Professor Forcing: A New Algorithm for Training Recurrent Networks

Presenter: Shijia Wang

Alex Lamb[1] Anirudh Goyal[1] Ying Zhang[1] Saizheng Zhang [1] Aaron Courville [1] Yoshua Bengio [1]

[1]MILA
Universite de Montreal

29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain

1. Introduction
   - Recurrent Neural Network
   - Contributions
2. Professor Forcing Model
   - Model Diagram
   - Definitions and Notation
   - Objectives
3. Conclusion
   - Experiments

Outline

1. Introduction
   - Recurrent Neural Network
   - Contributions
2. Professor Forcing Model
   - Model Diagram
   - Definitions and Notation
   - Objectives
3. Conclusion
   - Experiments

Definition

- RNNs have become the generative models of choice for sequential data.
- Decomposes distribution over the discrete time sequence $y_1, y_2, ..., y_T$
- $P(y_1, y_2, ...y_T) = P(y_1) \prod_{t=1}^{T} P(t_t | y_1, ..., y_{t-1})$

Problem

- So far RNNs show poor generating performance especially for longer sequences
- The training behavior and generating behavior do not match
- A slight error in the probability can compound into large differences

Past Solutions

- Teacher Forcing: feed ground-truths back into model to keep model close to ground-truth sequences
  Problem: small prediction errors away from ground-truth values lead to behavior divergence
  Williams and Zipser, 1989
- Scheduled Sampling Method: mix ground-truth and generated sequences for training
  Problem: With more generated sequences the model is not clear that the correct target is the ground-truth
  Bengio et al., 2015
- Also found that Scheduled Sampling Method yields a biased estimator
  Huszr 2015

Outline

1. Introduction
   - Recurrent Neural Network
   - Contributions
2. Professor Forcing Model
   - Model Diagram
   - Definitions and Notation
   - Objectives
3. Conclusion
   - Experiments

Contributions

- Professor Forcing method to train RNNs. Use 2 networks
- Improves long sequence sampling for RNNs
- Acts as a regularizer for RNNs
- Mixes the hidden states when doing sampling and teacher forcing
- Better generates samples longer than those used during training

Overview

- Uses 2 neural networks:
  - Generator RNN: switches between Teacher Forcing and Free Running modes and pass behavior values
  - Discriminator: determines if the passed hidden states comes from teacher forcing or free running modes

Outline

1. Introduction
   - Recurrent Neural Network
   - Contributions
2. Professor Forcing Model
   - Model Diagram
   - Definitions and Notation
   - Objectives
3. Conclusion
   - Experiments

Model

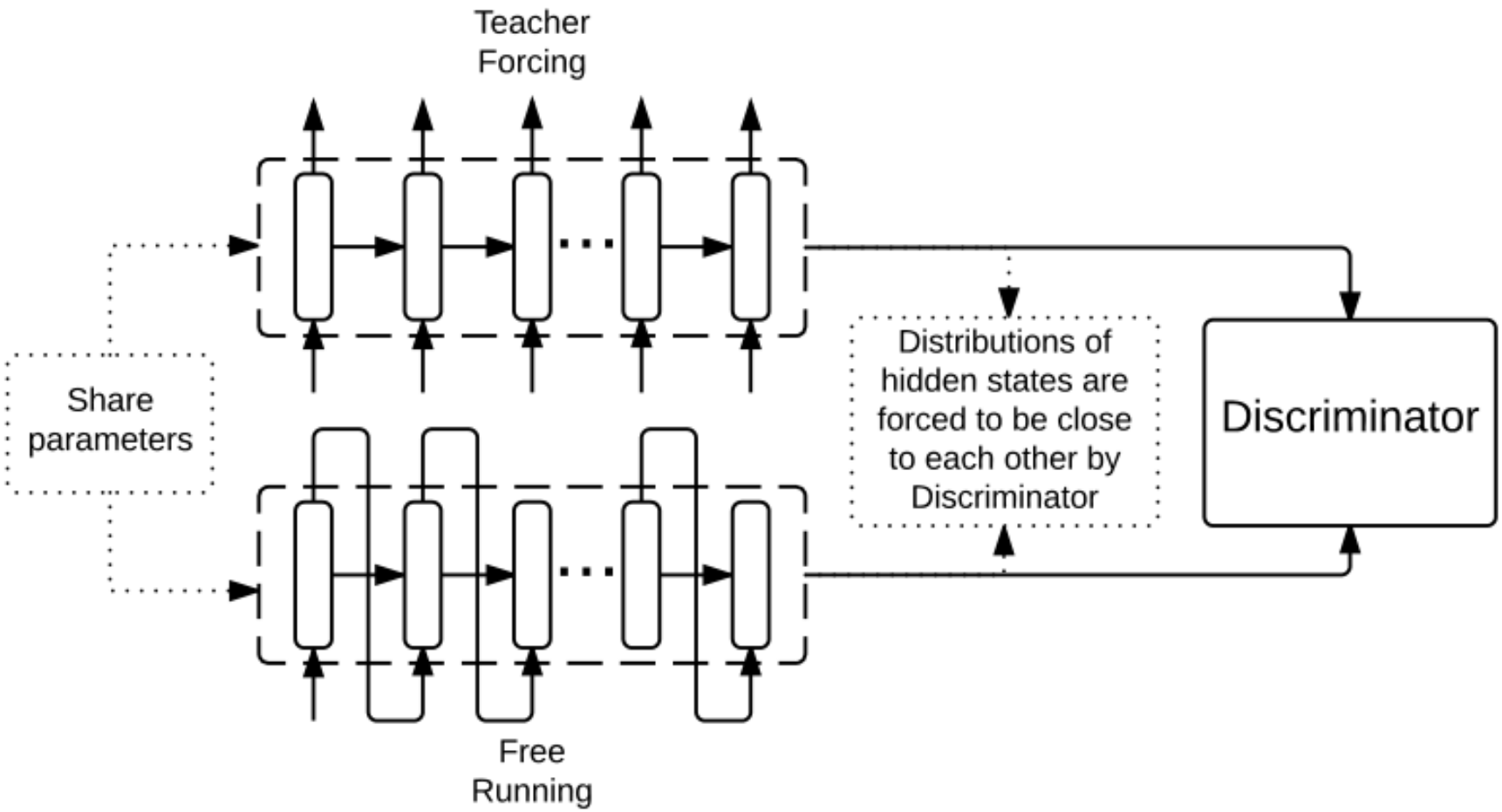


Figure 1: Architecture of the Professor Forcing - Learn correct one-step predictions such as to to obtain the same kind of recurrent neural network dynamics whether in open loop (teacher forcing) mode or in closed loop (generative) mode. An open loop generator that does one-step-ahead prediction correctly. Recursively composing these outputs does multi-step prediction (closed-loop) and can

Generative RNN

- the generative RNN has a single hidden layer of gated recurrent units (GRU), chosen because they are cheaper computationally than LSTM
- behavior function B outputs the pre-tanh activation of the GRU states, and optionally the softmax outputs for the next-step prediction

Discriminator

- discriminator is based on either a bidirectional RRN with GRUs or a CNN
- hidden states concatenate and fed to a 3-layer neural network
- each layer has an affine transformation and a ReLU
- output layer has an affine transformation and a sigmoid

Outline

1. Introduction
   - Recurrent Neural Network
   - Contributions
2. Professor Forcing Model
   - Model Diagram
   - Definitions and Notation
   - Objectives
3. Conclusion
   - Experiments

Definitions

- (x, y) input and
  - For teach forcing, y is the ground-truth from a training sequence
  - For free running, y is self-generated by the generator in respect to $P_{\theta_g}(y|x)$
- θ_g parameters of the generative RNN
- θ_d parameters of the discriminator
- $B(x, y, \theta_g)$ outputs the behavior sequence b (chosen hidden states and output values) given data
- $D(b)$ the output of the discriminator which estimates of probability that b was generated in teach forcing mode

Outline

1. Introduction
   - Recurrent Neural Network
   - Contributions
2. Professor Forcing Model
   - Model Diagram
   - Definitions and Notation
   - Objectives
3. Conclusion
   - Experiments

Discriminator

Maximize the likelihood of correctly classifying a behavior sequence.

$$C_d(\theta_d|\theta_g) = E_{(x,y)\sim data}[-logD(B(x,y,\theta_g),\theta_d) + E_{y\sim P_{\theta_g}(y|x)}[-log(1-D(B(x,y,\theta_g),\theta_d))] \quad (1)$$

Generator RNN

Maximize the likelihood of the data using a negative log-likelihood objective.

$$NLL(\theta_g) = E_{(x,y)\sim data}[-logP_{\theta_g}(y|x)] \tag{2}$$

Generator RNN

Fool the discriminator by changing the free running behavior so it matches with the teacher forced behavior.

$$C_f(\theta_g|\theta_d) = E_{x\sim data, y\sim P_{\theta_g}(y|x)}[-logD(B(x,y,\theta_g),\theta_d)] \quad (3)$$

Or additionally ask the teacher forced behavior to be indistinguishable from the free running behavior

$$C_t(\theta_g|\theta_d) = E_{(x,y)\sim data}[-log(1 - D(B(x,y,\theta_g),\theta_d))] \quad (4)$$

Update

- Perform SGD steps on $NLL + C_f$ or on $NLL + C_f + C_t$ to update the generative RNN
- Perform SGD on C_d to update the discriminator

Outline

1. Introduction
   - Recurrent Neural Network
   - Contributions
2. Professor Forcing Model
   - Model Diagram
   - Definitions and Notation
   - Objectives
3. Conclusion
   - Experiments

Character-Level Language Modeling

- character-level language modeling on Penn-Treebank corpus
- evaluated by bits-per-character (BPC)
- cost of Professor Forcing decreases faster compared to Teacher Forcing
- training time is 3 times slower than Teacher Forcing
- BPC on validation set using baseline is 1.50 while using PF is 1.48

Character-Level Language Modeling

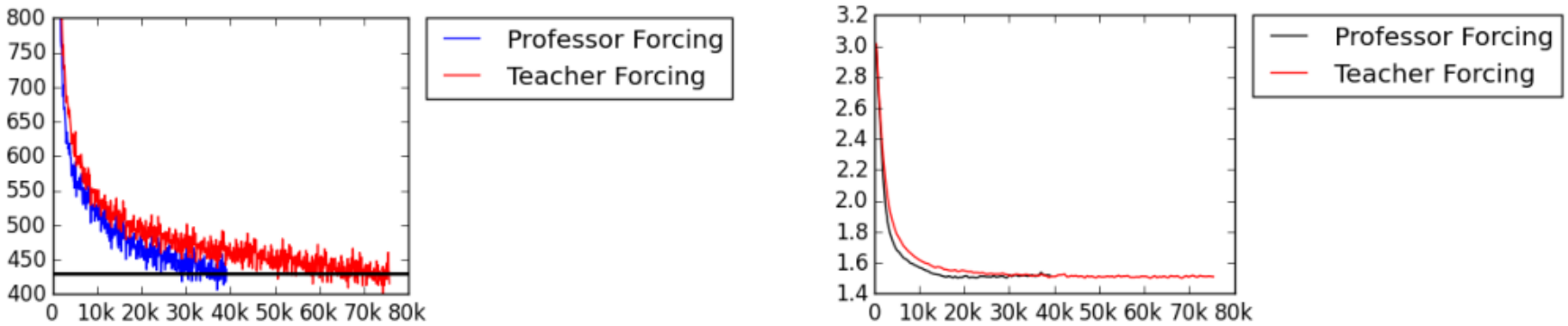


Figure 2: Penn Treebank Likelihood Curves in terms of the number of iterations. Training Negative Log-Likelihood (left). Validation BPC (Right)

Sequential MNIST

- sequentially generating the pixels in MNIST digits
- standard binarized MNIST dataset
- used convolutional network for the discriminator instead of bi-directional RNN

Sequential MNIST

Method	MNIST NLL
DBN 2hl (Germain *et al.*, 2015)	≈ 84.55
NADE (Larochelle and Murray, 2011)	88.33
EoNADE-5 2hl (Raiko *et al.*, 2014)	84.68
DLGM 8 leapfrog steps (Salimans *et al.*, 2014)	≈ 85.51
DARN 1hl (Gregor *et al.*, 2015)	≈ 84.13
DRAW (Gregor *et al.*, 2015)	≤ 80.97
Pixel RNN (van den Oord *et al.*, 2016)	**79.2**
Professor Forcing (ours)	79.58

Table 1: Test set negative log-likelihood evaluations

Sequential MNIST

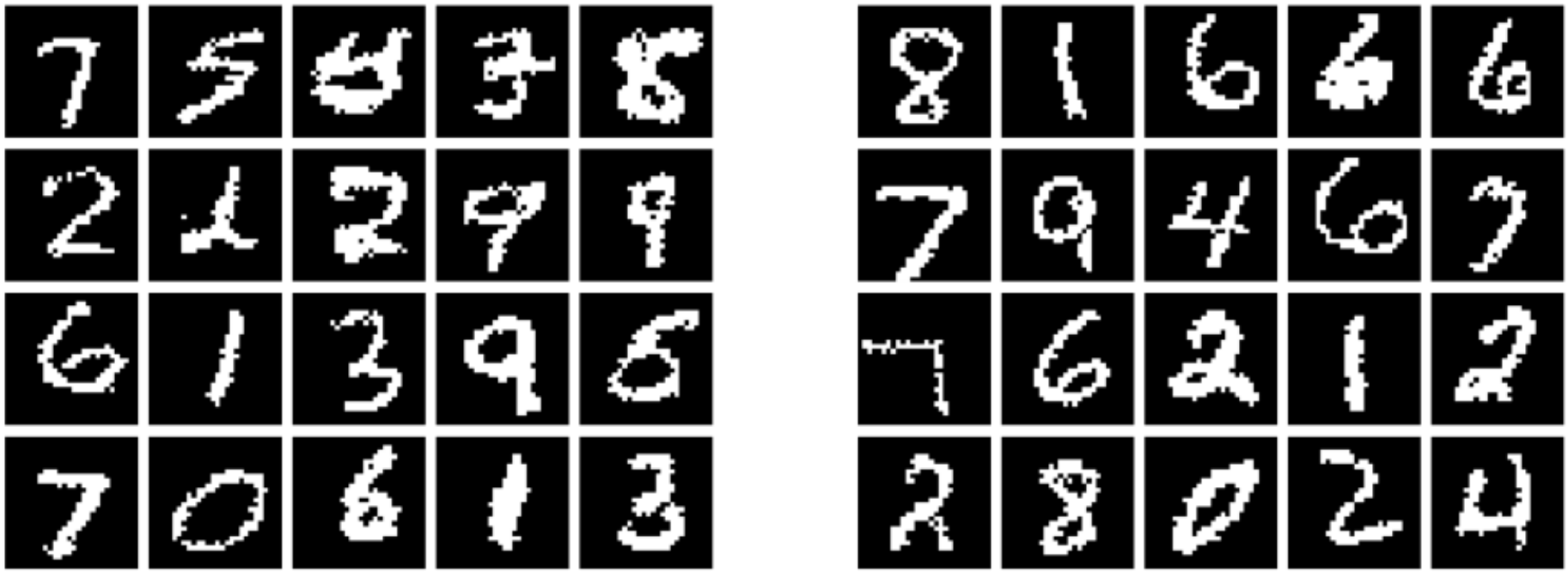


Figure 4: Samples with Teacher Forcing (left) and Professor Forcing (right).

T-SNE Visualization

t-distributed stochastic neighbor embedding

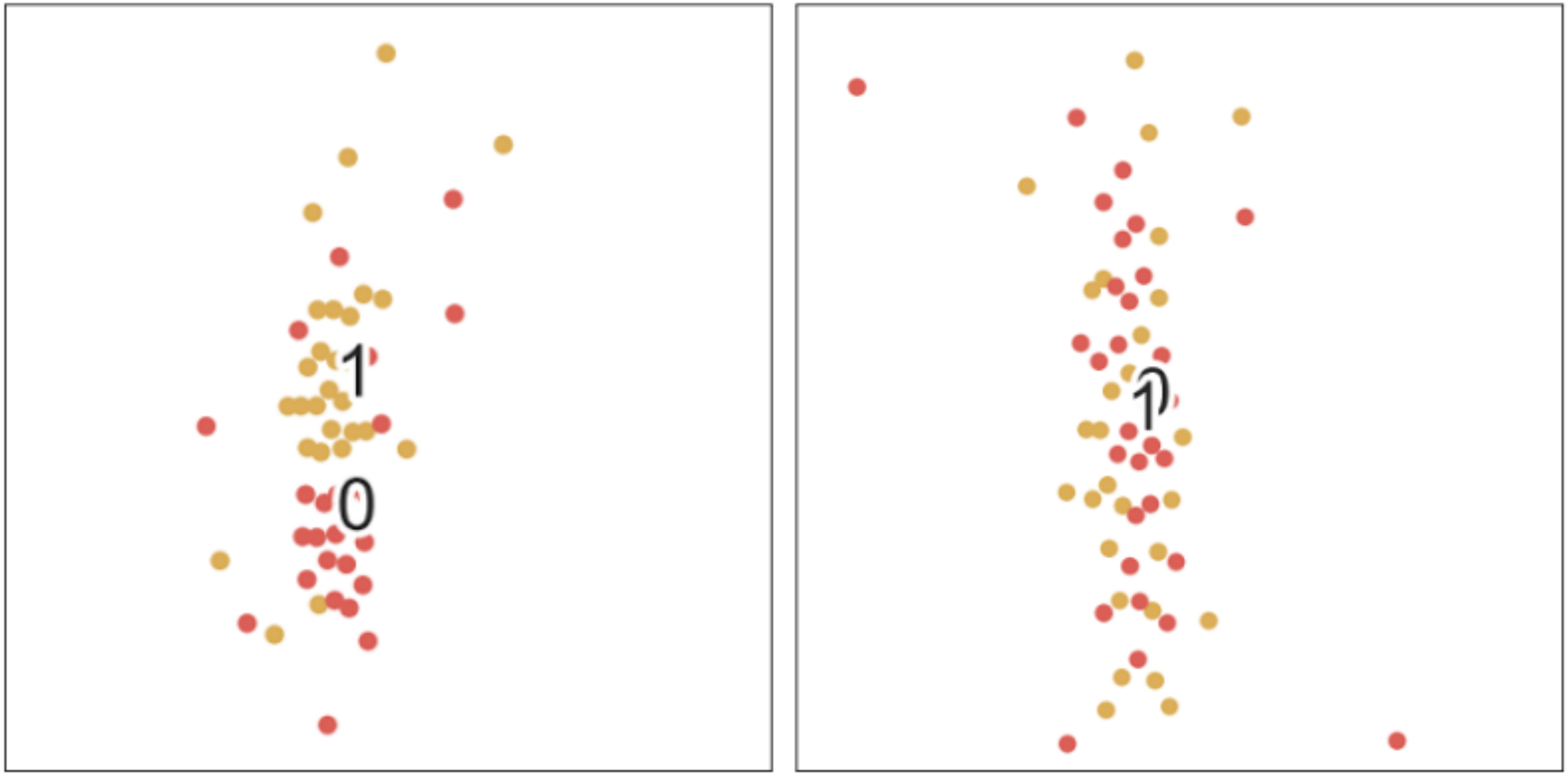


Figure 3: T-SNE visualization of hidden states, left: with teacher forcing, right: with professor forcing. Red dots correspond to teacher forcing hidden states, while the gold dots correspond to free running mode. At t = 500, the closed-loop and open-loop hidden states clearly occupy distinct regions with teacher forcing, meaning that the network enters a hidden state region during sampling distinct from the region seen during teacher forcing training. With professor forcing, these regions

Handwriting Generation

- test if Professor Forcing could to sampling much longer sequences than those use during trainining
- train on only 50 steps of text-conditioned handwriting which correlates to only a few letters
- sample for 1000 time steps
- use IAM-OnDB dataset
- performed human evaluation for quality

Handwriting Generation

Response	Percent	Count
Professor Forcing Much Better	19.7	151
Professor Forcing Slightly Better	57.2	439
Teacher Forcing Slightly Better	18.9	145
Teacher Forcing Much Better	4.3	33
Total	100.0	768

Table 2: Human Evaluation Study Results for Handwriting

Handwriting Generation

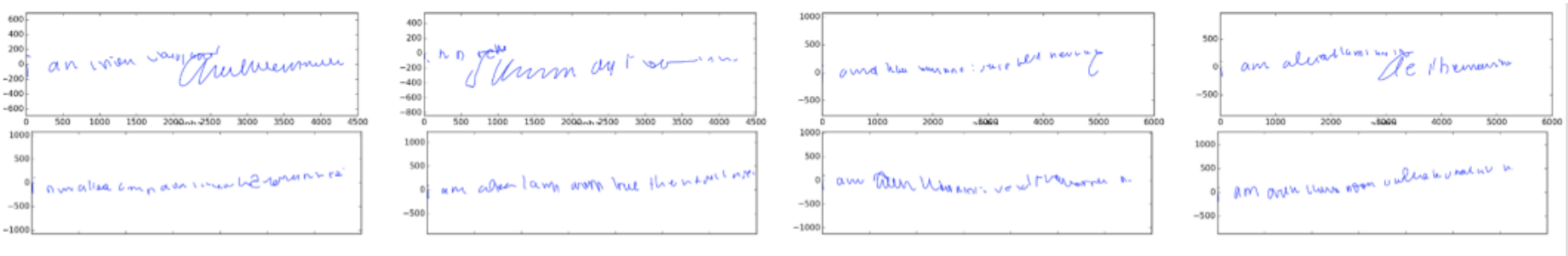
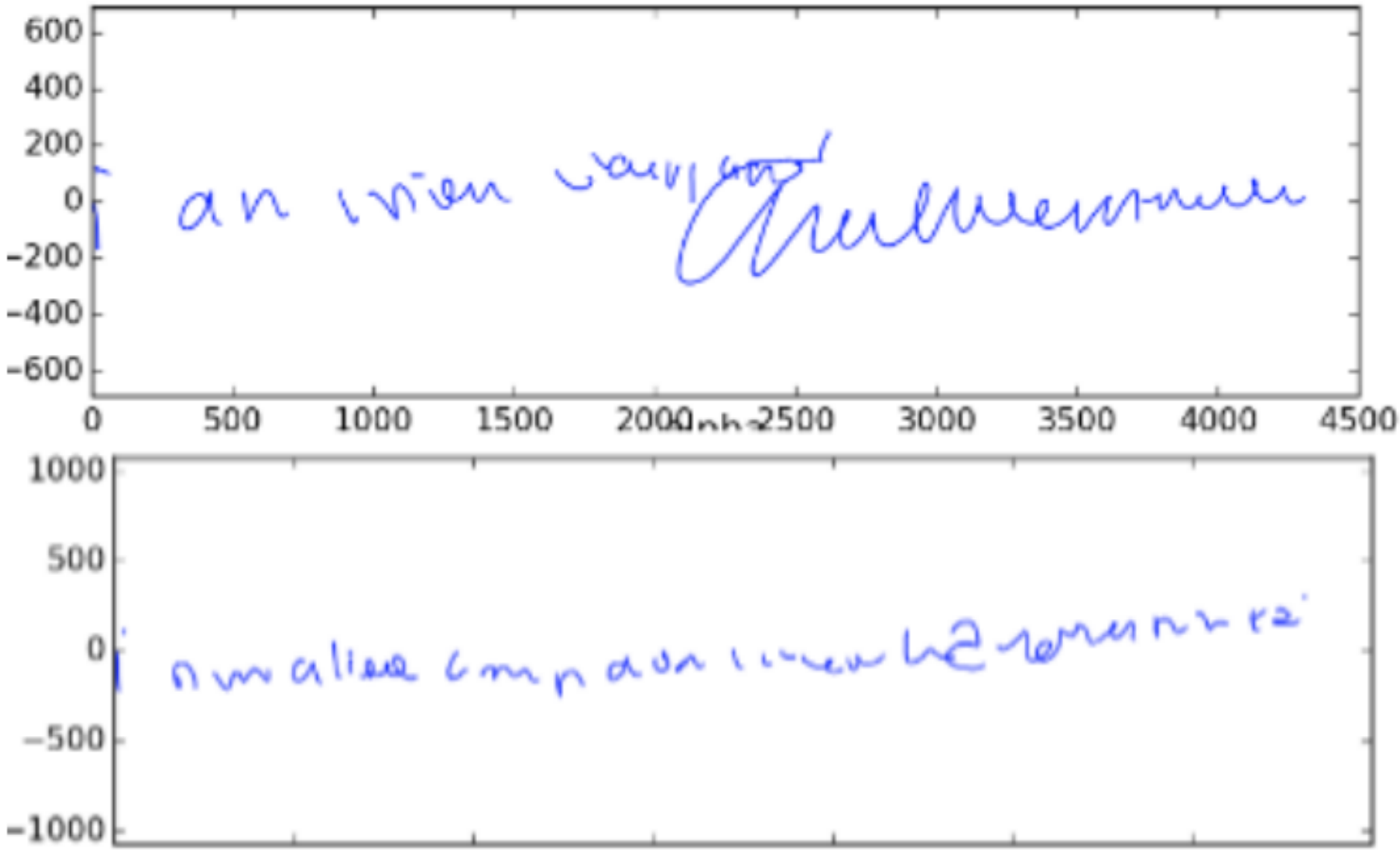


Figure 5: Handwriting samples with teacher forcing (top) and professor forcing (bottom). Note that in both cases the model is only trained on observed sequences of length 50 steps (a few letters) but is used to do conditional generation for 1000 steps.

Handwriting Generation



Music Synthesis on Raw Waveforms

- vocal synthesis on raw waveforms
- used 3 hours of monk chanting audio from YouTube

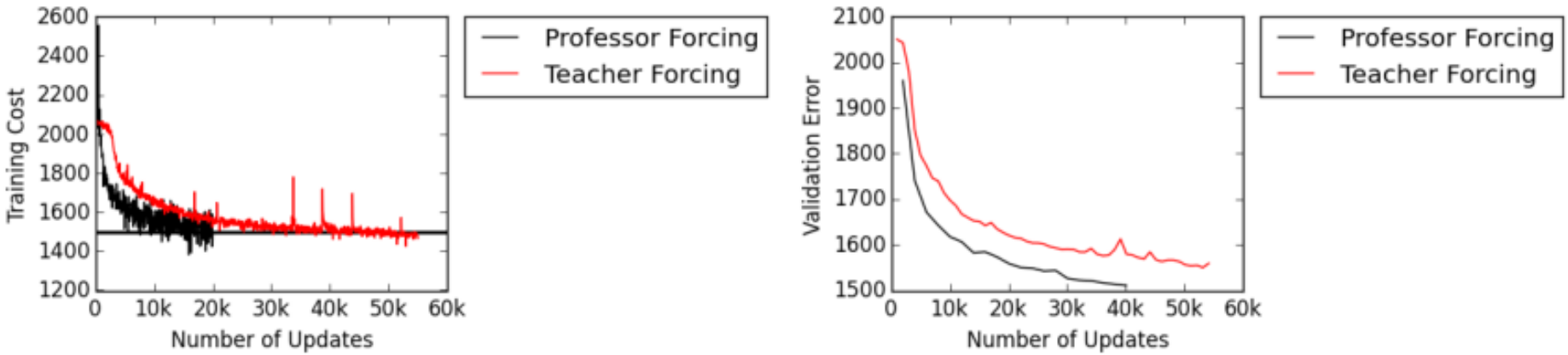


Figure 6: Left: training likelihood curves. Right: validation likelihood curves.

Music Synthesis on Raw Waveforms

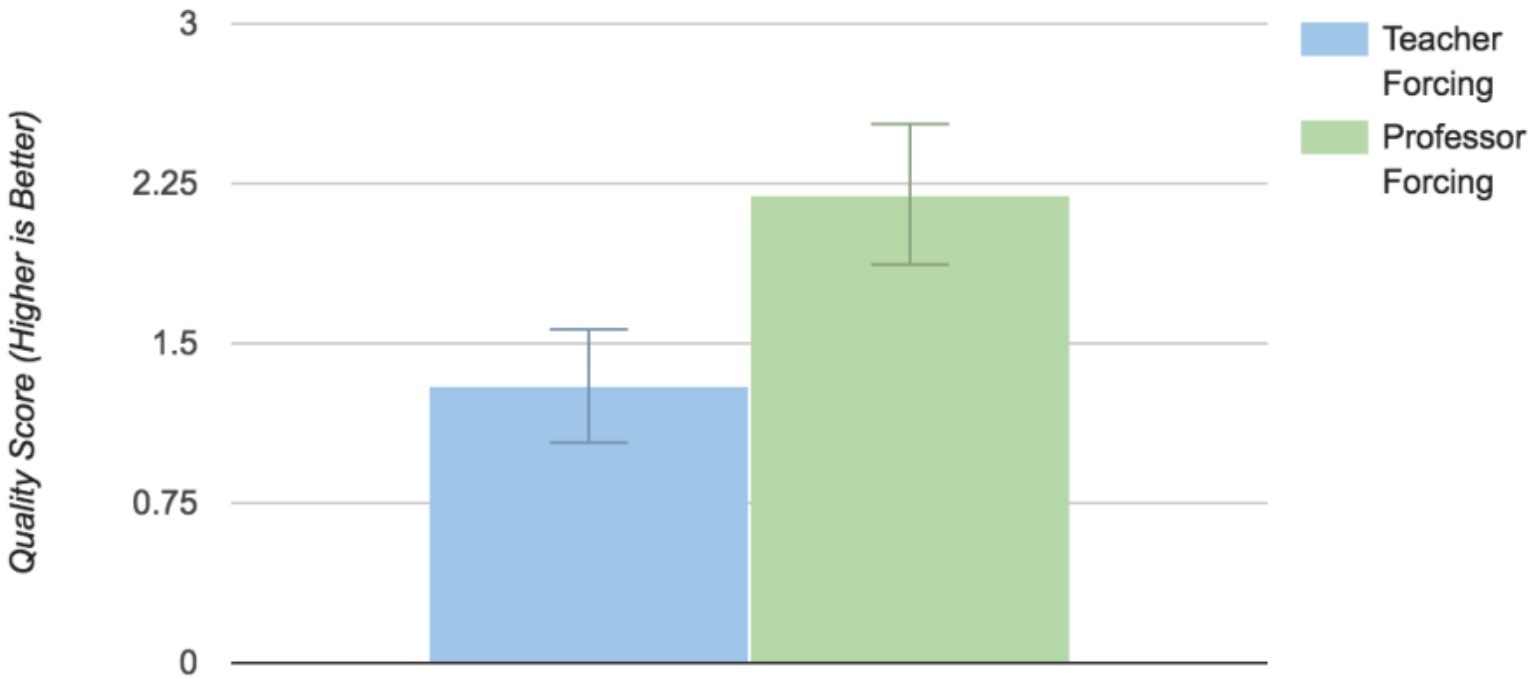


Figure 7: Human evaluator ratings for vocal synthesis samples (higher is better). The height of the bar is the mean of the ratings and the error bar shows the spread of one standard deviation.

Conclusion

- discriminator spot the differences in behavior of these 2 modes
- discriminator look at the statistics of the behavior and not just at the single-step predictions
- forces generator to behave the same when constrained by data and when generating outputs
- better generalization over sequences that are much longer than the training sequences
- generalize better in log-likelihood suggests it acts like a regularizer