# Korsak Ovbot: Exploring Artificial Creativity through Music Composition

George Moe
*Harvard College '21*

**The ability to create has long been heralded as the gift which distinguishes humanity from their cold, mechanical creations. But not for long: maturing artificial neural network technology has allowed us to create analogues of basic human perception and response with promising results. Building upon recent concepts in neural audio processing, we present Korsak Ovbot, a music-generation program which employs an iterative process to produce original music in a jazz-like style. We claim that this methodology mimics the human creative process, with the hope that, through proper development, artificial systems may one day produce art like humans do.**

## Introduction

Until recently, artwork created by computers has largely been generated by procedural methods, that is, through formulaic algorithms directed by human design. Such approaches to computer-generated artwork, or "artificial creativity," are criticized as being too closely guided by the designer's choices; they are said to be un-originally following hard-coded patterns or regurgitating pieces of human artwork that it has already seen.

However, recent developments in deep neural network (DNN) technology have put forth promising approaches to implementing a more organic creative process in software. At the heart of these improvements is a successful capability for DNNs to mimic human sensory processes, at least on an abstract level. Google's DeepDream (Mordvintsev & Tyka, 2015) visually demonstrated this ability by generating "hallucinations" from image-recognition networks, using feedback from the network to heighten perceived forms in the image data (Figure 1). The resulting methodology and visualizations seem analogous to imaginary visual perceptions in humans, similar to when we spot animals in clouds or see figures in the dark.

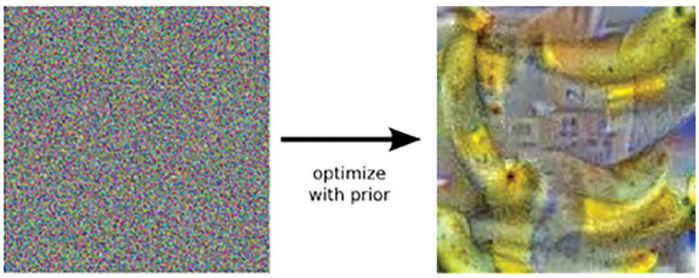


**Figure 1: Using DeepDream to see bananas in noise.**

Human artwork appears to be generated in a closely related way. Anecdotally, human artists seem to iteratively modify their output medium until they find it *satisfactory*[1]. For instance, a sketch artist may pencil, erase, and pencil again until the figure pleases her imagination; a composer may tweak and experiment with notes and melodies until it resonates with her tastes. This feedback-based approach seems just like the process used by DeepDream to generate hallucinations, but rather than altering the medium to maximize an object-recognition function to attain mirages, human artists seem to instead improve their medium to maximize an aesthetic pleasure function to produce art.

We propose an algorithm for "artificial creativity" inspired by this iterative procedure: Train a DNN to differentiate aesthetic inputs from random noise, and use it to greedily maximize the aesthetic value of the output medium via evolution. To test the merit of this approach, we applied the algorithm to the domain of artificial music composition[2] to create **Korsak Ovbot** (named after composer Nikolai Rimsky-Korsakov). Trained on about an hour's worth of music performed by Thelonius Monk (Thelonius Monk Quartet, 1963), Korsak has generated original melodies which seem to resemble the style of its source material without merely duplicating it. We fondly call this derivative style "Robo-Jazz." In this paper, we will examine the background which motivated Korsak's creation, as well as detail its architecture and operation. Then, we will evaluate Korsak's output and remark on how well it mimics human creativity. Finally, we will mention some leads for future work.

## Background and Related Work

A common approach to computational music generation is to use a procedural algorithm. This method relies on hard-coded rules for acceptable melodies. The result is music with high fidelity to recognizable motifs found in human-composed music. One recent example of this approach is MetaCompose (Scirea et al., 2016), which produces music by evolving a melody that optimizes for desirable compositional features (arpeggios, leaps, harmony with a chord progression, etc.). While this design was successfully able to generate music that was rated as "pleasant" and "interesting" by an audience, it is hard to credit the algorithm for producing these properties, since it was the creators who hard-coded these preferences in the first place. In general, this is the pitfall which makes it difficult to call procedural algorithms "creative."

On the other hand, the recent maturity of DNN technology has enabled the development of purely neural music generators. These approaches seem more organic because DNNs are able to learn desirable music patterns just by themselves. To this end, they often

1 Mace & Ward, 2002, offer a detailed model of the creative process which appears to support this intuition. In Phase 3 of their model, they describe the process of making the artwork, which includes a cycle of development and evaluation,with opportunities to complete or abandon the work.

2 This seemed to be good grounds for a proof-of-concept because humans can easily differentiate music from noise, while still maintaining a high degree of subjectivity as to what comprises good music. Contrast this to the visual arts, for which there exists disagreement around whether to consider even some instances of human artwork to be noise.

employ Recurrent Neural Networks (RNNs), which are adept at modeling timeseries such as audio data. Music is then generated by using the network to *predict* the next note from a history of previous notes. One instance of such an implementation is the Deep Artificial Composer (Colombo et al., 2017), which uses two RNNs for pitch and duration to generate music resembling Irish folk tunes. In terms of creativity, the authors found that the music generated was indeed more novel than the training set, but since it had only learned from that limited set, it was not as novel as outside music within the validation set. This appears to be the common pitfall with purely neural approaches, in that they seem to "regurgitate" patterns which they have already seen. While this may result in innovations upon the training set, it is difficult for these products to stray too far from what was learned.

RNNs also face difficulty when processing more granular forms of audio. The human ear is tremendously sensitive to auditory input, so most raw audio data are stored with 44,100 samples per second, amounting to hundreds of thousands of timesteps for just a few seconds of music. RNNs scale in computational complexity relative to the length of the input, so while it may be simple to process just a couple of *musical notes* per second of data, processing *ensemble instruments* and other more detailed effects found in modern music can quickly become intractable.

Fortunately, recent work has demonstrated ways to treat audio-processing as an image-processing problem by transforming sounds into spectrograms. Spectrograms are visual representations of audio data created by mapping the intensities of component frequencies over time. These may then be used as inputs to Convolutional Neural Networks (CNNs), the kind of network used in traditional image processing, which have already matured to handle enormous images with great speed. This method of treating audio as a visual problem was popularized by Google's work in WaveNet (Oord et al., 2016), which used a form of CNN to generate high-resolution text-to-speech vocals, and Tacotron 2 (Shen & Pang, 2017), which could generate these vocals in *real-time*. Figure 2 shows how Tacotron 2 uses a combination of CNNs with RNNs to generate a spectrogram from letter sequences.

Finally, YouTuber carykh later demonstrated that this CNN approach to audio-processing was accessible to consumer-grade hardware (Huang, 2017). It was his success in generating baroque music using the method that originally inspired this project.

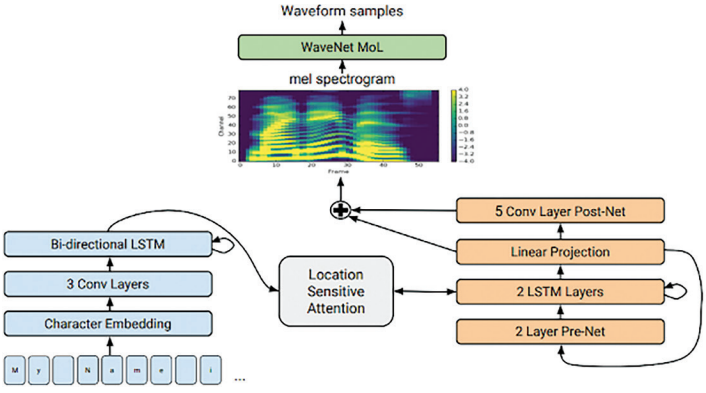


**Figure 2: Tacotron 2 maps letters to a spectrogram using RNNs and CNNs.**

## Algorithm Overview

Our approach to music generation is to mimic the human creative process of iteratively developing and reevaluating the work to maximize an aesthetic function. We believe that an aesthetic function may be learned by training a DNN on a large sample of human-composed music. To make this tractable, we choose to use a CNN-based architecture, inspired by the spectrogram processing methods discussed before. Also, we choose to use a DNN to learn the aesthetic function, instead of hard-coding it ourselves, so that the network may discover subtle patterns which may be difficult for humans to codify.

To generate music in a way that mimics the iterative nature of the human creative process, we choose to evolve the melody with an evolutionary algorithm, using the aesthetic function as the fitness metric. In particular, we seed a population of random spectrograms (also called "canvases") which are repeatedly scored, filtered, and mutated until a desired number of iterations have been performed. Then, the canvas spectrogram is converted to the final output audio. This pipeline is depicted in Figure 3.

## Preprocessing

To employ the CNN approach, we preprocessed the training music into spectrogram slices. The music was first converted to a mono-channel 44,100Hz WAV format, which is a raw amplitude timeseries format amenable to signal processing. This signal was sliced in sliding 8-second windows, in strides of 1 second, into 2,715 signal slices. These wave slices were then converted to spectrograms by applying the Fast Fourier Transform (FFT) on smaller sliding windows of size 8,192 samples (about 0.19s) with 50% overlap. These FFT decompositions described 4,097 component frequencies; in order to reduce network size, this was truncated to the lowest 200 components, discarding unused high frequencies while still retaining the melody. Figure 4 illustrates a breakdown of this process, and Figure 5 shows some sample spectrogram slices.

The spectrograms were then saved as greyscale images with values normalized to the interval [0, 255]. The final slices had

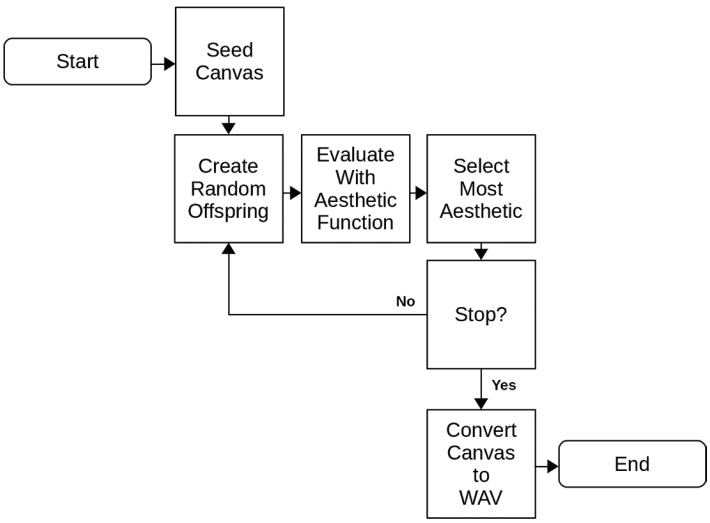


**Figure 3: Procedure to generate music by evolving melody on a trained aesthetic function.**

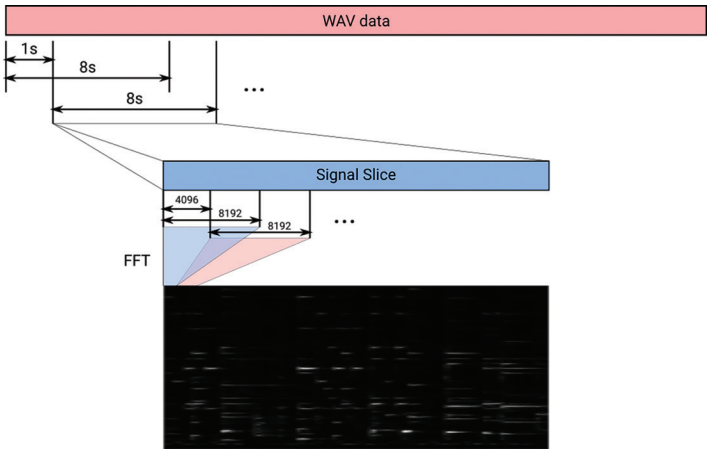


**Figure 4: Illustration of slicing scheme from WAV signal to spectrogram.**

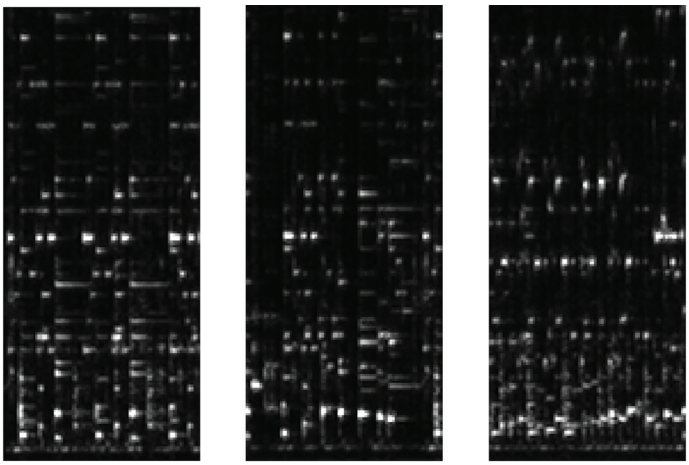

**Figure 5: Three sample spectrogram slices (Contrast adjusted for visibility).**

dimensions 200×87px, with the 87px vertical axis being the time axis (due to matrix representation). In our figures, we have rotated the spectrograms so that the horizontal axis is the time-axis to conform with display convention.

## Model Architecture

We designed a DNN capable of learning an aesthetic pleasure function. We chose to model the simplest such function, which is a mapping from input spectrograms to a rating of music vs. noise. Thus, the input to the DNN is a spectrogram image, normalized from [0, 255] down to [0.0, 1.0], and the output is a pair of indicators representing noise and music probability.

To accomplish this, we employed a series of convolution and max-pool layers topped by a fully-connected layer to encode the input spectrogram into a lower-dimensional embedding. (In this regard, the architecture is similar to a rudimentary image-recognition network.) Then, a fully-connected layer renders the embedding into the output music vs. noise predictions. This sequence of layers, their sizes, and parameters, are illustrated in Figure 6. The final model parameters were selected for performance on our development system, which has very little VRAM. They were discovered though a manual grid-search (trial and error).

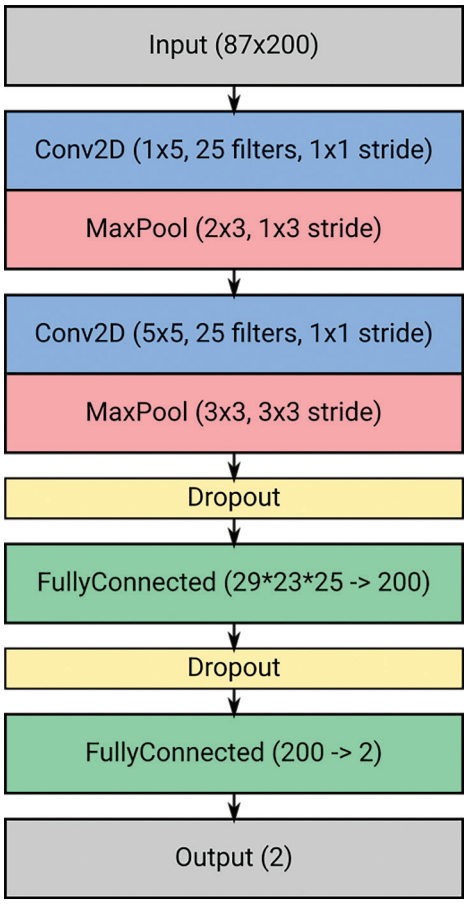


**Figure 6: Illustration of model, dimensions, and parameters.**

## Training

To model the simple aesthetic function described previously, we trained the network to distinguish between positive music slices and negative randomly-generated noise examples (Figure 7). There were two stages of training which corresponded to different types of noise examples. In the first 10 epochs, the network was shown examples of music and fully-random noise in order to prime the network for music recognition. Then, in the second stage, the network was shown harder examples of tonal noise in order to refine its recognition of musical patterns and harmonics. In both stages, positive and negative samples were randomly truncated to accustom the network to discerning the two even during our left-to-right generation scheme (detailed in the following section).

Training was performed in epochs of 1,000 iterations with 128 examples per batch. The final network was trained for approximately 2.7 million iterations over the course of almost 24 hours.

## Generating Music

Music was generated by iteratively altering component frequency intensities to maximize aesthetic pleasure as perceived by the network. These modifications took place in a series of "frames," which consisted of the 200×87px output canvas visible to the network. Each frame was generated in a series of smaller timesteps, which consisted of a vector of pixels sliced along the time domain.

Timesteps and frames were generated from left to right in a quasi-evolutionary model. An initial population of 16 frames were seeded

with random patterns of frequency intensities in the first timestep t=0. Then, each subsequent iteration altered the current timestep to produce 16 more candidate frames per frame in the population. These were scored, and the top 32 were selected to continue to the next timestep. When the last timestep in the frame was reached, the highest scoring frame was selected and appended to the final song; then the cursor was reset so that the next iteration examined the first timestep of the frame again. Figure 8 shows a sample portion of the final spectrogram for a typical generated song, and Figure 9 displays a full song. A high-level overview of this process was illustrated in Figure 2.

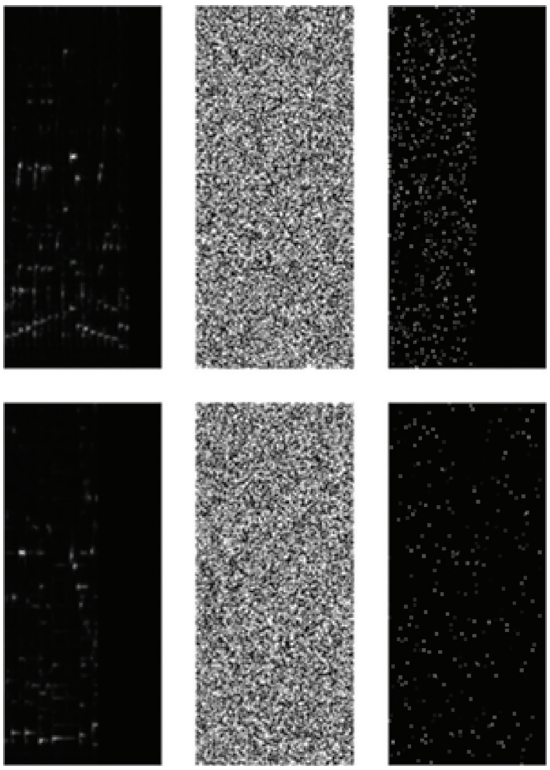

**Figure 7: Sample training slices. From left to right: music, fully-random noise, random tonal noice. Both positive and negative samples were randomly truncated to accustom the network to discerning the two even during our left-to-right generation scheme.**

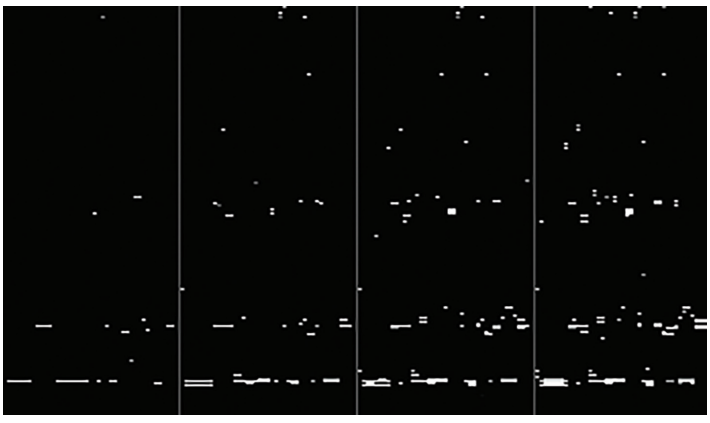

**Figure 8: A sample portion of the final spectrogram for a typical generated song. Gray bars have been added to highlight frame boundaries. Note how iterative improvement is visible across the frames.**

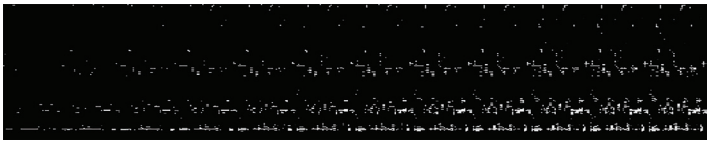

**Figure 9: A 12-frame song generated by Korsak Ovbot.**

This process of iterative improvement closely mimics the desired creative process of experimentally producing melodies and editing them until they sound better. The method of generating modifications in a Monte Carlo fashion is fast and also allows for novel, non-deterministic output. Finally, the evolutionary technique of keeping a selection of the top performers instead of exclusively selecting the single best choice allows for better long-run melodies to prevail over short-term maxima.

## Results

When converted to audio files, the spectrograms created by Korsak contain short melodies and harmonics that sound like music, in that they conform with aural patterns which humans find agreeable. In listening to the output, we find that that later frames do eventually settle upon a musical phrase, which has a concrete conclusion instead of an unending walk of melody. Finally, the output music seems to agree with the tonal and rhythmic style of its source material, "Monk's Dream" by Thelonius Monk (archive.org link available in reference).

We suggest that these features are because Korsak has successfully learned some heuristic rules for music from its training. In Figure 10, we compare a frame generated by Korsak to a sample training frame. Notice how the training frame can roughly be divided into three ranges of notes along the vertical frequency axis, comprised of bass, mid-range, and high-range notes. It appears that Korsak has learned to favor notes in these regions as well. Furthermore, we observe that the training samples often contained steady up and down progressions, including arpeggiated progressions,

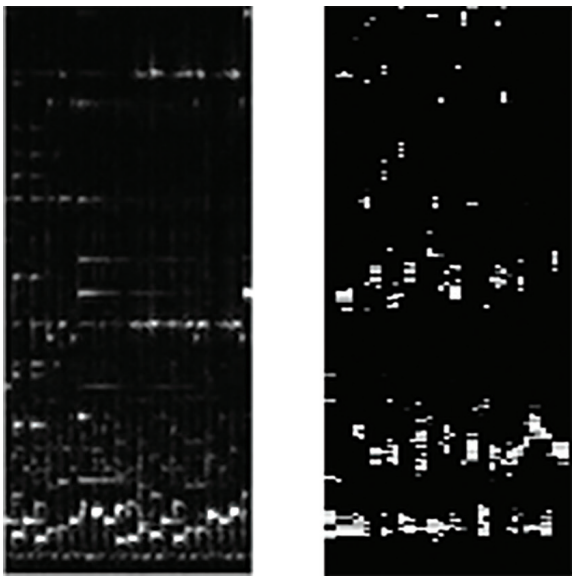

**Figure 10: Comparison between actual music (left) and generated music (right). Contrast has been increased to improve visibility. Notice the three general areas of tones near the top, middle, and bottom of both spectrograms.**

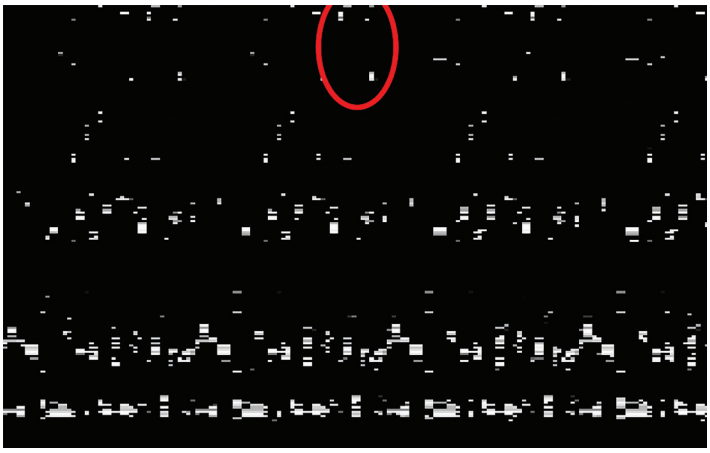

**Figure 11: Highlighted recurrent motif across generated samples.**

visible in the mid-range and bass portions of the spectrogram. It appears that Korsak has also learned to prefer these patterns. Overall, it seems that Korsak is able to successfully select evolved melodies in a way that, at least remotely, appeals to humans.

We do observe some shortcomings with this iterative method, however. Certain patterns which have extremely high appeal may appear disproportionately in the generated music. Korsak currently suffers from this by including a pattern with the same tonal and rhythmic motif in all its generated songs. This motif is highlighted in Figure 11. Although this motif does sound pleasing, and although Korsak's stochastic generative method does produce significant and interesting variations around this theme, it unfortunately stumbles upon this theme in almost every run, since it has such a high appeal that it dominates the evolution of the melody.

There are a number of possible ways to remedy this. If this behavior is due to over-training, then it may be solved with a more varied training set and higher dropout. If this is because maximizing the appeal of a sample tends towards a specific pattern of music, then perhaps a larger network can support many maxima for different variations of music. Alternatively, the recurrent pattern and rhythm might only seem uncreative because it appears in the same location in every generated frame—if this is the case, then randomly shifting the frame during generation may allow Korsak to focus more on musical patterns rather than absolute positions of notes within the sample. Finally, if this is just a problem of local maxima, then more melodies may be explored concurrently by increasing the population size and by randomly including lower-scoring melodies.

## How creative is this, really?

We offer an argument that, by mimicking the human creative process, the generative method employed in Korsak Ovbot is more "creative" than past procedural and pure neural approaches to music generation. The metric we use to compare this is the degree to which the aesthetic features of the generated music may be attributed to the algorithm, as opposed to the algorithm creators or the human composers of the training material.

To what degree can we attribute the pleasant products of a procedural algorithm to the algorithm itself? Since the key constraints, formulas, and preferences in such an algorithm are hard-coded by its developers, it seems that much of the credit for whatever music is produced belongs to the developers. What about a purely neural approach? Since these can only remix patterns found in the training data it seems that much of the credit here goes to the composers of the music in the training data.

Korsak Ovbot's approach seems to at least partially detach it from both of these human sources. By evolving the melody by random mutation, the final product is causally disconnected from the training material; that is, in theory, it is possible for Korsak to happen upon a brand new melody never before seen by humans that is still pleasurable because it satisfies the aesthetic function. At the same time, because the aesthetic function is learned rather than hard-coded, the preferences it captures are not limited by those of its developers; in fact, it is possible for it to discover subtle patterns for which even we humans are unaware.

From these observations, it seems that Korsak Ovbot occupies an interesting new position in artistic software. It seems hard to say that its music should be credited to the developers, since the developers had no role in specifying the music. Yet, it seems equally difficult to say that the credit belongs to the composers, since the algorithm could conceivably create novel melodies. Furthermore, it is not purely random, since the output artwork is the result of optimization by an aesthetic function, such that it contains discernible musical features. Although it would be too quick to say that Korsak Ovbot ought to take credit for the creativity here—for how can it exhibit creativity without awareness of its work?—we argue that the approach it employs allows for more "creative liberty" than possible with just a procedural algorithm or just a neural network. In this regard, we venture to say that Korsak Ovbot might exhibit a rudimentary but promising approach to artificial creativity.

## Conclusion

In Korsak Ovbot, we have explored a novel approach to artificial creativity. It is able to generate music that seems to contain recognizable melody, produced via a method which seeks to mimic the human creative process. Consequently, it seems that the resulting artwork is less attributable to either the developers or the composers of the source material when compared to procedural or pure neural approaches; we claim that this is an indication that it exhibits a higher degree of "creative liberty" than those approaches.

There are a number of possible avenues to improve upon the methods tested in Korsak Ovbot. It is worth exploring ways to prompt Korsak to generate different varieties of music by expanding its training exposure. It would also be interesting to train Korsak on significantly longer frames to try to accustom it to compositional plot. Finally, it might be worth investigating whether using instruments rather than just tones can allow Korsak to produce music that is more accessible to humans.

On that topic, we wish to identify a unique genre for Korsak's music. The iterative process of modifying the melody across repetitions to discover better-sounding combinations sounds similar to the improvisational process in jazz. Yet, its sounds are robotic and slightly random in a way that is distinct to computer-generated music. Because of this, we propose that the genre of Korsak Ovbot's music be designated as "Robo-Jazz."

We are quite satisfied with Korsak Ovbot's results. To demonstrate the endless variations which Korsak is capable of, and to make its works available for enjoyment by the public, we have created a website for it at *https://korsak.ovbot.net*. This page is updated

regularly with freshly-generated, fully original music. The future of such a site is promising: with speed improvements, Korsak may be able to generate music in real-time; and with further training on more accessible genres, it may be able to produce automatic background music to liven-up work or social settings. In any case, we are excited by what prospects future work might bring, and we look forward to enjoying the artwork of Korsak Ovbot and friends.

## Acknowledgements

The author would like to thank Professor Venkatesh Murthy for an amazing seminar that inspired many of the ideas which made this project possible. He would also like to thank his roommate, Sebastian Lindner-Liaw, for tolerating many hours of random blips and blops.

## References

**Colombo, Florian & Seeholzer, Alexander & Gerstner, Wulfram (2017). Deep Artificial Composer: A Creative Neural Network Model for Automated Melody Generation. EvoMUSART 2017, 81-96. DOI: 10.1007/978-3-319-55750-2_6.**

**Huang, C. K. (2017). Computer evolves to generate baroque music! YouTube. https://www.youtube.com/watch?v=SacogDL_4JU. Accessed 2018, May 10.**

**Mace, Mary-Anne & Ward, Tony (2002). Modeling the Creative Process: A Grounded Theory Analysis of Creativity in the Domain of Art Making. Creativity Research Journal 14:2, 179-192. DOI: 10.1207/S15326934CRJ1402_5**

**Mordvintsev, A., & Tyka, M. (2015). Inceptionism: Going deeper into neural networks. Google AI Blog. https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html. Accessed 2018, May 10.**

**Oord, A. V. D., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., ... & Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. arXiv preprint arXiv:1609.03499.**

**Scirea M., Togelius J., Eklund P., Risi S. (2016). MetaCompose: A Compositional Evolutionary Music Composer. EvoMUSART 2016, 202-217. DOI: 10.1007/978-3-319-31008-4_14**

**Shen, J., & Pang, R. (2017). Tacotron 2: Generating human-like speech from text. Google AI Blog. https://ai.googleblog.com/2017/12/tacotron-2-generating-human-like-speech.html. Accessed 2018, May 10.**

**Thelonius Monk Quartet (1963). Larga duracion - Monk's Dream (thelonius monk quartet). Internet Archive. https://archive.org/details/LargaDuracion3MonksDreamTheloniusMonkQuartet. Accessed 2018, May 10.**