CNN ARCHITEC....

- CONVOL
- PADDING
- FILTERS (Kernel)
- STRIDES

IMAGENET

what is the reason we learn these arch

CNN ARCHITECTURES

POOLING

Conv

Norm

Tested Archit

All these architectures → well reputed publications

↓

Significant impro → well tested

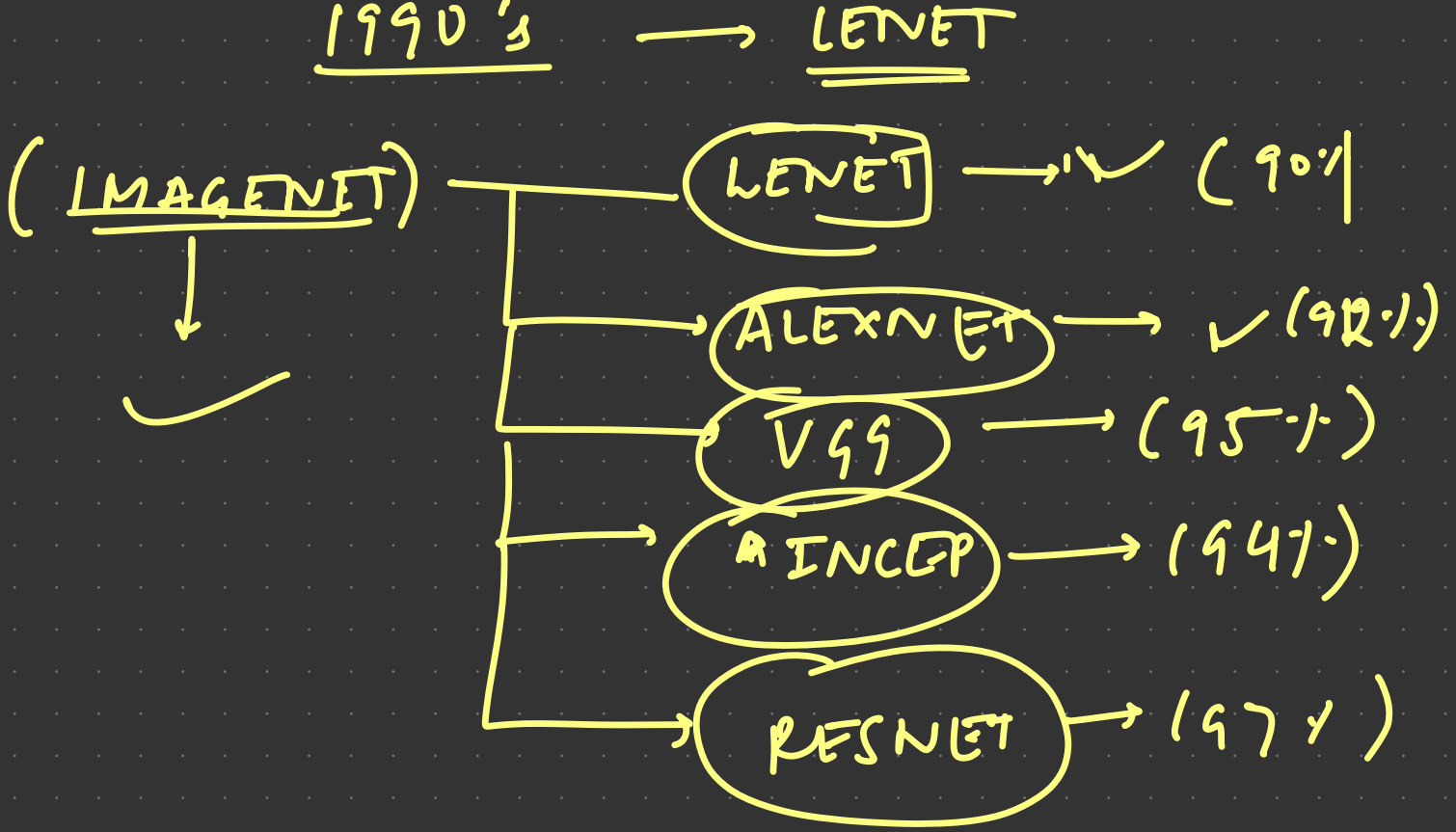
CNN ARCHIT

1. LENET - 5
2. ALEX NET → 8
3. VGG NET - 16/19
4. INCEPTION
5. RESNET - 50/101

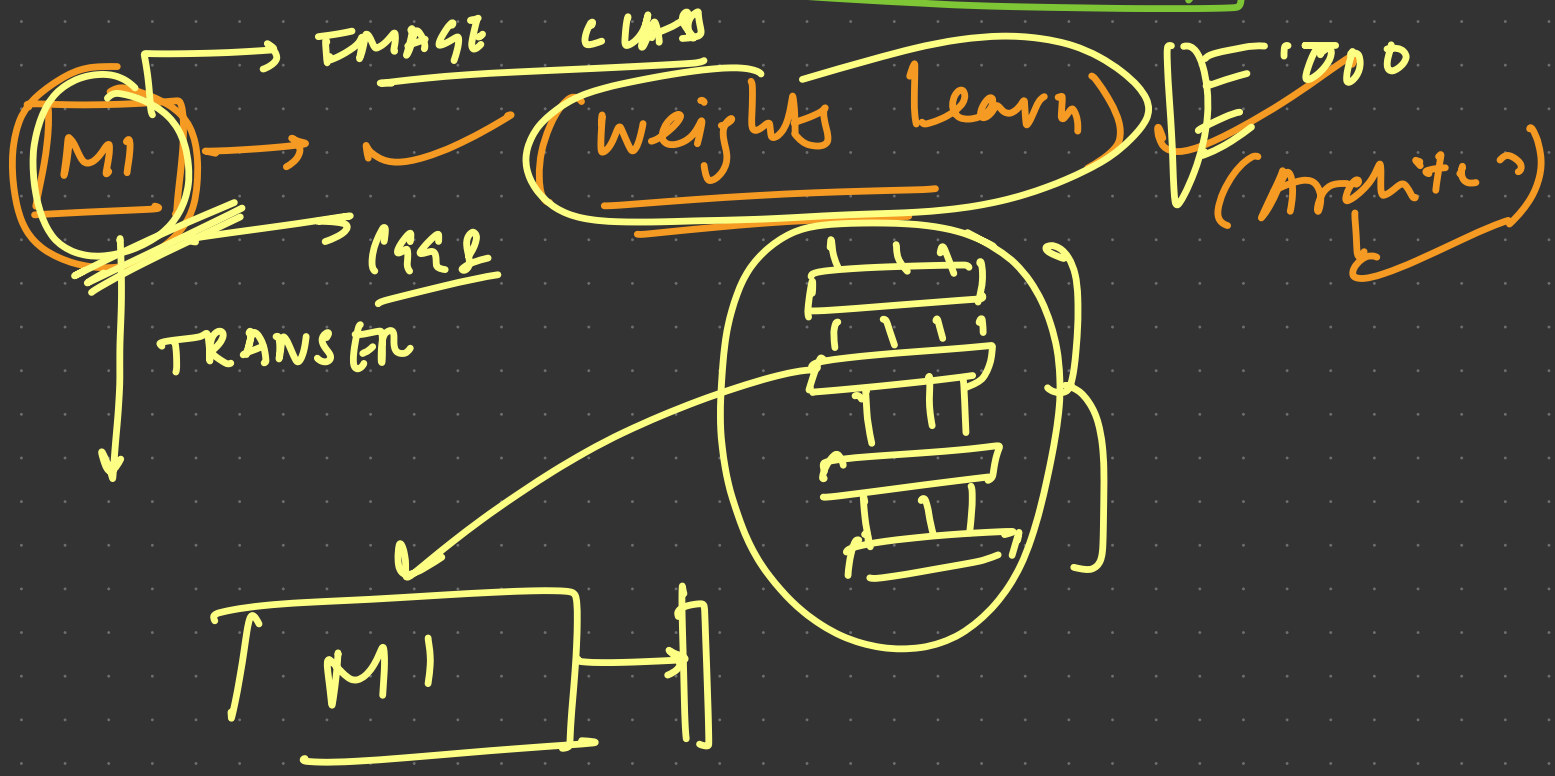
→ Sunday

→ Wednesday

→ (How can we use them)

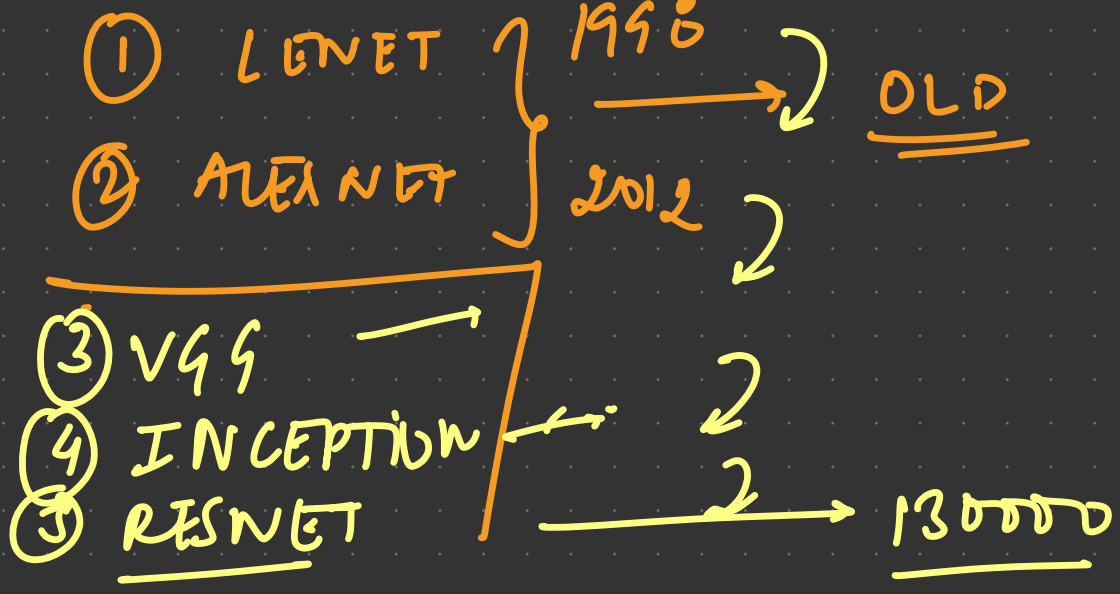

1990's → LENET
(IMAGENET)
LENET → (90%)
ALEXNET → (92.1%)
VGG → (95.1%)
INCEP → (94%)
RESNET → (97%)
Comb of CONV, POOL ....

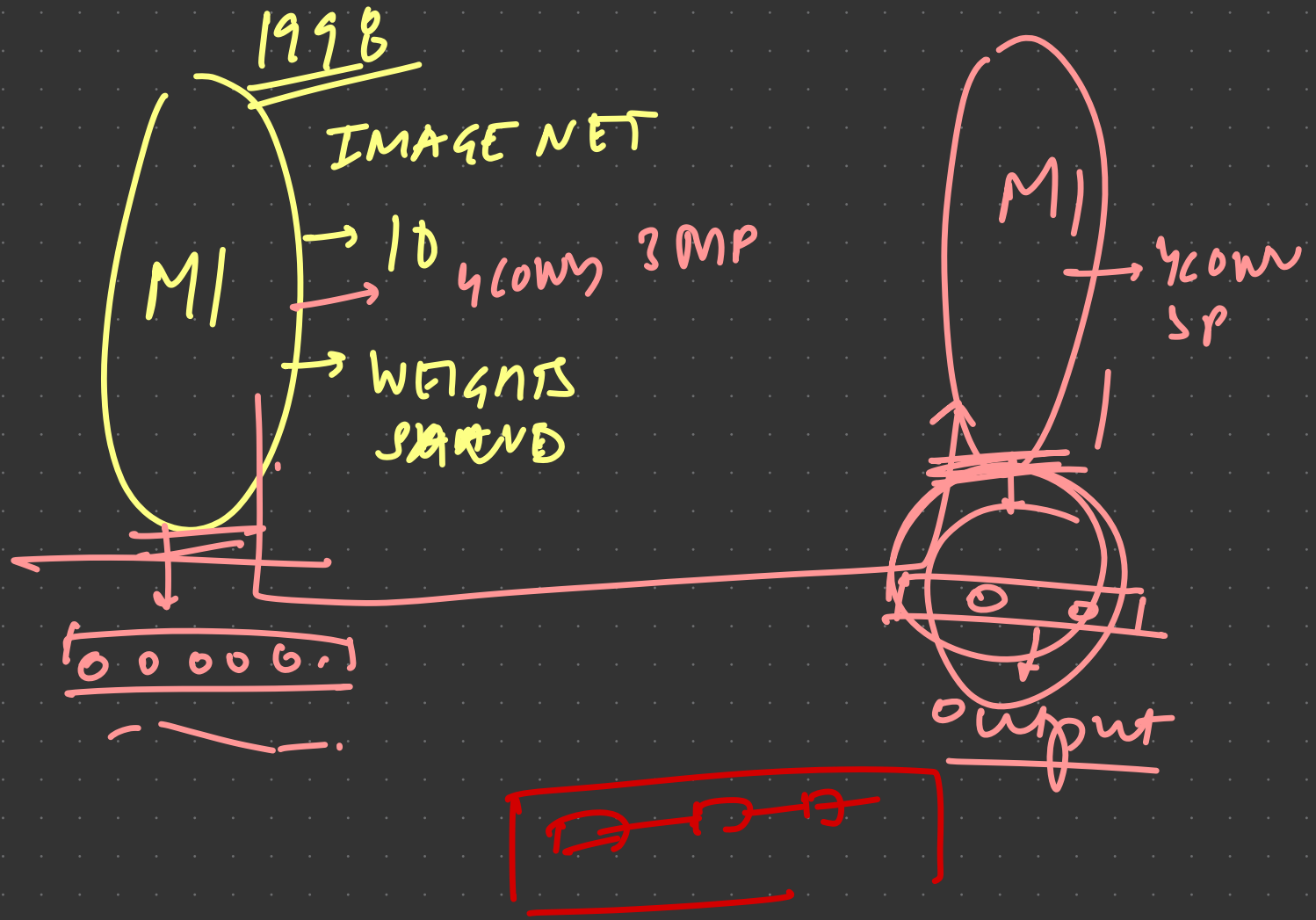
TRANSFER LEARNING



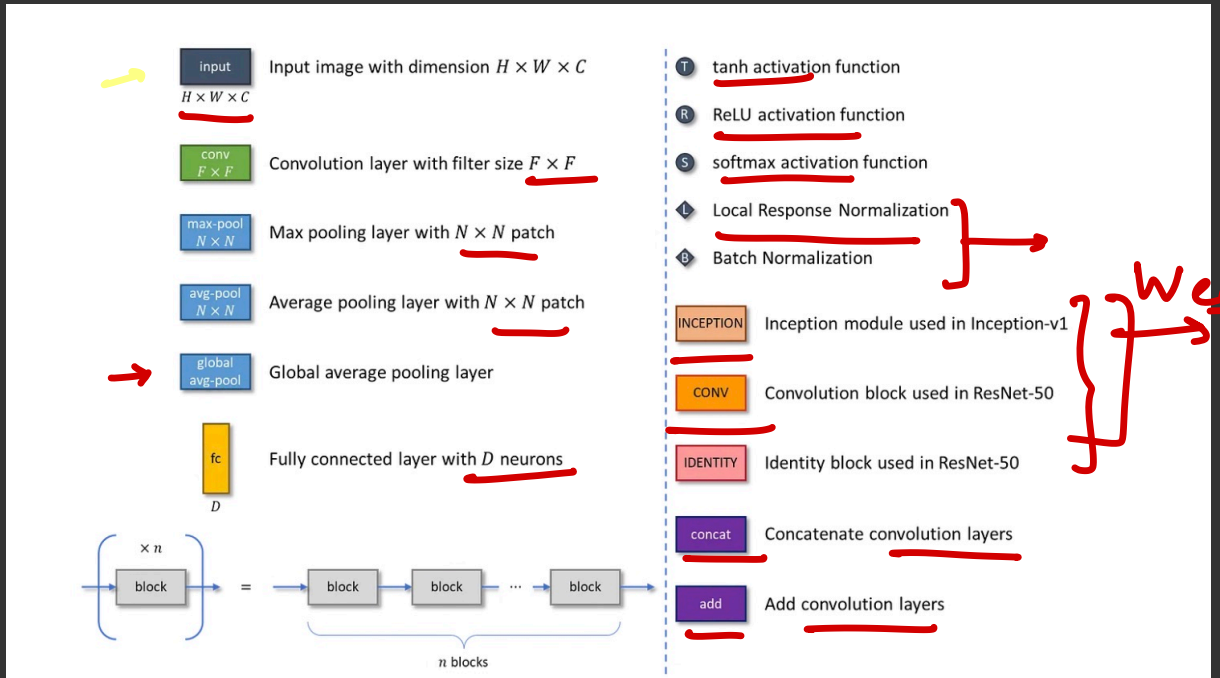
TRANSFER LEARNING

USING a PRE-TRAINED model we use it's weights for a new use case by making changes in the final layer.




1998
IMAGE NET
M1
10
4 CONV 3 MP
WEIGHTS
M1
4 CONV
3 P
output

GLOSSARY

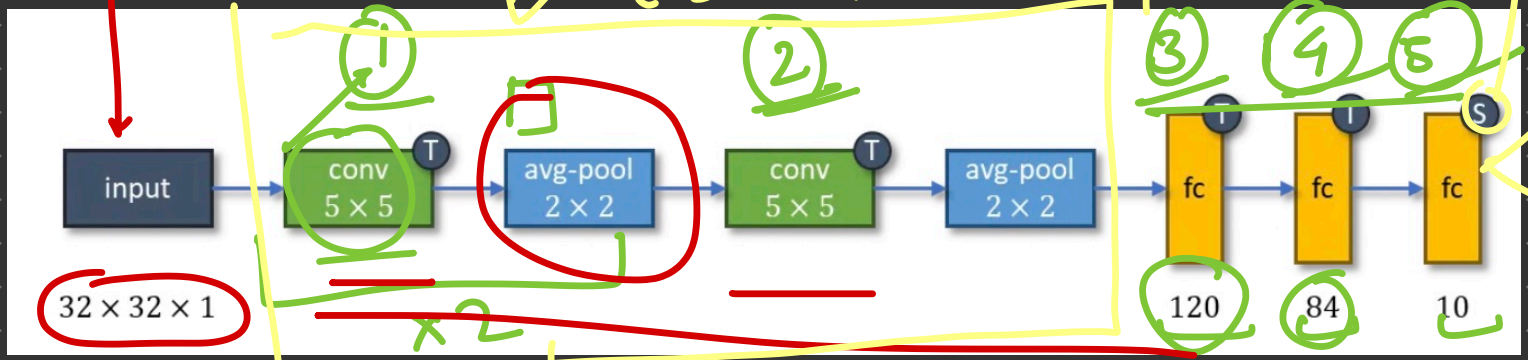


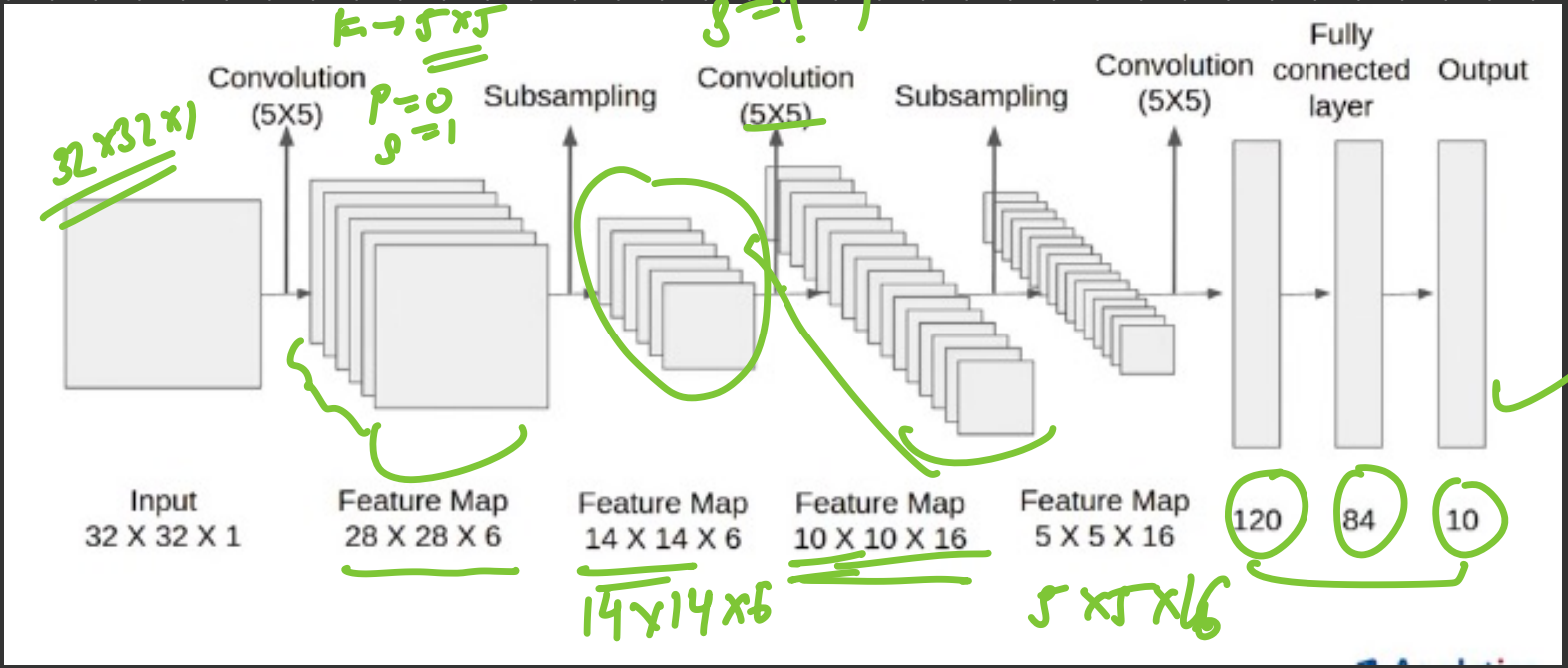
Legend used in this story | Image by author

LENET - 5 (1998) → IEEE

Activation F → tanh / Sigmoid

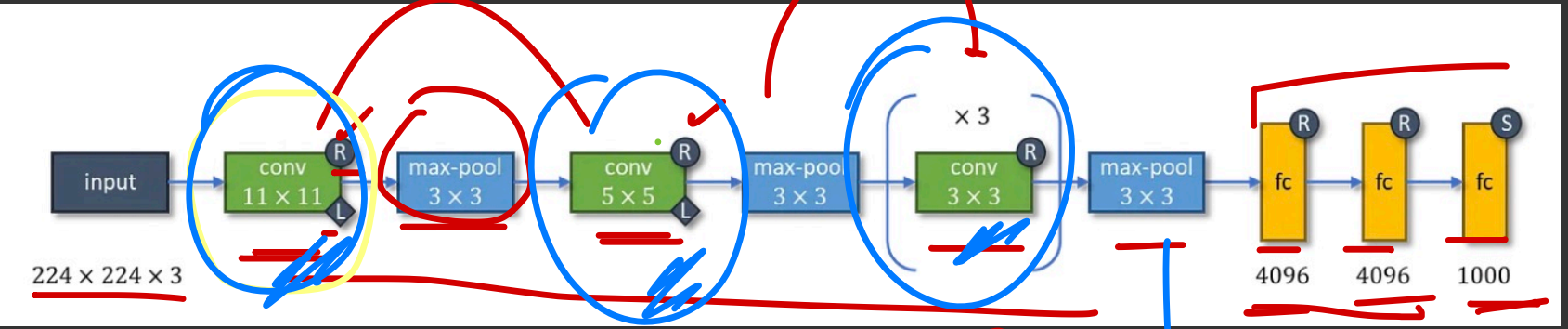
(SAME)





$$\left(\frac{32 - 5 + 2P}{1}\right) + 1$$

ALEX NET → ① ReLU over tanh
② 5 → 8
③ Max pooling



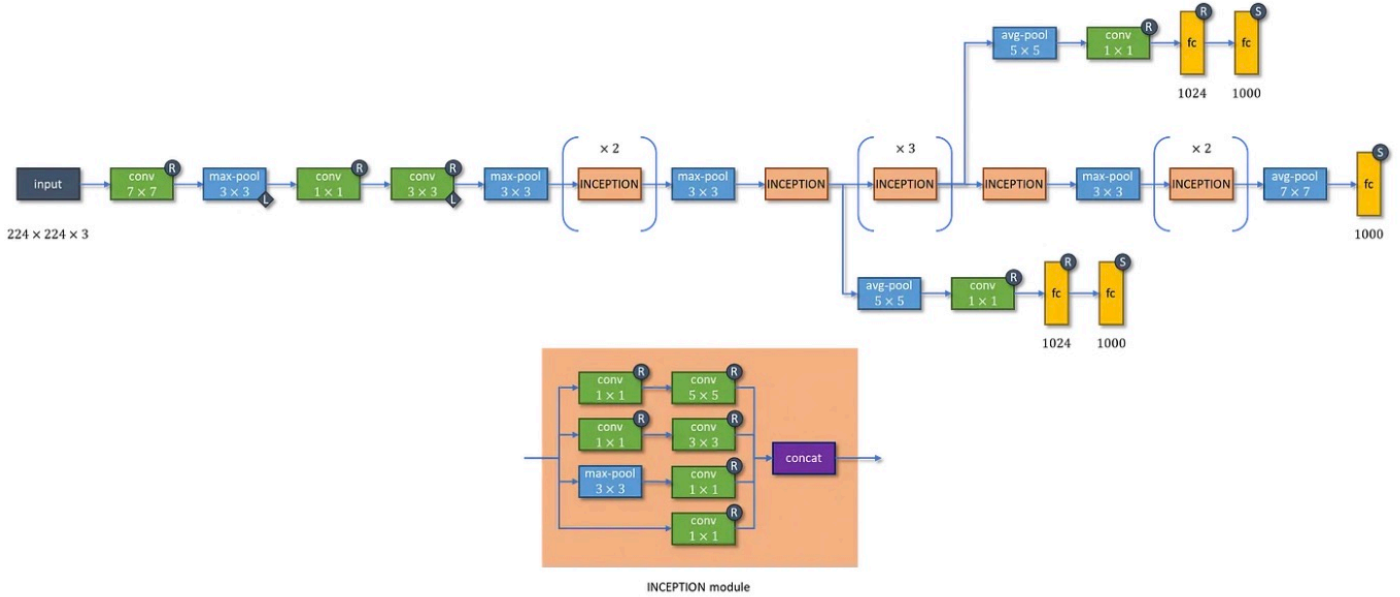
Conv 11x11

Conv 5x5

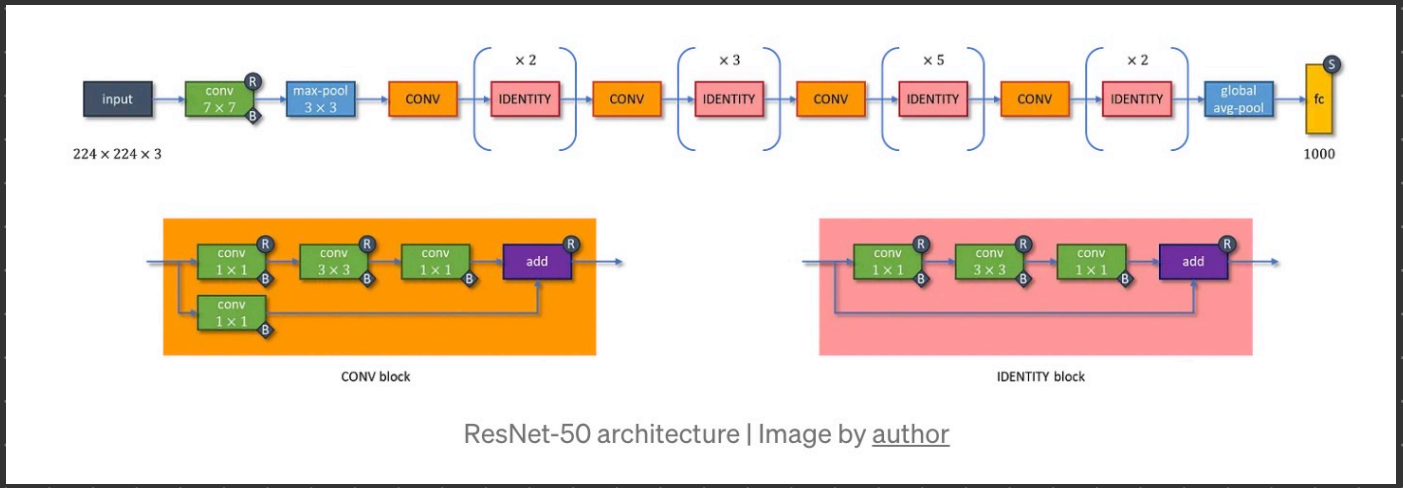
Conv 3x3

VGG

INCEPTION



Inception-v1 architecture | Image by author

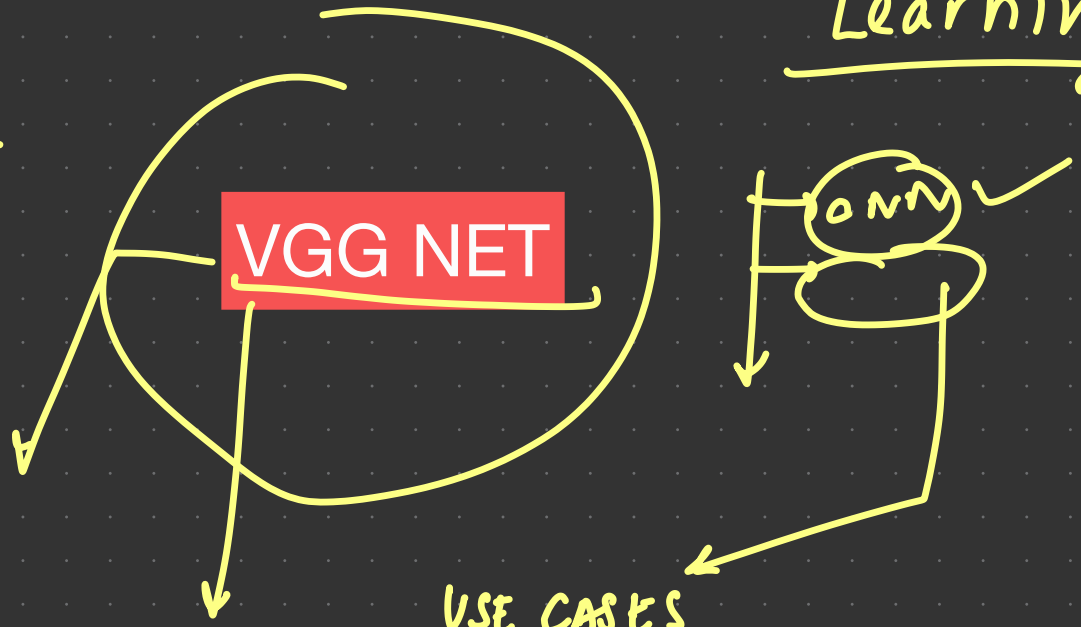


ResNet-50 architecture | Image by author

CNN

Transfer Learning

1st Solution



VGG-16 VGG-19

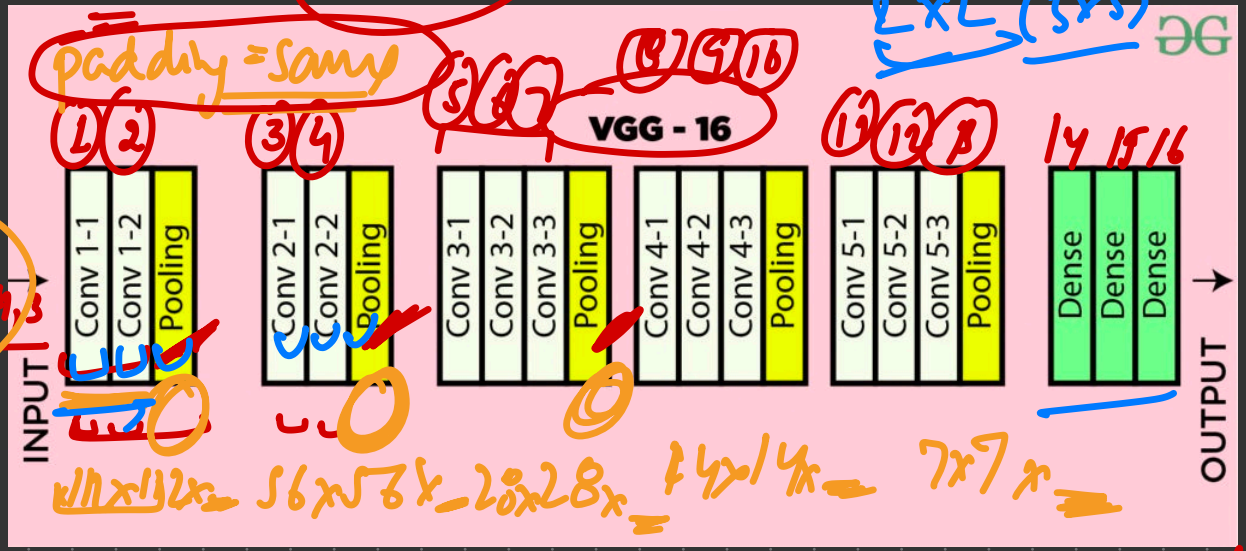
16 | 19

Number of layers with trainable params.

2015 → ImageNet

8 → 16

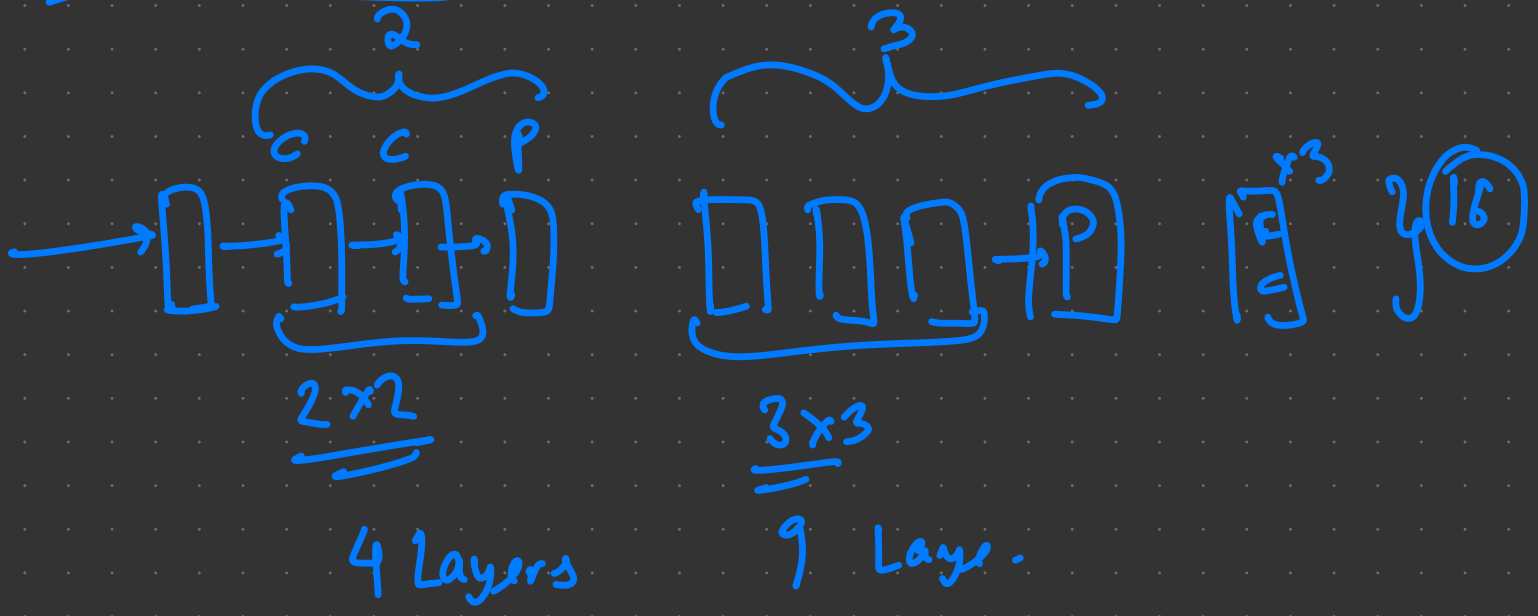
VGG NET



- All convolution layers were made to have kernel sizes equal to 3x3.
- CNN → deep → 16 layers → 19 layers
- LRN → Remove.

VGG-16

3×3 | 2×2, 2



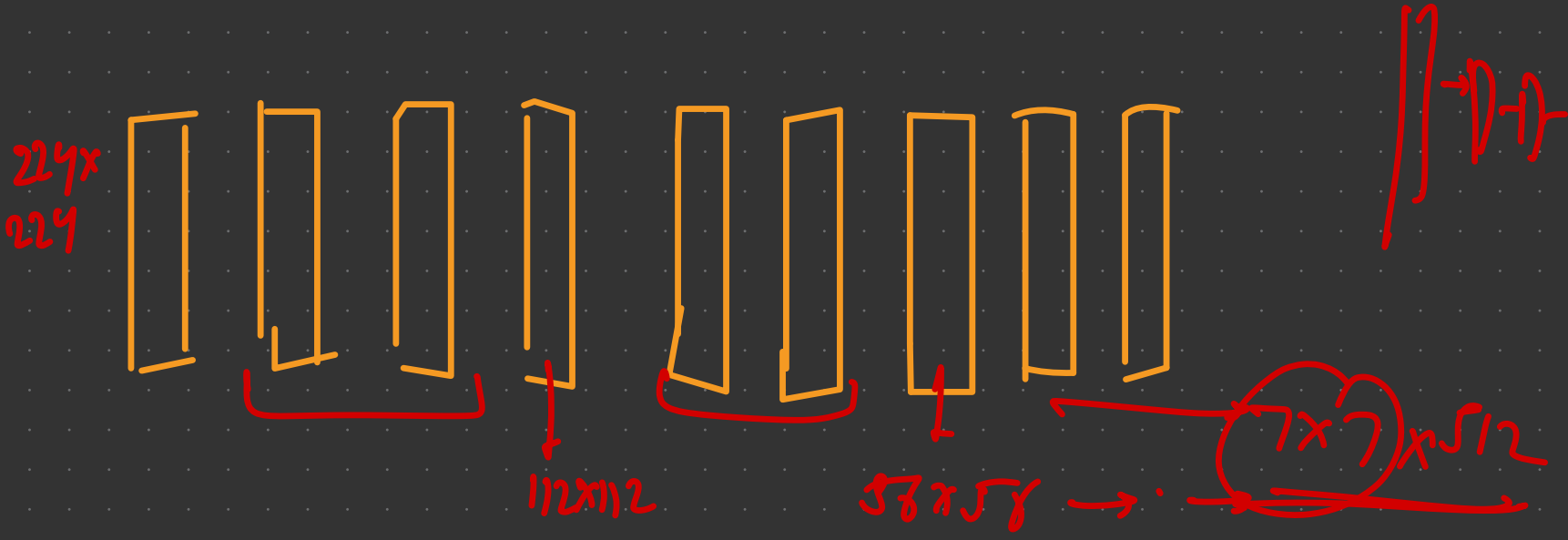
4 Layers

9 Layer.

- Maxpooling → 2×2, stride (2) (NO OVERLAP)
  - ↳ Alternate (3×3) 2 (OVERLAP)

VGG → 2014-15 → DEEP CONV NET

- → LRN ×
- → kernel size small
- → deep


224x
224
112x112
56x56
7x7x512

VGGNet made several changes and improvements compared to AlexNet. Here are the key changes made in VGGNet from AlexNet:

1. Deeper Architecture: VGGNet introduced a much deeper architecture compared to AlexNet. While AlexNet had eight layers, VGGNet had 16 or 19 layers, depending on the variant. The increased depth allowed VGGNet to capture more complex features and patterns.

2. Smaller Filter Sizes: VGGNet utilized smaller 3x3 convolutional filters throughout the network, whereas AlexNet used a mixture of 3x3 and 5x5 filters. The use of smaller filters allowed for a more localized feature representation and enabled deeper networks without a significant increase in parameters.

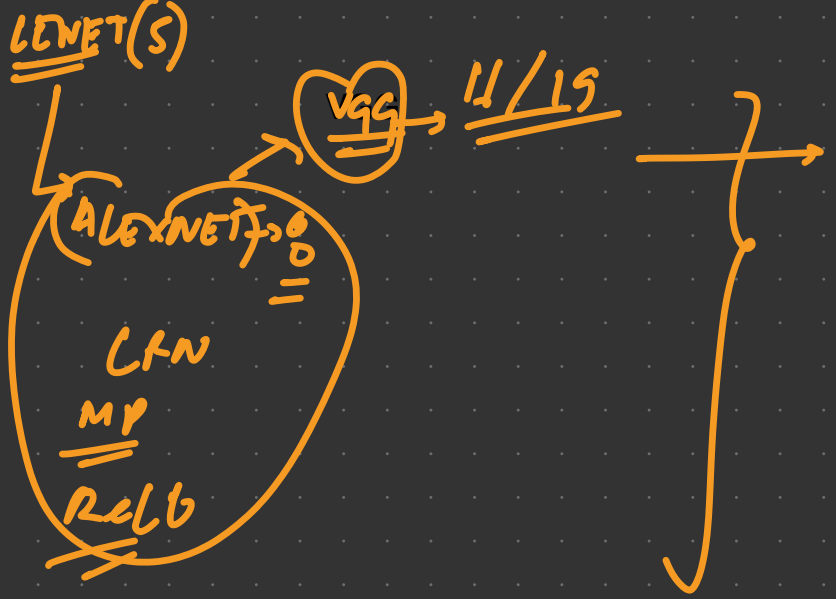
3. Uniform Structure: VGGNet maintained a uniform structure with a stack of convolutional layers followed by max pooling layers. It followed a consistent pattern of stacking multiple 3x3 convolutions and then downsampling using max pooling layers. This uniformity made it easier to design and understand the network architecture.

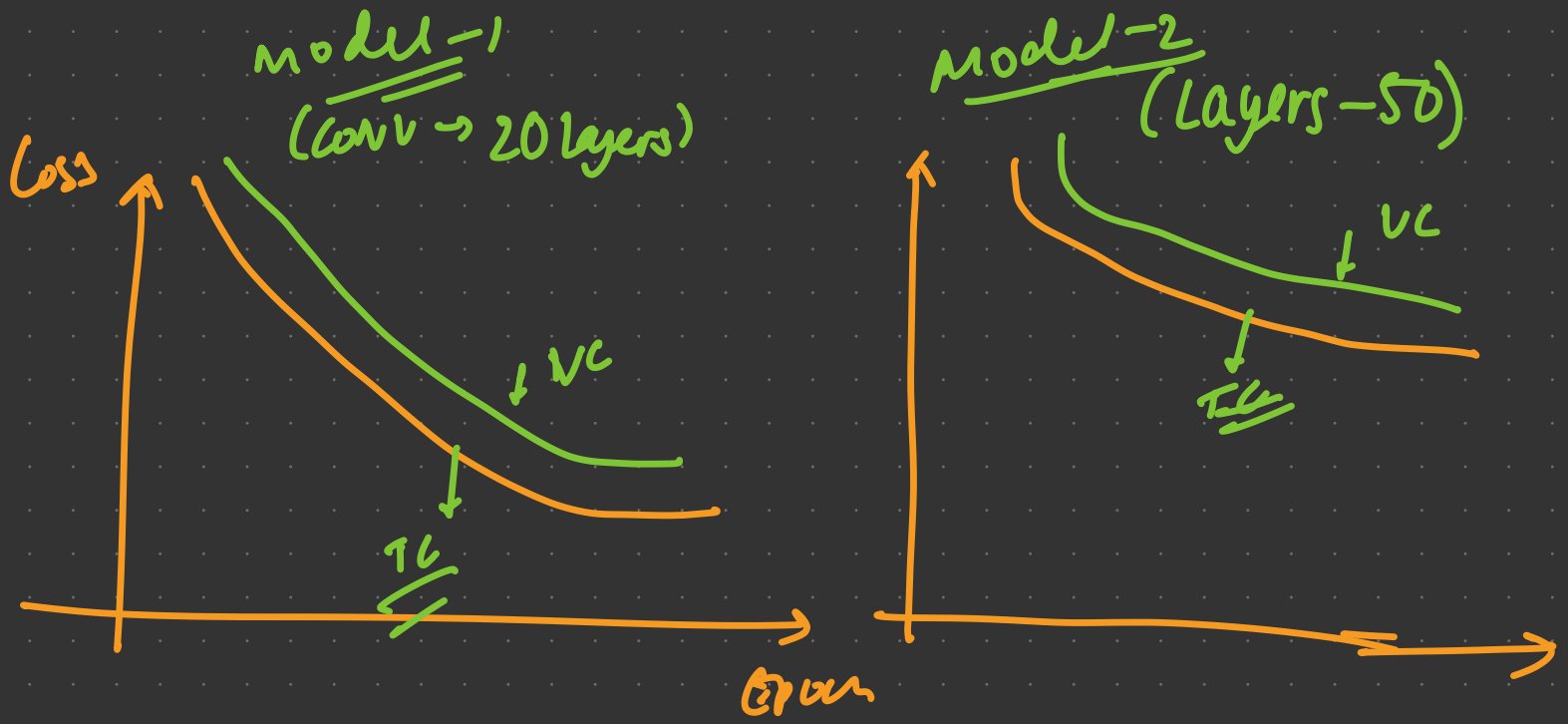
4. No Overlapping Pooling: VGGNet used max pooling layers with a stride of 2, resulting in non-overlapping pooling regions. In contrast, AlexNet used pooling regions with a stride of 2 but with overlapping regions of size 3x3. The non-overlapping pooling in VGGNet helped reduce spatial resolution more gradually.

5. Elimination of Local Response Normalization (LRN): VGGNet removed the LRN layers that were present in AlexNet. LRN was initially introduced in AlexNet as a technique to enhance local contrast. However, subsequent research found that LRN had limited impact on performance and was computationally expensive. Hence, VGGNet relied solely on convolutional and pooling layers without LRN.

6. Smaller Fully Connected Layers: VGGNet used smaller fully connected layers compared to AlexNet. Instead of having large fully connected layers with 4096 neurons, VGGNet used 4096 neurons in the last two fully connected layers and reduced the number of neurons to 1000 in the output layer. This change helped reduce the number of parameters and computational complexity.

Overall, VGGNet improved the depth and architectural design of CNNs compared to AlexNet. By using smaller filter sizes, a uniform structure, and deeper networks, VGGNet achieved state-of-the-art performance in image classification tasks at the time of its introduction.


LENET(5)
VGG
11/19
ALEXNET
D
LRN
MP
RELU



<INCEPTION>

The Inception network was an important milestone in the development of CNN classifiers. Prior to its inception (pun intended), most popular CNNs just stacked convolution layers deeper and deeper, hoping to get better performance.

The Inception network on the other hand, was complex (heavily engineered). It used a lot of tricks to push performance; both in terms of speed and accuracy. Its constant evolution lead to the creation of several versions of the network. The popular versions are as follows:

Inception v1.
Inception v2 and Inception v3
Inception v4 and Inception-ResNet.
Each version is an iterative improvement over the previous one. Understanding the upgrades can help us to build custom classifiers that are optimized both in speed and accuracy. ~~Also, depending on your data, a lower version may actually work better.~~ (Edit: Removed this sentence as it was rather speculative; please ignore the same).

.

Inception v1
This is where it all started. Let us analyze what problem it was purported to solve, and how it solved it. (Paper)

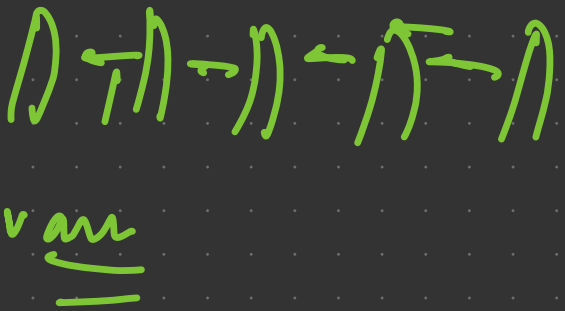
The Premise:
Salient parts in the image can have extremely large variation in size. For instance, an image with a dog can be either of the following, as shown below. The area occupied by the dog is different in each image.

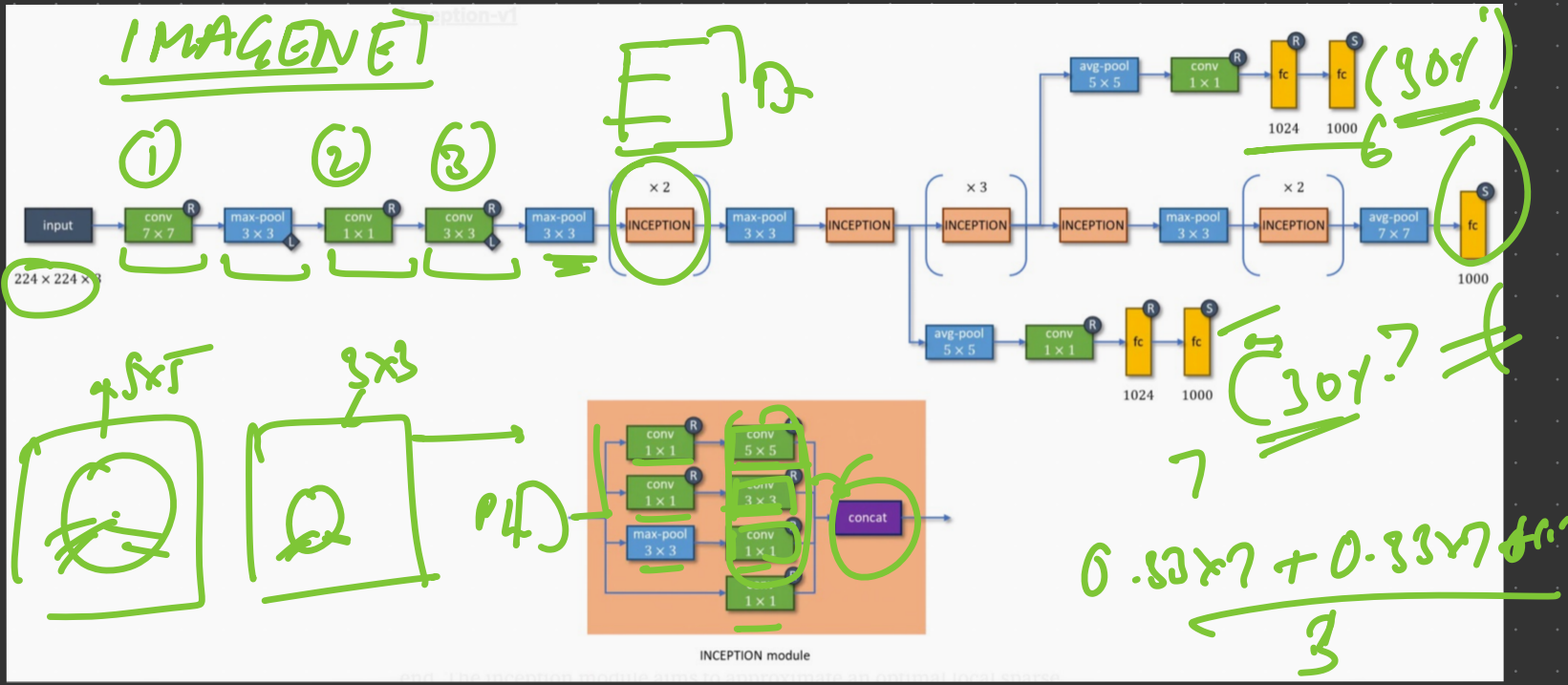
From left: A dog occupying most of the image, a dog occupying a part of it, and a dog occupying very little space (Images obtained from Unsplash).
Because of this huge variation in the location of the information, choosing the right kernel size for the convolution operation becomes tough. A larger kernel is preferred for information that is distributed more globally, and a smaller kernel is preferred for information that is distributed more locally.
Very deep networks are prone to overfitting. It also hard to pass gradient updates through the entire network.
Naively stacking large convolution operations is computationally expensive.

INCEPTION-V1

Going deeper has a caveat: exploding/vanishing gradients:

1. The exploding gradient is a problem when large error gradients accumulate and result in unstable weight updates during training.
2. The vanishing gradient is a problem when the partial derivative of the loss function approaches a value close to zero and the network couldn't train.




IMAGENET
input
224 × 224 × 3
conv 7 × 7
max-pool 3 × 3
conv 1 × 1
conv 3 × 3
max-pool 3 × 3
× 2
INCEPTION
max-pool 3 × 3
INCEPTION
× 3
INCEPTION
avg-pool 5 × 5
conv 1 × 1
fc
1024
fc
1000
INCEPTION
max-pool 3 × 3
× 2
INCEPTION
avg-pool 7 × 7
fc
1000
avg-pool 5 × 5
conv 1 × 1
fc
1024
fc
1000
conv 1 × 1
conv 1 × 1
max-pool 3 × 3
conv 5 × 5
conv 3 × 3
conv 1 × 1
1 × 1
concat
INCEPTION module

Inception-v1 tackles this issue by adding two auxiliary classifiers connected to intermediate layers, with the hope to increase the gradient signal that gets propagated back. During training, their loss gets added to the total loss of the network with a 0.3 discount weight. At inference time, these auxiliary networks are discarded.

Inception-v1 introduces the inception module, four series of one or two convolution and max-pool layers stacked in parallel and concatenated at the end. The inception module aims to approximate an optimal local sparse structure in a CNN by allowing the use of multiple types of kernel sizes, instead of being restricted to single kernel size.

Inception-v1 has fewer parameters than AlexNet and VGG-16, a mere 7 million, even though it consists of 22 layers:

3 convolution layers with 7×7, 1×1, and 3×3 kernel sizes, followed by
18 layers that consist of 9 inception modules where each has 2 layers of convolution/max-pooling, followed by
1 fully connected layer.
The last layer of the main classifier and the two auxiliary classifiers are equipped with a softmax activation function and all others are equipped with ReLU. The 1st and 3rd convolution layers, also the 2nd and 7th inception modules are followed by a 3×3 max-pooling. The last inception module is followed by a 7×7 average-pooling. LRN is applied after the 1st max-pooling and the 3rd convolution layer.

Auxiliary classifiers are branched out after the 3rd and 6th inception modules, each starts with a 5×5 average-pooling and is then followed by 3 layers:

1 convolution layer with 1×1 kernel size, and
2 fully connected layers.
Default Inception-v1 accepts colored images with dimensions 224×224 and outputs one of the 1000 classes.

RESNET

① CONV BLOCK

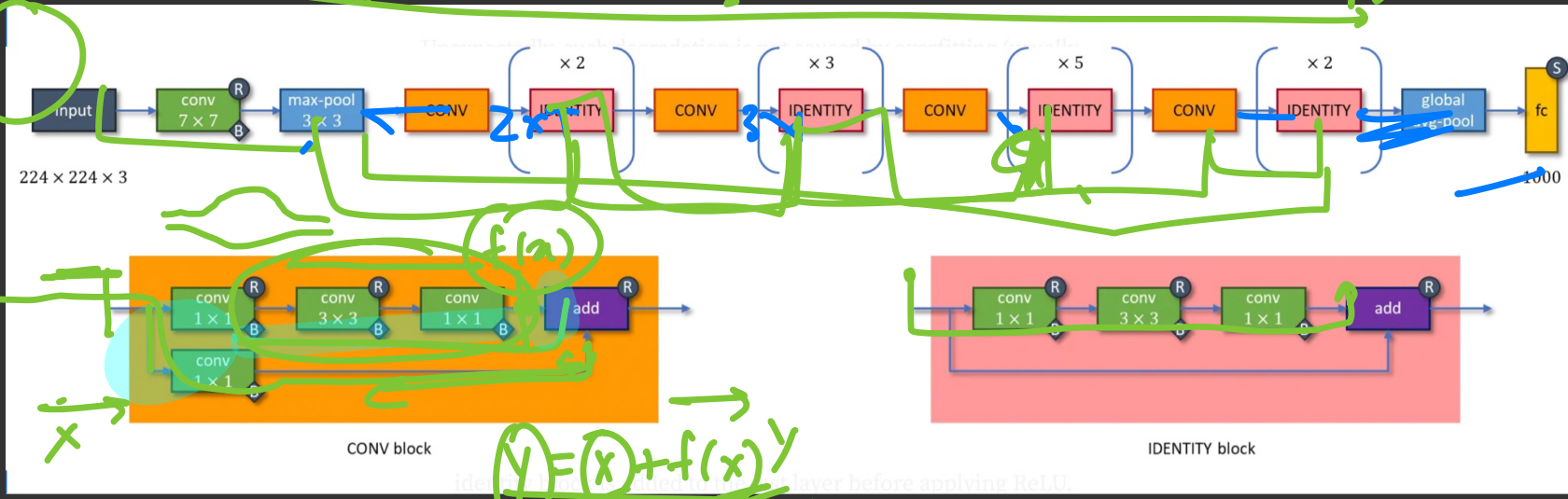
② IDENTITY BLOCK

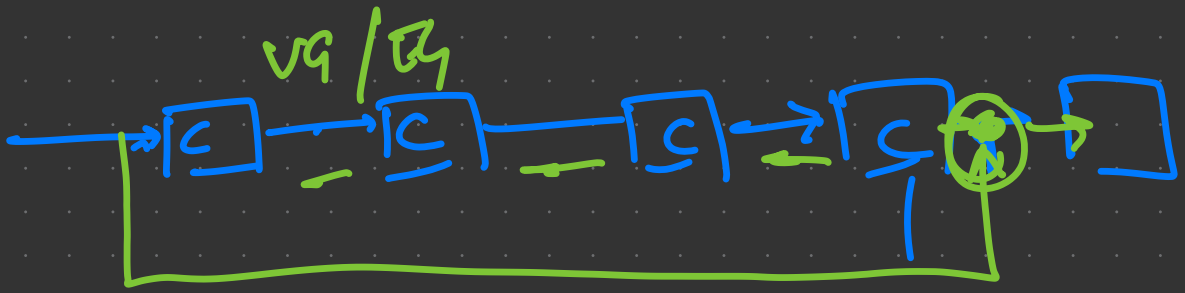
*SKIP CONNECTIONS

CONV

$y = x + \Delta x$

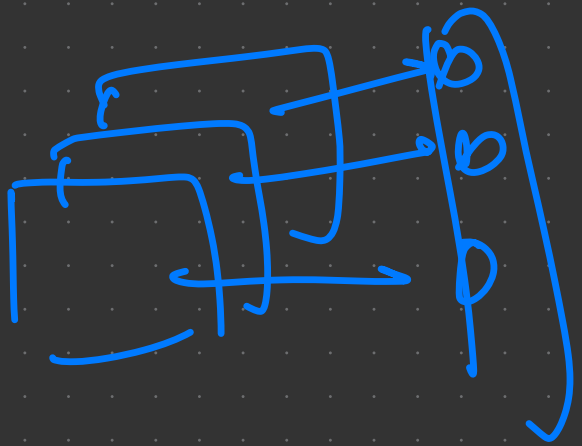
SKIP CONNECTIONS




VG / BG

When deeper networks can start converging, a degradation problem has been exposed: with the network depth increasing, accuracy gets saturated and then degrades rapidly.

Unexpectedly, such degradation is not caused by overfitting (usually indicated by lower training error and higher testing error) since adding more layers to a suitably deep network leads to higher training error.



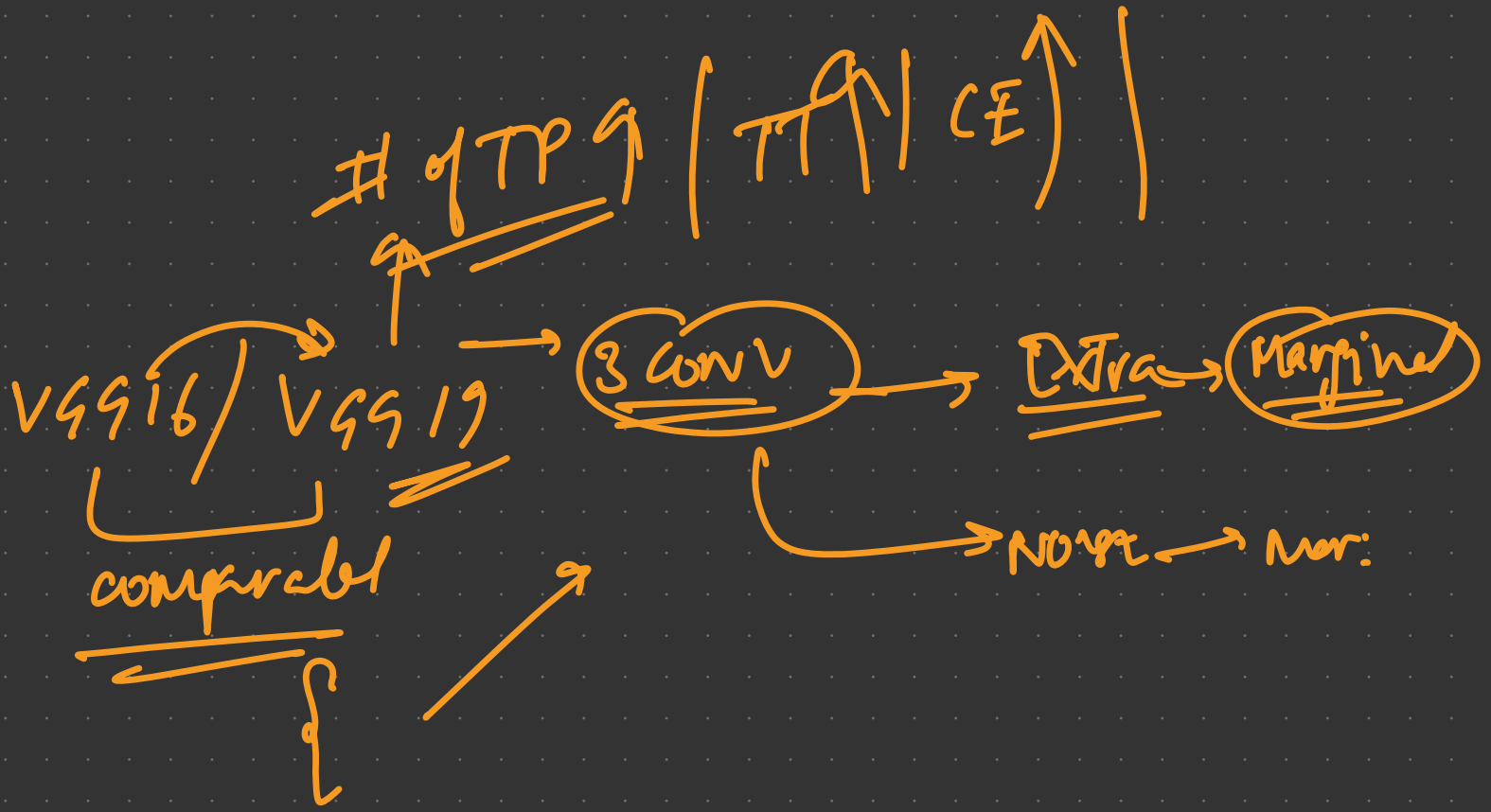
The degradation problem is addressed by introducing bottleneck residual blocks. There are 2 kinds of residual blocks:

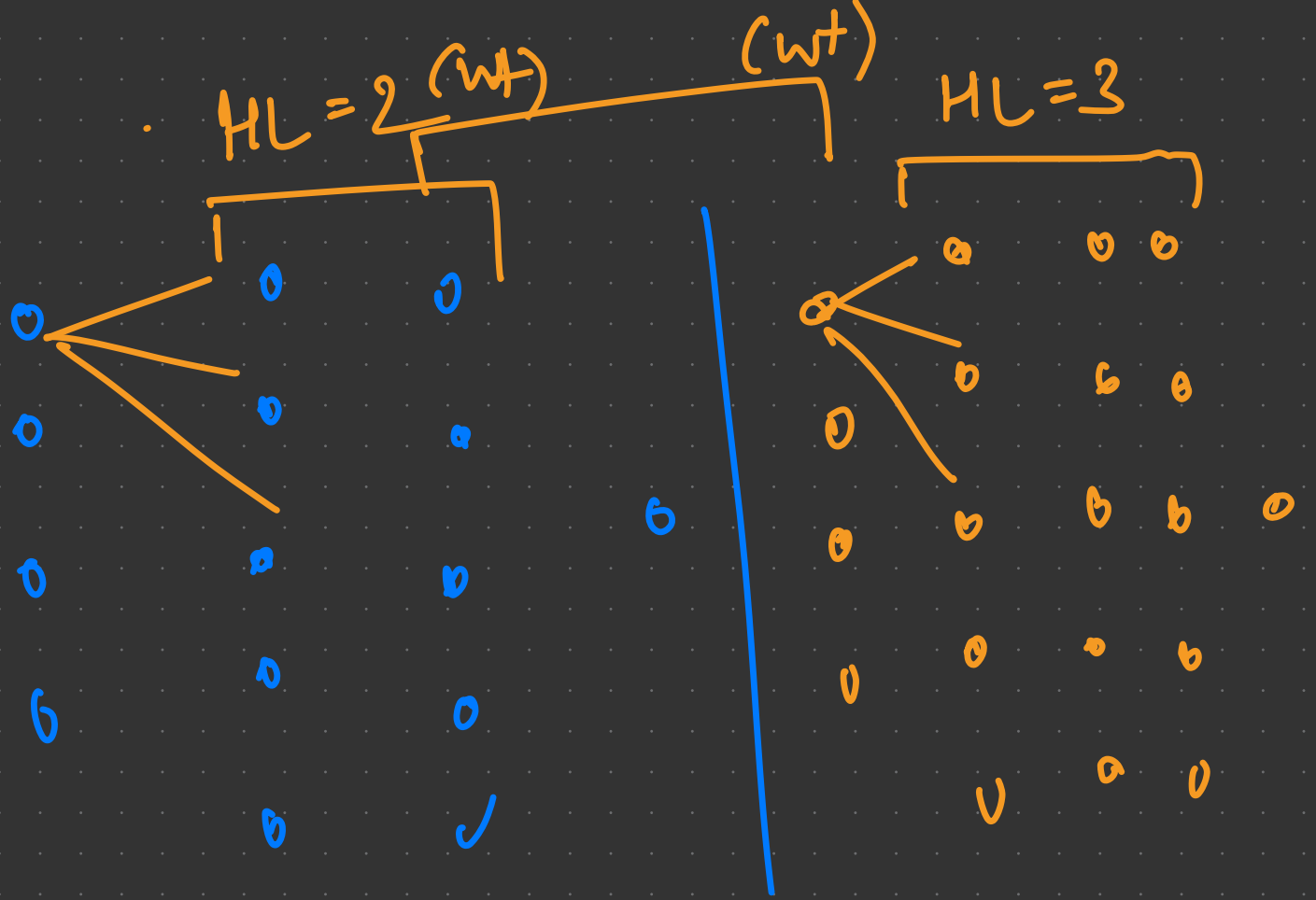
Identity block: consists of 3 convolution layers with 1×1, 3×3, and 1×1 kernel sizes, all of which are equipped with BN. The ReLU activation function is applied to the first two layers, while the input of the identity block is added to the last layer before applying ReLU.
Convolution block: same as identity block, but the input of the convolution block is first passed through a convolution layer with 1×1 kernel size and BN before being added to the last convolution layer of the main series.
Notice that both residual blocks have 3 layers. In total, ResNet-50 has 26 million parameters and 50 layers:

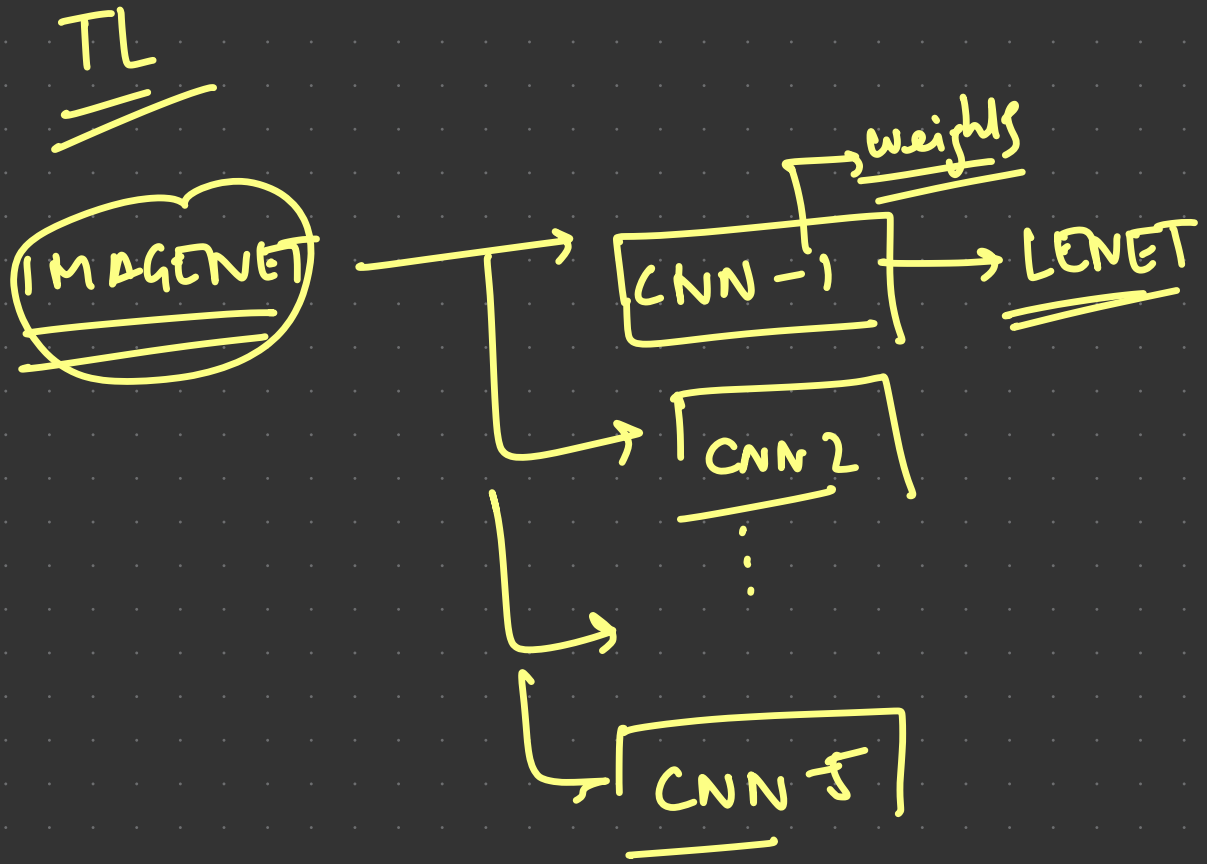
1 convolution layer with BN then ReLU is applied, followed by
9 layers that consist of 1 convolution block and 2 identity blocks, followed by
12 layers that consist of 1 convolution block and 3 identity blocks, followed by
18 layers that consist of 1 convolution block and 5 identity blocks, followed by
9 layers that consist of 1 convolution block and 2 identity blocks, followed by
1 fully connected layer with softmax.

The first convolution layer is followed by a 3×3 max-pooling and the last identity block is followed by a global-average-pooling. Default ResNet-50 accepts colored images with dimensions 224×224 and outputs one of the 1000 classes.

CNN Architecture	Default Input	Default Output	Number of Layers	Number of Parameters	Activation Function	New Additional Perks
LeNet-5	32×32×1	10	5	60K	tanh	Convolution Layer
AlexNet	224×224×3	1000	8	60M	ReLU	Local Response Normalization
VGG-16	224×224×3	1000	16	138M	ReLU	Very deep but still single thread
Inception-v1	224×224×3	1000	22	7M	ReLU	Auxiliary Classifiers & Inception Module
ResNet-50	224×224×3	1000	50	26M	ReLU	Batch Normalization & Residual Blocks


#of TPs
VGG16 / VGG19
3 conv
Extra
Multi
comparable
NMS
Nor:


HL = 2 (wt)
(wt)
HL = 3


TL
IMAGENET
weights
CNN-1
LENET
CNN 2
CNN J