CSCI-GA 2271: Assignment 1 — Completion Planner & Guide

# Page 1: Project Planner & Progress Tracking

Use this page to track your progress across the three parts of the assignment. Update status, timing, and completion as you work.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task Name | Estimated Hours | Difficulty | Status / Notes | File to Edit |
| Part 1: Filtering and Convolutions | 5–6 hrs | 🟡 Medium | Not started | A1\_Filtering.ipynb |
| Part 2: AutoDiff — Forward & Backward | 4–5 hrs | 🔴 Hard | Not started | auto\_diff.ipynb |
| Part 3: CNN from Scratch | 8–10 hrs | 🔴 Hard | Not started | convolutional\_networks.ipynb |

# Page 2: Part 1 — Filtering and Convolutions (30 pts)

Focus: Implement and visualize convolution, padding, and edge detection.

• Set Up Environment — Open A1\_Filtering.ipynb in Jupyter or Colab. Ensure zebra.png and cameraman.png are accessible in the same directory. (🟢 Easy, ~0.5 hr)

• Implement Padding Methods — Write functions for zero, reflect, and replicate padding. Test by printing small array outputs. (🟡 Medium, ~1 hr)

• Manual Convolution — Implement 2D convolution manually using nested loops (no cv2). Check boundary behavior with padding. (🔴 Hard, ~2 hrs)

• Edge Detection — Use 1D filters (Sobel or Prewitt) to compute gradients. Visualize horizontal/vertical edges. (🟡 Medium, ~1.5 hrs)

• Analysis & Markdown — In markdown cells, summarize in 1–2 lines how filter size/padding affects results. (🟢 Easy, ~0.5 hr)

# Page 3: Part 2 — Forward-mode and Backward-mode AutoDiff (20 pts)

Focus: Implement classes ValueFwd and ValueBwd to handle forward and backward automatic differentiation.

• Understand the Computation Graph — Review the chain rule f(X)=a(b(c(X))) and how Jacobians compose. (🟢 Easy, ~0.5 hr)

• Implement ValueFwd — Track values and partial derivatives through each function call. Each operation should store value and derivative. (🔴 Hard, ~2 hrs)

• Implement ValueBwd — Store references to children; perform reverse accumulation to compute gradients. (🔴 Hard, ~2 hrs)

• Validation — Compare results of manual differentiation vs your implementation on test functions. (🟡 Medium, ~1 hr)

# Page 4: Part 3 — Convolutional Neural Networks (50 pts)

Focus: Build and train a CNN from scratch using only tensor operations.

• Read the Notebook — Go through convolutional\_networks.ipynb for context and structure. (🟢 Easy, ~0.5 hr)

• Modify convolutional\_networks.py — Complete only the code blocks marked for implementation (do not modify other code). (🔴 Hard, ~3 hrs)

• Forward Pass — Implement convolution, ReLU, and pooling operations from scratch. (🔴 Hard, ~2.5 hrs)

• Backward Pass — Implement backward propagation for convolution and fully connected layers. (🔴 Hard, ~2 hrs)

• Training & Visualization — Run the notebook cells to train your CNN; visualize loss/accuracy. (🟡 Medium, ~1.5 hrs)

• Final Checks — Ensure all cells are executed with visible outputs before submission. (🟢 Easy, ~0.5 hr)