Convolutional Neural Network Architecture in NumPy

Aaron Avram

June 14 2025

Introduction

In this writeup I am going to outline the design of my Convolutional Neural Network for image classification. However, my hope is to build a module that abstracts away individual model design and provides a framework for deploying any model type, drawing some inspiration from the archictecture of PyTorch.

High-Level Design

At the foundation of my model module is the Tensor class. This is a class with two NumPy arrays as attributes, one representing a value matrix and the other a gradient matrix. Since practically all of the components of a Neural network store value and gradient data it felt natural to couple them under a single class. The Tensor class will support gradient update and setting the gradient to zero.

On top of this class is the Layer class. This is meant to capture the core functionality of a layer in a neural network: it has an input, an (optional) list of parameters and an output, all stored as Tensor objects. The class supports a forward and a backward pass. I will create subclasses of the layer class for individual layer types, e.g. Convolutional, Dense or Activation. However, the superclass allows for polymorphic behaviour when implementing an entire model.

Finally, I am going to build the model class which has a list of layers, which chain together from input to output, using aliasing to conserve memory. My CNN model will be a particular instance of the model class with its layers structured in the standard way for image classifying CNNs.