EE 413: Applied Digital Signal Processing

Project: Image Compression and Classification with Deep Learning

Abdullah F. Al-Battal, PhD

Department of Electrical Engineering
King Fahd University of Petroleum and Minerals

Spring, 2025 (242)

Project Overview

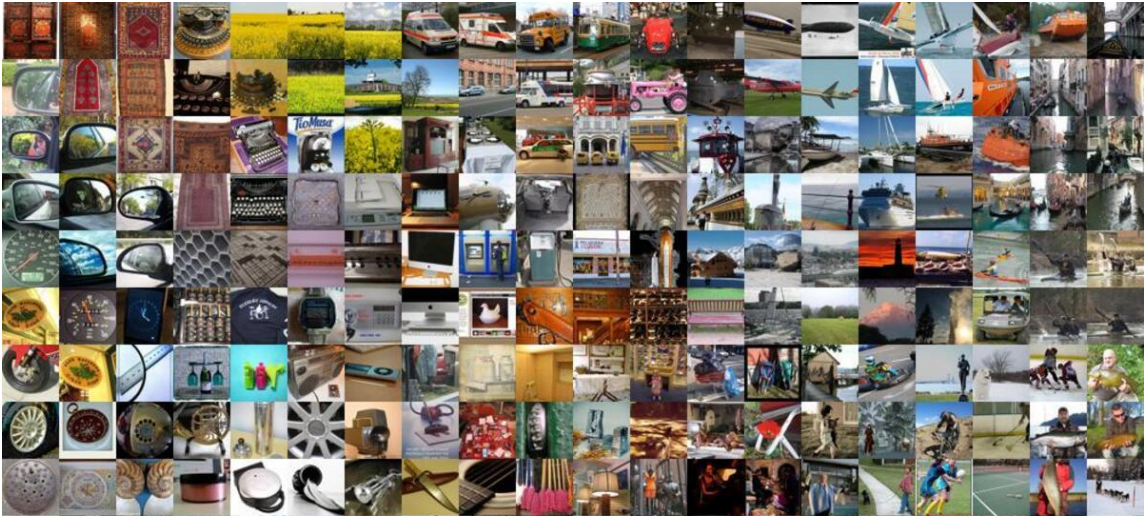
- **Focus:** Image compression, deep learning, and compressed sensing
- **Team structure:** Groups of 4 students
- **Duration:** 2-3 weeks
- **Dataset:** Mini-ImageNet
- **Application of course topics:**
  - Topic 1: Wavelet Transform
  - Topic 2: Convex Optimization
  - Topic 3: Compressed Sensing

Learning Objectives

By completing this project, you will:

- Apply pre-trained deep learning models to image classification tasks
- Implement wavelet-based image compression at various compression ratios
- Evaluate the impact of compression on deep learning models' classification performance
- Implement and evaluate compressed sensing techniques for image acquisition
- Compare model performance across different compression scenarios
- Gain experience with PyTorch for deep learning and image processing
- Develop technical presentation and collaborative coding skills
- Learn professional GitHub usage for code sharing and documentation

Dataset: Mini-ImageNet - Dataset Card [Example Images]



Dataset: Mini-ImageNet

Dataset characteristics:

- Subset of ImageNet with 100 classes
- 600 images per class (500 training, 100 testing)
- Resized to 84×84 pixels (original ImageNet images are 224x224 pixels).
- Diverse object categories

Advantages:

- Well-suited for transfer learning
- Sufficient to demonstrate compression effects
- Compatible with pre-trained models
- Rich visual content for compression experiments
- Small in size to allow for timely completion of the project

More Info: Click Here **Download: Click Here**

Project Requirements: Model Selection and Fine-tuning

1. Model Selection and Baseline Establishment

- Select two pre-trained models from PyTorch model zoo (e.g., ResNet, MobileNet, VGG, EfficientNet)
- Adapt these models for Mini-ImageNet classification
  - Might need to resize the images to 96x96 pixels to allow for divisibility by 2^5, needed for most models.
- Fine-tune the selected models on the original Mini-ImageNet training set
- Establish baseline performance metrics on the uncompressed test set
- Document model architecture, hyperparameters, and training process
- Implement proper validation strategies

Project Requirements: Wavelet Compression Analysis

2. Wavelet-based Image Compression

- Implement wavelet transform for image compression
- Create compressed versions of the test set at multiple compression ratios:
  - High compression (e.g., 10:1 ratio)
  - Medium compression (e.g., 5:1 ratio)
  - Low compression (e.g., 2:1 ratio)
  - **Note:** The ratios above refer to the ratio of all coefficients to the coefficients kept intact (not thresholded).
- Evaluate model performance degradation across compression levels
- Analyze which image classes are most affected by compression
- Visualize original vs. compressed images
- Implement proper wavelet coefficient thresholding techniques

Project Requirements: Fine-tuning on Compressed Data

3. Training on Compressed Data

- Create compressed versions of the training set using wavelet transform
- Fine-tune models on the compressed training data
- Compare performance with models trained on uncompressed data
- Analyze how training on compressed data affects robustness
- Determine optimal compression level for training
- Evaluate if models fine-tuned on compressed data generalize better to compressed test images
- Document changes in fine-tuning hyperparameters, if any

Project Requirements: Compressed Sensing Implementation

4. Compressed Sensing Experiment

- Select 20 diverse test images from different classes
- Implement compressed sensing acquisition and reconstruction:
  - Create random measurement matrices
  - Simulate CS measurements at different sampling rates [0.25, 0.5 and 0.75]
  - Implement a suitable reconstruction algorithms (e.g., L1-minimization)
- Compare reconstruction quality across sampling rates
- Test reconstructed images on both model versions:
  - Models fine-tuned on original data
  - Models fine-tuned on compressed data
- Analyze performance differences and determine which approach is more robust

Timeline and Deliverables

Week 1:

- Dataset download and exploration
- Model selection and initial fine-tuning
- Implementation of wavelet compression
- Set up GitHub repository

Week 2:

- Fine-tuning on compressed data
- Compressed sensing implementation
- Performance evaluation
- Comparative analysis

Week 3:

- Complete analysis and visualizations
- Finalize GitHub repository
- Prepare presentation

Final Deliverables:

- GitHub code repository
- 15-20 minutes presentation (slides and video recording)
- Individual Contribution Statement
- Teammate Evaluation forms

GitHub Repository Requirements

- **Repository Structure:**
  - Well-organized directory structure
  - Clear README.md with project description and setup instructions
  - requirements.txt with all dependencies
- **Code Quality:**
  - Well-commented Python code
  - Modular design with clear function/class responsibilities
  - Error handling
- **Documentation:**
  - Function docstrings explaining purpose, parameters, and returns
  - Markdown files explaining methodology
  - Sample results and visualizations
  - Usage examples

Presentation Guidelines

15-20 minute presentation

- All team members must participate equally

Required content:

- Introduction (1-2 minutes)
- Model selection and fine-tuning approach (1-2 minutes)
- Wavelet compression implementation and results (2-4 minutes)
- Fine-tuning on compressed data results (2-4 minutes)
- Compressed sensing experiments and findings (3-4 minutes)
- Discussion and Conclusions and future work (1-2 minutes)
- **Notes:**
  - Include relevant visualizations
  - Demonstrate code functionality with examples
  - Focus on DSP concepts and their application to deep learning

Evaluation Criteria

Presentation (70%):

- Technical content and depth (20%)
- Implementation details (15%)
- Results analysis (15%)
- Visualizations and demos (10%)
- Delivery and organization (10%)

Code Quality (30%):

- Correctness and completeness (10%)
- Code organization (5%)
- Documentation (5%)
- Reproducibility (5%)
- GitHub organization (5%)

Note: Individual grades will be adjusted based on teammate evaluations.

Teammate Evaluation Process

- Each team member will complete a confidential teammate and self evaluation form
- Evaluations will assess:
  - Technical contribution
  - Reliability and meeting attendance
  - Teamwork and communication
  - Initiative and leadership
  - Problem-solving abilities
- Team members will indicate contribution for each member
- Individual grades may be adjusted up or down based on evaluations
- **Maximum adjustment factor:** 0.7 to 1.1 of team grade capped at the project full mark score [Minimum factor can can be further lowered below 0.7 if temamates report lack of participation of a specific teammate that is validated/proven]
- Self-evaluation will be compared with teammate assessments

Important Reminders

- **GitHub Repository:**
  - Create repository early
  - Use meaningful commit messages
  - All team members should contribute
  - Submit final repository link via Blackboard
- **Team Collaboration:**
  - Schedule regular meetings
  - Distribute tasks evenly
  - Document decisions and progress
- **Deadlines:**
  - Presentations: TBD via Blackboard
- **AI Usage Policy:**
  - **Follow the course AI policy as outlined in the syllabus and Project 1 slides**
  - **Document any AI assistance appropriately**

Technical Guidance

- **Python Libraries:**
  - PyTorch: Deep learning framework with model zoo
  - torchvision: Dataset loading and transformations
  - PyWavelets (pywt): Wavelet transform implementation
  - scikit-learn: Evaluation metrics and utilities
  - numpy and scipy: Numerical computations
  - matplotlib: Visualization
- **DSP Concepts to Focus On:**
  - Wavelet transform for multi-resolution analysis
  - Thresholding techniques for compression
  - Random sampling matrices for compressed sensing
  - L1-minimization and convex optimization
  - Time-frequency representation of images
  - Feature preservation under compression
- **Remember:** Connect your implementation to course concepts

Potential Challenges

Common Challenges:

- Selecting optimal wavelet basis for image compression
- Balancing compression ratio with classification performance
- Implementing efficient compressed sensing reconstruction algorithms
- Managing computational resources during model fine-tuning
- Finding optimal hyperparameters for models on compressed data
- Comparing different models fairly across compression scenarios

Tips for Success:

- Start with exploratory data analysis to understand the dataset
- Implement simpler techniques first before advancing to complex ones
- Create visualization tools early to help with debugging
- Test on small subsets before scaling to the full dataset
- Document your work continuously

Implementation Examples

Wavelet Compression:

- Discrete Wavelet Transform (DWT)
- Hard vs. soft thresholding
- Different wavelet families (Haar, Daubechies, etc.)

Pre-trained Models:

- ResNet (18, 34, 50)
- MobileNetV2/V3
- VGG-16/19
- EfficientNet
- DenseNet

Compressed Sensing:

- Random Gaussian matrices
- Bernoulli measurement matrices
- LASSO reconstruction
- Total Variation minimization

Visualization Examples:

- Original vs. compressed images
- Reconstructed CS images
- Confusion matrices
- Performance vs. compression ratio plots
- Class-wise accuracy comparisons