Introduction to Convolutional Neural Networks (CNN) and Their Applications

Murt Sayeed
CSO 135
May 14, 2021

Background on Neural Networks:

In this short paper, I will review the Convolutional Neural Networks (CNN) model and its various applications. A quick recap on Neural Networks, a Neural Network is a collection of multiple layers consisting of an input layer, output layer that produces the prediction, and hidden layers in between the two with activation functions. Each neuron is computed by the linear function represented by the equation Wx+b=Y [7]. Then, we apply a small propagation for the output each one called the forward pass. Afterwards, we initialize the computation of 1) all the weights in a uniform small random distribution and 2) biases are initialized to zero. We determine the loss function that tells us the difference between what we predicted (y-hat) and what the actual output should be (y) [7]. Then, we use Graduate Descent with the method of back propagation to take the gradient and compute the best direction for each connection in which we need to update the loss function via minimum direction [7]. This is also referred to as Stochastic Gradient Descent. We update the gradient each time by applying learning rate (alpha) [7].

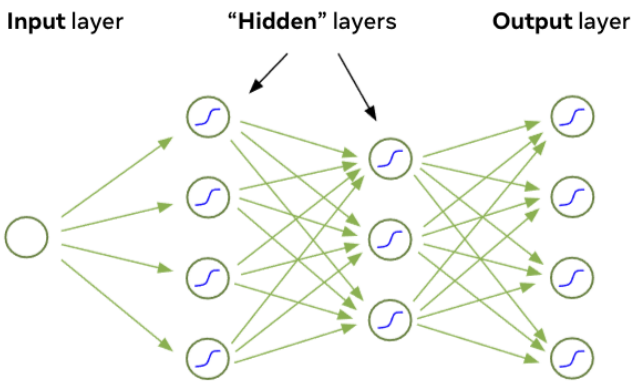

Figure 1

How our Visual Cortex Works:

I want to start by drawing a biological analogy between CNN and how our brain works. Human beings have different regions in the brains handle different tasks visualizing shapes. We have areas V1 to V4 that help with simple detection of edges or corners [8]. Other areas combined with previous area creates intermediate visual forms. Then, the brain has the A/T area in which high level applications are created such as trees or faces [8].

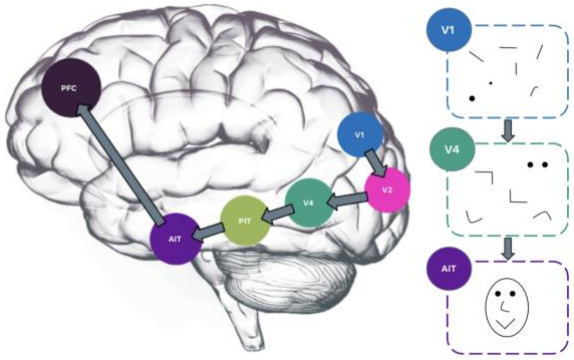

Figure 2

After is the prefrontal cortex (PFC), which handles categorical judgement and decision-making [8]. Now, the question arises, how we can replicate this through a machine? This is necessary to replicate when we take an image and assign weights/biases to many aspects of the same image in order to get classify whether such image is cat, tree or a car, for example.

Computer Vision Before and After 2012:

Let's explore how the computer vision looked like before the era of deep learning. Before 2012, scientists and engineers had to use pages worth of math and formulae on feature extraction and engineering in order to classify an image of a cat [2]. Example, to determine a cat's face, we need to create or identify a circle in the center for a face, then the distance between the two eyes and the proportion between various geometric shapes, would determine at the end the face of cat image.

In 2012, Alex Krizhevsky created 'AlexNet' which feed image pixels directly into a neural network [2]. This technique produces direct output of neural net, instead of doing any

feature extraction and engineering. AlexNet not only over-simplify the technique but also had much better accuracy [2]. Since then, every computer vision problem is based on deep learning for a best performing model.

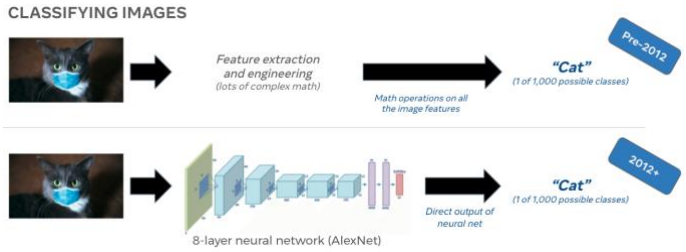

Figure 3

Convert Raw Pixels to Some/Reduced Output:

The prediction described as "AlexNet' is a black box that will take an image, which is a collection of numbers (i.e. a 200x200 pixel image will produce 40,000 numbers), then predict an image out of a 1000 possible options. We need to predict 1000 numbers that correspond to the relative probabilities that the picture feed into the black box cane be identified as that object. Thus, we need to convert 40,000 numbers into 1000 numbers.

Building a Convolution Layer:

We begin with an image with dimensions 32 as height, 32 as width, and 3 as depth in colors. Then, we would pick a filter or kernel of 5x5x3 that is a smaller size than the previous image. This filter will perform an operation called convolution, meaning slide the filter over the image spatially [1,3]. Based on the values of the filter and the values of the image, we will get a small set of numbers called convolved feature. The step size or stride is a way of moving or sliding one step at a time to compute each filter application on each image pixel [1,3]. It is important to extend the filter to the same depth as the original image volume - in this case three - so that we would have applied an operation on the entire set of inputs.

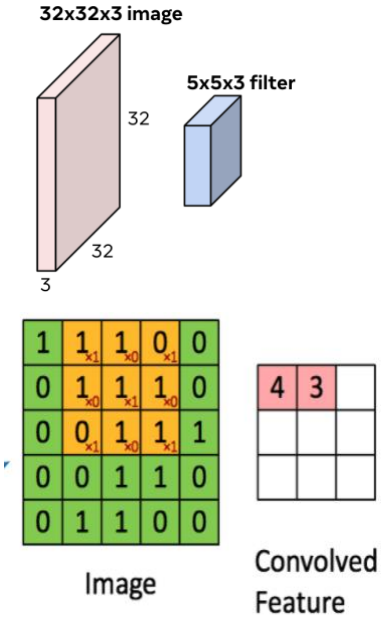

Figure 4

Once we compute the filter, it produces a feature. We start with one number, resulting from multiplying the values in a small 5x5x3 chunk of the original image and the filter. We take this number and the collection of numbers from filters (weights from filter), and apply it to the original image with biases: W^Tx+b. This formula is the same neural network model, but it has more numbers in matrices. This same one number will convolve or slide over all the spatial locations, resulting in an activation map of 28x28x1. We need to have more features to analyze the image, i.e. having a second filter would produce a different activation map. We could potentially apply six separate filters (5x5x3) and get 6 different activation maps (28x28x6). Thus, we have a new "image" of size 28x28x6, from the initial starting point of 32x32x3 dimension. In this case, convolution filters applied to the original image with a depth of three produces a new image with a depth of size. If we were to follow Fully Connected (FC) layer in original neural network [1,3], such that every input in the original image of 32x32x3 would be connected every output in the activation map image of 28x28x6, we would have 14.5M parameters or 14.5M multiplications [(32x32x3)x(28x28x6)] where every pixel in the original image is connected to every pixel on the activation map image. With convolution filters, we would get 450 parameters [(5x5x3)x6] or 350K multiplications [(5x5x3)x(28x28x6)]. We use fewer parameters with convolution, which makes it more efficient to do multiplication for the computers [1].

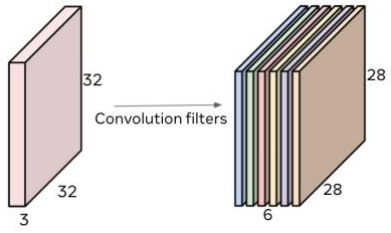

Figure 5

It is important to remember that there is a mathematical construct that is flowing in the network as we progress through it. We started with matrices, changed the shapes of the matrices, and finally predicted a vector size of 1000x1. This mathematical construct could change throughout the network or a generalization of an n-to-m dimensional vector mapping that

incudes matrices of different dimensions, vectors and scales [10]. This framework is called tensor flow, where it deals with manipulation of layers and graphs of neural networks [10].

Convolutional Neural Network (CNN):

The CNN is a sequence of convolutional layers described above. We start with an image 32x32x3, apply filters 28x28x6, then apply more filters to get 24x24x10, and keep going. Between each convolutional layer, we have activation functions ReLU, Sigmoid and Tanh [4].

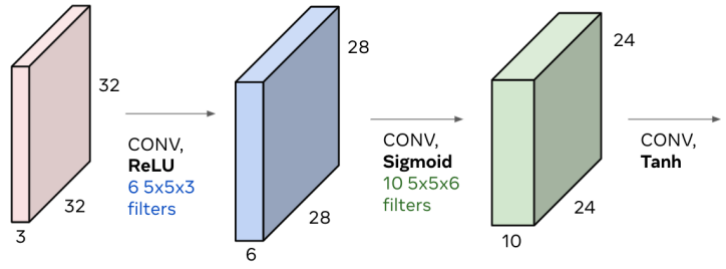

Figure 6

We can develop an algorithm to compute the Computer Visual problem by following these steps [5]. First, we can start by common basic features, represented by small filters. Then, we zoom out and try to find larger or more complex features using the features learned in the previously step. We repeat with zooming out and try to find more complicated features. Afterwards, we take the features and run them through few Fully Connected (FC) layers to get final results [1]. This is a common practice in many different CNN applications.

The zooming out process is nothing more than supplementary sampling where we take additional pixels and turn them into one pixel. Let's say we start with 224x224x64 volume and we shrink it by half to 112x112x64 volume but with the same depth. This process is called 'down-sampling' and the most common approach is 'Max Pooling' [2,3]. It looks at every slice of the image and picks a maximum value to be saved in the pooled slice. The activation map created by the filters are

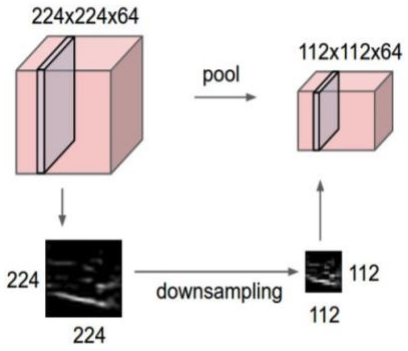

Figure 7

really maximum, meaning the data is saved across many different pixels and these pixels carry the value by showing the difference between the two [3, 4]. If we used the averages method, we would have lowered the value that the activation map creates, and it would not be as effect as compared to using the maximum value [3]. If we use the minimum value, we will completely negate signals that the activation map is giving us [3]. In practice, max pooling has shown better results.

Putting it All Together:

We will take a look at an example of VGG13, a common network with 13 layers of parameters. It has several chunks of input where we start by applying two convolution layers to an original image with filter size 64x64, and then we zoom out via max pool. We continue with larger filters of sizes 128x128, 256x256, 512x512, and 512x512 with max pool applied after each filter. Then, we take the output and pass it through several full-connected (FC) layers, ending with FC-1000. After which we apply the softmax function to turn it into normalize probability.

Figure 8

The reality of CNN is that it can be useful for other applications such as speech, sounds, and signal processing [5]. For example, we can read ECG signals across the sound signals to know what the spoken text was or to classify otherwise, we can apply single dimensional (1D) convolutions. Instead of having these filters on sections of an image, we can apply a filter to section the sound and reduce it by one dimension. We can also zoom out in between sections to get a prediction. In this illustration (right), we have

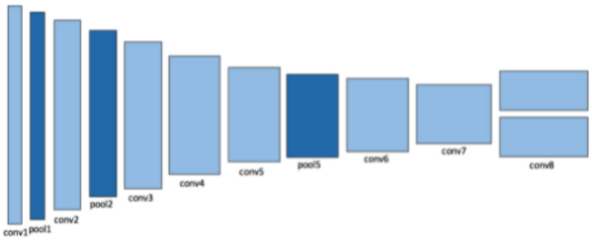

Figure 9

an architecture of a network called soundness, which consists of a convolution layers and pooling layers side by side. It's a same idea as we discussed above with cat image but in a different dimensional format.

One note here is that training these networks can be difficult. One of the advantages of CNN is that it gives us pre-trained representation or collection of weights to use in other applications. Thus, there is a technique called 'transfer learning', where we can transfer the learning of one training of neural network into another problem or even a different dataset that would be more applicable to other datasets [11]. For example, we can take pre-trained VGG13 network on somebody's dataset, and apply it to our dataset to get best predicted capability of the network. When we have a big dataset, we can train the entire network since we have enough parameters for the network to learn. However, if we have a small dataset, we could freeze all the weights received in the previous training and only train the last two layers that includes the last FC and softmax layers. This would ensure our small dataset network uses the learnings from the previous training and applies it to the current problem. If we have medium dataset (right illustration), we can get additional layers trainable with some freeze layers from a previous training dataset. Thus, we can have a balance between what we want to use from the previous pre-trained network versus what we want to retrain for the current problem [11]. The rule is that the more training data we have, the fewer layers we freeze. This has practical considerations when training on GPU memory due to reduced bottlenecks. Most memory and compute power are used in the early layers of CNN, while most computationally intensive operations are used in early layer of Fully Connected (FC) section. This

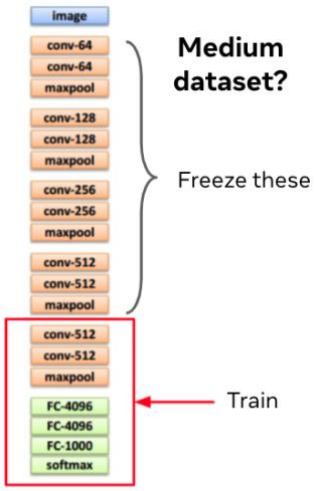

Figure 10

gives us an idea on how many parameters and how much memory it could take if we don't use the transfer learning method.

Another important thing to note is that these Computer Visual models are organized around benchmarks on public datasets that are being used to test these benchmarks to service the training dataset. The two common ones are ImageNet (Stanford) and COCO (Microsoft). The ImageNet has 10M hand annotated images, where 1M are with bounding boxes [6]. The COCO has 120K+ hand annotated images and also the shapes on these images are marked by their border so we can create some segmentation between them [6].

The deep learning industry has made huge progress in the last one decade (illustration below [6]). The biggest progress came from AlexNet in 2012 briefly described above. Then, we had VGG19 in 2014 and followed by ResNet. In recent years, there is an idea of retention and transformers that played a bigger role with previous Inception and ResNet models [6]. The parameters jumped from 61M in 2012 to 480M in 2020 [6], meaning the bigger the network the better the predicted value with reduced error. Based on such good progress in last decade, the future of CNN will be around Neural Topologies and Privacy Preservation [1].

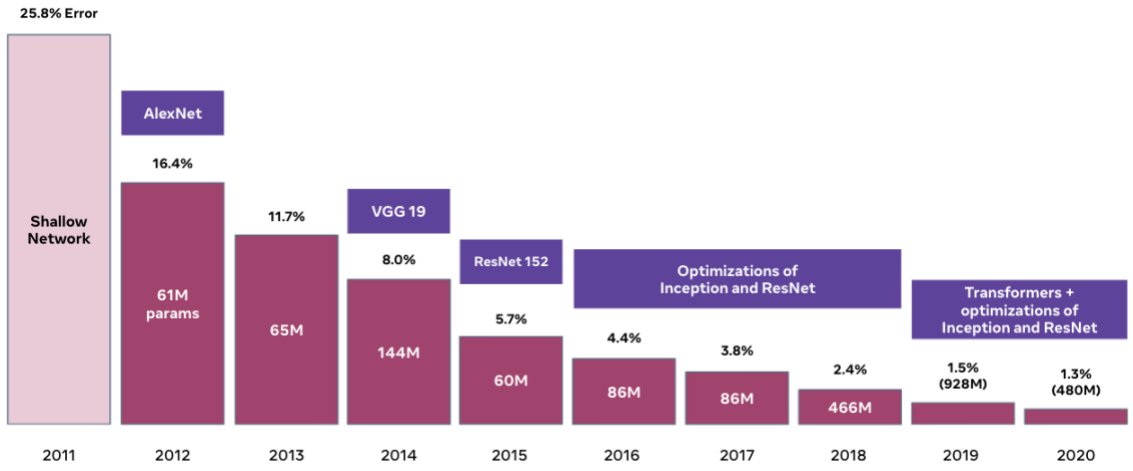

Figure 11

Reference:

[1] Wen Xu, Jing He, Yanfeng Shu and Hui Zheng (October 14th 2020). Advances in Convolutional Neural Networks, Advances and Applications in Deep Learning, Marco Antonio Aceves-Fernandez, IntechOpen, DOI: 10.5772/intechopen.93512.

[2] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press. https://www.deeplearningbook.org/contents/convnets.html

[3] Saha, S. (2018, December 15). A comprehensive guide to Convolutional neural Networks — the ELI5 way. Medium. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[4] Introduction to convolution neural network. (2017, August 21). GeeksforGeeks. https://www.geeksforgeeks.org/introduction-convolution-neural-network/

[5] Brownlee, J. (2020, April 17). How do Convolutional layers work in deep learning neural networks? Machine Learning Mastery. https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/

[6] Image Classification on ImageNet. (n.d.). Papers With Code. https://paperswithcode.com/sota/image-classification-on-imagenet

[7] Nielsen, M. (2018). Neural Networks and Deep Learning. Determination Press. http://neuralnetworksanddeeplearning.com/chap1.html

[8] Lindsay, G. W. (2020). Convolutional neural networks as a model of the visual system: Past, present, and future. Journal of Cognitive Neuroscience, 1-15. https://doi.org/10.1162/jocn_a_01544

[10] Asiri, S. (2019, November 11). Building a Convolutional neural network for image classification with TensorFlow. Medium. https://medium.com/@sidathasiri/building-a-convolutional-neural-network-for-image-classification-with-tensorflow-f1f2f56bd83b

[11] Koehrsen, W. (2018, November 26). Transfer learning with Convolutional neural networks in PyTorch. Medium. https://towardsdatascience.com/transfer-learning-with-convolutional-neural-networks-in-pytorch-dd09190245ce