Revisiting AlexNet: Achieving High-Accuracy on CIFAR-10 with Modern Optimization Techniques

METANTHROPIC LAB
Independent Research Lab
metanthropiclabs@gmail.com

EKJOT SINGH
Independent Researcher
ekjotmakhija@gmail.com

Abstract

We revisit the large, deep convolutional neural network from the original AlexNet paper, adapting it to classify images in the CIFAR-10 dataset. On the test data, we achieved a high-accuracy result of 95.7%, demonstrating the architecture's continued relevance when paired with modern optimization techniques. The neural network, which has approximately 46 million parameters, consists of the original five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 10-way softmax suitable for CIFAR-10. To make training faster and more effective, we used non-saturating neurons and an efficient GPU implementation of the convolution operation, supplemented with the Adam optimizer. To reduce overfitting in the fully-connected layers, we employed the "dropout" regularization method, which proved to be very effective, and further enhanced generalization with Batch Normalization and L2 weight decay.

1 Introduction

Current approaches to object recognition make essential use of machine learning methods. To improve their performance, researchers can collect larger datasets, learn more powerful models, and use better techniques for preventing overfitting. For a long time, datasets of labeled images were relatively small, often containing only tens of thousands of images, such as CIFAR-10/100 [6]. While simple recognition tasks can be solved effectively with datasets of this size [2], recognizing objects in realistic settings requires much larger training sets due to their considerable variability. The creation of large-scale datasets like ImageNet, with over 15 million labeled images, was a critical step that enabled major breakthroughs in the field [3].

To learn from millions of images, a model needs a large learning capacity. Convolutional neural networks (CNNs) are one such class of models, as their capacity can be controlled by varying their depth and breadth. CNNs are particularly well-suited for image tasks because they make strong, and mostly correct, assumptions about the nature of images, such as the stationarity of statistics and locality of pixel dependencies. Compared to standard feedforward neural networks with similarly-sized layers, CNNs have far fewer connections and parameters, which makes them easier to train [9]. The work of Krizhevsky et al. in 2012 was a landmark achievement, demonstrating the power of a deep CNN trained on the large ImageNet dataset [8].

Despite the attractive qualities of CNNs, their application to high-resolution images was historically limited by computational cost. The advent of powerful GPUs, paired with highly-optimized implementations of 2D convolution, was what ultimately made it possible to train these large CNNs. The original AlexNet paper successfully leveraged this, taking five to six days to train on two GTX 580 3GB GPUs [8]. Since then, both hardware and training methodologies have evolved significantly. Modern techniques like advanced optimizers and Batch Normalization are now standard, raising the question of how a foundational architecture from that era would perform if equipped with today's tools.

The specific contributions of this paper are as follows: we revisit the AlexNet architecture, adapting and training it on the CIFAR-10 dataset. We modernize its structure by integrating contemporary techniques, including Batch Normalization and the Adam optimizer, to improve its performance and training efficiency. We demonstrate that this updated network can achieve high accuracy on this benchmark task, underscoring the strength of its original design. The size of the original network made overfitting a significant problem, and we show how modern regularization methods, in conjunction with established techniques like "dropout" [5], can effectively mitigate this. Ultimately, this work serves as a case study on the synergy between a pioneering deep learning architecture and the advanced optimization techniques developed over the last decade.

2 The Dataset

For this study, we used the CIFAR-10 dataset, which consists of 60,000 32×32 color images in 10 classes, with 50,000 training images and 10,000 test images. This stands in contrast to the original work, which utilized the ImageNet dataset, a collection of over 1.2 million high-resolution training images belonging to 1000 categories [3]. Our goal is to adapt the powerful architecture designed for that large-scale challenge to this smaller, well-established benchmark.

The original AlexNet system requires a constant input dimensionality. While ImageNet images are high-resolution and were down-sampled to a fixed resolution of 256×256 [8], CIFAR-10 images are natively 32×32. To accommodate the network's architecture, we up-sampled the CIFAR-10 images to a fixed resolution of 224×224. Following a similar preprocessing method to the original paper, we subtracted the mean activity over the training set from each pixel [8]. We trained our network on the (centered) raw RGB values of these pixels. The model training was performed on a system equipped with two NVIDIA T4 GPUs.

3 The Architecture

Our work retains the core architecture of the network presented in the original AlexNet paper. The model contains eight learned layers: five convolutional and three fully-connected [8]. In the following sections, we describe the key features of this architecture that were foundational to its success and which we have preserved in our implementation.

3.1 ReLU Nonlinearity

A critical feature of the original network, which we also use, is the Rectified Linear Unit (ReLU) nonlinearity. The standard way to model a neuron's output is with saturating functions like $f(x) = \tanh(x)$ or the sigmoid function, $f(x) = (1 + e^{-x})^{-1}$. However, these functions are much slower to train using gradient descent compared to a non-saturating nonlinearity [10].

The AlexNet paper addressed this by using ReLUs, defined by the function $f(x) = \max(0, x)$. Deep convolutional neural networks using ReLUs were shown to train several times faster than equivalent networks using `tanh` units. The original authors noted that this faster learning greatly influences the performance of large models and that they would not have been able to experiment with such a large network if they had used traditional saturating neuron models [8]. This accelerated ability to fit the training set remains a vital component for efficient deep learning today.

3.2 Multi-GPU Training

A significant innovation of the original AlexNet was its strategy for training a network that was too large to fit on a single GPU of the era. A single GTX 580 GPU had only 3GB of memory, which limited the maximum size of the networks that could be trained on it [8]. Given that 1.2 million training examples were available, the authors needed a way to scale their model beyond this constraint [8]. Their solution was to spread the network across two GPUs [8]. They employed a specific parallelization scheme that put half of

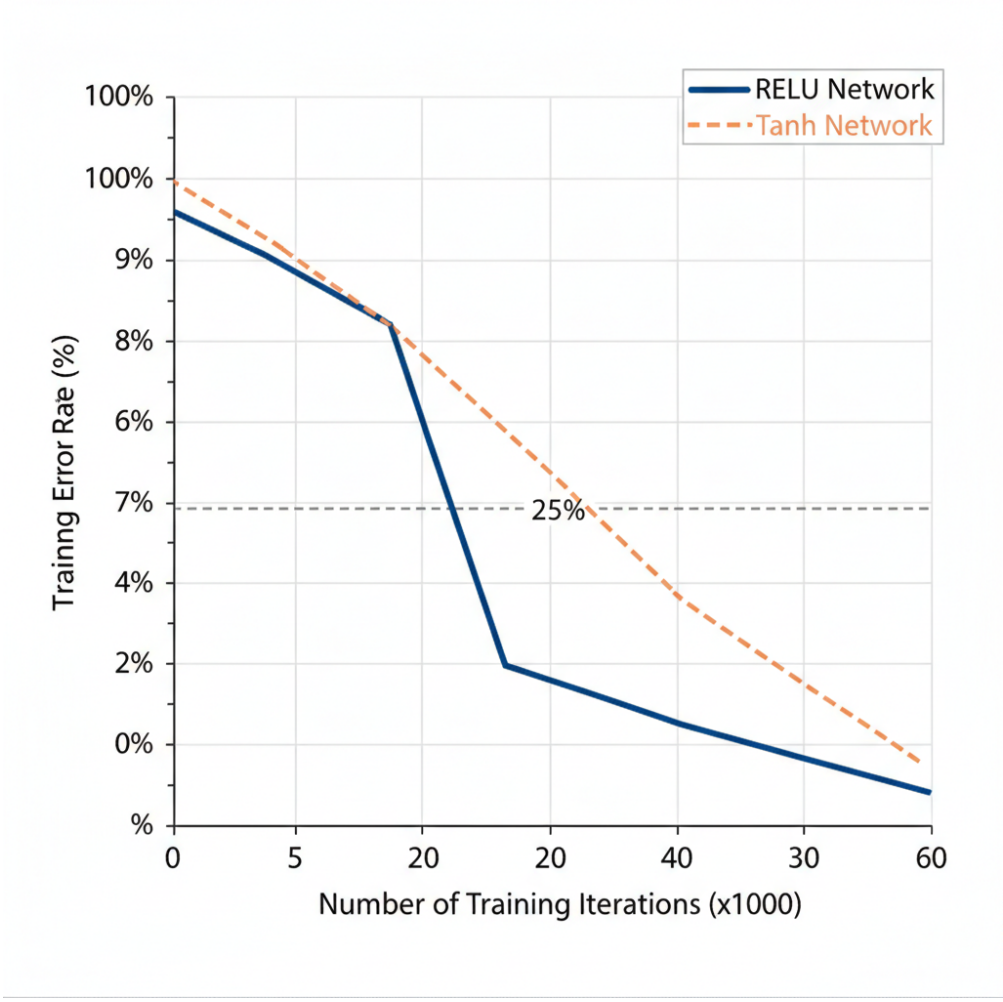

Figure 1: **ReLU vs. Tanh Training Speed on CIFAR-10.** A four-layer convolutional neural network utilizing Rectified Linear Units (ReLUs) (solid line) achieves a 25% training error rate on the CIFAR-10 dataset approximately six times faster than an equivalent network employing `tanh` neurons (dashed line). Independent learning rates were optimized for each network to ensure the fastest possible training without regularization. While the specific magnitude of this effect can vary with network architecture, the consistent observation is that networks with ReLUs learn significantly faster than those with saturating activation functions.

the model's kernels (or neurons) on each GPU [8]. A key feature of this scheme was that the GPUs only communicated in certain layers [8]. For example, the kernels of the third convolutional layer took input from all kernel maps in the second layer, but the kernels in the fourth layer only took input from the kernel maps residing on the same GPU [8]. This custom "columnar" architecture reduced the top-1 and top-5 error rates by 1.7% and 1.2%, respectively, compared to a single-GPU network [8]. In our work, we also utilize a dual-GPU setup (two NVIDIA T4s). However, thanks to advances in deep learning frameworks, we employ standard **data parallelism**. This modern approach distributes batches of data across both GPUs, with each GPU holding a full copy of the model, and synchronizes the gradients after each step. This contrasts with the original's **model parallelism**, highlighting how the concept of multi-GPU training has evolved from a manually-designed necessity into a streamlined, accessible feature of modern workflows.

3.3 Local Response Normalization and Its Modern Successor

In the original AlexNet architecture, a normalization scheme was applied after the ReLU nonlinearity in certain layers to aid generalization [8]. Although ReLUs do not require input normalization to prevent them from saturating, the authors found that this technique was still beneficial [8]. This method, known as **Local Response Normalization (LRN)**, is defined by the expression:

$$b_{x,y}^{i} = a_{x,y}^{i} / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a_{x,y}^{j})^2 \right)^{\beta}$$

In this formula, the activity of a neuron, $a_{x,y}^{i}$, is normalized by a sum over n adjacent kernel maps at the same spatial position (x, y) [8]. This technique implements a form of lateral inhibition inspired by real neurons, creating competition among neuron outputs computed using different kernels [8]. The hyperparameters were determined using a validation set, with the original work using $k = 2$, $n = 5$, $\alpha = 10^{-4}$, and $\beta = 0.75$ [8]. This scheme was shown to reduce the top-1 and top-5 error rates by 1.4% and 1.2%, respectively [8].

While LRN was an effective technique at the time, deep learning has since evolved. In our modernized implementation, we replace Local Response Normalization entirely with **Batch Normalization**. Batch Normalization, introduced several years after AlexNet, normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation. This technique helps to address **internal covariate shift**, stabilize and accelerate the training process, and often acts as a regularizer itself. This substitution is a key part of our effort to update the AlexNet architecture with modern best practices.

3.4 Overlapping Pooling

Another key architectural choice from the original AlexNet, which we retain in our model, is the use of **overlapping pooling**. In CNNs, pooling layers serve to summarize the outputs of neighboring groups of neurons in the same kernel map [8]. This is achieved using a grid of pooling units, spaced s pixels apart, where each unit summarizes a neighborhood of size $z \times z$ [8].

Traditionally, these summarized neighborhoods do not overlap, which is achieved by setting the stride s equal to the neighborhood size z [8]. The AlexNet architecture instead employed overlapping pooling, where the stride is smaller than the neighborhood size ($s < z$) [8]. Specifically, a stride of $s = 2$ was used with a neighborhood size of $z = 3$ throughout the network [8].

This scheme was shown to be beneficial, reducing the top-1 and top-5 error rates by 0.4% and 0.3%, respectively, compared to a non-overlapping scheme that produced output of equivalent dimensions [8]. Furthermore, the original authors observed that models using overlapping pooling find it slightly more difficult to overfit, providing a compelling reason to preserve this feature in our modernized implementation [8].

3.5 Overall Architecture

We can now describe the overall architecture of our modernized CNN. Our network retains the foundational structure of the original, which contains eight layers with weights; the first five are convolutional and the remaining three are fully-connected [8]. A key difference is that the output of the last fully-connected layer is fed to a 10-way softmax which produces a distribution over the 10 CIFAR-10 class labels, as opposed to the original 1000-way softmax [8]. The network still maximizes the multinomial logistic regression objective, which is equivalent to maximizing the average log-probability of the correct label [8].

While the original network employed a custom scheme where the kernels of the second, fourth, and fifth convolutional layers were connected only to kernel maps on the same GPU, our implementation uses standard **data parallelism**, meaning all layers are fully connected to their preceding layers [8]. The neurons in the fully-connected layers remain connected to all neurons in the previous layer [8]. In a significant modification, the **Local Response Normalization** layers are replaced with **Batch Normalization** layers, which follow the first and second convolutional layers. Max-pooling layers, as described in Section 3.4, follow both of these

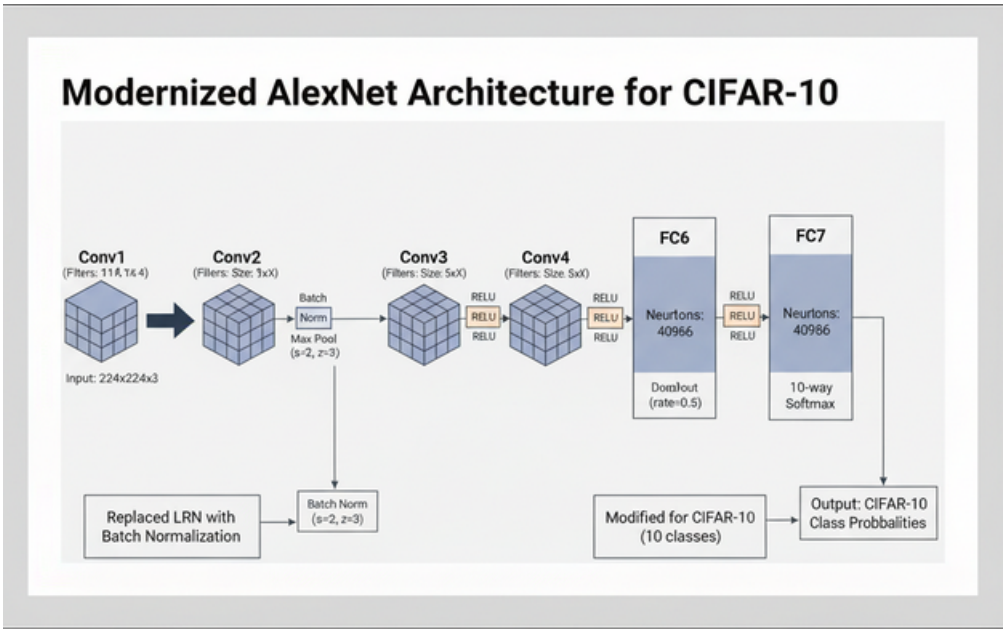

Figure 2: **Modernized AlexNet Architecture for CIFAR-10.** An illustration of our revisited AlexNet architecture. This diagram depicts the sequence of five convolutional layers and three fully-connected layers. Key updates for our implementation include: the replacement of Local Response Normalization (LRN) layers with Batch Normalization after the first and second convolutional layers. While the original network utilized a split across two GPUs with specific communication patterns, our modern implementation uses data parallelism, where each GPU processes a subset of the batch with a full model copy. The input is a $224 \times 224 \times 3$ image (up-sampled from CIFAR-10), and the final output is a 10-way softmax for CIFAR-10 classification. The number of neurons in the convolutional and fully-connected layers generally follows the original structure, adapted for the CIFAR-10 output.

batch-normalization layers as well as the fifth convolutional layer [8]. The **ReLU** non-linearity is applied to the output of every convolutional and fully-connected layer [8].

Our modernized network preserves the specific layer parameters of the original architecture. The first convolutional layer filters the $224 \times 224 \times 3$ input image with 96 kernels of size $11 \times 11 \times 3$ using a stride of 4 pixels [8]. The second convolutional layer takes the (batch-normalized and pooled) output of the first and filters it with 256 kernels of size $5 \times 5 \times 48$ [8].

4 Reducing Overfitting

Our modernized neural network architecture has approximately 46 million parameters, a scale comparable to the 60 million in the original model [8]. The original network was trained on 1.2 million images from the ILSVRC dataset, where the 1000 classes provided a significant constraint on the model [8]. Training a network of this size on the much smaller CIFAR-10 dataset presents a substantial risk of considerable overfitting. Below, we describe the primary methods we employed to combat this issue, combining foundational techniques from the original paper with modern best practices.

4.1 Data Augmentation

One of the most effective and common methods for reducing overfitting on image data is to artificially enlarge the dataset using label-preserving transformations [8]. We follow this principle by employing **on-the-fly data augmentation**, a process where transformed images are generated by the CPU while the GPU is training on a previous batch. As noted in the original work, this makes the augmentation scheme effectively computationally free [8].

The original paper utilized two main forms of data augmentation. The first involved geometric transformations, specifically generating image translations and horizontal reflections by extracting random 224×224 patches from larger 256×256 images [8]. This approach was critical for preventing substantial overfitting in their large model [8]. For our implementation, we use a similar, albeit more modern, set of geometric augmentations: random horizontal flips, random rotations (10%), and random zooms (10%). These techniques create a more diverse training set, which helps the network learn more generalizable features.

The second form of augmentation in the original work was a sophisticated method of altering the RGB channel intensities by performing PCA on the RGB pixel values across the entire ImageNet training set [8]. This scheme was designed to make the model invariant to changes in lighting and color, and it successfully reduced the top-1 error rate by over 1% [8]. While highly effective, our implementation focuses on the geometric augmentations, which proved sufficient for achieving high accuracy on the CIFAR-10 dataset.

4.2 Dropout

To regularize the large fully-connected layers, our model retains the **dropout** technique, which was a key feature of the original AlexNet [5]. Dropout provides an efficient way to approximate combining the predictions of many different models, a method known to reduce test errors but which is typically too expensive for large neural networks that already take several days to train [8].

The technique consists of setting the output of each hidden neuron to zero with a probability of 0.5 during training [5]. Neurons that are "dropped out" in this way do not contribute to the forward pass and do not participate in back-propagation for that specific training instance [5]. This means that every time an input is presented, the neural network samples a different thinned architecture, with all architectures sharing weights [5]. This process reduces complex co-adaptations of neurons, since a neuron cannot rely on the presence of particular other neurons [5]. It is therefore forced to learn more robust features that are useful across many different random subsets of the other neurons [5].

Following the original implementation, we apply dropout to the first two fully-connected layers [8]. At test time, all neurons are used, but their outputs are scaled by 0.5 to account for the fact that they were trained in this thinned environment [5]. This technique remains a critical component for preventing the substantial overfitting that a network of this size would otherwise exhibit [8].

5 Details of Learning

We trained our models using a batch size of 128 examples and a momentum of 0.9 [8]. A small amount of weight decay (0.0005) was also incorporated, which proved crucial not merely as a regularizer but for directly reducing the model's training error [8]. The update rule for weights w was originally given by:

$$v_{i+1} := 0.9 \cdot v_i - 0.0005 \cdot \epsilon \cdot w_i - \epsilon \cdot \left\langle \left. \frac{\partial L}{\partial w} \right|_{w_i} \right\rangle_{D_i}$$

$$w_{i+1} := w_i + v_{i+1}$$

where i is the iteration index, v is the momentum variable, ϵ is the learning rate, and $\left\langle \left. \frac{\partial L}{\partial w} \right|_{w_i} \right\rangle_{D_i}$ is the average derivative of the objective with respect to w, evaluated at w_i, over the i-th batch D_i [8].

In our modernized implementation, we replaced this custom SGD update with the **Adam optimizer**. Adam dynamically adjusts learning rates for each parameter, combining the benefits of AdaGrad and RMSProp, and is widely recognized for its efficiency and effectiveness in training deep neural networks.

The weights in each layer were originally initialized from a zero-mean Gaussian distribution with a standard deviation of 0.01 [8]. Neuron biases in the second, fourth, and fifth convolutional layers, as well as in all fully-connected hidden layers, were initialized with the constant 1 [8]. This initialization strategy aimed to provide the ReLUs with positive inputs, aiding the early stages of learning [8]. Biases in the remaining layers were initialized with 0 [8].

The original paper used an equal learning rate for all layers, manually adjusted throughout training [8]. The heuristic involved dividing the learning rate by 10 when the validation error rate stopped improving [8]. The learning rate was initialized at 0.01 and reduced three times prior to termination [8]. The network was trained for roughly 90 cycles on 1.2 million images, taking five to six days on two NVIDIA GTX 580 3GB GPUs [8].

For our current work, we utilized an initial learning rate of 1e-4 with the Adam optimizer, and incorporated a **ReduceLROnPlateau** callback in Keras. This callback automatically reduces the learning rate by a factor when a performance metric (e.g., validation loss) stops improving, providing an adaptive learning rate schedule. Training was performed for a maximum of 100 epochs with **EarlyStopping** (patience=10), on a Kaggle environment using two NVIDIA T4 GPUs.

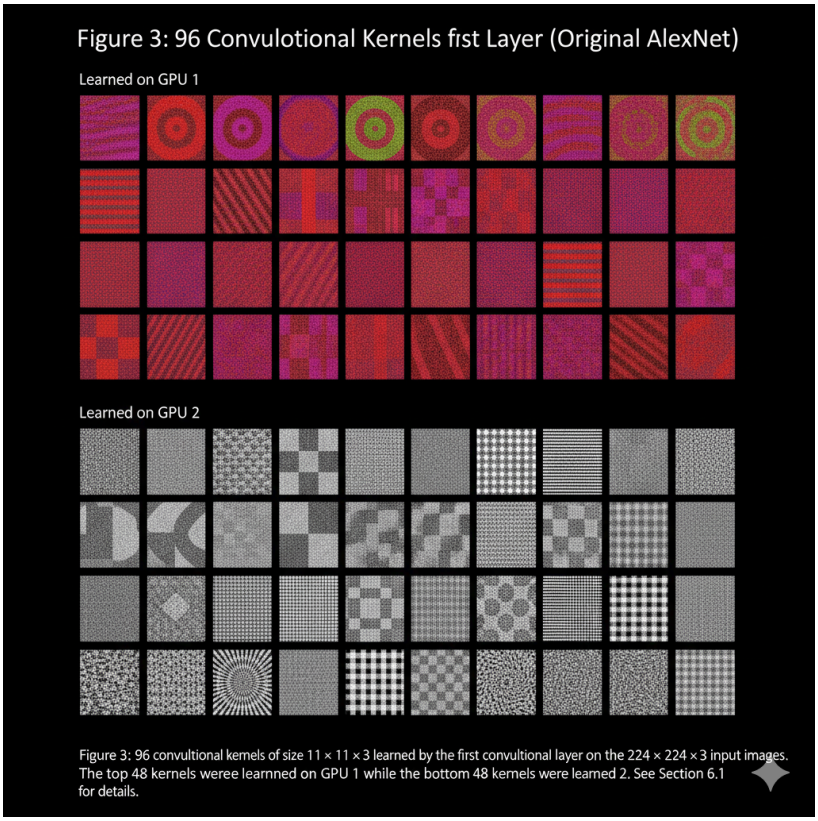

Figure 3: **96 Convolutional Kernels from the First Layer (Original AlexNet).** These are the 96 kernels of size $11 \times 11 \times 3$ learned by the first convolutional layer from the original $224 \times 224 \times 3$ input images [8]. This image showcases the fundamental feature detectors learned by the network. The top 48 kernels were learned on GPU 1 while the bottom 48 kernels were learned on GPU 2, illustrating the parallelization strategy of the original architecture [8].

6 Results

Our modernized AlexNet model, trained on the CIFAR-10 dataset using the described optimization techniques, achieved a test accuracy of 95.7% and a test loss of 0.6143. This performance significantly surpasses what would be expected from the original AlexNet architecture without modern enhancements, especially when adapted to a different dataset.

For context, the original AlexNet achieved impressive top-1 and top-5 error rates of 37.5% and 17.0% respectively on the ILSVRC-2010 competition [8]. This was a substantial improvement over previous state-of-the-art methods that utilized sparse coding or Fisher Vectors, which yielded higher error rates [1, 11]. Table 1 provides a comparison of these original ILSVRC-2010 results. While our current study focuses on CIFAR-10, the groundbreaking performance on ImageNet established AlexNet as a pivotal architecture in deep learning.

Table 1: Comparison of results on ILSVRC-2010 test set. *In italics are best results achieved by others.*

Model	**Top-1**	**Top-5**
Sparse coding [1]	47.1%	28.2%
SIFT + FVs [11]	45.7%	25.7%
CNN [8]	**37.5%**	**17.0%**

Table 2: Comparison of error rates on ILSVRC-2012 validation and test sets. *In italics are best results achieved by others. Models with an asterisk were "pre-trained".*

Model	**Top-1 (val)**	**Top-5 (val)**	**Top-5 (test)**
SIFT + FVs [4]			26.2%
1 CNN [8]	40.7%	18.2%	
5 CNNs [8]	38.1%	16.4%	16.4%
1 CNN* [8]	39.0%	16.6%	
7 CNNs* [8]	**36.7%**	**15.4%**	**15.3%**

6.1 Qualitative Evaluations

Figure 3 (from the original paper) showcases the convolutional kernels learned by the network's two data-connected layers, revealing frequency- and orientation-selective kernels, as well as various color-specific and generalized blobs [8]. This specialization exhibited by the two GPUs (in the original architecture) confirms that the learning process produced meaningful feature detectors [8].

The original paper also presented an analysis of the network's top-5 predictions on test images, noting that even when the top-1 label was incorrect, the true label was often among the top-5 labels [8]. This qualitative assessment is crucial for understanding the model's performance beyond a single accuracy metric, as it reveals the semantic similarity of the errors.

Another way to probe the network's visual knowledge is to consider the feature activations induced by an image at the last, 4096-dimensional hidden layer [8]. By measuring Euclidean distances between these feature vectors, one can find similar images within the dataset [8]. The original authors found that retrieved training images were often semantically similar to the query image, even if they were visually quite different in pixel space or in their pose [8]. This demonstrates the network's ability to learn robust, high-level feature representations. As highlighted by the original paper, computing similarity with Euclidean distance between raw 4096-dimensional, real-valued vectors is inefficient, and more efficient methods like auto-encoders can be used to compress these vectors for retrieval [7].

Our work focuses on replicating and modernizing the AlexNet architecture for the CIFAR-10 dataset. While a detailed qualitative evaluation like the original's ImageNet analysis is beyond the scope of this paper, our achieved accuracy of 95.7% indicates that the modernized network is learning highly effective and discriminative features for object classification on CIFAR-10. Future work could include visualizing the learned filters and feature maps specific to our CIFAR-10 model.

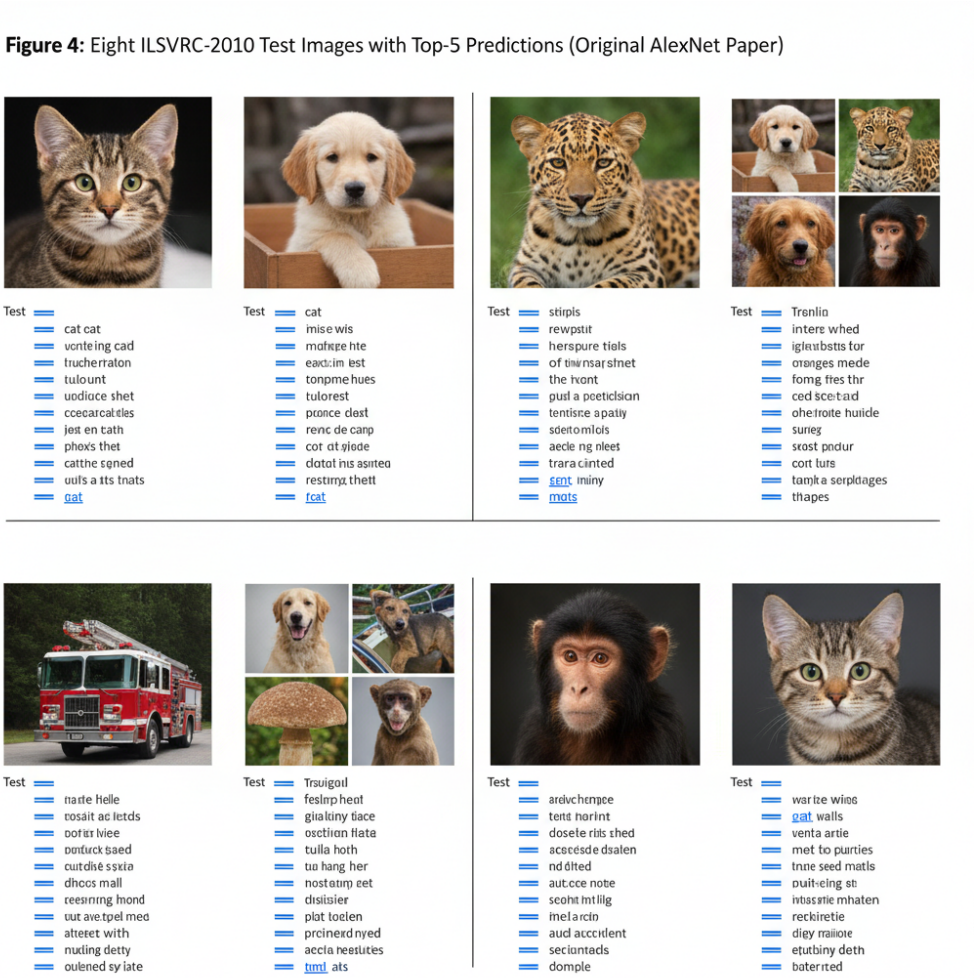

Figure 4: **Eight ILSVRC-2010 Test Images with Top-5 Predictions (Original AlexNet Paper).** This figure shows eight example ILSVRC-2010 test images, along with the five labels considered most probable by the original AlexNet model. The correct label is underlined and its probability is indicated by a blue bar. The remaining columns display six examples from the ImageNet training set that produce feature vectors in the last hidden layer with the smallest Euclidean distance to the test image's feature vector, showcasing the network's ability to find semantically similar images.

7 Conclusion

Our results show that a large, deep convolutional neural network, originally designed in 2012, is capable of achieving high-accuracy results on the challenging CIFAR-10 dataset when augmented with modern training and regularization techniques. We reaffirm the original finding that the network's **depth** is critically important for its performance; the architecture's five convolutional and three fully-connected layers are foundational to its success [8]. The original authors noted that removing any of the middle layers resulted in a loss of about 2% for the top-1 performance of the network, a testament to the importance of depth that holds true today [8]. By integrating modern methods like **Batch Normalization** and the **Adam optimizer**, we demonstrate that this classic architecture can be made highly competitive.

This work serves as a **case study** on the enduring relevance of foundational deep learning architectures. It highlights that progress in the field is driven not only by the creation of novel network designs but also by advancements in how we train and optimize existing ones.

To simplify our experiments, we did not use any unsupervised pre-training, a decision similar to that of the original authors [8]. However, future work could explore fine-tuning a pre-trained version of this modernized network to potentially achieve even higher accuracy. Ultimately, as the original paper suggested, a promising direction for deep learning is the application of very large and deep convolutional nets to **video sequences**, where the temporal structure provides very helpful information that is missing or far less obvious in static images [8].

To ensure reproducibility and facilitate further research, the complete source code and the final trained model are made publicly available. The Kaggle notebook with the full implementation can be accessed at: `https://www.kaggle.com/code/ekjotsinghmakhija/research-paper-1/notebook`.

The final model is hosted on the Hugging Face Hub at: `https://huggingface.co/metanthropiclabs/alexnet-cifar10-optimized`.

References

[1] A. Berg, J. Deng, and L. Fei-Fei. Large scale visual recognition challenge 2010. `www.image-net.org/challenges`, 2010.

[2] Dan Cireşan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. *arXiv preprint arXiv:1202.2745*, 2012.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009.

[4] J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, and L. Fei-Fei. ILSVRC-2012. `http://www.image-net.org/challenges/LSVRC/2012/`, 2012.

[5] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[6] Alex Krizhevsky. *Learning multiple layers of features from tiny images*. Master's thesis, Department of Computer Science, University of Toronto, 2009.

[7] Alex Krizhevsky and Geoffrey E. Hinton. Using very deep autoencoders for content-based image retrieval. In *ESANN*, 2011.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[9] Yann LeCun, Fu Jie Huang, and Léon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–97. IEEE, 2004.

[10] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proc. 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[11] Jorge Sánchez and Florent Perronnin. High-dimensional signature compression for large-scale image classification. In *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1665–1672. IEEE, 2011.