Algorithmic music composition has developed a lot in the last few years, but the idea has a long history. In some sense, the first automatic music came from nature: [Chinese windchimes](https://en.wikipedia.org/wiki/Wind_chime#Eastern_and_Southern_Asia), ancient Greek wind-powered [Aeolian harps](https://en.wikipedia.org/wiki/Aeolian_harp), or the Japanese water instrument [suikinkutsu](https://en.wikipedia.org/wiki/Suikinkutsu" \t "_blank). But in the 1700s automatic music became “algorithmic”: [Musikalisches Würfelspiel](https://en.wikipedia.org/wiki/Musikalisches_W%C3%BCrfelspiel" \t "_blank), a game that generates short piano compositions from fragments, with choices made by dice.

Algorithmic composition could be described as “a sequence (set) of rules (instructions, operations) for solving (accomplishing) a [particular] problem (task) [in a finite number of steps] of combining musical parts (things, elements) into a whole (composition)”, (Cope, 1993)1. From this definition we can see that it is not necessary to use computers for algorithmic composition as we often infer; Mozart did not when he described the Musical Dice

Game. The concept of algorithmic composition is not something new. Pythagoras (around 500 B.C.) believed that music and mathematics were not separate studies. Hiller and Isaacson (1959) were probably the first who used a computational model using random number generators and Markov chains for algorithmic composition. Since then many researchers have tried to address the problem of algorithmic composition from different points of view. We categorize, based on their most prominent feature, as follows:

* Mathematical models
* Knowledge based systems
* Grammars
* Evolutionary methods
* Systems which learn, and
* Hybrid systems

The categorization is not straightforward since many of the AI methods can be considered as equivalent: for example, Markov chains are similar to type-3 grammars (Chomsky, 1957)2. Furthermore, some of the systems have more than one prominent feature, for example EMI (see below) was categorized as a grammar, but it can also be seen as a knowledge-based system or even a system which learns. In such cases we chose the method which was more responsible for the generation of the musical output.

An interesting musical game, **Musikalisches Würfelspiel** (musical dice game) has often been attributed to Mozart. The basis of the musical dice game consists of 272 musical measures and a table of rules used to select specific measures given a certain dice roll.  The result is a randomly selected 16 bar minuet and 16 bar trio.

The famous version of the dice game attributed to Mozart, was first published only after his death in 1793 by J.J. Hummel in Berlin-Amsterdam, and afterwards several times in different forms. While this challenging idea had been known and tried out by other composers before Mozart, it was Mozart's game which became famous and successful.

Though neither the original manuscript of the "*Musikalisches Würfelspiel*" nor direct references to it by Mozart were ever found, his authorship was never really questioned by publishers or musicologists.

A game with identical musical sections was described in the February 1787 issue of a publications called Journal des Luxus und der Moden (Journal of Luxury and Fashions!).W. A. Mozart may have been connected with or influenced by this publication.... the later dice game attributed to him contains a rule table which is identical to the minuet table in the publication.

A close up of a newspaper

Description automatically generated

All possible choices were given by Mozart in such a way that by any selection the resulting melody is a pretty minuet fulfilling the harmonic and compositional requirements of minuets of that time.

According to Lawrence Zbikowski, "In truth, chance played little part in the success of the music produced by such games. Instead, what was required of the compilers...[was] a little knowledge about how to put the game together and an understanding of the formal design of waltzes, etc."[[3]](https://en.wikipedia.org/wiki/Musikalisches_W%C3%BCrfelspiel#cite_note-3)

According to Stephen Hedges, "The 'galant' middle class in Europe was playing with mathematics. In this atmosphere of investigation and cataloguing, a systematic device that would seem to make it possible for anyone to write music was practically guaranteed popularity.[[4]](https://en.wikipedia.org/wiki/Musikalisches_W%C3%BCrfelspiel#cite_note-4)

According to [Leonard Meyer](https://en.wikipedia.org/wiki/Leonard_Meyer), "Eighteenth-century composers constructed musical dice games while nineteenth century composers did not. ... [W]hat constrained the choice of figures [in seventeenth- and eighteenth-century music] were the claims of taste, coherent expression and propriety, given the genre of work being composed, rather than the inner necessity of a gradually unfolding, underlying process [as in nineteenth century music]".

The way these games work may be understood in analogy to [sentence construction](https://en.wikipedia.org/wiki/Generative_grammar).

* The cow ran past the field.
* The pig walked through the yard.
* The sheep ran into the marsh.

One rolls one die for each word and selects the word from the appropriate column according to the number. Thus if one rolls 1 2 3 1 2 3 one is given, "The pig ran past the marsh." Each progression is essentially the same, there may be more or less choices for different slots, and the choices offered for each slot are slight [variations](https://en.wikipedia.org/wiki/Variation_(music)) rather than being entirely different.

The most well-known was published in 1792, by [Mozart](https://en.wikipedia.org/wiki/Wolfgang_Amadeus_Mozart)'s publisher [Nikolaus Simrock](https://en.wikipedia.org/wiki/Nikolaus_Simrock) in Berlin ([K.](https://en.wikipedia.org/wiki/K%C3%B6chel_catalogue) 294dK3 or K. 516fK6). The game was attributed to Mozart, but this attribution has not been authenticated.[[6]](https://en.wikipedia.org/wiki/Musikalisches_W%C3%BCrfelspiel#cite_note-6) The dice rolls randomly selected small sections of music, which would be patched together to create a musical piece. This game is capable of producing 1116 = 45,949,729,863,572,161 different yet similar waltzes.[[7]](https://en.wikipedia.org/wiki/Musikalisches_W%C3%BCrfelspiel#cite_note-7) Some measures have only one possibility no matter what the roll of the dice (measure 8/16) while other measures have a different possibility for each roll (measure 1/16).[[8]](https://en.wikipedia.org/wiki/Musikalisches_W%C3%BCrfelspiel#cite_note-8)

Mozart's manuscript, written in 1787, consisting of 176 one-[bar](https://en.wikipedia.org/wiki/Bar_(music)) fragments of music,[[9]](https://en.wikipedia.org/wiki/Musikalisches_W%C3%BCrfelspiel#cite_note-9) appears to be some kind of game or system for constructing music out of two-bar fragments, but contains no instructions and there is no evidence that dice were involved.

The titles of the supposed Mozart compositions are:[[1]](https://en.wikipedia.org/wiki/Musikalisches_W%C3%BCrfelspiel#cite_note-Algorithmic-1)

Anleitung zum Componieren von Walzern so viele man will vermittelst zweier Würfel, ohne etwas von der Musik oder Composition zu verstehen (German for "Instructions for the composition of as many waltzes as one desires with two dice, without understanding anything about music or composition")

Anleitung zum Componieren von Polonaisen... (German for "Instructions for the composition of polonaises...")

[Robert Xavier Rodríguez](https://en.wikipedia.org/wiki/Robert_Xavier_Rodr%C3%ADguez) composed his Musical Dice Game for string orchestra based on K. 516f.[[10]](https://en.wikipedia.org/wiki/Musikalisches_W%C3%BCrfelspiel#cite_note-10)

The attribution of [Joseph Haydn](https://en.wikipedia.org/wiki/Joseph_Haydn)'s Gioco filarmonico o sia maniera facile per comporre un infinito numero de minuetti e trio anche senza sapere il contrappunto (Italian for "The game of harmony, or an easy method for composing an infinite number of minuet-trios, without any knowledge of counterpoint") has also not been authenticated.[[1]](https://en.wikipedia.org/wiki/Musikalisches_W%C3%BCrfelspiel#cite_note-Algorithmic-1)

[Markov chains](https://en.wikipedia.org/wiki/Markov_chain), formalized in the early 1900s to model probabilistic systems, can also be used to generate new musical compositions. They take the motivations behind the dice game a step further, in two ways. First, Markov chains can be built from existing material rather than needing fragments explicitly composed as interchangeable components. Second, instead of assuming fragments have equal probabilities, Markov chains encode the variation in probabilities with respect to context.

A close up of a logo

Description automatically generated

[Iannis Xenakis](https://en.wikipedia.org/wiki/Iannis_Xenakis) used Markov chains in his 1958 compositions, “[Analogique](https://www.youtube.com/watch?v=mXIJO-af_u8" \t "_blank)”. He describes his process in “[Formalized Music: Thought and Mathematics in Composition](https://monoskop.org/images/7/74/Xenakis_Iannis_Formalized_Music_Thought_and_Mathematics_in_Composition.pdf)”, down to the details of transition matrices that define the probabilities of certain notes being produced.

A screenshot of a cell phone

Description automatically generated

An excerpt from Chapter 3, “Markovian Stochastic Music: Applications”.

In 1981 [David Cope](https://en.wikipedia.org/wiki/David_Cope) began working with algorithmic composition to [solve his writers block](http://artsites.ucsc.edu/faculty/cope/experiments.htm). He combined Markov chains and other techniques (musical grammars and combinatorics) into a semi-automatic system he calls Experiments in Musical Intelligence, or Emmy. David cites Iannis Xenakis and Lejaren Hiller ([Illiac Suite](http://www.musicainformatica.org/topics/illiac-suite.php" \t "_blank) 1955, [Experimental Music](https://archive.org/details/experimentalmusi00hill) 1959) as early inspirations, and he describes Emmy in [papers](http://quod.lib.umich.edu/cgi/p/pod/dod-idx/experiments-in-music-intelligence-emi.pdf?c=icmc;idno=bbp2372.1987.025), [patent](https://www.google.com/patents/US7696426)s, and even source code on [GitHub](https://github.com/HeinrichApfelmus/david-cope-cmmc). Emmy is most famous for learning from and imitating other composers.

While Markov chains trained on a set of compositions [can only produce subsequences](http://nbviewer.jupyter.org/gist/yoavg/d76121dfde2618422139) that also exist in the original data, [recurrent neural networks](http://karpathy.github.io/2015/05/21/rnn-effectiveness/) (RNNs) attempt to extrapolate beyond those exact subsequences. In 1989 the [first attempts to generate music with RNNs](http://www.indiana.edu/~abcwest/pmwiki/pdf/todd.compmusic.1989.pdf), developed first by [Peter M. Todd, then Michael C. Mozer and others](https://mitpress.mit.edu/books/music-and-connectionism), were limited by their short-term coherence.

In 2002 Doug Eck [updated this approach](http://www.iro.umontreal.ca/~eckdoug/papers/2002_ieee.pdf) by switching from standard RNN cells to “long short term memory” (LSTMs) cells. Doug used his architecture to [improvise blues](http://www.iro.umontreal.ca/~eckdoug/blues/index.html) based on a short recording. He writes, “Remarkably […] LSTM is able to play the blues with good timing and proper structure as long as one is willing to listen.”

Doug now leads the [Magenta team](https://magenta.tensorflow.org/welcome-to-magenta) at [Google Brain](https://research.google.com/teams/brain/), where they have been [developing and sharing code](https://github.com/tensorflow/magenta/tree/master/magenta/models) related to machine learning & creativity since early 2016. Magenta has applied Doug’s LSTM-based approaches to [drum pattern generation](https://github.com/tensorflow/magenta/tree/master/magenta/models/drums_rnn), [melody generation](https://github.com/tensorflow/magenta/tree/master/magenta/models/melody_rnn), and [polyphonic music generation](https://github.com/tensorflow/magenta/tree/master/magenta/models/polyphony_rnn). They’ve built systems that [improvise duets](https://aiexperiments.withgoogle.com/ai-duet/) with human performers, and tools that generate [expressive dynamics and timing](https://magenta.tensorflow.org/performance-rnn) along with the polyphonic compositions. Initially, Magenta released examples using TensorFlow in Python with the hope that artists and musicians would explore these demos. In 2018 with the release [TensorFlow.js](https://js.tensorflow.org/) they have started to promote more [interactive demos](https://magenta.tensorflow.org/demos) in JavaScript and even [plugins for Ableton Live](https://magenta.tensorflow.org/studio). Two favorites: [multitrack VAE](https://magenta.tensorflow.org/multitrack) for interpolating between short melodic loops, and [beat blender](https://experiments.withgoogle.com/ai/beat-blender/view/) for interpolating between short drum loops.

A big leap in compositional complexity came out of Magenta in September 2018 with [Music Transformer](https://arxiv.org/abs/1809.04281v2) by Huang et al. Unlike Performance RNN, [the samples](https://storage.googleapis.com/music-transformer/index.html) from Music Transformer do not succumb to chaos after the first few measures. They trained on Bach chorales (without dynamics) as well as a piano competition data (with dynamics).

One of the recurring difficulties encountered when training these systems is deciding on a representation of music. Designing an encoding for a RNN might start with a metaphor of text: the RNN is processing a sequence of states (letters) unfolding over time or space (the page). But unlike text, a single moment in music can contain more than one symbol: it can be a chord, or it can have a combination of qualities that is best described by its components. There can also be long durations of silence, or states can have wildly varying lengths. These differences may be resolved by carefully crafting the representation, or by heavily augmenting the dataset and designing the architecture with the capacity to learn all the invariance.

Another significant challenge with data-driven algorithmic composition is: what data to use? Whose music counts? When any automated creative system needs to be trained on a large number of cultural artifacts, it can only perpetuate the dominance of what is already well-documented. In music, this means a lot of Bach, Beethoven, and other old white European men. (Two exceptions: some English and Irish [folk](https://thesession.org/) [music](http://ifdo.ca/~seymour/nottingham/nottingham.html), and some [video game music](https://arxiv.org/abs/1806.04278).) The data is also selected by machine learning researchers, who are also a relatively homogenous group (though decreasingly so).

While LSTMs and Transformers manage to maintain long-term consistency better than a standard RNN or Markov chain, there is still a gap between generating shorter phrases and generating an entire composition; something that has not yet been bridged without lots of tricks and hand-tuning. Startups like [Jukedeck](https://www.jukedeck.com/" \t "_blank), [Aiva](http://www.aiva.ai/" \t "_blank), [Amper](https://www.ampermusic.com/" \t "_blank), and others are trying to fill this space of on-demand, hand-tuned formulaic generative music. Some going so far as to produce [entire pop albums](https://www.youtube.com/watch?v=XUs6CznN8pw) as marketing. Big companies are getting in on the action, too. François Pachet, formerly at Sony Computer Science Laboratories and [now at Spotify](https://www.musicbusinessworldwide.com/welcome-future-spotify-poaches-ai-music-expert-sony/), has been working with algorithmic music for some time, from his [Continuator](http://francoispachet.fr/continuator/continuator.html) to the more recent [Flow Machines](http://www.flow-machines.com/).

[Eduardo Reck Miranda](https://en.wikipedia.org/wiki/Eduardo_Reck_Miranda), a composer and researcher previously at Sony CSL, has released an entire album of “computer-aided symphonic works” called “[Mind Pieces, Sound to Sea](https://open.spotify.com/album/0yNVAgeM7P8X488xDyEIEb?si=o3AY8LIHTjq5aBdXauqAKQ)” through an [otherwise traditional label](https://davinci-edition.com/product/c00107/) specializing in classical and jazz. While the technologies behind groups like Sony CSL are proprietary, we can make some guesses based on the researchers involved. For example: it’s likely that Flow Machines has continued with the same approach as Continuator, more akin to David Cope than Doug Eck. (But for RNN-based approaches to “duets” and “continuations”, check out [Deep Musical Dialogue](https://www.youtube.com/watch?v=AiAzf2EUAR8) by Mason Bretan, and [AI Duet](https://experiments.withgoogle.com/ai/ai-duet) by Magenta.)

At IBM the Watson team has developed a system called [Watson Beat](https://soundcloud.com/ibmresearch/fallen-star-amped) that can produce complete tracks in a limited number of styles, based on a melodic prompt.

Autonomy versus Assistance

When talking about computer-based music generation, there is actually some ambiguity about whether the objective is to design and construct autonomous music-making systems – two recent examples being the deep-learning based AmperTM and Jukedeck systems/companies aimed at the creation of original music for commercials and documentary; or to design and construct computer-based environments to assist human musicians (composers, arrangers, producers, etc.) – two examples being the FlowComposer environment developed at Sony CSL-Paris [153] (introduced in Section 6.11.4) and the OpenMusic environment developed at IRCAM [3].

6 Musical instrument digital interface, to be introduced in Section 4.7.1.

7 One of the first documented case of stochastic music, long before computers, is the Musikalisches Wurfelspiel (Dice Music), attributed to Wolfgang Amadeus Mozart. It was designed for using dice to generate music by concatenating randomly selected predefined music segments composed in a given style (Austrian waltz in a given key).

The quest for autonomous music-making systems may be an interesting perspective for exploring the process of composition and it also serves as an evaluation method. An example of a musical Turing test will be introduced in Section 6.14.2. It consists in presenting to various members of the public (from beginners to experts) chorales composed by J. S. Bach or generated by a deep learning system and played by human musicians. As we will see in the following, deep learning techniques turn out to be very efficient at succeeding in such tests, due to their capacity to learn musical style from a given corpus and to generate new music that fits into this style. That said, we consider that such a test is more a means than an end. A broader perspective is in assisting human musicians during the various steps of music creation: composition, arranging, orchestration, production, etc. Indeed, to compose or to improvise, a musician rarely creates new music from scratch. S/he reuses and adapts, consciously or unconsciously, features from various music that s/he already knows or has heard, while following some principles and guidelines, such as theories about harmony and scales. A computer-based musician assistant may act during different stages of the composition, to initiate, suggest, provoke and/or complement the inspiration of the human composer. That said, as we will see, the majority of current deep-learning based systems for generating music are still focused on autonomous generation, although more and more systems are addressing the issue of human-level control and interaction.

The first music generated by computer appeared in 1957. It was a 17 seconds long melody named “The Silver Scale” by its author Newman Guttman and was generated by a software for sound synthesis named Music I, developed by Mathews at Bell Laboratories. The same year, “The Illiac Suite” was the first score composed by a computer. It was named after the ILLIAC I computer at the University of Illinois at Urbana-Champaign (UIUC) in the United States. The human “meta-composers” were Lejaren A. Hiller and Leonard M. Isaacson, both musicians and scientists. It was an early example of algorithmic composition, making use of stochastic models (Markov chains) for generation as well as rules to filter generated material according to desired properties. In 1960, Russian researcher R. Kh. Zaripov published worldwide first paper on algorithmic music composing using the "[Ural-1](https://en.wikipedia.org/wiki/Ural_(computer))" computer. In 1965, inventor Ray Kurzweil premiered a piano piece created by a computer that was capable of pattern recognition in various compositions. The computer was then able to analyze and use these patterns to create novel melodies. The computer was debuted on Steve Allen's I've Got a Secret program, and stumped the hosts until film star Henry Morgan guessed Ray's secret.

In the domain of sound synthesis, a landmark was the release in 1983 by Yamaha of the DX 7 synthesizer, building on groundwork by Chowning on a model of synthesis based on frequency modulation (FM). The same year, the MIDI6 interface was launched, as a way to interoperate various software and instruments (including the Yamaha DX 7 synthesizer). Another landmark was the development by Puckette at IRCAM of the Max/MSP real-time interactive processing environment, used for real-time synthesis and for interactive performances.

Regarding algorithmic composition, in the early 1960s Iannis Xenakis explored the idea of stochastic composition7 [209], in his composition named “Atr´ees” in 1962. The idea involved using computer fast computations to calculate various possibilities from a set of probabilities designed by the composer in order to generate samples of musical pieces to be selected. In another approach following the initial direction of “The Illiac Suite”, grammars and rules were used to specify the style of a given corpus or more generally tonal music theory. An example is the generation in the 1980s by Ebcio˘glu’s composition software named CHORAL of a four-part chorale in the style of Johann Sebastian Bach, according to over 350 handcrafted rules [42]. In the late 1980s David Cope’s system named Experiments in Musical Intelligence (EMI) extended that approach with the capacity to learn from a corpus of scores of a composer to create its own grammar and database of rules [27]. Since then, computer music has continued developing for the general public, if we consider, for instance, the GarageBand music composition and production application for Apple platforms (computers, tablets and cellphones), as an offspring of the initial Cubase sequencer software, released by Steinberg in 1989.

Computational creativity in the music domain has focused both on the generation of musical scores for use by human musicians, and on the generation of music for performance by computers. The domain of generation has included classical music (with software that generates music in the style of [Mozart](https://en.wikipedia.org/wiki/Mozart) and [Bach](https://en.wikipedia.org/wiki/Bach)) and [jazz](https://en.wikipedia.org/wiki/Jazz).[[50]](https://en.wikipedia.org/wiki/Computational_creativity#cite_note-50) Most notably, [David Cope](https://en.wikipedia.org/wiki/David_Cope)[[51]](https://en.wikipedia.org/wiki/Computational_creativity#cite_note-51) has written a software system called "Experiments in Musical Intelligence" (or "EMI")[[52]](https://en.wikipedia.org/wiki/Computational_creativity#cite_note-52) that is capable of analyzing and generalizing from existing music by a human composer to generate novel musical compositions in the same style. EMI's output is convincing enough to persuade human listeners that its music is human-generated to a high level of competence.[[53]](https://en.wikipedia.org/wiki/Computational_creativity#cite_note-53)

In the field of contemporary classical music, [Iamus](https://en.wikipedia.org/wiki/Iamus_(computer)" \o "Iamus (computer)) is the first computer that composes from scratch, and produces final scores that professional interpreters can play. The [London Symphony Orchestra](https://en.wikipedia.org/wiki/London_Symphony_Orchestra) played a piece for full orchestra, included in [Iamus' debut CD](https://en.wikipedia.org/wiki/Iamus_(album)" \o "Iamus (album)),[[54]](https://en.wikipedia.org/wiki/Computational_creativity#cite_note-54) which [New Scientist](https://en.wikipedia.org/wiki/New_Scientist) described as "The first major work composed by a computer and performed by a full orchestra".[[55]](https://en.wikipedia.org/wiki/Computational_creativity#cite_note-55) [Melomics](https://en.wikipedia.org/wiki/Melomics" \o "Melomics), the technology behind Iamus, is able to generate pieces in different styles of music with a similar level of quality.

Creativity research in jazz has focused on the process of improvisation and the cognitive demands that this places on a musical agent: reasoning about time, remembering and conceptualizing what has already been played, and planning ahead for what might be played next.[[56]](https://en.wikipedia.org/wiki/Computational_creativity#cite_note-56) The robot Shimon, developed by Gil Weinberg of Georgia Tech, has demonstrated jazz improvisation.[[57]](https://en.wikipedia.org/wiki/Computational_creativity#cite_note-57) Virtual improvisation software based on researches on stylistic modeling carried out by Gerard Assayag and Shlomo Dubnov include OMax, SoMax and PyOracle, are used to create improvisations in real-time by re-injecting variable length sequences learned on the fly from live performer.[[58]](https://en.wikipedia.org/wiki/Computational_creativity#cite_note-58)

In 1994, a Creativity Machine architecture (see above) was able to generate 11,000 musical hooks by training a synaptically perturbed neural net on 100 melodies that had appeared on the top ten list over the last 30 years. In 1996, a self-bootstrapping Creativity Machine observed audience facial expressions through an advanced machine vision system and perfected its musical talents to generate an album entitled "Song of the Neurons"[[59]](https://en.wikipedia.org/wiki/Computational_creativity#cite_note-59)

In the field of musical composition, the patented works[[60]](https://en.wikipedia.org/wiki/Computational_creativity#cite_note-60) by [René-Louis Baron](https://en.wikipedia.org/wiki/Ren%C3%A9-Louis_Baron) allowed to make a robot that can create and play a multitude of orchestrated melodies so-called "coherent" in any musical style. All outdoor physical parameter associated with one or more specific musical parameters, can influence and develop each of these songs (in real time while listening to the song). The patented invention [Medal-Composer](https://en.wikipedia.org/wiki/Ren%C3%A9-Louis_Baron#Inventor.27s_path) raises problems of copyright.

Deep Learning for Music Generation

In addition to traditional tasks such as prediction, classification and translation, deep learning is receiving growing attention as an approach for music generation, as witnessed by recent research groups such as Magenta at Google and CTRL (Creator Technology Research Lab) at Spotify. The motivation is in using the capacity of deep learning architectures and training techniques to automatically learn musical styles from arbitrary musical corpora and then to generate samples from the estimated distribution. However, a direct application of deep learning to generate content rapidly reaches limits as the generated content tends to mimic the training set without exhibiting true creativity. Moreover, deep learning architectures do not offer direct ways for controlling generation (e.g., imposing some tonality or other arbitrary constraints). Furthermore, deep learning architectures alone are autistic automata which generate music autonomously without human user interaction, far from the objective of interactively assisting musicians to compose and refine music. Issues such as: control, structure, creativity and interactivity are the focus of our analysis. In this paper, we select some limitations of a direct application of deep learning to music generation, analyze why the issues are not fulfilled and how to address them by possible approaches. Various examples of recent systems are cited as examples of promising directions.

The motivation for using deep learning, and more generally machine learning techniques, to generate musical content is its generality. As opposed to handcrafted models for, e.g., grammar-based [39] or rule-based music generation systems [8], a machine-learning-based generation system can automatically learn a model, a style, from an arbitrary corpus of music. Generation can then take place by using prediction (e.g., to predict the pitch of the next note of a melody) or classification (e.g., to recognize the chord corresponding to a melody), based on the distribution and correlations learnt by the deep model which represent the style of the corpus. As stated by Fiebrink and Caramiaux in [12], benefits are: 1) it can make creation feasible when the desired application is too complex to be described by analytical formulations or manual brute force design; 2) learning algorithms are often less brittle than manually-designed rule sets and learned rules are more likely to generalize accurately to new contexts in which inputs may change.

A direct application of deep learning architectures and techniques to generation, although it could produce impressing results2 , suffers from some limitations. We consider here3 : – Control, e.g., tonality conformance, maximum number of repeated notes, rhythm, etc.; – Structure, versus wandering music without a sense of direction; – Creativity, versus imitation and risk of plagiarism; – Interactivity, versus automated single-step generation.

A comprehensive survey and analysis by Briot et al. of deep learning techniques to generate musical content is available in a book [2]. In [21], Herremans et al. propose a function-oriented taxonomy for various kinds of music generation systems. Examples of surveys about of AI-based methods for algorithmic music composition are by Papadopoulos and Wiggins [36] and by Fern´andez and Vico [11], as well as books by Cope [3] and by Nierhaus [30]. In [17], Graves analyses the application of recurrent neural networks architectures to generate sequences (text and music). In [12], Fiebrink and Caramiaux address the issue of using machine learning to generate creative music. We are not aware of a comprehensive analysis dedicated to deep learning (and artificial neural networks techniques) that systematically analyzes limitations and challenges, solutions and directions, in other words that is problem-oriented and not just application-oriented.

Control Musicians usually want to adapt ideas and patterns borrowed from other contexts to their own objective, e.g., transposition to another key, minimizing the number of notes. In practice this means the ability to control generation by a deep learning architecture.

2.1 Dimensions of control strategies Such arbitrary control is actually a difficult issue for current deep learning architectures and techniques, because standard neural networks are not designed to be controlled. As opposed to Markov models which have an operational model where one can attach constraints onto their internal operational structure in order to control the generation4 , neural networks do not offer such an operational entry point. Moreover, the distributed nature of their representation does not provide a direct correspondence to the structure of the content generated. As a result, strategies for controlling deep learning generation that we will analyze have to rely on some external intervention at various entry points (hooks), such as: – Input; – Output; – Encapsulation/reformulation.

2.2 Sampling

Sampling a model to generate content may be an entry point for control if we introduce constraints on the output generation (this is called constraint sampling). This is usually implemented by a generate-and-test approach, where valid solutions are picked from a set of generated random samples from the model. As we will see, a key issue is how to guide the sampling process in order to fulfill the objectives (constraints), thus sampling will be often combined with other strategies. 2.3 Conditioning The strategy of conditioning (sometimes also named conditional architecture) is to condition the architecture on some extra conditioning information, which could be arbitrary, e.g., a class label or data from other modalities. Examples are: – a bass line or a beat structure, in the rhythm generation system [28]; – a chord progression, in the MidiNet architecture [42]; – a musical genre or an instrument, in the WaveNet architecture [31]; – a set of positional constraints, in the Anticipation-RNN architecture [18]. In practice, the conditioning information is usually fed into the architecture as an additional input layer. Conditioning is a way to have some degree of parameterized control over the generation process.

2.3.1 Example 1: WaveNet Audio Speech and Music Generation The WaveNet architecture by van der Oord et al. [31] is aimed at generating raw audio waveforms. The architecture is based on a convolutional feedforward network without pooling layer7 . It has been experimented on generation for three audio domains: multi-speaker, text-to-speech (TTS) and music. The WaveNet architecture uses conditioning as a way to guide the generation, by adding an additional tag as a conditioning input. Two options are considered: global conditioning or local conditioning, depending if the conditioning input is shared for all time steps or is specific to each time step. An example of application of conditioning WaveNet for a text-to-speech application domain is to feed linguistic features (e.g., North American English or Mandarin Chinese speakers) in order to generate speech with a better prosody. The authors also report preliminary experiments on conditioning music models to generate music given a set of tags specifying, e.g., genre or instruments. 2.3.2 Example 2: Anticipation-RNN Bach Melody Generation Hadjeres and Nielsen propose a system named Anticipation-RNN [18] for generating melodies with unary constraints on notes (to enforce a given note at a given time position to have a given value). The limitation when using a standard note-to-note iterative strategy for generation by a recurrent network is that enforcing the constraint at a certain time step may retrospectively invalidate the distribution of the previously generated items, as shown in [33]. The idea is to condition the recurrent network (RNN) on some information summarizing the set of further (in time) constraints as a way to anticipate oncoming constraints, in order to generate notes with a correct distribution. Therefore, a second RNN architecture8 , named Constraint-RNN, is used and it functions backward in time and ts outputs are used as additional inputs of the main RNN (named Token-RNN), resulting in the architecture shown at Figure 1, with:

Algorithmic music composition has developed a lot in the last few years, but the idea has a long history. In some sense, the first automatic music came from nature: [Chinese windchimes](https://en.wikipedia.org/wiki/Wind_chime#Eastern_and_Southern_Asia), ancient Greek wind-powered [Aeolian harps](https://en.wikipedia.org/wiki/Aeolian_harp), or the Japanese water instrument [suikinkutsu](https://en.wikipedia.org/wiki/Suikinkutsu" \t "_blank). But in the 1700s automatic music became “algorithmic”: [Musikalisches Würfelspiel](https://en.wikipedia.org/wiki/Musikalisches_W%C3%BCrfelspiel" \t "_blank), a game that generates short piano compositions from fragments, with choices made by dice.

An example session of Musikalisches Würfelspiel.

[Markov chains](https://en.wikipedia.org/wiki/Markov_chain), formalized in the early 1900s to model probabilistic systems, can also be used to generate new musical compositions. They take the motivations behind the dice game a step further, in two ways. First, Markov chains can be built from existing material rather than needing fragments explicitly composed as interchangeable components. Second, instead of assuming fragments have equal probabilities, Markov chains encode the variation in probabilities with respect to context.

A close up of a logo

Description automatically generated

“[Remixing Noon](https://chatbotslife.com/notes-on-remixing-noon-generative-text-and-markov-chains-84ff4ec23937)” by Rev Dan Catt: a visualization of one possible path through a Markov chain trained on prose.

[Iannis Xenakis](https://en.wikipedia.org/wiki/Iannis_Xenakis) used Markov chains in his 1958 compositions, “[Analogique](https://www.youtube.com/watch?v=mXIJO-af_u8" \t "_blank)”. He describes his process in “[Formalized Music: Thought and Mathematics in Composition](https://monoskop.org/images/7/74/Xenakis_Iannis_Formalized_Music_Thought_and_Mathematics_in_Composition.pdf)”, down to the details of transition matrices that define the probabilities of certain notes being produced.

A screenshot of a cell phone

Description automatically generated

An excerpt from Chapter 3, “Markovian Stochastic Music: Applications”.

“Analogique A and B” (1958–1959) by Iannis Xenakis.

In 1981 [David Cope](https://en.wikipedia.org/wiki/David_Cope) began working with algorithmic composition to [solve his writers block](http://artsites.ucsc.edu/faculty/cope/experiments.htm). He combined Markov chains and other techniques (musical grammars and combinatorics) into a semi-automatic system he calls Experiments in Musical Intelligence, or Emmy. David cites Iannis Xenakis and Lejaren Hiller ([Illiac Suite](http://www.musicainformatica.org/topics/illiac-suite.php" \t "_blank) 1955, [Experimental Music](https://archive.org/details/experimentalmusi00hill) 1959) as early inspirations, and he describes Emmy in [papers](http://quod.lib.umich.edu/cgi/p/pod/dod-idx/experiments-in-music-intelligence-emi.pdf?c=icmc;idno=bbp2372.1987.025), [patent](https://www.google.com/patents/US7696426)s, and even source code on [GitHub](https://github.com/HeinrichApfelmus/david-cope-cmmc). Emmy is most famous for learning from and imitating other composers.

A composition by David Cope using Emmy, in the style of a Chopin mazurka. More performances [here](http://artsites.ucsc.edu/faculty/cope/mp3page.htm).

While Markov chains trained on a set of compositions [can only produce subsequences](http://nbviewer.jupyter.org/gist/yoavg/d76121dfde2618422139) that also exist in the original data, [recurrent neural networks](http://karpathy.github.io/2015/05/21/rnn-effectiveness/) (RNNs) attempt to extrapolate beyond those exact subsequences. In 1989 the [first attempts to generate music with RNNs](http://www.indiana.edu/~abcwest/pmwiki/pdf/todd.compmusic.1989.pdf), developed first by [Peter M. Todd, then Michael C. Mozer and others](https://mitpress.mit.edu/books/music-and-connectionism), were limited by their short-term coherence.

Three short compositions in the style of Bach generated by the [CONCERT](http://www.cs.colorado.edu/~mozer/Research/Selected%20Publications/music.html) system by Michael C. Mozer.

In 2002 Doug Eck [updated this approach](http://www.iro.umontreal.ca/~eckdoug/papers/2002_ieee.pdf) by switching from standard RNN cells to “long short term memory” (LSTMs) cells. Doug used his architecture to [improvise blues](http://www.iro.umontreal.ca/~eckdoug/blues/index.html) based on a short recording. He writes, “Remarkably […] LSTM is able to play the blues with good timing and proper structure as long as one is willing to listen.”

More variations on this approach from Doug are available [here](http://www.iro.umontreal.ca/~eckdoug/blues/).

Doug now leads the [Magenta team](https://magenta.tensorflow.org/welcome-to-magenta) at [Google Brain](https://research.google.com/teams/brain/), where they have been [developing and sharing code](https://github.com/tensorflow/magenta/tree/master/magenta/models) related to machine learning & creativity since early 2016. Magenta has applied Doug’s LSTM-based approaches to [drum pattern generation](https://github.com/tensorflow/magenta/tree/master/magenta/models/drums_rnn), [melody generation](https://github.com/tensorflow/magenta/tree/master/magenta/models/melody_rnn), and [polyphonic music generation](https://github.com/tensorflow/magenta/tree/master/magenta/models/polyphony_rnn). They’ve built systems that [improvise duets](https://aiexperiments.withgoogle.com/ai-duet/) with human performers, and tools that generate [expressive dynamics and timing](https://magenta.tensorflow.org/performance-rnn) along with the polyphonic compositions. Initially, Magenta released examples using TensorFlow in Python with the hope that artists and musicians would explore these demos. In 2018 with the release [TensorFlow.js](https://js.tensorflow.org/) they have started to promote more [interactive demos](https://magenta.tensorflow.org/demos) in JavaScript and even [plugins for Ableton Live](https://magenta.tensorflow.org/studio). Two favorites: [multitrack VAE](https://magenta.tensorflow.org/multitrack) for interpolating between short melodic loops, and [beat blender](https://experiments.withgoogle.com/ai/beat-blender/view/) for interpolating between short drum loops.

Original post with more example audio available [here](https://magenta.tensorflow.org/performance-rnn). For endless music in your browser go [here](https://goo.gl/magenta/performancernn-demo).

A big leap in compositional complexity came out of Magenta in September 2018 with [Music Transformer](https://arxiv.org/abs/1809.04281v2) by Huang et al. Unlike Performance RNN, [the samples](https://storage.googleapis.com/music-transformer/index.html) from Music Transformer do not succumb to chaos after the first few measures. They trained on Bach chorales (without dynamics) as well as a piano competition data (with dynamics).

Piano-e-Competition Unconditioned Samples from [Music Transformer](https://arxiv.org/abs/1809.04281v2). More example audio available [here](https://storage.googleapis.com/music-transformer/index.html).

One of the recurring difficulties encountered when training these systems is deciding on a representation of music. Designing an encoding for a RNN might start with a metaphor of text: the RNN is processing a sequence of states (letters) unfolding over time or space (the page). But unlike text, a single moment in music can contain more than one symbol: it can be a chord, or it can have a combination of qualities that is best described by its components. There can also be long durations of silence, or states can have wildly varying lengths. These differences may be resolved by carefully crafting the representation, or by heavily augmenting the dataset and designing the architecture with the capacity to learn all the invariance.

Another significant challenge with data-driven algorithmic composition is: what data to use? Whose music counts? When any automated creative system needs to be trained on a large number of cultural artifacts, it can only perpetuate the dominance of what is already well-documented. In music, this means a lot of Bach, Beethoven, and other old white European men. (Two exceptions: some English and Irish [folk](https://thesession.org/) [music](http://ifdo.ca/~seymour/nottingham/nottingham.html), and some [video game music](https://arxiv.org/abs/1806.04278).) The data is also selected by machine learning researchers, who are also a relatively homogenous group (though decreasingly so).

While LSTMs and Transformers manage to maintain long-term consistency better than a standard RNN or Markov chain, there is still a gap between generating shorter phrases and generating an entire composition; something that has not yet been bridged without lots of tricks and hand-tuning. Startups like [Jukedeck](https://www.jukedeck.com/" \t "_blank), [Aiva](http://www.aiva.ai/" \t "_blank), [Amper](https://www.ampermusic.com/" \t "_blank), and others are trying to fill this space of on-demand, hand-tuned formulaic generative music. Some going so far as to produce [entire pop albums](https://www.youtube.com/watch?v=XUs6CznN8pw) as marketing. Big companies are getting in on the action, too. François Pachet, formerly at Sony Computer Science Laboratories and [now at Spotify](https://www.musicbusinessworldwide.com/welcome-future-spotify-poaches-ai-music-expert-sony/), has been working with algorithmic music for some time, from his [Continuator](http://francoispachet.fr/continuator/continuator.html) to the more recent [Flow Machines](http://www.flow-machines.com/).

“The Continuator” (2000) by François Pachet, is designed to “extend the technical ability of musicians with stylistically consistent, automatically learnt musical material”.

“Daddy’s Car” (2016) by Flow Machines, a research project at Sony CSL coordinated by François Pachet. Flow Machines plans to “research and develop Artificial Intelligence systems able to generate music autonomously or in collaboration with human artists”. The arrangement, lyrics, and production are by composer Benoît Carré.

[Eduardo Reck Miranda](https://en.wikipedia.org/wiki/Eduardo_Reck_Miranda), a composer and researcher previously at Sony CSL, has released an entire album of “computer-aided symphonic works” called “[Mind Pieces, Sound to Sea](https://open.spotify.com/album/0yNVAgeM7P8X488xDyEIEb?si=o3AY8LIHTjq5aBdXauqAKQ)” through an [otherwise traditional label](https://davinci-edition.com/product/c00107/) specializing in classical and jazz. While the technologies behind groups like Sony CSL are proprietary, we can make some guesses based on the researchers involved. For example: it’s likely that Flow Machines has continued with the same approach as Continuator, more akin to David Cope than Doug Eck. (But for RNN-based approaches to “duets” and “continuations”, check out [Deep Musical Dialogue](https://www.youtube.com/watch?v=AiAzf2EUAR8) by Mason Bretan, and [AI Duet](https://experiments.withgoogle.com/ai/ai-duet) by Magenta.)

At IBM the Watson team has developed a system called [Watson Beat](https://soundcloud.com/ibmresearch/fallen-star-amped) that can produce complete tracks in a limited number of styles, based on a melodic prompt.

Other researchers on the Watson team have [worked with Alex Da Kid](https://www.ibm.com/watson/music/) to suggest themes and inspiration for music based on data mined from social media and culture.

Dice games, Markov chains, and RNNs aren’t the only ways to make algorithmic music. Some machine learning practitioners explore alternative approaches like [hierarchical temporal memory](https://www.youtube.com/watch?v=2y4549AjgEE), or [principal components analysis](https://soundcloud.com/bwhitman/01-radiant-bells). But I’m focusing on neural nets because they are responsible for most of the big changes recently. (Though even within the domain of neural nets there are some directions I’m leaving out that have fewer examples, such as restricted Boltzmann machines for composing [4-bar jazz licks](https://www.cs.hmc.edu/~keller/jazz/improvisor/ICCCX-Bickerman-Bosley-Swire-Keller.pdf), [short variations on a single song](https://www.youtube.com/watch?v=ThEAko6Vi_A&NR=1), or hybrid [RNN-RBM models](http://deeplearning.net/tutorial/rnnrbm.html), or hybrid [autoencoder-LSTM models](https://www.youtube.com/watch?v=BbyvbO2F7ug), or even [neuroevolutionary strategies](http://maestrogenesis.org/)).

The power of RNNs wasn’t common knowledge until [Andrej Karpathy’s](https://twitter.com/karpathy)viral post “[The Unreasonable Effectiveness of Recurrent Neural Networks](http://karpathy.github.io/2015/05/21/rnn-effectiveness/)” in May 2015. Andrej showed that a relatively simple neural network called [char-rnn](https://github.com/karpathy/char-rnn) could reliably recreate the “look and feel” of any text, from Shakespeare to C++. The same way that the popularity of dice games was [buffeted by a resurgence of rationalism](https://www.jstor.org/stable/734136) and interest in mathematics, Andrej’s article came at a time when interest in neural networks was exploding, triggering a renewed interest in recurrent networks. Some of the first people to test Andrej’s code applied it to symbolic music notation.

“Eight Short Outputs” by Bob Sturm, using char-rnn and 23,000 ABC transcriptions of [Irish folk music](https://thesession.org/). He has also [lead groups to perform these compositions](https://highnoongmt.wordpress.com/2017/07/04/folk-rnn-at-the-qmul-ideas-festival-2017/).

By [Gaurav Trivedi](http://www.trivedigaurav.com/blog/machines-learn-to-play-tabla/), using char-rnn and 207 tabla rhythms.

Some people started with char-rnn as inspiration, but developed their own architecture specifically for working with music. Notable examples come from [Daniel Johnson](http://www.hexahedria.com/2015/08/03/composing-music-with-recurrent-neural-networks/) and [Ji-Sung Kim](https://deepjazz.io/).

Custom RNN architecture trained on [classical music for piano](http://www.piano-midi.de/).

[deepjazz](https://deepjazz.io/) uses the same architecture as char-rnn and trains on a single song.

[Christian Walder](http://users.cecs.anu.edu.au/~christian.walder/) uses LSTMs in a more unusual way: starting with a pre-defined rhythm, and asking the neural net to fill in the pitches. This provides a lot of the global structure that is otherwise usually missing, but heavily constrains the possibilities.

Example from “[Modeling Symbolic Music: Beyond the Piano Roll](https://arxiv.org/abs/1606.01368)” by Christian Walder, trained on Baroque sonatas.

While all the examples so far are based on symbolic representations of music, some enthusiasts pushed char-rnn to its limits by feeding it raw audio.

By Joseph L. Chu, trained on [30 minutes](https://www.reddit.com/r/MachineLearning/comments/3c87b3/training_a_rnn_with_audio_data/csv4bla/) of “[a Japanese pop rock band](https://www.reddit.com/r/MachineLearning/comments/3x7poc/generating_sound_with_recurrent_neural_nets/cy39off/)”.

By Priya Pramesi, trained on Joanna Newsom.

Unfortunately it seems that char-rnn is fundamentally limited in its capacity to abstract higher level representations of raw audio. The [most inspiring results](https://www.youtube.com/watch?v=0VTI1BBLydE) on audio turned out to be nothing more than noisy copies of the source material (some people explain this when sharing their work, see [SomethingUnreal modeling his own speech](https://www.youtube.com/watch?v=NG-LATBZNBs" \t "_blank)). In machine learning this is related to the concept of “overfitting”: when a model can recreate the training data faithfully, but can’t effectively generalize to anything novel that it hasn’t been trained on. During training, initially first the model performs poorly on both the training and novel data, then it starts to perform better on both. But if you let it train too long, it gets worse at generalizing to novel data at the expense of recreating the training data. Researchers stop the training just before hitting that point. But overfitting is not so clearly a “problem” in creative contexts, where recombination of existing material is a common strategy that is hard to distinguish from “generalization”. Some people like David Cope go so far as to say “[all music [is] essentially inspired plagiarism](https://www.theguardian.com/technology/2010/jul/11/david-cope-computer-composer)” (but he has also been accused of [publishing pseudoscience](http://slab.org/tmp/wiggins-cope.pdf) and [straight-up](https://www.youtube.com/watch?v=Lt7fEchgFrU&lc=z13deduomr31zdyag04cfbfpgqe4jp0yoos0k) [plagiarism](https://www.youtube.com/watch?v=DqNcnIkYM4s&lc=z12bcr0qklm1u13w004cj3qromunsjdism40k)).

In September 2016 [DeepMind](https://deepmind.com/) published their [WaveNet research](https://arxiv.org/abs/1609.03499" \t "_blank) demonstrating an architecture that can build higher level abstractions of audio[sample-by-sample](https://deepmind.com/blog/wavenet-generative-model-raw-audio/).

A close up of a device

Description automatically generated

Diagram of the dilated convolutions used in the WaveNet architecture.

A picture containing object, fireworks

Description automatically generated

WaveNet sample-by-sample probability distributions across the range of 8-bit values.

Instead of using a recurrent network to learn representations over time, they used a [convolutional network](http://cs231n.github.io/convolutional-networks/). Convolutional networks learn combinations of filters. They’re normally used for processing images, but WaveNet treats time like a spatial dimension.

[Samples of WaveNet](https://deepmind.com/blog/wavenet-generative-model-raw-audio/) trained on piano music from YouTube.

Looking into the background of the co-authors, there are some interesting predecessors to WaveNet.

* [Sander Dieleman](https://twitter.com/sedielem) is first author on [End-to-end learning for music audio](https://dl.dropboxusercontent.com/u/19706734/paper_pt.pdf) (2014), a rare and early example of processing raw audio sample-by-sample with a neural net; in this case for genre classification (first use of neural nets for this task was [five years earlier](https://papers.nips.cc/paper/3674-unsupervised-feature-learning-for-audio-classification-using-convolutional-deep-belief-networks)).
* [Aäron van den Oord](https://twitter.com/avdnoord) is first author on [Pixel Recurrent Neural Networks](https://arxiv.org/abs/1601.06759) (2016), introducing networks that generate images pixel-by-pixel.
* Alex Graves, besides having a long history working with speech and recurrent neural networks, showed a demo of end-to-end trained neural net generated [synthetic speech](https://youtu.be/-yX1SYeDHbg?t=2545) in March 2015.

One of my favorite things to emerge from the WaveNet research is this rough piano imitation by Sageev Oore, who was on sabbatical at Google Brain at the time.

Sageev Oore performs “[sample\_3.wav](https://storage.googleapis.com/deepmind-media/pixie/making-music/sample_3.wav)” by WaveNet

In April 2017, Magenta built on WaveNet to create [NSynth](https://magenta.tensorflow.org/nsynth" \t "_blank), a model for analyzing and generating monophonic instrument sounds. They created an NSynth-powered “[Sound Maker](https://experiments.withgoogle.com/ai/sound-maker)” experiment in collaboration with Google Creative Lab New York. I worked with the Google Creative Lab in London to build NSynth into an open-source portable MIDI synthesizer, called “NSynth Super”.

A picture containing photo, sitting, room

Description automatically generated

Demonstration of linear interpolation between two sounds compared to NSynth interpolation.

[NSynth Super](https://nsynthsuper.withgoogle.com/) (2018) by Google Creative Lab.

In February 2017 a team from Montreal lead by Yoshua Bengio published [SampleRNN](https://arxiv.org/abs/1612.07837" \t "_blank) (with [code](https://github.com/soroushmehr/sampleRNN_ICLR2017)) for sample-by-sample generation of audio using a set of recurrent networks in a hierarchical structure. This research was influenced by experiments from [Ishaan Gulrajani](https://github.com/igul222) who trained a hierarchical version of char-rnn on raw audio.

A close up of a map

Description automatically generated

Simplified snapshot of the SampleRNN architecture: a hierarchy of recurrent networks (tier 2 and 3) at slower time scales, combined with one standard neural network (tier 1) at the fastest time scale, all using the same upsampling ratio (4).

SampleRNN trained on all over a hundred hours of speech from a single person (the Blizzard dataset).

SampleRNN trained on all 32 of Beethoven’s piano sonatas.

By Richard Assar, trained on 32 hours of Tangerine Dream, using his [port of the original code](https://github.com/richardassar/SampleRNN_torch).

By [DADABOTS](http://dadabots.com/), trained on the album [Diotima by Krallice](https://krallice.bandcamp.com/album/diotima" \t "_blank), accepted to NIPS 2017.

Both SampleRNN and WaveNet take an unusually long time to train (more than a week), and without optimizations (like [fast-wavenet](https://github.com/tomlepaine/fast-wavenet)) they are many times slower than realtime for generation. To reduce the training and generation time researchers use audio at 16kHz and 8 bits.

But for companies like Google or Baidu, the primary application of audio generation is text to speech, where fast generation is essential. In March 2017 Google published their [Tacotron](https://arxiv.org/abs/1703.10135" \t "_blank) research, which generates audio frame-by-frame using a [spectral representation](https://en.wikipedia.org/wiki/Spectrogram) as an intermediate output step and a sequence of characters (text) as input.

A close up of a map

Description automatically generated

Tacotron architecture, showing a mixture of techniques including attention, bidirectional RNNs, and convolution.

The Tacotron [demo samples](https://google.github.io/tacotron/publications/tacotron/index.html) are similar to WaveNet, with some small discrepancies. In May 2017, Baidu built on the Tacotron architecture with their [Deep Voice 2](https://arxiv.org/abs/1705.08947) research, increasing the audio quality by adding some final stages specific to speech generation. Because generating audio from amplitude spectra requires a phase reconstruction step, the quality of polyphonic and noisy audio from this approach can be limited. But this hasn’t stopped folks like [Dmitry Ulyanov](https://dmitryulyanov.github.io/audio-texture-synthesis-and-style-transfer/) from using spectra for audio stylization, while [Leon Fedden](https://twitter.com/LeonFedden/status/996179566828933120), [Memo Akten](http://www.memo.tv/portfolio/grannma-magnet/) and [Max Frenzel](https://towardsdatascience.com/neuralfunk-combining-deep-learning-with-sound-design-91935759d628) have used spectra for generation. For phase reconstruction, Tacotron, Dmitry and Max use [Griffin-Lim](https://ieeexplore.ieee.org/document/1164317), while Leon and Memo use [LWS](https://github.com/Jonathan-LeRoux/lws) . Leon, Memo and Max all use an autoencoder to build a latent space across spectrograms.

Besides Dmitry, other researchers who have looked into style transfer include [Parag Mital](https://arxiv.org/abs/1711.11160) in November 2017 (focused on audio stylization) and [Mor et al](https://arxiv.org/abs/1805.07848" \t "_blank) in May 2018 (focused on musical style transfer across instruments/genres). For more early work on audio style transfer with only concatenative synthesis, “[Audio Analogies](http://www.iansimon.org/audio_analogies/)” (2005) provides a lot of inspiration.

In November 2017, DeepMind published their “[Parallel WaveNet](https://arxiv.org/abs/1711.10433)” technique where a slow-to-train WaveNet teaches a fast-to-generate student. Instead of predicting a 256-way 8-bit output, they use a [discretized mixture of logistics](https://arxiv.org/abs/1701.05517) (DMoL), which allows for 16-bit output. Google immediately started using Parallel WaveNet in production. In December 2017, Google published [Tacotron 2](https://ai.googleblog.com/2017/12/tacotron-2-generating-human-like-speech.html" \t "_blank) using a parallel WaveNet as the synthesis (vocoder) step instead of Griffin-Lim phase reconstruction. This kicked off a wave of papers focusing on speech synthesis conditioned on mel spectra, including [ClariNet](https://arxiv.org/abs/1807.07281" \t "_blank) (which also introduces an end-to-end text-to-wave architecture), [WaveGlow](https://nv-adlr.github.io/WaveGlow" \t "_blank) and [FloWaveNet](https://arxiv.org/abs/1811.02155" \t "_blank). In October 2018, Google published a [controllable version](https://google.github.io/tacotron/publications/gmvae_controllable_tts/index.html) of their Tacotron system, allowing them to synthesize voice in different styles (something they proposed in the original Tacotron blog post). There is a wealth of other research related to speech synthesis, but it isn’t always relevant to the more general task of generating audio in a musical context.

In February 2018, DeepMind published “[Efficient Neural Audio Synthesis](https://arxiv.org/abs/1802.08435)” or “WaveRNN” which solves fast generation using a handful of optimizations. Instead of using DMoL outputs, they achieve 16-bit output by using two separate 8-bit outputs: one for the high bits, and one for the low bits.

Where might this head next?

One domain that seems under explored is corpus-based synthesis (granular or concatenative) combined with frame-level representations. Concatenative synthesis is common in speech synthesis (where it’s called “unit selection”). These techniques also have a long history in sound design for texture synthesis with tools like [CataRT](http://imtr.ircam.fr/imtr/CataRT" \t "_blank). One significant limitation of this sort of corpus-based approach is that it’s impossible to generate a “moment” of audio that never appeared in your original corpus. If you trained a corpus-based model on all of Bach, and Bach never wrote a C minor major 7th chord, then you will never be able to generate a C minor major 7th. Even if the model learns how to produce each of the notes in the chord, and even if it even learns how to represent the corresponding frame, you won’t have source material to sample for synthesis. To overcome this constraint, perhaps there is something waiting to be discovered at the intersection of frame-by-frame granular modeling and research on [audio decomposition/factorization](https://arxiv.org/abs/1609.03296).

In terms of the research approach, I see at least two recurring questions. First, what kind of representations should we use? Should we treat sound as individual samples, as spectral frames with mostly monophonic tonal content, as a pitches in a grid, as properties of a vocal synthesizer? How much domain-specific knowledge should we embed into our representation of sound? And second, how do we want to interact with these systems? Do we want them to learn from the entire documented history of music with a vague goal of producing something similar, or something novel? To construct entire compositions, or to improvise with us? I’m wary of anyone who suggests that there is only one answer to these questions, and if anything we need to expand our imagination in terms of sound representation and modes of interaction.

I’ve noticed the more “accessible” algorithmic compositions are likely to trigger the question from journalists: “does this make human musicians obsolete?” Usually the researchers say they’re “not trying to replace humans”, but they’re trying to “build new tools”, or they encourage musicians to “think of the algorithms as collaborators”. Talking about creative AI as “augmenting” the human creative process feels reassuring. But is there any reason that an AI won’t eventually create a pop hit from scratch? Or not a pop hit, but just one of your favorite songs? I think the big question is less about whether human artists and musicians are obsolete in the face of AI, and more about what work we will accept as “art”, or as “music”. Maybe your favorite singer-songwriter can’t be replaced because you need to know there is a human behind those chords and lyrics for it to “work”. But when you’re dancing to a club hit you don’t need a human behind it, you just need to know that everyone else is dancing too.

There’s also an opportunity here to look beyond traditional models for what makes music “work”. [Vocaloids](https://en.wikipedia.org/wiki/Vocaloid" \t "_blank) like [Hatsune Miku](https://en.wikipedia.org/wiki/Hatsune_Miku) have shown that a virtual persona backed by a vocal synthesizer can bring together millions of people in a massively crowdsourced act of composition and listening. Music is probably [older than language](http://www.npr.org/templates/story/story.php?storyId=129155123), but we’re still discovering all the things music can be, and all the ways it might be crafted.