2D Matters
An Attempt to Improve Vision Transformer

Liu Jinfan 3020202184
3020202184@tju.edu.cn
Tianjin University - Deep Learning Course

2023-1-15

Abstract

Vision Transformer (ViT)[1] has been a successful attempt that applies the Transformer model, proposed especially for Seq2Seq[2] problems on tasks involving data in 2 Dimensions. However, there still remains several spaces of improvements. In this report, experiments on the variations of the ViT model have been conducted in order to seek for the differences between the mechanism that help to reduce the drawbacks. Also, out of curiosity, I have tried to combined the ideas proposed by Liu in Swim Transformer[3] that utilizes an additional set of patches to obtain information between the edges of the original 16x16 blocks. The strcture of the report is organized in the following way. Section 1 is reviewing the related works and models in a state-of-the-art manner and some problems are revealed as well as possible solutions. Section 2 introduces some of the foundamental solutions to those problems discussed in section 1 in details while presenting the actual network structures after improvements. Section 3 includes foundamental elements of the expriments I have conducted, including datasets, platform and the process while a brief conclusion is generated. Section 4 summarizes the content above and reviews the contributions of the report to the current literature. References are listed in Section 5 in an IEEE style. All the code in this report is conducted with Python Jupyter Notebook and is available on Github: Youku video

Keywords – Transformer, ViT, Model Comparison

1 Introduction

Related Work

Original Transformer The original **Transformer** Model (Shown in Figure 1) was proposed in 2017 by google research team[4] aiming to solve tasks in Natural Language Processing (NLP) fields. The highlight of the paper is the introduction of **Attention**, a new mechanism that performs excellent on tasks involving Sequence to sequence (Seq2seq)[5] mapping. In the following years, this model has been extended in various directions with a massive numbers of improvements with respect to a numerous adaptations to meet different task requirements. Some are successful ([6, 7]) though, most of them are still in the field of Seq2seq. Yet, several attempts have been made on the tasks involving 2D information gatherings, such as embedding transformer in Convolutional Neural Networks (CNNs) to accelerate training process while boosting the training efficiency[8]. The difficulties of utilizing transformer in 2D tasks is that the Attention mechanism is designed for extranctions of similarities between tokens in a sequencial manner. However, when directly squeezing a 2D matrix into transformer as a 1D vector and pipelining it into the Transformer Model as 'word tokens', the predictions are in fact acceptable and can run over some of the advanced CNNs in the literature at that time. The paper that proposed the method, is the topic and core model I discuss here.

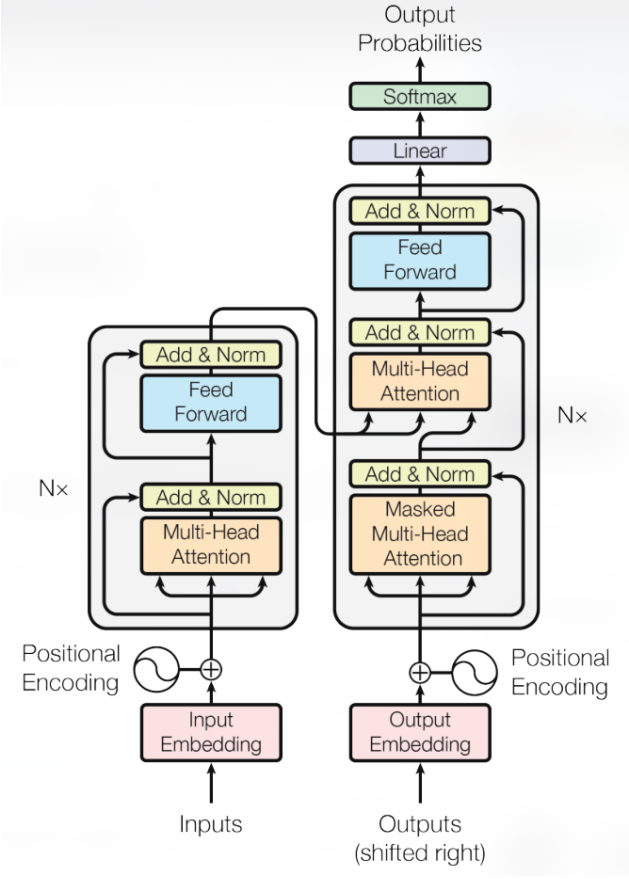

Figure 1: **Attention Model Architecture[4]** - The foundamental building block of the Transformer cluster of models.

Vision Transformer The Vision Transformer (ViT)[1] is known as the first to cut out convolutional layers in a model involving pure transformer layers. This model is designed to utilize the advanced performance of the state-of-the-art implementation to Transformer Model which could significently reduce unnecessary labor. The key concept of the

paper is an approach that squeezes a 2D matrix of image to a 1D-fixed-sized-vector by dividing the original image into several 16x16 size pathes and perform a mapping that maps each patch to a vector. As is shown in Figure 2, the image is divided into 9 patches with each path containing $16 \times 16 = 256$ pixels. And the model used only the **Encoder** part of the Transformer. The method could have preserved several information though, including the relative positional information inside each block, the relationships between blocks that stands next to each other vertically are lost. This meagre bias could be resolved by training on large scale datasets or by special training strategy of distillation[9]. Apart from that, some other models ([10, 11, 12, 13, 14]) have proposed their special solutions to this bias.

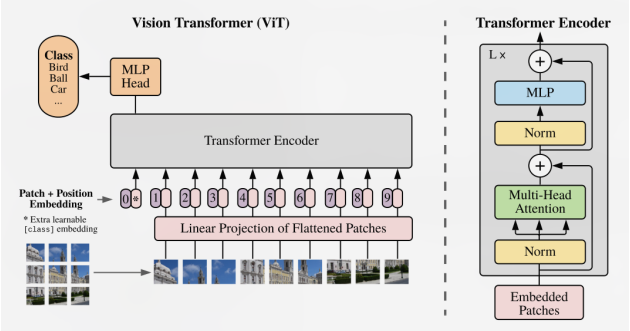

Figure 2: **Vision Transformer Model[1]** - The very model that brought pure transformer-based structure avaible for image classification tasks.

My Work

Experiment Basically, I have conducted codes that run classification tasks on the same dataset with models discussed above that have extra modifications on the ViT as well as the original ViT itself. Additionally, I have modified the basic structure of the input data so that to fix up one possible drawbacks of ViT that I have pointed out.

Little Improvement As is discussed above, the ViT does not consider the relative positional information of each pair of patches that stands besides vertically. Therefore, I have came up with a solution that modifies the input structure of the data so that the original ViT model could take border information into considerations. After a rough comparison, my method could be slightly better than the original ViT without no explicit modifications on the inner structure of the model itself. Instead, I have figured out a method that by preprocessing the input image, the same results could be achieved while utilizing the out-of-box, plug-and-play and state-of-the-art implementations of ViT from Github[15].

Contribution Summary

I would summarize my contribution to the report as the following points:

1. **Code.** I have been searched and coded for the whole part, including the borrowed code from [15] and customization input structure methods.
2. **Analysis.** I have done the analysis and comparison of **Five** variations of implementations of **Vision Transformer** in a rather casual manner.
3. **Writing.** Though in a rush, the report itself if thoroughly written in English all by myself. No translation application is used to write a single sentence.

2 Research Methods

In this section, the approach to collect border information is introduced together with my own simple yet effective implementation in Pytorch. Also, the overall structure of the Network is presented here.

Improvements on ViT

Vision Transformer[1] is a combination of visialized 2D data and sequencial encoder methodologies. In short, it views a 2D image as a sequence of small, fix-sized patches that can be encoded as "image tokens" and be inputed into normal transformer encoders to be proporgated the in same way as "word tokens" are. And because of this mechanism which does not present relative information about the second dimension, I believe there can be spaces for improvements.

Swin Transformer[3] proposed a possible solution that ***shifted windows*** for a new round of calculation where the new windows perfectly cross the boundaries of the previous windows division schema.

I borrowed from this method. In my solution, I send an additional set of patches that the include every cross boundaries of the original ones. Shown in Figure 3, the illustration of B is the addtional set of patches that take on all the cross boundaries in A, providing extra usefull information for classification.

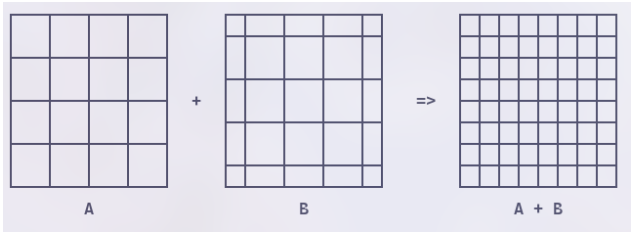

Figure 3: **My Proposed Methodologies** - A is the original patches while B is the additional patches and B's patches can perfectly match every cross boundaries in A

Addtional Patches The approach I took, for coding, is by performing a transformation on the input matrix of images. To be specific, I move $half\_patch \times W$ pixels from top to bottom and $H \times half\_patch$ pixels from left to right, or vice versa. More specifically illustrated, shown in Figure 4, the red pattern is moved from top to bottom while the light blue pattern is moved from ***Left*** to right and shown in Figure 5, while the original division cannot infer the relationship between the black and red stars, the new patches on the ***Right*** circle the two in one patch which infers the two have strong connections in 2 dimensional manners.

Note that: the actual patch size of can be quite small compared to the overall resolution of the image, which in this case is 32 : 256. Therefore, the addtional patch shall not interrupt with the classification process.

Concation After the addtional patches is produced, we can now perform a concation to bind the two patches and make the Normal ViT Network get the input as normal as possible. Therefore, I concat 4 sets of patches in a 2D manner to ensure the width and height are equaled. And as is shown in Figure 6, the inference between number 1, 2, 3 and 4 can be determined by the set of patches on the right top side.

Note that: the methodology here is to utilize the already implemented APIs with the minimum modification which is learnt from ViT[1] paper where they did the minimum modification on the image input method and took advantages of the already advancely implemented Transformer APIs.

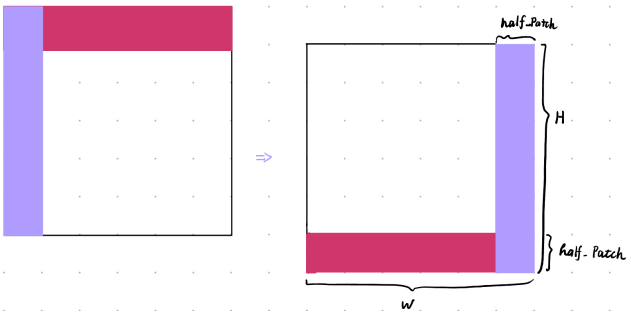

Figure 4: **Illustration 1** - The Colored Red and Light blue patterns are moved from top to bottom and left to right.

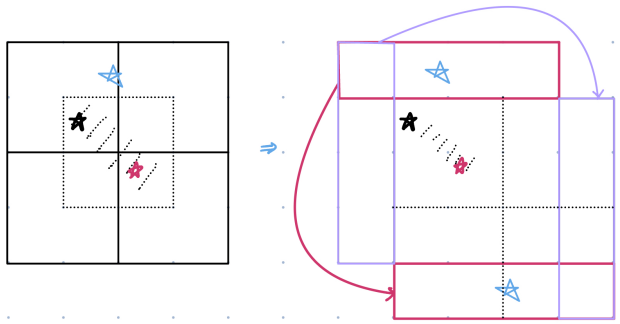

Figure 5: **Illustration 2** - The Original Division (Left) cannot refer the relationship between the black star and red star while the Additional Schema (Right) can infer the relationship between them since they are in the same patch.

3 Experiment and Analysis

Dataset

Food 101[16] Food 101 is chosen as the dataset for these sets of experiment. The data set is presented by L Bossard[16] in 2014 aiming for image classification. In the experiment, because we perform a rather complicated concation, hopefully, the inference from the image can be independent from the 2D information related to the original image such as position, relative position, etc. And because

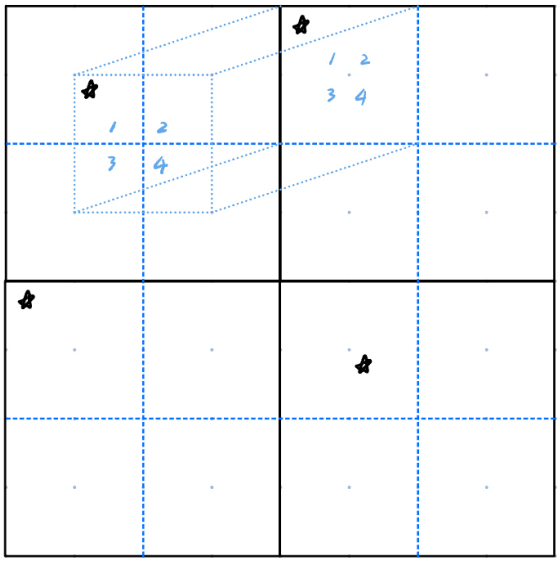

Figure 6: **Illustration 3** - The Colored Red and Light blue patterns are moved from top to bottom and left to right.

the classification tasks do not involve those information, the dataset is a perfect choise. On top of that, the food 101 dataset is an old yet large scale dataset which contains a variety of classifications and is still widely used. It is user friendly, providing interface that can be easily downloaded and used via HuggingFace[17] interface.

Listing 1: Code for Loading Food 101 via HuggingFace

```
food = load_dataset("food101", split="train[:1000]")
food = food.train_test_split(test_size=0.2)
"""
... (Data split, Concation) ...
"""
# Show some Figures
fig, axes = plt.subplots(3,3,figsize=(16,12))
for idx, ax in enumerate(axes.ravel()):
    label = train_data["label"][idx]
    img = train_data["img"][idx]
    ax.set_title(id2label[str(label)])
    image = transforms.ToPILImage()(img).convert('RGB')
    ax.imshow(image)
```

Comparison and Analysis

Choices of Model To make a comparison, we have to first determine which models to compare and which datasets to perform on. According to previous study, I have chosen "ViT"[1], "Effiecency Transformer"[18](Optimization on Transformer), "CaiT"[12], "DeepViT"[11] and our customization Model that built on normal "ViT" structure.

Basic Setup The models are coded with pytorch with help from vi-pytorch[15] and run on a Mechine armed with 3090Ti and is rent via autodl[19]. Detailed information is listed below with some figures for further illustration.

- OS: Arch Linux 6.1.4
- Platform: 3090Ti Nvidia Cuda 11.3
- Datasets: Food101[16]

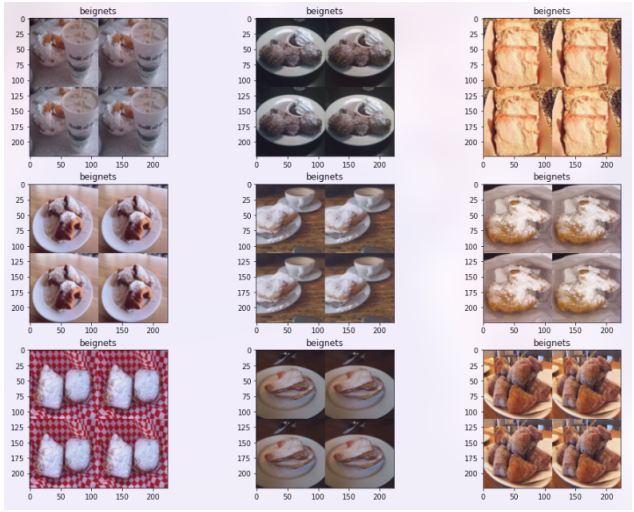

Figure 7: **Some samples in the Dataset Food 101** - After the concation

- Backend: Pytorch
- Epoch: 30

```
!nvidia-smi
Tue Jan 17 16:14:00 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 495.44       Driver Version: 495.44       CUDA Version: 11.5     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ...  On   | 00000000:E1:00.0 Off |                  N/A |
| 30%   24C    P8    31W / 350W |      0MiB / 24268MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Figure 8: **Nvidia Information**

Other conditions All four models are trained under the same circumstances, and according to previous knowledge and some pre-train estimation, I decided to run each for 30 loops and the learning rate is set to 3×10^{-5}. Also, as is illustrated before, the input size are all set to 256×256.

Results and Analysis Results are shown as below in Figure 9. According to the figure shown below, we can make the following Analysis:

1. The ViT and three of its variations all learnt to be ***overfitted*** yet DeepViT, due to it depths, which though, requires addtional hardware accelerator and a significant larger memory, holding a relatively higher train loss as Epoch grows, remains a relatively lower valuation loss compared to other three modules, indicating a better performance. Also, according to Table 1, the best performanced model here is still D-ViT, which holds the lowest Validation Error and the highest Validation Accuracy while costing a relatively acceptable time. And the Effiecency Model shows a realtively Low performance, which from my point of view, is due to its overfitting, since it reaches 100% accuracy in training sets right after 23 epochs.
2. The improved input method is working well but not directing the model in a correct direction. According to the Training Loss chart shown in Figure 9, the Loss of this kind decreases dramatically and is the lowest among all models. However, the low Loss in Training sets does not necessary means good performance. Instead, since the validation Loss is quite high at the end of 30 Epoch, I could infer that the model quickly overfitted after 30 Epoch. Yet, this overfitting phenomenon could prove, from a perspective, that a slight change in the patches could result in leading the model to learn quicker, adapt faster and get overfitted sooner and this saves time.

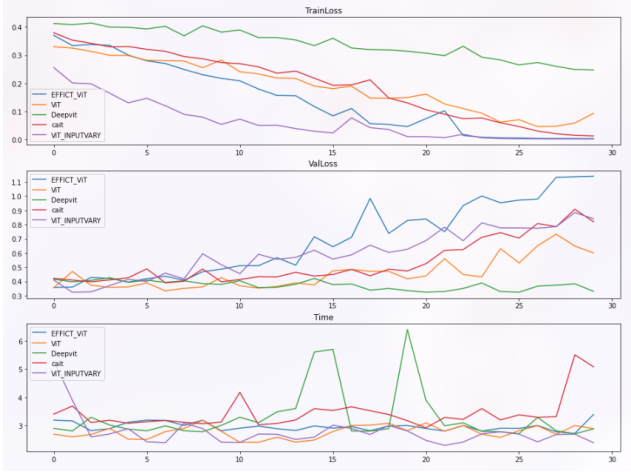

Figure 9: **Training Results** - **Top** is the Train Loss value with respect to training epoches; **Middle** is the Loss of Validation Set; **Bottom** is the Time Spent of each Epoch, indicating the efficiency. ***EFFECT_ViT***: Effiecency Vit Implementation[18]; ***ViT***: Normal ViT[1]; ***DeepViT***: DeepVit[11]; ***CAIT***: CAIT[12]; ***ViT_INPUTVARY***: My Special Implementation;

Model	Time	V-Accuracy	V-Loss
ViT	83.2	0.85	0.60
E-ViT	89.0	0.80	1.14
D-ViT	99.0	0.87	0.33
CAIT	103.3	0.81	0.82
C-ViT	83.5	0.84	0.84

Table 1: Time, V-Accuracy, V-Loss of 5 models after 30 Epoch Training

4 Conclusion

In a nutshell, I proposed a possible solution to fix the problem of lacking 2D inference information in the original structure of vision transformer and conducted several experiments on a portable platform to evaluate and compare three variations of vision transformer as well as the original one and my customized one. The results shows:

- The original ViT is designed to meet certain tradeoff that it is balanced to avoid overfitting.
- The Efficient way could speed the training process, yet could run into overfitting quickly.
- Deep Vision Transformer could avoid significently the overfitting trap, yet it requires a better hardware environment and could result in crashdown in smaller memory machines running on CPU (personally, my 32GB memory, together with its 32GB swapfile are both filled with the sources and eventually crashed the system).
- The proposed improvements on the preprocessing of the input image could provide addtional information that accelerates the speed of the normal vision transformer, yet the method can be improved if further edition could be done inside the ViT rather than providing a special input for varification.

Future work could be done in the following ways:

- The Modification inside transformer's attention mechanism which could be improved especially for 2D matrix information gathering.
- The combination of vision transformer and other transformers to conduct Multimodality Tasks.

References

[1] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[2] E. Egonmwan and Y. Chali, "Transformer and seq2seq model for paraphrase generation," in *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pp. 249–255, 2019.

[3] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 2021.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[5] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

[6] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *International conference on machine learning*, pp. 7354–7363, PMLR, 2019.

[7] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, "Neural speech synthesis with transformer network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 6706–6713, 2019.

[8] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018.

[9] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International Conference on Machine Learning*, pp. 10347–10357, PMLR, 2021.

[10] L. Beyer, X. Zhai, and A. Kolesnikov, "Better plain vit baselines for imagenet-1k," *arXiv preprint arXiv:2205.01580*, 2022.

[11] D. Zhou, B. Kang, X. Jin, L. Yang, X. Lian, Z. Jiang, Q. Hou, and J. Feng, "Deepvit: Towards deeper vision transformer," *arXiv preprint arXiv:2103.11886*, 2021.

[12] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. Jégou, "Going deeper with image transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 32–42, 2021.

[13] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z.-H. Jiang, F. E. Tay, J. Feng, and S. Yan, "Tokens-to-token vit: Training vision transformers from scratch on imagenet," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 558–567, 2021.

[14] A. Hassani, S. Walton, N. Shah, A. Abuduweili, J. Li, and H. Shi, "Escaping the big data paradigm with compact transformers," *arXiv preprint arXiv:2104.05704*, 2021.

[15] P. Wang, "vit pytorch," 12 2021.

[16] L. Bossard, M. Guillaumin, and L. Van Gool, "Food-101–mining discriminative components with random forests," in *European conference on computer vision*, pp. 446–461, Springer, 2014.

[17] S. M. Jain, "Hugging face," in *Introduction to Transformers for NLP*, pp. 51–67, Springer, 2022.

[18] Y. Xiong, Z. Zeng, R. Chakraborty, M. Tan, G. Fung, Y. Li, and V. Singh, "Nyströmformer: A nyström-based algorithm for approximating self-attention," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 14138–14148, 2021.

[19] "autodl gpu renting." https://www.autodl.com/home. Accessed: 2023-01-15.