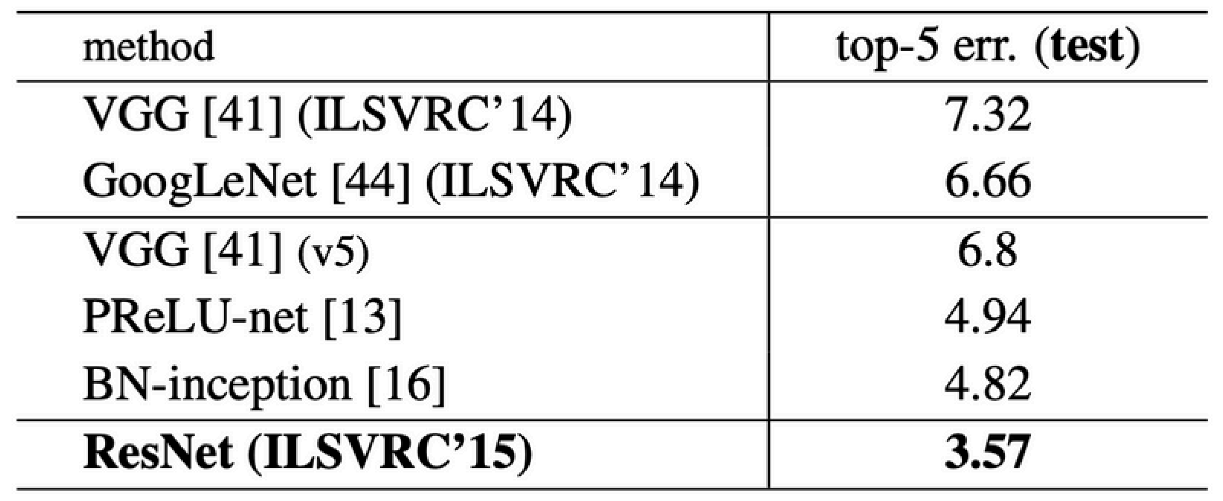
ResNet

by Ei Khaing and Thein Kyaw Lwin

Agenda

What is ResNet ?

- ResNet = Residual Network
- CNN architecture that use skip connections
- Developed by Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun (Microsoft Research)
- Won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2015



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.

Why ResNet ?

- Deeper networks → Better Results ??? No
- Becasue of
  - Vanishing/exploding gradients (gradients = 0 or too large)
  - Degradation problem (accuracy gets worse with depth)
- ResNet solves this by learning residuals instead of direct mappings.

Key Characteristics

- Skip connections or Shortcut Connections
- Solves Degradation Problem (Prevents accuracy drop in very deep models)
- Residual Blocks
    - Based on VGG’s 3×3 convolution design Include:
        - 3×3 Convolutional layers
        - Batch Normalization (BN)
        - ReLU activation
- Modular and Scalable Design
    - Built from repeatable residual blocks (basic or bottleneck), ResNet is easy to scale from 18 to 152 layers without redesigning the architecture.
- Efficient Computation with Bottlenecks
    - Deeper versions use 1×1 convolutions to compress and expand feature maps inside each block, reducing FLOPs and memory usage without sacrificing accuracy.

Architecture

Skip Connections
or
Shortcuts

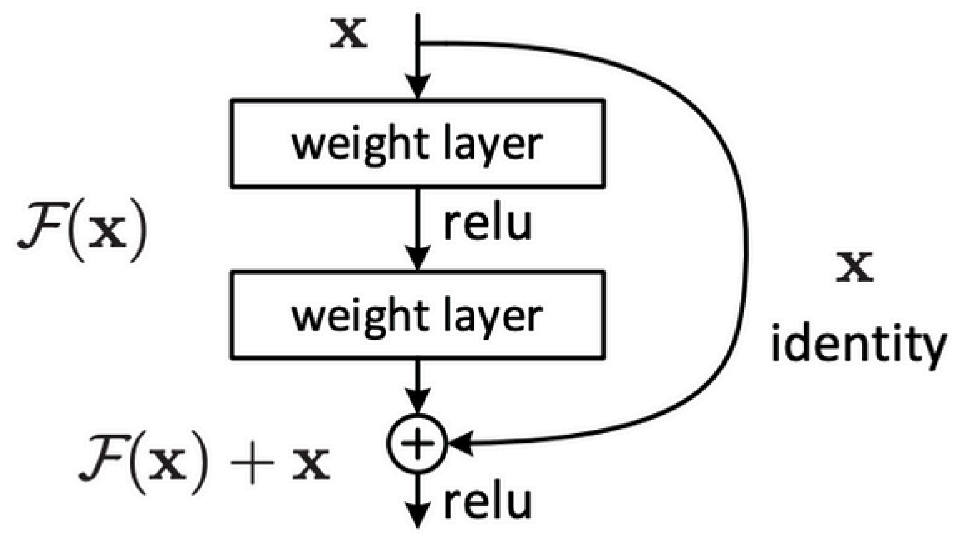


Figure 2. Residual learning: a building block.

$$y = F(x) + x$$

Basic Block

ResNet 18, 34

Bottleneck Block

ResNet 50+

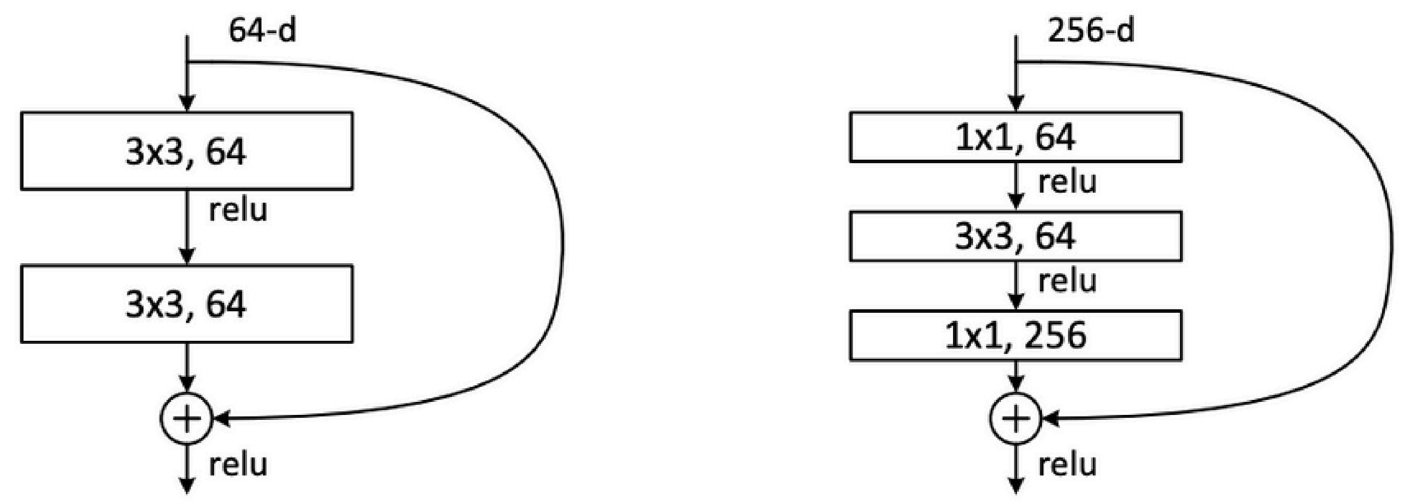


Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on $56{\times}56$ feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.

The very first thing we notice to be different is that there is a direct connection which skips some layers(may vary in different models) in between. This connection is called 'skip connection' and is the core of residual blocks. Due to this skip connection, the output of the layer is not the same now. Without using this skip connection, the input 'x' gets multiplied by the weights of the layer followed by adding a bias term.

Next, this term goes through the activation function, f() and we get our output as H(x).

$$H(x)=f(wx+b)$$
$$\text{or } H(x)=f(x)$$

Now with the introduction of skip connection, the output is changed to

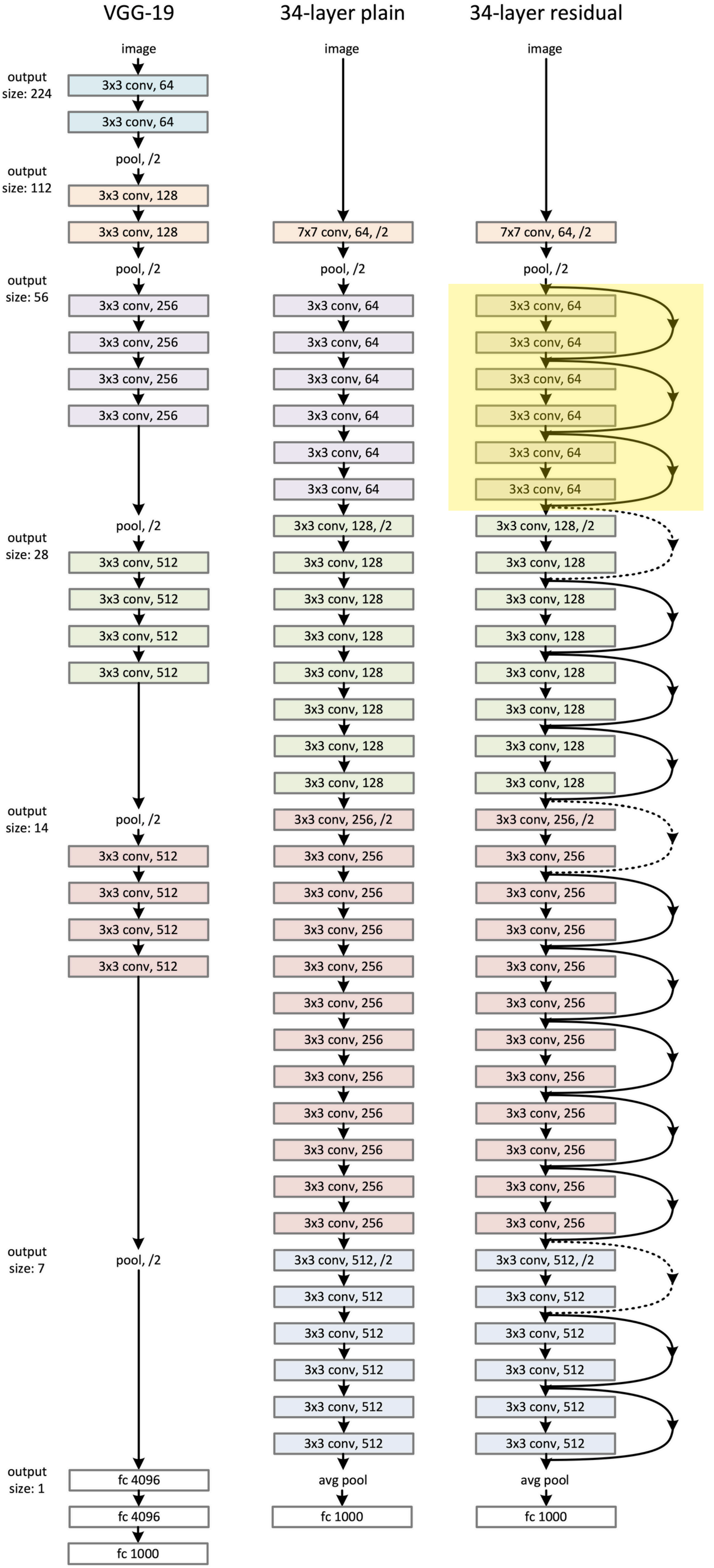
$$H(x)=f(x)+x$$

There appears to be a slight problem with this approach when the dimensions of the input vary from that of the output which can happen with convolutional and pooling layers. In this case, when dimensions of f(x) are different from x, we can take two approaches:

- The skip connection is padded with extra zero entries to increase its dimensions.
- The projection method is used to match the dimension which is done by adding 1×1 convolutional layers to input. In such a case, the output is:

$$H(x)=f(x)+w1.x$$

Here we add an additional parameter w1 whereas no additional parameter is added when using the first approach.



Total Layers =1 (Conv1) + (number of blocks × 2) + 1 (FC layer) ← Basic Block

Total Layers =1 (Conv1) + (number of blocks × 3) + 1 (FC layer) ← Bottleneck Block

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Downsampling is performed by
conv3_1,
conv4_1, and
conv5_1 with **a stride of 2**

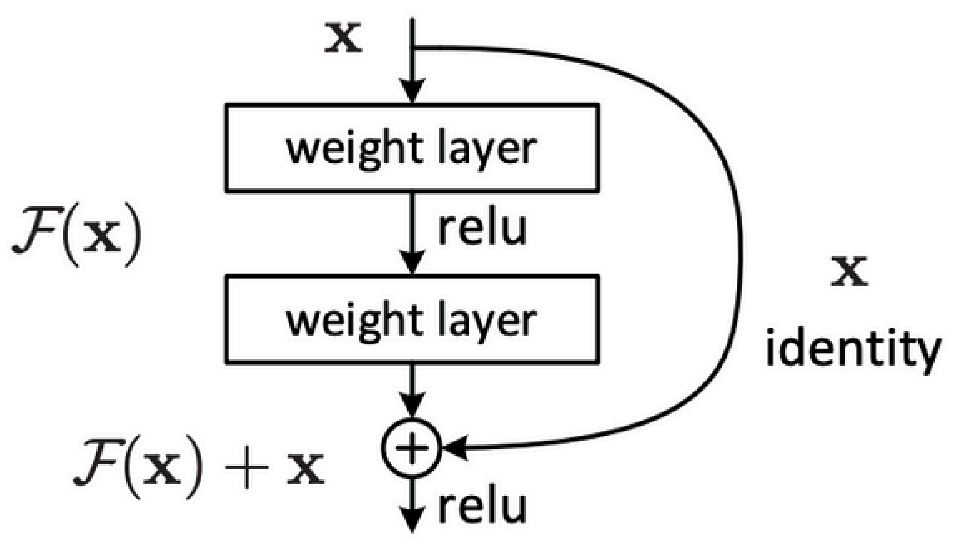
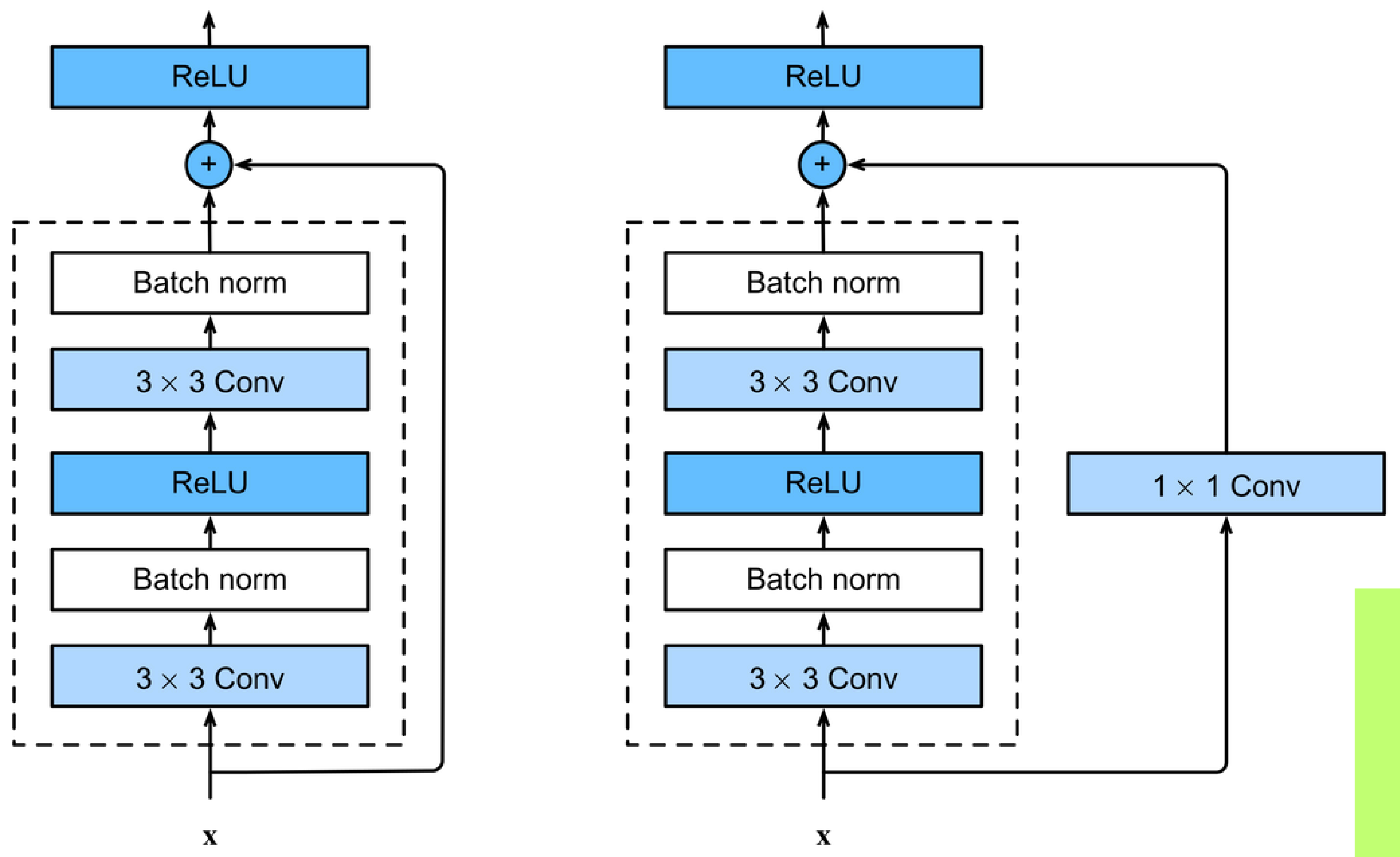


Figure 2. Residual learning: a building block.



Downsampling is performed by
conv3_1,
conv4_1, and
conv5_1 with a stride of 2

```python
def resnet_block(inputs, filters, kernel_size, strides):
    x = layers.Conv2D(filters, kernel_size, strides=strides, padding='same')(inputs)
    x = layers.BatchNormalization()(x)
    x = layers.Activation('relu')(x)
    x = layers.Conv2D(filters, kernel_size, padding='same')(x)
    x = layers.BatchNormalization()(x)

    if strides > 1:
        inputs = layers.Conv2D(filters, 1, strides=strides, padding='same')(inputs)

    x = layers.Add()([x, inputs])
    x = layers.Activation('relu')(x)
    return x
```

ResNet Variants

- ResNet-18 and ResNet-34
- ResNet-50 (used as a benchmark in computer vision due to its solid balance of accuracy, speed, and simplicity)
- ResNet-101 and ResNet-152 (intended for high-performance tasks that require greater depth)
    - ResNet-101 has been used as a backbone in object detection models like Faster R-CNN
    - ResNet-152 achieved a top-5 error rate of 3.57% on ImageNet, outperforming earlier architectures like VGG
- ResNetV2 (This version differs from the original (V1) ResNet primarily by using batch normalization before each weight layer)
- ResNeXt and Wide ResNet (WRN)

Limitations

- Learning False Associations: The model might prioritize surface-level patterns instead of genuine semantic characteristics, resulting in inadequate generalization in certain instances.
- High resource usage: ResNet-50 requires more memory and computing power than lightweight models, which can make it less suitable for mobile devices or real-time applications.
- ResNet can be prone to overfitting on small datasets without appropriate regularization techniques

References

- https://en.wikipedia.org/wiki/Residual_neural_network
- https://arxiv.org/pdf/1512.03385
- https://files.batistalab.com/teaching/attachments/chem584/Resnet.pdf
- https://notebooklm.google.com/notebook/7712db30-3afd-4229-b43b-0978213d5ec8

Why ?

- Why Stride 2 at the beginning of each stage ?
- How to understand FLOPS ?

$$P = \left\lfloor \frac{K - 1}{2} \right\rfloor$$