Генеративные модели. Домашнее задание #1. Автоэнкодеры.

Dmitry

Введение

В домашней работе предлагается происследовать преимущества и недостатки различных архитектур автоэнкодера. Задание можно разделить на 4 составляющие:

- Обучить разные архитектуры автоэнкодера на Omniglot
- Обучить классификатор на MNIST и посчитать FID для обученных автоэнкодеров
- Обучить классификатор на Omniglot
- Обучить классификатор на латентных представлениях автоэнкодеров и сравнить результаты с предыдущим пунктом

Структура и организация кода:

Экспериментировать в jupyter notebook крайне неудобно, более того, надоело писать одноразовый код для каждого домашнего задания. Возможно, мое решение не самое лучшее, но я решил сесть и написать свой "фреймворк" для обучения сеток. Конечно, написан он пока на коленке, но так или иначе, проводить эксперименты с ним удобнее.

В самой тетрадке я ничего не писал, вместо этого запускал скрипт для запуска экспериментов через командную строку. Аргументы можно посмотреть в самом начале файла main.py.

- Репозиторий

- Код main.py для экспериментов
- Логи экспериментов

Архитектуры автоэнкодеров

Ниже представлены изображения до и после применения к ним автоэнкодера. Как можно заметить, ванильный и denoising автоэнкодеры достаточно хорошо справляются с задачей, хотя и не идеально. Sparse автоэнкодеры с другой стороны, справляются значительно хуже. Что касается автоэнкодера обученного с добавлением KL лосса, то там очевидной проблемой является сигмоида в активации. По-хорошему, стоило не добавлять сигмоиды, а сделать по аналогии с L1 регуляризацией, но такой код был на семинаре, а я уже просрочил дедлайн на 3 дня. На самом деле, не очень понятно, почему тут должны были зажечь sparse автоэнкодеры, ведь если я правильно понимаю, то они были призваны лечить проблему с переобучением обычного ванильного автоэнкодера, но как мы можем заметить на графиках лосса, значение функции потерь на тестовой выборке также падает. Возможно, стоило значительно дольше учить сетки. Я учил где-то 20 эпох.

Что попробовал:

- Для каждой архитектуры чутка поварьировал количество каналов в свертках и количество сверток. К сожалению, эти эксперименты были проведены еще до того момента, как я написал логгирование на wandb. В целом, я остановился на 64 каналах и итоговом размере 4x4 в латентном слое. То есть, в них в 4 раза меньше информации, нежели в оригинальном изображении.

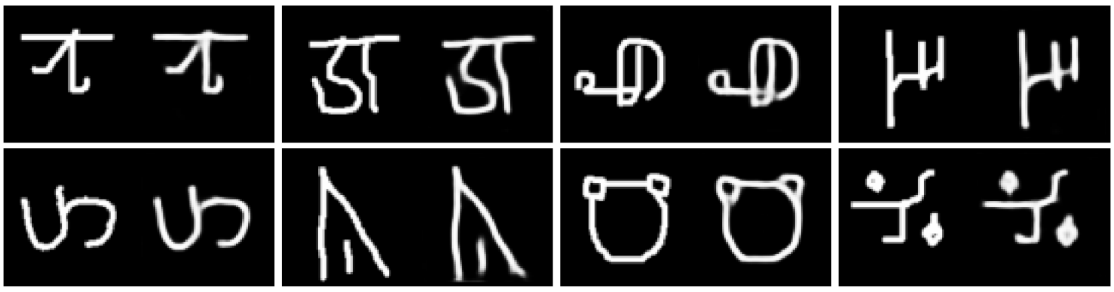
Что не успел:

- Нормальные эксперименты с sparse автоэнкодером с KL лоссом.

- Код обучения

Vanilla Autoencoder

- Код модели



Denoising Autoencoder

- Код модели

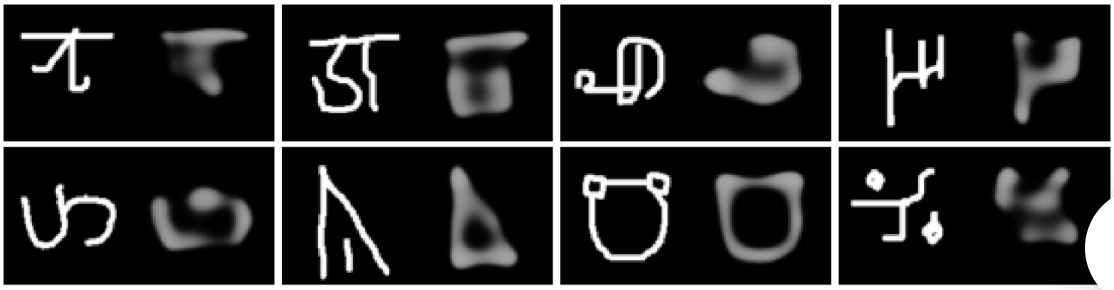
На семинаре показывали DAE с шумом на каждом слое, но он не пропадал на валидации, хотя кажется, что должен. Поэтому, я написал Noise, который не добавляет шум в режиме `eval`.

Sparse Autoencoder

KL

- Код модели
- Код лосса

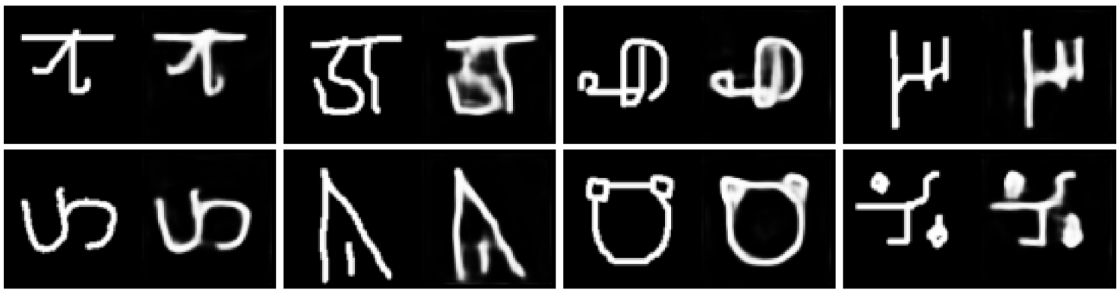
Очевидно, что из-за сигмоид градиенты просто не проходят. Какие-то очертания правильные, но с таким подходом вряд ли можно добиться нормального качества. Странно, что такое показывали на семинаре.

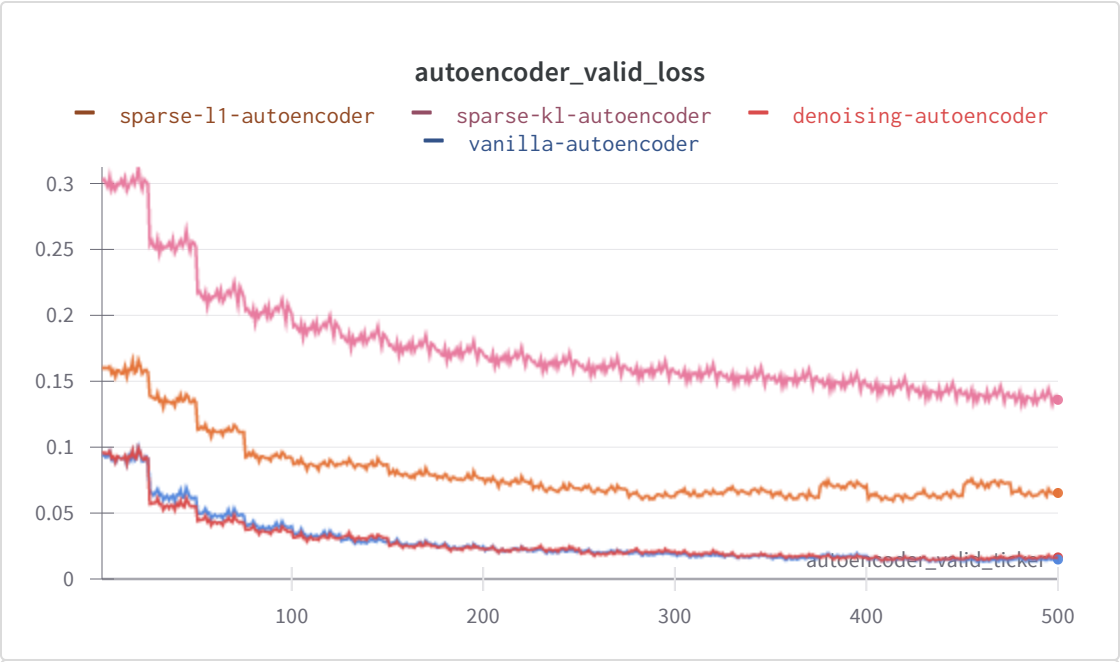


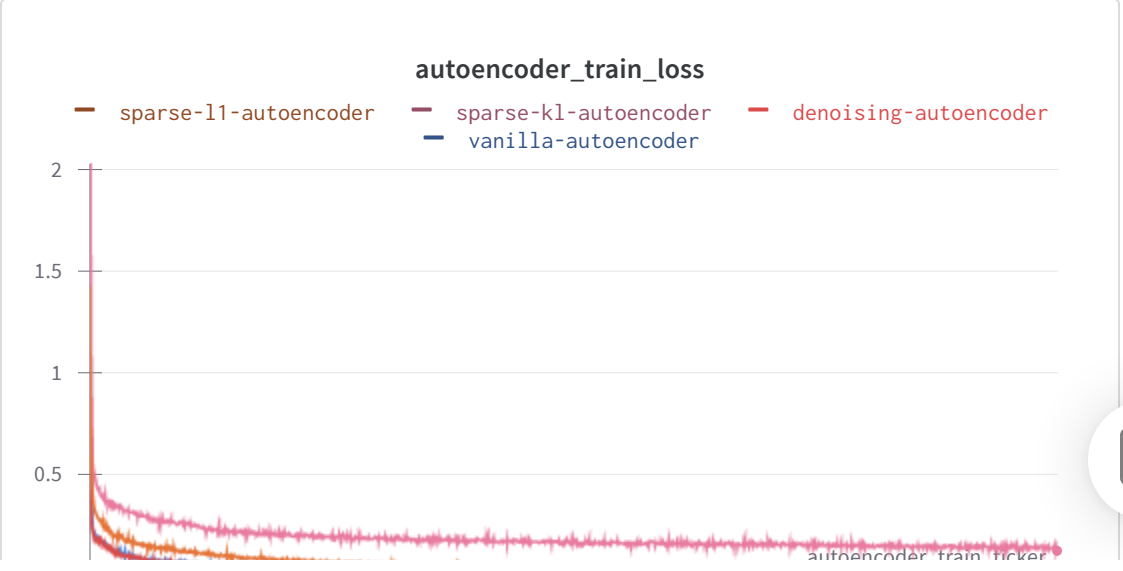
L1

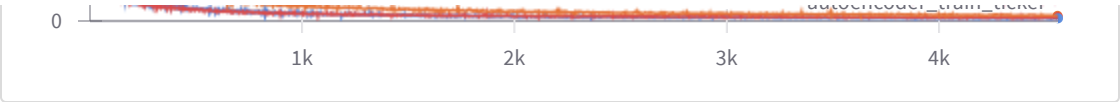
- Модель совпадает с моделью обычного автоэнкодера
- Код лосса

Картинки заметно хуже, чем с ванильным или denoising автоэнкодерами.





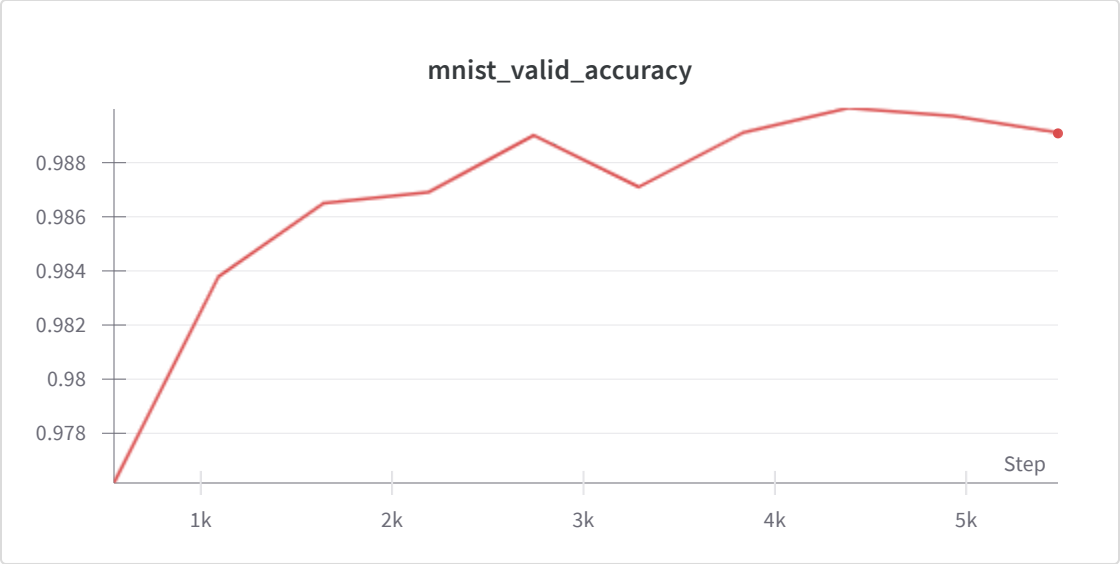


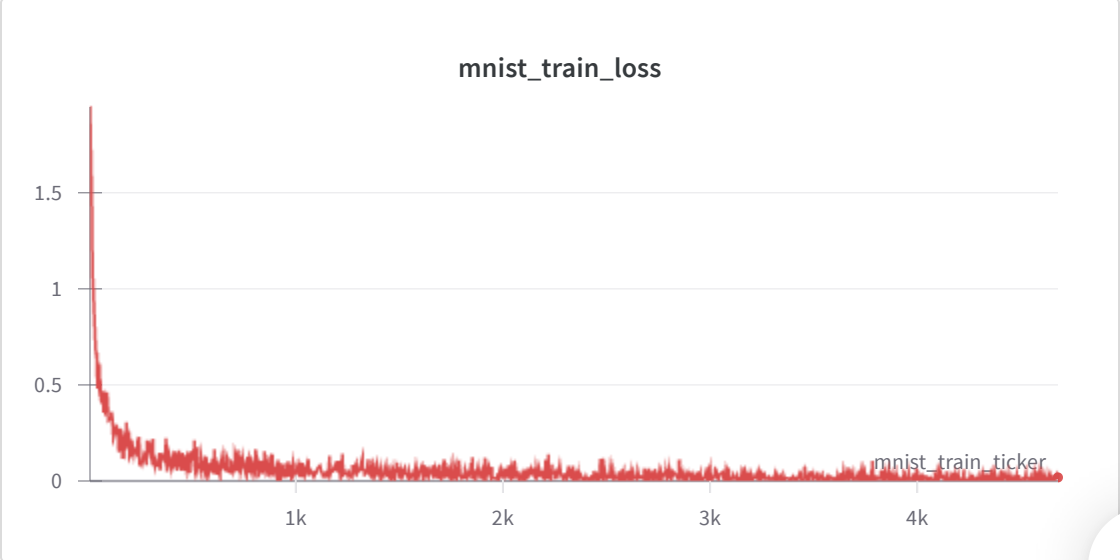


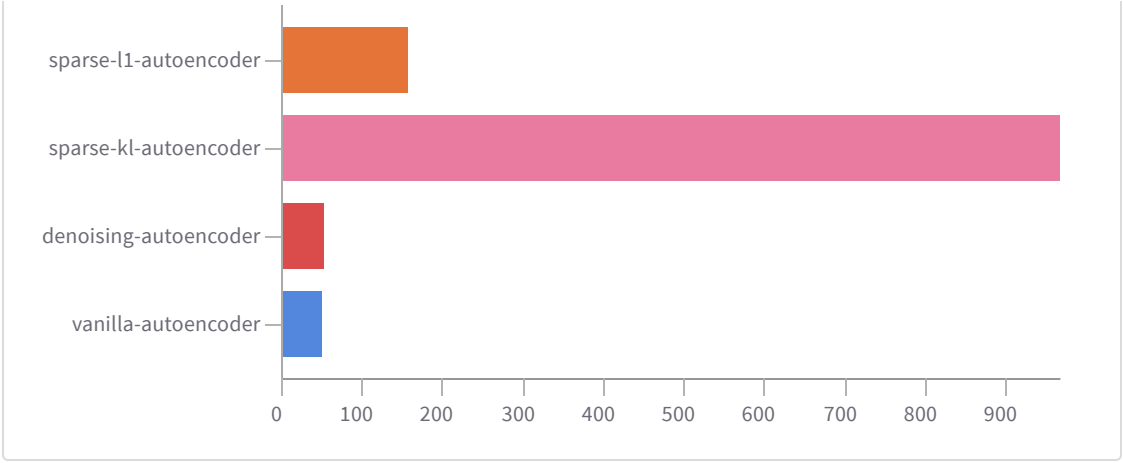
FID

- Код подсчета FID
- Код MNIST модели
- Код обучения MNIST

Как мы видим, FID у DAE и Vanilla AE примерно на одном уровне и значительно ниже, чем у Sparse AE, что в принципе не удивительно.



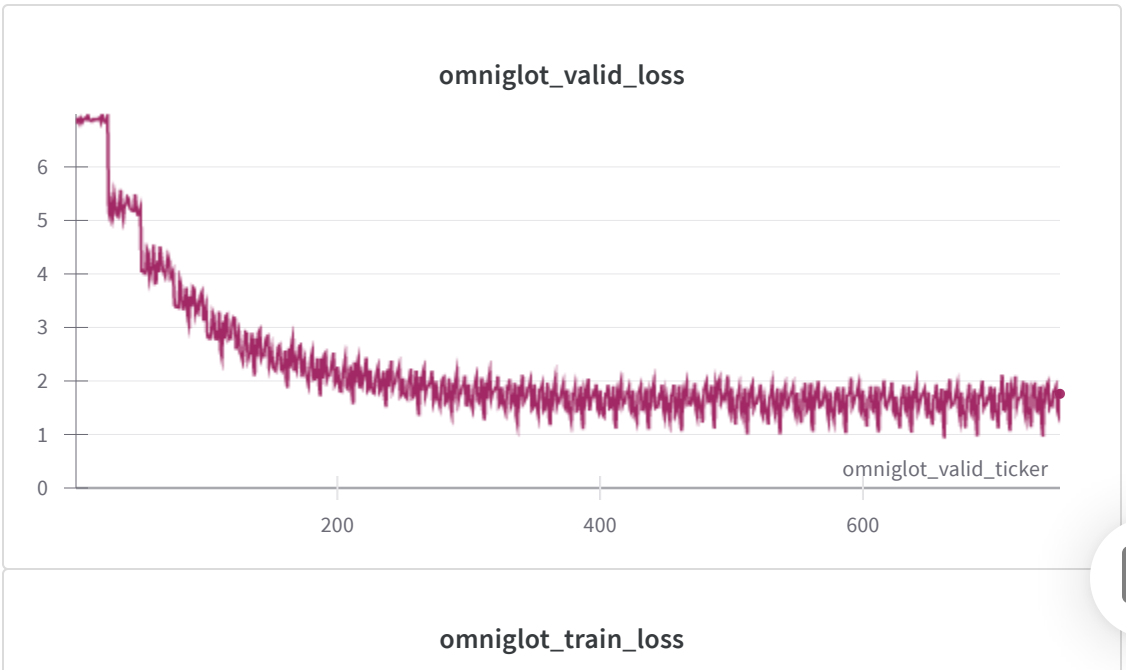


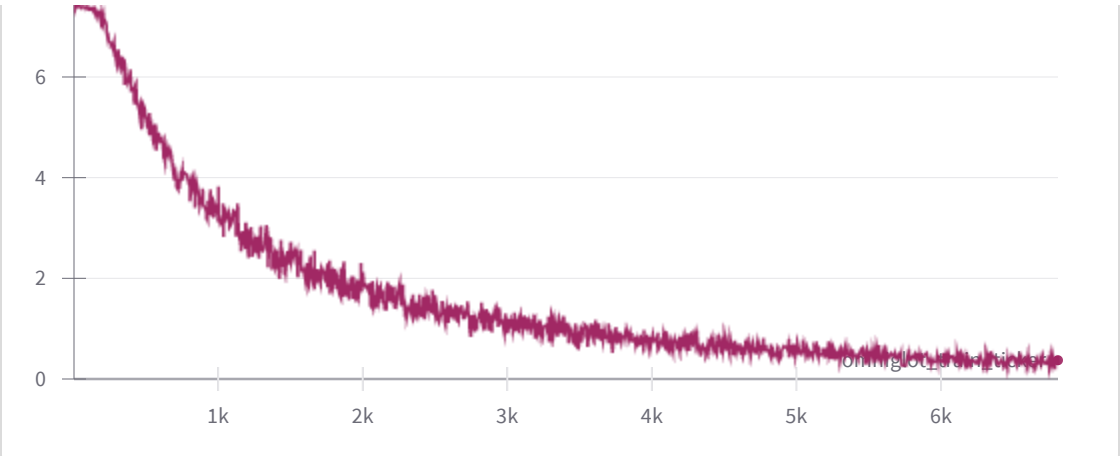


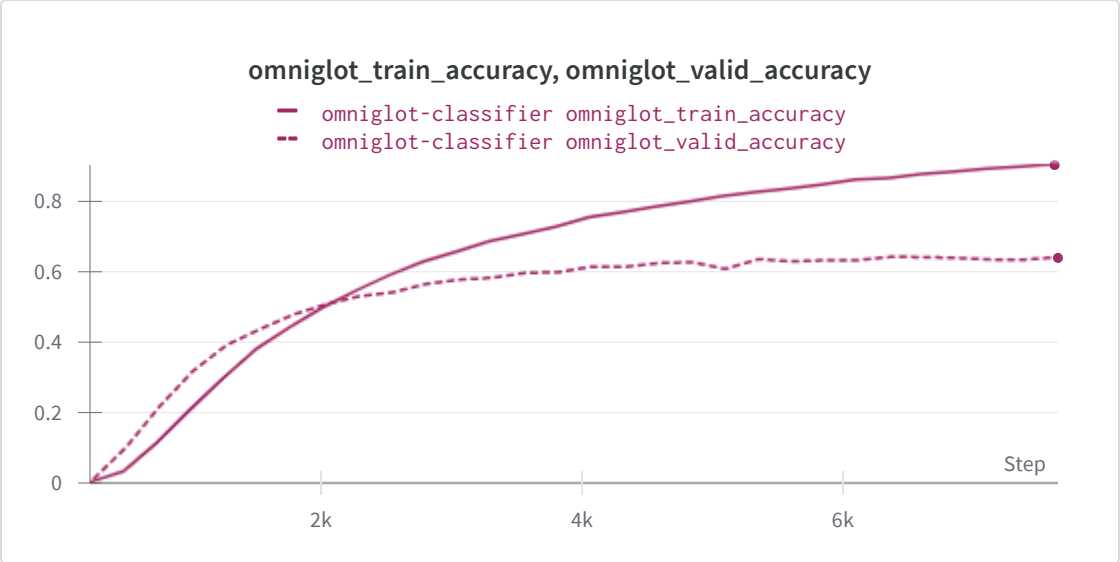
Omniglot classifier

- Код классификатора
- Код обучения

Особо заморачиваться насчет классификатора я тут не стал. Качество на тесте явно лучше чем у рандомного классификатора. Учитывая, что у нас всего 10 картинок на класс, а классов 1600+, то accuracy 63% на тесте выглядит вполне прилично. По графику видно, что модель уже начинает переобучаться, поэтому, для лучшего качества тут можно было бы рассмотреть как минимум scheduler для lr, ну и конечно же, stack more layers.







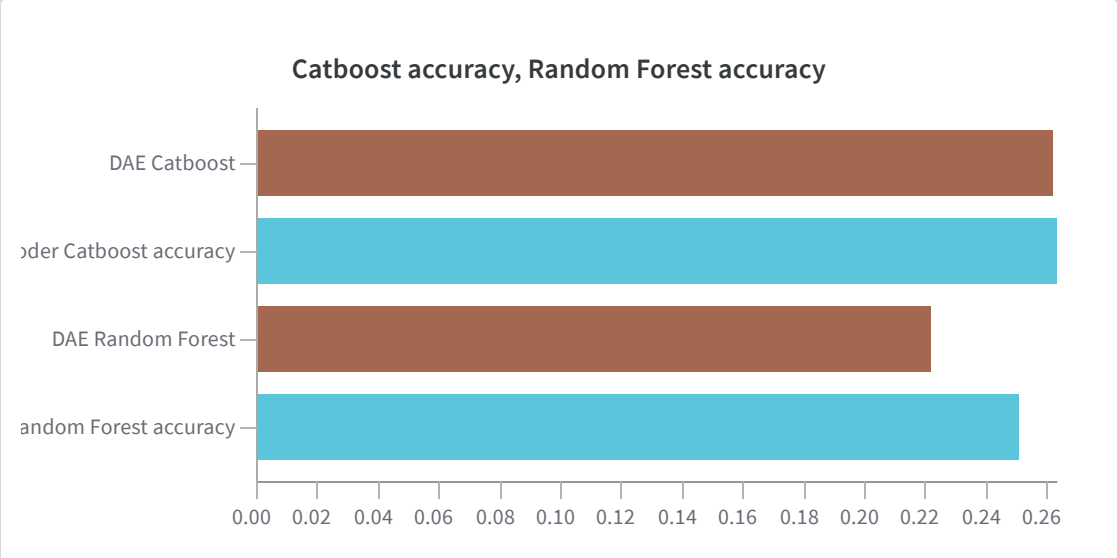
Обучение на латентных представлениях

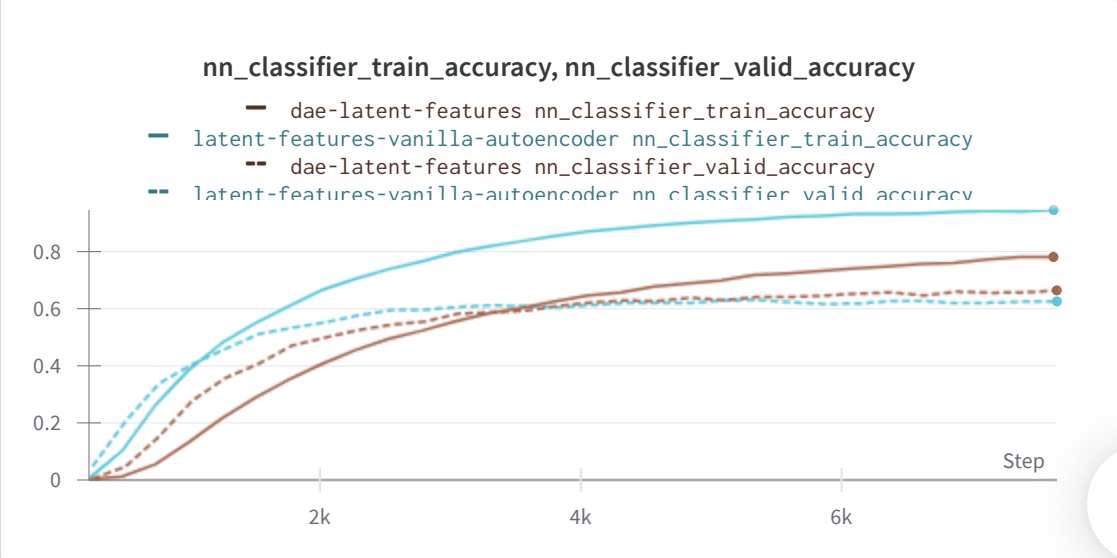
- Код обучения catboost и random forest
- Код нейросетевого классификатора
- Код обучения нейросетевого классификатора

Так как реконструкции оригинального изображения у Sparse AE выглядели слабо, я решил учить модели только на латентных представлениях Vanilla AE и DAE. Безусловно, интересно посмотреть на то, как справляются различные ансамбли деревьев: градиентный бустинг или случайные леса. К сожалению, учатся они долго, поэтому пришлось достаточно сильно ограничить некоторые параметры у этих

алгоритмов. Реализацию градиентного бустинга я взял от catboost, и только с max_depth=3 я смог учить его на GPU. На самом деле, с меньшими сетями, я учил его с `max_depth=4`, но по идее, в бустинге нам как раз нормально иметь неглубокие деревья. В random forest я поставил число деревьев 200 и максимальную глубину 30. В незалоггированных результатах с большим числом итераций в бустинге и без ограничения на глубину у меня получилось выбить порядка 0.3. Это конечно лучше, чем случайный классификатор, но явно недотягивает до результатов выше.

Повторить результаты выше удалось лишь с полносвязным нейросетевым классификатором. Как можно заметить, accuracy на тесте лучше у латентных представлений DAE, при этом, значение метрики на train заметно выше у Vanilla AE. То есть, эффект переобучения с латентными признаками DAE ниже.





Created with ❤️ on Weights & Biases.

https://wandb.ai/foksly/generative-models-homework-autoencoders/reports/-1---Vmlldzo0OTU5MTg