Understanding Data Efficiency: How Training Set Size Affects Model Performance

Shourya Kumar Srivastava

MS Computer Science – First Year

Abstract

Data collection and labeling are among the most expensive components of modern machine learning pipelines. A central practical question is: *How much labeled data is actually necessary to achieve acceptable model performance?* This project investigates how different machine learning algorithms respond to varying amounts of training data on the CIFAR-10 image classification benchmark. Using stratified subsets of the training set at multiple fractions of the full dataset, the study compares Logistic Regression, Random Forests, k-Nearest Neighbors, and a Convolutional Neural Network in terms of test accuracy, data efficiency, and overfitting behavior. The results provide empirical guidance on when simpler methods suffice and when complex deep models become necessary. All code and result artifacts are available in a public GitHub repository.[1]

1 Introduction

In modern machine learning practice, the informal mantra "more data is better" is widely accepted. However, collecting and labeling large datasets is expensive and often infeasible in real-world applications, particularly in domains such as healthcare, autonomous driving, and scientific imaging. As a result, practitioners frequently face strict annotation budgets and must ask a fundamental question: *How much data is actually required to achieve a desired performance level?* While increasing training set size typically improves model performance, the relationship between data volume and generalization error is highly non-linear and depends strongly on the learning algorithm.

This project addresses the following central question: *How does training set size affect the performance and data efficiency of different machine learning algorithms?* Concretely, the goal is to characterize learning curves for four representative algorithms on the CIFAR-10 dataset: Logistic Regression (linear baseline), Random Forest (tree-based ensemble), k-Nearest Neighbors (non-parametric instance-based learning), and a Convolutional Neural Network (CNN, deep learning). By training these models on stratified subsets ranging from 5% to 100% of the available training data while keeping the test set fixed, the analysis aims to answer three core questions: (1) At what point do models exhibit diminishing returns from additional data? (2) Do simpler models outperform complex ones when data is scarce? (3) How does data efficiency differ across algorithm families?

Analyzing CIFAR-10 in this context presents several challenges. First, the target variable is a 10-class categorical label corresponding to object categories in natural images, requiring models to capture complex spatial and semantic structure rather than simple tabular correlations. Second, the input dimensionality is high (3 072 raw pixel features per image), creating a substantial risk of overfitting, particularly when the number of training examples is small relative to the feature dimension. Third, maintaining methodologically

[1] `https://github.com/shouryasrivastava/cifar10-data-efficiency/tree/main`

consistent hyperparameters across different data scales is non-trivial: settings that work well with 50 000 training examples may lead to underfitting or unstable optimization when only 2 500 examples are available. Addressing these challenges motivates a careful experimental design with stratified sampling, fixed test sets, and repeated runs under different random seeds.

2 Related Work

The question of how much data is required for reliable classification has been studied from multiple perspectives, including classical statistical learning theory and modern empirical deep learning. Figueroa et al. investigated sample size requirements for classification in medical informatics, proposing the use of learning curves to predict performance as a function of training set size and highlighting that required sample size depends on the complexity of the decision boundary and class imbalance [2]. Their work emphasizes the importance of modeling performance as a curve rather than focusing on a single training size.

Beleites et al. studied sample size planning for classification, emphasizing that small sample sizes can severely distort performance estimates and lead to high variance in evaluation metrics [3]. This work motivates the use of repeated subsampling or cross-validation in small-data regimes to obtain more stable estimates. In the present study, this idea is reflected through the use of multiple random seeds for stochastic algorithms.

CIFAR-10 itself was introduced by Krizhevsky as part of a broader effort to study “tiny images” and learn multiple layers of visual features [1]. Early work on this dataset focused on hand-crafted features such as GIST or Histogram of Oriented Gradients (HOG), coupled with shallow classifiers like Support Vector Machines. With the rise of deep learning, much of the literature shifted toward designing increasingly sophisticated CNN architectures and data augmentation schemes targeting high absolute accuracy rather than data-efficiency characterization.

More recently, Hestness et al. empirically studied how deep learning performance scales with data and model size across several domains, demonstrating approximate power-law relationships between data size and generalization error [4]. Their results suggest that performance improvements from additional data are predictable up to a certain regime, particularly for large neural networks. However, their focus is on scaling behavior of deep models rather than on direct comparisons between deep and shallow methods in low- to medium-data regimes.

A complementary perspective comes from Banko and Brill, who demonstrated in the context of natural language processing that simple models can approach or match the performance of more complex models when trained on very large corpora [5]. This raises the question of whether similar effects hold for image classification tasks such as CIFAR-10, and whether simpler algorithms can be competitive given enough data, or conversely, whether complex models are necessary even in data-rich settings.

Cho and Lee studied CNN architectures on CIFAR-10, showing that deeper networks can achieve superior performance but are more prone to optimization difficulties and overfitting without careful design and sufficient data [6]. Their work reinforces the view that deep models have high representational capacity but are sensitive to training conditions and data scale. The present project builds on these insights by explicitly contrasting deep and non-deep algorithms across a range of training set sizes on a standardized benchmark.

3 Related Implementations

CIFAR-10 is a widely used benchmark on platforms such as Kaggle, where many public kernels and notebooks provide practical implementations of classification methods. One common class of implementations focuses on CNN models implemented in Keras or PyTorch, often using data augmentation and transfer learning. For example, a representative Kaggle kernel ("CIFAR-10 CNN with Augmentation") uses a relatively deep convolutional architecture with batch normalization, dropout, and aggressive data augmentation, achieving high test accuracy on the full dataset. Such implementations demonstrate the potential of deep learning on CIFAR-10 but typically assume access to the entire training set and, in some cases, leverage pre-trained ImageNet models, thereby obscuring the underlying data efficiency of the model.

Another class of Kaggle implementations applies classical machine learning algorithms to CIFAR-10, such as Support Vector Machines, Random Forests, and gradient boosting methods, often after flattening the images into vectors or extracting simple features. A representative kernel ("SVM vs Random Forest on CIFAR") compares these methods and reports that while SVMs can perform reasonably well, they are computationally expensive and scale poorly with the full 50 000-image training set. Random Forests, on the other hand, provide a more efficient trade-off between training time and accuracy in this context. However, such work generally focuses on single-point performance rather than on systematically characterizing the effect of training set size.

The present project differs from these implementations in its explicit focus on learning curves and data efficiency. Instead of optimizing a single model for maximum accuracy, the experimental design treats training set size as a primary independent variable and evaluates how each algorithm's performance changes as more labeled data becomes available. This shift in perspective enables a more principled discussion of algorithm choice under realistic data budget constraints.

4 Dataset and Data Analysis

The experiments are conducted on the CIFAR-10 dataset [1], which consists of 60 000 color images of size 32×32 pixels with three RGB channels. Each image is labeled with one of 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 in its standard form. The class distribution is uniform, with 6 000 images per class, ensuring balanced classification.

Each image can be represented as a vector in $\mathbb{R}^d$ with $d = 32 \times 32 \times 3 = 3{,}072$ dimensions obtained by flattening the pixel grid. Raw pixel intensities are integers in the range $[0, 255]$. To facilitate optimization and stabilize gradient-based methods, input features are standardized using per-channel normalization. For a pixel value x, the standardized value x' is computed as

$$x' = \frac{x - \mu}{\sigma},$$

where μ and σ denote the mean and standard deviation computed over the training set (or the current training subset) respectively. These statistics are then applied to transform both the training and test images, with care taken to avoid any leakage of test information into the training process.

Exploratory analysis confirms that the dataset is clean and contains no missing values. Class-wise averages and sample visualizations indicate that some classes (such as "frog") have distinctive color and shape patterns, whereas others (such as "truck" and "automobile") share substantial visual similarities, making them harder to distinguish. The combination of high dimensionality and moderate resolution suggests a non-trivial decision boundary that is unlikely to be well-approximated by linear models. This motivates the inclusion of both shallow and deep models in the comparative analysis.

For the purposes of studying data efficiency, stratified subsets of the 50 000-image training portion are constructed at the following fractions of the full training set: 5%, 10%, 25%, 50%, 75%, and 100%. These correspond to training set sizes of 2 500, 5 000, 12 500, 25 000, 37 500, and 50 000 images, respectively. Stratification ensures that each subset maintains the original class proportions, i.e., each class is equally represented in every subset. The test set of 10 000 images is kept fixed across all experiments.

5 Methods and Algorithms

This section describes the learning algorithms evaluated in the study and their mathematical formulations.

5.1 Logistic Regression

Multinomial Logistic Regression serves as a linear baseline. Given an input vector $\mathbf{x} \in \mathbb{R}^d$, the model assumes $K = 10$ classes with parameters $\{\mathbf{w}_k, b_k\}_{k=1}^{K}$. The probability that $\mathbf{x}$ belongs to class k is given by the softmax function:

$$P(y = k \mid \mathbf{x}) = \frac{\exp(\mathbf{w}_k^\top \mathbf{x} + b_k)}{\sum_{j=1}^{K} \exp(\mathbf{w}_j^\top \mathbf{x} + b_j)}.$$

The model is trained by minimizing the cross-entropy loss over the training set:

$$\mathcal{L} = -\sum_{i=1}^{N} \sum_{k=1}^{K} \mathbb{I}[y_i = k] \log P(y_i = k \mid \mathbf{x}_i),$$

possibly with ℓ_2 regularization on the weights. Logistic Regression has high bias but relatively low variance, and may thus be more robust on small datasets, though its linear decision boundary is not well suited to complex image manifolds.

5.2 k-Nearest Neighbors

The k-Nearest Neighbors (kNN) classifier is a non-parametric, instance-based learning method. There is no explicit training phase beyond storing the training data. Given a query point $\mathbf{x}_q$, the algorithm identifies the set $N_k(\mathbf{x}_q)$ of the k closest training examples according to a distance metric, typically Euclidean distance:

$$d(\mathbf{x}_q, \mathbf{x}_i) = \sqrt{\sum_{j=1}^{d} (x_{q,j} - x_{i,j})^2}.$$

The predicted class is the majority label among the neighbors:

$$\hat{y} = \arg\max_k \sum_{\mathbf{x}_i \in N_k(\mathbf{x}_q)} \mathbb{I}[y_i = k].$$

In this project, $k = 5$ is used. While kNN can capture highly non-linear decision boundaries, its performance may degrade in high dimensions due to the curse of dimensionality, and inference time scales linearly with the number of training samples.

5.3 Random Forest

Random Forests are ensemble methods that construct a collection of decision trees and aggregate their predictions. Each tree is trained on a bootstrap sample of the training data, and at each internal node, a random subset of features is considered when determining the best split. For classification, node purity is often measured using the Gini impurity:

$$G = \sum_{k=1}^{K} p_k (1 - p_k),$$

where p_k is the proportion of training samples from class k in the node. Splits are chosen to maximize the reduction in impurity:

$$\Delta G = G_{\text{parent}} - \left(\frac{n_L}{n} G_L + \frac{n_R}{n} G_R\right),$$

where G_L and G_R are the impurities of the left and right child nodes, and n_L, n_R, and n are the corresponding sample counts.

In this study, a Random Forest with 100 trees is used. The final prediction is obtained by majority voting across all trees. Random Forests are capable of modeling non-linear relationships and interactions without requiring feature scaling, and they often perform well with relatively modest feature engineering.

5.4 Convolutional Neural Network

The Convolutional Neural Network (CNN) considered here is a relatively shallow architecture with three convolutional layers followed by fully connected layers. The fundamental operation is discrete convolution. Given an input feature map I and a kernel K, the convolution $(I * K)$ at location (i, j) is defined as

$$(I * K)_{ij} = \sum_m \sum_n I_{i+m,j+n} K_{mn}.$$

Each convolutional layer applies multiple such kernels, followed by a non-linear activation function, typically the Rectified Linear Unit (ReLU),

$$f(x) = \max(0, x),$$

and optionally a pooling operation such as max pooling that reduces spatial resolution while preserving salient features.

The network concludes with one or more fully connected layers that map the final feature representation to class logits. The model is trained using the cross-entropy loss and optimized with Adam, a variant of stochastic gradient descent with adaptive learning rates. In this project, a learning rate of 0.001, batch size of 64, and 50 training epochs are used. To mitigate overfitting in low-data regimes, dropout with rate 0.3 is applied to fully connected layers, randomly zeroing out a fraction of activations during training.

6 Experimental Setup

All experiments use the standard 10 000-image CIFAR-10 test set as a held-out evaluation set. The 50 000-image training set is partitioned into stratified subsets of the following sizes: 2 500 (5%), 5 000 (10%), 12 500 (25%), 25 000 (50%), 37 500 (75%), and 50 000 (100%) images. Stratification ensures that each subset preserves the original class proportions, i.e., each class contributes an equal number of samples to each subset.

For each subset, the same preprocessing pipeline is applied. Specifically, per-channel means and standard deviations are computed from the subset and used to standardize both the subset and the fixed test set. Logistic Regression, Random Forest, and kNN models are trained using scikit-learn with largely default parameters, except for the number of trees in the Random Forest and the choice of k in kNN, which are set to 100 and 5, respectively. The CNN is implemented in a deep learning framework (e.g., PyTorch or TensorFlow) and trained on a GPU (e.g., Google Colab's T4) for 50 epochs with batch size 64 and Adam optimizer with learning rate 0.001.

To account for stochasticity in training, particularly for the CNN and Random Forest, experiments are repeated for three different random seeds. For each configuration, the mean test accuracy across seeds is reported. Training time is also recorded qualitatively to understand time–accuracy trade-offs, though detailed timing analysis is beyond the primary scope of the present report. Throughout, the test set is never used for model selection or hyperparameter tuning.

7 Results

Table 1 summarizes the test set accuracies achieved by each algorithm at different training set sizes.

Table 1: Test set accuracy (%) on CIFAR-10 for different training set sizes and algorithms.

Training Size	Logistic Regression	Random Forest	CNN (3-layer)	kNN (k=5)
2,500 (5%)	26.7	38.1	53.1	26.1
5,000 (10%)	27.3	40.6	59.2	28.0
12,500 (25%)	30.4	43.3	66.2	30.1
25,000 (50%)	33.3	45.2	71.9	31.5
37,500 (75%)	35.3	45.9	74.3	33.0
50,000 (100%)	37.2	46.6	77.1	33.9

Several patterns emerge from these results. First, the CNN is the best-performing model at every training set size, achieving 53.1% accuracy already at 5% of the data and reaching 77.1% at 100% of the data. Random Forest is consistently the strongest traditional baseline, but it saturates around 46.6% even with the full dataset, while Logistic Regression and kNN plateau at 37.2% and 33.9% respectively.

Second, the gap between CNN and Random Forest widens as more data is added: from roughly 15 percentage points at 5% (53.1% vs. 38.1%) to over 30 percentage points at 100% (77.1% vs. 46.6%). This indicates that additional data disproportionately benefits the convolutional model, whereas tree-based and linear methods hit a performance ceiling much earlier. Logistic Regression exhibits the clearest pattern of diminishing returns; its incremental gain from doubling the training data from 25% to 50% is modest, and from 50% to 100% is similarly small, suggesting that the model is bias-limited. kNN performs only slightly better than Logistic Regression and also displays diminishing returns, likely due to the curse of dimensionality and lack of learned feature extraction.

8 Analysis and Discussion

The empirical findings can be interpreted in light of the bias–variance trade-off. Random Forests, by aggregating many decision trees trained on bootstrapped samples with feature subsampling, achieve a low-variance, moderately low-bias model that is well suited to small and medium-sized datasets. They can exploit non-linear combinations of raw pixel features without requiring explicit feature engineering. However, their performance saturates once the easier patterns are captured, leading to a clear plateau around 47% accuracy.

CNNs, in contrast, have extremely high representational capacity and are capable of learning complex, hierarchical features that capture translation invariance, local patterns (e.g., edges, corners), and higher-order structures across the image. The results show that this capacity is beneficial even in low-data regimes: unlike the common belief that deep models only shine with massive datasets, the CNN already outperforms all baselines at 5% of the data. As the training set size increases, the variance of the estimator decreases and the model leverages its superior representational power, leading to a growing gap over traditional methods.

The poor performance and early saturation of Logistic Regression highlight the limitations of linear models on raw image pixels. Even with large amounts of data, such models cannot capture the non-linear manifolds on which image classes lie. This observation is consistent with the broader literature on image recognition, where deep and/or kernelized models are usually required for competitive performance. kNN's relatively weak performance further illustrates the curse of dimensionality: distance-based methods struggle when each input has thousands of correlated features.

These results fit qualitatively with prior work on data scaling in deep learning [4], which suggests that performance improvements from additional data tend to follow a predictable scaling law for large models, and with earlier findings by Banko and Brill [5] that simple models can be competitive when data is abundant. In the present context, however, even the largest training set considered (50 000 examples) is insufficient for a simple linear model to close the performance gap with non-linear methods. This underscores that the relationship between data volume, model complexity, and performance is domain-dependent.

In practical terms, the results suggest that for image classification tasks like CIFAR-10, convolutional architectures are preferable across essentially all data budgets. Traditional models such as Random Forests provide a reasonable baseline and are faster to train, but their accuracy ceiling is significantly lower. CNNs require more careful implementation and GPU resources, but their superior performance even at 5% of the data makes them the method of choice whenever such infrastructure is available.

9 Conclusion

This project studied how training set size affects the performance of different machine learning algorithms on the CIFAR-10 image classification benchmark. By constructing stratified subsets of the training data and evaluating Logistic Regression, k-Nearest Neighbors, Random Forests, and a Convolutional Neural Network, the analysis provided a comparative view of data efficiency and learning curves across algorithm families.

The main findings can be summarized as follows. First, CNNs outperform traditional machine learning baselines at all examined data fractions, achieving 53.1% accuracy with only 5% of the training data and 77.1% with the full dataset. Second, Random Forests are strong but ultimately limited competitors, saturating around 46.6% accuracy despite additional data. Third, linear models such as Logistic Regression and distance-based methods such as kNN are fundamentally constrained on raw pixel inputs for CIFAR-10, displaying early performance plateaus and relatively low ceilings.

For practitioners facing annotation budgets, these results suggest that convolutional architectures are a sound investment even in low-data regimes, provided GPU resources are available. Traditional models remain useful as quick baselines or when computational constraints are strict, but they are unlikely to match the accuracy of well-designed CNNs. Future work could extend this analysis by incorporating semi-supervised learning, data augmentation, and transfer learning, as well as by exploring other datasets and domains to assess the generality of the observed patterns.

Appendix: Code and Reproducibility

The full code used to generate the datasets, train the models, and produce the reported results and plots is available in a public GitHub repository at:

`https://github.com/shouryasrivastava/cifar10-data-efficiency/tree/main`

The repository includes:

- `data_prep.ipynb`: scripts for stratified sampling and normalization.
- `models.ipynb`: implementations and training loops for Logistic Regression, Random Forest, kNN, and CNN.
- `results_viz.py`: scripts for generating learning curve plots and result tables.

Statement of Contributions

This project was completed individually by Shourya Kumar Srivastava. All aspects of the work, including conceptualization, implementation, experimentation, analysis, and writing, were carried out by the same person.

References

[1] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical Report, University of Toronto, 2009.

[2] R. L. Figueroa, Q. Zeng-Treitler, S. Kandula, and L. H. Ngo. Predicting sample size required for classification performance. *BMC Medical Informatics and Decision Making*, 12(1):8, 2012.

[3] C. Beleites, U. Neugebauer, T. Bocklitz, C. Krafft, and J. Popp. Sample size planning for classification models. *Analytica Chimica Acta*, 760:25–33, 2013.

[4] J. Hestness, S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad, M. Patwary, Y. Yang, and Y. Zhou. Deep learning scaling is predictable, empirically. In *Proceedings of the Conference on Machine Learning Systems*, 2017.

[5] M. Banko and E. Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, 2001.

[6] J. Cho and K. Lee. Deep convolutional neural networks for CIFAR-10 classification. In *International Conference on Information Science and Applications*, 2017.