Отчёт по большому домашнему заданию 1

Введение

Понятное дело, что в первую очередь нам нужно определиться, как мы будем обрабатывать входные данные, какая модель подходит для нашей задачи и т.п. Давайте обо всём по порядку.

Промежуточный чекпоинт

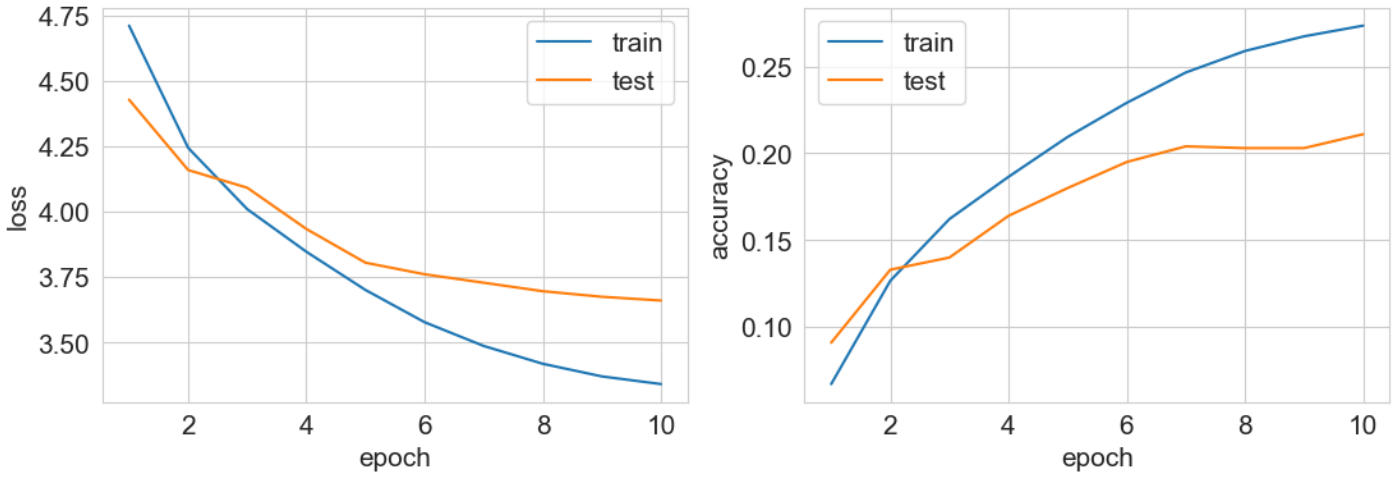

Рис. 1: График базовый.

Ну, тут я особо не думал и ничего почти не делал, просто взял с одного из семинаров функции для обучения, скрестил это с базовой моделью, которая у нас была во второй мальнькой домашкой и с лёгкостью получил 20% на тесте. Пришёл, увидел, победил, ничего более умного.

Основной этап

Данные

Я использовал некоторые аугментации. В финальной можели я использовал следующие: `RandomHorizontalFlip`, `RandomVerticalFlip`, `RandomGrayscale`, а также `GaussianBlur` и другие. В ходе экспериментов я пришёл к тому, что это самые лучшие аугментации из тех, которые я использовал. Так же предварительно я посчитал среднее значение и разброс по всему датасету, чтобы более корректно применить `Normalize`.

Модель

На лекции нам рассказывали про то, что `ResNet` как раз таки подходит под такие задачи. Поэтому за основу своей модели я решил взять `ResNet50` (код я скомуниздил из документации). Я рассматривал и более простые версии `ResNet`, но побоялся, что её мощей не хватит, чтобы корректно выучиться с высокой точностью. Для обучения я использовал `SGD`. Lerning rate я понижаю на плато (`ReduceLROnPlateau`).

Ход работы и лучший результат

Для начала я обучил обычный `ResNet50` без каких-либо модификаций. Как и обидалось, результат был не очень, accuracy было порядка 12%. К сожелению к более ранним экспериментам у меня графики не сохранились.

Потом я решил убрать `MaxPool2d`, потому что все таки у нас изображегние достаточно маленькие, возможно нам это и не нужно. Также я поменял параметры для первого слоя `Conv2d`. После этого обучение модели кратно замедлилось, но зато качество чуть повысилось (примерно до 15%).

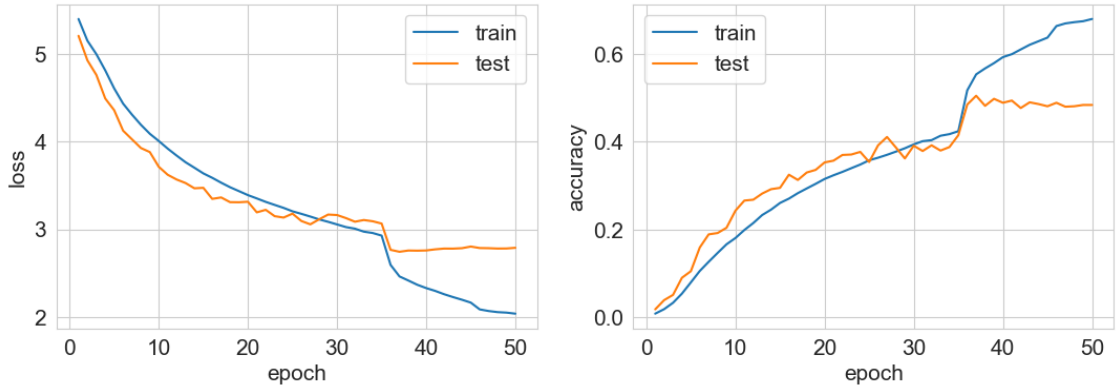

Рис. 2: График с Dropout(30).

Однако нельзя было не заметить, что модель очень сильно и быстро начинает переобучаться. Поэтому я решил добавить регуляризацию: `Dropout` с параметром 0.3 и параметр `weight_decay` в `SGD`. После этого результат заметно улучшился до 48% на валидации и 44.36% на тесте.

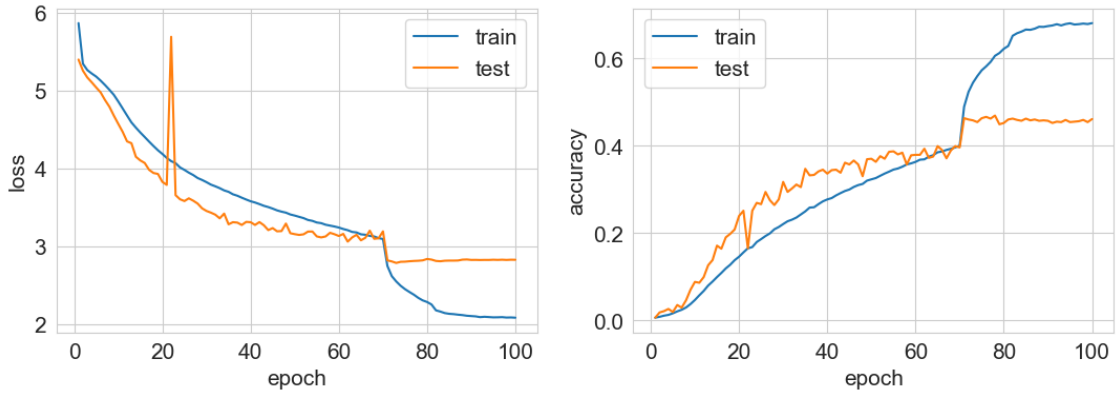

Рис. 3: График с Dropout(80).

Были и другие попытки. Я подумал, что можно было бы попробовать увеличить значение `Dropout` например до 0.8. Результат особо не улучшился, даже можно сказать, что в каком-то смысле слегка ухудшился, он стал где-то 46% на валидации. Но мне показалась, что эта модель всё таки более устойчивая к переобучению, и что её потенциал ещё не раскрыт.

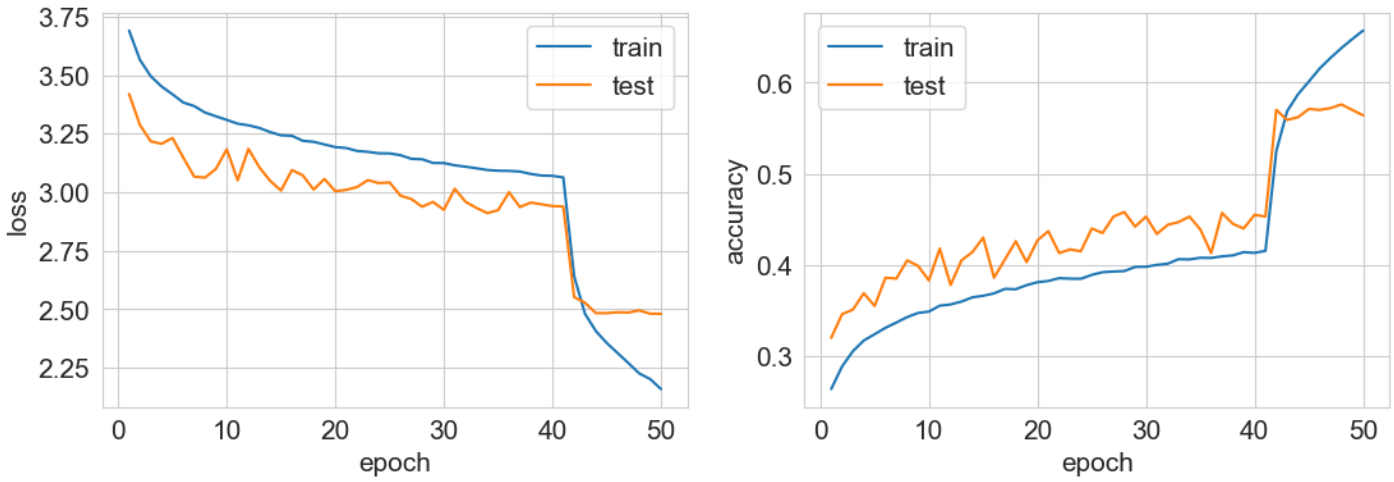

Рис. 4: График итоговый.

Я пробовоол дообучить обе модели. Что значит дообучить? Да просто взял немного другие аугментации, уменьшил чуть-чуть `batch_size` в `DataLoader` и взял немного другие параметря для learning rate и запустил ещё раз уже обученную модельку. Странно, но в модели с `Dropout` 0.3 почему-то особо качество не улучилось. Возможно это я дичь просто какую-то сделал. А вот с `Dropout` 0.8 результат прям реально поднялся, просто невероятно до 57% на валидации. В итоге получилось 50.84% на тесте.