Implementing and Evaluating Normalization Techniques in CNNs

Bhuvana Nagaraj
Northern Arizona University
bn522@nau.edu

Abstract

This report explores the implementation of three normalization techniques—Batch Normalization (BN), Layer Normalization (LN), and Weight Normalization (WN)—in convolutional neural networks using TensorFlow 2. We compare their effectiveness on the Fashion MNIST dataset in terms of classification accuracy and consistency with TensorFlow's built-in implementations. Our experiments show that Layer Normalization yields the highest test accuracy, while our custom BN is nearly identical to the built-in version.

1 Introduction

Normalization techniques are essential for stabilizing and accelerating the training of deep neural networks. This report presents a hands-on implementation of BN, LN, and WN from scratch using basic TensorFlow operations. The objective is to compare their performances and verify correctness against TensorFlow's own layers.

2 Methodology

2.1 Dataset

Fashion MNIST, a benchmark dataset for image classification, was used. The training set has 60,000 grayscale images and the test set has 10,000 images, each of size 28×28.

2.2 Model Architecture

We used a simple CNN consisting of:

- A 2D convolution layer with 32 filters of size 3×3
- Normalization (BN / LN / WN / None)
- ReLU activation
- MaxPooling (2×2)
- Flatten layer
- Dense output layer (10 units for classification)

2.3 Normalization Implementations

BatchNorm: Computes mean/variance across batch dimensions, normalizes input, and scales/shifts using learnable γ and β.

NeurIPS 2023 Generative AI and Biology Workshop.

LayerNorm: Normalizes across the feature dimension of each individual sample.

WeightNorm: Reparameterizes the weight as $\mathbf{w} = (g/\|\mathbf{v}\|)\mathbf{v}$, where $\mathbf{v}$ is the direction and g is a learnable scalar.

3 Experiments

3.1 Training Details

Each model was trained for 2 epochs using the Adam optimizer and Sparse Categorical Crossentropy loss. A batch size of 100 was used.

3.2 Accuracy Results

Normalization Method	Test Accuracy
No Normalization	87.29%
Batch Normalization	89.36%
Layer Normalization	89.52%
Weight Normalization	88.23%

Table 1: Test accuracy for each normalization method on Fashion MNIST.

3.3 Custom vs TensorFlow BatchNorm

To evaluate correctness, we compared our custom BN against `tf.keras.layers.BatchNormalization`. The mean absolute difference between outputs was:

```
Mean absolute difference:  0.000031
```

This confirms the functional equivalence of our BN implementation.

4 Discussion

Layer Normalization performed the best in our setting, likely due to its sample-wise normalization which avoids the instability from batch size variance. BN also worked well and matched the built-in implementation. WN improved convergence stability but slightly underperformed compared to LN and BN.

5 Conclusion

This assignment validated the importance of normalization in CNNs. Our custom implementations of BN, LN, and WN were successfully integrated, trained, and compared. LN achieved the best accuracy on Fashion MNIST, and BN showed high fidelity with TensorFlow's standard implementation.

References

- Ioffe, S., and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*
- Lei Ba, J., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv:1607.06450*
- Salimans, T., and Kingma, D. P. (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *NeurIPS*