We can use the same techniques we discussed during the training of simple neural networks to prevent overfitting like—

(a) Using regularization (L1 & L2)
(b) Using dropout layers
(c) Using early stopping
(d) Using batch-normalization layer.
(e) Give more data (can be generated using image augmentation).

# Transfer Learning

Transfer Learning means taking a model trained on one task (i.e., classifying 1000 ImageNet classes) and reusing it for a new task. (eg. classifying 5 types of flowers).

Types of Transfer Learning :

(a) Fixed feature extractor ⇒ Freeze the pre-trained model and use it just to extract features.
(b) Fine-tuning ⇒ Unfreeze a few top layers and retrain slightly for better adaptation.

| Strategy | ~~Adapt~~ Outcome | When to Use ? |
|---|---|---|
| → Feature extraction | Pre-trained model is frozen. Only a new head (classifier) is trained. | Smaller dataset similar to original dataset. |
| → Fine-tuning | Pre-trained model is partially unfrozen and retrain few top layers. | Medium-sized dataset. Slightly different task. |

**Ques.** Why do we need to make use of pre-trained models?

Ans → Training a deep neural network from scratch requires :
(a) A very large dataset (millions of images)
(b) Weeks of computation (even with GPUs)
(c) Risk of overfitting (if dataset is small).

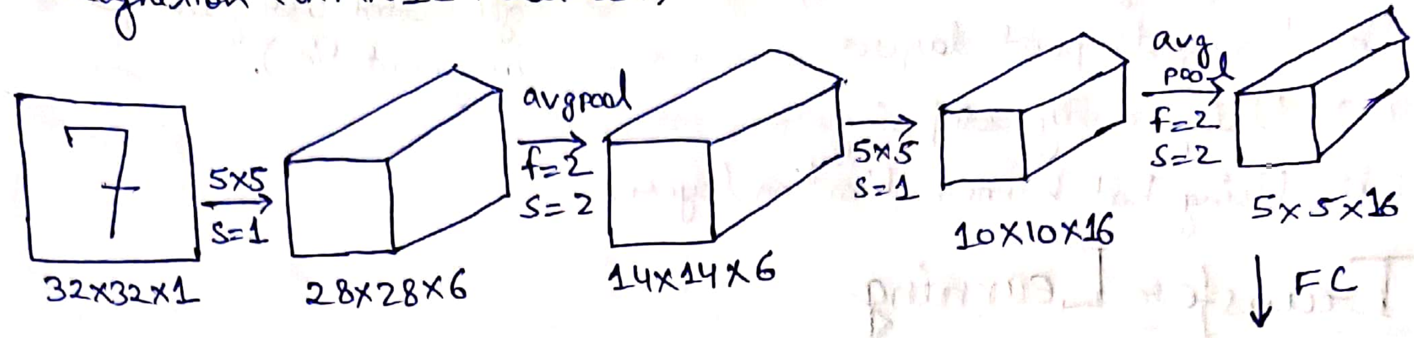
A pretrained model is a neural network that ~~a~~ has already learnt to extract features like :
(i) Edges (ii) Textures (iii) Shapes (iv) Object parts (v) Full objects)

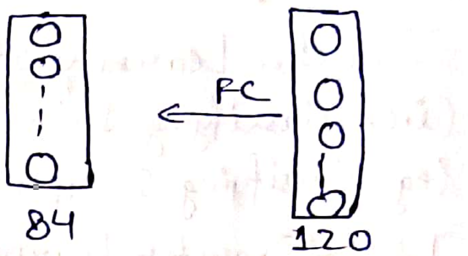
Embeddings are feature representations learned by a model.
We can reuse these pre-trained features instead of learning from Scratch.

Scanned with OKEN Scanner

# LeNet-5

It was developed by Yann Le Cun in 1998. and was used for hand-written digit recognition (on MNIST dataset).



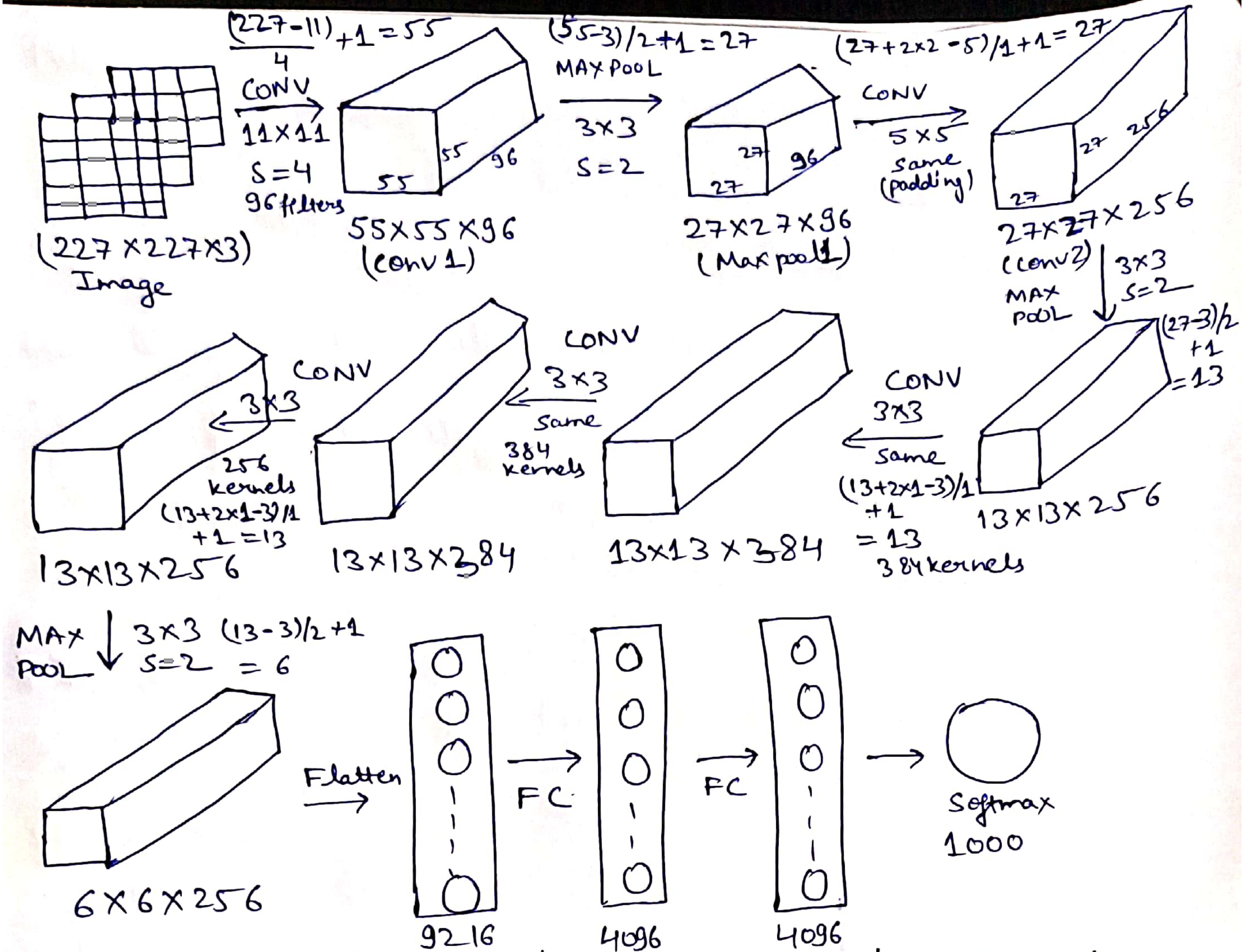




| Layer | Type | Output Shape | Details |
|---|---|---|---|
| Input | Gray scale Image | 32X32X1 | Grayscale Image |
| C1 | Convolution | 28X28X6 | 6 filters, 5X5 kernel, stride=1 |
| S2 | Subsampling (Avg Pooling) | 14X14X6 | 2X2 pooling |
| C3 | Convolution | 10x10x16 | 16 filters, 5X5 kernel, stride=1 |
| S4 | Subsampling (Avg Pooling) | 5X5X16 | Avg Pooling |
| C5 | Fully connected | 1X1X120 | 120 filters, (5x5 kernel fully connected to S4). |
| F6 | Fully connected | 84 | Dense Layer |
| Output | Fully Connected | 10 | For 10 classes (digits 0-9) |

The activation function used were tanh/sigmoid. It was a pioneering model that inspired modern CNNs and was trained on CPU back in 1998. At that time, no regularization and batch normalization was used.

# AlexNet

It was developed by Alex Krizhevsky, Ilya Sutskevar, Geoffrey Hinton in 2012. This model won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 with top-5 error rate of 15.3%, outperforming second best (~26%). It used ReLU, data augmentation, dropouts, and GPU parallelism (trained on 2 GPUs). It paved the way for better architectures like VGG, ResNet, etc.




| Layer | Type | Output | Details |
|---|---|---|---|
| conv 1 | convolution | 55X55X96 | 11X11 kernel, s=4 |
| Max pool 1 | Pooling | 27X27X96 | 3X3, stride = 2 |
| conv 2 | convolution | 27X27X256 | 5X5 kernel |
| Maxpool 2 | Pooling | 13X13X256 | 3X3, stride = 2 |
| conv 3 | convolution | 13X13X384 | 3X3 kernel |
| conv 4 | convolution | 13X13X384 | 3X3 kernel |
| conv 5 | convolution | 13X13X256 | 3X3 kernel |
| Maxpool 3 | Pooling | 6X6X256 | 3X3, stride = 2 |
| FC 1 | Fully Connected | 4096 | with ReLU + dropout |
| FC 2 | Fully Connected | 4096 | with Relu + dropout |
| FC 3 | Fully Connected | 1000 | Final layer (Imagenet classes) |