$$\frac{\partial L}{\partial W_1} = \text{Conv}\left(X, \frac{\partial L}{\partial Z_1}\right)$$

$$\frac{\partial L}{\partial b_1} = \text{Sum}\left(\frac{\partial L}{\partial Z_1}\right)$$

Pre-trained Models

Image NET Dataset

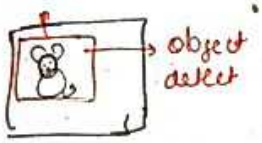
↳ visual database of image (why what and how discuss)

Why

2006 → Fei Fei Li → working on model and algo

build database → 1.4 million images → 20,000 categories
↳ with label

also 1 million image → boundig box



How

~~Goog~~ Croud sourcing like (Captcha) → ask about image and detect where's image.

ILSVRC

ImageNET challenge → Polo → Dataset → Subset → 1 million out of 14 million
↳ 1000 category's

Model in Challenge

↓

Machine Learning model

- feature extraction
- ensemble learning

error rate

2010 → 28% → ML Model → CPU

2011 → 25% → ML Model → CPU

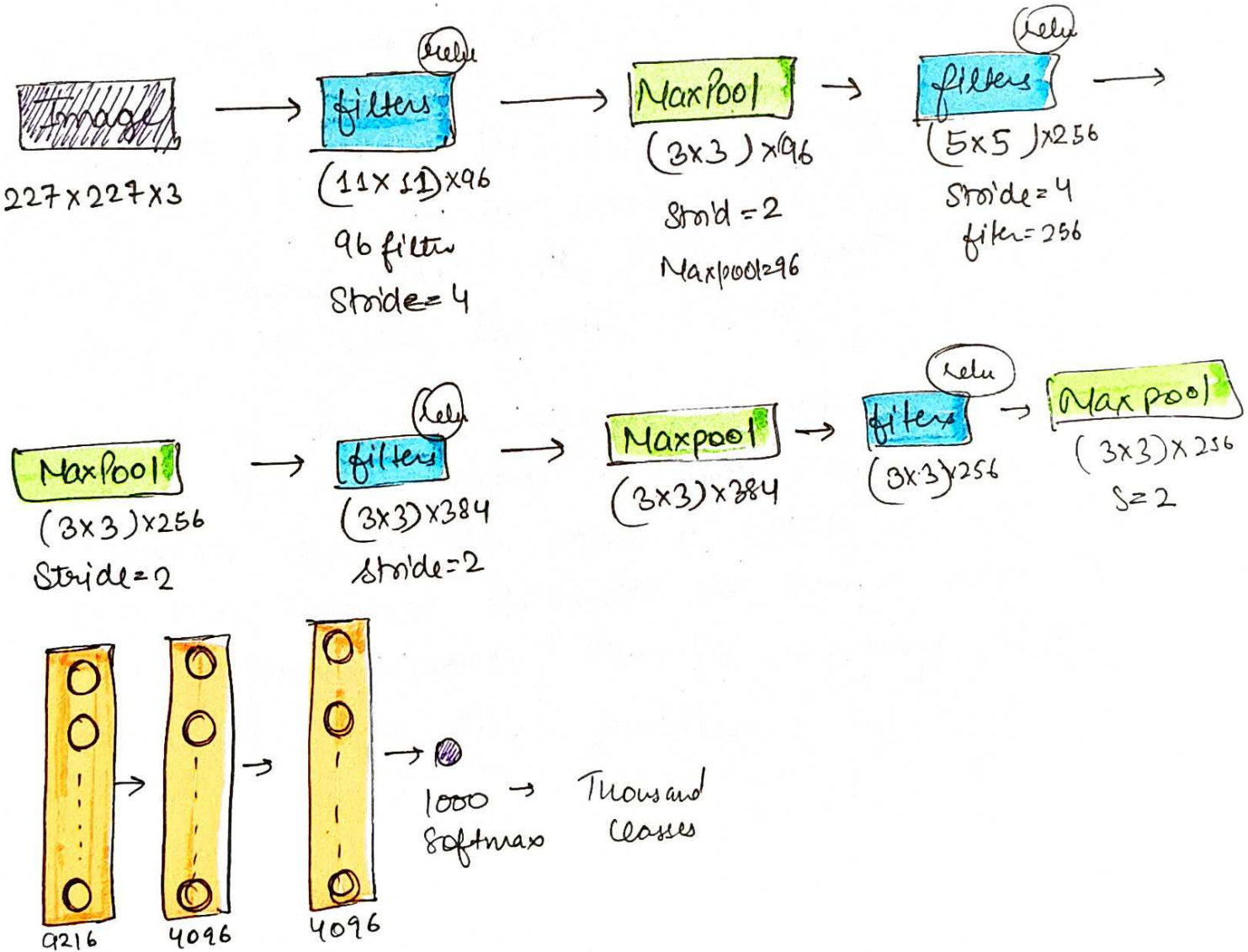
2012 → 16% → DL Model → GPU

↳ relu

↓

ALEXNET

AlexNET



2013 → ZFNET → 11.7%.

2014 → VGGG → 7.3%.

2015 → GoogleNET → 6.7%

2016 → ResNET → 3.5%.

Idea of Pretrained Model

VGGNET, RESNET → ImageNET → 1000 categories

If already have these model which is trained on 1000 categories of data then why we train our model and collect bunch of data.

Transfer Learning

Problem with training your own model

1) Data hungry → Labelled → 10,000.... data → scrap from google
   ↳ man/labour for labelling data

2. lots of time → train model
   ↳ week
   ↳ days

Using pretrain models.

for eg:- VGG, RESNET etc..

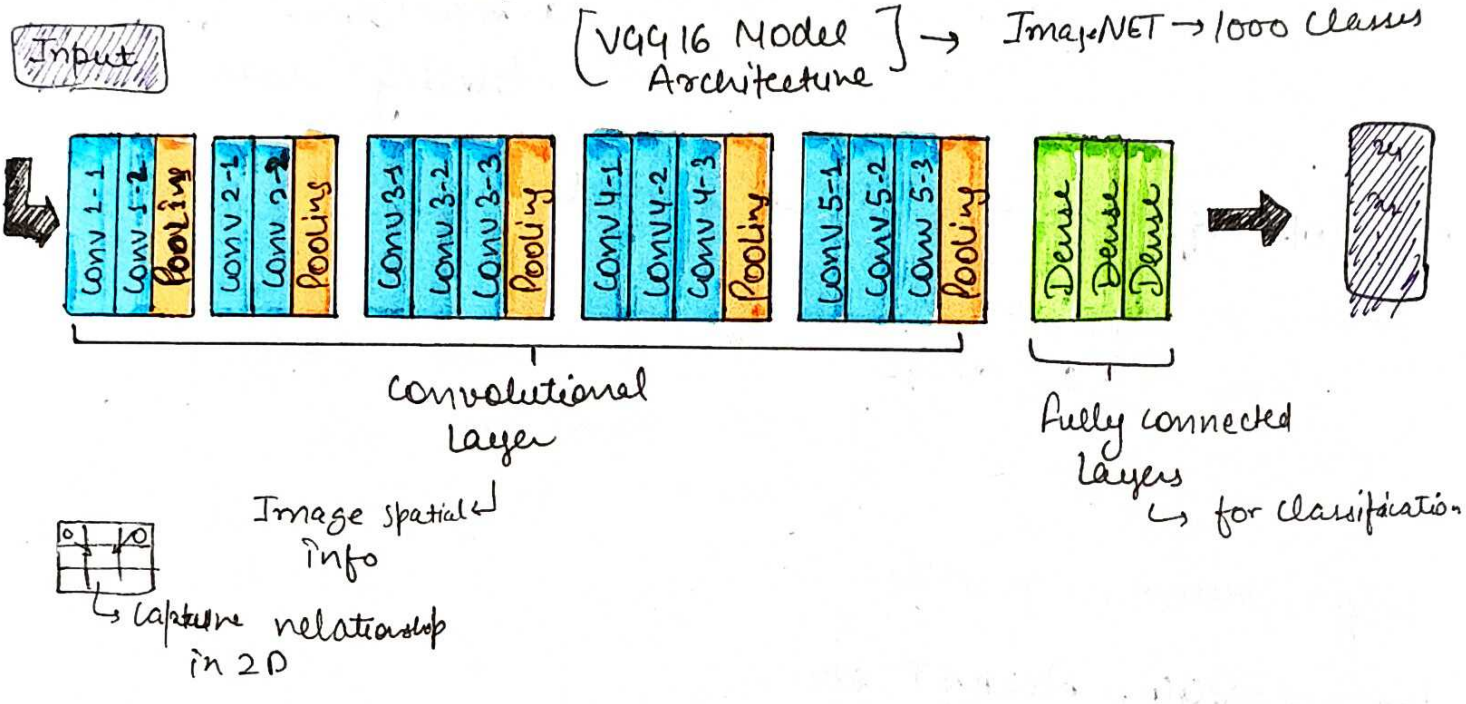
ImageNET dataset → 1.4 million → daily objects
1000 classes

Pretrain model train on this dataset and we easily use these Pretrain model.

→ what if we are making a project and data of the project is not available in ImageNET dataset. So, pretrain model is not trained on our required dataset. We cannot able to use pretrain model. Transfer learning was introduced for this problem.

Transfer Learning

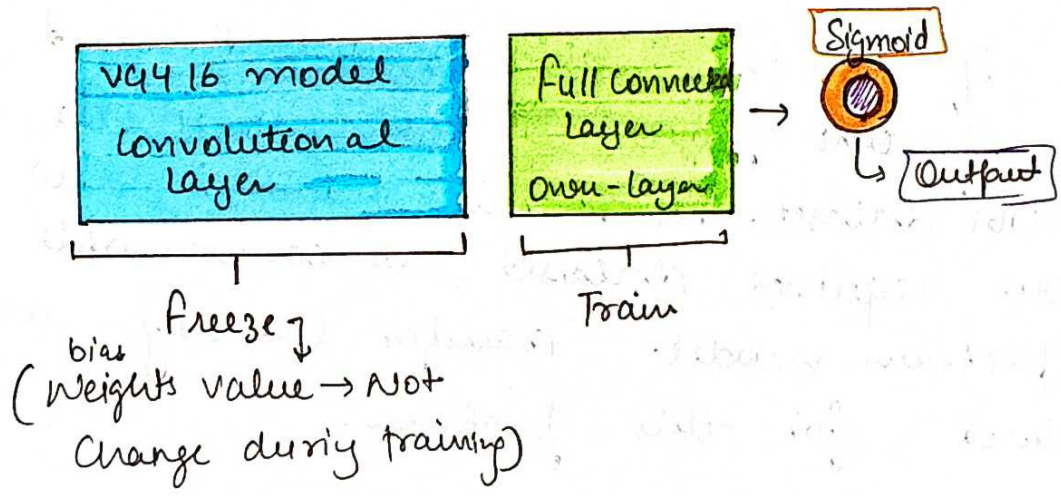
Transfer learning is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem.



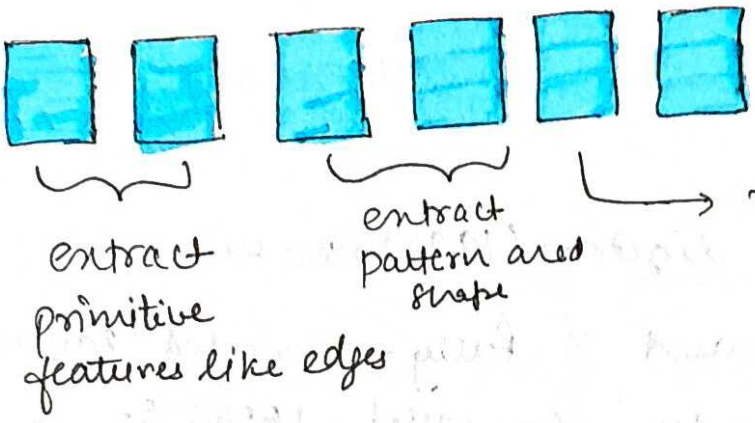
Using VGG16 Model → Cat/Dog classification

Assuming that cat and dog data are not present in ImageNET data.

Solution is



Why Transfer learning works



Any Image have → primitive features (edges)

So, Starting layers are common for any datasets
No need re-train the model. That why convolutional layer as it is use. We just change/train Fully connected layer.

Ways of doing Transfer learning

Feature extraction

↓

convolution layer ⇒ Freeze

Fully connected ⇒ Train layer

↓

When use?

your data (cat/dog)

↳ similar data present in ImageNET like any other animal.

Fine Tuning

↓

Last convolution layer unfreeze.

Add own dense layer.

↓

When to use?

your data (Phone/Tablet)

↳ but not present similar or related data in ImageNET.

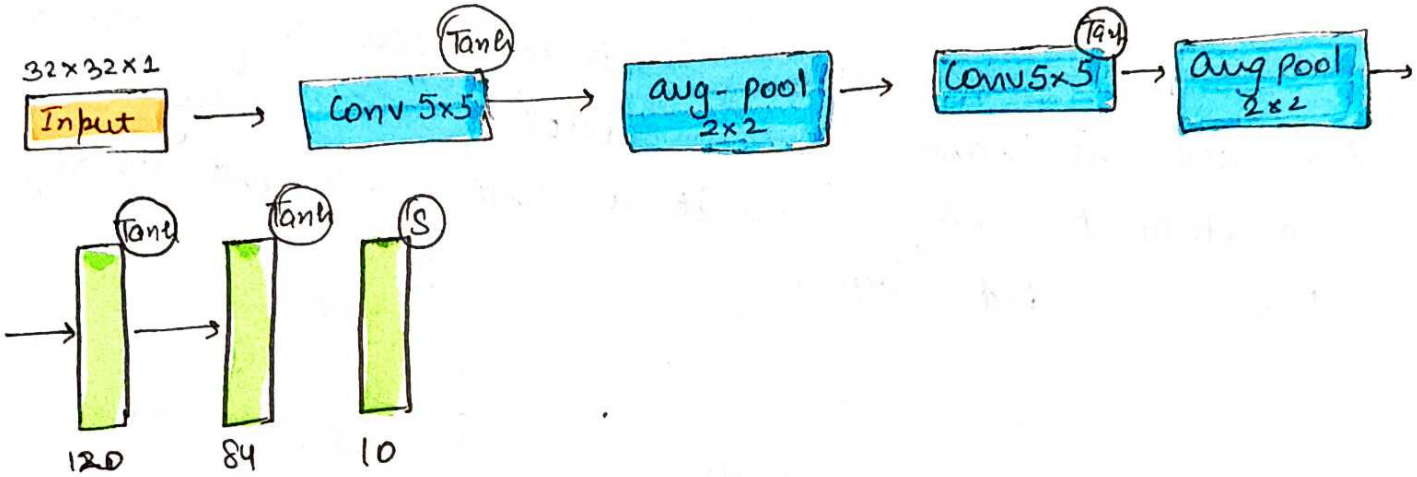
LENET

LENET → one of the most foundational and basic architecture.

↳ Handwritten digits → (1998) → Started.

2 convolutional layer and 3 fully-connected layers

Total → 5 (that's why also called LENET-5.



AlexNET

AlexNet → 2012 → 60M parameters

↳ 8 layers ⇒ 5 conv layer + 3 fully-connected

↳ first time use Relu activation function.

↳ first time train on RGB data.

Architecture → already defined → Pgno. → 98

Important Formulas

Input Dimensions:

Width (W): The width of the input image.

Height (H): The height of the input image.

Channels (C): The number of channels of color planes in the input image (eg 1 for grayscale, 3 for RGB).

Convolutional Layer:

Kernel size (K): The width and height of the convolutional kernel or filter.

Padding (P): The number of zeros added to the border of the input image to preserve spatial dimensions.

Stride (S): The step size which the kernel moves across the image.

Number of filters (N): The number of convolutional filter applied in the layer.

Output Dimension:

Width (W-out): $((W - K + 2P)/S) + 1$

Height (H-out): $((H - K + 2P)/S) + 1$

Channels (C-out) = N

Pooling Layer:

Pool size (P): The width and height of the pooling window.

Stride (S): The step size at which the pooling window moves across the input.

Output Dimension

Width (W_out) : ((W - P)/S) + 1

Height (H_out) : ((H - P)/S) + 1

channel (C_out) : C (remain same)

Parameter Formula

Convolutional Layer

$$\text{Parameters} = (k_h \times k_w \times C_{in} + 1) \times C_{out}$$

where:

k_h = Kernel height

k_w = Kernel width

C_{in} = No. of input channel

C_{out} = No. of output channel

1 represent the bias term for each output channel (if used)

Fully-Connected Layer:

$$\text{Parameters} = (n_{in} \times n_{out}) + n_{out}$$

where:

n_{in} = No. of input

n_{out} = No. of output

The second term for the bias unit.