Computational Visual Media
https://doi.org/10.1007/s41095-021-0203-2 Vol. 7, No. 1, March 2021, 153–157
Short Communication

Jittor-GAN: A fast-training generative adversarial network model zoo based on Jittor

Wen-Yang Zhou[1], **Guo-Wei Yang**[1], **and Shi-Min Hu**[1] (✉)

With the emergence of CNNs and massive datasets, the performance of many tasks in computer vision has been greatly improved, such as object detection, instance segmentation, and image generation. The last has many novel applications, including image-to-image translation, image inpainting, and image super-resolution. It can generate authentic and creative images.

GAN [1] is the current mainstream model for image generation. It usually consists of an encoder, a generator, and a discriminator, which are constructed from CNN layers. The encoder is responsible for mapping images to a latent space. The generator is responsible for generating images from latent vectors, using one or multiple images. The discriminator is responsible for distinguishing generated images from real images. Through joint adversarial training of the generator and the discriminator, the generative ability of the generator is continuously improved, thereby generating more and more realistic images.

However, training GAN models is time consuming. Thus, we have implemented a GAN model zoo based on Jittor, a fully just-in-time (JIT) complied deep learning framework by Tsinghua University [2]. This model zoo is a collection of 27 mainstream GAN models published from 2014 to 2019, listed in Table 1. These models have an average of 3070 citations per model, and they have great influences and have been widely used in both academia and industry. Our model zoo covers 4 kinds of tasks, including image generation (G), image-to-image translation (T), super-resolution (S), and image inpainting (I). Table 1 lists the models for different tasks and their representative models. Further details of these 27 GAN models can be found in Ref. [3].

1 BNRist, Tsinghua University, Beijing 100084, China. E-mail: W.-Y. Zhou, zhouwy19@mails.tsinghua.edu.cn; G.-W. Yang, ygw19@mails.tsinghua.edu.cn; S.-M. Hu, shimin@tsinghua.edu.cn (✉).

Manuscript received: 2020-10-26; accepted: 2021-01-02

1 Why can Jittor-GAN accelerate model training?

Training the Jittor models is 2.26 times faster than equivalent PyTorch models on average. There are three main reasons. Firstly, Jittor's unique operator fusion mechanism saves much memory access time. Secondly, Jittor's optimization makes better utilization of GPU computing resources. Thirdly, Jittor's precise back propagation algorithm avoids computing derivatives of parameters that do not need to be updated.

1.1 Operator fusion

Jittor proposed the concept of meta-operators, which cover three operator categories: reindex, reindex-reduce, and element-wise. Most common element operators can be fused, e.g., convolution and matrix multiplication. Jittor also has *a unique meta-operator fusion mechanism*, which can fuse adjacent operators together. After doing so, intermediate results do not need to be stored in memory, saving memory read and write time. In addition, Jittor uses lazy execution, allowing Jittor to fuse more meta-operators for increased optimization.

Jittor's unique lazy execution mechanism separates construction of the calculation graph from calculation, and performs calculation when a result is required or the calculation graph reaches a certain scale. In contrast, PyTorch uses eager execution for calculation graphs, so results are calculated as soon as the calculation graph is constructed. For example, when

Table 1 The 27 GAN models, their task (T, see text), batch size (B), image dimensions (D, height and width), and number of training iterations per second using Jittor (JT) and PyTorch (PT) frameworks. Speed up (SU) measures Jittor's relative speed to PyTorch (Max: 3.83, Min: 1.27, Avg: 2.26)

Model	T	B	D	PT	JT	SU
SoftmaxGAN	G	64	28	32.63	124.82	3.83
WGAN-GP	G	64	28	36.01	133.88	3.72
WGAN-DIV	G	64	28	7.13	25.66	3.60
ClusterGAN	G	64	28	54.37	184.47	3.39
RelativisticGAN	G	64	32	20.83	69.28	3.33
AAE	G	64	32	33.26	93.20	2.80
BGAN	G	64	28	45.07	125.32	2.78
GAN	G	64	28	44.68	122.56	2.74
DRAGAN	G	64	32	21.46	53.98	2.52
LSGAN	G	64	32	34.25	78.35	2.29
WGAN	G	64	28	64.51	145.75	2.26
EBGAN	G	64	32	36.17	76.62	2.12
CGAN	G	64	32	33.93	69.70	2.05
BEGAN	G	64	32	31.34	58.83	1.88
INFOGAN	G	64	32	24.00	44.75	1.86
SGAN	G	64	32	31.27	56.03	1.79
DCGAN	G	64	32	41.34	73.70	1.78
PIXELDA	G	64	32	14.09	24.47	1.74
ACGAN	G	64	32	30.40	51.00	1.68
COGAN	G	32	32	19.22	31.17	1.62
UNIT	T	1	64	7.55	15.13	2.00
CYCLEGAN	T	1	64	6.97	12.25	1.76
Pix2Pix	T	1	256	25.40	40.44	1.59
StarGan	T	16	128	7.90	10.14	1.28
BICYCLEGAN	T	8	128	5.65	7.16	1.27
ESRGAN	S	4	64	3.38	4.97	1.47
ContextEncoder	I	8	128	31.95	55.83	1.75
Average	—	—	—	27.58	66.28	2.26

computing convolutions, the calculation is performed immediately after the image data is input and the results are stored in memory. This has the advantage that the network structure can be particularly flexible, and the network can include conditional or loop statements. The disadvantage is that it limits the potential for optimization. If two element operators are in different operators and they can be merged, the eager execution mechanism will need to save the result of the first element operator in memory, so it cannot be merged with the second element operator. The lazy execution mechanism will optimize and determine the largest possible calculation graph, with maximal meta-operator fusion.

1.2 GPU utilization

When training a network, greater GPU utilization leads to more fully utilized computing resources and faster computation. Jittor improves GPU utilization through *Fetch Sync* methods.

1.2.1 Async fetch

Fetch Sync is a unique asynchronous interface in Jittor. When training a network, users often output the loss of each round for observation. In order to output network results, PyTorch forces GPU and CPU data synchronization using the `cuda_synchronize` function. This blocks the code running until the required output calculation is complete and copied to the CPU, causing the pipeline to be emptied, resulting in decreased GPU utilization. Fetch Sync supports asynchronous acquisition of network results, and the corresponding function is called for output after the network result is calculated and transmitted.

1.2.2 Kernel launches

Small amounts of data (for example, training WGAN using the MNIST dataset uses size $64 \times 1 \times 28 \times 28$) can be processed quickly in the GPU, making the GPU frequently wait, resulting in low GPU utilization in PyTorch. Jittor's operator fusion can reduce the number of kernel launches, thereby reducing CPU–GPU communication and improving GPU scheduling.

1.3 Precise backward algorithm

A GAN model has a generator and a discriminator. The discriminator determines whether an image is real or generated by the generator. The task of the generator is to generate an image that is difficult for the discriminator to distinguish, to provide confrontational training.

When calculating parameters' gradients, PyTorch uses `loss.backward()` to propagate the gradient to all related parameters, while Jittor uses `jt.grad(loss, parameters)` to avoid unnecessary gradient calculations. For example, gradients of the generator do not need to be calculated when training the discriminator. PyTorch calculates the gradients of both generator and discriminator if the variables fed to the discriminator are not detached, while Jittor just calculates gradients of the discriminator. Therefore, Jittor's gradient calculation method is *a point-to-point gradient calculation* which requires less computation than PyTorch.

2 Experiments

We first compare training speed for Jittor and Pytorch on the GAN model zoo. We then show generated

results for the 4 tasks mentioned in Table 1. Jittor-GAN model zoo has been available in GitHub[①].

2.1 Model training speed

The biggest advantage of our model zoo is that model training is very fast due to the targeted optimizations of the Jittor framework. To demonstrate our speed advantage, we compare with the currently popular deep learning framework PyTorch (version 1.3.0).

To ensure fairness, we ensure that the network architecture, input image, and network parameters are identical. All models were tested on an NVIDIA Titan RTX Graphics Card with E5-2678 v3 CPU. We ran 100 times to warm up the model, and then ran 1000 times to test the speed of model training.

Results are shown in Table 1, giving the number of training iterations per second for Jittor and PyTorch frameworks. It can be seen that the training speed of all Jittor models is faster than for Pytorch equivalents, ranging from 1.27 to 3.83 times faster, with an average of 2.26. Therefore, using our model zoo can greatly improve model training for development.

2.2 Applications

We now consider our models' performance on different datasets for 4 tasks.

Image generation was one of the first and is one of the most popular applications of GANs. We provide 20 GAN models for image generation, such including GAN, CGAN, DCGAN, and WGAN. Outputs for some common and important models using the MNIST dataset are shown in Fig. 1.

Image-to-Image translation aims to convert an image to another image domain while ensuring that the image content is consistent. We provide 5 GAN models: CYCLEGAN, Pix2Pix, BICYCLEGAN, UNIT, and StarGan. Example results using the map dataset are shown in Fig. 2.

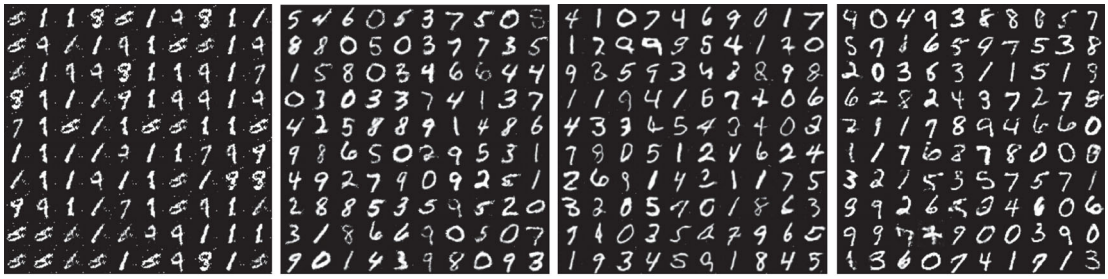

Fig. 1 Results generated using the MNIST dataset.

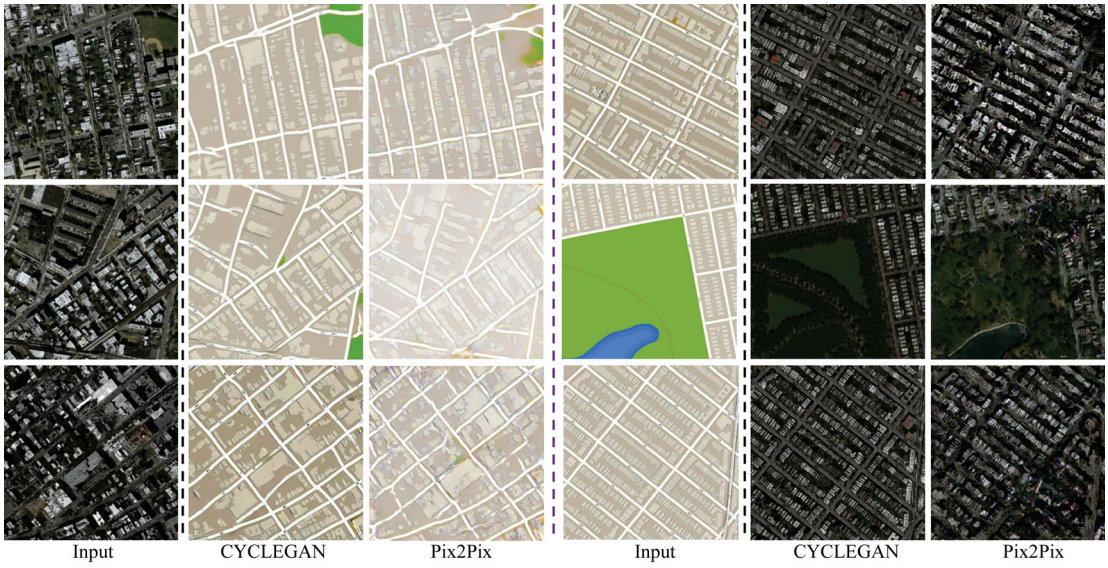

Fig. 2 Generated results on map dataset. On the left is the conversion of real images to sketches, and on the right is the conversion of sketches to real images.

① https://github.com/Jittor/ganjittor.

The super resolution task aims to generate high-resolution images from lower-resolution ones. Results from ESRGAN on the celeba dataset are shown in Fig. 3.

Image inpainting aims to fill in missing image blocks in an image. Results from ContextEncoder on the celeba dataset are shown in Fig. 4.

Fig. 3 Generated results of ESRGAN on celeba dataset. The left image of each group is a low-resolution image, and the right image is a high-resolution image output by ESRGAN.

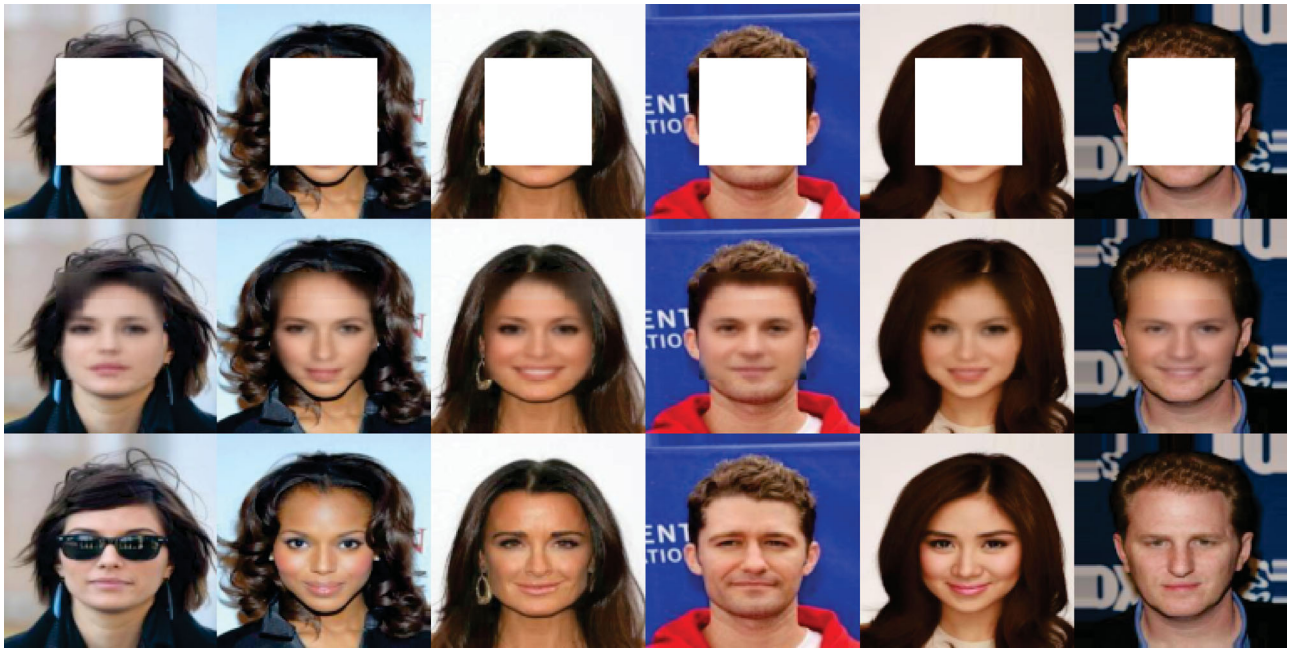

Fig. 4 Generated results of ContextEncoder on celeba dataset. The first line is the input image, the second line is the image predicted by ContextEncoder, and the third line is the real image.

Acknowledgements

This work was supported by National Natural Science Foundation of China (No. 61521002).

References

[1] Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. In: Proceedings of the 27th International Conference on Neural Information Processing Systems, Vol. 2, 2672–2680, 2014.

[2] Hu, S.-M.; Liang, D.; Yang, G.-Y.; Yang, G.-W.; Zhou, W.-Y. Jittor: A novel deep learning framework with meta-operators and unified graph execution. *Science China Information Science* Vol. 63, No. 12, 222103, 2020.

[3] Cao, Y.-J.; Jia, L.-L.; Chen, Y.-X.; Lin, N.; Yang, C.; Zhang, B.; Liu, Z.; Li, X.-X.; Dai, H.-H. Recent advances of generative adversarial networks in com-puter vision. *IEEE Access* Vol. 7, 14985–15006, 2019.

Wen-Yang Zhou is currently a Ph.D. student in the Department of Computer Science and Technology, Tsinghua University, Beijing. His research interests include computer graphics, image analysis, and computer vision.

Guo-Wei Yang is currently a Ph.D. student in the Department of Computer Science and Technology, Tsinghua University. His research interests include computer graphics, image analysis, and computer vision.

Shi-Min Hu received his Ph.D. degree from Zhejiang University, in 1996. He is currently a professor with the Department of Computer Science and Technology, Tsinghua University. He has authored over 100 papers. His research interests include digital geometry processing, video processing, rendering, computer animation, and computer-aided geometric design. He is the Editor-in-Chief of *Computational Visual Media*, and on the Editorial Board of several other journals, including *Computer Aided Design* and *Computer & Graphics* (both Elsevier).