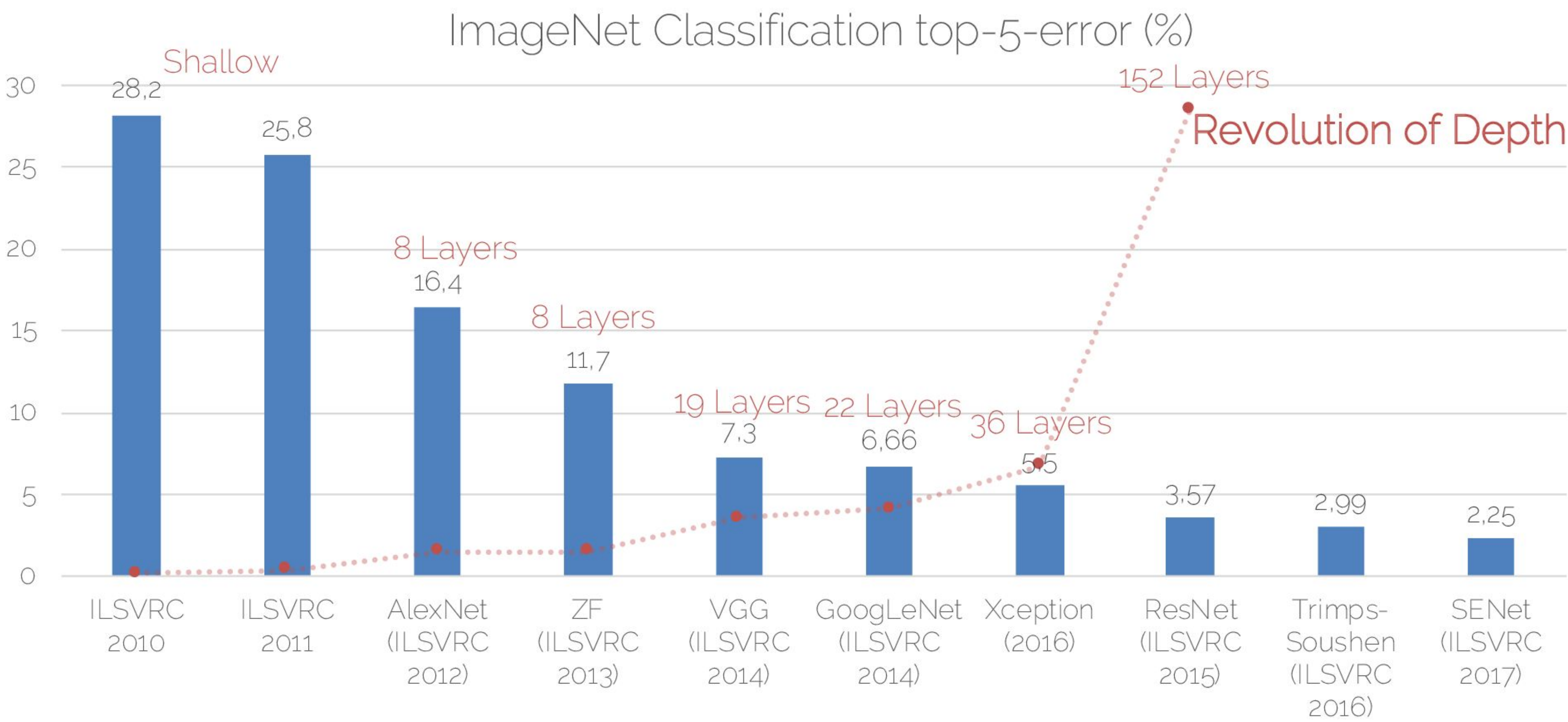
Deep learning for Computer Vision

ImageNet Benchmark (Recap)



Common Performance Metrics

- **Top-1 score**: check if a sample's top class (i.e. the one with highest probability) is the same as its target label

Common Performance Metrics

- **Top-1 score**: check if a sample's top class (i.e. the one with highest probability) is the same as its target label
- **Top-5 score**: check if your label is in your 5 first predictions (i.e. predictions with 5 highest probabilities)

Common Performance Metrics

- **Top-1 score**: check if a sample's top class (i.e. the one with highest probability) is the same as its target label
- **Top-5 score**: check if your label is in your 5 first predictions (i.e. predictions with 5 highest probabilities)
- **Top-5 error:** percentage of test samples for which the correct class was not in the top 5 predicted classes

Classical Architecture (Recap)

Architecture	Year	Layers	Key Innovations	Parameters	Accuracy (ImageNet)	Researchers
AlexNet	2012	8	ReLU, LRN	62 million	57.2%	Alex Krizhevsky et al.
VGGNet	2014	16-19	3x3 convolution filters, Deep architecture	138-144 million	74.4%	Karen Simonyan and Andrew Zisserman
Inception Net	2014	22-42	Inception modules, Auxiliary classifiers	4-12 million	74.8%	Szegedy et al.
Next?	2015	**50-152**			75.3%	He et al.

Problem of Depth

Going Deeper

- There has been a general trend in recent years to design deeper networks.
- Deeper network are known to produce more complex features and tend to generalise better.

Going Deeper

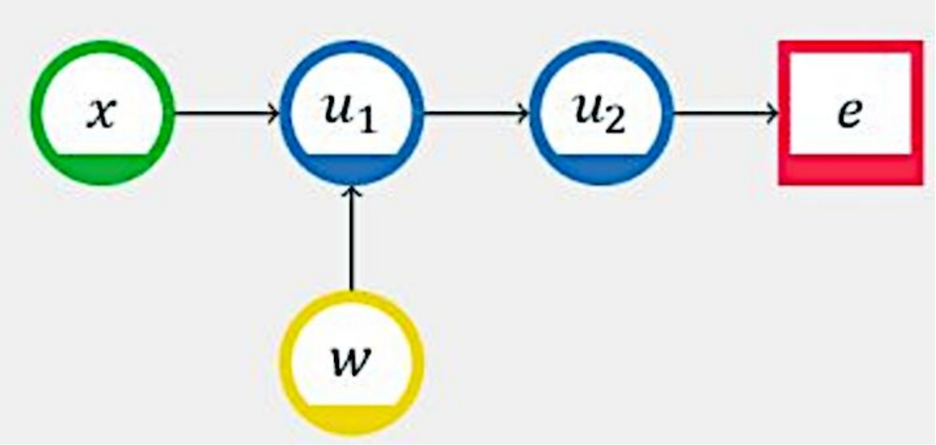
- There has been a general trend in recent years to design deeper networks.
- Deeper network are known to produce more complex features and tend to generalise better.
- Training deep networks is however difficult.
    - Problem of vanishing gradients
    - Problem of exploding gradient

Vanishing Gradient Problem

- Gradient of the loss function with respect to the weights in the lower layers becomes **very small** during backpropagation.

Vanishing Gradient Problem:

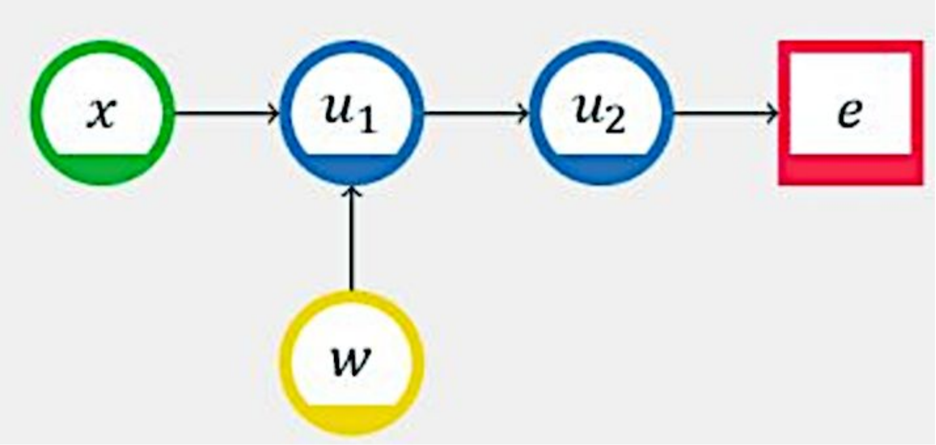
Vanishing gradients problem on this simple network:



$$\frac{\partial e}{\partial w} = \frac{\partial e}{\partial u_2}\frac{\partial u_2}{\partial u_1}\frac{\partial u_1}{\partial w} = errorDelta$$

Vanishing Gradient Problem:

Weight update issue

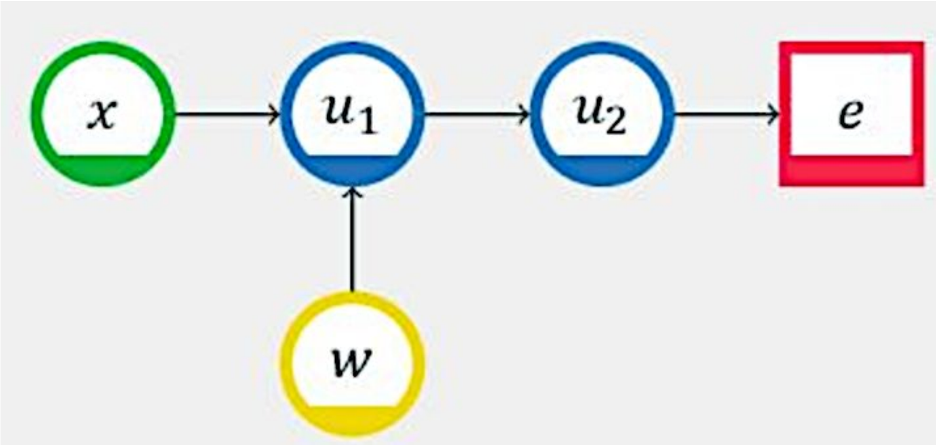


$$\frac{\partial e}{\partial w} = \frac{\partial e}{\partial u_2}\frac{\partial u_2}{\partial u_1}\frac{\partial u_1}{\partial w}$$

$$w[j] \leftarrow w[j] + \eta \times errorDelta$$

Vanishing Gradient Problem:

Vanishing gradients problem on this simple network:



$$\frac{\partial e}{\partial w} = \frac{\partial e}{\partial u_2}\frac{\partial u_2}{\partial u_1}\frac{\partial u_1}{\partial w} = errorDelta$$

During the gradient descent, we evaluate $\frac{\partial e}{\partial w}$ which is a product of the intermediate derivatives. If any of these is close to zero, then $\frac{\partial e}{\partial w} \approx 0$.

Vanishing Gradient Problem:

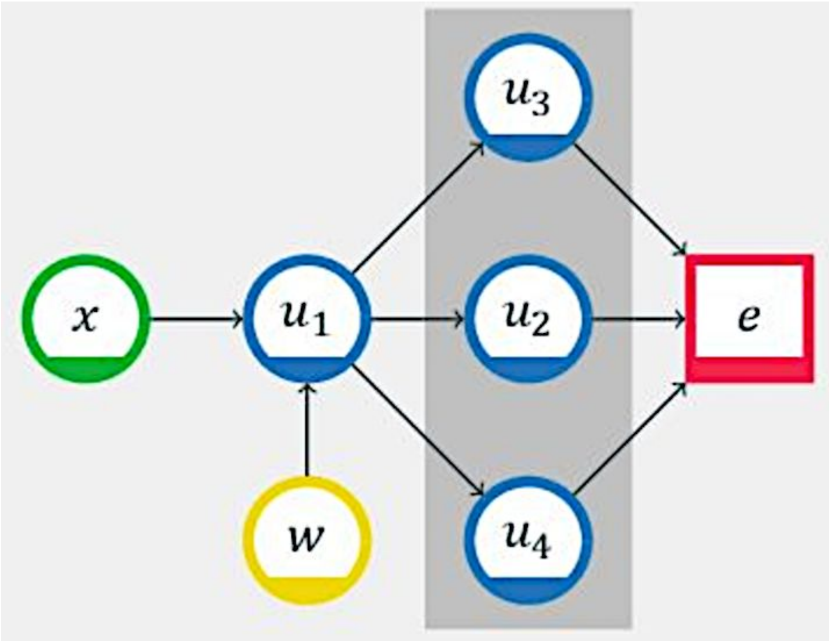
- The small gradient is propagated back through the layers, making it difficult for lower layers to learn meaningful representations of the data.

Vanishing Gradient Problem:

- The small gradient is propagated back through the layers, making it difficult for lower layers to learn meaningful representations of the data.
- Very challenging to train CNNs, where the gradient can become exponentially small.

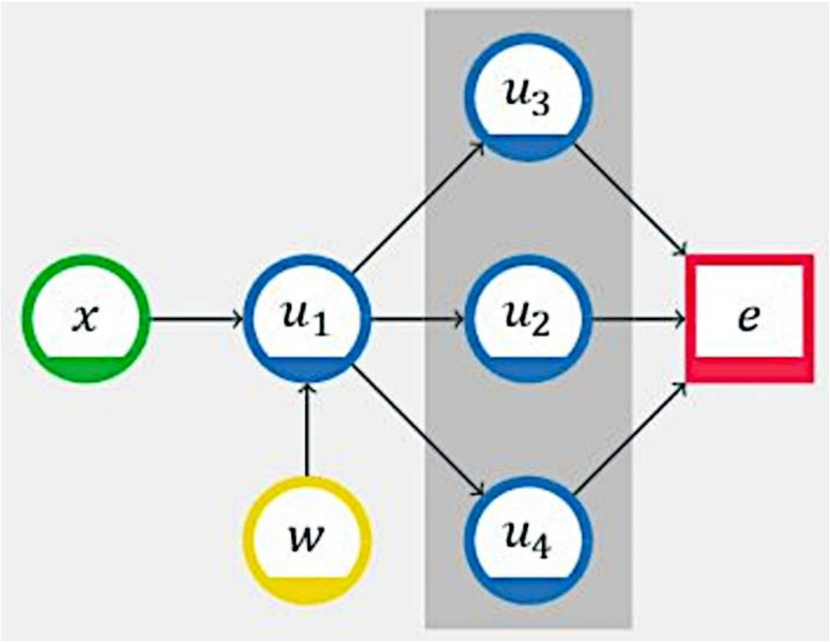
Going Deeper

Now consider the problem of vanishing gradients on this new network:



Going Deeper

Now consider the problem of vanishing gradients on this new network:



$$\frac{\partial e}{\partial w} = \frac{\partial e}{\partial u_2}\frac{\partial u_2}{\partial u_1}\frac{\partial u_1}{\partial w} + \frac{\partial e}{\partial u_4}\frac{\partial u_4}{\partial u_1}\frac{\partial u_1}{\partial w} + \frac{\partial e}{\partial u_3}\frac{\partial u_3}{\partial u_1}\frac{\partial u_1}{\partial w}$$

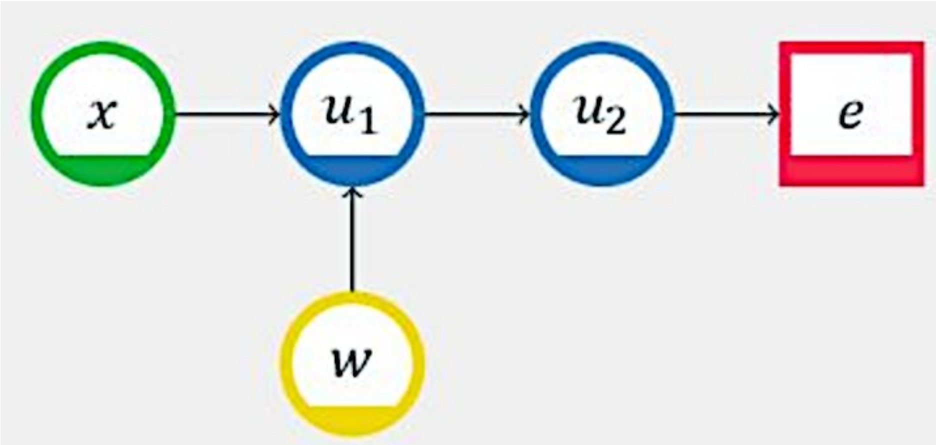
It is now less likely for $\frac{\partial e}{\partial w}$ to be null as all three terms need to be null.

Exploding Gradient Problem:

- Gradient of the loss function with respect to the weights in the lower layers becomes **very large** during backpropagation.

Exploding Gradient Problem:

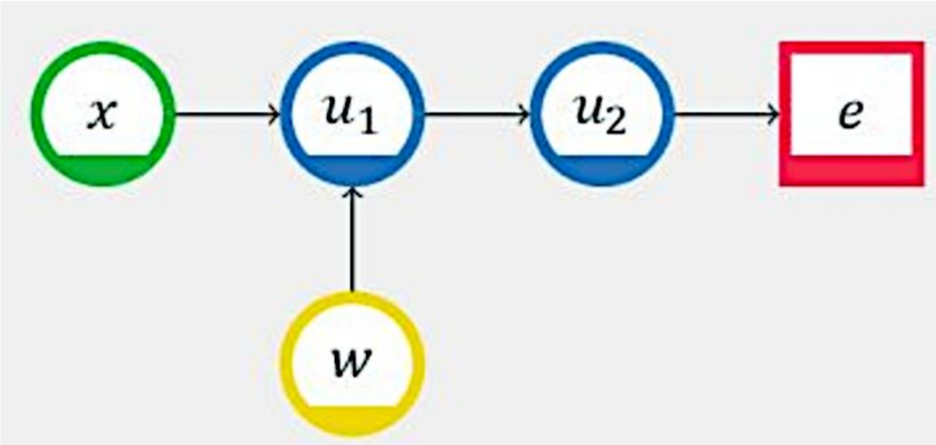
Exploding Gradient Problem on this simple network:



$$\frac{\partial e}{\partial w} = \frac{\partial e}{\partial u_2}\frac{\partial u_2}{\partial u_1}\frac{\partial u_1}{\partial w} = errorDelta$$

Exploding Gradient Problem:

Weight update issue

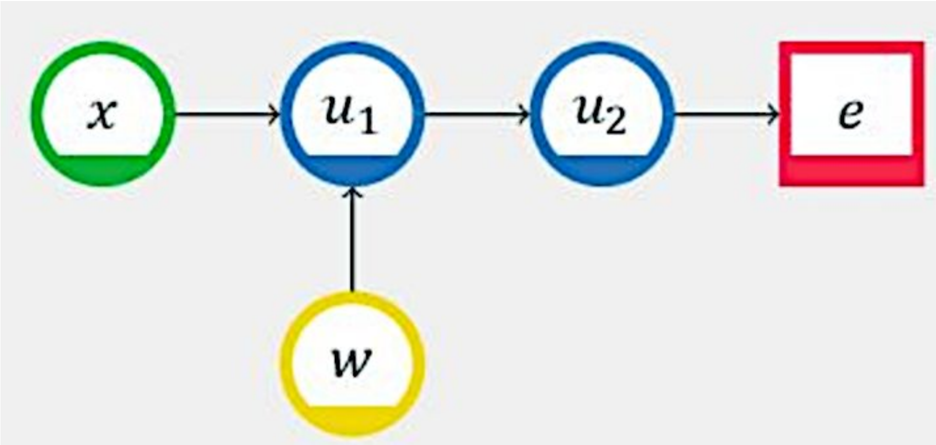


$$\frac{\partial e}{\partial w} = \frac{\partial e}{\partial u_2}\frac{\partial u_2}{\partial u_1}\frac{\partial u_1}{\partial w}$$

$$w[j] \leftarrow w[j] + \eta \times errorDelta$$

Exploding Gradient Problem:

Exploding Gradient Problem on this simple network:



$$\frac{\partial e}{\partial w} = \frac{\partial e}{\partial u_2}\frac{\partial u_2}{\partial u_1}\frac{\partial u_1}{\partial w} = errorDelta$$

During the gradient descent, we evaluate $\frac{\partial e}{\partial w}$ which is a product of the intermediate derivatives. If these are high value then $\frac{\partial e}{\partial w} \approx \infty$

Exploding Gradient Problem:

- The large gradient is propagated back through the layers, causing weight updates that are too large.

Exploding Gradient Problem:

- The large gradient is propagated back through the layers, causing weight updates that are too large.
- Very challenging to train CNNs, where the gradient can become exponentially large.

Exploding Gradient Problem:

- The large gradient is propagated back through the layers, causing weight updates that are too large.
- Very challenging to train CNNs, where the gradient can become exponentially large.
- **Weight clipping** can be done for Exploding Gradient.

Problem of Depth

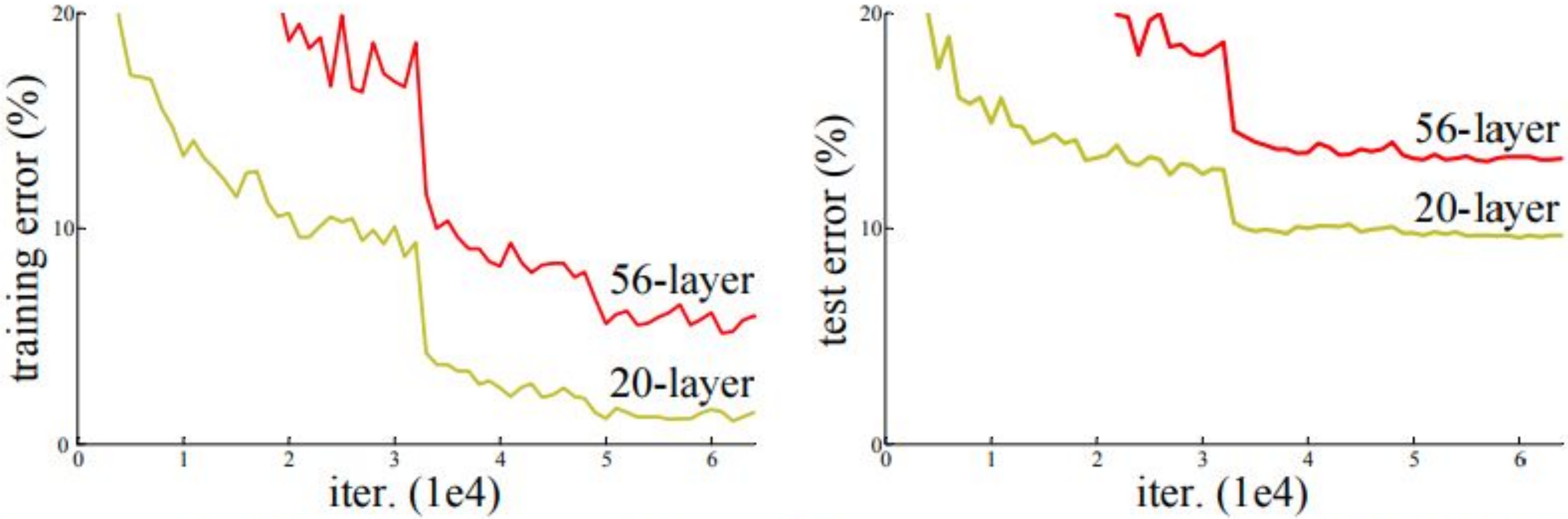


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error.

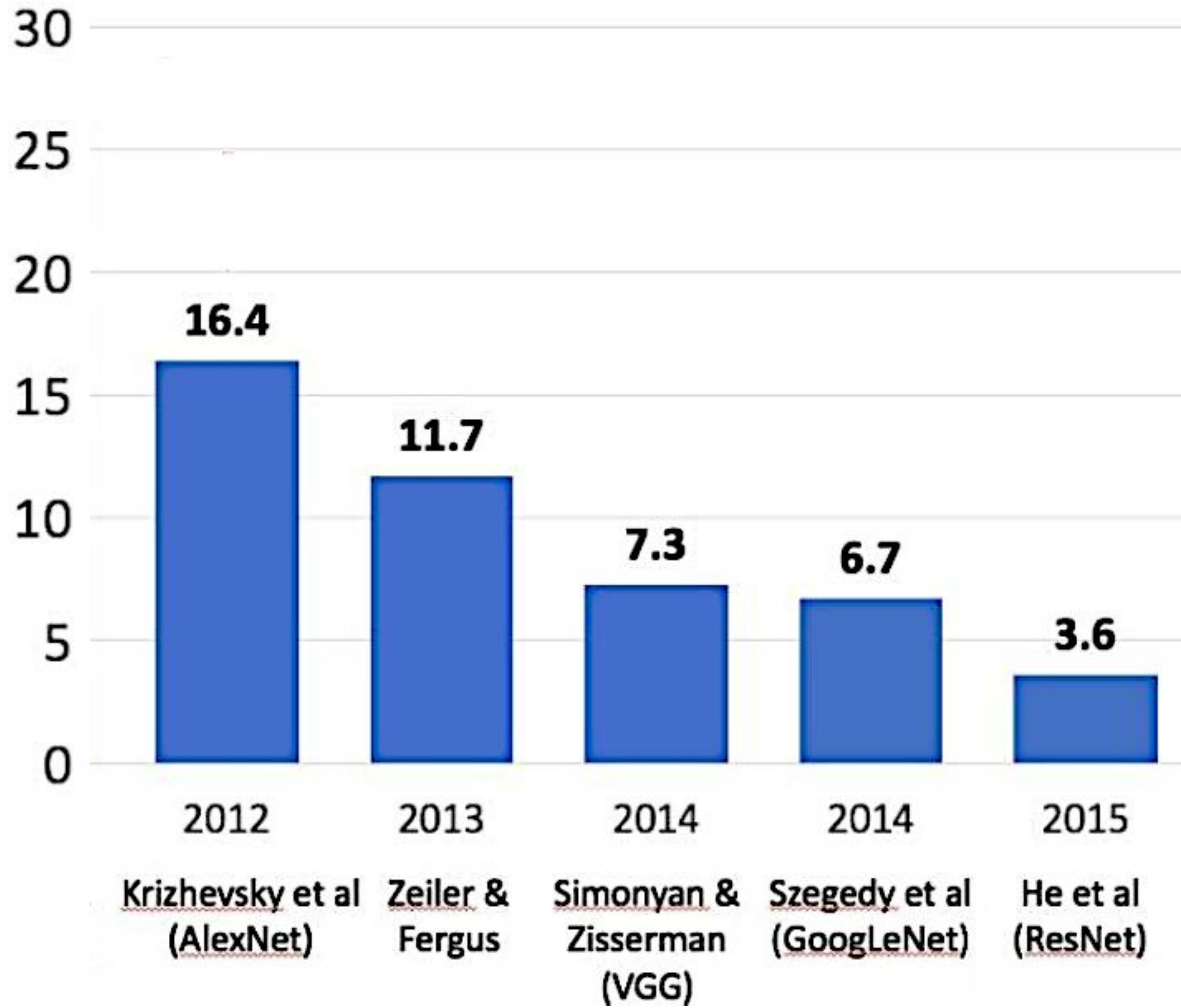
ResNet

Solution to Problem of Depth

Classical Architecture

Architecture	Year	Layers	Key Innovations	Parameters	Accuracy (ImageNet)	Researchers
AlexNet	2012	8	ReLU, LRN	62 million	57.2%	Alex Krizhevsky et al.
VGGNet	2014	16-19	3x3 convolution filters, Deep architecture	138-144 million	74.4%	Karen Simonyan and Andrew Zisserman
Inception Net	2014	22-42	Inception modules, Auxiliary classifiers	4-12 million	74.8%	Szegedy et al.
ResNet	2015	**50-152**	Residual connections, Shortcut connections	25.6-60 million	75.3%	He et al.

ResNet



30
25
20
15
10
5
0
16.4
11.7
7.3
6.7
3.6
2012
Krizhevsky et al (AlexNet)
2013
Zeiler & Fergus
2014
Simonyan & Zisserman (VGG)
2014
Szegedy et al (GoogLeNet)
2015
He et al (ResNet)

ResNet (Introduction)

- ResNet is a deep neural network architecture that was introduced by researchers at Microsoft in 2015.

ResNet (Introduction)

- ResNet is a deep neural network architecture that was introduced by researchers at Microsoft in 2015.
- The key innovation of ResNet is the use of residual connections, which allow for much deeper networks to be trained.

ResNet (Introduction)

- ResNet is a deep neural network architecture that was introduced by researchers at Microsoft in 2015.
- The key innovation of ResNet is the use of residual connections, which allow for much deeper networks to be trained.
- ResNet comes in several variants, including ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152.

ResNet (Introduction)

- ResNet is a deep neural network architecture that was introduced by researchers at Microsoft in 2015.
- The key innovation of ResNet is the use of residual connections, which allow for much deeper networks to be trained.
- ResNet comes in several variants, including ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152.
- ResNet has achieved good performance on many computer vision tasks, including classification, object detection, and segmentation.

Residual Block

Skip connection (key idea)

- ResNet is composed of a series of residual blocks, each of which contains one or more convolutional layers, batch normalization, and ReLU activation.

Skip connection (key idea)

- ResNet is composed of a series of residual blocks, each of which contains one or more convolutional layers, batch normalization, and ReLU activation.
- In residual blocks, there is a shortcut connection that bypasses one or more layers and allows the gradient to flow directly to earlier layers.

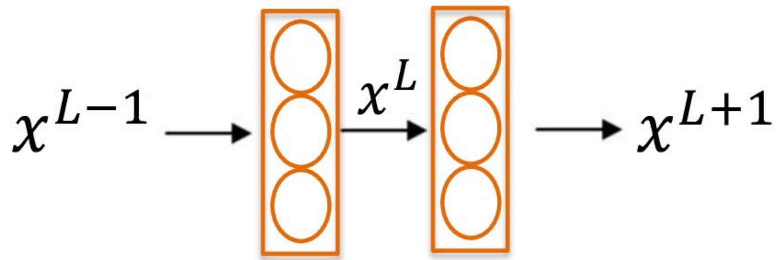
Skip connection (key idea)

- ResNet is composed of a series of residual blocks, each of which contains one or more convolutional layers, batch normalization, and ReLU activation.
- In residual blocks, there is a **shortcut connection** that bypasses one or more layers and allows the gradient to flow directly to earlier layers.
- This is shortcut connection known as a residual connection or **skip connection**.

Shortcut connection

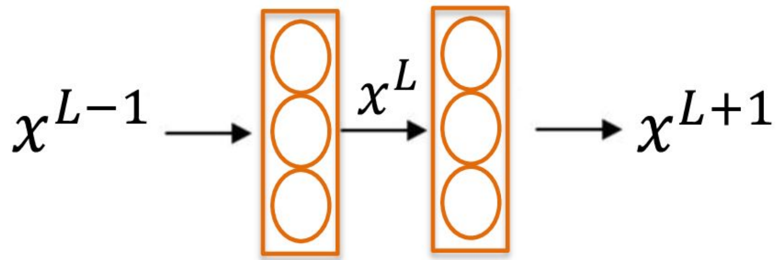
Skip connection.

- Two layers

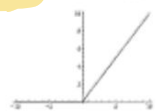


$x_L = f(w_L x_{L-1} + b_L)$

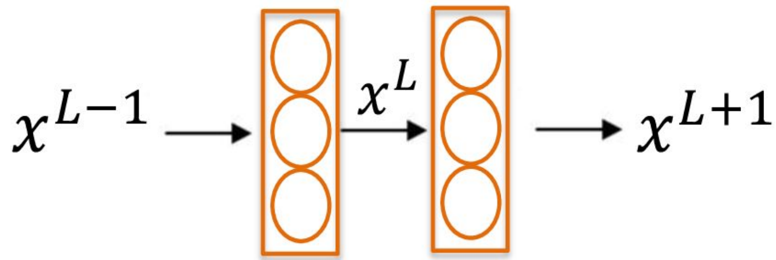
- Two layers



$$\text{Input} \longrightarrow \underbrace{W^L x^{L-1} + b^L}_{\text{Linear}} \longrightarrow x^L = \underbrace{f(W^L x^{L-1} + b^L)}_{\text{Non-linear}} \longrightarrow$$

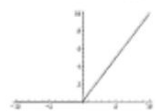


- Two layers



Input $\longrightarrow W^{L}x^{L-1} + b^{L} \longrightarrow x^{L} = f(W^{L}x^{L-1} + b^{L}) \longrightarrow$

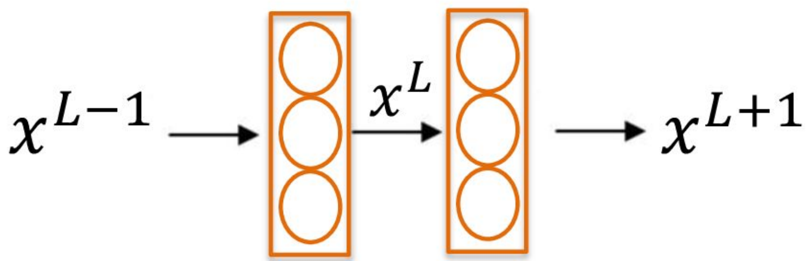
Linear

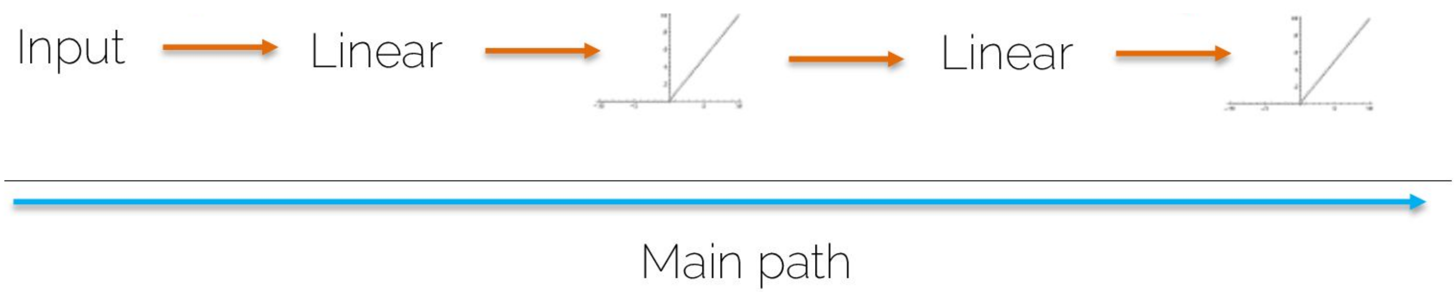


Non-linear

$$\longrightarrow x^{L+1} = f(W^{L+1}x^{L} + b^{L+1})$$

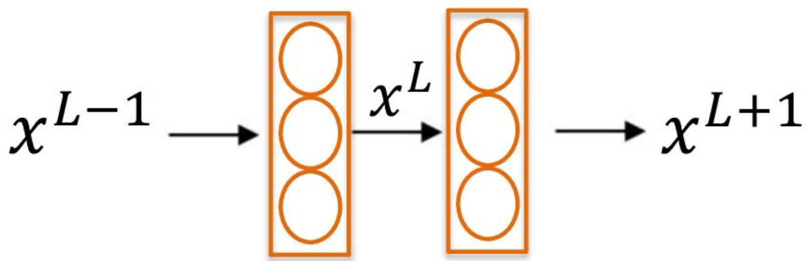
- Two layers

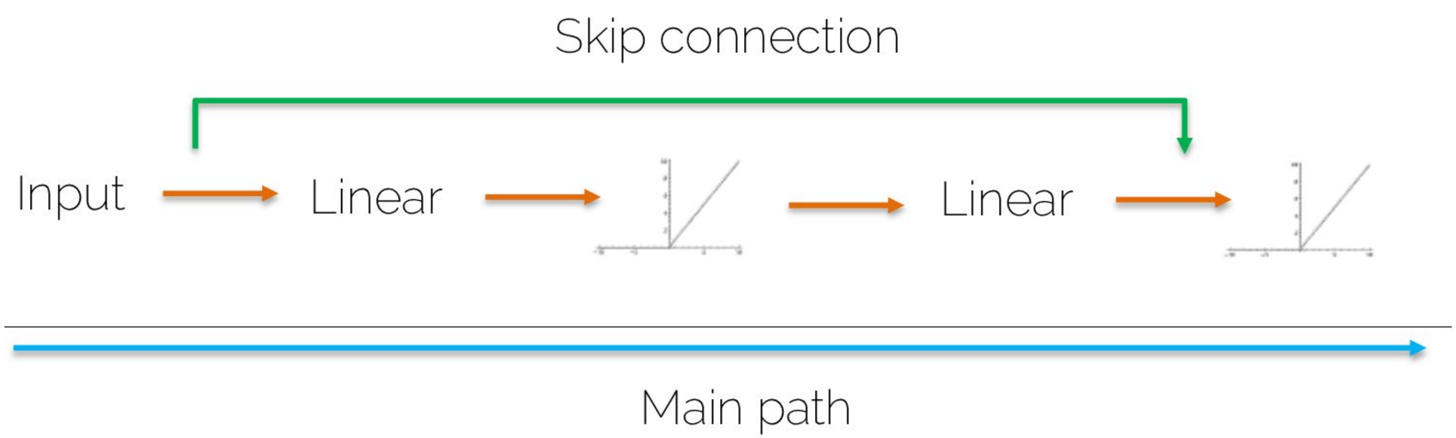




Residual Block (key idea)

- Two layers

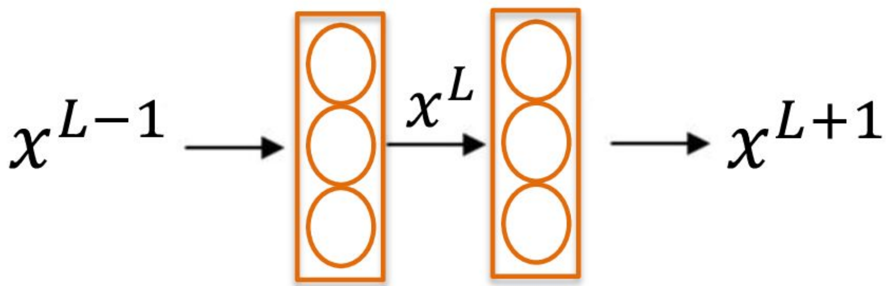




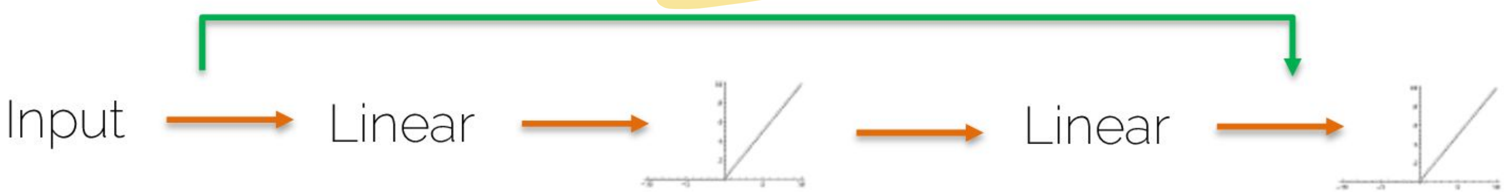
In each residual block, there is a skip connection (residual connection) that bypasses one or more layers and allows the gradient to flow directly to earlier layers.

Residual Block

- Two layers



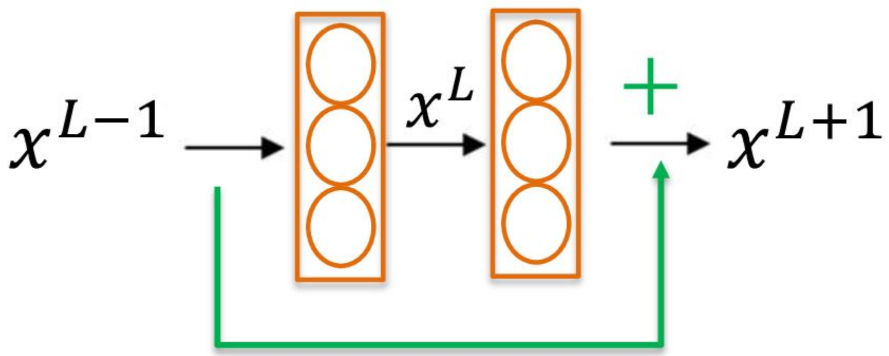
$$x^{L+1} = f(W^{L+1}x^{L} + b^{L+1} + x^{L-1})$$



In each residual block, there is a skip connection (residual connection) that bypasses one or more layers and allows the gradient to flow directly to earlier layers.

Residual Block

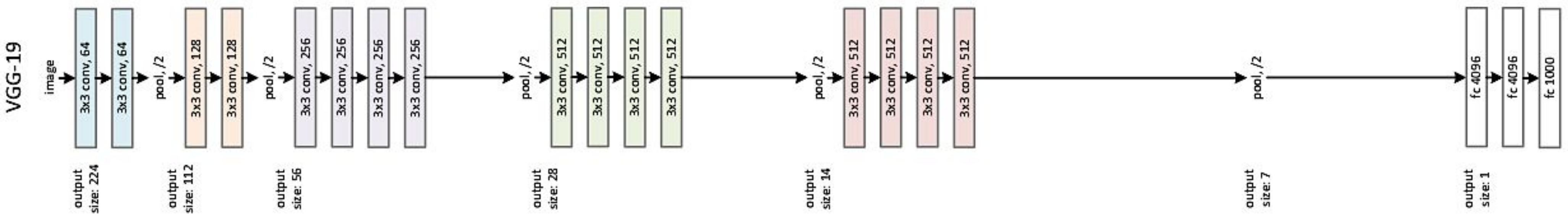
- Two layers



- Residual connection improve the gradient flow and enable the network to learn deeper and more complex features.

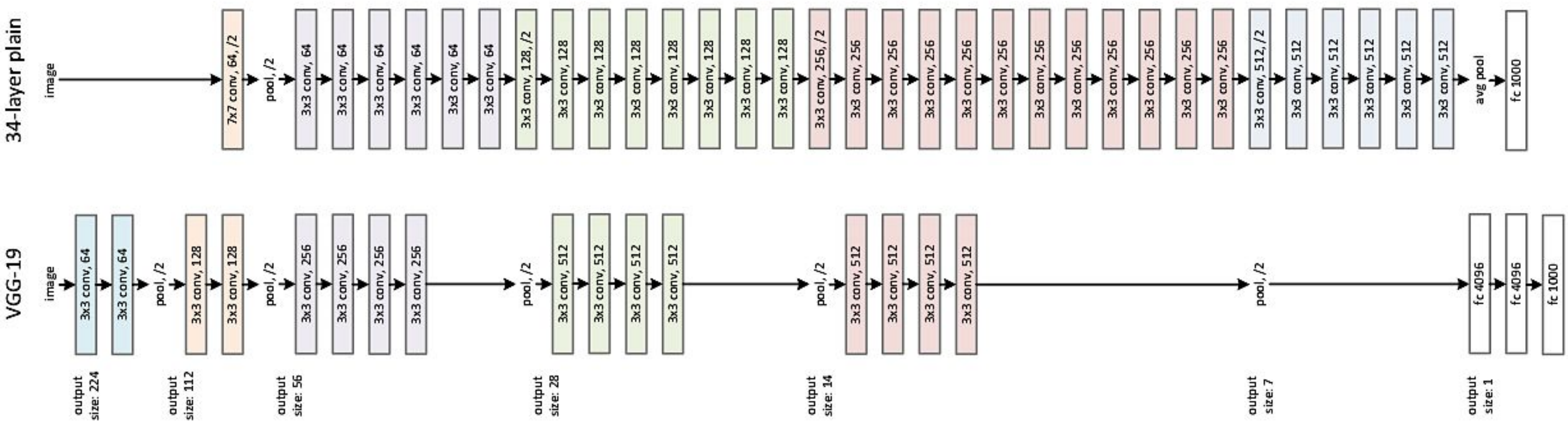
ResNet

ResNet



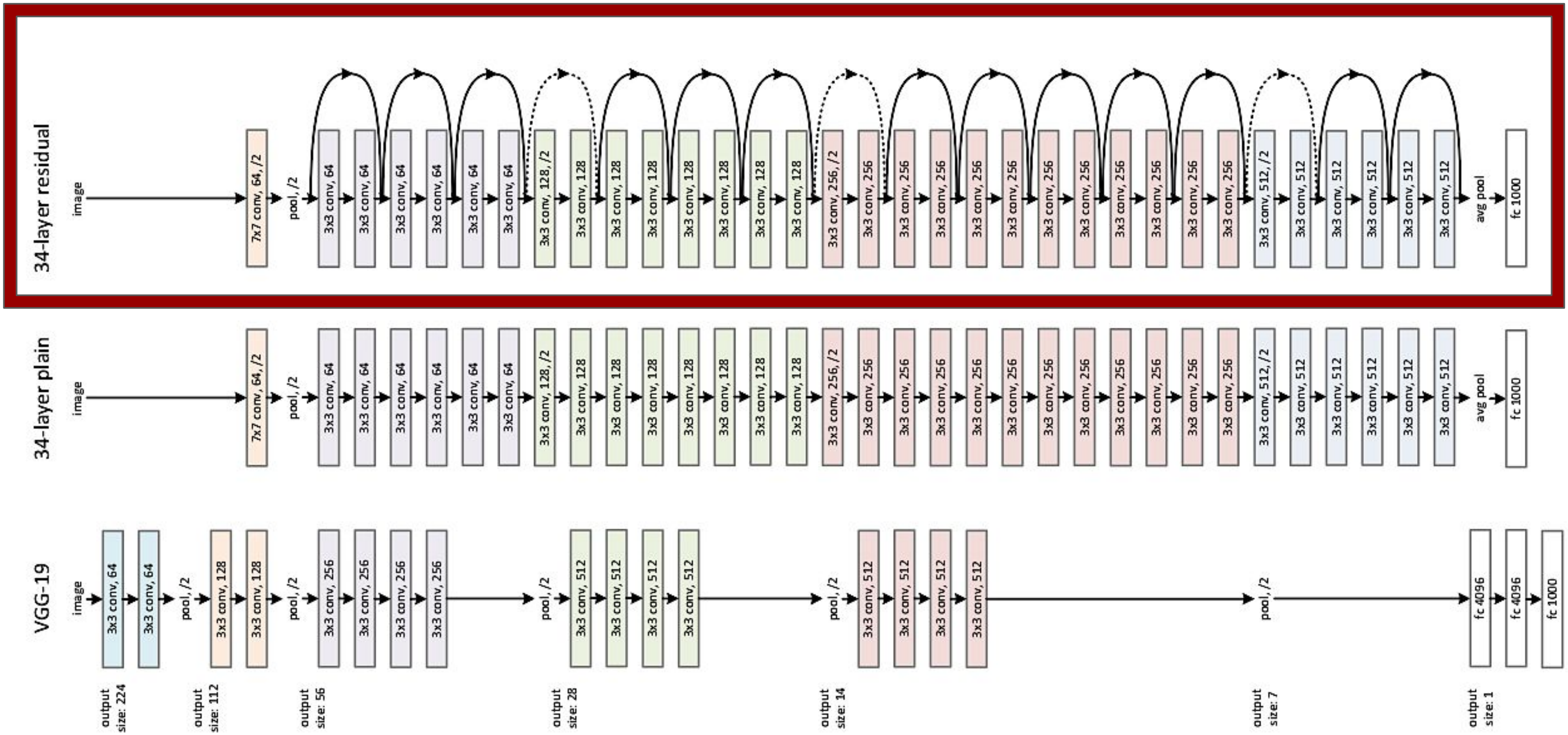
- **Bottom**: the VGG-19 model (19.6 billion FLOPs) as a reference.

ResNet



- **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs).
- **Bottom**: the VGG-19 model (19.6 billion FLOPs) as a reference.

ResNet



- **Top**: a residual network with 34 parameter layers (3.6 billion FLOPs).
- **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs).
- **Bottom**: the VGG-19 model (19.6 billion FLOPs) as a reference.

ResNet

- The ResNet architecture typically begins with a single convolutional layer, followed by a max pooling layer.

ResNet

- The ResNet architecture typically begins with a single convolutional layer, followed by a max pooling layer.
- After the initial layer, there are several stages of residual blocks with different number of convolutional layers.

ResNet

- The ResNet architecture typically begins with a single convolutional layer, followed by a max pooling layer.
- After the initial layer, there are several stages of residual blocks with different number of convolutional layers.
- Each residual block contains one or more convolutional layers, batch normalization, and ReLU activation functions.

ResNet

- The ResNet architecture typically begins with a single convolutional layer, followed by a max pooling layer.
- After the initial layer, there are several stages of residual blocks with different number of convolutional layers.
- Each residual block contains one or more convolutional layers, batch normalization, and ReLU activation functions.
- The convolutional layers in each residual block typically have small filter sizes, such as 3x3 or 1x1.

ResNet

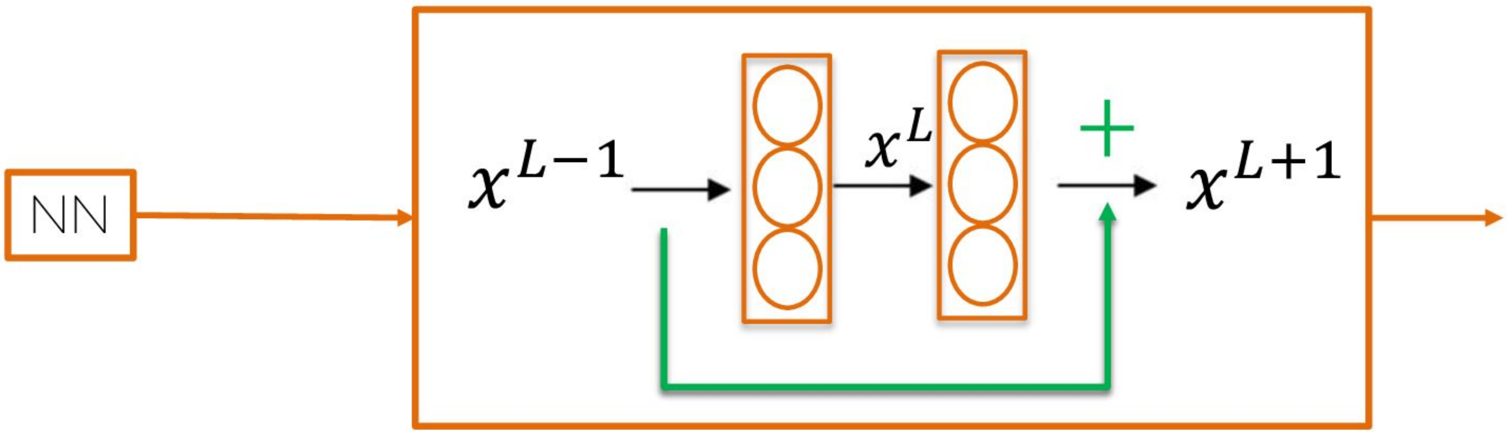
begins with single conv. Layer followed by max pooling.

- The ResNet architecture typically begins with a single convolutional layer, followed by a max pooling layer.
- After the initial layer, there are several stages of residual blocks with different number of convolutional layers.

several stages of residual blocks

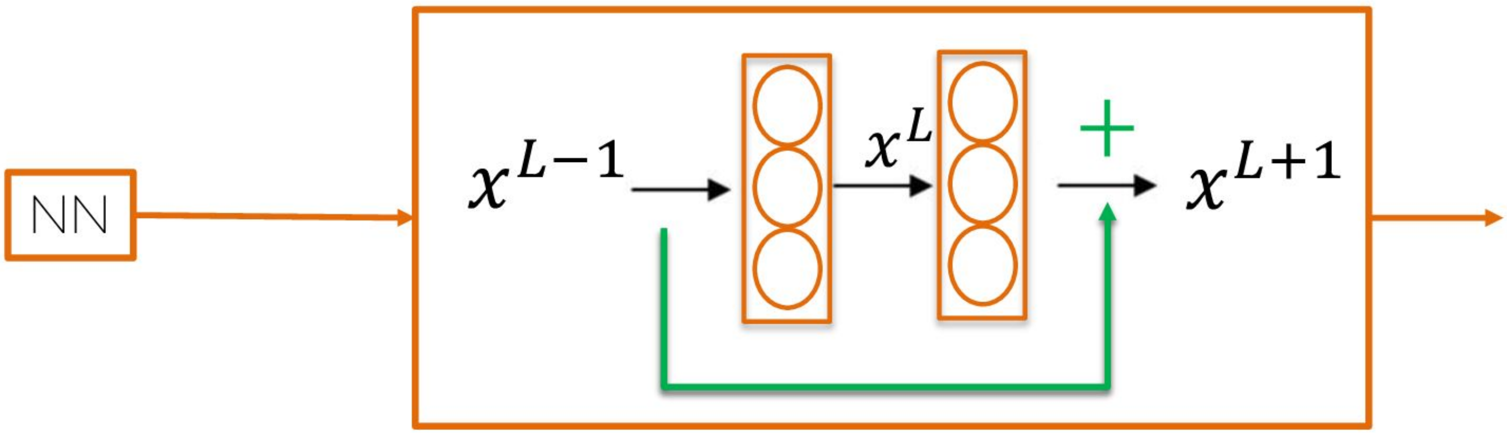
- Each residual block contains one or more convolutional layers, batch normalization, and ReLU activation functions.
- The convolutional layers in each residual block typically have small filter sizes, such as 3x3 or 1x1.
- The final layers of the network are typically global average pooling and a fully connected layer with a softmax activation function

Why do ResNets Work?



$$x^{L+1} = f(W^{L+1}x^{L} + b^{L+1} + x^{L-1})$$

Why do ResNets Work?

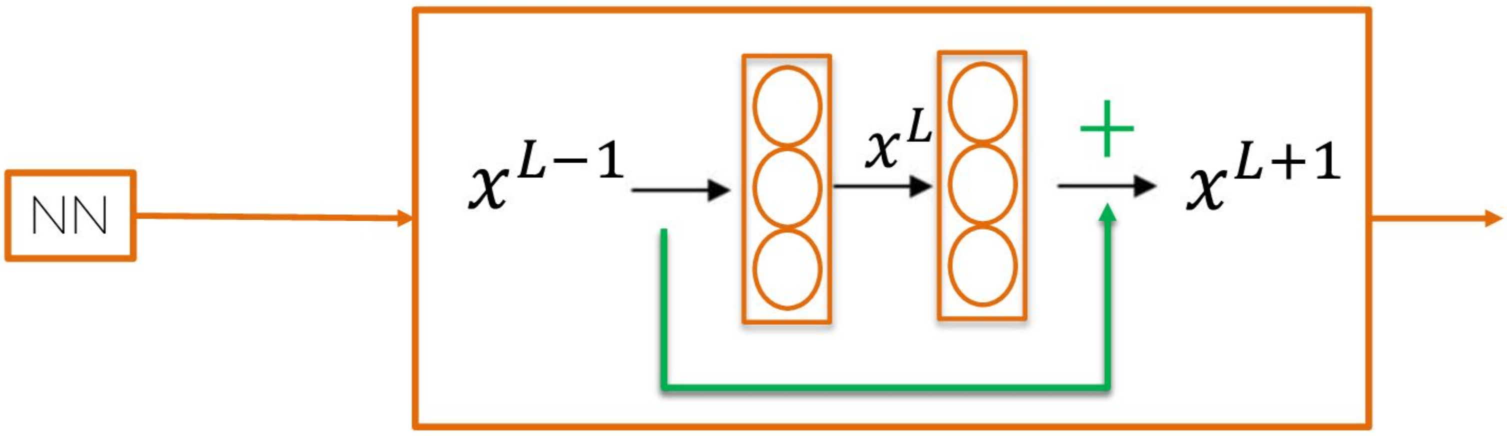


$$x^{L+1} = f(W^{L+1}x^{L} + b^{L+1} + x^{L-1})$$

~zero ~zero

$$x^{L+1} = f(x^{L-1})$$

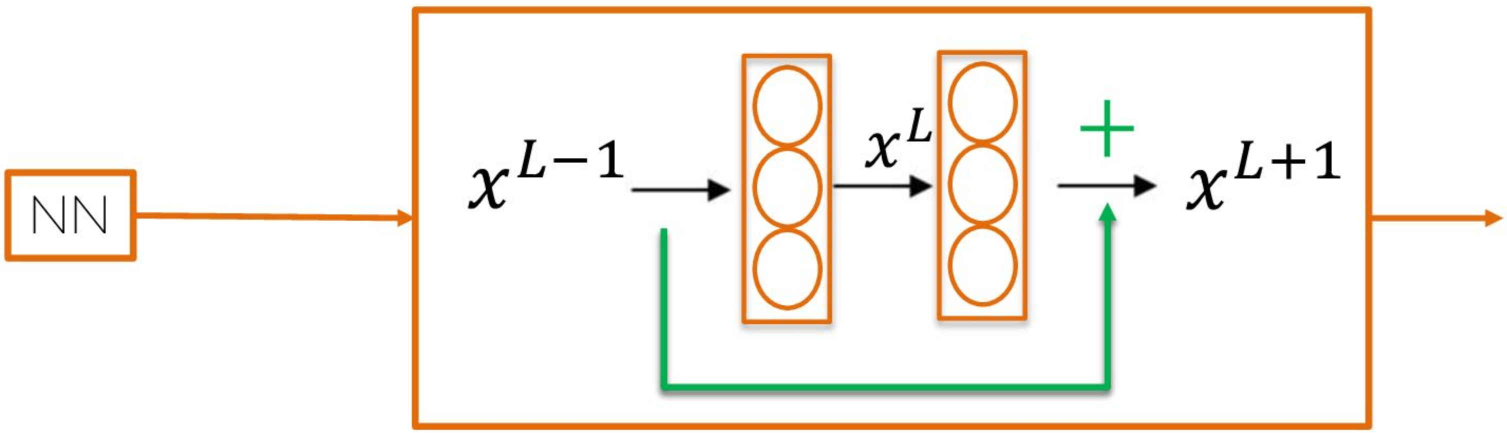
Why do ResNets Work?



We kept the same values and added a non-linearity.

$$x^{L+1} = f(x^{L-1})$$

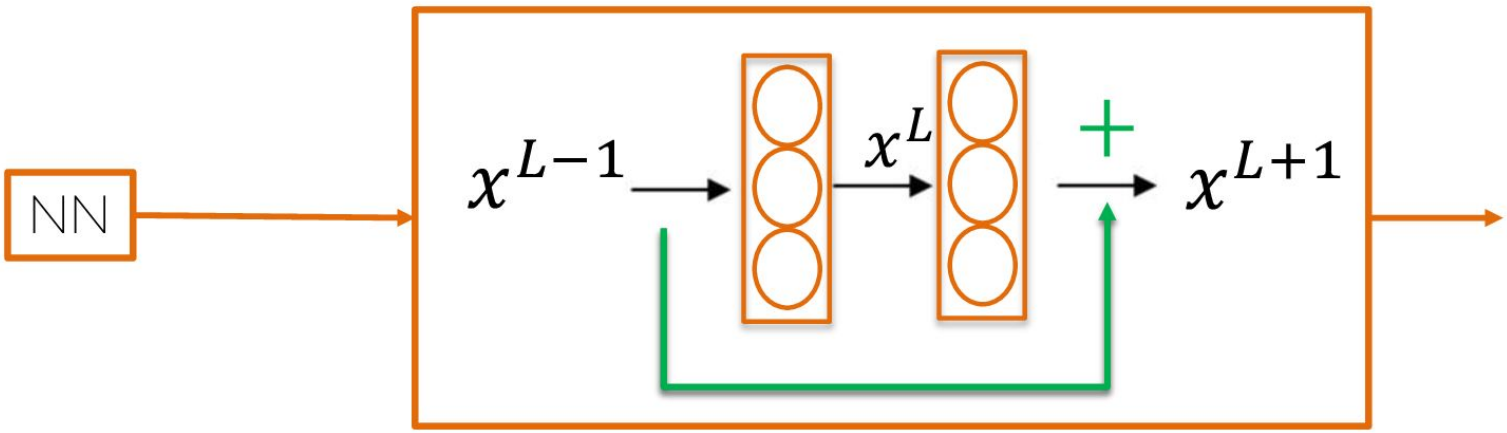
Why do ResNets Work?



The network can effectively choose to use fewer layers when it is not necessary, which can improve efficiency and reduce overfitting.

$$x^{L+1} = f(x^{L-1})$$

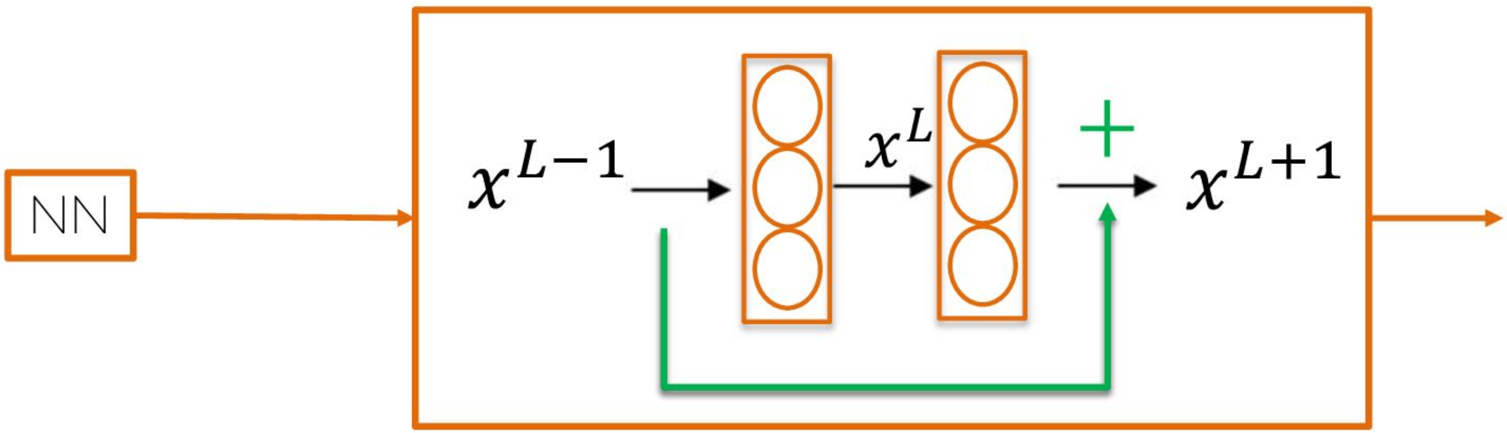
Why do ResNets Work?



Residual connections also allow the network to adaptively determine how many layers to use for a particular input.

$$x^{L+1} = f(x^{L-1})$$

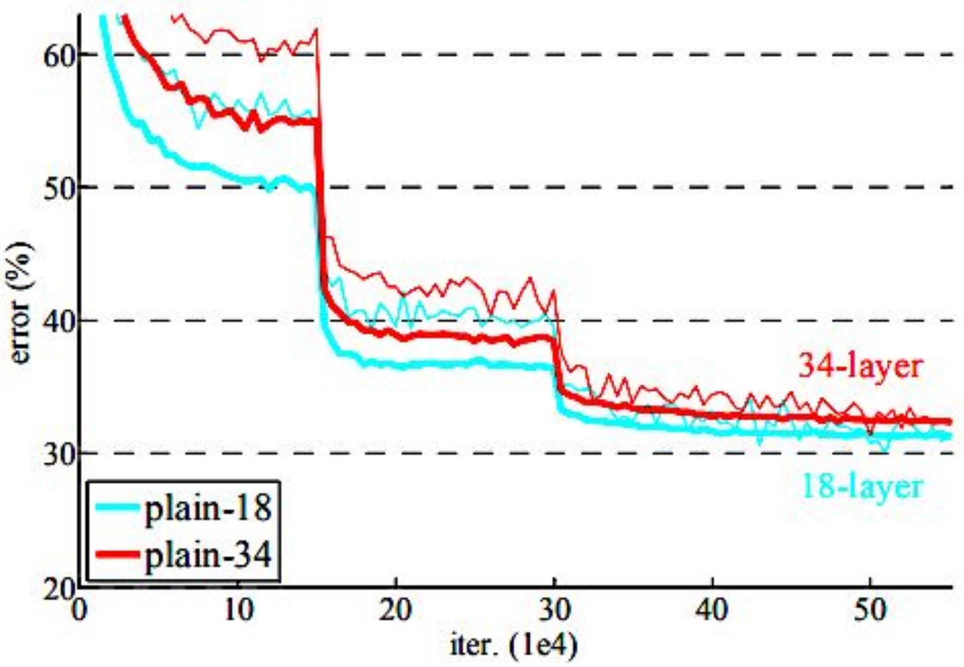
Why do ResNets Work?



- Shortcut connections in ResNets enable the gradient to flow more directly and efficiently through the network.
- ResNets address the problem of vanishing gradients that can occur in very deep neural networks.

ResNet

Training on ImageNet. If we make network deeper, at some point the performance starts to decrease.

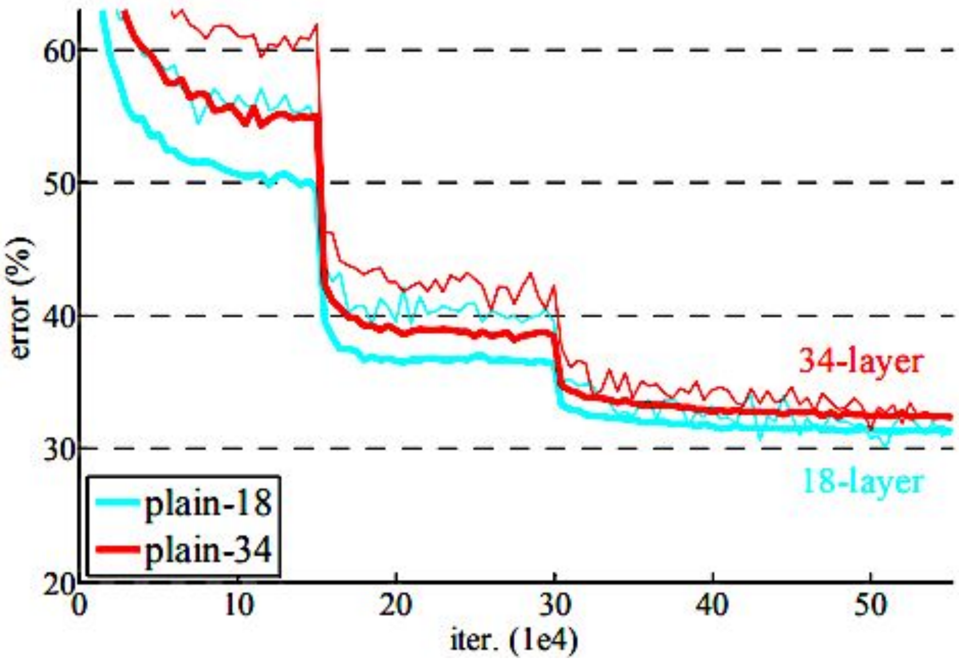


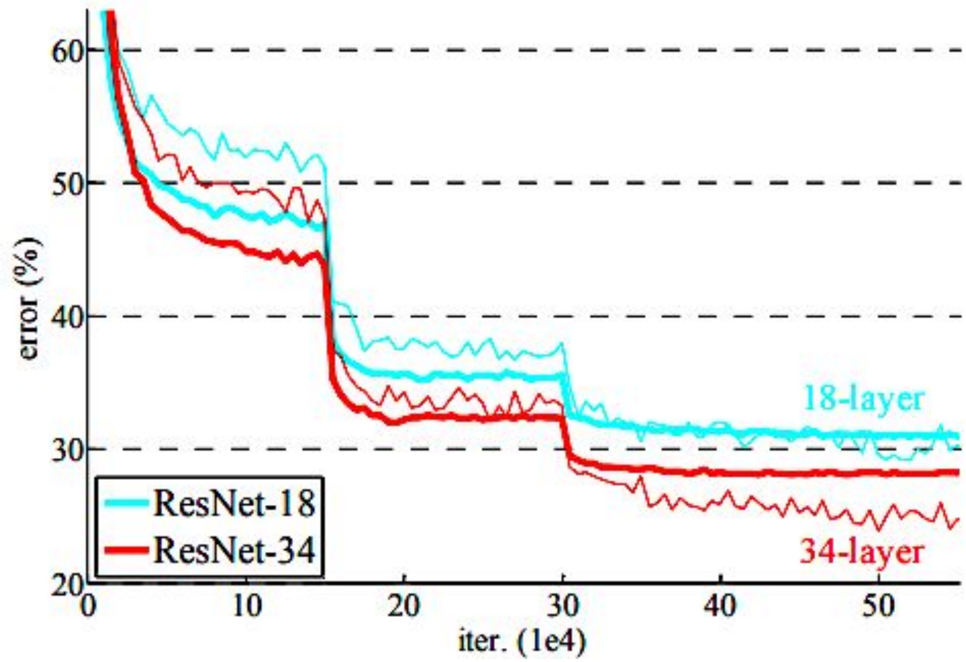
Left: plain networks of 18 and 34 layers.

Thin curves denote training error, and bold curves denote validation error of the center crops.

ResNet

Training on ImageNet. ResNet Solution





Left: plain networks of 18 and 34 layers.

Right: ResNets of 18 and 34 layers

Thin curves denote training error, and bold curves denote validation error of the center crops.

```python
import tensorflow as tf
from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input,
decode_predictions
from tensorflow.keras.preprocessing import image
import numpy as np

# Load the ResNet50 model
model = ResNet50(weights='imagenet')

# Load the image you want to classify
img_path = 'tiger_shark.jpeg'
img = image.load_img(img_path, target_size=(224, 224))

# Convert the image to an array
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

# Use the model to predict the class of the image
preds = model.predict(x)

# Print the top 5 predictions
print('Predicted:', decode_predictions(preds, top=5)[0])
```

References

- **AlexNet**: "ImageNet Classification with Deep Convolutional Neural Networks" by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton (2012).
- **VGGNet**: "Very Deep Convolutional Networks for Large-Scale Image Recognition" by Karen Simonyan and Andrew Zisserman (2014).
- **Inception Net**: "Going Deeper with Convolutions" by Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich (2015).
- **ResNet**: "Deep Residual Learning for Image Recognition" by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016).