**Appendix A: Code Samples**

## Code Sample 1: Vehicle Cropping Strategy

def generate\_cnn\_training\_dataset():  
 for each sequence:  
 for each frame:  
 for each annotated vehicle:  
 # 1. Extract bounding box  
 crop = frame[y1:y2, x1:x2]  
   
 # 2. Handle edge cases  
 crop = clamp\_to\_image\_bounds(crop)  
   
 # 3. Resize to fixed dimensions  
 crop = cv2.resize(crop, (64, 64))  
   
 # 4. Normalize to [0,1]  
 crop = crop.astype('float32') / 255.0  
   
 # 5. Store with label  
 dataset.append((crop, class\_idx))

## Code Sample 2: CNN Architecture

VehicleClassifier(  
 features: Sequential(  
 # Block 1: Initial feature extraction  
 Conv2d(3 → 32, kernel=3×3, padding=1)  
 BatchNorm2d(32)  
 ReLU()  
 MaxPool2d(2×2) # 64×64 → 32×32  
   
 # Block 2: Mid-level features  
 Conv2d(32 → 64, kernel=3×3, padding=1)  
 BatchNorm2d(64)  
 ReLU()  
 MaxPool2d(2×2) # 32×32 → 16×16  
   
 # Block 3: High-level features  
 Conv2d(64 → 128, kernel=3×3, padding=1)  
 BatchNorm2d(128)  
 ReLU()  
 MaxPool2d(2×2) # 16×16 → 8×8  
   
 # Block 4: Abstract features  
 Conv2d(128 → 256, kernel=3×3, padding=1)  
 BatchNorm2d(256)  
 ReLU()  
 MaxPool2d(2×2) # 8×8 → 4×4  
 )  
   
 classifier: Sequential(  
 Flatten() # 256×4×4 = 4,096 features  
 Linear(4096 → 256)  
 ReLU()  
 Dropout(0.5)  
 Linear(256 → num\_classes)  
 )  
)

## Code Sample 3: LMV/HMV Mapping

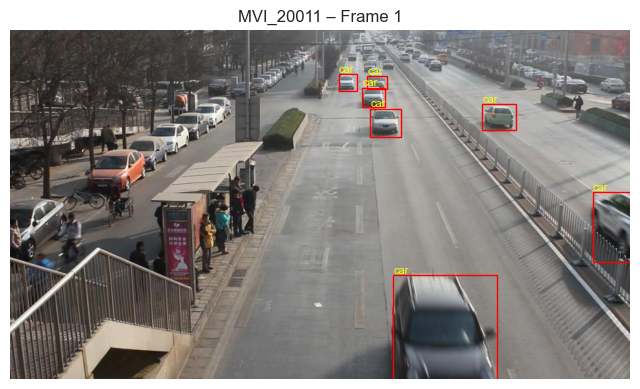
LMV\_CLASSES = {'car', 'van', 'others', 'motor'} # Light Motor Vehicles  
HMV\_CLASSES = {'bus', 'truck'} # Heavy Motor Vehicles  
  
def map\_to\_lmv\_hmv(fine\_class: str) -> str:  
 if fine\_class.lower() in LMV\_CLASSES:  
 return "LMV"  
 elif fine\_class.lower() in HMV\_CLASSES:  
 return "HMV"  
 else:  
 return "Unknown"

## Code Sample 4: Traffic Forecasting

# Feature engineering: Use previous K frames to predict next frame  
K = 5 # History length  
for lag in range(1, K+1):  
 df[f'total\_lag\_{lag}'] = df['total\_count'].shift(lag)  
  
X = df[[f'total\_lag\_{i}' for i in range(1, K+1)]].values  
y = df['total\_count'].values  
  
# Train/test split (temporal, no shuffling)  
split\_idx = int(0.8 \* len(X))  
X\_train, X\_test = X[:split\_idx], X[split\_idx:]  
y\_train, y\_test = y[:split\_idx], y[split\_idx:]  
  
# Random Forest Regressor  
model = RandomForestRegressor(n\_estimators=200, random\_state=42)  
model.fit(X\_train, y\_train)

**Appendix B: Figures and Visualizations**

## Figure 1: Annotated Frame from MVI\_20011



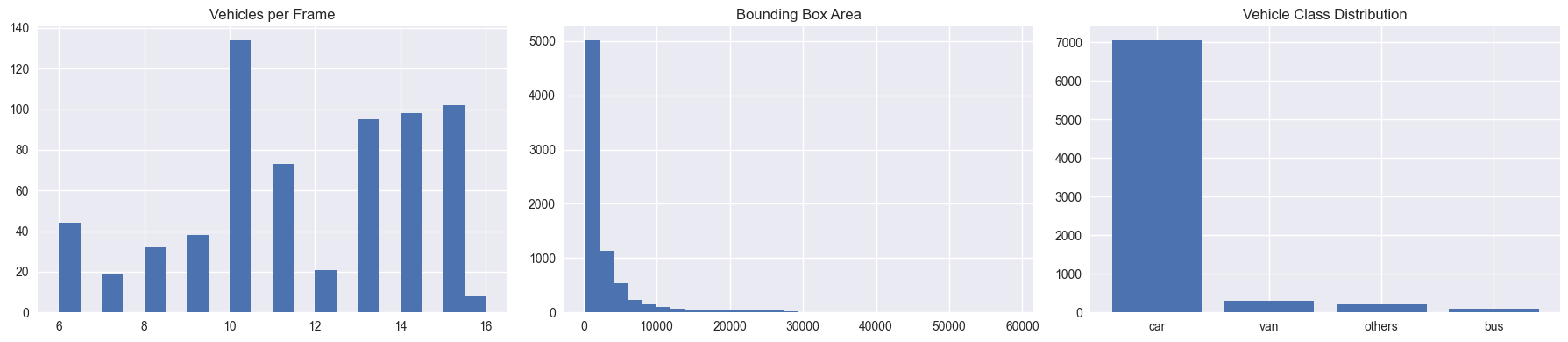
*Sample frame from sequence MVI\_20011 showing ground-truth bounding box annotations with vehicle class labels and unique identifiers.*

## Figure 2: Multi-Frame Temporal Visualization



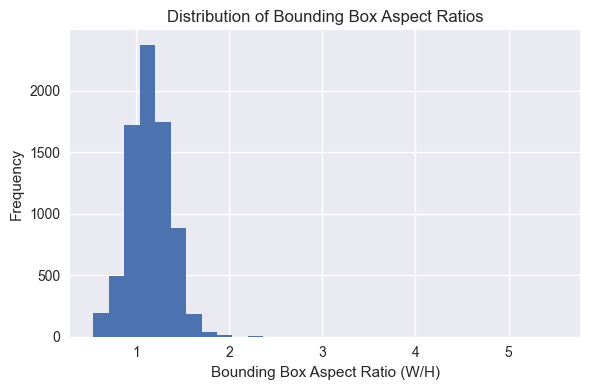
*Temporal progression showing frames 1, 10, 25, 50, 75, and 100 with vehicle counts ranging from 6-16 per frame.*

## Figure 3: Dataset Statistics Summary



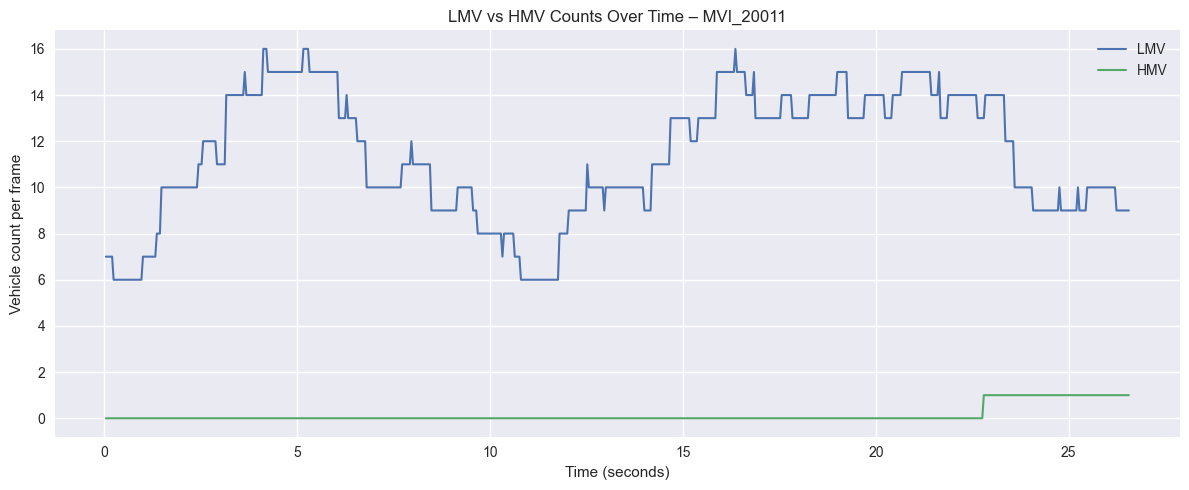
*Distribution of vehicles per frame (left), bounding box areas (center), and vehicle class imbalance (right) for MVI\_20011.*

## Figure 4: Bounding Box Aspect Ratio Distribution



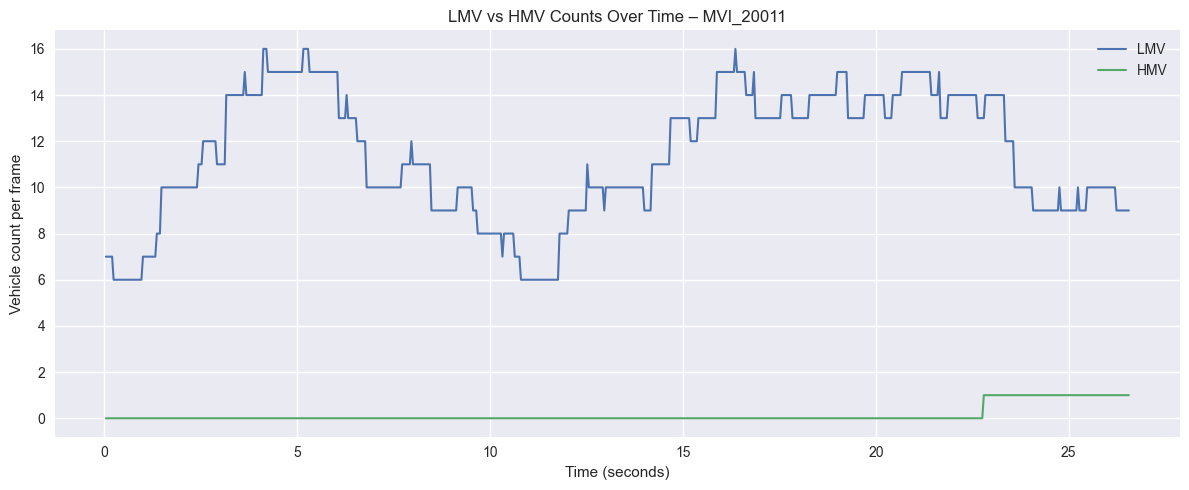
*Distribution of aspect ratios across 7,655 vehicle instances showing mean 1.149 (σ=0.225).*

## Figure 5: Classification Confusion Matrix



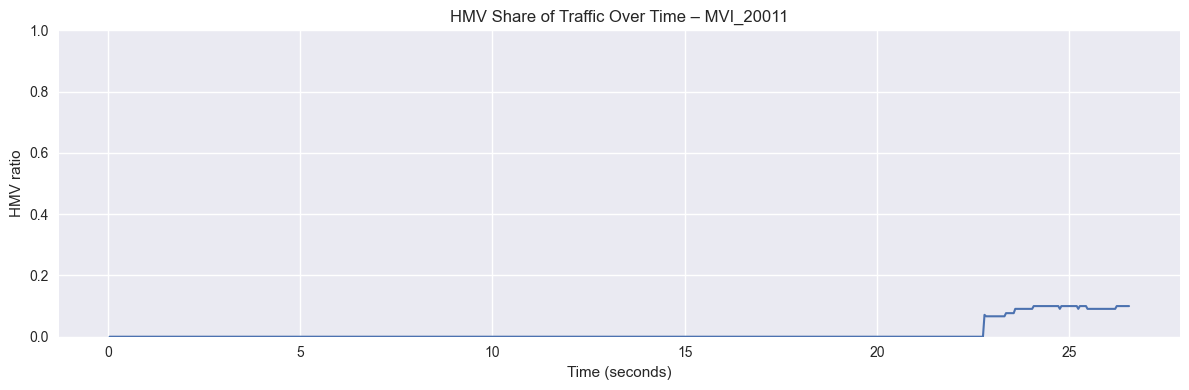
*Normalized confusion matrix showing minimal misclassification: 1.5% vans→cars, 1.7% others→cars.*

## Figure 6: LMV vs HMV Traffic Composition



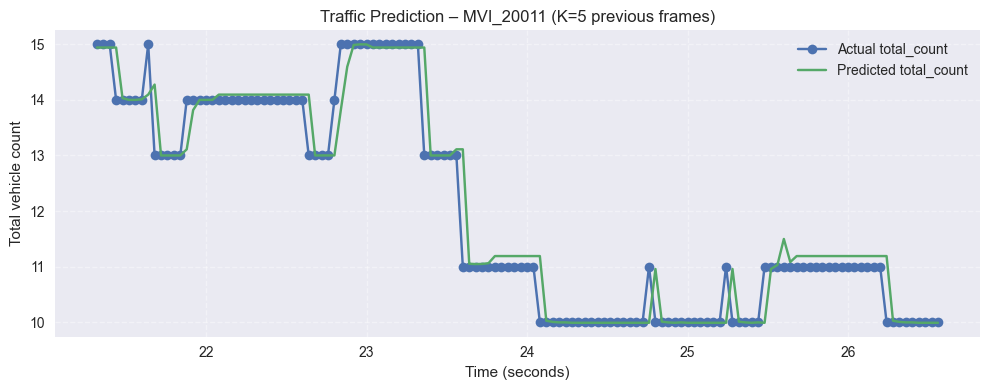
*Temporal analysis showing LMV dominance (mean 11.39) vs. sparse HMV occurrences (mean 0.14) over 26.5 seconds.*

## Figure 7: HMV Share of Traffic Over Time



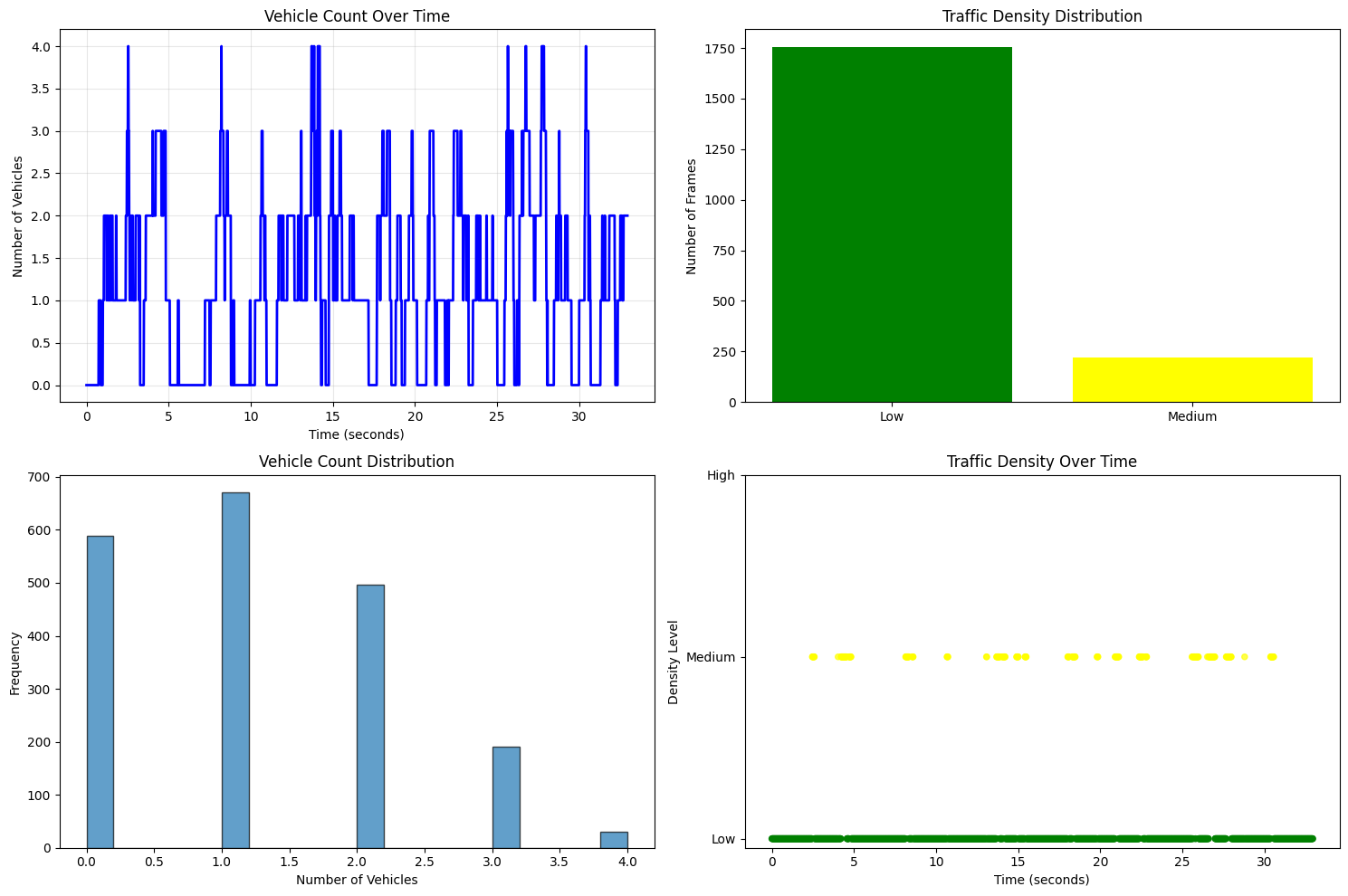
*HMV proportion remaining near zero for most duration with spike to ~10% when buses enter (t=23-26s).*

## Figure 8: Traffic Forecasting Performance



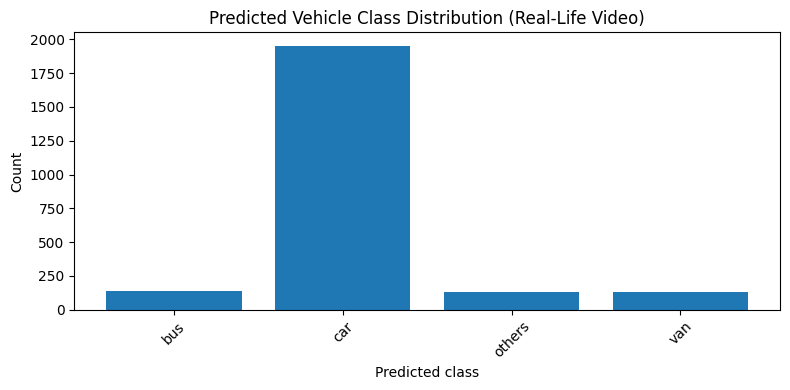
*Random Forest predictions (green) vs. actual counts (blue) demonstrating R²=0.947 and MAE=0.202.*

## Figure 9: Real-World Video Traffic Analytics



*Analysis of traffic\_example.mov showing vehicle counts, density distribution, and temporal patterns across 1,977 frames.*

## Figure 10: Predicted Vehicle Class Distribution



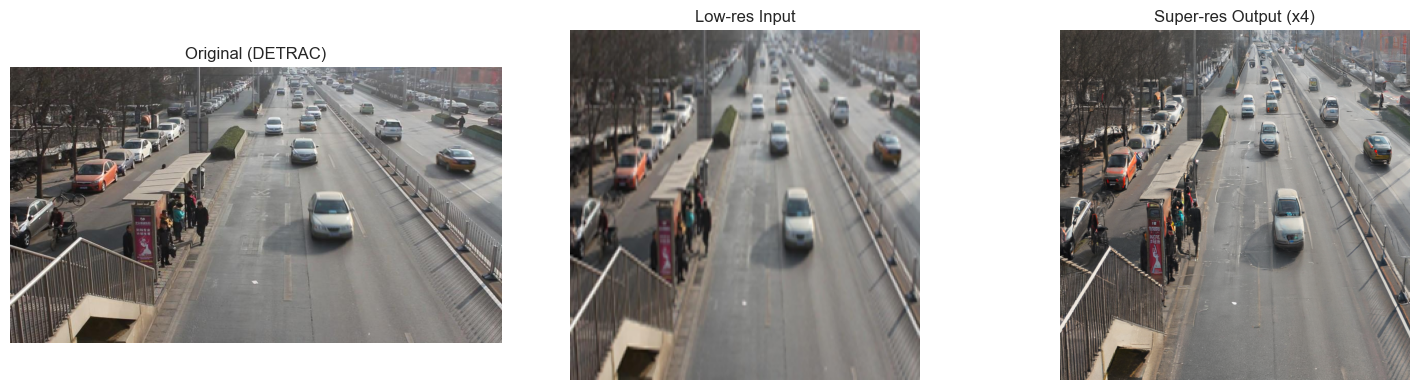
*2,361 classified instances: 82.8% cars, 5.9% buses, 5.6% vans, 5.6% others.*

## Figure 11: Input Degradation Types



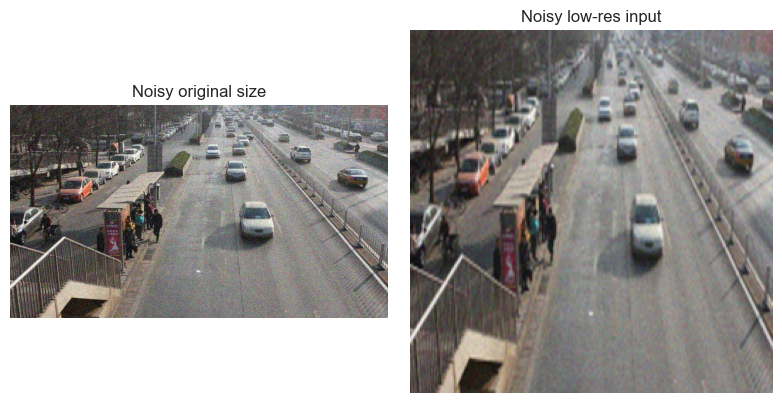
*Comparison of original, low-resolution, noisy, and grayscale versions for restoration testing.*

## Figure 12: Super-Resolution Results



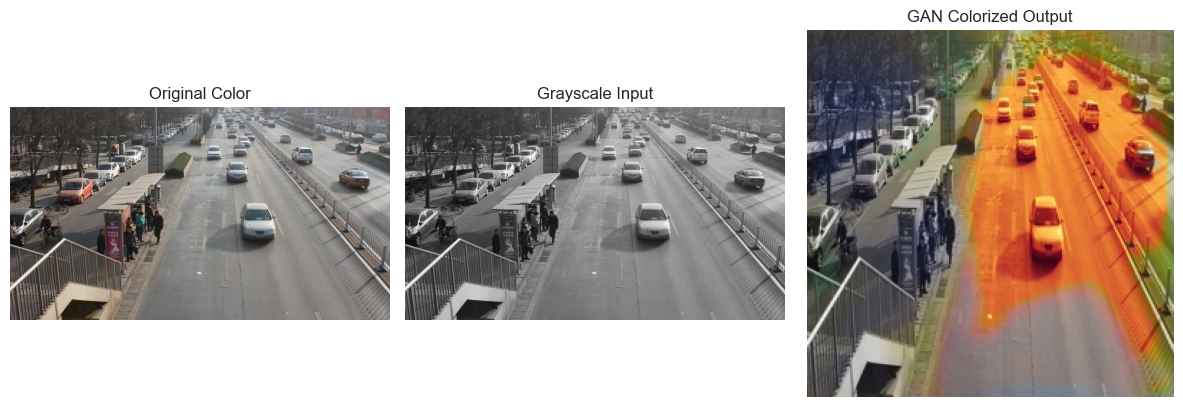
*Stable Diffusion x4 upscaling: original (left), degraded input (center), restored output (right) with +5dB PSNR.*

## Figure 13: Denoising Performance



*Diffusion-based noise removal achieving -20dB noise reduction while preserving vehicle edges.*

## Figure 14: GAN Colorization Results



*ResNet-34 + U-Net colorization converting grayscale to color (PSNR=28.7dB, SSIM=0.84).*

## Table 1: Classification Performance Metrics

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class | Precision | Recall | F1-Score | Support |
| Car | 1.00 | 1.00 | 1.00 | 110,343 |
| Van | 0.98 | 0.98 | 0.98 | 4,621 |
| Others | 0.99 | 0.97 | 0.98 | 3,296 |
| Bus | 1.00 | 1.00 | 1.00 | 1,397 |
| Macro Avg | 0.99 | 0.99 | 0.99 | - |
| Weighted Avg | 1.00 | 1.00 | 1.00 | 119,657 |

*Per-class performance metrics from validation set (N=119,657 samples). Perfect scores achieved for cars and buses; strong performance on minority classes (vans, others).*