Home task – CIFAR10 – BAR MADAR

1. Train resnet18 model and show accuracy and training plots.
2. How many parameters are there in the model?
3. Change Relu activations in model to Relu6 and show accuracy of new model vs. old one

DATA

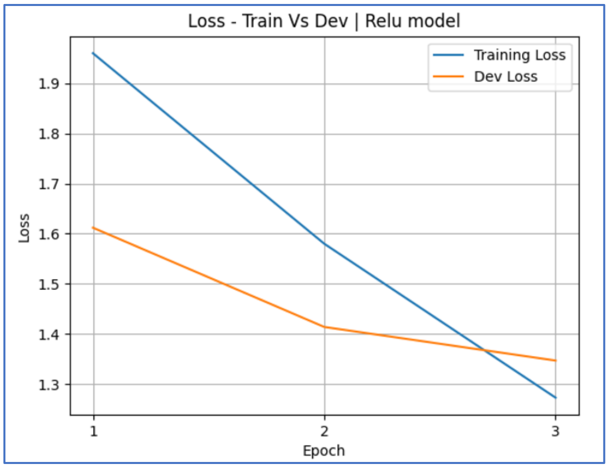
- I divided the CIFAR10 train data for 2 subsets:
  - Train – consists 90% of the data (45,000 photos)
    - Using to train the model
  - Dev – consists 10% of the data (5,000 photos)
    - Using for test the model before the evaluation
- CIFAR10 test set (10,000 photos) – to evaluate the models

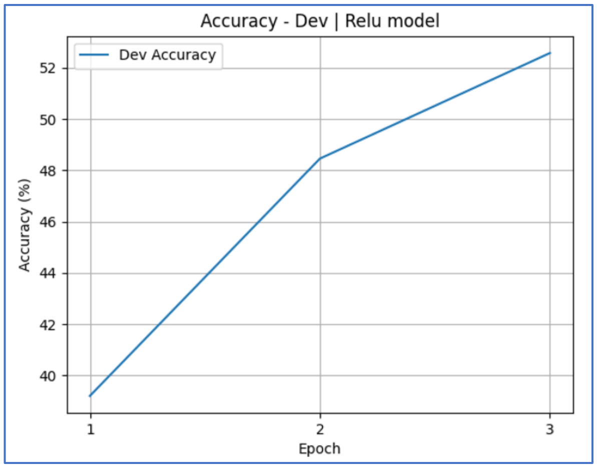
Training process

- Resnet18 architecture + cross entropy loss
- Augmentation
  - Randomcrop(padding=4)
  - RandomHorizontalFlip(0.5)
- Epochs – 3 (2.5 hours per epoch)
- Total learnable parameters – 11,173962 (~11M)

Relu model

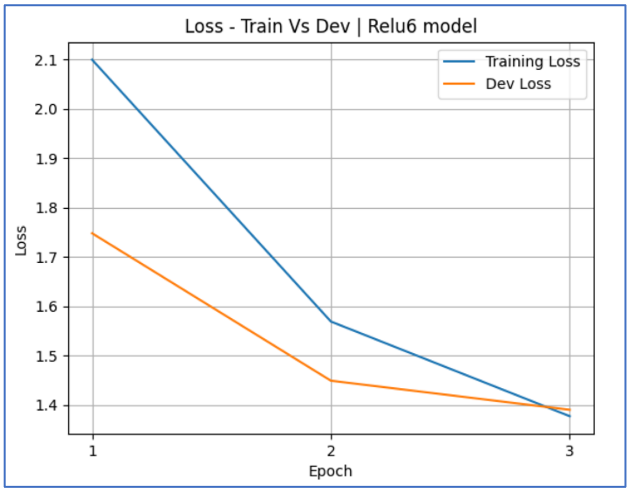
- Accuracy – **53.00%** (on test set)

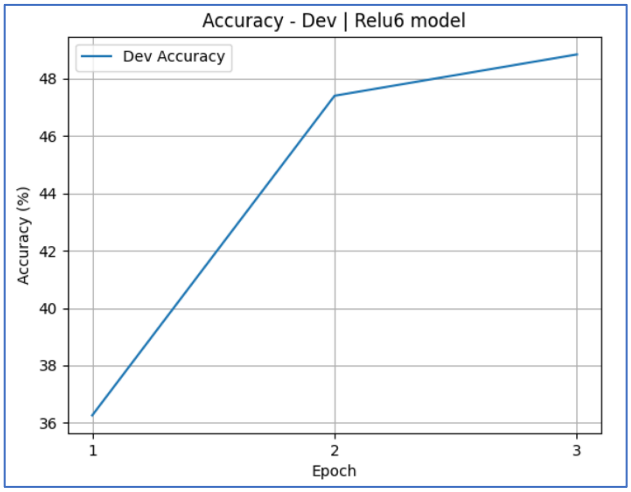




Relu6 model

- Accuracy – **49.68%** (on test set)





4. Export the model to onnx

- The models are exported to onnx using *“torch.onnx.export()”* and can be found on the git repository
    - “v1.onnx” – Relu model
    - “v2.onnx” – Relu6 model

5. Do post training quantization (implemented in pytorch) for both relu and relu6 models. Show new model size and accuracy of new vs old models.

- Static post training quantization is done for 2 of the already trained models using Pytorch
    - Loading the learned parameters into the resnet18 model
    - Fusing the “bn” layers to the “conv” layers using *“torch.quantization.fuse_modules”*
    - Prepare the model to the calibration step using *“torch.quantization.prepare”*
    - Calibrate the prepared model to determine quantization parameters
        - Dev set is used for the calibration process(by feed-forwarding)
    - Converting the model to a quantized model using *“torch.quantization.convert”*
- The quantized models are store in “checkpoint” folder
    - Ckpt_v1_quantize.pth – Relu model
    - Ckpt_v2_quantize.pth – Relu6 model
- Size comparison
    - Regular model – 44.8MB
    - Quantized model – 11.4MB
- Evaluation - <u>Not implemented</u>
    - Need to quantize the tensor data before it passed to a quantized kernel
        - Using “QuanStub()” module.
    - Then de-quantize the output of the fused quantized layers
        - Using “DeQuanStub()” module.
    - After the model modification – evaluate it using the Dev data

6. Try to cut number of parameters in half (from what you got in 2) while keeping as high accuracy as you can.

- Modified the architecture to a smaller one
- Pruning the most close to zero weights
    - Filter pruning
        - For example – running a mean average of each filter and remove the K% of filters that have the smallest mean
    - Dense layers pruning
        - Remove the K% smallest weights of the dense layer connected to the last convolution layer (after flatten)
- Knowledge distillation (https://arxiv.org/pdf/1503.02531.pdf)
    - Teaching a smaller network(student) from an already trained larger network(teacher)
    - Train the teacher network separately using the complete data set
    - Train the student net using a linear combination of student-loss and distillation-loss
        - Student-loss: the loss between the net prediction and the one-hot encoder target vector
        - distillation-loss: the difference between the soft student net prediction and the soft teacher net labels