DATA AUGMENTATION FOR LOW RESOURCE SENTIMENT ANALYSIS USING GENERATIVE ADVERSARIAL NETWORKS

Rahul Gupta

Amazon.com, USA

ABSTRACT

Sentiment analysis is a task that may suffer from a lack of data in certain cases, as the datasets are often generated and annotated by humans. In cases where data is inadequate for training discriminative models, generate models may aid training via data augmentation. Generative Adversarial Networks (GANs) are one such model that has advanced the state of the art in several tasks, including as image and text generation. In this paper, I train GAN models on low resource datasets, then use them for the purpose of data augmentation towards improving sentiment classifier generalization. Given the constraints of limited data, I explore various techniques to train the GAN models. I also present an analysis of the quality of generated GAN data as more training data for the GAN is made available. In this analysis, the generated data is evaluated as a test set (against a model trained on real data points) as well as a training set to train classification models. Finally, I also conduct a visual analysis by projecting the generated and the real data into a two-dimensional space using the t-Distributed Stochastic Neighbor Embedding (t-SNE) method.

Index Terms— Generative Adversarial Networks, sentiment analysis

1. INTRODUCTION

Since their introduction, Generative Adversarial Networks (GANs) [1] have established themselves as powerful models, outperforming other generative models in the tasks such as image and text generation [3, 2]. Variations of GANs, such as Wasserstein GAN [4], coupled GAN [5] and StackGAN [6], have been proposed to improve upon the GAN architecture for various tasks. Recently, the application of GANs has also been extended to affective computing [7] with applications to emotion transformation in images [8], image and video generation with emotional attributes [9] and emotional data augmentation [10]. In this work, I apply GAN models to another critical task: low resource sentiment analysis. Although several sentiment analysis tasks come with sufficient amount of data to train complex low-error models, large quantities of data may not be available on newly formed sentiment analysis tasks or other tasks with constrained resources. In order to improve the classification performance on such tasks, I experiment with GAN models as a channel to synthetically generate additional data-points. To achieve this, I propose a variation of the conditional GAN (cGAN) model [1] to generate data on low resource datasets. I conduct further analysis to understand value added by appending the cGAN generated data.

Previous Work: Sentiment analysis [11] is a classical problem to evaluate the affective states associated with a human opinion or a reaction given to a given event. Application of machine learning techniques for sentiment analysis is a widely researched field. Survey articles such as ones by Liu et al. [12] and Medhat et al. [13] provide a summary of such techniques. Sentiment analysis has been previously studied in low resource settings by application of methods such as transfer learning [14] and semi-supervised learning [15]. The set of techniques proposed for sentiment analysis in absence of labeled data include manifold regularization [14], semi-supervised recursive autoencoders [16], document word co-regularization [17] and latent variable models [18]. On the other hand, GAN models were proposed in 2014 and a tutorial by Goodfellow [19] provides a background on GANs. Jing et al. [7] provide a summary on the application of adversarial Training in Affective Computing and Sentiment Analysis. I pursue a data augmentation approach to aid sentiment analysis in this paper. Data augmentation through GANs has been used in other tasks such as enhancing emotion recognition through speech [20, 21], human pose estimation [22] and medical image synthesis [23]. In my work, I use a variant of cGAN and apply various heuristics to achieve their convergence. This is the first work that evaluates and analyzes GAN models as data augmentation tools for sentiment analysis.

In order to train cGAN models that can augment data samples for a given low resource task at hand, I propose a variant of cGANs that also uses a baseline classifier trained on the task of interest. I apply several other tricks to obtain convergence on the cGAN model training (such as model pre-training, noise addition to the inputs and one sided label smoothing). I evaluate the thus trained cGAN models on two different low resource sentiment analysis tasks: (i) movie review and, (ii) product review. Although, I observe that the data generated using such cGAN models can by itself be used to train discriminative models, they provide a marginal gain when appended to the rest of the available real data. On the movie and product review datasets, I obtain relative improvements of 1.6% and 1.7% (respectively) against a baseline model only using the real data. I conduct further analysis on the improvements yielded by data augmentation as more data is made available for training the cGAN model. To this end, I use a larger publicly available sentiment analysis data from a social media platform. I train cGAN models by incrementing data available during training and observe the gains in sentiment classification. Finally, I also perform a data distribution analysis (using t-SNE embedding), comparing the distribution of cGAN generated data against real dataset. In the next section, I describe the cGAN architecture used in my experiments, followed by a description of the low resource datasets.

2. CONDITIONAL GAN SETUP FOR DATA AUGMENTATION

In this section, I describe the setup for using cGANs in augmenting data for sentiment classification. First, I describe the architecture of the cGAN models trained on low resource sentiment analysis datasets. Followed by this, I describe the training methodology and tricks for the cGAN architecture.

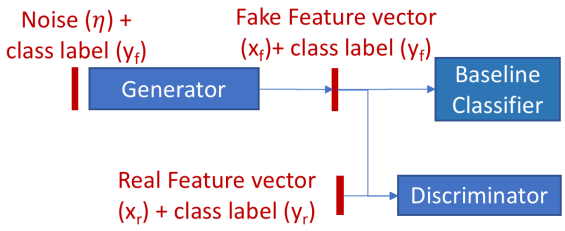

Fig. 1. cGAN architecture and notations used in my experiments. The generator parameters are updated based on signals from discriminator and the baseline classifier.

2.1. cGAN architecture

I use a cGAN architecture as shown in Figure 1, inspired by similar cGAN variants [24] except for the addition of a previously available baseline classifier (along with the generator and discriminator components). Below, I briefly describe the three cGAN components.

Generator: The generator is a feed-forward neural network with inputs as a noise vector η along with a one hot encoding of sentiment class $\mathbf{y_f}$ (e.g., in a binary sentiment classification task $\mathbf{y_f}$ would be a two dimensional vector). Given these inputs, the generator is expected to produce a feature vector $\mathbf{x_f}$ (a Doc2Vec representation [25] of sentences in my experiments), which corresponds to the class encoding $\mathbf{y_f}$ input to the generator.

Discriminator: Given pairs of real feature vectors $\mathbf{x_r}$ and the associated class vector $\mathbf{y_r}$ (again in the form of a one hot encoding) along with the fake feature vectors $\mathbf{x_f}$ produced by the generator as per the class encodings $\mathbf{y_f}$, the discriminator attempts to classify fake pairs $[\boldsymbol{x}_f, \boldsymbol{y}_f]$ against the real pairs of data-points $[\boldsymbol{x}_r, \boldsymbol{y}_r]$. In my experiments, the discriminator is also a feed-forward neural network which accepts inputs as feature vectors concatenated with associated class embeddings and outputs the probabilities of samples being real.

Baseline classifier: Apart from generator and discriminator as in a standard cGAN, I also append a baseline classifier trained previously on the dataset available for the task at hand. In low resource settings, this classifier is to be trained on the limited amount of data available. Hence, I chose a shallow neural network as the architecture for the baseline classifier, as it contains fewer parameters to be optimized. The objective of appending a baseline classifier in my cGAN architecture is, given an input class encoding $\mathbf{y_f}$, the generator is encouraged to produce a corresponding sample $\mathbf{x_f}$ with high confidence as is adjudged by the baseline classifier. I describe the cGAN training loss functions and the tricks I use to train the cGAN model in more details below.

2.2. Training cGAN model

The cGAN model is trained using an algorithm where discriminator and generator parameters are tuned alternately and iteratively. I do not update the baseline classifier parameter during cGAN training. During an iteration, discriminator parameters are tuned to minimize the cross entropy loss (L_D) as defined in equation 1. $[\boldsymbol{x}_f; \boldsymbol{y}_f]$ represents a vector concatenation of $\boldsymbol{x}_f$ and the associated label representation $\boldsymbol{y}_f$. $D([\boldsymbol{x}_r; \boldsymbol{y}_r])$ and $D([\boldsymbol{x}_f; \boldsymbol{y}_f])$ are the probabilities assigned by the discriminator to the pairs $[\boldsymbol{x}_r, \boldsymbol{y}_r]$ and $[\boldsymbol{x}_f, \boldsymbol{y}_f]$ being real, respectively. y is 1 for real data-points $\boldsymbol{x}_r$ and 0 for the fake data-points $\boldsymbol{x}_f$.

$$L_D = -y \log(D([\boldsymbol{x}_r; \boldsymbol{y}_r])) - (1-y) \log(1 - D([\boldsymbol{x}_f; \boldsymbol{y}_r])) \quad (1)$$

After updating the discriminator parameters, I tune the generator parameters using the loss function L_G in equation 2. Apart from the standard generator loss L_{G1} to fool the discriminator, I also add another cross entropy loss L_{G2} between the class prediction returned by the baseline classifier for $\mathbf{x_f}$ with respect to the expected class label $\mathbf{y_f}$. The objective of this part of the loss function is to encourage the generator to produce $\mathbf{x_f}$ such that the baseline classifier is in agreement with the association between $\mathbf{x_f}$ and $\mathbf{y_f}$.

$$L_G = L_{G1} + \lambda L_{G2} \quad (2)$$

$$\text{Where, } \begin{aligned} L_{G1} &= -\log(D([\boldsymbol{x}_f; \boldsymbol{y}_f])); \boldsymbol{x}_f = G(\boldsymbol{\eta}) \\ L_{G2} &= -\text{CE}(\boldsymbol{y}_f, C(\boldsymbol{x}_f)) \end{aligned} \quad (3)$$

In equation 3, $G(\boldsymbol{\eta})$ represents the output ($\boldsymbol{x}_f$) yielded by the generator when the input is a noise vector $\boldsymbol{\eta}$. $C(\boldsymbol{x}_f)$ is the prediction probabilities returned by the baseline classifier on the sample $\boldsymbol{x}_f$ for the sentiment classes and $\text{CE}(\boldsymbol{y}_f, C(\boldsymbol{x}_f))$ is the associated cross-entropy. λ is a hyper-parameter to tune relative weights between L_{G1} and L_{G2}. I note that the generator from a trained cGAN does implicitly learn the relationship between $\mathbf{x_f}$ and $\mathbf{y_f}$. The generator is targeted to produce $\mathbf{x_f}$ and $\mathbf{y_f}$ pairs such that the discriminator can not distinguish the fake pairs against a real pair of $\mathbf{x_r}$ and $\mathbf{y_r}$. I hypothesize that the explicit addition of L_{G2} helps the generator to generate fake pairs for which it is more confident of the association between $\mathbf{y_f}$ and $\mathbf{x_f}$, particularly in a low resource setting. Apart from the addition of the baseline classifier, I also use tricks such as addition of data points from an external dataset, noise addition to the inputs as well as one sided label smoothing to achieve better convergence of the cGAN losses. I briefly describe these tricks below.

Initialization with other dataset: Since my target tasks come with a limited amount of training data, I pre-train the cGAN model on a larger external dataset from a related task. The large dataset helps the cGAN model to converge to relatively stable parameters, which can then be fine tuned on the smaller dataset available for the task at hand. Since I use an external dataset for pre-training the cGAN models, I hypothesize that increasing λ towards the final few iterations of cGAN optimization can help the generator to tune better to the low resource dataset.

Noise addition: After pre-training the cGAN model, I add a Gaussian random noise to the in-domain real input feature vectors $\mathbf{x_f}$ for each iteration of cGAN training. The injected Gaussian noise carries a zero mean and a diagonal covariance matrix with values 0.02. Adding noise to inputs is another regularization method for GANs [19] and prevents the cGAN from over-fitting to the smaller dataset.

One sided label smoothing: Another trick that I found particularly useful in my experiments was one sided label smoothing [26]. It also acts as a regularizer and prevents providing large gradients to the generator. Hence, the generator parameters do not differ by a large value after pre-training.

Apart from the above tricks, I use batch normalization and randomly training the generator multiple times in each cGAN training

iteration to achieve better quality fake samples. The training batch is normalized to carry zero mean and unit variance per feature dimension (this implies that the noise added to $\boldsymbol{x}_r$ carries a strength of 2%, against the signal strength). In the next section, I describe the datasets I use in my experiments.

3. DATASETS

My experiments are primarily geared towards two sentiment classification tasks: (i) Movie review and, (ii) Product review. The datasets for the two tasks are described in more detail below:

Movie review dataset: The first dataset used to train cGAN models for my experiments is the movie review dataset [27]. Each review in the dataset comprises of multiple sentences, along with an associated positive/negative sentiment annotation. The dataset consists of $\sim$2k samples and I perform a 50:50 random split to define training and testing portions on the dataset.

Product review dataset: The product review dataset [28] consists of reviews on Amazon, yelp or IMDB. The dataset consists of $\sim$3k samples, annotated with positive/negative sentiment label. I split the dataset into training and testing portions of equal size.

The cGAN model (as well as the baseline classifier used in the cGAN model) for each dataset is trained using the associated training split. I conduct a classification evaluation on the testing portion, as described in the next section. I use the Twitter dataset [29] consisting of $\sim$1.6M tweets to pre-train the cGAN model. Each tweet is annotated with a positive/negative sentiment, also yielding a two dimensional $\boldsymbol{y}_r$ for pre-training. The feature representation $\mathbf{x_f}$ used in my experiments is a Doc2Vec [25] representation, trained on the Wikipedia corpus. I note that the cGAN models trained in my experiments are geared towards mimicking the Doc2Vec representations of the sentences, as opposed to more recent efforts around training GANs to obtain sentences themselves [3]. This is an avenue I will consider for future research. The feature and label representations $\mathbf{x_f}, \mathbf{y_f}$ are common in between the two primary datasets for investigation and the Twitter dataset. This allows for a pre-training followed by fine-tuning without altering the cGAN architecture.

4. EXPERIMENTAL SETUP

I initially train separate cGAN models for movie and review datasets, using the scheme described in section 2.2. Once trained, I generate 10k fake pairs of feature samples ($\mathbf{x_f}$) and class vectors ($\mathbf{y_f}$) from the cGAN generator. This fake data is then used to train another classifier C_f.

Apart from the baseline classifier C_b and classifier trained on the cGAN data C_f, I also train a classifier C_t on the twitter dataset. Through C_t, I aim to estimate the performance obtained using a transfer learning approach. In the next section, I describe my evaluation methodology.

4.1. Results

For each of the three classifiers C_b, C_f, C_t, I report the accuracy achieved on the test partition of the target datasets. I also report accuracies on a bagged approach, where I combine confidence scores from either a subset of or all three classifiers before the class assignment. The combination weight is computed based on a small held-out set from the training portion (we just need to tune three weights, so a small validation carve out is sufficient). Table 1 reports the accuracies.

Table 1. Classification accuracies obtained on the test partitions of movie review and UCI dataset using the various classifiers. Performance yielded by C_f are significantly better than chance (using a binomial proportions test at $<$5% significance level.)

Classifiers used	UCI dataset	Movie review
Chance	50.0	50.0
C_b	63.3	72.0
C_f	56.3	55.7
C_t	63.9	62.6
C_b, C_f	64.3	73.2
C_b, C_f, C_t	64.5	74.0

From the results, I observe that the accuracies yielded by the classifier C_f trained on the cGAN generated data is significantly higher than chance. This implies that the model trained on the fake data carries discriminative power. Although the C_f accuracies are not as high as the classifier C_b, when bagged together, I obtain better accuracies on both the datasets. I also observe that the accuracies yielded by the classifier C_t to be higher than chance, indicating that a transfer of knowledge from the twitter dataset is possible in the other two datasets of interest. Finally, the best accuracies are obtained by bagging across all the three classifiers.

5. ANALYSIS ON THE DATA GENERATED USING CGAN MODEL

Results in the previous section show that the data generated using cGAN models provide marginal improvements in classification performance in low resource settings. I perform further analysis to assess the impact of data size in training cGAN models, as well as a visual assessment of the generated fake data. Using the twitter dataset, I perform two sets of analysis: (i) evaluating the impact of the training dataset size on the cGAN model and, (ii) a data distribution analysis using t-SNE projections. I discuss them in detail below.

5.1. Evaluating the impact of the dataset size on cGANs

I evaluate the impact of the size of the available dataset on the cGAN model in this section. The quality of the cGAN data generated is evaluated in two settings: when used as a training set as well as a test set. The setup of my analysis is as follows. Initially, I segment the twitter dataset into a training (600k samples) and test set (1M samples). I train a neural network classifier C_b^{full} on the entire training dataset. In addition, a cGAN model is trained on a subset of N samples from the training set ($N = 500, 1k, 2k, 4k, 8k$). I sample 10k fake feature and class vectors pairs $(\boldsymbol{x}_f, \boldsymbol{y}_f)$ from the trained cGAN and use them as training and testing sets as described below.

As a test set: In this setting, I use the fake data pairs $\boldsymbol{x}_f$ and $\boldsymbol{y}_f$ generated using cGAN as a test set. This setting is same as the one during cGAN training where the generated data is fed to the baseline classifier to obtain the cross-entropy loss L_{G2}. For a given $\boldsymbol{x}_f$, I treat $\boldsymbol{y}_f$ as the ground truth. Given a set of N training samples, I train the cGAN model described in section 2.2. The baseline classifier used in the cGAN models is also trained on the available set of N training samples. The classifier C_b^{full} trained on the entire training set is then evaluated on the generated fake data. A better accuracy on the fake data as test set implies a better trained cGAN as the loss L_{G2} in equation 2 is expected to be lower. However, it may not directly imply a higher quality of the generated data when used to augment sparse datasets. The accuracies on the fake samples as I increase N for cGAN training is shown in Figure 2.

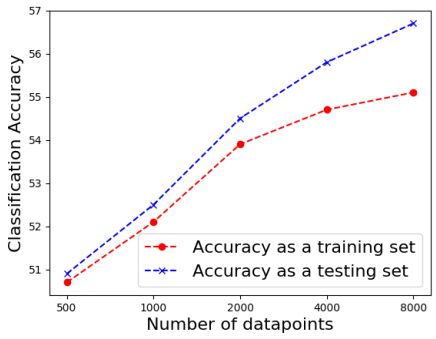

Fig. 2. Plot showing accuracies on the generated data when used as a test set as well as accuracy yielded by a model trained using the generated data on a real test set.

As a training set: In this setting, I use data generated using cGAN model as a training dataset (akin to the experiment in Section 4). I train a shallow NN on the fake data with $\boldsymbol{x}_f$ as input features and $\boldsymbol{y}_f$ as target labels. The NN is then evaluated on the real test set of $\sim$1M samples. The accuracy yielded by this NN on the real test set reflects the direct value of the generated data in enhancing classification through data augmentation. Figure 2 shows the accuracy achieved on the test set as I increase N for the cGAN training.

Figure 2 suggests that the accuracy of the fake data as test set is higher than when used as a training set for all values of N. Moreover, as N increases, accuracy on the fake data as test set increases more rapidly than when it is used as a training set. This implies that as N increases, the loss L_{G2} is expected to decrease and the overall loss value for cGAN is expected to be lower. However, performance due to data augmentation does not increase commensurately. Also, the figure suggests that the gain yielded by data augmentation saturates as N increases. This hints towards an upper limit to the gains yielded by the cGAN model. Overall, I observe that there is a discrepancy in the expected loss decrease in cGAN model training and the expected increase in performance due to data augmentation. I conduct further analysis on the cGAN generated data using t-SNE and observe plausible reasons for this discrepancy.

5.2. Analysis of the generated dataset using t-SNE

In order to further understand the distribution of the data generated using cGAN models, I project the data into a lower dimension for visual analysis. Given data-points from the real dataset and fake data generated using the cGAN, I use the t-SNE method to project the data into a two dimensional subspace. I sub-sample 2k data-points from real dataset as well as fake data generated by the cGAN model trained using 8k samples in the previous section. The t-SNE projection parameters are obtained on the combination of the fake and real data-points and Figure 3 shows the distribution of fake and real data-points. I observe that while the fake data overlaps with the real data points, it does not completely cover the subspace spanned by real dataset. This indicates that cGAN is generating data-points in a limited region of the actual distribution of real data-points. Despite the application of tricks mentioned in Section 2.2, I was not able to obtain a fake data distribution that completely resembles the distribution of real data. This problem is akin to mode collapse in GAN models, where a GAN model tends to produce data samples in a limited region of the feature space.

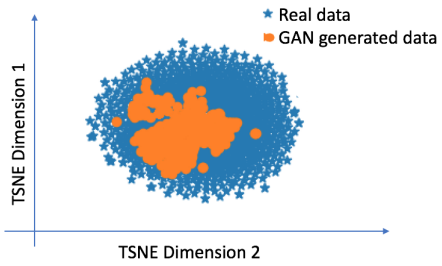

Fig. 3. Real and fake data distribution, as observed on a 2-D projection of data-points obtained using the t-SNE method.

It is also plausible that the discrepancy in using the fake data as training and testing sets in Section 5.1 could be explained by this difference in the distribution of real and fake data. While fake data may lie in a region of feature space that could be well classified by the baseline classifier, a classifier trained on the fake data can not correctly classify test instances drawn from a region not covered by the fake samples. I expect that addressing the discrepancy in the coverage of data distribution can lead to further improvement of classification. I aim to address this as a part of future research.

6. CONCLUSION

GANs are powerful generative models that have shown state of the art performance in several tasks related to image and text generation. In this work, I use a variant of cGAN models to augment training data for low resource sentiment tasks. Specifically, given Doc2Vec representation of sentences on datasets with a few training samples, I train a cGAN model using tricks such as pre-training and noise injection. Followed by this, I generate fake feature vectors from the generator along with the associated sentiment class labels. Empirically, I observe that augmenting classification with a model trained on the fake data provides gains in the low resource sentiment analysis tasks. Further analysis shows that as the number of real samples for cGAN training increase, a baseline classifier trained on real data yields better accuracy in classifying the fake data. However, this does not translate to an equivalent increase in performance on real test instances when a classifier is trained on the fake data. Furthermore, I observe that fake data does not cover the entire region of feature space as occupied by real data using the t-SNE analysis.

In the future, I aim to address the observations made in the analysis of fake data distribution. I aim to consider other techniques to obtain cGAN models that does not suffer from selective data generation limited to a smaller region of the feature space (as compared to the space spanned by real data). Other GAN models that directly generate sentences are another attractive avenue for such a data augmentation. The results of such as experiment would be more interpretable as one can directly observe the generated sentences. Finally, other variants of GAN model can be applied to the problem of sentiment analysis as well its extensions for this kind of data augmentation.

7. REFERENCES

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[2] Emily L Denton, Soumith Chintala, Rob Fergus, et al., "Deep generative image models using a laplacian pyramid of adversarial networks," in *Advances in neural information processing systems*, 2015, pp. 1486–1494.

[3] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu, "Seqgan: Sequence generative adversarial nets with policy gradient.," in *AAAI*, 2017, pp. 2852–2858.

[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou, "Wasserstein generative adversarial networks," in *International Conference on Machine Learning*, 2017, pp. 214–223.

[5] Ming-Yu Liu and Oncel Tuzel, "Coupled generative adversarial networks," in *Advances in neural information processing systems*, 2016, pp. 469–477.

[6] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris Metaxas, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," *arXiv preprint*, 2017.

[7] Jing Han, Zixing Zhang, Nicholas Cummins, and Björn Schuller, "Adversarial training in affective computing and sentiment analysis: Recent advances and perspectives," *arXiv preprint arXiv:1809.08927*, 2018.

[8] Hui Ding, Kumar Sricharan, and Rama Chellappa, "Exprgan: Facial expression editing with controllable expression intensity," *arXiv preprint arXiv:1709.03842*, 2017.

[9] Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, and Gang Hua, "Towards open-set identity preserving face synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6713–6722.

[10] Hai X Pham, Yuting Wang, and Vladimir Pavlovic, "Generative adversarial talking head: Bringing portraits to life with a weakly supervised neural network," *arXiv preprint arXiv:1803.07716*, 2018.

[11] Bo Pang, Lillian Lee, et al., "Opinion mining and sentiment analysis," *Foundations and Trends® in Information Retrieval*, vol. 2, no. 1–2, pp. 1–135, 2008.

[12] Bing Liu and Lei Zhang, "A survey of opinion mining and sentiment analysis," in *Mining text data*, pp. 415–463. Springer, 2012.

[13] Walaa Medhat, Ahmed Hassan, and Hoda Korashy, "Sentiment analysis algorithms and applications: A survey," *Ain Shams Engineering Journal*, vol. 5, no. 4, pp. 1093–1113, 2014.

[14] Rahul Gupta, Saurabh Sahu, Carol Espy-Wilson, and Shrikanth Narayanan, "Semi-supervised and transfer learning approaches for low resource sentiment classification," *arXiv preprint arXiv:1806.02863*, 2018.

[15] Andrew B Goldberg and Xiaojin Zhu, "Seeing stars when there aren't many stars: graph-based semi-supervised learning for sentiment categorization," in *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*. Association for Computational Linguistics, 2006, pp. 45–52.

[16] Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning, "Semi-supervised recursive autoencoders for predicting sentiment distributions," in *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, 2011, pp. 151–161.

[17] Vikas Sindhwani and Prem Melville, "Document-word co-regularization for semi-supervised sentiment analysis," in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 2008, pp. 1025–1030.

[18] Oscar Täckström and Ryan McDonald, "Semi-supervised latent variable models for sentence-level sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*. Association for Computational Linguistics, 2011, pp. 569–574.

[19] Ian Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[20] Saurabh Sahu, Rahul Gupta, and Carol Espy-Wilson, "On enhancing speech emotion recognition using generative adversarial networks," *arXiv preprint arXiv:1806.06626*, 2018.

[21] Saurabh Sahu, Rahul Gupta, Ganesh Sivaraman, Wael AbdAlmageed, and Carol Espy-Wilson, "Adversarial auto-encoders for speech based emotion recognition," *arXiv preprint arXiv:1806.02146*, 2018.

[22] Xi Peng, Zhiqiang Tang, Fei Yang, Rogerio S Feris, and Dimitris Metaxas, "Jointly optimize data augmentation and network training: Adversarial data augmentation in human pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2226–2234.

[23] Hoo-Chang Shin, Neil A Tenenholtz, Jameson K Rogers, Christopher G Schwarz, Matthew L Senjem, Jeffrey L Gunter, Katherine P Andriole, and Mark Michalski, "Medical image synthesis for data augmentation and anonymization using generative adversarial networks," in *International Workshop on Simulation and Synthesis in Medical Imaging*. Springer, 2018, pp. 1–11.

[24] Emily Denton, Sam Gross, and Rob Fergus, "Semi-supervised learning with context-conditional generative adversarial networks," *arXiv preprint arXiv:1611.06430*, 2016.

[25] Andrew M Dai, Christopher Olah, and Quoc V Le, "Document embedding with paragraph vectors," *arXiv preprint arXiv:1507.07998*, 2015.

[26] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems*, 2016, pp. 2234–2242.

[27] Bo Pang and Lillian Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," in *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2004, p. 271.

[28] Dimitrios Kotzias, Misha Denil, Nando De Freitas, and Padhraic Smyth, "From group to individual labels using deep features," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 597–606.

[29] Alec Go, Richa Bhayani, and Lei Huang, "Twitter sentiment classification using distant supervision," *CS224N Project Report, Stanford*, vol. 1, no. 12, 2009.