Report

Yongxi Huang

Introduce

After reading the paper of five topics about deep learning and run some project demos, I have got some general understandings of deep learning. But to be honest, the most difficult part of learning is the fundamental knowledge about deep learning, some statics knowledge for example. Here are something I learned from the five topics and the human part segmentation project.

1. Object detections

Object detections is an important part of deep learning, which means detecting different objects from an image or video. Unlike image classification, detection is much more complex because it needs the accurate localization of objects. That means we not only need to process numerous candidate object locations, but also need to refine the localization to make it more precise. Among the various approaches to solve this topic, I mainly pay attention to two kinds of solutions. One is the approaches based on region proposal methods and the other is end-to-end method.

For approaches based on region proposal methods, there are R-CNN, R-FCN, Fast R-CNN, Faster R-CNN and so on. R-CNN (Region-based Convolutional Neural Networks) use Region Proposal and CNN to detect object. It "bridging the gap between image classification and object detection." [1] It firstly uses CNN instead of some approach like HOG to solve object detection problem. According to my understanding, the process of R-CNN is like this:

1) Use selective search to extract around 2000 region proposals. And in order to read off features using CNN, the region proposals need to be wrapped to a specific size (227*227).

2) Then forward propagate the region proposals through the CNN to read off features.
3) Finally, two fully connected layers and a set of class-specific linear SVMs, the objects within the proposals can be detected. (R-CNN also uses greedy non-maximum suppression to reject regions. And use bounding-box regressors to refine the lcalozation. But I' m still puzzled by this theory. However, I think it is an important part of R-CNN.)

According to those details about R-CNN, we can see that there is a main problem in this approach. After getting the region proposals and begin to read off features by CNN, there is much repeated calculation. Besides, because that the features are extracted from each object proposal in each image for SVM and bounding-box regressor, the training process is very expensive in space and time.

The SSPnet model gives some optimization[2]. Instead of warping the region proposals before the convolutional layers, it replaces the warping process by a spatial pyramid pooling layer following the last convolutional layer. In that way, SSPnet can run the convolutional layers only once on the entire image. It can be seen as multiple networks that share all parameters. Although there are lots of optimizations, the training of SSPnet is still a multi-stage pipeline. And during fine-tunning process, SSPnet only fine-tune the fully-connected layers. The fine-tunning algorithm can' t update the convolutional layers. All of these are limitation of SSPnet.

So, here comes Fast R-CNN. According to my understanding, the architecture of Fast R-CNN is like this:

1) Input the entire image and a set of object proposals. And several convolutional and max pooling layers will produce a conv feature map.

2) For each object proposal, there is a region of interest (RoI) pooling layer that extracts a fix-length feature vector from the feature map. This layer dividing the RoI window into a grid of sub-windows and then max-pooling the values in each sub-window to get a fixed-size feature map. So it' s like a spatial pyramid pooling layer with only one pyramid level. And then the fixed-size feature map is mapped to a feature vector.
3) The feature vectors will be fed into some fully connected layers that finally give two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. That means it will get a multi-task loss and add the training of bounding box regressors to the CNN.

We can see that RoI polling layer and multi-task loss make the training a single-stage. And the training can update all network layers now because the RoI pooling layers make the process of selecting feature map differentiable.

Although the Fast R-CNN gives some optimization to the R-CNN, the region proposals are still given by selective search, which greedily merges superpixels based on engineered low-level features. To optimize that, there comes Faster R-CNN[4]. Faster R-CNN uses a deep CNN to compute proposals, which called Region Proposal Networks (RPNs). We can see the Faster R-CNN as two module. First is the RPNs and the second is the Fast R-CNN detector that uses the proposed regions proposed by RPNs.

A RPN input an image of any size and output a set of rectangular object proposals, each with an objectness score. Before the RPNs there are several conv layers which get the feature map of input image. And a small network will slide over the feature map, so that the RPN will work on different image size. In each sliding window, there are k reference boxes which are called anchors. They have different scales and aspect ratios. By feed the feature map into two fully connected layer (when thinking about the size of image, there also need to be a RoI pooling layer), a box-regression layer and a box-classification layer, k anchors will get 2k scores and 4k coordinates. Using these scores and coordinates, a multi-task loss can be

computed then, which can be used to train RPNs. Faster R-CNN also shares features for RPN and Fast R-CNN.

Those are object detection methods based on region proposals. We can see the process of these method as two part. The first is a classification problem and the second is a bounding-box regression problem. However, there are also some approaches that don’ t use region proposals, Such as the end-to-end method YOLO (You Only Look Once).

YOLO[5][6] uses a different way to implement object detection. It solve the object detection problem as an regression question rather than a classification question. YOLO only has convolution and pooling layers. In general, the convolutional layers are used to extract features and the pooling layers will predict the localization of the object and calculate the class probabilitiy. As for the implement details of YOLO, first, the input image would be divided by a grid. If the center of an object is in a cell of the grid, this cell would be responsible for detecting that object. Each grid cell predicts a bounding box and class probabilities associated with the bounding box. At the same time, it can calculate the probability that the object is a certain class. The object in image can be marked according to the information of bounding boxes.

2. Pose estimation

According to [8], the problem of human pose estimation is defined as the problem of localization of human joints. It is a key step toward understanding people in images and video. Many systems like DeepPose and SPPE implement it by extracting the precise pixel location of important key points of the body.

DeepPose[8] is an early approach using deep neural network to solve the pose estimation problem. In this system, the pose is expressed by the locations of all k body joints in a pose vector like y = (y1…yi…), where yi contains the location of i

joint (x,y). Using this definition, the pose estimation problem can be treated as regression. As for training, it trains a linear regression by minimizing L2 distance between the prediction and the true pose vector.

Stacked hourglass model[9] is another pose estimation approaches. Although I have trained and run it, the implementation details are still to complex for me. The hourglass network used in this approach pools down the image to a very low resolution, then upsamples and combines features across multiple resolutions. And stacked hourglass model performs repeated bottom-up, top-down inference by stacking two hourglasses. For layer implementation of the hourglass, it uses residual learning modules to build hourglass. The first predictions are from the first hourglass where the network has opportunity to process features at both local and global contexts. And the second hourglass module further capture and understand higher order spatial relationships over the first predictions.

But these single person pose estimation (SPPE) approaches don’ t work well when it comes to Multi-person Pose Estimation. A simple way to do muti-person Pose Estimation is to divide the problem into an objection detection problem and a single person pose estimation problem. But this method is time consuming and may meet some errors if the person detector fails. In addition, such top-down approaches may be more and more inefficient with more people in the image. So in the paper [9], it gives another bottom-up method to do multi-person pose estimation. That means, it does parts detection to locate the joints first. And then does parts association to get whole pose estimation. So body part detection and part association are the two principal tasks in this bottom-up approach.

For body part detection, they use the confidence map representation to models the part locations as Guassian peaks. After that, they need to assemble the body parts to form the full-body poses of an unknown number of people. More specifically, there should be a measurement of the confidence for two parts to detect whether they are from the same person. Their method is to predict confidence maps for n interpolated midpoints along the line connecting two parts. Besides, they not only encode the location information, but the eschew orientation

information of the limb too. That is implemented by a feature representation called part affinity fields(PAFs). It is a vector field which means for each pixel in the area belonging to a particular limb type, a 2D vector encodes the direction that points from one part to the other. Finally, they implement multi-person parsing by PAFs. They using Hungarian algorithm to obtain the max matching between different limbs of the same person.

3. Multi-object tracking

Multi-object tracking (MOT) means Tracking multiple objects in videos. After reading the paper [10] about MOT, I have some basic understanding of it. There are two main categories of MOT, offline-learning and online-learning. Offline-learning means that the learning is performed before the actual tracking. So it can’ t take account of the dynamic status and the history of the target. Online-learning conducts learning during tracking. To implement that, a strategy is to determine whether the training results are positive or negative, and train a function for data association. It can utilize the features based on the status and the history of the target. But the errors may be accumulated and cause tracking drift as there aew no grand truth annotations.

And in [10] the authors consider MOT as a problem of decision making in Markov Decision Processes (MDPs). They treat the process of learning a simplilarity function for data association as learning a MDP policy. This is a mapping from space of state to action space. They divide the state of the target into active state, tracked state and lost state. They can transform into each other in some case.

For active state, they train a SVM (Support Vector Machine) offline which can classify a detection into tracked or inactive. For tracked state, the MDP determine whether the target should be transferred into lost state. It will judge if the target is occluded or in the field of view. And for lost state, the MDF do the same and transfer the state to tracked of mark it inactive. Target is inactive when it has been lost for more than a certain number of frames.

After learning a MDP policy, it can be used to do MOT. There is a dedicated MDP for each object which follows the learned policy to track the object.

4. Object segmentation

Object segmentation means label each pixel with the class of its enclosing object or region. In [11], they train a fully convolutional network (FCN) end-to-end, pixels-to-pixels on semantic segmentation. In order to do a pixelwise prediction, we should guarantee the size of output image. So FCN replace some pooling layers of CNN into conv layers. And the FCN can use some pre-training network because of that. And FCN uses backwards strided convolution to upsampling.

However, FCNs are unaware of individual object instances. To optimize it, there comes a instance-aware semantic segmentation method via Multi-task Network Cascades (MNC) [12]. This model is called Multi-task because there are three networks used respectively to differentiate instances, estimate masks and categorize objects. These networks form a cascaded structure and can share the convolutional features.

The first network is constructed by RPNs model. And they use bounding boxes to proposes object instances. The conv features and boxes are shared with the second network. And it will output a pixel-level segmentation mask for each box proposal. Finally, the shared features and boxes will be the input of the last network which outputs category scores for each instance.

5. Scene parsing

Scene parsing is based on semantic segmentation while the scene parsing cares more on the whole scene. It predicts the label, location and shape of each element in the image. Compared with semantic segmentation, we can see that the scene and label are much more various than instances. The diverse scenes and unrestricted vocabulary make this problem very difficult. Many scene parsing

frameworks are also based on FCN. But FCN is not accurate enough. The FCN based models is lack of suitable strategy to utilize global scene category clues.

Pyramid scene parsing network (PSPNet) is designed to optimize other methods by incorporating suitable global features. It extend the pixel-level feature to the specially designed global pyramid pooling one.

According to PSPNets, when an image is input, CNN will be used to get the feature map of the last layer. Then the pyramid parsing module will generate different sub-region representations. It will get the context information from the feature map. Then there are some upsampling and concatenation layers to form the final feature representation. The convolution layer will turn these representations into per-pixel prediction.

6. References:

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

[2] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In ECCV, 2014

[3] R. Girshick, "Fast R-CNN," in IEEE International Conference on Computer Vision (ICCV), 2015.

[4] S. Ren., K. He., R. Girshick. and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In IEEE Transactions on Pattern Analysis & Machine Intelligence, 2016

[5] J. Redmon, S. Divvala, R. Girshick,A Farhadi. You Only Look Once: Unified, Real-Time Object Detection in Computer Science 2016

[6] Joseph Redmon ∗ , Ali Farhadi ∗ University of Washington ∗ , Allen Institute for AI. YOLO9000: Better, Faster, Stronger

[7] A Newell, K Yang, J Deng. Stacked Hourglass Networks for Human Pose Estimation arXiv preprint arXiv:1603.06937, 2016.

[8] Toshev, A., Szegedy, C.: Deeppose: Human pose estimation via deep neural networks. In: Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, IEEE (2014)

[9] Zhe Cao, Tomas Simon, Shih-En Wei, Yaser Sheikh, Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields

[10] Yu Xiang, Alexandre Alahi, and Silvio Savarese Stanford University, University of Michigan at Ann Arbor. Learning to Track: Online Multi-Object Tracking by Decision Making

[11] J Long，E Shelhamer，T Darrell. Fully convolutional networks for semantic segmentation. In CVPR 2015.

[12] Jifeng Dai Kaiming He Jian Sun. Instance-aware Semantic Segmentation via Multi-task Network Cascades

[13] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, Jiaya Jia. Pyramid Scene Parsing Network

About the small project

This is a small project using hourglass network module. It was a single person pose estimation. I have process the data from http://www.stat.ucla.edu/~xianjie.chen/pascal_part_dataset/pascal_pa rt.html to make it profitable for the input format of this model. However, I can’t made it by myself. So I want to thanks Chenxi Wang for his help here. The processing detail of data is here:

1) The model can only be used to estimate the pose of a single person. So I have to select the images that have people with their annotations from the data set.
2) There are different kinds of objects in the image. So the annotation of people’s pose may be mixed with the annotation of other objects. So I need to select the annotation of people’s pose from all of the annotation. Fortunately, all of the annotation is labeled by different classification.

3) Their might be more than one people in the image, while the data set can only solve single person pose estimation. So I need to choose the right object that the model will work on. And remove other annotation. I check the center of different people and select the one closest to the center of the image.
4) The size of image is note suitable for the model. I transport it simply by resize and crop the images. I make it by using the crop function in the model. And the annotation of these objects should obtain the same conversion.
5) Then the processed data should be fed into the model. That was the most complex process. I have thought about making some h5 files just like the original model. But it seems a little difficult. Finally, with the help from Wang, I using the functions in the model to get the data directly. That means using the data extracted from the dataset to create some tensors and use them directly.

However, as the training process is really time consuming, I have only train a simple model with 20 times training on the training data. As the training time is too small, this model can only get 50% accuracy on the validation data. I will train it again later to get a higher accuracy. Besides, as the trained model is too large to be uploaded to github, I will try another way to upload it.