# Week 2 Implementation Plan - My Strategy for Building This System

DS 340W Data Science Capstone - Week 2

Figuring Out How to Actually Build This Thing

September 28, 2025

## What I'm Trying to Accomplish

After spending the past week diving deep into Weber et al.'s methodology and analyzing the IncidentsDataset, I've been thinking a lot about how to actually turn all this theoretical knowledge into a working disaster detection system. The goal is ambitious but exciting: I want to create something that can look at images from social media in real-time and accurately identify what kind of disaster or emergency situation is happening.

The more I think about it, the more I realize this isn't just about building a machine learning model. This is about creating an entire system that could potentially help emergency responders react faster to disasters. That's both thrilling and a little intimidating, but I'm ready to tackle the challenge step by step.

## How I'm Thinking About the System Architecture

When I started sketching out the architecture for this system, I realized I need to think about four main pieces that all have to work together seamlessly. The first piece is what I'm calling the data ingestion pipeline, which will handle collecting images from social media APIs in real-time. This is actually more complicated than it sounds because I need to manage API rate limits, handle different image formats, and deal with the constant stream of new content.

The second component is the preprocessing module, and this is where I'll standardize all the incoming images and filter out anything that's obviously not useful. I've learned from my dataset analysis that consistency in image quality and format is crucial for getting good results from the machine learning model.

The third piece is the heart of the system - the multi-task CNN engine that will actually do the incident detection and place recognition. This is where all my research into ResNet-50 architectures and multi-task learning will come together.

Finally, I need an output interface that doesn't just spit out raw classification results, but provides meaningful information including confidence scores that emergency responders can actually use to make decisions.

## The Technical Requirements I Need to Meet

One thing I've learned from my research is that this kind of system has some serious hardware requirements. I'm planning to use an NVIDIA RTX 4090 or something equivalent with at least 24GB of VRAM. Without sufficient GPU memory, the training process would take forever, and real-time inference would be impossible.

I've decided to build everything in PyTorch 2.0 with CUDA 12.0 support because that gives me the flexibility I need for implementing the multi-task architecture and the performance I need for real-time processing. For storage, I'm planning on 2TB of SSD space to handle the massive dataset and store model checkpoints during training.

## My Development Timeline Strategy

I've been thinking about this project in three main phases, each building on the previous one. In Phase 1, which I'm planning for weeks 3-4, I want to get the foundation solid. This means downloading and setting up the complete IncidentsDataset, building the preprocessing pipeline, implementing the base ResNet-50 architecture, and getting the multi-task heads working together properly.

During this phase, I know I'll probably run into unexpected challenges with data handling and making sure all the components can talk to each other correctly. I'm budgeting extra time for debugging and making sure the foundation is rock solid before moving on.

Phase 2, planned for weeks 5-6, is all about getting the model trained and working well. I'll start with baseline model training using standard cross-entropy loss, then implement and optimize the class-negative loss function that Weber et al. showed was so effective. This phase will also involve a lot of hyperparameter tuning and validation work to make sure the model is actually learning what I want it to learn.

Phase 3, weeks 7-8, is where I get to implement the really cool stuff. This is when I'll integrate the multi-scale attention mechanism that I'm most excited about, implement the advanced data augmentation strategies I identified in my dataset analysis, and focus on performance optimization to make sure the system can handle real-time inference.

## The Challenges I'm Worried About

I'd be lying if I said I wasn't nervous about some potential problems. The biggest technical risk I see is that even with over a million images, the dataset might not be sufficient for some of the rarer disaster types. My plan to mitigate this is to get really creative with data augmentation and leverage transfer learning from models pre-trained on general image recognition tasks.

Computational resources are another concern. While I have access to good hardware, training these kinds of models can be unpredictable in terms of time and resource requirements. I'm keeping cloud GPU instances as a backup option, though I hope I won't need them given the cost.

Model overfitting is something I'm particularly worried about given the complexity of the multi-task architecture. I'm planning to address this through careful cross-validation and aggressive regularization strategies, but it's definitely something I'll need to monitor closely throughout the training process.

## How I'll Know If This Is Actually Working

I've been thinking a lot about how to measure success for this project. The primary metric I'm aiming for is achieving a Mean Average Precision (mAP) of over 70% on the test set. From my research, this would put the system in the range of state-of-the-art performance for disaster detection tasks.

I'm also really focused on reducing false positive rates because in an emergency response context, false alarms can waste precious resources and potentially put people at risk. I want to achieve at least a 40% reduction in false positive rates compared to baseline approaches.

Beyond the quantitative metrics, I'm also thinking about qualitative measures of success. Does the system make sense to domain experts? Can it handle the kinds of edge cases and unusual situations that come up in real disasters? Can it process images fast enough to be useful in actual emergency response scenarios?

## What I'm Most Excited About

As I've been putting together this implementation plan, I keep coming back to how much I'm looking forward to seeing this system come together. The combination of cutting-edge computer vision techniques, real-world social good applications, and the technical challenges involved makes this exactly the kind of project I love working on.

I'm particularly excited about implementing the multi-scale attention mechanism because I think that's where I might be able to make some novel contributions beyond just replicating Weber et al.'s work. The idea of having the system pay attention to different levels of detail simultaneously feels like it could really improve performance, especially for complex disaster scenes.

The timeline is aggressive, but I think it's doable if I stay focused and don't get sidetracked by trying to perfect every small detail along the way. The key will be getting something working end-to-end quickly, then iterating and improving from there.

I'm ready to start building!