

## + Outline

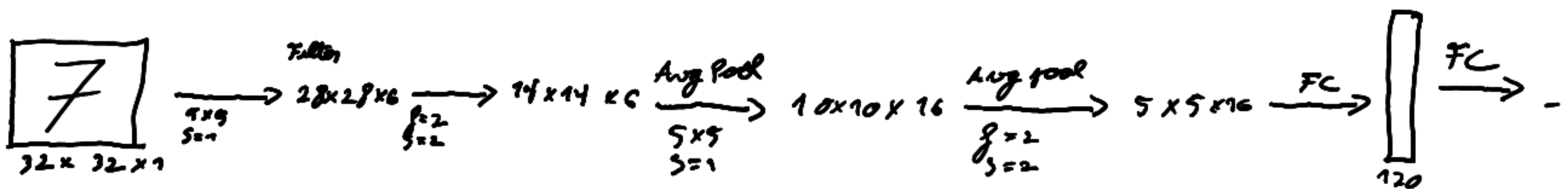
- Classic Networks: 1. Le Net-5  
2. Alex Net  
3. VGG

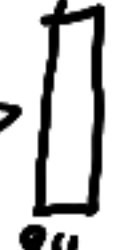
• Res Net

• Inception

## + Classic Networks:

1. Le Net-5  $\Rightarrow$  • Its goal was to recognize gray scaled images.
- Recognized hand-written digits
  - They didn't use padding.



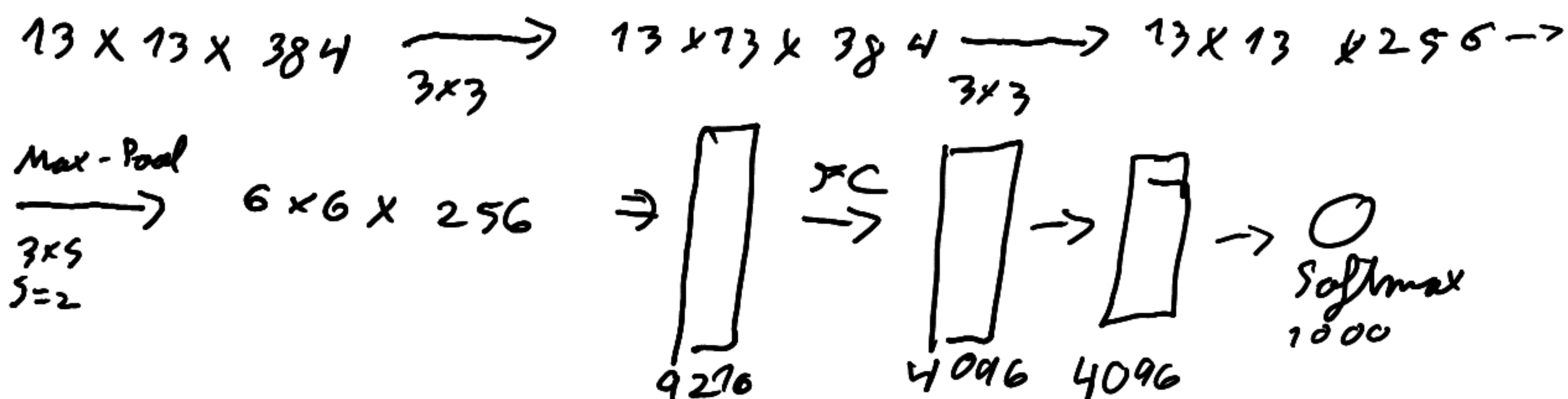
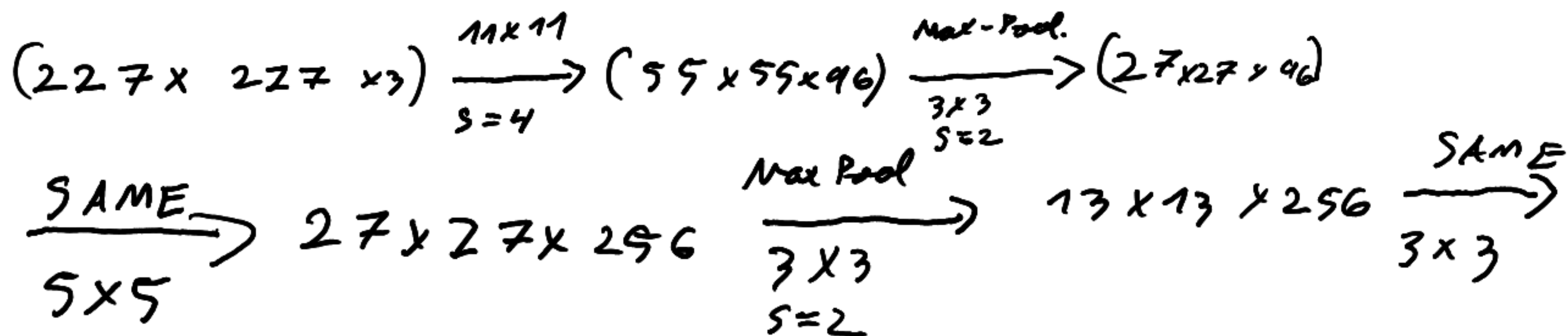
$\Rightarrow$    $\rightarrow 0 \rightarrow g$   $\hat{y}$  Took on 10 values to determine the digit

+ If the network was built now, it would have used a final "softmax"

+ It had 60k parameters

• Alex Net:

+  $227 \times 227 \times 3$  images



+ Classified images (1000)

+ About 60 million images

+ Used Relu activation function.

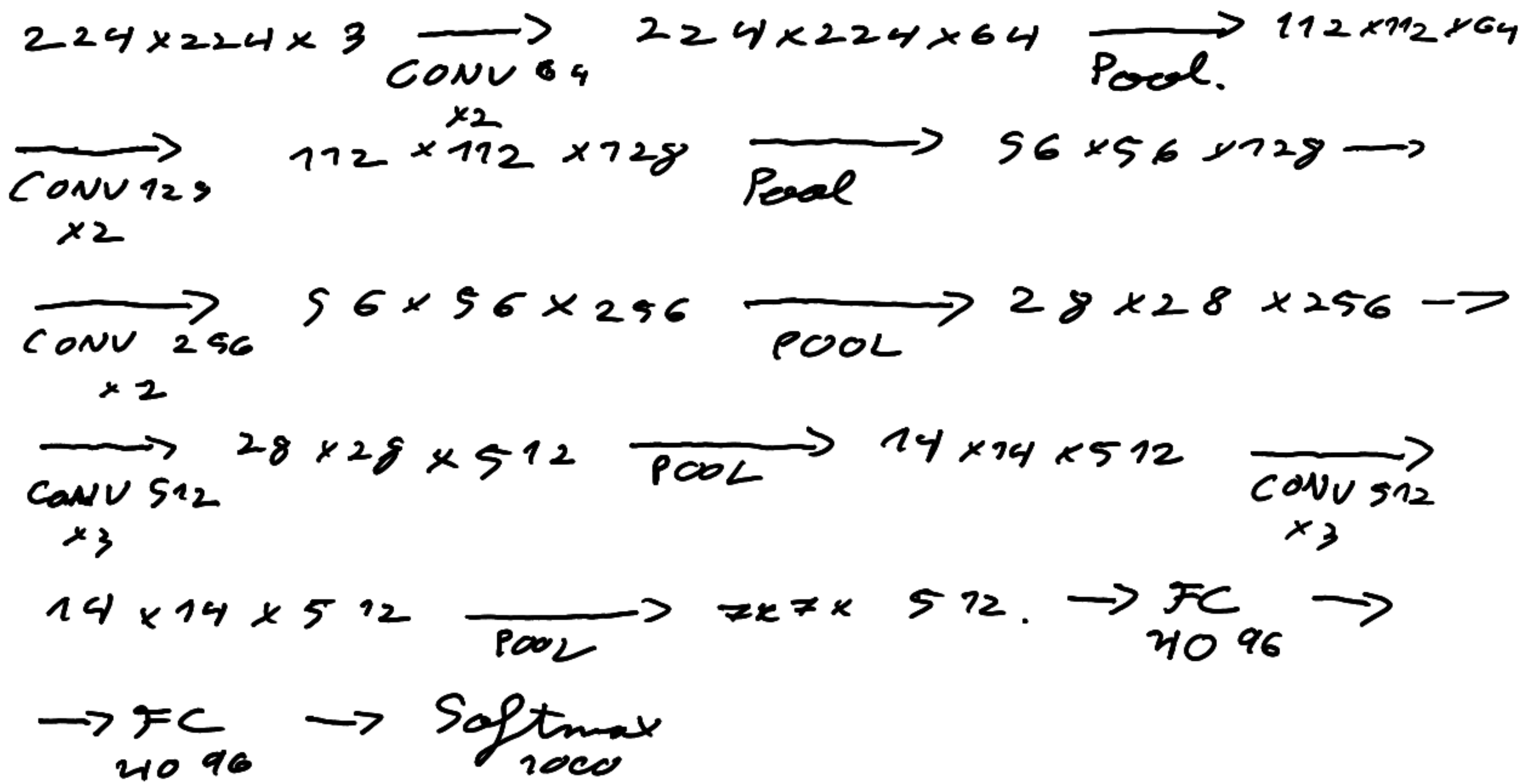
+ Local Response normalization  $\Leftrightarrow$  To prevent neurons with higher activation

$\Downarrow$   
Not useful

# VGG-16:

\* CONV =  $3 \times 3$  filter,  $S=1$ , same.

• Max-Pool =  $2 \times 2$ ,  $S=2$

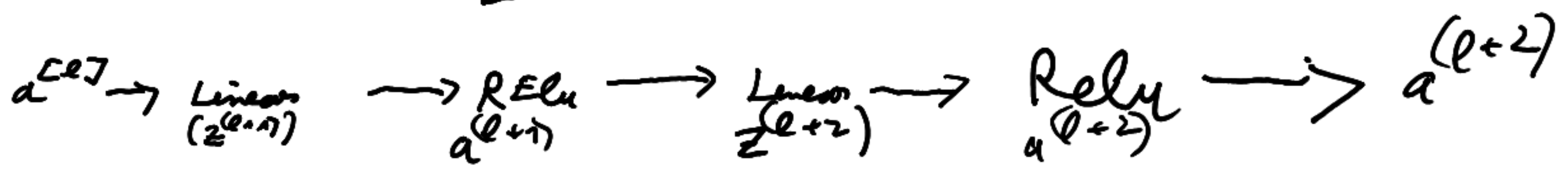
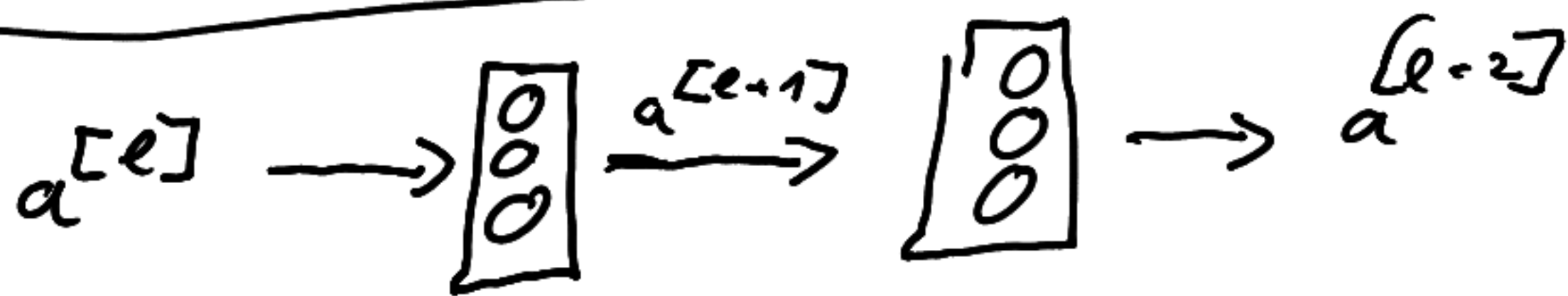


• Very large but simple due to repetitive layers.

• 16 layers.

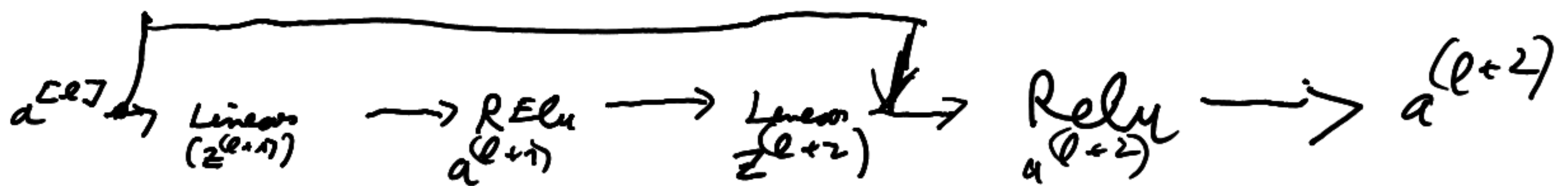
## + Res Nets (more complex)

- Enables you to train very deep Networks
- Skip connections allow you to take the activation from one layer and feed it to another.
- Residual block



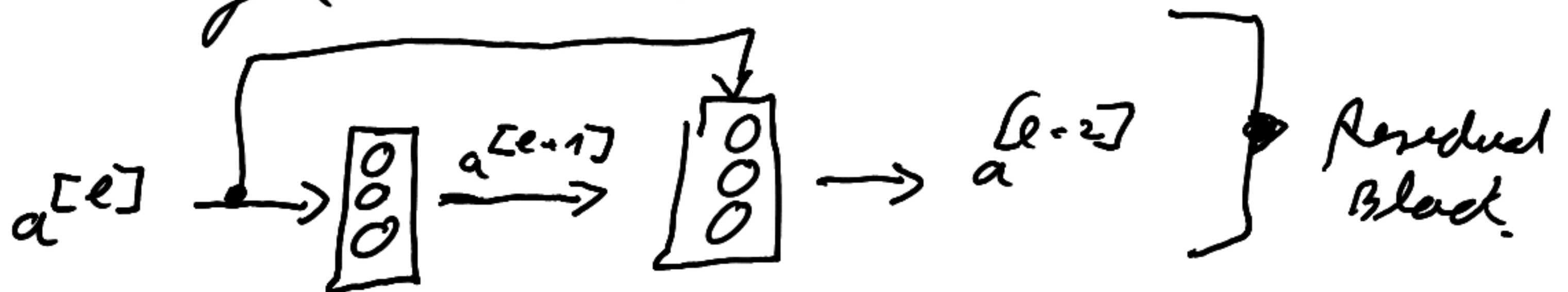
$$\left. \begin{aligned} z^{[l+1]} &= W^{[l+1]} a^{[l]} + b^{[l+1]} & a^{[l+1]} &= g(z^{[l+1]}) \\ z^{[l+2]} &= W^{[l+2]} a^{[l+1]} + b^{[l+2]} & a^{[l+2]} &= g(z^{[l+2]}) \end{aligned} \right] \rightarrow \text{Main Path.}$$

In a residual net: (shortcut/skip connection)



Now the output when taking the shortcut.

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




- To build a deeper network we would stack the skip connections.
- It helps with gradient descent problems.

## + Why do Re Nets work so well:

- We need to do well on The Training set.
- When we make The network is too deep The network ends up struggling to learn more parameters.
- The reason The Re nets work, is that it helps performance.

## + Networks in Networks and 1x1 Convolutions

- V, pretty much multiply if only 1 dimension.

$$6 \times 6 \times 32 \rightarrow 1 \times 1 \times 32 = \text{Relu.}$$


- Basically having a fully connected network, that is applied to each of the positions.



## + The Inception Network Motivation:

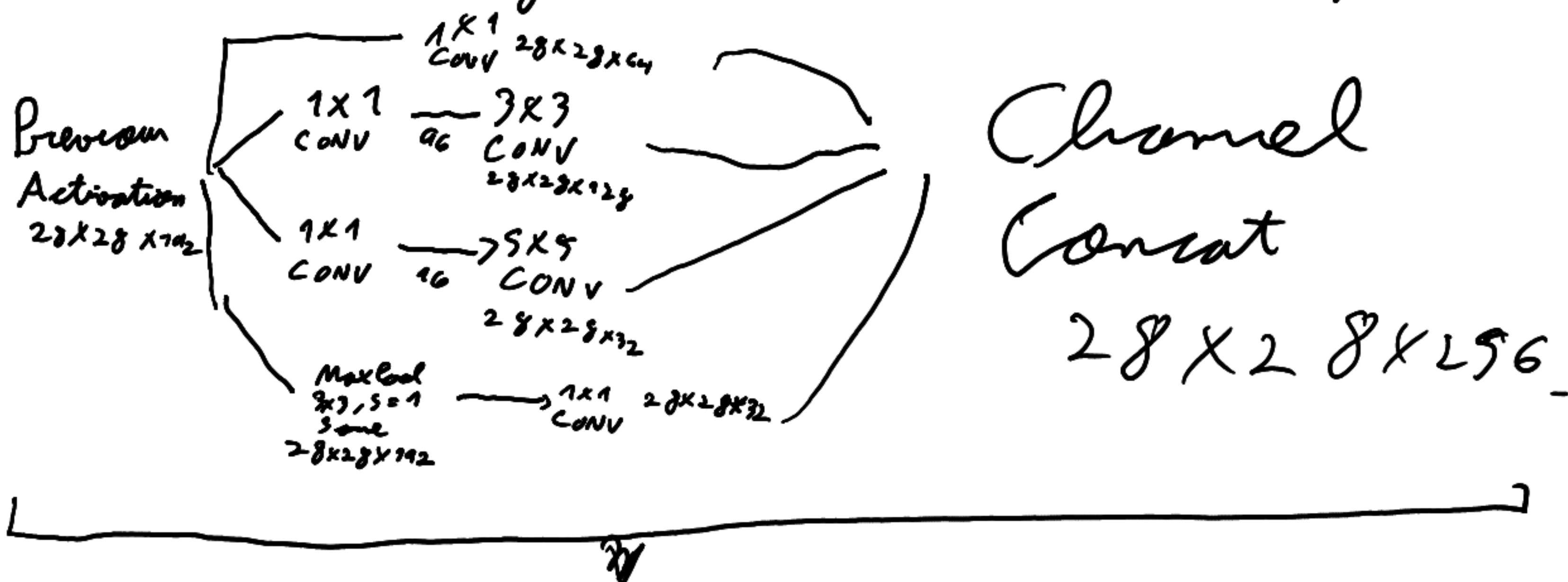
An inception layer, pretty much does all, filters, pooling, ...

With this concept, we input a module, and output all of them ( $1 \times 1$ ,  $3 \times 3$ ,  $5 \times 5$ , max-pool)

- The idea is to concatenate the parameters.
- One problem, is the computation cost.

## + Inception Network

- We use a  $1 \times 1$  filter to help with comp cost.



1 inception Module.

- The inception network puts together many blocks.

## + Mobile Net.

- Network To run in less powerful devices.
- Key idea: Normal vs depthwise-separable convolutions.
- Computational cost of a normal network  
Comp cost = # filter params  $\times$  # filter positions  $\times$  # of filters
- Depthwise separable convolutions.

$$n \times n \times m_c \quad * \quad \underbrace{f \times f}_{m_c \text{ filters}} \quad = \quad n_{out} \times n_{out} \times m_c$$

"one filter per layer."

- Depthwise cost:

$$\text{Computational cost} = \# \text{ filter params} \times \# \text{ filter positions} \times \# \text{ of filters}$$

- Pointwise Convolution

$$n_{out} \times n_{out} \times m_c \quad * \quad \underbrace{1 \times 1 \times m_c'}_{m_c' \text{ filters}} \quad = \quad n_{out} \times n_{out} \times m_c'$$

Full Depthwise separable Convolution

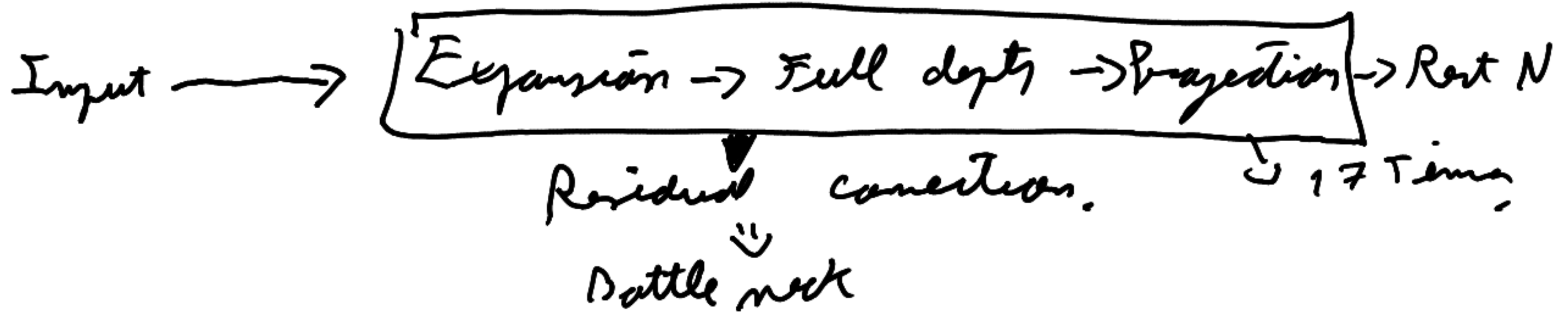
$$\text{Input} * \text{Depthwise} * \text{Pointwise} = \text{Output}$$

## + Mobile net architecture

### • Mobile Net v1



### • Mobile Net v2



## + Efficient Net:

Scaling Networks for different devices.

Resolution / Depth width

(compound scaling)