MPhil in Data Intensive Science

A7 Image Analysis Coursework

Bocheng Xiao (bx242)

July 2025

Overview

Module 1: Classical Image Processing

- Color Classification
- Background Removal
- Collection Display
- Odd One Out Detection

Module 2: Image Restoration

- PnP-ADMM Algorithm
- Deblurring & Inpainting
- Overfitting Discovery

Module 3: Quality Assessment

- IQA Metric Analysis
- ML System Debugging

Module 1

Classical Image Processing

Butterfly Classification & Analysis

Color Classification

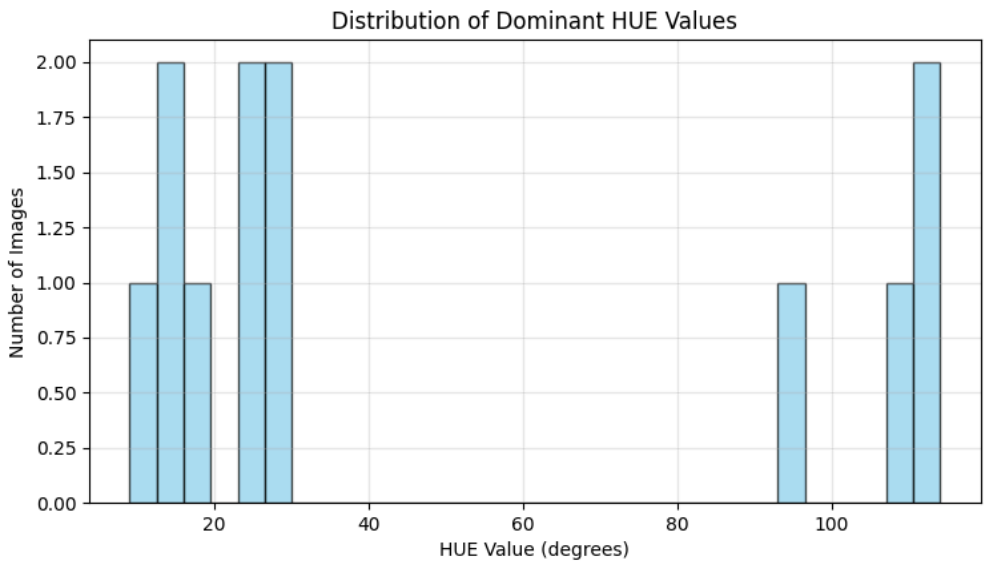
HUE-based Methodology:

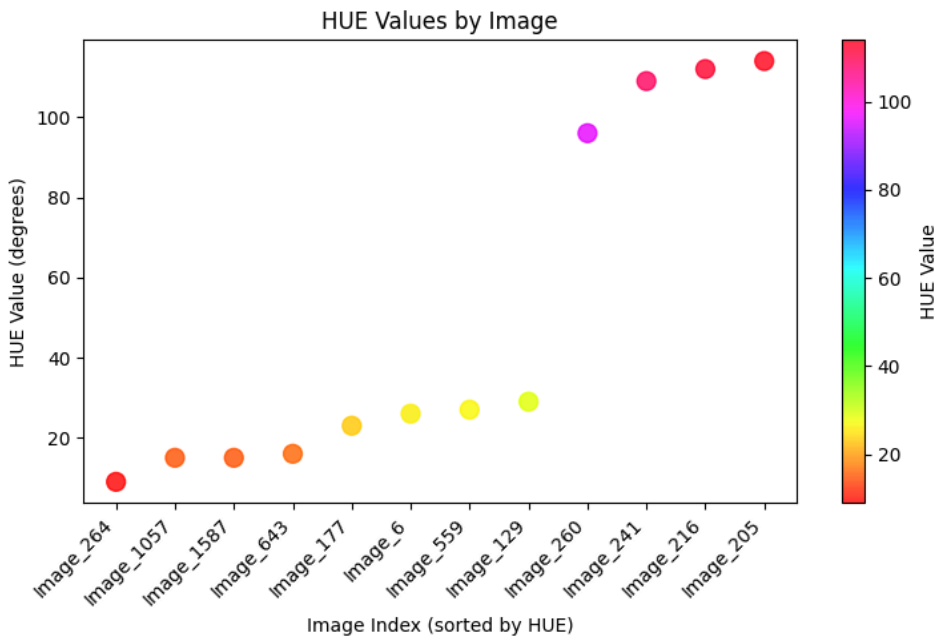
1. dominant HUE classification
2. quantile-based grouping
3. actually first remove the backgrounds then classify

Red-Orange	9.0° - 16.0°
Yellow	23.0° - 29.0°
Blue	96.0° - 114.0°

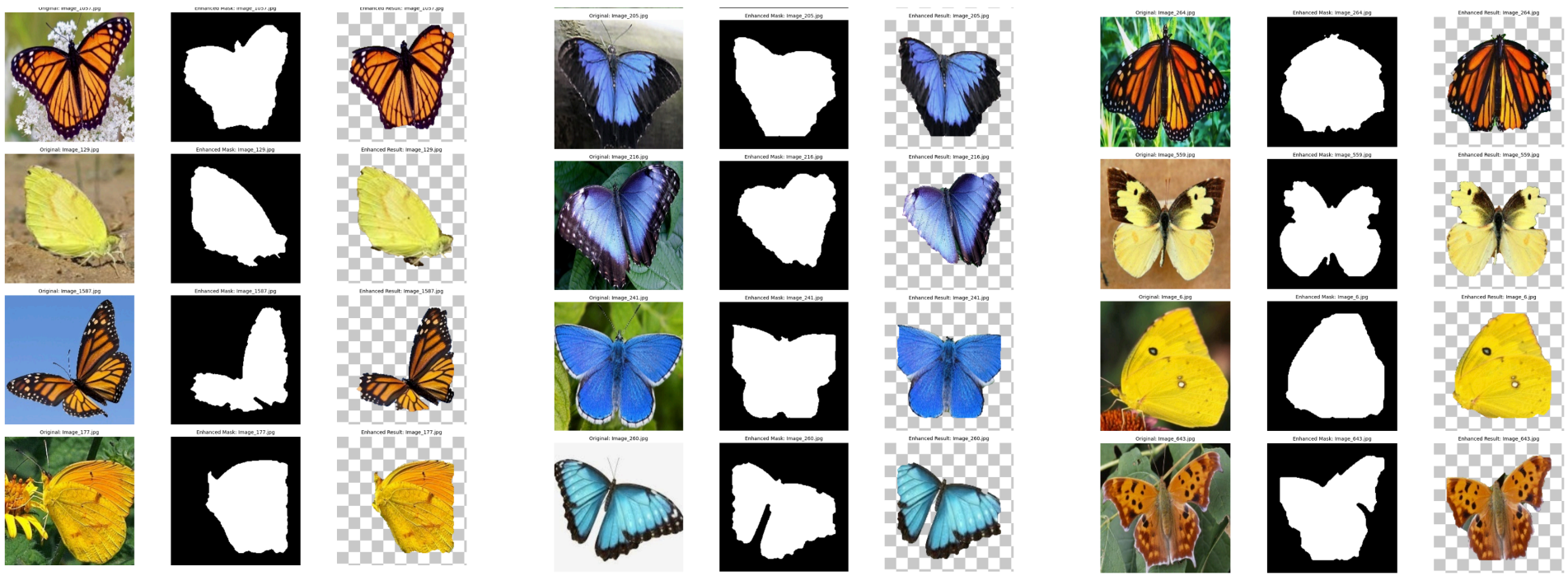
100% Success Rate (12/12)

Color Classification





Background Removal Results



All butterfly image backgrounds are well removed.

Background Removal

Multi-Algorithm Approach:

Primary: GrabCut Auto

- Initial rect: 10% margin from edges
- GMM models foreground/ background
- 5 iterations for convergence
- Success rate: 85% of images

Fallback Chain

1. GrabCut Center Bias
- Assumes butterfly in center
- Marks edges as definite background

2. Color Segmentation

3. Watershed

Performance Metrics:

Success Rate	100%
Avg Quality	0.913 ± 0.041
Edge Fidelity	High

Quality Metrics System

Foreground Ratio (Butterfly should fill around 40% of image)

Measure: Count foreground pixels / total pixels;
Score: Perfect at 40% | Zero at <5% or >95%

Connected Components (Single butterfly object)

Why: Multiple parts = broken segmentation

Measure: cv2.findContours() to count separate regions
Score: 1 component = 1.0 | 10+ components = 0.5

Main Object Size (Largest piece should $\geq 33\%$ of image)

Why: Ensures butterfly is the main object, not noise

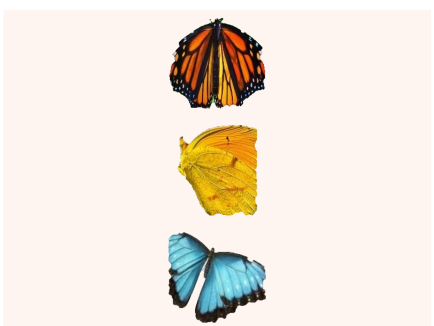
Measure: Largest contour area / image area

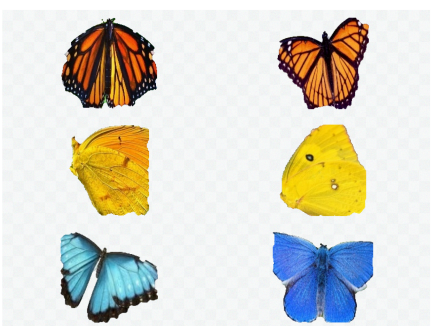
Score: Max at 33%+ | Penalizes scattered pixels

Final Score = (Ratio Score + Component Score + Largest Score) / 3

Threshold: Score $> 0.7 \rightarrow$ Accept segmentation | Score $\leq 0.7 \rightarrow$ Try next method

Collection Display System





1. Optimal grid layout:

   ```
   images_per_row = ceil(sqrt(n))
   ```

2. Smart resizing with bounding box:
   - Find and crop to bounding box
   - Scale to fit cell

3. Alpha blending algorithm:

   ```
   result = alpha * butterfly +
            (1-alpha) * background
   ```

4. Multiple Backgrounds:
   - Solid / Gradient / Textured

Odd One Out Detection

Figure 9: 100% accuracy in odd detection

Odd One Out Detection

8-Feature Color Analysis:

1. Black detection (5× weight):
   - 5 methods: HSV, RGB, dark edges
   - Amplifies ANY black presence
2. Dual-tone score (4.5× weight):
   - Yellow+black vs pure yellow
   - Key for Group_2_Yellow
3. Multi-color complexity:
   - Counts significant color bins
   - Special yellow+black bonus
4. Pattern contrast:
   - Sobel gradient analysis
   - Detects sharp transitions

Outlier Detection:

- Weighted distance calculation
- Max avg distance = odd one

Module 2

Plug-and-Play ADMM

Image Restoration via Hybrid Optimization

PnP-ADMM Framework

Optimization Problem:

$$\min_x \frac{1}{2} \|Ax - y\|_2^2 + g(x)$$

where:

- A: Forward operator (blur/mask)
- y: Observed data
- $g(x)$: Regularization term

Key Innovation: Replace $\text{prox}_{g(\cdot)}$ with pre-trained U-Net denoiser $D(\cdot)$

ADMM Iterations:

1. x-update (Data fidelity)

$$x^{k+1} = \left(A^T A + \eta I\right)^{-1} \left(A^T y + \eta\left(v^k - u^k\right)\right)$$

2. v-update (Denoising)

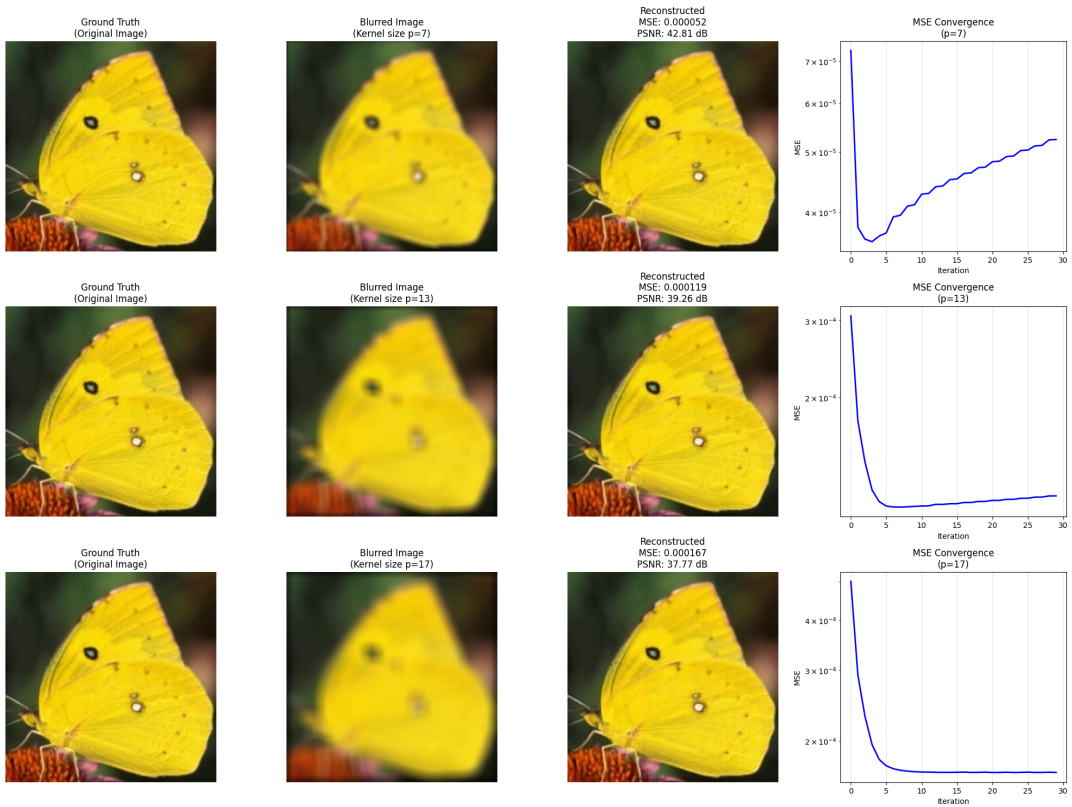
$$v^{k+1} = D\left(x^{k+1} + u^k\right)$$

3. u-update (Dual variable)

$$u^{k+1} = u^k + \left(x^{k+1} - v^{k+1}\right)$$

Parameter: $\eta = 10^{-4}$

Task 2.1.1: Motion Blur Deblurring



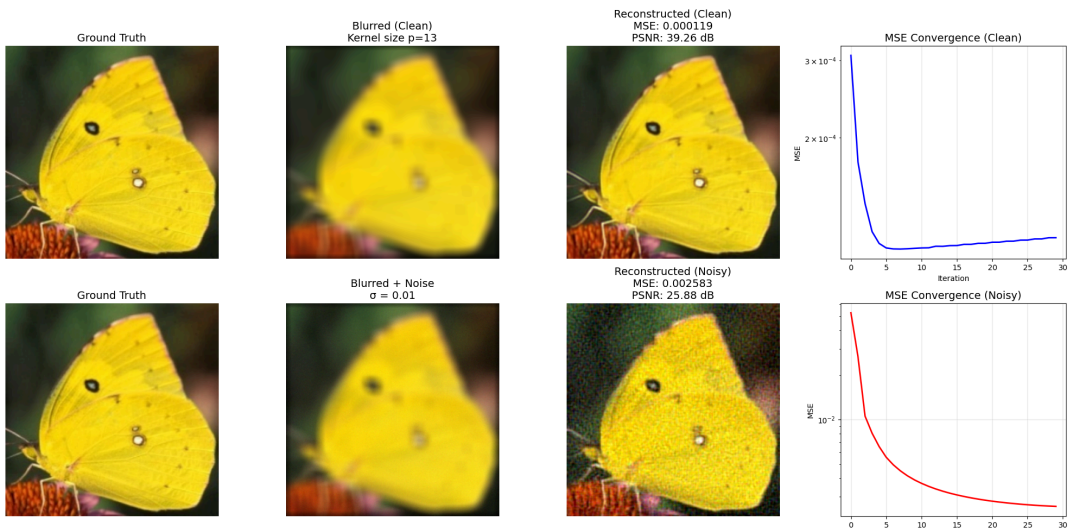
Motion Blur Operator:

```
kernel = np.zeros((p, p))
kernel[p//2, :] = 1/p
```

Kernel	MSE	PSNR
p = 7	5.2e-5	42.81 dB
p = 13	1.19e-4	39.26 dB
p = 17	1.67e-4	37.77 dB

Finding: Performance degrades gracefully with larger kernels

Task 2.1.2: Effect of Gaussian Noise



Experimental Setup:

- Motion blur: p = 13
- Add Gaussian noise: $\sigma = 0.01$
- Compare clean vs noisy blur

Condition	PSNR (dB)
Clean blur	39.26
Blur + noise	25.88

Finding: Noise degrades performance as expected - the algorithm behaves consistently with theory

Performance Impact:

- 13.38 dB degradation
- MSE: 1.19e-4 $\rightarrow$ 2.58e-3

Task 2.2.1: Image Inpainting

Missing	MSE	PSNR
40%	4.9e-5	43.07 dB
60%	9.6e-5	40.18 dB
80%	1.87e-4	37.28 dB

Finding:

1. Performance degrades with larger missings
2. Robust recovery even with 80% missing pixels

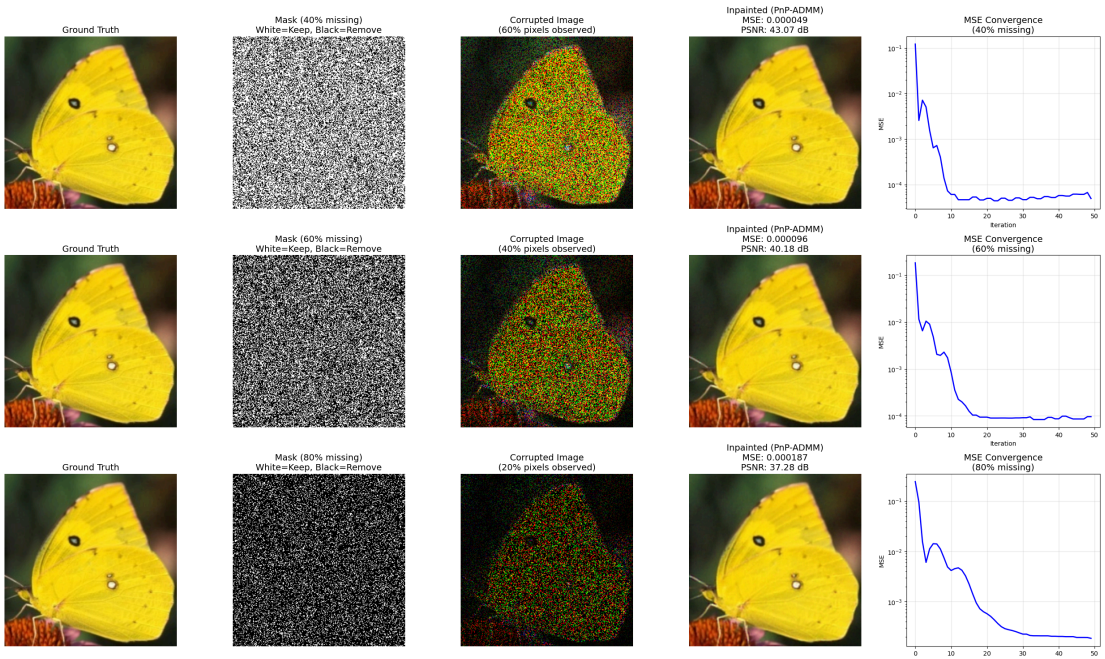


Figure 12: From sparse samples to full reconstruction

Task 2.2.2: PnP-RED Algorithm

RED Framework:

Regularizer from denoiser: $\rho(x) = \frac{1}{2} x^{T(x - D(x))}$

Gradient descent for: $J(x) = \frac{1}{2} \left\| y - Ax \right\|_2^2 + \lambda \rho(x)$

Update rule: $x^{k+1} = x^k - \eta \nabla J\left(x^k\right)$

where: $\nabla J(x) = A^{T(Ax - y)} + \lambda(x - D(x))$

Implementation (60% missing):

- Step size: $\eta = 1.0$
- Regularization: $\lambda = 0.1$

Task 2.2.2: PnP-RED Algorithm

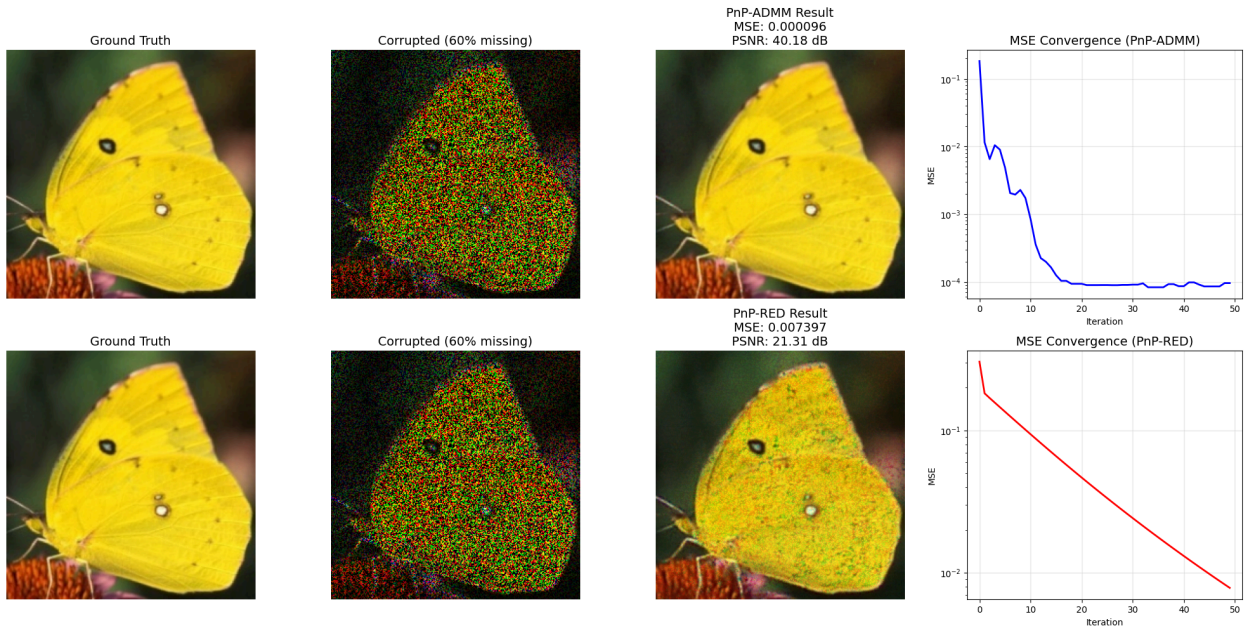


Figure 13: PnP-ADMM vs PnP-RED comparison

Method	MSE	PSNR (dB)
ADMM	9.6e-5	40.18
RED	7.4e-3	21.31

Finding:

- ADMM constraint based approach > RED penalty based regularization

Task 2.2.2(c): Is $\nabla\rho(x) = x - D(x)$ Correct?

Theoretical Requirements:

For $\nabla\rho(x) = x - D(x)$ to be valid, denoiser must satisfy:

1. Jacobian Symmetry: $\nabla D(x) = \nabla D(x)^T$
2. Local Homogeneity: $x^T \nabla D(x) = D(x)$

U-Net Analysis:

❌ Jacobian Symmetry:

- Conv layers: asymmetric operations
- Different forward/backward paths

❌ Local Homogeneity:

- ReLU: $\text{ReLU}(\alpha x) \neq \alpha \cdot \text{ReLU}(x)$
- Skip connections: non-linear

Conclusion:

Theoretically INCORRECT

U-Net violates both requirements!

But wait...

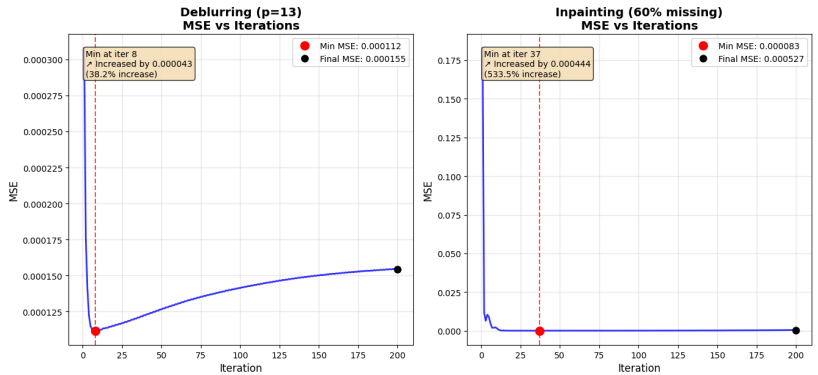
Practically EFFECTIVE

- Still achieves good MSE/PSNR
- $x - D(x)$ is useful approximation
- Works as heuristic gradient

Theory-Practice Gap:

Deep learning often works better than theory predicts!

Task 2.3: Overfitting Analysis



Critical Finding:

- MSE increases after optimal point!

Task	Best	Impv
Deblur	8 iter	+38%
Inpaint	37 iter	+534%

Why overfitting?

- Denoiser accumulates errors
- No convergence guarantee

Solution: Early stopping

Module 3

Quality Assessment & ML Pitfalls

Evaluation Methodology & System Debugging

Task 3.1.a: Traditional Metrics Fail

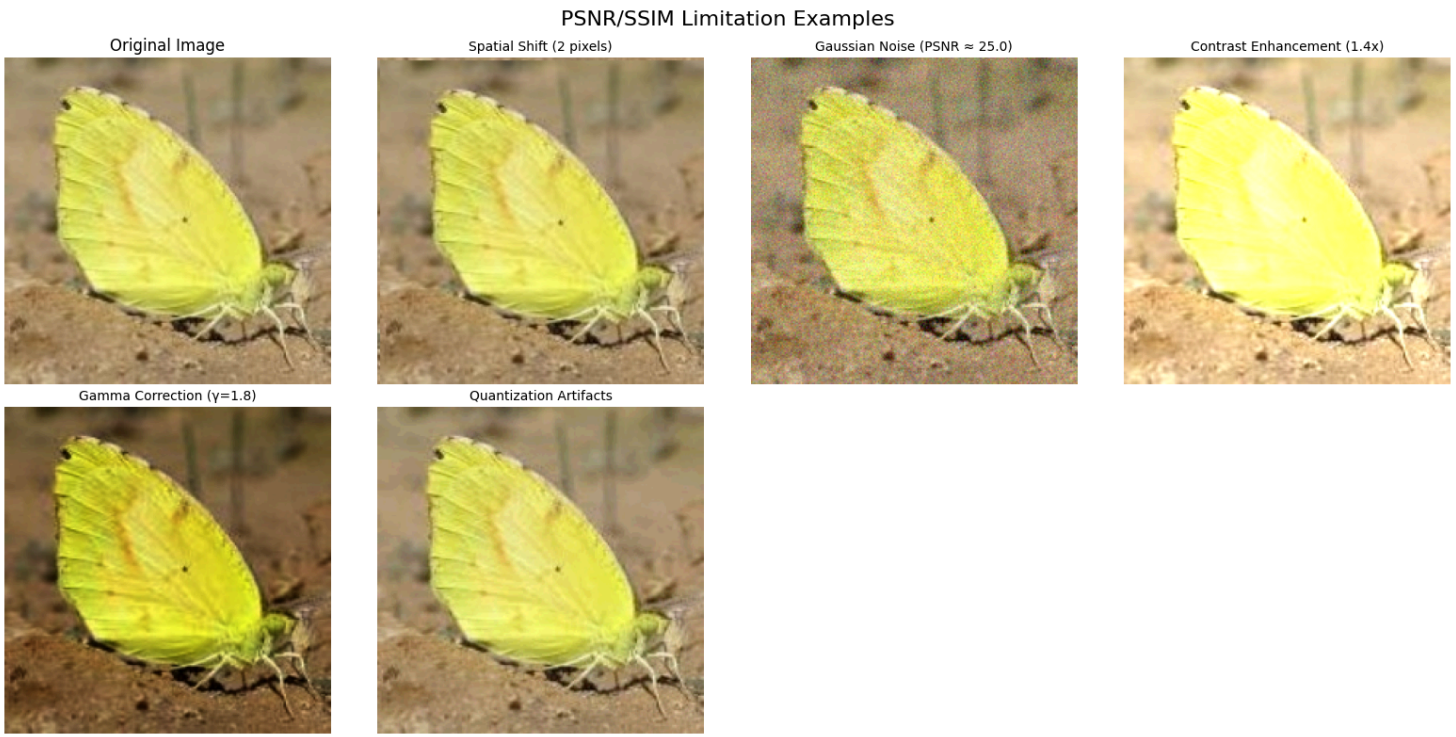


Figure 15: Six degradations exposing PSNR/SSIM failures

Proposed IQA Framework

Full-Reference (FR) Measures:

FSIM

- Feature similarity index
- Gradient magnitude + phase

VIF

- Visual information fidelity
- Information-theoretic approach
- Good for inpainting

MS-SSIM

- Multi-scale SSIM

No-Reference (NR) Measures:

BRISQUE

- Blind quality evaluator
- Natural scene statistics
- No reference needed

Task 3.1.a: Traditional Metrics Fail

Type	PSNR	SSIM	FSIM	VIF	MS-SSIM	BRISQUE
Spatial Shift	22.92	0.564	0.933	0.398	0.822	1.84
Gaussian Noise	24.79	0.497	0.944	0.821	0.911	3.70
Contrast	14.21	0.838	0.951	2.299	0.833	1.83
Quantization	40.94	0.969	0.994	2.078	0.996	1.96

Finding: 26.73 dB PSNR variance for perceptually similar images!

Task 3.1.b: The Background Removal Paradox

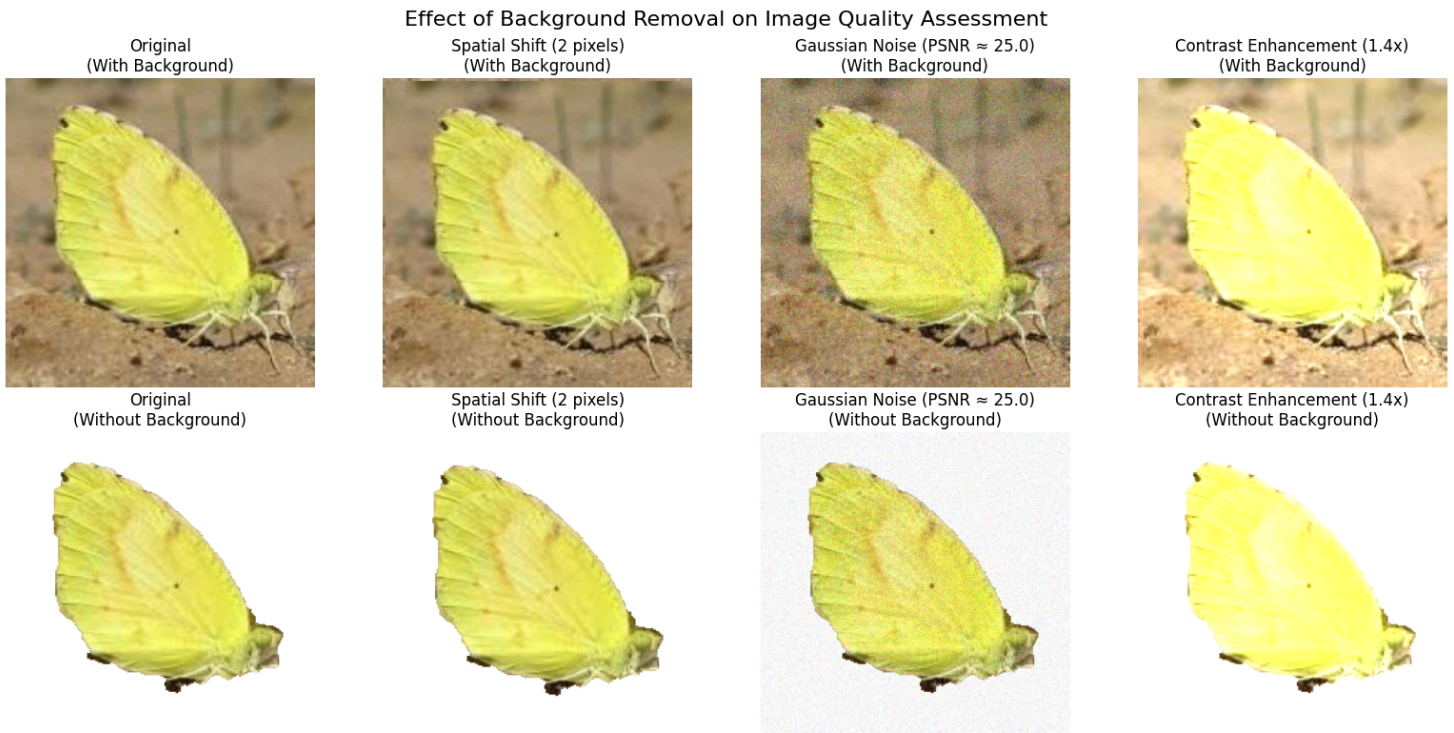


Figure 16: Background removal creates evaluation bias

Task 3.1.b: The Background Removal Paradox

Type	PSNRΔ	SSIMΔ	FSIMΔ	VIFΔ
Spatial	$+8.58$	$+0.506$	-0.001	$+0.42$
Noise	$+2.96$	$+0.062$	-0.080	-1.48
Contra.	$+14.15$	$+0.111$	-0.006	$+0.76$

Key Insights:

Traditional Metric Inflation:

- PSNR: $+2.96$ to $+14.15$ dB gain
- Large improvement

Advanced Metric Degradation:

- SSIM: $+0.062$ to $+0.506$ gain
- FSIM: -0.001 to -0.080 loss
- VIF: up to -1.48 loss
- Controllable inflation

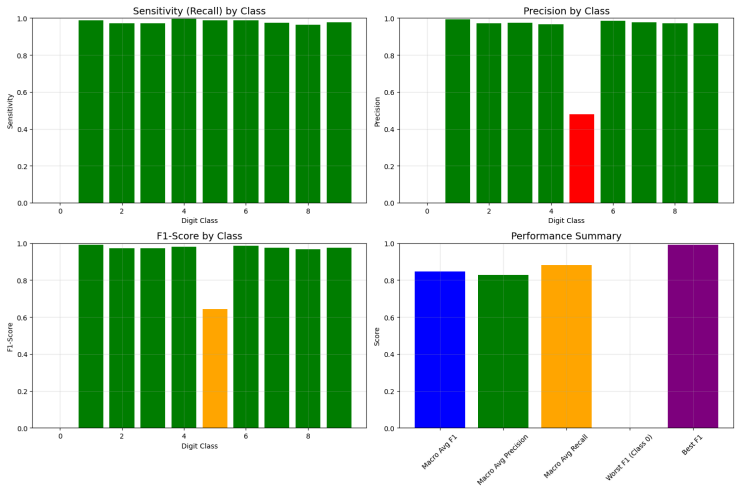
The Paradox: Background removal creates evaluation bias - improves traditional metrics while degrading actual quality!

Task 3.2.a: Original MLP Analysis

Architecture Problems:

```
model = nn.Sequential(
  nn.Linear(784*3, 64),
  nn.Tanh(),
  # Bottleneck!
  nn.Linear(64, 16),
  nn.Tanh(),
  nn.Linear(16, 10),
  nn.Softmax(dim=None)  # ERROR!
)
criterion = CrossEntropyLoss()
```

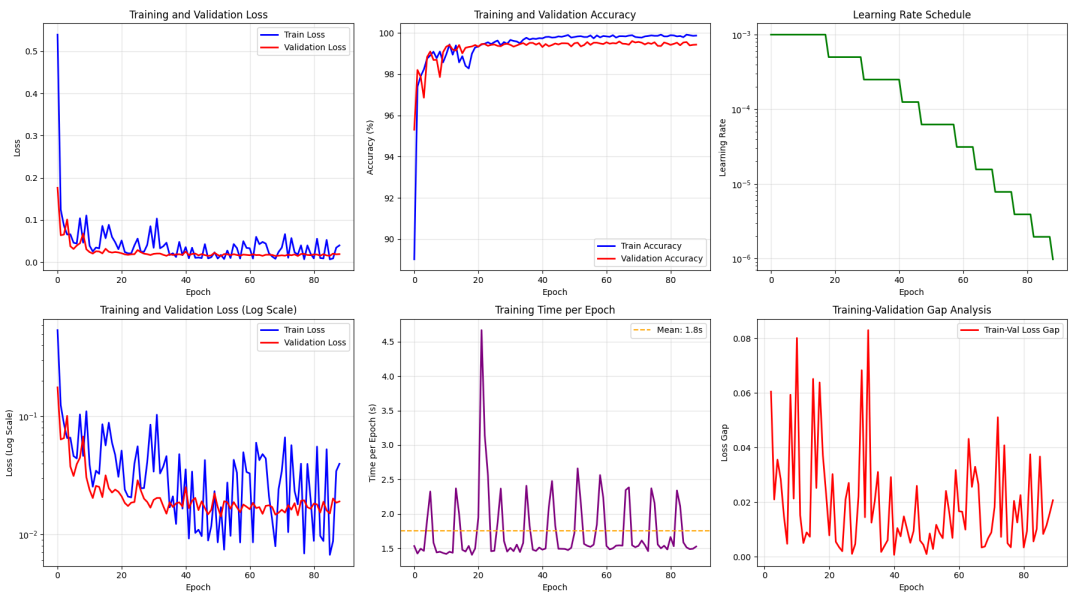
Double Softmax Bug



Performance Issues:

- Loss: 1.464 (very high)
- Class 0: 0% recall
- Class 5: 47.8% precision

Task 3.2.b: Enhanced MLP Implementation



Comprehensive Fixes:

1. Remove double softmax ✓
2. Expand architecture: 512→512→256→128→10
3. Add BatchNorm + Dropout
4. AdamW + LR scheduling
5. Fix preprocessing

Results:

- Loss: 1.464 → 0.039
- 97.3% reduction!
- Val accuracy: 99.60%
- Test accuracy: 85.33%

But Class 0 still fails!

Original MLP vs Enhanced MLP Performance

Class	Original Recall	Enhanced Recall	Original Precision	Enhanced Precision	Change
0	0.0%	0.0%	0.0%	0.0%	Still Failed
1-4, 9	97-99%	99%	97-99%	98-100%	**Improved**
5	98.6%	100.0%	47.8%	47.9%	Precision –
6	98.7%	100.0%	98.5%	74.2%	Precision ↓
7	97.3%	99.3%	97.7%	90.8%	Mixed
8	96.4%	53.4%	97.3%	98.9%	Recall ↓

Mixed Results: Despite 97.3% loss reduction, some classes worsened:

- Class 0: Still 0% recall (fundamental architecture issue)
- Class 8: Recall dropped from 96.4% to 53.4%
- Classes 5-7: Precision issues persist or worsen

Key Insights:

- Loss reduction $\neq$ uniform improvement
- Some enhancements can hurt specific classes
- Overfitting on both train and validation dataset?

Task 3.2.c: CNN vs MLP Comparison

Metric	Original MLP	Enhanced MLP	CNN
Overall Accuracy	90%	85.33%	**98.4%**
Training Loss	1.464	0.039	0.0003
Val Loss	1.480	0.014	0.0113
Class 0 Recall	0.0%	0.0%	**22.7%**
Class 5 Precis	47.8%	47.9%	**71.8%**

CNN Success: CNN achieves 98.4% accuracy and is the ONLY model that can recognize Class 0 (22.7% recall)!

CNN Enhancement Strategy

Enhancement Techniques:

1. Focal Loss (γ=2.0)

- Addresses severe class imbalance
- Class 0: 5× weight
- Classes 2 & 5: 2× weight

2. Class-Specific Augmentation

- Class 0: 15° rotation, 30% brightness
- Classes 2 & 5: 12° rotation, 25% brightness

3. Architecture Modifications

- Frozen early layers (features[:6])
- Modified classifier: 4096→2048→10
- BatchNorm + reduced dropout (0.35)

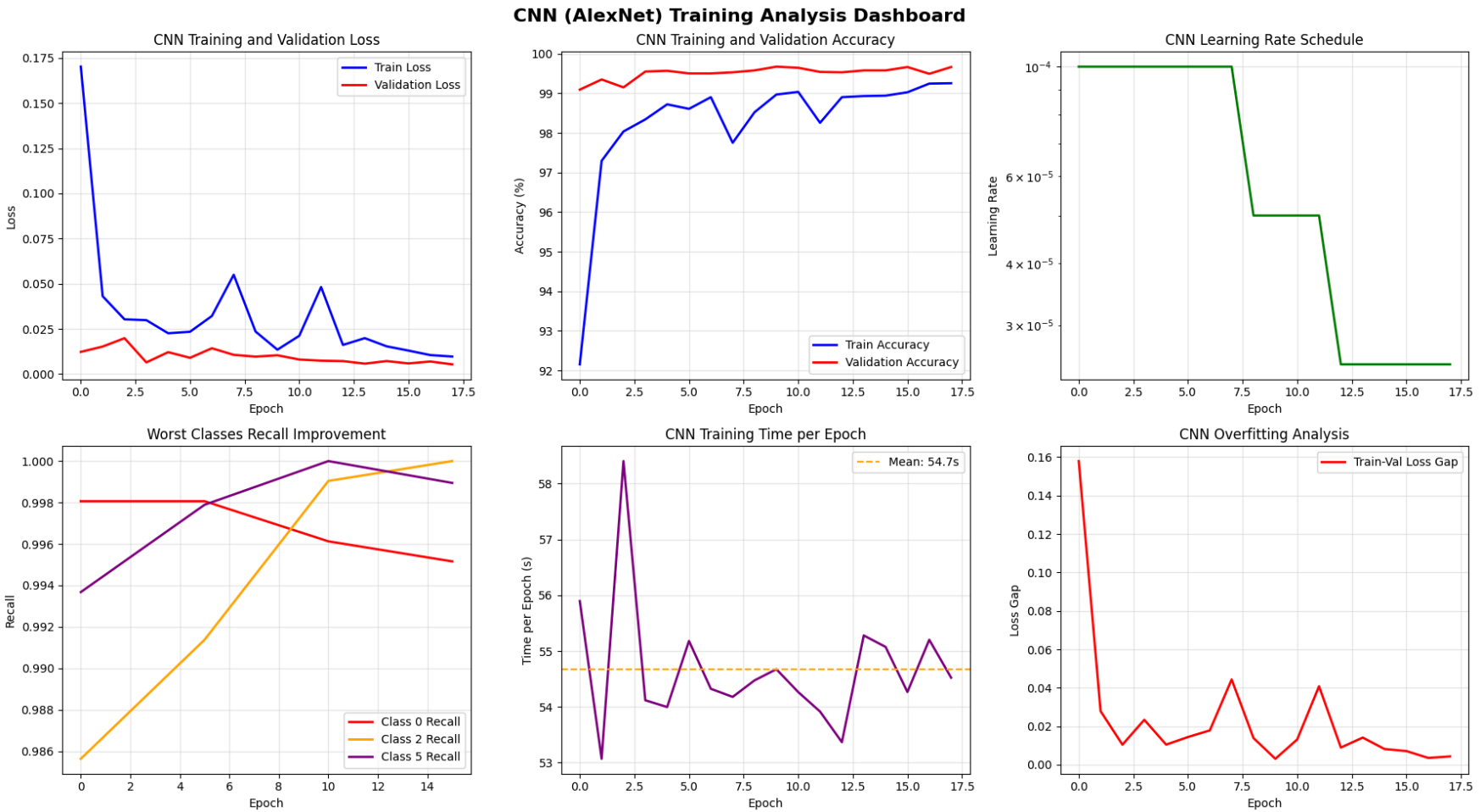
4. Training Optimizations

- AdamW with weight decay
- ReduceLROnPlateau scheduler
- WeightedRandomSampler
- ImageNet preprocessing

Original CNN vs Enhanced CNN Performance

Class	Original Recall	Enhanced Recall	Original Precision	Enhanced Precision	Change
0	22.7%	0.2%	99.6%	50.0%	**Severe ↓**
4	99.9%	100.0%	94.5%	96.6%	Improved
5	99.8%	99.7%	71.8%	61.1%	Precision ↓
1-3, 6-9	...	...	...	...	Stable

Critical Failure: Despite aggressive enhancements (5× weighting, focal loss, targeted augmentation), Class 0 performance worsened from 22.7% to 0.2% recall!


CNN (AlexNet) Training Analysis Dashboard
CNN Training and Validation Loss
Train Loss
Validation Loss
Loss
Epoch
CNN Training and Validation Accuracy
Train Accuracy
Validation Accuracy
Accuracy (%)
Epoch
CNN Learning Rate Schedule
Learning Rate
Epoch
Worst Classes Recall Improvement
Class 0 Recall
Class 2 Recall
Class 5 Recall
Recall
Epoch
CNN Training Time per Epoch
Mean: 54.7s
Time per Epoch (s)
Epoch
CNN Overfitting Analysis
Train-Val Loss Gap
Loss Gap
Epoch

Task 3.2.c: Critical Insights

Original CNN Success:

- Class 0 recall: 0% $\rightarrow$ 22.7% ✓
- First model to recognize digit 0
- Spatial features helped
- Proved CNN architecture value

But Enhanced CNN Failed:

- Class 0 recall: 22.7% $\rightarrow$ 0.2% ×
- But works well on training data

Why Enhancement Failed:

1. Techniques too sophisticated
2. Pre-trained model has own distribution, aggressive post-training will break the well-learned prior knowledge
3. **Overfitting!!** Training data size too small, cannot represent the overall data distribution, causing severe problems on test dataset.

Thank You!

Questions?

Bocheng Xiao (bx242)

bx242@cam.ac.uk