**Step 3: Encoding Faces**

**The most reliable way to measure a face**

Ok, so which measurements should we collect from each face to build our known face database? Ear size? Nose length? Eye color? Something else?

It turns out that the measurements that seem obvious to us humans (like eye color) don’t really make sense to a computer looking at individual pixels in an image. Researchers have discovered that the most accurate approach is to let the computer figure out the measurements to collect itself. Deep learning does a better job than humans at figuring out which parts of a face are important to measure.

The solution is to train a Deep Convolutional Neural Network. But instead of training the network to recognize pictures objects like we did last time, we are going to train it to generate 128 measurements for each face.

The training process works by looking at 3 face images at a time:

1. Load a training face image of a known person
2. Load another picture of the same known person
3. Load a picture of a totally different person

Then the algorithm looks at the measurements it is currently generating for each of those three images. It then tweaks the neural network slightly so that it makes sure the measurements it generates for #1 and #2 are slightly closer while making sure the measurements for #2 and #3 are slightly further apart.

After repeating this step millions of times for millions of images of thousands of different people, the neural network learns to reliably generate 128 measurements for each person. Any ten different pictures of the same person should give roughly the same measurements.

Machine learning people call the 128 measurements of each face an **embedding**. The idea of reducing complicated raw data like a picture into a list of computer-generated numbers comes up a lot in machine learning (especially in language translation). The exact approach for faces we are using [was invented in 2015 by researchers at Google](http://www.cv-foundation.org/openaccess/content_cvpr_2015/app/1A_089.pdf) but many similar approaches exist.

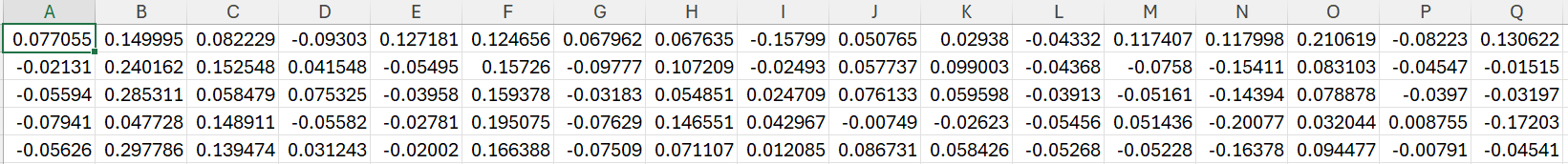
**Encoding our face image**

once the network has been trained, it can generate measurements for any face, even ones it has never seen before! So this step only needs to be done once. Lucky for us, the fine folks at [OpenFace](https://cmusatyalab.github.io/openface/) already did this and they [published several trained networks](https://github.com/cmusatyalab/openface/tree/master/models/openface) which we can directly use. Thanks [Brandon Amos](http://bamos.github.io/) and team!

So all we need to do ourselves is run our face images through their pre-trained network to get the 128 measurements for each face. Here’s the measurements for our test image:

A person taking a selfie

Description automatically generated



So what parts of the face are these 128 numbers measuring exactly? It turns out that we have no idea. It doesn’t really matter to us. All that we care is that the network generates nearly the same numbers when looking at two different pictures of the same person.

你需要在终端或命令行界面中输入这些指令。以下是每一步的详细说明：

**1. 安装Docker**

如果你还没有安装Docker，可以从Docker官网下载并安装Docker Desktop。

**2. 打开终端或命令行界面**

在Windows上，你可以使用PowerShell或命令提示符。在Mac或Linux上，你可以使用终端。

**3. 拉取OpenFace的Docker镜像**

在终端或命令行界面中输入以下命令：

bash

Copy code

docker pull bamos/openface

**4. 运行OpenFace的Docker容器并挂载你的本地目录**

确保你的本地目录 C:\Users\zouxu\OneDrive\Desktop\p poker\sum sprint\6\_3\training-images 已经准备好，并包含你要处理的图像。然后在终端或命令行界面中输入以下命令：

bash

Copy code

docker run -p 9000:9000 -p 8000:8000 -v "C:\Users\zouxu\OneDrive\Desktop\p poker\sum sprint\6\_3\training-images:/root/openface/training-images" -t -i bamos/openface /bin/bash

这个命令将启动Docker容器，并将你的本地目录挂载到Docker容器中的 /root/openface/training-images 目录。

**5. 进入OpenFace根目录**

在Docker容器的命令行中输入以下命令：

bash

Copy code

cd /root/openface

**6. 对齐图像**

在Docker容器的命令行中运行以下命令进行图像对齐：

bash

Copy code

./util/align-dlib.py ./training-images/ align outerEyesAndNose ./aligned-images/ --size 96

**7. 生成特征向量**

在Docker容器的命令行中运行以下命令生成对齐图像的特征向量：

bash

Copy code

./batch-represent/main.lua -outDir ./generated-embeddings/ -data ./aligned-images/

**8. 验证生成的嵌入**

在Docker容器的命令行中输入以下命令检查生成的CSV文件：

bash

Copy code

ls ./generated-embeddings/

这会显示生成的特征向量文件的列表。

**9. 退出Docker容器**

完成所有操作后，你可以输入 exit 来退出Docker容器。

通过这些步骤，你可以使用OpenFace在Docker容器中进行人脸对齐和特征向量生成。如果有任何问题或需要进一步的帮助，请告诉我。

4o

完成上面步骤了如何查看结果或者生成的CSV在哪里

ChatGPT

生成的CSV文件会存放在Docker容器中的 /root/openface/generated-embeddings/ 目录下。为了查看这些结果或将它们复制到本地计算机，可以按照以下步骤进行：

**1. 查看生成的CSV文件**

在Docker容器中，你可以使用以下命令查看生成的CSV文件：

bash

Copy code

ls /root/openface/generated-embeddings/

这会显示生成的特征向量文件的列表。例如，你可能会看到类似 reps.csv 的文件名。

**2. 将生成的CSV文件复制到本地计算机**

你可以使用 docker cp 命令将生成的文件从Docker容器复制到本地计算机。首先，你需要在另一个终端或命令行界面中找到运行的容器ID或名称。使用以下命令列出所有正在运行的Docker容器：

bash

Copy code

docker ps

输出示例：

bash

Copy code

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

d1b1f6e7355d bamos/openface "/bin/bash" 2 minutes ago Up 2 minutes 0.0.0.0:8000->8000/tcp, 0.0.0.0:9000->9000/tcp inspiring\_newton

记下容器ID或名称（例如，d1b1f6e7355d 或 inspiring\_newton）。

然后使用 docker cp 命令将文件复制到本地：

bash

Copy code

docker cp <container\_id>:/root/openface/generated-embeddings/reps.csv /path/to/local/directory/reps.csv

替换 <container\_id> 为你的容器ID或名称，/path/to/local/directory/ 为你希望保存CSV文件的本地目录。例如：

bash

Copy code

docker cp d1b1f6e7355d:/root/openface/generated-embeddings/reps.csv C:\Users\zouxu\Desktop\reps.csv