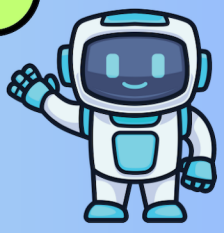
FUNDAMENTALS OF DEEP LEARNING

The concept of Transfer Learning

that's easy work lol

TRANSFER LEARNING MODEL

i can't handle these images :-(



STANDARD CNN MODEL

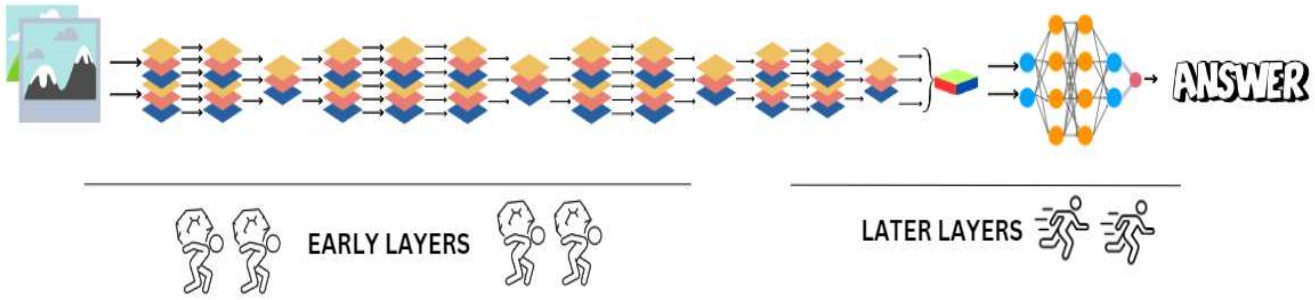
Shivang Kainthola

The first problem : Early Layers

⟶ In a CNN, the convolutional layers detect features from the images.

⟶ The early layers of a convolutional neural network capture very primitive, basic features like edges, corners from the images.

⟶ The later layers capture more advanced and complex features which will contribute particularly to the classification.



⟶ In the process of training CNNs with backpropagation, since the partial derivatives travel backwards the later layers are updated first, while the early layers are updated last, often suffering from vanishing gradient problems.

Early Layer	Later Layers
⟶ Detect basic features	⟶ Detects complex features relevant to the specific task
⟶ Difficult to train	⟶ Trained easily

The second problem : Big datasets, lots of images

⟶ Training convolutional neural networks to be able to classify or work on large number of images, requires for it to be trained on large number of images.

⟶ For a CNN to be able to handle complex images, it must be sufficiently deep i.e. it must have a sufficient number of layers, and this deep CNN must be trained on a lot of images.

⟶ To train a CNN with :

1) Large dataset, lots of complex images

2) Large number of layers (and thus large computations)

We need :

1) A lot of computation resources (Memory, GPU, CPU)

2) A lot of training iterations, large fitting time

3) A lot of time to experiment with different hyperparameters

To solve the first problem about early layers :

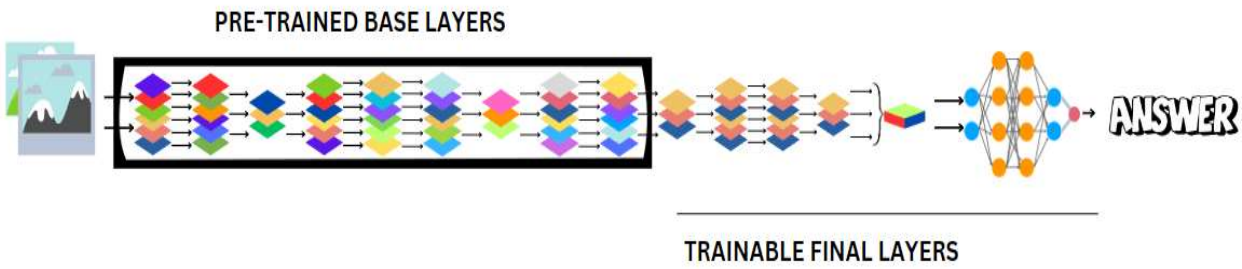
We can save the early layers of a pre-trained convolutional neural network, since they retain the ability to detect generic features.

Then, we can re-train the later layers (or add our own) according to our specific task, which is referred to as fine tuning.

To solve the second problem, using the solution for the first problem :

- Train a deep convolutional neural network on a large dataset like ImageNet, which contains over 14 million labelled images.
- Freeze the early layers of this deep CNN, which have been painstakingly trained to detect the basic features from a large number of images.
- Load the early layers of this CNN, which are already trained on a large number of images, and add your own later layers.

This is called Transfer Learning.



Just like save-games, but for convolutional neural networks.

⟶ The early layers have been pre-trained on a large number of images, and can generalize to most common tasks, unless your problem involves unusual or very specific images/requirements.

⟶ Using Transfer Learning, all you need to do add your own later layers and train the model with comparatively very less resources.

⟶ There are many pre-trained CNN models and family of models which were trained on large image datasets like ImageNet (which has 14.2 million images), namely :

LeNet, AlexNet, Inception, VGG16/19, MobileNet, ResNet, EfficientNet, Xception

⟶ You can download any of these models, and fine tune them for your image classification tasks. Similarly, you can download the saved models of many CNN-related projects and instead of starting from scratch.

⟶ Let us see some of the most popular CNN transfer learning models :

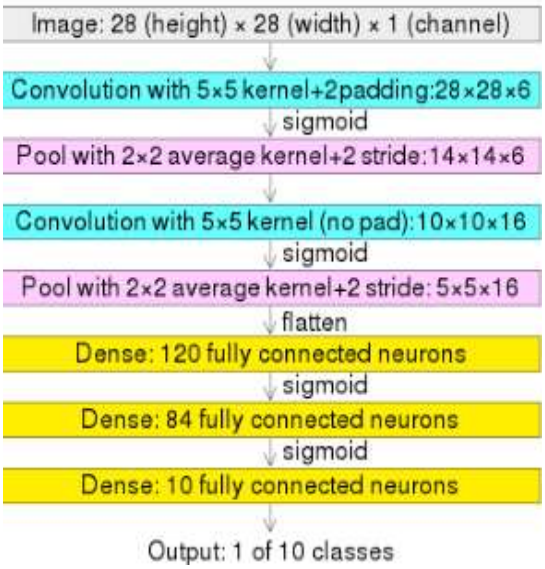
LeNet

⟶ One of the earliest CNN architectures.

⟶ Proposed by Yann LeCunn et al. in 1998

⟶ Used on the MNIST dataset (32x32 black and white images of handwritten digits)

⟶ Structure of LeNet :



Convolutional Layers (C1, C3, C5):

- Apply multiple small filters (e.g., 5x5) to the image, detecting edges and low-level features.

Pooling Layers (S2, S4):

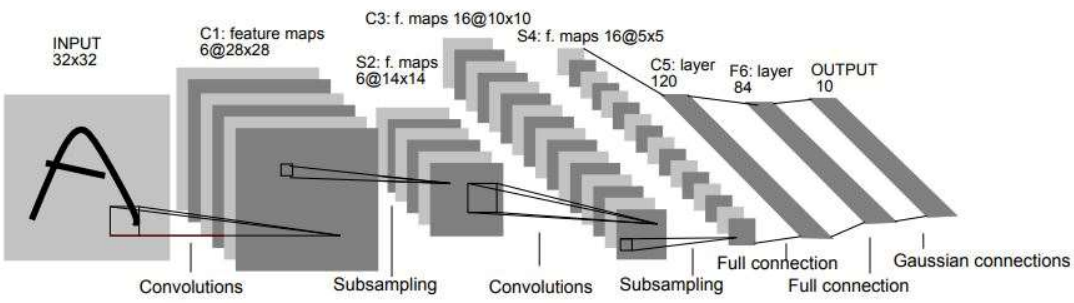
- Reduce the size of the feature maps by average pooling.

Fully Connected Layers (F6):

- Flatten the remaining feature maps into a single vector, perform complex computations on the extracted features.

Output Layer:

- Uses a Softmax function to classify the image based on its probabilities of belonging to categories (e.g., digits 0-9).



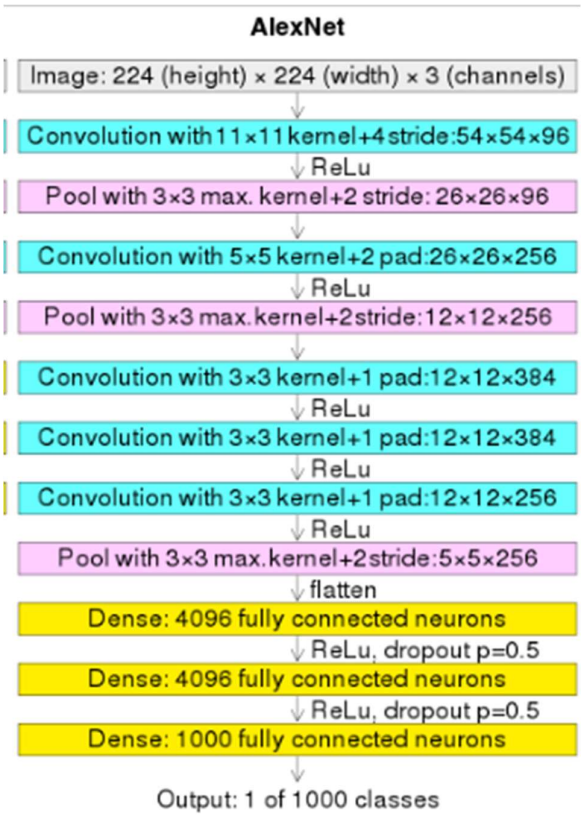
https://production-media.paperswithcode.com/methods/LeNet_Original_Image_48T74Lc.jpg

AlexNet

⟶ Made in 2012 for the ImageNet Large Scale Visual Recognition Challenge .

⟶ Designed by Alex Krizhevsky in collaboration with his colleagues

Ilya Sutskever and Geoffrey Hinton, who was Krizhevsky's Ph.D. advisor at the University of Toronto.



⟶ It was a major breakthrough in the field of deep learning.

It achieved superhuman performance on the ImageNet image recognition task, surpassing previous methods by a significant margin.

⟶ It was tasked with predicting correct labels from among 1000 classes, using the ImageNet dataset which has almost 1.2 million color (RGB) images of size (224x224) each.

⟶ Consists of 5 convolutional layers followed by 3 fully connected layers, and gave an error rate of 15.4%, the least in the competition, where the second best error rate was around 26.2%.

Inception

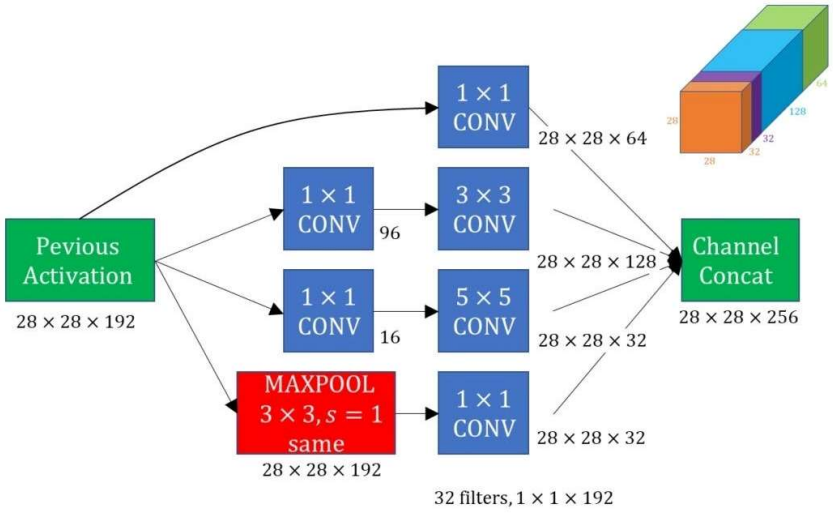
⟶ Introduced by Szagedy et al - Google researchers in 2014.

⟶ It had multiple principles behind it :

1) To efficiently capture features at multiple scales within the same layer of the network

2) Use 'inception modules' - composed of convolutional layers with varying kernel sizes (1x1, 3x3, 5x5) along with pooling operations.

An inception module looks like :



https://miro.medium.com/v2/resize:fit:1400/0*W5Dcz-bPesqqmEWO.jpg

This enables parallel processing of input data with filters of different sizes, allowing the network to extract a diverse range of features simultaneously.

ResNet

→ Introduced by Kaiming He et. al (2015)

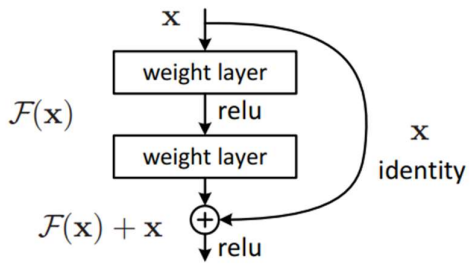
→ ResNet stands for Residual Network

→ The motivation behind ResNet :

1) At the time, although deeper neural networks were being built, they all suffered from problems similar to the vanishing gradient problem.

2) In typical neural networks, each layer is expected to directly learn a mapping from its input to its output.

→ ResNet introduces the idea of learning residual functions F(x) instead of directly learning the underlying mapping output H(x) for input x.



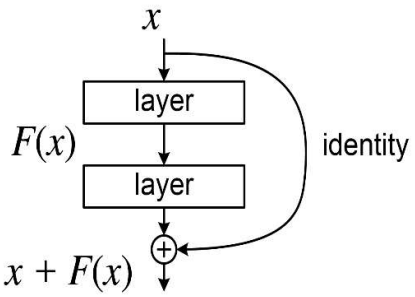
F(x) = H(x) – x

The network learns to approximate and minimise the residual mapping

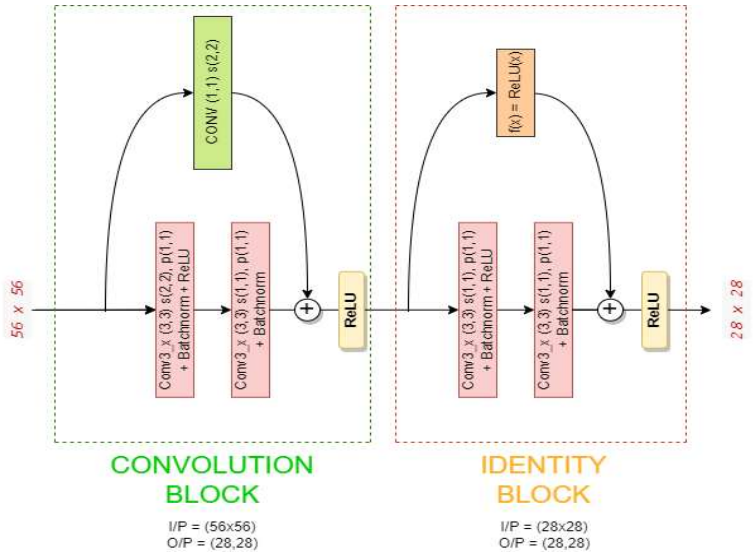
→ Essentially, the network learns to predict and minimise the difference between the desired output and input of that layer.

→ Working :

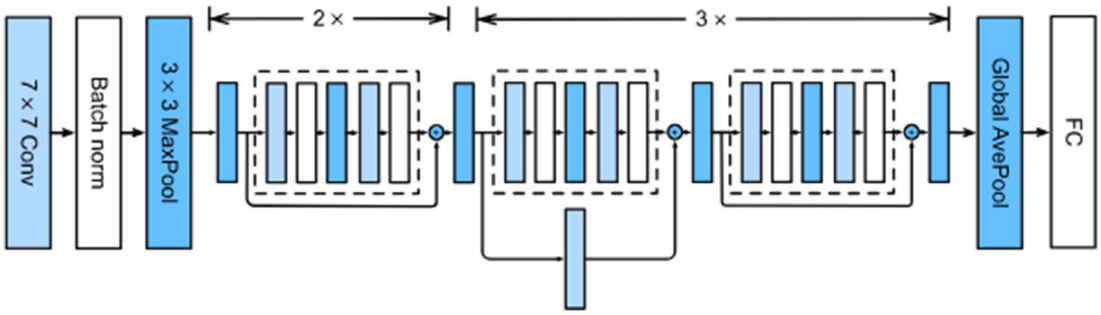
The working of ResNet is facilitated by adding shortcut connections called skip connections which allow the the input to bypass one or more layers and be added directly to output of deeper layers.



https://upload.wikimedia.org/wikipedia/commons/b/ba/ResBlock.png



These skip connections allow alternate paths for gradient flow, so even though it diminishes at some points, it can still reach deeper layers.



https://d2l.ai/_images/resnet18-90.svg

VGG

⟶ VGGNet, stands for Visual Geometry Group, named after the group that developed it at Oxford University.

⟶ It was developed by K. Simonyan and A. Zisserman.

⟶ Has two versions:

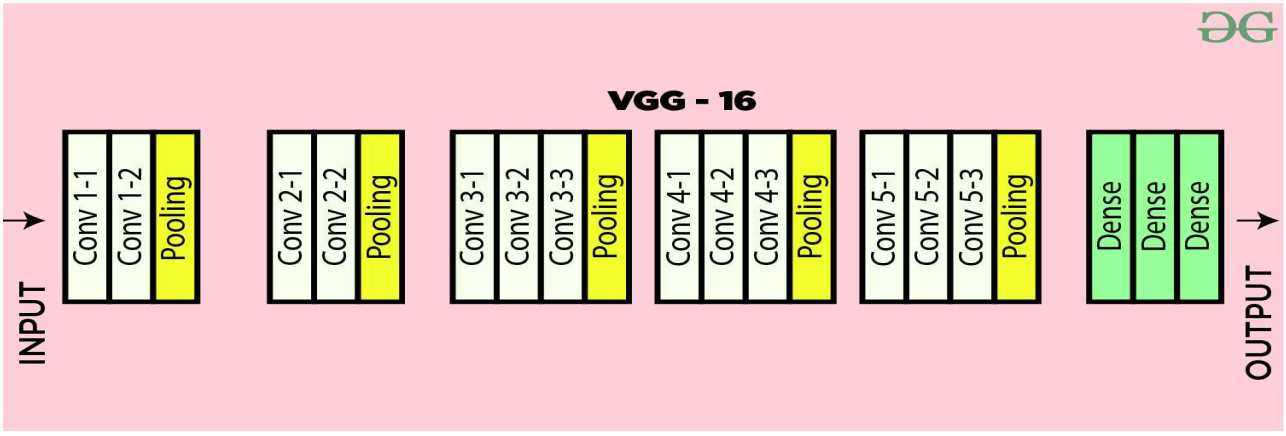
1) VGG-16 : Has 16 convolutional layers followed by 3 fully connected layers.

2) VGG-19 : Has 19 convolutional layers followed by 3 fully connected layers.

⟶ VGG uses small 3x3 filters in its convolutional layers throughout the network. This keeps the architecture relatively simple and easy to understand.

⟶ Incorporates max pooling layers in between convolutional layers to reduce the image size and computational complexity

⟶ Despite its simplicity, VGG models achieve strong performance on typical CNN tasks like image classification.



https://media.geeksforgeeks.org/wp-content/uploads/20200219152327/conv-layers-vgg16-1024x450.jpg

MobileNet

⟶ First introduced in 2017, it was developed by a team of researchers at **Google.**

⟶ It has three versions MobileNetV1, MobileNetV2 and MobileNetV3.

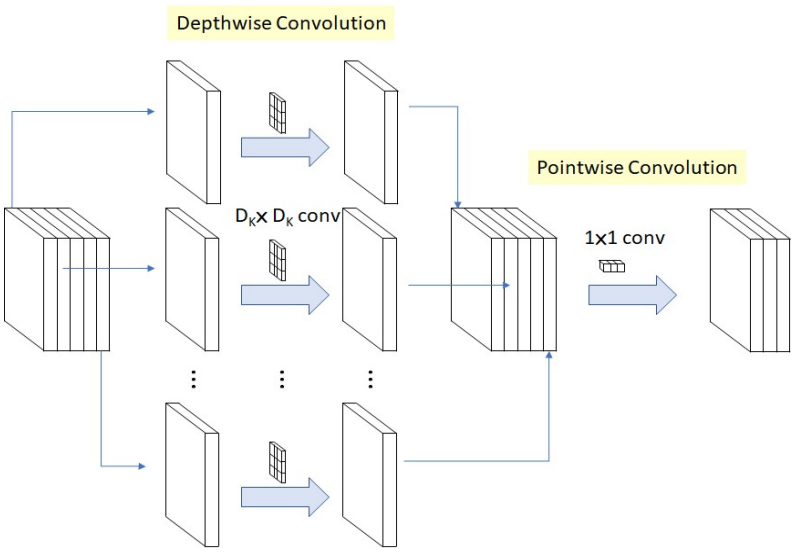
Further, each version, espcially V3,has different models with small and large number of parameters.

⟶ It is specifically designed for mobile and embedded devices where computational resources are limited

⟶ Prioritizes reducing the number of computations required for processing images, by a process called depthwise separable convolutions.

⟶ MobileNet separates this process into two steps:

- **Depthwise convolution:** Applies one filter per input channel, extracting features for each channel independently.
- **Pointwise convolution:** Uses a 1x1 filter to combine the features from the depthwise step, reducing computational cost.



⟶ It is aimed towards achieving the objectives on CNNs on mobile devices with less computations.

EfficientNet

⟶ It was introduced by Mingxing Tan and Quoc V. Le from Google Research in their 2019 paper "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks."

⟶ It uses a scaling method that uniformly scales all dimensions of depth/width/resolution using a *compound coefficient*.

⟶ EfficientNet relies on MobileNetV2's building blocks, leveraging depthwise separable convolutions for efficiency.

⟶ EfficientNet is a family of seven different models (EfficientNet-B0 to B7) with varying complexity levels.

Each model is created using the compound scaling method, allowing you to choose the one that best suits your needs based on the available resources and desired accuracy.

Vision Transformer (ViT)

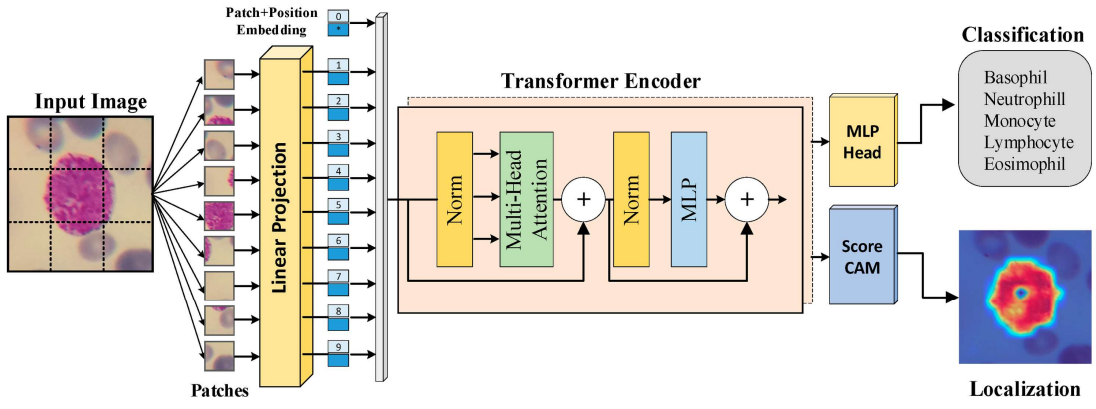
→ Transformers were introduced in 2017, in a paper "Attention Is All You Need" and have found widespread use in natural language processing.

In 2020, they were adapted for computer vision, yielding Vision Transformer.

→ Vision Transformer does not extract features from the image by convolutions, instead it utilizes the Transformer architecture, originally designed for natural language processing (NLP), for image recognition.

→ Instead of convolutions and pooling etc. , ViT breaks down an image into smaller patches. Each patch is converted into a vector using a linear projection.

→ The resulting sequence of vectors is fed into a standard Transformer encoder, similar to how words are processed in NLP tasks.



https://pub.mdpi-res.com/diagnostics/diagnostics-13-02459/article_deploy/html/images/diagnostics-13-02459-g001.png?1690196193

Thank You !

If you found this article helpful, please do leave a like and comment any feedback you feel I could use.

While I’m working on the next project, I’ll be over at :

www.linkedin.com/in/shivang-kainthola-2835151b9/

shivang.kainthola64@gmail.com

https://shivangkainthola28.medium.com/

https://github.com/HeadHunter28