

Playing with Perception: Fooling Traffic Sign Classifiers via Copy-Paste Manipulation — DCGAN Creation

Davide Zani (d.zani@student.rug.nl)

Supervised by Alexandru Dimofte (alex@aisig.org)

AI Safety Initiative Groningen

Abstract

In this project we created a Deep Convolutional Generative Adversarial Network (DCGAN) that is successfully capable of generating realistic traffic sign images. Training the DCGAN on German and Chinese traffic sign datasets reveals superior performance on the German dataset due to diverse lighting conditions and a larger dataset. The DCGAN’s implementation and performance on the German Traffic Sign Recognition Benchmark are discussed. We were able to create one of the essential components for our long term goal of generating adversarial copy-paste attacks against visual classifiers. More specifically, extending the targeted adversarial copy-paste attack method to traffic sign classifiers. Future work aims to integrate the DCGAN into existing adversarial systems, thereby enriching traffic signs datasets, and thus, contributing to safer and more robust autonomous driving technology.

Keywords: Autonomous Driving, Traffic Sign Classifiers, Adversarial Attacks, Deep Convolutional GAN (DCGAN), Copy-Paste Manipulation, Self-Driving Systems

Introduction

While most car manufacturers have adopted a diverse range of sensors and technologies for their self-driving endeavors, Tesla is attempting to reach self-driving capabilities by relying solely on visual-only systems. In February 2022, Elon Musk advocated for the “vision-only” approach, asserting that it offers simplicity, cost-effectiveness, and greater intuitiveness (Siddiqui, 2023). He expressed that the road system is tailored for cameras (acting as eyes) and computational models (representing brains) (Musk, 2022). However, attempting to replicate technology in an anthropomorphic manner also makes it inherit imperfections from the human model, in this case: eyes can be deceived, and so can visual classifiers. (Yelisetty, 2021)

Research has shown the effectiveness of copy/paste attacks against visual classifiers (Casper et al., 2022). Eykholt et al have shown that classifiers finetuned on traffic sign classification and recognition can be deceived with “Robust Physical Perturbations” (Eykholt et al., 2018), which consists of the calculated positioning of sticker-like patches on traffic sign images. This technique is capable of adjusting the patches in order for the original image to be misclassified as the so-called “target” image. Eykholt et al.’s paper focuses on obtaining misclassification, not specific to any category.

Casper et al. employ feature-level adversarial attacks to investigate how deep neural networks misinterpret images. Their focus is on causing the misclassification of images into specific target classes, like misclassifying bees as flies. To achieve this, they generate class-universal adversarial features. The term “class universal” refers to their ability to work for any source example, regardless of its class, and can cause

the network to misclassify the image as any target class. The term “adversarial features”, completing the definition, indicates that they are generated by feature-level adversarial attacks.

Subsequently, they integrate images of these objects into natural images and pass them through a target classifier. As a result, they successfully induce an unexpected misclassification of the image as the target class. This unexpected misclassification occurs because the images being manipulated may not inherently possess the characteristics of the target class, yet the adversarial features lead to misleading results.

Our final objective is to employ copy-paste attacks to deceive classifiers, adopting the “target-class” strategy presented by Casper et al.

However, due to time limitations and the complicated nature of this task, we decided to only work on one of the core elements that compose the attacker system: the GAN. This is also attributed to the fact that Casper et al. utilized external code to incorporate a generator, namely BigGAN (Hugging Face, 2019), in their implementation. The aforementioned GAN is finetuned on ImageNet1K, which does not present sufficient traffic sign classes in its dataset, the only class related to the topic is the general number 919: “traffic light, traffic signal, stoplight”, which is too general and not specific enough for our intended purpose. While Casper et al.’s approach was effective in their context, our objective required a specialized GAN that could produce realistic traffic sign images, crucial for deceiving traffic sign classifiers effectively.

For this reason, we created a Deep Convolutional GAN (DCGAN) to generate images of traffic signs from the ground up. The DCGAN was designed to provide us with more control and flexibility, allowing us to focus specifically on traffic sign images and tailor the generator to our needs.

Methods

The role of the GAN in the overall context of the paper is necessary, as the generator is one of the components responsible for generating localized adversarial perturbations, while the discriminator is used to assess the quality and perceptual similarity of the generated adversarial examples. This is done in order to create interpretable feature-level adversarial examples that can deceive the target classifier while appearing natural to human observers.

We were successfully able to create a DCGAN capable of generating realistic images of traffic signs using PyTorch. To ensure comprehensive training, we tested the DCGAN on two large traffic sign datasets: the German Traffic Sign Recog-

nition Benchmark (GTSRB) and the Chinese Traffic Sign Recognition Database (CTSRD). The same procedure was applied to both datasets. After downloading them, preprocessing was conducted to ensure seamless integration with the rest of the code. Specifically, all images were resized to dimensions of 32x32 to facilitate training. We trained the model using Google Colab's environment, running the code on NVIDIA T4 GPUs. The model was trained on each dataset separately to identify and select the better-performing one, based on the quality of the images generated.

For the training process, a learning rate of 0.0002, a batch size of 128, and 200 epochs were selected based on iterative trial and error tests. For example, it was observed that increasing the number of epochs past 200 did not yield any noticeable improvement in the model's performance. The structures for both the generator and the discriminator were kept relatively basic, adapting code examples from PyTorch's official website (Nathan Inkawich, 2019). This choice was made because the primary focus was to generate visually acceptable traffic sign images initially, with plans to further refine and enhance the models in subsequent stages of development. The subsequent explanations will incorporate PyTorch-specific terminology to provide a more precise and technically accurate depiction of the processes involved.

The generator consists of a sequence of layers that gradually upsample the input noise vector into a realistic image. The generator starts with a ConvTranspose2d layer that takes a 100-dimensional noise vector as input and transforms it into a 256-channel feature map with a kernel size of 4x4. Batch normalization is then applied to the feature map to improve training stability. A LeakyReLU activation function with a negative slope of 0.2 follows, introducing non-linearity to the network. The process is repeated twice more, with additional ConvTranspose2d layers gradually upsampling the feature map to 128 channels and then 64 channels, respectively, while maintaining the LeakyReLU activation and batch normalization. Finally, the last ConvTranspose2d layer transforms the 64-channel feature map into a 3-channel image (RGB) with a kernel size of 4x4, and a Tanh activation function ensures the output image's pixel values are in the range $[-1, 1]$. The architecture enables the generator to convert random noise into realistic 32x32 images of traffic signs.

The Discriminator's architecture is comprised of a series of Conv2d layers, which systematically downsample the input image while extracting relevant features. This progressive downsampling enables the Discriminator to determine whether the given image is similar to the images present in the training data or not. The discriminator begins with a Conv2d layer that takes a 3-channel RGB image as input and processes it into a 64-channel feature map with a kernel size of 4x4. Batch normalization is then applied to enhance training stability, followed by a LeakyReLU activation function with a negative slope of 0.2, introducing non-linearity to the network. This process is repeated two more times, with additional Conv2d layers downsampling the feature map to 128

channels and then 256 channels. Each layer includes batch normalization and LeakyReLU activation. The last Conv2d layer outputs a single-channel feature map representing the discriminator's decision, and a Sigmoid activation function ensures the output is in the range $[0, 1]$, where values close to 1 indicate a real image and values close to 0 indicate a fake image.

Reiterating what was previously stated, the implemented structures for both the generator and the discriminator were intentionally kept relatively basic. The primary objective at this stage was to generate visually acceptable traffic sign images. However, we plan to refine and enhance the models in the later stages of the project.

Results

As mentioned in the section above, To ensure comprehensive training, we tested the DCGAN on two large traffic sign datasets: the German Traffic Sign Recognition Benchmark (GTSRB) and the Chinese Traffic Sign Recognition Database (CTSRD). The images produced by the DCGAN (z4nix, 2022) displayed a notably higher rate of accurately generated visuals when evaluated on GTSRB's dataset in comparison to CTSRD. Moreover, the dataset encompasses a wide range of lighting conditions surrounding the traffic signs, which in turn facilitated improved overall generalization. In Figure 1, (a) represents the outputs obtained using the GTSRB dataset, while (b) represents the outputs obtained using the CTSRD dataset. The comparison illustrates the differences in the generated visuals between the two datasets. Expanding the results from the GTSRB dataset, Figure 2 shows a comparison between the generated images and randomly selected images from the training dataset (GTSRB).



(a) GTSRB (200 Epochs)



(b) CTSRD (200 Epochs)

Figure 1: Comparison of outputs obtained with GTSRB (a) and CTSRD (b)



Figure 2: Comparison with images from the training dataset (GTSRB)

Challenges

During this project, we encountered various challenges, the most important of which were time constraints, together with the underestimation of the complexity of this task. Despite our enthusiasm and determination to complete the project within the desired time-frame, we had to accept that achieving our initial vision required more time than anticipated. Although we were granted an extension, we recognized that some aspects of the project were best approached with a long-term perspective in mind. Even with an extension, we were only able to achieve a basic version of one of the components of the whole project. We realized that the code readily available for this task was scarce and that we were going to have to code most of the essential components of it ourselves, such as the protagonist of the paper: the GAN. This project has been a valuable learning experience, highlighting the significance of managing time effectively and setting realistic expectations. The challenges we encountered, particularly regarding the scarcity of readily available code, allowed us to exploit our technical skills and deepen our understanding of the subject matter.

Conclusion & Future

Relying on visual systems introduces potential vulnerabilities, as demonstrated by research on adversarial attacks against visual classifiers. Our objective in this project was to explore the use of copy-paste attacks to deceive classifiers, specifically adopting the "target-class" strategy proposed by Casper et al. We successfully implemented a component involved in the aforementioned strategy, a DCGAN capable of generating realistic traffic sign images. While Casper et al. used an external generator (BigGAN), we chose to develop a specialized GAN to tailor the generator specifically to traffic sign images. The DCGAN demonstrated superior performance on the German Traffic Sign Recognition Benchmark (GTSRB) dataset, capturing the distinctive characteristics of traffic signs.

Despite encountering challenges such as time constraints and the complexity of the task, this project provided valuable insights and technical skills development. Our work highlights the significance of developing specialized models and

deepening our understanding of adversarial attacks in the context of visual classifiers.

In the near future, we plan to integrate our DCGAN into Casper et al.'s code with substantial modifications to ensure compatibility. In order to integrate our DCGAN, we will need to replace all instances of usages of the BigGAN finetuned on ImageNet1K present in the code by the authors. Subsequently, we will need to alter their classifiers to an architecture capable of interpreting all of the classes our generator is capable of creating (all of GTSRB's labels).

Once both the classification models and the generator are complete, we will have to focus on making these work together, in order to correctly fool the classifiers, by altering the creations of the generator. However, before proceeding with the integration, our focus is on making significant improvements to the DCGAN. To improve the DCGAN's performance, we can explore several enhancements. Firstly, combining both the GTSRB and CTSRD datasets can create a more diverse and comprehensive dataset, helping the model produce more realistic samples. Secondly, introducing noise into the dataset can make the DCGAN more adaptable to real-world variations, increasing its robustness. Thirdly, optimizing the distribution of convolutional layers and trying different layer combinations can enhance the model's ability to extract relevant features from the data. By adding Linear and dropout layers, we can further improve feature extraction. Lastly, increasing the batch number during training may lead to more stable and efficient learning, positively impacting the DCGAN's overall performance. These potential improvements hold promise for advancing the DCGAN's capabilities and generating better outputs.

The research on adversarial attacks against traffic sign classifiers highlights the critical and fundamental importance of addressing safety and security implications for autonomous driving systems. The effectiveness of copy/paste attacks and other manipulations on traffic sign recognition raise concerns about the reliability of such systems in the real world. By continuously advancing our knowledge and refining our models, we hope to play a role in improving the integrity of self-driving systems and contributing to the safe and successful implementation of autonomous driving technology.

References

- Casper, S., Nadeau, M., Hadfield-Menell, D., & Kreiman, G. (2022). Robust feature-level adversaries are interpretability tools. *Advances in Neural Information Processing Systems*, 35, 33093–33106.
- Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., & Song, D. (2018). Robust physical-world attacks on deep learning visual classification. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1625–1634.
- Hugging Face. (2019). pytorch-pretrained-BigGAN.
- Musk, E. (2022). Tweet by elon musk.
- Nathan Inkawhich. (2019). *Dcgan tutorial - generating faces with pytorch* [Accessed: 2023]. https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html
- Siddiqui, F. (2023). How elon musk knocked tesla's 'full self-driving' off course. *Washington Post*. <https://www.washingtonpost.com/technology/2023/03/19/elon-musk-tesla-driving/>
- Yelisetty, A. (2021). How to fool a neural network? *Towards Data Science*. <https://towardsdatascience.com/how-to-fool-a-neural-network-958ba5d82d8a>.
- z4nix. (2022). Traffic-Sign-DCGAN: DCGAN for traffic sign generation [Accessed on: July 23rd 2022].