Report of the First MVIG Training Section

ZeLin Ye

1. Introduce

In this winter vacation, I have taken some actions to learn some knowledge of deep learning.

First, I installed linux operating system and three popular framework on deep learning, caffe, torch and tensorflow.

Second, I learned the materials in CS231n from [1], especially the parts of slides and notes. I also tried my best to finish some of the "Assignment" in this website.

Finally, I also find other deep learning materials and papers from some other platforms, like csdn, tuicool or even zhihu.

After a winter vacation study on deep learning, I have mastered some basic knowledge of deep learning. Besides, I have also learned something about the five important topics in deep learning area: object detection, pose estimation, multi-object tracking, object segmentation and scene parsing. To deepen my understands on the five topics. I also read some relevant papers and try to run corresponding project demos.

From the learning experience, I have not only mastered some modules to solve relevant problems, but I also have some own thoughts about these topics. Before the end of this learning section, I complete a small project on human part segmentation.

The following are some implement details and results of the five topics and my project. I mainly illustrate the topics through the models in some papers that are from "SJTU_MVIG_training".

2. Object Detection

As a vital topic of deep learning, object detection mainly deals with detecting a certain class objects, such as humans, dogs, ships and etc, in images or even videos.

There are many approaches to implement object detection, from traditional R-CNN to Fast-R-CNN and even Faster-R-CNN.

However, what appeals most to me among those approaches is

YOLO(You Only Look Once), and I have read relevant paper from [2], which gives me a deeper understanding of object detection.

YOLO has a totally different way to implement object detection. Most of previous systems make their own classifiers or localizers to achieve this goal. Those systems consider high scoring regions of the input image as detections.

YOLO is a real-time object detection system. It looks at the whole image when testing. Besides, it just uses a single network to do predictions. For these reasons, it can run much faster than those traditional systems like Faster-R-CNN.

Here are some implement details of YOLO from my understandings. First, the system divides the input image into many cells. Each cell will test the object whose center is in this cell. To implement its test, each cell will have some bounding boxes, and each bounding box has five parameters: x, y, w, h, pr(object).

x and y represent the central coordinate of the box, w and h represent its width and height, pr(object) represents the probability of an object in the box. If there are some objects in the box, it will calculate IOU(Intersection-Over-Union) to measure the accuracy of prediction.

At the same time, it can calculate the probability that the object is a certain class. Then the system will get the “score” of each class in each cell, we can set some proper thresholds to decide the result of predictions. Finally, we can mark out the object in image according to the information of bounding boxes, such as x, y, w, h.

The training network only contain convolution and pooling layers, so the system can accept input images that are in different sizes. The authors call the network as Darknet-19, which contains 19 convolution layers and 5 pooling layers. The network is similar to the VGG and it use quantities of 3*3 convolution. Besides, the network also refers to NIN to use 1*1 convolution and avg-pooling.

After understanding its implement details, I downloaded YOLO project and ran the demo it provide and I got satisfied result. Besides, I changed the detection threshold and observe the change of result.

3. Pose Estimation

After reading the paper of [4], I know a novelty way to implement pose estimation.

For most people, they may use top-down way to implement pose estimation, which means they use object detection system to detect people, then they make pose estimation. Neveras, in this paper, the authors use a novelty way: they do parts detection first to locate joints, such as elbow, wrist and etc. Then they do parts association to get the whole pose estimation.

Here are the implement details of their model from my understands.

First, to do part detection, they use confidence maps to represent, which models the part locations as Guassian peaks. What is interesting is that they take the maximum of the confidence maps instead of average, which can make the precision of close-by peaks remain distinct. When testing, they can obtain body part candidates by performing non-maximum suppression on predicted confidence maps.

Second, they use part affinity fields to implement part association. For part association, they need a measurement for what two joints should be associated. They predict confidence maps for n interpolated midpoints along the line segment connecting two body parts. However, this way does not contain orientation information of the limb. To improve this approach, they present part affinity fields, a vector field. Each pixel is belong to a certain limb type, a vector presents the direction that points from one part to the other. Each type of limb has an associated affinity field joining its two associated body parts.

Finally, they implement multi-person parsing by PAFs. After the former procedures, they get a set of body part detection candidates for multiple people using nonmaxima suppression on each predicted confidence map. In this part, the authors use Hungarian algorithm to obtain the max matching. There may be some problems in finding full body pose of multiple people, the authors do two relaxations to optimization: choose a minimal number of edges and further decompose the cliques partition problem into a set of bipartite matching subproblems.

Indeed, the system has its shortcomings, as is shown in the end of their video[5], when some people act in special way, it may make some mistakes.

4. Multi-Object Tracking

Multi-Object Tracking(MOT), is to track multiple objects in a video. I have learned more about MOT mainly by reading the paper in [6].

In this paper, the authors use a novel way to solve the MOT problem. They consider MOT as a problem of decision making in Markov Decision Processes(MDPs). They convert the process of learning a similarity function for data association into learning a MDP policy. Here are some details about the policy.

The policy, explain in detail, is a mapping from space of state to action space. That is to say, the policy will take some action according to the state of its target. The authors divide the state of its target into three types: active state, tracked state and lost state.

For the active state, the authors train a binary SVM(Support Vector Machine) offline, then they can classify a detection into tracked or inactive with a normalized 5D feature vector.

If a target in a tracked state, the MDP will determine whether the target stays tracked or should be transferred into lost state. The decision is mainly made by judging if the target is not occluded and is in the camera's field of view. If so, the system will keep tracking the target, otherwise, the target will be considered lost.

In a lost state, the MDP will do similar things like what it does in tracked state. It decides whether to keep the target marked as lost, transfer it into tracked state or mark it as inactive. Once the target has been lost for more than T_{Lost} frames, the system will mark the lost target as inactive and terminate the tracking.

However, there are some difficulties in determining whether to track the target or keep it as lost. The authors solve the problem by considering it as a data association problem, in other words, a target be associated with one of the detections from the object detector will be transferred into tracked, or it will keep lost.

After learning a MDP policy, they can use it to MOT. Each object

has its own MDP, which tracks the object according to the learned policy. When given a new video frame, the system first determine whether the targets are still be tracked or become lost. Then, they calculate pairwise similarity between the lost targets and objects that are not be tracked. After that, with the the similarity scores, they can use Hungarian algorithm to get the assignment between detections and lost targets. By the assignment, they can transfer the lost targets, which are linked to detections, into tracked. Also, the lost targets will stay as lost in opposite case.

5. Object Segmentation

Object segmentation is to fill some different kinds of objects with different colors. However, there are many approaches, even some leading approaches, which can not identify object instances. I learn a way for instance-aware semantic segmentation in the paper in [7]. Here are some of my understands about this paper.

In this paper, the authors present Multi-task Network Cascades(MNC), Their model contains three networks, used respectively to defferentiate instances, estimate masks and categorize objects. The three networks form a cascaded structure and they can share their convolutional features with each other.

Correspondingly, MNC also has three stages: proposing box-level instances, regressing mask-level instances, and categorizing each instance.

First, they use the form of bounding boxes to proposes object instances, and they use the RPNs model to construct their first network.

Second, they use the shared convolutional features and boxes in first stage as the input of stage 2. Then the network will outputs a pixel-level segmention mask for each box proposal.

To implement this, they do the following works. They use RoI(Region-of-Interest) pooling to extract a feature of a box from stage 1. Then they will get a fixed-size feature no matter the size of the box. After that, for each box, they append two fully-connected layers to the feature. The first layer is used to reduce the dimension while the second regresses a pixel-wise mask.

Finally, similar to the above two stages, they use shared convolutional features, boxes from stage 1 and masks from stage 2 as input of stage 3. This network will output category scores for each instance. For the boxes, the network do the similar thing in stage 2. And for the masks, they will “mask” the feature map.

6. Scene Parsing

Scene parsing is to assign a category label to each pixel in the image. It has some similarities with object segmetation, however, it can provides a complete understanding of the scene and can predict the label, location, even the shape of each element.

The mainstream algorithm of scene parsing is based on FCN (Full Convolution Network), but there exists a problem in the system that the algorithm does not make full use of the overall scene information. I learn a state-of-the-art model for this topic in the paper from [8].

In this paper, the authors present a pyramid scene parsing network(PSPNet). Their implement way can be divided into the following several steps.

First, when given an image, they use CNN to get the feature map of the last convolutional layer, then they can get different sub-region representations by their pyramid parsing module.

Besides, they can also get upsampling and concatenation layers to form the final feature representation, which carries both local and global context information in pyramid pooling module.

Finally, the representation is fed into a convolution layer to get the final per-pixel prediction.

To explain in detail, the pyramid pooling module has four layers. The thickest layer among them is global pooling, which can get a single bin, while the other layers can only get sub-region, therefore, after pooling, the feature maps have different size. Then, they do a 1*1 convolution for each pyramid level, which can transform the dimension of context representation into the original 1/N(N is the number of layers of pyramid).

After that, they can sample on the feature maps that have low dimension to get the size of original image. Finally, the features of

different layers can be connected and output after convolution.

7. Small-Project

The project is about human segmetation. With the offered materials, I can get a network, some codes and dataset to simplify this project. The followings are some implement details and training results of my project. The usage of the project can refer to [10]

7.1. Implement Details

In this section, I will introduce how I finish the project in detail.

7.1.1. Build the Dataset

First, I download the origin dataset from the official website[11].

Then, I find the pictures that contain the class of people, and I choose both the origin images and their annotations to build my own dataset. However, these images can not be used as input directly, I need to do some processing to them.

To processing the images, I use the “crop” function, provided by the authors, to resize the images. Because the network can only accept images of a certain size.

Finally, I use the information in the annotations, which include the coordinates of the various parts of the body, to generate a Gaussian graph in each part of the body. Up to this step, these images can be used as input of the network.

The final dataset can be found in [12].

7.1.2. Rewrite Parts of the Original Codes

Although the authors offer demo codes, it is unreasonable to apply them directly into my project.

Therefore, after I build my dataset, I need to modify the original codes according to the need of this project and my machine conditions. Indeed, I still use the original hourglass network structure. I just change some parts to make the model adapt to my machine conditions. It seems like a easy job, but I should first understand most of the original codes, which is proved to be a hard work.

7.1.3. Train and Test the Model

After step 1, I get approximately 1800 images, taking the factors of time and machine into account, I just use 1200 of them to train this model, and I just set the “epoch” as 10.

When I finish the training process, I use corresponding test dataset to test this model, and get results.

The trained model can be found in [15].

7.2. Results

After training my model, I test its accuracy with test dataset, and the final results are as follows:

At the beginning of training, the accuracy increase very fast, and it gradually become slower.

The final results can refer to the two figures:

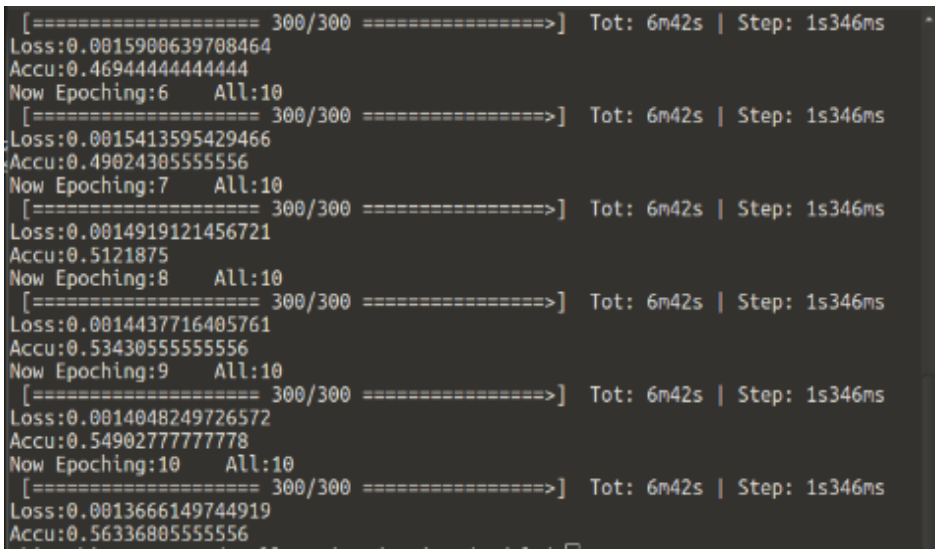

Figure1: Training Results

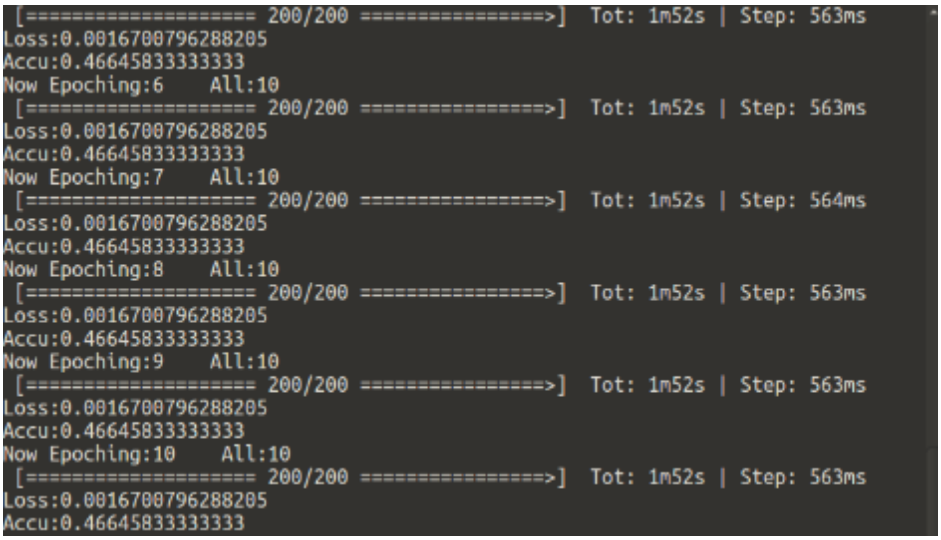

Figure2: Testing Results

Indeed, due to my machine conditions and limited time, I can only train this model with a small dataset and limited "epoch". The results may not be so perfect, but I really learn a lot from this project, and I believe this model will perform better if I use a bigger dataset to train it.

8. My Plan for the Next Training Section

According to the "SJTU_MVIG_training", our next training section is about deep reinforcement learning. In this section, there will be more missions and less time compared with first section.

Therefore, I think I should first learn some lessons from the first training section, especially the way to find learning materials and how to learn from papers of others. Then, I have to make a good plan for each mission. Finally, I think I should communicate with mentors when facing difficulties, where I have not done a good job in first section.

9. Acknowledgements

During this period of learning, I did meet a lot of difficulites. Therefore, I would ask and discuss with others. I would like to extend my sincere gratitude to ChenXi Wang and YongXi Huang, for their exquisite advice when I meet difficulites.

10. References

[1] http://cs231n.stanford.edu/syllabus.html
[2]https://arxiv.org/pdf/1612.08242.pdf
[3]http://blog.csdn.net/zhang_shuai12/article/details/52554604
[4] https://arxiv.org/pdf/1611.08050.pdf
[5] http://posefs1.perception.cs.cmu.edu/Users/ZheCao/humanpose.mp4
[6] http://cvgl.stanford.edu/papers/xiang_iccv15.pdf
[7] https://arxiv.org/pdf/1512.04412.pdf
[8] https://arxiv.org/pdf/1612.01105.pdf
[9] http://www.itdadao.com/articles/c15a977424p0.html

[10] https://github.com/shinshiner/deep-learning
[11] http://host.robots.ox.ac.uk/pascal/VOC/voc2010/
[12] https://pan.baidu.com/s/1o8dimWq
[13]https://arxiv.org/pdf/1506.01497.pdf
[14]https://arxiv.org/pdf/1612.00137.pdf
[15] http://pan.baidu.com/s/1gfJtYGb