Parts of Deep Learning

Konstantin Burlachenko (burlachenkok@gmail.com)

Last update: 9 MAY 2022

Contents

1 Introduction

This document is a draft which I have created for purpose to systematize used tricks in various application around deep learning models. The all original papers are driven firstly by practice. This is not a note about that principle and it's pros. and cons. For some areas it's probably fine, for some does not.

2 AlexNet

The AlexNet paper "ImageNet Classification with Deep Convolutional Neural Networks" [1] is arguably the paper that brought deep learning to the forefront. After reading the paper [1] we can obtain the following main ten ideas from it.

First idea Authors faced with situation that before them people work with smaller tasks and people provides in common the following suggestion in the area of computer vision: *"... collect larger datasets, learn more powerful models, and use better techniques for preventing overfitting..."*. For example before AlexNet was a work of

LeCun, 1998 [**lecun1998gradient**] and model with LeNet-5 which uses *60K parameters*. Authors tried go beyond that.

Second idea Architecture presented at [1], Fig.1 carefully transform original input image. With several steps spatial dimension is reduced with 1 strided convolution (stride 4), and 3 max-pools. From another side channel dimension is increasing. The work empirically try to loose information very carefully. Finally input image is converted into long 4096 vector. During inference images are projected into that space. Finally there is a softmax into 1000 classes. What has been observed practically the importance of the depth of the network (in terms number of layers) which seems matters for vision tasks.

Third Idea Authors have made the hugest up to date (at moment of 2012) model with **60 million parameters** to train a models with using 1.2 millions samples. In fact 1.2 million samples is still nothing for huge dimensional models. And due to that image classification task is not so easy due to a that inherited difficulty.

Fourth idea Authors use Relu activation function, also known as $max(0,x)$ or x_+ for people from Optimization community. Original motivation of the authors was that it was practically impossible to train with sigmoid in that time, due to practically observing slower training time in case of using sigmoid activation function.

Fifth idea Authors split layers across several GPUs by hands. And there is an additional trick as authors said the inter-GPU communication happened in a specific layers.

Sixth idea Local Response Layer (LRL) try to "normalize" along channel (i.e. in each specific location for each feature map). Authors stated that for the task it appends generalization ability. Inspiration for this author obtained from research how real neurons behave.

Seventh idea This paper make huge impact and momentum into Deep Learning technics modeling approach for tackle image classification tasks.

Eight idea Network is trained on RAW RGB images with central crops which after uniform rescaling into 256 x H or W x 256 are cropped by central crop via obtaining 256x256. There is no use of SIFT or any specific Computer Vision algorithms for feature detection / important points detection.

Ninth idea Data Augmentation generates artificially trained images on the fly. One extra thing authors found via PCA/SVD the need space for images and append unbiased noise for new images via adding random variables in that space.

Tenth idea Use CNN layers for architecture by itself and build computation graph with that primitives. Use *DropOut* regularization mechanism which in the same time can reduce compute time and prevent over fitting. Where *Dropout* is in fact a sampling mechanism across Neural Network architectures by itself.

3 Example with arithmetic for counting filters

Assume input image of size **55x55x3** and response or output of the convolution filter is **55x55x32**.

Number of filters Assume that by filter we use usual convolution which for depth tensor extension look completely into input channel. We need under this assumption 32 filter, because output volume has 32 activation maps stacked together in the depth dimension/axis.

To go from input volume [55,55,3] given in HWC format to another volume [55,55,32] with filter size 5 and bias term we need 32 filters with spatial dimension [5,5] HW and zero padding occurs with in a format of SAME convolution. Same convolution has padding such that $W_{out} = floor(W_{in} - F + 2P)/S + 1$ is the same as W_{in}.

Storage for all filter parameters The storage for this filter requires us $32x5x5x3$ parameters for convolution kernels and extra 32 bias terms. And finally, we need 2432 scalar parameters.

Storage for a single filter parameters For each filter, we need $5x5x3$ scalars describes filter and one single scalar for bias, a total of 76 scalars for one filter.

Affect of strides and padding for number of parameters in a filter Stride convolution is a step with which filter with the rectangular or square form is sliding over the input tensor in both spatial dimensions. For classical convolution applied in CNN, there is the freedom to move in the depth dimension of input tensor for convolution kernel, because the kernel's depth dimension is the same as the input dimension. Stride or Pad characteristics of convolution affect output tensor's dimension in spatial extend (i.e. W and H) but the number of weights is the same for strided or not strided convolution.

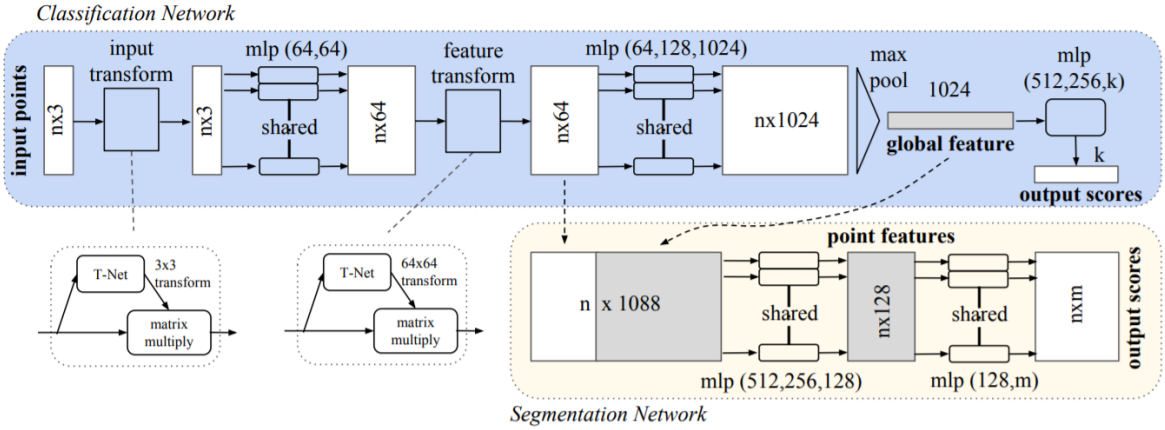
... Continue from Assignment 2.

4 PointNet

PointNet was introduced in work [**qi2017pointnet**]. PointNet is working with geometry in the form of point clouds from $\mathbb{R}^3$. PointNet works with the set of unordered points. The motivation behind having PointNet building blocks is to have internal mechanisms that works with a set of points and which which are permutation invariant w.r.t. to the order of points in input specification. Several operations are permutation invariant:

1. Sharing weights for parallel implementation (per point) neural net is permutation invariant w.r.t. to order of input
2. Max and Mean pooling are permutation invariant computation blocks w.r.t. to order of input

This idea motivates creates the following schema with several Multi Layer Perceptron(MLP) with weight sharing:



The near-final output is 128 feature vector per each point. PointNet achieved impressive performance on a few benchmarks, however:

1. It cannot capture the local context on different scales
2. No point in PointNet uses its local neighborhood, and it does not generalize well.

5 PointNet++

PointNet++ has been introduced in a work [**qi2017pointnet++**]. The main idea of PointNet++ is that it introduces a hierarchical neural network that applies PointNet

recursively on a nested partitioning of the input point set. The hierarchical structure is composed of a number of set abstraction levels. There are three main features of PointNet:

1. Usage of a Set Abstraction Level
2. Multi-scale grouping to help with solving the problem with non-uniform density. This is achieved with features at different scales are concatenated to form a multi-scale feature and with random input dropout.
3. In a set segmentation task such as semantic point labeling, we want to obtain point features for all the original points. The authors have appended a feature propagation mechanism to support that.

The limitation of PointNet++:

1. It works with points and ignores points relationship even if they exist
2. It works with static points
3. It does not support missing points or points with missing with one of the components
4. Another inherited limitation of MLP. For example, the authors state that MLP is a universal continuous function approximation, and it's only partially true. More precisely, it's true only if have no limit to train data, and have the ability to infinitely blow up hidden layer. Both requirements are not valid in typical applications of Neural Nets. In case limit number of train set or limit topology MLP are not general function approximation.

One of the most important components is a set abstraction level, and it is made of three key layers:

1. **Sampling layer**. The Sampling layer selects a set of points from input points, which defines the centroids of local regions. We can call it anchors.
2. **Grouping layer**. The grouping layer then constructs local region sets by finding "neighboring" points around the centroids.
3. **PointNet layer**. The PointNet layer uses a mini-PointNet to encode local region patterns into a feature vector. PointNet is applied for the neighborhood to mimic convolution.

The benefits of this layer:

1. Learns "kernels" in compact space
2. Kernels have compact spatial support similar to convolution with small compact support
3. Suggest a concrete strategy to find central points for operations like convolution.
4. It uses ideas like Inception with multi-scale
5. It contains ideas similar to Drop-Out from CNN's