HIGH SCHOOL EDITION
Journal of Student Research
Volume 12 Issue 4 (2023)

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

Linda Zeng

The Harker School, USA

ABSTRACT

Language and culture preservation is a serious challenge both socially and technologically. This paper proposes a novel, data augmentation approach to using machine learning to translate low-resource languages. Since low-resource languages, such as Aymara and Quechua, do not have many available translations that machine learning software can use as reference, machine translation models frequently err when translating to and from low-resource languages. Because models learn the syntactic and lexical patterns underlying translations through processing the training data, an 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, with one learning to generate sentences from noise and the other trying to tell if a given sentence is real or generated. My experiments show 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." The first of its kind, this novel application of a GAN is effective in augmenting low-resource language data to improve the accuracy of machine translation and provides a reference for future experimentation with GANs in machine translation.

Introduction

Although technology has become a staple of daily life, even society's best-performing computing systems struggle in basic translation tasks, failing to reflect the world's diversity in languages. Current state-of-the-art translation models frequently err when translating to and from "low-resource languages" (Gu et al., 2018), which are languages that do not have much digital data that the machine learning algorithms underlying the software can use as reference.

While vast and detailed language data exists for high-resource languages such as English and Spanish, low-resource languages, including many American indigenous languages like Aymara and Quechua (Zheng et al., 2021), are underrepresented in data sets, and consequently, models trained on them often generate incorrect translations (Zoph et al., 2016).

Although other approaches have been formulated to improve on translation models or augment the low-resource data, few have truly solved the issue. Previous approaches have focused on bridging the gaps between high-resource and low resource languages (Gu et al., 2018) (Zoph et al., 2016) (Zheng et al., 2021), which have limited efficacy depending on the similarity between the high-resource and low-resource languages being used. On the other hand, the direction of data augmentation with completely original sentences has not been fully explored (Fadaee et al., 2017) and holds promise for breakthrough, as data augmentation directly addresses the challenge of lacking training data.

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. Prior work has been done with GANs in Machine Translation as the Yang et al. (2018a) model directly uses GANs for language translation, and other models (Zhang et al., 2018) (Yang et al., 2018b) (Rashid et al., 2019) have implemented its model as a tool and improved upon it. However, no previous models have used GANs to tackle the issue of translating low-resource languages.

Using a simulated low-resource setting, I propose the novel application of a GAN to low resource language data augmentation in order to improve machine translation quality.

Background

Preliminaries on NMT

In machine translation, software systems are used for translating between two different languages. Neural Machine Translation (NMT) is a type of MT that uses neural networks for translations (Yang et al., 2018a), and the most common model consists of an end-to-end encoder-decoder (Ranathunga et al., 2021). A sequence-to-sequence encoder decoder framework (Cho et al., 2014) consists of one long short-term memory (LSTM) layer (Hochreiter and Schmidhuber, 1997), that translates the words into latent space (encoding) and another LSTM layer that translates them from la tent space into the other language (decoding). The latent space represents the core meaning of the sentence, and sequence-to-sequence refers to one LSTM that feeds its sequential predictions into the second LSTM. LSTMs are a type of neural network that can operate on a sequence of words, but because they evaluate from left to right, the encoder decoder does not examine the context that appears after a word. As a result, these networks require multiple instances of words appearing in diverse contexts in order to create vectors that accurately represent the context needed for these words in the latent space (Fadaee et al., 2017), causing NMT to frequently err with low-resource pairs (Zoph et al., 2016). NMT models based on Transformers (Vaswani et al., 2017) have also risen in popularity, as the Transformer framework uses attention to improve parallelizability of training. After implementing both the RNNSearch and the Transformer on their GAN model, Yang et al. (2018a) found that both architectures could be applied with similar performances.

Data Augmentation for Low-Resource NMT

Other models and software have tried to tackle this issue, using multilingual, transfer-learning approaches (Zheng et al., 2021) (Gu et al., 2018) (Zoph et al., 2016) and a data augmentation approach (Fadaee et al., 2017). The transfer-learning approach trains the model on high-resource pairs and then transfers learning to low-resource languages, but it involves finding high resource languages that are very close to the language at hand, which is difficult depending on the language family. While the Germanic language family has more related high-resource languages such as German and Danish (Tchistiakova et al., 2021) (Chen and Abdul-Mageed, 2021), African languages have only more unrelated languages to learn from, such as English and French (Dione, 2021). The Fadaee et al. (2017) data augmentation method alters translation data containing rare words to diversify con texts, but it helps with specific words rather than generating completely new sentences.

Preliminaries on GANs

Commonly used in image generation and computer vision (Yang et al., 2018a), GANs combine two machine learning models (Goodfellow et al., 2014). The first one is called the generator, a model that takes in input and

generates samples, which are then fed into the discriminator, the second machine learning model. The discriminator is given samples either from the real data or from the generator, and it must determine if this sample is real or generated. If it correctly predicts, this indicates that it is improving. If it incorrectly predicts, this indicates that the generator is improving. Then, depending on how well the discriminator predicted, both models tweak their weights to continue this cycle. The generator "wins" when the discriminator cannot tell the two samples apart, and the discriminator "wins" if it can tell them apart. Both models want to win against the other, so each one keeps improving by learning from the other until they finally reach equilibrium, where both models are performing optimally.

GANs in NLP

In the past, the Betti et al. (2020) and Ahamad (2019) models were successful in generating synthetic text through adversarial training. The Yang et al. (2018a) study used GANs for machine translation. The generator learned to translate a source language sentence into its target-language translation while the discriminator tried to distinguish between real and generated translations. Continuing with this research, they implemented two GANs to use as tools to ensure the efficacy of their encoder models in weight sharing for latent space embeddings (Yang et al., 2018b). The Zhang et al. (2018) model introduced bidirectional GANs to improve translation quality by creating another generator model to act as the discriminator, and the Rashid et al. (2019) study used a latent space based GAN to translate bidirectionally in both a supervised and unsupervised setting.

Unexplored Area

My model also applies GANs to NMT, but I focus on improving translations for specifically low resource languages through data augmentation. This approach is the first to implement GANs for low-resource language data augmentation. In this paper, I experiment with the structure of the Yang et al. (2018a) model, but rather than directly translating between languages with a GAN, my model generates new synthetic language data for low resource languages.

Data

Simulated Low-Resource Setting

In order to investigate the GAN model's ability to augment low-resource language data, I mimicked the characteristics of low-resource languages in English and Spanish. In my English to Spanish data set, I reduced the amount of data and used only 20,000 of the data set's 253,726 sentence pairs to train and test the model. As a result, I was able to replicate the effect on low-resource languages and evaluate the model's effectiveness after it was trained on 20,000 sentences.

Training Data

The English to Spanish data set I used to train the model is from Tatoeba[1] and has been pre-processed and cleaned by a third party[2] who downloaded the original file from Tatoeba. Founded by Trang Ho in 2006, Tatoeba is a database containing collections of parallel translations contributed by thousands of members. It offers

[1] https://tatoeba.org/en/
[2] http://www.manythings.org/

data for 419 languages that are easy to download. 19,000 sentence pairs of the 20,000 in the dataset were used to train the encoder-decoder.

Table 1. Characteristics of Training Data

Characteristic	English	Spanish
Average Sentence Length	4.72	4.52
Max Sentence length	8	11
Mean	204.38	300.87
Standard Deviation	595.44	1055.53

Within the training data, 1,000 sentence pairs were separated to be used as a test during each epoch of training to display validation accuracy and indicate overfitting of the encoder-decoder.

Test Data

The last 1,000 sentence pairs in the 20,000 sentence English to Spanish data set from Tatoeba were not seen by the encoder-decoder and were used as test data, given to the encoder-decoder after it finished training to evaluate the model's efficacy.

Table 2. Characteristics of Test Data

Characteristic	English	Spanish
Average Sentence Length	4.71	4.53
Max Sentence length	7	9
Mean	213.62	337.18
Standard Deviation	662.48	1285.73

Preprocessing

I began by removing punctuation from the language data and converting it to lowercase in order to standardize it and retain only the words themselves. I then tokenized the sentence data using Keras's built-in Tokenizer, which split each sentence into a list of probabilities representing its constituent words. Because not all of the sentences had the same number of words, I found the longest sentence and padded the rest with zeros so that every sentence would be the same length.

Model Architecture

My design consists of two models: an encoder decoder and a GAN, which also contains a generator and discriminator. It uses layers imported from the Keras Python library (Chollet et al., 2015).

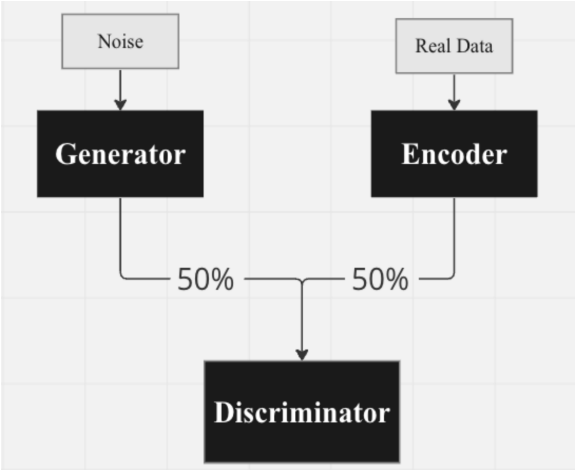

Figure 1. Model Workflow

Figure 1 describes the workflow of the model. First, the encoder-decoder learns to translate from Spanish to English using the training data. Once the encoder-decoder finishes training, the GAN trains, using the encoder-decoder as a tool. The generator takes in a batch of random noise, a list of randomly generated numbers between -1 to 1 which by themselves had no meaning. Then, the generator tries to generate sentence meanings out of them by transforming them into latent space representations. Next, a batch of Spanish sentences from the data set are fed into the trained encoder, and it generates "real" latent space representations. The encoder's output is labeled with a 1 and the generator's output with a 0 before both latent space representations are fed into the discriminator. Without looking at the label, the discriminator tries to identify from which model a given encoding comes and should return 1 for the encoder and 0 for the generator. Then, the discriminator compares its prediction with the actual label for the encoding to find out its accuracy. Depending on the discriminator's success or failure, the generator adjusts itself and learns to create encodings more similar to the encoder's in order to fool the discriminator.

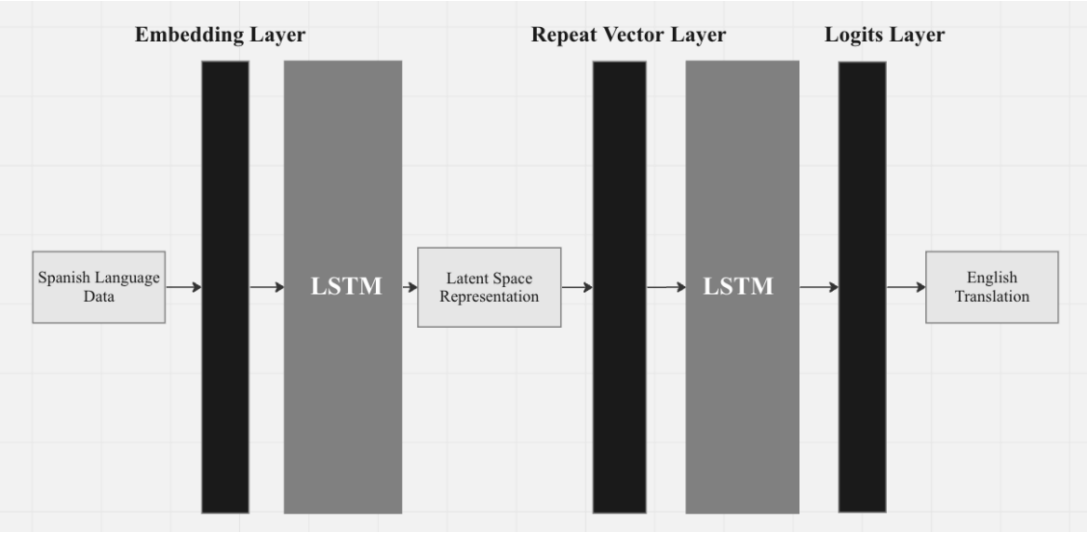

Figure 2. Encoder-Decoder

Shown in Figure 2, the encoder-decoder uses a LSTM architecture to encode the Spanish data into latent space representations and then decode the latent space into English. In between the major LSTM layers, 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.

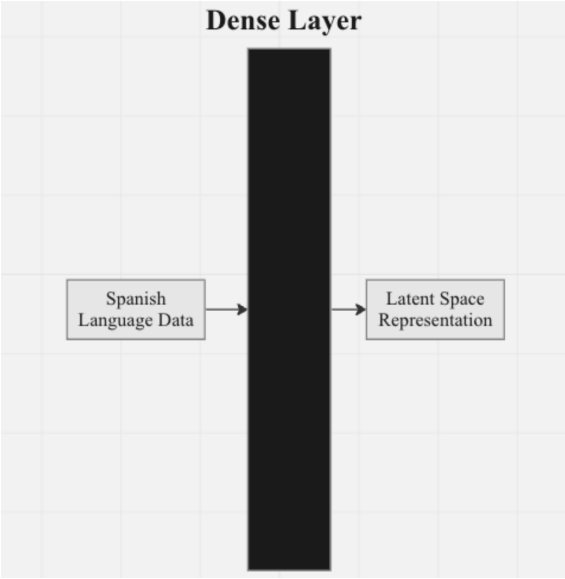

Figure 3. Generator

Shown in Figure 3, the generator consists of a dense layer, which is a layer of fully connected units. 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.

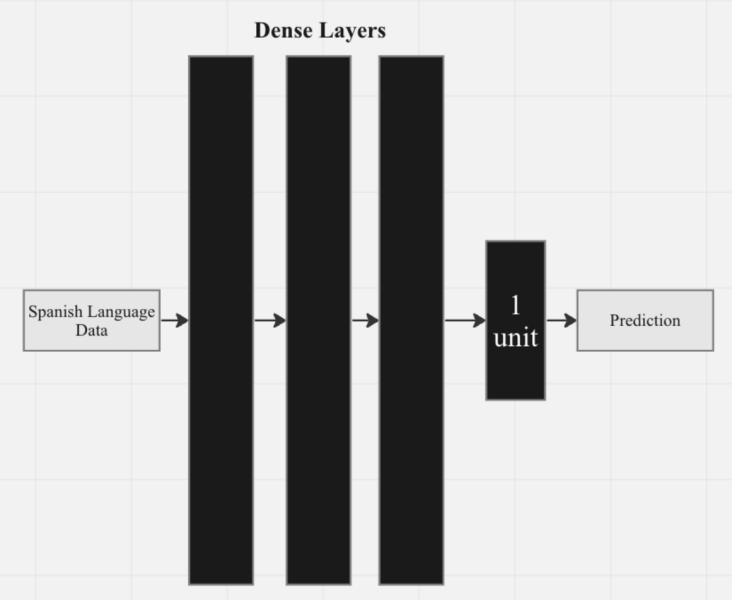

Figure 4. Discriminator

Shown in Figure 4, 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. Using trial-and-error, the discriminator learns how to differentiate encodings of natural Spanish data (from the encoder) from synthetic encodings (from the generator).

The combined GAN model feeds its input to the generator and the generator's output into the discriminator.

While the encoder receives sentences from training data and generates encodings representing their meanings, the generator receives language noise and generates its own, novel encodings. The discriminator receives both types of encodings and tries to differentiate them, learning from its success and failure. Meanwhile, the generator learns from the discriminator's results and tries to fool the discriminator by generating encodings more similar to the encoder's.

The training and test data is used as input for only the encoder-decoder while the GAN takes in only noise and the encoder-decoder's output.

Experimental Setting

The encoder-decoder's two LSTM layers each contained 64 units with weight decay and dropout to minimize overfitting. The encoder's LSTM had an L2 regularization of 5e-5 and dropout 0.5, the decoder's LSTM had an L2 regularization of 1e-5 and dropout 0.5, and the logits layer included a dropout of 0.5. The model used a softmax activation function and was compiled with a categorical cross entropy loss function and an Adam optimizer (Kingma and Ba, 2017) with a learning rate of 2e-3 and decay rates of 0.7 for beta1 and 0.97 for beta2. It trained in batches of 30 sentences for 400 epochs on the training data.

The generator's dense layer contained 64 units and a linear activation function. The model used a reLu activation function and 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 1024 units and reLu activation functions. It used a sigmoid activation function out of its single-unit dense layer, which categorized its prediction into a 1 (for the encoder) or 0 (for the generator). It was compiled with a binary cross entropy loss function and an Adam optimizer with a learning rate of 1e-4.

The GAN compiles with a binary cross entropy loss function and an Adam optimizer with a learning rate of 1e-4. Using a batch size of 1900, the GAN trained across 8000 epochs.

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 English sentences.

Hyperparameter Tuning

In order to find the hyperparameters for the most optimal results, I first varied the values by a factor of either 2 or 10 and tested every combination of the values with each other. To optimize 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 L2 regularizers, 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 by around 1 to 10. I continued this

process until I found the most optimal parameters and then increased the amount of training data and epochs, further improving the performance of the models.

Table 3. Hyperparameter Values

Hyperparameter	Value
Epochs (Encoder-Decoder)	400
Batch Size (Encoder-Decoder)	30
LSTM Units (Encoder-Decoder)	256
LSTM Dropout (Encoder)	0.5
LSTM Dropout (Decoder)	0.5
Logits Dropout (Encoder-Decoder)	0.5
L2 Regularizer (Encoder)	5e-5
L2 Regularizer (Decoder)	1e-5
Learning Rate (Encoder-Decoder)	2e-3
Beta1 Decay (Encoder-Decoder)	0.7
Beta2 Decay (Encoder-Decoder)	0.97
Epochs (GAN)	8000
Batch Size (GAN)	1900
Learning Rate (GAN)	1e-4
Dense Units (Generator)	256
Learning Rate (Generator)	4e-4
Dense Units (Discriminator)	1024
Learning Rate (Discriminator)	1e-4

Results

Encoder-Decoder Performance

The encoder-decoder performed with a final training accuracy of 92.8%. On validation data, it had a peak accuracy of 71.4%. Figure 5 shows the progression of training and validation loss and accuracy through epochs.

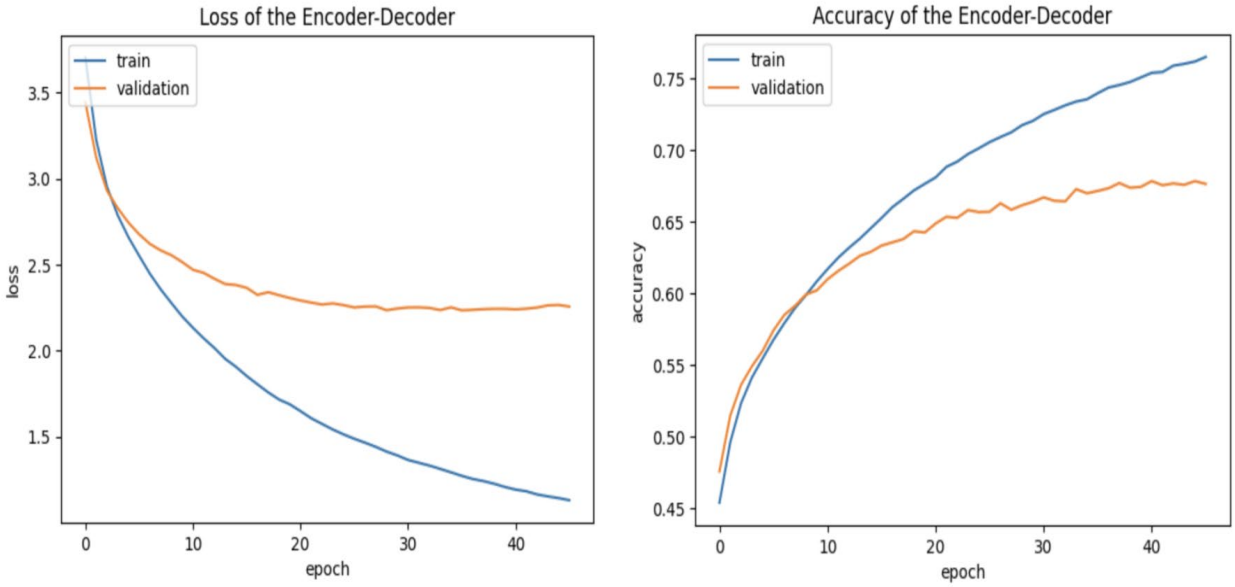

Figure 5. Loss and Accuracy of the Encoder-Decoder during Training

On the test data, the encoder-decoder had a final accuracy with 69.3%. This accuracy is respectable, considering the 50-60% baseline performance of other MT models with low-resource languages (Durrani and Ar shad, 2021). Making this result even more remarkable is 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 data sets, whose sizes are between 0.2 to 1 million (Ranathunga et al., 2021) (Zoph et al., 2016).

As this paper focuses more on the data augmentation aspect than the machine translation part of this research, accuracy was used instead of BLEU scores (Papineni et al., 2002). 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. In the future, I plan to use BLEU scores evaluation as I shift the research to more end-to-end translation performance.

GAN Performance

Shown in Figure 6, 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.

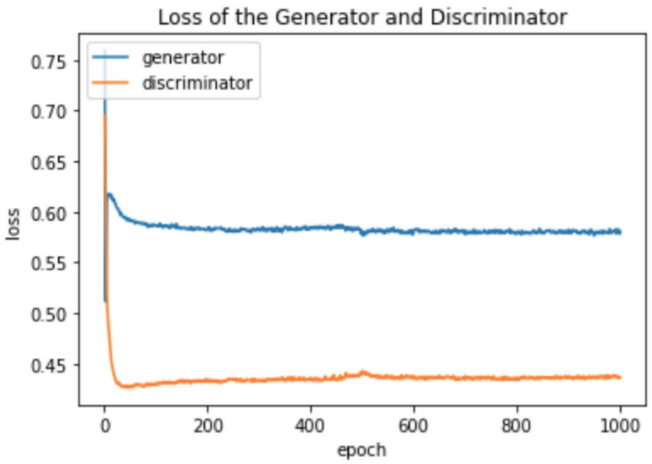

Figure 6. Loss of the GAN

When the GAN performed with the test data, it was able to generate *successful, coherent* sentences shown in the first three rows of Table 4. Upon manual inspection, I found that each sentence generally centered around its own cohesive theme, which is an indication of the model's successful understanding of word meanings. While some sentences were food-related, another centered around hiking, and others talked about a family member. 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.

Error Analysis

While the GAN was successfully able to generate some coherent sentences from scratch, the GAN made a significant number of errors. I qualitatively observed that the severity of the errors decreased as the GAN trained for more epochs. Thus, future models may train for a larger number of epochs to examine whether the frequency and severity of errors are able to reduce significantly further.

Repeated Words

Frequently, the GAN generated sentences with repeating words, shown in Table 4. I hypothesize that this issue, which occurs in most MT models (Fu et al., 2021), is due to the fact that the model is trying to generate words that are close in context to each other, which is necessary in order for a sentence to make sense. However, the model may not know or weigh in if it already used a word or not and ends up repeating that same word. It likely tries to find a word that is close in context to the prior word, and because related words have closer probabilities, it generates a probability very close to the previous word's probability. Then, after the decoder translates the latent space representations into probabilities, the close probabilities may be reduced to the same word because the words with the highest close probabilities are chosen to represent the probabilities the GAN returned. There may not be enough words with intricate probabilities close to a given one, so that one is chosen for all words with probabilities in a certain range around it. Fu et al. (2021) theorize that the issue is in the nature of languages themselves, as some words tend to predict themselves as the next word in context. Some incoherencies in the generated sentences could also be due to the fact the model may not fully understand the grammatical structure of sentences and resorts to repeating words it does know how to represent in order to fill up space. Potential solutions would be to train the model to remember the previous probabilities it generated and to vary its generated probabilities more to ensure that it does not repeat very similar ones.

Sample Generated Sentences Qualitative Evaluation 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 4. Sample Sentences Generated by the GAN

Sample Generated Sentences	Qualitative Evaluation
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

Nonsensical Grammar

Other sentences contained minimal repetition but were still grammatically incorrect or nonsensical, shown in Table 4. I believe that these sentences contain relatively randomly-placed words because the model has not learned about these words with enough context to know where to place them grammatically. Because it has seen these well-known yet complex words, it can generate them but is un sure of how many to generate, where to place them, or which words to surround them with. The model also may have mistakenly generated words with similar probabilities in search of words with close context. For example, “cheerful” and “weird” are both adjectives that describe “yourself,” so their probabilities may be similar enough for the model to generate them together. However, the model does not understand that these words have parallel meanings, and that only one should be used. A possible avenue for future work is to explore training the model on the difference between probabilities of words parallel in meaning and probabilities of related words that are required to be together in order to form a sentence.

Unrelated Words

Although the model generally uses related words like “studies” and “novels” together, it occasionally groups unrelated words together, shown in Table 4. I hypothesize that this is because it has not seen a word (like “gloves”) enough to understand its usual context (being put on people’s hands). However, it does understand that gloves is a noun and has previously seen nouns (such as people) being stressed, eating, worrying, and going online. A potential future path to explore would be incorporating a dictionary of words into the model’s training so that it better understands the words’ meanings.

Conclusion

Language and culture preservation is a serious problem, both socially and technologically. My proposal is the first of its kind, a generative-adversarial approach to low-resource language translation via data augmentation.

Experiments show promising results. Using less than 20,000 sentences, my model is able to generate unlimited, coherent sentences. These completely new sentences can be used to improve the accuracy of machine translation. This innovative approach will bring about more robust language translations for minority populations.

Because of its ability to generate an unlimited amount of original sentences despite being trained on minimal data, this GAN architecture 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. In addition, my 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. Improvements can be made on this research to increase the comprehensiveness when evaluating the model's performance and to minimize the repetition and incoherence in many of the generated sentences. One promising future direction is to train the model to understand the previous words it has generated and to remember the grammatical relationship between a word and others of similar probabilities.

Limitations

As the model was trained in a simulated low resource setting, its performance on real low resource languages would depend on these languages' similarities to English and Spanish. Specifically, isolating languages, which have limited morphology, would work better with this model. For the encoder-decoder, as the small amount of data lends itself to severe overfitting, further research could be done to minimize overfitting through reducing model capacity, implementing L1 regularization in addition to the current L2 regularization being used, experimenting with stronger dropout, and applying cross-validation. As mentioned in 6.1, I plan to use BLEU scores for more effective evaluation of sentence meaning as this research becomes more end-to-end. Thoroughly discussed in 6.3, various directions also exist to improve the accuracy and reliability of the GAN's generated sentences, from remembering previous probabilities to training the model to distinguish words parallel in meaning.

While this model contributes to machine translation by augmenting monolingual data and the use of monolingual corpuses is becoming increasingly prevalent in NMT models (Cai et al., 2021), extending this research to generate parallel translations would allow for a larger impact, as most NMT models take in parallel data rather than monolingual corpuses. Also, future software can be developed to clean the generated sentences from the GAN and extract only the coherent ones in order to add them to data sets. Reinserting punctuation and capitalization serves as another future area for exploration.

Acknowledgments

Many thanks to my teachers Anu Datar and Ricky Grannis-Vu for introducing me to the exciting fields of Computer Science and Natural Language Processing and for their helpful debugging tips and ongoing encouragement. Thank you!

References

Afroz Ahamad. 2019. Generating text through adversarial training using skip-thought vectors. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 53–60, Minneapolis, Minnesota. Association for Computational Linguistics.

Federico Betti, Giorgia Ramponi, and Massimo Piccardi. 2020. Controlled text generation with adversarial learning. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages

29–34, Dublin, Ireland. Association for Computational Linguistics.

Deng Cai, Yan Wang, Huayang Li, Wai Lam, and Lemao Liu. 2021. Neural machine translation with monolingual translation memory.

Wei-Rui Chen and Muhammad Abdul-Mageed. 2021. Machine translation of low-resource indo-european languages.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724– 1734, Doha, Qatar. Association for Computational Linguistics.

Francois Chollet et al. 2015. Keras.

Cheikh M. Bamba Dione. 2021. Multilingual dependency parsing for low-resource African languages: Case studies on Bambara, Wolof, and Yoruba. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 84–92, Online. Association for Computational Linguistics.

Sara Durrani and Umair Arshad. 2021. Transfer learning from high-resource to low-resource language improves speech affect recognition classification accuracy.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. 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)*, pages 567–573, Vancouver, Canada. Association for Computational Linguistics.

Zihao Fu, Wai Lam, Anthony Man-Cho So, and Bei Shi. 2021. A theoretical analysis of the repetition problem in text generation.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial networks.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor O.K. Li. 2018. 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)*, pages 344–354, New Orleans, Louisiana. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Surangika Ranathunga, En-Shiun Annie Lee, Mar jana Prifti Skenduli, Ravi Shekhar, Mehreen Alam, and Rishemjit Kaur. 2021. Neural machine translation for low-resource languages: A survey.

Ahmad Rashid, Alan Do-Omri, Md. Akmal Haidar, Qun Liu, and Mehdi Rezagholizadeh. 2019. Bilingual GAN: A step towards parallel text generation. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 55–64, Minneapolis, Minnesota. Association for Computational Linguistics.

Svetlana Tchistiakova, Jesujoba Alabi, Koel Dutta Chowdhury, Sourav Dutta, and Dana Ruiter. 2021. Edinsaar@wmt21: North-germanic low-resource multilingual nmt.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2018a. 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)*, pages 1346–1355, New Orleans, Louisiana. Association for Computational Linguistics.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2018b. Unsupervised neural machine translation with weight sharing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 46–55, Melbourne, Australia. Association for Computational Linguistics.

Zhirui Zhang, Shujie Liu, Mu Li, Ming Zhou, and En hong Chen. 2018. Bidirectional generative adversarial networks for neural machine translation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 190–199, Brussels, Belgium. Association for Computational Linguistics.

Francis Zheng, Machel Reid, Edison Marrese-Taylor, and Yutaka Matsuo. 2021. Low-resource machine translation using cross-lingual language model pre training. In *Proceedings of the First Workshop on Natural Language Processing for Indigenous Languages of the Americas*, pages 234–240, Online. Association for Computational Linguistics.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575, Austin, Texas. Association for Computational Linguistics.