University of North Carolina at Charlotte  
Department of Electrical and Computer Engineering

Homework Report 7

ECGR 4105 – Introduction to Machine Learning

Author: Matthew Anderson

Date: December 10th, 2024

GitHub: <https://github.com/Mand187/Intro-to-ML>

**Problem 1: CNN**

**Objective**

(a) Build a Convolutional Neural Network (CNN) to classify CIFAR-10 images into 10 classes.

* Adjust the fully connected layer to match the number of output classes.
* Train the model for 200 epochs and validate its performance based on training time, training loss, and evaluation accuracy.
* Compare results with the fully connected network from Homework 2 in terms of training time, accuracy, and model size.

(b) Extend the CNN by adding an additional convolution layer, activation function, and pooling function.

* Adjust the fully connected layer to accommodate the updated feature dimensions.
* Train the enhanced model for 200 epochs and evaluate its performance based on training time, loss, and evaluation accuracy.
* Analyze the impact of the additional layer on accuracy and model size.
* Discuss whether the extended CNN exhibits signs of overfitting compared to the baseline model.

**Methodology**

CNN Model Definition (Problem 1A & 1B):

* The CNN consists of two convolutional layers (conv1, conv2) with ReLU activation and max pooling.
* It includes two fully connected layers (fc1 and fc2) for classification, with dropout applied between them to prevent overfitting.

Training and Evaluation (Problem 1A):

* The model is trained for a set number of epochs, using cross-entropy loss and backpropagation to update weights.
* Early stopping is implemented based on validation loss to halt training if no improvement is seen after a specified number of epochs.
* During training, both training and validation losses are recorded, and progress is printed periodically.

Refined Model and Training (Problem 1B):

* The model definition is moved to a reusable function (createCnnModel), and batch processing is handled by a helper function (processBatch).
* Mixed-precision training with autocast and GradScaler is used to improve computational efficiency on compatible hardware.
* The total training time is measured and printed, with periodic updates on training and validation losses.

**Results:**

**Problem 1A:**

**Time: 1.5 Hours**

**Epoch 1/200, Training Loss: 1.4740**

**Epoch 10/200, Training Loss: 0.4993**

**Epoch 20/200, Training Loss: 0.2267**

**Epoch 30/200, Training Loss: 0.1598**

**Epoch 40/200, Training Loss: 0.1323**

**Epoch 50/200, Training Loss: 0.1071**

**Epoch 60/200, Training Loss: 0.0996**

**Epoch 70/200, Training Loss: 0.0962**

**Epoch 80/200, Training Loss: 0.0853**

**Epoch 90/200, Training Loss: 0.0785**

**Epoch 100/200, Training Loss: 0.0770**

**Epoch 110/200, Training Loss: 0.0750**

**Epoch 120/200, Training Loss: 0.0711**

**Epoch 130/200, Training Loss: 0.0646**

**Epoch 140/200, Training Loss: 0.0677**

**Epoch 150/200, Training Loss: 0.0632**

**Epoch 160/200, Training Loss: 0.0577**

**Epoch 170/200, Training Loss: 0.0648**

**Epoch 180/200, Training Loss: 0.0628**

**Epoch 190/200, Training Loss: 0.0622**

**Epoch 200/200, Training Loss: 0.0626**

**Test Accuracy: 73.45%**

**Problem 1B:**

**Evaluation Results:**

**Average Loss: 1.3343**

**Accuracy: 52.23%**

**F1 Score: 0.5122**

**Problem 2: ResNet-10**

**Objective:**

(a) Build a ResNet-10 architecture with skip connections for CIFAR-10 classification.

* Use 10 ResNet blocks with dimensions and channels similar to those in lecture examples.
* Train the model for 200 epochs and evaluate its performance based on training time, training loss, and evaluation accuracy.
* Compare results to the extended CNN from Problem 1(b), focusing on training time, accuracy, and model size.

**Methodology:**

Residual Block:

* A custom ResidualBlock is created using two convolutional layers with Batch Normalization and ReLU activation.
* The block adds the input (identity) to the output of the second convolution to create the residual connection.

ResNet10 Model:

* The ResNet10 class defines a ResNet with 10 layers (2 blocks per layer).
* It includes an initial convolution layer followed by four stages (layer1 to layer4) with varying numbers of blocks.
* Each stage consists of residual blocks and may include a downsampling operation.
* The model ends with an adaptive average pooling layer, flattening the output, and a fully connected layer to predict class labels.

Training Loop:

* The train\_model function trains the ResNet model using mixed-precision with PyTorch's autocast for reduced memory usage and increased speed.
* The model is trained for a specified number of epochs, and the loss is computed and accumulated for each batch.
* A GradScaler is used for scaling the gradients during backpropagation in mixed-precision training.
* The average training loss is printed every 10 epochs, along with the total training time.

**Results:**

**(Note Model Was Still Training with Time elapsed as 3.8 Hours – it crashed twice)**

**Epoch 1/200, Training Loss: 1.4634**

**Epoch 10/200, Training Loss: 0.3502**

**Epoch 20/200, Training Loss: 0.1839**

**Epoch 30/200, Training Loss: 0.1101**

**Epoch 40/200, Training Loss: 0.0740**

**Epoch 50/200, Training Loss: 0.0558**

**Epoch 60/200, Training Loss: 0.0440**