A Process-centric Analysis of Modern Generative AI As Computational Creativity

Dan Ventura
Department of Computer Science
Brigham Young University
ventura@cs.byu.edu

Abstract

We present a careful, general picture of modern, foundation-model-based, generative AI systems, discuss how they work, and analyze them in the context of historical work on the foundations of computational creativity. In particular, we revisit four approaches to *process* that characterize what is meant by computational creativity and consider to what extent, by these standards, modern generative systems should be judged computationally creative. We conclude by discussing experiments and research directions suggested by this analysis.

Introduction

The genesis of the field of computational creativity (CC) can be traced back to seminal work by Margaret Boden (Boden 1992; Boden 1998),[1] in which she posited that artificial intelligence (AI) might be developed to account for general creative ability. The ideas she put forward were quite forward-thinking and exciting, while at the same time being, in some ways, still quite informal and underdeveloped, at least from the standpoint of implementation. In particular, her work did not make it clear *how* such systems should work. In the following decade, many attempts were made using classical AI approaches (Saunders and Gero 2001; Machado and Cardoso 2002; Martins et al. 2004; Cope 2005; O'Donoghue, Bohan, and Keane 2006; Veale 2006; Riedl and Young 2006; Hull and Colton 2007; Strapparava, Valitutti, and Stock 2007; Ritchie et al. 2007; Gervás 2009). Because the systems being developed were ad hoc and because they addressed a wide variety of disparate domains, a domain-agnostic method for analyzing such systems was necessary.[2] An early proposal by Ritchie focused on what was created—the *product* of the system. He suggested an empirical methodology for sampling that product and a set of example metrics for measuring different characteristics of the results (2007). The approach is direct and concrete and easily implementable; however, it requires potentially significant runtimes for accumulating data, it offers only relative comparison between systems, and it does not consider the *process* used to produce artefacts.

Focusing on the idea of process, Wiggins revisited the ideas of Boden and proposed a formalization based on a general, abstract notion of search (2006). Because the result was still too abstract to admit a direct operationalization of the theory, Ventura later attempted to make the ideas more modular and concrete and offered examples of extant systems that did implement at least some modules (2017). In the meantime, Colton, Charnley, and Pease considered the problem of analyzing process in another way entirely, offering a descriptive framework that accounted for four process dimensions and two levels of abstraction (2011). Ventura suggested yet another approach to analysis of process by proposing a spectrum of abstract but well-defined creativity milestones against which CC systems can be judged (2016). All four of these approaches provide researchers the ability to talk about systems operating in different domains using a common language and offer the potential for critical analysis of systems at the process level.

Over the next decade, CC systems became more sophisticated, incorporating neural, hierarchical and hybrid approaches and tackling more complex and ill-defined domains (Smith and Mateas 2011; Cook, Colton, and Gow 2016; Liapis, Yannakakis, and Togelius 2015; Gervás 2011; Toivanen et al. 2012; Carlson et al. 2016; Li et al. 2012; Besold and Plaza 2015; Cunha et al. 2017; Morris et al. 2012; Xiao and Blat 2013; Llano et al. 2014; Confalonieri, Plaza, and Schorlemmer 2016; Colton 2012; Elgammal et al. 2017; Oliveira, Costa, and Pinto 2016).

Now, yet another decade has passed, and massive and unpredictable advances have swept the field of artificial intelligence and therefore also the subfield of computational creativity. As a result, many of the most recently proposed CC systems are built using foundation models, including for domains like irony (Veale 2025), humor (Inácio and Oliveira 2025), character generation (Tian 2024), and recipes (Taneja, Segal, and Goodwin 2023). However, in-

[1] While most CC researchers are likely to agree with this, depending on one's viewpoint, the beginnings of the field might be argued to pre-date Boden; indeed, she cites work that might be considered (proto-) computational creativity to support her arguments; in fact, some authors cite things at least as early as Mozart's dice game as examples of (proto-) CC; however, for our purposes, starting with Boden seems appropriate.

[2] While a domain-agnostic method of *analysis* is appealing, Jordanous makes an interesting argument for an approach to *evaluation* that is instead domain-specific (and possibly even system-specific) (2012).

terestingly, many recent papers have begun, at least implicitly, to assume that those same foundation models are themselves already CC systems and to focus on evaluating their creative capabilities (Góes et al. 2023; Ventura 2023; Peeperkorn et al. 2024; Nath, Dayan, and Stevenson 2024; Rabeyah et al. 2025; Ismayilzada, Stevenson, and van der Plas 2025; Morain and Ventura 2025). However, none of these recent attempts at evaluation have leveraged the foundational work on process discussed above. As a result, we see an opportunity to revisit those foundations and to ask how the process of modern AI foundation models measures up. This paper does just that: first, presenting a general abstraction of the modern generative AI system; then briefly discussing four approaches to talking about process in CC systems; and then considering how modern generative AI systems do or do not account for those theories.[3,4]

Modern Foundation Models

We broadly categorize modern generative AI systems as conditional models that transform an input context into a target artifact via one of two principal mechanisms: autoregressive prediction or iterative denoising. The conditioning context is typically encoded as embeddings, discrete tokens, or latent vectors; the generation may occur in discrete token space or in a continuous latent space; and the resulting intermediate representation is then decoded or rendered into the target modality. An abstract system architecture of such a system is shown in Figure 1. It is very general, and we assert that it adequately represents the majority of current state-of-the-art generative models. It employs a collection of models and processes that interact with each other, with data, and with the user. High-level interactions are shown in the diagram for both generation (blue) and training (red).

Generation

The generative process consists of seven main components, numbered 1-7 in blue in Figure 1:

1. token prediction
2. autoregression
3. user prompting
4. prompt embedding
5. conditioning
6. decoding latent tokens into an artefact
7. denoising/diffusion

[3]We recognize that there are other aspects of creativity beyond *product* and *process*—in particular those of *producer* and *press* discussed by (Jordanous 2016)—and it would be interesting to consider modern generative AI from those perspectives; however, that is beyond the scope of this work.

[4]We will limit our analysis to considering modern generative AI as an autonomous CC system, rather than as a co-creator, for several reasons: the scope of the paper does not allow treating the co-creative case; the frameworks employed here were all originally conceived for addressing the autonomous case; the additional complexity of incorporating human co-creators into the analysis likely requires a different set of tools.

Token Prediction The modern generative process is usually treated as a problem of token prediction. An artefact to be generated is considered a collection of tokens $\{a_1, ..., a_m\}$ related structurally in some domain-appropriate way G, where G can be thought of as a graph that defines which tokens are "neighbors". This is a very general and powerful representational approach. For example:

- for the domain of music, a token might be a symbolic musical note and the structure a multi-sequence (for multiple voices); or, the token might be a waveform sample and the structure a simple sequence; or, the token might be a mel spectrum representation of a complete song and the structure trivial.
- for the domain of images, a token might be a pixel or an image patch and the structure a two-dimensional grid; or the token might be the complete image and the structure trivial.
- for the domain of language, a token might be a phoneme, a letter, a partial word, a word or even a word sequence and the structure a sequence.
- for the domain of video, a token might be a frame of video and the structure a sequence; or, the token might be an image patch and the structure a three-dimensional grid.

The core of most state-of-the-art generative systems is the transformer (Vaswani et al. 2017). A transformer is a deep neural network model that generates tokens $a_1, ..., a_m$ by computing a conditional probability distribution p over a vocabulary $V = \{v_0, ..., v_n\}$ and sampling that distribution. p typically depends on how the tokens that compose an artefact are related structurally and may also depend on other information, such as a conditioning signal σ. These dependencies are modeled using an attention mechanism: self-attention for previously generated tokens and cross-attention for conditioning signals. Tokens are represented internally in the network as real-valued vector embeddings, with each layer computing new embeddings for each token.

It is this iterative layer-by-layer transformation of embeddings that gives the transformer its name. Attention (Vaswani et al. 2017) is used during the generation of tokens to allow previously generated tokens (self-attention) or tokens from another sequence (cross-attention) to influence the generation of the current token by influencing the token embeddings at each layer of the transformer.

Autoregression Each generated token is conditioned on other previously generated tokens that are structurally related to it (e.g., in language generation, a word depends on words earlier in the sequence). This is accomplished via a self-attention function α, which computes a matrix of attention scores at each transformer layer k, $A_k = \alpha(\tau_k(b_i), \tau_k(b_j)), 1 \leq i, j \leq m$, where b_i are embeddings (from layer $k-1$) of all tokens. In general, this matrix will likely have parts of it masked (e.g., in language modeling, tokens later in the sequence are masked, and A is lower triangular). The resulting attention information represented in the matrices A_k is used by the transformer in computing the new embeddings $\tau_k(\mathbf{b}_{k-1}, A_k)$ for layer k, where $\mathbf{b}_{k-1}$ is a

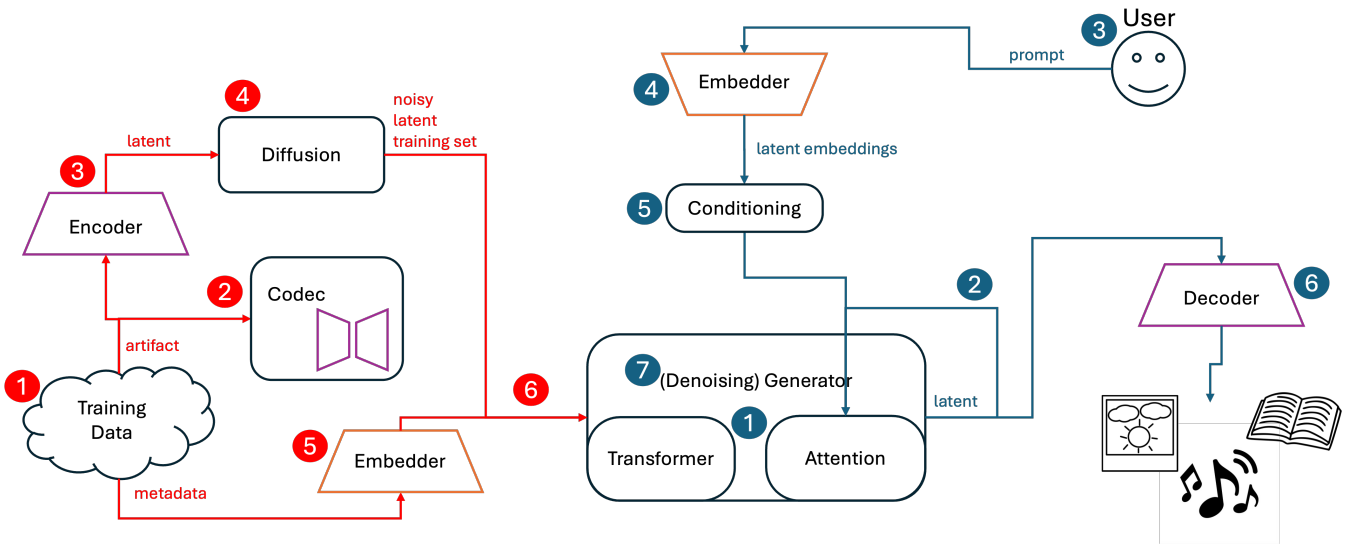

Figure 1: Diagram of an abstract modern generative AI system. High-level interactions are shown in two different modes: generation (blue) and training (red).

vector of all embeddings $b_1, ..., b_m$ from the previous layer $k-1$.

User Prompting It is usually desirable to provide the generative process additional information σ, typically referred to as a prompt. During generation, this additional information can "steer" the outcome towards some desired characteristics. For example, prompting information for a music generation system might include things like key, genre, lyrics, inspiring artists, mood, instrumentation, style, audio of other music, etc). Typically, a user provides the prompt; however, in many systems the initial user-provided prompt can be rewritten by the system for various reasons including efficiency, legality, bias avoidance, etc.; and, in extreme cases, the user may be another system rather than a human user.

Prompt Embedding Because this information σ may be of many different modalities (text, image, audio, video, etc.), it must be transformed into a signal that is semantically meaningful for the model to be conditioned. This is done using an embedding model β. The conditioning information is tokenized as a collection $\sigma = c_1, ..., c_l$, and each token is then converted to a real-valued vector embedding $\beta(c_i)$. If the prompt information is multi-modal, then β is typically implemented by a set $\{\beta_i\}$ of embedding models, one for each modality. These models might have been been co-developed with a common embedding space with a shared semantics, such as CLIP (Radford et al. 2021) for image and text or CLAP (Wu et al. 2023) for text and audio, or they might be off-the-shelf models developed independently, such as RoBERTa (Liu et al. 2019) for text or MERT (Li et al. 2024) for music. Finally, the multiple modalities must be combined into a coherent signal $\mathbf{d} = g(\beta_1(\sigma), ..., \beta_n(\sigma))$ that can be injected into the generative transformer.

Conditioning Conditioning allows the generative model to also be influenced by non-autoregressive information using the same mechanism used for autoregression—attention. This is accomplished via a cross-attention function χ, which computes a matrix of attention scores at each transformer layer k, $C_k = \chi(\tau_k(b_i), \beta(d_j)), 1 \le i \le m, 1 \le j \le l$. Because conditioning is based on information from outside the model, masking is typically unnecessary and the full conditioning information is usually provided to all tokens during both training and generation. The resulting attention information represented in the matrices C_k is used, along with that in the self-attention matrices A_k, by the transformer in computing the new embeddings $\tau_k(\mathbf{b}_{k-1}, A_k, \mathbf{d}, C_k)$.

Decoding Latent Tokens into an Artefact Most state-of-the-art models no longer work directly on tokens; instead, they operate in a latent space, consuming and generating latent representations, because this has been found both to be more efficient and to produce better quality results. A latent representation is similar to an embedding in that it is an abstract representation of data as a real-valued vector. The difference is in the motivation: embeddings are used to convert data from its original form into a model-consumable form in which key features are numerically encoded; latent representations are used to represent data in a compressed form that still contains all the salient features of the original data. Latent representations are useful because they are smaller than the original and thus allow more efficient computation and because they can easily be converted back into the original format. They can also help with generalization. To access this latent space, a pair of models (ϵ, δ) commonly referred to as a *codec*[5] is required. ϵ is an encoder model

[5]One common approach to building a codec is the variational autoencoder (VAE) (Kingma and Welling 2019).

that takes a token a as input and outputs a latent representation $\epsilon(a)$ for that token such that $|\epsilon(a)| < |a|$, and often $|\epsilon(a)| \ll |a|$. Conversely, δ is a decoder model that reverses the process, $\delta(\epsilon(a)) = a$. Rather than generating tokens a_i, the transformer actually generates latents l_i.[6] When all latents $l_1, ..., l_m$ have been generated, the decoder is used to decode them into tokens, $a_i = \delta(l_i)$ and the resulting tokens $a_1, ..., a_m$ compose the final artefact according to the structure G.

Denoising/Diffusion Some systems incorporate denoising/diffusion in the generative process. Diffusion models are based on noise models of physical processes and are composed of two parts: a forward diffusion process (discussed below in the subsection on training) and a backward de-noising process. The denoising process uses a noise prediction model ρ to remove noise from a latent and assumes that noisy latents are noisy due to a well-understood noise diffusion process. Given that assumption, the generative process can simply reverse that diffusion process by iteratively removing noise using a noise schedule, which determines how many denoising steps are used and the relative size of each step (linear, cosine, etc.) Using ρ the denoising generator generates a latent by taking as input a random vector z and iteratively denoises it to produce a clean latent representation $l_i = \rho(z)$, with both the context of previously generated latents and the conditioning signal acting to guide the process towards a result that has the desired characteristics. In this case, the transformer is ρ—instead of predicting latents directly, it predicts noise. Also, note that here the tokens are not generated autoregressively; instead, the full structure is generated in parallel, with the noise for each latent dependent on every other latent (through self-attention). This parallelism improves efficiency because all tokens can be generated as a single batch; it also improves artefact quality due to the additional attentional information available (no masking).

Putting It Altogether Algorithm 1 shows high-level pseudocode for how the autoregressive transformer generates an artefact. It works by iteratively sampling one latent vector at a time from p (lines 2 and 9), computing layer embeddings for all latents generated so far (lines 5-7), and using the final layer embeddings to update p (line 8). Once all latents are generated, they are decoded into tokens that compose the final artefact according to structure G (lines 10-12).

Alternatively, Algorithm 2 shows instead how a diffusion transformer generates an artefact. The system works by generating a complete batch of random latent vectors (line 1), iteratively denoising them by computing the layer embeddings for all latents (lines 3-6), using the final layer embeddings to predict noise for each one (line 7) and subtracting the noise from the latents (lines 8-9). Once the latents are denoised, they are decoded into tokens that compose the final artefact according to structure G (lines 10-12)).

[6]If the transformer is generating autoregressively and must sample from a finite vocabulary, the latent space for the codec must be discretized in some way, for example using a quantized codebook. If the transformer is instead diffusion-based, this is unnecessary.

Algorithm 1 GPT($\mathbf{d}$)

1: $m = 1$
2: $l_m \sim p$
3: **while** $l_m \neq$ EOS **do**
4: $\quad m = m + 1$
5: $\quad \mathbf{b}_0 = \tau_0(l_{<m})$
6: $\quad$ **for** $k = 1$ to K **do**
7: $\quad\quad \mathbf{b}_k = \tau_k(\mathbf{b}_{k-1}, A_k, \mathbf{d}, C_k)$
8: $\quad p = f(\tau_K(\mathbf{b}_{K-1}, A_K, \mathbf{d}, C_K))$
9: $\quad l_m \sim p$
10: **for** $i = 1$ to m **do**
11: $\quad a_i = \delta(l_i)$
12: **return** compose($\mathbf{a}, G$)

Algorithm 2 DiT($\mathbf{d}$)

1: $l_1^T, ..., l_m^T \sim \mathcal{N}(0, 1)$
2: **for** $t = T$ to 0 **do**
3: $\quad$ **for** $i = 1$ to m **do**
4: $\quad\quad b_i = \tau_0(l_i^t)$
5: $\quad$ **for** $k = 1$ to K **do**
6: $\quad\quad \mathbf{b}_k = \tau_k(\mathbf{b}_{k-1}, A_k, \mathbf{d}, C_k)$
7: $\quad \mathbf{e}^t = f(\tau_K(\mathbf{b}_{K-1}, A_K, \mathbf{d}, C_K))$
8: $\quad$ **for** $i = 1$ to m **do**
9: $\quad\quad l_i^{t-1} = l_i^t - e_i^t$
10: **for** $i = 1$ to m **do**
11: $\quad a_i = \delta(l_i^0)$
12: **return** compose($\mathbf{a}, G$)

There are, of course, many variants on these approaches, but this abstraction level captures the essence of modern generative AI at this moment in time. For example, although we have presented autoregression and diffusion as separate approaches to generation, many systems now combine these paradigms (e.g., a model may use autoregression to produce latent tokens followed by diffusion-based refinement, or use diffusion within a latent space learned by an autoencoder); augmentation techniques, such as retrieval-augmented generation (RAG) (Lewis et al. 2020), can be accounted for as additional sources of conditioning; and alignment methods, such as reinforcement learning from human feedback (RLHF) (Kaufmann et al. 2025), can be considered as additional training signal.

Training

The processes discussed above: ϵ (encoding), δ (decoding), α, χ (attention), β, τ (embedding), the probability distribution p, and helper functions (like f in Algorithms 1 and 2), as well as mechanisms for connecting them all, are implemented using DNN models composed of tunable parameters—real-valued numbers that represent the mutable part of the system. They interact with input to the system as well as with each other and define the functional behavior of the system—what artefacts it generates. Modern machine learning models can contain billions of parameters, all of which must have their values set "correctly" via training in order for the model to exhibit reasonable/desirable/useful

behavior. Training is an iterative, inductive process that requires examples of the desired model behavior and an objective function which provides a measure of error between the model's current behavior and the desired example.

Modern generative systems are often composed of many subsystems and models that work together and are typically trained end-to-end. This can be expensive in terms of many resources including time, money, energy and data. In particular, training modern foundation models can require months of time, terabytes of data and enough energy to power a medium-sized town. The training process consists of six main components, numbered 1-6 in red in Figure 1:

1. data collection and preparation
2. training a codec
3. encoding training data into latent space
4. noising the latent training data
5. embedding metadata
6. training the transformer

Data Collection and Preparation Training a generative system requires both a set of examples $X = \{x_i\}$ of the kinds of artefacts to be generated and a set of associated metadata $Y = \{y_i\}$ describing those artefacts. The examples X serve as training targets for generation and the metadata Y as a proxy for conditioning prompts. Because the models are very large (in terms of number of parameters), the set X must also be very large in order to avoid overfitting and memorization. Also, because the amount and type of user prompt information that may be encountered is practically limitless and ill-defined, it is critical that the set Y be as rich as possible. For example:

- for the domain of music, X is a set of songs in some audio format and Y may contain text descriptions of the music; classifying information such as time signature, key, genre, artist; instrumentation; audio of similar music.
- for the domain of images, X is a set of images in some image format and Y may contain text descriptions of the images.
- for the domain of language, X is a set of documents and Y may contain classifying information such as topic, author, source; translations to other languages; audio rendering of the text.
- for the domain of video, X is a set of videos in some video format, and Y may contain text descriptions of the video; summarization information; accompanying audio.

It is important that the data be paired so that $y_i \in Y$ is the metadata associated with $x_i \in X$. Finally, the training data is tokenized $x_i = a_{i1}, ..., a_{iq_i}$, with the tokens a_{ij} domain-specific. For example, if the training data were 4096×4096 pixel images, these could be partitioned into 32×32 non-overlapping patches, resulting in an image consisting of $16,384$ tokens, each of which is a 32×32 image patch.

Training a Codec During the generation process, an artefact is generated as a collection of latents. What makes a good latent representation for this process must be learned by the codec. It does this by learning to take an input token, compress it using an encoder ϵ and then decompress it using a decoder δ such that the output is equal to the input. In other words, it learns to compress the input into a latent representation from which the original can be recovered. The model is trained using an objective function that computes the difference between the input token a and the output of encoding and decoding it, $\delta(\epsilon(a))$, with that error driving the update of encoder and decoder model parameters. When training is complete, the encoder and decoder can be used to convert an original data representation into a latent representation (encoder) and to convert a latent representation back into the original data representation (decoder).

Encoding Training Data into Latent Space Once the codec is trained, its encoder model ϵ is used to build a latent version $\widehat{X}$ of the training data X. This is done by converting the tokenized training data $x_i = a_{i1}, ..., a_{iq_i}$ into a collection of latents $l_{ij} = \epsilon(a_{ij}), 1 \le i \le |X|, 1 \le j \le q_i$. Continuing the image example above, the 32×32 image patch tokens consist of 1024 real-valued numbers. If the codec bottleneck is 128, it will compress each of the 1024-vectors down to a latent 128-vector.[7]

Noising the Latent Training Data If the transformer uses diffusion, a noisy version $\widetilde{X}$ of the latent training data $\widehat{X}$ must be created to train the transformer to predict noise. $\widetilde{X}$ is constructed by adding varying levels of Gaussian noise to the latents l_{ij}. This process of adding noise is called diffusion and makes the assumption that noise can be modeled by a mathematical schedule over T timesteps. $\widetilde{X}$ consists of $(t, \tilde{\mathbf{l}}, \mathbf{e})$ triples, where t is a time step index, $\tilde{\mathbf{l}}$ is a batch of related latents (composing all or a part of an artefact) that contains t levels of added noise, and $\mathbf{e}$ is a batch of the t-th level of noise that was added to the batch of latents during diffusion and which must be removed from the latents to recover a less noisy version that contains only $t - 1$ levels of noise. To create the data set, a noise schedule of T steps is applied to each batch of latents $\mathbf{l}_i$, resulting in T examples for each batch, one for each noise level. If $\widehat{X} = \{\mathbf{l}_i\}$, then $\widetilde{X} = \{(1, \tilde{\mathbf{l}}_i^1, \mathbf{e}_1), ..., (T, \tilde{\mathbf{l}}_i^T, \mathbf{e}_T)\}$.

Embedding Metadata Because the metadata is used as a proxy for user prompting (see above), it must be processed in the same way: tokenized as a collection, $y_i = c_{i1}, ..., c_{il_i}$, and then converted to a collection of embeddings, $\beta(c_{ij})$, $1 \le i \le |Y|, 1 \le j \le l_i$. The result is an embedded version $\widehat{Y}$ of the training metadata Y.

Training the Transformer Once training data, metadata sets $\widehat{X}, \widehat{Y}$ or $\widetilde{X}, \widehat{Y}$ have been created, they are used to train the transformer model: either to autoregressively generate

[7] And, since there are an infinite number of such latents, something like a codebook will be used to quantize the latent space into a finite set; for example, the codebook (itself parameterized and learned during training) might have $65,536$ entries and each compressed latent is mapped to the closest of these, resulting in a latent vocabulary of size $65,536$. During autoregression, When the distribution p is sampled, one of these $65,536$ latent codebook entries is the result.

Algorithm 3 TRAINGPT($\widehat{X}, \widehat{Y}$)

1: Initialize transformer model parameters
2: **while** not done **do**
3: Randomly choose a training pair $(\hat{x}_i = l_{i1}, ...l_{iq_i}, \hat{y}_i)$
4: **for** $j = 1$ to q_i **do**
5: using $l_{i1}, ..., l_{ij}$ as input and $\hat{y}_i$ as conditioning, run lines 5-8 of Algorithm 1 to get updated p
6: `error` = `cross_entropy`$(p(l_{ij+1}))$
7: Backpropagate `error` through model
8: Update model parameters to reduce `error`

Algorithm 4 TRAINDIT($\widetilde{X}, \widehat{Y}$)

1: Initialize transformer model parameters
2: **while** not done **do**
3: Randomly choose a training pair $(\tilde{x}_i = (t, \tilde{\mathbf{l}}_i^t, e_i^t), \hat{y}_i)$
4: using $\tilde{\mathbf{l}}_i^t$ as input and $\hat{y}_i$ as conditioning, run lines 3-7 of Algorithm 2 to get noise predictions $\tilde{\mathbf{e}}_i^t$
5: `error` = $(\mathbf{e}_i^t - \tilde{\mathbf{e}}_i^t)^2$
6: Backpropagate `error` through model
7: Update model parameters to reduce `error`

a collection of latents or to denoise a collection of random vectors into a collection of latents.

The training process is similar for both transformer models, with the difference being in how error is computed—the two approaches have different objective functions: minimizing cross entropy for autoregression and minimizing squared-error for diffusion. See Algorithms 3 and 4.

Also, note that for the case of autoregression, masking is used to avoid cheating during training. Even though the training input $\hat{x}_i$ contains the entire sequence $l_{i1}, ...l_{iq_i}$, when predicting latent l_{ij+1} only latents $l_{i1}, ..., l_{ij}$ are available for attention, while any other tokens in the structure are masked (e.g., in language prediction, tokens later in the sequence are masked, and the attention matrices A_k are lower triangular).

Characterizing Process in CC Systems

We will examine this modern generative AI system through the lens of four different analytical accountings of the creative process in computational systems: the Creative Systems Framework (CSF) (Wiggins 2006), a formal approach to building a CC system (Ventura 2017), the FACE framework (Colton, Charnley, and Pease 2011) and a spectrum of computational creativity (Ventura 2016). For each, we briefly summarize the key points of the method and then examine the model of Figure 1 from that perspective. The intent here is twofold: to demonstrate that these classical CC frameworks are as useful in this new age of modern AI as they have ever been, if not more so; and to provide some perspective on what these modern AI models do and do not yet admit in terms of creativity, suggesting some research directions to benefit both CC and AI generally.

Creative Systems Framework

Wiggins's formalization of the computational creative process is elegantly summarized in the following equation:

$$c_{out} = [\![\mathscr{E}]\!](\langle\!\langle \mathscr{R}, \mathscr{T}, \mathscr{E} \rangle\!\rangle^{\diamond}(\{\top\}))$$

The process is proscribed by three rule sets: $\mathscr{R}, \mathscr{T}, \mathscr{E}$. $\mathscr{R}$ is a set of rules that define the conceptual space on which the creative process will operate—the domain in which potential output artefacts live; $\mathscr{T}$ is a set of rules that define how that conceptual space should be traversed while searching for potential output artefacts; $\mathscr{E}$ is a set of rules that define how potential output artefacts should be evaluated to determine their "fitness" for output. The notation $\langle\!\langle \cdot \rangle\!\rangle$ represents a compilation mechanism that consumes the three rule sets and produces a search function (not explicitly represented here, so for convenience, we will call it ϕ) that explores the universe $\mathscr{U}$ of possible artefacts and returns potential artefacts. At a timestep t, the search function ϕ takes as input a set $c_{t-1} \in \mathscr{U}$ of potential artefacts and returns another set $c_t \in \mathscr{U}$ of potential output artefacts. The notation $[\![\cdot]\!]$ represents another compilation mechanism that consumes only the $\mathscr{E}$ rule set and produces an evaluation function (again not explicitly represented here, so for convenience, we will call it μ). The evaluation function μ takes as input the set c_T for some final timestep T and returns the subset $c_{out} \in c_T$ which the rules in $\mathscr{E}$ identify as "fit" or "interesting" or "valuable" as output artefacts. The initial input to the function ϕ is the set $\{\top\}$ containing only the empty concept $\top$. The $\diamond$ notation represents the idea that the search process implemented by ϕ is recursive, with the output at one time step acting as the input for the next:

$$c_t = \phi(c_{t-1})$$
$$c_0 = \{\top\}$$

with the final output of the creative process then being:

$$c_{out} = \mu(c_T)$$

The function of the rule set $\mathscr{R}$ is represented by a combination of the structure G and the latent vocabulary, which is learned by the codec (ϵ, δ). Wiggins suggests that one type of creativity involves the system occasionally ignoring the conceptual rules in $\mathscr{R}$, but this is difficult to do for modern generative AI systems: in the case of GPT, it is only possible if composing latents from the vocabulary according to G can result in something "weird", which by design is at best very unlikely; for DiT, this is less obvious, since in theory, it can generate infinitely many denoising trajectories, but the resulting latent will still be decoded. If the decoder is stable, the resulting token is still likely to be unsurprising, and, of course, the structure G still imposes significant constraints that can not be circumvented.

The search function ϕ compiled from the rule sets $\mathscr{T}, \mathscr{R}, \mathscr{E}$ is represented in the embeddings β, τ, attention mechanisms α, χ and the mapping function f, which influence either the next sample l_m or the predicted noise $\mathbf{e}^t$. c_0 could be empty as suggested by Wiggins and this would correspond to the case of no user prompt. In most cases, however, c_0 will be the embedded user prompt $\mathbf{d}$ =

$g(\beta_1(\sigma), ..., \beta_n(\sigma))$. c_{t-1} is then equivalent either to the set of previous latents $l_{<m}$ (autoregression) or to the batch of slightly noisier latents l_i^t (diffusion).

The evaluation function μ compiled from the rule set $\mathscr{E}$ is not explicitly represented. "Evaluation" only happens in the sense of minimizing an objective function during training, with the resulting "quality" encoded in the model parameters. During generation, the model is biased toward latents that meet the objective. For GPT, this happens because the next latent is influenced by previous latents as well as conditioning—the next latent should have high-probability given those; for DiT, the model should predict noise so that the latents' final positions are close (in some semantic sense) to others in the batch and to conditioning embeddings. The model offers no mechanism for evaluation of final product in any other sense. To be fair, automatic artifact evaluation is a non-trivial problem that has not been solved in general.

Rule sets $\mathscr{T}$ and $\mathscr{R}$ are both encoded in the training data $\widehat{X}, \widehat{Y}(\widetilde{X}, \widehat{Y})$, and the compilation process $\langle\!\langle \cdot \rangle\!\rangle$ is the training process. New data will produce a new codec (and thus new vocabulary) and new G, and so also a new $\mathscr{R}$. New data will produce new embeddings and attention functions and thus a new $\mathscr{T}$. It can be argued that $\mathscr{E}$, too, is encoded in the data, but there is no analog for the separate compilation process $[\![\cdot]\!]$.[8]

How To Build a CC System

Ventura's evolution of the ideas originated in the CSF is more directly functional and can again be summarized with a single equation:

$$a = \Theta_p(\tau(\Theta_g(\gamma(\lambda_c(K), \rho))))$$

Considering this from the inside out, K is a knowledge-base of data and relationships from the domain of interest. $\lambda_c()$ is some learning mechanism that consumes the knowledge-base K to produce a conceptualization C of the domain, in the form of some learned model. ρ represents a catch-all for additional information beyond what the model C provides (inspiration sources, randomness, examples) that may benefit the artefact generation process (and it may, of course, be null). γ is a generative process that makes use of both the conceptual model C and the supplemental information ρ to produce a potential output artefact g. Importantly, this potential artefact is represented in some *genotypic* form consumable by the system but likely not understandable by other systems or humans. The genotypic potential artefact g is evaluated (for "quality", "value", "interestingness" by the function $\Omega_g()$, which returns either g (if it meets some threshold of "goodness" or the empty concept $\perp$ (if it does not). If that output is evaluated as good, the result is then translated into a domain-appropriate phenotypic representation p via a translator function τ. Because this representation is domain-specific and phenotypic, it will be understandable by other systems and humans. Finally, the phenotypic representation p of the potential output artefact is re-evaluated with a *different* evaluation function $\Omega_p()$. If it again passes muster, it is output; otherwise, the system outputs the empty concept $\perp$.

Not surprisingly, given the nature of the formulation, the mapping here is mostly straightforward:

- The knowledge-base K is the training data $\widehat{X}, \widehat{Y}(\widetilde{X}, \widehat{Y})$ along with any supplemental knowledge sources (e.g., RAG).
- The learning mechanism λ_c is the training process that produces the embeddings β, τ, attention mechanisms α, χ and the mapping function f.
- The supplemental information ρ is the user-supplied prompt σ.
- The generative process γ is implemented by iterating over the embeddings/attention as shown in Algorithms 1 and 2; note that unlike Wiggins, there is no explicit iteration in the formulation here.
- The genotypic representation g is the latent l; note that for systems that operate directly on tokens from the target domain, there is no genotypic representation.
- The genotypic evaluation function Θ_g is not explicitly represented but is to some extent implicitly encoded in the probability distribution p or predicted noise $\mathbf{e}$, using the same argument as for Wiggins's $\mathscr{E}$; however, there is no notion of aborting the process as defined here.
- The translator τ is the decoder δ.
- The phenotypic evaluation function Θ_p is not represented at all, which this formulation makes more explicit than does that of Wiggins because of the two levels of evaluation. Note that the argument for alignment-as-a-mechanism looks even weaker now because it is likely not done on the final product, which is important in this accounting. And again, there is no mechanism for aborting the process.

FACE Framework

Colton, Charnley, and Pease's approach elucidates eight different facets of a computationally creative system, characterized by four levels of complexity: expressions, concepts, aesthetics, framing; and two degrees of abstraction: ground-level and process-level. These combinatorically generate the eight types of creative act for which the FACE framework accounts:

E^g: an expression of a concept
E^p: a method for generating expressions of a concept
C^g: a concept
C^p: a method for generating concepts
A^g: an aesthetic measure
A^p: a method for generating aesthetic measures
F^g: an item of framing information
F^p: a method for generating framing information

Creative acts superscripted with a g are ground-level processes that generate concrete products. Those superscripted with a p are meta-level acts that generate processes for generating concrete products. Characterizing system behavior is done with a tuple that can contain exactly 0 or 1 of each of the eight acts listed above. For further precision, it is possible to attribute creative acts to a human user/developer

[8] One *might* be able to argue that alignment techniques are an attempt at something like this but probably not convincingly.

with an overbar notation. This is a powerful ontological approach that allows a careful characterization of what types of creativity a system is capable. For example, the tuple $\langle E^g, \overline{E^p}, C^g, \overline{C^p}, \overline{A^g} \rangle$ describes a system that employs a user-provided process $\overline{c^p}$ to generate a concept c^g. It then employs a user-provided process $\overline{e^p}$ to generate an expression e^g of the concept c^g. Finally, it applies a user-provided aesthetic measure $\overline{a^g}$ to the pair (c^g, e^g).

Considering each of the eight types of creative act for the modern generative AI system:

- E^g is the function $\texttt{compose}(a, G)$ from Algorithms 1 and 2.
- E^p is represented by the embeddings β, τ, attention mechanisms α, χ and the mapping function f.
- C^g is defined by the training data $\widehat{X}, \widehat{Y}(\widetilde{X}, \widehat{Y})$, for the same reasons given above for Wiggins' $\mathscr{R}$.
- C^p is not represented, for similar arguments as those given above—if C^g is defined by the training data, then generating a new concept is effected by collecting new training data, and current systems have no such mechanism.
- A^g is to some extent implicitly encoded in the probability distribution p or predicted noise $\mathbf{e}$, using the same arguments as for Wiggins's $\mathscr{E}$ and Ventura's Θ_g.
- A^p is not represented; if aesthetics are encoded by the objective function, then generating aesthetics must involve inventing new objective functions, which modern systems do not do.
- F^g is not represented; note such functionality might be thought of as producing the opposite of a prompt.
- F^p is not represented.

To summarize, Colton, Charnley, and Pease might characterize the modern generative system thusly: $\langle E^g, E^p, \overline{C^g}, \overline{A^g} \rangle$, with the overbars signifying that human developers have selected the training data that produces sets $\widehat{X}, \widehat{Y}(\widetilde{X}, \widehat{Y})$.

Computational Creativity Spectrum

Finally, Ventura provides a simple, elegant if somewhat whimsical account of a spectrum of computationally creative processes that ranges from "from *definitely-mere-generation* to *definitely-not-mere-generation*". It offers seven prototypical, abstract algorithms that fall at intervals along this spectrum and uses these landmark points to argue about where on the spectrum a system might become something more than merely generative. These landmark algorithms characterize different levels of generative/creative ability:

1. *Randomization*: pure stochastic generation
2. *Plagiarization*: randomly choose an item from the inspiration/training set
3. *Memorization*: model the inspiration/training set and regenerate an item from that set using the model
4. *Generalization*: model the inspiration/training set using some form of regularization that forces a compressed representation and generate an item using the regularized model
5. *Filtration*: introduce a fitness measure and use it to generate-and-test using a model of the inspiration/training set
6. *Inception*: additionally, introduce a knowledge-base and use it as part of the generative mechanism in the generate-and-test loop
7. *Creation*: additionally, introduce the ability for grounded-perception and evaluate the (grounded) perception of items in the generate-and-test loop

By comparing a CC system with the prototypical landmark algorithms associated with each level, we can situate the system along the spectrum. Considering each landmark in turn:

1. *Randomization* is realized by a model produced by stopping training after line 1 in Algorithms 3 and 4.
2. *Plagiarization* is implemented by skipping training and simply building a lookup table for the training data $\widehat{X}, \widehat{Y}(\widetilde{X}, \widehat{Y})$.
3. *Memorization* is viable if the number of model parameters is large compared to the dataset. In fact, this has been recently shown to be a problem even for systems trained on lots of data (Ahmed et al. 2026).
4. *Generalization* is the normal expectation for modern DNNs if the model size and amount of training data are compatible
5. *Filtration* is not done explicitly, but Ventura argues that filtering can also be done implicitly by baking it into the generative process, and that is done in modern generative systems during training—it is encoded in the embeddings β, τ, attention mechanisms α, χ and the mapping function f.
6. *Inception* is less clear. One might argue that training on vast amounts of data of many kinds can result in the ability to do things like zero- or few-shot generation that appears to suggest the (implicit) incorporation of knowledge. However, modern generative systems are often fluently wrong. Techniques like RAG attempt to incorporate knowledge sources more explicitly.
7. *Creation* is not yet reached by modern systems because they (at the least) do not incorporate grounded perception. Incorporating perceptual measures (such as perceptual evaluation of audio quality [PEAQ] (Thiede et al. 2000)) into the pipeline may be a step in the right direction, but the intent here is deeper than a model of human perception, it is declaring the necessity of *groundedness*, which is lacking in modern systems.

Conclusion

Unsurprisingly, modern generative AI systems show well against all four accountings of CC process. However, they also exhibit areas of weakness, and, what is more, these weaknesses are fairly consistent across methodologies.

Modern generative systems are very effective at producing plausible output, but they are constrained in how they do it, and by design they are minimally creative—the objective function explicitly penalizes deviation from the training data. Is there some way to facilitate "good" deviation? It has recently been shown that neither temperature (Peeperkorn et al. 2024) nor prompting (Morain and Ventura 2025) provide

such a mechanism. In fact, it is an open question whether the mechanism of an objective function is sufficient for computing creativity "error". "The obvious" approach is to design an objective function that rewards novelty and value (or, conversely, penalizes the lack thereof), but it is not at all obvious how this should be done, even for a specific domain, let alone in a domain-agnostic way that could apply to generative AI models in general.

Explicit artefact-level evaluation is also still lacking. Is there a way to incorporate this directly into the generative system? Or, could it be implemented as a separate module that is incorporated into the pipeline in such a way as to allow feedback into the generative mechanism? And, relatedly, is there a way for generative models to recognize "bad" results, throw them away and start over? Some modern chat models are beginning to entertain this idea in the sense of incorporating the ability to answer "I don't know" instead of fluently generating falsities; however, at least anecdotally this is currently more annoying than useful.

The invention of new concepts and aesthetic measures is beyond the capability of current systems. For modern generative AI systems, these can be framed as choosing the training data and generation of objective functions, respectively. Both of these meta-level tasks are currently performed by the (human) designers of AI systems. What would it take to allow instead the systems themselves to make these choices? At the least, it seems some meta-level objective function would be required to evaluate the efficacy of such choices. This could possibly be realized as a type of meta-learning system whose outer loop makes choices about data and/or objectives and whose inner loop makes use of those choices. Base-level objective error would have to be propagated back to the outer-level to inform updates to the objective function and/or training data. In the case of training data, this might be realized using something like task distillation (Wilhelm and Ventura 2025); in the case of objective function, it is less clear how this might be done, as the natural approach would trivially modify the objective function to eliminate error in the inner loop. This begs a compelling question—what makes a "good" objective function (which in this context is just a variation on the age-old question of what makes a "good" aesthetic)?

Modern generative AI systems lack the perceptual grounding necessary to "understand" what they are doing. One mechanism by which such grounding might provide this kind of understanding is by acting as a bridge between neural and cognitive models (Maher, Ventura, and Magerko 2023). Such a bridge may provide a synergism in which generative models can provide a mastery of domain (language, image, audio, etc.) that is not currently possible with cognitive models, while cognitive systems can provide models for guiding and evaluating the output from generative models, aligning it with human values, offering ethical guardrails and aesthetic judgments, and even improving explainability and trustworthiness.

Framing is not (yet, at least) generally considered in modern generative systems. Is framing something we should care about in this context? If so, how is it different than product? One (obvious) possible approach is to have another system produce framing, given the product, but that is problematic because it is more analogically similar to a critic, which is not what is wanted. Another possible approach here might be some kind of co-creative agentic system.

Speaking of co-creativity, it would be interesting to develop a complementary analysis that considers modern generative AI as a co-creator with a human partner. Of course, incorporating a human into the system changes the kinds of questions that are interesting to ask. To the extent that humans possess creativity, the system will *de facto* be creative. Instead, we may ask if the AI improves the human's creativity/experience in some way, and, if so, how and by how much; or, conversely, if it possibly makes the experience worse and in what way; if the pair is somehow more creative/productive than either partner individually and in what ways; if the partnership admits new kinds of creativity not possible without it; etc.

References

[Ahmed et al. 2026] Ahmed, A.; Cooper, A. F.; Koyejo, S.; and Liang, P. 2026. Extracting books from production language models. https://arxiv.org/abs/2601.02671.

[Besold and Plaza 2015] Besold, T. R., and Plaza, E. 2015. Generalize and blend: Concept blending based on generalization, analogy, and amalgams. In *Proc. 6th Int. Conf. on Comp. Creativity*, 150–157.

[Boden 1992] Boden, M. 1992. *The Creative Mind*. London: Abacus.

[Boden 1998] Boden, M. 1998. Creativity and artificial intelligence. *Artificial Intelligence* 103(1–2):347–356.

[Carlson et al. 2016] Carlson, K.; Pasquier, P.; Tsang, H. H.; Phillips, J.; Schiphorst, T.; and Calvert, T. 2016. Cochoreo: A generative feature in idanceforms for creating novel keyframe animation for choreography. In *Proc. 7th Int. Conf. on Comp. Creativity*, 380–387.

[Colton, Charnley, and Pease 2011] Colton, S.; Charnley, J.; and Pease, A. 2011. Computational creativity theory: The FACE and IDEA descriptive models. In *Proc. 2nd Int. Conf. on Comp. Creativity*, 90–95.

[Colton 2012] Colton, S. 2012. The painting fool: Stories from building an automated painter. In *Computers and Creativity*. Berlin: Springer-Verlag. 3–38.

[Confalonieri, Plaza, and Schorlemmer 2016] Confalonieri, R.; Plaza, E.; and Schorlemmer, M. 2016. A process model for concept invention. In *Proc. 7th Int. Conf. on Comp. Creativity*, 338–345.

[Cook, Colton, and Gow 2016] Cook, M.; Colton, S.; and Gow, J. 2016. The ANGELINA videogame design system—part i. *IEEE Transactions on Computational Intelligence and AI in Games* 9:192–203.

[Cope 2005] Cope, D. 2005. *Computer Models of Musical Creativity*. Cambridge, MA: MIT Press.

[Cunha et al. 2017] Cunha, J. M.; Gonçalves, J.; Martins, P.; Machado, P.; and Cardoso, A. 2017. A pig, an angel and a cactus walk into a blender: A descriptive approach to visual blending. In *Proc. 8th Int. Conf. on Comp. Creativity*, 80–87.

[Elgammal et al. 2017] Elgammal, A.; Liu, B.; Elhoseiny, M.; and Mazzone, M. 2017. CAN: Creative adversarial networks, generating "art" by learning about styles and deviating from style norms. In *Proc. 8th Int. Conf. on Comp. Creativity*, 96–103.

[Gervás 2009] Gervás, P. 2009. Computational approaches to storytelling and creativity. *AI Magazine* 30(3):49–62.

[Gervás 2011] Gervás, P. 2011. Dynamic inspiring sets for sustained novelty in poetry generation. In *Proc. 2nd Int. Conf. on Comp. Creativity*, 111–116.

[Góes et al. 2023] Góes, F.; Sawicki, P.; Grzes, M.; Volpe, M.; and Brown, D. 2023. Is GPT-4 good enough to evaluate jokes? In *Proc. 14th Int. Conf. on Comp. Creativity*, 367–371.

[Hull and Colton 2007] Hull, M., and Colton, S. 2007. Towards a general framework for program generation in creative domains. In *Proc. Fourth Int. Joint Workshop on Computational Creativity*.

[Inácio and Oliveira 2025] Inácio, M. L., and Oliveira, H. G. 2025. A full pipeline for context-aware pun generation. In *Proc. 16th Int. Conf. on Comp. Creativity*, 1–11.

[Ismayilzada, Stevenson, and van der Plas 2025] Ismayilzada, M.; Stevenson, C.; and van der Plas, L. 2025. Evaluating creative short story generation in humans and large language models. In *Proc. 16th Int. Conf. on Comp. Creativity*, 145–155.

[Jordanous 2012] Jordanous, A. 2012. A standardised procedure for evaluating creative systems: Computational creativity evaluation based on what it is to be creative. *Cognitive Computation* 4:246–279.

[Jordanous 2016] Jordanous, A. 2016. Four PPPPerspectives on computational creativity in theory and in practice. *Connection Science* 28(2):194–216.

[Kaufmann et al. 2025] Kaufmann, T.; Weng, P.; Bengs, V.; and Hüllermeier, E. 2025. A survey of reinforcement learning from human feedback. https://arxiv.org/abs/2312.14925.

[Kingma and Welling 2019] Kingma, D. P., and Welling, M. 2019. An introduction to variational autoencoders. *Foundations and Trends in Machine Learning* 12:307–392.

[Lewis et al. 2020] Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; tau Yih, W.; Rocktäschel, T.; Riedel, S.; and Kiela, D. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, 9459–9474.

[Li et al. 2012] Li, B.; Zook, A.; Davis, N.; and Riedl, M. 2012. Goal-driven conceptual blending: A computational approach for creativity. In *Proc. 3rd Int. Conf. on Comp. Creativity*, 9–16.

[Li et al. 2024] Li, Y.; Yuan, R.; Zhang, G.; Ma, Y.; Chen, X.; Yin, H.; Xiao, C.; Lin, C.; Ragni, A.; Benetos, E.; Gyenge, N.; Dannenberg, R.; Liu, R.; Chen, W.; Xia, G.; Shi, Y.; Huang, W.; Wang, Z.; Guo, Y.; and Fu, J. 2024. MERT: Acoustic music understanding model with large-scale self-supervised training. https://arxiv.org/abs/2306.00107.

[Liapis, Yannakakis, and Togelius 2015] Liapis, A.; Yannakakis, G. N.; and Togelius, J. 2015. Constrained novelty search: A study on game content generation. *Evolutionary Computation* 23(1):101–129.

[Liu et al. 2019] Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A robustly optimized BERT pretraining approach. https://arxiv.org/abs/1907.11692.

[Llano et al. 2014] Llano, M. T.; Hepworth, R.; Colton, S.; Gow, J.; Charnley, J.; Lavrač, N.; Žnidaršič, M.; Perovšek, M.; Granroth-Wilding, M.; and Clark, S. 2014. Baseline methods for automated fictional ideation. In *Proc. 5th Int. Conf. on Comp. Creativity*, 211–219.

[Machado and Cardoso 2002] Machado, P., and Cardoso, A. 2002. All the truth about NEvAr. *Applied Intelligence* 16(2):101–118.

[Maher, Ventura, and Magerko 2023] Maher, M. L.; Ventura, D.; and Magerko, B. 2023. The grounding problem: An approach to the integration of cognitive and generative models. In *Proc. AAAI Fall Symposium*, 320–325.

[Martins et al. 2004] Martins, J. M.; Pereira, F. C.; Miranda, E. R.; and Cardoso, A. 2004. Enhancing sound design with conceptual blending of sound descriptors. In *Proc. First Int. Joint Workshop on Computational Creativity*.

[Morain and Ventura 2025] Morain, R., and Ventura, D. 2025. Is prompt engineering the creativity knob for large language models? In *Proc. 16th Int. Conf. on Comp. Creativity*, 30–40.

[Morris et al. 2012] Morris, R.; Burton, S.; Bodily, P.; and Ventura, D. 2012. Soup over bean of pure joy: Culinary ruminations of an artificial chef. In *Proc. 3rd Int. Conf. on Comp. Creativity*, 119–125.

[Nath, Dayan, and Stevenson 2024] Nath, S. S.; Dayan, P.; and Stevenson, C. 2024. Characterising the creative process in humans and large language models. In *Proc. 15th Int. Conf. on Comp. Creativity*, 325–330.

[Oliveira, Costa, and Pinto 2016] Oliveira, H. G.; Costa, D.; and Pinto, A. M. 2016. One does not simply produce funny memes! In *Proc. 7th Int. Conf. on Comp. Creativity*, 238–245.

[O'Donoghue, Bohan, and Keane 2006] O'Donoghue, D.; Bohan, A.; and Keane, M. 2006. Seeing things: Inventive reasoning with geometric analogies and topographic maps. *New Generation Computing* 24(3):267–288.

[Peeperkorn et al. 2024] Peeperkorn, M.; Kouwenhoven, T.; Brown, D.; and Jordanous, A. 2024. Is temperature the creativity parameter of large language models? In *Proc. 15th Int. Conf. on Comp. Creativity*, 226–235.

[Rabeyah et al. 2025] Rabeyah, A. A.; Goes, F.; Volpe, M.; and Medeiros, T. 2025. Do LLMs agree on the creativity evaluation of alternative uses? In *Proc. 16th Int. Conf. on Comp. Creativity*, 217–227.

[Radford et al. 2021] Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; Krueger, G.; and Sutskever, I. 2021. Learning transferable visual models from natural language supervision. In *Proc. 38th Int. Conf. on Machine Learning*.

[Riedl and Young 2006] Riedl, M., and Young, M. 2006. Story planning as exploratory creativity: Techniques for expanding the narrative search space. *New Generation Computing* 24(3):303–323.

[Ritchie et al. 2007] Ritchie, G.; Manurung, R.; Pain, H.; Waller, A.; Black, R.; and O'Mara, D. 2007. A practical application of computational humour. In *Proc. Fourth Int. Joint Workshop on Computational Creativity*.

[Ritchie 2007] Ritchie, G. 2007. Some empirical criteria for attributing creativity to a computer program. *Minds and Machines* 17:76–99.

[Saunders and Gero 2001] Saunders, R., and Gero, J. S. 2001. A curious design agent: A computational model of novelty-seeking behaviour in design. In *Proc. of CAADRIA*.

[Smith and Mateas 2011] Smith, A. M., and Mateas, M. 2011. Knowledge-level creativity in game design. In *Proc. 2nd Int. Conf. on Comp. Creativity*, 16–21.

[Strapparava, Valitutti, and Stock 2007] Strapparava, C.; Valitutti, A.; and Stock, O. 2007. Automatizing two creative functions for advertising. In *Proc. Fourth Int. Joint Workshop on Computational Creativity*.

[Taneja, Segal, and Goodwin 2023] Taneja, K.; Segal, R.; and Goodwin, R. 2023. Monte Carlo tree search for recipe generation using GPT-2. In *Proc. 14th Int. Conf. on Comp. Creativity*, 20–28.

[Thiede et al. 2000] Thiede, T.; Treurniet, W.; Bitto, R.; Schmidmer, C.; Sporer, T.; Beerends, J. G.; Colomes, C.; Keyhl, M.; Stoll, G.; Brandenburg, K.; and Feiten, B. 2000. PEAQ–the ITU standard for objective measurement of perceived audio quality. *Journal of the Audio Engineering Society* 48(1/2):3–29.

[Tian 2024] Tian, Y. 2024. DiffCJK: Conditional diffusion model for high-quality and wide-coverage CJK character generation. In *Proc. 15th Int. Conf. on Comp. Creativity*, 2–11.

[Toivanen et al. 2012] Toivanen, J. M.; Toivonen, H.; Valitutti, A.; and Gross, O. 2012. Corpus-based generation of content and form in poetry. In *Proc. 3rd Int. Conf. on Comp. Creativity*, 211–215.

[Vaswani et al. 2017] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Łukasz Kaiser; and Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, 6000–6010.

[Veale 2006] Veale, T. 2006. Re-representation and creative analogy: A lexico-semantic perspective. *New Generation Computing* 24(3):223–240.

[Veale 2025] Veale, T. 2025. Me, myself and irony: Modelling the deceptive creativity of irony with large language models. In *Proc. 16th Int. Conf. on Comp. Creativity*, 12–19.

[Ventura 2016] Ventura, D. 2016. Mere generation: Essential barometer or dated concept. In *Proc. 7th Int. Conf. on Comp. Creativity*, 17–24.

[Ventura 2017] Ventura, D. 2017. How to build a CC system. In *Proc. 8th Int. Conf. on Comp. Creativity*, 253–260.

[Ventura 2023] Ventura, D. 2023. The emperor's new co-author. In *Proc. 14th Int. Conf. on Comp. Creativity*, 55–63.

[Wiggins 2006] Wiggins, G. A. 2006. A preliminary framework for description, analysis and comparison of creative systems. *Knowledge-Based Systems* 19(7):449–458.

[Wilhelm and Ventura 2025] Wilhelm, C., and Ventura, D. 2025. Distilling reinforcement learning into single-batch datasets. In *Proc. 28th European Conf. on AI*, 2074–2081.

[Wu et al. 2023] Wu, Y.; Chen, K.; Zhang, T.; Hui, Y.; Berg-Kirkpatrick, T.; and Dubnov, S. 2023. Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*.

[Xiao and Blat 2013] Xiao, P., and Blat, J. 2013. Generating apt metaphor ideas for pictorial advertisements. In *Proc. 4th Int. Conf. on Comp. Creativity*, 8–15.