

# Convolutional Neural Networks 5

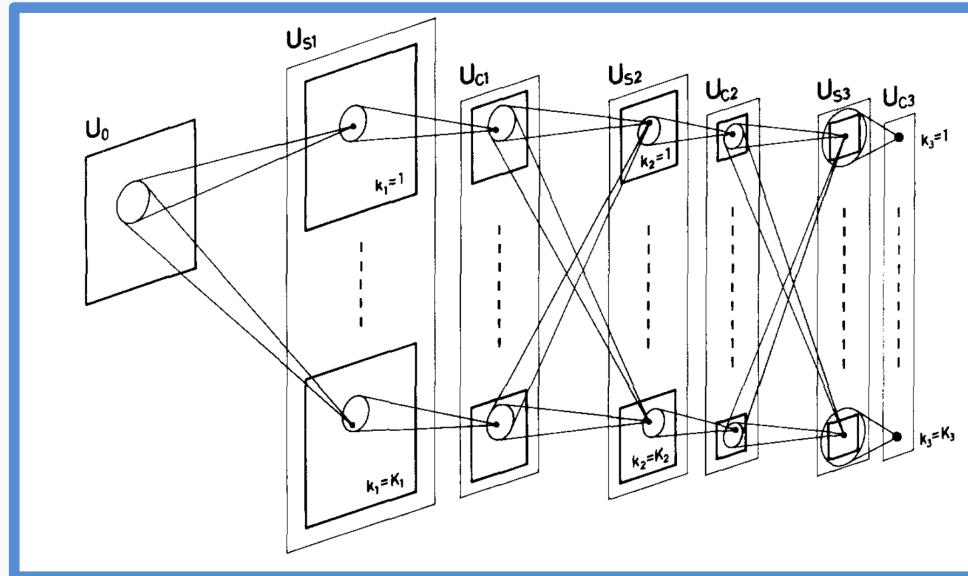
## State Of The Art (SOTA)

Pavlos Protopapas



# Initial ideas

- The first research proposing something similar to a Convolutional Neural Network was authored by [Kunihiko Fukushima](#) in **1980** and was called the **NeoCognitron**.
- Inspired by discoveries on visual cortex of mammals.
- Fukushima applied the NeoCognitron to hand-written character recognition.



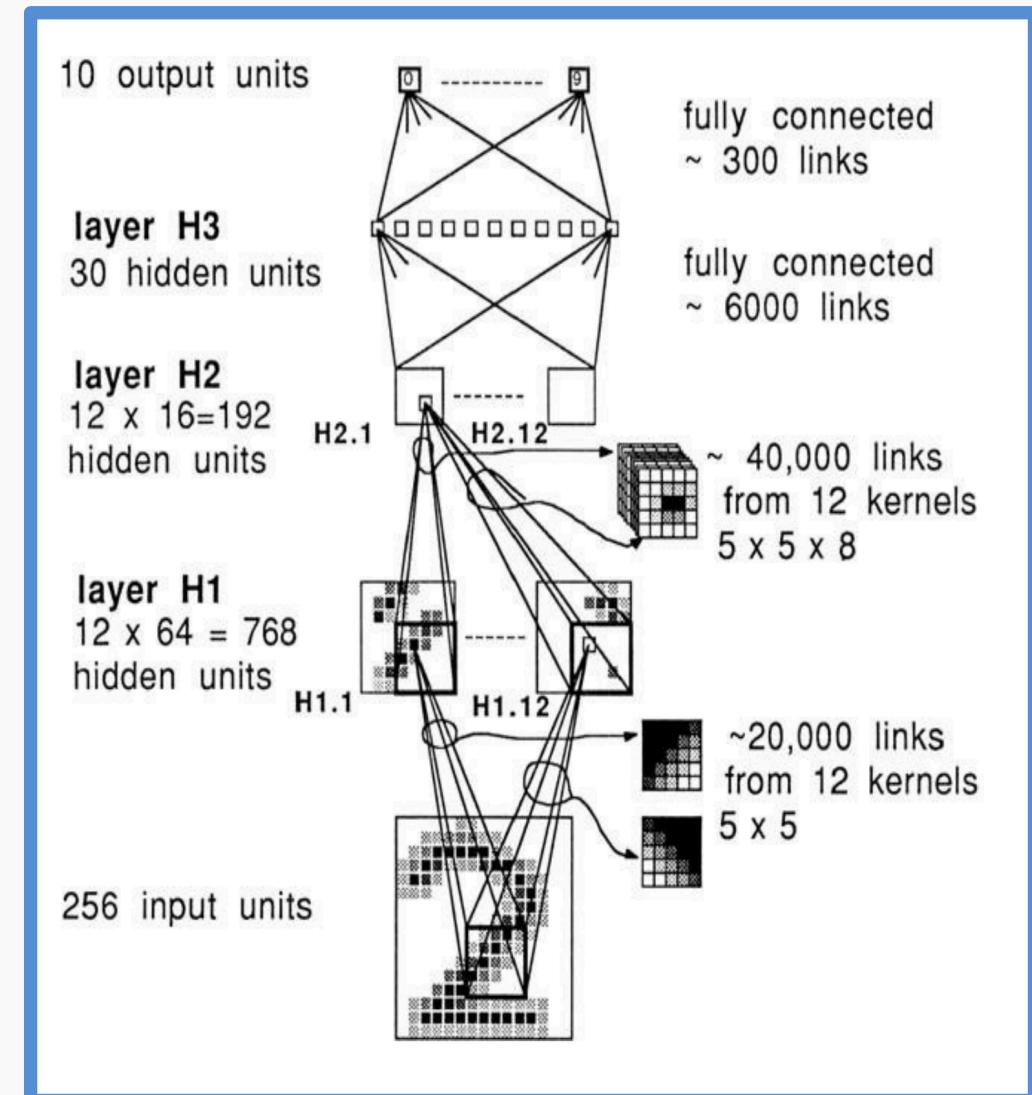
<sup>1</sup> K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4): 93-202, 1980.



# Initial ideas

End of the 80's: several papers advanced the field

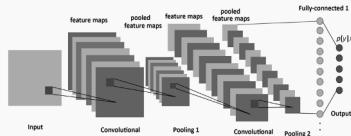
- **Backpropagation** published in French by Yann LeCun in 1985 (independently discovered by other researchers as well)
- TDNN by Waibel et al., 1989 - **Convolutional-like** network trained with backprop.
- Backpropagation applied to handwritten zip code recognition by LeCun et al., 1989



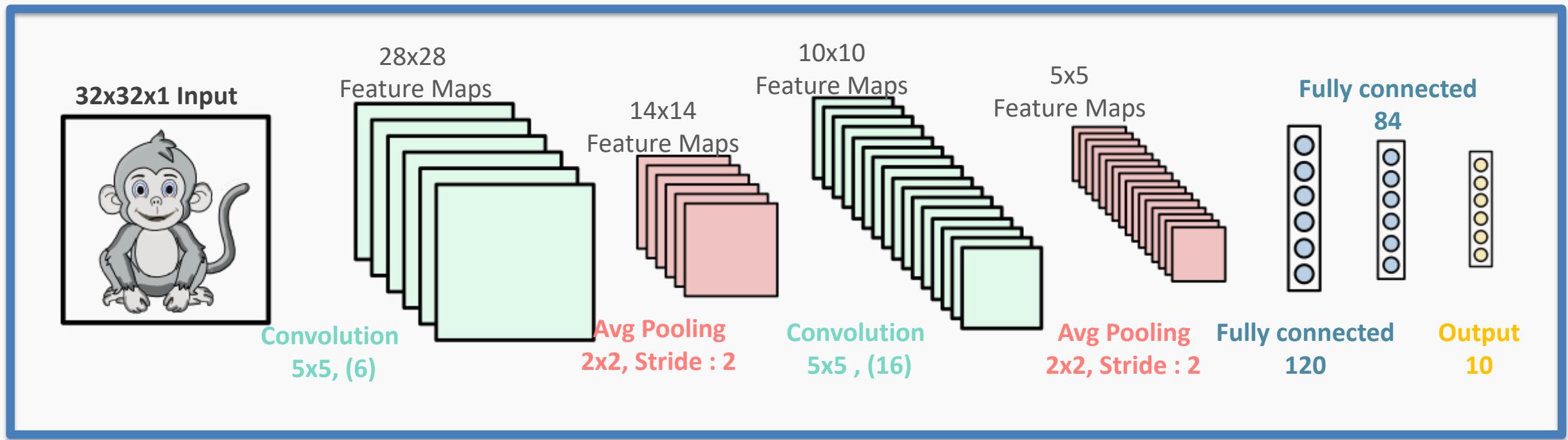
LeCun et al., 1989



# LeNet



- November 1998: LeCun publishes one of his most recognized papers describing a “modern” CNN architecture for document recognition, called LeNet<sup>1</sup>.
- Not his first iteration, this was in fact LeNet-5, but this paper is the commonly cited publication when talking about LeNet.

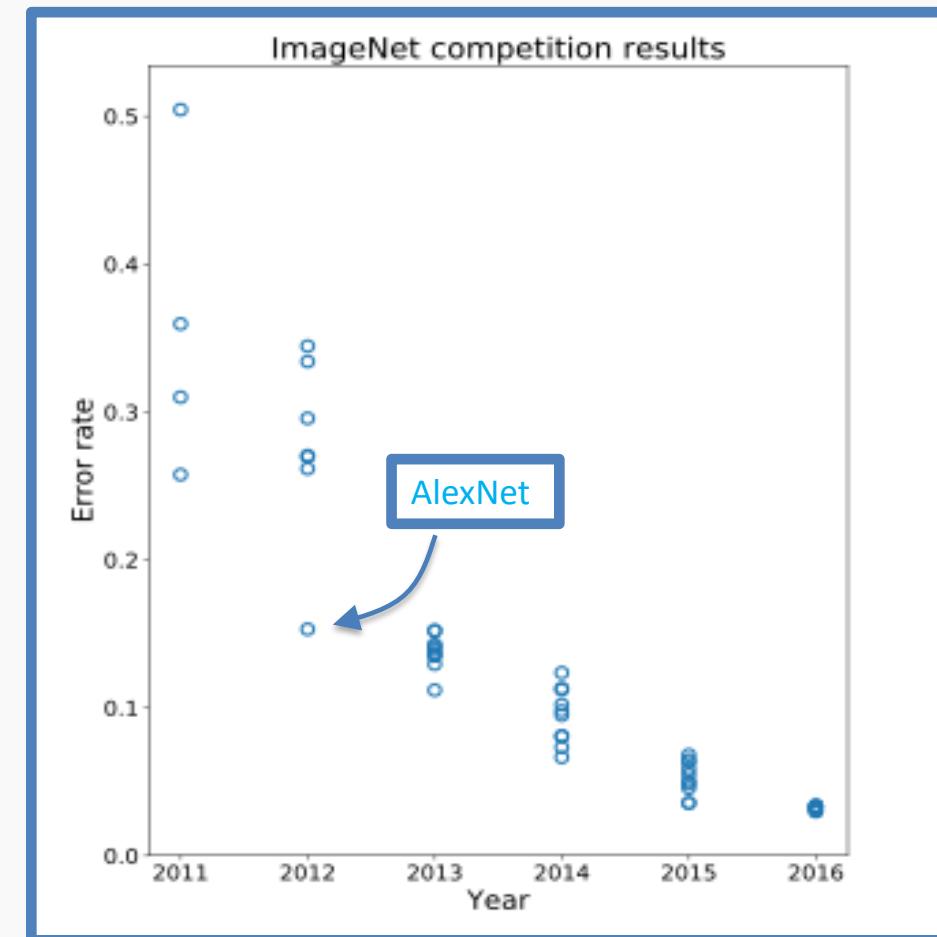


<sup>1</sup> LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE 86.11 (1998): 2278-2324.*



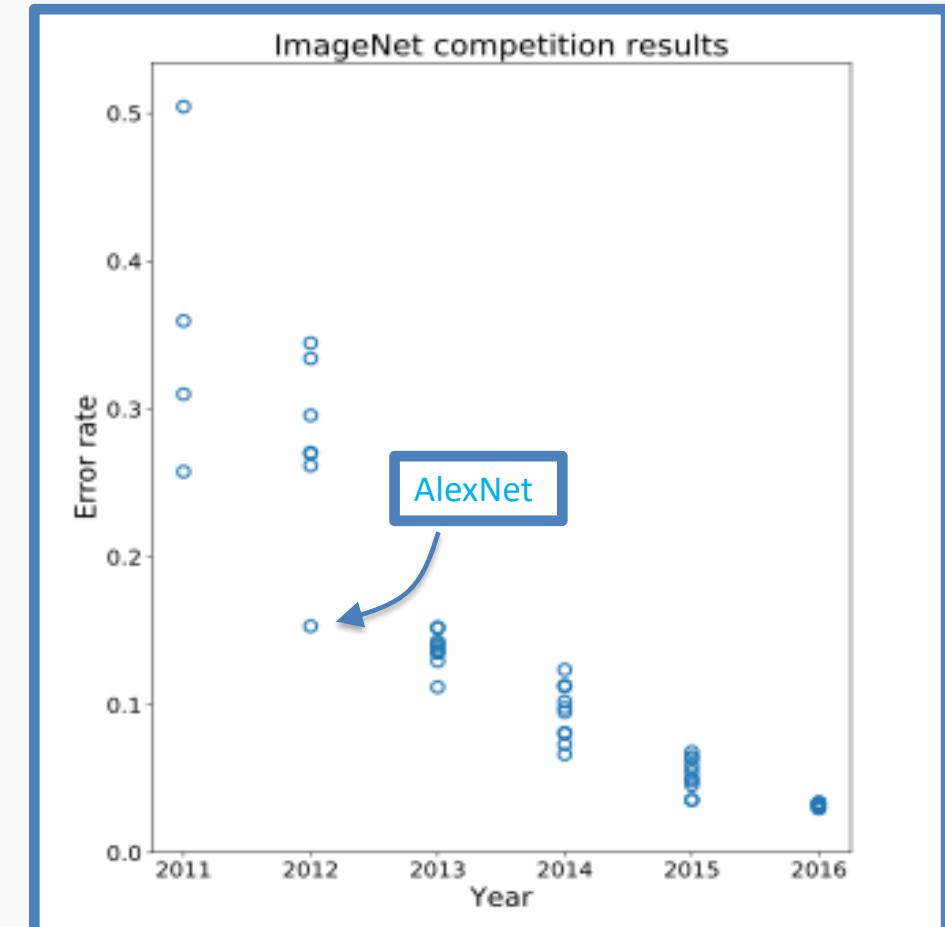
# AlexNet

- Developed by Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton at Utoronto in 2012. More than 25000 citations.
- Destroyed the competition in the 2012 [ImageNet Large Scale Visual Recognition Challenge](#). Showed benefits of CNNs and kickstarted AI revolution.
- top-5 error of 15.3%, more than 10.8 percentage points lower than runner-up.

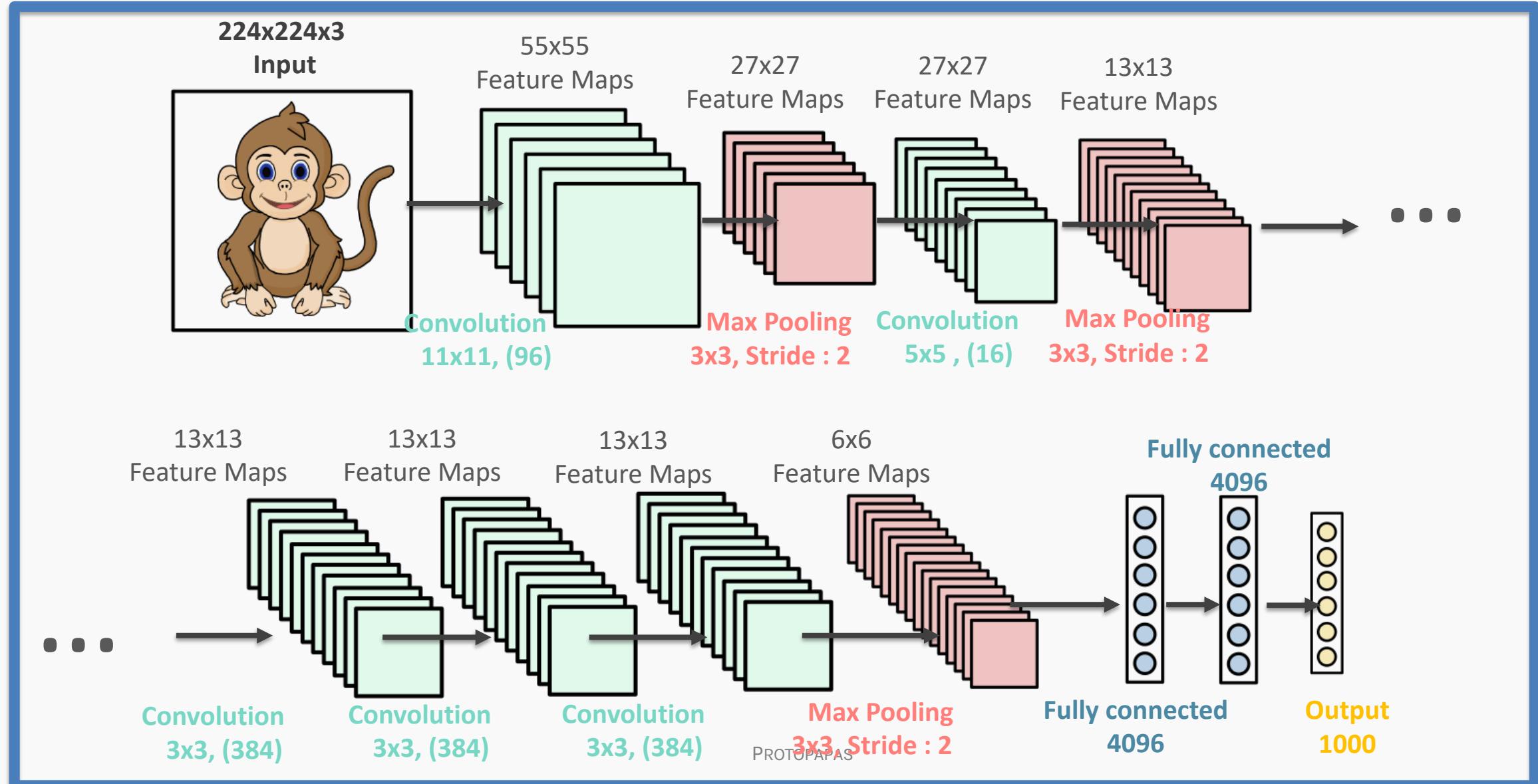


# AlexNet

- Developed by Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton at Utoronto in 2012. More than 25000 citations.
- Destroyed the competition in the 2012 [ImageNet Large Scale Visual Recognition Challenge](#). Showed benefits of CNNs and kickstarted AI revolution.
- top-5 error of 15.3%, more than 10.8 percentage points lower than runner-up.
- Main contributions:
  - Trained on ImageNet with data augmentation.
  - Increased depth of model, GPU training (six days).
  - Smart optimizer and Dropout layers.
  - ReLU activation!



- 1.2 million high-resolution ( $227 \times 227 \times 3$ ) images in the ImageNet 2010 contest
- 1000 different classes, NN with 60 million parameters to optimize (~ 255 MB)
- Uses ReLu activation functions; GPUs for training, 12 layers



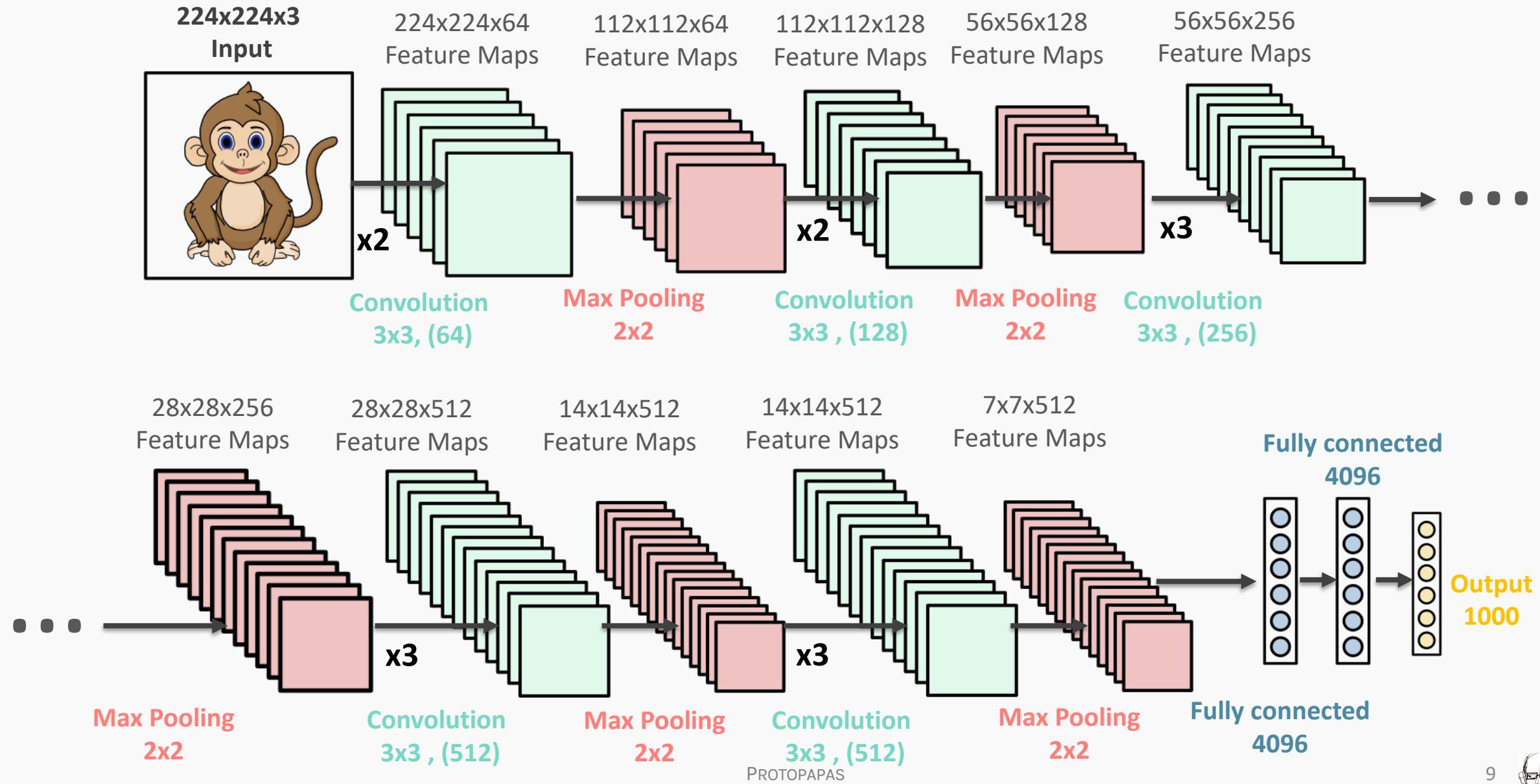
# VGG

- Introduced by Simonyan and Zisserman (Oxford) in 2014.
- Simplicity and depth as main points. Used **3x3 filters exclusively** and 2x2 MaxPool layers with stride 2.
- Showed that two 3x3 filters have an effective receptive field of 5x5.
- As spatial size decreases, depth increases.
- Convolutional layers use ‘same’ padding and stride s=1.
- Max-pooling layers use a window size 2 and stride s=2.

- ImageNet Challenge 2014; 16 or 19 layers; 138 million parameters.
- Trained for two to three weeks.
- Still used as of today.

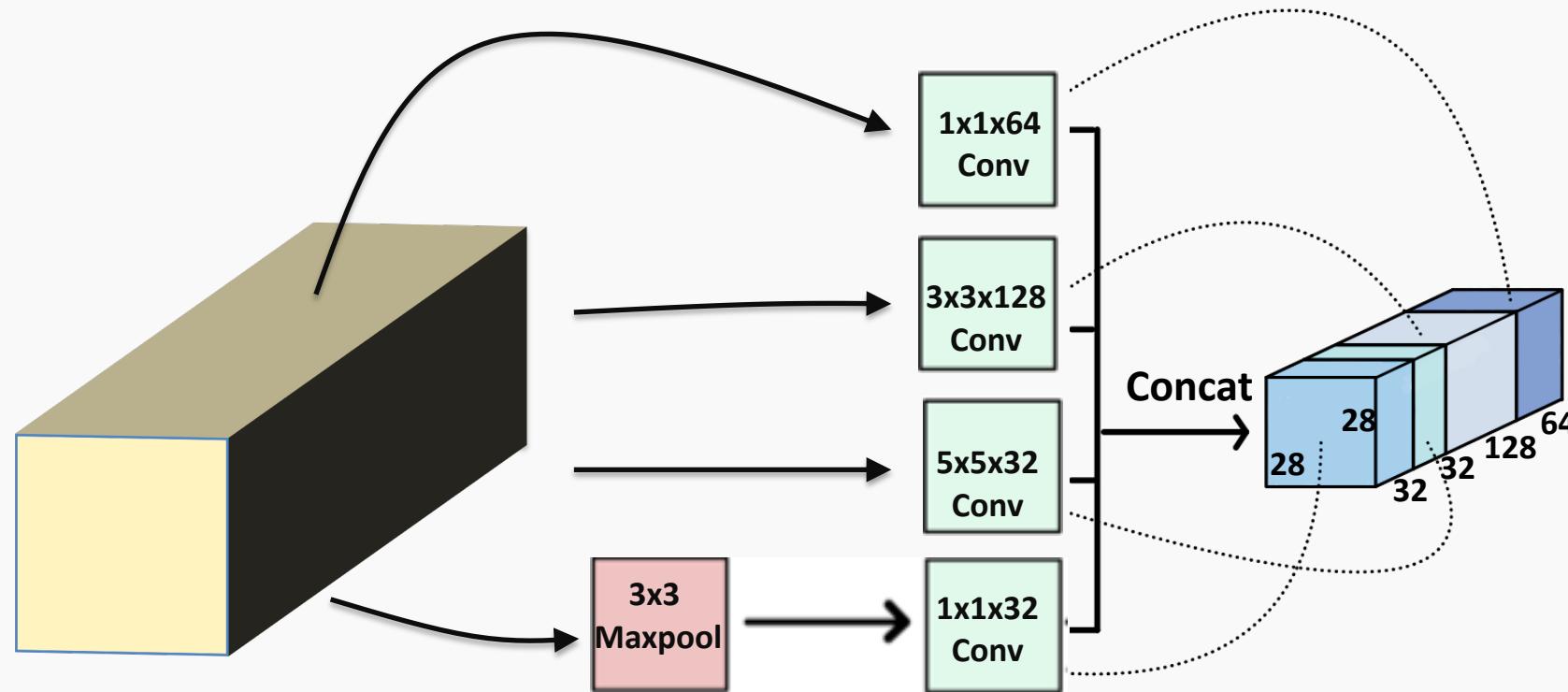


# VGG



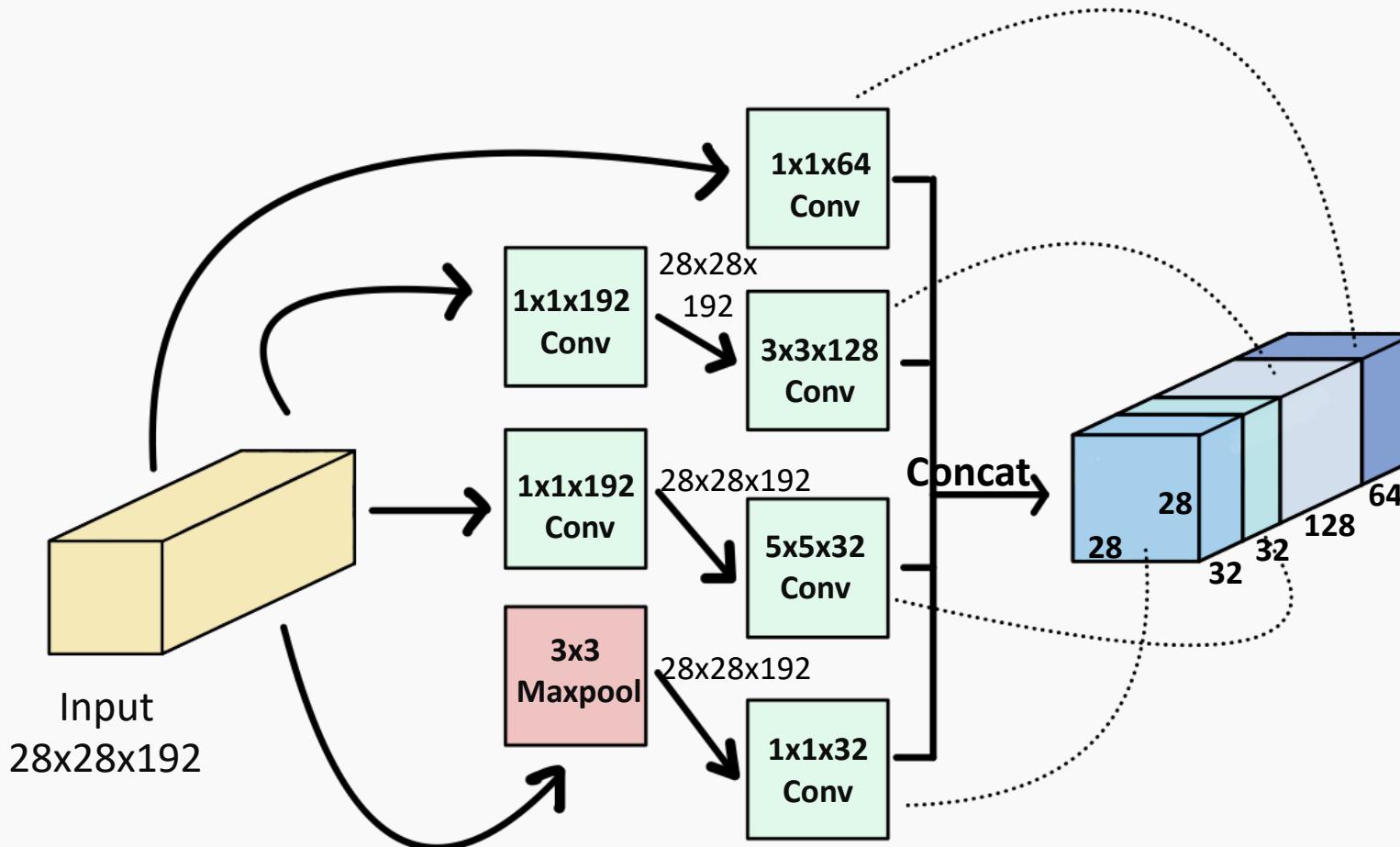
# SOTA Deep Models: Inception (GoogLeNet)

- The motivation behind inception networks is to use more than a single type of convolution layer at each layer.
- Use  $1 \times 1$ ,  $3 \times 3$ ,  $5 \times 5$  convolutional layers, and max-pooling layers in parallel.
- All modules use same convolution.



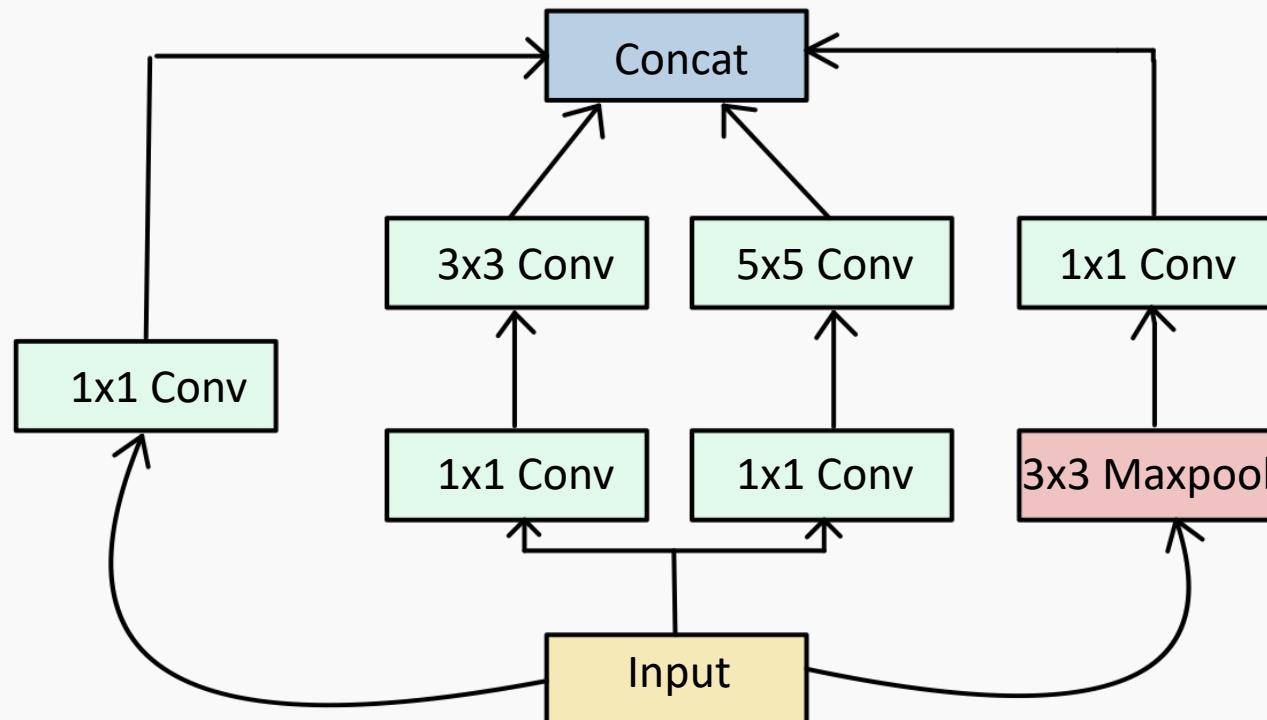
# SOTA Deep Models: Inception (GoogLeNet)

- Use  $1 \times 1$  convolutions that **reduce** the size of the channel dimension.
  - The number of channels can vary from the input to the output.

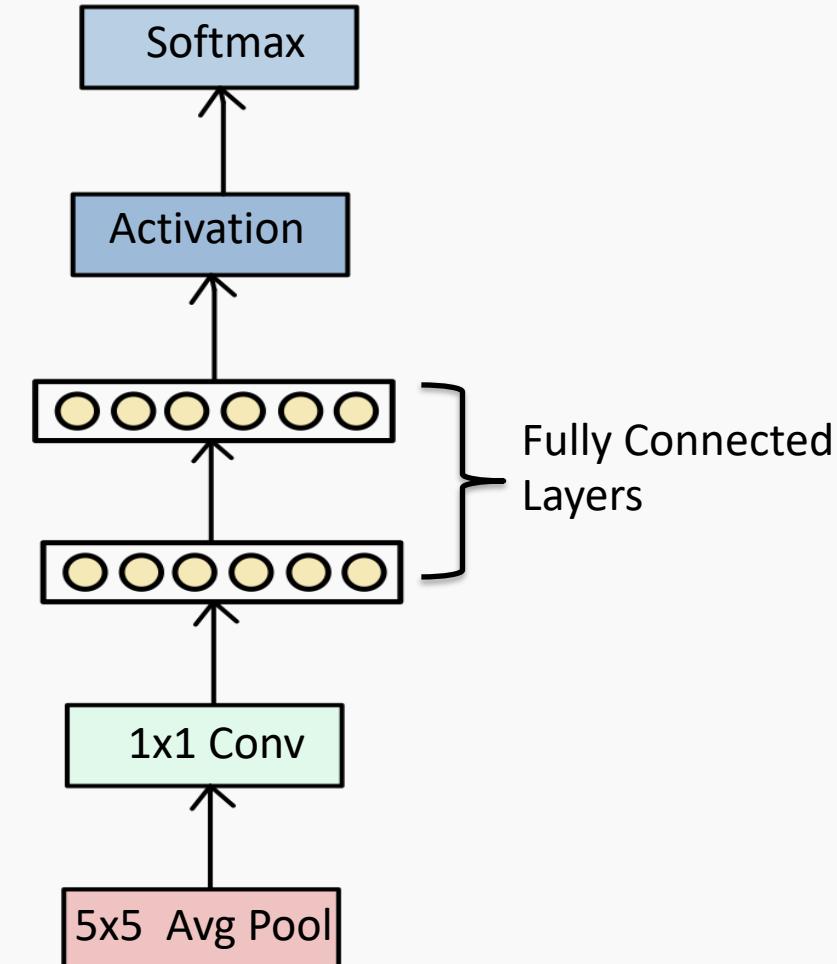


# SOTA Deep Models: Inception (GoogLeNet)

The Inception Block

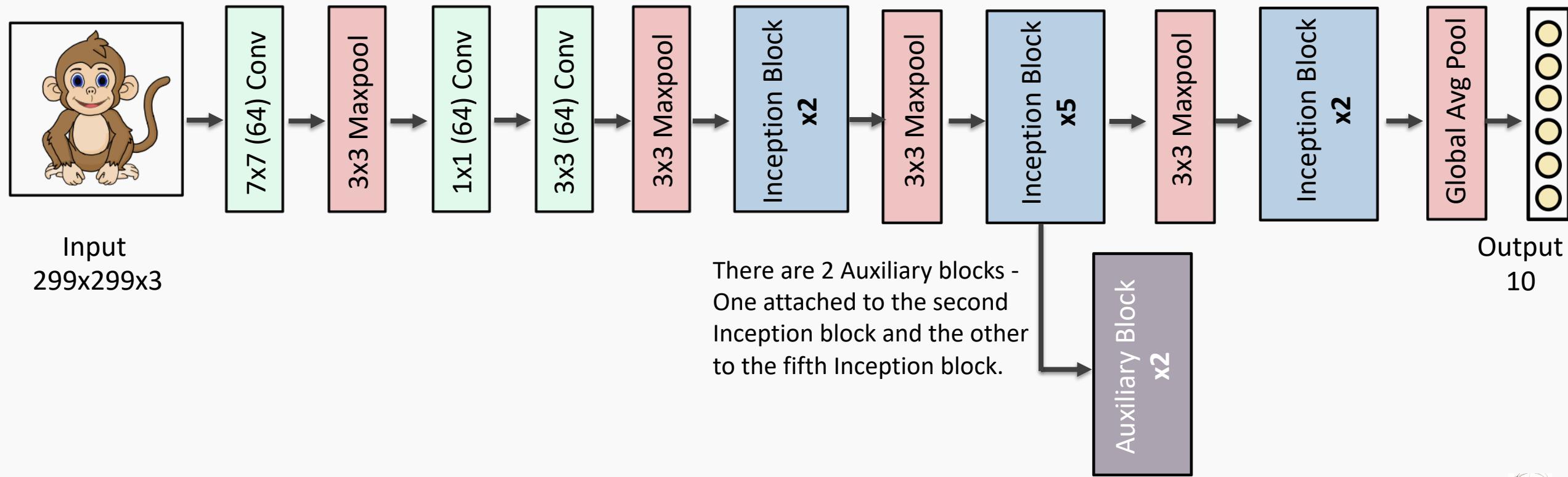


Auxiliary Block



# SOTA Deep Models: Inception (GoogLeNet)

- The inception network is formed by concatenating other inception modules.
- It includes several softmax output units to enforce regularization.

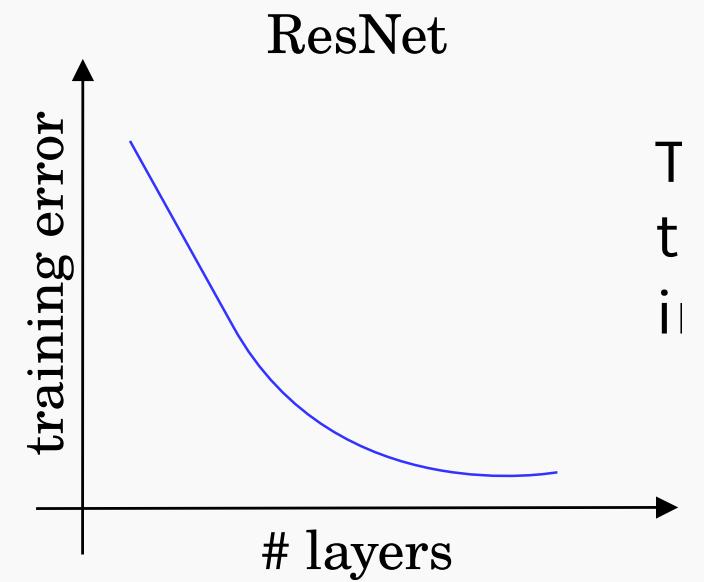
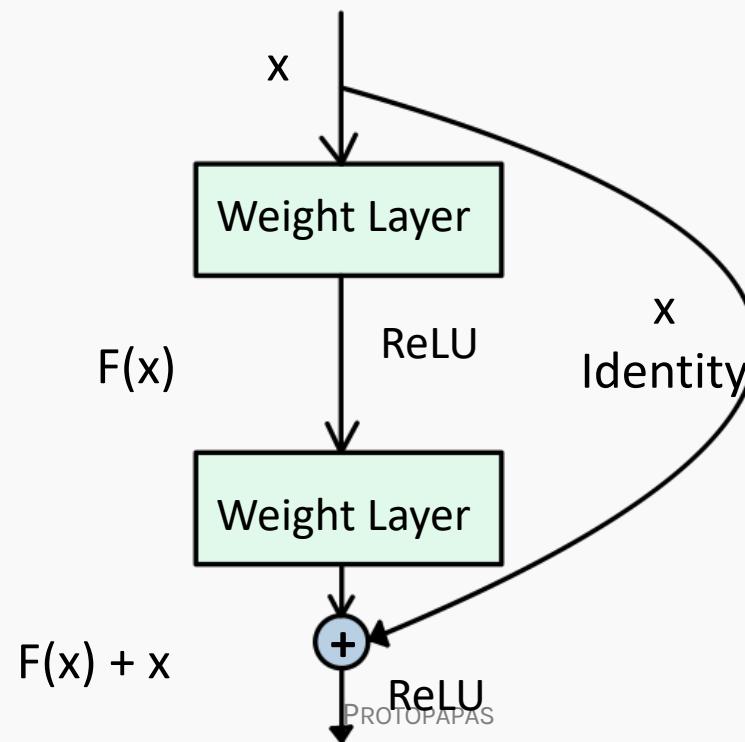
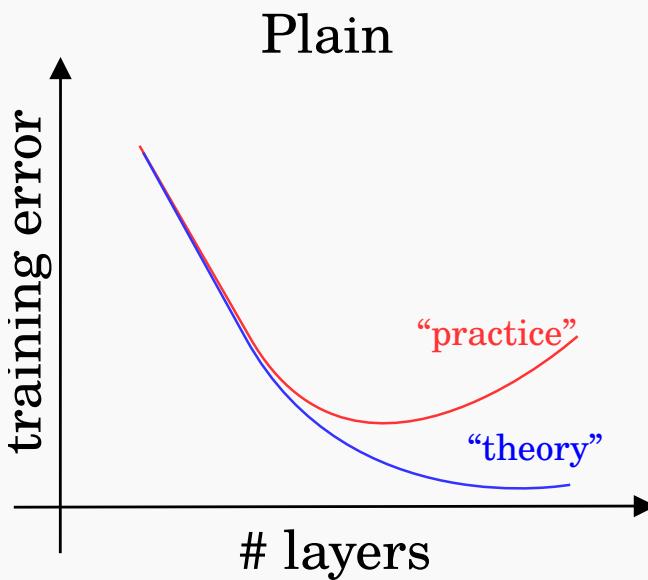


# Mandatory Meme



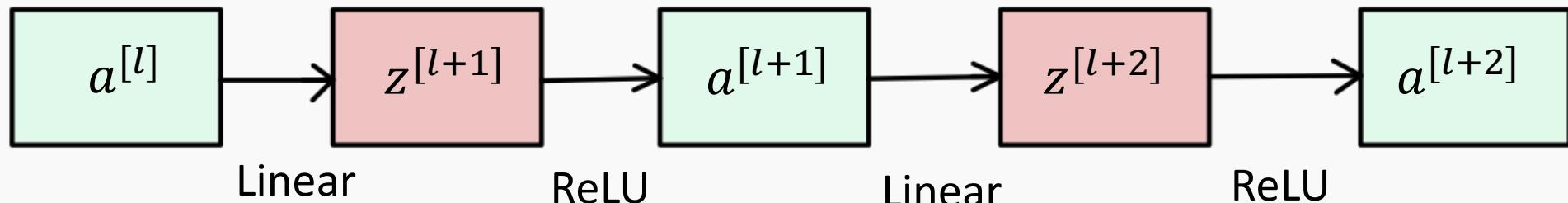
# ResNet

- Presented by [He et al.](#) (Microsoft), 2015. Won ILSVRC 2015 in multiple categories. Very similar to Highway Networks [Srivastava et al. 2015](#) introduced the same time.
- Main idea: **Residual block**. Allows for extremely deep networks.
- Authors believe that it is easier to optimize the residual mapping than the original one. Furthermore, residual block can decide to “shut itself down” if needed.

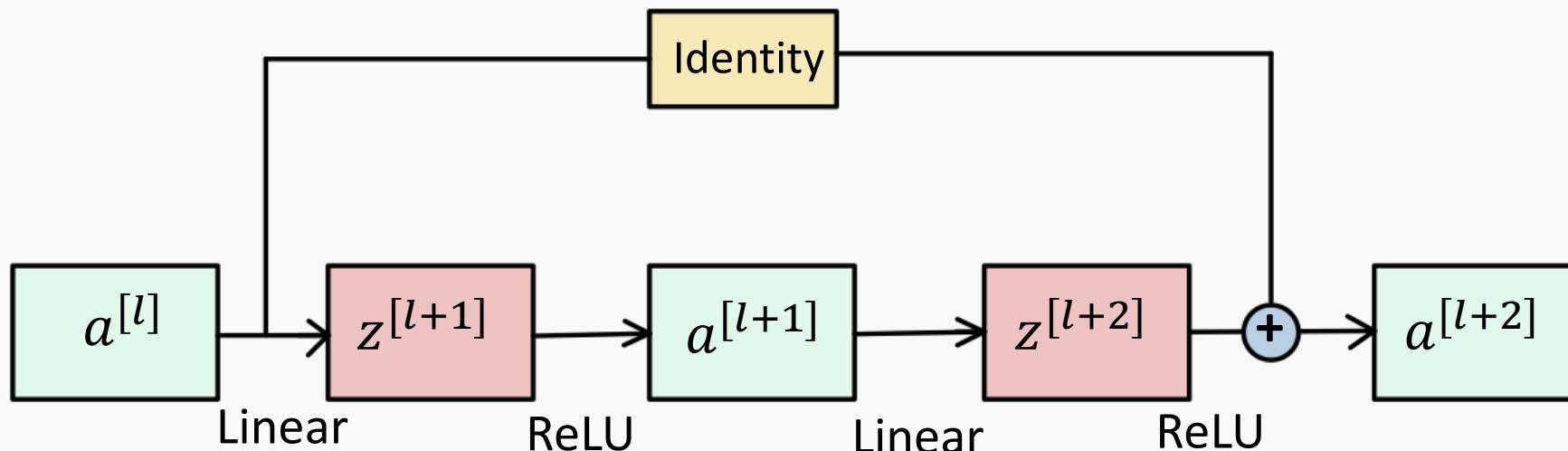


# ResNet

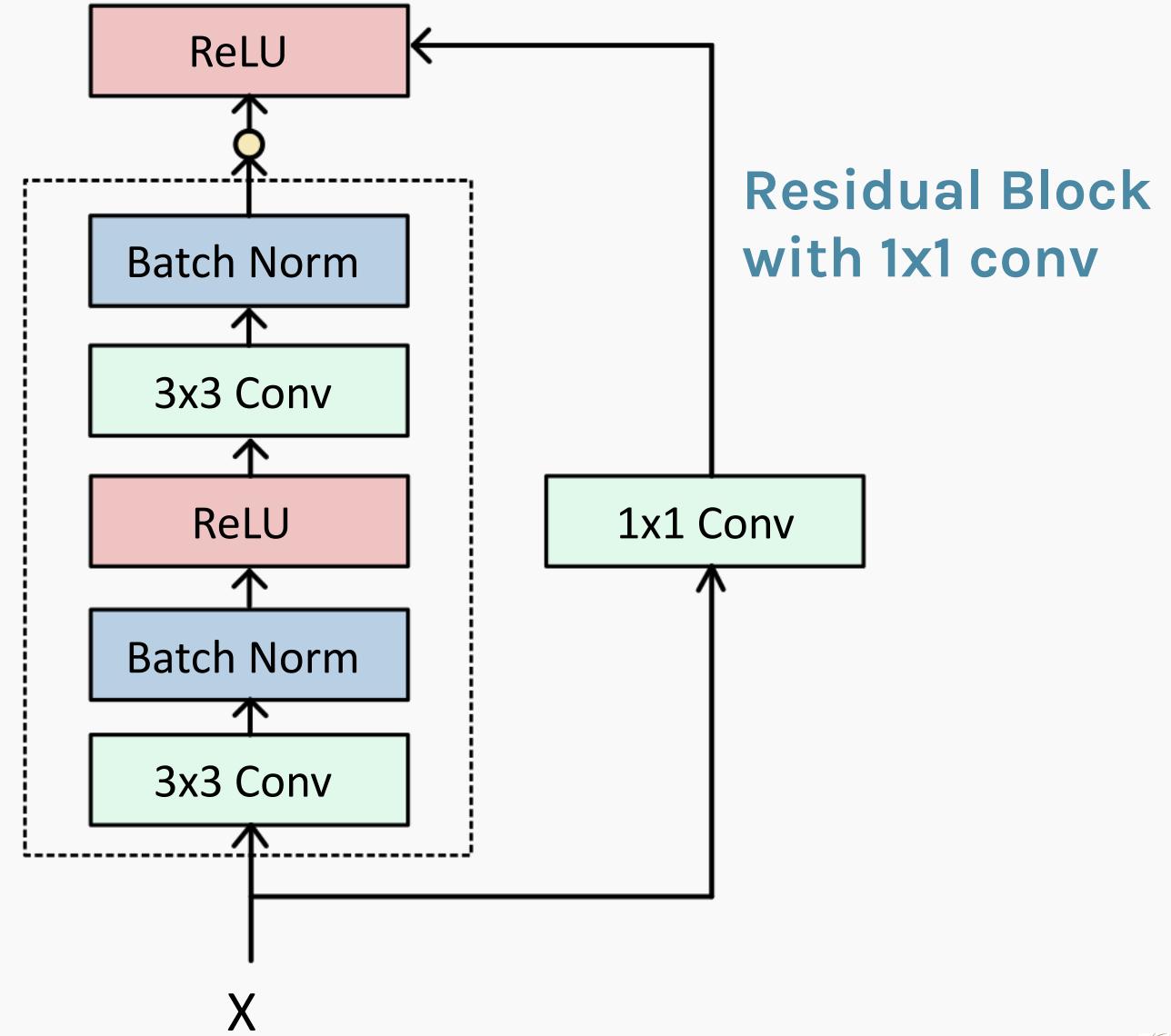
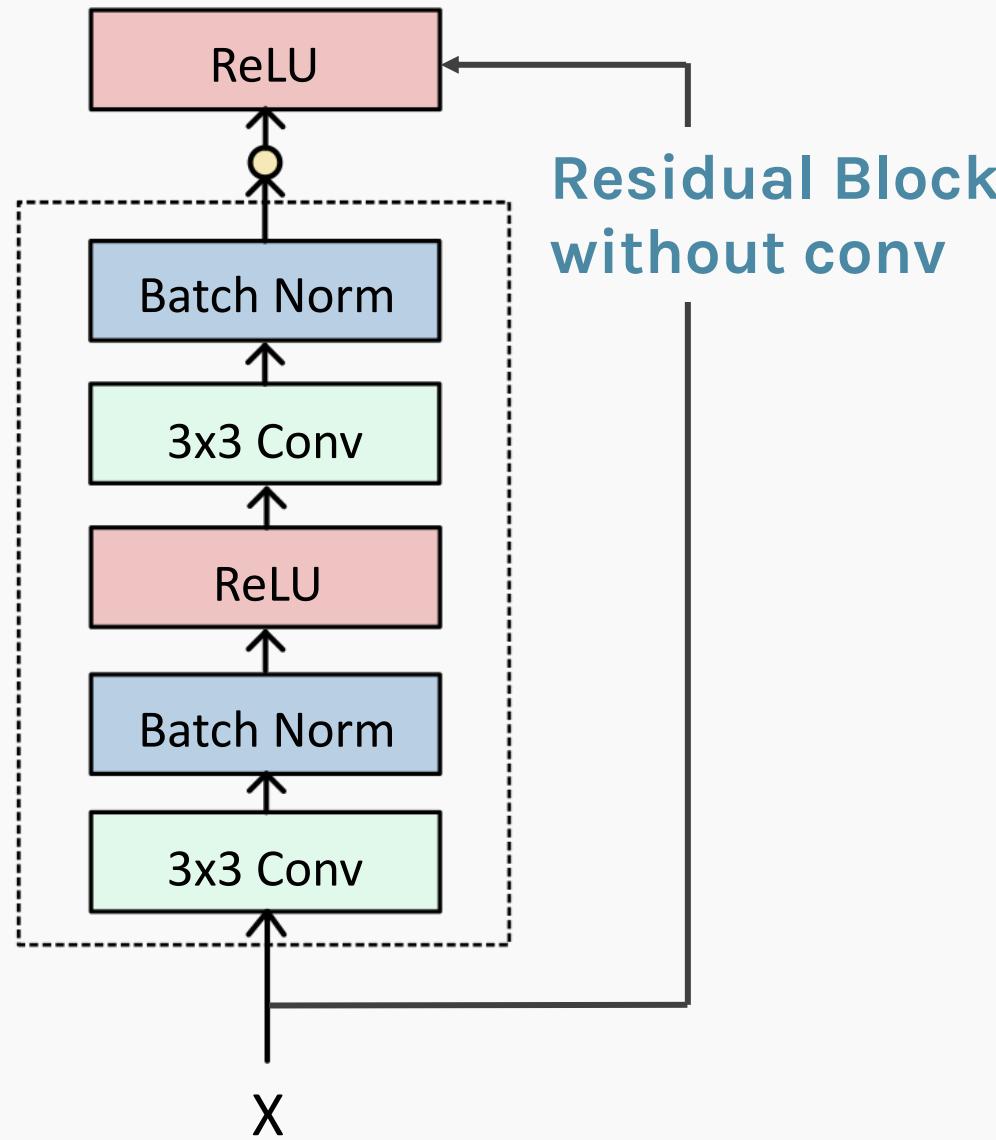
- Residual nets appeared in 2016 to train very deep NN (100 or more layers).
- Their architecture uses ‘residual blocks’.
- Plain network structure:



- **Residual network block**

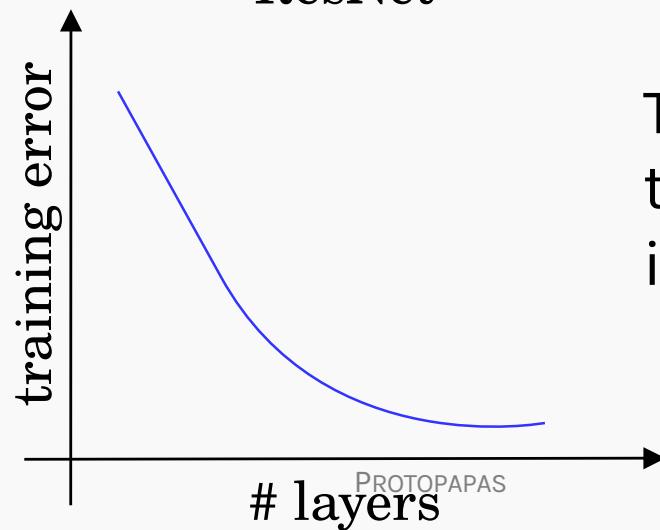
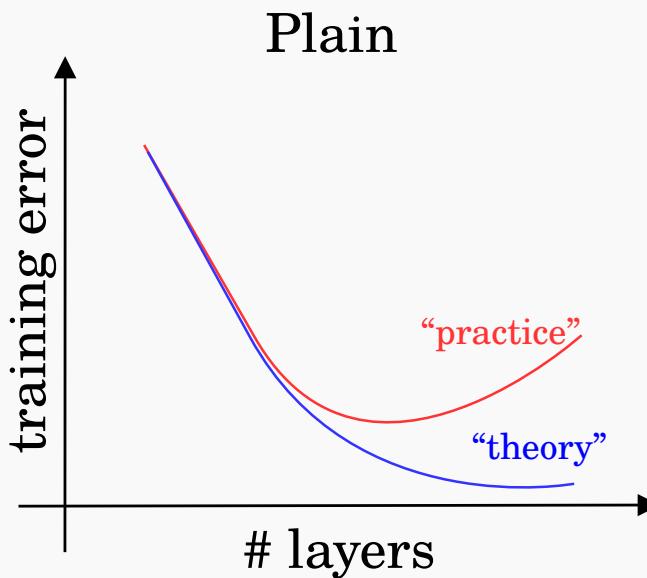
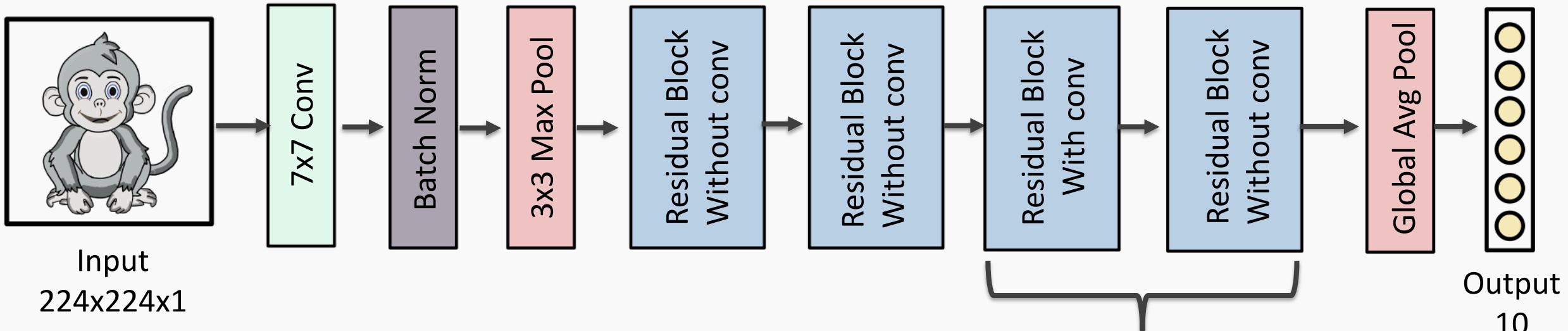


# ResNet



# ResNet

The residual network stacks blocks sequentially



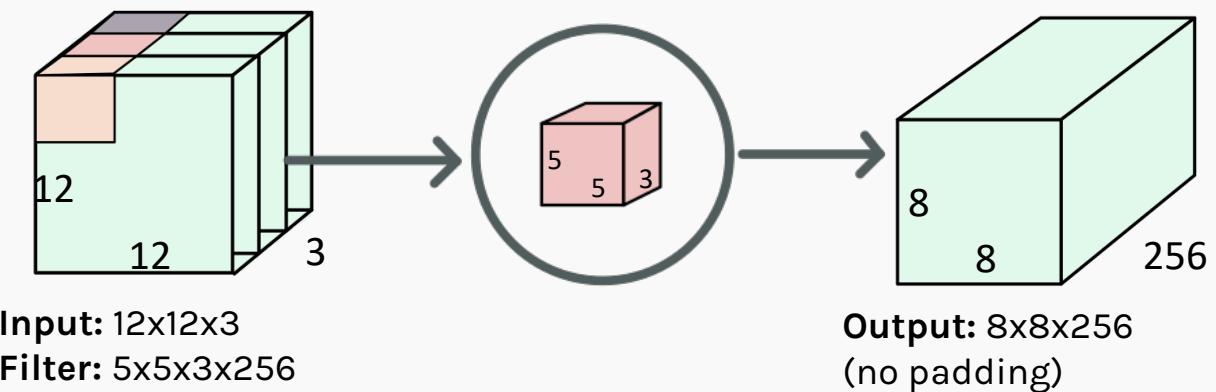
The idea is to allow the network to become deeper without increasing the training time



# SOTA Deep Models: MobileNet

## Standard Convolution

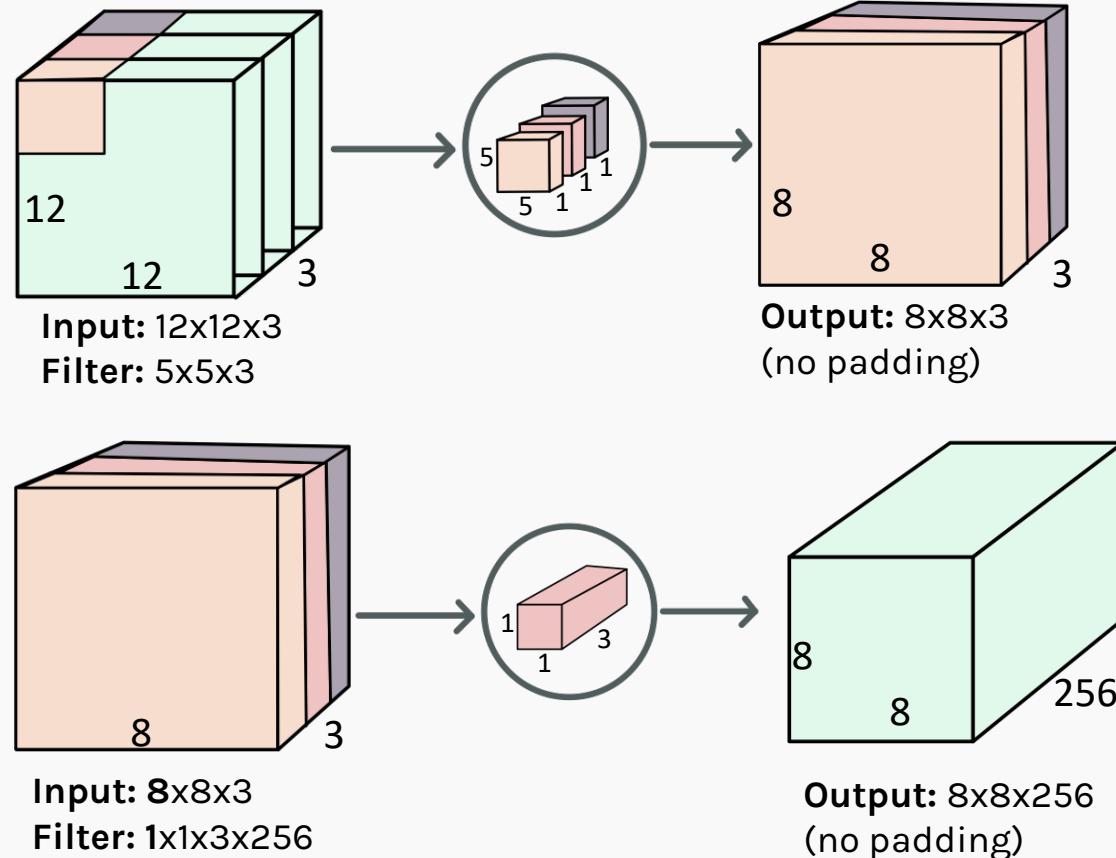
Filters and combines inputs into a new set of outputs in one step



**MACs:**  $(5 \times 5) \times 3 \times 256 \times (12 \times 12) \sim 2.8M$   
**Parameters:**  $(5 \times 5 \times 3) \times 256 + 256 \sim 20K$

## Depth-Wise Separable Convolution (DW)

It combines a depth wise convolution and a pointwise convolution

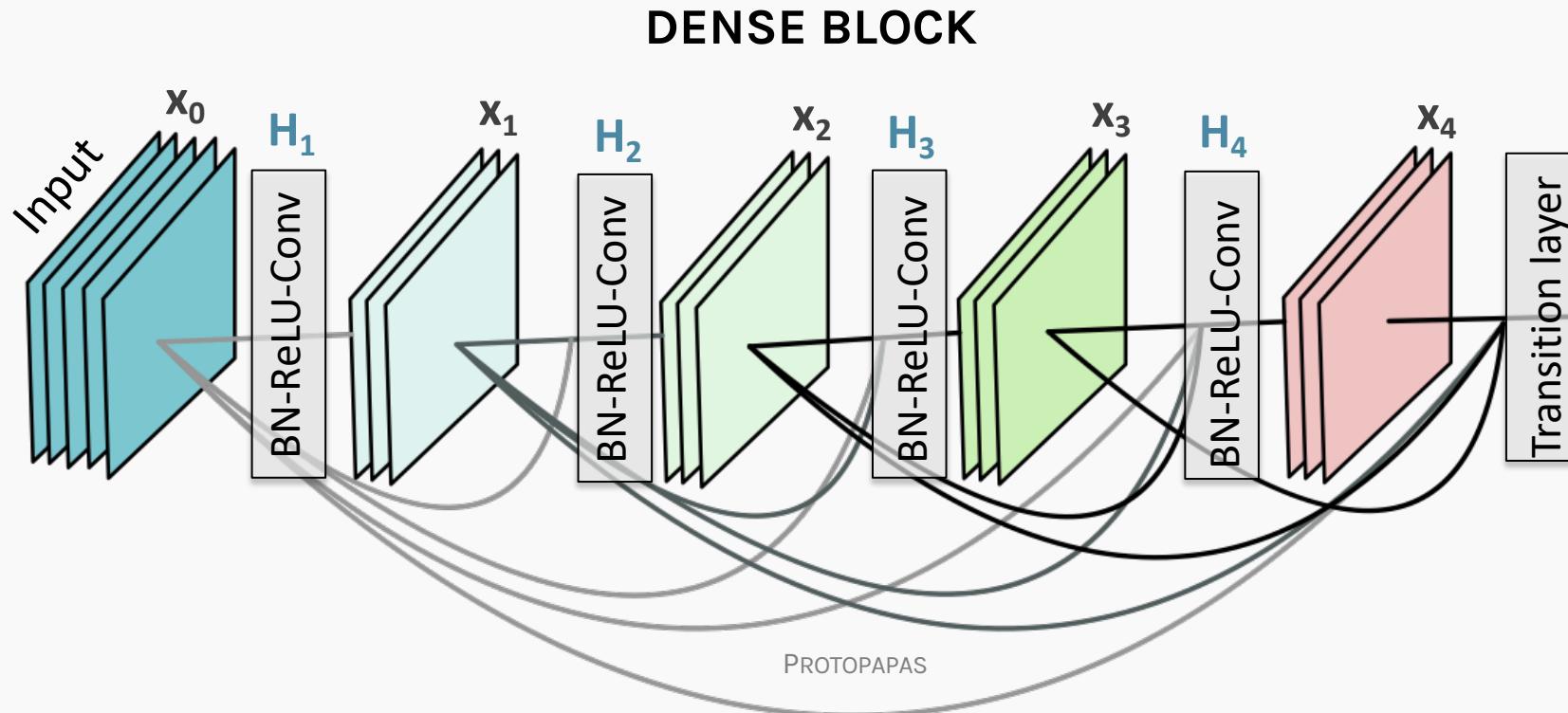


**MACs:**  $(5 \times 5) \times 3 \times (12 \times 12) + 3 \times 256 \times (8 \times 8) \sim 60K$   
**Parameters:**  $(5 \times 5 \times 3 + 3) + (1 \times 1 \times 3 \times 256 + 256) \sim 1K$



# SOTA Deep Models: DenseNets

- **Goal:** allow maximum information (and gradient) flow → connect every layer directly with each other.
- DenseNets exploit the potential of the network through feature reuse → no need to learn redundant feature maps.
- DenseNets layers are very narrow (e.g. 12 filters), and they just add a small set of new feature-maps.



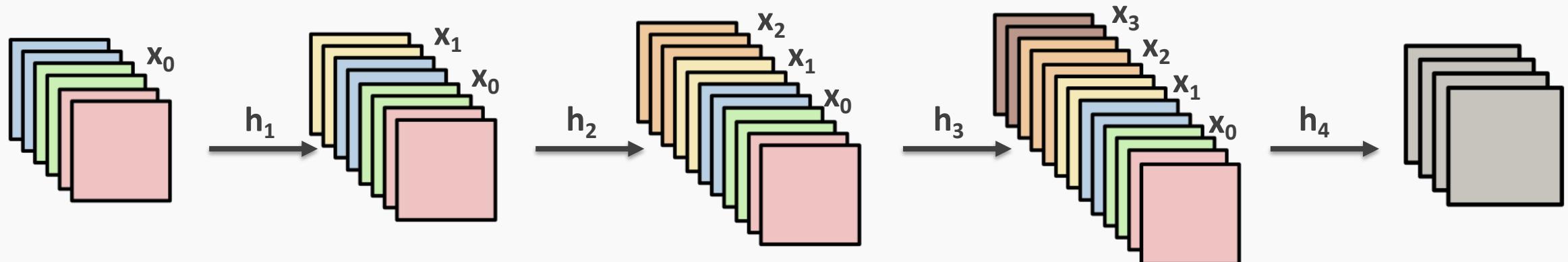
# SOTA Deep Models: DenseNets

- DenseNets do not sum the output feature maps of the layer with the incoming feature maps but concatenate them:

$$a^{[l]} = g([a^{[0]}, a^{[1]}, \dots, a^{[l-1]}])$$

- Dimensions of the feature maps remains constant within a block, but the number of filters changes between them → **growth rate**:

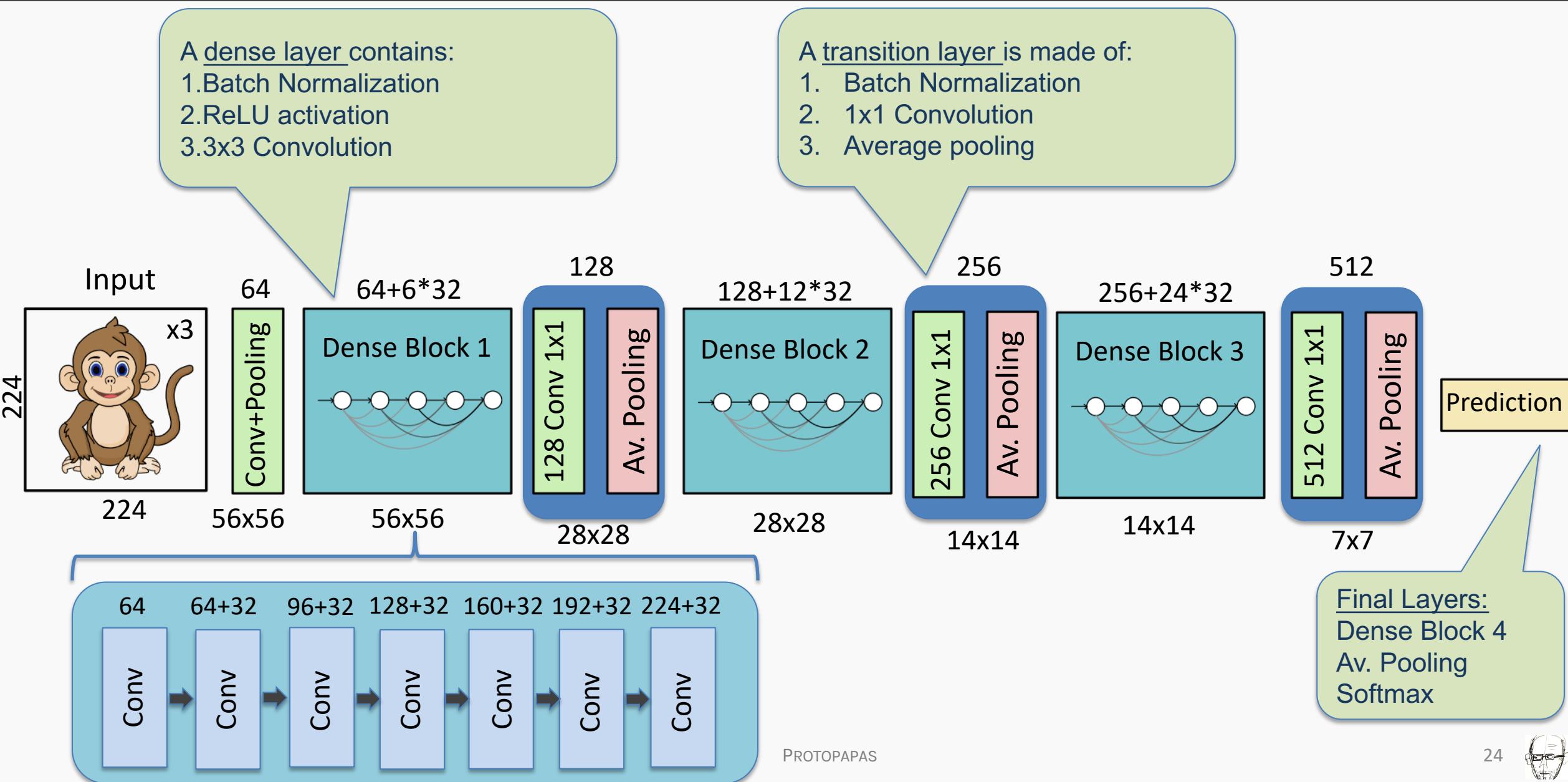
$$k^{[l]} = k^{[0]} + k(l - 1)$$



Concatenation during forward propagation



# SOTA Deep Models: DenseNets



# Beyond

---

- MobileNetV2 (<https://arxiv.org/abs/1801.04381>)
- Inception-Resnet, v1 and v2  
(<https://arxiv.org/abs/1602.07261>)
- Wide-Resnet (<https://arxiv.org/abs/1605.07146>)
- Xception (<https://arxiv.org/abs/1610.02357>)
- ResNeXt (<https://arxiv.org/pdf/1611.05431>)
- ShuffleNet, v1 and v2 (<https://arxiv.org/abs/1707.01083>)
- Squeeze and Excitation Nets  
(<https://arxiv.org/abs/1709.01507> )



# Exercise: Performance comparison of different SOTAs

The goal of this exercise is to compare different architectures on speed, size and performance.

There is very little coding to do in this exercise but familiarize yourself with colab and become comfortable using any of the SOTAs we have presented in this lecture.

**WARNING: DO NOT USE GPU for this exercises!**

Your final plot may resemble this one:

