Pushing the boundaries of domain adapted GAN

Md Omor Faruk
Electrical and Computer Engineering Department
University of Waterloo
Ontario, Canada
mofaruk@uwaterloo.ca

***Abstract*—The idea and paradigm of Generative Adversarial Network can be task specific- domain independent or vice versa. As a common terminology, domain adaptation is investigated here in this paper. Although, single way training from generator to target seems inferior to both way training from domain adaptation, it is better to stay with single way training due to simulation easiness and less complexity. This paper discusses few boundary conditions which is worthwhile to address for single way domain adaptation technique and hence, pushing the boundary of domain adapted GAN in the extent of modified image size and generator structure. The basic blocks of primary pixelda GAN is exploited and extended and analyzed with proper clarification. Extending the Generator architecture can have positive or adversarial effect depending upon input image size and noise vector, which is the chief impetus of this work. Furthermore, as a result, this redundant experiment is purported to open a horizon for early stage researchers who wishes to spend some time with GAN with limited resources, but getting early results.**

***Index Terms*—Domatin Adaptaton, Deep Learning, residual network in GAN, pixelDA GAN, SBADA GAN.**

I. Introduction

The ascendancy of neural network was unprecedented, yet well understood now throughout the vast amount of applications I have now a days. It ranges from image acquisition [**?**], detection [**?**], action and gesture detection in video recognition [**?**]. The history of neural network and multi level perceptron model [**?**], is known to all and became a household name for modern Artificial Neural networks. Interestingly, many of modern AI inventions were introduced during short term internships or during PhD thesis times. Among them, Generative Adversarial Network (GAN) is noteworthy, which was introduced by Ian Godfellow et. Al [**?**], while doing an internship at Google. Since its inception, GAN has become a role model for modern guided [**?**], or un-guided [**?**], image inpainting, comparing real-to fake [**?**] images. Its even widely spoken that its not that far when people will have to spend money to visit Louvre Museum to view GAN based images, which would be equally reasonable and precise as the real ones.

Among the various applications of GAN, domain adaptation is one of the most widely spoken one [**?**]

When two different domain stuffs such as painting and real photos, it is either to translate one into another, or it is even possible to find a common domain between those two [**?**]. And interestingly, transferring domain (source) to another domain (target) may induce label discrepancy, which means, source has labels but target has absence of that- which is known as unsupervised domain adaptation [**?**]. Due to the wide adaptation and tremendous prospect in various applications [**?**], this is the field of interest of this report.

CoGAN [**?**], is reportedly known as the inception of GAN based domain adaptation, where two neural networks i.e. generator and discriminator keep trying to reach towards a target domain where the output of the generator tries its best to match with source images, hence fooling the discriminator, which itself is another convolutional neural network.

Cycle consistency loss [**?**] was introduced and used as an optimization target at Cycle GAN [**?**], paper, which is treated as a landmark of domain adaptation. In this paper, not only source to target, but also the contrary was stressed to find optimal cycle consistency loss and getting an identity map. In pix2pix [**?**], same image and sketch of a cat were used to train the network and optimizing the loss. DiscoGAN [**?**], used cross-entropy loss [**?**] and found consistent results in quest of transforming one source dataset into a different target.

Interestingly, what all of those tasks omitted (probably it was not necessary for them), was introduced in SBADA-GAN [**?**], to add a classifier at the end of the network, hence, predicting the labels of both source and target- which was eventually transformed. Their goal was to back up the classification loss obtained from the pseudo labels assigned to the targets.

In the following parts, this report will try to assert the necessity and way of producing synthesized dataset for further research, along with the theoretical details. Moreover, pushing the synthesized data generation performance needs some experimental planning, which is also explained in the experiment design part. Furthermore, results are compared with the present documented publications. Finally, this report concludes with the possible ways of improving the present work.

It is noteworthy that, the main purpose of this report is to investigate a mechanism for early stage researchers to get an idea whether they should go with domain adapting technique or go for synthesized dataset for their particular tasks.

II. Problem statement and rooms of improvement

Although, all of the aforementioned techniques produced some staggering results in terms of training, accuracy and loss optimization, the real challenge relied with available training data made the research pretty challenging [**?**]. COCO [**?**], ImageNet [**?**], Pascal [**?**] are the most widely used datasets while someone wants to use pre-trained large dataset model for a particular task. One way of avoiding collecting large labeled dataset is to use synthetic data [**?**], where simulated

images are pretty close to the original real counterparts, which could be trained in adversarial networks. The imaging technique used for creating games could bolster producing synthesized datasets, but unfortunately, synthetic data are reportedly less performer in generalizing real images [?]. Even the domain adaptation techniques used in aforementioned works are limited to some extend in terms of repetitive training. To overcome the shortcoming, pixelDA GAN came up with a profound way of unsupervised domain adaptation [?] which can equally achieve independence from task specific architecture, interpretability, data augmentation and higher training stability. They also claimed their work helpful towards both image classification and object detection.

The mechanism can be improved by utilizing a deeper network as generator along with weight standardization [?], [?]. Furthermore, rather than using constant learning rate, cyclic learning rate can also enhance the capability of generalizing synthetic images, which will eventually be helpful for unsupervised modeling. Finally, the outcome is planned to compare with existing state of the art in terms of task performance accuracy and target accuracy. Also, increasing the iteration could enhance the performance in a substantial margin. It is expected that, it will help the network to reach a smooth learning and early reaching to local minima, hence producing better optimization ability.

III. Theoretical Background

Generative adversarial network [?] is a broad field with a wide variety of applications including guided image inpainting [?], domain specific target image or dataset achievement [?], fake image producing from the real one [?] and many more. The basic building block leverages the concept of two different convolutional neural network (CNN) [?] named as generator and discriminator. A random noise [?] is used to develop or generate images which is not the exact one, but more or less the replica of the original image. The combination of random noise and an input image can guide a convolutional neural network to produce images which can generate synthetic dataset, which is pretty closer towards the original one [?]. There has been a wide industry level requirements on this are which will be discussed later in this report.

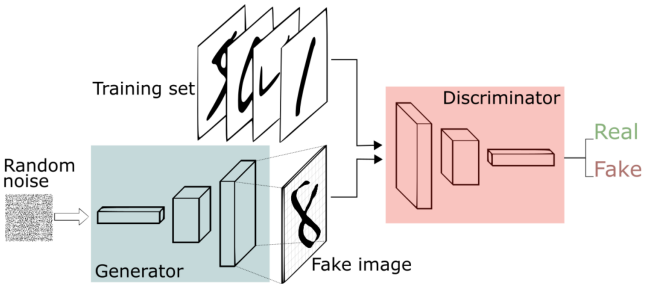

Fig. 1: Basic GAN architecture [?]

The basic architecture of GAN has been the chief impetus of many researchers and exploited into various intermittent tasks. For example, during training, High layer weights are shared with the source, while the low layer weights are shared with the discriminator [?]- to achieve unsupervised domain adaptation. Some other works such as super resolution GAN [?] used a deep neural network with another adversarial network and finally produced super resolution images, which is much better version of the regular blurry or even less quality images. As the theoretical explanation of GAN is quite big in general, I will stress only the parts relevant to this report- specially adjacent to the pixelDA GAN [?].

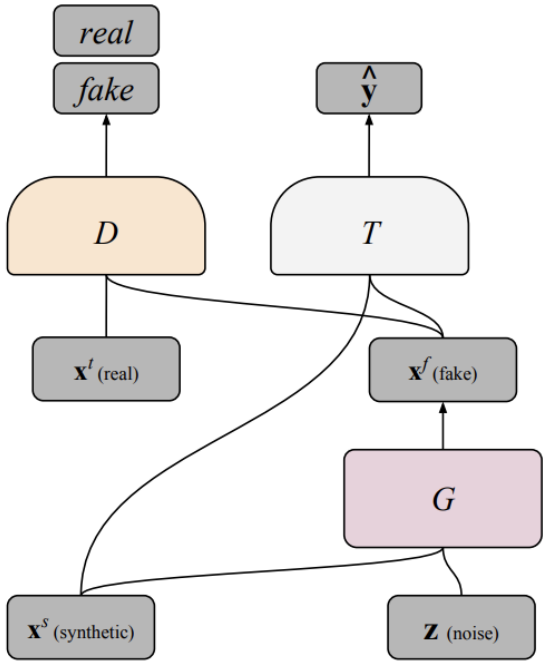

Fig. 2: Source and target domain matching while training [?]

A random noise is generated at the beginning to produce fake images and in parallel, training dataset is fed into another network, finally, they both go through a discriminator network, which decides the image as real or fake. The Generator is fed by random noise and a synthesized source vector to generate fake images. The target domain, denoted as T assigns labels and stores to a vector. As usual, the discriminator will take original and generated images as input and determines real and fake images. It is noteworthy that, the target is task specific and domain independent. Which is definitely a unique feature of this model.

Figure-2 shows a pixelDA GAN basic flow diagram where keeping the source dataset intact was the primary goal. Moreover, dataset was trained in a way that it only changes the source as if it samples from the target domain, hence reflecting

a day of indirect usage of both source and target domains [?]. Figure-3 shows the basic block of the source and discriminator of the network. Generator combines the input random noise vector with synthetic source domain vector via a fully connected layer (noise part) and combines them with strided convolutions and subsequently followed by inducing non-linearity into the dataset via leaky relu. Adopting from the resnet structure [?], few residual blocks were used to combine convolved and batch normalized [?] data with the input via additive operation.

This unique process also reflects the advantage in several ways. Firstly, for different task specific portfolios, it might be necessary to re-train the network to achieve target specific goals. But in this pixelDA structure, pixel level information is mapped so that to achieve a different task specific architecture, re-training the model would not be necessary.

Secondly, a task specific loss is introduced to instantiate the robust training stability as the loss optimization from both train and source end would result a reduced variance for same randomly initialized noise [?].

Thirdly, one of the greatest benefit of using pixelDA is to leverage the higher potentiality of unlimited amount of data augmentation by fusing noise vector to source and target domain.

Last but not the least, a domain adapted image- in this case pixelDA output -is way more easily interpretable than a feature vector adopted in other methods [?].

Fig. 3: Comparing original and synthetic images with pixel to pixel constraint in the background [?]

This model starts with a goal to target domain generalization based on a labeled source dataset. As mentioned earlier, this model is not task specific as no single network is being used for multiple assignment of adapting the domain and classifying an image. Therefore, it seems to show the network that the images are primarily sampled from the target domain, even though primary adaptation happened from the target domain. It is noteworthy that, most GAN are based on conditioning the input noise vector, but in contrary, pixelDA [?] introduces a condition on both noise and source domain labeled dataset. For more detail, it is better to generalize that formally.

A mapped output X^f is receiving from source dataset X^s via $G(\mathbf{x}^s, \mathbf{z})$, where $\mathbf{z}$ denotes the noise vector and X^f becomes the newly found image. Initially, the instantiated model can be influenced by a discriminator function $D(\mathbf{x}; \boldsymbol{\theta}_D)$, via which it is possible to realize that the output X^f is not formulated by source domain, rather it is sampled from the target domain.

As always, the differentiator between real dataset X^t and fake/newly produced dataset X^f is the Discriminator, which is also a convolutional neural network. Moreover, for new task assignments, a task centric tag or label y hat can add a value to the model as a class-shifter [?]. In this way, a general formulation of the domain independent but task specific tagging is perfectly done by the model. It is widely spoken that the formulation and success of neural network based applications are mostly incumbent upon optimization problems [?] and hence, pixelDA GAN is of no exception. It has hefty amount of objective functions, culminating loss functions which needs to be optimized during training the network. A summarized version of those functions and formulas are denoted below according to [?]:

The initial optimization goal is:

$$\min_{\boldsymbol{\theta}_G, \boldsymbol{\theta}_T} \max_{\boldsymbol{\theta}_D} \alpha \mathcal{L}_d(D, G) + \beta \mathcal{L}_t(G, T) \tag{1}$$

The coefficients are weights here for loss control and Ld is the loss in domain:

$$\begin{aligned} \mathcal{L}_d(D, G) = & E_{\mathbf{x}^t}\left[\log D\left(\mathbf{x}^t; \boldsymbol{\theta}_D\right)\right] + \\ & E_{\mathbf{x}^s, \mathbf{z}}\left[\log\left(1 - D\left(G\left(\mathbf{x}^s, \mathbf{z}; \boldsymbol{\theta}_G\right); \boldsymbol{\theta}_D\right)\right)\right] \end{aligned} \tag{2}$$

The task specific loss is denoted as Lt. In-case of classification task, softmax cross-entropy loss is used:

$$\begin{aligned} \mathcal{L}_t(G, T) = E_{\mathbf{x}^s, \mathbf{y}^s, \mathbf{z}}[& -\mathbf{y}^{\mathbf{s}\top} \log T\left(G\left(\mathbf{x}^s, \mathbf{z}; \boldsymbol{\theta}_G\right); \boldsymbol{\theta}_T\right) \\ & - \mathbf{y}^{\mathbf{s}\top} \log T\left(\mathbf{x}^s\right); \boldsymbol{\theta}_T] \end{aligned} \tag{3}$$

Image adaptation, specially at lower level [?] can have significant advantages and properties while rendering new images from the sources. For example, after taking the pixel wise information from the source dataset, if somebody would like to render some fresh images keeping one portion of the image or even more specifically, keeping hue of a certain part of image same but different in opposite end of the image, the best solution would be to use a rendered mask [?]. And that mask can be targeted as an optimization parameter where the goal can be understood via rewarding and penalizing accordingly. Therefore, if a certain portion of image is not rendered as expected, it can be optimized. In that way, the process of generalization can be written as:

$$\min_{\boldsymbol{\theta}_G, \boldsymbol{\theta}_T} \max_{\boldsymbol{\theta}_D} \alpha \mathcal{L}_d(D, G) + \beta \mathcal{L}_t(T, G) + \gamma \mathcal{L}_c(G) \tag{4}$$

Here, the new term Lc is denoted as content similarity loss and all other coefficients are weights.

For a better optimization, masked pairwise mean square error (PMSE) [?] is used in pixelDA for proper formulation of actual and synthetic images.

$$\begin{aligned} \mathcal{L}_c(G) = E_{\mathbf{x}^s, \mathbf{z}}\Bigg[& \frac{1}{k} \left\| \left(\mathbf{x}^s - G\left(\mathbf{x}^s, \mathbf{z}; \boldsymbol{\theta}_G\right)\right) \circ \mathbf{m} \right\|_2^2 \\ & - \frac{1}{k^2} \left(\left(\mathbf{x}^s - G\left(\mathbf{x}^s, \mathbf{z}; \boldsymbol{\theta}_G\right)\right)^\top \mathbf{m} \right)^2 \Bigg] \end{aligned} \tag{5}$$

There is a significant advantage of using this loss, as because it actually helps the model holding the quality of the pixels by doing pixel-wise comparison between two pair of pixels.

As a result, the color and quality of the image can be retained in a controllable manner via optimizing the aforementioned equation. Last but not the least, synthetic linemod dataset [?] was used to render some images in the mentioned pixelDA paper by optimizing a task specific loss, which is not my field of interest now. But for a better understanding, figure-5 can be a reference, where synthetic dataset is generalized from the linemod [?] dataset.

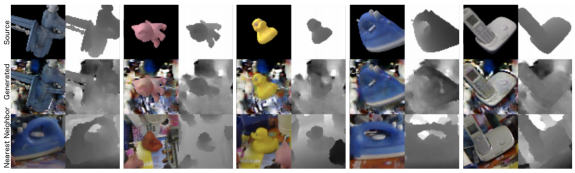

Fig. 4: Generalization capability of synthetic cropped image [?]

IV. Network Structure

As usual, a traditional GAN must have a generator and discriminator, which is depicted in figure-5. But the unique property of pixelDA GAN is having noise vector along with a substantial amount of training data, which is mentioned earlier in this report. A fully connected layer concatenates those and extend a channel with the input image. Subsequently, initially a nonlinear activation function is used to induce non-linearity in the data. For solving vanishing gradient [?] issue while back propagation, it uses a resnet [?] type structure at the generator which holds 6 residual blocks initially. To make the output differentiable and classify it properly, hyperbolic tangent function is used at the end.
The discriminator is another convolutional neural network [?] passing the images through several layers along with batch normalization and a sigmoid function at the end to classify the image as a real or fake one.

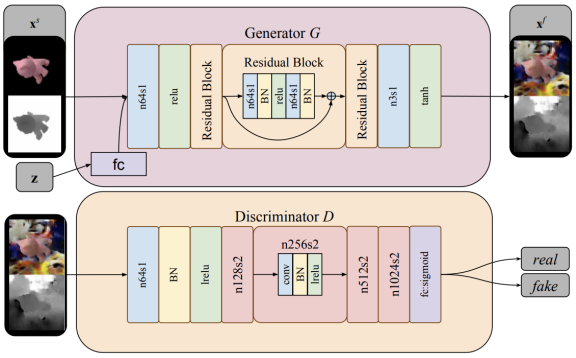

Fig. 5: pixelDA GAN inner network architecture [?]

V. Experimental details

The main idea of this report is to perform some kind of stress check to the Discriminator while moulding the Generator structure and input dataset and in this case, images.

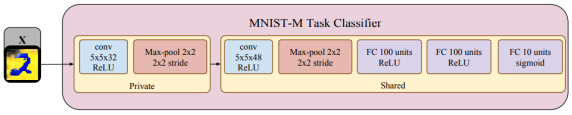

Fig. 6: Image Processing with MNIST-M Classifier [?]

Therefore, to check the extent of the entire network performance, it is purportedly devised as a fixed the Discriminator structure. For input dataset, MNIST [?] was picked up for better comparison, with random cropping [?] as data augmentation along with horizontal flipping was used. As a widely explored dataset, MNIST is a combination of 60k training and 10k testing images of handwritten digits [?]. The initial noise vector z was 10 and similar to the paper [?]. The input data or in this case image was made zero centered and normalized to [-1,1] for easier processing. It helps the dataset to hold a normal distribution and purported to earn a better learning layer wise.
The idea behind using multiple residual block is to reveal more features along with the increased or decreased input image shapes. A decrease in input shape could affect in a very adverse way, but a better comparison and clarity, it was decided to keep here.
Moreover, leaky relu is used in almost all the part of the existing reference paper [?], where in this report it is changed as relu for a clearer comparison. It might be interesting to clip the negative parts and have some intuitive understanding in terms of accuracy.
As a result, the generated image would be reshaped, containing more textures and hues, which would be optimized by equation-5. The generated image would be similar to the reference paper [?] named as MNISTM, which is a modified version of the source dataset [?].
It is noteworthy that I have kept the target structure [?] constant too, for stressing the generator part only. Following table shows the combination of the actual experimental sets. All blocks are preceded by relu activation rather than leaky relu.

TABLE I: Experimental plan with Generator

Image size	No. of residual blocks
32*32	six
64*64	six
16*16	six
32*32	Twelve
64*64	Twelve
16*16	Twelve

VI. Results and Discussion

The results of those experiments are reflected in target and task performances with various combinations. Initially, the intended residual block was 6 according to the reference paper

[**?**] and I started with reproducing that keeping the input image size to the generator at 32*32. Afterwards, with same block size, input image size was varied and task performance was measured.

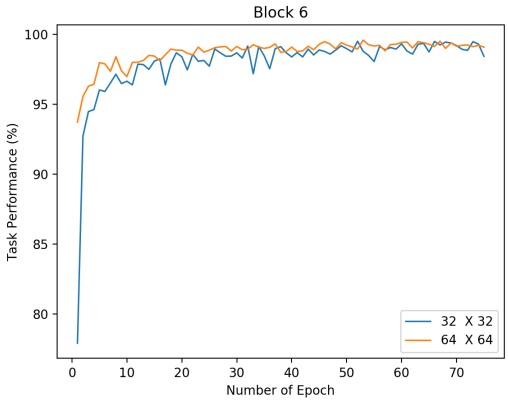

Fig. 7: Task performance of two image size for block 6

It is clearly visible that, a slight improvement in task performance is evident with image size 64*64. Definitely, the up-scaled image size made the task easier to measure. It is noteworthy that, the whole network is domain independent, so without further re-training, one can assign task specific performance analysis. At the moment, the image synthesis is a task along with assigned unsupervised target labels.
Furthermore, my interest shifted towards increasing the block-size, so that the increased image size can be better adopted.

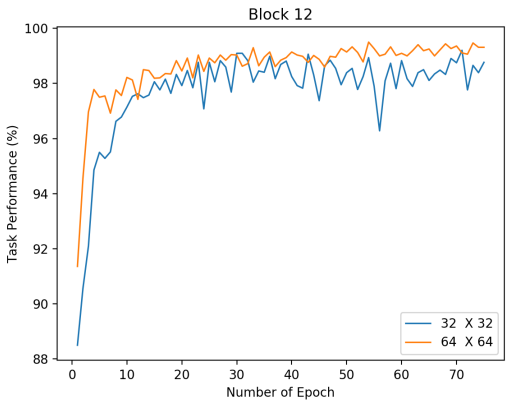

Fig. 8: Measured task performance for block 12

From the above figure, it is visible that increasing block size increases the task specific performance with a slight margin and after around 12 epochs, it was constant. And it is conceivable that, for very sophisticated tasks, even a slight marginal improvement is applaudable.
Moving swiftly on, the target performance was also analysed in a same way where the initial image size with block 6 was compared with the ones with increased block size.

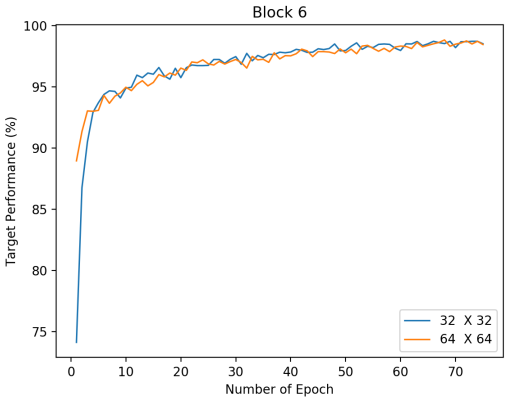

Fig. 9: Target performance with block 6

Fig-10 indicates that the target performance for both sized images for block 6 was almost similar, having no improvement in sized image. But it might be different for increased block size in the generator, which is depicted in the following figure.

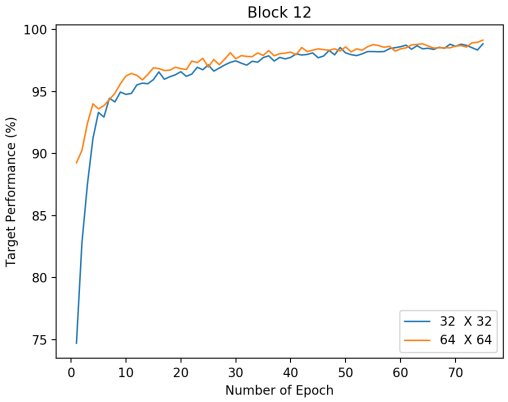

Fig. 10: Target performance with block 12

Above figure shows that increasing block size enhances the target performance by substantial margin throughout 70 epochs. The reason being that could be, larger input image combined with input noise needs to be learnt through out a deeper network with proper residual connections, which not only prevents the vanishing gradients, but also helps the generator producing a synthesized image properly.
As an apple to apple comparison, I were interested towards checking the situation with a single sized imaged with different generator residual block sizes. Following figure-11 indicates that, for a size of 32*32 image, changing the block from 6 to 12 wont create any kind of different. It is pretty straightforward that for this particular image, the generator already generalized that withing a shorter amount of residual blocks. And eventually, same as before, the task performance was pretty same or saturated after 12-15 epochs. But for simplicity and similarity with other experiments of this report, I continued with higher epochs. Perhaps, it can reveal something about

any trend between target and task performances along with increased block sizes. Moreover, this generalized trend can be useful for researchers to generate synthetic data for task specific applications.

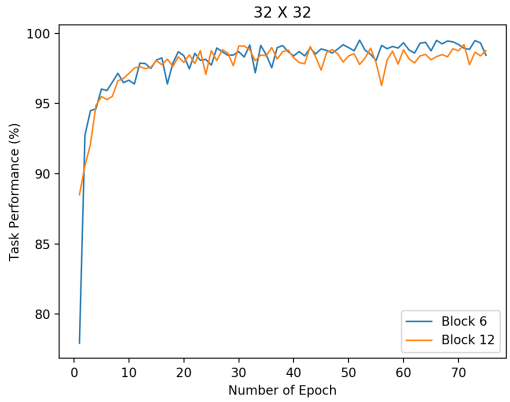

Fig. 11: Task performance for 32*32 images with different blocks

Being curious from the above figure, a comparison between different block sizes for higher sized image is performed also. The following figure-12 shows that even the input image size is 64*64 and block size increased from 6 to 12, task performance remained similar from beginning to end for each block size. But a point to be stressed here is that, the task performance magnitude is higher for higher sized image compared to figure-11.

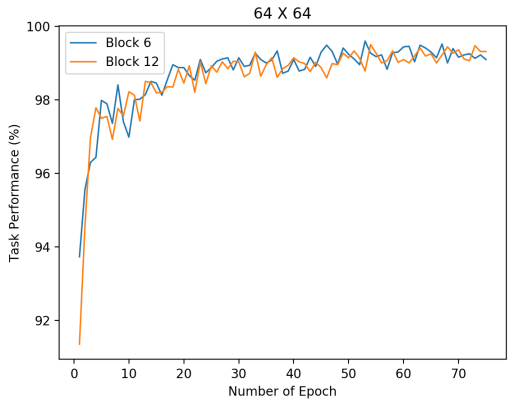

Fig. 12: Task performance for 64*64 image

Similarly, same analysis was purported to be considered for target performance, with exact fashion. The target performance of 32*32 image is depicted in figure-13. As a result, it is clearly visible that the target accuracy was unchanged for both number of blocks and the reason goes to same as aforementioned. So a conclusion can be drawn from here that the smaller image size is not beneficial or impact worthy in quest of achieving higher task performance even with an increased block size. Therefore, decrease image size from this point is not worthy at all but increasing the image size along with increased blocks might still add some values there.

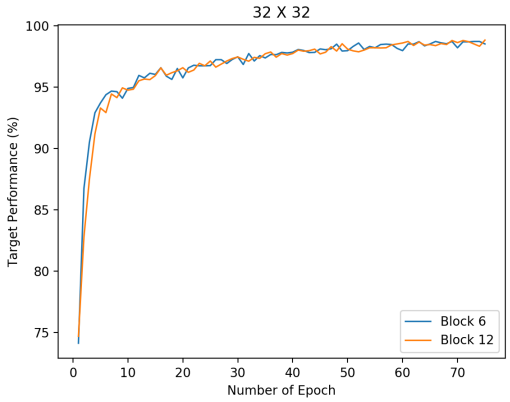

Fig. 13: Target performance for 32*32 image

For the final part of this analysis, I tried with 64*64 image and found that the target performance improved for higher number of blocks in the generator.

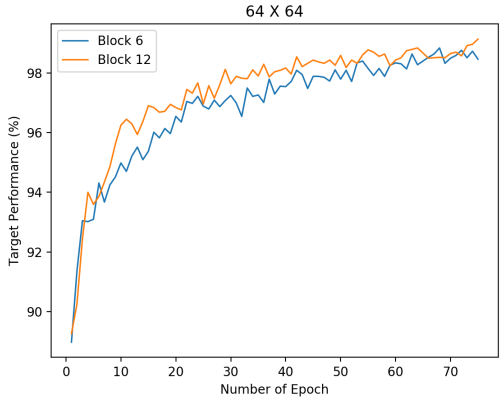

Fig. 14: Target performance for 64*64 image

Figure-14 shows that not only the increased block size can increase the target performance, but also increased image size compared to figure-13 would also have a positive impact on target performance. It is important to state at this point that, as the 32*32 image size was not having that positive impact on task and target performances after increasing block sizes, I therefore, decided to omit representing the similar task for 16*16 images, even though I have analyzed and found same result as conceived, but much lesser performance.

VII. Future work

Among the myriad of applications, domain adapted GAN can be just the beginning of a cherry on top of a cake in terms of the possibility to provide trustworthy and effective synthetic dataset. For airport surveillance to hand pose detection, it can be a great technique to efficiently produce synthetic data. Specially for hand pose detection [**?**], domain adapted GAN like pixelDA can be a building block to add a training mechanism, where rather than normal data augmentation, a slight portion of training dataset can be utilized to generate reasonable synthetic data to induce wide variety of training images for the network, which for sure will enhance the possibility of unseen image or pose detection.

In future, along with the generalized structure modification, the author would like to stress onto a novel loss function which will take the different viewpoint of an image as a consideration and trying to enhance the variation of synthesized images by optimizing the loss.

Moreover, this generalized idea can be incorporated to all kind of task based applications keeping in mind the types of data to be synthesized. Blending the newly generated data with a target and task performance accuracy can help training many neural networks more efficiently and precisely. Therefore, the workload and workforce behind the image collection and data generation for many complex analytical dataset would be reduced and well optimized.

Also, along with this, symmetric by directional GAN [**?**] should also be explored to have a meaningful impact in this paradigm.

VIII. Conclusion

The main purpose of this work was to aid an early stage researcher who is interested to use GAN for synthetic data rendering for various applications including dataset generation for hand pose detection [**?**], action detection [**?**] etc. A modified generator block structure was proposed by increasing the Generator block size (doubled to be precise) and feeding the network with two different sized images separately along with noise. As a result, the target and task accuracies were measured with several combinations.

Initially it was found that, task performance of 64*64 input image was increased while increasing the block size from 6 to 12. And in terms of target performance also improved for 64*64 images once the block size was increased.

To get an intuitive inner view, final part of those experiments made a head to head comparison for blocks with individual image sizes and found that task performance and target accuracies were not meaningfully improved within the simulated epochs after increasing block size. But, in contrary, improved in a much better for when 64*64 saw a increase in blocks from 6 to 12 in terms of target performance, while, task performance for the same image input size but different block remained unchanged.

In summary, the analysis is an initial leap forward towards helping or guiding synthesizing complex datasets. Even though the experiments were too open ended, some interesting conclusions can help to generate synthesized datasets for task specific applications.

IX. Acknowledgments

Of the many who provided us with constructive comments, I are especially grateful to Prof. Dr. Haitham Amar and Mr. Mahmoud Mohamed Mahmoud Nasr Mohamed for their kind feedback and clear direction. I would like to pay my sincere gratitude to all the authors of the main paper on which I based my work.