COL774 - Machine Learning

Assignment-3 : Enhancing Accuracy in High-Confidence Predictions

Submission-1

Yash Bansal(2022CS51133)

Satwik(2022CS51150)

Aneeket Yadav(2022CS11116)

Introduction

- This report presents the design choices and optimisations implemented for enhancing the accuracy of high confidence predictions on the CIFAR100 dataset.
- Since the parameter space of the model and the optimisation techniques is quite large, we were only able to arrive at crudely approximate optimal values of most parameters in the best interest of time. Likewise, instead of plotting graphs in each case, it was more convenient to eyeball the rate of change in the train accuracy and train loss.

Choice of Architecture

- We tried multiple CNN architectures such as ResNet, WideResNet, StochasticDepthResnet, EfficientNet and ConvNext(small and base). Since we were not allowed to use pretrained weights, the time taken to reach a decent test accuracy. This was computed by directly importing the CIFAR100 dataset rather than reading the .pkl files, since the latter did not contain correct labels and we found it pertinent that the model is first able to classify images with a high test accuracy before taking on the more challenging and specific task of improving its performance for high confidence predictions.
- ResNet and WideResNet did not perform sufficiently well and we were only able to achieve the test accuracy of ___.
- In the absence of pretrained weights, state of the art models like ConvNext converged at about 45% accuracy which was quite unsatisfactory.

- StochasticDepthResNet's implementation was available in multiple sizes. Since each CIFAR100 image id only 32x32, we used the smallest model – StochasticDepthResnet18.
- EfficientNet performed at par with StochasticDepthResNet but took much longer to converge.

Choice of Optimiser

- AdamW optimiser was used with parameters - lr=0.001, weight_decay=0.05.
- Adam tends to have less effective regularization when combined with adaptive learning rates. AdamW corrects this by separating weight decay from the gradient update, leading to more consistent weight decay and better generalization. For modern deep learning models, AdamW is typically a better choice.

Choice of Loss Function:

- We have designed a custom loss function which is a weighted combination of Label Smoothing Cross Entropy Loss, Focal Loss and a custom loss function which punishes low accuracy outputs. These combine to form our Confidence Aware Combo Loss function. Label Smoothing Cross entropy loss is using its default epsilon value of 0.1. Focal Loss uses default values alpha=1 and gamma=2. To combine the the two loss functions, we use weights __ and __ respectively. Further, we have chosen the confidence penalty to be 0.1 as that was found to give better results than 0.2, although we could not experiment much with it.

Choice of Scheduler

- Use of a scheduler leads to improved convergence, stability during training, faster training and better generalisation. We considered two schedulers – Reduce LR On Plateau and Cosine Annealing LR. Although both provided similar test accuracy, Cosine Annealing converged faster and was therefore chosen.

Use of Scaler

- Scalers are helpful in allowing mixed precision training, preventing underflow of small gradients and maximising GPU effciency by reducing memory consumption. GradScaler is a pretty standard scaler implementation.

Image Augmentation Methods

- Techniques like **RandomCrop** with padding and **RandomHorizontalFlip** introduce variations in object positioning and orientation, making the model more robust to spatial transformations.

- **AutoAugment**, based on the CIFAR-10 policy, automatically selects a set of augmentations that improve performance, adding diversity to the training data by applying transformations like color jittering and sharpness adjustments.
- **Normalization** standardizes the pixel values, improving convergence during training, while **RandomErasing** introduces random occlusions in parts of the image, encouraging the model to focus on discriminative features rather than overfitting to specific areas.

Batch Size

We tried batch sizes 64, 128 and 256, Out of these, 128 yielded the best results and was thus chosen.