



文本识别模型

Attention and Language
Ensemble for Scene Text
Recognition with Convolutional
Sequence Modeling 分享

王裕鑫

中国科学技术大学



目录



论文概览



算法模型剖析



代码复现



团队介绍

方山城 王裕鑫 屈亚东
中国科学技术大学



一、论文概览





基本信息

论文题目

《Attention and Language Ensemble for Scene Text Recognition
with Convolutional Sequence Modeling》

出 处

ACMMM 2018 (Proceedings of the 26th ACM international conference on Multimedia)

作 者

Shancheng Fang^{1,2} Hongtao Xie³ Zhengjun Zha³

Nannan Sun^{1,2} Jianlong Tan^{1,2} Yongdong Zhang³

¹Chinese Academy of Sciences

²University of Chinese Academy of Sciences, Beijing, China

³University of Science and Technology of China, Hefei, China



研究背景

场景文本识别(Scene Text Recognition, STR)是识别自然场景图片中的文字信息。自然场景图片中包含了丰富的语义信息，能够用于基于内容的图片修复、自动驾驶、图片中的文字翻译等。由于受自然场景中文本的多样性、背景的复杂性等因素影响，自然场景文本识别任务的难度远大于扫描文档的文字识别。

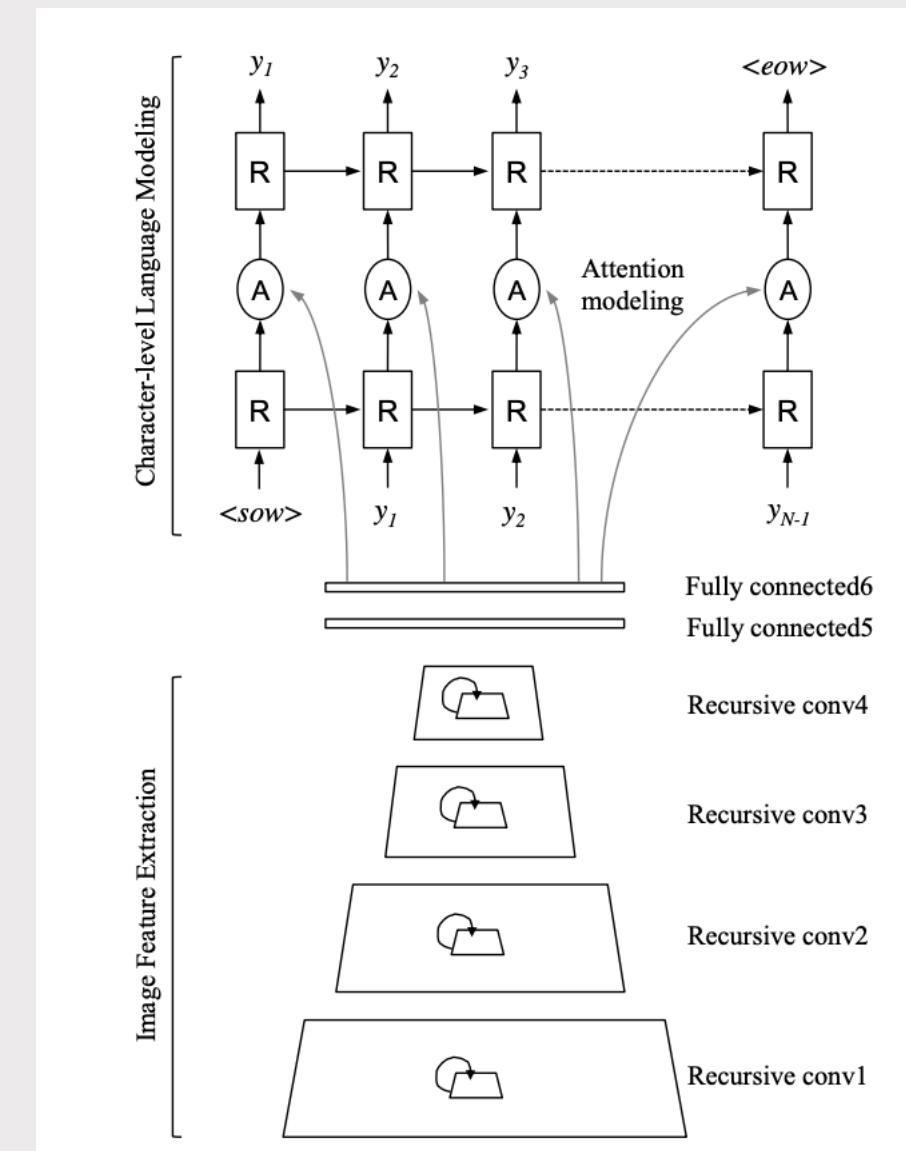


相关工作

1. 基于RNN的语言模型

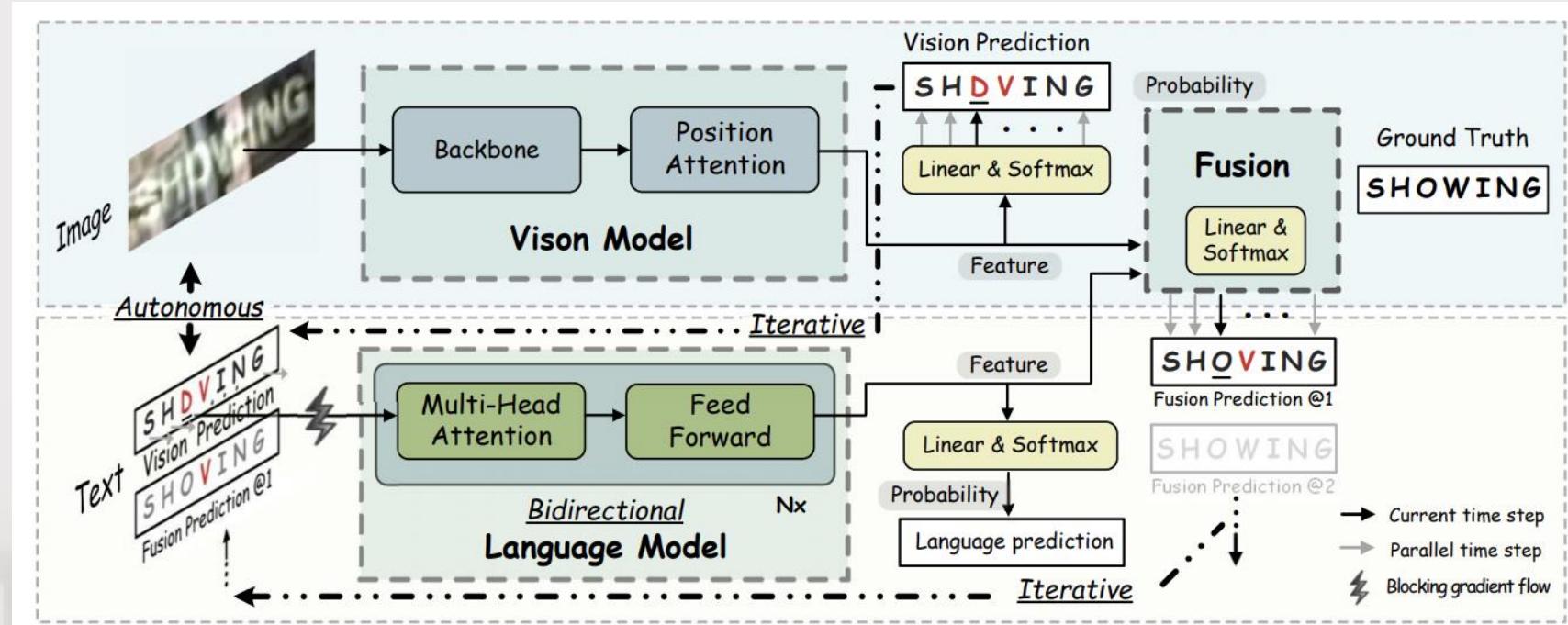
首先通过CNN对视觉特征提取，然后通过对每个时间步引入注意力的操作，使得每个时间步关注一个区域的视觉信息。通过参考之前时间步的信息进行字符预测。

优势：省去了复杂的n-gram语言建模方式。



2. 基于Transformer的语言模型

通过双向的语言模型建模，捕获更加鲁棒的语言信息。同时引入了迭代矫正的操作，进一步提升识别结果



优势：速度快、捕获的语言信息鲁棒。

主要贡献

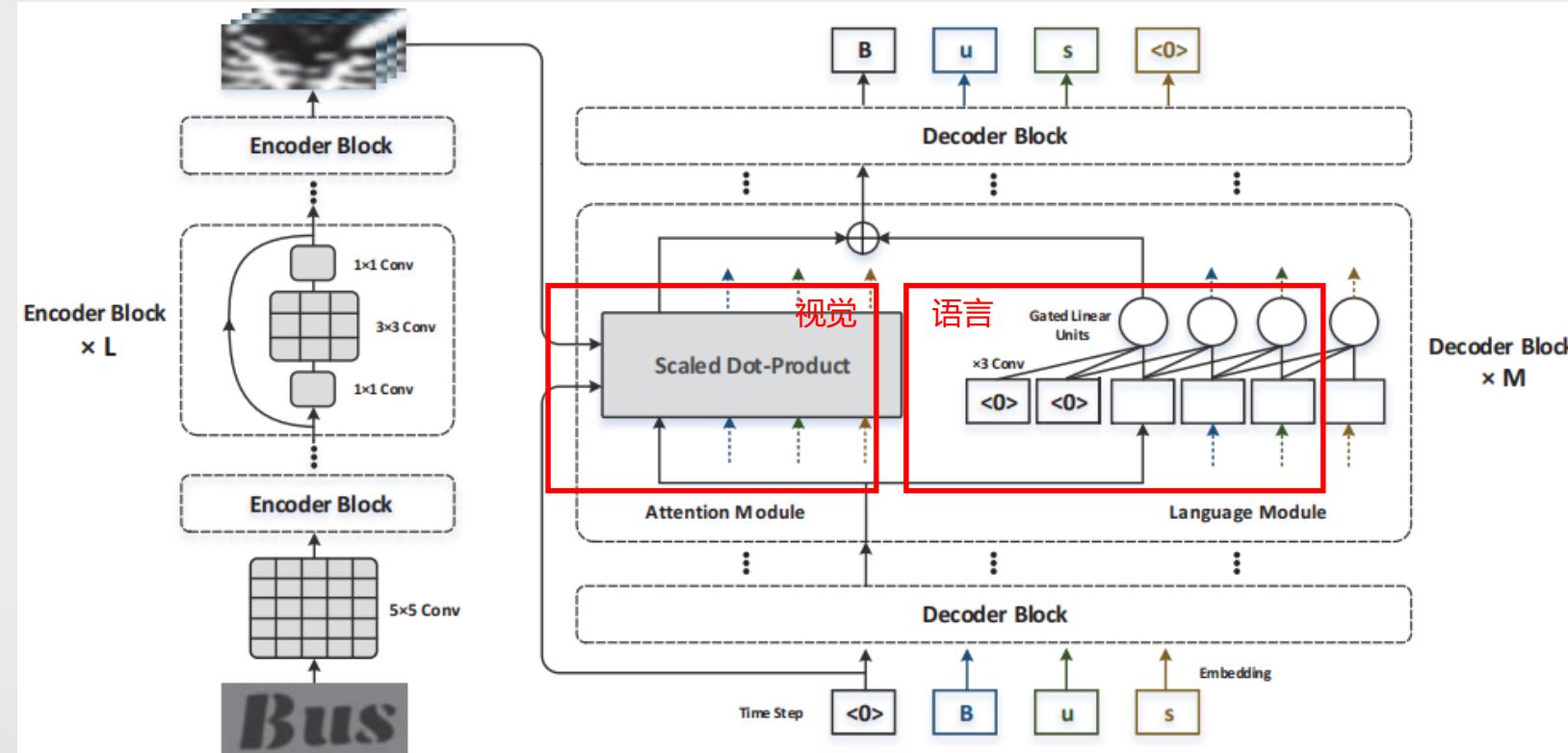
- 提出了一个全部基于CNN的文本识别模型，解决了RNN梯度消失的问题
- 提出新的联合考虑视觉信息和语言信息的方法，采用来自注意力模块和语言模块的多个loss同时监督，实现端到端训练。
- 在没有使用字典的情况下，该方法在SVT数据集上词准确度达到了83.9%



二、算法模型剖析



模型总览



该模型去除了RNN结构，仅仅使用CNN来完成。在解码器中同时考虑了视觉信息和语言信息。

模型总览

| layer name | conv1 | conv2_x | conv3_x | conv4_x | conv5_x | |
|------------|--|--|--|--|--|-------|
| input size | 32×100 | 16×50 | 8×25 | 8×25 | 8×25 | |
| block | $5 \times 5, 16, \text{stride } 2$ BN, ReLU | $1 \times 1, 16$ BN, ReLU $3 \times 3, 16$ BN, ReLU $1 \times 1, 64$ Shortcut BN, ReLU | n_1 $1 \times 1, 32$ BN, ReLU $3 \times 3, 32$ BN, ReLU $1 \times 1, 128$ Shortcut BN, ReLU | n_2 $1 \times 1, 64$ BN, ReLU $3 \times 3, 64$ BN, ReLU $1 \times 1, 256$ Shortcut BN, ReLU | n_3 $1 \times 1, 128$ BN, ReLU $3 \times 3, 128$ BN, ReLU $1 \times 1, 512$ Shortcut BN, [ReLU] | n_4 |

编码器具体结构

| layer name | embedding | conv_x | logit |
|------------|-----------|--|-------------------|
| dimension | d | d | f |
| block | Embedding | $\left[\begin{array}{c} \text{Conv1D,GLU,Shortcut,LN} \\ \text{Attention, Shortcut, LN} \\ \text{Add} \end{array} \right] \times n_5$ | Linear Softmax |

解码器具体结构



模型总览

基于CNN的attention

$$\mathbf{q}_k = \text{linear}(\mathbf{s}_k + \mathbf{s}_k^0),$$

$$b_{i,j,k} = \frac{\mathbf{x}_{i,j}^T \cdot \mathbf{q}_k}{\sqrt{d}},$$

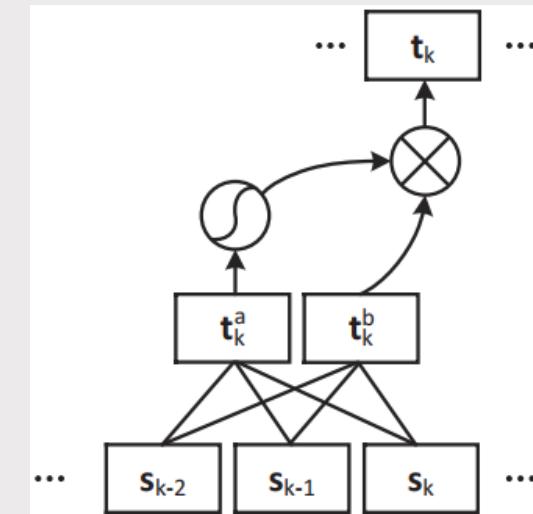
$$a_{i,j,k} = \frac{\exp(b_{i,j,k})}{\sum_{o=1,p=1}^{H,W} \exp(b_{o,p,k})}.$$

$$\mathbf{c}_k = \sum_{i=1,j=1}^{H,W} a_{i,j,k} \cdot \mathbf{x}_{i,j}.$$

$$\mathbf{c}_k = \text{norm}(\text{linear}(\mathbf{c}_k) + \mathbf{s}_k).$$



基于CNN的语言模型结构

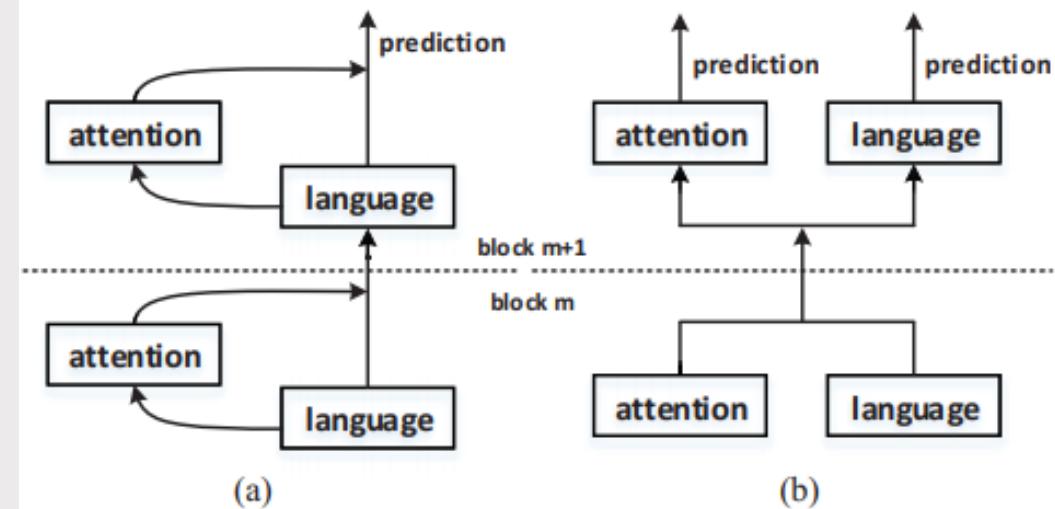


$$\begin{aligned} \mathbf{Y} &= [\mathbf{s}_{k-2}, \mathbf{s}_{k-1}, \mathbf{s}_k], \\ \mathbf{t}_k &= \sigma(\mathbf{Y} * \mathbf{W}^a + \mathbf{b}^a) \otimes (\mathbf{Y} * \mathbf{W}^b + \mathbf{b}^b), \end{aligned}$$

$$\mathbf{t}_k = \text{norm}(\mathbf{t}_k + \mathbf{s}_k).$$

算法创新点

1. 摒弃了RNN，整个模型仅使用CNN搭建而成，
2. 在解码器中同时考虑视觉信息和语言信息，通过对视觉和语言模型都加入监督实现视觉和语言的联合学习。



实验结果

| Type | Method | SVT | | IIIT5K | | IC03 | | IC13 | | IC15 | |
|-------------------|---------------------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| | | 50 | None | 50 | 1k | None | 50 | Full | None | None | None |
| I | Wang et al. [43] (ABBYY) | 35.0 | - | 24.3 | - | - | 56.0 | 55.0 | - | - | - |
| | Wang et al. [43] | 57.0 | - | - | - | - | 76.0 | 62.0 | - | - | - |
| | Mishra et al. [32] | 73.2 | - | 64.1 | 57.5 | - | 81.8 | 67.8 | - | - | - |
| | Alsharif and Pineau [2] | 74.3 | - | - | - | - | 93.1 | 88.6 | - | - | - |
| | Almazán et al. [1] | 89.2 | - | 91.2 | 82.1 | - | - | - | - | - | - |
| | Yao et al. [47] | 75.9 | - | 80.2 | 69.3 | - | 88.5 | 80.3 | - | - | - |
| | Jaderberg et al. [26] | 86.1 | - | - | - | - | 96.2 | 91.5 | - | - | - |
| | Su and Lu [40] | 83.0 | - | - | - | - | 92.0 | 82.0 | - | - | - |
| | Gordo [14] | 91.8 | - | 93.3 | 86.6 | - | - | - | - | - | - |
| II | Jaderberg et al. [25] (DICT) | 95.4 | 80.7 | 97.1 | 92.7 | - | 98.7 | 98.6 | 93.1 | 90.8 | - |
| | Jaderberg et al. [24] | 93.2 | 71.7 | 95.5 | 89.6 | - | 97.8 | 97.0 | 89.6 | 81.8 | - |
| | Shi et al. [38] | 96.4 | 80.8 | 97.6 | 94.4 | 78.2 | 98.7 | 97.6 | 89.4 | 86.7 | - |
| | Shi et al. [39] | 95.5 | 81.9 | 96.2 | 93.8 | 81.9 | 98.3 | 96.2 | 90.1 | 88.6 | - |
| | Lee and Osindero [29] | 96.3 | 80.7 | 96.8 | 94.4 | 78.4 | 97.9 | 97.0 | 88.7 | 90.0 | - |
| | Ghosh et al. [13] | 95.2 | 80.4 | - | - | - | 95.7 | 94.1 | 92.6 | - | - |
| | Ghosh et al. [13] (without ELM) | 91.7 | 75.1 | - | - | - | 93.4 | 91.0 | 89.3 | - | - |
| | Cheng et al. [6] | 95.7 | 82.2 | 98.9 | 96.8 | 83.7 | 98.5 | 96.7 | 91.5 | 89.4 | 63.3 |
| Our method | | 96.6 | 83.9 | 97.5 | 94.8 | 82.0 | 98.5 | 97.8 | 91.9 | 90.7 | 64.1 |



三、代码复现



复现思路

原则是先实现，再调优以达到论文要求。

我们首先在GitHub上下载我们自己开源的代码，在开源代码基础上进行调优。经过不断试错，最终选择一个基于PyTorch框架的开源代码。

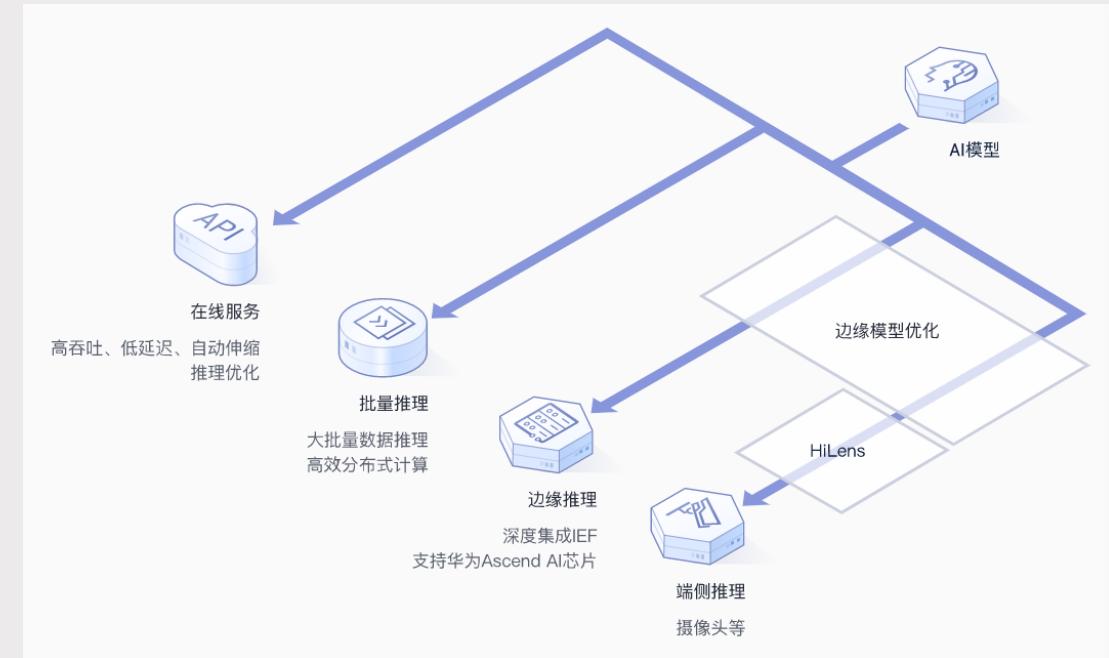
经过一些“小插曲”后，代码调通了，经过一些调优调整，在单卡和八卡训练下达到精度要求。



ModelArts环境介绍



模型构建

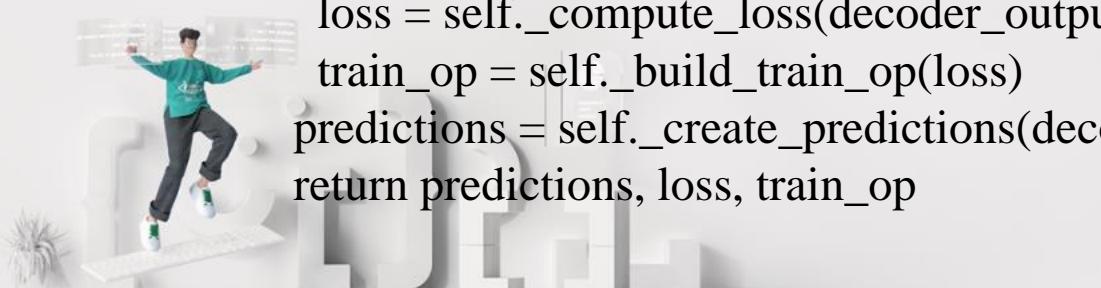


模型部署

代码讲解

1. Model主体: conv-ensemble-str/model/model.py

```
class Model(object):
    def __init__(self, params, mode):
        self.encoder = EncoderResnet(params, mode)
        self.decoder = DecoderConv(params, mode, self.charset.num_charset)
    def __call__(self, features, labels):
        with tf.variable_scope('model'):
            features, labels = self._preprocess(features, labels)      # tf.logging.info('Preprocess data.')
            encoder_output = self.encoder(features)                  # tf.logging.info('Create encoder.')
            decoder_output = self.decoder(encoder_output, labels) # tf.logging.info('Create decoder.')
            if self.mode == ModeKeys.TRAIN:
                loss = self._compute_loss(decoder_output, labels)    # tf.logging.info('Compute loss.')
                train_op = self._build_train_op(loss)
                predictions = self._create_predictions(decoder_output, features, labels)  tf.logging.info('Create predictions.')
            return predictions, loss, train_op
```



代码讲解

2. Encoder部分： conv-ensemble-str/model/encoder_resnet.py

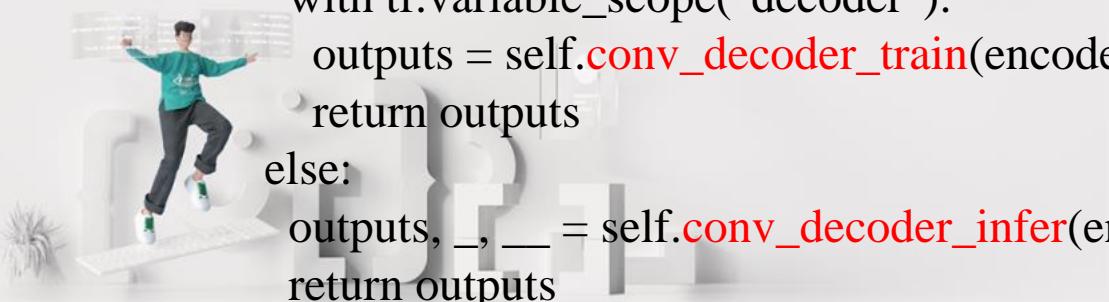
```
class EncoderResnet(object):
    def __call__(self, features):
        # conv1
        with arg_scope([layers_lib.conv2d], activation_fn=None, normalizer_fn=None):
            net = resnet_utils.conv2d_same(inputs, 16, 5, stride=2, scope='conv1')
        # resnet blocks
        blocks = []
        for i in range(len(self.encoder_params['block_name'])):
            block = resnet_v2.resnet_v2_block(scope=self.encoder_params['block_name'][i],
                                              base_depth=self.encoder_params['base_depth'][i], num_units=self.encoder_params['num_units'][i],
                                              stride=self.encoder_params['stride'][i])
            blocks.append(block)
        net, _ = resnet_v2.resnet_v2(net, blocks, is_training=(self.mode == ModeKeys.TRAIN),
                                     global_pool=False, output_stride=2, include_root_block=False, scope='resnet')
    return net
```

代码讲解

3. Decoder部分：conv-ensemble-str/model/decoder_conv.py

```
class DecoderConv(object):
    def __init__(self, params, mode, num_charset):
        self.params = params
        self.params.update(DECODER_DEFUALT_PARAM)
        self.mode = mode
        self.num_charset = num_charset
        self.max_sequence_length = self.params['dataset']['max_sequence_length']

    def __call__(self, encoder_output, labels):
        if self.mode == ModeKeys.TRAIN:
            with tf.variable_scope("decoder"):
                outputs = self.conv_decoder_train(encoder_output, labels)
            return outputs
        else:
            outputs, _, __ = self.conv_decoder_infer(encoder_output)
        return outputs
```



代码讲解

4. Decoder部分，卷积语言模型：conv-ensemble-str/model/decoder_conv.py

```
class ConvBlock(object):
    def __call__(self, encoder_output, input_embed):
        # 1D convolution
        for layer_idx in range(self.params['cnn_layers']):
            with tf.variable_scope("conv" + str(layer_idx)):
                # language module .....
                # shortcut and layer norm .....
                # attention module .....
                # shortcut and layer norm .....
            next_layer = language_layer + attention_layer
```

```
language_logit, scores = self.create_logit(language_layer,...)
attention_logit, scores = self.create_logit(attention_layer,...)
return language_logit, attention_logit, scores
```



代码讲解

5. 基于CNN的language模块: conv-ensemble-str/model/decoder_conv.py

```
language_layer = self.conv1d_weightnorm(  
    inputs=language_layer,  
    out_dim=nout * 2,  
    kernel_size=kernal_width,  
    padding="VALID",  
    output_collection=output_collection)  
# to avoid using future information  
language_layer = language_layer[:, 0:-kernal_width + 1, :]  
  
# add GLU  
language_layer = self.gated_linear_units(language_layer, output_collection)
```



代码讲解

6. attention模块: conv-ensemble-str/model/decoder_conv.py

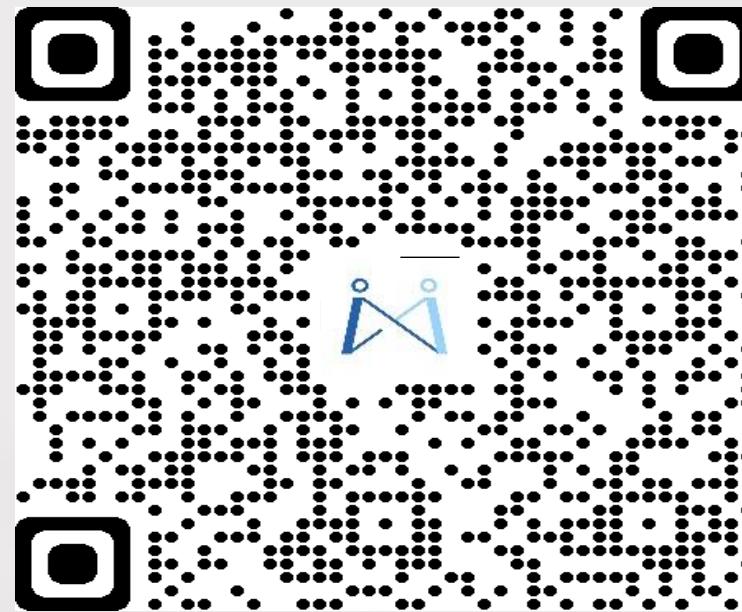
```
def attention_score(self, dec_rep, encoder_output):
    N, H, W, C = encoder_output.get_shape().as_list()
    N = N or tf.shape(dec_rep)[0]
    M = dec_rep.get_shape().as_list()[1] or tf.shape(dec_rep)[1]
    encoder_reshape = tf.reshape(encoder_output, [N, H * W, C]) # N*(H*W)*C
    # N*M*C ** N*(H*W)*C --> N*M*(H*W)
    att_score = tf.matmul(dec_rep, encoder_reshape, transpose_b=True) * tf.sqrt(1.0 / C)
    att_score = tf.nn.softmax(att_score) # N*M*(H*W)
    # N*M*(H*W) ** N*(H*W)*C --> N*M*C
    att_out = tf.matmul(att_score, encoder_reshape)
    att_score = tf.reshape(att_score, [N, M, H, W])
    return att_out, att_score
```





总结

感谢华为云ModelArts强大的硬件支持！



扫码获取算法源代码





Thank You!

