یادگیری عمیق

مدرس: محمدرضا محمدی

زمستان ۱۴۰۱

شبکه‌های عصبی کانولوشنی

Convolutional Neural Networks

نرمال‌سازی دسته‌ای (Batch Normalization)

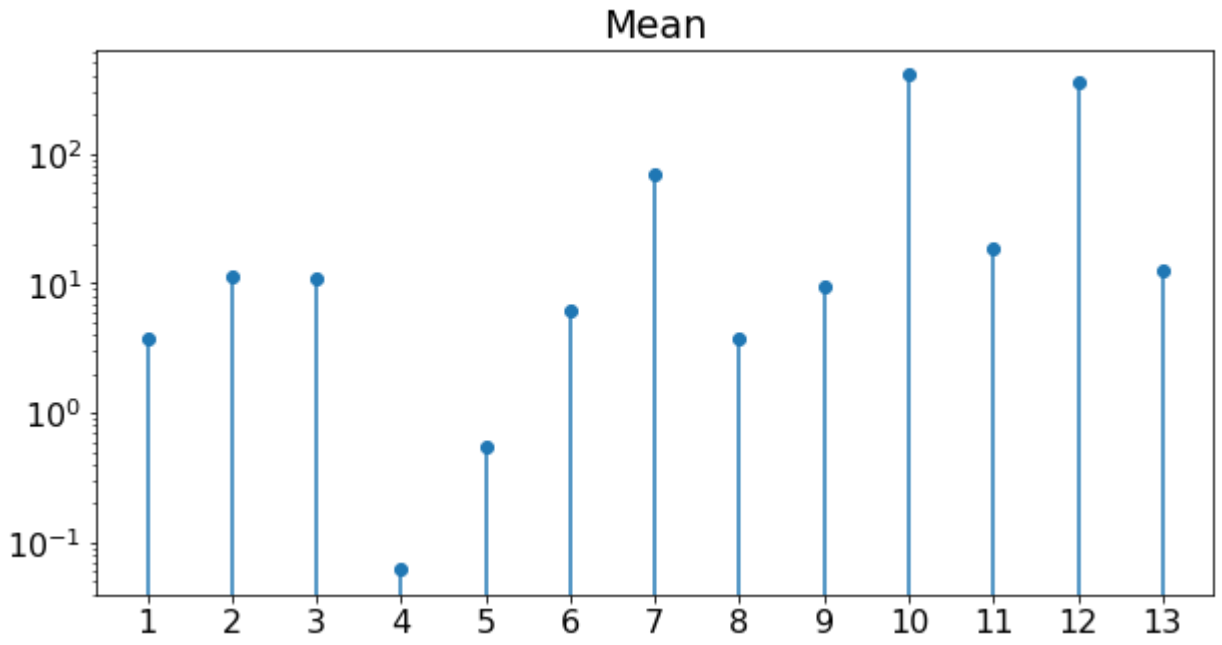
- آموزش شبکه‌های عمیق کار دشواری است
- نرمال‌سازی دسته‌ای یکی از تکنیک‌هایی است که همگرایی شبکه‌های عمیق را شتاب می‌دهد
- نرمال‌سازی در ورودی شبکه‌های عمیق هم بسیار مهم است
- مثال: تخمین قیمت خانه
  - مجموعه داده Boston Housing Price
    - نسبتا کوچک: ۴۰۴ داده آموزشی، ۱۰۲ داده آزمون
    - ویژگی‌های مجموعه داده دارای مقیاس‌های متفاوتی هستند
    - قیمت‌ها برحسب هزار دلار هستند

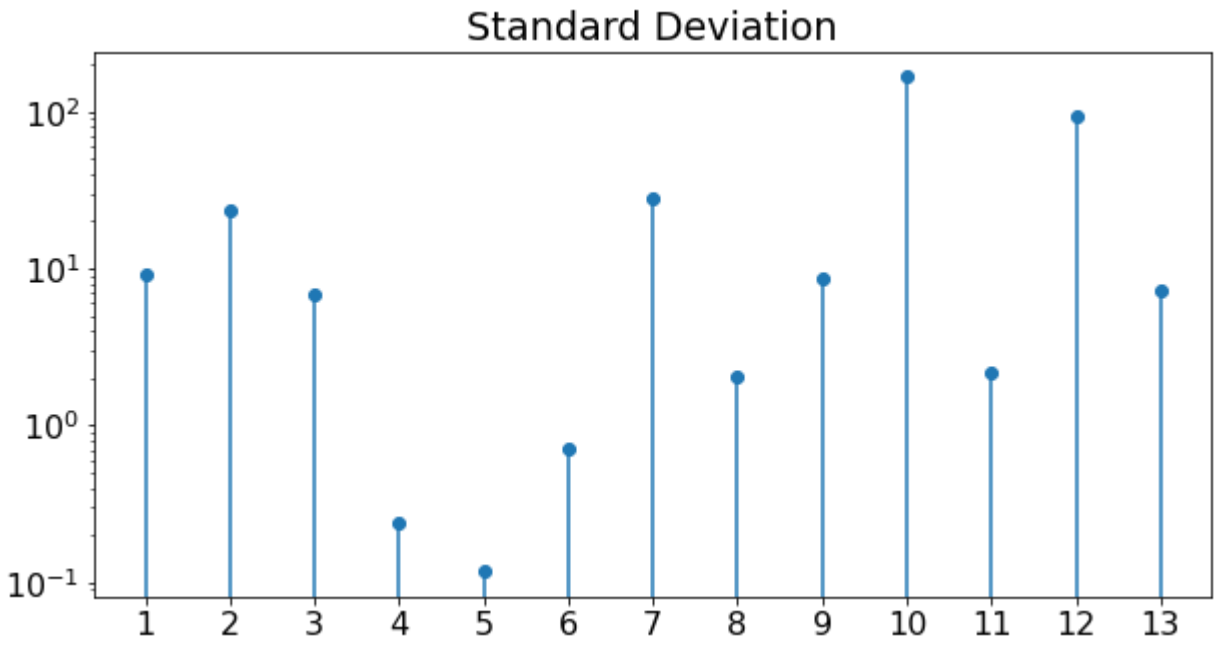
Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *ICML*, 2015.

مثال: تخمین قیمت خانه

```
mean = train_data.mean(axis=0)
std = train_data.std(axis=0)
train_data -= mean
train_data /= std
test_data -= mean
test_data /= std
```

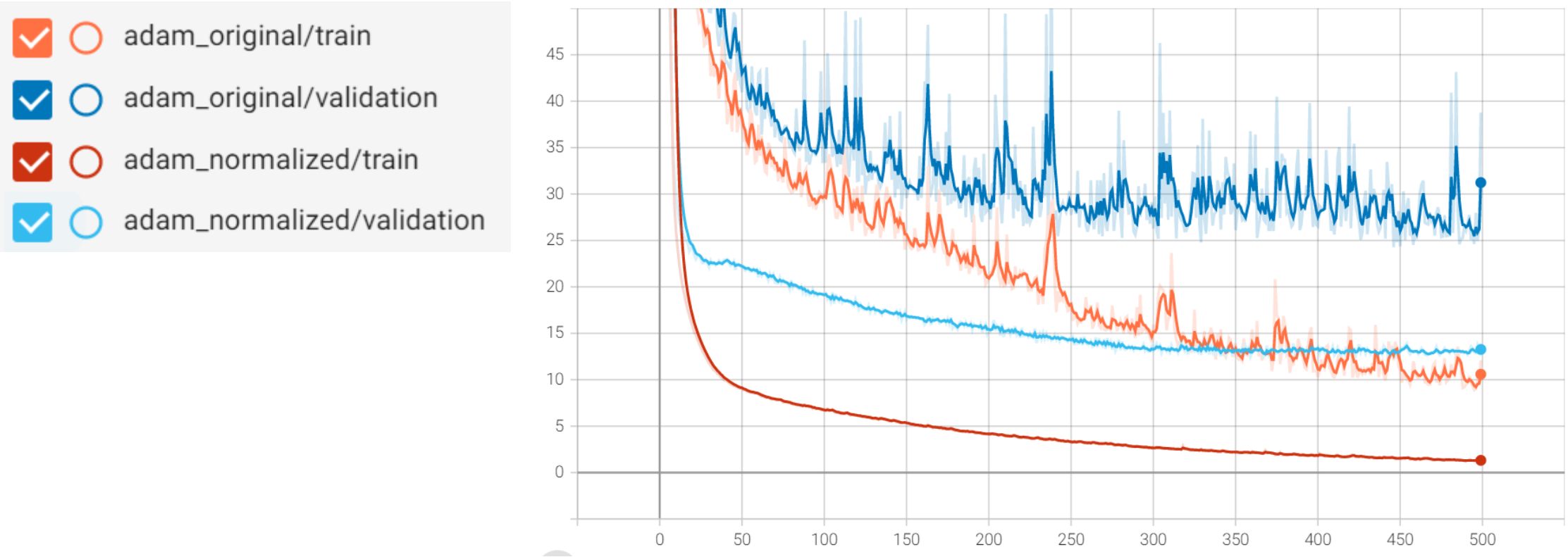
- ویژگی‌های مجموعه داده دارای مقیاس‌های متفاوتی هستند
- نرمال‌سازی ویژگی‌ها باعث ساده‌تر شدن فرآیند بهینه‌سازی می‌شود





مثال: تخمین قیمت خانه

- نرمال بودن داده‌ها هم در زمان آموزش و هم در زمان آزمون بسیار اثرگذار است



نرمال‌سازی دسته‌ای (Batch Normalization)

- نرمال‌سازی تأثیر چشمگیری بر عملکرد بهینه‌سازی دارد
  - به خصوص برای شبکه‌های کانولوشنی

- خروجی یک واحد از یک دسته $\mathcal{B}$ را در نظر بگیرید
  - می‌خواهیم میانگین آن صفر و واریانس آن یک شود
    - $\hat{\boldsymbol{\mu}}_{\mathcal{B}}$ و $\hat{\boldsymbol{\sigma}}_{\mathcal{B}}$ به ترتیب میانگین عددی و انحراف معیار عددی نمونه‌های این دسته هستند
    - مشتق این تابع به سادگی قابل محاسبه است

$$\frac{\mathbf{x} - \hat{\boldsymbol{\mu}}_{\mathcal{B}}}{\hat{\boldsymbol{\sigma}}_{\mathcal{B}}}$$

Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *ICML*, 2015.

نرمال‌سازی دسته‌ای (Batch Normalization)

- میانگین و واریانس عددی به صورت مستقل برای هر واحد محاسبه می‌شود
- ورودی: x $(N \times D)$
- میانگین هر کانال (به طول D)

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij}$$

- واریانس هر کانال (به طول D)

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{ij} - \mu_j)^2 + \epsilon$$

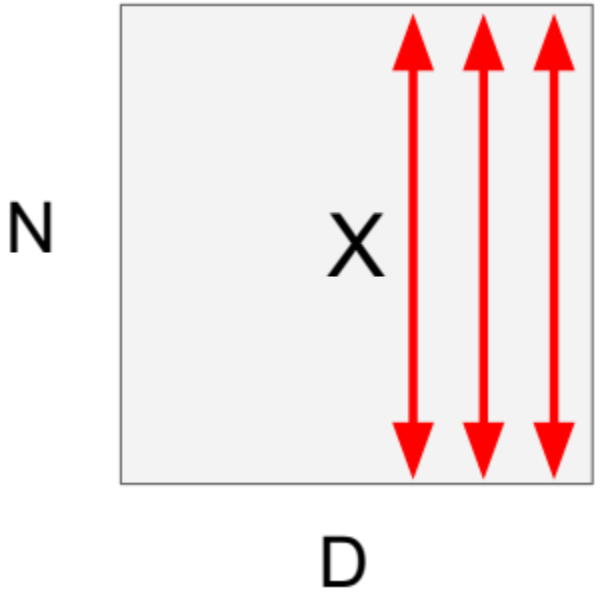
- x نرمال‌شده $(N \times D)$

$$\hat{x}_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}$$

- خروجی: y $(N \times D)$

Learnable scale and shift

$$y_{ij} = \gamma_j \hat{x}_{ij} + \beta_j$$



نرمال‌سازی دسته‌ای: زمان آزمون

- برآوردهای میانگین و واریانس به minibatch بستگی دارند
  - نمی‌توان این کار را در زمان آزمون انجام داد!
- از میانگین متحرک مقادیر (μ و σ^2) در حین آموزش استفاده می‌شود
- در زمان آزمون BN به یک عملگر خطی تبدیل می شود!
- می‌تواند با لایه کاملاً متصل یا کانولوشنی قبل ترکیب شود

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij}$$

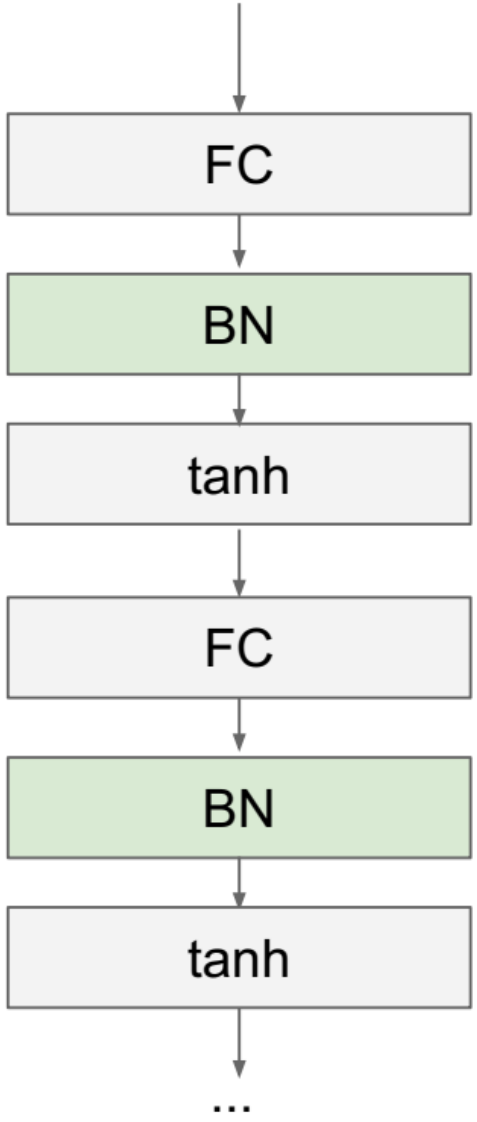
$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N} (x_{ij} - \mu_j)^2$$

$$\hat{x}_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}$$

$$y_{ij} = \gamma_j \hat{x}_{ij} + \beta_j$$

Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *ICML*, 2015.

نرمال‌سازی دسته‌ای



- معمولاً بعد از لایه‌های خطی و قبل از تابع فعال‌سازی غیرخطی استفاده می‌شوند
- جریان گرادیان را بهبود می‌بخشد
- آموزش شبکه‌های عمیق را ساده‌تر می‌کند!
- اجازه می‌دهد از نرخ یادگیری بالاتر استفاده کنیم و همگرایی را سرعت می‌دهد
- حساسیت به مقداردهی اولیه کاهش می‌یابد
- نویزی بودن تخمین میانگین و انحراف معیار باعث جلوگیری از بیش‌برازش می‌شود
  - در زمان آموزش به نوعی عمل منظّم‌سازی را انجام می‌دهد
- در زمان آزمون سرباری اضافه نمی‌کند
  - می‌تواند با لایه خطی قبل ترکیب شود

Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *ICML*, 2015.

BN برای لایه‌های کانولوشنی

Batch Normalization for fully-connected layers

$x: N \times D$

$\mu, \sigma: 1 \times D$

$\gamma, \beta: 1 \times D$

$$y = \gamma \frac{x - \mu}{\sigma} + \beta$$

Batch Normalization for convolutional layers

$x: N \times W \times H \times C$

$\mu, \sigma: 1 \times 1 \times 1 \times C$

$\gamma, \beta: 1 \times 1 \times 1 \times C$

$$y = \gamma \frac{x - \mu}{\sigma} + \beta$$

```
import torch
from torch import nn

def batch_norm(X, gamma, beta, moving_mean, moving_var, eps, momentum):
    # Use 'is_grad_enabled' to determine whether the current mode is training mode or prediction mode
    if not torch.is_grad_enabled():
        # If it is prediction mode, directly use the mean and variance obtained by moving average
        X_hat = (X - moving_mean) / torch.sqrt(moving_var + eps)
    else:
        assert len(X.shape) in (2, 4)
        if len(X.shape) == 2:
            # When using a fully-connected layer, calculate the mean and variance on the feature dimension
            mean = X.mean(dim=0)
            var = ((X - mean) ** 2).mean(dim=0)
        else:
            # When using a two-dimensional convolutional layer, calculate the mean and variance on the
            # channel dimension (axis=1). Here we need to maintain the shape of 'X', so that the
            # broadcasting operation can be carried out later
            mean = X.mean(dim=(0, 2, 3), keepdim=True)
            var = ((X - mean) ** 2).mean(dim=(0, 2, 3), keepdim=True)
        # In training mode, the current mean and variance are used for the standardization
        X_hat = (X - mean) / torch.sqrt(var + eps)
        # Update the mean and variance using moving average
        moving_mean = momentum * moving_mean + (1.0 - momentum) * mean
        moving_var = momentum * moving_var + (1.0 - momentum) * var
    Y = gamma * X_hat + beta  # Scale and shift
    return Y, moving_mean.data, moving_var.data
```

نرمال‌سازی لایه‌ای (Layer Normalization)

Batch Normalization for fully-connected layers

$x: N \times D$

$\mu, \sigma: 1 \times D$

$\gamma, \beta: 1 \times D$

$$y = \gamma \frac{x - \mu}{\sigma} + \beta$$

Layer Normalization for fully-connected layers

$x: N \times D$

$\mu, \sigma: N \times 1$

$\gamma, \beta: 1 \times D$

$$y = \gamma \frac{x - \mu}{\sigma} + \beta$$

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer normalization." *arXiv preprint,* 2016.

نرمال‌سازی لایه‌ای (Layer Normalization)

- نرمال‌سازی لایه‌ای برای هر نمونه محاسبات را به صورت جداگانه انجام می‌دهد
- رفتار آن در زمان آموزش و آزمون کاملا یکسان است و نیازی به ذخیره‌سازی مقادیر ندارد
- برای لایه‌های بازگشتی هم به سادگی قابل اعمال است
  - در هر زمان محاسبات را جداگانه انجام می‌دهد

Layer Normalization for fully-connected layers

$x: N \times D$

$\mu, \sigma: N \times 1$

$\gamma, \beta: 1 \times D$

$$y = \gamma \frac{x - \mu}{\sigma} + \beta$$

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer normalization." *arXiv preprint,* 2016.

نرمال‌سازی نمونه‌ای (Instance Normalization)

Batch Normalization for convolutional layers

$$x: N \times W \times H \times C$$

$$\mu, \sigma: 1 \times 1 \times 1 \times C$$

$$\gamma, \beta: 1 \times 1 \times 1 \times C$$

$$y = \gamma \frac{x - \mu}{\sigma} + \beta$$

Instance Normalization for convolutional layers

$$x: N \times W \times H \times C$$

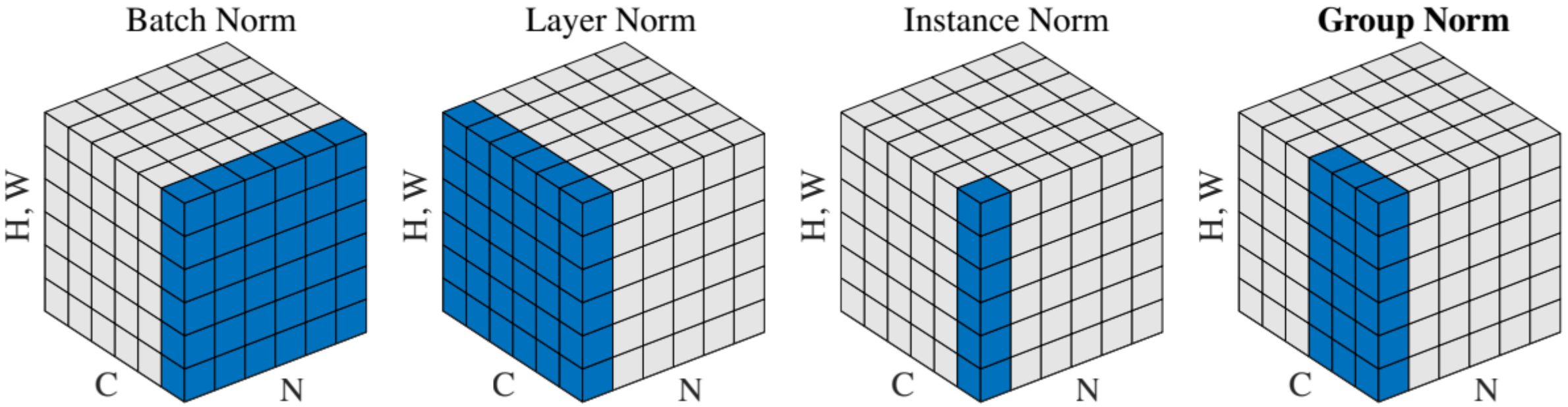
$$\mu, \sigma: N \times 1 \times 1 \times C$$

$$\gamma, \beta: 1 \times 1 \times 1 \times C$$

$$y = \gamma \frac{x - \mu}{\sigma} + \beta$$

D. Ulyanov, A. Vedaldi, and V. Lempitsky. "Instance normalization: The missing ingredient for fast stylization." *arXiv preprint,* 2016.

مقایسه روش‌های نرمال‌سازی



Y. Wu and K. He, "Group normalization." *ECCV*, 2018.

مقایسه روش‌های نرمال‌سازی

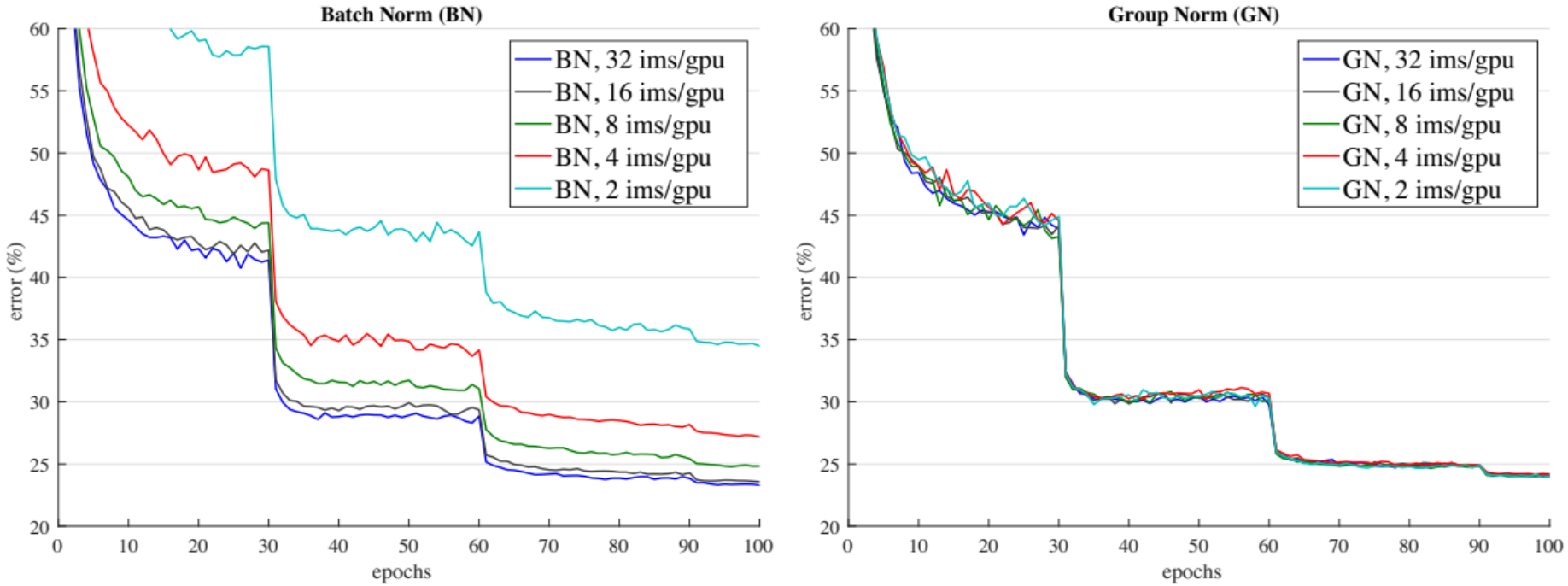
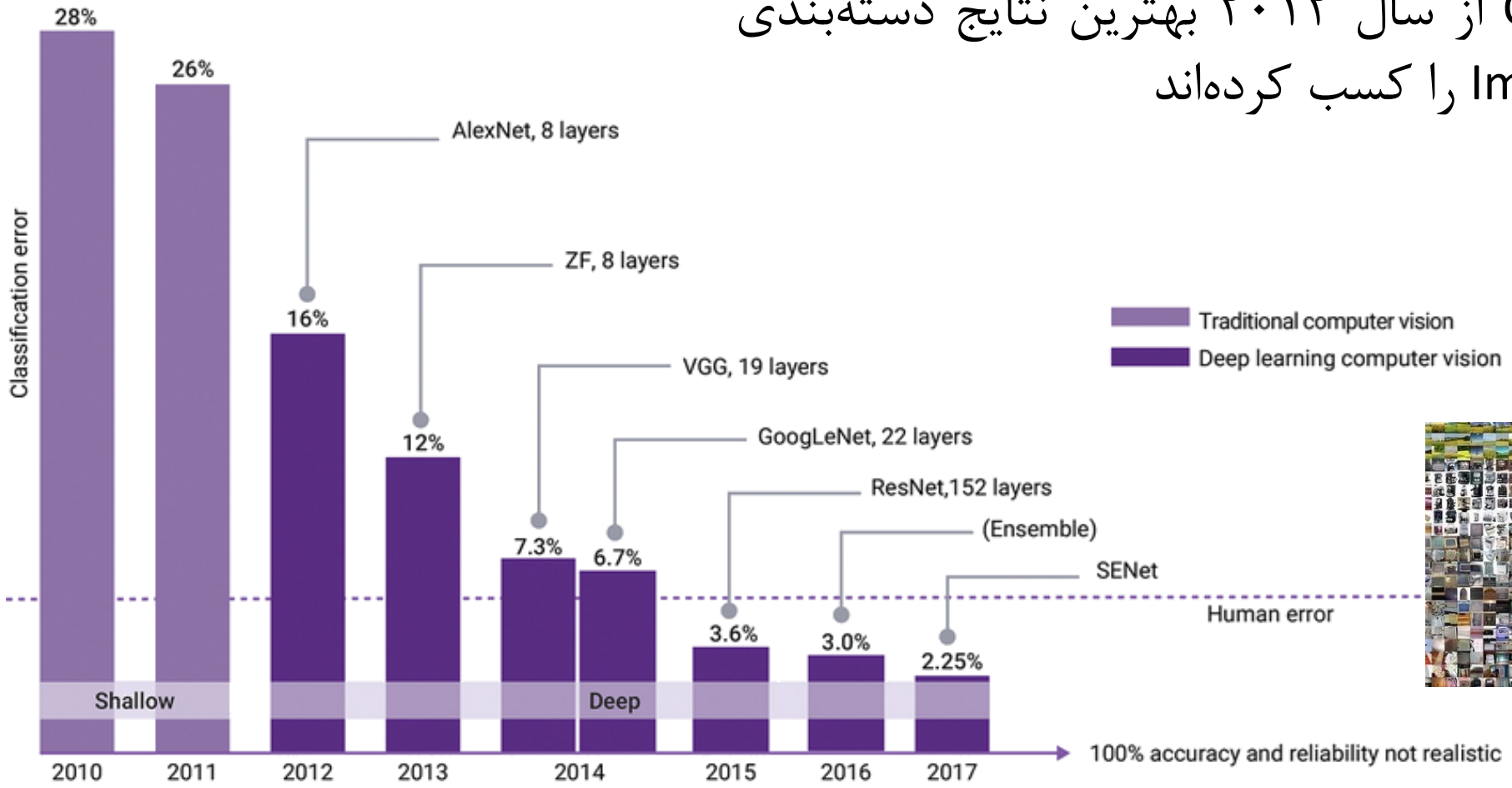


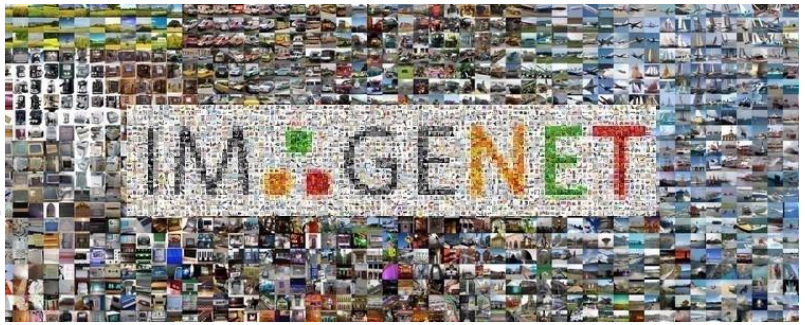
Figure 5. **Sensitivity to batch sizes**: ResNet-50's validation error of BN (left) and GN (right) trained with 32, 16, 8, 4, and 2 images/GPU.

Y. Wu and K. He, "Group normalization." *ECCV*, 2018.

نتایج ILSVRC

- معماری‌های مختلف CNN از سال ۲۰۱۲ بهترین نتایج دسته‌بندی تصویر در چالش ImageNet را کسب کرده‌اند

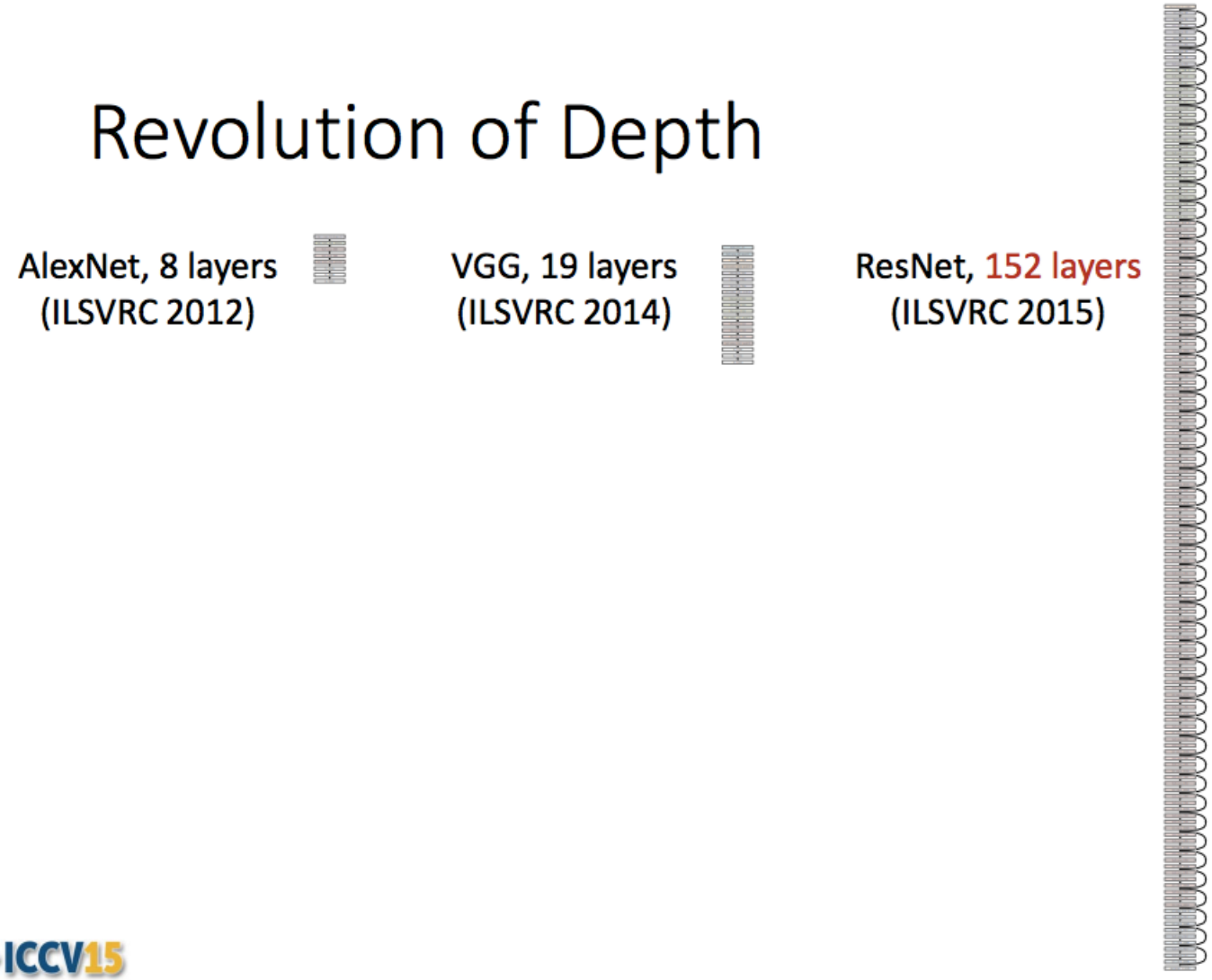




https://www.synopsys.com/designware-ip/technical-bulletin/computer-vision-lab-life.html

ResNet

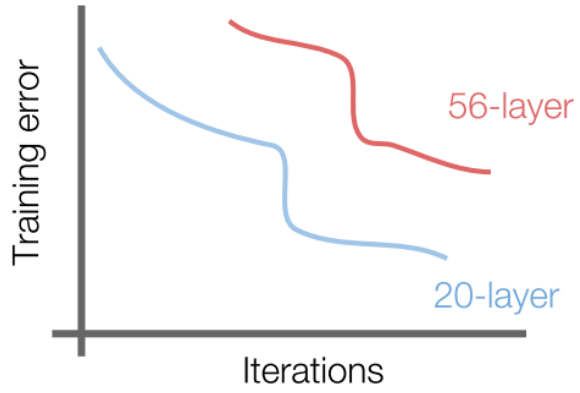
- شبکه ResNet برنده مسابقه ILSVRC’15 با خطای ۳.۵۷٪ شد
- با ۱۵۲ لایه، انقلابی در عمق شبکه‌های کانولوشنی به وجود آورد

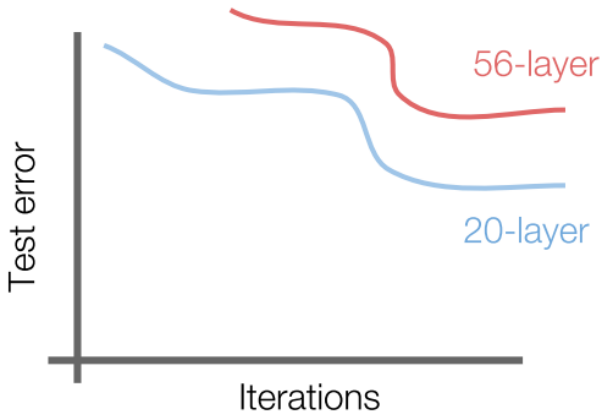


ICCV15

He, Kaiming, et al. "Deep residual learning for image recognition." *CVPR,* 2016.

ResNet



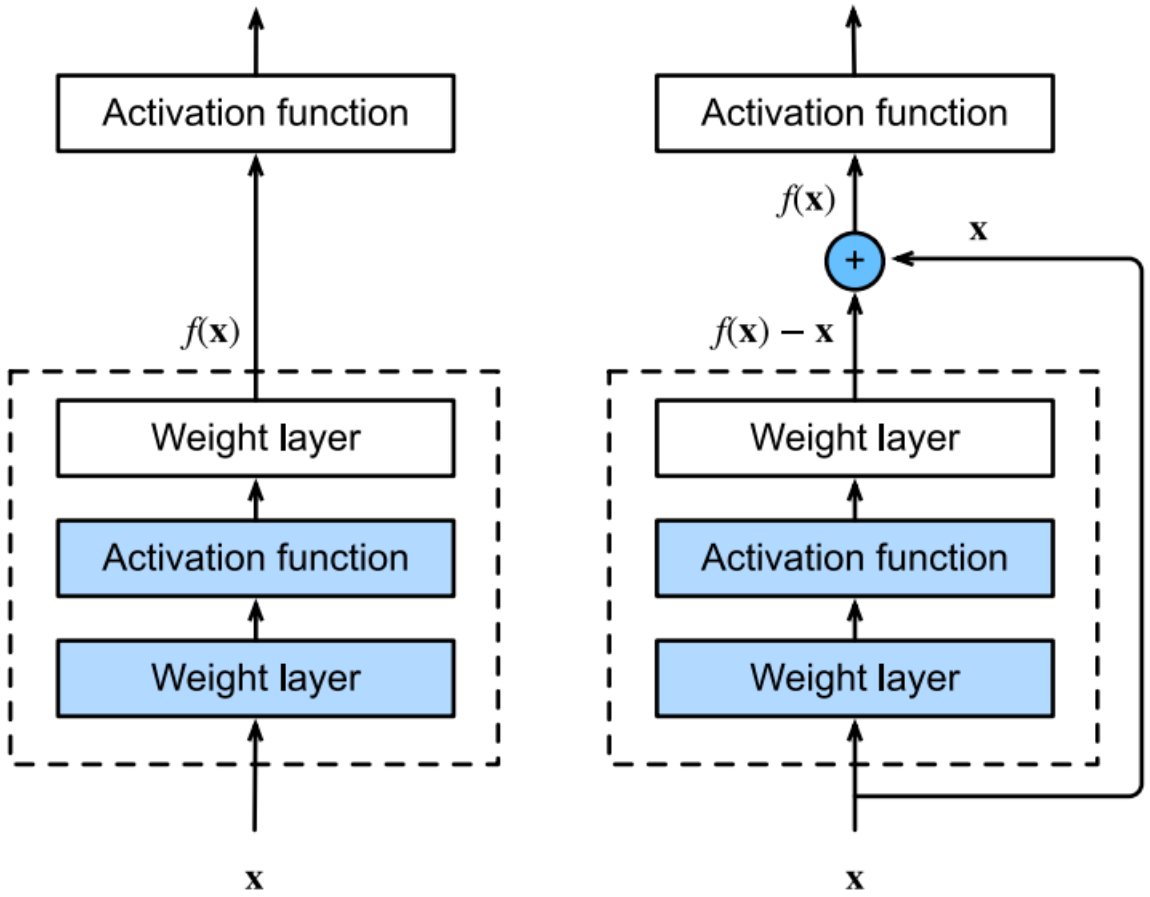


- اگر تعداد لایه‌های کانولوشنی ساده را بسیار زیاد کنیم چه اتفاقی می‌افتد؟
- چرا شبکه عمیق‌تر هم در آموزش و هم در آزمون عملکرد ضعیف‌تری دارد؟
  - البته مشکل از overfitting نیست!
- فرضیه: مشکل در مسئله بهینه‌سازی است
  - بهینه‌سازی مدل‌های عمیق‌تر دشوارتر است
- عملکرد مدل‌های عمیق‌تر باید حداقل به خوبی مدل‌های با عمق کمتر باشد
  - می‌توان وزن‌های مدل کم‌عمق را به لایه‌های نخست شبکه عمیق کپی کرد و لایه‌های اضافی را به گونه‌ای تنظیم کرد که نگاشت همانی را انجام دهند
- ایده ResNet آن است که لایه‌های شبکه بجای آموختن نگاشت مطلوب، باقی‌مانده آن را یاد بگیرند

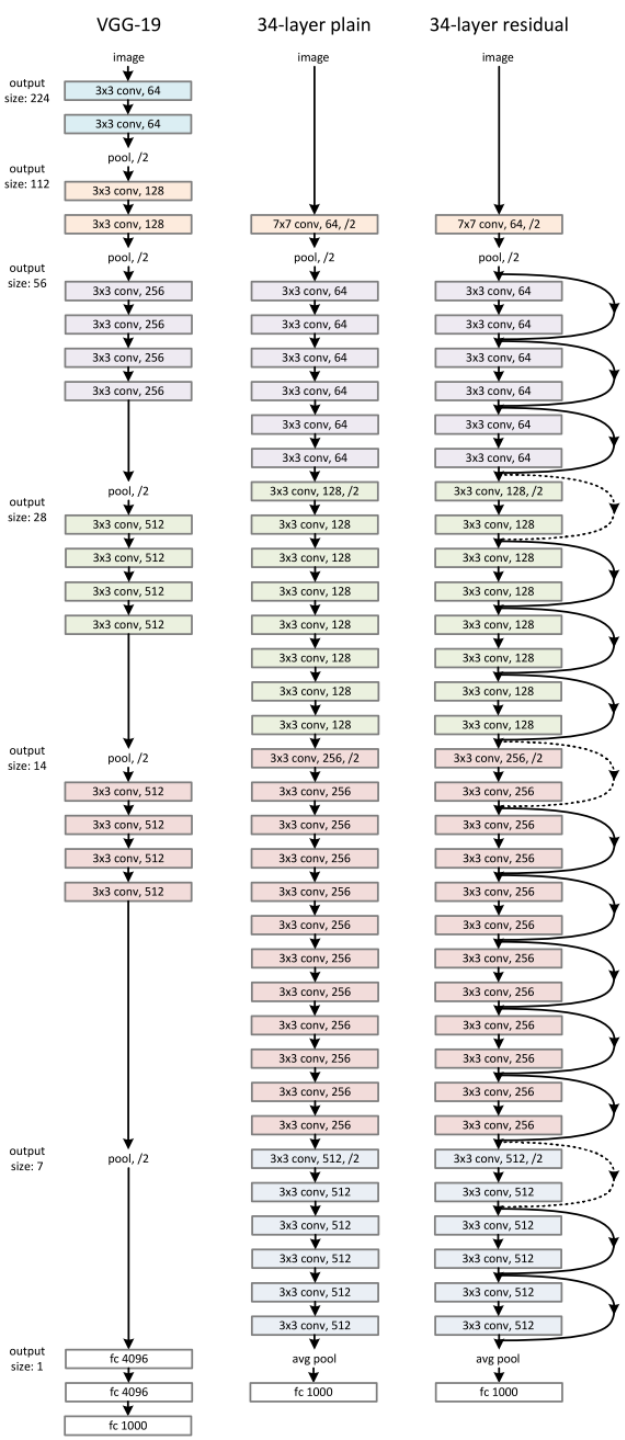
He, Kaiming, et al. "Deep residual learning for image recognition." *CVPR*, 2016.

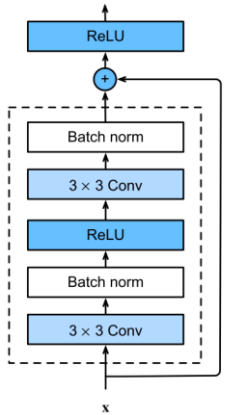
ResNet

- باید $f(\mathbf{x}) - \mathbf{x}$ بجای $f(\mathbf{x})$ آموخته شود



He, Kaiming, et al. "Deep residual learning for image recognition." *CVPR,* 2016.





ResNet

- از تعداد زیادی بلوک باقی‌مانده تشکیل شده است
- هر بلوک باقی‌مانده دارای ۲ لایه کانولوشنی ۳×۳ است
- به طور دوره‌ای، تعداد فیلترها ۲ برابر شده و رزولوشن مکانی نصف می‌شود
- در ابتدا دارای یک لایه کانولوشنی است
- پس از آخرین بلوک باقی‌مانده، ابعاد داده‌ها با استفاده از Average Pooling کاهش می‌یابد و یک لایه FC برای دسته‌بندی استفاده می‌شود
- برای مسئله ImageNet عمق‌های مختلف شبکه شامل ۳۴، ۵۰، ۱۰۱ و ۱۵۲ استفاده شده‌اند
- در شبکه‌های عمیق‌تر، از لایه کانولوشنی ۱×۱ برای بهبود بهره‌وری استفاده شده است

He, Kaiming, et al. "Deep residual learning for image recognition." *CVPR*, 2016.

ResNet

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

He, Kaiming, et al. "Deep residual learning for image recognition." *CVPR,* 2016.