Кратко:
1) Я дико извиняюсь за скрины из датасферы, но я что-то забывал сохранять графики, когда обучал. Надеюсь, они не очень ужасные.
2) Видимо, я не так понял задание. Процентов 50 времени я уделил подбору оптимальной архитектуры… К концу этого задания я понимаю, что это было ошибкой, но прошу оценить мои старания)
3) Большую часть кода я комментил, если вдруг нужно будет почитать, что я делал. Итоговый код я не комментил.

До чекпоинта:

1) Для начала я решил использовать готовые архитектуры типа googlenet и resnet. Но они работали с расширением картинки 256*256, а у нас 40*40. Поэтому первое, что я сделал - написал свой блок googlenet.
2) Он не заработал - не было сходимости - я очень должно искал лажу (нашел уже сильно потом), но решил забить и написать свой Резнет18. В нем conv2d 7x7 был заменен на 5x5, потому что для картинок 40*40 7x7 мне показалось жирно. И убран maxpool. Ну и дальше число каналов пропорционально уменьшилось.
3) Оказалось, в самописном гуглнете я пихнул relu прямо перед финальным softmax'ом. Но уже работал резнет. Я проверил разные оптимизаторы, норм вариант получился первые n эпох учить Adam'ом, а вторые SGD.

После чекпоинта:

Первые дней 5 я возился с архитектурой сети:

Я таки дописал блок гуглнета, застакал несколько штук - и эта конструкция все равно давала отстойное качество (но уже обучалась). Поэтому решил допиливать резнет.

Для начала я переписал блок резнета18 на универсальный блок с произвольным числом сверточных слоев, добавил туда возможность дропаута

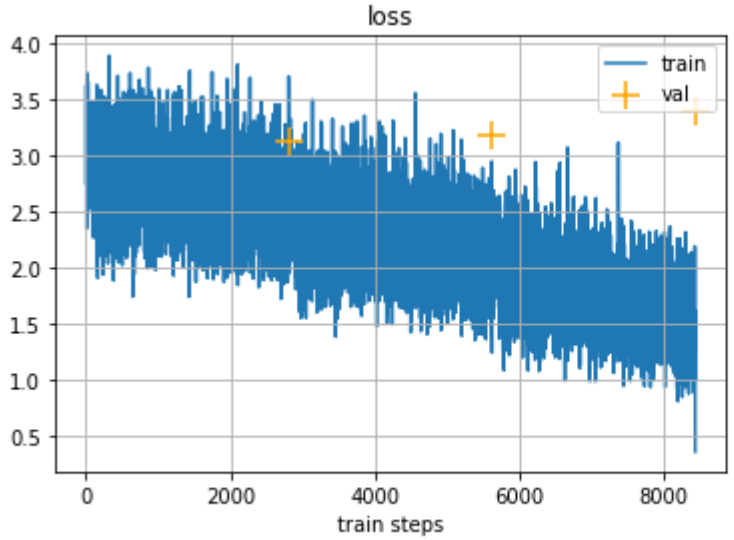
Далее все сравнения я проводил на 3 эпохах с адамом в качестве оптимайзера
Переписанный (но тот же) бейзлайн из чекпоинта выдавал аккураси 0.26

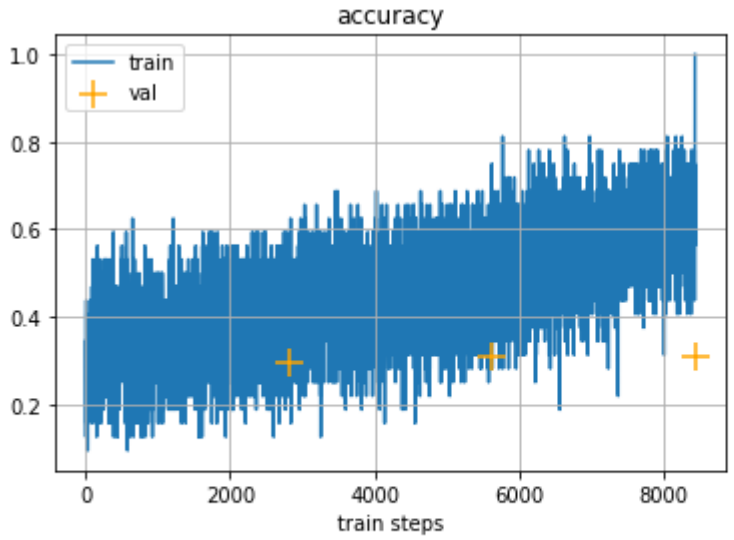
Потом попробовал первый слой сделать не 5*5, а 7*7, запустил по 3 раза - значимого эффекта не было

Потом я попробовал сделать почти в 2 раза больше блоков - дублировал каждый блок, который не понижал разрешение. Роста качества почти не дало ("почти" я буду иметь в виду не больше 1%)

Важно, что до сих пор сетка не начала переобучаться, значит можно еще накидать на вентилятор. Добавим еще один блок понижения размерности в конец. Опять незначительно((

Дальше я достаточно долго пытался заставить свою сетку начать переобучаться. Из-за skip-connections это было сделать непросто, но я увеличил кол-во сверток в блоках (с 2 до 3-4-5), напихал еще таких блоков. По графикам сетка начала переобучаться и уже на первой эпохе выдала 0.3.






```
Epoch: 2, val loss: 3.383990454978455, val accuracy: 0.308805912733078
```

Дальше попробовал layer norm. Добавил 5 штук в случайных местах после блоков. Но качество просело до 0.27, попробовал сигмоиду вместо релу после первого слоя - тоже 0.27

Дальше пробовал менять кол-во блоков и их глубины, перебрал вариантов 20 разных, побить 0.3 не вышло. Переобучиться тоже не получилось. Такое ощущение, что этот график выше - просто повезло, его так и не получилось воспроизвести(.

Дальше я решил сделать несколько веток резнета - в конце перед полносвязным их склеить.

Выбрал 2 лучшие и разные модели, поставил на ночь на 30 эпох, результат: +2% аккураси((

Дальше я вкорячил layernorm вместо batchnorm в самих блоках - качество на 3 эпохах просело до 0.23, идея плохая

На этом моменте я в полном отчаянии: спустя 2 дня тупления лучшая модель все еще модель из чекпоинта. Еще я понял, что не получатся объективно мерить качество, тк на 3 эпохах даже разные запуски одной модели дают разное качество (+-1%). Более того: на 3 эпохах сложно обнаружить переобучение. Поэтому теперь будет сравнивать модели на 5 эпохах. Теперь бейзлайн - 0.33. Начнем снова перебирать архитектуры.

Поскольку игры с глубинами блоков и их кол-вом успехом не увенчались, я попробовал сделать несколько выходов для ответов (как у гуглнета), нормализовать (layernorm) их, а потом просто сложить. Я сделал 3 такие ветки. На 5 эпохах такая штука начала переобучаться, но на валидации качество оставалось тем же (0.33).

Пока у моего резнет18 качесвто все еще как у бейзлайна, а значит большого прироста качества в итоге не ждать, я решил пойти в сторону резнет50. Написал его блоки, запустил. Качество на дефолтном было сильно хуже (0.11), начал убирать у него блоки - качество росло. Так, когда остались почти только те блоки резнета50, которые изменяли размерность картинки (т.е. это всего 5-6 блоков было), качество стало 0.17 Неплохо, но учитывая то, что у изначального резнет18 0.33 выглядит, как тупиковый путь…

Заходим с другой стороны. Пропробуем гуглнет еще раз. Я поправил все баги, сделал сетку из 4 последовательных “гуглблок-свертка 3*3” (ну с началом и концом, как в резнете). Такая штука выдала 0.17. Неплохо для первого раза, попробовал bn и ln: bn дал 0.27, ln дал 0.17.

Прошел очередной день поиска вариантов по улучшению качества. Ничего не помогало. Поэтому план был такой: **1) найти оптимальную архитектуру, 2) подобрать под нее оптимальные гиперпараметры, 3) подобрать хорошую процедуру обучения** (когда качество на валидации начнет падать). Обучить модель на всех данных и заслать.

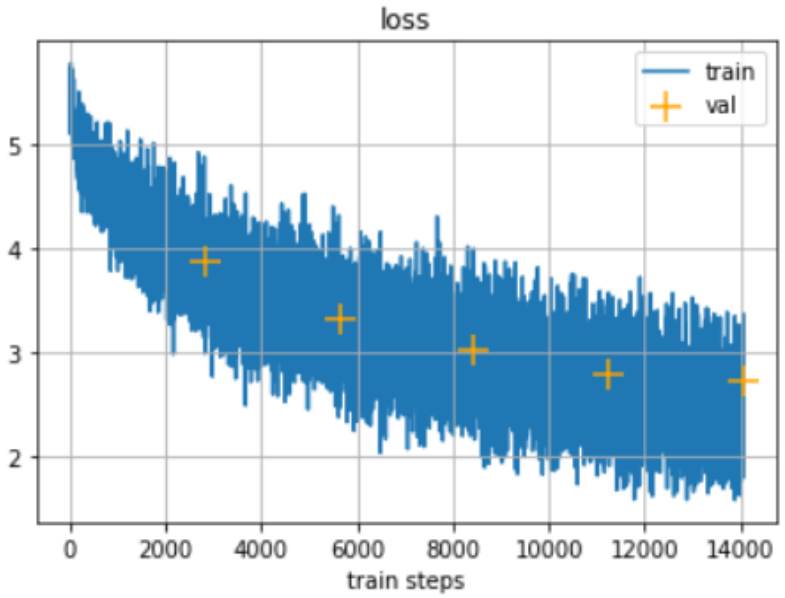
1) Начнем с первого - поиска архитектуры. На данный момент у меня написаны с разными настраиваемыми параметрами блоки гуглнет, резнет18 и резнет50. У вторых есть возможность понижать разрешение картинки в 2 раза по стороне (у увеличивать кол-во каналов в 2 раза). Т.к. размеры исходной картинки 40*40, то логично делать 3-4 понижения. Поэтому я запустил на ночь перебор лучшей архитектуры. То есть я выбирал случайную длину, потом в ней случайно выбирал места, когда понижать размерность, а потом случайно на каждое место выбирал один из 3 блоков.

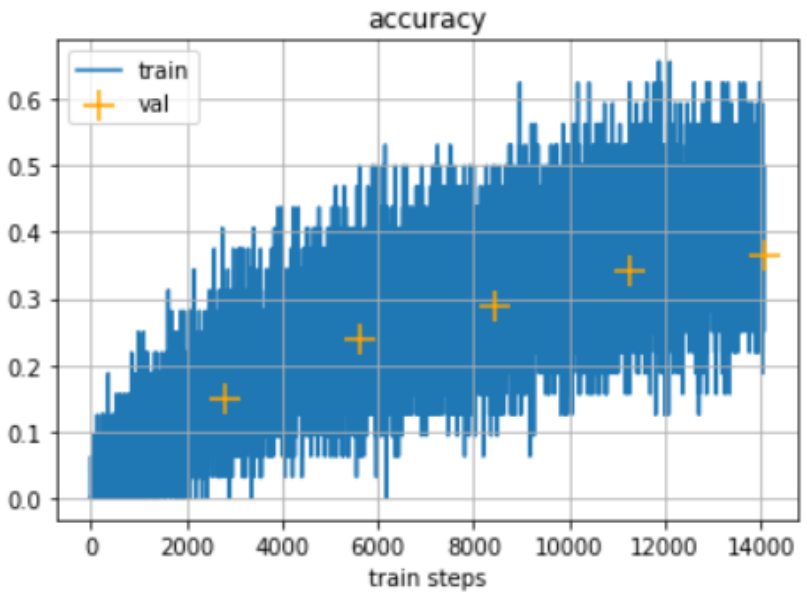
Предварительно я сделал запуски на 10 случайных архитектур из этих блоков (разной длины) и оказалось, что лучшие из них те, в которых почти нет резнет50. Поэтому я его выкинул. Потом позапускал еще раз 30 - оказалось, что, где минимум гуглнета, там тоже хорошо. И его я выкинул. Поставил перебор блоков резнет18 (разной глубины) на

ночь. Оказалось, что лучший вариант имел такое же качество, как текущая СОТА(. Я в отчаянии...

Дорогой дневник, мне не передать ту боль, которую я испытал, открыв ноутбук утром. Я надеялся увидеть, что датасфера мне переберет 100 моделей и найдет лучшую, у которой будет качество в 2 раза выше, чем у бейзлайна. Но в реальности она перебрала всего 20 и у лучшей качество выше на 1%.

Идем дальше: я запустил перебор по возможным kernel_size и out_channels на первой свертке, нашел оптимальную. Потом запустил перебор по случайным архитектурам длины 6-12 из блоков резнета18 глубины 1-3. Оказалось 1) лучше делать 3 downsamples, чем 4. 2) все архитектуры примерно одинаково неплохи, но лучшее качество выходит у резнета, глубиной 10 блоков из блоков разных глубин. На этом, я, считаю, можно заканчивать с выбором архитектуры. Текущая SOTA на 5 эпохох имеет качество 0.38:



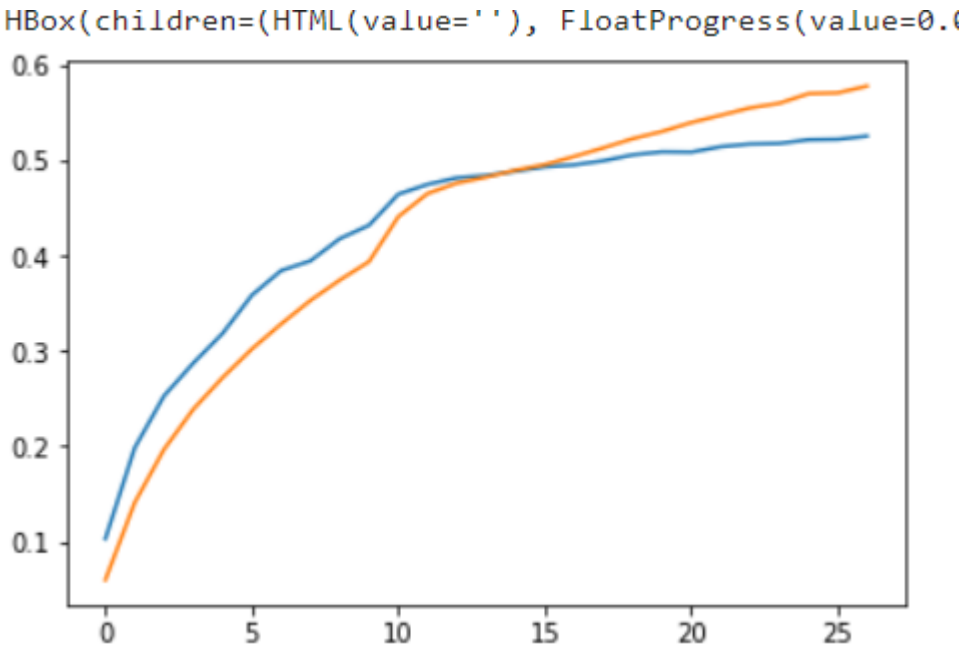



```
Epoch: 4, val loss: 2.7229991324793414, val accuracy: 0.36381790041923523
```

(Да, на графике 36, но, видимо, я тут что-то выключил типа дропаута)

2) Подбор гиперпараметров. На этом я фиксирую свою оптимальную модель и перехожу к ее обучению. Естественно, для начала врубаем аугментации. Теперь я буду перебирать следующие вещи: batch_size, dropout_p, momentum, bias (true/false) в слоях. Это дело на 5 эпохах. Потом параметры оптимизаторов. Оказалось, что дефолтные значения лучшие(Только dropout_p получился другой (0.3). batch size оставим на конец.

Начинаем играться с методами оптимизации. Тут нам уже нужно не 5 эпох, а ждать по конца. Первые 10 эпох я решил учить адамом - было неплохо. Дальше на 20 эпох я поставил oneCycleLR, получилось неплохо. Возьмем эту процедуру за бейзлайн.



Синяя - валидация (5%) Рыжая - трейн (95%)

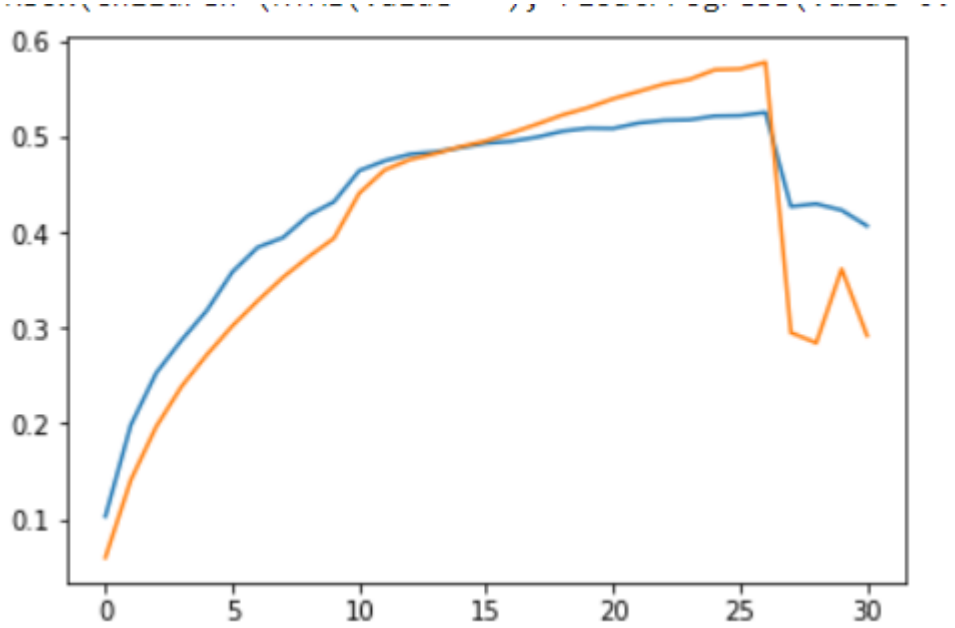
Тут на самом деле мало что можно написать. Бейзлайн, обученный на 95% выдал качество 0.50 на тесте (моя посылка на 0.50). Дальше я лишь перебирал все возможные методы и расписания к ним по дообучению бейзлайна. Не помогало почти ничего - какие-то методы просто дальше ничего не улучшали и жили на плато (как здесь)

ЗДЕСЬ ГРАФИК

accuracy на валидации. SGD с расписанием *0.1 каждые 20 эпох.

На похожая картинка встречалась часто.

Какаие-то методы сразу роняли качество и не стремились восстанавливать, как здесь:

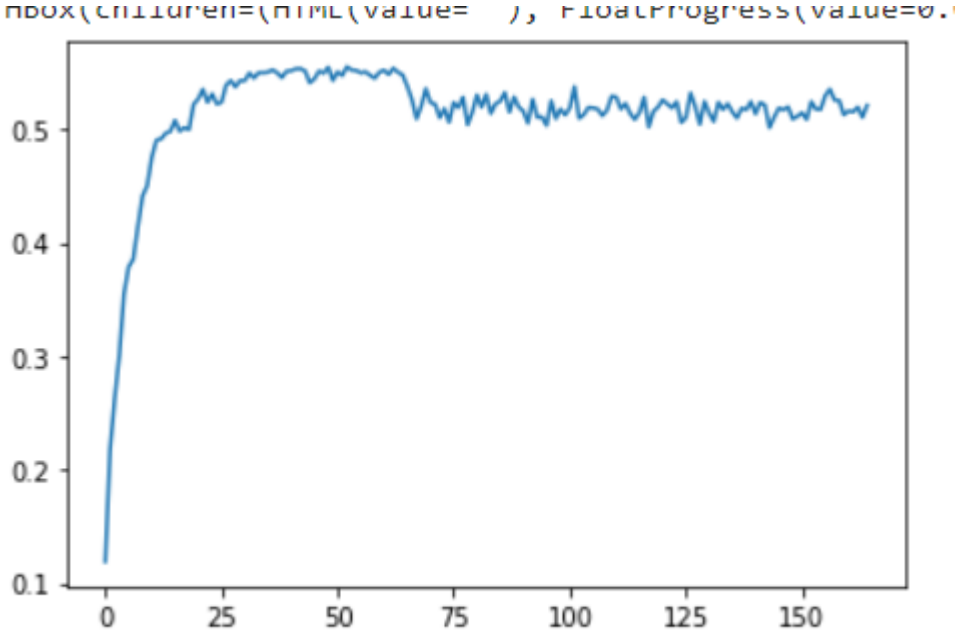


RMSprop с неудачно высоким lr.

Какие-то методы улучшали качество на трейне, но валидацию это не спасало.

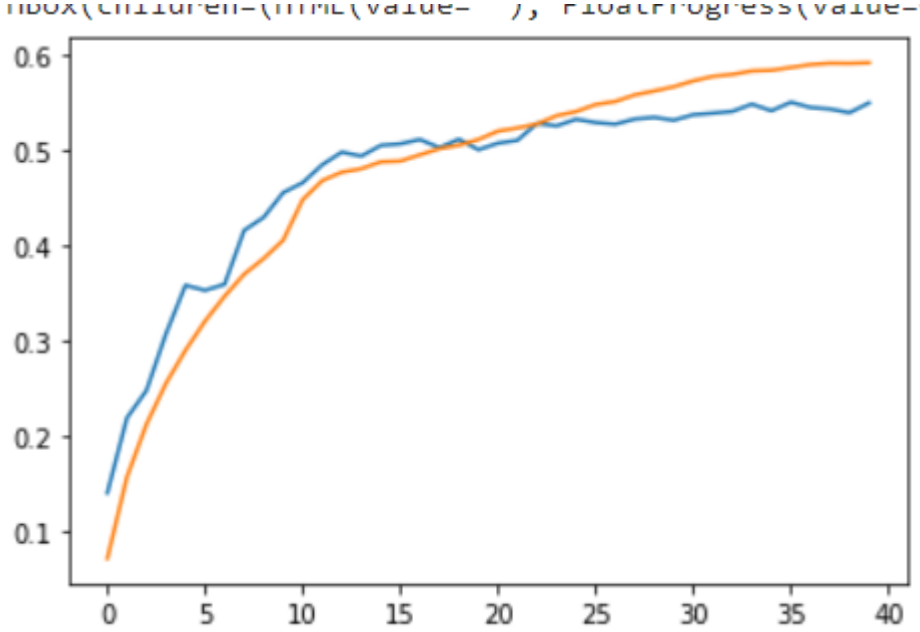
Я не хочу писать все в отчет, что я попробовал касаемо методов дообучения, но я пробовал следующие вещи:
optimizers: Adam, SGD, RMSprop
schedulers: OneCycleLR, CyclicLR, CosineAnnealingWarmRestarts, StepLR, CyclicLR, ReduceLROnPlateau с различными гаммами (от 0.1 до 0.9 за разное кол-во этох)

Далее пошли в бой последние отложенные штуки: mixup, перебор batch_size, обучение на 99%. Последнее подняло качество до 55% на валидации. mixup только ухудшил дело:



0.5208984613418579

SGD без mixup держал плато на уровне 55%. С mixup сначала просел до 52%, потом тоже ровно.
Для сравнения есть тот же график (до 40 эпохи без mixup) с отслеживанием трейна:

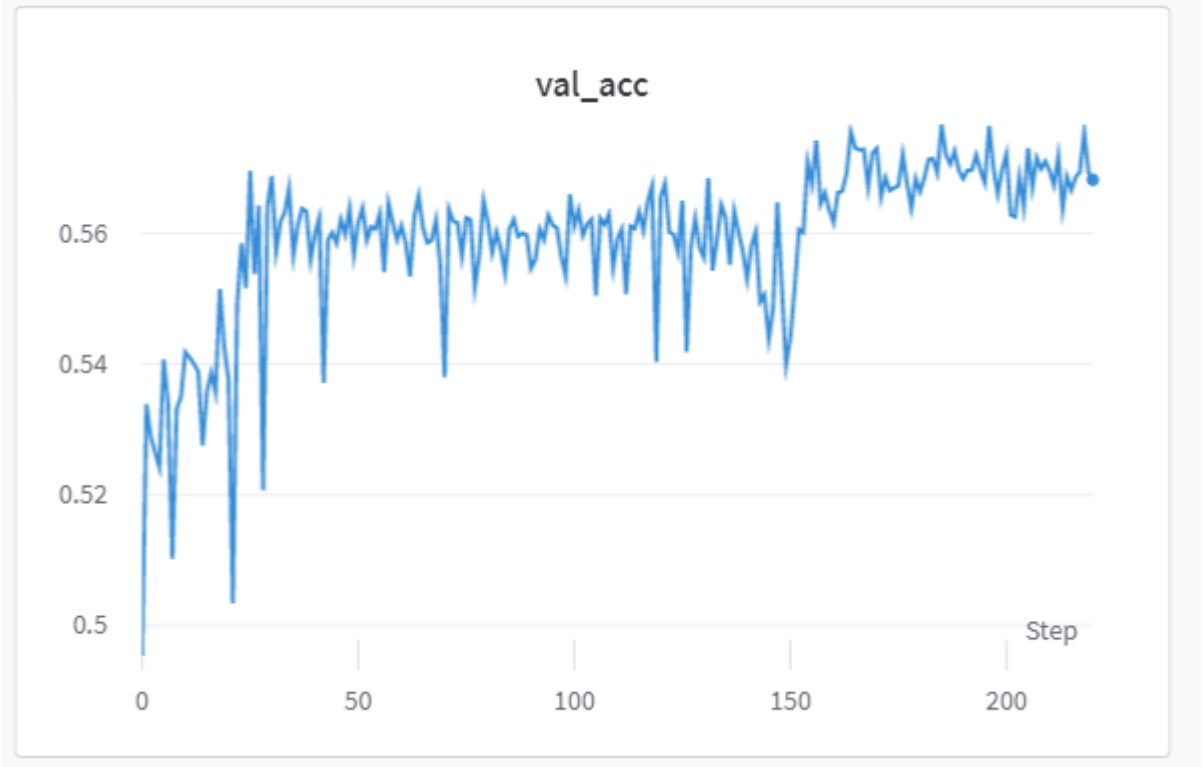


Синяя- валидация, рыжая - трейн

Дальше я перебрал размеры батчей для этого метода обучения, резы такие:
16: 0.5353816151618958
32: 0.5288461446762085
64: 0.5480769276618958
128: 0.5505558848381042

Поэтому итоговое обучение проведем на 128.

3) Финальное обучение
Берем предобученную модель, ставим расписание цифара (0.1 каждые 30 эпох) с SGD, обучаем на 150 эпох. mixup не включаем. Ожидаем.
К слову, на валидации такое обучение дало 57% аккураси:

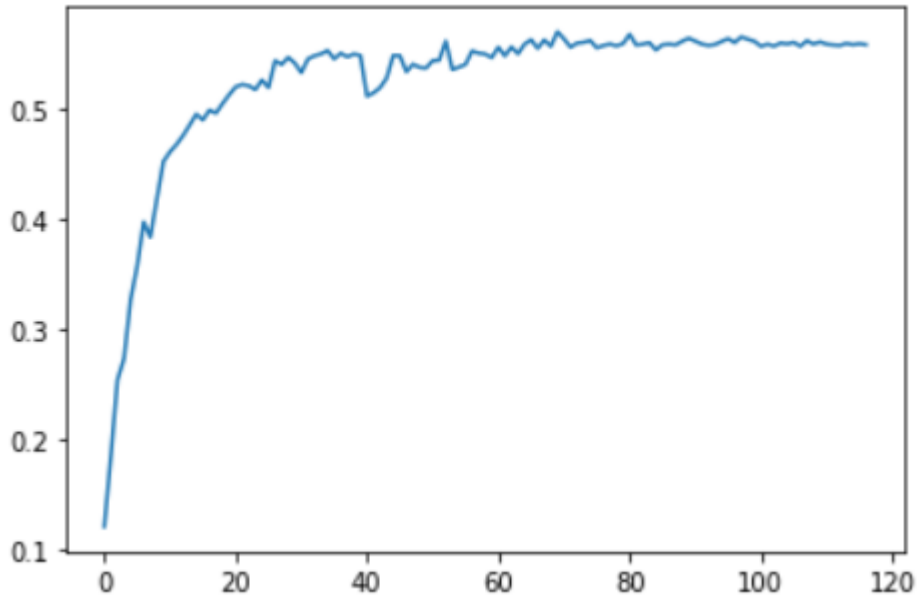


После 150 эпохи я выключил mixup

Но на тесте было 53.5(

Решил сделать последнюю отчаянную модификацию: запихнул в свой резнет блоки глубины 4. И запустил на той же процедуре обучения. Итог: 54.6%...
График обучения итоговой модели (качество на валидации):

HBox(children=(HTML(value=''), FloatProgress(value=0.



Итог.

Что попробовал:

1) разные блоки (googlenet, resnet18, resnet50)
2) разные глубины сетей
3) разное кол-во downsamples
4) несколько веток модели (несколько выходов)
5) разные гиперпараметры в архитектуррах
6) dropout с подбором p
7) layernorn/batchnorm
8) свертку первого слоя
9) аугментации, нормировку картинки
10) swa - про нее не писал, но попробовал в начале - ничего не дало
11) оптимайзеры (Adam, SGD, RMSprop)
12) расписания к ним (OneCycle, Cosine, Step, ReduceOnPlatou)
13) mixup
14) размеры батчей