ResNet

Computer Vision

Carnegie Mellon University (Kris Kitani)

Deep Residual Learning for Image Recognition

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

Abstract

Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers—8× deeper than VGG nets [41] but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test set. This result won the 1st place on the ILSVRC 2015 classification task. We also present analysis on CIFAR-10 with 100 and 1000 layers.

The depth of representations is of central importance for many visual recognition tasks. Solely due to our extremely deep representations, we obtain a 28% relative improvement on the COCO object detection dataset. Deep residual nets are foundations of our submissions to ILSVRC & COCO 2015 competitions[1], where we also won the 1st places on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation.

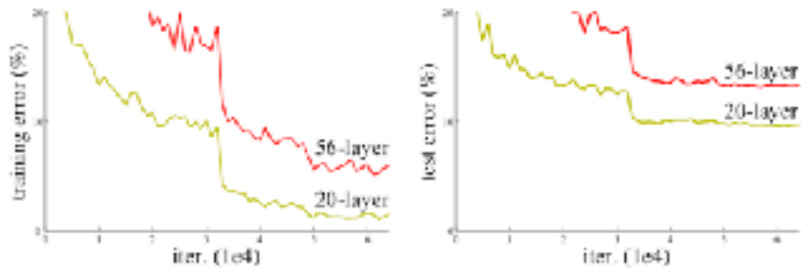


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

greatly benefited from very deep models.

Driven by the significance of depth, a question arises: *Is learning better networks as easy as stacking more layers?* An obstacle to answering this question was the notorious problem of vanishing/exploding gradients [1, 9], which hamper convergence from the beginning. This problem, however, has been largely addressed by normalized initialization [23, 9, 37, 13] and intermediate normalization layers [16], which enable networks with tens of layers to start converging for stochastic gradient descent (SGD) with back-propagation [22].

When deeper networks are able to start converging, a *degradation* problem has been exposed: with the network

CVPR 2016

ResNet

ILSVRC & COCO 2015

ImageNet Classification
ResNet 152

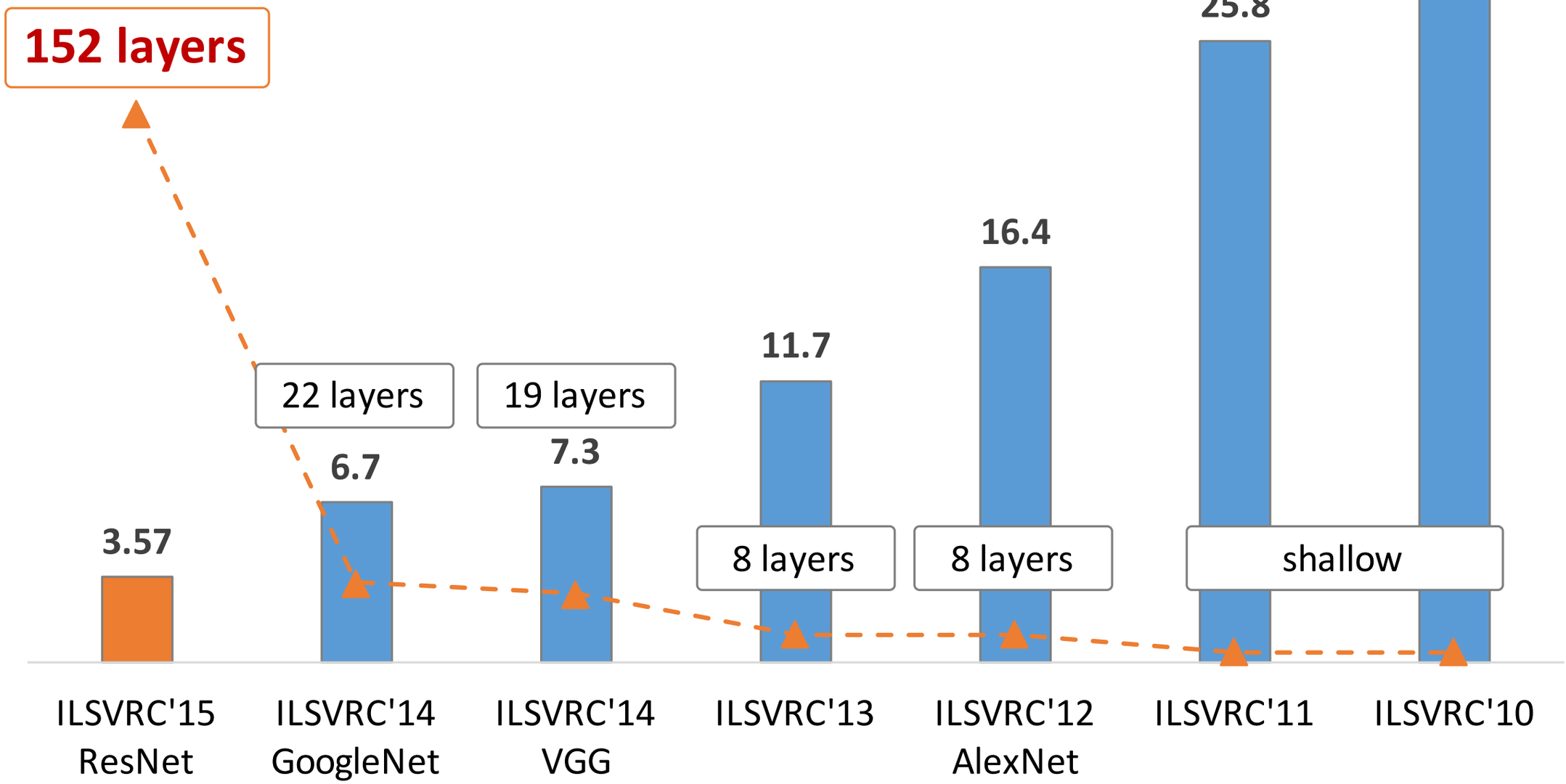
ImageNet Detection
16%
Better than 2nd

ImageNet Localization
27%
Better than 2nd

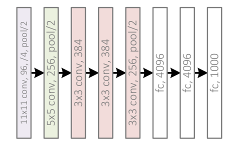
COCO Detection
11%
Better than 2nd

COCO Segmentation
12%
Better than 2nd

Revolution of Depth

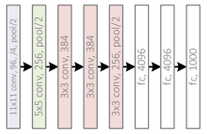


AlexNet

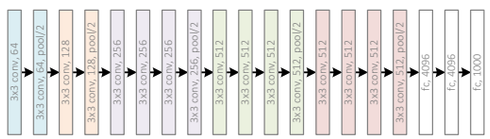


11x11 conv, 96, /4, pool/2
5x5 conv, 256, pool/2
3x3 conv, 384
3x3 conv, 384
3x3 conv, 256, pool/2
fc, 4096
fc, 4096
fc, 1000

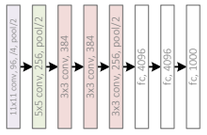
AlexNet



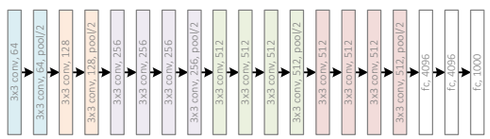
VGG-19



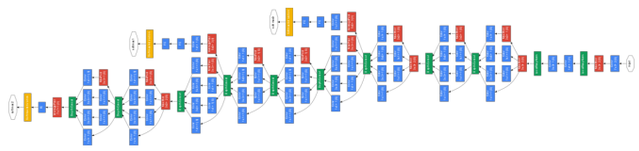
AlexNet



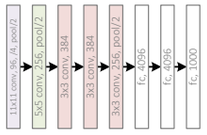
VGG-19



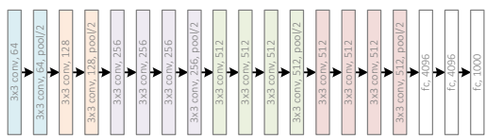
GoogLeNet



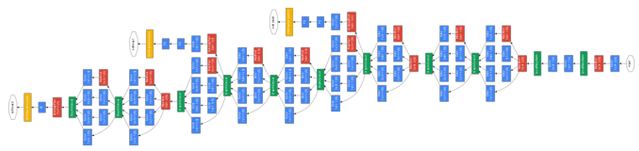
AlexNet



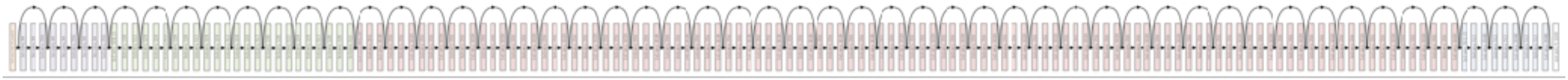
VGG-19



GoogLeNet



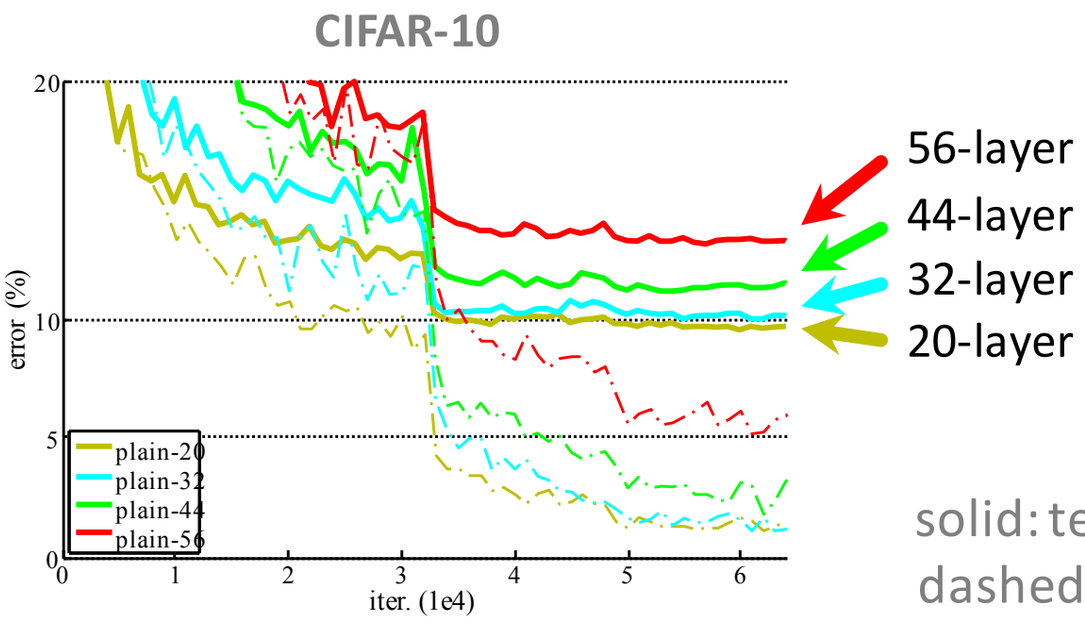
ResNet-152

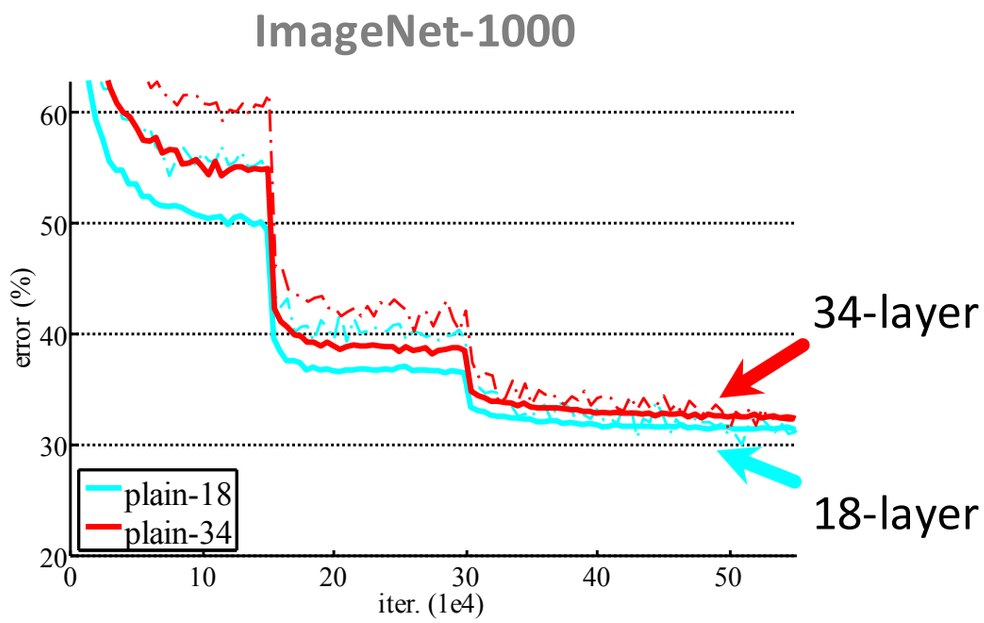


Deeper is better so just stack more convolutions.

Right?

NO!





What?! Deeper networks have higher training error!

What is causing this?

Expectation:

Add extra layers set to identity on top of shallow network.

Training error should be the same not worse.

Deeper models should not have higher training error.

Expectation:

Add extra layers set to identity on top of shallow network.

Training error should be the same not worse.

Deeper models should not have higher training error.

Reality:

Solvers (gradient descent) cannot find the solution…

Expectation:

Add extra layers set to identity on top of shallow network.

Training error should be the same not worse.

Deeper models should not have higher training error.

Reality:

Solvers (gradient descent) cannot find the solution…

Idea:

Let’s help deep networks start with the identity solution and then tune parameters.

Expectation:

Add extra layers set to identity on top of shallow network.

Training error should be the same not worse.

Deeper models should not have higher training error.

Reality:

Solvers (gradient descent) cannot find the solution…

Idea:

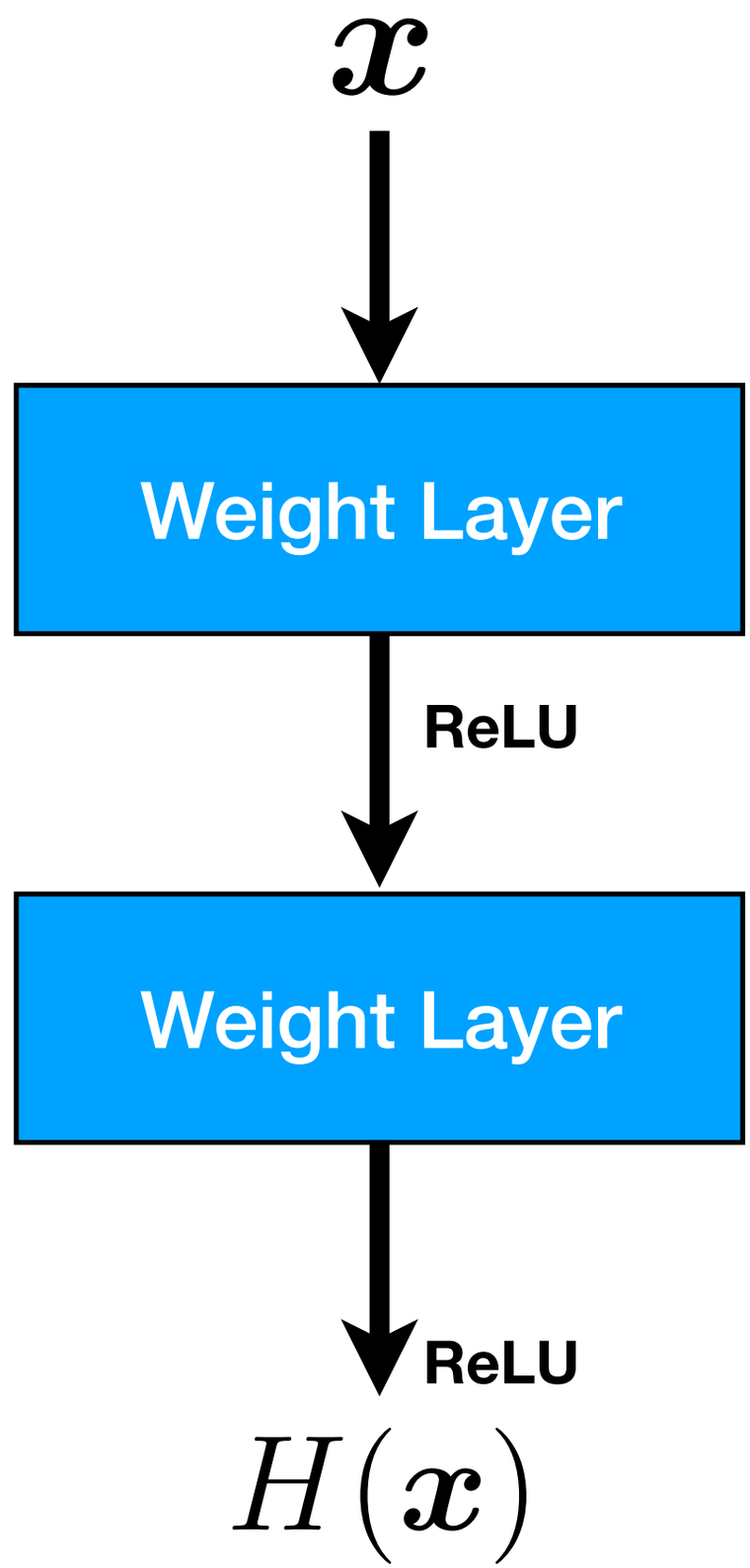
Let’s help deep networks start with the identity solution and then tune parameters.

‘Residual Learning’

Common Architectural Unit:

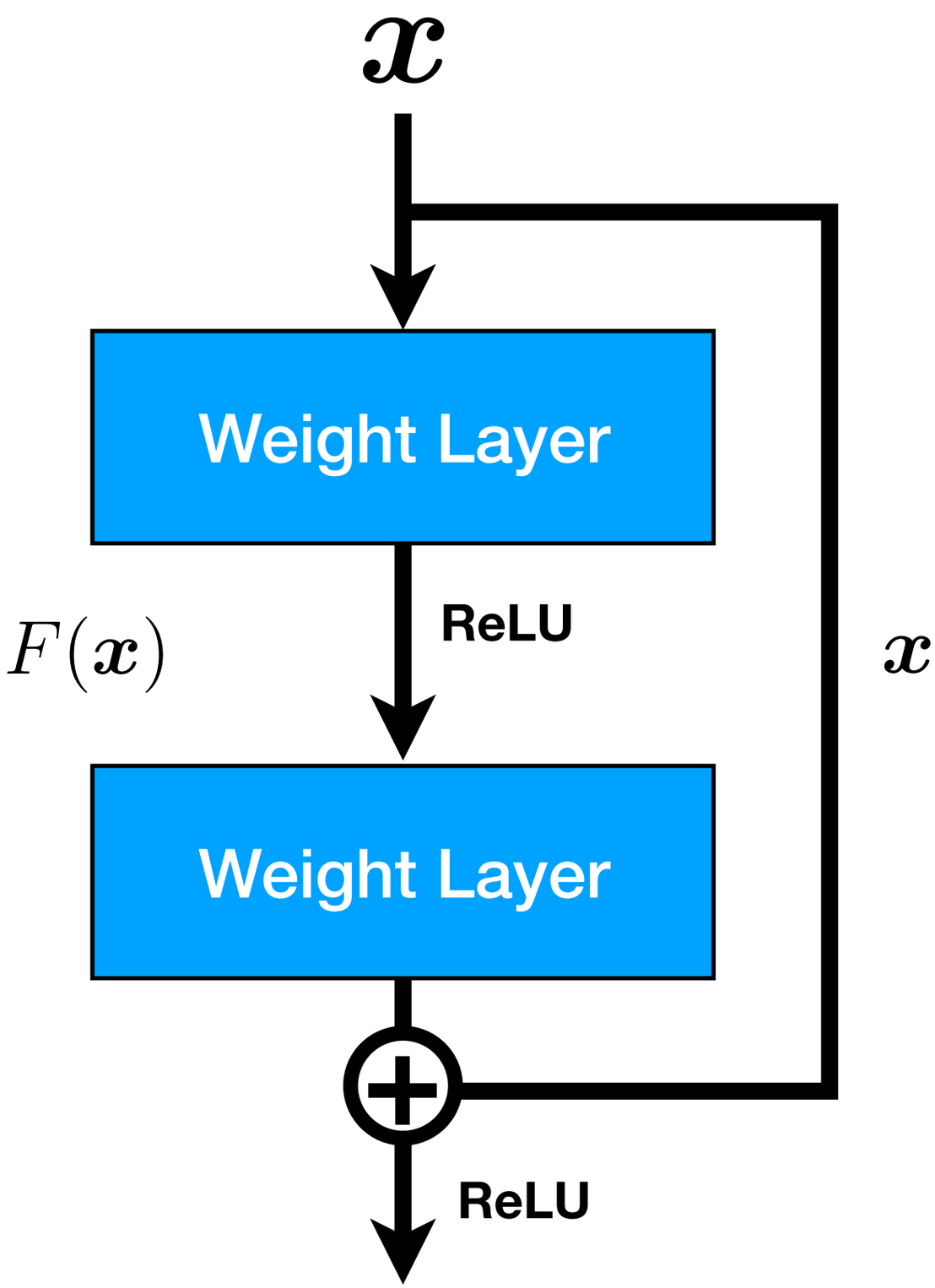
Does this neural network have the ability to learn an identity function?

Can you think of an easy way to allow the model to learn the identify function?



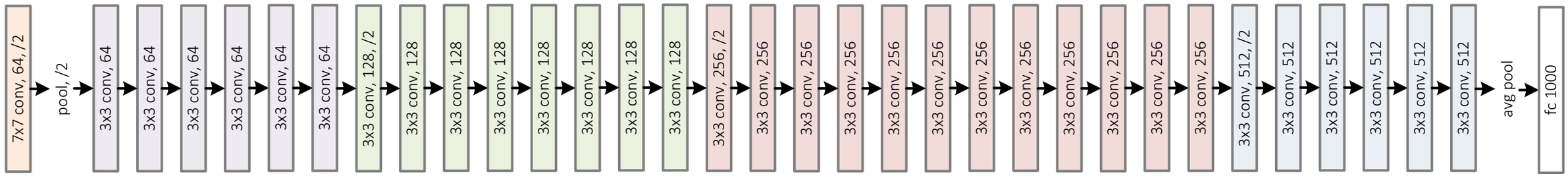
Residual Module

What happens when you set all weights to 0?

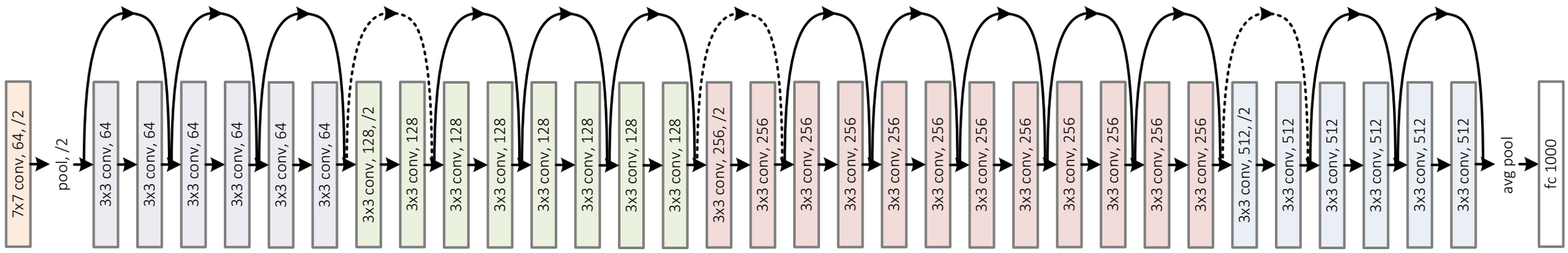


$$H(\boldsymbol{x}) = F(\boldsymbol{x}) + \boldsymbol{x}$$

VGG

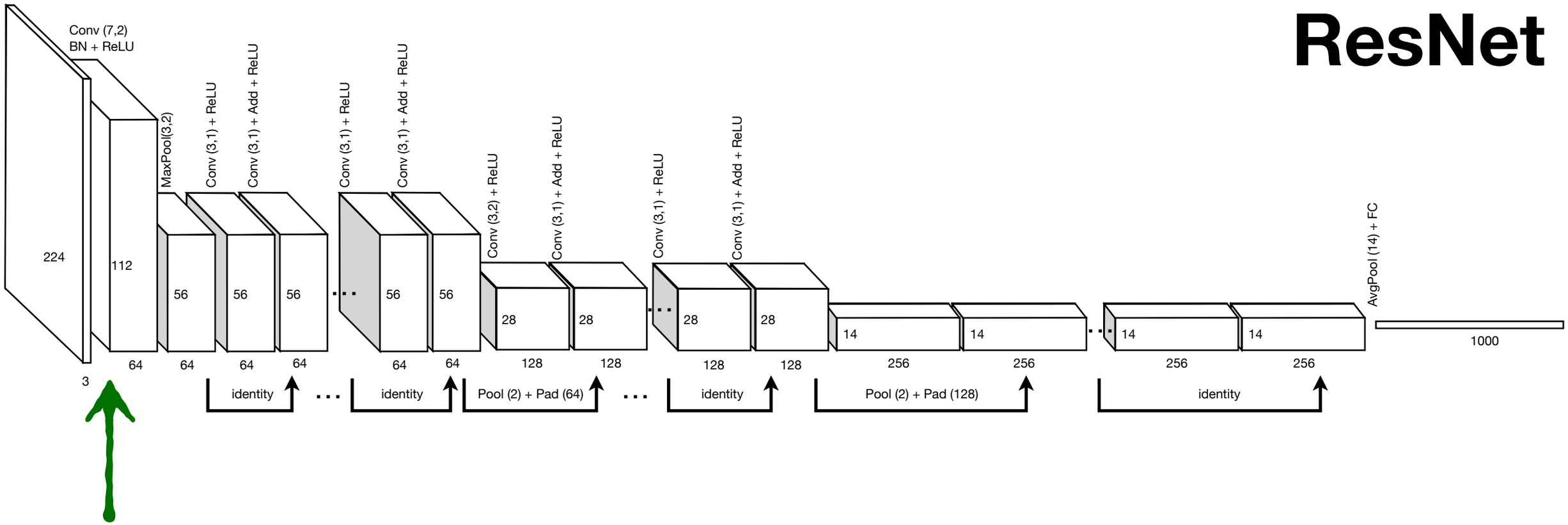


ResNet



- All 3x3 convolutions
- Reduce spatial resolution by 1/2, Increase channels by 2
- No FCs
- No Dropout

ResNet



Convolution + Batch Normalization + ReLU

7 x 7 convolution just for the first layer

Stride of 2 reduces the resolution in half

Batch Normalization to scale the responses

ReLU non-linearity

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

Sergey Ioffe SIOFFE@GOOGLE.COM
Christian Szegedy SZEGEDY@GOOGLE.COM
Google, 1600 Amphitheatre Pkwy, Mountain View, CA 94043

Abstract

Training Deep Neural Networks is complicated by the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. This slows down the training by requiring lower learning rates and careful parameter initialization, and makes it notoriously hard to train models with saturating nonlinearities. We refer to this phenomenon as *internal covariate shift*, and address the problem by normalizing layer inputs. Our method draws its strength from making normalization a part of the model architecture and performing the normalization *for each training mini-batch*. Batch Normalization allows us to use much higher learning rates and be less careful about initialization, and in some cases eliminates the need for Dropout. Applied to a state-

minimize the loss

$$\Theta = \arg\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} \ell(\mathrm{x}_i, \Theta)$$

where $\mathrm{x}_{1\ldots N}$ is the training data set. With SGD, the training proceeds in steps, at each step considering a *mini-batch* $\mathrm{x}_{1\ldots m}$ of size m. Using mini-batches of examples, as opposed to one example at a time, is helpful in several ways. First, the gradient of the loss over a mini-batch $\frac{1}{m}\sum_{i=1}^{m} \frac{\partial \ell(\mathrm{x}_i,\Theta)}{\partial \Theta}$ is an estimate of the gradient over the training set, whose quality improves as the batch size increases. Second, computation over a mini-batch can be more efficient than m computations for individual examples on modern computing platforms.

While stochastic gradient is simple and effective, it requires careful tuning of the model hyper-parameters, specifically the learning rate and the initial parameter values. The train-

ICML 2015

Batch Normalization (BN)

- Normalizing input (LeCun et al 1998 “Efficient Backprop”)
- BN: normalizing each layer, for each mini-batch
  - Greatly accelerate training
  - Less sensitive to initialization
  - Improve regularization

S. Ioffe & C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. ICML 2015

Recall:

Stochastic Gradient Descent

For each example sample $\{x_i, y_i\}$
(or mini-batch)

1. Predict

   a. Forward pass $\hat{y} = f_{\text{MLP}}(x_i; \theta)$

   b. Compute Loss $\mathcal{L}_i$

2. Update

   a. Back Propagation $\frac{\partial \mathcal{L}}{\partial \theta}$

   b. Gradient update $\theta \leftarrow \theta + \eta \frac{\partial \mathcal{L}}{\partial \theta}$

Problem:
Distribution can change dramatically!
(gradients are unstable)

Solution #1: Be conservative.
Make step size really small

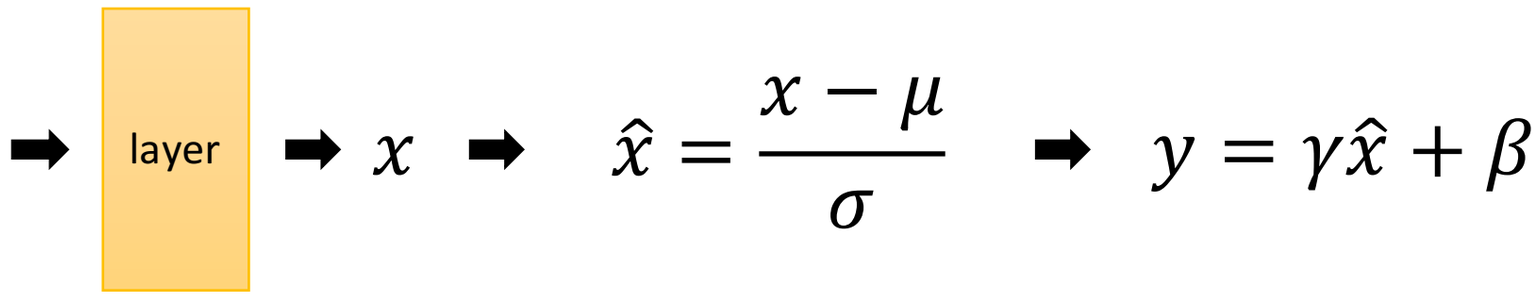
Solution #2: Scale the gradient with BN

What happens when you make the step size really small?

A really small step size makes the training time really long
(it takes forever for the parameters to change)

Instead, for each training mini-batch…

Batch Normalization (BN)

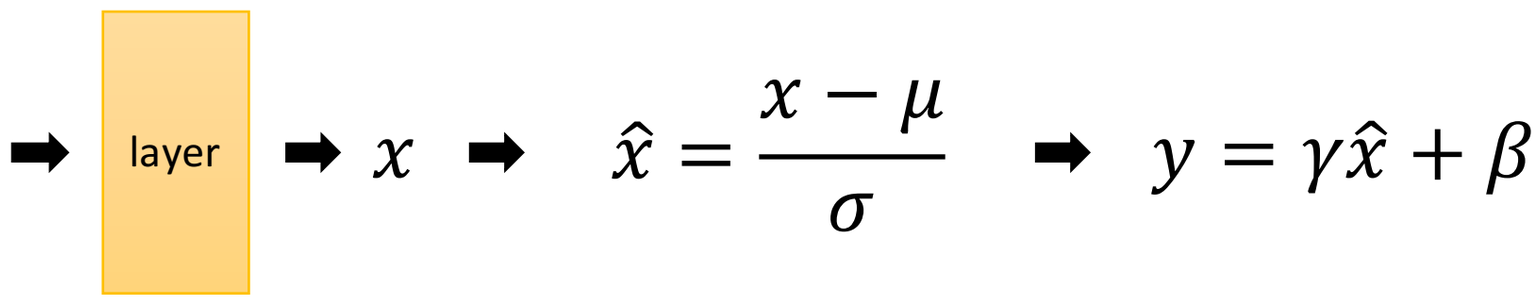


- μ: mean of x in mini-batch
- σ: std of x in mini-batch
- γ: scale
- β: shift

- μ, σ: functions of x, analogous to responses
- γ, β: parameters to be learned, analogous to weights

S. Ioffe & C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. ICML 2015

Batch Normalization (BN)



2 modes of BN:

- Train mode:
  - μ, σ are functions of x; backprop gradients
- Test mode:
  - μ, σ are pre-computed* on training set

Caution: make sure your BN is in a correct mode

*: by running average, or post-processing after training

S. Ioffe & C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. ICML 2015

Batch Normalization (BN)

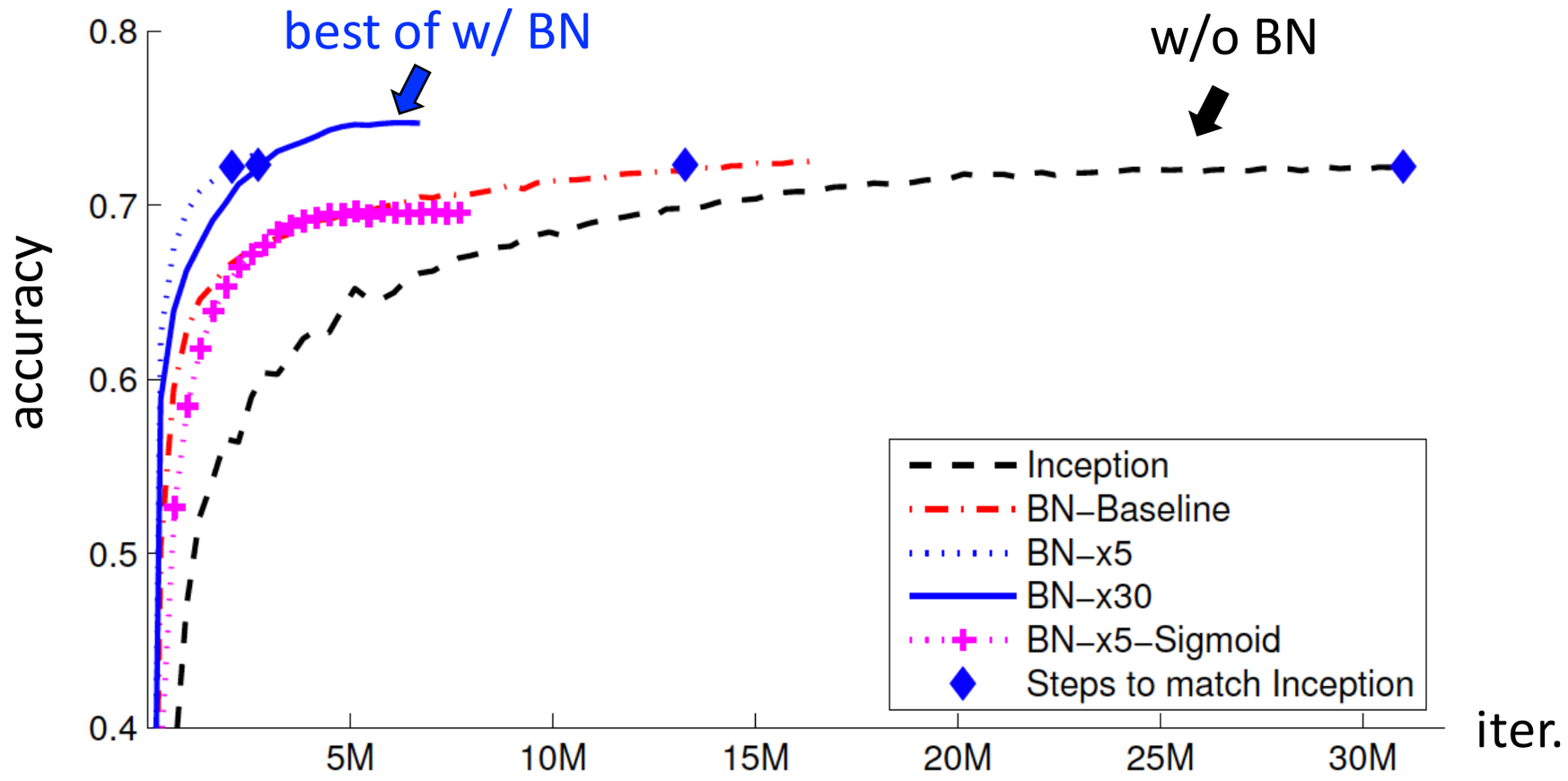
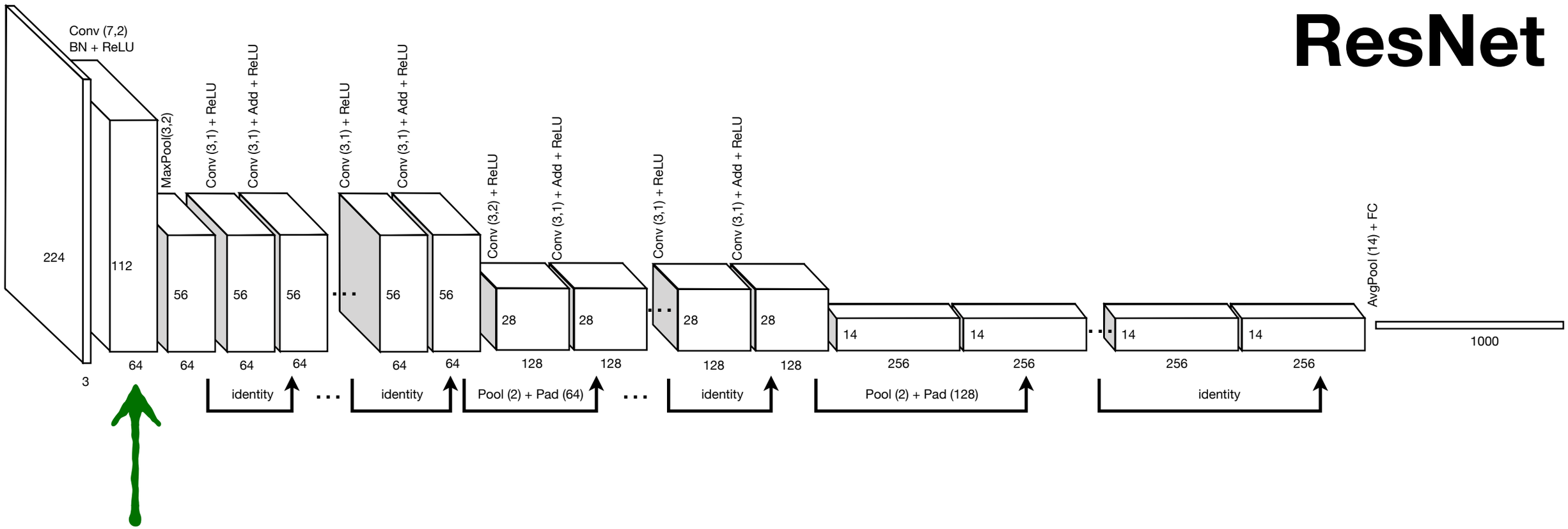


Figure taken from [S. Ioffe & C. Szegedy]

S. Ioffe & C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. ICML 2015

ResNet



Convolution + Batch Normalization + ReLU

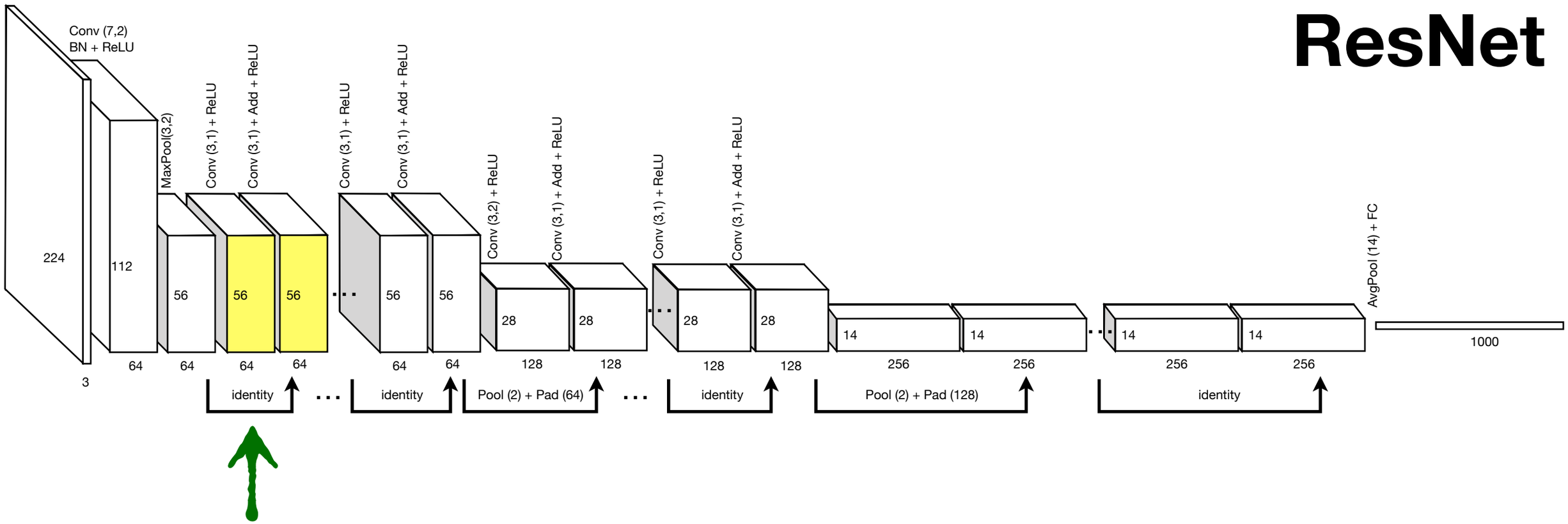
7 x 7 convolution just for the first layer

Stride of 2 reduces the resolution in half

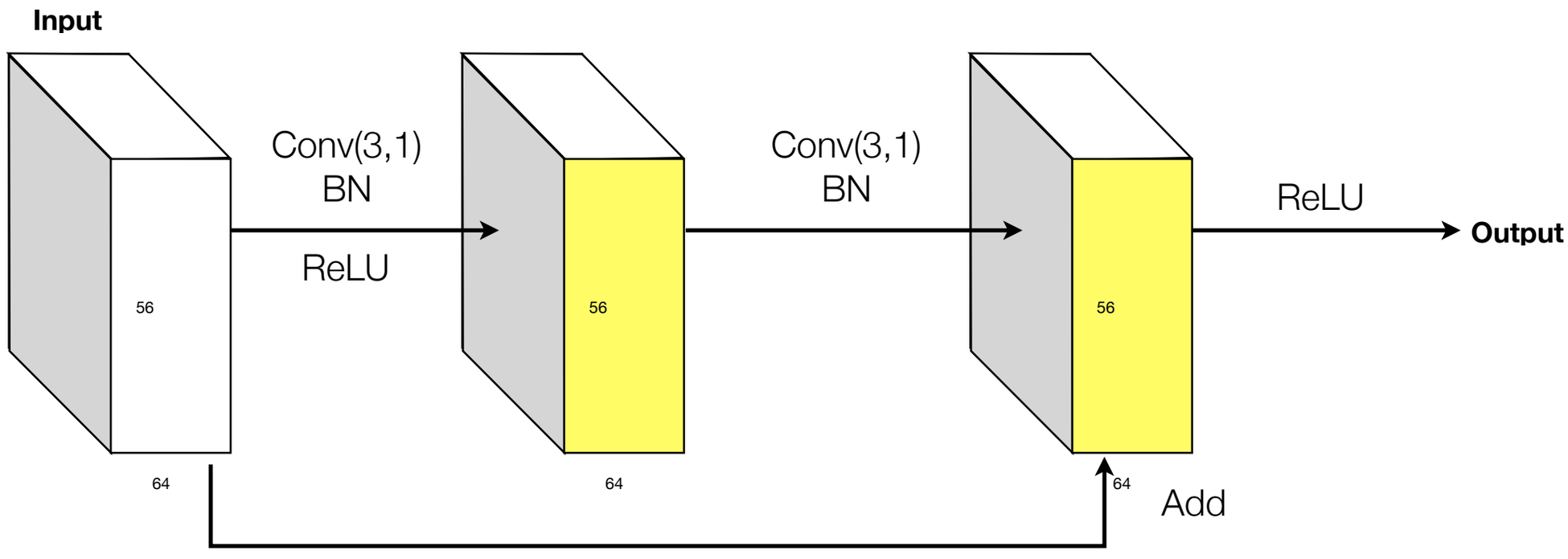
Batch Normalization to scale the responses

ReLU non-linearity

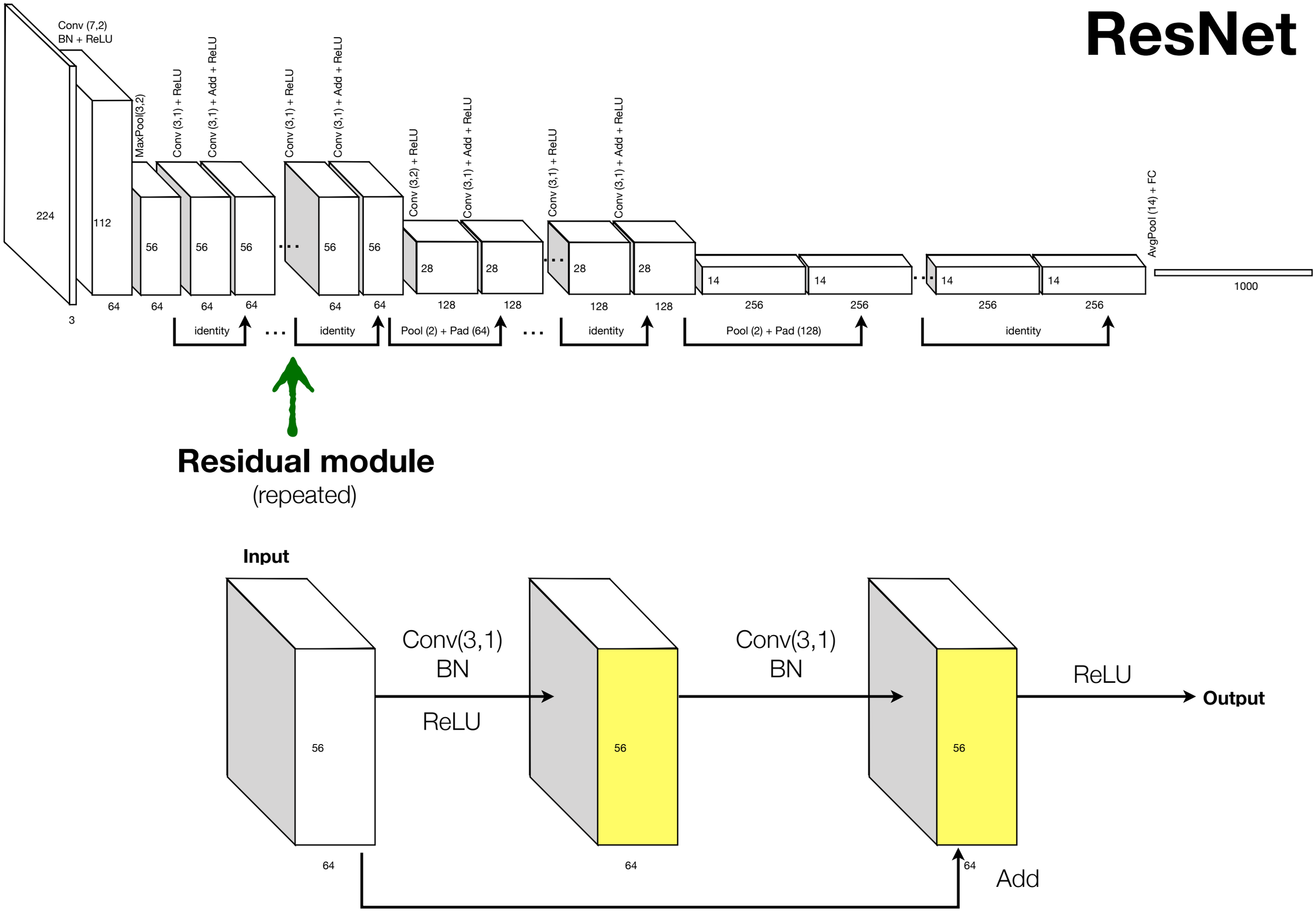
ResNet



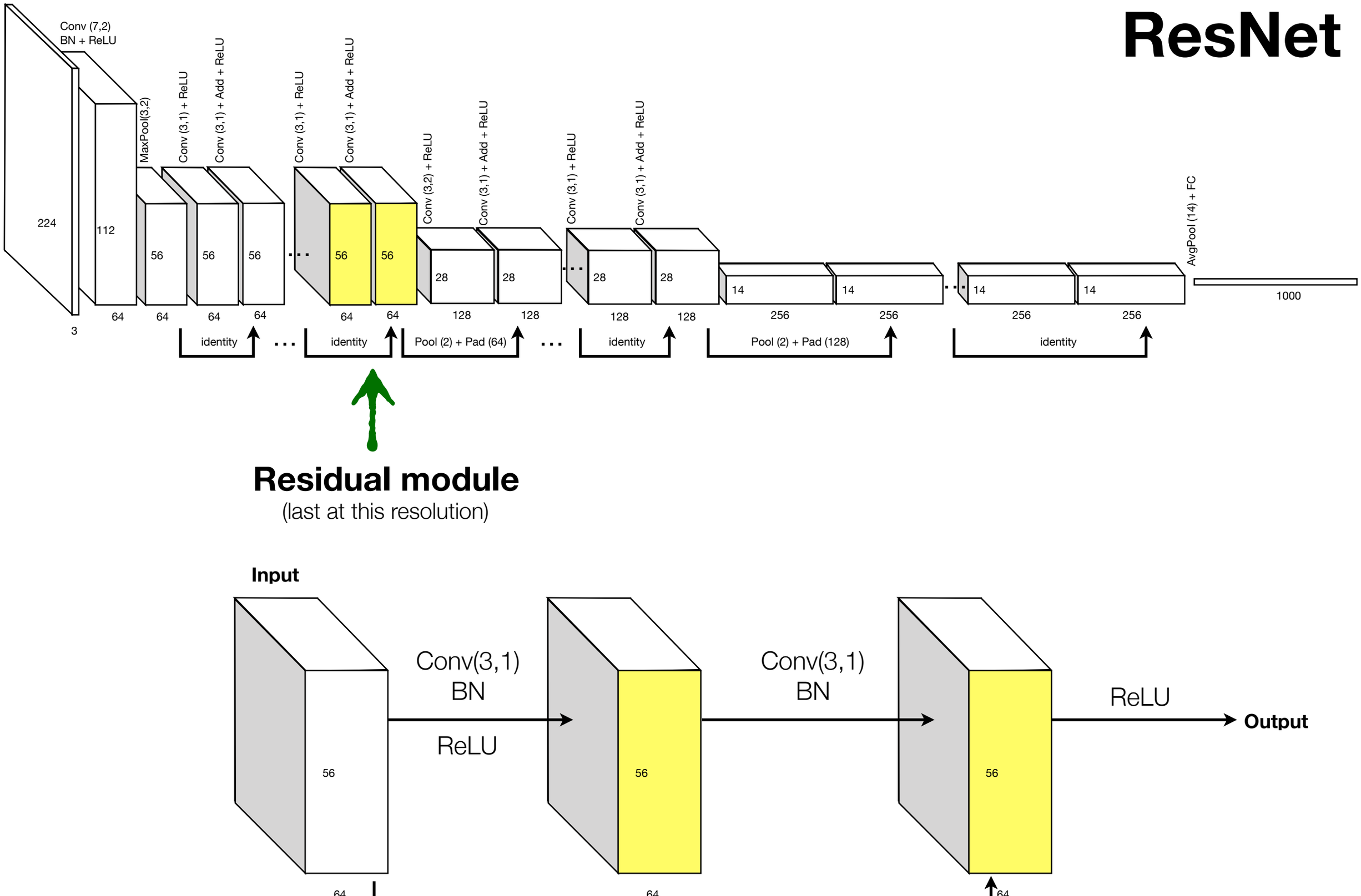
First Residual module



ResNet


Conv (7,2)
BN + ReLU
MaxPool(3,2)
Conv (3,1) + ReLU
Conv (3,1) + Add + ReLU
Conv (3,2) + ReLU
AvgPool (14) + FC
224
112
56
28
14
1000
3
64
128
256
identity
Pool (2) + Pad (64)
Pool (2) + Pad (128)
Residual module
(repeated)
Input
Conv(3,1)
BN
ReLU
Output
Add

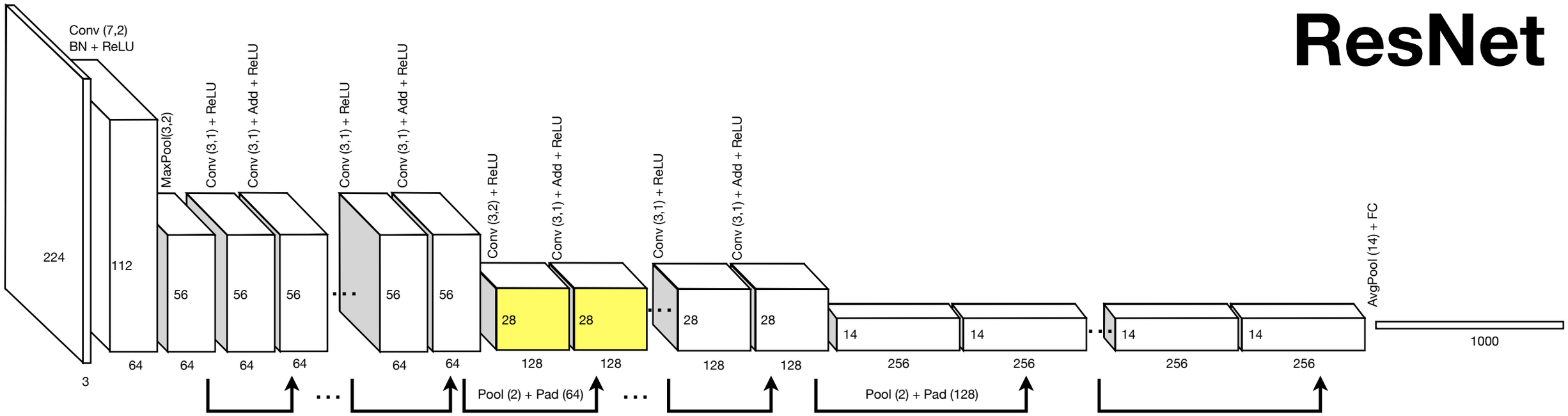
ResNet



Residual module

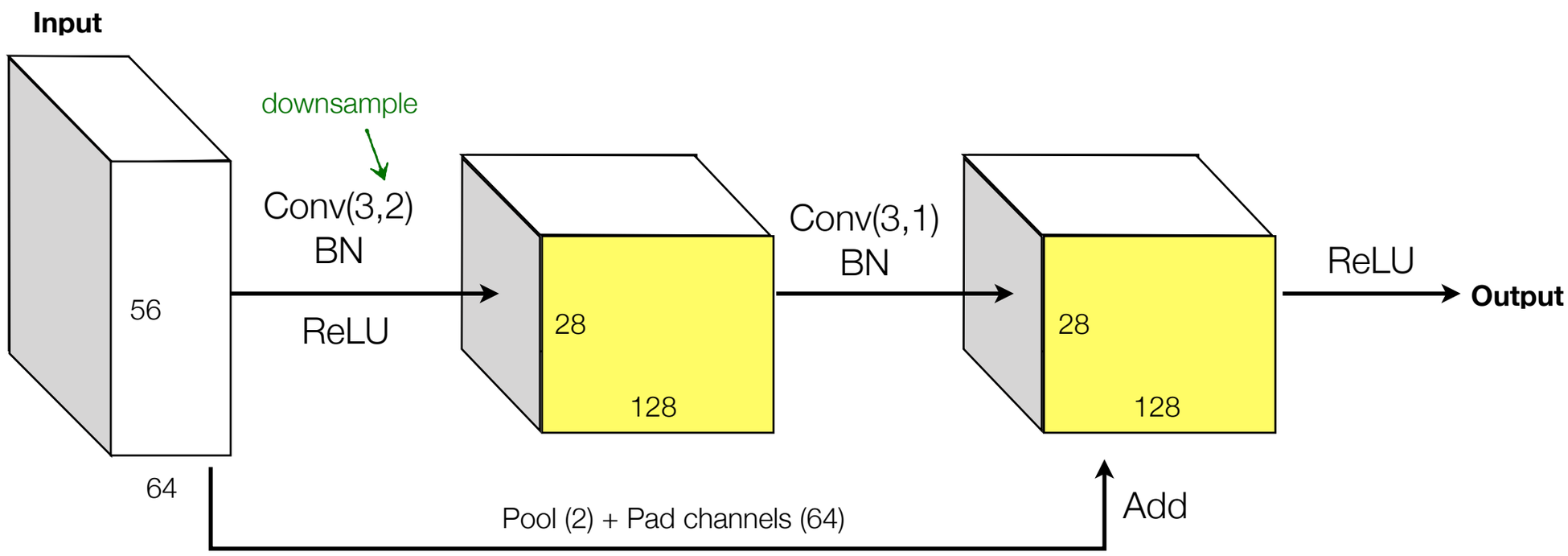
(last at this resolution)

ResNet

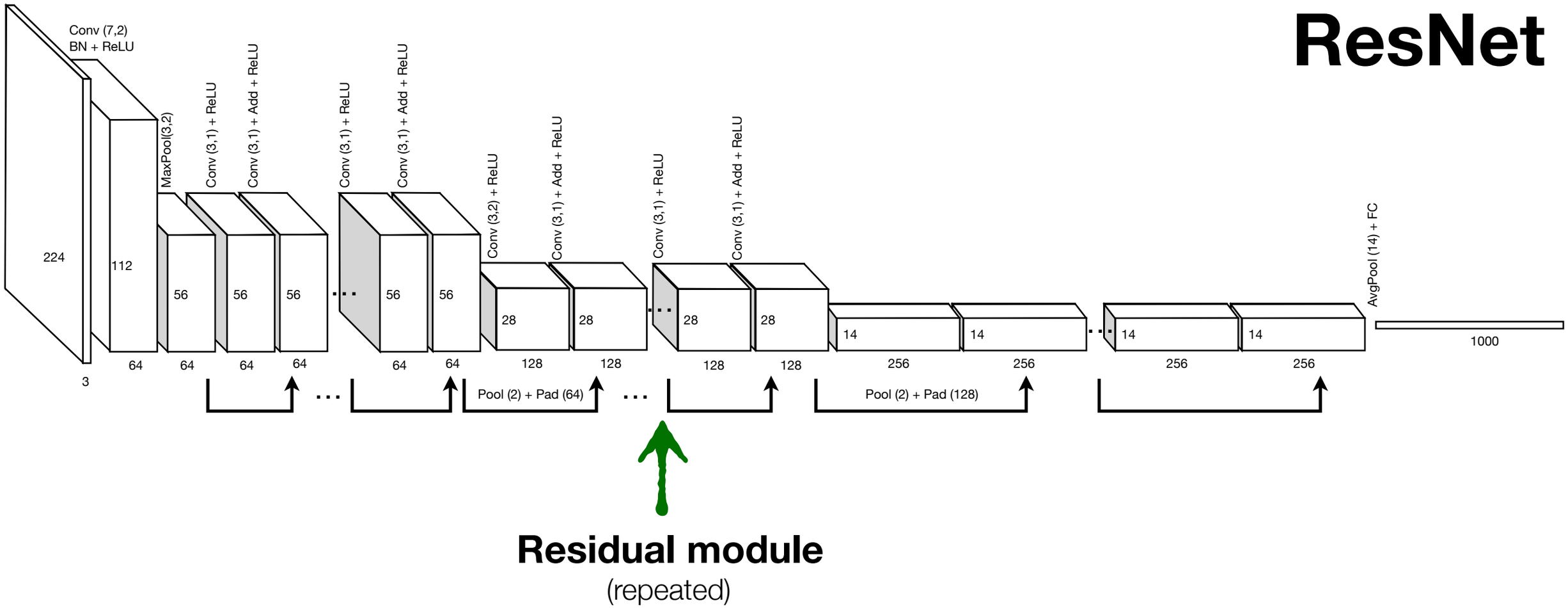


Transitional Residual module

(to half resolution)

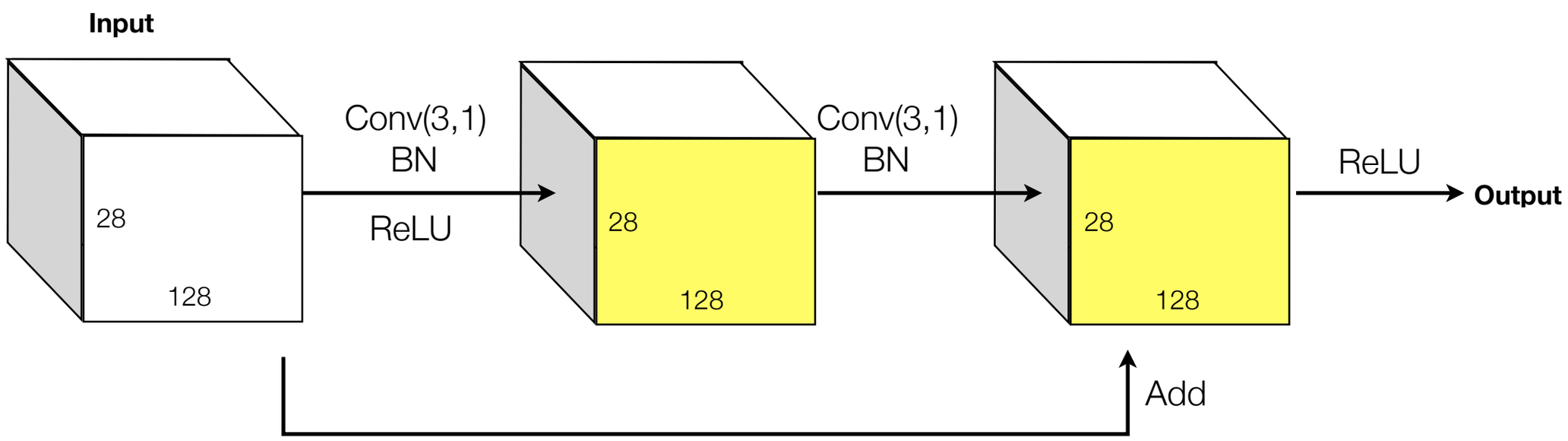


ResNet

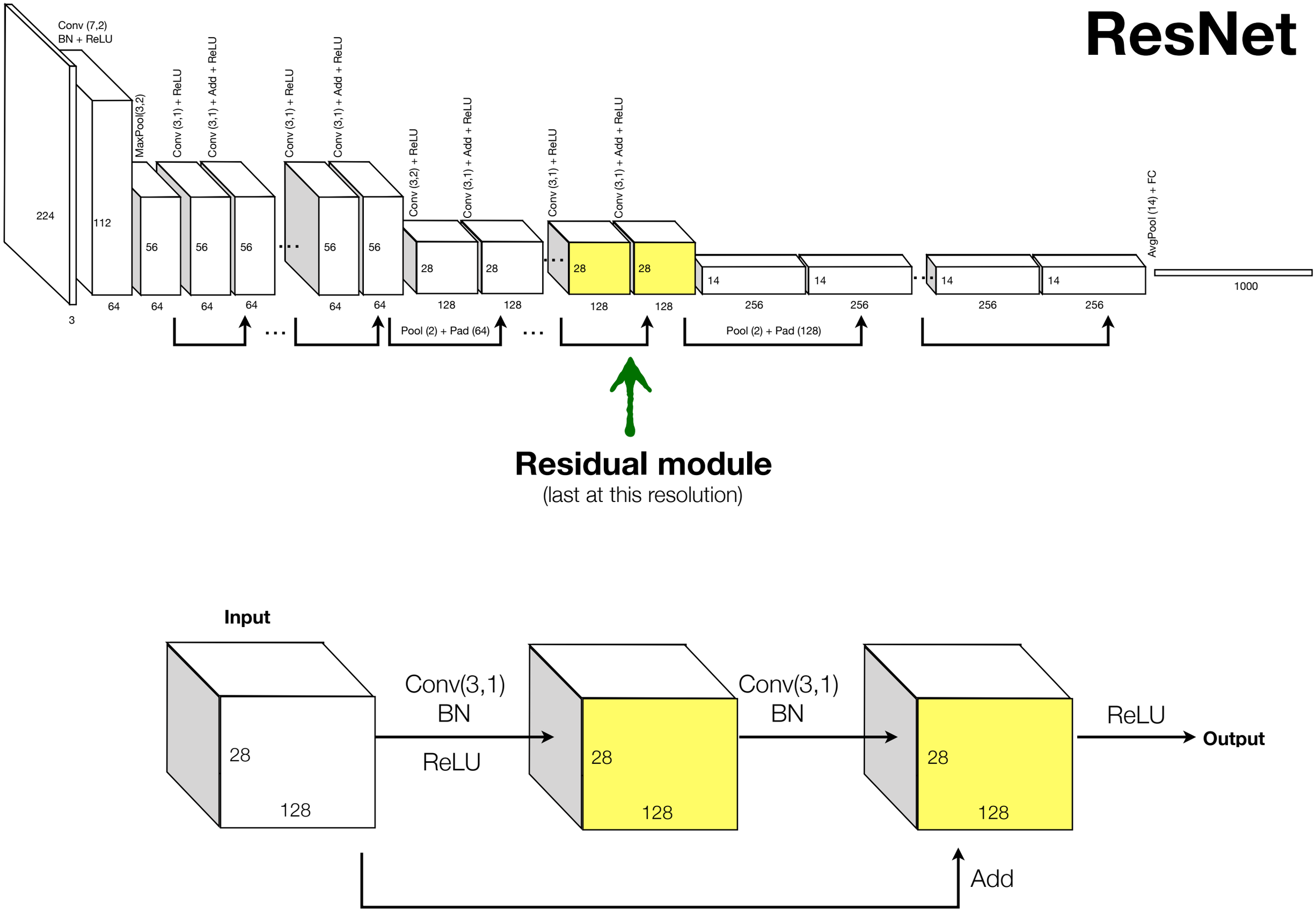


Residual module

(repeated)



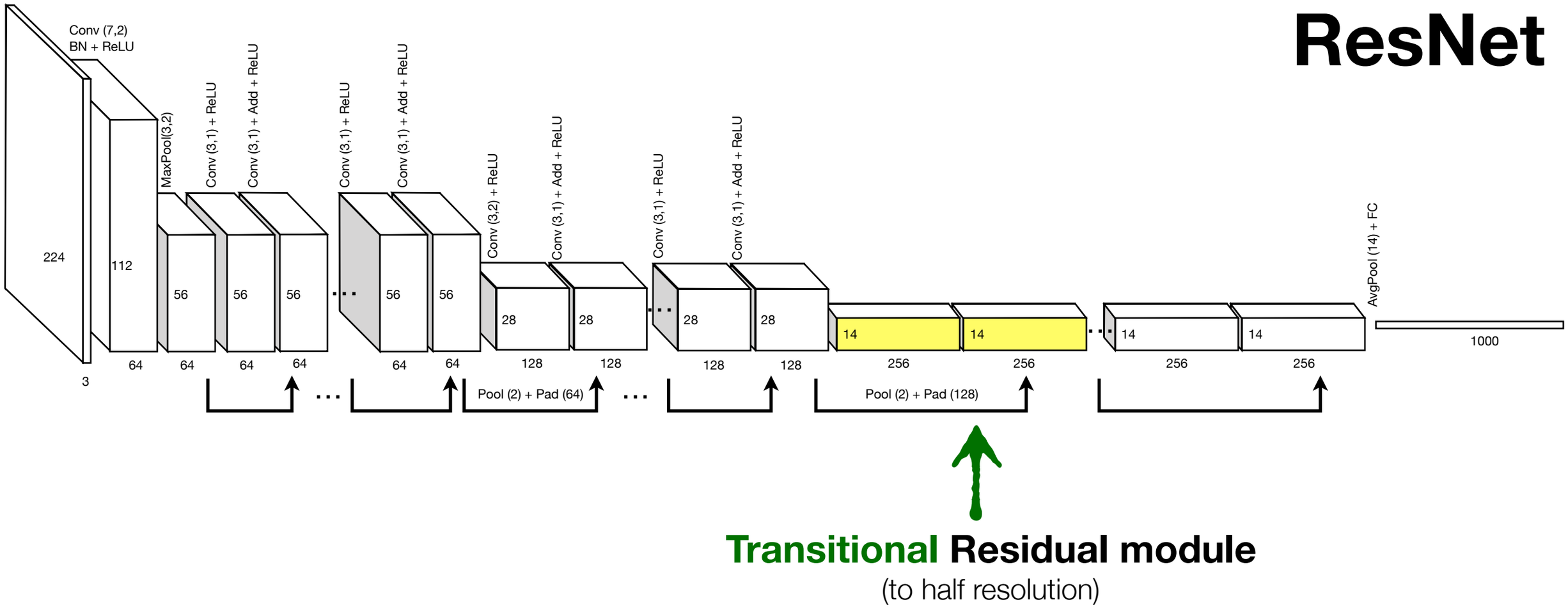
ResNet



Residual module

(last at this resolution)

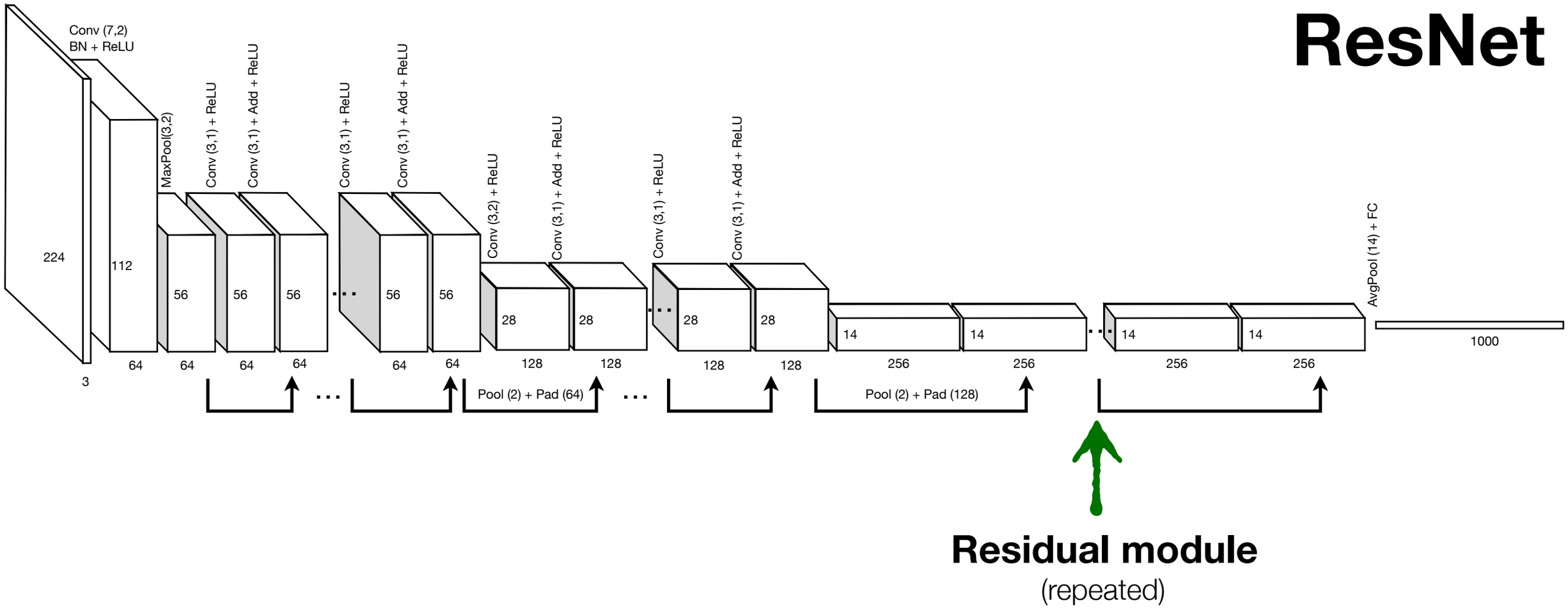
ResNet



Transitional Residual module

(to half resolution)

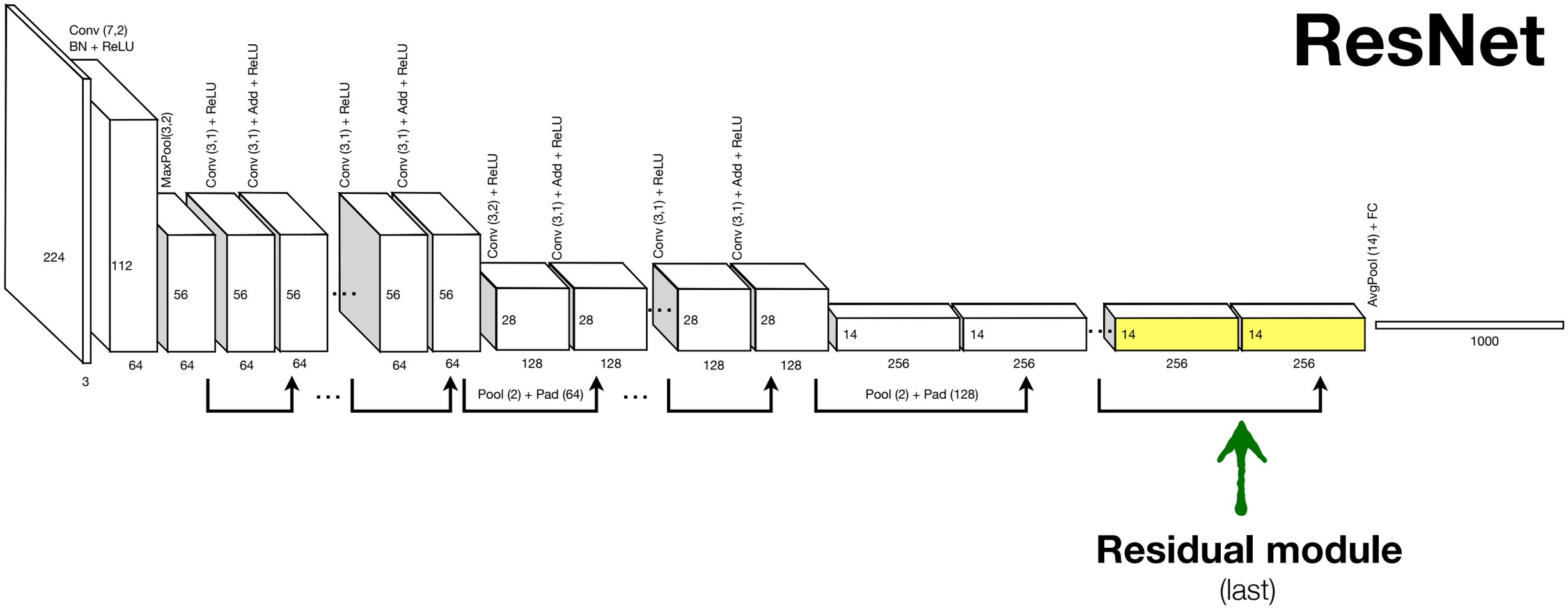
ResNet



Residual module

(repeated)

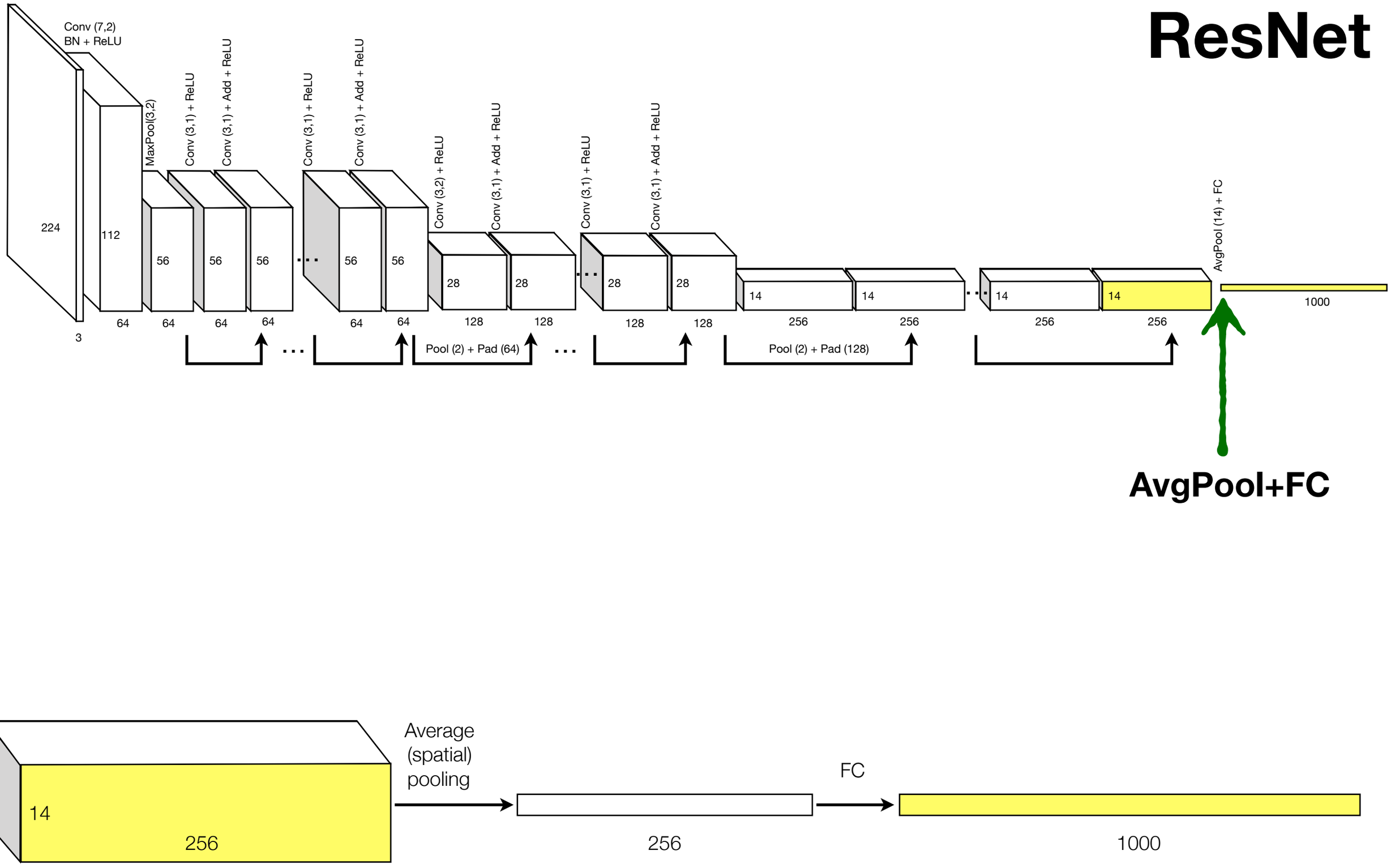
ResNet



Residual module

(last)

ResNet


Conv (7,2)
BN + ReLU
MaxPool(3,2)
Conv (3,1) + ReLU
Conv (3,1) + Add + ReLU
Conv (3,2) + ReLU
AvgPool (14) + FC
224
112
56
28
14
1000
3
64
128
256
Pool (2) + Pad (64)
Pool (2) + Pad (128)
AvgPool+FC
Average
(spatial)
pooling
FC

Training

- All plain/residual nets are trained from scratch
- All plain/residual nets use Batch Normalization
- Standard hyper-parameters & augmentation

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016.

ResNet Results

Single Model ILSVRC2014

method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43†
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

Table 4. Error rates (%) of **single-model** results on the ImageNet validation set (except † reported on the test set).

Multiple Model ILSVRC2014

method	top-5 err. (**test**)
VGG [41] (ILSVRC'14)	7.32
GoogLeNet [44] (ILSVRC'14)	6.66
VGG [41] (v5)	6.8
PReLU-net [13]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	**3.57**

Table 5. Error rates (%) of **ensembles**. The top-5 error is on the test set of ImageNet and reported by the test server.

Important Concepts

- Batch Normalization

- Skip connections

On the Importance of Identity Mapping

From 100 layers to 1000 layers

Identity Mappings in Deep Residual Networks

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun

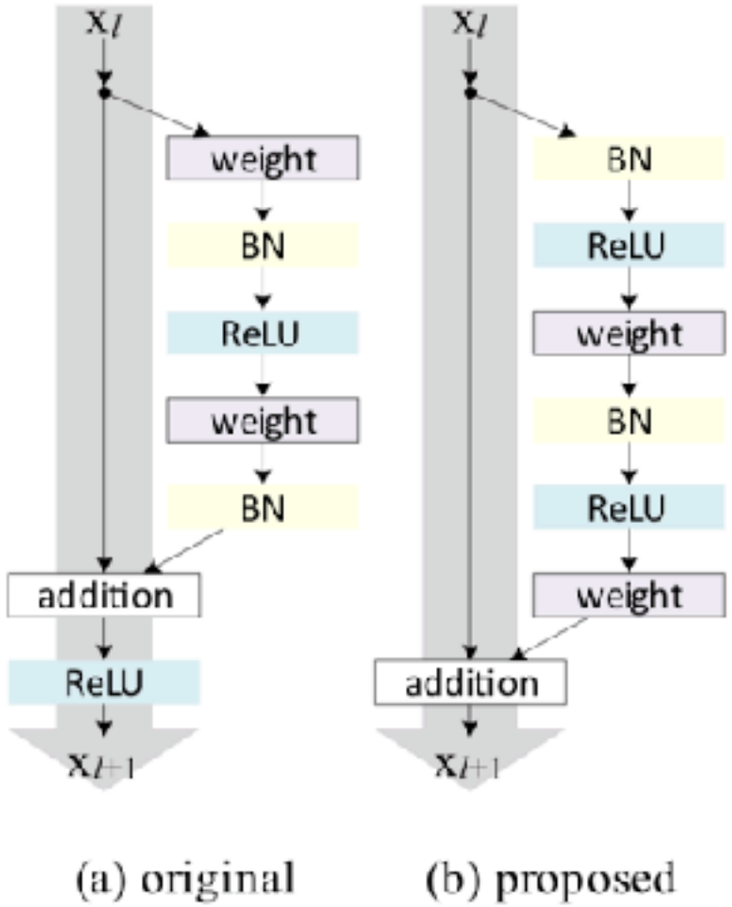
Microsoft Research

Abstract Deep residual networks [1] have emerged as a family of extremely deep architectures showing compelling accuracy and nice convergence behaviors. In this paper, we analyze the propagation formulations behind the residual building blocks, which suggest that the forward and backward signals can be directly propagated from one block to any other block, when using identity mappings as the skip connections and after-addition activation. A series of ablation experiments support the importance of these identity mappings. This motivates us to propose a new residual unit, which makes training easier and improves generalization. We report improved results using a 1001-layer ResNet on CIFAR-10 (4.62% error) and CIFAR-100, and a 200-layer ResNet on ImageNet. Code is available at: https://github.com/KaimingHe/resnet-1k-layers.

1 Introduction

Deep residual networks (ResNets) [1] consist of many stacked "Residual Units". Each unit (Fig. 1 (a)) can be expressed in a general form:

$$\mathbf{y}_l = h(\mathbf{x}_l) + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l),$$
$$\mathbf{x}_{l+1} = f(\mathbf{y}_l),$$



(a) original (b) proposed

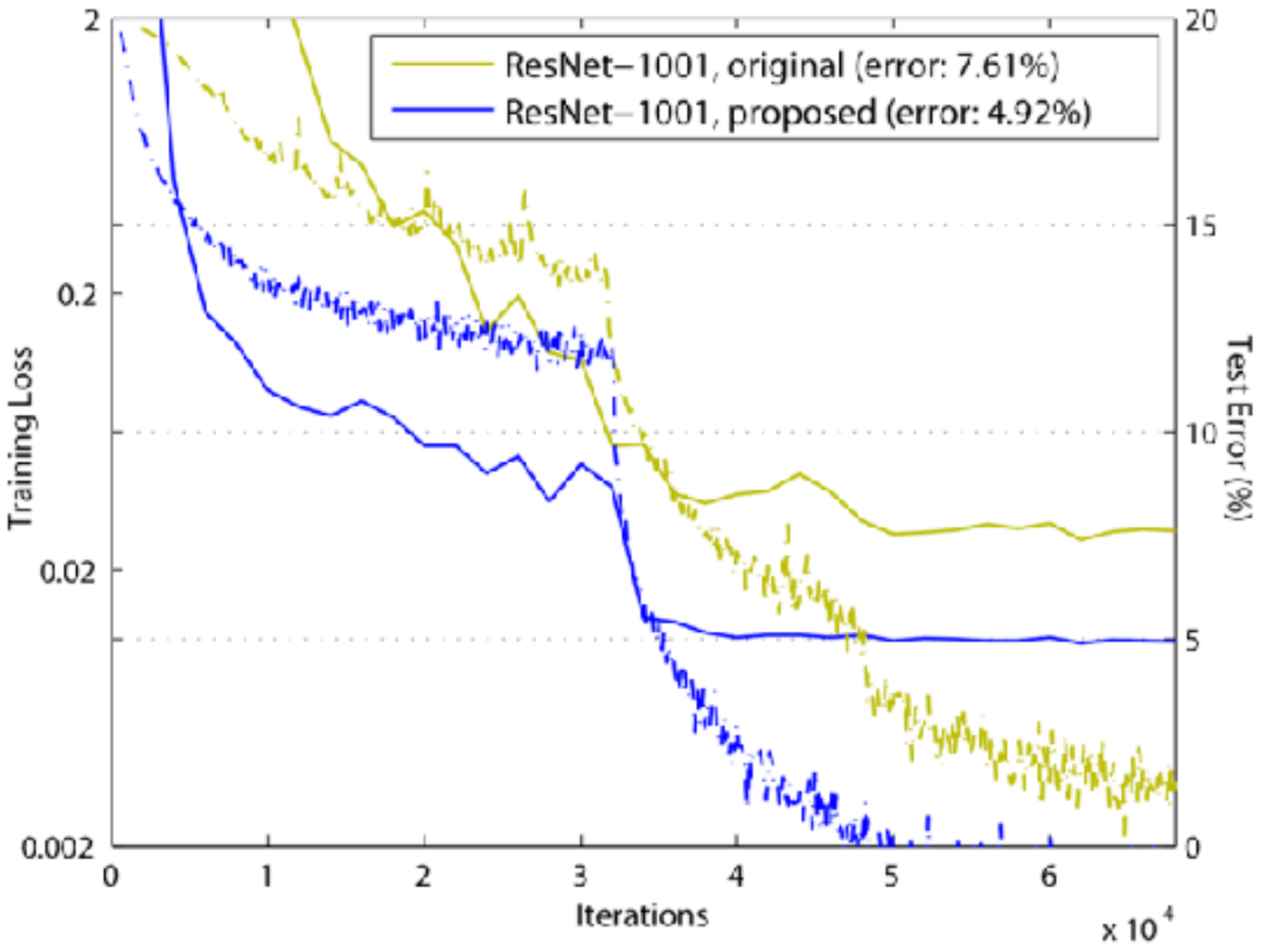
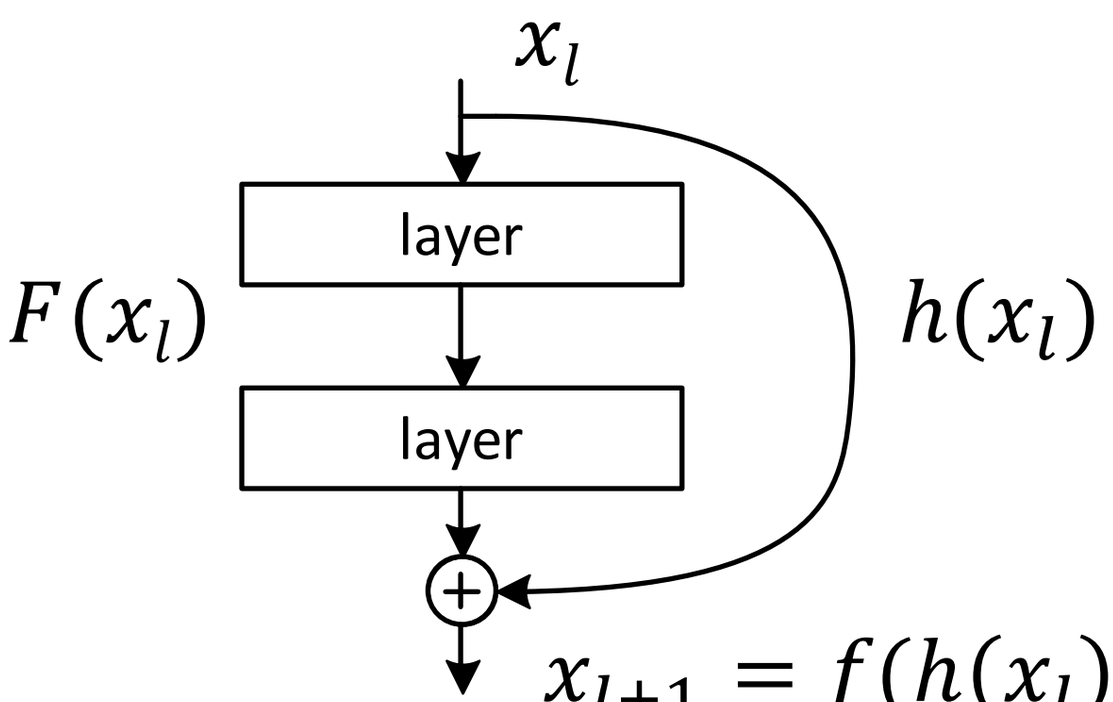


Figure 1. Left: (a) original Residual Unit in [1]; (b) proposed Residual Unit. The grey arrows indicate the easiest paths for the information to propagate, corresponding to the additive term "$\mathbf{x}_l$" in Eqn.(4) (forward propagation) and the additive term "1" in Eqn.(5) (backward propagation). **Right**: training curves on CIFAR-10 of **1001-layer** ResNets. Solid lines denote test error (y-axis on the right), and dashed lines denote training loss (y-axis on the left). The proposed unit makes ResNet-1001 easier to train.

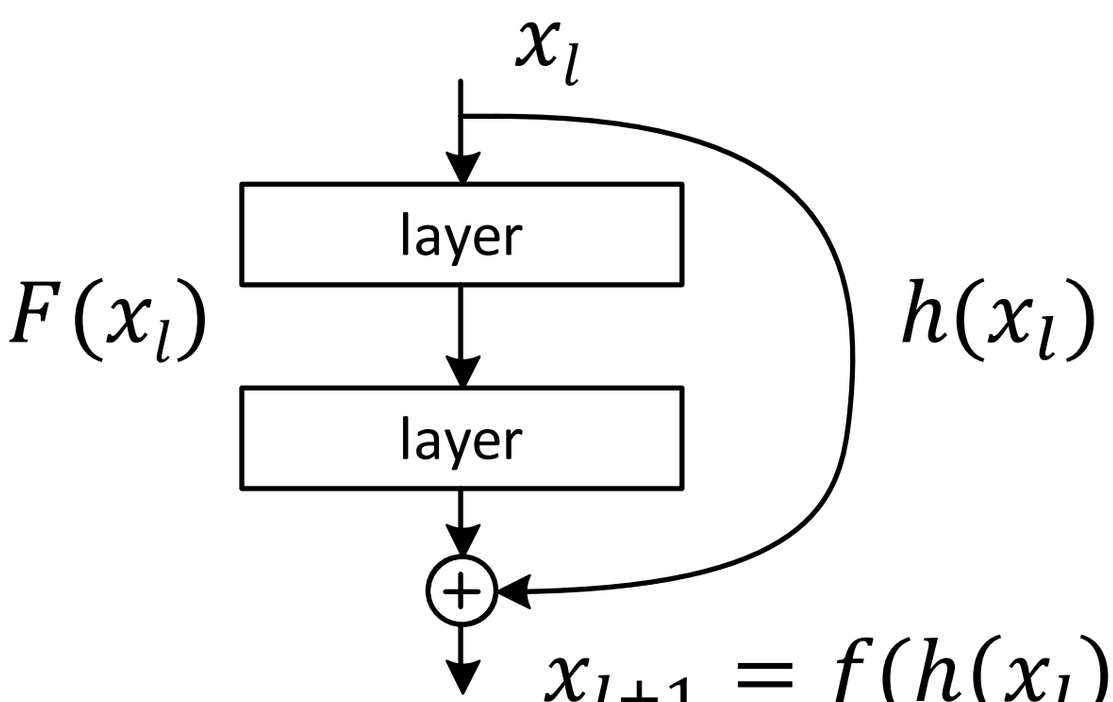
On identity mappings for **optimization**



- shortcut mapping: h = identity
- after-add mapping: f = ReLU

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Identity Mappings in Deep Residual Networks”. arXiv 2016.

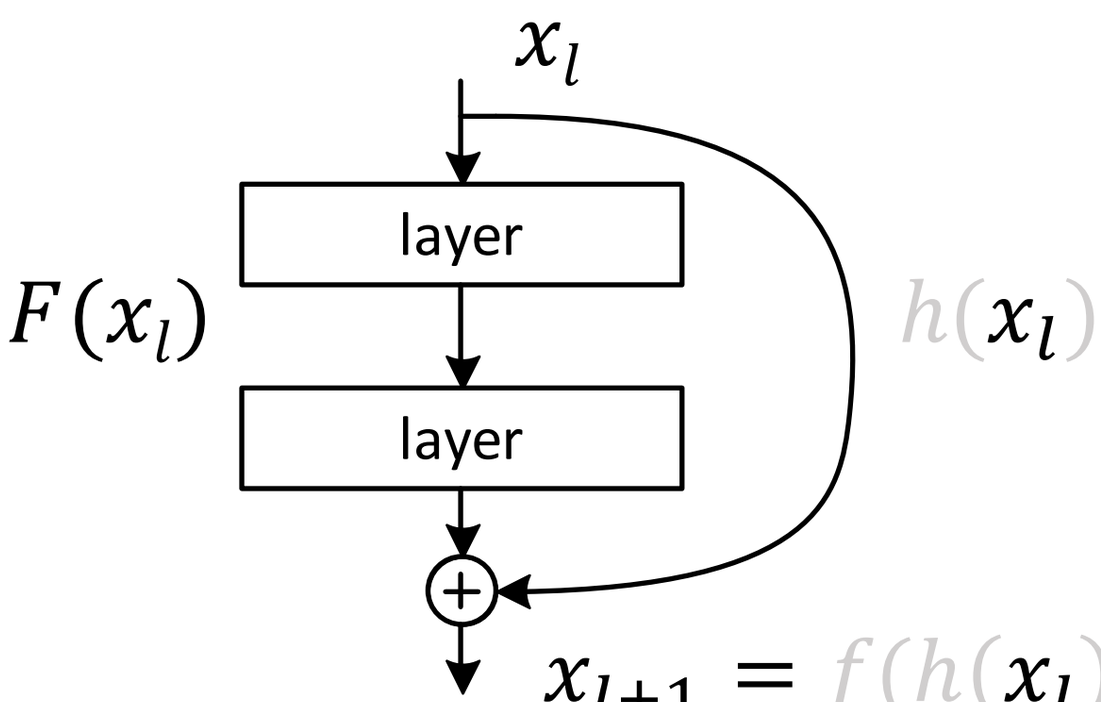
On identity mappings for **optimization**



- shortcut mapping: h = identity
- ~~after-add mapping: f = ReLU~~
- What if f = identity?

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Identity Mappings in Deep Residual Networks”. arXiv 2016.

On identity mappings for **optimization**



- shortcut mapping: h = identity
- ~~after-add mapping: f = ReLU~~
- What if f = identity?

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Identity Mappings in Deep Residual Networks”. arXiv 2016.

Very smooth forward propagation

$$x_{l+1} = x_l + F(x_l)$$

$$x_{l+2} = x_{l+1} + F(x_{l+1})$$

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Identity Mappings in Deep Residual Networks”. arXiv 2016.

Very smooth forward propagation

$$x_{l+1} = x_l + F(x_l)$$

$$x_{l+2} = x_{l+1} + F(x_{l+1})$$

$$x_{l+2} = x_l + F(x_l) + F(x_{l+1})$$

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Identity Mappings in Deep Residual Networks”. arXiv 2016.

Very smooth forward propagation

$$x_{l+1} = x_l + F(x_l)$$

$$x_{l+2} = x_{l+1} + F(x_{l+1})$$

$$x_{l+2} = x_l + F(x_l) + F(x_{l+1})$$

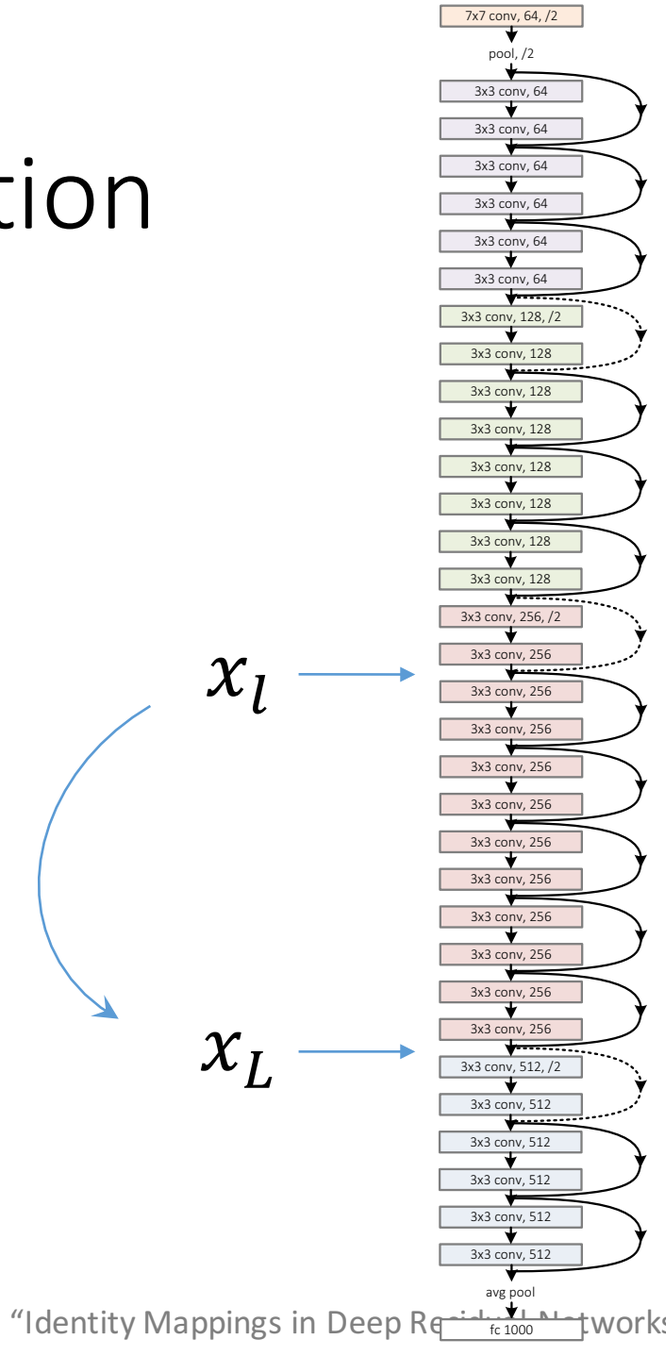
$$x_L = x_l + \sum_{i=l}^{L-1} F(x_i)$$

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Identity Mappings in Deep Residual Networks”. arXiv 2016.

Very smooth forward propagation

$$x_L = x_l + \sum_{i=l}^{L-1} F(x_i)$$

- Any x_l is directly forward-prop to any x_L, plus residual.
- Any x_L is an additive outcome.
  - in contrast to multiplicative: $x_L = \prod_{i=l}^{L-1} W_i x_l$

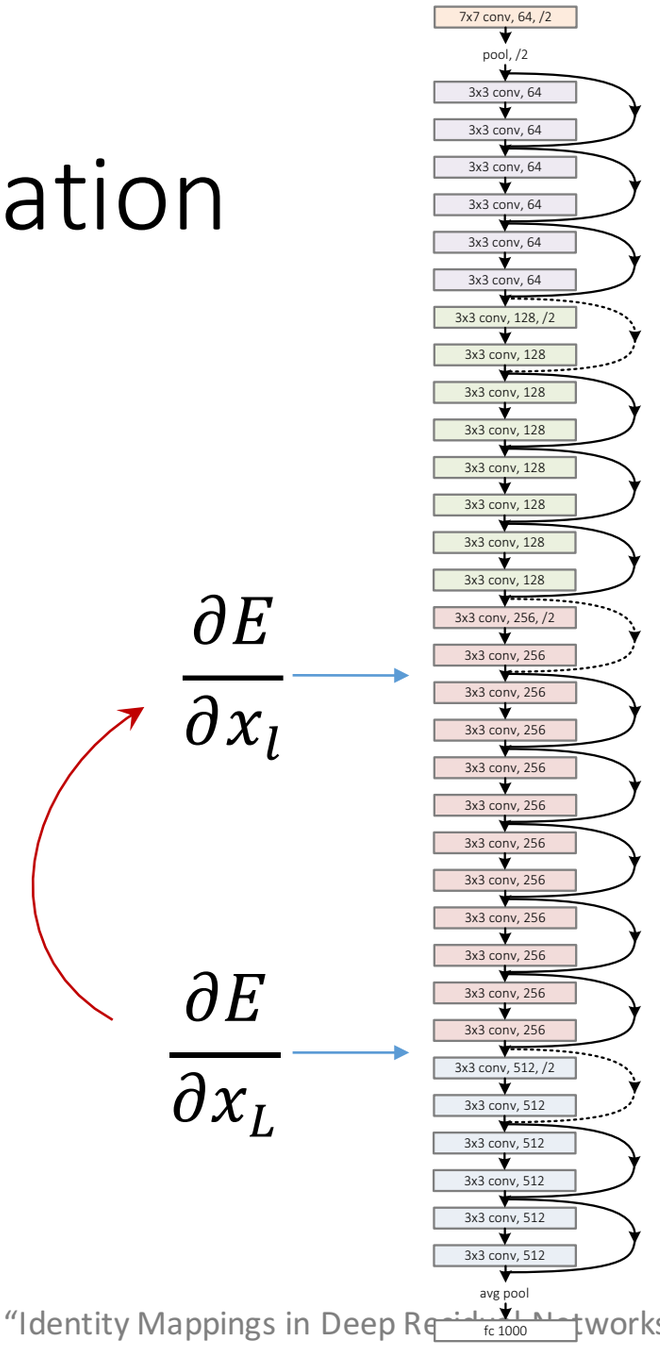


Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Identity Mappings in Deep Residual Networks". arXiv 2016.

Very smooth backward propagation

$$x_L = x_l + \sum_{i=l}^{L-1} F(x_i)$$

$$\frac{\partial E}{\partial x_l} = \frac{\partial E}{\partial x_L}\frac{\partial x_L}{\partial x_l} = \frac{\partial E}{\partial x_L}\left(1 + \frac{\partial}{\partial x_l}\sum_{i=1}^{L-1} F(x_i)\right)$$

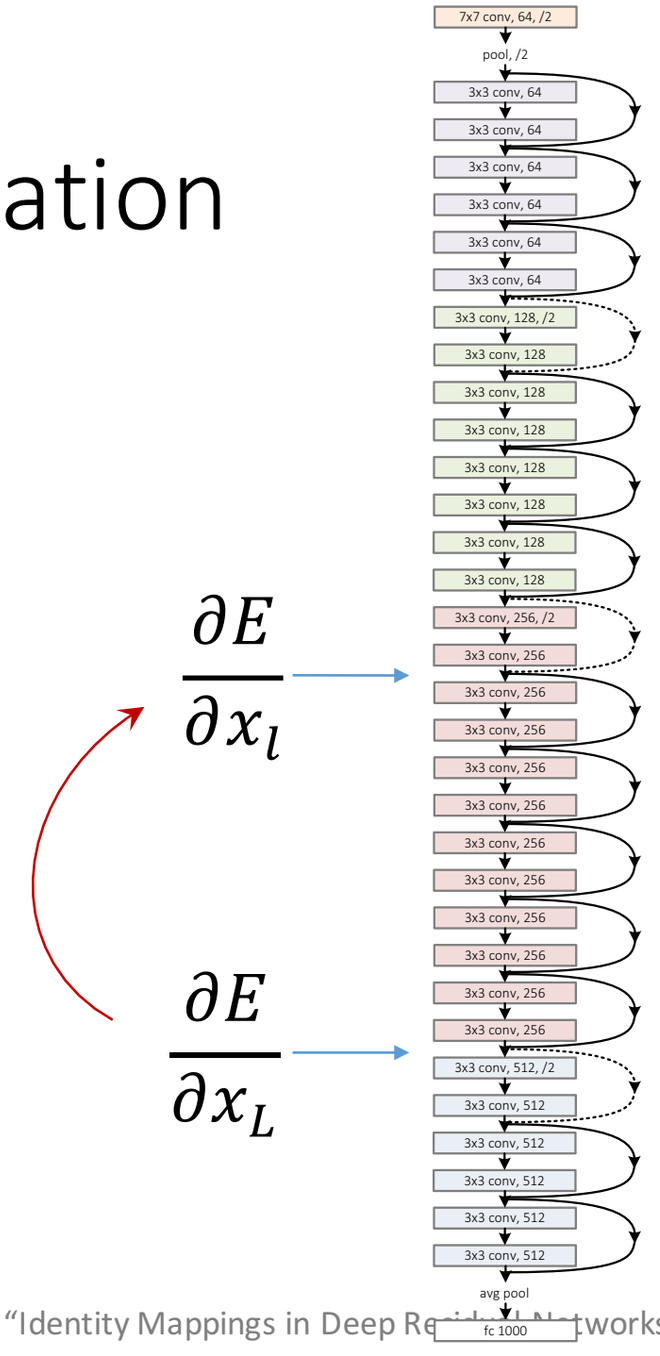


Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Identity Mappings in Deep Residual Networks”. arXiv 2016.

Very smooth backward propagation

$$\frac{\partial E}{\partial x_l} = \frac{\partial E}{\partial x_L}\left(1 + \frac{\partial}{\partial x_l}\sum_{i=1}^{L-1} F(x_i)\right)$$

- Any $\frac{\partial E}{\partial x_L}$ is directly back-prop to any $\frac{\partial E}{\partial x_l}$, plus residual.
- Any $\frac{\partial E}{\partial x_l}$ is additive; unlikely to vanish
  - in contrast to multiplicative: $\frac{\partial E}{\partial x_l} = \prod_{i=l}^{L-1} W_i \frac{\partial E}{\partial x_L}$



Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Identity Mappings in Deep Residual Networks". arXiv 2016.

Residual for every layer

forward: $x_L = x_l + \sum_{i=l}^{L-1} F(x_i)$

Enabled by:

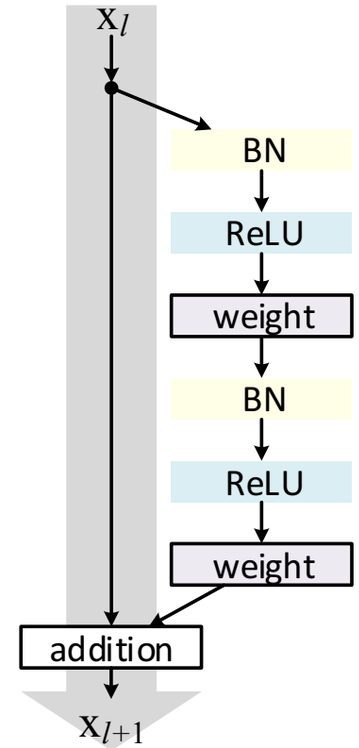
- shortcut mapping: h = identity
- after-add mapping: f = identity

backward: $\frac{\partial E}{\partial x_l} = \frac{\partial E}{\partial x_L}\left(1 + \frac{\partial}{\partial x_l}\sum_{i=1}^{L-1} F(x_i)\right)$

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Identity Mappings in Deep Residual Networks”. arXiv 2016.

Summary of observations

- Keep the shortest path as smooth as possible
  - by making h and f identity
  - forward/backward signals directly flow through this path
- Features of any layers are additive outcomes
- 1000-layer ResNets can be easily trained and have better accuracy



Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Identity Mappings in Deep Residual Networks”. arXiv 2016.