Learning to Surprise: A Composer-Audience Architecture

Razvan C. Bunescu and Oseremen O. Uduehi
School of Electrical Engineering and Computer Science
Ohio University
Athens, OH 45701
bunescu,ou380517@ohio.edu

Abstract

The ability to generate surprising outputs is essential for creative behavior. Surprise, or violation of expectation, has been hypothesized to be part of a fundamental mechanism enabling the capacity for emotion found in creative fields such as music, art, humor, or literature. Machine learning approaches to music generation train one model and sample from its distribution to generate new outputs. We show that this one-model sampling is fundamentally limited in its capacity for surprise. Drawing on insights from music and humor understanding, we propose a two-model architecture composed of an audience model for learning expectations connected to a composer model for learning to surprise. The new architecture facilitates a natural measure for surprise that is used in experimental evaluations on a set of synthetic tasks with binary strings. When instantiated with neural networks, the composer-audience model is shown to successfully learn to generate deterministic or random patterns of surprise, demonstrating its potential as a general framework for machine learning approaches to creative processes.

Introduction and Motivation

Creativity is widely considered to be an essential component of intelligent behavior (Boden 1991; Grace and Maher 2015). Surprise is a powerful driver for creativity and discovery, as such it has been used to guide search algorithms in models of computational creativity and discovery (Yannakakis and Liapis 2016). Owing to its importance for the creative process, surprise has become one of the core criteria for the evaluation of creative artifacts, together with novelty and value (Grace et al. 2015). As reviewed in (Itti and Baldi 2009), surprise is an essential concept in many studies on the neural basis of behavior, with surprising stimuli shown to be strong attractors of attention.

Surprise, or violation of expectation, has also been hypothesized to be an essential mechanism through which music and stories elicit emotion. According to (Meyer 1961), the principal emotional content of music arises from the composers manipulation of expectation. Composers build expectations in time, which then they purposely violate in order to elicit tension, prediction, reaction, and appraisal responses (Huron 2008). While significant progress has been made towards models that learn harmony, voice leading, and even long-term structure, e.g. (Boulanger-Lewandowski, Bengio, and Vincent 2012), (Hadjeres, Pachet, and Nielsen 2017), (Oore et al. 2018), (Huang et al. 2019), the importance of surprise for eliciting emotion is not reflected in the design of machine learning (ML) approaches to music generation, which use sampling from the trained model distribution to generate new musical output. In this paper, we show that one-model sampling is fundamentally limited in its capacity to generate surprising outputs and propose an architecture for learning to surprise comprised of an audience model that learns expectations and a composer model that learns patterns of violations of expectations. We instantiate the proposed architecture with LSTMs and evaluate it on three synthetic tasks with binary strings. Experimental results show that sampling from the two-model architecture enables much higher levels of surprise when compared with traditional one-model sampling. We conclude the paper by positioning our model in the context of previous work.

One-Model Sampling is Unsurprising

The vast majority of ML models used for generative tasks train **one model** P_M on **one dataset** D that is sufficiently large to enable a good approximation of the true data distribution. Given a trained model P_M, a sampling procedure is used to generate an output $\hat{\mathbf{x}} \sim P_M(\mathbf{x})$. When a language modeling approach is used for sequences of discrete events $\mathbf{x} = \langle x_1, x_2, ... \rangle$, such as tokens in text generation or chords in music composition, it is common to use a left-to-right sequential sampling based on the factorization below:

$$P_M(\mathbf{x}) = \prod_{k=1}^{|\mathbf{x}|} P_M(x_k|\mathbf{h}_k) = \prod_{k=1}^{|\mathbf{x}|} P_M(x_k|x_{k-1}, ..., x_1)$$

Such factorization could be provided for example by a unidirectional RNN. To generate an output sequence from left to right, at every step k a token $\hat{x}_k$ is sampled according the model, i.e. $\hat{x}_k \sim P_M(x_k|\mathbf{h}_k)$, where $\mathbf{h}_k = \langle \hat{x}_{k-1}, ..., \hat{x}_1 \rangle$ is the history (or context) of previously sampled tokens. If x_k itself is high-dimensional, as is the case with chords in music, models such as the restricted Boltzman machine (RBM) (Smolensky 1986; Hinton 2002) or the neural autoregressive distribution estimators (NADE) (Larochelle and Murray 2011) can be used to compute $P_M(x_k|\mathbf{h}_k)$ and generate approximate or exact samples, respectively. More complex

factorizations of the distribution, such as the bi-directional model of DeepBach (Hadjeres, Pachet, and Nielsen 2017) or general probabilistic graphical models require more sophisticated sampling procedures, e.g. MCMC methods, variational methods, or sampling via random projections. DeepBach, for example, uses a pseudo-Gibbs sampling procedure where at every iteration a note k is selected at random and then its pitch value $\hat{x}_k \sim P_M(x_k|\mathbf{h}_k)$ is re-sampled. In this case the history $\mathbf{h}_k$ contains all the other notes in the piece.

It is possible for this type of **one-model sampling** to generate surprising events. For example, if x_k is a binary variable and $P_M(x_k = 1|\mathbf{h}_k) = 0.9$, then on average 1 out of 10 times the sampling procedure will generate the "surprising" event $\hat{x}_k = 0$. The following informal observations can be made from this example:

Remark 1. Generation of *surprise* is possible only in event spaces with non-uniform probability distribution.

Remark 2. The *more surprising* an event needs to be, the less likely it is for it to be generated by one-model sampling.

Based on the later, one-model sampling would be ill-suited for tasks that require generating surprising events with high probability. Furthermore, one-model sampling generates surprising events in a completely random manner, which can be deeply unsatisfying if the task requires control over when to generate surprise or learning patterns of surprise.

We now present more formal versions of these statements, together with the corresponding proofs. While the two informal remarks seem obviously true, a formal specification has the advantage that it quantifies the notion of *surprise*, which will also help in clarifying the meaning of *more surprising* and its connection to sampling.

Definition 1. Let $\psi \in (0, 1]$ be an *expectation level*, with lower levels used to represent higher surprise. Let M be a model that computes the categorical distribution $P_M(x|\mathbf{h})$ over K categories. A discrete event x observed in a context $\mathbf{h}$ is called ψ-surprising for model M if $P_M(x|\mathbf{h}) < \psi/K$.

Definition 2. Let $S(\psi, \mathbf{h}, M)$ denote the expectation under M that a sampled event $\hat{x} \sim P_M(x|\mathbf{h})$ is ψ-surprising for M:

$$S(\psi, \mathbf{h}, M) = \mathrm{E}_{\hat{x}\sim P_M}\left[[\![P_M(\hat{x}|\mathbf{h}) < \psi/K]\!]\right]$$

where we use the Iverson bracket $[\![P]\!] = 1$ if the proposition P is satisfied, and 0 otherwise.

Using these definitions, the formal versions of the two remarks above are expressed as Theorems 1 and 2 below.

Theorem 1. *$S(\psi, \mathbf{h}, M) = 0$ for any uniform categorical distribution M, irrespective of the level $\psi \in (0, 1]$.*

Proof. If M is a uniform categorical distribution, $P_M(\hat{x}|\mathbf{h}) = 1/K$. Therefore $[\![P_M(\hat{x}|\mathbf{h}) < \psi/K]\!] = 0$ for any $\hat{x} \sim P_M(x|\mathbf{h})$, which means that the corresponding ψ-surprising expectation is $S(\psi, \mathbf{h}, M) = 0$. □

According to Theorem 1, a maximum entropy distribution offers no opportunity for surprise.

Theorem 2. *Let M be a non-uniform categorical distribution and $\rho_k = P_M(x = k|\mathbf{h})$ for each category $k \in 1..K$. Without loss of generality assume that the categories are indexed in the order of their sampling probabilities, i.e. $\rho_0 = 0 \le \rho_1 \le \rho_2 \le ... \le \rho_K \le 1$. Then:*

(a) $\dfrac{\psi}{K} \in (\rho_k, \rho_{k+1}]$ *for some* $k \in 0..K-1$.

(b) $S(\psi, \mathbf{h}, M) = \displaystyle\sum_{j=0}^{k} \rho_j \le \psi\frac{k}{K}$.

Proof. To prove (a) note that $\psi \in (0, 1] \wedge K \ge 2 \Rightarrow \psi/K \in (0, 1/K] \subset (0, 1]$. This means that ψ/K must belong to one of the sub-intervals from the following partition of $(0, 1]$:

$$(0, 1] = (0, \rho_1] \cup \bigcup_{k=1}^{K-1} (\rho_k, \rho_{k+1}] \cup (\rho_K, 1]$$

However, it is not possible for ψ/K to belong to the last sub-interval. Since M is non-uniform and ρ_K is the largest, this means $\rho_K > 1/K \ge \psi/K \Rightarrow \psi/K \notin (\rho_K, 1]$. Therefore, there can be only two scenarios:

1. $\psi/K \in (0, \rho_1]$, if $k = 0$ in (a).
2. $\psi/K \in (\rho_k, \rho_{k+1}]$, for some $k \in 1..K-1$ in (a).

We prove (b) separately for each of the two cases. In scenario 1, $\psi/K \le \rho_1$ implies $\psi/K \le \rho_k$ for all categories $k \in 1..K$. Therefore, $[\![P_M(\hat{x}|\mathbf{h}) < \psi/K]\!] = 0$ for any $\hat{x}$, which means that the expectation of a ψ-surprising event is:

$$S(\psi, \mathbf{h}, M) = 0 \tag{1}$$

Since $k = 0$ in this scenario, this means $S(\psi, \mathbf{h}, M) = \psi k/K$ which satisfies (b).

In scenario 2, because of how the categories were indexed, we have $\rho_1 \le \rho_2 \le ... \le \rho_k < \psi/K \le \rho_{k+1}$. Thus, for $P_M(\hat{x}|\mathbf{h}) < \psi/K$ to be true, $\hat{x}$ must satisfy $1 \le \hat{x} \le k$. Therefore, the ψ-surprising expectation is:

$$\begin{aligned} S(\psi, \mathbf{h}, M) &= \mathrm{E}_{\hat{x}\sim P_M}[[\![P_M(\hat{x}|\mathbf{h}) < \psi/K]\!]] \\ &= \sum_{j=1}^{k} P_M(\hat{x} = j|\mathbf{h}) \qquad (2) \\ &= \sum_{j=1}^{k} \rho_j \le \sum_{j=1}^{k} \rho_k = k\rho_k < \psi\frac{k}{K} \qquad (3) \end{aligned}$$

which satisfies (b). □

Corollary 2.1 below expresses the fact that generating very surprising events (small ψ/K) is impossible in the absence of very unlikely events (smaller $\rho_1 < \psi/K$).

Corollary 2.1. $S(\psi, \mathbf{h}, M) = 0$ *if* $\psi/K \le \rho_1$.

The dependence on the categorical distribution M can be removed from Theorem 2, as shown in Corollary 2.2 below.

Corollary 2.2. $S(\psi, \mathbf{h}, M) \le \psi\left(1 - \dfrac{1}{K}\right) < \psi$.

For Bernoulli distributions $K = 2$, for which the bound $S(\psi, \mathbf{h}, M) \le \psi/2$ in the theorem is tight.

Overall, Theorem 2 and its corollaries show that it is impossible for one-model sampling to generate very surprising events (i.e. very low level ψ) with high probability (i.e. large expectation $S(\psi, \mathbf{h}, M)$).

Why Controlled Surprise is Important

Using one-model sampling as the sole means of generating surprise has therefore two fundamental limitations:

1. Generation of truly surprising events, i.e. ψ-surprising with very small ψ, is very unlikely.
2. Surprising events are generated completely at random, with no mechanism available to (learn to) control surprise generation.

To better understand why the two limitations are important, consider the task of generating satirical news headlines. As observed by West and Horvitz (2019), changing a single word in a satirical news headline is often sufficient to make it sound like serious news, as in "*BP ready to resume oil* {*spilling, drilling*}". Furthermore, the changed word tends to reside towards the end of the headline. Using the notation introduced earlier, the context $\mathbf{h}$ can be seen as carefully building an expectation in the audience that is then turned upside-down by the word x_k appearing at position k towards the end of the headline. If M is the reader's model of expectation and V is the vocabulary, this can be expressed as $P_M(x_k|\mathbf{h}) < \psi/|V| \ll 1/|V|$, i.e. x_k is ψ-surprising for M with ψ very small. However, according to Theorem 2, a very small ψ makes it highly unlikely that sampling from a trained audience model would generate such a surprising event. Even when surprise is allowed at any position in the headline, the overall likelihood of sampling a surprising word is still very small because satirical headlines are usually very short. Thus, a writer generating headlines by sampling from the language model would have to discard a very large number of outputs before stumbling upon a satirical one. However, this is not how writers generate satirical headlines: while some randomness is probably still part of the process, there is also a significant mechanism at play that makes the generation of surprise substantially more likely than mere sampling from a language model shared with the audience.

The second limitation can manifest in multiple ways, for example as an inability to determine the required level of expectation violation or the frequency of surprising events. To illustrate, consider a headline generator primed with the word "*BP*" that generates phrases sequentially as shown below:

BP $\Rightarrow$ *BP wind farms*
$\Rightarrow$ *BP wind farms to provide*
$\Rightarrow$ *BP wind farms to provide grazing land*
$\Rightarrow$ *BP wind farms to provide grazing land to nearby ranchers for free*.

In the first step, it samples "wind farms", which happens to be just a bit surprising, as it is less expected to appear after "BP" than other phrases, such as "oil tankers". At the next step the high expectation verb "to provide" is sampled. Then the model samples "grazing land" which too happens to be just a bit more surprising in this context than other phrases, such as "electricity". Finally, the model samples relatively high expectation phrases, resulting in the complete headline "*BP wind farms to provide grazing land to nearby ranchers for free*". The level of surprise in this headline is much milder than in the satirical "*BP ready to resume oil spilling*", where "spilling" is much less expected than "drilling" given the previously generated words. While this type of control over the level of surprise is not available in one-model sampling, it can be achieved using a two-model architecture, as described in the next section.

A Two-Model Architecture for Surprise

To enable a data-driven control over the generation of surprise, we propose an architecture that contains two models: an **audience model** M^a and a **composer model** M^c that has access to expectations computed by M^a. These models will be trained on separate datasets, D^a and D^c, respectively. While the definition of ψ-surprising events remains the same as in the one-model sampling case, the definition of the ψ-surprising expectation is generalized to accommodate the two models, as shown in Definition 3 below.

Definition 3. Let $S(\psi, \mathbf{h}, M^a|M^c)$ denote the expectation that an event $\hat{x} \sim P^c_M(x|\mathbf{h})$ sampled from a composer model M^c is ψ-surprising for an audience model M^a:

$$S(\psi, \mathbf{h}, M^a|M^c) = \mathrm{E}_{\hat{x}\sim P^c_M}\left[[\![P^a_M(\hat{x}|\mathbf{h}) < \psi/K]\!]\right]$$

where the Iverson bracket $[\![P]\!] = 1$ if the proposition P is satisfied, and 0 otherwise.

The previous Definition 2 can be obtained from 3 by using the audience model also as a composer model, i.e. $M^c = M^a = M$. The adaptation of Theorem 1 for the two-model case still holds:

Theorem 3. $S(\psi, \mathbf{h}, M^a|M^c) = 0$ *for any uniform categorical distribution* M^a*, irrespective of* $\psi \in (0, 1]$.

The corresponding version of Theorem 2 (b) however does not hold anymore, as Equation 2 now changes to Equation 4:

$$S(\psi, \mathbf{h}, M^a|M^c) = \sum_{j=1}^{k} P^c_M(\hat{x} = j|\mathbf{h}) \tag{4}$$

While $S(\psi, \mathbf{h}, M^a|M^c)$ is still 0 for $\psi/K \leq \rho_1 = \min_k P^a_M(\hat{x} = k|\mathbf{h})$, it can now become arbitrarily large when $\psi/K > \rho_1$, depending on how much probability the composer allocates to categories that are unlikely in the audience distribution.

In order to learn to control violations of expectations (VoE), the composer model uses as input expectations computed by the audience model. We use the notation $M^c \leftarrow M^a$ to show this dependency. It is important that the two models are trained separately, on different datasets, in this sequence: M^a is first trained on data D^a, then it is plugged in the $M^c \leftarrow M^a$ architecture for training M^c on its dataset D^c while keeping M^a fixed. This training procedure is shown in Agorithm 1. During training of the M^c model in the $M^c \leftarrow M^a$ architecture, the same composer sequence x^c_k is provided as input to both M^a and M^c. To generate samples from the composer at test time, the previously trained M^a can be used or a new one can be trained on a different audience dataset D, as shown in Algorithm 2. When used in generation mode at test time, a token $\hat{x}^c_t$ is sampled according to the categorical distribution P^c_t and

Algorithm 1 TRAINCAMODEL(D^a, D^c)

Input: Datasets D^a and D^c.
Output: Composer model M^c.
1: train M^a on D^a
2: train $M^c \leftarrow M^a$ on D^c ▷ Keep M^a fixed.
3: **return** M^c

Algorithm 2 TESTCAMODEL(M^c, D)

Input: Composer model M^c.
Input: New audience examples D.
Output: Samples $\hat{x}^c$ from composer (and $\hat{x}^a$ from audience).
1: train M^a on D
2: sample $\hat{x}^c \sim M^c \leftarrow M^a$ (and $\hat{x}^a \sim M^a$)
3: **return** $\hat{x}^c$ (and $\hat{x}^a$)

fed as the next input token to both M^a and M^c. We use $\hat{x}^c \sim M^c \leftarrow M^a$ to denote the entire sequence sampled from the composer, whereas $\hat{x}^a \sim M^a$ is used to refer to a sequence sampled from the audience model.

The composer-audience (CA) architecture $M^c \leftarrow M^a$ can be instantiated using various ML models, depending on the type of data that needs to be processed. Figure 1 shows the architecture used in our experiments with synthetic data, which relies on LSTM units for processing sequential data. At the bottom, the audience LSTM processes the input sequence and computes at each time step $t-1$ a hidden state h_{t-1} and the categorical distribution P_t^a over the possible values for the next token. We call this the audience expectation for the next token. Together with the current token in the sequence, this expectation is used as input to the composer LSTM shown at the top of the figure. Optionally, the hidden state of the audience model could also be provided as input to the composer. The composer LSTM then computes its own categorical distribution P_t^c over the possible values for the next token.

Possible Scenarios for Text and Music

For the computational humor task described earlier, D^a could be a large collection of news headlines, perhaps augmented with text from news articles or open domain text. When trained on it, the M^a model would capture the expectation $P_M^a(x|\mathbf{h})$ of seeing word or phrase x in a textual context $\mathbf{h}$ in a normal, largely *non-humorous* text. The dataset D^c on the other hand would be composed only of satirical news headlines. Accordingly, training the composer model M^c on D^c using the CA architecture $M^c \leftarrow M^a$ would enable M^c to learn patterns of violations of expectations, such as generating a word that is ψ-surprising for M^a only when the audience expectation $P_M^a(x|\mathbf{h})$ for other words is very large. When used in the $M^c \leftarrow M^a$ architecture, the composer will also be able to learn the tendency for surprising words to be generated towards the end of the headline. In contrast, as shown earlier, training only one model on D^a will have very limited capacity for surprise and offer no control over when to violate expectations. Training one model on D^c is not going to work either, as it will not be able to distinguish between words that are expected by an audience and words that violate the audience's expectations.

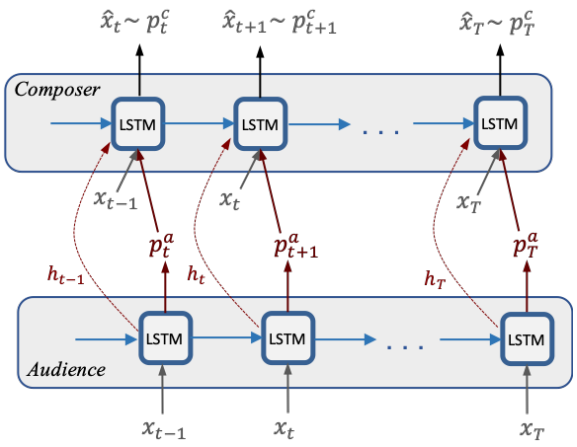

Figure 1: LSTM instantiation of $M^c \leftarrow M^a$ architecture.

The two-model architecture could be applied in a similar way to music generation. The common practice is to train one model on one corpus of music D and generate music by performing one-model sampling from it, a method that has limited capacity for surprise, as shown earlier. Instead, we propose training a composer-audience model, for which the music corpus needs to be partitioned into an audience corpus and a composer corpus, i.e. $D = D^a \cup D^c$. One solution is to choose a "present" time t and partition D around that time, i.e. store all music composed before t in the "old music" dataset D^a whereas all music composed after t is stored in the "new music" dataset D^c. After training the $M^c \leftarrow M^a$ architecture on this partition, the audience model would be re-trained on the entire dataset, plugged back in the $M^c \leftarrow M^a$ architecture, and new music would be generated by sampling from M^c in this architecture. The method can be refined in many ways, such as moving into D^a all music from D^c that is deemed too similar or derivative with respect to D^a. Furthermore, the fixed time cutoff for the partition can be avoided, as shown in the approach below aimed at addressing individual differences.

The Personalized Composer While it has the advantage of being simple, training just one $M^c \leftarrow M^a$ model does not consider individual differences in humor appreciation or music enjoyment. With respect to music, significant individual differences exist, from individuals who tend to experience a complex array of intense physiological and mental responses (Panksepp 1995) to individuals who report being unable to derive pleasure from listening to music (Mas-Herrero et al. 2014). While individual differences in aesthetic reward sensitivity were shown to have a neural basis (Sachs et al. 2016), differences in musical ability and familiarity were also observed to be important for experiencing intense emotional responses to music (Nusbaum and Silvia 2011). If we use the individual's performing or listening history $D = \{x_1, x_2, ..., x_T\}$ as a proxy for their musical ability and familiarity, then the CA architecture can be

used to train a composer model using a series of audience models. As shown in Algorithm 3 below, at each timestep t an audience model M_t^a is trained on all current music $D_t^a = \{x_1, ..., x_t\}$ and plugged in the $M^c \leftarrow M_t^a$ architecture to create together with x_{t+1} a training example for the composer model.

Algorithm 3 TRAINCASERIES(D)

Input: Chronological dataset $D = \{x_1, x_2, ..., x_T\}$.
Output: Composer model M^c.
1: **for** $t = 1$ to $T - 1$ **do**
2: train M_t^a on $D_t^a = \{x_1, ..., x_t\}$
3: train M^c on examples $\{M^c \leftarrow (M_t^a, x_{t+1})\}_{t=1..T-1}$
4: **return** M^c

Experimental Evaluation

The proposed CA architecture was evaluated on three synthetic tasks using binary strings: 1) violation of all high expectations, 2) violation followed by resolution of expectation, and 3) self-perpetuating random VoE. These synthetic tasks use clear patterns of expectations that enable us to determine the extent to which the models learn the expectations (audience) and their patterns of violation (composer). We use two evaluation measures for surprise throughout:

1. Expected maximum surprise:

$$S_{max}(M^a|M^c) = \mathrm{E}_{\hat{x}\sim P_M^c}\left[1 - \min_{1\le j\le|\hat{x}|} P_M^a(\hat{x}_j|\mathbf{h})\right]$$

2. Expected count of ψ-surprise:

$$S_{cnt}(\psi, M^a|M^c) = \mathrm{E}_{\hat{x}\sim P_M^c}\left[\left|\left\{\hat{x}_j \mid P_M^a(\hat{x}_j|\mathbf{h}) < \frac{\psi}{K}\right\}\right|\right]$$

S_{max} and S_{cnt} are calculated by averaging the sequence-level maximum surprise or count, respectively, over a set of generated sequences. Because all generated strings $\hat{x}$ in the experiment have the same fixed length N, the averaged count $S_{cnt}(\psi, M^a|M^c)/N$ can be seen as an estimate of the average ψ-surprising expectation from Definition 3.

The two models are trained using *teacher forcing*, i.e. the true token x_t is used as input for the next step. We use the cross entropy loss with respect to all the bits (random or pattern) in the training sequences. We emphasize that there is no explicit surprise-related loss and the only means for the composer to learn surprise is from the data. The extent to which the trained composer surprises the audience reflects the extent to which the patterns in D^c violate the expectations of a model trained on the patterns in D^a.

Violation of Expectation

In this scenario, the audience model learns when to generate high expectations, whereas the composer model learns to violate all expectations that are sufficiently high, where the expectation level required for VoE is learned from the data. Training and test examples are generated as quasi-random sequence of bits that are constrained to contain a given bit pattern. To compress the dataset description, let $x_j^m(b_1 b_2 ... b_k b_{k+1}, p, N)$ specify that example number j for the audience ($m = a$) or the composer ($m = c$) is a sequence of N bits that are generated at random with the following constraints:

1. Any time the sequence of bits $\langle b_1, b_2, ..., b_k\rangle$ is generated, it is immediately followed by the bit value b_{k+1}.
2. The complete pattern $\langle b_1, b_2, ..., b_k, b_{k+1}\rangle$ appears pN times in the entire string of N bits ($p < 1$).

For the rest of the paper we will be dropping the pattern frequency p and length N from the notation, as these will be global parameters that stay the same for all examples: $p = 0.1$ and $N = 200$. Table 1 shows training patterns used for the audience and composer in the experiments for this section. Below we show example training sequences generated for two patterns, one for the audience and one for the composer:

$$x_1^a(0011) = \langle 0, 1, \underline{0, 0, 1, 1}, 1, 0, 1, \underline{0, 0, 1, 1}, 1, 1, 0, ...\rangle$$
$$x_1^c(0010) = \langle 1, 1, 0, \underline{0, 0, 1, 0}, 1, 1, 0, 1, \underline{0, 0, 1, 0}, 1, ...\rangle$$

The composer patterns are designed to be the "opposite" of the audience patterns: any antecedent string 0, 0, 1 in the audience sequence above is followed by the consequent bit 1. For the composer sequence, the antecedent is the same but the consequent is flipped to 0. By training on this data, the composer is expected to learn that whenever the audience expectation is high (i.e. for a consequent bit) it should go against that expectation and generate the opposite bit.

We experimented with two training scenarios:

1. **Paired training**: In this scenario, shown in Algorithm 4, a separate audience model is trained for each pattern. When training the composer model on a composer pattern x_k^c, we plug in the audience model that was trained on the corresponding pattern x_k^a.
2. **Unpaired training**: This is the original training scenario shown in Algorithm 1 which is more realistic, as it does not require having knowledge of which patterns are used during training.

For paired training, we used the $K = 2$ training patterns shown in Table 1. For the more difficult case of unpaired

	Training patterns		Test
Audience	$x_1^a(0011)$	$x_2^a(1100)$	$x^a(0101)$
Composer	$x_1^c(0010)$	$x_2^c(1101)$	$\hat{x}^c(0100)$

Table 1: Audience & Composer examples for VoE.

Algorithm 4 PAIREDTRAINING(D^a, D^c)

Input: Audience dataset $D^a = \{x_1^a, ..., x_K^a\}$.
Input: Composer dataset $D^a = \{x_1^c, ..., x_K^c\}$.
Output: Composer model M^c.
1: **for** $k = 1$ to K **do**
2: train M_k^a on $\{x_k^a\}$
3: train M^c on examples $\{M^c \leftarrow (M_k^a, x_k^c)\}_{k=1..K}$
4: **return** M^c

training we added 3 more patterns to enable the composer model to better learn the importance of audience expectation: $x_3^a(1101)$, $x_4^a(1010)$, $x_1^a(0110)$ for the audience, and the corresponding opposite patterns for the composer. At test time, the audience model is trained on the new pattern $D = \{x^a\}$ and plugged in the $M^c \leftarrow M^a$ architecture, as shown in Algorithm 2. When used for sampling, the M^c model is expected to generate a string $\hat{x}^c$ that violates the expectation engendered by this new pattern. Note that the M^c model does not see the new pattern x^a during training.

The LSTMs were trained with Adam (Kingma and Ba 2015) for 10,000 epochs using a learning rate of 0.001. We generated 1,000 of training sequences of 200 bits each, for each pattern. The LSTM had two layers of neurons, with 2 neurons per layer for both the audience and composer model in the paired training mode. For unpaired training, where just one model had to learn all patterns, these were increased to 5 neurons for the audience and 4 for the composer. Overall, it was important to keep the capacity small so that the audience does not memorize the input sequences, while ensuring that the composer does not overfit to the bit pattern. We report results for two input scenarios for the composer:

1. Using only the audience expectation as input (first result in the table cells.
2. Using both the bit at the current position and the audience expectation (results in parentheses in the table cells).

Sampling model M	$S_{max}(M^a\|M)$	$S_{cnt}(\psi, M^a\|M)$
Audience: $M = M^a$	0.61 (0.64)	0.07 (0.13)
Composer: $M = M^c$	0.99 (0.99)	11.03 (12.14)

Table 2: Expected ψ-surprise for Paired training: $\psi = 0.1$, S_{cnt} is per 100 bits, composer accuracy 99% (99%).

Sampling model M	$S_{max}(M^a\|M)$	$S_{cnt}(\psi, M^a\|M)$
Audience: $M = M^a$	0.55 (0.54)	0.02 (0.05)
Composer: $M = M^c$	0.99 (0.99)	11.17 (11.23)

Table 3: Expected ψ-surprise for Unpaired training: $\psi = 0.1$. S_{cnt} is per 100 bits, composer accuracy 99% (99%).

Tables 2 and 3 report the surprise that the audience model M^a experiences on samples $\hat{x}^c$ from composer (two-model sampling), as well as on samples $\hat{x}^a$ from the audience itself (one-model sampling). For each model, the surprise numbers are averaged over 100 sampled sequences of 200 bits each. The results show that the audience model is much more surprised by examples sampled from the composer model, both in terms of maximum surprise S_{max} and average count of ψ-surprising events S_{cnt}. The sequences generated by the composer satisfy the opposite pattern $\hat{x}^c$ shown on dark background in Table 1 with an accuracy of 98% or higher (accuracy numbers shown in the caption).

Delayed Resolution of Expectation

Delayed resolution of dissonance is one major tool composers use to play with the audience's sense of expectation.

	Training patterns		Test
Audience	$x_1^a(0011)$	$x_2^a(1100)$	$x^a(0101)$
Composer	$x_1^c(00101)$	$x_2^c(11010)$	$\hat{x}^c(01001)$

Table 4: Audience & Composer examples for Delayed VoE.

To emulate this, we experimented with the dataset shown in Table 4 where the composer had to first violate the expectation (flip the consequent bit) and then satisfy it (follow with the expected bit). For example, by training on pattern $x_1^a(0011)$, the audience learns to compute a high expectation for the bit 1 whenever it follows the antecedent sequence 0, 0, 1. Thus, whenever the composer sees the antecedent 0, 0, 1, it also sees that the audience model has a high expectation for 1 to follow. By being trained on $x_1^c(00101)$, the composer learns that it should generate a 0 when the audience expectation is high, effectively violating the expectation, and then follow with the bit 1 expected by the audience.

Tables 5 and 6 show the results in the paired and unpaired training scenarios, respectively, using the same methodology as in the previous section. Here too the results show that the audience is much more surprised by examples sampled from the composer model, in terms of both maximum surprise and average frequency of ψ-surprising events.

Sampling model M	$S_{max}(M^a\|M)$	$S_{cnt}(\psi, M^a\|M)$
Audience: $M = M^a$	0.68 (0.67)	0.13 (0.14)
Composer: $M = M^c$	0.99 (0.99)	12.90 (11.71)

Table 5: Delayed ψ-surprise for Paired training, $\psi = 0.1$. S_{cnt} is per 100 bits, composer accuracy 99% (99%).

Sampling model M	$S_{max}(M^a\|M)$	$S_{cnt}(\psi, M^a\|M)$
Audience: $M = M^a$	0.54 (0.55)	0.00 (0.00)
Composer: $M = M^c$	0.99 (0.99)	11.98 (11.83)

Table 6: Delayed ψ-surprise for Unpaired training, $\psi = 0.1$. S_{cnt} is per 100 bits, composer accuracy 99% (98%).

Longer Patterns For all surprise scenarios from this section (delayed VoE) and the previous section (direct VoE), we also evaluated the composer model on longer patterns at test time, i.e. using an audience model trained on $x^a(01101)$, $x^a(111010)$, and $x^a(0101101)$. Even though at training time the composer had seen expectation patterns with only 4 bits, its performance on longer patterns was overall very similar with the performance reported in Tables 2 to 6, in terms of both surprise measures and accuracy. This can be seen as further evidence of the ability of the CA architecture with LSTMs to learn general VoE patterns.

Never-Ending Surprise

In this scenario, surprise is generated by violating high expectation at random. We first create a training dataset $D^a = \{\mathbf{o}_0\}$ for the audience that contains only a sequence of N random bits $\mathbf{o}_0$. Let $D^c = \{\mathbf{o}_1, \mathbf{o}_2, ...\}$ be a training dataset

for the composer containing two or more sequences that are similar to $\mathbf{o}_0$ but not exactly the same. Each sequence in D^c is generated by starting from $\mathbf{o}_0$ and randomly flipping bits, where at each position in the sequence the probability of flipping the bit is given by a Bernoulli distribution with mean $p = 1/\mathrm{N}$. If $\mathbf{Bernoulli}(p, \mathrm{N})$ is a sequence of N draws from this distribution, then each composer sequence $\mathbf{o}_j$ can be seen as the element-wise exclusive-or between $\mathbf{o}_0$ and this random vector, i.e. $\mathbf{o}_j = \mathbf{o}_0 \oplus \mathbf{r}_j = \mathbf{o}_0 \oplus \mathbf{Bernoulli}(p, \mathrm{N})$. For example, if N = 10, the two datasets could be as follows:

$$\begin{aligned} D^a &= \{\mathbf{o}_0 = \langle 0,1,0,0,1,1,0,1,0,0\rangle\} \\ &\quad\ \mathbf{r}_1 = \langle 0,0,1,0,0,0,0,0,0,0\rangle \\ &\quad\ \mathbf{r}_2 = \langle 0,0,0,0,0,1,0,0,0,0\rangle \\ &\quad\ \mathbf{r}_3 = \langle 0,0,0,1,0,0,0,1,0,0\rangle \\ D^c &= \{\mathbf{o}_1 = \mathbf{o}_0 \oplus \mathbf{r}_1 = \langle 0,1,1,0,1,1,0,1,0,0\rangle, \\ &\quad\ \mathbf{o}_2 = \mathbf{o}_0 \oplus \mathbf{r}_2 = \langle 0,1,0,0,1,0,0,1,0,0\rangle, \\ &\quad\ \mathbf{o}_3 = \mathbf{o}_0 \oplus \mathbf{r}_3 = \langle 0,1,0,1,1,1,0,0,0,0\rangle\} \end{aligned}$$

To train the composer model, first an audience model M^a is trained on D^a. Then the composer model $M^c \leftarrow (M^a, \mathbf{r})$ is trained on D^c, using as input the expectations computed by M^a for each example $\mathbf{o}_j$, as well as the random sample vectors $\mathbf{r}_j$. Upon training in this setting, the expectation is that the composer model will learn to violate the high expectations produced by the audience model only at the times specified by the random control vector $\mathbf{r}$. Thus, if the audience is trained on an arbitrary sequence $\mathbf{o}_0$ then, given a random control pattern $\mathbf{r}$, the composer should learn to compute mostly the element-wise exclusive-or between the two, i.e. $\mathbf{o} \approx \mathbf{o}_0 \oplus \mathbf{r}$ (deviations may happen due to the sampling done at each step in the sequence).

Algorithm 5 NEVERENDING($\mathbf{o}_0$, N, p)

Input: An initial sequence of bits $\mathbf{o}_0$.
Output: An infinite stream of bit sequences $\mathbf{o}_1, \mathbf{o}_2, ...$

1: let $k = 0$, $D_0^c = \{\mathbf{o}_0\}$
2: **for ever do**
3: set $k = k + 1$
4: train M_k^a on D_{k-1}^c
5: let $\mathbf{r}_k \sim \mathbf{Bernoulli}(p, \mathrm{N})$
6: let $D_k^c = \{\mathbf{o}_k \sim M^c \leftarrow (M_k^a, \mathbf{r}_k)\}$
7: **yield** $\mathbf{o}_k$

Once the composer model is trained, it can be used to generate an infinite stream of surprising sequences using Algorithm 5. The algorithm starts by initializing the set of compositions D_0^c with an input sequence $\mathbf{o}_0$. This can be an arbitrary sequence of N bits, for example all zeroes. At every iteration of the never-ending generation loop, the current audience model M_k^a is trained on the previous set of compositions (step 4). Then a random control pattern $\mathbf{r}_k$ is generated (step 5) and together with the current audience model M_k^a are used as input to the composer model M^c, which generates a new surprising sequence $\mathbf{o}_k$ (step 6). An example output is illustrated below:

$$\begin{aligned} D_0^c &= \{\mathbf{o}_0 = \langle 0,0,0,0,0,0,0,0,0,0\rangle\} \\ &\quad\ \mathbf{r}_1 = \langle 0,0,0,1,0,0,0,0,0,0\rangle \\ D_1^c &= \{\mathbf{o}_1 \approx \mathbf{o}_0 \oplus \mathbf{r}_1 \approx \langle 0,0,0,1,0,0,0,0,0,0\rangle\} \\ &\quad\ \mathbf{r}_2 = \langle 0,0,1,0,0,0,0,0,1,0\rangle \\ D_2^c &= \{\mathbf{o}_2 \approx \mathbf{o}_1 \oplus \mathbf{r}_2 \approx \langle 0,0,1,1,0,0,0,0,1,0\rangle\} \\ &\quad\ \mathbf{r}_3 = \langle 0,0,0,1,0,0,0,1,0,0\rangle \\ D_3^c &= \{\mathbf{o}_3 \approx \mathbf{o}_2 \oplus \mathbf{r}_3 \approx \langle 0,0,1,0,0,0,0,1,1,0\rangle\} \\ &\quad\ \mathbf{r}_4 = \langle 0,0,0,0,0,0,0,1,0,0\rangle \\ D_4^c &= \{\mathbf{o}_4 \approx \mathbf{o}_3 \oplus \mathbf{r}_4 \approx \langle 0,0,1,0,0,0,0,0,1,0\rangle\}... \end{aligned}$$

The composer LSTM was reduced to one layer with 2 neurons, and both models were trained for 5,000 epochs. The results in Table 7 show that the audience model experiences no surprise on samples from itself, whereas samples from the composer are very effective at eliciting surprise.

Sampling model M	$S_{max}(M^a\|M)$	$S_{cnt}(\psi, M^a\|M)$
Audience: $M = M^a$	0.00	0.00
Composer: $M = M^c$	0.99	1.65

Table 7: Expected Never-Ending ψ-surprise: $\psi = 0.1$, S_{cnt} is per 10 bits, composer accuracy 100%.

Relation to Previous Work

Itti and Baldi (2006; 2009) define the surprise of an agent M upon observing data D as the KL-divergence $KL(P(M|D)\|P(M))$ between the posterior distribution of beliefs after the agent observes the data and its prior distribution of beliefs. This Bayesian definition of surprise is shown to be a good predictor of events that attract human attention in video frames. Macedo and Cardoso (2001) define the surprise of an event E as $1 - P_M(E)$. Given that a low probability alone cannot fully account for surprise (Teigen and Keren 2003), such as in models with uniform distributions, Macedo, Reisezein, and Cardoso (2004) refined the definition of surprise to also consider the most likely event E_h, i.e. $\log(1 + P(E_h) - P(E))$. Note that our definition of surprise naturally solves the uniform distribution dilemma by using a threshold that depends on the number of categories. Similar to (Macedo and Cardoso 2001), Horvitz et al. (2005) define surprising events to be those with low likelihood, e.g. 0.1 or less. They also go one step further and train a Bayesian network to forecast surprising events 30 minutes in advance for their traffic flow model JamBayes. In the context of evaluating creative designs, Maher, Brady, and Fisher (2013) identify surprising designs as outliers with respect to predictions based on features from previous designs.

Overall, these approaches were aimed at recognizing or forecasting surprise. To the best of our knowledge, the two-model architecture described in this paper is the first to address the task of producing surprising outputs by learning patterns of surprise from data. In terms of models that learn to generate surprising data, the most relevant work is Schmidhuber's Formal Theory of Creativity, summarized in (Schmidhuber 2012). There, the learning agent is entirely unsupervised and is expected to create novel and surprising data on its own, using a reinforcement learning algorithm

that rewards the agent when it generates data that helps it better compress its history of interactions with the environment. In contrast, our learning approach is data-driven in the sense that it is trained to minimize loss on data that is given to it, e.g. bit sequences. At the same time, like Schmidhuber's creative agent, it does not require explicit supervision in terms of surprise, i.e. the composer is never told whether a particular event is surprising or not. The composer learns to surprise the audience only to the extent that the data provided to it is surprising for its model of the audience, which itself learns patterns of expectation from its own data.

There are also other two-model architectures, albeit designed for different purposes, such as the discriminator-generator model of generative adversarial nets (Goodfellow et al. 2014) or the student-teacher model used in the music theory learning system of (Yu and Varshney 2017).

Acknowledgments

We would like to thank Kristen Masada and Gordon Stewart for useful discussions on an earlier draft and the anonymous reviewers for their constructive feedback.

References

Boden, M. A. 1991. *The Creative Mind: Myths and Mechanisms*. New York, NY, USA: Basic Books, Inc.

Boulanger-Lewandowski, N.; Bengio, Y.; and Vincent, P. 2012. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proceedings of ICML'12*, 1–8.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *NIPS*, 2672–2680.

Grace, K., and Maher, M. L. 2015. Specific curiosity as a cause and consequence of transformational creativity. In *Proceedings of ICCC*, 260–267.

Grace, K.; Maher, M. L.; Fisher, D.; and Brady, K. 2015. Modeling expectation for evaluating surprise in design creativity. In *Design Computing and Cognition*, 189–206.

Hadjeres, G.; Pachet, F.; and Nielsen, F. 2017. DeepBach: a steerable model for Bach chorales generation. In *Proceedings of ICML'17*, 1362–1371.

Hinton, G. E. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation* 14(8):1771–1800.

Horvitz, E.; Apacible, J.; Sarin, R.; and Liao, L. 2005. Prediction, expectation, and surprise: Methods, designs, and study of a deployed traffic forecasting service. In *Proceedings of UAI'05*, 275–283.

Huang, C.-Z. A.; Vaswani, A.; Uszkoreit, J.; Simon, I.; Hawthorne, C.; Shazeer, N.; Dai, A. M.; Hoffman, M. D.; Dinculescu, M.; and Eck, D. 2019. Music transformer. In *International Conference on Learning Representations*.

Huron, D. 2008. *Sweet Anticipation: Music and the Psychology of Expectation*. MIT.

Itti, L., and Baldi, P. F. 2006. Bayesian surprise attracts human attention. In *NIPS*. MIT Press.

Itti, L., and Baldi, P. 2009. Bayesian surprise attracts human attention. *Vision Research* 49(10):1295 – 1306.

Kingma, D. P., and Ba, J. L. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations* 1–15.

Larochelle, H., and Murray, I. 2011. The neural autoregressive distribution estimator. In *AISTATS*, 29–37.

Macedo, L., and Cardoso, A. 2001. Modeling forms of surprise in an artificial agent. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 23.

Macedo, L.; Reisezein, R.; and Cardoso, A. 2004. Modeling forms of surprise in artificial agents: empirical and theoretical study of surprise functions. In *Proceedings of the 26th Annual Meeting of the Cognitive Science Society*.

Maher, M. L.; Brady, K.; and Fisher, D. H. 2013. Computational models of surprise in evaluating creative design. In *Proceedings of ICCC*, volume 147.

Mas-Herrero, E.; Zatorre, R. J.; Rodriguez-Fornells, A.; and Marco-Pallarés, J. 2014. Dissociation between musical and monetary reward responses in specific musical anhedonia. *Current Biology* 24(6):699–704.

Meyer, L. 1961. *Emotion and Meaning in Music*. University of Chicago.

Nusbaum, E. C., and Silvia, P. J. 2011. Shivers and timbres: Personality and the experience of chills from music. *Social Psychological and Personality Science* 2(2):199–204.

Oore, S.; Simon, I.; Dieleman, S.; Eck, D.; and Simonyan, K. 2018. This time with feeling: learning expressive musical performance. *Neural Computing and Applications*.

Panksepp, J. 1995. The emotional sources of "chills" induced by music. *Music Perception: An Interdisciplinary Journal* 13(2):171–207.

Sachs, M. E.; Ellis, R. J.; Schlaug, G.; and Loui, P. 2016. Brain connectivity reflects human aesthetic responses to music. *Social Cognitive and Affective Neuroscience* 11(6):884–891.

Schmidhuber, J. 2012. A formal theory of creativity to model the creation of art. In McCormack, J., and dInverno, M., eds., *Computers and Creativity*. Springer-Verlag.

Smolensky, P. 1986. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. MIT Press. chapter Information Processing in Dynamical Systems: Foundations of Harmony Theory, 194–281.

Teigen, K. H., and Keren, G. 2003. Surprises: Low probabilities or high contrasts? *Cognition* 87(2):55–71.

West, R., and Horvitz, E. 2019. Reverse-engineering satire, or "Paper on computational humor accepted despite making serious advances". In *Proceedings of AAAI*, 1–8.

Yannakakis, G. N., and Liapis, A. 2016. Searching for surprise. In *Proceedings of the Seventh International Conference on Computational Creativity*, 25–32.

Yu, H., and Varshney, L. R. 2017. Towards deep interpretability (MUS-ROVER II): Learning hierarchical representations of tonal music. In *ICLR*.