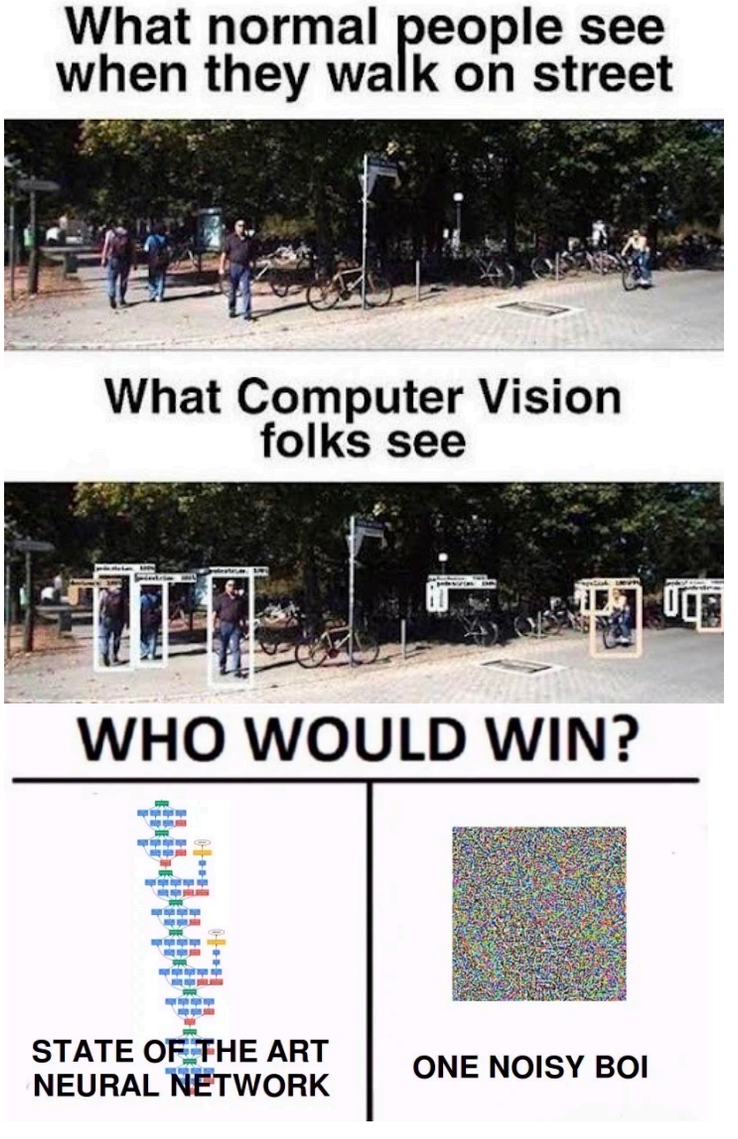
March 29, 2023

DSML: Computer Vision.

Image Similarity: Understanding Embeddings

Class starts @ 9:05 pm.

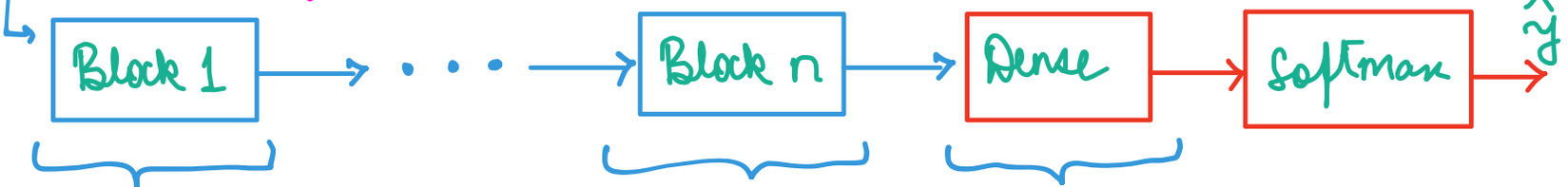


Recap:

* Problem: Classification.
* Approach: (a) Preprocess Image Data.
  (b) Design an architecture.
  (c) Train the architecture.
  (d) Enjoy the awesome results given by the CNN.

* Architecture:
  (a) AlexNet – CNNs with ReLU on a GPU.
  (b) VGG – Convolution blocks with 3×3 convolutions.

  Size of the model ≈ 100s of MBs.



→ Conv2D, RELU, Maxpool, Batchnorm

Most parameters are here.

Agenda:

* Study 2 new classification architectures:
  - → Inception.
  - → ResNet.

* Use a trained classifier for something else!!

REVERSE IMAGE SEARCH:

Given an image, find other images which look like this one.

Google Lens

Pinterest

Celebs who look like me.

Inception Network.

→ Google → ImageNet 2014 challenge.

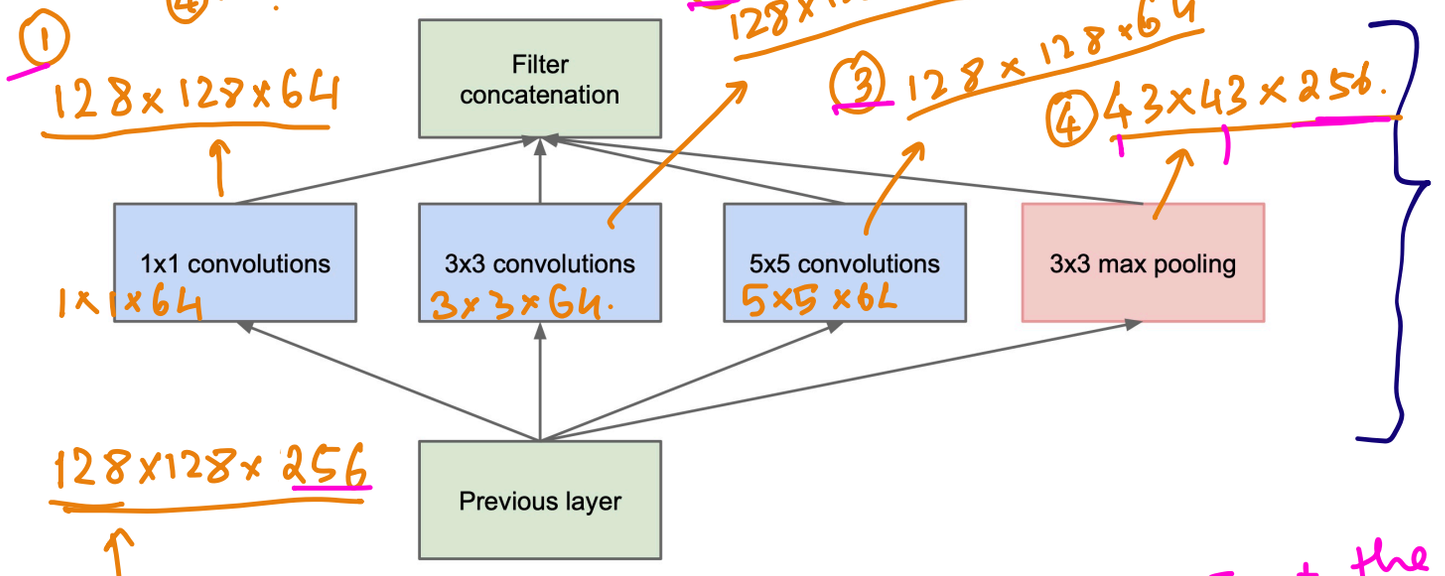
→ This network did slightly better than the VGG-19 network.

→ It used only 11 M parameters while VGG ~130M parameters.

128x128x48

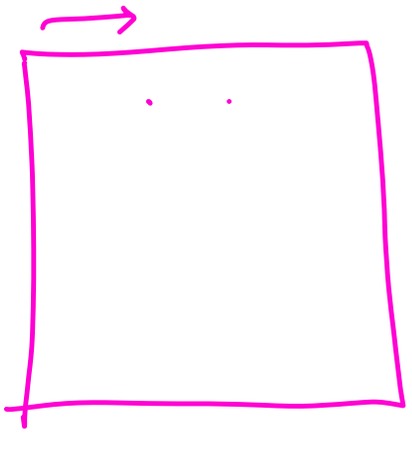
128x128x192.

④ ? Dilate + concat.



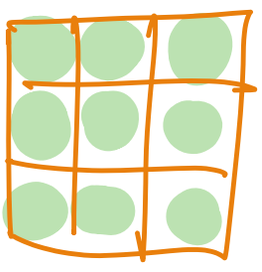
Concatenation will only affect depth.

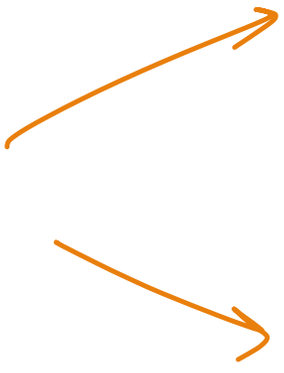
(a) Inception module, naïve version → Just the basic idea.

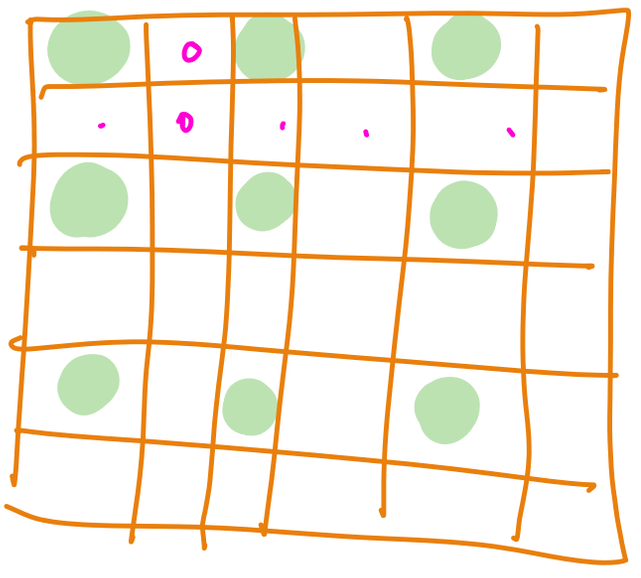


3×3

$$\left\lceil \frac{128}{3} \right\rceil \quad \lceil 42.66 \rceil = 43 \times 43$$

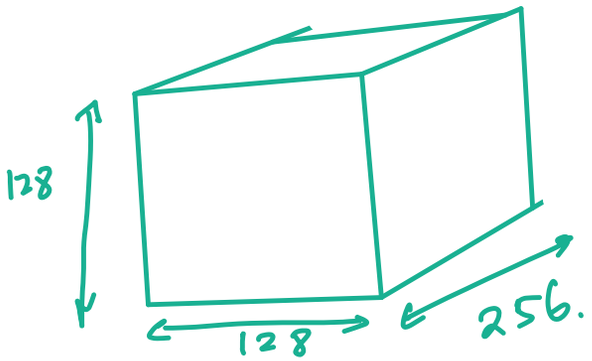




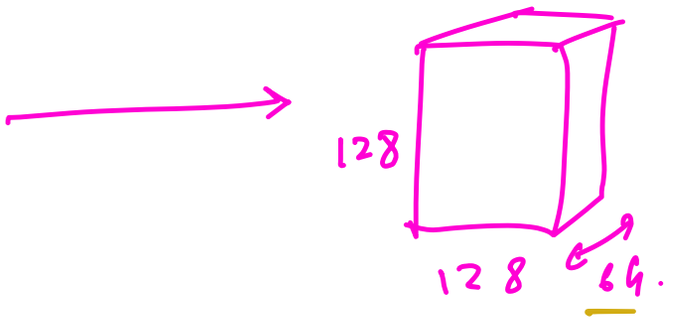


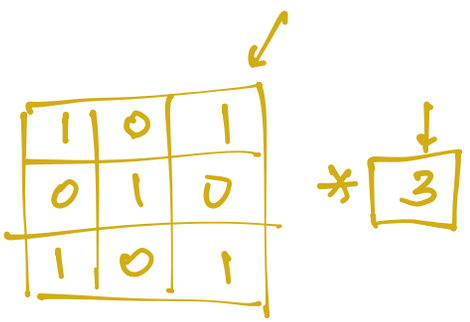
What are 1 × 1 convolutions?

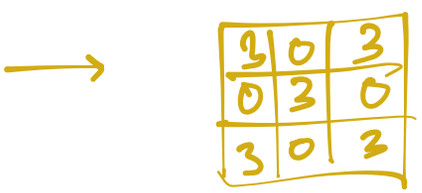
→ 1×1 convolutions help us reduce the filter depth information loss is minimized.

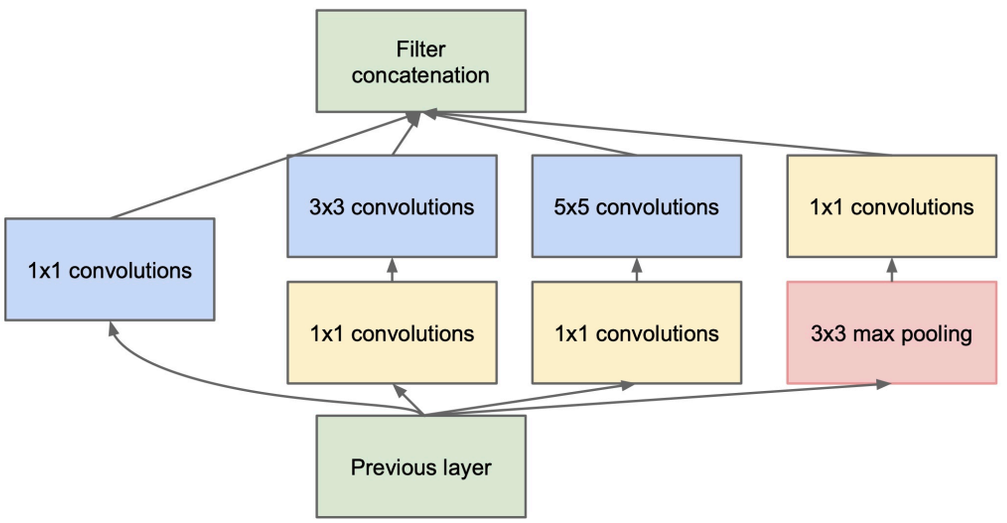


128 X 128 X 256

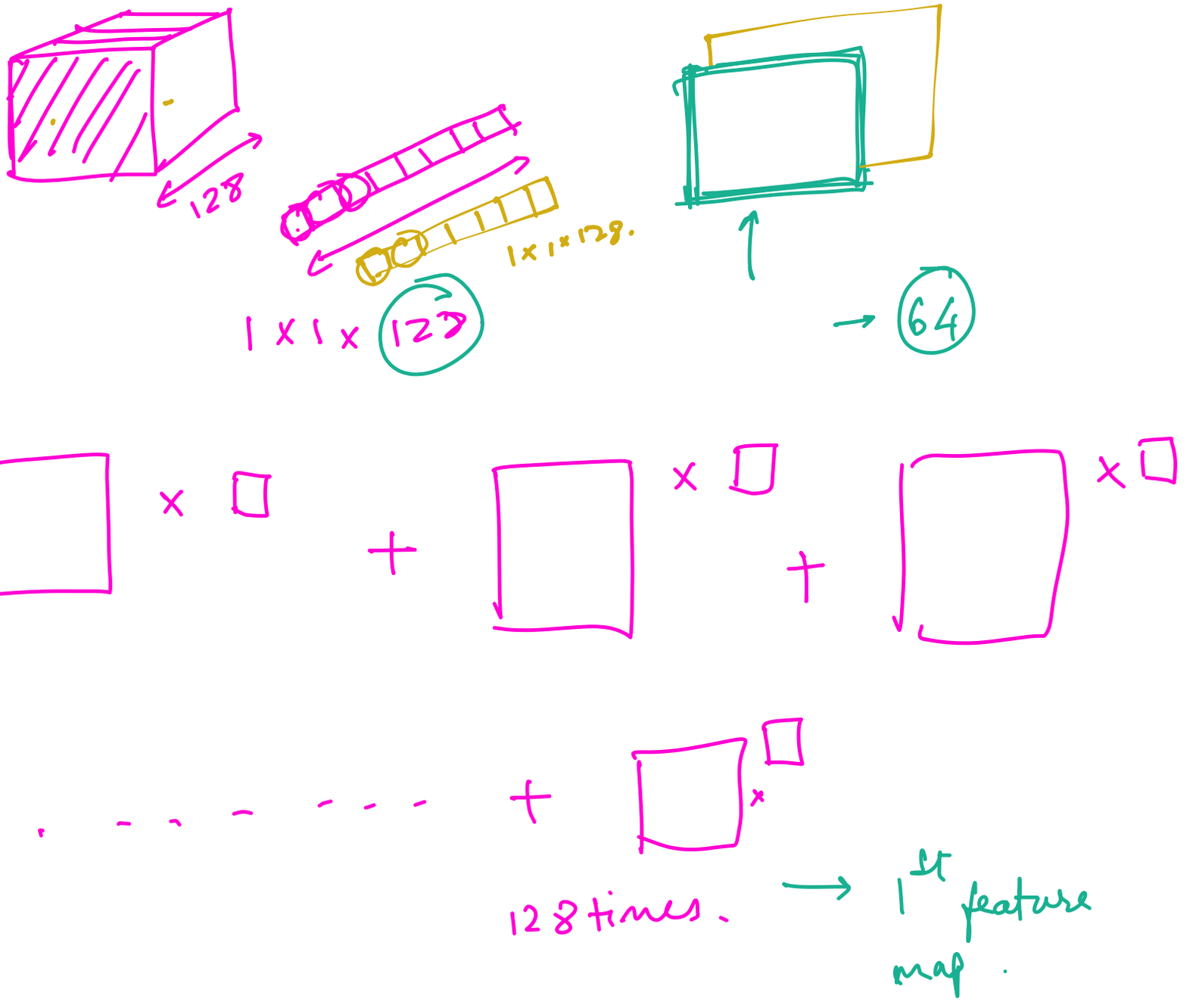


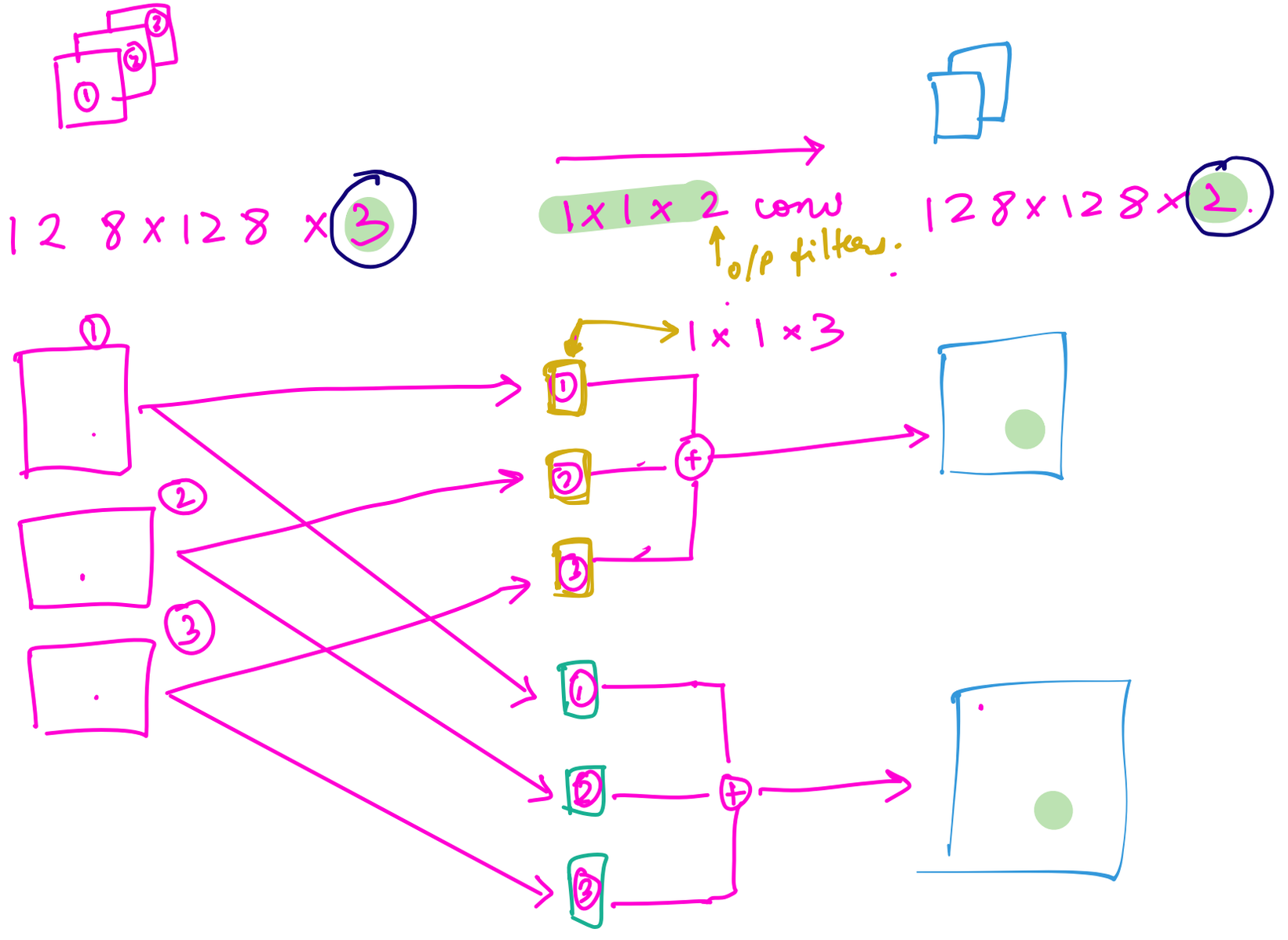




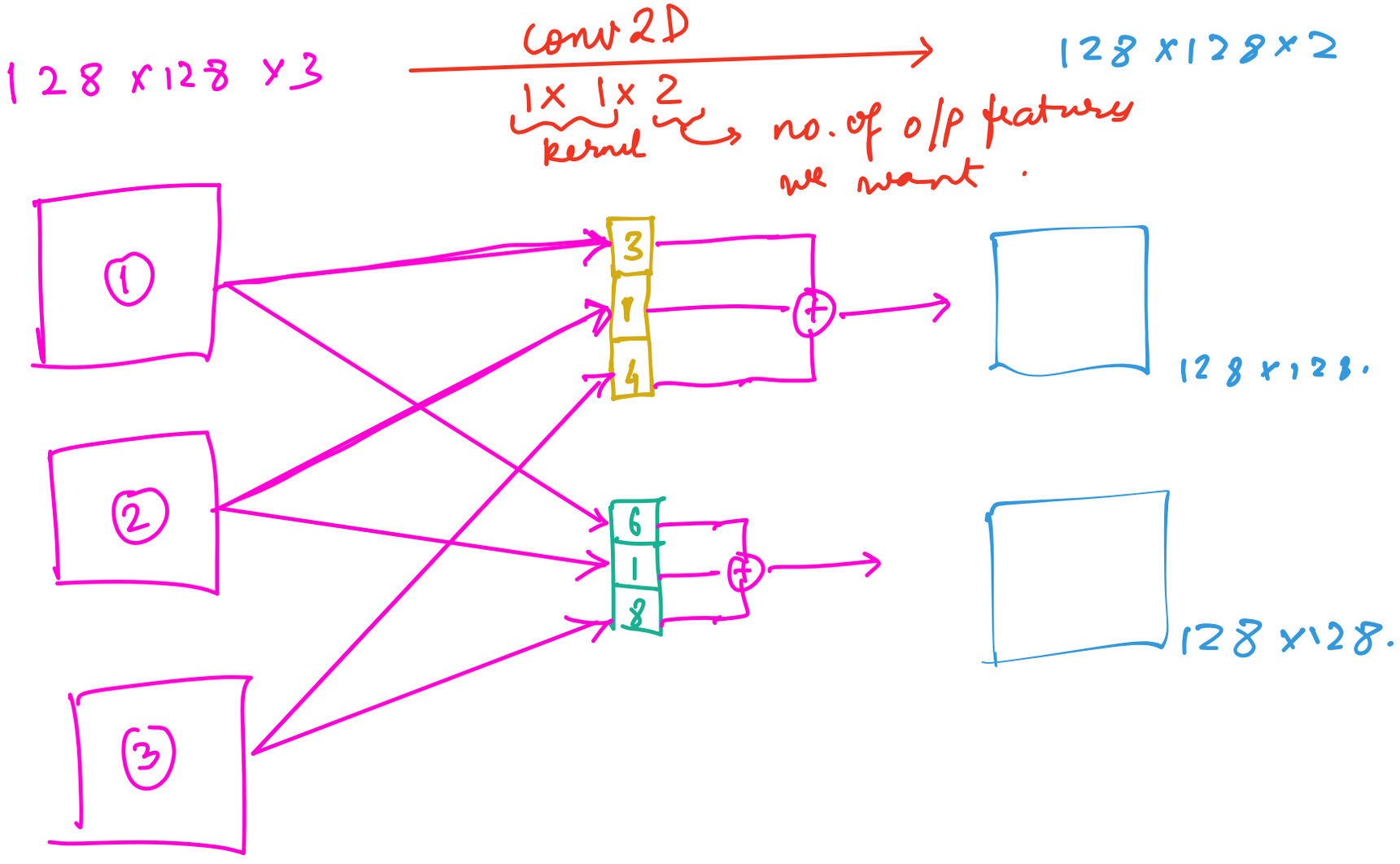


(b) Inception module with dimension reductions


128
1 x 1 x 128.
1 x 1 x 128
64
x
+
x
+
x
+
128 times.
1st feature map.



1 x 1 convolutions are nothing but weighted averages!!


128 x 128 x 3
Conv 2D
1 x 1 x 2
kernel
no. of o/p features we want.
128 x 128 x 2
1
2
3
3
1
4
6
1
8
128 x 128.
128 x 128.

Q] Why do 1×1 convolutions when the same reduction can be done with a 3×3 kernel?

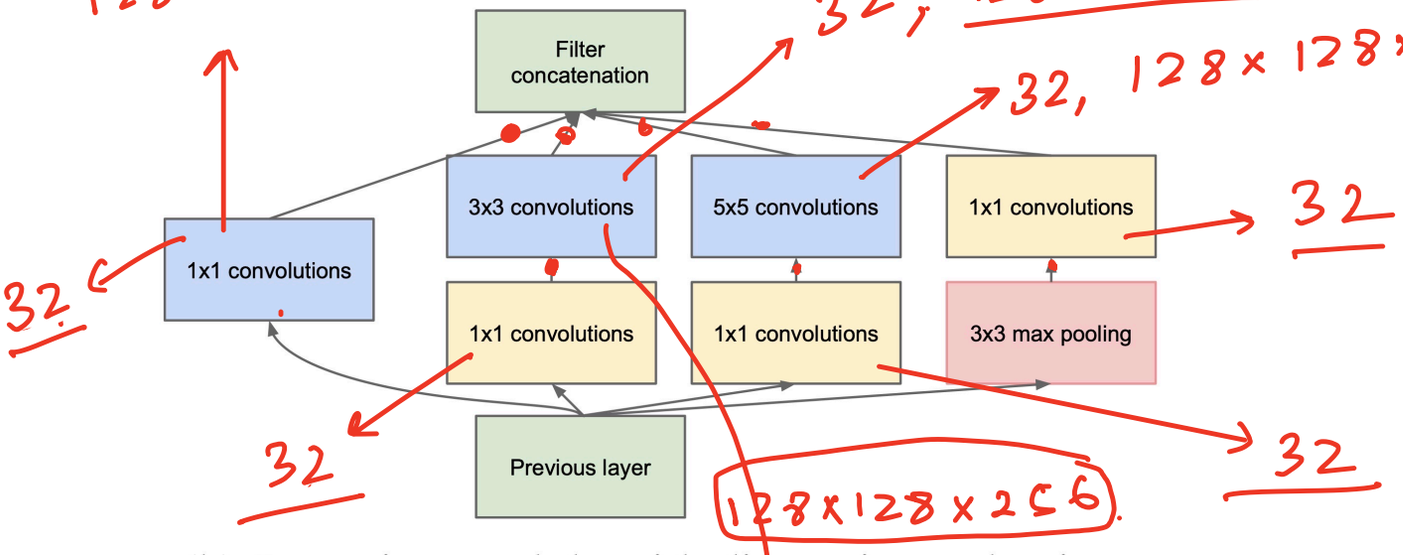
128×128

Input : m channels
Output : n channels.

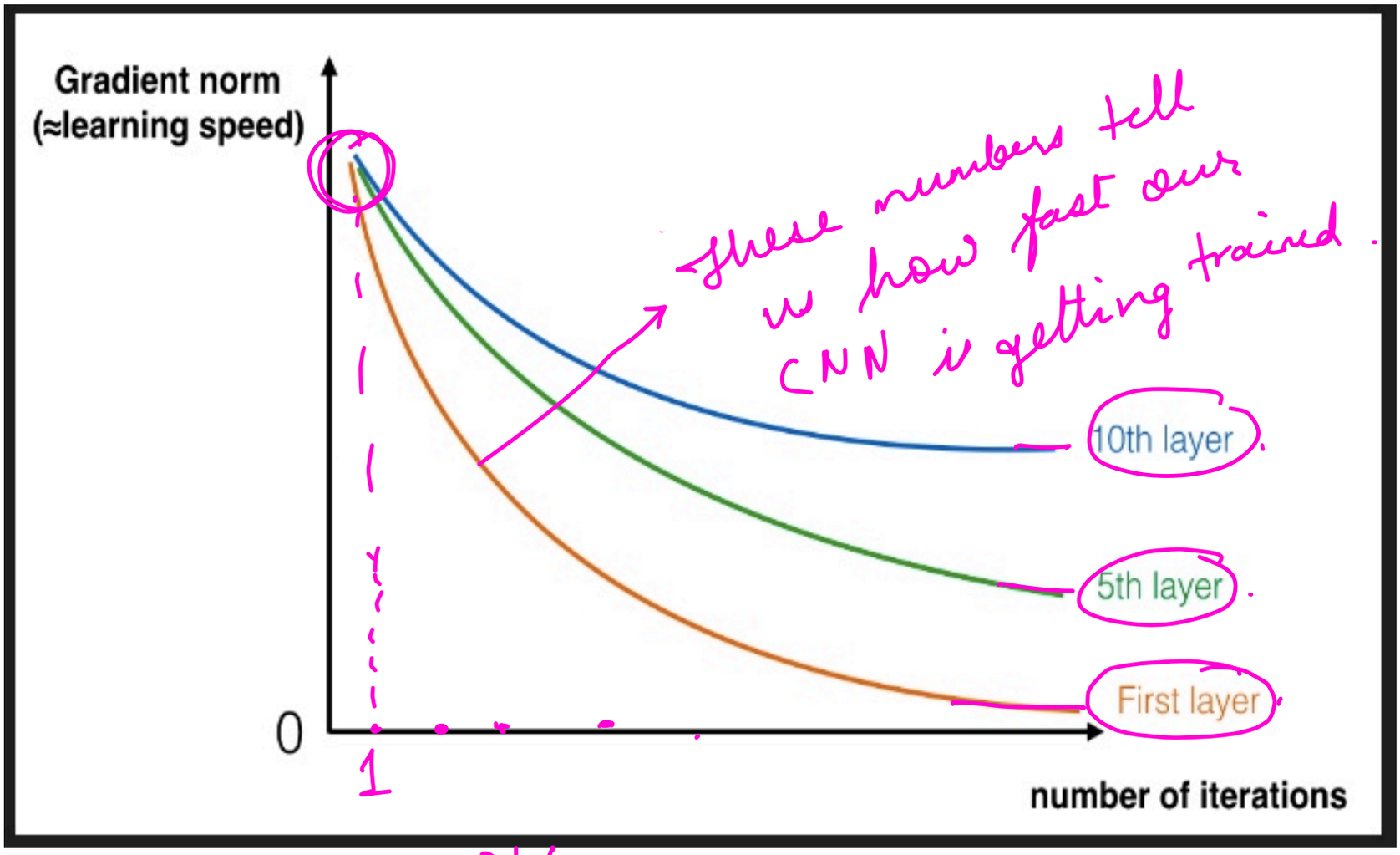
	3×3	1×1
No. of parameters	$9mn$	mn
No. of mul + add operations.	$128 \times 128 \times 9mn$	$128 \times 128 \times mn$

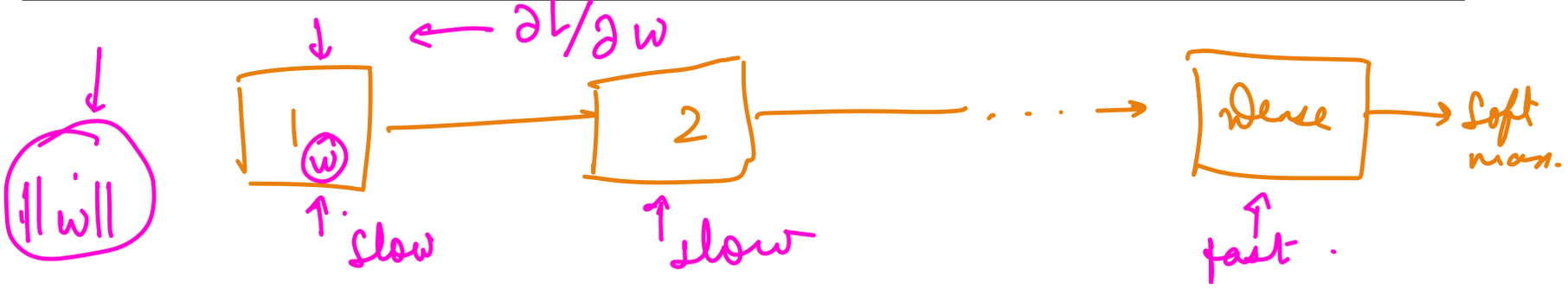
↓

Better space, time complexity

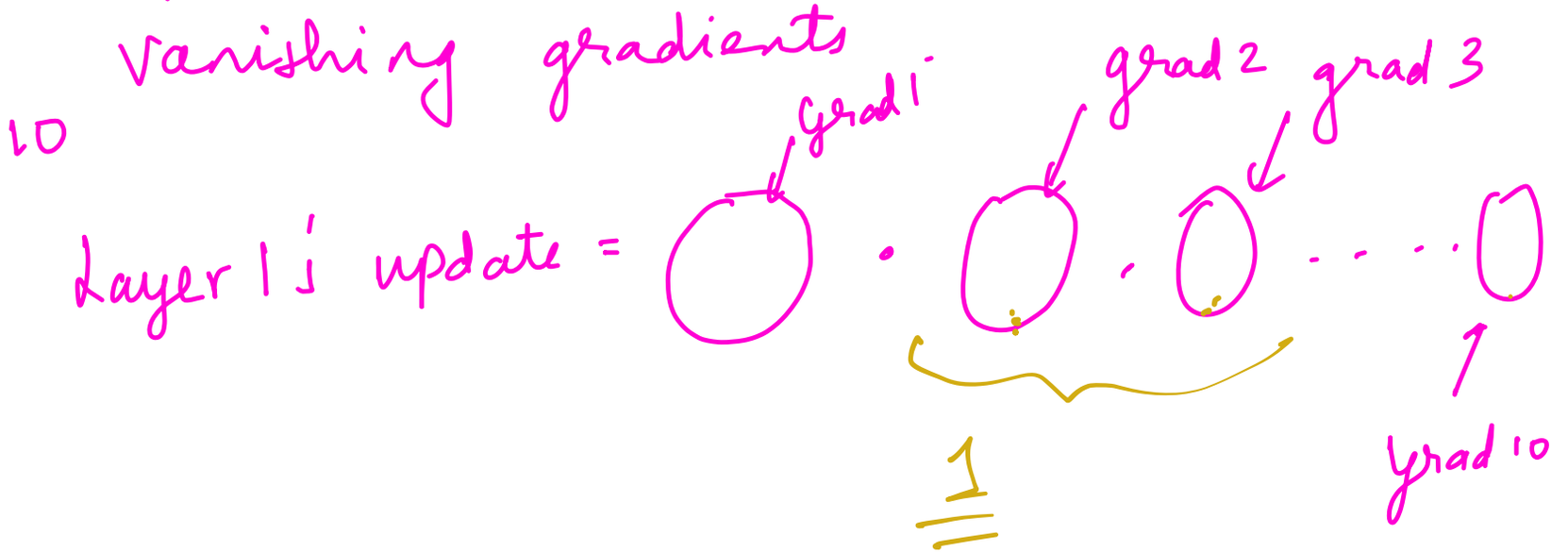


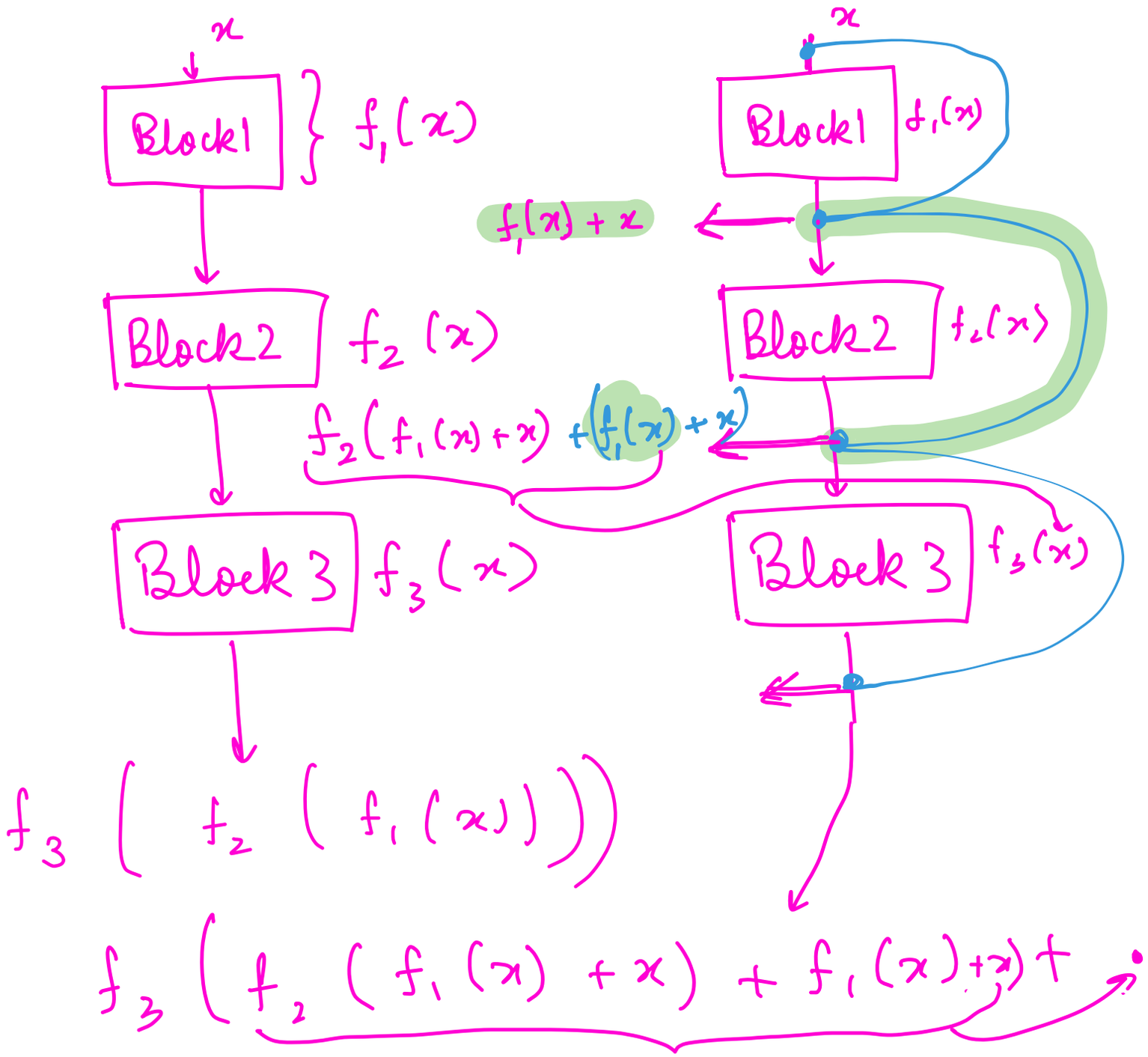
(b) Inception module with dimension reductions


Gradient norm
(≈learning speed)
these numbers tell us how fast our CNN is getting trained.
10th layer
5th layer
First layer
0
1
number of iterations

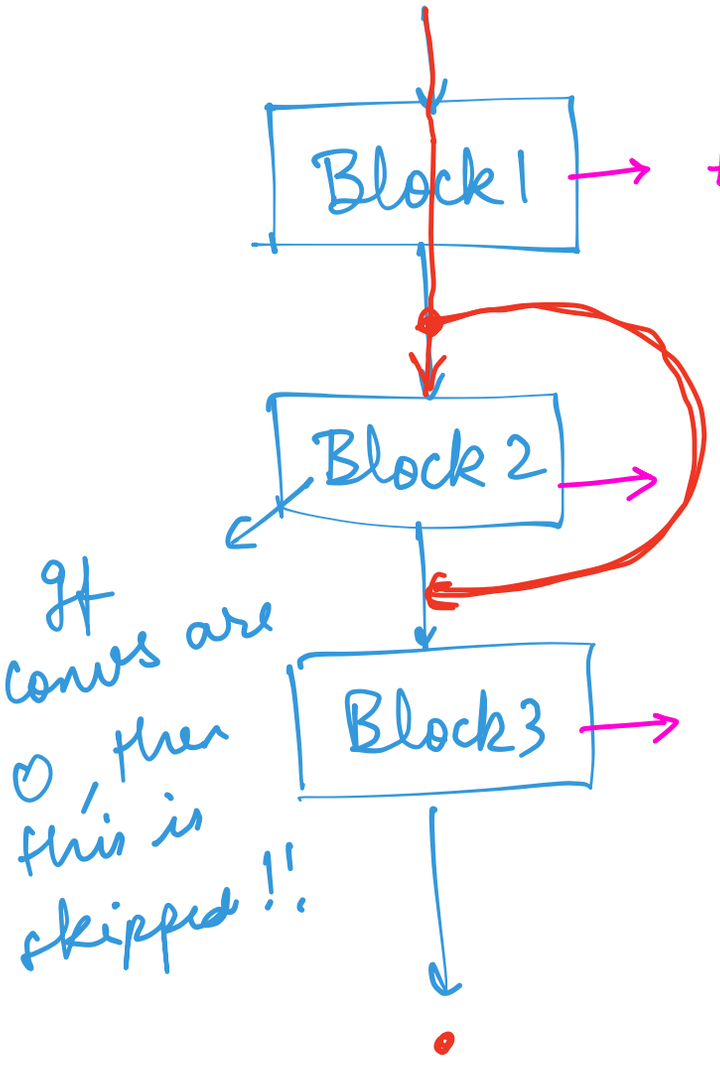

$\leftarrow \partial L/\partial w$
||w||
1
w
2
Dense
Soft max.
slow
slow
fast.

The problem we see is because of vanishing gradients




x
Block1
$f_1(x)$
Block2
$f_2(x)$
Block3
$f_3(x)$
$f_3\left(f_2\left(f_1(x)\right)\right)$
x
Block1
$f_1(x)$
$f_1(x)+x$
Block2
$f_2(x)$
$f_2(f_1(x)+x)+(f_1(x)+x)$
Block3
$f_3(x)$
$f_3\left(f_2(f_1(x)+x)+f_1(x)+x\right)+\cdots$

Forward pass:



Block 1 → + 3% increase in accuracy

Block 2 → + 0.05% accuracy

X

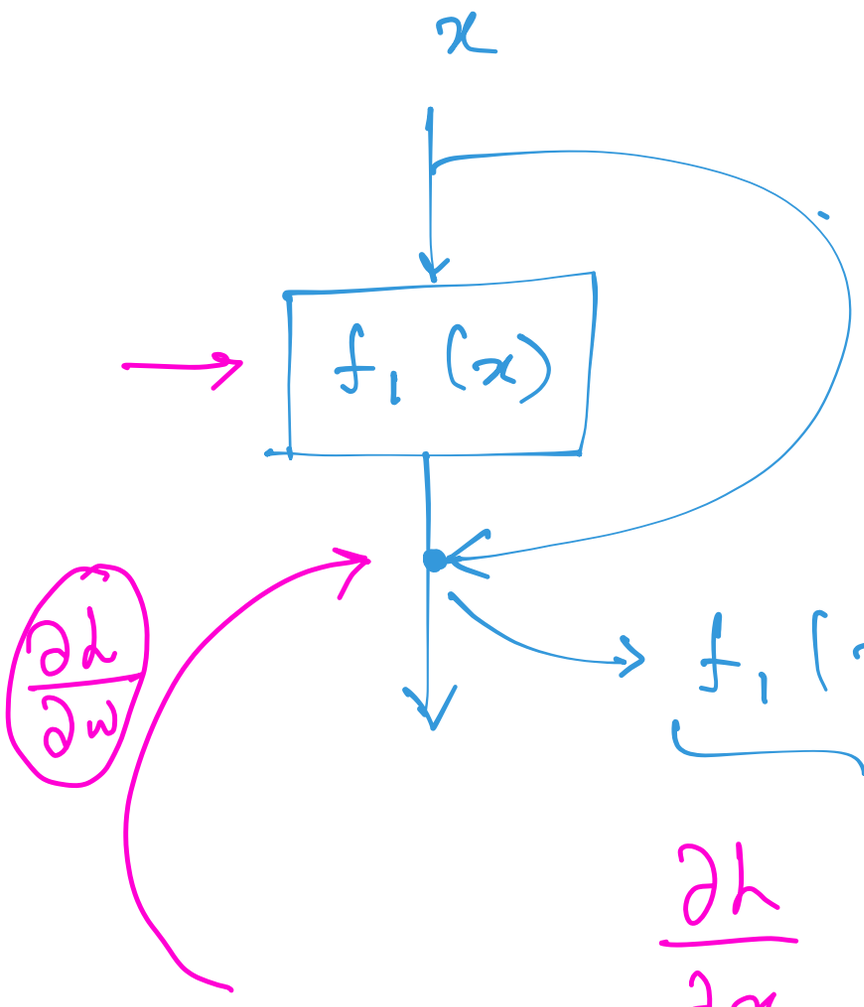
Block 3 → + 7% increase in accuracy.

Backward pass:

$$h'(x) = f_1'(x) + 1$$

$$\frac{\partial h}{\partial x} = \frac{\partial L}{\partial w} \times \frac{\partial w}{\partial x}$$

getting from backprop.



$$f_1(x) + x = h(x)$$

is an addition.

$$\frac{\partial h}{\partial x} = \frac{\partial L}{\partial w} \times \frac{\partial f_1}{\partial x} + \frac{\partial L}{\partial w}$$

$\frac{\partial f_1}{\partial x}$ close to 0 $\rightarrow$ 0

Takeaways:

① CNNs have many types of architecture.

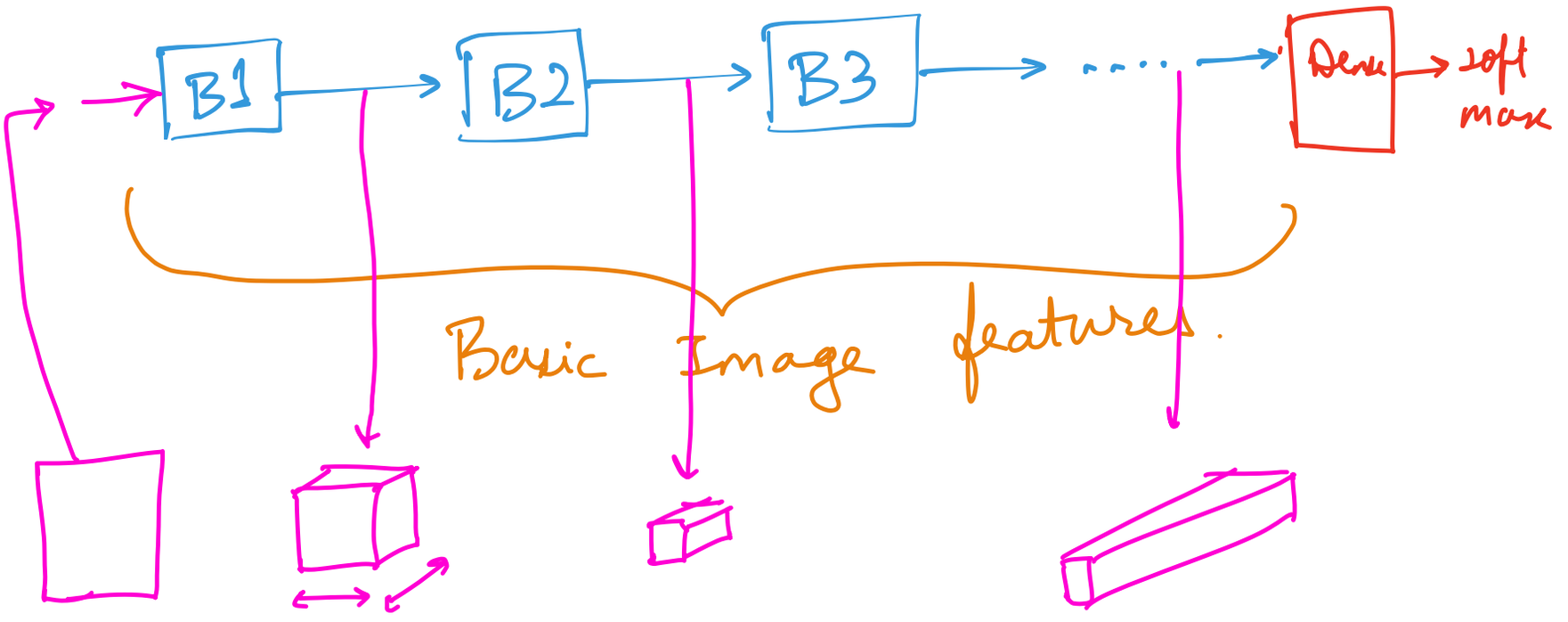
- → Alexnet.
- → VGG.
- → Inception
- → Resnet.

↓ Increasing order of performance.

② The best architecture uses:

* Conv + Relu + GPU.
* Same kernel size (3×3 are okay).
* 1×1 convs to reduce the no. of params.
* Skip connections.

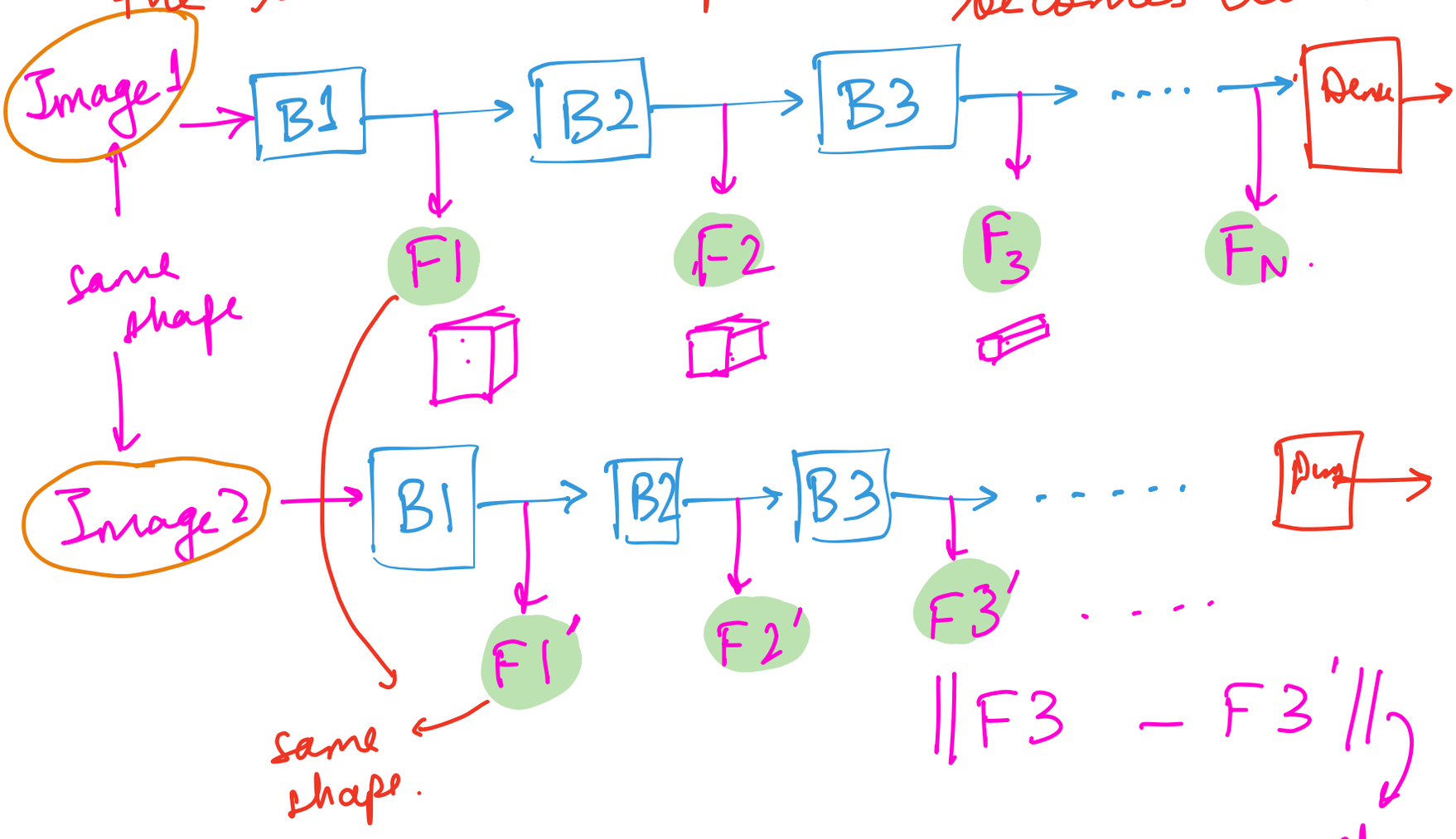
Basic CNN architecture:



How to check if 2 images are similar?

As you go deeper,

the intermediate representations of similar images becomes closer.



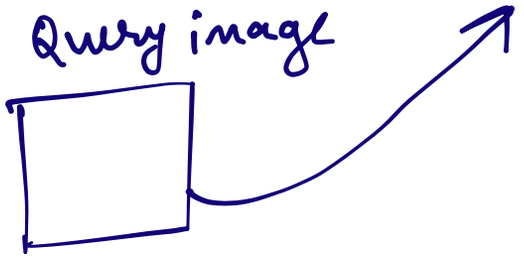
||Image 1 − Image 2|| ⟶ very high

Image Similarity problem:

(a) Start with an already trained network.

→ ResNet ⟶ F37 from ResNet50.

picked after some experimentation.



(b) Compute the F37 for all images in the dataset.

(c) Return the images which are closest to the query image in terms of F37

Embedding from a CNN

↓

Take one of the features, and flatten it!

Time for 1 img → 99.8 ms

≈ 100 ms = 0.1 s.

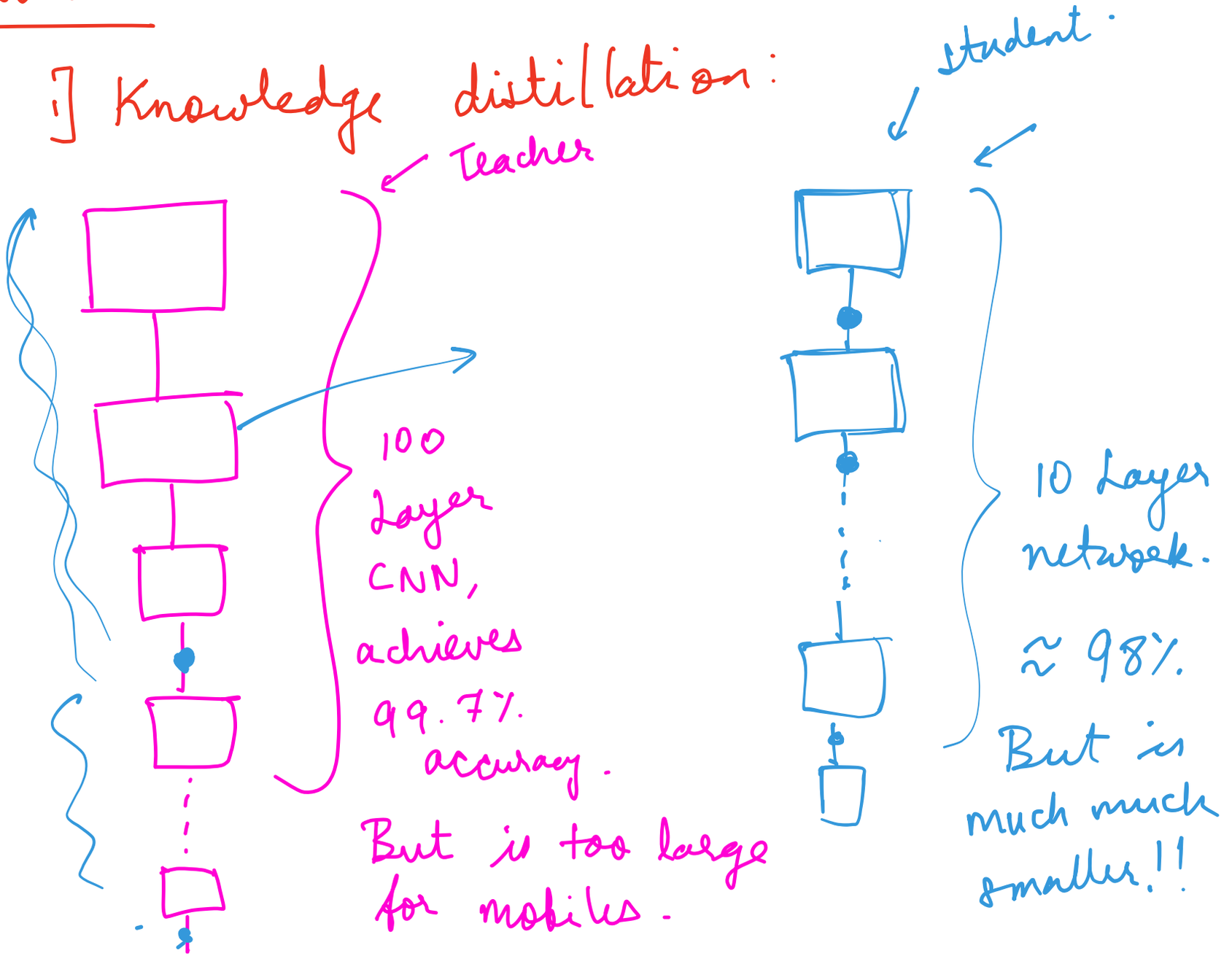
10,000,000 → Can get done pretty fast if we have multiple GPUs.

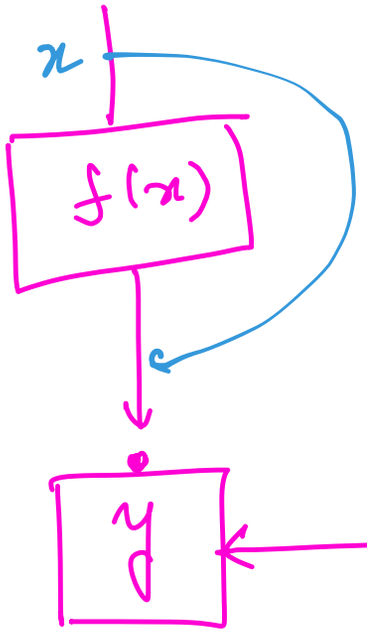
Once we have all f.v.s, apply KNN ↓ to return results!!

K - Nearest Neighbours.

Doubts:

1] Knowledge distillation:





$\frac{\partial L}{\partial y}$

$$y = f(x) + x$$

$$\frac{\partial y}{\partial x} = \frac{\partial f}{\partial x} + 1$$

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial x}.$$

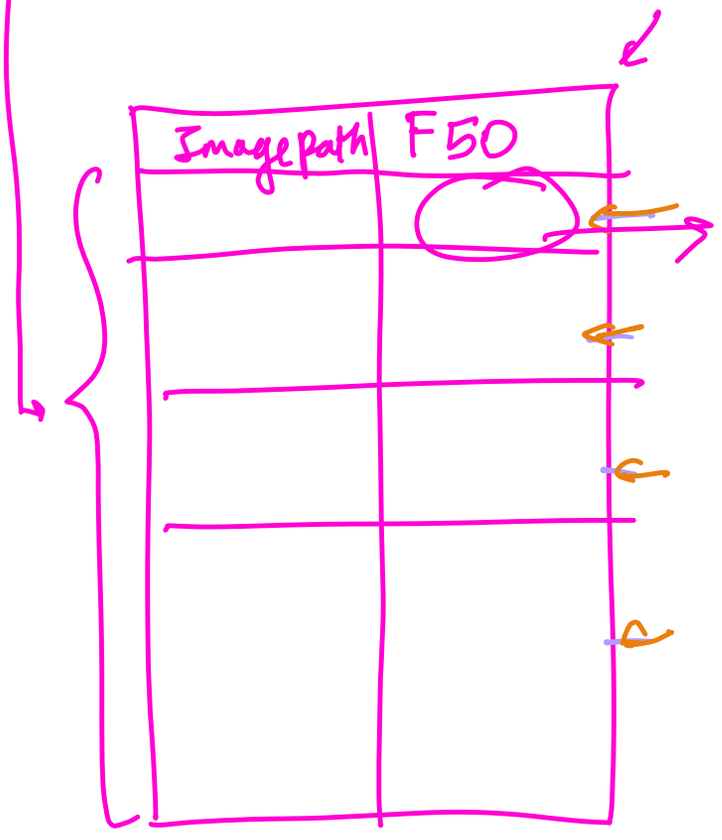
$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y}\left(\frac{\partial f}{\partial x} + 1\right)$$

$$= \underbrace{\frac{\partial L}{\partial y} \cdot \frac{\partial f}{\partial x}}_{\text{even if this goes to 0,}} + \frac{\partial L}{\partial y}.$$

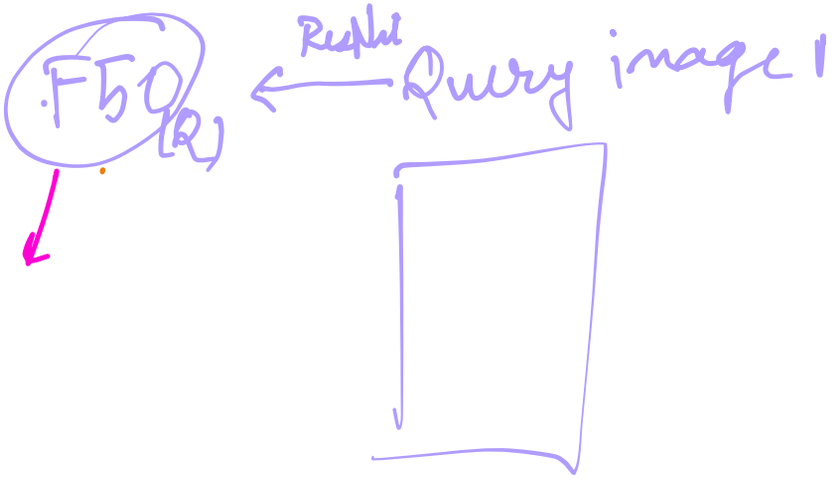
this will update my weights.

10,000,000 ⟶ F50 (of all).

(227 × 227 × 3) (2048).

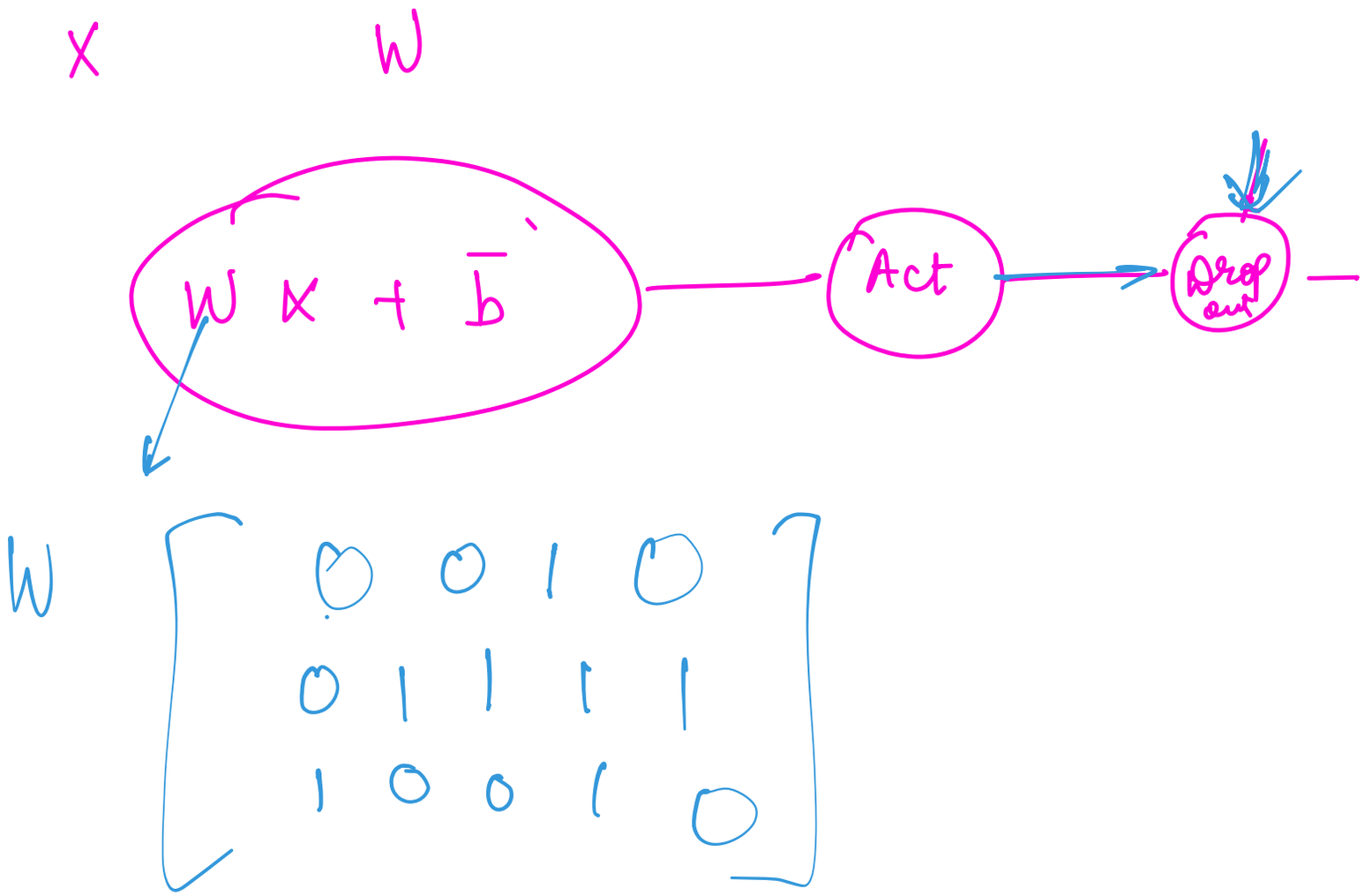


LSH → Locality Sensitive Hashing.



$(F50)_Q$ ← Query imag 2

10,000,000 calculations for each query point.


X
W
Wx + b
Act
Drop out
W
0 0 1 0
0 1 1 1 1
1 0 0 1 0