1. Report No. RailTEAM UNLV-8	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Real–Time Semantic Segmentation for Railway Anomalies Analysis		5. Report Date December 2022 6. Performing Organization Code:
7. Author(s) Paul Stanik III and Brendan Morris https://orcid.org/0000-0002-8592-8806		8. Performing Organization Report No. UNLV-8
9. Performing Organization Name and Address Department of Civil and Environmental Engineering University of Nevada, Las Vegas 4505 S. Maryland Pkwy. Las Vegas, NV 89154		10. Work Unit No. 11. Contract or Grant No. 69A3551747132
12. Sponsoring Agency Name and Address Office of Research, Development and Technology (RD&T) US Department of Transportation 1200 New Jersey Avenue, SE Washington, DC 20590		13. Type of Report and Period 14. Sponsoring Agency Code
15. Supplementary Notes		

16. Abstract

In the past few years, computer vision has made huge jumps due to deep learning which leverages increased computational power and access to data. The computer vision community has also embraced transparency to accelerate research progress by sharing open datasets and open-source code. Access to large scale datasets and bench-mark challenges propelled and opened the field. The autonomous vehicle community is a prime example.

While there has been significant growth in the automotive vision community, not much has been done in the rail domain. Traditional rail inspection methods require special trains that are run during down time, have sensitive sensing/imaging analysis equipment with high costs, or may require low speeds for analysis. In addition, the lack of available labeled datasets for the rail domain has limited progress in the field. In this study, we explored and evaluated machine learning algorithms for real-time railroad inspection systems from the ego-perspective of the locomotive. This was accomplished through a study on state-of-the-art semantic segmentation models on popular automotive datasets with semantic segmentation annotations. Second, transfer learning was performed on the models with a public rail dataset. Third, benchmarking was done on the newly trained rail models on an embedded system and PC. Finally, a custom dataset was created to highlight anomalies on rails (e.g., mud pumping and vegetation).

17. Key Words machine learning algorithms, real-time railroad inspection systems, semantic segmentation models, transfer learning, benchmarking	18. Distribution Statement No restrictions. This document is available to the public through the National Technical Information Service, Springfield, VA 22161. http://www.ntis.gov		
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 38	22. Price

Form DOT F 1700.7 (8-72) **Reproduction of completed page authorized**.

USDOT Tier 1
University Transportation Center
on Improving Rail Transportation
Infrastructure Sustainability and Durability

Final Report UNLV-8

REAL–TIME SEMANTIC SEGMENTATION FOR RAILWAY ANOMALIES ANALYSIS

By

Paul Stanik III
Graduate Research Assistant
Department of Computer Science
University of Nevada Las Vegas

and

Brendan Morris, Ph.D., Associate Professor
Department of Electrical and Computer Engineering
University of Nevada, Las Vegas
4505 S Maryland Pkwy Box 454026,
Las Vegas, NV 89154-4026
brendan.morris@unlv.edu

December 2022

Grant Number: 69A3551747132

DISCLAIMER

ABSTRACT

In the past few years, computer vision has made huge jumps due to deep learning which leverages increased computational power and access to data. The computer vision community has also embraced transparency to accelerate research progress by sharing open datasets and open source code. Access to large scale datasets and bench- mark challenges propelled and opened the field. The autonomous vehicle community is a prime example.

While there has been significant growth in the automotive vision community, not much has been done in the rail domain. Traditional rail inspection methods require special trains that are run during down time, have sensitive sensing/imaging analysis equipment with high costs, or may require low speeds for analysis. In addition, the lack of available labeled datasets for the rail domain has limited progress in the field. In this study, we explored and evaluated machine learning algorithms for real- time railroad inspection systems from the ego-perspective of the locomotive. This was accomplished through a study on state-of-the-art semantic segmentation models on popular automotive datasets with semantic segmentation annotations. Second, transfer learning was performed on the models with a public rail dataset. Third, benchmarking was done on the newly trained rail models on an embedded system and PC. Finally, a custom dataset was created to highlight anomalies on rails (e.g., mud pumping and vegetation).

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1. INTRODUCTION

1.1 Motivation

While there has been an increase of deep learning techniques for computer vision tasks in the automotive domain, there is a lack of research in the rail domain despite the long history of camera-based systems for maintenance (Nakhaee et al, 2019). This lack of research affects the safety of autonomous cars since autonomous cars share the same environment with rail intersections and trams. Traditional rail inspection methods require specialized locomotives that are run during downtime which are either equipped with expensive sensor equipment, run at low speeds for analysis, or use high resolution cameras. While the rail community has started to embrace deep learning for rail inspections in the past few years, there is still a lack of available labeled datasets and public benchmarks since most data is small or proprietary [Nakhaee et al, 2019].

1.2 Overall Objective

The objective of this project is to contribute on the implementation of identifying and localizing anomalies on railroad tracks that affect the tracks' health and condition using a low-cost camera-based railway anomalies analysis system using a single RGB, forward facing camera on a locomotive using real-time semantic segmentation. The proposed system differs from rail inspection and previous works: (1) anomalies differ from defects as anomalies can lead to rail defects while defects can lead to accidents on the rails; (2) we want a system that flags anomalies at their location, where multiple flags from multiple locomotives indicates that the area should be checked out; (3) instead of using a high resolution view, we want to use a more low resolution view available on standard RGB cameras; and (4) we want the process to be faster than rail inspection methods

1.3 Outline

Chapter 2 will focus on past research regarding rail systems inspections and back- ground in a selected GAN model and semantic segmentation. Chapter 3 will focus on the selected semantic segmentation algorithms. Chapter 4 focuses on our experiments and results using a public rail dataset and our custom dataset. Chapter 7 concludes the study, highlighting the advantages and shortcomings of our approach and the future work.

CHAPTER 2. BACKGROUND

2.1 Previous and Related Works

In the past few years, the rail community has started to embrace deep learning to detect both the support system of railways and irregularities in geometric measurements (i.e., rail defects) (Nakhaee et al, 2019). There have been specialized sensing setups that are designed for high-resolution view of rail elements. To improve the visual based track inspection (VTIS) for rail surface detection, Tracknet (James et al. 2018), a neural network architecture, was designed around using a high speed camera on the underside of a test train. They reduced false alarms by using a multiphase approach based on segmentation and cropped classification. A multi-task learning approach (Gibert et al

2017) was used to inspect cross ties and rail fasteners using single-view line-scan cameras on a commercial VITAS. The weights were shared between a material classifier and multi-class fastener defect classifier. A line-scan camera was used for wide field of view and high resolution for overhead rigid lines in subway tunnels (Jiang et al. 2019). Outside of the previous specialized, high-performance sensing systems, other works have looked into leveraging more standard RGB cameras and the work from the automotive field. For example, transfer learning to the target rail task was used to track defects like sun kinks (buckle on the railroad due to extreme hot weather) and loose ballast, and railways assets of switches and signals were monitored (Mittal and Rao2017). Further examples can be found in the 2019 survey (Nakhaee et al. 2019).

As noted in the 2019 survey (Nakhaee et al. 2019), the lack of available datasets and public benchmarks has limited advances in the field since most data is small and proprietary. This shortcoming was addressed through simulation of training images for sun kinks and vegetation overgrowth (Ritika and Rao 2018). Simulation was required since railway track anomalies are rare, making it difficult to train a generalized detector with only real images.

In 2019, the RailSem19 (Zendel et al. 2019) dataset was made publicly available for semantic scene understanding for trains and trams. This new dataset provides sufficient public data for more widespread development of deep learning models and advancements in the field. The dataset is made up of 8,500 annotated images from the ego-perspective of locomotives, including over 1,000 examples with railway crossings and 1,200 tram scenes with rail-specific labels.

2.1.2 Parts of Railroads

For our approach in analyzing railroads, there are four main components of railroads that will be discussed. Figure 2.1 will be used as an illustration for the different parts. **(a)** The rail is typically made of steel and is the main part of the track. They are made to withstand the forces of the locomotive and disperse the forces to the ties and ballast. **(b)** The ties support the railway tracks and maintains the position of the rail. It transmits the pressure the rail experiences from locomotives to the track bed. Railroad ties are typically made of wood or prestressed concrete in order to have some flexibility and elasticity. **(c)** The ballast is the typical track bed used under most railroad tracks. It is made of crushed stones that transfer the pressure of the ties to the subgrade beneath, fixes the position of the ties to maintain the correct position of the track, promotes water drainage, and increases the elasticity of the track to allow the rail to return to its original position after a locomotive passes. **(d)** The subgrade is the composed of native materials (e.g., soil) that is beneath the ballast (Agico Group 2020).

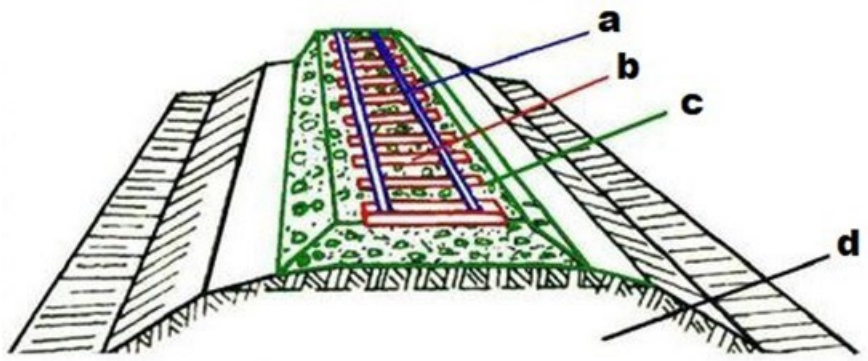


Figure 2.1: Parts of Rail Track (Agico Group 2020). a = rail, b = tie, c = ballast, and d = subgrade.

2.1.3 Railroad Anomalies

There are anomalies that affects the condition and health of railroad tracks which require maintenance to find and make repairs. These include tie wear, surface ballast fouling, vegetation growth, poor drainage, and mud pumping. In our work, we mainly focused on three anomalies as illustrated in Figure 2.2: **(a)** vegetation growth, **(b)** poor drainage, and **(c)** mud pumping.

Poor drainage can cause standing water in the rail tracks, which can cause wooden ties to degrade through rot. Vegetation growth is indicative of weed killer not working properly and a possible sign of poor drainage. Mud pumping is a type of surface ballast fouling that occurs when wet beds appear in clay-like subgrade, causing a clay slurry to push out when locomotives are passing through by pushing the ballast down. This surface fouling inhibits drainage and prevents the ballast to correctly absorb the pressure from the ties due to the space created (Hudson et al. 2016) Maintenance measurements to fix poor drainage and mud pumping can be costly as the ballast and ties need to be repaired or replaced.

(a) Vegetation growth (b) Standing water (c) Mud pumping

Figure 2.2: Rail Track Anomalies

2.2 Data Augmentation

Deep learning models, like deep convolution neural networks (CNNs) have been successful for computer vision tasks. One of the biggest weakness in using deep CNNs is the reliance on large sources of data for training to avoid overfitting -- which is when the deep CNN learns a function with high variance, causing it to perfectly model the training data. This causes the network to not reliably predict data outside of the training data. To overcome this problem in deep learning, the standard solution is

to use data augmentation. Data augmentation provides several ways to increase the size and quality of the training datasets by making copies and slightly modifying those copies (e.g., flipping images on the x-axis, using kernel filters, rotating the image) (Shorten and Khoshgoftaar 2019). See Figure 2.3 for examples of data augmentation.

Figure 2.3: Examples of Data Augmentation for Images (Agico Group Gond20)

2.3 Semantic Segmentation

Semantic segmentation, or image segmentation, is a difficult computer vision task that creates regions of labels in an image depending on the different pixel proper- ties. In the past few years, the use of deep learning approaches has been effective in this task by taking advantage of deep CNNs (Lateef and Ruichek 2019). Although similar to object detection, segmentation is important in understanding and analyzing images for specific tasks (e.g., autonomous driving and medical image analysis) (Lateef and Ruichek 2019). There are three different types of segmentation techniques: semantic segmentation, instance segmentation, and panoptic segmentation. Semantic segmentation labels every pixel with an associated class label. Instance segmentation creates segmentation masks of each instance of an object independently like object detection. Panoptic segmentation combines semantic segmentation and instance segmentation by labeling both the classes and instances of each class (Kirillov et al. 2018). The work in this study focuses on semantic segmentation. See Figure 2.4 for an example of each.

2.4 Real-Time Semantic Segmentation

Real-time semantic segmentation is the same task as semantic segmentation, but approached differently by making semantic segmentation more computationally efficient. Current approaches in semantic segmentation are focused more on accuracy rather than time efficiency. Typically, semantic segmentation performs well but not fast enough for time sensitive tasks (e.g., autonomous driving). Some solutions to make semantic segmentation models be more real-time are to either perform convolution computations more efficiently or to apply network compression to reduce the size of the network. Despite real-time semantic segmentation models performing faster than semantic

segmentation, they still lack the higher accuracy needed for their tasks (Lateef and Ruichek 2019).

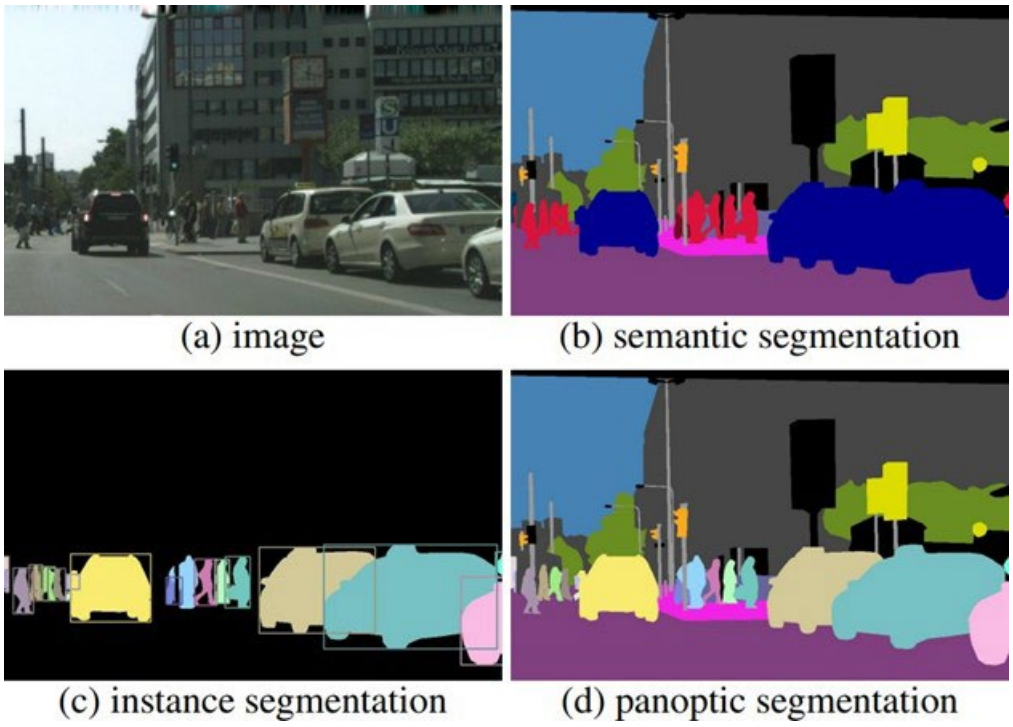


Figure 2.4: Different Types of Segmentation Tasks (Kirillov et al. 2018)

2.5 Generative Adversarial Networks

Machine learning algorithms can be split into two groups: supervised learning and unsupervised learning. Supervised learning is more popular and uses labeled data compared to unsupervised learning using unlabelled data. Supervised learning algorithms learn to map their inputs to their respected outputs, and unsupervised learning algorithms try to learn something meaningful by examining the labeled data (e.g., clustering and dimension reduction). Generative modeling is one unsupervised learning approach. Generative modeling tries to learn a model $p_{model}(x)$ that closely approximates $p_{data}(x)$ while using examples x drawn from the unknown distribution source of data $p_{data}(x)$. Generative adversarial networks (GANs) are a type of generative modeling that are based on game-theory between two machine learning model that try to mimic real data distribution.

The first player of the game is the generator which is defined by a prior distribution $p(z)$ over a vector z, where z is a parameter for the generator function $G(z; \theta^{(G)})$ and $\theta^{(G)}$ serves as the parameters that defines the strategy in the game. $p(z)$ is typically an unstructured distribution (e.g., Gaussian), and samples z can be viewed as a source of unstructured noise. The goal of the generator is to learn the function $G(z)$ that converts z to simulate samples from the training data (i.e., create realistic data based on the training data) (Goodfellow et al. 2020).

The second player is the discriminator which examines samples x and creates an estimate $D(x; \theta^{(D)})$ if x was sourced from the training data or created by the generator. Like a classifier, the discriminator determines the probability that x is real or fake. Basically the goal for the discriminator is to

distinguish real data from the training dataset and fake data from the generator (Goodfellow et al. 2020). Figure 2.5 represents the process.

Application-oriented research on GANs found that they are good in data generation (e.g., images) (Cai et al. 2021). In our case with the limited amount of available data on rail systems with anomalies, we conducted some experiments to generate meaningful data to label and train with for semantic segmentation. Since we used StyleGAN2-ADA, explanations of StyleGAN and StyleGAN2 will be discussed below.

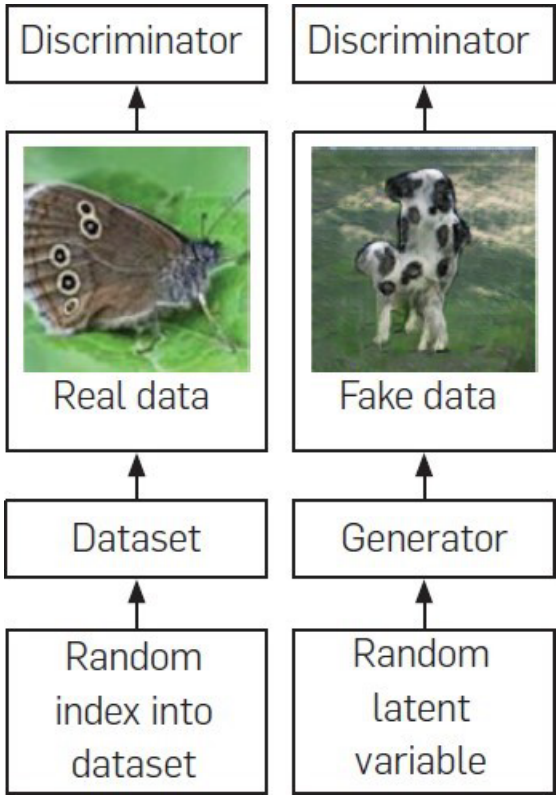


Figure 2.5: Depiction of the Process in GANs (Goodfellow et al. 2020)

2.6 Selected GAN Model

We selected StyleGAN2-ADA for the GAN model to synthetically create a rail anomalies dataset. We selected StyleGAN2-ADA because it provides two functions to aid in generating images: *style mixing* and *image projection*. The following subsections will cover what these functions are in the previous variants of the StyleGAN2- ADA model.

2.6.1 StyleGAN

Before StyleGAN, the generators in GANs operated like black boxes with little under- standing how the image synthesis process worked. While these models can generate realistic, large-resolution images, they leave out the ability to regulate their output (e.g., pose, hairstyles). StyleGAN is a GAN architecture that had the generator redesigned so that the image synthesis processes can be

controlled (Karras et al. 2021).

While the input for the generators in traditional GANs is a point from latent space, StyleGAN instead employs a nonlinear mapping network to map the latent code in the latent space to produce style vectors. The style vectors are used in additional layers in the network called adaptive instance normalization (AdaIN) operations which standardizes the output of each convolutional layer in the synthesis network. In addition, StyleGAN also provides the generator the ability to create stochastic details with the introduction of noise inputs. The noise inputs are fed into the network and added to the output of their corresponding convolutional layer. The style vectors provide control over the style of the output in the generator (Karras et al. 2021). See Figure 2.6 for an illustration of the architectural difference of the generators between traditional GANs and StyleGAN. Figure 2.7 shows some images of people who do not exist generated by StyleGAN trained on the Flickr-Faces-HQ dataset.

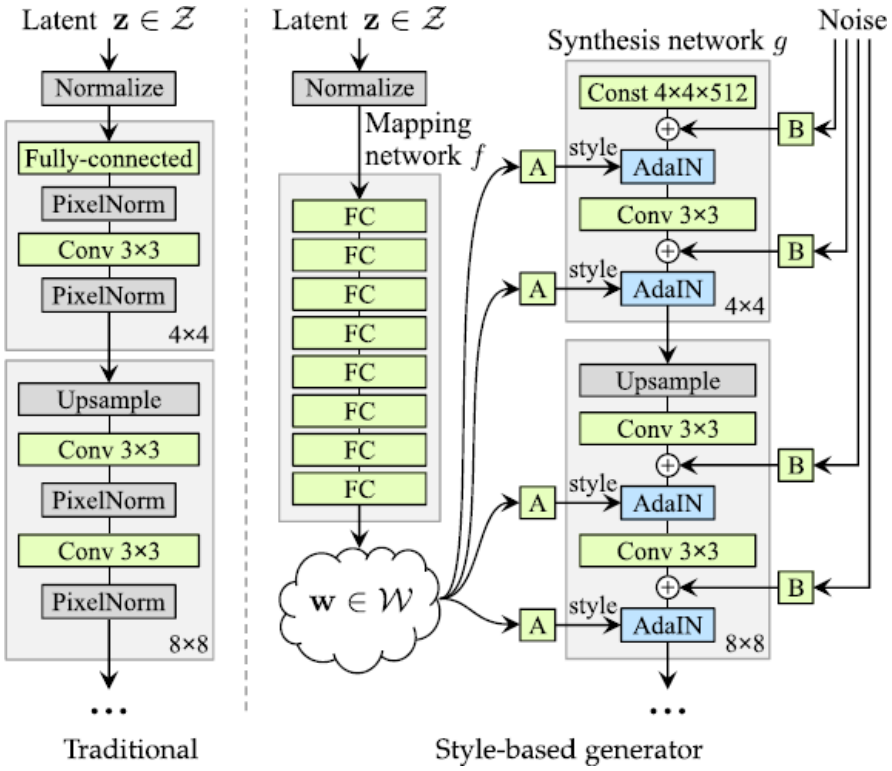


Figure 2.6: Architectural Differences of Generators Between Traditional GANs and Style- GAN (Karras et al. 2021)

StyleGAN introduced a method to encourage localization of styles in the generated images. This is done by mixing regularization, which is when the generator uses two random latent codes to generate images during training instead of one latent code. Style mixing is done in the generation of an image by switching from two latent codes at a random point in the synthesis network (Karras et al. 2021). Figure 2.8 shows some examples of style mixing. Note that the source images in the top row and left column are generated by StyleGAN.

Figure 2.7: Generated images from StyleGAN. These people do not exist (Karras el al. 2021).

2.6.2 StyleGAN2

StyleGAN was redesigned as StyleGAN2 to fix the design flaws that caused artifacts to be present in every generated image. StyleGAN2 also provides functionality to project images to latent space, which is a method that embeds an image into the GAN's latent space. This is useful for applications like image morphing and style transferring (Abdal et al. 2019). See Figure 2.9 for examples of transferring the content of given images (first row) to the images' styles in the first column.

2.6.3 StyleGAN2-ADA

Training GANs with a small dataset usually causes the discriminator to overfit. To overcome this, StyleGAN2-ADA introduces adaptive discriminator augmentation (ADA) to stabilize training with a small dataset. With a given probability, both fake and real images can be augmented before they go into the discriminator net- work, causing the discriminator to not only be exposed to clean images from both the generator and training dataset (Karras et al. 2020).

Figure 2.8: Example of Style Mixing in StyleGAN (Karras et al. 2021)

2.7 Evaluation Metrics

The main methods in evaluating the semantic segmentation models will be the intersection over union (IoU) and mean intersection over union (mIoU). These evaluation methods were selected as they are the standard in public benchmarks (Benchmark Suite, 2022, Papers with Code 2022a). The IoU is calculated by the area of the intersection between the ground truth and prediction and dividing it by the union of the ground truth and prediction, as illustrated in Figure 2.10. When there are two or more classes in a semantic segmentation model, the IoU for each class is taken into account by finding the mean intersection over union (mIoU). In measuring the speed of semantic segmentation models on different hardware configurations, only the speed of the forward pass during inferencing will be taken into account. We will measure it as the frames per second (FPS).

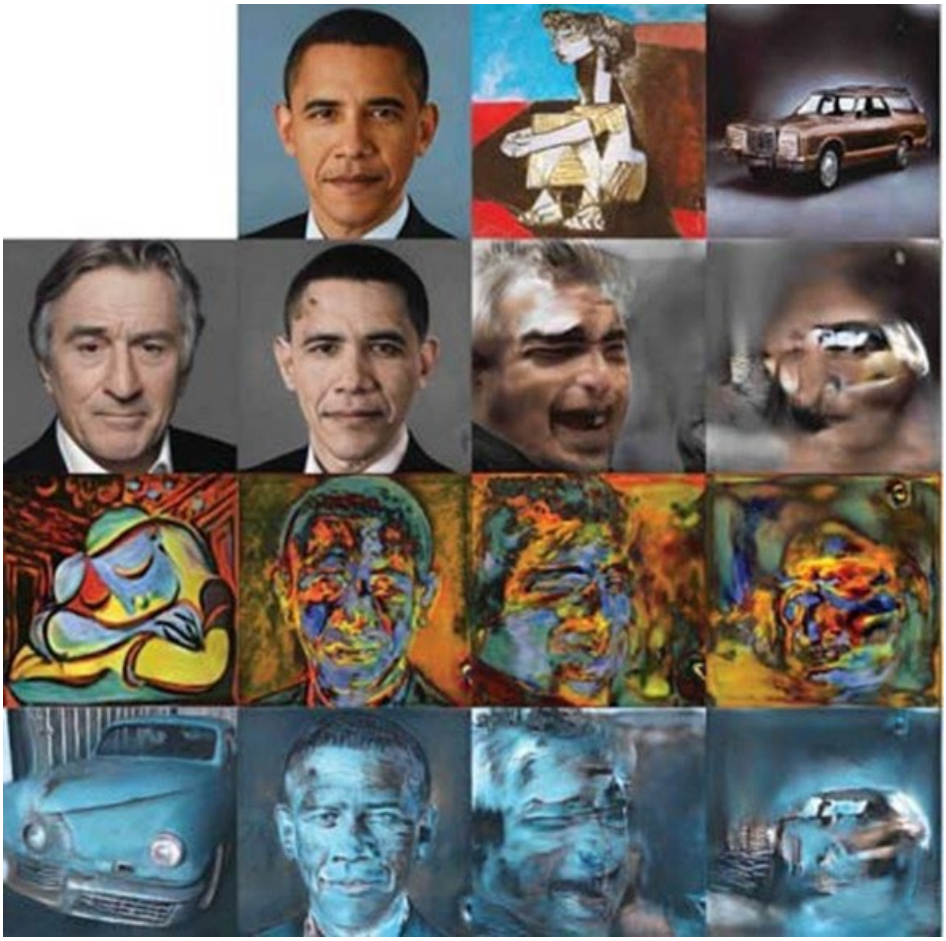


Figure 2.9: Examples of Style Transferring Through Image Projection to GAN Latent Space. The first row of images transfer their content to the style of images in the first column (Abdal et al. 2019).

For evaluating GANs, the main method for evaluation will be the Fréchet Inception Distance (FID) (Heusel et al. 2017). FID compares the distance of the calculated feature vectors between the

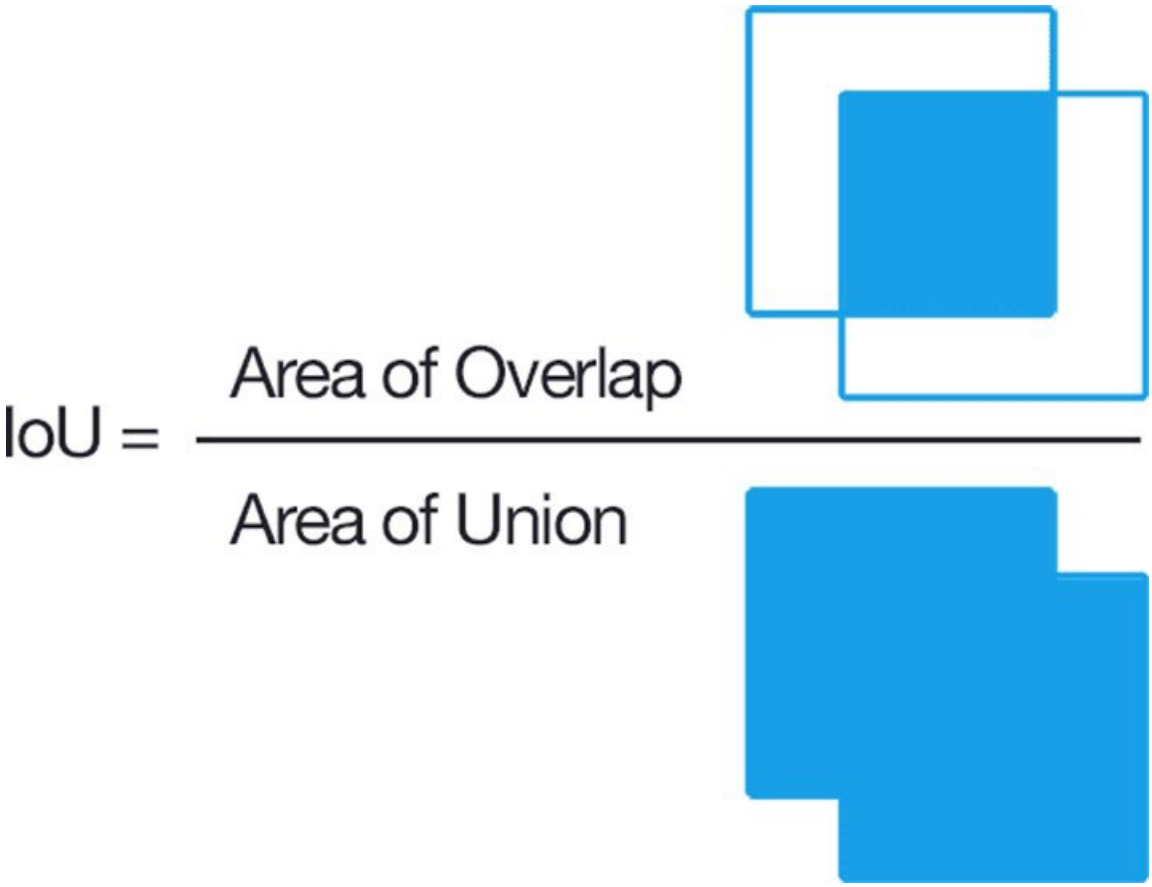


Figure 2.10: Illustration of the Intersection Over Union Equation (Rosebrock 2022)

vectors between the real and fake images of a GAN model. A lower FID score indicates the images from the real set and generated set are more similar than compared to a higher FID score. For GANs, there has been a correlation between higher quality images generated by the model and a lower FID score (Brownlee 2019). The FID can be calculated with the following equation:

$$FID(r,f) = \|\mu_r - \mu_f\|^2 + Tr(C_r + C_f - 2 * \sqrt{C_r * C_f})$$

In the equation above, r represents the real images and f represent the fake (i.e., gen- erated) images. μ_r and μ_f represents the feature-wise mean of the real and generated images. C_r and C_f are the covariance matrices for feature vectors of the real and fake images. Tr is the trace of the matrix (Brown 2019).

CHAPTER 3 SEMANTIC SEGMENTATION ALGORITHMS

In this chapter two different semantic segmentation models will be discussed. The state-of-the-art models were chosen based off of their rankings in the tasks of *Semantic Segmentation on Cityscapes test* and *Real-Time Semantic Segmentation on Cityscapes test* on Papers with Code (2022a] and Cityscapes' Benchmark Suite (2022] for the test set. We also took into account the difficulty in trying to train a custom dataset on the selected models (i.e., if they were open source), and if the accessible machines

had the available drivers to use the same software. The AI Server at the ECE Department at UNLV was used extensively for training and inferencing models, but it currently only supports up to CUDA 11.0 for the GPUs. This leaves out the possibility of using higher-performing models due to their required software versions.

We looked for models that performed well on the Cityscapes dataset since it was the dataset used in the RailSem19's experiment to see if their dataset is feasible [ZMM+19], and that it is large and provides a wide range of potential algorithms. The selected models were HRNet + OCR + Multi-Scale Attention (ranked 11th on Cityscapes' Benchmark Suite) and SFNet(ResNet-18) (ranked 2nd on Real- Time Semantic-Segmentation on Cityscapes on Papers with Code). Even though HR- Net + OCR + Multi-Scale Attention ranked 11th, it was the highest ranked model with available, compatible code at the time. The sections below will explain how they work.

3.1 HRNet + OCR + Multi-Scale Attention

The HRNet + OCR + Multi-Scale Attention model is a semantic segmentation algorithm that created state-of-the-art results in Cityscapes test (85.1 mIoU) at the time of their publication. The model's overview, architecture, and steps to repeat their state-of-the-art results will be discussed below.

3.1.1 Overview

The model uses multi-scale inference to address the issue that certain types of pre- dictions are better in higher or lower resolutions (e.g., predictions of fine details are better when an image is scaled up and predictions of large structures are better when the image is scaled down to make use of the global context). Figure 3.1 demonstrates examples for both cases. For input image **(a)**, the thin posts are inconsistently segmented in prediction **(b)** in a scaled down image, but the predictions are better when the image is scaled up in prediction **(c)**. For input image **(d)**, the road is segmented more consistently when the image is scaled down (Tao et al. 2020a).

(a) Input Image (b) Prediction at 0.5x Scale (c) Prediction at 2.0x Scale

(d) Input Image (e) Prediction at 0.5x Scale (f) Prediction at 2.0x Scale

Figure 3.1: Examples of Inconsistencies in Semantic Segmentation with Different Inference Scales (Tao et al. 2020a)

Multi-scale inference is done by making multiple predictions with different scales, and then the resulting predictions are combined using averaging or max-pooling. However there are caveats in using these two methods. Averaging the different scaled results combines both the best and poor conditions together. Max-pooling only selects one of the scales for a given pixel, ignoring optimal predictions using a weighted combination across all the scales (Tao et al. 2020a)

A solution that the model provides is using a hierarchical attention mechanism to combine the multi-scale predictions (Tao et al. 2020a). Attention approaches in CNNs allow models to focus on the more important information of the given data. For computer vision, attention allows the model to "pay attention" to certain details in images that need to be localized and classified.

The hierarchical attention mechanism causes the model to learn an attention mask based off of the pair of scales during training. This causes the model to predict the relative attention with a set of scales. As seen in Figure 3.2, the model hierarchically applies the learned attention to combine the set of prediction scales during inferencing. Precedence is given to the lower scales and it works its way up with higher scales, causing lower scale predictions to have more global context from the higher scale predictions. Regarding the hierarchical attention mechanism, the network is able to be flexible with their scales during inference (e.g., inferencing with scales 0.5x or 1.5x with a model trained with 1.0x and 2.0x). This is an improvement over previously proposed methods as they are limited to only use the same scales the model was trained on (Tao et al. 2020a).

3.1.2 Architecture

The model's architecture is broken down into the following parts: backbone, semantic head, attention head, and auxiliary semantic head. The backbone used for their best results was HRNet-OCR [YCC+19]. The semantic head performs the semantic predictions and is a dedicated fully convolutional head. The attention head is separate and performs the attention predictions and is structurally identical to the semantic head. The outputs of the OCR [TSC20a] block are fed into the semantic head and attention head, and the HRNet trunk's output is fed into the auxiliary semantic head before the OCR block (Yuan et al. 2019).

3.1.3 Experimental Results

To achieve the 85.1 mIoU on Cityscapes test, the HRNet part of the model was initialized with weights trained on ImageNet (Russakovsky et al. 2015) classification. The entire model is then trained on the Mapillary dataset and then fine-tuned on the Cityscapes dataset.

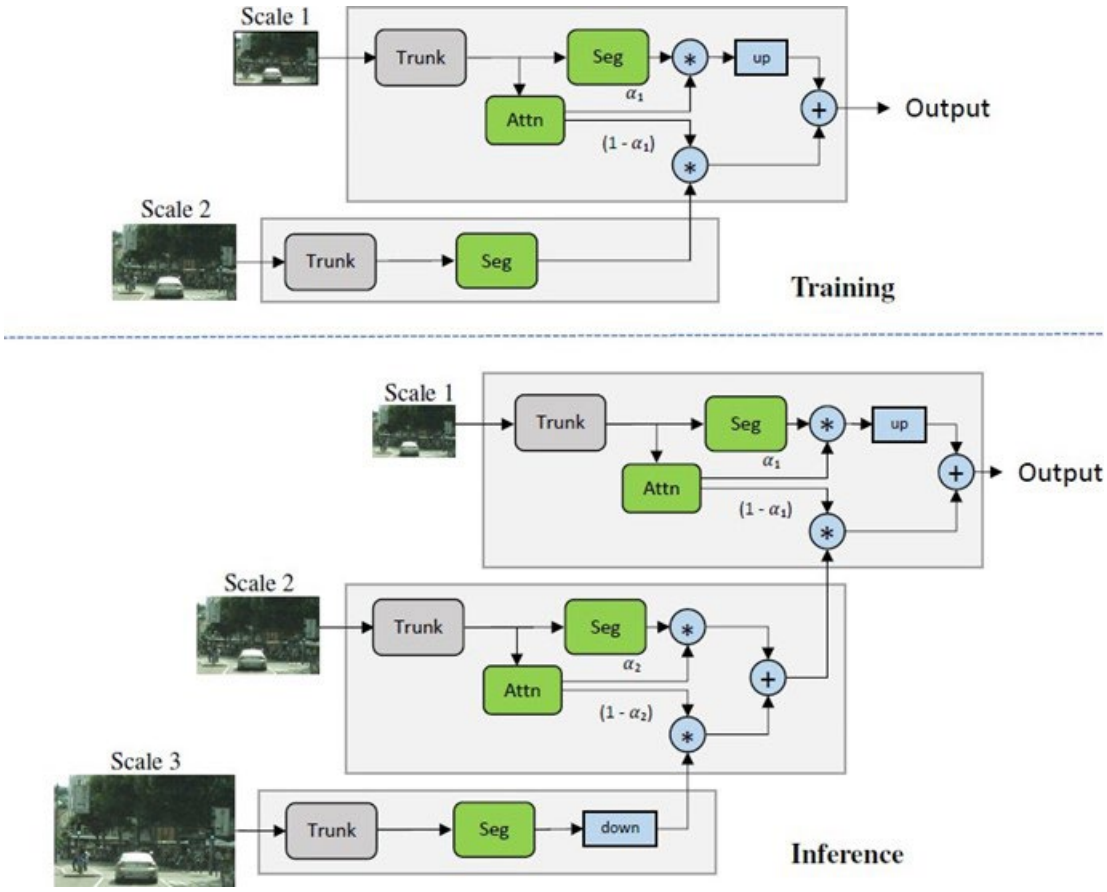


Figure 3.2: Architecture of HRNet + OCR + Multi-Scale Attention (Tao et al. 2020a)

3.2 SFNet(ResNet-18)

The SFNet(ResNet-18) model is a real-time semantic segmentation algorithm that achieved state-of-the-art results on Cityscapes test (80.4 mIoU) with 18 FPS. The model's overview, architecture, and steps to repeat their state-of-the-art results will be discussed below.

3.2.1 Overview

Semantic segmentation performance is heavily influenced by detailed information and strong semantic representation. The use of fully CNNs can build strong semantic representation. However, the down-sampling layers can cause the loss of detailed object boundary information (Li et al. 2020a). Some popular solutions are the use of dilated convolutions (Yu and Koltun 2016) in the last stages in the networks or to build models that are like a feature pyramid network (FPN). While dilated convolutions generate feature maps with strong semantic representation and maintaining high resolution, they are computationally more expensive than the last stages of fully convolution networks. FPN-like models are able to extract features with strong semantic representation and maintain detailed resolution information by fusing feature maps from top to down by using the lateral path. This causes the strong features in the last layers to strengthen the weaker features with high resolution. However, FPN-like models still pale in comparison to models that have large feature

maps in the last stages of the network. Also, FPNs rely on upsampling the smaller feature maps, but the bilinear upsampling method only works on one type of fixed, predefined misalignment and the repeated downsampling and upsampling operations in the residual connections cause misalignment in the feature maps (Li et al. 2020a).

SFNet (Li et al. 2020a) proposed the idea for the model to learn the semantic flow be- tween the layers of the network with different resolutions to solve the misalignment issue. This idea was inspired from optical flow in computer vision, which represents the motion of objects between frames caused by the movement of the camera or objects, in order to make the model be more flexible and dynamic in the alignments between the feature maps. Here optical flow can be represented as the motion of the pixels from consecutive feature maps. The flow alignment module (FAM) is designed around semantic flow, causing the feature maps following FAM to be enriched with information that improves the accuracy and keeps the efficiency since the transmission of this information from distant layers are done with simple operation. Since FAM is end-to-end trainable, it can use any backbone network with minor computational overhead. Networks that use FAM with a specific backbone are denoted as SFNet(backbone) [LYZ+20a].

3.2.2 Architecture

The architecture of SFNet is illustrated in Figure 3.3. The pyramid pooling module (PPM) (Zhao et al. 2017) is used to capture contextual information. The PPM and last residual module share the same resolution and both are treated as the last stage for input for the FPN decoder. The FPN decoder parses the final scene by using the feature maps from the encoder and aligned feature pyramid. The upsampling operations are replaced with FAM due to misalignments during the previous step.

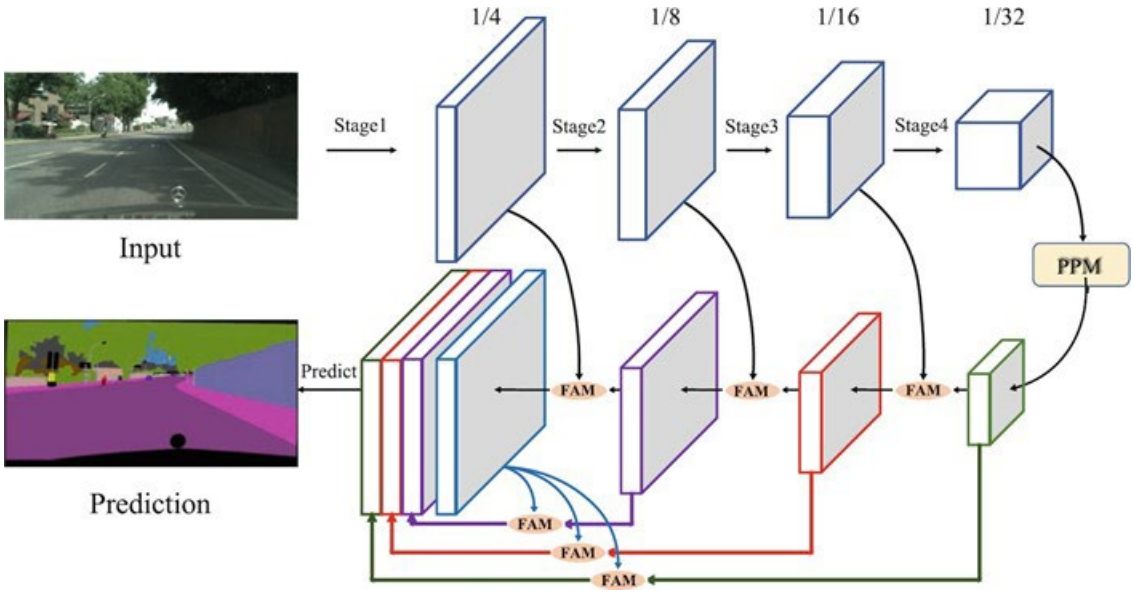


Figure 3.3: Architecture of SFNet (Li et al. 2020a)

3.2.3 Experimental Results

The SFNet(ResNet-18) model was able to achieve 80.4 mIoU on Cityscapes test with a higher FPS compared to state-of-the-art models of the time of their publication. This was done by using the backbone ResNet-18, pretrained on ImageNet classification. Next, the model was pretrained on Mapillary and then fine-tuned on Cityscapes. The 18 FPS was achieved with input size 1024 x 2048 by only measuring the forward pass using a single GTX 1080 Ti GPU.

CHAPTER 4 EXPERIMENTS AND RESULTS

4.1 Single-Board Implementation

Traditional rail inspection methods use specialized locomotives that run during down time, use specialized expensive equipment, or require low speed for analysis. These or other inspection methods can use high resolution views with specialized cam- eras and can be time consuming. An approach for detecting and localizing anomalies can be done by using a single-board implementation using a trained real-time semantic segmentation model using a more standard RGB camera with a lower resolution. This approach has an advantage of taking little space on a locomotive and that the anomaly detection can be done real-time while the train is running at its standard speed. With the amount of available real estate that locomotives have compared to automobiles, it is also possible to use a PC with a power GPU.

For our approach we decided to use a single-board system for semantic segmentation. We chose NVIDIA's Jetson AGX Xavier which has enough resources to run the selected models.

4.2 Datasets

This section will go over the different datasets that were used for the semantic segmentation models. The datasets are split between automotive and railway specific. The automotive datasets were selected due to their use in public benchmarks and in the available railway specific dataset's paper [ZMM+19].

4.2.1 Automotive Datasets

The publicly available automotive datasets for semantic segmentation were primarily used to pretrain models before training on railway specific datasets. While automotive vehicles share the same environment as locomotives (e.g., rail intersections and trams using the same roads as cars), locomotives are underrepresented in existing automotive datasets (Zendel et al. 2019). The automotive datasets are used to pretrain the models because they share the similar environments, have similar classes, are large, and provide a wide range of potential algorithms.

Cityscapes

Cityscapes in Benchmark Suite (2022) contains 5,000 annotated images and 20,000 coarsely annotated images from the ego-perspective of an automobile. The dataset contains 35 class labels, but only two of them are specific to the rail domain: *rail track* and *train*. The dataset is provided

in three splits: training (2,974), validation (500), and test (1,525). The ground truth annotations for the test set are withheld for benchmarking purposes. Cityscapes also provides a benchmark suite where anyone can upload their inference results from the provided test set to see how well their models performed in Benchmark Suite (2022).

Mapillary Vistas

Mapillary Vistas [NOBK17] is a street-level image dataset containing 25,000 images and 66 classes. In regards to the rail domain, it only contains two classes that are also found in Cityscapes: *construction-flat-rail-track* and *object-vehicle-on-rails*. Not all the images are from the ego-perspective of automobiles. Mapillary Vistas was primarily used for pretraining the state-of-the-art semantic segmentation models before training on Cityscapes.

4.2.2 Railway Specific Datasets

With the lack of available public datasets in the rail domain there is only one known available, RailSem19 (Neuhold et al. 2019). However, the dataset does not focus on the anomalies that we are looking for. So we created our own dataset to highlight the rail anomalies. We used two approaches in creating our own dataset: synthetically generate it with GANs and scraping scenes from online sources.

RailSem19

RailSem19 is the first public semantic segmentation dataset in the rail domain. The annotations are a mix of manual annotations and dense labeling done using geometric shapes and weakly supervised annotations created by existing semantic segmentation networks from the road domain. There are 8,500 images from 38 countries in all four seasons and in different weather conditions (Zendel et al. 2019). RailSem19 used a lot of the same labels or combination of Cityscapes, and it ended up with the 19 classes.

In annotating the images the ties were not considered. Instead almost everything between the raised rails were labeled as *rail-track*. In addition, the anomalies we are interested in were not labeled (e.g., vegetation found in rail tracks were ignored). To test the feasibility of their model, the authors pretrained a FRRNB (Pohlen et al. 2017) model on Cityscapes and then fine-tuned it on RailSem19 using only 4,000 randomly selected images split into subsets 3,000 (training), 500 (validation, and 500 (test). They decided to use this partition of their dataset to replicate the size and split of the Cityscapes dataset. They ended up getting a mIoU of 62.7 on Cityscapes test and then mIoU of 57.6 on their RailSem19 test split (Zendel et al. 2019).

GAN Based

While searching for scenes of rails from the ego-perspective to create a custom dataset can be time consuming, we attempted to synthetically generate our own using a GAN based system. In particular, we decided to use StyleGAN2-ADA (Karras et al. 2020) due to the limited data we have (i.e., only RailSem19), the ability to generate high-quality large-resolution images, the ability to style mix the generated images (e.g., place anomalies on the rail track), and the ability to project a target image

to the latent space of a model. In particular, we used a PyTorch implementation of the StyleGAN2-ADA (Karras et al. 2020).

The StyleGAN2-ADA model was trained from scratch using the entire RailSem19 dataset on four Quadro RTX 6000 GPUs until the FID score flattened to **5.67**. While the trained model does generate images similar to those found in RailSem19, the model only generates some images with vegetative growth and not the other anomalies that we wanted as the dataset was not focused around rail anomalies. While trying to style mix the vegetative growth style to other styles with no vegetative growth, the resulting images were not satisfactory for us. In Figure 4.1, the generated images make up the top row and left column. The images in the left column, Source A, contain vegetation in the rails. The style of the vegetation in the rails does not transfer over while performing style mixing. We also tried to use the projection function to project images to the latent space. As seen in Figure 4.2, the target images are projected into the latent space of the StyleGan2-ADA model. We tried to simulate vegetative growth and standing water using stand-alone target images and images of railroad tracks being affected by both. We hoped to generate images with rail anomalies, but resulting projection images were not realistic enough, so we didn't want to do annotations on these images. So we used a more labors approach that will be discussed in the next section.

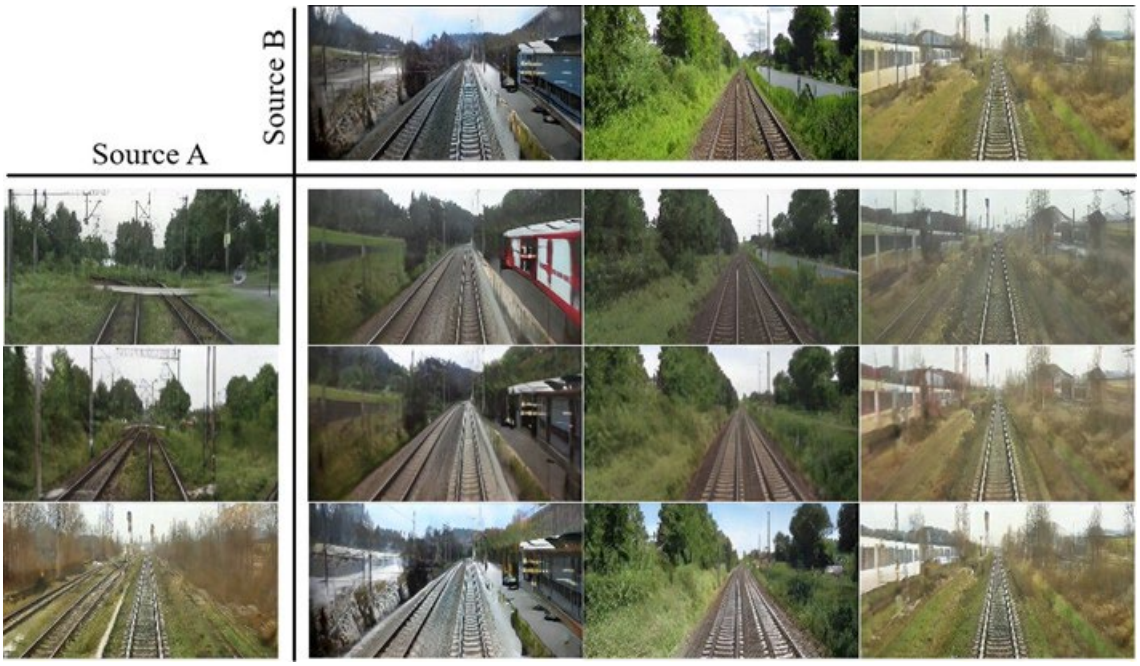
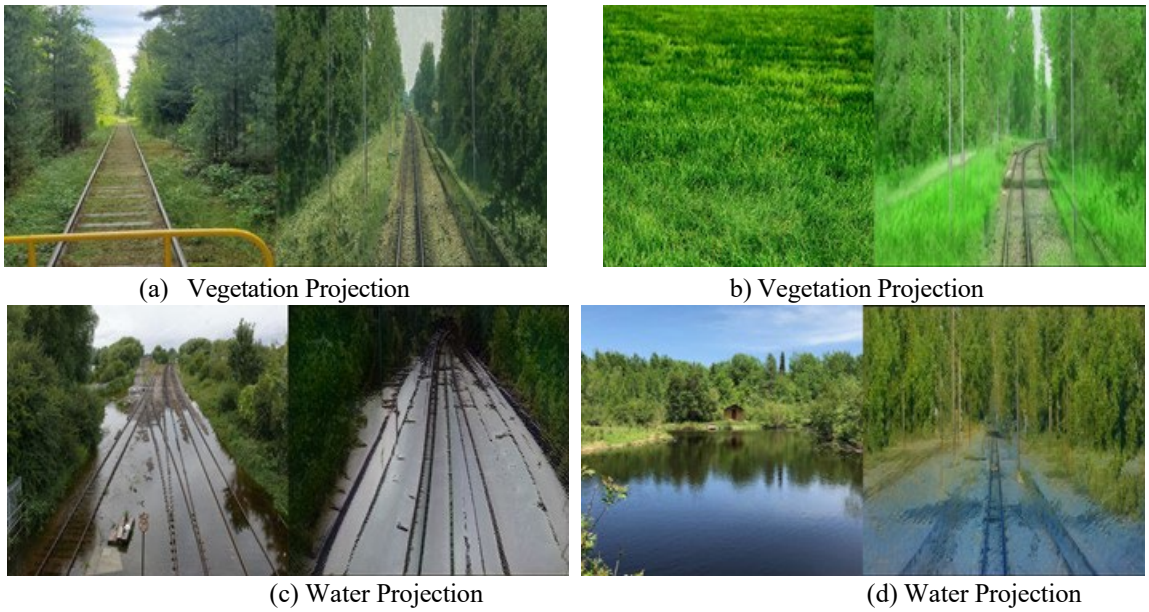


Figure 4.1: Style Mixing with StyleGan2-ADA Trained on RailSem19

Scraped Images

After trying to generate synthetic images with a GAN based approach didn't work as well as we wanted, we ended up doing the laborious task of scraping scenes from online images and videos. It was difficult to find high-quality large-resolution images of rail anomalies from the ego-perspective of locomotives. While there are many public videos available on the internet with an ego-perspective

of a locomotive, the locomotives generally were going too fast to capture the anomalies clearly or if there were any anomalies to begin with. So we had to find videos with locomotives going the right speed for us to capture each frame manually as we watch the videos. The videos that we did find were available on online, and they tend to be a few hours in length each.



(a) Vegetation Projection b) Vegetation Projection

(c) Water Projection (d) Water Projection

Figure 4.2: Resulting Image Projections. The target images (left) are the input images that are projected into the latent space of the trained StyleGan2-ADA model on RailSem19 where the projection images (right) are the resulting projections.

The images we were looking for were any high-quality image from the ego-perspective of the locomotive that contains the rail anomalies we were interested in (e.g., surface ballast fouling, vegetation, standing water). While we did find images for both surface ballast fouling and vegetation, we were only able to find a handful of images that contained standing water on the rail tracks. Also, we were only able to find a few videos containing mud-pumping. Because of the sparse available data of the anomalies from the ego-perspective, we ended up using videos and images from different points of view.

4.2.3 Dataset Labeling

Annotations were done using Supervisely (2022) using the labels in Table 4.1. The labels are similar to RailSem19, but we did modify the labels by combining *truck* and *car* as *automotive* due to the number of frames containing the truck class. We also replaced *rail-track* with *tie*, and added *mud-pumping*. A total of 152 images were given manual annotations. Depending on the point of view and how busy the image is, an image typically took thirty minutes to a four hours to fully annotate. See the Appendix for some examples of the annotated images. Like how RailSem19 (Karras et al. 2019) has 19 classes, we also wanted to keep the same number of classes. This custom dataset will be called the Rail Anomalies Dataset.

4.3 Experimental Setup

Before using our selected models, we wanted to first recreate RailSem19's results to get a baseline of RailSem19. In their paper [COR+16] they selected the FRRNB (Pohlen et al. 2017) from the same GitHub repository they used (Shah 2017). We will follow their approach in creating a baseline for RailSem19 by first training on Cityscapes until the validation loss flattens. Then, we will do two fine-tuning experiments on RailSem19: **(1)** using their partition of 4,000 images that they provided and **(2)** using the entire dataset. Both experiments were done without freezing any layers and were trained until the validation loss flattened. With the entire dataset, we created our own split of images for training (6,800), validation (850), and testing (850).

Table 4.1: Table of our Rail Anomalies Dataset Annotation Labels with the Color Legend. Pixel-wise statistics: **Objects Area** - percentage of pixels each label takes up in the entire dataset; **In Frames** - percentage of the frames containing the respected label; **IoU HRNet** - IoU results of the test set from the custom dataset using the HRNet + OCR + Multi- Scale Attention; **IoU SFNet** - IoU results of the test set of the custom dataset using SFNet(ResNet-18).

Color							
Label	tie	rail-raised	trackbed	vegetation	terrain	sky	mud-pumping
Objects Area	2.76%	6.84%	16.14%	27.55%	2.63%	11.06%	26.68%
In Frames	50.55%	100.00%	94.74%	78.29%	36.84%	34.21%	74.34%
IoU HRNet	52.58	96.65	79.06	91.73	38.41	99.42	87.49
IoU SFNet	60.00	95.25	78.83	91.99	38.81	99.12	88.78
Color							
Label	automotive	pole	person	on-rails	fence	traffic-sign	traffic-light
Objects Area	0.07%	0.75%	0.07%	0.45%	0.49%	0.03%	0.03%
In Frames	7.23%	30.26%	3.29%	5.92%	14.47%	10.53%	5.26%
IoU HRNet	50.78	83.32	97.76	86.95	69.57	90.19	79.51
IoU SFNet	47.61	79.26	92.24	86.01	18.27	45.74	65.20
Color							
Label	rail-embedded	road	sidewalk	construction	tram-track	void	
Objects Area	0.01%	0.52%	0.59%	2.29%	0.13%	0.26%	
In Frames	2.63%	13.16%	13.82%	31.58%	2.63%	11.18%	
IoU HRNet	0.00	57.69	34.64	79.53	0.00	-	
IoU SFNet	0.00	53.99	45.28	73.52	1.26	-	

Next, we wanted to see how well some state-of-the-art PyTorch based models per- form on RailSem19. As discussed in Chapter 3, we chose the semantic segmentation model HRNet + OCR + Multi-Scale Attention (Tao et al. 2020a) and real-time semantic seg- mentation model SFNet(ResNet-18) (Li et al. 2020a). We used the provided pretrained checkpoints of each model trained on Mapillary Vistas and fine-tuned on Cityscapes from their respected public repositories (Tao et al. 2020b, Li et al. 2020b). Transfer learning was done on the models with our Rail Anomalies Dataset using four Quadro RTX 6000 GPUs, and training was stopped when the validation loss flattened.

Since we are interested in running a model on a locomotive in real-time, we wanted to test the speed of the models on different hardware setups. The different hardware setups we used were a single Quadro RTX 6000 GPU, single Titan X GPU, and a Jetson Xavier. The speed for each model will be measured by measuring the FPS for only the forward pass, which is similar to what was done for the speed tests on SFNet(ResNet-18) (Li et al. 2020a,b).

4.4 Experimental Results

After following RailSem19's approach, we ended up with a mIoU **61.7** on Cityscapes test compared to their mIoU 62.7. After transfer learning on RailSem19's 4,000 image partition and the entire dataset, we ended up with mIoU **59.5** and **61.7**, respectively, compared to RailSem19's mIoU 57.6 on their partition.

After fine-tuning the models on our Rail Anomalies Dataset (which were pre- trained on Mapillary Vistas, trained on Cityscapes, and transfer learned to RailSem19), HRNet + OCR + Multi-Scale performed best in terms of predictions (mIoU **67.1**) but was much slower than SFNet(ResNet-18) (mIoU **61.0**) on RailSem19. Looking at the individual IoU of each class in Table 4.1, the classes *rail-embedded* and *tram-track* performed very poorly in the test set. We expect this to be the case since the entire dataset didn't contain enough examples of those classes (i.e., only a few images had these classes in it) and those classes covered a small area of the dataset. See Fig- ures for some examples of the output on the test set 4.3 and 4.4. In terms of speed, the Jetson Xavier was particularly slow with only **1.14** FPS with SFNet(ResNet-18) and **0.08** FPS with HRNet + OCR + Multi-Scale Attention. On the GPUs, the SFNet(ResNet-18) had a speed of **20.83** FPS on the Quadro RTX 6000 and **10.75** FPS on the Titan X, and the HRNet + OCR + Multi-Scale Attention had a speed of **0.86** FPS on the Quadro RTX 6000 and **0.33** FPS on the Titan X. The performance of the models used for the experiments can be seen in Tables 4.1, 4.3, and 4.2.

In regards of the other rail specific classes, both models performed similarly in detecting the anomalies *vegetation* and *mud-pumping* as seen in Table 4.1. Surprisingly, SFNet(ResNet-18) performed better in detecting the *tie* class. However, since the Rail Anomalies Dataset was not as big as we would want it to be, it needs more images to be more robust in detecting rail anomalies. The majority of the frames containing mud-pumping were sourced from a single video from the point-of-view of a person repairing the rail track damaged by mud-pumping. So there are only a few images containing mud-pumping from the ego-perspective of the locomotives. Because of this, there were a lot of false positives of mud-pumping when we ran inference on the RailSem19 test set on our trained models, as seen in Figures 4.5 and 4.6. A lot of the false positives occurred when inferencing was done on street scenes since the model misinterpreted the *road* and *tram-track* classes as *mud-pumping*. This was probably the case since our dataset didn't contain any street scenes containing *road* and *tram-track*. The second row fourth column of Figure 4.6 shows a mislabeled ground truth from RailSem19 most likely due to the weakly supervised labeling. The ground truth indicates that there is sky in the middle of the mountain in the background, but that entire mountain should have been labeled as *terrain* or *vegetation*. The predictions made by my models were able to predict the mountain and sky correctly.

Table 4.2: Inference time on selected models on RailSem19 in FPS. The speed only takes into account the forward pass of each model.

Model	Jetson Xavier	Quadro RTX 6000	Titan X
HRNet+OCR+Multi-Scale Attn	0.08	0.86	0.33
SFNet(ResNet-18)	1.14	20.83	10.75

Table 4.3: Semantic Segmentation Results. The top section represents the benchmarks provided for the selected models and the corresponding datasets. The bottom section represents our results. **HRNet** = HRNet + OCR + Multi-Scale Attention, **SFNet** = SFNet(ResNet-18) **RS19** = RailSem19, **City** = Cityscapes, **Map** = Mapillary Vistas, **Anomalies** = Rail Anomalies

Model	Dataset	Pretrained	Training	Test mIoU
FRRNB	City	-	Scratch	62.7
FRRNB	RS19 4k Split	City	Fine-Tune	57.6
HRNet	City	Map	Fine-Tune	85.4
SFNet	City	Map	Fine-Tune	80.4
FRRNB	City	-	Scratch	**61.7**
FRRNB	RS19 4k Split	City	Fine-Tune	**59.5**
FRRNB	RS19	City	Fine-Tune	**61.7**
HRNet	RS19	City + Map	Fine-Tune	**73.2**
SFNet	RS19	City + Map	Fine-Tune	**69.4**
HRNet	Anomalies	RS19 + City + Map	Fine-Tune	**67.1**
SFNet	Anomalies	RS19 + City + Map	Fine-Tune	**61.0**

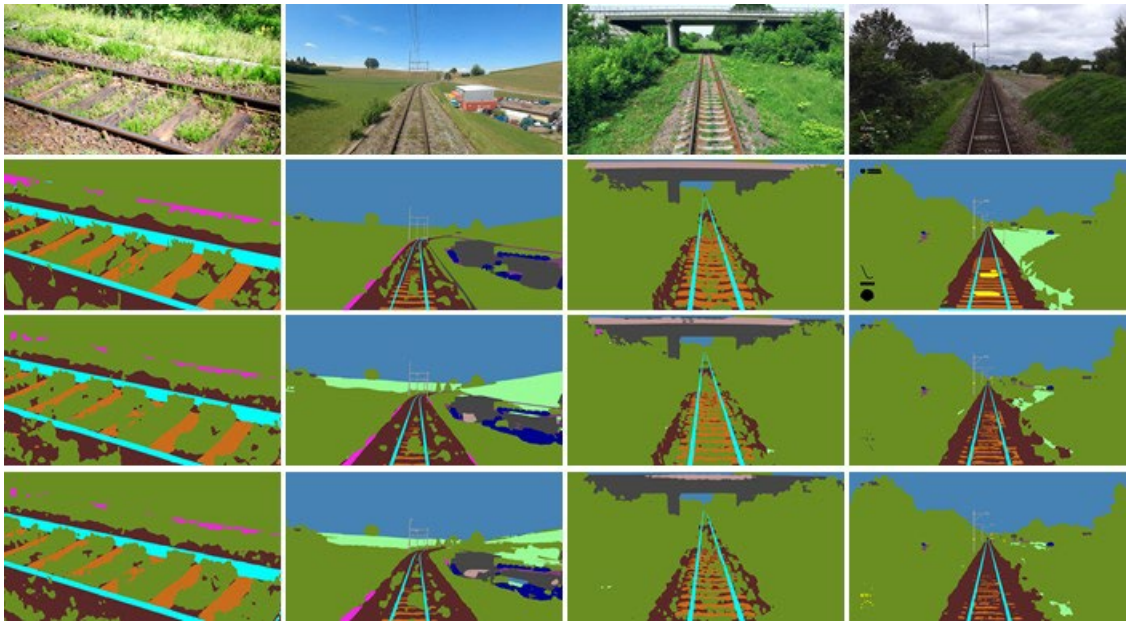


Figure 4.3: Examples of our experiment on the test set of the custom dataset on the selected models. First row = input images; second row = ground truth; third row = output from HRNet + OCR + Multi-Scale Attention; fourth row = output from SFNet(ResNet-18). For the color legend, see Table 4.1.

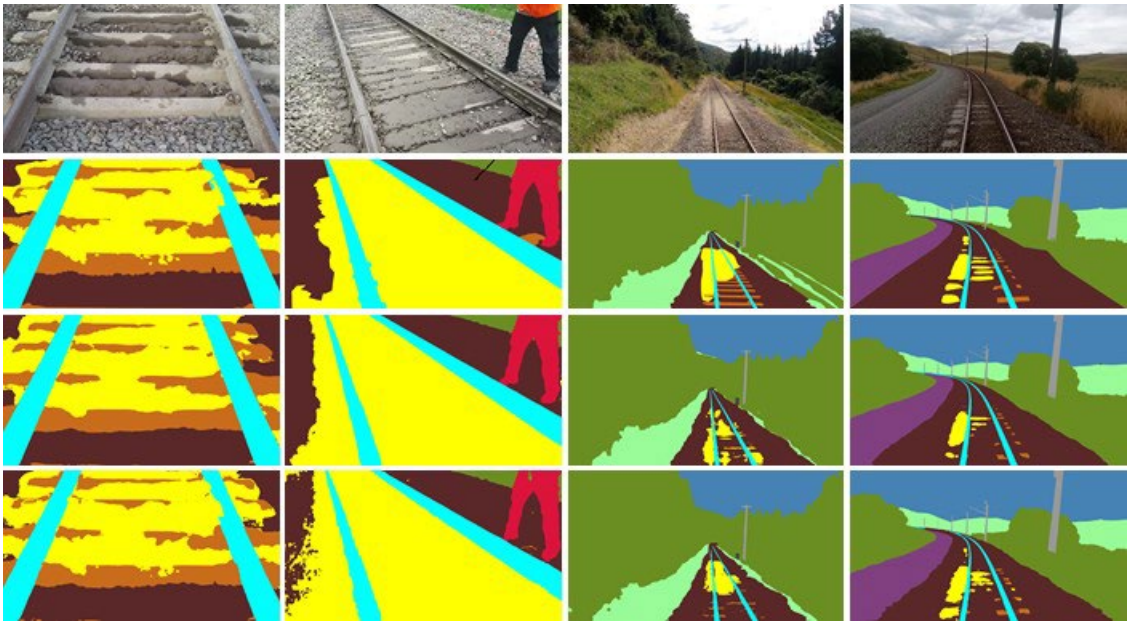


Figure 4.4: Examples of our experiment on the test set of the custom dataset on the selected models. First row = input images; second row = ground truth; third row = output from HRNet + OCR + Multi-Scale Attention; fourth row = output from SFNet(ResNet-18). For the color legend, see Table 4.1.

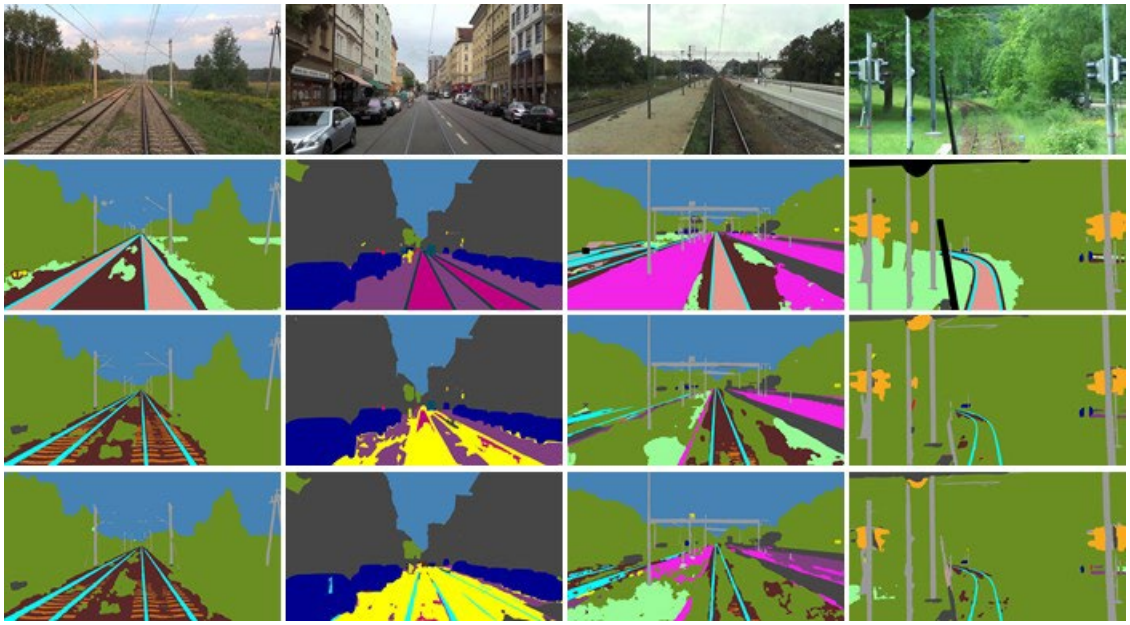


Figure 4.5: Examples of our experiment on the test set of RailSem19 on the selected models. First row = input images; second row = ground truth; third row = output from HRNet + OCR + Multi-Scale Attention; fourth row = output from SFNet(ResNet-18). For the color legend, see Table 4.1.

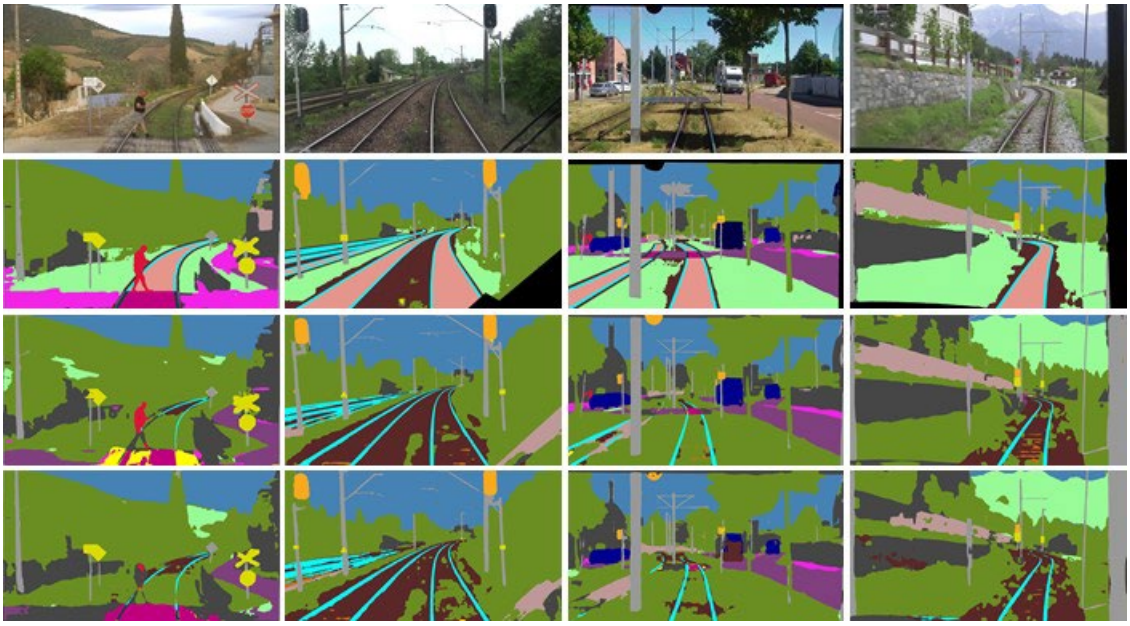


Figure 4.6: Examples of our experiment on the test set of RailSem19 on the selected models. First row = input images; second row = ground truth; third row = output from HRNet + OCR + Multi-Scale Attention; fourth row = output from SFNet(ResNet-18). For the color legend, see Table 4.1.

CHAPTER 5. CONCLUSION AND FUTURE WORKS

In this study, we demonstrated a proof-of-concept of analyzing rail tracks with a low- cost option using semantic segmentation. We looked into semantic segmentation and real-time semantic segmentation Pytorch based models based on how they performed on the Cityscapes dataset, and we chose one for each category which were HRNet + OCR + Multi-Scale Attention and SFNet(ResNet-18), respectively. We then used the provided checkpoints for the selected models to fine-tune on the RailSem19 dataset. HRNet + OCR + Multi-Scale attention ended up with mIoU 73.2 on the test set, outperforming SFNet(ResNet-18) with mIoU 69.4. However, SFNet(ResNet-18) has much higher FPS than the other model, which is imperative for real-time performance on a locomotive.

After seeing how well the RailSem19 dataset performed on the selected models, we needed to create our own dataset of rail anomalies. We first tried to generate synthetic images using StyleGan2-ADA, a GAN model, by training it on RailSem19 until the FID score flattened to 5.67. However, the StyleGAN2-ADA model wasn't able to create usable images containing anomalies when we tried to perform style-mixing and projection.

We used a different, more laboriously approach in created our own dataset. Without access to proprietary data, we had to scrape frames sourced from images and videos online that contain a clear view of the rail anomalies. Since the rail anomalies are rare, it was difficult finding good quality frames. With only 152 frames collected and annotated manually, we fine-tuned our models trained on RailSem19. HRNet + OCR + Multi-Scale attention ended up with a mIoU 73.2 on the test set, and SFNet(ResNet-18) ended up with mIoU 69.4. However, SFNet(ResNet-18) has much higher FPS than the other model. While the models are able to identify the anomalies, there were a lot of false positives when we ran the test set of RailSem19. With our results, we recommend using a PC setup using a powerful GPU instead of a single- board computer system due to the better computation power over the Jetson Xavier. With the lack of available data containing rail anomalies, we were not able to train a robust semantic segmentation model. We would like to have a sizeable dataset, but there wasn't much available.

Fore future work, it would be best if we are able to gain access to proprietary data to train a more robust model. As our proposed system uses a standard RGB camera, depending on the camera, the speed of the locomotive can be an issue. If the locomotive is going too fast for the camera, then the images will be too blurry for the rail anomalies detection model. Another issue with using an RGB camera is lighting can cause issues in anomaly detection. If the lighting is poor, like at night, it will be difficult to detect the anomalies. Also, as locomotives pass through dark tunnels, the operator might not turn on the lights on the locomotive. So anomalies won't be detected correctly or at all in tunnels. Finally, having access to a locomotive to run the proposed system was difficult for us to find.

REFERENCES

1. Abdal, R., Qin, Y., & Wonka, P. (2019). Image2StyleGAN: How to Embed Images into the StyleGAN Latent Space? 2019 IEEE/CVF International Conference on Computer Vision (ICCV 2019), 2019, 4431-4440.
2. Brownlee, J. (2019). How to Implement the Frechet Inception Distance (FID) for Evaluating GANs. Machine Learning Mastery. https://machinelearningmastery.com/how-to-implement-the-frechet-inception- distance-fid-from-scratch
3. Nakhaee, M., Hiemstra, D., Stoelinga, M., Noort, M., Collart-Dutilleul, S., Lecomte, T., & Romanovsky, A. (2019). The Recent Applications of Machine Learning in Rail Track Maintenance: A Survey. Lecture Notes in Computer Science, 91-105.
4. Papers with Code. (2022). Semantic Segmentation on Cityscapes test. Papers with Code. https://paperswithcode.com/sota/semantic-segmentation-on- cityscapes
5. Papers with Code. (2022). Real-Time Semantic Segmentation on Cityscapes test. Papers with Code. https://paperswithcode.com/sota/real-time-semantic- segmentation-on-cityscapes
6. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S. & Schiele, B. (2016). The Cityscapes Dataset for Semantic Urban Scene Understanding. 2016 IEEE Conference On Computer Vision and Pattern Recognition (CVPR), 2016, 3213-3223.
7. Cai, Z., Xiong, Z., Xu, H., Wang, P., Li, W., & Pan, Y. (2021) Generative Adversarial Networks. ACM Computing Surveys, 54(6), 1–38. https://doi.org/10.1145/3459992
8. Chen, L., Yang, Y., Wang, J., Xu, W., & Yuille, A. (2016). Attention to Scale: Scale-Aware Semantic Image Segmentation. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, 3640-3649.
9. Gadhiya, R. & Kalani, N. (2021) Deep Learning Architectures for Semantic Segmentation of 2D Image: A Review. In AIP Conference Proceedings, 2407:1–10. New York: American Institute of Physics. https://doi.org/10.1063/5.0074633.
10. Gondhalekar, A. (2020) Data Augmentation - Is it really necessary? Medium. https://medium.com/analytics-vidhya/data-augmentation-is-it-really- necessary-b3cb12ab3c3f
11. Gibert, X., Patel, V., & Chellappa, R. (2017). Deep Multitask Learning for Railway Track Inspection. IEEE Transactions on Intelligent Transportation Systems, 18(1), 153-164.
12. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative Adversarial Networks. Communications of the ACM 63, no. 11 (2020): 139–44. https://doi.org/10.1145/3422622.
13. Agico Group. (2020). Main Parts of A Railroad Track. https://railroadrails.com/knowlege/main-parts-of-railroad-track/
14. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). GANs trained by a two time-scale update rule converge to a local nash equilibrium. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 6629–6640.
15. Hudson, A., Watson, G., Le Pen, L., & Powrie, W. (2016). Remediation of Mud Pumping on a Ballasted Railway Track. Procedia Engineering, 143, 1043- 1050.
16. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778.
17. Jang, J., Shin, M., Lim, S., Park, J., Kim, J., & Paik, J. (2019). Intelligent Image-Based Railway Inspection System Using Deep Learning-Based Object Detection and Weber Contrast-

Based Image Comparison. Sensors (Basel, Switzerland), 19(21), 4738.
18. James, A., Wang, J., Yang, X., Chenghao, Y., Ngan, N., Yuxin, L., Su, Y., Chandrasekhar, V., & Zeng, Z. (2018). TrackNet - A Deep Learning Based Fault Detection for Railway Track Inspection. 1-5. 10.1109/ICIRT.2018.8641608.
19. Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., & Aila, T. (2020). Training Generative Adversarial Networks with Limited Data. Proceeding NeurIPS. https://github.com/NVlabs/stylegan2-ada-pytorch
20. Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2017). Progressive Growing of GANs for Improved Quality, Stability, and Variation.
21. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., & Aila, T. (2019). Analyzing and Improving the Image Quality of StyleGAN.
22. Karras, T., Laine, S., & Aila, T. (2021). A Style-Based Generator Architecture for Generative Adversarial Networks. IEEE Transac- tions on Pattern Analysis and Machine Intelligence, 43(12), 4217–4228. https://doi.org/10.1109/TPAMI.2020.2970919
23. Kirillov, A., He, K., Girshick, R., Rother, C., & Dollár, P. (2018). Panoptic Segmentation. arXiv. arXiv:1801.00868
24. Lateef, F. and Ruichek, Y. Survey on Semantic Segmentation Using Deep Learning Techniques. Neurocomputing (Amsterdam) 338 (2019): 321–48. https://doi.org/10.1016/j.neucom.2019.02.003. Railway Systems. Modelling, Analysis, Verification, and Certification, pp. 91-105, 2019.
25. Li, X., You, A., Zhu, Z., Zhao, H., Yang, M., Yang, K., & Tong, Y. (2020). Semantic Flow for Fast and Accurate Scene Parsing. In Computer Vision – ECCV 2020 (Vol. 12346, Lecture Notes in Computer Science, pp. 775-793). Cham: Springer International Publishing
26. Li, X., You, A., Zhu, Z., Zhao, H., Yang, M., Yang, K., & Tong, Y. (2022). SFSegNets(ECCV-2020-oral) and SFNet-Lite (Extension). https://github.com/lxtGH/SFSegNets
27. Mittal, S., & Rao, D. (2017). Vision Based Railway Track Monitoring using Deep Learning. arXiv:1711.06423.
28. Neuhold, G., Ollmann, T., Bulo, S., & Kontschieder, P. (2017). The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes. 2017 IEEE International Conference On Computer Vision (ICCV), 2017, 5000-5009.
29. Pohlen, T., Hermans, A., Mathias, M., & Leibe, B. (2017). Full-Resolution Residual Networks for Semantic Segmentation in Street Scenes, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3309-3318, doi: 10.1109/CVPR.2017.353.
30. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., & Fei- Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision, 115(3), 211-252.
31. Rosebrock, A. (2022). Intersection over Union (IoU) for object detection. pyimagesearch. https://pyimagesearch.com/2016/11/07/intersection-over- union-iou-for-object-detection/
32. Ritika, S. & D. Rao, (2018). Data Augmentation of Railway Images for Track Inspection, arXiv:1802.01286.
33. Shorten, & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. Journal of Big Data, 6(1), 1–48. https://doi.org/10.1186/s40537-019-0197-0
34. Shah, M. P. (2017), Semantic Segmentation Architectures Implemented in PyTorch. https://github.com/meetshah1995/pytorch-semseg
35. Shelhamer, E., Long, J., & Darrell, T. (2017). Fully Convolutional Networks for Semantic

Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(4), 640-651.
36. Benchmark Suite. (2022). Cityscapes Dataset. https://www.cityscapes-dataset.com/benchmarks/#scene-labeling-task
37. Supervisely. (2022). Supervisely. https://supervise.ly
38. Tao, A., Sapra, K, and Catanzaro, B. (2020). Hierarchical Multi-Scale Attention for Semantic Segmentation. arXiv. arXiv:2005.10821
39. Tao, A., Sapra, K, and Catanzaro, B. (2020). semantic-segmentation. https://github.com/NVIDIA/semantic-segmentation
40. Yuan, Y., Chen, X., Chen, X., & Wang, J. (2019). Segmentation Trans- former: Object-Contextual Representations for Semantic Segmentation. arXiv. arXiv:1909.11065
41. Yu, F., & Koltun, V. (2016). Multi-Scale Context Aggregation by Dilated Convolutions. arXiv. arXiv:1511.07122
42. Zendel, O., Murschitz, M., Zeilinger, M., Steininger, D., Abbasi, S., & Beleznai, C. (2019). RailSem19: A Dataset for Semantic Rail Scene Understanding. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 1221-1229.
43. Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J. (2017). Pyramid Scene Parsing Network. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2881-2890

APPENDIX A: ANNOTATED IMAGES IN OUR DATASET

Below are some of the annotated images in the custom dataset. For the color legend, see Table 4.1.

Image **Ground Truth**

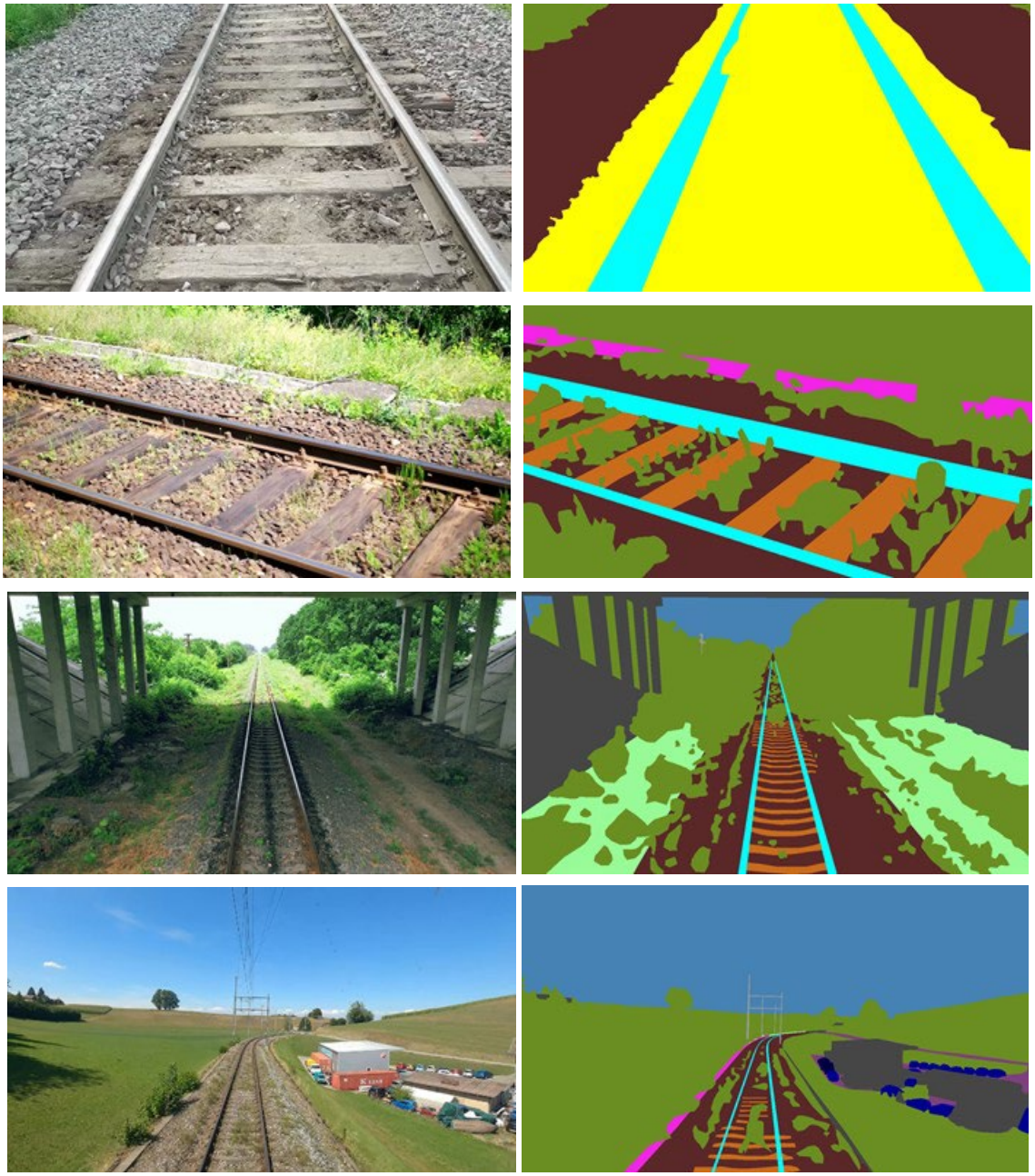


Image **Ground Truth**

ACKNOWLEDGEMENTS

The authors wish to thank and acknowledge the US Department of Transportation, University Transportation Center Program (RailTEAM UTC) for funding support for this research. Additionally, the authors wish to acknowledge CSX Transportation for providing data for this research.

ABOUT THE AUTHORS

Paul Stanik III was a graduate research assistant when he worked on this research project. He obtained his Bachelor of Science in Computer Science in 2020 with minor in Mathematics from the University of Nevada, Las Vegas

Brendan Morris was an Associate Professor in the Department(s) of Electrical and Computer Engineering in the University of Nevada Las Vegas. His expert areas include computer vision, intelligent systems, pattern recognition, machine learning, intelligent transportation systems, and intelligent vehicles. In the area of unmanned aerial systems, he is specialized in on-board computer vision and image processing, object detection and recognition, selective filtering, surround awareness, and sense and avoid technology. He obtained his Bachelor degree from the University of California at Berkeley and his Ph.D. from the University of California at San Diego.