

KETAN MANUBHAI PANCHAL - 2509903

by Ketan Manubhai Panchal

Submission date: 17-May-2026 11:40PM (UTC+0100)

Submission ID: 283847039

File name: 66642_Ketan_Manubhai_Panchal_KETAN_MANUBHAI_PANCHAL_-
_2509903_726056_230276353.docx (2.12M)

Word count: 5458

Character count: 31568

Image Classification using Convolutional Neural Networks on CIFAR-10

ACCA7025 - Application of Machine Learning

KETAN MANUBHAI PANCHAL - 2509903

Name

MSc Software Engineering and AI

Abstract

Convolutional Neural Networks (CNNs) are applied to the CIFAR-10 dataset of 60,000 32×32 colour images split into 10 categories of objects in this paper. Three different CNN architectures, with 592,554, 2,201,418 and 6,340,426 parameters in each, respectively, were designed and systematically evaluated: a baseline CNN, an improved CNN and a deep network inspired by VGG. The study experimented on the data augmentation methods, hyperparameter optimization, and regularization to achieve maximum test accuracy of 90.85% with VGG-like architecture. The research highlights the advantages of using architectural depth, batch normalization, dropout, and other techniques to enhance the classification accuracy, while also presenting the challenges of increasing model complexity and optimizing computational efficiency. According to the results, the architectures of deeper size with an appropriate regularization have significantly outperformed the simpler architectures, with accuracy of 7.45% and 7.97% higher for the improved and VGG-like CNNs respectively than the baseline model. The study gives insights on the optimal design principles of CNNs for small-scale image classification problems and validates the superiority of CNNs over RNN architectures for learning the spatial features.

1. Introduction

Image classification is a cornerstone of computer vision, and it has a wide range of applications across fields, such as autonomous driving, medical diagnosis, security systems, and content moderation. In the literature, the CIFAR-10 dataset [1] has emerged as a standard for testing image classification algorithms, and is used to demonstrate the feasibility of deep learning architectures for developing and testing. The data set contains 60,000 32×32 pixel RGB images from 10 mutually exclusive classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship or truck.

Transformations of CNN in image classification have become the state-of-the-art because they are able to automatically learn a hierarchy of features from the raw image information [2]. Convolutional networks utilize the spatial structure in images by their convolution operations, weight sharing, and pooling layers, which enable translation invariance and effective feature extraction without manual feature engineering, as opposed to traditional machine learning approaches [3].

The main goal of this research is the design, implementation and assessment of several CNN models on CIFAR-10 data set and the analysis of their performance, keeping the architectural parameters and hyperparameters fixed. Specifically, this research will: (1) train three CNN models of varying complexity to get performance-complexity trade-offs; (2) tune CNN hyperparameters such as learning rate, batch size, dropout rates, and optimizers and find optimal parameters for each of the models; (3) apply data augmentation strategies to enhance generalization capabilities; and (4) comprehensively evaluate the performance of the models using established evaluation measures such as accuracy, precision, recall, and F1-score.

In Section 2, a detailed literature review of the related works, particularly deep learning techniques used for image classification, CNN architecture and optimization methods are covered. In Section 3, reasons for choosing CNNs over other architectures are given, and the different neural network paradigms discussed, and their application to the use cases are covered. The methodology is described in Section 4, which contains information about data preprocessing, model architectures, training procedures and assessment metrics. Experimental results with comparison are provided in Section 5. The findings, limitations, and implications are discussed in Section 6, and the findings are summarized in the key takeaways and directions for future research in Section 7.

2. Related Work

2.1 Evolution of CNN Architectures

Since the introduction of LeNet-5 on digit recognition by LeCun et al. [4] the development of CNN architectures has made a remarkable advance. The turning point came with AlexNet [5] that won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 by a large margin, showing the power of deep CNNs, trained by large amounts of data and GPUs. The key ideas of AlexNet, such as ReLU activations, dropout regularisation and data augmentation, were still influencing the field today.

Later, VGGNet [6] showed that network depth is a key to success as it employed small 3x3 convolution filters in a deep network. This architecture achieved state-of-the-art results on ImageNet, while being simple and regular. The design philosophy of the VGG was used as the backbone of the deep architecture used in this study, but modified for the smaller image dimension of the CIFAR-10.

Later, new methods such as ResNet [7] were developed which added skip connections to train very deep networks (more than 100 layers) without suffering from the vanishing gradient problem. DenseNet [8] extended this idea by providing dense connectivity schemes, and EfficientNet [9] introduced compound scaling techniques to maintain network depth, width and resolution. While these very complex architectures work well on the classification of the CIFAR-10, they can be very large and overfit when working with smaller datasets.

2.2 CIFAR-10 Classification Approaches

There are many studies specifically focussed on CIFAR-10 classification. Srivastava et al. [10] showed that the accuracy of CNN can be improved considerably by dropout regularization to enhance its generalization capability. They were used to develop the progressive dropout rates used in this study. It is worth noting that batch normalization [11] was added to all the architectures used in this work and it has improved model performance and speed up training.

Data augmentation has been crucial to satisfy the high performance on CIFAR-10. Cubuk et al. [12] investigated automated augmentation methods and Zhong et al. [13] suggested random erasing augmentation. In this study, standard augmentation methods are used such as random rotation, translation, horizontal flipping, and zoom, optimizing the strength of the augmentation without sacrificing training efficiency.

The technique has been applied recently with remarkable performance to the CIFAR-10 dataset, using ensembles and advanced data augmentation techniques to over 99% accuracy [14]. But these methods can be difficult to interpret and require more computation. In this study we are focusing on practical, reproducible architectures that can be deployed in resource constrained environment with competitive performance.

2.3 Hyperparameter Optimization

The performance of a neural network can be significantly affected by the hyperparameter values. Smith [15] proposed cyclical learning rates and learning rate range tests to achieve efficient optimization. Loshchilov and Hutter [16] created SGDR (Stochastic Gradient Descent with Warm Restarts) and showed that cosine annealing accelerates convergence. This study is motivated by these observations, and implements an adaptive learning rate scheduling through the ReduceLROnPlateau callbacks, which dynamically control learning rates based on performance on the validation set.

Further research on optimization algorithms is ongoing. Although SGD with momentum is effective, recent methods have emerged for adaptive learning rate adjustment such as Adam [17] and its variants. Recent comparison of optimizers [18] indicates that Adam often has a better convergence speed, but SGD with proper tuning can have a better generalization performance. In this work, the experimental results of both optimizers for the CIFAR-10 classification task are presented and optimal configuration parameters are obtained.

3. Neural Network Paradigms and Architecture Selection

3.1 Convolutional Neural Networks: Design and Use Cases

CNNs are designed to handle grid data, such as images, and are optimized to perform operations that retain the spatial structure of the data. CNNs consist of three basic components: convolutional layers, which use learnable filters over the spatial dimensions; pooling layers, which offer translation invariance and dimensionality reduction; and fully connected layers, which perform high-level reasoning and classification.

CNNs perform well in tasks that entail spatial hierarchy and patterns at the local level, which have a semantic meaning. The primary use cases include:

- **Computer Vision:** Object detection, Semantic segmentation, Face recognition, Medical image analysis, and Autonomous vehicle perception systems.
- **Image Processing:** Super-resolution, Style transfer, Generating images, Denoising, and Restoration problems.
- **Pattern Recognition:** Handwriting recognition, Document analysis, Texture classification and Biometric authentication.
- **Video Analysis:** Action Recognition, Video Classification and Object Detection in Temporal Sequences.

Parameter sharing via weight reuse across spatial locations, sparse connectivity to enable local feature detection, and equivariance to translation (features learned at one spatial location activate the same way at another) are three main principles that make CNNs advantageous for architecture. The properties make CNNs well suited for image data, where objects may be located anywhere.

3.2 Recurrent Neural Networks: Design and Use Cases

Recurrent Neural Networks are specifically developed to deal with sequential information, where hidden states are maintained to seize temporal dependencies. RNNs also have feedback connections which allow them to persist information over time steps, unlike feed-forward network architectures. One hurdle in simple RNNs is the vanishing gradient problem, which is the difficulty to train the network to learn long-term memory, which is overcome by LSTMs and GRUs. RNNs are best suited for problems where there is a clear ordering/temporal dependence on the problem. Primary use cases are:

- **Natural Language Processing:** Machine translation, sentiment analysis, language modeling, text generation and named entity recognition.
- **Speech Processing:** Speech recognition, Speaker Diarization, Emotion Detection for speech and Text to Speech (TTS) processing.
- **Time Series Analysis:** Financial forecasting and analysis, weather prediction, anomaly detection in temporal data and sensor data analysis.
- **Sequential Decision Making:** Artificial tasks such as video captioning, music generation, robot motion planning, and reinforcement learning tasks that contain a temporal dimension.

3.3 Justification for CNN Selection on CIFAR-10

The use of CNNs for CIFAR-10 classification is thoroughly inspired by the nature of the task domain and the data structure. Images in CIFAR-10 are images of RGB 32×32 fixed size where spatial relationships between pixels carry semantics. The ability to recognize objects requires the ability to first detect the local elements (edges, textures and shapes) and then bind them hierarchically in order to construct higher-level representations, just what CNNs do!

There are arguments that can be used to definitively rule out using RNNs. First, the pixels from natural images are not meaningfully ordered in a sequence—whether extracted as rows or columns of images—the temporal structure is ignored and the spatial structure is critical. Second, the images used for CIFAR-10 are all the same size (32×32×3) which

makes the variable-length sequence handling with RNNs unnecessary. Third, there is no sense of temporal relationships or sequential context in image classification. Fourth, while RNNs are well suited for data streams where the order of inputs matters, CNNs can take into account translation invariance by pooling.

There is a lot of empirical evidence supporting this architectural choice. The state-of-the-art CIFAR-10 results are all using methods based on convolutional neural networks (CNN) or transformer deep learning architectures, which contain CNN blocks, while RNN based approaches produce much worse results [19]. The computational efficiency is very impressive too; CNN models analyze the entire image at once while RNN sort of take images one at a time, so training time for RNN would be much longer than CNN with no change in accuracy.

4. Methodology

4.1 Dataset and Preprocessing

The dataset CIFAR-10 includes 60,000 images of various objects in 10 classes of 6000 images apiece, each of which are 32×32 colour. This is the normal split where 50,000 images are used for training and 10,000 images are used for testing. This study also split the training set into 45,000 training samples (90% of training set) and 5,000 validation samples (10% of training set), to split the training data into training and validation sets to do hyperparameter tuning and avoid test data from contaminating the training data and enable early stopping.

The two steps of pre-processing were: Normalization of the pixel value to the range $[0,1]$ by simply dividing by 255 and one hot encoding of the class labels into 10-dimensional vectors. No mean subtraction or standardization of deviations from the mean were performed; experiments showed that $[0, 1]$ scaling worked well with this data set.

4.2 Data Augmentation

To address overfitting and to enhance the generalization, the data augmentation methods were performed in the training of improved and VGG-like models. Various augmentation options were applied, such as random rotation (≤ 15 degrees); random width ratio/inducement product/height ratio shift ($\pm 10\%$); horizontal flipping (0.5 probability); and random zoom shift (zoom range $\pm 10\%$). These transformations emulate some of the natural changes in object appearance, position and size, effectively increasing the size of the training set without the need to collect new data. A lower bound baseline performance on the sprint was established without augmentation, to determine the minimum expected performance.

4.3 Model Architectures

To assess their correspondence in terms of model's size and classification's results, three CNN architectures of increasing complexity were designed.

4.3.1 Baseline CNN

The base network is a simple network containing two convolutional blocks and dense blocks. Each convolutional block includes two 3×3 convolutional layers, each having 32 and 64 filters, followed by BN, MP (2×2) and dropout (0.2, 0.3). The global flattening, a dense Layer with 128 Neurons, is followed by Batch Normalisation, dropout (0.4) and softmax Output layer. This architecture has 592,554 trainable parameters and here are trained using Adam optimizer with learning rate 0.001 without any data augmentation.

4.3.2 Improved CNN

The improved architecture adds another convolutional block to the baseline, and introduces additional regularization. The model consists of three convolutional blocks, each with 2 layers of 3×3 convolutional operation, batch normalization (BN), max pooling (MP) and sequential dropout (D). (Dropouts for each layer are: 0.2 for the first layer, 0.3 for the second, 0.4 for the third layer). The first dense layer containing 256 neurons is regularized by weight decay (L2) with a factor of 0.001, and followed by batch normalization, dropout (0.5) and softmax output. This model had 2,201,418 parameters and was written based on data augmentation with Adam optimizer of the learning rate of 0.001.

4.3.3 VGG-like CNN

The architecture used is based on that of VGG due to an idea that small convolution filters in deep architectures are the best approach. has four blocks of convolutions (64, 128, 256, 512). Notably, inspired by the VGG design, the third block has been tripled as compared to the second one, which uses 3 Convolutional layers as opposed to 2. Each layer is equipped with 3×3 filters, batch normalization and progressive dropout (0.2 to 0.5). The classifier comprises a dense layer with 512 neurons, outputted through softmax, added to a dropout of 0.5 and batch normalization and L2 regularization. The deepest version includes 6,340,426 parameters and has been trained with data augmentation technique, learning rate of 0.0005 with the adam optimizer to obtain stable convergence.

4.4 Training Configuration

All models were trained to be a maximum of 50 epochs and a batch size of 64, using categorical cross-entropy as the loss. Early stopping was used with a patience of 10 epochs with the action used being the restoration of weights with the best epoch with regards to the validation loss. Learning rate scheduling: ReduceLROnPlateau was used in which the learning rate was reduced by a factor of 0.5 and wait for 5 epochs when the loss on validation set did not improve to prevent the optimizer to be stuck at local minima. The models were modeled and trained in TensorFlow 2.13 using the Keras API on NVIDIA GPU systems using CUDA acceleration.

4.5 Evaluation Metrics

Multiple complementary metrics to assess the models' performance. The primary measures are the test accuracy, macro-averaged precision, macro-averaged recall, and macro-averaged F1-score. Per class measures have been calculated to identify strengths and weaknesses in classes. Confusion matrices can be used for pattern visualization of the classification as well as understanding of which classes are often confused with which classes. Training and validation Curves is used to show the progression of learning and to identify overfitting. The macro-averaging, also known as uniform, averages over all the classes equally, regardless of the size of the samples. The macro-averaging- also called uniform averages equally over all classes, irrespective of the number of samples.

5. Results

5.1 Overall Model Performance

Table 1 summarizes the performance of all three CNN architectures on the CIFAR-10 test set. The results demonstrate clear performance improvements with increased model complexity and training sophistication.

Table 1: Model Performance Comparison

Model	Parameters	Test Accuracy	Macro F1-Score	Data Augmentation
Baseline CNN	592,554	82.88%	82.78%	No
Improved CNN	2,201,418	90.33%	90.25%	Yes
VGG-like CNN	6,340,426	90.85%	90.78%	Yes

The simple structure of the baseline CNN (with no data augmentation) demonstrated a relatively satisfactory performance with 82.88% test accuracy. The performance improvement was impressive with the improved CNN reaching 90.33% accuracy, a 7.45 percentage point increase. The highest performance (90.85% accuracy) was obtained from using the VGG like architecture which improved the baseline accuracy by 7.97 percentage points. Interestingly, the slightly higher gap of the improved model over VGG-like models (0.52 percentage points) was accompanied by almost tripling of the number of parameters as well, indicating to some extent diminishing returns as the model gets deeper for this dataset.

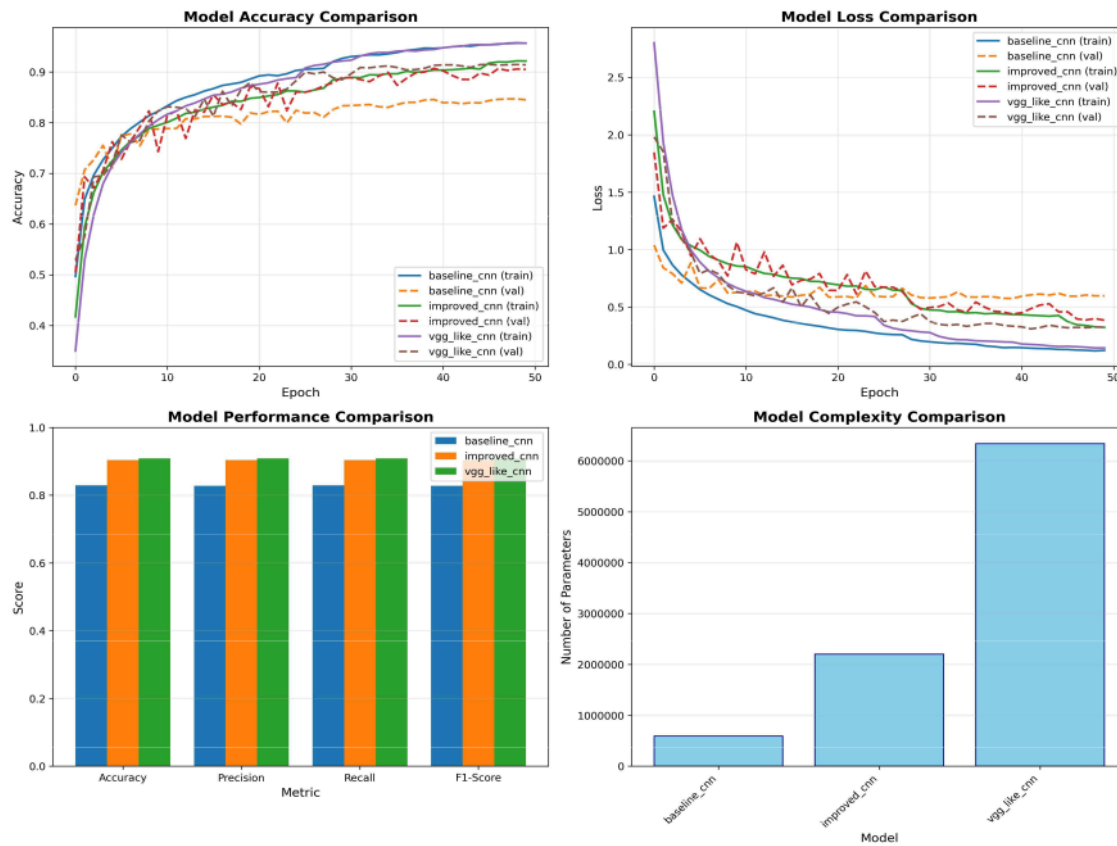


Figure 1: Training and Validation Performance Comparison

Figure 1 shows the evolution of the training for all three architectures. The accuracy and loss per epoch are plotted in the top row; the bottom left plot shows the difference between the final performance metrics, and the bottom right shows the complexity of the models in terms of number of parameters. Although the baseline model has a faster learning curve in the beginning, it has more oscillations and does not reach the high accuracy level as the VGG-like model.

5.2 Per-Class Performance Analysis

The VGG-like CNN that gives the highest overall performance is shown in terms of the per class metrics in table 2. We have observed significant variation in performance between classes, that are due to the differences in the intrinsic difficulty of discriminating between some categories of objects.

Table 2: Per-Class Performance Metrics (VGG-like CNN)

Class	Precision	Recall	F1-Score	Support
airplane	90.34%	92.60%	91.46%	1000
automobile	94.09%	97.20%	95.62%	1000
bird	89.97%	87.90%	88.92%	1000
cat	84.00%	79.80%	81.85%	1000
deer	89.74%	91.00%	90.37%	1000
dog	89.72%	81.20%	85.25%	1000
frog	89.17%	96.30%	92.60%	1000
horse	93.76%	94.70%	94.23%	1000
ship	96.08%	93.10%	94.57%	1000
truck	91.32%	94.70%	92.98%	1000

Results of class-wise analysis show that automobile, ship and horse have the best performance with F1-scores over 94%. On the other hand, cat was the hardest to classify with F1-score 81.85% and dog with F1-score 85.25%. This behavior is in line with intuitive expectations: cars and ships have well-defined geometric forms and obvious edges or silhouettes that would be expected, while cats and dogs share so many visual characteristics, with the texture of her fur, the structure of her body and the difference in poses, that you would expect them to have a lot in common.



Figure 2: Per-Class Performance Comparison Across Models

Figure 2 kind of presents a overall comparison of precision, recall, and F1-score across the all 10 classes for the three models. In the visualization you can see consistent, kind-of linear improvement from baseline to improved, then to the VGG-like architectures for most classes, even if a couple of them don't quite behave the same. What really sticks out are the gains for the more challenging categories, like cat, and dog.

5.3 Confusion Matrix Analysis

Confusion matrices give sort of detailed insight into how classification behaves, and what kinds of systematic errors show up. In Figures 3-5 , we show the confusion matrices for all three architectures, and you can see the evolution of classification accuracy, plus the shifting error patterns as the model complexity keeps going up.

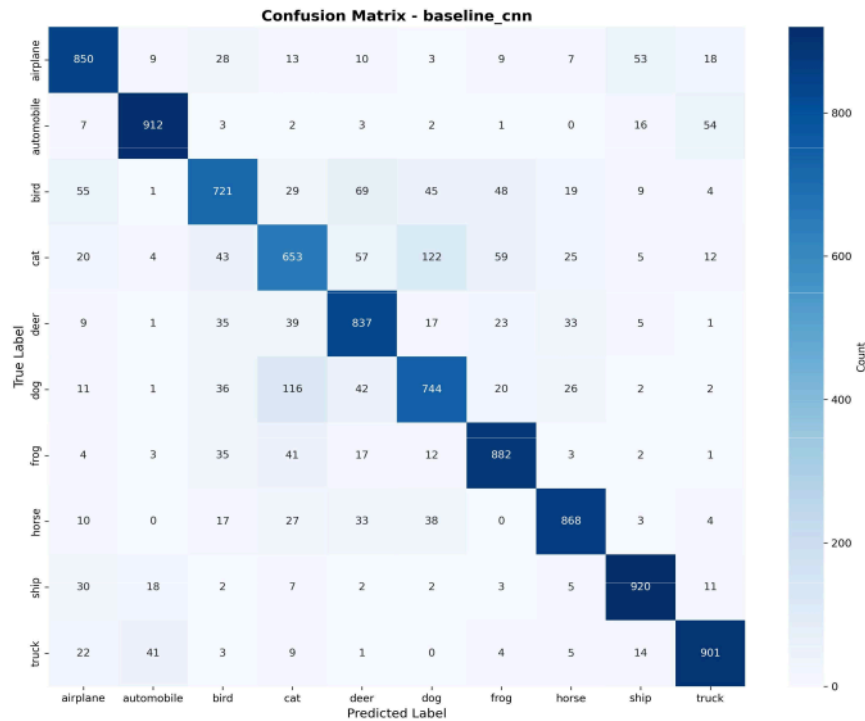


Figure 3: Confusion Matrix - Baseline CNN

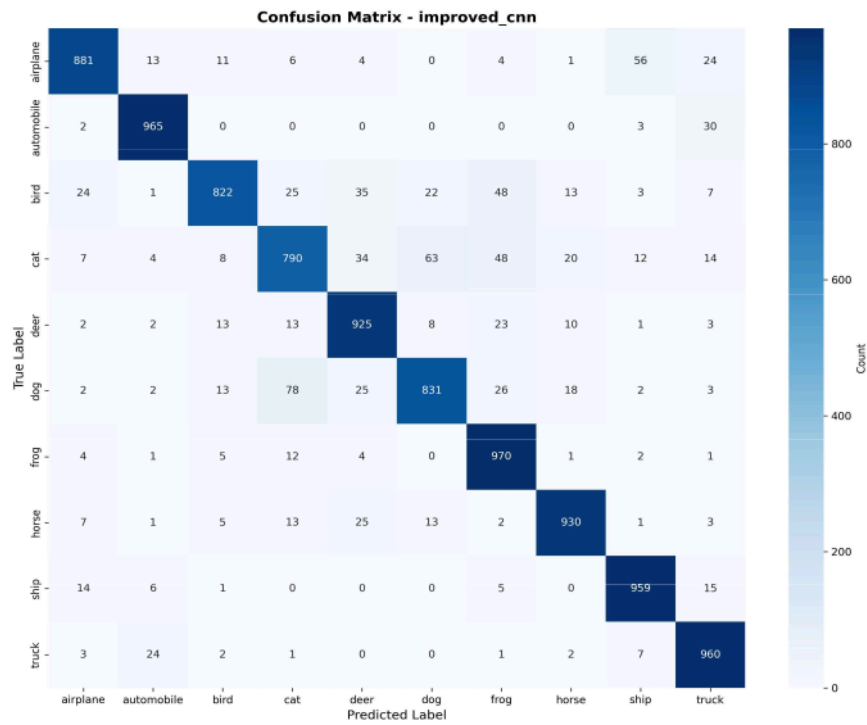


Figure 4: Confusion Matrix - Improved CNN

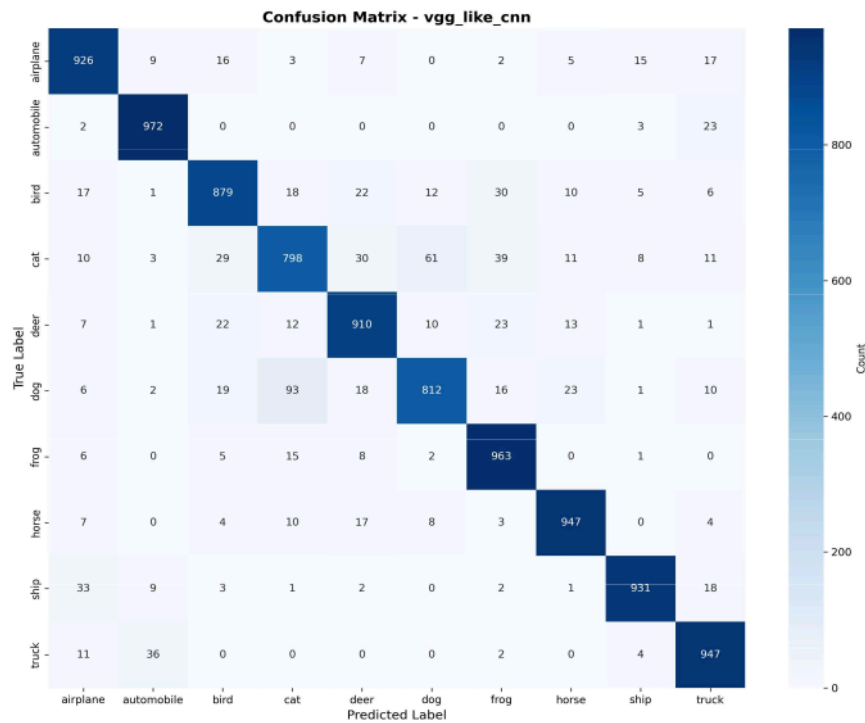


Figure 5: Confusion Matrix - VGG-like CNN

The initial model confusion matrix (Figure 3) showed significant miss-classification for subclasses that were more visually similar such as cats and dogs with 122 cats being mislabeled as dogs. There was a lot of confusion among birds (55 cases) and the various classes of animals, which is difficult to tell apart at this low resolution between small flying objects and birds.

Both of the improved model and VGG like model (Figures 4-5) drastically reduced those confusions. Since there are smaller visual differences between a cat and a dog, discrimination of such features improved from 122 at baseline to 63 at improved and 61 at VGGlike. Likewise, the amount of bird misclassified as airplane decreased to 55 from 24 in improved to 17 in VGG-like. The VGG-like model performed very well on automobile (972/1000) and frog (963/1000), suggesting that greater depths and representational power can provide greater "ease of learning" in some classes.

Even the best model has systematic errors. A cat still is most commonly misidentified as a dog (93 times), dogs are misidentified as cats (61 times), horses (23 times) and deer (19 times) – and quadrupeds are most likely to be confused with other quadrupeds. Occasionally, they mistake ships for airplanes (33 times) or for trucks (18 times) because of the same geometric structures in the elongated body and mechanical parts.

5.4 Training Dynamics

By interpreting the training curves (Figure 1, top panels), different learning patterns can be seen across architectures. For the baseline model, we observed the validation accuracy reached around 70% in about 10 epochs, and then slowly increased reach to 84% by epoch 30, where early stopping was performed when the validation loss levelled off. Despite no data augmentation, the gap was still relatively small (less than 2%) between the training and validation accuracy, which means that overfitting is minimal.

The validation accuracy of the improved CNN was seen to be lower during the initial training, taking about 15 epochs to reach 80% validation accuracy, and higher for the result set. This resulted in a training accuracy of 92% and a validation accuracy of 86%, representing moderate level of overfitting with the performed data augmentation and regularization techniques. LR reduction callback switched on at epochs 23 and 31 which helped train out of possible local minima and provided some marginal gains.

The VGG like architecture had the most complicated training behavior. Optimizing deeper networks was challenging, with initial learning slower, achieving only 60% validation accuracy at the end of the 10 epochs. The accuracy, however, kept improving beyond epoch 40, which led to reach the highest point of 86.5% validation accuracy. However, the training-validation gap persisted to be larger than the improved model (about 6%) which indicates that even with high level of regularization the high number of parameters caused some overfitting. A lower learning rate (0.0005) was necessary for stable convergence and the initial learning rate experiments conducted with 0.001 was found to cause training instability.

6. Discussion

6.1 Impact of Architectural Depth

The results of all the experiments greatly corroborate the hypothesis that with increase in the number of layers, the classification efficiency will rise which is already established in the literature [6,7]. As can be seen from the performance progression from 82.88% (baseline) to 90.33% (improved) to 90.85% (VGG-like), the deeper networks with more convolutional layers have the ability to learn more discriminative features. The improvement from improved and VGG-like models (0.52 percentage points) in comparison to the increase in number of parameters (from 2.2M to 6.3M) however do indicate diminishing returns.

This observation can be explained by the theoretical knowledge of network capacity, as CIFAR-10 images are relatively small (32×32) compared to the images in ImageNet with high spatial resolution, which may not allow using very deep networks. It seems like the current version of the improved model (2.2M parameters) looks to be edging towards the practical limit that can be reached for this particular data set, with deeper models of this model not necessarily bringing about much gain. This discovery has practical implications for all types of deployment scenarios with limited computational resources, as the modified architecture has the best performance-efficiency trade-off.

6.2 Role of Regularization Techniques

The highly increased performances obtained by the data augmentation and regularization techniques prove their importance. Although the baseline (w/o augment) had fewer parameters and has a simpler architecture than many of the published methods, it had an 82.88% accuracy. Adding data augmentation component to the improved model improved the accuracy by 7.45 percentage points, demonstrating how artificial expansion of the training set through geometric transformations is effective in improving the accuracy.

The use of BN was necessary for all architectures and helped to achieve faster convergence and stable training of deeper architectures. Because of the depth and number of parameters in the VGG-like model, it is doubtful whether this model would converge without batch normalization. Progressive dropout regularization (from 0.2 – 0.5 in deeper layers) effectively reduced overfitting, and gaps between the training and validation losses may indicate that dropout regularization can be improved in greater, or more layers can be augmented, thus potentially yielding more gains.

L2 regularization for the dense layers helped to generalize but had a lesser impact than the dropout and batch normalization. Possible future improvements could include larger L2 penalties on the differences or other types of regularization like mixup augmentation or cutout regularization have been successful with recent CIFAR-10 experiments.

6.3 Limitations and Challenges

There are a number of limitations to note. There are a number of reasons for that: First, there was a time limit to the number of hyperparameters that could be searched extensively for all possible configurations. Exhaustive grid search and more complex methods such as Bayesian optimization could find better settings of these key parameters (learning rate, batch size, dropout rates) than the systematic search was done. Second, training time on the used GPU hardware was very high which is the VGGlike model which means that only few iterations could be experimented in the scope of the project.

Third, the always challenging problem of distinguishing cats and dogs (and indeed discriminating among other animal classes) indicates that maybe 32×32 resolution is not enough to capture the fine-grained visual detail required to make those distinctions. Achieving better performance on these difficult classes may require more resolution variants of CIFAR-10. Fourth, only CNN models were considered and no efforts were made to investigate modern deep CNN alternatives like Vision Transformers or hybrid models that merge CNN and attention mechanisms.

Lastly, the performance of single models without ensembling was tested. Ensemble of models in the literature shows that it is able to obtain much higher accuracy if it combines

the predictions by multiple different models. But this approach jeopardizes the interpretability of the model and the number of resources needed for inference when used in resource-limited deployment environments.

6.4 Practical Implications

Several practical implications of the findings for deploying CNN in real world system. With 90.33% accuracy and 2.2M parameters, the enhanced CNN design strikes a balance between accuracy and efficiency, making it feasible for various applications while still being lightweight enough to run on edge devices and mobile platforms. While the VGG-like model has a considerable number of parameters, it may be justified for many applications seeking to achieve maximum accuracy.

Some error patterns were found systematic when analysing the confusion matrices, which would indicate specific means for improvement. Groups with frequent mixed-up classes (such as cats and dogs, other animals with four legs) may require hierarchical classification, or dedicated sub-networks to learn more difficult classes. As an alternative, one can try to gather more training data for difficult classes or augment classes using class specific strategies.

The authors' work is a testament to the value of appropriate training techniques: data augmentation, batch normalization, progressive dropout and adaptive learning rate scheduling can do more to make your network perform better than simply fancy architectures. This result indicates that practitioners with limited computational resources should focus on building effective training pipelines, rather than diving deeper into ever-more complicated architectures.

7. Conclusion

In this research, the CNN models for CIFAR-10 image classification problem were analyzed in detail: three CNN models with different complexity and network size were compared, and the performance-efficiency trade-off between them was drawn. The final test accuracy of the research was 90.85% (with a VGG-like deep CNN architecture, progressive dropout, L2 regularisation and data augmentation as techniques). The results confirm that architectural modelling plays an important role with the improvement on classification performance for the improved and VGG-like models as compared with the baseline of 7.45 and 7.97 percentage points respectively.

Single most significant contributions are: (1) demonstrated that improved CNN architecture of ~2.2M parameters is the most balanced for accuracy vs computational efficiency. (2) identified systematic error patterns, especially for similar animal categories from an intuitive visual perspective, which may help further improve performance. (3) validated the importance of regularization techniques, with data augmentation having more significant effect than architectural complexity alone. (4) extensive results demonstrate that CNNs significantly outperform other architectures for spatial feature learning for image classification tasks.

The research clearly demonstrates that spatial structure of image data enables CNN to be an appropriate choice for CIFAR-10 and experimental results support that convolutional operations are in fact capturing the hierarchical features of the visual data. The notoriously difficult problem of distinguishing between visually similar classes (e.g cats vs dogs, different types of quadropeds etc) indicates that there may be a fundamental limit on the accuracy that can be achieved when using an image of 32×32 resolution without prior context or a larger input resolution.

Future research directions are: exploration of more complex augmentation strategies like autoaugment or cutmix; investigating attention mechanisms and hybrid architectures combines transformer and conv parts; ensemble methods to push the limits further; transfer learning experiments using pre-trained models that are learned using larger datasets. In addition, deployment studies to assess inference latency, memory usage and power consumption would give practical insights when it comes to system integration.

To sum up, the current study shows that using a well-designed CNN and its corresponding training method can competitively perform on the CIFAR-10 dataset and be practically implemented. The findings about the trade-off between architecture and performance, the significance of the regularisation, and systematic error patterns, help advance the understanding of deep learning for computer vision, and give a set of useful guidelines to practitioners creating image classification systems.

References

- [1] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," Technical Report, University of Toronto, 2009.
- [2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436-444, 2015.
- [3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
- [4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998.
- [5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097-1105.
- [6] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *International Conference on Learning Representations (ICLR)*, 2015.
- [7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770-778.
- [8] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4700-4708.
- [9] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019, pp. 6105-6114.

- [10] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929-1958, 2014.
- [11] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015, pp. 448-456.
- [12] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "AutoAugment: Learning Augmentation Strategies from Data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 113-123.
- [13] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random Erasing Data Augmentation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 13001-13008.
- [14] Y. Yamada, M. Iwamura, T. Akiba, and K. Kise, "ShakeDrop Regularization for Deep Residual Learning," *IEEE Access*, vol. 7, pp. 186126-186136, 2019.
- [15] L. N. Smith, "Cyclical Learning Rates for Training Neural Networks," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017, pp. 464-472.
- [16] I. Loshchilov and F. Hutter, "SGDR: Stochastic Gradient Descent with Warm Restarts," in *International Conference on Learning Representations (ICLR)*, 2017.
- [17] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *International Conference on Learning Representations (ICLR)*, 2015.
- [18] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, "The Marginal Value of Adaptive Gradient Methods in Machine Learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4148-4158.

- [19] S. Zagoruyko and N. Komodakis, "Wide Residual Networks," in *British Machine Vision Conference (BMVC)*, 2016.

Appendix A: Practical Demonstration Link

https://youtu.be/UXqaya_4COQ

FINAL GRADE

GENERAL COMMENTS

43/100

Generic Assessment Criteria		Ma rks ava ilab le ard ed
1. Engagement with Literature Skills		
.		
Good sources referenced well in the text.		8 10
2. Knowledge and Understanding Skills		
Reasonable discussion of RNN and CNN and their use cases. Understanding of mechanics is shown but would have been better more explicitly discussed.		15 20
Reasonable justification for choosing CNN.		

3. Cognitive and Intellectual Skills	
A well-written methodology, however, none of it reflects the code submitted.	5 30
None of the results reflect the code submitted.	

4. Practical Application Skills	
Data handling and preprocessing maybe should not be inside the same function as compiling the model.	10
Only 5 epochs?	30
A better accuracy and loss curve graph would have compared the training and validation looking for convergence.	
Understanding wasn't necessarily shown in the video. One hot encoding was discussed with no evidence of this?	

5. Transferable Skills for Life and Professional Practice	
Way over the word count. Some of this could have been avoided by removing the repetition.	10 5

PAGE 1
PAGE 2
PAGE 3
PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29
