FractalCNN AI Image Detection - Progress Report v1.0

Date: August 2025
Project: Generalizable AI-Generated Image Detection Based on Fractal Self-Similarity
Implementation: PyTorch FractalCNN Architecture

Summary

This report details the initial training results of our FractalCNN implementation for AI-generated image detection. The model achieved a **62.5% validation accuracy** with early stopping after 9 epochs, demonstrating clear learning progress despite dataset limitations.

Experimental Setup

Dataset Configuration

- **Source:** Kaggle Deepfake and Real Images Dataset
- **Total Images Used:** 800 (400 real + 400 fake)
- **Training Split:** 640 images (320 real + 320 fake) - 80%
- **Validation Split:** 160 images (80 real + 80 fake) - 20%
- **Preprocessing:** RGB → Grayscale, Normalized to [0,1], FFT magnitude spectrum

Model Architecture

- **Fractal Levels:** 3 recursive levels
- **Hidden Channels:** 32 feature maps
- **Parameters:** 77,730 trainable parameters
- **Device:** CUDA-enabled GPU
- **Noise Residual Extraction:** Modified from original median blur to average pooling

Training Configuration

- **Batch Size:** 8 images per batch
- **Learning Rate:** 0.001
- **Optimizer:** Adam with weight decay (1e-4)
- **Loss Function:** CrossEntropyLoss
- **Early Stopping:** Patience = 5 epochs

Training Results

Performance Metrics

Epoch	Train Loss	Val Loss	Val Accuracy	Status
1	0.6939	0.7055	50.0%	Baseline (Random)
2	0.6965	0.6934	50.0%	No improvement
3	0.6955	0.6904	56.25%	Learning begins
4	0.6914	0.6887	**62.5%**	**Best model**
5-9	~0.68	~0.67	52-61%	Plateau/decline

Key Achievements

- **Early Stopping Triggered:** Model correctly stopped at epoch 9 to prevent overfitting
- **Clear Learning Pattern:** Validation accuracy improved from 50% → 62.5% (**+12.5%**)
- **Architecture Validation:** Model processes spectral features correctly
- **Stable Training:** No crashes or convergence issues

Analysis & Insights

What We Learned

Positive Indicators

1. **Model is Learning:** Clear improvement over random baseline (50%)
2. **Architecture Works:** Fractal units process spectral features effectively
3. **Early Stopping Effective:** Prevented overfitting automatically
4. **Spectral Analysis Valid:** FFT-based preprocessing functions correctly

Performance Context

- **Current Result:** 62.5% accuracy
- **Random Baseline:** 50% accuracy
- **Paper Baseline:** 91%+ accuracy (with full dataset)
- **Improvement: 25% better than random** (significant for limited data)

Identified Limitations

1. Dataset Size Constraint

- **Current:** 400 image pairs (800 total)
- **Paper Standard:** 360,000+ training images
- **Impact: 900x smaller dataset** severely limits learning capacity
- **Evidence:** Model plateaued quickly, suggesting insufficient training examples

2. Learning Rate Configuration

- **Current:** 0.001 (Adam optimizer)
- **Issue:** May be too aggressive for limited dataset
- **Effect:** Possible overshooting of optimal weights

3. Implementation Deviations

- **Noise Residual Method:** Used `F.avg_pool2d` instead of median blur (as specified in paper)
- **Reason:** PyTorch compatibility and implementation simplicity
- **Impact:** May affect artifact detection sensitivity
- **Original Paper:** `I_res = I - MedianBlur(I, kernel_size=7)`
- **Our Implementation:** `I_res = I - F.avg_pool2d(I, kernel_size=7)`

4. Architecture Scale

- **Current:** 32 hidden channels, 3 fractal levels
- **Consideration:** May be over-parameterized for small dataset
- **Risk:** Overfitting with limited training data

Next Steps

Priority 1: Dataset Expansion

- [ ] Increase to **1000+ image pairs** (2000 total)
- [ ] Implement data augmentation (rotation, noise, compression)
- [ ] Add train/val/test split (70/15/15)

Priority 2: Hyperparameter Tuning

- [ ] Reduce learning rate: `0.001 → 0.0001`
- [ ] Increase patience: `5 → 10 epochs`
- [ ] Experiment with batch size: `8 → 16`

Priority 3: Implementation Refinements

- [ ] **Replace average pooling with median blur** for noise residual extraction
- [ ] Implement proper median filtering: `cv2.medianBlur()` or custom kernel
- [ ] Compare performance impact of different blur methods
- [ ] Add dropout layers for regularization
- [ ] Implement learning rate scheduling

Priority 4: Architecture Optimization

Technical Implementation Notes

Noise Residual Extraction Modification

During implementation, we encountered a technical constraint with the original paper's median blur approach:

Original Paper Method:

```
I_res = I - MedianBlur(I, kernel_size=7)
```

Our Implementation:

```
I_res = I - F.avg_pool2d(I, kernel_size=7, stride=1, padding=3)
```

Rationale for Change:

- MedianBlur requires CPU-GPU memory transfers with OpenCV
- Average pooling maintains full GPU computation pipeline
- Simplified implementation for initial proof-of-concept

Potential Impact:

- Median blur better preserves edges while removing noise
- Average pooling may blur important high-frequency artifacts
- Could contribute to performance gap vs. paper results

Technical Context

Comparison with Recent Research

Method	Year	Key Innovation	Performance
Our FractalCNN	2025	Fractal spectral analysis	62.5% (limited data)
SPAI	2025	Self-supervised spectral learning	SOTA +5.5% AUC
Original Paper	2024	Multi-level fractal similarity	91.17% average

Field Context

- Spectral analysis remains cutting-edge approach
- Recent SPAI paper validates spectral domain importance
- Our fractal approach offers unique perspective on spectral artifacts

Conclusions

Key Takeaways

1. **Proof of Concept Successful:** FractalCNN architecture functions correctly
2. **Learning Demonstrated:** Model improves beyond random baseline
3. **Dataset is Limiting Factor:** Primary bottleneck identified
4. **Foundation Established:** Ready for systematic improvements

Success Metrics for v1.1

- **Target Accuracy:** 75%+ validation accuracy
- **Dataset Size:** 1000+ training pairs
- **Training Stability:** 15+ epochs without early stopping

Long-term Vision

With proper dataset scaling and hyperparameter optimization, this implementation has potential to achieve performance closer to the original paper's 91%+ accuracy, contributing to the evolving field of AI-generated image detection.