Landon Buell

August 2022

Bilinear Interpolation for Deep Image Classification

Abstract

Computer vision is a prominent and ever-growing subfield of Artificial Intelligence born out of digital image processing. As increasingly powerful processors and more disk space becomes available to the commercial, academic, and consumer worlds, the size of image datasets has also increased as well. Storing large masses of input data is a common problem in the world of AI and is routinely revisited. Similarly, most computer vision models are architected to accept images that all have a consistent size within a dataset. These problems combined highlight the need for a more efficient method of storing volumes of image data, without compromising the performance of the models that will process them. In this experiment, we explore a possible solution to this issue where we down-sample images to store them at a fraction of the disk size, and then use various interpolation techniques to rebuild them up to the original size. We explore how the down-sized, then upscaled images compare a baseline, and then offer a discussion for the viability of this as a long term solution.

1. Introduction
   1. Image processing is widely used in the modern world
      1. Computer Vision: Extract meaning from Images
      2. Types of Vision: Classification, Detection, Segmentation
         1. We will be performing classification of images
      3. Use Cases: Facial recognition, Handwriting parsing
   2. Image Formats and Sizes
      1. Images are 2D or 3D Objects
      2. Typical Image formats: PNG, PJEG, RAW
      3. Hundreds of thousands of bytes to store a single image in Disk or RAM
      4. Stores as bytes, but upscaled to single/double precision floats
   3. Power of Automation vs. Human Biology
      1. Modern computers can read/write hundreds to thousands of images/sec
      2. Must be trained to map images to objects using thousands of samples
   4. Limits of Human Vision
      1. Humans have eyesight “resolution” limited by biology
2. Related Works
3. Dataset
   1. Case Study 1 – MNIST 784 Dataset
      1. 70,000 Images of single digits 0 through 9
      2. 28 x 28 pixels (single gray-scale channel)
      3. Down-Sample to 7 x 7 (avg of 2 x 2 blocks) to simulate smaller sizes
      4. Allows images to be stored at 1/16 the size in DISK
4. Interpolation Technique – Bilinear Interpolation
5. Deep Neural Network Architecture
   1. Convolutional Neural Network
      1. Groups of Layers Conv2D + Conv2D + MaxPool2D
      2. Flatten out activations
   2. Model Configuration
      1. Optimizer – Adaptive Momentum (ADAM)
         1. Expansion of SGD
         2. Description of method
         3. Hyperparameters selection
      2. Loss Function – Categorical Crossentropy
         1. Indirect optimization
         2. Minimize to is learning
      3. Metrics – Precision, Recall, Accuracy
      4. Regularization?
         1. Motivation
         2. Other considerations
   3. Outputs
      1. Softmax activation function
         1. Exponential of each output activation
         2. L1-Norm of 1
      2. Treat outputs as probability distribution
      3. Compare to expected outputs,
6. Methodology
   1. For all Images in the full dataset
      1. Sample from (m,n,k) down to (p,q,k)
      2. Scale each pixel to unit variance and zero mean
      3. This simulates the “compacted dataset”
   2. For all Experiments, we hold constant
      1. Basic Network architecture – (28 x 28) input to (1 x 10) output
      2. ADAM Optimizer (Constant Hyper parameters)
      3. Categorical Cross entropy loss function
      4. No Regularizes for each layer
      5. (80/20) train-test split
      6. 2 epochs over full dataset
      7. Repeat Experiment 10 times
   3. Baseline
      1. Images presented to neural networks “as-is”
      2. Results processed to store as baseline
   4. Case-Study 1
      1. Downsize Image Images using Average Pooling
      2. (2 x 2) filter, (1 x 1) stride size
      3. Interpolate back up to (28 x 28)
   5. Case-Study 2
      1. Downsize Image Images using Average Pooling
      2. (2 x 2) filter, (2 x 2) stride size
      3. Interpolate back up to (28 x 28)
   6. Case-Study 3
      1. Downsize Image Images using Average Pooling
      2. (3 x 3) filter, (1 x 1) stride size
      3. Interpolate back up to (28 x 28)
   7. Case-Study 24
      1. Downsize Image Images using Average Pooling
      2. (3 x 3) filter, (2 x 2) stride size
      3. Interpolate back up to (28 x 28)
7. Results
   1. Compare Final Classification scores (Precision/Recall/F1)
      1. Compare by filter size
      2. Compare by stride size
   2. Compare LEARNING-RATE (Loss/step)
      1. Compare by filter size
      2. Compare by stride size
8. Conclusions