A Generative-Adversarial Approach to Low-Resource Language Translation via Data Augmentation

Linda Zeng[1]

***Abstract*—Language and culture preservation are serious challenges, both socially and technologically. In response to this issue, this paper takes a data-augmenting approach to low-resource machine translation, helping to diversify the field and preserve underrepresented cultures. Since low-resource languages, such as Aymara and Quechua, do not have many available translations that machine learning software can use as a reference, machine translation models frequently make errors when translating to and from low-resource languages. Because models learn the syntactic and lexical patterns underlying translations through processing the training data, insufficient amount of data hinders them from producing accurate translations. In this paper, I propose the novel application of a generative-adversarial network (GAN) to automatically augment low-resource language data. A GAN consists of two competing models, one learning to generate sentences from noise and the other interpreting whether a given sentence is real or generated. The paper shows that even when training on a very small amount of language data (< 20,000 sentences) in a simulated low-resource setting, such a model is able to generate original, coherent sentences, such as "ask me that healthy lunch im cooking up," and "my grandfather work harder than your grandfather before." This GAN architecture is effective in augmenting low-resource language data to improve the accuracy of machine translation and provides a reference for future experimentation with GANs.**

I. Introduction

Languages play a vital role in preserving cultural beliefs and traditions. As technology becomes increasingly prevalent, it is crucial for translation software to reflect the world's linguistic diversity, ensuring that valuable cultural identities are not left behind. However, current state-of-the-art translation models frequently make mistakes when translating to and from "low-resource languages" [1], or languages that do not have enough digital data that machine learning algorithms can use as reference. For example, many American indigenous languages, such as Aymara and Quechua [2], are underrepresented in datasets, resulting in models trained on them generating incorrect translations.

Previous approaches have focused on bridging the gaps between high-resource and low-resource languages through transfer-learning and cross-lingual pretraining [1]–[3], which have limited efficacy depending on the similarity between the high-resource and low-resource languages being used. The direction of data augmentation focusing solely on low-resource language generation has not been fully explored and holds promise for breakthrough [4].

Artificially increasing the size of datasets with original, generated samples allows translation models to receive more variety and volume in training, improving their generalizability. Both monolingual and parallel data augmentation is important for Neural Machine Translation (NMT), which refers to translation powered by neural networks. Training on monolingual corpora in addition to parallel data has been used to improve NMT models [5]–[7], especially in low-resource NMT [8]. As a result, this paper introduces a system for monolingual data augmentation.

In contrast to the human labor of creating new sentences in low-resource languages by hand, a generative-adversarial network (GAN) is capable of autonomously generating as much new data in a low-resource language as needed. Previous studies have implemented GANs for machine translation [9-12], but at the time of our research, no previous models have used them for translation of low-resource languages.

In this study, I propose applying a GAN to monolingual low-resource language data augmentation to improve translation quality. My model generates new synthetic language data for low-resource languages and is the first to combine GANs, data augmentation, and low-resource NMT.

II. Model Architecture

My design consists of two models: an encoder-decoder and a GAN, which contains a generator and a discriminator. Figure 1 displays a preliminary example of the model workflow.

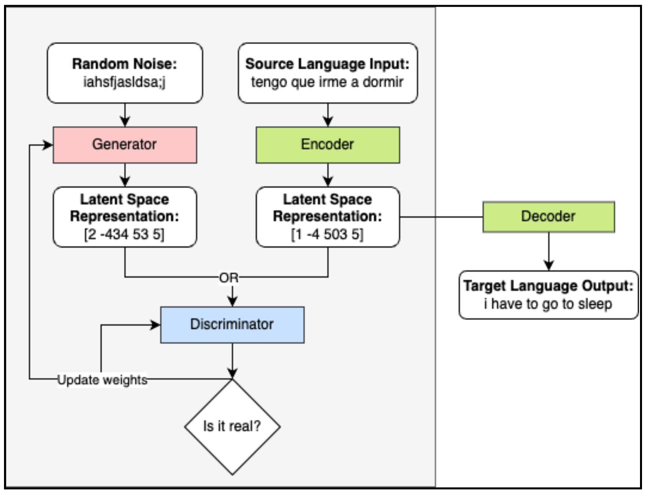

Figure 1. *Overall workflow with basic examples.*

First, the encoder-decoder learns to translate from a source language to the target language using the training data. In Figure 1, the source language is Spanish, and the data point is "tengo que irme a dormir." The encoder transforms the source language into a latent space representation, an abstract representation of the sentence's meaning, and then the decoder "decodes" the representation

[1]L.Z. is with The Harker School, 500 Saratoga Ave, San Jose, CA 95129 (corresponding author to email: 26lindaz@students.harker.org).

into the target language—in this case, “i have to go to sleep” in English. Similar sentences are positioned closer together in this internal latent space, which reflects samples’ external meanings regardless of language. After the encoder-decoder finishes training, a batch of sentences in the source language from the dataset are fed into the trained encoder, and generates latent space representations. Figure 1 shows a simplified version of the representation: [1 -4 503 5].

Next, we train the GAN using the encoder-decoder as a tool, as shown in the left box in Figure 1. The generator takes in a batch of random noise, a list of randomly generated numbers between -1 to 1 which, by themselves, have no meaning (the figure represents noise as gibberish). Then, the generator tries to create sentence meanings out of the noise by transforming them into latent space representations, shown as [2 -434 53 5] in Figure 1.

The discriminator receives either the encoder’s or the generator’s latent space representation, without knowing from which model the representation comes from. The discriminator predicts whether a given encoding is real (from the encoder) or fake (from the generator), and then compares its prediction with the encoding’s real label to evaluate its own performance. Depending on the discriminator’s success or failure, the generator also adjusts its weights and learns to create encodings more similar to the encoder’s in order to fool the discriminator. This process continues until ultimately, through trial-and-error, the generator learns to generate latent space representations so similar to the real ones that the discriminator is not able to tell them apart. As a result, the generator is able to generate unlimited new latent space representations from only noise and no extra data. Like the encoder’s encodings, these representations can then be decoded by the decoder into real sentences in the target language.

Figure 2 displays the neural-network architectures underlying the encoder-decoder, generator, and discriminator.

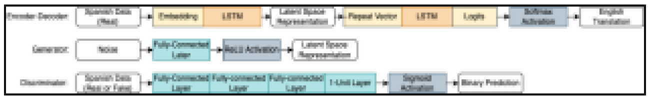

Figure 2. *Generator, discriminator, and encoder-decoder architectures.*

The encoder-decoder uses a long-short term memory (LSTM) network for both the encoder and decoder. Long-short term memory (LSTM) networks [13] are a special type of neural networks that learn when to remember information that may be important later on in a data sequence. In NMT, since related words may be placed far from each other in a sentence, encoder-decoders commonly use LSTMs because LSTMs can capture the long-term dependencies between such words.

In between the major LSTM layers of the encoder-decoder, the embedding layer learns to map words with analogous meanings to similar numerical vectors, the repeat vector copies each latent space representation into the decoder, and the logits layer maps’ numerical outputs into probabilities. Last, a softmax activation function normalizes the probabilities, producing the output.

The generator consists of a dense layer, which is a layer of fully connected units, and a ReLU activation function, which is commonly used in GANs to optimize efficiency and capture complex data distributions. The generator’s main purpose is to learn through trial and error to mimic the encoder’s encodings despite only being given noise as input.

The discriminator consists of three dense layers and then a one-unit dense layer, which allows it to produce a prediction of whether the input is from the encoder or the generator. Each of these hidden layers include a ReLU activation function to capture complexity. Finally, the model uses a sigmoid activation function to categorize its prediction into a 1 (for encoder) or 0 (for generator). Using trial-and-error, the discriminator learns how to differentiate encodings of natural source language data (from the encoder) from synthetic encodings (from the generator).

The combined GAN model feeds its noise input to the generator and the generator’s encoding output into the discriminator. The discriminator’s prediction is used to update the weights for both models. Ultimately, once both models optimize their performances, the generator’s outputs serve as novel data points for data augmentation.

III. Data

The data used in this study was derived from a Tatoeba dataset [14] processed by a third-party [15]. The training, validation and test data were given only to the encoder-decoder while the GAN operated solely on noise and the encoder-decoder’s output.

To mimic the characteristics of low-resource languages using English and Spanish, I reduced the amount of data from 253,726 sentence pairs to only 20,000. Of the 20,000, 18,000 sentence pairs were used to train the encoder-decoder, 1,000 were kept as validation data to verify accuracy and prevent overfitting during the training process, and the last 1,000 sentence pairs served as test data and were not seen or used until after training and hyperparameter tuning for the encoder-decoder. I retained only words, lowercase, and no punctuation. Using Keras’s built-in Tokenizer [16], I then split each sentence into a list of probabilities. I found the longest sentence and padded the rest with zeros to the same length.

IV. Experimental Setting

This section describes conditions, procedures, hyperparameters, and observations relevant to training.

I imported all of the model layers from the Python Keras library [16]. Through experimentation, I chose the encoder-decoder’s two LSTM layers to contain 64 units. To minimize overfitting, I included weight decay, which shrinks the weights of neural networks, and dropout, which drops out a few randomly selected neurons during training. These values were empirically determined for each layer in the encoder-decoder. For the encoder LSTM, I used a type of weight decay known as L2 regularization with a value of 5e-5, and a dropout of 0.5. For the decoder LSTM, I used an L2 regularization of 1e-5 and a dropout of 0.5. The model was compiled with a categorical cross entropy loss function and an Adam optimizer [17] with a learning rate of 2e-3 and weight decay rates of beta1 0.7 and beta2 0.97. The encoder-decoder trained in batches of 30 samples for 400 epochs on the training data.

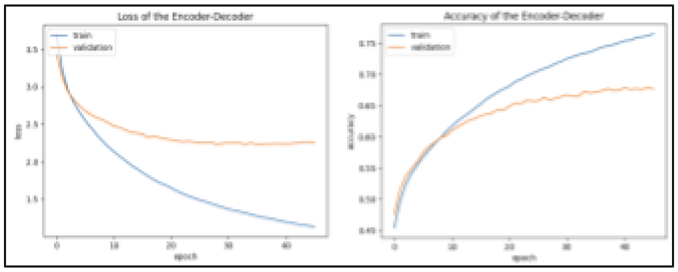

Figure 3. *Loss and accuracy of the encoder-decoder.*

While training, the encoder-decoder reached an accuracy of 92.8%, and a peak accuracy of 71.4% on validation data. Figure 3 shows the progression of training and validation loss and accuracy through epochs.

The generator's dense layer contained 64 units. The model was compiled with a categorical cross entropy loss function and an Adam optimizer with learning rate 4e-4. The discriminator's three dense layers each had 1,024 units, followed by a single-unit layer that represented its prediction. The model was compiled with a binary cross entropy loss function and an Adam optimizer with a learning rate of 1e-4. The combined GAN compiled with a binary cross entropy loss function and an Adam optimizer with a learning rate of 1e-4. With a batch size of 1900, the GAN trained across 8,000 epochs.

Shown in Figure 4, while training, the GAN's loss values plateaued for both the generator and the discriminator, indicating that the models reached convergence and were both performing optimally against each other. Its final loss values hovered around 0.581 for the generator and 0.438 for the discriminator.

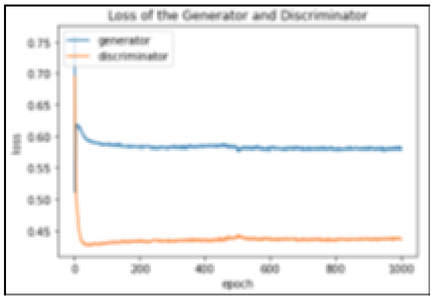

Figure 4. *Loss of the GAN.*

Once the generator and discriminator were performing optimally, the decoder was run on the generator's encodings, converting their meanings into probabilities. Each probability mapped to the closest word in the tokenizer's dictionary of probability word pairs, forming sentences.

Hyperparameter Tuning

In order to find the hyperparameters for optimal results, I experimented by first varying values by a factor of either 2 or 10 and testing every combination. To optimize for time, I used 5,000 sentences and 80 epochs. For the learning rates of the encoder-decoder, generator, discriminator, and GAN, as well as the weight decay, I tried a range of values from 1e-1 to 1e-8, decreasing in magnitude by a factor of 10 each time. When varying the number of units and batch sizes for the encoder-decoder's LSTM layers, the generator's dense layer, and the discriminator's dense layers, I chose powers of 2 between 16 and 2048. For the encoder-decoder's dropouts, I tried a range from 0.5 to 0.8.

After finding the approximate values to optimize performance, I tested more specific values within the ideal range I found, isolating each of the models and incrementing values. Once reaching optimal parameters, I increased the training data and epochs to further improve the models.

V. Results

Encoder-Decoder Performance

On the test data, the encoder-decoder had a final accuracy of 69.3%. This accuracy is respectable, considering the fact that the model was trained on less than 20,000 sentences, which is less than one-tenth of the size of other low-resource language datasets, whose sizes are between 0.2 to 1 million [3][18].

As this paper focuses more on the data-augmentation aspect than the machine translation part of this research, accuracy was used instead of BLEU score [19], a metric that assesses the quality of a machine translation relative to a human-curated translation by counting overlapping n-grams. Using accuracy, the value of incomplete yet coherent translations could be considered and did not affect results as strongly as BLEU scores would have caused it to.

GAN Performance

After training, the GAN was able to successfully generate coherent sentences, such as sentences 1, 2, and 3 labeled "good" in Table 1. Generated samples generally centered around its own cohesive theme, which is an indication of the model's successful understanding of word meanings. From random noise, the generator was able to create its own completely new and logical sentences, a significant feat considering the lack of training data.

Sample Generated Sentences	
my grandfather work harder than your grandfather before	good
to consider quit job is this dream man	good
ask me that healthy lunch im cooking up	good
maryam discovered hes hes am am are are	repetition
home actually was everything everything listen actually everything	repetition
cheerful weird yourself punished music alone everybody everybody	nonsensical
those in so friends so complicated english comes	nonsensical
stressed gloves eating eating worried online online online	unrelated

Table 1. *Raw Samples by the GAN.*

Error Analysis and Future Work

The model made a series of errors, shown in lines 4–7 in Table 1. The severity of errors decreased as the GAN trained for more epochs. Thus, future models may train for a larger number of epochs to examine whether errors can be reduced further. I propose future work for three main errors: repeated words, nonsensical grammar, and unrelated words.

Repeated words occur in most MT models due to the model trying to generate words that are close in context to each other, which is necessary [20]. Once the decoder translates the latent space representations into probabilities, they may be reduced to the same word. Future work should involve training the model to remember the previous probabilities it generated and to vary them.

Some sentences are grammatically incorrect or nonsensical with randomly placed words. Due to the low-resource setting, the model did not learn enough context to understand how to place these words. For example, in Table 1, the model predicted “weird” to follow “cheerful.” The model likely assigned similar embeddings to these two words since they are both adjectives that may have been used in comparable contexts. However, while the adjectives may be used interchangeably as modifiers for a person, they cannot follow each other directly. Future work should attempt to train the model on which semantically related words can be placed together syntactically.

The model occasionally groups unrelated words together. It has not seen a word (e.g. “gloves” in Table 1) enough to understand its usual context (i.e. being put on people’s hands). Future work should focus on incorporating a dictionary of words into the model so that it better understands words’ meanings.

In the future, I plan to post-process the samples by cleaning and inserting punctuation and capitalization. I also plan to test this method’s performance in a true low-resource setting.

VI. Conclusion

This work takes strides to address data inequality in neural machine translation, using a generative-adversarial approach to augment low-resource languages. Experiments show promising results: using less than 20,000 sentences, the GAN was able to generate unlimited, coherent sentences. These new sentences can be added to the original corpus to improve the accuracy of machine translation. Improvements can be made on this research to increase comprehensiveness when evaluating the model’s performance and to minimize the repetition and incoherence in many of the generated sentences.

This innovative approach has the potential to bring about more robust language translations for minority populations. Because of this GAN architecture's ability to generate an unlimited amount of original sentences despite being trained on minimal data, it can be used as an effective tool to augment low-resource language data, allowing translation software to train on more sentences and therefore generate more accurate translations. The next steps of this research are to explore this model’s feasibility on a variety of low-resource languages from a diverse set of language families. This novel application of a GAN to low-resource language translation serves as a reference for future work that combines GANs and Natural Language Processing, specifically MT.

VII. Acknowledgement

I would like to thank my teachers, Anu Datar and Ricky Grannis-Vu, for their ongoing support and encouragement.

VIII. References

[1] J. Gu, H. Hassan, J. Devlin, and V. O. Li, “Universal neural machine translation for extremely low resource languages,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 344–354. [Online]. Available: https://aclanthology.org/N18-1032

[2] F. Zheng, M. Reid, E. Marrese-Taylor, and Y. Matsuo, “Low-resource machine translation using cross-lingual language model pretraining,” in *Proceedings of the First Workshop on Natural Language Processing for Indigenous Languages of the Americas*. Online: Association for Computational Linguistics, Jun. 2021, pp. 234–240. [Online]. Available: https://aclanthology.org/2021.americasnlp-1.26

[3] B. Zoph, D. Yuret, J. May, and K. Knight, “Transfer learning for low-resource neural machine translation,” in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 1568–1575. [Online]. Available: https://aclanthology.org/D16-1163

[4] M. Fadaee, A. Bisazza, and C. Monz, “Data augmentation for low-resource neural machine translation,” in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 567–573. [Online]. Available: https://aclanthology.org/P17-2090

[5] J. Zhang and C. Zong, “Exploiting source-side monolingual data in neural machine translation,” in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 1535–1545. [Online]. Available: https://aclanthology.org/D16-1160

[6] R. Sennrich, B. Haddow, and A. Birch, “Improving neural machine translation models with monolingual data,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 86–96. [Online]. Available: https://aclanthology.org/P16-1009

[7] D. Cai, Y. Wang, H. Li, W. Lam, and L. Liu, “Neural machine translation with monolingual translation memory,” in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 7307–7318. [Online]. Available: https://aclanthology.org/2021.acl-long.567

[8] A. Currey, A. V. Miceli Barone, and K. Heafield, “Copied monolingual data improves low-resource neural machine translation,” in *Proceedings of the Second Conference on Machine Translation*. Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 148–156. [Online]. Available: https://aclanthology.org/W17-4715

[9] Z. Yang, W. Chen, F. Wang, and B. Xu, “Improving neural machine translation with conditional sequence generative adversarial nets,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 1346–1355. [Online]. Available: https://aclanthology.org/N18-1122

[10] Z. Zhang, S. Liu, M .Li, M. Zhou, and E. Chen,“Bidirectional generative adversarial networks for neural machine translation,” in *Proceedings of the 22nd Conference on Computational Natural Language Learning*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 190–199. [Online]. Available: https://aclanthology.org/K18- 1019

[11] Z. Yang, W. Chen, F. Wang, and B. Xu, “Unsupervised neural machine translation with weight sharing,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1:*

Long Papers). Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 46–55. [Online]. Available: https://aclanthology.org/P18-1005

[12] A. Rashid, A. Do-Omri, M. A. Haidar, Q. Liu, and M. Rezagholizadeh, "Bilingual-GAN: A step towards parallel text generation," in *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 55–64. [Online]. Available: https://aclanthology.org/W19-2307

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] Tatoeba: Collection of sentences and translations. Tatoeba.org. [Online] Available: https://tatoeba.org/en/

[15] C. Kelly and L. Kelly, "Interesting Things for ESL/EFL Students (Fun English Study)," 2023. Manythings.org. [Online]. Available: https://manythings.org.

[16] F. Chollet *et al.* (2015) Keras. [Online]. Available: https://github.com/fchollet/keras

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017. [Online] Available: https://arxiv.org/abs/1412.6980

[18] S. Ranathunga, E.-S. A. Lee, M. P. Skenduli, R. Shekhar, M. Alam, and R. Kaur, "Neural machine translation for low-resource languages: A survey," 2021. [Online] Available: https://arxiv.org/abs/2106.15115

[19] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Jul. 2002, pp. 311–318. [Online]. Available: https://aclanthology.org/P02-1040

[20] Z. Fu, W. Lam, A. M.-C. So, and B. Shi, "A theoretical analysis of the repetition problem in text generation," 2021. [Online] Available: https://arxiv.org/abs/2012.14660