Composing like a human:
Adapting generative networks to few-shot learning in the musical domain

Tudor Paisa
Student Number: 2019551
Administration Number: 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
April 30, 2019

Abstract

Lorem Ipsum

Contents

1 The return of the Neural Net

Over the past sixty years, Artificial Neural Networks (ANNs) have experienced a number of popularity cycles that were noted by extensive publicity of over-inflated expectations with regards to the promises of connectionism (the field of science that tries to explain mental phenomena with the help of ANNs [CITATION NEEDED]) [CITATION NEEDED]. More specifically, it promised that a system could learn to do anything it was programmed to do (Minsky, Papert, & Bottou, 1988). However, each wave of praise was followed by a mass of public disappointment, criticism, and funding cuts [CITATION NEEDED]. These upturns and downturns of the field of connectionism - or more generally A.I. - are now known in the literature as the A.I. winters [CITATION NEEDED].

The recent revival of ANNs under the scope of Deep Learning (DL) has brought new ways to tackle Machine Learning problems, most notably in the context of classification (but also regression, although a less popular application; see LeCun, Bengio, and Hinton, 2015 for several DL examples). In addition, with the advent of chatbots (Dale, 2016), voice-activated personal assistants (Xiong et al., 2018), and general-purpose A.I. systems (Vinyals et al., 2017), DL introduced new frontiers in artificial intelligence research.

Broadly speaking, DL as a field, promises to solve tasks which are easily (intuitively) performed by people, but hard to formalize (e.g., recognizing faces or spoken word; Goodfellow, Bengio, Courville, & Bach, 2016). The solution comprises of enabling computers to learn from experience and understand the world through the discovery of hierarchical relationships between concepts (LeCun et al., 2015). This way, there is no need for manual input on the sort of knowledge that the computer needs (Goodfellow et al., 2016). A more simplistic and concise interpretation of DL would be to see it as a statistical technique for identifying patterns in sample data using large (deep) ANNs (Marcus, 2018).

The reasons for presenting these facts will be clarified in the following sections (Section 1.3 provides a concrete explanation) however, a terse argumentation is that progress towards general A.I. systems is marked by scant explorations of ANNs outside of the classification task, and even more so when trying to overcome some of their shortcomings, such as being highly reliant on massive amounts of data [CITATION NEEDED].

1.1 We do not discriminate

Typically, in a neural network, the data samples go through a set of input units (that might represent pixels, word embeddings, etc.), then through multiple hidden layers, each with a given number of nodes, and reaching the output units where the answer is given (Marcus, 2018). An overwhelming majority of experiments with ANNs involve discriminative models (Goodfellow et al., 2014), where the goal is to assign the high-dimensional inputs to a class label (such as an animal or piece of furniture; Krizhevsky, Sutskever, & Hinton, 2012; X. Zhang, Zhao, & LeCun, 2015). However, other possible applications of DL models include (but are not limited to): regression [CITATION NEEDED], data compression (Cheng, Sun, Takeuchi, & Katto, 2018), and generating new data points (Graves, 2013).

For the purposes of this paper (detailed in Section 1.3), generative models are a central element in our experiments (Chapter 3). Borrowing the analogy from Goodfellow (2016), one might wonder what is the value of studying generative models. After all, in the image domain, such a framework would merely generate more images (something which the Internet as no shortage of). The answer to this lies in the fact that generative models can be used for simulating possible futures for scheduling and planning [CITATION NEEDED], providing predictions to missing data [CITATION NEEDED], or enabling work on multi-modal outputs, where a single input may have more than one correct answer (Goodfellow, 2016).

However, successes in generative models have been scarce, with Fully Visible Belief Networks (Frey, Hinton, & Dayan, 1996) being the most suitable (and popular) method for the longest run. Yet, this method requires generating one sample at a time (ramping up the cost of the process to $O(n)$). Coupled with the fact that the computation is done through a neural network, creating n samples requires a substantial amount of time (Goodfellow, 2016). Variational Autoencoders (Kingma & Welling, 2013) are another popular method that is used for this sort of task. Their main drawback however, is that when a weak approximation of a prior or posterior distribution of the data is used, the model might learn something other than the true data distribution (Goodfellow, 2016).

The discovery of Generative Adversarial Networks (GANs; Goodfellow et al., 2014) has leapfrogged the state of developments in this field. Besides countering the drawbacks outlined above (Goodfellow, 2016), they have been particularly successful at a variety of tasks such as image super-resolution (Ledig et al., 2016, see Figure 1.1), style-based image generation (Karras, Laine, & Aila, 2018), or image-to-image translation (such as converting a satellite image into a map, or a sketch into a photorealistic image; Isola, Zhu, Zhou, & Efros, 2016, see Figure 1.2).

In addition, although not as popular as the abovementioned, Recurrent Neural Networks (RNNs) have been successfully implemented for various generative tasks (Jenal, Savinov, Sattler, & Chaurasia, 2019; Sutskever, Martens, & Hinton, 2011). More specifically, a variation of these, the Long Short-Term Memory Networks (LSTMs; Hochreiter & Schmidhuber, 1997), are of particular interest. RNNs (and by extension LSTMs) contain a high-dimensional hidden state which allows them to remember and predict from previous inputs (Sutskever et al., 2011). This makes them great candidates for predicting se-

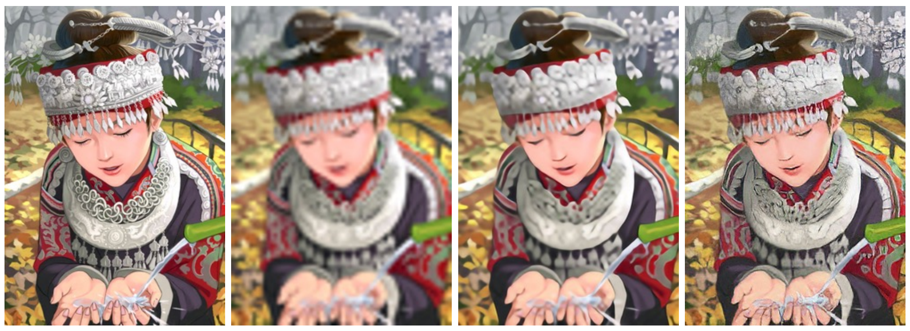


Figure 1.1: Example of single-image super-resolution results that highlight the advantages of a GAN. The original high-resolution image (1.1a) has been downsampled to make a low-resolution image. The results of bicubic interpolation (1.1b) can be seen in the second image, followed by the results of SRResNet (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).

quences (Graves, 2013) of text or characters, where the input at t_0 might be influenced by prior input at t_{-1}, t_{-2}, etc. (Fan, Qian, Xie, & Soong, 2014). That being said, the purpose-built memory cell of an LSTM permits better storing and accessing of information (Graves, 2013; Hochreiter & Schmidhuber, 1997) which in turn makes this RNN variation a better candidate for this sort of task.

Having said all of these, the majority of developments with generative models lie in the field of Computer Vision. But that is not to say that we can generate only images. Generative models have been applied with varying degrees of success also 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). Herein lies the scope of this paper. We evaluated the samples generated by a GAN (Mogren, 2016) and LSTM (Oore, Simon, Dieleman, Eck, & Simonyan, 2018) when trained under a meta-learning algorithm (see Section 1.2). To reiterate, a full explanation of the purpose of the paper is provided in Section 1.3.

1.2 We always want more

DL models usually rely on a procedure called stochastic gradient descent (SGD) which implies computing an output and an error for a given input vector, followed by calculating the input's average gradient and adjusting the network's parameters in a manner that minimizes the error (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 over few examples. In particular, when faced with non-convex optimization problems, DL algorithms do not guarantee convergence

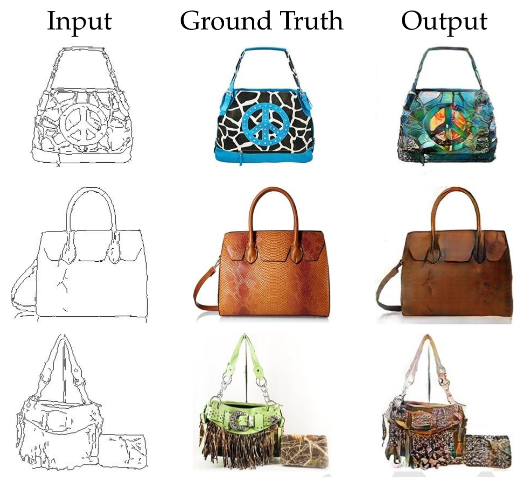


Figure 1.2: Isola, Zhu, Zhou, and Efros (2016) coined the term image-to-image translation where an input image is transformed into a revisualization of the original input. This figures illustrates how an outline of a purse (leftmost column) is transformed into a colored photorealistic version of it (rightmost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

within a reasonable amount of time. In addition, they also emphasize that for each new problem (e.g., new dataset), the model needs to reinitialize with a new set of random parameters. Ultimately, this hurts the network's ability to converge to a good solution when constrained by a limited number of updates (Ravi & Larochelle, 2016). Thus they lay out two clear drawbacks of DL models: "data-hunger", and inability to perform separate tasks.

A clear outcome from solving these issues would be that DL models are easier to train (less data) and multifunctional however, this list is not exhaustive. There is also fact that humans have the ability to generalize after one (or few) example(s) of a given object (W.-Y. Chen, Liu, Kira, Wang, & Huang, 2018; Ravi & Larochelle, 2016; Vinyals, Blundell, Lillicrap, Kavukcuoglu, & Wierstra, 2016) - something which DL models tend to lack. Moreover, there are many fields where the data exhibits a large number of classes, with few examples per class. Bridging the gap between human-type learning and current learning architectures would allow models to properly capture this sort of sparse data (Larochelle, Finn, & Ravi, 2017; Ravi & Larochelle, 2016). This has motivated a series of explorations in the direction of "few-shot" learning (i.e., learning a class from a few examples) or more broadly, into the field of "meta-learning" (learning to learn in hopes of generalizing to new tasks; W.-Y. Chen et al., 2018; Vinyals et al., 2016; R. Zhang, Che, Ghahramani, Bengio, & Song, 2018).

In spite of its proposed benefits, developments and experiments with few-shot learning are still scarce (Larochelle et al., 2017). Moreover, 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). Having said that, few-shot experiments with generative networks are even less frequent; Clouâtre and Demers (2019), Dong et al. (2017)

and R. Zhang et al. (2018) being some of the early explorers in this direction.

1.3 Composing like a human

At the time of writing, generative models and meta-learning are two exciting areas of research in DL which have been successfully combined. However, to the knowledge of the researcher, no such experiments have been conducted in the musical domain. Thus, this paper examined the extent to which generative models can create novel and qualitative samples of music under the constraints posed by few-shot learning. In other words, it sought to evaluate the extent to which generative models can create novel and high-fidelity samples when training data is scarce. To avoid ambiguity, we define novel as new (i.e., not in the training set), and qualitative as consistent with common scales and diverse (combines chord structures with single-note sequences, is not restricted to only a few notes, etc.). 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 et al., 2018) that have been trained under a meta-learning framework, Reptile (Nichol, Achiam, & Schulman, 2018), on a limited number of piano recordings. The performances of these two models were compared to the baseline proposed by Larochelle et al. (2017) and to recorded performances played by expert piano players at the International e-Piano Competition (University of Minnesota, 2019).

In other words, this study sought 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?

The following chapters will discuss more about generative models, meta-learning, and some of the models and algorithms involved in this paper (Chapter 2), followed by a detailed description of the setup of the experiments (Chapter 3), and the results obtained from them (Chapter 4). Finally, Chapter 5 closes with a discussion of the results and limitations of the experimental setup, whereas Chapter 6 draws some conclusions that answer the above research questions.

2 Related Work

This chapter takes a deep-dive into the fields of meta-learning and generative networks in order to highlight their current state of affairs. This will lay the groundwork for the experiments detailed in Chapter 3. Section 2.1 will present the principal network architecture of this paper, the Long Short-Term Memory network, and argument for its use as the foundation of a generative network. Section 2.2 will introduce Generative Adversarial Networks, a training method build specifically for the task, whereas Section 2.3 will formally introduce meta-learning, several approaches to it, and finally narrow down to current state-of-the-art algorithms.

2.1 Long Short-Term Memory Networks

In a standard ANN, the data traverses the network through the input layer, to the hidden layer, and finally reaches the output layer; setup which is generally known as a feedforward neural network (Figure 2.1; Graves, 2012). Broadly, this type of network is defined by the fact that its connections do not form cycles. If however we relax this condition, we arrive at a recurrent neural network (Figure 2.2a; Rumelhart, Hinton, & Williams, 1986): a family of neural networks specifically designed for processing sequential data (Goodfellow et al., 2016).

The information passes through the RNN in a similar manner to the feedforward network except that the activations which arrive at the hidden layer are from the current external input and prior hidden activations from the previous timestep (Graves, 2012). Moreover, recurrent networks also benefit from the fact that parameters (weights) are shared across different parts of a model. This allows extending and applying the model to data points of different lengths and generalizing across all of them (Goodfellow et al., 2016). In contrast, a feedforward network would require separate parameters for each input feature, at each position. Thus, a formal definition of the network's forward pass would be

$$h^t = f(h^{t-1}; x^t; \theta) \tag{2.1}$$

where h^t is the state of the hidden layer at time t, x^t is the input at time t, and θ the bias.

The backward pass in a RNN is similar to that of a feedforward network: given the partial derivatives of a differentiable loss function $\mathcal{L}$ we calculate the derivatives with respect to the network's weights (Graves, 2012). In this setting, a popular algorithm is called backpropagation through time

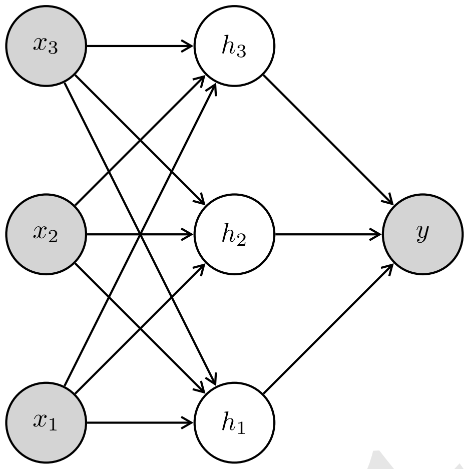
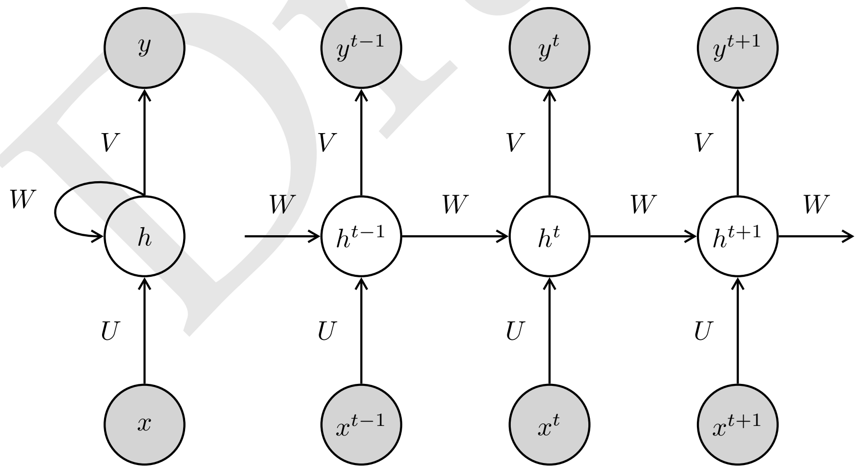


Figure 2.1: Example of a feedforward neural network with three nodes in the input layer (x_1, x_2, x_3), three nodes in the hidden layer (h_1, h_2, h_3), and an output layer a single node (y).

Equation 2.1 can be drawn as in Figure 2.2a: input sequences x are mapped to a hidden layer h and an output y, with the state of the hidden layer at time t feeding back into itself with a time delay of 1. In other words, the state of h_t will influence the decision of the recurrent layer at time $t + 1$. Another way to illustrate this is to unfold the computational graph such that each component of the network is represented by different variables per step, as in Figure 2.2b.



(a) Folded (b) Unfolded

2.2 Generative Adversarial Networks

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 et al., 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).

2.3 Learning to learn

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:

Although simplistic, this algorithm reaches similar results to another popular few-shot algorithm called MAML (Finn, Abbeel, & Levine, 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-

Algorithm 1: Reptile (serial version)

```
Initialize $\phi$, the vector of initial parameters
for iteration = $1, 2, \ldots$ 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
```

shot learning method for this research.

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.

Cheng, Z., Sun, H., Takeuchi, M., & Katto, J. (2018). Deep Convolutional AutoEncoder-based Lossy Image Compression. *arXiv:1804.09535 [cs]*. arXiv: 1804.09535 [cs]

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]

Fan, Y., Qian, Y., Xie, F.-L., & Soong, F. K. (2014). TTS synthesis with bidirectional LSTM based recurrent neural networks. In *INTERSPEECH*.

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. (2012). *Supervised Sequence Labelling with Recurrent Neural Networks*. Studies in Computational Intelligence. Berlin Heidelberg: Springer-Verlag.

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]

Jenal, A., Savinov, N., Sattler, T., & Chaurasia, G. (2019). RNN-based Generative Model for Fine-Grained Sketching. *arXiv:1901.03991 [cs]*. arXiv: 1901.03991 [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]

Marcus, G. (2018). Deep Learning: A Critical Appraisal. *arXiv:1801.00631 [cs, stat]*. arXiv: 1801.00631 [cs, stat]

Minsky, M., Papert, S. A., & Bottou, L. (1988). *Perceptrons* (Reissue edition). Cambridge, MA: MIT Press.

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]

Rumelhart, D. E., Hinton, G. E., & Williams, R. (1986). Learning Representations by Back Propagating Errors. *Nature*, *323*, 533–536. doi:10.1038/323533a0

Sutskever, I., Martens, J., & Hinton, G. (2011). Generating Text with Recurrent Neural Networks. In L. Getoor & T. Scheffer (Eds.), *Proceedings of the 28th International Conference on Machine Learning (ICML-11)* (pp. 1017–1024). ICML '11. ACM.

University of Minnesota. (2019). International Piano-e-Competition. http://www.piano-e-competition.com/.

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]

Xiong, W., Wu, L., Alleva, F., Droppo, J., Huang, X., & Stolcke, A. (2018). The Microsoft 2017 Conversational Speech Recognition System. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 5934–5938). doi:10.1109/ICASSP.2018.8461870

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.