The Journal of Supercomputing (2022) 78:12224–12241
https://doi.org/10.1007/s11227-022-04354-1

Using a Multi-GPU node to accelerate the training of Pix2Pix neural networks

M. Lupión[1] · J. F. Sanjuan[1] · P. M. Ortigosa[1]

Accepted: 3 February 2022 / Published online: 28 February 2022

Abstract
Generative adversarial networks are gaining importance in problems such as image conversion, cross-domain translation and fast styling. However, the training of these networks remains unclear because it often results in unexpected behavior caused by non-convergence, model collapse or overly long training, causing the training task to have to be supervised by the user and vary with each dataset. To increase the speed of training in Pix2Pix (image-to-image translation) networks, this work incorporates multi-GPU training using mixed precision, along with optimizations in the GPU image input process. In addition, in order to make the training unsupervised and to terminate it when the best transformations are performed, an early stopping method using the peak signal noise ratio (PSNR) metric is proposed.

Keywords Generative adversarial networks · Pix2Pix · Multi-GPU · Mixed precision · Early stopping

1 Introduction

Currently, smart homes are gaining popularity especially due to the benefits they bring to the elderly, dependent or people with some kind of disease [1]. The aim is to create a smart environment in which the user can be monitored so that he or she can live independently or as safely as possible [2]. Previously, binary sensors

Juan F. Sanjuan and P. M. Ortigosa have contributed equally to this study.

M. Lupión
marcoslupion@ual.es

J. F. Sanjuan
jsanjuan@ual.es

P. M. Ortigosa
ortigosa@ual.es

1 Group of Supercomputation-Algorithms, Department of Informatics, ceiA3, University of Almería, Carr. Sacramento, s/n, La Cañada, 04120 Almería, Andalucía, Spain

were used due to their low invasiveness [3], in addition to some multimodal sensors such as smartwatches and location sensors [4]. However, to increase the level of monitoring of such users, cameras are incorporated in the visible spectrum, which thanks to algorithms and machine and deep learning techniques [5], allow us to delimit with enough certainty the activity that the user is performing and therefore to detect emergency situations such as falls [6]. The use of these types of cameras can be compromised in terms of privacy, so the use of thermal cameras [7] has been proposed as an alternative. These thermal cameras allow us to obtain thermal images both at night and during the day, so that by using deep learning techniques, applications such as person recognition and identification, fall detection and activity recognition, can be developed.

Generative adversarial networks (GANs) [8] are a type of neural network in which two models are trained together on a zero-sum problem. In them, a network (generator) generates images from noise in order to fool the discriminator (another network), which has to determine if the received image is real or produced by the generator. Within these neural networks are conditional generative adversarial networks (cGANs) [9] (see Fig. 1), whose difference is that the user provides an input to the system and the outcome of this system is dependent on that input. These neural networks have been used in image super-resolution tasks [10], photograph inpainting [11], video prediction [12] and text-to-image translation [13]. Another application within this type of neural network is domain-to-domain translation [14]. Specifically, in Pix2Pix [15], starting from paired images, an image is allowed to translate from one domain to another.

Thus, Pix2Pix has been used in other tasks such as [16] to transform images from the thermal domain provided by thermal cameras embedded in smart homes, to the visible domain, with the goal of obtaining visible images and that such images can be processed in smart environments to recognize activities, falls or identify people. Nevertheless, the training of such neural networks in the Pix-2Pix solution is quite computationally expensive [17] as it involves the parallel

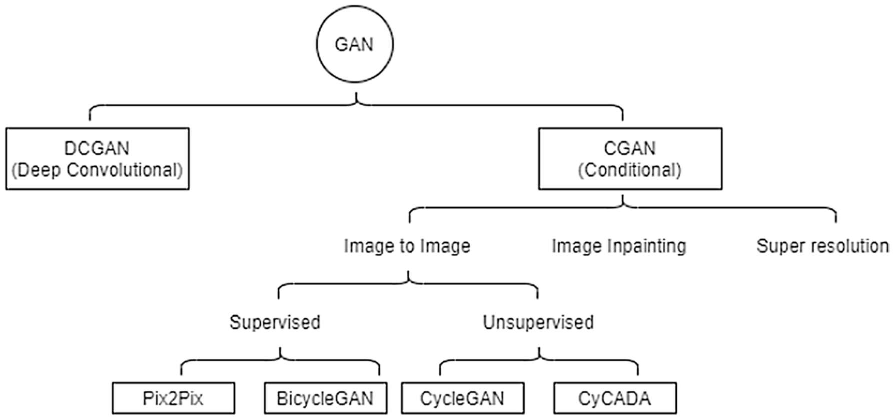

Fig. 1 GANs classification

training of two neural networks. Moreover, the user must decide based on his experience, when to stop training, so such training is supervised and subject to perceptual errors by the user.

In this work we try to improve the training of Pix2Pix, making the following contributions.

- A new early stopping method which calculates peak signal noise ratio (PSNR) on validation images is developed, letting the training be unsupervised, different from original paper.
- The impact of the batch size in training is studied as a bigger batch size lets the neural networks training accelerate.
- Incorporate mixed precision in the training, taking advantage of the Tensor Cores of NVIDIA Volta architectures. Speedup when using mixed precision is calculated in order to check the improvements carried out by this type of computation.
- The different loss functions are parameterized to be able to train in a multi-GPU system using TensorFlow.

In the remainder of this article, we provide further detailed descriptions of the proposed approach. Section 2 reviews related works and the state of the art of similar approaches to improve of GAN training. Section 3 presents the proposed methodology and optimizations carried out. Section 4 introduces the evaluation of the methodology. Finally, in Sect. 5, conclusions and ongoing and future works are discussed.

2 Related works

As for the problem of transforming images from one domain to another, there are a variety of solutions. One of the first solutions is Pix2Pix [15]. In this, it learns to perform the transformation between domains from a pair of images, so it is a supervised transformation (the target image exists). In the same way, BicycleGAN [18] from paired images can generate not only the target image, but it allows us to generate a set of target images with variations between them (for example in landscapes, varying light, sky and clouds). However, it may be the case that the dataset available to perform the transformation from one domain to another does not contain paired images, but images from both domains unrelated. In this case, CycleGAN [19], making use of two generative networks and two discriminator networks, allows it to perform such transformation. To do so, it makes use of a loss function called Cycle Consistency Loss. To calculate this error, the system transforms an image X of domain A into an image Y of domain B. In addition, the other generator network transforms the image Y into an image X' of domain A. This function compares the difference between X and X' in order to check that both generators perform transformations between domain A and B in both directions. The incorporation of this function makes it possible to learn to perform the transformation between domains in an unsupervised way since there are no target images. In addition, the training of this solution, as there are 4 neural networks instead of 2, is more expensive than in Pix2Pix. Similar to CycleGAN, CyCADA [20] incorporates the cycle consistency function. However, CyCADA pays more attention to pixel and feature level, improves the cycle consistency function and optimizes the transformation of synthetic images into real images. Figure 1 shows the classification of the different GANs.

As discussed above, generative adversarial neural networks that perform image transformation tasks contain at least several neural networks, as well as several loss functions. This makes the training of these networks quite expensive. To reduce the training time, an evolutionary algorithm is proposed in [21] that stabilizes the weights in the first iterations to also avoid problems of collapse (the same output is always generated from the input image) and non-convergence (the transformation is never performed correctly). This method achieves faster convergence. However, this solution does not drastically improve the training speed of the system, since it does not exploit the hardware resources that mainly provide speed to the system. In [22] the authors train Lapras-GAN, an image super-resolution method that makes use of SSIM and L1 to obtain realistic images. To perform PyTorch training, several nodes with multiple GPUs are used, reducing the training time by up to 4 times. In VAE-GAN [23], TensorFlow is used to perform multi-GPU training. In short, the incorporation of multiple GPUs allows for a dramatic increase in training speed. However, the solutions mentioned above (Pix2Pix, CycleGAN, BicycleGAN) do not include multi-GPU training.

Thus, to take advantage of the speed in training, it is necessary to increase the size of the training batch. This parameter is very important in training, since a very large batch size makes the training faster, but may cause the model to overgeneralize and not be able to transform image details. In several works, some strategies to vary the batch size depending on the epoch [24, 25] and network layer [26] in non-generative adversarial networks are proposed.

However, the effects on the increase in the batch size in the GANs (neither on the Pix2Pix or CycleGAN) have not been previously studied. In these solutions, a batch size is established in the different papers, but increasing it is not suggested.

3 Methodology

3.1 Pix2Pix

In the original paper where Pix2Pix was introduced [15], a solution is obtained for image translation problems from one domain to another, and the proposed architecture is similar to Fig. 2 where the pair of images provided as input to the system consists of a thermal image (x) and its correspondence with an image in the visible spectrum of a person's face (y). In addition, a neural network called generator, takes as input the thermal image and generates a visible image from it (x′). Subsequently, another discriminator neural network receives the visible image created by the generator and the input visible image, having to decide whether each of these images is true or false. Therefore, to train such neural networks, several loss functions are defined. The first one is the adversarial loss, i.e., the degree to which the

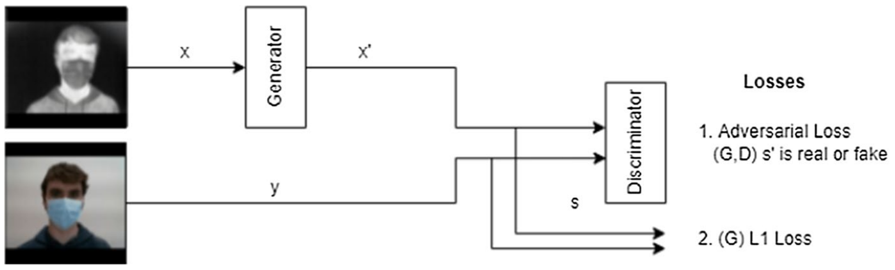

Fig. 2 Original Pix2Pix infrastructure

discriminator network is wrong or not. The second is the L1 loss, which compares the error between the image made by the generator and the input visible image, so that the generating network generates transformations as similar as possible to the initial visible image.

However in this work, in order to improve the level of facial recognition of the images produced by the generator, new elements are added to the Pix2Pix solution (Fig. 3). The person included in the image is added at the input of the system (z). Subsequently, when the generator network produces the visible image, this image becomes the input of a new network called Face Recognizer Network. This network is a pre-trained network on the visible images of the dataset whose objective is to recognize the person in the image (z′). Therefore, face recognition loss function (function number 3 in Fig. 3) is finally included which calculates the images in which the person has been correctly recognized. This loss is therefore incorporated in the generator network, generating as the system is trained images with the faces of the people better defined.

In general, this solution makes use of the Adam gradient optimization algorithm with a learning rate of 0.0002 and with the momentum parameters $\beta1 = 0.5$, $\beta2 = 0.999$. In addition, the data batch size with which the training is performed is 1, following the original paper guidelines. By using several layers of Batch Normalization [27], when the batch size is 1, these layers have the behavior of Instance Normalization [28] allowing a better transformation of the images between one domain and

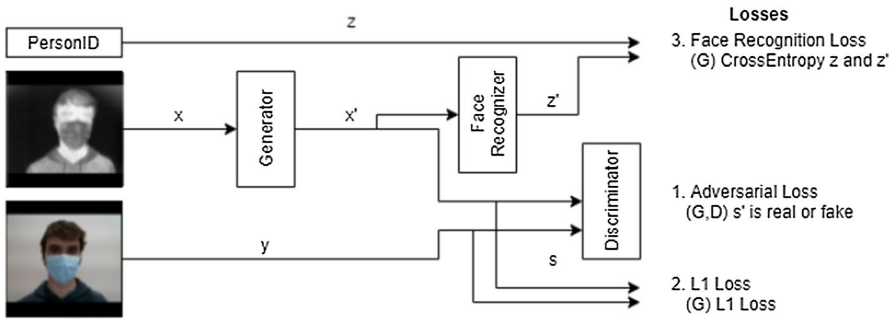

Fig. 3 Custom Pix2Pix infrastructure

another since it removes image-specific contrast information, simplifying the transformation process.

However, in the original approach batch sizes of 1 and 4 are used on different datasets, demonstrating that the use of Instance Normalization is not essential for these networks to work well. Thus, in the following sections several optimizations are performed on the original solution, obtaining better results even at higher speed.

3.2 Early stopping method

One of the problems of generative adversarial networks is about the end of training. Normally, training is performed until the resulting images, to the naked eye, are indistinguishable from the images in the dataset. If training continues for a large number of epochs, training becomes off-target and starts to result in unwanted images, so attention must be paid to this. However, to compare different configurations in the system (different batch sizes, learning rates, number of layers of neural networks) it is necessary to know when each of the models has been fully trained in order to compare them with each other.

Therefore, to detect when the training has to finish, in normal neural networks, the evolution of the loss in the training images and in the validation images is analyzed. Thus, validation loss is observed on new images and at the point where this starts to rise, training stops. The idea was to apply a similar approach to our problem. However, in Pix2Pix, the generator loss does not measure the quality of the generated images, so this metric cannot be used to stop when generated images seem to be real.

Therefore, it is necessary to establish metrics that allow us to know the visual quality of the different images that are generated. In generative conditional adversarial networks, the evaluation of the generated images is an active topic of discussion [17] due to its complexity and different possibilities. In this work, in order to compare the images created by the model with respect to the ground truth images, mean squared error (MSE) [29], peak signal noise ratio (PSNR) [30] and structural similarity index measure (SSIM) [31] were analyzed.

Finally, PSNR metric results to be the most descriptive metric but contains some noisy data. The values in each of the epochs oscillate so it is difficult to determine at which point the best images are obtained and therefore, stop the training at that point.

For this reason, to clearly see the evolution of the PSNR metric, the smoothing of the data is firstly proposed. Thus, being *i* an epoch and *N* the total number of epochs, the PSNR value of *i* is calculated as the average of the *i-50* values, so the first value is obtained at epoch 50. In this way, the highest PSNR value will be given by the previous 50 epochs, when the training is stable and avoids a minimum and also instabilities.

In Fig. 4, the evolution of the smoothed PSNR metric on a 3000 epochs execution can be seen. The highest value is reached at epoch 405 and though similar PSNR values may exist for further epochs, the corresponding images do not improve on images from earlier epochs. Therefore, the stopping criterion for the training is to

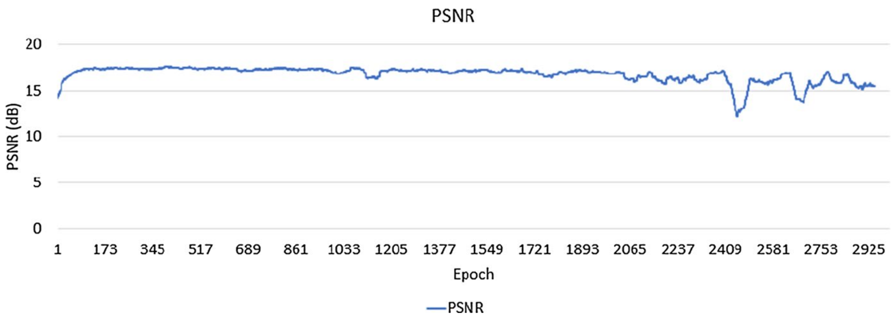

Fig. 4 Smoothed PSNR evolution in validation data

stop when the highest value is obtained and no improvement is found during the 50 following epochs.

In Fig. 5, different transformations can be observed at different points of the training. As can be seen, the best image is obtained in epoch 405, being quite similar to the image of epoch 1620 but almost 1200 epochs before.

In brief, the stopping method calculates the moving average between epochs and stops when the value of the greatest PSNR value is found and no improvements are made after 50 epochs. The 50 epochs results to be a param called *patience* which has to be defined by the user. Therefore, the training becomes unsupervised and only 50 more training epochs in this case are performed from the point at which the system is fully trained, saving training time.

3.3 Optimizations

Starting from the original Pix2Pix solution and the early stopping method defined in the previous section, this section includes different optimizations to the original solution in order to reduce the training time of the system. In this case, only 1 GPU is used for training.

3.3.1 Varying batch size and learning rate

In neural networks, one parameter that can drastically increase the training speed is the batch size. When the batch size of images is 1, in the processing of each image, an update of the neural network weights occurs, which can lead to an overfitting

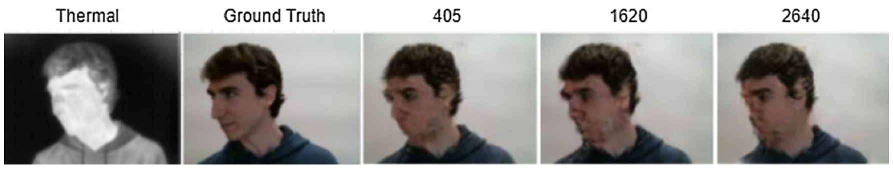

Fig. 5 Image transformation evolution in the 3000 epochs execution

of the network. In case of increasing the number of images per batch, less model updates occur, but it generalizes better and increases the speed of training. However, there may come a point when increasing the batch size too much causes the network to undertrain.

In generative adversarial networks (GANs), the selection of hyperparameters is a task that requires domain knowledge and in most cases, it is a trial and error process [32]. Therefore, there are no references where the impact of the most important hyperparameters such as batch size and learning rate [27] on this type of neural networks has been studied.

In this work, we define 9 configurations, with batch sizes that are powers of 2 [33] (including the default configuration with batch of 1) and with a constant value of learning rate.

After analyzing the results with the different configurations (Table 1), PSNR values between 17 and 18 are obtained in the configurations with batch sizes that range between 4 and 256, allowing the acceleration of the training with respect to the version with a batch of 1, which provide a value 10.71. It can be seen that the best configuration is the one with a batch size of 64, reaching this value at epoch 385. These results differ from the original paper, where the batch size of 1 is used in a dataset of similar number of images.

Figure 6 shows the images corresponding to the epoch with the best PSNR value that are collected in Table 1.

In addition, in some works it is indicated that together with increasing the batch size to improve the training speed, it is necessary to increase the size of the learning rate [24] as well.

Thus, [34, 35] define that when the batch size is increased by k, it is necessary to increase the learning rate by $\sqrt{k}$. On the other hand, in [25] it is stated that instead of multiplying the learning rate by $\sqrt{k}$, it should be simply multiplied by k. In our work, both approaches have been followed to check which method is more suitable in our case.

Figure 7 shows the evolution of the PSNR metric in three configurations with a batch size of 4 with a different learning rate strategy in each. The configuration *Default* has a static batch size, which is the one defined in [15]. In *SQRT*, the

Table 1 Batch size comparison

Batch	Epoch Stop	PSNR
1	131	10.71
2	251	9.89
4	264	17.22
8	224	17.39
16	265	17.52
32	225	17.69
64	385	17.80
128	476	17.57
256	950	17.38

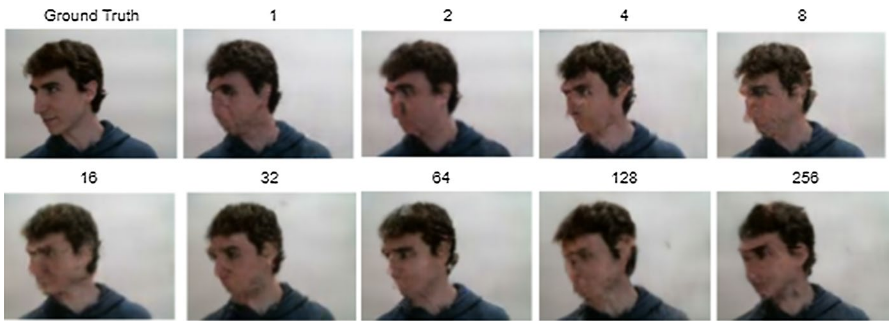

Fig. 6 Generated images in the different configurations

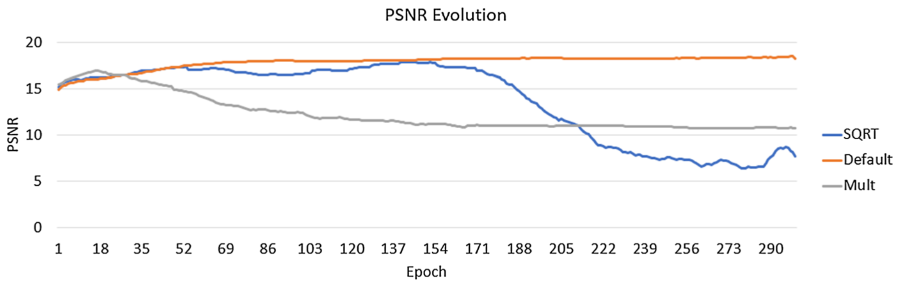

Fig. 7 PSNR evolution with different learning rate strategies

learning rate is multiplied by $\sqrt{4}$ as defined in [34]. In *Mult* multiply, the learning rate by 4 as defined in [25].

In view of the results, the *SQRT* and *Mult* configurations increase the learning speed in the early epochs as the network weights change more strongly. However, they are too aggressive and deviate from the correct path in training later on.

Thus, the default configuration follows a progressive curve obtaining the best PSNR values, so it is concluded that in this case, it is not necessary to increase the learning rate when the batch size is increased.

3.3.2 Input pipeline

The GPU has a large number of cores that allows the image processing to be very fast because it can be performed in parallel between the different cores. The time it takes to process images on the GPU is much less than the time it takes to store the images and exchange the results between the CPU and GPU. For this reason, it is desirable to optimize the process of loading images and exchanging information between the CPU and GPU. This section includes several improvements to the Pix-2Pix input process.

In the original solution, the CPU is in charge of obtaining all the images to be trained and performing their transformations (normalization, resizing) before loading them on the GPU. These tasks are repeated each epoch, wasting CPU resources. Thus, the cache is incorporated so that, once the images are processed, they are stored and available at the next epoch.

In addition, once the system is ready to process a batch, the CPU must process the images and load them to the GPU, so during this time, the GPU is unused and wasting its services. Therefore, a preloading of the images to the GPU has been incorporated into the system, so that the GPU does not have to wait to receive the images from the CPU to start processing.

3.3.3 Mixed precision

Currently, some architectures such as NVIDIA Volta and Turing architectures incorporate Tensor Cores. These are multiprocessors that optimize data processing on the GPU. Tensor Cores also allow calculations to be performed using mixed precision instead of single precision, increasing the processing speed by up to 8x.

Therefore, to take advantage of the potential of Tensor Cores, mixed precision, supported by TensorFlow, has been included in our system. In this precision, the operations are performed in a half-precision way (16 bits) while the models are stored in full precision (32 bits). The incorporation of mixed precision in the training of the system is done in a straightforward way thanks to the libraries provided by TensorFlow. However, some aspects must be taken into account.

First, when using mixed precision, the model weights are still stored in 32 bits, so it is necessary to specify this data type in the model architecture when using TensorFlow.

Secondly, when making use of mixed precision, the calculation of the neural network weights update gradients is done with values in 16 bits. Since sometimes such values are very small, it may be the case that they cannot be represented using only 16 bits, so they become 0. Therefore, it is necessary to incorporate a scaling function that obtains a value from which the gradients are calculated. Once the gradients have been calculated with the scaling factor, this value is de-scaled and applied to the network. In this way, overflow is prevented.

3.4 Multi-GPU training

In the previous section, we developed the improvements that were implemented over the original Pix2Pix configuration. These improvements focus on the optimization of the learning process using a GPU. In this section, the aim is to adapt the system so that the processing is performed on several GPUs in parallel.

3.4.1 Parallelism strategy

Currently, there are several approaches to parallelize training using multiple GPUs. Firstly, there is model parallelization and secondly, data parallelization. In the first case, the model is divided among the different GPUs, so that the model is distributed, processing the batch in a sequential way. In the second case, the model is copied to each GPU, and the batch is divided into mini batches of equal size that are distributed over the different GPUs. In this way, each GPU processes a different batch, and before backpropagation of the data, a reduction of the values of each GPU is performed so that there is only one model which is duplicated in all GPUs.

In our case, the second approach *Data parallelism* is better suited to the problem since the models used are not very large and also because it is the approach currently incorporated in TensorFlow. Specifically, we make use of the *MirroredStrategy* class of TensorFlow.

3.4.2 Losses

To train neural networks, different loss functions are available. These functions work well when single GPU training is used, since no data distribution is performed to train the model. This section explains the loss functions used to train the model and details the modifications made to support multi-GPU training.

Adversarial loss: In the case of the Pix2Pix networks, the generator produces images from the source (thermal) images which are provided to the discriminator as well as the corresponding ground truth (visible) images in order to differentiate between the images created by the generator and the images in the dataset.

Thus, the discriminator has to know how to identify which images do not belong to the Ground Truth, so it should label the images produced by the generator as 0 (false) and the ground truth images as 1 (true). In this case, the discriminator network has output dimensions of 30 ×30, since it is a [36] PatchGAN.

In GANs, the output of the discriminator network is used to train both the generator and the discriminator. The loss function used is binary cross-entropy. This function provides the error between each discriminator network output value and its desired value (0 or 1 depending on each case).

Thus, natively, TensorFlow's binary cross-entropy function calculates the dimensions of the input and batch tensors and averages the probabilities of the output of the discriminator network. However, when making use of multiple GPUs, this aggregation cannot be performed automatically and it is necessary to do the aggregation manually.

To do this, the documentation states that the values must be added and divided by the global batch size. However, it is not taken into account that the output is 30*30. Therefore, to obtain the average value of the loss, it must also be divided by the number of dimensions. This can be seen in Fig. 8, where the comparison between the single and 2 GPUs approach is shown.

L1 loss: In the case of the generator, the loss of the GAN network is obtained through the function described before, as in the case of the discriminator. However, another loss function is included taking into account the MSE. In this case, an aggregation of the mean square errors of each of the pixels is produced and divided by the number of GPUs being used. Once the batch has been processed, the L1 values are summed across the CPU automatically with Keras.

Face recognition loss: In our system, an auxiliary neural network is incorporated to perform a recognition of people in the images generated by the generator. In this way, it is intended to guide the generator so that the images it generates

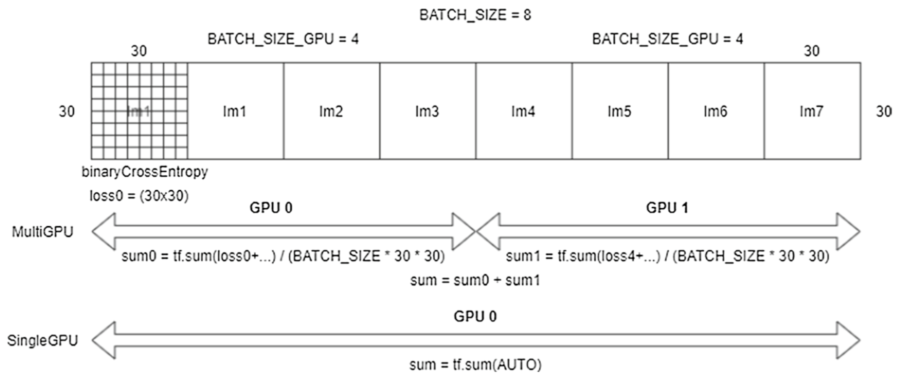

Fig. 8 Adversarial loss

are associated with the correct person in each case. For this purpose, a loss function is incorporated, which is applied on the auxiliary network of person recognition. In this case, instead of making use of binary cross-entropy, sparse categorical cross-entropy is used. The difference between them is that sparse categorical cross-entropy calculates the loss when the number of output elements is greater than 1. In this case, the total number of people to be recognized is 12.

Thus, in the same way as with binary cross-entropy, in multi-GPU the aggregation and loss calculation cannot be performed automatically. Therefore, this function provides a loss value for each image, so in each GPU the sum of the errors of each image is added and divided by the batch size, as shown in Fig. 9.

4 Results

4.1 Infrastructure

The design and testing of the different neural networks have been carried out in the cluster of the University of Almería. In this cluster, a node with 2 NVIDIA TESLA

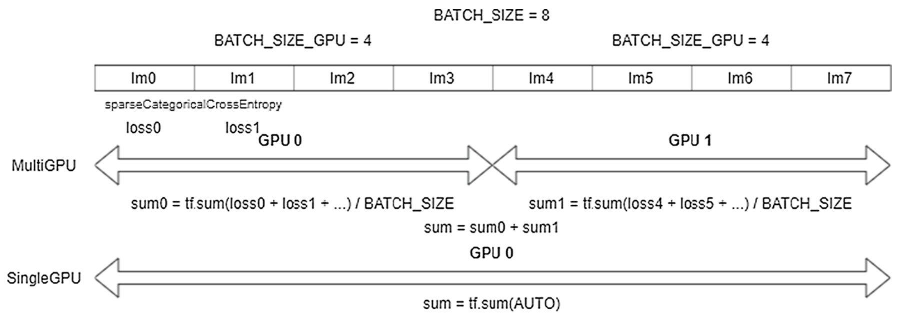

Fig. 9 Face recognition loss

V10 GPUs was used. The operative system of the node is CentOS 8.2 (OpenHPC 2), with DDR4 3200 MHz RAM, and using CUDA version 11.0.2 and TensorFlow version 2.4.1.

4.2 Experiments

Once the optimizations in the input data processing, the mixed precision and the incorporation of 2 GPUs have been made, different runs with different batch sizes have been performed in order to understand and visualize how much the system improves with each of the optimizations.

First, in Fig. 10 the improvement factor of the different optimizations is shown. These optimizations are: input (input pipeline optimization), MP (mixed precision optimization) and both (input and mixed precision optimization). Thus, each of the optimizations has been tested in a single and multi-GPU system. Thus, the improvement factor is defined in Eq. 1. In this Equation, T_1 is the time of the version without optimizations and T_{opt} is the time having the optimizations.

$$\text{Improvement Factor} = T_1/T_{\text{opt}}. \tag{1}$$

Regarding the input pipeline optimization, its speedup is almost negligible independently of the batch size. However, the optimization of mixed precision implies an increase in the speedup when the batch size becomes bigger. Thus, when executing the system with input and mixed precision optimization, it can be seen that on both 1 GPU and 2 GPUs, a value of 2 is obtained when having a batch size of 64, which means that these optimizations reduce the execution time by half on that batch size. In general, the optimizations are more noticeable the larger the batch size.

As for the runs making use of 1 GPU, the total time for the different batch sizes with the different configurations (default, with input pipeline optimization, with mixed precision and with both optimizations) is shown in Fig. 11. It can be seen that the default configuration of the Pix2Pix [15] job takes a total time of 177 seconds each epoch. This time decreases as the batch size increases in each of the configurations. On the other hand, upon reaching epoch 128, a runtime error occurs;

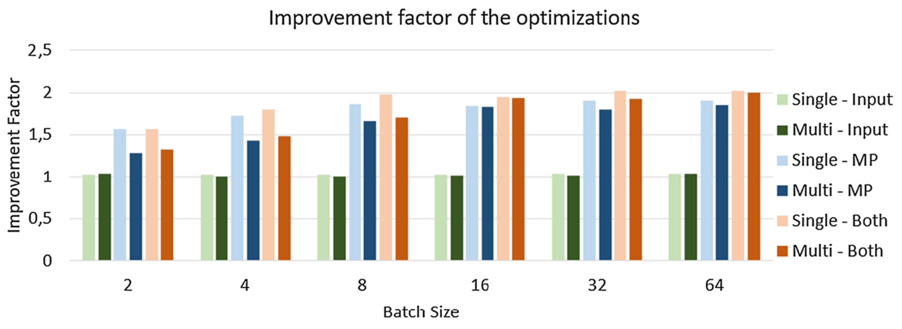

Fig. 10 Improvement factor of the different optimizations

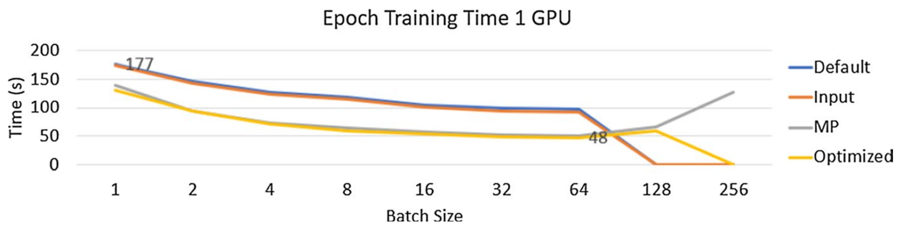

Fig. 11 Epoch training time using 1 GPU

the default configuration and input pipeline optimization cannot be completed due to a memory failure, i.e., the 128 images cannot be loaded into the GPU for processing. However, the mixed accuracy causes the in-memory size of the model and computations to be reduced, allowing processing. As for the batch size of 256, only the configuration using mixed precision provides training. This figure shows that the maximum optimization occurs with a batch size of 64, reducing the epoch execution from 177 to 48 seconds.

As for the runs using 2 GPUs, Fig. 12 shows the same information as in Fig. 11. In this one, it can be seen that as before, the highest optimization occurs at the batch size of 64, reducing from 114s with a batch size of 2 to 25s with a batch size of 64. In the case of a batch size of 256, the default configuration cannot be executed due to a memory failure.

To compare the executions in sequential and parallel and see the acceleration of each of the optimizations introduced, speedup, which is defined in Eq. 2, is used. In this equation, T_1 is the time of the initial sequential algorithm and T_p the parallel algorithm.

$$S_p = T_1/T_p. \tag{2}$$

Finally, Fig. 13 shows the speedup between the optimized runs of 1 and 2 GPUs. As can be appreciated, there is a maximum of 1.92 in batch size 64, reaching almost 2 which is the ideal speedup (halving the execution time by doubling the number of GPUs). In this case, the ideal speedup is not reached because part of the time is spent on the communication between the GPUs and the host CPU.

Finally, comparing the default configuration with 1 GPU, with the batch size of 1 defined in [15] and the optimized version with batch size 64 on 2 GPUs, a speedup

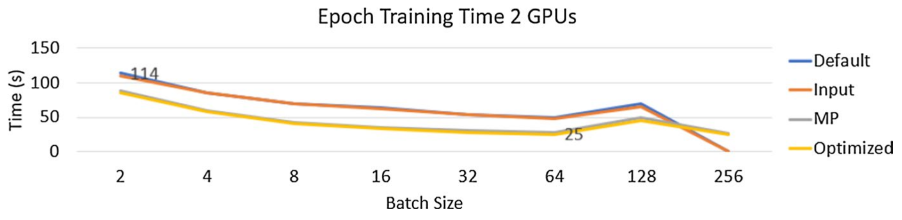

Fig. 12 Epoch training time using 2 GPUs

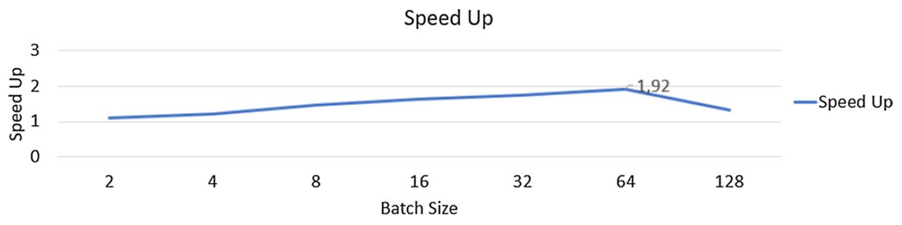

Fig. 13 Speedup

of 7.08 is obtained, so it can be said that the built-in optimizations together with the adaptation to multi-GPU allow to successfully optimize the problem in question.

5 Conclusions and future works

Currently, generative adversarial networks (GANs) are being used in a multitude of applications. Specifically, one use deals with the transformation of images from one domain to another. One solution is Pix2Pix, which performs this transformation from pairs of images. Such a system performs the two-domain image transformation satisfactorily, but the training of such a system is complicated as it is quite expensive in terms of computational complexity and is eminently supervised.

In our work, firstly, an early stopping system has been proposed in which the dataset is divided into a training and evaluation set. At the end of each epoch, the PSNR value of each of the validation images is calculated. Starting from epoch 50, the moving average of these values is performed, saving the model when the PSNR value is higher. If such value does not improve in 50 epochs, it is concluded that such model is fully trained. Therefore, such a stopping method allows us to establish the end of the system training and to be able to compare between different configurations.

Unlike the original work, in our work, the batch size that generates the best PSNR value is 64, using the original learning rate. In addition, tests are performed by changing the learning rate but no improvement is found. Thus, it is concluded that the batch size does not have to be 1 or 4, but depends on the size of the dataset and the type of problem.

On the other hand, we included the use of the cache to store the processed input images before sending them to the GPU, as well as a preload of these images in the GPU. This optimization provides a slight improvement in the speed (speedup is near 1.05).

In addition, mixed precision has been incorporated to exploit the Tensor Cores present in the NVIDIA Volta architecture. By incorporating such precision, it has been necessary to update the models and implement loss scaling of these models in order to avoid overflow. After performing several experiments, a speedup near 2 is obtained.

Thus, combining both optimizations, a speedup of up to 2 is obtained with respect to the default configuration.

On the other hand, multi-GPU processing has been incorporated following the data parallelism approach. To do so, it has been necessary to adapt the loss functions of the system. By incorporating 2 GPUs, the speedup reaches a value of 1.92 when having a batch size of 64.

Therefore, applying the optimizations of the input pipeline, mixed precision and multi-GPU, a speedup of 7.08 is obtained compared to the original paper configuration. Thus, all the optimizations that have been carried out allow converting the training of the system into an unsupervised training, ensuring that it stops at the best epoch.

As future work, it is intended to incorporate a larger number of GPUs to increase the speed of the training and test how much the speedup is. In addition, the system will be adapted to a multi-machine system, each with a different number of GPUs.

Acknowledgements This research has been funded by the R+D+i project RTI2018-095993-B-I00, financiated by MCIN/AEI/10.13039/501100011033/ and FEDER “Una manera de hacer Europa”; by the Junta de Andalucía with reference P18-RT-1193; by the University of Almería with reference UAL18-TIC-A020-B and by the Department of Computer Science of the University of Almería. Marcos Lupión Lorente is a beneficiary of the program “Formación del Profesorado Universitario” with reference number (FPU19/02756).

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

References

1. Mtshali P, Khubisa F (2019) A smart home appliance control system for physically disabled people. In: 2019 Conference on Information Communications Technology and Society (ICTAS), pp 1–5. https://doi.org/10.1109/ICTAS.2019.8703637
2. Stefanov DH, Bien Z, Bang W-C (2004) The smart house for older persons and persons with physical disabilities: structure, technology arrangements, and perspectives. IEEE Trans Neural Syst Rehabil Eng 12(2):228–250. https://doi.org/10.1109/TNSRE.2004.828423
3. Ordó nez FJ, De Toledo P, Sanchis A (2013) Activity recognition using hybrid generative/discriminative models on home environments using binary sensors. Sensors 13(5):5460–5477. https://doi.org/10.3390/s130505460
4. Lupión M, Medina-Quero J, Sanjuan JF, Ortigosa PM (2021) Dolars, a distributed on-line activity recognition system by means of heterogeneous sensors in real-life deployments—a case study in the smart lab of the University of Almería. Sensors **21**(2). https://doi.org/10.3390/s21020405
5. Mehr HD, Polat H (2019) Human activity recognition in smart home with deep learning approach. In: 2019 7th International Istanbul Smart Grids and Cities Congress and Fair (ICSG), pp 149–153. https://doi.org/10.1109/SGCF.2019.8782290

6. Shojaei-Hashemi A, Nasiopoulos P, Little JJ, Pourazad MT (2018) Video-based human fall detection in smart homes using deep learning. In: 2018 IEEE International Symposium on Circuits and Systems (ISCAS), pp 1–5. https://doi.org/10.1109/ISCAS.2018.8351648
7. Gochoo M, Tan T-H, Alnajjar F, Hsieh J-W, Chen P-Y (2020) Lownet: Privacy preserved ultra-low resolution posture image classification. In: 2020 IEEE International Conference on Image Processing (ICIP), pp 663–667. https://doi.org/10.1109/ICIP40778.2020.9190922
8. Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Proceedings of the 27th International Conference on Neural Information Processing Systems, vol 2, pp 2672–2680. MIT Press, Cambridge, MA, USA
9. Mirza M, Osindero S (2014) Conditional generative adversarial nets. arXiv: arXiv:1411.1784
10. Xue X, Zhang X, Li H, Wang W (2020) Research on gan-based image super-resolution method. In: 2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA), pp 602–605. https://doi.org/10.1109/ICAICA50127.2020.9182617
11. Miao F, Feng L (2020) Research on character image inpainting based on generative adversarial network. In: 2020 International Conference on Culture-oriented Science Technology (ICCST), pp 137–140. https://doi.org/10.1109/ICCST50977.2020.00032
12. Liang X, Lee L, Dai W, Xing EP (2017) Dual motion gan for future-flow embedded video prediction. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp 1762–1770. https://doi.org/10.1109/ICCV.2017.194
13. Mishra P, Singh Rathore T, Shivani S, Tendulkar S (2020) Text to image synthesis using residual gan. In: 2020 3rd International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things (ICETCE), pp 139–144. https://doi.org/10.1109/ICETCE48199.2020.9091779
14. Regmi K, Borji A (2018) Cross-view image synthesis using conditional gans. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp 3501–3510. IEEE Computer Society, Los Alamitos, CA, USA. https://doi.org/10.1109/CVPR.2018.00369
15. Isola P, Zhu J-Y, Zhou T, Efros A (2017) Image-to-image translation with conditional adversarial networks, pp 5967–5976. https://doi.org/10.1109/CVPR.2017.632
16. Zhang T, Wiliem A, Yang S, Lovell B (2018) Tv-gan: Generative adversarial network based thermal to visible face recognition. In: 2018 International Conference on Biometrics (ICB), pp 174–181. https://doi.org/10.1109/ICB2018.2018.00035
17. Salimans T, Goodfellow I, Zaremba W, Cheung V, Radford A, Chen X (2016) Improved techniques for training gans. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. NIPS'16, pp 2234–2242. Curran Associates Inc., Red Hook, NY, USA
18. Zhu J-Y, Zhang R, Pathak D, Darrell T, Efros AA, Wang O, Shechtman E (2017) Toward multimodal image-to-image translation. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS'17, pp 465–476. Curran Associates Inc., Red Hook, NY, USA
19. Zhu J-Y, Park T, Isola P, Efros AA (2017) Unpaired image-to-image translation using cycle-consistent adversarial networks. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp 2242–2251. https://doi.org/10.1109/ICCV.2017.244
20. Hoffman J, Tzeng E, Park T, Zhu J-Y, Isola P, Saenko K, Efros A, Darrell T (2018) Cycada: Cycle-consistent adversarial domain adaptation. In: International Conference on Machine Learning, pp 1989–1998
21. Korde CG, Reddy K M, M H, V, Y B, NK (2019) Training of generative adversarial networks with hybrid evolutionary optimization technique. In: 2019 IEEE 16th India Council International Conference (INDICON), pp 1–4. https://doi.org/10.1109/INDICON47234.2019.9030352
22. Huang J, Li K, Wang X (2019) Single image super-resolution reconstruction of enhanced loss function with multi-gpu training. In: 2019 IEEE Intl Conf on Parallel Distributed Processing with Applications, Big Data Cloud Computing, Sustainable Computing Communications, Social Computing Networking (ISPA/BDCloud/SocialCom/SustainCom), pp 559–565. https://doi.org/10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00085
23. Larsen ABL, Sønderby SK, Larochelle H, Winther O (2016) Autoencoding beyond pixels using a learned similarity metric. In: Balcan, M.F., Weinberger, K.Q. (eds.) Proceedings of The 33rd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol 48, pp 1558–1566. PMLR, New York, USA
24. Hu Z, Xiao J, Tian Z, Zhang X, Zhu H, Yao C, Sun N, Tan G (2019) A variable batch size strategy for large scale distributed dnn training. In: 2019 IEEE Intl Conf on Parallel Distributed Processing

with Applications, Big Data Cloud Computing, Sustainable Computing Communications, Social Computing Networking (ISPA/BDCloud/SocialCom/SustainCom), pp 476–485. https://doi.org/10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00074
25. Goyal P, Dollár P, Girshick RB, Noordhuis P, Wesolowski L, Kyrola A, Tulloch A, Jia Y, He K (2017) Accurate, large minibatch SGD: training imagenet in 1 hour. arXiv arXiv:1706.02677
26. You Y, Gitman I, Ginsburg B (2017) Scaling SGD batch size to 32k for imagenet training. arXiv arXiv:1708.03888
27. Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Bach, F., Blei, D. (eds.) Proceedings of the 32nd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol 37, pp 448–456. PMLR, Lille, France
28. Ulyanov D, Vedaldi A, Lempitsky VS (2016) Instance normalization: The missing ingredient for fast stylization. arXiv: arXiv:1607.08022
29. Sammut C, Webb GI (eds.) (2010) Mean Squared Error, pp. 653–653. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-30164-8_528
30. Horé A, Ziou D (2010) Image quality metrics: Psnr vs. ssim. In: 2010 20th International Conference on Pattern Recognition, pp 2366–2369. https://doi.org/10.1109/ICPR.2010.579
31. Wang Z, Bovik AC, Sheikh HR, Simoncelli EP (2004) Image quality assessment: from error visibility to structural similarity. IEEE Trans Image Process 13(4):600–612. https://doi.org/10.1109/TIP.2003.819861
32. Ghosh B, Dutta IK, Carlson A, Totaro M, Bayoumi M (2020) An empirical analysis of generative adversarial network training times with varying batch sizes. In: 2020 11th IEEE Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON), pp 0643–0648. https://doi.org/10.1109/UEMCON51285.2020.9298092
33. Radiuk P (2017) Impact of training set batch size on the performance of convolutional neural networks for diverse datasets. Inf Technol Manag Sci 20:20–24. https://doi.org/10.1515/itms-2017-0003
34. Krizhevsky A (2014) One weird trick for parallelizing convolutional neural networks. arXiv arXiv:1404.5997
35. Keskar N, Nocedal J, Tang P, Mudigere D, Smelyanskiy M (2017) On Large-batch Training for Deep Learning: Generalization Gap and Sharp Minima. 5th International Conference on Learning Representations, ICLR 2017 ; Conference date: 24-04-2017 Through 26-04-2017
36. Li C, Wand M (2016) Precomputed real-time texture synthesis with markovian generative adversarial networks. In: Leibe B, Matas J, Sebe N, Welling M (eds) Computer vision—ECCV 2016. Springer, Cham, pp 702–716