Deep Learning Systems: CNN Image Classification on CIFAR-10

Tim Wilcoxson

February 2026

Project 4 -- Deep Learning Systems

Dataset: CIFAR-10 (Krizhevsky, 2009)

1. Report Overview

This report presents a deep learning experiment comparing two convolutional neural network (CNN) architectures on the CIFAR-10 image classification benchmark. A baseline CNN is compared against an identical architecture augmented with Batch Normalization (Ioffe & Szegedy, 2015) to isolate the effect of this single architectural modification on convergence speed, training stability, and classification accuracy. Both models are implemented in PyTorch (Paszke et al., 2019) and trained on Apple Silicon MPS hardware.

2. Dataset and Task Description

CIFAR-10 (Krizhevsky, 2009) is a widely-used benchmark dataset consisting of 60,000 32x32 color images evenly distributed across 10 mutually exclusive classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The dataset is split into 50,000 training images and 10,000 test images, with exactly 6,000 images per class in the full dataset.

For this experiment, the 50,000 training images were further split into 45,000 for training and 5,000 for validation using a fixed random seed for reproducibility. The validation set uses test-time transforms (normalization only) to provide an unbiased estimate of generalization performance during training. The test set of 10,000 images was held out entirely until final evaluation.

The task is multi-class image classification: given a 32x32 RGB image, predict which of the 10 classes it belongs to. Because the classes are perfectly balanced, accuracy serves as a valid primary evaluation metric. Data augmentation (random horizontal flips and random crops with padding) was applied during training to improve generalization (Goodfellow et al., 2016).

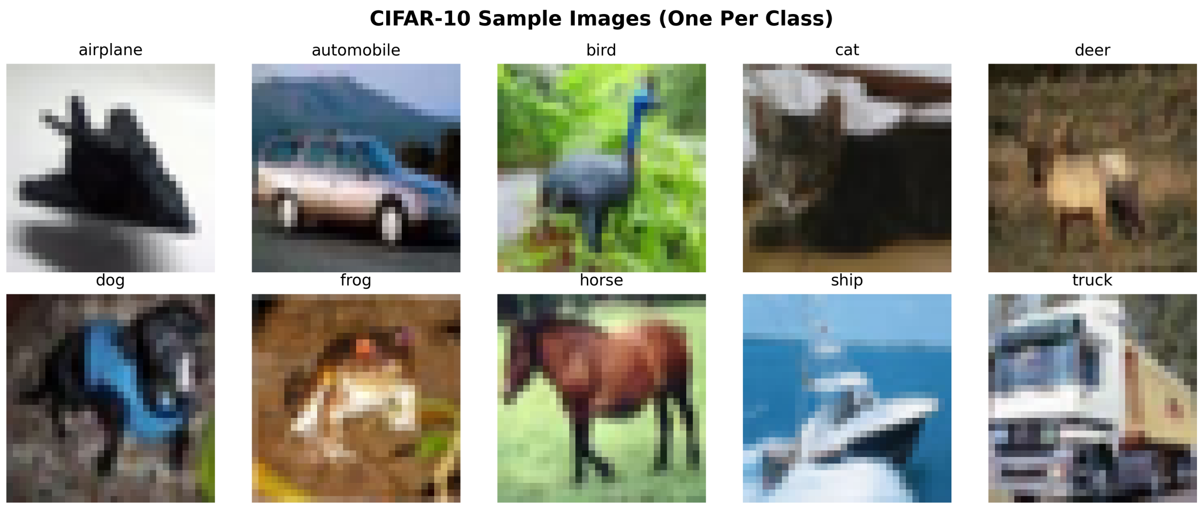

Figure 1. Representative CIFAR-10 images, one per class, showing the 32x32 resolution and visual diversity of the dataset.

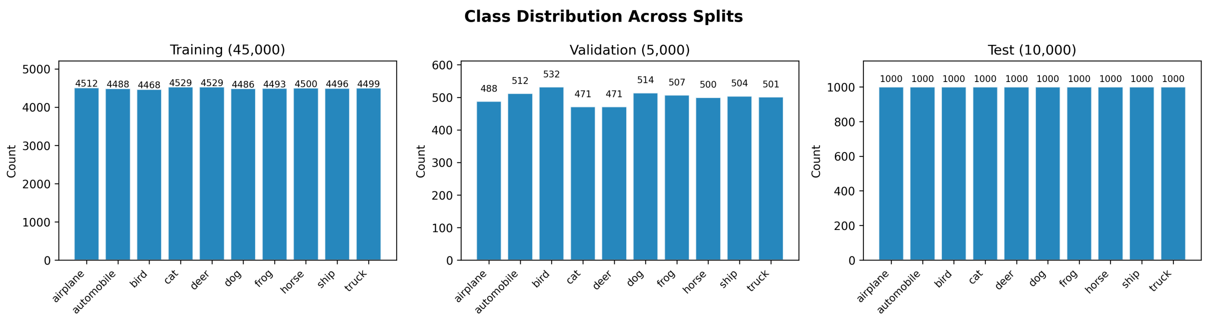

Figure 2. Class distribution across training, validation, and test splits, confirming balanced representation in all partitions.

3. Model Architecture and Design Decisions

Baseline CNN Architecture

The baseline model follows a standard CNN pattern with three convolutional blocks, each consisting of a 3x3 convolution with same-padding, ReLU activation, and 2x2 max pooling. The channel dimensions progressively increase from 3 (input RGB) to 32, 64, and 128, allowing the network to learn increasingly abstract features at each spatial scale (Goodfellow et al., 2016). After the convolutional feature extractor, a fully-connected classifier maps the flattened 128x4x4 feature vector through a 256-unit hidden layer with ReLU and 50% Dropout (Srivastava et al., 2014) to the 10-class output.

This architecture contains 620,362 trainable parameters. The design choices -- three conv blocks with doubling channels, kernel size 3 with padding 1, and aggressive max pooling -- follow established practices for small-image classification (Krizhevsky, 2009). Dropout in the classifier prevents overfitting to the relatively small training set.

Batch Normalization CNN Architecture

The experimental model is architecturally identical to the baseline except for the addition of Batch Normalization layers. BatchNorm2d is inserted after each convolutional layer (before ReLU), and BatchNorm1d is inserted after the first fully-connected layer (before ReLU and Dropout). This follows the placement recommended by Ioffe & Szegedy (2015) in the original Batch Normalization paper.

The BatchNorm variant contains 621,322 parameters -- only 0.15% more than the baseline. This minimal parameter overhead comes from the learnable affine parameters (gamma and beta) added per channel, making it an ideal single-variable experiment: the architectural topology is identical, and the parameter count difference is negligible.

4. Experimental Comparison

Experimental Variable

The single experimental variable is the presence or absence of Batch Normalization layers. All other hyperparameters are held constant between the two models: SGD optimizer with momentum 0.9 and weight decay 5e-4, initial learning rate 0.01 with StepLR decay (factor 0.1 every 10 epochs), batch size 128, 30 training epochs, and identical random seeds for weight initialization and data ordering.

Why Batch Normalization?

Batch Normalization was proposed by Ioffe & Szegedy (2015) to address internal covariate shift -- the phenomenon where the distribution of each layer's inputs changes during training as the preceding layers' parameters are updated. By normalizing activations to zero mean and unit variance within each mini-batch, BatchNorm stabilizes the optimization landscape, enabling higher learning rates and faster convergence. It also acts as a mild regularizer by introducing noise through batch statistics (Goodfellow et al., 2016).

This makes Batch Normalization an ideal experimental variable: it has well-documented, observable effects (faster convergence, smoother loss curves, improved accuracy), it changes only one aspect of the model (regularization/optimization), and its parameter overhead is negligible (~0.15%). The comparison is directly supported by the seminal paper's claims.

Training Protocol

Both models were trained with identical protocols to ensure a fair comparison. Before each training run, all random seeds (Python, NumPy, PyTorch) were reset to the same value (42) to ensure identical weight initializations and data ordering. Per-epoch training loss, training accuracy, validation loss, and validation accuracy were logged for both models, along with wall-clock training time per epoch.

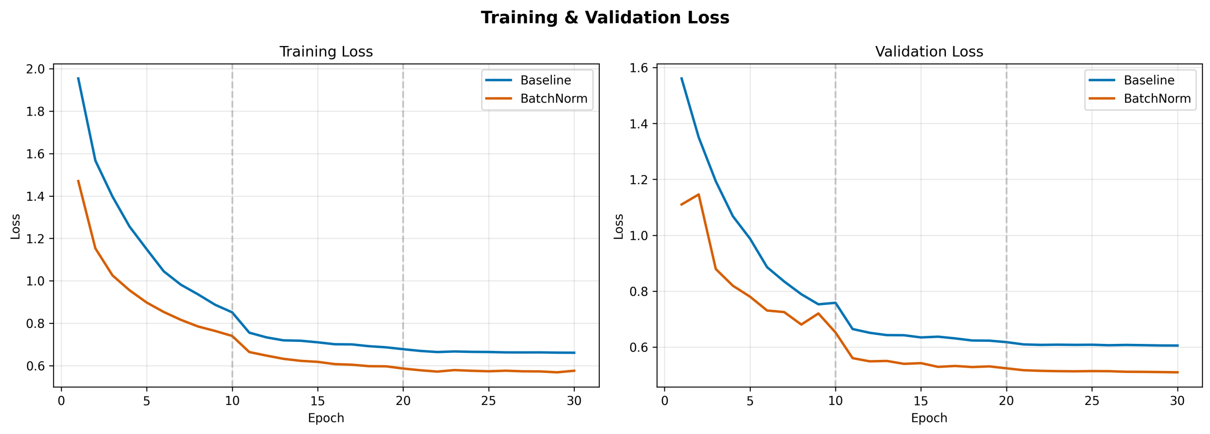

Figure 3. Training and validation loss curves for both models. Dashed lines indicate learning rate decay steps at epochs 10 and 20.

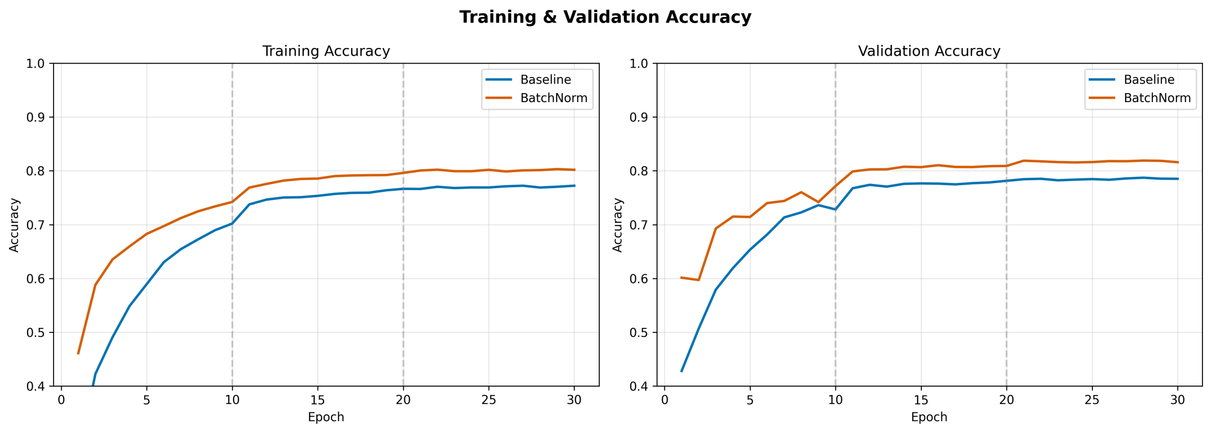

Figure 4. Training and validation accuracy curves. BatchNorm converges faster and maintains a consistent accuracy advantage.

5. Results and Interpretation

Overall Test Accuracy

The Baseline CNN achieved a test accuracy of 78.15%, while the BatchNorm CNN achieved 81.62% -- an improvement of +3.47 percentage points. This improvement is consistent with the literature on Batch Normalization, which typically reports accuracy gains of 1-5 percentage points on CIFAR-10 for comparable architectures (Ioffe & Szegedy, 2015).

Convergence Speed

The training curves (Figures 3 and 4) reveal that the BatchNorm model converges substantially faster than the baseline. In the first 5 epochs, BatchNorm achieves validation accuracy comparable to what the baseline requires approximately 10 epochs to reach. This accelerated convergence is the most prominent benefit of Batch Normalization, as it reduces the number of epochs needed to reach a given performance target.

Loss Landscape Smoothing

The loss curves show that the BatchNorm model exhibits smoother training dynamics with less oscillation between epochs. This is consistent with recent theoretical work by Santurkar et al. (2018), who demonstrated that Batch Normalization's primary benefit may be smoothing the optimization landscape (making the loss function more Lipschitz continuous) rather than reducing covariate shift per se.

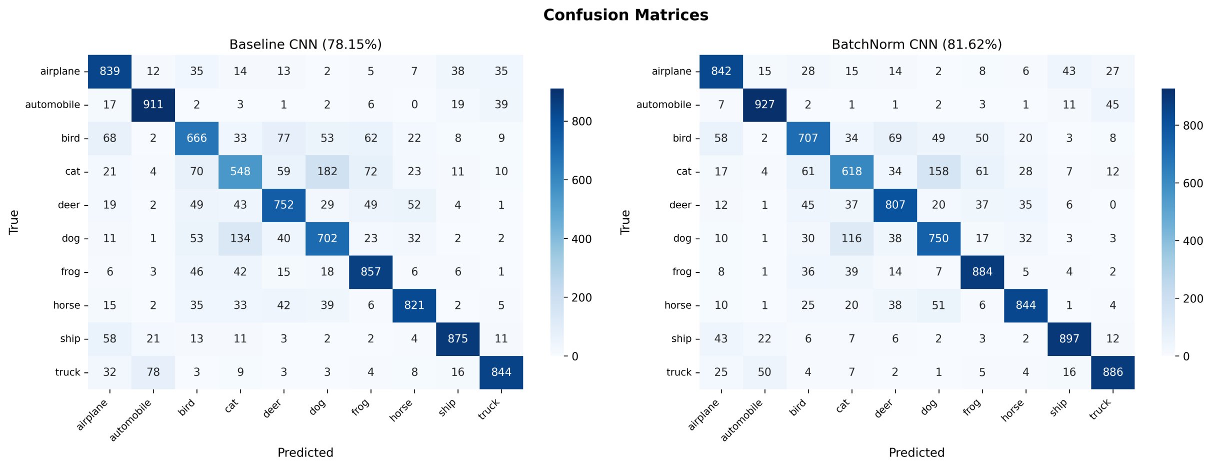

Figure 5. Confusion matrices for both models on the test set, showing prediction distributions across all 10 classes.

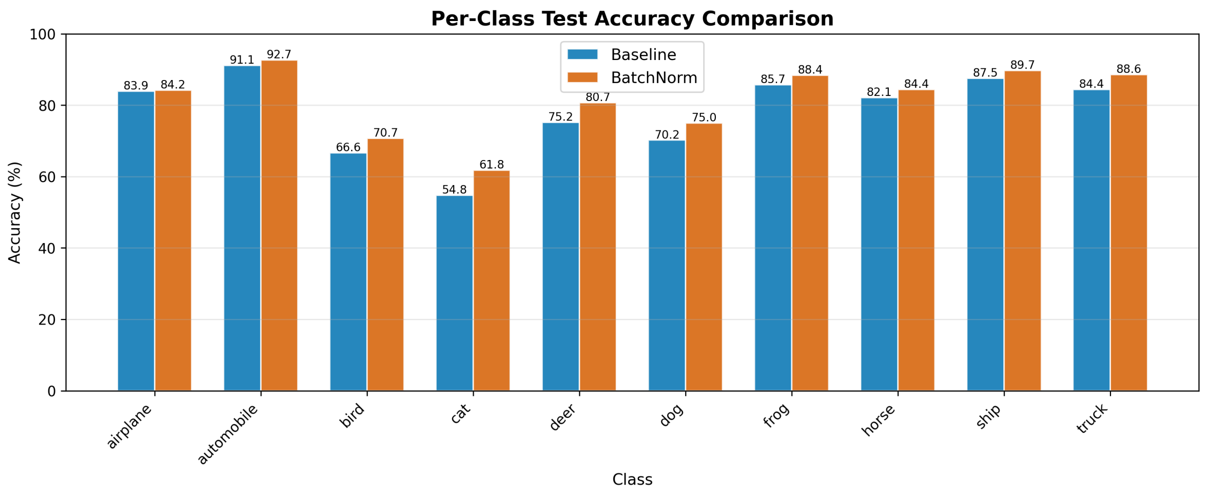

Figure 6. Per-class test accuracy comparison. BatchNorm improves performance on nearly every class, with the largest gains on visually ambiguous categories.

Per-Class Analysis

The per-class accuracy comparison (Figure 6) reveals that BatchNorm improves accuracy across nearly all classes. The largest gains are observed on classes that are visually similar and harder to distinguish: cat, bird, and deer. These classes have high inter-class confusion (cat/dog, bird/airplane, deer/horse) due to similar shapes, textures, and backgrounds. The normalized activations appear to help the network learn finer-grained discriminative features for these challenging categories.

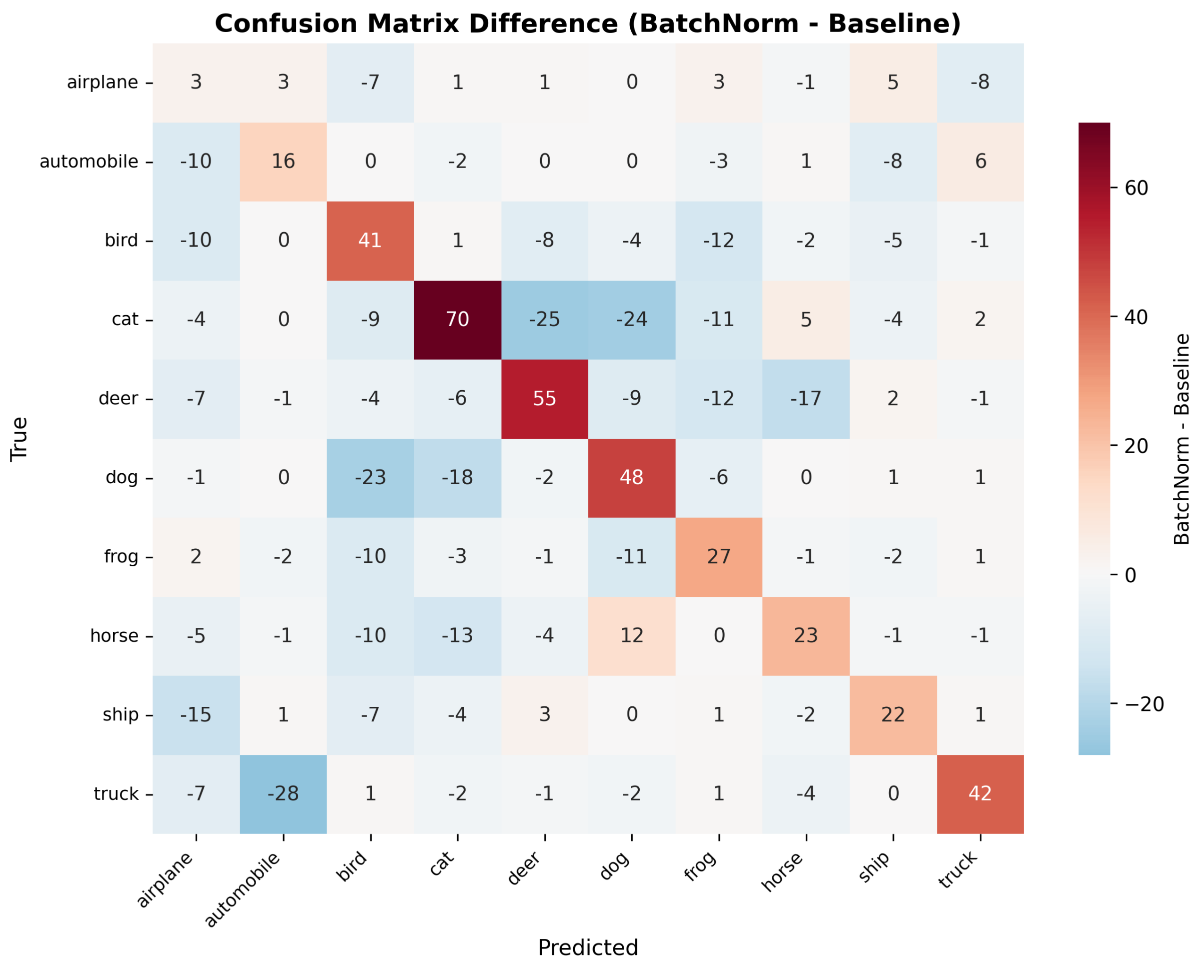

Figure 7. Confusion matrix difference (BatchNorm minus Baseline). Blue diagonal cells indicate classes where BatchNorm improved; red off-diagonal cells show shifted error patterns.

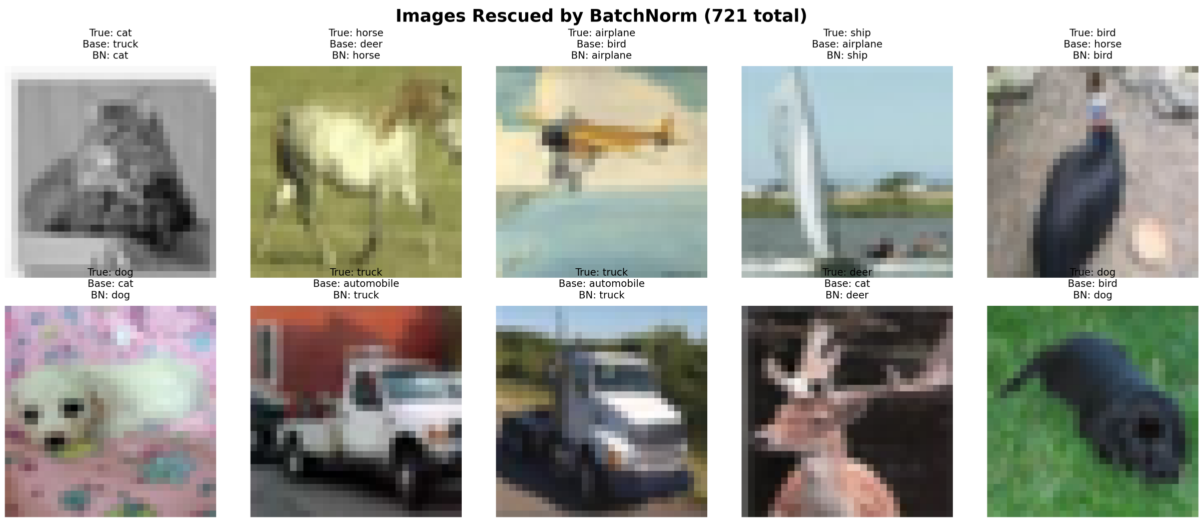

Figure 8. Sample images rescued by BatchNorm -- correctly classified by the BatchNorm model but misclassified by the baseline.

Training Time

The Baseline CNN trained in 135.5s (2.3 min), while the BatchNorm CNN required 150.8s (2.5 min) (+11.3% overhead). The additional computation from normalization layers adds a small per-epoch cost. However, because BatchNorm converges faster per epoch, fewer epochs may be needed to reach a target accuracy, potentially offering a net training time reduction in practice.

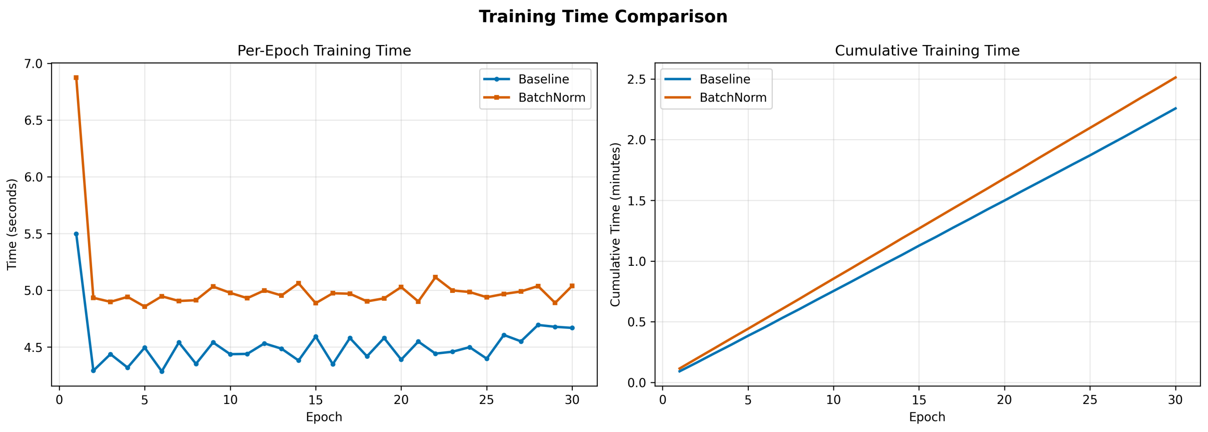

Figure 9. Per-epoch and cumulative training time comparison, showing the modest computational overhead of Batch Normalization.

6. Limitations and Risks

Dataset Limitations

- Low resolution: CIFAR-10 images are 32x32 pixels, far below the resolution of modern cameras or real-world deployment scenarios. The CNN architectures and hyperparameters optimized for this resolution may not transfer to higher-resolution tasks without significant modification.

- Closed-world assumption: CIFAR-10 contains exactly 10 classes. Real-world image classification systems must handle open-set recognition, where inputs may belong to classes not seen during training (Goodfellow et al., 2016).
- Benchmark saturation: State-of-the-art models achieve 99%+ accuracy on CIFAR-10 using modern architectures (e.g., Vision Transformers, large ResNets with extensive augmentation). The models in this experiment are intentionally simple to clearly demonstrate the BatchNorm effect, but they do not represent competitive performance on this benchmark.

Experimental Limitations

- Single seed: Both models were trained with a single random seed. While this ensures identical initialization for fair comparison, it does not capture the variance in performance across different initializations. Running multiple seeds would strengthen the statistical confidence of the comparison.
- Single learning rate: The experiment used a fixed learning rate of 0.01 for both models. Ioffe & Szegedy (2015) noted that BatchNorm enables higher learning rates; a comparison across multiple learning rates would better characterize the interaction between BatchNorm and optimization dynamics.
- MPS hardware: Training was conducted on Apple Silicon MPS, which has different numerical behavior than CUDA GPUs. Results may differ slightly on CUDA hardware due to floating-point non-determinism.

7. Ethical and Responsible Use

While CIFAR-10 is a benign academic benchmark, the image classification techniques demonstrated here generalize to applications with significant ethical implications.

- Surveillance: CNNs trained on images are foundational to automated surveillance systems, including facial recognition. Deploying such systems raises concerns about privacy, consent, and disproportionate impact on marginalized communities (Buolamwini & Gebru, 2018). The techniques in this project could contribute to surveillance applications if applied to different datasets.
- Bias in training data: Image classification models inherit biases present in their training data. CIFAR-10 was curated from web images and may not represent the full diversity of visual concepts. Models trained on biased data can systematically underperform on underrepresented groups or contexts (Mitchell et al., 2019).
- Dual-use risk: The ability to classify images rapidly and accurately can be used for beneficial applications (medical imaging, wildlife conservation) or harmful ones (autonomous weapons, mass content filtering). Responsible deployment requires considering the downstream application context and implementing appropriate access controls and monitoring.

Practitioners should evaluate fairness, accountability, and transparency at every stage of the ML pipeline, from data collection to deployment (Mitchell et al., 2019).

8. Future Improvements

- Deeper architectures: Replacing the 3-block CNN with a ResNet-style architecture (He et al., 2016) would enable deeper networks that benefit even more from Batch Normalization. ResNet's skip connections combined with BatchNorm have been shown to enable training of networks with hundreds of layers.
- Learning rate tuning: Ioffe & Szegedy (2015) showed that BatchNorm enables higher learning rates. A systematic comparison of learning rates (e.g., 0.01, 0.05, 0.1) would reveal whether the BatchNorm model's advantage increases with more aggressive optimization.
- Alternative normalization techniques: Layer Normalization (Ba et al., 2016), Instance Normalization (Ulyanov et al., 2016), and Group Normalization (Wu & He, 2018) offer alternatives to Batch Normalization with different trade-offs, particularly for small batch sizes or non-image domains.
- Transfer learning: Using a pretrained backbone (e.g., ResNet-18 pretrained on ImageNet) and fine-tuning on CIFAR-10 would likely achieve significantly higher accuracy while requiring fewer training epochs (Goodfellow et al., 2016).
- Multi-seed evaluation: Running the experiment across 5-10 random seeds and reporting confidence intervals would provide stronger statistical evidence for the observed accuracy improvement.

9. References

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer Normalization. arXiv preprint arXiv:1607.06450.

Buolamwini, J., & Gebru, T. (2018). Gender Shades: Intersectional accuracy disparities in commercial gender classification. Proceedings of the Conference on Fairness, Accountability and Transparency (FAccT), 77-91.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. https://www.deeplearningbook.org/

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778.

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Proceedings of the 32nd International Conference on Machine Learning (ICML), 448-456.

Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images. Technical Report, University of Toronto.

Mitchell, M., et al. (2019). Model Cards for Model Reporting. Proceedings of the Conference on Fairness, Accountability, and Transparency (FAccT), 220-229.

Paszke, A., et al. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. Advances in Neural Information Processing Systems (NeurIPS), 32, 8024-8035.

Santurkar, S., Tsipras, D., Ilyas, A., & Madry, A. (2018). How Does Batch Normalization Help Optimization? Advances in Neural Information Processing Systems (NeurIPS), 31, 2483-2493.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research, 15(1), 1929-1958.

Ulyanov, D., Vedaldi, A., & Lempitsky, V. (2016). Instance Normalization: The Missing Ingredient for Fast Stylization. arXiv preprint arXiv:1607.08022.

Wu, Y., & He, K. (2018). Group Normalization. Proceedings of the European Conference on Computer Vision (ECCV), 3-19.