CSCE 636: Deep Learning: Final Project

Shashank Santosh Jagtap
CSCE
Texas A&M University, College Station
shashankjagtap@tamu.edu
UIN: 534002338

1 Abstract

In this research, we introduce a deep learning model tailored for 10-class image classification, emphasizing its development and implementation. Our goal is to create a robust model that effectively sorts images into predefined categories. By utilizing a provided training dataset, the model undergoes iterative training and fine-tuning processes to enhance its accuracy. Additionally, we assess its performance using a public test dataset before final submission. The culmination involves evaluating the model's effectiveness on a private test dataset, shedding light on its ability to generalize and its potential real-world utility. Through experimentation and analysis, our study aims to advance deep learning methodologies in the realm of image classification.

2 Introduction

In recent times, deep learning has emerged as a potent tool in the realm of image classification, transforming our capacity to identify and categorize visual information. This project seeks to push the boundaries of deep learning-based image classification by delving into innovative neural network designs, data preprocessing methods, and training approaches. Our main objective is to create and deploy a tailored deep learning model that accurately categorizes images into ten distinct classes. Importantly, project guidelines necessitate the exclusion of commonly used network architectures and forbid the use of additional image data for transfer learning or pre-training, aiming to stimulate creativity and innovation while challenging conventional methods. In our endeavor, we will explore various aspects of deep learning model construction, including the integration of new operations, blocks, and modules within the network architecture. Moreover, significant attention will be devoted to investigating data preprocessing techniques, such as normalization and augmentation, to bolster the model's resilience and generalization abilities.

3 Proposed Method

To address the challenge of 10-class image classification, I propose a novel deep learning architecture termed ImprovedBlock, integrated within our custom MyNet model. The ImprovedBlock module incorporates various stages of feature extraction and refinement, employing a combination of convolutional layers and batch normalization to enhance model performance. Notably, the architecture diverges from conventional approaches by introducing adaptive stages within the block design, allowing for dynamic adjustment of feature transformations based on network depth. Through iterative stacking of these ImprovedBlock modules, our MyNet model achieves hierarchical feature extraction, culminating in a fully connected layer for classification. By leveraging this novel architecture, we aim to explore the efficacy of adaptive feature learning in image classification tasks, surpassing the limitations of traditional network architectures.

4 Implementation Details

The architecture consists of a series of residual blocks, each containing convolutional layers with batch normalization and ReLU activation functions. The 'ImprovedBlock' class represents the basic building block of MyModel, featuring residual connections to facilitate training of deep networks. The 'MyModel' class orchestrates the training, validation, and testing processes of the model. It initializes the network, defines training procedures including data loading, optimization, and evaluation, and provides methods for saving and loading trained models. 'MyModel' model is built by arranging numerous 'ImprovedBlock' modules in layers, gradually enhancing the sophistication of feature representation. It comprises three layers stacked on top of each other, with different numbers of blocks and output feature dimensions. After extracting features, global average pooling is employed to compress spatial information, preparing it for classification through a fully connected layer. The model is trained using Stochastic Gradient Descent (SGD) with momentum and employs cross-entropy loss as the optimization criterion. I opted for the SGD optimizer, believing it to be the most suitable for implementing a residual neural network. Additionally, the training process includes data augmentation techniques such as random cropping and random erasing. Moreover, I've opted to store the model checkpoint exclusively when the model attains its highest validation accuracy. This choice ensures that the saved model snapshot represents the configuration that achieves the greatest accuracy on unseen data. By exclusively retaining checkpoints linked to peak validation accuracy, unnecessary storage space is avoided, and the final saved model is tailored for superior performance on real-world datasets. Overall, the method aims to enhance classification performance on image data, particularly demonstrated here on the CIFAR-10 dataset.

4.1 Increased batch size

In the context of training neural networks on datasets like CIFAR-10, the choice of batch size plays a crucial role in shaping the training dynamics and influencing the model's performance. I used a batch size of 128 which I think offer several advantages over smaller batch sizes like 64 or 32:

1. Efficient GPU Utilization: Larger batch sizes typically result in better GPU utilization. With a batch size of 128, the GPU can process more samples in parallel, maximizing its computational throughput. This leads to faster training times and allows for more efficient utilization of expensive GPU resources.

2. Stable Gradients: Larger batch sizes provide more stable estimates of gradients since they are computed over a larger set of samples. This stability can help mitigate the effects of noisy gradients and lead to smoother convergence during training. As a result, models trained with larger batch sizes often exhibit more stable training behavior and converge to better solutions.

3. Improved Generalization: By exposing the model to a diverse range of samples within each batch, larger batch sizes can help regularize the training process and encourage the learning of more robust features, potentially leading to better generalization on unseen data.

Overall, while smaller batch sizes like 64 or 32 may offer benefits such as faster convergence to local minima and lower memory requirements, a batch size of 128 strikes a balance between efficiency, stability, and generalization performance, making it a suitable choice for training deep neural networks on datasets like CIFAR-10.

4.2 Data Augmentation

The data augmentation techniques employed in our architecture aim to enhance the generalization capability of the model by diversifying the training data. Specifically, the following augmentation techniques are implemented:

1. Random Horizontal Flip: Images are randomly flipped horizontally with a certain probability during training. This helps the model to learn features that are invariant to horizontal flips, thereby improving its robustness to variations in object orientation.

2. Random Crop: Random cropping is performed on the input images during training. This involves selecting random regions of the image and resizing them to the desired input size. By doing so, the model learns to focus on different parts of the image, promoting spatial invariance and robustness.

3. Random Erasing: This technique randomly selects rectangular regions in the input images and replaces them with zeros. It acts as a form of regularization by introducing noise and occlusions, forcing the model to learn more robust features and reducing overfitting.

These augmentation techniques help the model learn invariant features, improve its generalization ability, and reduce overfitting by exposing it to a wider range of variations in the training data. As a result, the model becomes more robust and performs better on unseen data, leading to improved overall performance and accuracy.

4.3 Improved Block

The Improved Block, as implemented in the code, is a fundamental component of the model architecture designed to address the challenges of training deep neural networks. It introduces residual connections within each block, allowing for the direct propagation of gradients during backpropagation. This mitigates the vanishing gradient problem and facilitates the training of deeper networks by enabling the network to learn residual mappings. Additionally, the Improved Block employs batch normalization and ReLU activation functions, enhancing the stability and convergence properties of the network. By providing shortcut connections and enabling the learning of residual features, the ImprovedBlock promotes efficient gradient flow, accelerates convergence, and ultimately leads to improved performance on various image classification tasks.

5 Results

5.1 Training model

The model was trained using a set of hyperparameters specified in the training configs dictionary. These hyperparameters include a learning rate of 0.1, a batch size of 128, a momentum term of 0.9 for SGD optimization, a weight decay of 2e-4 for regularization, and a maximum training epoch of 200. Additionally, a learning rate schedule was employed, where the learning rate was reduced to 0.1 at epoch 75, further reduced to 0.01 at epoch 125, and finally to 0.001 at epoch 200. During training, the model achieved a final **training loss of 0.0822**, a **validation loss of 0.4220**, and a **validation accuracy of 91.460%**. These results indicate that the model successfully learned to generalize well to unseen data, demonstrating its effectiveness in learning meaningful features from the CIFAR-10 dataset.

5.2 Testing model on test data

Test accuracy on test data of CIFAR-10 was **90.130%** and test loss was **0.3671**.

5.3 Predicting on given images

Prediction results are stored in **predictions.npy** file in which for each image, we store a vector of the probabilities for the 10 classes. This file is attached with my submission.

6 Conclusion

In conclusion, the implementation of a our model trained on the CIFAR-10 dataset using specified hyperparameters and augmentation techniques yielded promising results. With a batch size of 128 and a learning rate schedule that gradually decreased over epochs, the model achieved a final training loss of 0.0822 and demonstrated strong generalization with a validation loss of 0.4220 and a validation accuracy of 91.460%. The utilization of Improved Blocks within our architecture facilitated efficient gradient flow and enabled the training of deeper networks, contributing to the model's effectiveness. The use of data augmentation techniques, including random horizontal flips, random cropping, and random erasing, further enhanced the model's robustness and generalization capabilities. Overall, these findings underscore the effectiveness of the proposed methodology in achieving high performance on image classification tasks like CIFAR-10.