Introduction to Machine Learning Final Report

Kyle Keeton[1] and Syed Rizvi[2]

[1]ID: KHK210001 , email: KHK210001@utdallas.edu , Department: Computer Science
[2]ID: SAR180000 , email: SAR180000@utdallas.edu , Department: Computer Science

CONTENTS

I. ABSTRACT

Compared to many image recognition models for colored images, LeNet does not require a high amount of computational resources. As a result, it is largely beneficial for use in educational settings over algorithms such as AlexNet to allow students to program and run neural networks. This paper seeks to modify the Lenet algorithm to work with 3 input layers and successfully classify image datasets with color images. In this experiment, we programmed the original LeNet and tested it against MNIST to prove the algorithm could work successfully. Next, we ran benchmarks against AlexNet on the CIFAR10 and CIFAR100 datasets to determine how the modified LeNet compares to a proven algorithm.

II. INTRODUCTION

In this set of experiments, we decided to explore how we can modify LeNet to handle RGB images. As students without access to a powerful server, we have found ourselves to be quite limited in the neural networks we can create because we do not have the computational resources to program certain algorithms. While the overall accuracy of an algorithm is important to those who want to learn more about machine learning, it is irrelevant when they are unable to run the algorithm in question.

In this experiment, we decided to explore modifying the LeNet algorithm for color images. While LeNet met our requirement of being computationally cheap, it was unable to process color images. However, it is vital for students to be able to explore algorithms with multiple input layers. When deciding what algorithm to test our network against, we tested multiple algorithms to find the most computationally expensive convolutional neural network we could run on our hardware within a reasonable time period. After attempting to benchmark VGGNet and ResNet, we settled on AlexNet.

III. RELATED WORK

We explored LeNet [2], VGG [3], AlexNet [1], and ResNet [4] on different image classifaction datasets.

While LeNet [2] works well in classifying the MNIST dataset, it is unable to handle RGB datasets at all without some extensive preprocessing.

VGG [3] and ResNet [4] were prohibitively computationally expensive to reasonably run and test on our hardware.

AlexNet [1] can successfully handle RGB image datasets. However, it struggles when the datasets have few categories and is computationally expensive.

Demonstrating to students a method of performing small modifications on an existing neural network to create new functionality is vital to help them better understand the concept and expand on future research. Additionally, creating a network that can more easily be run allows students to better explore the field of Machine Learning to eventually contribute more to research in the field.

IV. PROPOSED APPROACH

In this set of experiments, we used the following neural nets with the specified parameters:

```
LeNet [6]:{
  { cnn layers}: Sequential{
    (0): Conv2d(1, 6, kernel_size = 5, stride=1, padding = 0)
    (1): BatchNorm2d(6)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2)
    (4): Conv2d(6, 16, kernel_size = 5, stride=1, padding = 0)
    (5): BatchNorm2d(16)
    (6): ReLU()
    (7): MaxPool2d(kernel_size=2, stride=2)
  }
  (Linear Layers): Sequential{
    (0): Linear(in_features=400, out_feature=120)
    (1): ReLU(inplace=True)
    (2): Linear(in_features=120, out_features=84)
    (3): F.ReLU()
    (4): F.Linear(in_features=84, out_features=num_classes)
  }
}

LeNet_RGB:{
  {cnn layers}: Sequential{
    (0): Conv2d(3, 13, kernel_size = 5, stride=1, padding = 0)
```

```
      (1): BatchNorm2d(13)
      (2): ReLU()
      (3): MaxPool2d(kernel_size=2, stride=2)
      (4): Conv2d(13, 60, kernel_size = 5, stride=1, padding =
0),
      (5): BatchNorm2d(60),
      (6): ReLU(),
      (7): MaxPool2d(kernel_size=2, stride=2)
   }
   (Linear Layers): Sequential{
      (0): Linear(in_features=1500, out_feature=360)
      (1): ReLU(inplace=True)
      (2): Linear(in_features=360, out_features=256)
      (3): F.ReLU()
      (4): F.Linear(in_features=256,
out_features=num_classes) }
}
```

```
AlexNet [5]: {
   (features): Sequential{
      (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4),
padding=(2, 2))
      (1): ReLU(inplace=True)
      (2): MaxPool2d(kernel_size=3, stride=2, padding=0, di-
lation=1, ceil_mode=False)
      (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2))
      (4): ReLU(inplace=True)
      (5): MaxPool2d(kernel_size=3, stride=2, padding=0, di-
lation=1, ceil_mode=False)
      (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (7): ReLU(inplace=True)
      (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (9): ReLU(inplace=True)
      (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
      (11): ReLU(inplace=True)
      (12): MaxPool2d(kernel_size=3, stride=2, padding=0, di-
lation=1, ceil_mode=False)
   }
   (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))
   (classifier): Sequential{
      (0): Dropout(p=0.5, inplace=False)
      (1): Linear(in_features=9216, out_features=4096,
bias=True)
      (2): ReLU(inplace=True)
      (3): Dropout(p=0.5, inplace=False)
      (4): Linear(in_features=4096, out_features=4096,
bias=True)
      (5): ReLU(inplace=True)
      (6): Linear(in_features=4096, out_features=num_classes,
bias=True)
   }
}
```

In this set of experiments, we decided to use LeNet, AlexNet, and a modified LeNet algorithm, referred to as LeNet_RGB. LeNet_RGB was reached by modifying all parameters in LeNet by a factor of 3. Next, we fine tuned various parameters to reach better results with the CIFAR-10 dataset. Preliminary testing showed this to outperform only changing the input of the first convolutional layer.

We evaluate the performance of both Adam and SGD optimization algorithms across a range of learning rates. After assessing various combinations, we identify the most promising ones. For these selected combinations, we conduct training for 20 epochs each.

We found that AlexNet performed significantly better over a shorter number of epochs compared to the modified LeNet algorithm that we created. However, each epoch for AlexNet took approximately (INSERT NUMBER) times as long compared to the LeNet algorithm. Both algorithms converged at a similar number of epochs.

V. Experiments

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[2] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[5] S. Gross, "Alexnet.py." https://github.com/pytorch/vision/blob/main/torchvision/models/alexnet.py, 2017.

[6] Nouman, "Writing lenet5 from scratch in pytorch," 2022.

[7] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, *Dive into Deep Learning*. Cambridge University Press, 2023. https://D2L.ai.

[1] [2] [3] [4] [5] [6] [7]