运行环境

Python版本：Python 3.5

框架版本：PaddlePaddle 1.5.1

GPU：Tesla V100

显存：16GB

CPU：8核

RAM：32GB

运行方法

预测测试集数据 运行预测脚本文件

```
sh infer.sh
```

首先解压缩测试集到work目录下，再执行预测文件infer.py

网络模型

官方提供的crnn baseline模型，先用CNN提取图片的feature map，再将feature输入RNN对图片进行解码，将识别问题转化为序列问题

标签映射

因为最终的评价指标不考虑字符、全角半角以及大小写，加上本身汉字种类就很多，在做标签映射的时候就对label进行了一些预处理，进行了简繁体转换然后去掉了一些字符，代码实现如下所示，最终加上空白符"_"，一共有3862个类别

```
def _normalize_text(text):
    res = ""
    print('raw text is %s' % text)
    for c in text:
        inside_code = ord(c)
        if inside_code == 12288:
            continue
        if (inside_code >= 65281 and inside_code <= 65374):
            inside_code -= 65248
             if inside_code >= 48 and inside_code <= 57:
                 res += chr(inside_code)
             if inside_code >= 65 and inside_code <= 90:
                 res += chr(inside_code+32)
             if inside_code >= 97 and inside_code <= 122:
                 res += chr(inside_code)
         else:
             res += Converter('zh-hans').convert(chr(inside_code))
     print('after transfer is %s' % res)
     return res
```

数据增强

为了增强模型的泛化性能，在测试集上有更好的表现，在训练过程中随机对图片进行了增强处理，代码如

下：

```
class ImageTransfer(object):
    """crop, add noise, change contrast, color jittering"""
    def __init__(self, image):
        """image: a ndarray with size [h, w, 3]"""
        self.image = image

    def slight_crop(self):
        h, w = self.image.shape[:2]
        k = random.random() * 0.08  # 0.0 <= k <= 0.1
         ch, cw = int(h * 0.9), int(w - k * h)     # cropped h and w
         hs = random.randint(0, h - ch)      # started loc
         ws = random.randint(0, w - cw)
         return self.image[hs:hs+ch, ws:ws+cw]

     def add_noise(self):
         img = self.image * (np.random.rand(*self.image.shape) * 0.2 + 0.8)
         img = img.astype(np.uint8)
         return img

     def change_contrast(self):
         # if random.random() < 0.5:
         #     k = random.randint(7, 9) / 10.0
         # else:
         #     k = random.randint(11, 13) / 10.0
         # b = 128 * (k - 1)
         # img = self.image.astype(np.float)
         # img = k * img - b
         # img = np.maximum(img, 0)
         # img = np.minimum(img, 255)
         # img = img.astype(np.uint8)
         r = np.random.RandomState(0)
         clahe = cv2.createCLAHE(clipLimit=round(r.uniform(0, 2), 2), tileGridSiz
         b, g, r = cv2.split(self.image)
         b1 = clahe.apply(b)
         g1 = clahe.apply(g)
         r1 = clahe.apply(r)
         img = cv2.merge((b1, g1, r1))
         return img

     def perspective_transform(self):
         h, w = self.image.shape[:2]
         short = min(h, w)
         gate = int(short * 0.1)
```

```
        mrg = []
        for _ in range(8):
            mrg.append(random.randint(0, gate))
        pts1 = np.float32(
            [[mrg[0], mrg[1]], [w - 1 - mrg[2], mrg[3]], [mrg[4], h - 1 - mrg[5]
        pts2 = np.float32([[0, 0], [w - 1, 0], [0, h - 1], [w - 1, h - 1]])
        M = cv2.getPerspectiveTransform(pts1, pts2)
        return cv2.warpPerspective(self.image, M, (w, h))


    def gamma_transform(self, a=1.0, gamma=2.0):
        image = self.image.astype(np.float)
        image = image / 255
        image = a * (image ** gamma)
        image = image * 255
        image = np.minimum(image, 255)
        image = image.astype(np.uint8)
        return image

    def change_hsv(self):
        img = cv2.cvtColor(self.image, cv2.COLOR_BGR2HSV)
        s = random.random()
        def ch_h():
            dh = random.randint(2, 11) * random.randrange(-1, 2, 2)
            img[:, :, 0] = (img[:, :, 0] + dh) % 180
        def ch_s():
            ds = random.random() * 0.25 + 0.7
            img[:, :, 1] = ds * img[:, :, 1]
        def ch_v():
            dv = random.random() * 0.4 + 0.6
            img[:, :, 2] = dv * img[:, :, 2]
        if s < 0.25:
            ch_h()
        elif s < 0.50:
            ch_s()
        elif s < 0.75:
            ch_v()
        else:
            ch_h()
            ch_s()
            ch_v()
        return cv2.cvtColor(img, cv2.COLOR_HSV2BGR)

def random_augmentation(image, allow_crop=True):
```

```
    f = ImageTransfer(image)
    seed = random.randint(0, 6)     # 0: original image used
    switcher = random.random() if allow_crop else 1.0
    if seed == 1:
        image = f.add_noise()
    elif seed == 2:
        image = f.change_contrast()
    elif seed == 3:
        image = f.change_hsv()
    elif seed == 4:
        a = random.random() * 0.4 + 0.8
        gamma = random.random()
         image = f.gamma_transform(a=a, gamma=gamma)
     elif seed >= 5:
         f1 = ImageTransfer(f.add_noise())
         f2 = ImageTransfer(f1.change_hsv())
         f3 = ImageTransfer(f2.gamma_transform(1.0, 1.5))
         image = f3.change_contrast()
     if switcher < 0.4:
         fn = ImageTransfer(image)
         image = fn.slight_crop()
     elif switcher < 0.7:
         fn = ImageTransfer(image)
         image = fn.perspective_transform()
     return image

```

预训练

最终提交的结果模型准确率达到57.149%，是在合成数据集上进行预训练之后再在比赛数据集上finetune。