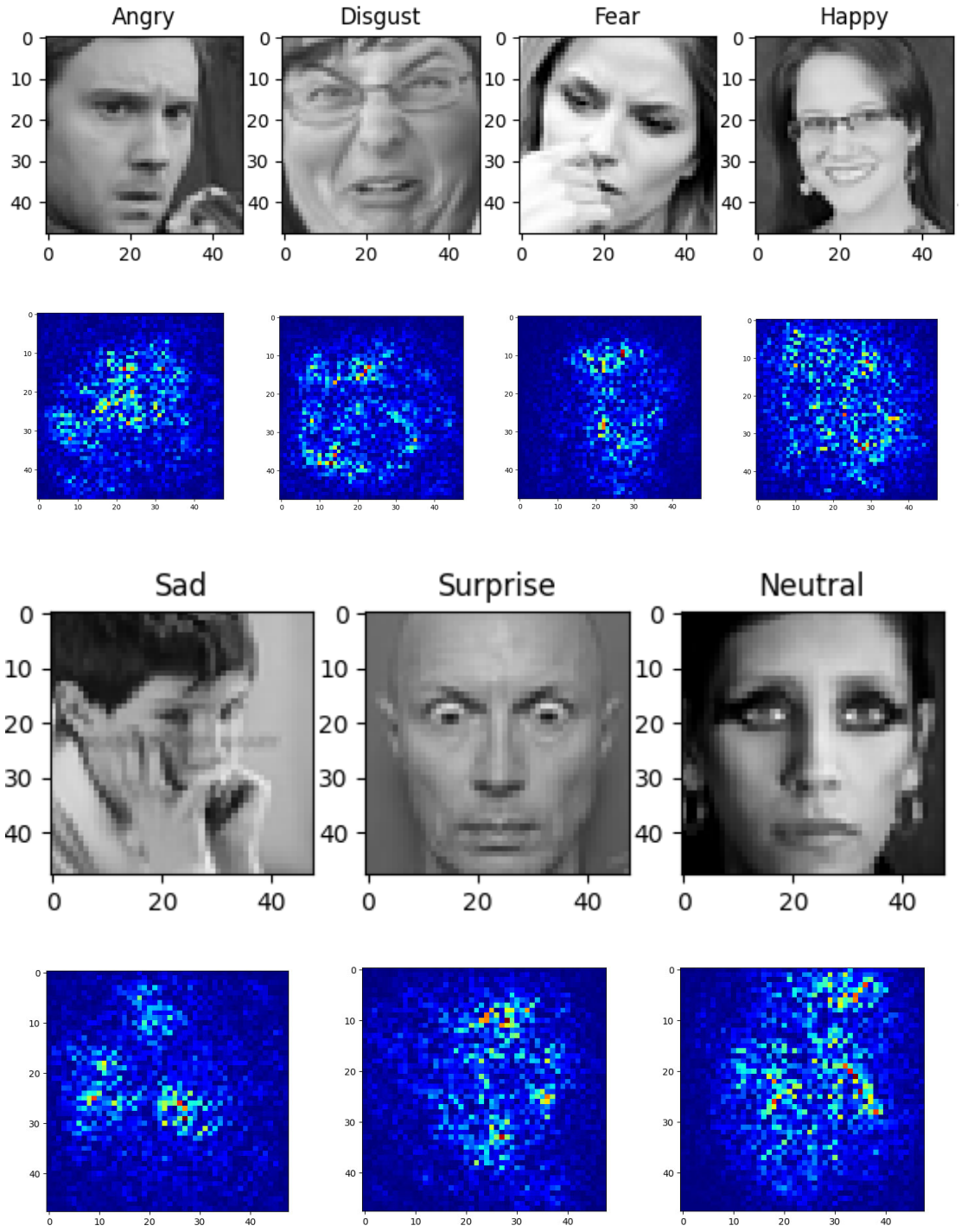
①请尝试绘制 saliency maps，观察模型在做 classification 时，是 focus 在图片的哪些部份？

下图是尝试绘制的对应不同表情的 saliency maps：



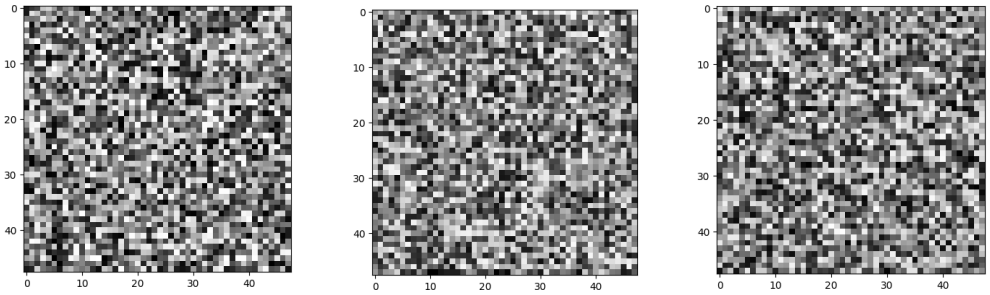
将损失传播回像素值，突出对损失值影响最大的像素，突出了 CNN 可以从输入中捕获的视觉特征。观察显著性图，我认为模型分类时，关注图片中的面部特征，尤其是眼睛、鼻子、嘴巴，很多图片中（比如这次例子中的 angry 图、happy 图、surprise 图、neutral 图）脸颊有时也会得到关注。大部分时候，无关信息被去除了，比如头发，angry 图、fear 图中的手，但是，sad 图中左侧的手受到了一定的关注，无效信息受到关注可能会对模型的结果造成不良影响。

②利用上课所提到的 gradient ascent 方法，观察特定层的 filter 最容易被哪种图片 activate 与观察 filter 的 output。

要使用梯度上升方法找出特定层的滤波器最容易被哪种图片激活，可以创建一个损失函数，将特定滤波器的激活最大化，然后使用梯度上升来修改输入图像以最大化这个损失函数。

对于每个可视化的层，可以生成图像，最大化所选滤波器的激活，此图是所选滤波器“想要看到”的图像。通过观察这个图像，可以了解滤波器正在寻找什么样的模式或特征

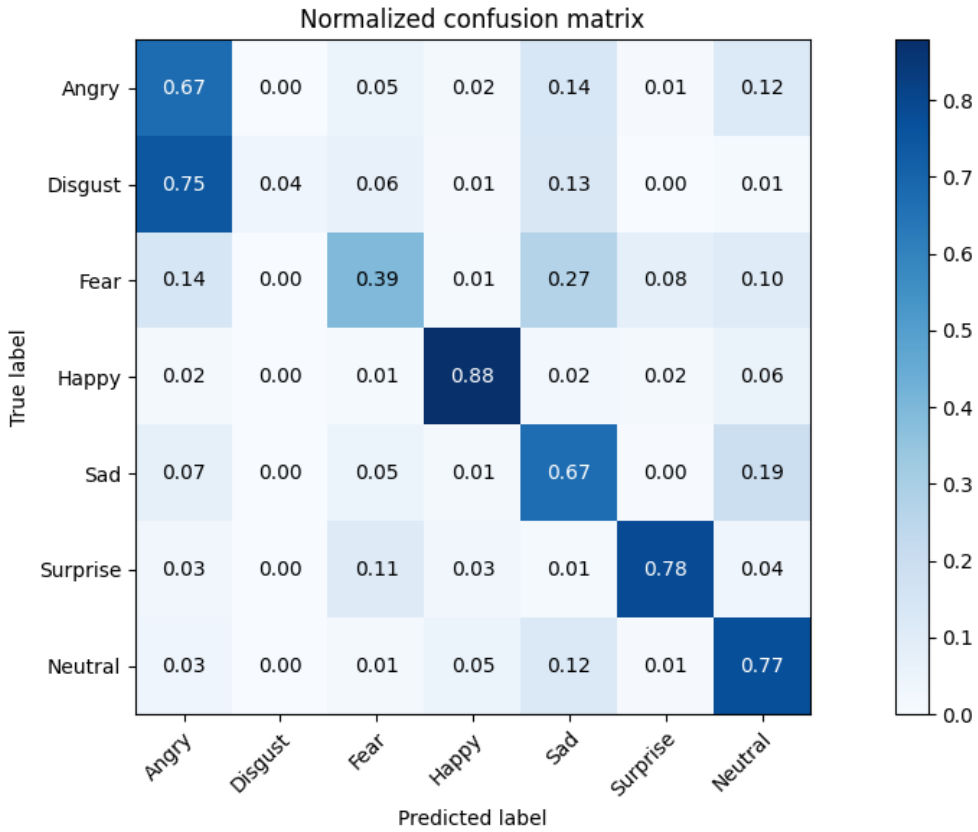
如图，这三张图分别是'conv2d_24', 'batch_normalization_24', 'activation_24',对应的灰度图像。（cmap='gray'）



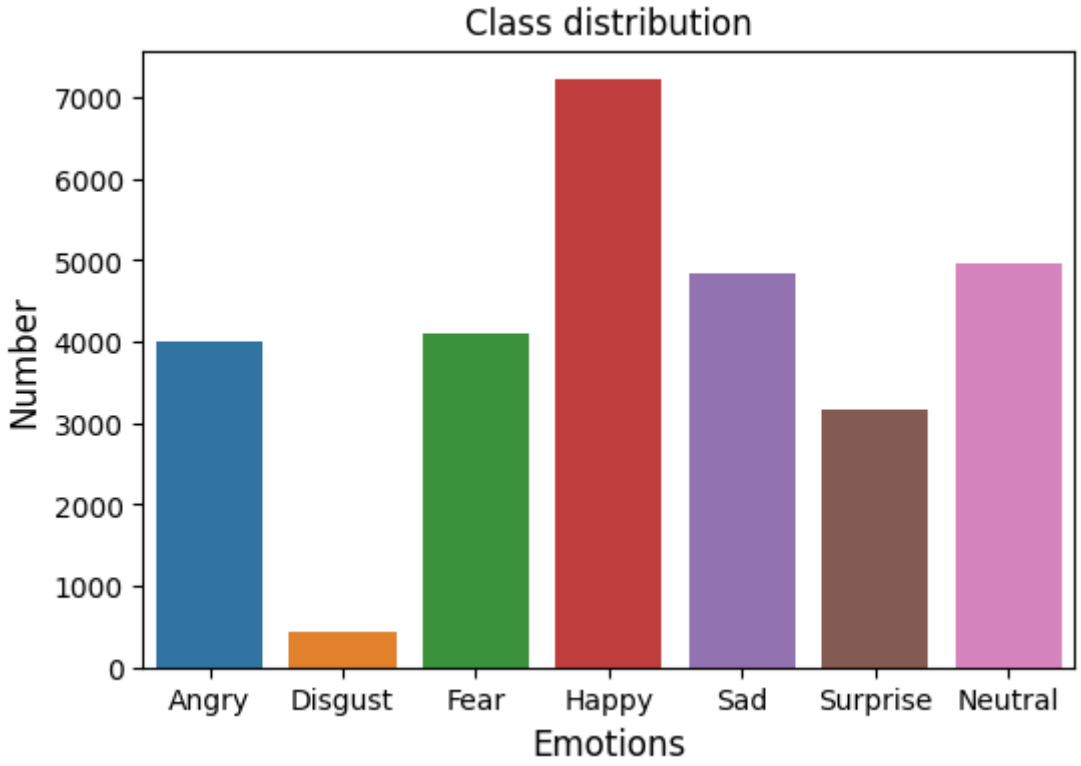
但是这长得像二维码一样实在观察不出啥。

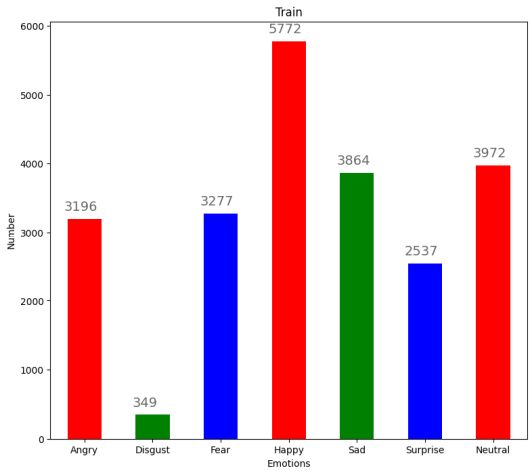
③尝试分析你的模型对于各种表情的判断方式，并解释为何你的模型在某些 label 表现得特别好。 （LIME？，本题可选）

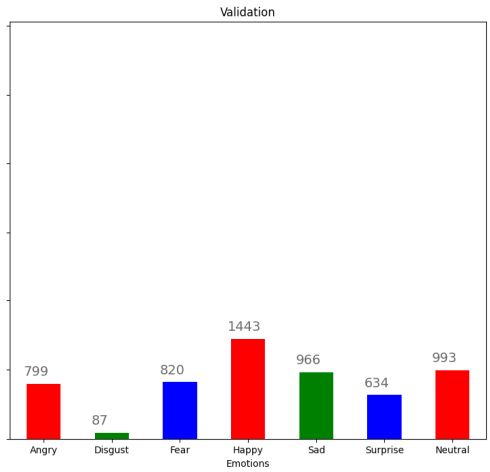
如图，我绘制了以 train.csv 为数据源的预测与数据中表情的比较图：



可以看到，在 happy 的预测表现特别好，surprise 和 neutral 也很不错，angry 和 sad 尚可，但是 fear 很一般，disgust 非常差，并且把大量的 disgust 都识别为了 angry，可能因为 disgust 的样本数本身就很少（如下图所示 train.csv 中的数据量，以及把 train.csv 分割成训练和验证集的数据量），比较容易和其他表情混淆，样本数最多的 happy 表现也最好。从模型的角度解释，也可能 disgust 和 angry 之间数据集有阶层间相似性，在模型捕捉图像关键特征时，未能很好地找到 fear 和 disgust 的关键特征，最终表现很差；而 happy 的面部特征得到了很好的捕获，模型准确率很高。

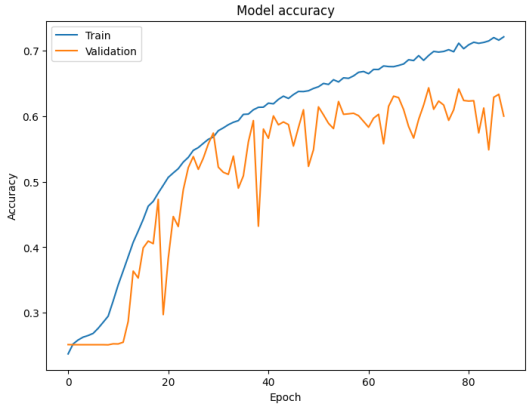


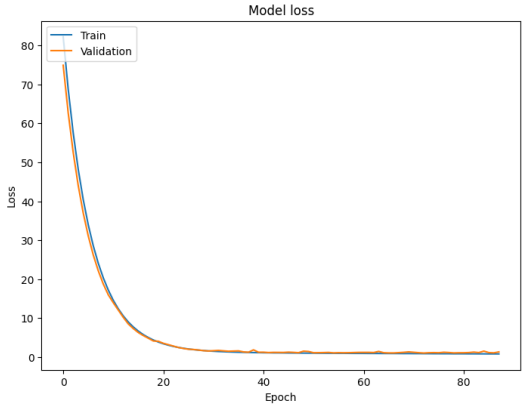




④请同学自行搜寻或参考上课曾提及的内容，实作任一种方式来观察 CNN 模型的训练，并说明你的实作方法及呈现 visualization 的结果。

在程序运行时，已经用代码进行了训练的可视化，如下图所示，根据验证集的准确率曲线 validation，可以看到，神经网络后期可能存在轻微的过拟合现象，验证集准确率下降。





实作方式：该部分具体代码如下：

```
fig, axes = plt.subplots(1, 2, figsize=(18, 6))
# 绘制训练和验证精度曲线
axes[0].plot(history.history['accuracy'])
axes[0].plot(history.history['val_accuracy'])
axes[0].set_title('Model accuracy')
axes[0].set_ylabel('Accuracy')
axes[0].set_xlabel('Epoch')
axes[0].legend(['Train', 'Validation'], loc='upper left')

# 绘制训练和验证损失曲线
axes[1].plot(history.history['loss'])
axes[1].plot(history.history['val_loss'])
axes[1].set_title('Model loss')
axes[1].set_ylabel('Loss')
axes[1].set_xlabel('Epoch')
axes[1].legend(['Train', 'Validation'], loc='upper left')
plt.show()
```

除此之外，观察 CNN 模型训练的一种常见方法是使用 TensorBoard，一个用于可视化 TensorFlow 运行的工具。TensorBoard 可以跟踪和可视化模型在训练过程中的指标，如损失和准确性，也可以可视化模型图，显示张量的直方图，显示特定层的激活和梯度等，具体实现如下：

```
import tensorflow as tf
from tensorflow.keras.callbacks import TensorBoard

model = ...# 定义模型
# 创建一个 TensorBoard 回调
tensorboard_callback = TensorBoard(log_dir='./logs', histogram_freq=1)
# 训练模型，并将 TensorBoard 回调传递给`fit`函数
model.fit(X_train, y_train, epochs=10, callbacks=[tensorboard_callback])
```

log_dir 参数是日志文件将被写入的目录，histogram_freq=1 表示每一轮都计算直方图。然后，可以在命令行中启动 TensorBoard 并指向日志目录：tensorboard --logdir=./logs