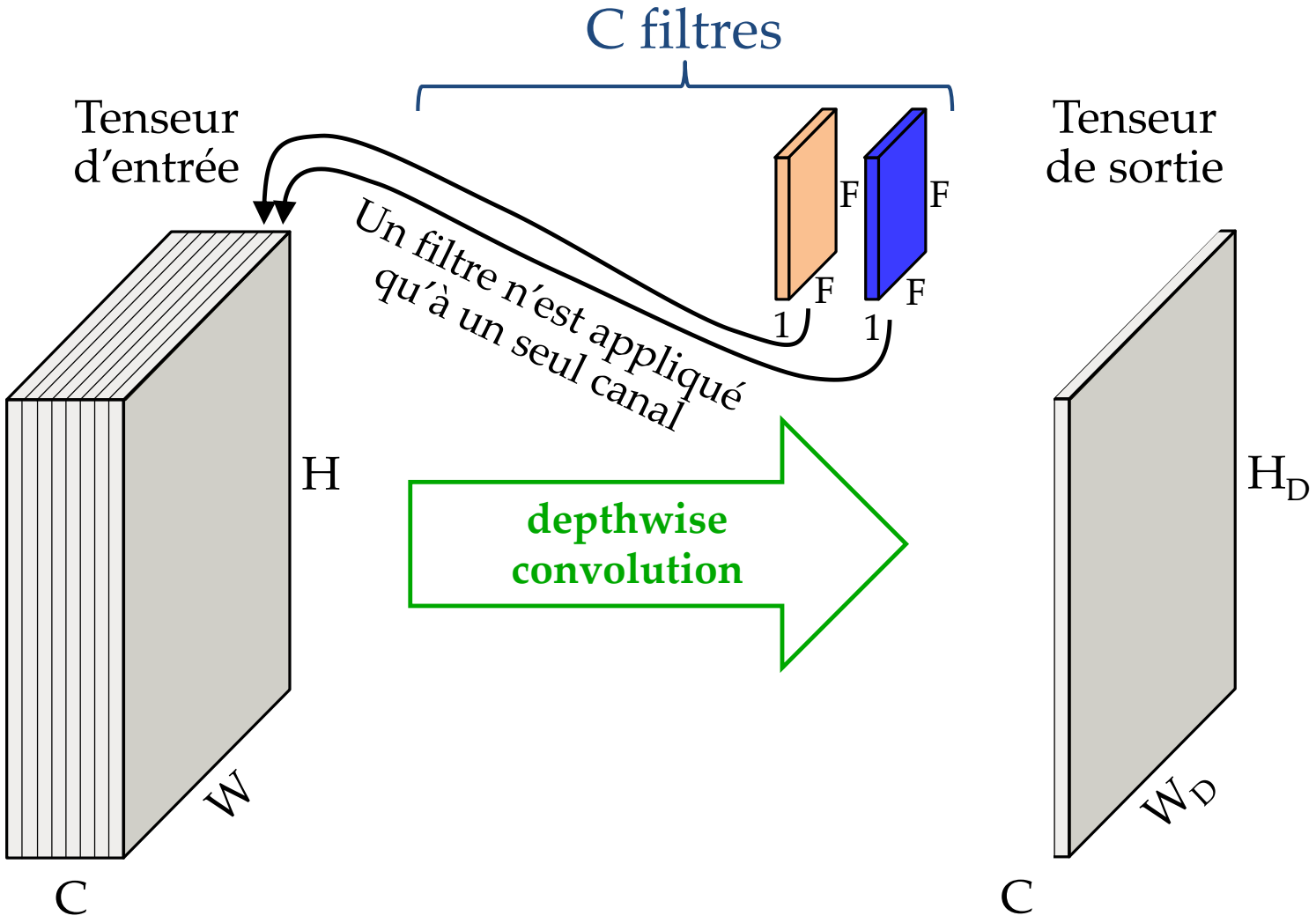
Standard



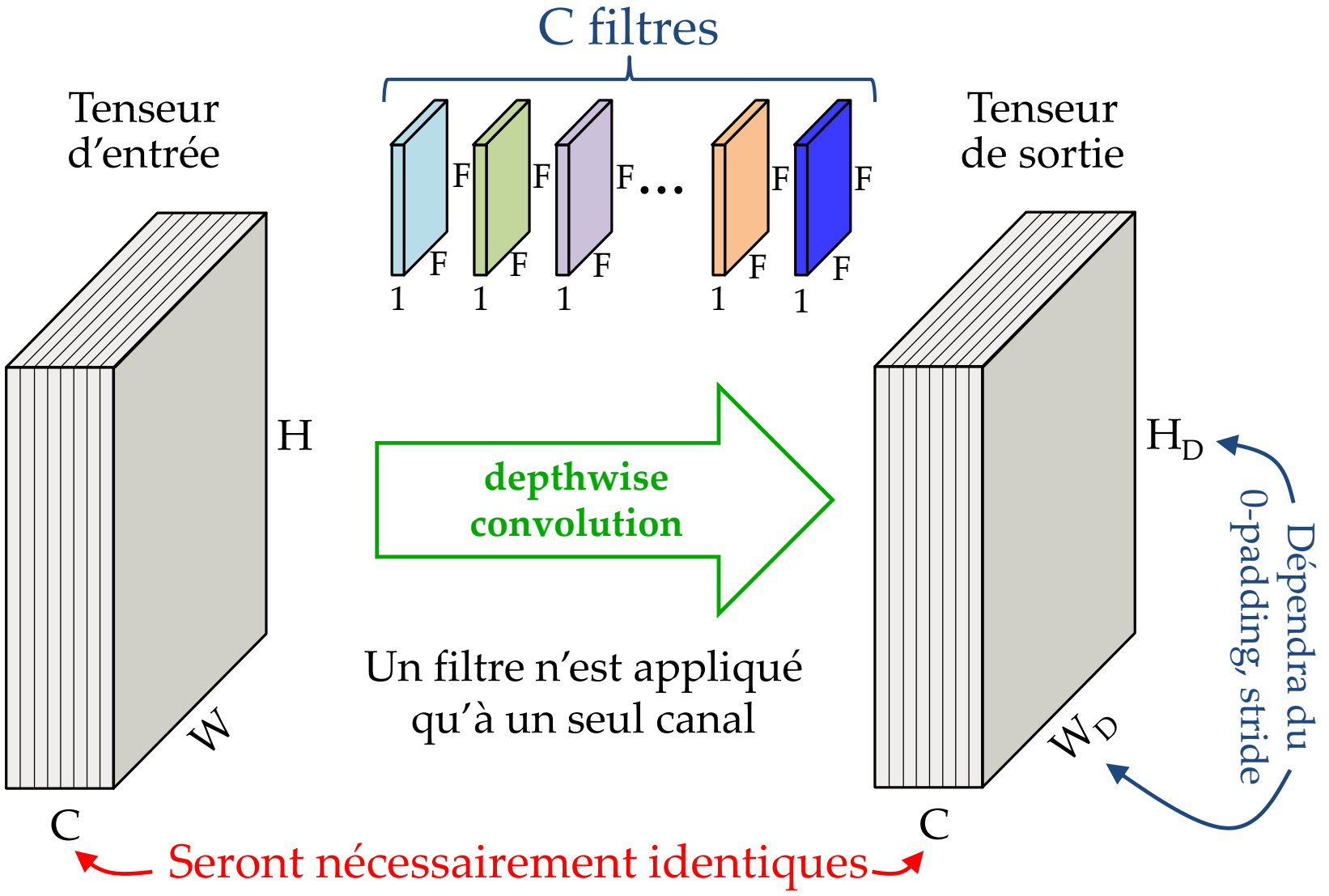
depthwise



Étape 1 : convolution depthwise

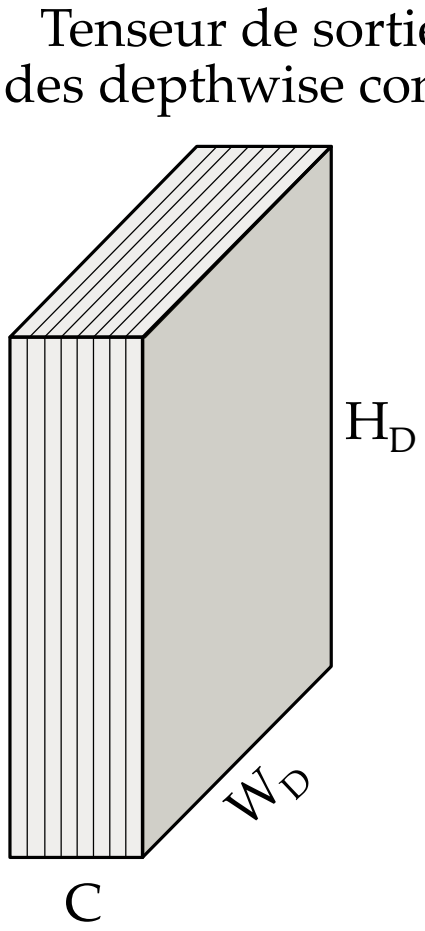


Étape 1 : convolution depthwise



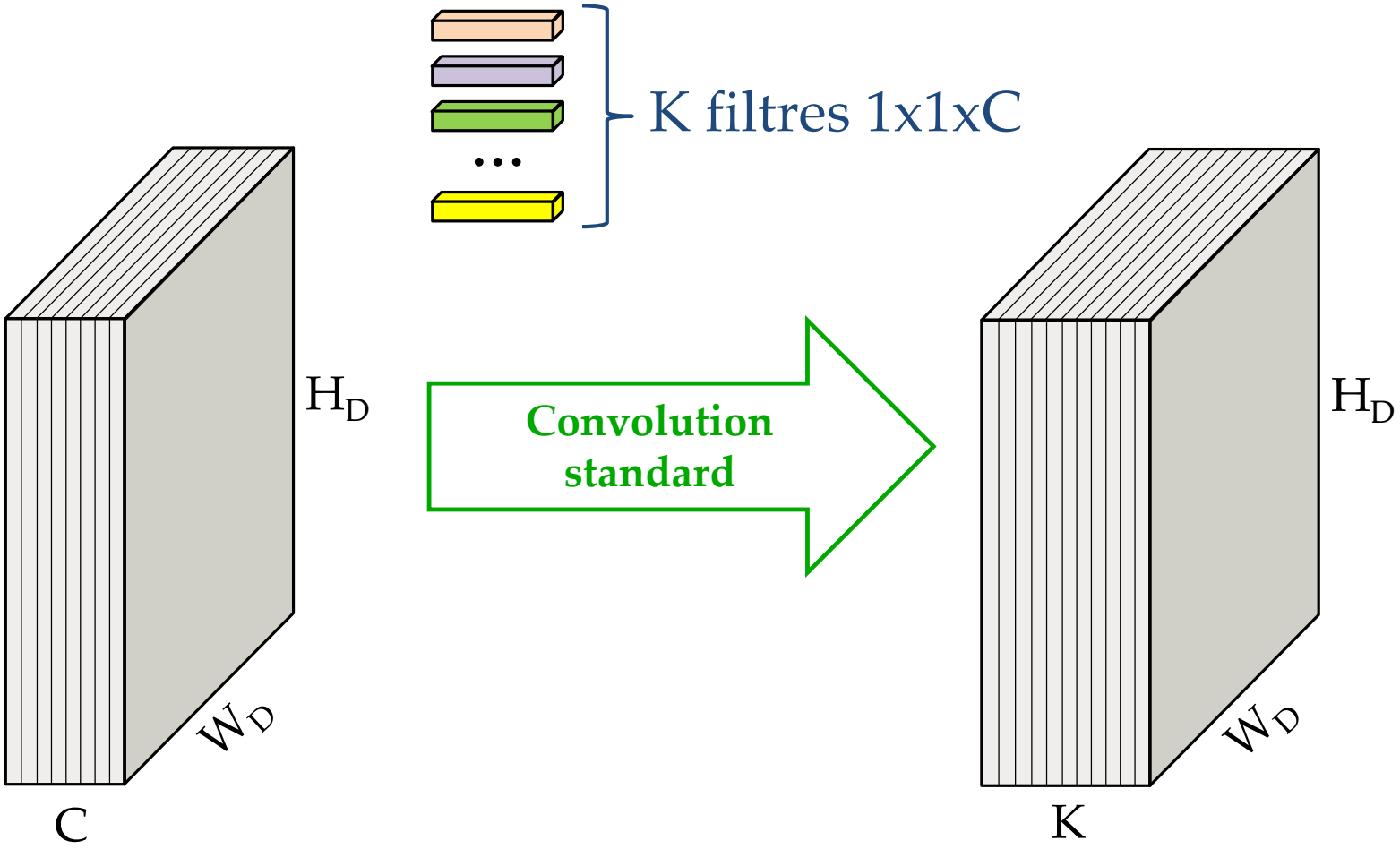
Étape 2 : pointwise convolution

- Simplement des convolutions 1x1

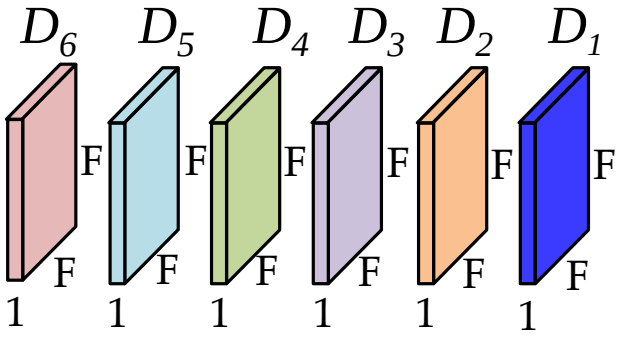


Étape 2 : pointwise convolution

- Simplement des convolutions 1x1

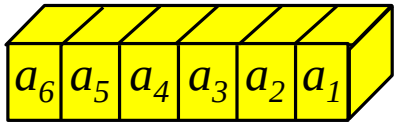


Depthwise separable : Combinaison linéaire



FxFx1

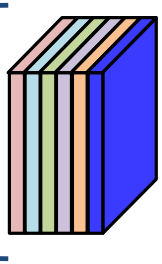
x



1x1xC

=

Équivalent à convoluer sur tenseur d'entrée

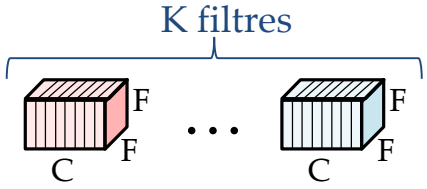


FxFxC

Temps de calcul (# multipl.)

Tenseur HxWxC (avec 0-pading)

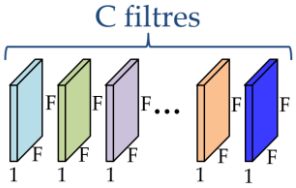
Standard



K Filtres FxFxC

Temps : CF^2HWK

depthwise separable



C Filtres FxFx1

K filtres 1x1xC

Temps : $\underbrace{CHWF^2}_{\text{depthwise}} + \underbrace{CHWK}_{\text{1x1}}$

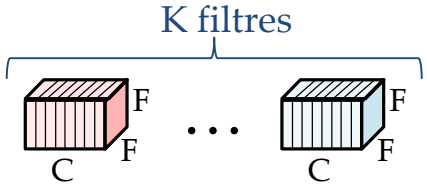
$= CHW(F^2+K)$

$$\frac{\#\text{ mult. depthwise separable}}{\#\text{ mult. convolution standard}} = \frac{F^2+K}{F^2K} = \frac{1}{K} + \frac{1}{F^2} \approx \frac{1}{F^2}$$ (Pour K>>F²)

Nombre de paramètres

Tenseur HxWxC (avec 0-pading)

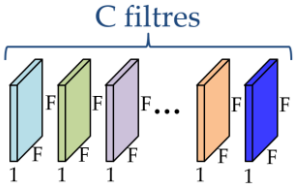
Standard



K Filtres FxFxC

Nombre: CF^2K

depthwise separable



C Filtres FxFx1

K filtres 1x1xC

Nombre : $\overbrace{CF^2}^{\text{depthwise}} + \overbrace{CK}^{\text{1x1}}$

$= C(F^2+K)$

$$\frac{\#\text{ param. depthwise separable}}{\#\text{ param. convolution standard}} = \frac{F^2+K}{F^2K} = \frac{1}{K}+\frac{1}{F^2} \approx \frac{1}{F^2}$$ (Pour K>>F²)

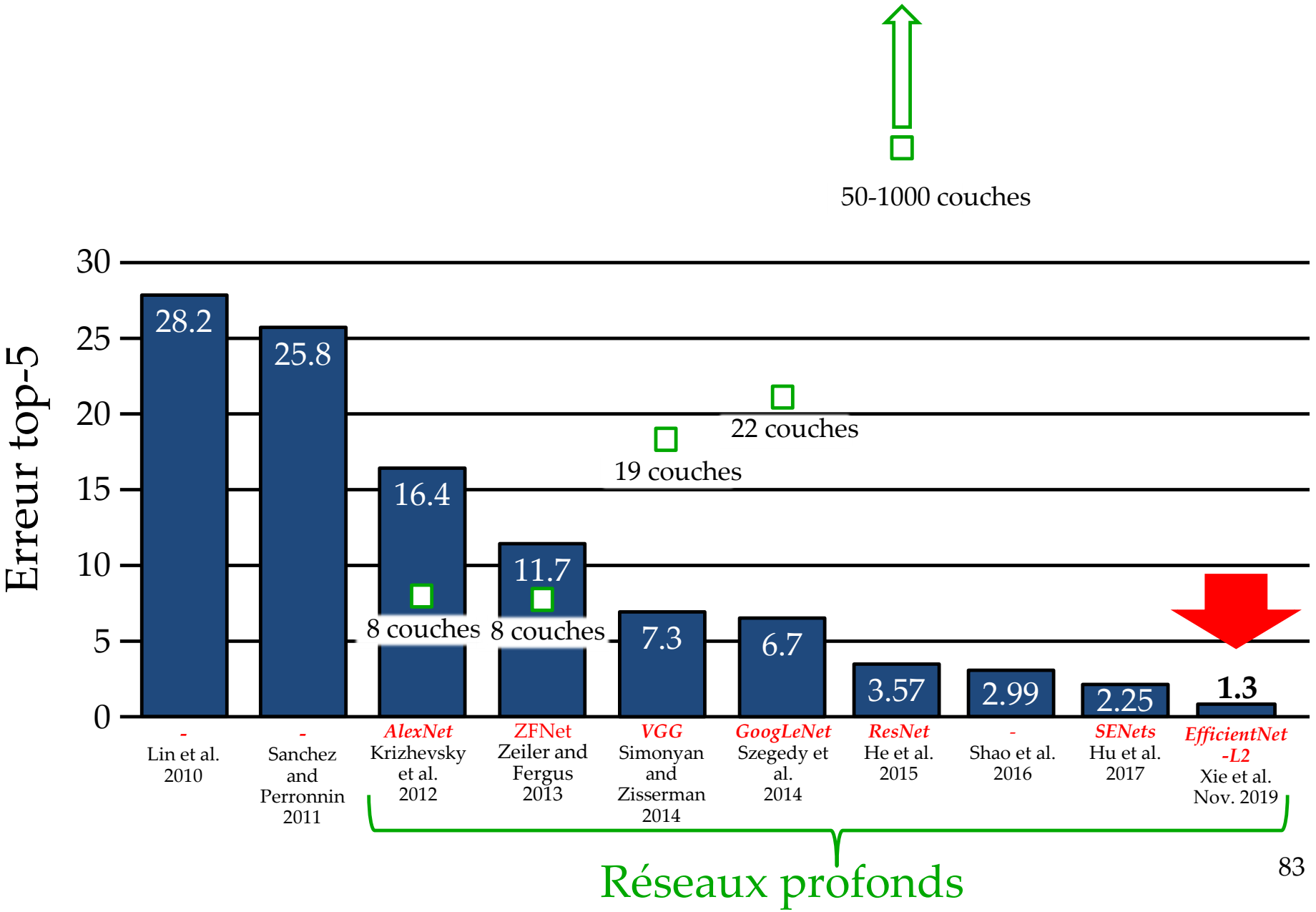
Résumé *depthwise separable*

- Forme de factorisation des convolutions
  - Convolutions ***depthwise*** (FxFx1)
  - Convolutions ***pointwise*** (1x1xC)

  } ~ FxFxC
- Réduction d'un facteur $1/F^2$
  - temps de calcul
  - nombre de paramètres
- Utilisé par plusieurs nouvelles architectures

EfficientNet (2019)

Large Scale Visual Recognition Challenge

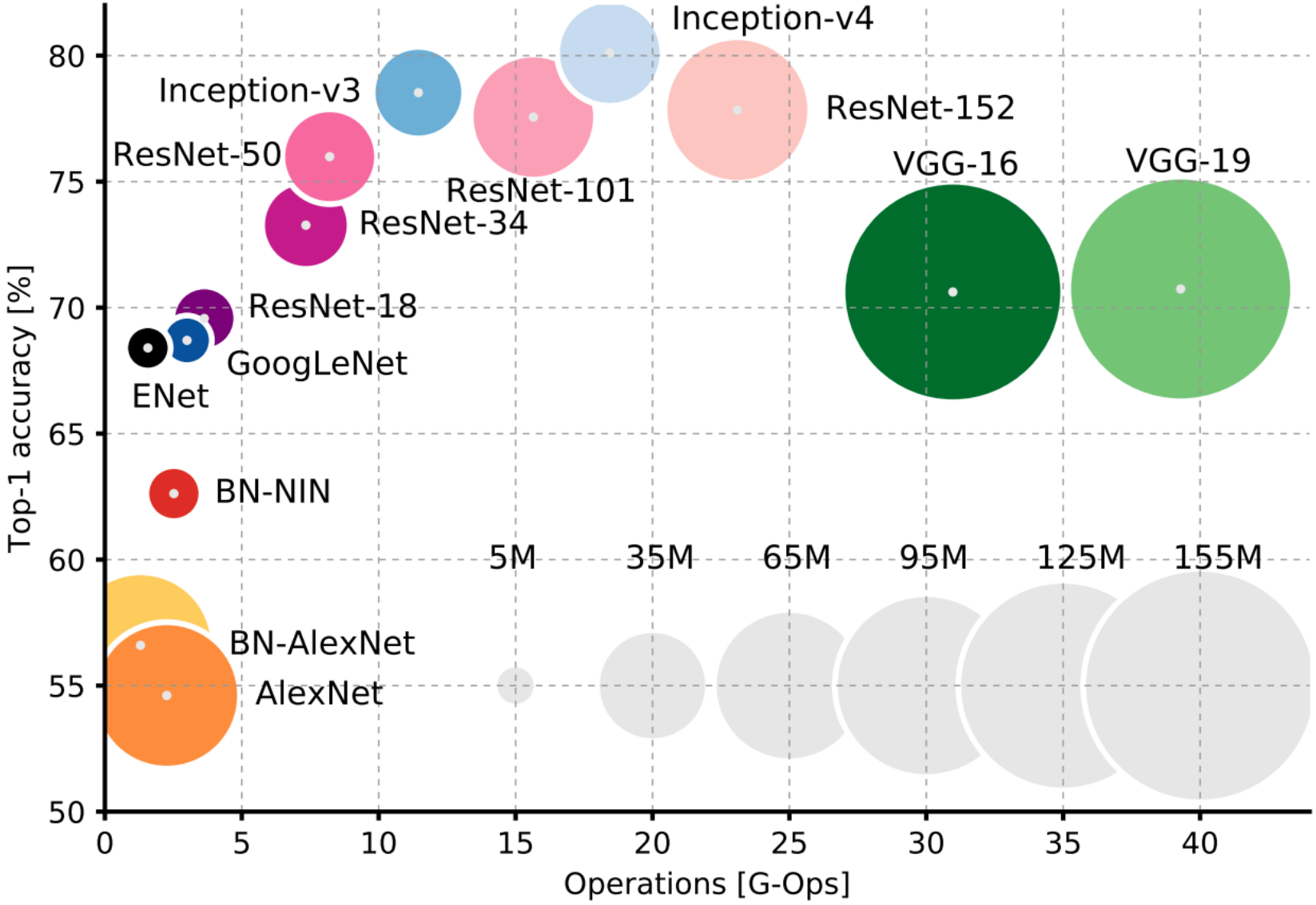


EfficientNet

- Étude systématique de la mise à l'échelle des réseaux, avec un budget de calcul fixe
  - Profondeur (L)
  - Largeur (C)
  - Résolution de l'image (HxW)

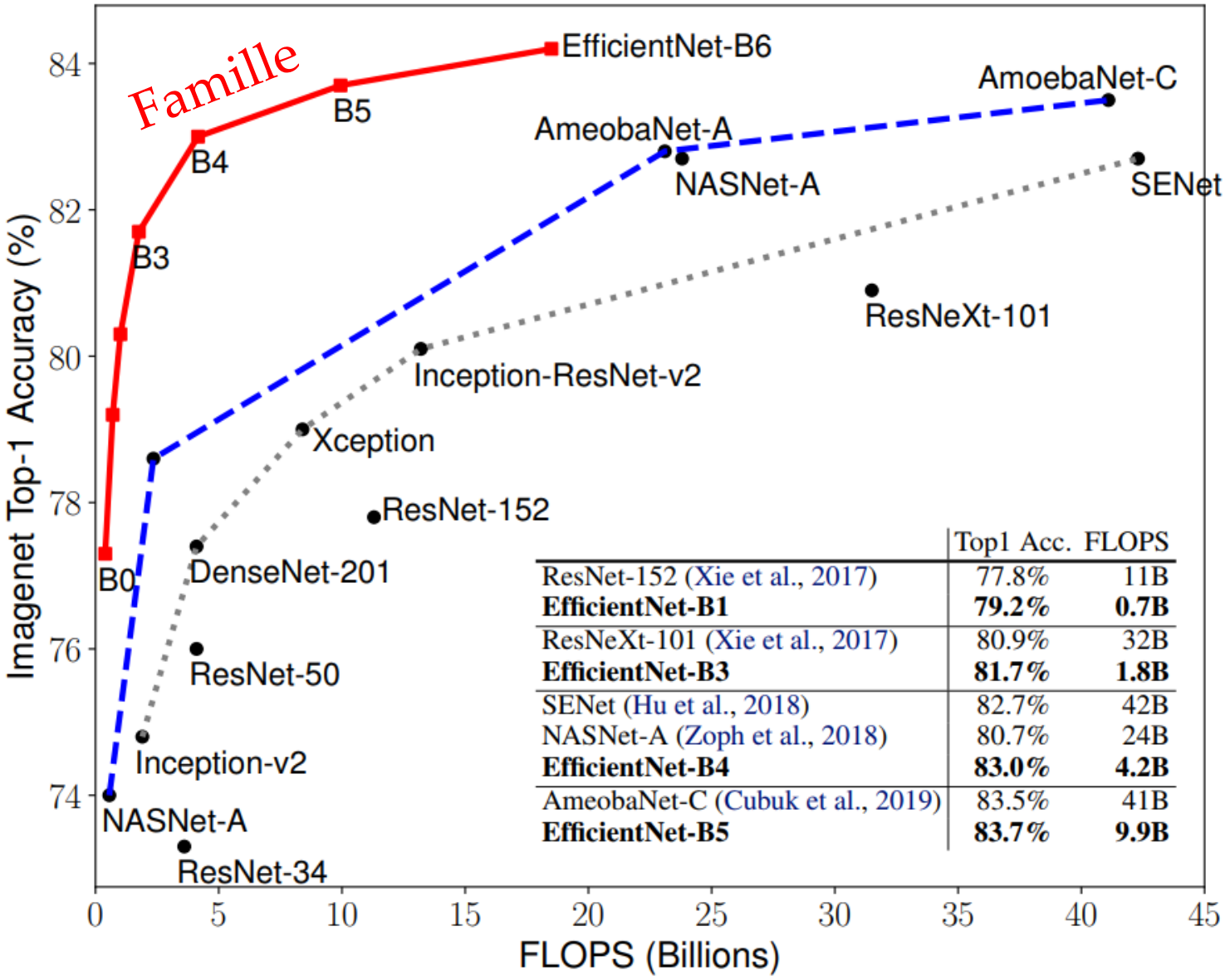
  } Traditionnellement un seul à la fois
- Fait une recherche de réseaux de neurones automatisé (NAS)
  - EfficientNet 8.4x plus petit et 6.1x plus rapide
- Exploite SENet + depthwise conv.

Efficacité des réseaux (2017)



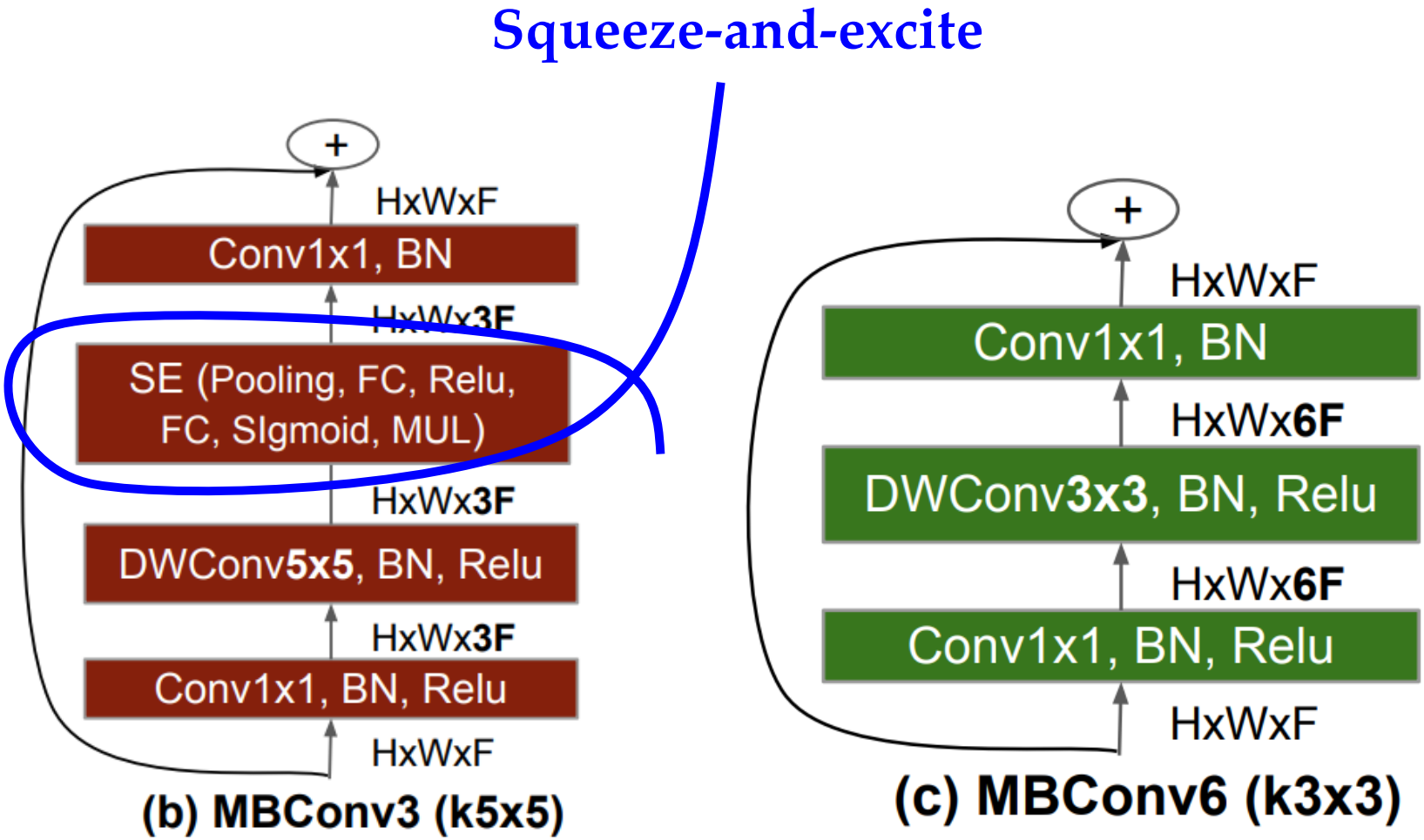
A. Canziani et al. An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Efficacité d'EfficientNet (2019)



	Top1 Acc.	FLOPS
ResNet-152 (Xie et al., 2017)	77.8%	11B
EfficientNet-B1	**79.2%**	**0.7B**
ResNeXt-101 (Xie et al., 2017)	80.9%	32B
EfficientNet-B3	**81.7%**	**1.8B**
SENet (Hu et al., 2018)	82.7%	42B
NASNet-A (Zoph et al., 2018)	80.7%	24B
EfficientNet-B4	**83.0%**	**4.2B**
AmeobaNet-C (Cubuk et al., 2019)	83.5%	41B
EfficientNet-B5	**83.7%**	**9.9B**

Bloc de base : MBConv



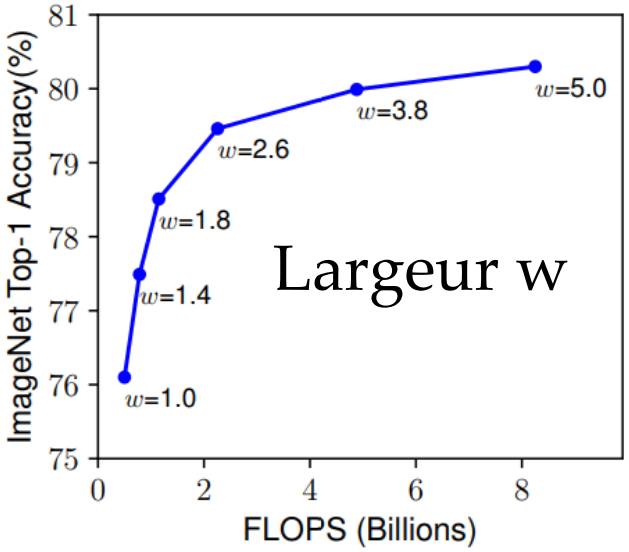
NAS: Network architecture search

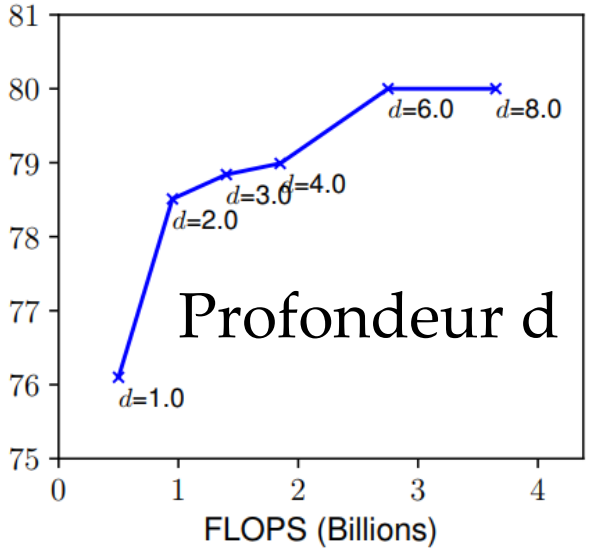
- Identifié un modèle de base via NAS
  - EfficientNet-B0

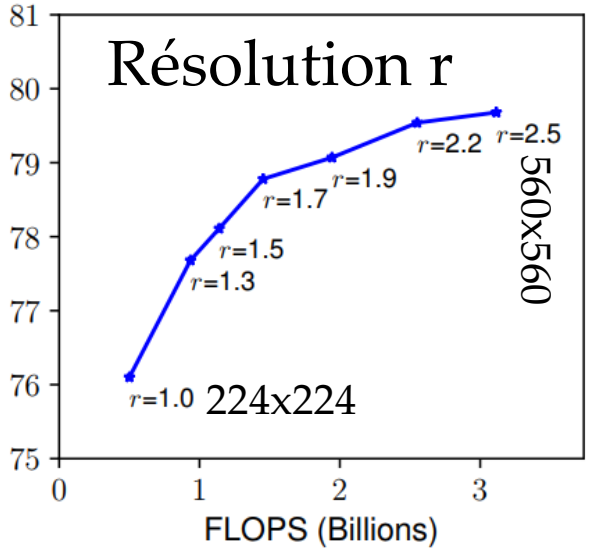
Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Impact des échelles

- Plafonnement lorsqu'on ne fait varier qu'une seule des échelles à la fois







Règle de mise à l'échelle

$$\text{depth: } d = \alpha^\phi$$
$$\text{width: } w = \beta^\phi$$
$$\text{resolution: } r = \gamma^\phi$$
$$\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$
$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

tient compte du temps de calcul (FLOPS) d'une convolution standard : $O(dw^2r^2)$

- α, β et γ trouvé par une recherche en grille
  - fait sur EfficientNet-B0 : α=1.2, β=1.1 γ=1.15
- ϕ est le paramètre d'échelle
- temps de calcul augmente approx. par 2^ϕ

Vérification de la règle d'échelle

- La mise à l'échelle combinée est la meilleure

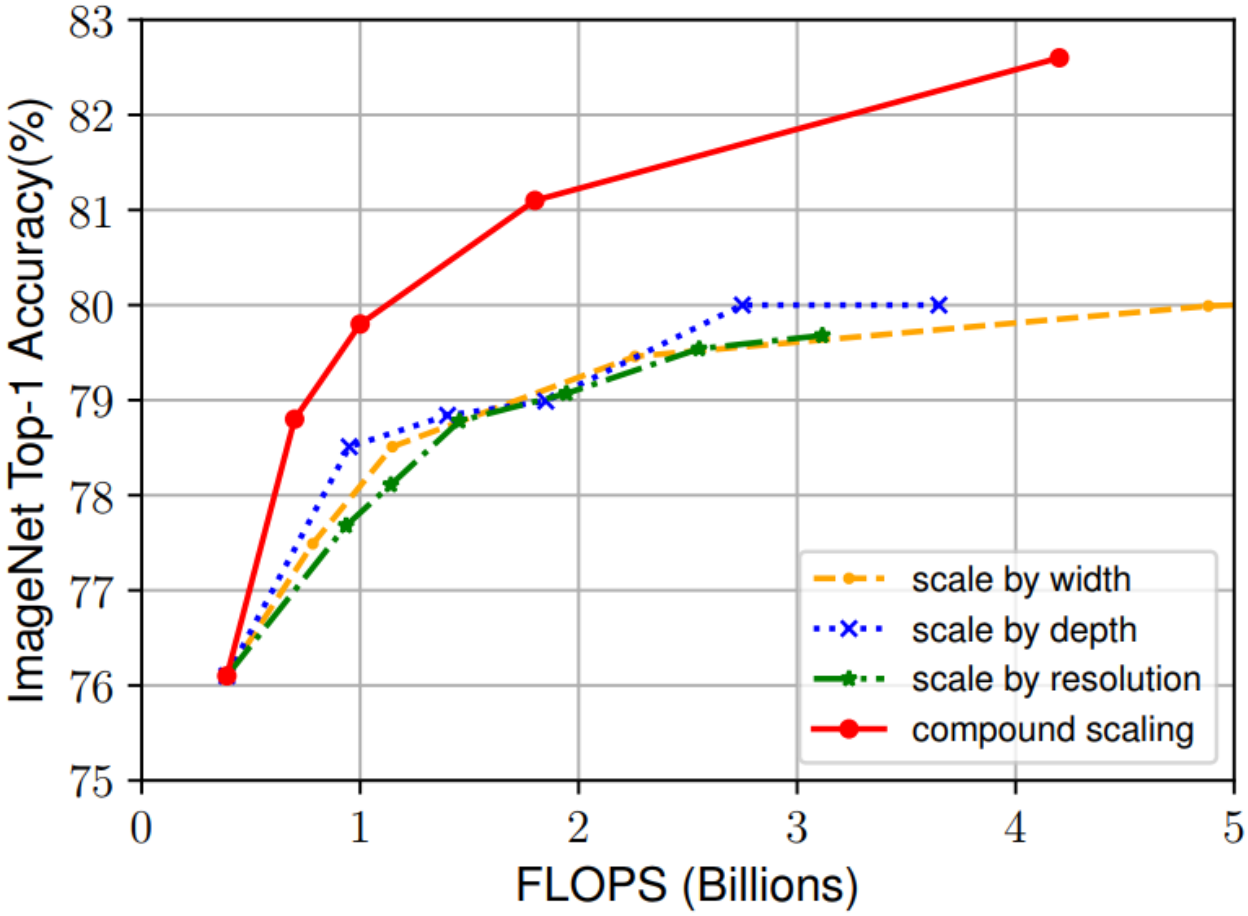
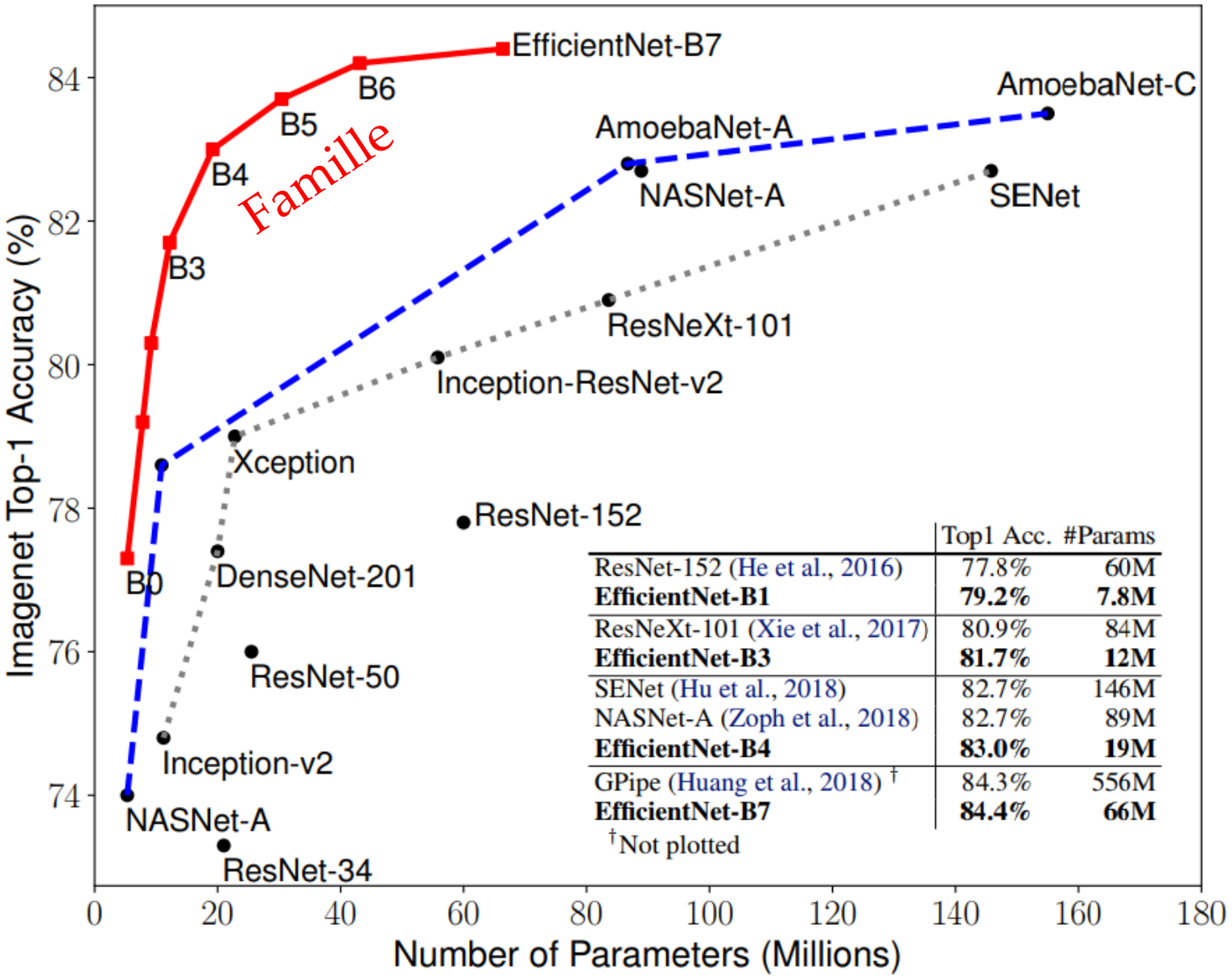


Figure 8. **Scaling Up EfficientNet-B0 with Different Methods.**

EfficientNet (#param)



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.2%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.7%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**83.0%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

†Not plotted

EfficientNet : résumé

- Semble être le state-of-the-art actuel
- Offre 8 versions différentes
  - EfficientNet-B0 à EfficientNet-B7
- Rapide d'exécution/petit
- Fonctionne très bien aussi pour faire du transfert learning