Thesis Title

Tudor Paisa
SNR: 2019551 | ANR: 315146
t.paisa@tilburguniversity.edu

Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Data Science and Society,
at the School of Humanities and Digital Sciences
of Tilburg University

Thesis Committee:
Dr. Menno van Zaanen
Dr. SECOND READER

Tilburg University
School of Humanities and Digital Sciences
Department of Cognitive Science & Artificial Intelligence
Tilburg, The Netherlands
March 7, 2019

Abstract

Lorem Ipsum

Contents

1 Introduction

The reemergence of neural networks (NN) has brought new ways to tackle Machine Learning problems (such as classification; see LeCun, Bengio, and Hinton, 2015 for several examples), as well as new possibilities to further artificial intelligence research (such as chatbots, or general-purpose AI systems; e.g., Dale, 2016 and Vinyals et al., 2017 respectively). Currently, two exciting areas of research in the field of Deep Learning (DL) are in designing generative models (Goodfellow, 2016) and training NNs on small datasets (Finn, Abbeel, & Levine, 2017; Nichol, Achiam, & Schulman, 2018). This research will combine both of these tasks by exploring the extent to which generative models can create novel and qualitative samples of music under the constraints posed by the few-shot learning problem. In other words, when data is scarce, can a generative model create samples of music that are new (i.e., not in the training set), consistent with common scales, and diverse (tone sequences are not repeated over and over, combines chord structures with single-note sequences, is not restricted to - say - one octave). This research will evaluate the generated samples of two state-of-the-art generative models (C-RNN-GAN of Mogren, 2016 and Performance-RNN of Oore, Simon, Dieleman, Eck, and Simonyan, 2018) that have been trained under the Reptile learning framework (Nichol et al., 2018) on small amounts of MIDI data.

DL as a field, promises to discover rich hierarchical models (LeCun et al., 2015), over various artificial intelligence applications, such as speech recognition, image classification, or text generation. The majority of developments usually involved discriminative models (Goodfellow et al., 2014), where high-dimensional input is assigned to a class label (Krizhevsky, Sutskever, & Hinton, 2012; X. Zhang, Zhao, & LeCun, 2015). In contrast, successes in generative models have been scarce, with Fully Visible Belief Networks (Frey, Hinton, & Dayan, 1996) being the most suitable for the longest run, followed years later by Bidirectional Long Short Term Memory (Graves, 2013) and Variational Autoencoders (Kingma & Welling, 2013).

However, studies on generative architectures were fairly uncommon until the discovery of the Generative Adversarial Network (GAN) by Goodfellow et al. (2014). GANs have had significant successes in image super-resolution (Ledig et al., 2016, see Figure 1.1), style-based image generation (Karras, Laine, & Aila, 2018), image-to-image translation such as a satellite image into a map, or a sketch into a photorealistic image (Isola, Zhu, Zhou, & Efros, 2016, see Figure 1.2).

These developments in generative models lie in the field of Computer Vision, but that is not to say that we can generate only images. GANs (and others)

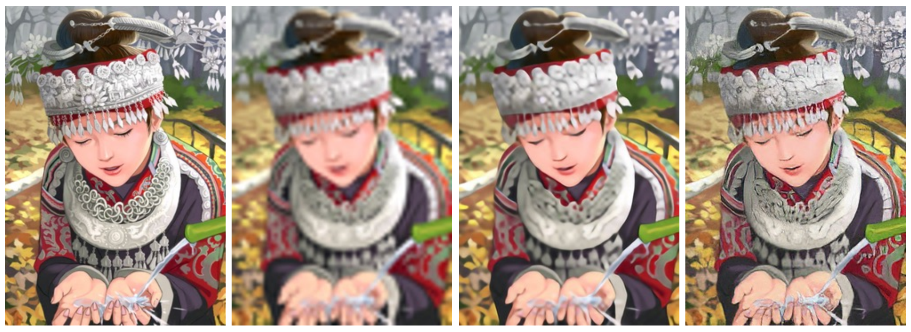

Figure 1.1: Example of single-image super-resolution results that highlight the advantages of a GAN. The original high-resolution image (1.1a) has been downsampled to make a low-resolution image. The results of bicubic interpolation (1.1b) can be seen in the second image, followed by the results of SRResNet (1.1c), a neural network trained on mean squared error. Lastly, the results of SRGAN can be seen in the final image (1.1d). Figure adapted from Ledig et al. (2016).

have been applied - with varying degrees of success - in Natural Language Processing where the scope was to generate high-quality text with sufficient diversity (e.g., L. Chen et al., 2018; Yu, Zhang, Wang, & Yu, 2016). Less explored, but still successful, have been the efforts in generating music (e.g., Dong, Hsiao, Yang, & Yang, 2017; Mogren, 2016). This paper will focus on the problem of generating music with a NN-based model. Respectively, this paper will evaluate the samples generated by C-RNN-GAN (Mogren, 2016) and Performance-RNN (Oore et al., 2018) when trained under the few-shot learning problem (detailed below).

It is fairly well known that DL models tend to be very "data-hungry", a caveat imposed by its optimization algorithm (W.-Y. Chen, Liu, Kira, Wang, & Huang, 2018; Ravi & Larochelle, 2016). DL usually relies on a procedure called stochastic gradient descent (SGC) which implies computing for a given input vector the output and error, followed by calculating the input's average gradient and consequently adjusting the network's weights (LeCun et al., 2015). Ravi and Larochelle (2016) argue that the iterative nature of gradient-optimization algorithms does not allow them to perform well under the constraint of a set number of updates: when faced with non-convex optimization problems, DL algorithms do not guarantee convergence within a reasonable amount of time. Finally, they add by saying that for each new problem (e.g., new dataset), the model needs to reinitialize with a new set of random parameters which hurts the network's ability to converge to a good solution when constrained by a limited number of updates (Ravi & Larochelle, 2016).

Besides the promise of making DL algorithms easier to train, primary reasons why this issue needs to be addressed are as follows. Firstly, humans have the ability to generalize after one (or few) example(s) of a given object (W.-Y. Chen et al., 2018; Ravi & Larochelle, 2016; Vinyals, Blundell, Lillicrap, Kavukcuoglu, & Wierstra, 2016), something which DL algorithms do not. Secondly, this would alleviate the constant need for data collection in order to pro-

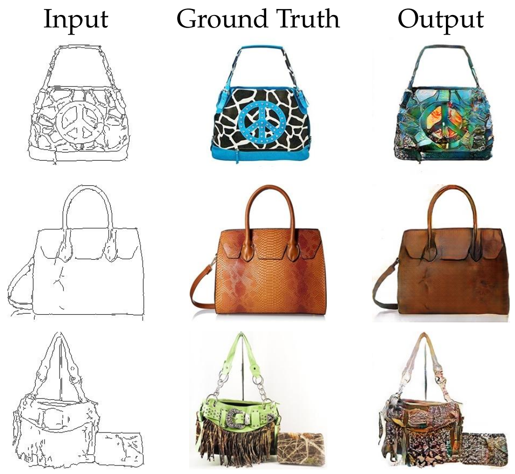

Figure 1.2: Isola, Zhu, Zhou, and Efros (2016) coined the term image-to-image translation where an input image is transformed into a revisualization of the original input. This figures illustrates how an outline of a purse (leftmost column) is transformed into a colored photorealistic version of it (rightmost column). In the middle we have the ground truth: the real image of the purse. Figure adapted from Isola, Zhu, Zhou, and Efros (2016).

vide reasonable performances. Finally, there are many fields where the data exhibits a large number of classes, with few examples per class and Bridging the gap between human-type learning and current learning architectures would allow models to properly capture this sort of data (Larochelle, Finn, & Ravi, 2017; Ravi & Larochelle, 2016). This is known as the few-shot problem (W.-Y. Chen et al., 2018; Larochelle et al., 2017; Vinyals et al., 2016).

In spite of its proposed benefits, developments and experiments in few-shot learning are still scarce (Larochelle et al., 2017) and evaluations are largely concentrated on image data (see W.-Y. Chen et al., 2018; Clouâtre & Demers, 2019; Lake, Salakhutdinov, & Tenenbaum, 2019; Ravi & Larochelle, 2016; Vinyals et al., 2016). Few-shot experiments with generative networks are even less frequent (Clouâtre & Demers, 2019; R. Zhang, Che, Ghahramani, Bengio, & Song, 2018). This research aims to explore the viability of this learning approach in the context of generating music data. More concretely, in light of recent developments, this research will evaluate two generative networks modelled on MIDI data under the constraints of few-shot learning, and comparing its performance to the baseline proposed by Larochelle et al. (2017) and properties extracted from real music performances.

2 Related Work

Lake, Salakhutdinov, and Tenenbaum (2015) initially popularized the idea of learning from a few class examples using a probabilistic learning framework capable of generalizing and learning a large number of concepts from a single class example (i.e., one-shot learning). Rezende, Mohamed, Danihelka, Gregor, and Wierstra (2016) were among the first to provide a solution to the one-shot problem in the context of a neural network (NN) by implementing Bayesian reasoning into a deep NN embedded within hierarchical latent variable models. The Reptile algorithm (Nichol et al., 2018) is a recent approach to this problem. It counteracts the shortcomings of gradient-optimization algorithms by learning the model's initial parameters in order to maximize its performance on novel tasks. Furthermore, it allows for any type of network to be used and does not place any restrictions on the type of loss function that can be used (Nichol et al., 2018). The algorithm for Reptile is as follows:

Algorithm 1: Reptile (serial version)

```
Initialize $\phi$, the vector of initial parameters
for iteration = $1, 2, \dots$ do
    Sample task $\tau$, corresponding to loss $L_\tau$ on weight vectors $\tilde{\phi}$
    Compute $\tilde{\phi} = U_\tau^k(\phi)$, denoting $k$ steps of SGD or Adam
    Update $\phi \leftarrow \phi + \epsilon(\tilde{\phi} - \phi)$
end
```

Although simplistic, this algorithm reaches similar results to another popular few-shot algorithm called MAML (Finn et al., 2017). Both algorithms are initialization-based learning methods however, a major distinction between the two is that the vanilla MAML implementation requires calculating a second-order derivative, which can significantly slowdown computation (Nichol et al., 2018). This forms a compelling reason to use Reptile as the few-shot learning method for this research.

Turning over to the generative side of things, nowadays, the most popular form of generating data is through the use of a Generative Adversarial Network (GAN; Goodfellow et al., 2014). Here a generative model G captures the data distribution and creates samples from it, whereas a discriminative model D estimates the probability that the input is real (i.e., comes from the training data) rather than fake (i.e., comes from G). Formally, given x sampled from real data, and z being input noise, D tries to keep $D(x)$ near 1, while making $D(G(z))$ near 0. Conversely, G tries to make $D(G(z))$ near 1 (Goodfellow, 2016). A notable implementation is C-RNN-GAN (Mogren, 2016), a continuous recurrent neural network with adversarial training. It consists of a network with two LSTM layers and 350 hidden units per layer, in both G and D. However, D has a bidirectional layout which allows is to take context from both past and future for its decisions, while G is unidirectional (Mogren, 2016). As is the standard practice for GANs, G will create fake samples of music, whereas D will try to classify correctly which sample is real.

Notwithstanding GANs, other generative successes have made use of adaptation on the Long Short-Term Memory (LSTM) network (Hochreiter & Schmidhuber, 1997). Here, the cells of a network are designed to allow for information to persist by implementing a recursive state unit, an input gate, a forget gate, and an output gate. The input of the network is allowed into the state unit if it is allowed to pass through by the input gate. Next, the forget gate controls the self-loop of the state cell. Finally, the output gate has the power to shut off the output cell (Goodfellow, Bengio, Courville, & Bach, 2016). Here, a notable implementation is Performance-RNN (Oore et al., 2018); a three layer LSTM network with 512 layers each. Compared to C-RNN-GAN, here the network is completely unidirectional. Probably more important than the network architecture, a central piece of the model's performance is due to the final representation of the MIDI data: each time slice is represented as a one-hot encoded vector containing Note-On (128 values), Note-Off (128 values), Time-Shift (125 values), and Velocity (32 values) events (Oore et al., 2018).

C-RNN-GAN and Performance-RNN have not been tested and adapted to the few-shot learning problem, but more general GANs and LSTM networks have been (see Vinyals et al., 2016; R. Zhang et al., 2018). However, as mentioned previously, no evaluation has been made in the musical domain. Therefore, this study will seek to answer the following research questions:

- To what extent is the music created by a few-shot generative model comparable to the music of a generative model that is trained on the entire dataset?
- To what extent is the music created by a few-shot generative model comparable to real music?

3 Methods

To provide an answer to the above research questions, this study will develop (in Keras) two generative models based on C-RNN-GAN and Performance-RNN, where the learning procedure will incorporate the Reptile algorithm. Other than that, the models will not differ from their original implementation. These models will firstly be compared to the baseline proposed by Larochelle et al. (2017) (a LSTM network trained on the entire train set) and then with the music from the dataset. The evaluation between the models with the baseline will be based on the number of statistically different bins (NDB; Richardson & Weiss, 2018) and domain specific measurements (polyphony, scale consistency, repetitions, tone span). The evaluation between the model's generated samples and real music in the dataset will be based on the domain specific metric. Refer to the evaluation section for more details on these metrics.

3.1 Dataset

The dataset for this research will be similar to that of (Oore et al., 2018) namely, the MAESTRO Dataset (Hawthorne et al., 2018). It consists of recorded MIDI data collected from each installment of the International Piano-e-Competition. All performances were done by piano experts on a Yamaha Disklavier, instrument which integrates a highly precise MIDI capture and playback system. Recorded MIDI events are of sufficient fidelity to allow judges to remotely listen to the contestant's performance (also on a Disklavier).

The dataset contains MIDI recordings from nine years of the International Piano-e-Competition. This amounts to 1,184 piano performances, approximately 430 compositions, 6.18 million notes played, and approximately 172 hours of playback. There is also a recommended train/validation/test split created on the following criteria (Hawthorne et al., 2018):

- No composition should appear in more than one split
- Training set is approx 80% of the dataset (in time), and the remaining is split equally on between the validation and test sets. Where possible, these proportions are true also within each composer.
- Popular compositions are in the training set
- The validation and test splits should maintain a variety of compositions

Moreover, each performance comes with additional metadata namely, the name of the composer, the title of the performance, the suggested train, validation, test splits, year of the performance, name of the file, and duration in seconds of the performance.

3.2 Evaluation

Generally, DL models work by the principle of likelihood maximization, which simply says to choose the parameters that maximize the probability that the model assigns to the training data (Goodfellow, 2016). Formally, this means selecting the parameters that maximize $\prod_{i=1}^{N} p_{\text{model}}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$:

$$\boldsymbol{\theta}^{*} = \arg\max \prod_{i=1}^{N} p_{\text{model}}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}). \tag{3.1}$$

However, calculating the product over many probabilities is prone to numerical problems such as underflow (Goodfellow, 2016). This is alleviated by calculating $\boldsymbol{\theta}^{*}$ in $\log$ space where the product is transformed into a sum:

$$\boldsymbol{\theta}^{*} = \arg\max \sum_{i=1}^{N} \log p_{\text{model}}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}). \tag{3.2}$$

That being said, calculating maximum likelihood can be thought of minimizing the Kullback-Leibler (KL) divergence: minimizing the dissimilarity between the data generating distribution p_{data} and the model p_{model}. Thus, $\boldsymbol{\theta}^{*} = \arg\min_{\boldsymbol{\theta}} D_{KL}(p_{\text{data}}(\boldsymbol{x}) \| p_{\text{model}}(\boldsymbol{x}; \boldsymbol{\theta}))$. The KL divergence is given by:

$$DL_{KL} = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}}[\log p_{\text{data}}(\boldsymbol{x}) - \log p_{\text{model}}(\boldsymbol{x})]. \tag{3.3}$$

Having all these in mind, $\log p_{\text{data}}$ is a result of the data-generating process, and not the model. Therefore, a final simplification is applied where the maximum likelihood estimate would be calculated as:

$$\boldsymbol{\theta}^{*} = -\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}}[\log p_{\text{model}}(\boldsymbol{x})]. \tag{3.4}$$

Note that Eq. 3.2 is the same as Eq. 3.4. The reason why this is brought up, is because negative log-likelihood (NLL; Eq. 3.4) is used both as a loss function and as an evaluation metric (also for generative models; Borji, 2018; Yu et al., 2016).

However, as Borji (2018) notes, NLL is uninformative about the quality of the samples generated and it does not allow to answer whether the generative network is simply memorizing training examples. As such, this study will also rely on other evaluation metrics when assessing generative models: the number of statistically different bins (NDB; Richardson & Weiss, 2018), polyphony, scale consistency, repetitions, and tone span (Mogren, 2016).

NDB is an evaluation metric specifically designed for generative models to measue the diversity of the generated samples. It follows the intuition that given two samples from the same distribution, the number of samples that fall into a bin (read as: class or cluster) should be the same (Richardson & Weiss, 2018). Let $I_B(\mathbf{x})$ be the indicator function for bin B. Then $I_B(\mathbf{x}) = 1$ if the sample falls into B, and zero otherwise. Let also $\{\mathbf{x}_i^p\}$ define N_p samples from a p distribution (e.g., test set samples) and $\{\mathbf{x}_i^q\}$ be the N_q samples from a q distribution (e.g., generated samples). If $p = q$ then the followings is also true:

$$\frac{1}{N_p}\sum_i I_B(\mathbf{x}_i^p) \approx \frac{1}{N_q}\sum_i I_B(\mathbf{x}_i^q) \tag{3.5}$$

The pooled sample proportion P_B is the proportion of samples (from the joined sets) that fall into B. The standard error for bin B is given by:

$$SE_B = \sqrt{P_B(1 - P_B)[1/N_p + 1/N_q]} \tag{3.6}$$

The test statistic is a z-score $z = \frac{P_B^p - P_B^q}{SE_B}$ where P_B^p and P_B^q are the proportions of each sample that fall into B. If z is smaller than a threshold (a significance level such as $\alpha = 0.05$), then the samples within that bin are statistically different. This whole process is repeated for each bin, and the number of statistically different bins is reported. The selection of bins is done through a K-means clustering performed on the $\mathbf{x}^p$ samples. Each of the generated samples $\mathbf{x}^q$ is by assigned to the nearest L_2 K centroid (Richardson & Weiss, 2018).

The more domain-specific evaluation measures are adapted from Mogren (2016). Polyphony measures how often a minimum of two tones are played simultaneously (when the start time is the same). It should be noted that this is a restrictive metric: it can give a low score to music where the two notes start at different times. Scale consistency is the fraction of notes that are part of a standard scale (e.g., major, minor, lydian, etc.), and reporting the results for the best matching scale. Repetitions counts consecutive subsequences of notes, and tone span is the difference between the lowest note and highest note (counted in semi-tones) (Mogren, 2016).

The choice of NDB comes from the fact that it is good at detecting overfitting (Borji, 2018), whereas the domain-specific metrics would favor models that generate high fidelity samples. In addition, these evaluation metrics have the benefit of being model-agnostic; they do not require a specific type of generative model.

4 Results

5 Discussion

6 Conclusion

Bibliography

Borji, A. (2018). Pros and Cons of GAN Evaluation Measures. *arXiv:1802.03446 [cs]*. arXiv: 1802.03446 [cs]

Chen, L., Dai, S., Tao, C., Shen, D., Gan, Z., Zhang, H., … Carin, L. (2018). Adversarial Text Generation via Feature-Mover's Distance. *arXiv:1809.06297 [cs]*. arXiv: 1809.06297 [cs]

Chen, W.-Y., Liu, Y.-C., Kira, Z., Wang, Y.-C. F., & Huang, J.-B. (2018). A Closer Look at Few-shot Classification.

Clouâtre, L., & Demers, M. (2019). FIGR: Few-shot Image Generation with Reptile. *arXiv:1901.02199 [cs, stat]*. arXiv: 1901.02199 [cs, stat]

Dale, R. (2016). The return of the chatbots. *Natural Language Engineering*, *22*(5), 811–817. doi:10.1017/S1351324916000243

Dong, H.-W., Hsiao, W.-Y., Yang, L.-C., & Yang, Y.-H. (2017). MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment. *arXiv:1709.06298 [cs, eess]*. arXiv: 1709.06298 [cs, eess]

Finn, C., Abbeel, P., & Levine, S. (2017). Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *arXiv:1703.03400 [cs]*. arXiv: 1703.03400 [cs]

Frey, B. J., Hinton, G. E., & Dayan, P. (1996). Does the Wake-sleep Algorithm Produce Good Density Estimators? In *Advances in Neural Information Processing Systems* (pp. 661–667). MIT Press.

Goodfellow, I. (2016). NIPS 2016 Tutorial: Generative Adversarial Networks. *arXiv:1701.00160 [cs]*. arXiv: 1701.00160 [cs]

Goodfellow, I., Bengio, Y., Courville, A., & Bach, F. (2016). *Deep Learning*. Cambridge, Massachusetts: The MIT Press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., … Bengio, Y. (2014). Generative Adversarial Networks. *arXiv:1406.2661 [cs, stat]*. arXiv: 1406.2661 [cs, stat]

Graves, A. (2013). Generating Sequences With Recurrent Neural Networks. *arXiv:1308.0850 [cs]*. arXiv: 1308.0850 [cs]

Hawthorne, C., Stasyuk, A., Roberts, A., Simon, I., Huang, C.-Z. A., Dieleman, S., … Eck, D. (2018). Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset. *arXiv:1810.12247 [cs, eess, stat]*. arXiv: 1810.12247 [cs, eess, stat]

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735. doi:10.1162/neco.1997.9.8.1735

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2016). Image-to-Image Translation with Conditional Adversarial Networks. *arXiv:1611.07004 [cs]*. arXiv: 1611.07004 `[cs]`

Karras, T., Laine, S., & Aila, T. (2018). A Style-Based Generator Architecture for Generative Adversarial Networks. *arXiv:1812.04948 [cs, stat]*. arXiv: 1812.04948 `[cs, stat]`

Kingma, D. P., & Welling, M. (2013). Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*. arXiv: 1312.6114 `[cs, stat]`

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 25* (pp. 1097–1105). Curran Associates, Inc.

Lake, B. M., Salakhutdinov, R., & Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, *350*(6266), 1332–1338. doi:10.1126/science.aab3050

Lake, B. M., Salakhutdinov, R., & Tenenbaum, J. B. (2019). The Omniglot Challenge: A 3-Year Progress Report. *arXiv:1902.03477 [cs]*. arXiv: 1902.03477 `[cs]`

Larochelle, H., Finn, C., & Ravi, S. (2017). *Few-Shot Distribution Learning for Music Generation*. AI-ON.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. doi:10.1038/nature14539

Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., ... Shi, W. (2016). Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. *arXiv:1609.04802 [cs, stat]*. arXiv: 1609.04802 `[cs, stat]`

Mogren, O. (2016). C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *arXiv:1611.09904 [cs]*. arXiv: 1611.09904 `[cs]`

Nichol, A., Achiam, J., & Schulman, J. (2018). On First-Order Meta-Learning Algorithms. *arXiv:1803.02999 [cs]*. arXiv: 1803.02999 `[cs]`

Oore, S., Simon, I., Dieleman, S., Eck, D., & Simonyan, K. (2018). This Time with Feeling: Learning Expressive Musical Performance. *arXiv:1808.03715 [cs, eess]*. arXiv: 1808.03715 `[cs, eess]`

Ravi, S., & Larochelle, H. (2016). Optimization as a Model for Few-Shot Learning.

Rezende, D. J., Mohamed, S., Danihelka, I., Gregor, K., & Wierstra, D. (2016). One-Shot Generalization in Deep Generative Models. *arXiv:1603.05106 [cs, stat]*. arXiv: 1603.05106 `[cs, stat]`

Richardson, E., & Weiss, Y. (2018). On GANs and GMMs. *arXiv:1805.12462 [cs]*. arXiv: 1805.12462 `[cs]`

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., & Wierstra, D. (2016). Matching Networks for One Shot Learning. *arXiv:1606.04080 [cs, stat]*. arXiv: 1606.04080 `[cs, stat]`

Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., ... Tsing, R. (2017). StarCraft II: A New Challenge for Reinforcement Learning. *arXiv:1708.04782 [cs]*. arXiv: 1708.04782 `[cs]`

Yu, L., Zhang, W., Wang, J., & Yu, Y. (2016). SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. *arXiv:1609.05473 [cs]*. arXiv: 1609.05473 `[cs]`

Zhang, R., Che, T., Ghahramani, Z., Bengio, Y., & Song, Y. (2018). MetaGAN: An Adversarial Approach to Few-Shot Learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 31* (pp. 2371–2380). Curran Associates, Inc.

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level Convolutional Networks for Text Classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 28* (pp. 649–657). Curran Associates, Inc.