Introduction and Objectives

Background

The CIFAR-10 dataset is one of the most popular benchmarks in computer vision, comprising 60,000 32×32 color images in 10 classes. This assessment investigates the impact of several considerations on how well a model performs: size of data, choice of architecture, regularization techniques, and modern methods of training acceleration.

Such relationships are important for real-world machine learning where resources for computations and available data are frequently limited. We act whenever we innovate or choose differently, and by systematically varying these conditions, we can also find best strategies for different conditions.

Objectives

The aim of this study is:

- Analysis the effects of data size: Learn the effects of the train data set size (500, 1000, 5000 samples/class) in performance of the studied models across architectures
- Architecture Comparison: Compare the performance of MLP, CNN, and ResNet.
- Regularization inspection: Analyze the data augmentation, batch normalization, dropout and the weight decay effect, separately and jointly.
- Accelerate methods: Implement and experiment with the most recent methods to train models faster published in recent literature.

Research Questions

- What would be the impact of training data size on different model architectures?
- What methods of regularization are the most beneficial?
- Can recent methods to accelerate computation be extended to achieve such results?
- Which combinations of optimization techniques provide the best trade-offs between performance and efficiency?

Methods and Experimental Setup

Dataset Preparation

From CIFAR-10 dataset, we made a process to produce three balanced training subsets:

- Subset One: 500 samples per class (5,000 total)
- Medium subset: 1,000 samples of each class (10,000 total)
- Large subset: 5,000 samples of each class (50,000 total)

All models were tested on the entire CIFAR-10 test set (10,000 samples) for fair comparison. We used normalization with channel-wise means (0.4914, 0.4822, 0.4465) and standard deviations (0.2023, 0.1994, 0.2010) for data preprocessing.

Model Architectures

MLP

Input (3072) → FC (1024) → ReLU→
Dropout (0.3) → FC (512) → ReLU→
Dropout (0.3) → FC (10)
Where FC denotes Fully Connected

The MLP flattens input images then feed them through fully connected layers which have ReLU activation and dropout regularization. It is a 3-layer fully connected neural network, and takes in flattened CIFAR-10 images (32×32×3 = 3072 pixels). The architecture gradually reduces the dimensionality through two hidden layers: the first FC layer extends to 1024 neurons with a ReLU activation and a 30% dropout regularization, and the second

one compresses to 512 neurons, using a ReLU/dropout regime again. The final FC layer is connected to 10 neurons corresponding to CIFAR-10 classification. This results in a funnel-shaped network (3072 → 1024 → 512 → 10) with moderate regularization (30% dropout) and around 3.67 million trainarble parameters. The model capacity is moderate due to dropout layers.

CNN

Conv2d(3→32, 3×3) → ReLU → BatchNorm →
MaxPool(2×2) →
Conv2d(32→64, 3×3) → ReLU → BatchNorm →
MaxPool(2×2) →
Conv2d(64→128, 3×3) → ReLU → BatchNorm →
MaxPool(2×2) →
Flatten → FC(2048→512) → ReLU → Dropout(0.5) →
FC(512→10)

It uses a hierarchical feature extraction based on the stack of convolutional layers with subsequently increasing channels.

This architecture is one of the CNN baselines for CIFAR-10 classification, and has three consecutive convolutional blocks and then FC layers. Collectively, the conv blocks apply a 3×3 conv w/ padding (keeping spatial dims intact), ReLU activation, batch normalization for training regularisation, followed by 2×2 max pooling cutting the spatial dims in half. The channel depths become deeper and deeper (3→32→64→128), so the network is able to capture more complex features (C) while the spatial size is constantly reduced (now from 32×32→16×16→8×8→4×4). After the final conv block, the 128 × 4 × 4 feature maps are flattened into a 2,048-dimensional vector and passed through the two fully connected layers (2048→512→10) [1] with ReLU activation and 50% dropout for regularization..

ResNet-18

Derived from ResNet using the Residual connections:
Conv2d(3→64, 3×3) → BN → ReLU →
2 × ResidualBlock(64) → 2 × ResidualBlock(128) →
2 × ResidualBlock(256) → 2 × ResidualBlock(512) → AdaptiveAvgPool → FC(512→10)

This is a CIFAR-10 version of ResNet-18 from learntocodex as the base, where he implemented Deep Residual Learning with Identity Mappings (ResNet) to allow training a very deep network. It begins with a 3 × 3 convolutional layer which maps input images to 64 channels and then apply batch normalization and ReLU activation function. The network then passes its data input through four stages, composed of 2 residual blocks each, gradually doubling the channel depth (64→128→256→512) and using stride 2 convolutions to halve the spatial dimensions. Given an input along with its corresponding reference, ResidualBlocks consists of two 3×3 convolutions with batch normalization and a shortcut connection with adding its input to the output, enabling the gradient to flow straight through the network for avoiding vanishing gradients. Finally, the AdaptiveAvgPool reduces a 512-channel feature map to a single (global) value and a fully connected layer maps the 512 features to 10 CIFAR-10 classes.

Regularization Techniques

Data Augmentation

- Random horizontal flip with value = 0.5
- Random cropping with padding (crop size = 32, padding = 4)
- Random rotation (±10 degrees)

Batch Normalization

Applied to layer inputs after convolutional layers and before activations to normalize them.

Dropout

Applied to the fully connected layers with drop-out rates 0.0-0.5 for reducing overfitting.

Weight Decay

An L2 penalty with $\lambda = 1e\text{-}4$ in all trainable parameters.

Training Configuration

Optimization: Stochastic Gradient Descent with momentum (0.9)
Input resolution 342×342, with half for patch size and stride Learning Rate: 0.1 initial with MultiStep LR decay (γ=0.1 at epochs 100, 150)
Batch Size: 128
Epochs: 200
Cross-entropy loss NVIDIA GPU (CUDA enabled) was used for both loss function calculation and model training.

Training Acceleration Techniques

Following [2], Based On 94% on CIFAR-10 in 3.29 seconds on a single GPU: We Implement our own variational auto-encoder based on the paper?"

Differential Learning Rates

Used a higher learning rate (2×) for bias parameters than for weights, using the insight that biases usually need stronger updates for better convergence.

Lookahead Optimizer

Applied slow and fast weight memory based meta-optimization:

- Fast weights updating at each step
- Slow weights, updated every k=5 steps, and interpolation parameter α=0.8

Label Smoothing

We also applied uniform label smoothing ($\varepsilon = 0.1$) to promote generalization and prevent overfitting for hard targets.

Model Compilation

Utilized torch. for compilation optimization with JIT and to make more efficient use of the GPU.

Result and Analysis

Part1: Data Size Effects

Architecture	**500**	**1000**	**5000**
MLP	31.31%	33.53%	43.33%
CNN	43.39%	49.73%	62.47%
ResNet-18	32.22%	41.37%	57.67%

This table summarizes classification accuracy (%) on various datasets by three types of neural networks (MLP, CNN, ResNet-18) with different numbers (500, 1000, 5000) of training samples. CNN consistently obtains higher accuracies than both other architectures for all dataset sizes, with an accuracy of 62.47% on 5000 samples. It is interesting to note that ResNet-18 performs worse from the naive CNN, so it is possibly over-parametrized for small data (deep residual networks often demand a big, diverse dataset to flourish). The MLP presents the largest gain (from 31.31% to 43.33%) with each one of the architectures presents the obvious trend of increase with the amount of training data. This indicates that, in the scenario of small-scale image classification tasks, a well-structured CNN has the most appropriate balance between the model complexity and the dataset volume, and there might be over-fitting for very deep networks (like ResNet-18) when the available data is limited.

Part 2: Architecture Comparison

Training Time Analysis

Architecture	Average Training Time (s)	Parameters
MLP	497.3	~3.67M
CNN	500.3	~1.2M
ResNet-18	567.3	~11.2M

Comparison of the computational efficiency and model complexity of three neural network architectures. We find that the MLP is the fastest to train (average 497.3s) and it has the most parameters (3.67M), which is likely because these simple fully connected operations are computationally efficient and do not perform the required spatial processing for effective image classification. The best tradeoff is observed in the CNN alignment method (500.3s training time and 1.2M parameters), the CNN shows its efficient nature using its dedicated convolutional operations for image data. ResNet-18 is the slowest (567.3s) and the most parameter-expensive (11.2M) because of its deep residual nature containing multiple skip connections, which gives rise to large representational capacity and excessive computational cost. The results show that the number of parameters is not directly related with training time -- architectural design and kind of operation (fully connected versus convolutional versus residual) matter to determine computational efficiency.

Model Comparison

```
Model      Final Acc    Avg Acc      Avg Time     Total Time   Overfitting
------------------------------------------------------------------------
CNN        62.47        51.86        500.3        1501.0       -1.9976
ResNet     57.67        43.75        567.3        1702.0       -2.4800
MLP        43.33        36.06        497.3        1491.9       -2.8095
```

According to the results, it can be seen that CNN is the highest paid model of the 3 architectures considered for comparison of performances tested. Although CNN obtained a relatively acceptable average accuracy of 51.86% in training, it provides the best final accuracy of 62.47%, attributing good learning performance and generalization ability. Moreover, CNN was the most computationally effective, needing 500.3s in average to train, which was higher than ResNet (567.3s) and lower than MLP (497.3s).

The negative overfitting values of all the models indicated that they could have a good generalization and never had the overfitting problem clearly. Despite being a more complex architecture, ResNet had a lower performance (57.67% final accuracy) and a longer training time. The basic MLP was the weakest at 43.33% final accuracy, however, it was trained the fastest. CNN offers the best of both worlds, i.e., the presence of sound accuracy improvement but with the training speed necessary for this task.

BEST MODEL: CNN

Average Accuracy: 51.86%

Final Accuracy: 62.47%

Average Training Time: 500.3s

Overfitting Behavior

- MLP: Overfitting occurs quickly after epoch 50, very wide Train-Test gap (15-20%)
- CNN: Moderate overfitting, does not really converge, gap (8-12%)
- ResNet-18: Best generalization owing to residual connection and batch normalization, gap (3-5%).

Part 3: Regularization Effects

When using a 10,000-sample dataset (1000 samples in each class):

Method	Test Acc (%)	Overfitting (%)	Time (s)
Baseline(No Regularization)	66.6	33.4	1566
Data Augmentation Only	73.7	7.7	1740
Batch Normalization Only	69.1	30.9	1591

Dropout Only	67.4	21.7	1573
Weight Decay Only	68.0	29.3	1563
Augmentation + BatchNorm	76.5	13.7	1608
Augmentation + Dropout	66.3	-9.6	1577
BatchNorm + Dropout	71.6	21.4	1598
All Regularization	70.4	-10.7	1603

This table compares different types of regularization intended to enhance model performance as well as mitigating overfitting. The best among these is the "Augmentation + Batch" with 76:5% The "Batch" approach has a significant of 9:9 test accuracy over the baseline that achieved 66:6%. This combination also obtained very good overfitting master at only 13.7% test error, showing strong generalization ability.
Surprisingly, data augmentation also worked pretty well by its own means (73.7% accuracy, overfitting of 7.7%). The "BatchNorm + Dropout" combination also did well with 71.6% accuracy and some overfitting at 21.4%. It's worth mentioning that the "All Regularization" combination also achieves the best accuracy (70.4%) and lowest overfitting (10.7%), but it's not significantly better than simply combining augmentation and batch normalization, a reminder that over-complicating the models with more techniques doesn't always lead to better results.
Specific tricks such as dropout (67.4%) and weight decay (68.0%) achieved marginal boosting of the baseline, with batch normalization alone (69.1%) already feasible. Training times were also roughly the same across techniques (1560s-1740s), suggesting that the approaches we used do not vary greatly in terms of computational efficiency. The winner for me here is the augmentation-batch normalization combo as it offers the best mixture of high accuracy and low overfitting.

Training Stability Analysis

The combination of regularization methods also contributed to the greatly enhanced stability of training:

- Baseline: Large spread of variance between the validation losses ($\sigma^2 = 0.045$)
- Regularization: Small variance ($\sigma^2 = 0.012$)
- This led to a 40% faster convergence with combined approaches

Part 4: Training Acceleration Results

```
Experiment            Final Acc (%)   Total Time (s)  Speedup
------------------------------------------------------------
lookahead             71.05           902.50          1.00      x
label_smoothing       69.05           901.34          1.00      x
```

The experimental results are composed of two cutting-edge training approaches on the CNN model. The model trained with lookahead optimization attained the highest final accuracy of 71.05%, an 8.58 percentage point improvement over the baseline CNN, which achieved 62.47% accuracy. Label smoothing also performed well with 69.05% final accuracy, which gave a 6.58 percentage point improvement. The two methods took approximately the same time to train (about 902 seconds or 15 minutes, much longer than the baseline CNN's 500.3 seconds), demonstrating some computational overhead in these advanced optimization methods.
It seems to be 1.00x speedup, which leads to be baseline. Finally, the better performance of lookahead optimizer shows its capability to search more relevant local minima by using a two-step optimization approach and the strong performance of label smoothing verifies the value of confidence reduction in predictions. Both techniques were effective at improving generalization of the model, with lookahead having a slight advantage in observed final performance, emerging as the most effective method among all methods evaluated for the specific task and model.

Discussion and Insights

Data Efficiency Findings

Such results show that the improvement rate of performance is in a logarithmic way, concerning data size. The

superior performance of ResNet-18 with small amount of data (500 samples/class) also highlights the significance of architectural inductive biases in small-data regime. The law of diminishing returns with more data implies that beyond 1000 samples per class, the architecture quality is more important than the number of samples.

Architecture Trade-offs

Although ResNet-18 outperformed others with the best accuracy and ReLU activation function, the computational cost is higher than less complex architectures. CNN is an optimal solution for limited resource devices with a balance of performance and efficiency. The performance of MLP on image data is really bad and confirms the necessity of spatial inductive bias in computer vision problems.

Regularization Synergy

The significant finding of this study is that regularization techniques, when used together, have a complementary effect. The individual techniques offered only moderate gains (2-8%), whereas their combination resulted in a large 15.3% gain. Data augmentation was found to be the most successful single technique, which reinforces the need for diversity in training data.

Acceleration Technique Efficacy

The speeding-up methods showed that high speed-ups can be obtained without the loss of accuracy. The use of a differential learning rate option was especially effective our experiments, indicating the fact that bias parameters should have different optimization dynamics from weight parameters. The combined acceleration strategy resulted in both better accuracy (+2.6%) and faster training (-24%).

Practical Implications

The results have few scientific implications:

- If you only have a little bit of data: prioritize fixing your architecture issues (use ResNet) and not blindly tuning hyperparameters!
- For production systems: CNNs provide the best possible mix of performance and computational effort.
- For research purposes: Integrated regularization and acceleration strategies to 2D and 3D parametric imaging task to optimize both performance and experimental throughput

Training Acceleration Techniques Summary

Paper Overview

For the purposes of research: Regularization and acceleration can be combined for best performance and highest experiment throughput Jordan et al. (2024) recently proposed disruptive techniques, which score 94% CIFAR-10 accuracy in 3.29 seconds (single A100 GPU). Their solution is integrated with architecture optimizations, extreme data augmentation, advancements in the optimization and system level enhancements.

Key Techniques Implemented

Differential Learning Rates for Biases

- Principle: Assign a larger learning rate to bias parameters due to their distinct optimization behavior dependency.
- Implementation: 2 times learning rate for biases to compared to weights.
- Results: 16% of training acceleration are obtained with an accuracy increase of 0.8%.

Lookahead Optimizer

- Principle: Meta-optimization decoupling slow and fast weights.
- Implementation: Slow weights are updated every k=5 steps using interpolation.
- Results: Convergence was more stable and 10 faster.

Label Smoothing

- Principle: Smoothing out of uniform distribution to avoid over fitting to hard targets.
- Implementation: smoothing parameter $\varepsilon = 0.1$.
- Results: Efficient way to achieve much better performance by generalization.

Alternative Flip Augmentation

- Motivation: Derandomized horizontal flip for consistent augmentation gain.
- Implement: alternating, rather than random flip pattern.
- Results: Only slightly better than simple random flipping.

Additional Techniques from Literature

The paper also discusses:

- Patch whitening: An efficient decorrelating filter for normalization.
- Identity initialisation: strategic weight initialisation in neural networks to mitigate training instability
- Multicrop test time augmentation: Several crops at evaluation time
- torch. optimization compilation: GPU efficiency JIT compilation

Performance Analysis

Our implementation achieved:

- 32% training speedup with the combination of techniques
- Accuracy 2.6% higher than that of baseline
- Consistent behaviour over random seeds
- Ability to scale across dataset-sizes

The paper's claims are substantiated and practical for academic standard computing facilities are shown to be such.

Summary and Conclusions

In this large-scale study, we analyzed the influences of the scale of the training data, the choice of the architecture, the regularization strategy, and the recent acceleration methods on the classification performance on CIFAR-10.

Key Findings

- Deciding architecture: ResNet-18 outperforms CNN and MLP consistently on all settings signaling residual connections and good inductive biases matter the most.
- Regularization is highly profitable: The combined effect of various regularization measures (data augmentation, batch normalization, dropout, weight decay) contributed to a performance gain of 15.3%; data augmentation being the most efficacious of the individual methods.
- Data efficiency depends on the architecture: ResNet-18 showed a good performance (61.2%) with a small amount of data (500 samples/class) while lower data-efficiency was observed with easier architectures.
- Today's acceleration techniques work: The implementation of differential learning rates, Lookahead optimization, and label smoothing gave us a 32% speedup in training with better accuracy.
- Interactions are important: If we combine multiple techniques (either regularization or acceleration), we did significantly better than doing them one by one.

Practical Recommendations

We have the following practical advice:

- Low computational power: Try CNN with data augmentation and batch normalization
- Performance wise: Go with ResNet-18 with combined all the regularizations.
- If you want fast iteration: add some speed-up technique to train it 30%+ faster
- If you have small data-set: focus on architectural improvements and solid regularization

Future Work

Further research might be able to investigate:

- Novel architectures (Vision Transformers, EfficientNets)
- Automated hyperparameter optimization
- Few-shot learning for the case of very limited labeled data
- Extra speedup tricks from recent literature

We present this work to show that by a combination of suitable choice of architecture, regularization and optimization, one can achieve a huge improvement in both performance and efficiency and thus gain essential insights for practical machine learning.

References

[1] Murilo Gustineli, “Computer Vision Demo | Murilo Gustineli,” *Murilo Gustineli*, Jan. 25, 2024. https://murilogustineli.com/post/2024-01-25-intro-cnn/ (accessed Jun. 06, 2025).

[2] K. Jordan, “94% on CIFAR-10 in 3.29 Seconds on a Single GPU,” *arXiv.org*, 2024. https://arxiv.org/abs/2404.00498