Smart CIFAR-10: Dynamic Weighted-Branch Convolutional Network

QMUL - ECS7026P, Neural Networks and Deep Learning

Author: Rasul Abdullayev

Date: 18.04.2025

Contact: https://github.com/Rasul1908

Abstract

This work presents a compact convolutional network for CIFAR-10 that processes inputs via parallel convolutional branches and fuses them with input-dependent softmax weights derived from per-channel means. The model is implemented in PyTorch and trained end-to-end on a single GPU (e.g., Colab T4) for 50 epochs using standard data augmentation. With Adam (lr = 0.001) and batch size 128, the network attains a best test accuracy of 88.24% at epoch 45. Code, settings, and training plots are included for reproducibility.

Keywords

CIFAR-10 · Convolutional Neural Networks · Mixture-of-Experts · Dynamic Routing · PyTorch

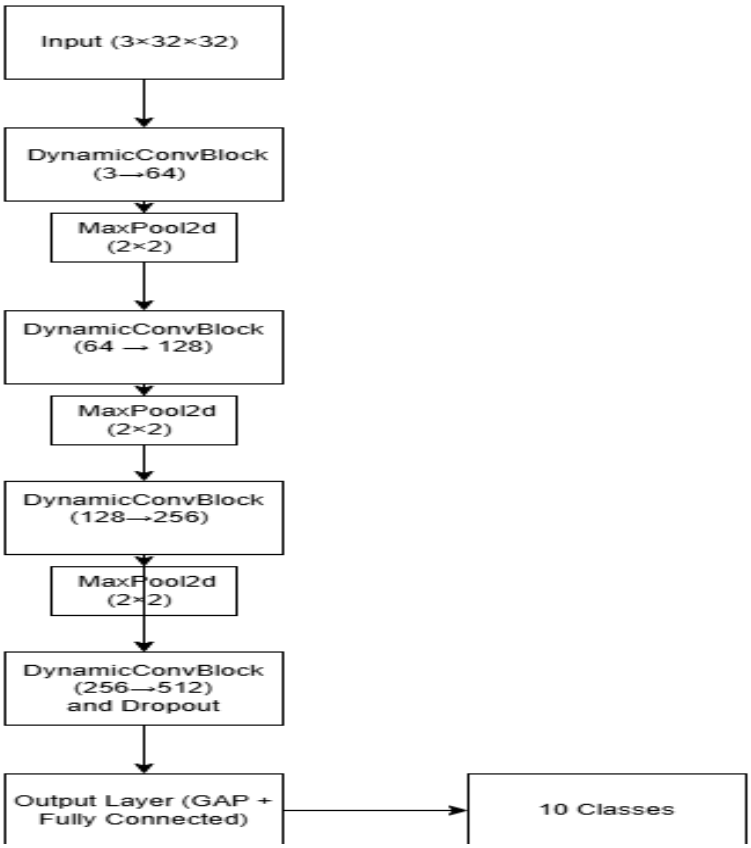
Version: v1.0

Repository: https://github.com/Rasul1908/smart-cifar10

1. Introduction

The project implements a complete construction of convolutional neural networks (CNN) that classifies CIFAR-10 images through PyTorch. The main goal involved optimizing classification results through modifications of both network structure and training procedures and performance optimization methods above the Section 2 course introduction design.

2. Model Architecture



The model design of SmartCIFAR10Net includes four sequential DynamicConvBlock modules in its modular structure. The parallel convolution branches inside each block contain learnable soft attention weights to perform dynamic feature extraction. A 2×2 MaxPooling layer reduces spatial dimensions as well as introduces translation invariance after each block. Regularization is achieved by the dropout layer together with the final Output Block that contains Global Average Pooling followed by a fully connected layer predicting 10 output classes.

3. Model Comparison

The base model includes several CNN layers with ReLU activations and integrates a fully connected output module. One difference between the base model and my proposed version lies in the depth of architecture which includes DynamicConvBlocks for soft attention-based combination of multiple convolution branches. Model applies sequential depth expansion of features from 3 to 512 channels with MaxPooling as a spatial downscaling operation in-between stages Batch normalization is applied within each block, and dropout is used for regularization. The output layer uses global average pooling instead of flattening due to the fact that it reduces overfitting. I applied data augmentation during training such as random cropping, horizontal flipping, and cutout, along with a learning rate scheduler for better convergence.

4. Training Techniques and Hyperparameters

The optimization process used Adam to train the model with an initial learning rate value 0.001.The training schedule used the step rate by 0.5 after every 10 epoch cycle. in total 50 training epochs were performed at each batch scale of 64. Cross-entropy was used to define final loss value. Data augmentation techniques such as horizontal flipping, random cropping with padding and cutout were performed to improve generalization and to prevent overfitting. Normalization has also been performed according to the best recommendations for this dataset. Dropout was included after the last convolutional block to enhance regularization. All training and evaluation were performed using CIFAR-10 dataset, with standard splits for training and testing.

5. Results and Evaluation

The model training process included 50 epochs. The training and testing accuracy was measured after each epoch and training loss was recorded per each batch. The best test results reached 88.12% while the training results exceeded 94.12% accuracy. The learning

rate decreased by 50 percent every twenty epochs to achieve stable final training stages. The two visualisations are included into the report. The first one is illustrating training loss across all batches and the second one illustrates training accuracy across epochs. The results show steady improvement throughout training, with accuracy almost not changing towards the end, which indicates effective convergence and no major overfitting.