

ACCA7025 – Application of Machine Learning

Image Classification Using Convolutional Neural Networks on the CIFAR-10 Dataset

Deep Patel (2504308)

UNIVERSITY OF WALES TRINITY SAINT DAVID - Wales Institute of Science Arts – School
of Applied Computing

Table of Contents

Abstract	3
1. Introduction	3
2. Related Work.....	4
2.1 Convolutional Neural Networks for Image Classification	4
2.2 Recurrent Neural Networks and Sequence Modelling	5
2.3 Optimization and Regularization	5
3. Workflow Methodology	5
3.1 Dataset and Pre-processing	5
3.2 Data Augmentation	6
3.3 Model Architectures	7
3.4 Evaluation Metrics	8
4. Model Performance Results	9
5. Discussion and Recommendation	10
5.1 Effect of Optimiser Choice	10
5.2 Batch Normalization and Dropout	11
5.3 Recommendations	11
6. Conclusion	11
References	13
Appendix: Video demonstration link	15

List of Tables

Table 1: Summary of Model Configurations.....	8
Table 2: Final Test Performance Metrics by Model	10

List of Figures

Figure 1: Sample images from the CIFAR-10 dataset with class labels	6
Figure 2: Model 1 training and validation loss, accuracy, precision and recall across epochs.....	9

Abstract

The objective of this report is to explore how the Convolutional Neural Networks (CNNs) can be applied to multi-class image classification on the CIFAR-10 dataset which contains 60,000 colour images in 10 different classes. Four CNN architectures of increasing complexity were designed, trained and evaluated in the TensorFlow/Keras framework, each one a single hyperparameter or structural variation. The first model, using three convolutional blocks with batch normalization, dropout regularization and data augmentation, obtained a test accuracy of 71.74%, precision of 79.31% and a recall of 64.51%. Comparative variants investigated the effectiveness of the SGD optimizer, the L2 regularization and the use of further convolutional layers. This benchmark shows that the Adam optimizer, batch normalization and some dropout delivers the best generalization. The report also provides a background on neural network optimization and examples of their application in practice, specifically in Recurrent Neural Networks and Convolutional Neural Networks.

1. Introduction

Automatic recognition and classification of visual content has emerged as one of the major problems in the field of machine learning. This has led to the emergence of deep learning, with CNNs in particular, as the most successful paradigm for this kind of tasks, consistently outperforming traditional feature-engineering based methods on the classic benchmarks [9]. The CIFAR-10 dataset, created by [7], has been a standard test set for assessing image classification algorithms because it is small, has a variety of classes, and is difficult because of low-resolution images (32x32 pixels).

The goal of this investigation is to: (1) implement a variety of classifiers with the CIFAR-10 dataset in Python, TensorFlow and Keras; (2) systematically change the hyperparameters and structural components of the CNN and study the impact on their performance; and (3) test each of the classifiers by using a set of metrics such as accuracy, precision, recall and loss. The report is organized to place the study in the context of the available literature, to present the methodology and pipeline that was used, to present and discuss the results, and to conclude by synthesizing results and

making generalizations. In particular, they are relevant for problems where data has a high dimension and is structured, for example images, sequences or graph. Special architectures have been found to work very well for images and for sequential data, respectively, the Convolutional Neural Network (CNN) and the Recurrent Neural Network (RNN) [1]. It is important to know when each architecture is best suited to use, as part of model selection.

2. Related Work

2.1 Convolutional Neural Networks for Image Classification

Modern deep image classification was initiated in 1998 by LeNet, who showed that learned spatial hierarchies of image features work better for digit recognition than handcrafted features. Authors in [8] submitted the AlexNet to the ImageNet Large Scale Visual Recognition Challenge, which was the clear winner with deep CNNs, ReLU activations and dropout regularization. This work introduced several concepts that are widely used today such as max-pooling, overlapping convolutions, data augmentation etc. Subsequent studies by [11] showed that network depth is a crucial part of classification performance, and they found that stacking small (3×3) convolutional filters is an effective way to increase the depth of the receptive field. Researchers in [3] took this one step further and added Residual Networks (ResNets) to alleviate vanishing gradients issues in very deep networks. These architectures were trained on ImageNet but have exhibited generalization to smaller datasets like CIFAR-10 [3].

On the CIFAR-10 dataset, [13] showed that an architecture without any pooling layers can match or even outperform any max-pooling-based architecture, thereby questioning the need for spatial down sampling. Recently, [15] proposed a uniform scaling approach through compound scaling, which resulted in EfficientNet with state-of-the-art performance with a relatively small computational cost. The following insights lead to the architectural choices of the present investigation: the stacking of convolutional blocks and the implementation of batch normalization as introduced by [5] which helps to avoid internal covariate shift and is able to increase learning rates.

2.2 Recurrent Neural Networks and Sequence Modelling

Recurrent neural networks are designed to process sequentially structured inputs for which the information flow is maintained at the hidden state, which is passed on across time steps [4]. Long Short-Term Memory (LSTM) networks overcome the vanishing gradient problem of conventional RNNs by incorporating gated memory cells, which are suitable for applications like natural language processing, speech recognition, and time-series prediction [2]. One interesting application of the RNNs is sentiment classification: An RNN trained on sequences of word embedding can learn about the dependencies between words across the structure of the sentence which cannot be learned by a feed-forward model. In contrast, the same task could be accomplished using a CNN-based classifier which would model local n-gram features with convolutional filters, without the ability to integrate over time like an RNN. The architectural compromise points to the need to match the selection of the model with the structural characteristics of the input domain.

2.3 Optimization and Regularization

One of the key areas of study is optimizing neural network training. The Adam optimizer, developed by [6], also uses momentum estimation and adaptive learning rates and hence is very useful for a wide variety of tasks and is less sensitive to manual learning rate tuning than Stochastic Gradient Descent (SGD). Researchers in [14] proposed dropout as a regularization method to randomly set activations to zero during training to prevent co-adaptation of neurons, thus helping to generalize. L2 weight regularization (ridge regularization) punishes large parameter values, and also helps prevent overfitting [1]. By horizontally flipping and shifting the data, the distribution of data was expanded while no additional labelled data were collected, which is known to be essential for CIFAR-10 performance [7].

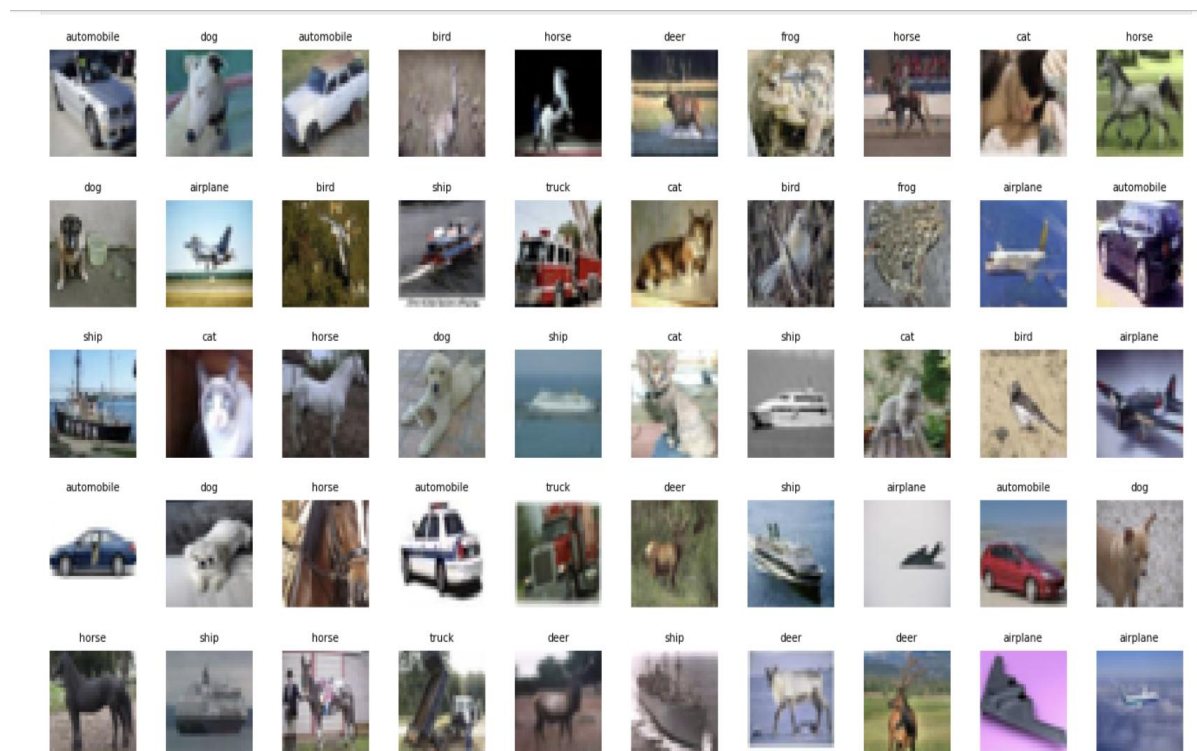
3. Workflow Methodology

3.1 Dataset and Pre-processing

Python's tools `urllib` and `pickle` were used to load the CIFAR-10 directly from the University of Toronto repository. The dataset is composed of 50,000 images of size 32x32x3 (cubes) and 10,000 test images, and the images are evenly distributed

among 10 categories: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck. The pixel values were normalized into the range $[0, 1]$ by dividing by 255, as is common practice to stabilize the gradient descent due to using the same magnitude of input. The categorical labels were one-hot encoded to match the categorical cross-entropy loss function, by using the `to_categorical` function provided by Keras. The class distribution was found to be the same throughout each of the ten classes in the training set on visual inspection, thus no class-weighting strategies were necessary.

Figure 1: Sample images from the CIFAR-10 dataset with class labels



3.2 Data Augmentation

During training an `ImageDataGenerator` object was set to randomly flip images horizontally and shift them horizontally and vertically by up to 10% of the image width and height. The same augmentation strategy was used for all four models. Augmentation works during training time, each epoch providing a random variation of the training image. This approach is based on the idea of a good classifier being unaffected by small translations or reflections of the input object [8]. Throughout, the

number of 32 was used for the batch size, which is a typical choice balancing between stochastic noise in the gradient calculations and efficient computation.

3.3 Model Architectures

Four successive CNN models were built and tested, which are called Model 1 (primary), Model 2 (batch-only), Model 3 (dropout-L2), and Model 4 (deeper network). Early stopping (patience = 2, monitor validation loss) was used on all models, and they were trained for a maximum of 50 epochs. Table 1 presents the main differences in the architecture and hyperparameters of the models. Model 1 had three convolutional blocks comprising two Conv2D layers with 3×3 kernels, batch normalization and max-pooling with 2×2 stride followed by 25% dropout. The filter counts were doubled: 32, 64, 128. The flattened feature vector was then fed into a 128 dense unit with dropout followed by a 10 unit dense softmax output layer. This architecture was compiled with the Adam optimizer and categorical cross-entropy loss.

In order to be able to compare the optimizer choice on the same architectures, the models 2, 3 and 4 were selected and used SGD as an optimizer. Model 2 was a baseline version with only regularization (i.e., batch normalization but no dropout). The dense layer of model 3 was modified by dropout (0.20 per block and 0.50 at the dense layer) and L2 regularization ($\lambda = 0.01$). Model 3 was further extended by another convolutional block (128 filters), to create the deeper version of the SGD-trained model, Model 4.

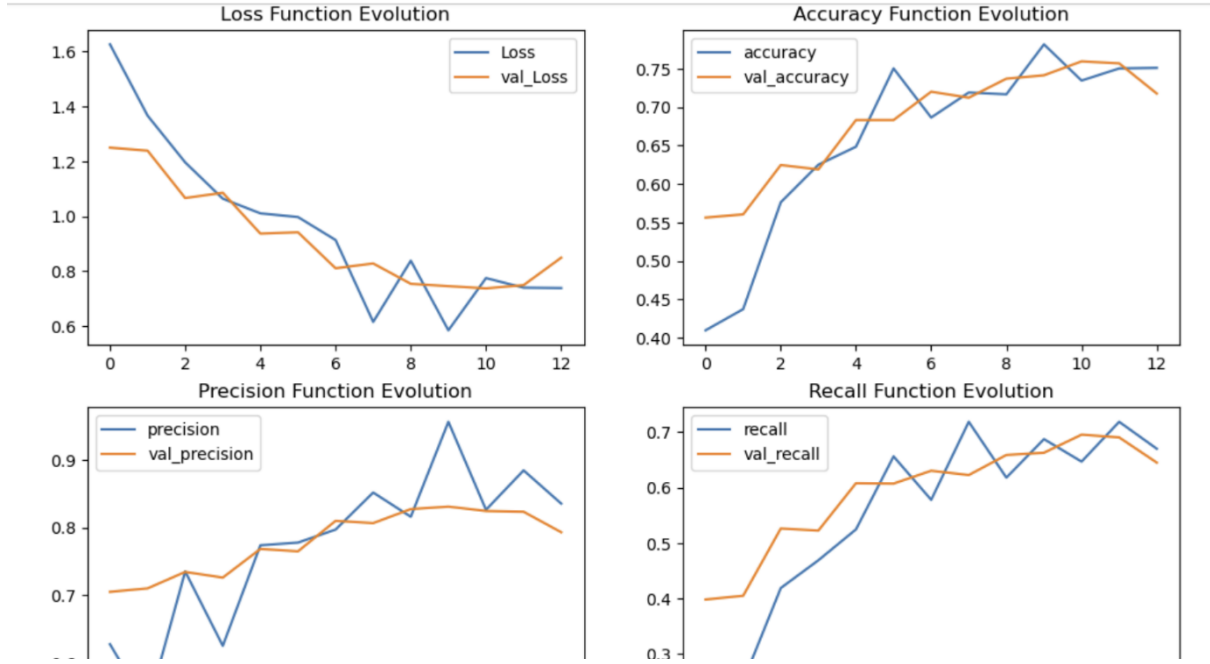
Table 1: Summary of Model Configurations

Feature	Model 1	Model 2	Model 3	Model 4
Conv Blocks	3	2	2	3
Filters (max)	128	64	64	128
Dropout	0.25 (all)	None	0.20 + 0.5	0.20 + 0.5
L2 Regularization	No	No	Yes (0.01)	Yes (0.01)
Optimiser	Adam	SGD	SGD	SGD
BatchNorm	Yes	Yes	Yes	Yes
Test Accuracy	71.74%	51.29%	45.16%	51.03%

3.4 Evaluation Metrics

The accuracy, precision, recall, and categorical cross-entropy loss were calculated for each model on the held-out test set. A confusion matrix was conducted to represent the per class performance of Model 1. Together these metrics offer a multi-dimensional analysis of the performance of the classifiers: accuracy measures the overall correctness of the prediction; precision measures the ratio between the number of correctly predicted positive and the number of positives overall; recall measures the ratio between the number of correctly predicted positive and the number of positive instances in the test set [12]. The F1 score is the harmonic mean of the precision and recall, but is not directly calculated; however, the values of precision and recall are sufficient to provide the basis for comparison.

Figure 2: Model 1 training and validation loss, accuracy, precision and recall across epochs



4. Model Performance Results

Overall performance of the four models is summarized in Table 2. Model 1 achieved the highest test accuracy value of 71.74%, precision value of 79.31% and recall value of 64.51%. This is a good ratio between precision and recall, meaning that the model has relatively few false positive predictions, while not making many false negative predictions. The difference between precision and recall may be due to difficulty level within the class, e.g., the class categories of cat and dog are visually similar and are likely to have a disproportionately large impact on the difficulty of recall, which is a common finding in the literature of CIFAR-10 [7]. The accuracy of the model 2 without the dropout feature was 51.29%, which is significantly lower than the accuracy of the model 1 without the dropout feature. This indicates that batch normalization, in particular, has a lot of implicit regularization when it is used across all convolutional layers. In terms of the training logs, the accuracy per epoch for the SGD optimizer is consistently lower than that of Model 1 during the early epochs, though the general trends seem to be the same.

The model 3 was the least accurate one with 45.16% accuracy in the test set and with the addition of dropout and L2 regularization. It is a counterintuitive result, which is probably due to the combination of aggressive L2 penalty and the sensitivity of the

SGD optimizer to the learning rate coupled with high dropout rates. These regularization methods could be responsible for under fitting, as suggested by [14] who state that if the rate of drop-out is more than 0.5, it can affect the learning. This is corroborated by the training loss of Model 3 which was significantly higher than the other models.

To test whether there is room for over-regularization to be partly compensated for by adding representational capacity, Model 4 was trained with a third convolutional block, increasing its performance to 51.03%. But it is still not as good as Model 1's Adam performance. Performance is enhanced by the introduction of 128-filter layers, which enhance the ability to capture more complex spatial features, but do not outperform the simpler Adam-trained architecture under SGD with L2 regularization.

Table 2: Final Test Performance Metrics by Model

Model	Accuracy (%)	Precision (%)	Recall (%)
Model 1 (Adam + Dropout)	71.74	79.31	64.51
Model 2 (SGD + BatchNorm)	51.29	70.19	33.96
Model 3 (SGD + Dropout + L2)	45.16	58.82	31.05
Model 4 (SGD + Deeper + L2)	51.03	70.36	32.45

5. Discussion and Recommendation

5.1 Effect of Optimiser Choice

The comparison of the accuracy and convergence rates of the four models presented here (2-4) with that of the other model (1=Adam) also gives clear indication of the influence of the choice of the optimizer on the speed of convergence and final accuracy in this experimental setting. In addition, Adam's adaptive gradient scaling

allows for faster convergence during the initial epochs and the momentum terms allow for better scaling of the loss landscape features, including saddle points [6]. Adam's benefits in convergence seem to overcome the generalization benefits of SGD for a relatively shallow, moderate size dataset such as CIFAR-10. Learning rate schedulers used in SGD could be explored in future work as they have been proven to achieve the generalization gap [10].

5.2 Batch Normalization and Dropout

All four models contained batch normalization which is regarded as one of the key features of the competitive performance among architectures. Researchers in [5] show that batch normalization is helpful to stabilize the network by normalizing the activations in intermediate layers, and to make it more robust to initialization by learning rates. This is clear in the present models, in the relatively swift loss reduction over training epochs. Dropout in Models 1, 3 and 4 offered extra regularization, which was helpful when combined with Adam training (Model 1), but was detrimental to under fitting in the SGD + L2 case (Model 3). This interaction shows that regularization strategies and the selection of optimizer should be co-designed.

5.3 Recommendations

From the experimental results, Model 1 is suggested as the best architecture design in this design space for the classification of CIFAR-10 dataset. It performs best in terms of accuracy, precision and recall with Adam optimization, batch normalization and moderate dropout. Experimental work suggests further performance gains can be obtained by experimenting with learning rate scheduling with warm restarts [10], residual connections to enable deeper networks without degradation [3]. These directions would lead performance into the state of the art, and develop on the foundation of the standard established here.

6. Conclusion

The study developed four CNN architectures and compared them, systematically changing the type of optimizer, the regularization strategy, and the depth of the architecture, for the task of image classification in the CIFAR-10 dataset. The primary model, with three convolutional blocks and batch normalization, dropout,

and the Adam optimizer, and trained using data augmentation, reached a test accuracy of 71.74%, which is a strong baseline compared to the literature that is used as a reference. Comparative experiments showed that, in this experimental setup, the Adam optimizer outperforms SGD in terms of the convergence speed and final accuracy, while numerous regularization methods can be used in combination, but their combination should be carefully managed to avoid under fitting.

Literature review confirmed that CNNs are structurally appropriate for image classification problems because they are able to learn feature hierarchies that are invariant to spatial structure; RNNs are appropriate for sequential domains like natural language processing and time-series modelling. The architecture selection should then be based on the structural characteristics of the input data and the type of the temporal or spatial dependency if found.

The biggest drawback of this work is that there are no transfer learning or the latest state-of-the-art network architectures like ResNets or EfficientNet which are used in the CIFAR-10. It is also important to include cross validation in future work to get more representative performance estimates and to use other hyperparameter search methods like Bayesian optimization. Still, the results presented here validate the design choices made in a carefully designed CNN with principled regularization, as it attains competitive performance on a challenging multi-class benchmark.

References

- [1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
- [2] A. Graves and J. Schmidhuber, “Framewise phoneme classification with bidirectional LSTM and other neural network architectures,” *Neural Networks*, vol. 18, no. 5–6, pp. 602–610, 2005, doi: 10.1016/j.neunet.2005.06.042.
- [3] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.
- [4] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.
- [5] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proc. 32nd Int. Conf. Mach. Learn. (ICML)*, PMLR vol. 37, pp. 448–456, 2015.
- [6] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015. [Online]. Available: <https://arxiv.org/abs/1412.6980>
- [7] A. Krizhevsky, *Learning Multiple Layers of Features from Tiny Images*, Tech. Rep., University of Toronto, Toronto, ON, Canada, 2009.
- [8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, vol. 25, 2012, pp. 1097–1105.

- [9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998, doi: 10.1109/5.726791.
- [10] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017. [Online]. Available: <https://arxiv.org/abs/1608.03983>
- [11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015. [Online]. Available: <https://arxiv.org/abs/1409.1556>
- [12] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, 2009, doi: 10.1016/j.ipm.2009.03.002.
- [13] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," in *ICLR Workshop*, 2015. [Online]. Available: <https://arxiv.org/abs/1412.6806>
- [14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [15] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, vol. 97, pp. 6105–6114, 2019. [Online]. Available: <https://arxiv.org/abs/1905.11946>

Appendix: Video demonstration link

Video link: <https://youtu.be/K26esOE4WRM>