# Week 2 Architecture Documentation - My Learning Journey

**DS 340W Data Science Capstone - Week 2**

*Understanding System Design Through Real Discovery*

September 28, 2025

## My Introduction to This Complex System

When I first started diving into Weber et al.'s disaster detection methodology this week, I have to admit I was pretty overwhelmed by all the technical components. But as I worked through understanding each piece, I began to see how everything fits together into what's actually a really elegant system design. Let me walk you through what I've learned about implementing this multi-task disaster detection architecture.

## Getting My Head Around the Big Picture

So here's what I discovered about how this whole system works: it's basically a smart computer vision setup that can look at disaster images and figure out multiple things at once - what kind of disaster it is, where it might be happening, and how confident we should be about those predictions. The really cool part is that instead of having separate systems for each task, everything shares the same "brain" (the feature extraction part) but then branches out to make different kinds of decisions.

When I was trying to understand why this approach works so well, I realized it's kind of like how humans process visual information - we don't look at a scene three separate times to identify objects, location, and our confidence level. We do it all at once, and that's exactly what this architecture is trying to mimic.

## Breaking Down the Core Components (What I Learned)

### The Input Processing - My First "Aha!" Moment

One thing that really clicked for me this week was understanding why we need to standardize all the input images. At first, I thought "why resize everything to 224x224? Aren't we losing information?" But then I realized that having consistent input sizes is crucial for the neural network to work properly. It's like having a standardized test format - everyone needs to answer in the same structure.

Here's what happens to every image that comes into our system:

- Gets resized to 224x224 pixels with 3 color channels

- Goes through normalization using ImageNet statistics (which I learned is like giving the network a familiar "language" to work with)

- During training, we throw in some data augmentation to make the model more robust

### The Feature Extraction Backbone - Where the Magic Happens

This part took me a while to really grasp, but I'm excited about how much I understand now! We're using ResNet-50 as our shared backbone, and here's how I think about each component:

**Convolutional Block 1:** This is like the system's "first impression" - 64 filters with 7x7 kernels that start picking up basic patterns and edges

**Convolutional Block 2:** Here we bump up to 128 filters and start doing some progressive downsampling. I think of this as the system starting to recognize more complex shapes and patterns

**Convolutional Block 3:** With 256 filters, this is where spatial attention comes in. This was a concept I struggled with initially, but now I understand it's like the system learning to focus on the most important parts of the image

**Global Average Pooling:** This final step reduces everything down to 2048 features, which is like creating a compact "summary" of everything important the system found in the image

### The Multi-Task Classification Heads - The Decision Makers

What I find really fascinating is how the system branches out after feature extraction:

**Incident Classifier:** This handles 43 different disaster classes with softmax activation. When I first saw 43 classes, I thought "wow, that's a lot!" but then I realized how many different types of disasters and incidents we actually need to detect in real-world scenarios.

**Place Classifier:** 49 classes for location context - this was something I hadn't initially considered, but it makes so much sense. Knowing WHERE something is happening can be just as important as knowing WHAT is happening.

**Confidence Estimation:** This uncertainty quantification layer is probably my favorite innovation. It's like the system saying "here's what I think is happening, and here's how sure I am about it."

## My Exploration of the Novel Enhancements

### Multi-Scale Attention - The Part That Blew My Mind

When I was researching attention mechanisms this week, I came across this multi-scale approach and it just clicked. Instead of looking at images at just one resolution, the system examines them at multiple scales simultaneously:

**Scale 1 (28x28 feature maps):** Preserves high-resolution details - perfect for catching small but important visual cues

**Scale 2 (14x14 feature maps):** Focuses on mid-level patterns - great for recognizing objects and structures

**Scale 3 (7x7 feature maps):** Captures global context - helps understand the overall scene

The attention fusion mechanism that weights and combines these different scales is what really makes this approach powerful. It's like having multiple experts examine the same evidence at different levels of detail and then collaborate on the final decision.

### Advanced Data Augmentation - Making It Real-World Ready

This section really opened my eyes to the challenges of real-world deployment. The domain-specific augmentation strategies include:

**Weather simulation:** Adding fog, rain, and snow effects to training images - because disasters don't always happen in perfect weather conditions

**Lighting variations:** Simulating day/night cycles - emergency responders need systems that work around the clock

**Perspective distortion:** Training on both aerial and ground-level viewpoints - because disaster images come from drones, satellites, smartphones, security cameras, you name it

## My Deep Dive into Training Specifications

### The Hyperparameters I Had to Learn About

Getting these settings right seems like both an art and a science:

**Learning rate:** 0.001 with cosine annealing schedule - I learned that this helps the model learn efficiently without overshooting optimal solutions

**Batch size:** 32 images per GPU - this balances memory constraints with training stability

**Optimizer:** Adam with weight decay 1e-4 - Adam is apparently great for this type of problem because it adapts to different parameters

**Training epochs:** 100 with early stopping - prevents overfitting while ensuring the model fully learns the patterns

### Understanding the Loss Function Design

The multi-component loss function was initially confusing, but now I see how each piece contributes:

**Incident Classification Loss:** Uses class-negative enhanced cross-entropy, which I learned helps with imbalanced datasets (some disaster types are much rarer than others)

**Place Classification Loss:** Standard cross-entropy works well here since location classes are more balanced

**Regularization Terms:** L2 weight penalty and dropout prevent overfitting - basically keep the model from memorizing instead of learning

## How I'm Thinking About Performance Evaluation

The evaluation framework includes several metrics that each tell us something different:

**Mean Average Precision (mAP):** Perfect for multi-class evaluation - gives us a comprehensive view of performance across all disaster types

**False Positive Rate analysis:** Critical for emergency response systems where false alarms can waste precious resources

**Inference latency measurements:** Because in real disasters, every second counts

**Memory usage profiling:** Ensures the system can actually run on the hardware available to emergency responders

## My Thoughts on Deployment Architecture

When I started thinking about actually deploying this system in the real world, I realized there are so many practical considerations:

**Containerized inference servers:** Using Docker makes deployment consistent across different environments

**Load balancing:** Essential for handling the surge of images that would come in during a major disaster

**Model versioning and A/B testing:** Allows for continuous improvement without disrupting emergency operations

**Monitoring and alerting:** Because system downtime during a disaster could literally be a matter of life and death

This whole architecture documentation process has really helped me understand not just what we're building, but why each design decision matters in the context of real-world disaster response. I'm excited to see how this theoretical understanding translates into actual implementation next week!