# Короткий отчет по заданию.

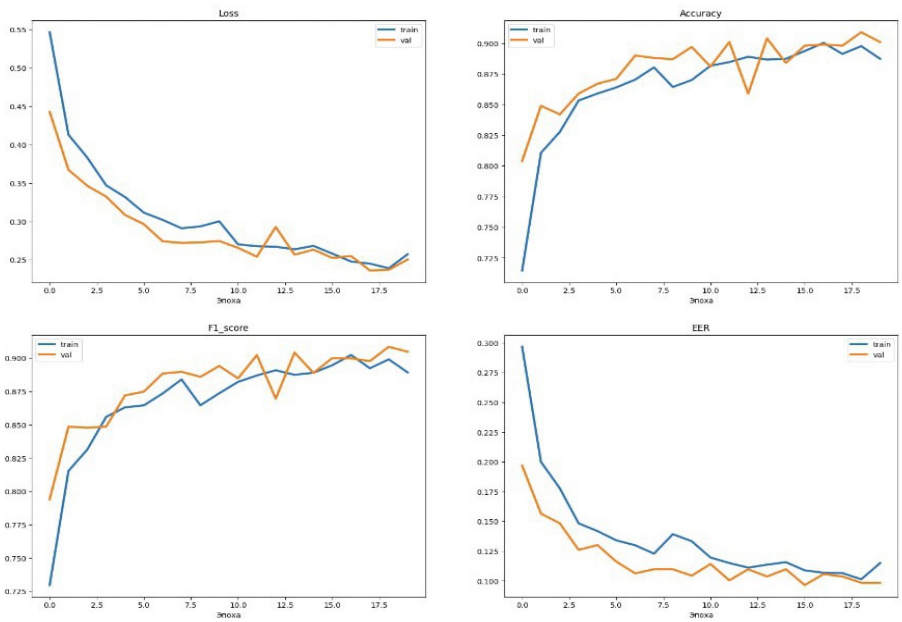
1. **Постановка задачи**: необходимо создать классификатор открытых/закрытых глаз, используя заданную обучающуюся выборку.

2. **Обработка данных**: в задаче было всего 4000 изображений. Это очень маленькое количество, поэтому для увеличения обуающей выборки была добавлена аугментация: RandomHorizontalFlip (горизонтальная симметрия) и RandomRotation (поворот изображения на от 0 до 10 градусов). Также были вычислены параметры нормализации на обучающей выборке. Получились следующие показатели: mean = 0.6117,  std = 0.1954.

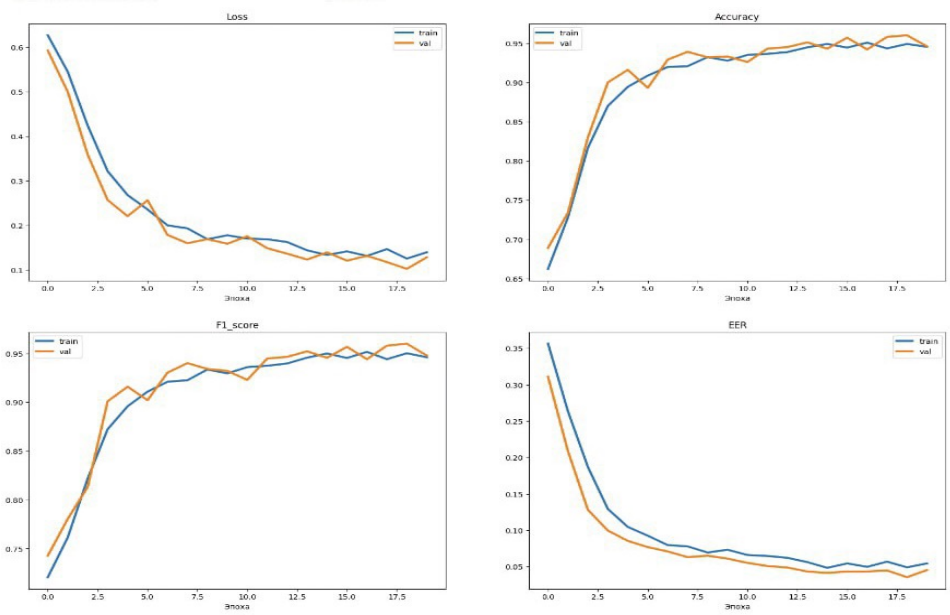
3. **Построение модели и их результаты**: так как выборка у нас небольшая, то использовать реализованные модели из pytorch будет плохо, так как будет сильное переобучение (слишком много параметров, сложные модели). Поэтому было принято решение самому реализовать модель. В данной задаче я реализовал 4 модели. В процессе обучения вычислялись следующие метрики: accuracy, f1_score и EER.

   Рассмотрим особенности и архитектуры построенных моделей:

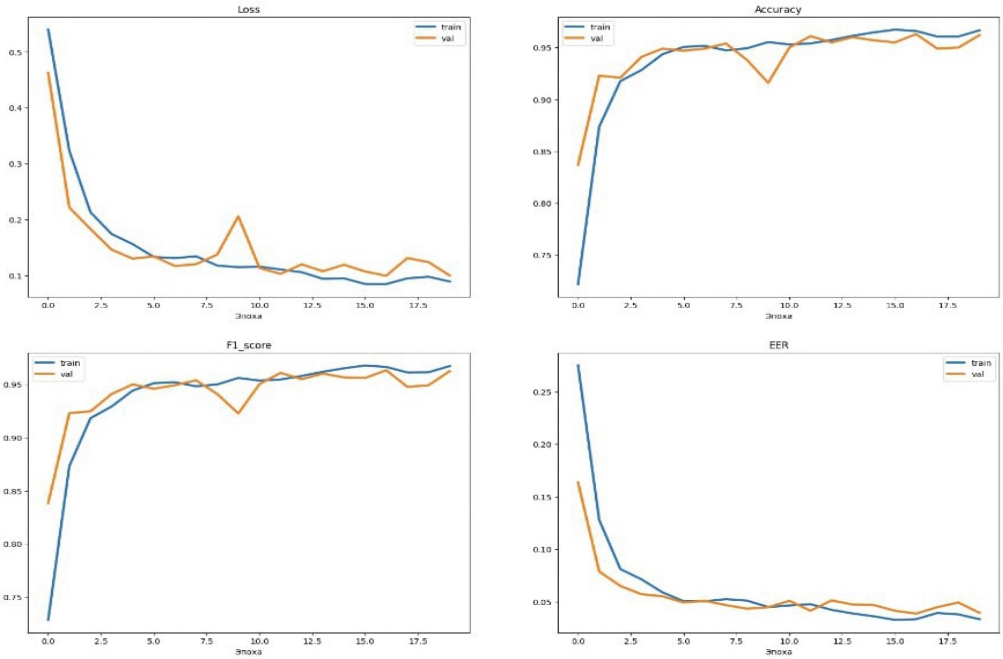
   a. Первая модель состоит только из FC слоев, а именно 3 FC слоя с Dropout. Понятное дело, эта модель не будет самой лучшей, это было сделано, чтобы просто посмотреть на адекватность данных. К большому удивлению эта модель очень хорошо себя показала и выдала неплохие результаты: accuracy = 90.10%, f1_score = 90.49% и EER = 0.10%



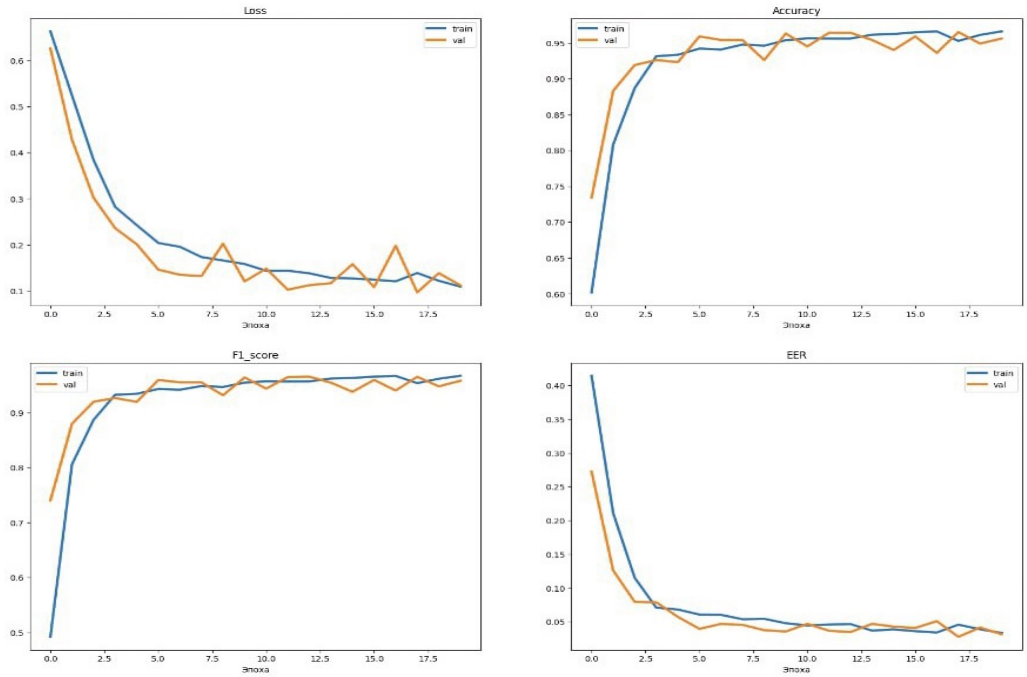

   b. Вторая модель состоит из двух CNN, MaxPool и 3 FC слоев. Эта модель показывает себя намного стабильнее, что в принципе логично, так как CNN инвариантна относительно смещений и поворотов, в отличии от обычных FC слоев. Вот какой получился результат на валидации: accuracy = 94.60%, f1_score = 94.79%, EER = 0.05%




c. Третья модель точно такая же, как вторая, но добавлена BatchNorm. Эта модель намного быстрее обучалась, чем предыдущие, и показала очень хорошие результаты: accuracy = 96.20%, f1_score = 96.25%, EER = 0.04%.

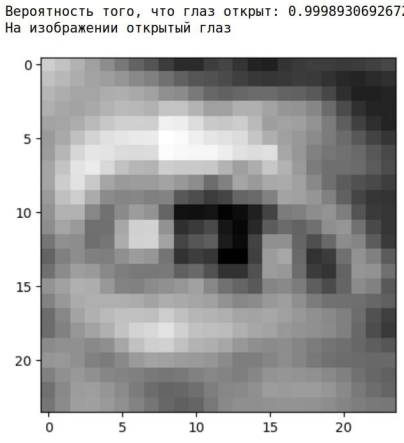


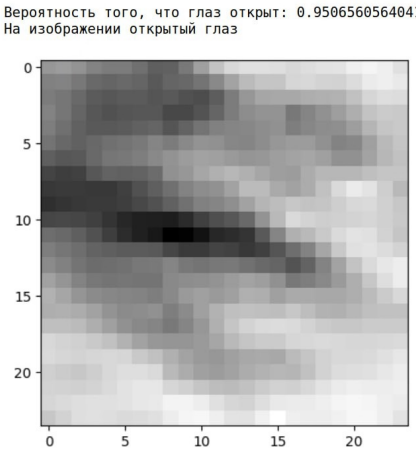

d. В четвертой модели был добавлен еще один CNN. Результаты стали практически такими же: accuracy = 95.60%, f1_score = 95.77% и EER = 0.03%.

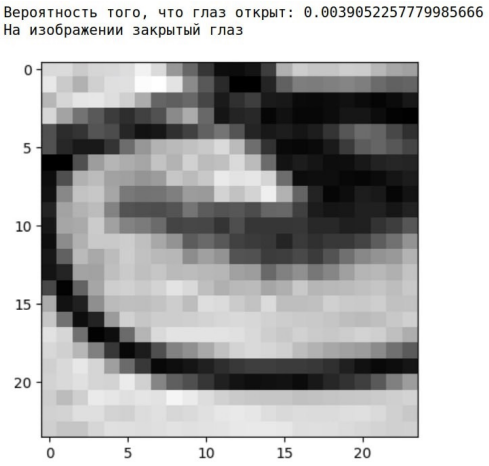


По тенденции обучения видно, что усложнять еще модель нету смысла, так как модель не сможет показать выше скор, чем мы получили. Ну это логично, так как данных мало и в данной задаче лучше использовать маленькие модели. Сам код обучения данных и их графики есть в jupyter файле под названием “eye_proj_inter.ipynb”. В итоге я взял самую последнюю модель, так как он показал  самый низкий EER на валидации.

4. **<u>Примеры работы модели</u>**: я взял несколько примеров изображений глаз, в которых, как мне показалось, сам человек не понял бы, открыт глаз у человека или закрыт:




5. **<u>Вывод</u>**: в этом задании, добавив аугментацию на нашей выборке (так как данных не так много), мы обучили 4 модели, которые показали неплохие результаты. По графикам видно, что модели стабильно обучаются. В итоге самый лучший результат EER стал 0.03% последней модели.