Distributed Training of Generative Adversarial Networks(GANs)

Ankitha Conjevaram Sai Narayan(A0228519X), Rashi Sharma(A0228492X), Manish Sridhar(A0225502U)

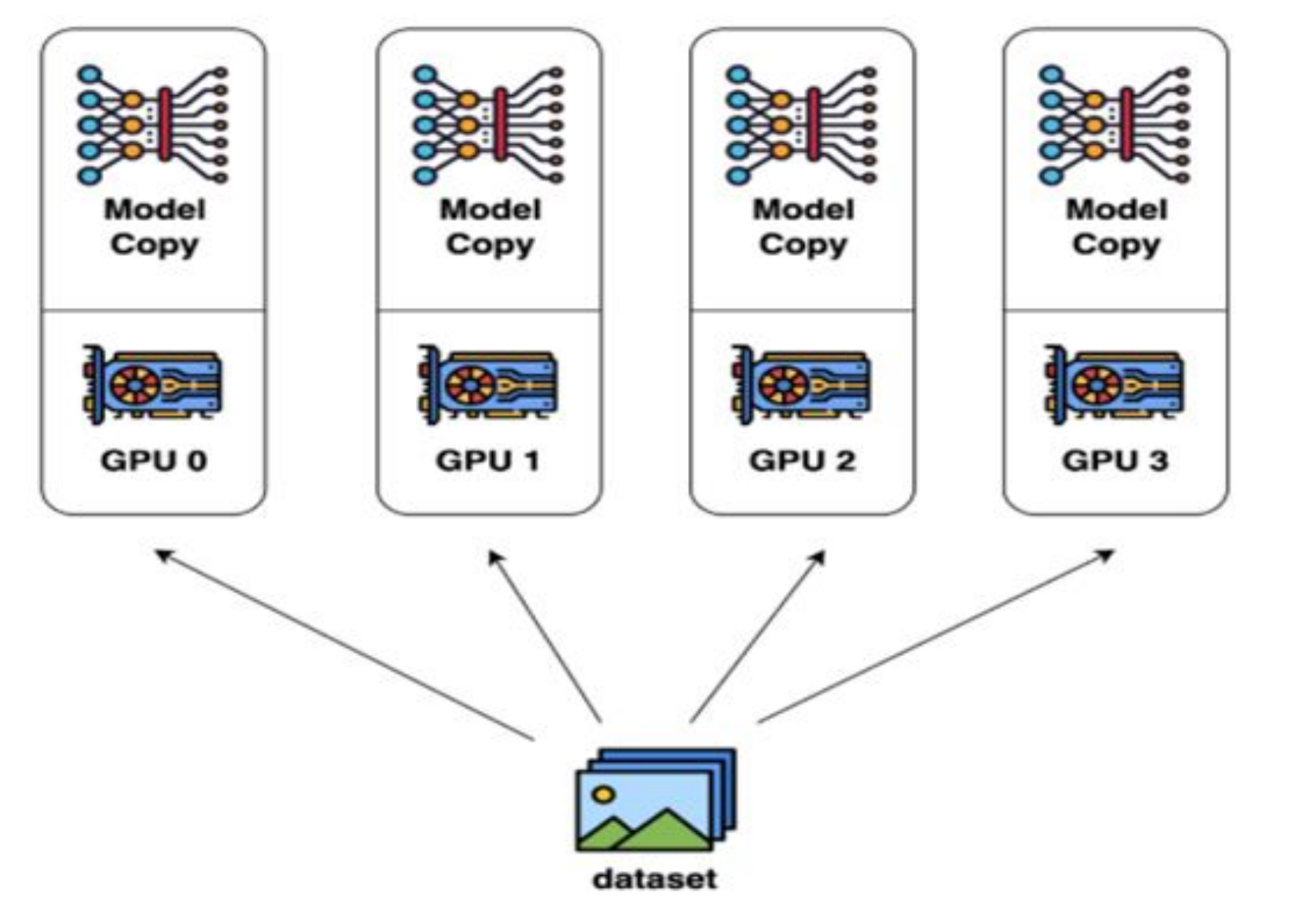
NUS National University of Singapore

Motivation

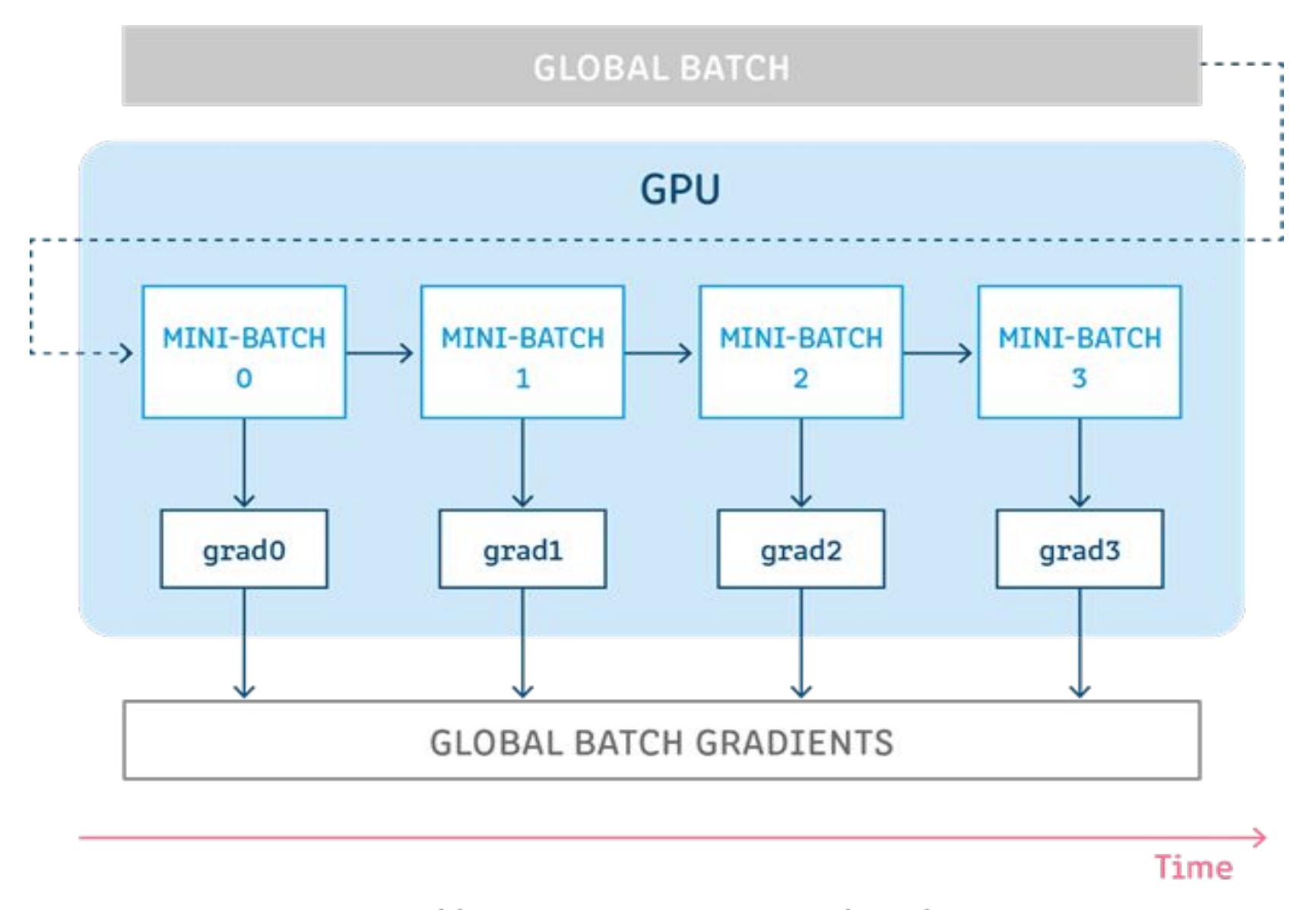
Although the results generated by GANs can be remarkable, it can be challenging to train a stable model. The reason they are difficult to train is that both the generator model and the discriminator model are trained simultaneously. This means that improvements to one model come at the expense of the other model.
As a result, training a GAN on a single GPU can take hours and for bigger GANs, even days.
In order to optimize the training process, we use Colossal-AI, a deep learning system that provides a variety of features that aid in large scale parallel training using multiple GPUs.

Experiments

Colossal AI provides a number of ways for faster training of larger models by parallelization and optimizations. The pix2pix GAN model which has been experimented on in this project has a total of approximately 57 million parameters to be trained. For stable training and due to limited resources the improvement in training time for the pix2pix model was observed by training it on a single GPU, with a batch size of 1.



Data Parallelism

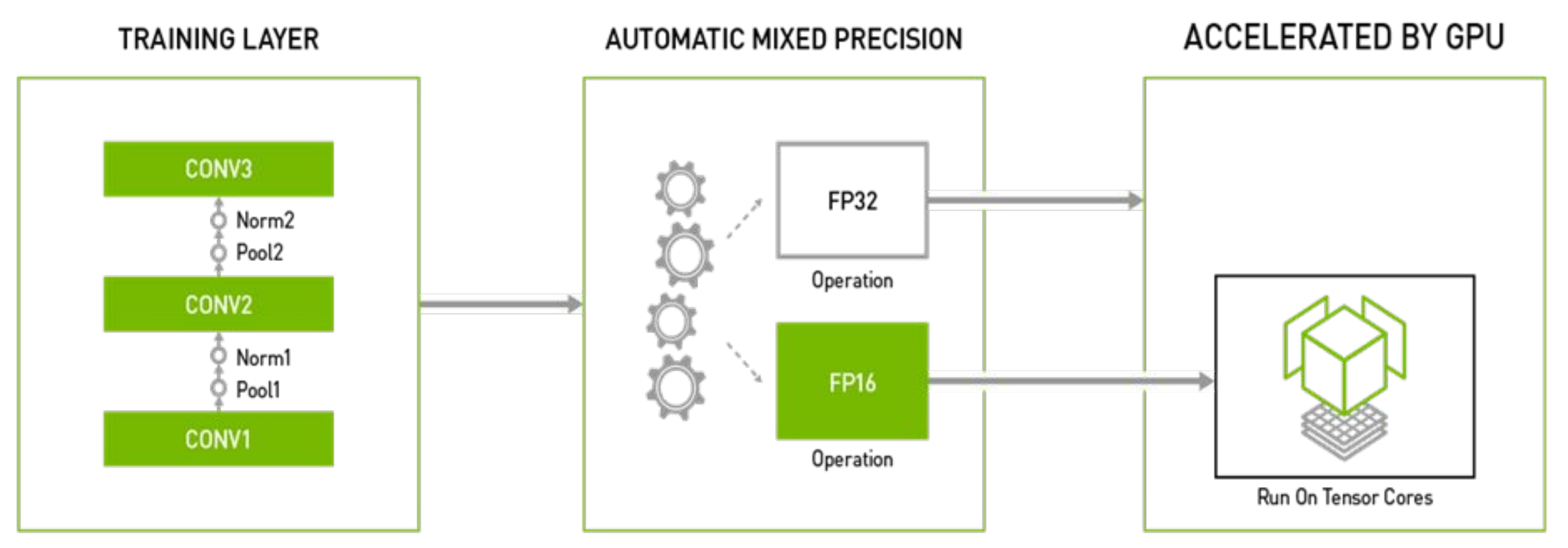


Gradient Accumulation

Methodology

We have experimented with the following optimization techniques provided by Colossal-AI for training the pix2pix GAN model:

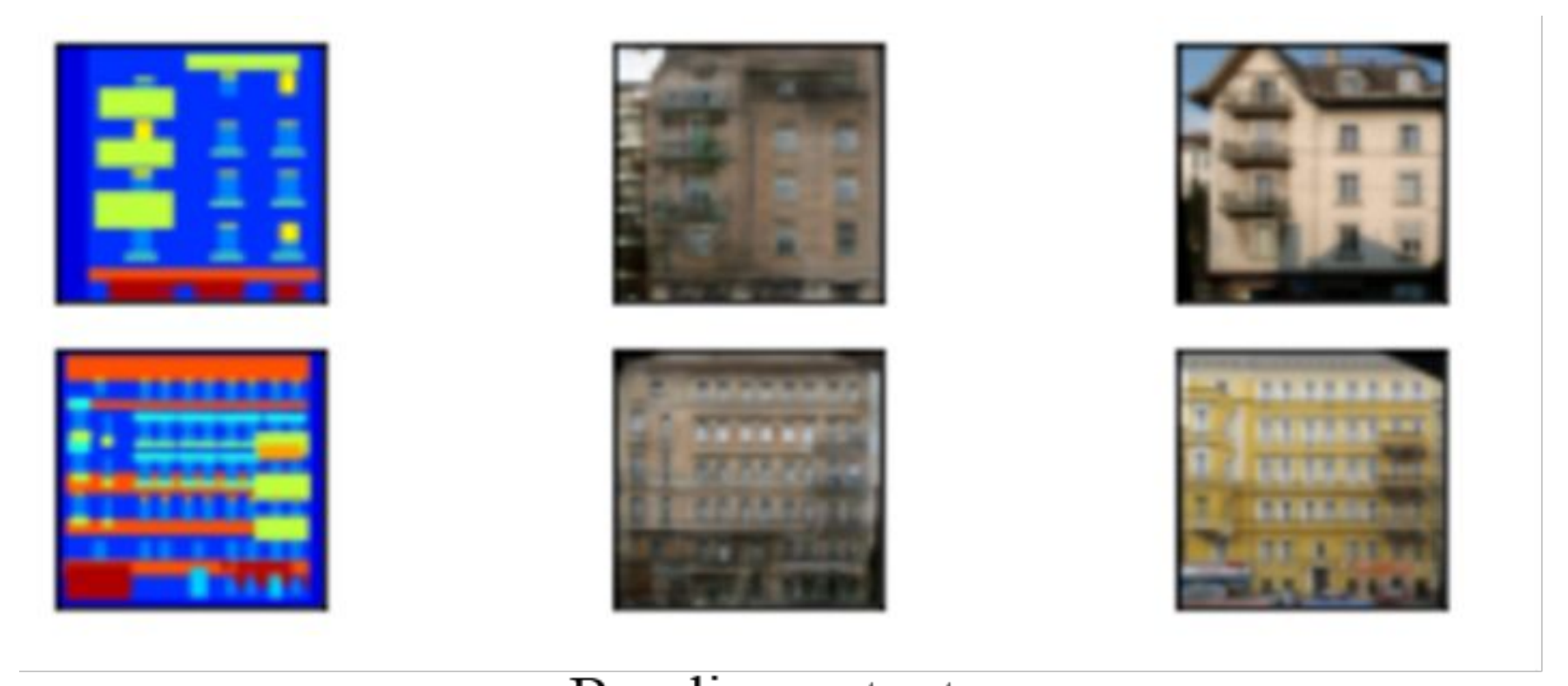
1. Data Parallelism
   - This is the simplest and most common form of parallelism where every GPU gets a copy of the model and trains on the batch of data assigned to it. The gradients are all-reduced to ensure synchronization.
2. Automatic Mixed Precision(AMP)
   - Mixed precision training is an optimization technique used to reduce the memory used to store parameters, as well as provide faster training times. By maintaining certain parts of the model in 32 bit and others in 16 bit, the step time is reduced and performance remains unaffected.
   - Colossal-AI provides automatic mixed precision training, which attempts to match the data types on the operation level. This is provided in three modes - AMP_TYPE.TORCH, AMP_TYPE.NAIVE and AMP_TYPE.APEX. For our experiments, we have made use of the AMP_TYPE.TORCH mode.
3. Gradient Accumulation
   - Fitting large DL models on limited memory is difficult in most cases, so it becomes necessary to use small training batches - leading to slower convergence and lower accuracy.
   - Gradient Accumulation works by updating the network weights in the present iteration by adding the gradients from multiple batches, and in our experiment, across 4 iterations.
4. Gradient Clipping
   - This is a convenient method used to change or clip the gradient vector to avoid Vanishing and Exploding gradient problems. By normalising the gradient, using a norm of 2 in our experiment, the decent pace during model training can be controlled while dramatically decreasing the likelihood of an underflow or overflow.
5. Zero Redundancy Optimizer(ZeRO)
   - Large deep learning models have high order of magnitude in terms of parameters (usually in the range of billions or trillions), which can cause memory redundancies in parallel training. ZeRO removes this problem by partitioning the optimizer states, model and parameters instead of replicating them across parallel processes, as well as offloading optimizer states to CPU to improve training time and efficiency.
   - In our experiments, we use ZeRO with very small batch sizes (1,4,8) and notice that the model training takes longer for smaller batch sizes, as compared to larger batch size.



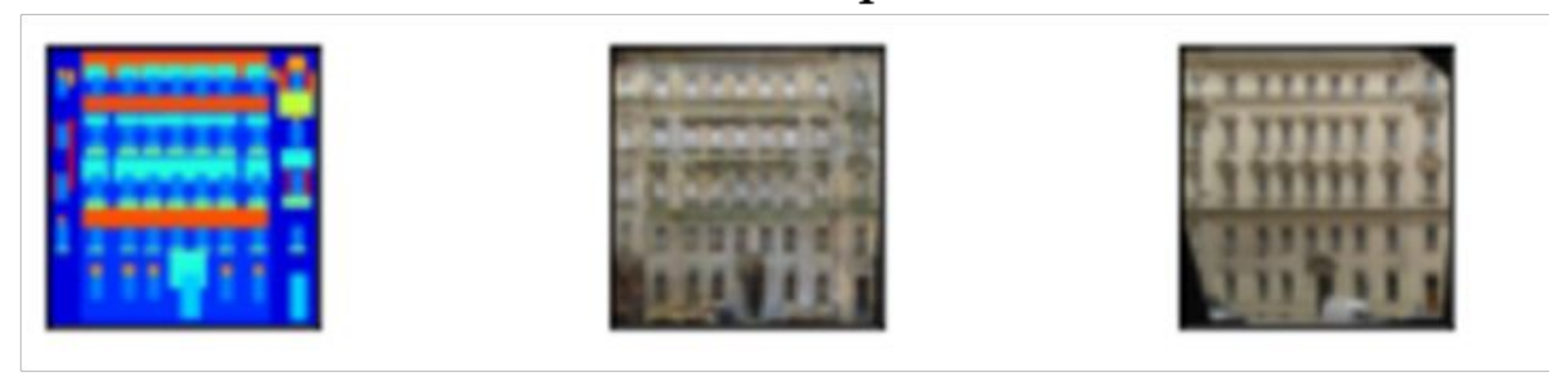
AMP

Experimental Results

Training was performed using multiple optimization techniques for Vanilla GANS and pix2pix model. The following results were observed on training pix2pix model for 100 epochs using a single GPU on facades dataset, with a batch size of 1:

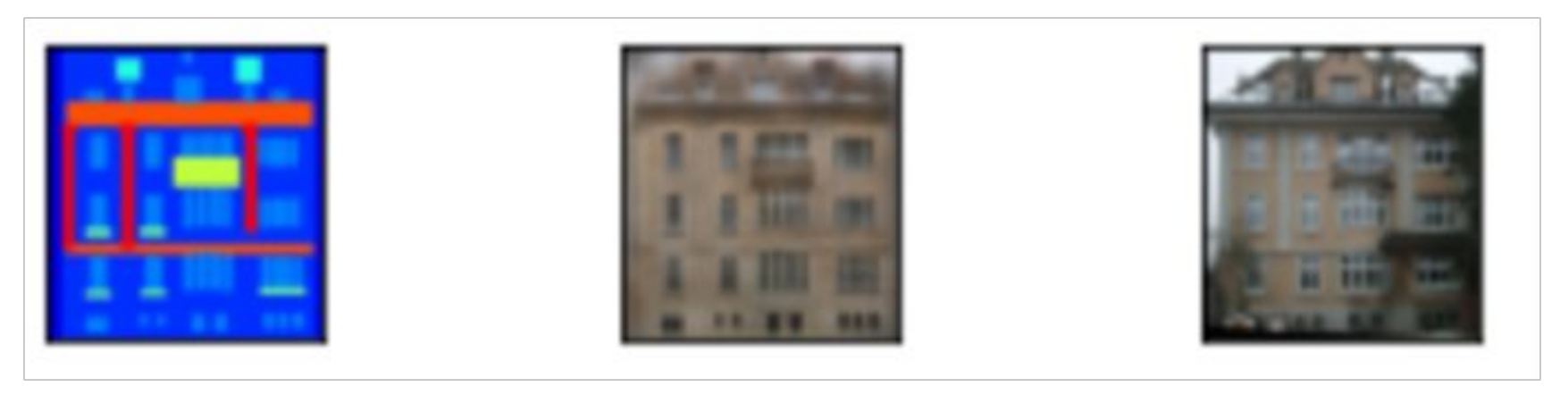


Baseline output

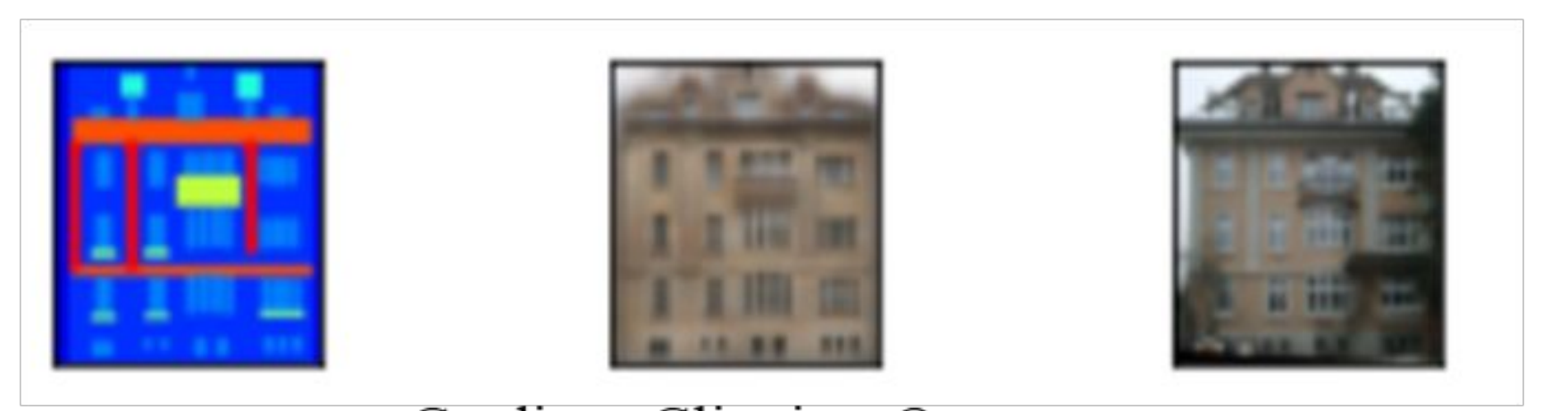


Zero Output

AMP Output



Gradient Accumulation Output



Gradient Clipping Output

Feature	Avg. Epoch Time (seconds)	Discriminator Loss	Generator Loss
Baseline	**40.38**	**0.283**	**25.547**
Data Parallel	37.84	0.288	25.451
AMP	35.04	0.319	24.991
Gradient Accumulation	34.22	0.459	19.803
Gradient Clipping	34.91	0.586	20.010
ZeRO	271.41	0.400	23.391

Table 1: Results: Training time and losses for various optimization techniques