Deep Learning for Dynamic Network Analysis (DLDNA) - Course Project Guidelines

Overview

This project allows students to apply deep learning techniques learned throughout the course to solve a practical problem relevant to civil engineering or related fields. Students will work in groups of 2-5 members to implement, test, and evaluate a deep learning solution for a real-world application.

Timeline

- **Project Proposal Due:** [Session 4] This is not graded and the format is flexible. It is just to ensure you have a plan and to catch problems early.
- **Final Submission:** [The day before final presentation] - **NO EXTENSIONS** (except documented exceptional circumstances)
- **Presentations:** [Session 10]

Grading Distribution

Component	Weight	Evaluation Type
Technical Report	40%	Group grade (same for all members)
Presentation & Q&A	40%	Individual grade (based on contribution and understanding)
Code Repository	20%	Group grade (same for all members)

Deliverable 1: Technical Report (40%)

Format Requirements

- **Length:** 6-10 pages (excluding references and appendices)
- **Format:** PDF, clear formatting with section headers
- **Language:** English, clear technical writing
- **Submission:** Via Moodle.

Report Structure

1. Title & Project Summary (5%)

- **Title:** Clear description of your application/problem
- **Executive Summary:** 200 words describing what you built and why it matters

2. Problem Context & Motivation (10%)

- **Application Domain:** What civil engineering or practical problem are you addressing?
- **Current Approaches:** How is this problem currently solved (if at all)?
- **Why Deep Learning?** Justify why DL is suitable for this problem

- **Project Goals:** What do you aim to achieve with your solution?

3. Data & Preprocessing (15%)

- **Data Description:**
    - Source and nature of data (images, time series, sensor data, etc.)
    - Dataset size and characteristics
    - Key statistics and visualizations
    - Any data quality issues and how you handled them
- **Preprocessing Pipeline:**
    - Data cleaning steps
    - Feature engineering (if any)
    - Data augmentation techniques
    - Train/validation/test split strategy

4. Deep Learning Solution (25%)

- **Model Selection:**
    - Which architecture(s) did you choose and why?
    - Did you use pre-trained models or train from scratch?
    - Key hyperparameters and how you selected them
- **Implementation Details:**
    - Training process (optimizer, learning rate, epochs)
    - Regularization techniques used
    - Computational resources and training time
    - Any implementation challenges faced

5. Results & Evaluation (25%)

- **Performance Metrics:**
    - Choose metrics relevant to your application
    - Present results clearly (tables, graphs, confusion matrices)
    - Compare different model variants or configurations
- **Practical Validation:**
    - Test cases or examples showing your model in action
    - Visual results (if applicable)
    - Performance on edge cases or difficult examples
    - Comparison with traditional methods (if available)

6. Discussion & Applications (15%)

- **Key Findings:** What worked well? What didn't?
- **Practical Deployment:** How could this be used in practice?
- **Limitations:** Be honest about where your solution falls short
- **Future Improvements:** Realistic next steps to improve the solution

7. Conclusion (5%)

- Brief summary of achievements and main takeaways

8. References

- All sources properly cited (papers, tutorials, documentation)

9. Mandatory Appendices

A. Individual Contributions (MANDATORY - without this, group receives 0%)

- **Detailed table** showing who did what:
    - Data collection/preprocessing
    - Model implementation
    - Experiments and testing
    - Report writing
    - Code development
    - Each member must have substantial contributions

B. AI/LLM/Chatbot Usage Declaration (MANDATORY)

- **Complete transparency required**
- List all AI tools used (ChatGPT, Claude, Copilot, etc.)
- **For each AI usage, specify:**
    - Which parts of the project involved AI assistance
    - Type of assistance (code generation, debugging, writing, concept explanation)
    - How you verified and understood the AI-generated content
- **Remember:** You must understand everything in your project. Inability to explain AI-generated content during Q&A results in a grade of zero
- **Example format:**
    - Data preprocessing code: Used ChatGPT to generate initial pandas operations, modified for our specific needs, verified output manually
    - Model architecture: Discussed design with Claude, implemented ourselves
    - Report section 3.2: Grammar check with Grammarly

C. GitHub Repository Link (MANDATORY) - Public repository with complete code

D. Data Sources - Links to datasets or description of data collection process

Deliverable 2: Presentation & Q&A (40%)

Presentation Format

- **Duration:** 15 minutes presentation + 15 minutes Q&A
- **Focus:** Practical demonstration and results
- **Structure:**
    1. Problem Introduction (2 min)
    2. Data and Challenges (3 min)
    3. Your DL Solution (4 min)
    4. Live Demo or Results Walkthrough (4 min)
    5. Lessons Learned & Conclusions (2 min)

Evaluation Criteria

- **Group Component (20%):**
    - Clear explanation of the problem and solution
    - Quality of demonstrations/visualizations
    - Time management
- **Individual Component (20%):**
    - Ability to answer questions about your contributions
    - Understanding of the deep learning concepts used
    - **Critical:** Must be able to explain any AI-generated components
    - Questions will be based on declared individual contributions

Q&A Expectations

- All members must attend and be ready to answer questions
- General questions: Any member should be able to explain the overall approach
- Specific questions: Based on individual contribution declarations
- **AI Usage Test:** Be prepared to explain any code or concept in detail

Deliverable 3: Code Repository (20%)

Repository Requirements (Aim for Best Practices)

We encourage you to follow software engineering best practices. While perfect adherence isn't required, make your code as clean and organized as possible. Here's a suggested structure:

Suggested Repository Structure

```
project-name/
├── README.md (REQUIRED)
├── requirements.txt or environment.yml (REQUIRED)
├── data/
│   └── (your data files or download instructions)
├── notebooks/
│   └── (Jupyter notebooks for experiments)
├── src/ or code/
│   └── (Python scripts if you have any)
├── models/
│   └── (saved model files, if not too large)
└── results/
    └── (figures, predictions, evaluation results)
```

Minimum README Requirements

Your README must include:

1. **What This Project Does** (1-2 paragraphs)
2. **How to Run the Code** - Python version and main libraries needed - Installation instructions - How to run training/testing
3. **Dataset Information** - Where to get the data - Any preprocessing needed
4. **Quick Start Example** - A simple command or notebook to run
5. **Team Members** (all names)

Code Guidelines (Do Your Best)

- **Make it runnable:** Someone should be able to clone and run your code
- **Comment complex parts:** Help others (and future you) understand
- **Include at least one notebook:** Show your experimental process
- **Organization:** Try to separate data loading, model definition, and training
- **No need for perfect structure:** Focus on functionality over form

What We're Looking For

- Code that works and solves the stated problem
- Evidence of experimentation and iteration
- Clear documentation of how to use it
- Reasonable organization (doesn't have to be perfect)

Dataset Guidelines

Choosing Your Dataset

Pick data relevant to civil engineering or your domain of interest:

- **Infrastructure:** Bridge inspection images, road quality data, structural health monitoring
- **Environmental:** Weather data, flood prediction, air quality monitoring
- **Construction:** Equipment detection, safety monitoring, progress tracking
- **Transportation:** Traffic flow, accident prediction, route optimization
- **Geotechnical:** Soil classification, landslide prediction, seismic data
- **Urban Planning:** Satellite imagery analysis, building detection, land use classification

Dataset Requirements

- **Size:** Large enough to train a neural network (typically >1000 samples)
- **Complexity:** Should benefit from deep learning (not solvable with simple statistics)
- **Availability:** Must be accessible to instructors for evaluation
- **Practicality:** You should be able to work with it given time and resource constraints

Options

1. **Use course-provided datasets** (see course repository)
2. **Public datasets** from Kaggle, UCI, government databases
3. **Create your own** (with proper documentation)

Important Policies

AI Usage Policy

- **You are FREE to use AI tools** (ChatGPT, Claude, Copilot, etc.) for any part of the project
- **Two requirements:**
    1. Full transparency in Appendix B

2. Complete understanding of what you submit

- **Testing:** Random Q&A questions will verify understanding
- **Consequence:** Inability to explain AI-generated content = grade of zero

Submission Policy

- **Deadline is STRICT:** No late submissions accepted
- **Exception only for documented exceptional circumstances:**
  - Medical emergencies (with documentation)
  - Family emergencies (with documentation)
  - Must notify instructor BEFORE deadline when possible

Academic Integrity

- Using AI is allowed WITH declaration
- Copying from other groups is NOT allowed
- Using external code is allowed WITH attribution

Tips for Success

1. **Choose familiar domains:** Pick problems related to your CE background
2. **Start simple:** Basic CNN for image classification or RNN for time series
3. **Focus on application:** Don't worry about novel algorithms, focus on solving a real problem
4. **Leverage pre-trained models:** Transfer learning is perfectly acceptable
5. **Document your process:** Show your learning journey, including failures

Practical Approach

1. Choose problem, find/create dataset
2. Data exploration and preprocessing
3. Implement baseline model
4. Improve model, run experiments
5. Finalize results, create visualizations
6. Write report, prepare presentation

Using AI Effectively (if you use AI)

- **Good uses:** Debugging, visualization, code & concept explanation
- **Stay engaged:** Read everything AI generates, ask for evidence, proof and references, assess their correctness
- **Still your work:** Understand and modify AI suggestions
- **Document everything:** Keep track of what AI helped with
- **Stay in charge:** Use AI as an assistant, not a ghost writer or a mastermind

Resources and Support

Getting Help

1. Ask in class.
2. Course discussion forum in Moodle.
3. Instructor consultation (book via email).

Computational Resources

- Google Colab (free GPUs)
- Your own machines (for smaller models)

Frequently Asked Questions

Q: What if my model performance isn't great? A: That's okay! Focus on understanding why and documenting your attempts to improve it.

Q: Can we apply DL to a standard CE problem that doesn't typically use DL? A: Yes! In fact, this could be very interesting. Just justify why you think DL might help.

Q: Do we need to innovate or can we implement existing solutions? A: Implementation of existing solutions for new applications is perfectly fine.

Q: Can I use ChatGPT to write my entire code? A: Yes, IF you declare it and can explain every line during Q&A. Though this is not reccommended and is very risky.

Q: What happens if I can't explain the AI-generated code during Q&A? A: You risk getting zero for the project. Make sure you understand everything you submit!

Project Examples (for inspiration)

1. **Traffic flow prediction** using LSTM with sensor data
2. **Building energy consumption** forecasting with neural networks
3. **Flood prediction** from weather and geographical data
4. **Construction site safety** monitoring using object detection
5. **Soil type classification** from borehole data
6. **Structural damage assessment** from earthquake sensor data
7. **Bridge vibration analysis** for health monitoring