Deep Learning Training Report

Table of Contents

Stanford Deep Learning Courses Report

The Stanford Deep Learning course (cs231n) is the most important thing I've learned over this winter vacation. It is a great introduction for me to computer vision. In the following part I conclude a couple of keywords and important fundamental methods I've learned.

Data-driven Approach

When people think about vision, we come up with a lot of description words, like the tree is tall; the basketball is round and tough. That's why people wanted to solve the image classification problem by detecting all the features to identify an object. Soon people found it can't work well because it is impossible for people to conclude all the features of everything. So data-driven approach came up. Instead of giving a direct method to identify an object, we are going to show a large number of pictures to the computer and let it teaches itself.

In my opinon, this approach is the **cornerstone** of machine learning. It shows that machine learning does not only rely on the methods, the algorithm, the architecture of the network, but also the data sets. Only with the large data sets will the computer teach itself well. But the data sets must be appropriate. A million picture of cats is a not good one because the computer will only learn something about cat. A million pictures of a million categories are also bad because with little examples of each category the computer would not learn anything. Data sets are important in Computer vision.

Loss Function

Suppose our neural network produce an output. If the output is exactly the same with our labels, that is great. If not, we are going to adjust the network according to the 'unhappiness' of our output. There we have loss function. Currently we have two commonly used losses, called SVM and Softmax.

SVM:

Softmax:

What's more, in order to prohibit the over-fitting, here we need to add regularization penalty. I think the reason why regularization is necessary and important is that in the training set there might be some noise. To avoid the model fits the noise in the data, we must set a special parameter to adjust the network. But we can't adjust them too much because sometimes we 'over-punished' the network. So selecting a proper regularization penalty is important. Usually people select it by doing lots of experiments and find one they may think work best.

I got a question over this part: When I was playing with the demo provided by Prof. Andrej Karpathy in the course, I found out that Softmax and SVM has a little difference. For the same dataset, the lines that divide the three colors ended in different position when I used different loss function apparently. I was told that SVM and Softmax will produce similar result, so why I got different result over this small network?

Gradient Descent

We developed the intuition of the loss function as a high-dimensional optimization landscape in which we are trying to reach the bottom. Gradient Descent is the most common way to optimize the network, and it works well. The most ingenious part is using **back propagation** to compute partial derivatives, and the gradient. It avoids large computation wasted in computing gradients.

Here I have another question: Since the derivation formula is calculated by people, and the gradient is the multiplication of several partial derivatives, how come it would become false? Why do we need the gradient check? What is the problem that probably comes out?

Activation Functions

Activation Functions are used to make the network less linear. CNN is a process to transfer image space to semantic space. Therefore, it needs strong non-linearity. Simple network is linear operations. With activation functions one layer contains non-linearity. Therefore, the network with deeper architecture, which contains more layers, will result in stronger non-linearity.

The most commonly used activation functions are Sigmoid, Tanh, ReLU, Leaky ReLU, Maxout. Usually we use ReLU as the activation functions in our network.

Preprocessing

There are three common forms of data preprocessing a data: Mean Subtraction, Normalization and PCA and whitening. The former two are widely used in computer vision. These methods are simple but necessary. Once I built networks with and without preprocessing and compare them. It shows a difference of 5% accuracy. I am not sure how preprocessing influence the accuracy after long-time training, but as far as I can see, preprocessing matters so much when you train the network in a short time.

Weight Initialization

Initialization is simple but really important. We can't simply initialize the matrix as zeros because gradient will diminish. We can't simply initialize it with small random number because of similar reasons. So there is a way called Xavier Initialization: w = np.random.randn(n) / sqrt(n)   (numpy format). Notice when you are using ReLU as activation function, you need to change the formula to: np.random.randn(n) / sqrt(n / 2)

Dropout

Over all the basic methods in neural network, there are two methods impress me a lot. One is back propagation, and the other one is dropout. While training, dropout is implemented by only keeping a neuron active with some probability or setting it to zero otherwise. This will ensure that every neuron trained well during the training. It avoids several specific features focused by the network, and the over-fitting following. It is ingenious because it seems did much more useless work but actually active every neuron.

Parameter updates

There are several ways of parameter updating. Simply using gradient and learning rate does not reach fine result. So physics is involved in the method. We use momentum as another parameter in parameter updating. We have a lot of methods to update parameters and they perform differently in different data sets.

Convolutional Layer:

Convolutional Layer consists of a set of learnable filters. Every filter is small spatially (along width and height), but extends through the full depth of the input volume. Each filter will produce a 2-dimensional activation map that gives the responses of that filter at every spatial position. And finally we will stack these activation maps along the depth dimension and produce the output volume.

Notice that when dealing with high-dimensional inputs such as images, it is impractical to connect neurons to all neurons in the previous volume (**comparing with the fully-connected layer**). Instead, we will connect each neuron to only a local region of the input volume. This feature is called Local Connectivity.

There are three hyper parameters in convolutional layer: the depth, stride, and zero-padding. The depth equals to the number of filters that we use in this layer. The stride shows how many pixels each time when sliding the filter. The zero-padding shows how we pad the input volume with zeros. The nice feature of zero padding is that it will allow us to control the spatial size of the output volumes (most commonly we will use it to exactly preserve the spatial size of the input volume so the input and output width and height are the same).

Pooling Layer

Pooling Layer is used to reduce the spatial size of the representation to reduce the number of parameters and computation in the network, and hence to control over-fitting. Usually we use MAX operation in the pooling layer, but sometimes we use other operations such as AVERAGE.

It is worth noting that there are only two commonly seen variations of the max pooling layer found in practice: A pooling layer with spatial extent is 3, and stride is 2, and more commonly spatial extent is 2, and stride is 2. Pooling sizes with larger receptive fields are too destructive.

Object Detection Report

Faster R-CNN

Faster R-CNN is an object detection framework. Faster R-CNN comes from Fast R-CNN, which comes from R-CNN. Faster R-CNN reaches high accuracy and short processing time.

Object detection is one of the hottest topics in computer vision. Unlike simple classification, in object detection the picture main contain several object, and they are at different positions.

If we are trying to consider object detection as classification, then the problem could be transformed into cutting a region out of the picture, and using classification to determine what object it is in the particular region. But how could we find the region? At first people cut all possible sizes of regions and do the classification. Obviously this method won't work well, because it is too slow. In fact, only a few regions are possible for object. So we can pick out these regions by Region Proposals. R-CNN is a method that combines Region Proposals and CNN.

But we still need a long time to train every possible region in CNN. So Fast R-CNN proposes a new idea that the network share computation of convolutional layers between proposals for an image. And it train the whole system end-to-end all at once. Fast R-CNN surprisingly speeds up the training time and the test time when region proposals time excluded.

Since CNN is fast, we can use CNN to deal with region proposals. Faster R-CNN use RPN (Region Proposal Network) to produce region proposals directly, so there's no need for external region proposals. They use 'joint training' to make the pipeline concise, where only one network with four losses (RPN classification, RPN regression, Fast R-CNN classification, Fast R-CNN regression). The RPN classification part gives a judgment whether the region is an object or not. The Fast R-CNN part identifies the object. The regression parts are mainly used to adjust the region. Since external region proposals are excluded, and the pipeline is cleaner, Faster R-CNN has a much better result than Fast R-CNN.

YOLO

YOLO (You Only Look Once) comes up with a different view of object detection. It solves detection as a regression problem. The algorithm seems pretty intuitive. It divides the entire picture into S * S grids (in the paper they use S = 7). If the center of an object falls in a grid, then the grid detects the object. So when a grid does detection, it should return B bounding boxes. Each bounding box should not only return the value of the bounding box, but also a ‘confidence value’, which suggests possibility of the occurrence of the object and the accuracy of the bounding box. YOLO uses 24 convolutional layers and 2 fully connected layers. And in the loss function part rhe authors distinguish the localization error and classification error by adding two hyper parameters.

There are some limitations about YOLO. Since it only provides one result for each grid, it will have unsatisfied result of closed objects. About the loss function, the model simply uses sum-squared error loss, which may cause errors in the large bounding boxes and small ones seem equal.

YOLO performs worse than Fast R-CNN in accuracy, but it does have a high speed. The author also did an experiment in combining Fast R-CNN and YOLO. It seems that the combination works better than Fast R-CNN itself.

Pose Estimation Report

RMPE

RMPE comes from our lab, and as far as I hear of, some senior students are still working on it. So I will conclude a general architecture of it in the report instead of a detailed one.

RMPE consists of three novel techniques, namely symmetric spatial transformer network (SSTN), deep proposals generator (DPG) and parametric pose nonmaximum suppression (NMS). The general idea is to pick up humans in the picture and estimate each person's pose. So firstly the model picks up all the humans in the picture. Then we use a symmetric STN, which consists of STN and SDTN. STN and SDTN are configured before and after the SPPE (Single Person Pose Estimation). Detected human proposal is received by STN and SDTN outputs pose proposal. Finally, a parametric Pose NMS (p-Pose NMS) is carried out before final predictions.

Different from traditional training method, the authors train the SSTN + SPPE module with images generated by deep proposals generator (DPG). DPG is carried out mainly because of the quantity of data if not sufficient for training a good model.

Considering about my current ability, I have no idea about how to improve this model. I even have a difficult time understanding the three novel techniques. However, I wonder that when two network produced by different group are putting together, will there be any compatibility problem? I mean surely it can work, but can it transfer data from one network to another smoothly? Maybe there will be some efficiency lost during the process. That’s what I’ve thought about RMPE.

Real-time Multi-Person Pose Estimation by Cao et al

Traditional Top-Down Approach does the work by detecting people and their pose. But Real-time Multi-Person Pose Estimation does it by detecting parts of humans and associating these parts.

The network works as sequence learning scheme. One of the branches predicts confidence maps for part detection, while the other one predicts part affinity fields for part association. And they stacked the stage of two branches a couple of time to make the final model.

They use Convolutional Pose Machines to detect the parts of humans. At each position, it provides a value of confidence for human part. So a stage will provide a heat map for a part. Stacking stages will result in a pretty precise position.

In the part association step, the authors introduce a quite original idea. The key point is to encode the part affinity score on the image plane. But the formula is too complex to describe, so I'll skip that formula in the report.

The last problem is to select different limbs linked in PAFs to combine as one person's skeleton. This is a classical generalized maximum clique problem.

Multi-object Tracking Report

Online Multi-Object Tracking by Decision Making

(I find this paper pretty interesting because it related to some knowledge in reinforcement learning which I will study in the next session. Maybe after I finished deep reinforcement learning I'll read this paper again. So this report will be a general report about the paper.)

This paper associates MOT (Multi-Object Tracking) with decision making, specifically, MDPs (Markov Decision Processes). It models the lifetime of an object with a MDP.

Let's talk about the MDP first. In the paper there are four states namely 'Active', 'Tracked', 'Lost', and 'Inactive'. When an object detected, it goes to 'Active'. And as time goes by, the object may be lost because of covered by some other objects or tracked by us directly, which corresponds to the 'Tracked' and 'Lost' states. Finally all of the objects will go to the 'Inactive' part. There are seven possible transitions are designed between the state of a object. They are 'Active' -> 'Tracked', 'Tracked' -> 'Tracked', 'Tracked' -> 'Lost', 'Lost' -> 'Tracked', Lost -> 'Lost', 'Lost' -> 'Inactive', and

‘Active’ -> ‘Inactive’. And the reward function will be learned from training data.

In MDP, a policy is a mapping from the state space to the action space. Given the current state of the target, a policy determines which action to take. Equivalently, the decision making in MDP is performed by following a policy. The goal of policy learning is to find a policy which maximizes the total rewards obtained.

After learning the policy and the reward of the MDP, they design a MDP for each target. By learned strategies, they can get features to determine the target’s action and states.

Object Segmentation Report

Instance-aware

Fully convolutional network (FCN), which is proposed in CVPR 2015, rapidly improves the accuracy of sematic segmentation. FCNs and improvements are designed to predict a category label for each pixel, but are unaware of individual object instances. Therefore, the paper proposed by Microsoft Research focuses on solving the instance-aware semantic segmentation.

The paper mainly works on three tasks: differentiating instances, estimating masks, and categorizing objects. The authors proposed a Multi-task Network Cascades (MNCs) to deal with the multi-task work. Unlike traditional multi-task learning, MNCs do not only share the feature in each task, but also use the result of the earlier stages as dependency. So the network could be divided into three stages as the tasks described. And the three stages are designed to share convolutional features (e.g. the 13 convolutional
layers in VGG-16).

In the first stage (differentiating instances), the model use Region Proposal Network) to predict the bounding boxes. RPN is in the form of fully convolutional network. More specifically, it uses 3 * 3 convolutional kernels to do downsampling and produce two loss functions: one related to the bounding box, and the other one related to object.

In the second stage (estimating masks), the model uses Region of Interest to pick up the features in the bounding box from the earlier stage. Then add two fully-connected layers. The first one decreases the dimension of the features to 256. And the second one make it pixel-wise mask.

In the third stage (categorizing objects), it uses the bounding box from the first stage adding two convolutional layers to get features, and connect it with mask feature from the second stage. The output is the score for each instance.

In fact, the authors set five stages in the model. The first and second are those described above. Since there are too many boxes, the third stage picks up some preferred bounding boxes (300 boxes) as the new bounding boxes. And the fourth stage is the same as the second one but just working on the 300 picked bounding boxes. Finally, the last stage is just the third stage we discussed above. The last stage use Softmax as the loss function.

Scene Parsing Report

Pyramid Scene Parsing Network

In this paper, the authors point out three kinds of problems: mismatched relationship, confusion categories and inconspicuous classes. In order to solve these problems, the authors introduce the Pyramid Pooling Module. And the authors point out that the receptive field of CNN is much **smaller** than the theoretical one especially on high-level layers.

The network looks like pyramid. The authors make the feature map pooled into four other sizes, and connect all of them together to do prediction.

Also the authors change the ResNet. They add auxiliary loss in it, and change the first 7 * 7 convolutional kernel into three 3 * 3 convolutional kernel.

Hourglass Model Report

First of all I should commit that I didn’t complete an hourglass model for human part segmentation. But I’ve delved into the architecture of hourglass model. Here is my understanding about the hourglass model.

The hourglass module is the key part of the hourglass model. It consists of several residual modules. Let’s talk about the simplest one, rank 1 hourglass module. One branch consists of some residual modules. It picks up features from the original size. The other branch consists of a max pooling layer (downsampling), some residual modules, and an upsampling layer (using nearest neighbor algorithm). Another way of upsampling is deconv. So rank 1 hourglass module make a combination of the features from the original size and 1/2 of original size.

Rank 2 hourglass module replace a certain residual module with rank 1 hourglass module. And rank 4 hourglass module is made in similar way. The regular rule is that: before every downsampling part, add a new branch to find features in the original size; after upsampling, combine the result with the other branch; between two downsampling parts, add some (three) residual modules to pick up the features; between two combination parts, add one residual module to pick up the features. Therefore, rank N hourglass module can pick up the features from the original size to 2^(-N) of the original size. In the paper, the author chooses the rank 4 hourglass module, and they stacked two hourglass module in the network. The input of the second module consists of three part: the input of the first module, the output of the first module, and the prediction made by the first module.

I think this is the main idea about hourglass module, but I had a difficult time in reading the codes of it. I don't know the meaning of each part of the code. So I don't know how to modify the code in order to do our project. I do think this is a big problem that I've found over the winter training. I'll talk about it later in the Summary part.

Summary

Over the winter vacation, I’ve learned a lot about deep learning. However, I found out some unsatisfied things during the study and report writing.

1. Code writing

During the training I feel that I’ve acknowledged some theoretical knowledge, but lack of experiment. I constructed some small network, but I didn’t know how the network was constructed in this way, how the hyper parameters were chosen. I just followed the instructions step by step. I think I need to follow someone to help him build a network, which I believe that I can experience with a lot of guidance coming from the senior students. Another reason that I didn’t do much experiments is that I am not clear about how long a network will run. I’m always worried about it will take a long time even beyond my computer’s ability… Lack of experiments results in disability in reading codes from large project. So I hope in the next month I can improve this ability.

2. English

Once I believed that my English is pretty good. However, during the time writing the report I found I was wrong. About this report, I am trying to describe everything in my own words instead of simply putting some pictures from the paper together. I want to get rid of mathematics formula and try to explain every algorithm well in words because I think this promotes better understanding about the algorithm. But I think I didn't express my thoughts well in this report. I just repeat same patterns of words over the report because of my shortage in vocabulary. If I'm going to publishing some papers, my poor English will cost me a lot of time in writing. What's worse, others may be confusing about what I represent in the paper. I will work on this thing. But I am proud even though I had a difficult time in writing the report, I insisted to the end after all!

3. Paper Reading

I want to talk about problems I've met in reading papers. Every paper is around 10 pages, and contains a lot of information. I wonder is there any priority in reading paper? If so, which part would be the most important one? And should we memorize all the details in the paper, or the general idea of it?

Thanks for Prof.Lu for designing the training plan for us. Thanks for the senior students who helped us during the winter vacation. This is only a small step in the long journey, and I am sure that I’ll be the one make it to the final lane.