复现报告

吕温和 PB19051177

一、 文章的贡献与创新

绝大多数的视网膜血管图像数据集都是小样本集，《SA-UNet: Spatial Attention U-Net for Retinal Vessel Segmentation》这篇文章的贡献在于使用数据增强(data augmentation)的方法扩展了数据集大小，提高了神经网络拟合的准确性；而为了防止训练集的过拟合(overfitting)，文章在 U-Net 网络结构的基础上将卷积模块替换为有丢弃的(dropout)卷积模块，将 DropBlock 和批量标准化加入网络的主干，效果如下图(右一：过拟合消失)；文章还在网络主干

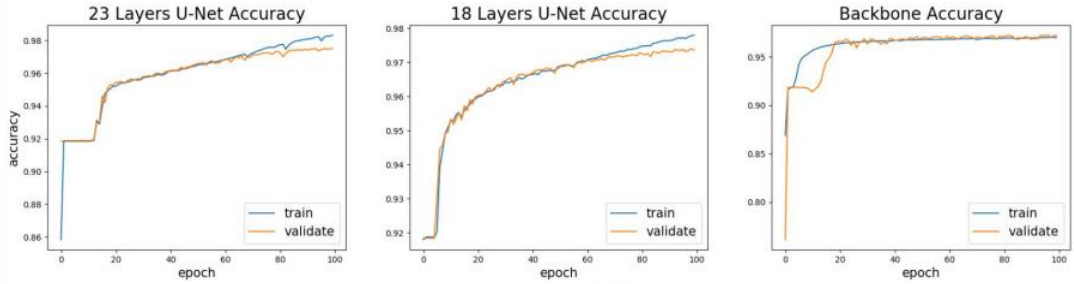


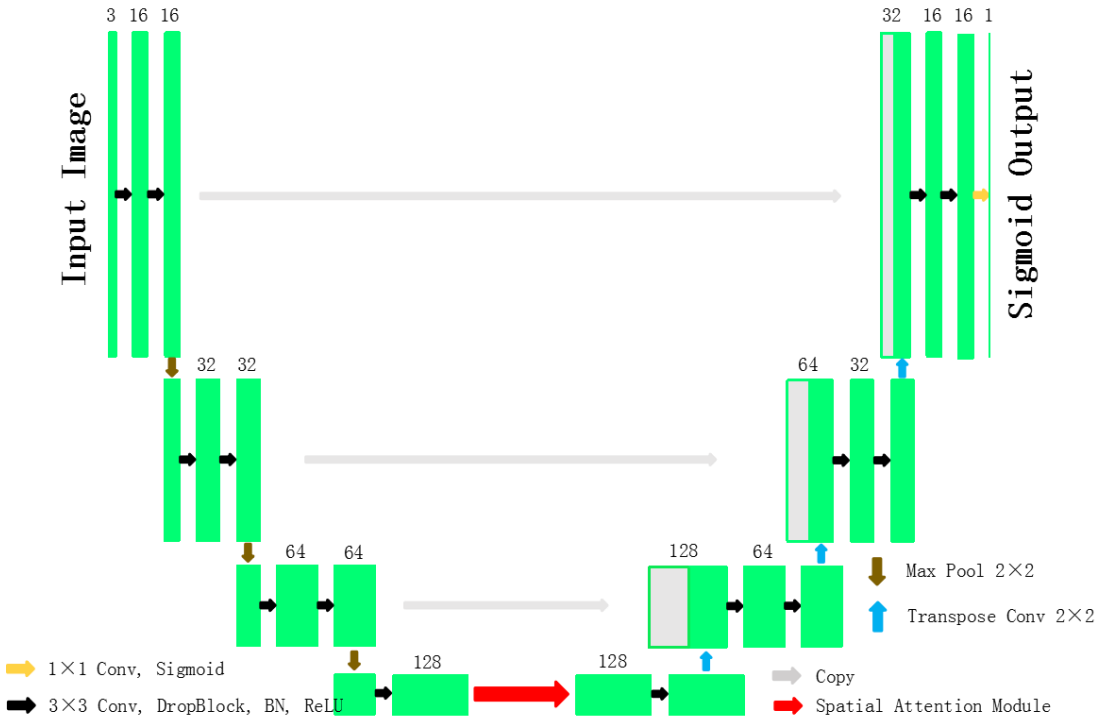
Fig. 2. Comparison of different models training 100 epochs on DRIVE.

的编码器和解码器之间加入一个空间注意模块(spatial attention module)，较好的解决了视网膜血管图像中血管和背景噪音难以区分的问题。该网络最终被称为 SA-Unet (Spatial Attention U-Net)，文章在两个视网膜血管数据集上比较了 SA-Unet 和传统 Unet、SD-Unet 等模型的表现差异，结果 SA-Unet 在准确率、敏感性、F1 分数等指标上优于其他传统，说明 SA-Unet 可以作为一种有效的血管图像分割的网络来使用，更加有效的进行图像分割处理。

二、 代码结构描述

Github 上，文章作者给出了如下几个代码文件：filp.py，keras_dataAug.py，SA_Unet.py，Spatail_Attention.py，Dropblock.py，util.py 以及两个训练数据集的文件。

文件作用描述：filp.py 的作用是将训练集原始数据进行图像水平、垂直、对角的翻转，而 keras_dataAug.py 将原始数据添加随机的高斯噪音、随机的转动、随机的颜色改变得到更多数据；util.py 将图像数据裁剪成合适的大小；SA_Unet.py 是网络结构的代码，它的模型层次结构图在 github 中，SA_Unet.py 调用了 Spatail_Attention.py，相当于在 U 型网络的底端引入了空间注意力的模块，而 Dropblock.py 文件起的作用是丢弃一些神经网络不参与训练，每个卷积层之后是一个 DropBlock，一个批标准化（BN）



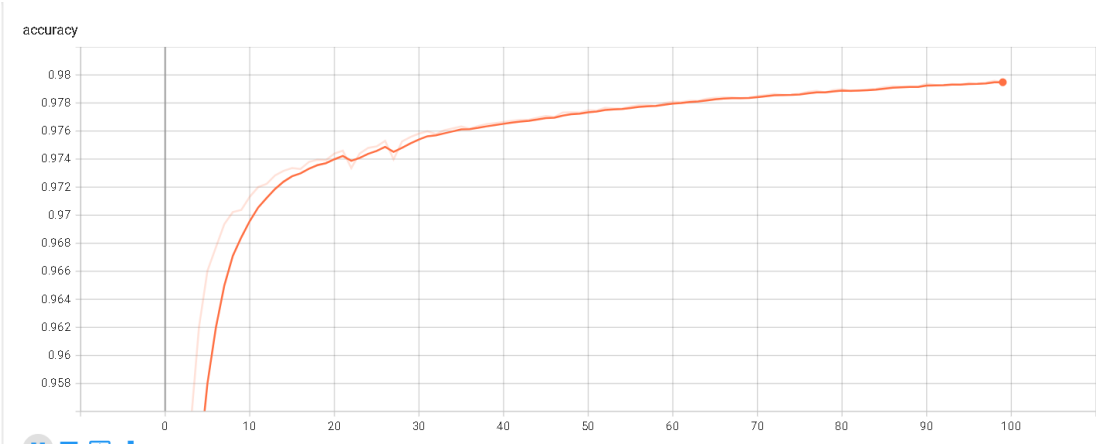
层和一个 ReLU 激活函数层。最终是一个具有 U 形编码器（左侧）-解码器（右侧）结构的 SA-Unet。如上图。

两个训练数据集的文件 Train_chase.py,Train_drive.py，它们先执行数据导入、再调整图像大小（reshape、resize）、最后导入模型并训练。

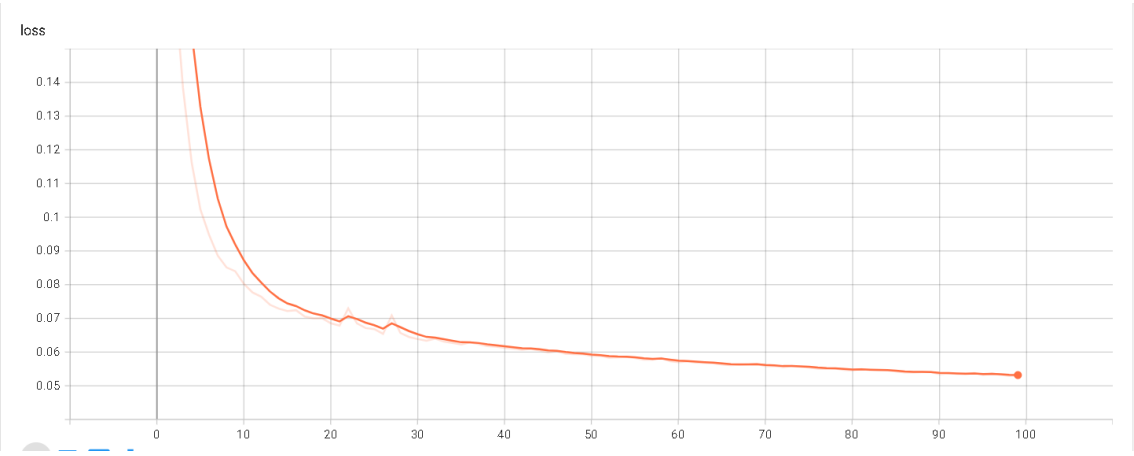
三、 模型拟合训练集

Epochs=100，batch_size=4，shuffle=True

精度：随着时间上升，且增速越来越慢。向 0.98 趋近。



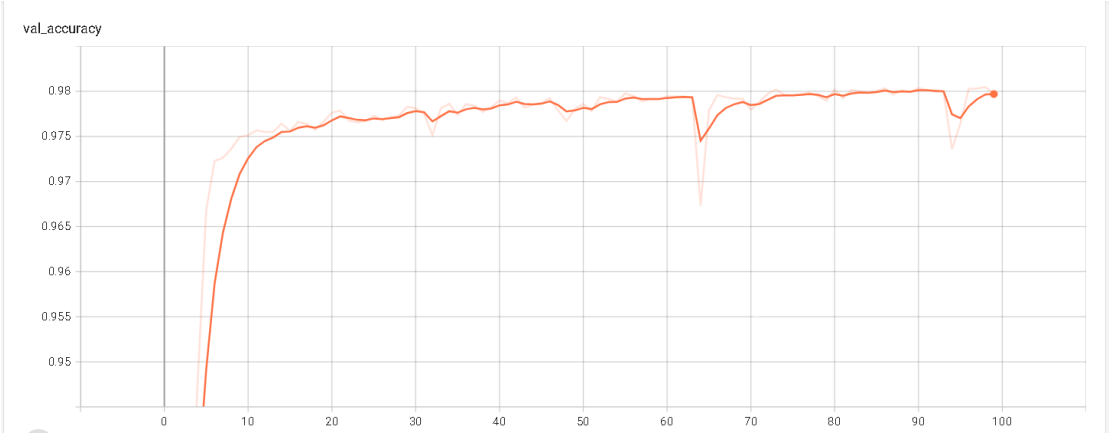
损失：随着时间减小，且减小的越来越慢。



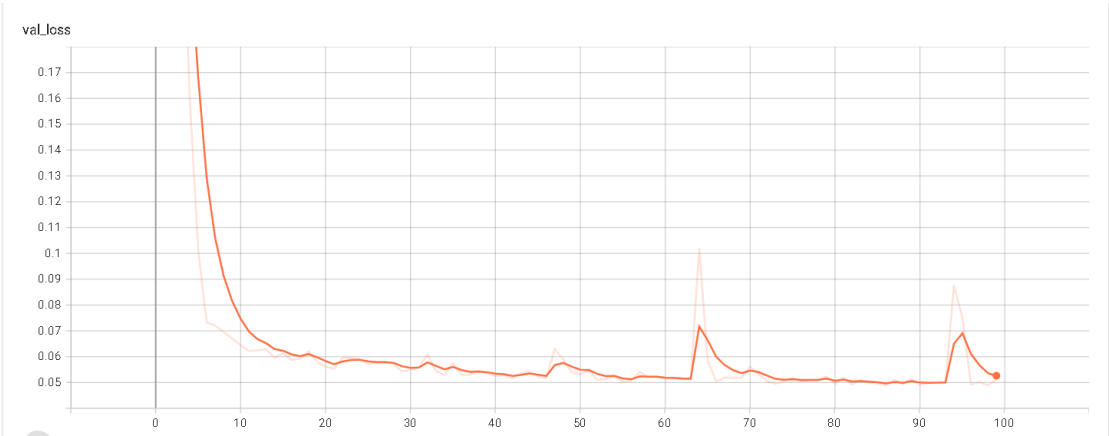
四、 模型拟合验证集

Epochs=100，batch_size=4，shuffle=True

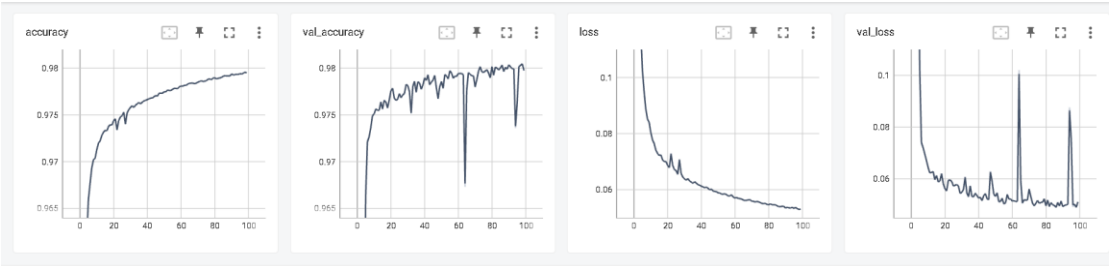
精度：随着时间增大。对比训练集，可以看出没有过拟合。



损失：随着时间减小，且减小的越来越慢。无过拟合。

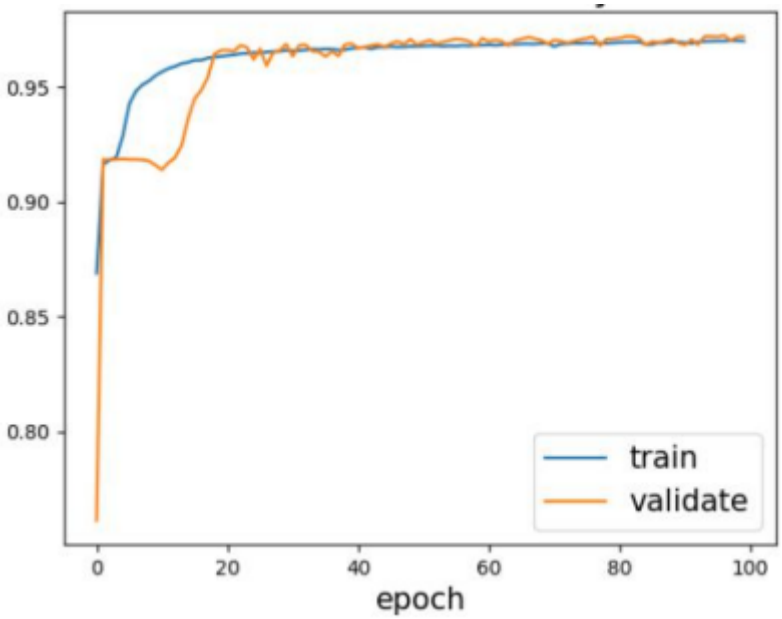


Smooth=0，不平滑曲线。

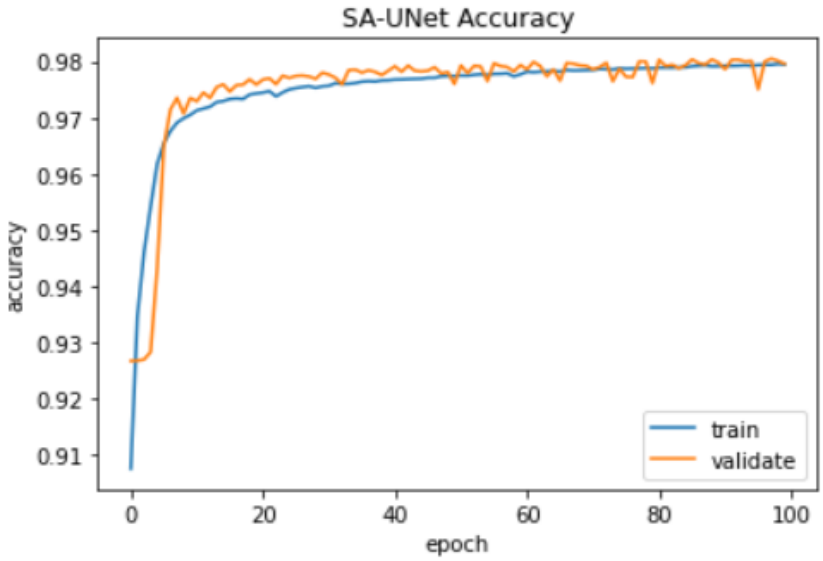


五、　复现结果与文章结果对比

文章的训练集、验证集精度:



复现的训练集、验证集精度:



对比后发现，文章的验证集在前十个 epochs 精度并没有提升，而训练集是正常的精度提升，可能是前十个 epochs 存在过拟合，并在后面的 epochs 中过拟合被慢慢矫正。但是我自己训练的结果看出:验证集的精度总比训练集的精度略大，可能是我复现的代码还是有一点过拟合的影响。

六、 复现过程的心得与吐槽

整个复现的过程太累了，主要是和考试周一起来的。刚开始看完文章，理解完作者的工作后，就马上在 github 上下载代码来跑了，在我之前就配置好的 tensorflow2.8 的版本下。然而作者的 tensorflow 版本还有其他依赖的版本是比较低的，所以一开始就遇到很多由于版本改动导致的导入模块不存在、函数名改变了、函数参数改变、类的继承关系变了等问题。解决这些问题后又遇到了代码中数据类型 Tensor 与 numpy.array 混用的问题，改的我心力憔悴，最后就是卡在一个 tf.Variable 的 debug 上怎么改也改不好，就打算休息一两天，再来 debug，但没能成功，放弃了。之后我又上 github 在文章作者那找到了他当时用的各种依赖的版本，然后就重新在 anaconda 里面配置环境，尽量与作者用的保持一致，感觉最后能够做出来就是因为用了一样的版本，花了完整一天去认真配置，再运行代码就可以比较顺利的运行啦，最后自己在 tensorboard 中输出训练过程的精度、损失就算是完成了。

七、##日常 debug 记录（发疯……）

（scipy.misc.pilutil 报错" No module named 'scipy.misc.pilutil' "，导致 imread 函数用不了，解决：scipy.misc.imread — SciPy v1.2.1 Reference Guide）

（离谱，data 增强后不知道输出去哪里）

（from keras.utils.conv_utils import normalize_data_format 可以用）

（什么 keras 版本，太老了）

（AttributeError: 'KerasTensor' object has no attribute '_keras_shape'）。。。。。解决方法：将_keras_shape 修改为 shape。

心累，运行 keras.models.Model 时 TypeError: ('Keyword argument not understood:', 'input')解决 (keras 版本问题，把参数 input=改为 inputs=、output=改为 outputs=就行)

opt = Adam(learning_rate=0.01, beta_1=0.85, beta_2=0.999) 出现错误：NameError: name 'Adam' is not defined 这是 TensorFlow 版本不一样导致的问题

from keras.optimizer_v2 import adam as adam_v2

#'adam': adam_v2.Adam

AttributeError: module 'h5py' has no attribute 'File'解决：重装 h5py 即可

OperatorNotAllowedInGraphError: using a `tf.Tensor` as a Python `bool` is not allowed: AutoGraph did convert this function. This might indicate you are trying to use an unsupported feature.不能解决：将输入的 Tensor 转化为 numpy,用 tensor.eval()，不过得先启动 tf.compat.v1session()，这和 tensorflow 版本有关。

寄了，不行。换个方法：错误在于把 Tensor 值当成布尔值用了，该 Tensor 值在代码中是输出图像的长宽，即 outputs.shape 的第二、三个元素，我不把长宽转为 Tensor 型，直接赋值给参数就好了。还是不行，现在是这个问题 ValueError: tf.function only supports singleton tf.Variables created on the first call.我真的解决不了。

重新配置虚拟环境安装 Tensorflow，所有的依赖版本都与作者用的一致，运行训练文件依然有一些报错，解决后终于能正常跑起来了，yes!（虽然有点慢？...）

（去 autoDL 里面充点钱，简单配一下环境就好了，租好点的机子，跑的挺快的）