Weekly Work Report

Hongzhi Liu

VISION@OUC

September 9, 2018

1 Research problem

During this week, we summarize and review the competition 2018URPC. First, We look at the photos of the forum to learn about the latest technology and sensor knowledge of underwater object recognition. Then we sum up past race experiences and insufficiencies in order to maintain the continuous development of this advantage. Besides, we make one slide together which can will be shown on Tuesday. After the exercise of contest, we can grow faster.

Because of my text to image research direction, I read some paper of GAN and AttnGAN. I try to learn the paper and understand the network structure. Although having encountered some difficulties when I trained the model, I succeed in saving model of AttnGAN.

2 Research approach

In the process of contest summary, I use the method of documentary analysis, comparative analysis and experimental research method. I read the thesis of GAN[1] and AttnGAN[2]. I try to unferstand core ideology in paper and learn about concept introduced by author.

Besides, I learn grammatical application of python on the one hand, and on the other hand, I try to write code files to achieve the goal. By this method, I can have a better understanding of python.

For deep learning, I watch the fifth course videos and write down the issues which I think are much important for further research. And then, I not only have learned the lessons of deep learning, but also put them into coding action.

3 Research progress

During summary and review for URPC2018, I have looked at the photos of the forum to learn about the latest technology and sensor knowledge of underwater object recognition. Our team make one slide together which includes lots of sharing things to be shown on Tuesday. I begin reading thesis about AttnGAN [2]. And I will list details about weekly work in Tab. 1 below.

Table 1: Weekly work progress.

URPC2018	Finish summary and review for URPC2018.
	Finish one slide about experience sharing.
	Finish learning about WSRC and UGAN.
CVPR2019	Finish learning paper about GAN and AttnGAN.
	Finish training model about AttnGAN.

4 Progress in this week

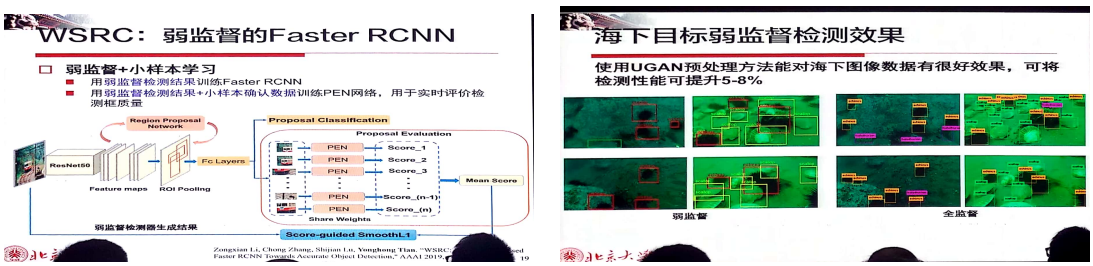
When we are summarizing about URPC2018 site, we learn about WSRC and UGAN which can be used in image enhancement. And I read paper about my research direction and train a model.

Step 1 Finish summary and review for URPC2018 competition.

Step 2 Finish one PPT file about experience of contest.

Step 3 Finish learning about WSRC and UGAN.

Step 4 Finish learning paper about AttnGAN and training model.



(a) (b)

Figure 1: Network architectures and results of WSRC.

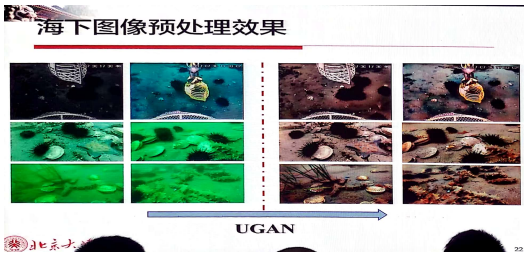


Figure 2: Underwater image preprocessing effect of UGAN.

4.1 Summary of the Competition

From this contest of URPC2018, I know the gap with other teams and get some new ideas about underwater target recognition from professors. We should have more underwater image enhancement methods to use. WSRC is one of examples as shown in Fig. 1 which is proposed by Professor Tian of Peking University. This algorithm is an improvement to the Faster RCNN algorithm based on weak supervised method.

From Fig. 1(a), we can see the WSRC's network architecture which includes ResNe50, Feature maps and ROI Pooling. Then the output into FC layers. The team use weak supervised and small sample in order to test results and train PEN network as well.

The undersea target weak supervision detection effectas shown in Fig. 1(b). Finally, we can see this method has a better result than state-of-the-art ones.

Besides, I also learn a new improved algorithm called UGAN as shown in Fig. 2.

4.2 Text-to-Image Convertor of AttnGAN

This week, I begin learn about Attentional Generative Adversarial Network (AttnGAN) that allows attention-driven, multi-stage refinement for fine-grained text-to-image generation. With a novel attentional generative network, the AttnGAN can synthesize fine-grained details at different subregions of the image by paying attentions to the relevant words in the natural language description.

In addition, a deep attentional multimodal similarity model is proposed to compute a fine-grained image-text matching loss for training the generator. The proposed AttnGAN significantly outperforms the previous state of the art, boosting the best reported inception score by 14.14% on the CUB dataset and 170.25% on the more challenging COCO dataset. A detailed analysis is also performed by visualizing the attention layers of the AttnGAN. It for the first time shows that the layered attentional GAN is able to automatically select the condition at the word level for generating different parts of the image.

```
1        def train(dataloader, cnn_model, rnn_model, batch_size,
2        labels, optimizer, epoch, ixtoword, image_dir):
3        cnn_model.train()
4        rnn_model.train()
```

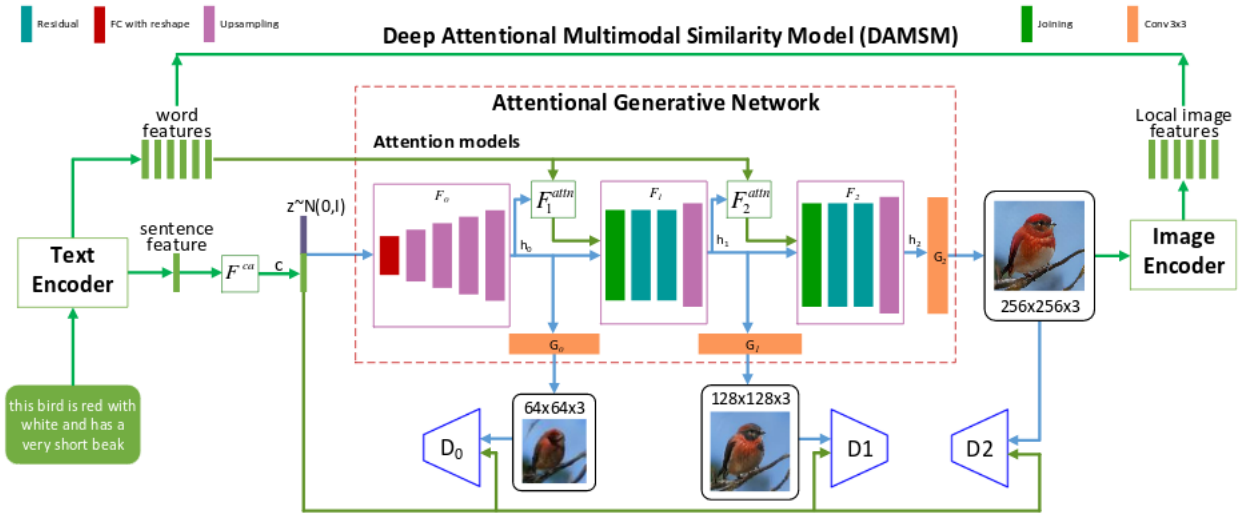


Figure 3: The architecture of the proposed AttnGAN. Each attention model automatically retrieves the conditions (i.e., the most relevant word vectors) for generating different sub-regions of the image; the DAMSM provides the fine-grained image-text matching loss for the generative network

Figure 4: Example results of the proposed AttnGAN. The first row gives the low-to-high resolution images generated by G_0, G_1 and G_2 of the AttnGAN; the second and third row shows the top-5 most attended words by F_1^{attn} and F_2^{attn} of the AttnGAN, respectively. Here, images of G_0 and G_1 are bilinearly upsampled to have the same size as that of G_2 for better visualization.

```
        s_total_loss0 = 0
        s_total_loss1 = 0
        w_total_loss0 = 0
        w_total_loss1 = 0
        count = (epoch + 1) * len(dataloader)
        start_time = time.time()
        for step, data in enumerate(dataloader, 0):
        # print('step', step)
        rnn_model.zero_grad()
        cnn_model.zero_grad()

        imgs, captions, cap_lens, \
        class_ids, keys = prepare_data(data)

        # words_features: batch_size x nef x 17 x 17
        # sent_code: batch_size x nef
        words_features, sent_code = cnn_model(imgs[-1])
        # --> batch_size x nef x 17*17
        nef, att\_sze = words\_features.size(1), words\_features.size(2)
        # words_features = words_features.view(batch_size, nef, -1)

        hidden = rnn_model.init_hidden(batch_size)
        # words_emb: batch_size x nef x seq_len
        # sent_emb: batch_size x nef
        words_emb, sent_emb = rnn_model(captions, cap_lens, hidden)

        w_loss0, w_loss1, attn_maps = words_loss(words_features, words_emb, labels,
        cap_lens, class_ids, batch_size)
        w_total_loss0 += w_loss0.data
        w_total_loss1 += w_loss1.data
        loss = w_loss0 + w_loss1

        s_loss0, s_loss1 = \
        sent_loss(sent_code, sent_emb, labels, class_ids, batch_size)
        loss += s_loss0 + s_loss1
        s_total_loss0 += s_loss0.data
        s_total_loss1 += s_loss1.data
        loss.backward()
        torch.nn.utils.clip_grad_norm(rnn_model.parameters(),
        cfg.TRAIN.RNN_GRAD_CLIP)
        optimizer.step()

        if step % UPDATE_INTERVAL == 0:
        count = epoch * len(dataloader) + step

        s_cur_loss0 = s_total_loss0[0] / UPDATE_INTERVAL
        s_cur_loss1 = s_total_loss1[0] / UPDATE_INTERVAL

        w_cur_loss0 = w_total_loss0[0] / UPDATE_INTERVAL
        w_cur_loss1 = w_total_loss1[0] / UPDATE_INTERVAL

        elapsed = time.time() - start_time
        print('| epoch {:3d} | {:5d}/{:5d} batches | ms/batch {:5.2f} | '
        's_loss {:5.2f} {:5.2f} | '
        'w_loss {:5.2f} {:5.2f}'
        .format(epoch, step, len(dataloader),
```

```
            elapsed * 1000. / UPDATE_INTERVAL,
            s_cur_loss0, s_cur_loss1,
            w_cur_loss0, w_cur_loss1))
            s_total_loss0 = 0
            s_total_loss1 = 0
            w_total_loss0 = 0
            w_total_loss1 = 0
            start_time = time.time()
            # attention Maps
            img_set, _ = \
            build_super_images(imgs[-1].cpu(), captions,
            ixtoword, attn_maps, att_sze)
            if img_set is not None:
            im = Image.fromarray(img_set)
            fullpath = '%s/attention_maps%d.png' % (image_dir, step)
            im.save(fullpath)
            return count
```

The overall architecture of the AttnGAN is illustrated in Fig. 3. The model consists of two novel components. The first component is an attentional generative network, in which an attention mechanism is developed for the generator to draw different sub-regions of the image by focusing on words that are most relevant to the sub-region being drawn as shoen in Fig. 4 and the training code can be seen in Algorithm 4.2. More specifically, besides encoding the natural language description into a global sentence vector, each word in the sentence is also encoded into a word vector. The generative network utilizes the global sentence vector to generate a low-resolution image in the first stage.

In the stages, it uses the image vector in each sub-region to query word vectors by using an attention layer to form a word-context vector. It then combines the regional image vector and the corresponding word-context vector to form a multimodal context vector, based on which the model generates new image features in the surrounding sub-regions. This effectively yields a higher resolution picture with more details at each stage. The other component in the AttnGAN is a Deep Attentional Multimodal Similarity Model (DAMSM). With an attention mechanism, the DAMSM is able to compute the similarity between the generated image and the sentence using both the global sentence level information and the fine-grained word level information. Thus, the DAMSM provides an additional fine-grained image-text matching loss for training the generator.

5 Plan

Objective: Finish thesis with senior students for CVPR2019.
Deadline: 2018.11.16

2018.09.03—2018.09.09 Finish reading CVPR2018 paper about AttnGAN [2].

2018.09.10—2018.09.16 Finish recurrenting results of benchmark table from CVPR2018 paper.

References

[1] I. J. Goodfellow, J. Pougetabadie, M. Mirza, B. Xu, D. Wardefarley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. In *NIPS*, 2014. 1

[2] T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He. AttnGAN: Fine-grained text to image generation with attentional generative adversarial networks. In *CVPR*, 2018. 1, 5