Тестовое задание

Задача: Натренировать классификатор открытых/закрытых глаз.

Исходные данные: 4 тыс. неразмеченных изображений глаз 24x24.

Выходные: Модель с Accuracy >95% на закрытой выборке. (В конце я понял, что использую другую целевую метрику)

**1. Анализ данных и разметка данных**

Картинки небольшие. Первая идея, которая возникла обучить слабый не нейросетевой классификатор с целью посмотреть, как модели классического обучения сработают на таких данные.

Поэтому я решил обучить RandomForest, за одно с этим он мне разметит большинство изображений и останется вручную проверить и доразметить неправильные

Разметил 100 картинок вручную на и обнаружил, что достаточный процент из них не лучшего качества.

Примеры из первых 100 картинок:



Такие изображения я не стал добавлять в обучающую выборку в надежде, что закрытая выборка почище

**2. Обучение случайного леса**

Попробовал обучить Random Forest на этих картинках

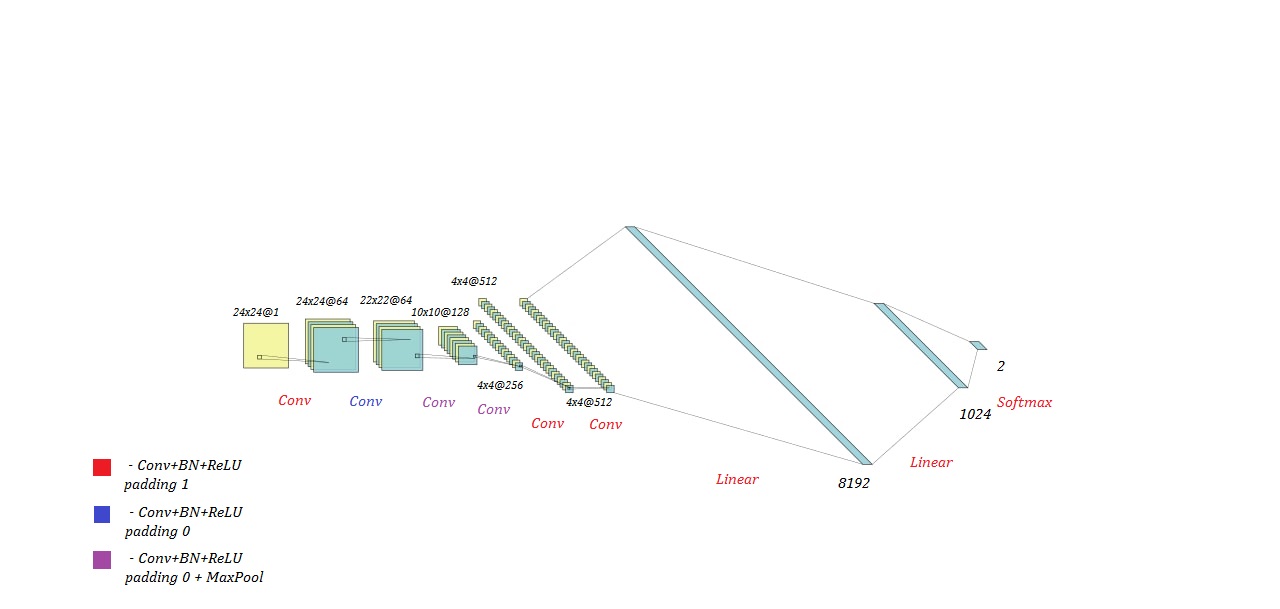
Ход решения представлен в Baseline.ipynb

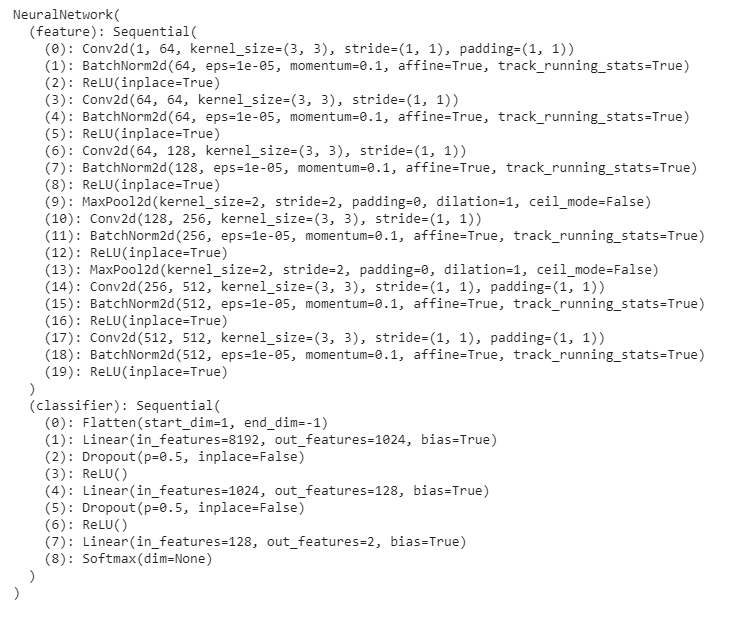
Модель по 100 картинкам случайного леса выдавала метрики:

AUC - 0.875, Accuracy - 0.80

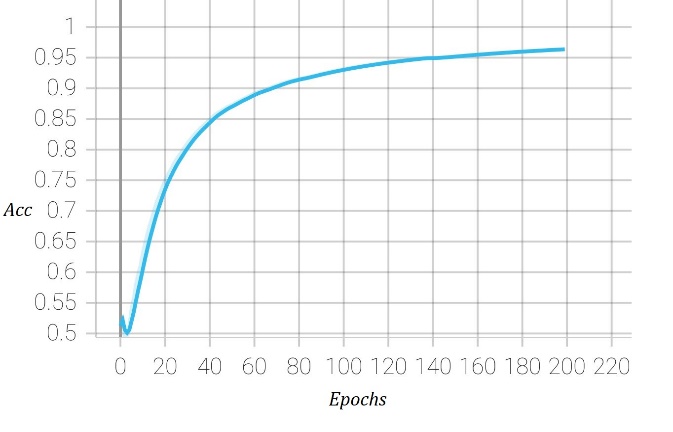
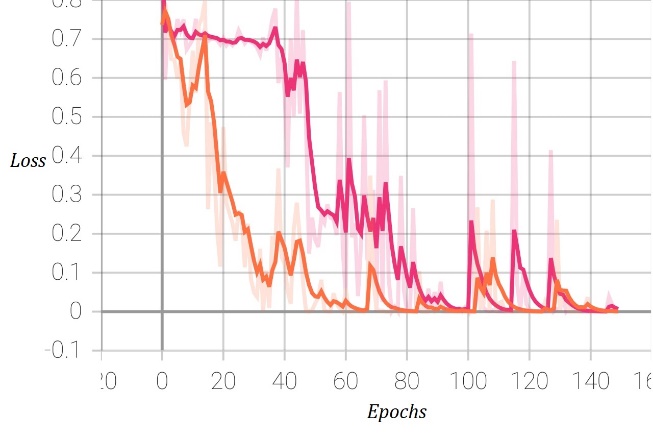
Чего явно не хватает, даже чтобы автоматически предразметить, поэтому я решил доразметить еще 400, однако улучшения не произошло и подбор гиперпараметров тоже не дал результата, поэтому я решил оставить не нейронные методы.

**3. Обучение сверточной нейросети**

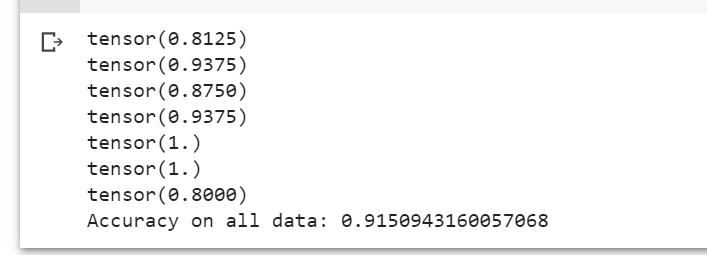
Особенность задачи – размер картинок 24x24. Это не позволяет использовать предобученные модели Pytorch’а. Поэтому необходимо подобрать подходящую для задачи архитектуру.

Задавал архитектуру по подобию моделей из статьи K. Simonyan, A.Zisserman “ Very Deep Convolutional Networks for Large-Scale Image Recognition”(VGG) 

В качестве функции потерь использовал бинарную перекрестную энтропию.  
Стратегия оптимизации: Адам.



Итоговый Accuracy такой модели получился 0.95 на обучающей и 0.91 на тестовой



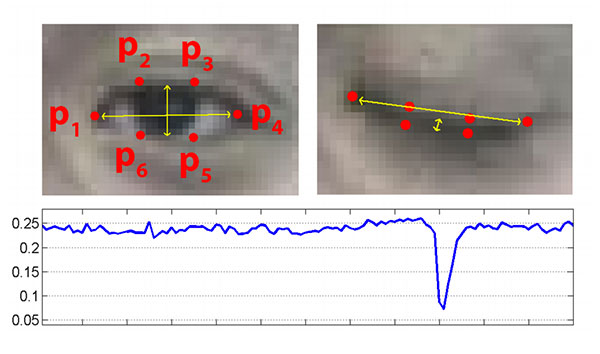
Увеличение на 0.11, но все ещё недостаточная точность.

**4. CNN Regressor + EAR**

В ходе изучения статей решил использовать подход из статьи T. Soukupova and J.Cech

“Real-Time Eye Blink Detection using Facial Landmarks”

Авторы предлагают использовать метрику Eye aspect ratio:



Таким образом, переводим задачу в задачу регрессии 6 точек. После чего мы однозначно, отсекая по порогу, можем сказать открыт глаз или закрыт.

Для регрессии используем ту же архитектуру, с заменой выходного слоя на 12 выходных нейронов.

Подготовку данных делал вручную. Разметил 300 изображений закрытых и открытых глаз, в сервисе от Intel CVAT(Computer Vision Annotation Tool).

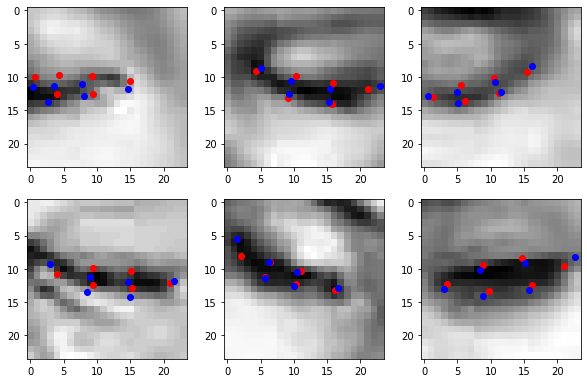
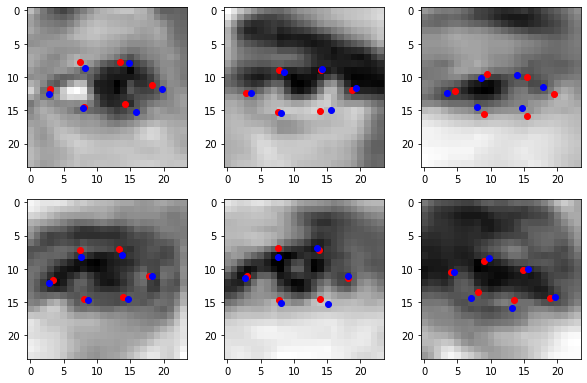
В начале использовал функцию ошибки Mean Squared Error.

Пришлось переписать загрузчик данных и разобраться с загрузкой разметки из формата xml

Первая попытка обучения выдала плохой результат модель явно недоучилась и ставила всем картинкам почти один и тот же паттерн по середине изображения.

Поэтому я решил сильно увеличить время обучения и заменить функционал на MAE, чтобы модель не боялась ошибиться на выбросах.

Это поспособствовало улучшению, регрессированные точки стали выглядеть явно лучше:

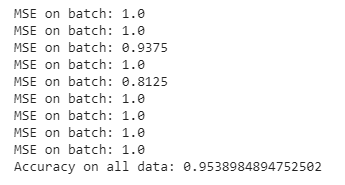


*(Синие точки – точки разметки, красные – регрессированные точки.)*

Соответственно модель хорошо работает и на открытых и на закрытых глазах

Подбор порога дабы не настроиться на тестовую выборку делался на валидационной.  
И был выбран 0.24.

Проверка на тестовой выборке дала значение Accuracy около 95,5%



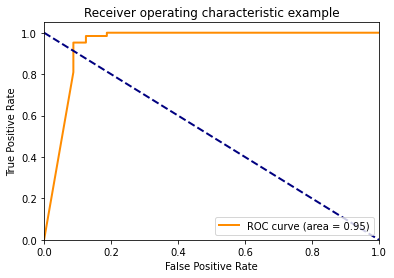
**5. Исправления**

Перед тем как отдать работу перечитал задание и понял, что меня выходные данные должны быть не метками(int), а конфиденсами(float). И целевая метрика не Err =1 – Acc,a Eer.

Но в таком случае идея решения остается такой же, однако после регрессирования точек, вместо кусочно-линейной функции, используем её сглаженную версию – сигмоиду.

Таким образом, примеры с большим AER буду в правой ветке сигмоиды прижаты к 1, а с меньшим AER в левой, близки к 0

Остается только задать подобрать множители в сигмоиде так, чтобы она правильно распределяла значиения.

 примем равным середине между средними распределений закрытых и открытых глаз – 0.24, а так, чтобы функция была не слишком пологой и нормально разделяла примеры – 300

На тестовой выборке алгоритм имеет AUC ≈ 0.94-0.95

EER - - точка, в которой линии FNR и FPR пересекаются. Однако т.к значения массивов FNR и FPR дискретны, то мы принимаем EER равным одному из значений в точке где разница минимальна

Итоговый EER ≈ FNR(A) ≈ FPR(A), где A – точка где FNR и TPR пересекаются.(Она же может быть получена пересечением кривой AUC диагональной линией

Итог:

1. Была получена модель с точностью > 95% и

2. Реализованы ещё 2 алгоритма, не давшие необходимой метрики

3. Разобрался как размечать и работать с файлами формата xml.