# Image Translation and UDA

**Reinelle Jan Bugnot**
MSAI,
Matric No: G2304329L
email: bugn0001@e.ntu.edu.sg

**Rohin Kumar Maheswaran**
MSAI,
Matric No: G2303513K
email: rohinkum001@e.ntu.edu.sg

**Aradhya Dhruv**
MSAI,
Matric No: G2303518F
email: ar0001uv@e.ntu.edu.sg

*This project explores the implementation and enhancement of image-to-image (I2I) translation techniques, focusing on the CycleGAN framework. The first task involves training an I2I translation network using datasets such as GTA5 as sources and Cityscapes for semantic segmentation. The project report details the implementation process, highlighting major constraints observed in the image translation network and analyzing the translated images. In the second task, the project delves into unsupervised domain adaptation (UDA) via I2I translation. Leveraging the image translation model from the first task, a comparative analysis is conducted between a Source-only semantic segmentation model trained on labeled source data and evaluated on target data, and a domain adaptive model trained with translated source data. The project report provides insights into the performance differences between these models and explores alternative translation networks for potential UDA improvements*

*Keywords: UDA, Image Translation, CycleGAN*

## 1 Introduction

Recent computer vision research has been captivated by the intersection of image-to-image (I2I) translation and unsupervised domain adaptation (UDA). CycleGAN, a pioneering unsupervised image translation method rooted in generative adversarial networks (GANs), stands out for its seamless cross-domain translation without the need for paired training data. The project's initial phase focuses on synthesizing images from disparate sources, such as the Cityscape and GTAV datasets. Precision in transformations and the preservation of crucial visual features are central to achieving faithful translations across inherently different visual landscapes. This involves fine-tuning hyperparameters, optimization techniques, and selecting loss functions to strike a delicate balance between realism and fidelity in translated images. In the second phase, our exploration delves into UDA powered by I2I translation, comparing two semantic segmentation models—one relying solely on source data and the other, trained on translated source data, exhibiting domain-adaptable capabilities. The project report meticulously dissects and contrasts these models, revealing the factors influencing their diverse performance.

Datasets Used: We have used Cityscapes and GTA5 Dataset to implement the requirements of the project

**1.1 Cityscapes:.** The Cityscapes Dataset focuses on semantic understanding of urban street scenes, incorporating a set of design choices to enhance its applicability.

**Features of the Dataset:**

- **Number of Classes:** Cityscapes includes annotations for 30 distinct classes. Refer to the Class Definitions for a comprehensive list.
- **Diversity:** Data is collected from 50 cities, spanning several months (spring, summer, fall), and includes various daytime and weather conditions. Manually selected frames ensure diversity in dynamic objects, scene layouts, and backgrounds.
- **Volume:** The dataset comprises 5,000 annotated images with fine annotations and an additional 20,000 annotated images with coarse annotations.

**1.2 GTAV :.** The GTA5 dataset is a collection of 24,966 synthetic images meticulously annotated at the pixel level for semantic understanding. These images are generated within the expansive open-world environment of the popular video game, Grand Theft Auto 5 (GTAV). All images are captured from the perspective of a car navigating the virtual streets of American-style cities within the game.

**Features of the Dataset:**

- **Number of Images:** The dataset comprises a total of 24,966 synthetic images, each offering a unique perspective of the virtual urban landscape.
- **Rendering Source:** The images are rendered using the GTAV game engine, ensuring a realistic representation of urban scenes. This unique source of synthetic data enables the exploration of machine learning models within a controlled yet diverse virtual environment.
- **Semantic Classes:** There are 19 semantic classes defined within the dataset, aligning with the classes of the Cityscapes dataset. This compatibility allows for seamless integration and comparison of models trained on both datasets.

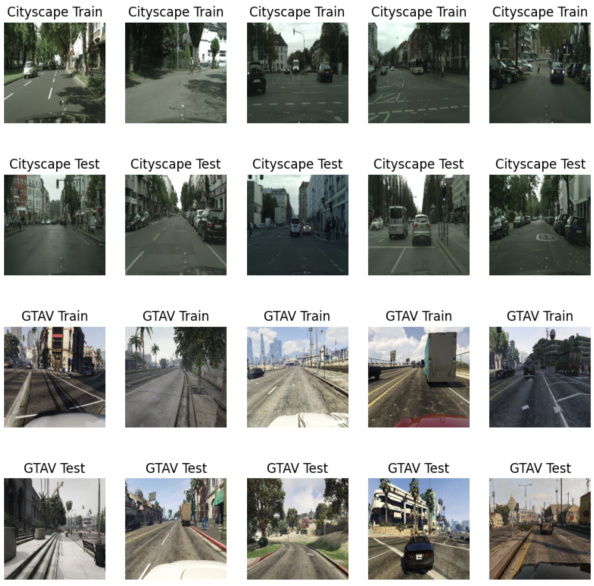



**Fig. 1 Cityscape and GTAV Dataset**

Version 1.18, November 18, 2023

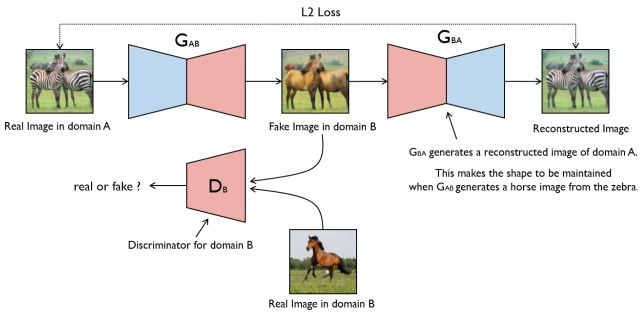



**Fig. 2 CycleGAN Overview**

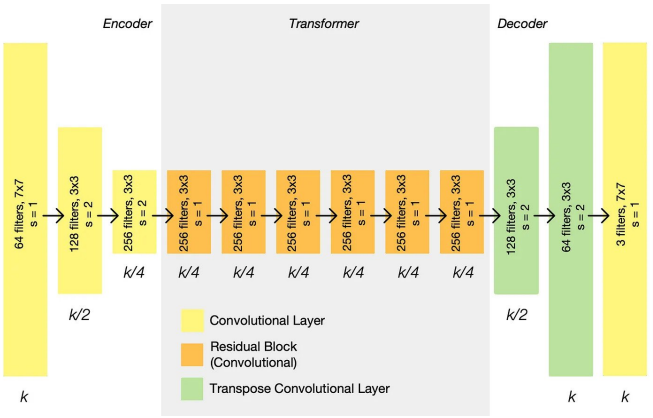



**Fig. 3 Generator Architecture**

## 2 CycleGAN

CycleGAN, which stands for Cycle Generative Adversarial Network, is a ground-breaking approach in unsupervised image translation in the field of computer vision. Its architecture, presented in 2017 by Jun-Yan Zhu et al., signified a paradigm shift by avoiding the requirement for paired training data, which was a restriction in earlier techniques.

**GANs (Generative Adversarial Networks):** The GAN architecture is at the heart of CycleGAN. It consists of two neural networks, a generator and a discriminator, which are undergoing adversarial training.The generator attempts to generate realistic images, whereas the discriminator attempts to distinguish between genuine and generated images.

**Cycle-Consistency:** What distinguishes CycleGAN is the incorporation of cycle-consistency, a unique idea that ensures the validity of the picture translation process. CycleGAN enforces a self-consistency constraint by establishing a cyclic process in which translated pictures can be restored to their original form and vice versa. This not only improves the translated images but also makes training without paired datasets easier.

**Loss Functions:**

- **Adversarial Loss:** Guides the generator to produce realistic images that deceive the discriminator.
- **Cycle-Consistency Loss:** Enforces cycle-consistency between domains, improving translation quality.
- **Identity Mapping Loss:** Ensures translated images, when reverted, remain unchanged for model stability.
- **Adversarial Training:** Generator aims to deceive the discriminator, which aims to distinguish real from generated images.
- **Cycle-Consistency Training:** Involves mapping images across domains and back, promoting translation consistency.

## 3 Image-to-Image (I2I) Translation

The initial Image-to-Image translation phase systematically loads and preprocesses data from Cityscape training and testing datasets, along with the GTAV dataset. Dataset division and shuffling eliminate biases. Generator and discriminator models, critical in CycleGAN, are created. The generator, handling domain translation, features coordinated convolutional layers, batch normalization, and activation functions for effective feature extraction. Simultaneously, the discriminator, distinguishing real from generated images, employs convolutional layers and leaky rectified linear units (Leaky ReLU) for attribute discernment.

CycleGAN, employing generator_XY and generator_YX, translates between Cityscape and GTAV datasets. Training, using batches of cityscape and GTAV images, iteratively refines generators with adversarial and cycle consistency losses. Discriminators, discriminator_X and discriminator_Y, contribute to adversarial training by distinguishing real and generated images. This architecture ensures effective cross-domain image translation.

During testing, the trained CycleGAN model employs generator_XY and generator_YX for inference on cityscape and GTA5 test images. Generator_XY transforms Cityscape images to GTAV style, while generator_YX generates Cityscape-style images from GTAV images. The model adeptly generates realistic images in the target domain, demonstrating successful cross-domain translation.

## 4 Architecture Description

### Generators

- **Residual Blocks** The generator's foundation lies in ResNet-style residual blocks, a well-established and effective approach in image-to-image translation tasks. Each residual block comprises two convolutional layers accompanied by batch normalization and rectified linear unit (ReLU) activation functions. These blocks play a crucial role in enabling the model to capture and propagate essential features, mitigating challenges associated with the vanishing gradient problem.
- **Downsample and Upsample Blocks** Incorporating downsample and upsample blocks, the generator adeptly manages the reduction and augmentation of spatial dimensions during the image translation process. The downsample block utilizes convolutional layers, along with batch normalization and ReLU activation, to decrease spatial resolution. Conversely, the upsample block employs transposed convolutional layers to increase spatial resolution effectively
- **Final Block** The final block of the generator holds the responsibility of generating the translated image. This block features a convolutional layer with a hyperbolic tangent (tanh) activation function, effectively scaling pixel values to the range [-1, 1], making them suitable for real images.
- **Discriminators** The discriminator architecture adopts a patch-based strategy, aiming to discriminate between real and synthetic images in localized regions. Utilizing convolutional layers with leaky ReLU activation functions and integrating dropout layers for regularization, the discriminator enhances its capability to capture hierarchical features by increasing the number of filters with each downsampling block.
- **Training Procedure** The training process unfolds through iterative batch processing of cityscape and GTA5-style images. Generators and discriminators are updated via backpropagation, leveraging the calculated gradients from the specified loss functions. Training encompasses the minimization of adversarial, cycle consistency, and identity losses for generators, along with adversarial losses for discriminators.

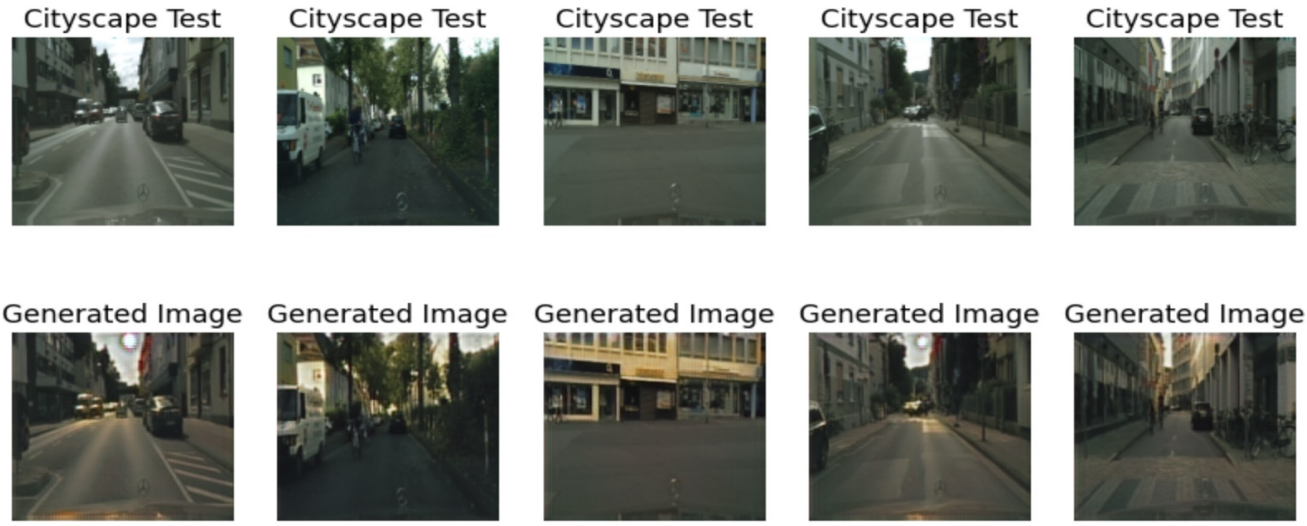



**Fig. 4 Translated Images Generated by CycleGAN (Real Cityscape Images to GTA Style Cityscape images)**

**4.1 Insights Acquired during CycleGAN Implementation.** We encountered several challenges in implementing the CycleGAN model. One significant hurdle was determining the optimal learning rate, as we observed instances where the discriminator would overpower the generator, resulting in the generation of poor-quality images. Addressing this issue was crucial for achieving a balanced training process. Additionally, selecting an appropriate loss function posed a challenge. While we initially used Binary Cross Entropy loss, it became evident that this loss function is more suitable for classification problems and less effective for generative networks. Subsequently, we explored the integration of dropout layers and gradient penalty into the discriminator model. This adjustment aimed to prevent the discriminator from dominating the generator, thereby promoting a more stable and effective training dynamic.

## 5 Unsupervised Domain Adaptation via I2I Translation

Unsupervised Domain Adaptation (UDA) is a machine learning tecnique that addresses the challenge of adapting a model trained on a source domain to perform well on a target domain where the data distribution may be different. In the context of supervised learning, models are typically trained on labeled data from a source domain and may struggle when applied directly to a target domain with different characteristics. In the case of this project, our source domain is the Cityscapes dataset while our target domain is the GTA V dataset. Both datasets feature real and synthetic images of a city space (particularly in the perspective of a car/driver), respectively. Unsupervised domain adaptation aims to mitigate this issue by leveraging unlabeled data from the target domain to improve the model's performance.

In this project, we want to observe how UDA affects the performance of a semantic segmentation model. Particularly, we will be training two models:

- A source only model trained with the labelled source data and evaluated over the target data, and
- A domain adapted model trained with the translated source data and evaluated over the target data

Our hypothesis is that the model trained on the translated source data (Cityscapes -> GTAV) will perform better than the source-only model (Cityscapes) when tested on the target data (GTAV).

**5.1 Semantic Segmentation.** The first step in this process is to train a semantic segmentation model using the source data (Cityscapes). In this project, we decided to utilize a U-Net model to perform the segmentation task. U-Net models have proven to be effective for semantic segmentation tasks due to their architectural design, which combines a contracting path to capture context and a symmetric expansive path to enable precise localization. An example architecture of a U-Net model is shown in Fig. 5.

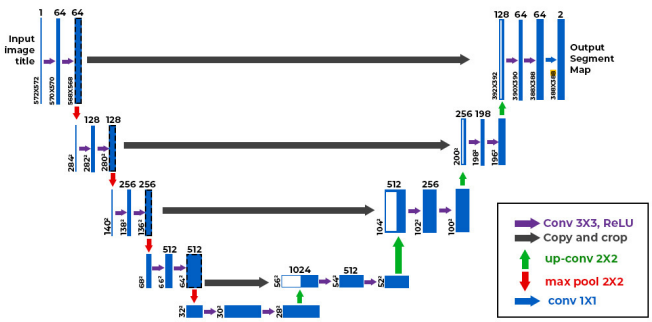



**Fig. 5 U-Net Architecture**

Prior to the training process, each input image is pre-processed and augmented in order to facilitate model with better bias and variance. Fig. 6 shows a snippet of the code that augments the input images via random cropping, horizontal flip, pixel value normalization, and image resizing. In the end, our input images are transformed into a much smaller (256, 256, 3) image with 3 channels. Shrinking the image this way prove to be more memory efficient and easier to train.

Our model adopts a 24-layer U-Net architecture, encompassing a total of 31,047,586 trainable parameters. The design follows the structure of a standard U-Net network, as depicted in Figure 5, where each subsequent convolutional layer compresses and expands accordingly. To mitigate overfitting, L2 Regularization is applied with a discount factor of 0.01 in the first and last 4 convolutional layers. Although dropout layers were initially considered, their impact on the model performance was found to be more pronounced than desired.

The dataset is divided into training (2082 images), validation (500), and testing (893) sets, which are then randomly shuffled before training. A batch size of 32 is employed, and the model undergoes training for 15 epochs. Figure 7 illustrates the loss curves

```python
def rand_crop(img, label):
    concat_img = tf.concat([img, label], axis=-1)
    concat_img = tf.image.resize(concat_img, [280, 560], method=tf.image.ResizeMethod.NEAREST_NEIGHBOR)
    crop_img = tf.image.random_crop(concat_img, [256, 256, 4])
    return crop_img[:, :, :3], crop_img[:, :, 3:]

def norm(img, label):
    img = tf.cast(img, tf.float32)/127.5-1
    label = tf.cast(label, tf.int32)
    return img, label

# Load image functions + Augmentations
def load_img_train(img, label):
    img = read_png(img)
    label = read_png_label(label)

    img, label = rand_crop(img, label)

    if tf.random.uniform(()) > 0.5:
        img = tf.image.flip_left_right(img)
        label = tf.image.flip_left_right(label)
    return norm(img, label)
```

**Fig. 6 Pre-processing Step for Segmentation Model**

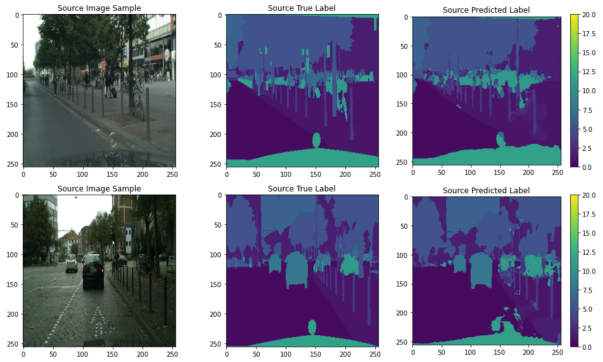



**Fig. 8 Sample Segmentation Output for U-Net trained in Source, tested in Source**

throughout the training process. It is evident that the training and validation loss converge, yet a slight gap between the learning curves hints at potential overfitting, suggesting a priority for future improvement in this project.

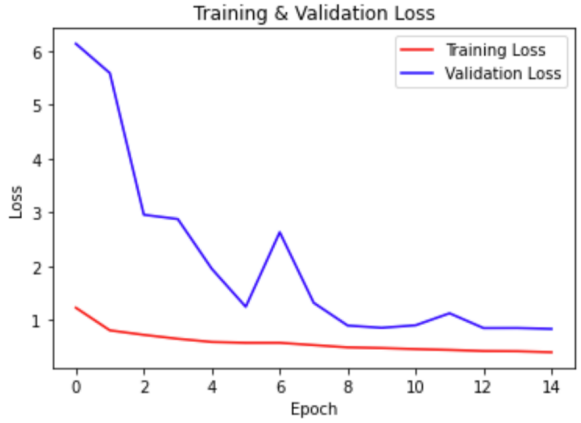



**Fig. 7 Training and Validation Loss Curves**

Figure 8 visually demonstrates the segmentation capability of the trained model, evaluated on the test data from the Cityscapes dataset (source data). This showcases the model's effectiveness in performing semantic segmentation in the images.

**5.2 Input Space and Label Alignment.** Before we can proceed with our cross domain analysis, it is important that both label spaces for source and target are properly aligned. That is, a class label that corresponds to 'road', for example, 8, must be the same for both source and target images. This is crucial, since if we are to evaluate our model on a different label space that where it was trained on, a label mismatch will occur, which will yield meaningless IoU, loss, or accuracy values. Given that the cityscapes dataset have 34 unique class labels (0-33) and the GTA V dataset have 19 class labels (0-20), we need to find a way to map the labels of our target labels from our source labels. To do this, we noticed that the original colored masks for both cityscapes and GTA V are derived from the same set of RGB values, as illustrated in Fig. **??**. That means we can map the class labels directly from RGB values, and the resulting mapping will be aligned regardless whether the label mask comes from the source (Cityscapes) data or the Target (GTA V) data.

**5.3 Source Only Segmentation.** Given that the model is trained in our source data (Cityscapes), when the model is evaluated directly on the target data (GTA V), we expect to observe a drastic decline in model performance since the pixel value domain on which the model is trained on is different that the pixel value domain of the testing (evaluation) data. That is, a pixel that represents a 'tree' object in a cityscapes image and a pixel that represents a 'tree' object in the GTA V image is likely operating in two different domains of pixel values, which skews the model inference capability, leading to poor evaluation. As shown in Fig. 10, the model is not able to properly segment the image.

**5.4 Domain Adaptive Semantic Segmentation.** One of the key challenges of performing domain adaptive semantic segmentation on two independent datasets with different label spaces is that the label space must match exactly between the two datasets. Without doing this, we cannot meaningfully measure performance metrics such as IoU and accuracy. In our particular implementation, our primary bottleneck was deriving the exact mapping between the two label spaces because the original papers that reference the two datasets did not explicitly mention the class label assignment for each label category. This information, unfortunately, is only accessible via a MATLAB file available in the main data repository of the GTA V file, which is what we used to derive the mapping logic as illustrated in Fig. 9.

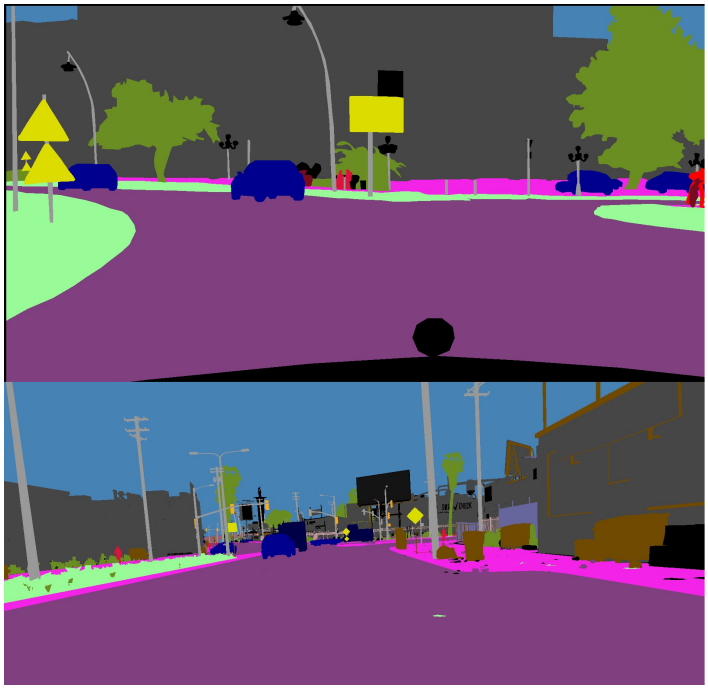
However, given our understanding of Unsupervised Domain Adaptation, we can make reliable assumptions as to what the output behaviour would be for a segmentation model trained on our source dataset (cityscapes) that is translated into the domain of our target dataset (GTA V), and evaluated on the target dataset (GTA V). Firstly, as shown in Fig. 4, we are able to effectively translate the source image into the domain of the target image. That means upon training, for each label category, the pixel value domain that the model was trained on should align with the pixel value domain that the model will be evaluated upon, hence, improving model inference.

However, despite domain translation and adaptation efforts, there may still be challenges related to domain gaps. If the source and target domains have significant differences in terms of lighting, textures, or other visual characteristics that are not well-captured by the translation model, the semantic segmentation model may struggle to generalize accurately. Striking a balance between adapting to the target domain and avoiding overfitting to domain-specific nuances is a key consideration. Care must be taken to ensure that the model generalizes well to various scenarios within the target domain.

## 6 Critical Analysis

While the CycleGAN architecture is well-structured, there are some considerations for improvement:

- **Normalization:** The implementation lacks explicit normalization of input images. Ensuring that pixel values are prop-



```
{(0, 0, 0): 0,
 (20, 20, 20): 1,
 (111, 74, 0): 2,
 (81, 0, 81): 3,
 (128, 64, 128): 4,
 (244, 35, 232): 5,
 (250, 170, 160): 6,
 (230, 150, 140): 7,
 (70, 70, 70): 8,
 (102, 102, 156): 9,
 (190, 153, 153): 10,
 (180, 165, 180): 11,
 (150, 100, 100): 12,
 (150, 120, 90): 13,
 (153, 153, 153): 14,
 (250, 170, 30): 15,
 (220, 220, 0): 16,
 (107, 142, 35): 17,
 (152, 251, 152): 18,
 (70, 130, 180): 19,
 (220, 20, 60): 20,
 (255, 0, 0): 21,
 (0, 0, 142): 22,
 (0, 0, 70): 23,
 (0, 60, 100): 24,
 (0, 0, 90): 25,
 (0, 0, 110): 26,
 (0, 80, 100): 27,
 (0, 0, 230): 28,
 (119, 11, 32): 29}
```

**Fig. 9 Label Space Mapping: (Top Left) Cityscapes Mask, (Bottom Left) GTAV Mask, (Right) Mapped RGB values to common class labels between Source (Cityscapes) and Target (GTA V)**

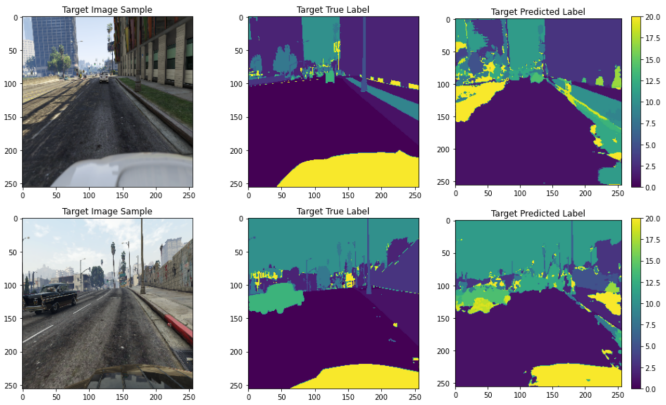



**Fig. 10 Sample Segmentation Output for U-Net trained in Source, tested in Target**

erly normalized to the range [-1, 1] is crucial, especially when using Tanh activation in the output layer.

- **Residual Block Count:** The number of residual blocks in the generator might be revisited. Depending on the complexity of the data, adjusting the count could enhance the model's ability to capture intricate features.
- **Batch Normalization Placement:** The placement of batch normalization before the activation function in the generator might be reconsidered. A common practice is to apply batch normalization after the activation for better information flow.
- **Learning Rate Scheduling:** Experimenting with learning rate scheduling could contribute to more stable convergence. Implementing a dynamic learning rate schedule might be considered for improved training efficiency.
- **Metric Monitoring:** The current training script prints limited training information. Incorporating more detailed metrics visualization, such as generator and discriminator losses over time, could provide valuable insights into the training progress.
- **Data Augmentation:** Depending on the dataset, incorporating data augmentation techniques during training might enhance the model's generalization capabilities.
- **Advanced Loss Functions:** Experimentation with advanced loss functions, such as Wasserstein GAN (WGAN) loss, could contribute to the stability and convergence of the model.
- **Instance Normalization:** Evaluating the use of instance normalization instead of batch normalization in the generator could be explored for potential improvements in style transfer tasks.

## 7 References


- Zhu, J.-Y., Park, T., Isola, P., Efros, A. A. (2017). *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*. arXiv preprint arXiv:1703.10593 [cs.CV]. Retrieved from https://arxiv.org/abs/1703.10593

- Brownlee, J. (2020, September 1). *How to Develop a CycleGAN for Image-to-Image Translation with Keras*. Generative Adversarial Networks. Retrieved from https://machinelearningmastery.com/cyclegan-tutorial-with-keras/
- Brownlee, J. (2019, August 17). *A Gentle Introduction to CycleGAN for Image Translation*. Generative Adversarial Networks. Retrieved from https://machinelearningmastery.com/cyclegan-introduction/
- *Cyclic Generative Networks*. "Others who with the help of their intelligence, transform a yellow spot into sun" — Diego Gomez Mosquera

**Generated Images by CycleGAN (Real Cityscape Images to GTA style Cityscape images)**

## List of Figures



## List of Tables