Architectures convolutives

Apprentissage Profond

A. Carlier

2025

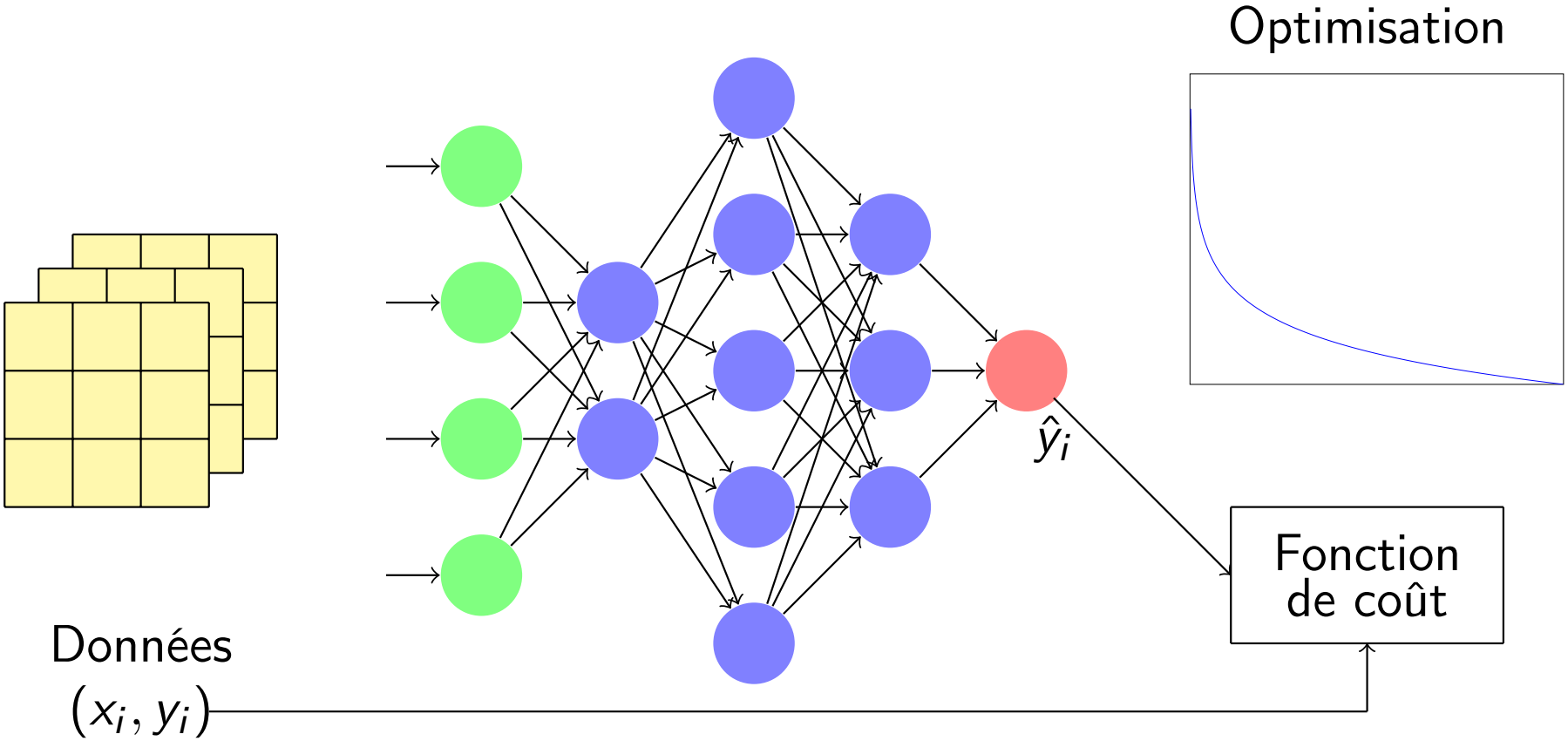
Aperçu du cours

- Traitement d'images
  - Architectures convolutives pour la classification
  - Détection d'objets dans les images
- Traitement du Langage Naturel
  - Architecture du `Transformer`
  - LLM
- Projet de détection d'objet ou de génération de texte

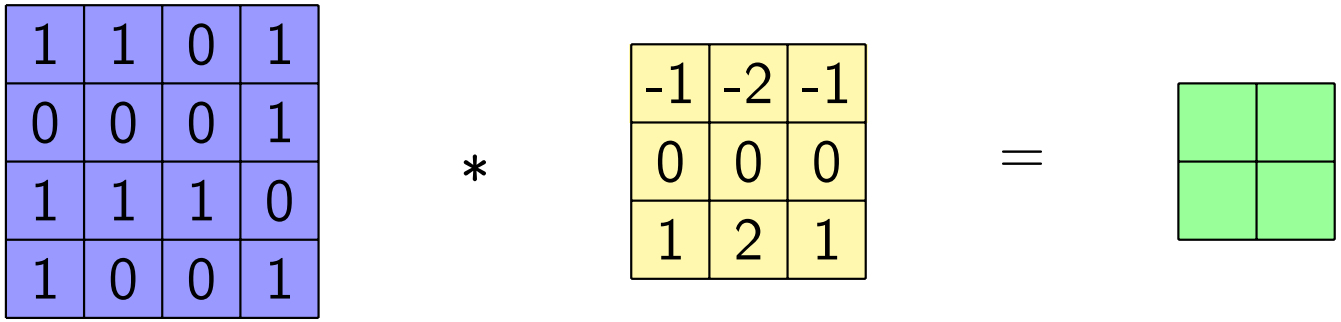
Plan du cours

Rappels de 1A

Entraînement d'un réseau de neurones



Convolution

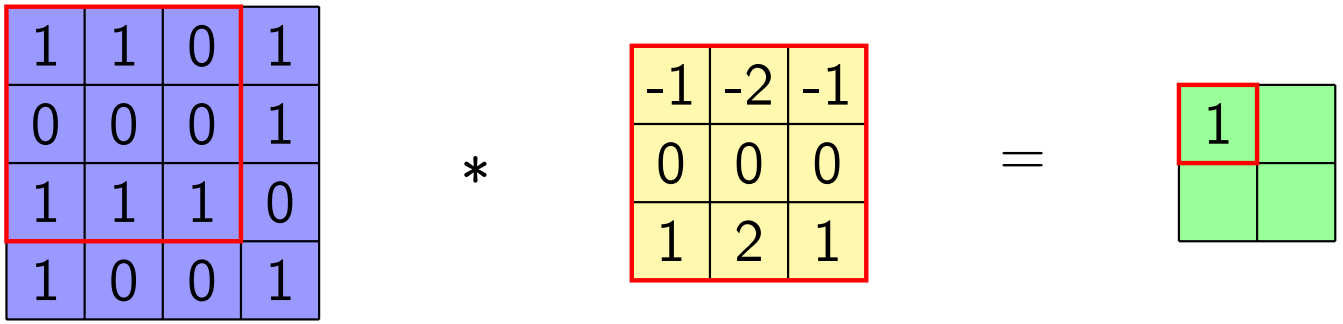


Image

Filtre 3×3
$f = 3$

Réponse

Convolution



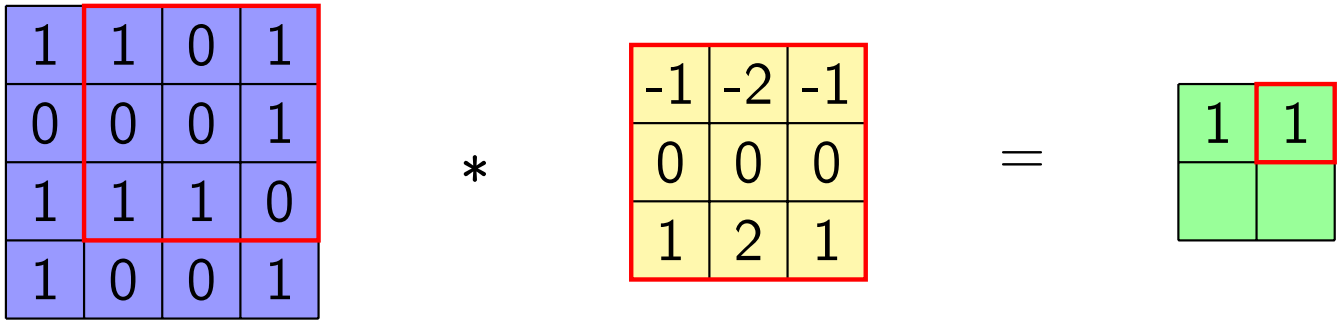
Image

Filtre 3×3
$f = 3$

Réponse

$$1 * -1 + 1 * -2 + 0 * -1 + 0 * 0 + 0 * 0 + 0 * 0 + 1 * 1 + 1 * 2 + 1 * 1 = 1$$

Convolution

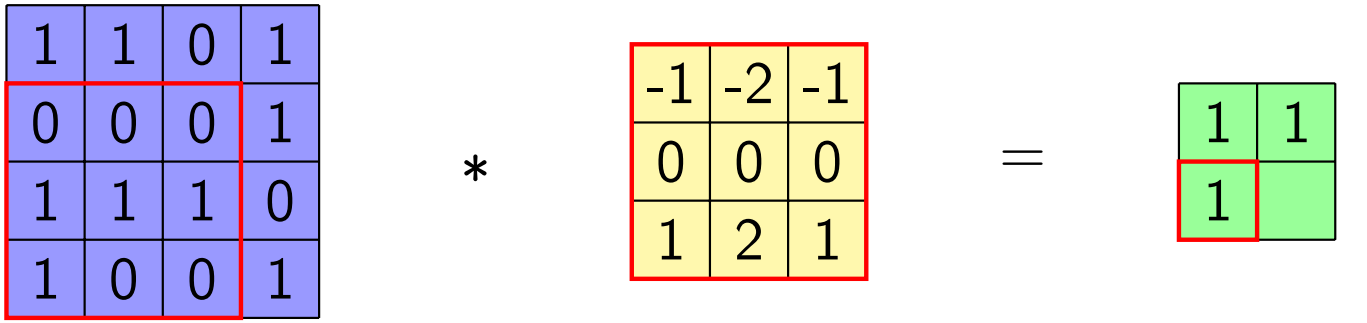


Image

Filtre 3×3
$f = 3$

Réponse

Convolution

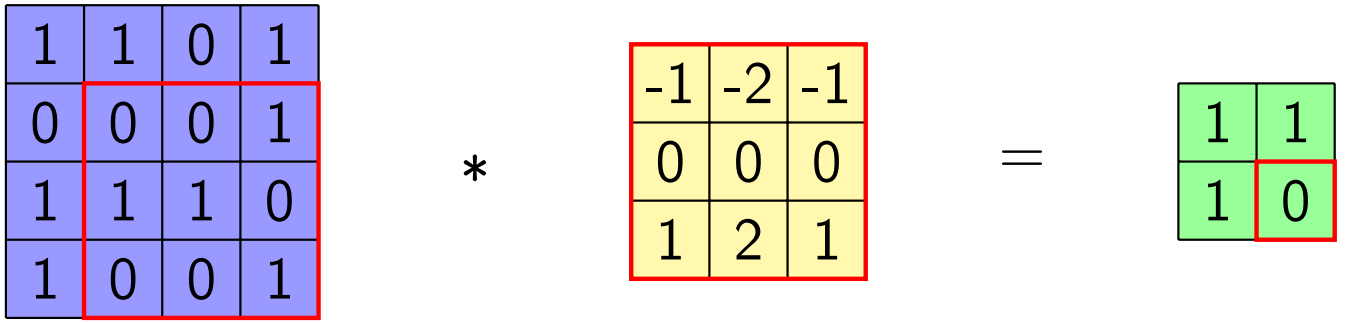


Image

Filtre 3×3
$f = 3$

Réponse

Convolution

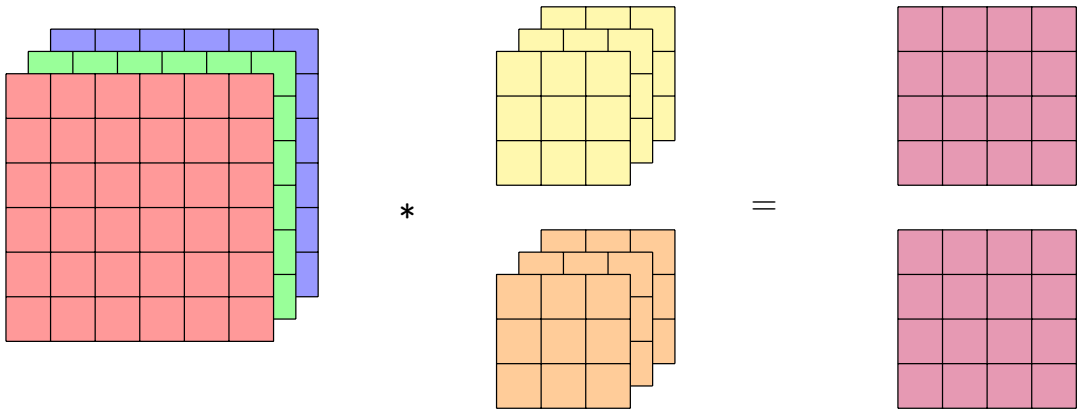


Image

Filtre 3×3
$f = 3$

Réponse

Couche de convolution



Il y a 2 types de paramètres dans une couche de convolution :

- Les **coefficients des filtres de convolution** : il y en a donc $f \times f \times$ #canaux $\times$ #filtres
- Les **biais** additionnés à la réponse des filtres de convolution, avant l'application de la fonction d'activation. Il y a exactement un biais par filtre de convolution.

Ainsi dans l'exemple ci-dessus, il y a 3x3x3x2 + 2 = 56 paramètres.

Padding

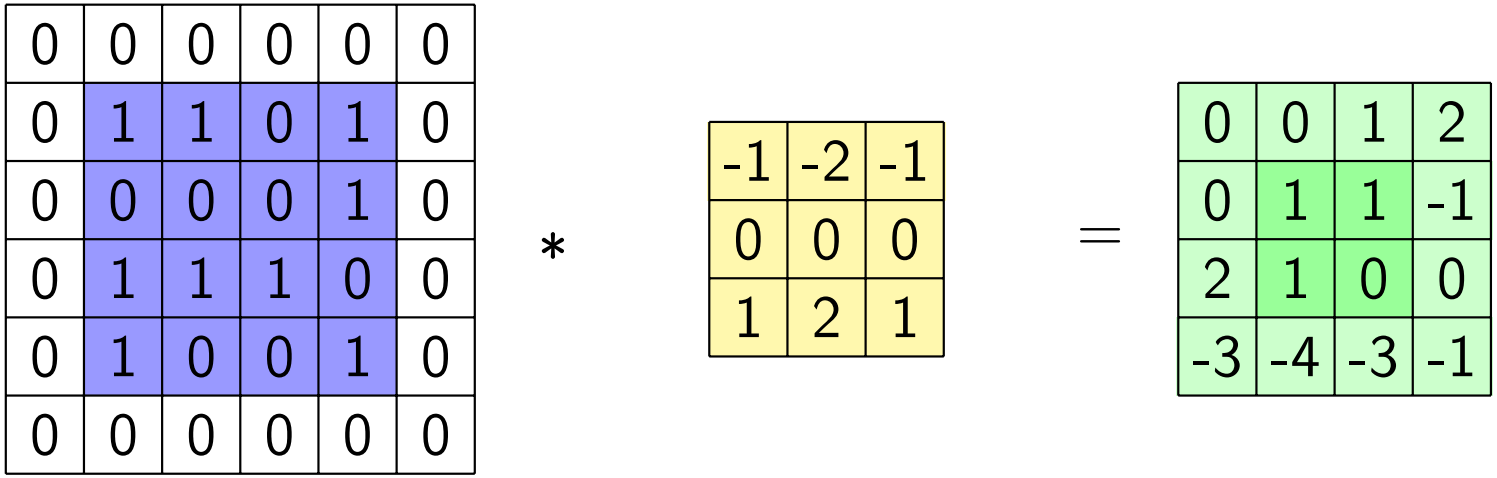
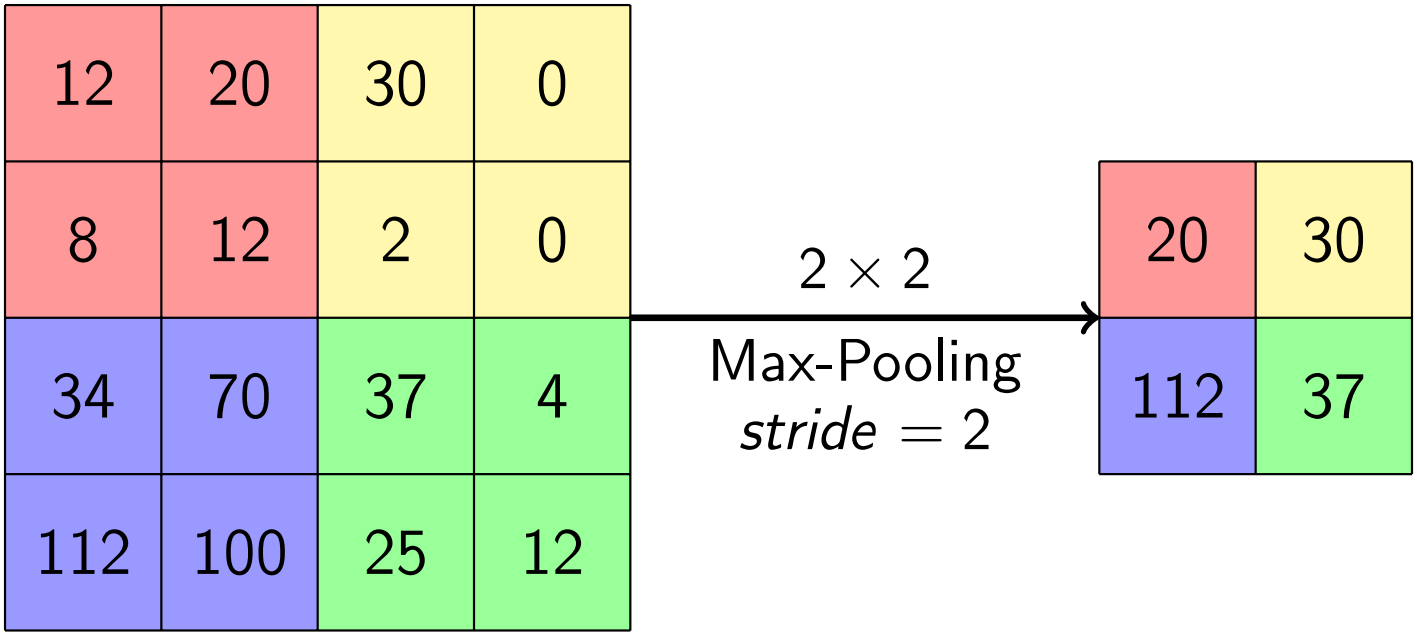


Image
$p = 1$

Filtre

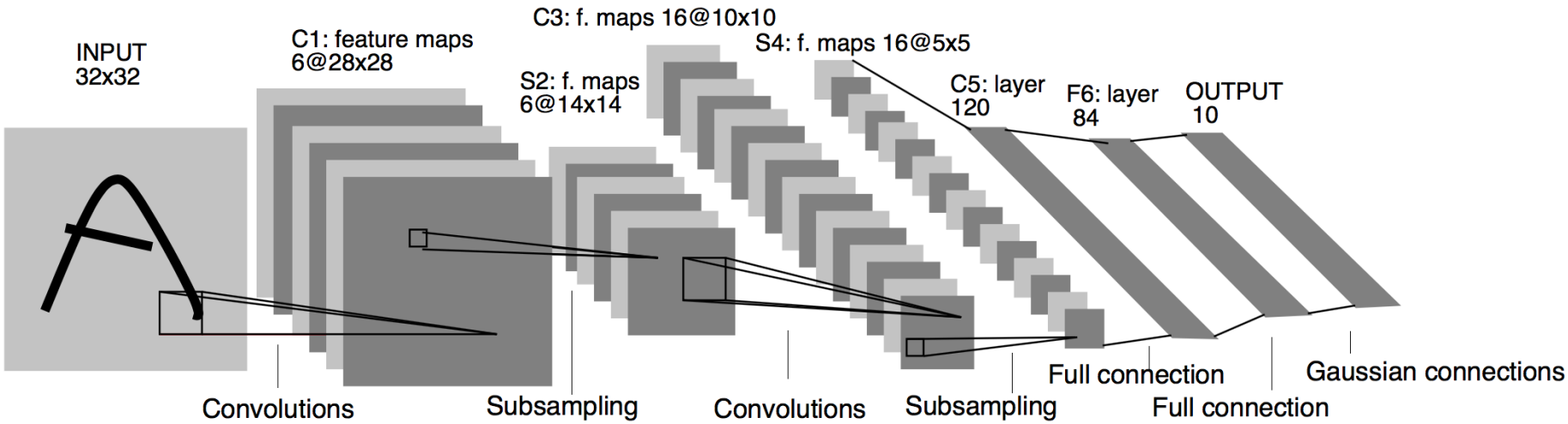
Réponse

Couche de Pooling



- Permet de **réduire la dimension** des tenseurs, pour contrebalancer la multiplication des réponses aux filtres de convolution.
- **Préserve les hautes réponses** des filtre de convolution.
- Introduit une **invariance à la translation**.
- **Pas de paramètres** à apprendre !

Un pionnier : LeNet (1998)

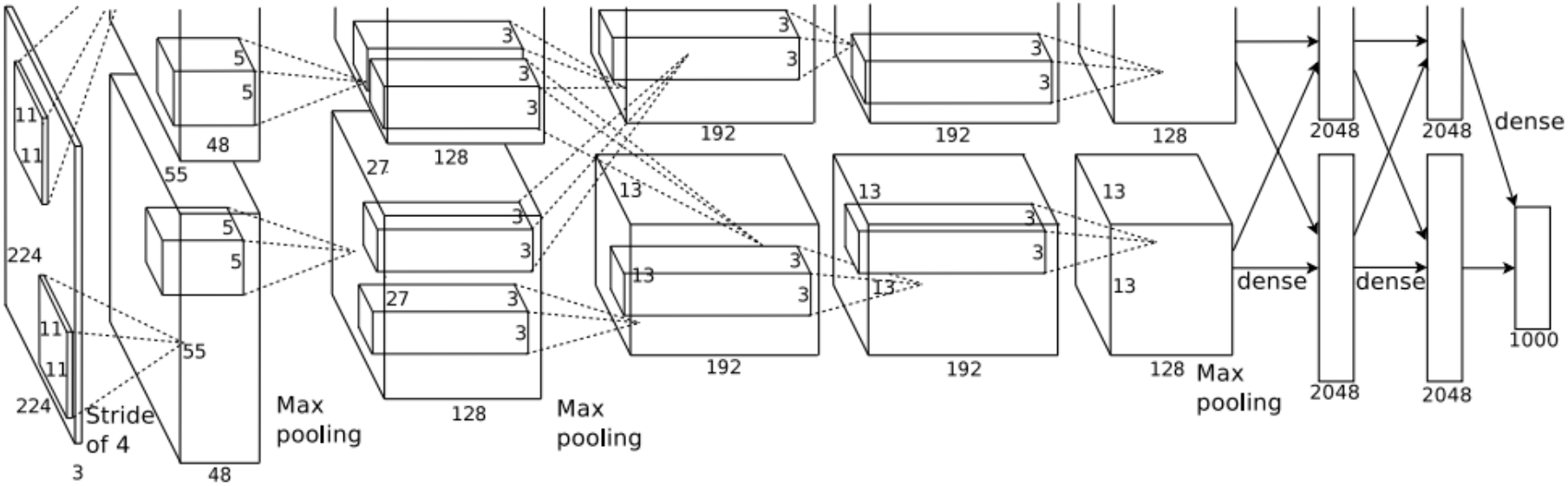


- $\simeq$ 60k paramètres
- 2 à 3 jours d'entraînement pour 20 *epochs* sur MNIST (en 1998 !).
- Fonctions d'activation sigmoïdes en majorité.

Visualisation : `https://scs.ryerson.ca/~aharley/vis/conv/`

[LeCun et al.] Gradient-based learning applied to document recognition.

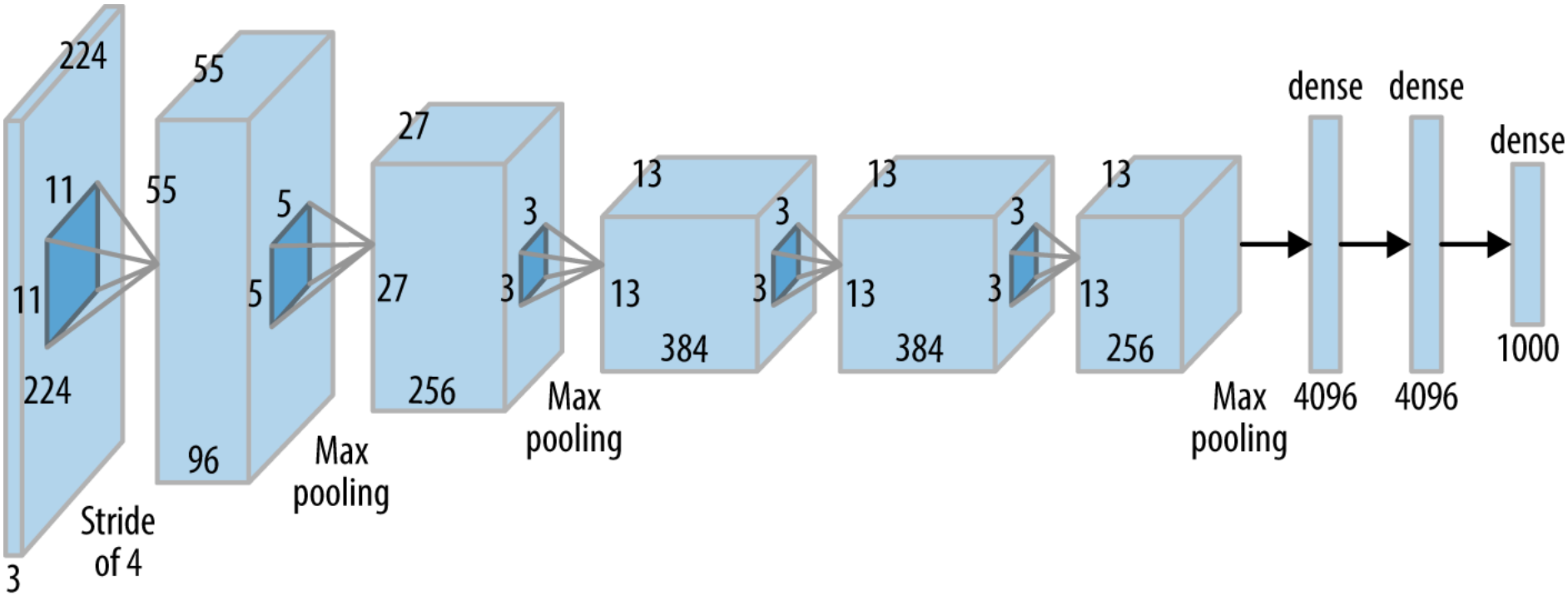
La bascule : AlexNet (2012)



- $\simeq$ 60M paramètres, 8 couches
- l'architecture est conçue pour être implémentée sur 2 GPUs.
- introduit l'usage de la fonction `reLU` comme un standard pour l'entraînement de réseaux profonds.

[Krizhevsky et al.] ImageNet Classification with Deep Convolutional Neural Networks.

La bascule : AlexNet (2012)

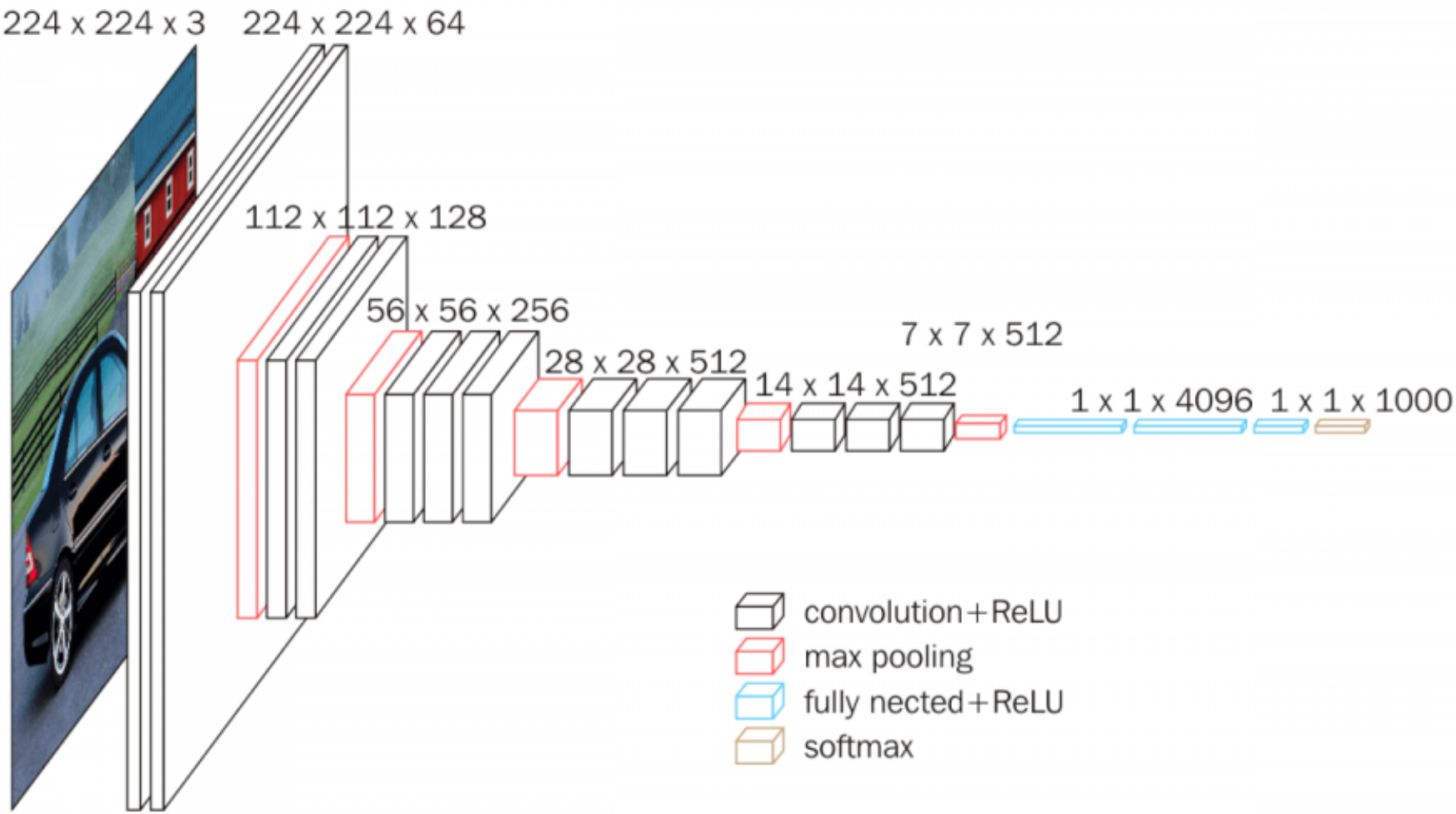


Observations :

- Diminution progressive de la taille des filtres ($11 \to 5 \to 3$)
- Diminution progressive de la taille de l'image ($224 \to 55 \to 27 \to 13$)
- Augmentation progressive du nombre de filtres ($96 \to 256 \to 384$)
- *Stride* puis *Max Pooling*

[Krizhevsky et al.] ImageNet Classification with Deep Convolutional Neural Networks.

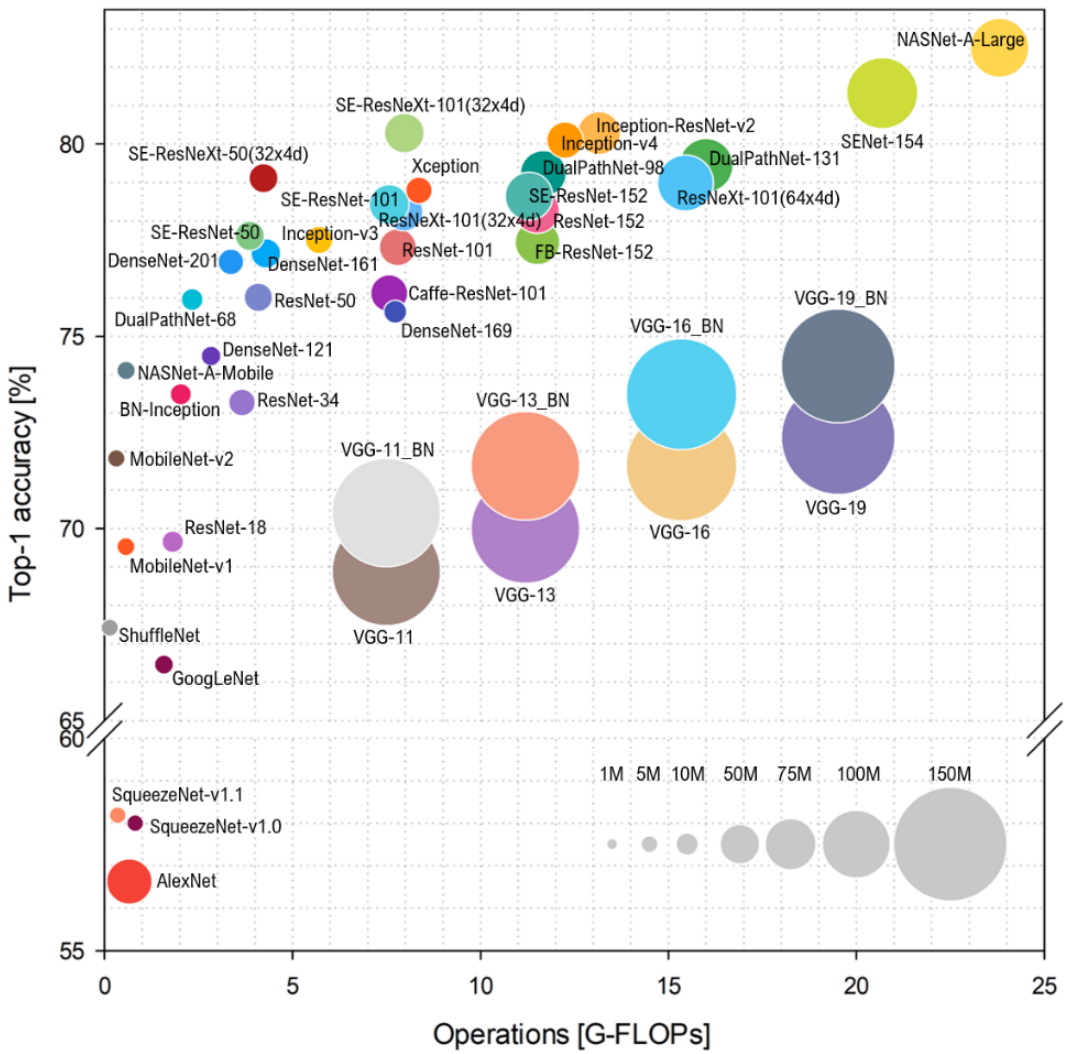
Une version "simplifiée" : VGG-16 (2014)



$\simeq$ 138M paramètres, 16 couches dans sa version standard.

[Simonyan et Zisserman] Very Deep Convolutional Networks for Large-Scale Image Recognition

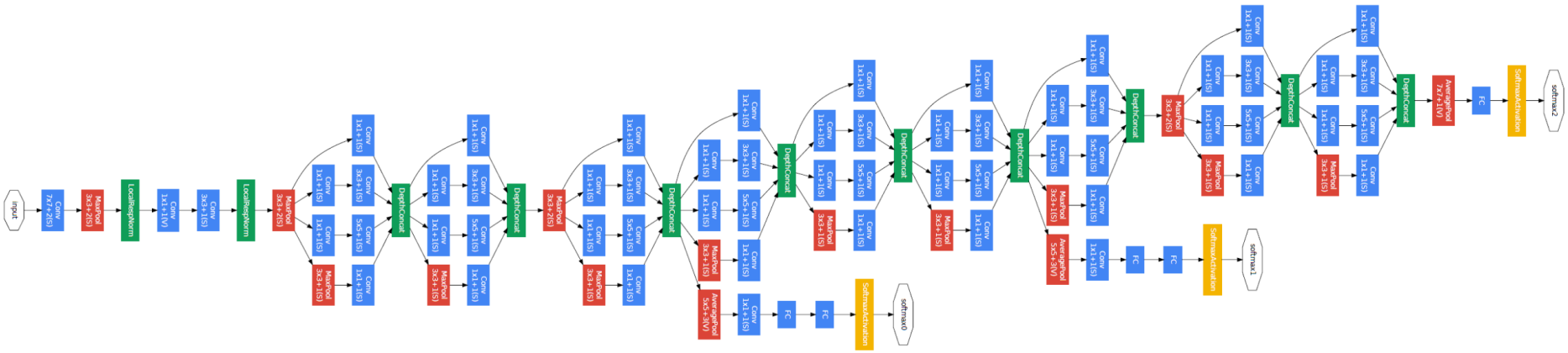
Depuis 2015



[Bianco et al.] Benchmark Analysis of Representative Deep Neural Network Architectures

Plan du cours

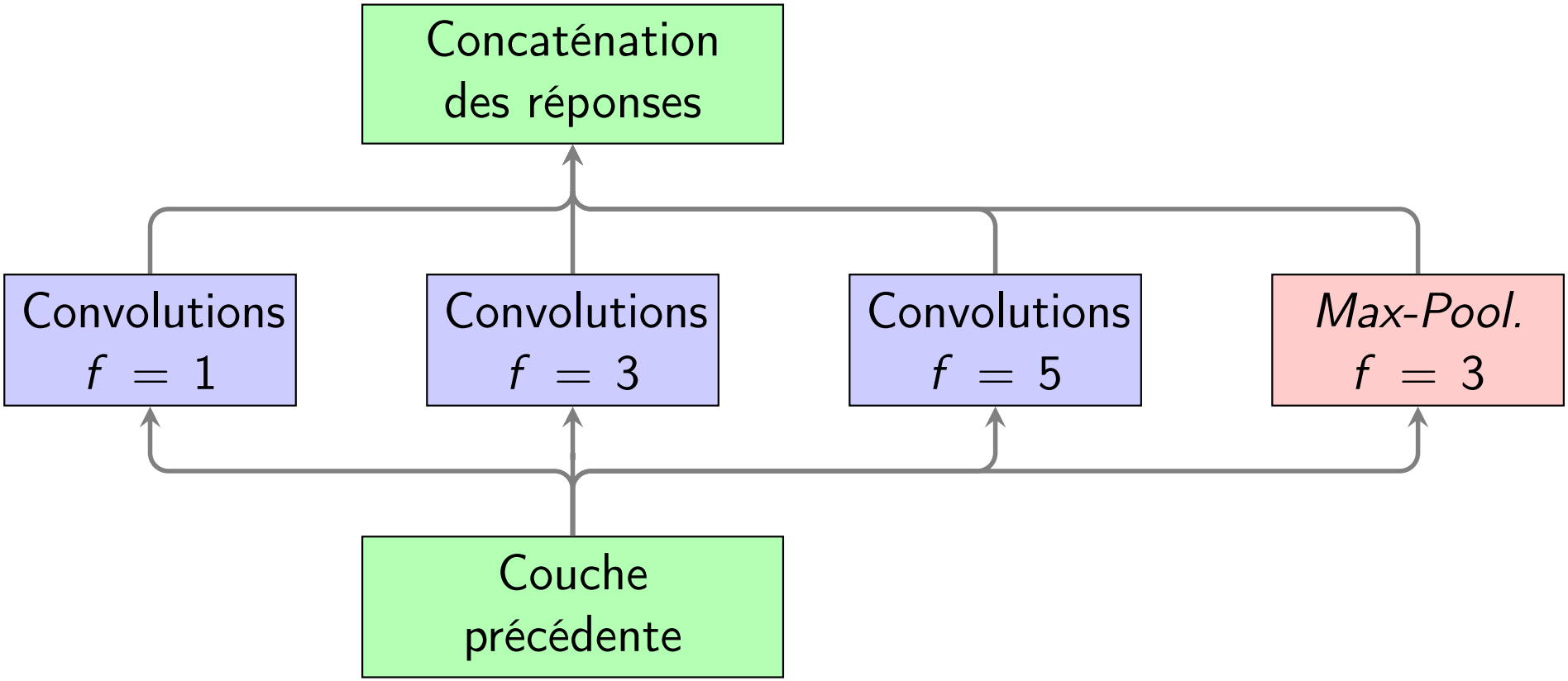
Choisir, c'est renoncer : GoogLeNet (ou Inception, 2014)



$\simeq$ 7M paramètres, 22 couches

[Szegedy et al.] Going deeper with convolutions.

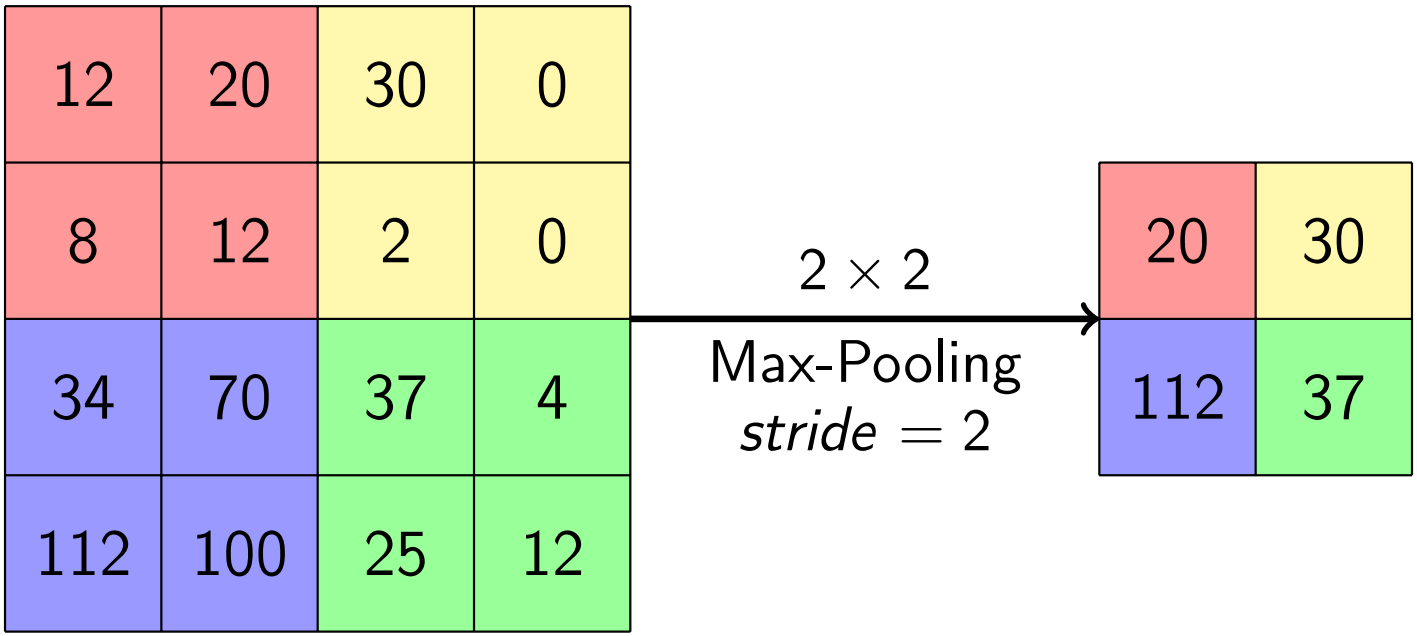
La brique de base d'Inception



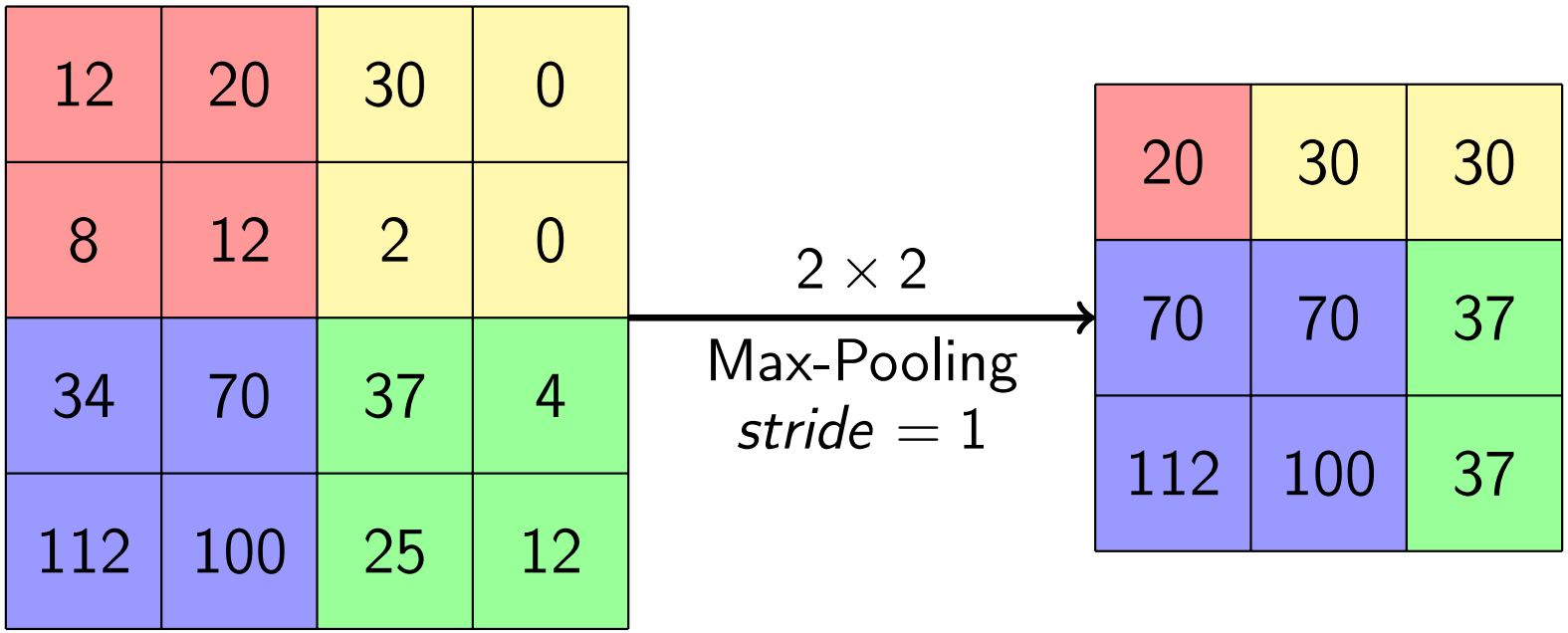
Utilisation de *padding* et d'une variante particulière du *Max-Pooling* pour conserver des tenseurs de la même dimension.

[Szegedy et al.] Going deeper with convolutions.

Couche de *Pooling* "classique"

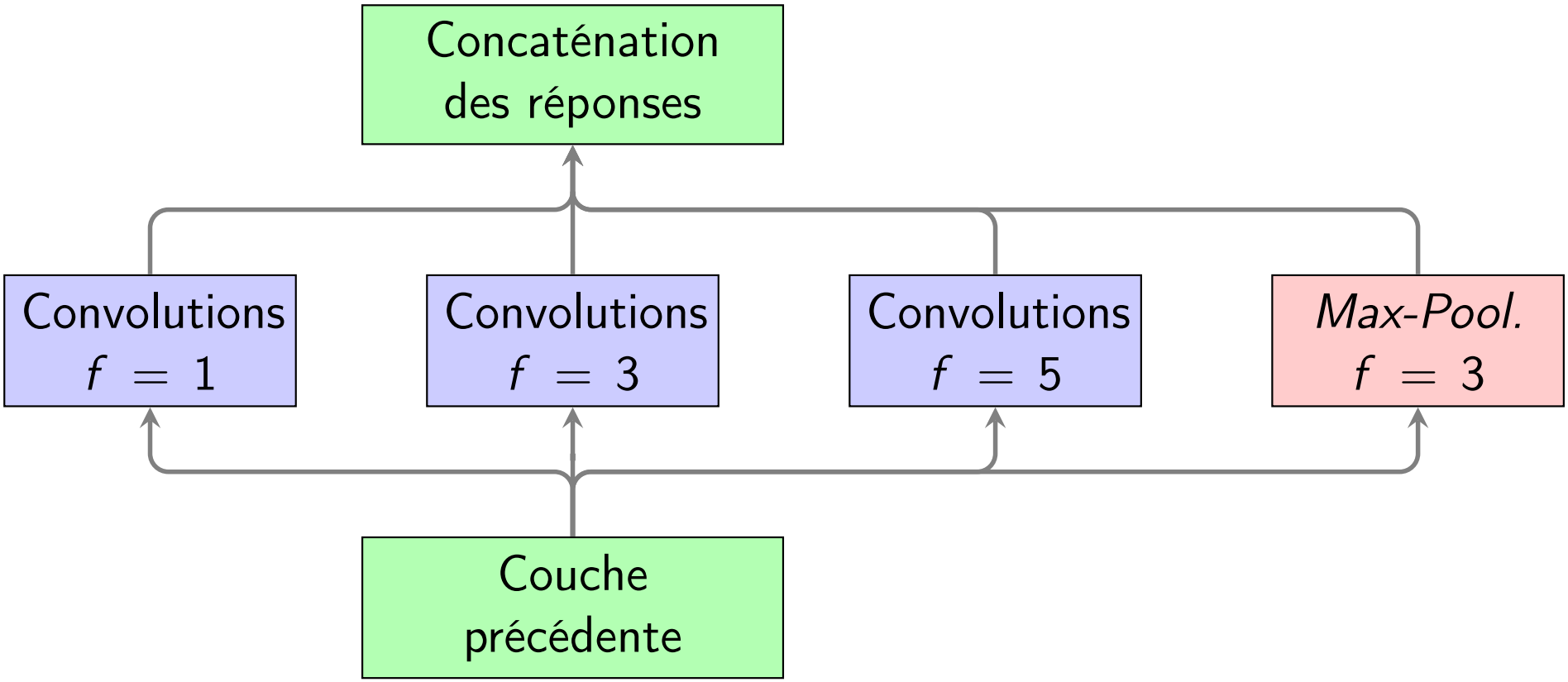


Couche de *Pooling* préservant la dimension



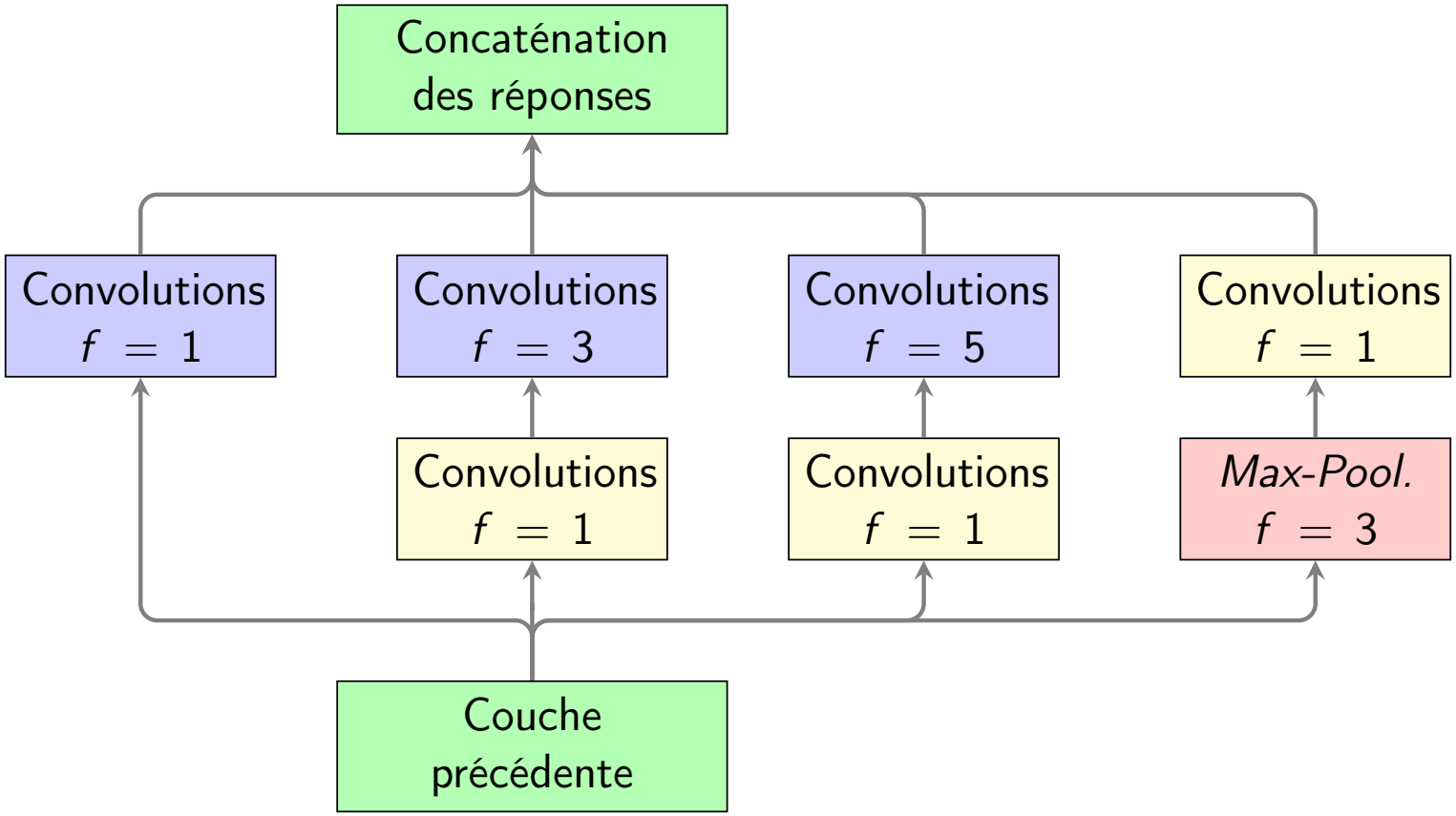
- En utilisant un *stride* de 1 et du *padding*, on peut obtenir une réponse de même dimension que l'entrée.

La brique de base d'Inception



[Szegedy et al.] Going deeper with convolutions.

La brique de base, optimisée, d'Inception



Ajout de convolutions 1×1 pour diminuer la complexité des calculs et le nombre de paramètres, mais aussi pour ajouter de la non-linéarité.

[Szegedy et al.] Going deeper with convolutions.

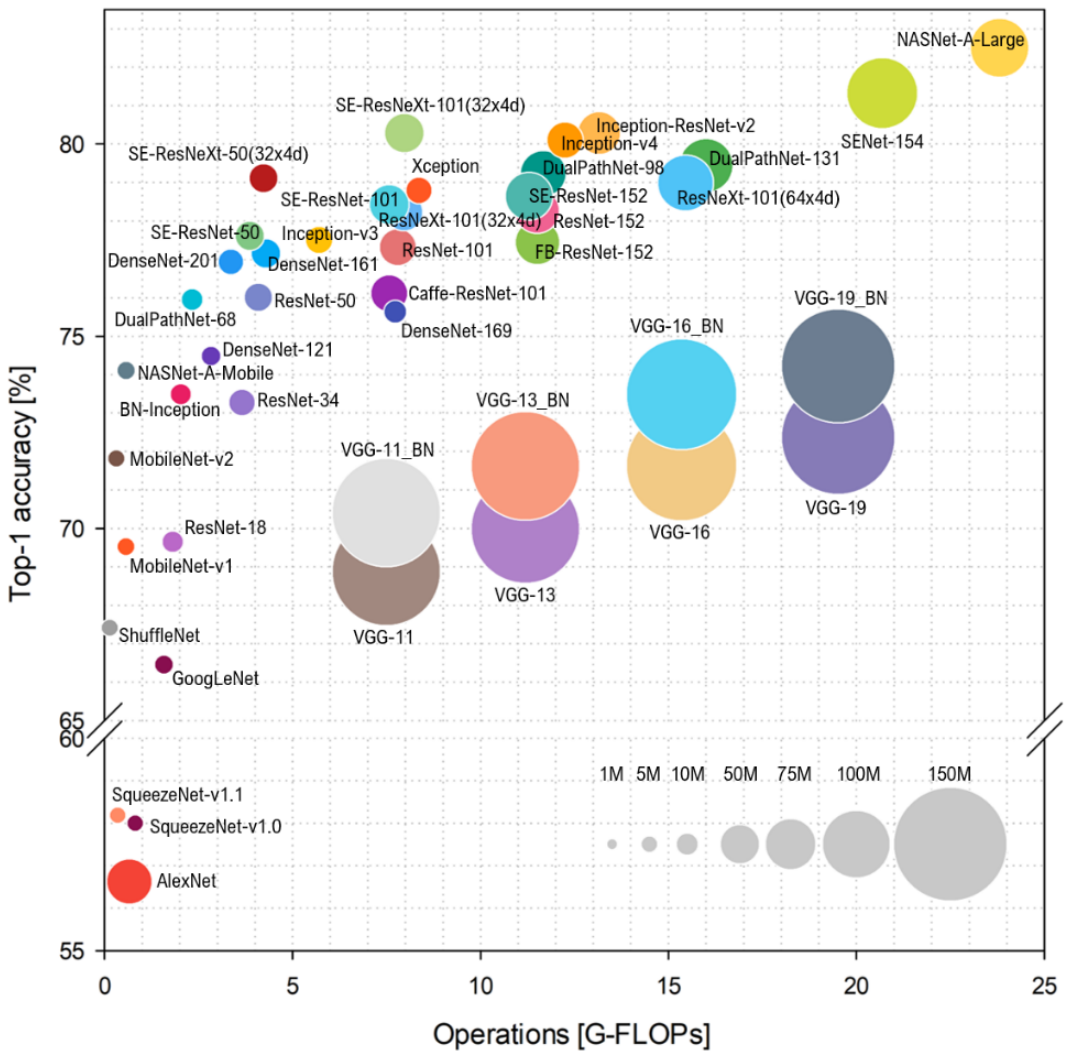
Évolutions d'Inception (v2, v3)

Nombreuses optimisations mises au point :

- Remplacement des convolutions 5×5 par 2 convolutions 3×3 successives (notion de **champ réceptif**).
- Diminution du nombre de blocs, et blocs plus "larges"
- Régularisation par *Label Smoothing*
- ...
- et plus tard encore, combinaison avec ResNet...

[Szegedy et al.] Rethinking the Inception Architecture for Computer Vision

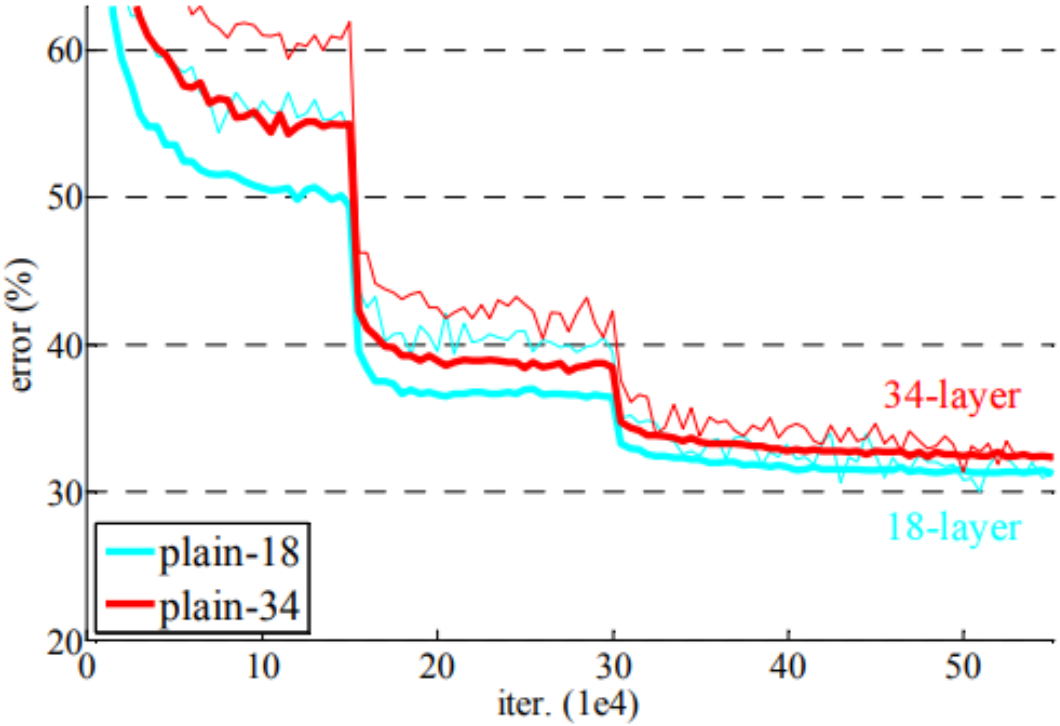
Benchmark



[Bianco et al.] Benchmark Analysis of Representative Deep Neural Network Architectures

Plan du cours

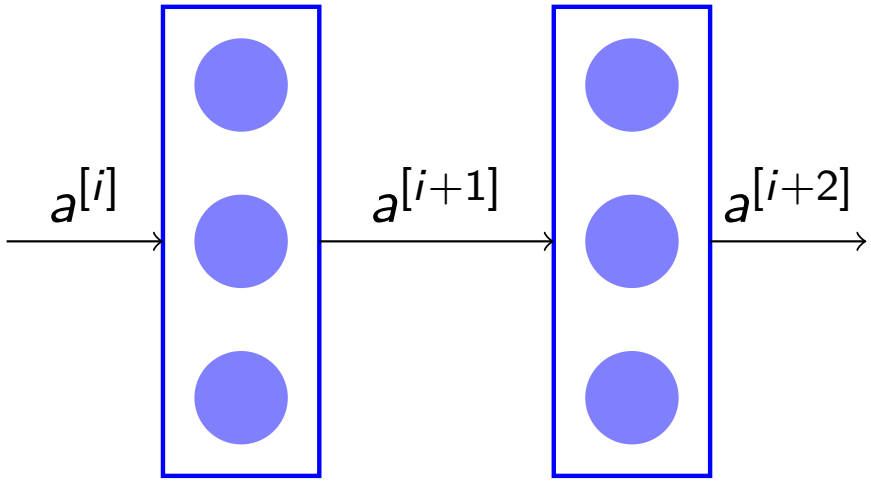
L'entraînement des réseaux profonds



Les réseaux plus profonds devraient en théorie permettre d'approximer de plus en plus efficacement l'ensemble d'apprentissage. En pratique, la difficulté à optimiser les paramètres augmente avec la profondeur.

[He et al.] Deep Residual Learning for Image Recognition

Blocs résiduels



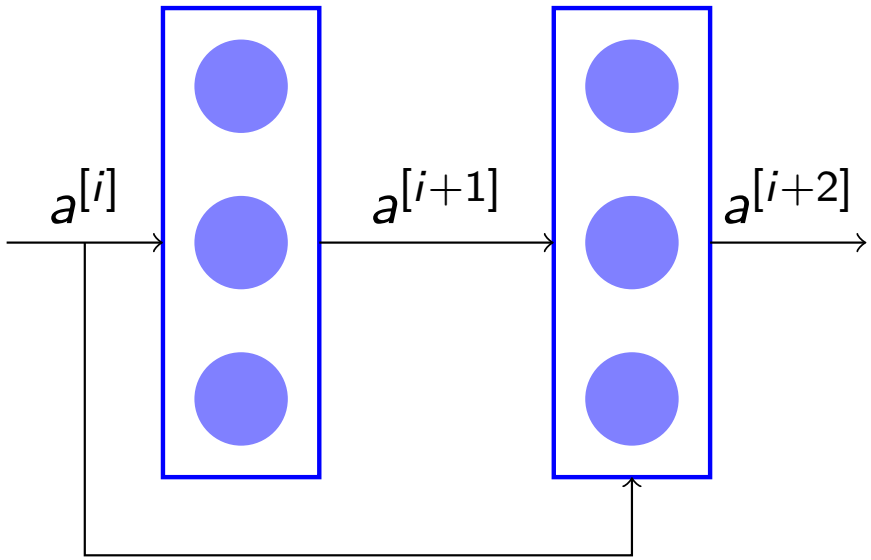
Cette vue abstrait les liaisons synaptiques entre les 2 couches $i+1$ et $i+2$. On a ici par exemple :

$$a^{[i+1]} = \texttt{ReLU}(W^{[i+1]}a^{[i]} + b^{[i+1]})$$

et

$$a^{[i+2]} = \texttt{ReLU}(W^{[i+2]}a^{[i+1]} + b^{[i+2]})$$

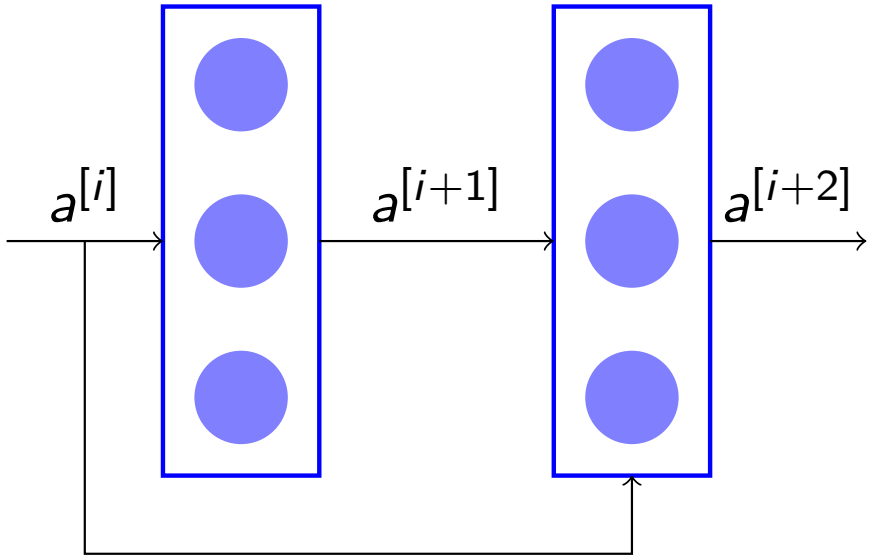
Blocs résiduels



On peut ajouter une connexion comme présenté ci-dessus (*skip connection*) pour former un **bloc résiduel**, d'équation :

$$a^{[i+2]} = \texttt{ReLU}(W^{[i+2]}a^{[i+1]} + b^{[i+2]} + a^{[i]})$$

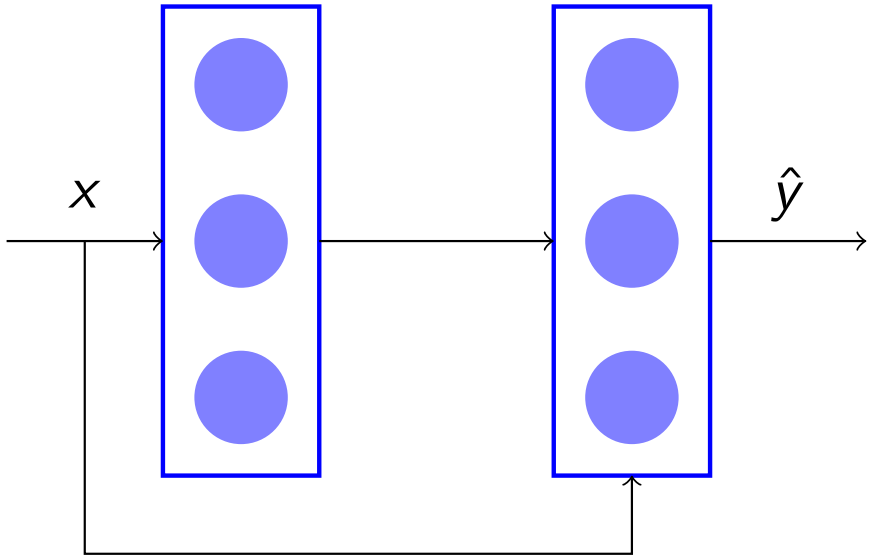
Blocs résiduels



Une condition importante est que les dimensions de $a^{[i]}$ et $a^{[i+2]}$ soient identiques !

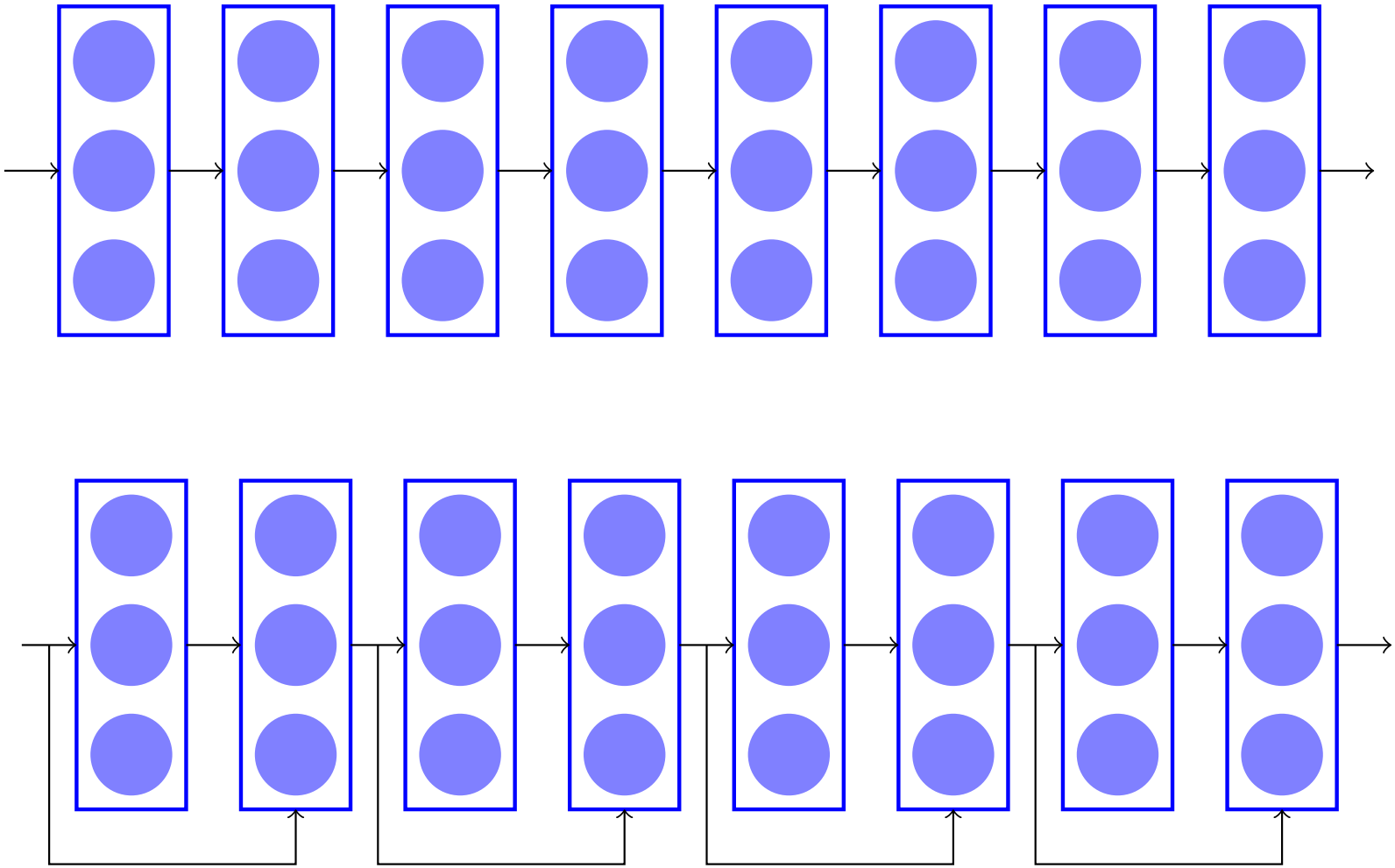
On utilise donc souvent des convolutions qui conservent la dimension (*same*) dans les réseaux résiduels.

Blocs résiduels

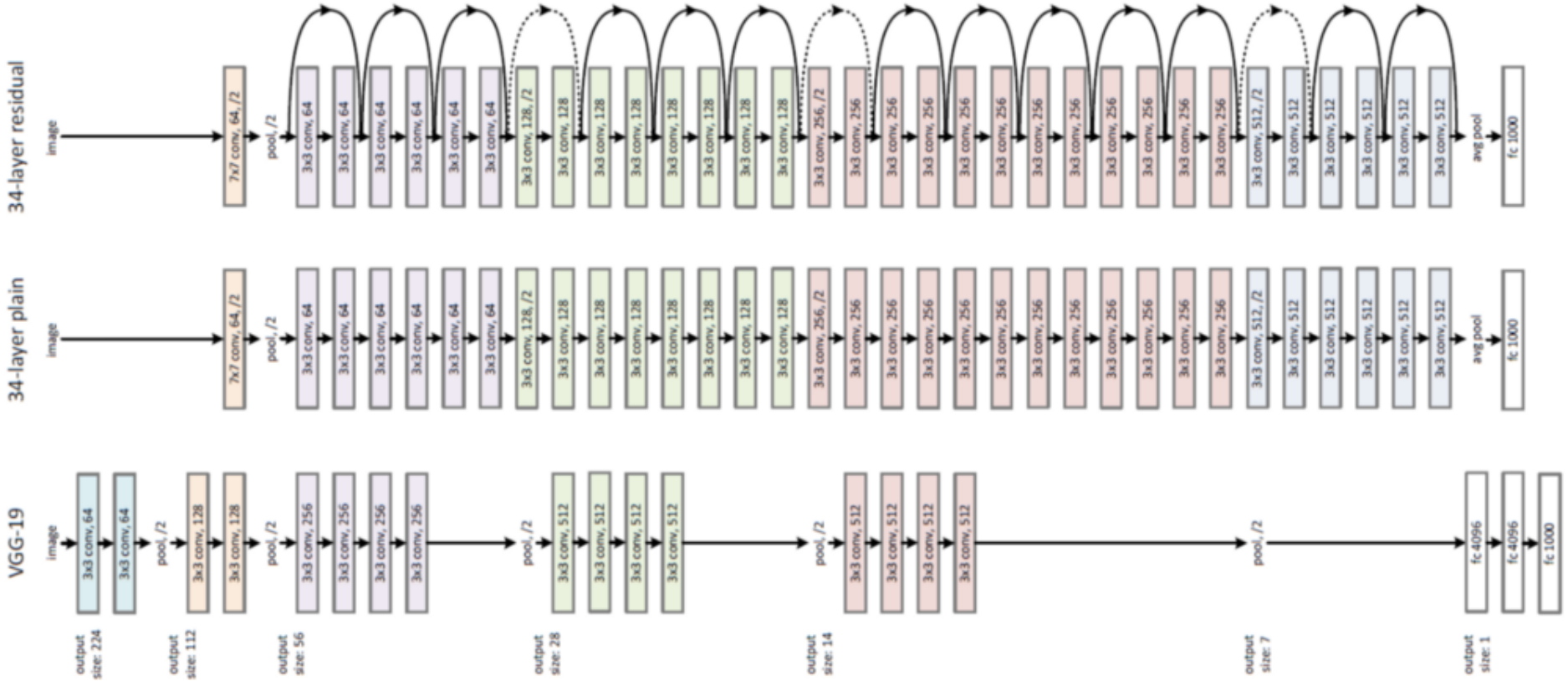


Alors que l'entraînement vise d'ordinaire à estimer une fonction F telle que $\hat{y} = F(x)$, on cherche ici la fonction telle que : $\hat{y} = F(x) + x$.
En d'autres termes, la fonction F estime le **résidu** $\hat{y} - x$ (d'où le nom de bloc résiduel).

Rendre "résiduel" un réseau de neurones

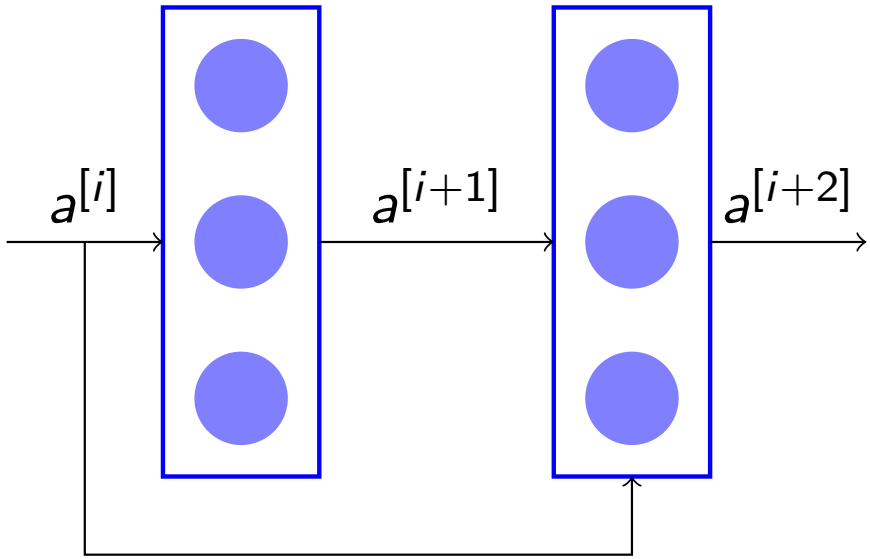


ResNet (2015)



[He et al.] Deep Residual Learning for Image Recognition

Une bonne propriété des blocs résiduels



$$a^{[i+2]} = \texttt{ReLU}(W^{[i+2]}a^{[i+1]} + b^{[i+2]} + a^{[i]})$$

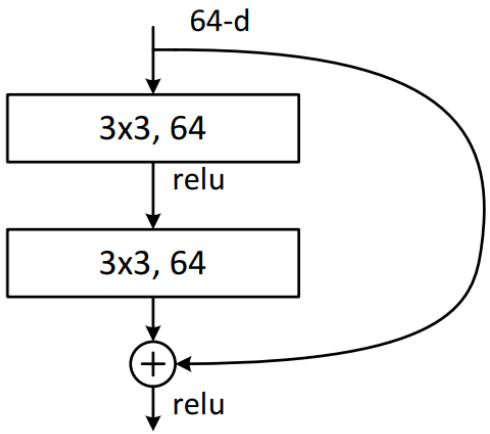
Si les poids synaptiques $W^{[i+2]}$ et les biais $b^{[i+2]}$ sont proches de 0, alors :

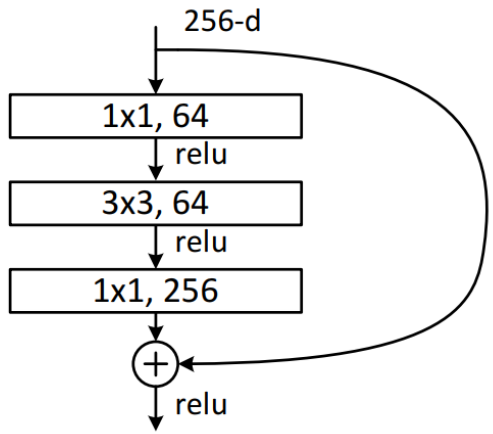
$$a^{[i+2]} \simeq \texttt{ReLU}(a^{[i]}) \simeq a^{[i]}$$

La fonction identité est facile à modéliser par un bloc résiduel. A l'initialisation, le réseau est "conditionné" à prédire l'identité.

ResNet (2015)

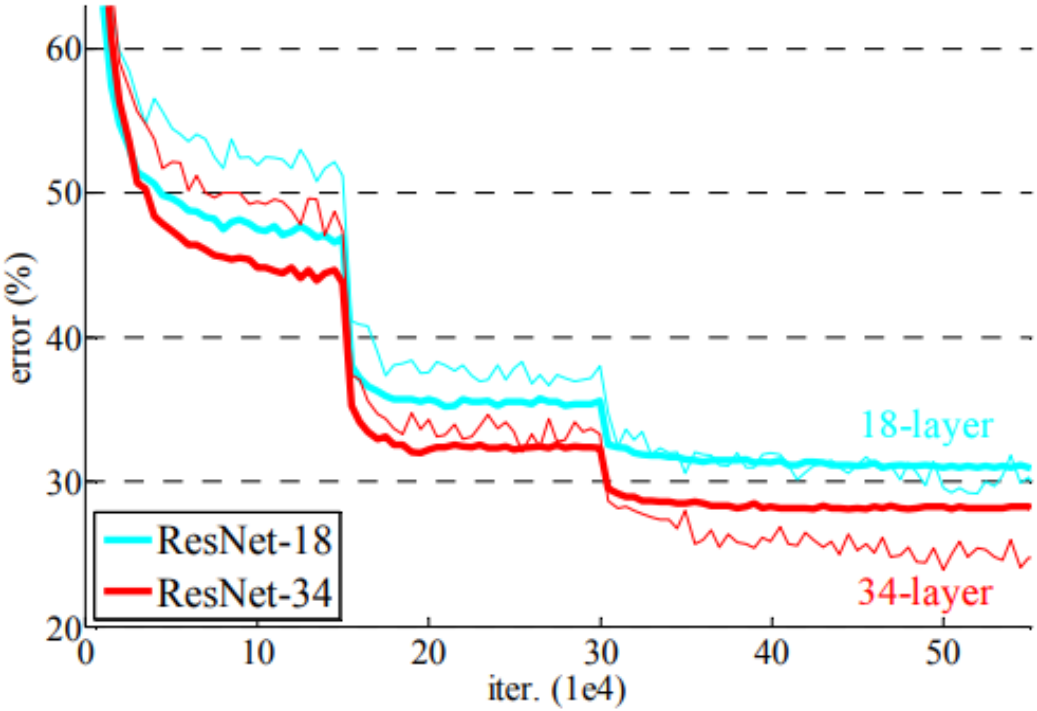
Les blocs résiduels ont permis aux auteurs d'entraîner des réseaux de plusieurs centaines de couches. Le réseau qui a remporté le *challenge* `ImageNet` en 2015 comptait d'ailleurs 152 couches.
Les problèmes de ces architectures très profondes ne sont plus liés à l'optimisation mais aux performances, ce qui a motivé les auteurs à introduire des blocs résiduels d'étranglement (*bottlenecks blocks*) :





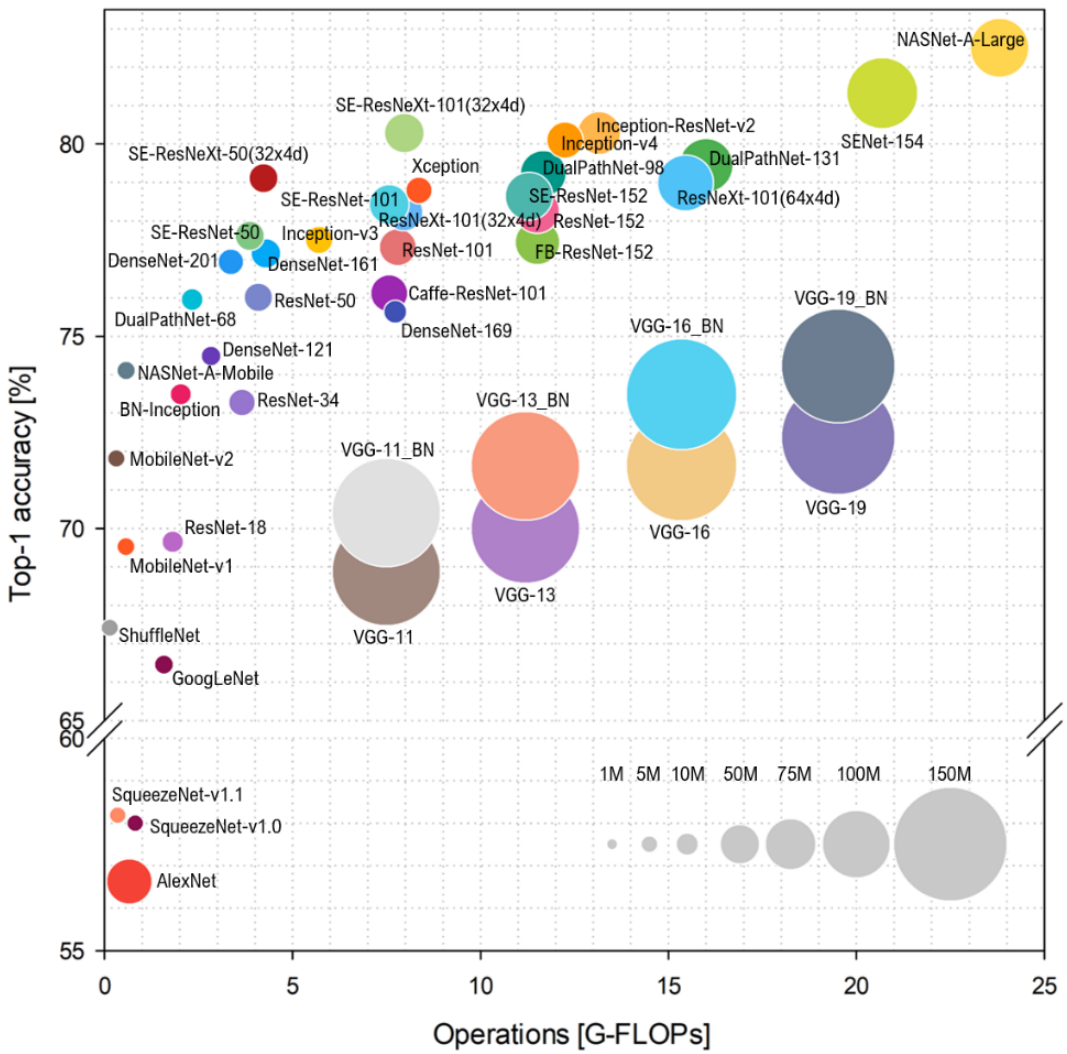
[He et al.] Deep Residual Learning for Image Recognition

Résultats obtenus avec ResNet



[He et al.] Deep Residual Learning for Image Recognition

Benchmark

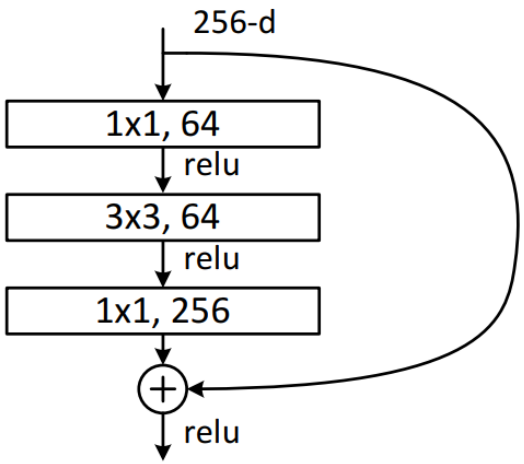


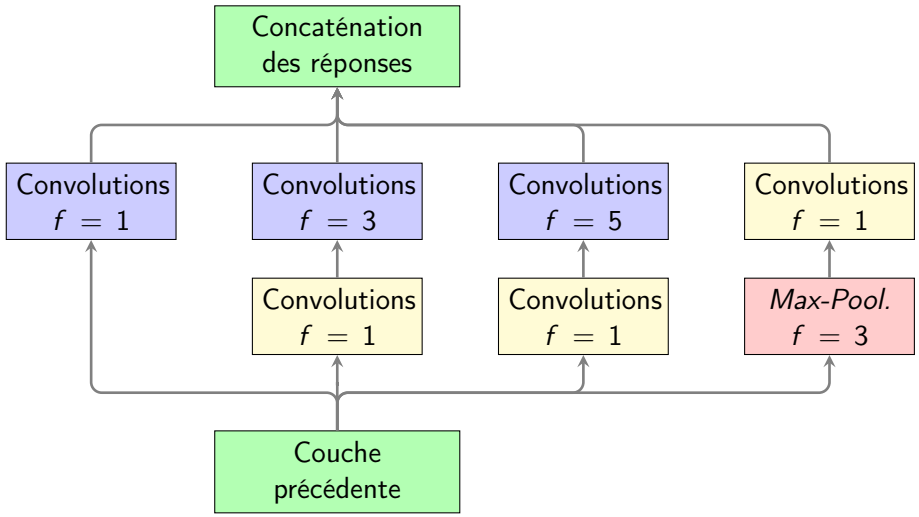
[Bianco et al.] Benchmark Analysis of Representative Deep Neural Network Architectures

Plan du cours

Variantes de ResNet et Inception

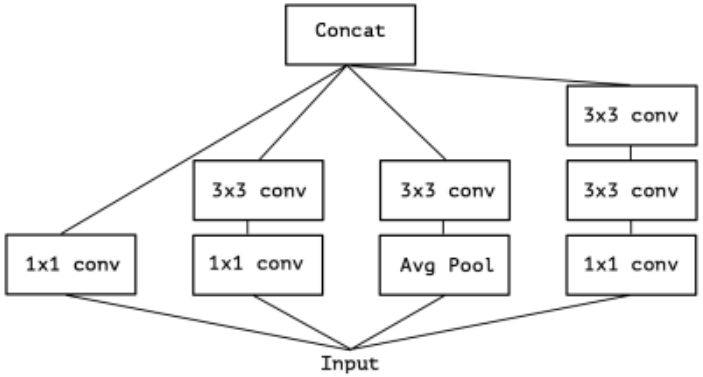
Après 2015, de nombreux travaux ont cherché à mettre au point des architectures reprenant les bonnes propriétés d'Inception (nombre de paramètres réduit pour une performance maintenue) et de ResNet (connexions résiduelles).



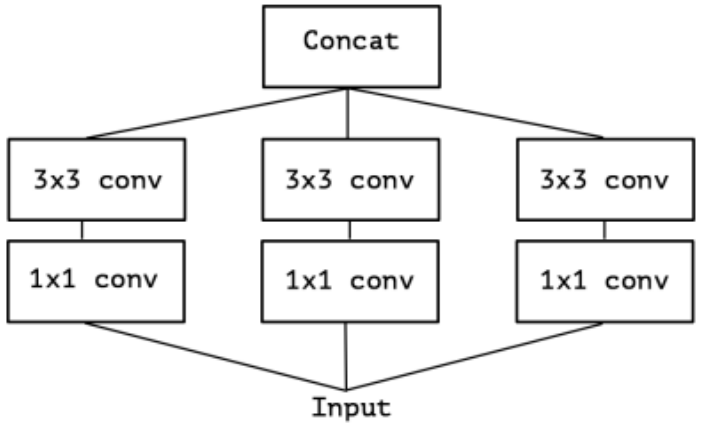


Xception

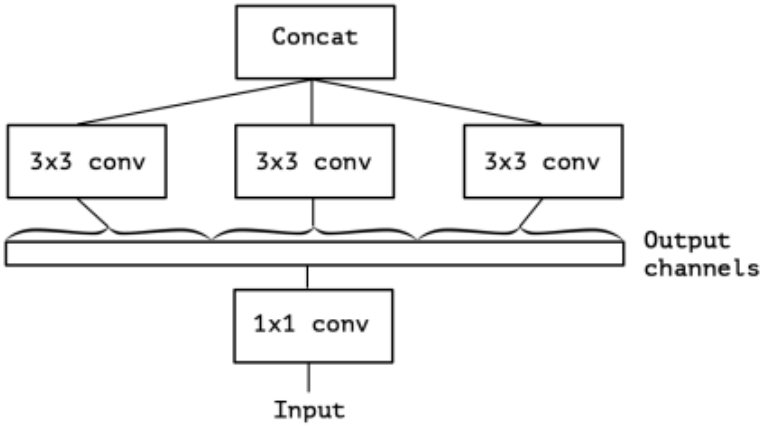
Point de départ : une reformulation et généralisation du bloc Inception.



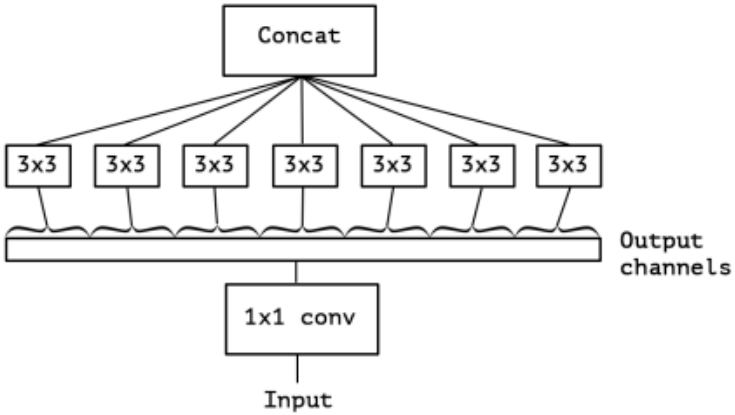
Bloc Inception v3



Bloc Inception simplifié



Bloc Inception reformulé



Bloc Inception généralisé

[Chollet] Xception : Deep Learning with Depthwise Separable Convolutions

Depthwise separable convolution

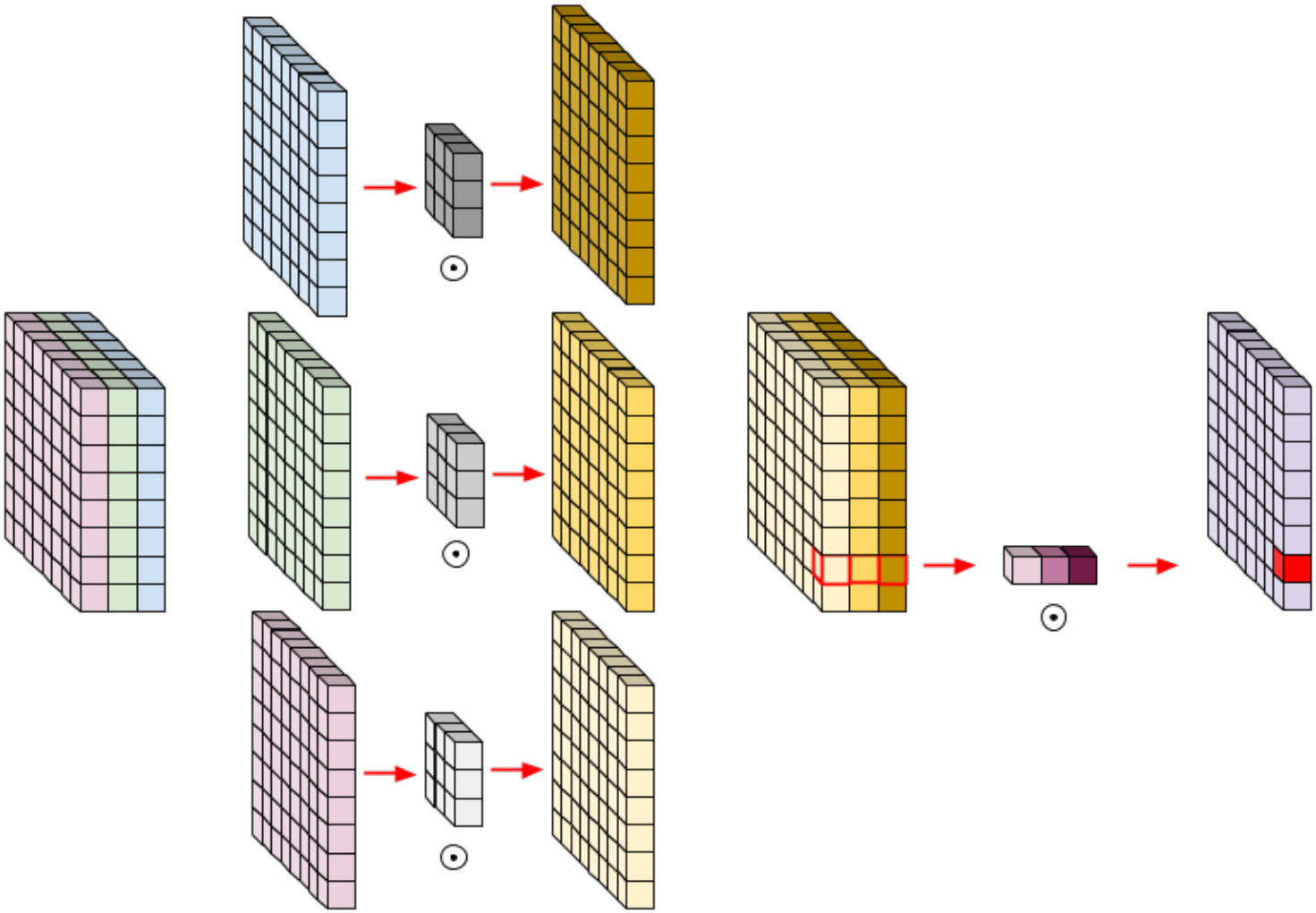
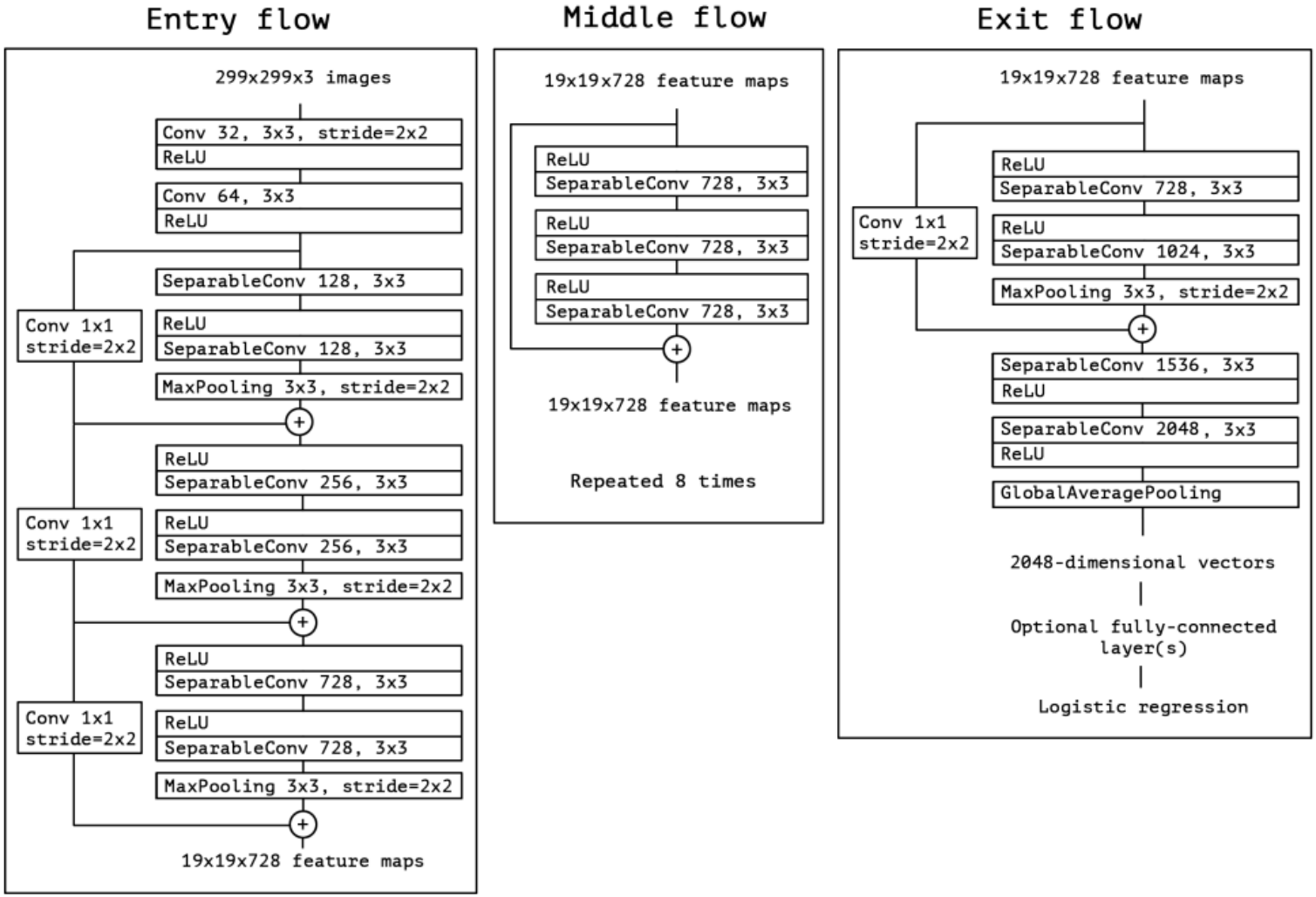


Image de `https://eli.thegreenplace.net/2018/depthwise-separable-convolutions-for-machine-learning/`

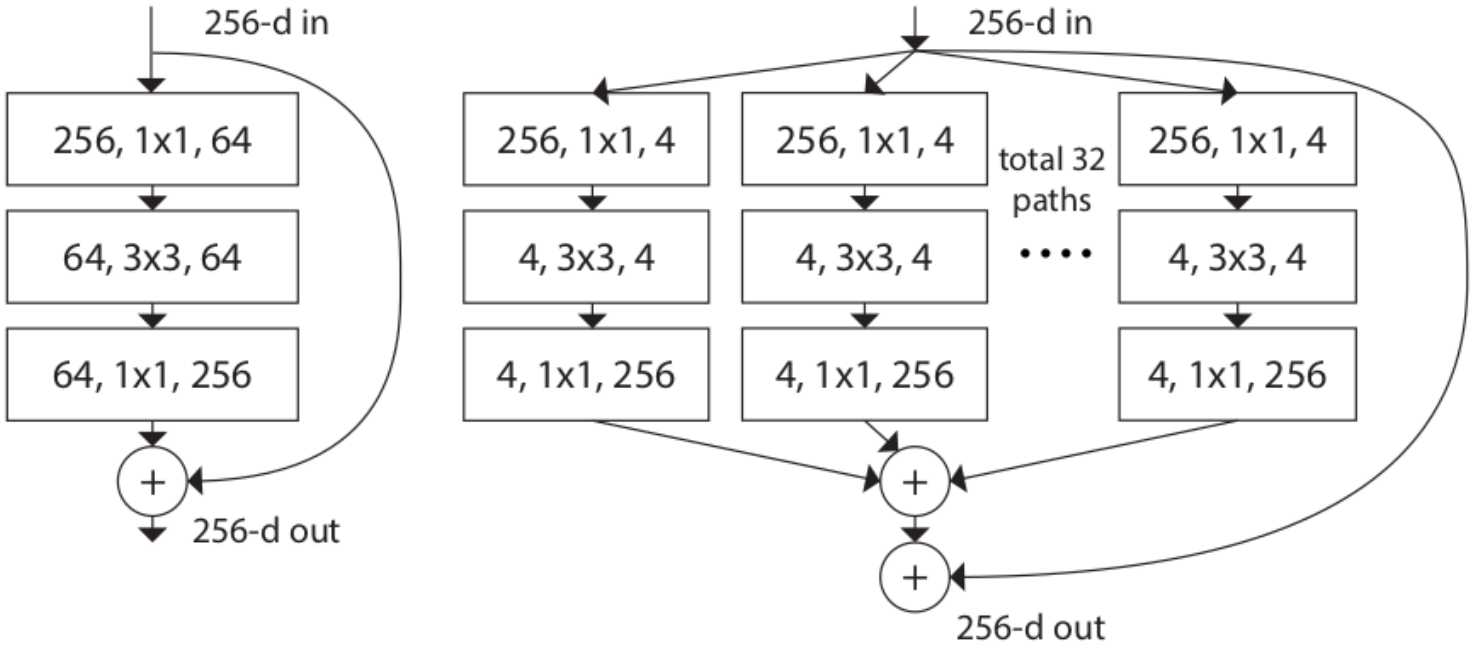
Xception



[Chollet] Xception : Deep Learning with Depthwise Separable Convolutions

ResNeXt

ResNeXt part d'une idée similaire à Xception (les deux articles ont été déposés sur `Arxiv` à une semaine d'intervalle) mais avec une implantation différente :

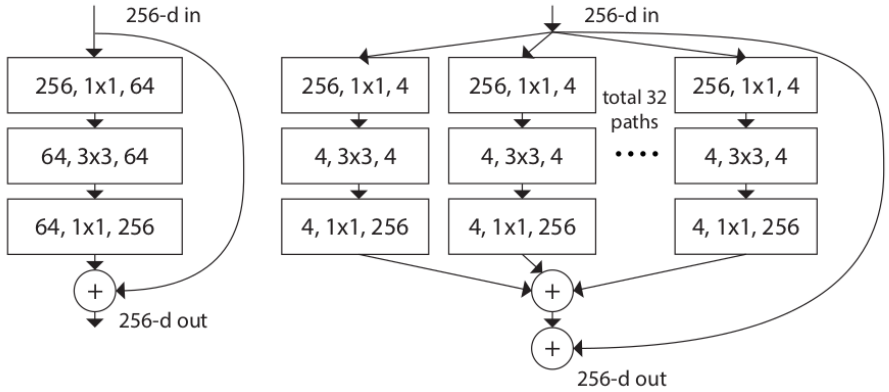


Les auteurs proposent d'intégrer les idées d'Inception à un bloc *bottleneck* de Resnet.

[Xie et al] Aggregated Residual Transformations for Deep Neural Networks

ResNeXt

Un des objectifs affichés dans ResNext est de rationaliser l'ensemble des hyperparamètres de bloc.



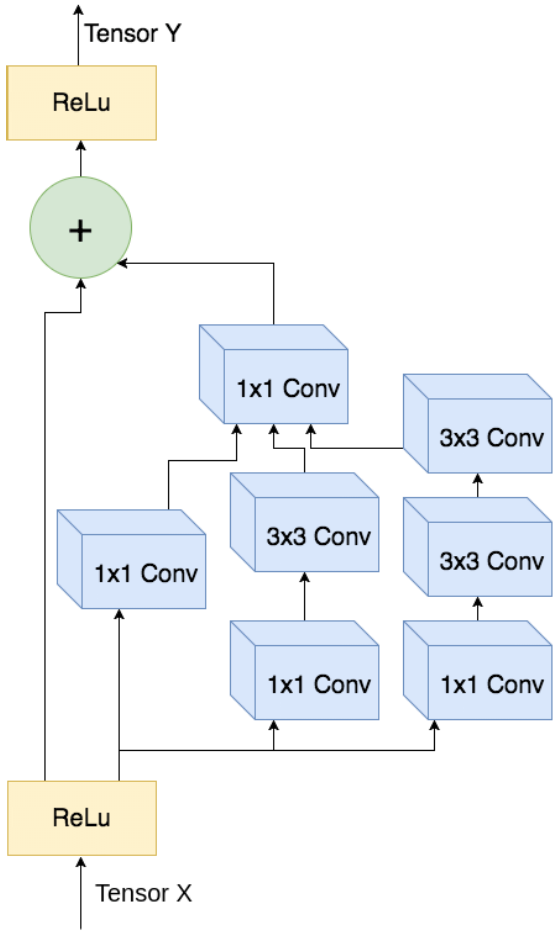
cardinality C	1	2	4	8	32
width of bottleneck d	64	40	24	14	4

On peut lier le nombre de chemins et le nombre de filtres de la convolution *bottleneck* pour obtenir un nombre de paramètres approximativement constant. Ceci limite le nombre d'hyperparamètres de l'architecture.

[Xie et al] Aggregated Residual Transformations for Deep Neural Networks

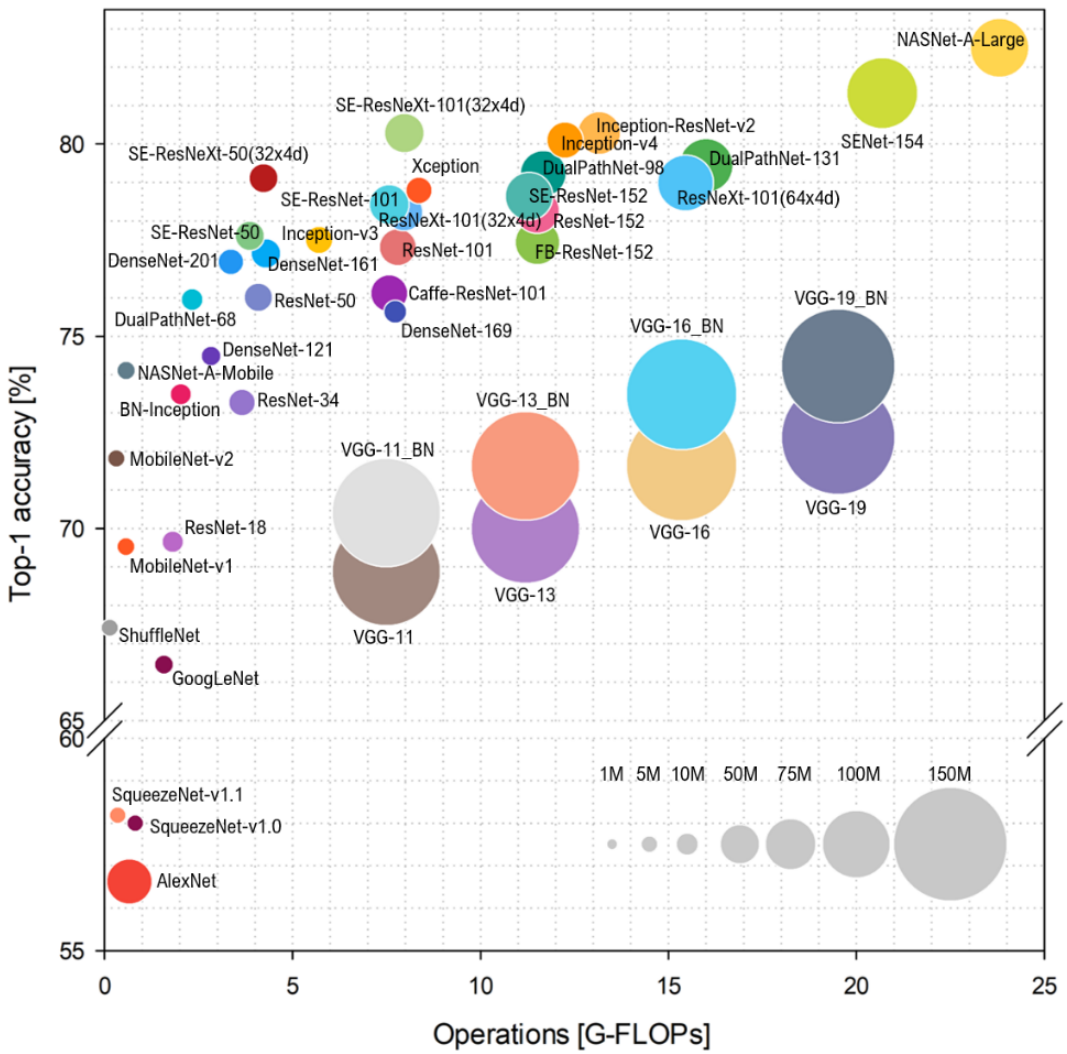
Inception-ResNet

Enfin, une autre alternative consiste simplement à rajouter une connexion résiduelle aux blocs Inception.



[Szegedy et al] Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning

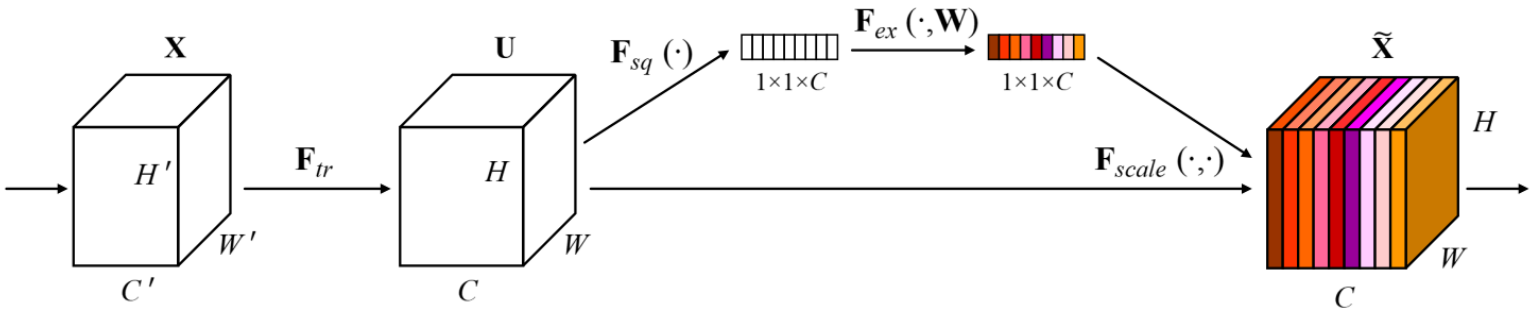
Comparaison



[Bianco et al.] Benchmark Analysis of Representative Deep Neural Network Architectures

Plan du cours

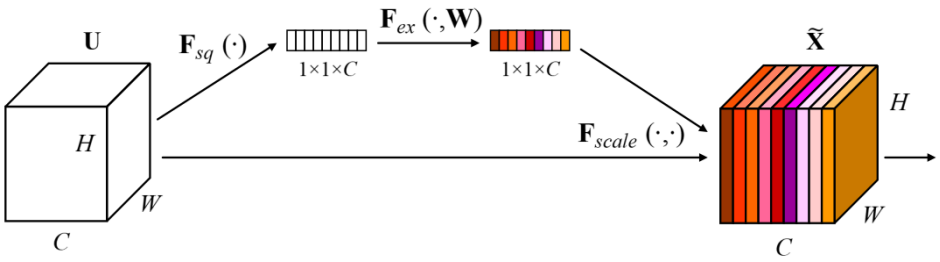
Squeeze and Excitation

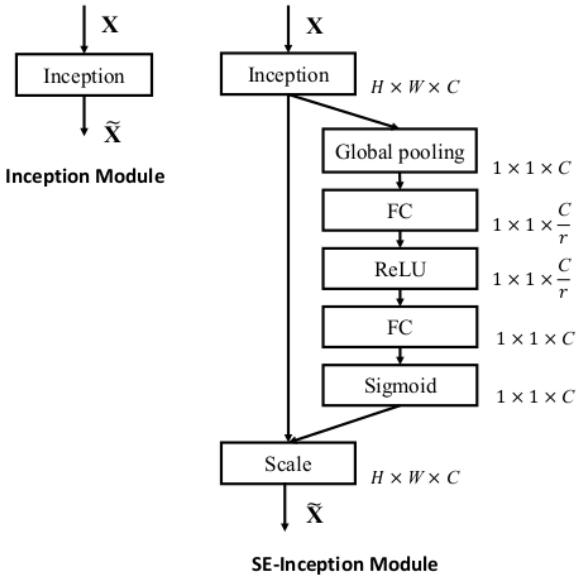


Plus que des évolutions architecturales majeures, certains travaux s'attachent à définir des blocs qui, ajoutés aux architectures existantes, permettent d'en améliorer les performances.

[Hu et al.] Squeeze-and-Excitation Networks

Le bloc *Squeeze and Excitation*





Le bloc SE permet de mettre en valeur certaines cartes de caractéristiques au détriment d'autres.

[Hu et al.] Squeeze-and-Excitation Networks

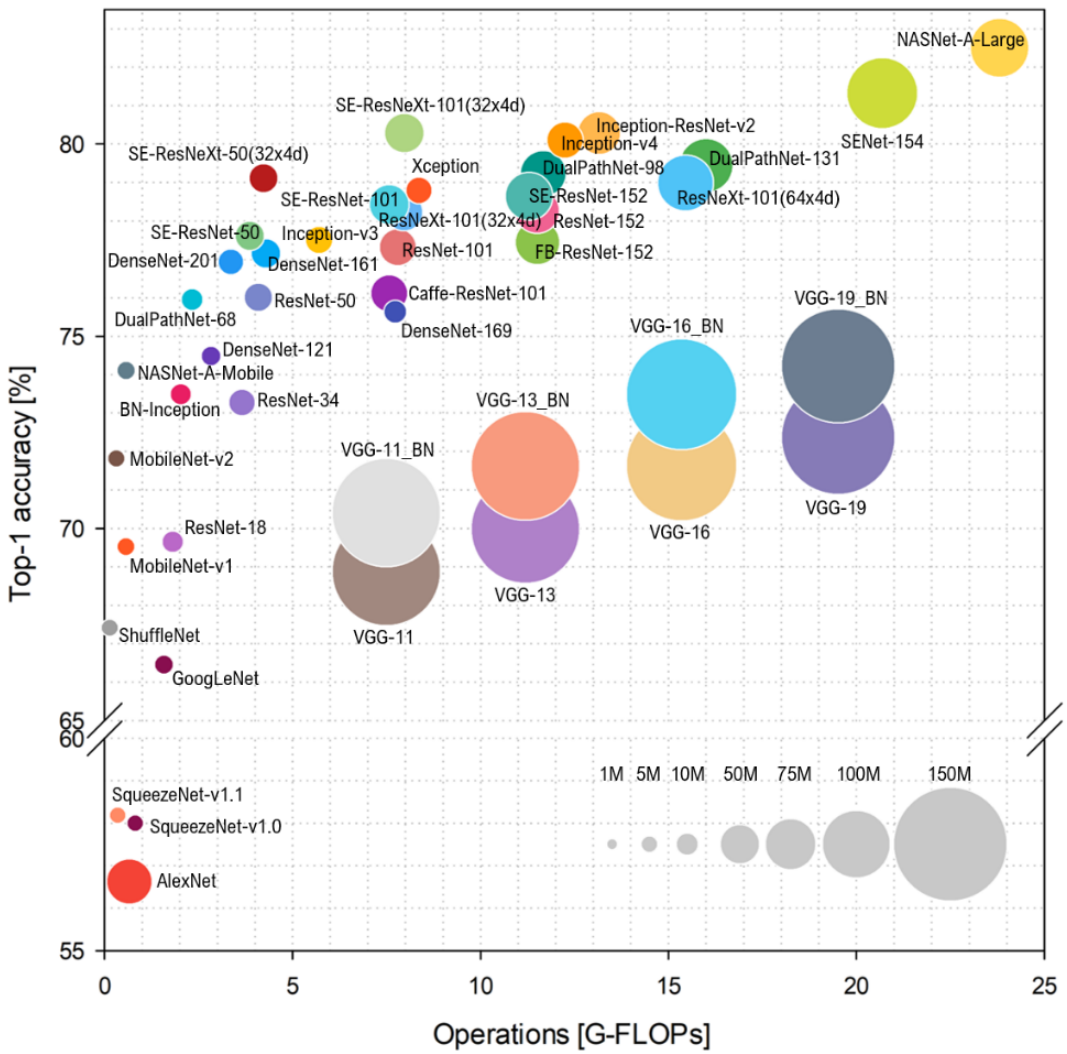
Le bloc *Squeeze and Excitation*

	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 [13]	24.7	7.8	24.80	7.48	3.86	$23.29_{(1.51)}$	$6.62_{(0.86)}$	3.87
ResNet-101 [13]	23.6	7.1	23.17	6.52	7.58	$22.38_{(0.79)}$	$6.07_{(0.45)}$	7.60
ResNet-152 [13]	23.0	6.7	22.42	6.34	11.30	$21.57_{(0.85)}$	$5.73_{(0.61)}$	11.32
ResNeXt-50 [19]	22.2	-	22.11	5.90	4.24	$21.10_{(1.01)}$	$5.49_{(0.41)}$	4.25
ResNeXt-101 [19]	21.2	5.6	21.18	5.57	7.99	$20.70_{(0.48)}$	$5.01_{(0.56)}$	8.00
VGG-16 [11]	-	-	27.02	8.81	15.47	$25.22_{(1.80)}$	$7.70_{(1.11)}$	15.48
BN-Inception [6]	25.2	7.82	25.38	7.89	2.03	$24.23_{(1.15)}$	$7.14_{(0.75)}$	2.04
Inception-ResNet-v2 [21]	$19.9^{\dagger}$	$4.9^{\dagger}$	20.37	5.21	11.75	$19.80_{(0.57)}$	$4.79_{(0.42)}$	11.76

Le bloc SE améliore systématiquement les performances des réseaux existants, pour un nombre d'opérations supplémentaire minimal.

[Hu et al.] Squeeze-and-Excitation Networks

Comparaison



[Bianco et al.] Benchmark Analysis of Representative Deep Neural Network Architectures

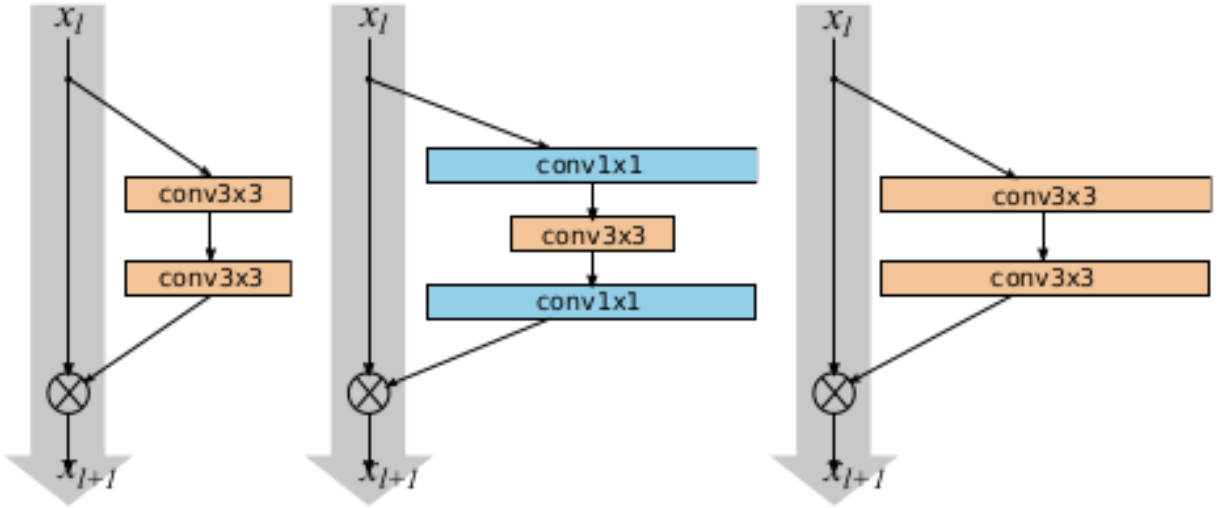
Plan du cours

Hyperparamétrisation d'architectures

La plupart des architectures présentées précédemment sont mises au point pour illustrer certaines idées, mais conservent un large nombre d'hyperparamètres définis manuellement.
Dans cette dernière section nous allons présenter quelques travaux populaires qui cherchent à systématiser la mise au point de topologies optimales de réseaux.

WideResNet

Les connexions résiduelles permettent d'entraîner des réseaux très profonds et très performants (cf. ResNet).



(a) basic (b) bottleneck (c) basic-wide

Dans WideResNet, les auteurs questionnent le compromis entre profondeur et largeur (nombre de filtres) des réseaux résiduels.

[Zagoruyko et al.] Wide Residual Networks

WideResNet

Les calculs des WideResNet sont plus facilement parallélisables et donc moins coûteux. Les meilleurs résultats sont obtenus avec des réseaux peu profonds (< 50 couches) mais plus larges (jusqu'à un facteur 10).

Dataset	model	dropout	test perf.
CIFAR-10	WRN-40-10	✓	3.8%
CIFAR-100	WRN-40-10	✓	18.3%
SVHN	WRN-16-8	✓	1.54%
ImageNet (single crop)	WRN-50-2-bottleneck		21.9% top-1, 5.79% top-5
COCO test-std	WRN-34-2		35.2 mAP

Dans la table, WRN-N-k signifie un réseau résiduel à N couches et appliquant un facteur multiplicatif k sur la largeur originellement décrite des réseaux résiduels.

[Zagoruyko et al.] Wide Residual Networks

ResNet Strikes Back

	Previous approaches					Ours		
Procedure →	ResNet	PyTorch	FixRes	DeiT	FAMS (×4)	A1	A2	A3
Reference	[13]	[1]	[48]	[45]	[10]			
Train Res	224	224	224	224	224	224	224	160
Test Res	224	224	224	224	224	224	224	224
Epochs	90	90	120	300	400	600	300	100
# of forward pass	450k	450k	300k	375k	500k	375k	188k	63k
Batch size	256	256	512	1024	1024	2048	2048	2048
Optimizer	SGD-M	SGD-M	SGD-M	AdamW	SGD-M	LAMB	LAMB	LAMB
LR	0.1	0.1	0.2	1×10^{-3}	2.0	5×10^{-3}	5×10^{-3}	8×10^{-3}
LR decay	step	step	step	cosine	step	cosine	cosine	cosine
decay rate	0.1	0.1	0.1	-	$0.02^{t/400}$	-	-	-
decay epochs	30	30	30	-	1	-	-	-
Weight decay	10^{-4}	10^{-4}	10^{-4}	0.05	10^{-4}	0.01	0.02	0.02
Warmup epochs	✗	✗	✗	5	5	5	5	5
Label smoothing ε	✗	✗	✗	0.1	0.1	0.1	✗	✗
Dropout	✗	✗	✗	✗	✗	✗	✗	✗
Stoch. Depth	✗	✗	✗	0.1	✗	0.05	0.05	✗
Repeated Aug	✗	✗	✓	✓	✗	✓	✓	✗
Gradient Clip.	✗	✗	✗	✗	✗	✗	✗	✗
H. flip	✓	✓	✓	✓	✓	✓	✓	✓
RRC	✗	✓	✓	✓	✓	✓	✓	✓
Rand Augment	✗	✗	✗	9/0.5	✗	7/0.5	7/0.5	6/0.5
Auto Augment	✗	✗	✗	✗	✓	✗	✗	✗
Mixup alpha	✗	✗	✗	0.8	0.2	0.2	0.1	0.1
Cutmix alpha	✗	✗	✗	1.0	✗	1.0	1.0	1.0
Erasing prob.	✗	✗	✗	0.25	✗	✗	✗	✗
ColorJitter	✗	✓	✓	✗	✗	✗	✗	✗
PCA lighting	✓	✗	✗	✗	✗	✗	✗	✗
SWA	✗	✗	✗	✗	✓	✗	✗	✗
EMA	✗	✗	✗	✗	✗	✗	✗	✗
Test crop ratio	0.875	0.875	0.875	0.875	0.875	0.95	0.95	0.95
CE loss	✓	✓	✓	✓	✓	✗	✗	✗
BCE loss	✗	✗	✗	✗	✗	✓	✓	✓
Mixed precision	✗	✗	✗	✓	✓	✓	✓	✓
Top-1 acc.	75.3%	76.1%	77.0%	78.4%	79.5%	80.4%	79.8%	78.1%

[Wightman et al.] ResNet strikes back : An improved training procedure in timm

MobileNet

MobileNet est un réseau très proche d'Xception en le sens qu'il fait usage des *depthwise separable convolutions*. Son architecture est décrite ci-dessous :

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

[Howard et al.] MobileNets : Efficient Convolutional Neural Networks for Mobile Vision Applications

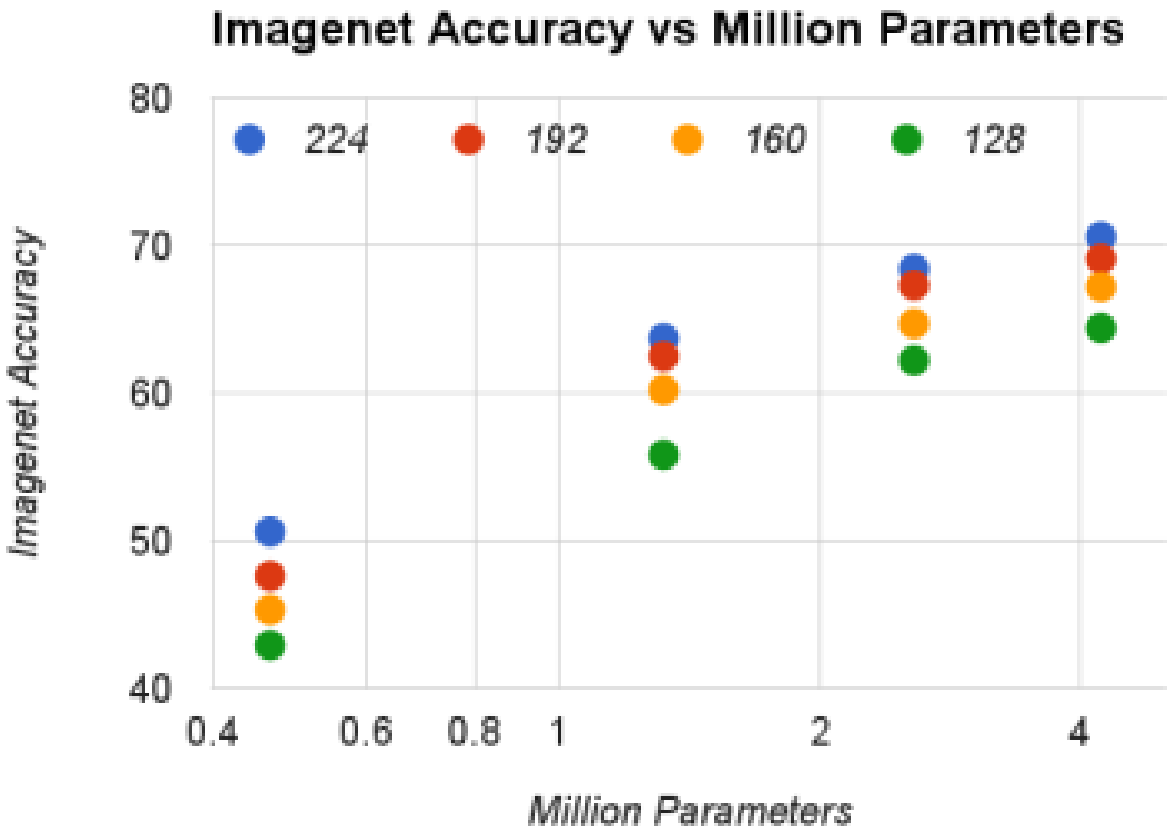
Adaptation topologiques

MobileNet introduit deux hyperparamètres d'adaptation de la topologie du réseau : α, un coefficient multiplicatif appliqué au nombre de filtres de chaque couche, et ρ, un coefficient multiplicatif appliqué à la résolution des images d'entrée.

Layer/Modification	Million Mult-Adds	Million Parameters
Convolution	462	2.36
Depthwise Separable Conv	52.3	0.27
$\alpha = 0.75$	29.6	0.15
$\rho = 0.714$	15.1	0.15

[Howard et al.] MobileNets : Efficient Convolutional Neural Networks for Mobile Vision Applications

MobileNet

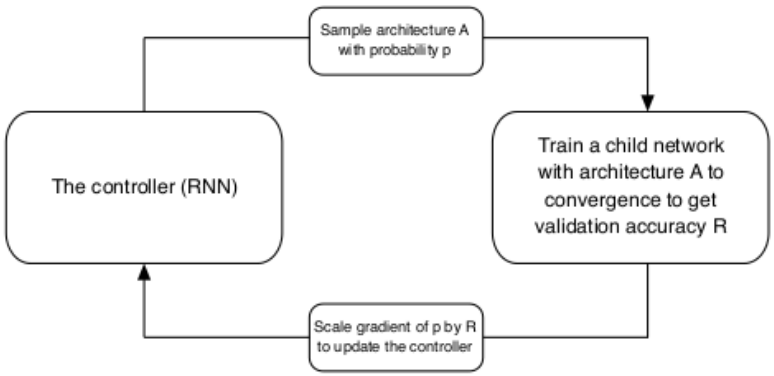


Les performances sont calculées pour 4 valeurs de ρ et d'α.

[Howard et al.] MobileNets : Efficient Convolutional Neural Networks for Mobile Vision Applications

NASNet

NAS signifie *Neural Architecture Search*. On commence par définir un réseau récurrent qui peut générer des séquences d'opérations définissant un bloc de réseau de neurones.



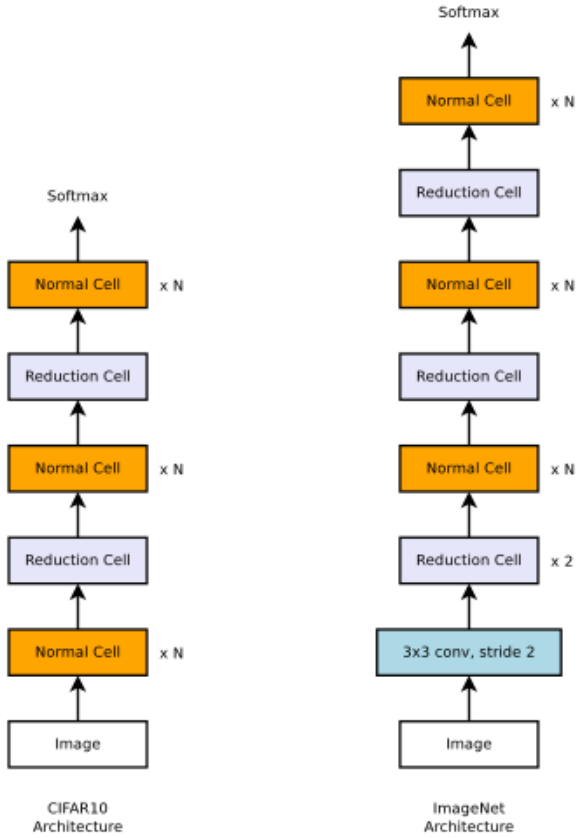
- identity
- 1x7 then 7x1 convolution
- 3x3 average pooling
- 5x5 max pooling
- 1x1 convolution
- 3x3 depthwise-separable conv
- 7x7 depthwise-separable conv
- 1x3 then 3x1 convolution
- 3x3 dilated convolution
- 3x3 max pooling
- 7x7 max pooling
- 3x3 convolution
- 5x5 depthwise-seperable conv

Le réseau récurrent est entraîné par renforcement.

[Zoph et al.] Learning Transferable Architectures for Scalable Image Recognition

NASNet

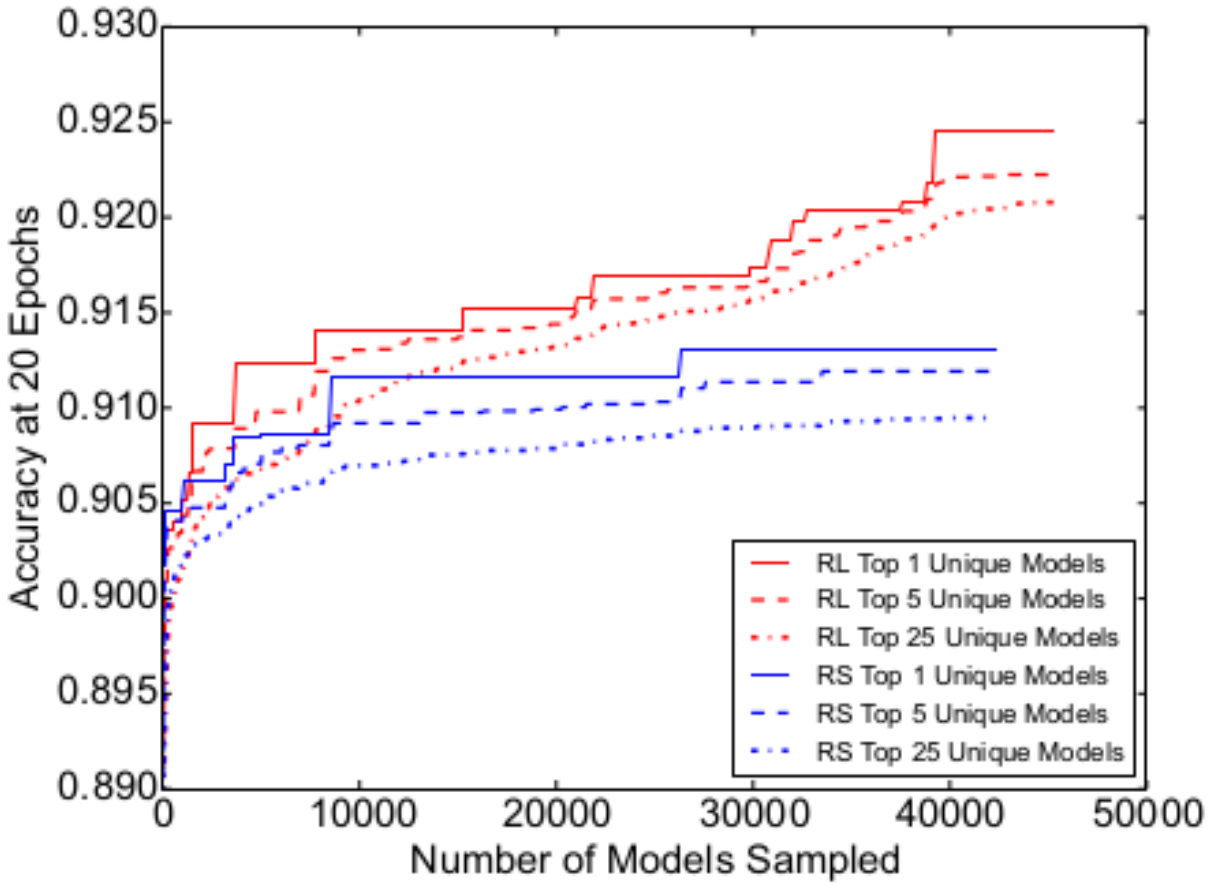
Au préalable, on a pré-défini le besoin de deux types de blocs et l'enchaînement de ces blocs.



[Zoph et al.] Learning Transferable Architectures for Scalable Image Recognition

NASNet

Plus de 50000 réseaux ont été entraînés jusqu'à convergence sur CIFAR-10 pour obtenir des topologies optimales pour les blocs :

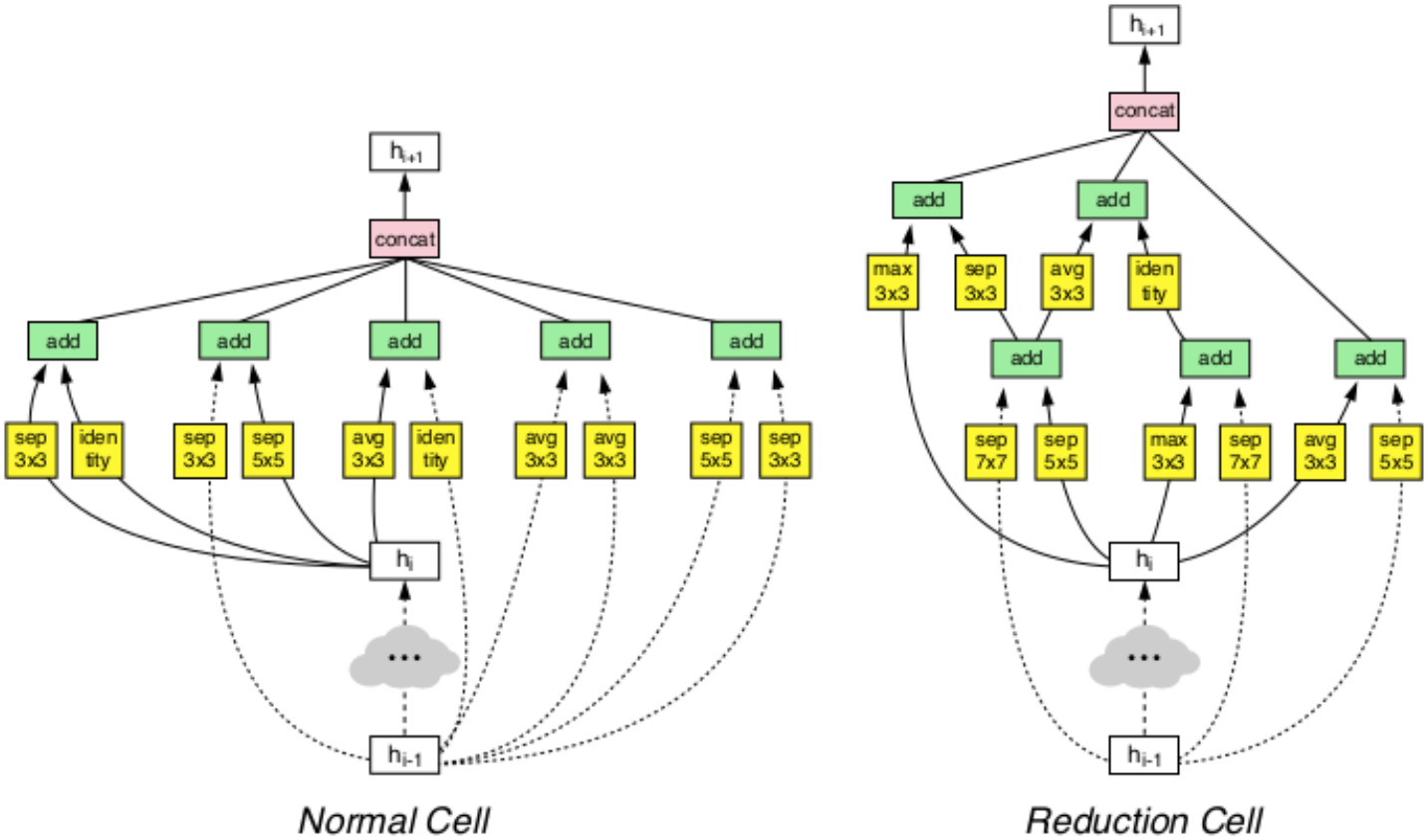


4 jours d'entraînement sur 500 GPUs...

[Zoph et al.] Learning Transferable Architectures for Scalable Image Recognition

NASNet

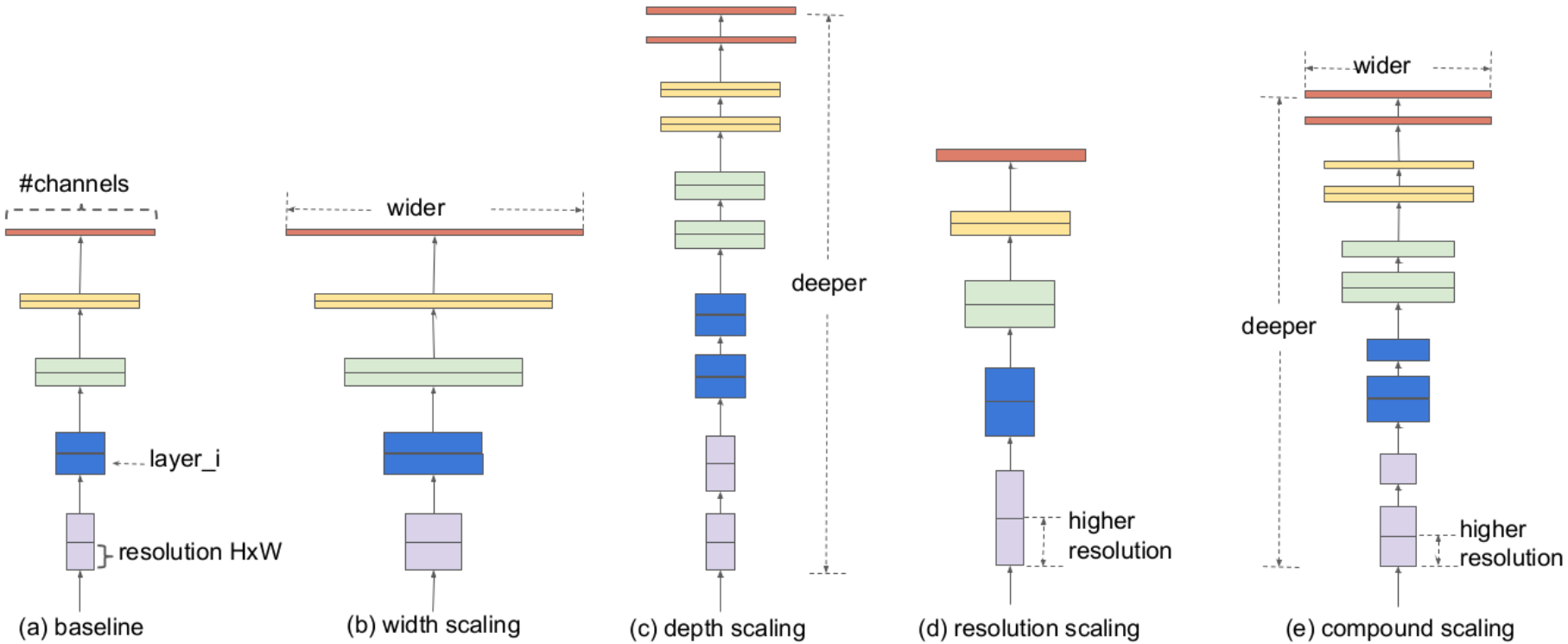
Topologies obtenues pour le bloc normal (gauche) et de réduction (droite) :



[Zoph et al.] Learning Transferable Architectures for Scalable Image Recognition

EfficientNet

Idée de départ : mise à l'échelle d'un réseau sur plusieurs dimensions simultanées



[Tan et al.] EfficientNet : Rethinking Model Scaling for Convolutional Neural Networks

EfficientNet : méthodologie

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

Le nombre d'opérations d'un réseau dépend linéairement de la profondeur, et quadratiquement du nombre de filtres et de la résolution de l'image d'entrée.
Le problème est formulé pour faire en sorte qu'à chaque nouvelle valeur de ϕ,le nombre d'opérations du réseau soit doublé.
Les auteurs ont établi les valeurs $\alpha = 1.2$, $\beta = 1.1$ et $\gamma = 1.15$ comme optimales.

[Tan et al.] EfficientNet : Rethinking Model Scaling for Convolutional Neural Networks

EfficientNet : architecture de base

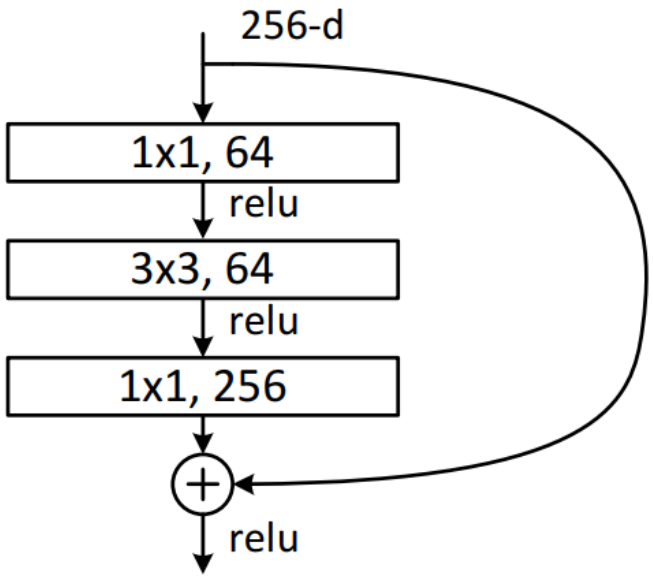
Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14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

Cett architecture a été établie par renforcement, comme pour NASNet (mêmes auteurs). Le bloc MBConv est une variante du bloc *bottleneck* de ResNet.

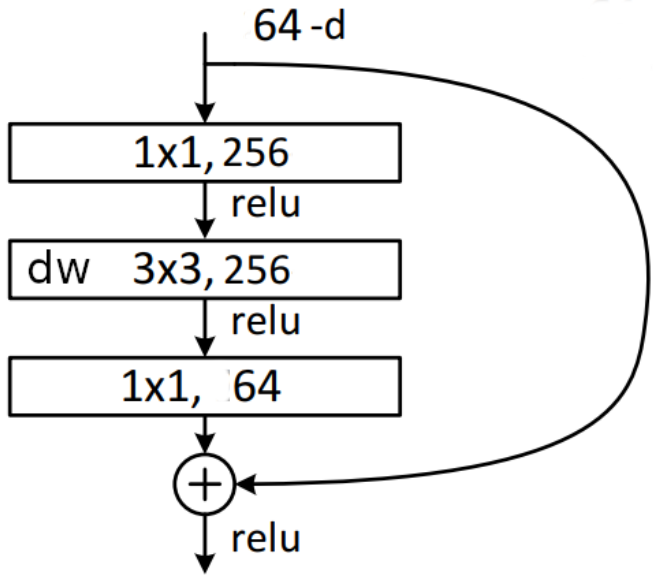
[Tan et al.] EfficientNet : Rethinking Model Scaling for Convolutional Neural Networks

EfficientNet

Brique de base : bloc *inverted bottleneck* avec le module *Squeeze-and-Excitation*.



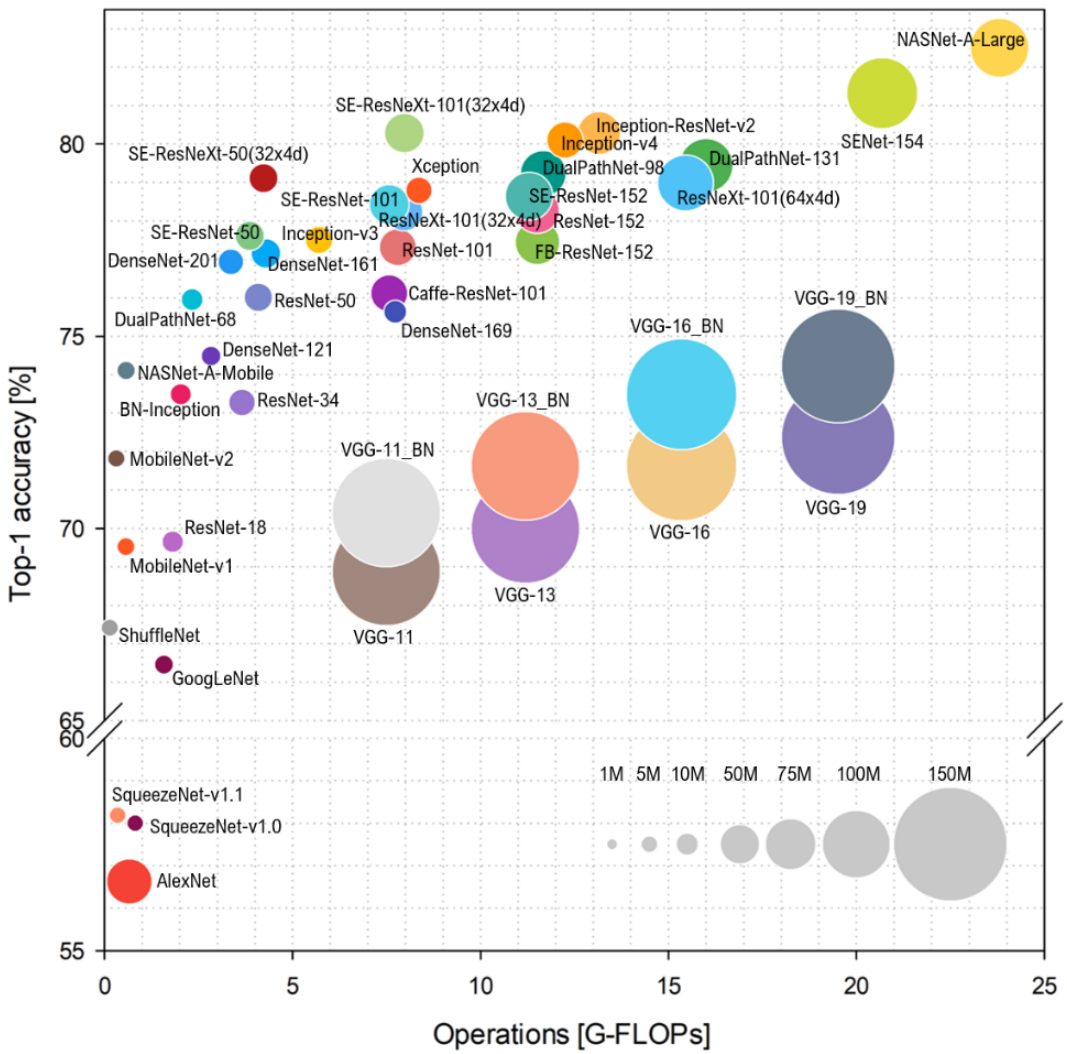
Bloc *bottleneck* de ResNet



Bloc *inverted bottleneck*

[Sandler et al.] Mobilenetv2 : Inverted residuals and linear bottlenecks

Comparaison



[Bianco et al.] Benchmark Analysis of Representative Deep Neural Network Architectures

Que retenir ?

- Après ResNet qui a marqué un tournant dans les performances et la capacité d'entraînement des réseaux, l'essentiel des travaux s'est attaché à optimiser le rapport performance/nombre de paramètres, et performance/nombre d'opérations .
- La méthodologie couramment adoptée pour résoudre un problème consiste à choisir un réseau qui fournit les performances attendues, puis à le mettre à l'échelle pour s'adapter aux contraintes de ressources, ou augmenter la performance.
- 11 réseaux présentés aujourd'hui : 6 Google, 2 Facebook, 1 Microsoft...

Il est inutile d'essayer d'élaborer votre propre architecture !