WWW: Widescale Weather Witchcraft

Ritvik Goradia[1], Kinjalk Parth[2] and Animesh Singh[3]

[1] *Department of Electrical and Computer Engineering*
E–mail: *goradia3@illinois.edu*

[2] *Department of Computer Science*
E–mail: *kinjalk2@illinois.edu*

[3] *Department of Mechanical Engineering, UIUC*
E–mail: *animesh8@illinois.edu*

SUMMARY: We present WWW, an easy and reliable way for users to improve the training and validation of their lane tracking models. Users are able to drop in their road images with certain weather conditions and translate the scene of the image into any of 8 different conditions.

Key words. GAN, Autonomous Vehicle, Data Augmentation, Style transfer, Weather augmentation

1. INTRODUCTION

Accurate lane tracking algorithms are the backbone of autonomous vehicles. Tracking must be done correctly and efficiently. Very often, a model might perform well on road images that have sunny, clear weather, but fail in less ideal scenarios. This includes shadows being cast on the road, snow covering some lanes, dark, and foggy conditions. It would thus be very useful to be able to train and validate a prospective model on all these conditions. An issue that comes up when dealing with this problem is the lack of image datasets that cover all conditions. Further, it is quite rare to find the same road or scene in multiple conditions.

Style transfer GANs open the door for a neat solution to this problem. We could take existing training and validation sets and use these models to augment and generate more expansive, thorough data. Our proposal is an open-source program that allows users to drop in images and convert them to any of the aforementioned conditions. This conversion is done using an amalgamation of different models pre-trained with our weights. The user is thus able to not only create new training images but also validate their models on different conditions.

2. DETAILS OF THE APPROACH

Generative Adversarial Networks use a CNN backbone to extract features of the images and then replicate those features on a different set of image. There are two parts for a GAN - Generator and Discriminator. A generator is the engine that is responsible for generating fake images and a discriminator is responsible for classifying the image as fake or real. The goal of GAN is to make the fake image generator so good that the descriminator can't differentiate it from a real image. The training for GAN is very critical as if one of the model is learning at a higher rate, then the other model will saturate. For example, if the discriminator becomes really efficient, then the generator won't be able to learn as all its generated image would be classified as fake. We are using CycleGAN and Pix2Pix models. CycleGAN is used for training the weather GAN and Pix2Pix is an exploration of the use of GAN on segmentation.

2.1. CycleGAN

CycleGAN, short for Cycle-Consistent Generative Adversarial Network, is a type of generative adversarial network (GAN) architecture designed for image-to-image translation tasks. It aims to learn mappings between two domains without requiring paired data for training. It consists of 3 main parts:

1. Generator Networks (G):
CycleGAN consists of two generator networks, often denoted as corresponding to the translation from domain A to domain B and from domain B to domain A, respectively. Each generator takes an input image from one domain and generates a corresponding output image in the other domain.

2. Discriminator Networks (D):
CycleGAN also includes two discriminator networks,

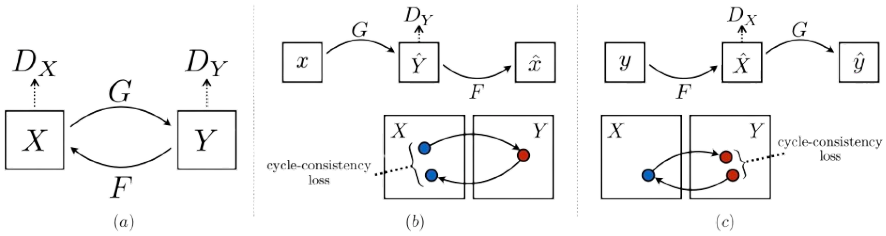

Fig. 1: CycleGAN architecture: (a) The model contains two mapping functions G : X → Y and F : Y → X, and associated adversarial discriminators DY and DX. DY encourages G to translate X into outputs indistinguishable from domain Y , and vice versa for DX and F. To further regularize the mappings, two cycle consistency losses that capture the intuition that if we translate from one domain to the other and back again we should arrive at where we started: (b) forward cycle-consistency loss: x → G(x) → F(G(x)) x, and (c) backward cycle-consistency loss: y → F(y) → G(F(y)) y

often denoted as corresponding to the discrimination of fake images generated by the generators and real images from the respective domains. The discriminators are trained to distinguish between real and fake images and provide feedback to the generators to improve their image generation process.

3. Loss:
The loss function of CycleGAN involves the normal adversarial loss of GANs along with cycle-consistency loss, which enforces the translation between domains to be cycle-consistent. Cycle-consistency loss ensures that if an image from domain A is translated to domain B and then back to domain A, the resulting image should be similar to the original input image.

During training, the generators and discriminators are trained alternately similar to a normal GAN with the cycle consistency loss applied on the Generator. The cycle-consistency loss term acts as a regularization term, ensuring that the learned mappings are consistent and meaningful. Fig 1 depicts how the the two generators and discrimators are trained along with an intuitive understanding of Cycle-consistency loss.

CycleGAN achieves extremely good results with very less training data (as evident in the results that we will showcase later in this report) due to the consistency constraints. As it trains two GANs at the same time, we can multiply the results as we can convert the dataset of A into B and B into A. This will result in not only doubling the amount of data but also keeping the ratio between the data in different weather the same. Let's say we have 4 different datasets with Fog, Snow, Day and Night. We can actually increase the dataset by 12x using the different combinations of the above models.

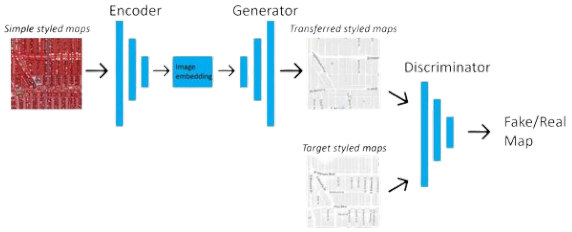

Fig. 2: Pix2Pix architecture

2.2. Pix2Pix

Pix2Pix, short for "Image-to-Image Translation with Conditional Adversarial Networks," is a deep learning architecture used for paired image-to-image translation tasks. The characteristics of the model are :-

1. Generator Network (G):
Pix2Pix consists of a generator network G that takes an input image from one domain and generates a corresponding output image in another domain. The generator typically follows an encoder-decoder architecture(U-Net in our case), along with skip connections, facilitating the flow of low-level details from the input to the output.

2. Discriminator Network (D):
The discriminator receives pairs of images in Pix2Pix rather than a single image. The pair consists of an input image from one domain and its corresponding output image from the generator or a real image pair from the dataset. The discriminator is trained to distinguish between real image pairs and fake image pairs generated by the generator.

3. Conditional Adversarial Loss:
The key innovation of Pix2Pix is the use of conditional adversarial training, where both the generator and discriminator are conditioned on the input image. The generator aims to produce output images that are indistinguishable from real images in the target domain, while the discriminator aims to distinguish between real and fake image pairs. This adversarial training setup encourages the generator to produce high-quality translations that capture the semantic content and structure of the input images.

The conditional adversarial training process encourages the generator to learn meaningful mappings between input and output image domains, resulting in realistic and visually appealing translations. Overall, Pix2Pix is a versatile architecture for paired image-to-image translation tasks, enabling tasks such as colorization, style transfer, semantic segmentation, and more, by learning mappings between input and output image domains from paired training data.

2.3. FoggyGAN

Foggy GAN is also an implementation of CycleGAN along with a few additional aspects :-

1. To achieve different intensities of fog, the generator gets the input images along with the intensity as its parameters.
2. The discriminator also has an intensity parameter
3. The generated image is penalized if the intensity of the fog does not match the intensity specified. This loss is in addition to the adversarial and normalization loss.

The dataset for FoggyGAN is not freely available and hence we are using pre-trained weights.

2.4. Application

The novelty in our approach is that we have taken the models defined above and applied them to our own datasets in order to use them in a very different context as what was originally envisaged.

As mentioned in the introduction, we want to apply all types of weather transformations to road images. Thus we have built an application where users an input 1 or more images/videos and simply select the condition they would like to transform their scene into. Fig 3 shows what the front end of this application looks like. The left side is the image the user has input and at the bottom there is a panel with the possible transformations available for the user to apply i.e. 8 different weather/time-of-day transformations along with image segmentation and viewing previously generated images.

On the backend, this app loads the models with our own pretrained weights and then interfaces with these models to generate the desired outputs.

3. RESULTS

We have compiled the results of the different trained models in Fig 4. All the datasets used are listed in the Dataset section. The goal of the project was converting the driving videos to different weather conditions. To that end, we actually used a dashcam video and applied our model to individual frames. We have presented the video in the YouTube link for the project[11].

The results shown in Fig 4 illustrate how different translations appear when applied to different kinds of images. On the leftmost side we have the original images that were inputs to the application, and to the right of them we have the outputs generated by the app labelled with the name of the transformation applied to them.

4. DISCUSSION AND CONCLUSION

Through the course of this project, we have successfully shown how Style Transfer GANs can be applied to the world of autonomous systems – an application which is seldom explored by researchers. We have shown that systems like ours can be used to augment training and validation datasets in order to create new images covering all types of conditions. Training on these more expansive and complete datasets will result in more robust lane-tracking algorithms that are less biased to weather conditions.

While implementing these models, we came across a few issues. The first was that it is quite hard to come up with any meaningful metrics in order to evaluate style transfer GANs. One possible metric we considered was the popular Inception Score (IS) which scores a generated image based on its quality and diversity. However, this too comes with several limitations, some of which are the lack of training classes – IS is trained on an image classification network of about 1000 classes and would thus provide an artificially low score for a generated image that does not fall in any of those classes, and IS only works on small, square images (roughly 300x300 pixels) so in order to apply this to road

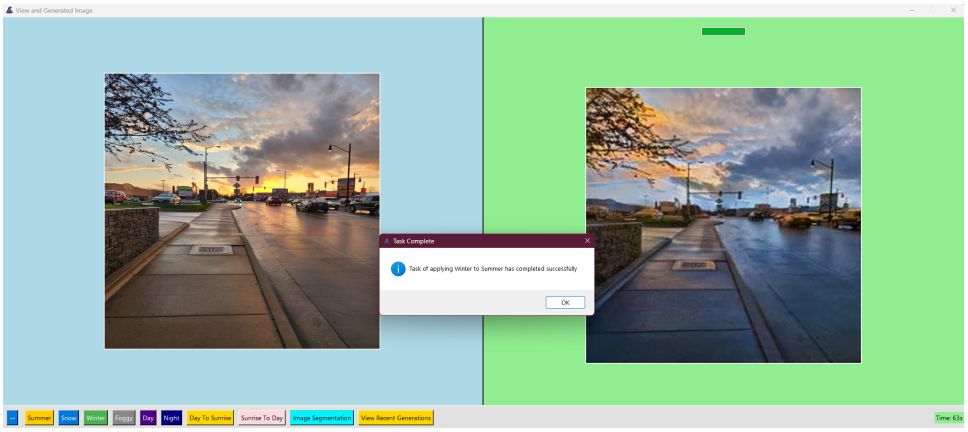

Fig. 3: View of our application where the user has applied a Winter transformation on the input image

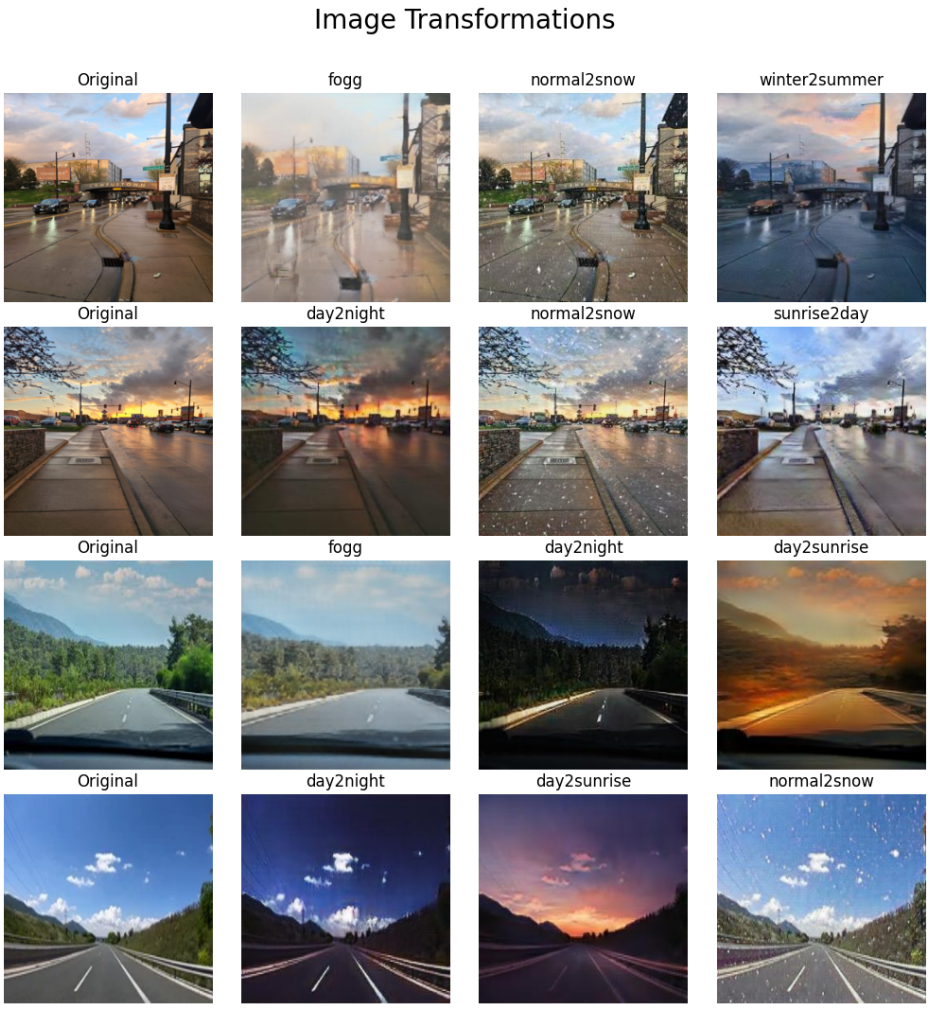

Fig. 4: Examples of results using WWW

images (which tend to be landscape) means they would need to be significantly cropped.

Fig. 5: Example of bad snow transformation where snowflakes are only uniformly applied throughout image

Fig. 6: Example of bad sunrise transformation with unnaturally orange road

Fig. 7: AUGAN Night → Day ouput for non-road image

Another significant challenge we faced was the lack of labelled road datasets publicly available. The only comprehensive dataset for road images we were able to find was the Berkeley DeepDrive bdd100k dataset [4] (consisting of 100k road images in different conditions). The problem with this though was that this set is not labelled. In the end, we trained the Pix2Pix model on the Cifar10 dataset and CycleGAN models on different weather datasets, all datasets which are not specific to roads. The consequence of this can be seen in the images we generate. For example the snow transformations (Fig 5) simply apply snowflakes evenly throughout the picture whereas a real snowy road to have some clumps of snow piled up on the side and darker, mushy snow in between lanes. Another example is the sunrise transformations Fig 6 – this model was trained on images of the actual orange sun during sunrise, not how a road appears during sunrise. Hence we can see the generated images are unnaturally orange for a road.

With a labelled dataset specific to roads, many of these errors would be resolved and the generated images would look significantly more accurate. Dataset is one of the most important aspect of the training as can be seen from Fig 7. This result is from AuGAN another model that we tried that only uses day and night images from road. In the image of the grass, the model added some cars assuming that the grass fields corresponds to a road. This clearly states how important is it to have a good representative dataset for training GANs.

5. FUTURE WORK

One useful expansion of this project would be to apply the same principles of image augmentation in different conditions to other types of sensors as well such as Radar and Lidar. Just as foggy, rainy, snowy conditions alter the optical data received by an autonomous car, these conditions cause changes to the other sensors as well.

Modern autonomous vehicles use several different sensors in their perception of their environment. The perception data from these sensors are used for lane tracking, object detection, and state estimation models. Applying a similar system to ours to all the different sensors involved in perception would result in models that are far more robust to weather and situational biases, and will make these vehicles safer for everyone on the road.

Diffusion models are another aspects that we can explore on this front. The reason we did not try them is due to computational constraints along with the low amount of data. Even if we trained a diffusion model to generate the data, it is not guaranteed that the generated data would be similar to the input data. GAN does a more consistent mapping between input and output. An interesting application - the corner cases encountered in real-life could be made more pronounced in the dataset if the GAN models are able to generate these scenarios along with inputs from diffusion models.

Another important aspect of the problem is decreasing the inference time of the Model so that is can be run on videos in real-time. We can also add a layer on top of the code that allows the user to add lane detection algorithm, object detection etc on the gen-

erated image/video for validation of their algorithm along with the generation of new data.

6. STATEMENT OF INDIVIDUAL CONTRIBUTIONS

The project was highly collaborative where individuals from 3 different department came together to work on the project.
1. Kinjalk - Involved in the app development and seamless integration of different models
2. Ritvik - involved in understanding the code and architecture
3. Animesh - involved in dataset collection, model training, fine tuning, autonomous driving data research

We believe the grade for this project should be divided evenly i.e. w = [0.33, 0.33, 0.33].

7. DATASETS

We are using the BDD100k[4] dataset for AU-GAN. Foggy GAN and Cycle GAN have a dataset directly associated with the pretrained weights. For example we are using summer2winter_yosemite[5] dataset for CycleGAN. It is a really efficient model that could help us switch the results of the model by using a different weight. FoggyGAN uses selective images from many different datasets - Cityscape[6], FoggySynscapes[7] and RESIDE[8]. In addition for training normal to snow, we are using Snow100k[9] dataset. For day-night-sunrise conversion we are using [10]. For segmentation we are using the subset of Cifar10 used in MP5 of our Assignment.

8. VIDEO DESCRIPTION

A narrative of the whole project is available at Youtube[11]. We have also included dynamic results on video.

9. GitHub Repository

The GitHub repository for the project is at https://github.com/MasterKinjalk/Gan-App-For-Data-Generation

10. REFERENCES

1.https://github.com/jgkwak95/AU-GAN
2.https://arxiv.org/pdf/2103.05422.pdf
3.https://github.com/ghaiszaher/Foggy-CycleGAN
4.http://bdd-data.berkeley.edu/
5.https://www.kaggle.com/datasets/balraj98/summer2winter-yosemite/code
6.https://www.cityscapes-dataset.com/
7.https://github.com/MartinHahner/FoggySynscapes
8.https://sites.google.com/view/reside-dehaze-datasets/reside-standard?authuser=0
9.https://sites.google.com/view/yunfuliu/desnownet
10.https://www.kaggle.com/datasets/aymenkhouja/timeofdaydataset?select=nighttime
11.https://www.youtube.com/watch?v=TipmDBbVJ8o

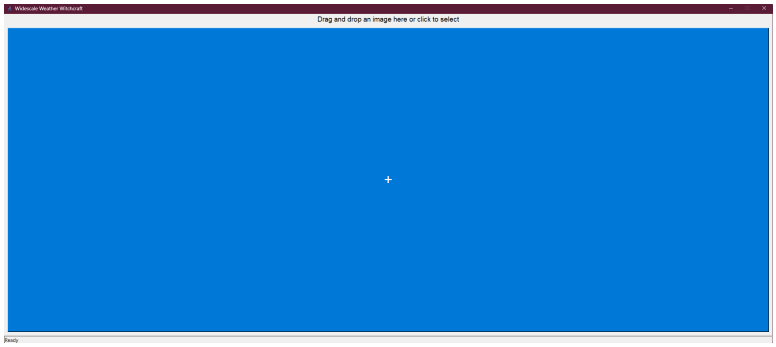

Fig. 1: View of the Home/Landing page for the application. Users can drag-and-drop images or videos here

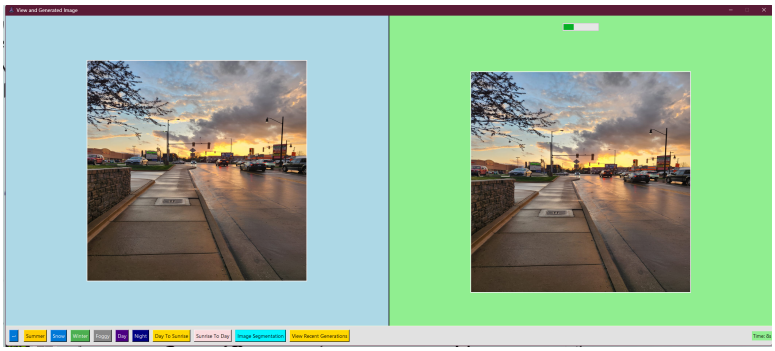

Fig. 2: Image showing the loading bar on the app which lets the user know how much time is remaining till all images are transformed

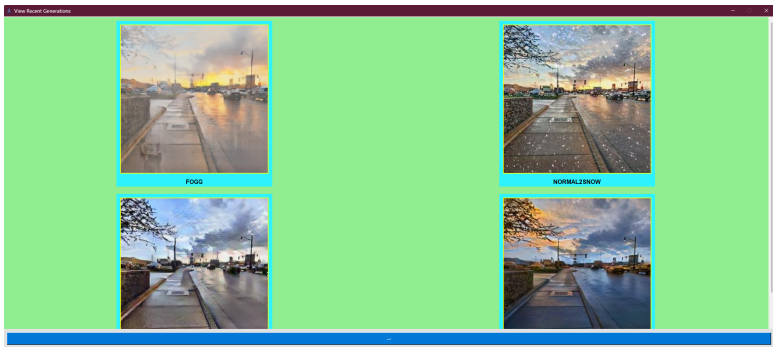

Fig. 3: Users are also able to view some of their previously generated transformations