ON ADVERSARIAL TRAINING AND LOSS FUNCTIONS FOR SPEECH ENHANCEMENT

Ashutosh Pandey[1] *and Deliang Wang*[1,2]

[1]Department of Computer Science and Engineering, The Ohio State University, USA
[2]Center for Cognitive and Brain Sciences, The Ohio State University, USA
{pandey.99, wang.5664, wang.77}@osu.edu

ABSTRACT

Generative adversarial networks (GANs) are becoming increasingly popular for image processing tasks. Researchers have started using GANs for speech enhancement, but the advantage of using the GAN framework has not been established for speech enhancement. For example, a recent study reports encouraging enhancement results, but we find that the architecture of the generator used in the GAN gives better performance when it is trained alone using the L_1 loss. This work presents a new GAN for speech enhancement, and obtains performance improvement with the help of adversarial training. A deep neural network (DNN) is used for time-frequency mask estimation, and it is trained in two ways: regular training with the L_1 loss and training using the GAN framework with the help of an adversary discriminator. Experimental results suggest that the GAN framework improves speech enhancement performance. Further exploration of loss functions, for speech enhancement, suggests that the L_1 loss is consistently better than the L_2 loss for improving the perceptual quality of noisy speech.

Index Terms— Speech enhancement, generative adversarial networks, deep learning, L_1 loss, fully connected

1. INTRODUCTION

Speech enhancement or separation is the task of removing additive noise from a speech signal. It is important for many applications, such as robust automatic speech recognition, automatic speaker verification, mobile speech communication and hearing aids design. Traditional speech enhancement approaches include spectral subtraction [1], Wiener filtering [2], statistical model-based methods [3] and nonnegative matrix factorization [4].

Deep learning has been applied for speech enhancement in past few years, and deep learning based methods have become the state of the art, due to its ability to learn complex hierarchical functions from data. Some of the most popular deep learning based methods are: deep denoising autoencoders [5], DNNs [6, 7], and convolutional neural networks [8]. An overview of deep learning based methods for speech separation is given recently in [9].

In recent years, GAN [10] based approaches were explored [11, 12]. In these studies, new architectures were proposed for speech enhancement, and trained adversarially. These works, however, do not present a convincing picture for the suitability of GANs for speech enhancement. We have explored the architecture proposed in [11], and found that training of the generator with L_1 loss alone performs better in terms of short term objective intelligibility (STOI) [13] and perceptual evaluation of speech quality (PESQ) [14] scores, accounting for reported improvements. This does not answer the question that whether GANs are a promising framework for speech enhancement or not. In this paper, we address this question by exploring GANs for a simple DNN. Specifically, we use a DNN for spectral magnitude mask (SMM) estimation, and train it in two ways: regular training of the generator with L_1 loss and adversarial training. Adversarial training means that we use the mask estimation network as the generator in the GAN framework.

In image processing tasks, researchers claim that the L_1 loss is better than L_2 loss because it promotes less blurring of images [15, 16]. This raises the question whether L_1 loss is better for speech enhancement or not. In fact, in [11] and [12] the authors have used L_1 loss for speech enhancement citing the arguments used for image processing. In this study, we also do a comparison between L_1 and L_2 loss so that an informed decision can be taken for their use for speech enhancement.

This paper is organized as follows: We first describe GANs in the next section followed by an explanation of GAN used in this study. Section 4 talks about the comparison between L_1 and L_2 loss. Experiment details are given in Section 5. Section 6 presents and discusses the results and Section 7 concludes the paper.

2. GENERATIVE ADVERSARIAL NETWORKS

Generative adversarial networks (GANs) [10] are models that learn to generate samples from a given data distribution, $p_{data}(x)$. GANs consist of two networks: a generator G and a discriminator D. The job of G is to map a noise vector z, from a given prior distribution $p_z(z)$, to an output sample $G(z)$ resembling the data samples in the training data.

The discriminator D is a classifier network which determines whether its input is real or fake. A sample coming from the training data, x, is considered real, and a sample coming from the generator output, $G(z)$, is considered fake. The generator and discriminator are trained in an adversarial way where both networks play a mini-max game, and try to maximize their own utility function. The generator tries to fool the discriminator by producing samples which are very close to the samples from the training data, and the discriminator tries to be good at classifying real and fake data. The loss functions for the mini-max game are defined as following [10]:

$$J^{(D)} = -\frac{1}{2}E_{x\sim p_{data}(x)}\log D(x) - \frac{1}{2}E_{z\sim p_z(z)}\log(1 - D(G(z))) \quad (1)$$

$$J^{(G)} = -J^{(D)} \quad (2)$$

Conditional GANs [17] can be used for the regression task. In a conditional GAN, some extra information, a vector x_c, is provided along with the noise vector z at the input of the generator. The equation for the cost function of the discriminator in a conditional GAN is given as:

$$J^{(D)} = -\frac{1}{2}E_{x,x_c\sim p_{data}(x,x_c)}\log D(x, x_c) - \frac{1}{2}E_{x_c\sim p_{data}(x_c),z\sim p_z(z)}\log(1 - D(G(z, x_c))) \quad (3)$$

The equations presented above are prone to cause the vanishing gradient problem in the early stages of training [18], so we use slightly modified loss functions which were proposed in [19] and also used in [11]. In this GAN, the loss functions of the generator and discriminator use mean squared error as given in equation (4) and (5).

$$J^{(D)} = \frac{1}{2}E_{x,x_c\sim p_{data}(x,x_c)}[(D(x, x_c) - 1)^2] + \frac{1}{2}E_{x_c\sim p_{data}(x_c),z\sim p_z(z)}[D(G(z, x_c))^2] \quad (4)$$

$$J^{(G)} = \frac{1}{2}E_{x_c\sim p_{data}(x_c),z\sim p_z(z)}[D(G(z, x_c) - 1]^2 \quad (5)$$

3. FULLY CONNECTED GAN FOR SPEECH ENHANCEMENT

As mentioned in the previous section, a conditional GAN can be used for a regression task. In this work, we use a GAN for the regression task of time-frequency mask estimation. Short-term Fourier transform (STFT) magnitude features of noisy speech are used as extra information (x_c in equations (4) and (5)) to generate the target SMM from the GAN. The equation for SMM is defined as:

$$SMM(t, f) = \frac{|S(t, f)|}{|Y(t, f)|} \quad (6)$$

where $S(t, f)$ and $Y(t, f)$ are STFT coefficients of clean speech and noisy speech respectively.

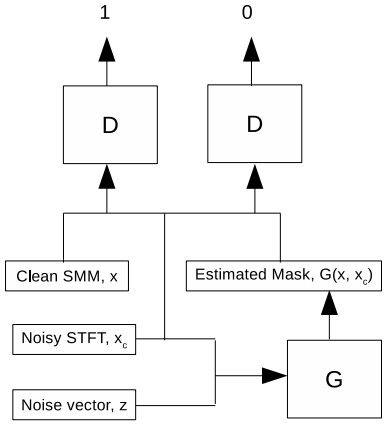

Fig. 1. Mask estimation framework using conditional GAN

The G network here is a fully connected DNN which takes as input, the concatenation of two vectors, x_c and z, where x_c is the STFT magnitude of noisy speech and z is a randomly sampled noise vector from a normal distribution. Output of the G network, $G(z, x_c)$, is the estimated SMM.

The D network is also a fully connected DNN with the same number of hidden layers as that in the generator network. The discriminator loss is computed using two different batches of data: a real batch and a fake batch. For the real batch, the input to the discriminator is a concatenation of the target SMM and the noisy speech STFT magnitude, $[x, x_c]$ and the expected output is 1. Similarly, for the fake batch, the input to the discriminator is a concatenation of the estimated SMM and the noisy speech STFT, $[G([z, x_c]), x_c]$ and the expected output is 0. This setting is illustrated in Fig. 1.

It is difficult to train a conditional GAN using the loss functions given in equations (4) and (5). Therefore, we additionally use the L_1 loss in the generator to stabilize the training. The use of L_1 loss to stabilize the training was proposed in [15], and adopted for speech enhancement in [11, 12]. The modified loss function of the generator becomes:

$$J^{(G)} = \frac{1}{2}E_{x_c\sim p_{data}(x_c),z\sim p_z(z)}[D(G(z, x_c) - 1]^2 + \lambda||G(z, x_c) - x||_1 \quad (7)$$

where λ is a hyper parameter which can be tuned to stabilize the training.

4. L_1 LOSS VERSUS L_2 LOSS

In our initial experiments, we compare the GAN training with the L_2 loss training and observe some improvement. Further exploration reveals that the observed improvement is due to the additional L_1 loss in the generator (equation 7), but not due to adversarial training. Therefore, we decide to compare

SNR	System	Babble				Factory				SSN				Engine				Oproom			
		STOI (%)		PESQ		STOI (%)		PESQ		STOI (%)		PESQ		STOI (%)		PESQ		STOI (%)		PESQ	
		Z OFF	Z ON	Z OFF	Z ON	Z OFF	Z ON	Z OFF	Z ON	Z OFF	Z ON	Z OFF	Z ON	Z OFF	Z ON	Z OFF	Z ON	Z OFF	Z ON	Z OFF	Z ON
-5 dB	Mixture	54.85	54.85	1.41	1.41	54.31	54.31	1.30	1.30	56.75	56.75	1.49	1.49	57.26	57.26	1.40	1.40	59.54	59.54	1.41	1.41
	DNN	63.64	63.76	1.70	1.70	66.88	66.86	1.75	1.75	72.76	72.90	1.97	1.97	79.81	80.47	2.31	2.34	79.49	80.11	2.34	2.37
	GAN	**65.15**	**65.56**	1.71	1.72	**68.15**	**68.25**	1.77	1.78	**74.09**	**73.98**	2.01	2.00	**80.79**	**80.66**	2.36	2.35	**80.03**	**81.10**	2.35	2.40
0 dB	Mixture	66.57	66.57	1.74	1.74	66.56	66.56	1.64	1.64	69.16	69.16	1.75	1.75	68.96	68.96	1.66	1.66	69.76	69.76	1.78	1.78
	DNN	76.49	76.64	2.16	2.16	78.60	78.79	2.25	2.25	82.92	83.12	2.40	2.41	86.46	86.90	2.68	2.70	84.82	85.26	2.67	2.70
	GAN	**77.84**	**78.27**	2.18	2.18	**79.94**	**80.06**	2.26	2.27	**83.99**	**83.95**	2.44	2.43	**87.26**	**87.17**	2.68	2.67	**85.34**	**86.05**	2.66	2.72
5 dB	Mixture	77.48	77.48	2.09	2.09	77.71	77.71	2.01	2.01	80.81	80.81	2.05	2.05	80.19	80.19	1.97	1.97	79.07	79.07	2.16	2.16
	DNN	85.15	85.52	2.59	2.61	86.42	86.80	2.66	2.66	89.41	89.66	2.79	2.80	91.17	91.39	2.97	2.99	88.83	89.18	2.97	2.99
	GAN	**86.00**	**86.33**	2.61	2.62	**87.48**	**87.47**	2.67	2.68	**90.30**	**90.34**	2.81	2.81	**91.67**	**91.61**	2.96	2.95	**89.32**	**89.75**	2.96	2.99

Table 1. Performance comparison between L_1 loss training and GAN training on 3 different SNR conditions.

adversarial training and regular L_1 loss training. This observation also raises the question of the L_1 loss being better than the L_2 loss for speech enhancement. Also, in image processing tasks, [15] and [16] claim that L_1 loss encourages less blurring of images than L_2 loss. [11] and [12] use L_1 loss for speech enhancement citing the arguments given in [15] and [16]. We perform a comparison between L_1 and L_2 loss for speech enhancement to have a better understanding. Specifically, we train DNNs for three different targets: ideal ratio mask (IRM) [9], SMM and STFT magnitude, using L_1 and L_2 loss. Performance comparisons are reported in Table 3 and Table 4, and discussed in Section 6.

5. EXPERIMENTAL SETTINGS

5.1. Dataset

We create the training and test sets in the same manner as in [20]. 2000 utterances are randomly selected from the TIMIT [21] dataset as training utterances. 192 utterances from the core test set of TIMIT are used as test utterances. Noise specific and noise generalized models are trained. Performance is evaluated using STOI and PESQ scores. For the noise specific case, models are trained and tested on the same noise. Specifically, five noises, babble, factory, SSN, oproom and engine are used. For the noise generalized case, the model is trained on all five noises mentioned above and tested on two unseen noises, factory2 and buccaneer1. All the noises except SSN are obtained from Noisex dataset [22]. SSN is generated using the training utterances. Training noisy utterances are created by adding noises at -5 dB and 0 dB. Test noisy utterances are created by mixing noises at -5 dB, 0dB and 5 dB SNR where 5dB is an unseen SNR condition used to assess the SNR generalizability of the model.

5.2. Preprocessing

The audio signals are resampled to 16kHz. A 512-point STFT is computed using a 32 ms Hanning window and a shift of 16 ms (256 samples). 512 points STFT magnitudes are reduced to 257 points by removing the symmetric half. Five consecutive time frames are concatenated to form the input, and the masks for corresponding five frames are predicted together at the output. This makes the input and the output dimensions of the generator to be 1285. Multiple predictions of the mask, for a given time frame, are averaged together. Inputs to the generator are normalized to have zero mean and unit variance.

All SMM values greater than 10 are clipped to 10 and then transformed to the range $[-1, 1]$. Range transformation is required because we use tanh at the generator output in the GAN. Tanh is used for training convergence [18].

SNR	System	Factory2				Buccaneer1			
		STOI (%)		PESQ		STOI (%)		PESQ	
		Z OFF	Z ON	Z OFF	Z ON	Z OFF	Z ON	Z OFF	Z ON
-5 dB	Mixture	65.64	65.64	1.59	1.59	51.97	51.97	1.19	1.19
	DNN	76.45	76.60	2.20	2.23	58.94	59.02	1.44	1.43
	GAN	76.37	**76.76**	2.25	2.22	**59.42**	**60.09**	1.43	1.45
0 dB	Mixture	76.05	76.05	1.95	1.95	63.48	63.48	1.49	1.49
	DNN	84.45	84.59	2.60	2.63	71.46	71.59	1.71	1.71
	GAN	84.42	**85.05**	2.61	2.61	**71.99**	**72.45**	1.71	1.72
5 dB	Mixture	84.48	84.48	2.30	2.30	75.09	75.09	1.83	1.83
	DNN	89.40	89.48	2.92	2.93	81.95	82.02	2.12	2.12
	GAN	89.22	**89.86**	2.92	2.94	**82.39**	**82.57**	2.10	2.11

Table 2. Generalization performance of L_1 loss training and CGAN training on 2 unseen noises and 3 SNR conditions.

5.3. Training

All DNNs use 3 hidden layers. Batch normalization [23] is used before every layer except the output layer of the discriminator and the input layer of the generator. A dropout rate of 0.2 is used for all the hidden layers. The discriminator uses leaky ReLUs at the hidden layers and no activation at the output layer [11]. The generator uses parametric ReLUs at the hidden layers and the output layer activation is determined by targets. For comparison between L_1 loss training and adversarial training, tanh is used at the output layer of the generator. For L_1 and L_2 loss comparison, ReLU is used for STFT magnitude and SMM, and sigmoid is used for IRM. SMM is not transformed to $[-1, 1]$ after clipping when comparing the loss functions.

The generator in the GAN is trained for two cases, with and without an input noise vector. The noise vector is of dimension 100, and is sampled from a normal distribution with zero mean and unit variance. With noise, the dimension of input to the generator is 1385, and the number of units in each hidden layer are 1124. Without noise, the number of units in each hidden layer are 1024. The number of hidden units in the discriminator are twice of that in the generator.

SNR	Target	Babble				Factory				SSN				Engine				oproom			
		STOI (%)		PESQ		STOI (%)		PESQ		STOI (%)		PESQ		STOI (%)		PESQ		STOI (%)		PESQ	
		L_1	L_2	L_1	L_2	L_1	L_2	L_1	L_2	L_1	L_2	L_1	L_2	L_1	L_2	L_1	L_2	L_1	L_2	L_1	L_2
-5 dB	Mixture	54.85	54.85	1.41	1.41	54.31	54.31	1.30	1.30	56.75	56.75	1.49	1.49	57.26	57.26	1.40	1.40	59.54	59.54	1.41	1.41
	IRM	**63.66**	63.08	1.67	1.66	**66.62**	65.74	**1.74**	1.69	**72.37**	71.30	**1.96**	1.89	**80.31**	79.81	**2.32**	2.24	**80.12**	79.45	**2.35**	2.25
	STFT	**65.56**	64.54	**1.78**	1.59	**69.38**	68.62	**1.87**	1.69	**75.05**	73.63	**2.09**	1.83	**81.67**	81.19	**2.19**	1.86	**81.07**	80.30	**2.22**	1.87
	SMM	**63.80**	**63.75**	**1.68**	1.63	66.74	66.75	**1.74**	1.67	72.69	73.72	**1.95**	1.90	80.19	81.15	**2.27**	2.18	80.01	80.42	**2.32**	2.21
0 dB	Mixture	66.57	66.57	1.74	1.74	66.56	66.56	1.64	1.64	69.16	69.16	1.75	1.75	68.96	68.96	1.66	1.66	69.76	69.76	1.78	1.78
	IRM	**76.84**	75.99	2.15	2.11	**78.79**	77.88	**2.24**	2.18	**82.95**	82.13	**2.42**	2.33	**86.80**	86.28	**2.71**	2.59	**85.33**	84.64	**2.70**	2.59
	STFT	**78.59**	77.63	**2.17**	1.85	**80.73**	79.94	**2.28**	1.97	**84.28**	83.32	**2.41**	2.00	**87.16**	86.71	**2.47**	2.12	**85.62**	85.10	**2.46**	2.12
	SMM	**76.62**	76.35	**2.15**	2.07	**78.73**	78.62	**2.24**	2.14	82.91	83.52	**2.41**	2.30	86.61	87.18	**2.64**	2.50	85.06	85.28	**2.65**	2.51
5 dB	Mixture	77.48	77.48	2.09	2.09	77.71	77.71	2.01	2.01	80.81	80.81	2.05	2.05	80.19	80.19	1.97	1.97	79.07	79.07	2.16	2.16
	IRM	**85.83**	85.26	**2.59**	2.54	**86.90**	86.32	**2.67**	2.60	**89.92**	89.48	**2.83**	2.73	**91.56**	91.24	**3.04**	2.91	**89.33**	88.82	**3.01**	2.89
	STFT	**85.77**	84.98	**2.44**	2.03	**86.53**	85.27	**2.43**	2.06	**89.19**	88.02	**2.57**	2.10	**90.47**	89.90	**2.65**	2.26	**88.51**	87.78	**2.63**	2.25
	SMM	**85.48**	85.28	**2.60**	2.48	86.74	86.70	**2.66**	2.54	89.69	90.02	**2.79**	2.65	91.32	91.61	**2.95**	2.78	89.05	89.19	**2.95**	2.78

Table 3. Performance comparison between L_1 and L_2 loss on 3 different SNR conditions.

For equation (7), we set λ to 100. The training of the generator and discriminator is done as described in [10]. One-sided label smoothing with α equal to 0.9 is used [24]. The discriminator is updated twice for each update of the generator. The Adam optimizer [25] is used for SGD based optimization with a batch size of 1024. A learning rate of 0.0002 and $\beta_1 = 0.5$ is used, as in [18]. Adversarial training shows some oscillatory behaviour initially, but converges when trained for a long time. Once converged, the value of the generator loss, in equation (5), converges to 0.81.

SNR	Target	Factory2				Buccaneer1			
		STOI		PESQ		STOI		PESQ	
		L_1	L_2	L_1	L_2	L_1	L_2	L_1	$L2$
-5 dB	Mixture	65.64	65.64	1.59	1.59	51.97	51.97	1.19	1.19
	IRM	76.53	76.82	2.19	2.17	59.38	59.58	1.47	1.43
	STFT	76.90	77.57	**1.97**	1.70	62.53	63.29	**1.49**	1.25
	SMM	76.59	77.78	**2.22**	2.16	60.03	61.11	**1.46**	1.41
0 dB	Mixture	76.05	76.05	1.95	1.95	63.48	63.48	1.49	1.49
	IRM	**84.92**	84.74	**2.64**	2.55	**71.94**	71.72	1.75	1.75
	STFT	83.36	83.94	**2.24**	1.94	74.48	75.59	**1.82**	1.59
	SMM	84.26	85.22	**2.61**	2.53	72.10	72.61	1.73	1.70
5 dB	Mixture	84.48	84.48	2.30	2.30	75.09	75.09	1.83	1.83
	IRM	**89.95**	89.88	**2.99**	2.88	**82.13**	81.92	2.14	2.14
	STFT	85.71	86.27	**2.38**	2.07	83.54	83.63	**2.19**	1.85
	SMM	89.10	89.97	**2.93**	2.85	82.01	82.21	2.11	2.08

Table 4. Generalization performance of L_1 and L_2 loss for 2 unseen noises at 3 different SNR conditions.

6. RESULTS AND DISCUSSIONS

We first compare the performance of L_1 loss training and adversarial training. Results are reported in Table 1. In Table 1, Z OFF and Z ON denote the presence and absence of noise vector at the input of the generator, respectively. We observe that the GAN gives consistently better STOI score, but there is not much of an improvement in PESQ score. STOI improvements are better at low SNR conditions. The importance of using a noise vector at the input of the generator is not very clear from the noise specific models. Therefore, a noise generalized model is also trained. The generalization performance is given in Table 2. We observe that on factory2 noise, the GAN gives consistent improvement with the noise vector but is not able to improve the performance without it. Also, for buccaneer1 noise, the performance improvement is better when the noise vector is used. This indicates the importance of the noise vector for generalization.

Next, we compare L_1 and L_2 loss for speech enhancement. Results are reported in Table 3 and Table 4. L_1 loss gives a better PESQ score for all the cases. Specifically, a significant improvement in PESQ is observed for the STFT magnitude estimation. Additionally, we observe that L_1 loss gives a better STOI score when the targets are IRM and STFT magnitude and a noise specific model is trained. L_2 loss gives a better generalization performance in terms of STOI score.

7. CONCLUSIONS

In this work, we demonstrate the effectiveness of adversarial training for speech enhancement. We train a DNN with L_1 loss and with adversarial training, and show that a given DNN performs better speech enhancement with adversarial training. Additionally, we compare L_1 and L_2 loss for speech enhancement using three different targets, and find that L_1 loss consistently gives a better PESQ score, but does not give a better generalization performance for the STOI score. This work makes a case for further exploration of adversarial training for speech enhancement. A possible future work is the use of GANs for fine tuning, in which the generator further enhances a signal already enhanced with regular L_1 or L_2 loss training.

8. ACKNOWLEDGEMENTS

This research was supported in part by two NIDCD (R01 DC012048 and R02 DCDC015521) grants and the Ohio Supercomputer Center. We thank H. Zhang for discussions on the paper.

9. REFERENCES

[1] S. Boll, "Suppression of acoustic noise in speech using spectral subtraction," *IEEE Transactions on acoustics,*

speech, and signal processing, vol. 27, no. 2, pp. 113–120, 1979.

[2] P. Scalart et al., "Speech enhancement based on a priori signal to noise estimation," in *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*. IEEE, 1996, vol. 2, pp. 629–632.

[3] P. C. Loizou, *Speech enhancement: theory and practice*, CRC press, 2013.

[4] N. Mohammadiha, P. Smaragdis, and A. Leijon, "Supervised and unsupervised speech enhancement using nonnegative matrix factorization," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 10, pp. 2140–2151, 2013.

[5] X. Lu, Y. Tsao, S. Matsuda, and C. Hori, "Speech enhancement based on deep denoising autoencoder.," in *Interspeech*, 2013, pp. 436–440.

[6] Y. Wang and D. Wang, "Towards scaling up classification-based speech separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 7, pp. 1381–1390, 2013.

[7] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, "A regression approach to speech enhancement based on deep neural networks," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 23, no. 1, pp. 7–19, 2015.

[8] S. R. Park and J. Lee, "A fully convolutional neural network for speech enhancement," *arXiv preprint arXiv:1609.07132*, 2016.

[9] D. Wang and J. Chen, "Supervised speech separation based on deep learning: an overview," *arXiv preprint arXiv:1708.07524*, 2017.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[11] S. Pascual, A. Bonafonte, and J. Serr, "Segan: Speech enhancement generative adversarial network," in *Proc. Interspeech 2017*, 2017, pp. 3642–3646.

[12] D. Michelsanti and Z.-H. Tan, "Conditional generative adversarial networks for speech enhancement and noise-robust speaker verification," in *Proc. Interspeech 2017*, 2017, pp. 2008–2012.

[13] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "An algorithm for intelligibility prediction of time–frequency weighted noisy speech," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, pp. 2125–2136, 2011.

[14] A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra, "Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*. IEEE, 2001, vol. 2, pp. 749–752.

[15] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arXiv preprint arXiv:1611.07004*, 2016.

[16] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2536–2544.

[17] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[18] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[19] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," *arXiv preprint ArXiv:1611.04076*, 2016.

[20] Y. Wang, A. Narayanan, and D. Wang, "On training targets for supervised speech separation," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 22, no. 12, pp. 1849–1858, 2014.

[21] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1," *NASA STI/Recon technical report n*, vol. 93, 1993.

[22] A. Varga and H. J. Steeneken, "Assessment for automatic speech recognition: Ii. noisex-92: A database and an experiment to study the effect of additive noise on speech recognition systems," *Speech communication*, vol. 12, no. 3, pp. 247–251, 1993.

[23] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[24] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems*, 2016, pp. 2234–2242.

[25] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.