# GAP, Localization and CNN Explanations

Giacomo Boracchi

giacomo.boracchi@polimi.it

# Global Averaging Pooling

G. Boracchi

# Network In Network

**Min Lin[1,2], Qiang Chen[2], Shuicheng Yan[2]**
[1]Graduate School for Integrative Sciences and Engineering
[2]Department of Electronic & Computer Engineering
National University of Singapore, Singapore
{linmin,chenqiang,eleyans}@nus.edu.sg

Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." *ICLR 2014*

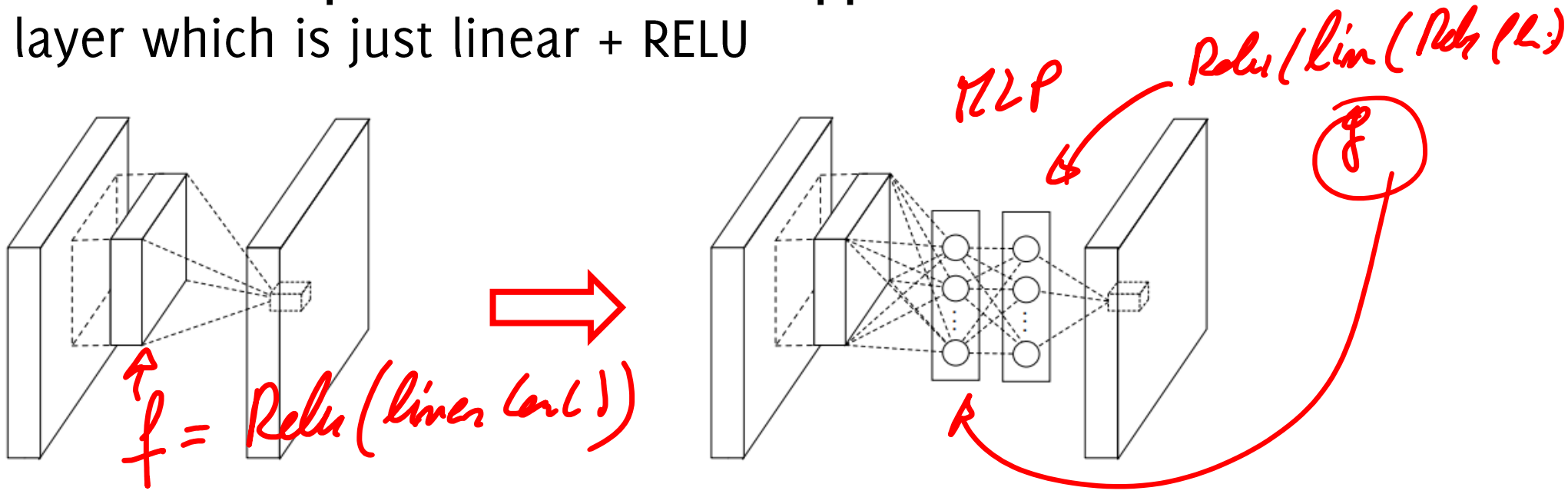
# Network in Network

MLP conv

**Mlpconv layers:** instead of traditional convolutions, **a stack of 1x1 convolutions + RELU**

- 1x1 convolutions used in a stack followed by RELU corresponds to a MLP networks **used in a sliding manner on the whole image**

Each layer features a **more powerful functional approximation** than a convolutional layer which is just linear + RELU



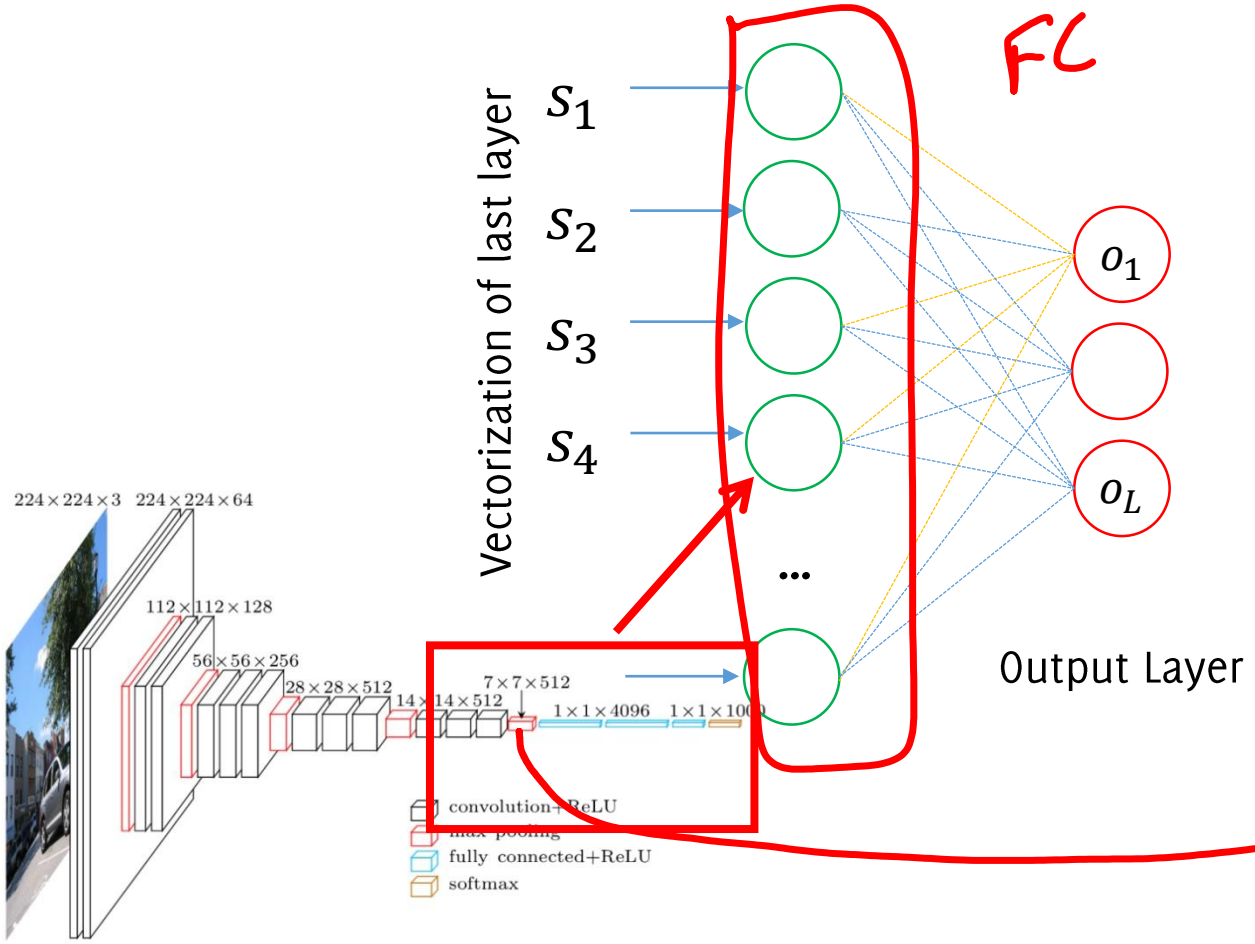

(a) Linear convolution layer (b) Mlpconv layer

Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." *ICLR 2014*

# Network in Network

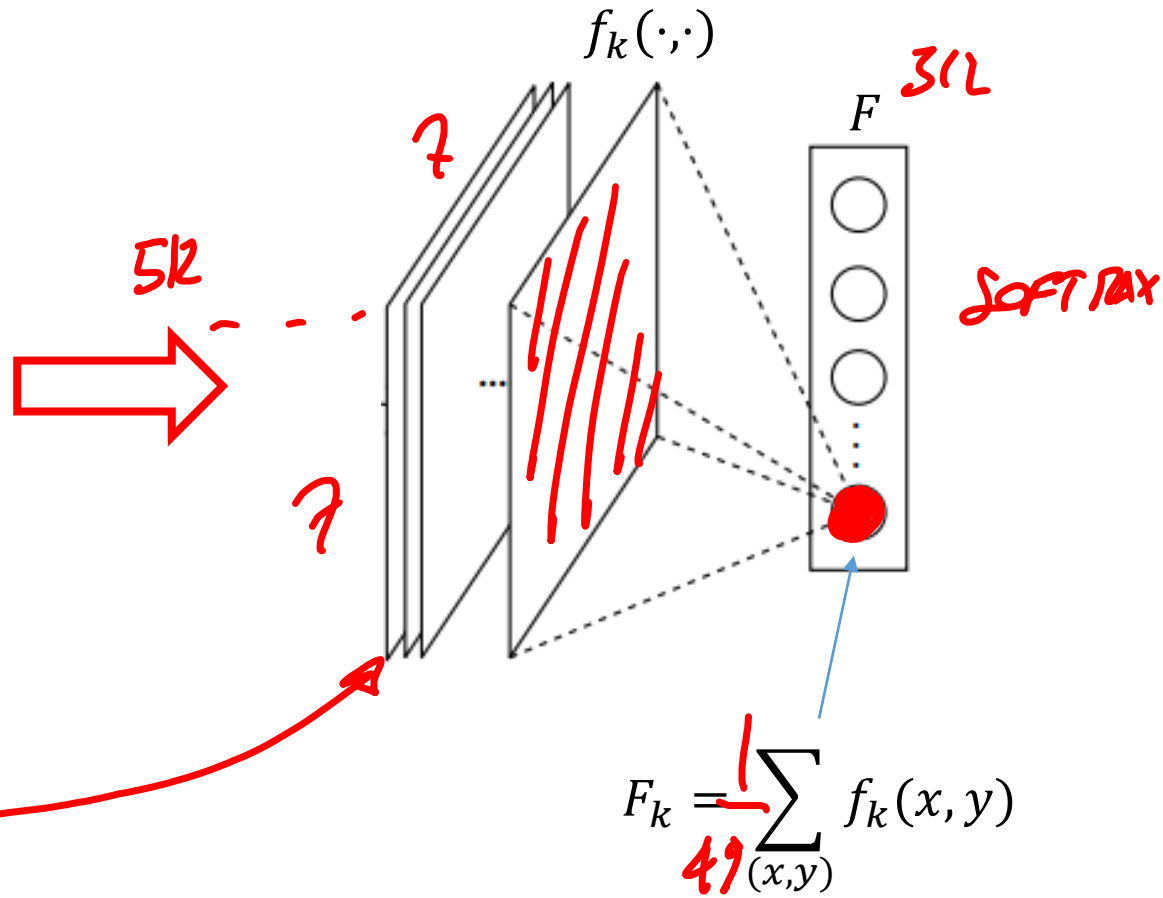
**They introduce Global Averaging Pooling Layers**

Fully Connected Layer




GAP

Global Averaging Pooling Layer

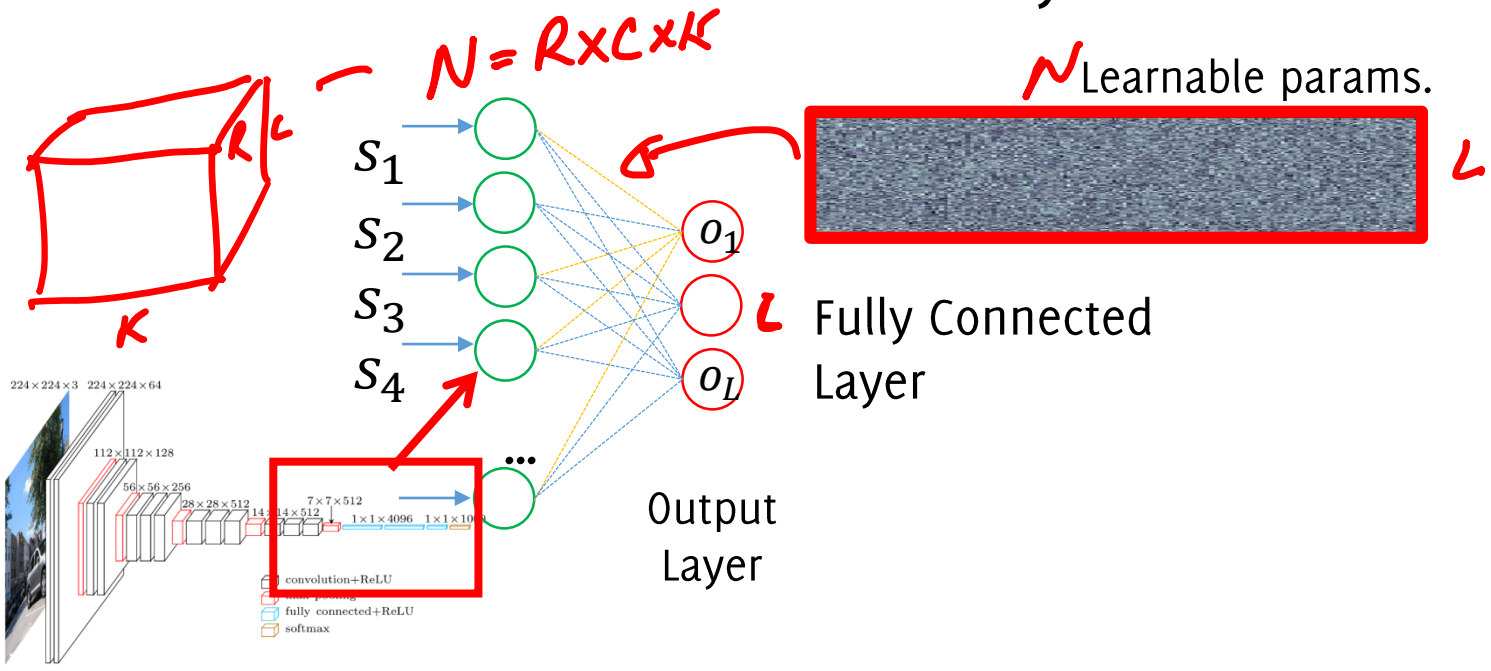


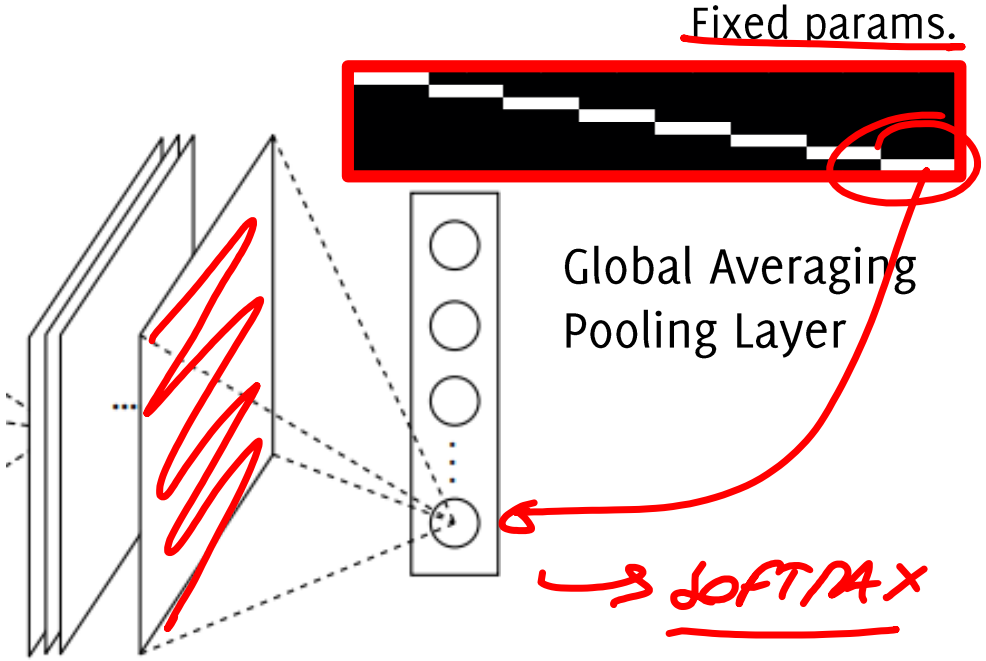

Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." *ICLR 2014*

# Network in Network: GAP

**Global Averaging Pooling Layers:** instead of a FC layer at the end of the network, **compute the average of each feature map.**

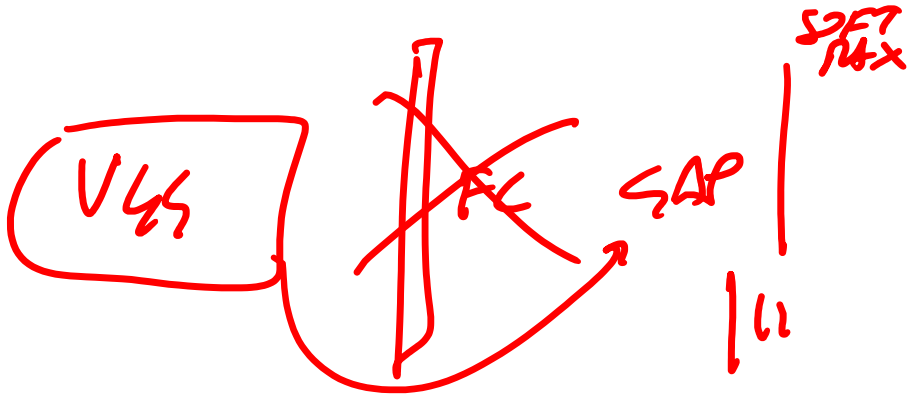
- The transformation corresponding **to GAP is a block diagonal, constant matrix** (consider the input unrolled layer-wise in a vector)
- The transformation of each layer in **MLP corresponds to a dense matrix.**



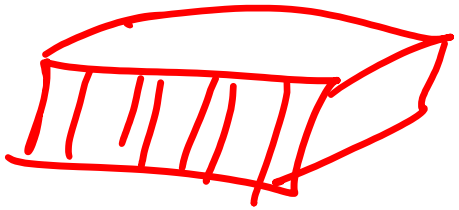

Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." *ICLR 2014*

# Rationale behind GAP




**Fully connected layers are prone to overfitting**

- They have many parameters
- Dropout was proposed as a regularized that randomly sets to zero a percentage of activations in the FC layers during training



The GAP strategy is:

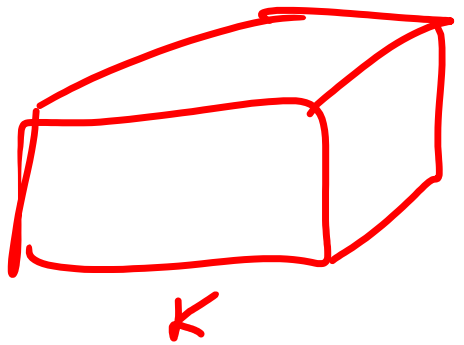
1. **Remove the fully connected layer** at the end of the network!
2. **Introduce a GAP layer.**
3. **Predict by a simple soft-max after the GAP.**

**Watch out:** the number of feature maps has to correspond to the number of output classes!

IN THIS CASE

# The Advantages of GAP Layers:

- No parameters to optimize, lighter networks less prone to overfitting
- Classification is performed by a softMax layer at the end of the GAP
- More interpretability, creates a direct connection between layers and classes output (we'll see soon)
- This makes GAP a structural regularizer
- **Increases robustness to spatial transformation** of the input images
- **The network can be used to classify images of different sizes**




Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." ICLR 2014

# Network in Network

The whole NiN stacks

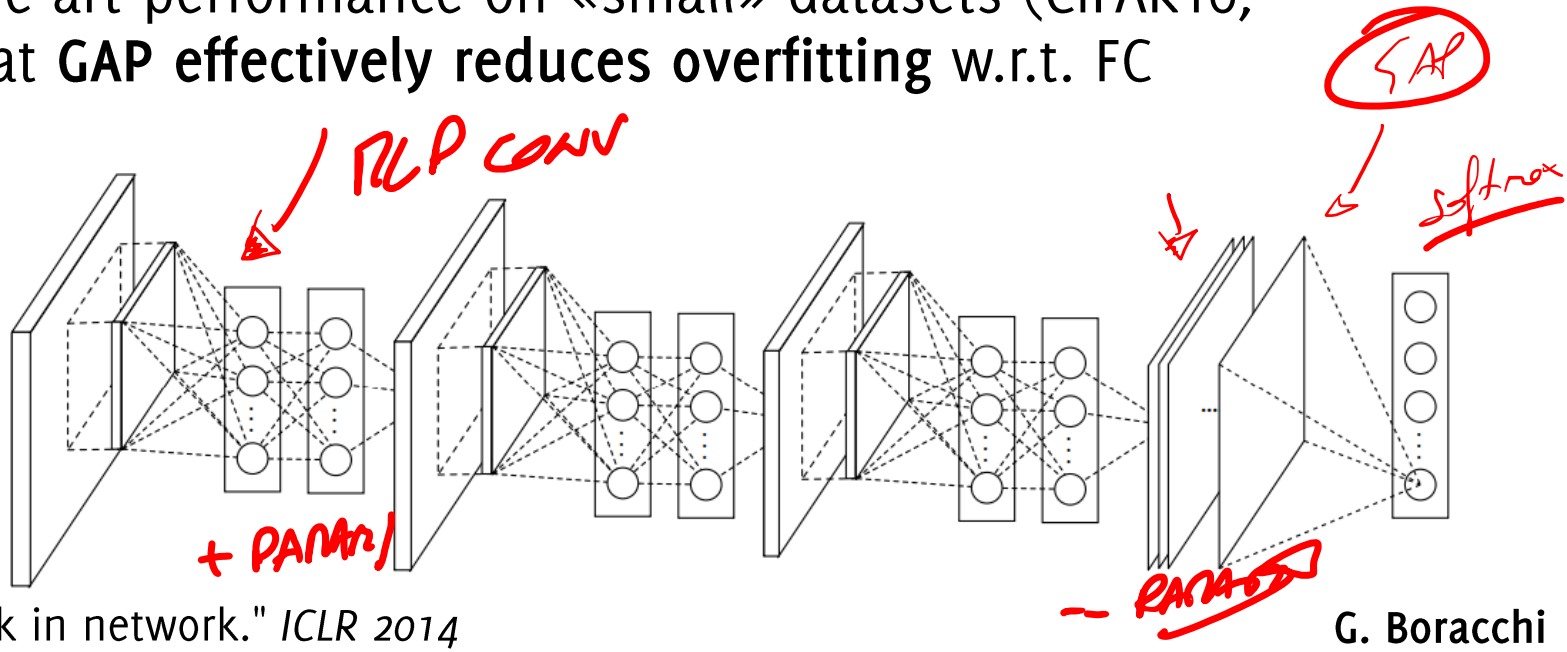
- mlpconv layers (RELU) + dropout
- Maxpooling

} A few layers of these

- Global Averaging Pooling (GAP) layer
- Softmax

} At the end of the network

simple NiNs achieve state-of-the-art performance on «small» datasets (CIFAR10, CIFAR100, SVHN, MNIST) and that **GAP effectively reduces overfitting** w.r.t. FC

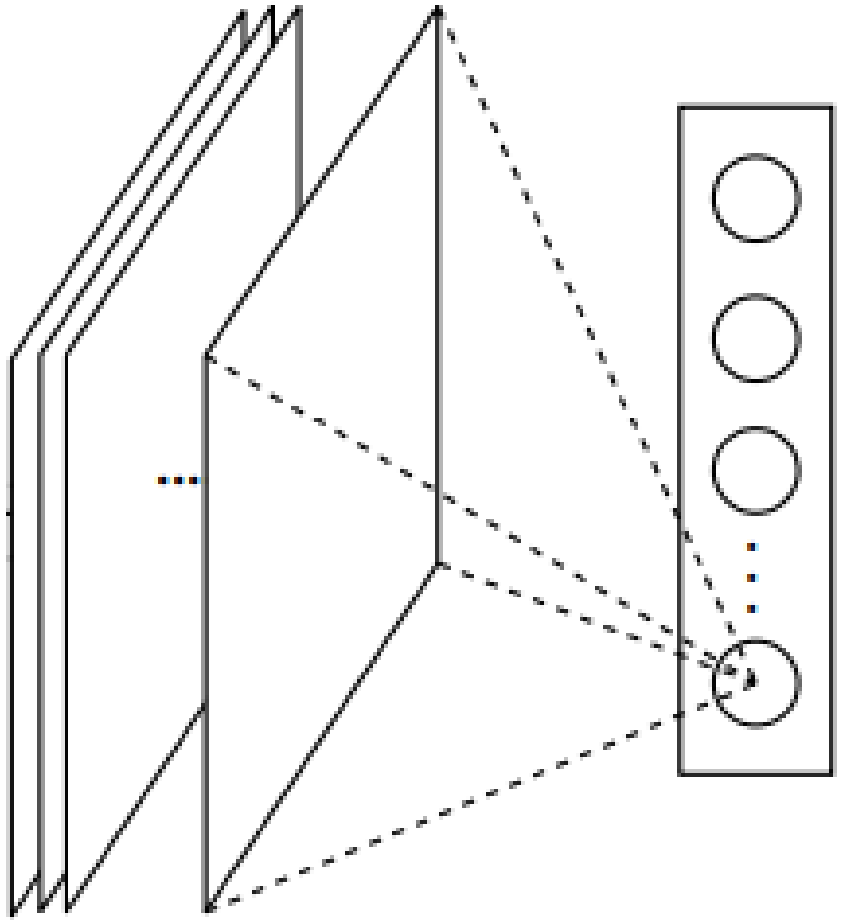



Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." *ICLR 2014*

# The Global Averaging Pooling (GAP) Layer

We indeed see that GAP is acting as a (structural) regularizer

| Method | Testing Error |
|---|---|
| mlpconv + Fully Connected | 11.59% |
| mlpconv + Fully Connected + Dropout | 10.88% |
| mlpconv + Global Average Pooling | 10.41% |

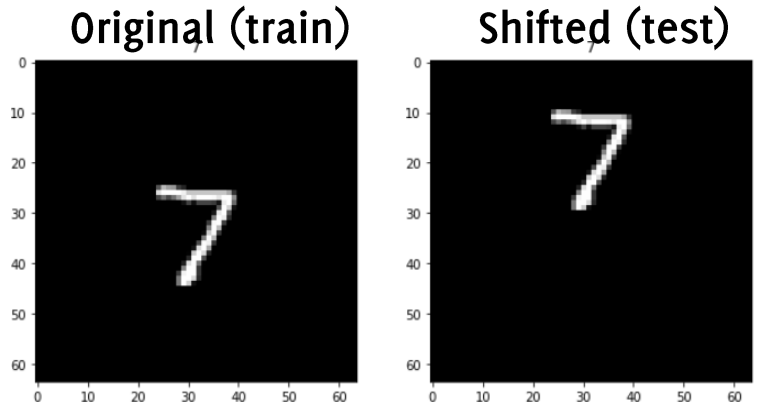


Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." *arXiv preprint arXiv:1312.4400v3* (2014).

# GAP Increases Invariance to Shifts

**Features extracted by the convolutional part of the network are invariant to shift of the input image**

The MLP after the flattening is not invariant to shifts (different input neurons are connected by different weights)




Therefore, a CNN trained on centered images might not be able to correctly classify shifted ones

The GAP solves this problem, since there is no GAP and the two images lead to the same or very similar features

# GAP Increases Invariance to Shifts

Example:

**Dataset:** a 64x64 (zero padded) MNIST

**CNN-flattening:** a traditional CNN with flattening, trained

**CNN-GAP:** the same architecture CNN but with GAP instead of MLP

Train both CNNs over the same training set without shift

Test both CNNs over both

- Original test set
- Sifted test set

# CNN-flattening Architecture

| Layer (type) | Output Shape | Param # |
|---|---|---|
| Input (InputLayer) | [(None, 64, 64)] | 0 |
| reshape_1 (Reshape) | (None, 64, 64, 1) | 0 |
| conv1 (Conv2D) | (None, 62, 62, 32) | 320 |
| pool1 (MaxPooling2D) | (None, 31, 31, 32) | 0 |
| conv2 (Conv2D) | (None, 29, 29, 64) | 18496 |
| pool2 (MaxPooling2D) | (None, 14, 14, 64) | 0 |
| conv3 (Conv2D) | (None, 12, 12, 128) | 73856 |
| pool3 (MaxPooling2D) | (None, 6, 6, 128) | 0 |
| flatten (Flatten) | (None, 4608) | 0 |
| dropout1 (Dropout) | (None, 4608) | 0 |
| classifier (Dense) | (None, 64) | 294976 |
| dropout2 (Dropout) | (None, 64) | 0 |
| Output (Dense) | (None, 10) | 650 |

Total params: 388,298
Trainable params: 388,298
Non-trainable params: 0

Credits Eugenio Lomurno

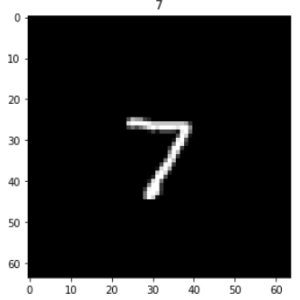
# CNN-GAP Architecture

| Layer (type) | Output Shape | Param # |
|---|---|---|
| Input (InputLayer) | [(None, 64, 64)] | 0 |
| reshape_3 (Reshape) | (None, 64, 64, 1) | 0 |
| conv1 (Conv2D) | (None, 62, 62, 32) | 320 |
| pool1 (MaxPooling2D) | (None, 31, 31, 32) | 0 |
| conv2 (Conv2D) | (None, 29, 29, 64) | 18496 |
| pool2 (MaxPooling2D) | (None, 14, 14, 64) | 0 |
| conv3 (Conv2D) | (None, 12, 12, 128) | 73856 |
| gpooling (GlobalAveragePooli | (None, 128) | 0 |
| dropout1 (Dropout) | (None, 128) | 0 |
| classifier (Dense) | (None, 64) | 8256 |
| dropout2 (Dropout) | (None, 64) | 0 |
| Output (Dense) | (None, 10) | 650 |

Total params: 101,578
Trainable params: 101,578
Non-trainable params: 0

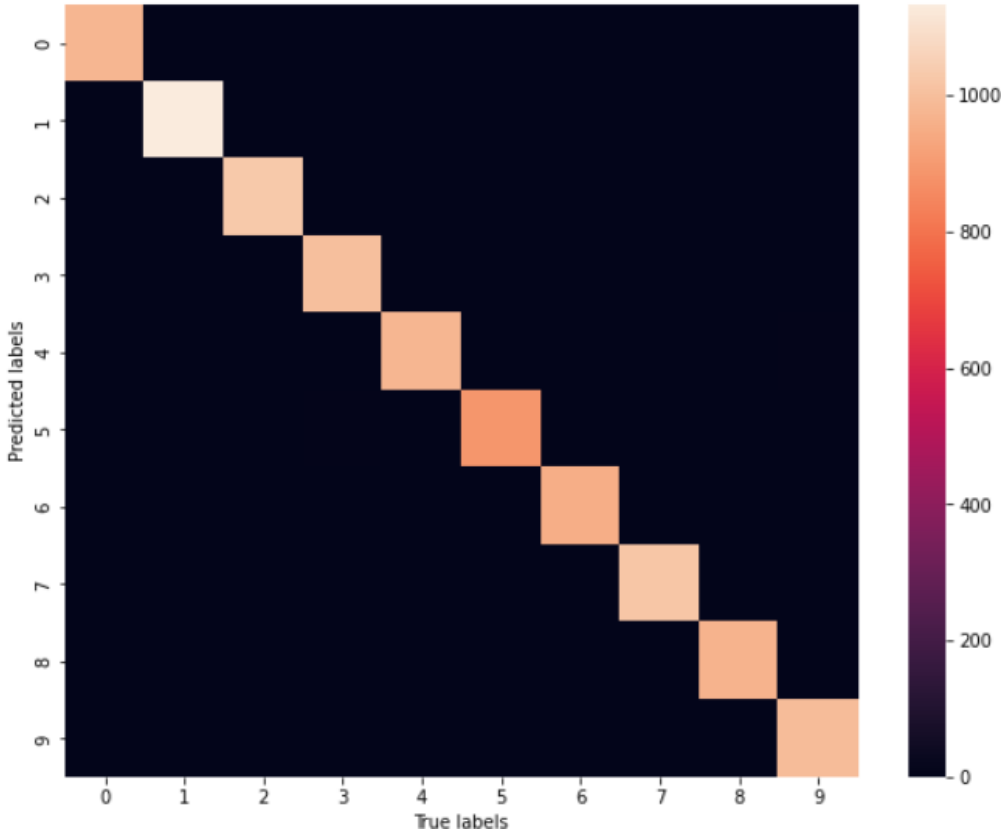
Credits Eugenio Lomurno

# Accuracy over Original Test Set



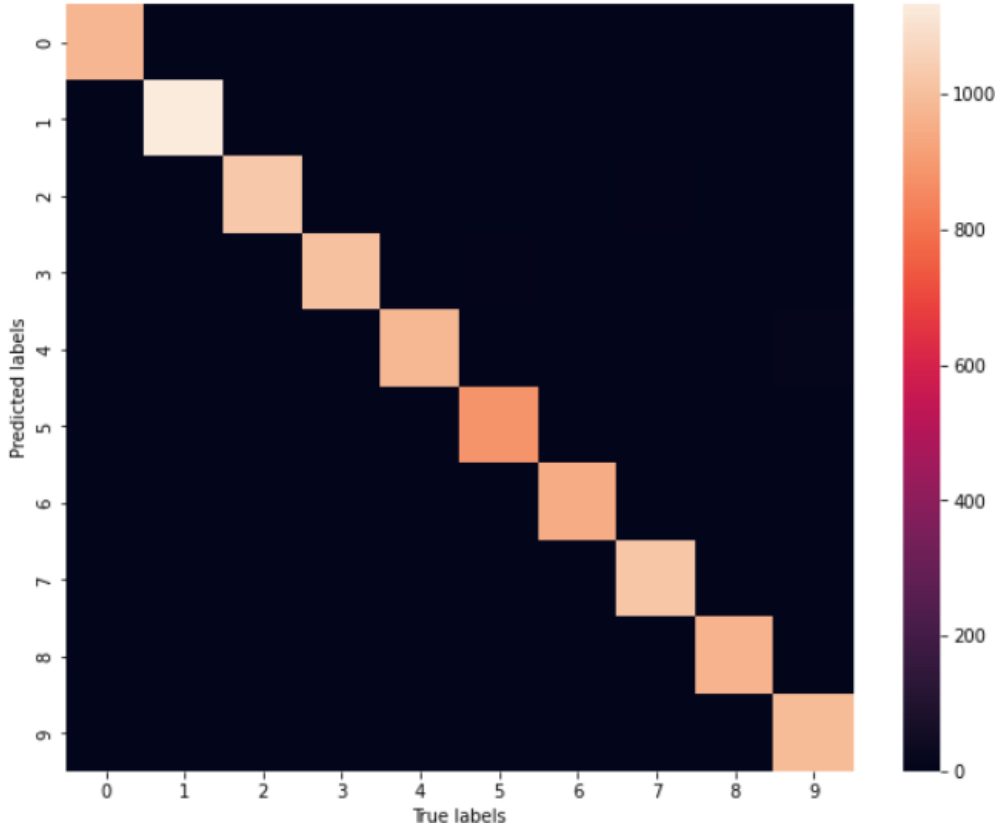

## CNN-flattening:

Accuracy: 0.9936
Precision: 0.9935
Recall: 0.9936
F1: 0.9935




## CNN-GAP

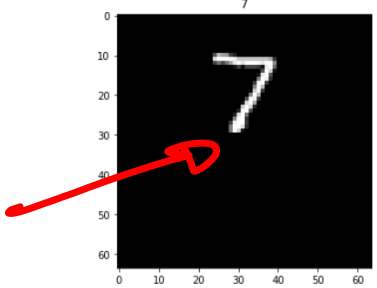
Accuracy: 0.9934
Precision: 0.9934
Recall: 0.9933
F1: 0.9933




Credits Eugenio Lomurno

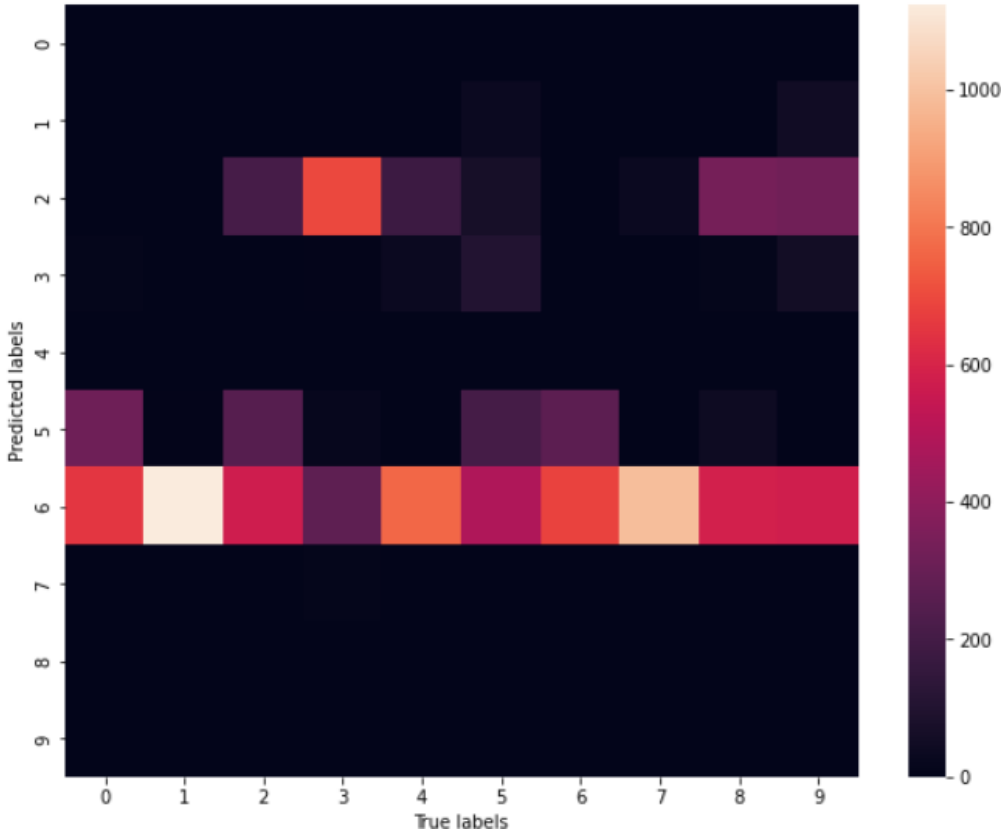
G. Boracchi

# Accuracy over Shifted Test Set



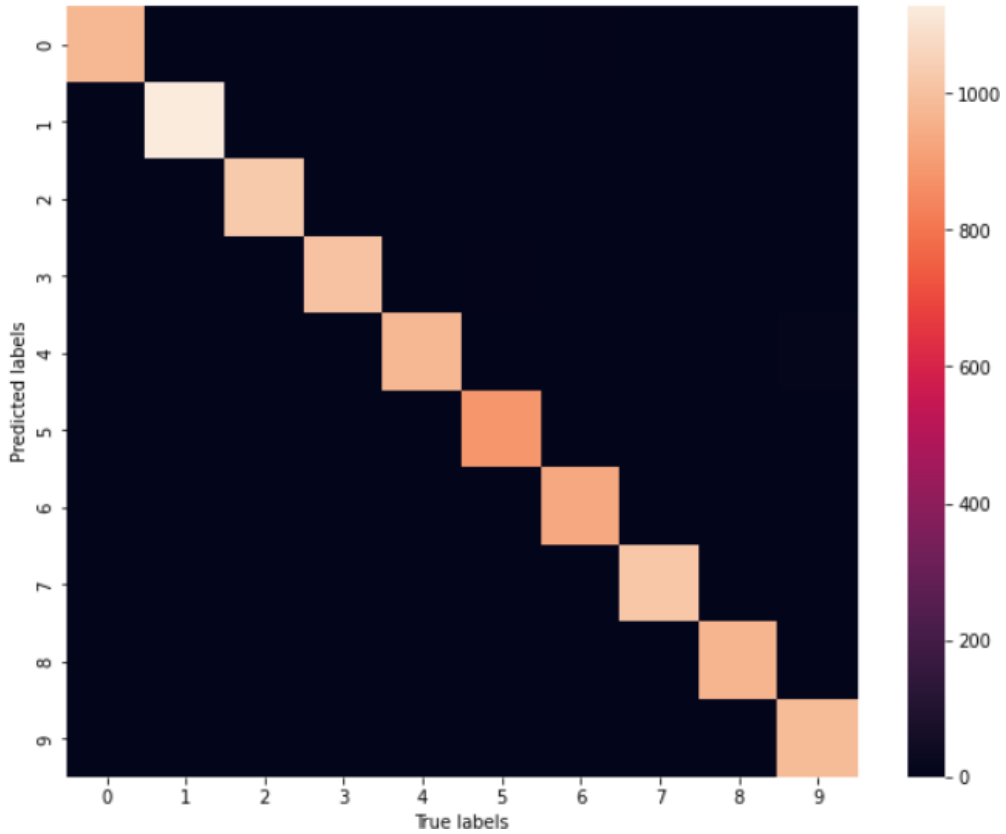

**CNN-flattening:**

Accuracy: 0.1103
Precision: 0.0435
Recall: 0.1151
F1: 0.0537




**CNN-GAP**

Accuracy: 0.9906
Precision: 0.9906
Recall: 0.9905
F1: 0.9905




Credits Eugenio Lomurno

G. Boracchi

https://colab.research.google.com/drive/1s108-oylignuFBNTscfj9ZgDymi5sU0X?usp=sharing

# Localization

# The Localization Task

The input image contains a single relevant object to be classified in a fixed set of categories

The task is to:

1) assign the object class to the image

hawk

# The Localization Task

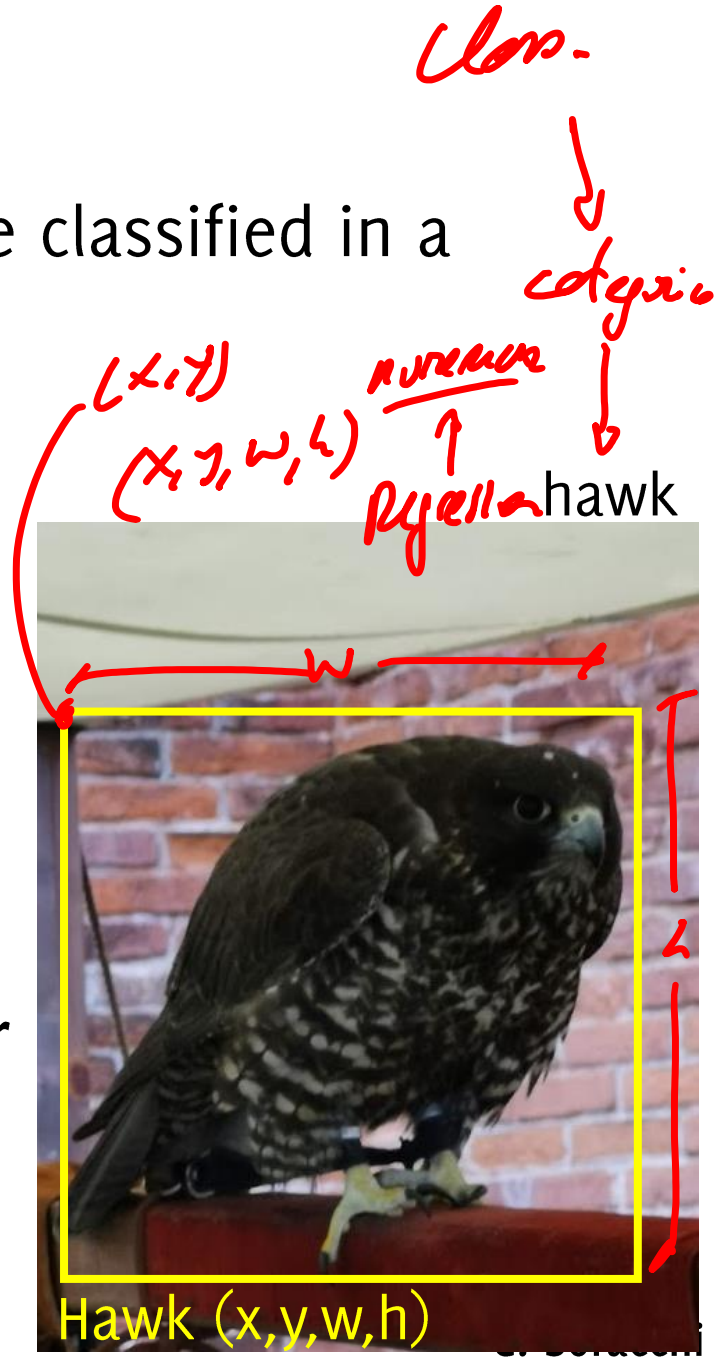
The input image contains a single relevant object to be classified in a fixed set of categories

The task is to:

1) assign the object class to the image
2) locate the object in the image by its bounding box

A training set of annotated images with **label and a bounding box** around each object is required

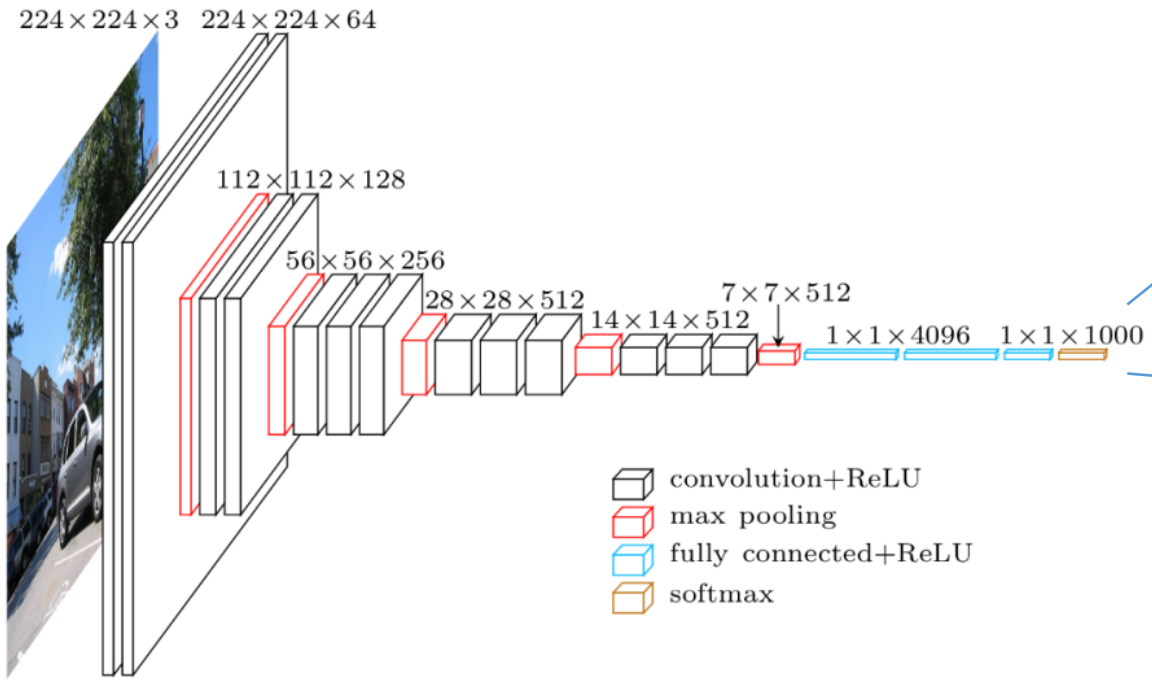
**Extended localization problems involve regression over more complicated geometries (e.g. human skeleton)**



# The Simplest Solution

Train a network to predict both the class label and the bounding box

"Two heads"




Target class

Soft-max loss $\mathcal{S}$

Categorical cross entropy loss

Bounding Box Coordinates

$(x, y, w, h)$

Regression loss $\mathcal{R}$, e.g. the $\ell^2, \ell^1, ..$ bounding box.

$$\|[\hat{x}, \hat{y}, \hat{w}, \hat{h}] - [x, y, w, h]\|_2$$

ESTIMATED one

TRUE B.B location

G. Boracchi

# The Simplest Solution

**The training loss has to be a single scalar** since **we compute gradient of a scalar function** with respect to network parameters.

**Minimize a multitask loss to merge two losses:**

$$\mathcal{L}(x) = \alpha\, \mathcal{S}(x) + (1 - \alpha)\mathcal{R}(x)$$

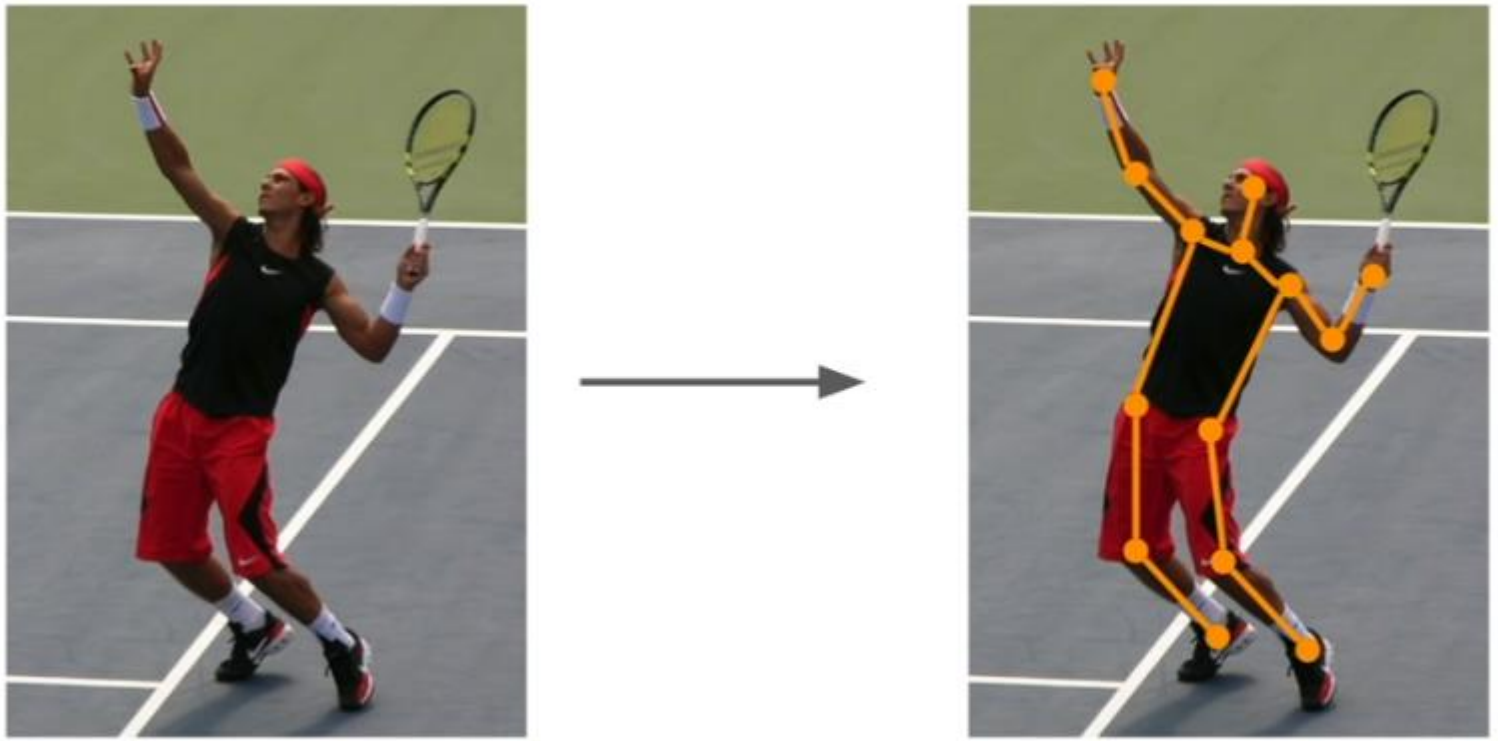
and $\alpha$ is an hyper parameter of the network.

Watch out that $\boldsymbol{\alpha}$ **directly influences the loss definition**, tuning might be difficult. Better to do cross-validation looking at some other loss (loss value for different values of $\alpha$ might be meaningless).

It is also possible to **adopt a pre-trained model** and then train the two FC separately… however it is always better to perform at least some fine tuning to train the two jointly.

# Extension to Human Pose Estimation

Pose estimation is formulated as a **CNN-regression problem towards body joints**. It is possible to **address localization only here**





Represent pose as a set of 14 joint positions:

- Left / right foot
- Left / right knee
- Left / right hip
- Left / right shoulder
- Left / right elbow
- Left / right hand
- Neck
- Head top

6 x 2

2

14 locations

28 output neurons

Alexander Toshev and Christian Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks", CVPR 2014

# Extension to Human Pose Estimation

Pose estimation is formulated as a **CNN-regression problem towards body joints.**

- **The network receives as input the whole image, capturing the full-context of each body joints.**
- The approach is **very simple to design and train.** Training problems can be **alleviated by transfer learning** of existing classification networks

**Pose is defined as a vector of $k$ joints location** for the human body, possibly normalized w.r.t. the bounding box enclosing the human.

Train a CNN to predict **a $2k$ vector as output** by using an Alexnet-like architecture

Alexander Toshev and Christian Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks", CVPR 2014

# Training Human Pose Estimation Networks

Adopt a $\ell^2$ regression loss of the estimated pose parameters over the annotations.

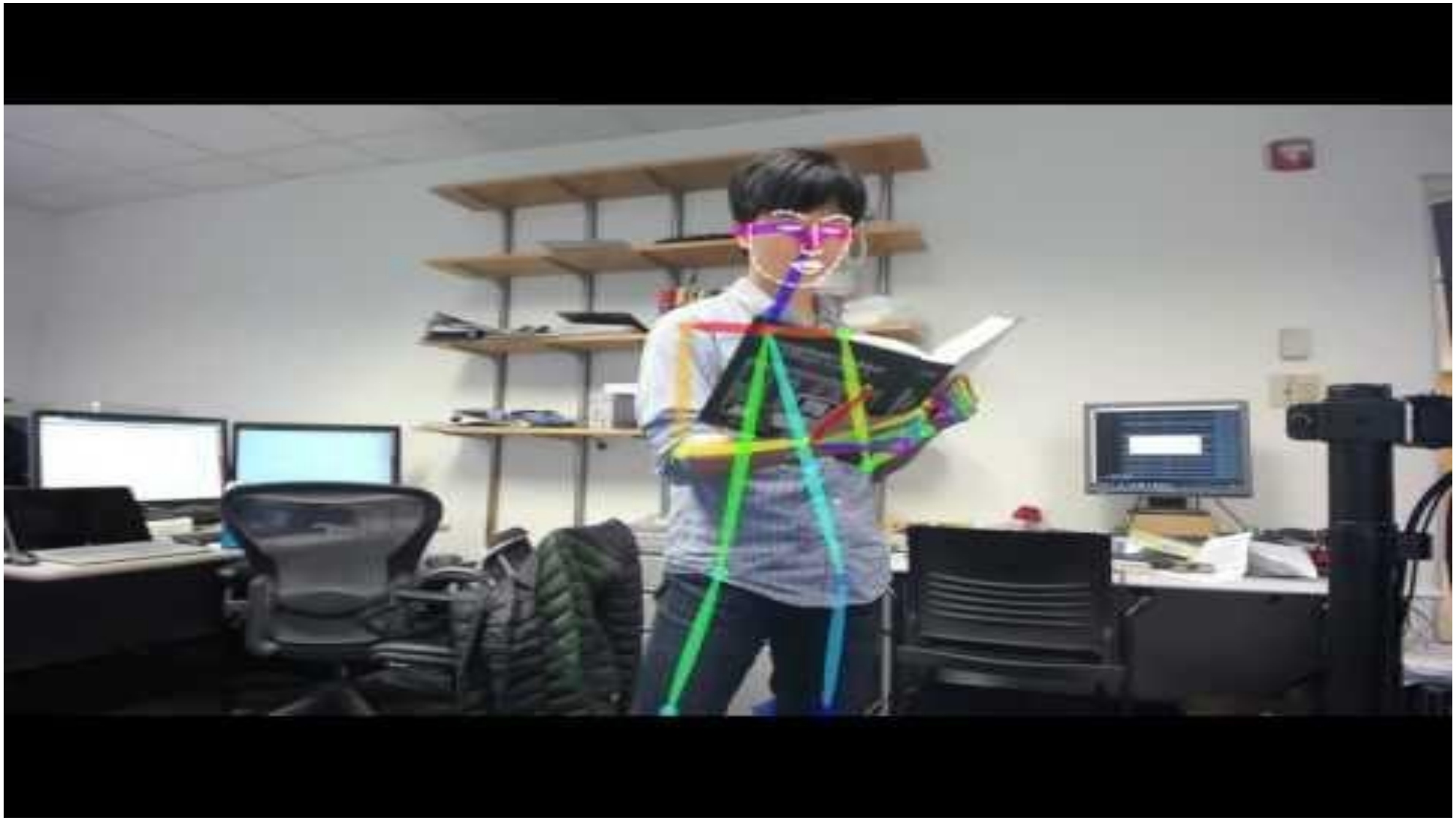
- This can be also defined when a few joints are not visible.

Reduce overfitting by augmentation (translation and flips).

Multiple networks have been trained to improve localization by refining joint position in a crop around the initial detection.

Alexander Toshev and Christian Szegedy, “DeepPose: Human Pose Estimation via Deep Neural Networks”, CVPR 2014

# Open Pose



Cao, Z et al.. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. CVPR 2017

# Weakly-Supervised Localization

… Global Averaging Pooling Revisited

… visualizing what matters most for CNN predicitons

G. Boracchi

# Weakly supervised localization

Perform localization over an image without images with annotated bounding box

- Training set provided as for classification with image-label pairs $\{(I, \ell)\}$ where no localization information is provided



2016

# Learning Deep Features for Discriminative Localization

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, Antonio Torralba
Computer Science and Artificial Intelligence Laboratory, MIT
{bzhou,khosla,agata,oliva,torralba}@csail.mit.edu

G. Boracchi

# The GAP revisited

The advantages of GAP layer **extend beyond** simply acting as a structural **regularizer** that prevents overfitting

In fact, **CNNs can retain a remarkable localization ability** until the final layer. By a simple tweak it is possible to easily identify the discriminative image regions leading to a prediction.

*A **CNN trained on object categorization** is successfully able to **localize the discriminative regions for action classification** as the objects that the humans are interacting with rather than the humans themselves*

Zhou, Bolei, et al. "Learning deep features for discriminative localization.“ CVPR 2016.

# Class Activation Mapping

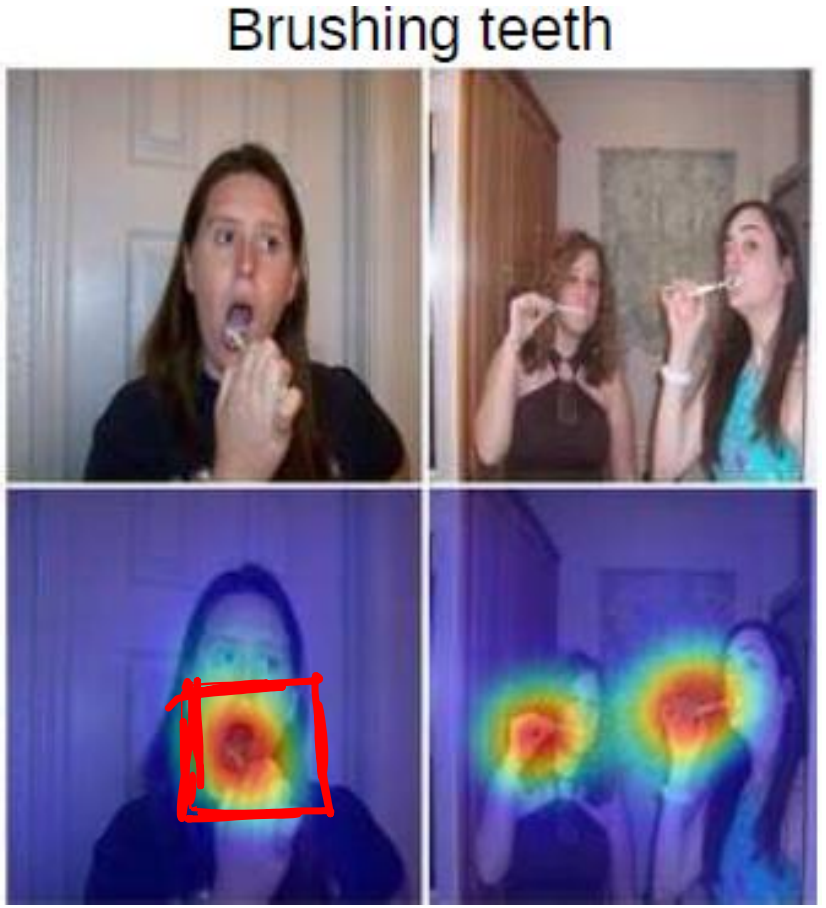

Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

# Class Activation Mapping (CAM)

**Identifying** exactly **which regions** of an image **are being used for discrimination**.

CAM are very easy to compute. It just requires:

- FC layer after the GAP
- a minor tweak



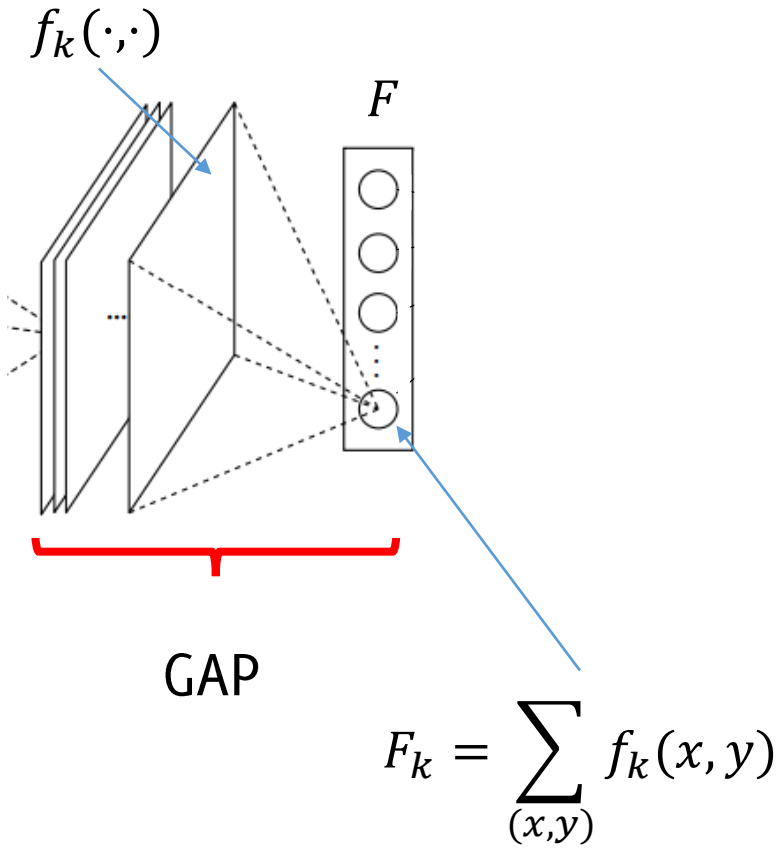

Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

# The Global Averaging Pooling (GAP) Layer

A very simple architecture made only of convolutions and activation functions leads to a final layer having:

- $n$ feature maps $f_k(\cdot,\cdot)$ having resolution "similar" to the input image
- a vector after GAP made of $n$ averages $F_k$

GAP + FC




Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

# The Global Averaging Pooling (GAP) Layer

Add (and train) a **single FC layer** after the GAP.

The FC computes $S_c$ for each class $c$ as the weighted sum of $\{F_k\}$, where weights are defined during training

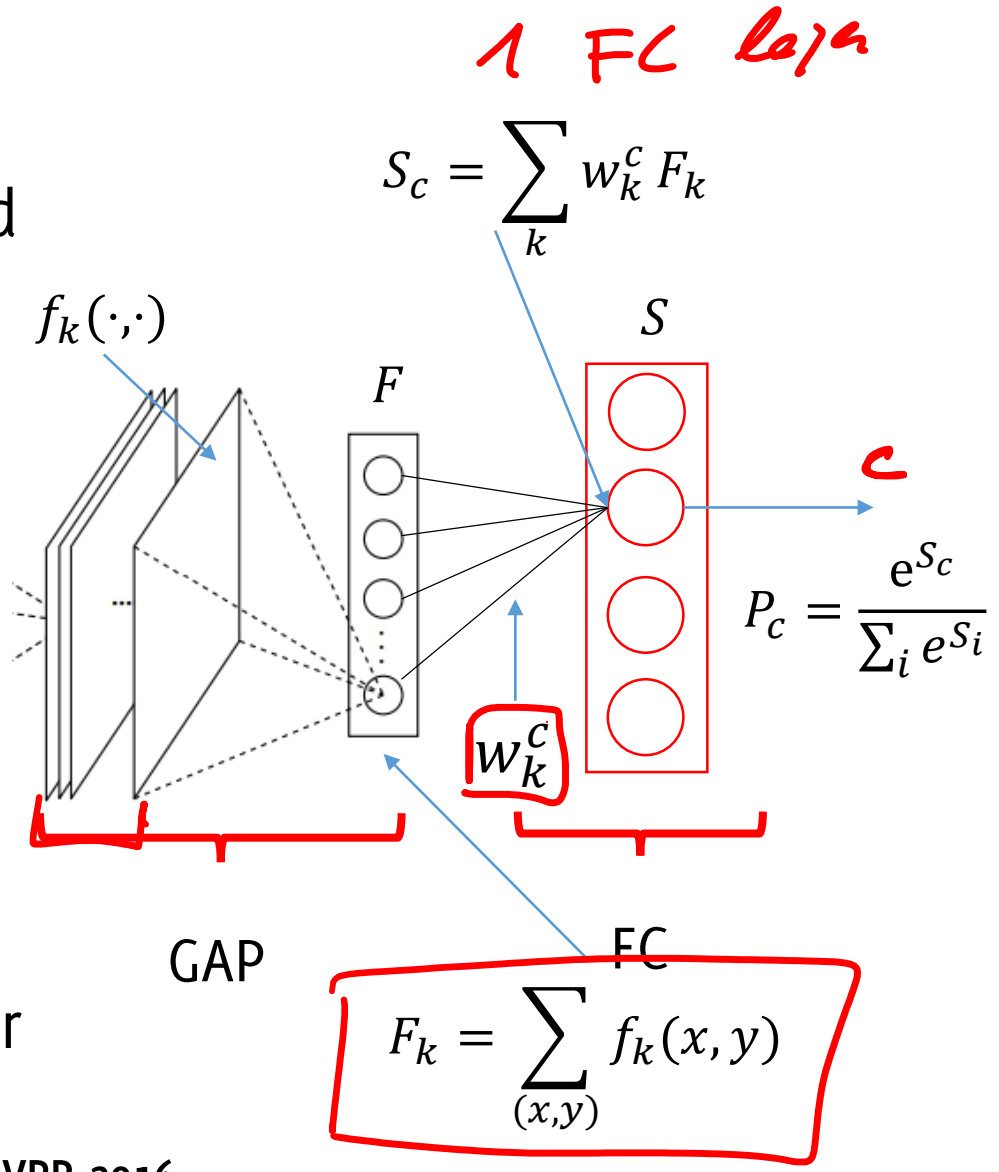
Then, the class probability $P_c$ via soft-max (class $c$)

**Remark:** when computing

$$S_c = \sum_k w_k^c F_k$$

$w_k^c$ encodes the importance of $F_k$ for the class $c$,

$\{w_k^c\}_{k,c}$ are all the parameters of the last FC layer




Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

# The Global Averaging Pooling (GAP) Layer

However

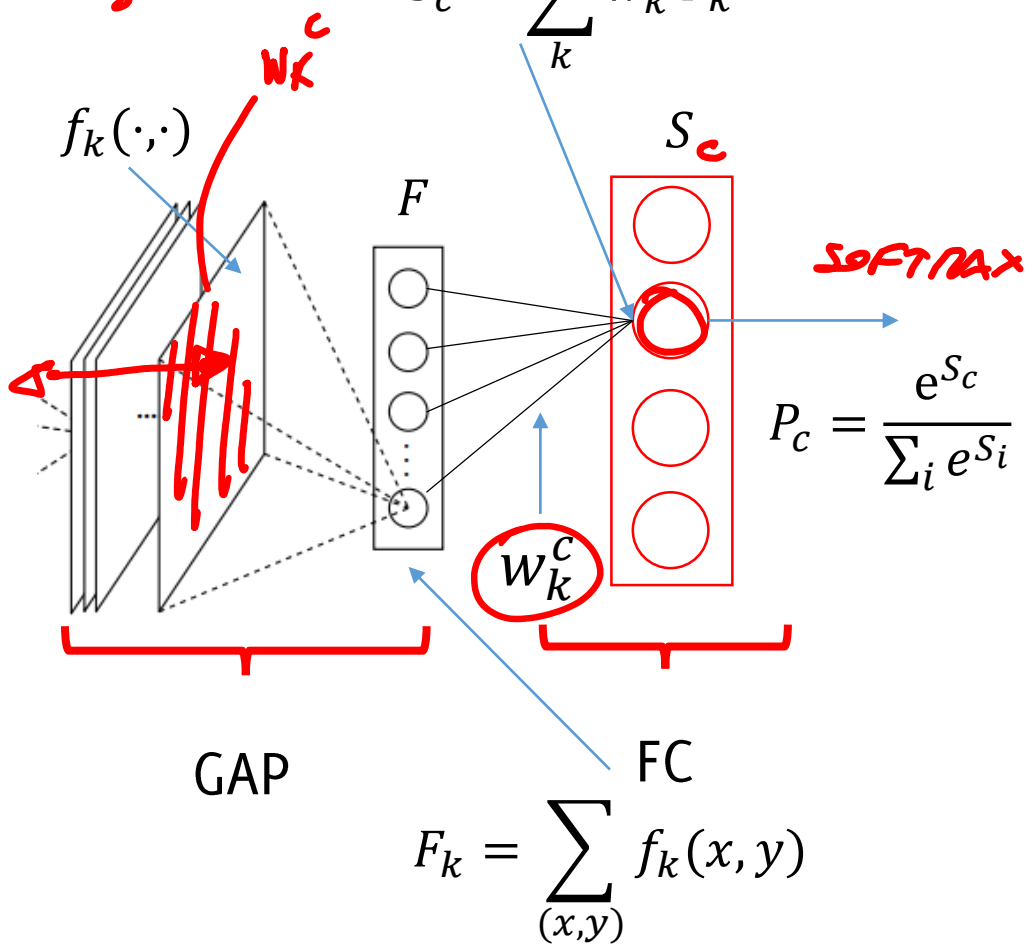
$$S_c = \sum_k w_k^c \sum_{x,y} f_k(x,y) = \sum_{x,y} \sum_k w_k^c \, f_k(x,y)$$

And CAM is defined as

$$M_c(x,y) = \sum_k w_k^c \, f_k(x,y)$$

where $M_c(x,y)$ directly indicates the importance of the activations at $(x,y)$ for predicting the class $c$

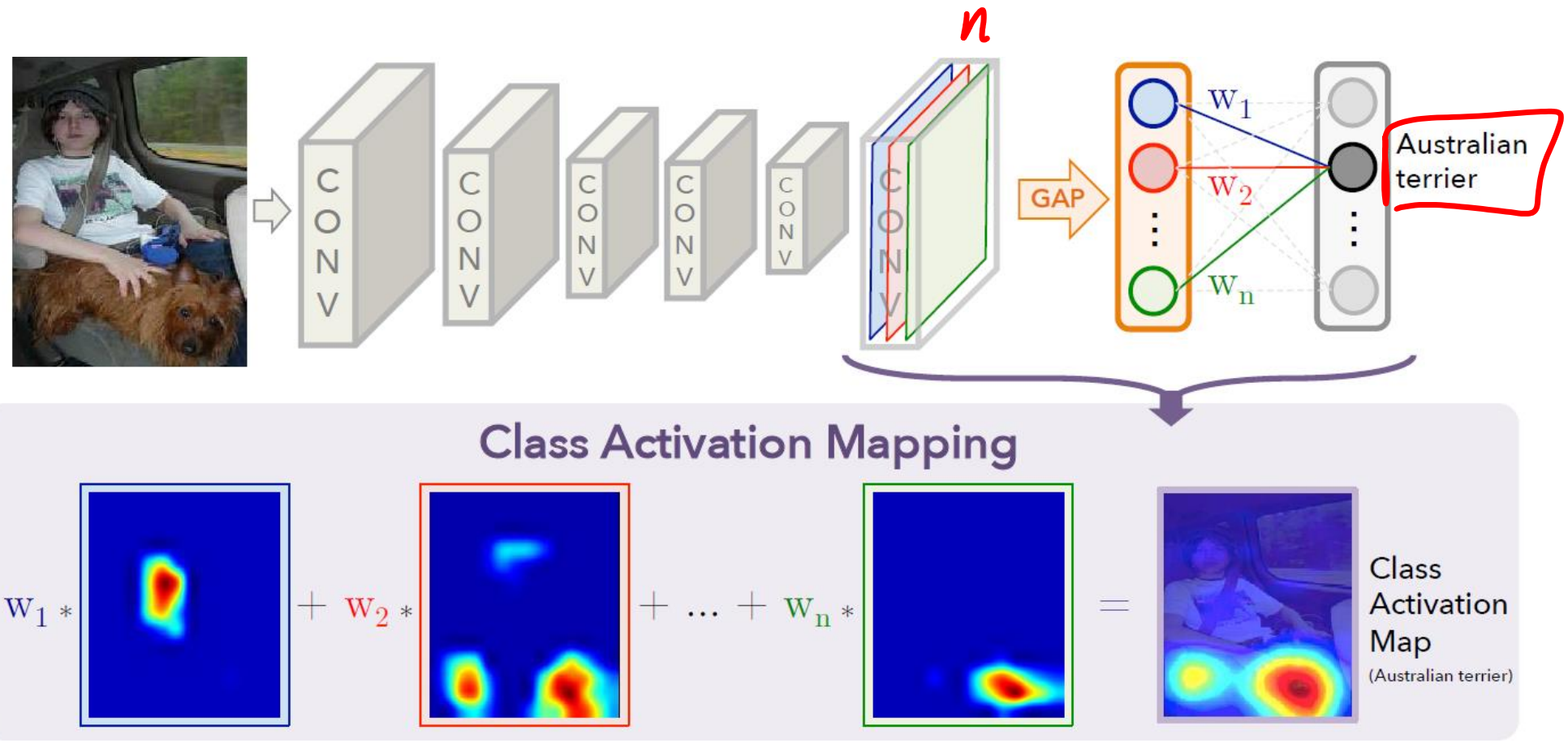
**Rmk:** unlike GAP, thanks to the softmax, **the depth of the last convolutional activations can differ from the number of classes**




Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

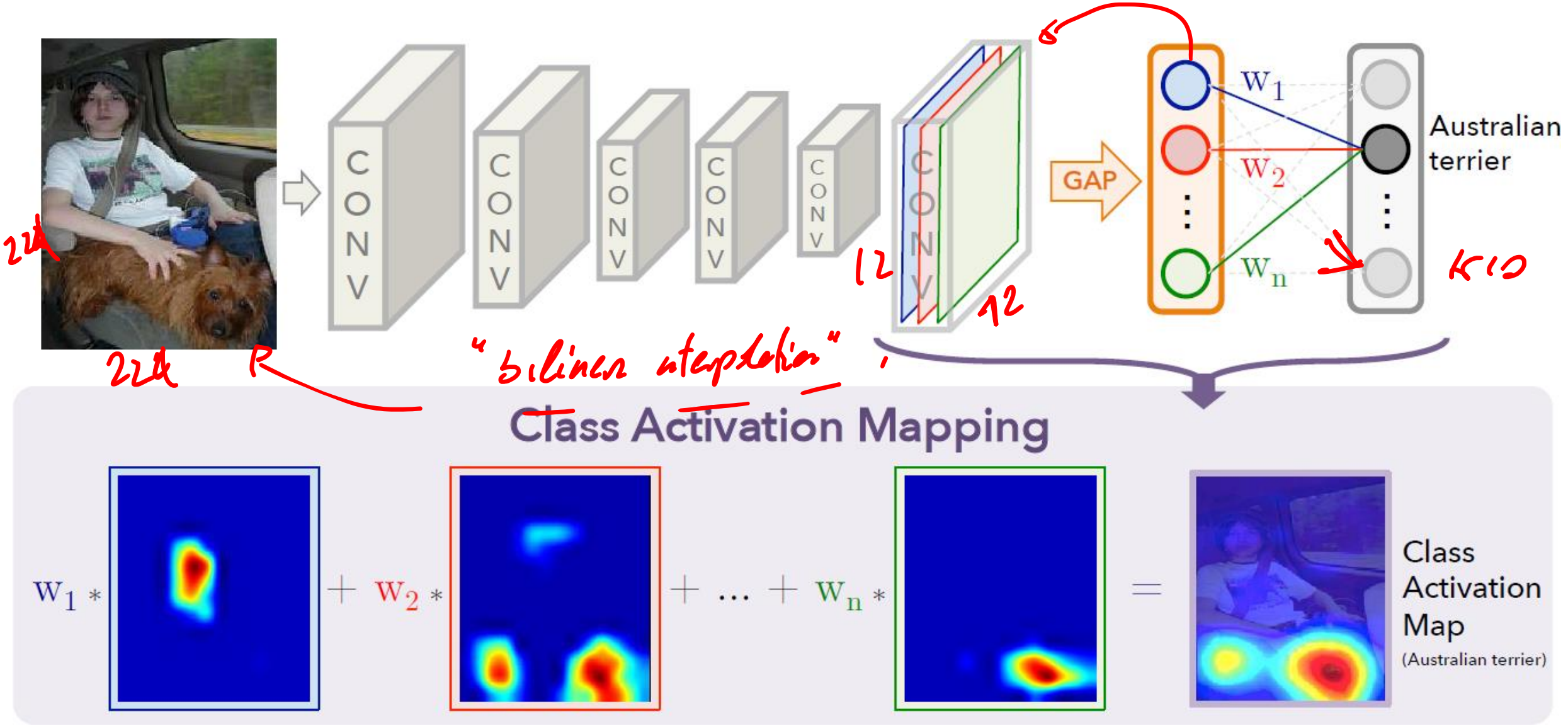
# Class Activation Mapping

Now, the weights represents the importance of each feature map to yield the final prediction. Upsampling might be necessary to match the input image



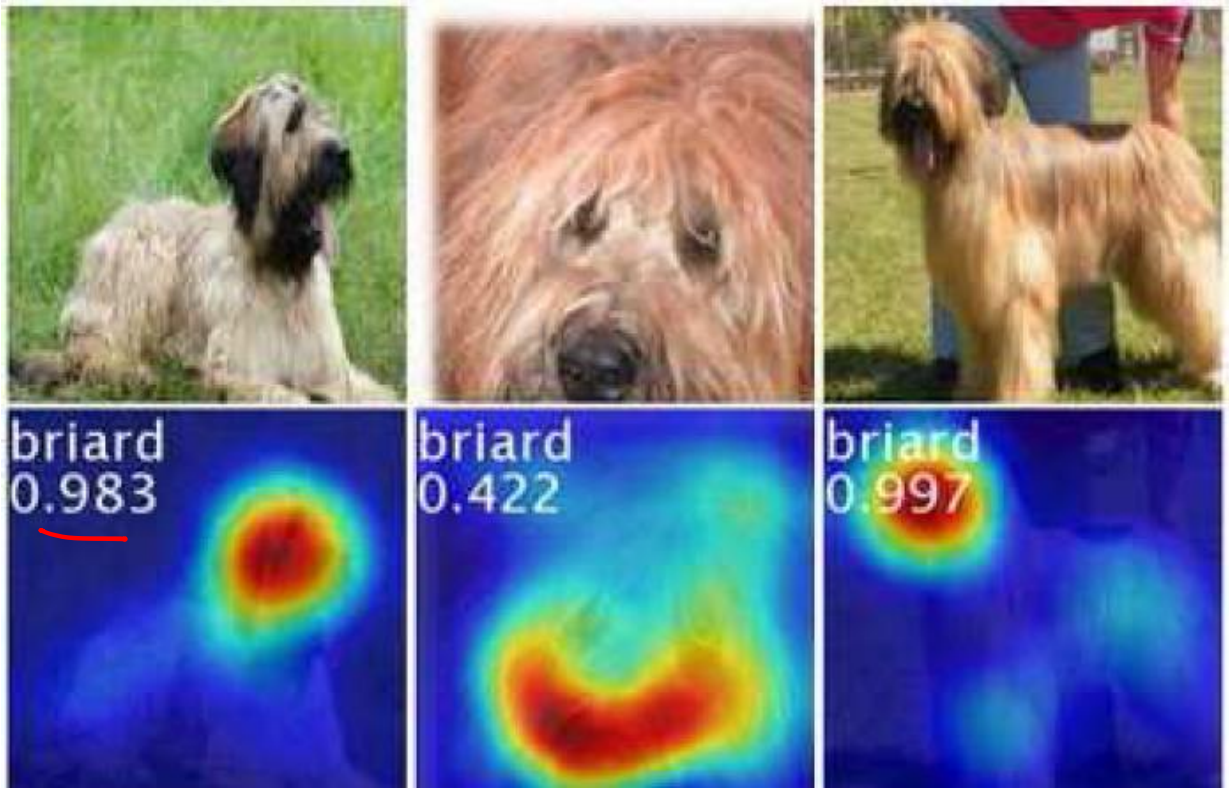

Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

# Class Activation Mapping




Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

# Class Activation Mapping





Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

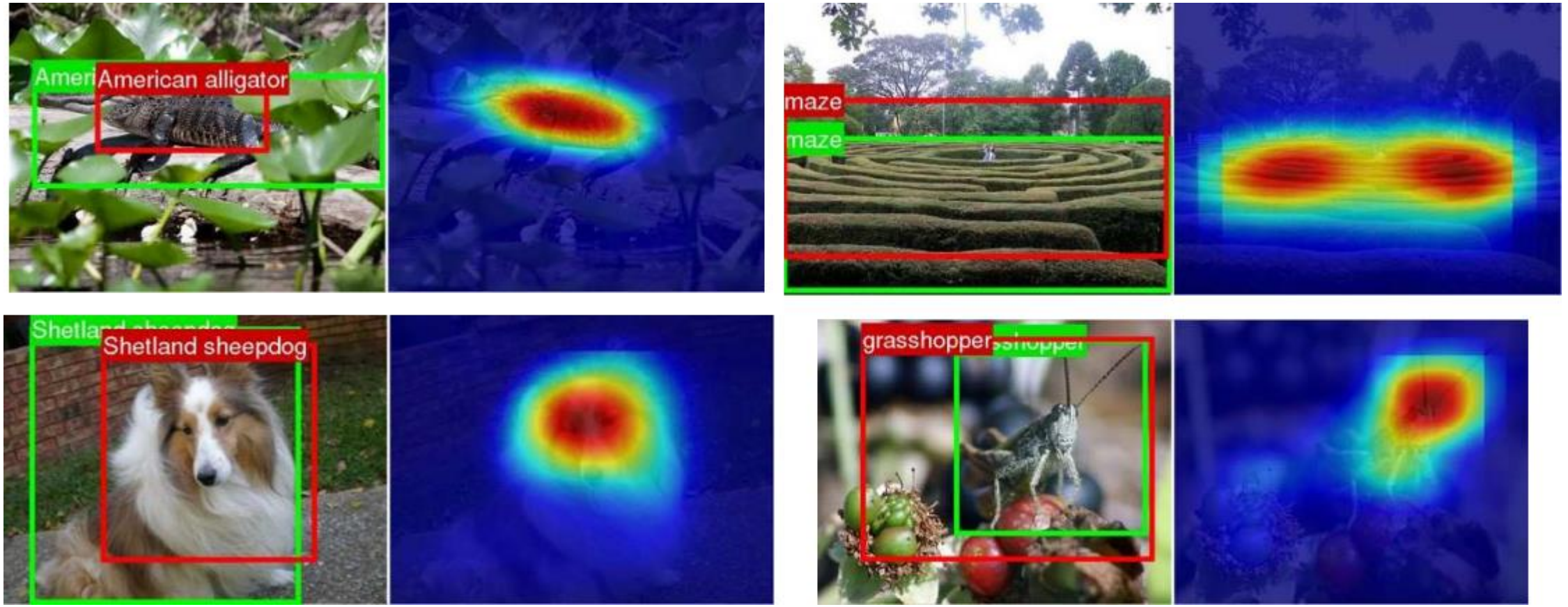
# Remarks

- CAM can be **included in any pre-trained network**, as long as all the FC layers at the end are removed
- The **FC used for CAM is simple**, few neurons and no hidden layers
- **Classification performance might drop** (in VGG removing FC means loosing 90% of parameters)
- **CAM resolution** (localization accuracy) can improve by «**anticipating**» GAP to larger convolutional feature maps (but this reduces the semantic information within these layers)
- **GAP**: encourages the **identification of the whole object**, as all the parts of the values in the activation map concurs to the classification
- **GMP (Global Max Pooling)**: it is enough to have a high maximum, thus **promotes specific discriminative features**

Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

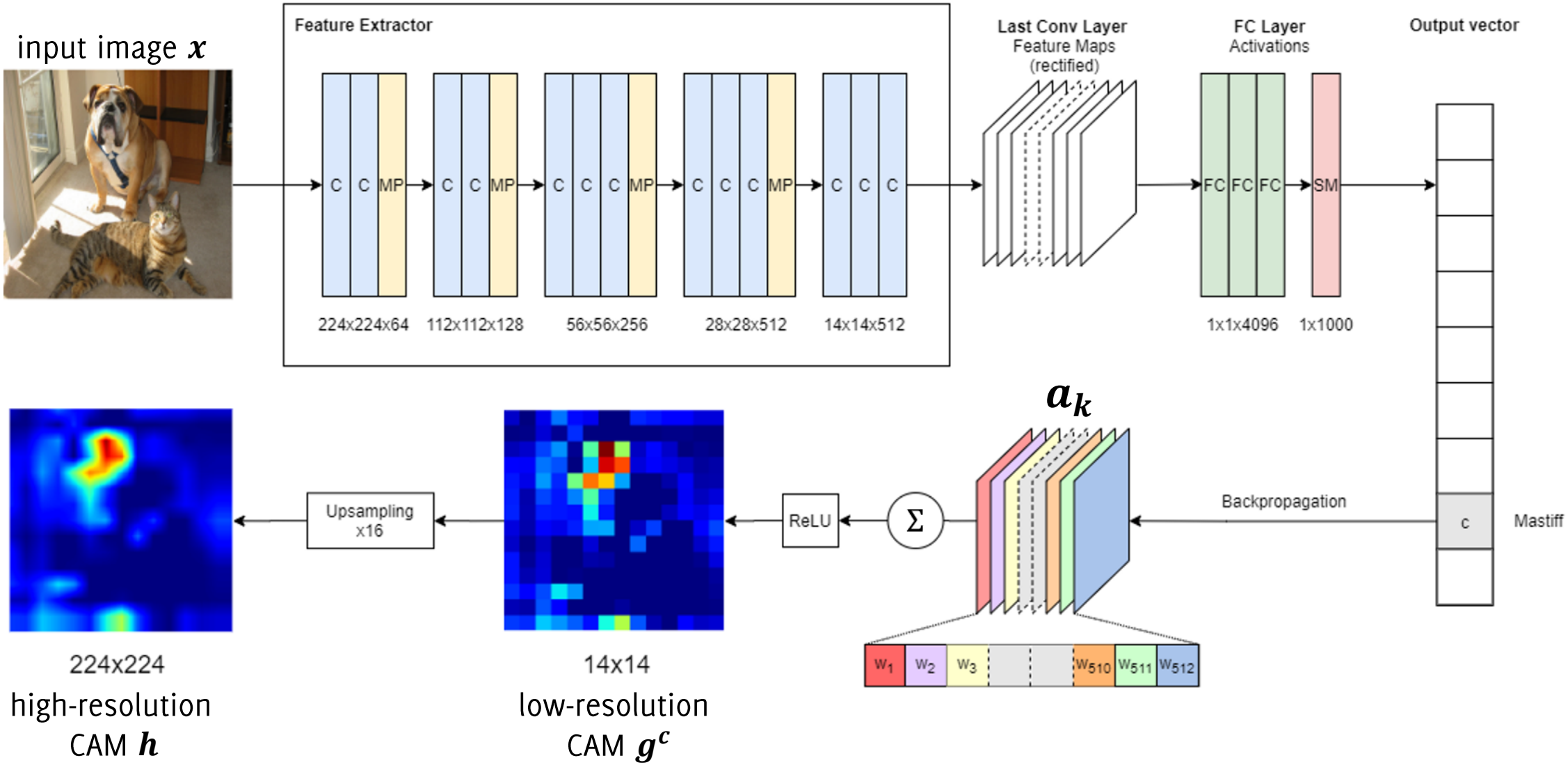
# Weakly Supervised Localization

Use thresholding CAM values: > 20% max(CAM), then take the largest componet of the thresholded map (green GT, red estimated location)



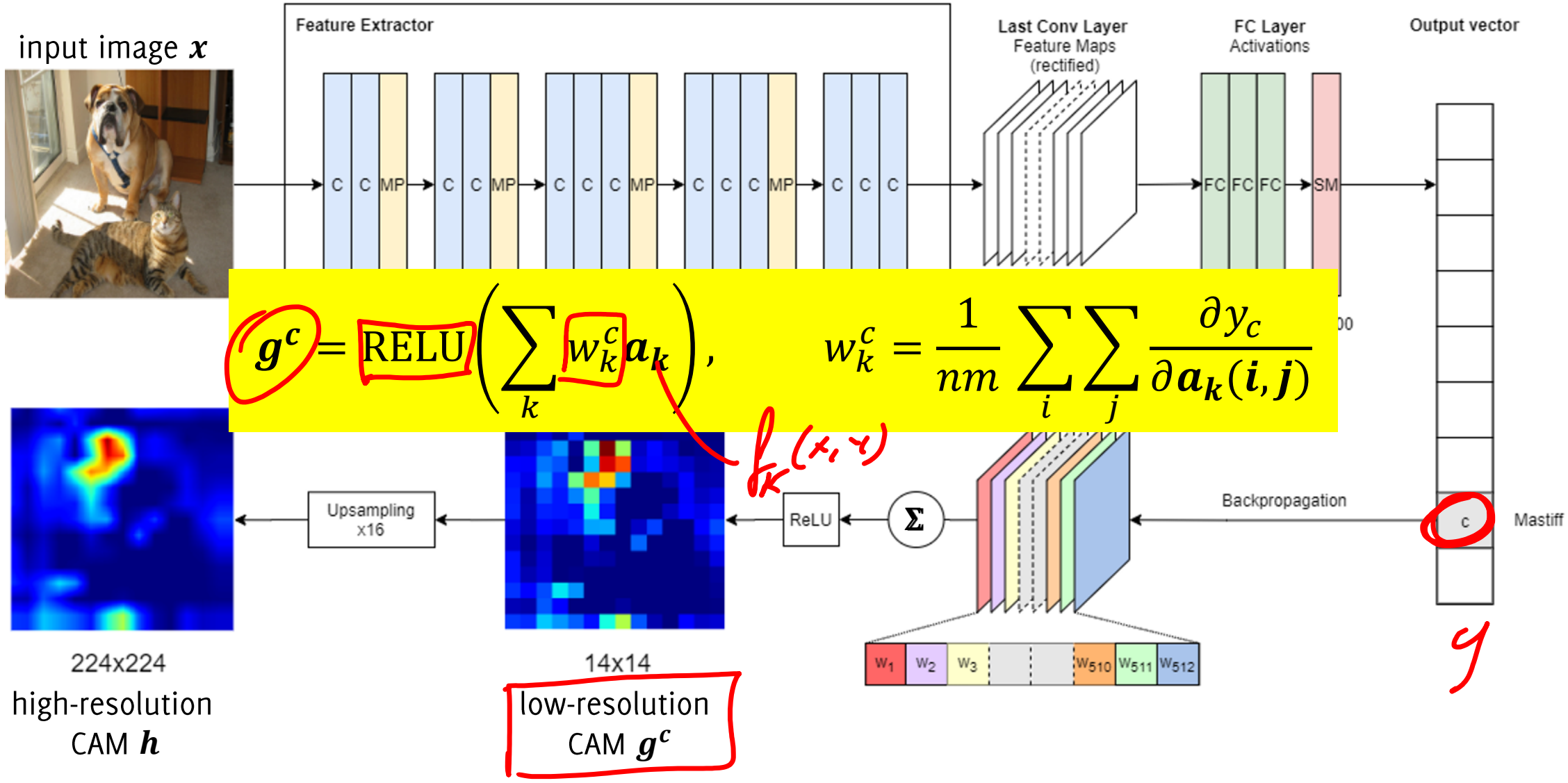

Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016

# Grad-CAM and CAM-based techniques



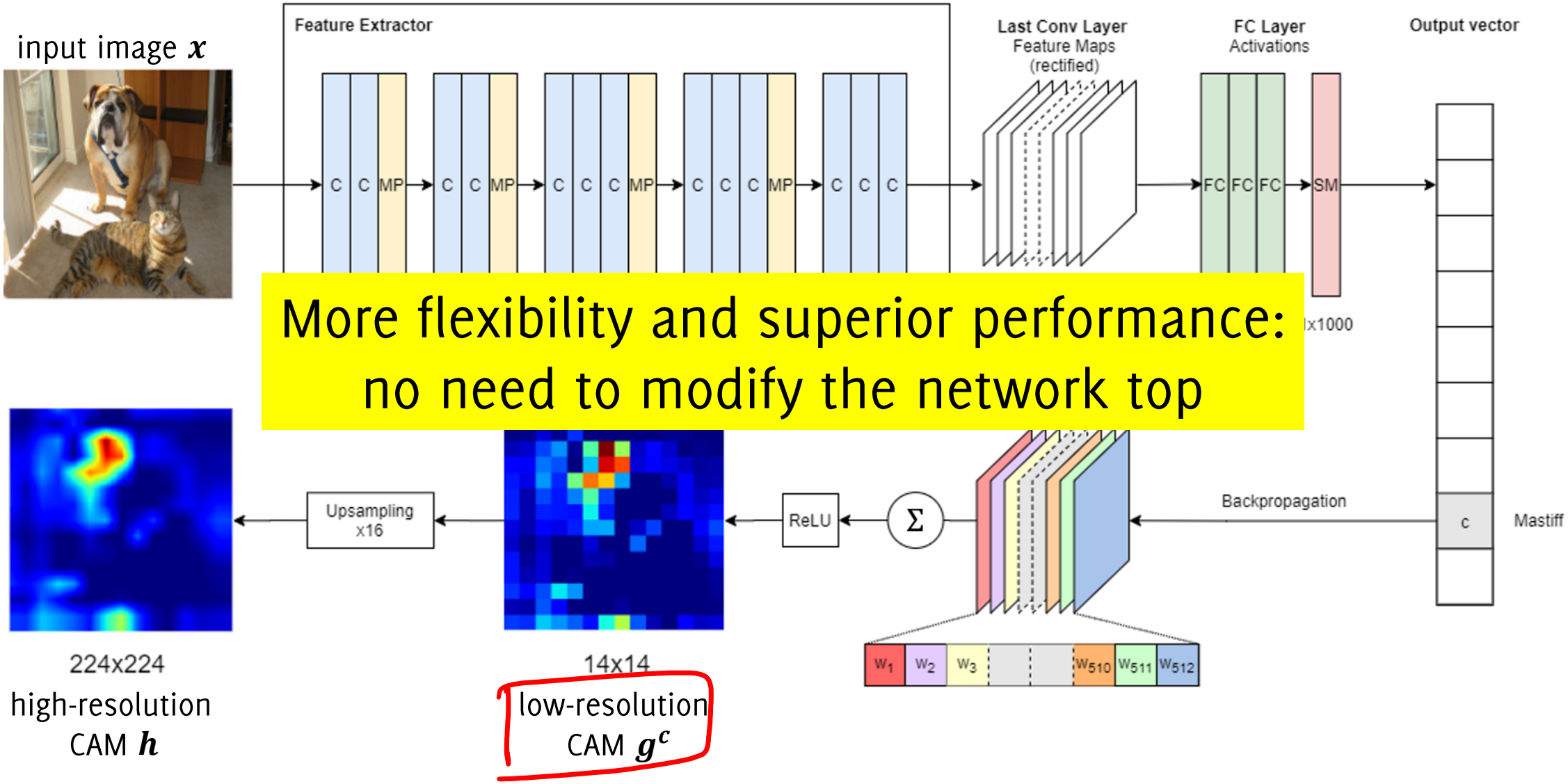

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV 2017*

# Grad-CAM and CAM-based techniques




Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. *ICCV2017*

# Grad-CAM and CAM-based techniques




More flexibility and superior performance:
no need to modify the network top

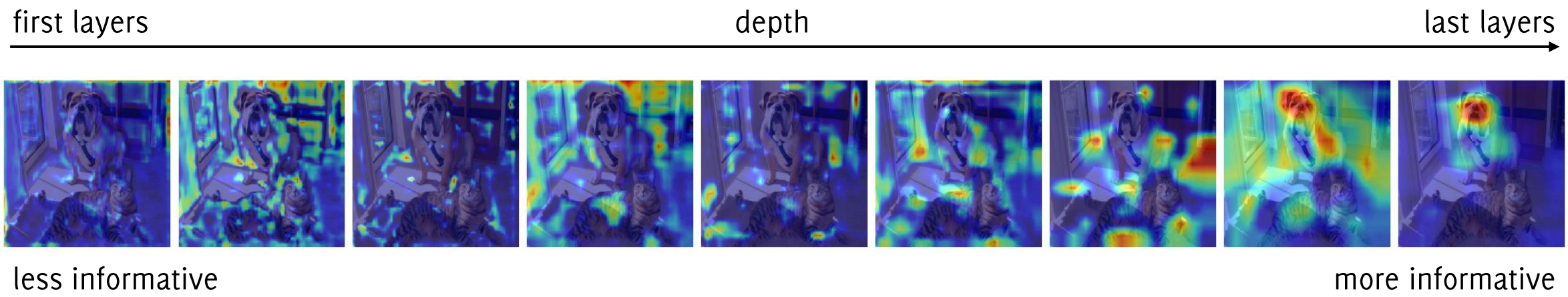
Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. *ICCV2017*

# Heatmaps Desiderata

Should be **class discriminative**

Should **capture fine-grained details** (high-resolution)

- This is critical in many applications (e.g. medical/industrial imaging)

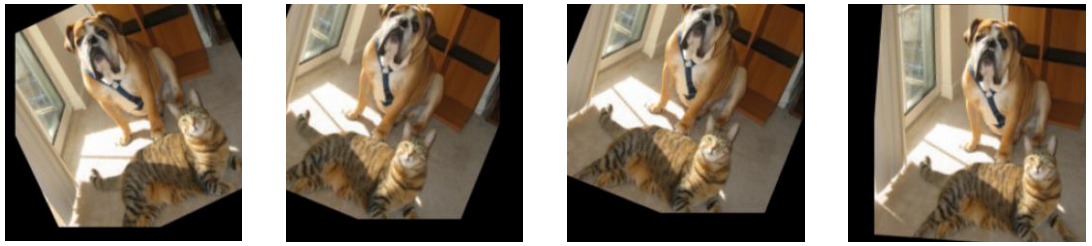


# Augmented Grad-CAM

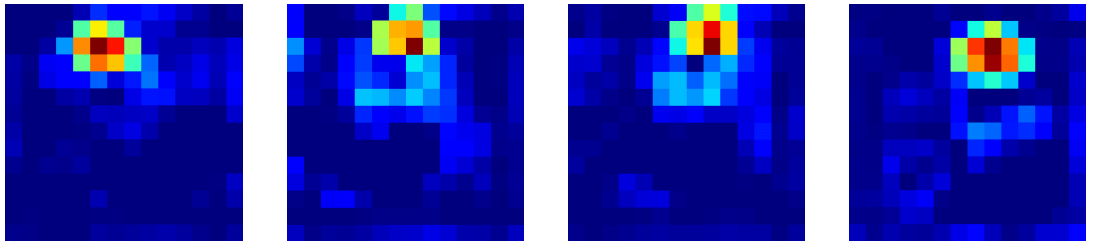
We consider the augmentation operator $\mathcal{A}_l: \mathbb{R}^{N \times M} \rightarrow \mathbb{R}^{N \times M}$, including random rotations and translations of the input image $\boldsymbol{x}$

Augmented Grad-CAM: increase heat-maps resolution through image augmentation

All the responses that the CNN generates to the **multiple augmented versions of the same input image** are very informative for reconstructing the high-resolution heat-map $\boldsymbol{h}$



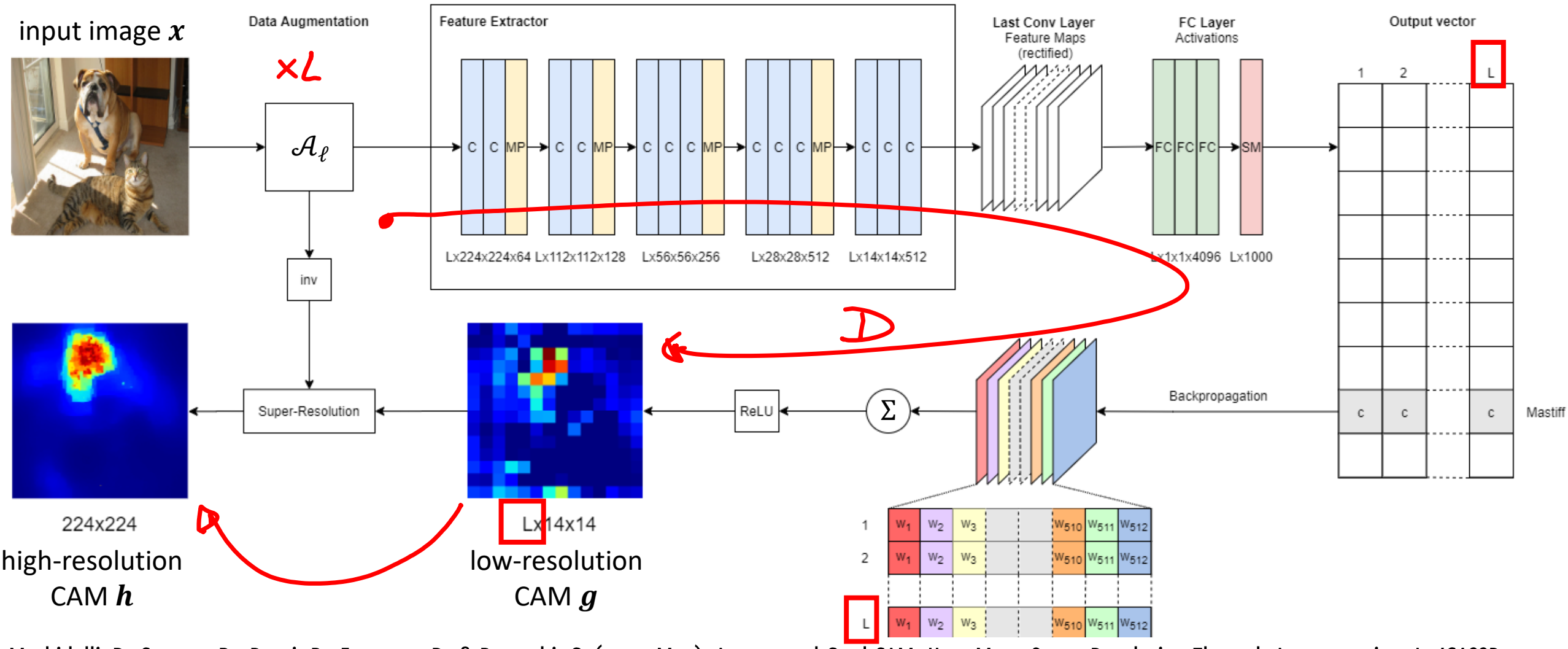
$x_1 = \mathcal{A}_1(x)$ $x_2 = \mathcal{A}_2(x)$ $x_3 = \mathcal{A}_3(x)$ $x_4 = \mathcal{A}_4(x)$



$g_1$ $g_2$ $g_3$ $g_4$

Morbidelli, P., Carrera, D., Rossi, B., Fragneto, P., & Boracchi, G. (2020, May). Augmented Grad-CAM: Heat-Maps Super Resolution Through Augmentation. In *ICASSP 2020*

# Augmented Grad-CAM

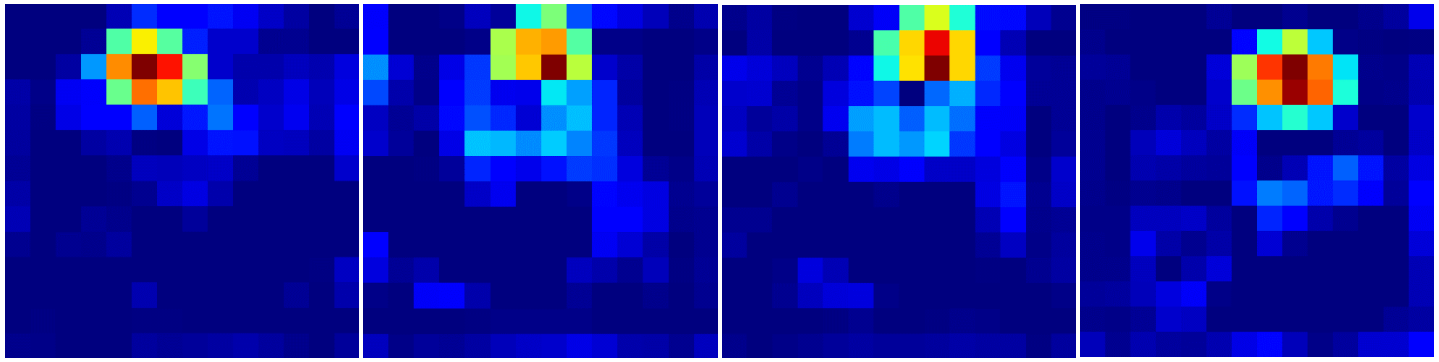



Morbidelli, P., Carrera, D., Rossi, B., Fragneto, P., & Boracchi, G. (2020, May). Augmented Grad-CAM: Heat-Maps Super Resolution Through Augmentation. In *ICASSP 2020*

# The Super-Resolution Approach

We perform heat-map Super-Resolution (**SR**) by taking advantage of the information shared in multiple low-resolution heat-maps computed from the **same input under different – but known – transformations**

CNNs are in general invariant to roto-translations, in terms of predictions, but each $\boldsymbol{g}_\ell$ actually contains different information



$g_1$ $g_2$ $g_3$ $g_4$

General approach, our SR framework can be combined with any visualization tool (not only Grad-CAM)

Morbidelli, P., Carrera, D., Rossi, B., Fragneto, P., & Boracchi, G "Augmented Grad-CAM: Heat-Maps Super Resolution Through Augmentation". ICASSP 2020

# The Super-Resolution Formulation

We model heat-maps computed by Grad-CAM as the result of an **unknown downsampling operator** $\mathcal{D} : \mathbb{R}^{N \times M} \rightarrow \mathbb{R}^{n \times m}$

The high-resolution heat-map $\boldsymbol{h}$ is recovered by solving an inverse problem

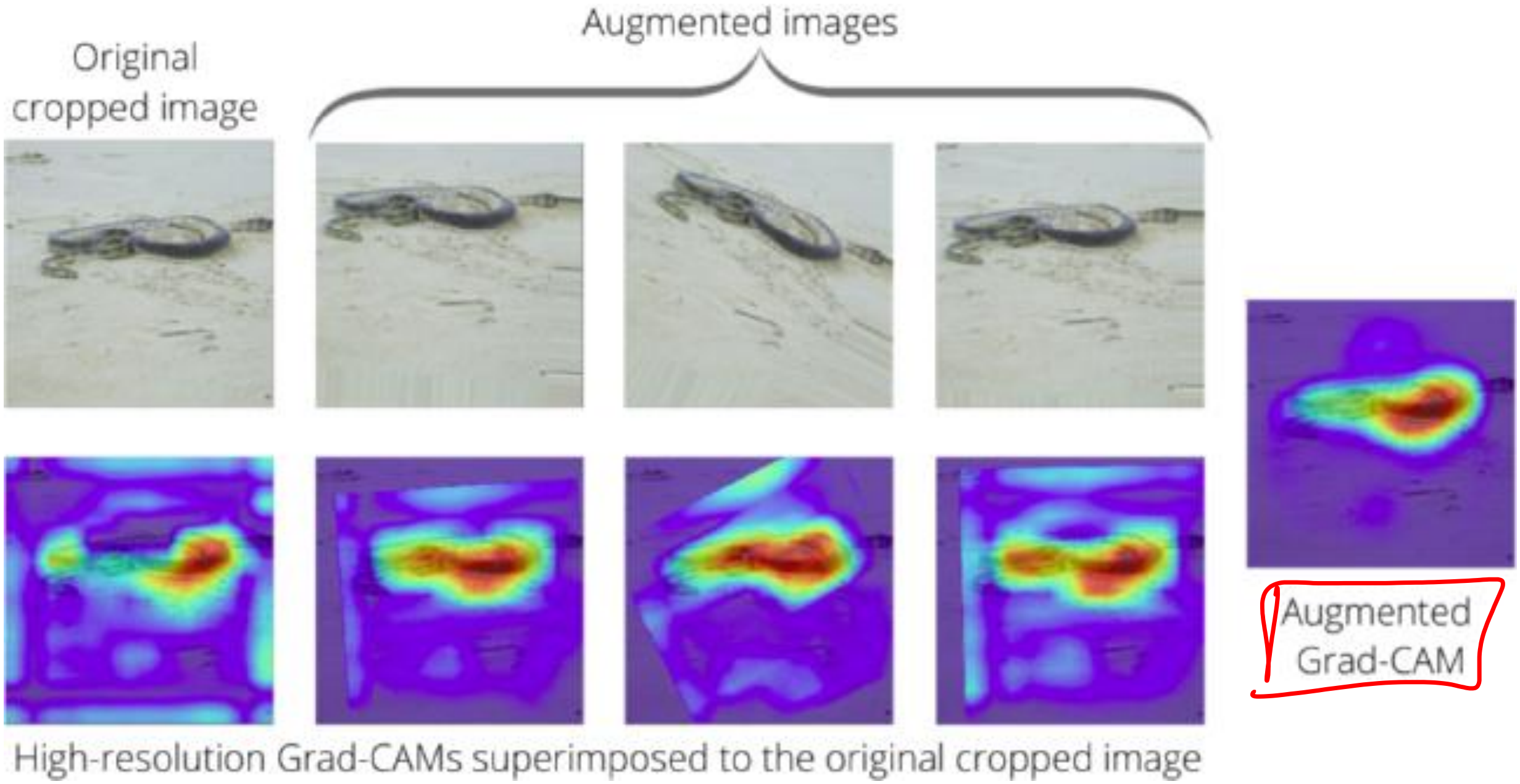
$$\underset{h}{\operatorname{argmin}} \frac{1}{2} \sum_{l=1}^{L} \| \mathcal{D}\mathcal{A}_\ell h - g_\ell \|_2^2 + \lambda TV_{\ell_1}(h) + \frac{\mu}{2} \|h\|_2^2 \quad (1)$$

$TV_{\ell_1}$: Anistropic Total Variation regularization is used to preserve the edges in the target heat-map (high-resolution)

$$TV_{\ell_1}(\boldsymbol{h}) = \Sigma_{i,j} \|\partial_x \boldsymbol{h}(i,j)\| + \|\partial_y \boldsymbol{h}(i,j)\| \quad (2)$$

This is solved through Subgradient Descent since the function is convex and non-smooth

Morbidelli, P., Carrera, D., Rossi, B., Fragneto, P., & Boracchi, G "Augmented Grad-CAM: Heat-Maps Super Resolution Through Augmentation". ICASSP 2020




Morbidelli, P., Carrera, D., Rossi, B., Fragneto, P., & Boracchi, G "Augmented Grad-CAM: Heat-Maps Super Resolution Through Augmentation". ICASSP 2020

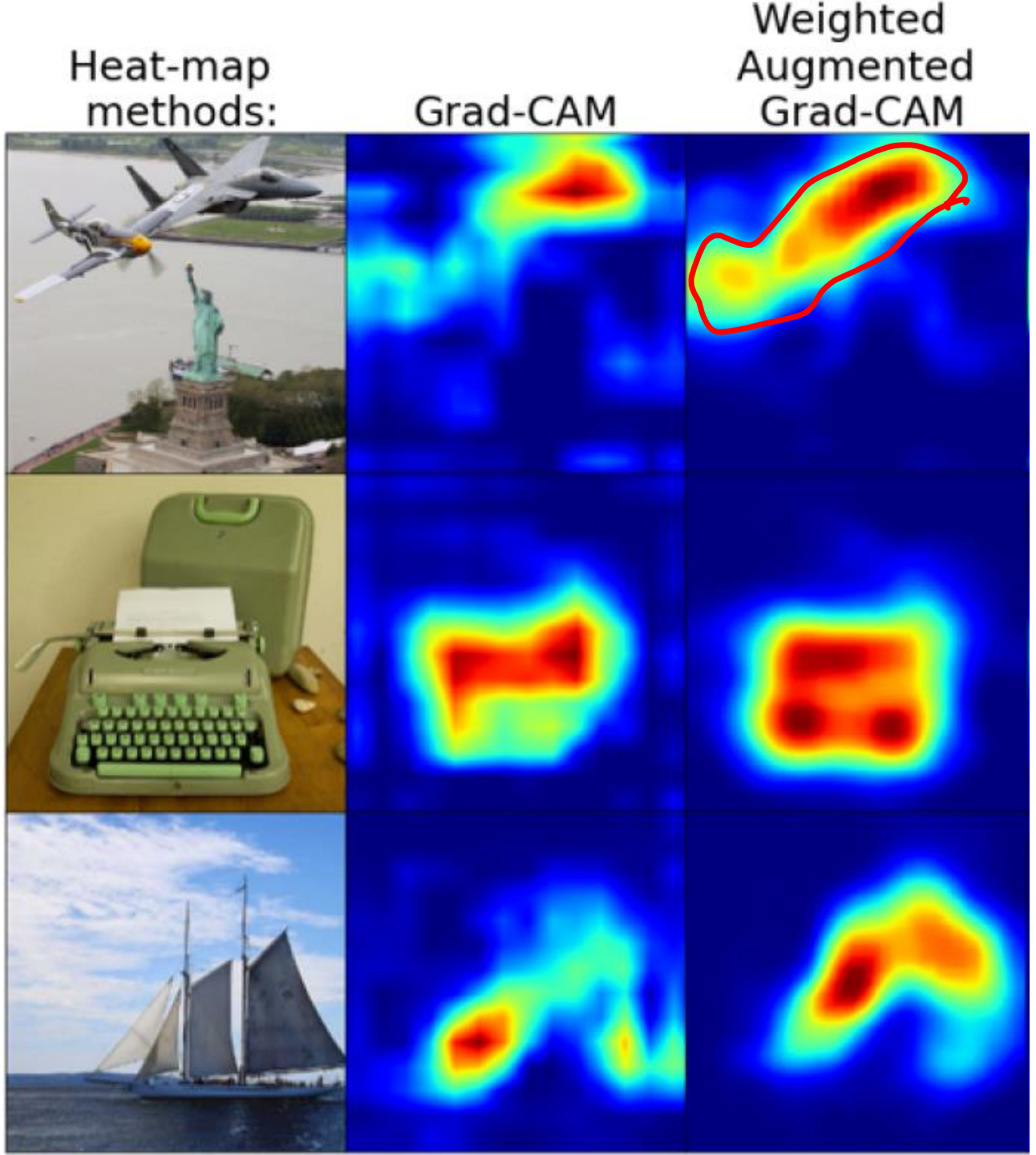
# Augmented Grad-CAM

"mash"

(a) Grad-CAM. (b) Augmented Grad-CAM.

Morbidelli, P., Carrera, D., Rossi, B., Fragneto, P., & Boracchi, G. (2020, May). Augmented Grad-CAM: Heat-Maps Super Resolution Through Augmentation. In *ICASSP 2020*

# Augmented Grad-CAM



# Other Gradient-based Saliency Maps

**Grad-CAM++** : Same formulation of CAM as Grad-CAM, but weights are computed by higher-order derivatives of the class score with respect to the feature maps. Increases the localization accuracy of the heat-maps in presence of multiple occurrence of the same object in the image.

**Sharpen Focus:** highlights only the pixels where the gradients are positive.

$$w_k^c = \frac{1}{nm} \sum_i \sum_j \text{RELU}\left(\frac{\partial y_c}{\partial \boldsymbol{a_k}(\boldsymbol{i}, \boldsymbol{j})}\right)$$

**Smooth Grad-CAM++:** it averages multiple heat-maps corresponding to noisy versions of the same input image.

A. Chattopadhay, A. Sarkar, P. Howlader, and V. N. Balasubramanian, "Grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks," WACV, 2018.

D. Omeiza, S. Speakman, C. Cintas, and K. Weldermariam, "Smoothgrad-CAM++: An enhanced inference level visualization technique for deep convolutional neural network models"

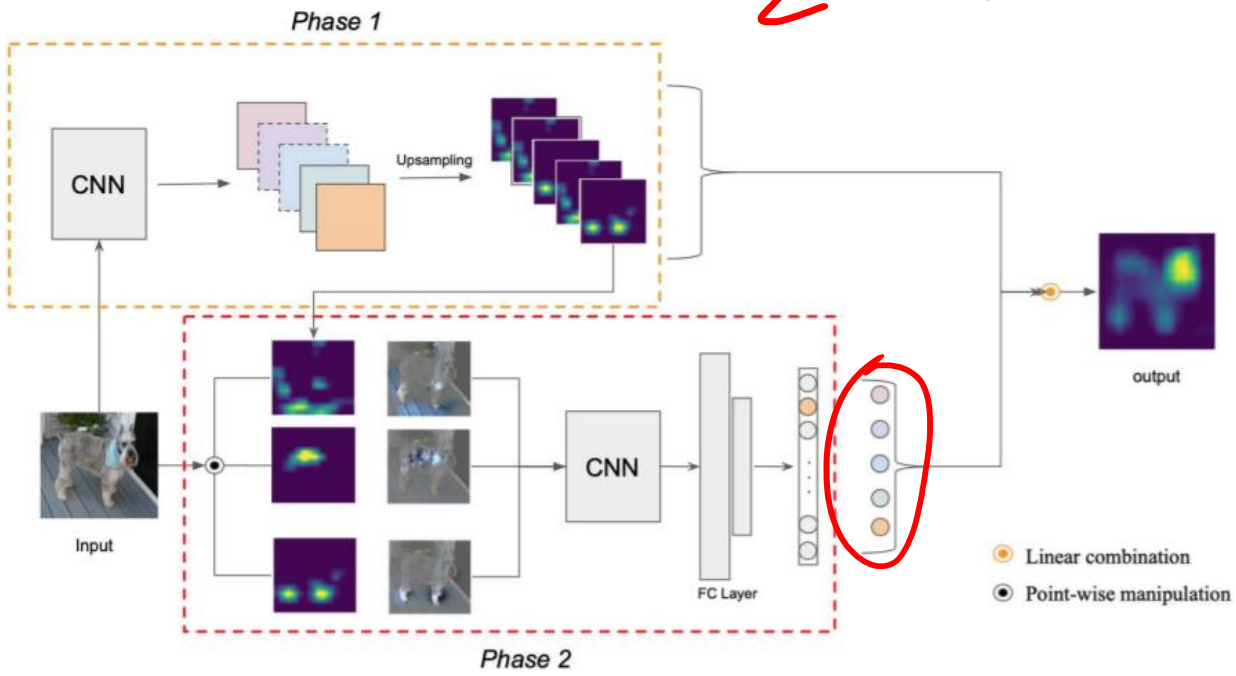
# Other Perturbation-based Saliency Maps

**Idea:** Perturb the input image and assess how the class score changes.

**Score-CAM:** Each feature map (upsampled and normalized in [0,1]) operates as a mask on the original image, which is then forward-passed to obtain the score on the target class.

$$w_k^c = CNN(\boldsymbol{h_k^c} \circ \boldsymbol{x}) - CNN(\boldsymbol{x})$$

$\sum w_k^c f_k$




H. Wang, Z. Wang, M. Du, F. Yang, Z. Zhang, S. Ding, P. Mardziel, and X. Hu, "Score-cam: Score-weighted visual explanations for convolutional neural networks," 2020

# Limitations of Saliency Maps


Figure 1: Based on saliency maps it is unclear why this image is labelled as a *cat* rather than a *laundry basket*. Grad-CAM [27] explanations are essentially the same for both classes.

L. Giulivi M.J. Carman G. Boracchi "Perception Visualization: Seeing Through the Eyes of a DNN" BMVC 2021

# Perception Visualization

**Perception Visualization:** provides explanations by exploiting a neural network to invert latent representations

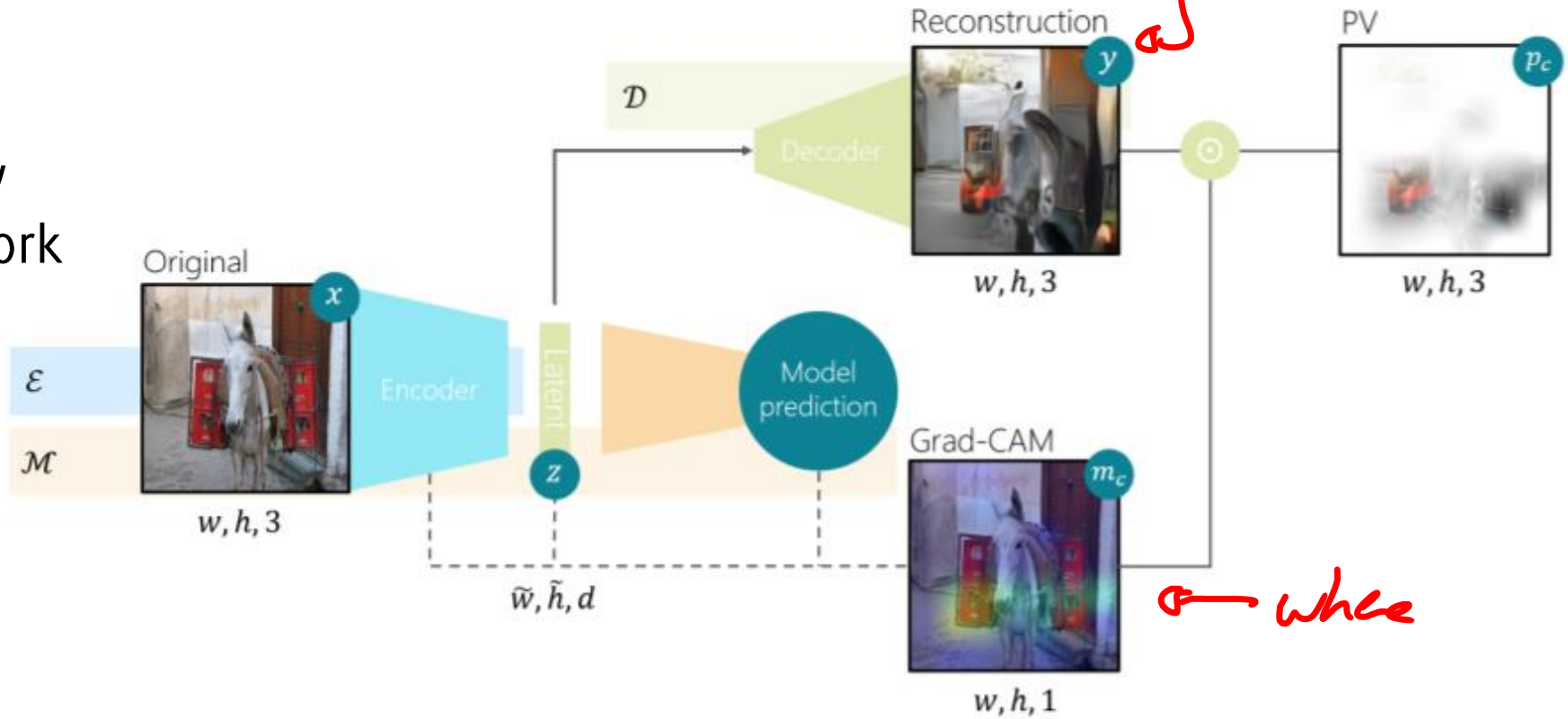


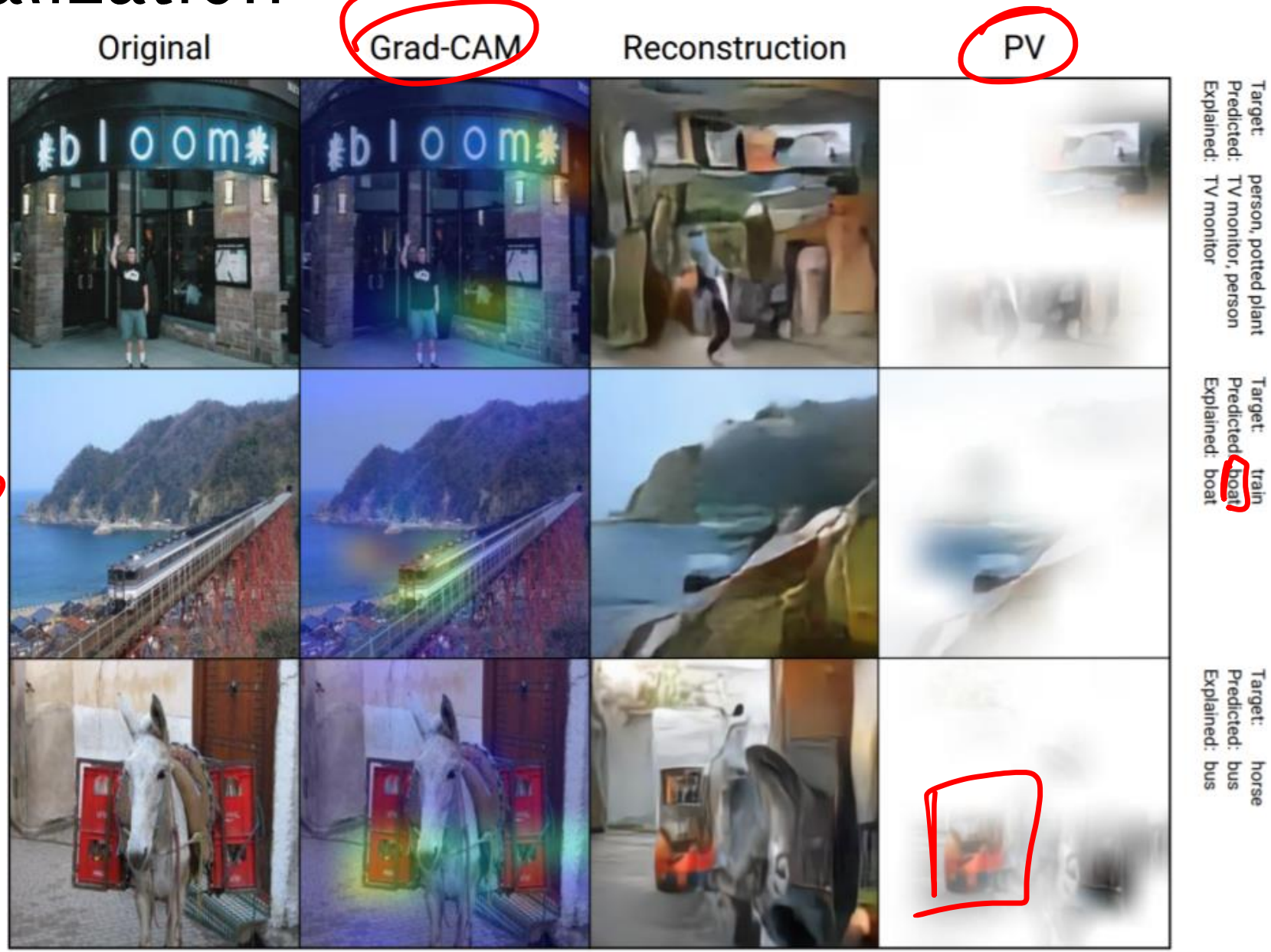

Figure 3: An overview of our method and interactions between the models involved. Encoder $\mathcal{E}$ is a truncation of the model $\mathcal{M}$ which we want to explain, decoder $\mathcal{D}$ is trained to reconstruct the encoder's latent representations. From these, we compute Grad-CAM saliency maps and reconstructions, which are then combined to obtain PV.

L. Giulivi M.J. Carman G. Boracchi "Perception Visualization: Seeing Through the Eyes of a DNN" BMVC 2021

# Perception Visualization

Give better insight on the model's functioning than what was previously achievable using only saliency maps.

A study on circa 100 subjects shows that PV is able to **help respondents better determine the predicted class** in cases where the **model had made an error**




L. Giulivi M.J. Carman G. Boracchi "Perception Visualization: Seeing Through the Eyes of a DNN" BMVC 2021

# Other popular architectures

# Outline

Lecture inspired to:

Notes accompanying the Stanford CS class CS231n: Convolutional Neural Networks for Visual Recognition http://cs231n.github.io/

… and papers cited in here!

# Going Deeper with Convolutions

Christian Szegedy[1], Wei Liu[2], Yangqing Jia[1], Pierre Sermanet[1], Scott Reed[3],
Dragomir Anguelov[1], Dumitru Erhan[1], Vincent Vanhoucke[1], Andrew Rabinovich[4]
[1]Google Inc. [2]University of North Carolina, Chapel Hill
[3]University of Michigan, Ann Arbor [4]Magic Leap Inc.

[1]{szegedy,jiayq,sermanet,dragomir,dumitru,vanhoucke}@google.com

[2]wliu@cs.unc.edu, [3]reedscott@umich.edu, [4]arabinovich@magicleap.com

# Inception Module

The most straightforward way of improving the performance of deep neural networks is by **increasing their size** (**either in depth or width**)

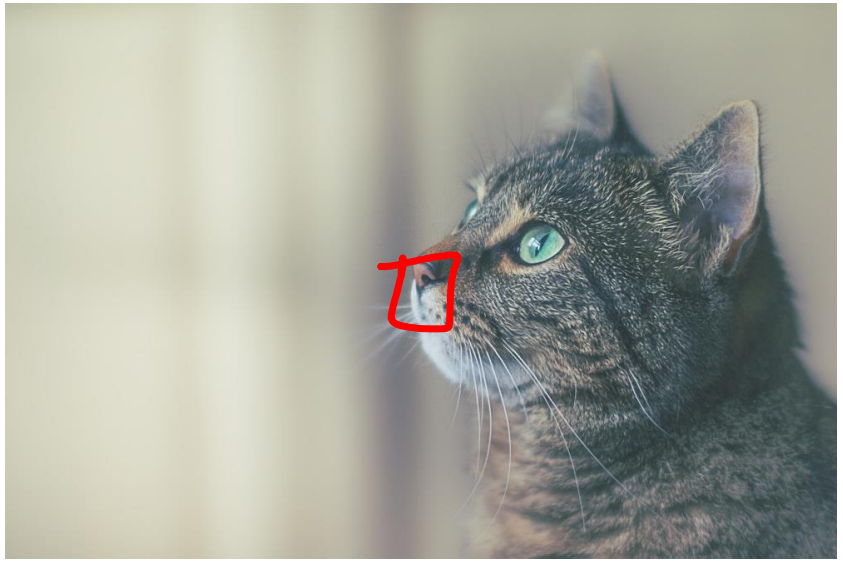
Bigger size typically means

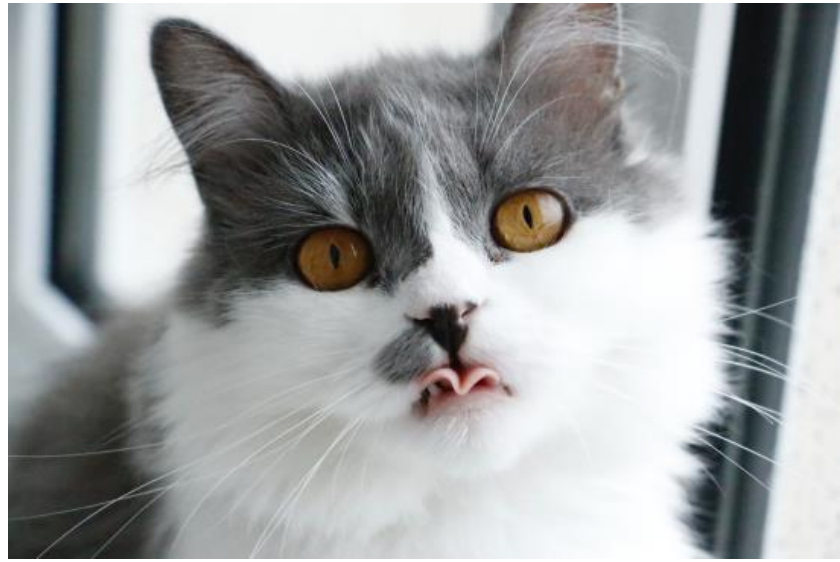
- a larger number of parameters, which makes the enlarged network more **prone to overfitting**
- dramatically **increase** in the use of **computational resources**

Moreover image features might appear at different scale, and it is **difficult to define the right filter size**

# Features might appear at different scales

Difficult to set the right kernel size!





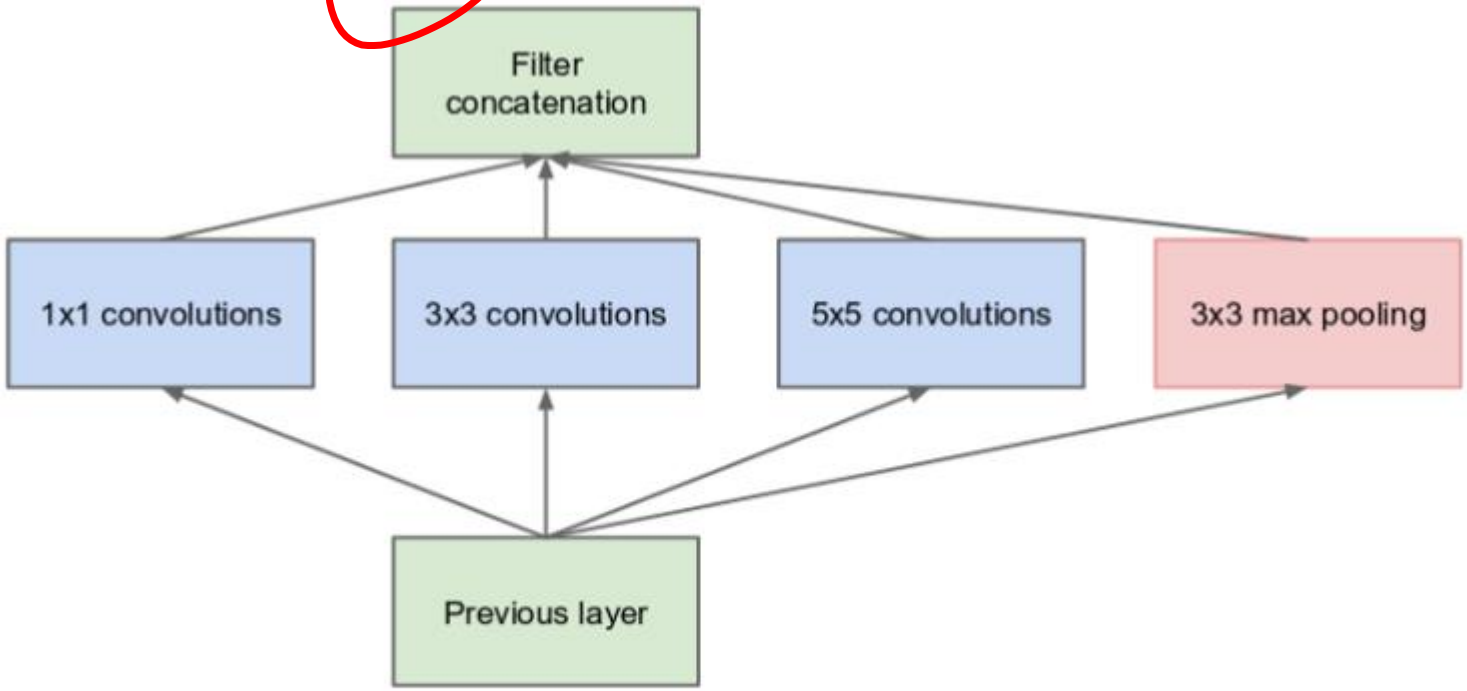
Photos from Unsplash

# GoogLeNet and Inception v1 (2014)

Deep network, with **high computational efficiency**

Only **5 million parameters**, 22 layers of Inception modules

Won 2014 ILSVR-classification challenge (6,7% top 5 classification error)



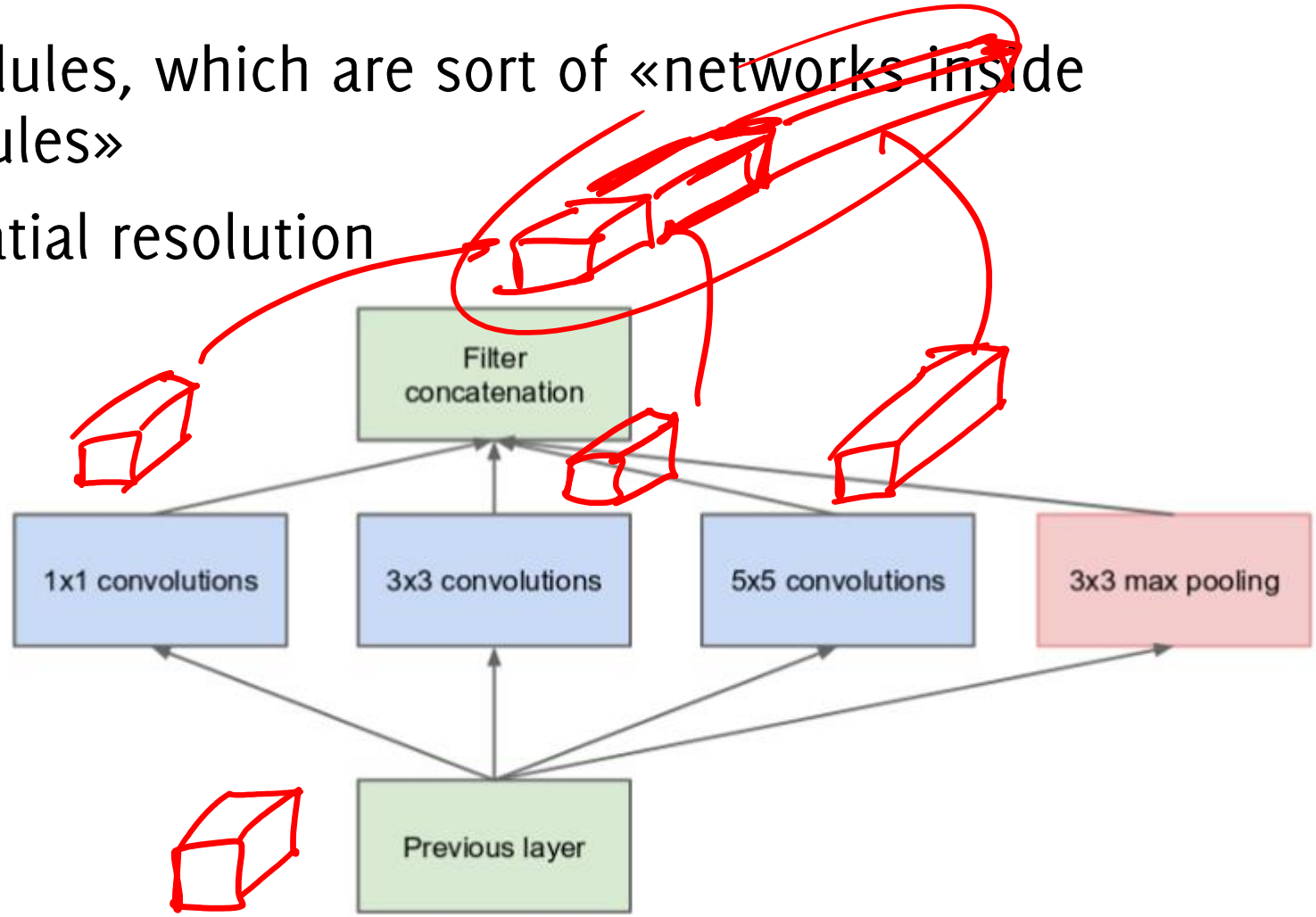

(a) Inception module, naïve version

Szegedy et al. "Going deeper with convolutions." *CVPR 2015*

# GoogLeNet and Inception v1 (2014)

It is based on inception modules, which are sort of «networks inside the network» or «local modules»

Concatenation preserves spatial resolution

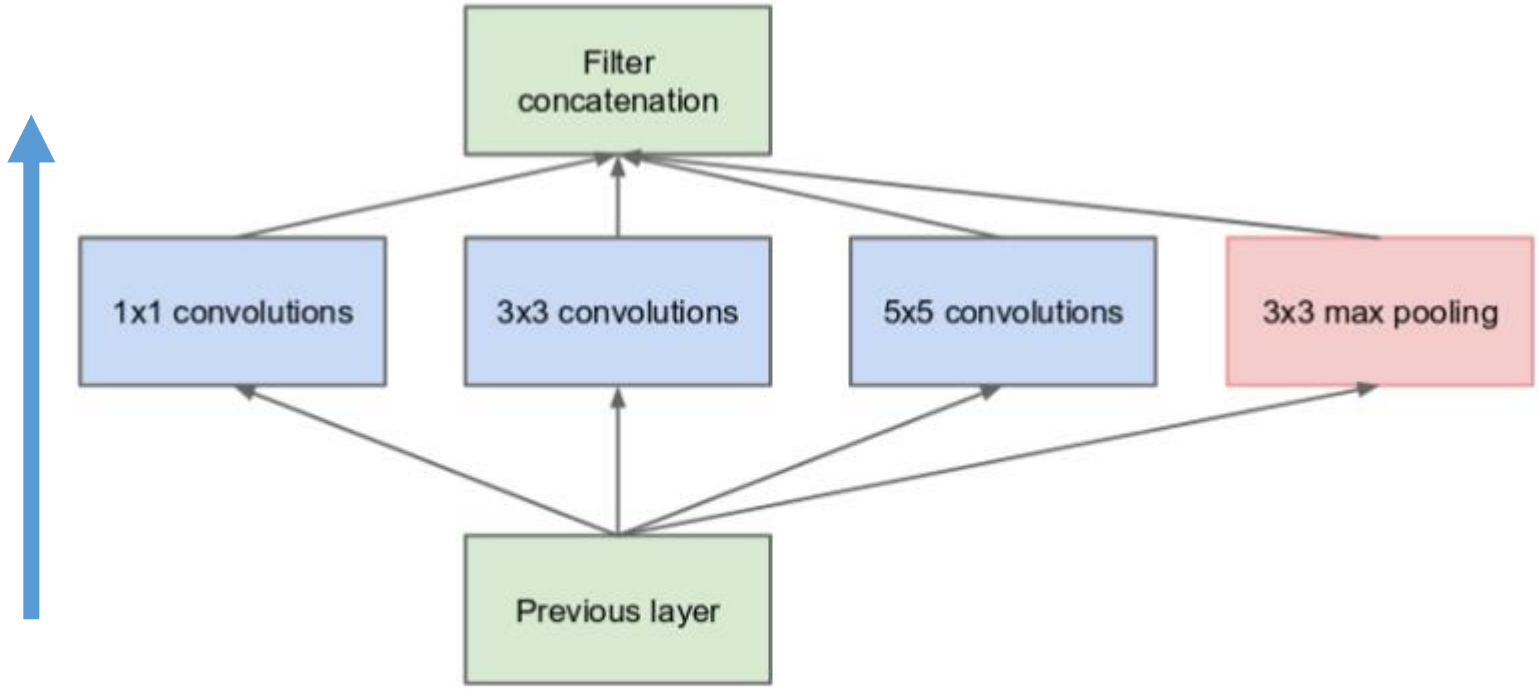



(a) Inception module, naïve version

Szegedy et al. "Going deeper with convolutions." *CVPR 2015*

# Inception Module (2014)

- Choosing the **right kernel size** for the convolution operations is difficult, as image might show relevant features at different scale
- Deep networks **tend to over fit** and to become very computationally expensive




(a) Inception module, naïve version

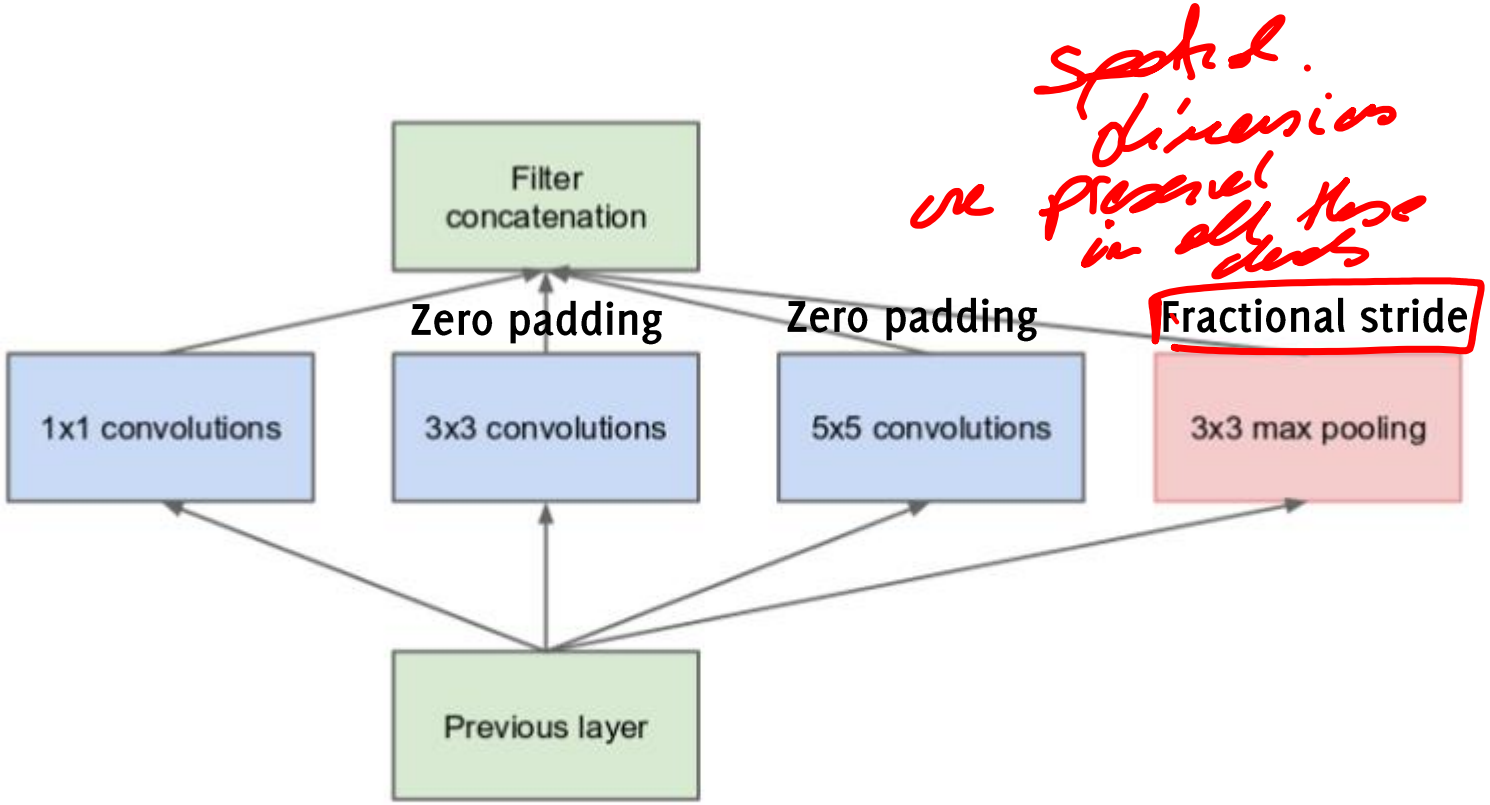
Szegedy et al. "Going deeper with convolutions." *CVPR 2015.*

# Inception Module (2014)

The solution is to **exploit multiple filter size** at the same level (1x1, 3x3, 5x5) and then **merge by concatenation** the output activation maps together

All the blocks preserve the spatial dimension by zero padding (convolutional filters) or by fractional stride (for Maxpooling)

Thus, outputs can be concatenated depth-wise



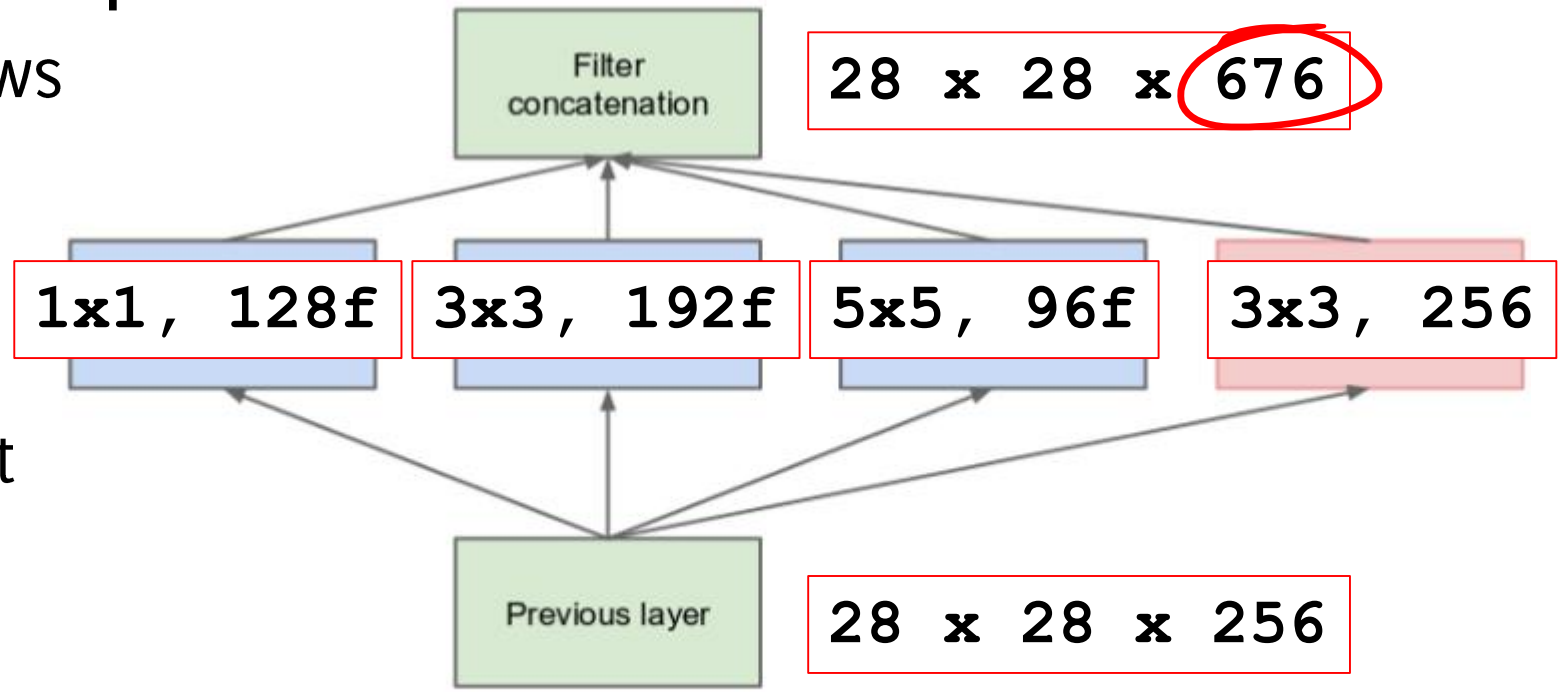

(a) Inception module, naïve version

Szegedy et al. "Going deeper with convolutions." *CVPR 2015*

# Inception Module (2014)

The solution is to exploit multiple filter size at the same level (1x1, 3x3, 5x5) and then **merge by concatenation** the output activation maps together

- **Zero padding to preserve spatial size**
- The activation map grows much in depth
- A large number of operations to be performed due to the large depth of the input of each convolutional block: **854M operations** in this example




(a) Inception module, naïve version

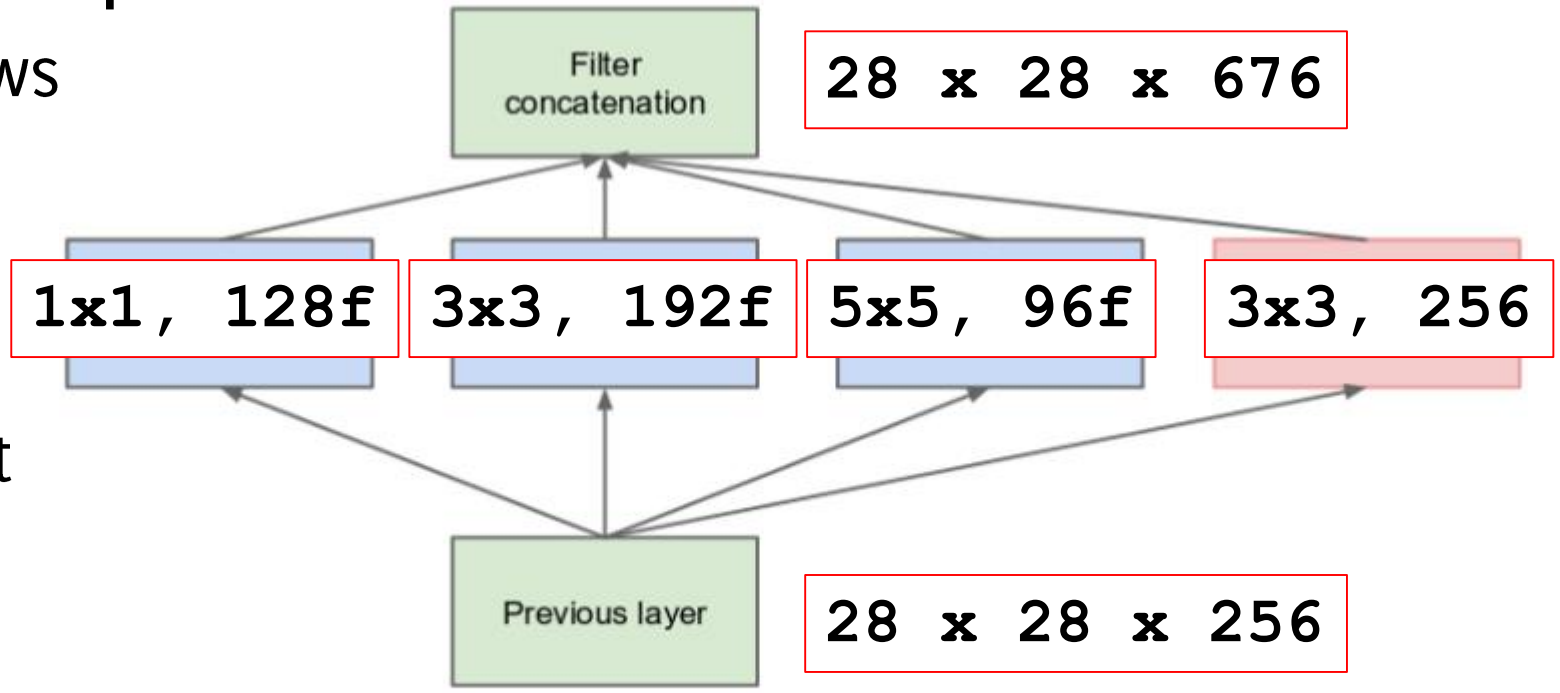
Szegedy et al. "Going deeper with convolutions." *CVPR 2015*

# Inception Module (2014)

The sol… 3x3, 5x5) an… togethe…

**The spatial extent is preserved, but the depth of the activation map is much expanded. This is very expensive to compute**

- **Zero padding to preserve spatial size**
- The activation map grows much in depth
- A large number of operations to be performed due to the large depth of the input of each convolutional block: **854M operations** in this example




(a) Inception module, naïve version

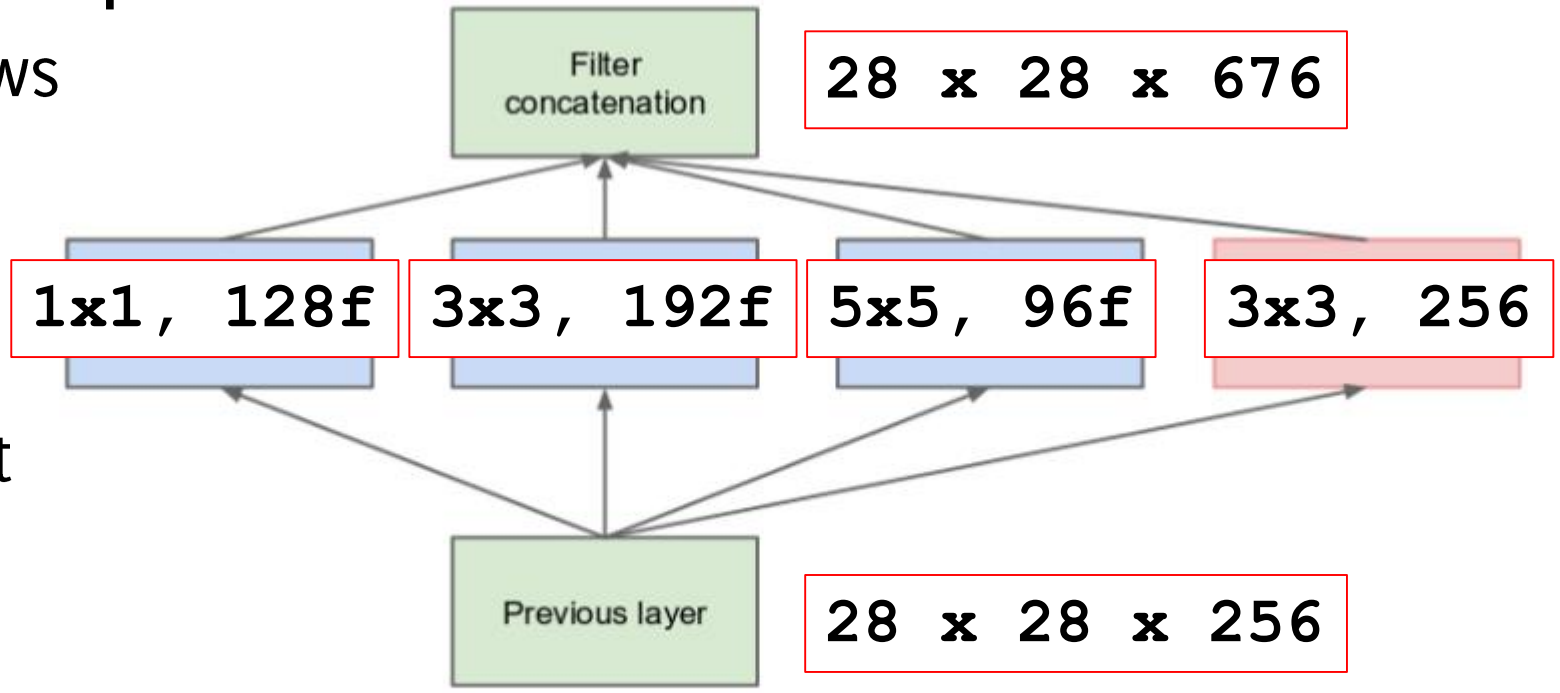
Szegedy et al. "Going deeper with convolutions." *CVPR 2015*

# Inception Module (2014)

The sol… 3x3, 5x5) an… togethe…

**Computational problems will get significantly worst when stacking multiple layers…**

- **Zero padding to preserve spatial size**
- The activation map grows much in depth
- A large number of operations to be performed due to the large depth of the input of each convolutional block: **854M operations** in this example



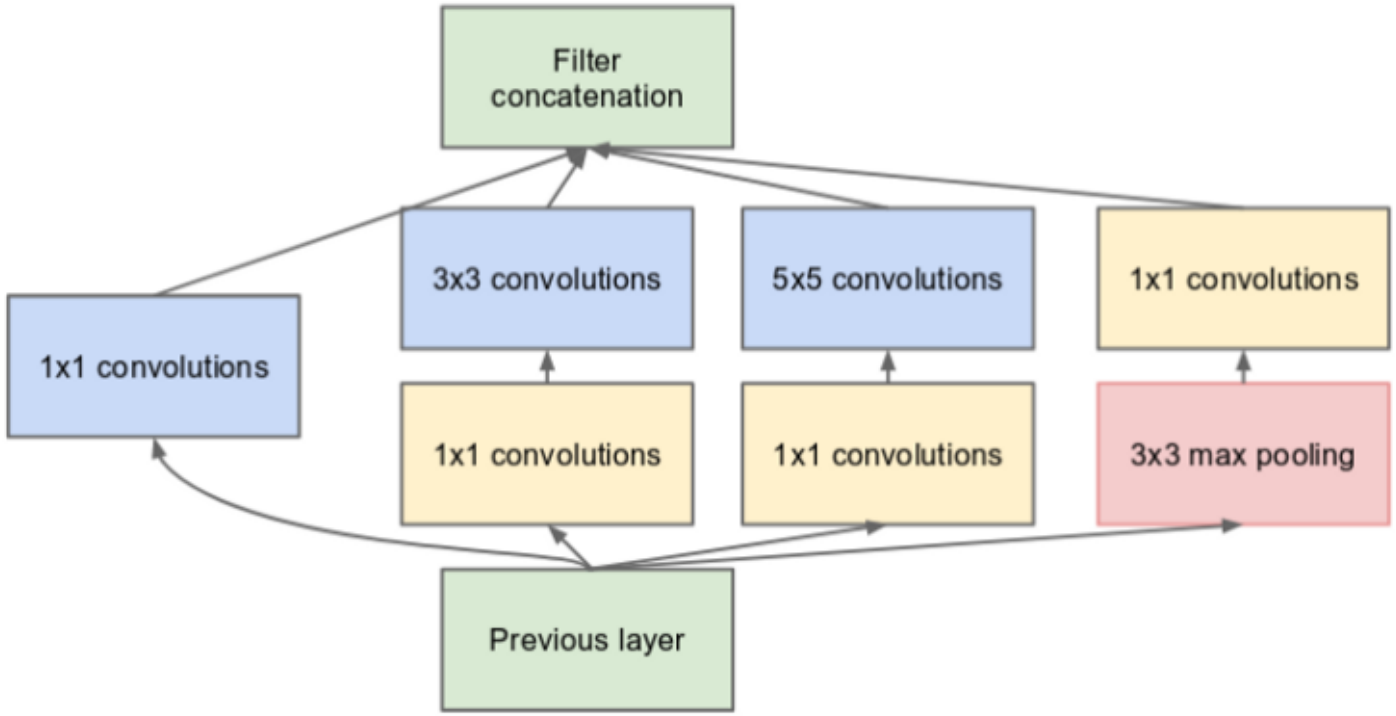

(a) Inception module, naïve version

Szegedy et al. "Going deeper with convolutions." *CVPR 2015*

# Inception Module (2014)

To reduce the computational load of the network, the number of **input channels** is reduced by adding an **1x1 convolution** layers before the 3x3 and 5x5 convolutions

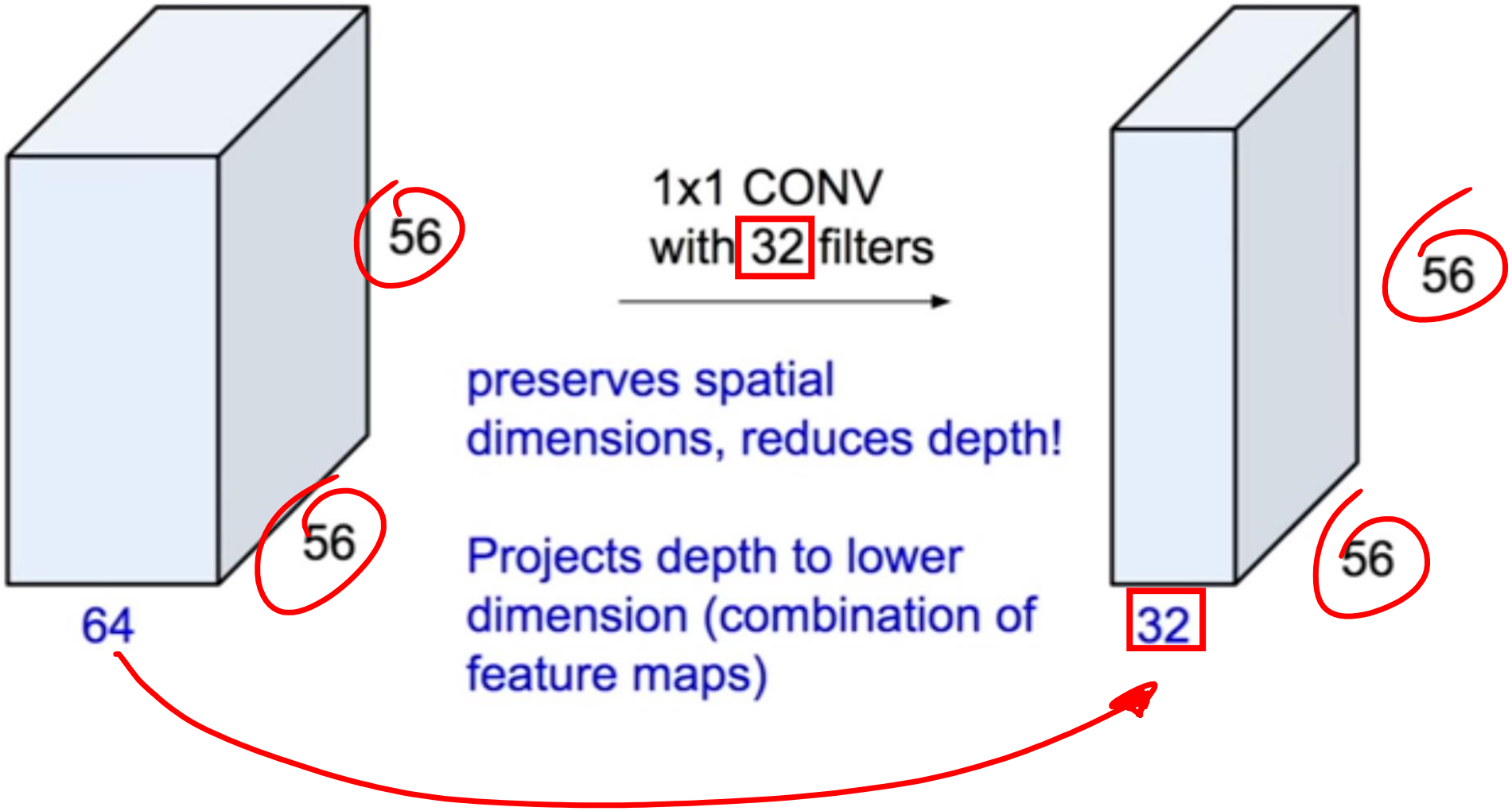
Using these 1x1 conv is referred to as **"bottleneck" layer**




(b) Inception module with dimension reductions

Szegedy et al. "Going deeper with convolutions." *CVPR 2015*

# 1x1 convolution layers as bottleneck

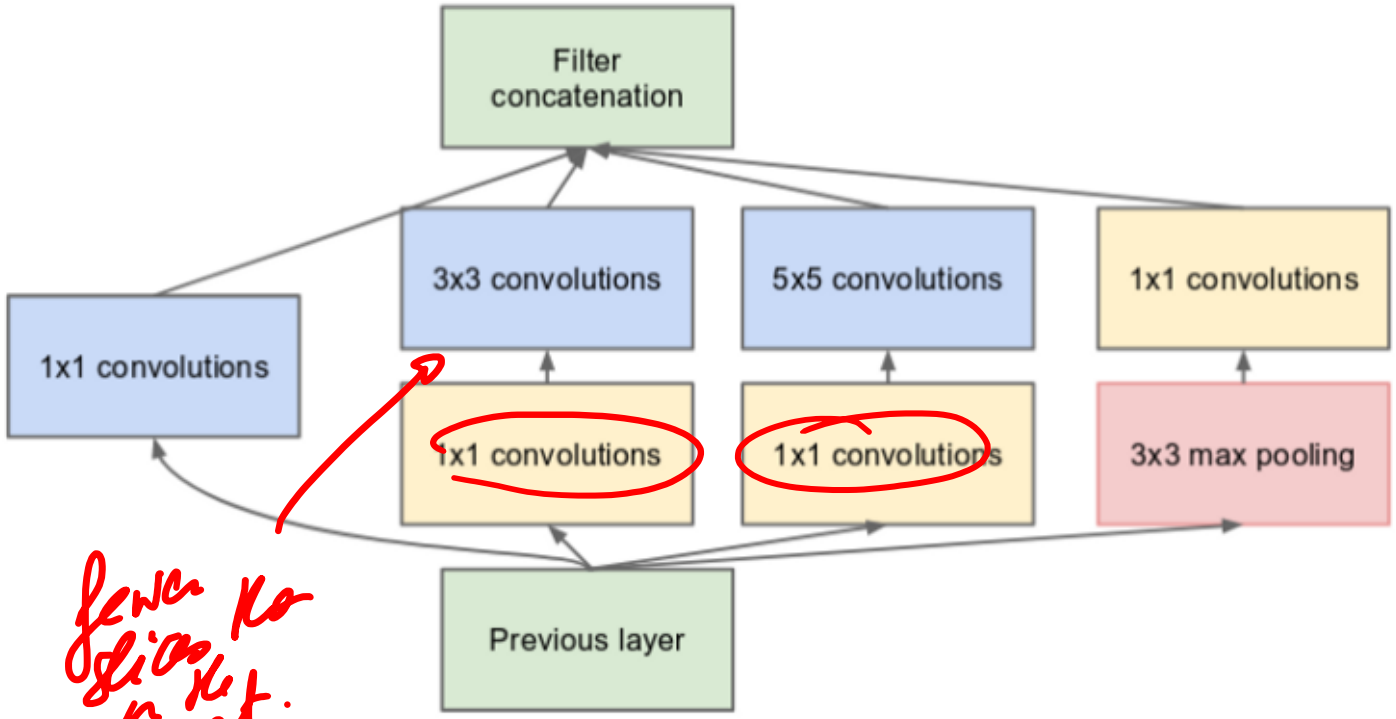


# Inception Module (2014)

To reduce the computational load of the network, the number of **input channels** is reduced by adding an **1x1 convolution** layers before the 3x3 and 5x5 convolutions

The output volume has similar size, but the number of operation required is significantly reduced due to the 1x1 conv:
**358M operations** now

**Adding 1x1 convolution** layers increases the number of nonlinearities



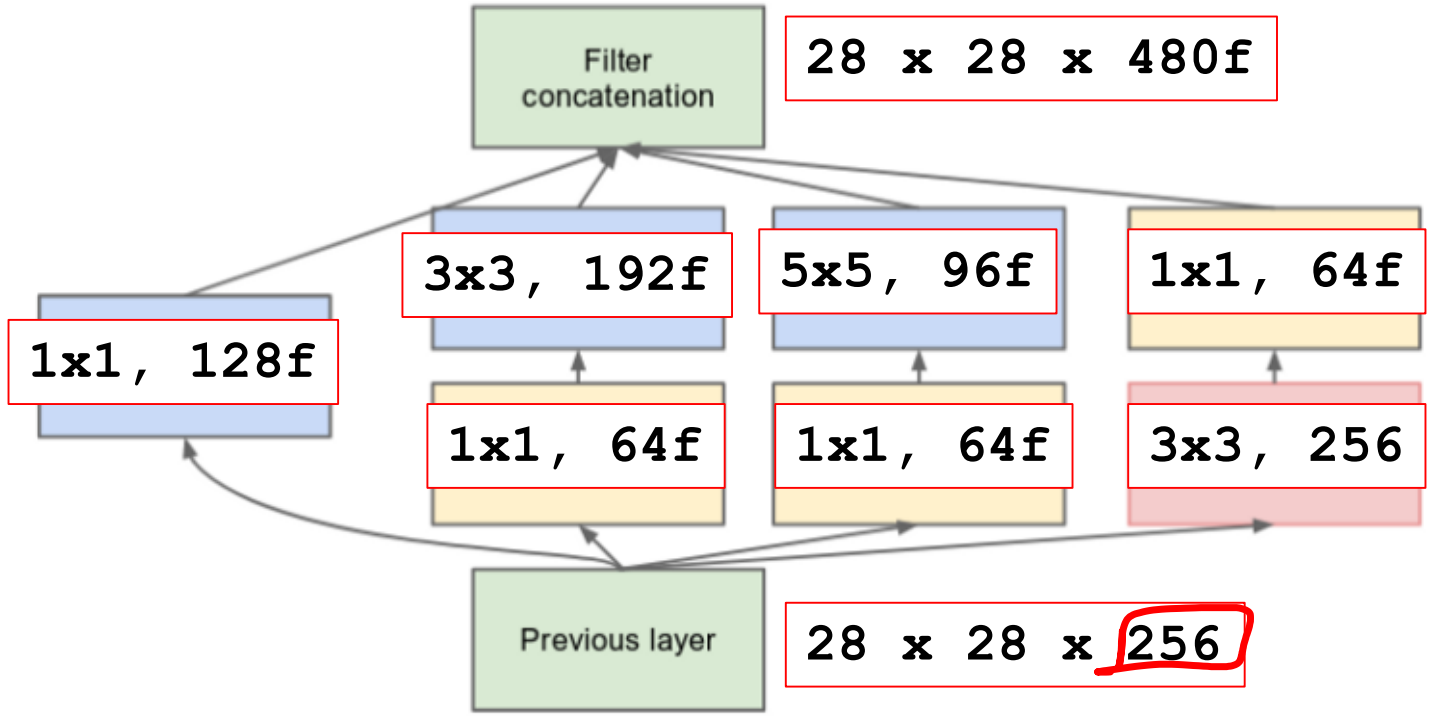

(b) Inception module with dimension reductions

Szegedy et al. "Going deeper with convolutions." *CVPR 2015*

# Inception Module (2014)

To reduce the computational load of the network, the number of **input channels** is reduced by adding an **1x1 convolution** layers before the 3x3 and 5x5 convolutions

The output volume has similar size, but the number of operation required is significantly reduced due to the 1x1 conv:
**358M operations** now



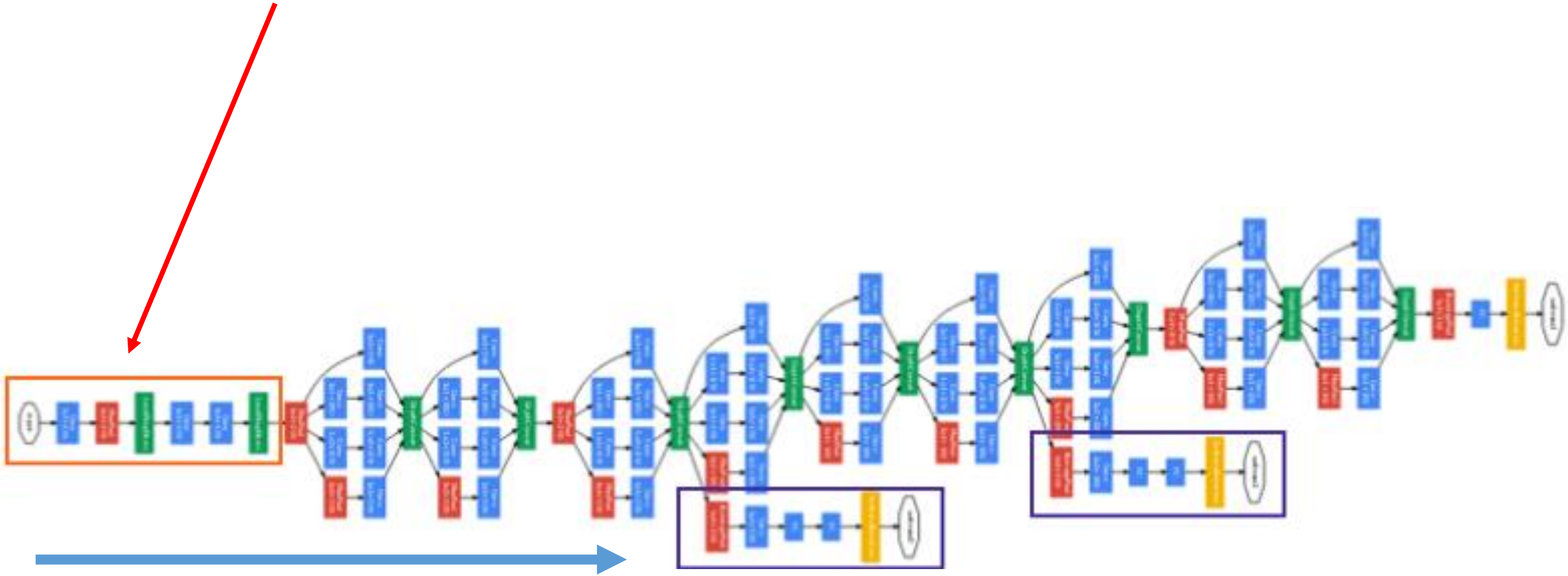

(b) Inception module with dimension reductions

Szegedy et al. "Going deeper with convolutions." *CVPR 2015*

# GoogLeNet (2014)

GoogleNet stacks 27 layers considering pooling ones.

At the beginning there are two blocks of conv + pool layers



Szegedy et al. "Going deeper with convolutions." *CVPR 2015*

# GoogLeNet (2014)

GoogleNet stacks 27 layers considering pooling ones.

Then, there are a stack of 9 of inception modules

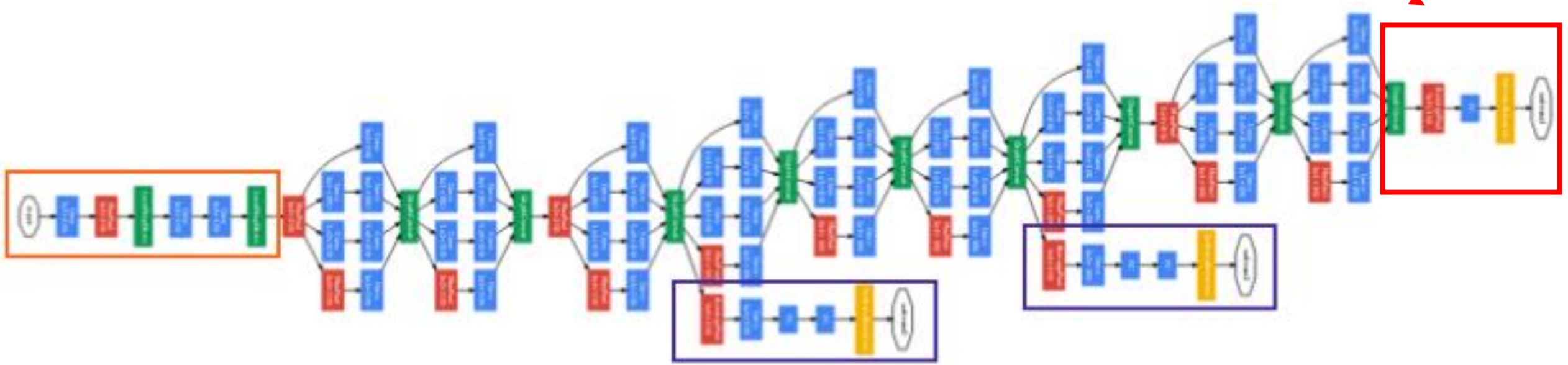
Szegedy et al. "Going deeper with convolutions." *CVPR 2015*

# GoogLeNet (2014)

GoogleNet stacks 27 layers considering pooling ones.

No Fully connected layer at the end, simple global averaging pooling, linear classifier + softmax.

Overall, it contains only 5 M parameters.

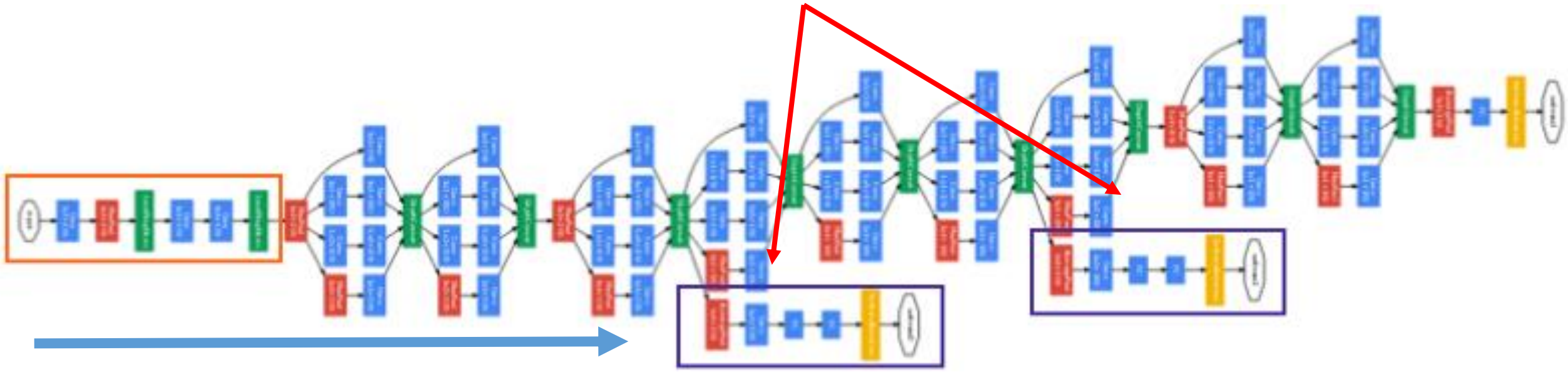


Szegedy et al. "Going deeper with convolutions." *CVPR 2015*

# GoogLeNet (2014)

It also suffers of the **dying neuron problem**, therefore the authors **add two extra auxiliary classifiers** on the intermediate representation **to compute an intermediate loss that is used during training.**

You expect intermediate layers to provide meaningful features for classification as well.



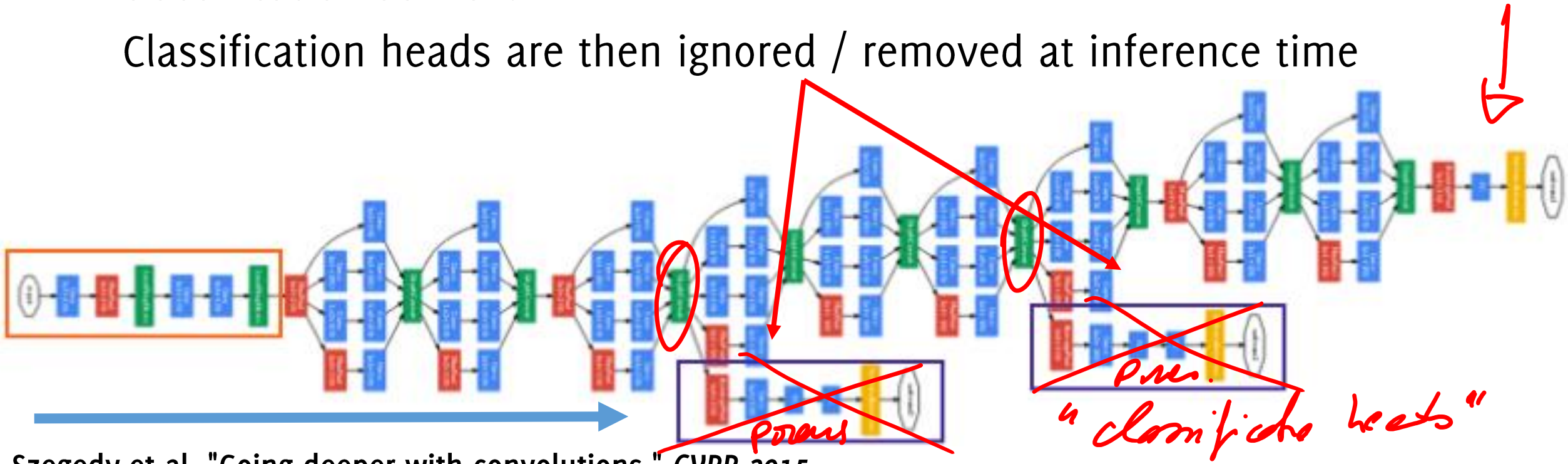
Szegedy et al. "Going deeper with convolutions." *CVPR 2015*

# GoogLeNet (2014)

It also suffers of the **dying neuron problem**, therefore the authors **add two extra auxiliary classifiers** on the intermediate representation **to compute an intermediate loss that is used during training.**

You expect intermediate layers to provide meaningful features for classification as well.

Classification heads are then ignored / removed at inference time




Szegedy et al. "Going deeper with convolutions." *CVPR 2015*



# Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

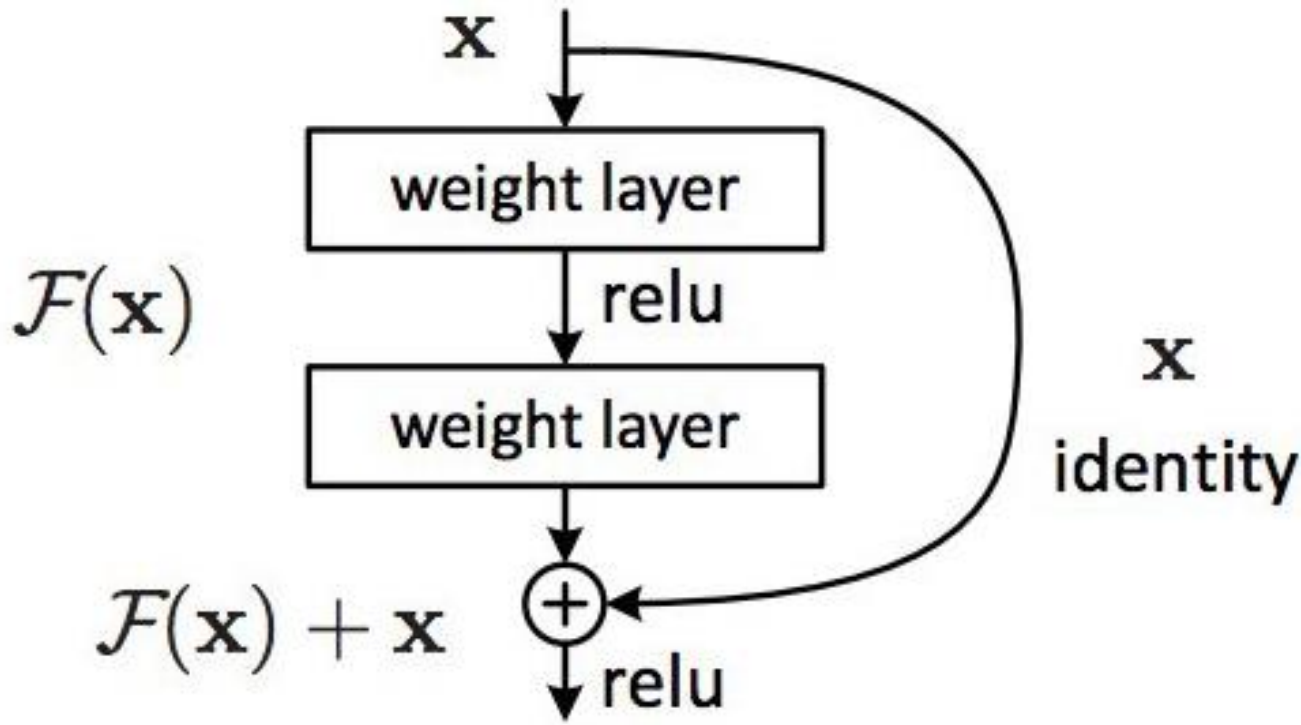
# ResNet (2015)

**Very Deep network**: 152 layers for a deep network trained on Imagenet!

1202 layers on CIFAR!

**2015 ILSVR winner** both localization and classification (3.57% top 5 classification error). **Better than *human performance***

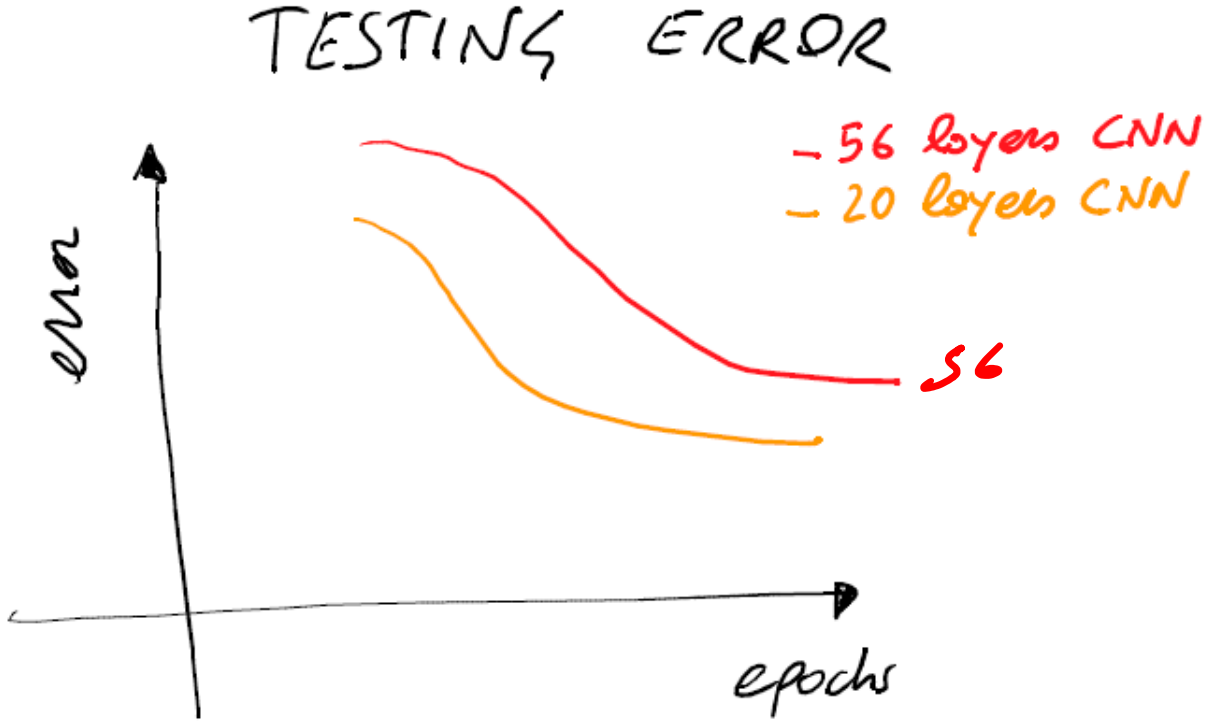
The main investigation was:
is it possible to continuosly improve accuracy by stacking more and more layers




Kaiming He, et al. "Deep Residual Learning for Image Recognition“ CVPR 2016

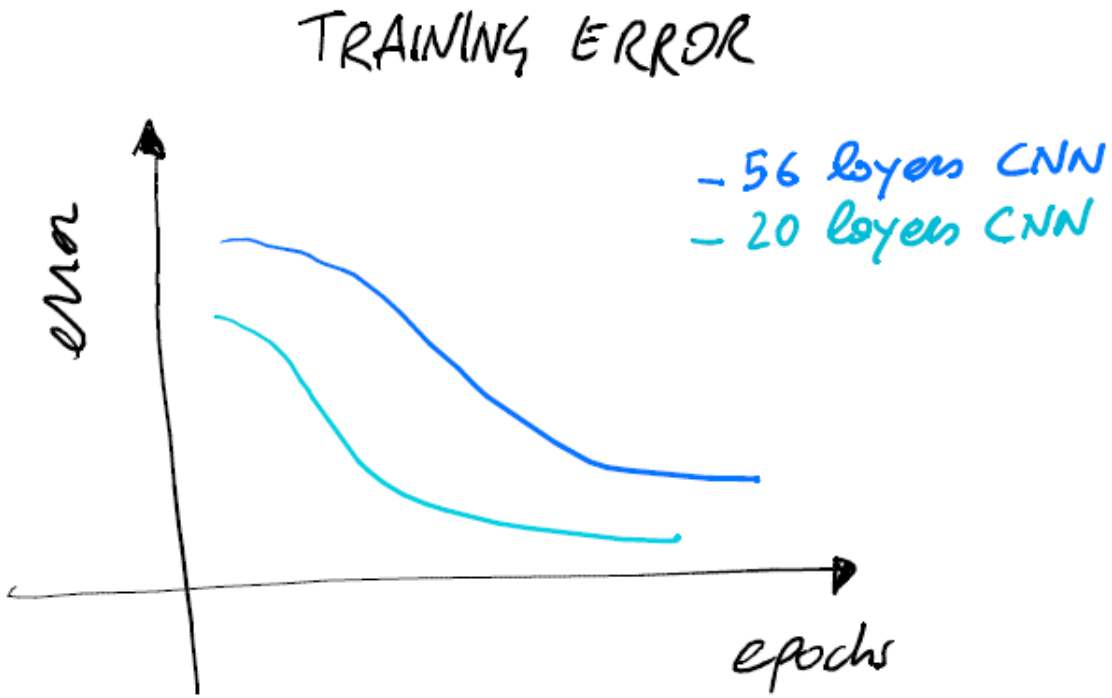
# ResNet (2015): The rationale

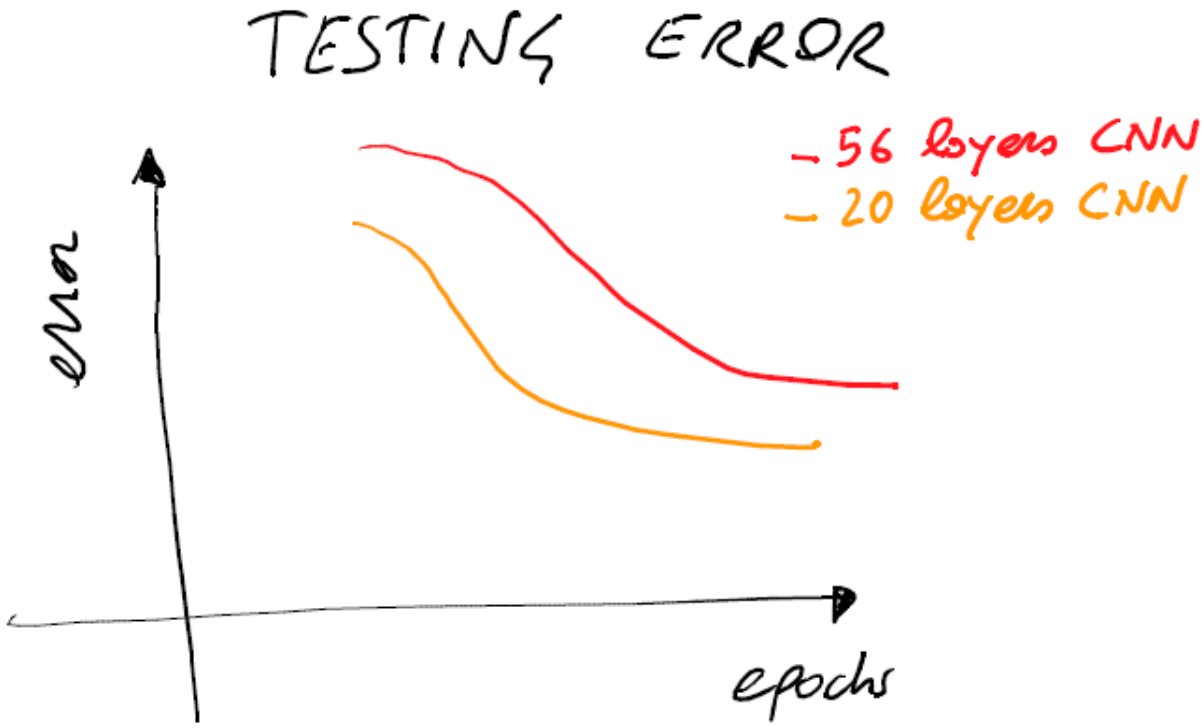
**Increasing the network depth**, by stacking an increasingly number of layers, **does not always improve performance**



# ResNet (2015): The rationale

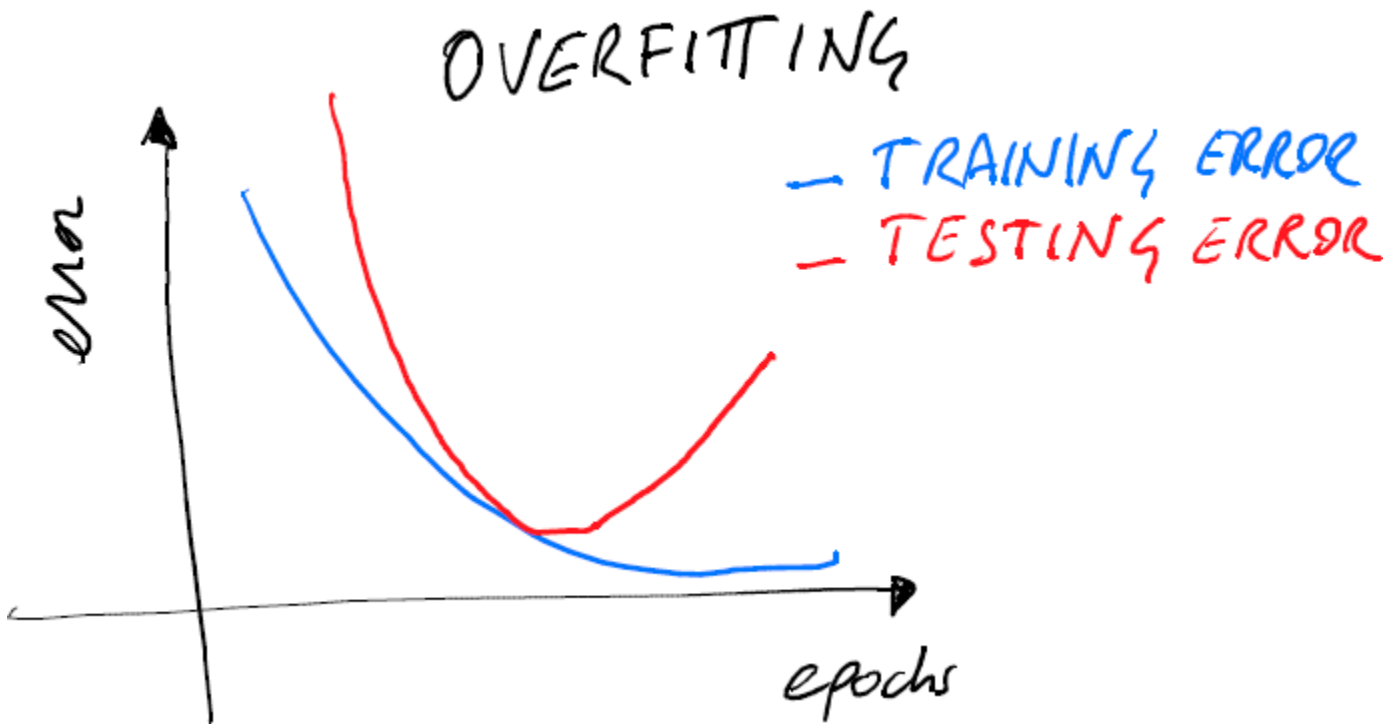
But **this is not due to overfitting**, since the same trend is shown in the training error



# ResNet (2015): The rationale

But **this is not due to overfitting**, since the same trend is shown in the training error, while for overfitting we have that training and test error diverge



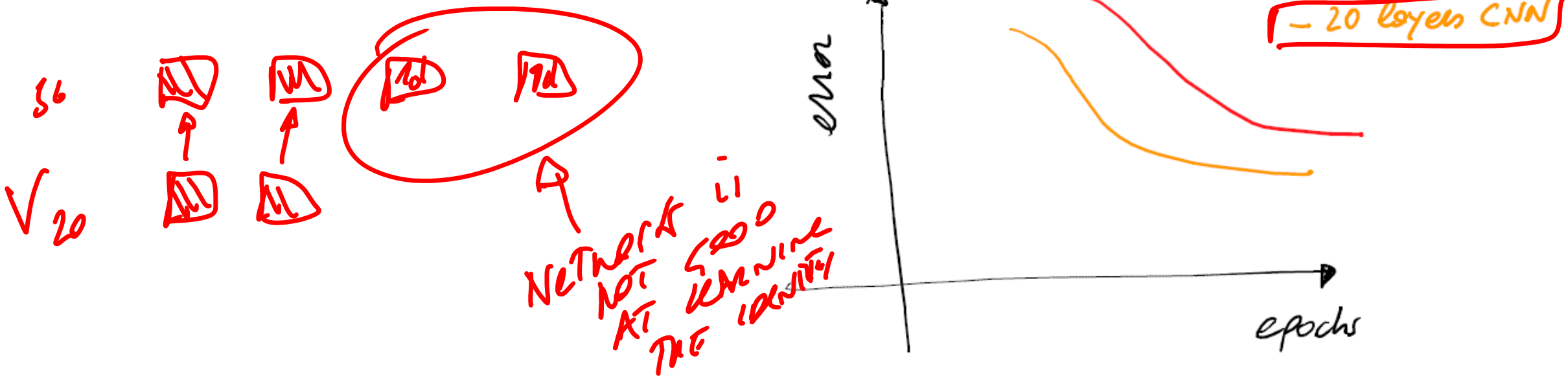

G. Boracchi

# ResNet (2015): the intuition

**Deeper model are harder to optimize than shallower models.**

However, we might in principle **copy the parameters of the shallow** network **in the deeper one** and then **in the remaining part**, set the weights to **yield an identity mapping**.

This should not impair the performance of the deeper model



# ResNet (2015): the intuition

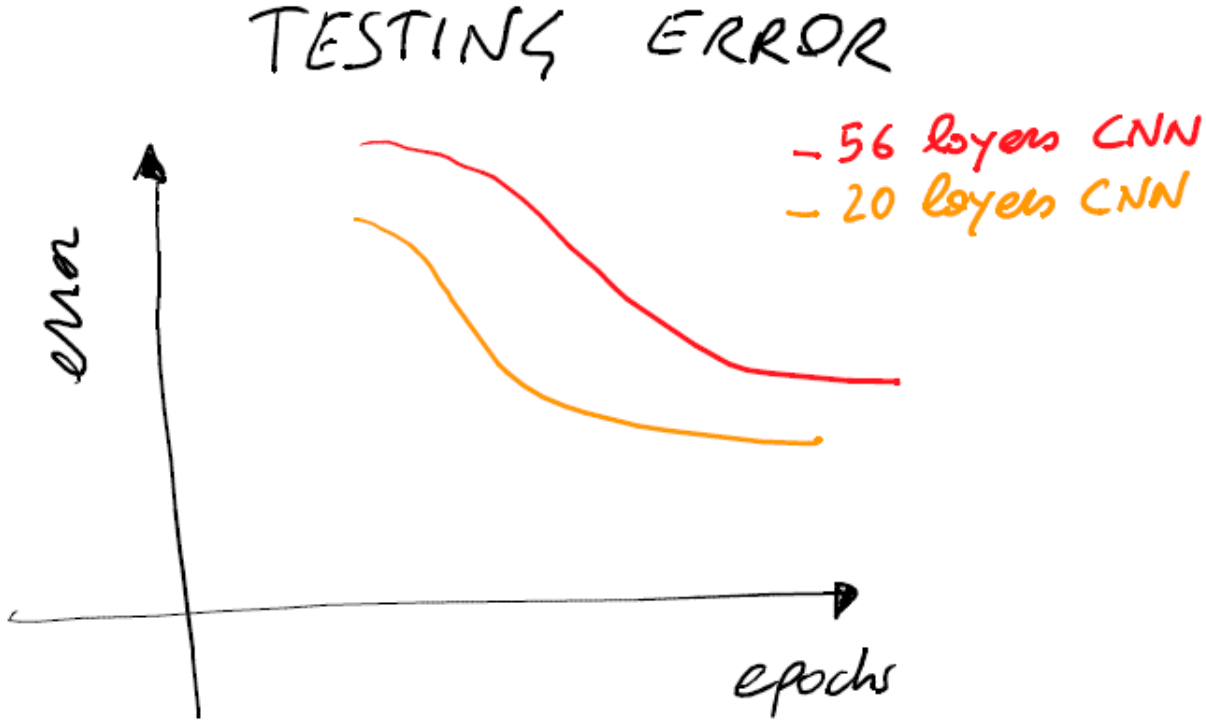
**Deeper model are harder to optimize than shallower models.**

However, we might in principle **copy the parameters of the shallow** network **in the deeper one** and then in the remaining part, set the weights to **yield an identity mapping**.

This should not impair the performance of the deeper model

**Deeper networks should be in principle as good as the shallow ones**

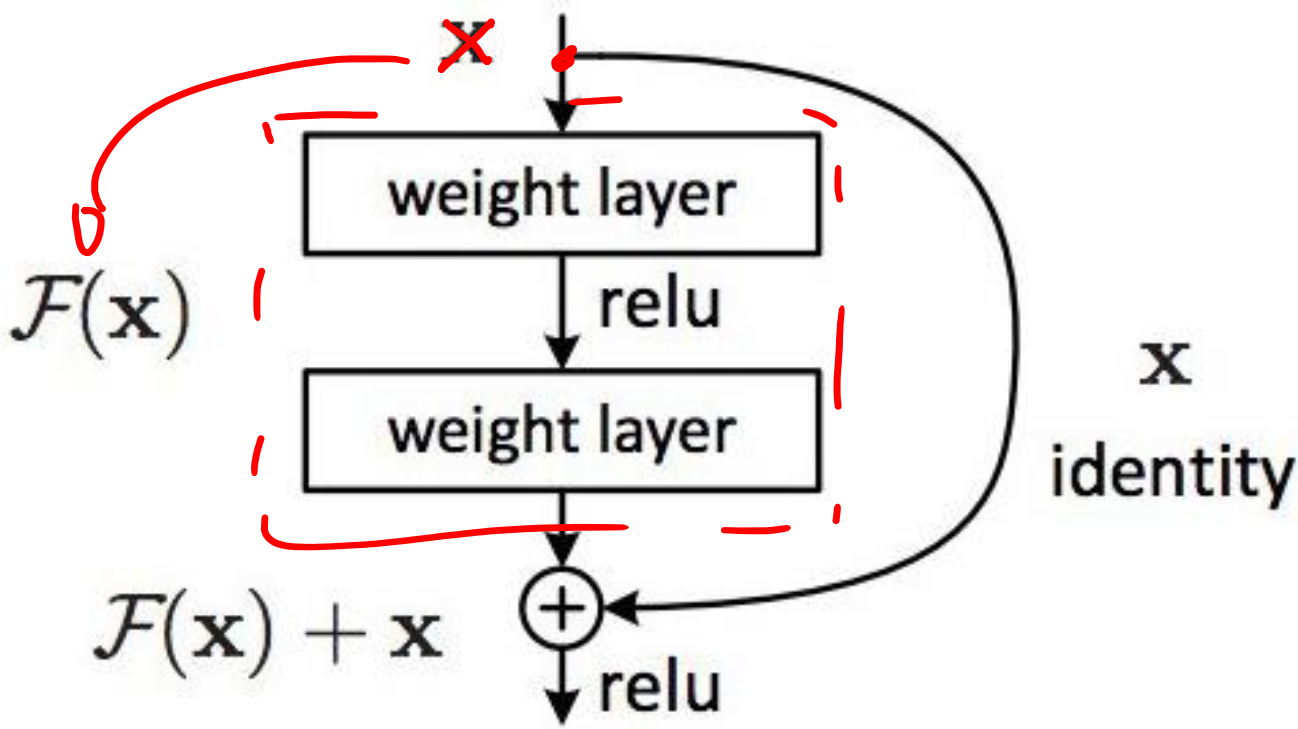
Then, the identity function is not easy to learn!



# ResNet: Very deep by residual connections

**Adding an "identity shortcut connection" :**

- helps in **mitigating the vanishing gradient problem** and enables deeper architectures
- Does not add parameters
- In case the **previous network was optimal**, the **weights to be learned goes to zero** and information is propagated by the identity
- The network can still be trained through back-propagation

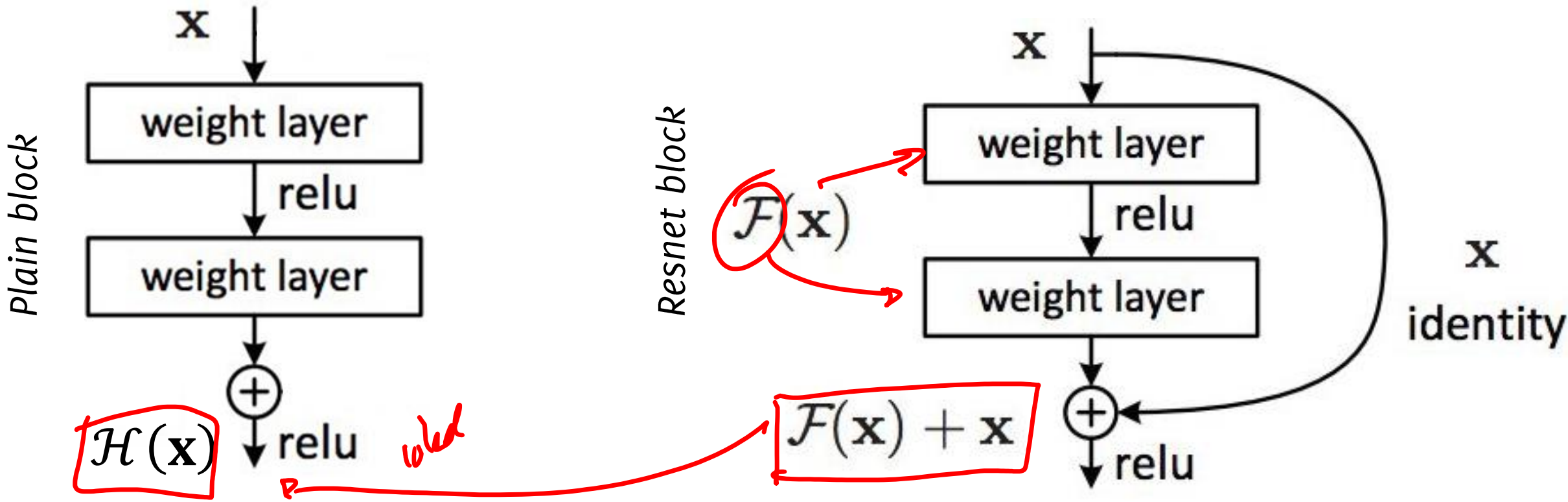



Kaiming He, et al. "Deep Residual Learning for Image Recognition" CVPR 2016

# ResNet: Very deep by residual connections

**Intuition: force the network to learn a different task** in each block.

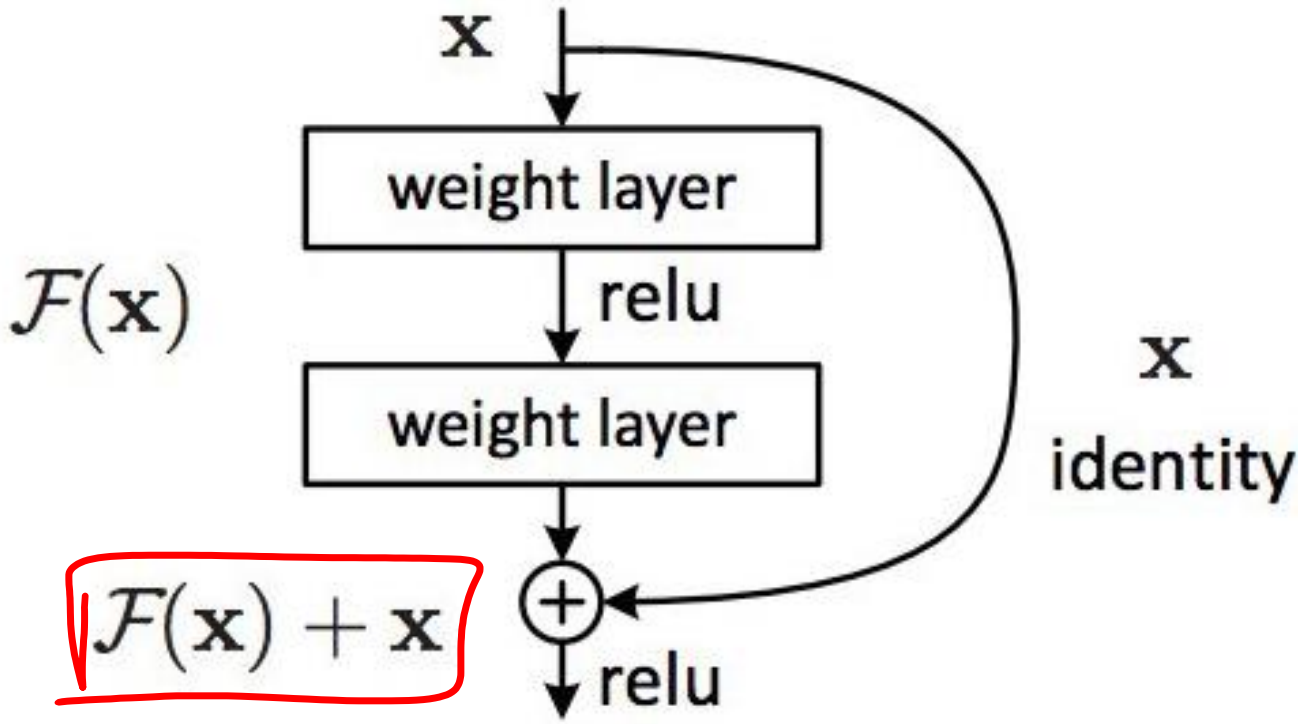
If $\mathcal{H}(\mathbf{x})$ is the **ideal mapping** to be learned from a plain network, by skip connections **we force the network to learn** $\mathcal{F}(\mathbf{x}) = \mathcal{H}(\mathbf{x}) - \mathbf{x}$.




Kaiming He, et al. "Deep Residual Learning for Image Recognition" CVPR 2016

# ResNet: Very deep by residual connections

- $\mathcal{F}(\mathbf{x})$ is called the **residual** (something to add on top of identity), which turns to be easier to train in deep networks.
- **Weights in between the skip connection can be used to learn a «delta», a residual i.e., $F(\mathbf{x})$ to improve over the solution that can be achieved by a shallow network**
- The weights (convolutional layers) are such that to **preserve dimension depth-wise or are re-arranged by 1x1 convolutions**

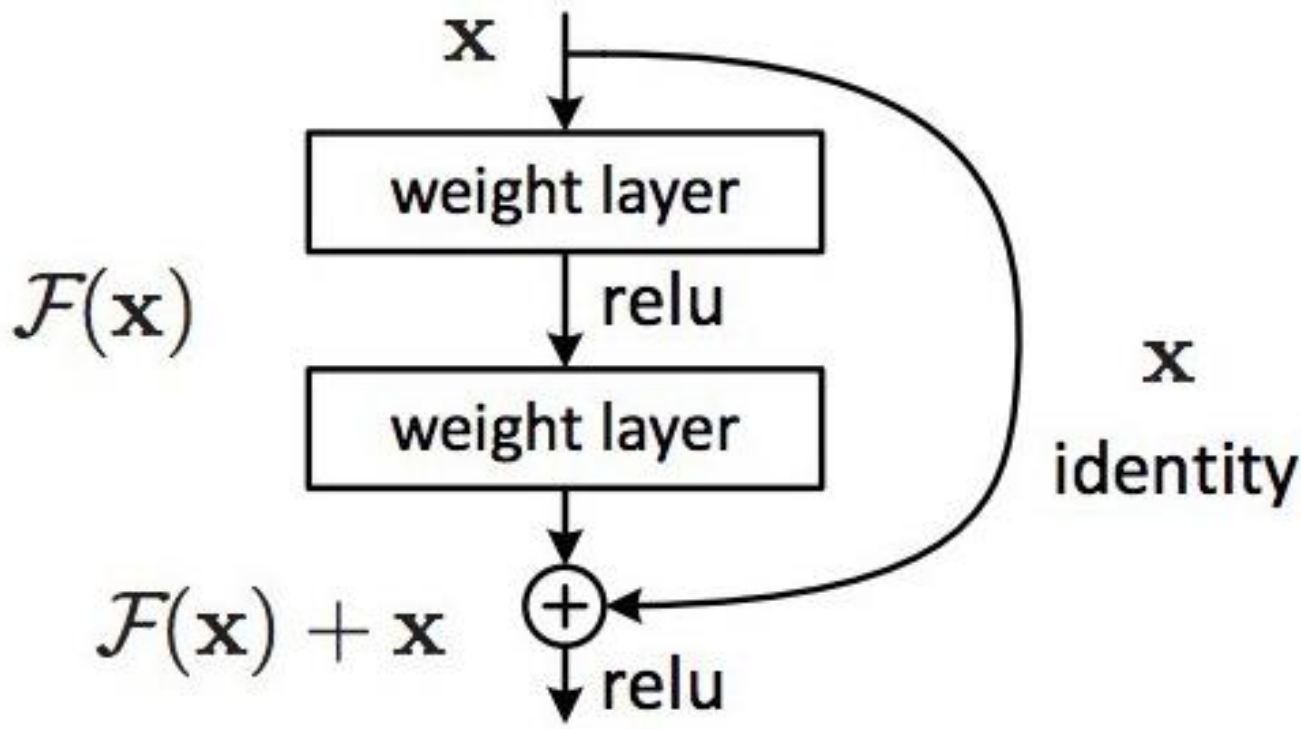



Kaiming He, et al. "Deep Residual Learning for Image Recognition“ CVPR 2016

# ResNet (2015)

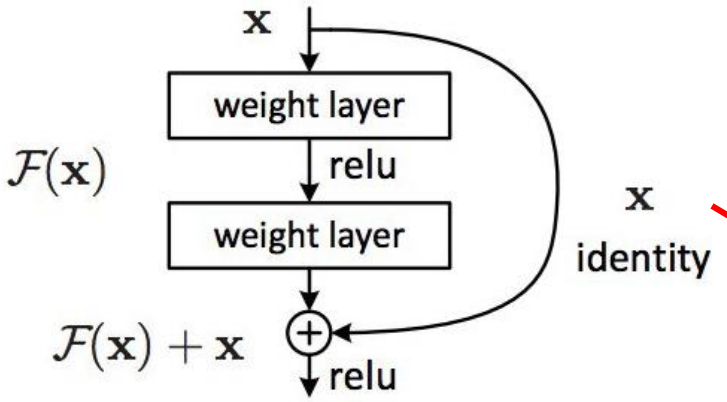
The rationale behind adding this identity mapping is that:

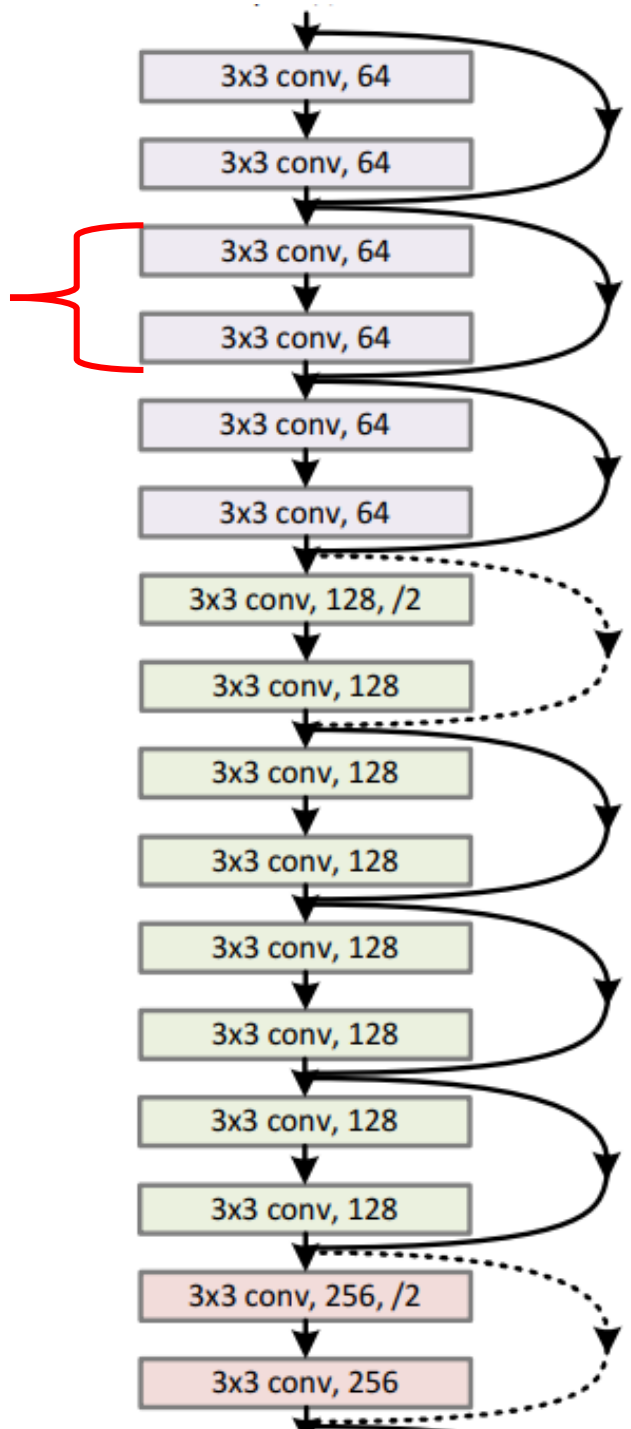
- It is easier for the following layers to learn features on top of the input value
- In practice the weights between the identity mapping would fail by themselves at learning the identity function to transfer the input to the output
- The performance achieved by resNet suggests that **probably most of thee deep layers have to be close to the identity**!




Kaiming He, et al. "Deep Residual Learning for Image Recognition“ CVPR 2016

# ResNet (2015)




The ResNet is a stack of 152 layers of this module

The network alternates

- some spatial **pooling by convolution with stride 2**
- **Doubling the number of filters**

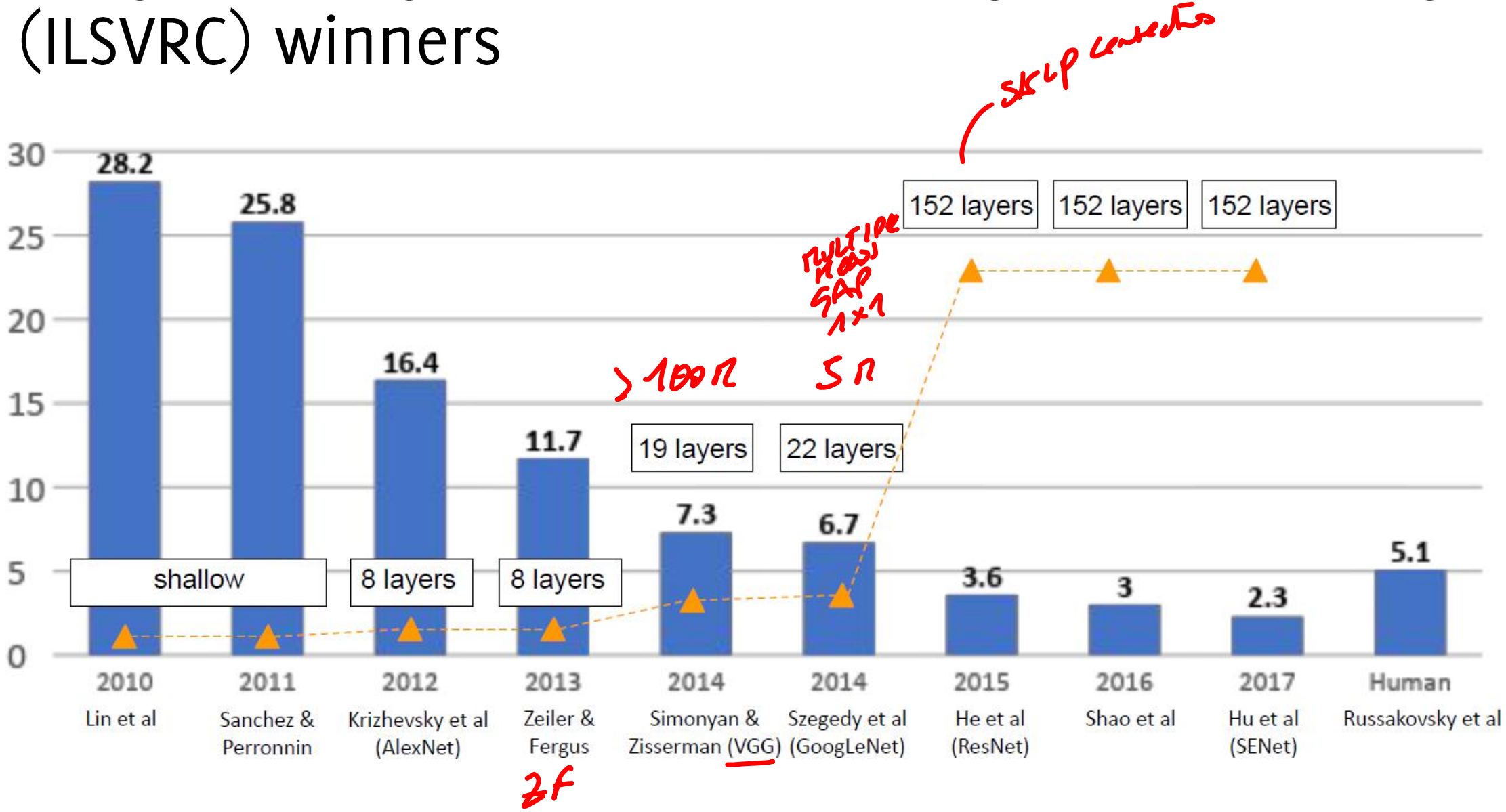
At the beginning there is a **convolutional layer**

At the end, no FC but just a **GAP to be fed in the final softmax**

Deeper networks are able to achieve lower errors as expected

**Kaiming He, et al. "Deep Residual Learning for Image Recognition" CVPR 2016**

# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



# Aggregated Residual Transformations for Deep Neural Networks

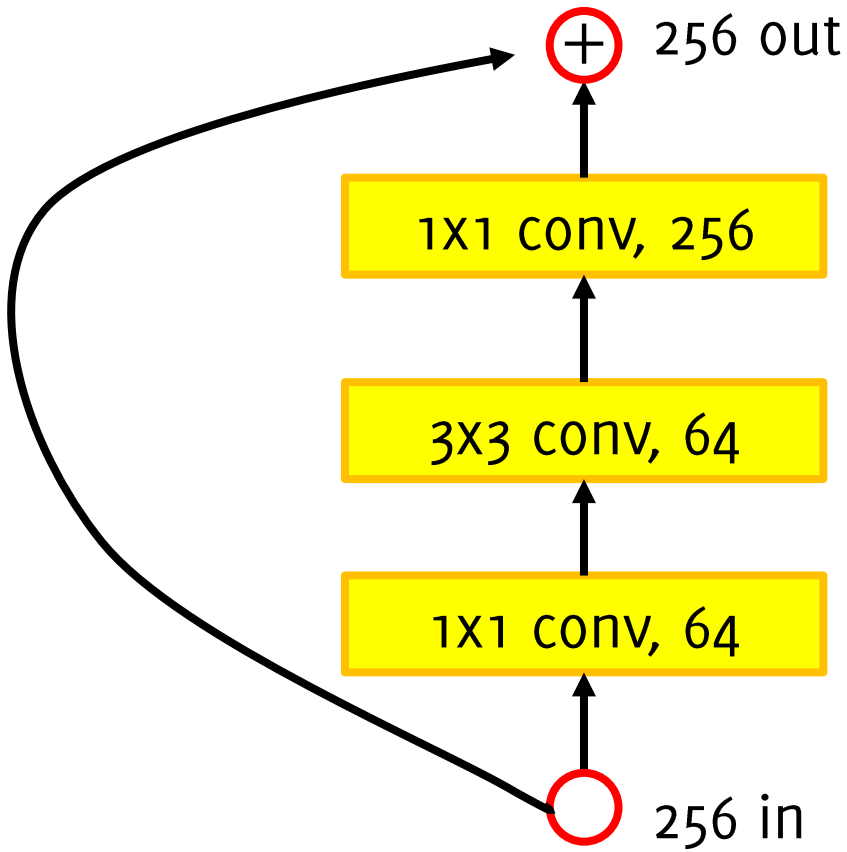
Saining Xie[1] Ross Girshick[2] Piotr Dollár[2] Zhuowen Tu[1] Kaiming He[2]

[1]UC San Diego [2]Facebook AI Research

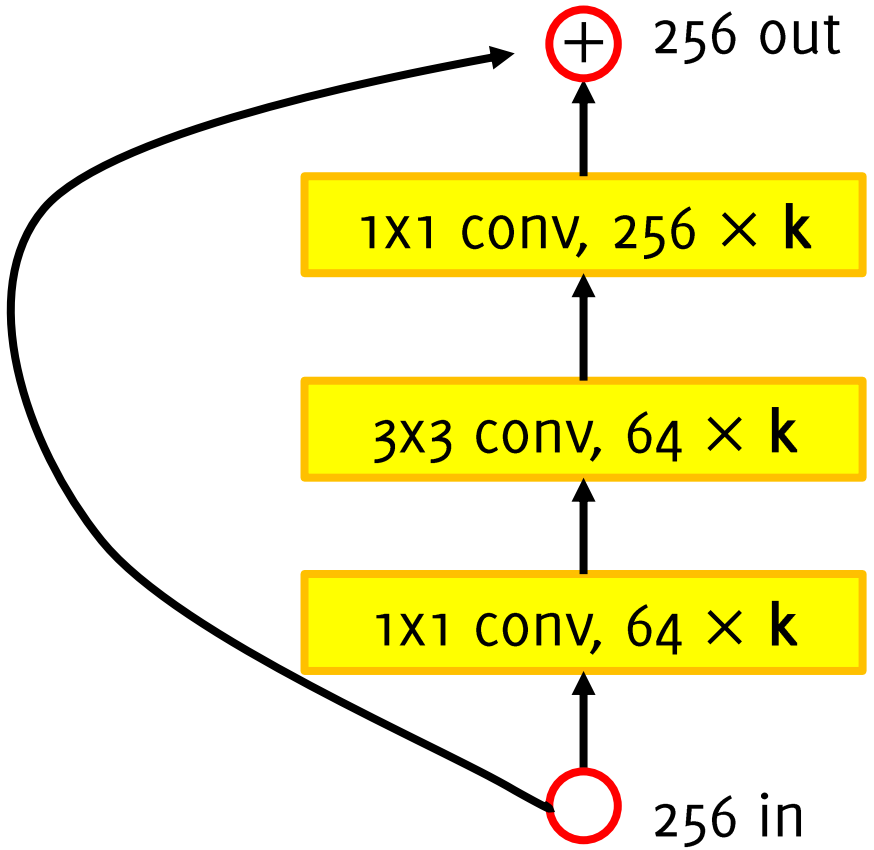
{s9xie,ztu}@ucsd.edu {rbg,pdollar,kaiminghe}@fb.com

# Wide Resnet

**ResNet Module**



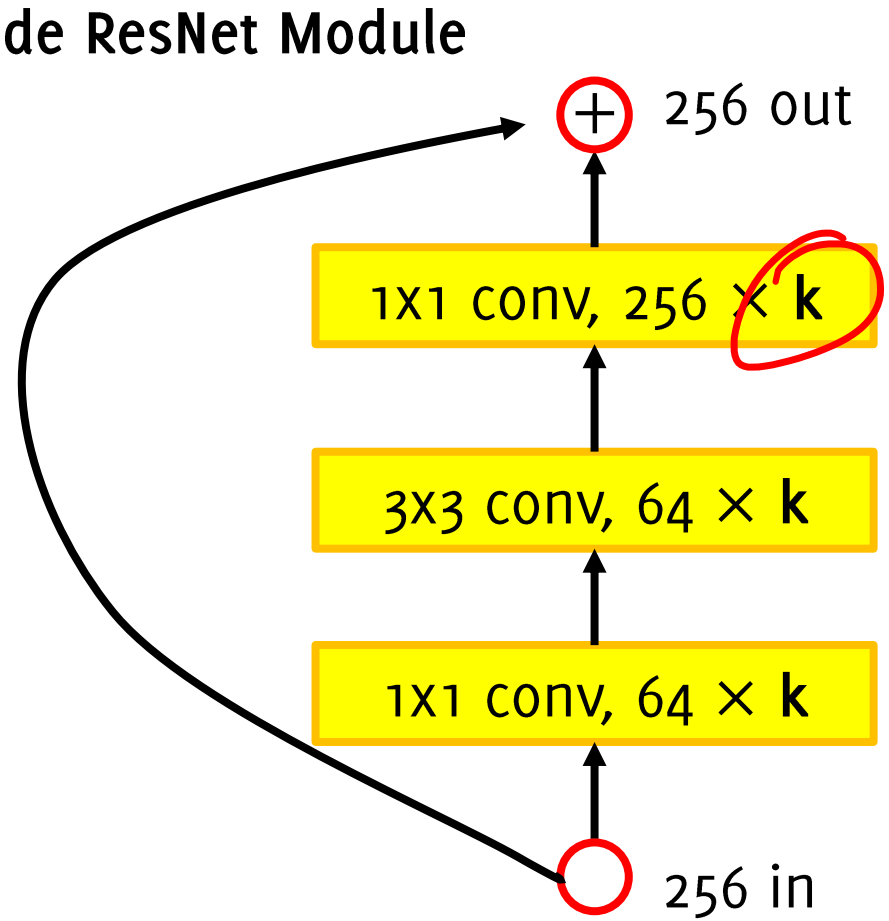

**Wide ResNet Module**




Xie et al "Aggregated Residual Transformations for Deep Neural Networks", CVPR 2017

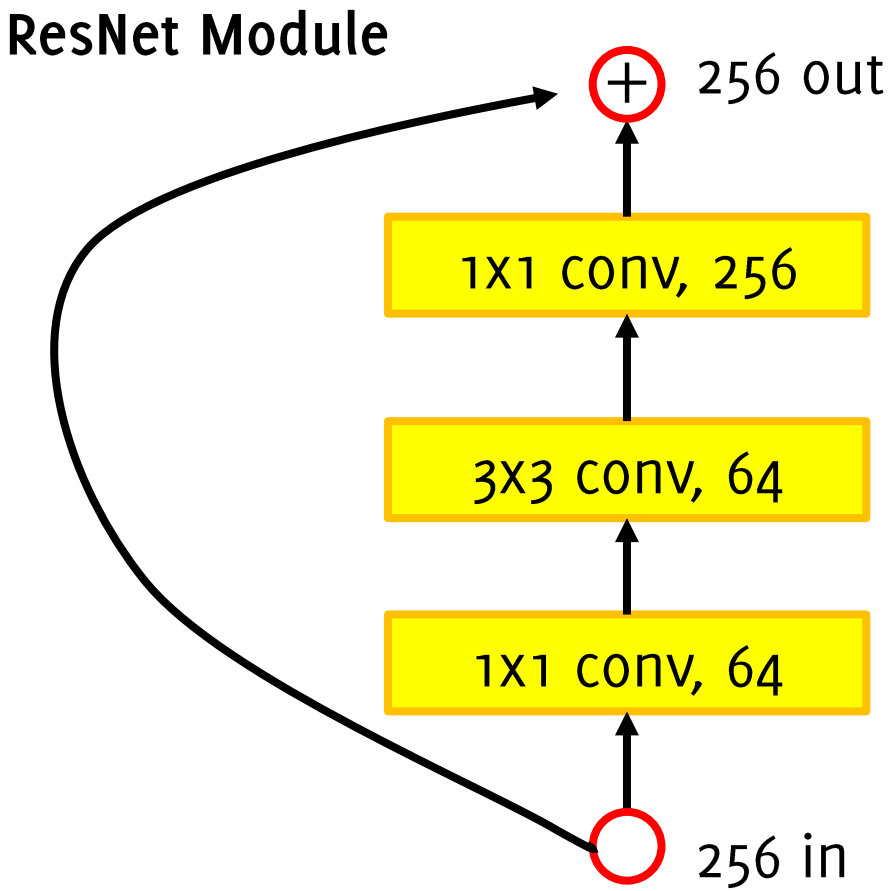
# Wide Resnet

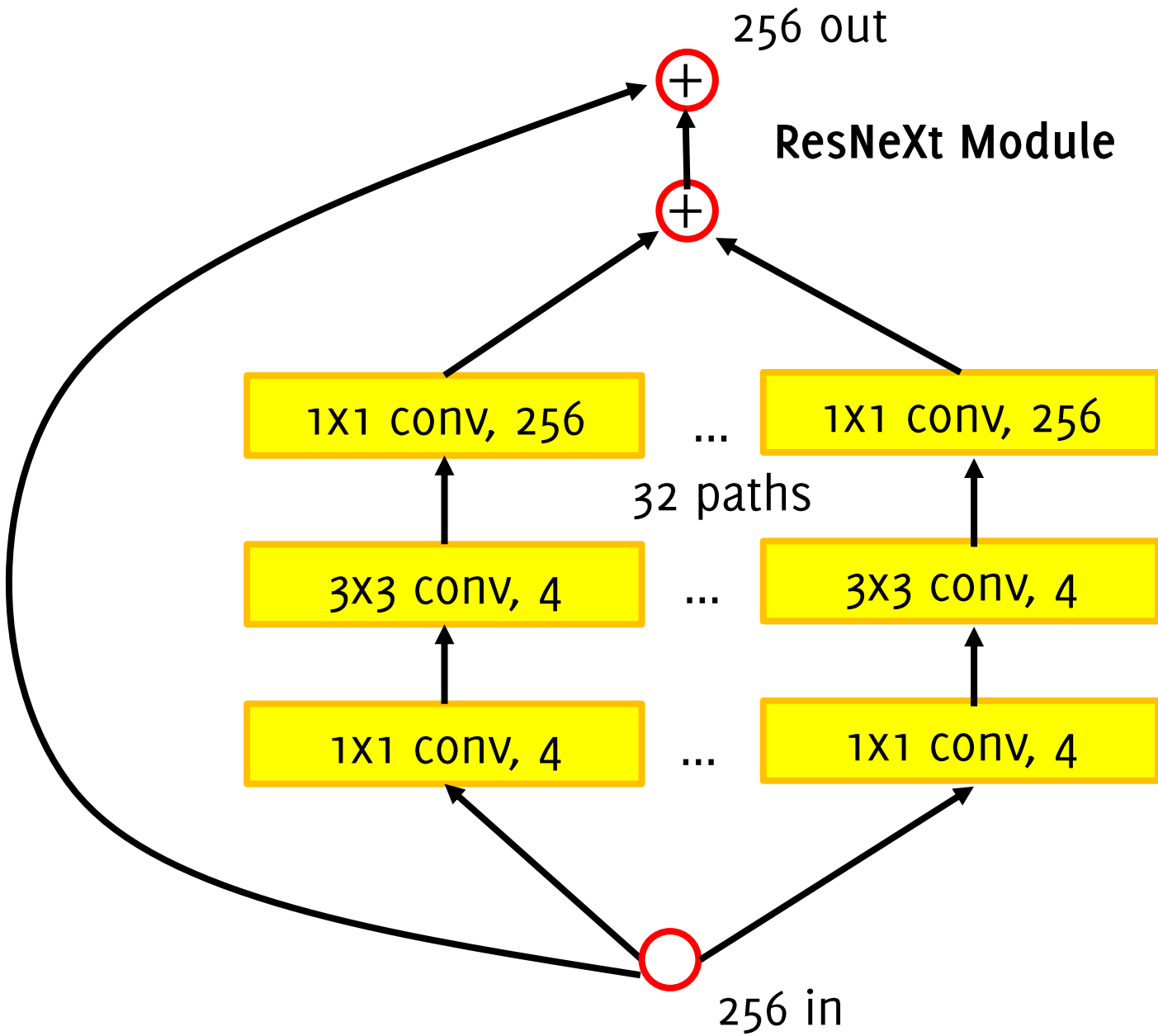
- Use wider residual blocks (F x k filters instead of F filters in each layer)
- 50-layer wide ResNet outperforms 152-layer original ResNet
- Increasing width instead of depth more computationally efficient (parallelizable)




Xie et al "Aggregated Residual Transformations for Deep Neural Networks", CVPR 2017

# ResNeXt



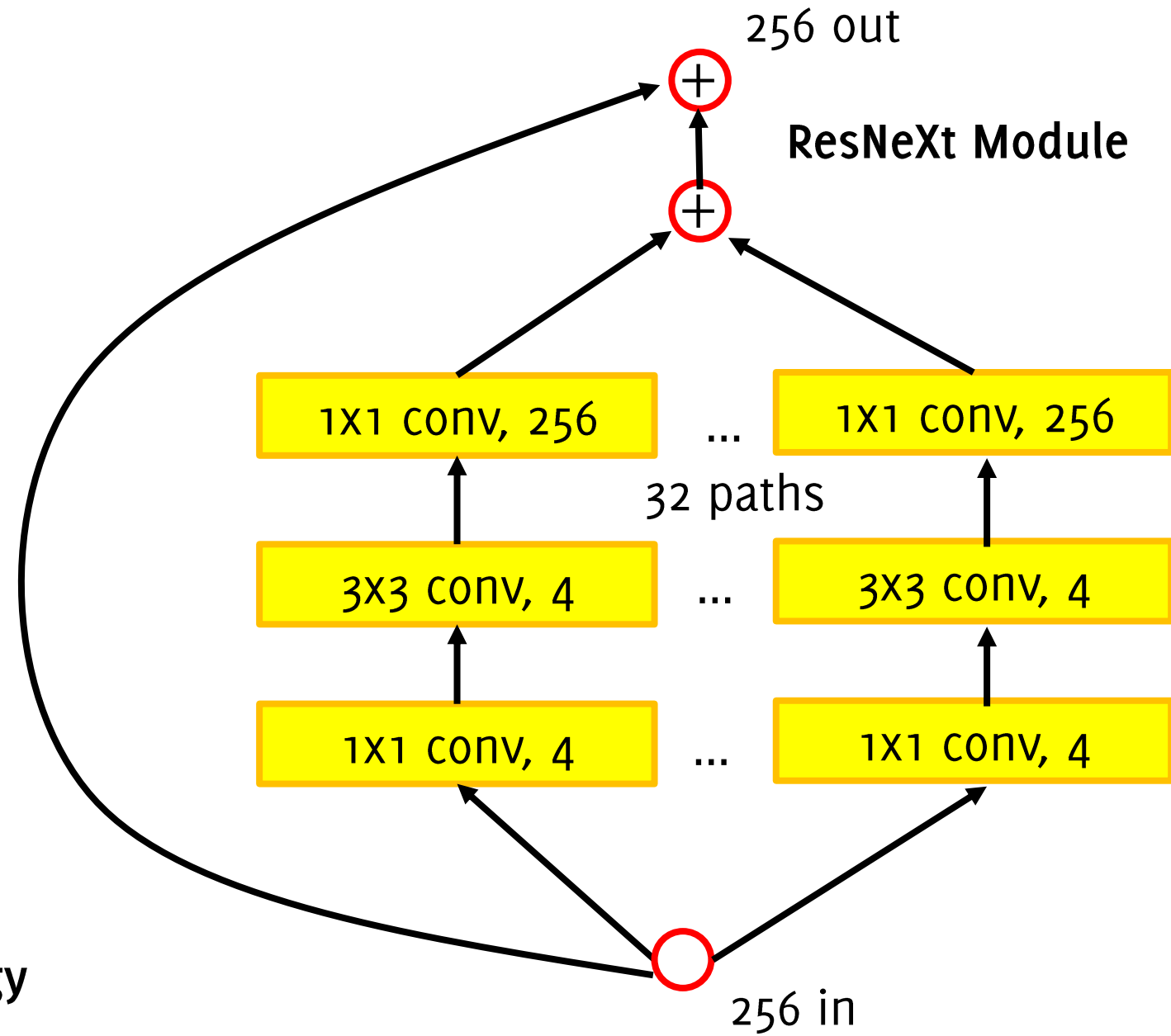

Xie et al "Aggregated Residual Transformations for Deep Neural Networks", CVPR 2017

# ResNeXt

**Widen the ResNet module** by adding multiple pathways in parallel (previous wide Resnet was just increasing the number of filters and showing it achieves similar perf. with fewer blocks)

**Similar to inception** module where the **activation maps are being processed in parallel**

**Different from inception module, all the paths share the same topology**




Xie et al "Aggregated Residual Transformations for Deep Neural Networks", CVPR 2017

# Densely Connected Convolutional Networks

Gao Huang*
Cornell University
gh349@cornell.edu

Zhuang Liu*
Tsinghua University
liuzhuang13@mails.tsinghua.edu.cn

Laurens van der Maaten
Facebook AI Research
lvdmaaten@fb.com

Kilian Q. Weinberger
Cornell University
kqw4@cornell.edu

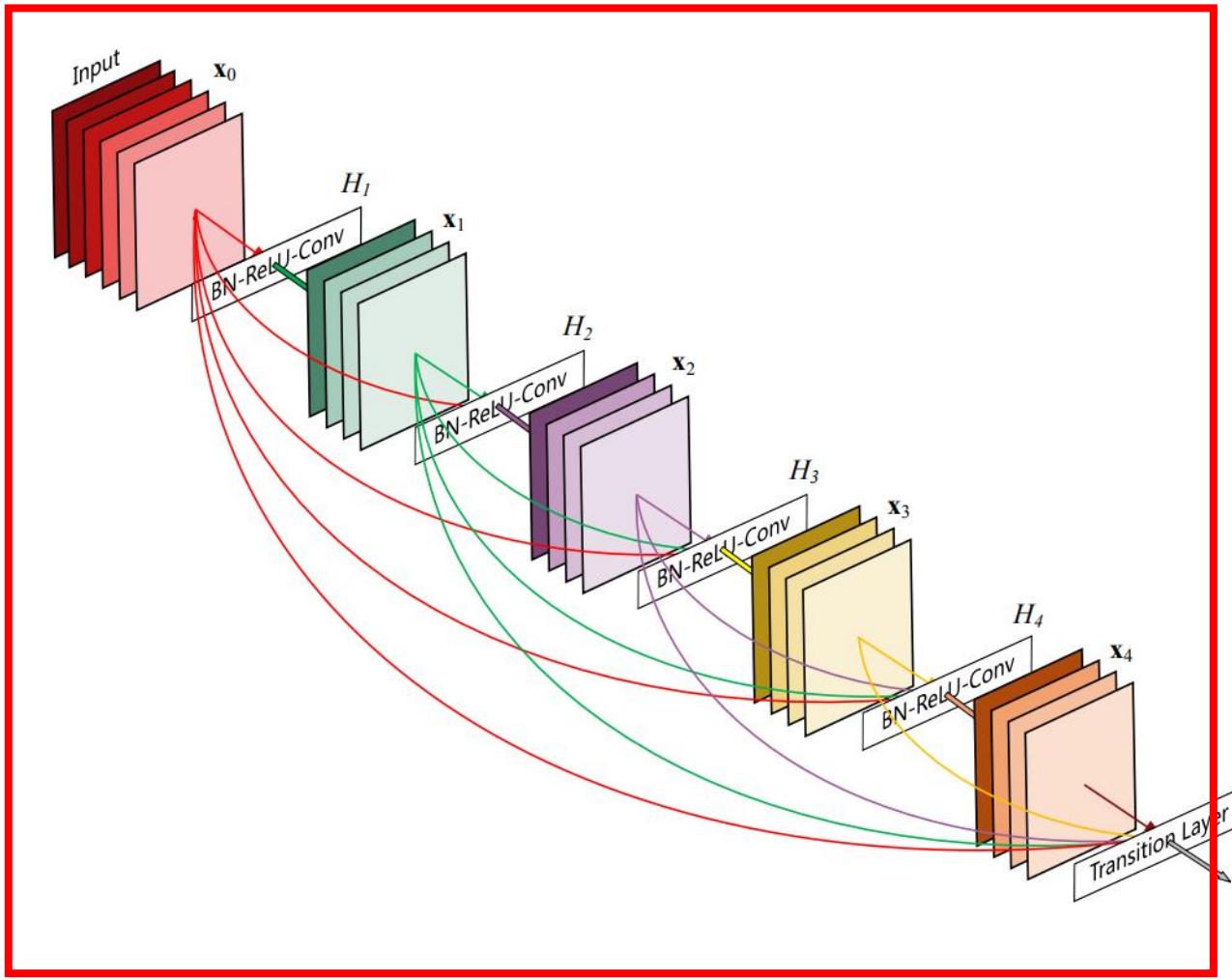
# DenseNet

In **each block** of a DenseNet, **each convolutional layer takes as input the output of the previous layers**

Short connections between convolutional layers of the network

Each layer is connected to every other layer in a feed-forward fashion

**Dense block**



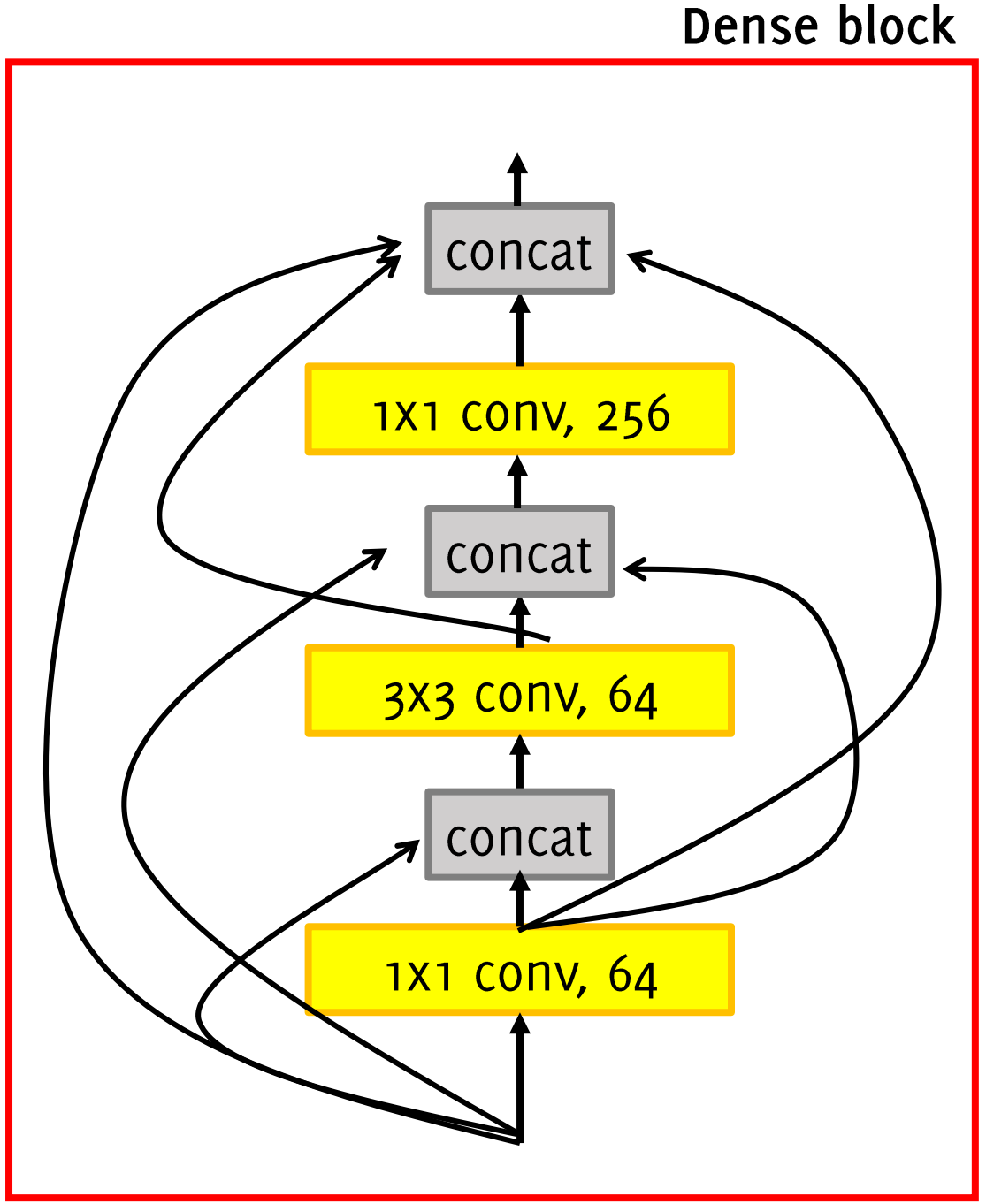

Huang G. et al, "Densely Connected Convolutional Network" CVPR 2017

# DenseNet

In each block a DenseNet, each **convolutional layer takes as input the output of the previous layers**

Each layer is connected to every other layer in a feed-forward fashion

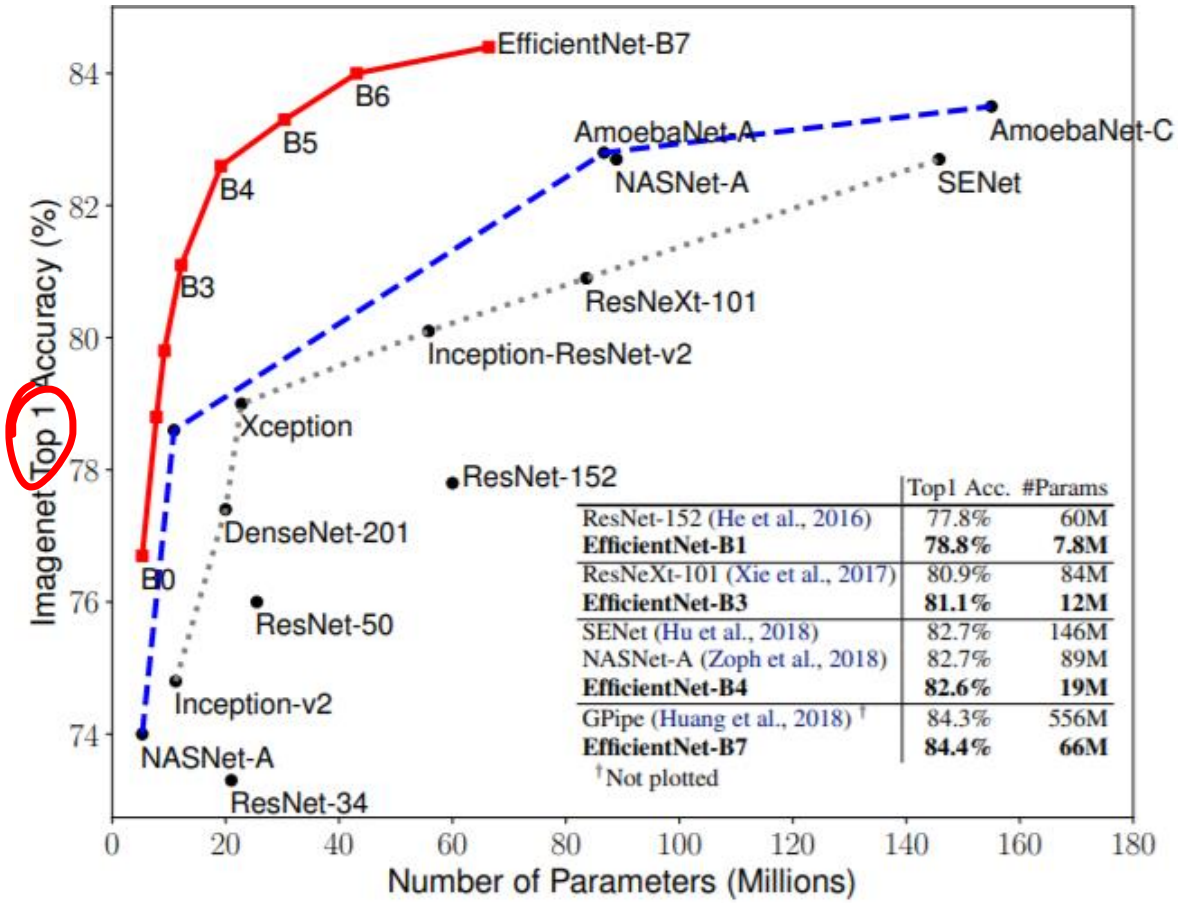
This alleviates vanishing gradient problem, promotes feature re-use since each feature is spread through the network




Huang G. et al, "Densely Connected Convolutional Network" CVPR 2017

# EfficientNet:

*We propose a new scaling method that uniformly scales all dimensions of depth/width/resolution using a simple yet highly effective compound coefficient*



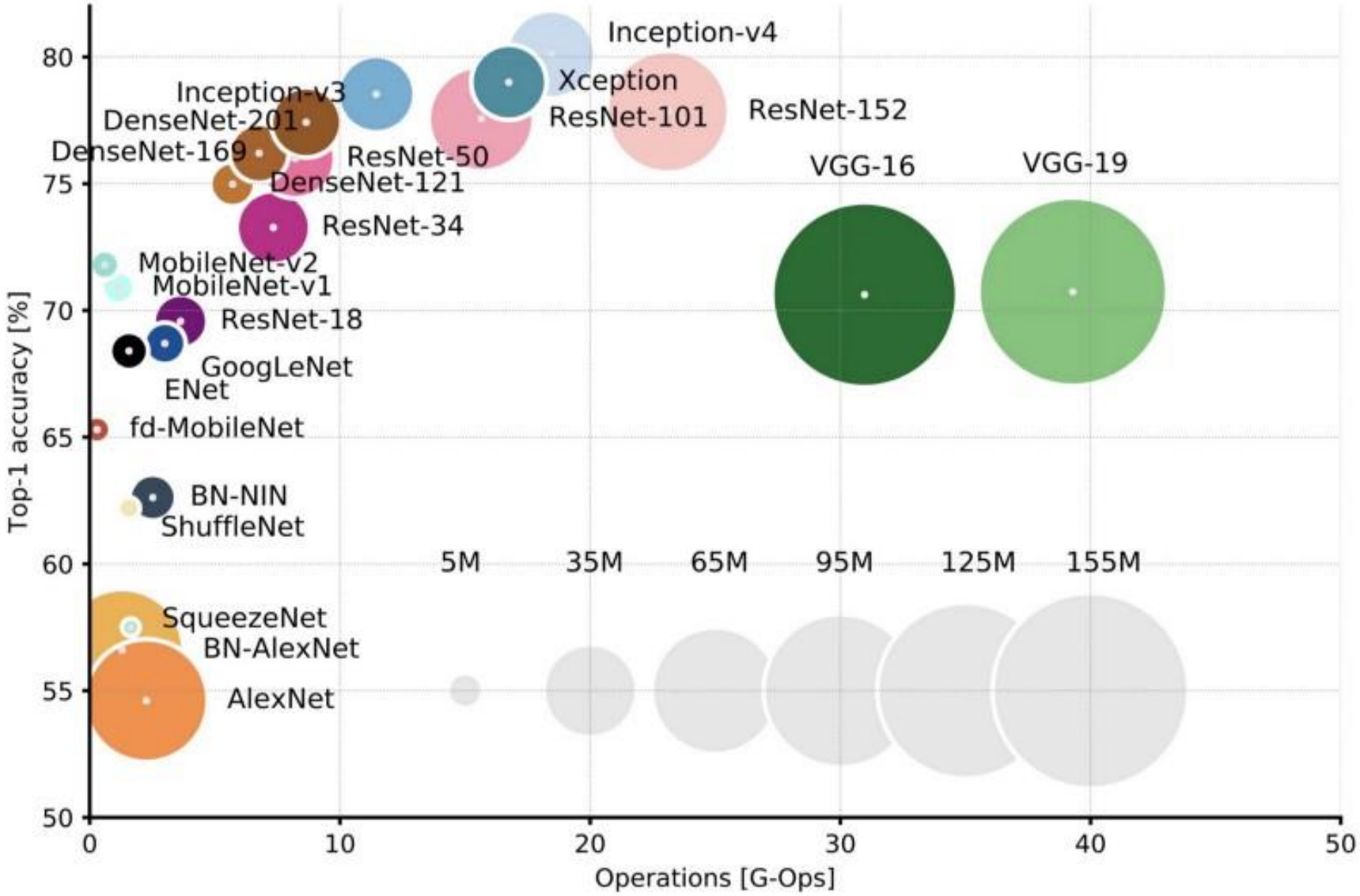

| | Top1 Acc. | #Params |
|---|---|---|
| ResNet-152 (He et al., 2016) | 77.8% | 60M |
| **EfficientNet-B1** | **78.8%** | **7.8M** |
| ResNeXt-101 (Xie et al., 2017) | 80.9% | 84M |
| **EfficientNet-B3** | **81.1%** | **12M** |
| SENet (Hu et al., 2018) | 82.7% | 146M |
| NASNet-A (Zoph et al., 2018) | 82.7% | 89M |
| **EfficientNet-B4** | **82.6%** | **19M** |
| GPipe (Huang et al., 2018) † | 84.3% | 556M |
| **EfficientNet-B7** | **84.4%** | **66M** |

† Not plotted

*Figure 1.* **Model Size vs. ImageNet Accuracy.** All numbers are for single-crop, single-model. Our EfficientNets significantly outperform other ConvNets. In particular, EfficientNet-B7 achieves new state-of-the-art 84.4% top-1 accuracy but being 8.4x smaller and 6.1x faster than GPipe. EfficientNet-B1 is 7.6x smaller and 5.7x faster than ResNet-152. Details are in Table 2 and 4.

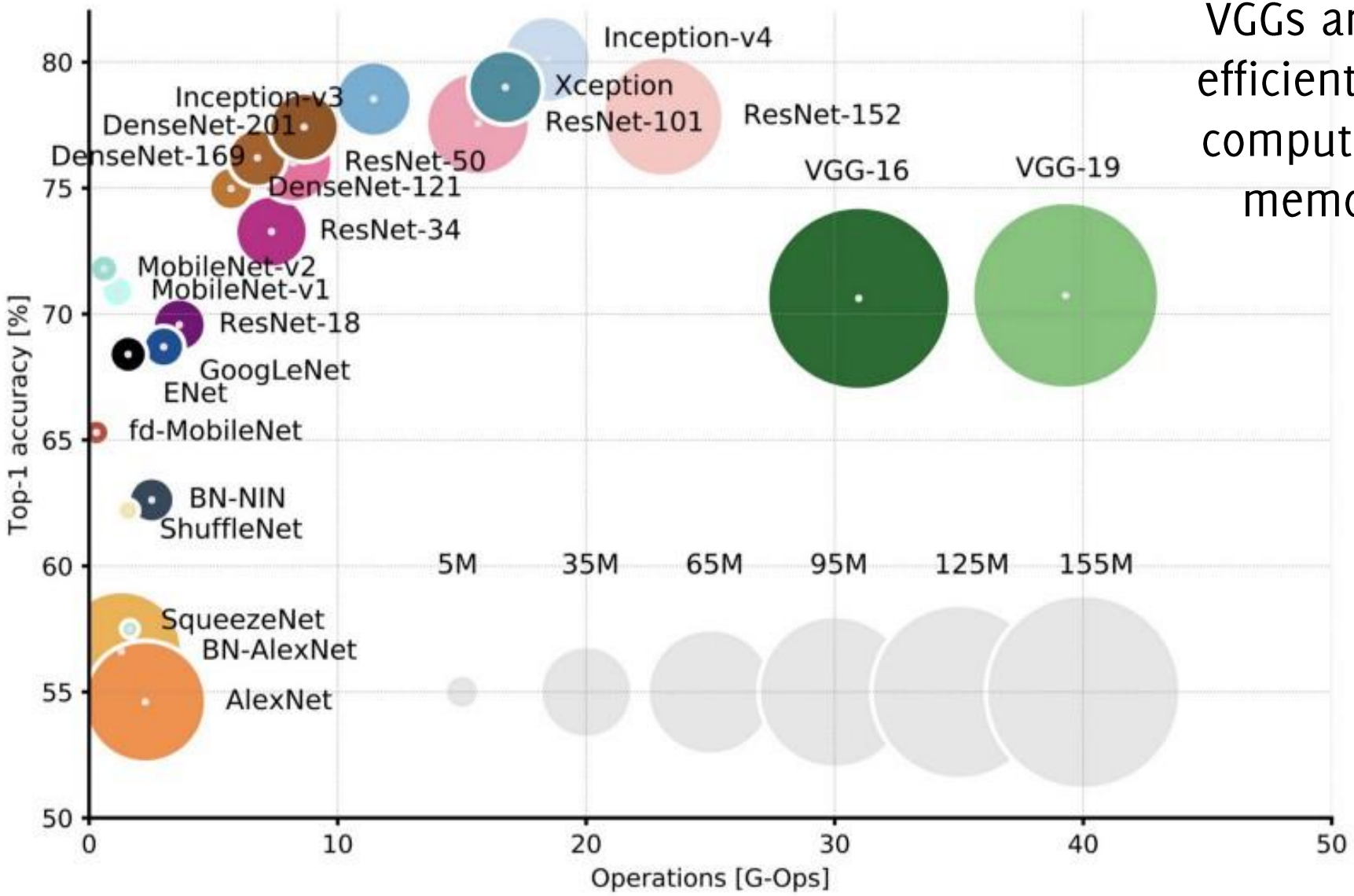
Tan, Mingxing, and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." *International Conference on Machine Learning*. PMLR, 2019.

# Comparison




Canziani, A, Paszke A., Culurciello E.. "An analysis of deep neural network models for practical applications." *arXiv preprint* (2016).

# Comparison

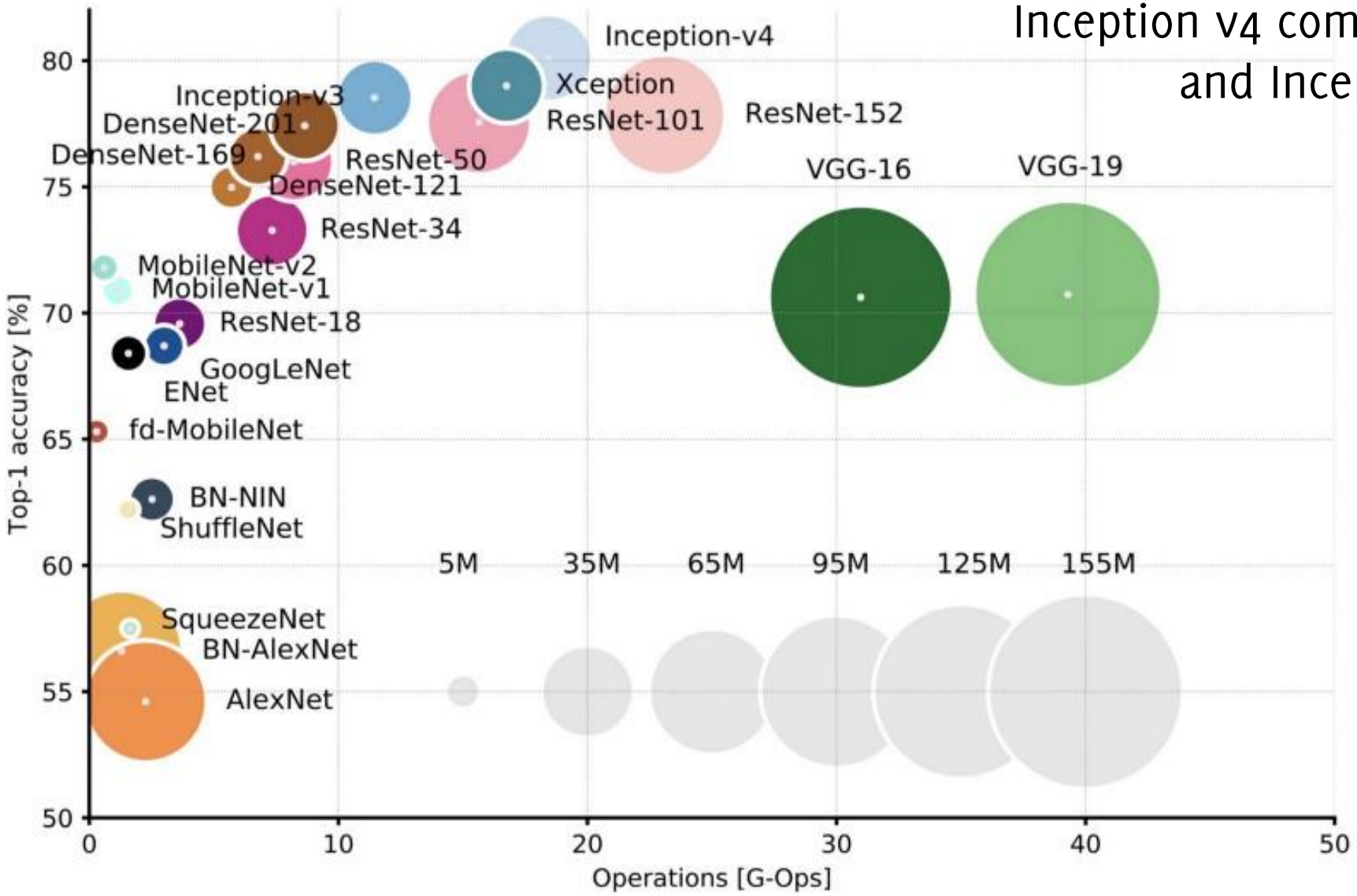



VGGs are the least efficient: very large computational and memory usage

Canziani, A, Paszke A., Culurciello E.. "An analysis of deep neural network models for practical applications." *arXiv preprint* (2016).

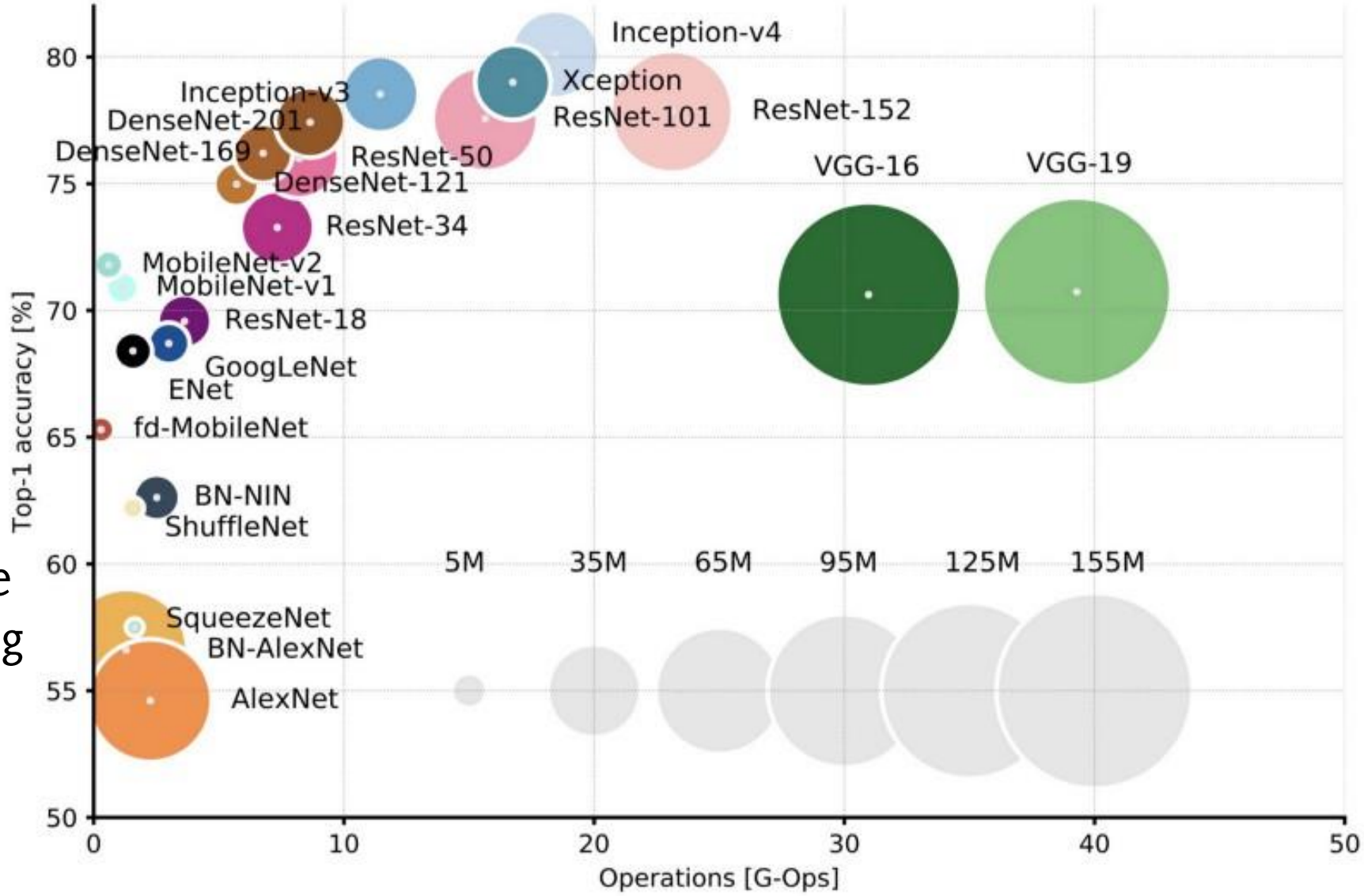
# Comparison

Inception models are the most efficient and best performing
Inception v4 combines ResNet and Inception




Canziani, A, Paszke A., Culurciello E.. "An analysis of deep neural network models for practical applications." *arXiv preprint* (2016).

# Comparison

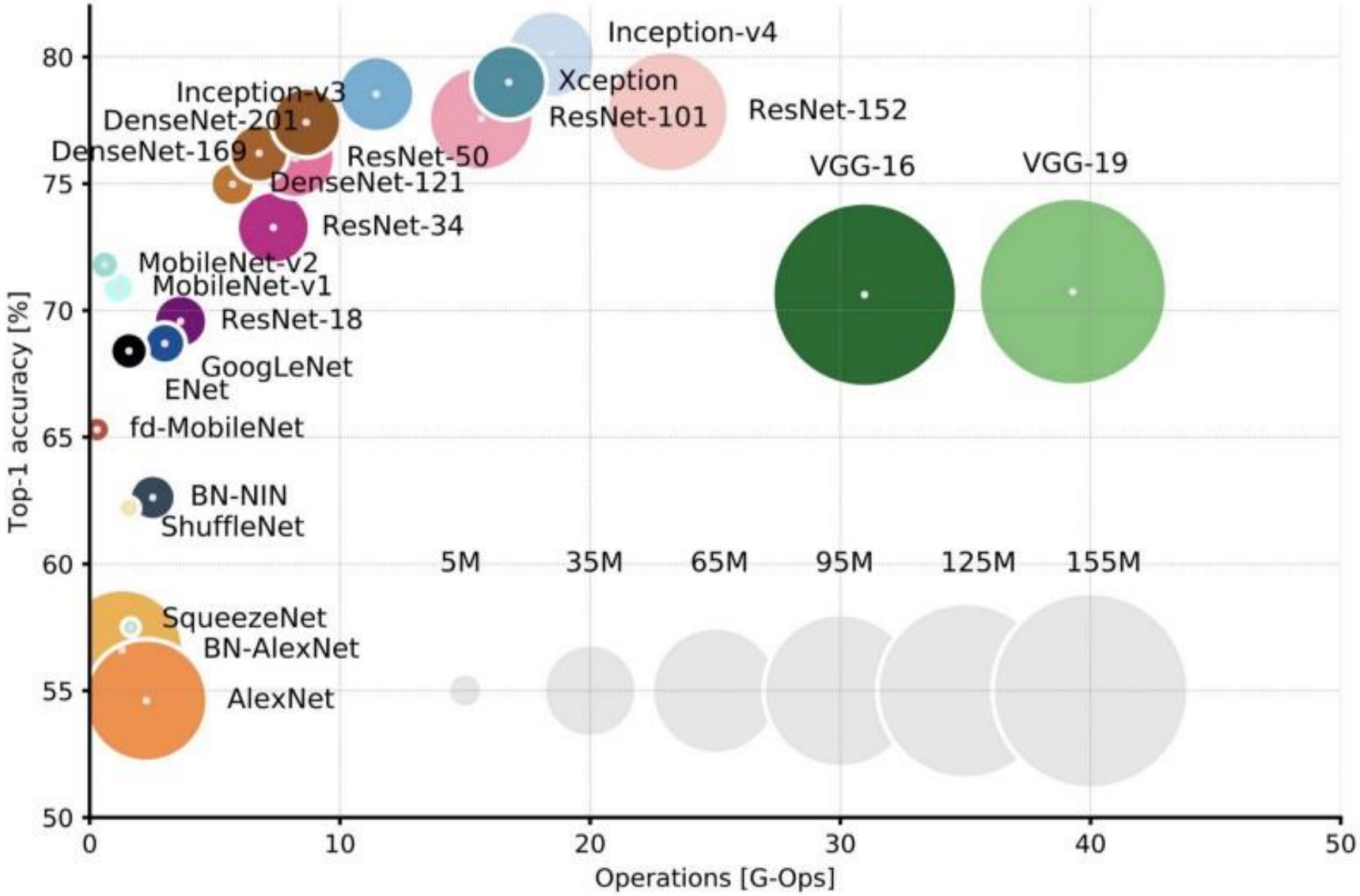



AlexNet are the least performing and not particularly efficient

Canziani, A, Paszke A., Culurciello E.. "An analysis of deep neural network models for practical applications." *arXiv preprint* (2016).

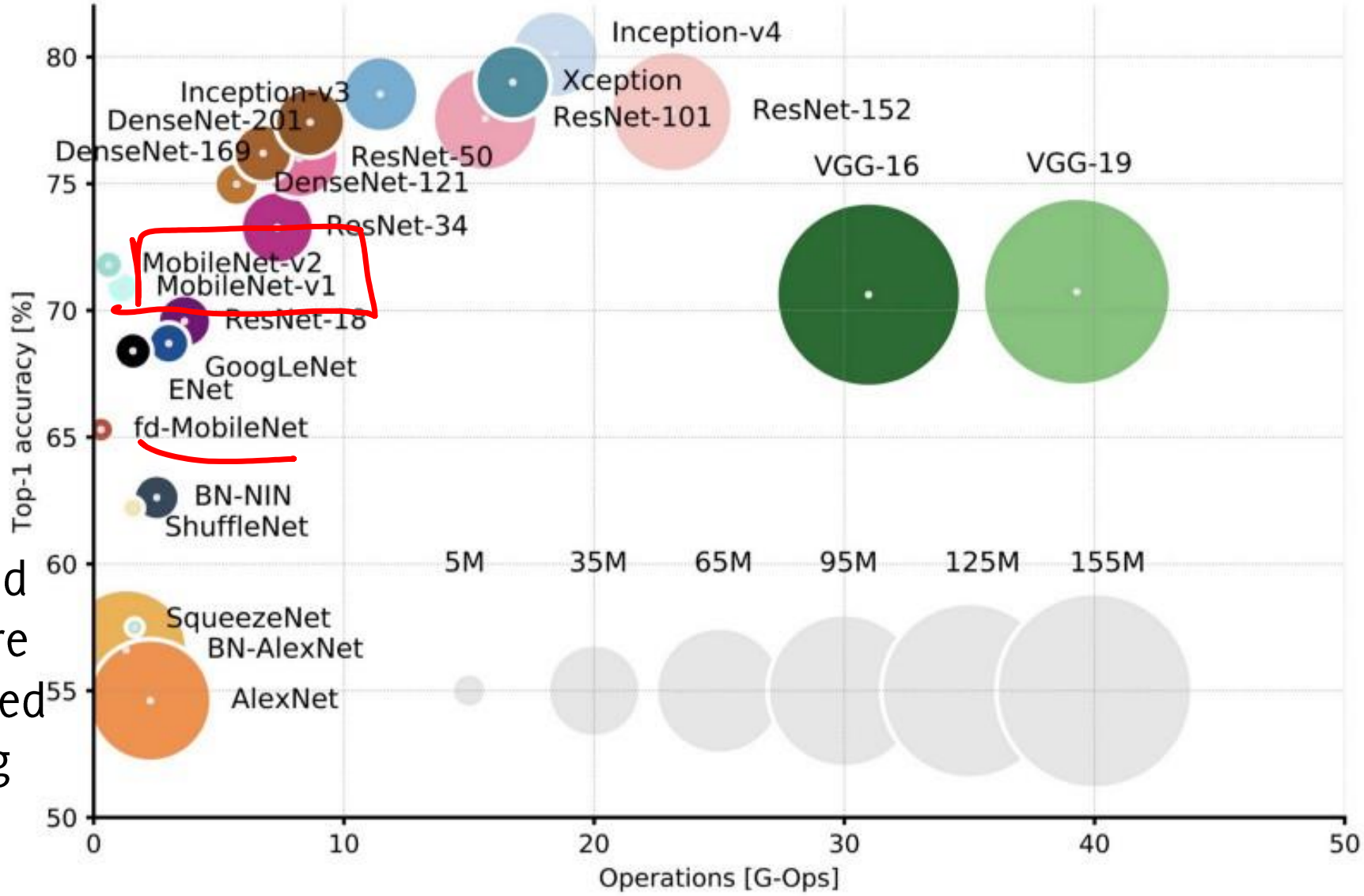
# Comparison

Inception-v4:
Resnet + Inception




Canziani, A, Paszke A., Culurciello E.. "An analysis of deep neural network models for practical applications." *arXiv preprint* (2016).

# Comparison




Mobile net and squeezenet are instead designed for improving efficiency

Canziani, A, Paszke A., Culurciello E.. "An analysis of deep neural network models for practical applications." *arXiv preprint* (2016).