Neural Networks: Assignment I

刘闵 191240030@smail.nju.edu.cn

Kuang Yaming Honors School, Nanjing University

Basic Settings:

Python 3.8.12 + PyTorch 1.9.1 + CUDA 10.2.89 (packages tqdm, lime required)

GPU: GeForce GTX 1080 Ti

Architecture: A ResNet-18 model with pretrained parameters is used for analyses in this report. The prediction accuracy of this model on train set and dev set are 73.09% and 68.03% separately. Please notice that the model (0.7102 on public leaderboard) I used for prediction on Kaggle is ensembled and trained under complex settings. You can refer to the code or contact me directly for more details.

1. Saliency Maps

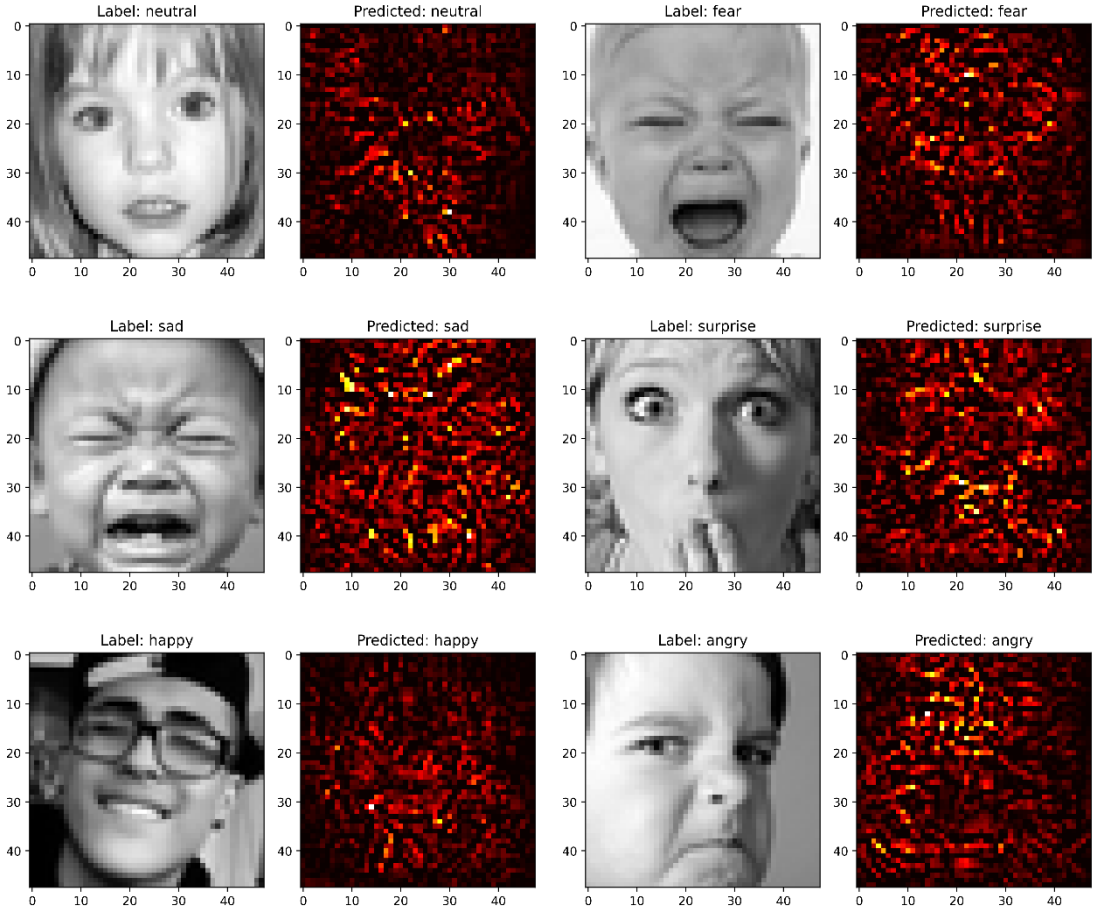

Figure 1 Saliency Maps

通过观察大量 saliency maps 可以发现，模型基本能准确判断人脸在画面中的位置，并且对人脸各个部位的 focus 程度有所不同。模型一般重点关注五官的轮廓（Figure 1 的 1, 2, 4, 5）或是重点区域的皮肤褶皱（Figure 1 的 3, 6）。从第五张图可以看出，在图像的显著特征不多的情况下（皮肤平滑，眼睛鼻子受眼镜和表情的影响形变较大），在整张图像中得到的 activation 较少，但是仍能通过捕捉到嘴巴的轮廓准确判断情绪。可能原因之一在于训练过程中使用了 random crop。

Saliency maps 的代码可在 analysis.py 的 saliency_map 函数中找到。

2. Gradient Ascent

通过 Gradient Ascent 可将卷积层 filter 可视化。Figure 2 为 ResNet18 处理原始输入的卷积层的前六个 filter（先左右后上下的方式编号为 0-5）。这一层的 filters 只能提取到一些初级的特征，可视化后相较于随机生成的图像稍微规整一些。

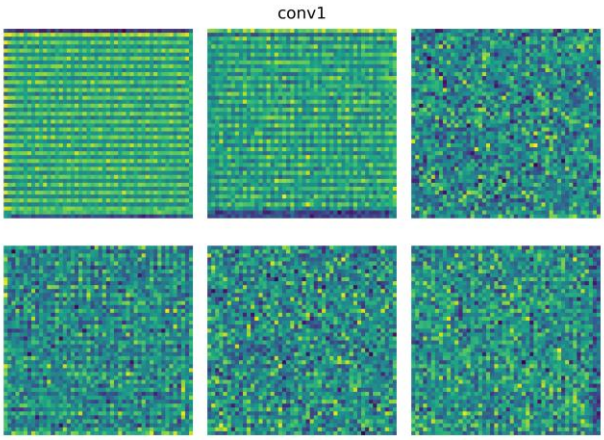

Figure 2 Visualized filters for the first conv layer

Figure 3 为 ResNet18 四个 Block 的第一个卷积层。可以发现，越靠后的卷积层能提取的特征越为复杂。通过可视化后的 filter 不能直接判断出提取了与什么情绪相关的特征，可能原因之一在于人脸的情绪特征模式较多，一个 filter 往往只能用于提取局部特征或某一模式下的特征。另一原因在于，模型理解情绪的方式很可能和人类有本质上的不同，filter 可能已经提取到了相当好的特征，但其可视化的结果不能直接通过肉眼理解。

Table 1 中列出了 Figure 2, 3 中每个 filter 最容易被哪种情绪激活。结合表格和图像仍然很难通过肉眼观察到特别的联系。不过一个有趣的发现是，越往后的卷积层越容易被某种特定的 label 激活，也即该类型下的平均数出值和其他 label 区分度越明显。一种可能的解释是：由于靠前的 filter 学到的是一些基础的特征，例如局部的轮廓和阴影走势等，不同 label 的图像共有的特征较多。往后得到的特征越来越复杂，filter 也随之越来越特化，更加倾向于学习到某种 label 独有的特征以增加不同输入的区分度。

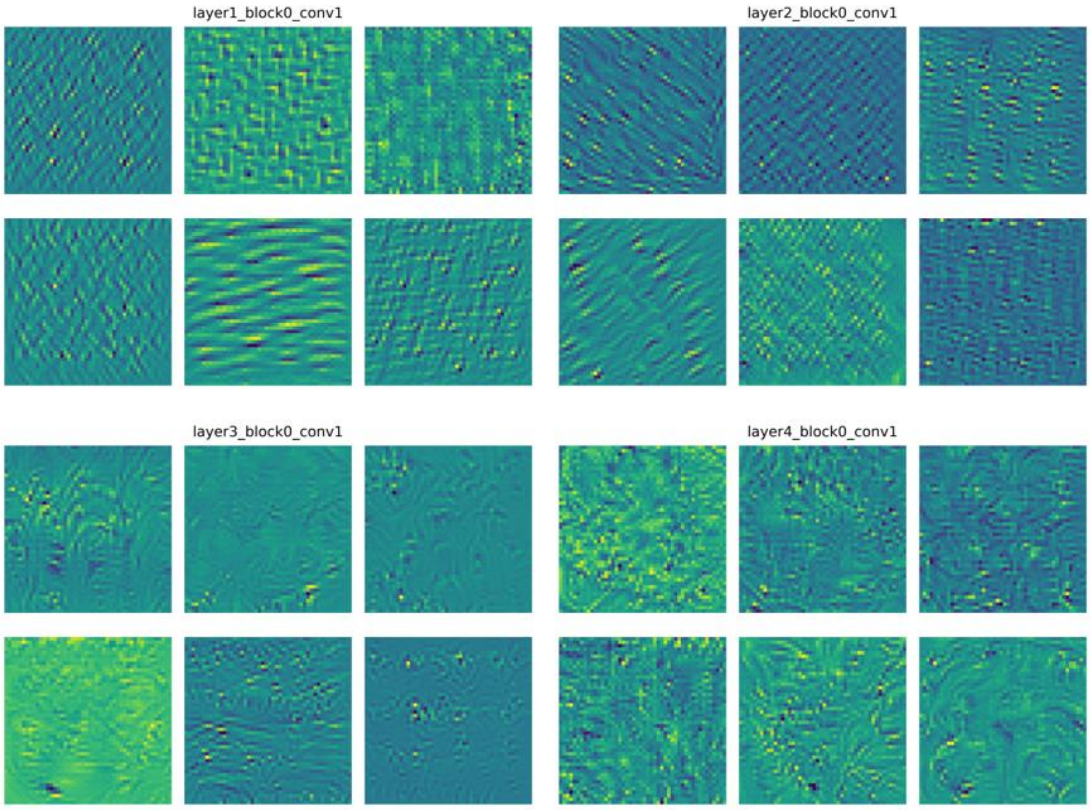

Figure 3 Visualized filters for the four conv layers

Filters	Conv1	Layer1 conv1	Layer2 conv1	Layer3 conv1	Layer4 conv1
Filter0	Neutral	Surprise	Sad	Surprise	Disgust
Filter1	Sad	Surprise	Surprise	Disgust	Neutral
Filter2	Surprise	Neutral	Disgust	Surprise	Disgust
Filter3	Sad	Surprise	Happy	Happy	Angry
Filter4	Surprise	Disgust	Sad	Disgust	Angry
Filter5	Surprise	Sad	Disgust	Angry	Happy

Table 1 The emotion activating each filter.

为了直观看出卷积层输出的内容，我在 Figure 4 中展示了 ResNet18 第一个卷积层 64 个 filter 的输出（后面卷积层的 filter 输出没有可视化的意义）。可以发现部分 filter 提取了脸部轮廓特征，部分提取了阴影特征，还有一些卷积层保留了大多数的输入特征。

本部分代码可在 analysis.py 的 gradient_ascent, filter_out 和 test_filter 函数中找到。

Figure 4 Outputs of filters in conv1

3. LIME

通过 LIME 可以分析模型在特定输入下判定其为某种 label 的原因。本部分将先通过 confusion matrix 评价模型在各个 label 下的表现。随后会通过 LIME 分析模型在某些 label 下表现好的原因以及出错的原因。

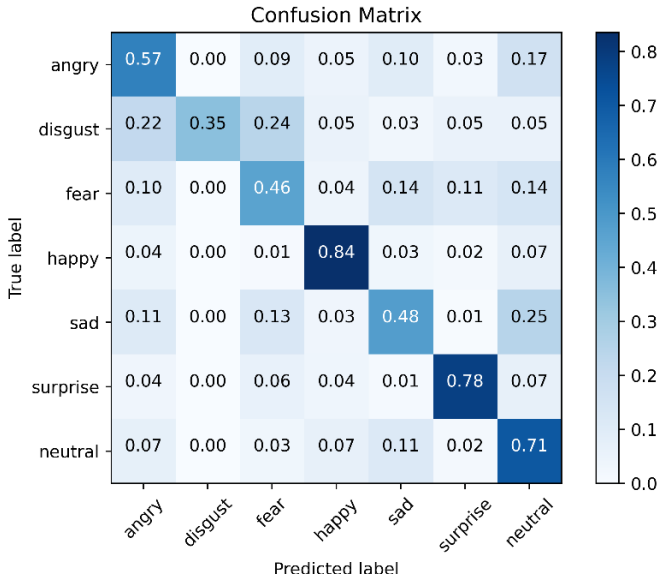

Figure 5 Confusion Matrix

通过 confusion matrix 可以发现模型在 happy, surprise 和 neutral 类型的输入中表现最好，在 disgust, fear, sad 准确率较低。除了 happy 和 disgust 外，FER13 的数据集的分布相对比较平衡，这些 label 准确率的差异更可能来自于其自身属性。

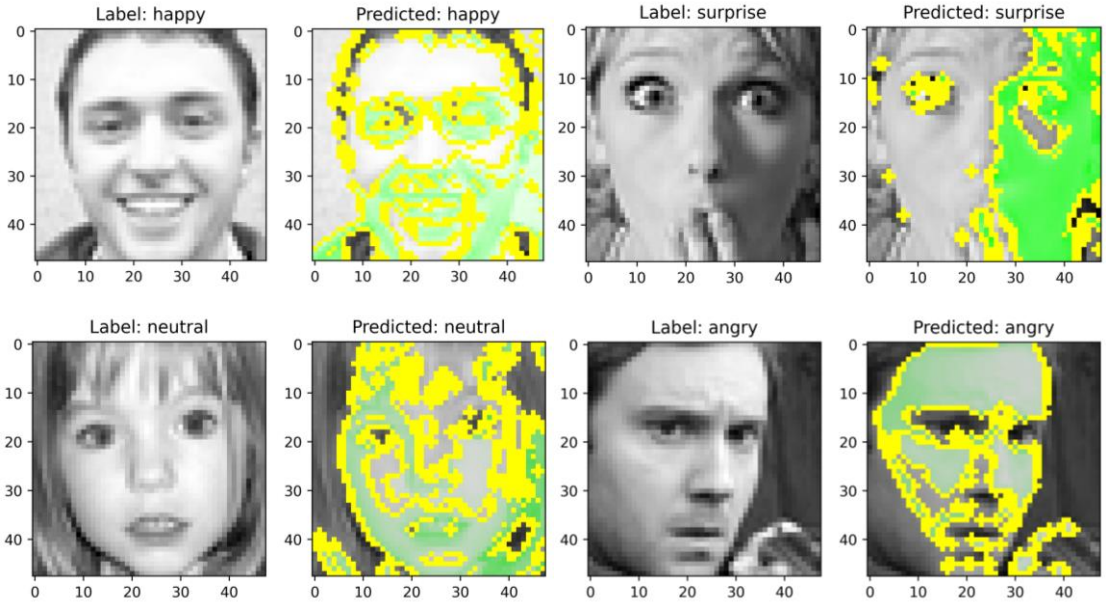

Figure 6 LIME for correctly predicted inputs

Figure 6 列举了 happy, surprise, neutral 和 angry 通过 LIME 分析后的典型结果。在这几个 label 上模型表现较好。在所有 label 中，happy 的嘴部特征最为明显。Figure 6 的第一张图也准确判断出嘴型的弧度以及法令纹一带皮肤的起伏。结合 Figure 6 中的分析结果，可以发现模型能准确捕捉到脸部的轮廓，并且对五官和脸部的阴影尤其敏感。在五官清晰、光线良好且表情具有明显特征的情况下，模型具有良好的表现。

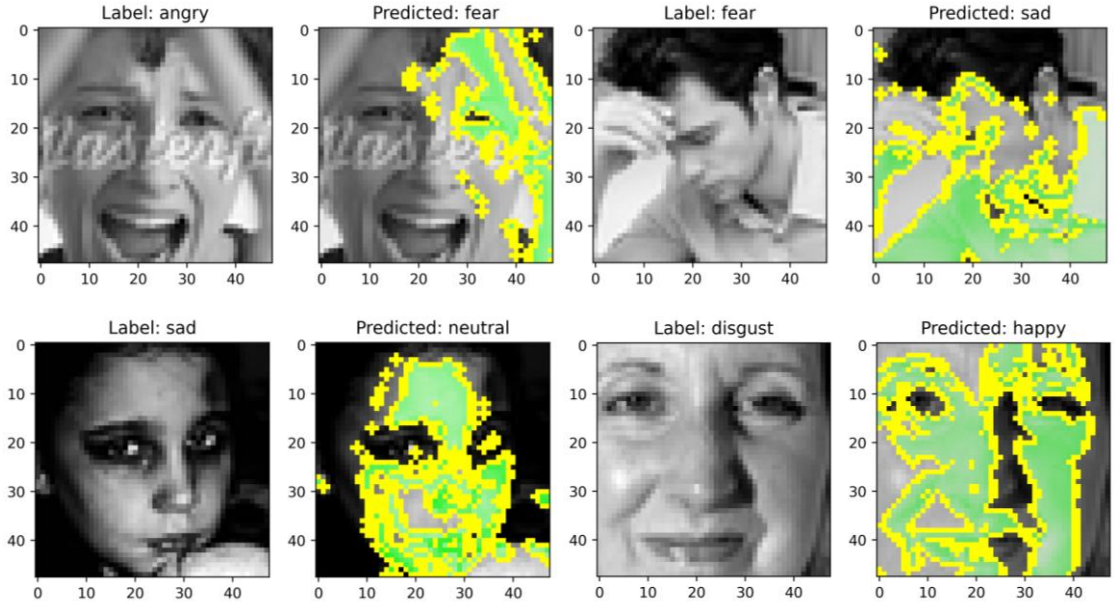

Figure 7 LIME for wrongly predicted inputs

Figure 7 分析了模型出错可能的原因。第 1，2 张图中模型关注的重点是人物的姿势，对脸部特征的关注较少。训练集中少量数据带有人物的动作信息，在某些场景下姿势确实对情绪具有较强的暗示性，例如第 2 张撑头的姿势与悲伤的情绪有较强的关联。不过由于训练集中带有姿势的数据量很少，对这一特征的判断很容易过拟合。在本例中，模型直接通过动作进行了预测。

第 3, 4 张图包含了另一种可能的错误，也即特征不够明显。结合 confusion matrix 可以发现，sad 经常误判为 neutral，fear 经常误判为 sad 和 neutral。观察数据集可以发现，这两类表情很少带有夸张的表情特征。第 3 张图中 sad 的情绪直观上是通过眼神传达的，但模型对眼睛的关注并不多。第 4 张图的误导性很强，其嘴部和眼睛轮廓和 happy 很相近。在很多判错的样例中，人类出错的概率也不小。

本部分代码可在 analysis.py 的 lime 函数中找到。

4. CNN Training Visualization

Confusion Matrix: 详见 Figure 5

Training / Dev Loss:

在 analysis 的模型中，我采用了 early stop 防止过拟合，最终 loss 曲线如 Figure 8 左图。

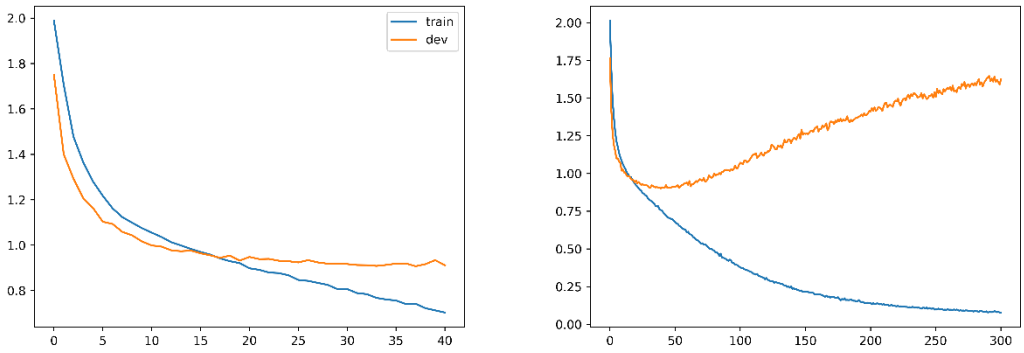

Figure 8 analysis model and overfitting model

早期的实验过程中，我的大多数模型虚拟蓝结果如由上图所示，但尽管 dev loss 在上升，验证集上的 acc 却也在小幅上升。检查之后可以发现，模型在验证集上的这一表现是因为它预测错时更加笃定了。如第三部分的分析所示，模型有时候会捕捉到一些次要的特征（例如手的姿势），在训练集上反复迭代后，模型对这类特征的过拟合越发严重，以至于验证集合上的这些样例给错误的预测权重越来越大。但实际上在本来就会犯错的样例上出错并不会降低 acc。另一个发现是，随着随机初始化的结果和 training setting 的不同，模型过拟合时所找到的特征依据也经常不同，因此多个过拟合的模型集成时，模型之间可能会通过投票缓和受到的过拟合的影响。我在提交的代码中仅使用了最朴素的集成方法，但这已经能够将准确率提高 2-3 个 point。

4. 代码运行说明

我提交了本报告中模型的参数，可以通过运行脚本直接训练和测试。我用于 Kaggle 的集成模型的参数文件很大，可以取消我在 train.sh 和 test.sh 中的注释以重新训练。所有模型的随机种子都已在脚本中设置好，结果均可复现。训练环境可通过修改脚本中的参数来设定，命令行参数详见 options.py（目前多 GPU 尚不支持）。