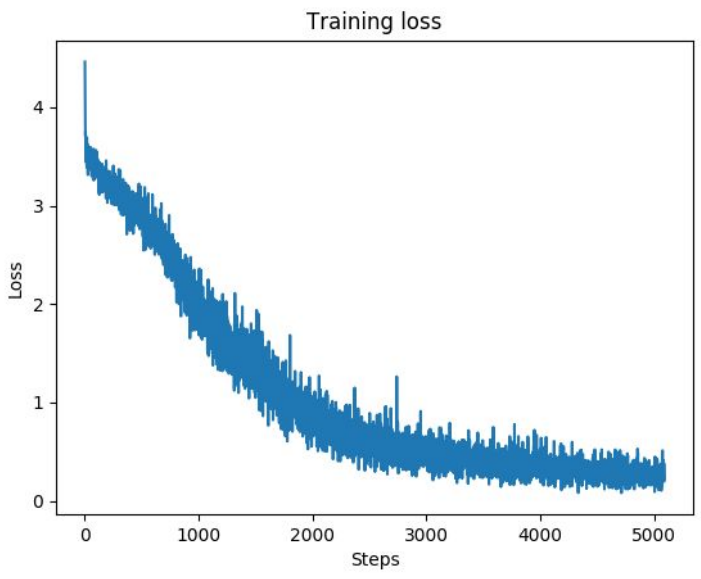
IMAGE DATA GENERATION and TRAINING:
1. Generated 50k images and trained the model but model was not predicting numbers and '@' (common in business cards) so generated more emails and numbers text and added to corpus. Discarded previous images.
2. Generated 50k images and trained the model which was failing to predict text which had all uppercase characters. Added a corpus with all uppercase words. Discarded previous images.

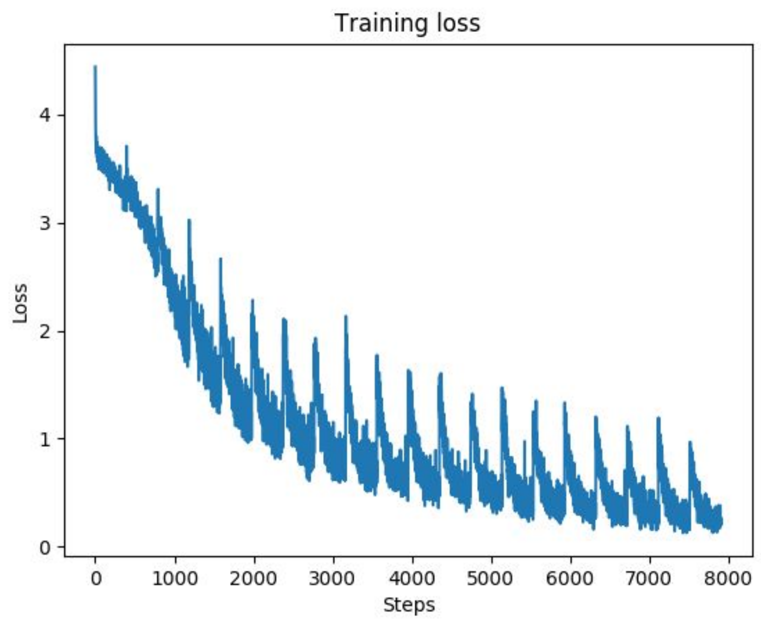
3.1 Generated 20000 one word and 5000 two word images using min:10 and max:22 font size and augmentations in text_renderer/configs/default.yaml.
3.2 Generated 20000 one word and 5000 two word images using min:5 and max:12 font size and augmentations in text_renderer/configs/default.yaml.
3.3 Trained a model on above data which was almost always predicting common words like 'and', 'his', etc for highly blurred data.
3. Generated 10000 one word images with length greater than 4 with 95% of them blurred. Trained model with combined data.
4. Concatenated canon data from public dataset in training data.

Converting images to tfrecords
1. Used utility_scripts/create_annotate_data.py to make annotation files suitable to be used by aocr.
2. Used <aocr dataset> command to generate tfrecords.

MODEL-DEFAULT:
1. Used default cnn of the library, embedding size 20 and 2 attention layers with 128 hidden units each.
2. Trained the model using batch size of 64 for 30 epochs.



MODEL-CUSTOM:
1. Added an extra convolutional layer with relu activation at the top of default cnn architecture. Used Embedding size 20 and 2 attention layers with 128 hidden units each.
2. Trained the model using batch size of 64 for 5 epochs



CHANGES TO text_renderer:
1. Added different fonts from font-squirrel.com (open-source) to generate images.
2. Added coloured background images and added more font colours in config/default.yaml
3. Added emails and numbers corpus, uppdercase and lowercase words corpus to data/corpus.

CHANGES TO attention-aocr/aocr:
1. Modified train method of Model class in model/model.py to save training loss at every step to a file and save the loss vs steps plot as png.
2. Modified test method of Model class in model/model.py to save predicted vs actual text for every incorrect preciction. Used it to see if there is a pattern in incorrect predictions.
3. Created CNN_cust class in model/cnn_custom_two.py containing custom cnn architecture.
4. Added variable self.custom_cnn in Model in model/model.py. If true use CNN_cust as cnn layer else use CNN (default).
5. Change CHARMAP of DataGen class in util/datagen.py to include larger alphabet.
6 Changed default values of FORCE_UPPERCASE=False (To allow lowercase too) and TARGET_VOCAB_SIZE = 84 (After counting size of input data alphabet using utility_scripts/get_alphabet.py)