PRML Final Homework Report (25 Spring)

Xinyuan Guo **Yuan Gao** **Tianlin Pan** **Shuwen Xu** **Yu Shi** **Hanlin Pan**

TransfArmer, University of Chinese Academy of Sciences*

Abstract

This report presents a comparative analysis of four image classification methods on the CIFAR-10 dataset consisting of 60,000 color images across 10 object categories: **1)** logistic regression, **2)** boosting with NN and CNN feature extraction, **3)** ResNet-based convolutional neural networks, and **4)** Vision Transformers (ViT). The study aims to analyze the impact of feature representation and model architecture on classification performance. Moreover, evaluation highlights how feature representation and model architecture influence classification performance. While traditional machine learning models offer interpretability, with logistic regression and boosting achieving 38.89% and 69.45% accuracy respectively, deep learning approaches yield significantly higher performance, achieving up to 95% accuracy. These results underscore the effectiveness of deep architectures in image classification tasks.

Our codes are available at https://github.com/UCAS-transfArmer/PRML-Final.

1 Introduction

Image classification is a fundamental task in computer vision, serving as a building block and feature attractor for numerous downstream applications such as object detection, scene understanding, and autonomous systems. Early approaches to image classification relied heavily on hand-crafted features, such as SIFT [1] and HOG [2], combined with traditional classifiers like support vector machines (SVMs) [3]. These methods, while effective to some extent, were limited by their reliance on manually designed features and shallow architectures.

Over the past 15 years, the advent of GPUs and enhanced computing power have propelled deep learning to a dominant position in computer vision. Convolutional Neural Networks (CNNs), known for their weight sharing and translation invariance, have become the seminal architecture for image classification. Architectures such as AlexNet [4], VGG [5], and ResNet [6] have significantly improved performance on large-scale benchmarks by increasing network depth and incorporating residual connections. For example, ResNet-50 achieves a top-1 accuracy of 75.3% in the ImageNet dataset in 2016. However, CNNs still exhibit limitations in modeling long-range dependencies and global context, which has led to the emergence of transformer-based models in computer vision.

Inspired by the great performance in natural language processing of the transformer architecture [7], Vision Transformers (ViTs) [8] apply self-attention mechanisms to image patches, enabling global

*__Group member contributions__:

1. *Xinyuan Guo*: Code implementation and experiments for ResNet.
2. *Yuan Gao*: Report writing for Section 1, 2, 3.1 and 3.2.
3. *Tianlin Pan*: Project leader. Codebase framework, report revisions and experiment on LR.
4. *Shuwen Xu*: Code implementation and experiments for ViT.
5. *Yu Shi*: Code implementation and experiments for AdaBoost.
6. *Hanlin Pan*: Report writing for Section 3.3, 3.4 and 4.

Preprint.

feature interactions without relying on convolutional operations. ViTs have achieved competitive or superior performance on various image classification tasks, with ViT-B/16 reaching 85% top-1 accuracy on ImageNet when pre-trained on large-scale datasets. However, they typically require large-scale datasets and high computational resources for effective training, making them less accessible in resource-constrained settings.

The specific contributions of our study are as follows:

- We provide a comprehensive comparative analysis of four distinct image classification methodologies on the CIFAR-10 dataset: logistic regression, AdaBoost with various weak learners, ResNet, and the Vision Transformer (ViT). This selection spans the spectrum from classical machine learning to state-of-the-art deep learning architectures.
- Our unified experimental framework offers practical insights into the trade-offs between model complexity, feature representation, and classification performance on a small-scale dataset, demonstrating the performance gap between traditional methods (38.89% to 69.45% accuracy) and deep learning models (up to 94% accuracy).
- We investigate the effectiveness of a hybrid approach by combining AdaBoost with a CNN-based feature extractor, analyzing how traditional ensemble methods can leverage features from deep models.
- The study empirically demonstrates the impact of architectural choices and training configurations, including the degradation problem in plain deep CNNs versus the stability of ResNets, the data-dependency of Vision Transformers, and the sensitivity of various models to hyperparameters like learning rates.

2 Dataset

The CIFAR-10 dataset [9] consists of 60,000 32×32 low-resolution color images drawn from 10 mutually exclusive categories, including airplanes, automobiles, birds, cats, deer etc., as shown in Fig. 1. The dataset was constructed by labeling a subset of the Tiny Images dataset, with human annotators instructed to remove mislabeled or ambiguous images based on strict visual criteria, such as realism, object prominence, and clarity of class identity. To ensure class balance, each category contains 6,000 images, with 5,000 used for training and 1,000 for testing. Duplicate images were removed using L_2 norm comparison.

In our experiments, we used the standard training/test split and applied minimal pre-processing. All images were normalized by subtracting the per-channel mean computed over the training set. No additional data augmentation techniques were applied unless otherwise specified in the experiment settings.

Figure 1: Example images from the CIFAR-10 dataset. Each image is 32×32 pixels and belongs to one of 10 object categories

3 Experiments

3.1 Logistic Regression (LR)

Model Architecture. We implement a simple logistic regression model as a baseline classifier. The model consists of a single fully connected layer that maps the flattened input image vector to the output logits corresponding to each class. Given an input image of size $32 \times 32 \times 3$, the input is first reshaped into a 3072-dimensional vector and then passed through a linear transformation to produce

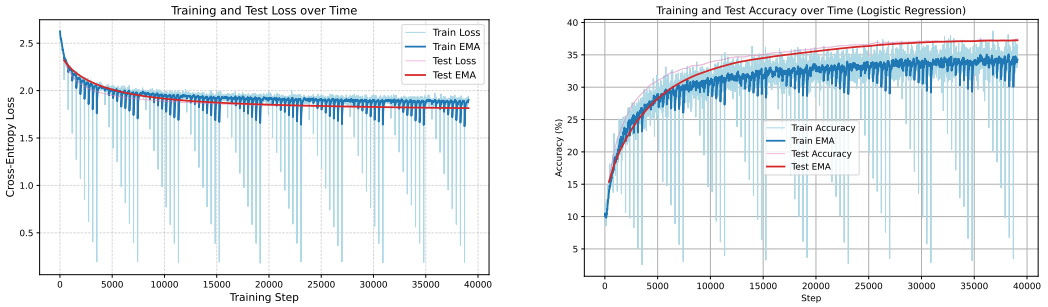
a 10-dimensional output, representing the unnormalized class scores. In our implementation, the model directly outputs unnormalized logits, which are then passed to the standard cross-entropy loss function. Although no explicit softmax layer is applied, the loss function internally performs the softmax operation in PyTorch followed by negative log-likelihood computation. This architecture captures only linear relationships in the input features and does not leverage spatial structure, leading to poor performance in image classification tasks.

However, this basic single-layer neuron network is the only method in our work that has convergence guarantees. Logistic regression with softmax output minimizes a convex objective: the multi-classes cross-entropy loss, which ensures that any local minimum is also a global minimum [10]. Specifically, the loss function:

$$\mathcal{L}(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \log \frac{e^{\theta_{y_i}^\top x_i}}{\sum_{j=1}^{k} e^{\theta_j^\top x_i}} \tag{1}$$

is convex with respect to the model parameters θ, and its gradient is Lipschitz continuous. As a result, optimization method like Adam is guaranteed to converge to the global optimum under standard assumptions on the step size.

Implementation details. We implement logistic regression as a single-layer linear classifier with 30,730 trainable parameters, mapping each $32 \times 32 \times 3$ image (flattened into a 3072-dimensional vector) to 10 class logits. The model is trained on CIFAR-10 using the Adam optimizer with a learning rate of 1×10^{-5}, a batch size of 128, and for 100 epochs.



(a) Training and test loss of logistic regression on CIFAR-10 over training steps.

(b) Training and test accuracy of logistic regression on CIFAR-10 over training steps.

Figure 2: Training curves of logistic regression on CIFAR-10. The training curve exhibits a sawtooth pattern due to periodic logging and reset of accumulated loss in fixed-size intervals, as explained in the implementation.

Main Results. As shown in Fig. 2a, the training loss for logistic regression exhibits a distinct undulating pattern, characterized by sharp drops followed by gradual increases. This non-monotonic behavior is primarily due to the logging mechanism in the training loop. Specifically, the accumulated loss (`running_loss`) is averaged and recorded only every fixed number of iterations (`log_per_iter`), after which it is reset to zero. This periodic reset results in sharp drops at each logging point, followed by a gradual rise as loss accumulates again, producing a sawtooth-like appearance in the loss curve.

Fig. 2b illustrates the performance of the logistic regression model on the CIFAR-10 dataset over 5000 training steps. The training accuracy (light blue curve) exhibits a highly fluctuating pattern, ranging approximately around 35.5%. Similar to the training loss, this rapid oscillation is due to the per-interval logging of accuracy, where `running_acc` is reset after each logging interval, causing the reported accuracy to dip before accumulating again. The test accuracy (pink line) shows a more stable and consistently increasing trend, reaching approximately 37.8% by the end of training. The test EMA (red curve) closely follows the test accuracy, providing a smoothed view of its progression.

A notable observation from the accuracy curve is that the test accuracy consistently exceeds the training accuracy throughout most of the training process. This counterintuitive result tends to happen when using a extremely simple model like logistic regression on a complex dataset such as CIFAR-10. Due to its limited expression capacity, the model struggles to capture patterns in the training data, leading to lower training accuracy. However, this simplicity also prevents overfitting, and the model

may generalize reasonably well to the test set. Furthermore, training accuracy is computed on mini-batches during ongoing weight updates, while test accuracy is evaluated periodically on the full test set using a fixed model snapshot, which can result in a more stable and occasionally higher performance on the test set.

3.2 AdaBoost

Model Architecture. The AdaBoost model is an ensemble learning method that sequentially trains multiple weak learners and combines their predictions [11]. For the CIFAR-10 dataset, the AdaBoost is configured to use a specified number of weak estimators (`num_estimators`) and a chosen `weak_learner_type` during its fitting time.

The Boost model itself does not have a fixed, monolithic architecture like a standard neural network. Instead, it ensembles many simpler models. During the training process, it iteratively trains `self.n_estimators` (maximun 20) individual weak learners. In order to compare the overall performance with respect to basic weak classifiers, we chose weak learners to be one of three types as follows:

- **Multilayer Perceptron**: This network consists of two fully connected `linear` layers separated by a ReLU activation function. For CIFAR-10 images $32 \times 32 \times 3$, the input dimension would be 3072 after flattening, which would then be mapped to a `hidden_dim_1` (default 512) and finally to 10 classes for CIFAR-10.
- **CNN**: This network is designed for image data and includes a convolutional layer with 32 output channels, followed by a ReLU activation and a max-pooling layer. The output of the pooling layer is then flattened to 8192 dimension for 32×32 inputs, and passed through two fully connected layers with a ReLU activation and dropout layer between them.
- **Logistic Regression**: This is the simplest option, comprising a single fully connected layer that maps the flattened input directly to the 10 classes output, effectively performing logistic regression.

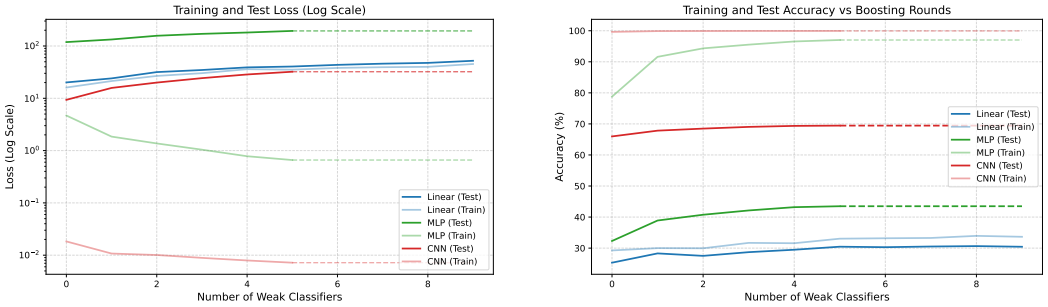
Implementation details. Unlike other models evaluated in our work, the performance improvement in ensemble learning primarily manifests after each new weak learner is integrated into the model. Therefore, a specialized training and evaluation pipeline was adopted. After completion of training each individual weak learner, the overall performance of the ensemble model was assessed in both the training and the test sets. These performance metrics were then meticulously logged to track the incremental improvements of the boosting process.

To effectively demonstrate the cumulative performance gains from integrating multiple weak learners, each individual weak learner was trained for 50 epochs. The maximum number of weak estimators in the ensemble was restricted up to 20. Specific termination conditions were established for the ensemble's boosting process as shown in Tab. 1.

Table 1: Ensemble boosting termination conditions for different weak learner types

Weak Learner Type	Termination when error rate exceeds
Logistic Regression	0.8
MLP	0.6
CNN	0.5

During the boosting process, each weak learner was trained using a Stochastic Gradient Descent (SGD) optimizer combined with a custom weighted cross-entropy loss function. This loss function dynamically adjusted sample weights, which is an essential mechanism of AdaBoost to emphasize misclassified examples in subsequent rounds. After each training phase, the error rate of each weak learner on the full training set was computed, and a corresponding weight (α) was assigned to reflect its influence in the final ensemble. The sample weights were then updated to further highlight previously misclassified instances, ensuring that the next weak learner focused on harder examples. The final prediction of the AdaBoost ensemble was obtained by computing a weighted sum of the logits produced by all trained weak learners, with each learner's contribution scaled by its respective α value.



(a) Training and test loss of AdaBoost on CIFAR-10 over training steps in log scale

(b) Training and test accuracy of AdaBoost on CIFAR-10 over training steps.

Figure 3: Training curves of AdaBoost on CIFAR-10.

Main Results. Fig. 3b illustrates AdaBoost performance on CIFAR-10, using Linear (10), MLP (5), and CNN (5) weak classifiers. The Linear model proved insufficient, showing low training and test accuracy due to poor feature extraction. While MLP improved training performance, generalization remained limited. A shallow CNN ultimately achieved 100% training accuracy, but test accuracy plateaued around 70%, highlighting AdaBoost's limited generalization capacity on complex datasets despite stronger base learners.

Fig. 3a summarizes AdaBoost's training and test loss (log scale) for these weak classifiers. The Linear model exhibited persistently high training and test losses, proving insufficient for CIFAR-10's complexity. MLP significantly reduced training loss (from 10^0), but its test loss remained high, indicating generalization challenges. The CNN achieved very low training loss (from 10^{-2}), perfectly fitting the training data, yet its test loss, starting around 10^1 and showing a slight increase, confirmed the model's ongoing difficulty in generalizing to unseen data.

A peculiar observation in the loss curves is the fluctuating behavior where some training losses appear to increase while others decrease as more weak classifiers are added. Specifically, for the Linear and CNN weak learners, the training loss tends to remain relatively stable or even slightly increase after an initial drop, while the MLP training loss shows a consistent decrease. This counterintuitive increase in training loss for certain weak learners, even as the ensemble grows, can be attributed to the nature of AdaBoost's re-weighting mechanism. At each iteration t, a new weak classifier $h_t(x)$ is trained on the training data using a weight distribution D_t. The weighted error is computed as:

$$\varepsilon_t = \sum_{i=1}^{n} D_t(i) \cdot \mathbb{I}\left(h_t(x_i) \neq y_i\right) \tag{2}$$

Then, the classifier's contribution to the final ensemble is determined by:

$$\alpha_t = \frac{1}{2} \log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) \tag{3}$$

Subsequently, the sample weights are updated:

$$D_{t+1}(i) = \frac{D_t(i) \cdot \exp\left(-\alpha_t y_i h_t(x_i)\right)}{Z_t} \tag{4}$$

where Z_t is a normalization factor. This update rule increases the weights of misclassified samples, forcing the next weak learner to focus on the "harder" examples. While this strategy aims to reduce the ensemble error, it can also make the training distribution increasingly challenging. As a result, individual weak learners may struggle to minimize loss under this re-weighted distribution, leading to fluctuations or even increases in observed training loss—even as overall accuracy improves. Therefore, the loss curve reflects not just model performance, but also the evolving difficulty of the training distribution. The overall trend, however, still points towards a more accurate and robust ensemble.

3.3 Convolutional Neutral Network (CNN)

Model Architecture. We adopt a modified version of the ResNet architecture adapted for CIFAR-10 images. The model begins with a convolutional stem followed by multiple residual blocks that progressively increase the number of channels while reducing the spatial dimensions. Each residual block includes two convolutional layers with batch normalization and GeLU [12] activations. The shortcut connection is implemented as an identity mapping when the input and output dimensions match, or as a 1×1 convolution with batch normalization when the number of channels or spatial resolution changes.The final output is average pooled and passed through a fully connected layer producing logits for the 10 CIFAR-10 classes.

This residual connection design allows gradients to bypass layers, addressing the degradation problem in deep networks . By encouraging identity mapping, ResNet improves convergence speed and final accuracy, and enables the training of deeper models without performance saturation.

Implementation details. We implement a 56-layer CNN based on the ResNet architecture, consisting of a stem convolution, three stages of residual blocks, and a final global average pooling followed by a fully connected classification layer. Each residual stage includes multiple basic blocks with batch normalization, GELU activation, and identity skip connections. The model is tailored to CIFAR-10 resolution and does not use bottleneck blocks.

The model is trained on CIFAR-10 using data augmentation strategies including random horizontal flip and random cropping. The optimizer is Adam with weight decay set to 1×10^{-4}. Instead of a fixed learning rate, we use a one-cycle learning rate policy with a maximum learning rate of 0.01 over 100 epochs. The batch size is 128.

Main Results. As shown in Tab. 2, the experiment results demonstrate that ResNet's skip connections are crucial for training deep models, preventing the performance degradation seen in plain CNNs and improving robustness to hyperparameter changes.

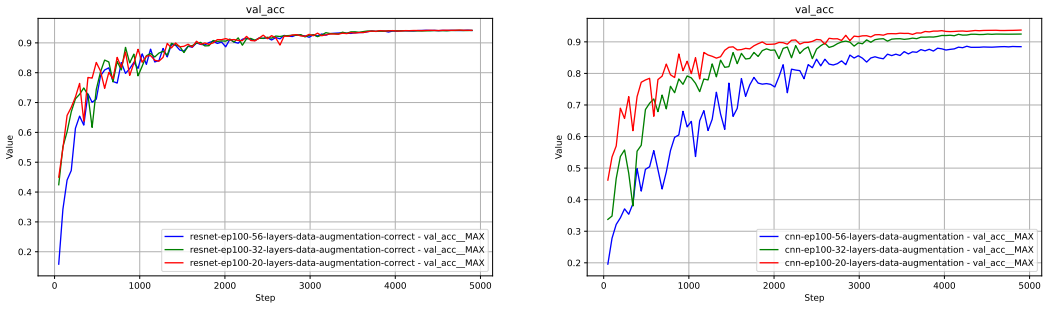
Table 2: Peak validation accuracy (%) for ResNet and Plain CNNs with/without data augmentation on CIFAR-10.

Architecture	Depth	With Data Augmentation	Without Data Augmentation
ResNet	20	94.1%	89.2%
	32	**94.3%**	89.5%
	56	94.2%	89.3%
Plain CNN	20	**93.7%**	85.1%
	32	92.5%	84.5%
	56	88.5%	82.3%

Additionally, in Fig. 4, ResNet models trained with data augmentation achieve a consistent validation accuracy of approximately 94% across all depths (20, 32, and 56 layers). The smooth convergence indicates effective generalization. In contrast, plain CNNs exhibit the classic degradation problem: while a 20-layer CNN reaches 93.7% accuracy, deeper models see performance drop significantly to 92.5% (CNN-32) and 88.5% (CNN-56). This highlights that simply adding layers to a plain network can hinder performance due to optimization challenges, such as vanishing gradients, which ResNet's skip connections are designed to mitigate.

The importance of data augmentation as a regularizer is also evident from Tab. 2. Without it, all models show signs of overfitting and achieve lower accuracy. ResNet's performance drops to around 89%, and the plain CNNs' peak accuracy falls to approximately 85%.

Further experiments reveal that the ResNet architecture provides greater training stability. When trained with various learning rates using the OneCycleLR scheduler (0.01 to 0.12), both ResNet-32 and ResNet-56 maintain stable training and consistent validation accuracy. Conversely, plain CNNs of similar depths (CNN-32 and CNN-56) exhibit significant performance fluctuations and instability, particularly at higher learning rates. This reinforces that ResNet's skip connections create a more tractable optimization landscape, improving robustness to hyperparameter settings.



(a) ResNet with Data Augmentation

(b) Plain CNN with Data Augmentation

Figure 4: Validation accuracy on CIFAR-10. (a) ResNet models maintain high accuracy as depth increases. (b) Plain CNNs show performance degradation with increased depth.

3.4 Vision Transformer (ViT)

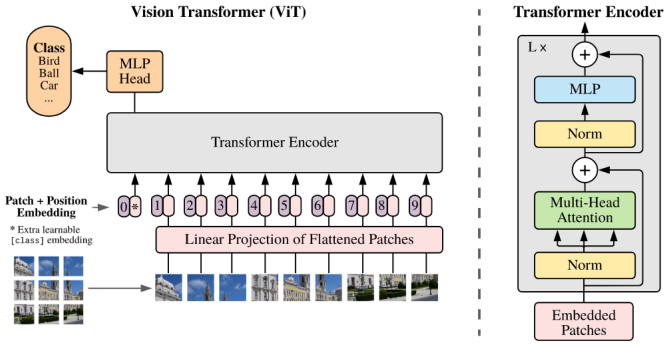


Figure 5: Vision Transformer Architecture

Model Architecture. We adopt the standard Vision Transformer (ViT) architecture for image classification, as the Fig. 5 showed. The input image of size (H, W, C) is first divided into non-overlapping square patches of size (P, P), resulting in $N = \frac{HW}{P^2}$ patches. Each patch is projected to a D-dimensional embedding vector via a convolutional layer with kernel size and stride equal to the patch size. These embeddings are concatenated with a learnable [CLS] token at the beginning and added with one-dimensional positional encodings to preserve spatial information.

The sequence of tokens is then passed through multiple Transformer encoder layers, each consisting of multi-head self-attention and MLP blocks. The final representation of the [CLS] token is used for classification. This token interacts with all patch tokens through attention, aggregating global information about the image. The design allows ViT to model long-range dependencies without relying on inductive biases such as locality or translation invariance, which are intrinsic to CNNs.

Implementation details. We experiment with two Vision Transformer configurations on the CIFAR-10 dataset. The first is a lightweight ViT variant with 6 transformer layers, 6 attention heads, embedding dimension 384, patch size 4, and MLP hidden size 1536. This model processes 32×32 images directly and has approximately 11 million parameters. The second is the standard ViT-Base/16 model with 12 layers, 12 heads, embedding dimension 768, patch size 16, and MLP size 3072. The input images are resized to 224×224, and the model contains about 86 million parameters.

Both models are trained from scratch on CIFAR-10 using the Adam optimizer with an initial learning rate of 1×10^{-4}, batch size of 128, and OneCycleLR scheduling. Data augmentation includes random cropping and horizontal flipping. For fine-tuning experiments, we pretrain ViT-Base/16 on ImageNet-1K (1.28M images, 1000 classes) and then finetune it on CIFAR-10 using a single linear classification head on top of the [CLS] token.

Main Results. We report results of two Vision Transformer configurations trained from scratch on CIFAR-10 (60,000 images, 10 classes). The small ViT variant (11M parameters) achieves a validation accuracy of about 82%, while the standard ViT-Base/16 (86M parameters) reaches around 85%(Fig. 6).

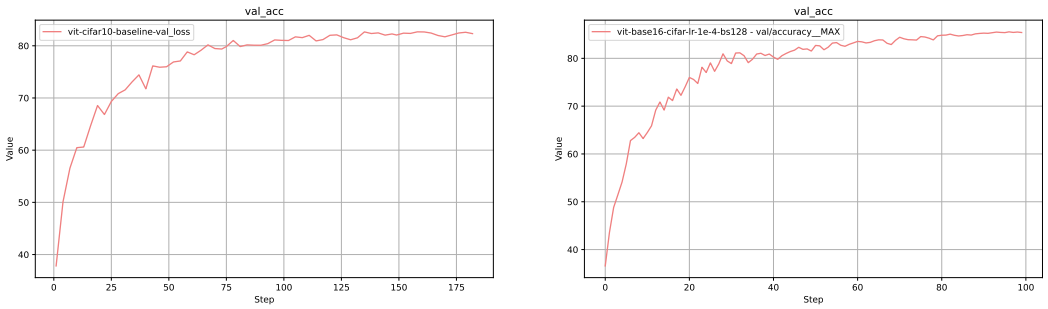


Figure 6: Validation accuracy of two ViT configurations

Both configurations underperform standard ResNet baselines (93–95% accuracy) when trained without pretraining. This aligns with prior findings [8] that Vision Transformers require large-scale datasets or pretraining to achieve competitive performance on smaller datasets.

In addition, we pretrain the standard ViT-Base/16 model on a subset ($\approx$ 880K images in 1K classes) of the ImageNet-1K dataset for 130 epochs. The model achieves a training accuracy of 84% and a validation accuracy of 56% on this subset, showing that it has learned valuable visual representations, though full convergence was not reached. We then fine-tune this pretrained model on CIFAR-10, replacing the classification head with a new linear layer for 10 classes and training the entire network. A discriminative learning rate strategy is applied, setting the learning rate of the classification head 10 times higher than that of the backbone. Using AdamW with warmup and decay scheduling, fine-tuning achieves a validation accuracy of 95% on CIFAR-10, demonstrating the significant performance boost provided by large-scale pretraining.

Table 3: Peak validation accuracy (%) for Vision Transformer models on CIFAR-10.

Model	**Training Setting**	**Peak VAcc**
Small ViT (11M params)	From scratch	82.0%
ViT-Base/16 (86M params)	From scratch	85.0%
ViT-Base/16 (86M params)	Pretrained on ImageNet subset + fine-tuned	**95.0%**

4 Discussion and Conclusion

Our experimental results provide several insights into the relative strengths and limitations of the models evaluated.

1. Boosting methods demonstrate that while ensembling can improve performance over weak learners, their effectiveness is bounded by the capacity of the base model and the complexity of the dataset. Even with CNN-based weak learners, AdaBoost struggles to generalize well, achieving limited gains in test accuracy.
2. ResNet architectures consistently outperform Vision Transformers (ViTs) and classical methods such as logistic regression and AdaBoost on CIFAR-10 when trained from scratch. This advantage is largely attributable to the inductive biases of CNNs (locality, translation invariance), which enable effective learning even on small-scale datasets.
3. Our experiments on learning rate sensitivity highlight the robustness of ResNet architectures. ResNet-32 and ResNet-56 maintain stable convergence across a range of learning rates, in stark contrast to plain CNNs of equivalent depth, which exhibit significant instability.

This underscores the critical role of skip connections in stabilizing training and facilitating optimization.

4. Vision Transformers, while conceptually powerful and capable of modeling long-range dependencies, underperform ResNet baselines without pretraining. The small ViT variant (11M parameters) and standard ViT-Base/16 (86M parameters) achieve validation accuracies of around 82% and 85%, respectively—well below ResNet-50's typical performance of 93–95% on CIFAR-10. This reinforces the findings of prior work [8] that ViTs require large-scale datasets or pretraining to reach their full potential.

In conclusion, our results offer a clear, practical perspective on the trade-offs between classical, convolutional, and transformer-based models for image classification. The experiments reaffirm that for datasets on the scale of CIFAR-10, ResNet's architecture provides a "sweet spot" of performance, stability, and efficiency. Its inherent inductive biases are well-suited for visual tasks, and its design effectively mitigates the optimization challenges of deep networks. Conversely, our findings caution against the direct application of Vision Transformers in data-scarce environments without a suitable pre-training strategy. This highlights a critical lesson: the theoretical potential of newer models like ViT does not always translate to superior performance in practice, and careful consideration of data requirements and computational overhead is essential for successful deployment.

References

[1] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[2] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '05)*, volume 1, pages 886–893. IEEE, 2005.

[3] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012.

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[9] Alex Krizhevsky, Geoffrey Hinton, and Balasubramanian Lakshminarayanan. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. Technical Report.

[10] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 2004.

[11] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[12] Dan Hendrycks and Kevin Gimpel. Gaussian Error Linear Units (GELUs). *arXiv preprint arXiv:1606.08415*, 2023.

Appendix

A EXPERIMENT DETAILS

This part details the hyperparameters used in the ViT experiments.

A.1 Common Hyperparameters and Techniques

The hyperparameters listed in Tab. 4 were consistently set across all the experimental setups. In our experiments, we also leveraged several advanced training techniques to enhance efficiency and stability, as detailed below.

Automatic Mixed Precision (AMP): We employed Automatic Mixed Precision (AMP) training, enabled via the `-use_amp` flag set to `True` across all experimental setups (Tab. 4). AMP combines 16-bit and 32-bit floating-point computations, significantly reducing memory usage and accelerating training on modern GPUs without compromising model accuracy.

Data Parallelism (DP): To further enhance training efficiency, we utilized Data Parallelism (DP) through the `-use_data_parallel` flag set to `True` (Tab. 4). DP distributes the input batch across multiple GPUs, with each GPU processing a subset of the data and computing gradients independently.

Gradient Clipping: Gradient clipping, implemented with `-grad_clip_norm=1.0` (Tab. 4), played a critical role in maintaining training stability. Gradient clipping constrains the L_2 norm of gradients to a maximum value of 1.0, preventing exploding gradients that could destabilize optimization. This was particularly important during ImageNet-1K pre-training, where the large batch size and complex data distribution increased the risk of gradient instability.

Table 4: Common hyperparameters shared across all experiments.

Argument	Value	Description
`model`	`ViT`	Model architecture used.
`image_size`	224	Input image size (pixels).
`patch_size`	16	Patch size (pixels).
`dim`	768	Transformer embedding dimension.
`depth`	12	Number of Transformer encoder blocks.
`heads`	12	Number of heads in multi-head attention.
`mlp_dim`	3072	Dimension of the MLP's inner hidden layer.
`enhanced_augmentation`	True	Use of enhanced data augmentation strategies.
`num_workers`	16	Number of worker processes for data loading.
`use_data_parallel`	True	Use `torch.nn.DataParallel` for multi-GPU training.
`use_amp`	True	Use Automatic Mixed Precision (AMP) training.
`grad_clip_norm`	1.0	Gradient clipping norm.

A.2 Experiment-Specific Hyperparameters

Tab. 5 provides a comparison of hyperparameters that varied across the different training configurations. To ensure clarity and adhere to standard formatting guidelines, the table is presented without scaling.

Table 5: Comparison of experiment-specific hyperparameters.

Argument	**Pre-train (ImageNet)**	**Fine-tune (CIFAR-10)**	**From Scratch (CIFAR-10)**	**Description**
`dataset`	`imagenet-1K`	`cifar-10`	`cifar-10`	Dataset used.
`bs` (batch_size)	1600	512	512	Batch size.
`ep` (epochs)	300 (130 actually)	20	100	Number of training epochs.
`lr` (learning_rate)	1.6e-3	2.0e-4	1.0e-4	Base learning rate (max LR).
`warmup_epochs`	6	3	10	Number of LR warmup epochs.
`warmup_start_lr`	1.0e-6	1.0e-6	1.0e-5	Warmup starting learning rate.
`min_lr`	1.0e-5	1.0e-6	1.0e-6	Minimum learning rate.
`dropout`	0.0	0.1	0.1	Dropout probability.
`weight_decay`	0.05	0.05	0.1	Weight decay coefficient.
`crop_padding`	28	28	4	Padding for random cropping pixels.