CONTEXT-INVARIANT AUTOENCODER TRAINING VIA UNSUPERVISED DOMAIN ADAPTATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS OF
THE MIDDLE EAST TECHNICAL UNIVERSITY
BY

ÖZGE KÖKTÜRK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF DATA INFORMATICS

MARCH 2025

CONTEXT-INVARIANT AUTOENCODER TRAINING VIA UNSUPERVISED DOMAIN ADAPTATION

Submitted by Özge KÖKTÜRK in partial fulfillment of the requirements for the degree of **Master of Science in Data Informatics Department, Middle East Technical University** by,

Prof. Dr. Banu GÜNEL KILIÇ
Dean, **Graduate School of Informatics** _____

Prof. Dr. Pekin Erhan EREN
Head of Department, **Data Informatics** _____

Prof. Dr. Altan KOÇYİĞİT
Supervisor, **Information Systems Dept., METU** _____

Examining Committee Members:

Prof. Dr. Pekin Erhan EREN
Information Systems Dept., METU _____

Prof. Dr. Altan KOÇYİĞİT
Information Systems Dept., METU _____

Assoc. Prof. Dr. Serhat PEKER
Management Information Systems, Bakırçay University _____

Date: **27.03.2025**

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Özge KÖKTÜRK

Signature : _____

ABSTRACT

CONTEXT-INVARIANT AUTOENCODER TRAINING VIA UNSUPERVISED DOMAIN ADAPTATION

KÖKTÜRK, ÖZGE
MSc., Department of Data Informatics
Supervisor: Prof. Dr. Altan KOÇYİĞİT

March 2025, 213 pages

In practical use of machine learning models, generalizability is of crucial importance. When a model is trained on a dataset obtained in a specific context, it often performs poorly in similar situations but under different contexts. This can lead to unreliable predictions and potentially harmful decisions in real-life applications. This thesis proposes a training methodology for context-invariant autoencoders through unsupervised domain adaptation, aiming to learn representations that remain stable across varying contexts. Consequently, any application can be built on top of these domain-invariant representations. In this study, domain-adversarial training and data augmentation strategies have been employed to extract features that capture the essential structures of input images while disregarding features associated with contextual changes. For the experiments, image data collected from the CARLA (Car Learning to Act) simulator system under different weather conditions and various times of day have been used.

Keywords: Context-Invariant Autoencoders, Unsupervised Domain Adaptation, Domain-Adversarial Training, Data Augmentation, CARLA Simulator

ÖZ

DENETİMSIZ ALAN UYARLAMASI İLE BAĞLAMDAN BAĞIMSIZ OTOKODLAYICI EĞİTİMİ

KÖKTÜRK, Özge
Yüksek Lisans, Enformatik Enstitüsü
Tez Yöneticisi: Prof. Dr. Altan KOÇYİĞİT

Mart 2025, 213 sayfa

Makine öğrenimi modellerinin pratikte kullanımında, genellenebilirlik büyük öneme haizdir. Belirli bir bağlamda elde edilmiş bir veri kümesi üzerinde eğitilen bir model, farklı bağlamlardaki benzer durumlarda genellikle kötü bir başarım sergiler. Bu durum, gerçek hayat uygulamalarında güvenilmez tahminlere ve potansiyel olarak zararlı kararlara yol açabilir. Bu tez denetimsiz alan uyarlaması yoluyla bağlamdan bağımsız otokodlayıcılar için bir eğitim yöntemi önermekte ve farklı bağlamlarda tutarlı kalan temsili özellikleri öğrenmeyi hedeflemektedir. Sonuç olarak herhangi bir uygulama bu alan-bağımsız temsili özelliklerin üzerine inşa edilebilir. Bu çalışmada, giriş görüntülerinin temel yapılarını yakalayan ancak bağlamsal değişikliklerle ilişkili özellikleri göz ardı eden özellikleri çıkarmak için alan-çekişmeli eğitim ve veri artırma stratejileri kullanılmıştır. Deneylerde farklı hava koşulları ve günün çeşitli saatlerinde CARLA (Car Learning to Act) benzetim sisteminden toplanan görüntü verileri kullanılmıştır.

Anahtar Sözcükler: Bağlamdan Bağımsız Otokodlayıcılar, Denetimsiz Alan Uyarlaması, Alan-Çekişmeli Eğitim, Veri Artırma, CARLA Simülatör

To My Family and To My Fiancé

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor Prof. Dr. Altan KOÇYİĞİT, for his invaluable guidance, insightful feedback, and support throughout this study. Your knowledge and encouragement have been pivotal in shaping this work. I am also proudly thankful to my family for their endless support and encouragement. Your belief in me has been my greatest motivation. To my fiancé, your understanding, patience, and unwavering support have meant the world to me.

TABLE OF CONTENTS

ABSTRACT iv
ÖZ v
DEDICATION vi
ACKNOWLEDGMENTS vii
TABLE OF CONTENTS viii
LIST OF TABLES x
LIST OF FIGURES xiv
LIST OF ACRONYMS xix
CHAPTER

1. INTRODUCTION 1
   1.1. Objective 2
   1.2. Scope 3
   1.3. Methodology 3
   1.4. Thesis Organization 5
2. BACKGROUND 7
   2.1. Carla Simulator and Data Generation 7
   2.2. Autoencoders and Domain-Invariant Representations 8
   2.3. Domain Adaptation 11
3. RELATED WORK 13
   3.1. Autoencoders in Machine Learning 13
   3.2. Domain Adaptation in Machine Learning 14
   3.3. Domain-Adversarial and Feature Learning in Machine Learning 15
   3.4. Novelty of This Work 16
4. DATA GENERATION 17
5. CONTEXT-INVARIANT AUTOENCODER TRAINING 23
   5.1. Modeling 23

5.2. Learning....26
5.2.1. Training Process....26
5.2.2. Optimization and Adversarial Learning....27
5.2.3. Performance Metrics....27
5.3. Experiments....30
5.3.1. Implementation Details....30
5.3.2. Tools and Frameworks....35
5.4. Model Results and Analysis....38
5.4.1. Reconstruction Results....39
5.4.2. Domain Classifier Results....55
5.4.3. Denoising Autoencoder Results....60
5.4.4. Model Discussions....65
6. CONCLUSION....77
REFERENCES....79
APPENDICES....83
APPENDIX A....83
APPENDIX B....91
APPENDIX C....99
APPENDIX D....107
APPENDIX E....119
APPENDIX F....131
APPENDIX G....143
APPENDIX H....155
APPENDIX I....167
APPENDIX J....179
APPENDIX K....193
APPENDIX L....199
APPENDIX M....211

LIST OF TABLES

Table 1: CARLA Weather Parameters (Data sourced from https://carla.org/Doxygen/html/db/ddb/classcarla_1_1rpc_1_1WeatherParameters.html) 19
Table 2: Key Hyperparameters and Their Impact on Autoencoder Performance. This table comprises important hyperparameters and their impact on model performance..... 30
Table 3: The Hyperparameters of The Base Model. This table shows the hyperparameter values used in the base model..... 31
Table 4: Model Runs Based on Domain Adversarial Autoencoder Architecture. This table shows the model runs assigned for domain adversarial autoencoder architecture. 32
Table 5: Model Runs Based on Source Only Autoencoder Architecture. This table shows the model runs assigned for source only autoencoder architecture..... 33
Table 6: Model Runs Based on Domain Adversarial Autoencoder Architecture with Noise Function. This table shows the model runs assigned for domain adversarial autoencoder architecture by adding noise function..... 34
Table 7: Tools and Frameworks Used in Models. This table explains which tools and frameworks are used with which purpose and their key features. 36
Table 8: CIAE Train Reconstruction Loss Results (epoch = 100). This table shows the CIAE training results regarding reconstruction loss which is combined with MSE loss and SSIM loss..... 41
Table 9: SOAE Train Reconstruction Loss Results (epoch = 100). This table shows the SOAE training results regarding reconstruction loss which is combined with MSE loss and SSIM loss..... 41
Table 10: CIAE Train Reconstruction Loss Results (epoch = 500). This table shows the CIAE training results regarding reconstruction loss which is combined with MSE loss and SSIM loss..... 42
Table 11: SOAE Train Reconstruction Loss Results (epoch = 500). This table shows the SOAE training results regarding reconstruction loss which is combined with MSE loss and SSIM loss..... 43
Table 12: CIAE Train Reconstruction Loss Results (epoch = 1000). This table shows the CIAE training results regarding reconstruction loss which is combined with MSE loss and SSIM loss..... 44
Table 13: SOAE Train Reconstruction Loss Results (epoch = 1000). This table shows the SOAE training results regarding reconstruction loss which is combined with MSE loss and SSIM loss..... 45
Table 14: CIAE Validation Reconstruction Loss Results (epoch = 100). This table shows the CIAE validation results regarding reconstruction loss which is combined with MSE loss and SSIM loss..... 46

Table 15: SOAE Validation Reconstruction Loss Results (epoch = 100). This table shows the SOAE validation results regarding reconstruction loss which is combined with MSE loss and SSIM loss.....46
Table 16: CIAE Validation Reconstruction Loss Results (epoch = 500). This table shows the CIAE validation results regarding reconstruction loss which is combined with MSE loss and SSIM loss.....47
Table 17: SOAE Validation Reconstruction Loss Results (epoch = 500). This table shows the SOAE validation results regarding reconstruction loss which is combined with MSE loss and SSIM loss.....48
Table 18: CIAE Validation Reconstruction Loss Results (epoch = 1000). This table shows the CIAE validation results regarding reconstruction loss which is combined with MSE loss and SSIM loss.....49
Table 19: SOAE Validation Reconstruction Loss Results (epoch = 1000). This table shows the SOAE validation results regarding reconstruction loss which is combined with MSE loss and SSIM loss.....50
Table 20: CIAE Test Reconstruction Loss Results (epoch = 100). This table shows the CIAE test results regarding reconstruction loss which is combined with MSE loss and SSIM loss.....51
Table 21: SOAE Test Reconstruction Loss Results (epoch = 100). This table shows the SOAE test results regarding reconstruction loss which is combined with MSE loss and SSIM loss.....52
Table 22: CIAE Test Reconstruction Loss Results (epoch = 500). This table shows the CIAE test results regarding reconstruction loss which is combined with MSE loss and SSIM loss.....53
Table 23: SOAE Test Reconstruction Loss Results (epoch = 500). This table shows the SOAE test results regarding reconstruction loss which is combined with MSE loss and SSIM loss.....53
Table 24: CIAE Test Reconstruction Loss Results (epoch = 1000). This table shows the CIAE test results regarding reconstruction loss which is combined with MSE loss and SSIM loss.....54
Table 25: SOAE Test Reconstruction Loss Results (epoch = 1000). This table shows the SOAE test results regarding reconstruction loss which is combined with MSE loss and SSIM loss.....55
Table 26: CIAE Domain Classifier Results (epoch = 100). This table shows the CIAE domain classifier results regarding adversarial loss, accuracy, learning rate, and lambda values.....58
Table 27: CIAE Domain Classifier Results (epoch = 500). This table shows the CIAE domain classifier results regarding adversarial loss, accuracy, learning rate, and lambda values.....59

Table 28: CIAE Domain Classifier Results (epoch = 1000). This table shows the CIAE domain classifier results regarding adversarial loss, accuracy, learning rate, and lambda values..... 60
Table 29: Denoising Autoencoder Train Reconstruction Loss Results (epoch = 1000). This table shows the CIAE training results with added noise function regarding reconstruction loss which is combined with MSE loss and SSIM loss. 61
Table 30: Denoising Autoencoder Validation Reconstruction Loss Results (epoch = 1000). This table shows the CIAE validation results with added noise function regarding reconstruction loss which is combined with MSE loss and SSIM loss. 62
Table 31: Denoising Autoencoder Test Reconstruction Loss Results (epoch = 1000). This table shows the CIAE test results with added noise function regarding reconstruction loss which is combined with MSE loss and SSIM loss. 63
Table 32: Denoising CIAE Domain Classifier Results (epoch = 1000). This table shows the CIAE domain classifier results regarding adversarial loss, accuracy, learning rate, and lambda values..... 64
Table 33: CIAE Model Performance Comparison Based on Batch Size. This table shows how batch size affects the model performances in terms of train/validation/test reconstruction loss. 66
Table 34: SOAE Model Performance Comparison Based on Batch Size. This table shows how batch size affects the model performances in terms of train/validation/test reconstruction loss. 67
Table 35: CIAE Model Performance Comparison Based on Batch Size. This table shows how batch size affects the model performances in terms of train/validation/test domain classifier results. 68
Table 36: CIAE Model Performance Comparison Based on Latent Dimension. This table shows how latent dimension affects the model performances in terms of train/validation/test reconstruction loss 69
Table 37: SOAE Model Performance Comparison Based on Latent Dimension. This table shows how latent dimension affects the model performances in terms of train/validation/test reconstruction loss. 70
Table 38: CIAE Model Performance Comparison Based on Latent Dimension. This table shows how latent dimension affects the model performances in terms of train/validation/test domain classifier results..... 72
Table 39: The Top Three CIAE Model Configurations Based on Test Accuracy. This table shows the top three model configurations based on test accuracy..... 72
Table 40: The Top Three Model Configurations Based on Test Reconstruction Loss. This table shows the top three model configurations based on test reconstruction loss..... 73
Table 41: Comparison of CIAE and Denoising CIAE Models in Terms of Reconstruction Loss, MSE And SSIM Loss. This table shows the model runs based on CIAE and denoising CIAE models for 1000 epochs. 74

Table 42: Comparison of CIAE and Denoising CIAE Models in Terms of Domain Adversarial Loss and Domain Accuracy This table shows the model runs based on CIAE and denoising CIAE models for 1000 epochs.....75

LIST OF FIGURES

Figure 1: The block diagram of autoencoder and domain classifier architecture..... 9
Figure 2: CARLA Town 07..... 17
Figure 3: CARLA RGB camera output image..... 18
Figure 4: CARLA simulator route..... 20
Figure 5: The CIAE model structure in terms of encoder, decoder, and domain classifier. 24
Figure 6: CIAE train reconstruction loss, MSE loss, and SSIM loss results for 100 epochs 83
Figure 7: SOAE train reconstruction loss, MSE loss, and SSIM loss results for 100 epochs 84
Figure 8: CIAE validation reconstruction loss, MSE loss, and SSIM loss results for 100 epochs..... 84
Figure 9: SOAE validation reconstruction loss, MSE loss, and SSIM loss results for 100 epochs..... 85
Figure 10: CIAE test reconstruction loss, MSE loss, and SSIM loss results for 100 epochs 86
Figure 11: SOAE test reconstruction loss, MSE loss, and SSIM loss results for 100 epochs 86
Figure 12: Average CIAE/SOAE reconstruction loss, MSE loss, and SSIM loss comparison for 100 epochs..... 87
Figure 13: CIAE train-validation-test adversarial loss results for 100 epochs..... 88
Figure 14: CIAE validation - test accuracy results for 100 epochs..... 89
Figure 15: CIAE lambda and learning rate values for 100 epochs 89
Figure 16: SOAE learning rate values for 100 epochs..... 90
Figure 17: CIAE train reconstruction loss, MSE loss, and SSIM loss results for 500 epochs 91
Figure 18: SOAE train reconstruction loss, MSE loss, and SSIM loss results for 500 epochs..... 92
Figure 19: CIAE validation reconstruction loss, MSE loss, and SSIM loss results for 500 epochs..... 92
Figure 20: SOAE validation reconstruction loss, MSE loss, and SSIM loss results for 500 epochs..... 93
Figure 21: CIAE test reconstruction loss, MSE loss, and SSIM loss results for 500 epochs 93
Figure 22: SOAE test reconstruction loss, MSE loss, and SSIM loss results for 500 epochs 94
Figure 23: Average CIAE/SOAE reconstruction loss, MSE loss, and SSIM loss comparison for 500 epochs..... 94
Figure 24: CIAE train-validation-test adversarial loss results for 500 epochs..... 95

Figure 25: CIAE validation-test accuracy results for 500 epochs 96
Figure 26: CIAE lambda and learning rate values for 500 epochs 96
Figure 27: SOAE learning rate values for 500 epochs 97
Figure 28: CIAE train reconstruction loss, MSE loss, and SSIM loss results for 1000 epochs.....99
Figure 29: SOAE train reconstruction loss, MSE loss, and SSIM loss results for 1000 epochs.....100
Figure 30: CIAE validation reconstruction loss, MSE loss, and SSIM loss results for 1000 epochs.....100
Figure 31: SOAE validation reconstruction loss, MSE loss, and SSIM loss results for 1000 epochs.....101
Figure 32: CIAE test reconstruction loss, MSE loss, and SSIM loss results for 1000 epochs 101
Figure 33: SOAE test reconstruction loss, MSE loss, and SSIM loss results for 1000 epochs.....102
Figure 34: Average CIAE/SOAE reconstruction loss, MSE loss, and SSIM loss comparison for 1000 epochs.....102
Figure 35: CIAE train-validation-test adversarial loss results for 1000 epochs 103
Figure 36: CIAE validation-test accuracy results for 1000 epochs 104
Figure 37: CIAE lambda and learning rate values for 1000 epochs 104
Figure 38: SOAE learning rate values for 1000 epochs 105
Figure 39: CIAE_model_bs_08_100 107
Figure 40: CIAE_model_bs_16_100 108
Figure 41: CIAE_model_bs_32_100 109
Figure 42: CIAE_model_dr_01_100.....110
Figure 43: CIAE_model_dr_03_100.....111
Figure 44: CIAE_model_dr_05_100.....112
Figure 45: CIAE_model_ld_128_100.....113
Figure 46: CIAE_model_ld_256_100.....114
Figure 47: CIAE_model_ld_512_100.....115
Figure 48: CIAE_model_ttv_602020_100 116
Figure 49: CIAE_model_ttv_701515_100 117
Figure 50: CIAE_model_ttv_801010_100 118
Figure 51: SOAE_model_bs_08_100 119
Figure 52: SOAE_model_bs_16_100 120
Figure 53: SOAE_model_bs_32_100 121
Figure 54: SOAE_model_dr_01_100.....122
Figure 55: SOAE_model_dr_03_100.....123
Figure 56: SOAE_model_dr_05_100.....124
Figure 57: SOAE_model_ld_128_100.....125
Figure 58: SOAE_model_ld_256_100.....126

Figure 59: SOAE_model_ld_512_100 127
Figure 60: SOAE_model_ttv_602020_100 128
Figure 61: SOAE_model_ttv_701515_100 129
Figure 62: SOAE_model_ttv_801010_100 130
Figure 63: CIAE_model_bs_08_500..... 131
Figure 64: CIAE_model_bs_16_500..... 132
Figure 65: CIAE_model_bs_32_500..... 133
Figure 66: CIAE_model_dr_01_500..... 134
Figure 67: CIAE_model_dr_03_500..... 135
Figure 68: CIAE_model_dr_05_500..... 136
Figure 69: CIAE_model_ld_128_500 137
Figure 70: CIAE_model_ld_256_500 138
Figure 71: CIAE_model_ld_512_500 139
Figure 72: CIAE_model_ttv_602020_500 140
Figure 73: CIAE_model_ttv_701515_500 141
Figure 74: CIAE_model_ttv_801010_500 142
Figure 75: SOAE_model_bs_08_500..... 143
Figure 76: SOAE_model_bs_16_500..... 144
Figure 77: SOAE_model_bs_32_500..... 145
Figure 78: SOAE_model_dr_01_500..... 146
Figure 79: SOAE_model_dr_03_500..... 147
Figure 80: SOAE_model_dr_05_500..... 148
Figure 81: SOAE_model_ld_128_500 149
Figure 82: SOAE_model_ld_256_500 150
Figure 83: SOAE_model_ld_512_500 151
Figure 84: SOAE_model_ttv_602020_500 152
Figure 85: SOAE_model_ttv_701515_500 153
Figure 86: SOAE_model_ttv_801010_500 154
Figure 87: CIAE_model_bs_08_1000..... 155
Figure 88: CIAE_model_bs_16_1000..... 156
Figure 89: CIAE_model_bs_32_1000..... 157
Figure 90: CIAE_model_dr_01_1000..... 158
Figure 91: CIAE_model_dr_03_1000..... 159
Figure 92: CIAE_model_dr_05_1000..... 160
Figure 93: CIAE_model_ld_128_1000 161
Figure 94: CIAE_model_ld_256_1000 162
Figure 95: CIAE_model_ld_512_1000 163
Figure 96: CIAE_model_ttv_602020_1000..... 164
Figure 97: CIAE_model_ttv_701515_1000..... 165
Figure 98: CIAE_model_ttv_801010_1000..... 166
Figure 99: SOAE_model_bs_08_1000..... 167

Figure 100: SOAE_model_bs_16_1000.....168
Figure 101: SOAE_model_bs_32_1000.....169
Figure 102: SOAE_model_dr_01_1000.....170
Figure 103: SOAE_model_dr_03_1000.....171
Figure 104: SOAE_model_dr_05_1000.....172
Figure 105: SOAE_model_ld_128_1000.....173
Figure 106: SOAE_model_ld_256_1000.....174
Figure 107: SOAE_model_ld_512_1000.....175
Figure 108: SOAE_model_ttv_602020_1000.....176
Figure 109: SOAE_model_ttv_701515_1000.....177
Figure 110: SOAE_model_ttv_801010_1000.....178
Figure 111: CIAE train reconstruction loss comparison based on epochs and batch size.....179
Figure 112: CIAE validation reconstruction loss comparison based on epochs and batch size.....180
Figure 113: CIAE test reconstruction loss comparison based on epochs and batch size.....180
Figure 114: SOAE train reconstruction loss comparison based on epochs and batch size.....181
Figure 115: SOAE validation reconstruction loss comparison based on epochs and batch size.....181
Figure 116: SOAE test reconstruction loss comparison based on epochs and batch size.....182
Figure 117: CIAE train adversarial loss comparison based on epochs and batch size..183
Figure 118: CIAE validation adversarial loss comparison based on epochs and batch size.....183
Figure 119: CIAE test adversarial loss comparison based on epochs and batch size....184
Figure 120: CIAE validation accuracy comparison based on epochs and batch size....184
Figure 121: CIAE test accuracy comparison based on epochs and batch size.....185
Figure 122: CIAE train reconstruction loss comparison based on epochs and latent dimension.....185
Figure 123: CIAE validation reconstruction loss comparison based on epochs and latent dimension.....186
Figure 124: CIAE test reconstruction loss comparison based on epochs and latent dimension.....186
Figure 125: SOAE train reconstruction loss comparison based on epochs and latent dimension.....187
Figure 126: SOAE validation reconstruction loss comparison based on epochs and latent dimension.....188
Figure 127: SOAE test reconstruction loss comparison based on epochs and latent dimension.....188

Figure 128: CIAE train adversarial loss comparison based on epochs and latent dimension 189
Figure 129: CIAE validation adversarial loss comparison based on epochs and latent dimension..... 189
Figure 130: CIAE test adversarial loss comparison based on epochs and latent dimension 190
Figure 131: CIAE validation accuracy comparison based on epochs and latent dimension 190
Figure 132: CIAE test accuracy comparison based on epochs and latent dimension... 191
Figure 133: Denoising CIAE train reconstruction loss, MSE loss, and SSIM loss results for 1000 epochs..... 193
Figure 134: Denoising CIAE validation reconstruction loss, MSE loss, and SSIM loss results for 1000 epochs 194
Figure 135: Denoising CIAE test reconstruction loss, MSE loss, and SSIM loss results for 1000 epochs..... 194
Figure 136: Average Denoising CIAE reconstruction loss, MSE loss, and SSIM loss comparison for 1000 epochs..... 195
Figure 137: Denoising CIAE train-validation-test adversarial loss results for 1000 epochs 196
Figure 138: Denoising CIAE validation - test accuracy results for 1000 epochs..... 197
Figure 139: Denoising CIAE lambda and learning rate values for 1000 epochs 197
Figure 140: CIAE_model_denoising_bs_08_1000 199
Figure 141: CIAE_model_denoising_bs_16_1000 200
Figure 142: CIAE_model_denoising_bs_32_1000 201
Figure 143: CIAE_model_denoising_dr_01_1000..... 202
Figure 144: CIAE_model_denoising_dr_03_1000..... 203
Figure 145: CIAE_model_denoising_dr_05_1000..... 204
Figure 146: CIAE_model_denoising_ld_128_1000..... 205
Figure 147: CIAE_model_denoising_ld_256_1000..... 206
Figure 148: CIAE_model_denoising_ld_512_1000..... 207
Figure 149: CIAE_model_denoising_ttv_602020_1000 208
Figure 150: CIAE_model_denoising_ttv_701515_1000 209
Figure 151: CIAE_model_denoising_ttv_801010_1000 210
Figure 152: Average CIAE/Denoising CIAE reconstruction loss, MSE loss, and SSIM loss comparison for 1000 epochs..... 211
Figure 153: CIAE/Denoising CIAE train-validation-test adversarial loss results for 1000 epochs..... 212
Figure 154: CIAE/Denoising CIAE validation - test accuracy results for 1000 epochs213

LIST OF ACRONYMS

AAE	Adversarial Autoencoders
ADAS	Advanced Driver Assistance Systems
ADDA	Adversarial Discriminative Domain Adaptation
API	Application Programming Interface
ASAM	Adaptive Structural Alignment Method
CAE	Convolutional Autoencoders
CARLA	Car Learning to Act
CIAE	Context Invariant Autoencoder
CORAL	Correlation Alignment
DANN	Domain-Adversarial Neural Network
GPS	Global Positioning System
LIDAR	Light Detection and Ranging Sensors
MMD	Maximum Mean Discrepancy
MSE	Mean-Squared-Error
NLP	Natural Language Processing
PCA	Principal Component Analysis
RGB	Red Green Blue
ROS	Robot Operating System
SOAE	Source Only Autoencoder
SSIM	Structural Similarity Index Measure
VAE	Variational Autoencoder
WandB	Weights and Bias

CHAPTER 1

INTRODUCTION

Autoencoders are known as one of the most powerful tools for unsupervised learning and due to their ability to learn efficient data representations they have gained significant attention (Pascal Vincent et al., 2010). Even they are used widely, these models face with crucial challenges when exposed to domain shifts – differences between the contexts in which the models are trained and deployed. These shifts often happen due to variations in environmental conditions like lighting, weather, or other contextual changes which might severely degrade the performance of autoencoders, limiting their ability in real-world applications (Zhu et al., 2017).

Addressing domain shift issues requires methods that enable models to generalize effectively across domains. With the contribution of domain adaptation techniques, representations from different contexts are allowed without labeled target data. By applying domain adaptation techniques, the learned representations can be made context-invariant and they show robust performance across different scenarios (Ganin et al., 2015). In this study, we focused on context-invariant representation extraction using autoencoders enriched with domain adaptation strategies.

In the proposed framework, domain-adversarial training and data augmentation are combined to archive context-invariant autoencoder training. To guarantee that the encoder learns features that are stable across many contexts, domain-adversarial training uses an adversarial learning technique with a domain classifier (Ganin & Lempitsky, 2014). At the same time, data augmentation techniques are employed to simulate different scenarios, improving the model's capacity to ignore context-specific variations and capture the fundamental structure of input data.

This study uses a dataset generated from the CARLA (Car Learning to Act) simulation platform, a versatile system capable of simulating diverse environmental conditions such as different weather patterns and times of day to evaluate the effectiveness of the proposed framework (Dosovitskiy et al., 2017). The experiments provide an advanced comparison of autoencoder models under different hyperparameter configurations. By altering latent dimension size, batch normalization, dropout rates, and train-test-validation distribution, their impact on model performance was analyzed and investigated.

The potential circumstances of combining unsupervised domain adaptation techniques with autoencoders to address domain shifts is highlighted in this research. The results

reflect improvements in reconstruction loss and feature robustness which is critical in terms of the value of this approach in producing domain-invariant representations.

Throughout my professional career, I have encountered various real-world challenges that required innovative solutions. One notable example was a project focused on the production of electronic mirrors for vehicles. In this project, the developers had to consider all possible driving scenarios, ensuring that the system performed optimally under different conditions. One of the most critical challenges arose in tunnels. When a vehicle enters a tunnel, the sudden change in lighting conditions can cause temporary blindness for the driver, significantly reducing visibility and increasing the risk of accidents.

To address this issue, electronic mirrors were implemented to provide an enhanced and adaptive view, helping drivers navigate safely in such environments. However, this solution could be further improved using domain adaptation techniques, which would allow the system to adapt to different weather conditions as well as other challenging environments like underground parking lots or densely shadowed roads. By integrating AI-driven enhancements, the electronic mirrors could dynamically adjust brightness, contrast, and real-time reflections, ensuring optimal visibility regardless of external factors. This approach not only improves safety but also paves the way for more advanced driver assistance systems (ADAS) in the future.

This research emphasizes the need to create machine learning models that can reliably run in diverse and dynamic environments. Domain adaptation techniques are used to improve autoencoders, this study seeks to handle the limitations of conventional approaches, showing the way for more adaptable and robust systems. The main focus in this study is to address the impact of hyperparameters, such as latent dimension size, batch normalizations, and dropout, train-test-validation dataset distribution, considering the study's commitment to optimize model performance across a variety of contexts.

Lastly, another focus in this study is the challenge of generalizability which is related to creating robust machine learning models that take part in a wide range of contexts in the literature. This study can also be extended as a future work changing its focus to semantic segmentation, object detection, or autonomous driving.

1.1. Objective

The primary objective of this research is to build and assess a framework by utilizing unsupervised domain adaptation techniques for investigating the use of context-invariant autoencoder training. The research is concentrated on:

- Investigation of domain adaptation techniques, such as data augmentation and domain-adversarial training, in order to examine how to improve the generalization capacities of autoencoders in diverse contexts.

- Evaluation and comparison of the impact of key hyperparameters, such as latent dimension size, dropout rates, batch size, and train-test-validation distribution, on autoencoder performance during domain shifts.

In the light of this study, a robust framework is targeted to draw attention to the challenges associated with contextual shifts while assuring reliable autoencoder performance in various dynamic conditions. Hence, this study discusses the issue of how to extract context-invariant features needed by reliable and flexible machine learning models.

1.2. Scope

The scope of this study is to focus on the development and evaluation of autoencoders improved by using unsupervised domain adaptation techniques to address the challenges due to domain shifts. Accordingly, we involved:

- The implementation of domain-adversarial training focusing on domain invariance by using domain classifier and adversarial learning techniques.

- Data augmentation techniques to simulate dynamic environmental conditions and improve the model performance.

- Comparison of the key hyperparameters, such as latent dimension size, batch normalization, dropout, and train-test-validation splits, to optimize model performance.

- Experimental analysis on CARLA Simulator based generated data, including different weather conditions and time of day, to simulate domain shifts.

On the other hand, we have not considered full deployment in real-world systems. The main focus is to investigate generalizability through the use of simulated and well-defined contexts.

1.3. Methodology

To develop context-invariant autoencoders by applying integration of unsupervised domain adaptation techniques, a systematic and experimental approach has been applied. The process includes:

- Autoencoder Architecture Design:
    - To obtain compact latent representations encoder-decoder architecture has been utilized.
    - To ensure domain invariance in the learned features domain classifier has been employed.
- Domain-Adversarial Training:
    - Gradient reversal layer used to establish adversarial learning to force the encoder to produce domain-invariant representations by confusing the domain classifier.
- Data Augmentation:
    - Dynamic environmental conditions have been simulated in CARLA Simulator by using data augmentation techniques to train the model within a wide range of context.
- Experimental Evaluation:
    - Datasets have been gathered in CARLA Simulator by capturing different weather conditions and times of day to perform train-test-validation steps.
- Performance Analysis:
    - Models have been evaluated based on robustness of reconstruction loss and adversarial loss.
    - The impact of key hyperparameters such as latent dimension size, batch normalization, dropout rates, train-test-validation dataset split distribution, and number of epochs have been analyzed.
    - CIAE (Context Invariant Autoencoder) models have been compared with SOAE (Source Only Autoencoder) results.
    - Noise factor has been added to CIAE models to obtain more realistic result while considering real world image qualities.

With all these steps, this study aims to improve the performance of the model combining domain adaptation techniques with autoencoders to obtain robust models capable of generalizing across contexts.

1.4. Thesis Organization

This thesis is structured into six main chapters, each serving a specific purpose in presenting the research conducted. This Introduction chapter establishes the foundation for the study, explaining the motivation behind the research, the objective of the work, and its scope. It also outlines the methodology adopted to address the research question and provides a roadmap for the thesis organization.

Chapter 2 delves into the fundamental concepts and technologies relevant to the research. It introduces CARLA, a simulation platform for autonomous driving research, and explains the working principles of autoencoders and domain adaptation techniques, which form the theoretical basis for the proposed framework. This chapter ensures that readers understand the core concepts necessary to comprehend the study. Additionally, the principles of autoencoders and domain adaptation techniques are discussed to establish the knowledge framework essential for understanding the proposed methodology.

Chapter 3 provides an extensive review of existing literature, focusing on domain adaptation and weather robustness in autonomous driving. It outlines and compares related studies and highlights the gaps in current research. The chapter emphasizes the novelty of the proposed approach, demonstrating its potential contributions to the field.

Chapter 4 focuses on a detailed explanation of data collection and preparation processes. Moreover, data augmentation techniques are explained.

Chapter 5 details the experimental framework employed in the study. It describes the design of the model architecture, and the integration of domain-adversarial training. Key aspects such as optimization strategies, performance metrics, and the experimental setup are discussed comprehensively. Additionally, this chapter presents the results of the experiments, their analysis, and the limitations observed during the study. In this chapter, SOAE comparisons are also discussed to compare the CIAE model results.

Finally, Chapter 6 summarizes the findings of the research and discusses the contributions made to the field. It provides insights into how the study advances the understanding of domain adaptation for autoencoders and proposes directions for future work. The thesis concludes with a list of references cited throughout the document and appendices that include supplementary materials, such as detailed experimental configurations and additional results.

CHAPTER 2

BACKGROUND

This chapter outlines the kex concepts and tools used in the study, including the CARLA simulator, autoencoders, and domain adaptation. CARLA provides a cost-effective platform to generate diverse image datasets in a range of weather situations. Autoencoders enable image reconstruction with their encoder-decoder structure. In the thesis, autoencoders are encouraged to learn domain-invariant features using domain-adversarial training with a gradient reversal layer. In particular, domain adaptation techniques improve model generalization in dynamic environments.

2.1. Carla Simulator and Data Generation

CARLA is introduced as an open-source simulator for autonomous driving research (Dosovitskiy et al., 2017). CARLA is highly preferred to use for development, training, and validation of studies in autonomous driving systems. It is a well-structured platform due to its comprehensive open-source ability, code and protocols contribution, which are accessed freely by researchers and developers. CARLA also provides an extensive set of open digital assets, such as urban layouts, buildings, and vehicles, especially designed for use in autonomous vehicle testing. Thanks to its advanced features, the flexible specification of sensor suites, the ability to simulate varied environmental conditions, and full control over static and dynamic actors can be enabled. Additionally, the custom maps are generated and complex driving scenarios are simulated with CARLA (*Carla Introduction*, 2024).

A variety of advanced features are included in CARLA to enhance the simulation quality. It supports a server multi-client architecture which allows multiple clients to manage various actors at the same time, even across different nodes. It has a flexible API structure and enables users to control different simulation aspects, such as traffic generation, pedestrian behaviors, weather conditions, and sensor setups. Users are able to configure comprehensive sensor suites, including LIDARs, cameras, depth sensors, and GPS which are necessary for autonomous driving applications. Additionally, CARLA allows users to choose faster, non-rendered simulation modes for effective control and planning tasks. RoadRunner and other ASAM OpenDRIVE tools contribute to the creating custom maps. The integrated ScenarioRunner engine facilities the definition and execution of various scenarios using modular behavior models for traffic scenario simulation. Furthermore, CARLA provides ROS integration through a ROS-bridge and includes pre-configured

autonomous driving baselines, such as AutoWare and Conditional Imitation Learning agents for testing and development (*Carla Introduction*, 2024).

CARLA provides a realistic, robust and complex driving environments and it is preferred across autonomous driving, traffic safety, motion planning, and sensor research gathering various studies together in vehicle dynamics, perception, and control in environments (Dosovitskiy et al., 2017; Maierhofer & Althoff, n.d.; Malik et al., 2023). Simulation has high impacts and it is necessary for research and development of autonomous driving scenarios; since, many challenges and risks are involved in real-world testing. In physical environments, training and validation require vast resources and consider safety concerns (e.g., sudden pedestrian movements). Regarding these facts, CARLA addresses these issues by enabling a safe, scalable, and cost-effective platform for testing driving strategies and system responses in controlled and realistic environments. Therefore, without the logistical and financial constraints of real-world setup, CARLA gives opportunity to researchers to perform their experiments in a robust platform. (Dosovitskiy et al., 2017)

Given the research goal of exposing domain shifts in machine learning models, CARLA was selected due to its simulation ability across diverse environmental conditions. CARLA's robust environment and dataset generation abilities under controlled variations made our job easier and effective. A rural setting was set up utilizing the Town07 map, specifically to make room for future studies (*Town 7*, 2024). The specified route was traversed by the same vehicle under identical parameters by altering weather condition parameters. While the simulation was running, images were captured under nine different weather conditions, ranging from clear noon to rainy and wet configurations, reflecting environmental changes. An RGB sensor mounted on the vehicle and images were captured during each run. Later, these images were split into training, validation, and testing datasets to use in the autoencoder model. Then, to enhance the robustness of the dataset, data augmentation techniques such as rotations, width shift range, height shift range, shearing, zooming, horizontal flipping, and filling were applied.

2.2. Autoencoders and Domain-Invariant Representations

Autoencoders are such powerful structures for artificial neural network designs for unsupervised learning cases. They are formed by two main components which are encoder and decoder. Encoders are responsible for compressing data into a lower-dimensional latent representation and decoders reconstruct the original data from this latent representation. Autoencoders can be used for dimensionality reduction, data denoising, and feature learning. Primarily, for the image processing tasks autoencoders are highly preferred (Dave Bergmann & Cole Stryker, 2023; Pier Paolo Ippolito, 2023). The CIAE block diagram can be seen in Figure 1.

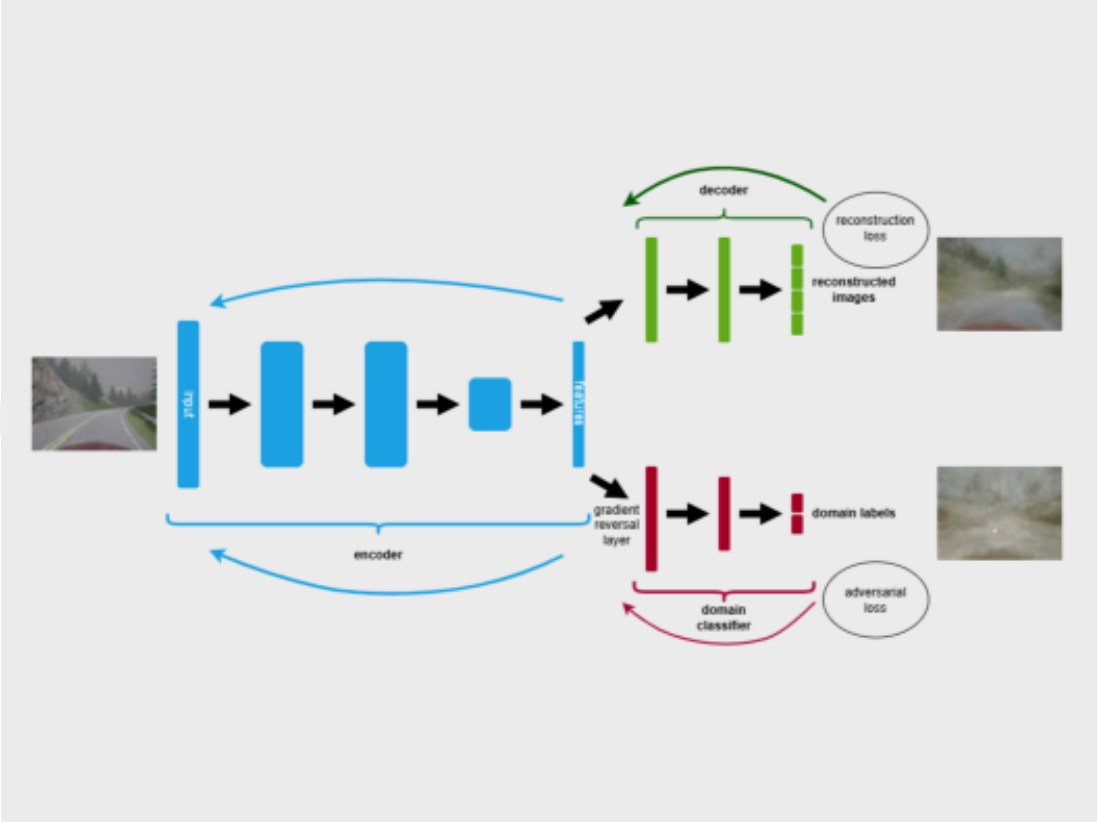

Figure 1: The block diagram of autoencoder and domain classifier architecture.

Compared to more traditional methods like Principal Component Analysis (PCA), which can capture nonlinear relationships within the data, autoencoders have significant advantages. While PCA is a linear technique that identifies the axes of maximum variance, autoencoders use non-linear activation functions and multi-layer architectures, enabling them to model more complex data distributions. Autoencoders are therefore an excellent option for some certain tasks which involve high-dimensional, nonlinear datasets, like images. Furthermore, with autoencoders custom optimization is possible such as using reconstruction loss specific to the data type and even enhancing the PCA flexibility (S. Chen & Guo, 2023; Fournier & Aloise, 2019).

Autoencoders are widely used in image models by applying denoising, image reconstruction, and feature extractions. Denoising autoencoders learn to identify noisy images, eliminating unwanted distortions from the data effectively (Pascal Vincent et al., 2010). Creating an image reconstruction that matches the original input data is a means to evaluate the quality of compressed data (Kingma & Welling, 2013). In a similar manner, while identifying notable features. feature extraction eliminates redundant information. Other issues, such as image classification or anomaly detection, are handled using this compressed form (Ardelean et al., 2023; Zhou et al., 2022).

In the literature, several types of autoencoders exist such as vanilla, variational (VAEs), denoising, sparse, convolutional autoencoders, etc. In this study, the main structure was established based on vanilla autoencoder which is known as the most basic one. Vanilla autoencoders are used for unsupervised learning as the basis of neural networks, aiming to learn from a compressed representation of input data which is encoder and reconstructs the original data from this representation which corresponds to decoder. For other specific tasks, autoencoders have evolved beyond their basic structure. These variants are highly important in terms of adaptability to diverse tasks, especially model generalization, feature extraction, and denoising. Each variant represents a different specific target and optimized performance targets.

To evaluate the models, three main evaluation metrics were used. During the training phase, two loss functions which are reconstruction loss and adversarial loss are calculated. During the training process, autoencoders focus on minimum loss function to reduce the reconstruction error. For continuous data MSE and SSIM are used but for binary data, binary cross-entropy loss is preferred. Moreover, optimizers like Adam are also necessary to adapt learning rates to improve convergence efficiency during backpropagation (Battikh & Lenskiy, 2021).

However, these advantages come at the expense of increased computing resources and overfitting, particularly when small training dataset or improper regularization techniques are used. Despite these facts, the ability of autoencoders to generalize and adapt to complex data are sufficient enough to make them a powerful alternative to PCA in machine learning applications (S. Chen & Guo, 2023; Fournier & Aloise, 2019).

In order to maintain the model's stability and consistency across dynamic environment and context, domain-invariant representations focus on the learned features. This helps the model to generalize fluctuations in environmental conditions, such as lighting, weather, or object variations.

This study incorporates domain-adversarial training using a gradient reversal layer to ensure domain invariance. By transferring the gradient during backpropagation, the encoder focuses on key features which are indistinguishable across different domains. By confusing domain classifier, the encoder is forced to focus on capturing the essential structure of the input data while disregarding domain-specific variations.

By doing so, the learned latent representations become robust to domain shifts, making them suitable for generalization across diverse and dynamic contexts. Unlike traditional approaches that might overfit to specific training conditions, this adversarial approach enhances the model's adaptability, allowing it to perform well in scenarios it has not encountered during training.

2.3. Domain Adaptation

In terms of domain adaptation, distribution shift occurs when there is a discrepancy between the data distribution of the source domain and the target domain. The main objective in domain adaptation is to reduce the impact of distribution shift to make the model perform well on the target domain. Unlike traditional transfer learning models, domain adaptation focuses one the scenarios with distributional mismatches while performance of target domain is minimized. This concept fits to autonomous driving applications, healthcare, and natural language processing (NLP), where the concerns of data availability or quality across domains are prevalent (Liu et al., 2022).

For supervised machine learning models, labeled data is crucial for its training steps. Domain adaptation is an option when gathering labeled data in the target domain is challenging. It might require huge time and cost where expertise in data annotation is necessary (Norrelykke & Flyvbjerg, 2009). Other than that, some cases address other considerations such as privacy laws, confidentiality agreements, or ethical concerns which limit access to labeled data in the target domain. Hence, these domains contain data ambiguity. To illustrate, in neural language applications sentiment analysis may be subjective. Thus, the data distribution changes rapidly in dynamic domains like social media, requiring frequent labeling (Ben-David et al., 2010; Zhuang et al., 2021).

Domain adaptation involves a source domain (D_S) with labeled data and a target domain (D_T) with limited or no labels. The goal is to bridge the performance gap caused by differences in covariate shift where input distribution varies but label distributions remain consistent; label shift where target class distribution differs; and concept shift where changes in input-label relationships (Singhal et al., 2023).

The availability of labeled data in the target domain identifies the domain adaptation strategies. In supervised domain adaptation, labeled data from the target domain is used in model fine tuning; since, they are initially trained on the source domain which provides computational efficiency with the need for target annotations constraint.

However, for unsupervised domain adaptation; the model lacks target domain labels. The model focuses on gathering key features and their distributions between domains, employing adversarial approaches which adapt neural networks to minimize domain discrepancies while keeping key features (Ganin et al., 2015). For such cases where the labeled data is limited in the target domain, semi-supervised domain adaptation techniques may be introduced to improve model generalization (Brückner et al., 2020).

Domain adaptation applies several key techniques to highlight distributional differences between source and target domains. Feature alignment is one common approach where integrates methods like Maximum Mean Discrepancy (MMD) and Correlation Alignment (CORAL) to align feature spaces of the two domains in a shared latent space which makes

feature distributions indistinguishable. Therefore, these methods reduce the statistical distance between domains and provide effective knowledge transfer (Saito et al., 2017).

Adversarial training is another critical technique, which preserves task-relevant information to make domain-specific features indistinguishable. For example, Ganin et al., introduces the Domain-Adversarial Neural Network (DANN), which aligns domain features using a domain classifier adversarial trained against the feature extractor (Ganin et al., 2015). This approach is particularly useful for unsupervised domain adaptation. Thanks to the adversarial training process, to gain robust performance in the target domain, the task-relevant information is preserved while domain-specific variations are suppressed.

Despite its several advantages, domain adaptation faces several key challenges. Due to large domain gaps between source and target domains, model transferability can be hindered, especially when the data distributions differ significantly, such as synthetic and real-world data (Kothandaraman & Nambiar, n.d.). In addition to that, the scalability of these methods to complex, real-world scenarios remain an issue, as models often struggle to adapt to dynamic and unpredictable environments (Hu et al., 2022). Finally, ethical concerns may arise when biases present in the source domain. This may lead to unfair or discriminatory outcomes (Hu et al., 2022; Kothandaraman & Nambiar, n.d.).

CHAPTER 3

RELATED WORK

This chapter focuses on the related work in the literature in terms of autoencoders in machine learning, domain adaptation in machine learning, domain-adversarial and feature learning in machine learning, and novelty of this work.

3.1. Autoencoders in Machine Learning

Autoencoders have essential impact in machine learning studies with its applications like dimensionality reduction including nonlinear dimensionality (Hinton & Salakhutdinov, 2006), feature extraction for clustering and classification, data reconstruction, and data denoising (Vincent et al., 2008).

Autoencoders have various types and each of them represents different goals to enhance model performance. To illustrate, denoising autoencoders focus on improving robustness by learning from noisy inputs to reconstruct clean data (Vincent et al., 2008). Therefore, variational autoencoders (VAEs) provide a probabilistic framework to model latent spaces for generative tasks (Kingma & Welling, 2013). Moreover, adversarial autoencoders (AAEs) enforce structured latent spaces by combining reconstruction loss and adversarial training (Makhzani et al., 2015). And convolutional autoencoders (CAEs) use convolutional layers to handle spatial data effectively, particularly in image-based tasks (Bengio et al., 2013).

Even though they have several utilities, traditional autoencoders handle some challenges in generalizing across domains, most of the time they fail to solve domain shift problems caused by weather conditions, lighting, or other environmental factors. For example, models trained on clear weather images might struggle with rainy or foggy weather scenarios. At this point, integrating domain adaptation techniques with autoencoders is an essential solution. As Ganin & Lempitsky proposed a solution in terms of domain-adversarial training to align feature spaces, while reconstruction loss with adversarial objectives to ensure robust, domain-invariant representations were combined (Ganin & Lempitsky, 2014).

(*The BrAIn Roads*, 2024)In addition to these studies, a group of researches of Computer Vision Centre has implemented AI for autonomous driving on rural roads. This project offers an economical and sustainable solution to the lack of public transport in rural and

isolated areas. The team faced with the complex challenge of training the AI on a narrow mountain road with poor visibility, adverse weather conditions, and no mobile connectivity (*The BrAIn Roads*, 2024).

To summarize, while the foundational autoencoders excel in representation learning, they have limitations under domain shifts need to be highlighted to enhance the models. Recent studies show that integration of adversarial training and domain adaptation techniques have significant impact in terms of context-invariant feature learning.

3.2. Domain Adaptation in Machine Learning

Domain adaptation handles the performance gap between source and target domains caused by distributional differences, concentrating on challenges such dynamic environmental conditions or visual impairments. Domain adaptation is highly preferred in autonomous driving to generalize models trained on structured urban settings to unstructured rural environments. To illustrate, a sim-to-real framework leveraged simulated data to train models that could generalize to-real-world scenarios. This approach emphasized transferring knowledge from structured urban settings to unstructured or challenging environments, mirroring the focus on rural adaptation (Hu et al., 2022).

In the literature, visual impairments addressing rainy and foggy weather conditions have been discussed using domain adaptation techniques. For instance, models that had been trained on urban datasets were refined using data from simulated adverse weather data, which improved object detection in challenging environments. This is consistent with our study's use of CARLA simulation to evaluate model robustness under varied weather conditions (Li et al., 2023).

Among the methods used in domain adaptation are adversarial training and statistical feature alignment. For example, unsupervised methods like Domain-Adversarial Neural Networks (DANN) use gradient reversal layers to identify key feature spaces across domains in unlabeled data. It retains task-specific information while guaranteeing domain invariance (Ganin et al., 2015). Complementary approaches like Maximum Mean Discrepancy (MMD) and Correlation Alignment (CORAL) reduce statistical distribution differences in shared latent spaces, further improving transferability between domains (Sun & Saenko, 2016; Tzeng et al., 2014). Adversarial and statistical methods are often combined with reconstruction-based objectives to enhance both robustness and feature relevance.

Despite significant advancements, some challenges such as large domain gaps, scalability to complex environments, and the potential transfer of biases from source to target domains arise. To solve these issues, hybrid techniques are required to balance adversarial and statistical alignment methods. In this study, combines adversarial training with

autoencoder to focus on context-invariant representation learning and create robust model performance across different weather conditions.

3.3. Domain-Adversarial and Feature Learning in Machine Learning

Domain-adversarial training is a key approach in machine learning that ensures domain-invariant feature learning by using adversarial objectives. Feature representations are indistinguishable across domains thanks to this method. It can also be combined with feature learning, making unsupervised domain adaptation advanced.

As mentioned in the previous sections, domain-adversarial uses gradient reversal layer to reverse the gradient direction during backpropagation. In this way, feature extractor learns representations, minimizing domain-specific information while keeping task-relevant features (Ganin et al., 2015). Hence, DANN provides better results in image classification, object detection, and NLP, providing a robust framework for unsupervised domain adaptation.

In addition to that, adversarial training can be extended to reconstruction tasks. It combines reconstruction loss with adversarial objectives to advance structured latent spaces (Makhzani et al., 2015).

In terms of feature learning, the main target is to extract task-relevant and compact representation from raw data. When it is integrated with domain-adversarial techniques, it becomes a great tool to mitigate domain shifts. As Tzeng et al. proposed Adversarial Discriminative Domain Adaptation (ADDA) framework, to separate encoder training for source and target domains and use adversarial loss to align their latent spaces (Tzeng et al., 2014). Hence, feature spaces are not only domain-invariant but also discriminative for downstream tasks.

For feature learning, two main feature alignment methods are used which are Maximum Mean Discrepancy (MMD) and Correlation Alignment (CORAL). These methods target to minimize distributional discrepancies between source and target domains, bringing an additional layer of feature alignment (Sun & Saenko, 2016).

Despite all these facts, combining domain-adversarial techniques with feature alignments requires high-dimensional or complex data distributions and the hyperparameter tuning shall be handled carefully to balance the adversarial loss with task-specific objectives (Goodfellow et al., 2014).

3.4. Novelty of This Work

This study introduces a novel framework which integrates domain-adversarial training with autoencoders to achieve context-invariant representation learning. Apart from traditional autoencoders that are not concerned with domain shifts, the proposed approach combines reconstruction objectives with adversarial training, providing a robust framework across dynamic environmental contexts. To ensure that the latent space remains context-invariant, a domain discriminator component is used alongside the decoder and encoder blocks of the autoencoder.

The novelty comes from the dual optimization strategy. While the model minimizes reconstruction loss to keep key features, adversarial loss aligns feature distributions across domains. Hence, the adaptability of autoencoders is enhanced and suitable for applications where the environment contains dynamic variables, such as weather and lighting changes.

While previous research has demonstrated the effectiveness of domain adaptation in urban driving this study diverges by addressing rural environments, which remain underexplored in the domain of autonomous driving (E. Chen et al., 2023; Yoo et al., 2017). Consequently, moving beyond static urban scenarios, our focus is to use CARLA-simulated data on dynamic weather conditions unique to rural environments. The datasets used for training in this study have been generated using CARLA in a diverse and realistic platform across different weather conditions and times of day. Moreover, evaluating the model's capability to choose weather-specific conditions that optimize accuracy, introducing a decision-making layer to adapt to environmental variations. This study bridges the gap in literature regarding domain-adversarial training and data augmentation strategies have been employed to extract features that capture the essential structures of input images while disregarding features associated with contextual changes.

CHAPTER 4

DATA GENERATION

This study utilizes the CARLA Simulator to generate artificial image data (Dosovitskiy et al., 2017). CARLA is an open-source simulator designed for autonomous driving research and it supports developments, training, and validation of autonomous driving systems. In addition to open-source code and protocols, CARLA provides open digital assets such as urban and rural maps, buildings, vehicles, sensors, etc. It also allows for customizable sensor configurations and gives control over static elements, dynamic actors, environmental conditions and more (Espadinha et al., 2021).

To be able to create a rural environment in CARLA 0.9.14 Release, Town 07 is used to test autonomous driving. It has a small and simple road network, a variety of farm buildings and wooden barns, set of junctions and unmarked roads with a couple of residential parking streets, some structures that are only found in rural areas, such as grain silos and windmills (*Town 7*, 2024). There is also a small body of water spanned by a short road bridge. Some of the roads are lined by corn fields Figure 2. To be able manage traffic in a controlled way, only one single car was used.



Figure 2: CARLA Town 07.

Sensors in CARLA are actors that retrieve data from their surroundings to create learning environments for driving agents (*Sensors and Data*, 2024). The sensor configuration consists of an RGB camera which is mounted on the car, 1.6m up from the ground and 0.9m forward. The images retrieved from the RGB camera (Figure 3) have 640 width, 480 height and 3 pixels sizes and the time is set as 1.0 seconds between each sensor captures with 90 degrees horizontal field of view.



Figure 3: CARLA RGB camera output image.

In CARLA, it is allowed to change weather parameters to obtain different weather conditions. The dataset consists of images captured under nine distinct weather conditions, divided into three main categories based on time of day and environmental factors.

- Clear Weather: Clear Noon, Clear Night, Clear Sunset.
- Hard Rain Weather: Hard Rain Noon, Hard Rain Night, Hard Rain Sunset.
- Wet Weather: Wet Noon, Wet Night, Wet Sunset.

The defined parameters for each weather condition are given in Table 1.

Table 1: CARLA Weather Parameters (Data sourced from https://carla.org/Doxygen/html/db/ddb/classcarla_1_1rpc_1_1WeatherParameters.html)

Public Attributes	**cloudiness = 0.0f, dust_storm = 0.0f, fog_density = 0.0f, fog_distance = 0.0f, fog_falloff = 0.0f, mie_scattering_scale = 0.0f, precipitation = 0.0f, precipitation_deposits = 0.0f, rayleigh_scattering_scale = 0.0331f, scattering_intensity = 0.0f, sun_altitude_angle = 0.0f, sun_azimuth_angle = 0.0f, wetness = 0.0f, wind_intensity = 0.0f**
Clear Night	5.0f, 0.0f, 0.0f, 10.0f, -1.0f, -90.0f, 60.0f, 75.0f, 1.0f, 0.0f, 1.0f, 0.03f, 0.0331f, 0.0f
Clear Noon	5.0f, 0.0f, 0.0f, 10.0f, -1.0f, 45.0f, 2.0f, 0.75f, 0.1f, 0.0f, 1.0f, 0.03f, 0.0331f, 0.0f
Clear Sunset	5.0f, 0.0f, 0.0f, 10.0f, -1.0f, 15.0f, 2.0f, 0.75f, 0.1f, 0.0f, 1.0f, 0.03f, 0.0331f, 0.0f
Hard Rain Night	100.0f, 100.0f, 90.0f, 100.0f, -1.0f, -90.0f, 100.0f, 0.75f, 0.1f, 100.0f, 1.0f, 0.03f, 0.0331f, 0.0f
Hard Rain Noon	100.0f, 100.0f, 90.0f, 100.0f, -1.0f, 45.0f, 7.0f, 0.75f, 0.1f, 0.0f, 1.0f, 0.03f, 0.0331f, 0.0f
Hard Rain Sunset	100.0f, 100.0f, 90.0f, 100.0f, -1.0f, 15.0f, 7.0f, 0.75f, 0.1f, 0.0f, 1.0f, 0.03f, 0.0331f, 0.0f
Wet Night	5.0f, 0.0f, 50.0f, 10.0f, -1.0f, -90.0f, 60.0f, 75.0f, 1.0f, 60.0f, 1.0f, 0.03f, 0.0331f, 0.0f
Wet Noon	5.0f, 0.0f, 50.0f, 10.0f, -1.0f, 45.0f, 3.0f, 0.75f, 0.1f, 0.0f, 1.0f, 0.03f, 0.0331f, 0.0f
Wet Sunset	5.0f, 0.0f, 50.0f, 10.0f, -1.0f, 15.0f, 2.0f, 0.75f, 0.1f, 0.0f, 1.0f, 0.03f, 0.0331f, 0.0f

The route was specified using path IDs available in CARLA and repeated across all weather conditions. A single complete route was defined in the simulation starting and ending at path 63, passing through all types of rural environment areas shown in Figure 4. During the simulation, the car's RGB sensor captured images every 1.0 second. This ensured that images from different weather conditions showcased nearly identical views, enabling direct comparison. Each image was resized to a uniform shape of (480, 640, 3) to ensure consistency during training and testing and they are normalized to the [0, 1] range.



Figure 4: CARLA simulator route.

During the simulation, sometimes the car was crashing unexpectedly. To avoid extending the simulation time, the route was optimized to minimize unnecessary delays, resulting in the capture of 200 images. However, as this dataset was insufficient for the training process, data augmentation was employed to enhance the model's training dataset. The augmented images were replaced with the original images in the batch and are returned along with the corresponding domain labels. Data augmentation techniques were applied within the dataset using Tensorflow's keras.preprocessing.image module. The following transformations are employed to generate augmented data:

- rotation_range=20: Random rotation of the image within a range of ±20 degrees.
- width_shift_range=0.1: Random horizontal shift of the image by up to 10% of its width.
- height_shift_range=0.1: Random vertical shift of the image by up to 10% of its height.
- shear_range=0.1: Shear transformations within a range of ±10 degrees.
- zoom_range=0.1: Random zoom in or out by up to 10%.
- horizontal_flip=True: Random horizontal flipping of images.
- fill_mode='nearest': Filling in missing pixels created by transformations using the nearest pixel values.

Key advantages of this method include:

- Dynamic Augmentation: During the training phase, augmented data was generated to prevent repetition and increase the diverse inputs.
- Memory Efficiency: The process was performed in batches to minimize large dataset storage in memory.
- Randomized Input: By shuffling the data, it was ensured that the models doesn't produce repetitive patterns.

The dataset was split into different combinations based on comparison:

- Training Data: For the optimization of the model parameters.
- Validation Data: For hyperparameter tuning and monitoring overfitting.
- Testing Data: Evaluating final results, visualization and checking model performance.

During dataset splitting, random states were used to keep them constant, by ensuring the same subset of images to use them across different model configuration experiments. Hence, comparisons were fair and consistent while evaluating different model configurations.

Since the training dataset were limited at the beginning, data augmentation was applied to the training data to enhance generalization and increase the diversity of the dataset. While data augmentation enriches the dataset, it also ensures that missing pixel areas caused by these transformations are filled using special techniques. In this way, the dataset is effectively expanded, making the model more robust to real-world applications and increasing its generalization capabilities.

Furthermore, during the training phase to differentiate the source domain and target domains, source mask technique was applied, giving labels to domains; 0 for source domain and 1 for target domains. In this way, the model can differentiate the images based on the domain labels. All (source and target) images are passed through the encoder and the domain classifier, but only source images are passed through the encoder and the decoder. Hence, the decoder uses only source domain images for reconstruction and the reconstruction loss is calculated regarding this dataset. In addition to that, dataset is shuffled by keeping the domain information and labels. Thus, to visualize and analyze the output results in a better way, the number of samples were taken randomly. Finally, to gather denoised images, a noise function has been used with Gaussian distribution and 0.5 noise factor. Both source and target data images have been processes before the training.

CHAPTER 5

CONTEXT-INVARIANT AUTOENCODER TRAINING

This chapter outlines the methodology used to train context-invariant autoencoders via unsupervised domain adaptation. The dataset was generated from the CARLA simulator across nine different weather conditions under different times of day. The model architecture consists of encoder, decoder, and domain classifier blocks. Therefore, to balance reconstruction quality and domain alignment, data augmentation techniques were implemented during the training phase to improve generalization, Moreover, hyperparameters such as latent dimension, batch size, train-test-validation dataset split distribution, and epoch size, were identified and evaluated on different model configurations to examine their impacts on model performance and generalization across unseen domains.

5.1. Modeling

5.1.1. Objectives of the Model

The main goal of this model is to establish context-invariant representation by using autoencoder and unsupervised domain adaptation together. By training an autoencoder that captures the key features of the dataset while eliminating environmental changes like weather conditions and times of day. Training the model capable of handling diverse weather conditions by learning domain-invariant features through adversarial training was targeted in this study. Also, compressing images into a meaningful latent space, retaining key features while ensuring computational efficiency were another targeted topic. By using data augmentation techniques, the model is enhanced to handle dynamic conditions and avoids overfitting. Further objective is to focus on the real-world applications, such as autonomous driving systems which require robust and reliable performance across dynamic environmental conditions. Finally, verifying the quality of latent representations by reconstructing the original data and evaluating its similarity to the input was crucial.

5.1.2. Model Architecture

The model includes two main parts: image autoencoder and domain classifier. The image autoencoder is composed of encoder and decoder components. Encoder is responsible for compressing high-dimensional input images (an image of size (480, 640, 3)) into a lower-dimensional representation, a latent space. Inside the encoder, there exist four

convolutional blocks, and each includes Conv2D, LeakyRELU, and MaxPooling2D. Conv2D (convolutional layers) extract spatial features from the input by applying filters that detect patterns like edges, textures, or shapes. LeakyRELU (activation function) introduces non-linearity to enable the model to learn complex patterns and with a slope of 0.3 it is better for gradient flow especially for dark images, and max-pooling layers for spatial down-sampling. The leaky variant allows a small gradient for negative inputs, which helps prevent dead neurons during training. MaxPooling2D (pooling layers) reduces the spatial dimensions of the data, making it computationally efficient while preserving important features. Pooling layers also add robustness to spatial variations in the input like slight shifts or rotations. After these, to flatten the output and map it to a latent space, dropout and regularization techniques are applied. The block diagram is given on Figure 1 in Section 2.2. Moreover, the structure of the model can be seen below on Figure 5. This model structure has been referenced from the paper of Ganin et al. which focuses on domain-adversarial training of neural networks (Ganin et al., 2015).

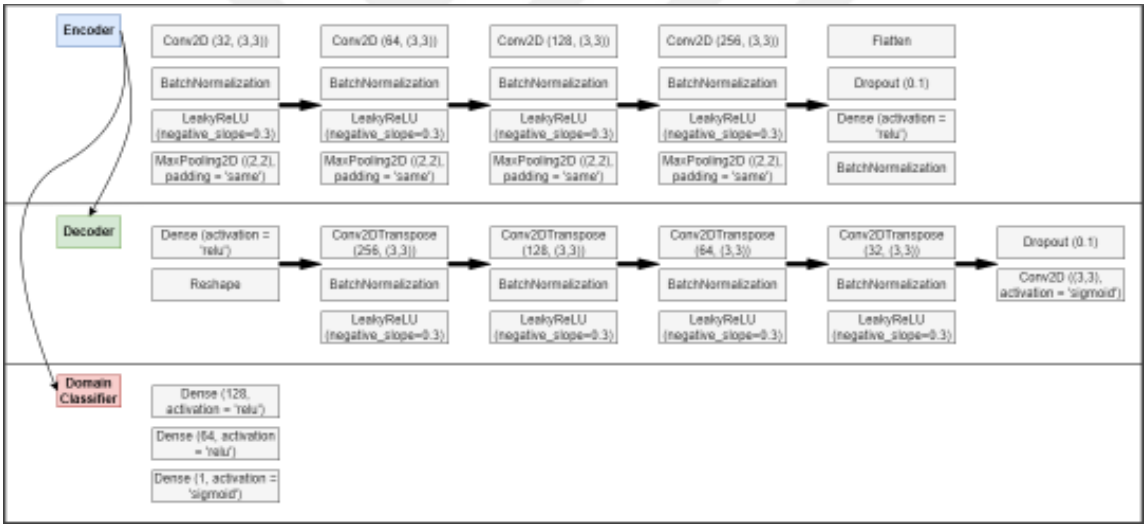

Figure 5: The CIAE model structure in terms of encoder, decoder, and domain classifier.

The main purpose of using latent space is to reduce the dimensionality of input data while capturing the most salient or meaningful features. The size of the latent space can be limited; so, the model starts to generalize better. This helps to minimize the impact of memorizing the input data which might lead to overfitting. The latent layers also provide a compact intermediate representation of the input which is used by the decoder to reconstruct the original data. Depending on the preserved information on the latent space, the quality of the reconstructed data can be gathered. In this study, the latent layer is used to detect the key domain-invariant features in the model across different weather conditions. Then this information was used in domain adaptation. After encoding, the decoder starts to reconstruct the latent space back into the original image dimensions. Like encoder, the decoder has also transposed convolutional layers which include Dense,

Reshape, and Conv2DTranspose layers. Dense (fully connected layer) expands the compressed latent representation into an intermediate representation that can be reshaped into a spatial format. Reshape layer converts the intermediate representation into a 3D tensor suitable for further processing by convolutional layers. Conv2DTranspose (transposed convolutional layers) refines the upsampled data and adds details to reconstruct the original image. It can be also thought of as a reverse convolution, adding pixels instead of reducing them. Output layer is the final layer that produces an image with the same dimensions and pixel value range as the original input. It also uses batch normalization and Leaky ReLU at each layer and the final layer produces reconstructed images with the same shape as the input images, using sigmoid activation function for pixel intensity normalization. A sigmoid activation is often used to ensure pixel values remain in the range [0,1].

During domain adaptation, forward propagation makes the features pass in a normal way. On the domain classifier part, the domain labels of the encoded latent representations are predicted whether source or target domains. However, with backward propagation, the gradient is reversed before passed back to the encoder. To do so, a gradient reversal layer is applied before the domain classifier to reverse gradients during backpropagation as referring to adversarial training. The final output is generated by using two fully connected Dense layers and sigmoid activation function for binary classification. Hence, the encoder is forced to produce domain-invariant features by confusing the domain classifier.

Briefly, the autoencoder learns to reconstruct inputs while compressing them into meaningful latent representations while the domain classifier ensures the latent representation is robust across multiple domains, enabling effective domain adaptation. During all phases only source domain datasets have been used.

5.1.3. Key Components

The key components in the model are encoder, decoder, and domain classifier which operate together for context-invariant feature learning. The encoder part aims to compress data into latent representations, confusing the domain classifier to make it challenging to identify the domain of the input. The decoder is responsible for reconstruction of the input data from the latent representations, preserving the main structure of the inputs the same. Moreover, the domain classifier works with a gradient reversal layer to facilitate adversarial training. It is targeted that encoders identify the domain-invariant features which are indistinguishable across different weather conditions. During forward propagation, the layer behaves like an identity operation, passing the input through without modification. During backward propagation, the gradients flowing back from the domain classifier are reversed (i.e., multiplied by -1) before reaching the encoder. Hence, while minimizing domain classification accuracy and reconstruction loss, domain-specific information from the latent representation is eliminated. Together with these techniques, data augmentation was also applied during training to expose the model to a diverse range

of transformation which improves generalization. Therefore, in the model data diversity exists due to different weather conditions and times of day. Each of them corresponds to a different domain. That is why the model is designed to handle this diversity to prevent overfitting and enable generalization. The dataset includes nine distinct weather conditions, grouped into three main categories based on time of day: Clear Night, Clear Noon, Clear Sunset, Hard Rain Night, Hard Rain Noon, Hard Rain Sunset, Wet Night, Wet Noon, and Wet Sunset. Each domain introduces unique variations in lighting, texture, and overall appearance (e.g., bright and sharp features during "Clear noon" vs. low light and blurry features during "Wet Night"). Training on "Clear noon" condition ensures that the encoder learns generalizable features that are applicable for "Clear noon". Testing on all weather conditions provides a comprehensive evaluation of the model's domain adaptation performance.

Images from all weather conditions are resized and preprocessed to have uniform dimensions of (480, 640, 3). Data augmentation techniques are applied to artificially increase diversity. Rotation (up to 10 degrees), translation (10% width/height), and horizontal flipping were the applied techniques in data augmentation. Balanced splits ensure that all nine weather conditions are well-represented in the training, validation, and test sets.

These components work together to enable the model to adapt effectively to different domains while maintaining high reconstruction quality and classification accuracy.

5.2. Learning

5.2.1. Training Process

The first step in the training phase is to feed the batches of images into the encoder to obtain latent representations which are required to be context- invariant. Images labeled which belong to the source domain - which is Clear Noon – transferred to the decoder with latent representation by using source mask technique. Then, the decoder tries to reconstruct the original images, aiming to minimize a combined loss from MSE and SSIM. Simultaneously, the latent representations are fed into the domain classifier, which are managed by gradient reversal layer. The encoder is adversarially trained to align the distributions of latent spaces across different domains. During this phase, the lambda parameter (λ) is dynamically adjusted to highlight the impact of adversarial loss during training and it balances reconstruction and domain invariance objectives. Moreover, in terms of data augmentation while learning rate is decreasing which helps to improve optimization stability by using Adam optimizer over epochs. This iterative process ensures the encoder learns representations that generalize across varying contexts, enabling consistent performance in diverse environments. Hence, the model becomes sufficient enough to confuse about the reconstructed images belonging to which domain. Binary

cross entropy loss has been used to train the domain classifier to predict the correct domain label.

5.2.2. Optimization and Adversarial Learning

The combination of the Adam optimizer and gradient reversal enables the model to effectively balance reconstruction accuracy (through the autoencoder) and domain robustness (through the adversarial training of the encoder). To achieve domain adaptation, the gradient reversal layer is used to reverse gradients from the domain classifier during backpropagation. This adversarial training mechanism forces the encoder to learn features that are not specific to any particular domain, ensuring that it can generalize well across varying weather conditions.

In the model, two different dynamic parameters are defined. Lambda (λ) scales the contribution of the adversarial loss during the training and it is adjusted per each epoch by using:

$$\lambda = \left(\frac{2}{1+exp\,(-10.p)} - 1\right)\, x\ 0.001 \tag{6}$$

where p is the normalized epoch index. It takes a small value close to 0 at the beginning of training to focus on learning key features for reconstruction and later it approaches its maximum value to make latent representations domain-invariant.

And the other dynamic parameter is learning rate which is decreasing progressively with the epoch using the formula:

$$\eta = \frac{\eta_0}{(1+10.p)^{0.75}} \tag{7}$$

In terms of optimization, Adam optimizers are used for these three main components with a dynamic learning rate. We used the Adam optimizer which includes an adaptive learning rate mechanism and it adjusts the effective learning rate for each parameter individually during training. Because of this adaptive nature, explicitly updating or scheduling the learning rate is often not strictly necessary. Nevertheless, while the training is ongoing, we decrease the learning rate to allow the model to fine-tune its parameters more carefully in later stages of training. As the learning rate gradually reduces, the model becomes stable and prevents oscillations around the optimal point.

5.2.3. Performance Metrics

The model includes a combination of performance metrics to evaluate how the model gives results in achieving both reconstruction and domain adaptation goals. Regarding reconstruction, MSE and SSIM were used as part of the combined loss function, reflecting how well the decoder can reconstruct input images while keeping structural and perceptual

details. MSE measures the accuracy of the autoencoder in reconstructing the input data. A lower reconstruction loss indicates a better representation of the original data. SSIM is a perceptual measure of image quality and evaluates the visual similarity between the original and reconstructed images. The equations are given below. Higher SSIM values indicate that the reconstruction maintains important structural features of the original image. For domain invariance, binary cross entropy loss was used to evaluate the domain classifier accuracy. Simultaneously, it evaluates the ability of the encoder indirectly to align latent space distributions across different domains. In addition to that, overall classification accuracy was calculated for domain prediction tasks to monitor adversarial training progress. All of these metrics provide that the model reaches a balance between maintaining high-quality image reconstructions and learning domain-invariant latent representations.

To obtain reconstruction loss, in the encoder part, nonlinear transitions are preferred. The main goal is to minimize the reconstruction loss, using such metrics like Mean-Squared-Error (MSE) and Structural Similarity Index Measure (SSIM) as a combined loss function for dimensionality reduction, feature extraction, and providing high-qualified reconstruction of source images. The reconstruction loss is calculated as;

$$L_{reconstruction} = (1 - \alpha).\, L_{MSE} + \alpha.\, L_{SSIM} \qquad (1)$$

- $L_{MSE} = -\frac{1}{N}\sum_{i=1}^{N} \|y_i - \hat{y_i}\|_2^2$: y_i represents the original image, and $\hat{y_i}$ is the reconstructed image.
- L_{SSIM} = 1 – SSIM (y, $\hat{y}$): SSIM measures the structural similarity between y and $\hat{y}$, with a maximum value of 1 and a minimum value of 0.
- α: A weighting factor between SSIM and MSE (e.g., $\alpha = 0.3$)

Therefore, binary cross-entropy loss is used for domain classification, encouraging domain invariance in the latent space. The adversarial loss is calculated as;

$$L_{adversarial} = \lambda.E\left[-\frac{1}{N}\sum_{i=1}^{N}[d_i \cdot log(\widehat{d_i}) + (1 - d_i) \cdot log(1 - \widehat{d_i})]\right] \qquad (2)$$

where:

- *N:* The number of samples in the batch.
- d_i: The ground truth domain label for the ith sample (0 for source domain, 1 for target domain).
- $\widehat{d_i}$: The predicted domain probability for the ith sample, output of the domain classifier.

- λ: A scaling factor controlling the weight of the adversarial loss (e.g., $\lambda = 0.1$)

In addition to reconstruction and adversarial loss explained above, accuracy is another metric which is used in the model to measure the binary classification of domain labels.

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions} \quad (3)$$

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N}(d_i = \widehat{d_i}) \quad (4)$$

where:

- N: The number of samples in the batch.
- $\widehat{p_i}$: The predicted probability output by the domain classifier.
- d_i: The ground truth domain label for the ith sample (0 for source domain, 1 for target domain).
- $\widehat{d_i}$: The predicted domain label for the ith sample, typically derived as:

$$\widehat{d_i} = \{1, if\ \widehat{p_i} > 0.5;\ 0, if\ \widehat{p_i} \leq 0.5 \quad (5)$$

(Kothandaraman & Nambiar, 2021) In the literature, MSE and SSIM are highly preferred two performance metrics. For the purpose of semantic segmentation of near-infrared images using unsupervised domain adaptation, have employed SSIM loss to ensure that the model maintains structural consistency between the source and target domains. In order to preserve the fine structural details in the segmented images while enhancing the quality of transferred features, SSIM is incorporated into loss function in that research. On the other hand, another research suggests a domain-adaptive knowledge distillation method to address the issues of memory limitations and domain gaps in autonomous driving systems. In that study, MSE loss has been used to align feature representations between teacher and student networks, ensuring accurate knowledge transfer and improved segmentation performance. Hence, transferring knowledge from a complex teacher network to a compact student network for effective semantic segmentation is possible (Kothandaraman & Nambiar, 2021). Moreover, another paper explores few-shot domain adaptation for enhancing low-light RAW images, where the goal is to transfer knowledge from a well-lit source domain to a poorly-lit target domain using minimal data. In this study, MSE is used to minimize pixel-wise reconstruction errors between enhanced and ground truth images. In parallel, SSIM loss is incorporated to preserve perceptual quality by ensuring structural similarity between the enhanced and original images. In this way, the combination of these losses helps improve generalization across different lighting conditions (Ram Prabhakar et al., 2021).

5.3. Experiments

To perform the experiments, the key hyperparameters were identified based on the impact on the model. As a base model, the combination of hyperparameters was decided. Moreover, the models were run based on the specified configurations and the model results were obtained to perform comparisons.

5.3.1. Implementation Details

To investigate which hyperparameters behave in a way that improves the model performance and which ones impact the model in a negative way, the key hyperparameters have been identified on Table 2. The base model has been created with these parameters as described in Table 3. The model runs have been grouped based on the number of epochs which are identified as 100, 500 and 1000. And for each type of hyperparameter, three models have been built to investigate the impact of the hyperparameters based on epoch numbers and they can be examined on Table 4.

For latent dimension, some adjustments have been made in the model such as adapting Conv2D layers, Dense layers, and changing latent dimension inside encoder, decoder, and domain classifier parts. Then, batch sizes have been identified as 8, 16, and 32. Train-test-validation dataset split distributions were specified as (60, 20, 20), (70, 15, 15), and (80, 10, 10) respectively. Furthermore, dropout rate in encoder and decoder was selected as 0.1, 0.3, and 0.5. Each of them was run with 100, 500, and 1000 epochs.

Table 2: Key Hyperparameters and Their Impact on Autoencoder Performance. This table comprises important hyperparameters and their impact on model performance.

Hyperparameter	**Description**	**Impact on Performance**
Latent dimension	The size of the compressed representation in the latent space of the autoencoder.	A larger latent dimension allows the model to capture more information but may lead to overfitting. A smaller latent dimension helps avoid overfitting but may reduce the model's capacity to learn complex features.

Table 2 cont.

Batch size	Batch size refers to the number of samples used in one update of the model's weights.	A smaller batch size may provide noisier updates but can help escape local minima, while a larger batch size provides more stable updates but requires more memory.
Train-test-validation dataset split distribution	The proportion of data allocated for training, testing, and validation sets.	Proper distribution ensures the model generalizes well to unseen data. A poor split may lead to overfitting (if too little data is used for validation/testing) or underfitting (if insufficient data is available for training).
Dropout rate	Batch normalization normalizes the input to each layer to improve stability during training. Dropout randomly disables neurons during training to prevent overfitting.	Batch normalization accelerates training and stabilizes learning by reducing internal covariate shift. Dropout helps prevent overfitting by ensuring the model doesn't rely too heavily on any neuron.
Epoch	Epoch refers to one complete pass through the entire training dataset.	Increasing epochs generally leads to better training but may also increase the risk of overfitting if not balanced.

Table 3: The Hyperparameters of The Base Model. This table shows the hyperparameter values used in the base model.

Hyperparameter	**Value**

Table 3 cont.

Latent dimension	256
Batch size	32
Train-Test-Validation dataset split distribution	70-15-15
Dropout rate	0.1
Epoch	100

Table 4: Model Runs Based on Domain Adversarial Autoencoder Architecture. This table shows the model runs assigned for domain adversarial autoencoder architecture.

Hyperparameter	**Epochs = 100**	**Epochs = 500**	**Epochs = 1000**
Latent dimension = 128	model_ld_128_100	model_ld_128_500	model_ld_128_1000
Latent dimension = 256	model_ld_256_100	model_ld_256_500	model_ld_256_1000
Latent dimension = 512	model_ld_512_100	model_ld_512_500	model_ld_512_1000
Batch size = 8	model_bs_8_100	model_bs_8_500	model_bs_8_1000
Batch size = 16	model_bs_16_100	model_bs_16_500	model_bs_16_1000
Batch size = 32	model_bs_32_100	model_bs_32_500	model_bs_32_1000
Train-test-validation split = (60, 20, 20)	model_ttv_602020_100	model_ttv_602020_500	model_ttv_602020_1000

Table 4 cont.

Train-test-validation split = (70, 15, 15)	model_ttv_701515_100	model_ttv_701515_500	model_ttv_701515_1000
Train-test-validation split = (80, 10, 10)	model_ttv_801010_100	model_ttv_801010_500	model_ttv_801010_1000
Dropout rate = 0.1	model_dr_01_100	model_dr_01_500	model_dr_01_1000
Dropout rate = 0.3	model_dr_03_100	model_dr_03_500	model_dr_03_1000
Dropout rate = 0.5	model_dr_05_100	model_dr_05_500	model_dr_05_1000

To present and compare the model results, the models have also been run on source only autoencoder architecture (SOAE) for 100, 500, and 1000 epochs. In these models, the autoencoder is only trained with source domain (Clear Noon) and tested with target domains (Others). The details are provided in Table 5.

Table 5: Model Runs Based on Source Only Autoencoder Architecture. This table shows the model runs assigned for source only autoencoder architecture.

Hyperparameter	**Epochs = 100**	**Epochs = 500**	**Epochs = 1000**
Latent dimension = 128	model_ld_128_100_soae	model_ld_128_500_soae	model_ld_128_1000_soae
Latent dimension = 256	model_ld_256_100_soae	model_ld_256_500_soae	model_ld_256_1000_soae
Latent dimension = 512	model_ld_512_100_soae	model_ld_512_500_soae	model_ld_512_1000_soae
Batch size = 8	model_bs_8_100_soae	model_bs_8_500_soae	model_bs_8_1000_soae

Table 5 cont.

Batch size = 16	model_bs_16_100_soae	model_bs_16_500_soae	model_bs_16_1000_soae
Batch size = 32	model_bs_32_100_soae	model_bs_32_500_soae	model_bs_32_1000_soae
Train-test-validation split = (60, 20, 20)	model_ttv_602020_100_soae	model_ttv_602020_500_soae	model_ttv_602020_1000_soae
Train-test-validation split = (70, 15, 15)	model_ttv_701515_100_soae	model_ttv_701515_500_soae	model_ttv_701515_1000_soae
Train-test-validation split = (80, 10, 10)	model_ttv_801010_100_soae	model_ttv_801010_500_soae	model_ttv_801010_1000_soae
Dropout rate = 0.1	model_dr_01_100_soae	model_dr_01_500_soae	model_dr_01_1000_soae
Dropout rate = 0.3	model_dr_03_100_soae	model_dr_03_500_soae	model_dr_03_1000_soae
Dropout rate = 0.5	model_dr_05_100_soae	model_dr_05_500_soae	model_dr_05_1000_soae

When considering real-world cases, the images which are captured would not be smooth in reality. That is why additional experiments have been performed by adding noise function to the CARLA simulator images. These experiments have only been run 1000 epochs for CIAE models. The model runs can be examined in Table 6.

Table 6: Model Runs Based on Domain Adversarial Autoencoder Architecture with Noise Function. This table shows the model runs assigned for domain adversarial autoencoder architecture by adding noise function.

Hyperparameter	**Epochs = 1000**

Table 6 cont.

Latent dimension = 128	model_denoising_ld_128_1000_soae
Latent dimension = 256	model_ denoising_ld_256_1000_soae
Latent dimension = 512	model_ denoising_ld_512_1000_soae
Batch size = 8	model_ denoising_bs_8_1000_soae
Batch size = 16	model_ denoising_bs_16_1000_soae
Batch size = 32	model_ denoising_bs_32_1000_soae
Train-test-validation split = (60, 20, 20)	model_ denoising_ttv_602020_1000_soae
Train-test-validation split = (70, 15, 15)	model_ denoising_ttv_701515_1000_soae
Train-test-validation split = (80, 10, 10)	model_ denoising_ttv_801010_1000_soae
Dropout rate = 0.1	model_ denoising_dr_01_1000_soae
Dropout rate = 0.3	model_ denoising_dr_03_1000_soae
Dropout rate = 0.5	model_ denoising_dr_05_1000_soae

5.3.2. Tools and Frameworks

The models have been implemented using different types of tools and frameworks. They allow for a modular, efficient, and reproducible implementation of the domain-adversarial autoencoder. TensorFlow's flexibility has been combined with other tools like WandB for

tracking model results, OpenCV for visualization. For the detailed explanations Table 7 can be examined.

Table 7: Tools and Frameworks Used in Models. This table explains which tools and frameworks are used with which purpose and their key features.

Tool/Framework	**Purpose**	**Key Features Utilized**
Python	Primary programming language for implementation and experimentation.	Versatility and integration with machine learning libraries.
TensorFlow	Framework for building and training the domain-adversarial neural network (DANN).	- Keras API for modular design - Mixed precision training - XLA JIT Compiler for speed optimization.
WandB (Weights & Biases)	Experiment tracking, logging, and visualization.	- Real-time metrics logging - Model comparisons - Visualization of outputs and metrics.
NumPy	Numerical operations and data manipulation.	Dataset shuffling, loss calculations, and pre-processing operations.

Table 7 cont.

OpenCV	Image handling and visualization, particularly during evaluation.	Saving high-resolution reconstructed images and converting RGB to BGR for storage.
Matplotlib	Visualizing model outputs and evaluation metrics.	Plotting original and reconstructed images for qualitative analysis.
Scikit-Learn	Dataset splitting into training, validation, and test sets.	Stratified and randomized splits to ensure unbiased distribution.
CARLA Simulator	Source of the dataset used for training and evaluation.	Provided diverse environmental conditions for domain adaptation tasks.
ImageDataGenerator	On-the-fly data augmentation for training data.	Augmentation techniques such as rotation, flipping, zooming, and shearing.
Custom Gradient Reversal Layer	Enables adversarial training by reversing gradients in the backpropagation step.	Implements domain alignment in the latent space.

Table 7 cont.

CUDA and NVIDIA GPUs	Hardware acceleration for training and evaluation.	Significantly reduced training time for high-resolution images and large datasets. NVIDIA-SMI 555.42.06 Driver Version: 555.42.06 CUDA Version: 12.5 4 x NVIDIA A100 80GB PCIe

5.4. Model Results and Analysis

To evaluate the models, several configurations have been applied to the model to change the hyperparameters like latent dimension, batch size, train-test-validation dataset split distribution, dropout rates, and epoch size. Based on the results, the models reacted to the changes in different manners. Each hyperparameter were run according to different epoch sizes, which are 100, 500, and 1000. The model results were grouped into three main categories to examine them in detail.

To compare the model results on CIAE models, the experiments have also been run for SOAE models. Based on CIAE results, the reconstruction loss has been calculated as a combined loss which includes MSE and SSIM values as described in Section 2.2. The CIAE and SOAE model results have been shown in APPENDIX A for 100, in APPENDIX B for 500 epochs, and in APPENDIX C for 1000 epochs based on these criteria:

- CIAE Train Reconstruction Loss, MSE Loss, and SSIM Loss Results,
- SOAE Train Reconstruction Loss, MSE Loss, and SSIM Loss Results,
- CIAE Validation Reconstruction Loss, MSE Loss, and SSIM Loss Results,
- SOAE Validation Reconstruction Loss, MSE Loss, and SSIM Loss Results,
- CIAE Test Reconstruction Loss, MSE Loss, and SSIM Loss Results,
- SOAE Test Reconstruction Loss, MSE Loss, and SSIM Loss Results,
- CIAE Train-Validation-Test Adversarial Loss Results,
- CIAE Validation - Test Accuracy Results,
- CIAE Lambda and Learning Rate Values,

- SOAE Learning Rate Values.

5.4.1. Reconstruction Results

According to the CIAE and SOAE model results, the following outcomes and conclusions can be drawn regarding the reconstruction loss which includes MSE loss and SSIM loss across train, validation, and test phases for 100, 500, and 1000 epochs.

After 100 epochs; both CIAE and SOAE give low MSE losses across train and validation phases. CIAE models slightly outperform SOAE in test MSE values; this is primarily due to CIAE's powerful generalization ability and better structural preservation. SSIM loss values are consistently lower in CIAE across all phases, suggesting better preservation of structural details. While CIAE models demonstrate a better balance between MSE and SSIM contributions, enabling superior reconstruction performance overall; SOAE models tend to rely more heavily on MSE for reducing reconstruction loss but lag behind in structural detail preservation (higher SSIM losses). The optimal configurations can be identified for CIAE as small batch sizes, balanced latent dimensions, and tuned learning rates, reflecting low MSE and SSIM losses. Moreover, SOAE models are competitive in training and they face challenges in generalization, especially in maintaining structural similarity (high SSIM losses on validation and test data.) To improve SOAE results we can focus on regularization techniques, better SSIM optimization, and hyperparameter tuning to enhance its structural reconstruction capabilities.

For 500 epochs, CIAE achieves slightly better training reconstruction losses overall. They are more consistent across configurations, while SOAE models exhibit higher variability due to hyperparameters like dropout and batch size. In terms of validation phases, CIAE outperforms SOAE in reconstruction loss which indicates that CIAE models generalize better to the validation set. CIAE also demonstrates significant advantages in test reconstruction loss. CIAE's lower test reconstruction loss highlights its superior robustness and generalization. Regarding MSE and SSIM losses, CIAE models achieve lower MSE across training, validation, and test datasets. Moreover, lower MSE in CIAE models reflect their robustness and ability to capture fine details effectively. SOAE models generally achieve better SSIM scores, especially in test datasets. Higher SSIM in SOAE indicates its strength in maintaining the overall structure and visual similarity of reconstructed images.

CIAE performs more consistently across various setups and yields marginally better training reconstruction losses overall for 1000 epochs. SOAE models exhibit higher variability due to hyperparameters like dropout and batch size. The training loss curves show that both models continue improving with more epochs, but CIAE maintains a steadier convergence pattern. CIAE outperforms SOAE in validation reconstruction loss, indicating better generalization to unseen validation data. The SOAE models show higher

reconstruction loss variance, likely due to their sensitivity to architectural and training hyperparameters. The gap in MSE loss also suggests that CIAE models are more effective at minimizing pixel-wise reconstruction errors. CIAE demonstrates a clear advantage in test reconstruction loss, reaffirming its robustness and ability to generalize beyond the training dataset. Lower MSE values in CIAE across test datasets further support its superior reconstruction accuracy. Meanwhile, SOAE models achieve higher SSIM scores, particularly in test results, which suggests that SOAE excels in preserving overall structural and perceptual image quality. CIAE models achieve lower MSE across training, validation, and test phases, confirming their effectiveness in precise reconstruction. The lower MSE in CIAE models reflects their robustness and ability to capture fine details. However, SOAE models tend to have better SSIM scores, especially in test results, which means they retain structural integrity and perceptual similarity better than CIAE.

5.4.1.1.Training Performance

- CIAE Results (# of Epochs = 100):

    - In terms of training reconstruction loss, the best-performing CIAE configuration is "model_ttv_602020_100" with a loss value of 0.0868, which includes MSE Loss = 0.0125 and SSIM Loss= 0.2602.

    - Other top configurations are examined as "model_ld_512_100" (Reconstruction Loss = 0.0878, MSE Loss= 0.0126, SSIM Loss= 0.2631).

    - The worst CIAE reconstruction loss performance belongs to "model_dr_05_100" (Reconstruction Loss = 0.0961, MSE Loss= 0.0159, SSIM Loss= 0.2832).

    - MSE loss values in CIAE are consistently low (mostly below 0.014), indicating excellent pixel-level accuracy, while SSIM loss values are slightly higher, emphasizing a focus on preserving visual structure.

- SOAE Results (# of Epochs = 100):

    - In terms of training reconstruction loss, the best-performing SOAE configuration is "model_ld_512_100_soae" with a loss value of 0.0870, comprising MSE Loss = 0.0130 and SSIM Loss= 0.2640.

    - The other powerful configuration "model_bs_16_100_soae" also demonstrates the importance of optimizing batch size (Reconstruction = 0.0874, MSE = 0.0131, SSIM = 0.2652).

- The worst SOAE reconstruction loss performance belongs to "model_dr_05_100_soae" (Reconstruction Loss = 0.1009, MSE = 0.0195, SSIM = 0.2908).

- SOAE training MSE values are close to those of CIAE, suggesting similar pixel accuracy. However, the SSIM losses are slightly higher, implying that SOAE is marginally less effective in retaining structural details.

Table 8: CIAE Train Reconstruction Loss Results (epoch = 100). This table shows the CIAE training results regarding reconstruction loss which is combined with MSE loss and SSIM loss.

Name	train/reconstruction_loss	train/mse_loss	train/ssim_loss
model_bs_08_100	0.0901	0.0145	0.2664
model_bs_16_100	0.0903	0.0139	0.2686
model_bs_32_100	0.0884	0.0136	0.2629
model_dr_01_100	0.0882	0.0128	0.2641
model_dr_03_100	0.0943	0.0159	0.2771
model_dr_05_100	0.0961	0.0159	0.2832
model_ld_128_100	0.0909	0.0149	0.268
model_ld_256_100	0.0886	0.0133	0.2643
model_ld_512_100	0.0878	0.0126	0.2631
model_ttv_602020_100	0.0868	0.0125	0.2602
model_ttv_701515_100	0.0916	0.0154	0.2694
model_ttv_801010_100	0.0892	0.0134	0.266

Table 9: SOAE Train Reconstruction Loss Results (epoch = 100). This table shows the SOAE training results regarding reconstruction loss which is combined with MSE loss and SSIM loss.

Name	train/reconstruction_loss	train/mse_loss	train/ssim_loss
model_bs_08_100_soae	0.0883	0.0129	0.264
model_bs_16_100_soae	0.0874	0.0119	0.2636
model_bs_32_100_soae	0.0882	0.0119	0.266
model_dr_01_100_soae	0.0891	0.0128	0.2673
model_dr_03_100_soae	0.0951	0.0155	0.2808
model_dr_05_100_soae	0.1009	0.0195	0.2908
model_ld_128_100_soae	0.0907	0.0138	0.2703
model_ld_256_100_soae	0.0882	0.0122	0.2656
model_ld_512_100_soae	0.087	0.0116	0.2628
model_ttv_602020_100_soae	0.0886	0.0126	0.266
model_ttv_701515_100_soae	0.0893	0.0125	0.2686

Table 9 cont.

model_ttv_801010_100_soae	0.0897	0.0125	0.2699

- CIAE Results (# of Epochs = 500):
    - The best CIAE train model result belongs to "model_ld_256_500" with reconstruction loss at 0.0793 (MSE loss = 0.0076, SSIM loss = 0.2467).
    - As a second best CIAE train model, "model_bs_16_500" also gives good results (Reconstruction loss = 0.0794, MSE loss = 0.0077, SSIM loss = 0.2468).
    - The highest train reconstruction loss is seen in "model_dr_05_500" at 0.0917, with an MSE of 0.0129 and SSIM of 0.2754.
    - For the train phase, lower latent dimensions and minimal dropout rates tend to lower reconstruction loss while maintaining competitive SSIM values.
- SOAE Results (# of Epochs = 500):
    - The best SOAE train model is seen as "model_ld_512_500_soae" with the lowest training reconstruction loss at 0.0804, with MSE 0.0080 and SSIM 0.2493.
    - Similarly, another best SOAE train model is "model_bs_32_500_soae" with 0.0807 reconstruction loss, 0.0082 MSE loss, and 0.2501 SSIM loss.
    - "model_dr_05_500_soae" gives the worst train result (Reconstruction loss = 0.0910, MSE loss = 0.0120, SSIM loss = 0.2754).
    - For the training phase, larger latent dimensions improve training MSE but may slightly reduce SSIM.

Table 10: CIAE Train Reconstruction Loss Results (epoch = 500). This table shows the CIAE training results regarding reconstruction loss which is combined with MSE loss and SSIM loss.

Name	train/reconstruction_loss	train/mse_loss	train/ssim_loss
model_bs_08_500	0.0819	0.0086	0.2531
model_bs_16_500	0.0794	0.0077	0.2468
model_bs_32_500	0.0802	0.0083	0.248

Table 10 cont.

model_dr_01_500	0.0814	0.0087	0.2511
model_dr_03_500	0.0838	0.0093	0.2576
model_dr_05_500	0.0917	0.0129	0.2754
model_ld_128_500	0.0819	0.0089	0.2523
model_ld_256_500	0.0793	0.0076	0.2467
model_ld_512_500	0.0808	0.0088	0.2486
model_ttv_602020_500	0.0803	0.0086	0.2479
model_ttv_701515_500	0.0804	0.0085	0.2482
model_ttv_801010_500	0.0807	0.0081	0.2501

Table 11: SOAE Train Reconstruction Loss Results (epoch = 500). This table shows the SOAE training results regarding reconstruction loss which is combined with MSE loss and SSIM loss.

Name	**train/reconstruction_loss**	**train/mse_loss**	**train/ssim_loss**
model_bs_08_500_soae	0.0823	0.0088	0.2539
model_bs_16_500_soae	0.0808	0.008	0.2506
model_bs_32_500_soae	0.0807	0.0081	0.2501
model_dr_01_500_soae	0.0821	0.0085	0.2538
model_dr_03_500_soae	0.0881	0.0114	0.2669
model_dr_05_500_soae	0.091	0.012	0.2754
model_ld_256_500_soae	0.0816	0.0084	0.2524
model_ld_512_500_soae	0.0804	0.0081	0.2492
model_ls_128_500_soae	0.0834	0.0097	0.2552
model_ttv_602020_500_soae	0.0825	0.0096	0.2525
model_ttv_701515_500_soae	0.0814	0.0087	0.2509
model_ttv_801010_500_soae	0.0809	0.0077	0.2519

- CIAE Results (# of Epochs = 1000):

  - The best CIAE train model result belongs to “model_ld_512_1000” with reconstruction loss at 0.0762 (MSE loss = 0.0067, SSIM loss = 0.2382).

  - As a second best CIAE train model, “model_ttv_602020_1000” also gives good results (Reconstruction loss = 0.0766, MSE loss = 0.0066, SSIM loss = 0.2400).

  - The highest train reconstruction loss is seen in “model_dr_05_1000” at 0.0861, with an MSE of 0.0097 and SSIM of 0.2643.

- For the train phase, higher dropout rates effect performance in a negative way while larger latent dimensions improve performance.

- SOAE Results (# of Epochs = 1000):

  - The best SOAE train model result is observed in "model_ld_512_1000_soae", which achieves the lowest reconstruction loss at 0.0776, with the lowest MSE loss of 0.0066 and the lowest SSIM loss of 0.2433. This suggests that increasing the latent dimension to 512 significantly improves reconstruction quality.

  - The second-best performing model is "model_bs_32_1000_soae", with a reconstruction loss of 0.0782, MSE loss of 0.0072, and SSIM loss of 0.2438—very close to the best performer. This indicates that a batch size of 32 is a sweet spot among the tested sizes.

  - The highest train reconstruction loss is recorded for "model_dr_05_1000_soae", at 0.0901, with an MSE of 0.0113 and SSIM of 0.2740, indicating that a dropout rate of 0.5 significantly degrades model performance.

  - There is a clear trend that increasing the dropout rate results in higher reconstruction.

Table 12: CIAE Train Reconstruction Loss Results (epoch = 1000). This table shows the CIAE training results regarding reconstruction loss which is combined with MSE loss and SSIM loss.

Name	train/reconstruction_loss	train/mse_loss	train/ssim_loss
model_bs_08_1000	0.0778	0.0073	0.2424
model_bs_16_1000	0.0772	0.0072	0.2405
model_bs_32_1000	0.0781	0.0073	0.2436
model_dr_01_1000	0.0775	0.007	0.2419
model_dr_03_1000	0.0809	0.0083	0.2504
model_dr_05_1000	0.0861	0.0097	0.2643
model_ld_128_1000	0.0796	0.0079	0.2468
model_ld_256_1000	0.0772	0.0072	0.2406
model_ld_512_1000	0.0762	0.0067	0.2382
model_ttv_602020_1000	0.0766	0.0066	0.24
model_ttv_701515_1000	0.0772	0.007	0.241
model_ttv_801010_1000	0.0781	0.0074	0.2429

Table 13: SOAE Train Reconstruction Loss Results (epoch = 1000). This table shows the SOAE training results regarding reconstruction loss which is combined with MSE loss and SSIM loss.

Name	train/reconstruction_loss	train/mse_loss	train/ssim_loss
model_bs_08_1000_soae	0.0807	0.0081	0.2501
model_bs_16_1000_soae	0.0818	0.0083	0.2533
model_bs_32_1000_soae	0.0782	0.0072	0.2438
model_dr_01_1000_soae	0.0812	0.0083	0.2513
model_dr_03_1000_soae	0.0861	0.0101	0.2634
model_dr_05_1000_soae	0.0901	0.0113	0.274
model_ld_128_1000_soae	0.0825	0.0092	0.2535
model_ld_256_1000_soae	0.0814	0.0086	0.2512
model_ld_512_1000_soae	0.0776	0.0066	0.2433
model_ttv_602020_1000_soae	0.0784	0.0075	0.2437
model_ttv_701515_1000_soae	0.0808	0.0083	0.25
model_ttv_801010_1000_soae	0.0802	0.0075	0.2499

5.4.1.2.Validation Performance

- CIAE Results (# of Epochs = 100):

    - The best validation reconstruction loss is achieved by "model_bs_08_100" at 0.0647, with MSE Loss = 0.0082 and SSIM Loss = 0.1967.

    - The next best performance belongs to "model_bs_16_100" which reflects the importance of smaller batch sizes for better generalization (Reconstruction = 0.0775, MSE = 0.0099, SSIM = 0.2353).

    - The worst validation result in CIAE is from "model_dr_05_100" with 0.0926 reconstruction loss, including MSE loss = 0.0147 and SSIM loss = 0.2741.

    - CIAE validation results show a strong balance between MSE and SSIM, which translates to high reconstruction accuracy on unseen data.

- SOAE Results (# of Epochs = 100):

- The best SOAE configuration on the validation set is "model_bs_08_100_soae" with a reconstruction loss of 0.0883, comprising MSE = 0.0095 and SSIM = 0.2721. While the pixel accuracy (MSE) is competitive, the SSIM loss is higher than that of CIAE, indicating that SOAE struggles to maintain structural similarity as effectively.

- Higher dropout rates in models such as "model_ttv_602020_100_soae" indicate worse performance on validation reconstruction loss due to higher MSE and SSIM loss values.

- Overall, SOAE models display higher SSIM loss values (~0.26-0.30), indicating a gap in structural preservation compared to CIAE.

Table 14: CIAE Validation Reconstruction Loss Results (epoch = 100). This table shows the CIAE validation results regarding reconstruction loss which is combined with MSE loss and SSIM loss.

Name	validation/reconstruction_loss	validation/mse_loss	validation/ssim_loss
model_bs_08_100	0.0647	0.0082	0.1967
model_bs_16_100	0.0775	0.0099	0.2353
model_bs_32_100	0.0895	0.0117	0.2708
model_dr_01_100	0.0891	0.0114	0.2703
model_dr_03_100	0.0907	0.0127	0.2729
model_dr_05_100	0.0926	0.0147	0.2741
model_ld_128_100	0.0895	0.0116	0.2712
model_ld_256_100	0.0866	0.01	0.2651
model_ld_512_100	0.0882	0.0111	0.2681
model_ttv_602020_100	0.0835	0.0099	0.2552
model_ttv_701515_100	0.0905	0.0127	0.272
model_ttv_801010_100	0.091	0.011	0.2779

Table 15: SOAE Validation Reconstruction Loss Results (epoch = 100). This table shows the SOAE validation results regarding reconstruction loss which is combined with MSE loss and SSIM loss.

Name	validation/reconstruction_loss	validation/mse_loss	validation/ssim_loss
model_bs_08_100_soae	0.0883	0.0095	0.2721
model_bs_16_100_soae	0.0889	0.0101	0.2728
model_bs_32_100_soae	0.0901	0.0107	0.2754
model_dr_01_100_soae	0.1006	0.0183	0.2924
model_dr_03_100_soae	0.0962	0.0158	0.284
model_dr_05_100_soae	0.1062	0.0211	0.3047

Table 15 cont.

model_ld_128_100_soae	0.094	0.0134	0.2821
model_ld_256_100_soae	0.0919	0.0121	0.2781
model_ld_512_100_soae	0.0908	0.011	0.2768
model_ttv_602020_100_soae	0.1068	0.0197	0.3098
model_ttv_701515_100_soae	0.0941	0.0133	0.2828
model_ttv_801010_100_soae	0.0887	0.0133	0.2644

- CIAE Results (# of Epochs = 500):

    - The best validation CIAE reconstruction loss is 0.0615, observed in "model_bs_08_500" with MSE 0.0062 and SSIM 0.1908.

    - The second-best validation CIAE reconstruction loss is 0.0711, observed in "model_bs_16_500" with MSE 0.0064 and SSIM 0.2221.

    - The worst validation CIAE model is seen as "model_ttv_801010_500" with 0.0848 reconstruction loss, 0.0077 MSE loss, and 0.2648 SSIM loss.

    - For the validation phase, smaller batch sizes and balanced latent dimensions yield lower validation loss and improve MSE.

- SOAE Results (# of Epochs = 500):

    - The best validation SOAE reconstruction loss is 0.0768, seen in "model_ttv_801010_500_soae", with MSE 0.0070 and SSIM 0.2395.

    - "model_bs_16_500_soae" also gives good results such as 0.0824 reconstruction loss, 0.0070 MSE loss, and 0.2585 SSIM loss.

    - The worst validation SOAE result belongs to "model_ttv_602020_500_soae" with 0.0916 reconstruction loss, 0.0098 MSE loss, and 0.2827 SSIM loss.

    - Larger proportions of training data improve validation performance.

Table 16: CIAE Validation Reconstruction Loss Results (epoch = 500). This table shows the CIAE validation results regarding reconstruction loss which is combined with MSE loss and SSIM loss.

Name	**validation/reconstruction_loss**	**validation/mse_loss**	**validation/ssim_loss**

Table 16 cont.

model_bs_08_500	0.0615	0.0062	0.1908
model_bs_16_500	0.0711	0.0064	0.2221
model_bs_32_500	0.0817	0.0075	0.2547
model_dr_01_500	0.0818	0.0074	0.2554
model_dr_03_500	0.0815	0.0077	0.2537
model_dr_05_500	0.0832	0.0092	0.2558
model_ld_128_500	0.0835	0.0084	0.2586
model_ld_256_500	0.0816	0.0075	0.2544
model_ld_512_500	0.0801	0.007	0.2507
model_ttv_602020_500	0.0768	0.0066	0.2405
model_ttv_701515_500	0.0819	0.0076	0.2552
model_ttv_801010_500	0.0848	0.0077	0.2648

Table 17: SOAE Validation Reconstruction Loss Results (epoch = 500). This table shows the SOAE validation results regarding reconstruction loss which is combined with MSE loss and SSIM loss.

Name	validation/reconstruction_loss	validation/mse_loss	validation/ssim_loss
model_bs_08_500_soae	0.0829	0.0071	0.2597
model_bs_16_500_soae	0.0824	0.007	0.2585
model_bs_32_500_soae	0.0835	0.0073	0.2612
model_dr_01_500_soae	0.084	0.0076	0.2624
model_dr_03_500_soae	0.0879	0.0102	0.2691
model_dr_05_500_soae	0.089	0.0112	0.2705
model_ld_256_500_soae	0.0843	0.0077	0.2631
model_ld_512_500_soae	0.0832	0.0072	0.2606
model_ls_128_500_soae	0.0858	0.0085	0.266
model_ttv_602020_500_soae	0.0916	0.0098	0.2827
model_ttv_701515_500_soae	0.0846	0.0079	0.2635
model_ttv_801010_500_soae	0.0768	0.007	0.2395

- CIAE Results (# of Epochs = 1000):

  - The best validation reconstruction loss is achieved by "model_bs_08_1000" at 0.0582, with MSE Loss = 0.0048 and SSIM Loss = 0.1830.

  - The next best performance belongs to "model_bs_16_1000" (Reconstruction = 0.0689, MSE = 0.0055, SSIM = 0.2167).

- The worst validation result in CIAE is from “model_dr_05_1000” with 0.0825 reconstruction loss, including MSE loss = 0.0091 and SSIM loss = 0.2539.

- CIAE validation results show smaller batch sizes, moderate dropout, and higher latent dimensions improve generalization while larger batch sizes and excessive dropout harm performance.

- SOAE Results (# of Epochs = 1000):

  - The best validation performance is achieved by “model_ttv_801010_1000_soae”, with the lowest reconstruction loss of 0.0762, MSE loss of 0.0070, and SSIM loss of 0.2377. This model clearly outperforms others in generalization, indicating that the 80/10/10 TTV split may be beneficial for validation stability.

  - Second best validation model is “model_ld_512_1000_soae”, with a reconstruction loss of 0.0805, MSE of 0.0064 (the lowest among all), and SSIM of 0.2534. This reaffirms that a larger latent dimension (512) supports better generalization.

  - The worst performing model on validation is “model_ttv_602020_1000_soae”, with the highest reconstruction loss of 0.0880 and SSIM loss of 0.2746, despite a decent MSE of 0.0080. This might imply that this TTV split results in more challenging validation samples.

  - High dropout rates negatively impact validation performance.

Table 18: CIAE Validation Reconstruction Loss Results (epoch = 1000). This table shows the CIAE validation results regarding reconstruction loss which is combined with MSE loss and SSIM loss.

Name	validation/reconstruction_loss	validation/mse_loss	validation/ssim_loss
model_bs_08_1000	0.0582	0.0048	0.183
model_bs_16_1000	0.0689	0.0055	0.2167
model_bs_32_1000	0.0813	0.0074	0.2539
model_dr_01_1000	0.0791	0.0064	0.2488
model_dr_03_1000	0.0805	0.0076	0.2507
model_dr_05_1000	0.0825	0.0091	0.2539

Table 18 cont.

model_ld_128_1000	0.0815	0.0075	0.2542
model_ld_256_1000	0.0795	0.0068	0.2491
model_ld_512_1000	0.0784	0.0062	0.2468
model_ttv_602020_1000	0.076	0.0063	0.2384
model_ttv_701515_1000	0.0795	0.0066	0.2495
model_ttv_801010_1000	0.0813	0.0063	0.2563

Table 19: SOAE Validation Reconstruction Loss Results (epoch = 1000). This table shows the SOAE validation results regarding reconstruction loss which is combined with MSE loss and SSIM loss.

Name	**validation/reconstruction_loss**	**validation/mse_loss**	**validation/ssim_loss**
model_bs_08_1000_soae	0.082	0.0067	0.2579
model_bs_16_1000_soae	0.0841	0.0078	0.2622
model_bs_32_1000_soae	0.0818	0.0068	0.2569
model_dr_01_1000_soae	0.0837	0.0076	0.2611
model_dr_03_1000_soae	0.0867	0.0095	0.2666
model_dr_05_1000_soae	0.0875	0.0105	0.2673
model_ld_128_1000_soae	0.0852	0.0083	0.2645
model_ld_256_1000_soae	0.0846	0.0081	0.263
model_ld_512_1000_soae	0.0805	0.0064	0.2534
model_ttv_602020_1000_soae	0.088	0.008	0.2746
model_ttv_701515_1000_soae	0.0848	0.008	0.2638
model_ttv_801010_1000_soae	0.0762	0.007	0.2377

5.4.1.3. Test Performance

- CIAE Results (# of Epochs = 100):

    - The best CIAE model on the test set is “model_bs_08_100” with a reconstruction loss of 0.0525, made up of MSE = 0.0066 and SSIM = 0.1594.

    - Other configurations like “model_bs_16_100” and “model_ld_256_100” also demonstrate efficient results, gathering low test reconstruction losses below 0.09 with balanced MSE and SSIM losses.

- The worst CIAE model on the test set is “model_ttv_602020_100” with 0.0981 reconstruction loss, including 0.0123 MSE loss and 0.2981 SSIM loss.

- CIAE models indicate low MSE (~0.006–0.010) and SSIM (~0.15–0.25) losses on the test set, confirming their superior generalization ability.

- SOAE Results (# of Epochs = 100):

  - The best SOAE configuration on the test set is “model_bs_32_100_soae” with a reconstruction loss of 0.1132, comprising MSE loss= 0.0159 and SSIM loss= 0.3401.

  - The worst SOAE model on the test set belongs to “model_bs_16_100_soae” with a reconstruction loss of 0.3123, comprising MSE loss= 0.2482 and SSIM loss= 0.4619.

  - Overall, the models demonstrate high test reconstruction losses (~0.18), driven by MSE and SSIM loss values. This means a challenge in generalization for SOAE models.

Table 20: CIAE Test Reconstruction Loss Results (epoch = 100). This table shows the CIAE test results regarding reconstruction loss which is combined with MSE loss and SSIM loss.

Name	test/reconstruction_loss	test/mse_loss	test/ssim_loss
model_bs_08_100	0.0525	0.0066	0.1594
model_bs_16_100	0.0781	0.0098	0.2376
model_bs_32_100	0.0955	0.0119	0.2904
model_dr_01_100	0.0949	0.0115	0.2897
model_dr_03_100	0.0963	0.0125	0.2919
model_dr_05_100	0.098	0.0144	0.2932
model_ld_128_100	0.0958	0.012	0.2913
model_ld_256_100	0.0928	0.0104	0.2852
model_ld_512_100	0.0944	0.0114	0.2882
model_ttv_602020_100	0.0981	0.0123	0.2981
model_ttv_701515_100	0.0967	0.0128	0.2923
model_ttv_801010_100	0.0901	0.0111	0.2744

Table 21: SOAE Test Reconstruction Loss Results (epoch = 100). This table shows the SOAE test results regarding reconstruction loss which is combined with MSE loss and SSIM loss.

Name	test/reconstruction_loss	test/mse_loss	test/ssim_loss
model_bs_08_100_soae	0.1368	0.0359	0.3722
model_bs_16_100_soae	0.3123	0.2482	0.4619
model_bs_32_100_soae	0.1132	0.0159	0.3401
model_dr_01_100_soae	0.118	0.0191	0.3488
model_dr_03_100_soae	0.1566	0.0437	0.4202
model_dr_05_100_soae	0.1554	0.0398	0.4249
model_ld_128_100_soae	0.1203	0.0199	0.3546
model_ld_256_100_soae	0.1193	0.0187	0.3539
model_ld_512_100_soae	0.1159	0.0172	0.3461
model_ttv_602020_100_soae	0.1489	0.0207	0.4482
model_ttv_701515_100_soae	0.1479	0.0142	0.4599
model_ttv_801010_100_soae	0.1881	0.0884	0.4209

- CIAE Results (# of Epochs = 500):

  - The lowest CIAE test reconstruction loss is 0.0505, seen in "model_bs_08_500", with an MSE of 0.0053 and SSIM of 0.1560.

  - Higher test reconstruction losses like 0.0905 are seen in "model_ttv_602020_500", with MSE 0.0084 and SSIM 0.2819.

  - Lower batch sizes and balanced configurations result in better test performance.

- SOAE Results (# of Epochs = 500):

  - "model_dr_01_500_soae" achieves the lowest SOAE test reconstruction loss at 0.0961, with MSE loss 0.0078 and SSIM loss 0.3020.

  - "model_bs_16_500_soae" shows the highest loss at 0.2697, with MSE loss 0.2126 and SSIM loss 0.4030.

  - Minimal dropout rates significantly improve test reconstruction loss and MSE.

Table 22: CIAE Test Reconstruction Loss Results (epoch = 500). This table shows the CIAE test results regarding reconstruction loss which is combined with MSE loss and SSIM loss.

Name	test/reconstruction_loss	test/mse_loss	test/ssim_loss
model_bs_08_500	0.0505	0.0053	0.156
model_bs_16_500	0.0725	0.0066	0.2261
model_bs_32_500	0.0886	0.0081	0.2763
model_dr_01_500	0.0882	0.0079	0.2757
model_dr_03_500	0.0881	0.0082	0.2747
model_dr_05_500	0.0898	0.0097	0.2767
model_ld_128_500	0.0901	0.009	0.2793
model_ld_256_500	0.0884	0.0081	0.2759
model_ld_512_500	0.0866	0.0074	0.2714
model_ttv_602020_500	0.0905	0.0084	0.2819
model_ttv_701515_500	0.0885	0.0081	0.276
model_ttv_801010_500	0.0844	0.0078	0.2633

Table 23: SOAE Test Reconstruction Loss Results (epoch = 500). This table shows the SOAE test results regarding reconstruction loss which is combined with MSE loss and SSIM loss.

Name	test/reconstruction_loss	test/mse_loss	test/ssim_loss
model_bs_08_500_soae	0.2375	0.1297	0.489
model_bs_16_500_soae	0.2697	0.2126	0.403
model_bs_32_500_soae	0.1933	0.0717	0.477
model_dr_01_500_soae	0.0961	0.0078	0.302
model_dr_03_500_soae	0.0989	0.0106	0.3049
model_dr_05_500_soae	0.1629	0.0431	0.4426
model_ld_256_500_soae	0.1629	0.0375	0.4554
model_ld_512_500_soae	0.1678	0.0413	0.4628
model_ls_128_500_soae	0.1688	0.0416	0.4658
model_ttv_602020_500_soae	0.219	0.1166	0.458
model_ttv_701515_500_soae	0.108	0.0149	0.3252
model_ttv_801010_500_soae	0.1889	0.0681	0.4709

- CIAE Results (# of Epochs = 1000):

    - The best CIAE model on the test set is "model_bs_08_1000" with a reconstruction loss of 0.0477, made up of MSE = 0.0040 and SSIM = 0.1497.

- Other configuration like “model_bs_16_1000” also demonstrates efficient results, gathering low test reconstruction losses below 0.08 with balanced MSE and SSIM losses.

- The worst CIAE model on the test set is “model_ttv_602020_1000” with 0.0895 reconstruction loss, including 0.0080 MSE loss and 0.2796 SSIM loss.

- While smaller batch sizes, lower dropout, and larger latent dimensions improve generalization, large batch sizes, high dropout, and suboptimal train-test splits harm test performance.

- SOAE Results (# of Epochs = 1000):

  - The best SOAE test model result is achieved by “model_bs_16_1000_soae”, which records the lowest reconstruction loss at 0.1016, along with the lowest MSE loss of 0.0107 and lowest SSIM loss of 0.3138. This model generalizes extremely well and shows strong performance consistency across all datasets.

  - The second-best test model is “model_dr_01_1000_soae”, with a reconstruction loss of 0.1570, MSE loss of 0.0331, and SSIM loss of 0.4460. It demonstrates solid generalization ability, especially considering it was trained with a small amount of dropout (0.01), which seems to help avoid overfitting.

  - The worst test performance is observed in “model_ld_512_1000_soae”, with the highest reconstruction loss of 0.3571 and highest MSE loss of 0.3196, indicating severe overfitting. Despite excellent training and validation performance, this model fails to generalize to unseen data.

  - Dropout analysis shows that a small dropout rate of 0.01 (“model_dr_01_1000_soae”) leads to better generalization. As the dropout rate increases, performance worsens.

Table 24: CIAE Test Reconstruction Loss Results (epoch = 1000). This table shows the CIAE test results regarding reconstruction loss which is combined with MSE loss and SSIM loss.

Name	test/reconstruction_loss	test/mse_loss	test/ssim_loss
model_bs_08_1000	0.0477	0.0040	0.1497

Table 24 cont.

model_bs_16_1000	0.0699	0.0055	0.2202
model_bs_32_1000	0.0884	0.0080	0.2761
model_dr_01_1000	0.0857	0.0068	0.2699
model_dr_03_1000	0.0873	0.0080	0.2722
model_dr_05_1000	0.0892	0.0095	0.2749
model_ld_128_1000	0.0883	0.0080	0.2755
model_ld_256_1000	0.0861	0.0071	0.2705
model_ld_512_1000	0.0850	0.0066	0.2679
model_ttv_602020_1000	0.0895	0.0080	0.2796
model_ttv_701515_1000	0.0861	0.0070	0.2708
model_ttv_801010_1000	0.0813	0.0064	0.2562

Table 25: SOAE Test Reconstruction Loss Results (epoch = 1000). This table shows the SOAE test results regarding reconstruction loss which is combined with MSE loss and SSIM loss.

Name	**test/reconstruction_loss**	**test/mse_loss**	**test/ssim_loss**
model_bs_08_1000_soae	0.2014	0.0906	0.46
model_bs_16_1000_soae	0.1016	0.0107	0.3138
model_bs_32_1000_soae	0.3567	0.3111	0.4632
model_dr_01_1000_soae	0.157	0.0331	0.446
model_dr_03_1000_soae	0.2106	0.0886	0.495
model_dr_05_1000_soae	0.2044	0.1018	0.4439
model_ld_128_1000_soae	0.1908	0.0882	0.4301
model_ld_256_1000_soae	0.1914	0.0994	0.4061
model_ld_512_1000_soae	0.3571	0.3196	0.4446
model_ttv_602020_1000_soae	0.3446	0.2985	0.4523
model_ttv_701515_1000_soae	0.2037	0.0996	0.4467
model_ttv_801010_1000_soae	0.1916	0.0749	0.4637

5.4.2. Domain Classifier Results

In the context of a context-invariant autoencoder, high-domain accuracy and low adversarial loss indicate that the model is still able to differentiate between domains, which contradicts the goal of learning domain-invariant representations. A truly invariant model should confuse the adversary, leading to higher adversarial loss and a lower classification accuracy on domain-related tasks.

Based on 100 epoch CIAE model results, while the train adversarial loss values for all models appear to be well-minimized (0.0001 or 0.0002), this suggests that the adversarial

network is struggling to challenge the encoder, implying that the learned features may still contain domain-specific information. The model "model_bs_16_100", with a higher validation adversarial loss of 0.1483, shows better domain confusion, implying that the encoder is learning more domain-invariant representations.

In contrast, "model_ttv_801010_100" shows an extremely high validation adversarial loss (167.6713), which appears inconsistent with expectations. Such a high value may indicate a failure in training stability or an error in loss computation, rather than strong domain invariance. Normally, adversarial loss values are bounded (e.g., near 0 for success, near log(num_domains) for confusion), so this unusually large value should be further investigated.

Similarly, in the test phase, "model_bs_16_100" has the lowest test adversarial loss (0.1296), indicating that the adversary is still somewhat successful at predicting domains, which implies less domain confusion and therefore weaker domain invariance. On the other hand, "model_ttv_801010_100" shows the highest test adversarial loss (112.1894), which theoretically suggests stronger domain confusion, as the adversary fails to distinguish between domains. However, the extremely high loss value is unusual and may indicate instability or training issues, rather than true domain invariance. The high validation/test accuracy of "model_ld_256_100" (0.8476 / 0.8734) on domain classification suggests that its learned representations still contain domain-specific features, which contradicts the goal of achieving domain invariance. In contrast, "model_ttv_801010_100", with the lowest accuracy (0.7041 / 0.6956), may reflect stronger domain confusion. However, when considered alongside its extremely high adversarial loss, this result is more likely due to generalization failure or poor training performance, rather than successful suppression of domain information.

The learning rate follows an exponential decay pattern, starting at approximately 0.01 and stabilizing around 0.002, allowing for efficient early-stage exploration followed by fine-grained convergence. The lambda value increases in a stepwise manner, reaching 0.001 by the 40th epoch, which progressively enforces regularization and strengthens adversarial training. However, the consistently low adversarial loss across most models suggests that the discriminator is successfully distinguishing domain information, meaning the encoder is not effectively confusing the adversary. This implies that domain-specific features are still present in the learned representations. While low adversarial loss might superficially appear to indicate good training progress, in the context of domain confusion, it actually points to a failure to achieve domain invariance. This is supported by the high validation/test accuracy of domain classifiers in some models, further indicating the persistence of domain-related signals. To improve domain confusion and achieve more robust domain-invariant representations, further tuning may be needed—such as adjusting the adversary's capacity, modifying the lambda scaling schedule, or exploring architectural changes in the encoder-adversary setup. The key is to balance

domain invariance with task relevance, avoiding over-regularization that might degrade performance on the primary task.

For the 500-epoch CIAE models, all report a training adversarial loss of approximately 0.0001, indicating that the adversary is highly effective at distinguishing domain-specific features during training. This suggests that adversarial robustness is low, and the encoder is not sufficiently confusing the adversary, meaning domain-invariant representations are not being effectively learned.

During validation, "model_bs_32_500" achieves an adversarial loss of 0.1638, indicating limited domain confusion and weaker domain invariance. In contrast, "model_ttv_602020_500" shows a slightly higher adversarial loss of 0.2422, suggesting stronger domain confusion and improved domain invariance on the validation set.

Interestingly, in the test phase, "model_bs_08_500" achieves the highest adversarial loss of 0.3083, indicating the strongest domain confusion and most robust domain-invariant representations. Meanwhile, "model_ttv_602020_500" reports the lowest test adversarial loss of 0.1415, implying that its learned features retain more domain-specific information and exhibit weaker domain invariance on unseen data.

In terms of classification performance, "model_ttv_801010_500" achieves the highest validation accuracy (0.8302), while "model_bs_08_500" shows the lowest (0.7584). On the test set, "model_ld_256_500" performs best (0.8498), suggesting it generalizes well to unseen data. Again, "model_bs_08_500" has the lowest test accuracy (0.7858), which contrasts with its stronger domain invariance (indicated by its higher adversarial loss), suggesting that improved invariance does not necessarily lead to better task-specific performance in this case.

The lambda schedule follows a stepwise increase, stabilizing at 0.001, a common technique to gradually introduce adversarial regularization, allowing early flexibility and later enforcing stronger domain confusion. Meanwhile, the learning rate decays exponentially, starting around 0.01 and stabilizing at 0.002, which promotes rapid initial learning and fine-tuning in later stages to enhance generalization.

Based on the 1000-epoch CIAE domain classifier results, all models exhibit a training adversarial loss of 0.0001, indicating that the domain classifier is highly successful at distinguishing between domains. This suggests poor domain confusion and that the encoder has not fully learned domain-invariant features during training.

In the validation phase, 'model_ttv_801010_1000' achieves the lowest adversarial loss (0.1534), which actually indicates that the adversary is still performing relatively well, and domain-specific information remains. Conversely, 'model_ttv_602020_1000' exhibits

the highest adversarial loss (0.3887), suggesting that the adversary is less effective, indicating stronger domain confusion and more domain-invariant representations.

In the test phase, 'model_bs_16_1000' achieves the lowest adversarial loss (0.1656), which again means the adversary is more successful — so this reflects weaker domain confusion. On the other hand, 'model_dr_01_1000' shows the highest adversarial loss (0.2979), implying stronger domain confusion and possibly better domain-invariant learning. This suggests that dropout regularization may, in this case, have helped suppress domain-specific features.

In terms of classification accuracy, 'model_ttv_801010_1000' achieves the highest validation accuracy (0.8330), while 'model_bs_08_1000' scores the lowest (0.7648). In the test phase, 'model_ttv_701515_1000' performs best (0.8480), showing strong generalization to unseen data. Meanwhile, 'model_ttv_801010_1000' achieves the lowest test accuracy (0.7889), possibly indicating overfitting to the validation set or train-test domain shift affecting generalization.

The learning rate follows an exponential decay, stabilizing around 0.0017, which supports rapid early learning followed by fine-tuning. Similarly, the lambda schedule increases in a stepwise manner, stabilizing at 0.001, allowing for gradual regularization and progressive enforcement of domain confusion.

These findings suggest that a balanced train-validation-test split (70/15/15), moderate dropout, and carefully chosen latent dimensions contribute to better generalization and more effective domain-invariant feature learning. In contrast, excessive dropout or imbalanced data splits may degrade performance by either over-regularizing the model or reducing its ability to generalize.

All the details can be examined in APPENDIX A, APPENDIX B, and APPENDIX C.

Table 26: CIAE Domain Classifier Results (epoch = 100). This table shows the CIAE domain classifier results regarding adversarial loss, accuracy, learning rate, and lambda values.

Name	learning_rate	lambda	train/adversarial_loss	validation/adversarial_loss	test/adversarial_loss	validation/accuracy	test/accuracy
model_bs_08_100	0.0017	0.001	0.0002	0.2267	0.1796	0.7437	0.7727
model_bs_16_100	0.0017	0.001	0.0002	0.1483	0.1296	0.8188	0.8314
model_bs_32_100	0.0017	0.001	0.0002	0.1575	0.1473	0.8107	0.8366
model_dr_01_100	0.0017	0.001	0.0001	0.1614	0.1584	0.8407	0.8374
model_dr_03_100	0.0017	0.001	0.0002	0.1744	0.1502	0.8396	0.8698

Table 26 cont.

model_dr_05_100	0.0017	0.001	0.0002	0.1667	0.1544	0.8177	0.8432
model_ld_128_100	0.0017	0.001	0.0002	0.1581	0.1476	0.7777	0.7929
model_ld_256_100	0.0017	0.001	0.0001	0.1668	0.1451	0.8476	0.8734
model_ld_512_100	0.0017	0.001	0.0001	0.1542	0.1427	0.7973	0.8159
model_ttv_602020_100	0.0017	0.001	0.0002	0.183	0.1424	0.8088	0.8361
model_ttv_701515_100	0.0017	0.001	0.0002	0.1606	0.1529	0.7927	0.8126
model_ttv_801010_100	0.0017	0.001	0.0002	167.6713	112.1894	0.7041	0.6956

Table 27: CIAE Domain Classifier Results (epoch = 500). This table shows the CIAE domain classifier results regarding adversarial loss, accuracy, learning rate, and lambda values.

Name	**learning_rate**	**lambda**	**train/adversarial_loss**	**validation/adversarial_loss**	**test/adversarial_loss**	**validation/accuracy**	**test/accuracy**
model_bs_08_500	0.0017	0.001	0.0001	0.2053	0.3083	0.7584	0.7858
model_bs_16_500	0.0017	0.001	0.0001	0.179	0.1875	0.7979	0.8376
model_bs_32_500	0.0017	0.001	0.0001	0.1638	0.1527	0.7949	0.8261
model_dr_01_500	0.0017	0.001	0.0001	0.2046	0.178	0.827	0.8487
model_dr_03_500	0.0017	0.001	0.0001	0.1852	0.1812	0.8057	0.8391
model_dr_05_500	0.0017	0.001	0.0001	0.2045	0.1619	0.8044	0.8357
model_ld_128_500	0.0017	0.001	0.0001	0.2204	0.1764	0.8092	0.8296
model_ld_256_500	0.0017	0.001	0.0001	0.1939	0.1639	0.8226	0.8498
model_ld_512_500	0.0017	0.001	0.0001	0.201	0.2019	0.822	0.8287
model_ttv_602020_500	0.0017	0.001	0.0001	0.2422	0.1415	0.7762	0.8234
model_ttv_701515_500	0.0017	0.001	0.0001	0.1739	0.2036	0.819	0.8363
model_ttv_801010_500	0.0017	0.001	0.0001	0.2214	0.2423	0.8302	0.7877

Table 28: CIAE Domain Classifier Results (epoch = 1000). This table shows the CIAE domain classifier results regarding adversarial loss, accuracy, learning rate, and lambda values.

Name	learning_rate	lambda	train/adversarial_loss	validation/adversarial_loss	test/adversarial_loss	validation/accuracy	test/accuracy
model_bs_08_1000	0.0017	0.001	0.0001	0.2354	0.1784	0.7648	0.8076
model_bs_16_1000	0.0017	0.001	0.0001	0.1626	0.1656	0.7914	0.8172
model_bs_32_1000	0.0017	0.001	0.0001	0.2076	0.279	0.8042	0.8324
model_dr_01_1000	0.0017	0.001	0.0001	0.2491	0.2979	0.8211	0.8361
model_dr_03_1000	0.0017	0.001	0.0001	0.196	0.2695	0.8111	0.8193
model_dr_05_1000	0.0017	0.001	0.0001	0.2261	0.2397	0.8094	0.8385
model_ld_128_1000	0.0017	0.001	0.0001	0.1904	0.2264	0.8227	0.8456
model_ld_256_1000	0.0017	0.001	0.0001	0.179	0.2324	0.782	0.8226
model_ld_512_1000	0.0017	0.001	0.0001	0.3019	0.2132	0.8287	0.8402
model_ttv_602020_1000	0.0017	0.001	0.0001	0.3887	0.2332	0.7918	0.8291
model_ttv_701515_1000	0.0017	0.001	0.0001	0.2273	0.2104	0.8203	0.848
model_ttv_801010_1000	0.0017	0.001	0.0001	0.1534	0.2456	0.833	0.7889

5.4.3. Denoising Autoencoder Results

The CIAE models have been run by adding noise function to the CARLA simulator images for 1000 epochs. Based on the results;

- The models with the lowest reconstruction loss (0.3919) are:
    - model_denoising_bs_16_1000,
    - model_denoising_dr_01_1000,
    - model_denoising_ld_256_1000,
    - model_denoising_ttv_602020_1000,
    - model_denoising_ttv_801010_1000.

These models show the most effective pixel-level reconstruction, indicating robust denoising performance.

- The highest reconstruction loss (0.3922) is observed in "model_denoising_bs_8_1000", indicating slightly weaker denoising ability compared to the best models.

- Batch size variations have minimal impact, but smaller batch sizes (8) slightly degrade performance.

- Dropout (0.03 and 0.05) increases reconstruction loss slightly, but SSIM remains high, ensuring structural preservation.

- Latent dimension changes have a minor impact, with "ld_128_1000" showing marginally higher reconstruction loss.

- All models demonstrate strong SSIM scores (~0.9926 - 0.9927), meaning they effectively maintain perceptual quality.

Table 29: Denoising Autoencoder Train Reconstruction Loss Results (epoch = 1000). This table shows the CIAE training results with added noise function regarding reconstruction loss which is combined with MSE loss and SSIM loss.

Name	train/reconstruction_loss	train/mse_loss	train/ssim_loss
model_ denoising_bs_8_1000	0.3922	0.1348	0.9927
model_ denoising_bs_16_1000	0.3919	0.1344	0.9926
model_ denoising_bs_32_1000	0.392	0.1346	0.9927
model_ denoising_dr_01_1000	0.3919	0.1345	0.9926
model_ denoising_dr_03_1000	0.392	0.1346	0.9927
model_ denoising_dr_05_1000	0.392	0.1346	0.9927
model_denoising_ld_128_1000	0.3921	0.1347	0.9927
model_ denoising_ld_256_1000	0.3919	0.1345	0.9926
model_ denoising_ld_512_1000	0.392	0.1346	0.9927
model_ denoising_ttv_602020_1000	0.3919	0.1345	0.9927
model_ denoising_ttv_701515_1000	0.392	0.1345	0.9927
model_ denoising_ttv_801010_1000	0.3919	0.1344	0.9927

For the validation results;

- "model_denoising_bs_8_1000" achieves the lowest reconstruction loss (0.2706), indicating superior reconstruction quality.

- The highest validation reconstruction loss belongs to "model_denoising_ttv_801010_1000" (0.3845).

- Smaller batch sizes (8, 16) achieve lower reconstruction loss and MSE but struggle with structural preservation (low SSIM). Larger batch sizes (32) maintain strong SSIM but have higher reconstruction losses.

- Dropout regularization shows minimal impact on SSIM but slightly affects reconstruction loss.

- Latent dimension variations show minor performance differences.

- "model_denoising_ttv_602020_1000" has a lower SSIM (0.9090), suggesting potential domain adaptation issues.

Table 30: Denoising Autoencoder Validation Reconstruction Loss Results (epoch = 1000). This table shows the CIAE validation results with added noise function regarding reconstruction loss which is combined with MSE loss and SSIM loss.

Name	validation/ reconstruction_loss	validation/ mse_loss	validation/ ssim_loss
model_ denoising_bs_8_1000	0.2706	0.0866	0.6999
model_ denoising_bs_16_1000	0.3160	0.1015	0.8165
model_ denoising_bs_32_1000	0.3832	0.1226	0.9915
model_ denoising_dr_01_1000	0.3815	0.1201	0.9913
model_ denoising_dr_03_1000	0.3830	0.1223	0.9915
model_ denoising_dr_05_1000	0.3843	0.1241	0.9915
model_denoising_ld_128_1000	0.3828	0.1219	0.9915
model_ denoising_ld_256_1000	0.3818	0.1205	0.9914
model_ denoising_ld_512_1000	0.3836	0.1231	0.9915
model_ denoising_ttv_602020_1000	0.3542	0.1165	0.909
model_ denoising_ttv_701515_1000	0.3820	0.1209	0.9914
model_ denoising_ttv_801010_1000	0.3845	0.1243	0.9915

Based on the test results;

- "model_denoising_bs_8_1000" achieves the lowest test reconstruction loss (0.2143), suggesting superior performance in reconstructing clean images.

- The highest test reconstruction loss belongs to "model_denoising_ttv_801010_1000" (0.3839).

- Small batch sizes (8, 16) yield lower reconstruction and MSE losses but significantly reduce SSIM, meaning they reconstruct images more accurately at the pixel level but fail to retain overall structure.

- Larger batch sizes (32) and dropout models (dr_03, dr_05) maintain higher SSIM values (~0.9915) but have slightly higher reconstruction losses.

- Latent dimension adjustments have minimal impact on performance, with all models stabilizing near 0.9915 SSIM.

- Train-test split variations show little difference, except for "model_denoising_ttv_801010_1000", which has a slightly higher reconstruction loss.

Table 31: Denoising Autoencoder Test Reconstruction Loss Results (epoch = 1000). This table shows the CIAE test results with added noise function regarding reconstruction loss which is combined with MSE loss and SSIM loss.

Name	test/reconstruction_loss	test/mse_loss	test/ssim_loss
model_ denoising_bs_8_1000	0.2143	0.0687	0.5541
model_ denoising_bs_16_1000	0.3157	0.101	0.8165
model_ denoising_bs_32_1000	0.3841	0.1237	0.9916
model_ denoising_dr_01_1000	0.3821	0.1209	0.9915
model_ denoising_dr_03_1000	0.3832	0.1224	0.9916
model_ denoising_dr_05_1000	0.3830	0.1222	0.9916
model_denoising_ld_128_1000	0.3837	0.1231	0.9916
model_ denoising_ld_256_1000	0.3824	0.1214	0.9915
model_ denoising_ld_512_1000	0.3834	0.1228	0.9915
model_ denoising_ttv_602020_1000	0.3833	0.1226	0.9916
model_ denoising_ttv_701515_1000	0.3826	0.1216	0.9915
model_ denoising_ttv_801010_1000	0.3839	0.1235	0.9915

The domain classifier results show;

- Among the models, "model_denoising_bs_16_1000" and "model_denoising_bs_32_1000" exhibit extremely high adversarial losses on both the validation (3111.74 and 2214.59) and test datasets (4130.19 and 2821.30). These high adversarial losses suggest that the domain classifier is struggling to

distinguish between domains, indicating strong domain confusion and potentially better domain-invariant feature learning.

- Interestingly, despite this, their domain classification accuracy remains moderate (validation: 0.6361, 0.6741; test: 0.6064, 0.6804), suggesting that while domain confusion is high (as indicated by adversarial loss), some domain-specific information may still be retained or the model may be experiencing instability due to the extreme adversarial loss values. Such high losses could also point to training anomalies or numerical issues, especially if adversarial loss isn't bounded or properly scaled.

- In contrast, "model_denoising_bs_8_1000" and "model_denoising_ttv_602020_1000" demonstrate low adversarial losses (validation: 0.3417 and 24.3546; test: 0.2989 and 25.2322), meaning that the adversary is succeeding at domain classification. This indicates that domain-specific information is still present, and domain confusion is weaker.

- However, these models also achieve the highest task classification accuracies (validation: 0.8646 and 0.8617; test: 0.8888 and 0.8783), suggesting that while domain invariance is not fully achieved, task-relevant features are well preserved, resulting in strong performance on the main classification task.

- Overall, in the CIAE framework, the ideal model strikes a balance between high adversarial loss (i.e., strong domain confusion) and high classification accuracy (i.e., preservation of task-relevant features). Models with low adversarial loss may perform well on the task, but at the cost of retaining domain-specific features, which undermines the goal of learning truly domain-invariant representations.

All the details can be found in APPENDIX K and APPENDIX L.

Table 32: Denoising CIAE Domain Classifier Results (epoch = 1000). This table shows the CIAE domain classifier results regarding adversarial loss, accuracy, learning rate, and lambda values.

Name	learning_rate	lambda	train/adversarial_loss	validation/adversarial_loss	test/adversarial_loss	validation/accuracy	test/accuracy
model_denoising_bs_8_1000	0.0017	0.001	0.0004	0.3417	0.2989	0.8646	0.8888
model_denoising_bs_16_1000	0.0017	0.001	0.0003	3111.7405	4130.1934	0.6361	0.6064
model_denoising_bs_32_1000	0.0017	0.001	0.0003	2214.5928	2821.303	0.6741	0.6804

Table 32 cont.

model_ denoising_dr_01_ 1000	0.0017	0.0 01	0.0003	0.505	0.4368	0.8318	0.8714
model_ denoising_dr_03_ 1000	0.0017	0.0 01	0.0003	0.3242	0.3649	0.817	0.8485
model_ denoising_dr_05_ 1000	0.0017	0.0 01	0.0003	5.0856	0.2698	0.8576	0.8721
model_denoising _ld_128_1000	0.0017	0.0 01	0.0003	0.4508	0.378	0.8311	0.8548
model_ denoising_ld_256 _1000	0.0017	0.0 01	0.0003	90.8461	85.8306	0.7807	0.7874
model_ denoising_ld_512 _1000	0.0017	0.0 01	0.0003	0.3986	0.3311	0.8142	0.858
model_ denoising_ttv_60 2020_1000	0.0017	0.0 01	0.0003	24.3546	25.2322	0.8617	0.8783
model_ denoising_ttv_70 1515_1000	0.0017	0.0 01	0.0003	6.203	9.3168	0.8111	0.8294
model_ denoising_ttv_80 1010_1000	0.0017	0.0 01	0.0003	64.8734	59.7937	0.7316	0.7513

5.4.4. Model Discussions

5.4.4.1.Impact of Hyperparameters for CIAE and SOAE Comparison

To examine how changing hyperparameters affects the model performances in terms of train/validation/test reconstruction loss for 100, 500, and 1000 epochs, further analysis has been made for CIAE and SOAE. The comparison of the values for CIAE and SOAE can be found in APPENDIX J.

Based on the results in Table 33, the comparison has been made for batch size:

- Increasing the number of epochs leads to better performance in terms of lower reconstruction loss for train, validation, and test sets.

- Smaller batch sizes (8 or 16) and higher number of epochs (1000) lead to better performance across train, validation, and test sets for reconstruction loss, MSE, and SSIM loss.

- Larger batch sizes tend to overfit, as seen from higher validation and test losses at 1000 epochs.

- Test performance follows the same trend as validation; smaller batch sizes generalize better, while larger batch sizes degrade performance.

Table 33: CIAE Model Performance Comparison Based on Batch Size. This table shows how batch size affects the model performances in terms of train/validation/test reconstruction loss.

CIAE	Name	train/reconstruction_loss	train/mse_loss	train/ssim_loss	validation/reconstruction_loss	validation/mse_loss	validation/ssim_loss	test/reconstruction_loss	test/mse_loss	test/ssim_loss
epochs=100	model_bs_08_100	0.0901	0.0145	0.2664	0.0647	0.0082	0.1967	0.0525	0.0066	0.1594
	model_bs_16_100	0.0903	0.0139	0.2686	0.0775	0.0099	0.2353	0.0781	0.0098	0.2376
	model_bs_32_100	0.0884	0.0136	0.2629	0.0895	0.0117	0.2708	0.0955	0.0119	0.2904
epochs=500	model_bs_08_500	0.0819	0.0086	0.2531	0.0615	0.0062	0.1908	0.0505	0.0053	0.156
	model_bs_16_500	0.0794	0.0077	0.2468	0.0711	0.0064	0.2221	0.0725	0.0066	0.2261
	model_bs_32_500	0.0802	0.0083	0.248	0.0817	0.0075	0.2547	0.0886	0.0081	0.2763
epochs=1000	model_bs_08_1000	0.0778	0.0073	0.2424	0.0582	0.0048	0.183	0.0477	0.004	0.1497
	model_bs_16_1000	0.0772	0.0072	0.2405	0.0689	0.0055	0.2167	0.0699	0.0055	0.2202
	model_bs_32_1000	0.0781	0.0073	0.2436	0.0813	0.0074	0.2539	0.0884	0.008	0.2761

Furthermore, as it can be seen in Table 34, for SOAE models we can conclude that;

- Longer training durations (e.g., 500–1000 epochs) generally lead to lower training and validation reconstruction losses, indicating that the model continues to optimize and generalize well with extended learning.

- Batch size 16 consistently achieves the best test reconstruction performance, especially after 1000 epochs. While smaller batch sizes (e.g., 8) show good generalization early on, they do not outperform batch size 16 at longer training durations. Conversely, batch size 32 starts off well but tends to overfit or generalize poorly after extended training.

- Training and validation reconstruction losses steadily decrease and stabilize over time, showing effective convergence and generalization. However, test reconstruction loss does not always follow this trend, suggesting some sensitivity to overfitting and possible need for early stopping or regularization in longer runs.

Table 34: SOAE Model Performance Comparison Based on Batch Size. This table shows how batch size affects the model performances in terms of train/validation/test reconstruction loss.

SOAE	Name	train/reconstruction_loss	train/mse_loss	train/ssim_loss	validation/reconstruction_loss	validation/mse_loss	validation/ssim_loss	test/reconstruction_loss	test/mse_loss	test/ssim_loss
epochs=100	model_bs_08_100_soae	0.0883	0.0129	0.264	0.0883	0.0095	0.2721	0.1368	0.0359	0.3722
	model_bs_16_100_soae	0.0874	0.0119	0.2636	0.0889	0.0101	0.2728	0.3123	0.2482	0.4619
	model_bs_32_100_soae	0.0882	0.0119	0.266	0.0901	0.0107	0.2754	0.1132	0.0159	0.3401
epochs=500	model_bs_08_500_soae	0.0823	0.0088	0.2539	0.0829	0.0071	0.2597	0.2375	0.1297	0.489
	model_bs_16_500_soae	0.0808	0.008	0.2506	0.0824	0.007	0.2585	0.2697	0.2126	0.403
	model_bs_32_500_soae	0.0807	0.0081	0.2501	0.0835	0.0073	0.2612	0.1933	0.0717	0.477
epochs=1000	model_bs_08_1000_soae	0.0807	0.0081	0.2501	0.082	0.0067	0.2579	0.2014	0.0906	0.46
	model_bs_16_1000_soae	0.0818	0.0083	0.2533	0.0841	0.0078	0.2622	0.1016	0.0107	0.3138

Table 34 cont.

	model_bs_32_1000_soae	0.0782	0.0072	0.2438	0.0818	0.0068	0.2569	0.3567	0.3111	0.4632

In terms of domain classifier model results, the values can be seen in Table 35. Based on the results;

- Smaller batch sizes (e.g., batch_size=8) tend to result in higher adversarial losses for validation and test phases, particularly at epoch 500 and 1000, indicating weaker domain separation.

- Larger batch sizes (e.g., batch_size=32) generally achieve lower adversarial losses in the validation phase (especially at epoch 500), suggesting better stability during training. However, at epoch 1000, test adversarial loss for batch_size=32 increases significantly (0.2790), indicating potential overfitting.

- Smaller batch sizes generally result in lower accuracy for both validation and test datasets, particularly in earlier epochs (100, 500). At epoch 1000, batch_size=8 still has the lowest accuracy compared to larger batch sizes.

- Larger batch sizes tend to provide higher validation and test accuracy across all epochs, with batch_size=32 achieving the highest test accuracy (0.8324) at epoch 1000. However, the gap between batch sizes narrows at higher epochs, suggesting that longer training benefits smaller batch sizes as well.

Table 35: CIAE Model Performance Comparison Based on Batch Size. This table shows how batch size affects the model performances in terms of train/validation/test domain classifier results.

CIAE	Name	train/adversarial_loss	validation/adversarial_loss	test/adversarial_loss	validation/accuracy	test/accuracy
epochs=100	**model_bs_08_100**	0.0002	0.2267	0.1796	0.7437	0.7727
	model_bs_16_100	0.0002	0.1483	0.1296	0.8188	0.8314
	model_bs_32_100	0.0002	0.1575	0.1473	0.8107	0.8366
epochs=500	**model_bs_08_500**	0.0001	0.2053	0.3083	0.7584	0.7858
	model_bs_16_500	0.0001	0.179	0.1875	0.7979	0.8376
	model_bs_32_500	0.0001	0.1638	0.1527	0.7949	0.8261
epochs=1000	**model_bs_08_1000**	0.0001	0.2354	0.1784	0.7648	0.8076

Table 35 cont.

	model_bs_16_1000	0.0001	0.1626	0.1656	0.7914	0.8172
	model_bs_32_1000	0.0001	0.2076	0.279	0.8042	0.8324

Similarly, the models have been compared based on latent dimension parameter. For CIAE models, the values can be examined in terms of reconstruction loss in Table 36. Based on the results, we can conclude that;

- Smaller latent dimensions (e.g., ld=128) result in higher training reconstruction loss compared to larger dimensions (e.g., ld=512).
- Validation reconstruction loss decreases as the latent dimension increases, with the trend remains consistent across 100, 500, and 1000 epochs.
- Test reconstruction loss shows similar behavior, improving with larger latent dimensions.
- Increasing the number of epochs significantly reduces the training reconstruction loss, reflecting better optimization.
- Validation loss decreases consistently with longer training.
- Test loss also benefits from longer training, with reductions across all latent dimensions.

Table 36: CIAE Model Performance Comparison Based on Latent Dimension. This table shows how latent dimension affects the model performances in terms of train/validation/test reconstruction loss

CIAE	**Name**	**train/reconstruction_loss**	**train/mse_loss**	**train/ssim_loss**	**validation/reconstruction_loss**	**validation/mse_loss**	**validation/ssim_loss**	**test/reconstruction_loss**	**test/mse_loss**	**test/ssim_loss**
epochs=100	**model_ld_128_100**	0.0909	0.0149	0.268	0.0895	0.0116	0.2712	0.0958	0.012	0.2913
	model_ld_256_100	0.0886	0.0133	0.2643	0.0866	0.01	0.2651	0.0928	0.0104	0.2852
	model_ld_512_100	0.0878	0.0126	0.2631	0.0882	0.0111	0.2681	0.0944	0.0114	0.2882
epochs	**model_ld_128_500**	0.0819	0.0089	0.2523	0.0835	0.0084	0.2586	0.0901	0.009	0.2793

Table 36 cont.

=500	**model_ld_256_500**	0.0793	0.0076	0.2467	0.0816	0.0075	0.2544	0.0884	0.0081	0.2759
	model_ld_512_500	0.0808	0.0088	0.2486	0.0801	0.007	0.2507	0.0866	0.0074	0.2714
epochs=1000	**model_ld_128_1000**	0.0796	0.0079	0.2468	0.0815	0.0075	0.2542	0.0883	0.008	0.2755
	model_ld_256_1000	0.0772	0.0072	0.2406	0.0795	0.0068	0.2491	0.0861	0.0071	0.2705
	model_ld_512_1000	0.0762	0.0067	0.2382	0.0784	0.0062	0.2468	0.085	0.0066	0.2679

Similarly, for SOAE models the comparison has been made for latent dimension. As it can be figured out from Table 37;

- Larger latent dimensions (ld=512) offer better training and validation loss, but show signs of overfitting during extended training (1000 epochs), as seen in elevated test losses.
- Latent dimension = 256 provides the best trade-off: low training/validation loss, relatively strong SSIM, and good generalization on test data across all epochs.
- Training up to 500 epochs appears optimal; pushing to 1000 yields diminishing returns or even performance degradation, especially for larger models.
- Future runs might benefit from early stopping or regularization strategies (e.g., dropout, weight decay) to avoid overfitting in high-capacity models.

Table 37: SOAE Model Performance Comparison Based on Latent Dimension. This table shows how latent dimension affects the model performances in terms of train/validation/test reconstruction loss.

SOAE	**Name**	**train/reconstruction_loss**	**train/mse_loss**	**train/ssim_loss**	**validation/reconstruction_loss**	**validation/mse_loss**	**validation/ssim_loss**	**test/reconstruction_loss**	**test/mse_loss**	**test/ssim_loss**
epochs	**model_ld_128_100_soae**	0.0907	0.0138	0.2703	0.094	0.0134	0.2821	0.1203	0.0199	0.3546

Table 37 cont.

=100	**model_ld_256_100_soae**	0.0882	0.0122	0.2656	0.0919	0.0121	0.2781	0.1193	0.0187	0.3539
	model_ld_512_100_soae	0.087	0.0116	0.2628	0.0908	0.011	0.2768	0.1159	0.0172	0.3461
epochs=500	**model_ld_128_500_soae**	0.0834	0.0097	0.2552	0.0858	0.0085	0.266	0.1688	0.0416	0.4658
	model_ld_256_500_soae	0.0816	0.0084	0.2524	0.0843	0.0077	0.2631	0.1629	0.0375	0.4554
	model_ld_512_500_soae	0.0804	0.0081	0.2492	0.0832	0.0072	0.2606	0.1678	0.0413	0.4628
epochs=1000	**model_ld_128_1000_soae**	0.0825	0.0092	0.2535	0.0852	0.0083	0.2645	0.1908	0.0882	0.4301
	model_ld_256_1000_soae	0.0814	0.0086	0.2512	0.0846	0.0081	0.263	0.1914	0.0994	0.4061
	model_ld_512_1000_soae	0.0776	0.0066	0.2433	0.0805	0.0064	0.2534	0.3571	0.3196	0.4446

In terms of domain classifier models, the model performance results can be examined based on latent dimension as it can be seen in Table 38. Based on the results;

- Larger latent dimensions improve early performance but may lead to instability in longer training. While “model_ld_256_100” performs best at 100 epochs, “model_ld_512_1000” achieves higher accuracy at 1000 epochs, despite increasing adversarial loss.

- Extended training (1000 epochs) does not always improve performance. Some models, like “model_ld_256_1000”, validation accuracy decline over time, suggesting potential overfitting.

- Test performance is more stable than validation performance. Validation accuracy fluctuates more across epochs, while test accuracy remains within a narrower range, indicating stronger generalization.

- Larger latent spaces require fine-tuned regularization. “model_ld_512_1000” shows high adversarial loss, suggesting the need for dropout adjustments, weight decay, or learning rate scheduling.

- Early stopping and adaptive training strategies could improve results. Since longer training increases adversarial loss, stopping around 500–750 epochs or using cosine decay learning rate schedules may help stabilize performance.

Table 38: CIAE Model Performance Comparison Based on Latent Dimension. This table shows how latent dimension affects the model performances in terms of train/validation/test domain classifier results

CIAE	Name	train/adversarial_loss	validation/adversarial_loss	test/adversarial_loss	validation/accuracy	test/accuracy
epochs =100	model_ld_128_100	0.0002	0.1581	0.1476	0.7777	0.7929
	model_ld_256_100	0.0001	0.1668	0.1451	0.8476	0.8734
	model_ld_512_100	0.0001	0.1542	0.1427	0.7973	0.8159
epochs =500	model_ld_128_500	0.0001	0.2204	0.1764	0.8092	0.8296
	model_ld_256_500	0.0001	0.1939	0.1639	0.8226	0.8498
	model_ld_512_500	0.0001	0.201	0.2019	0.822	0.8287
epochs =1000	model_ld_128_1000	0.0001	0.1904	0.2264	0.8227	0.8456
	model_ld_256_1000	0.0001	0.179	0.2324	0.782	0.8226
	model_ld_512_1000	0.0001	0.3019	0.2132	0.8287	0.8402

Other than these, it can be gathered from the final results the top three model configurations based on test accuracy can be gathered in Table 39. All the output images can be examined in APPENDIX D, APPENDIX E, APPENDIX F, APPENDIX G, APPENDIX H, and APPENDIX I.

Table 39: The Top Three CIAE Model Configurations Based on Test Accuracy. This table shows the top three model configurations based on test accuracy.

Name	model_ld_256_100	model_dr_03_100	model_ld_256_500
train/reconstruction_loss	0.0886	0.0943	0.0793
train/mse_loss	0.0133	0.0159	0.0076
train/ssim_loss	0.2643	0.2771	0.2467
validation/reconstruction_loss	0.0866	0.0907	0.0816

Table 39 cont.

validation/mse_loss	0.01	0.0127	0.0075
validation/ssim_loss	0.2651	0.2729	0.2544
test/reconstruction_loss	0.0928	0.0963	0.0884
test/mse_loss	0.0104	0.0125	0.0081
test/ssim_loss	0.2852	0.2919	0.2759
batch_size	32	32	32
epochs	100	100	500
input_shape	[480,640,3]	[480,640,3]	[480,640,3]
latent_dim	256	256	256
learning_rate	0.0017	0.0017	0.0017
lambda	0.001	0.001	0.001
train/adversarial_loss	0.0001	0.0002	0.0001
validation/adversarial_loss	0.1668	0.1744	0.1939
test/adversarial_loss	0.1451	0.1502	0.1639
validation/accuracy	0.8476	0.8396	0.8226
test/accuracy	0.8734	0.8698	0.8498

Moreover, the top three model configurations based on test reconstruction loss can be examined in Table 40.

Table 40: The Top Three Model Configurations Based on Test Reconstruction Loss. This table shows the top three model configurations based on test reconstruction loss.

Name	**model_bs_08_1000**	**model_bs_08_500**	**model_bs_08_100**
train/reconstruction_loss	0.0778	0.0819	0.0901
train/mse_loss	0.0073	0.0086	0.0145
train/ssim_loss	0.2424	0.2531	0.2664
validation/reconstruction_loss	0.0582	0.0615	0.0647
validation/mse_loss	0.0048	0.0062	0.0082
validation/ssim_loss	0.183	0.1908	0.1967
test/reconstruction_loss	0.0477	0.0505	0.0525
test/mse_loss	0.004	0.0053	0.0066
test/ssim_loss	0.1497	0.156	0.1594
batch_size	8	8	8
epochs	1000	500	100
input_shape	[480,640,3]	[480,640,3]	[480,640,3]
latent_dim	256	256	256
learning_rate	0.0017	0.0017	0.0017
lambda	0.001	0.001	0.001
train/adversarial_loss	0.0001	0.0001	0.0002

Table 40 cont.

validation/adversarial_loss	0.2354	0.2053	0.2267
test/adversarial_loss	0.1784	0.3083	0.1796
validation/accuracy	0.7648	0.7584	0.7437
test/accuracy	0.8076	0.7858	0.7727

5.4.4.2.CIAE and Denoising CIAE Comparison

To reflect the reality of real-world images, noise function has been added to CARLA simulator images and run for 1000 epochs as discussed in the previous chapters. Based on the model performances, the comparison has been made for CIAE and denoising CIAE models. The details can be found in APPENDIX M.

In terms of average reconstruction loss, MSE and SSIM loss;

- CIAE achieves significantly lower reconstruction and MSE losses than denoising CIAE on all datasets. For example, on the test set, CIAE has an MSE of 0.0071 vs. Denoising CIAE's 0.1162, indicating better pixel-level accuracy.

- Lower SSIM loss indicates higher structural similarity to the original image. CIAE again performs better with much lower SSIM losses. For instance, test SSIM loss for CIAE is 0.2570, while denoising CIAE is 0.9405, suggesting that CIAE preserves image structure significantly better.

CIAE is clearly the superior model in terms of both pixel-level accuracy (MSE/Reconstruction Loss) and perceptual quality (SSIM). Denoising CIAE, despite its name, seems to degrade performance substantially, likely due to an overly aggressive denoising approach that harms reconstruction quality.

Table 41: Comparison of CIAE and Denoising CIAE Models in Terms of Reconstruction Loss, MSE And SSIM Loss. This table shows the model runs based on CIAE and denoising CIAE models for 1000 epochs.

	train/avg_reconstruction_loss	**train/avg_mse_loss**	**train/avg_ssim_loss**	**validation/avg_reconstruction_loss**	**validation/avg_mse_loss**	**validation/avg_ssim_loss**	**test/avg_reconstruction_loss**	**test/avg_mse_loss**	**test/avg_ssim_loss**
CIAE	0.0785	0.0075	0.2444	0.0772	0.0067	0.2418	0.082	0.0071	0.257
Denoising CIAE	0.392	0.1346	0.9927	0.3656	0.117	0.9457	0.3635	0.1162	0.9405

In terms of average domain adversarial loss and domain accuracy;

- CIAE shows a massive spike in test adversarial loss: 594.48, while its train loss is near zero and validation loss is 0.79. CIAE may have overfit its adversarial component or encountered instability at test time.

- Denoising CIAE adversarial losses are much more consistent and reasonable across validation (0.2265) and test (0.2326), implying better generalization.

- On the test set, denoising CIAE has higher accuracy: 0.8271 vs. CIAE's 0.8106.

- While the difference isn't huge, it suggests CIAE may preserve features more relevant to classification.

Table 42: Comparison of CIAE and Denoising CIAE Models in Terms of Domain Adversarial Loss and Domain Accuracy This table shows the model runs based on CIAE and denoising CIAE models for 1000 epochs.

	train/avg_adversarial_loss	validation/avg_adversarial_loss	test/avg_adversarial_loss	validation/avg_accuracy	test/avg_accuracy
CIAE	0.0003	0.7926	594.4791	459.9764	0.8106
Denoising CIAE	0.0001	0.2265	0.2326	0.8067	0.8271

CHAPTER 6

CONCLUSION

This study has made several contributions to the development of context-invariant autoencoders through unsupervised domain adaptation. By using domain adversarial training and advanced data augmentation techniques, the suggested methodology enables extracting domain-invariant representations of images that remain stable across varying contexts. To this end, we utilized a domain classifier which guesses the domain from which the image is coming based on the latent representation of the image obtained by the encoder block. Gradient reversal layer is employed by the encoder to confuse the domain classifier. At the same time, the output of the decoder, which is the reconstructed image, is used to extract features that are necessary to reconstruct the original image. A systematic evaluation of key hyperparameters, including latent dimensions, batch sizes, number of epochs, and train-test-validation dataset splits emphasize the balance between domain-invariance and classification accuracy.

The proposed context-invariant autoencoder training methodology is capable of generating stable and domain-invariant representations across different varying contexts like weather changes and time of day within the CARLA simulator dataset. Furthermore, during the encoding-decoding phase, the combined reconstruction loss ensures the main image structures are absorbed. Overall, the models indicate consistent performance across training, validation, and test datasets, validating their generalizability.

The significance of this work lies in enhancing the generalizability of machine learning models for real-world scenarios. By extracting domain-invariant features, the suggested methodology helps models become less sensitive to contextual changes. Applications like autonomous driving require adjustment to weather and lighting changes. This study ensures robust and stable framework in unseen contexts by facilitating downstream tasks like classification, detection, and reconstruction. In the future, this study can be used to improve the real-world scenarios that has been faced with rural areas to forecast different weather condition and environmental factors changes. In addition to them, autonomous driving approaches might be enhanced by applying domain-invariant features. The target here is to adapt the model to the changing and unpredictable environments by providing realistic solutions. Consequently, the results reflect more reliability and robustness in dynamic and unpredictable environments by altering hyperparameter settings.

The denoising variant of CIAE achieved superior generalization and stable results compared to standard CIAE models. These results clearly indicate that adding noise to source images aids in robust feature learning, especially in unpredictable environments. Moreover, this work compared Source-Only Autoencoders (SOAE) and CIAE models across various hyperparameter configurations. CIAE consistently outperformed SOAE in terms of structural preservation (lower SSIM loss) and generalization.

During this study, many challenges were faced. The primary obstacle among them was processing power. Since the model has complex architecture and long training cycles, high memory and computing power were needed. Due to that reason, the complexity of the model and image sizes were kept at a manageable level.

Capturing images for nine different weather conditions on the same route presented another obstacle. Under real circumstances, to be able to arrange environmental setup based on that might have required constant situations and parameters. Therefore, CARLA was chosen to be used in this model in order to capture mostly the similar environment pictures under nine different weather conditions.

Even though this study has been added value to the literature, several further steps can be considered as future works. During the literature review, the gap has been identified in terms of lack of studies in rural areas. Due to that reason, this study has been established in rural area conditions by using Town07 map in CARLA Simulator (*Town 7*, 2024). To illustrate, this study can be applied to autonomous driving models to be able to get more concrete results. Furthermore, by using semantic segmentation algorithms, the images can be examined and labeled to provide efficient results in autonomous driving. To obtain better results, the number of epochs can be increased such as 1500 or 20000 by considering overfitting or deeper architectures (e.g., β-VAE) can be used.

REFERENCES

Ardelean, E. R., Coporîie, A., Ichim, A. M., Dînşoreanu, M., & Mureşan, R. C. (2023). A study of autoencoders as a feature extraction technique for spike sorting. *PLoS ONE*, *18*(3 March). https://doi.org/10.1371/journal.pone.0282810

Battikh, M. S., & Lenskiy, A. A. (2021). *Latent-Insensitive autoencoders for Anomaly Detection*. http://arxiv.org/abs/2110.13101

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., & Vaughan, J. W. (2010). A theory of learning from different domains. *Machine Learning*, *79*(1–2), 151–175. https://doi.org/10.1007/s10994-009-5152-4

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1798–1828. https://doi.org/10.1109/TPAMI.2013.50

Brückner, D. B., Ronceray, P., & Broedersz, C. P. (2020). *Inferring the dynamics of underdamped stochastic systems*. https://doi.org/10.1103/PhysRevLett.125.058103

Carla Introduction. (2024). https://carla.org/

Chen, E., Ho, C., Maulimov, M., Wang, C., & Scherer, S. (2023). *Learning-on-the-Drive: Self-supervised Adaptation of Visual Offroad Traversability Models*. http://arxiv.org/abs/2306.15226

Chen, S., & Guo, W. (2023). Auto-Encoders in Deep Learning—A Review with New Perspectives. In *Mathematics* (Vol. 11, Issue 8). MDPI. https://doi.org/10.3390/math11081777

Dave Bergmann, & Cole Stryker. (2023, November 23). *What is an autoencoder?* https://www.ibm.com/think/topics/autoencoder

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., & Koltun, V. (2017). *CARLA: An Open Urban Driving Simulator*. http://arxiv.org/abs/1711.03938

Espadinha, J., Lebedev, I., Lukic, L., & Bernardino, A. (2021). LiDAR data noise models and methodology for sim-to-real domain generalization and adaptation in autonomous driving perception. *IEEE Intelligent Vehicles Symposium, Proceedings*, *2021-July*, 797–803. https://doi.org/10.1109/IV48863.2021.9576034

Fournier, Q., & Aloise, D. (2019). Empirical comparison between autoencoders and traditional dimensionality reduction methods. *Proceedings - IEEE 2nd International Conference on Artificial Intelligence and Knowledge Engineering, AIKE 2019*, 211–214. https://doi.org/10.1109/AIKE.2019.00044

Ganin, Y., & Lempitsky, V. (2014). *Unsupervised Domain Adaptation by Backpropagation*. http://arxiv.org/abs/1409.7495

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., & Lempitsky, V. (2015). *Domain-Adversarial Training of Neural Networks*. http://arxiv.org/abs/1505.07818

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). *Generative Adversarial Networks*. http://arxiv.org/abs/1406.2661

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507. https://doi.org/10.1126/science.1127647

Hu, C., Hudson, S., Ethier, M., Al-Sharman, M., Rayside, D., & Melek, W. (2022). Sim-to-Real Domain Adaptation for Lane Detection and Classification in Autonomous Driving. *IEEE Intelligent Vehicles Symposium, Proceedings*, *2022-June*, 457–463. https://doi.org/10.1109/IV51971.2022.9827450

Kingma, D. P., & Welling, M. (2013). *Auto-Encoding Variational Bayes*. http://arxiv.org/abs/1312.6114

Kothandaraman, D., & Nambiar, A. (n.d.). *Domain Adaptive Knowledge Distillation for Driving Scene Semantic Segmentation*.

Li, J., Xu, R., Liu, X., Ma, J., Li, B., Zou, Q., Ma, J., & Yu, H. (2023). *Domain Adaptation based Object Detection for Autonomous Driving in Foggy and Rainy Weather*. http://arxiv.org/abs/2307.09676

Liu, X., Yoo, C., Xing, F., Oh, H., Fakhri, G. El, Kang, J.-W., & Woo, J. (2022). *Deep Unsupervised Domain Adaptation: A Review of Recent Advances and Perspectives*. http://arxiv.org/abs/2208.07422

Maierhofer, S., & Althoff, M. (n.d.). *CommonRoad-CARLA Interface: Bridging the Gap between Motion Planning and 3D Simulation*.

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., & Frey, B. (2015). *Adversarial Autoencoders*. http://arxiv.org/abs/1511.05644

Malik, S., Khan, M. A., Aadam, El-Sayed, H., Iqbal, F., Khan, J., & Ullah, O. (2023). CARLA+: An Evolution of the CARLA Simulator for Complex Environment Using a Probabilistic Graphical Model. *Drones*, *7*(2). https://doi.org/10.3390/drones7020111

Norrelykke, S. F., & Flyvbjerg, H. (2009). *Power spectrum analysis with least-squares fitting: Amplitude bias and its elimination, with application to optical tweezers and atomic force microscope cantilevers*. https://doi.org/10.1063/1.3455217

Pandey, P., Tyagi, A. K., Ambekar, S., & AP, P. (2020). *Unsupervised Domain Adaptation for Semantic Segmentation of NIR Images through Generative Latent Search*. http://arxiv.org/abs/2006.08696

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, & Pierre-Antoine Manzagol. (2010). Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion Pascal Vincent Hugo Larochelle Yoshua Bengio Pierre-Antoine Manzagol. In *Journal of Machine Learning Research* (Vol. 11).

Pier Paolo Ippolito. (2023, December 14). *Introduction to Autoencoders: From The Basics to Advanced Applications in PyTorch*. https://www.datacamp.com/tutorial/introduction-to-autoencoders

Ram Prabhakar, K., Vinod, V., Ranjan Sahoo, N., & Venkatesh Babu, R. (2021). *Few-Shot Domain Adaptation for Low Light RAW Image Enhancement*.

Saito, K., Watanabe, K., Ushiku, Y., & Harada, T. (2017). *Maximum Classifier Discrepancy for Unsupervised Domain Adaptation*. http://arxiv.org/abs/1712.02560

Sensors and data. (2024). https://carla.readthedocs.io/en/latest/core_sensors/

Singhal, P., Walambe, R., Ramanna, S., & Kotecha, K. (2023). Domain Adaptation: Challenges, Methods, Datasets, and Applications. *IEEE Access*, *11*, 6973–7020. https://doi.org/10.1109/ACCESS.2023.3237025

Sun, B., & Saenko, K. (2016). *Deep CORAL: Correlation Alignment for Deep Domain Adaptation*. http://arxiv.org/abs/1607.01719

The BrAIn Roads. (2024).

Town 7. (2024). https://carla.readthedocs.io/en/latest/map_town07/

Tzeng, E., Hoffman, J., Zhang, N., Saenko, K., & Darrell, T. (2014). *Deep Domain Confusion: Maximizing for Domain Invariance*. http://arxiv.org/abs/1412.3474

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. (2008). Extracting and composing robust features with denoising autoencoders. *Proceedings of the 25th International Conference on Machine Learning*, 1096–1103. https://doi.org/10.1145/1390156.1390294

Yoo, J., Hong, Y., Noh, Y., & Yoon, S. (2017). *Domain Adaptation Using Adversarial Learning for Autonomous Navigation*. http://arxiv.org/abs/1712.03742

Zhou, Y., Song, X., Zhang, Y., Liu, F., Zhu, C., & Liu, L. (2022). Feature Encoding With Autoencoders for Weakly Supervised Anomaly Detection. *IEEE Transactions on Neural*

Networks and Learning Systems, *33*(6), 2454–2465. https://doi.org/10.1109/TNNLS.2021.3086137

Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2017). *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*. http://arxiv.org/abs/1703.10593

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., & He, Q. (2021). A Comprehensive Survey on Transfer Learning. In *Proceedings of the IEEE* (Vol. 109, Issue 1, pp. 43–76). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/JPROC.2020.3004555

APPENDICES

APPENDIX A

CIAE vs. SOAE MODEL RESULTS (EPOCH=100)

The CIAE and SOAE model results in terms of reconstruction loss and domain classifier results have been shown in APPENDIX A for 100 epochs.

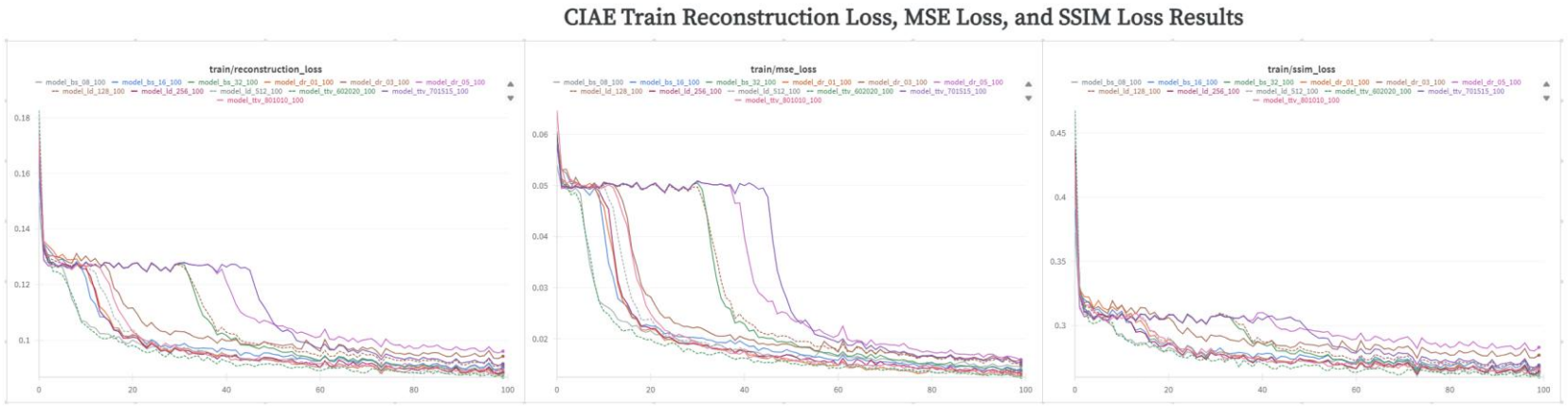

Figure 6: CIAE train reconstruction loss, MSE loss, and SSIM loss results for 100 epochs

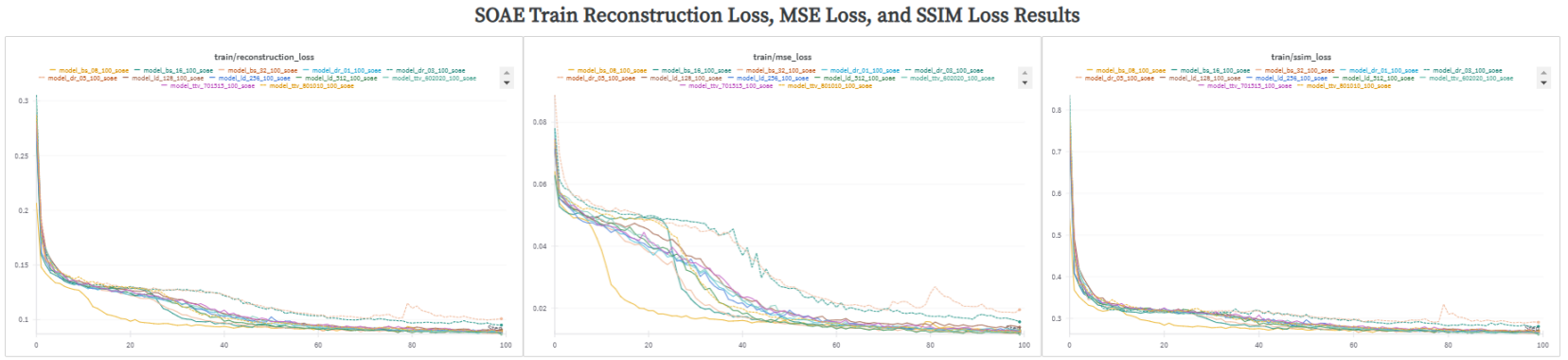

Figure 7: SOAE train reconstruction loss, MSE loss, and SSIM loss results for 100 epochs

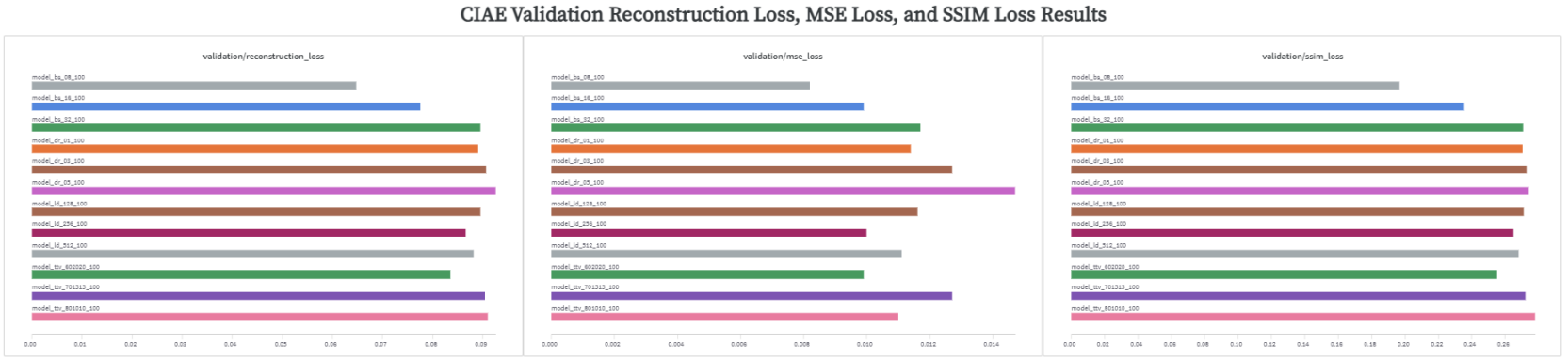

Figure 8: CIAE validation reconstruction loss, MSE loss, and SSIM loss results for 100 epochs

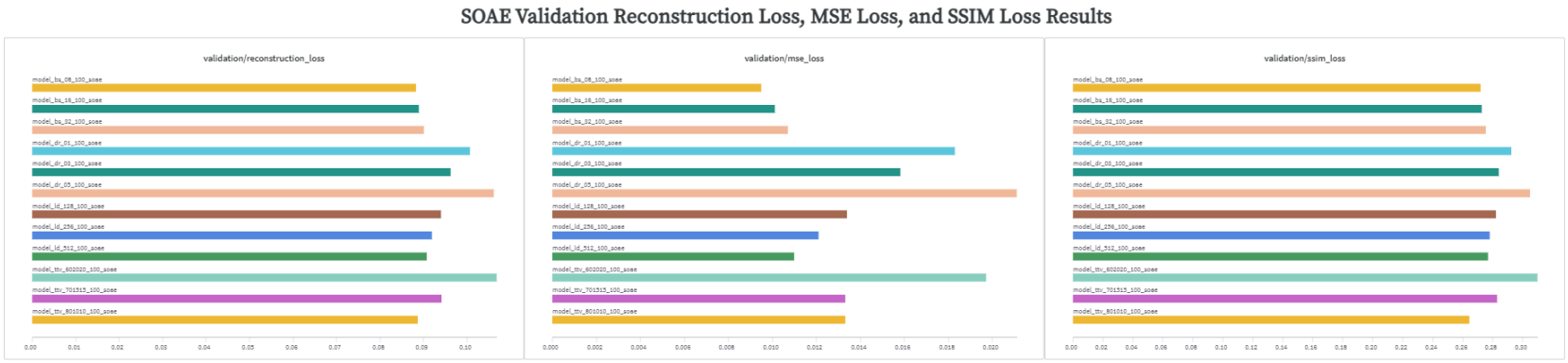

Figure 9: SOAE validation reconstruction loss, MSE loss, and SSIM loss results for 100 epochs

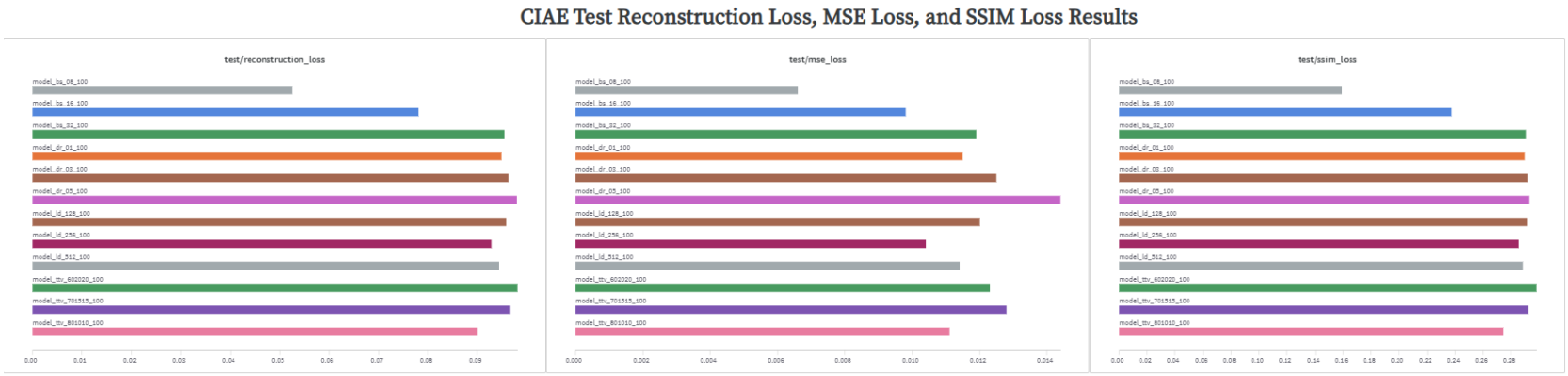

Figure 10: CIAE test reconstruction loss, MSE loss, and SSIM loss results for 100 epochs

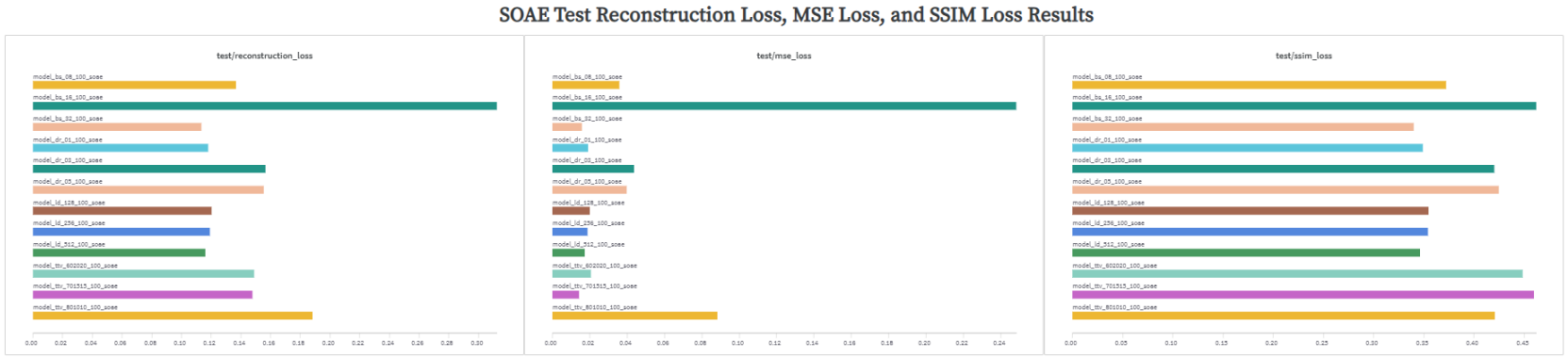

Figure 11: SOAE test reconstruction loss, MSE loss, and SSIM loss results for 100 epochs

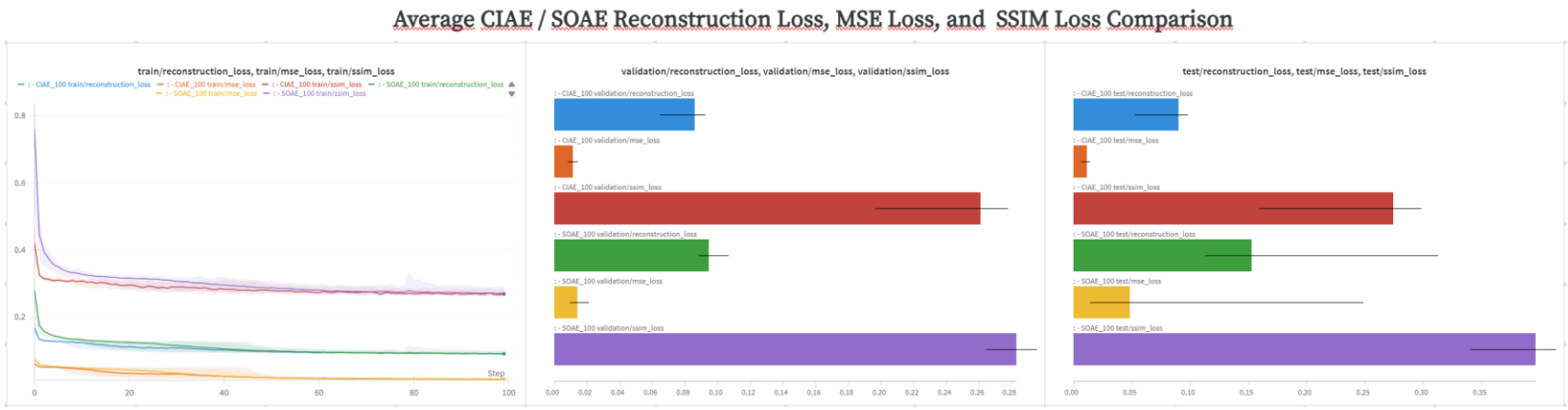

Figure 12: Average CIAE/SOAE reconstruction loss, MSE loss, and SSIM loss comparison for 100 epochs

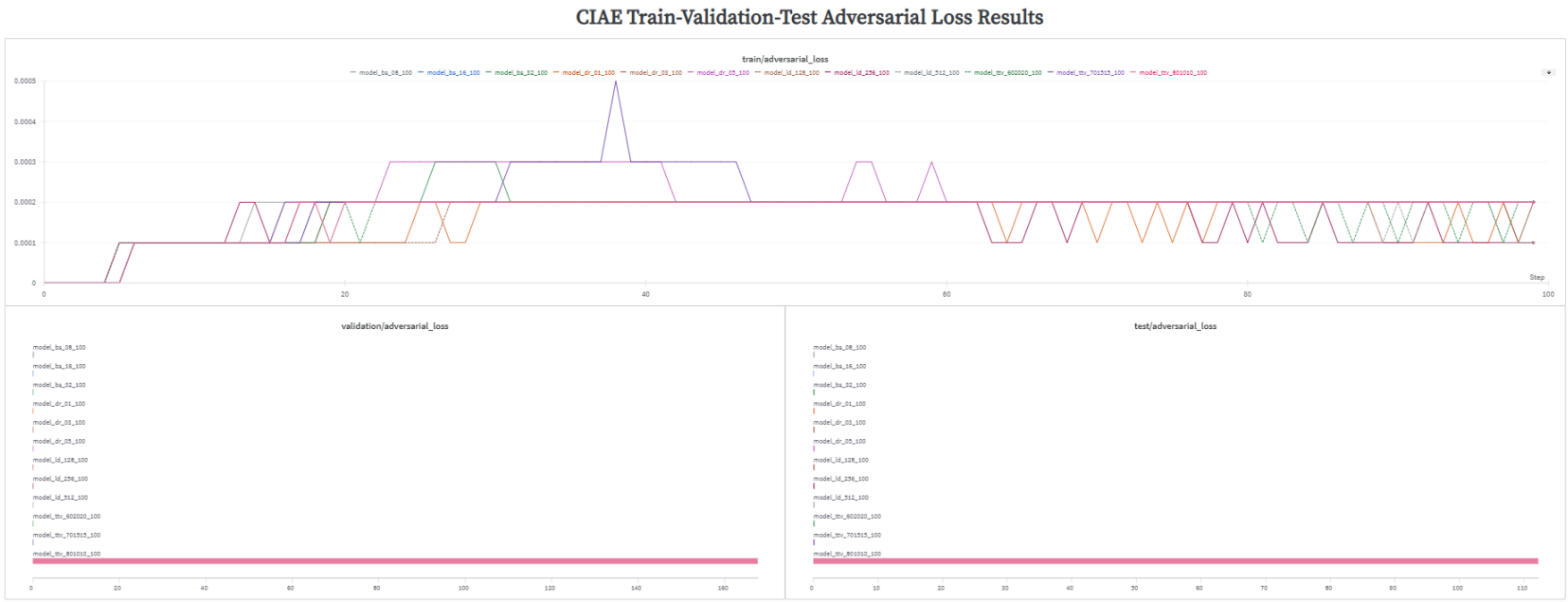

Figure 13: CIAE train-validation-test adversarial loss results for 100 epochs

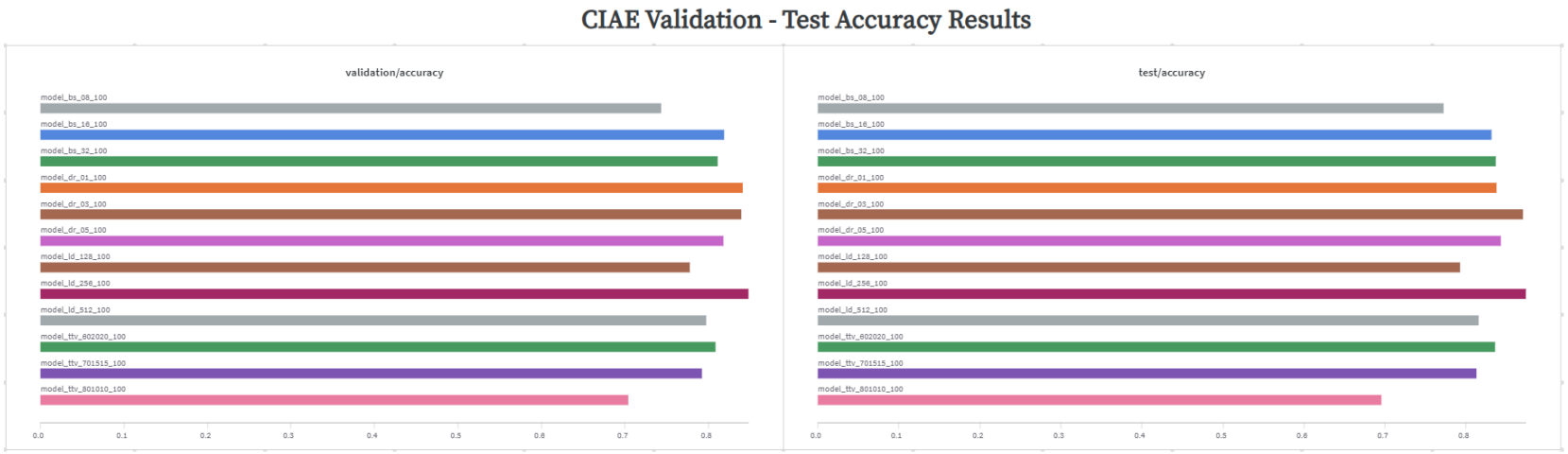

Figure 14: CIAE validation - test accuracy results for 100 epochs

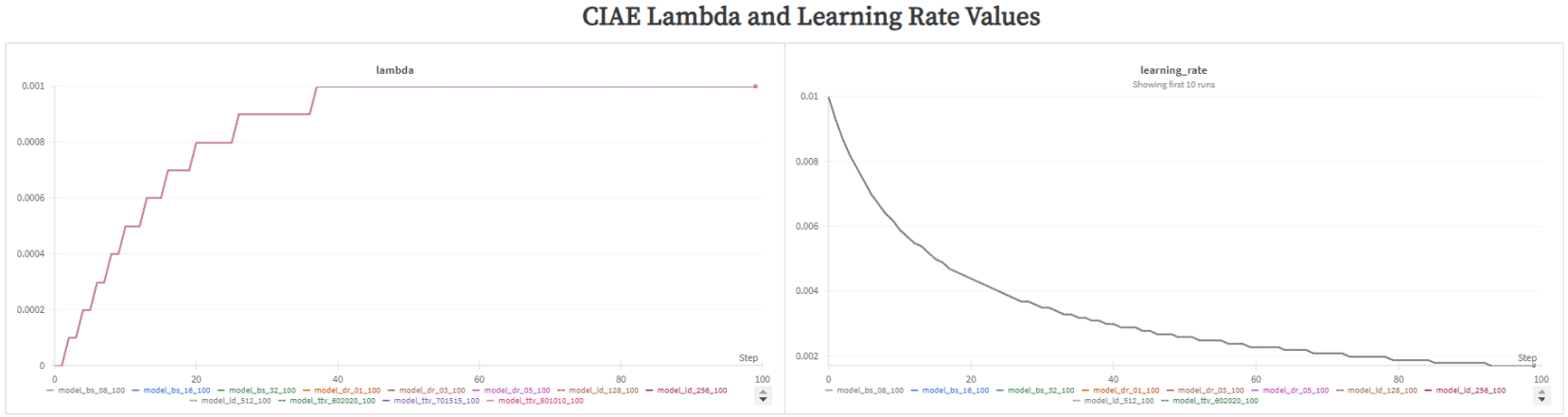

Figure 15: CIAE lambda and learning rate values for 100 epochs

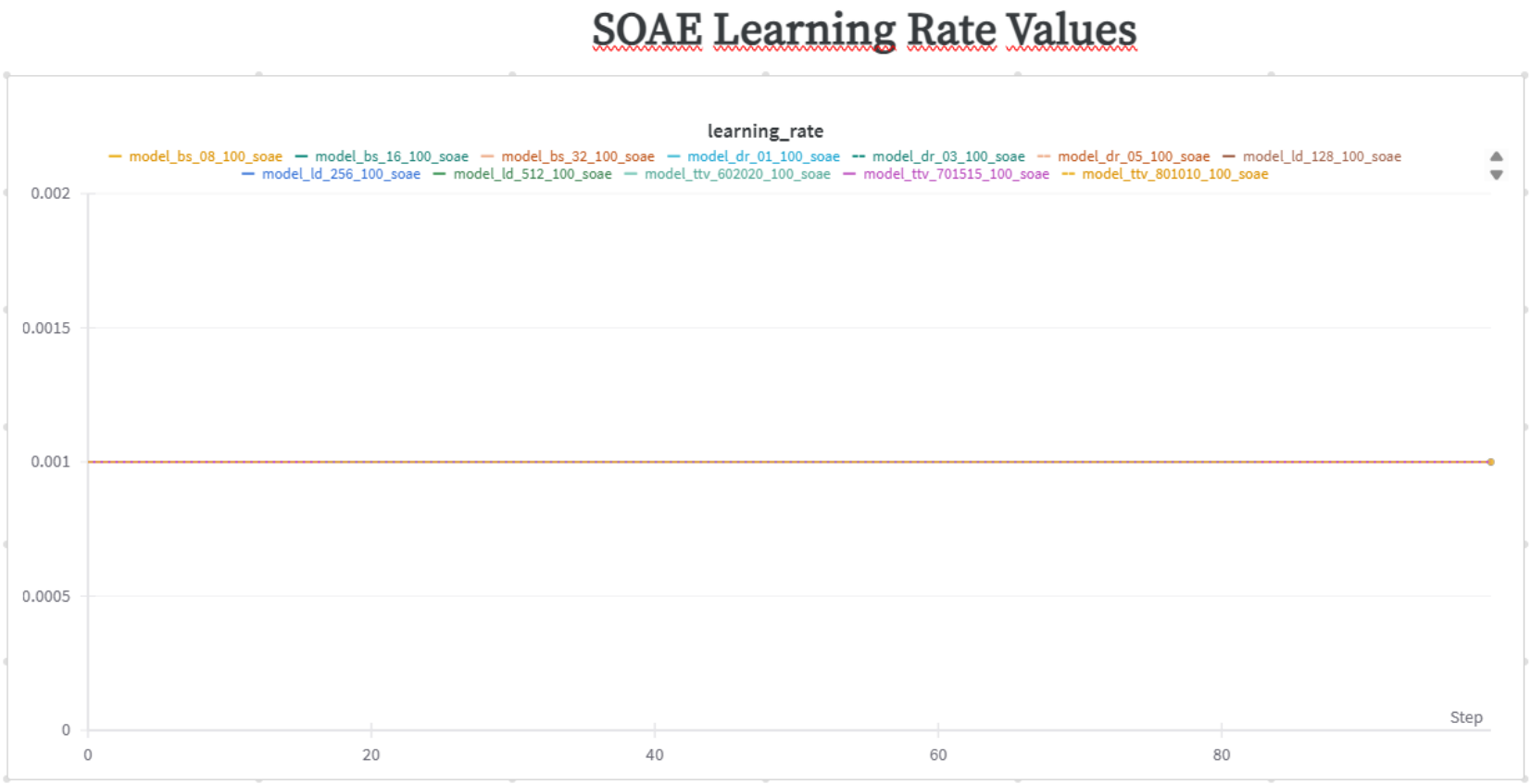

Figure 16: SOAE learning rate values for 100 epochs

APPENDIX B

CIAE vs. SOAE MODEL RESULTS (EPOCH=500)

The CIAE and SOAE model results in terms of reconstruction loss and domain classifier results have been shown in APPENDIX B for 500 epochs.

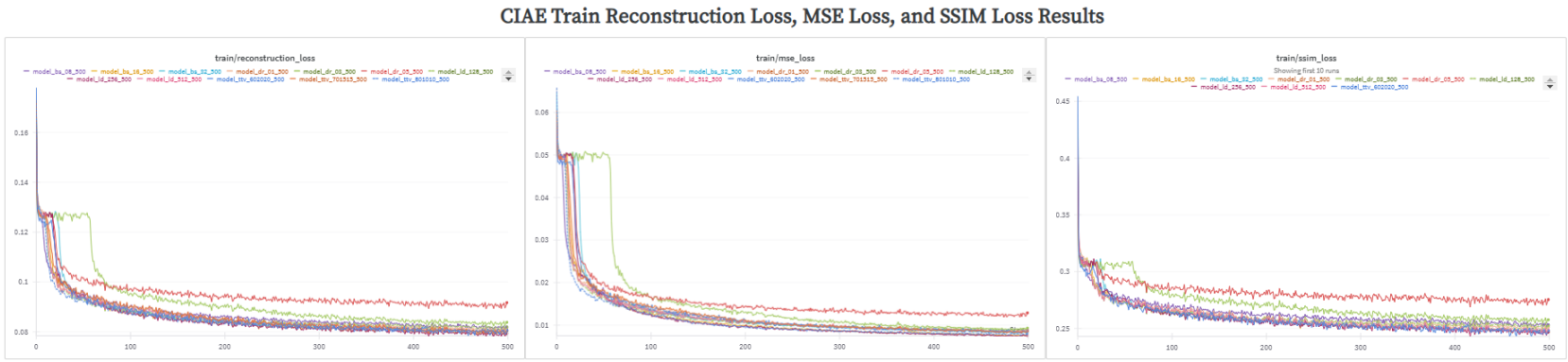

Figure 17: CIAE train reconstruction loss, MSE loss, and SSIM loss results for 500 epochs

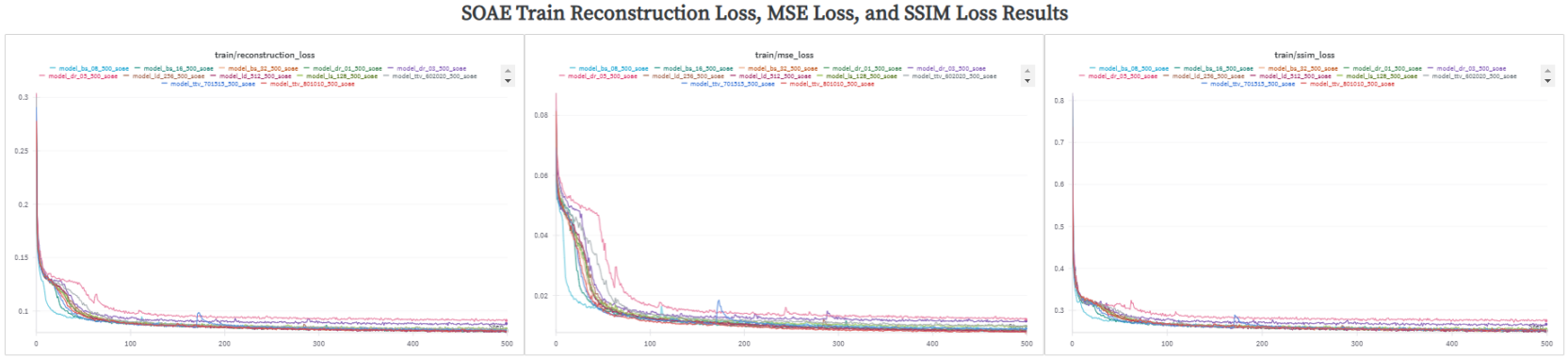

Figure 18: SOAE train reconstruction loss, MSE loss, and SSIM loss results for 500 epochs

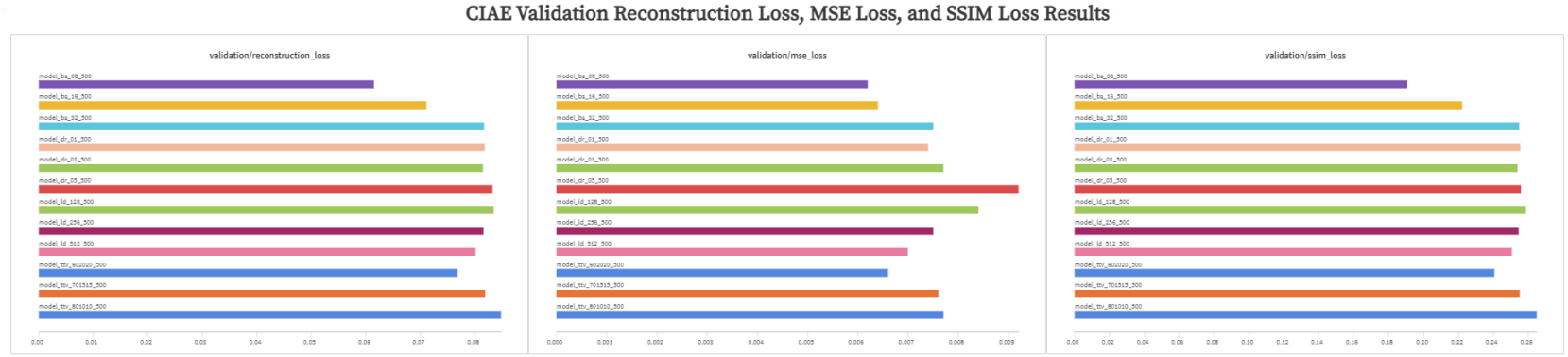

Figure 19: CIAE validation reconstruction loss, MSE loss, and SSIM loss results for 500 epochs

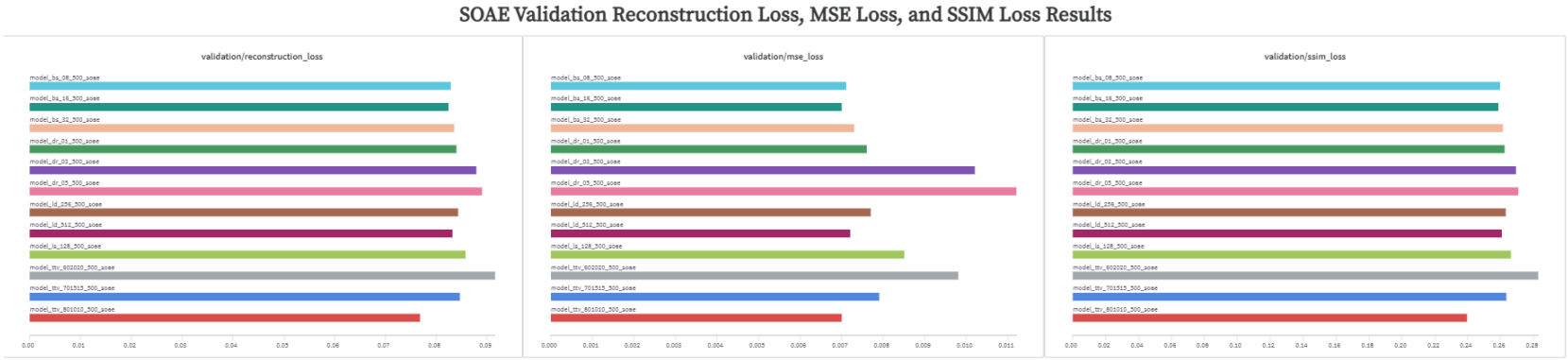

Figure 20: SOAE validation reconstruction loss, MSE loss, and SSIM loss results for 500 epochs

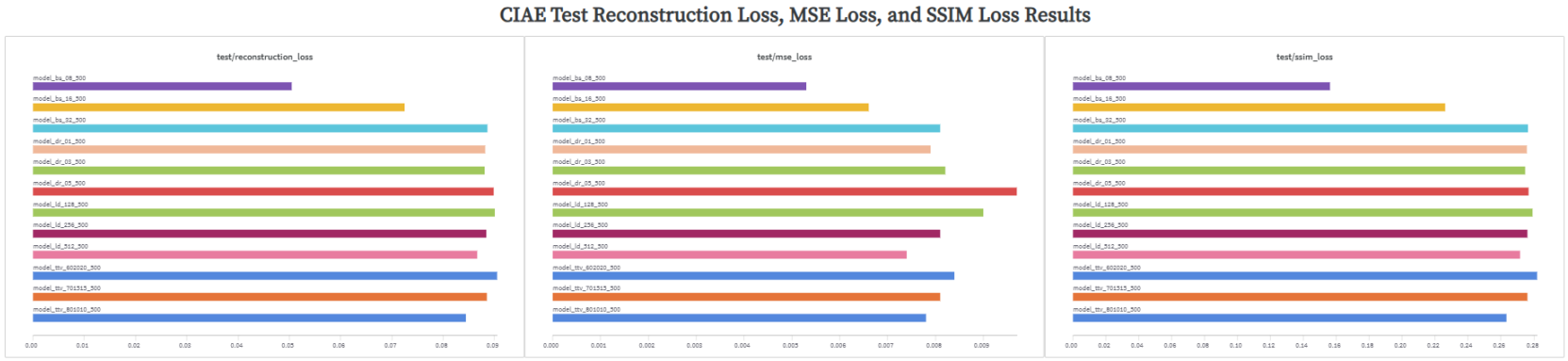

Figure 21: CIAE test reconstruction loss, MSE loss, and SSIM loss results for 500 epochs

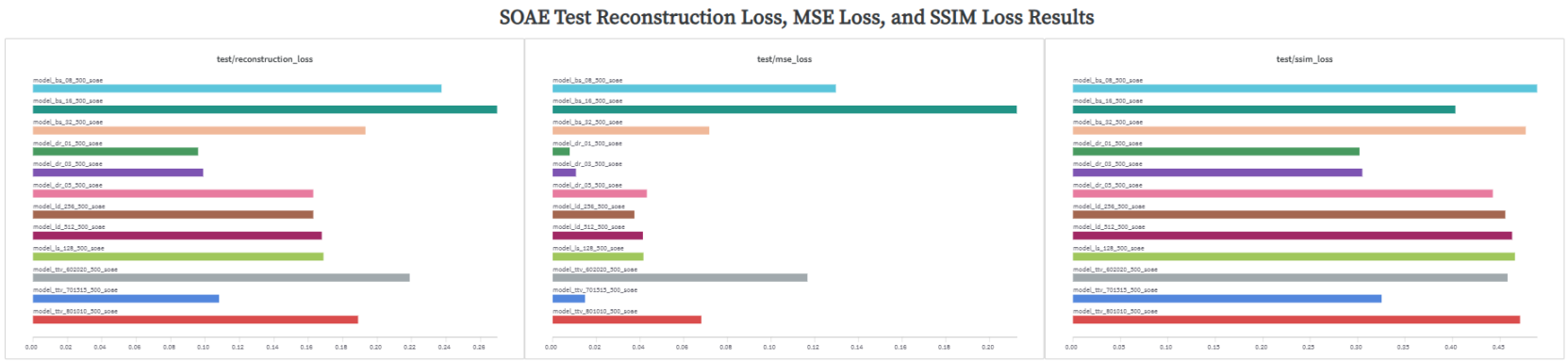

Figure 22: SOAE test reconstruction loss, MSE loss, and SSIM loss results for 500 epochs

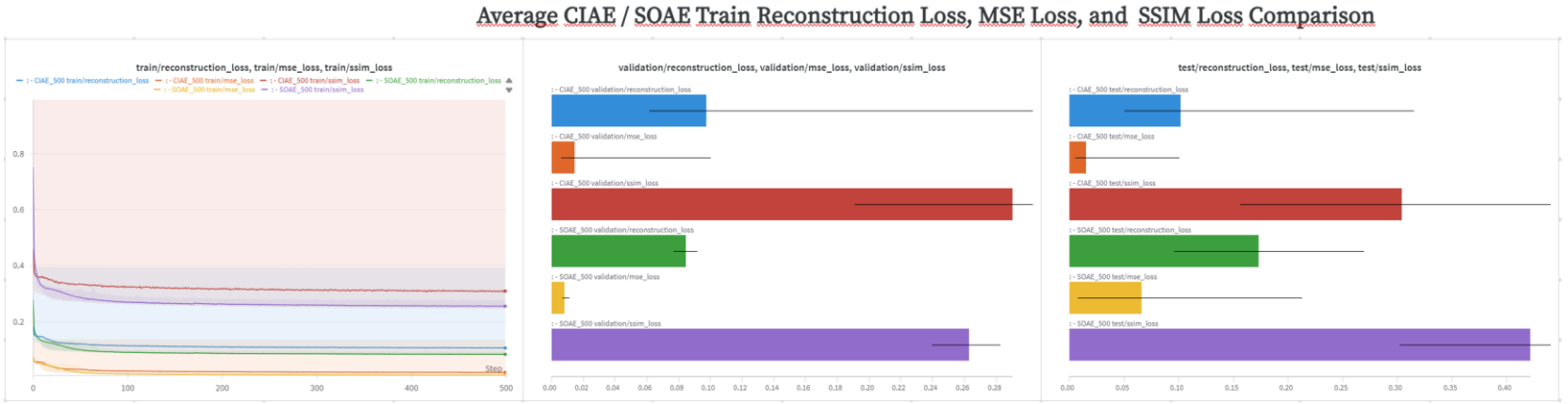

Figure 23: Average CIAE/SOAE reconstruction loss, MSE loss, and SSIM loss comparison for 500 epochs

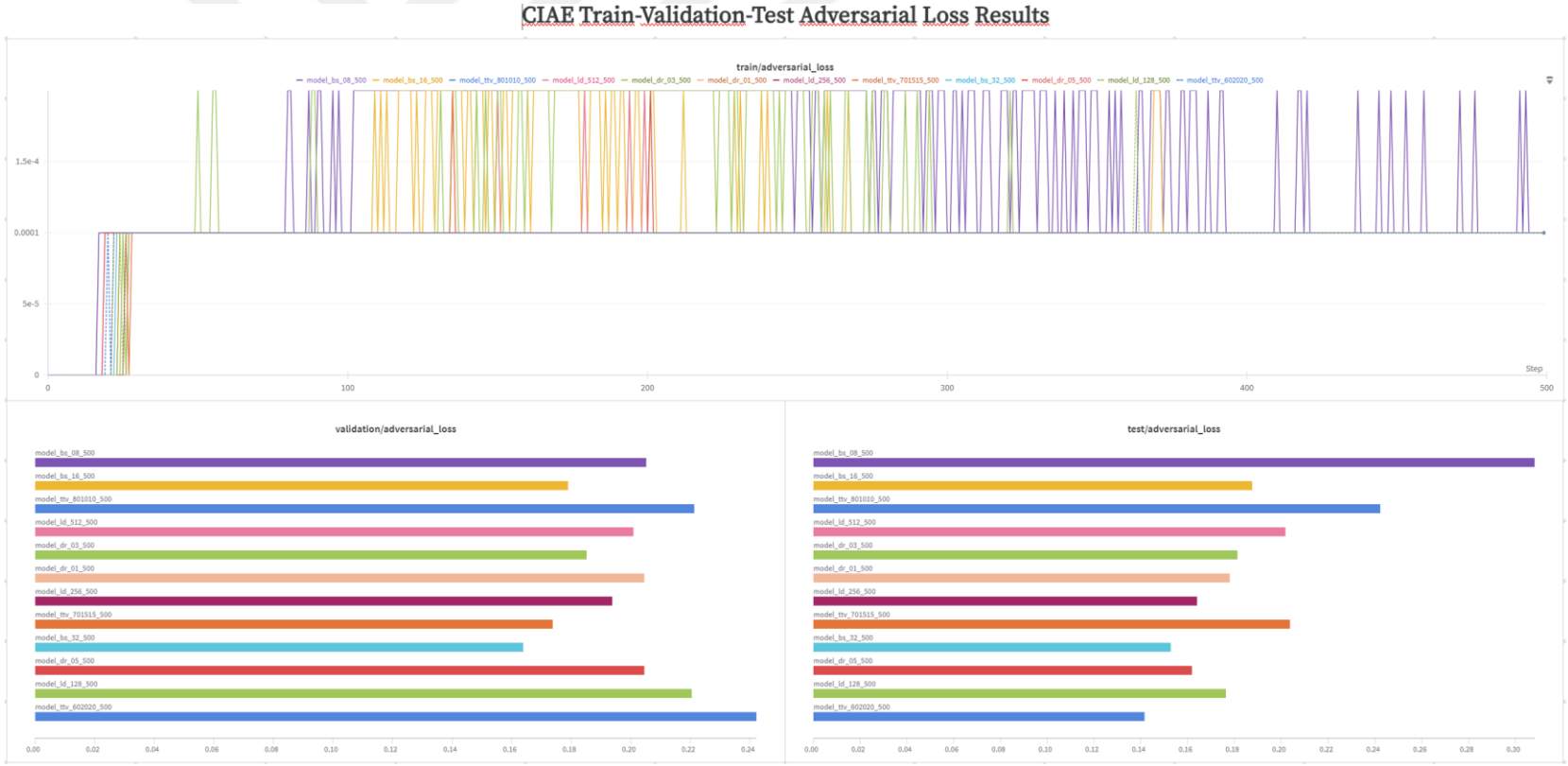

Figure 24: CIAE train-validation-test adversarial loss results for 500 epochs

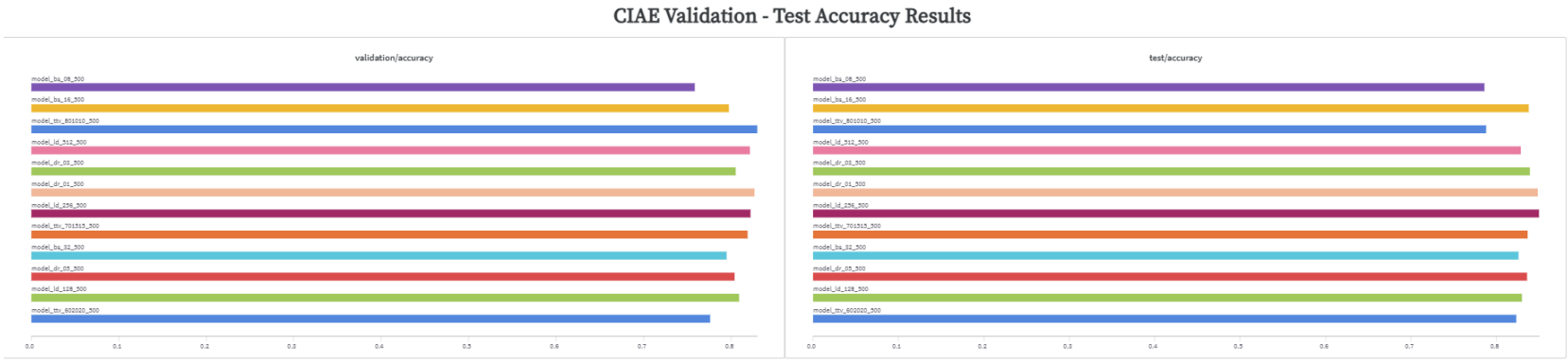

Figure 25: CIAE validation-test accuracy results for 500 epochs

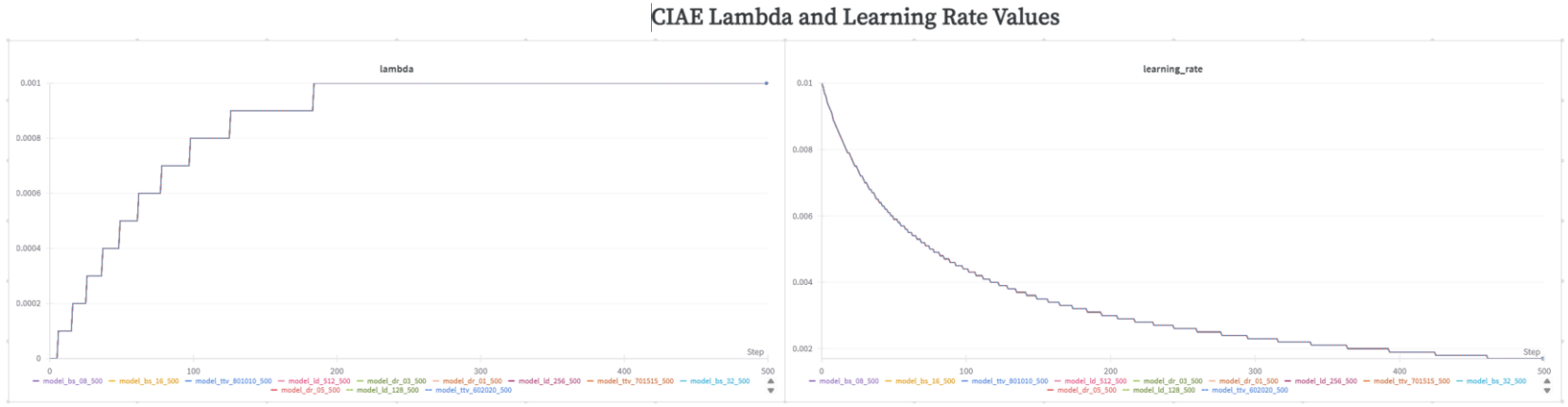

Figure 26: CIAE lambda and learning rate values for 500 epochs

SOAE Learning Rate Values



Figure 27: SOAE learning rate values for 500 epochs

APPENDIX C

CIAE vs. SOAE MODEL RESULTS (EPOCH=1000)

The CIAE and SOAE model results in terms of reconstruction loss and domain classifier results have been shown in APPENDIX C for 1000 epochs.



Figure 28: CIAE train reconstruction loss, MSE loss, and SSIM loss results for 1000 epochs

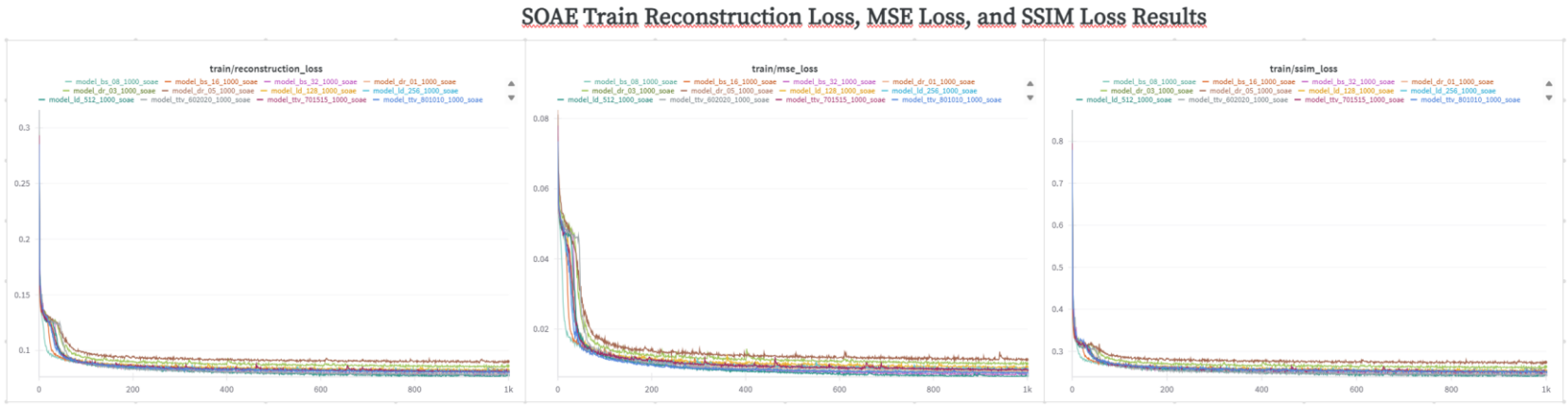

Figure 29: SOAE train reconstruction loss, MSE loss, and SSIM loss results for 1000 epochs

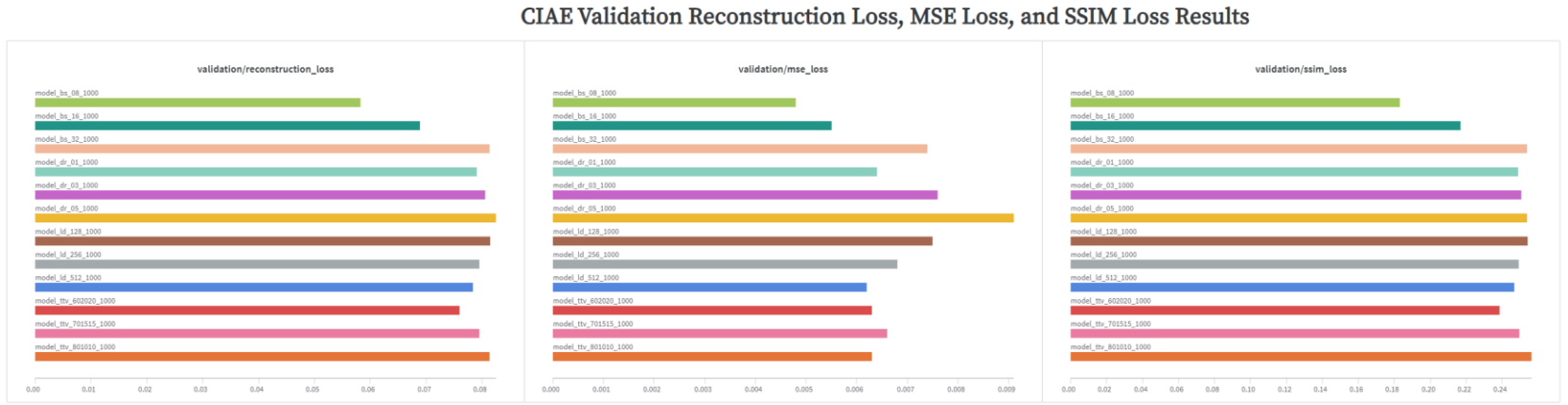

Figure 30: CIAE validation reconstruction loss, MSE loss, and SSIM loss results for 1000 epochs

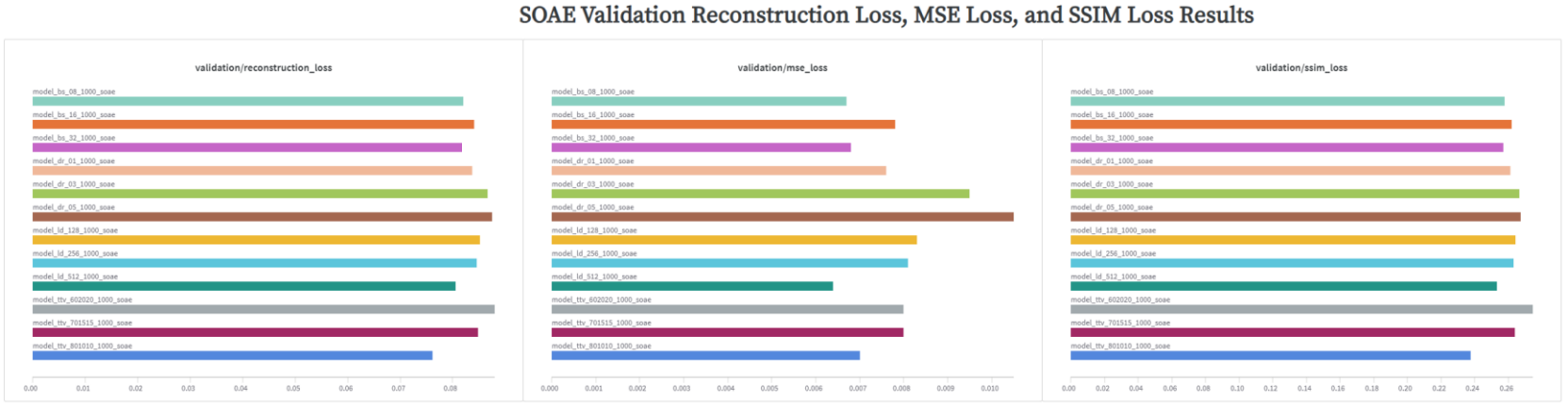

Figure 31: SOAE validation reconstruction loss, MSE loss, and SSIM loss results for 1000 epochs

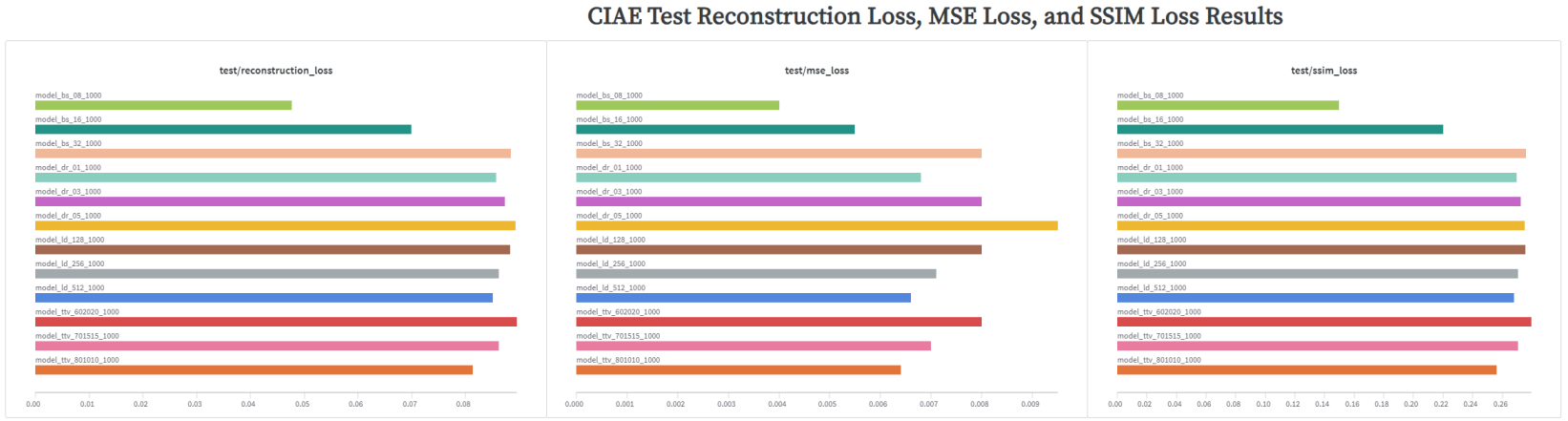

Figure 32: CIAE test reconstruction loss, MSE loss, and SSIM loss results for 1000 epochs

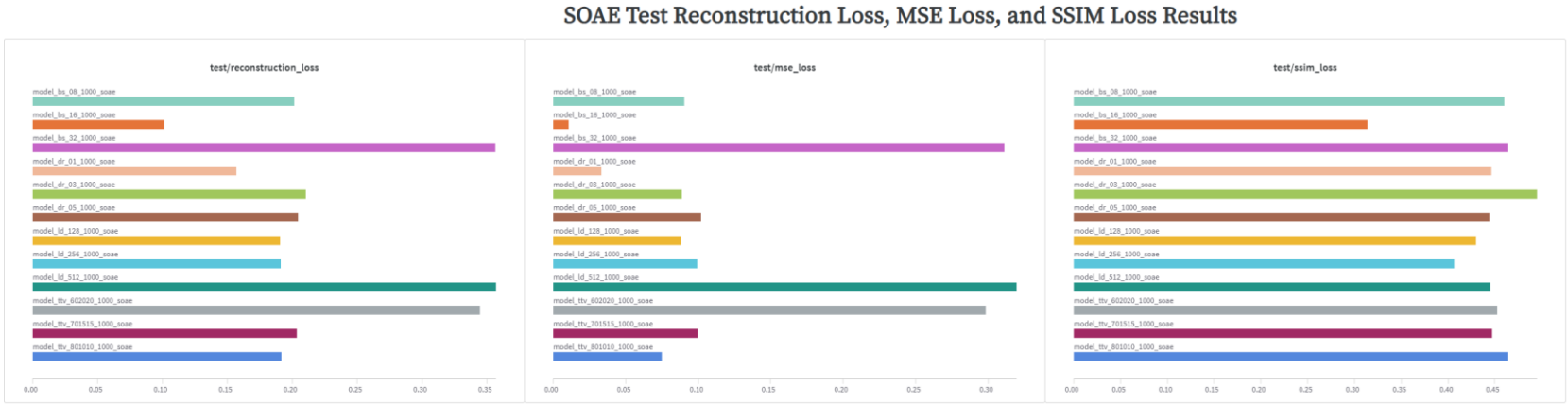

Figure 33: SOAE test reconstruction loss, MSE loss, and SSIM loss results for 1000 epochs

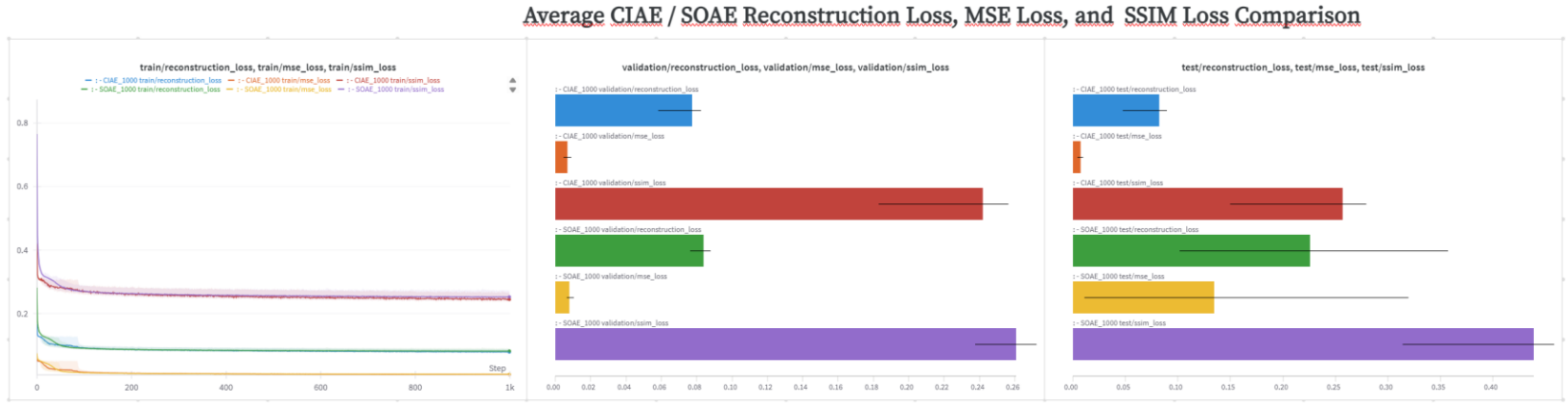

Figure 34: Average CIAE/SOAE reconstruction loss, MSE loss, and SSIM loss comparison for 1000 epochs

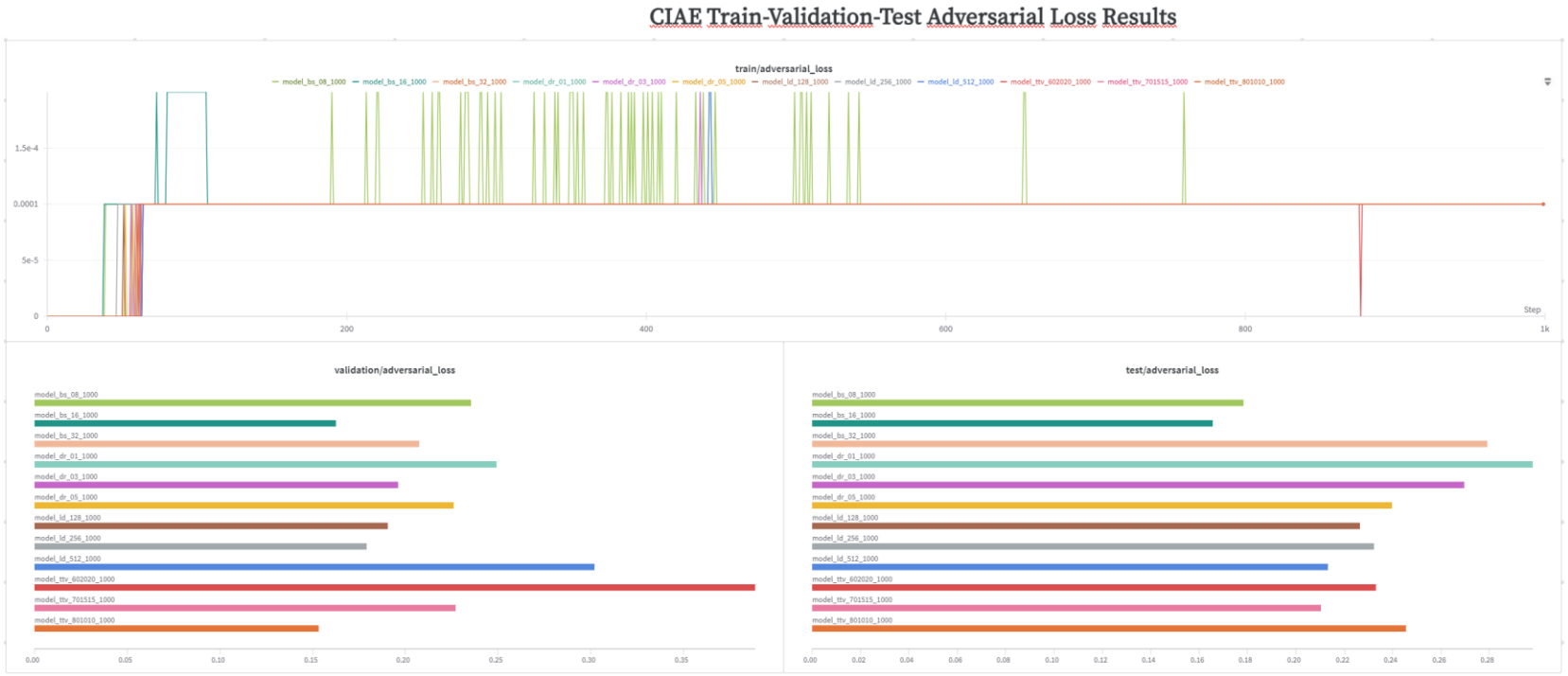

Figure 35: CIAE train-validation-test adversarial loss results for 1000 epochs

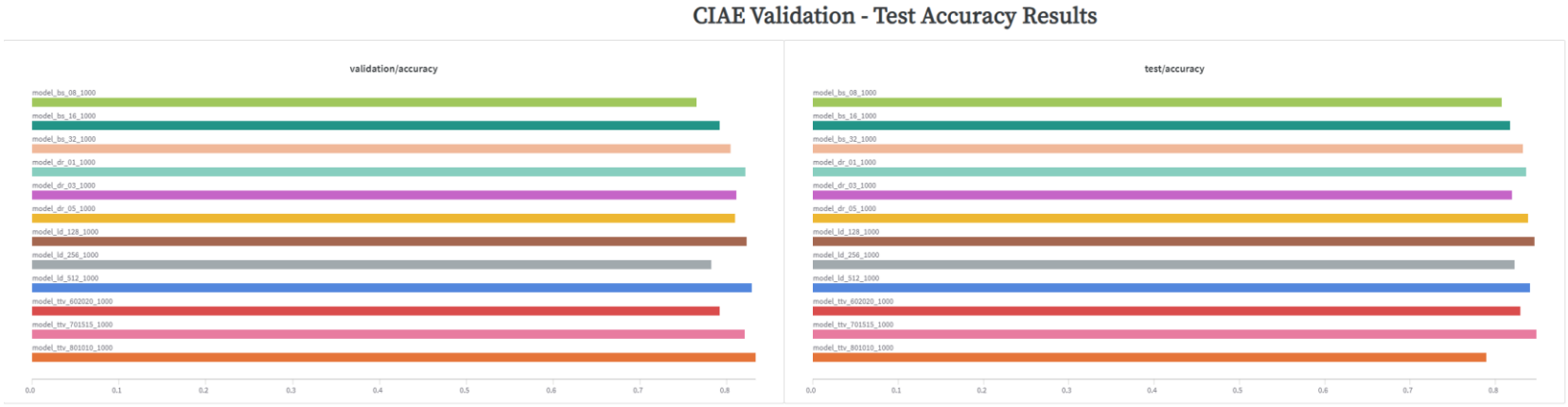

Figure 36: CIAE validation-test accuracy results for 1000 epochs

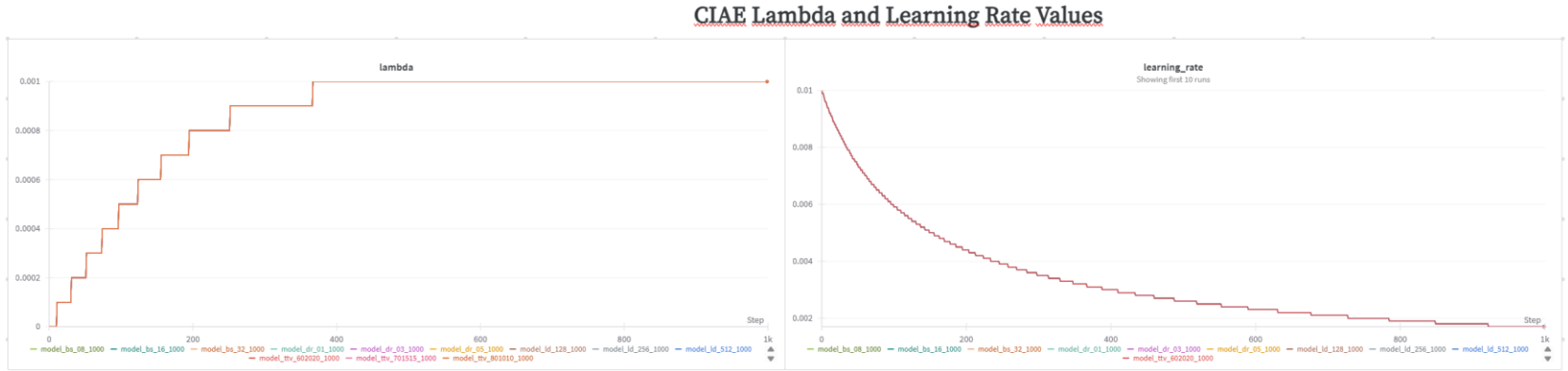

Figure 37: CIAE lambda and learning rate values for 1000 epochs

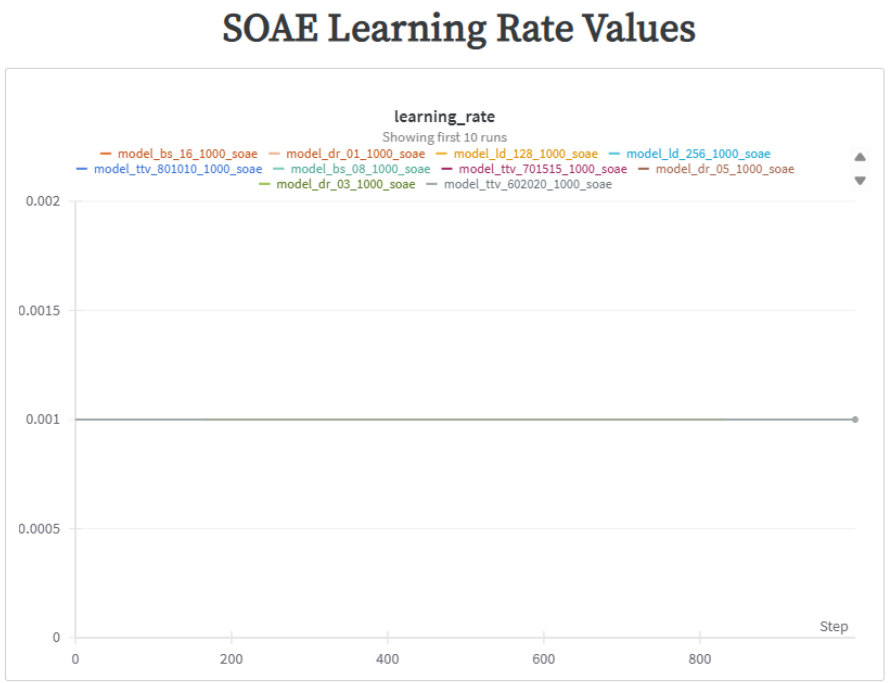

Figure 38: SOAE learning rate values for 1000 epochs

APPENDIX D

CIAE MODEL OUTPUTS (Epoch = 100)

The CIAE model outputs have been shown in APPENDIX D for 100 epochs.

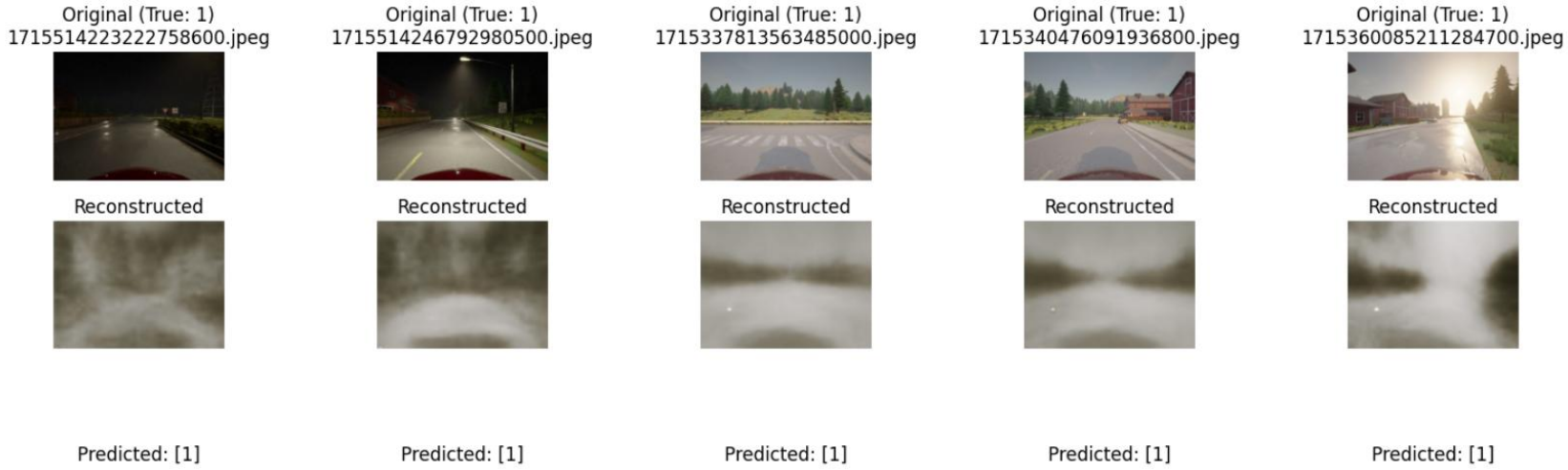

Figure 39: CIAE_model_bs_08_100

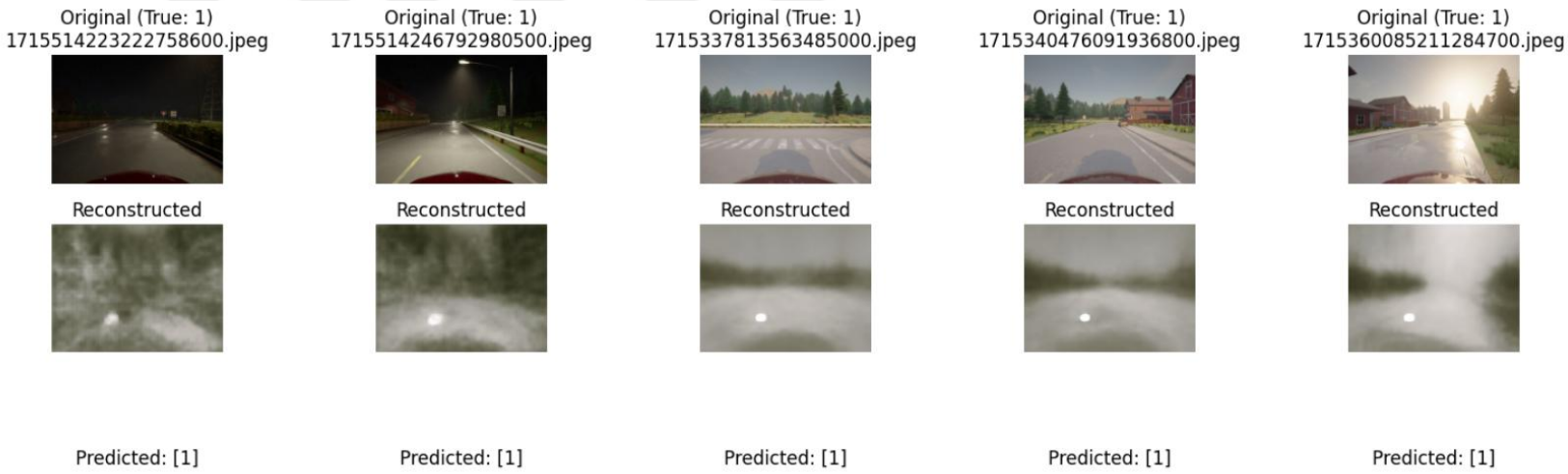

Figure 40: CIAE_model_bs_16_100

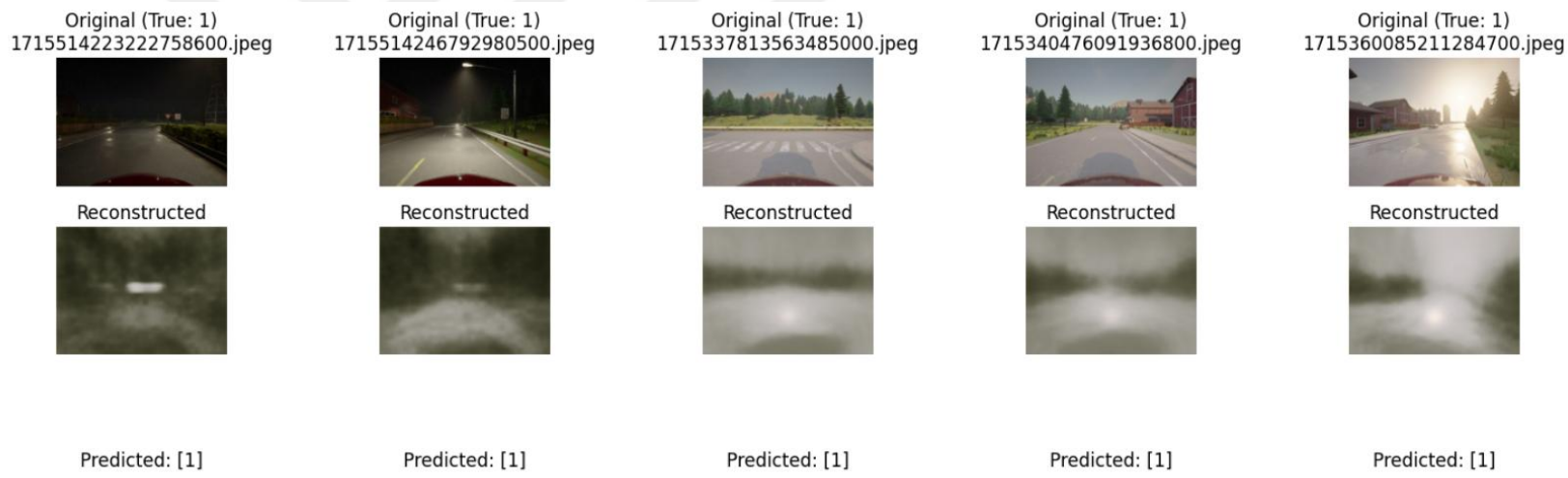

Figure 41: CIAE_model_bs_32_100

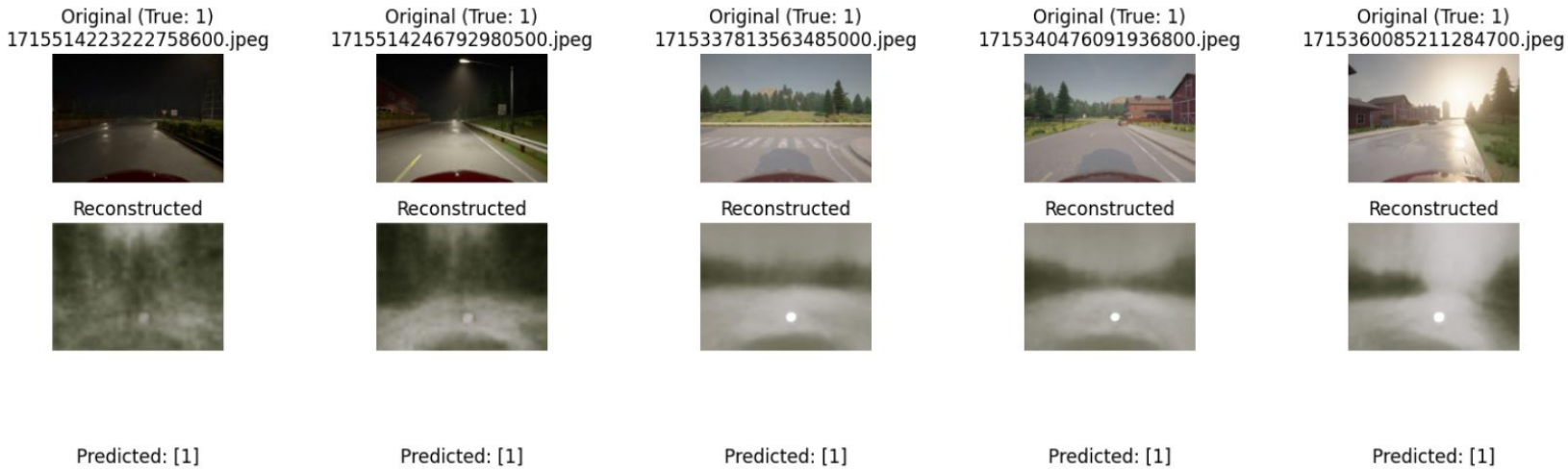

Figure 42: CIAE_model_dr_01_100

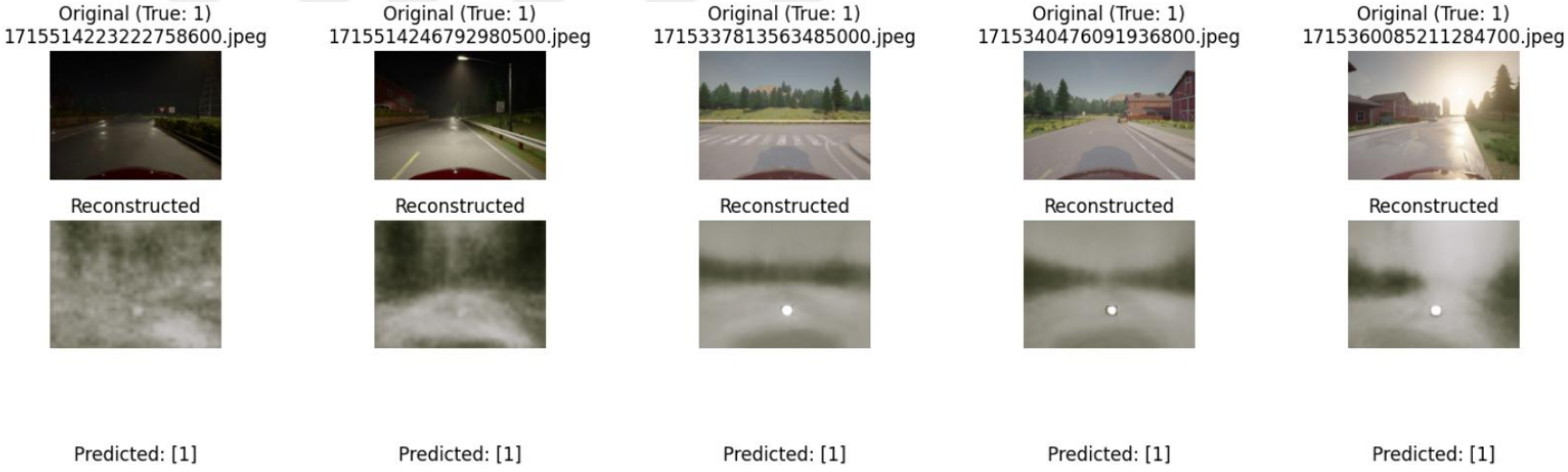

Figure 43: CIAE_model_dr_03_100

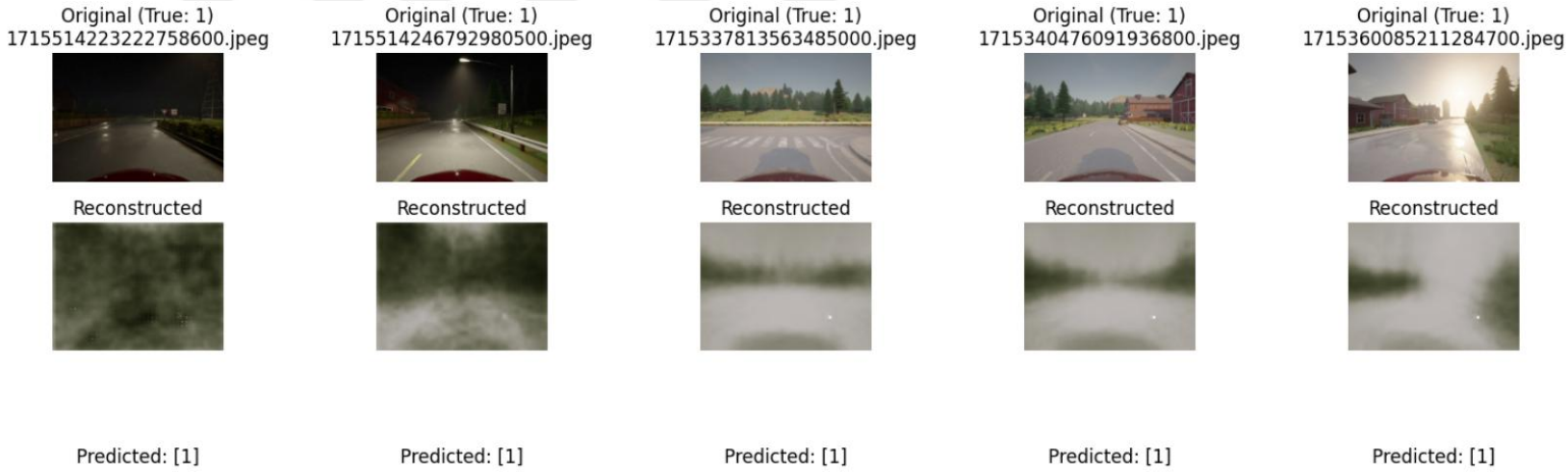

Figure 44: CIAE_model_dr_05_100

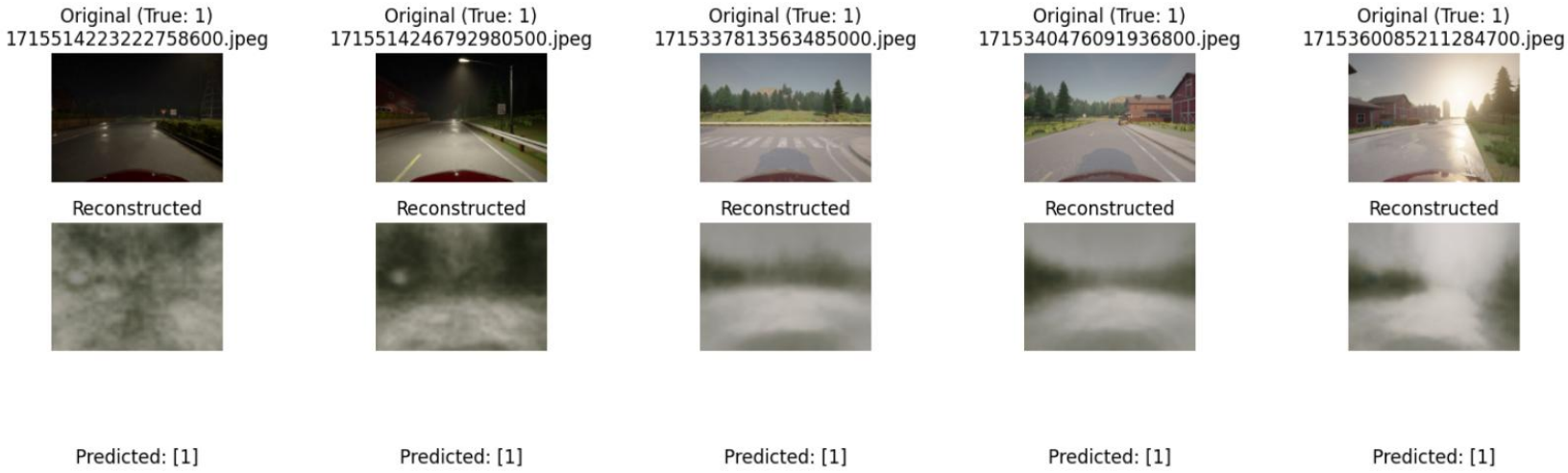

Figure 45: CIAE_model_ld_128_100

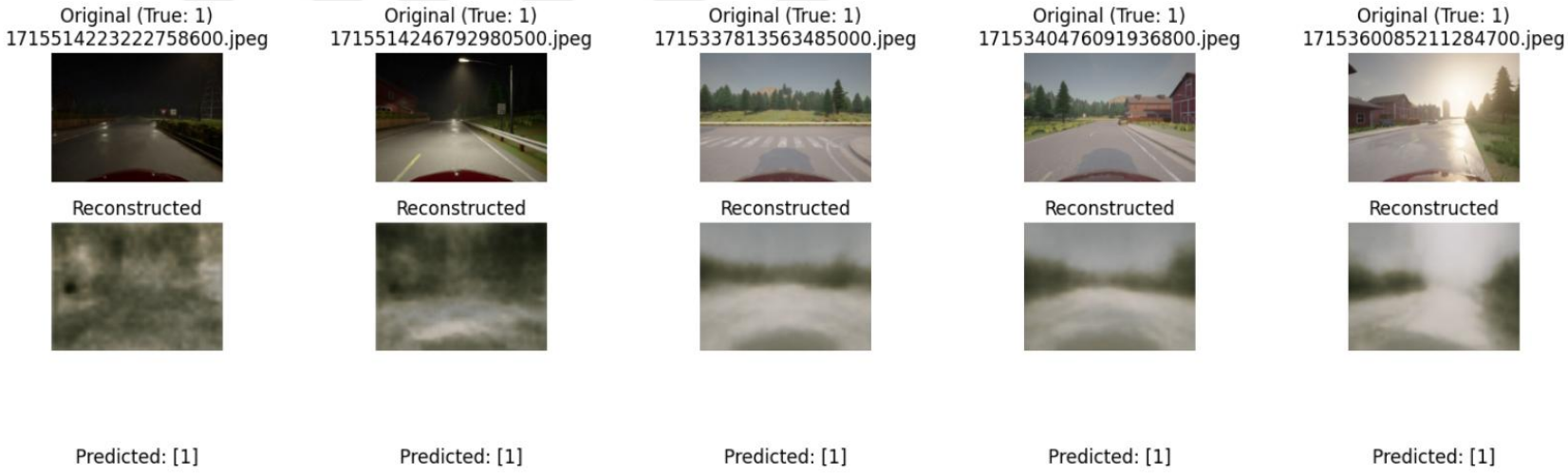

Figure 46: CIAE_model_ld_256_100

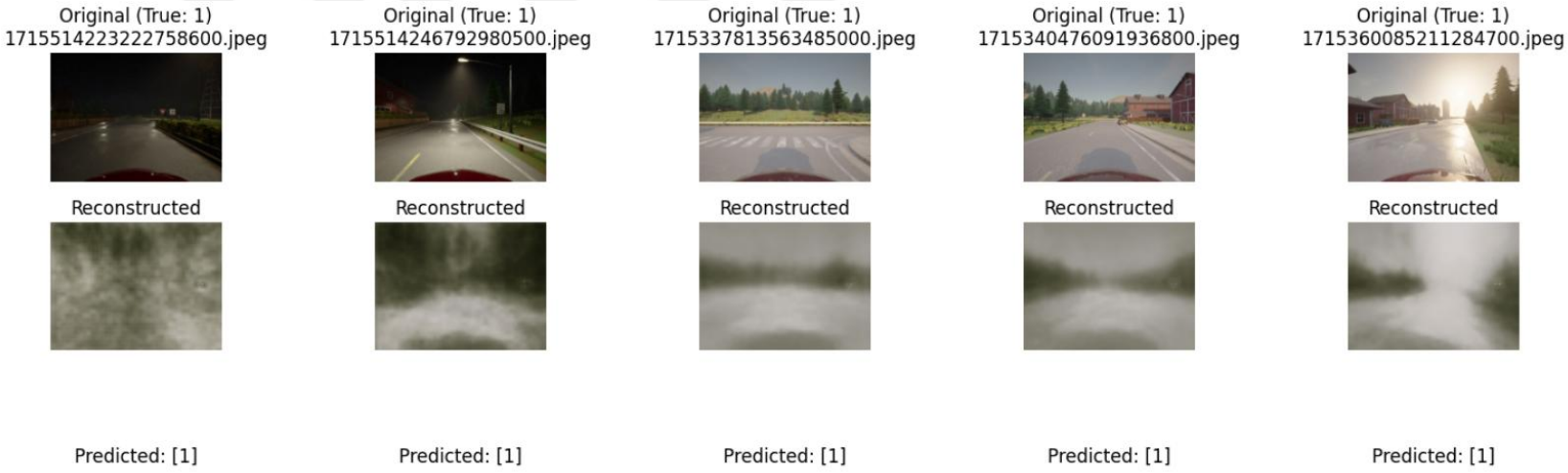

Figure 47: CIAE_model_ld_512_100

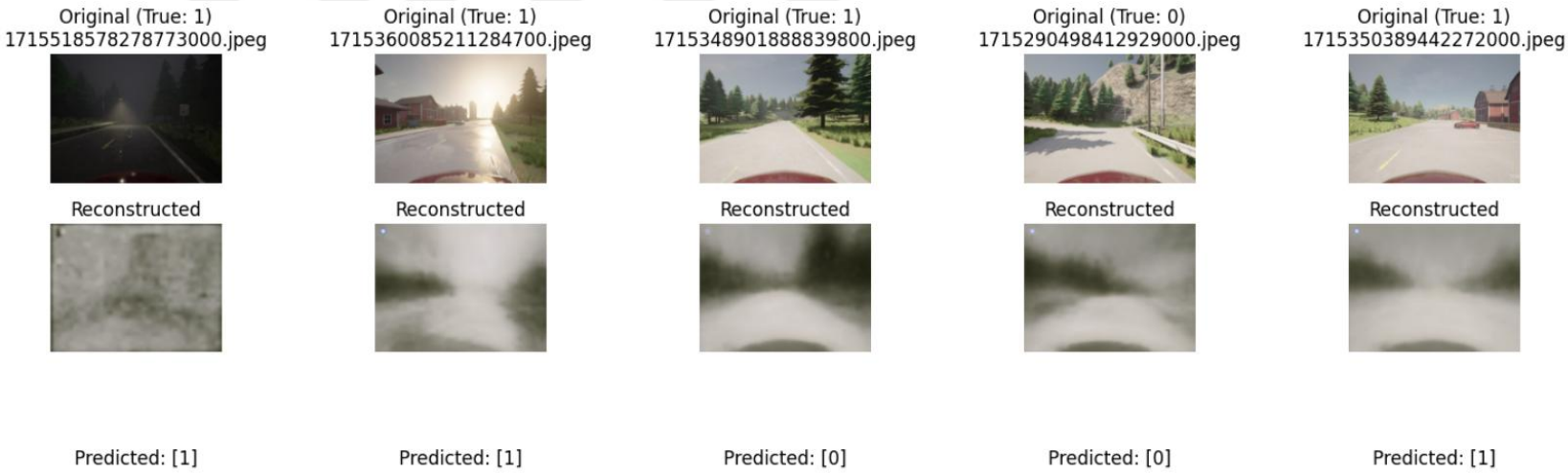

Figure 48: CIAE_model_ttv_602020_100

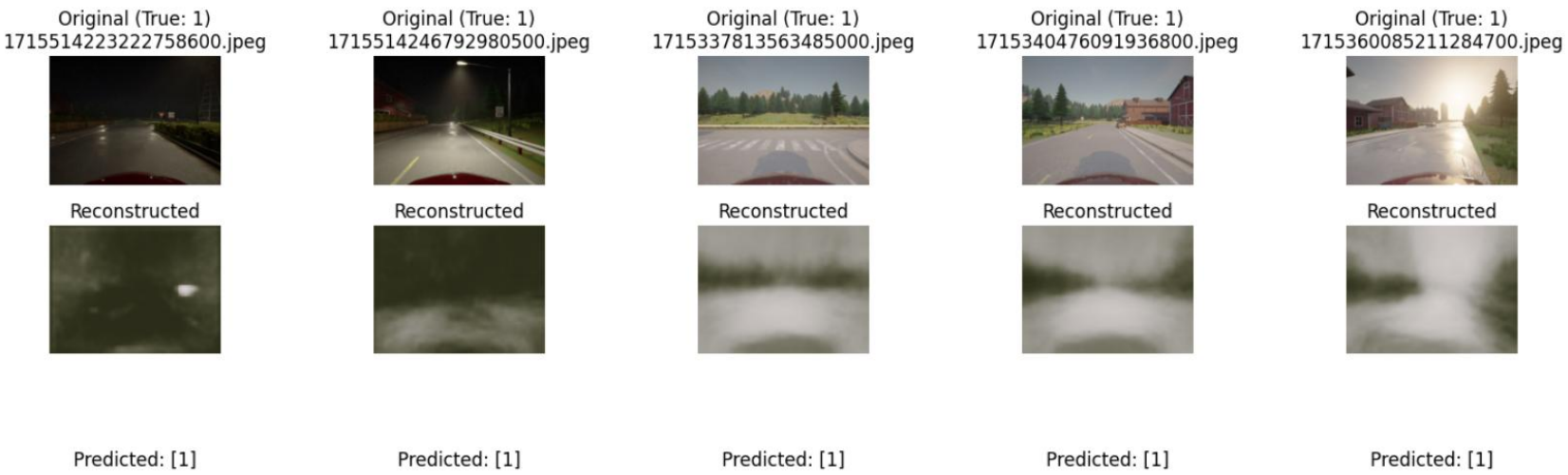

Figure 49: CIAE_model_ttv_701515_100

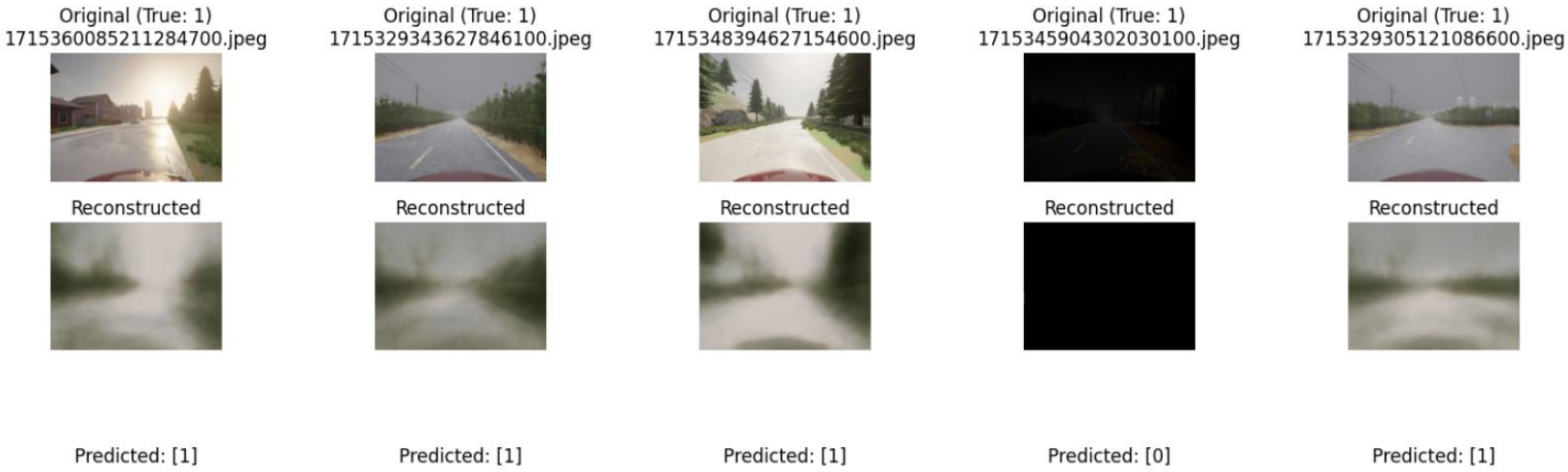

Figure 50: CIAE_model_ttv_801010_100

APPENDIX E

SOAE MODEL OUTPUTS (Epoch = 100)

The SOAE model outputs have been shown in APPENDIX E for 100 epochs.



Figure 51: SOAE_model_bs_08_100



Figure 52: SOAE_model_bs_16_100

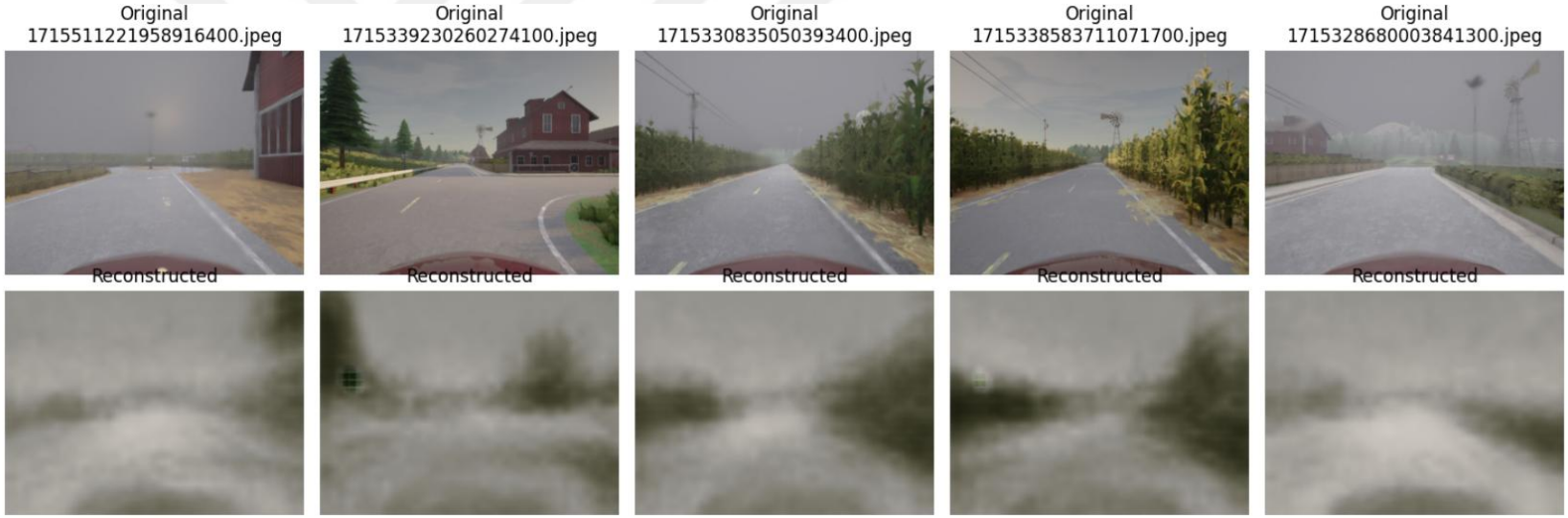

Figure 53: SOAE_model_bs_32_100



Figure 54: SOAE_model_dr_01_100

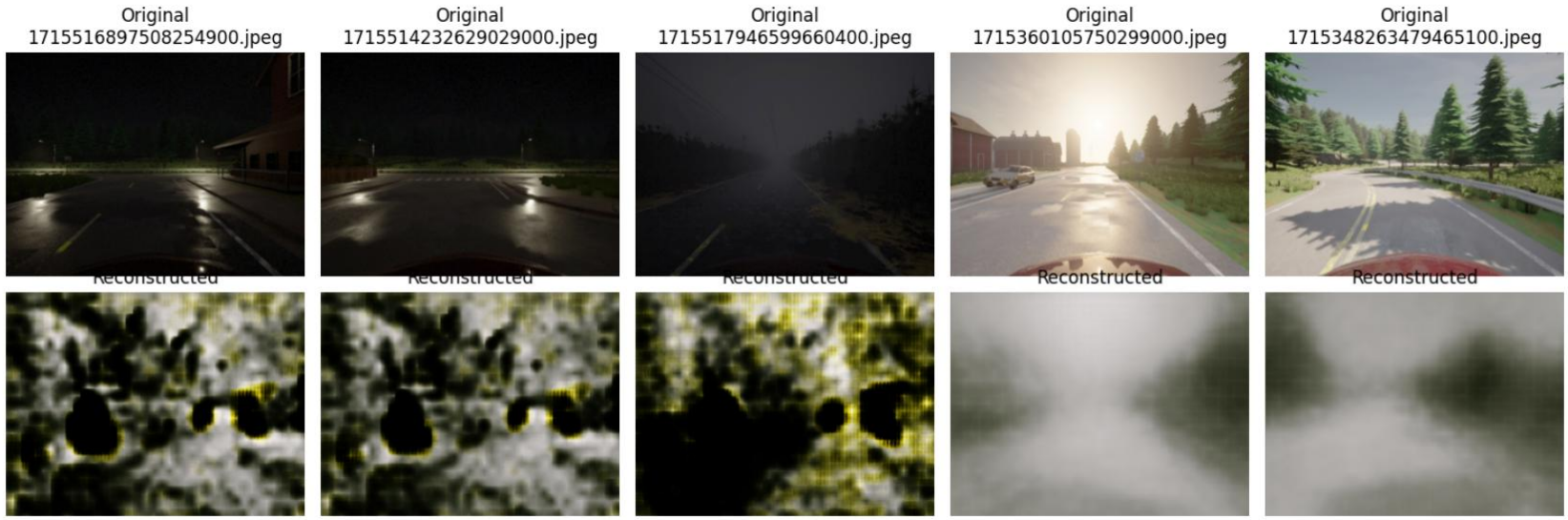

Figure 55: SOAE_model_dr_03_100



Figure 56: SOAE_model_dr_05_100



Figure 57: SOAE_model_ld_128_100

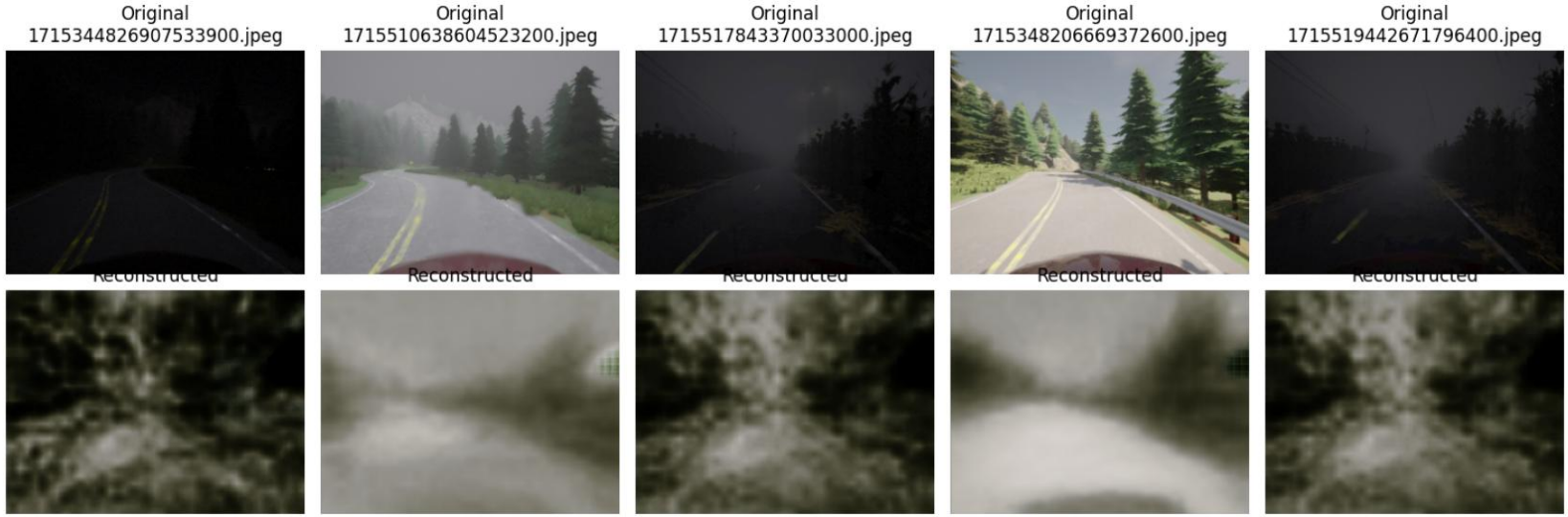

Figure 58: SOAE_model_ld_256_100

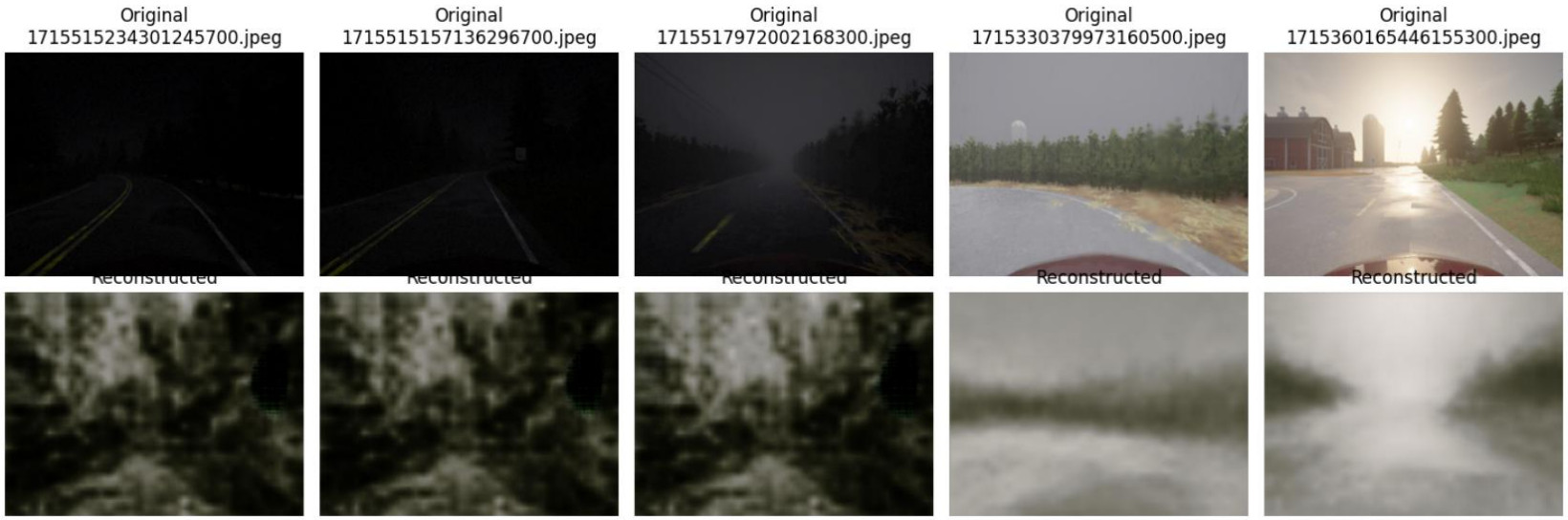

Figure 59: SOAE_model_ld_512_100

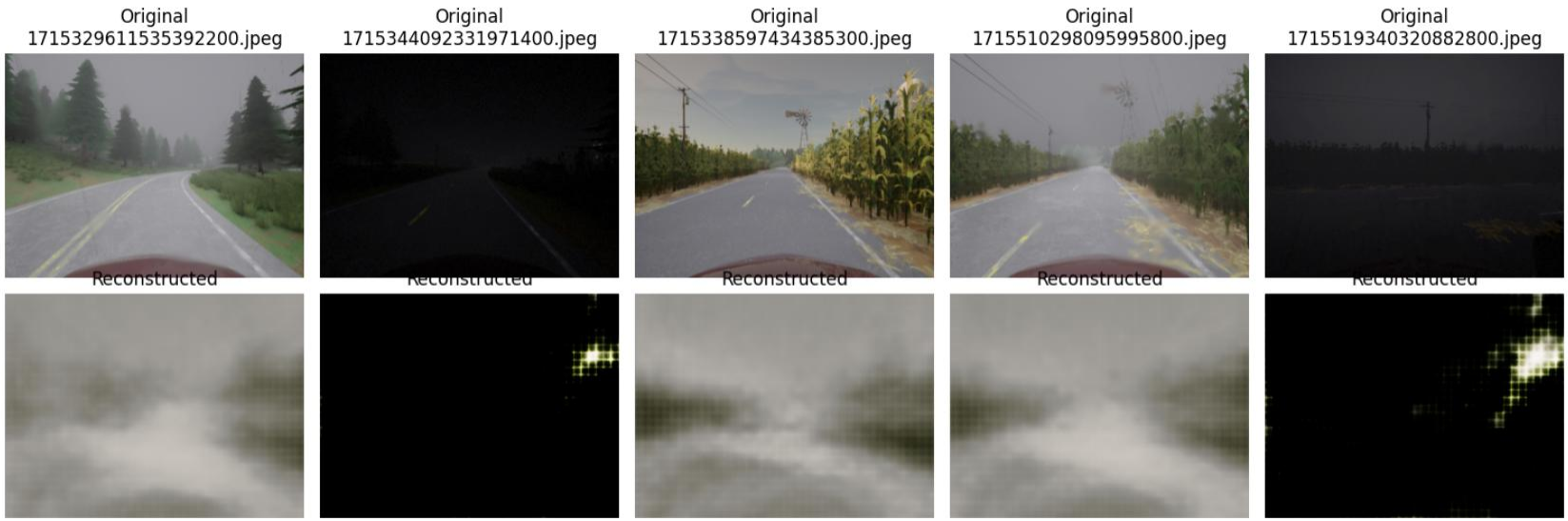

Figure 60: SOAE_model_ttv_602020_100



Figure 61: SOAE_model_ttv_701515_100

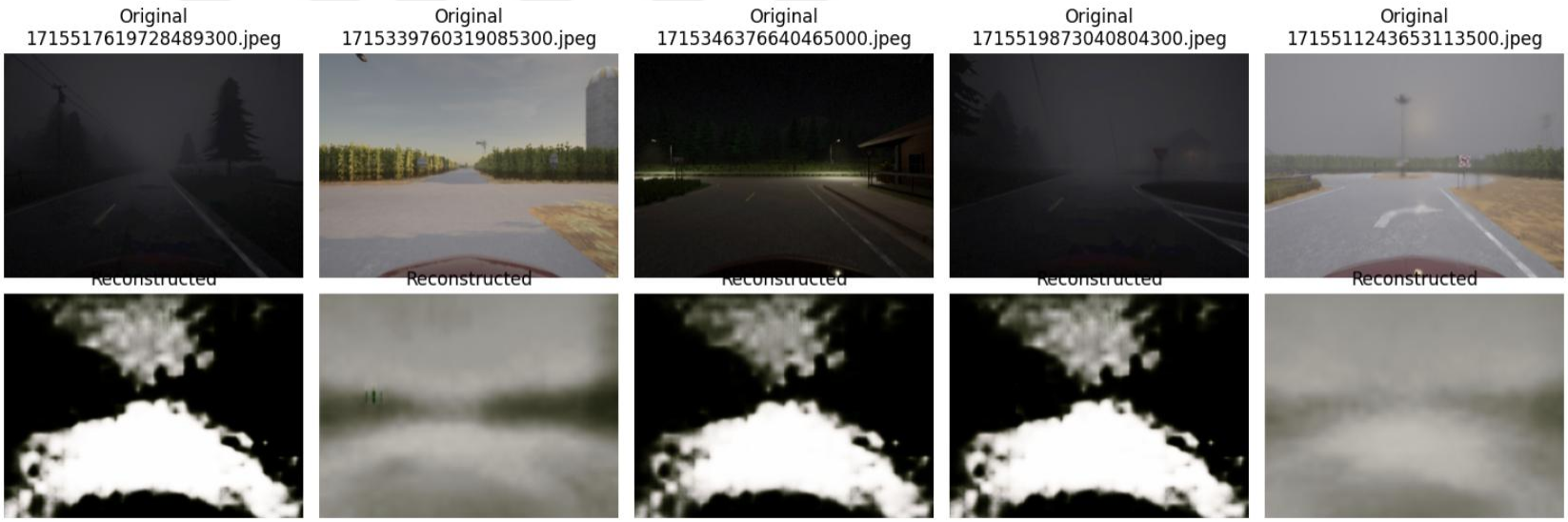

Figure 62: SOAE_model_ttv_801010_100

APPENDIX F

CIAE MODEL OUTPUTS (Epoch = 500)

The CIAE model outputs have been shown in APPENDIX F for 500 epochs.

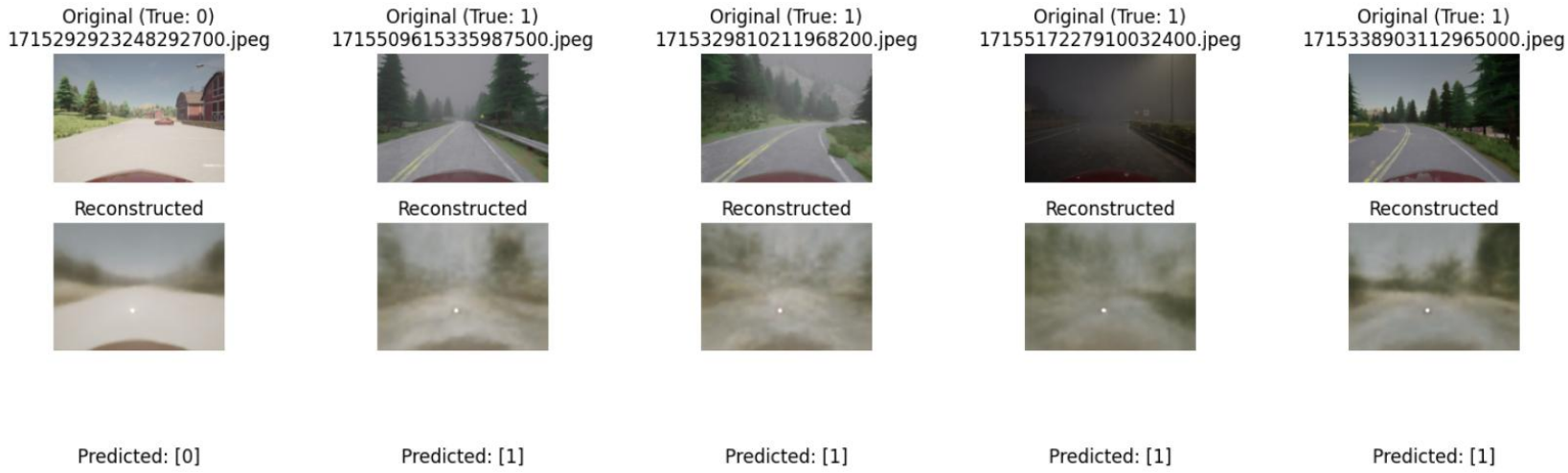

Figure 63: CIAE_model_bs_08_500

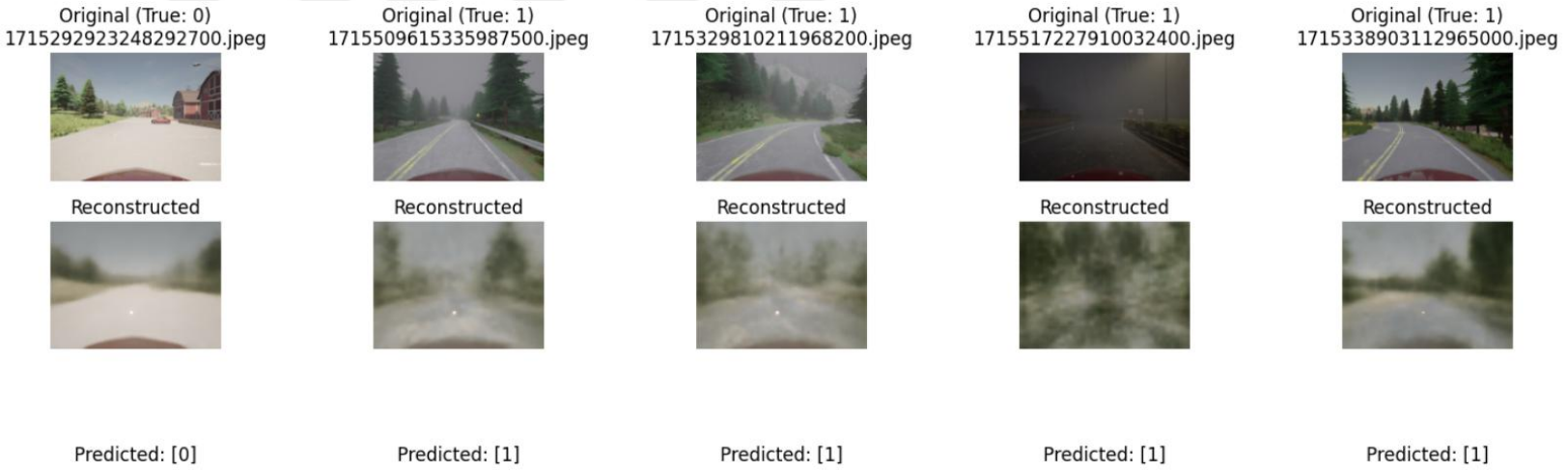

Figure 64: CIAE_model_bs_16_500

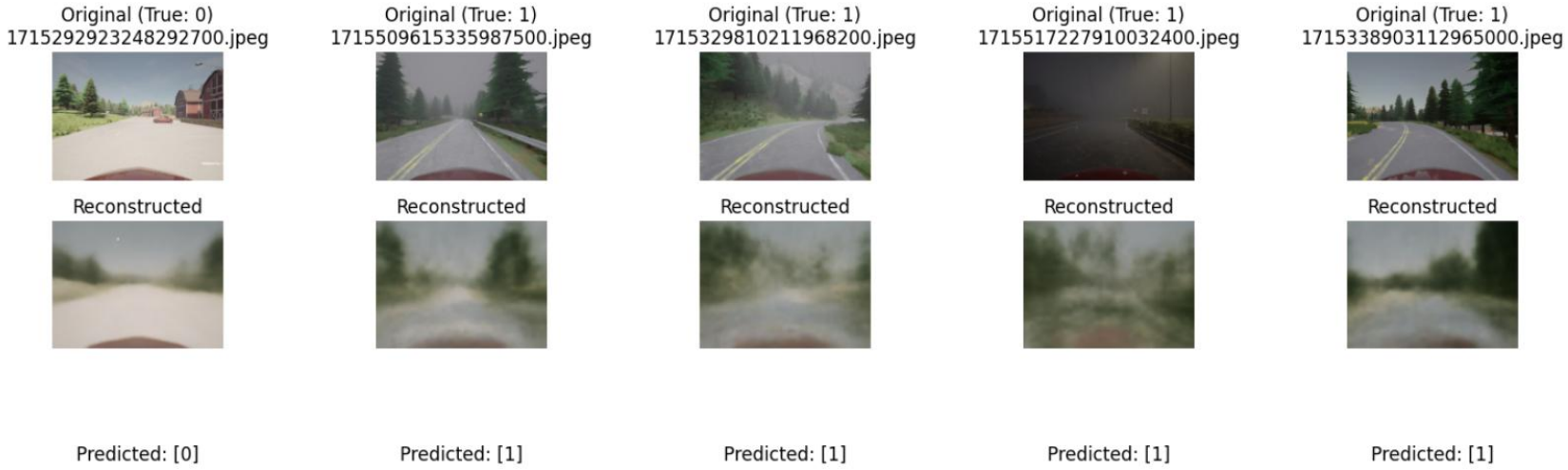

Figure 65: CIAE_model_bs_32_500

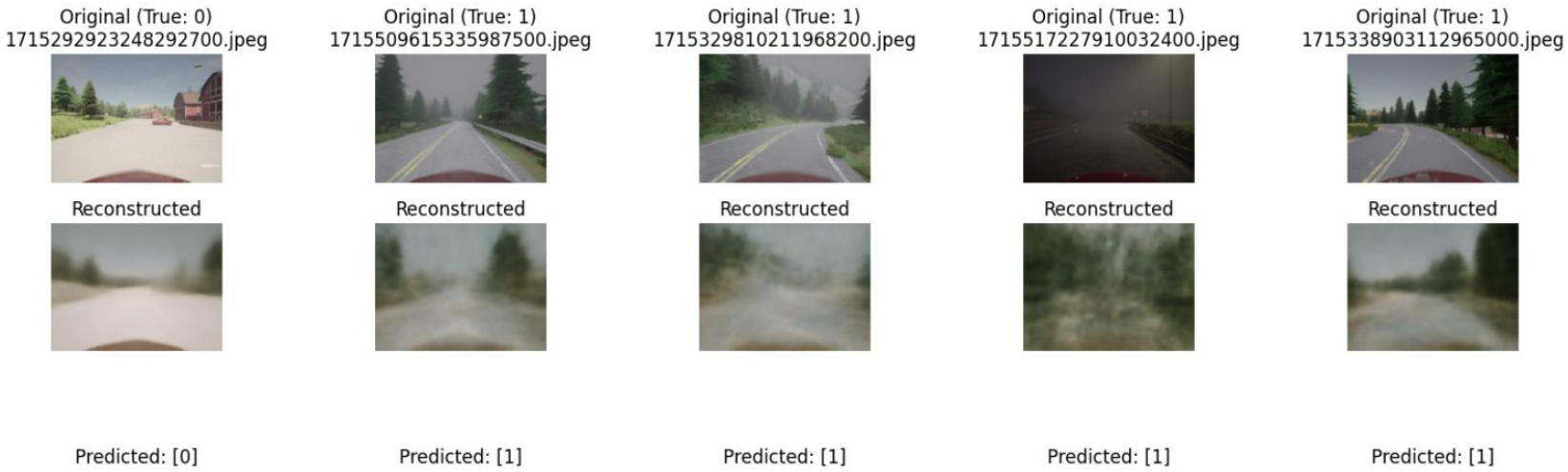

Figure 66: CIAE_model_dr_01_500

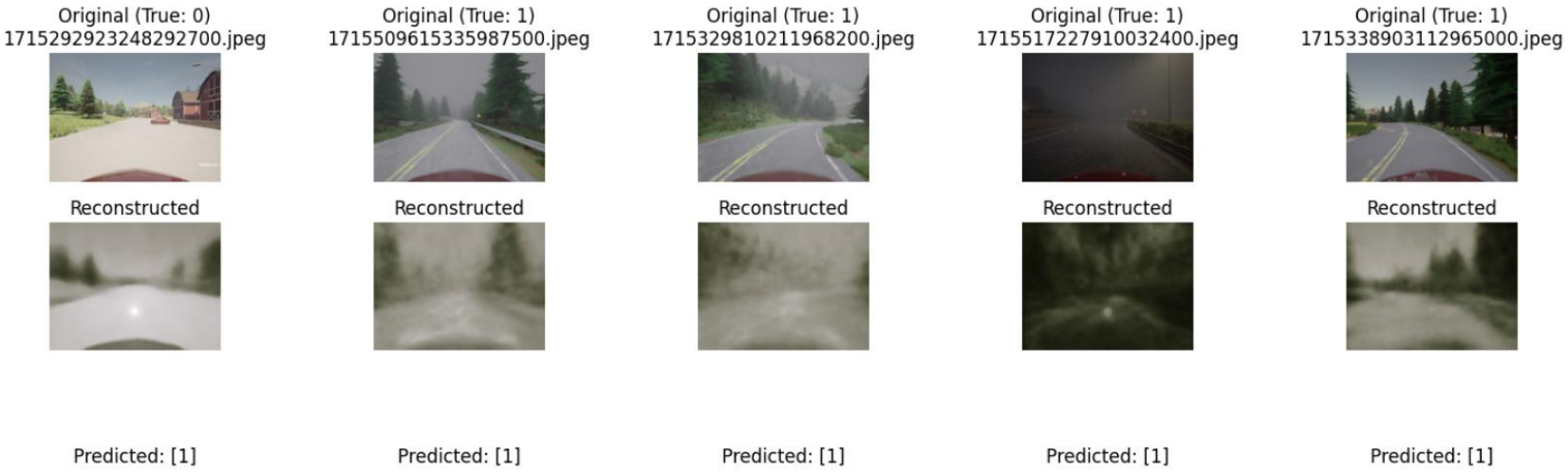

Figure 67: CIAE_model_dr_03_500

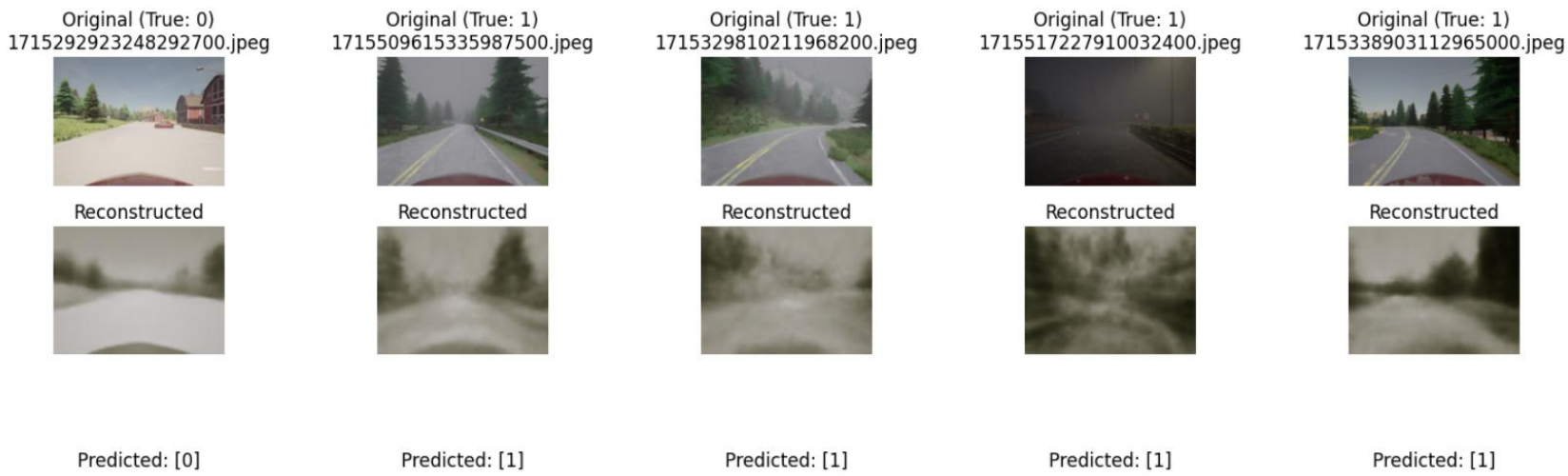

Figure 68: CIAE_model_dr_05_500

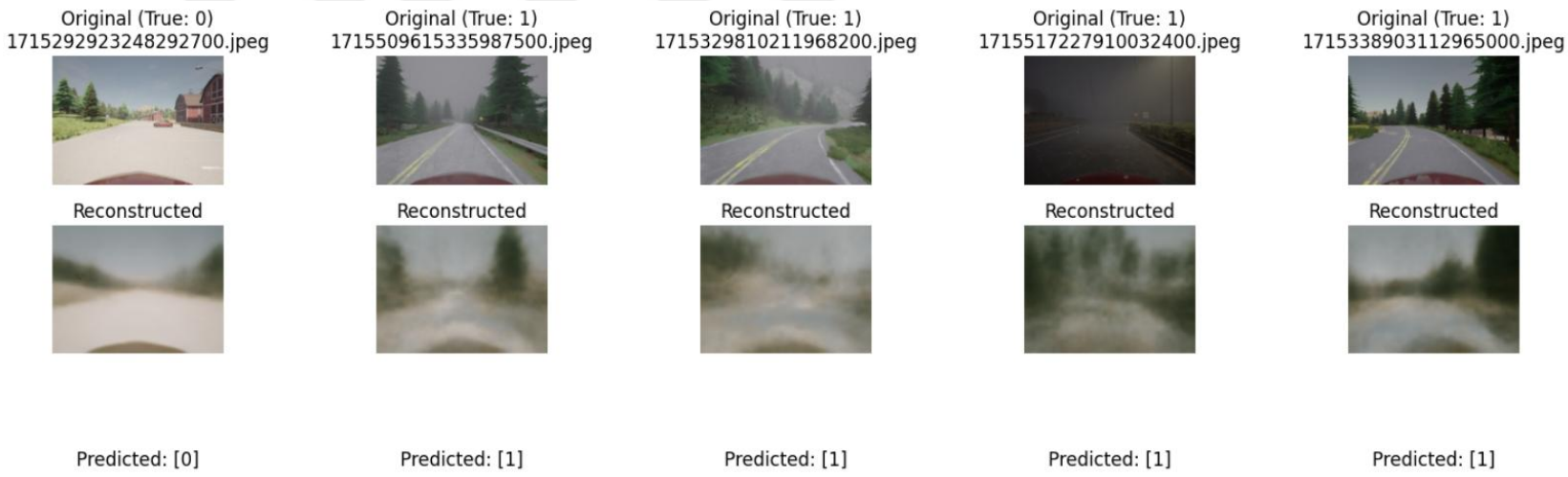

Figure 69: CIAE_model_ld_128_500

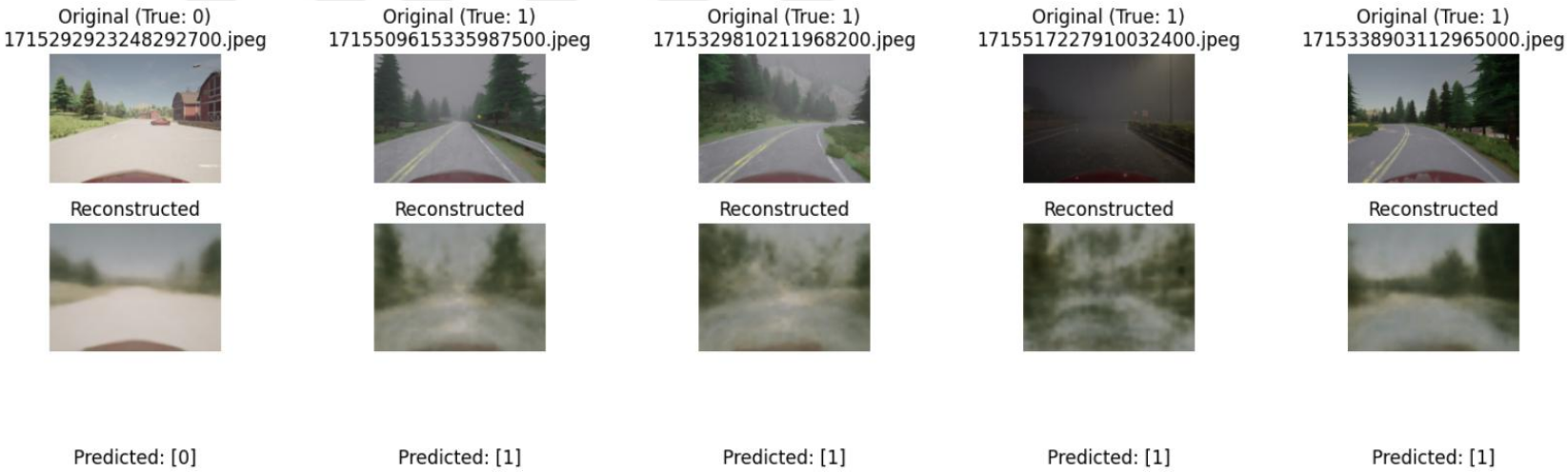

Figure 70: CIAE_model_ld_256_500

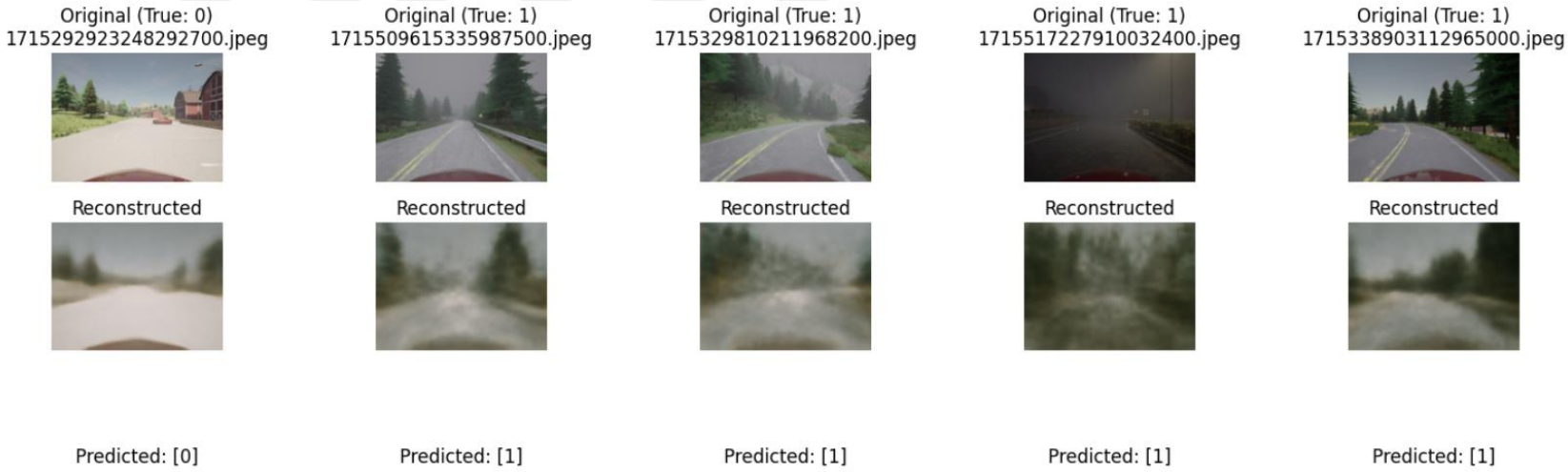

Figure 71: CIAE_model_ld_512_500

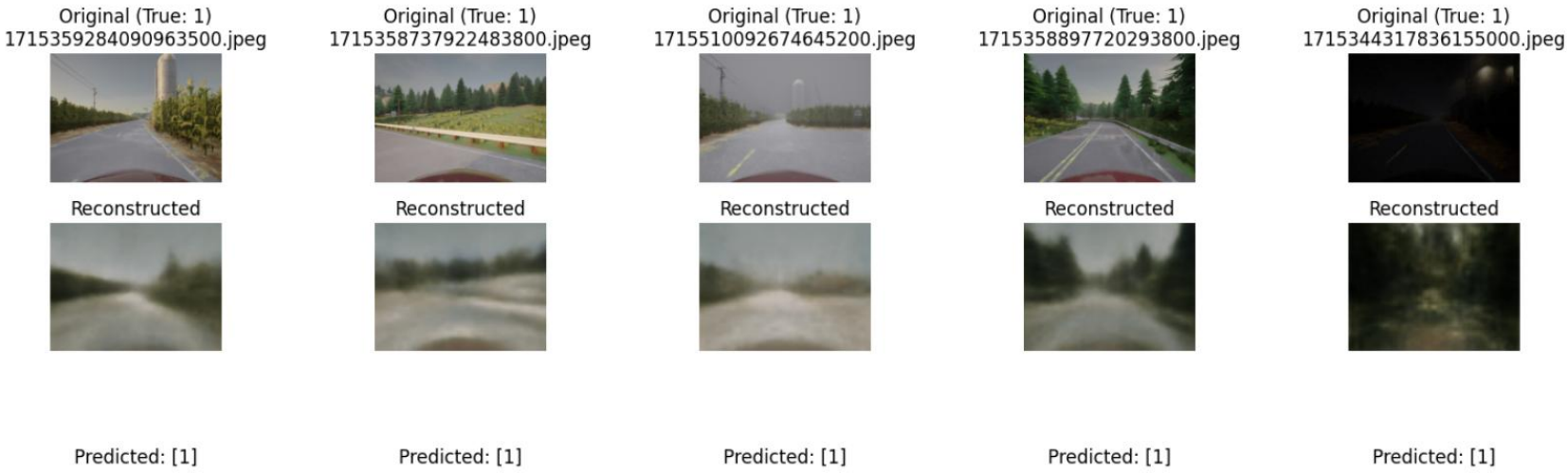

Figure 72: CIAE_model_ttv_602020_500

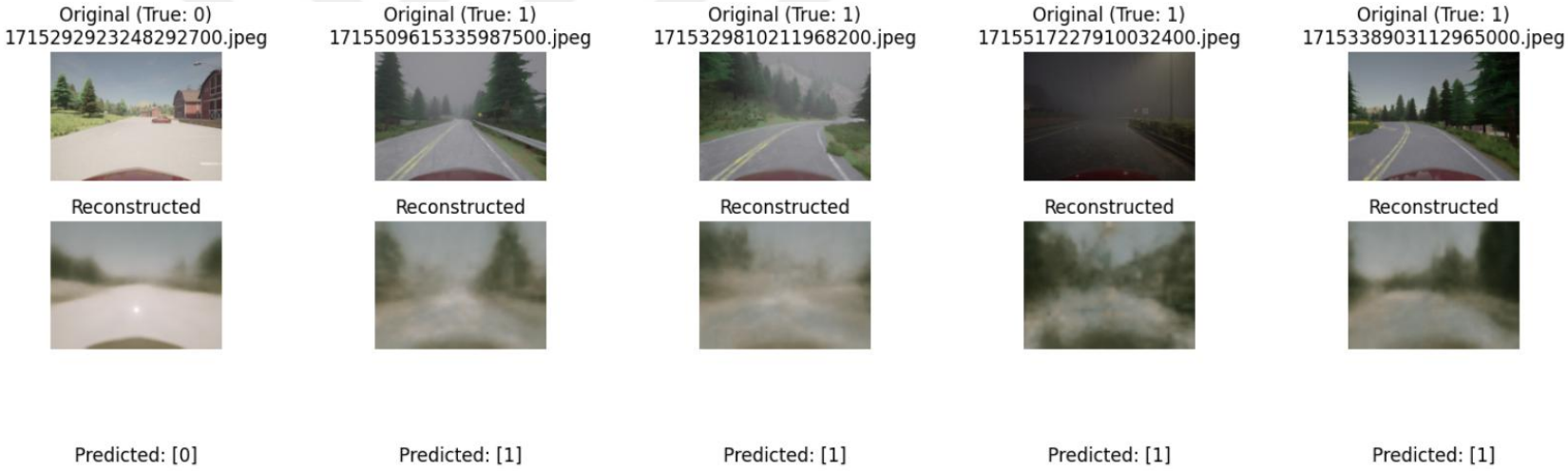

Figure 73: CIAE_model_ttv_701515_500

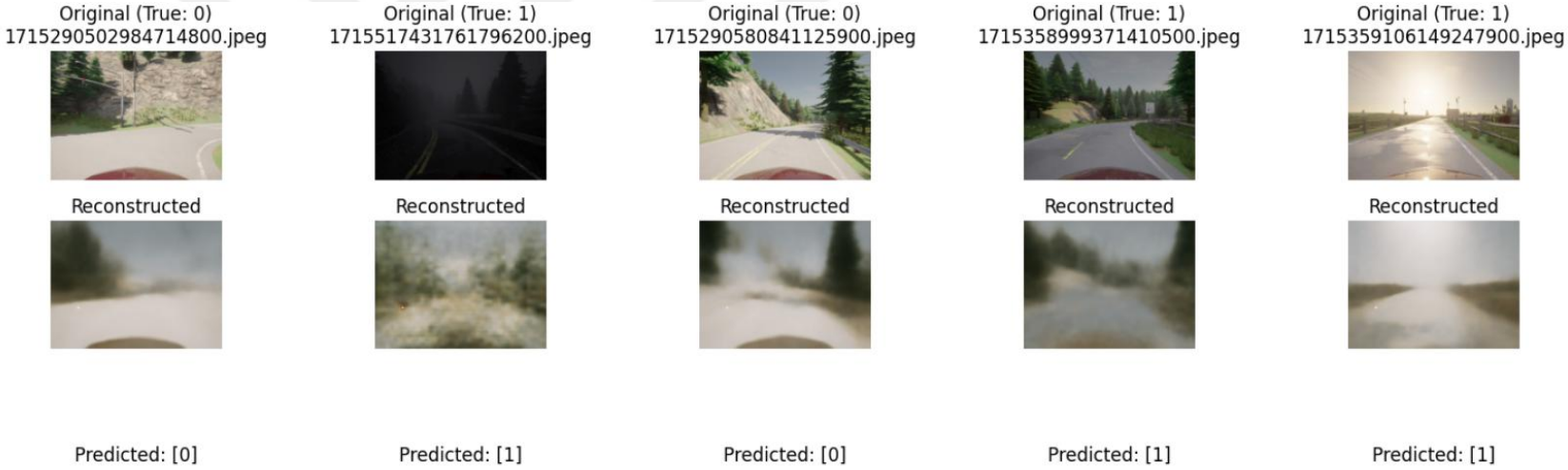

Figure 74: CIAE_model_ttv_801010_500

APPENDIX G

SOAE MODEL OUTPUTS (Epoch = 500)

The SOAE model outputs have been shown in APPENDIX G for 500 epochs.

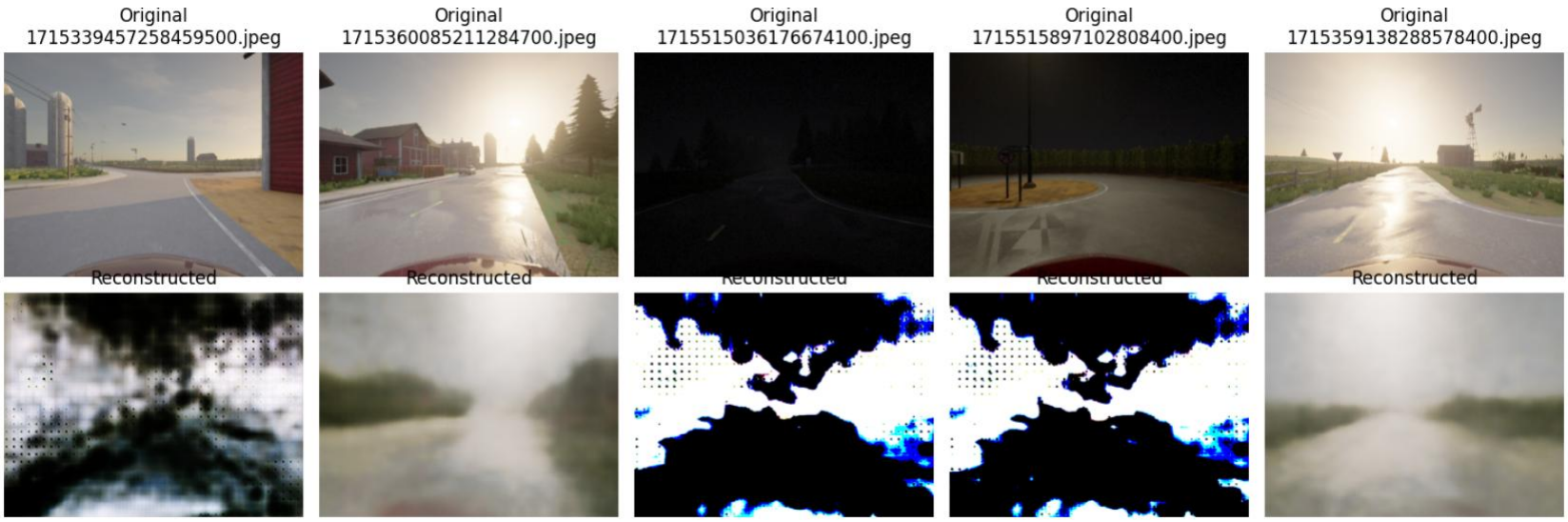

Figure 75: SOAE_model_bs_08_500



Figure 76: SOAE_model_bs_16_500

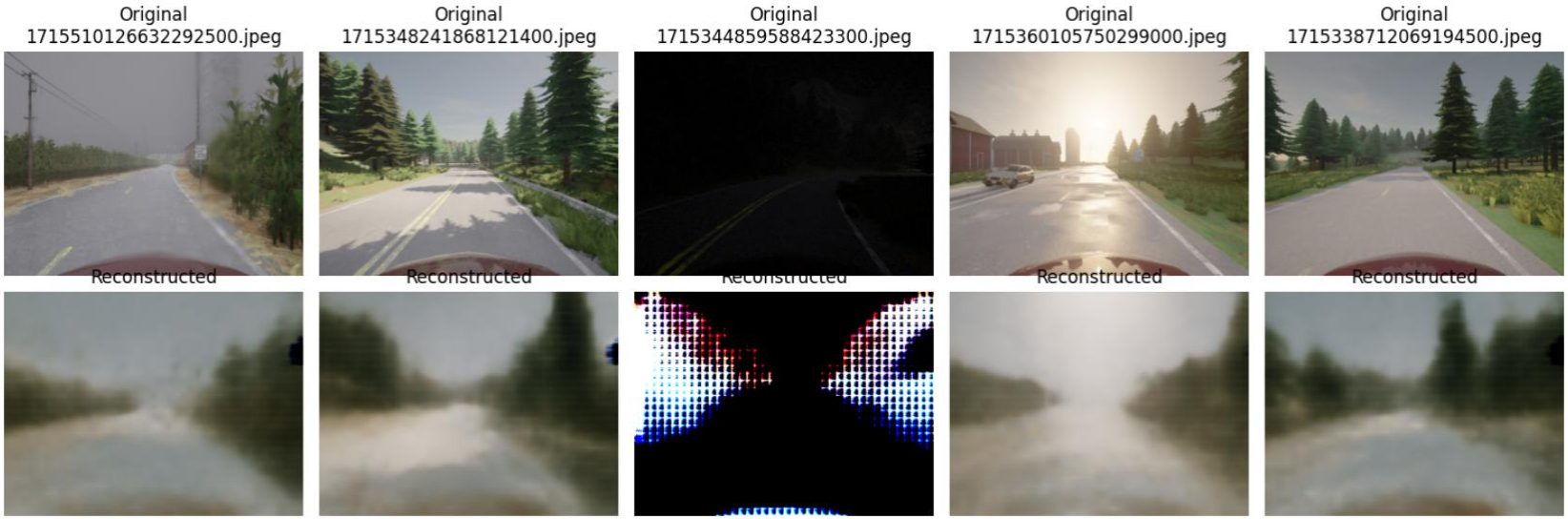

Figure 77: SOAE_model_bs_32_500

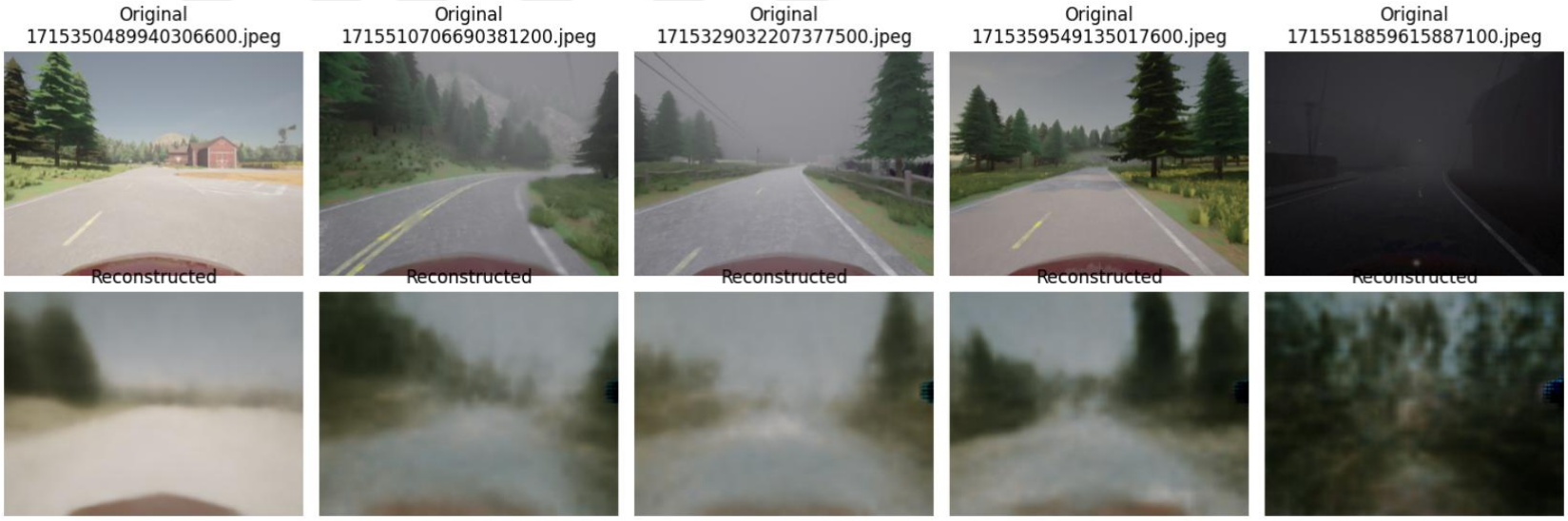

Figure 78: SOAE_model_dr_01_500



Figure 79: SOAE_model_dr_03_500



Figure 80: SOAE_model_dr_05_500

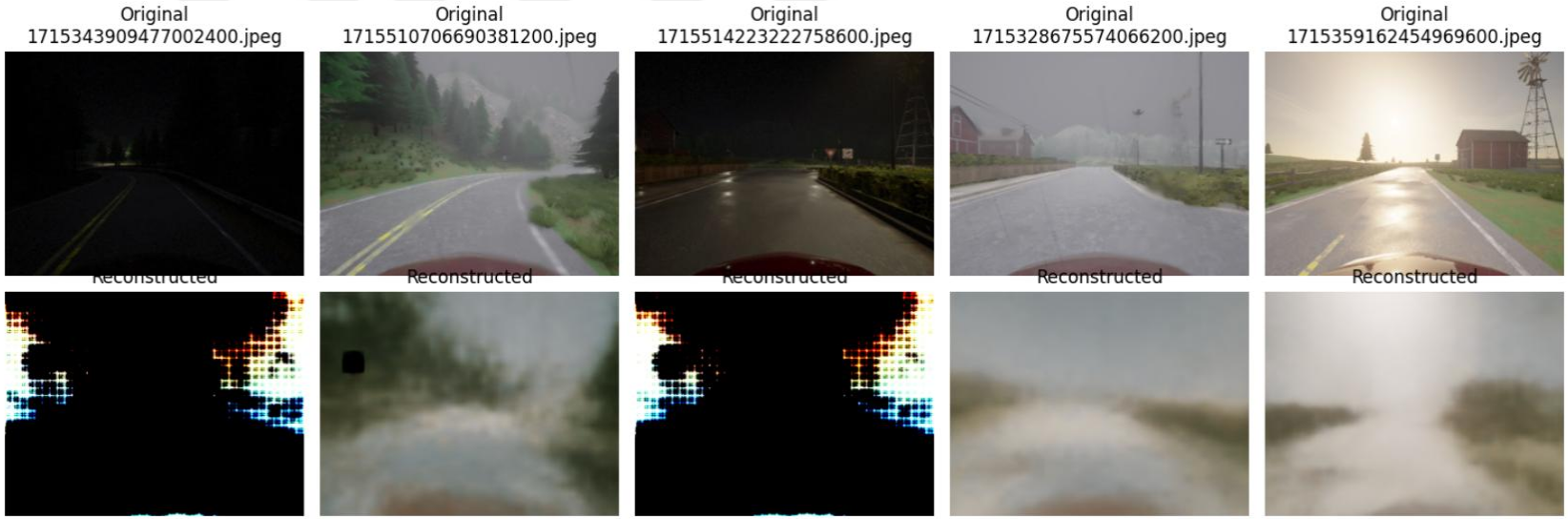

Figure 81: SOAE_model_ld_128_500

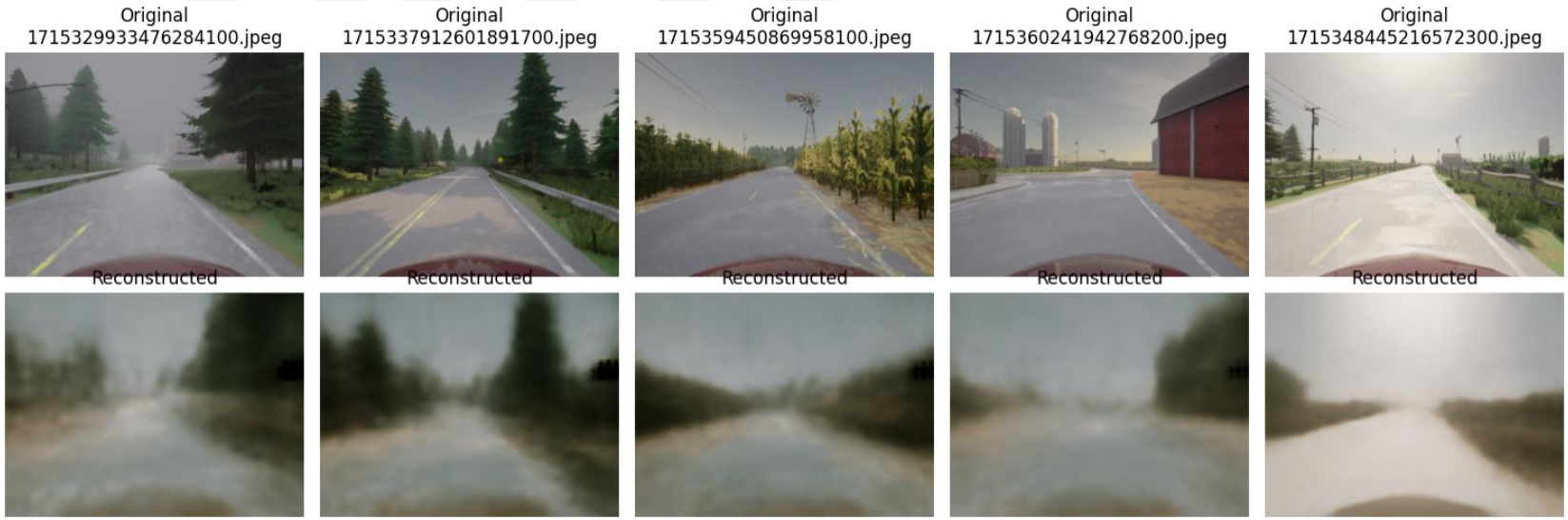

Figure 82: SOAE_model_ld_256_500

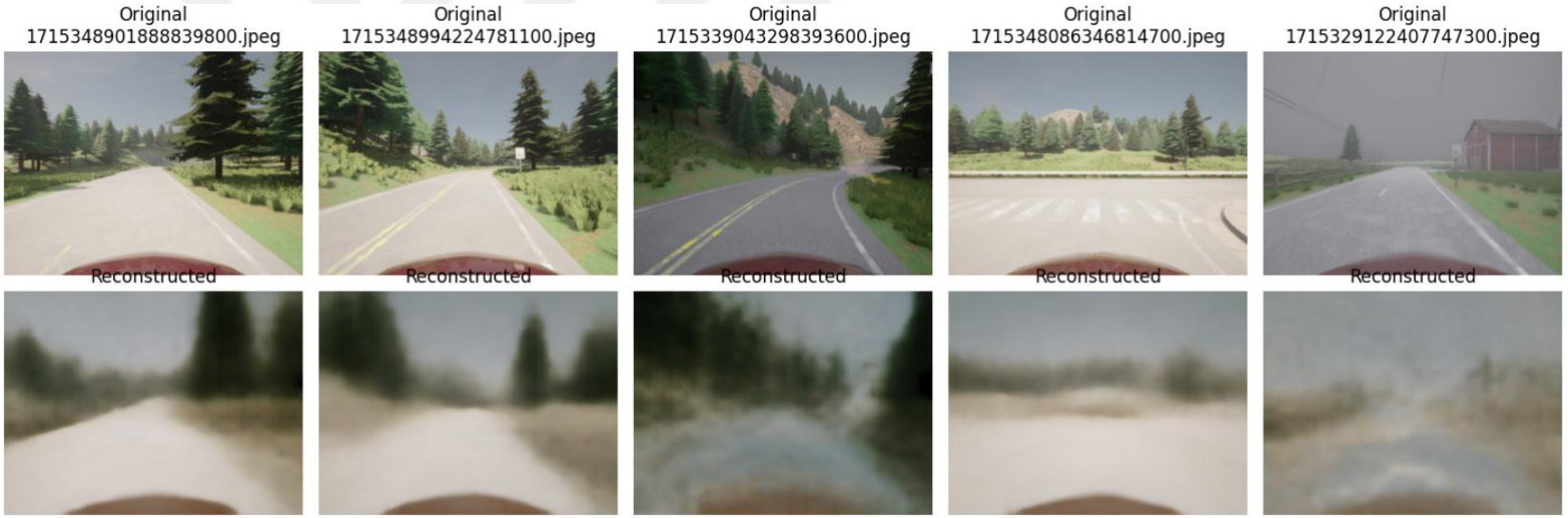

Figure 83: SOAE_model_ld_512_500

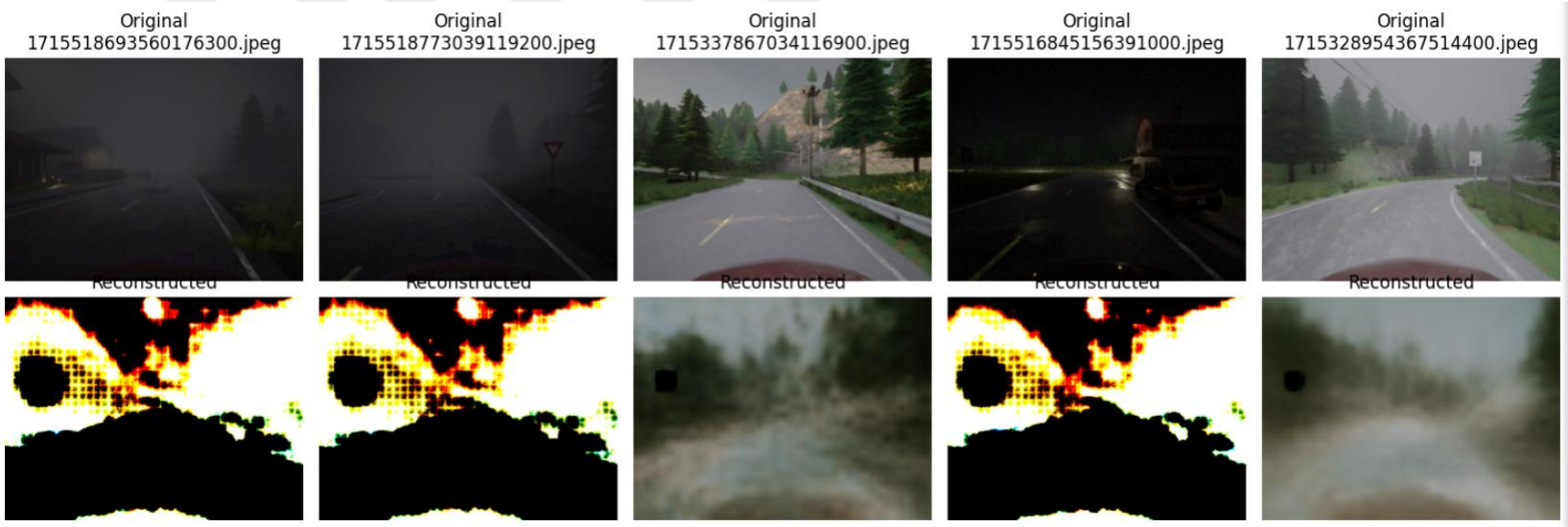

Figure 84: SOAE_model_ttv_602020_500



Figure 85: SOAE_model_ttv_701515_500

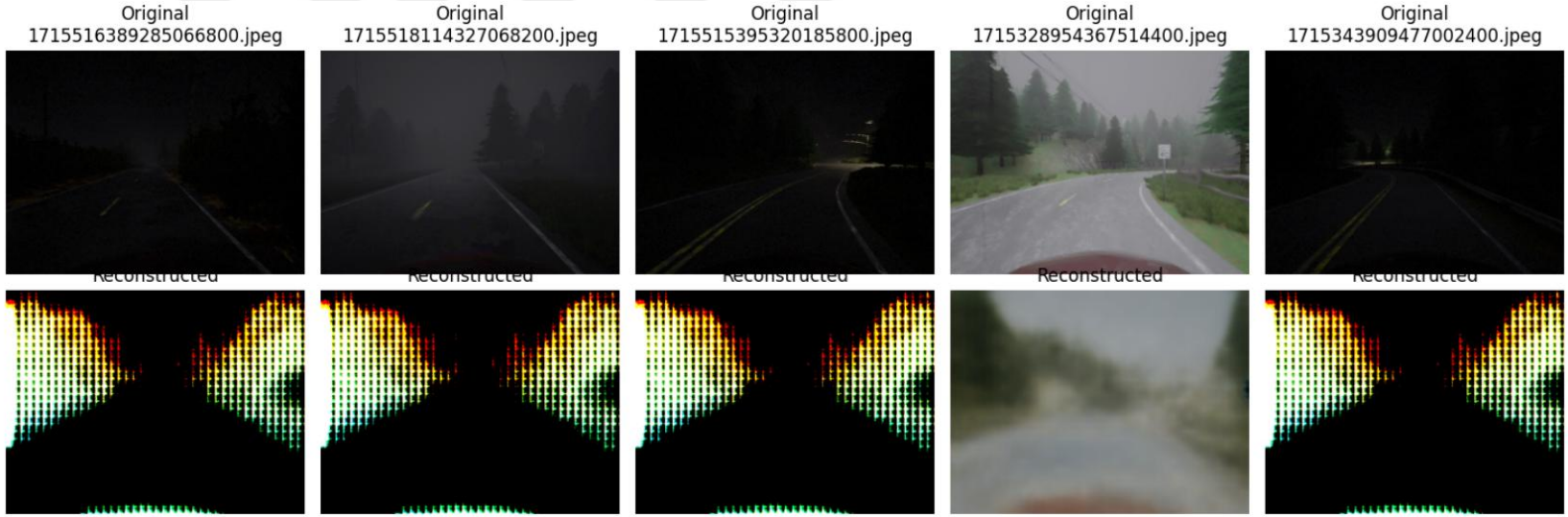

Figure 86: SOAE_model_ttv_801010_500

APPENDIX H

CIAE MODEL OUTPUTS (Epoch = 1000)

The CIAE model outputs have been shown in APPENDIX H for 1000 epochs.



Figure 87: CIAE_model_bs_08_1000



Figure 88: CIAE_model_bs_16_1000



Figure 89: CIAE_model_bs_32_1000



Figure 90: CIAE_model_dr_01_1000

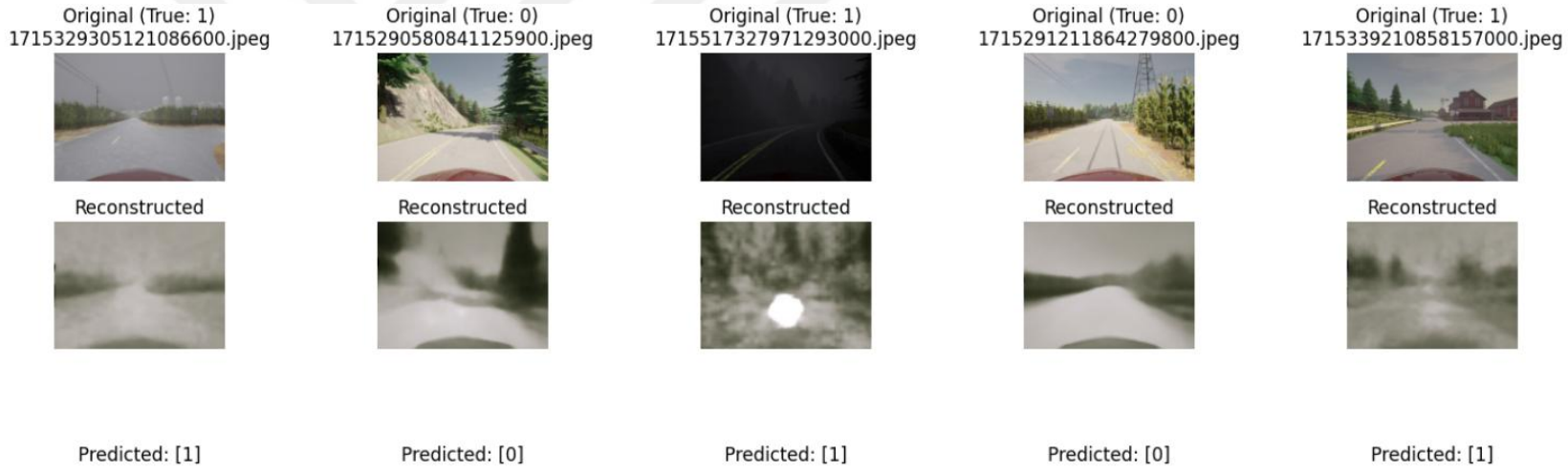

Figure 91: CIAE_model_dr_03_1000

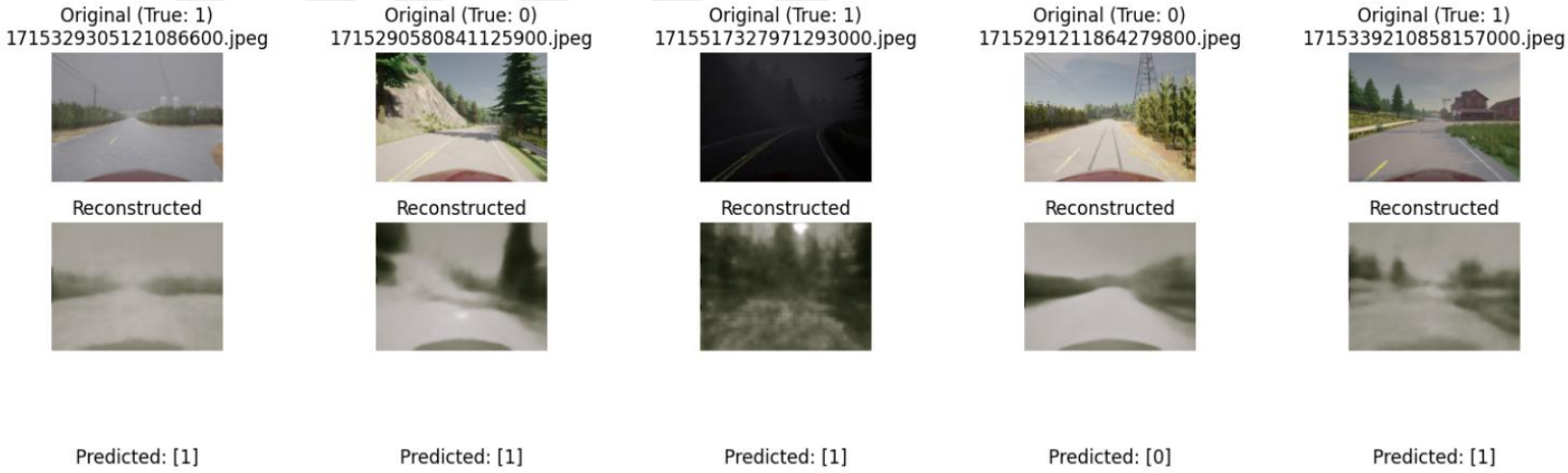

Figure 92: CIAE_model_dr_05_1000



Figure 93: CIAE_model_ld_128_1000



Figure 94: CIAE_model_ld_256_1000



Figure 95: CIAE_model_ld_512_1000



Figure 96: CIAE_model_ttv_602020_1000



Figure 97: CIAE_model_ttv_701515_1000

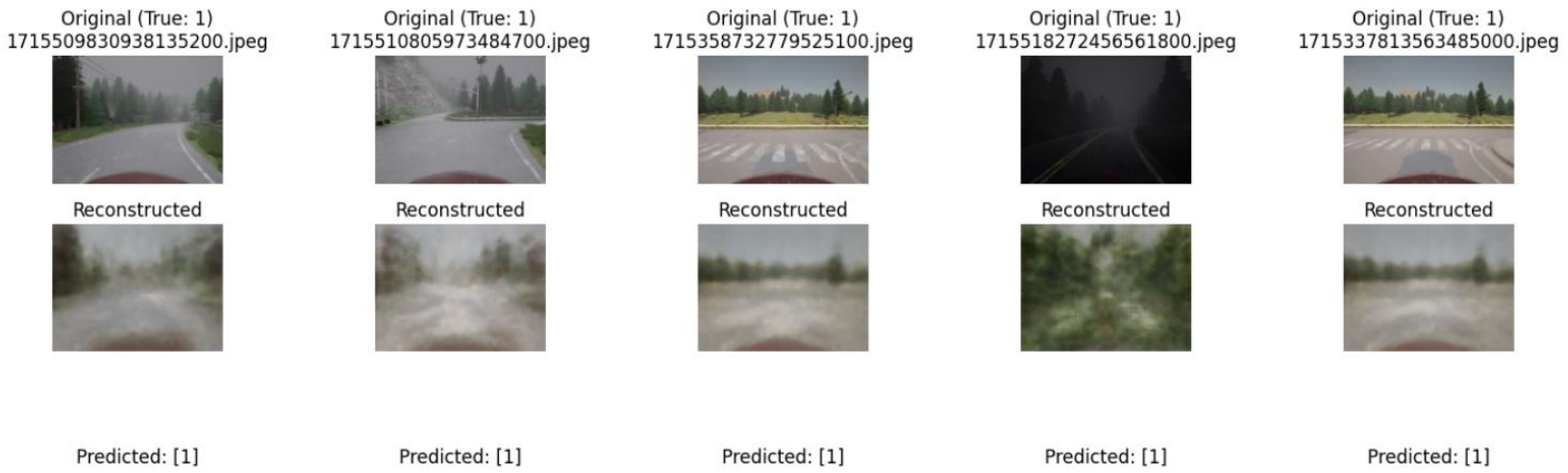

Figure 98: CIAE_model_ttv_801010_1000

APPENDIX I

SOAE MODEL OUTPUTS (Epoch = 1000)

The SOAE model outputs have been shown in APPENDIX I for 1000 epochs.

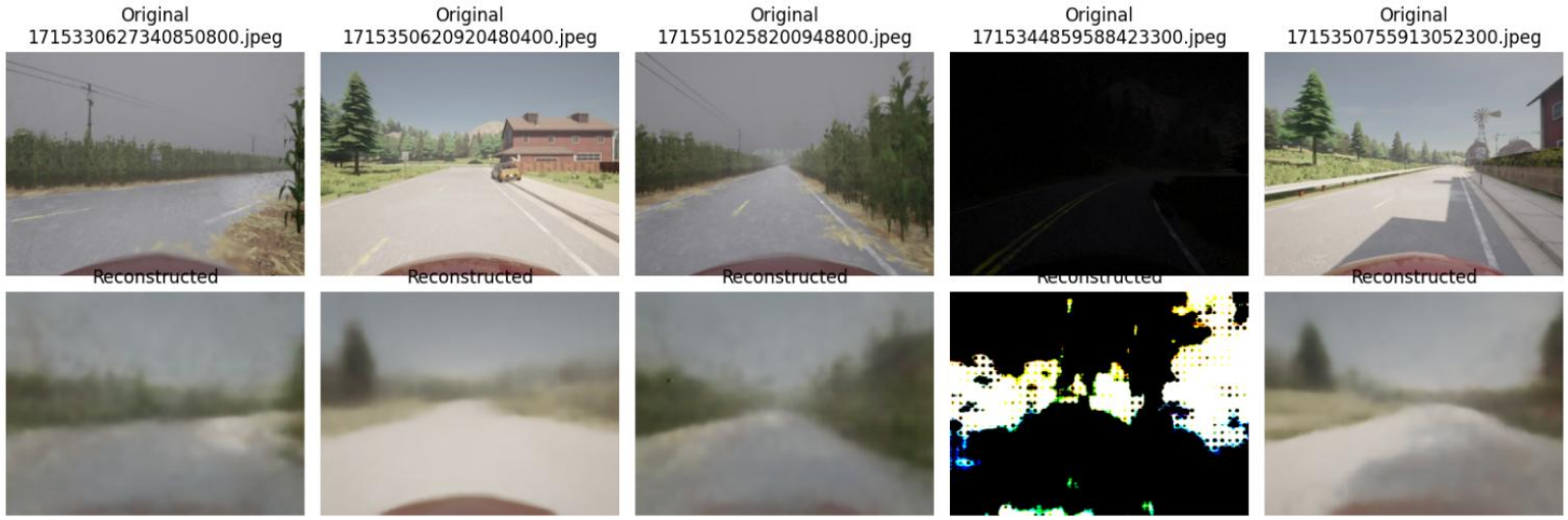

Figure 99: SOAE_model_bs_08_1000



Figure 100: SOAE_model_bs_16_1000

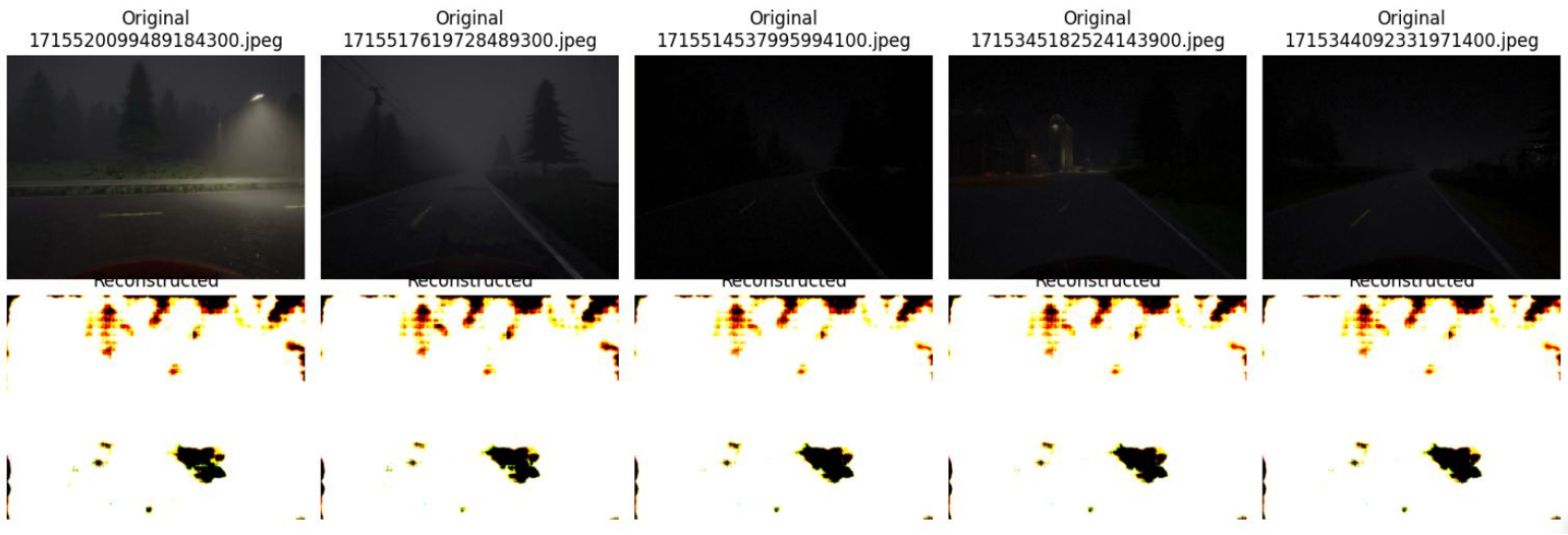

Figure 101: SOAE_model_bs_32_1000

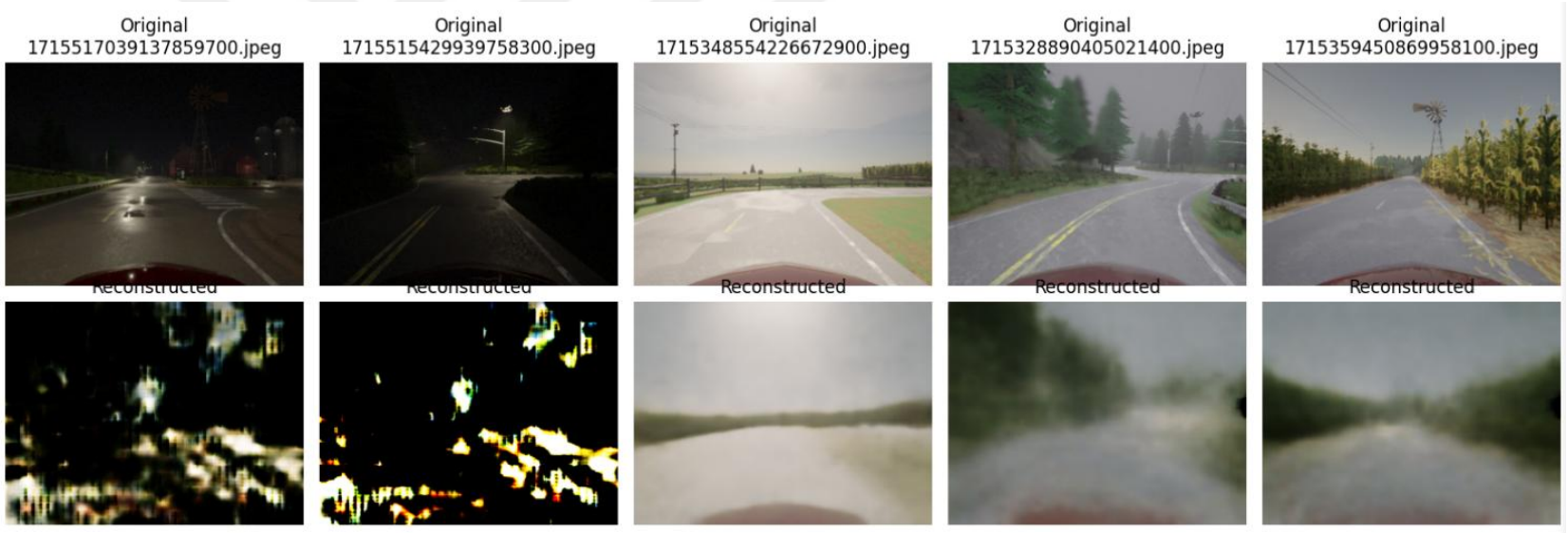

Figure 102: SOAE_model_dr_01_1000

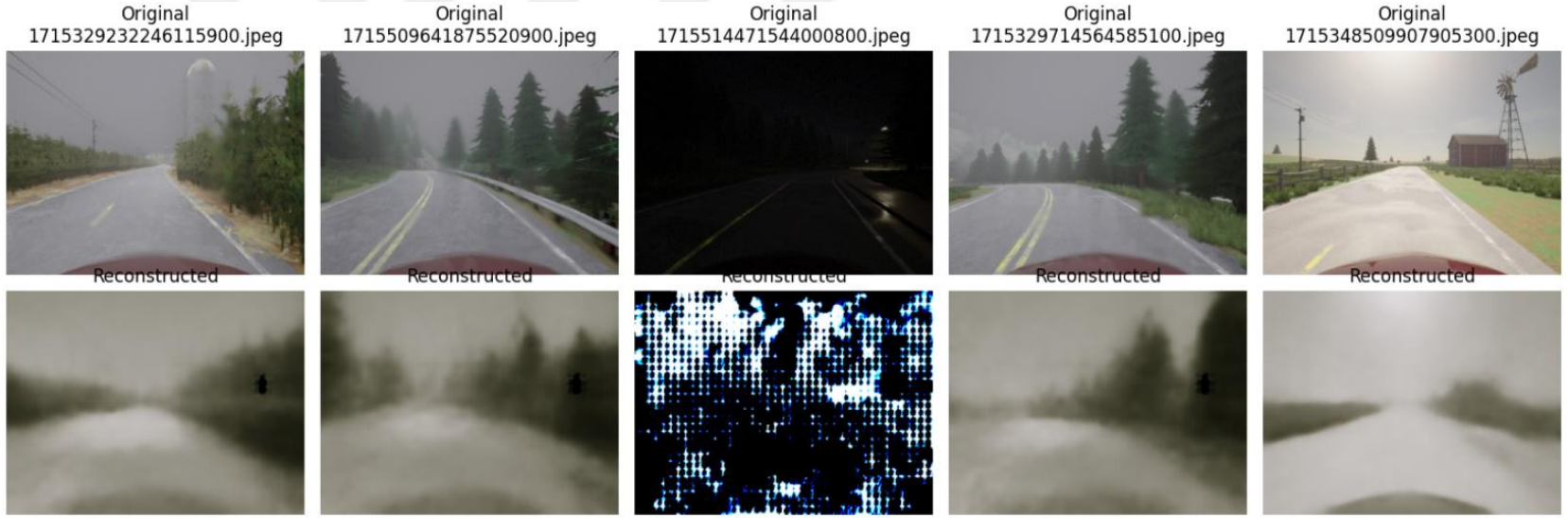

Figure 103: SOAE_model_dr_03_1000

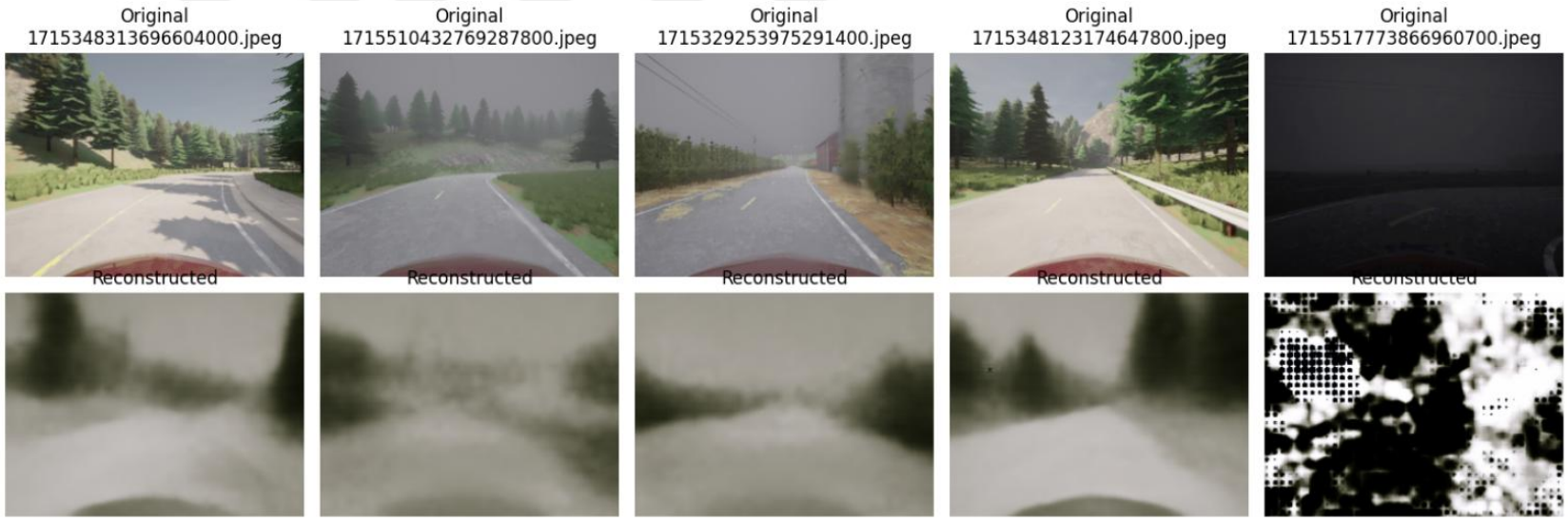

Figure 104: SOAE_model_dr_05_1000



Figure 105: SOAE_model_ld_128_1000

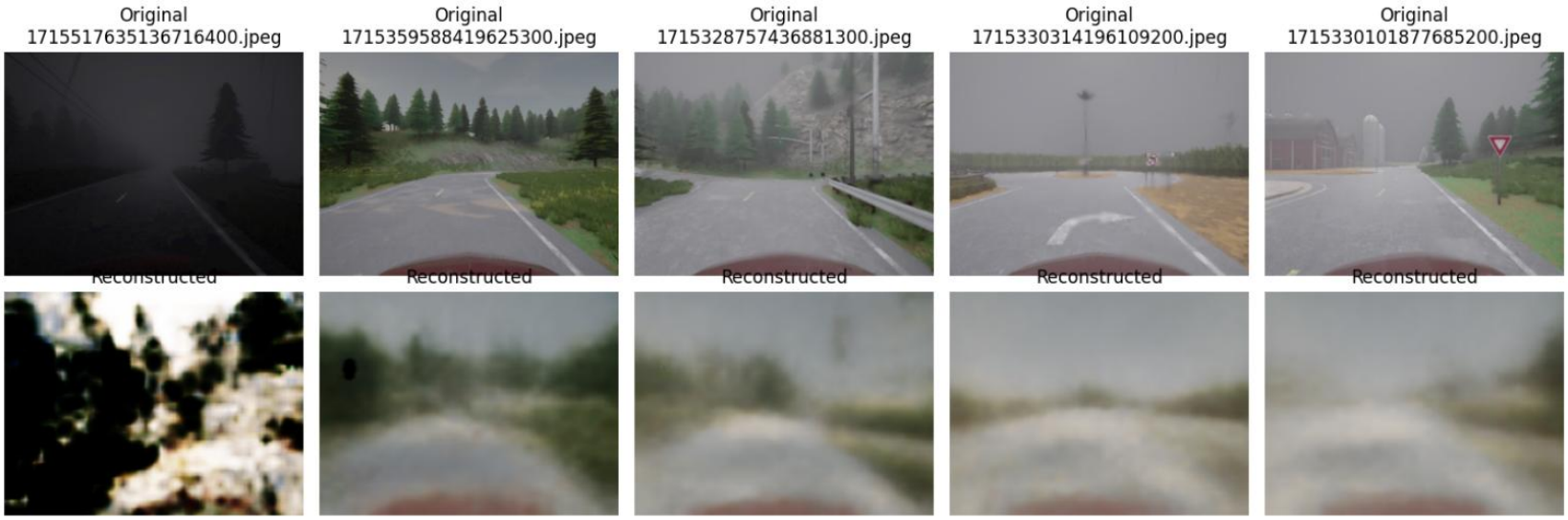

Figure 106: SOAE_model_ld_256_1000

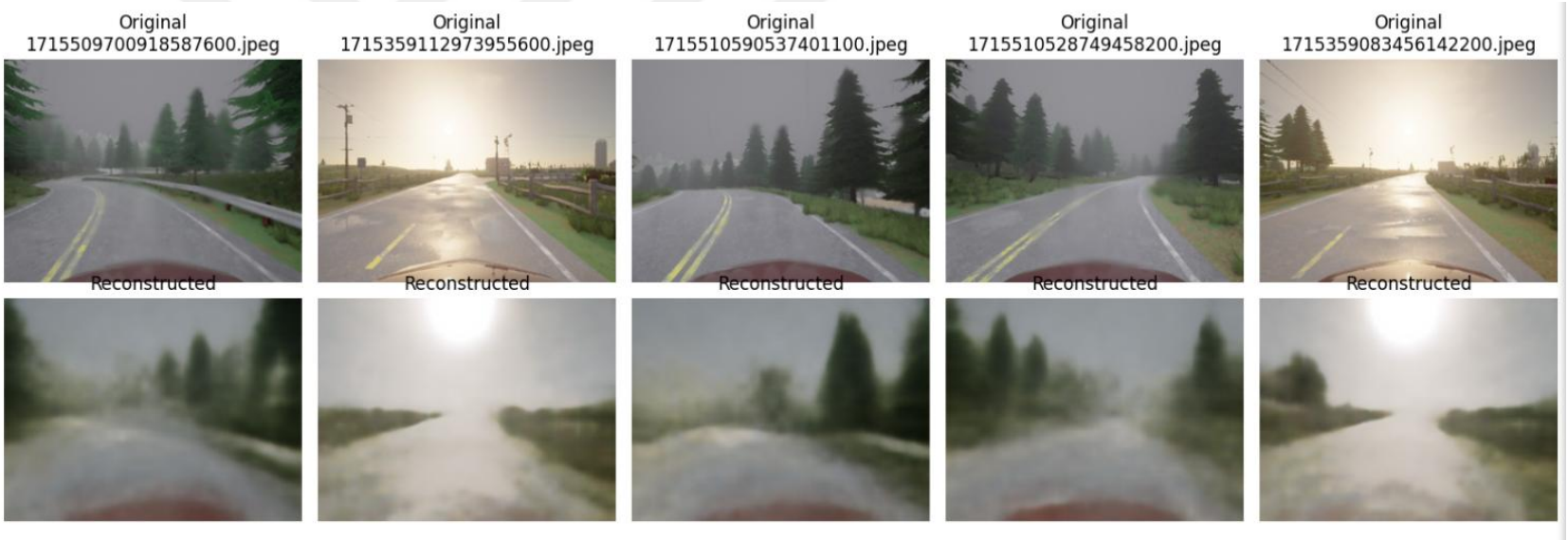

Figure 107: SOAE_model_ld_512_1000

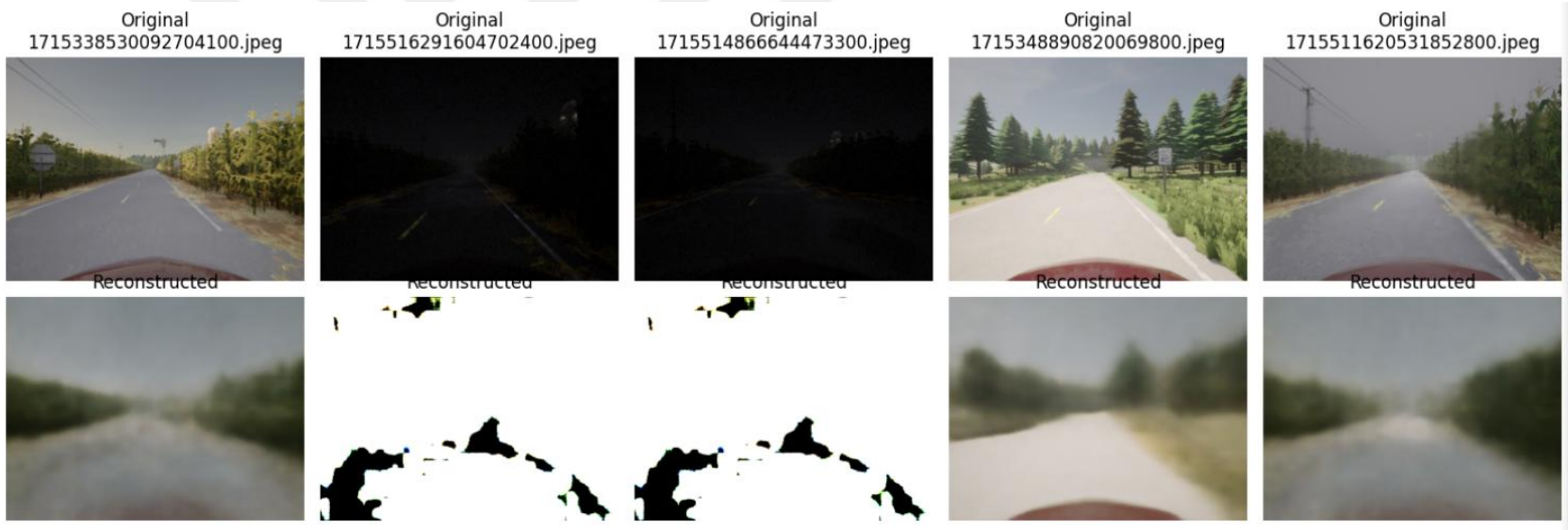

Figure 108: SOAE_model_ttv_602020_1000

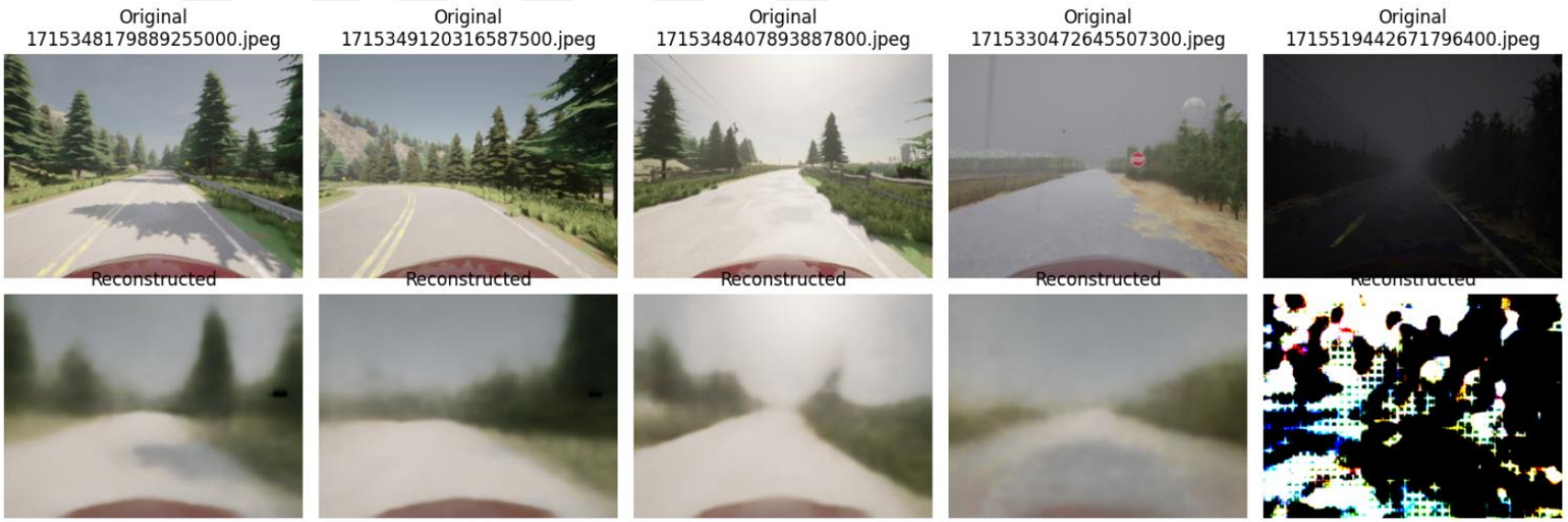

Figure 109: SOAE_model_ttv_701515_1000

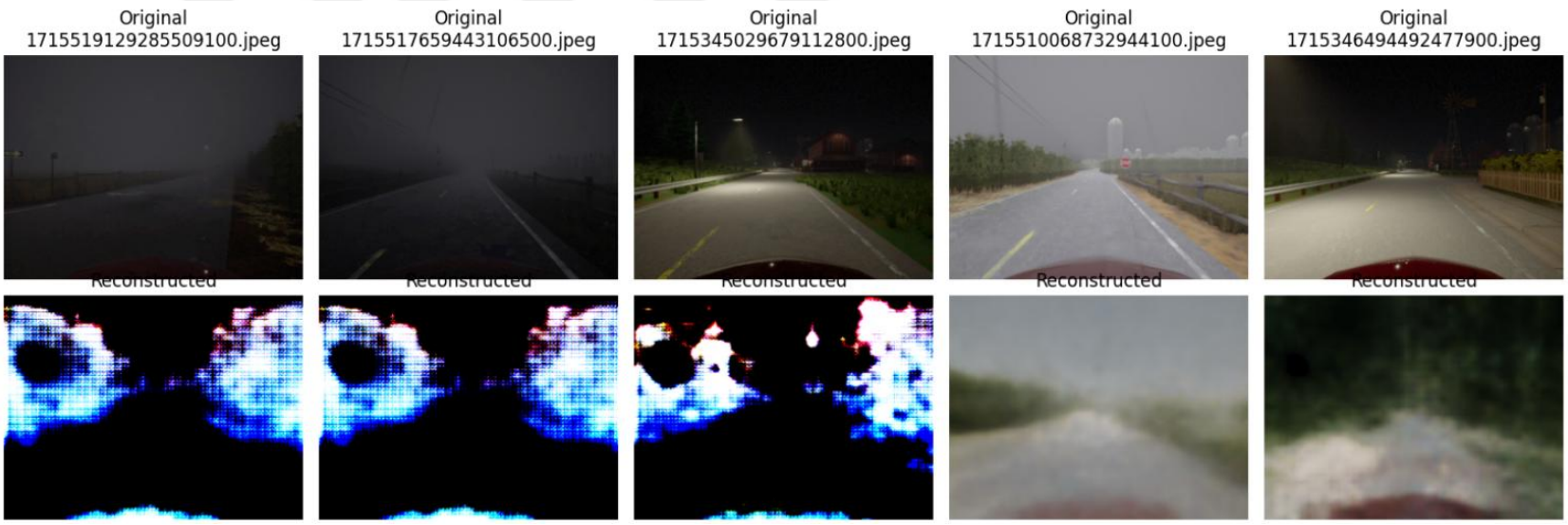

Figure 110: SOAE_model_ttv_801010_1000

APPENDIX J

CIAE VS SOAE MODEL COMPARISONS

The CIAE and SOAE model result comparisons have been shown in APPENDIX J.

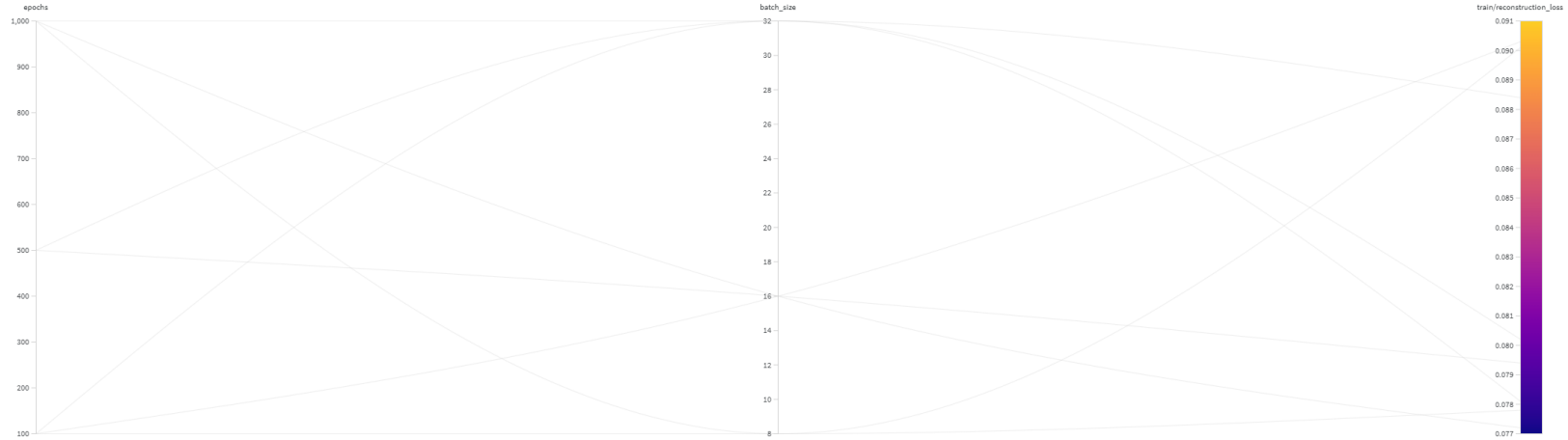

Figure 111: CIAE train reconstruction loss comparison based on epochs and batch size

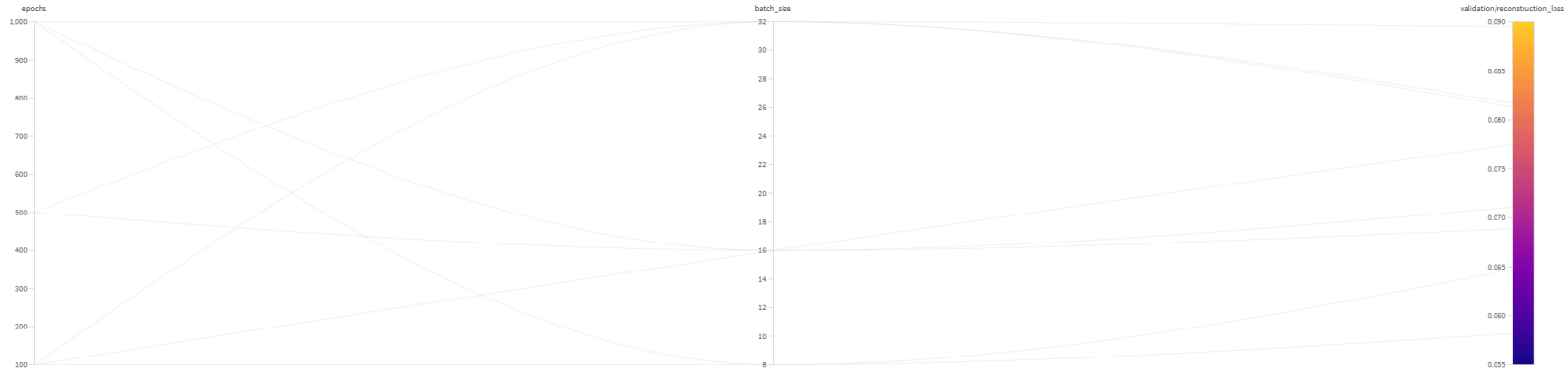

Figure 112: CIAE validation reconstruction loss comparison based on epochs and batch size

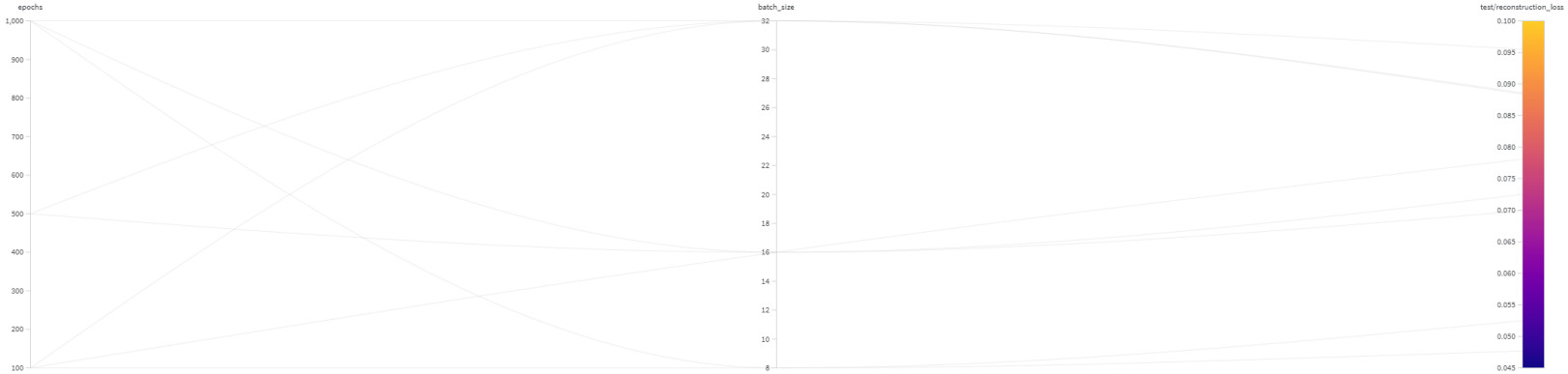

Figure 113: CIAE test reconstruction loss comparison based on epochs and batch size

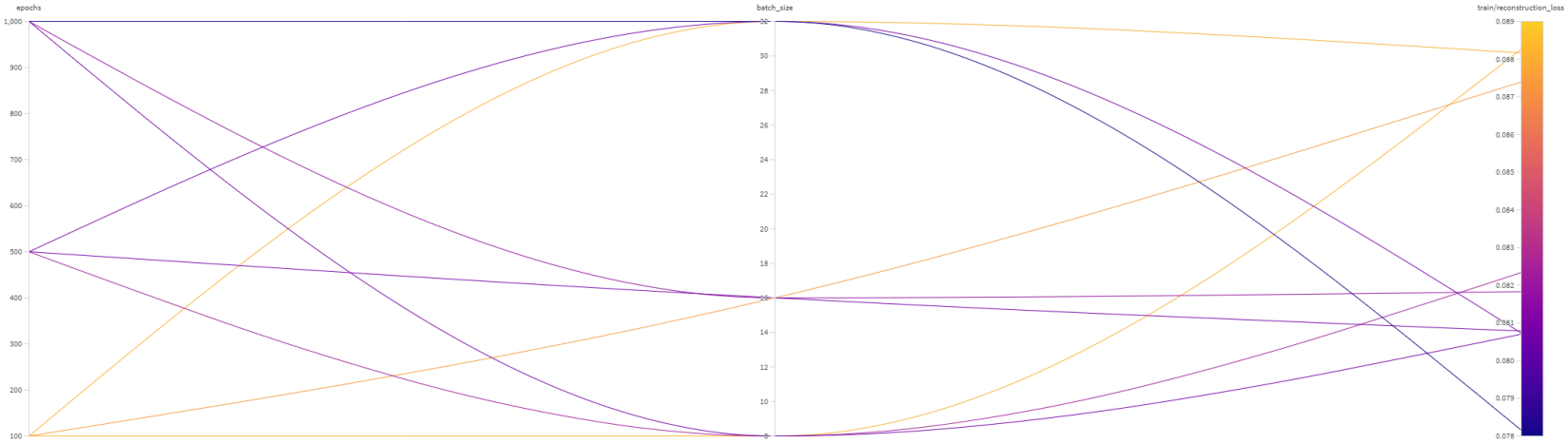

Figure 114: SOAE train reconstruction loss comparison based on epochs and batch size

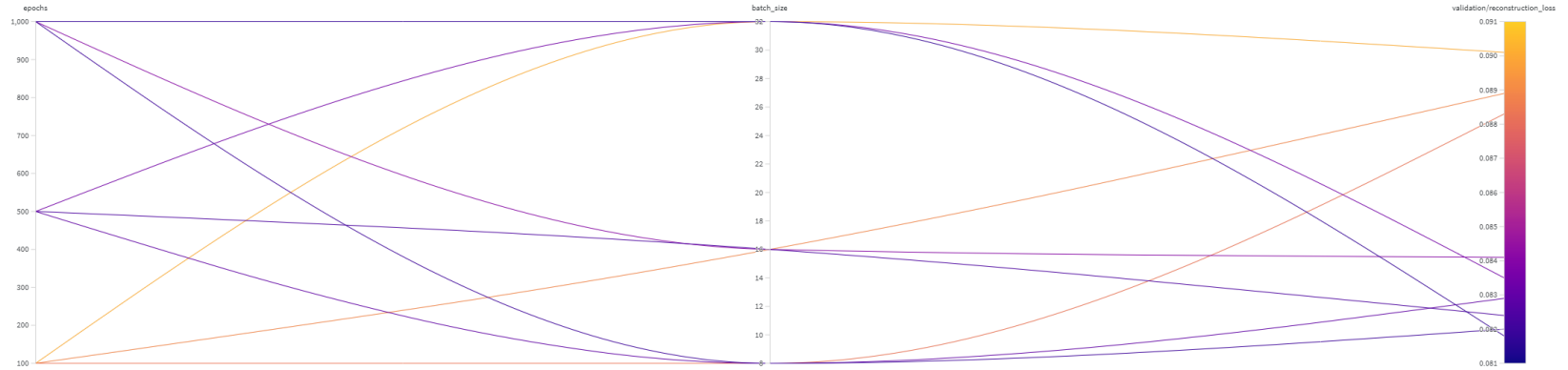

Figure 115: SOAE validation reconstruction loss comparison based on epochs and batch size

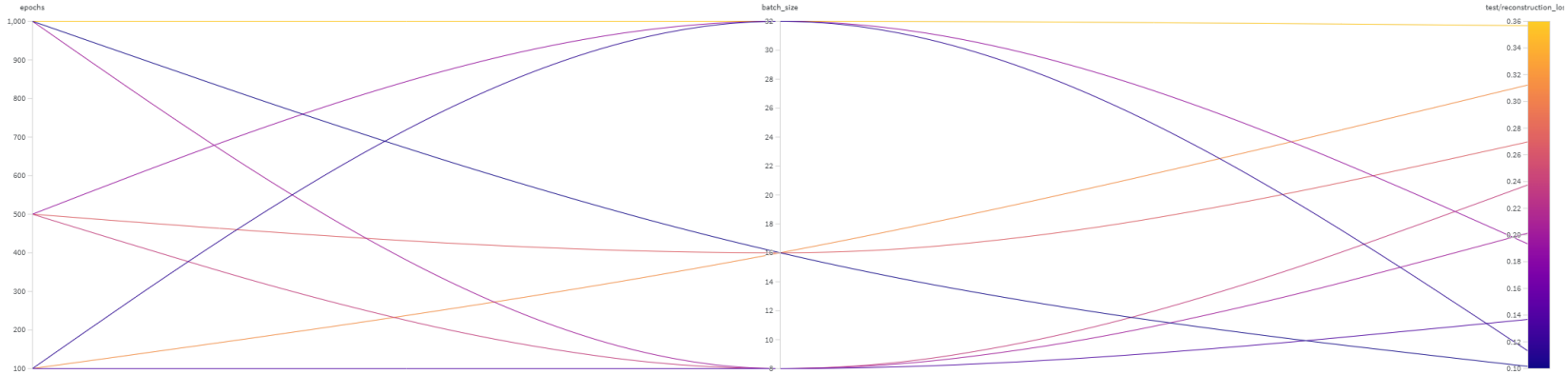

Figure 116: SOAE test reconstruction loss comparison based on epochs and batch size

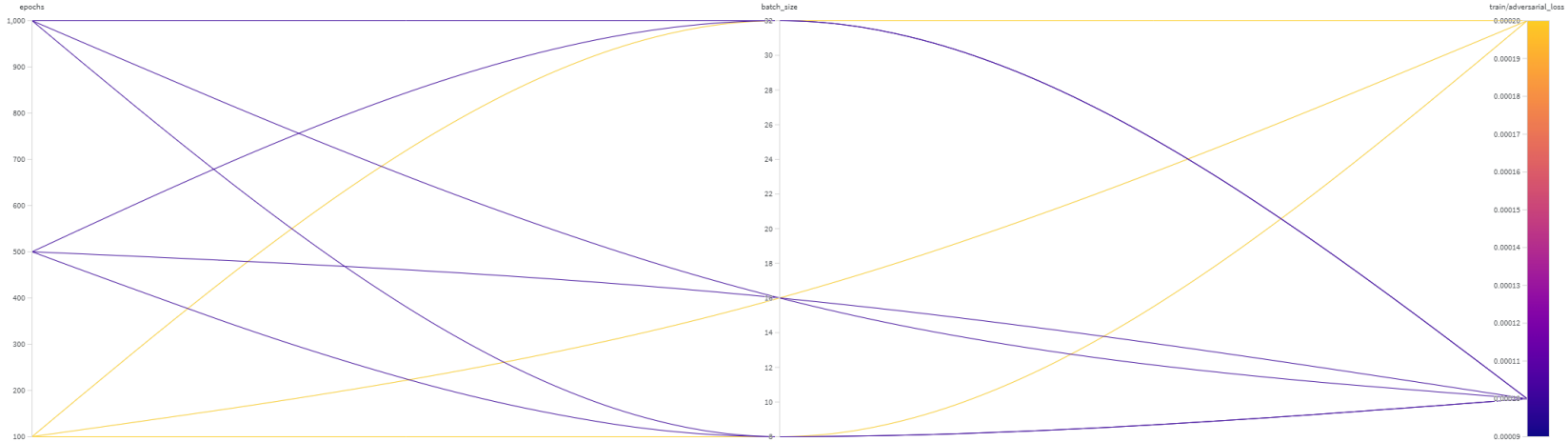

Figure 117: CIAE train adversarial loss comparison based on epochs and batch size

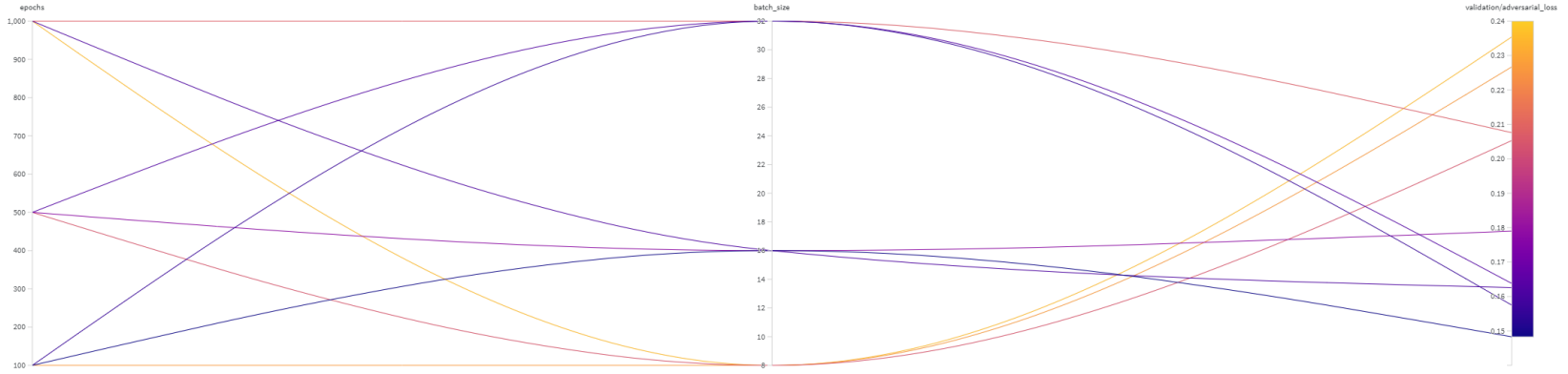

Figure 118: CIAE validation adversarial loss comparison based on epochs and batch size

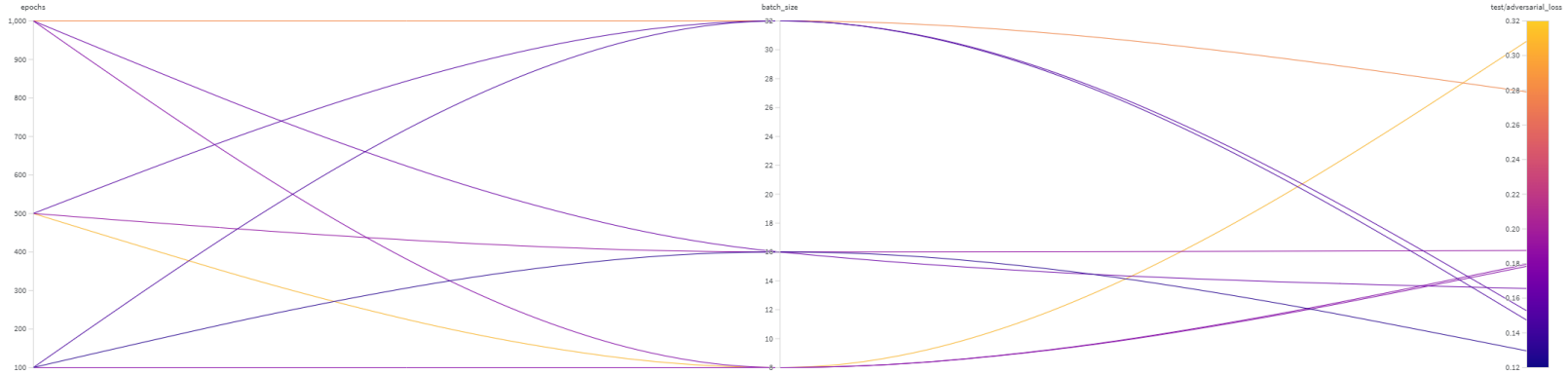

Figure 119: CIAE test adversarial loss comparison based on epochs and batch size

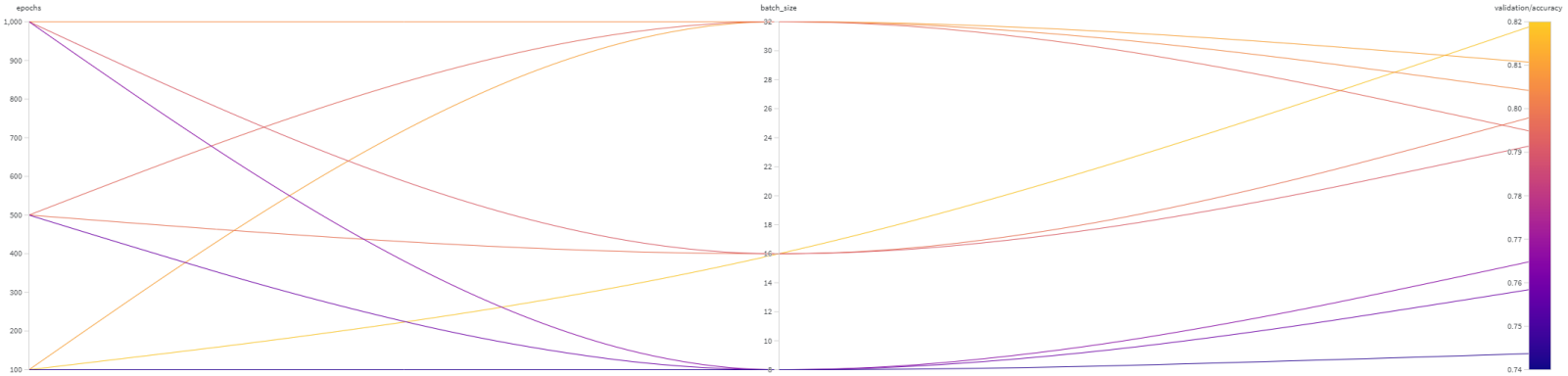

Figure 120: CIAE validation accuracy comparison based on epochs and batch size

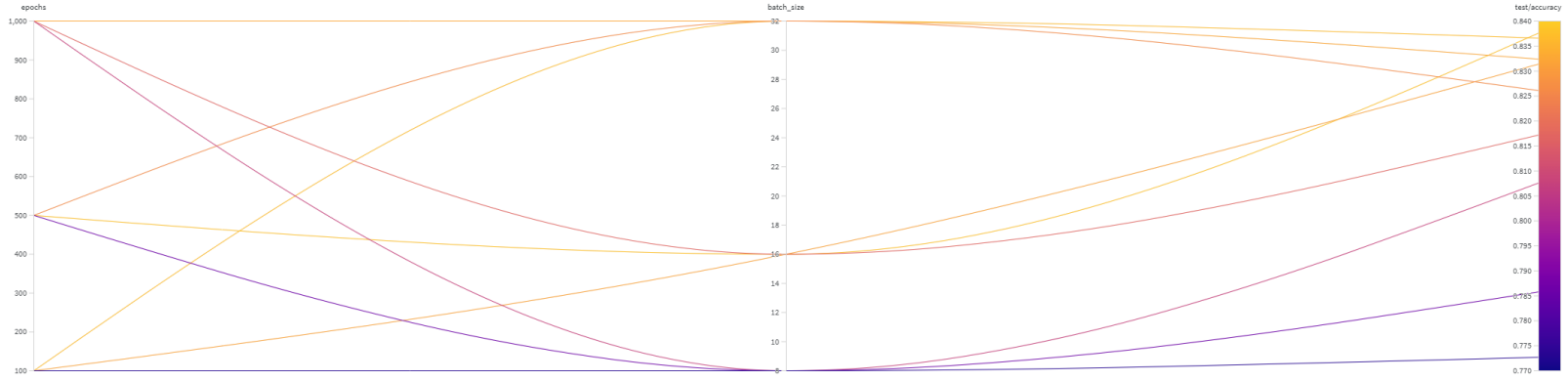

Figure 121: CIAE test accuracy comparison based on epochs and batch size

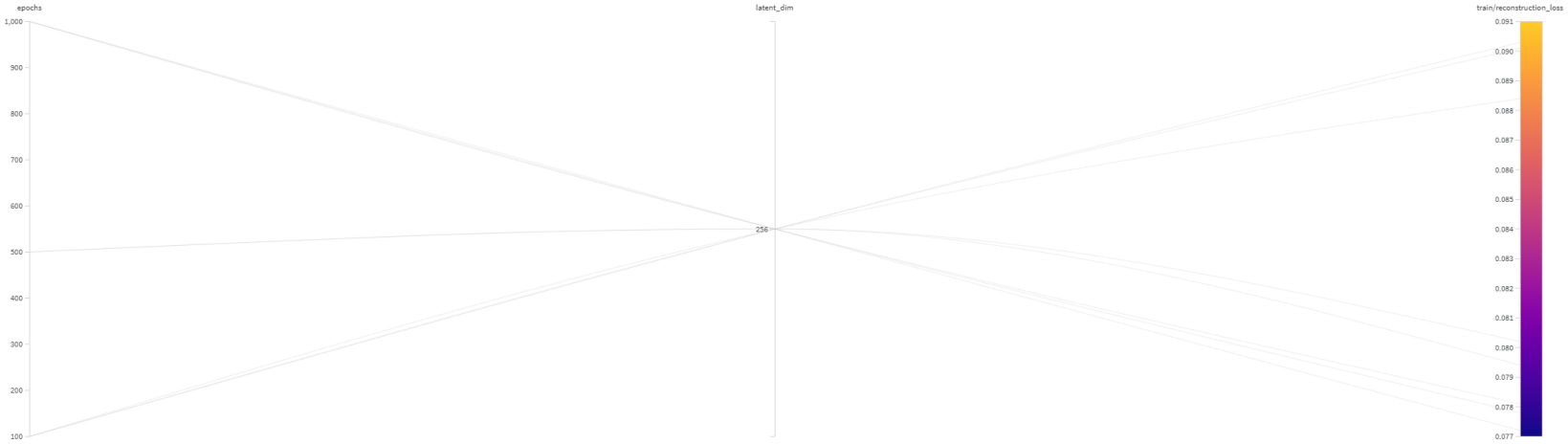

Figure 122: CIAE train reconstruction loss comparison based on epochs and latent dimension

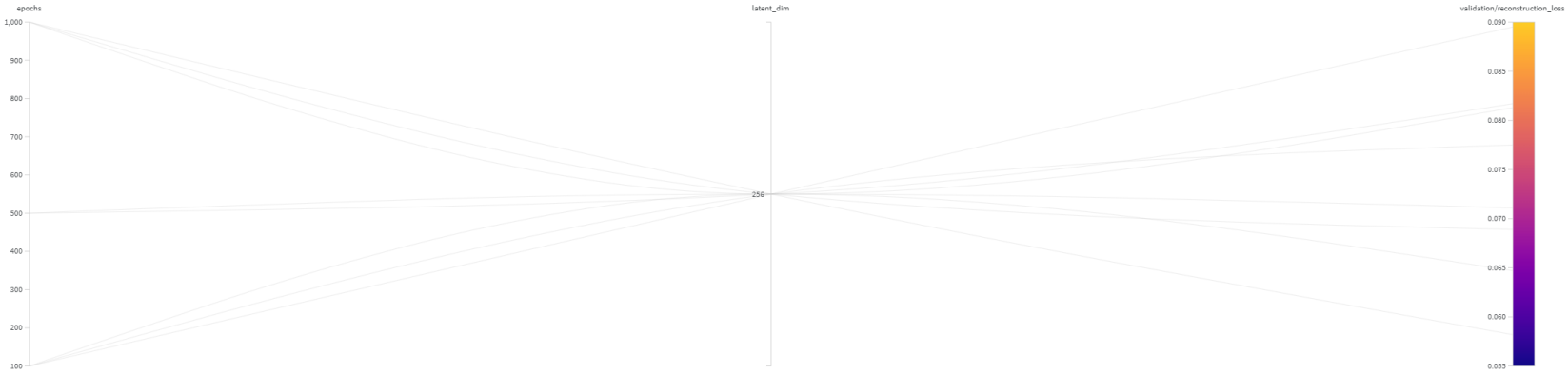

Figure 123: CIAE validation reconstruction loss comparison based on epochs and latent dimension

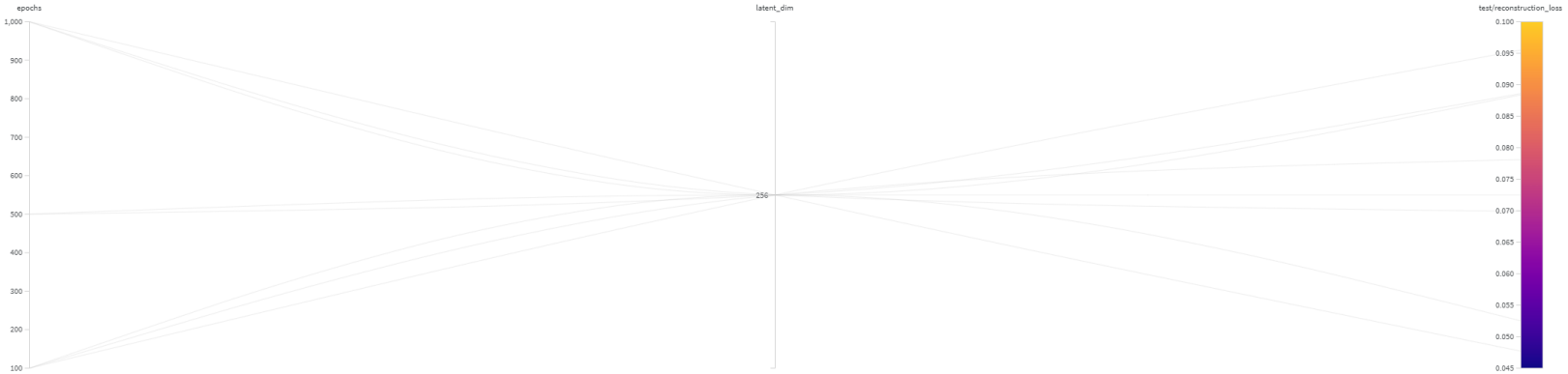

Figure 124: CIAE test reconstruction loss comparison based on epochs and latent dimension

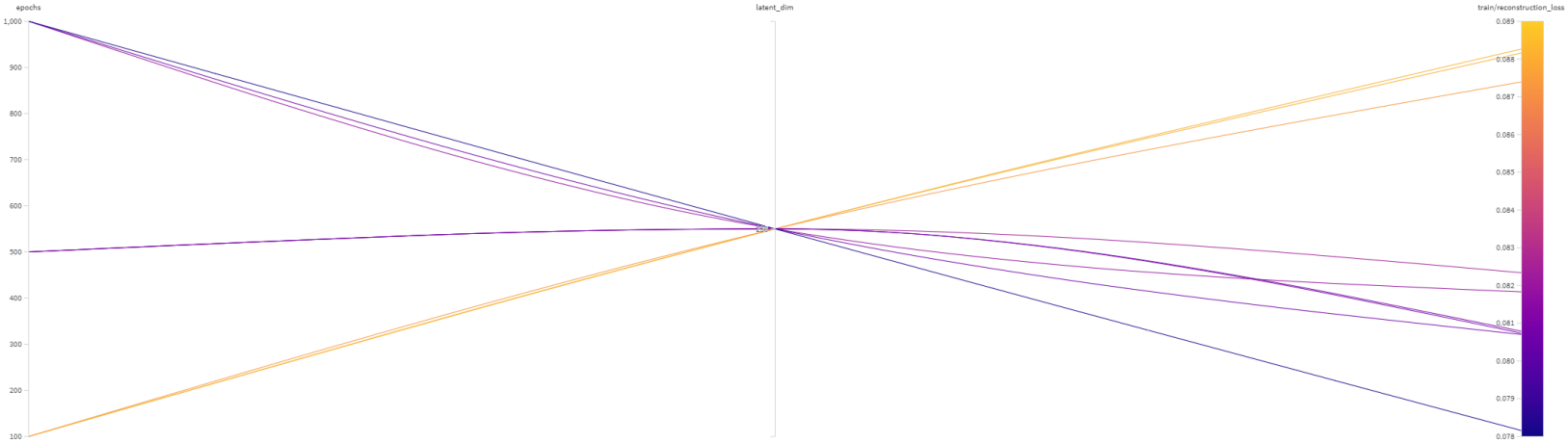

Figure 125: SOAE train reconstruction loss comparison based on epochs and latent dimension

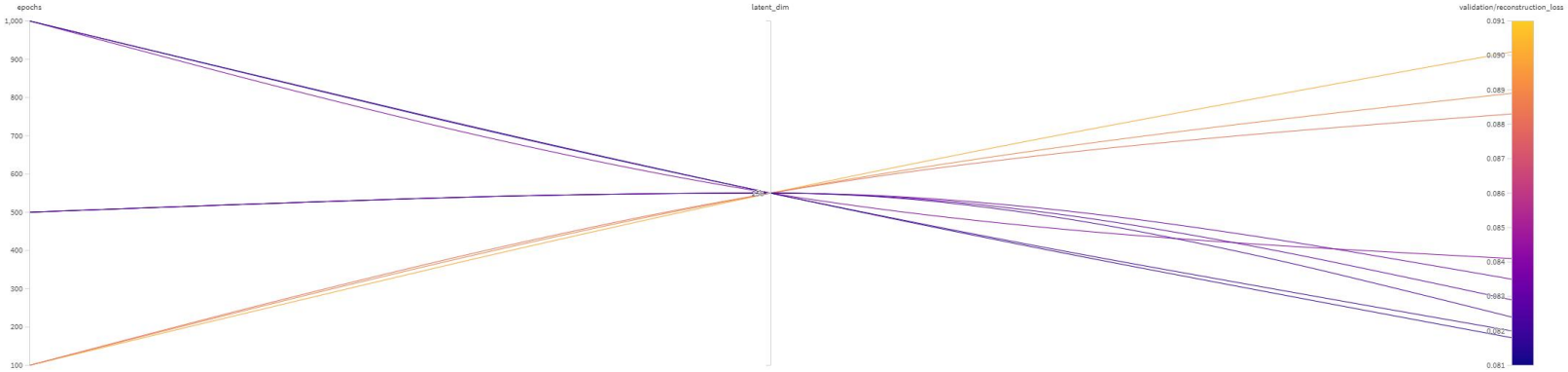

Figure 126: SOAE validation reconstruction loss comparison based on epochs and latent dimension

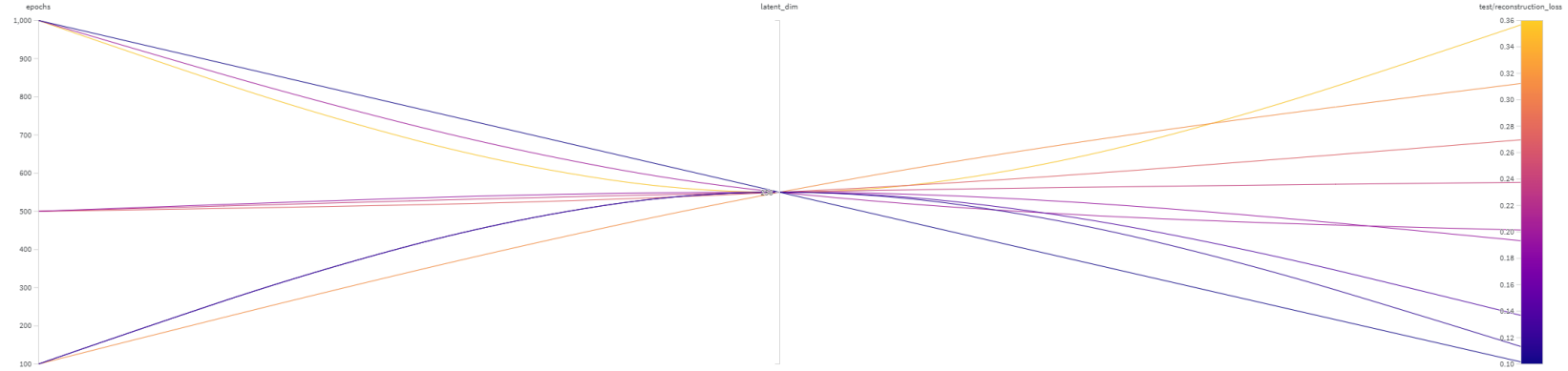

Figure 127: SOAE test reconstruction loss comparison based on epochs and latent dimension

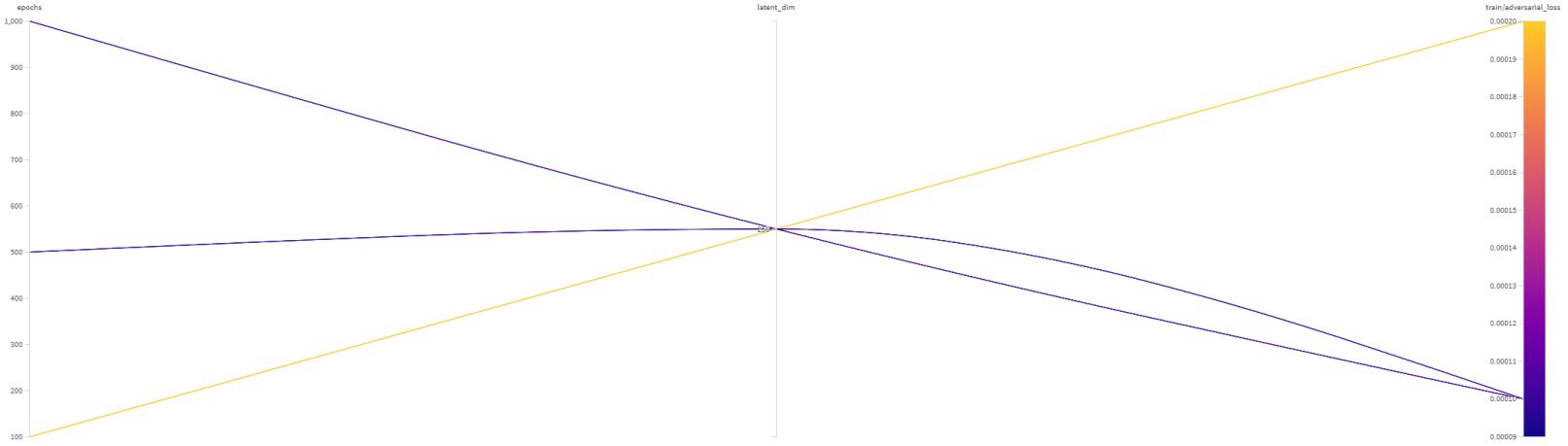

Figure 128: CIAE train adversarial loss comparison based on epochs and latent dimension

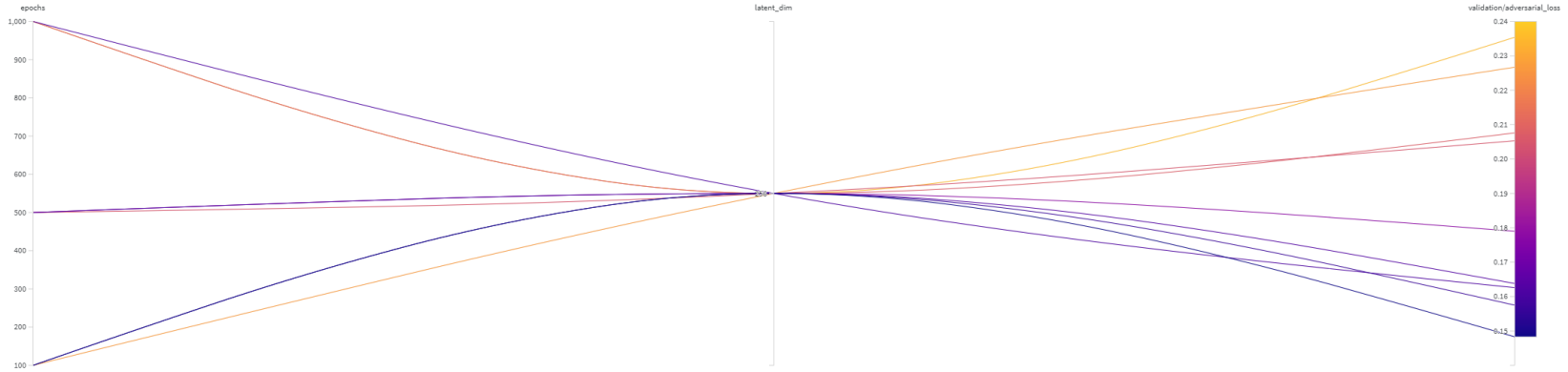

Figure 129: CIAE validation adversarial loss comparison based on epochs and latent dimension

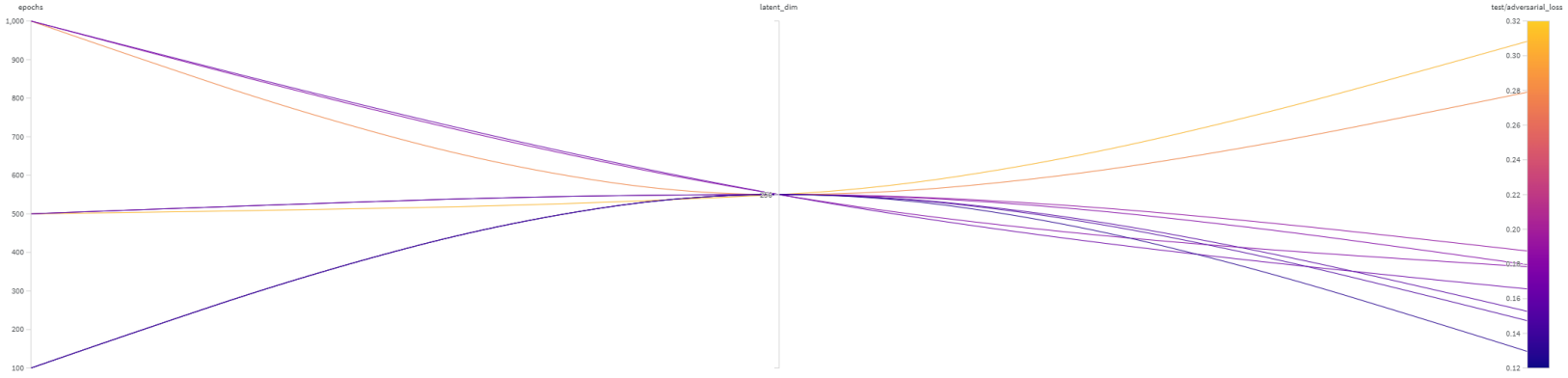

Figure 130: CIAE test adversarial loss comparison based on epochs and latent dimension

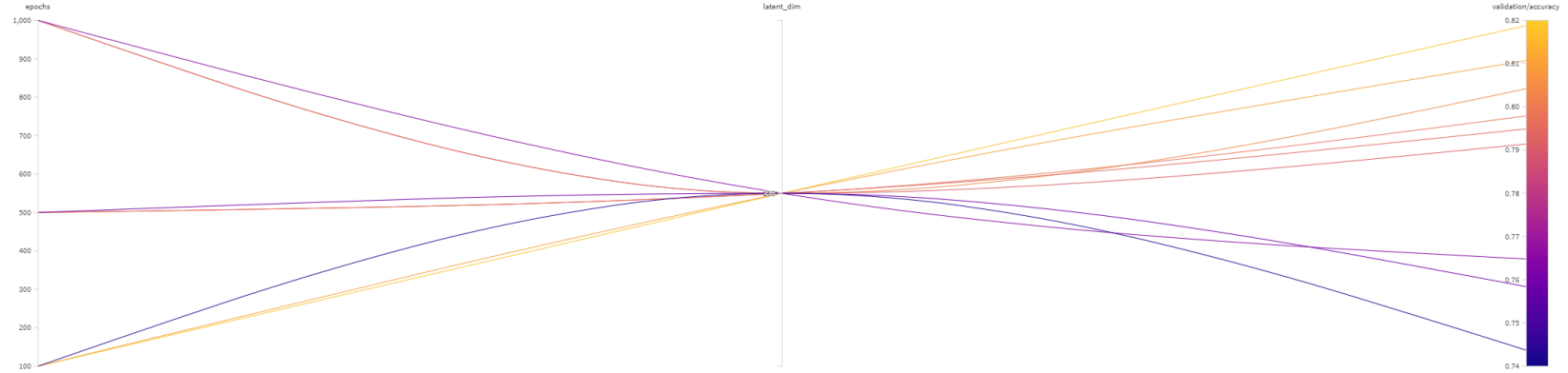

Figure 131: CIAE validation accuracy comparison based on epochs and latent dimension

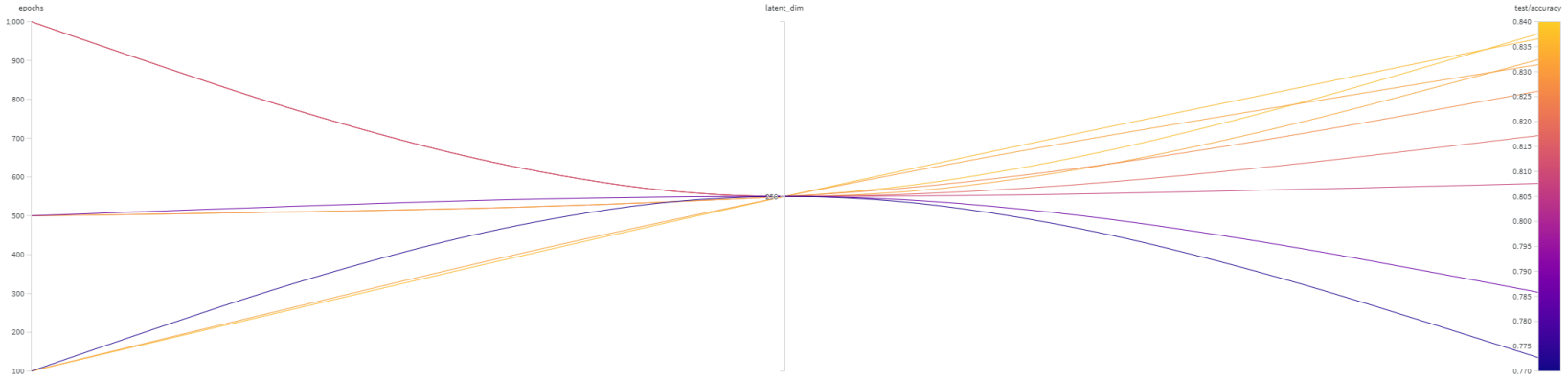

Figure 132: CIAE test accuracy comparison based on epochs and latent dimension

APPENDIX K

CIAE DENOISING MODEL OUTPUTS (Epoch = 1000)

The denoising CIAE model results in terms of reconstruction loss and domain classifier results have been shown in APPENDIX K for 1000 epochs.

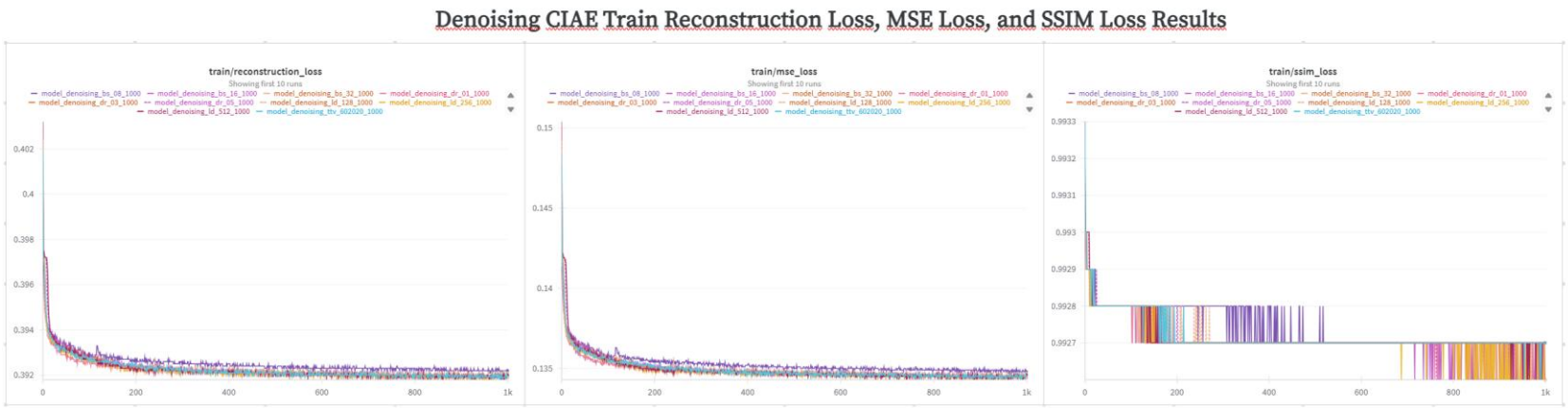

Figure 133: Denoising CIAE train reconstruction loss, MSE loss, and SSIM loss results for 1000 epochs

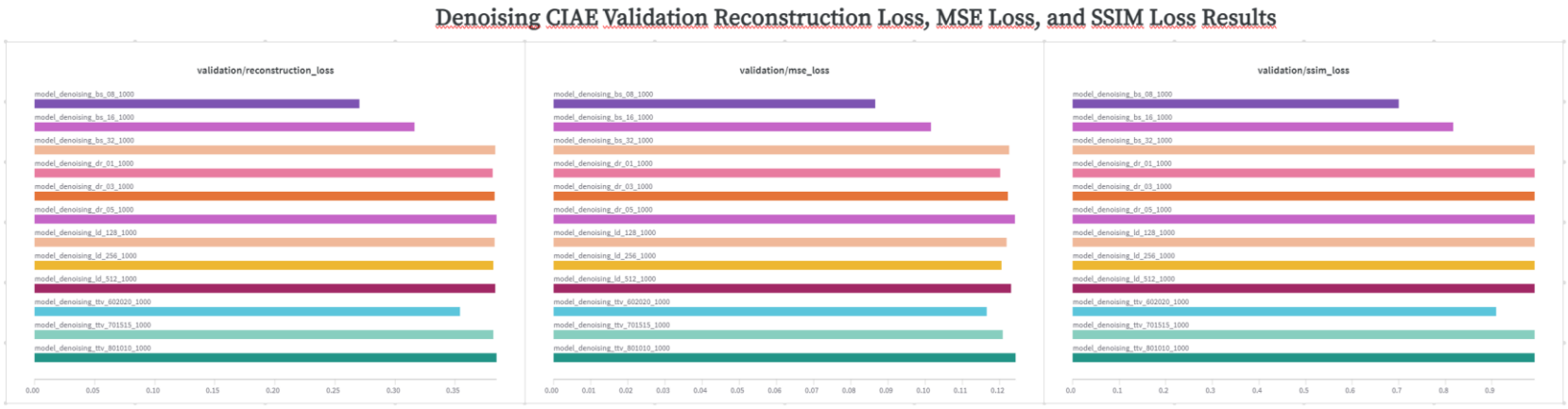

Figure 134: Denoising CIAE validation reconstruction loss, MSE loss, and SSIM loss results for 1000 epochs

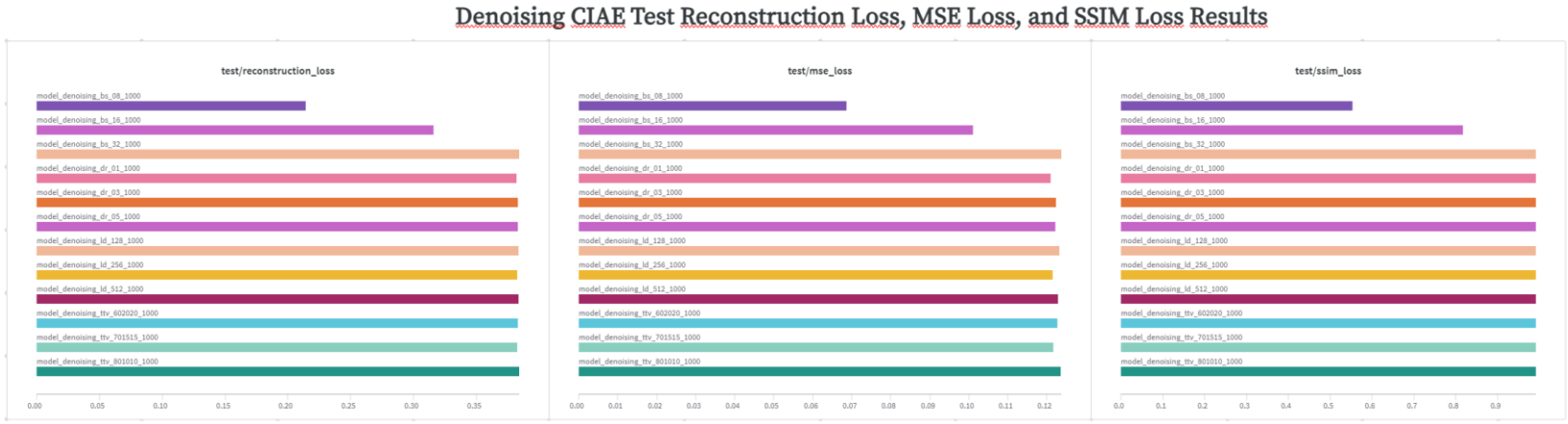

Figure 135: Denoising CIAE test reconstruction loss, MSE loss, and SSIM loss results for 1000 epochs

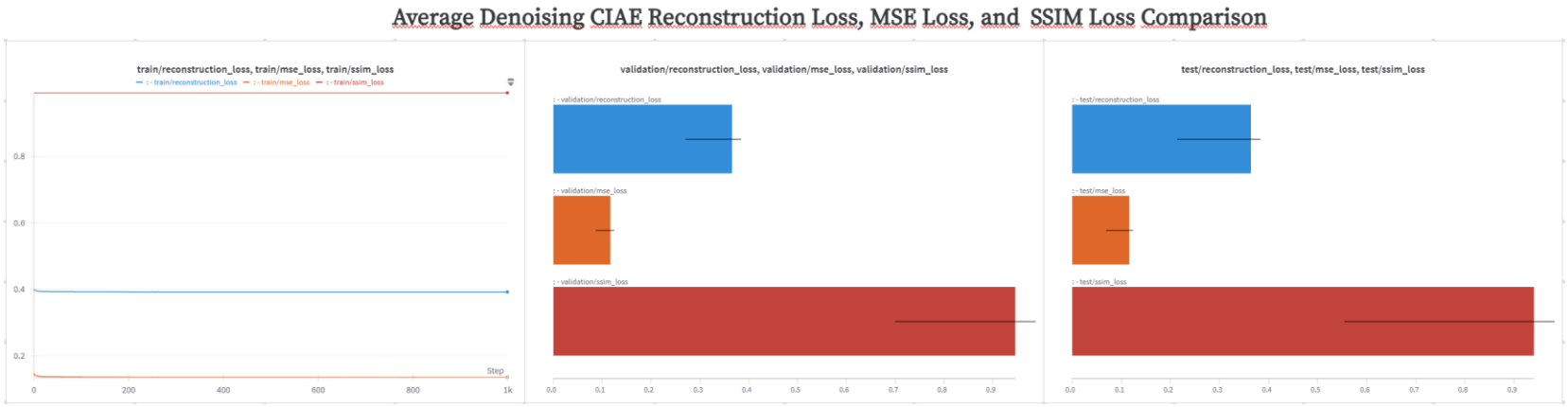

Figure 136: Average Denoising CIAE reconstruction loss, MSE loss, and SSIM loss comparison for 1000 epochs

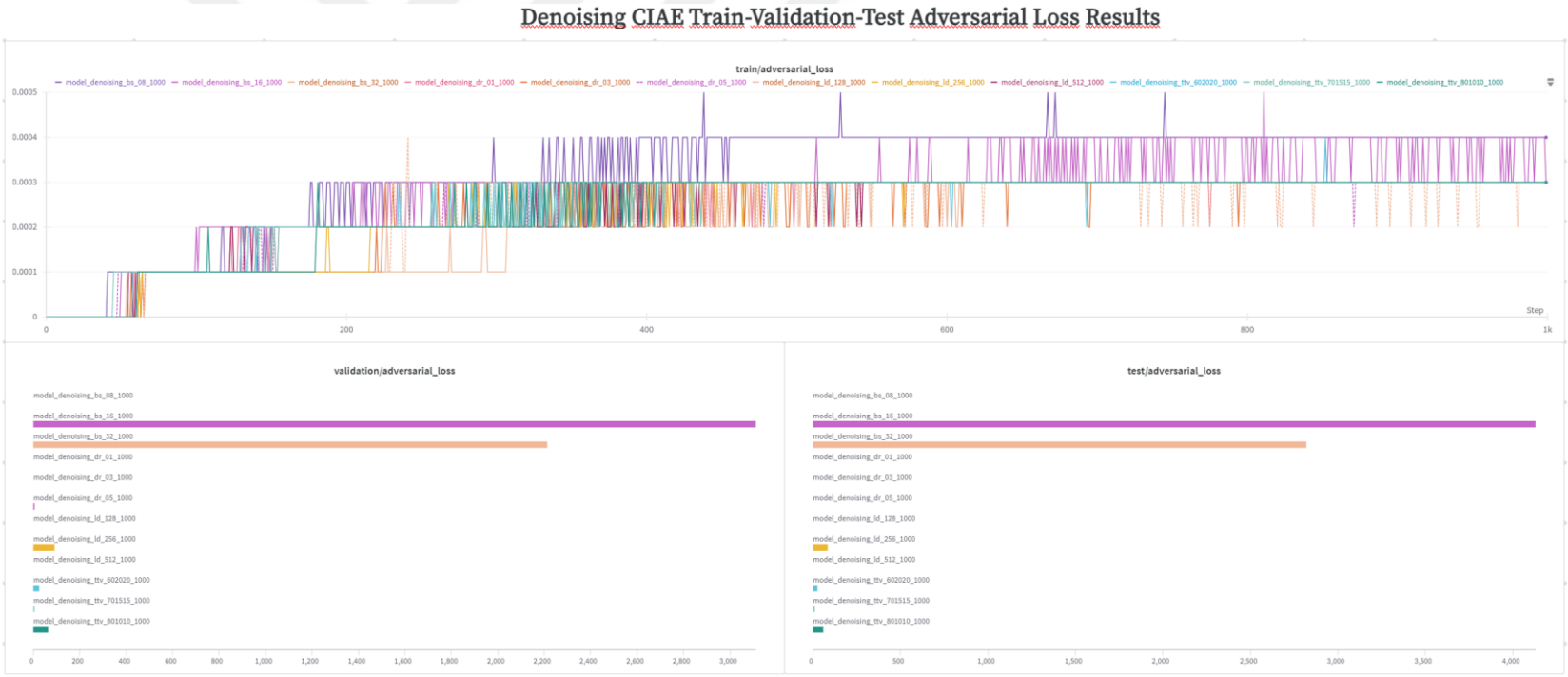

Figure 137: Denoising CIAE train-validation-test adversarial loss results for 1000 epochs

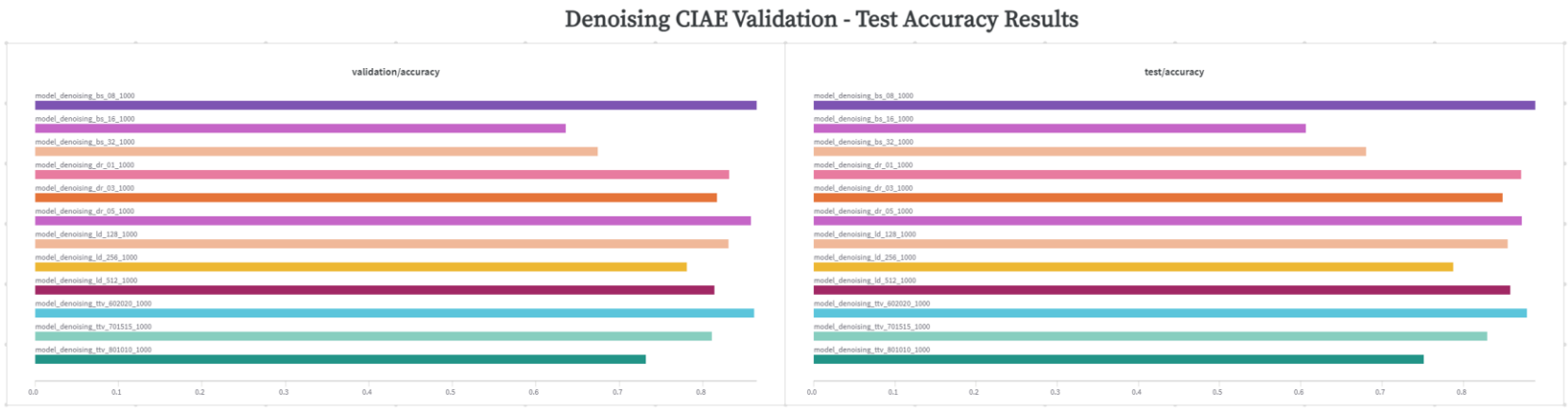

Figure 138: Denoising CIAE validation - test accuracy results for 1000 epochs

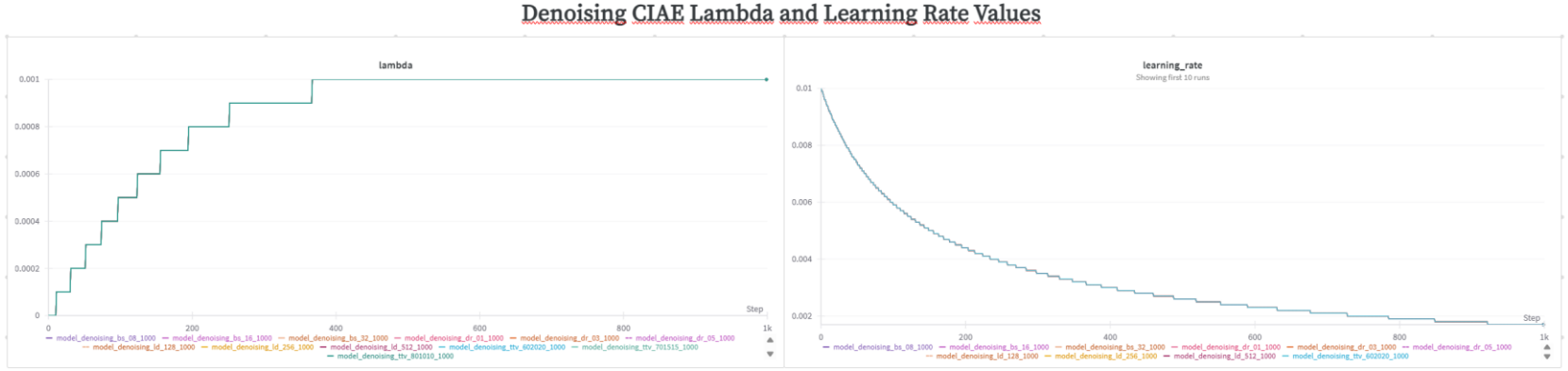

Figure 139: Denoising CIAE lambda and learning rate values for 1000 epochs

APPENDIX L

CIAE DENOISING MODEL OUTPUTS (Epoch = 1000)

The CIAE denoising model outputs have been shown in APPENDIX L for 1000 epochs.

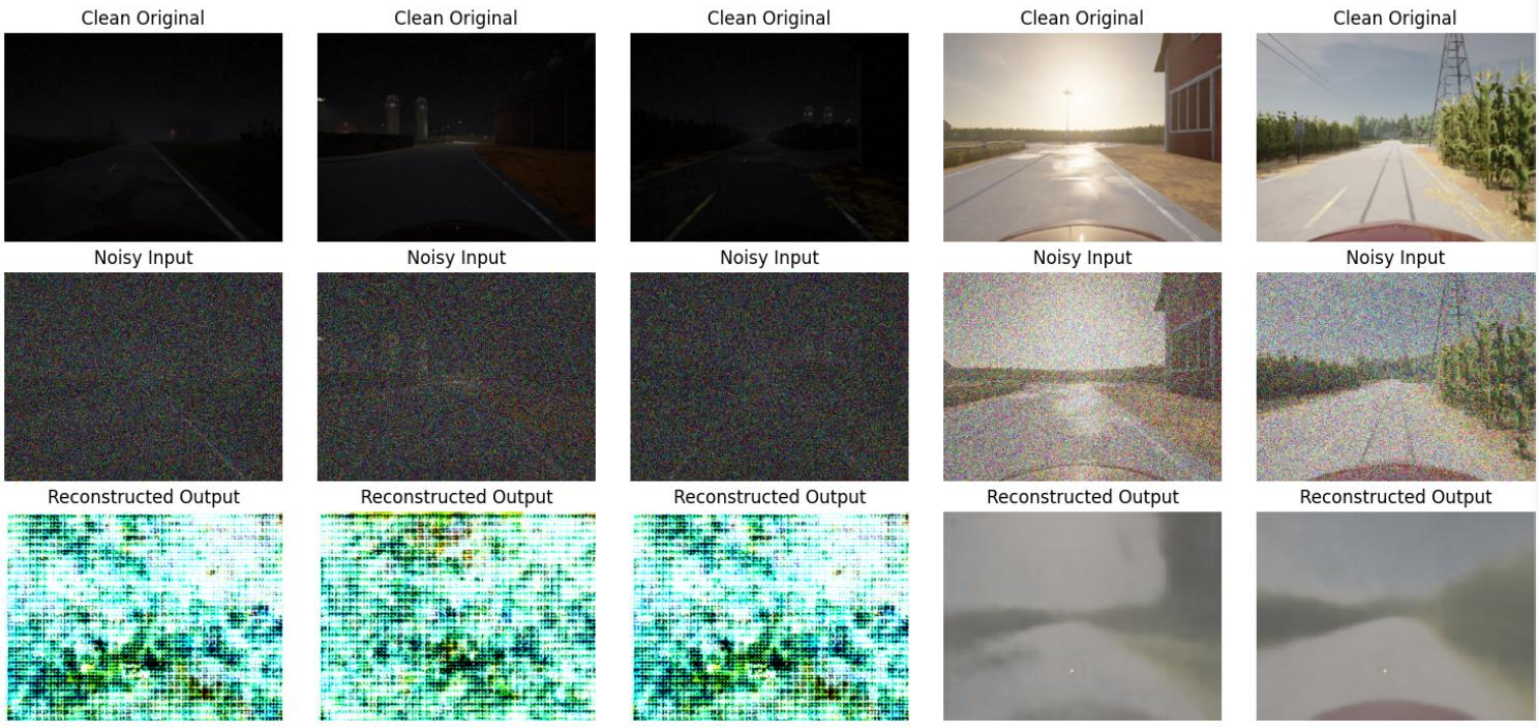

Figure 140: CIAE_model_denoising_bs_08_1000

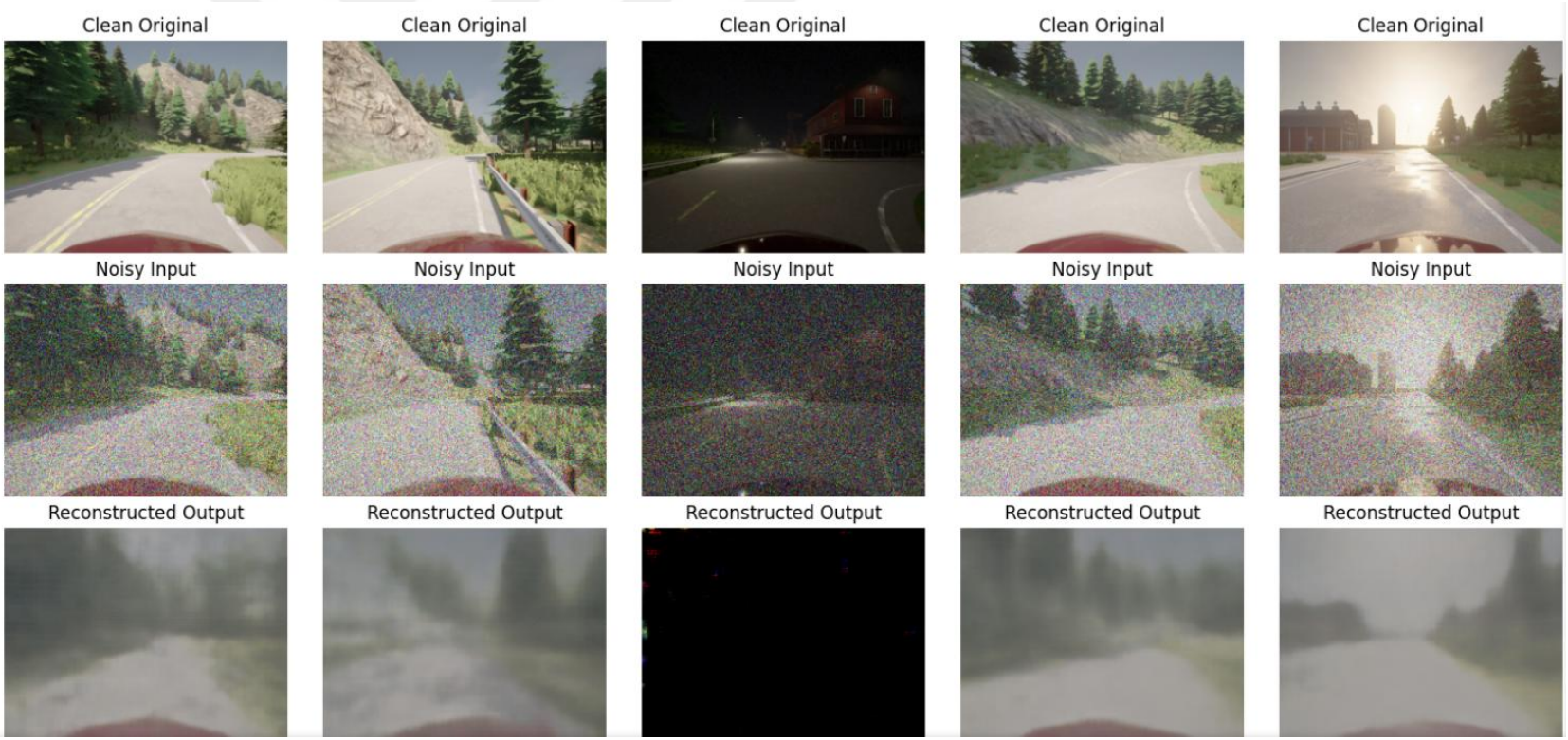

Figure 141: CIAE_model_denoising_bs_16_1000

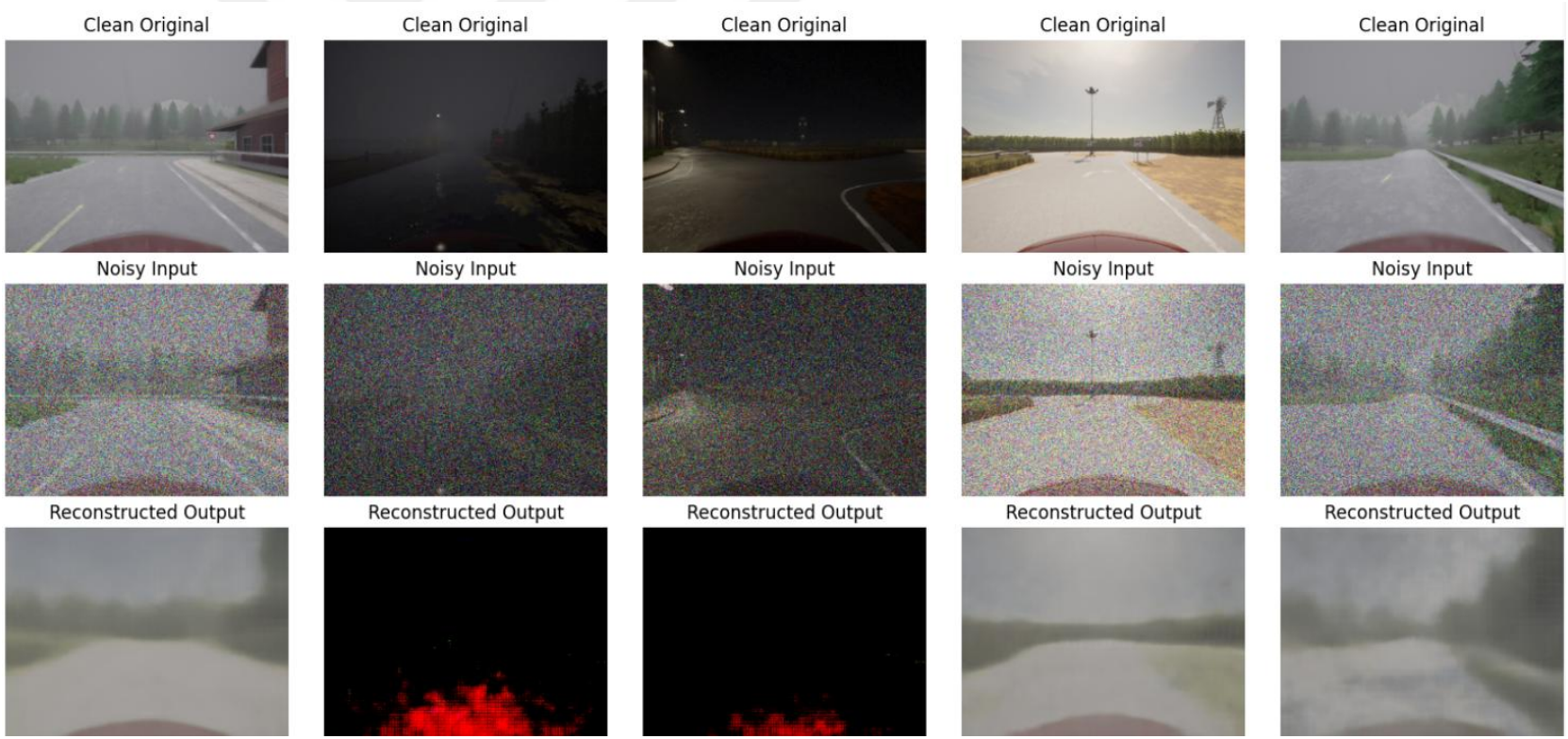

Figure 142: CIAE_model_denoising_bs_32_1000

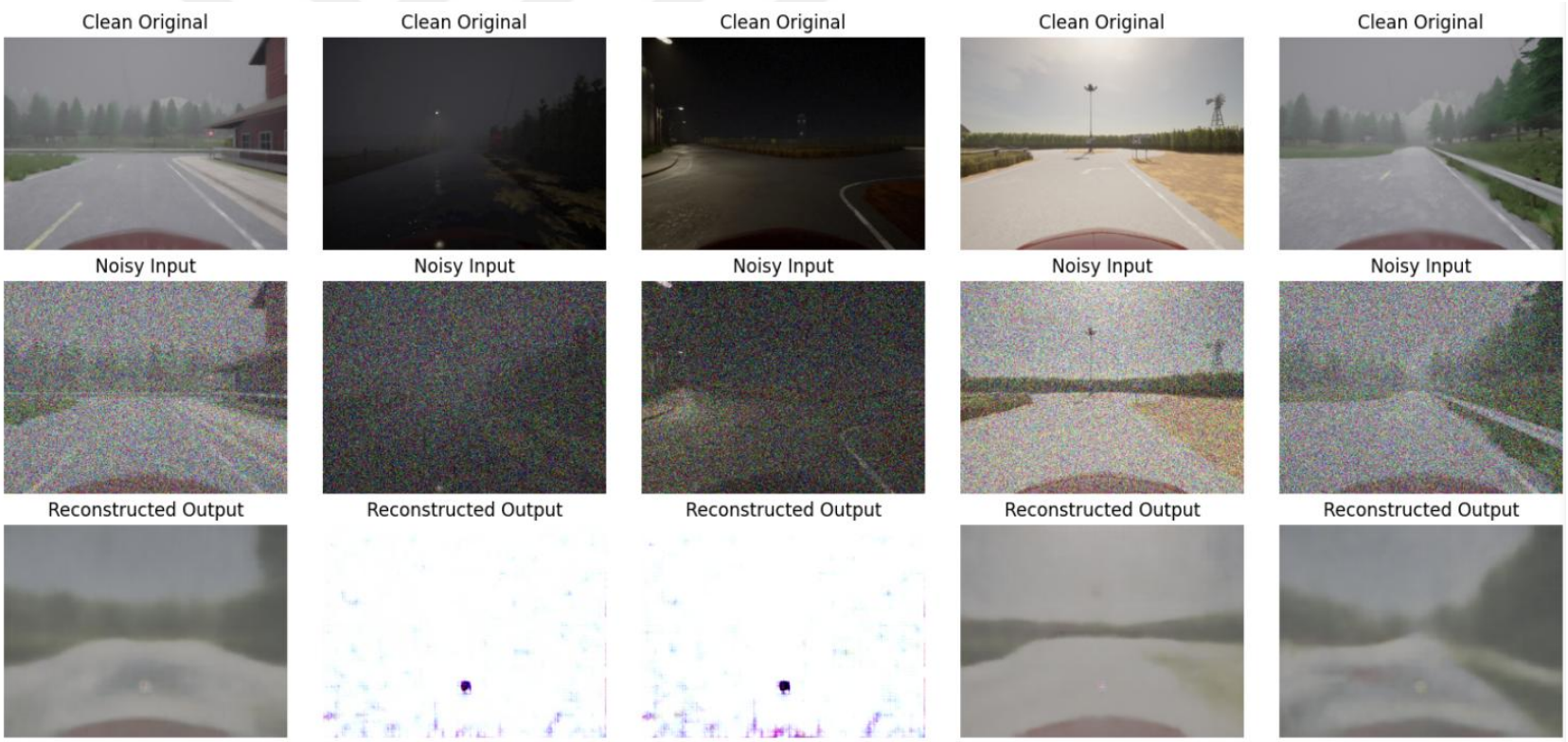

Figure 143: CIAE_model_denoising_dr_01_1000

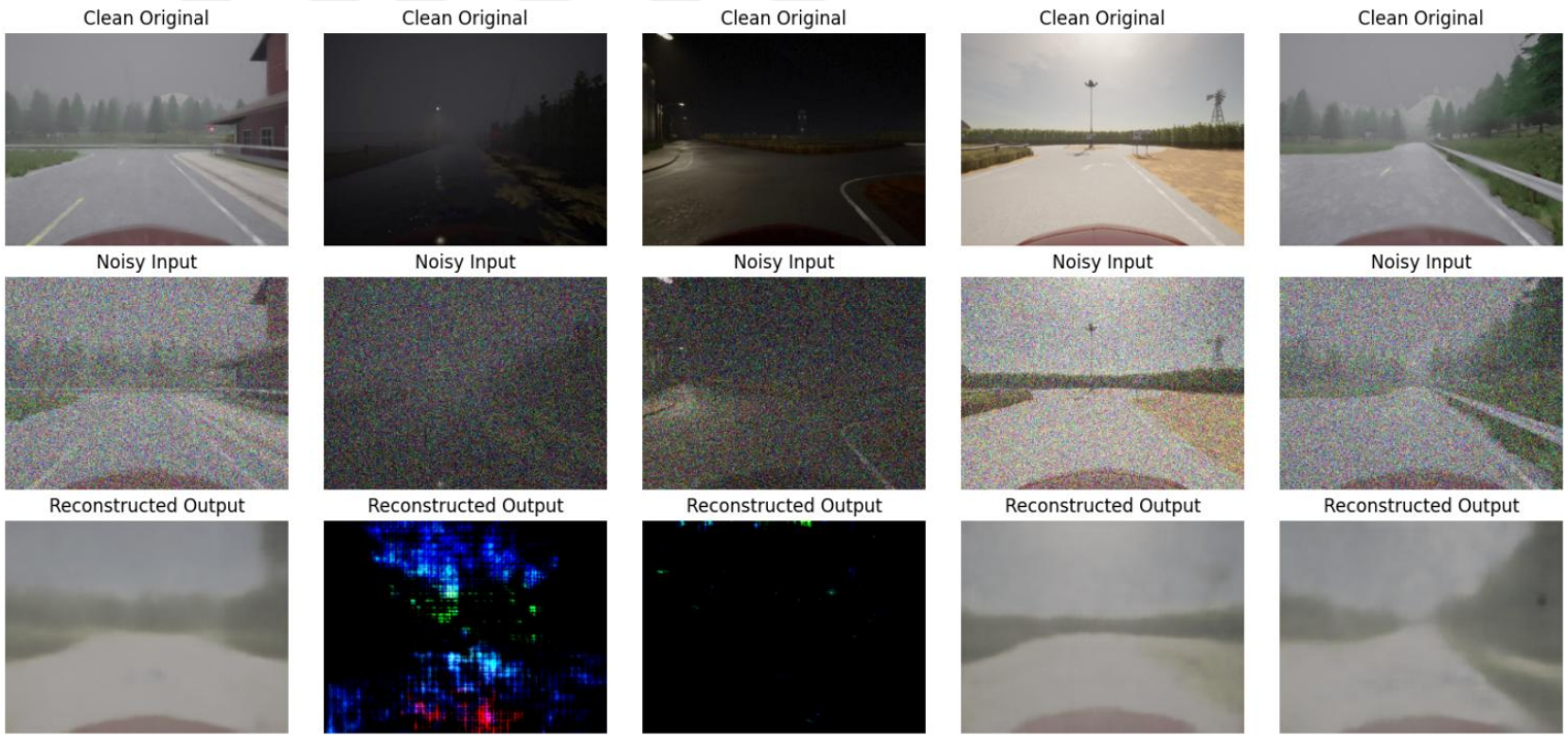

Figure 144: CIAE_model_denoising_dr_03_1000

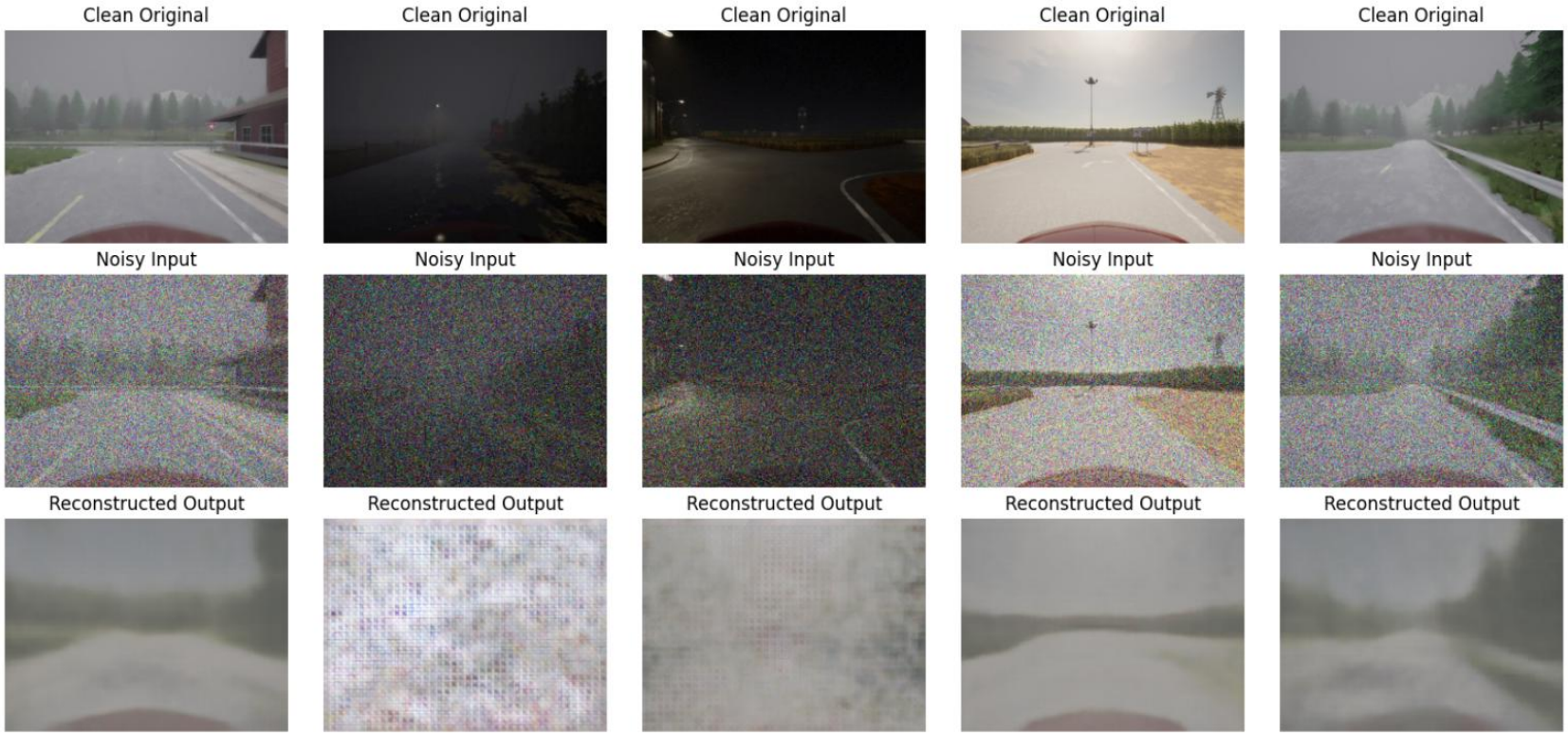

Figure 145: CIAE_model_denoising_dr_05_1000

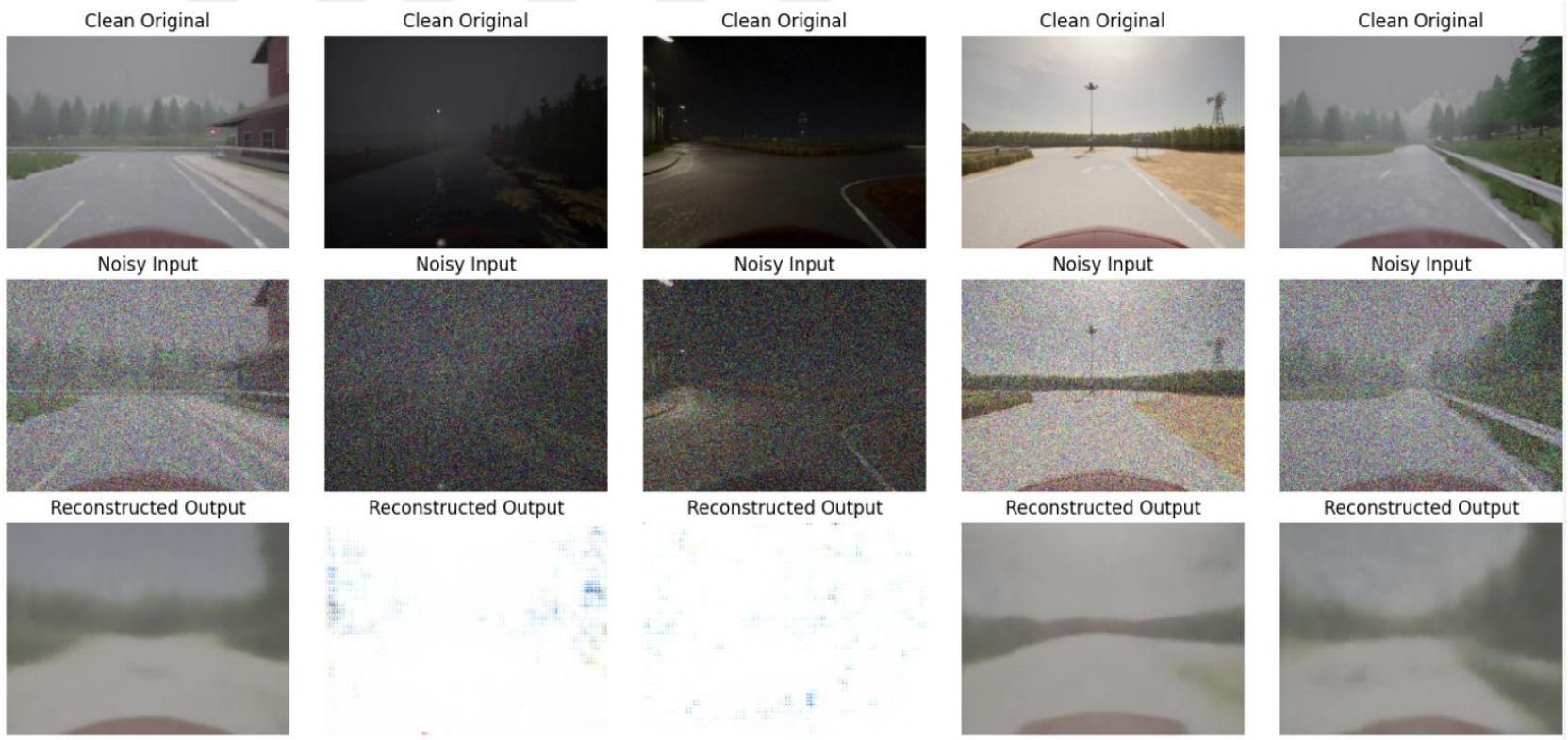

Figure 146: CIAE_model_denoising_ld_128_1000

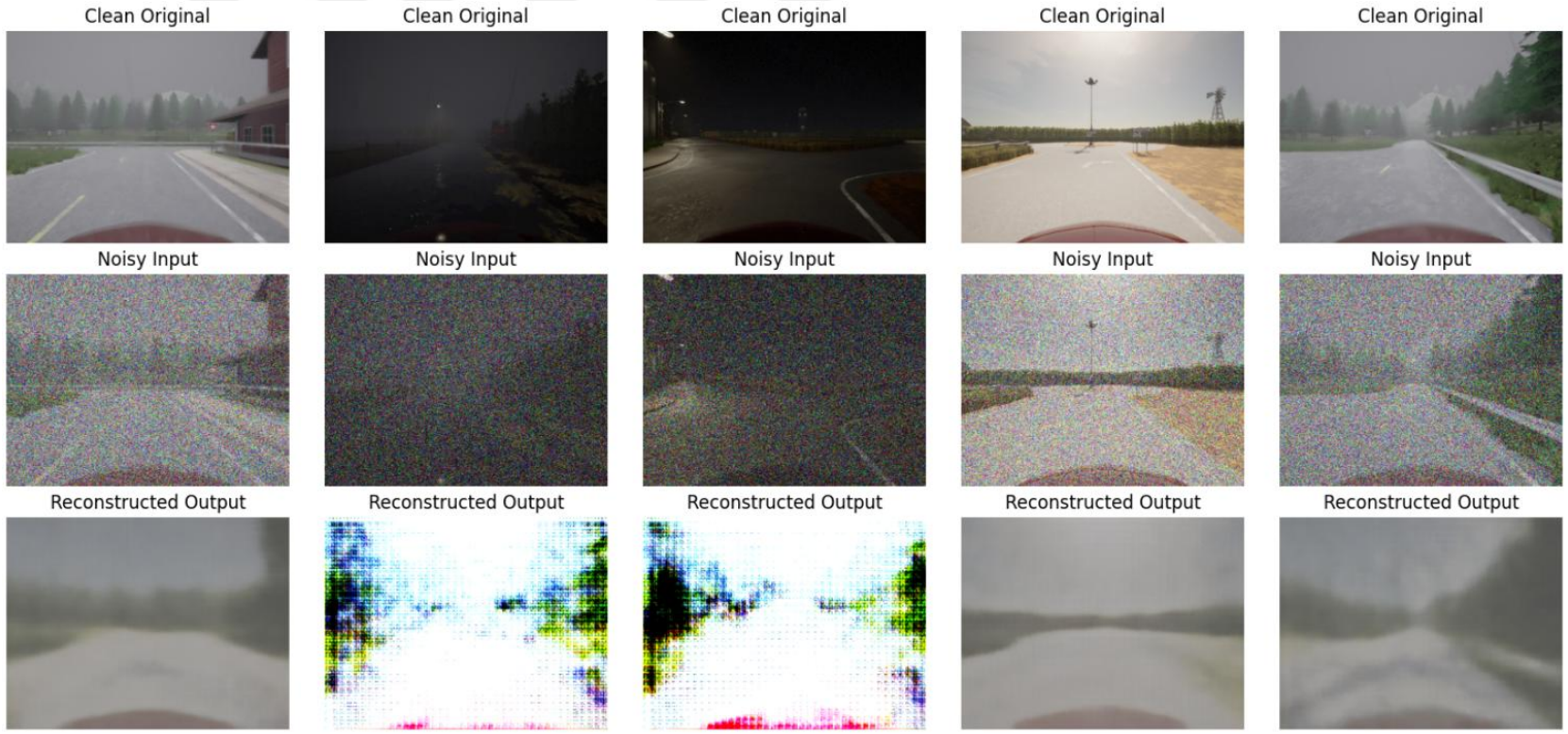

Figure 147: CIAE_model_denoising_ld_256_1000

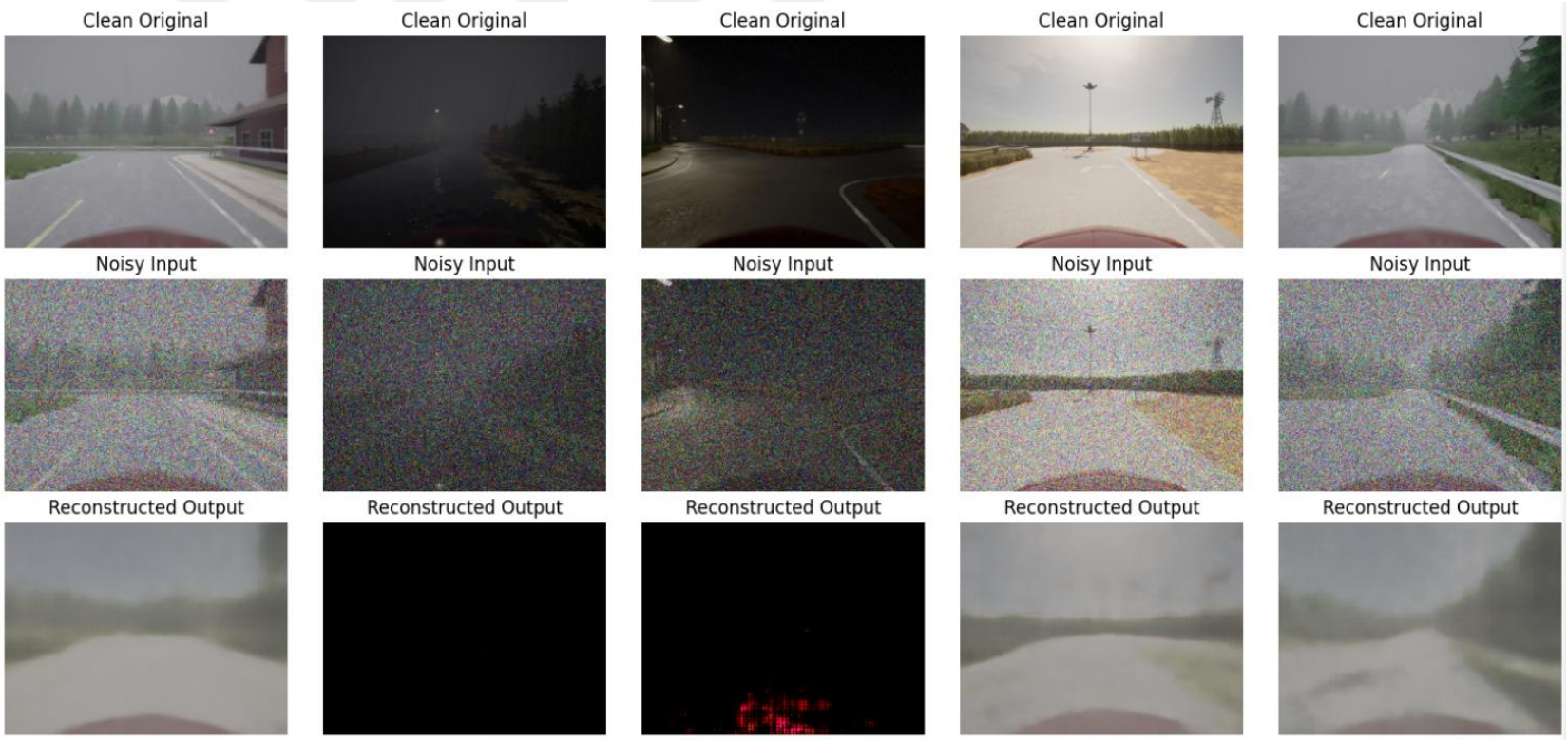

Figure 148: CIAE_model_denoising_ld_512_1000

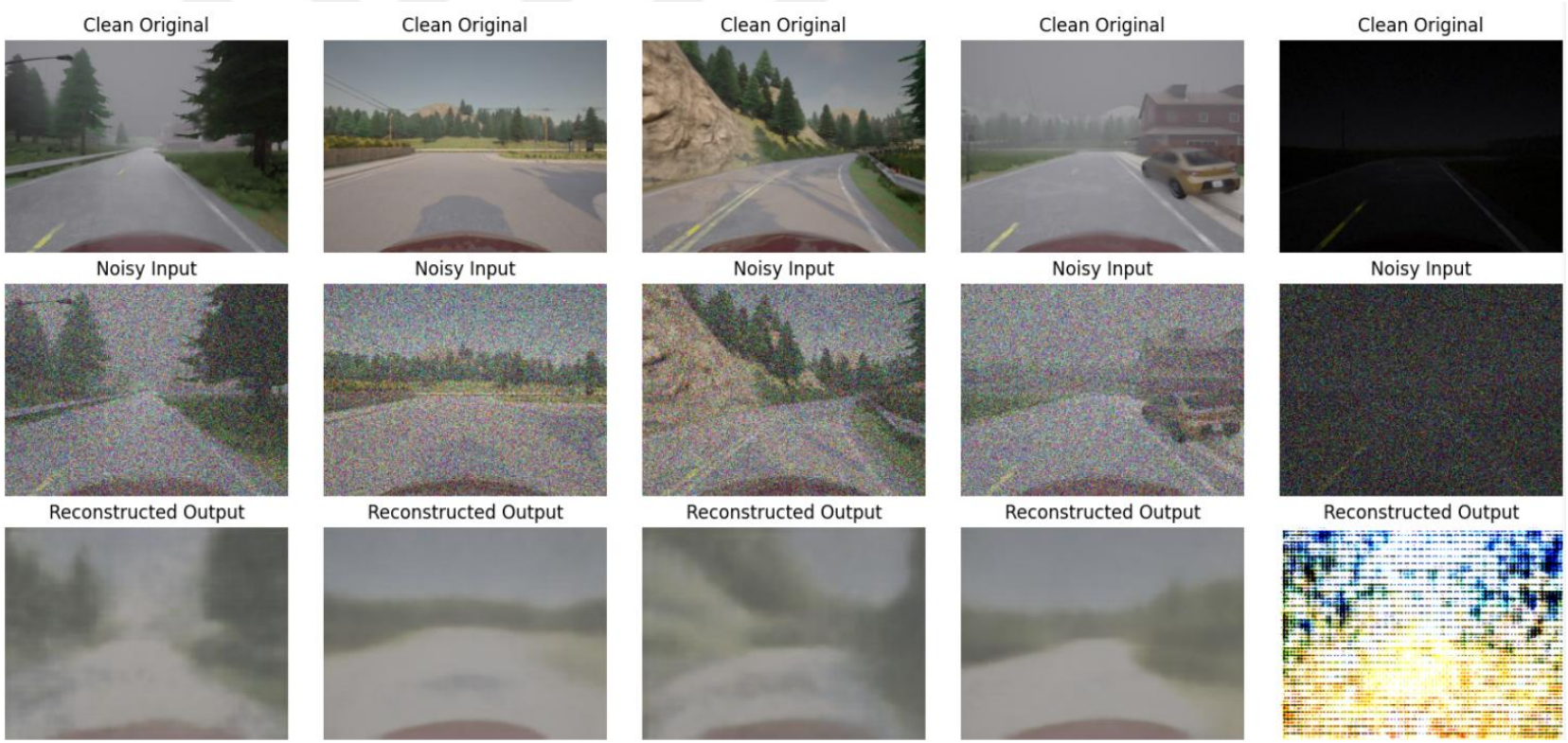

Figure 149: CIAE_model_denoising_ttv_602020_1000

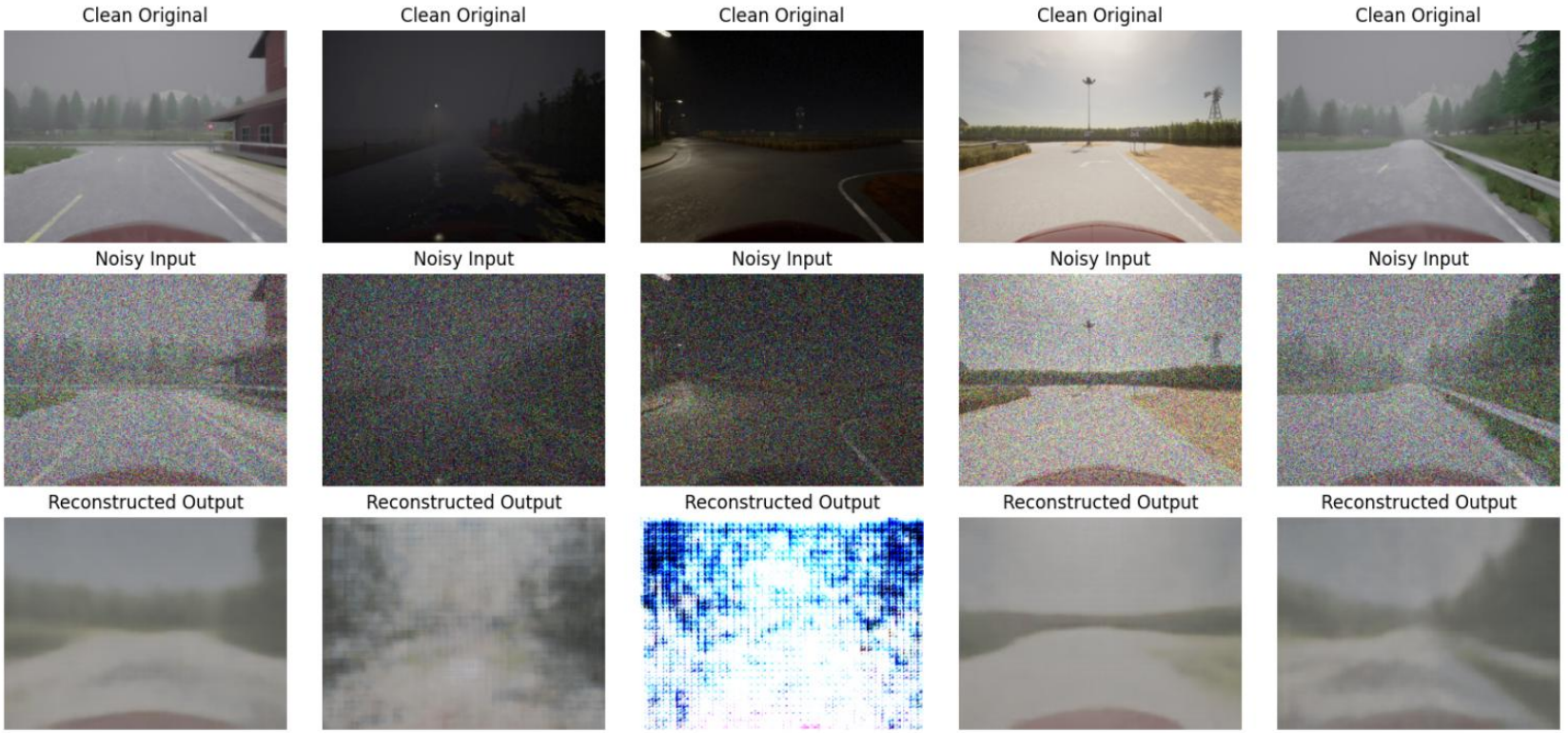

Figure 150: CIAE_model_denoising_ttv_701515_1000

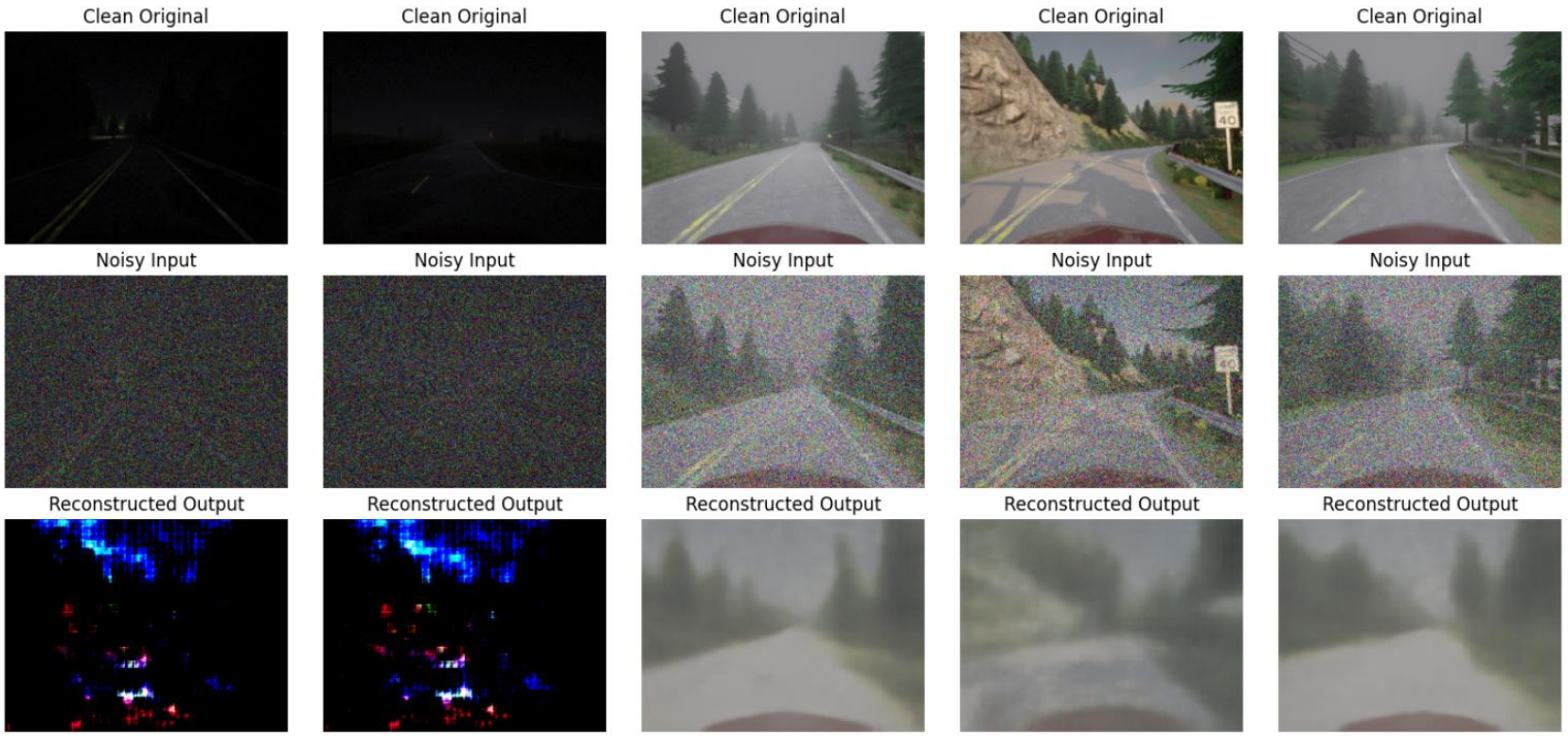

Figure 151: CIAE_model_denoising_ttv_801010_1000

APPENDIX M

CIAE VS DENOISING CIAE MODEL COMPARISONS (Epoch = 1000)

The CIAE and denoising CIAE model outputs have been compared in APPENDIX M for 1000 epochs.

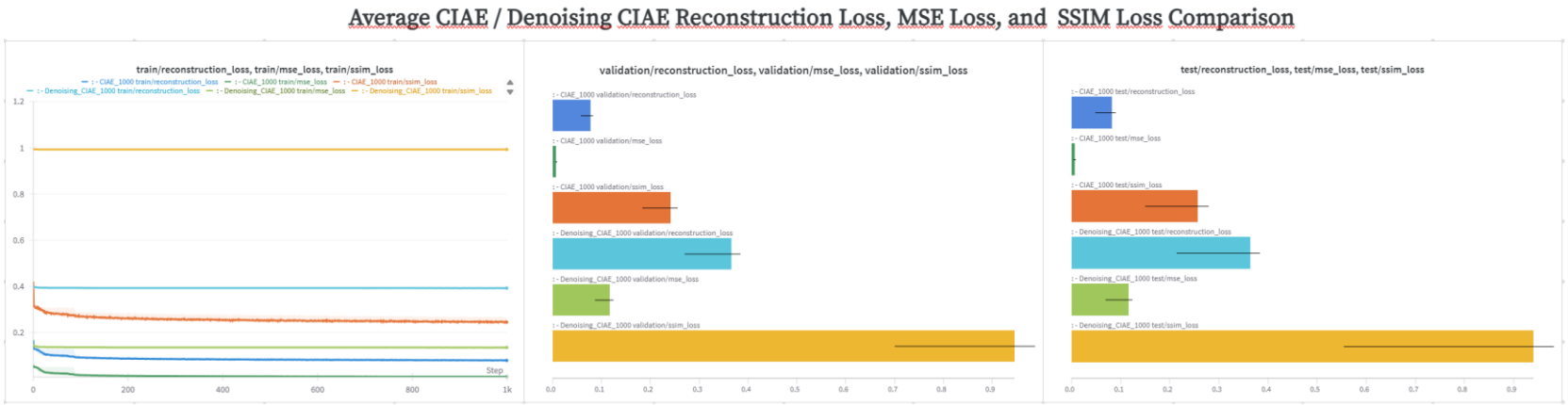

Figure 152: Average CIAE/Denoising CIAE reconstruction loss, MSE loss, and SSIM loss comparison for 1000 epochs

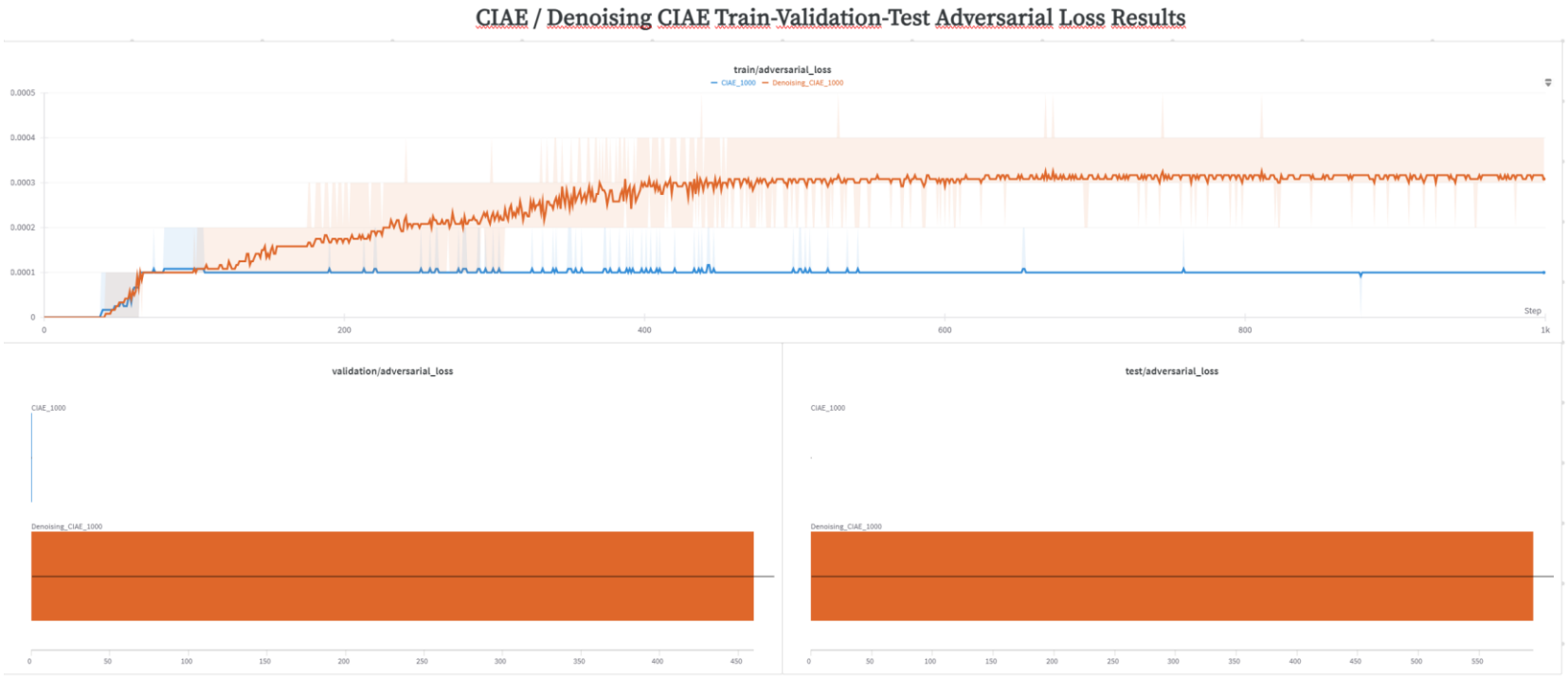

Figure 153: CIAE/Denoising CIAE train-validation-test adversarial loss results for 1000 epochs

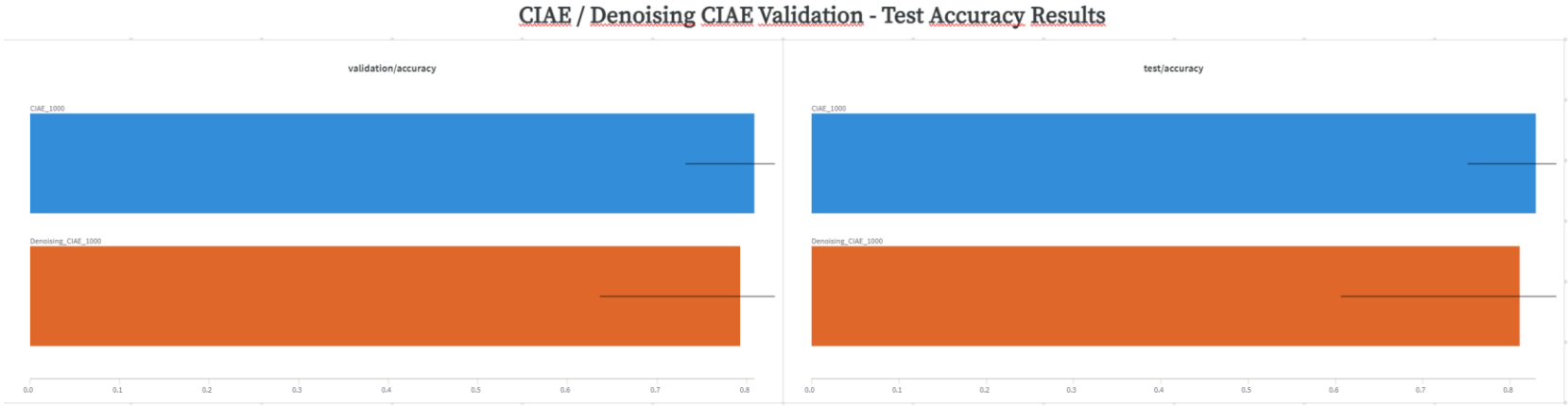

Figure 154: CIAE/Denoising CIAE validation - test accuracy results for 1000 epochs