

000
001
002
003
004
005
006
007
008
009
010
011
012
013
014
015
016
017
018
019
020
021
022
023
024
025
026
027
028
029
030
031
032
033
034
035
036
037
038
039
040
041
042
043
044
045
046
047
048
049
050
051
052
053

THE COOPER UNION
ALBERT NERKEN SCHOOL OF ENGINEERING

A Deep Partitioned Autoencoder
for De-Noising Live Audio

by
Ethan Lusterman

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Engineering

September 2016

Professor Sam Keene, Advisor

054 THE COOPER UNION FOR THE
055
056 ADVANCEMENT OF SCIENCE AND ART
057
058
059
060

061 ALBERT NERKEN SCHOOL OF ENGINEERING
062
063
064
065
066
067
068
069

070 This thesis was prepared under the direction of the Can-
071 didate's Thesis Advisor and has received approval. It was
072 submitted to the Dean of the School of Engineering and
073 the full Faculty, and was approved as partial fulfillment of
074 the requirements for the degree of Master of Engineering.
075
076
077
078
079
080
081
082
083
084
085
086

087 _____
088 Dean, School of Engineering Date
089
090
091
092
093
094

095 _____
096 Prof. Sam Keene, Thesis Advisor Date
097
098
099
100
101
102
103
104
105
106
107

108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161

Acknowledgements

Ack example

162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215

Abstract

Abstract

216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269

Contents

1	Introduction	1
2	Background	2
2.1	Machine Learning	2
2.1.1	Regression	3
2.1.2	Overfitting and Curse of Dimensionality	3
2.1.3	Loss functions and Regularization	3
2.1.4	Gradient Stuff?	3
2.2	Neural Networks	3
2.2.1	Dense Layer	3
2.2.2	Convolutional Layer	3
2.2.3	Nonlinearity Choice	3
2.3	Signals and Systems	3
2.3.1	Signals	3
2.3.2	Convolution	4
2.3.3	Frequency Transforms	5
2.3.4	Windowing and Perfect Reconstruction	6
2.3.5	Window Size and Frequency v. Time Resolution Tradeoff	7
2.3.6	Noise and Signal-to-Noise Ratio	7
2.3.7	Magnitude and Phase Spectrum	7
3	Signal Model and Data	8
3.1	Network Input and Output	8
3.2	Signal and Noise Choices	9
3.3	Other Network Parameters	10
4	De-noising Architectures	11
4.1	Classic Supervised Autoencoder	11
4.2	Paris Autoencoder	11
4.3	Partitioned Autoencoder	11

270	4.3.1	Phase Reconstruction	11
271			
272	4.4	Curro Autoencoder	11
273			
274			
275	5	Results	12
276	5.1	Supervised Autoencoder	12
277			
278	5.1.1	Batch Normalized Input	12
279			
280	5.1.2	Non-Batch Normalized Input	12
281			
282	5.2	Partitioned Autoencoder	12
283			
284	5.3	Partitioned Curro Autoencoder	12
285			
286	5.4	Comparison of Loss Convergence	12
287			
288	5.5	Comparison of Mean Squared Error Convergence	12
289			
290	6	Conclusions and Future Work	13
291	6.1	Conclusions	13
292			
293	6.2	Future Work	14
294			
295	6.2.1	Models	14
296			
297	6.2.2	Data	15
298			
299			
300			
301			
302			
303			
304			
305			
306			
307			
308			
309			
310			
311			
312			
313			
314			
315			
316			
317			
318			
319			
320			
321			
322			
323			

324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377

List of Figures

1	Loss at various SNRs for Supervised Single-Layer Autoencoder with Batch Normalization at the Input	12
2	MSE at various SNRs for Supervised Single-Layer Autoencoder with Batch Normalization at the Input	13
3	Loss at various SNRs for Supervised Single-Layer Autoencoder without Batch Normalization at the Input	13
4	MSE at various SNRs for Supervised Single-Layer Autoencoder with Batch Normalization at the Input	14
5	Loss at various SNRs for Single-Layer Partitioned Autoencoder [1]	14
6	MSE at various SNRs for Single-Layer Partitioned Autoencoder [1]	15
7	Loss at various SNRs for Single-Layer Curro Autoencoder . .	15
8	MSE at various SNRs for Single-Layer Curro Autoencoder . .	16
9	Loss Comparison of Various Networks at -6 dB	16
10	Loss Comparison of Various Networks at -3 dB	17
11	Loss Comparison of Various Networks at 0 dB	17
12	Loss Comparison of Various Networks at 3 dB	18
13	Loss Comparison of Various Networks at 6 dB	18
14	MSE Comparison of Networks at -6 dB	19
15	MSE Comparison of Networks at -3 dB	19
16	MSE Comparison of Networks at 0 dB	20
17	MSE Comparison of Networks at 3 dB	20
18	MSE Comparison of Networks at 6 dB	21

378	Table of Nomenclature
379	
380	
381	
382	
383	
384	
385	
386	
387	
388	
389	
390	
391	
392	
393	
394	
395	
396	
397	
398	
399	
400	
401	
402	
403	
404	
405	
406	
407	
408	
409	
410	
411	
412	
413	
414	
415	
416	
417	
418	
419	
420	
421	
422	
423	
424	
425	
426	
427	
428	
429	
430	
431	

Introduction

Advances in smartphone technology have led to smaller devices with more powerful audio hardware, allowing for common consumers to make higher quality recordings. However, recorded speech and music are subject to noisy conditions, often hampering intelligibility and listenability. The goal of denoising audio recordings is to improve intelligibility and perceived quality. A variety of applications of audio denoising exist, including listening to a recording of a band or an artist’s live performance in a noisy crowd, or listening to a recorded conversation or speech under noisy conditions.

A common technique for denoising involves the use of deep neural networks (DNN). [PARIS] Advances in parallel graphics processing units (GPU) and in machine learning algorithms have allowed for training deeper networks faster, utilizing more hidden layers with more neurons.

Prior work in denoising audio has involved access to noise-free training data. Since common consumers do not often have access to clean audio, we seek to denoise without the use of clean audio.

In this thesis, we compare several neural network architectures and problem scenarios, ranging from data input types, level of noise, depth of network, training objectives, and more. In Chapter 2, we present background information on machine learning, neural networks, and signal processing as well as prior work in audio denoising. In Chapter 3, we detail all considered network architectures. In Chapter 4, we compare results from different data inputs, levels of noise, network architectures, and training objectives and discuss methods of evaluation. Finally, we make conclusions and recommendations for future work in Chapter 5.

Background

Machine Learning

Machine learning involves the use of computer algorithms to make decisions based on training data. Generally, this falls into categorizing input data (classification) or determining a mathematical function to determine a continuous output given an input (regression). Popular classification examples include recognizing handwritten digits (MNIST) as well as determining whether an image contains a cat or a dog. (REF) An example of a regression problem is determining the temperature given a set of input features (humidity, latitude, longitude, date, etc.).

Problems where training data contain input data vectors as well as the correct output vectors (targets) are known as supervised learning problems. Training a model to denoise audio where noise was introduced to the clean audio would be a supervised learning problem. On the other hand, training a model to denoise audio where the underlying clean signal is not known is an unsupervised learning problem. Different loss functions and neural network architectures can be exploited to accomplish denoising without the clean data.

For the purposes of this thesis, we use machine learning to determine an underlying nonlinear function that removes noise from time slices of audio (i.e. regression). These slices can then be pieced back together through overlap-add resynthesis. To clarify, this is a general linear model that maps an input noisy audio vector $y[n] = x[n] + N[n]$ to $\tilde{x}[n]$, a target denoised audio vector, where $x[n]$ is the underlying clean signal and $N[n]$ is the additive background noise.

Regression

A classical regression technique is linear regression, where one or more independent variables x_i are used to determine a scalar dependent variable y . The case of a single independent variable x is known as simple linear regression. On the other hand, the case of estimating A canonical example would be estimating a sine wave $x[n]$ given noisy samples

Overfitting and Curse of Dimensionality

Loss functions and Regularization

Gradient Stuff?

Neural Networks

Dense Layer

Convolutional Layer

Nonlinearity Choice

Signals and Systems

Domain knowledge of discrete audio signals and systems better informs our decisions for an audio denoising system, so some background information on signals and systems as it pertains to this thesis is detailed below.

Signals

We deal exclusively with discrete-time audio signals in this thesis. A discrete-time audio signal $x[n]$ is represented as a sequence of numbers (samples), where each integer-valued slot n in the sequence corresponds to a unit of time based on the sampling frequency f_s . This comes from sampling the continuous-time audio signal $x_c(t)$:

$$x[n] = x_c(nT) \quad (1)$$

where $T = 1/f_s$. For example, a 1-second speech signal sampled at 8kHz has 8000 samples. Furthermore, digital signals also have discrete valued sample amplitudes. For the purposes of this thesis, the bit depths of computers we use for analysis are high enough to allow for perfect reconstruction between continuous-time signals and digital signals.

We also assume signals collected have been properly sampled according to the Nyquist-Shannon sampling theorem, which states that a discrete-time signal must be sampled at at least twice the highest frequency present in the signal to prevent aliasing of different frequencies. For example, speech signals generally have information up to 8kHz, so many speech signals are sampled at 16kHz. Music is more complex in that signals often span up to about 20kHz, so CD quality recordings are often sampled at 44.1kHz or higher. For this thesis, we use recordings sampled at 44.1kHz or lower.

Convolution

The discrete-time convolution operation takes two sequences $x[n]$ and $h[n]$ and outputs a third sequence $y[n] = x[n] * h[n]$:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] \quad (2)$$

Convolution is commutative, so $x[n] * h[n] = h[n] * x[n]$ holds true.

A linear, time-invariant (LTI) system is characterized by its impulse response $h[n]$, which allows us to determine samples $y[n]$ when $x[n]$ is subject to $h[n]$. For the purposes of this thesis, our underlying clean signal $x[n]$ might be

subject to the conditions of an acoustic environment $h[n]$ and crowd noise $N[n]$:

$$y[n] = h[n] * x[n] + N[n] \quad (3)$$

In this scenario, our system would attempt to recover $h[n] * x[n]$ and possibly even $x[n]$ if the acoustic environment were deemed “noisy enough” due to echo and reverberation.

One of our proposed systems also incorporates convolutional neural networks (CNN) which use convolutions between frames of samples instead of simple linear combinations (discussed later).

Frequency Transforms

In some of our proposed systems, we use a frequency transformed version of the input signal as a preprocessing step to the system input. While no new information is gained from transforming the input, networks often respond better to determining the value of the magnitude of varying frequencies at a time slice instead of the individual time samples.

The frequency transform we use in this thesis is the discrete-time Fourier transform (DTFT). A sequence of N discrete-time samples is transformed into another sequence of N samples where each index then corresponds to a frequency bin. The DTFT $X[k]$ of a signal $x[n]$ is given by the following:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn} \quad (4)$$

where the twiddle factor W_N is given by $W_N = e^{-j(2\pi/N)}$. Then the reconstruction of $x[n]$ from $X[k]$ is given by:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] W_N^{-kn} \quad (5)$$

In this thesis, we also exploit the main duality between the time and frequency domain using the convolution theorem, which states that convolution in time is equivalent to multiplication in frequency and vice versa:

$$\mathcal{F}\{h[n] * x[n]\} = H[k]X[k] \quad (6)$$

$$\mathcal{F}^{-1}\{H[k] * X[k]\} = h[n]x[n] \quad (7)$$

This allows us to effectively treat our network as a non-linear filter that can denoise small time/frequency slices of our noisy signal, which can then be pieced back together using overlap-add resynthesis. We detail this in the next section.

Windowing and Perfect Reconstruction

To window a signal is to multiply a window function $w[n]$ by the frame, i.e. $w[n]x[n]$ over the frame length N . Because we are training a network to denoise small segments of a larger audio signal, we window the signal segments. This accommodates the finite-length requirement of the DTFT and helps to prevent spectral leakage. [DSPBOOK]

Also, to be able to properly reconstruct our signal, we use a window function and corresponding overlapping frame percentage to accomplish perfect reconstruction. The corresponding overlapping frame percentage is set such that the window sums to a constant for all time. For example, a rectangular window $w[n] = 1$ over an interval of length N has an overlap of 0% to sum to a constant 1 for all time. Another popular window is the Hanning window, defined over an interval N by the following:

$$w[n] = \frac{1}{2} \left(1 - \cos \left(\frac{2\pi n}{N-1} \right) \right) \quad (8)$$

For the Hann window, the perfect reconstruction overlap is a frame length of $N = 50\%$.

Window Size and Frequency v. Time Resolution Tradeoff

We must consider window size as a hyperparameter to our system. In general, shorter windows give rise to better time resolution at the cost of frequency resolution. On the other hand, longer windows give rise to better frequency resolution at the cost of time resolution. To illustrate, consider FIGURE.

Noise and Signal-to-Noise Ratio

Since we are trying to denoise audio signals, we must discuss how we measure noise. One of the most common measures of degradation of signal quality from additive noise is signal-to-noise ratio (SNR), defined as the ratio of signal variance to noise variance. [DSP] For the signal $y[n] = x[n] + N[n]$, where $x[n]$ is the signal of interest and $N[n]$ is the additive noise, the SNR is defined as

$$SNR = \frac{\sigma_x^2}{\sigma_n^2} \quad (9)$$

where σ^2 refers to the variance of the signal in question over some time interval. For the purposes of this thesis, we achieve desired a desired SNR for a simulation by scaling the noise to match the variance to the signal, then scaling the noise or the signal to achieve the desired SNR.

Magnitude and Phase Spectrum

Signal Model and Data

Network Input and Output

To simulate an audio denoising scheme, we define the following inputs and outputs. We take a known clean signal $x[n]$ which we subject to additive noise $N[n]$ using a specified SNR, resulting in the following noisy signal $y[n]$:

$$y[n] = x[n] + N[n] \quad (10)$$

To achieve a particular average SNR per simulation, we take the average signal energy for each minibatch of size B to determine a multiplicative scale factor k on the noise signal $N[n]$. For example, for additive white Gaussian noise (AWGN), we sample from the zero-mean, unit variance normal distribution (“randn” in Python) and determine our scale factor k as σ using the specified SNR in decibels:

$$\sigma_n^2 = \frac{1}{SNR_{lin}} \frac{1}{BN} \sum_{b=0}^{B-1} \sum_{n=0}^{N-1} x_b^2[n] \quad (11)$$

where SNR_{lin} is given by

$$SNR_{lin} = 10^{\frac{SNR_{db}}{10}} \quad (12)$$

In supervised scenarios, we allow the network to train with access to the ground truth $x[n]$. On the other hand, in semi-supervised scenarios, we only allow the network to train with access to a “soft label” indicating if the signal is (1) noise-only or (2) noise and possibly signal. [DanStowell] However, in both supervised and semi-supervised scenarios, our neural network input is one of the following:

1. Frames of $y[n]$
2. Frames of $\|Y[k]\|$
3. Magnitude spectrogram frames of $Y[k]$
4. Complex spectrogram frames of $Y[k]$

We choose the frame length L , time-domain window $w[n]$, and frame overlap percentage p as hyperparameters. Generally, we use 1024-sample frames with a Hanning window with 50% overlap unless otherwise specified.

Since we want to evaluate the level of denoising in the time domain, we recombine the network outputs with the noisy phase components of the spectrum if necessary to obtain an estimate $\hat{x}[n]$. We then compare $\hat{x}[n]$ to $x[n]$, in general using the mean squared error (MSE). For example, when our network outputs frames of $\|\hat{X}[k]\|$, we take the inverse Fast Fourier transform (IFFT) using the noisy phase $\angle Y[k]$ and use overlap-add to recombine the frames. (Anything to add about phase denoising and failures here?)

Signal and Noise Choices

Our choice of signals include the following:

1. Sine waves with multiple frequencies and random amplitudes and phases
2. Clean speech signals
3. Studio music recordings
4. Live concert recordings

Similarly, our choice of noise signals include the following:

1. Additive white Gaussian noise (AWGN)
2. Restaurant noise

As mentioned above, we can use the average energy per minibatch to specify a given SNR for an experiment. We take several combinations of clean and noise signals and compare across multiple SNRs.

Other Network Parameters

Since our networks involve one or more neural network layers, we show some results compared to choices of nonlinearity, number of layers (depth), and number of nodes in each layer (width). Generally, we use an identity at the network output and either the rectified linear unit (ReLU), a modified ReLU (mReLU), leaky rectify, hyperbolic tangent (tanh), or an exponential linear unit (elu).

972	De-noising Architectures
973	
974	
975	Classic Supervised Autoencoder
976	
977	
978	Paris Autoencoder
979	
980	
981	Partitioned Autoencoder
982	
983	
984	Phase Reconstruction
985	
986	Curro Autoencoder
987	
988	
989	
990	
991	
992	
993	
994	
995	
996	
997	
998	
999	
1000	
1001	
1002	
1003	
1004	
1005	
1006	
1007	
1008	
1009	
1010	
1011	
1012	
1013	
1014	
1015	
1016	
1017	
1018	
1019	
1020	
1021	
1022	
1023	
1024	
1025	

Results

Supervised Autoencoder

For the following results, we show

Batch Normalized Input

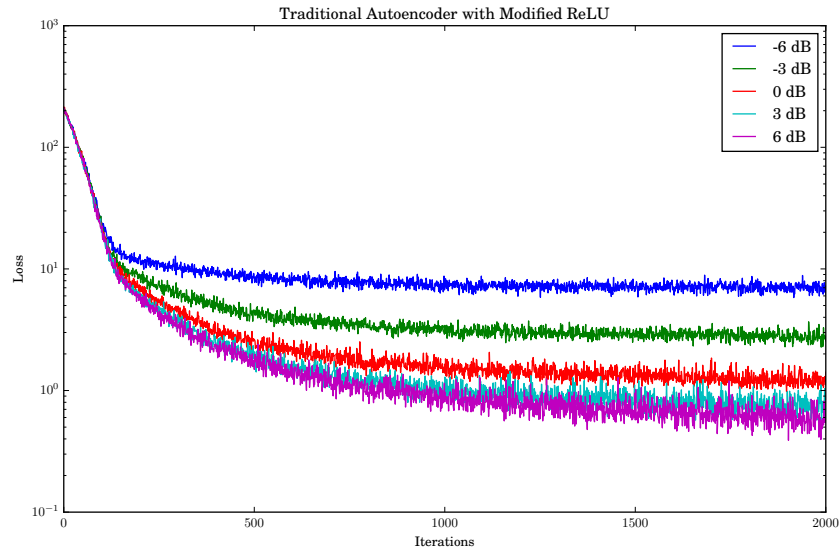


Figure 1: Loss at various SNRs for Supervised Single-Layer Autoencoder with Batch Normalization at the Input

Non-Batch Normalized Input

Partitioned Autoencoder

Partitioned Curro Autoencoder

Comparison of Loss Convergence

Comparison of Mean Squared Error Convergence

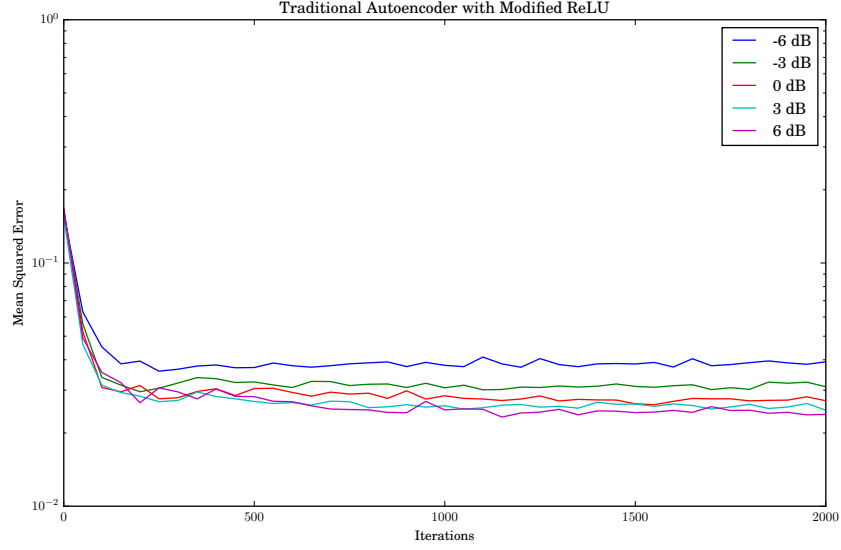


Figure 2: MSE at various SNRs for Supervised Single-Layer Autoencoder with Batch Normalization at the Input

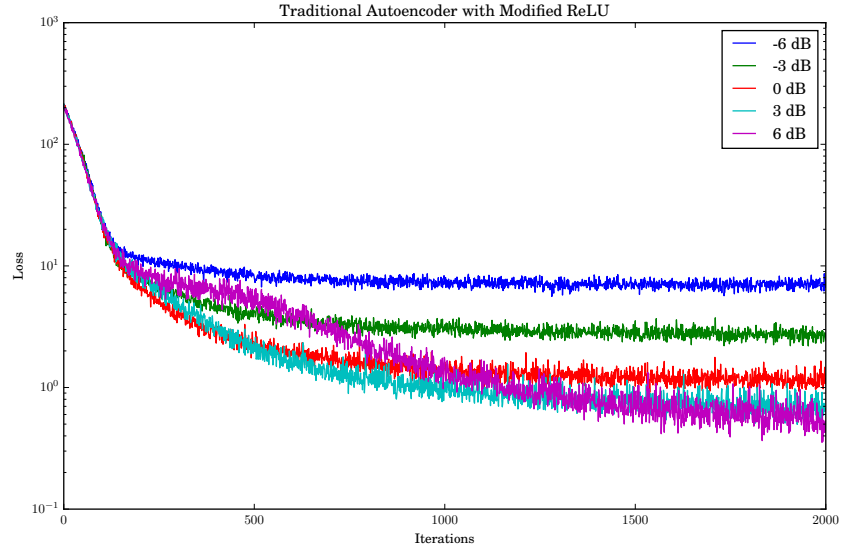


Figure 3: Loss at various SNRs for Supervised Single-Layer Autoencoder without Batch Normalization at the Input

Conclusions and Future Work

Conclusions

While more work is needed, deep partitioned neural network architectures using time and frequency data seem promising in long-term solutions for denoising speech and music signals.

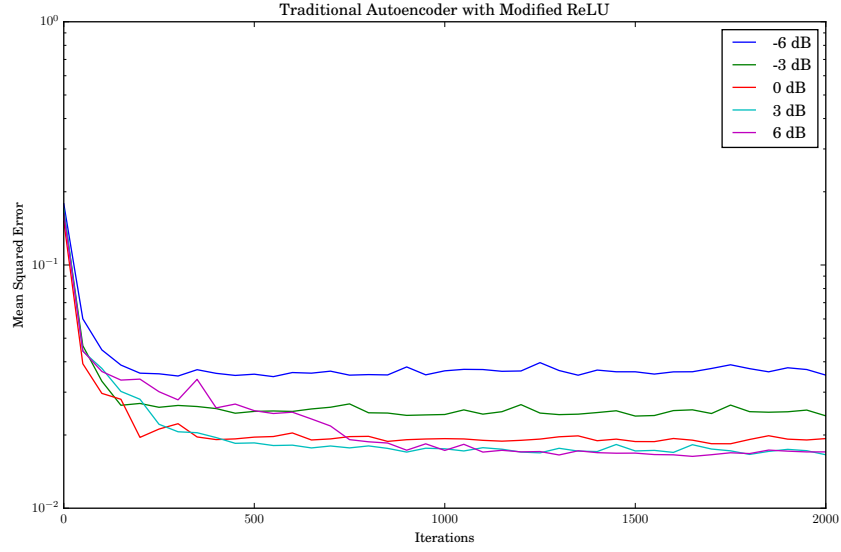


Figure 4: MSE at various SNRs for Supervised Single-Layer Autoencoder with Batch Normalization at the Input

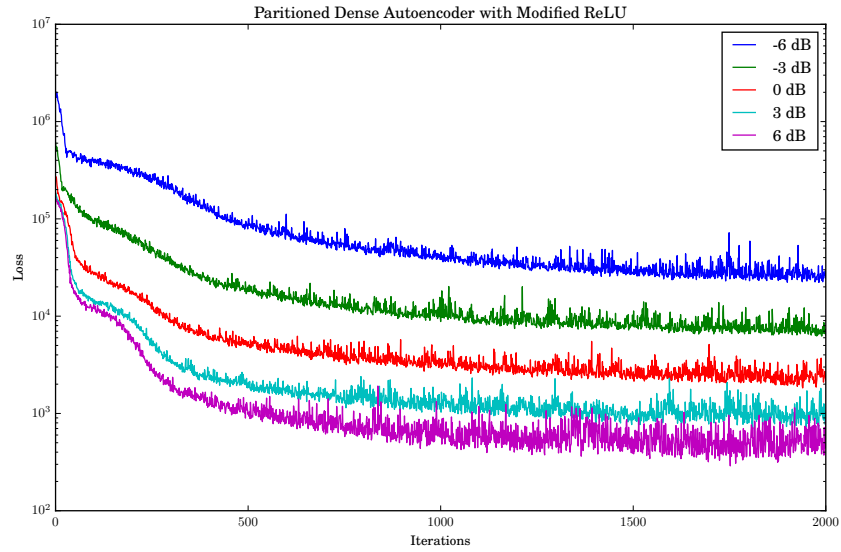


Figure 5: Loss at various SNRs for Single-Layer Partitioned Autoencoder [1]

Future Work

Models

Make network deeper. Consider gradual partitioning instead of hard.

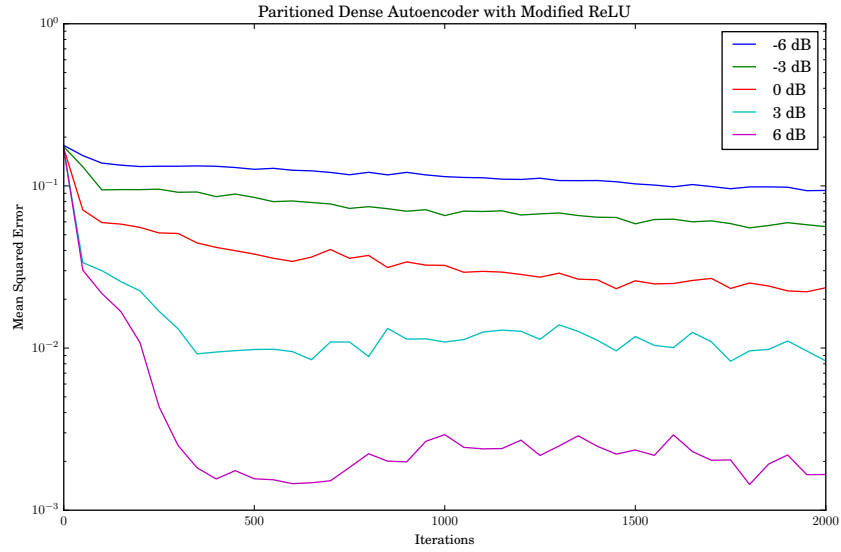


Figure 6: MSE at various SNRs for Single-Layer Partitioned Autoencoder [1]

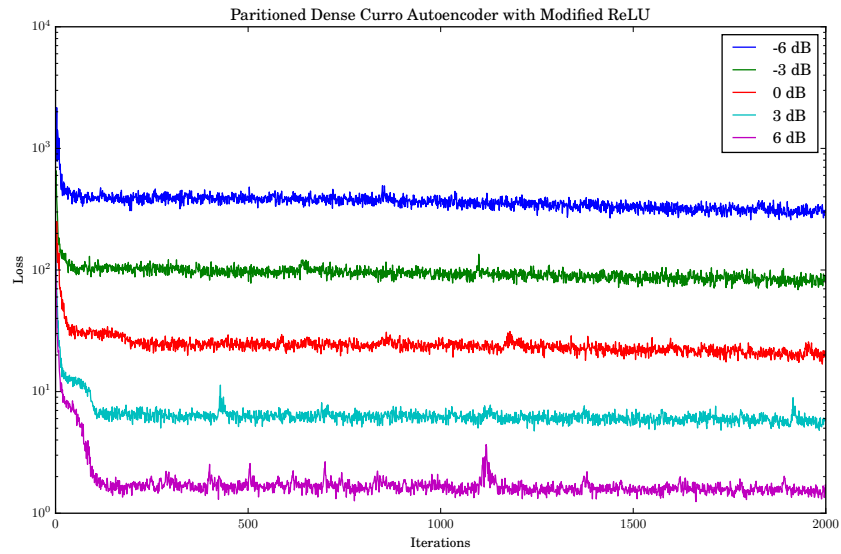


Figure 7: Loss at various SNRs for Single-Layer Curro Autoencoder

Data

Get more data. Consider different noise levels and types of signals.

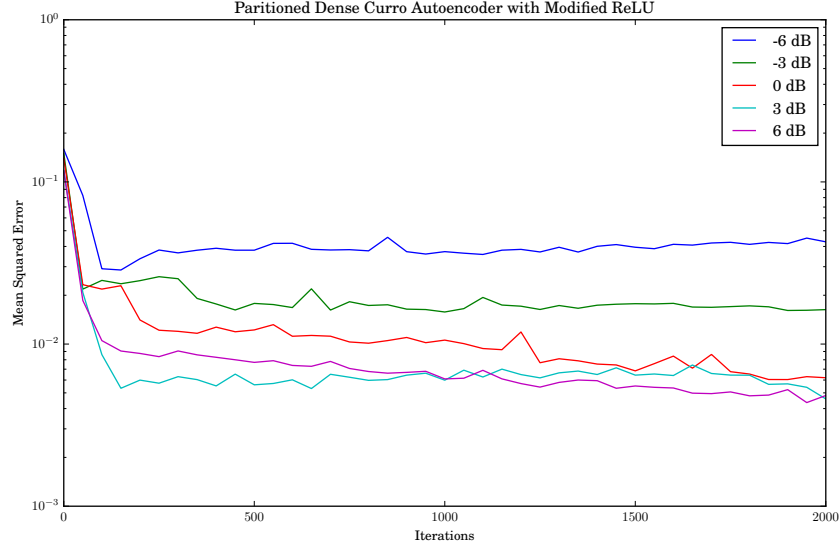


Figure 8: MSE at various SNRs for Single-Layer Curro Autoencoder

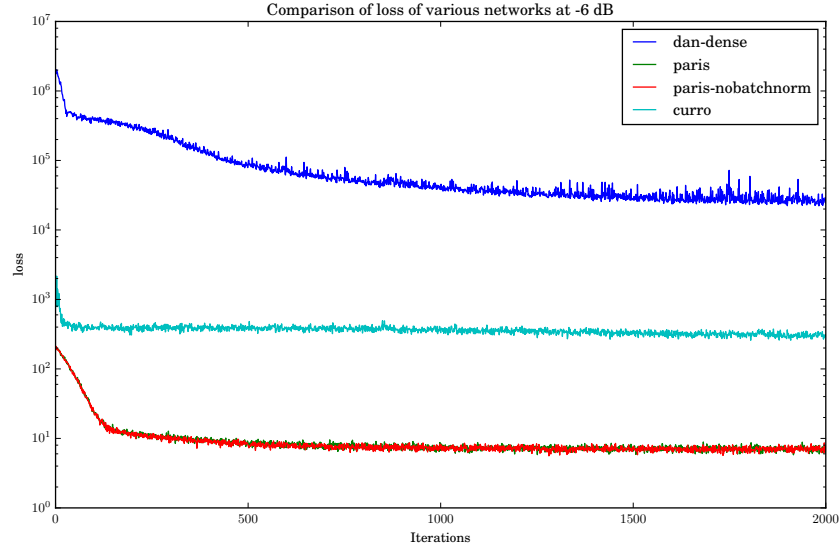


Figure 9: Loss Comparison of Various Networks at -6 dB

References

- [1] D. Stowell and R. E. Turner, “Denoising without access to clean data using a partitioned autoencoder,” *CoRR*, vol. abs/1509.05982, 2015. [Online]. Available: <http://arxiv.org/abs/1509.05982>
- [2] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.
- [3] D. Liu, P. Smaragdis, and M. Kim, “Experiments on deep learning for speech denoising.” in *INTERSPEECH*, 2014, pp. 2685–2689.

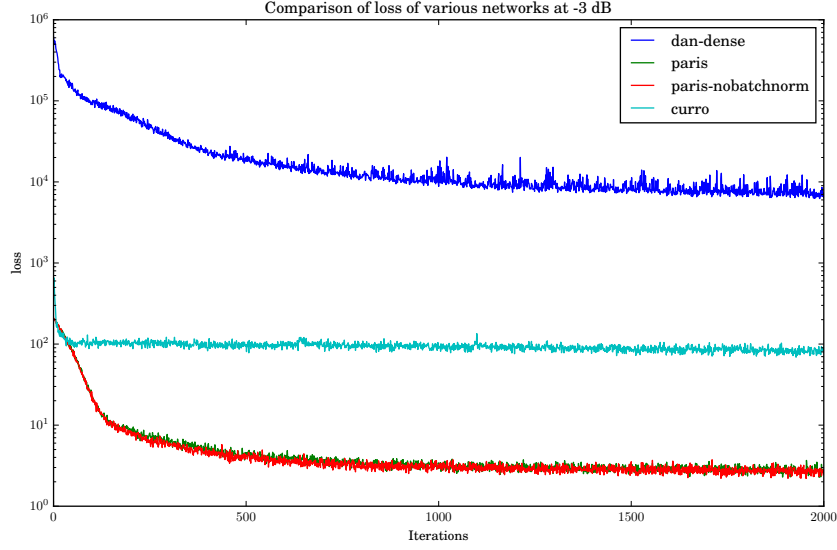


Figure 10: Loss Comparison of Various Networks at -3 dB

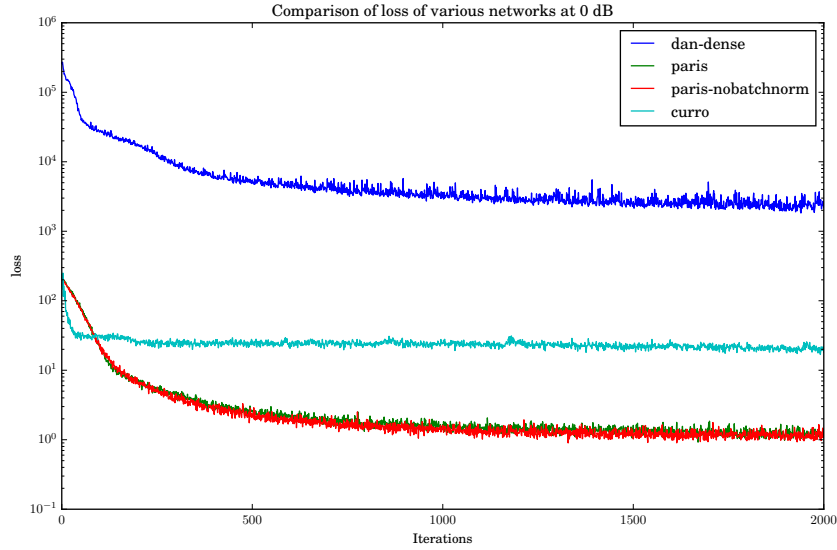


Figure 11: Loss Comparison of Various Networks at 0 dB

- [4] P. Baldi and Z. Lu, “Complex-valued autoencoders,” *Neural Networks*, vol. 33, pp. 136–147, 2012.
- [5] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, “An experimental study on speech enhancement based on deep neural networks,” *IEEE Signal Processing Letters*, vol. 21, no. 1, pp. 65–68, 2014.
- [6] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, “Stacked denoising autoencoders: Learning useful representations in a

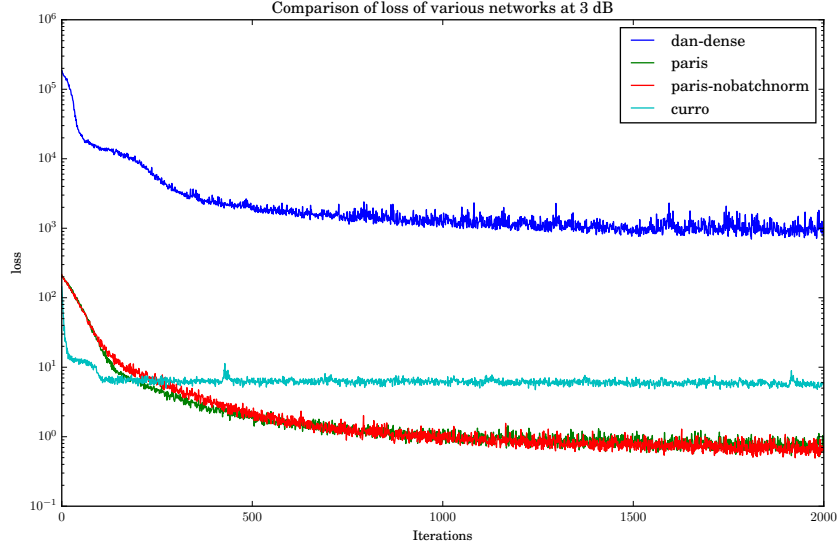


Figure 12: Loss Comparison of Various Networks at 3 dB

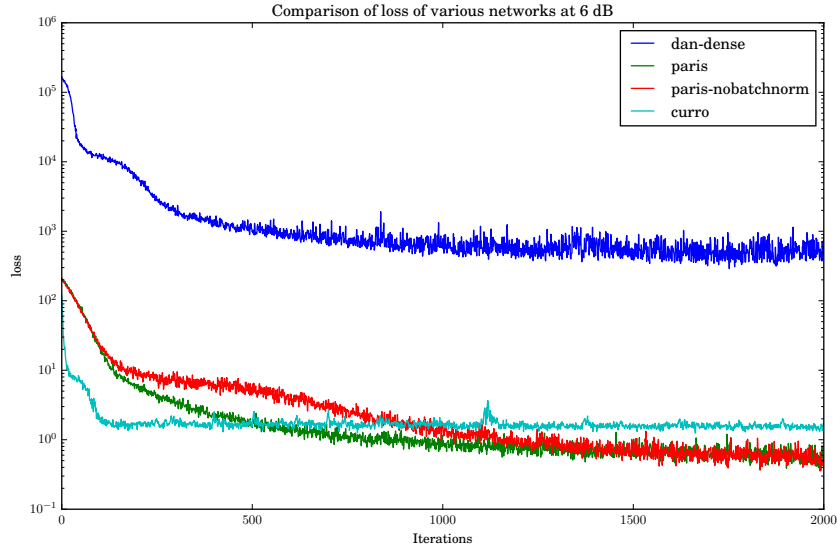


Figure 13: Loss Comparison of Various Networks at 6 dB

deep network with a local denoising criterion,” *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

- [7] T. Ishii, H. Komiyama, T. Shinozaki, Y. Horiuchi, and S. Kuroiwa, “Reverberant speech recognition based on denoising autoencoder.” in *INTER-SPEECH*, 2013, pp. 3512–3516.
- [8] V. O. Alan, W. S. Ronald, and R. John, “Discrete-time signal processing,” *New Jersey, Printice Hall Inc*, 1989.

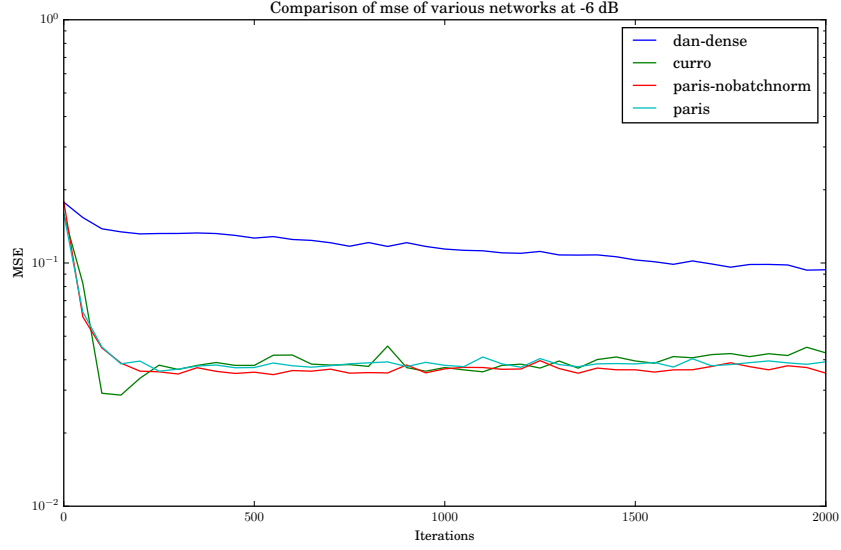


Figure 14: MSE Comparison of Networks at -6 dB

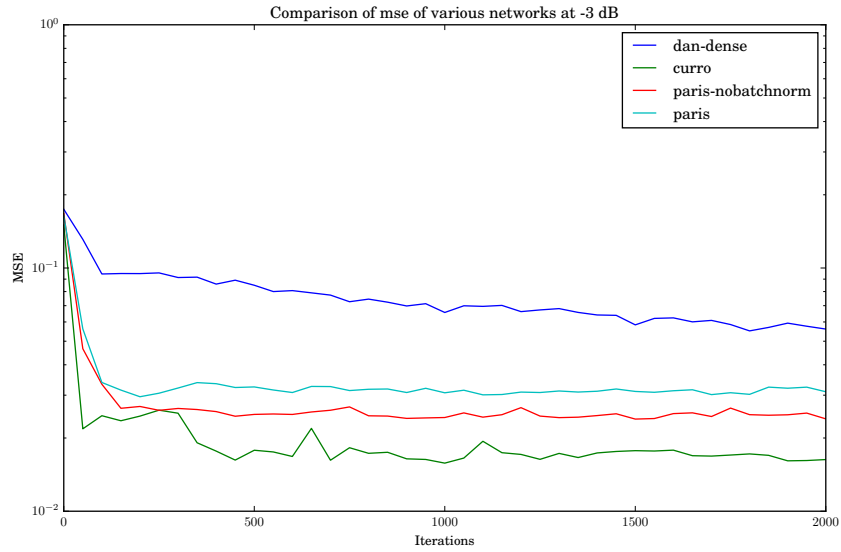


Figure 15: MSE Comparison of Networks at -3 dB

- [9] B. Gold, N. Morgan, and D. Ellis, "Speech and audio signal processing: processing and perception of speech and music," 2011.
- [10] M. Kayser and V. Zhong, "Denoising convolutional autoencoders for noisy speech recognition."
- [11] U. Zölzer, *Digital audio signal processing*. John Wiley & Sons, 2008.
- [12] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507,

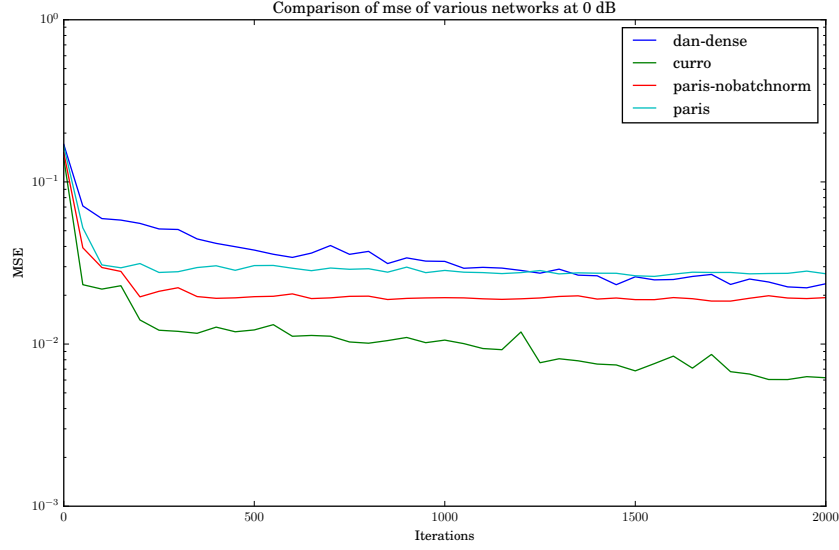


Figure 16: MSE Comparison of Networks at 0 dB

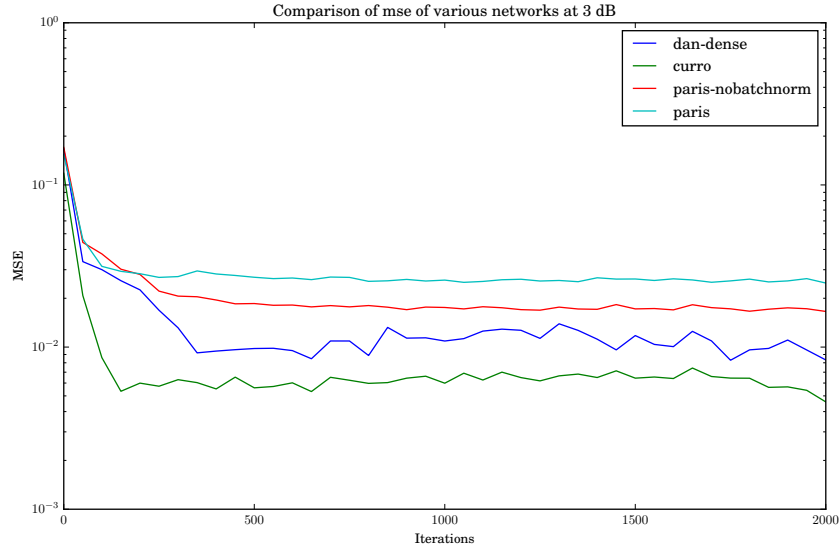


Figure 17: MSE Comparison of Networks at 3 dB

2006.

- [13] A. Rad and T. Virtanen, "Phase spectrum prediction of audio signals," in *International Symposium on Communications Control and Signal Processing (ISCCSP)*, 2012, pp. 1–5.
- [14] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016. [Online]. Available: <http://arxiv.org/abs/1605.02688>

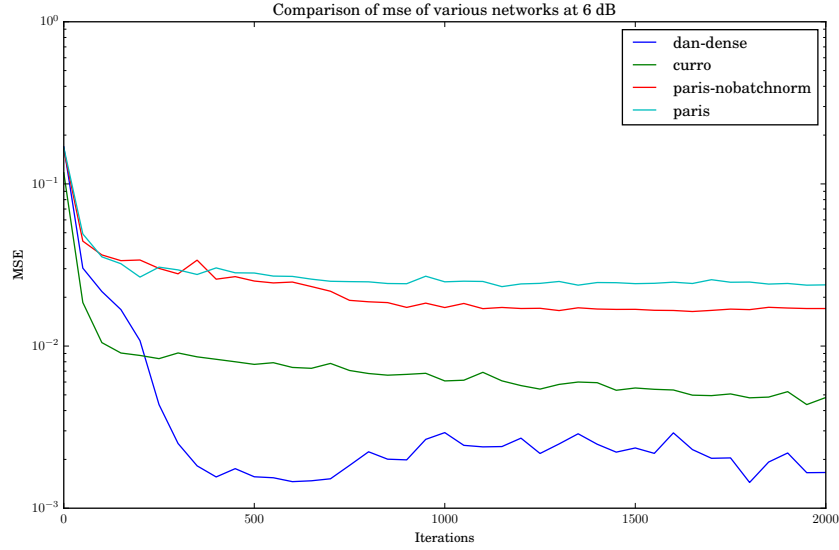


Figure 18: MSE Comparison of Networks at 6 dB

- [15] S. Dieleman, J. Schlüter, C. Raffel, E. Olson, S. K. Sønderby, D. Nouri, D. Maturana, M. Thoma, E. Battenberg, J. Kelly, J. D. Fauw, M. Heilman, diogo149, B. McFee, H. Weideman, takacsg84, peterderivaz, Jon, instagibbs, D. K. Rasul, CongLiu, Britefury, and J. Degraeve, “Lasagne: First release.” Aug. 2015. [Online]. Available: <http://dx.doi.org/10.5281/zenodo.27878>