

**Направления работ**

Изначально были определены следующие **направления работ**:

1) Адаптация модели, предобученной на распознавание лиц, без дообучения

2) Дообучение модели на предложенном датасете (подбор лосса, бэкбоуна и гиперпараметров)

3) Работа с датасетом (предобработка, аугментации и т.д.)

4) Эксперименты по нахождению фэйкового фото без работы модели

**Описание экспериментов каждого из направлений**

**1. Адаптация модели, предобученной на распознавание лиц, без дообучения**

**Целью** данных экспериментов являлось проверить, возможно ли **применение готовой модели «из коробки»** с небольшими модификациями (без дополнительного обучения) для поставленных задач.

Было протестировано несколько предобученных на распознавание лиц моделей.

В экспериментах применялся фреймворк DeepFace с 8 бэкбоунами *(блокнот DeepFace.ipybn в папке other-experiments)*.

Сначала сравнили скорость инференса и косинусную близость в пределах одного id ‘000000’ (‘0.jpg’ с остальными изображениями этого же id).

Полученные результаты приведены в таблицах ниже.

**Скорость инференса** всей папки (сначала был извлечен эмбеддинг нулевого изображения, а затем для каждого последующего фото замерялось полное время, включая загрузку второго изображения, извлечение эмбеддинга моделью и нахождение косинусного расстояния между данным фото и ‘0.jpg’):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | img | VGG | Facenet | Facenet512 | OpenFace | DeepID | ArcFace | SFace | Ghost |
| 0 | 1.jpg | 0.283330 | 0.233232 | 0.249847 | 0.123224 | 0.038991 | 0.196290 | 0.049927 | 0.200233 |
| 1 | 2.jpg | 0.321746 | 0.249881 | 0.255564 | 0.108740 | 0.042475 | 0.197044 | 0.049970 | 0.216774 |
| 2 | 3.jpg | 0.302622 | 0.250910 | 0.236072 | 0.099964 | 0.029066 | 0.203160 | 0.049855 | 0.215912 |
| 3 | 4.jpg | 0.325660 | 0.247659 | 0.241143 | 0.116575 | 0.042246 | 0.196977 | 0.040926 | 0.234051 |
| 4 | 5.jpg | 0.288249 | 0.250927 | 0.249858 | 0.119626 | 0.043900 | 0.196247 | 0.054073 | 0.248986 |
| 5 | 6.jpg | 0.344581 | 0.233027 | 0.241378 | 0.113519 | 0.040056 | 0.199785 | 0.045856 | 0.216546 |
| 6 | 7.jpg | 0.283063 | 0.249872 | 0.244299 | 0.116653 | 0.039062 | 0.200263 | 0.042444 | 0.199950 |
| 7 | 8.jpg | 0.300061 | 0.272656 | 0.265758 | 0.133218 | 0.057970 | 0.222380 | 0.066701 | 0.216736 |

Наиболее быстрыми оказались **DeepID** и **SFace**,(примерно одинаковое время), чуть медленнее – **OpenFace** (примерно в 3 раза медленнее на данном замере). Остальные модели работали значительно медленнее.

**Результаты работы бэкбоунов**:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | img | Fake | VGG | Facenet | Facenet512 | OpenFace | DeepID | ArcFace | SFace | Ghost |
| 0 | 1.jpg | 0 | 0.619067 | 0.795937 | 0.810955 | 0.621508 | 0.977524 | 0.736856 | 0.500815 | 0.330522 |
| 1 | 2.jpg | 1 | 0.608000 | 0.666713 | 0.695080 | 0.625966 | 0.919044 | 0.583269 | 0.577125 | 0.552727 |
| 2 | 3.jpg | 1 | 0.430599 | 0.698489 | 0.679240 | 0.596473 | 0.919890 | 0.531124 | 0.449707 | 0.565779 |
| 3 | 4.jpg | 0 | 0.557660 | 0.807505 | 0.809689 | 0.786489 | 0.973824 | 0.803240 | 0.710185 | 0.550898 |
| 4 | 5.jpg | 0 | 0.622190 | 0.842795 | 0.870042 | 0.631177 | 0.980442 | 0.731705 | 0.633875 | 0.616968 |
| 5 | 6.jpg | 1 | 0.440144 | 0.704719 | 0.825052 | 0.352457 | 0.790755 | 0.651886 | 0.497414 | 0.492213 |
| 6 | 7.jpg | 1 | 0.518890 | 0.742352 | 0.809848 | 0.674120 | 0.889377 | 0.633849 | 0.571472 | 0.486151 |
| 7 | 8.jpg | 1 | 0.539757 | 0.762128 | 0.825844 | 0.678561 | 0.975122 | 0.673043 | 0.498756 | 0.527255 |

Какие-то фэйковые изображения модели сразу определяли, как непохожие на ‘0.jpg’, какие-то нет. Поэтому следующим этапом стала **проверка точности** работы 3-х самых быстрых моделей на части трейновой выборки.

**Описание эксперимента:** были выбраны первые 100 папок из трейновой выборки и для каждого изображения был получен эмбеддинг с помощью каждой из трех моделей. Данные эмбеддинги были записаны в таблицу (всего 982 эмбеддинга). Затем для каждого изображения в таблице была рассчитана косинусная близость с каждым изображением (964324 cos\_sim). На полученных данных была посчитана следующая статистика: среднее значение косинусной близости для реальных фотографий одного и того же id (доппроверку на «косяки» датасета, когда в одной папке лежат фото разных людей, в рамках данного эксперимента не проводили), среднее значение косинусной близости реальных фотографий одного id с реальными фотографиями других id, среднее значение косинусной близости реальных фотографий одного id с его же фэйковыми фотографиями. Результаты приведены в таблице ниже.

|  |  |  |  |
| --- | --- | --- | --- |
|  | deepid\_cos\_sim | sface\_cos\_sim | opface\_cos\_sim |
| Same person real photos | 0.9343414487119127 | 0.48981736947983173 | 0.6664091677375096 |
| Different persons real photos | 0.9021543670804951 | 0.09648977788178763 | 0.40434615172362237 |
| Same person real vs fake photos | 0.9082266999815607 | 0.36681315908704193 | 0.5364059379049465 |

DeepID показал низкую различающую способность между различными типами фотографий, показав при этом очень высокую степень схожести между ними («все мы люди, все мы человеки»).

Sface сомневался «то-ли девушка, а то-ли виденье», оценив все типы фотографий как разные.

OpenFace был более уверен в ответах, оценив сходство между реальными фотографиями разных людей ниже всего (в среднем, около 0,4), затем реальные против сгенерированных (около 0,54), и выше всего – реальные фотографии одного и того же человека (сильно веря в то, что в папках так и лежало 😊).

Также в рамках проверки точности были сделаны выгрузки сабмита на паблик-тестовом датасете для моделей DeepID и OpenFace. **Полное время** расчета сабмита (включая всё) составило 19423.72913956642 секунд для DeepID и 42728.32036161423 секунд для OpenFace, и эта скорость ~~огорчила нас до глубины души~~, по нашему мнению, не очень подходит для нашего проекта. **Результат на публичном лидерборде** у обеих моделей был примерно одинаково плохой (~0,37 – мы надеемся, что этим цифрам можно верить, с учетом всех неисправностей метрики на лб, но если потерялись три нолика сразу после запятой и перед тройкой, - это будет очень грустно ☹), и ~~мы опять долго плакали~~ т.к. данный результат был на уровне бэйлайна, а вот скорость где-то потерялась, было решено не пользоваться этим фреймворком для дообучения.

Так же еще немного посмотрели на **визуализацию эмбеддингов** DeepID (раздел Check embeddings в блокноте), и в принципе на видно, что два верхних фото (настоящие фото) имеют между собой больше общего по сравнению с двумя нижними фото (фэйковые): меньшее количество синих полосочек слева у фейковых, и большее количество белых справа у реальных. Но повторить данный анализ с другими моделями (в том числе дообученными на предложенном датасете) так же не хватило времени.

Аналогичным способом была проверена **другая модель**: torch.hub.load('facebookresearch/dinov2', 'dinov2\_vits14') *(блокнот Dinov2.ipybn в папке other-experiments).*

Средние **значения косинусной близости** на первых 100 папках трейновой выборки составили:

|  |  |
| --- | --- |
| Same person | 0.59913665 |
| Different person | 0.30194122 |
| Real vs Fake | 0.43516958 |

В принципе модель отличает и других людей, и фэйковые фото, но **на лб** эта модель набрала лишь 0.3577 с общей **скоростью инференса** 8647.822365522385 секунд. Это уже сильно быстрее, чем прошлые модели, но все равно гораздо медленнее бэйзлайна. Поэтому было решено прекратить развлекаться с готовыми моделями и начать хард-ворк с дообучением чего-нибудь супербыстрого на предложенном датасете.

**2. Дообучение модели на предложенном датасете**

**Цели** данного этапа:

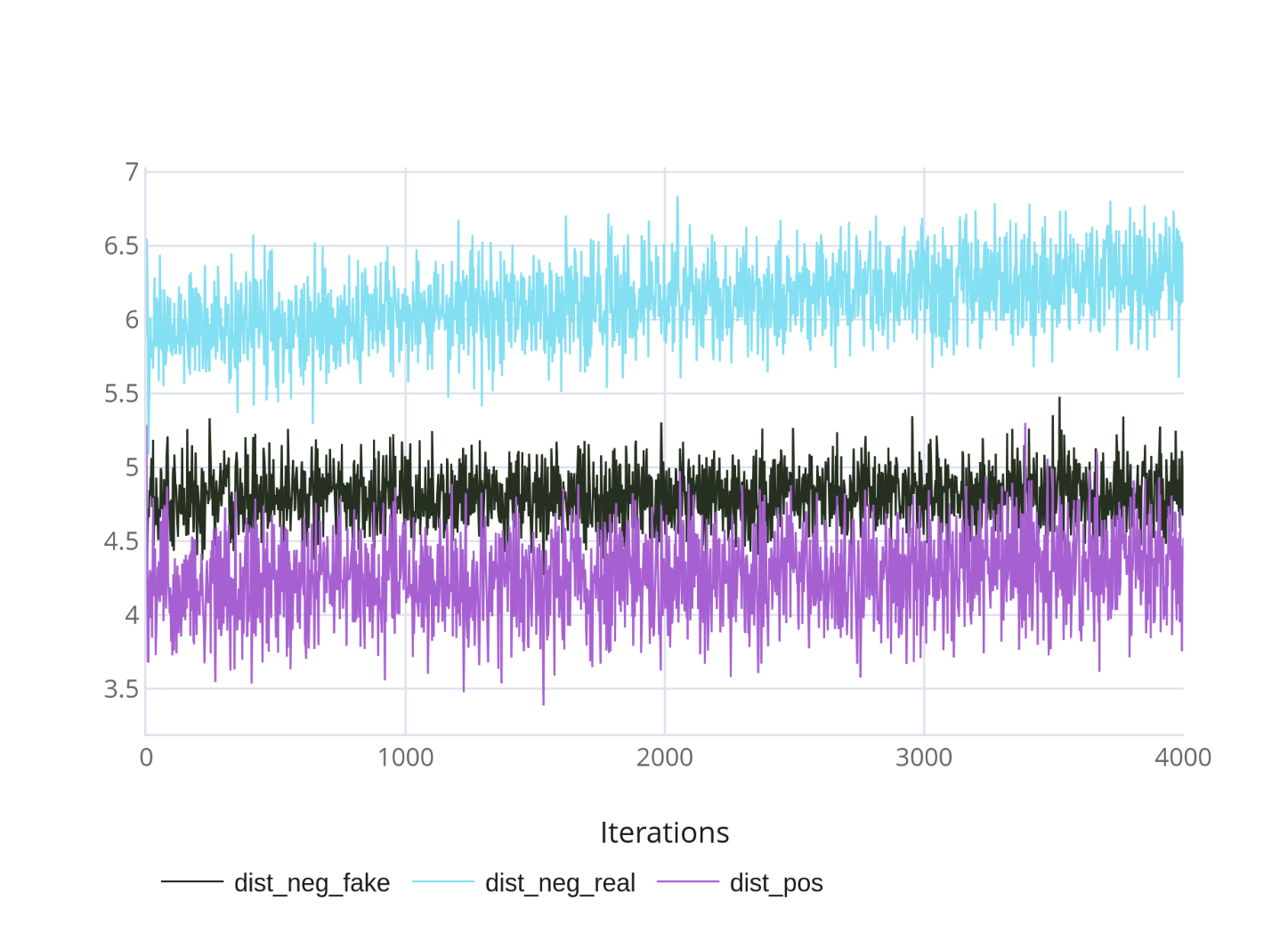
1) разработать лосс-функцию, подходящую под наши задачи,

2) подобрать бэкбоун для модели и

3) подобрать гиперпараметры модели.

**1. Выбор лосс-функции**

Мы провели предобученную на распознавание лиц модель facenet на базовом триплет-лоссе: модель неплохо различала разных людей, однако с задачей отличить фейковые фото она справлялась слабо. Ниже представлен график с измерениями косинусных расстояний между позитивными, негативными и фейковыми парами:



Для того, чтобы увеличить косинусное расстояние между реальными и фейковыми фото, был разработан кастомный лосс «Quadruplet\_loss», рассматривающий пару реальное-фейковое как негативный элемент классического триплет-лосса.

Таким образом, общий лосс складывается из двух триплет-лоссов:

negative\_real\_loss = torch.relu(self.margin + positive\_dist - negative\_real\_dist)

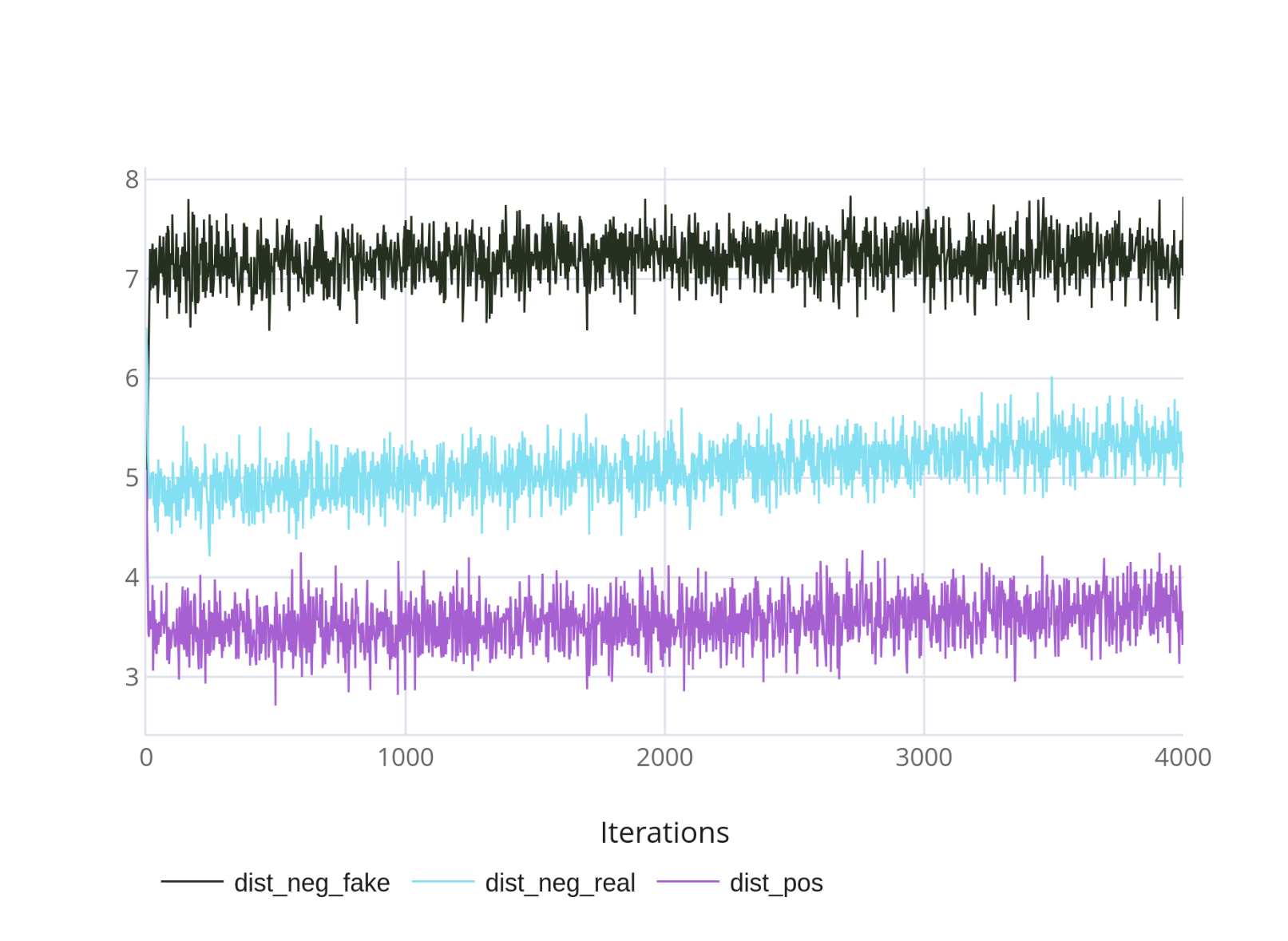
negative\_fake\_loss = self.lambda\_fake \* torch.relu(self.margin + positive\_dist - negative\_fake\_dist)

total loss = negative\_real\_loss + negative\_fake\_loss

Уже после 2 эпох обучения модель показала значительное улучшение результата по сравнению с классическим триплет-лоссом:

|  |  |  |
| --- | --- | --- |
| Method | eer\_real | eer\_fake |
| Triplet loss | 0.1041 | 0.425 |
| Quadruplet\_loss | 0.1374 | 0.025 |

Изменение разницы расстояний между реальными и фэйковыми фото после применения кастомного лосса представлено на рисунке ниже:

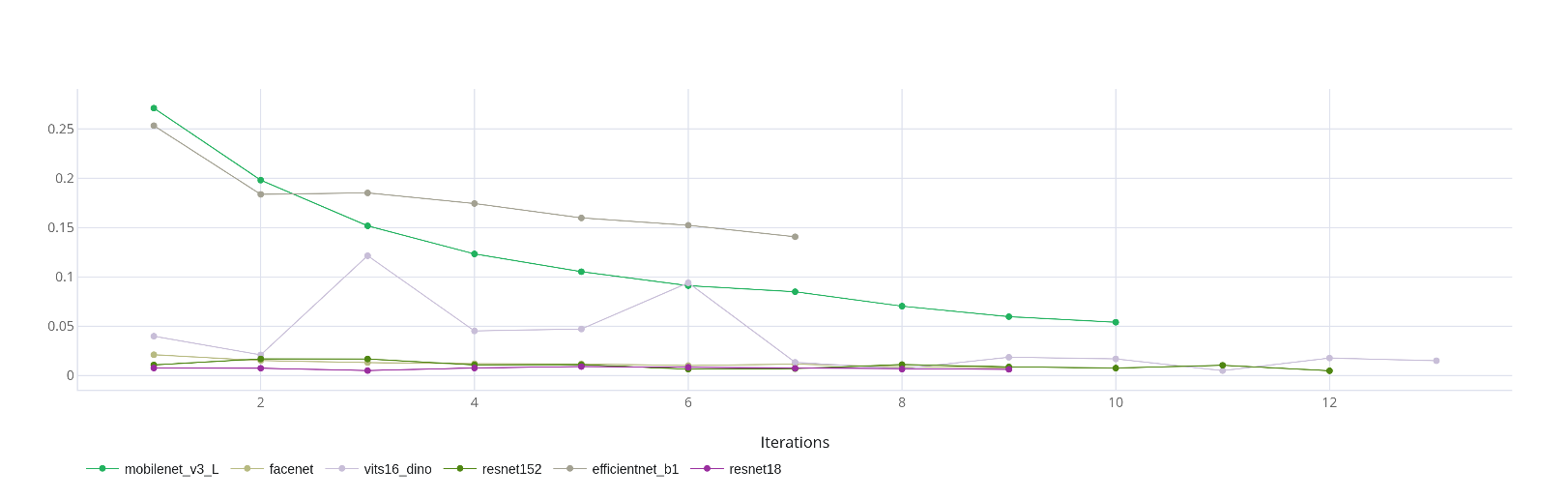


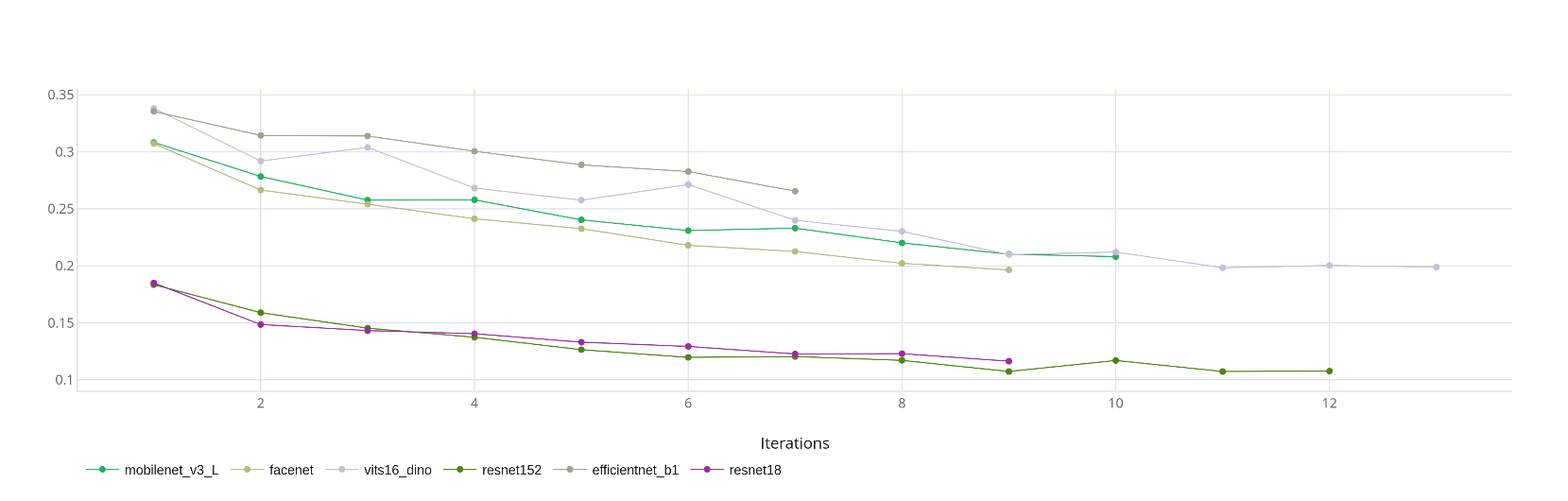
Таким образом результатом данного этапа стал выбор лосс-функции: **Quadruplet\_loss**

**2. Выбор бэкбоуна модели**

Были протестированы **7 различных бэкбоунов**, предобученных на ImageNet: Mobilenet\_v3\_large, Efficientnet\_b0, Fastface, Resnet152, Resnet101, Resnet18, Vits16\_dino.

Для того, чтобы точнее измерить разницу между парами: настоящее - фэйковое фото одного человека и настоящие фото двух разных людей, во время обучения использовались 2 раздичные метрики eer. Результаты оценки на валидации по этим двум метрикам представлены на графиках ниже.

Настоящее - фэйковое фото одного человека:

Настоящие фото двух разных людей: 

Те модели, которые на валидации показывали невысокую скорость / плохое качество, не проверяли на паблик-тесте.

Результат на паблик-тесте и скорость работы на машине, на которой проводилось обучение, представлены в таблице ниже ():

|  |  |  |
| --- | --- | --- |
| **Модель** | **Public test EER\*** | **Скорость \*\*** |
| Facenet | 0.0639 | 0.115536451 s |
| Resnet152 | 0.056 | 0.044033289 s |
| Resnet18 | 0.0632 | 0.018081665 s |
| Vits16\_dino | 0.0684 | 0.257577419 s |

\* *Так как расчет Public test EER был скомпрометирован, результат одного и того же сабмита за разные дни был разным – в таблице дан на момент загрузки.*

\*\* *Скорость получения одного эмбеддинга на той машине, где проходило обучение.*

Таким образом, несколько моделей показали очень близкий результат по метрике, поэтому была **выбрана Resnet18** как самая быстрая.

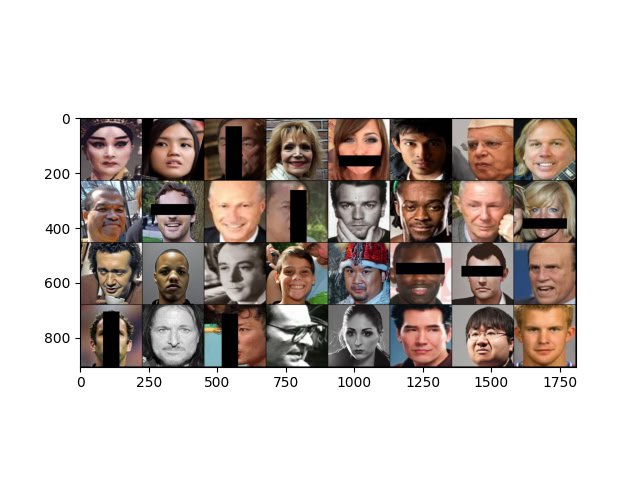
**3. Работа с датасетом**

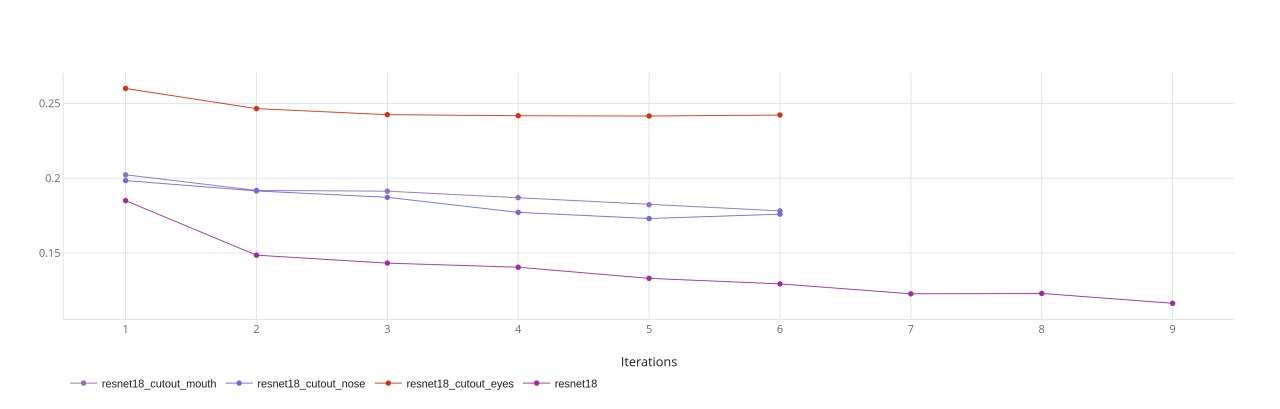
**Целью** данного этапа являлось протестировать различные улучшения датасета, которые могли бы повысить качество модели.

1. А**угментация** датасета для повышения робастности модели.

С помощью предобученной модели мы определили для каждого фото в трейновой выборке 5 основных точек на лице *(блокнот Alignment.ipybn**в папке other-experiments)*: центр левого глаза, центр правого глаза, кончик носа, левый угол рта, правый угол рта (координаты х,у для каждой точки), результат сохранен в points\_df\_with\_img\_index.csv.

На основании данных координат мы разработали следующие «патчи для лица» *(блокнот Polygons.ipybn в папке other-experiments)*:



Сравнение разных типов «патчей» с датасетом без аугментаций на валидации представлено на графике ниже:

2. **Кроп** трейнового датасета на предобработке для улучшения точности модели.

На основании все тех же 5 основных точек был сделан кроп всех фотографий трейновой выборки *(блокнот Crop.ipybn в папке other-experiments)*:

Фотографии были обрезаны таким образом, чтобы максимально исключить задний фон, волосы, головные уборы, и ресайзнуты в 224х224.

Оригинальные фото без кропа:

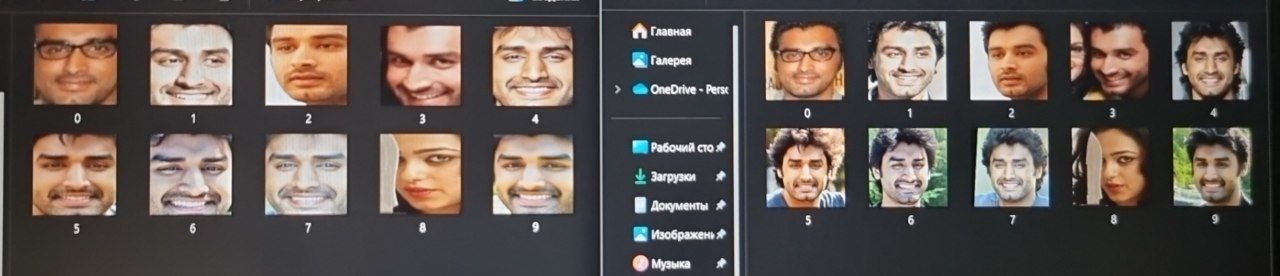
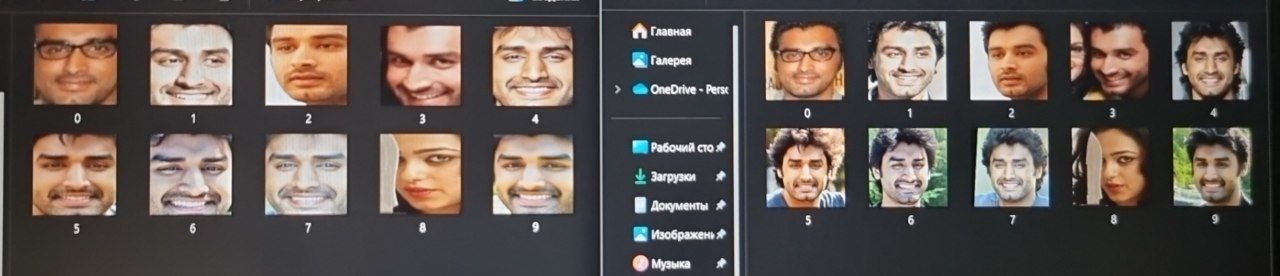


Фото с кропом:



К сожалению, данный подход так же не помог улучшить метрику ☹

Сравнение метрики после обучения на 2 эпохи на финальной модели resnet18 с применением кастомного лосса:

|  |  |
| --- | --- |
| **Датасет** | **Метрика** |
| Оригинальный | 0.1944476217031479 |
| С кропом | 0.23119506239891052 |

**4. Эксперименты по нахождению фэйкового фото без работы модели**

*(блокнот Artefacts.ipybn в папке other-experiments)*

**Цель этапа:** обнаружить отличия между реальными и фэйковыми фотографиями в самой файле без применения моделей глубокого обучения.

**Проведенные эксперименты:**

1) были протестированы различные цветовые фильтры, никаких особенностей замечено не было

2) были протестированы различные типы сверток на оригинальных фото. Диагональная свертка 3х3 показала, что при повышении яркости на фэйковых картинках отчетливо видны глаза, в то время как на настоящих они пропадают вместе с остальными частями лица. Были опробованы различные коэффициенты:

[0, 0, 1], [0, 0, 2], [0, 0, 3], [0, 0, 4],

[0, 1, 0], [0, 2, 0], [0, 3, 0], [0, 4, 0],

[1, 0, 0], [2, 0, 0], [3, 0, 0], [4, 0, 0],

В зависимости от собственной яркости и экспозиции фото, где-то нужен совсем небольшой коэффициент в свертке (1-2), где-то лучше видно артефакт глаз при коэффициенте 4.

Некоторые группы фотографий (например, id 000002 и 000006) "засвечивались" полностью с той сверткой, вероятно, имела место какая-то предварительная обработка фото.

Оптимальным коэффициентом для диагональной свертки на большинстве фото кажется 2:

[0, 0, 2],

[0, 2, 0],

[2, 0, 0]

Но требуются дополнительные эксперименты для уточнения коэффициента и разработки возможностей применения этого метода на различных этапах идентификации фейковых изображений. Для данного продолжения эксперимента, к сожалению, не хватило времени ☹