Music Playlist Cover Generator: Fine-tuning Diffusion Models on Spotify Playlist Data

Ben Caffee
University of Washington
bncaffee@uw.edu

Andrew Zhang
University of Washington
azhang26@uw.edu

Alexander Wirthlin
University of Washington
awirth03@uw.edu

Abstract

While text-to-image models have become increasingly popular, there have yet to be applications for generating music playlist covers. To tackle this problem statement of finding optimal playlist covers, we employ various diffusion models and prompt engineering to generate a playlist cover that encapsulates the general themes of an image's corresponding prompt. We also curate a dataset of approximately 87,000 playlist covers and their respective prompts gathered by scraping Spotify. Focusing on the Stable Diffusion and DeepFloyd IF models, which utilize state-of-the-art diffusion architectures, we compare the FID scores and output images of pre-trained models to fine-tuned ones on our playlist cover dataset. We find that our fine-tuned small Stable Diffusion model achieves the best FID score compared to the pre-trained small Stable Diffusion, Stable Diffusion v1-5, and DeepFloyd IF models, and that all models were able to capture the main characteristics from a prompt. Furthermore, we discuss the capabilities of diffusion models in generating legible text within images and, particularly for larger ones, how their higher resolution produces more appealing visuals.

1. Introduction

Music apps such as Spotify and Apple Music are hugely popular. Their primary appeal is customizable features, such as creating music playlists personalized with songs and a cover image. However, choosing a cover image to represent your playlist can be annoying. People typically rely on their miscellaneous photos or a quick search on Google Images, but finding a single existing image to embody a complex set of songs is a challenging task. There is currently no direct way to automatically generate a unique (and non-deterministic) playlist cover from the data in a playlist. The closest thing to this is Spotify's method of forming a 2x2 grid containing the playlist's first four songs' distinct album covers or, if there are only three albums, the first song's album cover.

Although many people have used text-to-image generators to create artwork, including for music-related items, only in the past few months have people realized the full potential of these models through prompt engineering. However, this can be a very arduous process as prompt engineering isn't accessible to everyone despite an increase in publicly available dedicated guidebooks, and it simply can take a lot of time. Additionally, the playlist cover problem generalizes to the issue of producing an image from text aggregates. Other tasks include generating social media profile pictures based on a person's bio or YouTube thumbnails from the video title and description. By determining a successful method for creating effective playlist covers, we may find it easier to tackle these other problems.

We compare different text-to-image models on their abilities to generate a playlist cover from a prompt containing a playlist's core attributes[1]. Specific features would be more appropriately represented in the playlist's cover image depending on the type of playlist. For example, a workout playlist consisting of hip-hop songs could be described as more "hype," a generated image could feature brighter red color themes with gym equipment. A late-night drive R&B playlist cover could depict darker blue tones with roads or landscapes.

In addition, we propose that fine-tuning these models on a dataset comprised of actual playlist covers and associated prompts may yield improved outputs compared to without fine-tuning, and model architectures can affect the quality of the images produced.

2. Related Work

Text-to-image generative models have seen much recent success both in research and industry. With natural language processing and computer vision evolving rapidly, combining techniques from both fields has created models that take in a custom text prompt and generate an image

[1]The code we used to analyze our models is available at https://github.com/bcaffee/Playlist_Cover_Generator

representing the prompt. One of the most popular models was DALL-E [12], released by OpenAI in January 2021. DALL-E uses an autoregressive model combined with a Transformer model [18], particularly GPT-3 [1], as well as Generative Adversarial Network (GAN) [2]. DALL-E can handle various prompts and was one of the earliest successful models to utilize zero-shot learning (i.e., it could produce output without requiring additional training). The Transformer model is responsible for understanding the text input and creating an abstract representation, and then the GAN generates the corresponding image based on that representation. The more recent DALL-E 2 [11], released in April 2022, explicitly utilizes the diffusion model architecture instead of a GAN for the image creation phase. Diffusion models consist of two main steps: the forward diffusion process adds noise to the input data, and the parameterized reverse process denoises the results of forward diffusion to generate new data to recover the original data. These models have rendered more realistic and detailed images compared to GANs, hence their rise in use.

Stable Diffusion [13], open-sourced in August 2022, is another text-to-image model popularized lately. It also uses the diffusion architecture, which performs the image generation process as a series of denoising autoencoders. However, Stable Diffusion performs diffusion in a lower dimensional latent space than the original pixel space, allowing for fewer computation resources while retaining effectiveness and quality. Stable Diffusion employs the variational autoencoder [6] for dimensional latent space reduction during forward diffusion. A U-Net [14] reverses forward diffusion by denoising the data. Finally, a text encoder (in this case, a CLIP ViT-L/14 [9]) is used for text processing. Stable Diffusion is pre-trained on the LAION 5B dataset [16]. Due to being relatively lightweight yet still powerful, many other models or applications build off Stable Diffusion.

DeepFloyd IF [17] is a recent state-of-the-art open-source text-to-image model that excels in embedding text into images. It merges multiple independent diffusion networks into a single network. A T5 Transformer [10] feeds into a base diffusion model that generates a lower-resolution image based on the text input. The output gets fed into two super-resolution models that sequentially upscale the image. Like Stable Diffusion, DeepFloyd IF incorporates the U-Net architecture and cross-attention with attention pooling. DeepFloyd IF is primarily pre-trained on the LAION A dataset, a subset of high quality images from LAION 5B. DeepFloyd IF is the largest diffusion model in terms of the number of parameters and outperforms all other leading text-to-image models, such as DALL-E 2, in terms of zero-shot FID score.

While text-to-image models have seen much success, there have been few applications for more specific tasks, such as playlist cover generation. Many applications utilizing text-to-image for digital art and design, such as Photosonic and Canva, merely serve as a playground for users, requiring them to input a text prompt and allowing them to customize the resulting generated image. Existing playlist cover generation applications, like Spotlistr and Coverify, only allow users to customize playlists by adding text boxes and changing color themes but still require users to submit an input image. To our knowledge, no such product exists that automatically synthesizes a given playlist into a prompt and generates an appropriate playlist cover image from that prompt.

3. Methods

3.1. Data Collection

We used the Spotify API to gather playlist data. Specifically, we implemented a scraper that obtained the playlist cover image along with the title, top artist, and top genre within the playlist. We defined nearly 70 unique keywords related to music genres and themes, such as ”rock” and ”nostalgic,” which helped the scraper find a variety of playlists. This made up around 40% of the dataset. The other 60% was obtained through random letter prefix searching to get a wider data spread.

There was a lot of noise in the public playlist data. For instance, as previously described, Spotify auto-generates playlist covers when users don’t upload one. We had to filter these playlists out since these covers contain related album covers which aren’t expressive of the entire playlist’s characteristics. Likewise, we also filtered out official Spotify playlists that followed a particular template containing uninteresting and unrepresentative images, such as simple colors and shapes but no concrete visuals. However, most of the noise we couldn’t filter out came from images that had nothing to do with a playlist’s themes. This set included everything from selfies to random objects to digital art designs. Since we couldn’t remove this noise, we had to settle with the fact that a substantial percentage, but perhaps not majority, of user-added covers were indeed not representative of the actual playlist content.

One big hurdle was getting rate limited by the API, which severely slowed down the scraping process. We attempted a couple of workarounds which, despite eventually working, resulted in less data overall. Finally, we had to deal with some playlists missing data and multiple edge cases that required further parsing.

Regarding prompt engineering, we first considered including the title and the most frequent artist and genre of the songs. However, we found that diffusion models had difficulties with generating accurate faces. This would not be ideal as famous artists should be immediately recognizable on a playlist cover, and disfigured faces would detract from the overall image. Also, lesser known artists wouldn’t

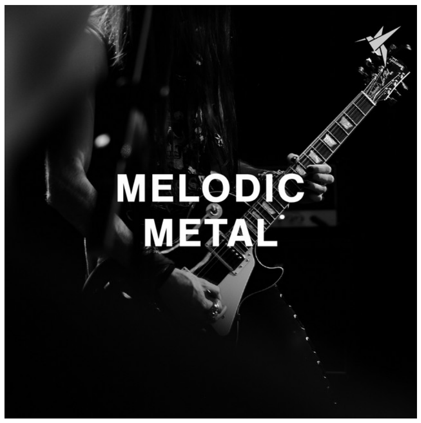


Figure 1. A sample playlist cover with the prompt: ”Melodic Metal - 100 Best Melodic Metal Songs playlist cover image”

be rendered correctly. As a result, to limit this potential behavior, we decided to include only the playlist’s title in the prompt. This heuristic should fundamentally encapsulate the playlists as they often already include the primary genre, theme, or even artists. Additionally, we added the phrase ”playlist cover” to each prompt to guide the diffusion models to generate images in the style of playlist covers. See Figure 1 for an example of a real cover image and its associated prompt.

We also note that any text embedded in playlist covers is advantageous to the models because numerous successful playlist covers include text that display the primary genre or a common artist’s name, as seen in Figure 1 as well.

In total, our data set includes around 87,000 prompt-image pairs.

3.2. Evaluation Procedure

We performed a 80/10/10 train, validation, and test split when analyzing these different approaches so that each model could be consistently assessed.

For evaluation, we calculate the FID scores of the generated images from each model against the test set. For hyperparameter-tuning, this was calculated on the validation set. The FID score [4] measures the distance between real and constructed images’ feature vectors. Lower scores, which are desired, demonstrate closer distances between the real and generated images. This evaluation methodology also allows us to check the quality of the ideas generated by the models with authentic images, and assess how much the output incorporated the input prompt. Moreover, we can subjectively evaluate models by seeing which one produces the most visually pleasing results.

3.3. Small Stable Diffusion Baseline

Our baseline model was Small Stable Diffusion [8], initialized from Stable Diffusion v1-4 [13], due to its fast inference speed on GPU and CPU and low computational requirements compared to other text-to-image models. We downloaded this model from Hugging Face into a Google Colab notebook. Using the provided GPU from Google Colab, we were able to input prompts from our test set into the model and get an output image, which was of resolution 256×256. We evaluated this baseline model against three alternative approaches.

3.4. Small Stable Diffusion Fine-Tuned

The first approach was fine-tuning Small Stable Diffusion on our training set. Since our goal is solely to generate playlist covers, having a specialized dataset of playlist images may help the model perform stronger in this area. LAION is not very representative of playlist covers, since it mainly consists of everyday images such as pets and nature. Furthermore, it seems that patterns exist based on the type of playlist and their respective covers. For example, jazz playlists generally feature instruments such as saxophones or trumpets and contain orange color hues. As such, using the playlist dataset might make the Stable Diffusion model better learn these styles and create more accurate images. To fine-tune Small Stable Diffusion, we used an Accelerate [3] environment within our notebook. Accelerate assists in abstracting away much of the boilerplate code for training loops in PyTorch models, simplifying the fine-tuning process.

Since training requires a lot of computational power, we utilized a Google Cloud VM–specifically a V100 GPU. After experimenting with memory constraints, we found that setting the resolution to 256×256, from the default of 512 $\times$ 512, significantly increased training speed speed while output images were still of sufficient quality. We reduced to batch size to 1 from the default of 32 also to increase speed and reduce memory usage. Other hyperparameters we used from the beginning were a default learning rate of 1e-5, a constant learning rate scheduler, zero learning rate warm-up steps, and a single gradient accumulation step. We trained the model on this configuration for one epoch, i.e. approximately 70,000 iterations, and a batch size of 1. Initially, on visual inspection, the output images from a sample of the validation set were quite noisy containing just static and no discernible visuals and the FID score on validation set was much too large. Thus, we tweaked the learning rate to 1e-8 and retrained, keeping the rest of the configuration as is. This improved the output images from the validation set; however we still noticed that noise was present, so we brought the learning rate down to 1e-10, which proved enough.

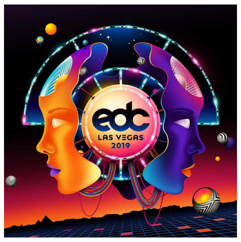
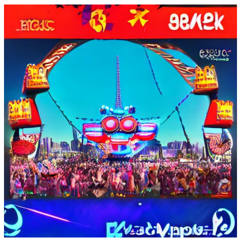
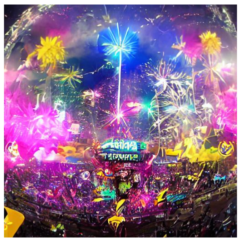
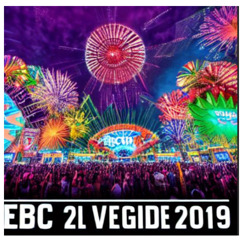
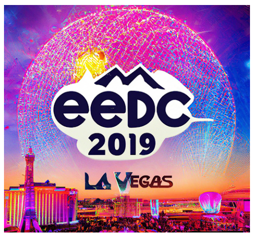
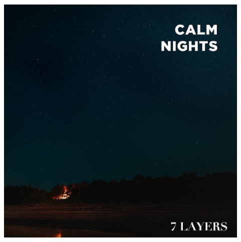
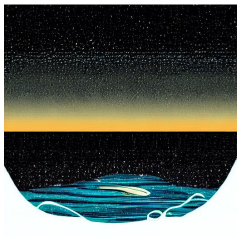
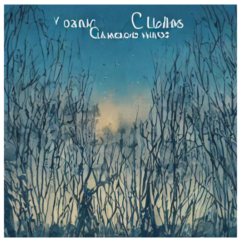
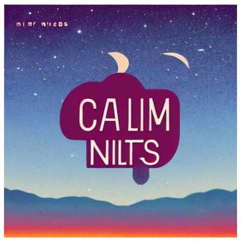
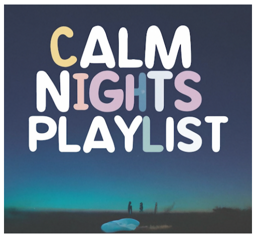
Prompt	True Image	Small Stable Diffusion	Small Stable Diffusion Fine-Tuned	Stable Diffusion	DeepFloyd IF
"EDC 2019 - Las Vegas playlist cover image"					
"Calm Nights playlist cover image"					

Table 1. Prompts along with their true image and generated images from different models

Model	FID Score
Small Stable Diffusion	62.72
Small Stable Diffusion Fine-Tuned	**60.01**
Stable Diffusion	68.42
DeepFloyd IF	104.60

Table 2. FID scores for evaluated models on the test set

3.5. Stable Diffusion

The second approach utilizes Stable Diffusion v1-5, which is a bigger model fine-tuned on more data from LAION as compared to Stable Diffusion v1-4 and Small Stable Diffusion. This approach will demonstrate if different fine-tuned versions of Stable Diffusion on the same dataset affects the quality of the results. Additionally, this will reveal if there are noticeable differences with Small Stable Diffusion, where optimizations were used to decrease inference time, and raw Stable Diffusion models. We downloaded Stable Diffusion v1-5 from Hugging Face into our notebook. Again, due to the high computational costs required for running Stable Diffusion v1-5, we set up a Google Cloud VM with V100 GPU. Similar to the baseline model, we ran each prompt in our test set through Stable Diffusion v1-5 and stored the outputs to calculate the FID score. Stable Diffusion generates 512 $\times$ 512 images, which is a higher resolution compared to Small Stable Diffusion and our fine-tuned model.

3.6. DeepFloyd IF

The final approach utilizes DeepFloyd IF due to its different architecture and high performance in generating words within images. We noticed many playlist covers included words, such as names of artists or genres. Christmas playlists for instance tended to have the phrase "Happy Holidays" or some variation of this expression. This means DeepFloyd IF may generate more appropriate playlist covers than Stable Diffusion, which struggles to embed text into images. We downloaded DeepFloyd IF from Hugging Face into our notebook to process the test set. Unfortunately, DeepFloyd IF's architecture of several diffusion models linked together meant there was a high memory cost that exceeded the available RAM provided by Google Colab. Therefore, we implemented a workaround where we only loaded in one diffusion model at once. When all data was processed by this model and saved, we deleted it from the notebook and downloaded the next model to pass in the saved data. The high computation cost also forced us to try a Google Cloud VM. Unfortunately, we encountered similar issues and in all were only able to run a quarter of the test set prompts. We used the XL version of DeepFloyd IF, using which we able to generate 256 $\times$ 256 images. However, the final scaling-up resolution stage to generate 1024 $\times$ 1024 images for the entire test set was infeasible given our time and resources. As such, we generated a couple 1024 $\times$ 1024 images for reference, but evaluated the FID score on the 256 $\times$ 256 images.

4. Experiment Results

4.1. Small Stable Diffusion Baseline

We ran our baseline Small Stable Diffusion model on the test set of 8,700 image-prompt pairs. As each image took a couple seconds to generate, the entire process took several hours. During this process, we noticed that several prompts led Small Stable Diffusion to produce images that contained inappropriate or NSFW content. To handle this issue, Small Stable Diffusion instead outputted a completely black image. Unfortunately, this placeholder data would lead to undesired results and possibly skew our FID score, as a black square would likely contribute a high dissimilarity with the true image. Thus, we filtered out any completely black images from our output that resulted from NSFW content. We kept this policy consistent for our other models as well. In Table 1, we demonstrate some sample results. We can see that generally, the prompts perform well in allowing Small Stable Diffusion to recognize the overall theme of the playlist. For example, the ”EDC 2019 - Las Vegas” prompt, which represents a playlist for the Electric Daisy Carnival (EDC) electronic dance music festival in Las Vegas, allowed the model to recognize the resulting image should display some sense of a festival. Although, the generated text was not very legible. For the ”Calm Nights” prompt, the model was able to output an image representing a calming beach night-time scene successfully, but no text was generated.

The FID score between the generated images and the true images was 62.72 as seen from Table 2.

4.2. Small Stable Diffusion Fine-Tuned

We ran our fine-tuned Stable Diffusion model on the test set as well which took a similar amount of time. In Table 1, we can see some sample results. In general, this model's outputs were more noisy compared to other models, due to the smaller dataset size and existence of noisy data. However, the output images still captured similar themes of the true image. For instance, the ”EDC 2019 - Las Vegas” prompt led to an image containing festivals and fireworks, which were expected. The ”Calm Nights” prompt generated a more dusk-like setting, but still contained nature imagery to capture the ”calm” aspect. Text generation, like Small Stable Diffusion, was not very legible.

From Table 2, the FID score was 60.01. This score is lower than our baseline model, meaning we have successfully achieved a better performing model than Small Stable Diffusion in terms of FID score.

4.3. Stable Diffusion

Running Stable Diffusion similarly took several hours on the test set. Again in Table 1, we can see sample results. With the ”EDC 2019 - Las Vegas” prompt, Stable Diffusion similarly generated a festival setting with fireworks, although the output was more clear than the fine-tuned Small Stable Diffusion result. The ”Calm Nights” prompt led to a night-time setting with the moon, which was ideal. A major improvement of Stable Diffusion to the previous models was the legibility of text generation. It can clearly be read that ”EBC 2L VEGIDE 2019” and ”CALIM NILTS” were generated in the images. However, spelling was still a struggle for this model.

From Table 2, the FID score was 68.421, which is higher than both the baseline and fine-tuned models.

4.4. DeepFloyd IF Baseline

Due to DeepFloyd IF's architecture and size, trying to analyze this model on the test set took the longest by far. DeepFloyd IF consists of three different diffusion models that continuously upscale the resolution of an generated image. Since each model took several seconds to produce an output, trying to analyze the entire model on thousands of images took numerous hours. Further, the memory restrictions on both Google Colab and Google Cloud Computing VM meant it was not possible to have the entire DeepFloyd IF model loaded in an once. Therefore, manual downloading and deletion of diffusion models was required. We were able to run DeepFloyd IF on a few thousand images, but were only able to generate to the second stage, which brings an image to a 256×256 resolution whereas the final diffusion layer brings the image to 1024×1024. From the sample results in Table 1, it is clear that DeepFloyd IF possesses the best text generation compared to previous models, with ”EEDC 2019 LA VEGAS” and ”CALM NIGHTS PLAYLIST” being the closest to the input prompts. Moreover, the images were still able to capture core themes from the prompt and true images, such as vibrant colors, the Las Vegas skyline and fireworks, and again a nighttime nature setting.

The FID score was 104.60 from Table 2, which is significantly larger than the previously tested models. We attribute this to possibly being only able to generate images for about a quarter of the test, and only being able to achieve the second highest resolution possible. With more access to GPU and memory, we imagine this score could have been lower and closer to the values attained by the other evaluated models.

5. Discussion

We have found that fine-tuning a Stable Diffusion model achieved the best FID score compared to the other approaches we considered. This means that models trained on datasets specific to playlist covers will generally generate images that resemble the true playlist cover for a given prompt compared to larger and more complex models that were pre-trained on other datasets.

However, the FID scores we achieved are poor compared to the FID scores of state-of-the-art text-to-image models on datasets such as COCO [7]. For example, DeepFloyd IF achieved a FID score of 6.66 on COCO. We believe the poor FID scores stem from our task being incredibly specific to playlist covers and there likely is little data for playlist covers on which the dataset that diffusion models were trained on. Plus, there is a lot of noise in our dataset in that a cover could simply be an abstract portrayal of the playlist title or even unrelated to the title at all. And as we know, data is everything, so in a perfect world, our dataset would consist of a balance between abstract visuals and more images with cover-type structures, giving the model a chance to learn both how to represent the high level ideas from a prompt and also how to fit that within the concept of playlist cover. As seen for the "EDC 2019 - Las Vegas" prompt, the actual playlist cover contained abstract drawings of faces, which are not visuals attributed to Las Vegas nor EDC. Therefore, it would be difficult for models to generate images that contain the exact same or similar visuals as the actual playlist cover. Additionally, Stable Diffusion and DeepFloyd IF's ability to generate very detailed images could lead to lower FID scores, as more content within images could result in greater differences from the true image especially if the resulting image contains the wrong content. As a result, we treat the FID score with caution as being the only metric to evaluate the performance of our models on.

Visually, we can see that Stable Diffusion and DeepFloyd IF perform text generation much better than Small Stable Diffusion and our fine-tuned model. It is possible that Small Stable Diffusion's optimization in inference time could negatively impact the ability of the model to generate text. As for our fine-tuned model, a lack of text within existing playlist covers meant the model likely could not generalize to text very well. Further, Stable Diffusion was generate much higher quality images around 512 $\times$y 512, whereas Small Stable Diffusion was only able to generate 256 $\times$ 256 images. Thus, we find that larger and more complex diffusion models can generate text and details with higher quality, leading to more visually pleasing images but potentially lower FID scores.

5.1. Conclusion

We present a custom dataset of Spotify playlist covers and prompts as well as evaluate multiple diffusion models on this dataset as an effective way of generating image covers for a playlist. We also fine-tune a smaller diffusion model on our dataset and find that it is successful in beating the baseline model in terms of FID score.

We also find that FID scores on the test set in itself may not be the most accurate metric to evaluate a model's performance for our specific task. Larger and more complex diffusion models are able to generate text and details with higher accuracy and quality, resulting in images that may stray from the true image yet still be a valid playlist cover.

With more data, time, and computational resources, we are confident that we would be able to generate more pleasant images and possibly decrease the FID scores attained.

5.2. Future work

Overall, we hope to have achieved meaningful results that could see use by music listeners. Our findings could eventually culminate in an app or extension that gives users the opportunity to input a playlist and choose which playlist features they would want to provide in the prompt to generate the most appropriate playlist cover. Future work can aim to build a better-cleaner dataset of playlist covers, experiment with using other model architectures, and evaluation metrics. In addition, future work relating to our project, could utilize low-rank adaptation [5], which has gained momentum over the last couple of years for large language models, as an efficient way to help quickly fine-tune diffusion models [15], thus significantly saving compute time.

Another key thing to explore would be the structure of the prompts. For instance, a prompt, in addition to the title of a playlist, could include words like the top genre and artist. An example prompt would be "R&B late night drive playlist image cover with text that reads 'R&B'." For DeepFloyd IF especially, this could work very well and is something with which we wish we had more time to experiment. The prompt itself, in terms of what features to include, could also be learned by a model rather than being defined by the implementer.

Lastly, we hope our method of constructing prompts from text aggregates and feeding them into text-to-image models could evolve and generalize to solutions for more problems related to thematic image generation, such as for video thumbnails based on a description and title or profile pictures from a person's bio.

References

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. 2

[2] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. 2

[3] Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, and Sourab Mangrulkar. Accelerate: Training and inference at scale made simple, efficient and adaptable. `https://github.com/huggingface/accelerate`, 2022. 3

[4] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018. 3

[5] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. 6

[6] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. 2

[7] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014. cite arxiv:1405.0312Comment: 1) updated annotation pipeline description and figures; 2) added new section describing datasets splits; 3) updated author list. 6

[8] Chengqiang Lu, Jianwei Zhang, Yunfei Chu, Zhengyu Chen, Jingren Zhou, Fei Wu, Haiqing Chen, and Hongxia Yang. Knowledge distillation of transformer-based language models revisited. *ArXiv*, abs/2206.14366, 2022. 3

[9] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 18–24 Jul 2021. 2

[10] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. 2

[11] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *ArXiv*, abs/2204.06125, 2022. 2

[12] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8821–8831. PMLR, 18–24 Jul 2021. 2

[13] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022. 2, 3

[14] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. cite arxiv:1505.04597Comment: conditionally accepted at MICCAI 2015. 2

[15] Simo Ryu. Using low-rank adaptation to quickly fine-tune diffusion models. `https://github.com/cloneofsimo/lora`, 2023. 6

[16] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. Laion-5b: An open large-scale dataset for training next generation image-text models, 2022. 2

[17] Alex Shonenkov, Misha Konstantinov, Daria Bakshandaeva, Christoph Schuhmann, Ksenia Ivanova, and Nadiia Klokova. If by deepfloyd lab at stabilityai. `https://github.com/deep-floyd/IF`, 2023. 2

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 2