(VGG-16/19 contd.)

Why is there a smaller number of parameters?

— 5×5 filter: $5\times5+1$(bias) $= 26$ parameters

2 layers of $3\times3 \rightarrow 2(3\times3+1) = 2\times10 = 20$ parameters

— 7×7 filter: $7\times7+1$(bias) $= 50$ parameters

3 layers of $3\times3 \rightarrow 3(3\times3+1) = 3\times10 = 30$ parameters

→ enough to cover the same pixel area / lattice

— 11×11 filter: $11\times11+1$(bias) $= 122$ parameters

5 layers of $3\times3 \rightarrow 5(3\times3+1) = 5\times10 = 50$ parameters

→ enough to cover the same pixel area / lattice

However, cannot go deeper than 16/19 layers, as issues cropped up with gradients, which would only be solved later, with residual connections in the ResNet family (ResNet: Microsoft Research)

Physical Significance: cost-effectiveness: is in terms of the number of parameters of the convolution (# of weights to be tuned, to avoid overfitting issues) The number of multiplications/additions/operations is not an important parameter for this application (we had also considered computing this as well for all layers of LeNet for instance: operation count)

Next up: ResNet (we have already covered the main points)

Inception Architecture (Google): 2015 variants

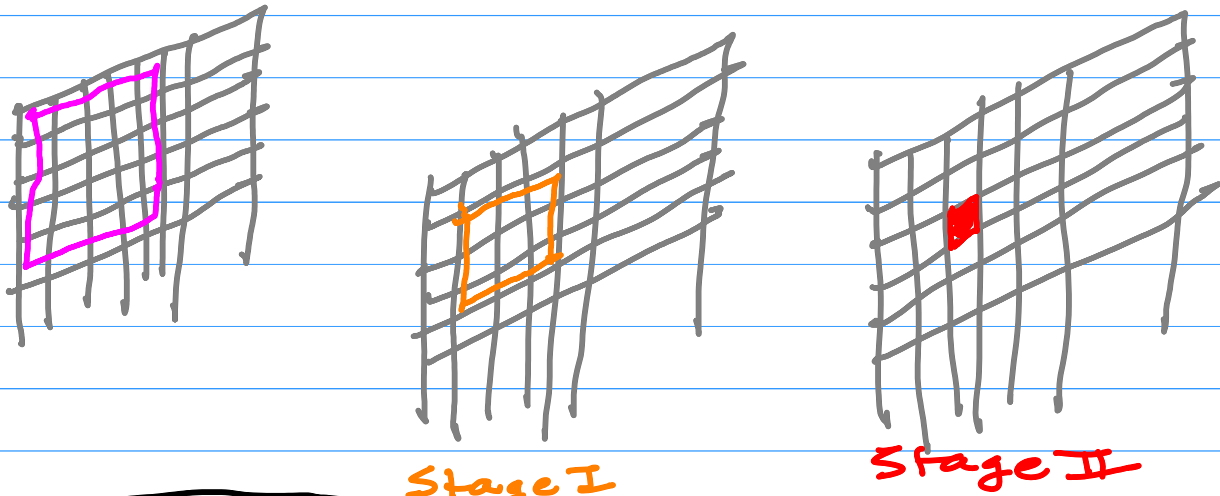
GoogLeNet, Batch Normalisation, Factorising Convolutions

we will see what this is

we have seen this before

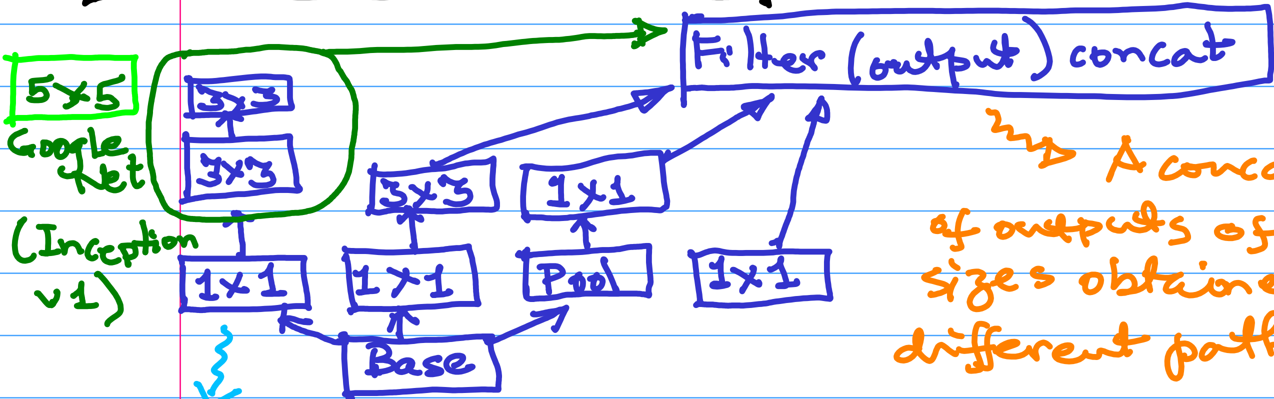
we have seen the basic philosophy in VGG-16/19

Factorising Convolutions:



Hollywood Movie "Inception": "dream within a dream"

"Inception Module A"



(To be discussed later)

A concatenation of outputs of different sizes obtained along different pathways → concept of scales

This is a bit like looking at a signal across scales. (Gaussian pyramid): a concatenation of the outputs of the features in a signal at all possible scales → this information can be put to use later.

128-length feature detector output at a pixel which includes information at various scales

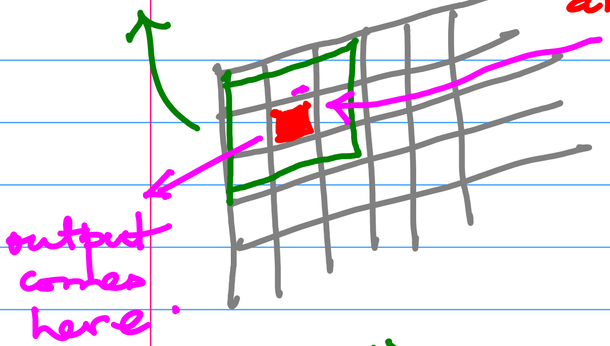
Handcrafted feature detectors: SIFT / SURF

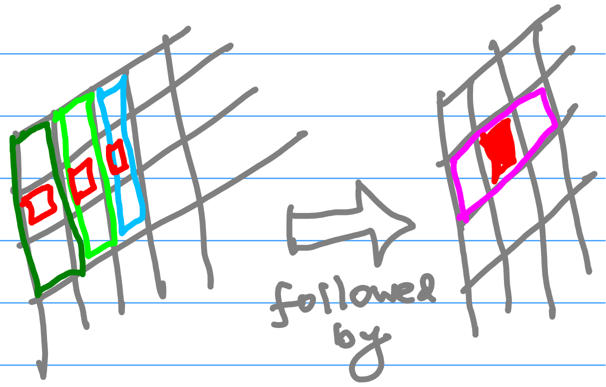
Scale Invariant Feature Transform
(has information at various scales, which in turn leads to scale-invariance)

$\boxed{\text{Inception Module B}}$ Asymmetric Convolutions

One 3x3 convolution is replaced with two stages three 3x1 convolutions followed by one 1x3, or vice versa

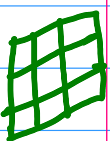
3x3 filter gives as its output at this position in the next level





three 3x1 convolutions

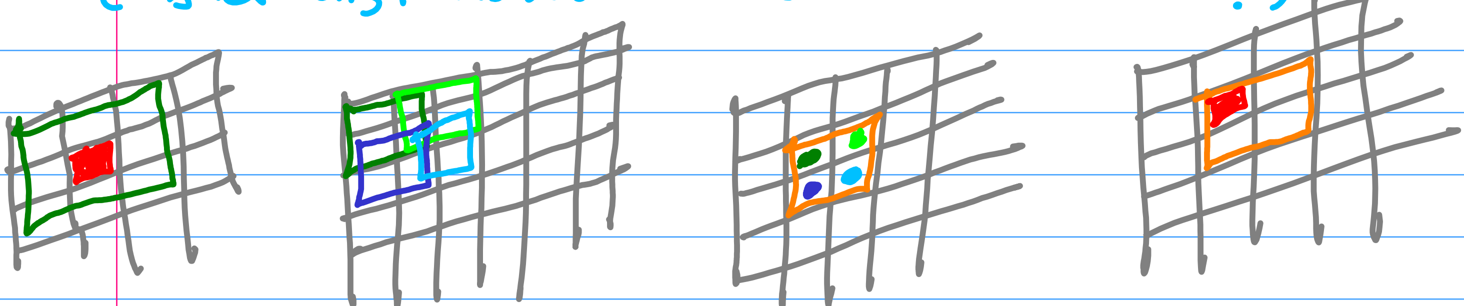
one 1x3



$(3\times3 + 1) = 10$ parameter cost

$$\underbrace{(3+1) + (3+1)}_{=8 \text{ (less)}}$$

Question: Why don't we use two 2x2 filters filter layers to replace one 3x3?
(after all, this was the basic VGG idea!)



parameter count

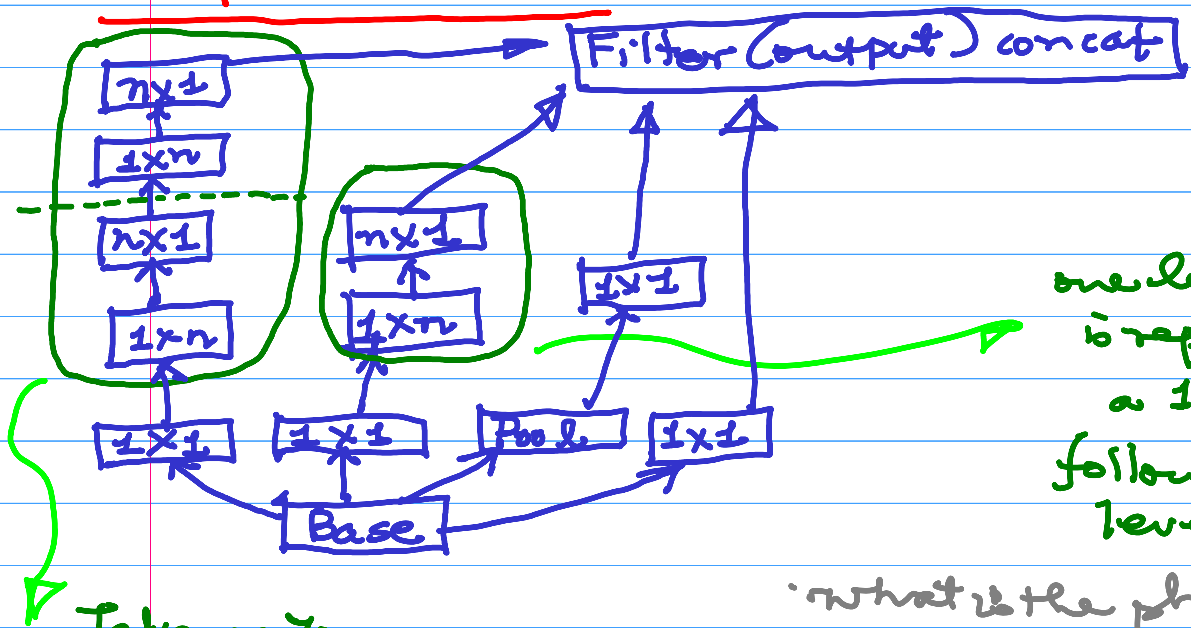
$3\times3+1=10$

parameter count

$= 2\times2+1$ $\quad$ $2\times2+1$

the total parameter count hasn't changed!

Inception Module B



one level of 7x7 is replaced with a 1x7 level followed by a 7x1 level

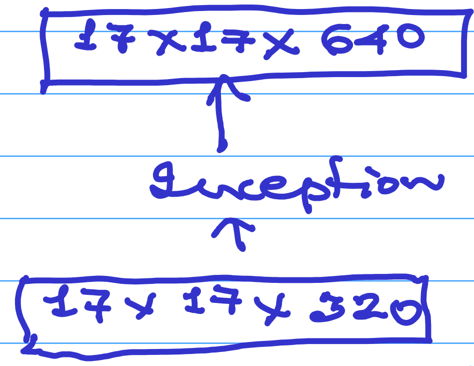
Take n=7

Two levels of 7x7: each level of 7x7 is replaced with a 1x7 level followed by a 7x1 level

· what is the physical significance of the concatenated filter output? The output size is often larger (in cases, double as well)

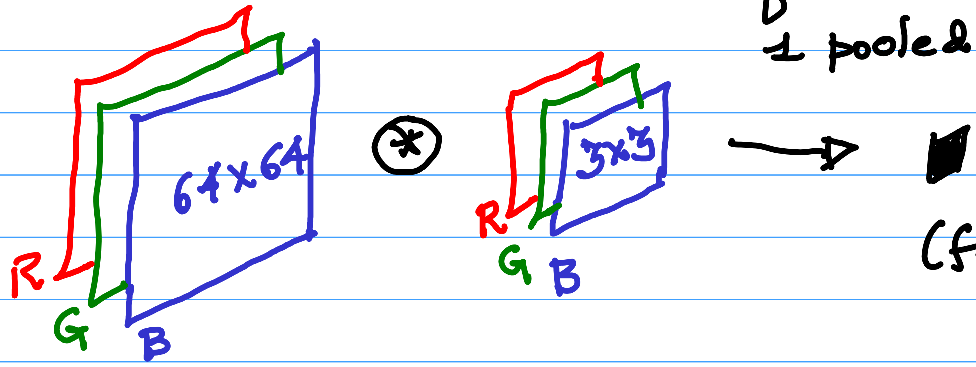
→ features from all scales

e.g.,

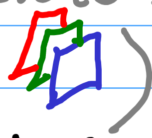


The output image size is the same as the input image size but has a richer representation in terms of the number of channels.

What is a 1x1 convolution? seems to be pointless (an image convolved with one value: plus a bias parameter)

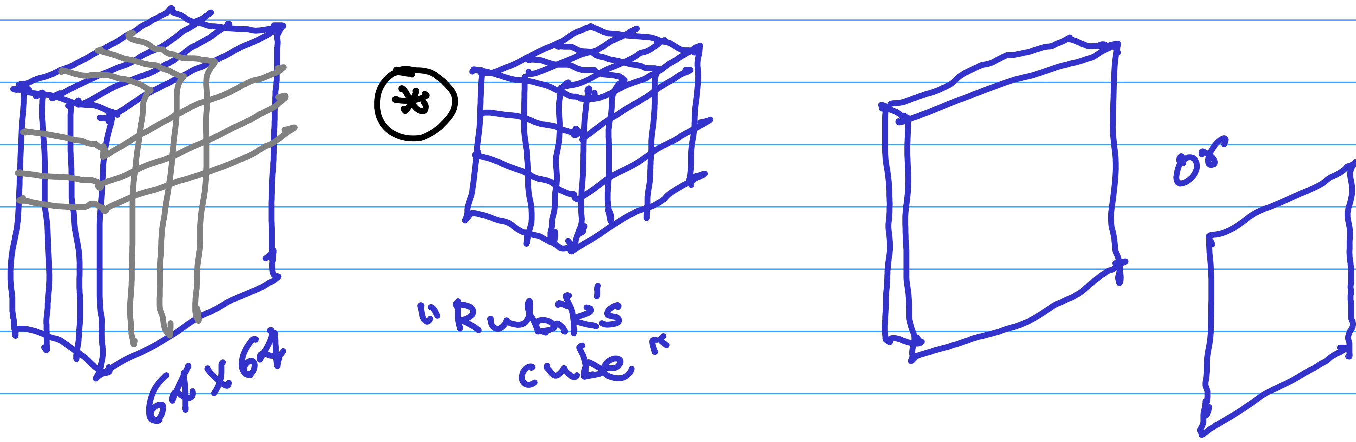


if we want to create 1 pooled output (in place of

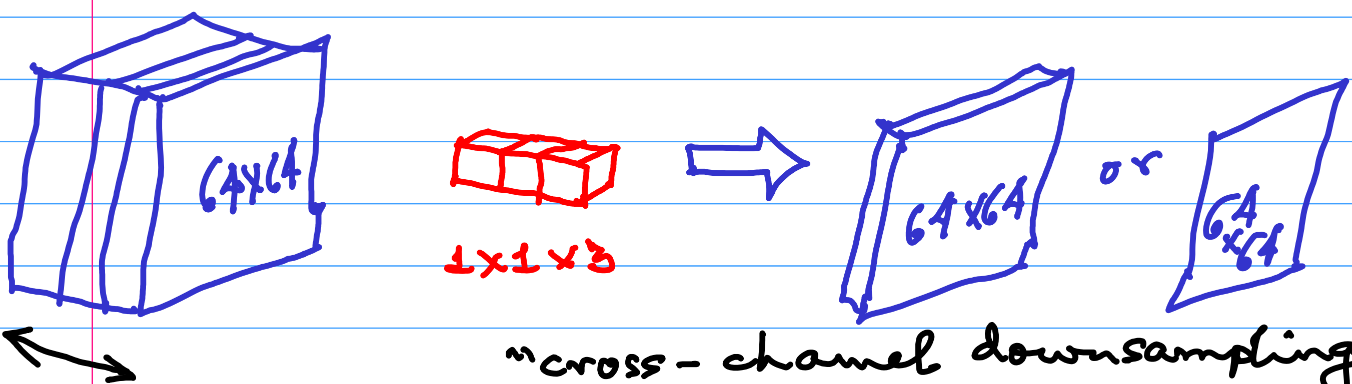


)

(for each pixel position)



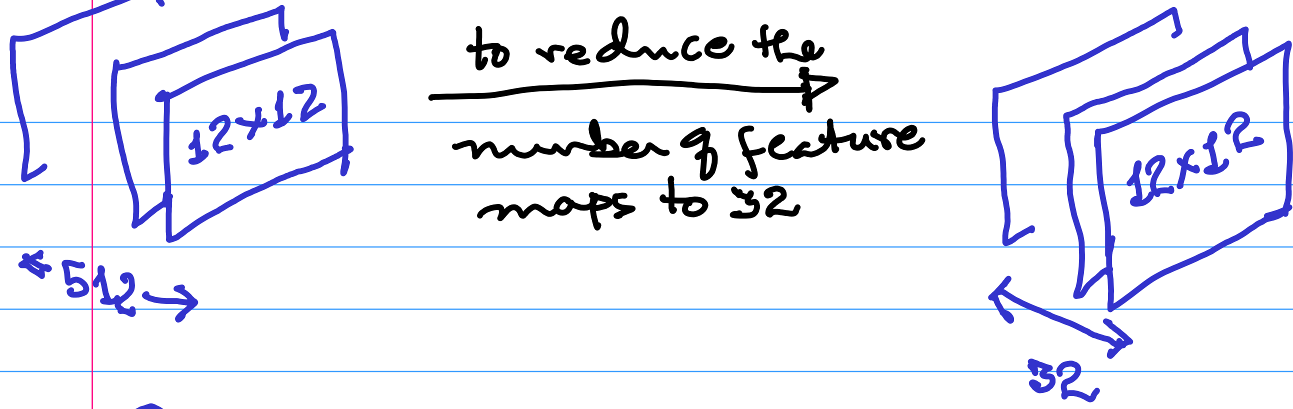
64x64x3 (RGB)

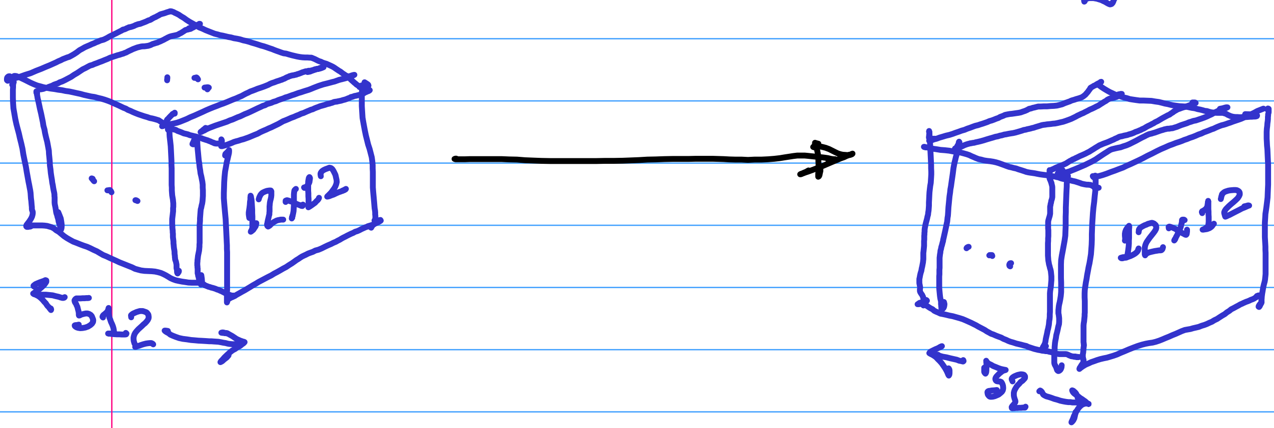


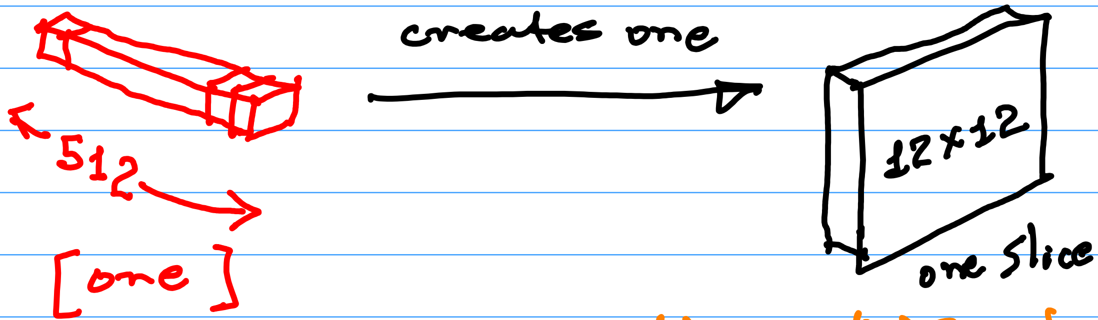
"cross-channel downsampling"

"cross-channel pooling"

Another example



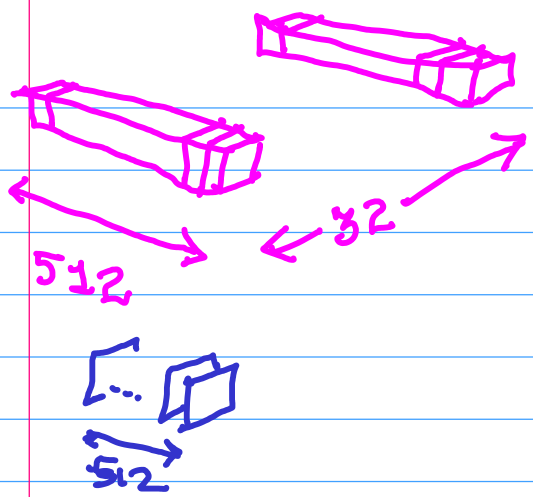




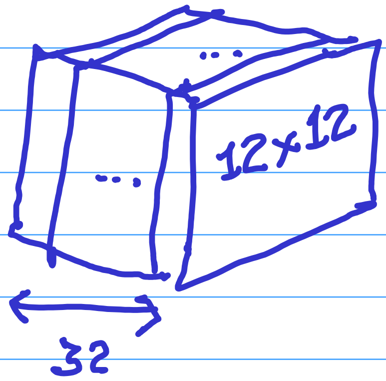
Physical significance:
Basically have 512 weights for each pixel position in the 12×12 image

now, we take 31 other such "1×1" entities, each with 512 layers
→ each of these has different weight combinations for the 512 constituent channels

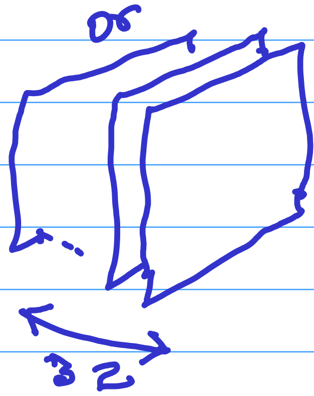
We want 32 such slices
(⇒ 32 such filters)
The resultant 12×12 image has the weighted sum at each pixel positions of the constituent 512 pixel values at each pixel position



$\Rightarrow$ stack the 12×12 slices: all 32 of them, together



or

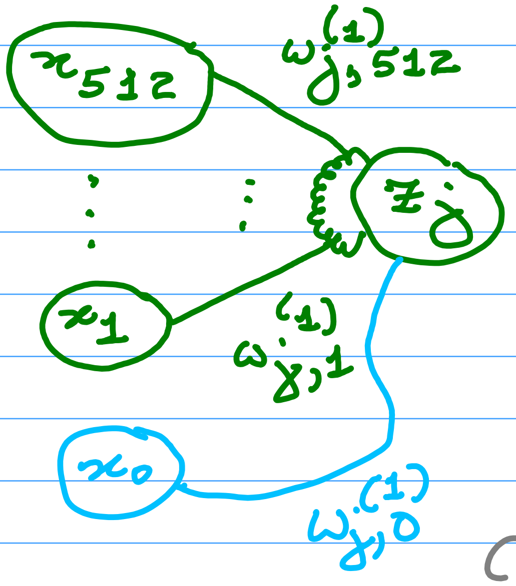


parameters?

$$32 \times (\underbrace{512 \times 1 \times 1}_{} + \overbrace{1}^{\text{bias}})$$

other option: convolutionally reduce it $\rightarrow$ more parameters.

How to introduce non-linearities?



By default, activation

$$a_j = \sum_{i=1}^{512} x_i \cdot \omega^{(1)}_{j,i} + \omega^{(1)}_{j,0}$$

Now, we can apply

$h(a_j)$ $\rightarrow$ activation function (possibly non-linear)

e.g., ReLU or tanh or sigmoid