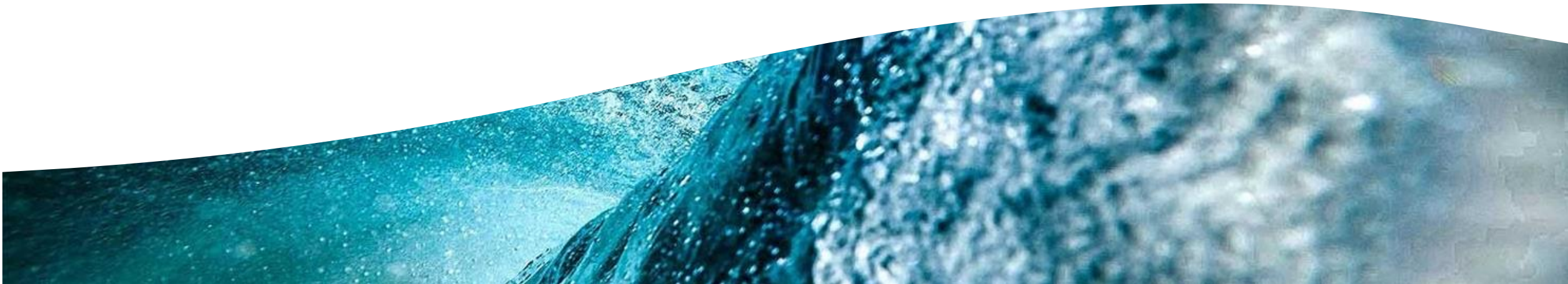


ResNet Overview*

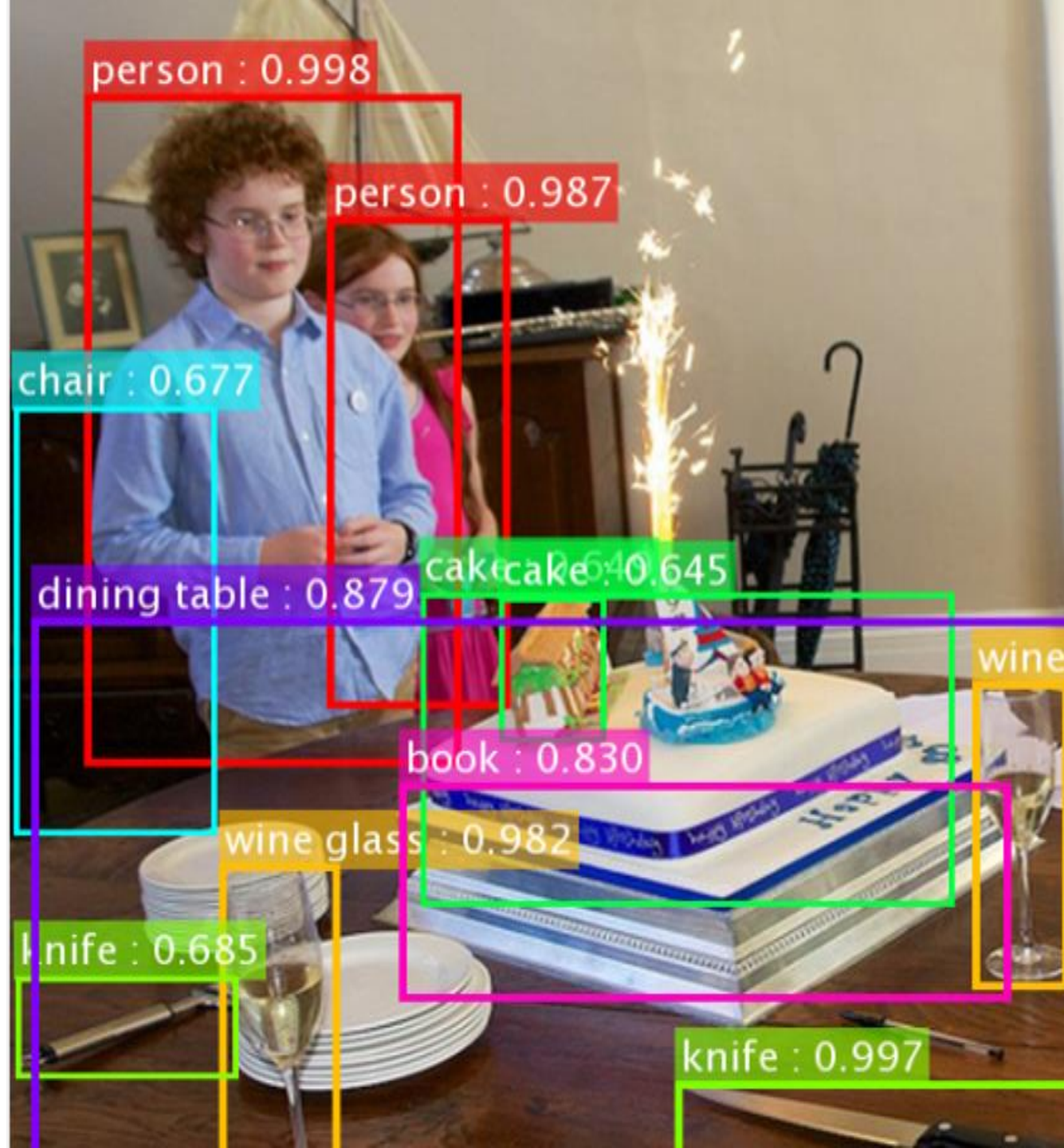
Implementation using Keras



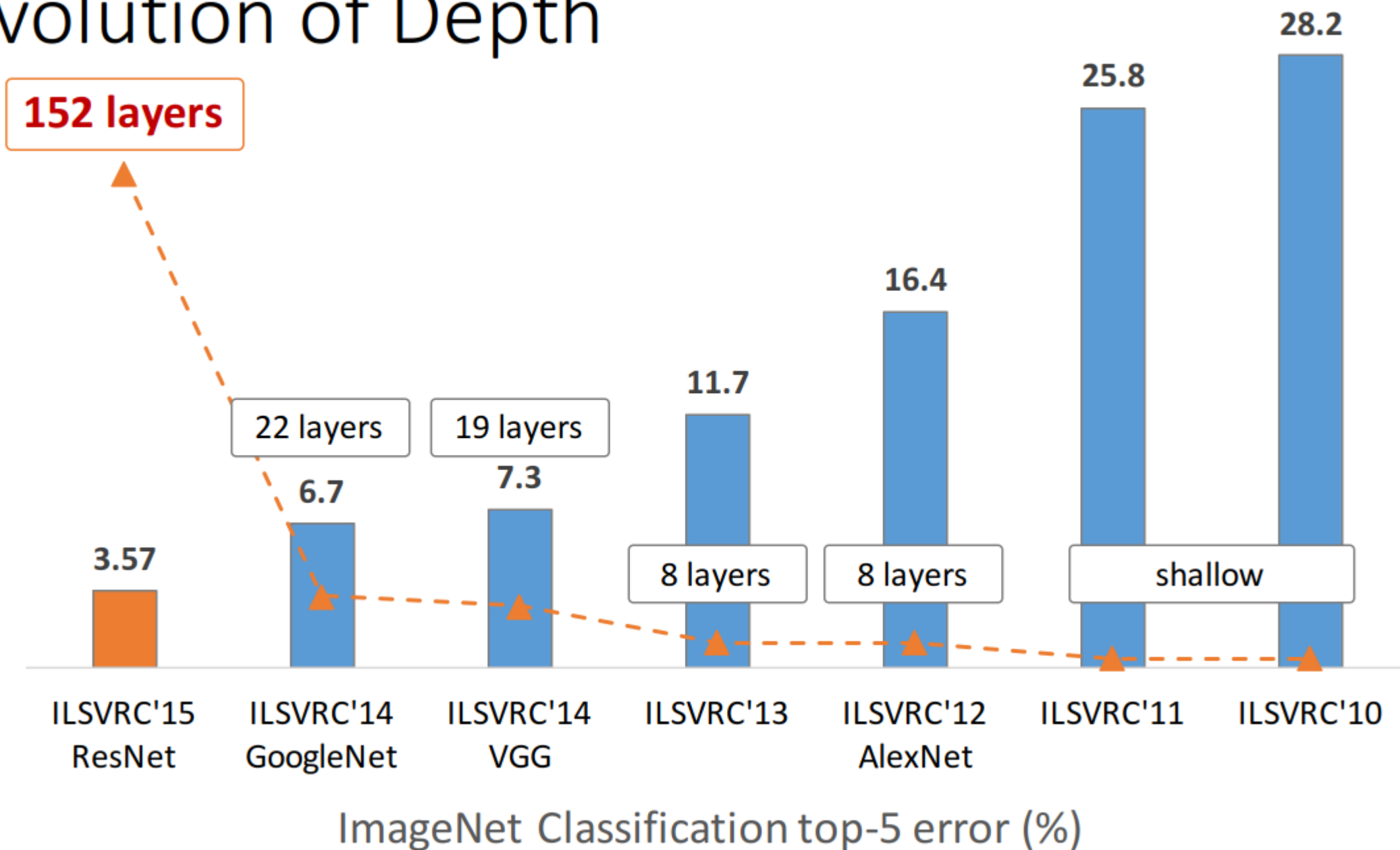
* Slides are modified from the original paper Deep Residual Learning for Image Recognition by Kaiming He, Xiangyu Zhang , Shaoqing Ren and Jian Sun

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.

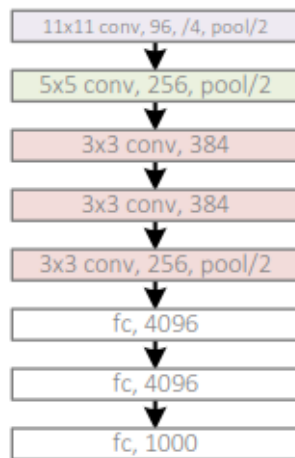


Revolution of Depth

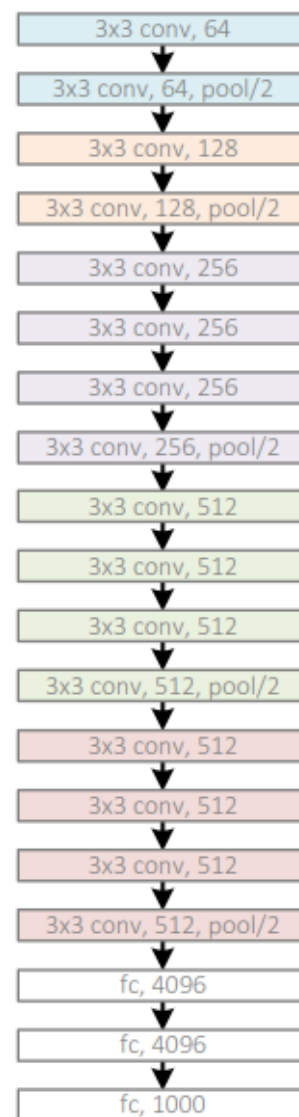


Revolution of Depth

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

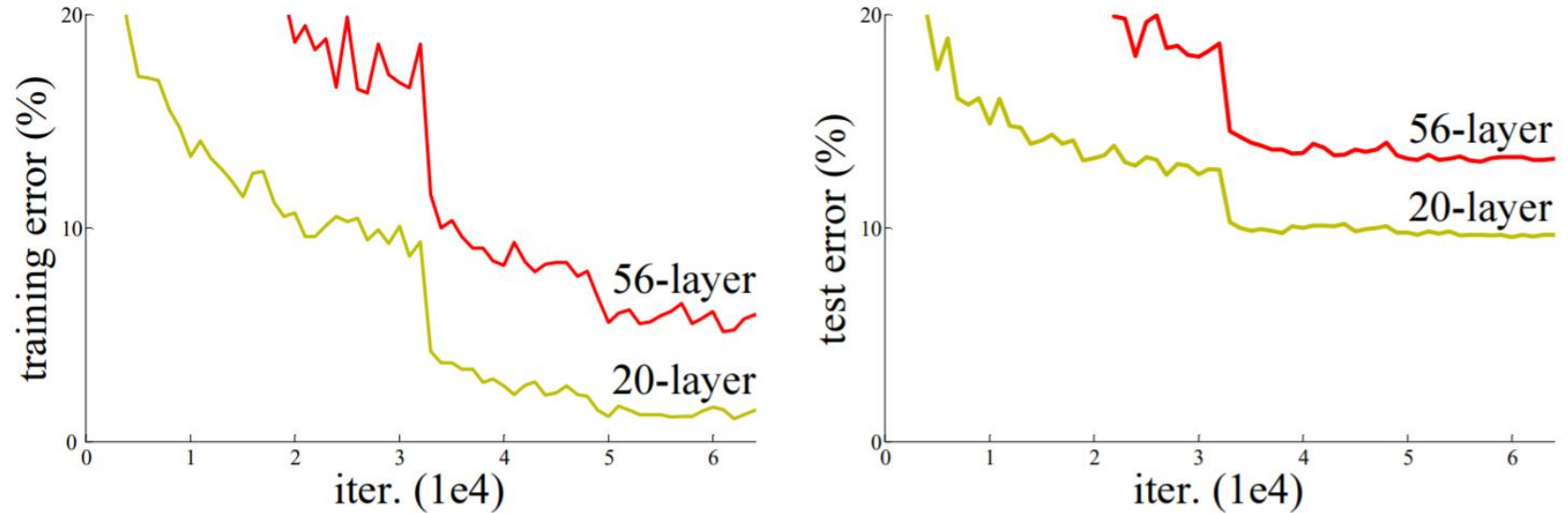
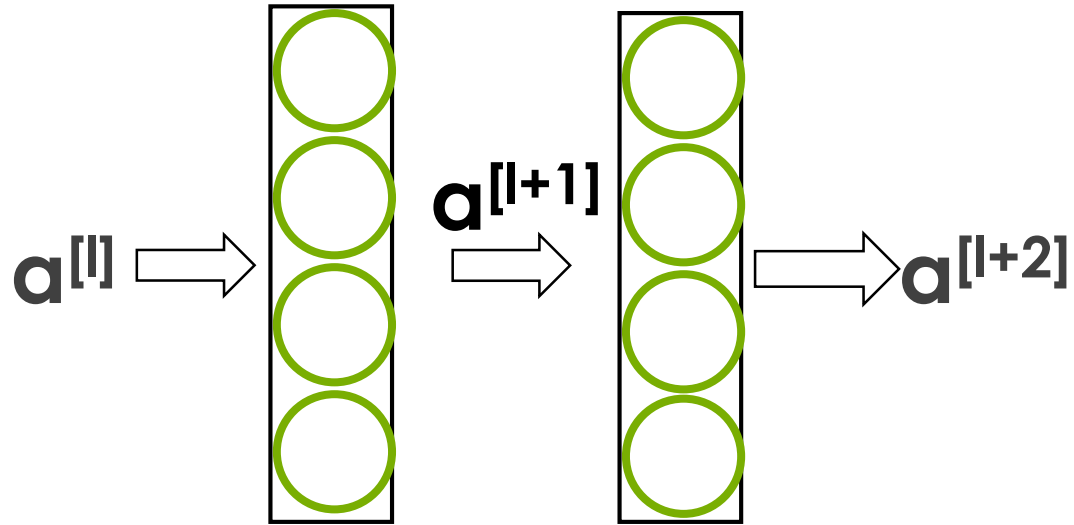


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer “plain” networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Sequential Block



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

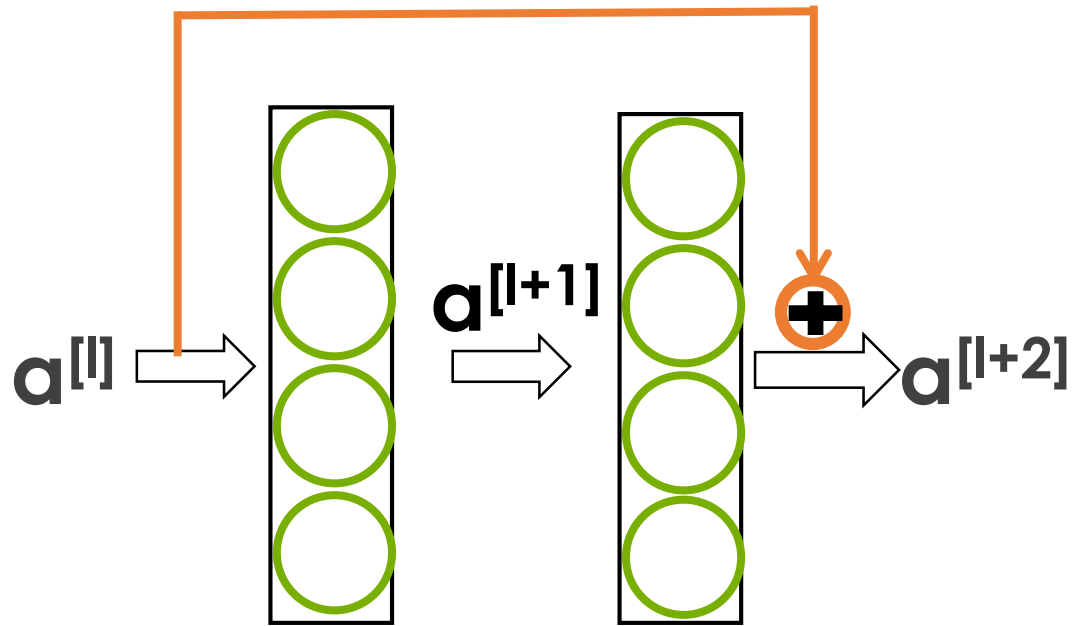
$$\mathbf{a}^{[l+1]} = g(\mathbf{z}^{[l+1]})$$

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

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

$\mathbf{a}^{[l]} \rightarrow$ Linear \rightarrow ReLu \rightarrow Linear \rightarrow ReLu

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



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

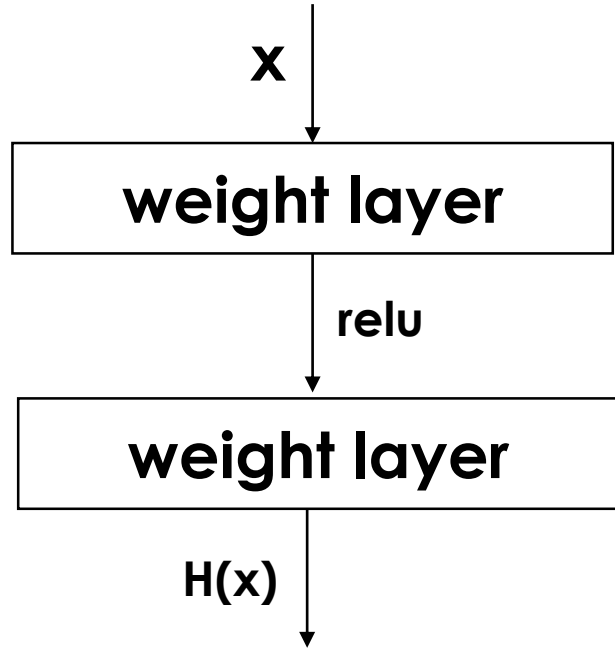
$$a^{[l+1]} = g(z^{[l+1]})$$

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

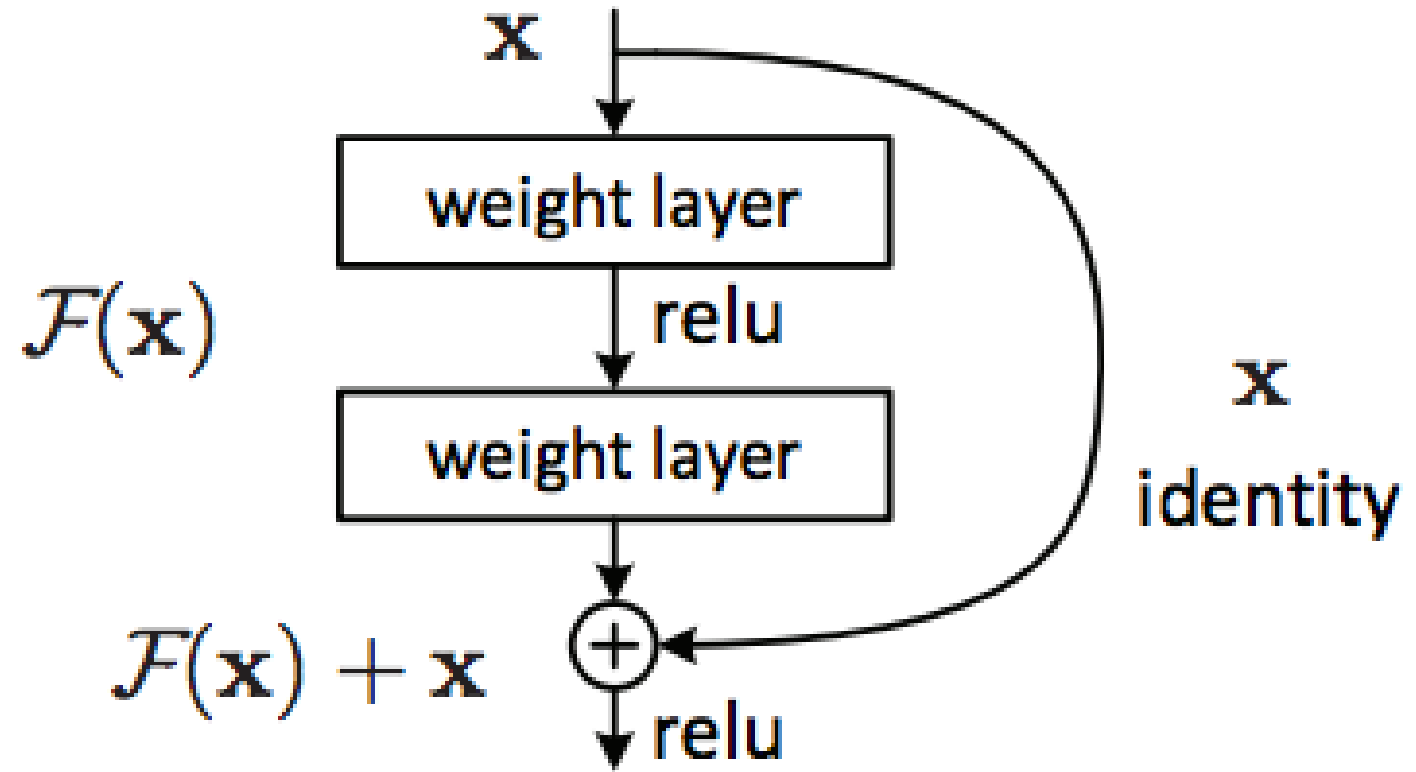
$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$

$a^{[l]} \rightarrow$ Linear \rightarrow ReLu \rightarrow Linear \rightarrow ReLu

Building block

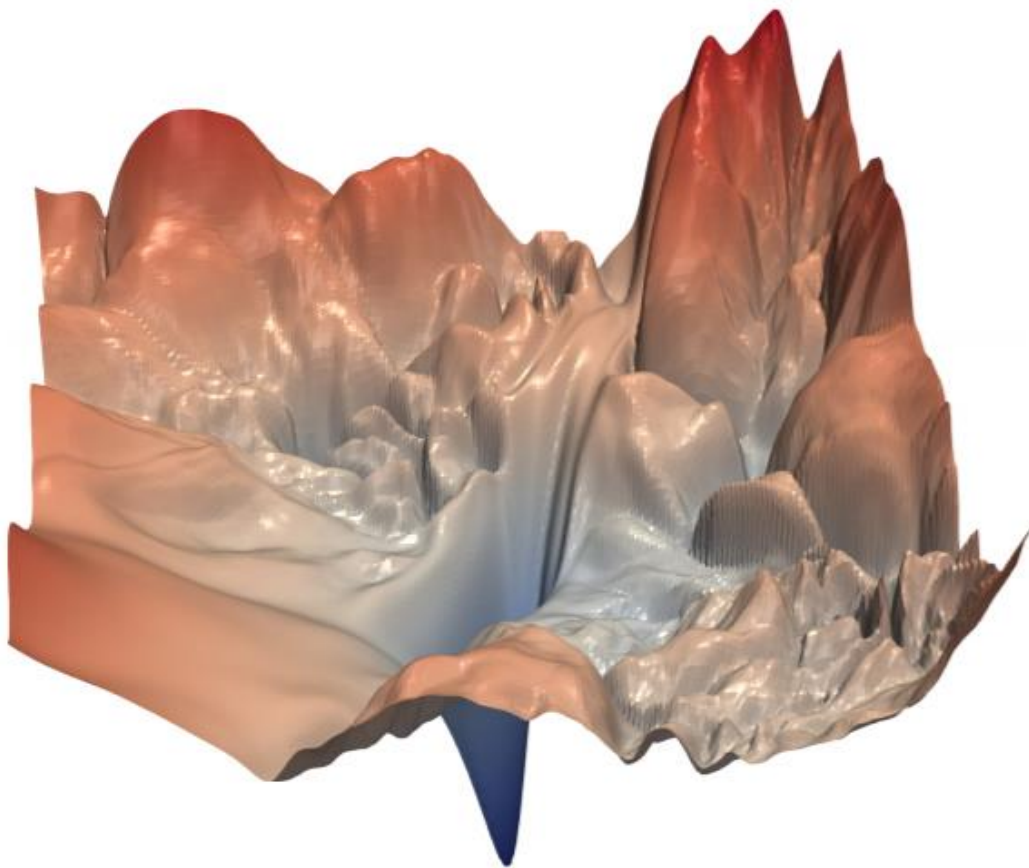


Plain block

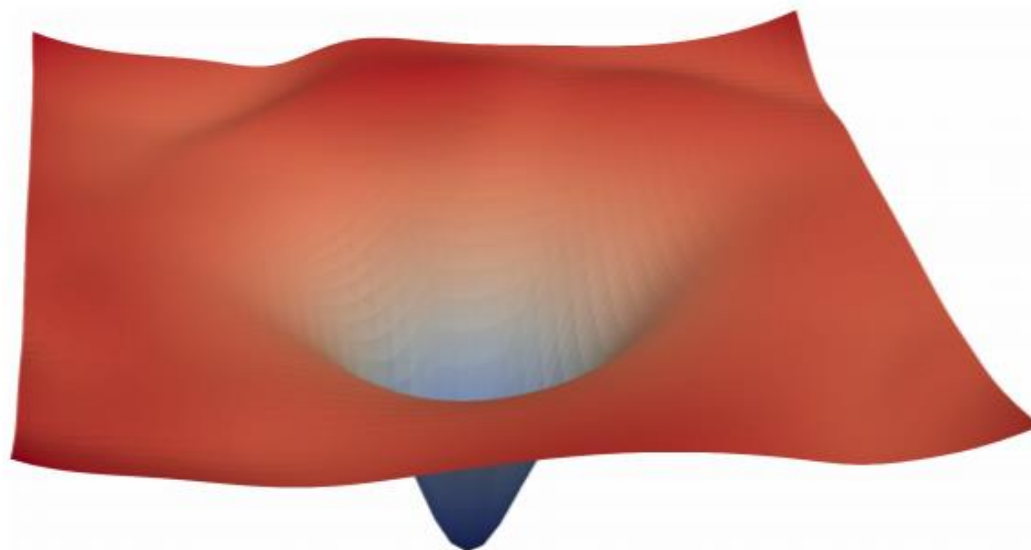


Residual block

Minimizing high-dimensional non-convex loss function



(a) without skip connections

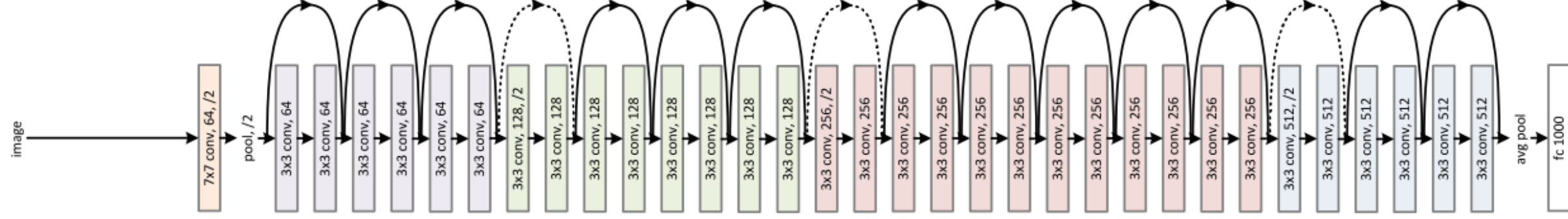


(b) with skip connections

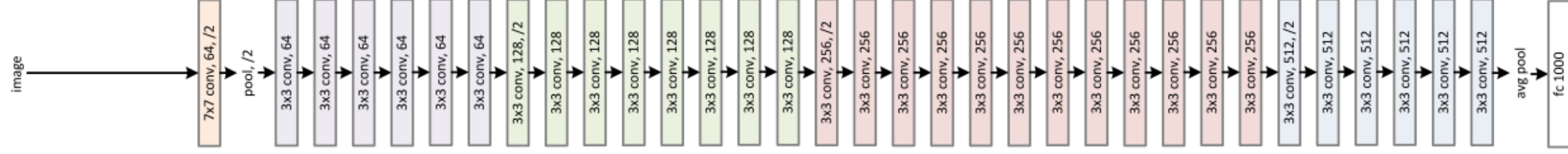
Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The vertical axis is logarithmic to show dynamic range. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.

ResNet Structure

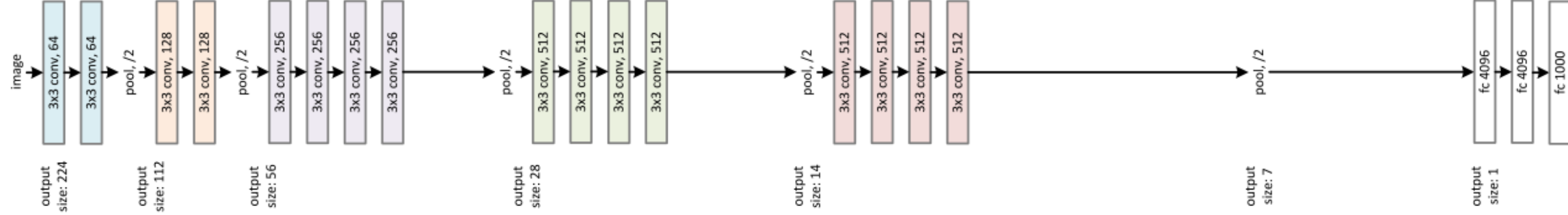
34-layer residual



34-layer plain

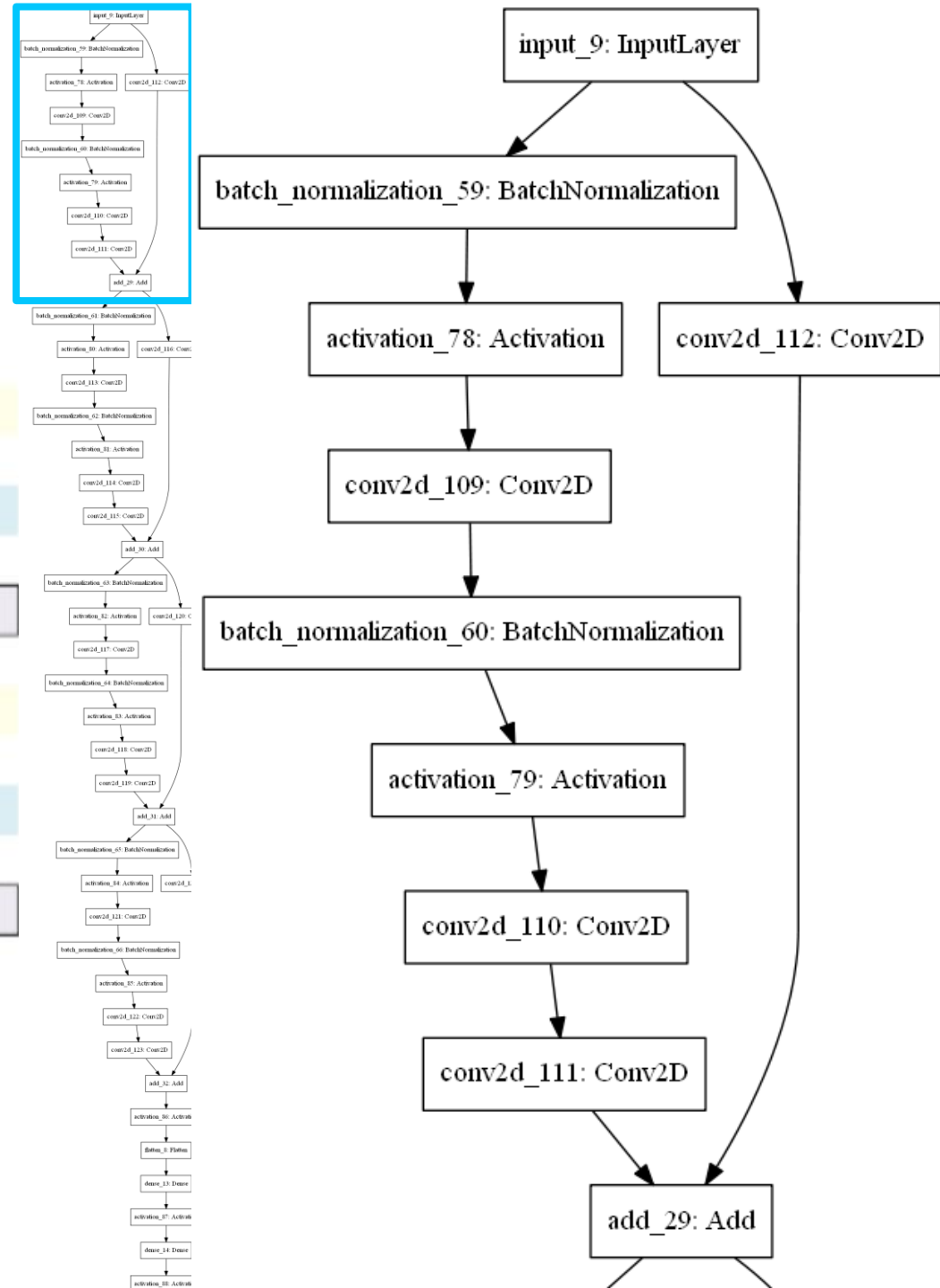
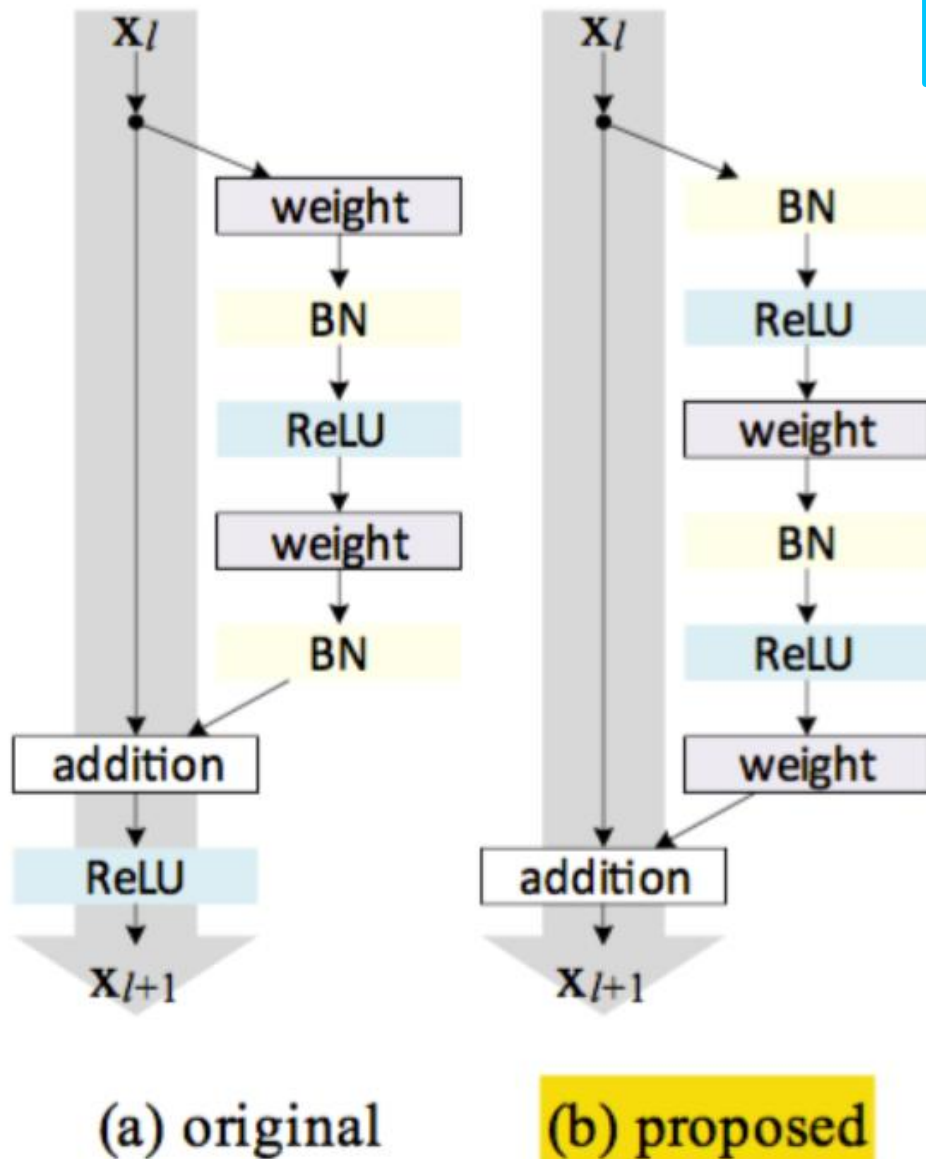


VGG-19



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.



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				
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	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


```

7 # Residual block BN -> relu -> conv -> bn -> relu -> conv
8 def res_block(x, filters):
9     bn1 = BatchNormalization()(x)
10    act1 = Activation('relu')(bn1)
11    conv1 = Conv2D(filters=filters, kernel_size=(3, 3), data_format='channels_first', strides=(2, 2), padding='same',
12                  kernel_initializer=glorot_uniform(seed=0))(act1)
13
14    bn2 = BatchNormalization()(conv1)
15    act2 = Activation('relu')(bn2)
16    conv2 = Conv2D(filters=filters, kernel_size=(3, 3), data_format='channels_first', strides=(1, 1), padding='same',
17                  kernel_initializer=glorot_uniform(seed=0))(act2)
18
19    residual = Conv2D(1, (1, 1), strides=(1, 1), data_format='channels_first')(conv2)
20
21
22    x = Conv2D(filters=filters, kernel_size=(3, 3), data_format='channels_first', strides=(2, 2), padding='same',
23              kernel_initializer=glorot_uniform(seed=0))(x)
24
25    # Combing shortcut and residual block
26    out = Add()([x, residual])
27
28    return out

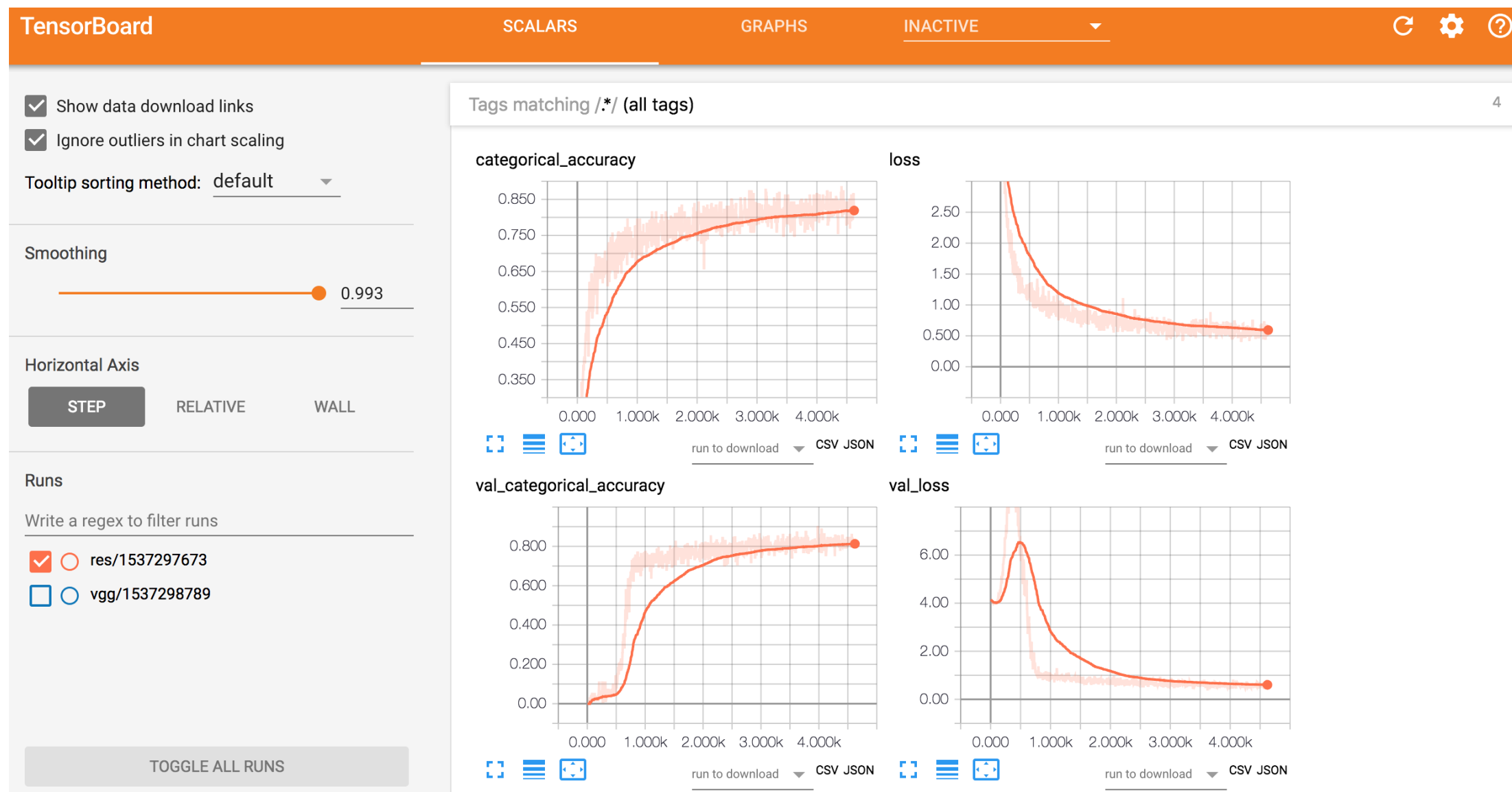
```

```

30 # Combining residual blocks into a network
31 res1 = res_block(input1, 64)
32 res2 = res_block(res1, 128)
33 res3 = res_block(res2, 256)
34 res4 = res_block(res3, 512)
35
36 # Classifier block
37 act1 = Activation('relu')(res4)
38 flatten1 = Flatten()(act1)
39 dense1 = Dense(512)(flatten1)
40 act2 = Activation('relu')(dense1)
41 dense2 = Dense(62)(act2)
42 output1 = Activation('softmax')(dense2)
43
44 model = Model(inputs=input1, outputs=output1)
45
46 # Compiling the model
47 model.compile(loss='categorical_crossentropy',
48               optimizer=Adadelta(lr=0.1),
49               metrics=['categorical_accuracy'])
50
51 model.summary()
52

```

Results



What's next?

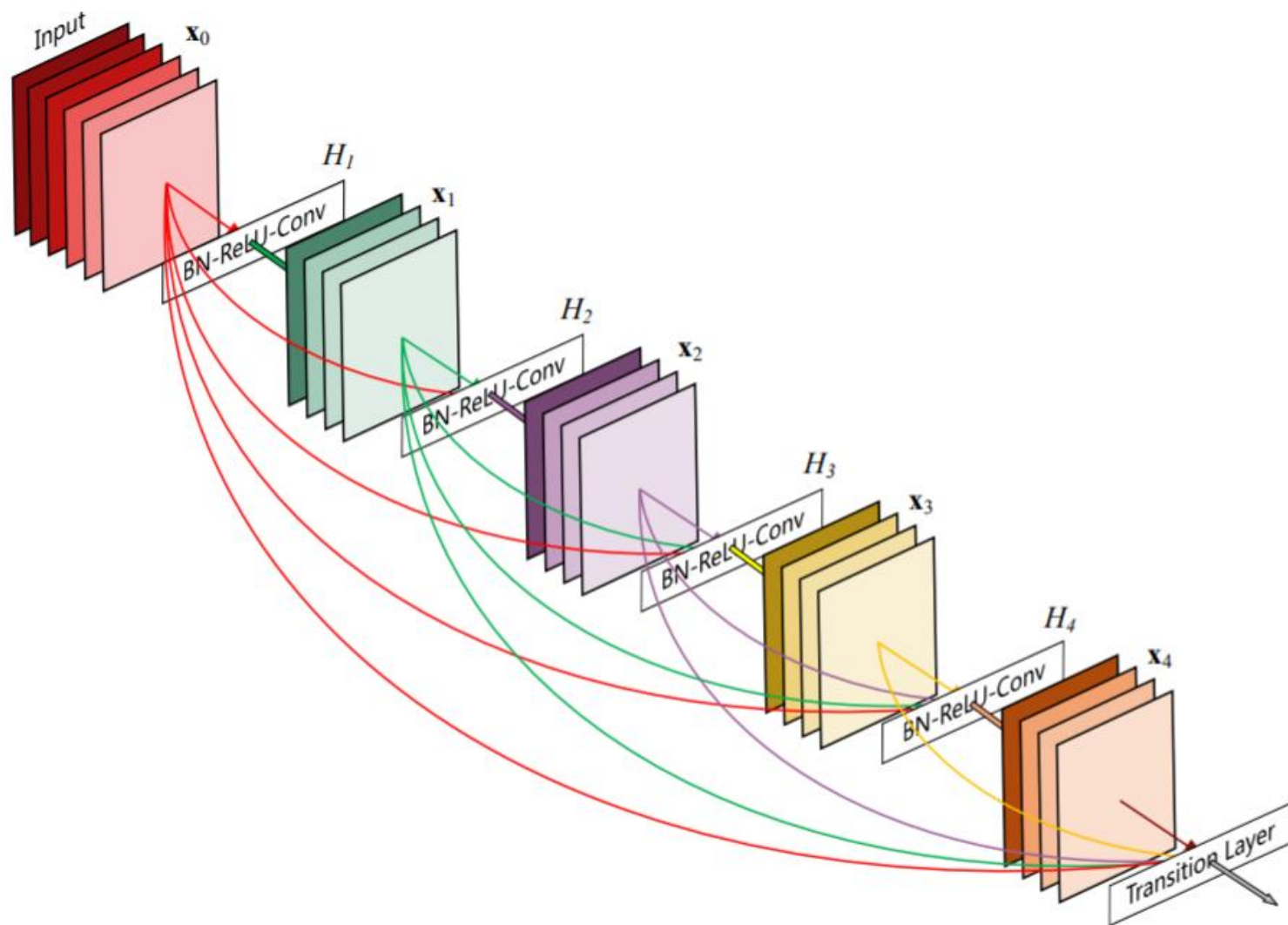
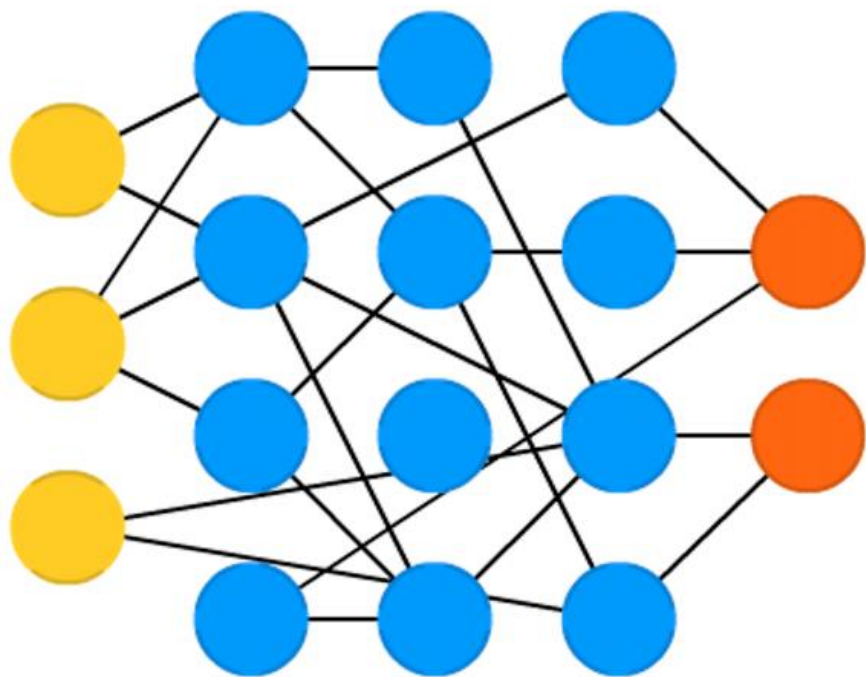


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

References

- Li, Hao, Zheng Xu, Gavin Taylor, and Tom Goldstein. "Visualizing the loss landscape of neural nets." *arXiv preprint arXiv:1712.09913* (2017).
- Huang, Gao, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. "Densely Connected Convolutional Networks." In *CVPR*, vol. 1, no. 2, p. 3. 2017.
- He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Identity mappings in deep residual networks." In *European conference on computer vision*, pp. 630-645. Springer, Cham, 2016.
- He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778. 2016.
- Andrew Ng "Deep Learning" course at Coursera
- Article @ Medium "Understanding and Implementing Architectures of ResNet and ResNeXt for state-of-the-art Image Classification: From Microsoft to Facebook"