EasyChair Preprint

№ 9782

Modeling Graphene Extraction Process Using Generative Diffusion Models

Modestas Grazys

February 26, 2023

Modeling graphene extraction process using generative diffusion models

No Author Given

No Institute Given

Abstract. Graphene, a two-dimensional material composed of carbon atoms arranged in a hexagonal lattice, possesses a unique array of properties that make it a highly sought-after material for a wide range of applications. Its extraction process, a chemical reaction's result is represented as an image which shows areas of synthesized material. Knowing initial conditions (oxidizer) the synthesis result could be modeled by generating possible visual outcome. A novel text2image pipeline to generate experimental images from chemical oxidizers are proposed. Key components of such pipeline are a textual input encoder and a conditional generative model. In this work the capabilities of certain text model and generative diffusion model are investigated and some conclusions are drawn providing further suggestions for further full text2image pipeline development.

1 Introduction

Graphene is an exotic wonder material with many advantages. It is the thinnest and strongest measured material in the universe. Graphene can maintain 10^6 times higher current density than copper, has record thermal conductivity and stiffness, is impermeable to gases, and combines contradictory properties such as brittleness and ductility [6]. It is a one-atom-thick material - a plane made up of carbon atoms lined up in a hexagonal lattice, resembling a beehive. Graphene has a very wide range of applications: manufacture of transistors [1]; production of conductive plates with single-electron-transistor (SET) circuitry; digital memory for quantum repeaters [27]; plasma displays [23] and much more.

Graphene could be synthesized using methods described in [21, 20]. Yet, development of a digital tool capable of visualizing the potential output of a graphene synthesis reaction using a selected oxidant could give an insight about expected result beforehand. Instead of complext chemical process modelling, such process could be modelled by analyzing images. The mathematical model for image encoding in our case is a neural network, which uses the process of automatic conversion of text into an image (*text2image* [19]), like Stable Diffusion or DALL-E 2 [22, 18]. Model input - chemical name (text) of the oxidant used in graphene synthesis. Then the output of the model - a visual representation (picture) of the result of a possible reaction. However, this will be discussed in more detail in a later work, in the future. And in the current context, more attention is paid to the separate components of the model architecture: the component that

prepares the input text embeddings that condition the output and the generative component that provides the visual result of the synthesis simulation (diffusion model selected).

For the component which prepares the input text embeddings (text encoder) the Contrastive Language-Image Pre-training model (CLIP) was selected [17]. The CLIP model can be conceptualized as a conventional image classifier with the ability to learn from the natural textual description of a photo. The usage of this model does not need complex and limiting data labeling methodologies. Also, the CLIP model can extract latent text embeddings, which can later influence the generative diffusion model in the process of conditional image generation. The simulation tool for the graphene synthesis reaction described in this paper specifically requires the text encoder component of the CLIP model for the latent text representations (text embeddings), precisely because of the influence on the diffusion model. The challenge here is to adapt appropriately this model to classify images of graphene synthesis correctly. As the images among oxidant classes look similarly and the amount of data available is very limiting currently.

For the generative component a diffusion model [24, 12] was selected. In this context, it is a generative neural network, like generative adversarial networks (*GANs*) [7] or variational autoencoders (*VAEs*) [4] capable of generating images from the input vector by applying a denoising process. The text latents encoded by the CLIP text encoder can influence the input vector of the diffusion model, thus conditioning the output of the diffusion model. In this way, potential imaging results of the graphene synthesis reaction can be generated conditioned by the entered name of the oxidant. However, this paper presents the capabilities of the diffusion model to generate imaging results for the graphene synthesis reaction by discarding this conditional input application and leaves this task for the future. The challenge here is to inspect the capability of a diffuser to generate graphene synthesis images and draw further conclusions about usage of a conditional diffuser for the general development objective.

The novelty of this work is a proposed usage of CLIP for digital components required for the subsequent modeling of the graphene synthesis simulation tool. This proposed solution allows to create a digital tool that can visually represent the possible output of the graphene synthesis reaction using the chosen oxidant. The tasks of the work include investigating the applicability of the CLIP model, the training process, and the ability of the diffusion-based generator (diffuser) to unconditionally generate images matching graphene synthesis reaction results.

2 Methodology

2.1 CLIP

The CLIP model [17] was introduced by the OpenAI company and is able to learn to predict which caption matches which image from the natural textual description of the photo. In this way, the limiting usability of computer vision systems is avoided, since conventional systems of this type require additional labeling information for any other visual concept. In implement experimentation

the CLIP model is trained by providing (photo, text) data pairs, where text corresponds to the oxidant which was used to synthesize the result seen in the given photo. Conventional image classifiers simultaneously train a visual feature extractor and a linear classifier to predict the class label, while the CLIP model simultaneously trains a visual feature encoder and a text encoder to predict suitable (photo, text) pairs. Encoded oxidizer namings (latents) during training are encouraged to match synthesis result image latents as much as possible by minimizing pairs of their the dot products. The minimized values locate on the main diagonal of this resulting matrix. The used architecture in our experimentation can be seen in Fig. 1.

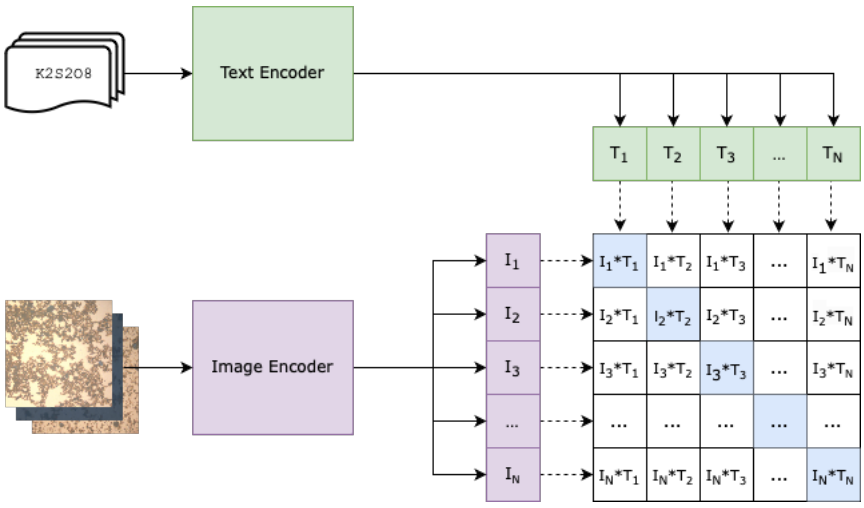

Fig. 1. CLIP training

During inference the text encoder simulates a zero-shot linear classifier by embedding the oxidizer class names of the graphene images dataset and classifies the graphene synthesis results by their corresponding oxidizers. The image latent vector is scalar multiplied with each of the possible naming latent vectors and the smallest value of all obtained is selected. Then an oxidizer naming, which latent vector provided this smallest value in product with image latent, is selected and passed as an output. This process is illustrated by Fig. 2.

Due to the property of learning from the natural textual description of an image, the CLIP model requires less data to achieve the same accuracy result compared to conventional computer vision classification models. These require a larger volume of training material corresponding to the distribution of use. This is relevant as the available amount of graphene data is very limited.

The CLIP model and its ability to embed textual input is also used in models for generating images from text: DALL-E 2, Stable Diffusion [18, 22]. It is this use case that is relevant to the purpose of this paper. Other use cases of the

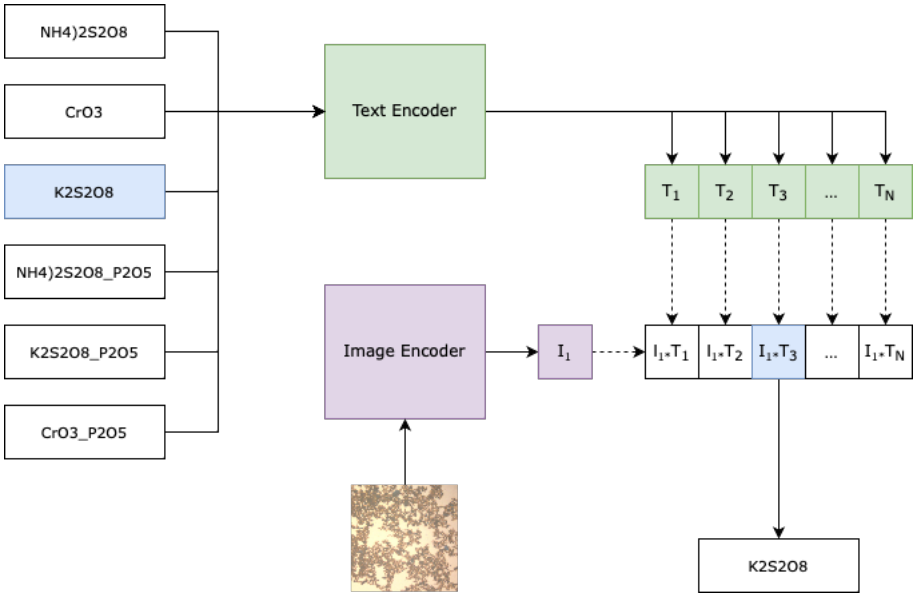

Fig. 2. CLIP inference

CLIP model: content moderation [5]; image captioning [2]; image search engine [14]; and much more.

2.2 Diffusion model

The invention of the diffusion model was inspired by the idea of non-equilibrium thermodynamics: to systematically and slowly destroy the data structure through an iterative process of diffusion. The reverse diffusion process can then be learned by reconstructing the previously destroyed data structure - this method creates an extremely flexible and easy-to-manage generative model [24].

The diffusion process (forward process) is a Markov chain [15] and is performed by gradually adding Gaussian noise to the image data based on the noise variance schedule (*variance schedule*) $\beta_1, \beta_2, ..., \beta_T$, where T is the duration of the schedule (how many times noise will be added to the data). Fig. 3 diffusion process is marked $q(x_t|x_{t-1})$, where $x_0 \sim q(x_0)$ - initial data; $x_1, x_2, ..., x_{t-1}, x_t, ..., x_T$ - latents (data with applied Gaussian noise).

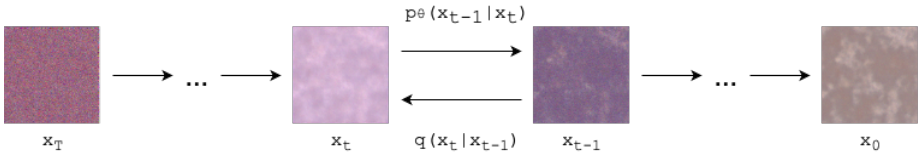

Fig. 3. Diffusion process - Markov chain

Mathematically, the diffusion process $q(x_{1:T}|x_0)$ is written [12]:

$$q(x_{1:T}|x_0) := \prod_{t=1}^{T} q(x_t|x_{t-1}), \qquad q(x_t|x_{t-1}) := \mathcal{N}(x_t|\sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I}) \tag{1}$$

The reverse process Fig. 3 is denoted by $p_\theta(x_{t-1}|x_t)$. Is run to reconstruct image data destroyed by noise and is described by a Markov chain with learned Gaussian transitions starting from $p(x_T) = \mathcal{N}(x_T|\mathbf{0}, \mathbf{I})$:

$$p_\theta(x_{0:T}) := p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t), \qquad p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}|\boldsymbol{\mu}_\theta(x_t,t), \boldsymbol{\Sigma}_\theta(x_t,t)) \tag{2}$$

We can write a closed-form expression for the loss function [12, 13]:

$$L = c + \sum_{t=1}^{T} \kappa_t \mathbb{E}_{x_0,\epsilon}\left|\left|\epsilon - \epsilon_\theta(\sqrt{\prod_{s=1}^{t}(1-\beta_t)}\cdot x_0 + \sqrt{1-\prod_{s=1}^{t}(1-\beta_t)}\cdot\epsilon, t)\right|\right|_2^2 \tag{3}$$

Here, c and κ_t are constants independent of θ. $\kappa_t = \frac{\beta_t}{2(1-\beta_t)(1-\prod_{s=1}^{t}(1-\beta_{t-1}))}$ when $t > 1$, but $\kappa_1 = \frac{1}{2(1-\beta_1)}$. Also, $\epsilon_\theta : \mathbb{R}^L \times \mathbb{N} \to \mathbb{R}^L$ is a neural network that takes a latent variable x_t and a diffusion step t as the input

Below are the pseudocodes for the diffusion model training and inference (sampling) algorithms based on all the above information.

Algorithm 1 Training

repeat
  $x_0 \sim q(x_0)$
  $t \sim Uniform(\{1, ..., T\})$
  $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  Gradient descent step:
  $\nabla_\theta\left|\left|(\epsilon - \epsilon_\theta(\sqrt{\prod_{s=1}^{t}(1-\beta_t)}\cdot x_0 + \sqrt{1-\prod_{s=1}^{t}(1-\beta_t)}\cdot\epsilon, t)\right|\right|^2$
until converged

Algorithm 2 Sampling

$x_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, ..., 1$ **do**
  $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $z = \mathbf{0}$
  $x_{t-1} = \frac{1}{\sqrt{(1-\beta_t)}}(x_t - \frac{\beta_t}{\sqrt{1-\prod_{s=1}^{t}(1-\beta_t)}}\epsilon_\theta(x_t, t)) + \sigma_t z$
end for
return x_0

During inference, a noise latent is passed through the diffuser and an image representing graphene synthesis result is expected to appear.

To solve this work tasks, it is enough to realize that the diffusion model can generate images of possible graphene synthesis results, so this model was chosen for this very reason. Using the cross-attention layers in the model architecture can influence the model's generative processes, thus, through the input, conditioning the model's output. And this is relevant for the general goal of the work, which will be worked on in the future. This work is limited to research to determine the capabilities of the diffusion model to unconditionally generate images of possible graphene synthesis results.

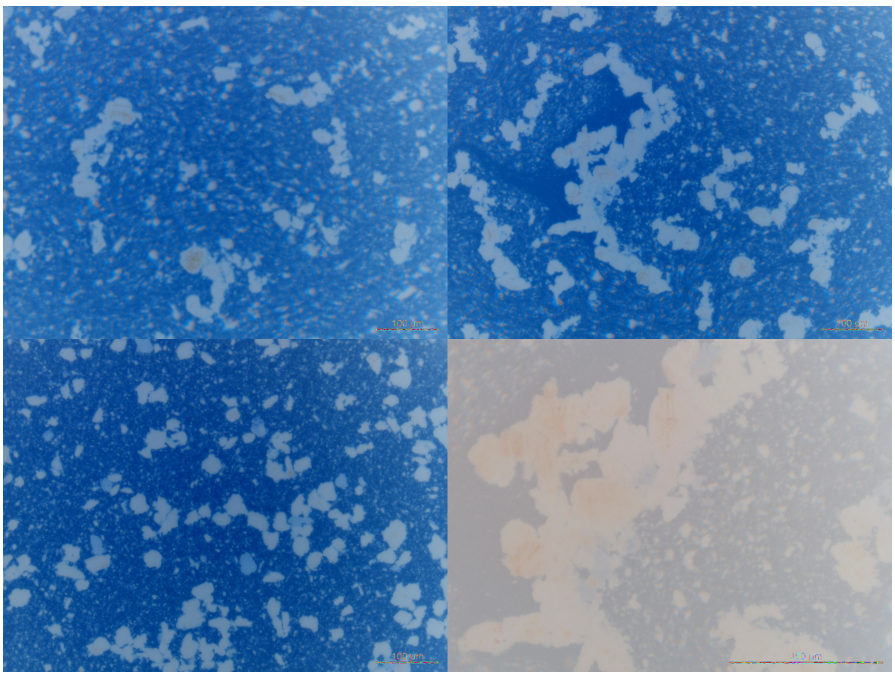

Fig. 4. Optical microscopy samples of graphite bisulfate (GBS)

3 Computational experiments

All studies were performed in the Google Colab Premium environment using A100 SXM4 40GB GPU.

3.1 Data

Before conducting the research, 79 images of graphene synthesis results (optical microscopy photographs of graphite bisulfate (GBS) samples) with specified oxidants and reagents, which were used during the synthesis, were obtained from the Faculty of Chemistry and Geosciences of Vilnius University. In GBS synthesis, three different oxidants were used:

- Ammonium persulfate - $(NH_4)_2S_2O_8$
- Potassium persulfate - $K_2S_2O_8$
- Chromium trioxide - CrO_3

The GBS synthesis procedure was repeated using all the oxidants and additionally adding a water binding reagent - phosphorus pentoxide P_2O_5 to the oxidizing mixture. Thus, a total of six data classes were obtained, which are named:

- NH4)2S2O8
- K2S2O8
- CrO3
- NH4)2S2O8_P2O5
- K2S2O8_P2O5
- CrO3_P2O5

3.2 Data processing

The raw data are JPG format photos with a resolution of 2560 x 1920. The pictures show the result in a large scale, but the number of pictures is not large, so for a better quality of network generalization, a 1000 x 1000 resolution frame is selected at a random place in the picture and the picture is cropped around the frame. However, the resolution of a 1000 x 1000 photo is extremely high and its processing would theoretically require a very large neural network, the calculations of which would be extremely expensive, so the resolution of the photo is reduced to 128 x 128. These augmentations are chosen for optimal model training and subsequent exploitation without damaging essential information in the data.

3.3 CLIP model training

The individual components that make up the CLIP model were downloaded using the Hugging Face `transformers` library [26]: the CLIP photo encoder component and the CLIP text encoder component. This decomposition is chosen for convenience in the later implementation of the overall goal of the work, which requires only the CLIP text encoder component. Also downloaded: CLIP feature extractor and CLIP tokenizer in `"openai/clip-vit-base-patch32"` configuration. Settings for the CLIP video encoder and text input encoder components

can be found in appendix A. Projection heads of photo embeddings and text embeddings were also used, which unify the dimensions of the embeddings. These heads are small neural networks consisting of one residual block [9], the diagram is presented in appendix B.

The CLIP model was trained by providing it with (picture, text) data pairs:

1. The photo is passed through the CLIP image feature extractor
2. The resulting processed photo is passed through the CLIP image encoder
3. The resulting latents are passed through the projection head

1. The text is passed through the CLIP tokenizer.
2. The resulting tokens are passed through the CLIP text encoder
3. The resulting latents are passed through the projection head

Later, the probabilities are calculated, the value of the cross entropy loss function is estimated, the gradient is calculated and the CLIP components weights are updated. For the training process an Adam optimizer [3] and a cosine learning rate scheduler [10] were used. It was performed using the Hugging Face `accelerate` library for training over 200 epochs [8].

3.4 Diffusion model training

The diffusion model was downloaded from the Hugging Face `diffusers` library [16]. Unet architecture for this model was used. A scheme describing the model architecture is given in appendix D. Settings and hyperparameters are given in appendix C. Also, from the `diffusers` library a `DDPM` noise scheduler according to [12] was downloaded.

The diffusion model was trained by loading batches of pictures to it:

1. Noise is added to photos according to the noise schedule
2. Noisy pictures are passed through a diffusion model
3. The obtained noise estimate of the model is evaluated with the noise schedule reference and the value of the mean square error (MSE) loss function is calculated
4. Based on the received value, the gradient is calculated and the model weights are updated

For training, over 200 epochs, the Adam optimizer [3] with decoupled weight loss regularization [11], the cosine learning rate scheduler [10] were used, and the training process was performed using the Hugging Face `accelerate` library [8]. Afterwards, the diffuser images were generated using the `DDPMPipeline` class from the `diffusers` library. The implementation code will be shared in the future.

4 Results

4.1 CLIP results

For this work, CLIP model inference provided disappointing results. Currently, the model is not able to reliably classify the imaging data of graphene synthesis - for each of the classes of oxidants with which graphene was synthesized, the model assigns approximately equal probabilities during classification, when trying to guess the real oxidant.

Table 1. CLIP inference 1

Top predictions	
NH4)2S2O8_P2O5	16.76%
NH4)2S2O8	16.71%
K2S2O8_P2O5	16.67%
Actual text prompt	NH4)2S2O8_P2O5

Table 2. CLIP inference 2

Top predictions	
CrO3	16.78%
K2S2O8	16.74%
CrO3_P2O5	16.72%
Actual text prompt	NH4)2S2O8

Table 3. CLIP inference 3

Top predictions	
NH4)2S2O8_P2O5	16.73%
NH4)2S2O8	16.70%
K2S2O8_P2O5	16.67%
Actual text prompt	NH4)2S2O8

The figures above show three attempts to determine the oxidant, used to synthesize graphene, from a photograph. The "*Actual text prompt*" box in the tables shows the actual graphene synthesis result class, and above it, in descending order, the class predictions with probabilities. From the Table 1 example, we can see that the model correctly identified the oxidant class, but the other guesses get slightly lower probabilities. Similar classification probabilities are also seen in the Table 2-3 examples, where the model failed to correctly identify the graphene synthesis oxidant, and Table 2 the correct oxidizer class is not even among the three most likely guesses. It is believed that in this design, the CLIP model fails to draw strong differences between graphene synthesis samples synthesized with different oxidants, and in the latent space, all latents: both text and photos, are too closely spaced.

4.2 Diffusion model results

A mathematical expression for evaluating the quality of the model, except for the loss function, was not designed, therefore the most optimal hyperparameter values were selected by comparing the values of the loss function calculated after the appropriate number of training epochs (40). It was found that the most optimal tested learning rate value for this model is equal to $3 * 10^{-4}$. With values

of this hyperparameter higher than 10^{-3}, the model diverged. The study result figure for the learning rate is provided in Fig. 5.

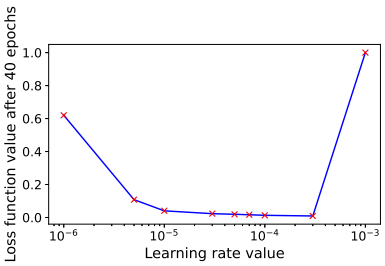

Fig. 5. Learning rate study for the diffuser

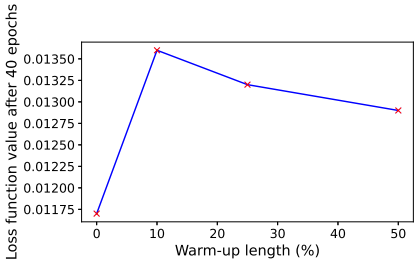

Fig. 6. Learning rate scheduler warm up study for the diffuser

The warm-up [10] of the cosine training step planner was also investigated. Warm-up values of different lengths were tested as a percentage (%) of the number of warm-up steps from the total number of model training steps. The optimal value of this hyperparameter was found to be 0%. A warm-up study of the cosine training step scheduler is displayed in Fig. 6.

The results of diffusion model generation were evaluated visually and by Frechet Inception Distance (FID) metric [25]. The images generated unconditionally by the model look similar to the images that were fed to the model during training. However, a closer look at the results reveals a lack of detail clarity - it is difficult to accurately distinguish graphene regions in generated images. This problem may be caused by the small amount of data and the low resolution of the generated photo (128 x 128). Examples of generated images are provided in Fig. 9.

Frechet Inception Distance was calculated comparing original photos (training data), disregarding the classes of the sample, with:

1. Same sample of original data
2. Same sample of original data, but within the same classes
3. Random images
4. Generated images by the first iteration model
5. Generated images by the best model

The results are shown in Fig. 7 - original FID score values and Fig. 8 - percentages of the scores compared with the highest score. The highest FID score, indicating the worst result, are reached by the first iteration's model generated images. Best model's generated images reach 68.51% of the highest score and indicate better generated photos resemblance to the original than comparing original data with random images. Yet the score is far from original data's comparison as it reaches negligible FID value and is the target of this task.

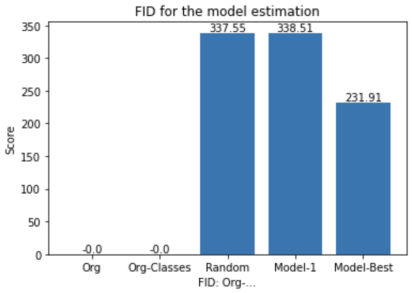

Fig. 7. Frechet Inception Distance scores

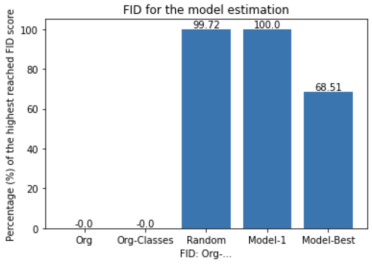

Fig. 8. Frechet Inception Distance score percentages

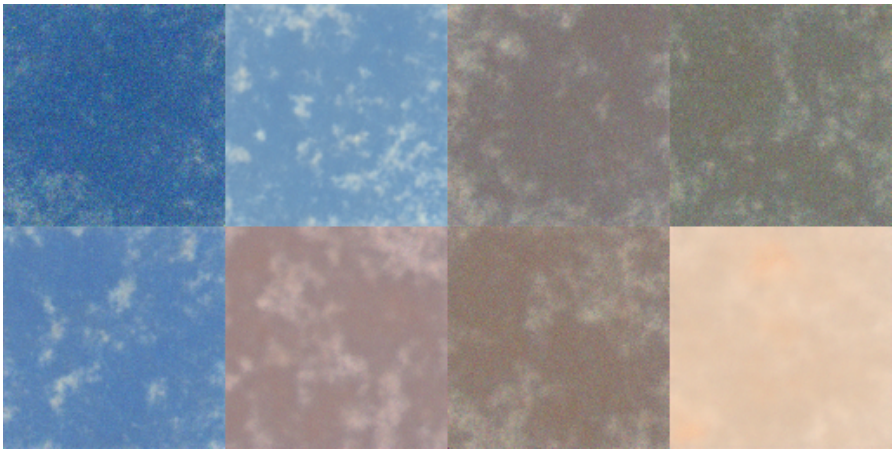

Fig. 9. Generated samples (resolution 128 x 128)

Conclusion

The novel text2image usage to encode chemical formulas for chemical images generation was proposed and successfully implemented. The results demonstrated the image generation part to be working more sufficient than graphene synthesis results classification. The encoder model assigns similar probabilities to all guess variants of the class. Furthermore, the diffusion model (generator) generates images similar to what would be expected, but these images are not detailed enough, lacking resolution. However, based on the tasks of text2image, these two models can be adapted to predict the results of graphene synthesis reactions.

On the other hand, for the development of a high-quality image prediction tool, there remain a number of tasks related to the adaptation of the CLIP model and the generative diffuser.

References

1. A.K.Geim, K.N.: The rise of graphene. Nature Materials **6**, 183–191 (2007)
2. David Nukrai, Ron Mokady, A.G.: Text-only training for image captioning using noise-injected clip. https://arxiv.org/abs/2211.00575 (2022), [v1] Tue, 1 Nov 2022 16:36:01 UTC (798 KB)
3. Diederik P. Kingma, J.B.: Adam: A method for stochastic optimization. https://arxiv.org/abs/1412.6980 (2014), [v9] Mon, 30 Jan 2017 01:27:54 UTC (490 KB)
4. Diederik P Kingma, M.W.: Auto-encoding variational bayes. https://arxiv.org/abs/1312.6114 (2013), submitted: [v11] Sat, 10 Dec 2022 21:04:00 UTC (3,451 KB)
5. Fatih Cagatay Akyon, A.T.: Deep architectures for content moderation and movie content rating. https://arxiv.org/abs/2212.04533 (2022), [v2] Mon, 12 Dec 2022 07:53:17 UTC (801 KB)
6. Geim, A.K.: Graphene: Status and prospects. Science **324**, 1530–1534 (2009)
7. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Sherjil Ozair, Aaron Courville, Y.B.: Generative adversarial networks. https://arxiv.org/abs/1406.2661 (2014), submitted: [v1] Tue, 10 Jun 2014 18:58:17 UTC (1,257 KB)
8. Gugger, S., Debut, L., Wolf, T., Schmid, P., Mueller, Z., Mangrulkar, S.: Accelerate: Training and inference at scale made simple, efficient and adaptable. https://github.com/huggingface/accelerate (2022)
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (2016). https://doi.org/10.1109/CVPR.2016.90
10. Ilya Loshchilov, F.H.: Sgdr: Stochastic gradient descent with warm restarts. https://arxiv.org/abs/1608.03983 (2016), [v5] Wed, 3 May 2017 16:28:09 UTC (1,385 KB)
11. Ilya Loshchilov, F.H.: Decoupled weight decay regularization. https://arxiv.org/abs/1711.05101 (2017), [v3] Fri, 4 Jan 2019 21:01:49 UTC (8,347 KB)
12. Jonathan Ho, Ajay Jain, P.A.: Denoising diffusion probabilistic models. https://arxiv.org/abs/2006.11239 (2020), submitted: [v2] Wed, 16 Dec 2020 21:15:05 UTC (9,137 KB)
13. Kong, Z., Ping, W., Huang, J., Zhao, K., Catanzaro, B.: Diffwave: A versatile diffusion model for audio synthesis. https://arxiv.org/abs/2009.09761 (2020), submitted: [v3] Tue, 30 Mar 2021 19:48:38 UTC (1,145 KB)
14. Mikhalevich, Y.: rclip: An ai-powered command-line photo search tool. https://mikhalevi.ch/rclip-an-ai-powered-command-line-photo-search-tool/ (2021)
15. Norris, J.R.: Markov Chains. No. 2, Cambridge University Press (1998)
16. von Platen, P., Patil, S., Lozhkov, A., Cuenca, P., Lambert, N., Rasul, K., Davaadorj, M., Wolf, T.: Diffusers: State-of-the-art diffusion models. https://github.com/huggingface/diffusers (2022)

17. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning transferable visual models from natural language supervision. https://arxiv.org/abs/2103.00020 (2021), submitted: [v1] Fri, 26 Feb 2021 19:04:58 UTC (6,174 KB)

18. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., Chen, M.: Hierarchical text-conditional image generation with clip latents. https://arxiv.org/abs/2204.06125 (2022), [v1] Wed, 13 Apr 2022 01:10:33 UTC (41,596 KB)

19. Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., Lee, H.: Generative adversarial text to image synthesis. https://arxiv.org/abs/1605.05396 (2016), [v2] Sun, 5 Jun 2016 13:39:27 UTC (2,147 KB)

20. Rimkutė, G., Gudaitis, M., Barkauskas, J., Zarkov, A., Niaura, G., Gaidukevič, J.: Synthesis and characterization of graphite intercalation compounds with sulfuric acid. Crystals **12**, 421 (03 2022). https://doi.org/10.3390/cryst12030421, https://doi.org/10.3390/cryst12030421

21. Rimkutė, G., Niaura, G., Pauliukaite, R., Gaidukevič, J., Barkauskas, J.: Wet synthesis of graphene-polypyrrole nanocomposites via graphite intercalation compounds. Crystals **12** (12 2022). https://doi.org/10.3390/cryst12121793, https://doi.org/10.3390/cryst12121793

22. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. https://arxiv.org/abs/2112.10752 (2022), [v2] Wed, 13 Apr 2022 11:38:44 UTC (38,971 KB)

23. Sankaran, K.J., Bikkarolla, S.K., Desta, D., Roy, S.S., Boyen, H.G., Lin, I.N., McLaughlin, J., Haenen, K.: Laser-patternable graphene field emitters for plasma displays. Nanomaterials **9** (2019), https://www.mdpi.com/2079-4991/9/10/1493

24. Sohl-Dickstein, J., Weiss, E.A., Maheswaranathan, N., Ganguli, S.: Deep unsupervised learning using nonequilibrium thermodynamics. https://arxiv.org/abs/1503.03585 (2015), submitted: [v8] Wed, 18 Nov 2015 21:50:51 UTC (6,095 KB)

25. Szegedy, C., Vanhoucke, V., Sergey Ioffe, J.S., Wojna, Z.: Rethinking the inception architecture for computer vision. https://arxiv.org/abs/1512.00567 (2015), [v3] Fri, 11 Dec 2015 20:27:50 UTC (228 KB)

26. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M.: Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. pp. 38–45. Association for Computational Linguistics, Online (Oct 2020), https://www.aclweb.org/anthology/2020.emnlp-demos.6

27. Wu, G.Y., Lue, N.Y.: Graphene-based qubits in quantum communications. https://arxiv.org/abs/1204.6365 (2012), submitted: [v2] Mon, 9 Jul 2012 02:32:19 UTC (1,426 KB)

A CLIP configuration

A.1 CLIP image encoder configuration

Table 4. CLIP image encoder configuration

Hidden layers dimension (*hidden_size*)	768
Encoder's feed forward layer dimension (*intermediate_size*)	3072
Number of hidden layers (*num_hidden_layers*)	12
Number of attention heads (*num_attention_heads*)	12
Number of image color channels (*num_channels*)	3
Resolution of images (*image_size*)	128 x 128
Resolution of an attention patch (*patch_size*)	32
Hidden layers' activation function (*hidden_act*)	"quick_gelu"
Normalization layers' epsilon ϵ value (*layer_norm_eps*)	1e-05
Dropout probability in fully connected layers (*dropout*)	0.0
Dropout probability in attention layers (*attention_dropout*)	0.0
Standard deviation of model weights initiation (*initializer_range*)	0.02
Model weights initiation factor (*initializer_factor*)	1.0

A.2 CLIP text encoder configuration

Table 5. CLIP text encoder configuration

Vocabulary size (*vocab_size*)	49408
Hidden layers dimension (*hidden_size*)	512
Encoder's feed forward layer dimension (*intermediate_size*)	2048
Number of hidden layers (*num_hidden_layers*)	12
Number of attention heads (*num_attention_heads*)	8
The maximum tokenized sequence length (*max_position_embeddings*)	77
Hidden layers' activation function (*hidden_act*)	"quick_gelu"
Normalization layers' epsilon ϵ value (*layer_norm_eps*)	1e-05
Dropout probability in fully connected layers (*dropout*)	0.0
Dropout probability in attention layers (*attention_dropout*)	0.0
Standard deviation of model weights initiation (*initializer_range*)	0.02
Model weights initiation factor (*initializer_factor*)	1.0
The ID of the padding token (*pad_token_id*)	1
The ID of the beginning-of-stream token (*bos_token_id*)	0
The ID of the end-of-stream token (*eos_token_id*)	2

A.3 General CLIP configuration and hyperparameters

Table 6. General CLIP configuration and hyperparameters

Training batch size	26
Number of training epochs	200
Gradient accumulation steps	1
Learning rate	$5 \cdot 10^{-2}$
CLIP temperature	0
Learning rate warm-up steps	0
Latents dimension	256

B Embeddings projection head scheme

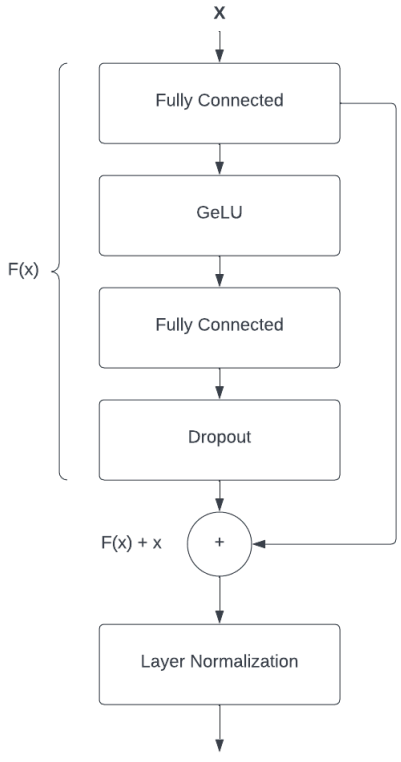

Fig. 10. Embeddings projection head scheme

C Diffusion model configuration and hyperparameters

Table 7. Diffusion model configuration and hyperparameters

Resolution of generated photos	128
Number of input image channels	3
Number of output image channels	3
Number of training epochs	200
Gradient accumulation steps	1
Learning rate	$3 \cdot 10^{-4}$
Learning rate warm-up steps	0
Number of diffusion steps	1000

D Architecture of the diffusion model - Unet

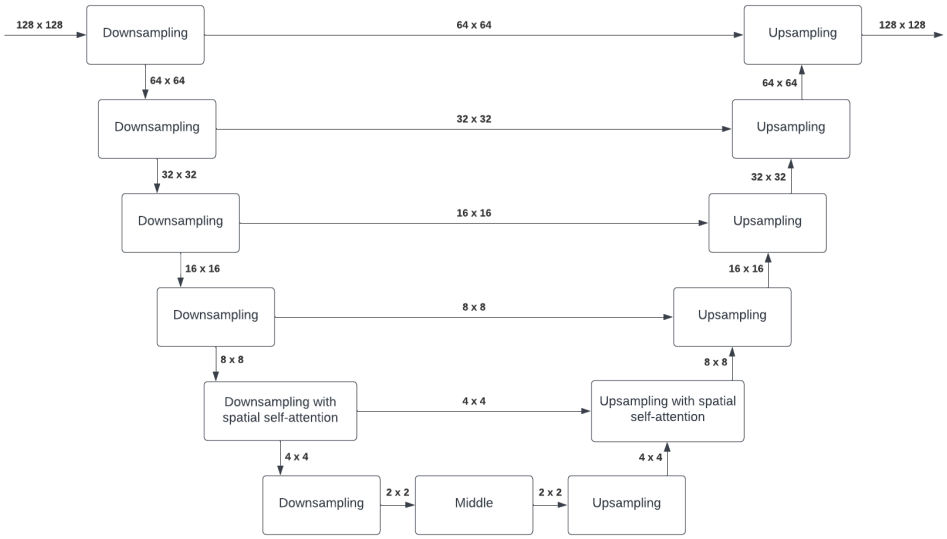

Fig. 11. Architecture of the diffusion model - Unet