Deep Learning for Connector Cooler Prediction

Lei Liu, Mukesh Mantan

Nov 10, 2018

1 Solution Approach for IOT Based Image Detection

1.1 Deep Learning

Deep learning refers to neural networks with multiple hidden layers that can learn increasingly abstract representations of the input data. This is obviously an oversimplification, but it is a practical definition for us right now. For example, deep learning has led to major advances in computer vision. We are now able to classify images, find objects in them, and even label them with captions. To do so, deep neural networks with many hidden layers can sequentially learn more complex features from the raw input image: The first hidden layers might only learn local edge patterns. Then, each subsequent layer (or filter) learns more complex representations. Finally, the last layer can classify the image as a cat or kangaroo.

2 Why is This Important

Image detection is a vital component in automation and security systems. Image recognition can be important in social automation or human-machine-interfaces such as an advertising bot that can look at images and extract attributes such as branding, color, quantity and could use that information to find something good to show you, something you are likely to buy. Thus image recognition is ideal for marketing in the future, don't underestimate it.

Image recognition is also important in image search whereby you use rich image content to query for similar stuff. Like in cooler photos where the system uses image recognition to categorize cooler images into things like redbull, nonredbull and so on so that you can quickly search things.

You can also use the concepts of image recognition in imaging such as inventory detection in cooler images to assist inventory management and so on. You can also use computer vision techniques to help blind people in their day-to-day interactions.

Image recognition algorithms can also be used in inventory systems to track stocks of mechandise or camera tracking. The same can be used to recover 3D information from multiple cameras such as in structure from motion (SFM).

Thus Image recognition and it's big brother image understanding are more powerful than voice or speech recognition technical wise but when it comes to applicability maybe not so much but it is still there, it is very important for machines to start understanding what the cameras are capturing.

3 Choice of Algorithms and the Best Fit

3.1 Implementation

- Initialize the model
- if we are using "channels first", update the input shape
- first set of CONV - RELU - POOL layers
- second set of CONV - RELU - POOL layers
- first (and only) set of FC - RELU layers
- softmax classifier

3.2 Algorithm

- initialize the number of epochs to train for, initial learning rate, and batch size
- initialize the data and labels
- grab the image paths and randomly shuffle them
- loop over the input images
  - load the image, pre-process it, and store it in the data list
  - extract the class label from the image path and update the labels list
- scale the raw pixel intensities to the range [0, 1]
- partition the data into training and testing splits
- convert the labels from integers to vectors
- construct the image generator for data augmentation
- initialize the model

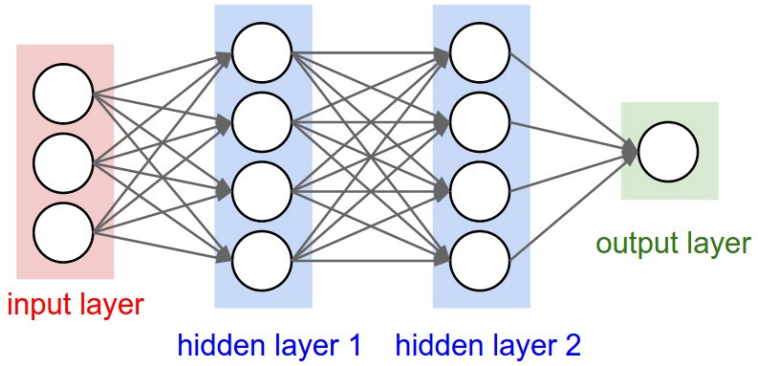
- train the network
- save the model to disk
- plot the training loss and accuracy

4 Set Up and Architecture

4.1 Convolutional Neural Network

Convolutional Neural Networks (CNN's) are multi-layer neural networks (sometimes up to 17 or more layers) that assume the input data to be images. By making this requirement, CNN's can drastically reduce the number of parameters that need to be tuned. Therefore, CNN's can efficiently handle the high dimensionality of raw images. Convolutional Neural Networks are very similar to ordinary Neural Networks from the previous chapter: they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer and all the tips/tricks we developed for learning regular Neural Networks still apply.

Figure 1: Regular Neural Network



ConvNet architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network.

4.1.1 Scale to Full Images

Recall: Regular Neural Nets. As we saw in the previous chapter, Neural Networks receive an input (a single vector), and transform it through a series of hidden layers. Each hidden layer is made up of a set of neurons, where each neuron is fully connected to all neurons in the previous layer, and where neurons in a single layer function completely independently and do not share any connections. The last fully-connected layer is called the ?output layer? and in classification settings it represents the class scores.

Regular Neural Nets don't scale well to full images. In some cases, images are only of size 32x32x3 (32 wide, 32 high, 3 color channels), so a single fully-connected neuron in a first hidden layer of a regular Neural Network would have 32*32*3 = 3072 weights. This amount still seems manageable, but clearly this fully-connected structure does not scale to larger images. For example, an image of more respectable size, e.g. 200x200x3, would lead to neurons that have 200*200*3 = 120,000 weights. Moreover, we would almost certainly want to have several such neurons, so the parameters would add up quickly! Clearly, this full connectivity is wasteful and the huge number of parameters would quickly lead to overfitting.

4.1.2 3D Volumes of Neurons

Convolutional Neural Networks take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way. In particular, unlike a regular Neural Network, the layers of a ConvNet have neurons arranged in 3 dimensions: width, height, depth. (Note that the word depth here refers to the third dimension of an activation volume, not to the depth of a full Neural Network, which can refer to the total number of layers in a network.) For example, the input images in some cases are an input volume of activations, and the volume has dimensions 32x32x3 (width, height, depth respectively). As we will soon see, the neurons in a layer will only be connected to a small region of the layer before it, instead of all of the neurons in a fully-connected manner. Moreover, the final output layer would for some cases have dimensions 1x1x10, because by the end of the ConvNet architecture we will reduce the full image into a single vector of class scores, arranged along the depth dimension. Here is a visualization:

A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

4.1.3 Layers of CNN

A ConvNet is a sequence of layers, and every layer of a ConvNet transforms one volume of activations to another through a differentiable function. We use three main types of layers

Figure 2: CNN Model

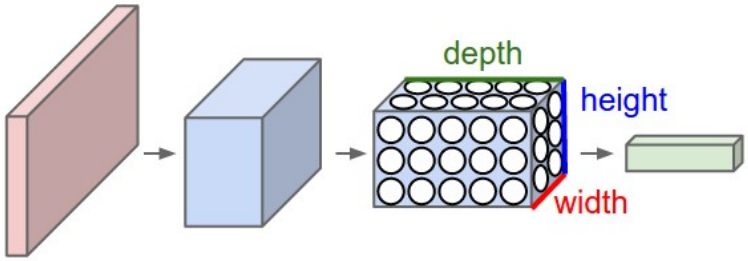
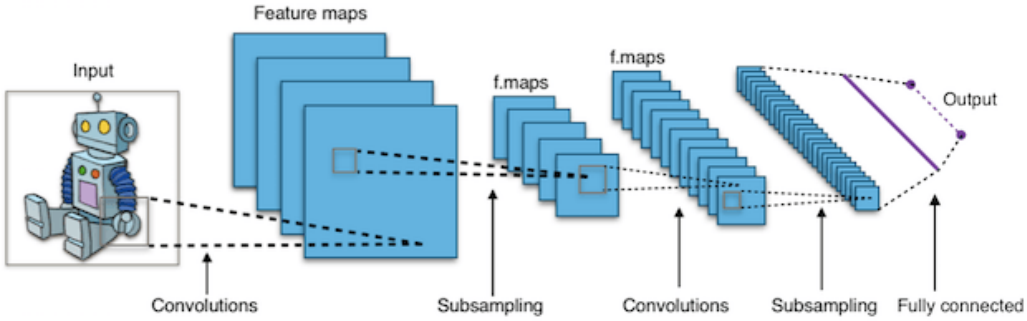


Figure 3: A Typical CNN Architecture



to build ConvNet architectures: Convolutional Layer, Pooling Layer, and Fully-Connected Layer (exactly as seen in regular Neural Networks). We will stack these layers to form a full ConvNet architecture.

A typical ConvNet for image classification could have the architecture [INPUT - CONV - RELU - POOL - FC].

INPUT [32x32x3] will hold the raw pixel values of the image, in this case an image of width 32, height 32, and with three color channels R,G,B. CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This may result in volume such as [32x32x12] if we decided to use 12 filters. RELU layer will apply an elementwise activation function, such as the max(0,x) thresholding at zero. This leaves the size of the volume unchanged ([32x32x12]). POOL layer will perform a downsampling operation along the spatial dimensions (width, height), resulting in volume such as [16x16x12]. FC (i.e. fully-connected) layer will compute the class scores, resulting in volume of size [1x1x10], where each of the 10 numbers correspond to a class score, such as among the 10 categories of CIFAR-10. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume. In this way, ConvNets transform the original image layer by layer from the original pixel values to the final class scores. Note that some layers contain parameters and other

don?t. In particular, the CONV/FC layers perform transformations that are a function of not only the activations in the input volume, but also of the parameters (the weights and biases of the neurons). On the other hand, the RELU/POOL layers will implement a fixed function. The parameters in the CONV/FC layers will be trained with gradient descent so that the class scores that the ConvNet computes are consistent with the labels in the training set for each image.

In summary, a ConvNet architecture is in the simplest case a list of Layers that transform the image volume into an output volume (e.g. holding the class scores) There are a few distinct types of Layers (e.g. CONV/FC/RELU/POOL are by far the most popular) Each Layer accepts an input 3D volume and transforms it to an output 3D volume through a differentiable function Each Layer may or may not have parameters (e.g. CONV/FC do, RELU/POOL don?t) Each Layer may or may not have additional hyperparameters (e.g. CONV/FC/POOL do, RELU doesn?t)

4.1.3.1 Convolutional Layers

The Conv layer is the core building block of a Convolutional Network that does most of the computational heavy lifting.

Figure 4: Convolutional Layers

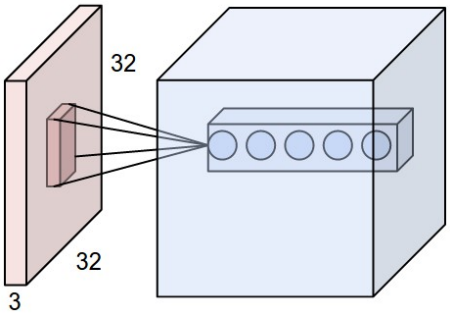
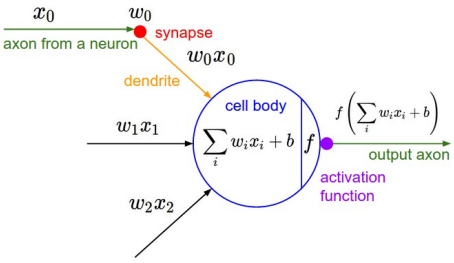


Figure 5: Convolutional Layers Neuron Model



4.1.3.2 Pooling Layer

It is common to periodically insert a Pooling layer in-between successive Conv layers in a ConvNet architecture. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation. The most common form is a pooling layer with filters of size 2x2 applied with a stride of 2 downsamples every depth slice in the input by 2 along both width and height, discarding 75% of the activations. Every MAX operation would in this case be taking a max over 4 numbers (little 2x2 region in some depth slice). The depth dimension remains unchanged. More generally, the pooling layer:

4.1.3.3 Normalization Layer

Many types of normalization layers have been proposed for use in ConvNet architectures, sometimes with the intentions of implementing inhibition schemes observed in the biological brain. However, these layers have since fallen out of favor because in practice their contribution has been shown to be minimal, if any. There are various types of normalizations.

4.1.3.4 Fully-connected layer

Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.

4.1.3.5 Converting FC layers to CONV layers

It is worth noting that the only difference between FC and CONV layers is that the neurons in the CONV layer are connected only to a local region in the input, and that many of the neurons in a CONV volume share parameters. However, the neurons in both layers still compute dot products, so their functional form is identical. Therefore, it turns out that it?s possible to convert between FC and CONV layers.

5 Results Observed

The accuracy is above 90%.

6 Technical Benefits of the Proposed Solution

From this approach that help the visually impaired and inventory features in cooler that detect large mechandise to auto-organizing untagged photo collections and extracting business insights from socially shared pictures, the benefits of image recognition, or computer vision, are only just beginning to make their way into the world-but they're doing so with increasing frequency and depth. The advancements in image detection from the proposed method are creating tremendous new opportunities in analyzing images that are exponentially impacting every business vertical, from automotive to advertising to augmented reality,

7 The case of Bias and how it can be evened out

The case of bias is to having more redbull photos than nonredbull photos. It can be evened out by creating relative equivalence classes of photos.

8 Scope for improvements in the model and solution

The model can be extended to detect the empty and non-empty of cooler storage. It also can be enhanced to estimate the quantity of inventory.