Research Idea

Application: Understanding classification failure in Machine Learning particularly for Road Signs, using CNNs and GANs.

Models used: Convolutional Neural Networks, Deconvolutional Neural Networks and Deep Convolutional Generative Adversarial Networks (DCGANs), Pre-trained Inception and ResNet networks.

Introduction

Understanding how and why neural networks can be fooled into misclassifying images is an interesting topic that has been widely explored. This topic has gained much attention of researchers as we are rapidly entering the world of self-driving cars and these kinds of failures can causes major hurdles in bringing technologies to the mainstream society. Some papers [5] show that one-pixel change is enough to force a misclassification. This can be done using CNNs or GANs [5], and even uses evolutionary algorithms in some cases [6]. We want to try to this for ourselves and explore some possible reasons to explain this phenomenon. We plan on trying to work with GANs, something outside both of our machine learning experience.

Datasets

1. German Traffic Sign Recognition Benchmark Dataset [2]
   Published by researchers at the Ruhr-Universität Bochum, Germany in 2011 for the International Joint Conference on Neural Networks (IJCNN). The dataset has the following properties:
   - Single-image, multi-class classification problem
   - More than 40 classes
   - More than 50,000 images in total

The images in this dataset are ~ 32x32 images of road signs. Below are a few samples:

Hypothesis:

Deep neural networks have been widely used for classifying images with the high accuracy. This project is our quest to explore the tolerance and identify particular weaknesses of image classifying networks to different types and degrees of structured, random and specialized noises. The addition of the noise is almost indistinguishable to the human eyes but surprisingly it can completely fool neural networks. Moreover, another important point which drives this research is its application in a real-time scenarios e.g. If a self-driving car just ignores or misclassifies a stop-sign or a pedestrian because its neural networks have mis-classified its sensors input images. With this we hypothesize that:

1) It should be possible to identify certain noise thresholds beyond which a network starts regularly failing,

2) Similarly, certain noise patterns may be more disruptive to the classification process of trained networks and even pre-trained networks.

3) Training GANs on random and structured noise datasets may help us understand the failure of the machine learning models or the solution to overcome this problem.

Min Goals:

Our minimum goals are to train a CNN to high accuracy on the above-mentioned traffic signs dataset and then elicit a misclassification from this high accuracy network by augmenting its expected input images with random and structured noise. One question we would like to see answered what are the possible effects of slightly changes in an image vs huge changes on its classification?

After we have done this, we would need to learn how to build and use GANs and how to tweak them. Building the Deconvolutional CNNs for the Generator and CNNs for the Discriminator and updating both together is possible with the Keras/TensorFlow API.

Possible Experiments

1. CNNs:
   a. Train CNN model to high accuracy on part of our dataset.
   b. Manually write a program to add random noise to our test dataset.
   c. Write a program to add directed/targeted forms of noise on the images: (colored patches, blurring, black rows/cols of pixels.)
   d. Compare the classification accuracy of the classifier with varying degrees of the above (and below) mentioned types of noise.
2. Pretrained Models:
   a. Use transfer learning to finetune a pretrained (e.g. MobileNetV2) model for the classification of images in the GTSRB dataset. We can then take advantage of these learned feature maps without having to start from scratch by training a large model on a large dataset.
   b. Run experiments Similar to the CNN experiments on the pretrained network.
3. GANs:
   a. Write a simple Deep Convolutional GAN with the aim of generating images similar to a given class X.
      i. We will need to train two models simultaneously. One would be generator model which can produce images which look real but in essence they are fake, another is discriminator which learns to tell about real images. During training this process will be balanced when the discriminator is not able to differentiate between actual and fake images.
4. Try to experiment with how the inputs to the generator can be more than just random input.
   a. Random Noise
   b. Gaussian Noise
   c. Salt-and-pepper noise
   d. Periodic noise

    e. Poisson(gamma) noise
5. While experimenting with adding noise to our images, we thought of trying varying both the types of noise and the amount of noise in our experiments e.g.:
    a. 0% - 50% - 85% noise in three different experiments.
6. Working with ~50,000 images is a large task and running each of the three experiments sequentially may take a long time. Instead, we propose writing Map-Reduce jobs to preprocess the images in bulk and create the three required datasets for quick usage later.
7. GAN inputs are usually random noise. This allows the generator to explore various types of image generation combinations better. Our research has led us to a specialized Image to Image GANs (Fig 2.) that take Images as inputs (Latent space in Fig 2.), but those are usually used to train networks to transform images from one style to another [10,13].

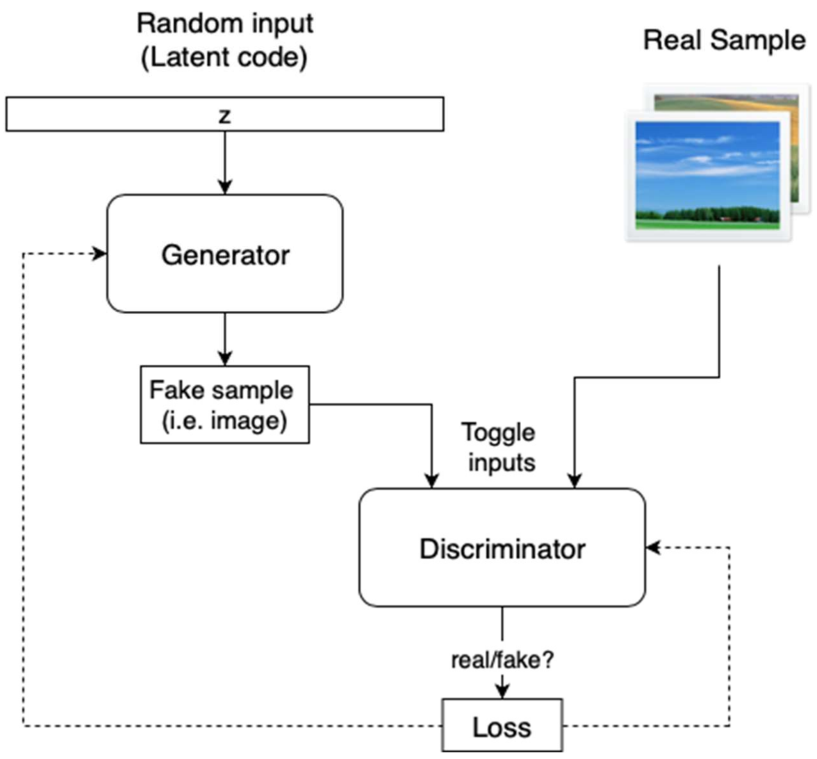

Figure 1: GAN Description: Ref[8]

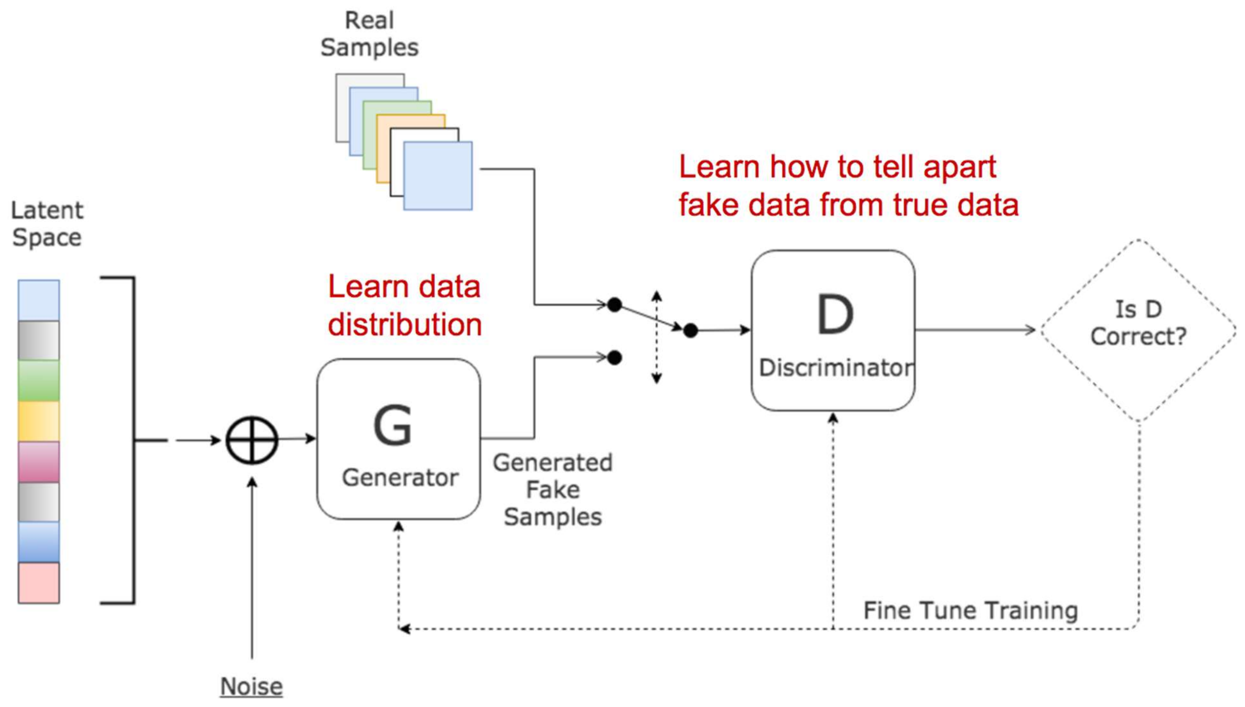

*Figure 2: **Not Random** Inputs to GANs [17]*

Qualitative Metrics:

True Positive Rate, False Positive Rate, Precision, Recall, F1 score, SKLearn's Classification report, Saliency diagrams, Inception score [19], Frechet-Inception-distance [18], Kernel-Inception-distance [18].

Things to focus on:

1. Using this new type of network for the first time (implemented using the tf/keras API).
2. Short and simple implementations of the networks.
3. Implementation of the neural network in such a way that training of the model leads to homogeneous responsibilities of each neuron rather than overloading few neurons with the dominating load. This can be achieved by experimenting with Dropout layers in our network.
4. Hyper parameter tweaking and comparison experiments.
5. Large dataset means that the train-test split must be carefully calculated. It may train the network to perform better but will take a long time and may be prone to overfitting issues.
6. Well documented code.

Points of Failure

- We're not exactly sure how the classification/misclassification error is accommodated in the generator for it to generate better images, we plan to figure that out.
- We may not be able to incorporate more than random noise into the input of the Generator.
- One possible point of error is the development of a loss /cost function for the GAN. We usually plan to use categorical cross-entropy to measure correctness of classification, but we are not sure what loss function we will have to develop or use to promote the generator to generate images to better force misclassification of the images.
- We have read about certain drawbacks faced by all GANs[16] that hinder the generator's learning greatly. Some of these are:

    - Vanishing gradient
    - Mode collapse

Successful Outcomes

- Observe the robustness of image classifying networks to different types of noise. Note any weaknesses to particular forms of noise.
- Identifying minimum thresholds for the noise level to cause failure depending on the type of the noise.
- Develop an effective generative adversarial network capable of both the task of generator and discriminator which would help to generate noisy images to a level where discriminator fails to classify images correctly.
- Implementing or identification of a cost/loss function for the adversarial network to evaluate how well the classification of the noisy images.
- Evaluate performance of the network specific to noise type and range with the abovementioned metrics.

References and Related Work

1. Reference CNN architecture for road sign classification - navoshta/traffic-signs: Building a CNN based traffic signs classifier. (github.com)
2. German Traffic Sign Recognition Benchmark (GTSRB) – Institut fur Neuroinformatik, Ruhr-Universität Bochum, published dataset for IJCNN 2011 Competition. Available at German Traffic Sign Benchmarks (rub.de)
3. Metrics for Multi-Class Classification: An Overview - Margherita Grandini, Enrico Bagli, Giorgio Visani
4. Zeiler, M. D., Krishnan, D., Taylor, G. W., & Fergus, R. (2010). Deconvolutional networks. In 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2010 (pp. 2528-2535). [5539957] (Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition). https://doi.org/10.1109/CVPR.2010.5539957
5. J. Su, D. V. Vargas and K. Sakurai, "One Pixel Attack for Fooling Deep Neural Networks," in *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828-841, Oct. 2019, doi: 10.1109/TEVC.2019.2890858.
6. https://github.com/Hyperparticle/one-pixel-attack-keras - Reference code for paper [5]
7. Tensorflow Documentation for DCGANs - https://www.tensorflow.org/tutorials/generative/dcgan
8. Style-based GANs – Generating and Tuning Realistic Artificial Faces | Lyrn.AI
9. de-Maupeou-D'AbleigesS-One-noise-to-fool-them-all.pdf (imperial.ac.uk)
10. [1905.13456] Combining Noise-to-Image and Image-to-Image GANs: Brain MR Image Augmentation for Tumor Detection (arxiv.org)
11. [1701.00160] NIPS 2016 Tutorial: Generative Adversarial Networks (arxiv.org)
12. CSC321 Lecture 22: Adversarial Learning (toronto.edu)
13. Sketch to Image translation paper sketch-image-translation.pdf (lisa.fan)
14. Generating Adversarial Noise for GANs TensorFlow-Tutorials/12_Adversarial_Noise_MNIST.ipynb at master · Hvass-Labs/TensorFlow-Tutorials · GitHub
15. GAN Explained | Papers With Code
16. From GAN to WGAN (lilianweng.github.io)
17. http://www.kdnuggets.com/2017/01/generative-adversarial-networks-hot-topic-machine-learning.html
18. GitHub - taki0112/GAN_Metrics-Tensorflow: Simple Tensorflow implementation of metrics for GAN evaluation (Inception score, Frechet-Inception distance, Kernel-Inception distance)
19. GitHub - zurutech/ashpy: TensorFlow 2.0 library for distributed training, evaluation, model selection, and fast prototyping.