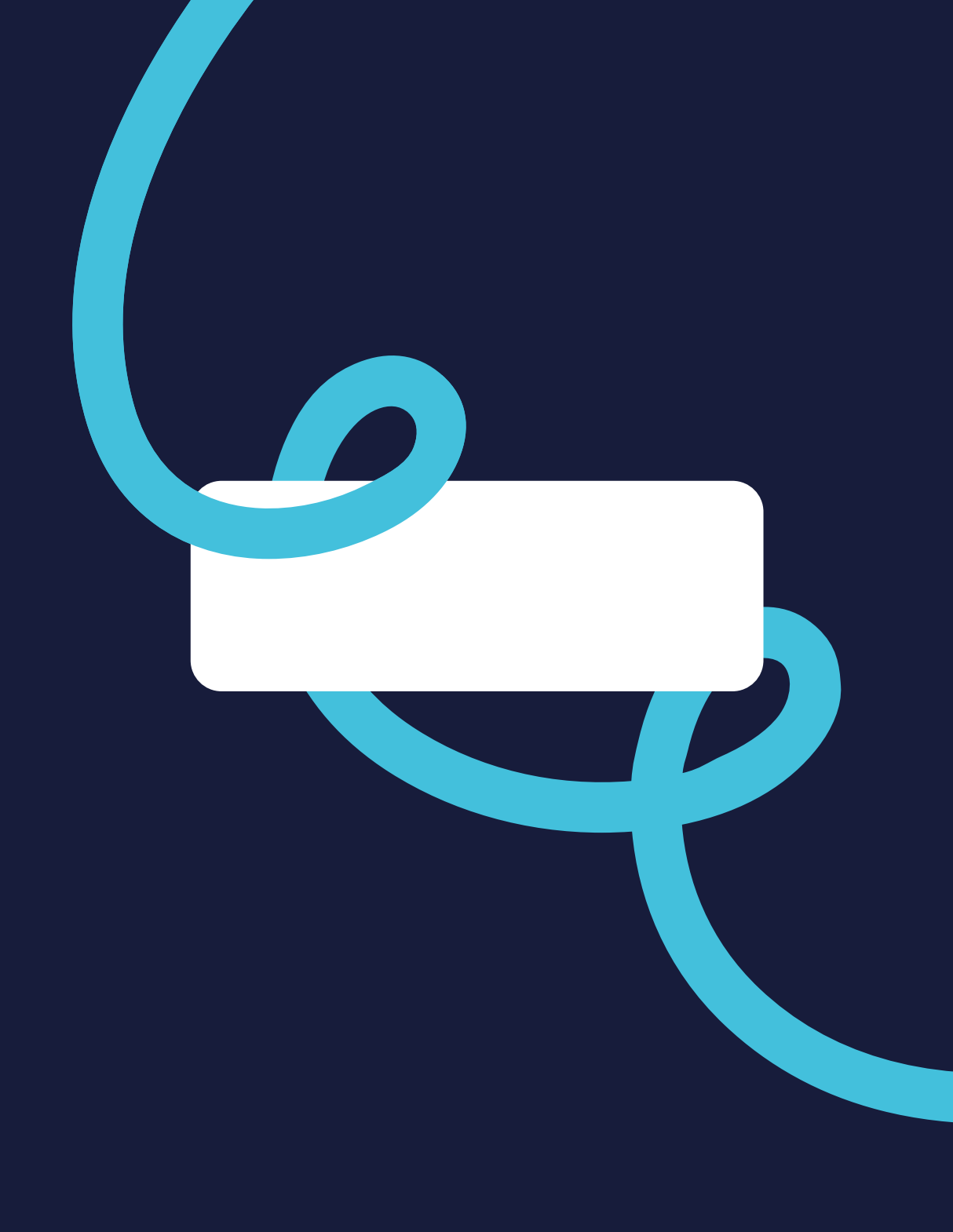
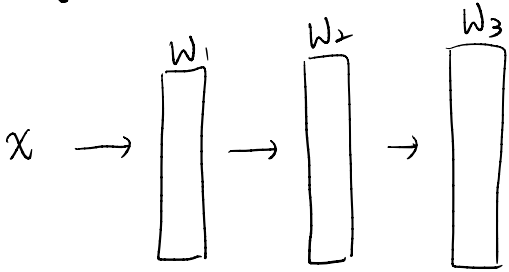


ResNet

1. What is the main problems of general network?

- Gradient Vanishing / Explosion

Assuming a single model



$$\mathcal{L} = \frac{1}{2}(\hat{y}-y)^2$$

If we want to calculate gradient related to each param...

$$a = W_1 X$$
$$b = W_2 a \Rightarrow y = c = W_3(W_2(W_1 X))$$
$$c = W_3 b$$
$$\mathcal{L} = \frac{1}{2}(\hat{y}-c)^2$$

$$\frac{d\mathcal{L}}{dW_3} = \frac{d\mathcal{L}}{dc}\frac{dc}{dW_3} = (\hat{y}-c)\cdot b$$

$$\frac{d\mathcal{L}}{dW_2} = \frac{d\mathcal{L}}{dc}\frac{dc}{db}\frac{db}{dW_2} = (\hat{y}-c)\cdot W_3 \cdot a$$

$$\frac{d\mathcal{L}}{dW_1} = \frac{d\mathcal{L}}{dc}\cdots = (\hat{y}-c)\cdot W_3 \cdot W_2 \cdot x$$

If we have 1000 layers, the params in first layer will be $(\hat{y}-c)\cdot \underbrace{W_{1000}\cdots W_2}_{1000-1}$

a. What if $w_2 \sim w_{1000}$ are small?

$$\frac{\partial \mathcal{L}}{\partial w_1} \sim 0 \quad , \quad w_1' \leftarrow w_1 - \eta \frac{\partial \mathcal{L}}{\partial w_1} \sim w_1$$

The weight will not update!

G vanishing

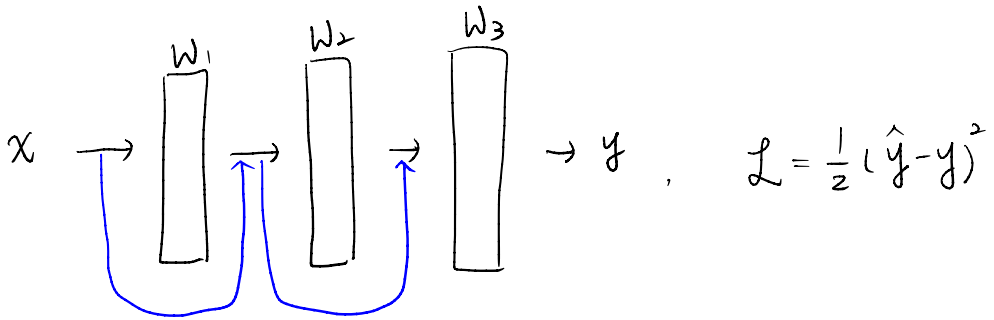
b. What if $w_2 \sim w_{1000}$ are large?

$$\frac{\partial \mathcal{L}}{\partial w_1} \sim \infty \quad , \quad w_1' \rightarrow -\infty$$

G explosion

2. Skip Connection

* Dealing with gradient vanishing problem.



$$\mathcal{L} = \frac{1}{2}(\hat{y}-y)^2$$

Now we change to ...

$$a = w_1 x + x = x(w_1+1)$$
$$b = w_2 a + a = a(w_2+1)$$
$$c = w_3 b + b = b(w_3+1)$$

$$\frac{\partial \mathcal{L}}{\partial w_3} = (\hat{y}-c)\cdot b$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = (\hat{y}-c)\cdot(w_3+1)(w_2+1)\cdot x$$
$$= (\hat{y}-c)w_3w_2x + (\hat{y}-c)w_2x + (\hat{y}-c)w_3x + (\hat{y}-c)x$$

skip W_3 skip skip both W_2, W_3

Which can prevent gradient vanishing.

For gradient explosion, please check

- He initialisation / Xavier initialisation
- gradient clipping.

3. Architecture of ResNet

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
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

a. For every convolution, add padding to retain the H×W.

b. Notice the output block of each conv_block will decrease ½, so we need to scale the figure at the very beginning of conv_block

✡ Except conv2_X, since it already exists max pool.

The reason is in traditional CNN, we like to use less resource to enhance computation efficiency. So at the very first of each layer, we will decrease the dimension while increase the channels.

4. Output dimension (Supplement)

The main param to control output dim is stride

Kernel size and padding are just adjusting or maintaining the output shape.

$$\text{output size} = \text{floor}\left(\frac{(\text{Input} + 2\times\text{padding} - \text{kernel})}{\text{stride}}\right) + 1$$

5. Deep Dive into Conv block.

$$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$$

- The first layer is used to decrease image shape, increase computational efficiency.
- 3×3 is used to extract image pattern.

- Final 1x1 is used to retain image recognition ability, add more channels to make sure the ability.

5. Some important tips.

a. We need to make sure the size of channel of x and $f(x)$ are same so that they can add together.

To downsample (modify) the size, we use 1 x 1 kernel to adjust.

b. There are two situation we need to modify the size.

① Channel

$$(B_x, C_x, H_x, W_x) \leftrightarrow (B_x, C_{f(x)}, H_x, W_x)$$

② Figure shape

$$(B_x, C_x, H_x, W_x) \longleftrightarrow (B_x, C_x, H_{f(x)}, W_{f(x)})$$

① will happen when ResNet are 50/101/152

If 18/34, then we don't need downsampling.

② Will happen at each layer, since every new conv-block we need to adjust both channel size and image shape.

C. How to decide padding?

For conv_1, input = 224, output = 112, stride = 2

$$112 = \text{floor}\left(\frac{224 + 2p - 7}{2}\right) + 1$$

$$\Rightarrow p = 3$$

For conv2 ~ conv5.

1 x 1 are used to decrease the shape $\Rightarrow$ $p = 0$

3 x 3 are pattern recognition, to maintain image shape $\Rightarrow$ $p = 1$