# Data Augmentation Strategies for Music Composition using Generative Adversarial Networks ⋆

**Matheus Bitarães** * **Frederico Guimarães** ** **Frederico Coelho** ***

* *Machine Intelligence and Data Science Laboratory (MINDS), Graduate Program in Electrical Engineering - Universidade Federal de Minas Gerais - Av. Antônio Carlos 6627, 31270-901, Belo Horizonte, MG, Brazil, (e-mail: matheusbitaraes@ufmg.br).*
** *Machine Intelligence and Data Science Laboratory (MINDS), UFMG, (e-mail: fredericoguimaraes@ufmg.br).*
*** *Computational Intelligence Laboratory (LITC), UFMG, (e-mail: fredgfc@ufmg.br)*

**Abstract:** The field of Algorithmic Art has been following technological advances in Artificial Intelligence and, as Generative Adversarial Networks (GANs) have become popular, applications on art generation began to emerge. For most deep neural networks, large amounts of training data are essential to achieve satisfactory model quality. But there are cases, such as in MIDI musical melodies, where it is not trivial to acquire data in such a high volume. Data augmentation strategies play an important role on these cases. This paper presents a data augmentation pipeline, composed of three strategies, with the objective of improving the quality of a GAN-based musical melody generator. The proposed data augmentation pipeline was compared with a non-augmented dataset and a replicated dataset, which had the same size of the augmented dataset, but composed only of replicas. From the statistical tests performed it can be stated that the augmented dataset outperformed the non-augmented dataset and the replicated dataset, when evaluating the Fréchet Inception distance (FID) score.

*Keywords:* Generative Adversarial Networks, Neural Network, Generative Music, Data Augmentation, Algorithmic Music Composition, Agorithmic Art, Machine Learning.

## 1. INTRODUCTION

The use of Artificial Intelligence (AI) techniques for creating musical pieces has been gaining interest since early 1990s and it was already the theme of chapters and also entire books, such as Miranda and Al Biles (2007) and Romero et al. (2008). It is a field with many approaches, once art is not a deterministic problem to be solved. First initiatives of using machine learning and evolutionary computing for creating music date back to early 1990s, according to Loughran and O'Neill (2020), Horner and Goldberg (1991) and in *GenJam* project, which aimed to generate jazz solos using evolutionary computing (Biles et al., 1994), and now is an interactive improvisation system used in jazz performances (Biles, 2013). Another approaches to the problem are present in the literature, such as in Jhamtani and Berg-Kirkpatrick (2019), where a generative model with focus on self-repetition is proposed and in Xu et al. (2021), where the use of generative adversarial networks to create melodies is developed to serve as inspiration to music composers.

A diverse set of methodologies has been applied to music algorithmic composition. In Papadopoulos and Wiggins (1999), the methodologies were divided into: Mathematical Models, Knowledge Based Models, Grammar Models, Evolutionary Models, Learning Systems and Hybrid Systems. For some of the methodologies, there is need to have real data to serve as training material to the model. Specially in the case of Deep Neural Networks (which is the case of this paper), the larger the training data, the better. In this direction, data augmentation strategies play an important role for increasing models performance. Data augmentation can be interpreted as a set of strategies used to increase the amount of data by creating modifications of original samples. Such heuristics aim to enhance model's performance and generalization. They are specially valuable when original data availability is scarce.

This paper explores the effect of a set of data augmentation strategies into improving the training performance of a Generative Adversarial Network (GAN) for musical melody composition. A GAN can be used alone for generating melodies or within an architecture, as shown in Figure 1. The generated melodies can work as an initial population for an evolutionary algorithm, which will se-

⋆ This work has been supported by the Brazilian agencies (i) National Council for Scientific and Technological Development (CNPq), Grant no. 312991/2020-7; (ii) Coordination for the Improvement of Higher Education Personnel (CAPES) and (iii) Foundation for Research of the State of Minas Gerais (FAPEMIG, in Portuguese). MINDS Laboratory – https://minds.eng.ufmg.br/

lect melodies that fit best according to a *fitness function* composed by weights inputted by the user.

## 2. RELATED WORK

The algorithmic music composition field is heterogeneous regarding the technical approaches to the problem. Therefore, data augmentation strategies appear in different ways according to the problem modeling. For example, in Huang et al. (2018) the authors proposed a system called *Music Transformer*, an auto-regressive generative model that used self-attention mechanisms to generate musical pieces with recurring elements. For this system, the authors used augmentation strategies such as pitch transposition and time stretching. Later in Donahue et al. (2019), the *Transformer* architecture was adapted to multi-instrumental score generation. In this article, the authors introduced two other augmentation strategies, in addition to the previous ones: random instrument removal and randomly putting one instrument to play other instrument's melody.

There are examples of data augmentation in another approaches, as in Yong et al. (2019), where three augmentation methods were proposed for improving automatic piano transcription (which is to process piano sound recordings and transcript the notes that were recognized). They were: key transition, key change, and tempo change. In Yang et al. (2017) the MidiNet, a convolutional generative adversarial network for symbolic-domain music generation was proposed. The data augmentation strategy used was shifting melodies to other chords, different from the original one. In López and Fujinaga (2020), the authors proposed a technique for data augmentation based on roman numeral analysis annotations. They advocate that such annotation style can generate *harmonic reductions* by removing all musical features that do not interfere with the harmonic movement of the piece. After extracting such harmonic movement from the piece, new data could be generated including musical features that differ from the original one.

There is also effort on developing frameworks for data augmentation, as can be seen in McFee et al. (2015) where a framework called *MUDA* was proposed. It is a software architecture for applying data augmentation into music information retrieval tasks. The work was focused on defining the framework principles instead of proposing augmentation strategies, but during the examples the following strategies were used to augment the audio data: Pitch swift, Time stretch, Background noise and Dynamic range compression.

As can be seen, the modeling possibilities of data augmentation for music composition are diverse. But some ideas of augmentation, such as changing note times and changing pitch, have been present in a significant percentage of the approaches. This paper proposes a data augmentation pipeline, with three strategies, for increasing a GAN training data size and therefore increasing model's performance. It also explores the effect of time stretching and pitch change, as previously used in other articles. But it also implements a note addition strategy, which adds a note five semitones up or down on a random chosen note.

## 3. BACKGROUND

This paper is developed over some established concepts and techniques which are reviewed in this section.

### *3.1 Generative Adversarial Network (GAN)*

GANs were first proposed by Goodfellow et al. (2014) and are an unsupervised learning technique which is composed by two networks – the Generator and Discriminator. The Generator is a model that implicitly defines the probability density function of the real dataset $p_{model}(x)$. It does not necessarily have the ability to evaluate the density function $p_{model}$, but it can draw samples from this distribution. The Generator function can be defined as $G(z, \theta_G)$, where $z$ is a vector containing a random seed and $\theta_G$ is a set of learnable parameters. The main role of the Generator is to make function $G(z)$ transforms the random noise input $z$ into samples indistinguishable from real training data. The Discriminator is a model that receives samples $x$ and returns value $D(\theta_D)$, which estimates if the input came from real distribution or if it is a fake data made by the Generator model. Discriminator model responds from 0 to 1, where 0 is evaluated as a fake data and 1 is evaluated as real data (Goodfellow et al., 2020).

The relation between Generator and Discriminator is a game, where each player aims to minimize its own loss function, which is $J_G(\theta_G, \theta_D)$ for the Generator and $J_D(\theta_G, \theta_D)$ for the discriminator, as in Goodfellow et al. (2020). $J_G$ is the cross entropy between the Discriminator's evaluation and the optimal result, which is 1. Therefore, the Generator's loss is lowest when Generator can make the Discriminator evaluate its sample as 1. The Discriminator loss $J_D$ is the cross entropy between a real image and 1 plus the cross entropy between a fake image and 0.

### *3.2 Fréchet Inception distance (FID)*

GANs are nowadays a popular method for generating data, specially images. The field of quantitative GAN quality evaluation has been emerging and there are some approaches on how to evaluate the quality of a trained GAN. One of those methods is the Fréchet Inception distance (FID) , available in Heusel et al. (2017), which aims to compare generated images to real ones and evaluate its similarity. This score is based on comparing the distance between the probability distribution of real data $p_w(.)$ and the probability of generated data $p(.)$. It is based on the Inception Score (IS) proposed in Szegedy et al. (2016), which uses a third-party neural network, already trained on a supervised classification task. But, different from the IS, the FID compares the mean and covariance of a deep layer from the network. The layers are then encoded as multidimensional Gaussians and the distance between them is measured by the Fréchet distance, introduced by Fréchet (1957). The Fréchet Inception distance (FID) $d(.,.)$ between the Gaussian distribution with mean and covariance $(m, C)$ obtained from $p(.)$ and the Gaussian distribution with mean and covariance $(m_w, C_w)$ obtained from $p_w(.)$ is then given by the following equation:

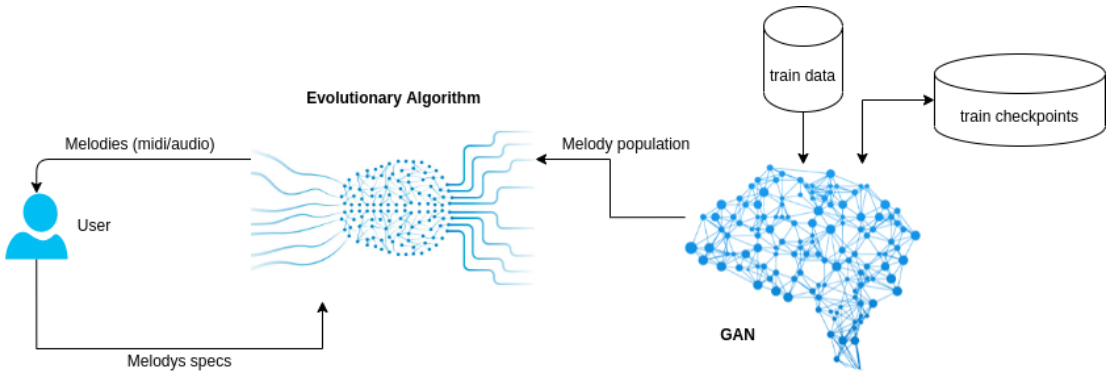


Figure 1. Example of a possible architecture of melody musical composition using a GAN model. A trained GAN can serve as a population generator for an Evolutionary Algorithm, which will tailor the melodies according with user inputs.

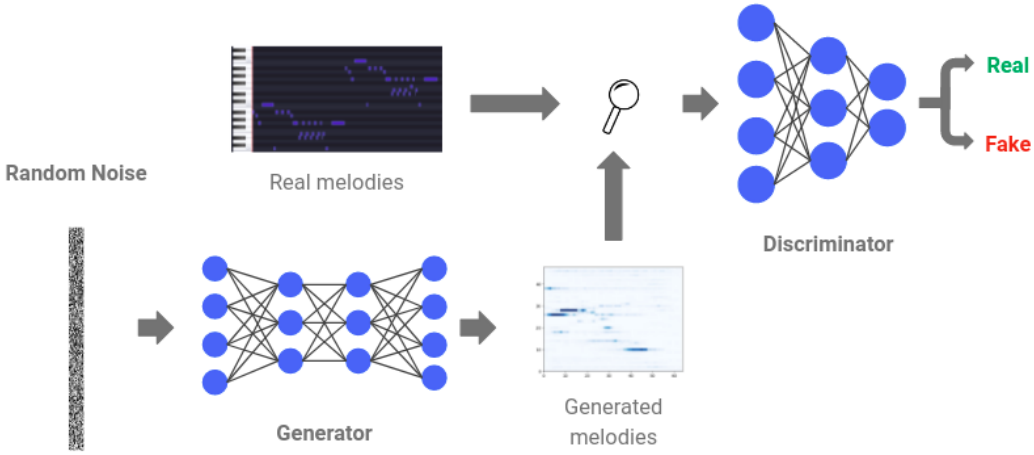


Figure 2. Relation between Generator and Discriminator models in a GAN. The Generator receives a random noise as input and generates a melody that is evaluated by the Discriminator model, which evaluates if received melody is real or generated.

$$d^2((m,C),(m_w,C_w)) = ||m-m_w||_2^2 + Tr(C+C_w-2(CC_w)^{1/2}) \quad (1)$$

The FID metric is widely used on evaluating the quality of GANs, including high-resolution GANs as in Karras et al. (2019) and Karras et al. (2020).

This paper compares GANs trained with augmented dataset with GANs trained with non-augmented datasets. For an statistically valid comparison, a suitable statistical test needs to be selected. Since we cannot state that the GAN accuracy samples follow normal distribution, a non-parametric statistical test needs to be performed. The Friedman test, developed by Friedman (1937), is a non-parametric test which repeatedly evaluate the variance of ranks between samples. The Friedman Test indicates that at least one of the groups contains a distribution that differs from others. To know the relation between the distributions, the Nemenyi test (Nemenyi, 1963) is applied. It is a *post-hoc* test that makes pair-wise comparisons in order to find the relation between groups of data.

## 4. METHODOLOGY

### *4.1 Data representation*

The choice of a suitable data representation strategy plays an important role on algorithmic music composition, once a representation can impose limitations to the model and change complexity of all training and optimization processes.

The representation chosen for this project is the Scaled Piano Roll (SPR), created as a variation of the commonly called Piano Roll representation. The Piano Roll is a representation where each note is distributed over the y axis while the time is disposed in the x axis. The Scaled Piano Roll, as can be seen in Figure 3, can be interpreted as the Piano Roll notes, translated in y axis according to the chord key on which the melody phrase was recorded.

Such representation improves generalization, in comparison with the "Piano Roll", because all the training process happens without influence of the chord being played, since all training inputs are shifted to the same key. The notes are translated to the chord after melodies are generated with GANs. Musical melody dynamics and intentions are

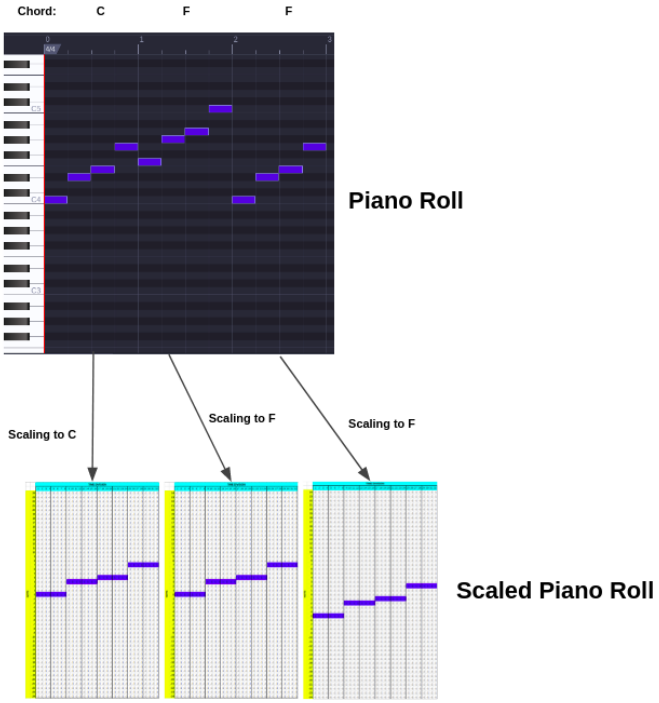


Figure 3. Scaled Piano Roll (SCP) representation in comparison with the Piano Roll representation

composed regardless chords and tonalities. The relations between notes is what makes a good or bad melody. Therefore, such attributes do not need to be present for training the GAN.

### *4.2 Data Augmentation Strategies*

Data augmentation plays an important role in contexts where training data is scarce or where sample perturbations improve generalization, such as in images. The augmentation strategies to be used in this model are described next.

*Time Change Strategy* Changes the duration of an arbitrary note from the melody. It can increase or decrease note time, clipping it to a suitable time division, as can be seen in Figure 4. This strategy can be run multiple times over the dataset to generate many configurations.

*Octave Change Strategy* Translate all melody up or down (chosen randomly) one octave, as can be seen in Figure 5. If this strategy runs more than one time, it can generate duplicate melodies. Therefore, this augmentation strategy is not recommended to be ran more than two times.

*Fifth Addition Strategy* Add a note five semitones up or down relative to a random chosen note. It is not recommended to run such strategy more than 4 times because it will generate melodies with multiple notes being played at the same time and can create unnatural melodies.

### *4.3 Data Augmentation Process*

The augmentation process was designed to support augmenting the dataset multiple times with multiple strategies. Every strategy $i$ has an augmentation factor $k_i$, which is how many times that strategy will be applied into the dataset. The augmentation pipeline can be seen in Figure 7. The following equation describes the augmentation process influence over a dataset of size $n$, with $i$ augmentation strategies.

$$n_{aug}(i) = n_{aug}(i-1)(k_i + 1)$$

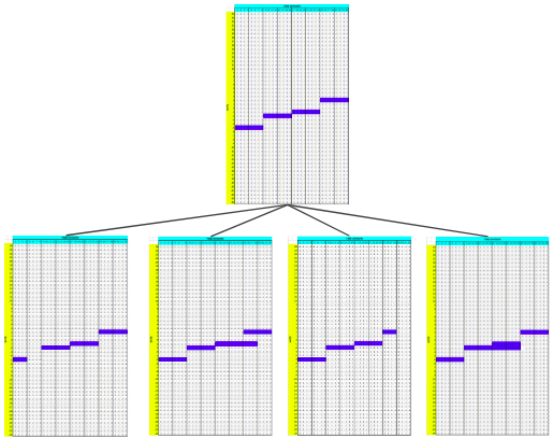

Figure 4. Time Change Strategy: Changes the time of random notes.

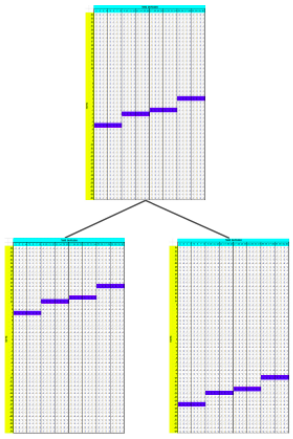

Figure 5. Octave Change Strategy: Changes octaves of entire melodies.

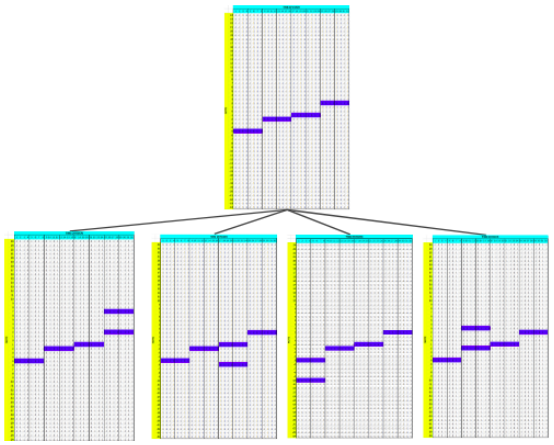

Figure 6. Fifth Addition Strategy: Adds notes five semitones up or down on a random note.

where $n_{aug}(i)$ is the total data size after applying augmentation strategy $i$, which has augmentation factor $k_i$. A dataset without any augmentation is as below:

$$n_{aug}(0) = n$$

Therefore, for example, if there are 10 test samples and the following factors:

- 1 - Time Change Strategy: $k_1 = 6$
- 2 - Octave Change Strategy: $k_2 = 1$
- 3 - Fifth Addition Strategy: $k_3 = 3$

we will have the following:

$$n_{aug}(0) = n = 10$$
$$n_{aug}(1) = 10(6 + 1) = 70$$
$$n_{aug}(2) = 40(1 + 1) = 140$$
$$n_{aug}(3) = 80(3 + 1) = 560$$

After all three augmentation strategies, the size of training dataset would go from 10 to 560 samples.

## 5. MODEL ARCHITECTURE

GAN is an architecture composed by a Generator model and a Discriminator model. The Generator's topology is according to Figure 8. It receives a random noise with dimension *noise_dim* (which is in our case 100), pass through a dense layer (layer where every neuron receives input from all the neurons of previous layer), a batch normalization layer, which aims to stabilize the learning process by normalizing output from dense layer, and a leaky ReLu layer, which is an activation layer with linear positive values and also a negative value propagation for training improvement. After that, there are two convolutions, batch normalization and leaky ReLu activation until it gets into output dimension, which is (48 x 64). The discriminator topology is according to Figure 9. It is a topology that receives a melody with dimension (48 x 64), applies two convolutions and dimension changes and responds a value between 0 and 1 indicating if the input image is real or fake (1 for real and 0 for fake).

As can be seen in the GAN topologies of Figure 8 and Figure 9, the GAN is modeled as if it was designed for generating images. The result of the GAN is a matrix that is then converted from the SPR format to a MIDI output.

## 6. EXPERIMENTS

### *6.1 Setup*

The training dataset used in this article was composed by a group of MIDI files, with a total of 55 bars, manually recorded by the author. It is available in this training data repository. The language used in the project was Python (Van Rossum and Drake Jr (1995)). The GAN architecture was implemented with support from TensorFlow package (Abadi et al. (2015)) and MIDI files importing was made by using the Mingus package (Spaans (2015)). For statistical tests, the Scikit Posthocs package, available in Terpilowski (2020), was used.

### *6.2 Test Design*

The same GAN architecture, described in previous sections, was trained with three different datasets. The Original Dataset, which was a dataset that contained only the recorded data, the Augmented Dataset, which contained the original data plus the data that was augmented, and the Replicated Dataset, which had the same size of the Augmented Dataset, but contained only copies of original data. The Replicated Dataset was important to be on the test in order to evaluate if augmenting the dataset would cause the same effect than simply replicating it.

The following augmentation strategy pipeline was applied:

- 1 - Time Change Strategy: $k_1 = 6$
- 2 - Fifth Addition Strategy: $k_2 = 3$
- 3 - Octave Change Strategy: $k_3 = 1$

Therefore, for a training dataset of 55 samples, the augmented dataset had 3080 samples.

The GAN specification for the test was the one below:

- *Epochs*: 60
- *Batch size*: 50
- *Maximum samples used for quantitative quality measurement*: 100

Each *epoch* represents one training session, where all samples, divided in *batches*, were used for optimizing the losses of both Generator and Discriminator models. There was the need of setting a value for *maximum samples used for quantitative quality measurement*, because the Fréchet Inception distance (FID) calculation can consume all available memory. Therefore, a maximum sample number must be fixed with the objective of preserving computer memory.

The training process was executed 55 times for all three datasets. The results are shown next.

### *6.3 Results and Discussion*

The box plot of the distribution of all FID evaluation scores computed after each training process can be seen on Figure 10. It indicates that the Augmented Dataset achieved better FID scores in comparison with the other two. But a another test was needed in order to identify if the difference was statistically significant.

Table 1 presents the average FID for each dataset training and Table 2 shows the p-values of the Nemenyi Test.

Table 1. Training results

| **Dataset** | **Dataset size** | **Fréchet Inception distance (FID)** |
|---|---|---|
| *Original* | 55 | 72.4334±6.44 |
| *Augmented* | 3080 | **21.7541±12.26** |
| *Replicated* | 3080 | 39.5702±20.15 |

Table 2. p-values of posthoc Nemenyi Test (FID)

| **datasets** | *Original* | *Augmented* | *Replicated* |
|---|---|---|---|
| *Original* | 1 | 0.001 | 0.001 |
| *Augmented* | 0.001 | 1 | 0.001 |
| *Replicated* | 0.001 | 0.001 | 1 |

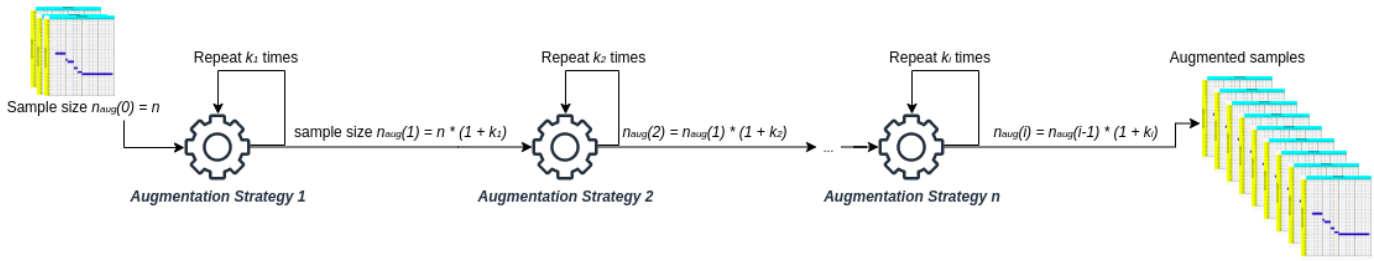


Figure 7. Proposed augmentation pipeline: Augments the data sequentially, $k_i$ times for each strategy $i$, with the final sample size $n_{aug}(i)$ being defined as $n_{aug}(i-1)(k_i + 1)$

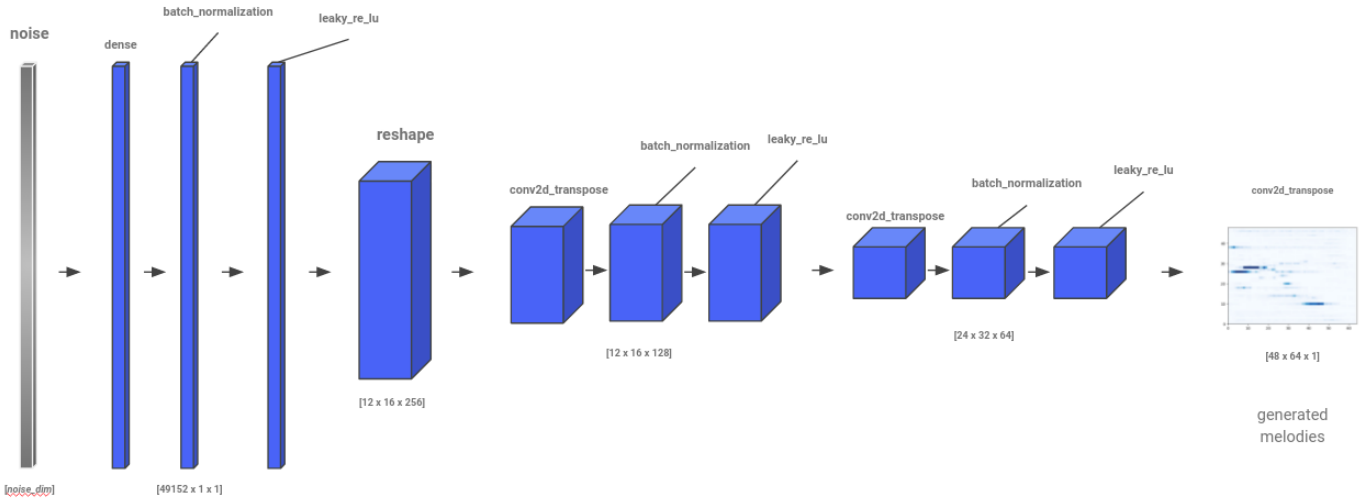


Figure 8. Generator Topology: receives a random noise (seed) with dimension *noise_dim*, pass through a dense layer, a batch normalization and a leaky ReLu layer. After that, there are two convolutions, batch normalization and leaky ReLu activation until it gets into output dimension.

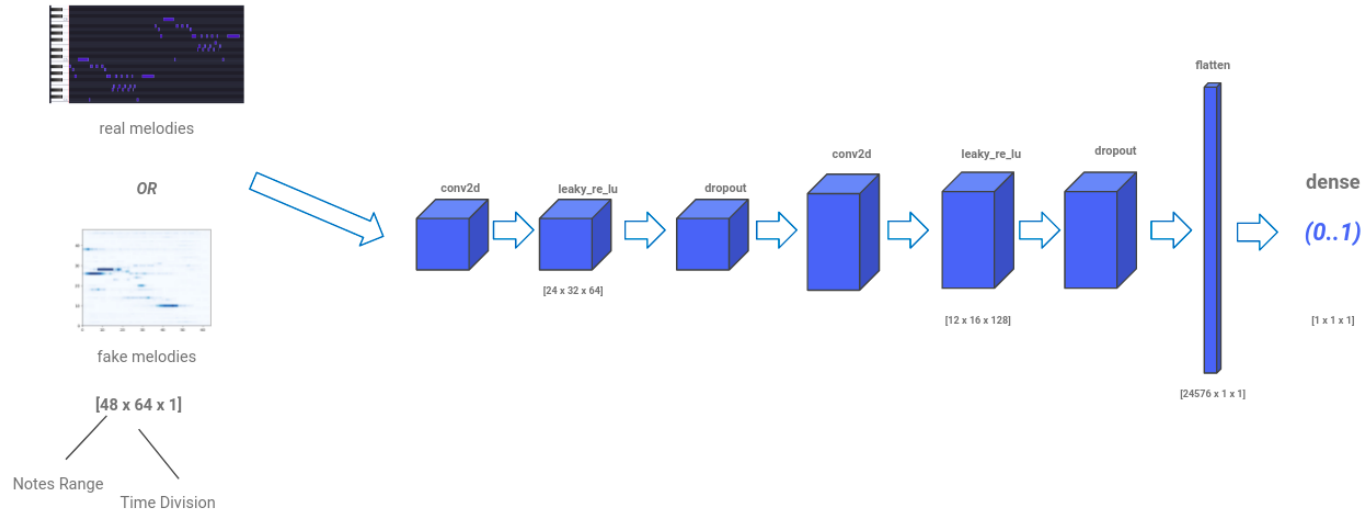


Figure 9. Discriminator Topology: Receives a melody with dimension (48 x 64), applies two convolutions and dimension changes and responds a value between 0 and 1 indicating if the input image is real or fake (1 for real and 0 for fake).

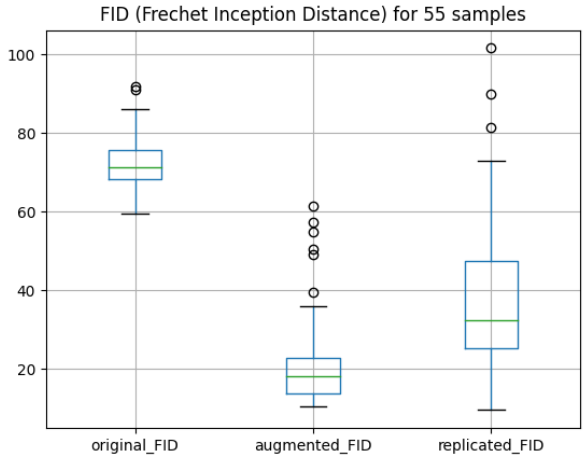


Figure 10. Distribution of the FID evaluation for 55 executions of each dataset type

By analyzing the p-values presented on Table 2, it is possible to state with 99% confidence that the three distributions differ from each other. The Augmented Dataset had the best FID value, followed by the Replicated Dataset, and at last the Original Dataset. By the results presented, we can notice the positive effect of the data augmentation strategies proposed in this article, which surpasses the effect of simply inflating the dataset by presenting copies of the original data.

## 7. CONCLUSION

In this paper, a data augmentation pipeline composed by three strategies was proposed and it was demonstrated that this method outperformed no augmentation and replication, when looking for the Fréchet Inception distance (FID) quantitative GAN evaluation score. The improvement in the FID metric stated by the statistical test gives a strong evidence that data augmentation improves the GAN melody generation. Future works can be done to also make qualitative evaluations, using humans to evaluate the musical quality of the piece. The next step will be about evaluating the complete architecture, such as described in Figure 1, with a GAN generating populations for an evolutionary algorithm which will optimize melodies according to users inputs. By interacting with this system, the user will be able to also evaluate generated melodies and, with such results, more evidence will be collected to evaluate the real efficiency of the complete model.

## REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Biles, J. (2013). Straight-ahead jazz with genjam: A quick demonstration. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 9.

Biles, J. et al. (1994). Genjam: A genetic algorithm for generating jazz solos. In *ICMC*, volume 94, 131–137.

Donahue, C., Mao, H.H., Li, Y.E., Cottrell, G.W., and McAuley, J. (2019). Lakhnes: Improving multi-instrumental music generation with cross-domain pre-training. *arXiv preprint arXiv:1907.04868*.

Fréchet, M. (1957). Sur la distance de deux lois de probabilité. *Comptes Rendus Hebdomadaires des Seances de L Academie des Sciences*, 244(6), 689–692.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, 32(200), 675–701.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, 63(11), 139–144.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30.

Horner, A. and Goldberg, D.E. (1991). *Genetic algorithms and computer-assisted music composition*, volume 51. Ann Arbor, MI: Michigan Publishing, University of Michigan Library.

Huang, C.Z.A., Vaswani, A., Uszkoreit, J., Shazeer, N., Simon, I., Hawthorne, C., Dai, A.M., Hoffman, M.D., Dinculescu, M., and Eck, D. (2018). Music transformer. *arXiv preprint arXiv:1809.04281*.

Jhamtani, H. and Berg-Kirkpatrick, T. (2019). Modeling self-repetition in music generation using generative adversarial networks. In *Machine Learning for Music Discovery Workshop, ICML*.

Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4401–4410.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. (2020). Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8110–8119.

López, N.N. and Fujinaga, I. (2020). Harmonic reductions as a strategy for creative data augmentation. In *Late-Breaking Demo at 21st International Society for Music Information Retrieval Conference*.

Loughran, R. and O'Neill, M. (2020). Evolutionary music: applying evolutionary computation to the art of creating music. *Genetic Programming and Evolvable Machines*, 1–31.

McFee, B., Humphrey, E.J., and Bello, J.P. (2015). A software framework for musical data augmentation. In

*ISMIR*, volume 2015, 248–254.

Miranda, E.R. and Al Biles, J. (2007). *Evolutionary computer music*. Springer.

Nemenyi, P.B. (1963). *Distribution-free multiple comparisons*. Princeton University.

Papadopoulos, G. and Wiggins, G. (1999). AI methods for algorithmic composition: A survey, a critical view and future prospects. In *AISB symposium on musical creativity*, volume 124, 110–117. Edinburgh, UK.

Romero, J., Romero, J.J., and Machado, P. (2008). *The art of artificial evolution: A handbook on evolutionary art and music*. Springer Science & Business Media.

Spaans, B. (2015). Mingus documentation. URL `https://bspaans.github.io/python-mingus/`.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2818–2826.

Terpilowski, M. (2020). Scikit posthocs documentation. URL `https://scikit-posthocs.readthedocs.io/en/latest/`.

Van Rossum, G. and Drake Jr, F.L. (1995). *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam.

Xu, Y., Yang, X., Gan, Y., Zhou, W., Cheng, H., and He, X. (2021). A Music Generation Model Based on Generative Adversarial Networks with Bayesian Optimization. In Y. Jia, W. Zhang, and Y. Fu (eds.), *Proceedings of 2020 Chinese Intelligent Systems Conference*, volume 705, 155–164. Springer Singapore, Singapore. doi:10.1007/978-981-15-8450-3_17. URL `http://link.springer.com/10.1007/978-981-15-8450-3_17`. Series Title: Lecture Notes in Electrical Engineering.

Yang, L.C., Chou, S.Y., and Yang, Y.H. (2017). Midinet: A convolutional generative adversarial network for symbolic-domain music generation. *arXiv preprint arXiv:1703.10847*.

Yong, S., Kim, C., Kim, J., and Telecom, S. (2019). Data augmentation and model optimization for piano transcription.