DCGAN in Traffic Sign Image Inpainting

Tianqi Liu, Houyi Du, Jiaying Wu
University of Illinois at Urbana Champaign
Champaign, IL
tliu51@illinois.edu, houyidu2@illinois.edu, jwu86@illinois.edu

Abstract

Image inpainting is the process of reconstruct the lost parts of images and videos. It is one of the most difficult tasks in computer vision field, and can have multiple real world applications. In this paper, we trained a deep convolutional generative adversarial network(DCGAN) model and achieved the inpainting by using images generated from the generator of GAN, from which choose the image that could most likely recover the cropped images. Two methods are used to generate cropped images, randomly crop pixels and crop a whole chunk of pixels. We juxtapose the performance of each method through another convolutional neural network, by comparing accuracy of classification of cropped images, inpainted images, and original images we can evaluate the performance of image inpainting pipeline.

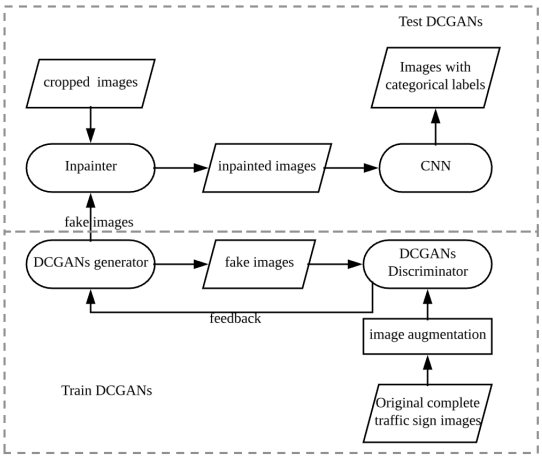

Figure 1: Image Inpainting Pipeline

1. Introduction

Generative Adversarial Network(GAN)[6] is a well studied direction in recent years and it play a central role in many practical image quality improvement tasks such as low resolution photo to high resolution photo transformation. This project is motivated by improve the quality of advanced driver-assistance system input image. The Advanced driver-assistance systems(ADAS) make use of neural networks to recognize traffic signs and road line. However, the performance of ADAS can be influenced by bad weather condition since the traffic signs can not be correctly recognized if it was partially blocked by other objects such as rain drops or snow. There are multiple ways such as semantic segmentation that can distinguish the pixels that have been "contaminated" from the pixels belongs actual traffic signs. However, we still need to recover those pixels. The past decade has witnessed great advancement in image inpainting techniques such as deep convolutional adversarial generative network (DCGAN)[10] and Variational Autoencoder (VAE). In this project, we provide a new direction to improve the stability of the traffic sign recognition system on ADAS under poor quality images by pre-processing the input image using a DCGAN image inpainter before entering image classifier. The "damaged" pixels shall be recovered if the missing part is not too big, thus the image classifier would have better accuracy.

On the other hand, we can also use this pipeline to evaluate the effectiveness of DCGAN image inpainter. Though, GAN have been extremely effective in producing complex distributions of high-dimensional fake images in practice. However, there is no systematic way to evaluate image inpainting result generated by DCGAN. Methods like human perceptual scores[9] is still need human-labor to evaluate. So one of our contribution is to offer a new evaluation method. We add another convolutional neural network to classify images from three sources: original images, cropped images that have been removed some pixels and inpainted images generated by generator of DCGAN. We compare the accuracy improvement from cropped images to inpainted images as a criterion to evaluate the result of inpainting.

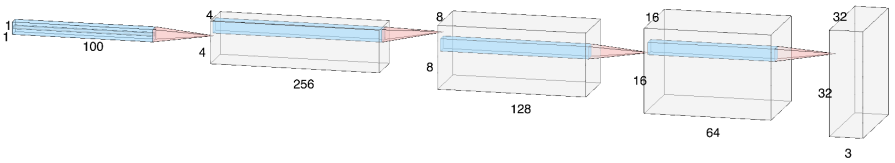

Figure 2: Generator Structure

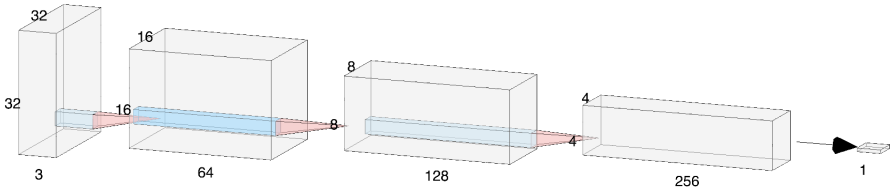

Figure 3: Discriminator Structure

The main component of the whole project is Generative Adversarial Network[6], which is one of the most explored area. Many modified versions of GAN has also been published in recent years such as LSGAN, DCGAN, and WGAN. While GAN suffers from unstable training process and the algorithms may fail to converge[11], the modified versions of GAN improve the training stability. In our project, we will use the Deep Convolutional Generative Adversarial Networks[10], which could be used in many image generation tasks such as image inpainting. Image inpainting is the technique of modifying an image in an undetectable form[3] and could be realized in many different ways like texture-based inpainting[5]. In this paper, we reproduce the inpainting algorithm of Raymond A. Yeh's paper[13].

Overall, based on the existing DCGAN image inpainting work, we verified the feasinility of using DCGAN image inpainting to improve the performance of image classifiers on "dameged" images. Also, we proposed a new evaluation method to measure the effectiveness of image inpainters by comparing the accuracy of image classifier on cropped images and inpainted images.

2. Methodology

The overall code pipleline is shown in Figure 1. Our pipeline consists of four main parts which are training data pre-processing, generator training, image inpainting and inpainted image evaluation. The training data pre-processing serves the purpose of avoid generator overfitting. The generator training is accomplished by applying deep convolutional adversarial neural network, and the generator will be used in image inpainting process to generate the missing part. Finally, a convolutional neural network traffic sign classifier is build to evaluate the performance of inpainter. The inpainting method we use are based on paper Semantic Image Inpainting with Deep Generative Models [13]. Our DCGAN structure is modified from DCGAN tutorial in Pytorch website[1], and classifier structure is a modified version of Keras-Tutorial Traffic Sign Recognition[4].

2.1. Data Augmentation

German Traffic Sign images in 43 categories are used as data source[12]. The original images size are varied, due to the limitation of device, we re-size the images to $32 * 32$. In addition, since the number of images in each category varies from around 200 to 2000, in order to avoid generator from bias to a particular category of traffic sign, we increase the number of images of some categories by rotating, resizing, and adjusting brightness of corresponded category to generate enough copies (roughly 2000 per class) of them. By such adjustment, we are able to avoid data overfitting in some extend and improve performance of generator.

2.2. DCGAN Training

In this section, we will briefly introduce GAN parallel with detailed introduction to DCGAN, what we actually applied for training generator.

2.2.1 GAN

In our inpainting task, we need to generate a fake image first, so for this purpose choose generative adversarial network published in recent years as our overall approach to train a image generator. The GAN model consists of two main part, a generator and a discriminator. The generator is denoted as G and it serves the purpose of generate fake images. The discriminator is classifier denoted as D, and the purpose of D is to classify whether the input is fake or real. The overall idea of generative adversarial neural network is

we throw some random noise into generator to produce a fake image. Then use fake images and real images to train the discriminator, and use discriminator's feedback to enhance the performance of generator. The loss function of GAN is the following:

$$loss = \log D(x) + \log(1 - D(G(z))) \tag{1}$$

The x here stands for real images, and z is latent vector, the input for generator. The first part of the loss function $\log D(x)$ is used to improve the discriminator's performance on recognizing real images, and the second part $\log(1 - D(G(z)))$ helps the discriminator classify the fake image better. The second part also serves purpose of indicating the generator to fool the discriminator[6].

for *number of training iterations* **do**
  Sample m training images from minibatch;
  Randomly pick latent vector z;
  Generate m fake image through generator G;
  Update discriminator D by ascending its stochastic gradient:
  $\nabla_{\theta_d} = \frac{1}{m}\sum_{i=1}^{m}[\log D(x^i) + \log(1 - D(G(z)))]$
  Update the generator G by descending its stochastic gradient:
  $\nabla_{\theta_g} = \frac{1}{m}\sum_{i=1}^{m}[\log(1 - D(G(z)))]$
end

Algorithm 1: GAN Training Using Stochastic Descent

2.2.2 DCGAN

Since the generative adversarial network is unstable and we didn't get a workable generator after trying different hyperparameters, we use DCGANs(deep convolutional generative adversarial networks)[10] a modified version of GAN published in 2015 instead.

The DCGAN paper suggests DCGAN model replaces the fully connected hidden layers in GAN model with deep convolutional neural network. In the network architecture, ReLU is used for generator's activation function except on the output layer. LeakyReLu is used as activation function for discriminator. The batch normalization is used in discriminator and generator[10]. Even with the help of DCGAN, our generator is not stable in the training procedure, and the discriminator always significantly outperformed in the generator. In order to get better results, we apply some tricks to enhance the performance of DCGAN generator.

First, instead of using hard labels (1 or 0) in the discriminator, we use soft labels. Some recent years research shows smooth label will improve the stability of neural network[7], so we set the soft label bound to be randomly distributed within a certain range. During our training and experiment, we found out choosing Gaussian distributed in $[0, 0.2]$ and $[0.9, 1.1]$ will give best performance. In addition, we flip the real label and fake label $real' = fake, fake' = real$ to help the generator to fool the discriminator and indeed improve the performance of generator[8].

2.3. Image Inpainter

The inpainting algorithm comes from paper Semantic Image Inpainting with Deep Generative Models[13]. Before we inpaint images, we would need to generate cropped images by using mask matrix. Same as the definition in the blog[2], we will define the mask M as a matrix which only contain 0 and 1. Cropped images will be the Hadamard product of mask with original images. Thus, the inpainted image will be combination of generated fake image with outside frame of original image:

$$X_{inpaint} = \|M \circ y + (1 - M) \circ G(z)\|$$

where y is the original image and $G(z)$ is the generated fake image. $\circ$ stands for the Hadamard operation.

The general work flow of our image inpainter is shown in the following pseudocode:

Initialize latent vector $\hat{z}$ randomly;
for *iteration = 1,2...,5000* **do**
  Generate fake image G();
  Use fake image to fill in the cropped pixels in cropped Image;
  New image $I_2 = M \circ I_1 - (1 - M) \circ G(\hat{z})$;
  Content loss = $\|M \circ I - M \circ G(\hat{z})\|_1$;
  Perceptual loss = $log(1 - D(I_2))$;
  Total loss = content loss * 0.9 + perceptual loss * 0.1;
  Use gradient descent to update $\hat{z}$;
end

Algorithm 2: Image Inpainting

The loss of image inpainter consists of two parts. The first part is contextual loss, which measures the difference between generated fake image and original image. The equation of the contextual loss is the following:

$$L_{context} = \|M \circ G(z) - M \circ y_{real}\|$$

Here, we are calculate the difference of the uncropped part between original image and generated image. We are assuming the generated image whose uncropped part best fit the original image will have better fitness when we pass the cropped part of generated image back to original image.
The second part of inpainter loss is the perceptual loss, being used to make sure the fake image generated by Generator looks real. Otherwise we might get a image from which the cropped part shares many features with the uncroped

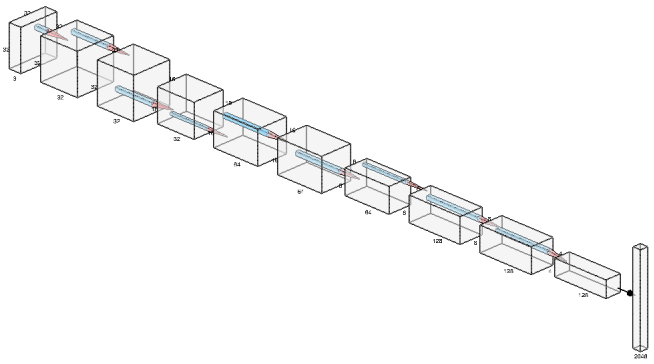

Figure 4: CNN Structure

part of original image, but looks unreal. The inpainted image is passed into discriminator. The result can be used to calculate the perceptual loss:

$$L_{percept} = log(1 - D(X_{inpaint}))$$

The total loss of image inpainter is the sum of the contextual loss and perceptual loss:

$$Loss = L_{context} + L_{percept}$$

By searching through latent space, we want to find a latent vector z* that could minimize the overall loss and thus generate the best fake image. The optimal latent vector z* can be found using gradient descent.
Once we found the best latent vector z, it will be passed into our generator and generate fake image $G(z)$. Then, we apply the inverse of crop mask to $G(z)$, and get $(1-M)\circ G(z)$, the recovered pixels on cropped area. Finally, we insert $(1 - M) \circ G(z)$ back into the cropped part in the original image, and get the inpainted image[13].

2.4. CNN Traffic Sign Classifier

By using the framework of DCGANs, we crop a set of images and feed them to our CNN classifier. There are two croped methods we have applied to build our test image sets. The first crop method is crop with large square trunk. In this method, we remove 10% of pixels from every test image and the shape of the removed part is square. The second method is random crop method. In this method, we agian remove 10% of the pixels from every test image but the shape of the removed part is random, and the removed part could be pixels scattered in the image. The first column in Figure 5 shows our two cropped methods. By using the two crop methods, we will get two sets of images and each have three categories. The first set is $\{$original images, images with a large chunk cropped, inpainted images$\}$, the second is$\{$original images, images with a random cropped pixels, inpainted images$\}$.

In the architecture of our CNN model, we repeatedly using following concatenated multiple layers: Convolutional Layer, Rectifier layer, and Pooling Layer. In addition with some dropout layers and a last fully connected layer.

We compare the accuracy of each categories to evaluate our image inpainting performance. Our CNN architecture is shown in Figure4.

(a) Large Chunk Cropped, Image Recover and Random Pixel Cropped Image Recover

Figure 5: Image Inpainting Results

3. Experimental Results

Figure 5 and Figure 6 are some image inpainter result. Figure 5 shows the inpaint result respect to different cropped methods of stop sign. The first column in Figure 5 corresponds to the original cropped image, and the second columns are the inpainting result after first round.

(a) Large Chunk Cropped Image Recover

Figure 6: Image Inpainting Results

The third column corresponds to inpainting result after 5000 rounds. Figure 6 shows large chunk cropped image recovering. The first column in Figure 6 corresponds to result after 1 round, and the second column corresponds to result after 5000 round. During the training process of DCGAN, we try to train it with 10, 15 and 20 epochs, and finally choose 15-epoch one, because we observed that the 20 epoch one brings overfitting problem to our generator.

To verify the effect of our DCGAN image inpainting pipeline, we took 5 test images from each classes to form a test set of 215 images. For each image, first we remove a square at fixed position that counts 10% of the image size. The accuracy of classifier dropped to 47%. Then we use the DCGAN image inpainter to recover the image and put the recovered image into CNN classifier again. The accuracy of classifier with recovered image as input raised to 68%. We also tried remove 10% of pixels randomly. The accuracy of traffic sign classifier on those randomly cropped image dropped to 10%, while the accuracy after inpainting became 20%.

Our resuls show that the image inpainting pipeline significantly improve the quality of test input image, increases the accuracy of cropped traffic sign image classification and proves the effectiveness of our DCGAN image inpainting pipeine.

	Original	Cropped	Recovered
Remove Square	89%	47%	68%
Randomly Removal	89%	10%	20%

Table 1: Accuracy Improvement Results

(a) Randomly Cropped Image Recover

Figure 7: Image Inpainting Results

4. Discussion and Conclusion

In this project, we aims to build a pipleline which could input data quality of advanced driver-assistance system by applying DCGAN image inpainting, and able to evaluate the performance of our image inpainter. We successfully reconstruct the DCGAN, and image inpainter. Our novelty and contribution can be summarized to two main aspects. First, instead of use eyes to judge the quality of image inpainting results, we are now able to use a more sophisticated way to judge and give a concrete performance evaluation of image inpainter. Our original goal for construct a image inpainter for traffic sign is we want to improve the effi-

ciency of Advanced driver-assistance systems especially in bad weather or at night condition. Assuming the cropped part is the blocked part or unclear part of image in bad weather condition, from our result, it can be seen that the CNN classifier performance improved after we processed by using our image inpainter. Thus, the second contribution for our work would be the image inpainter framework provide a possible direction of improvement in advanced driver-assistance system in bad weather condition. The future direction of our work could be embed the image inpainter into the advanced driver-assistance system's neural network to improve classification efficiency by enhance input data quality. Also, due to the limitation of hardware, the size of image is set to 32. With better hardware, image inpainting on larger sized image would be possible. Another way to improve the performance of DCGAN is to use image classifier as the discriminator. Hence, the generated fake images would look closer to each certain class of images. Additionally, there still some space for us to improve our CNN classifier.

References

[1] Dcgan tutorial.

[2] B. Amos. Image Completion with Deep Learning in TensorFlow. `http://bamos.github.io/2016/08/09/deep-completion`. Accessed: [Insert date here].

[3] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 417–424, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[4] S. Chilamkurthy. Keras tutorial - traffic sign recognition. *Sasank's Blog*, Jan 2017.

[5] A. Criminisi, P. Perez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on Image Processing*, 13(9):1200–1212, Sept 2004.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[7] T. Hazan, G. Papandreou, and D. Tarlow. *Adversarial Perturbations of Deep Neural Networks*. MITP, 2017.

[8] F. Huszar. Instance noise: A trick for stabilising gan training, Oct 2016.

[9] D. J. Im, H. Ma, G. W. Taylor, and K. Branson. Quantitatively evaluating gans with divergences proposed for training, 2018.

[10] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015.

[11] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen. Improved techniques for training gans. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2234–2242. Curran Associates, Inc., 2016.

[12] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *IEEE International Joint Conference on Neural Networks*, pages 1453–1460, 2011.

[13] R. A. Yeh, C. Chen, T. Yian Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do. Semantic image inpainting with deep generative models. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.