🔬 Hyperspectral Image Classification - Master README

Project: Deep Learning Approaches for Hyperspectral Image Classification

Author: Jaykumar Darji

Course: EE258 - Neural Networks

Institution: Department of Electrical Engineering, San Jose State University

---

📋 Project Overview

This repository contains a comprehensive implementation of various deep learning architectures for hyperspectral image classification. The project focuses on the **Indian Pines** dataset, a benchmark hyperspectral image dataset collected by NASA's AVIRIS sensor, and evaluates multiple neural network architectures including CNNs, RNNs, LSTMs, and MLPs.

Dataset Information

- **Dataset:** Indian Pines Hyperspectral Image
- **Spatial Dimensions:** 145 × 145 pixels
- **Spectral Bands:** 200 (after removing water absorption bands)
- **Classes:** 16 land-cover types (agricultural crops, forests, urban areas)

📁 Project Structure

```
Hyperspectral_Image_Classification/
│
├── 01_Clean_Preprocess.ipynb          # Data preprocessing pipeline
├── 02_model_3DCNN.ipynb               # 3D Convolutional Neural Netwo
├── 02_model_CNN1D.ipynb               # 1D CNN for spectral classific
├── 02_model_CNN2D.ipynb               # 2D CNN for patch-based classi
├── 02_model_LSTM.ipynb                # LSTM with attention mechanism
├── 02_model_MLP.ipynb                 # Multi-Layer Perceptron varian
├── 02_model_MobileNet.ipynb           # MobileNet-style lightweight C
├── 02_model_ResNet.ipynb              # Residual Network architecture
├── 02_model_RNN.ipynb                 # GRU-based RNN with attention
├── 03_Adapter_Warmup.ipynb            # Adapter warmup transfer learn
├── 03_FineTuning.ipynb                # Fine-tuning transfer learning
├── indian_pines_eda.ipynb             # Exploratory Data Analysis
│
├── data/                              # Dataset files
│   ├── Indian_pines_corrected.mat
│   ├── Indian_pines_gt.mat
│   ├── Pavia.mat
│   ├── Pavia_gt.mat
│   ├── PaviaU.mat
│   ├── PaviaU_gt.mat
│   ├── Salinas_corrected.mat
│   └── Salinas_gt.mat
│
├── outputs/                           # Generated outputs
│   ├── artifacts_ip/                  # Preprocessed data artifacts
│   ├── figs/                          # Visualization figures
│   ├── figs_external/                 # External dataset results
│   └── runs_*/                        # Model checkpoints and logs
│
└── README.md                          # This file
```

🔬 Notebook Descriptions

1. Exploratory Data Analysis (`00_indian_pines_eda.ipynb`)

Purpose: Comprehensive exploratory data analysis and visualization of the Indian Pines dataset.
Key Analyses:

- Dataset overview and characteristics
- Band-wise statistics (mean, std, min, max)
- Band correlation matrix
- Spectral signatures for different land-cover classes
- PCA transformation and explained variance analysis
- Outlier detection using Isolation Forest
- Class distribution analysis

2. Data Preprocessing (`01_Clean_Preprocess.ipynb`)

Purpose: Complete data preprocessing pipeline for hyperspectral image classification.
Key Operations:

- Loads hyperspectral cube and ground truth labels from MATLAB files
- Removes noisy water absorption bands (optional)
- Creates stratified train/validation/test splits (75%/5%/20%) to maintain class balance
- Applies z-score normalization using only training statistics (prevents data leakage)
- Optional PCA for dimensionality reduction
- Saves all processed artifacts for downstream use

Output Artifacts:

- `cube_clean_norm.npy` : Normalized hyperspectral cube
- `labels.npy` : Ground truth label map
- `mask_train.npy` , `mask_val.npy` , `mask_test.npy` : Boolean masks for data splits
- `scaler.pkl` : Normalization parameters (mean and std per band)
- `pca.pkl` (optional): Fitted PCA object
- `config_clean_preprocess.json` : Configuration settings

3. 3D Convolutional Neural Network (`02_model_3DCNN.ipynb`)

Purpose: Implements a 3D CNN that simultaneously extracts spatial and spectral features.
Architecture:

- Input: 3D patches (1, 200, 15, 15) - treats spectral dimension as depth
- 3D convolutions operating across spatial and spectral dimensions
- Multiple 3D convolutional blocks with batch normalization
- Spatial and depth pooling layers
- Fully connected classification head

Performance: **99.61% Accuracy**

4. 1D Convolutional Neural Network (`02_model_CNN1D.ipynb`)

Purpose: Processes individual pixel spectra as 1D signals using spectral convolutions.
Key Features:

- Pixel-level classification (no spatial context)
- Hyperparameter search (random search and grid search)
- Cross-validation support
- Band importance analysis using gradient-based methods

Performance: **93.51% Accuracy**

5. 2D Convolutional Neural Network (`02_model_CNN2D.ipynb`)

Purpose: Processes spatial patches treating spectral bands as channels.
Key Features:

- Spatial-spectral feature learning
- No dropout layers (clean architecture)
- Weighted random sampling for class imbalance
- Generates prediction maps for visualization

Performance: **99.76% Accuracy** - Best performing model!

6. Long Short-Term Memory Network (`02_model_LSTM.ipynb`)

Purpose: Treats pixel spectra as sequential data using bidirectional LSTM with attention.
Key Features:

- Sequence modeling approach
- Attention highlights discriminative spectral regions
- Bidirectional processing captures context in both directions
- Cross-validation support

Performance: 56.29% Accuracy

7. Multi-Layer Perceptron (`02_model_MLP.ipynb`)

Purpose: Baseline MLP with various input configurations and feature selection methods.
Experiments:

- Plain MLP with all 200 bands
- MLP with ANOVA feature selection (top 20, 30, 100, 150 bands)
- MLP with PCA components (top 20, 30 components)
- Class-weighted loss variants

Performance: 92.20% Accuracy

8. MobileNet-Style 2D CNN (`02_model_MobileNet.ipynb`)

Purpose: Lightweight CNN using depthwise separable convolutions for efficient feature extraction.
Key Features:

- Efficient architecture with fewer parameters
- Depthwise separable convolutions reduce computational cost
- Faster training and inference
- Lower memory footprint

Performance: 99.80% Accuracy

9. Residual Network (`02_model_ResNet.ipynb`)

Purpose: ResNet architecture with residual blocks for deeper networks and better gradient flow.
Key Features:

- Residual connections enable deeper networks
- Better gradient flow through skip connections
- Robust to deeper architectures
- Excellent performance

Performance: 98.98% Accuracy

10. Recurrent Neural Network (`02_model_RNN.ipynb`)

Purpose: GRU-based RNN with attention for sequential spectral processing.
Key Features:

- Sequential processing of spectral bands
- Bidirectional GRU captures context
- Attention highlights discriminative bands
- Cross-validation support

Performance: 72.78% Accuracy

11. Adapter Warmup (`03_Adapter_Warmup.ipynb`)

Purpose: Demonstrates adapter warmup transfer learning method with frozen backbone.
Transfer Learning Strategy:

- Loads pre-trained CNN2D weights from Indian Pines
- Adds adapter layer (1×1 convolution) for different band counts
- Trains only the adapter layer with frozen backbone
- Fast training with minimal parameters

External Datasets:

- **Salinas:** Agricultural scene with 16 classes
- **Pavia:** Urban scene with 9 classes
- **PaviaU:** University of Pavia scene with 9 classes

12. Fine-Tuning (`03_FineTuning.ipynb`)

Purpose: Demonstrates fine-tuning transfer learning method with unfrozen backbone.

Transfer Learning Strategy:

- Loads pre-trained CNN2D weights from Indian Pines
- Unfreezes backbone layers (or last layers)
- End-to-end training of adapter and backbone
- Two-phase approach: adapter warmup → fine-tuning

Key Features:

- Full model training after adapter warmup
- Better adaptation to target dataset
- Comprehensive evaluation with confusion matrices

🚀 Getting Started

Prerequisites

```
# Required Python packages
torch >= 1.9.0
```

```
numpy >= 1.21.0
scikit-learn >= 0.24.0
matplotlib >= 3.3.0
scipy >= 1.7.0
pandas >= 1.3.0
```

Installation

1. Clone the repository:

```
git clone <repository-url>
cd Hyperspectral_Image_Classification
```

2. Install required packages:

```
pip install torch numpy scikit-learn matplotlib scipy pandas
```

3. Place dataset files in the `data/` directory

Usage

1. **Start with Data Preprocessing:** Run `01_Clean_Preprocess.ipynb`
2. **Explore the Data:** Run `indian_pines_eda.ipynb`
3. **Train Models:** Run any of the model notebooks
4. **Transfer Learning:** Run `03_Adapter_Warmup.ipynb` or `03_FineTuning.ipynb`

📊 Model Comparison

Model	Architecture Type	Input Format	Parameters	Key Features
3D CNN	3D Convolutions	3D Patches	~140K	Simultaneous spatial-spectral features
2D CNN	2D Convolutions	2D Patches	~380K	Spatial-spectral learning, best performance
1D CNN	1D Convolutions	Pixel Spectra	~1.7M	Spectral patterns, no spatial context
ResNet	Residual Blocks	2D Patches	~710K	Deep networks, skip connections
MobileNet	Depthwise Separable	2D Patches	~110K	Efficient, lightweight
LSTM	Bidirectional LSTM	Spectral Sequences	~2.5M	Sequential modeling, attention
RNN/GRU	Bidirectional GRU	Spectral Sequences	~2.0M	Sequential processing, attention
MLP	Fully Connected	Pixel Spectra	~86K	Baseline, fast training

📈 Results Summary

Indian Pines Dataset Performance

Model	Test Accuracy	Test Kappa	Test F1 (Macro)
3D CNN	99.61%	99.55%	99.46%
2D CNN	99.76%	99.72%	99.72%
1D CNN	93.51%	92.60%	94.97%
ResNet	98.98%	98.83%	99.37%
MobileNet	99.80%	99.78%	99.74%
LSTM	56.29%	50.92%	60.00%
RNN/GRU	72.78%	69.51%	77.53%
MLP	92.20%	91.12%	92.60%

🔑 Key Findings

1. **2D CNN achieves the best performance** on Indian Pines dataset, making it ideal for transfer learning.
2. **Spatial context is crucial** - Patch-based models (2D CNN, 3D CNN, ResNet, MobileNet) significantly outperform pixel-based models (1D CNN, MLP).
3. **3D CNN effectively captures** both spatial and spectral features simultaneously, achieving excellent results.
4. **MobileNet provides a good balance** between accuracy and efficiency with significantly fewer parameters.
5. **Sequence models (LSTM, RNN)** show lower performance, suggesting that spectral band order is less critical than spatial-spectral relationships.

6. **Transfer learning is effective** - Pre-trained models generalize well to external datasets with minimal fine-tuning.

📝 Notes

- All models use **stratified train/validation/test splits** (75%/5%/20%) to maintain class balance
- **Early stopping** is implemented based on validation macro F1 score
- **Weighted random sampling** is used during training to handle class imbalance
- All experiments use **fixed random seeds** (42) for reproducibility
- Models are saved with checkpoints in `outputs/runs_*/` directories
- Comprehensive evaluation includes confusion matrices, classification reports, and prediction maps

---

🔬 Additional Experiments & Techniques

Several advanced techniques and experimental approaches were implemented and evaluated during this project, though they were not extensively discussed in the main report. This section provides insights into these additional experiments and their outcomes.

Cross-Validation & Hyperparameter Optimization

- **5-Fold Cross-Validation:** Implemented across multiple models (1D CNN, MLP, LSTM, RNN) to ensure robust performance evaluation and reduce variance in results. Cross-validation was particularly useful for models with hyperparameter search capabilities.
- **Random Search & Grid Search:** Comprehensive hyperparameter optimization was performed for 1D CNN and MLP models, exploring learning rates, batch sizes,

dropout rates, and architecture variations to find optimal configurations.

Feature Selection & Analysis

- **ANOVA Feature Selection:** Statistical feature selection using Analysis of Variance (ANOVA) F-test was applied to identify the most discriminative spectral bands. Experiments were conducted with top 20, 30, 100, and 150 bands selected based on ANOVA scores.
- **Gradient-Based Feature Ranking:** Implemented gradient-based feature importance analysis for 1D CNN models to identify which spectral bands contribute most significantly to classification decisions. This technique provides interpretability by highlighting discriminative spectral regions.
- **PCA Dimensionality Reduction:** Principal Component Analysis was explored as an alternative to band selection, with experiments using top 20 and 30 principal components to reduce input dimensionality while preserving variance.

Key Observations & Limitations

Why These Techniques Didn't Improve Accuracy:
The top-performing models (2D CNN, 3D CNN, MobileNet, ResNet) achieved near-perfect accuracy (99%+) on the Indian Pines dataset. These models demonstrated exceptional performance without requiring additional regularization or feature selection techniques. The high accuracy suggests that:

- The dataset is well-suited for deep learning approaches with sufficient training data
- Spatial-spectral feature learning in patch-based models captures discriminative patterns effectively
- Additional regularization techniques (dropout, feature selection) were unnecessary as models were not overfitting
- The models reached a performance ceiling where further optimization provided marginal gains

Future Work & Time Constraints

While feature selection techniques (ANOVA, gradient-based ranking) were implemented and evaluated, a comprehensive analysis of **model size reduction through feature selection** was not completed due to time constraints. Potential future work includes:

- Systematic evaluation of reduced-band models to create more compact architectures
- Trade-off analysis between model accuracy and computational efficiency
- Development of lightweight models for resource-constrained deployment scenarios
- Integration of feature selection into the training pipeline for end-to-end optimization

These techniques remain valuable for scenarios where model size, inference speed, or interpretability are critical constraints, even if they did not improve accuracy for the current high-performing models.

📚 References

- Indian Pines dataset: AVIRIS sensor, NASA
- Related papers in `papers/` directory
- Report: `NN_Report (2).pdf`

👤 Author

Jaykumar Darji
Department of Electrical Engineering
San Jose State University

Course: EE258 - Neural Networks

Acknowledgments: NASA AVIRIS for the Indian Pines dataset | San Jose State University Department of Electrical Engineering | Course instructor and teaching assistants

Last Updated: 2024