DeepFake 检测实验报告

一.实验目的

实现 *Multi-task Learning For Detecting and Segmenting Manipulated Facial Images and Videos* 论文提出的方法，复现该模型在 FaceForensics 和 FaceForensics++数据集上的运行结果，检验该方法用于 Deepfake 视频检测（包括造假率判定以及造假区域分割）中达到的效果。

二.论文原理

1.模型结构

论文主要提出了一种 Y 型自动编码器模型，其在两个总体分支上分别实现了对于视频伪造概率的确定、视频伪造区域的分割和重建原视频的功能。其中分割和重建功能是在其中一个总体分支上进一步划分成两个分支分别完成。使用 Y 型结构使得不同分支上的任务可以在早期阶段共享信息，从而提高模型多任务学习的性能。

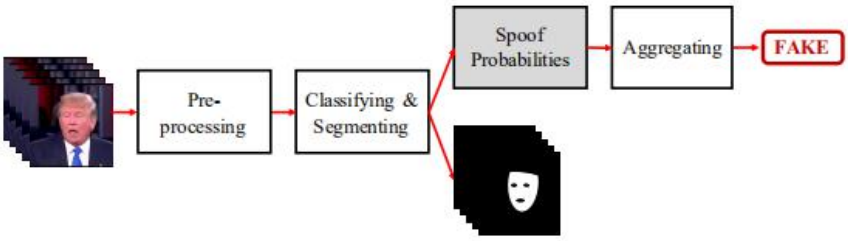

Figure 1 Y 型自动编码器结构

2.计算网络

整个模型将视频提取为帧序列，对于每一帧将经过预处理提取出的人脸区域调整为(256, 256)大小后作为网络的输入。此后输入被送入全连接卷积神经网络进行处理。

在 Y 型自动编码器的第一阶段(即多任务共有阶段)，使用(3,3)大小的卷积核进行卷积运算，每一个卷积层后紧接着一个批正则化层和一个 ReLU 激活函数层，这三者共同构成网络结构中的一个全连接层，网络中各层卷积核的步长在 1 和 2 之间进行交替。第一阶段由 9 个全连接层构成。在第一阶段完成后，视频帧被转化为隐空间特征。

对隐空间特征计算激活损失：

$$L_{act}=\frac{1}{N}\sum\left|a_{i,1}-y_i\right|+\left|a_{i,0}-(1-y_i)\right|,\ y_i\in\{0,1\}\qquad(1)$$

其中 y_i 是对于样本是否造假的标签，N 是样本数量，$a_{i,0}$ 和 $a_{i,1}$ 分别是对应标签下隐空间特征矩阵的 L_1 范式，即：

$$a_{i,c}=\frac{1}{2K}\left\|h_{i,c}\right\|_1,\ c\in\{0,1\}\qquad(2)$$

通过激活损失给出视频帧是否造假的分类结果以及造假的概率（最终视频造假的概率取为所有视频帧造价概率的平均值），并且对于标签为 c 的样本，强制使得 $a_{i,c}=0$，从而确保隐空间中只有对应标签 c 的特征能够被激活用于第二阶段的处理。

在第二阶段的网络中，使用(3,3)大小的卷积核进行反卷积运算，每一个反卷积层后紧接着一个批正则化层和一个 ReLU 激活函数层，网络中各层卷积核的步长依然是在 1 和 2 之间进行交替。在经过 4 个全连接层后分化为两个小分支，分别再经过 4 个全连接层。

最后对于造假区域分割分支，通过 Softmax 函数生成分割图像的 mask，将计算得到的 mask 与 ground-truth 通过交叉熵损失函数进行一致性比对，计算方法如下：

$$L_{seg}=\frac{1}{N}\sum_i\left\|m_i\log(s_i)+(1-m_i)\log(1-s_i)\right\|_1\qquad(3)$$

其中 m 为 ground-truth，s 为 Softmax 函数生成分割图像的 mask。

对于重建原视频帧分支，通过 Tanh 函数生成重建帧图像，之后计算原视频帧和重建帧的 L_2 距离得到损失函数，即：

$$L_{rec}=\frac{1}{N}\sum_i\left\|x_i-\hat{x}_i\right\|_2\qquad(4)$$

其中 x 为原视频帧，$\hat{x}$ 为重建帧。

由于在该多任务学习方法中，分类任务（判定造假概率）和分割任务（识别造假区域）被认为同等重要，且视频帧重建对分割任务起到了重要作用，因此对于上述三种损失函数采用平均加权的计算方法得到对模型起总体评价作用的损失函数，即：

$$L=\gamma_{act}L_{act}+\gamma_{seg}L_{seg}+\gamma_{rec}L_{rec},\ \gamma_{act}=\gamma_{seg}=\gamma_{rec}\qquad(5)$$

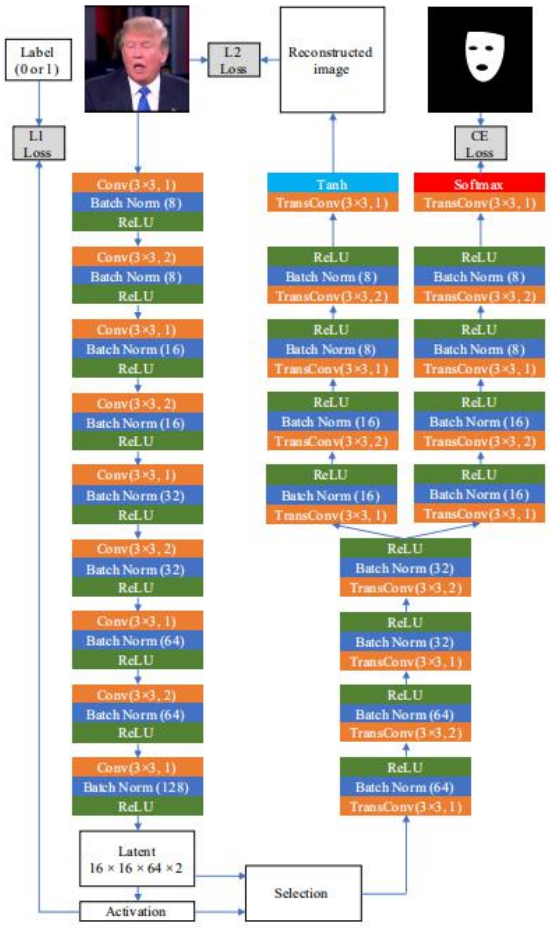

Figure 2 计算网络

三. 论文复现

1. 数据集生成

从 FaceForensics++上下载数据，每个视频对应以下三个文件，逐帧处理，将问题转化到图像层面。

Figure 3 原视频、换脸视频、Mask

裁剪 mask 附近（被换脸区域附近）区域，并调整至模型输入大小(256, 256)，拼接图像与对应的 mask，同时作为分类和分割任务的输入。

Figure 4裁剪后图像

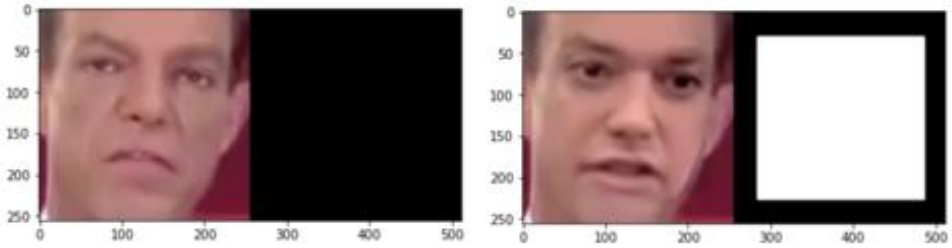

Figure 5拼接后的原图和换脸图

2.论文中的实验部分

实验数据

Table 1. Design of training and testing datasets.

Name	Source dataset	Description	Manipulation Method	Number of Videos
Training	FaceForensics **Source-to-Target**	Training set used for all tests	Face2Face	704 ×2
Test 1	FaceForensics **Source-to-Target**	Match condition for seen attack	Face2Face	150 ×2
Test 2	FaceForensics **Self-Reenactment**	Mismatch condition for seen attack	Face2Face	150 ×2
Test 3	FaceForensics++ **Deepfakes**	Unseen attack (deep-learning-based)	Deepfakes	140 ×2
Test 4	FaceForensics++ **FaceSwap**	Unseen attack (computer-graphics-based)	FaceSwap	140 ×2

训练集使用 Face2Face Source-to-Target 数据集中的 704 个换脸后视频，704 个原视频，每个视频中取前 200 帧。

原论文的实验结果如下表所示

测试集	分类准确率(%)	分割准确率(%)
Test1	92.77	92.77
Test2	92.50	90.20
Test3	52.32	70.37
Test4	54.07	84.67

Finetune

原文中使用 Faceswap 数据进行模型的微调，正负类各 100 个视频，训练 50Epoch 在 Test 4 上的实验结果。

模型	分类准确率(%)	分割准确率(%)
Proposed New	54.07	84.67
Finetune Proposed New	83.71 (↑ 29.64)	93.01 (↑ 8.34)

3.实验结果分析

可以看到在与训练集使用相同方法生成换脸视频的测试集上分类和分割准确率都达到了不错的水平，但在通过其他方式生成的换脸是视频上效果一般。
这说明模型的鲁棒性很差。除有模型本身产生的原因外，在生成数据集的阶段，每个视频只采用前 200 帧。每两帧之间差异很小。尤其对于新闻中的人脸，位置基本固定，不同帧时间的差异微乎其微。所以，虽然训练集的数据量很多但效果可能重复性也很强。
在使用少量新训练集进行微调后在新的任务上效果有了不错的提升。

4..复现

复现阶段实现了数据集的下载与生成。值得注意的是，在这篇论文发表的时间，FaceForensics++并没有提供 Deepfakes 和 Faceswap 换脸数据的 Mask（论文中的训练集选用 Face2Face Source-to-Target 的原因可能也是如此），在开放的代码中 Mask 是通过其他方式标记的。但现在可以直接通过 FaceForensics++进行下载。

前面的实验(预训练)使用的数据集规模为 2×704×200 过于庞大，复现并不现实。因此本次实验只针对 Finetune 部分进行复现。

原文中只记录了在 Faceswap 上 Finetune 的结果，复现过程中使用 Deepfakes 数据进行 Finetune。

使用 Deepfakes 数据，在正负类各 100 个视频，每个视频抽取 10 帧，训练 50Epoch。这里做出的改进是在视频中等间距的抽取 10 帧（间距为总帧数/10），使得同一个视频中抽取的不同帧有更大的差异。并在选用 150 个视频，每个视频抽取 2 帧作为测试集。另外，虽然原文中也是在 150 个视频上进行验证，但并没有指出是数据集中的哪 150 个，所以本次实验中选用的测试集与原文中并不相同，关于准确率的提升仅供参考。

数据集下载:

使用 FaceForensics++上的数据需要先提交申请，获取 download 脚本

执行脚本，下载 Deepfakes 数据

```
python download++.py datasets/ -d original -c c23 -t videos
```

```
python download++.py datasets/ -d Deepfakes -c c23 -t masks
python download++.py datasets/ -d Deepfakes -c c23 -t videos
```

生成训练集，执行 create_dataset_Deepfakes.py 脚本

```
python create_dataset_Deepfakes.py\
    --input_real datasets/original_sequences/youtube/c23/videos\
    --input_fake datasets/manipulated_sequences/Deepfakes/c23/videos\
    --mask datasets/manipulated_sequences/Deepfakes/masks/videos\
    --output_real datasets/finetune/train/original\
    --output_fake datasets/finetune/train/altered
```

修改 create_dataset_Deepfakes.py 中逐帧抽取部分

```
vidcap_real.set(cv2.CAP_PROP_POS_FRAMES, start_frame_number)
success_real, image_real = vidcap_real.read()
vidcap_fake.set(cv2.CAP_PROP_POS_FRAMES, start_frame_number)
success_fake, image_fake = vidcap_fake.read()
vidcap_mask.set(cv2.CAP_PROP_POS_FRAMES, start_frame_number)
success_mask, image_mask = vidcap_mask.read()
start_frame_number += pass_img_num
```

使用生成的 mask 替代原实验中标记的 mask

```
#image_mask=cv2.imread(os.path.join(input_mask,filename, '0000.png'))
vidcap_mask = cv2.VideoCapture(os.path.join(input_mask, f))
success_mask, image_mask = vidcap_mask.read()
```

训练(Finetune)

```
python finetune.py\
    --dataset datasets/finetune \
    --train_set train \
    --val_set validation \
    --outf checkpoints/finetune \
    --batchSize 64 --niter 50
```

5.实验结果

训练集上的三个损失函数的衰减情况

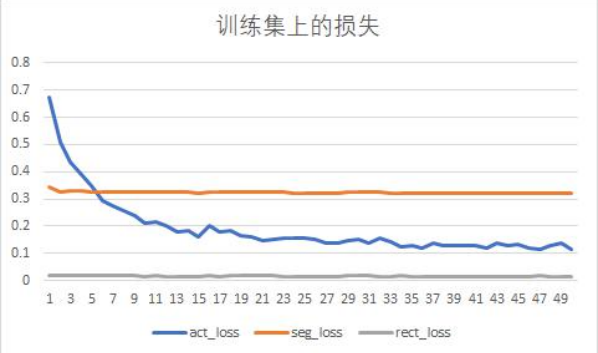

Figure 6 训练集上的损失函数

在 Deepfakes 测试集上的准确率提升

如原文中所述，在少量训练集上就可以有不错的提升，但最终模型的分类准确率也只有 74.64 并没有很高。

模型	分类准确率(%)	分割准确率(%)
Proposed New	52.32	70.37
Our Finetune	74.64 (↑ 21.32)	83.83 (↑ 13.46)

四. 总结

从实验结果来看，使用论文提供的模型在我们从 FaceForensics 上下载的 FaceToFace 数据集上的测试准确率达到了 90%，而在其他数据集上的测试准确率只有大概 50%，说明该模型训练过程中存在着过拟合的情况，泛化能力较差。

且我们认为论文中提出的某些论点缺乏可信的依据。

如作者认为在该多任务学习方法中，分类任务（判定造假概率）和分割任务（识别造假区域）同等重要，且视频帧重建对分割任务起到了重要作用，因此对于三种损失函数采用平均加权的计算方法，而实际上这三种损失函数存在着未归一化的问题，实际训练过程中只有激活损失存在明显的下降现象，而其他两种损失基本保持不变。

同时作者在训练过程中使用了两种 FaceToFace 的数据类型：即造假视频和原视频中人脸来源于不同对象以及造假视频和原视频中人脸来源于相同对象，对于后者我们认为在实际训练过程中对模型改进的意义并不大。

由于时间有限，对于上述问题我们未能做出更深入的探讨和研究，将会在未来继续对作者提出的模型尝试改进和完善。