CAI 4104: Machine Learning Engineering

Project Report: Image Classification Using a CNN

Sebastian Villamizar
villamizar.s@ufl.edu

Jason Chen
jchen8@ufl.edu

Matthew Segura
matthewsegura@ufl.edu

Adrian Lehnhaeuser
alehnhaeuser@ufl.edu

Logan Dukes
email5@ufl.edu

April 24, 2025

1 Introduction

The final project for CAI4104 is about implementing a complete ML pipeline for object classification from images of everyday objects. We classified 12 objects: pen, paper, book, clock, phone, laptop, chair, desk, water bottle, keychain, backpack, calculator. To accomplish this, we used a Convolutional Neural Network because it is well-suited to computer tasks on image data. CNNs have high memory usage during training so we leveraged computing power from HiPerGator and Google Colab. TODO: ADD SIMPLE RESULTS

2 Approach

To implement our pipeline we imported several python libraries: numpy, pandas, tensorflow, sklearn, and pytorch. These libraries helped us process the data and train the model. First, we applied label encoding to convert the object classes into integer class labels ranging from 0 to 11. These integer labels are compatible with the Sparse Categorical Cross Entropy loss function used in training our CNN model. To load and preprocess the image data we iterated over each image file and added them to a numpy array X, our feature matrix, and the labels to Y. Since the images have RGB values that range from 0 to 255, we normalized the data by dividing each pixel value by 255 to get them in the range [0, 1]. This is important because it accelerates convergence and improves numerical stability by stabilizing the gradients calculated by the optimizer, Adam. Then we used a standard seed, 42, and an 80, 10, 10 percent training, test, and validation split. We tested this data on a simple CNN architecture using tensorflow.

The architecture consists of 3 convolutional layers with filter size 3x3, increasing channel depth (32, 64, 128), each with ReLU activation and max-pooling layers. This is followed by a flattening layer, a dense layer with 256 units, a drouput layer with a 0.5 rate, and finally a dense output layer with 12 units and softmax activation. After training this model over 30 epochs using the adam optimizer and SparseCategoricalCrossentropy loss function, we achieved a training accuracy of 97% and a loss of 0.0755. But when we evaluaed it on the unseen test data, the accuracy dropped significantly to 56%. Therefore, our model was overfitting due to a lack of data augmentation. To augment the data we used a YouTube video by codebasics to guide us[1] For every input we generated multiple new samples using edge enhancement, smoothing, grayscale, gaussian blur, oversaturation, and color inversion. This increased amount of images from 4757 to 57084.

3 Evaluation Methodology

We used accuracy as our main metric (number of correct labels / total number of examples). A simple baseline to use would be to always guess the most common class. In the given dataset, the water bottle class had the highest number of examples (418) and the total number of images was 4,757, so a baseline accuracy would be

$$\frac{418}{4,757} \approx 0.0878705066218205. \tag{1}$$

We split did an 80–10–10 train-validation-test set split.

[1]Data augmentation to address overfitting: `https://www.youtube.com/watch?v=mTVf7BN7S8w&list=WL&index=2&t=1601s`.

4 Results

Write here.

5 Conclusions

Write here.