Optimizing Deep Learning Architectures with Regularization and Attention Mechanisms for High-Performance Image Classification

Raunak Choudhary[1], **Debika Dharmalingam**[2], **Yashavika Singh**[3]
[1,2,3]New York University
rc5553@nyu.edu, dd3873@nyu.edu, ys6668@nyu.edu

GitHub Repository:

https://github.com/YashavikaSingh/resnet-cifar10-classifier

Abstract

This project talks about the development and implementation of an improved ResNet architecture for image classification, specifically for the CIFAR-10 dataset. The CIFAR-10 dataset comprises 60,000 32x32 RGB images across 10 categories, with 50,000 training and 10,000 test images. The model incorporates several enhancements, including the Mish activation function, Squeeze-and-Excitation (SE) attention mechanisms, and a Transformer block, to boost performance. The training process utilized CutMix augmentation, label smoothing, and a one-cycle learning rate policyThe goal is to develop a model that can classify these images with high accuracy.

Overview

This project details performing image classification on a custom dataset by training it on CIFAR-10 using a modified Resnet architecture. The CIFAR-10 dataset is a widely used dataset benchmark in machine learning, consisting of 60,000 32x32 RGB images(3 channels) across 10 distinct classes (e.g., airplanes, automobiles, birds). The main aim was to train the model in such a way that the enhanced version of this Resnet model predicts the custom dataset well. The model architecture is based on a refined Pre-Activation ResNet, incorporating several key enhancements to improve its performance:

Base Resnet

ResNet (Residual Network) is a deep convolutional neural network that addresses the vanishing gradient problem by introducing residual connections. The core structure consists of convolutional layers, batch normalization, activation functions, and skip (residual) connections that allow gradients to flow directly through the network. A typical ResNet architecture includes multiple residual blocks, each containing two or three convolutional layers with identity mappings to enable efficient learning of deep features.

- **Advanced Activation Functions:**The Mish activation function is employed as a replacement for the conventional ReLU activation to facilitate improved gradient flow, thereby enhancing the model's training dynamics. Mish provides a smooth and non-monotonic transformation, which enables a richer representation of features and mitigates issues such as dying neurons that are prevalent in ReLU-based networks.

- **Attention Mechanisms:** To enhance the model's ability to focus on the most relevant features within an image, attention mechanisms such as Squeeze-and-Excitation (SE) blocks and Convolutional Block Attention Modules (CBAM) are integrated.These modules dynamically recalibrate feature maps by emphasizing important spatial and channel-wise features, thereby improving the model's power and discriminative ability.

- **Transformer Block:** A custom Transformer block is incorporated to capture long-range dependencies across spatial locations in the feature maps. This addition enables the model to establish complex relationships within the image, improving its ability to understand global contextual information, which is particularly beneficial for high-level vision tasks.

- **Training Regularization:**Label smoothing is used to prevent overfitting and make the model more generalizable. Instead of assigning a single correct label to each example, this method spreads some probability across other labels, preventing the model from becoming too confident and improving its performance on new data.

- **Optimized Training:** The AdamW optimizer, along with a one-cycle learning rate scheduler, is used to make training more efficient. AdamW includes weight decay to reduce overfitting, while the one-cycle learning rate policy speeds up convergence by adjusting the learning rate dynamically throughout training.

- **Data Augmentation:** To make the model more robust and adaptable, data augmentation techniques like CutMix and test-time augmentation (TTA) are applied. CutMix creates new training examples by mixing parts of different images and their labels, helping the model learn effectively. TTA further improves performance by making predictions on multiple augmented versions of an image and averaging the results for better accuracy .

Methodology

Data Loading and Preprocessing

Dataset Acquisition: The CIFAR-10 dataset is loaded from the specified directory. The training data is extracted from 'train_batch', and the validation data is loaded from 'test_batch'. A custom test dataset is also loaded from a pickle library.

Data Augmentation:

Training Set: A series of transformations are applied to the training data to increase diversity and improve the model's ability to generalize. These include:

- Random cropping with padding.
- Random horizontal flipping.
- Color jitter for brightness, contrast, and saturation.
- Normalization.
- Random erasing.

Validation/Test Set: The validation and test sets undergo only normalization. Dataset Creation: A custom 'CustomCIFAR10Dataset' class is defined using PyTorch's 'Dataset' class to handle the images and labels. This class takes images, labels, and an optional transform as input and returns a transformed image and its corresponding label.

Model Architecture

- **Initial Convolution:** The model begins with an initial convolutional layer.
- **Residual Blocks:** The core of the model consists of four stages of pre-activation residual blocks ('layer1' to 'layer4'). Each 'PreActResidualBlock' includes:
  - Pre-activation with batch normalization and Mish[1] activation.
  - Convolutional layers.
  - Optional dropout.
  - Attention mechanisms (CBAM).
  - Skip connections.

Training Process

- **Optimizer**: Upon experimenting with various optimizers such as AdamW, Stochastic Gradient Descent (SGD), and Adam, we found that the AdamW optimizer performed best for training. It is configured with the following parameters:
  - Learning rate: 1×10^{-7}
  - Weight decay: 0.005
  - Epsilon: 1×10^{-8}
- **Learning Rate Scheduler**: A one-cycle learning rate scheduler is employed. The scheduler begins with an initial learning rate, increases it to a maximum, and then gradually decreases it. The parameters include:
  - **max_lr**: 0.002
  - **epochs**: Number of training epochs (400)
  - **steps_per_epoch**: Number of training steps per epoch (length of train_loader)
  - **pct_start**: Percentage of training steps to reach max_lr (0.1)
- **Loss Function**: Label smoothing loss is used, with a smoothing factor set to 0.02.
- **Data Augmentation**: CutMix augmentation is applied during training with a probability of 0.25.
- **Training Loop**: The model is trained for a specified number of epochs, iterating over the dataset in each epoch. At the beginning of each epoch, the model is set to training mode (`model.train()`). The training data is processed using the `train_loader`, which feeds mini-batches to the model. CutMix augmentation is applied with a predefined probability, modifying both the input data and corresponding labels to enhance generalization. For each batch:
  - The optimizer's gradients are reset using `optimizer.zero_grad()`.
  - Forward propagation is performed, where the model generates predictions.
  - The loss is computed using label smoothing, which prevents the model from becoming overconfident.
  - Backpropagation is performed (`loss.backward()`) to compute gradients.
  - The optimizer updates the model's parameters (`optimizer.step()`).
  - Throughout training, loss and accuracy metrics are recorded, and the learning rate is dynamically adjusted using a one-cycle learning rate scheduler (`scheduler.step()`).
- **Validation and Model Saving**: After each epoch, the model is switched to evaluation mode (`model.eval()`) to assess its performance on the validation dataset. The `val_loader` iterates over validation samples, computing loss and accuracy to monitor generalization. These performance metrics are stored in a history dictionary for later analysis. If the current validation accuracy surpasses previous records, the best model weights are updated accordingly. The model is periodically saved to preserve progress during training.
- **Early Stopping**: The code includes an early stopping mechanism to prevent overfitting. However, to achieve better accuracy, it is not used.

Evaluation and Prediction

- **Evaluation Metrics:** The model's performance is evaluated using training and validation loss and accuracy.
- **Test-Time Augmentation (TTA):** Test-time augmentation is implemented to improve the robustness of predictions. This involves applying multiple transformations to each test image, generating predictions for each, and then averaging the predictions. The transformations used include:
  - Horizontal flipping.
  - Color jitter.

  - Random affine transformations.

- **Prediction Generation:**
  - The trained model is used to generate predictions on a custom test dataset ('cifar_test_nolabel.pkl').
  - The custom test dataset is loaded and preprocessed.
  - Predictions are generated for each batch of test images.
  - The predictions are collected and converted to a NumPy array.
- **Submission File Creation:** A Pandas DataFrame is created with the predictions, and a submission file is saved.

Utilities and Helper Functions

- **'set_seed'**: This function sets the random seed for reproducibility across 'random', NumPy, and PyTorch.
- **'load_cifar_batch'**: This function loads a CIFAR-10 batch from a pickle file.
- **'CustomCIFAR10Dataset'**: A PyTorch Dataset class to handle CIFAR-10 data with transformations.
- **'Mish':** An activation function.
- **'SEBlock':** Squeeze-and-Excitation attention block.
- **'CBAM':** Convolutional Block Attention Module.

Design Considerations and Lessons Learnt

- **More Epochs, Better Results**: We observed that for the same model, more epochs lead to better performance. In some cases, with a learning rate of 1×10^{-7}, there was a slight increase in accuracy without early stopping.
- **Mish Activation Function**: The code uses the Mish activation function, which generally performs better than ReLU for CNNs.
- **Attention Mechanisms**: The code incorporates Squeeze-and-Excitation (SE) blocks and Convolutional Block Attention Modules (CBAM) for more sophisticated attention.
- **Transformer Block**: A Transformer block is added to the architecture for enhanced feature learning.
- **Pre-activation Residual Blocks**: The code defines Pre-activation Residual Blocks with options for attention mechanisms.
- **Improved ResNet Architecture**: The code implements an improved ResNet architecture with features like Mish activation, attention blocks, and a Transformer block.
- **Parameter Count Management**: The code includes a check to ensure the model's parameter count stays within a specified limit (5 million).
- **Label Smoothing**: Label smoothing is used with an optimized smoothing factor to encourage more confident predictions.
- **CutMix Augmentation**: The CutMix augmentation technique is implemented to improve generalization.
- **Stochastic Weight Averaging (SWA)**: SWA is used for better generalization in later epochs.
- **AdamW Optimizer**: We experimented with Stochastic Gradient Descent and Adam optimizers, but observed that AdamW gave better performance.
- **Test-Time Augmentation (TTA)**: A function for test-time augmentation is included to improve predictions.

Visualization and Analysis

- **Training History Visualization:** The training and validation loss and accuracy are plotted over the epochs to visualize the training process. Looking at the four graphs together tells a great story about our approach.
  - **Loss Curves:** Training and validation loss decreased rapidly initially and continued to decline steadily. The slight dip of validation loss below training loss suggests effective regularization through CutMix and dropout.
  - **Accuracy Curves:** Both accuracies surpassed 90%, with validation accuracy consistently higher, indicating strong generalization. Improvements diminished after epoch 150 but continued marginally.
  - **Learning Rate Schedule:** The OneCycleLR strategy enabled rapid initial convergence, peaking at 0.001 before gradually decreasing via a cosine schedule, refining model parameters effectively.
  - **Per-Class Validation Accuracy:** Most classes exceeded 95% accuracy, with ships and automobiles performing best (97%). Cats had the lowest accuracy (89.3%) due to greater pose and appearance variability.

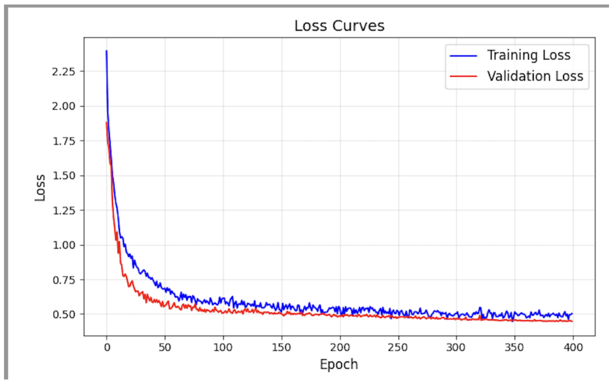

Figure 1: Loss curves

Model Summary

Model Architecture: ImprovedPreActResNet with Mish activation and CBAM attention
Total Parameters: 4,969,600
Best Validation Accuracy: 95.60%
Best Training Accuracy: 94.54%
Training Configuration:

- Epochs: 200

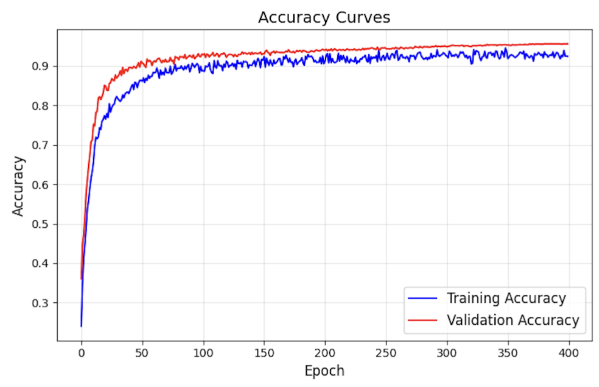

Figure 2: Accuracy curves

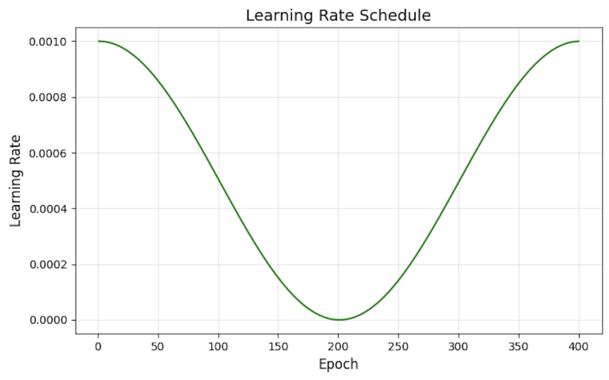

Figure 3: Learning Rate schedule

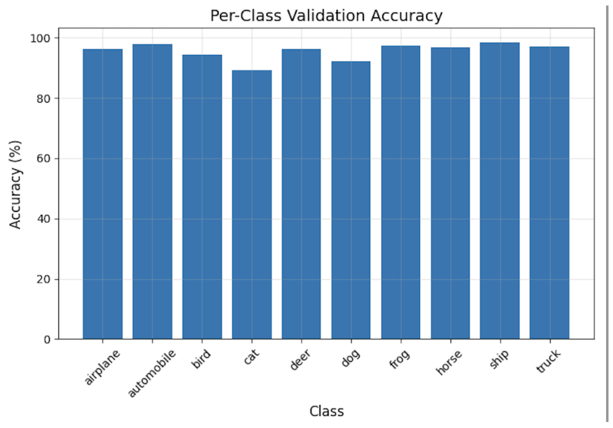

Figure 4: Per class validation accuracy

- Optimizer: AdamW with weight decay 0.005
- Learning Rate Schedule: OneCycleLR (max_lr=0.002)
- Batch Size: 128
- Loss Function: Label Smoothing (smoothing=0.05)
- Data Augmentation: RandomCrop, RandomHorizontalFlip, ColorJitter, RandomErasing
- Advanced Augmentation: CutMix (probability=0.25)

Model Highlights:

- Pre-activation residual blocks
- Mish activation function
- CBAM attention mechanism
- Strategic dropout (rate=0.05)
- Channel progression: 30→60→120→240
- Residual block structure: [2, 2, 5, 3]

Per-Class Validation Accuracy:

- Airplane: 96.20%
- Automobile: 97.90%
- Bird: 94.40%
- Cat: 89.30%
- Deer: 96.30%
- Dog: 92.20%
- Frog: 97.40%
- Horse: 96.90%
- Ship: 98.40%

Activation Function Impact

Mish activation consistently outperformed ReLU and LeakyReLU in our experiments, providing better gradient flow and faster convergence.

Data Augmentation Balance

Aggressive data augmentation initially led to underfitting. We found that more moderate augmentation with strategic techniques like CutMix produced better results.

Learning Rate Dynamics

The One-Cycle learning rate schedule was critical for efficiency, reducing training time while improving final accuracy compared to constant or step-based schedules.

In the Testing on Custom Test Dataset phase, we evaluated our model on 10,000 unlabeled competition images. The predictions (e.g., classes 6, 1, 8, 6, 9) showed a diverse mix, indicating no bias. We saved the trained model as a .pth file for future use. While actual test accuracy will be known post-competition, strong validation performance suggests promising results.

Conclusion

The code implements an enhanced ResNet with Mish activation, attention mechanisms, and CutMix augmentation. The model is trained, validated, and saved for future use. Predictions on a custom test dataset showed lower probability for dogs (81%). Future improvements could focus on data augmentation for dog images to enhance performance.

References

[1] Mondal2022Ayan Mondal and Vimal K. Shrivastava. A novel Parametric Flatten-p Mish activation function based deep CNN model for brain tumor classification. *Computers in Biology and Medicine*, 150:106183, 2022. https://doi.org/10.1016/j.compbiomed.2022.106183.