День 4: загрузка и сохранение моделей, SVHN датасэт, автоэнкодеры, глубокие сети.

1 Сохранение и загрузка натренированных моделей.

Сейчас мы попробуем сохранять и загружать модели, это бывает особенно ценно если тренировка длится многими часами или даже днями.
Прочитайте статью с официальной документацией по классу `tf.train.Saver`[1]. Достаточно для понимания прочитать до главы properties. Затем следует взять одно из решений к предыдущему дню из папки в дропбоксе, потренировать сеть 10000 итераций, сохранить результаты. Затем создать новую сессию (новый блок `with tf.Session() as sess: ...` и загрузить только что сохранённую модель.

2 Анализ результата

Тренировка нейронных сетей требует постоянного анализа получаемых результатов - довольно часто по специфике ошибок, которые допускает наша модель, можно догадаться какие параметры сети стоит изменить и как именно. Простейшим способом является подсчёт confusion matrix. Теперь когда мы научились сохраниять и загружать модели мы можем сравнивать разные модели между собой.

1. Натренируйте две baseline модели используя код из `day3/solutions/baseline_augm.py`.
   Модели отличаются тем, что одна будет аугментировать флиппая картинку(`np.fliplr()/np.flipud()`),
   а в другой флиппов нет - для этого вам нужно будет поправить код в `day3/solutions/simple_augmentat`
2. Посчитайте confusion matrix для обеих моделей.
3. Визуализируйте 10 случайных картинок из теста на которых сеть ошиблась. Проделайте это для обеих сетей.
4. Как вы объясните различия.

3 Усложнение датасэта. Street View House Numbers (SVHN)

Раньше мы научились варьировать параметры сети, теперь мы попробуем изменить сам датасэт, который мы используем. Мы будем использовать SVHN датасэт[2]. Обратите внимание, что данные представлены в двух различных форматах, нам потребуется второй формат, в котором цифры уже вырезаны.

[1] `https://www.tensorflow.org/api_docs/python/tf/train/Saver`
[2] `http://ufldl.stanford.edu/housenumbers/`

1. Скачайте датасэт и разберитесь как читать данные в формате .mat. Визуализируйте несколько картинок и выведите соответствующий им класс.

2. Снова начните с файла `day3/solutions/baseline_augm.py`.

3. Теперь нам недоступен класс mnist и нам нужно самостоятельно написать класс svhn, который буде иметь схожий функционал: иметь поля train, test, каждое из которых имеет поля image, labels. Также содержать метод для загрузки датасэта из файла и метод `next_batch()`.

4. Поменяйте некоторые параметры самой сети(самого графа) чтобы он принимал SVHN картинки.

5. Натренируйте сеть и выведите 10 случайных картинок из теста на которых сеть ошибается. В этом задании перформанс будет сильно хуже чем на MNIST датасэте. Чем вы это объясните?

4 Pretrained VGG16

Используем pretrained сеть VGG16 для классификации изображений.

1. Классификация. Разобраться с кодом в файле `vgg16_clsf.py`. Запустить на паре примеров.

2. Сегментация. Разобраться с кодом в файле `vgg16_segm.py`. Запустить на паре примеров.

3. * Сегментация со скользящим окном.

4. ** Повторить эксперимент используя другую сеть [3]. Веса доступны по ссылке `https://github.com/tensorflow/models/tree/master/research/slim`.

[3] `https://github.com/tensorflow/models/blob/master/research/inception/inception/slim/README.md`