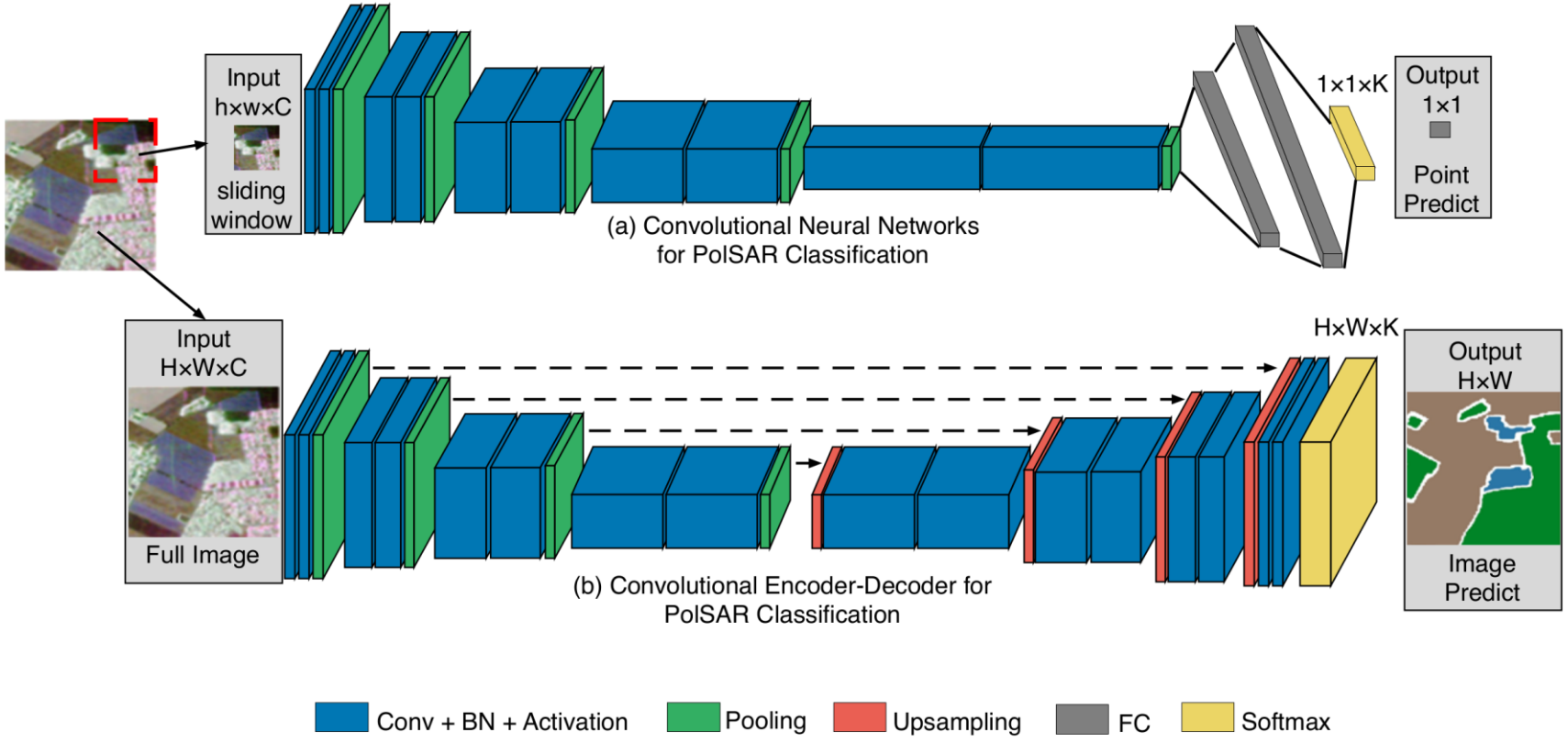
UNIVERSITE ABDELMALEK ESSAADI
FACULTE DES SCIENCES
TETOUAN

جامعة عبد المالك السعدي
كلية العلوم
تطوان

Apprentissage profond pour la vision par ordinateur

Presented by Pr. Abdellatif El Ouissari

Architectures



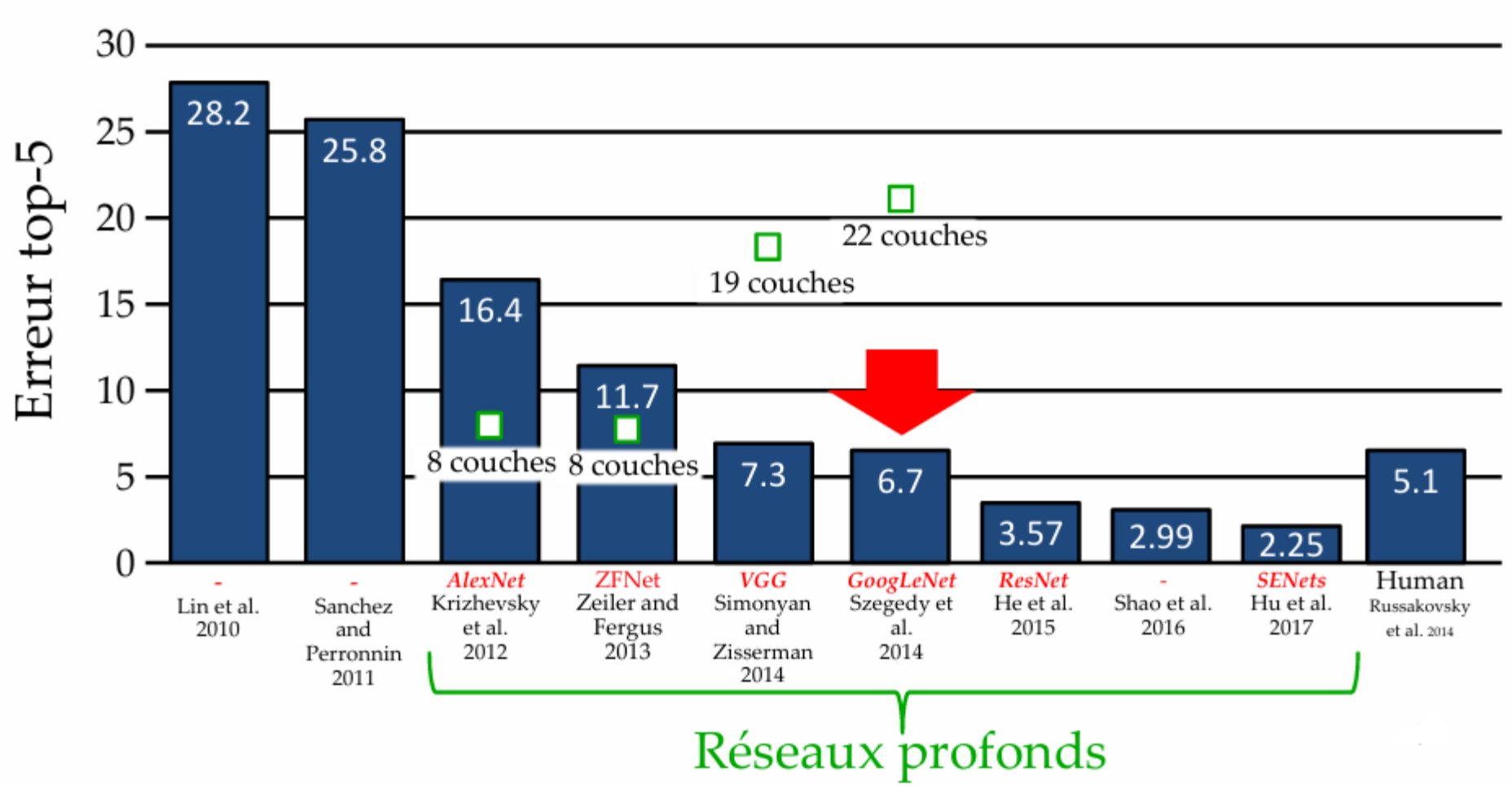
Introduction

- Décrire et comparer les architectures célèbres : **GoogLeNet (Inception)**, **ResNet**
- Identifier les **avantages et limites** de chaque architecture
- Analyser les performances selon les métriques comme **Top-1 / Top-5 accuracy**
- Comprendre l'évolution vers des réseaux **plus profonds**, **plus rapides** et **plus efficaces**

CNN

- 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

Large Scale Visual Recognition Challenge



GoogLeNet

Une architecture de réseau neuronal convolutif profond appelée Inception a été proposé, qui a permis d'établir un nouvel état de l'art en matière de classification et de détection lors du concours ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14).

La principale caractéristique de cette architecture est l'amélioration de l'utilisation des ressources informatiques au sein du réseau. Ce résultat a été obtenu grâce à une conception soigneusement élaborée qui permet d'augmenter la profondeur et la largeur du réseau tout en maintenant le budget de calcul constant.

GoogLeNet

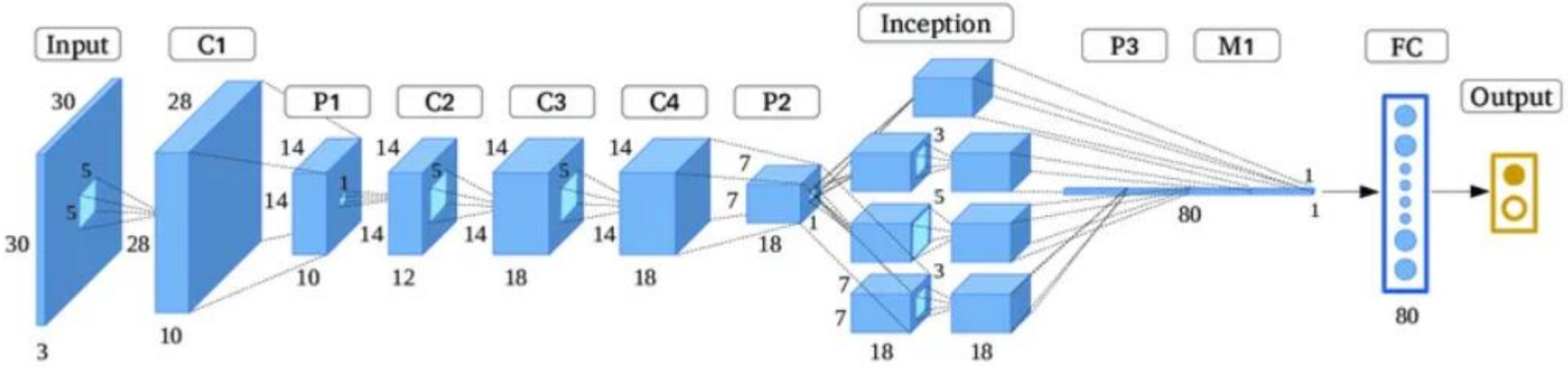
Pour optimiser la qualité, les décisions architecturales ont été basées sur le principe de "Hebbian learning" et l'intuition du traitement multi-échelle.

Une incarnation particulière utilisée est appelée GoogLeNet, un réseau profond de 22 couches, dont la qualité est évaluée dans le contexte de la classification et de la détection.

GoogLeNet

GoogLeNet, également connu sous le nom d'Inception-v1, est une réalisation historique dans le domaine des réseaux neuronaux convolutifs (CNN).

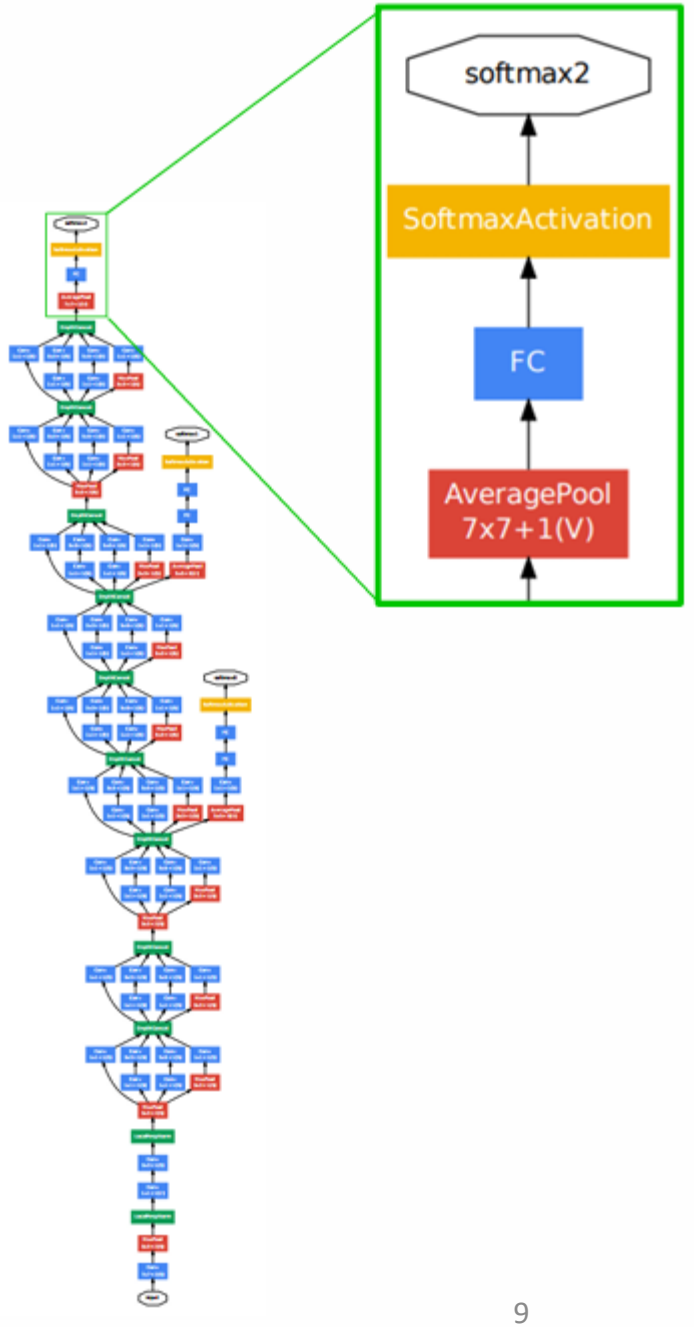
Dévoilé par les chercheurs de Google en 2014, il a introduit une nouvelle approche de la conception des réseaux profonds qui a non seulement permis d'atteindre une précision exceptionnelle, mais a également fait preuve d'une efficacité de calcul remarquable.



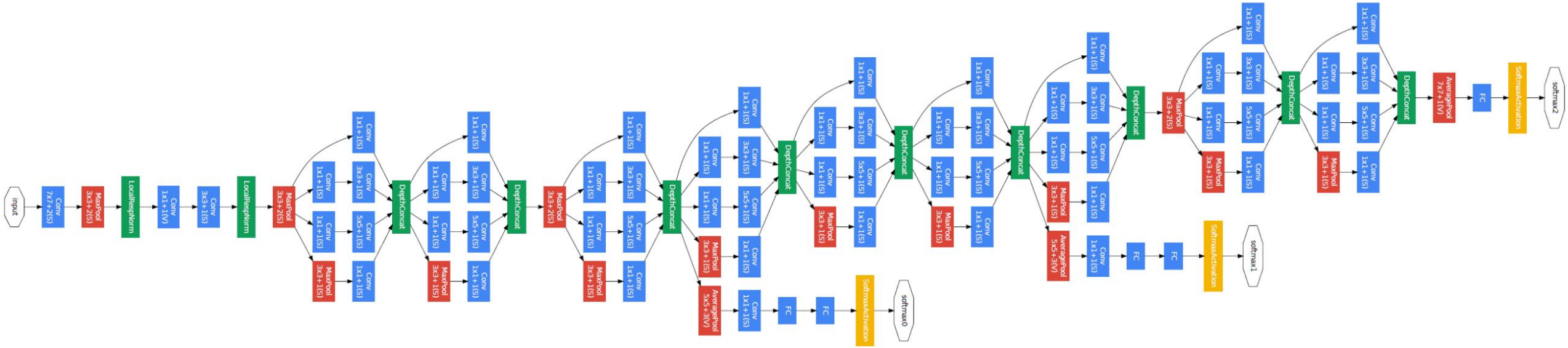
GoogLeNet (Inception v1) architecture

GoogLeNet

- Réseau plus profond (22 couches)
- Seulement 5 millions de paramètres, 12 fois moins qu'AlexNet
- Toujours pas de batch norm
- GAP + une couche fully-connected (tendance classificateur faible)
- La couche fully connected ne sert (au dire des auteur) qu'à adapter les features finaux vers le nombre de sorties (labels) désirées.

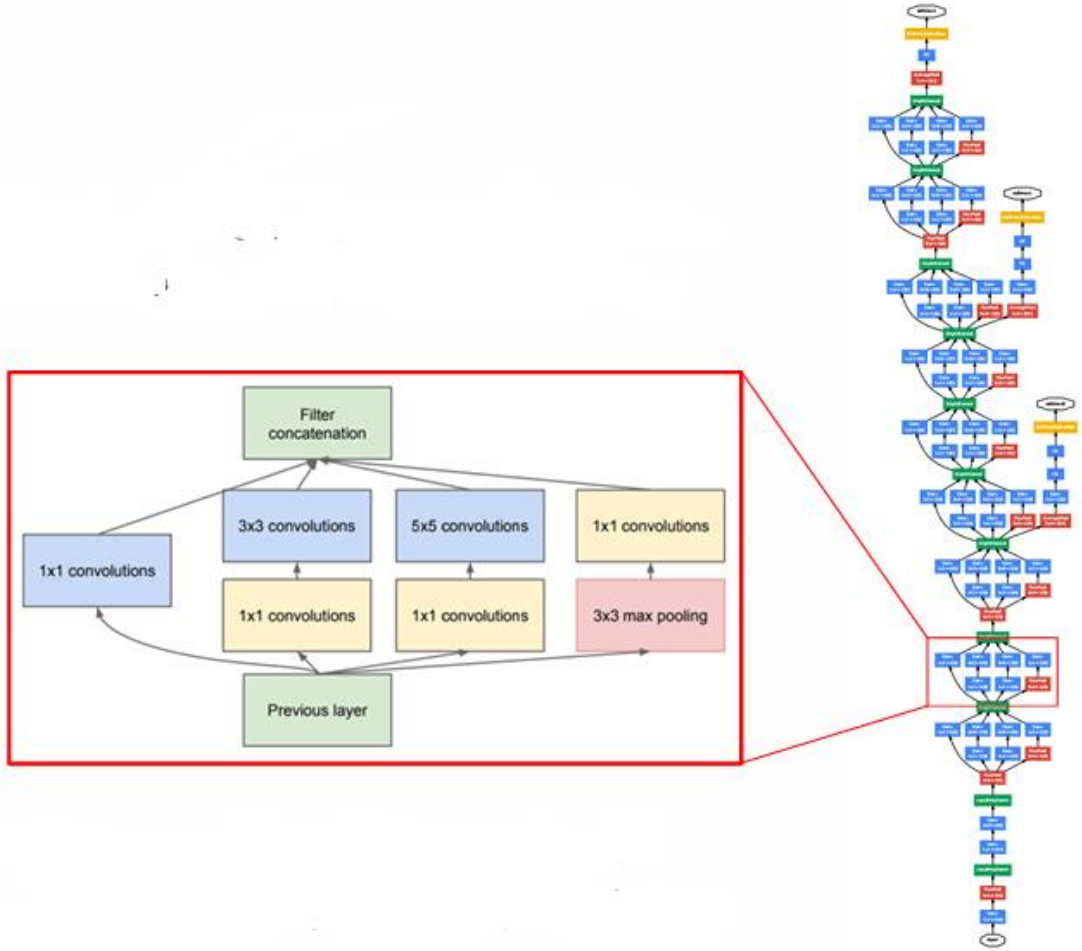


GoogLeNet


input
Conv 7x7+2(S)
MaxPool 3x3+2(S)
LocalRespNorm
Conv 1x1+1(V)
Conv 3x3+1(S)
LocalRespNorm
MaxPool 3x3+2(S)
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
MaxPool 3x3+1(S)
Conv 1x1+1(S)
DepthConcat
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
MaxPool 3x3+1(S)
Conv 1x1+1(S)
DepthConcat
MaxPool 3x3+2(S)
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
MaxPool 3x3+1(S)
Conv 1x1+1(S)
DepthConcat
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
MaxPool 3x3+1(S)
Conv 1x1+1(S)
DepthConcat
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
AveragePool 5x5+3(V)
Conv 1x1+1(S)
FC
FC
SoftmaxActivation
softmax0
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
MaxPool 3x3+1(S)
Conv 1x1+1(S)
DepthConcat
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
AveragePool 5x5+3(V)
Conv 1x1+1(S)
FC
FC
SoftmaxActivation
softmax1
MaxPool 3x3+2(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 1x1+1(S)
Conv 5x5+1(S)
MaxPool 3x3+1(S)
Conv 1x1+1(S)
DepthConcat
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
MaxPool 3x3+1(S)
Conv 1x1+1(S)
DepthConcat
AveragePool 7x7+1(V)
FC
SoftmaxActivation
softmax2

GoogLeNet

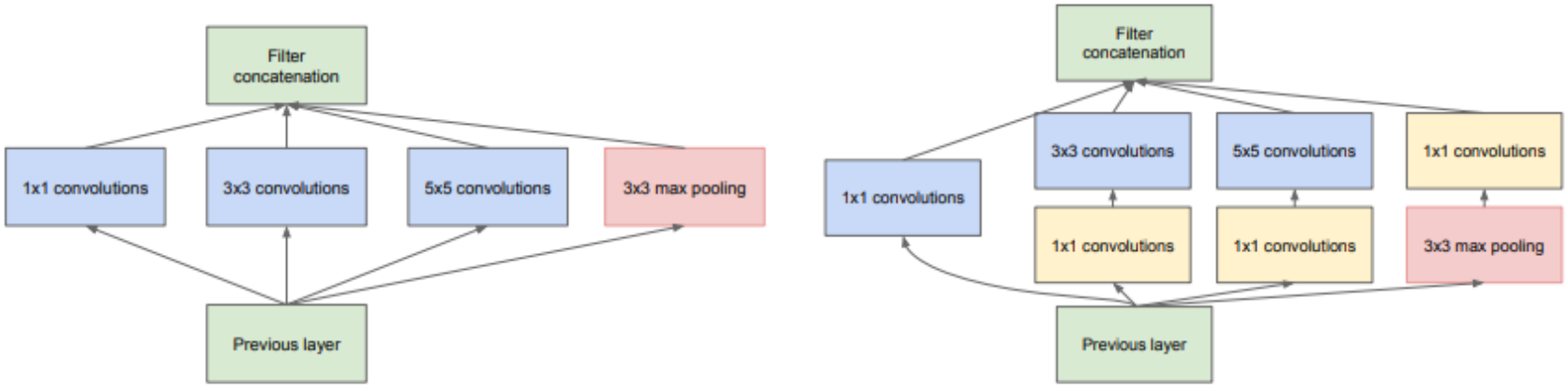
Emphase sur minimisation des calculs via module Inception



- S’éloigne ainsi de l’approche convolution avec taille de filtre unique suivie de maxpool

GoogLeNet

Le module Inception combine plusieurs convolutions (1x1, 3x3, 5x5) et une mise en commun parallèle pour capturer des informations à différentes échelles, tout en optimisant les coûts de calcul.



(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 2: Inception module

GoogLeNet

Network-in-Network est une approche proposée par Lin afin d'augmenter le pouvoir de représentation des réseaux neuronaux.

Appliquée aux couches convolutives, la méthode peut être considérée comme des couches convolutives 1×1 supplémentaires suivies typiquement par l'activation linéaire rectifiée.

Cela permet de l'intégrer facilement dans les pipelines CNN actuels.

GoogLeNet

Nous utilisons largement cette approche dans notre architecture. Cependant, dans notre cadre, les convolutions 1 × 1 ont un double objectif :

le plus important est qu'elles sont utilisées principalement comme modules de réduction de la dimension pour éliminer les goulets d'étranglement informatiques qui, autrement, limiteraient la taille de nos réseaux.

Cela permet d'augmenter non seulement la profondeur, mais aussi la largeur de nos réseaux sans pénalité significative en termes de performances.

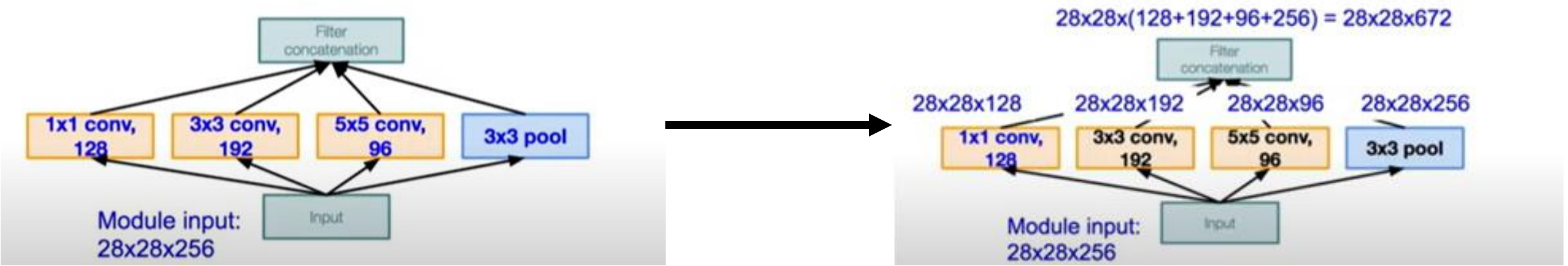
En général, un réseau d'Inception est un réseau composé de modules du type ci-dessus empilés les uns sur les autres, avec des couches occasionnelles de max-pooling avec stride 2 pour diviser par deux la résolution de la grille.

Pour des raisons techniques (efficacité de la mémoire pendant l'apprentissage), il a semblé avantageux de commencer à utiliser les modules d'intériorisation uniquement dans les couches supérieures, tout en conservant les couches inférieures en mode convolutionnel traditionnel.

Cela n'est pas strictement nécessaire, mais reflète simplement certaines inefficacités infrastructurelles dans notre implémentation actuelle.

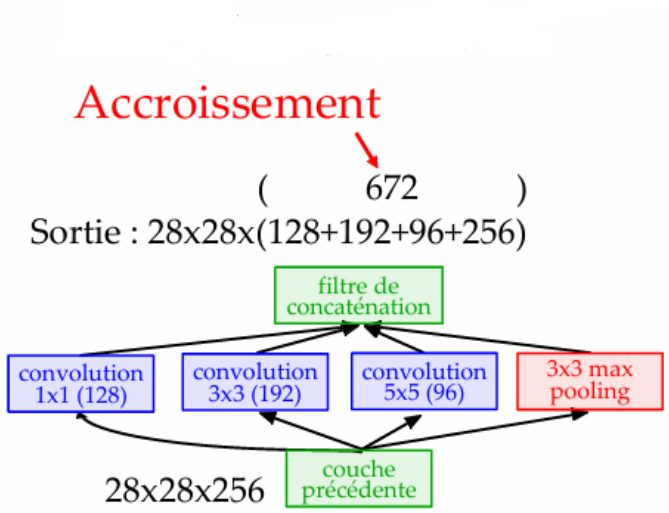
GoogLeNet

Quelle est la taille de la sortie après concaténation des filtres ?



GoogLeNet

• Idée de base : avoir des filtres en parallèle avec des champs récepteurs de taille multiple (pyramide spatiale)
• La couche suivante a donc accès à des features à plusieurs échelles spatiales
• Version naïve :



Coût en calcul

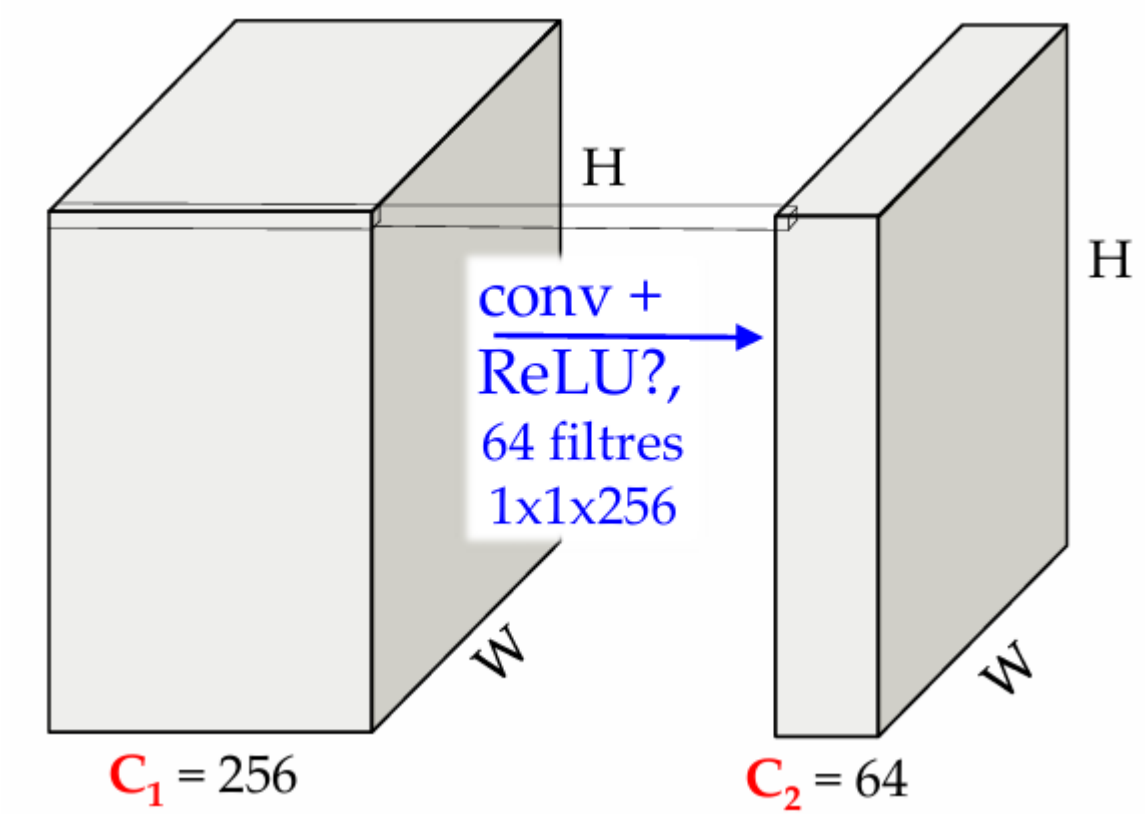
Conv Ops:
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x256
[5x5 conv, 96] 28x28x96x5x5x256
Total: 854M ops

Détail des calculs dans la vidéo Stanford
https://youtu.be/DAOcjicFr1Y?t=1717

Convolution 1x1

- Origine dans NiN
- Popularisées par GoogLeNet (prochaine architecture)
- Semble inutile…
- Rappel : 1x1xC
- Préserve les dimensions H, W

- Sert à réduire le nombre de dimensions C, via une projection linéaire (style PCA)
- Équivalent à Fully connected sur features
- Forme de bottleneck



Convolution 1×1

- Une convolution 1×1 prend un pixel et applique une transformation linéaire sur les canaux.

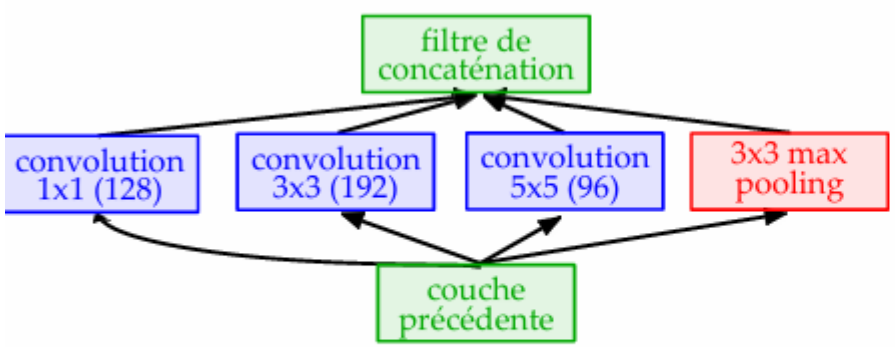
- Elle agit comme une couche dense appliquée spatialement.

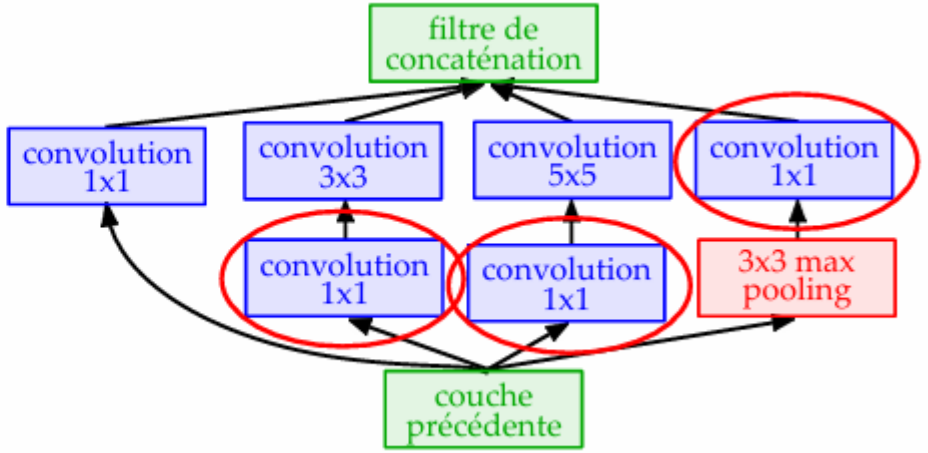
- Elle permet donc de réduire le nombre de canaux tout en conservant la taille spatiale (hauteur × largeur).

- *Par exemple : transformer un tenseur de taille (H×W×256) en (H×W×64).*

GoogLeNet

- Ajout de convolutions 1x1 comme bottleneck
- Permet de choisir la dimension d'entrée des opérations de convolution couteuse



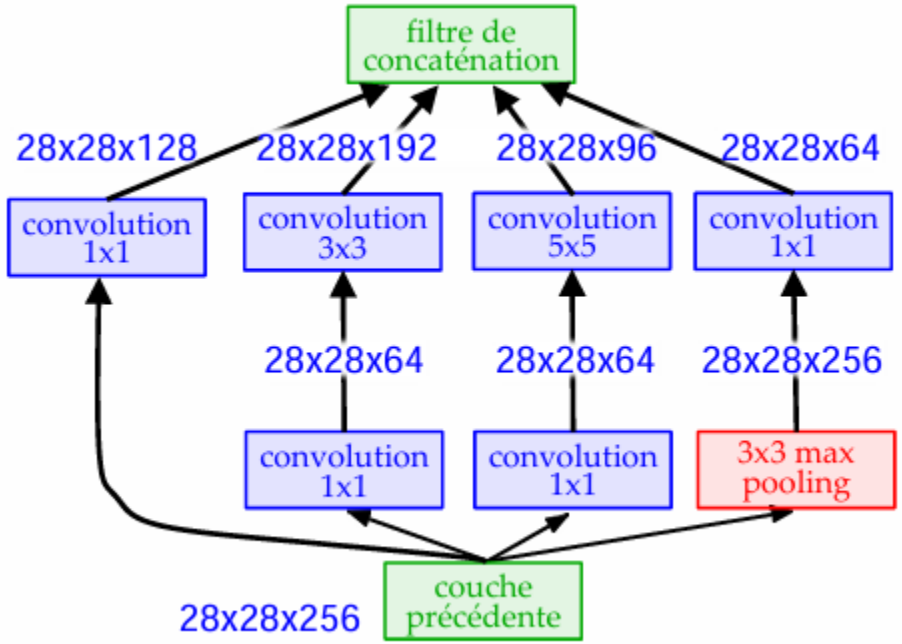


GoogLeNet

- Fera diminuer :

– complexité de calcul

– dimension en sortie



Coût en calcul

Conv Ops:

[1x1 conv, 64] 28x28x64x1x1x256

[1x1 conv, 64] 28x28x64x1x1x256

[1x1 conv, 128] 28x28x128x1x1x256

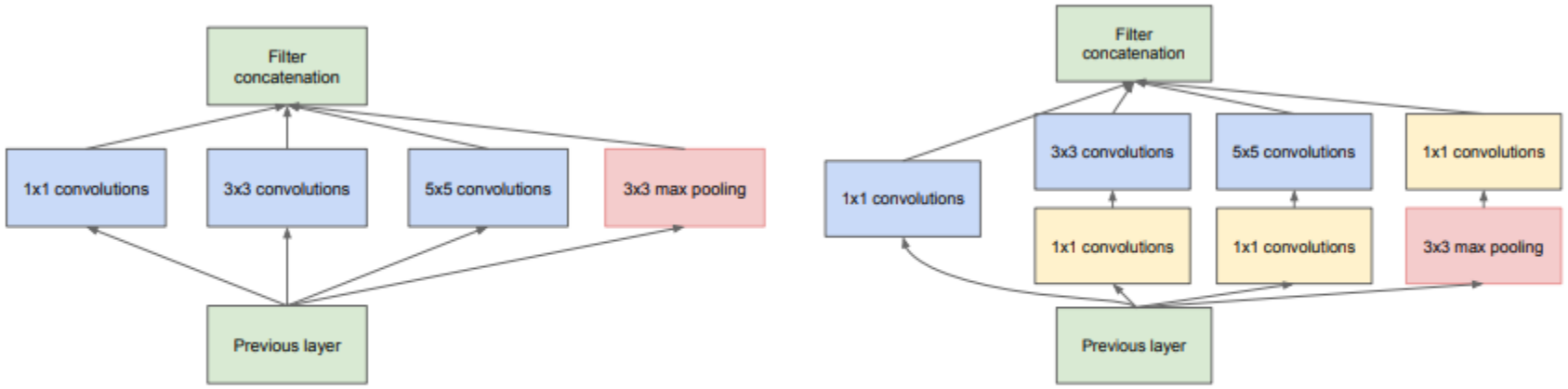
[3x3 conv, 192] 28x28x192x3x3x64

[5x5 conv, 96] 28x28x96x5x5x64

[1x1 conv, 64] 28x28x64x1x1x256

Total: 358M ops

Passe de 854Mops à 358 Mops pour cet exemple



(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 2: Inception module

Nous aimerions que notre représentation reste peu dense à la plupart des endroits et que les signaux ne soient compressés que lorsqu'ils doivent être agrégés en masse.
En d'autres termes, les convolutions 1×1 sont utilisées pour calculer les réductions avant les convolutions 3×3 et 5×5, qui sont coûteuses.

En plus d'être utilisées comme réductions, elles comprennent également l'utilisation de l'activation linéaire rectifiée, ce qui leur confère une double utilité.

GoogLeNet

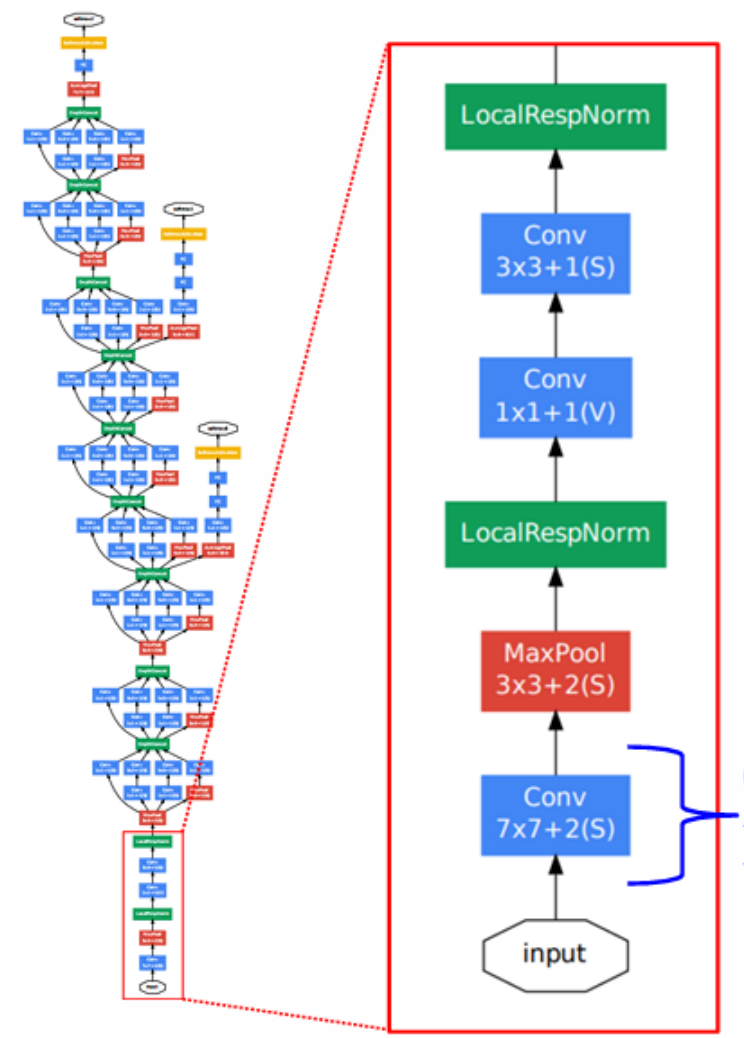
Entrée

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Augmente le nombre de filtres selon la distance de l'entrée

Sortie

GoogLeNet : base du réseau



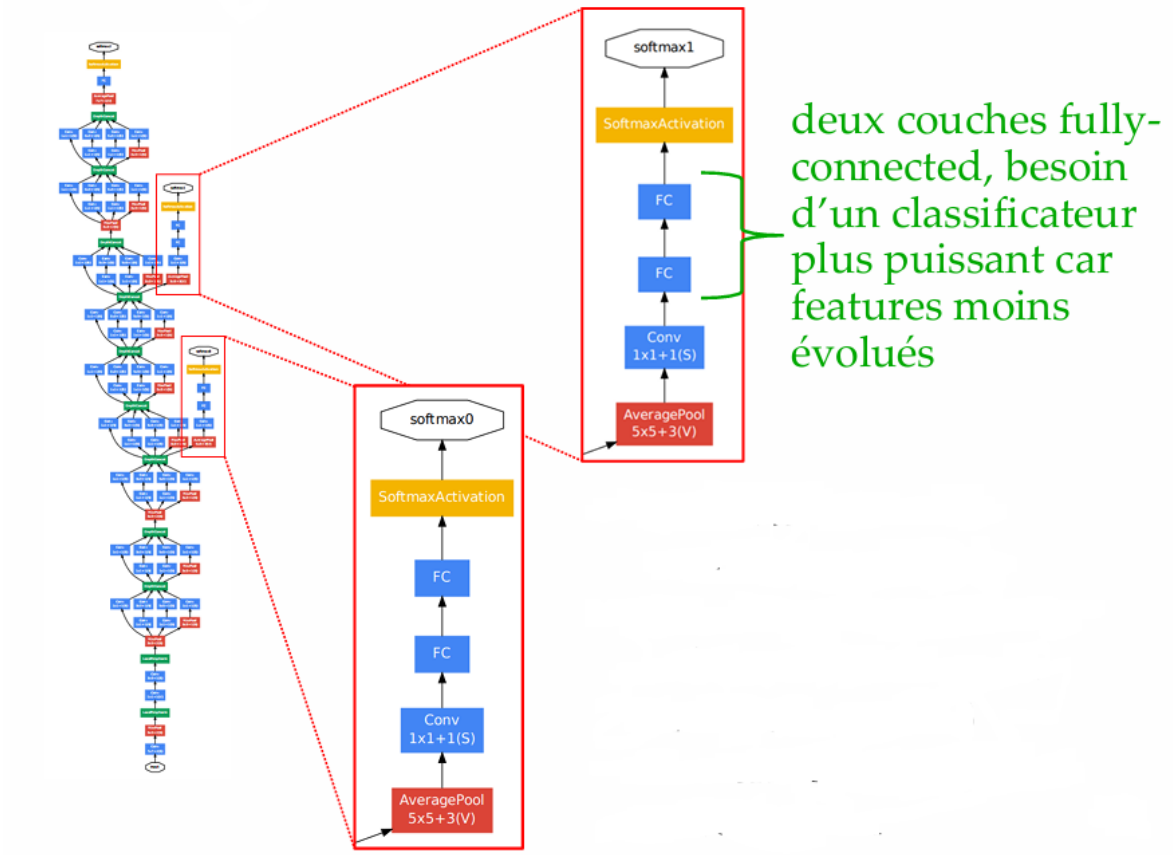
Convolution plus large à la base
Par compar. AlexNet : 11x11

GoogLeNet

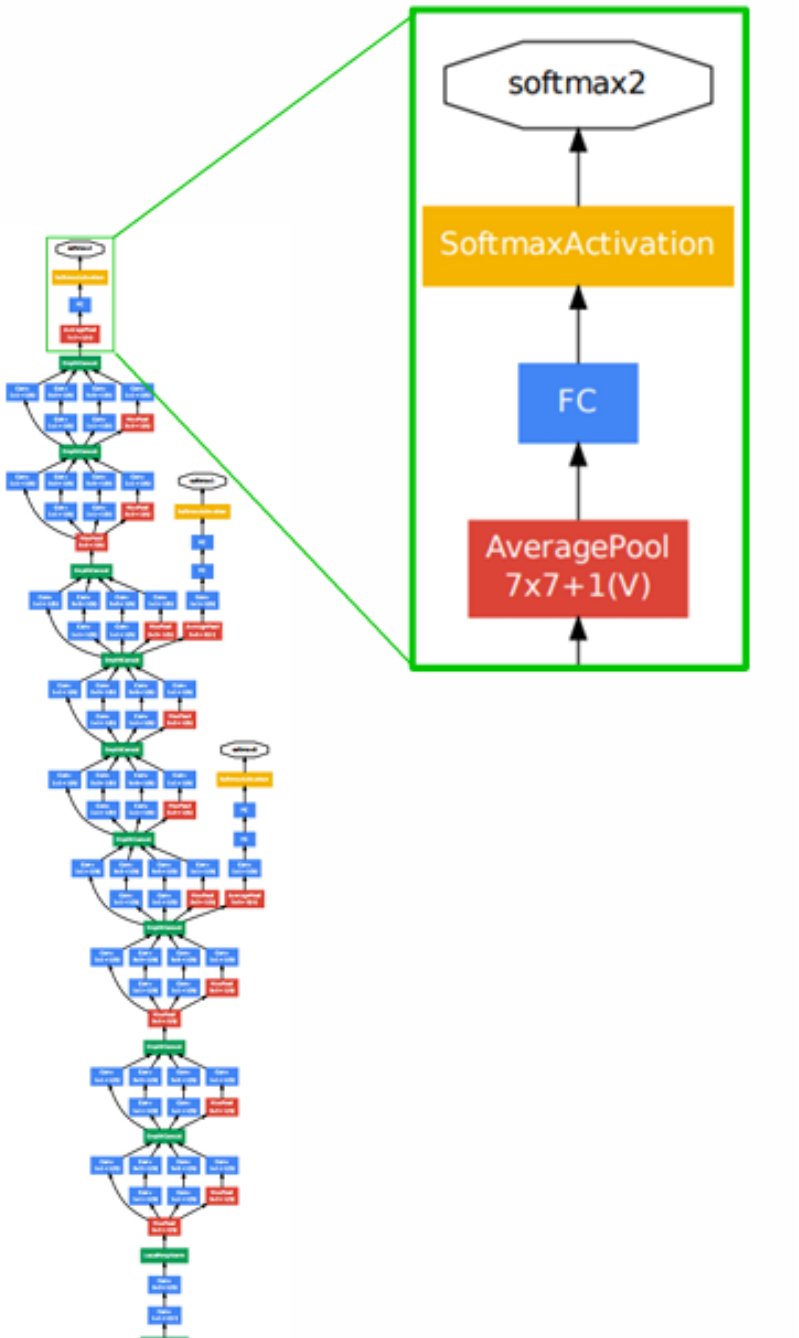
Lors de l’entraînement de réseaux très profonds, le gradient peut se dissiper en remontant vers les premières couches.

Ce phénomène est connu sous le nom de vanishing gradient.

GoogLeNet : têtes auxiliaires



- Pour combattre le vanishing gradient
- Fonction des pertes sur softmax0 et softmax1 multipliée par 0.3
- Limites : Le gain en performance final est minime (~0.5%), mais l'effet sur la convergence est important durant l'entraînement.

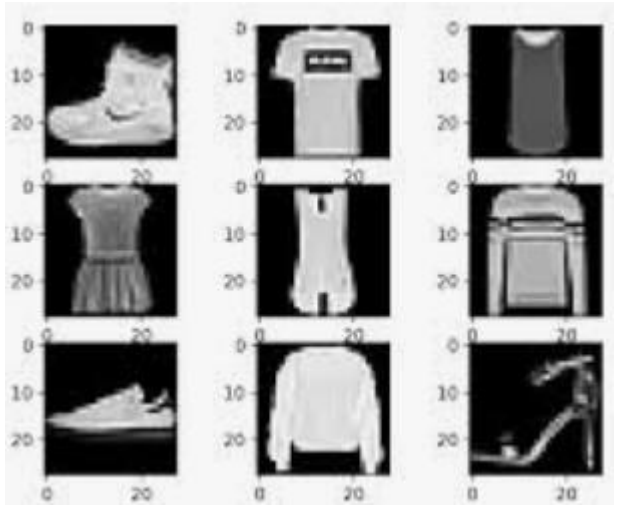


Le classificateur de sortie

Fashion-MNIST Dataset

Fashion-MNIST est un ensemble de données d'images d'articles de Zalando, composé d'un ensemble d'apprentissage de 60 000 exemples et d'un ensemble de test de 10 000 exemples. Chaque exemple est une image en niveaux de gris de 28x28, associée à une étiquette parmi 10 classes.

Zalando souhaite que Fashion-MNIST remplace directement l'ensemble de données MNIST original pour l'évaluation comparative des algorithmes d'apprentissage automatique. Il partage la même taille d'image et la même structure de divisions d'entraînement et de test.



Fashion-MNIST Dataset

The classes are:

Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

Fashion-MNIST Dataset

```
(X_train, y_train), (X_test, y_test) = tf.keras.datasets.fashion_mnist.load_data()
X_val, y_val = X_train[:5000], y_train[:5000]
X_train, y_train = X_train[5000:], y_train[5000:]
```

Fashion-MNIST Dataset

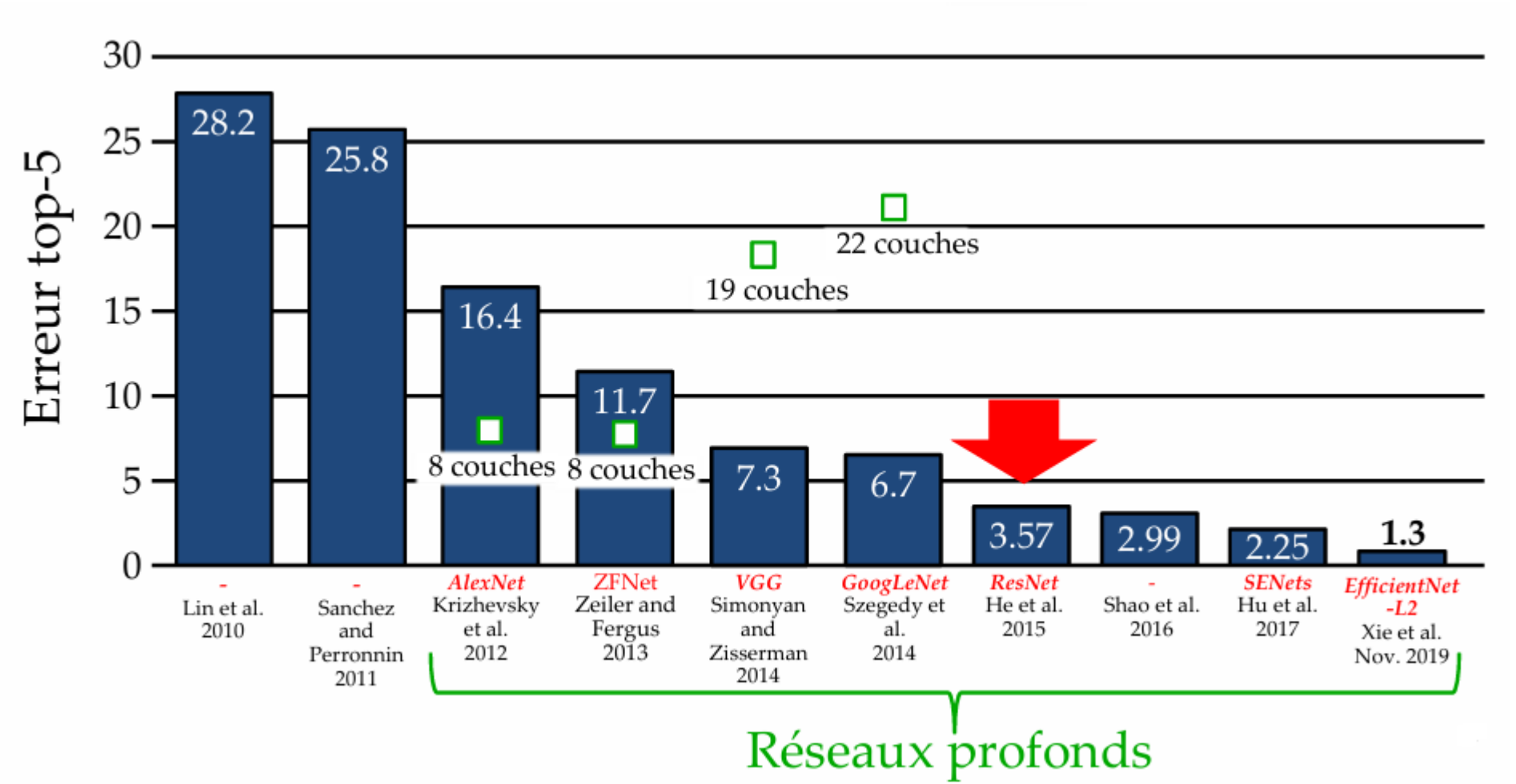
Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 28, 28, 1)	0	-
conv2d (Conv2D)	(None, 14, 14, 64)	3,200	input_layer[0][0]
batch_normalization (BatchNormalizatio…	(None, 14, 14, 64)	256	conv2d[0][0]
activation (Activation)	(None, 14, 14, 64)	0	batch_normalizat…
max_pooling2d (MaxPooling2D)	(None, 7, 7, 64)	0	activation[0][0]
conv2d_1 (Conv2D)	(None, 7, 7, 64)	4,160	max_pooling2d[0]…
conv2d_2 (Conv2D)	(None, 7, 7, 192)	110,784	conv2d_1[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 7, 7, 192)	768	conv2d_2[0][0]
activation_1 (Activation)	(None, 7, 7, 192)	0	batch_normalizat…
max_pooling2d_1 (MaxPooling2D)	(None, 4, 4, 192)	0	activation_1[0][…
max_pooling2d_12 (MaxPooling2D)	(None, 1, 1, 832)	0	concatenate_7[0]…
conv2d_51 (Conv2D)	(None, 1, 1, 384)	319,872	concatenate_7[0]…
conv2d_53 (Conv2D)	(None, 1, 1, 384)	663,936	conv2d_52[0][0]
conv2d_55 (Conv2D)	(None, 1, 1, 128)	153,728	conv2d_54[0][0]
conv2d_56 (Conv2D)	(None, 1, 1, 128)	106,624	max_pooling2d_12…
concatenate_8 (Concatenate)	(None, 1, 1, 1024)	0	conv2d_51[0][0], conv2d_53[0][0], conv2d_55[0][0], conv2d_56[0][0]
global_average_poo… (GlobalAveragePool…	(None, 1024)	0	concatenate_8[0]…
dropout (Dropout)	(None, 1024)	0	global_average_p…
dense (Dense)	(None, 10)	10,250	dropout[0][0]

Total params: 5,978,554 (22.81 MB)
Trainable params: 5,978,042 (22.80 MB)
Non-trainable params: 512 (2.00 KB)

Fashion-MNIST Dataset

```
history = GoogLeNet.fit(X_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1, validation_data=(X_val, y_val),
                        callbacks=my_callbacks)
```

```
Epoch 1/5
430/430 ━━━━━━━━━━━━━━━━━━━━ 0s 1s/step - accuracy: 0.6256 - loss: 0.9620WARNING:absl:You are saving your model as an HDF5 fi
430/430 ━━━━━━━━━━━━━━━━━━━━ 509s 1s/step - accuracy: 0.6259 - loss: 0.9613 - val_accuracy: 0.8360 - val_loss: 0.4964
Epoch 2/5
430/430 ━━━━━━━━━━━━━━━━━━━━ 0s 1s/step - accuracy: 0.8461 - loss: 0.4320WARNING:absl:You are saving your model as an HDF5 fi
430/430 ━━━━━━━━━━━━━━━━━━━━ 494s 1s/step - accuracy: 0.8461 - loss: 0.4319 - val_accuracy: 0.8648 - val_loss: 0.3998
Epoch 3/5
430/430 ━━━━━━━━━━━━━━━━━━━━ 0s 1s/step - accuracy: 0.8828 - loss: 0.3337WARNING:absl:You are saving your model as an HDF5 fi
430/430 ━━━━━━━━━━━━━━━━━━━━ 504s 1s/step - accuracy: 0.8828 - loss: 0.3337 - val_accuracy: 0.8758 - val_loss: 0.3689
Epoch 4/5
430/430 ━━━━━━━━━━━━━━━━━━━━ 497s 1s/step - accuracy: 0.8999 - loss: 0.2873 - val_accuracy: 0.8774 - val_loss: 0.3874
Epoch 5/5
430/430 ━━━━━━━━━━━━━━━━━━━━ 506s 1s/step - accuracy: 0.9031 - loss: 0.2764 - val_accuracy: 0.8456 - val_loss: 0.4685
```


Erreur top-5
30
25
20
15
10
5
0
28.2
25.8
16.4
11.7
7.3
6.7
3.57
2.99
2.25
1.3
8 couches
8 couches
19 couches
22 couches
-
Lin et al.
2010
-
Sanchez
and
Perronnin
2011
AlexNet
Krizhevsky
et al.
2012
ZFNet
Zeiler and
Fergus
2013
VGG
Simonyan
and
Zisserman
2014
GoogLeNet
Szegedy et
al.
2014
ResNet
He et al.
2015
-
Shao et al.
2016
SENets
Hu et al.
2017
EfficientNet
-L2
Xie et al.
Nov. 2019
Réseaux profonds

ResNet : Introduction

Les réseaux neuronaux plus profonds sont plus difficiles à former.

Nous présentons un cadre d'apprentissage résiduel pour faciliter la formation de réseaux beaucoup plus profonds que ceux utilisés précédemment.

Les créateurs reformulent explicitement les couches comme apprenant des fonctions résiduelles par rapport à leurs entrées, plutôt que d'apprendre des fonctions absolues non référencées.

Nous fournissons des preuves empiriques complètes montrant que ces réseaux résiduels sont plus faciles à optimiser et qu'ils peuvent gagner en précision grâce à une profondeur considérablement accrue.

ResNet : Introduction

- Sur l'ensemble de données ImageNet, nous évaluons les réseaux résiduels avec une profondeur allant jusqu'à 152 couches, soit 8 fois plus que les réseaux VGG, tout en ayant une complexité moindre.

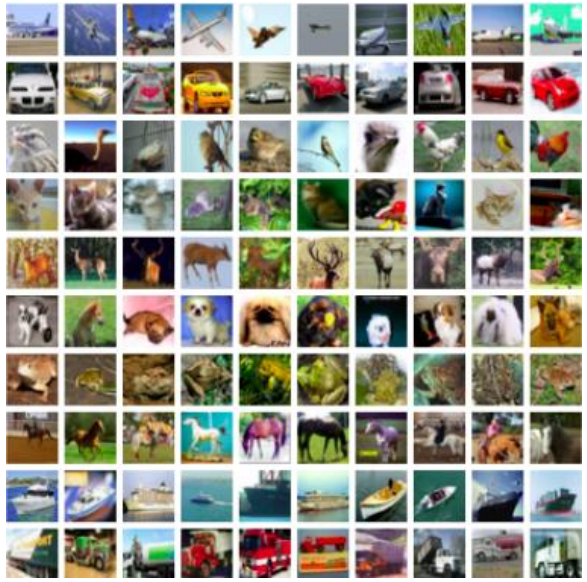
- Un ensemble de ces réseaux résiduels obtient une erreur de 3,57 % sur l'ensemble de données ImageNet.

- Ce résultat a remporté la première place à la tâche de classification ILSVRC 2015. Nous présentons également une analyse sur CIFAR-10 avec 100 et 1000 couches.

ResNet : Introduction

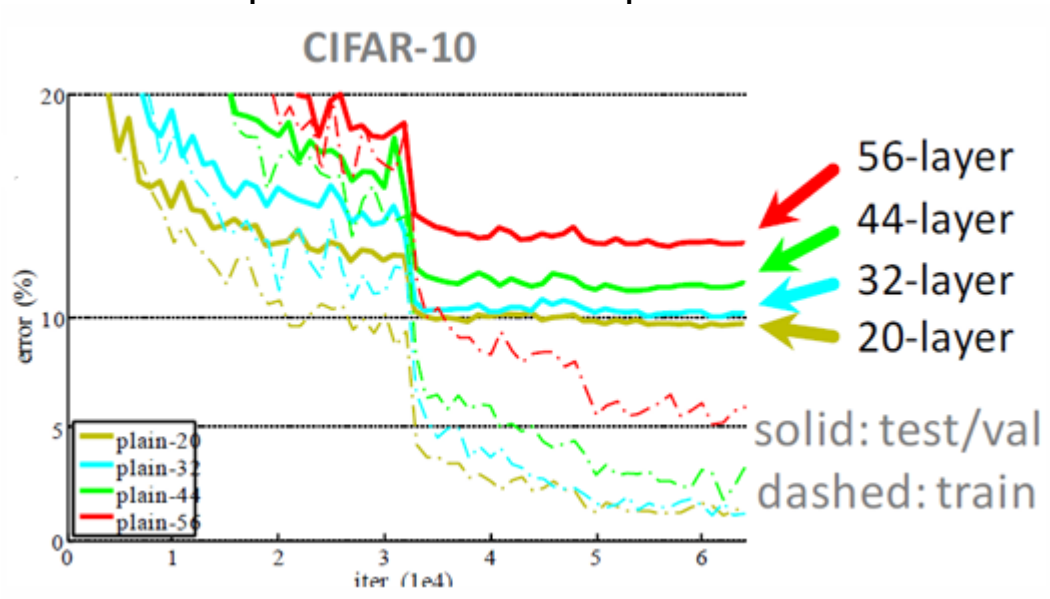
La profondeur des représentations est d'une importance capitale pour de nombreuses tâches de reconnaissance visuelle.
Grâce à leurs représentations extrêmement profondes, ils ont obtenu une amélioration relative de 28 % sur le jeu de données de détection d'objets COCO.

Les réseaux résiduels profonds sont à la base de soumissions aux concours ILSVRC et COCO 2015, où ils ont également remporté la première place pour les tâches de détection ImageNet, de localisation ImageNet, de détection COCO et de segmentation COCO.



Introduction

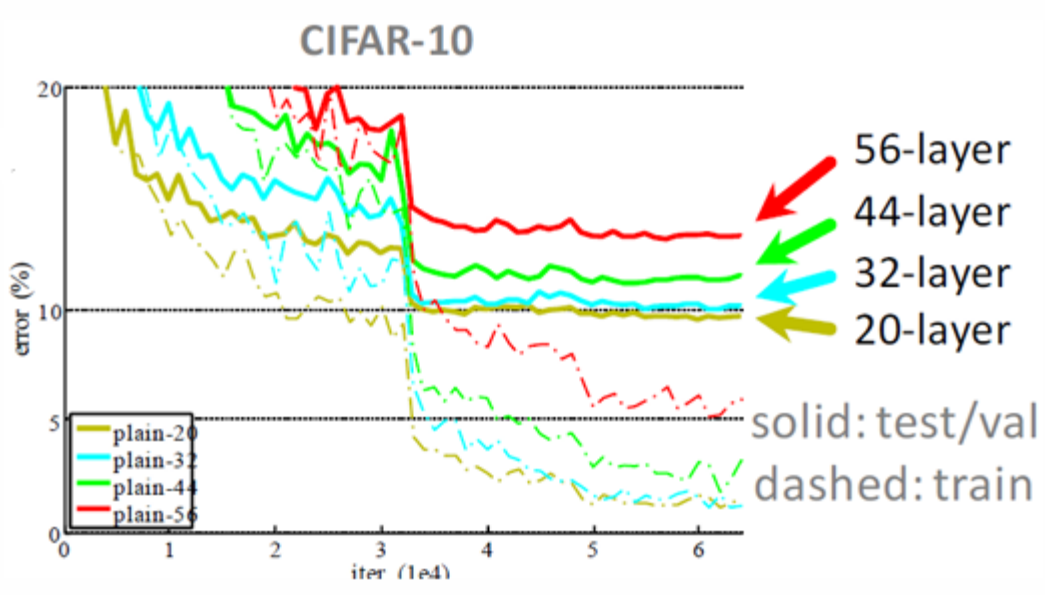
- Vise l'entraînement de réseaux très profonds (30+ couches)
- Problème du vanishing gradient en partie réglé par la Batch Norm
- Constat : dégradation des résultats passé une certaine profondeur



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

Les limites

Lorsque des réseaux plus profonds peuvent commencer à converger, un problème de dégradation est apparu : avec l'augmentation de la profondeur du réseau, la précision est saturée (ce qui n'est pas surprenant) et se dégrade ensuite rapidement.
De manière inattendue, cette dégradation n'est pas due à un surajustement, et l'ajout de couches supplémentaires à un modèle suffisamment profond entraîne une erreur d'apprentissage plus élevée et vérifié de manière approfondie par les expériences.



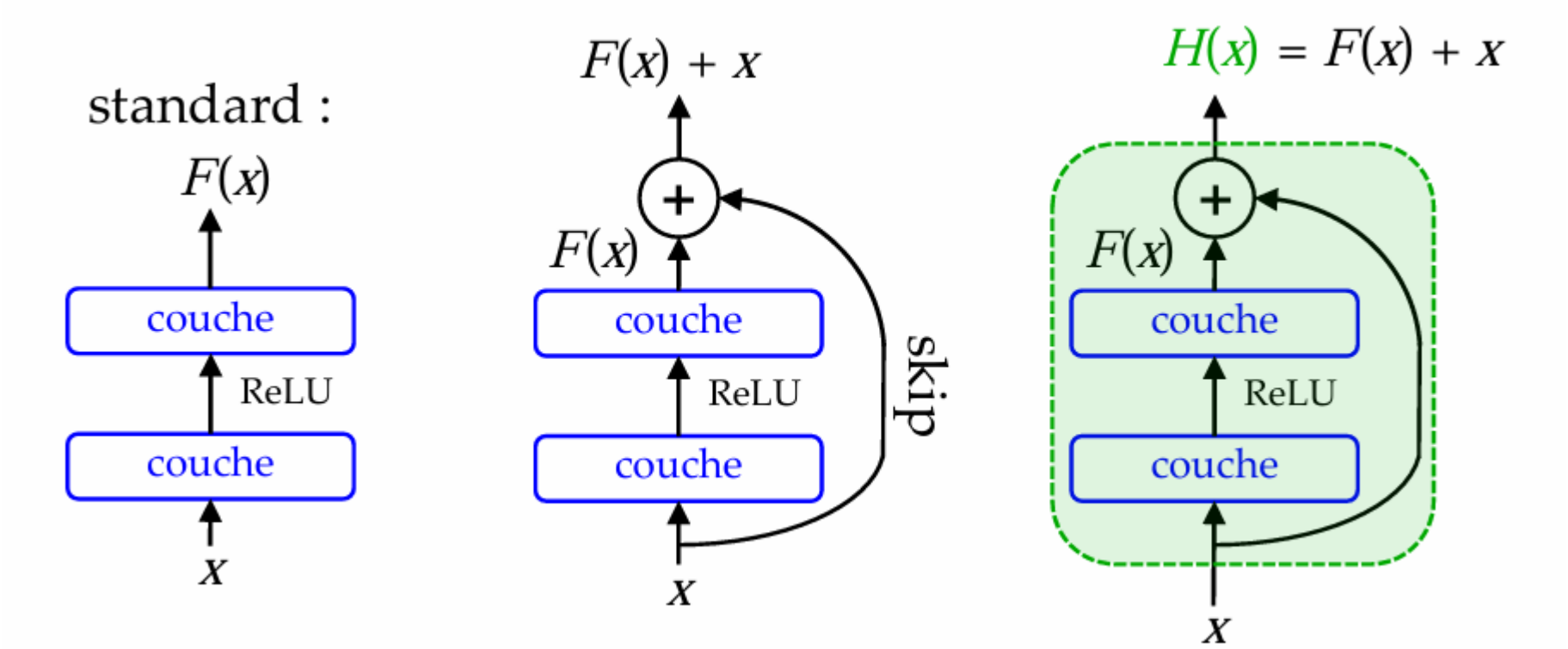
Les limites

- Dilution de l’information de correction (gradient)

- Difficile pour une couche de réutiliser des features des couches précédentes
– perte de l’information flow

- Difficulté d’apprendre la fonction identitaire

ResNet : idée de base

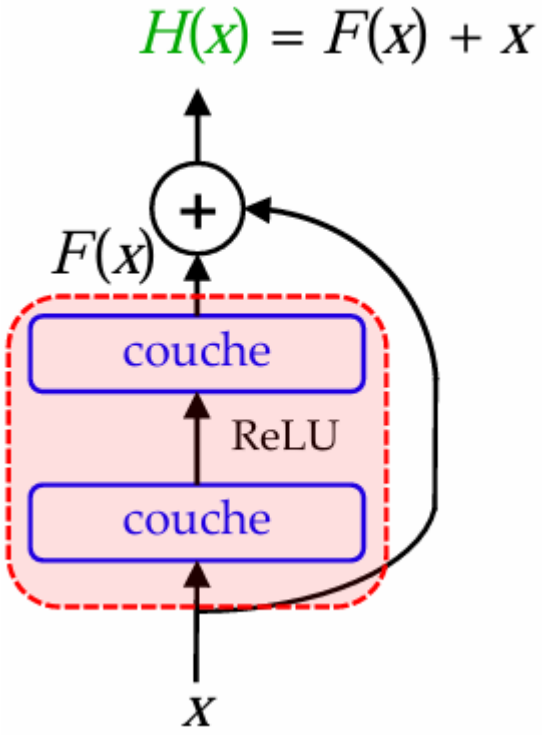


F(x) est le résiduel entre la fonction H(x) désirée et la fonction identitaire : F(x) = H(x) - x

- 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)

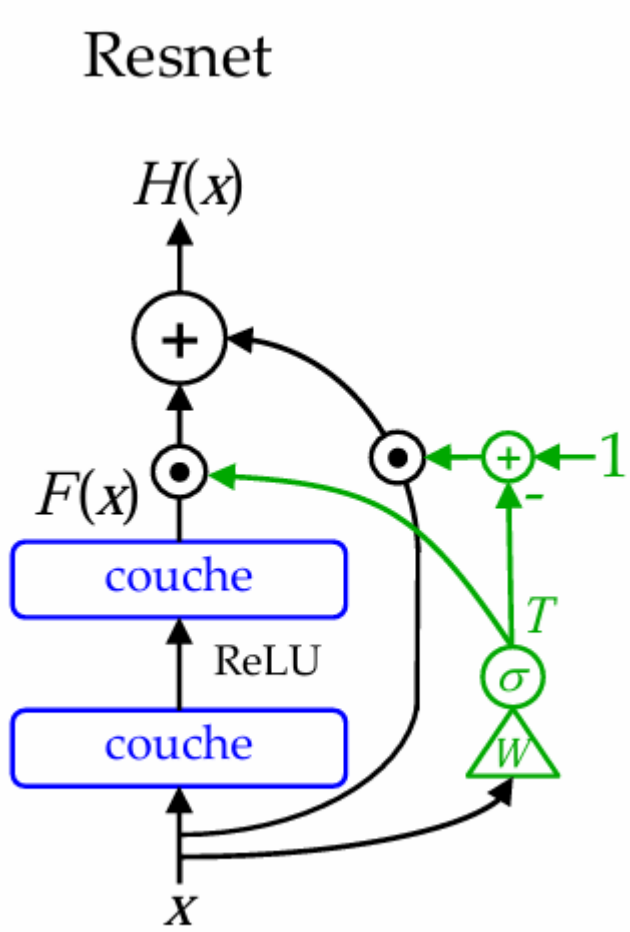
Highway network

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

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

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

T(x) = poids de gating,



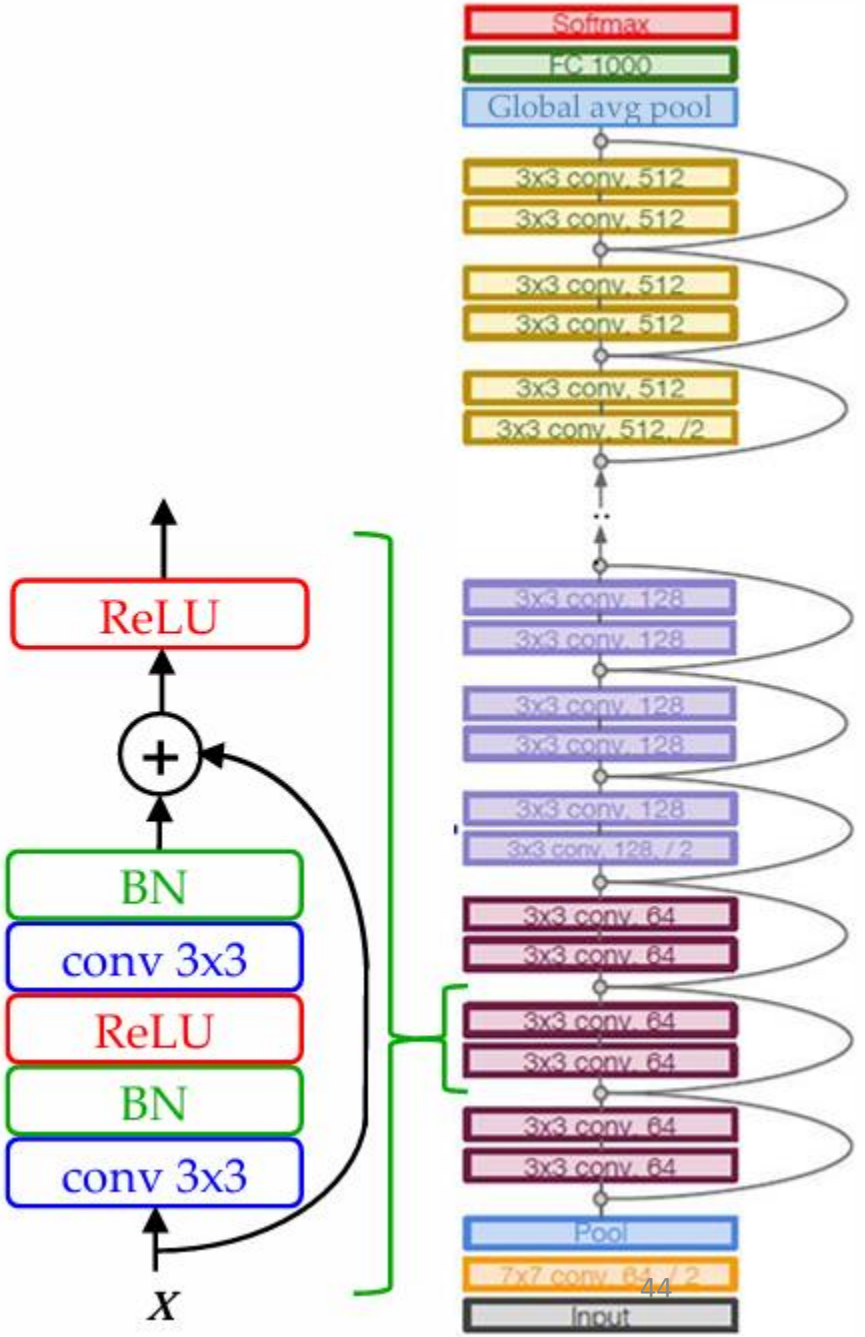
- ResNet a fait le pari qu'il est toujours mieux de faire la somme des deux : architecture plus simple

Srivastava, R. et al., Training Very Deep Networks, NIPS 2015

Différences avec ResNet

Élément	Gated Block	ResNet
Formule	$H(x)=F(x)T+x(1-T)$	$H(x)=F(x)+x$
Contrôle	Adaptatif, dépend de l'entrée	Fixe (toujours somme directe)
Complexité	Plus complexe (paramètres en plus)	Plus simple, plus rapide
Philosophie	Mélange souple entre identité et transformation	Addition directe, sans choix

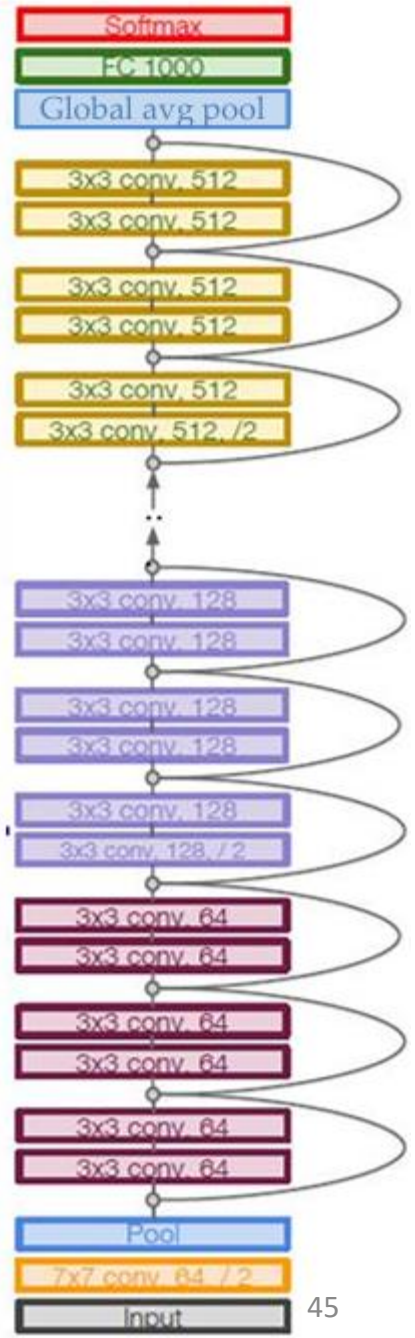
ResNet


Softmax
FC 1000
Global avg pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
Pool
7x7 conv, 64, / 2
Input
ReLU
BN
conv 3x3
ReLU
BN
conv 3x3
x

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



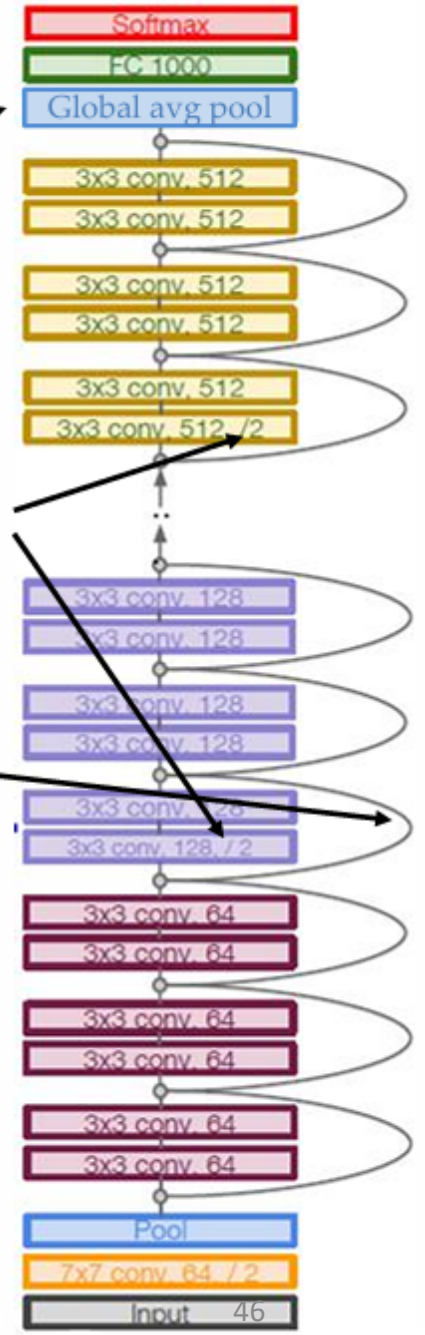
ResNet

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



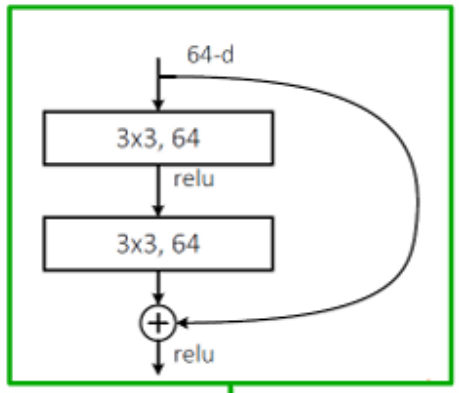
Dans certains cas, besoin de warm up

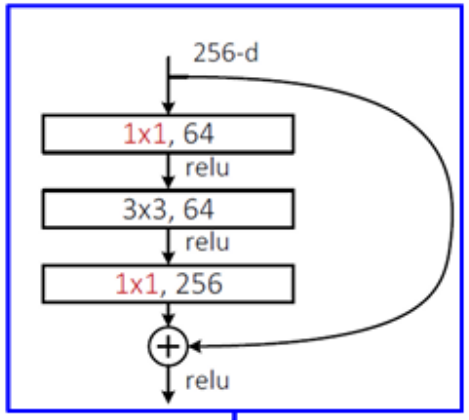
"…we find that the initial learning rate of 0.1 is slightly too large to start converging. So we use 0.01 to warm up the training until the training error is below 80% (about 400 iterations), and then go back to 0.1 and continue training."

ResNet

Bottleneck

(pour efficacité, pas pour régularisation)

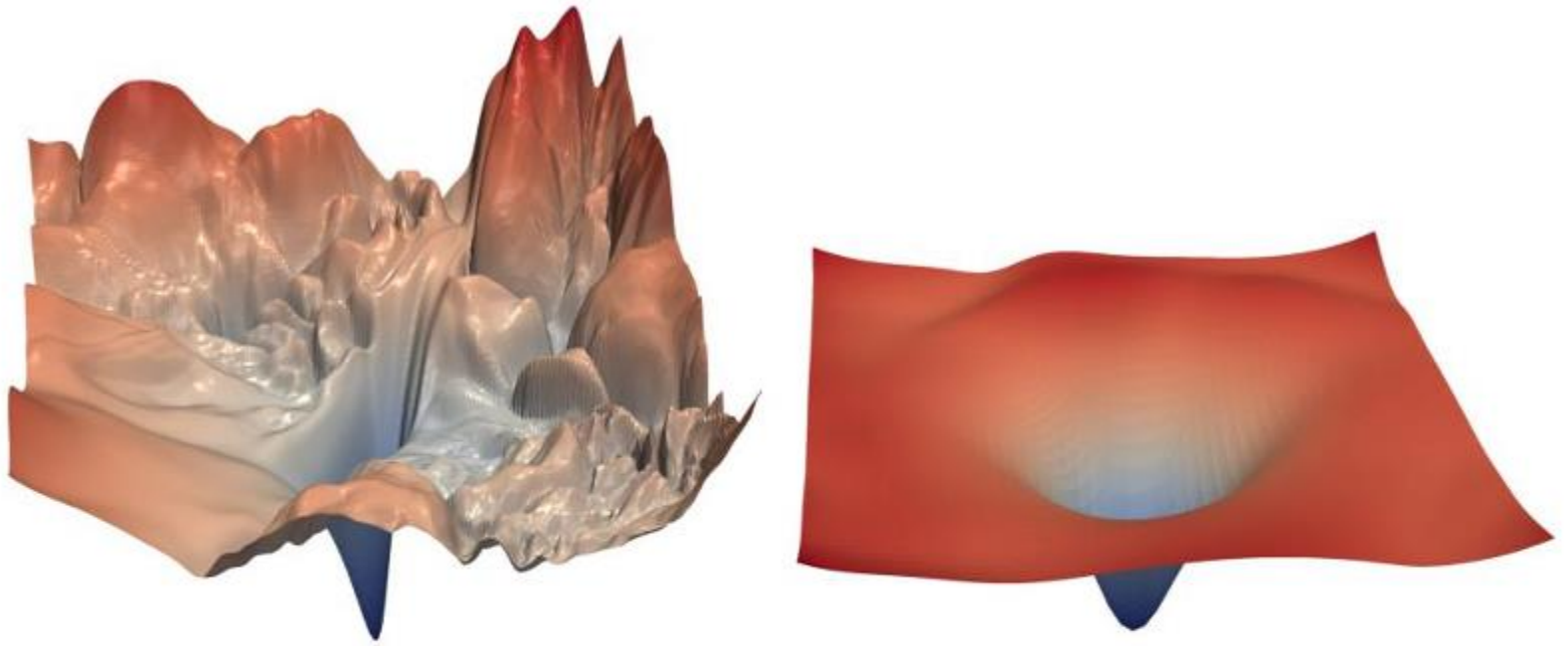




layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	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

6

ResNet : lissage de la fonction de coût



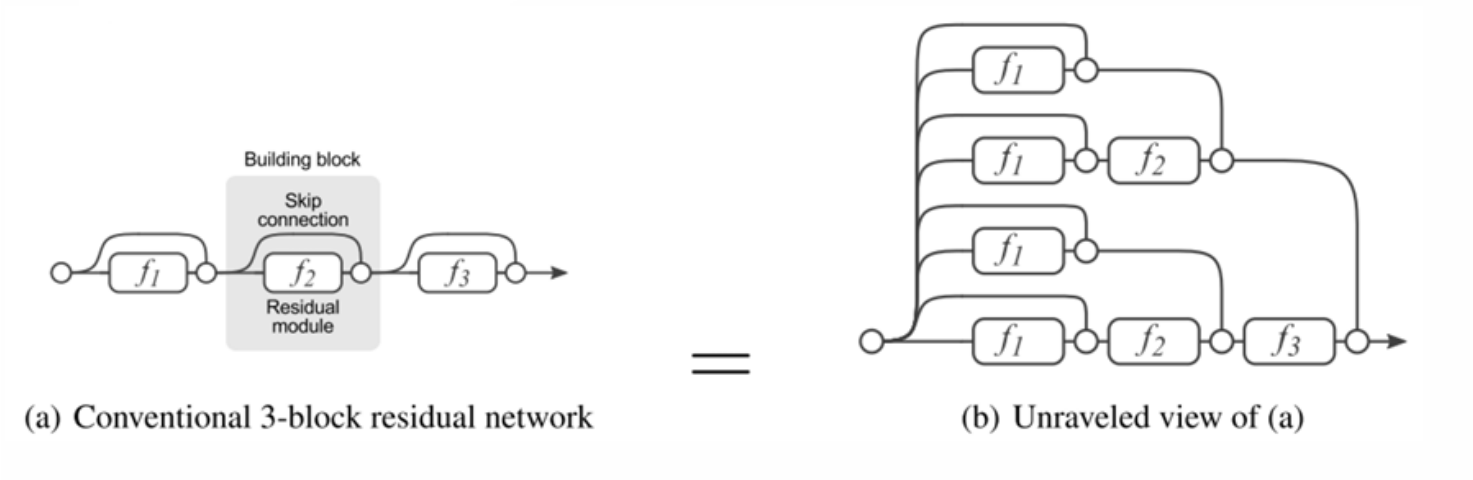
(a) without skip connections (b) with skip connections

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^{N-1} de réseaux (similaire à Dropout)



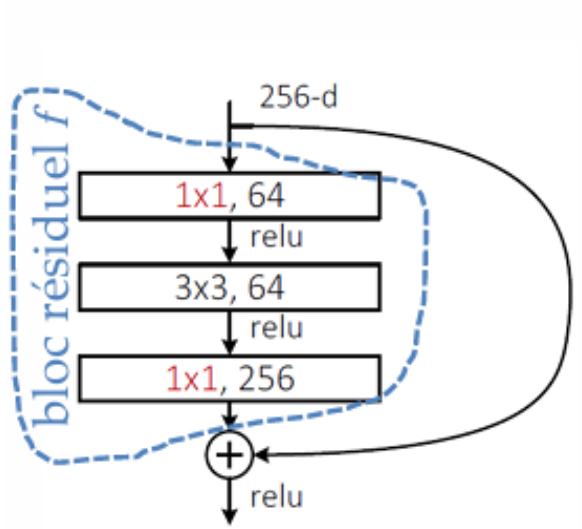
(a) Conventional 3-block residual network (b) Unraveled view of (a)

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

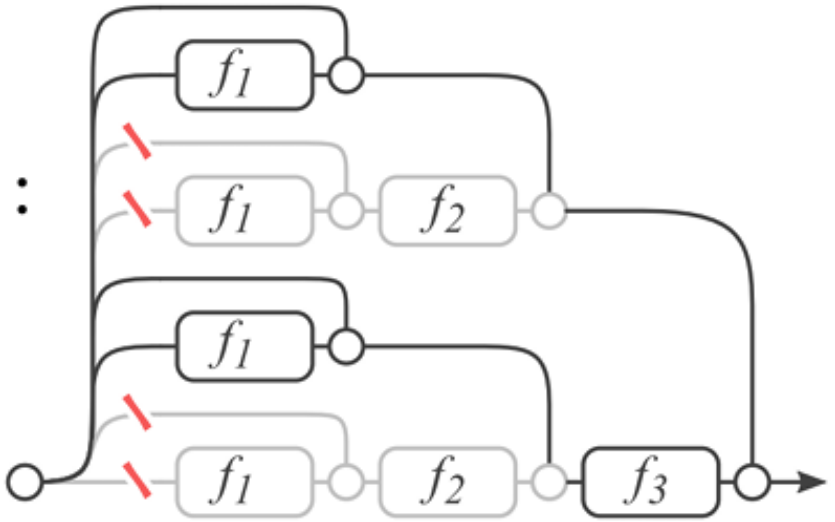
A. Veit, M. Wilber and S. Belongie. Residual Networks Behave Like Ensembles of Relatively Shallow Networks. arXiv:1605.06431v2,2016.

ResNet

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



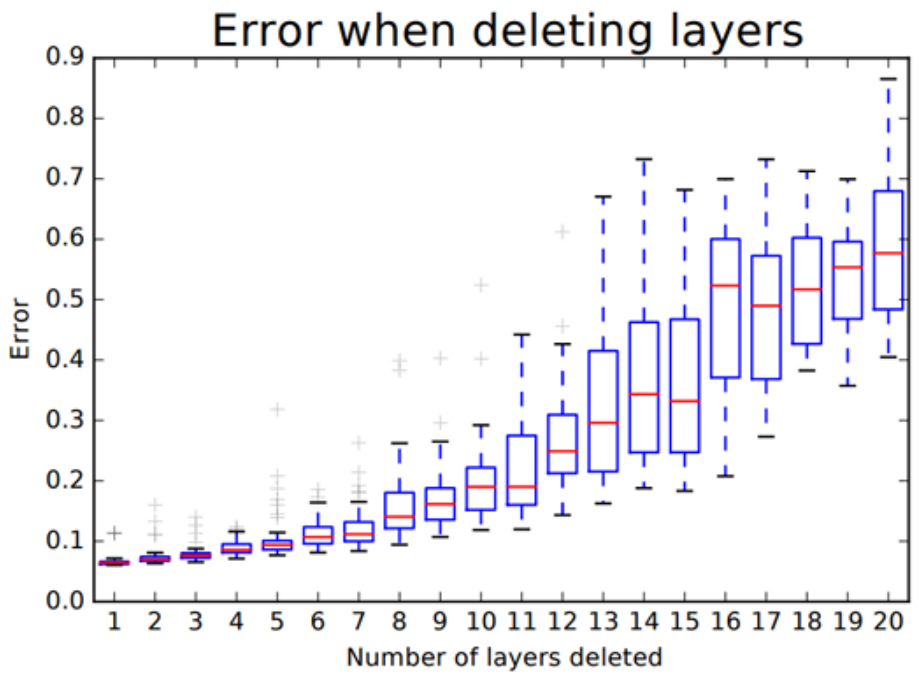
On retire f_2 :



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

ResNet

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

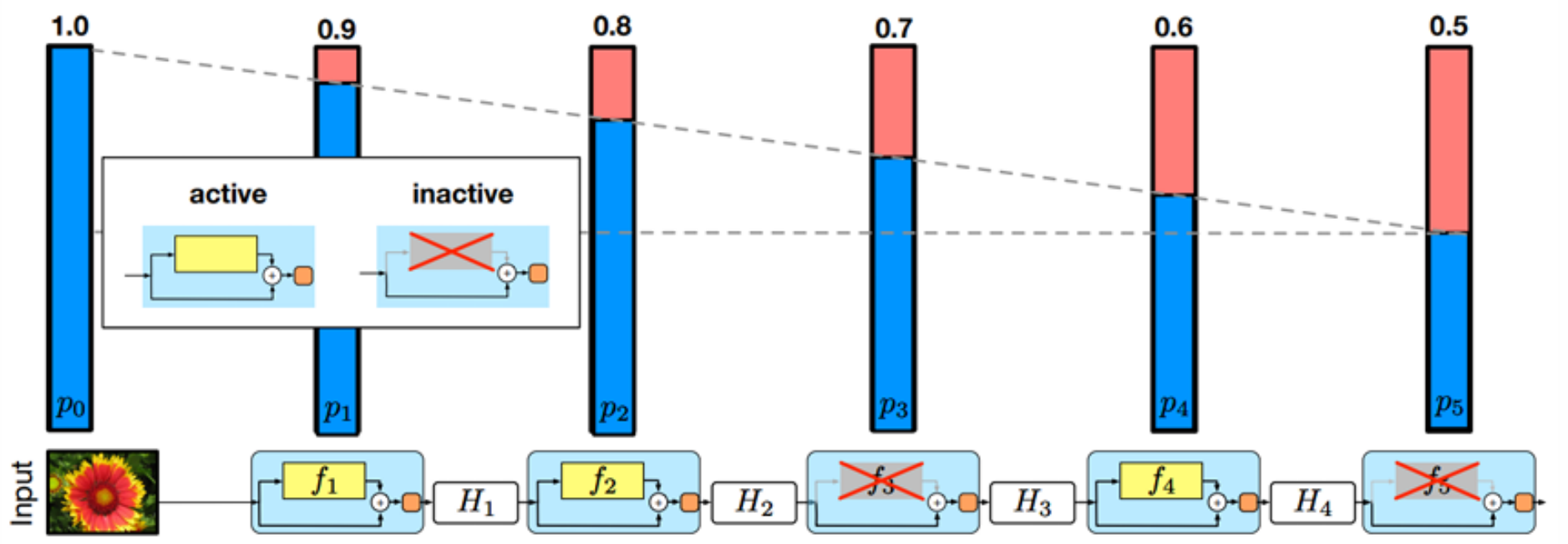


- Retrait de couche : catastrophique pour VGG!

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

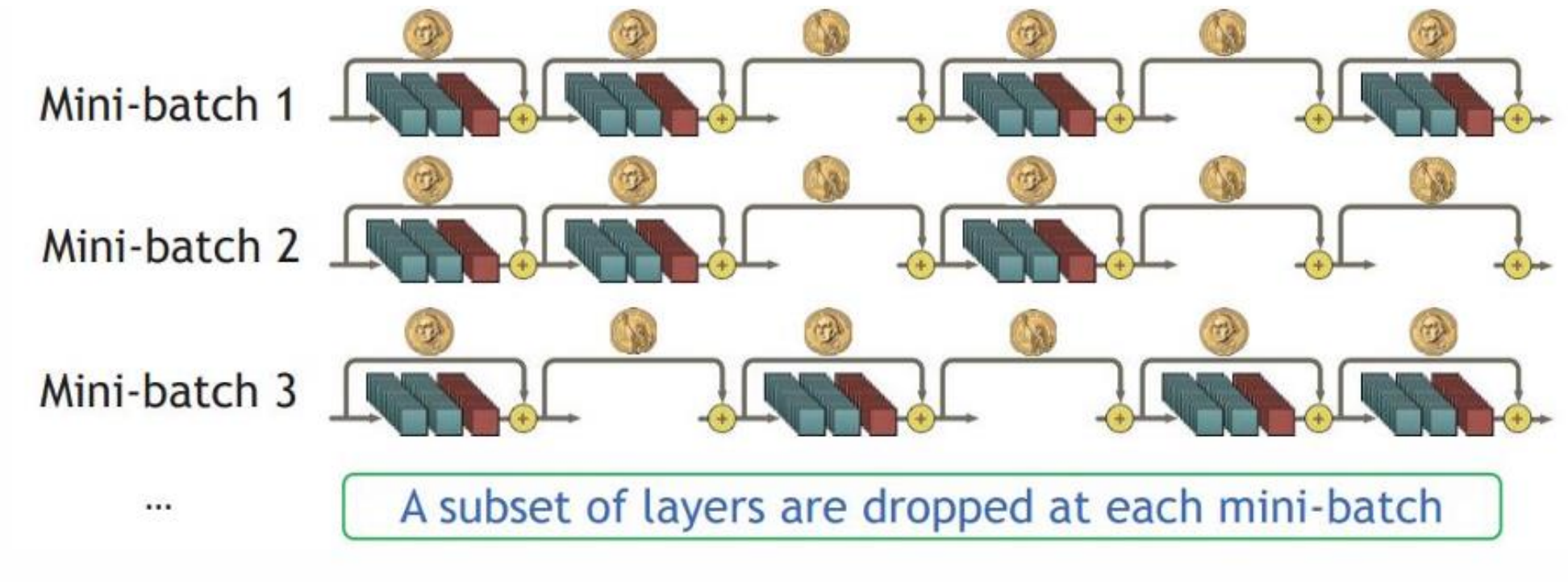
Stochastic Depth

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



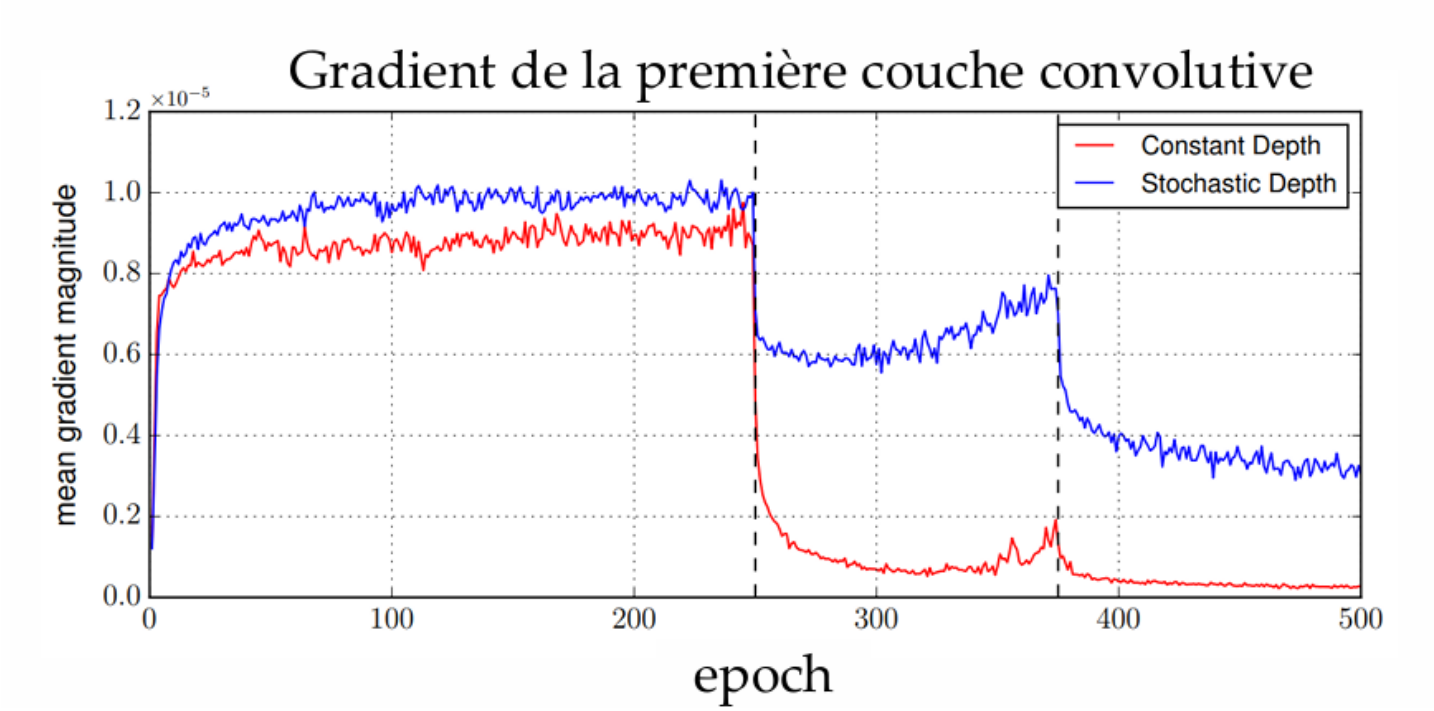
Huang et al. Deep Network with Stochastic Depth, 2016.

Stochastic Depth



Stochastic Depth

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

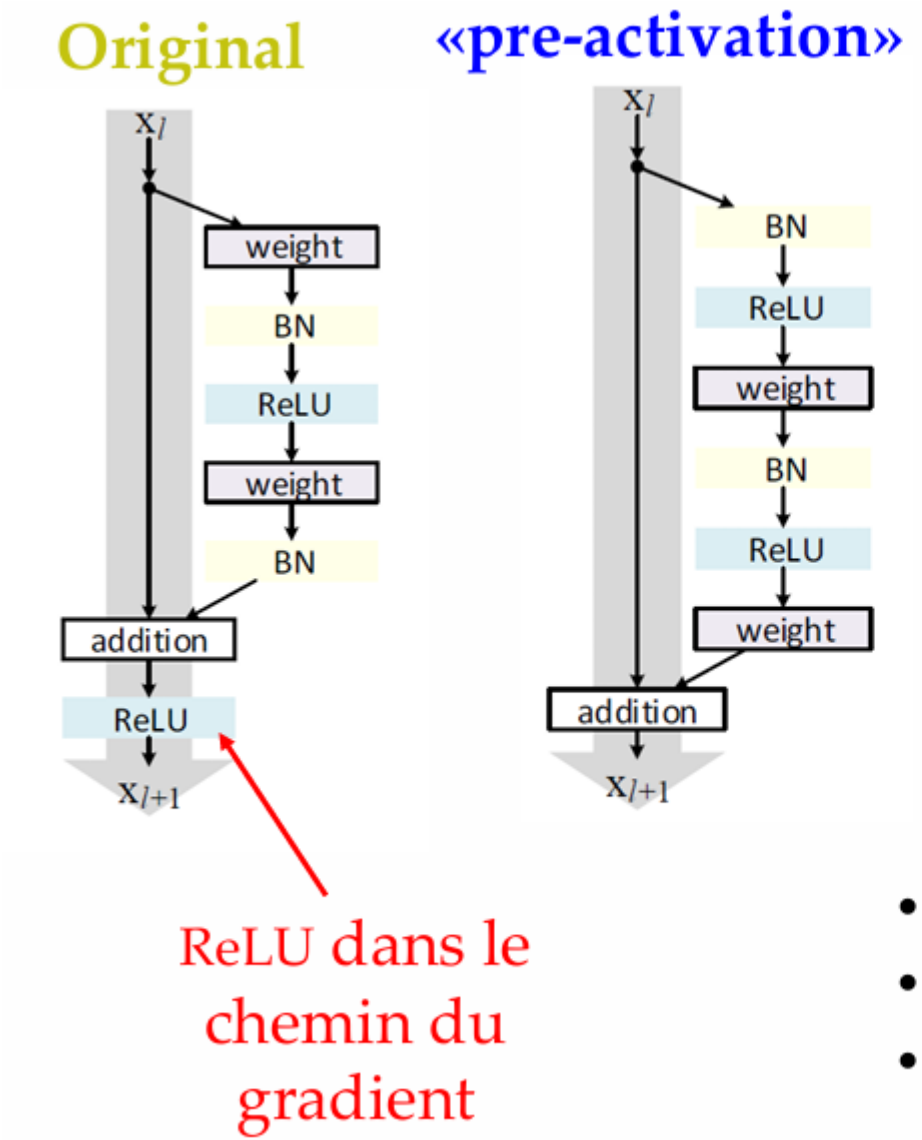


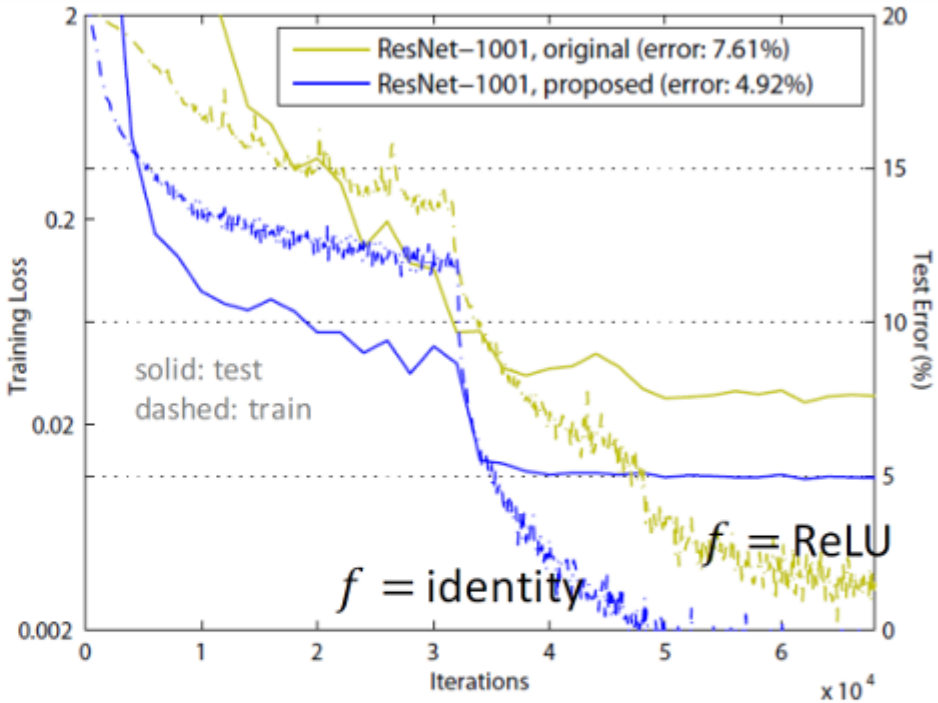
Huang et al. Deep Network with Stochastic Depth, 2016.

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 p1 droper le bloc l)
- Forme de régularisation
- Utilise le plein réseau en test
  – comme avec dropout
  – calibrer les forces des features en fonction de p1

ResNet version preactivation





tiré : ICML workshop 2016, He.

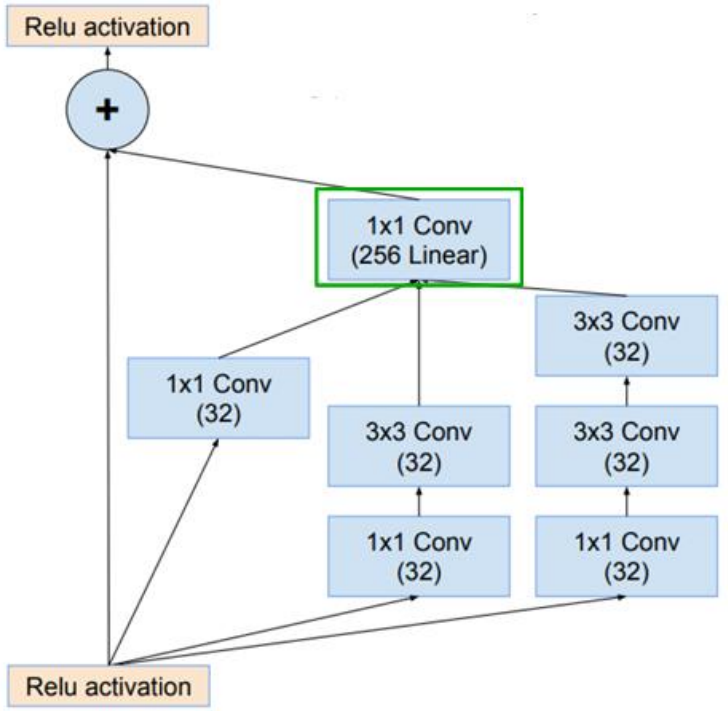
- Bloc résiduel amélioré
- Meilleur flot du gradient
- Améliore les résultats
  - 6.7% → 4.8% top-5

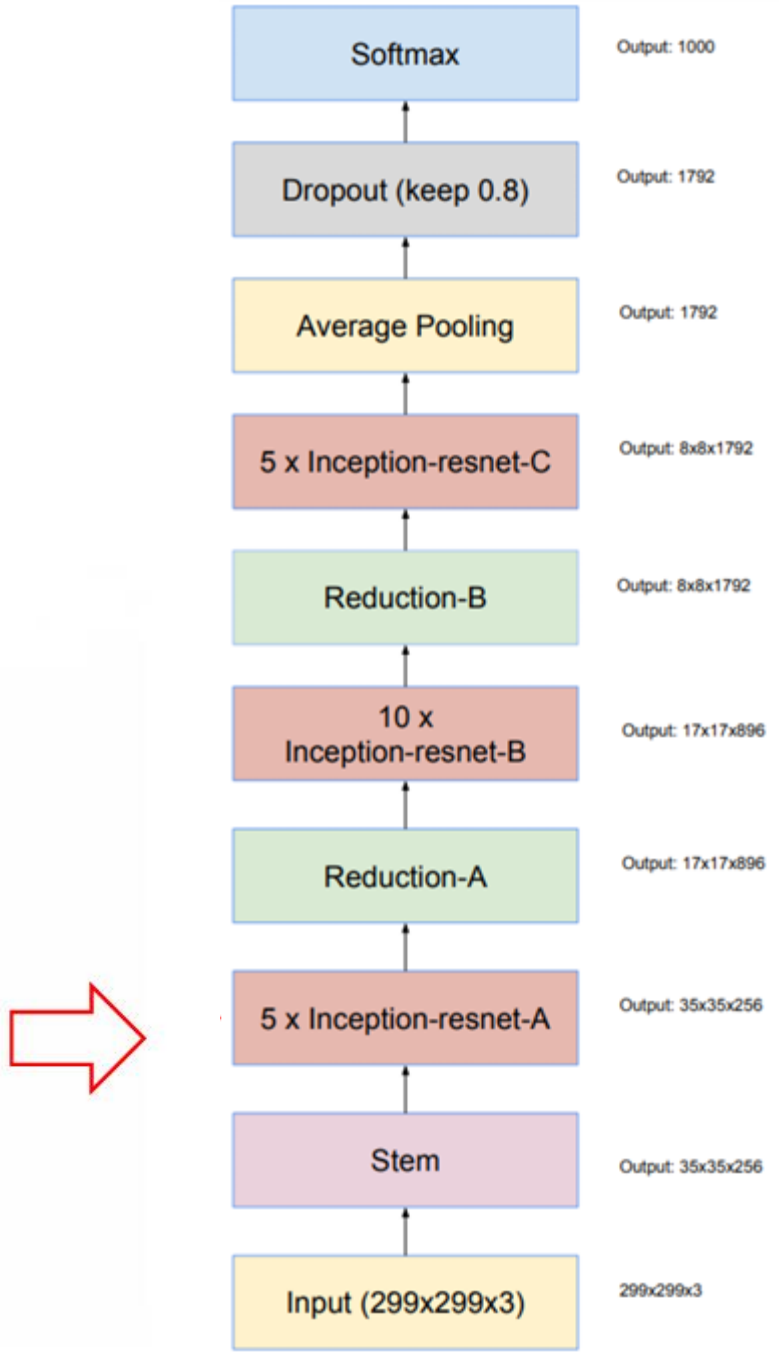
He et al., Identity Mappings in Deep Residual Networks, ECCV 2016.

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.





- 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 profondeur1/largeur2
- Cherche à améliorer la performance sous un budget fixe de FLOPS et de paramètres
- (Visitera ce sujet avec EfficientNet)

- 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

Les réseaux neuronaux convolutifs (ConvNets) sont généralement développés avec un budget de ressources fixe, puis augmentés pour une meilleure précision si davantage de ressources sont disponibles.

Nous allons étudier de manière systématique la mise à l'échelle des modèles et nous constatons qu'un équilibre judicieux entre la profondeur, la largeur et la résolution du réseau peut conduire à de meilleures performances.
Sur la base de cette observation, une nouvelle méthode de mise à l'échelle a été proposé qui met uniformément à l'échelle toutes les dimensions de profondeur/largeur/résolution à l'aide d'un coefficient composé simple mais très efficace.
Nous démontrons l'efficacité de cette méthode pour la mise à l'échelle des MobileNets et des ResNet.

Pour aller encore plus loin, nous utilisons la recherche d'architecture neuronale pour concevoir un nouveau réseau de base et le mettre à l'échelle pour obtenir une famille de modèles, appelés EfficientNets, qui atteignent une précision et une efficacité bien meilleures que les ConvNets précédents. En particulier, EfficientNet-B7 atteint une précision de pointe de 84,3 % sur ImageNet, tout en étant 8,4 fois plus petit et 6,1 fois plus rapide pour l'inférence que le meilleur ConvNet existant.

EfficientNets se transfèrent également bien et atteignent une précision de pointe sur CIFAR-100 (91,7 %), Flowers (98,8 %) et 3 autres ensembles de données d'apprentissage par transfert, avec un ordre de grandeur de moins de paramètres.

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)
- Fait une recherche de réseaux de neurones automatisé (NAS)
  - EfficientNet 8.4x plus petit et 6.1x plus rapide
- Exploite SENet + depthwise conv.

Source code is at
https: //github.com/tensorflow/tpu/tree/master/models/official/efficientnet

EfficientNet

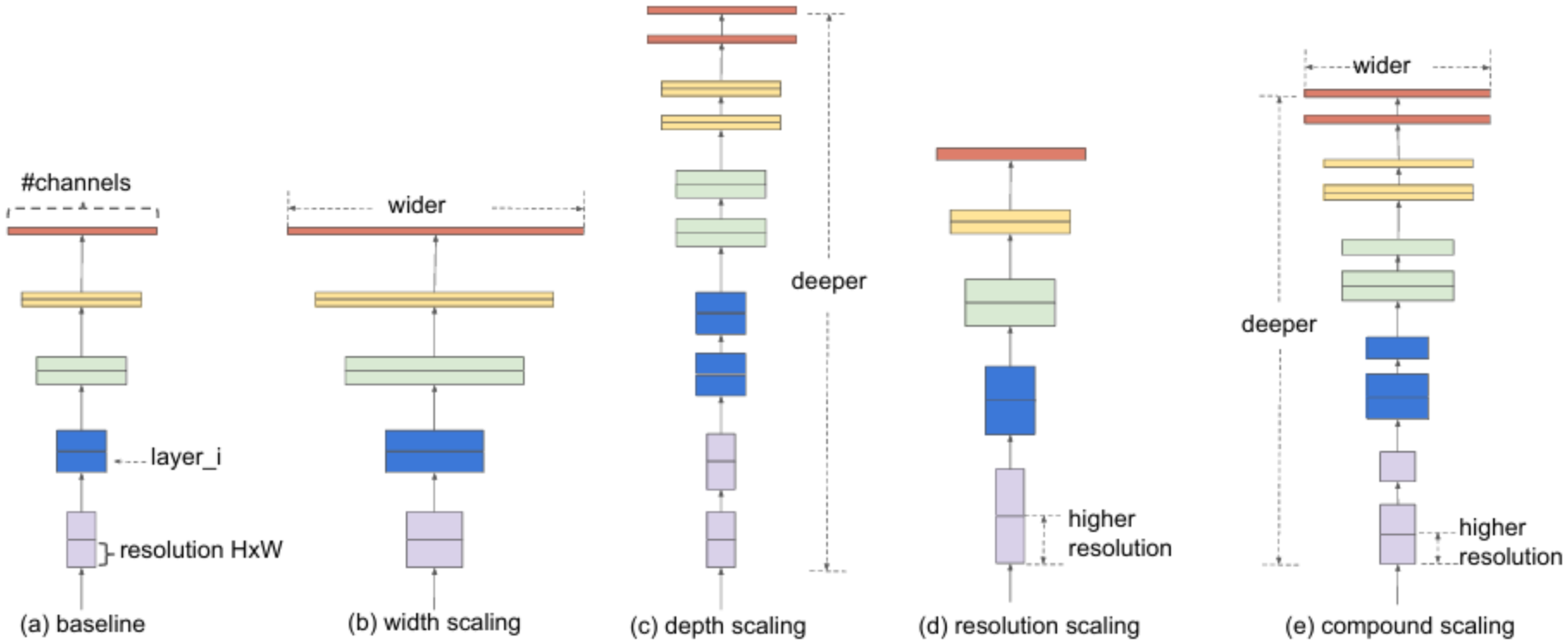
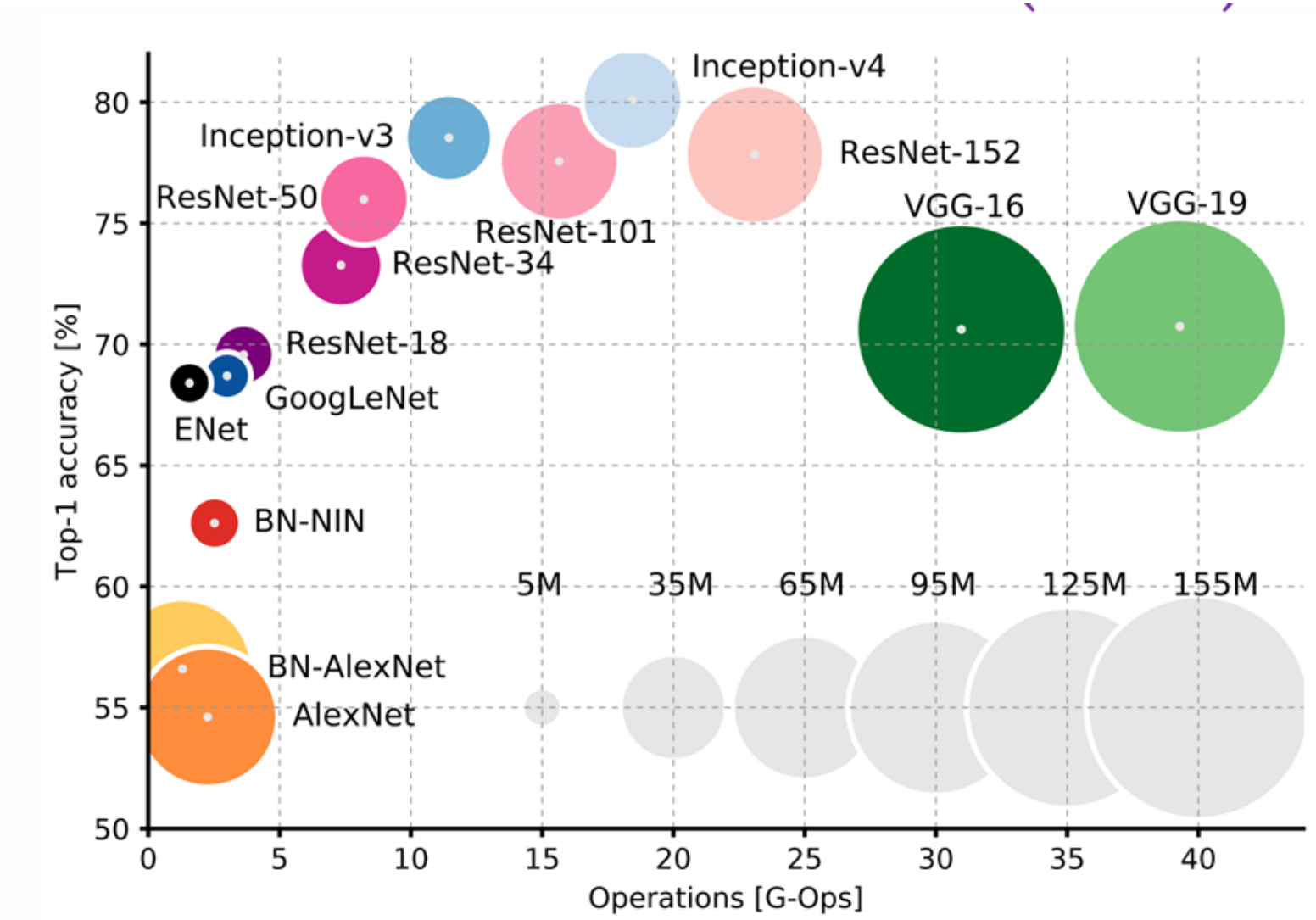


Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

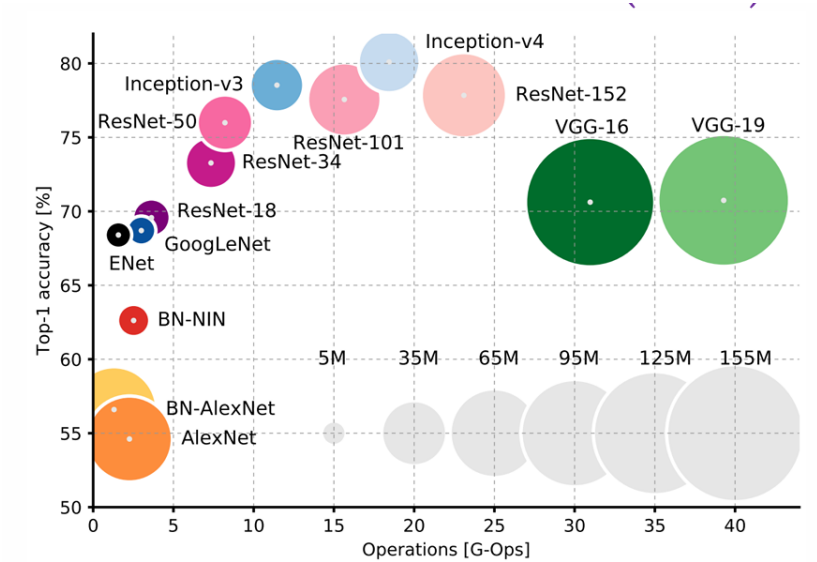
EfficientNet



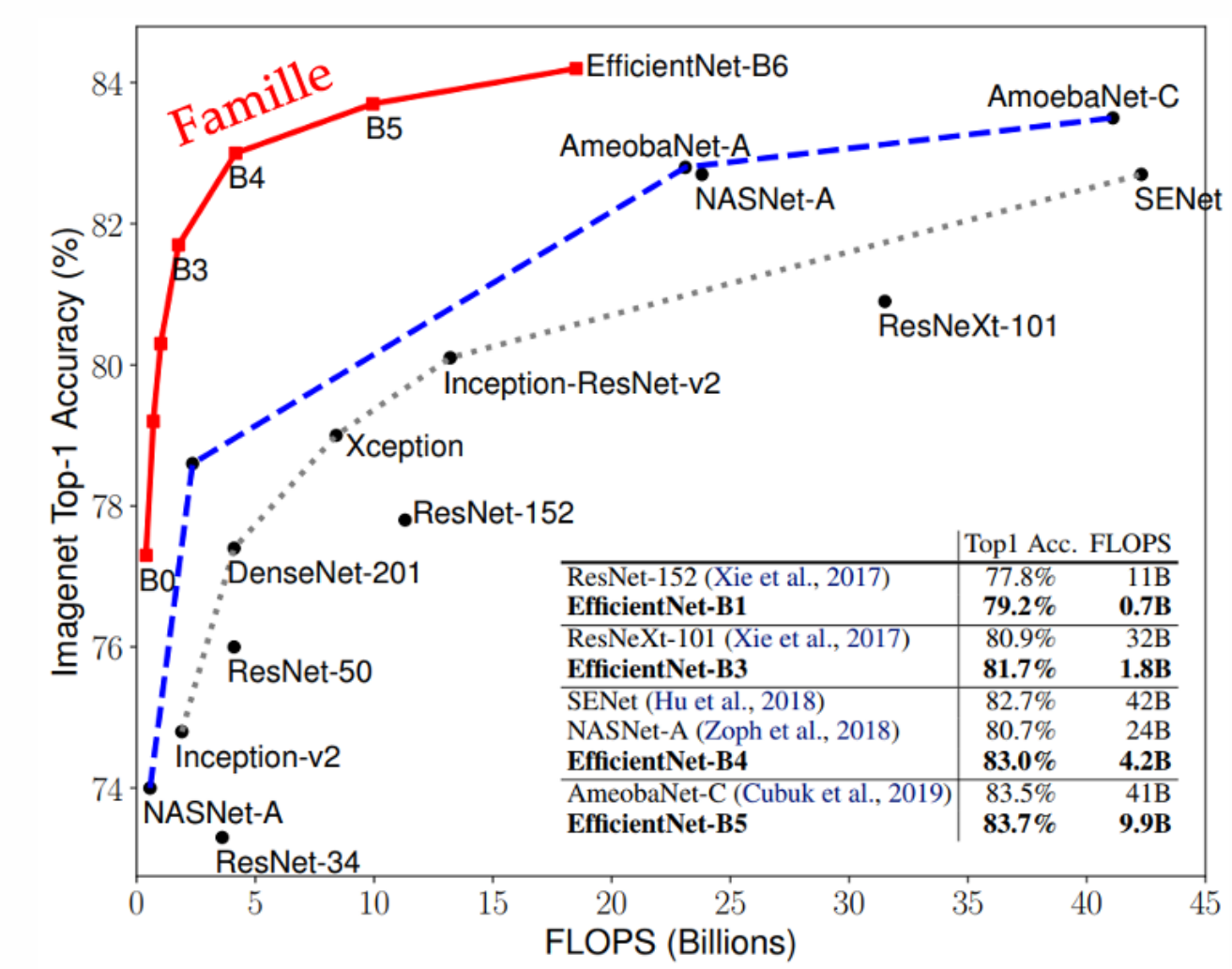
https://arxiv.org/pdf/1905.11946

La figure offre une vue différente, mais plus informative, des valeurs de précision, car elle visualise également le coût de calcul et le nombre de paramètres du réseau. La première chose qui saute aux yeux est que le VGG, même s'il est largement utilisé dans de nombreuses applications, est de loin l'architecture la plus coûteuse - à la fois en termes de besoins de calcul et de nombre de paramètres.

Ses implémentations à 16 et 19 couches sont en fait isolées de tous les autres réseaux. Les autres architectures forment une ligne droite raide, qui semble commencer à s'aplanir avec les dernières incarnations d'Inception et de ResNet. Cela pourrait suggérer que les modèles atteignent un point d'inflexion sur cet ensemble de données.
À ce point d'inflexion, les coûts - en termes de complexité - commencent à l'emporter sur les gains de précision.



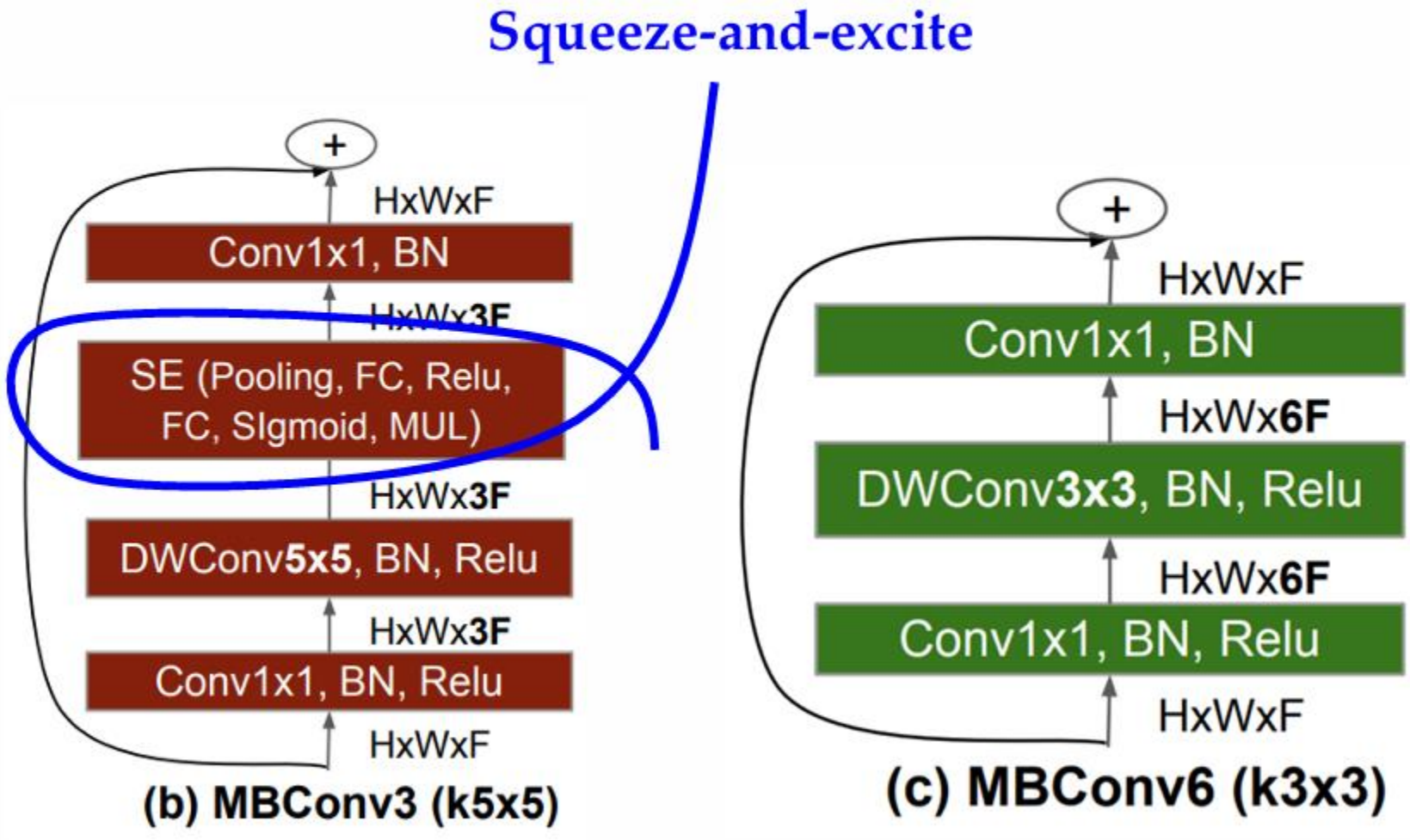
EfficientNet



	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**

https://arxiv.org/pdf/1905.11946

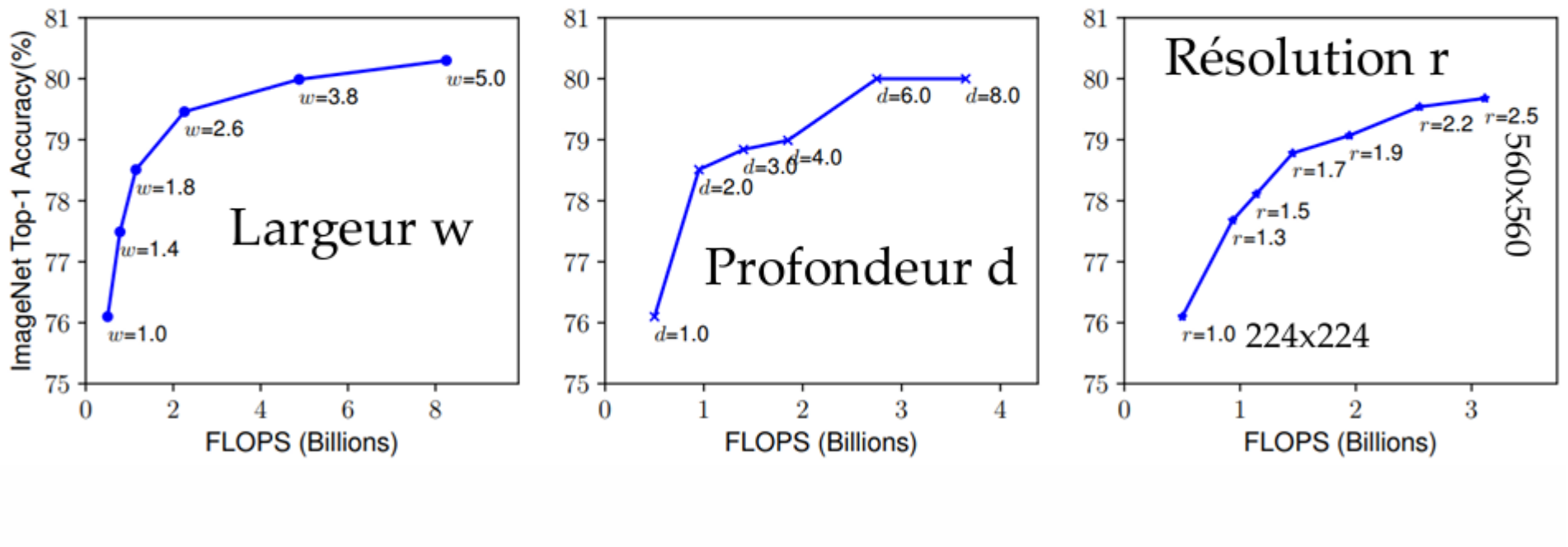
Bloc de base : MBConv

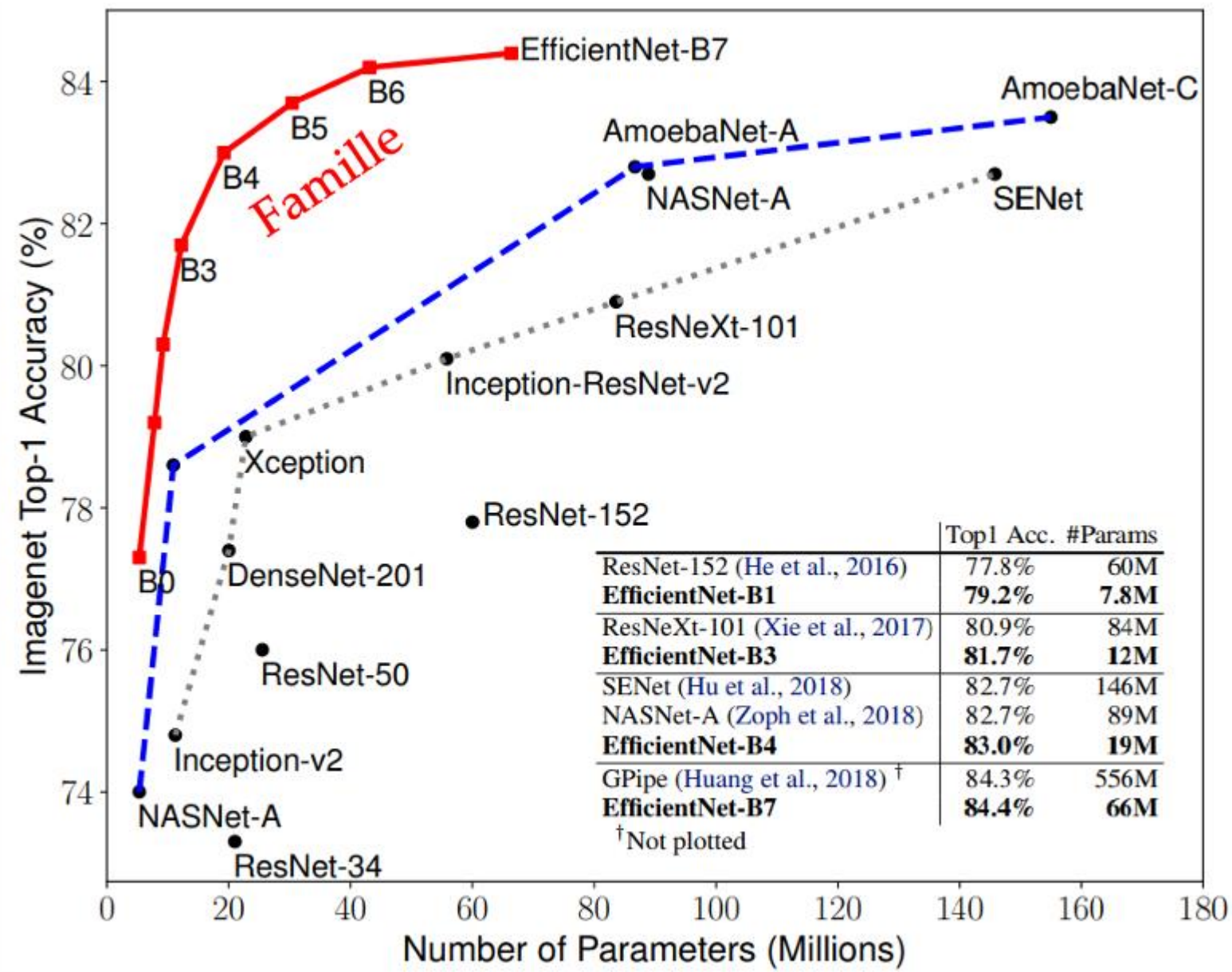


- Identifié un modèle de base via NAS
  - EfficientNet-B0

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

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





	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

Fin