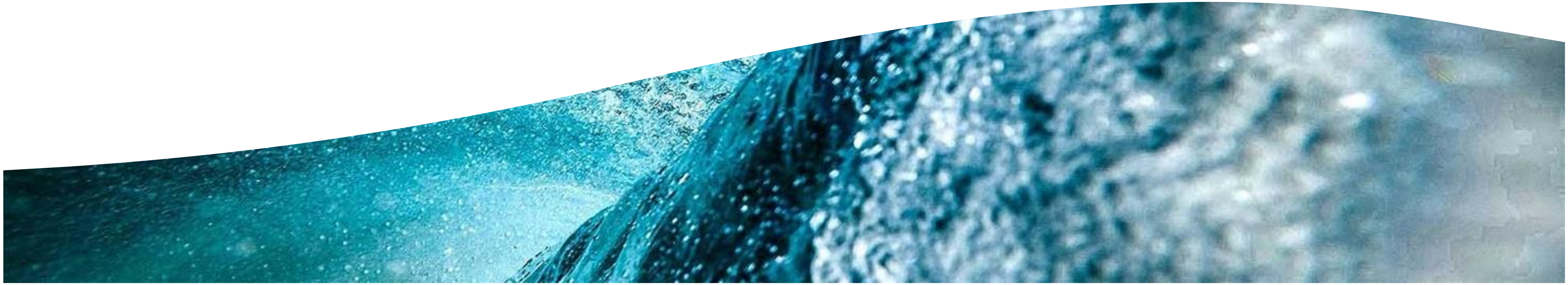
ResNet Overview*

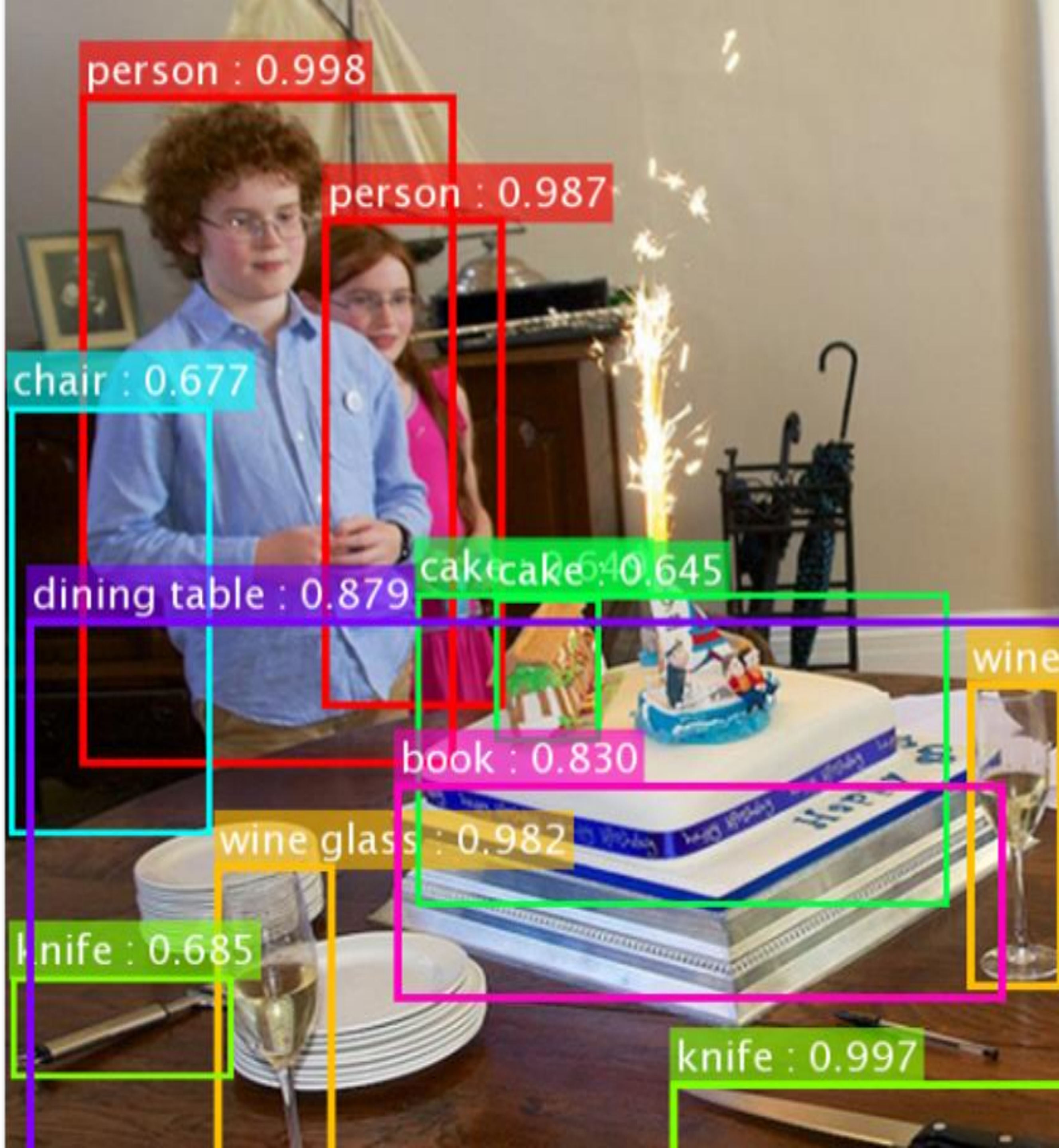
Implementation using Keras



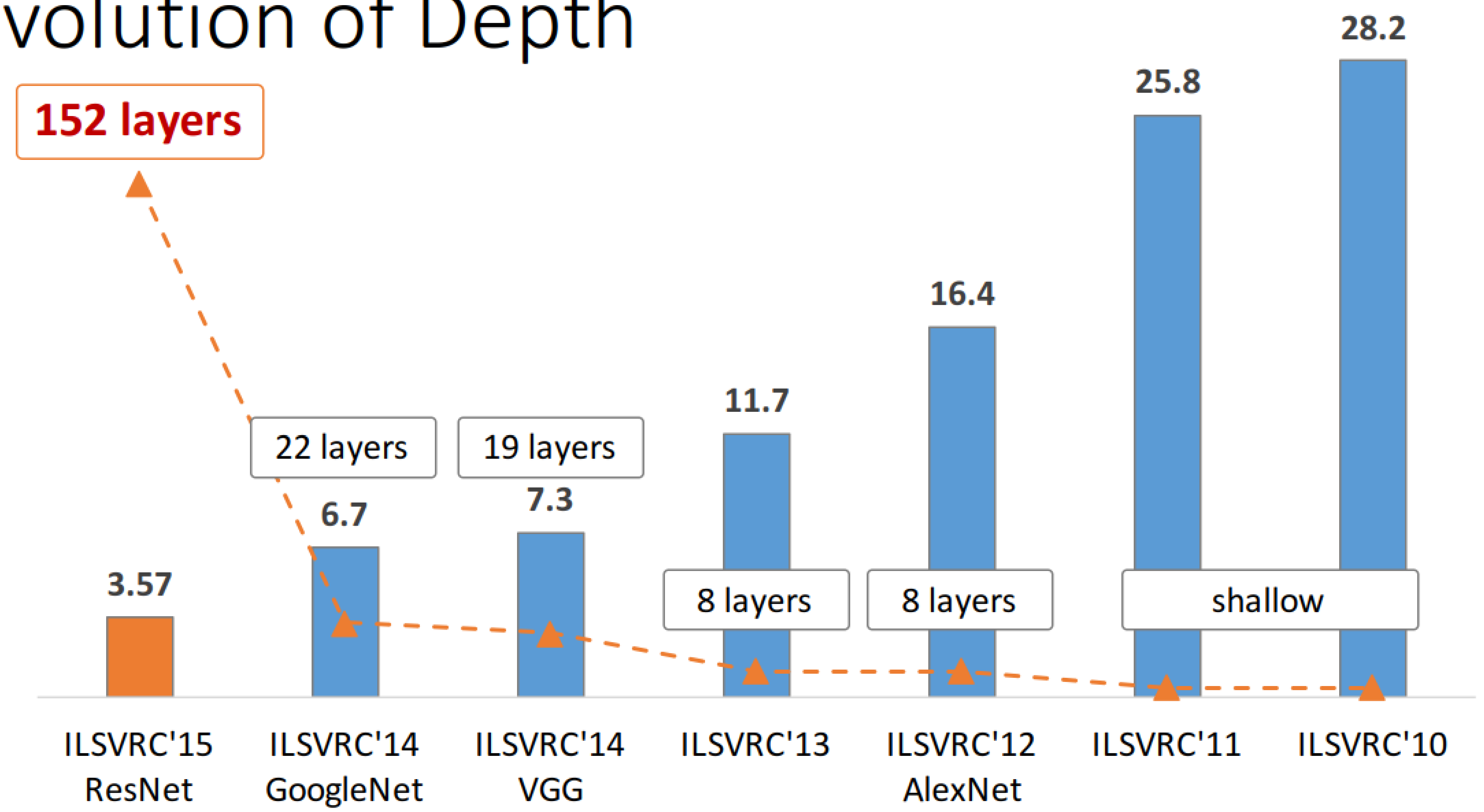
* Slides are modified from the original paper Deep Residual Learning for Image Recognition by Kaiming He, Xiangyu Zhang , Shaoqing Ren and Jian Sun

Winning Model

- Won 1st place in the ILSVRC 2015 classification competition with top-5 error rate of 3.57%
- Won the 1st place in ILSVRC and COCO 2015 competition in ImageNet Detection, ImageNet localization, Coco detection and Coco segmentation.
- Replacing VGG-16 layers in Faster R-CNN with ResNet-101. They observed a relative improvements of 28%
- Efficiently trained networks with 100 and 1000 layers.



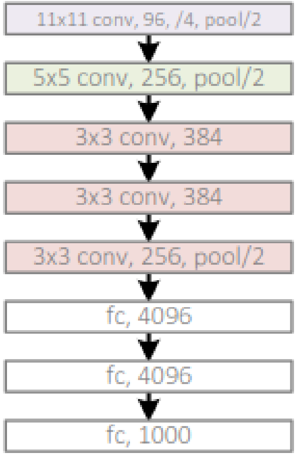
Revolution of Depth



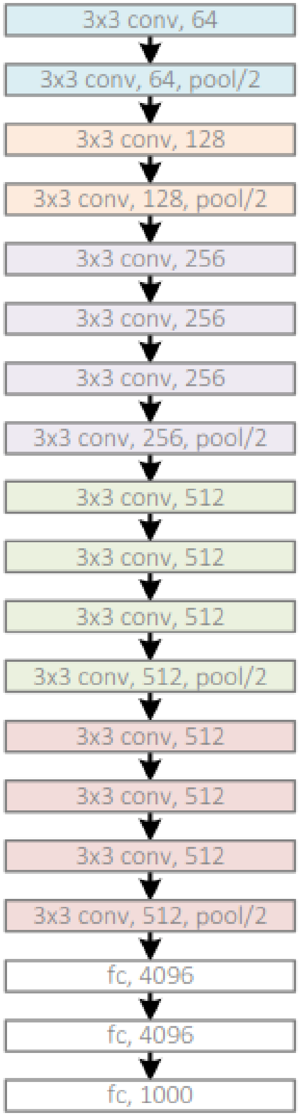
ImageNet Classification top-5 error (%)

Revolution of Depth

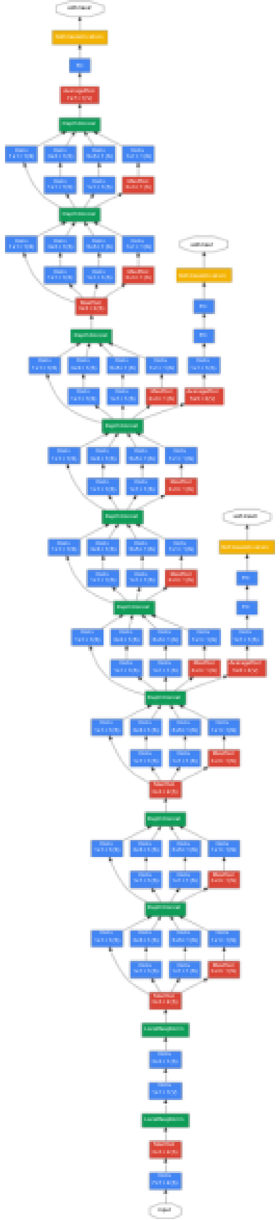
AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)

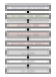


GoogleNet, 22 layers
(ILSVRC 2014)

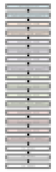


Revolution of Depth

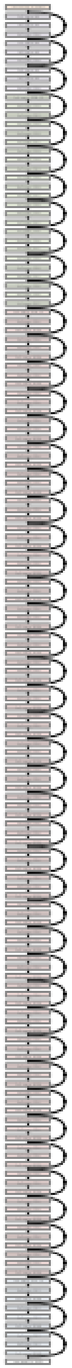
AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)



ResNet, 152 layers
(ILSVRC 2015)



Problem

- With the network depth increasing, accuracy gets saturated and then degrades rapidly.
- A huge barrier to training NN is vanishing gradients: very deep networks often have a gradient signal that goes to zero quickly, thus making gradient descent unbearably slow

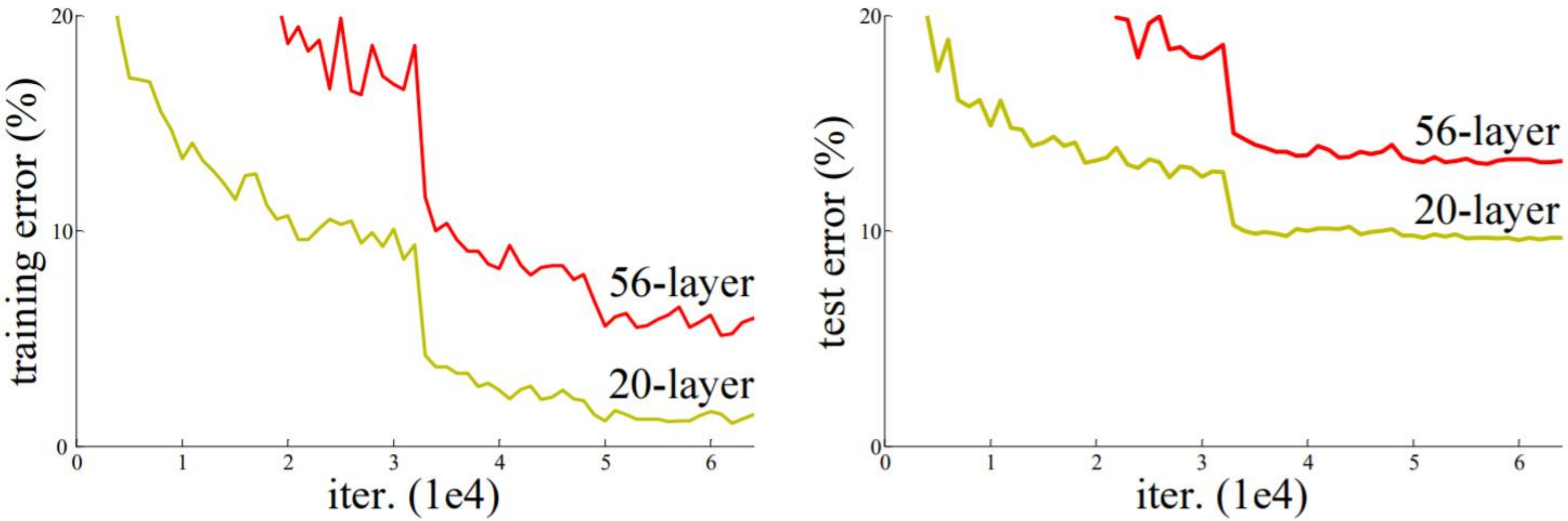
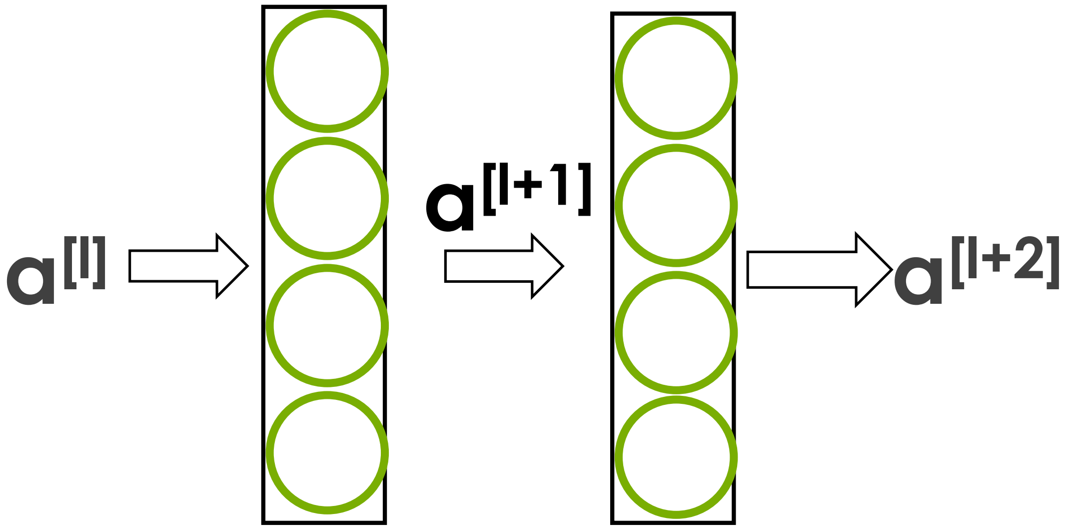


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Sequential Block



$$z^{[l+1]} = W^{[l+1]} * a^{[l]} + b^{[l+1]}$$

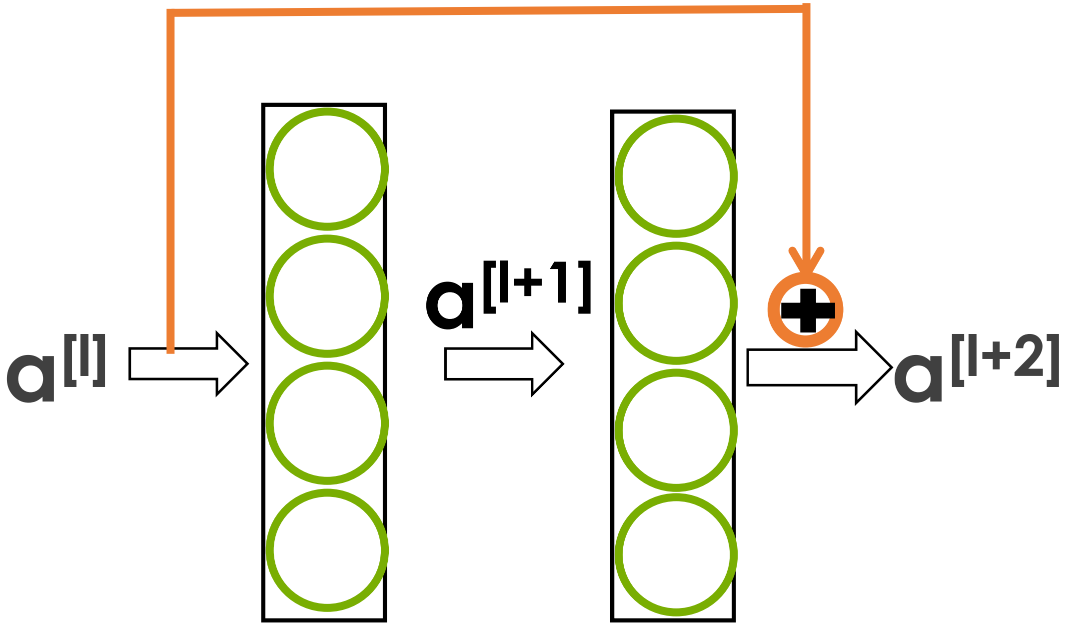
$$a^{[l+1]} = g(z^{[l+1]})$$

$$z^{[l+2]} = W^{[l+2]} * a^{[l+1]} + b^{[l+2]}$$

$$a^{[l+2]} = g(z^{[l+2]})$$

$a^{[l]}$ -> Linear -> ReLu -> Linear -> ReLu

Residual Block ("shortcut" or "skip connection")



$$z^{[l+1]} = W^{[l+1]} * a^{[l]} + b^{[l+1]}$$

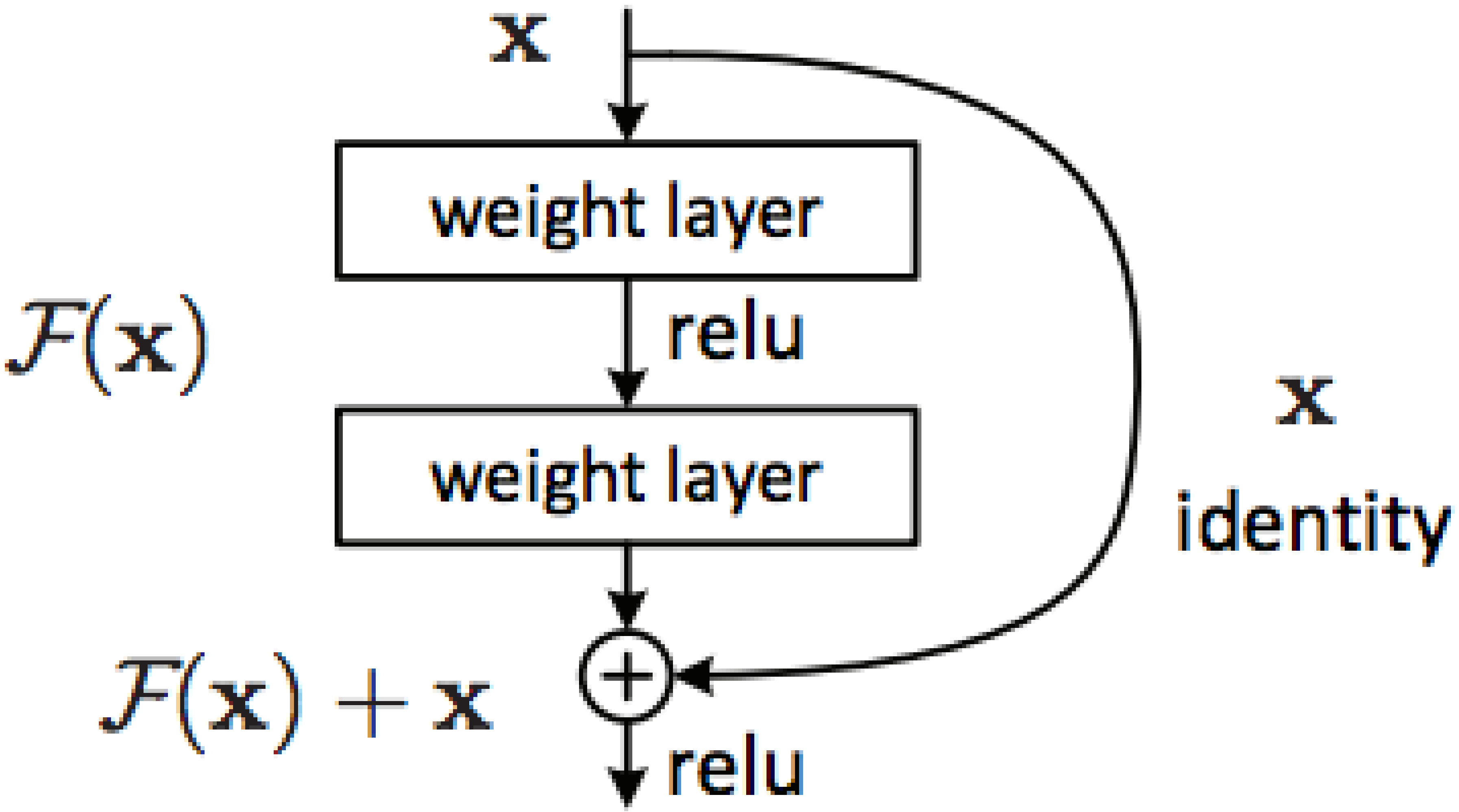
$$a^{[l+1]} = g(z^{[l+1]})$$

$$z^{[l+2]} = W^{[l+2]} * a^{[l+1]} + b^{[l+2]}$$

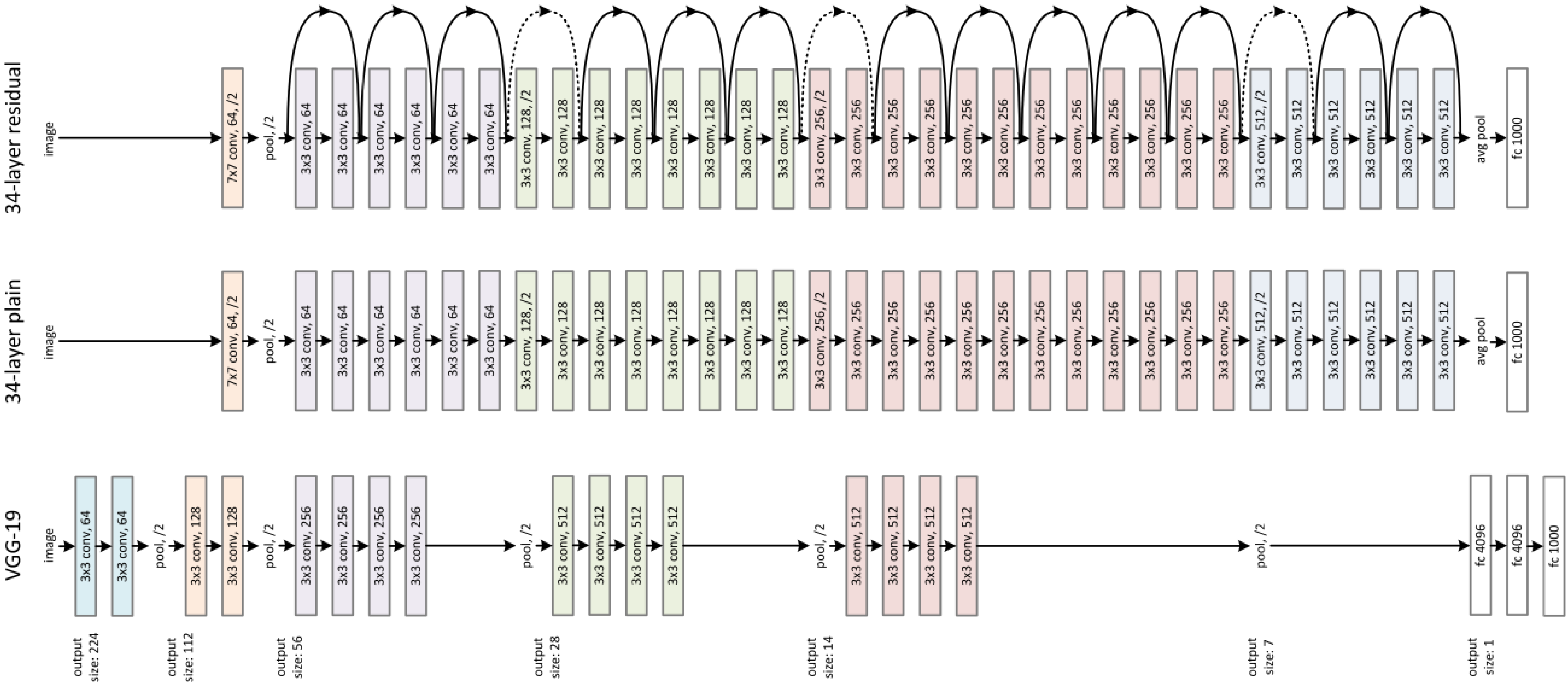
$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$

$a^{[l]}$ -> Linear -> ReLu -> Linear -> ReLu

Building block

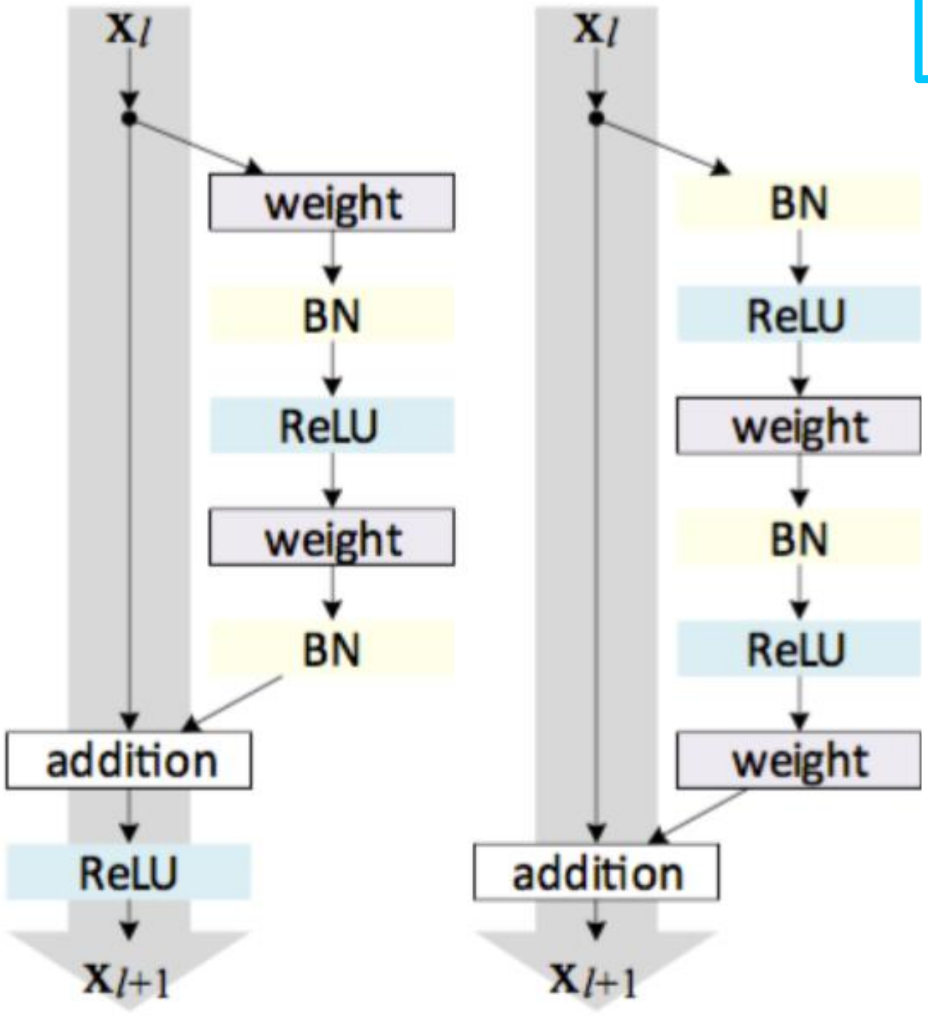


ResNet Structure

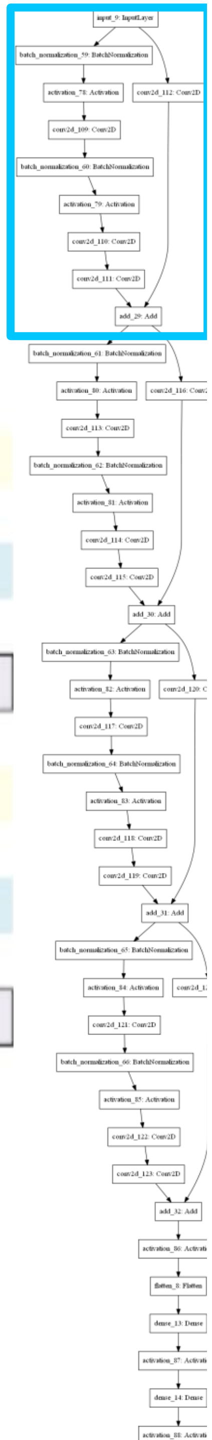


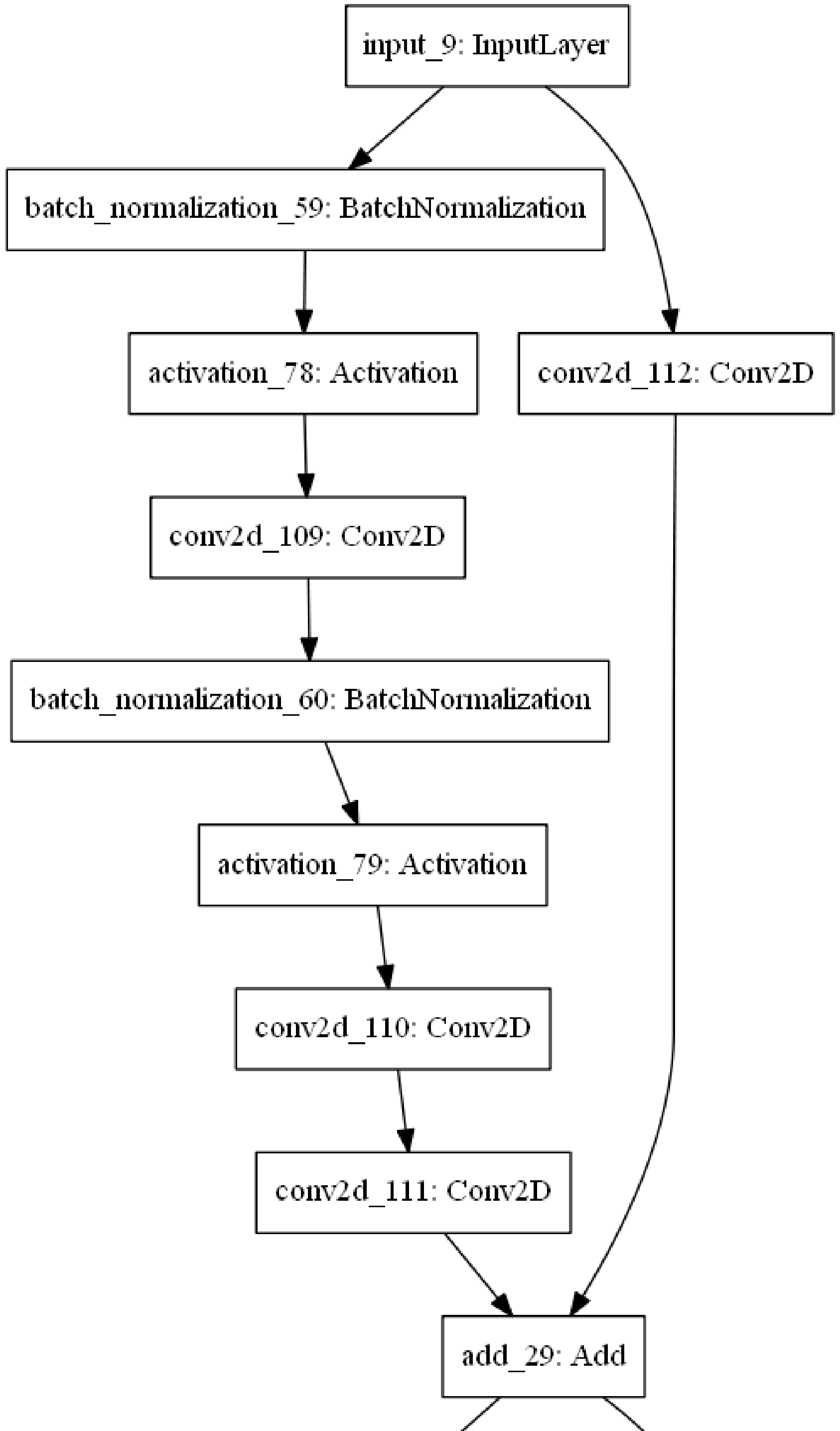
Implementation

The residual blocks are based on the improved scheme proposed in “Identity Mappings in Deep Residual Networks” by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun.



(a) original (b) proposed





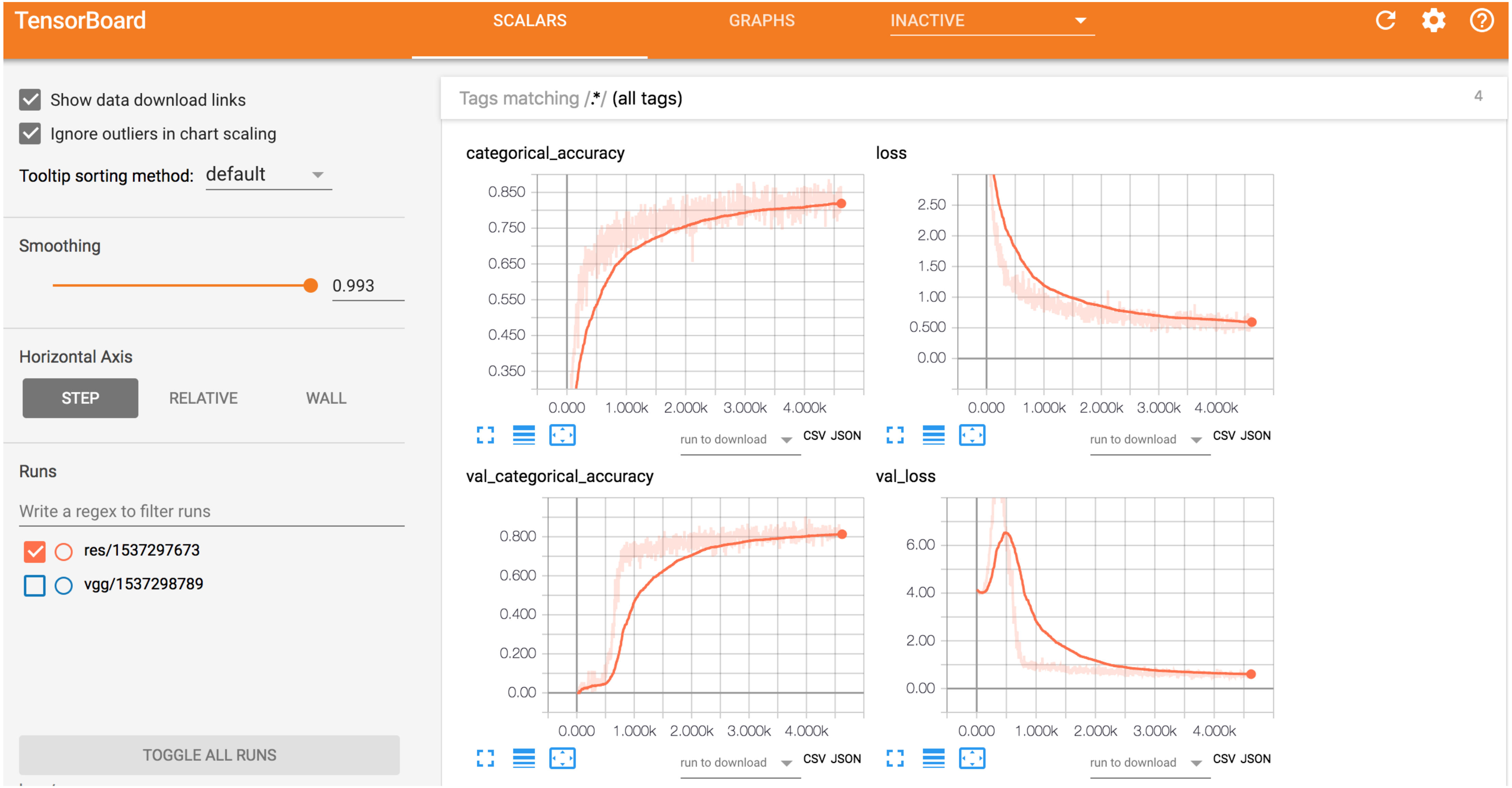
Architecture

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

```
# Resuidal block BN -> relu -> conv -> bn -> relu -> conv
def res_block(x, filters):
    bn1 = BatchNormalization()(x)
    act1 = Activation('relu')(bn1)
    conv1 = Conv2D(filters=filters, kernel_size=(3, 3), data_format='channels_first', strides=(2, 2), padding='same',
                   kernel_initializer=glorot_uniform(seed=0))(act1)

    bn2 = BatchNormalization()(conv1)
    act2 = Activation('relu')(bn2)
    conv2 = Conv2D(filters=filters, kernel_size=(3, 3), data_format='channels_first', strides=(1, 1), padding='same',
                   kernel_initializer=glorot_uniform(seed=0))(act2)

    residual = Conv2D(1, (1, 1), strides=(1, 1), data_format='channels_first')(conv2)


    x = Conv2D(filters=filters, kernel_size=(3, 3), data_format='channels_first', strides=(2, 2), padding='same',
                   kernel_initializer=glorot_uniform(seed=0))(x)

    # Combing shortcut and resuidal block
    out = Add()([x, residual])

    return out
```

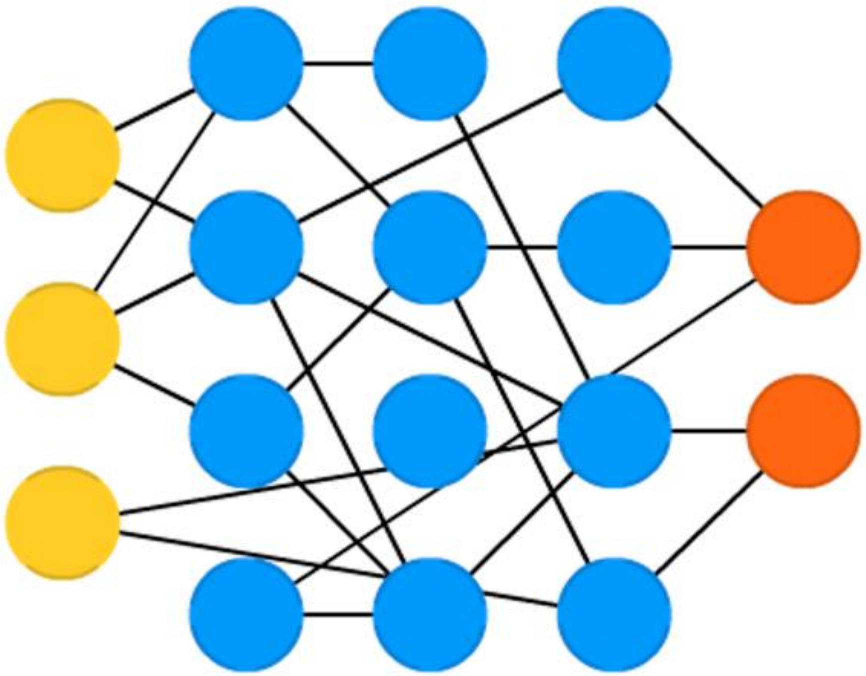
```
# Combining resuidal blocks into a network
res1 = res_block(input1, 64)
res2 = res_block(res1, 128)
res3 = res_block(res2, 256)
res4 = res_block(res3, 512)

# Classifier block
act1 = Activation('relu')(res4)
flatten1 = Flatten()(act1)
dense1 = Dense(512)(flatten1)
act2 = Activation('relu')(dense1)
dense2 = Dense(62)(act2)
output1 = Activation('softmax')(dense2)

model = Model(inputs=input1, outputs=output1)

# Compiling the model
model.compile(loss='categorical_crossentropy',
              optimizer=Adadelta(lr=0.1),
              metrics=['categorical_accuracy'])

model.summary()
```

Results



TensorBoard
SCALARS
GRAPHS
INACTIVE
Show data download links
Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing
0.993
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
res/1537297673
vgg/1537298789
TOGGLE ALL RUNS
Tags matching /.*/ (all tags)
4
categorical_accuracy
0.850
0.750
0.650
0.550
0.450
0.350
0.000
1.000k
2.000k
3.000k
4.000k
run to download
CSV JSON
loss
2.50
2.00
1.50
1.00
0.500
0.00
0.000
1.000k
2.000k
3.000k
4.000k
run to download
CSV JSON
val_categorical_accuracy
0.800
0.600
0.400
0.200
0.00
0.000
1.000k
2.000k
3.000k
4.000k
run to download
CSV JSON
val_loss
6.00
4.00
2.00
0.00
0.000
1.000k
2.000k
3.000k
4.000k
run to download
CSV JSON

What’s next?



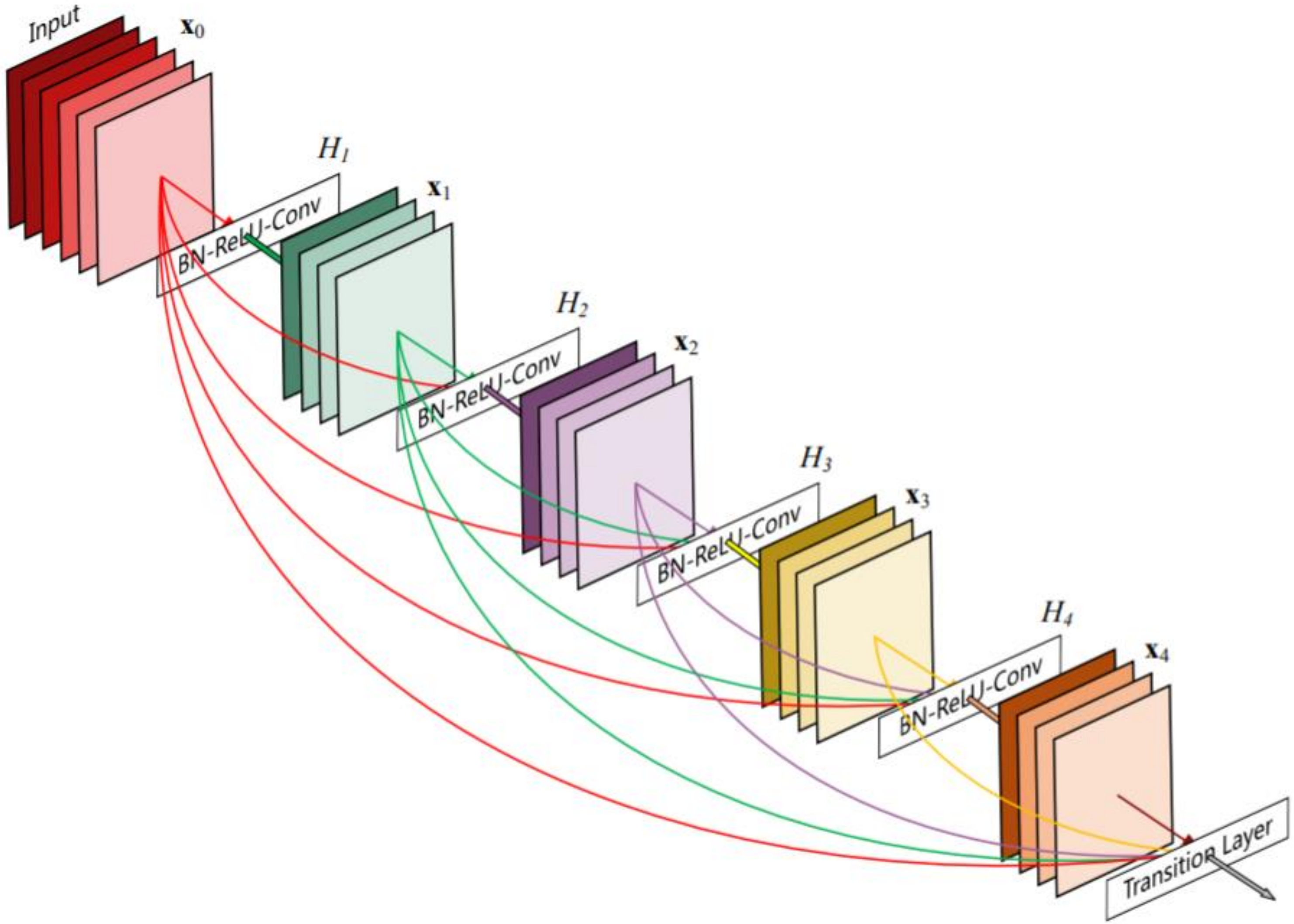


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

“Densely Connected Convolutional Networks” by Gao Huang, Zhuang Liu, Laurens van der Maaten


THANK YOU