AI vs. Human-Generated Images Detection: A Deep Learning Approach

Course: INFO-6147 DL with PyTorch
Student: Harrison Kim, 1340629
Date: August 15, 2025

Abstract

This project implements binary classification to distinguish between AI-generated and human-created images using CNNs. The initial plan involved using the CityScapes dataset with SAM (Segment Anything Model), but implementation challenges with 1024×1024 inputs made this approach impractical. The strategic pivot to classification proved successful. Training accuracy reached **92.46%**, and validation plots indicated accuracy above **90%**. The final test performance achieved **87.59%**.

The small image resolution limited available detail, and despite CUDA being enabled, training did not effectively utilize the GPU. This resulted in longer processing times than anticipated.

1. Introduction and Project Evolution

1.1 Original Vision and Pivot

The project began with a goal of semantic segmentation using CityScapes and SAM. Several challenges emerged during implementation:

Input Size: The 1024×1024 requirement significantly increased training time.
Integration Issues: CityScapes format and SAM integration proved more complex than anticipated.
Implementation Complexity: Additional attempts to incorporate YOLO for object detection led to excessive complexity, necessitating multiple project restarts.

Given these constraints, a strategic pivot was made. The new focus shifted to AI vs. Human image detection. This problem offers computational simplicity while maintaining relevance in current research contexts.

1.2 Ethical Motivation

AI tools now generate realistic images. This raises questions of misuse, transparency, and authenticity. The project aims to contribute a small step toward detection systems that help preserve trust in digital media.

2. Dataset and Methodology

2.1 Dataset Characteristics

Total Images: 60,000 (reduced from original 120,000 for training efficiency)
Classes: 2 (FAKE: AI-generated, REAL: human-created)
Split: 40k train / 10k validation / 10k test

The dataset maintained balanced class distribution. The size reduction addressed practical training constraints while preserving sufficient data for meaningful learning.

2.2 Data Preprocessing

Normalization: Standard range conversion applied
Augmentation: Applied exclusively to training dataset (horizontal flips and +/-10` rotations)
Resolution: Uniform resizing to 32×32 pixels

This approach prioritized consistency over preserving original image detail. While not optimal for detecting subtle AI artifacts, it maintained computational feasibility.

3. Model Architecture

3.1 Barebone CNN

Here's the basic structure I used:

```python
import torch.nn as nn

class Net(nn.Module):
    def __init__(self, num_classes):
        super(Net, self).__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 16, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(16, 32, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(32, 64, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2)
        )
        self.classifier = nn.Sequential(
            nn.Flatten(),
            nn.Linear(64 * 4 * 4, 128),
            nn.ReLU(),
            nn.Dropout(0.5),
            nn.Linear(128, num_classes)
        )

    def forward(self, x):
        x = self.features(x)
        x = self.classifier(x)
        return x
```

4. Training Process and Results

4.1 Setup

Batch Size: 64
LR: 0.0001 (Adam optimizer)
Max Epochs: 25
Early Stopping: Patience of 5
Device: CUDA GPU

4.2 Training Dynamics

Here's how training progressed:

Epoch	Train Acc	Train Loss
1	73.4%	0.5261
5	86.9%	0.3133
10	89.7%	0.2529
25	92.5%	0.1943

The model demonstrated consistent improvement over 25 epochs. Early stopping was not triggered in this training run, unlike some earlier experimental attempts.

4.3 Observations

Training Results: The model demonstrated healthy learning patterns throughout training. Both training and validation accuracy increased together toward ~92%, with validation accuracy occasionally exceeding training accuracy. This indicated successful generalization rather than overfitting.

5. Hyperparameter Experiments

5.1 Grid Search Results

I tested different combinations:

LRs tested: 0.0005, 0.001, 0.005
Batch sizes: 128, 192, 256
Duration: 5 epochs per run

Results varied but most configurations performed well:

LR	Batch	Best Val Acc
0.0005	128	91.13%
0.0005	192	89.83%
0.0005	256	90.41%
0.001	**128**	**92.61%**
0.001	192	92.08%
0.001	256	92.19%
0.005	128	91.21%
0.005	192	92.12%
0.005	256	92.09%

Best configuration: LR=0.001, Batch=128 achieved **92.61%** validation accuracy.

6. Evaluation and Visualization

6.1 Test Set Results

Test Accuracy: 87.59%
Precision: 89.19%
Recall: 87.59%
F1-Score: 87.46%

The validation-test performance alignment (**87.58%** vs **87.59%**) demonstrates good model generalization. These results indicate successful learning despite the architectural simplicity.

7. Challenges and Limitations

7.1 GPU Issues

Despite CUDA being enabled, training performance indicated suboptimal GPU utilization. Monitoring the resources tab revealed that GPU RAM usage remained extremely low throughout training, suggesting the computations were actually running on CPU. This resulted in significantly longer training times than anticipated for the dataset size and model complexity.

7.2 Resolution Constraints

32×32 is quite low for detecting subtle AI artifacts. Modern AI generators create highly realistic content that probably requires much higher resolution analysis. But computational constraints forced this compromise.

8. Future Improvements

8.1 Technical Upgrades

Higher Resolution: Train on 224×224 or larger images
Transfer Learning: Use pre-trained models (ResNet, EfficientNet)
Ensemble Methods: Combine multiple model predictions
K-fold Cross-validation: More robust performance estimates
Learning Rate Scheduling: Adaptive learning rates
Better Regularization: Batch norm, weight decay
Adversarial Training: Robustness against detection evasion (experiment purpose)

9. Ethics and Broader Impact

9.1 Applications

Misinformation: Detect fake media in news and social media
Authenticity: Help verify digital content
Transparency: Let users know when content is AI-generated
Trust: Maintain confidence in media authenticity

9.2 Risks and Considerations

False Positives: Risk of flagging legitimate human content
Bias Issues: Model might show bias toward certain artistic styles
Arms Race: As detection improves, generation methods evolve to evade detection

10. Conclusion

Despite initial challenges with CityScapes/SAM implementation, the project achieved meaningful results. The **87.59%** test accuracy demonstrates that basic CNN architectures can effectively address AI detection tasks.

Key Findings:

- Task complexity assessment proves crucial for project planning
- Resource constraints (GPU utilization) significantly impact workflow efficiency
- Ethical considerations surrounding AI-generated content detection continue gaining importance

The evaluation pipeline discrepancy provided valuable learning insights. This experience emphasized that evaluation system verification must accompany model development throughout the research process.