CSCE 636: PROJECT

Dev Garg
Student, Texas A&M
dev.garg@tamu.edu

Abstract

Proposed a convolutional neural network to perform 10-class image classification on Cifar-10 dataset consisting of 50000 32x32 color images. Data pre-processing, normalizations, training strategies, optimizers, parameter initializations and regularizations techniques are explored to achieve a validation accuracy of 93.20%.

1 Model and Methods

The proposed model is based on the architecture described in [1], which draws inspiration from VGG [2]'s use of 3x3 convolutions and 2x2 max-pooling layers but uses only 1.97M parameters. The initial layer is an uncommon 2x2 convolution layer with no padding, which allows the internal feature maps to be 31x31 → 15x15 → 7x7 → 3x3 allowing for better throughput. The non-linear activations used are GELU [3] and along with BatchNorm[4]. Biases are disabled for all layers except the first one. The last max-pooling layer uses kernel of size 3x3, which is then feeds into the final linear layer. The output of the final layer is scaled down by a factor of 1/9. Few methods are used to improve the performance of the model.

Figure 1: Model Architecture

1.1 Parameter Initialization:

The first layer is initialized using patch-whitening transformation, which uses the eigenvectors of the covariance matrix of 2x2 patches across the training distribution and kept frozen. The remaining convolution layers employ Dirac weight parameterization [5], allowing for similar performance without the need for skip connections as in ResNets [6]. Additionally, the affine scale parameters of the BatchNorm layers are disabled.

1.2 Image Pre-processing

The image preprocessing techniques include normalization, random cropping after translation with reflective padding of 4 pixels on each side, and random horizontal flipping. Normalization is experimented with both along the channel and along the length of each image.

1.3 Label Smoothing

For better regularization, label smoothing [7] is also applied.

3 Experiments

The objective of the experiments is to maximize the validation set accuracy on the CIFAR-10 dataset. Each training run comprises 200 epochs, with the learning rate initialized at 0.08 for SGD and 0.001 for Adam, reducing by a factor of 10 every 30 epochs. The weight decay is set to 0.02, and the momentum is 0.9. The optimizers explored include AdamW, SGD, and SGD with Nesterov momentum. The batch size is either 1024 or 128. Separate runs are conducted to analyze the impact of patch-whitening initialization and label smoothing. The preprocessing techniques involve normalizing the images either along channels or length, with padding being done in constant or reflect mode. The loss function used is the Cross-Entropy Loss.

Table 1: Experiments with different hyperparameters and training methods

#	Batch Size	Whitening init	Padding mode	Optimizer	Normalization	Label smoothing	Validation Accuracy %
1	1024	No	constant	AdamW	length	No	87.70
2	1024	No	constant	SGD	length	No	90.73
3	128	No	constant	SGD	length	No	90.72
4	128	Yes	constant	SGD	length	No	91.09
5	128	Yes	Reflect	SGD-nesterov	length	No	91.43
6	128	Yes	Reflect	SGD-nesterov	channel	No	92.03
7	128	Yes	Reflect	SGD-nesterov	channel	Yes	**93.20**

1.3 Results

The AdamW optimizer (#1) converges early compared to SGD (#2) but achieves lower validation accuracy, with a difference of approximately 3%. The performance of the setup with batch sizes of 128 and 1024 is similar. The use of patch-whitening initialization and reflect padding improves the performance compared to constant padding. Normalizing the data across channels yields better results than normalizing along the length. However, the most significant improvement in accuracy is observed when label smoothing with a factor of 0.2 is applied to the loss function. The testing accuracy (for public available dataset) obtained is 92.47%.

Additionally, it is noted that with a batch size of 128 (#7), the average training time per epoch is 13.5 seconds when using an NVIDIA A100 GPU on TAMU's HPRC-Grace cluster. However, when using pre-processed images, the same training time per epoch is reduced to 2.8 seconds, resulting in an 80% improvement. However, this approach leads to a 4% drop in validation accuracy, potentially due to the lack of random augmentation for each epoch.

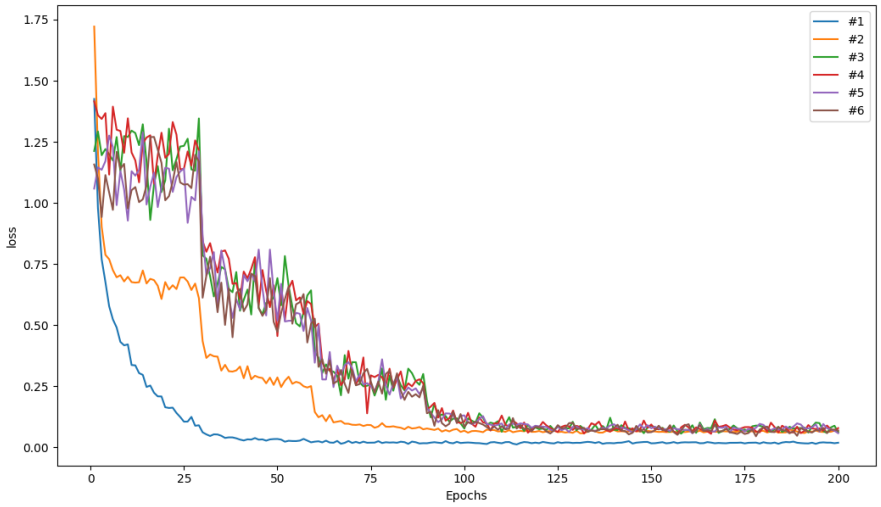

Figure 2: Training Loss Curve plots (#1-#6)
(*#7 cannot be represented in this plot as torch implementation expects sum of loss while using label smoothing.)

References

[1] Keller Jordan. 2024. 94% on CIFAR-10 in 3.29 Seconds on a Single GPU

[2] Karen Simonyan, Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition

[3] Dan Hendrycks, Kevin Gimpel. 2023. Gaussian Error Linear Units (GELUs)

[4] Sergey Ioffe, Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

[5] Sergey Zagoruyko, Nikos Komodakis. 2018. Diracnets: Training very deep neural networks without skip-connections

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. 2015. Deep Residual Learning for Image Recognition

[7] Rafael Müller, Simon Kornblith, Geoffrey Hinton. 2020. When Does Label Smoothing Help?

[8] https://www.cs.toronto.edu/~kriz/cifar.html

A Appendix

A.1 Model Structure

```
MyNetwork(
  (net): Sequential(
    (0): Conv(3, 24, kernel_size=(2, 2), stride=(1, 1))
    (1): GELU(approximate='none')
    (2): Block(
      (conv1): Conv(24, 64, kernel_size=(3, 3), stride=(1, 1), padding=same, bias=False)
      (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (norm1): BatchNorm(64, eps=1e-12, momentum=0.4, affine=True, track_running_stats=True)
      (conv2): Conv(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=same, bias=False)
      (norm2): BatchNorm(64, eps=1e-12, momentum=0.4, affine=True, track_running_stats=True)
      (activ): GELU(approximate='none')
    )
    (3): Block(
      (conv1): Conv(64, 256, kernel_size=(3, 3), stride=(1, 1), padding=same, bias=False)
      (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (norm1): BatchNorm(256, eps=1e-12, momentum=0.4, affine=True, track_running_stats=True)
      (conv2): Conv(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=same, bias=False)
      (norm2): BatchNorm(256, eps=1e-12, momentum=0.4, affine=True, track_running_stats=True)
      (activ): GELU(approximate='none')
    )
    (4): Block(
      (conv1): Conv(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=same, bias=False)
      (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (norm1): BatchNorm(256, eps=1e-12, momentum=0.4, affine=True, track_running_stats=True)
      (conv2): Conv(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=same, bias=False)
      (norm2): BatchNorm(256, eps=1e-12, momentum=0.4, affine=True, track_running_stats=True)
      (activ): GELU(approximate='none')
    )
    (5): MaxPool2d(kernel_size=3, stride=3, padding=0, dilation=1, ceil_mode=False)
    (6): Flatten()
    (7): Linear(in_features=256, out_features=10, bias=False)
    (8): Scale()
  )
)
```

Number of parameters: 1972792

A.2 Setup environment and dependencies

TAMU HPRC Grace2 cluster
Python version 3.9.6

GCCcore 11.2.0
PyTorch 2.2.2

A.3 Experiment Logs

```
[dev.garg@grace2 starter_code]$ tail -f out.10236682
Epoch 194 Loss 112.425575 Duration 13.458 seconds.
Epoch 195 Loss 113.144409 Duration 13.694 seconds.
Epoch 196 Loss 112.389168 Duration 13.572 seconds.
Epoch 197 Loss 112.332848 Duration 13.559 seconds.
Epoch 198 Loss 112.565308 Duration 13.524 seconds.
Epoch 199 Loss 112.305283 Duration 13.639 seconds.
Epoch 200 Loss 111.999878 Duration 13.661 seconds.
Checkpoint has been created.
### Validation ###
Valid accuracy: 0.9320 Loss 10042.439453
```

Extracted Logs for #7 from logs/out.10236682
(*Completed logs for all runs can be found in '/logs/' directory and saved model for #7 can be found in '/saved_models' directory)