Tacotron: towards end-to-end speech synthesis

date 6.04.2017, paper on arxiv (https://arxiv.org/pdf/1703.10135.pdf)

Model overview

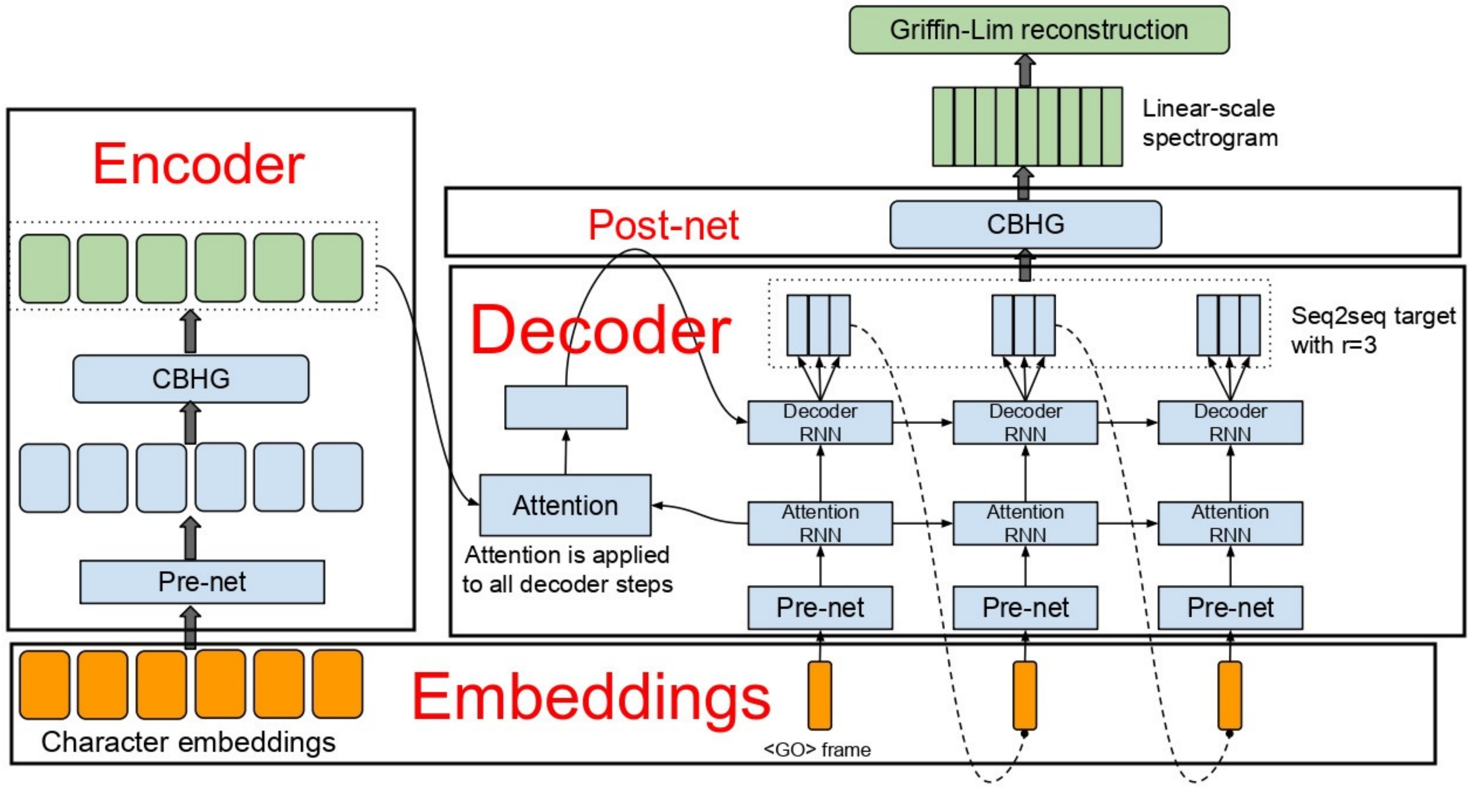
Since models like wavenets requires external models for prediction of linguistic features (grapheme to phoneme, fundamental frequency, phoneme duration) which are trained independently (and joint training reported to be complicated) these models are not end-to-end. In this model authors used as input - characters, and output - spectrogram, which is then converted to raw wavenet using Griffin-Lim algorithm.

Model architecture

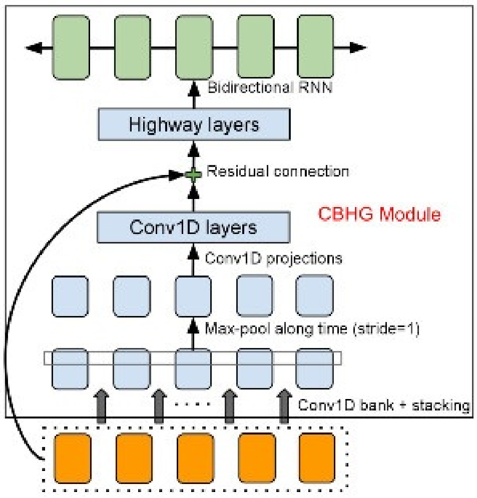
Model used several blocks (will be discussed later)

- CBHG module;
- Pre-net (2 stack linear layer with dropout);
- Attention RNN;
- Decoder RNN.
- Post-net

Overall architecure is shown below:



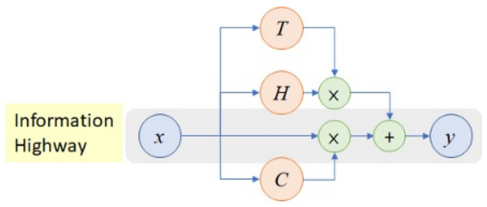
CBHG module



In CBHG module all convoltuion is done on time axis (e.g. using pytorch notation if you have batch size B in each batch size T characters with embedding size E after pre-net then input tensor to CBHG module has size (B, T, E) and in CBHG convolutions is done on dimesnsion T, it could be done via swapping E and T dimension and using nn.conv1d)

All convoluitons must preserve the time dimension (working dimension) and since stride=1, must use padding = kernel_size // 2.

After convoluton is done wone, working dimension (dimension for matrix multiplication in linear layers) becomes E. Highway network consists of 4 highway blocks. Highway block is shown below (T = linear layer with sigmoid activation, C equals to 1 - T, and H is linear layer with relu activation).



After applying highway network bidirectional GRU is applied.

Attention

It is not properly covered in article, so i looked into [code (https://github.com/r9y9/tacotron_pytorch/blob/master/tacotron_pytorch/attention.py)](https://github.com/r9y9/tacotron_pytorch/blob/master/tacotron_pytorch/attention.py). At each decoder timestep query q_t is calculated as output of decoder prenet. Attention is calculated between query $q_t \in R^{Batch \times 1 \times dim}$ and encoder outputs $O \in R^{Batch \times T \times dim}$ as following and used for calcutalion context vector as follwing:

$$attention_t = softmax(Linear(tanh(Linear(q, W) + O)), v) \in R^{Batch \times 1 \times T},$$

$$context\ vector_t = attention_t @ O \in R^{Batch \times 1 \times dim},$$

where @ is batch matrix-matrix multiplication (simple matrix matrix multiplication for every element in batch, check torch.bmm), $Linear(x, v) : R^{Batch \times T \times dim} -> R^{Batch \times 1 \times T}$ - linear transformation (dim -> 1) with unsqueezing dimension and $Linear(x, w) : R^{Batch \times 1 \times dim} -> R^{Batch \times 1 \times dim}$ - linear transformation.

Attention RNN

Attention RNN does pass input through RNN cell and uses its output to calculate context vector using attention. At each timestep context vector and the RNN cell output are concateneted to form the input to the decoder RNN.

Decoder RNN

At each timestep decoder rnn outputs several non-overlapping mel spectorgrams (vectors). Predicting r frames at once divides the total number of decoder steps by r, which reduces model size, training time and inference time. Authors also reported this trick to substantially increase convergence speed, as measured by a much faster (and more stable) alignment learned from attention.

Post-Net

Since model uses Griffin-Lim algotirhm for synthesis raw waveforms, mel spectrograms must be converted to linear spectograms. Post-net takes all generated mel spectrograms and passes it through another CBHG module to predict linear spectograms.

Training

During training groundtruth mel-spectogram of previous frame used as input to decoder (for the first frame 0 spectrogram is used). L1 loss was used for both linear and mel spectrograms. Authors also report that learning model to predict zero spectrogram after stop token is essential for model to learn where to stop generating.

Testing

During training previous generated mel spectogram used as input to decoder (for the first frame 0 spectrogram is used). After sequantial generating of all mel-spectograms, they are used as input to post-net to generate linear spectrograms, which are used as input to Griffin-Lim algorithm to generate raw waveforms. This model doesn't have time drawback of the wavenet even using sequential generating since spectrograms are higly compressed in time domain comparing to raw waveforms.

Experiments

Iternal North American English dataset, which contains about 24.6 hours of speech data spoken by a professional female speaker, was used for training.Mean opinion score was used for evaluating. Model performed better than parametiric model using LSTM, but worse than concatenating appoach. There were no comparance in the articale with wavenet model, but wavenet model was reported (in wavenet paper) to outperformm concatenative appoach.

Pros and cons

The main pros of the model is being really end-to-end (excluding Griffin-Lim), models doesn't use complicated linguistic features as wavenet model. Original Wavenet model used external (unpublished!) methods to predict linguistic features, and Deep Voice version of wavenet used a lot of DL models for predicting these features, however all these models are trained separately and tuning model of other language is not trivial and requires much time and expertise. The sequential generation is not a drawback of Tacotron (unlike wavenet) due high compression of spectograms in time domain comparing to raw waveforms. However, the quality of generated samples are significanlty lower comparing to wavenet generation (Listened to several samples with same text). Tacotron samples tends to have metallic pitchs and sometimes skips words in long sentences.