Neural Dissection: interpreting and modifying the behavior of deep audio generative models

Pablo Dumenil **Pharoah Jardin** **Aurélien Manté** **Benjamin Quiédeville**

Abstract

In the dynamic landscape of deep learning, neural networks have made remarkable strides, yet their inherent complexity often casts them as enigmatic "black boxes." This investigation marks an advance by extending feature visualization techniques, traditionally employed in image-based models, to GANSynth—a model trained on spectrograms. GANSynth showcases its capability by generating audio waveforms with precision, effectively capturing and reproducing instantaneous frequency variations in a range of synthesized sounds. Departing from the customary emphasis on visual data, feature visualization not only enhances our understanding of auditory representations but also explores interpretability and diversity within the audio domain. While visualization techniques have been developed, their application has been predominantly confined to the image domain until now. Through the optimization of audio spectrograms and iterative adjustments to inputs, this study aims to reveal specific behaviors within GANSynth, showcasing the efficacy of feature visualization in unraveling the intricacies of neural networks. In summary, this investigation paves the way for a deeper understanding of neural networks trained on non-image data, providing a fundamental tool to demystify their internal workings beyond waveform analysis.

1 Introduction

The realm of machine learning has witnessed a remarkable transformation over the past decade, propelled by the groundbreaking advancements in deep learning. Pioneering models like DALL-E [3], GPT-4 [8], and Music-LM [1] have shattered conventional expectations, demonstrating deep learning's ability to generate creative visual content, hold conversations with impressive fluency, and produce quite convincing music.

As deep learning continues to evolve, it is crucial to develop tools that can effectively visualize and interpret the complex internal workings of these models. Feature visualization techniques, such as optimization [10], can provide insights into the model's decision-making process, revealing the patterns and features it identifies in the data. By understanding these patterns, we can gain a deeper appreciation of deep learning's capabilities and limitations, ultimately paving the way for more nuanced and trustworthy applications.

Feature visualization serves as a fundamental aspect of neural network research, primarily within the domain of image-based models. This study marks a significant departure by extending the concept of feature visualization to GANSynth [6], a model specially designed for audio waveform synthesis. Diverging from traditional image-centric models, GANSynth's feature visualization not only enriches our comprehension of auditory representations but also explores the intricacies of interpretability and diversity within the audio domain.

Our methodology involves optimizing audio waveforms starting from random noise to activate specific neurons or channels within GANSynth. Through iterative adjustments to the input, we

Preprint. Under review.

seek to discern the types of inputs that lead to desired behaviors, such as neuron activations or final output behaviors. Various optimization objectives are explored, including neuron activations, channel responses, layer-wise visualization, and class-related logits and probabilities—concepts adapted from image-based feature visualization.

The significance of optimization objectives in generating meaningful visualizations is emphasized. By leveraging optimization, we showcase the capability to create audio examples that induce specific behaviors, offering a potent approach for unraveling the inner workings of our model. Additionally, we investigate the interplay between neurons in GANSynth, acknowledging that combinations of neurons collaborate to represent audio features. Techniques such as interpolation and joint optimization are employed to unveil how neurons interact within the model's activation space, thereby contributing to the exploration of diversity.

Addressing challenges, we tackle the generation of high-frequency artifacts during optimization and propose regularization techniques. These approaches are categorized into weak, strong, and learned prior categories, with methodologies adapted from image-based models to suit the audio domain. Finally, we introduce the concept of preconditioning in optimizing audio waveforms, exploring how transforming the gradient space can expedite descent and potentially serve as a form of regularization, thereby enhancing the quality and interpretability of feature visualizations.

In conclusion, our study extends feature visualization techniques to GANSynth, shedding light on interpretability, diversity, and challenges associated with audio waveform representations. We underscore the potential of feature visualization as a foundational tool for unraveling the complexities of neural networks, providing profound insights into models trained on non-image data.

2 Generative models

Generative neural network models are a type of deep learning architecture that aims to learn the underlying patterns and characteristics of a dataset and use this information to generate new data that is similar to the training data. The most common generative models are variational autoencoders (VAE, that learns a latent representation of the data), diffusion models (that learns to reconstruct the data from a noisy or incomplete representation) and generative adversarial networks (GAN, see subsection 2.1). All three of these types of generative models aim to approach a probability distribution representing the data it is trained on.

2.1 GANs

Generative Adversarial Networks (GANs) are based on a clever interplay between two competing neural networks: the generator and the discriminator. The generator is tasked with producing new data that is indistinguishable from real data. It takes as input a random noise vector and transforms it into a generated sample. The goal of the generator is to fool the discriminator into believing that the generated samples are real. The discriminator is responsible for distinguishing between real and generated data. The goal of the discriminator is to become increasingly accurate at identifying fake samples, forcing the generator to improve its ability to produce realistic data.

The learning process is based on the two-player minimax game [9]:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log 1 - D(G(\boldsymbol{z}))] \tag{1}$$

where $G(\boldsymbol{z}, \theta_g)$ is the differentiable function describing the generator, $\boldsymbol{z}$ is the input noise variables, $p_{\boldsymbol{z}}$ its probability distribution, $D(\boldsymbol{x}, \theta_d)$ the function of the discriminator of parameters θ_d, that outputs a scalar expressing the probability that $\boldsymbol{x}$ comes from p_{data} rather than $G(\boldsymbol{z})$.

As described in [9], the training of the discriminator is performed by ascending its stochastic gradient (with N the batch size):

$$\nabla_{\theta_d} \frac{1}{N} \sum_{i=1}^{N} [\log D(\boldsymbol{x}_i) + \log(1 - D(G(\boldsymbol{z}_i)))] \tag{2}$$

The training of the generator consists in descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{N} \sum_{i=1}^{N} \log(1 - D(G(\boldsymbol{z}_i))) \tag{3}$$

See the subsection 3.1 for our implementation of this double gradient descent.

GANs have been trained on the MNIST dataset [5] since their first implementation [9], so this is the dataset we will also be using to test our GAN architecture (see section 3). Indeed, as the original work developped fully connected architectures for both the generator and the discriminator, we will implement convolutional networks, as suggested by [11].

2.2 Generative models applied to audio synthesis

Generative models applied to audio synthesis, exemplified by WaveNet, RAVE, and DDSP, have significantly contributed to the advancement of artificial intelligence in creating diverse and authentic sounds. WaveNet, as one of the pioneering models, introduced a groundbreaking approach by generating audio samples directly in the time domain, albeit at a relatively slow pace due to its single-sample generation process [12]. RAVE, a hybrid model combining a Variational Autoencoder (VAE) and a Generative Adversarial Network (GAN) [4], demonstrated the capability to synthesize audio in real-time at a high sampling frequency of 44.1 kHz, although it comes with the trade-off of a time-consuming training phase. DDSP, in contrast, offers a unique hybrid model where a neural network controls the parameters of an additive synthesizer and filtered noise [7], resulting in a small-sized model with rapid inference capabilities, albeit limited to harmonic signals.

Furthering the exploration on audio generative models, the GANSynth, detailed in the paper "GANSynth: Adversarial Neural Audio Synthesis" [9], stands out as a cutting-edge model that delves into the potential of Generative Adversarial Networks (GANs) for audio synthesis. The fundamental idea behind GANSynth involves the application of GANs to learn and replicate the complex distribution of audio data, ultimately producing high-quality and diverse audio samples. The architecture of GANSynth comprises a generator and a discriminator working in tandem. The generator is responsible for creating spectrogram representations of audio samples, while the discriminator evaluates the realism of these spectrograms. Through an adversarial training process, GANSynth refines its generator to create increasingly realistic audio samples that can be perceptually indistinguishable from real recordings.

3 GAN implementation

3.1 DCGAN

Deep convolutional GANs have been developped in [11] mainly to benefit from the great capacity of CNNs to extract information from images. To ensure convergence and stability, it is recommended to use batchnorm in both the generator and the discriminator, remove any fully-connected hidden layer, and use LeakyReLU activation functions.

The architecture of the discriminator is composed of two blocks of [Conv2d $\rightarrow$ BatchNorm2d $\rightarrow$ LeakyReLU $\rightarrow$ Dropout] followed by a block of [Flatten $\rightarrow$ Linear (to scalar) $\rightarrow$ Sigmoid].

The architecture of the generator is composed of an initialization step consisting in reshaping with [Linear $\rightarrow$ BatchNorm1d $\rightarrow$ LeakyReLU $\rightarrow$ Reshape], followed by two blocks of [ConvTranspose2d $\rightarrow$ BatchNorm2d $\rightarrow$ LeakyReLU], and one final ConvTranspose2d before a Tanh function giving the output.

As for the training, we used the gradient descent presented in the original GAN paper [9] (eq.2 and 3), which come down to minimizing a *Binary Cross Entropy* loss function:

$$\text{BCELoss}(\boldsymbol{x}, \boldsymbol{y}) = -\frac{1}{N}\sum_{i=1}^{N}[\boldsymbol{y}_i \log \boldsymbol{x}_i + (1-\boldsymbol{y}_i)\log(1-\boldsymbol{x}_i)] \tag{4}$$

The training of the discriminator is split between the training on all-real samples and the training on all-fake samples. For the all-real samples, the output of the discriminator $\tilde{\boldsymbol{x}} = D(\boldsymbol{x})$ if compared to an all-one vector $\tilde{\boldsymbol{y}}^* = \mathbf{1}$. For the all-fake samples, the predicted classes $\hat{\boldsymbol{x}} = D(G(\boldsymbol{z}))$ are compared to the all-zeros vector $\hat{\boldsymbol{y}}^* = \mathbf{0}$. The sum of both participations gives the full loss for the discriminator:

$$\text{DiscLoss}(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{x}}) = -\frac{1}{N}\sum_{i=1}^{N}[\log \tilde{\boldsymbol{x}}_i + \log(1-\hat{\boldsymbol{x}}_i)] \tag{5}$$

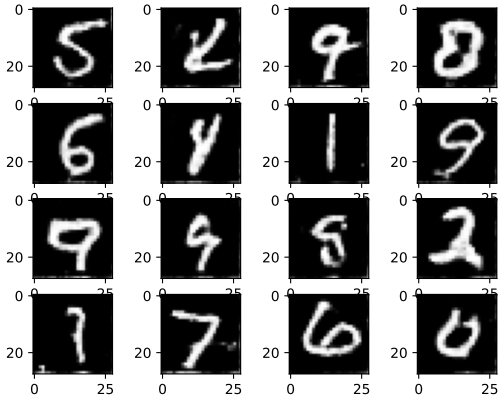


Figure 1: 16 output images from the DCGAN generator from 16 vectors of size 100 of random gaussian noise, after 200 epochs of training with a learning rate of 10^{-4}

which corresponds to the function being differentiated in eq.2.

Following the same principle, we can use the BCELoss function for the generator to define its loss function, by comparing $\hat{\boldsymbol{x}} = D(G(\boldsymbol{z}))$ and $\tilde{\boldsymbol{y}}^* = \mathbf{1}$:

$$\text{GenLoss}(\hat{\boldsymbol{x}}) = -\frac{1}{N}\sum_{i=1}^{N} \log \hat{\boldsymbol{x}}_i \tag{6}$$

As we will minimize GenLoss, this expression is equivalent to the gradient presented in eq.3 for the gradient descent.

This architecture and training process gave us convincing results (see fig.1). At the beginning of the training though, the discriminator outperforms the generator. Balance is reached only after about 50 epochs with a learning rate of 10^{-4}.

3.2 Wasserstein GAN

As the training process of the DCGAN seems quite unstable, especially at the beginning, we implemented a new gradient descent, based on the Wasserstein metrics, as presented in [2]. The main difference here is the creation of a *gradient penalty*, adding to the loss function of the discriminator based on the norm of the gradient (with respect to the input $\boldsymbol{x}$). This lets the network satisfy the K-Lipschitz condition on teh set of functions the optimum is searched in ($||f||_L \leq K$).

With this penalty added, the training of the discriminator is slower than the generator, so the process is repeated 5 times for 1 training step for the generator (as suggested in [2]). The training process is significantly slower than for the DCGAN (due to the 5 iterations on the discriminator and the "manual" computation of the loss), but shows more balanced results after fewer epochs.

The architecture being exactly the same as for the DCGAN, and the training process differing mainly by its stability, it is no surprise that randomly generated samples from the trained generator (fig.2) do not significantly differ from the ones from the DCGAN (fig.1).

4 Feature visualization

In order to understand how these networks generate or classify data, individual hidden neurons, channels or layers may be inspected through feature visualization. This technique consists in the maximization of the activations of a set of neurons. While gradient descent can be used to optimize the weights of a neural network, it can also be used on any (sufficiently regular) optimization problem. Here, a variable input image is considered and a classifier network (whose weights are frozen) is applied to it. The optimization problem is as follows: how can the input image be modified to

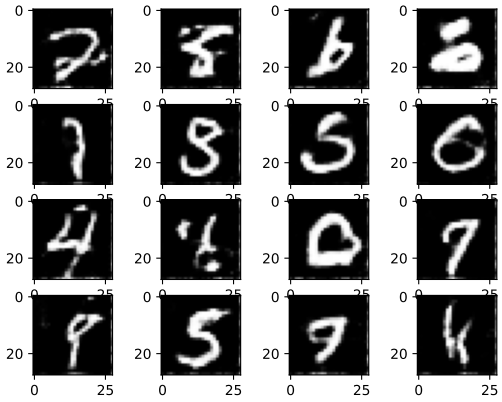


Figure 2: 16 output images from the WGAN generator from 16 vectors of size 100 of random gaussian noise, after 100 epochs of training with a learning rate of 10^{-4}

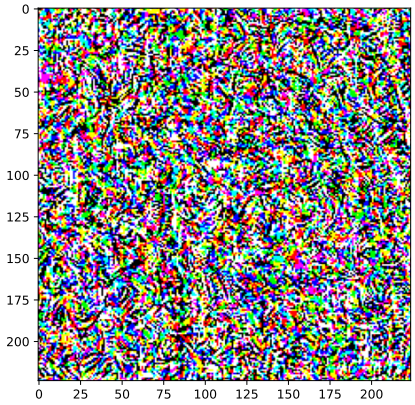


Figure 3: Adversarial example obtained by maximizing the output neuron corresponding to the “grand piano" output neuron in the GoogleNet classifier with no regularization techniques applied

maximize the given objective? The algorithm iteratively converges to a satisfying input image that maximizes the chosen objective.

While the basic concept behind feature visualization is simple, many tricks are required to obtain satisfactory results. If no regularization techniques are used and the objective is maximizing the output of a neuron in the final layer, the optimization results in an *adversarial example*. The resulting image appears to be noise to a human viewer (see fig.3) and yet, inference predicts the desired class with very high confidence. In the case of fig.3, the grand piano class is predicted with a confidence of 100.00%.

On the one hand, this is very useful to understand weaknesses in classifiers: for instance if a small proportion of this adversarial example is blended into an existing image, it may alter its classification dramatically, completely fooling the discriminator which could be dangerous.

On the other hand, such images give no insight into where and how features are detected or synthesized. In order to more closely match the distribution of images the network was trained on, many regularization techniques may be put to use. In the case of image discriminators/classifiers, reducing high frequencies thanks to the use of blurring or denoising filters, such as simple gaussian kernal convolutions, or the use of bilateral filters, help with generation as this helps spatially larger features to appear more easily. These transforms are applied to the optimized input image between each iteration.

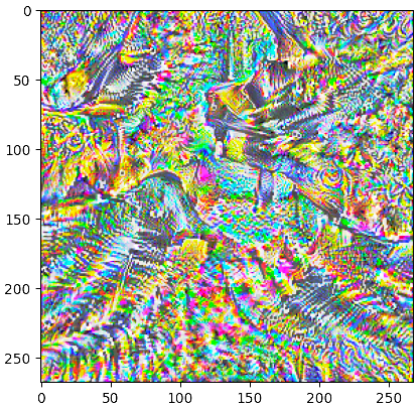


Figure 4: Maximization of the output neuron corresponding to the “grand piano" output neuron in the GoogleNet classifier with regularization techniques applied. A piano-like feature is present in the top of the image. On the bottom left and right, key like features seem to be visible.

Further regularization can be applied by accounting for various expected invariances: for instance, an image that maximizes a certain output class is expected to do so even if it has been subject to small perturbations such as scaling, rotation, translation, as well as (but not necessarily) some color transforms (adjusting contrast, brightness or saturation). This improves general coherence and avoids generating adversarial examples, but this is generally at the cost of a less optimal solution to our optimization objective. This is perfectly acceptable since our true goal is feature visualization and network interpretability—not the underlying objective maximization. With such regularization, we obtained the optimized input for the class "grand piano" presented in fig.4. Changing the objective maximize more shallow layers (i.e. closer to the input), lower level features apear in the input as can be seen in figures 5 and 6[1].

In the case of two-part model such as a GAN, both the generator and discriminator may be dissected. Dissecting the discriminator may shed light on what features it looks for to differentiate a real image from a fake one. These features may be tailored to generator artifact detection, so generating actual numbers (in the case of a model trained of MNIST) using the discriminator may be too much to ask for. It may still be interesting to visualize what the discrimination features look (or sound) like although successfully generating examples that follow the training distribution is unlikely.

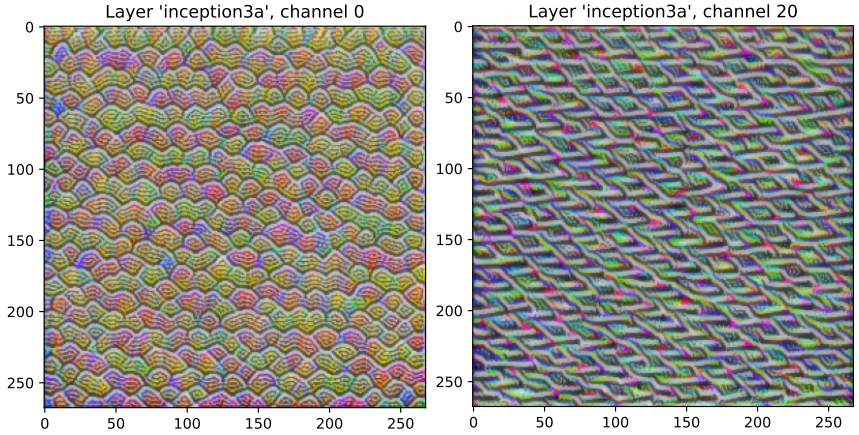
In light of this observation, generation using the GAN generator is more compelling but a new problem arises: the input of the generator—which is what will be optimized during feature visualization—is a latent variable that is not easily interpreted. For instance, the result of maximizing a certain hidden layer objective will yield a latent vector instead of the previously available image input. A simple way of understanding this latent variable is simply to feed it forward through the same network once our objective has been met. This has the added advantage of generating not only an image that lets us understand the role of a set of neurons, but also pairs it with the corresponding latent vector for free, allowing for conjoint latent space interpretation.

5 Discussion: applications to audio

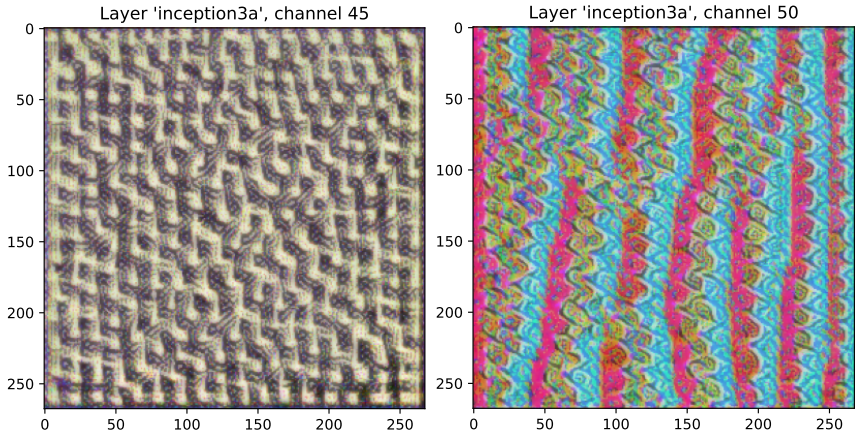
The primary goal of this study was to find ways of transposing these general feature visualization ideas to improve interpretablity of audio-oriented networks such as GANSynth or even RAVE. With limited time available, many ideas have not had the chance to be evalutated but some of them will be briefly explained here.

In the case of discriminator/classifier network dissection, some of the previous image transformations performed between iterations on the input may be used as-is on the audio-oriented networks. If the input is an audio spectrogram, time translations, gain adjustments, time stretching and bilateral filtering could be acceptable transformations to explore, whereas rotations and frequency translations or frequency stretching will break any harmonic coherence or mix non-homogeneous dimensions

[1] so beautiful waw haha waw youpii



(a) Channel 0 (b) Channel 20



(c) Channel 45 (d) Channel 50

Figure 5: Optimized inputs to maximize the activation of some channels of the layer "inception3a" in the GoogleNet classifier, with regularization techniques applied, after 1000 iterations and a learning rate of 10^{-1}

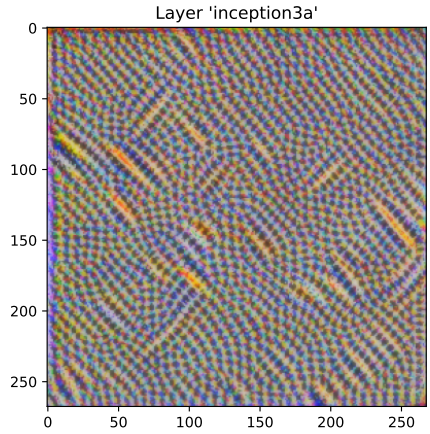


Figure 6: Optimized input to maximize the activation the layer "inception3a" in the GoogleNet classifier, with regularization techniques applied, after 1000 iterations and a learning rate of 10^{-1}

which would probably incur more damage than actual improvements as they would not make physical sense. For example, rotating a spectrum is perhaps not a reasonable expected invariant perturbation with respect to the objective, at least for large angles. In contrast, rotating an object in an image should leave many of the extracted high-level features in place.

As for the case of optimizing a latent variable of generators, adding noise perturbations can help avoid converging to weak local maxima. If the latent space is sufficiently orthogonal, using a form of dropout on the latent vector could help better optimize it along each of its dimension before propagating it forward, although further experiments are required to verify this.

6 Conclusion

This paper explores the application of feature visualization techniques to enhance the interpretability of deep generative models. Unfortunately, we did not have enough time to apply our research to GANSynth or other audio generativ models. However, we suggest to study specific behaviors within GANSynth by optimizing audio spectrograms and iteratively adjusting inputs, offering insights into the model's internal workings. This exploration includes various optimization objectives, addressing challenges like high-frequency artifacts through regularization techniques. Our paper also provides a brief overview of generative models in audio synthesis, highlighting GANSynth as a state-of-the-art model utilizing Generative Adversarial Networks. This work contributes to a deeper understanding of neural networks in the audio domain, fostering advancements in both creative and technical aspects of audio generation.

Our work is available at our GitHub repo which may continue to evolve.

References

[1] Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank. Musiclm: Generating music from text, 2023.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[3] James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2:3, 2023.

[4] Antoine Caillon and Philippe Esling. Rave: A variational autoencoder for fast and high-quality neural audio synthesis, 2021.

[5] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[6] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. Gansynth: Adversarial neural audio synthesis, 2019.

[7] Jesse H. Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts. DDSP: differentiable digital signal processing. *CoRR*, abs/2001.04643, 2020.

[8] OpenAI et al. Gpt-4 technical report, 2023.

[9] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[10] Christopher Olah, Ludwig Schubert, and Alexander Mordvintsev. Feature visualization. *Distill*, 2017.

[11] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.

[12] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016.