16th International Conference on Machine Vision Applications (MVA)
National Olympics Memorial Youth Center, Tokyo, Japan, May 27-31, 2019.

A Three-Player GAN: Generating Hard Samples To Improve Classification Networks

Simon Vandenhende, Bert De Brabandere, Davy Neven and Luc Van Gool
KU Leuven
ESAT-PSI, Belgium
firstname.lastname@esat.kuleuven.be

Abstract

We propose a Three-Player Generative Adversarial Network to improve classification networks. In addition to the game played between the discriminator and generator, a competition is introduced between the generator and the classifier. The generator's objective is to synthesize samples that are both realistic and hard to label for the classifier. Even though we make no assumptions on the type of augmentations to learn, we find that the model is able to synthesize realistically looking examples that are hard for the classification model. Furthermore, the classifier becomes more robust when trained on these difficult samples. The method is evaluated on a public dataset for traffic sign recognition.

1 Introduction

Deep convolutional neural networks have brought significant progress to the area of computer vision. However, training the models still requires vast amounts of data. As intelligent vision systems are being deployed in increasingly dynamic environments, collecting the necessary data becomes a tedious task.

Recent work in generative modeling, based on Generative Adversarial Networks (GANs) [1–4], allows to efficiently synthesize novel samples that belong to the data distribution. GANs derive the data distribution from an adversarial game, played between two entities: the generator G synthesizes new samples, and the discriminator D tries to separate real samples from the ones synthesized by G. The goal of the generator is to confuse D so that it cannot discriminate between real and fake examples. The game ends when the two players are at a Nash equilibrium.

GANs prove useful to improve the performance of classification networks. For example, [5] proposes an adversarial approach which jointly optimizes the data augmentation and a network for pose estimation. The generator learns to synthesize augmentations from the training data that are hard to label for the classification network. The augmentations are composed of rotations, scaling transformations and occlusions.

Furthermore, [6–9] have successfully employed GANs in a semi-supervised learning setting. In [6], the discriminator learns the classification task from unlabeled data. The discriminator has to classify each sample into a chosen number of categories. Since the conditional distribution $p\left(c|x\right)$ is unknown, a goodness of fit measure is included to ensure correspondence between the categories and the class labels. [8] trains a classifier in a semi-supervised manner by considering images from the GAN as samples from an additional class. [9] trains the classifier and the generative model simultaneously. They find that both generator and classifier represent a conditional distribution between labels and images. This observation leads to a compatibility criterion between the generator and classifier.

Our work implements a three-player adversarial game in which the classification network participates. The generator adapts itself to both the discriminator and classifier. This allows the generator to estimate the distribution of samples that are hard to label correctly for the classifier. In contrast to [5], our work does not restrict the type of augmentations that can be learned. Also, the proposed method simply relies on backpropagation, which makes it a very general approach. We show that the three-player game can improve classification networks, when annotated data is scarce. The proposed method is evaluated on CURE-TSR [10], a publicly available dataset for traffic sign recognition.

2 Method

A regular Generative Adversarial Network [1] comprises a min-max game, played between the discriminator D and generator G. Additionally, we now introduce a competition between the generator and classifier. The objective for G changes from synthesizing images that are realistic, to generating images that are both realistic and challenging for the classification network.

As before, the discriminator is trained to predict whether a sample is real or fake. The generator, in turn, optimizes the sum of two losses. The first term is the regular GAN loss, provided by the discriminator. In order for the generator to compete with the classifier, the second loss term needs to be chosen appropriately. To this end, backpropagation should yield the maximization of the classification model's loss, on samples from G. This encourages G to move towards the distribution of samples that confuse the classifier. The classifier is trained by minimizing the classification loss on samples from G. The game is played by

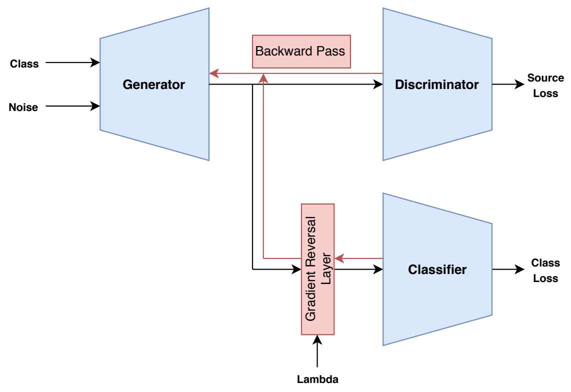


Figure 1: Setup for the three-player game. Images from the generator are propagated through both the discriminator D and the classifier C. The gradient that is backpropagated through D proceeds as usual. The gradient that is backpropagated through C is rescaled and inverted as $-\lambda\nabla_{\theta_C}L_C$. The loss from D penalizes G for synthesizing unrealistic samples, while the inverted loss from C rewards G for synthesizing difficult samples.

updating all three models one after another.

Inspired by [11], the objective for the second loss term, seen by G, is realized by implementing a gradient reversal layer between the generator and classifier. During the forward pass, samples from G are simply passed to the classification network. When backpropagating the classification loss, the sign of the gradient is reversed, causing the update in G to maximize the classification loss. This technique is related to [12], which finds adversarial examples by applying perturbations that lie along directions where the classification loss is likely to increase. The setup of our system is shown in figure 1.

The three-player GAN shows some similarities with auxiliary classifier GANs (ACGANs) [13]. In the ACGAN model, the discriminator categorizes the images in addition to predicting their source. This allows the discriminator to be deployed as a classification model. There are two main differences with our approach. First, in the three-player game, the generator tries to maximize the classification loss rather than minimize it. The focus of this work is on the generation of hard samples. Secondly, the three-player GAN separates the network architecture of the discriminator and classifier. This allows to specialize the architecture of the discriminator and classifier for their respective tasks.

The complete training procedure for the three-player game is defined in algorithm 1. A hyperparameter λ is introduced to weigh the classification loss against the discriminative loss.

Algorithm 1: The three-player GAN

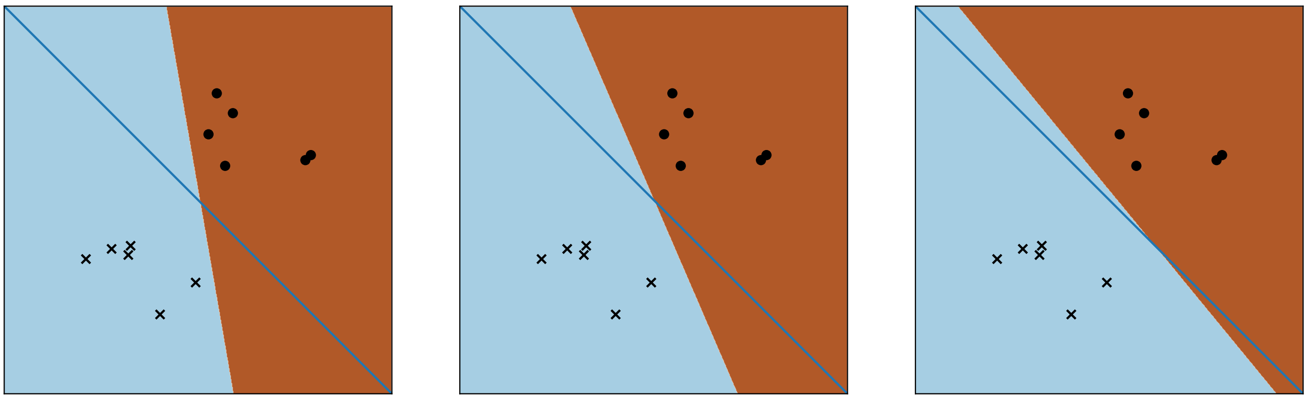
```
for number of training iterations do
    •Sample a batch $(x_g, y_g)$ of size $m$ from the generator, and a batch $(x, y)$ of size $m$ from the training data.
    •Update the discriminator by ascending its stochastic gradient:
        $$\nabla_{\theta_d}\left[\frac{1}{m}\sum_{(x,y)}\log D(x,y) + \frac{1}{m}\sum_{(x_g,y_g)}\log\left(1-D(x_g,y_g)\right)\right]$$
    •Sample a batch $(x_g, y_g)$ of size $m$ from the generator.
    •Update the generator by descending its stochastic gradient:
        $$\nabla_{\theta_g}\left[\frac{1}{m}\sum_{(x_g,y_g)}\log\left(1-D(x_g,y_g)\right) - \lambda\nabla_{\theta_c}\left(\frac{1}{m}\sum_{(x_g,y_g)}L_C\left(x_g, C(y_g)\right)\right)\right]$$
    •Sample a batch $(x, y)$ of size $m$.
    •Update the classifier by descending its stochastic gradient:
        $$\nabla_{\theta_c}\left[\frac{1}{m}\sum_{(x,y)}L_c\left(x, C(y)\right)\right]$$
end
```

3 Experiments

We first consider a toy example, which demonstrates that the three-player game acts as a regularizer for the decision surface of the classifier. In the second part we evaluate our method on CURE-TSR [10]. Both experiments compare the performance of a classification network trained through the three-player game against several other training scenarios.

3.1 Training details

We initialize the discriminator and generator in the three-player game by training a conditional GAN. When updating the classifier, we sample batches containing both real images, images synthesized by the initial generator and images synthesized by the current generator. The samples from the initial generator



(a) Trained on real samples.

(b) Trained on real and synthesized samples.

(c) Trained on real, synthesized and difficult samples.

Figure 2: Decision surface of the classification model at the end of different training procedures.

serve to avoid catastrophic forgetting of examples that are difficult early on.

The learning rates and the weighing parameter λ are updated according to the scheme from [6],

$$\lambda = \frac{2 \cdot w_c}{1 + \exp(-10 \cdot p)} - 1, \mu = \frac{\mu_0}{(1 + \alpha \cdot p)^\beta}$$

with p the training progress growing linearly from 0 to 1, $\alpha = 10$, $\beta = 0.75$, $w_c = 0.1$ and μ_0 the initial learning rate. The value of w_c is chosen smaller than one to ensure that synthesizing realistic samples has priority over synthesizing difficult ones. The weighing parameter λ gradually grows during training, allowing the generator to come up with difficult samples even when the classification model becomes better.

3.2 Toy example

We demonstrate that the three-player GAN effectively acts as a regularizer, by means of a toy example. Consider the case where samples from two classes need to be separated. Both classes are distributed as two-dimensional Gaussians, parameterized by $\mu_X = \pm 1, \mu_Y = \pm 1$ and $\sigma_X = \sigma_Y = 0.5$. The training data consists of eight examples per class, drawn as dots and crosses in figure 2. The classifier, represented as a simple linear mapping, is trained using a hinge loss.

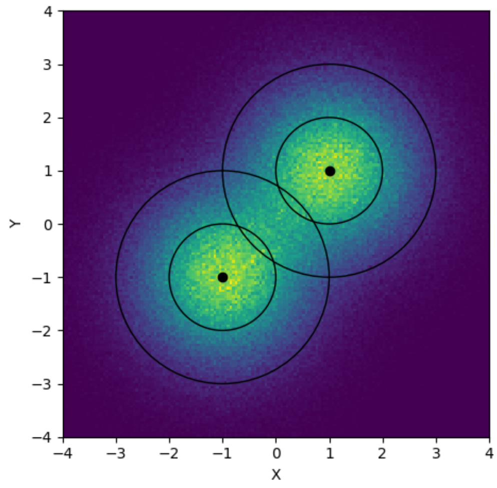
A baseline classifier and conditional GAN are trained using the available training examples. A second classification model is trained on a combination of real and synthesized samples. Thirdly, we also train a classification model based on the three-player game. For this particular example, we initialize the classifier as the baseline model and freeze its parameters. The game is played for a few epochs, allowing the generator to estimate the distribution of samples that are difficult for the baseline model. The parameters of the classification model were initialized randomly. To ensure a fair comparison, we made sure that the classification model uses the same initial weights. Figure 2 shows the decision boundary of the classification models obtained by different training schemes. Through comparison we find that the three-player GAN is able to regularize the decision surface.

Consider again the two Gaussian distributions from before, but with an increased variance. When sampling from the two classes, we find that the distributions show a significant overlap near the origin. If the three-player game behaves as intended, we expect the generator to synthesize samples which lie near the origin. We train a classification model and a conditional GAN by sampling from the two Gaussian distributions. Afterwards, the generator is updated through the three-player game in order to synthesize difficult samples. Figure 3 shows the results. We find that the generator learns to synthesize samples at locations where the classifier has a hard time.

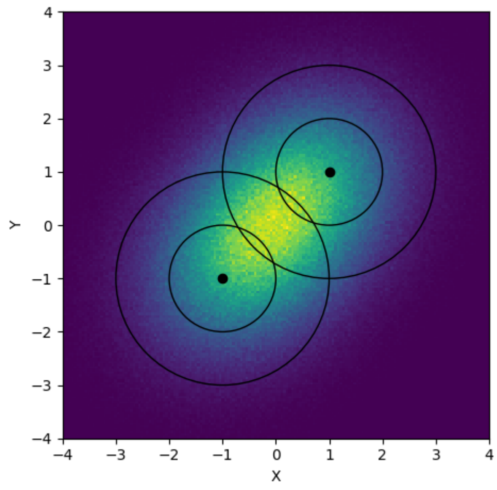
3.3 CURE-TSR

The CURE-TSR dataset [10] is composed of both real and simulated images of 14 traffic sign classes under various weather conditions. The set of simulated traffic sign instances is considered here under the following conditions: clear weather, low-mid-high levels of snow, low-mid-high levels of rain and low-mid-high levels of dark weather. For the training (resp. validation) set we took the first 100 (resp. last 50) images per class from each weather condition. Since the data contains sequences of images for which the camera gradually moves closer to the traffic sign, the data selection comes down to using only a few of such sequences.

Again, we train a classifier and conditional GAN using the available training data. A second classification model is trained using both real samples and samples synthesized by the conditional GAN. Thirdly, we compare with an auxiliary classifier GAN. Finally, a classifier is learned by means of the three-player game. As mentioned in section 3.1, the discriminator and generator are initialized as the models from the conditional GAN.



(a) The true distribution consists of two classes, both distributed as two-dimensional Gaussians.



(b) Distribution learned by generator during the three-player game. The generator synthesizes samples located in the area where the two class distributions overlap.

Figure 3: The real distribution consists of two classes which partially overlap. Both are two-dimensional Gaussians of which the mean is indicated as a dot. The circles in the figure correspond to multiples of the standard deviation. We find that the three-player GAN learns to synthesize samples at locations where the two classes overlap. These are samples which are hard to label correctly for a classification model.

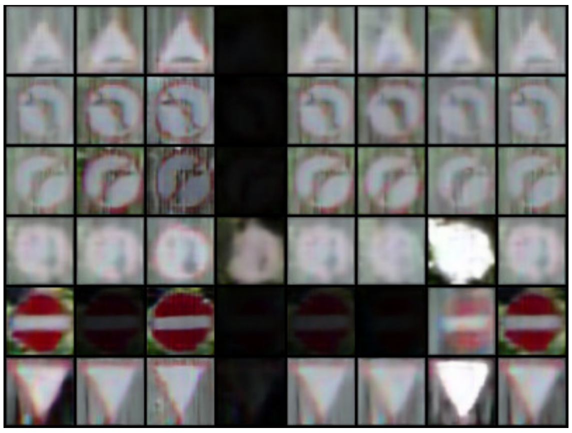


Figure 4: Images generated during the three-player game.

The network architecture for the classifier is based on one column from the multi-column deep neural network used for traffic sign recognition in [14]. The discriminator and generator networks are based on earlier work [2]. More details can be found in the supplemental materials. The classification network was trained for 150 epochs with an Adam Optimizer [15] ($\mu_0 = 0.001, \beta_1 = 0.5, \beta_2 = 0.999$). The learning rate is degraded by a factor 10 every 60 epochs. A weight decay term of $1e - 4$ is included in the classification loss. The conditional GAN was trained for 500 epochs using batches of size 64. For the auxiliary classifier GAN, we reused the architecture and training scheme from the original work [13]. We used an Adam Optimizer with the same learning parameters as [16] ($\mu_0 = 0.0002, \beta_1 = 0.0, \beta_2 = 0.9$). The initial learning rates for the three-player game are the same as for the other training strategies. The results can be found in table 1. We find that training the classification model by means of the three-player game improves the test accuracy. Figure 4 shows images that were generated during the three-player game.

Table 1: Accuracy on the CURE-TSR test set for different training schemes.

Method	Without real data	With real data
Baseline	-	83.38
cGAN	77.08	83.23
ACGAN	-	79.23
Threeplayer	**79.83**	**85.41**

4 Conclusion

We have proposed an effective, yet simple method to improve classification networks, by having a generative model synthesize difficult samples. The method is based on a regular GAN game, but includes an adversarial loss which steers the generator towards difficult samples. In comparison to previous work, we do not restrict, nor limit the kind of augmentations that the generative model can learn. We find that the generative model is able to synthesize realistically looking images which are hard to label correctly for the classification model. Since our method simply relies on backpropagation, future research can look whether the idea also applies to different tasks.

Acknowledgement: The work was supported by Toyota, and was carried out at the TRACE Lab at KU Leuven (Toyota Research on Automated Cars in Europe - Leuven).

5 Supplemental Materials

Network Architectures - CURE

Table 2: Discriminator

Operation	Features	Output size
48 x 48 input	3	
ResNet block	48	24 x 24
ResNet block	96	12 x 12
ResNet block	192	6 x 6
ResNet block	384	3 x 3
ReLU		
Sum Pool	384	1 x 1
Linear	1	
Filter size	3 x 3	
Initialization	Xavier - $\sqrt{2}$	

Table 3: Generator

Operation	Features	Output size
100 x 1 noise		
Linear	384	3 x 3
ResNet block	384	6 x 6
ResNet block	192	12 x 12
ResNet block	96	24 x 24
ResNet block	48	48 x 48
BatchNorm		
ReLU		
Convolution	3	48 x 48
Filter size	3 x 3	
Initialization	Xavier - $\sqrt{2}$	

Table 4: Classifier

Operation	Features	Kernel	Nonlinearity
Convolution	100	7 x 7	ReLU
Convolution	150	4 x 4	ReLU
Convolution	250	4 x 4	ReLU
Linear	300		ReLU
Linear	43		
Initialization	Xavier - $\sqrt{2}$		
Batch normalization after each convolution			
Dropout ($p = 0.5$) in linear layers			

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[2] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, “Spectral normalization for generative adversarial networks,” *arXiv preprint arXiv:1802.05957*, 2018.

[3] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, “Self-attention generative adversarial networks,” *arXiv preprint arXiv:1805.08318*, 2018.

[4] A. Brock, J. Donahue, and K. Simonyan, “Large scale gan training for high fidelity natural image synthesis,” *arXiv preprint arXiv:1809.11096*, 2018.

[5] X. Peng, Z. Tang, F. Yang, R. S. Feris, and D. Metaxas, “Jointly optimize data augmentation and network training: Adversarial data augmentation in human pose estimation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2226–2234.

[6] J. T. Springenberg, “Unsupervised and semi-supervised learning with categorical generative adversarial networks,” *arXiv preprint arXiv:1511.06390*, 2015.

[7] A. Odena, “Semi-supervised learning with generative adversarial networks,” *arXiv preprint arXiv:1606.01583*, 2016.

[8] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved techniques for training gans,” in *Advances in Neural Information Processing Systems*, 2016, pp. 2234–2242.

[9] C. Li, K. Xu, J. Zhu, and B. Zhang, “Triple generative adversarial nets,” *arXiv preprint arXiv:1703.02291*, 2017.

[10] D. Temel, G. Kwon, M. Prabhushankar, and G. AlRegib, “Cure-tsr: Challenging unreal and real environments for traffic sign recognition,” *arXiv preprint arXiv:1712.02463*, 2017.

[11] Y. Ganin and V. Lempitsky, “Unsupervised domain adaptation by backpropagation,” *arXiv preprint arXiv:1409.7495*, 2014.

[12] I. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” 2015.

[13] A. Odena, C. Olah, and J. Shlens, “Conditional image

synthesis with auxiliary classifier gans," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2642–2651.

[14] D. Cireşan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," *arXiv preprint arXiv:1202.2745*, 2012.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[16] T. Miyato and M. Koyama, "cgans with projection discriminator," *arXiv preprint arXiv:1802.05637*, 2018.