International Journal on Engineering Technology (InJET)

www.kec.edu.np/journal
Volume 1, issue no. 1, Nov 2023, Page 37-52
Received Date: June 30, 2023
Accepted Date: Oct 8, 2023
ISSN: 3021-940X (print)

Comparison of CNN Architecture of Image Classification Using CIFAR10 Datasets

Yogesh Pant[1*], Gaurav Shah[2*], Roshan Ojha[3], Roshan Thapa[4], Bharat Bhatta[5]

[1]*Sagarmatha Engineering College,* Sanepa, Lalitpur Nepal*, 076bct047.yogesh@sagarmatha.edu.np*
[2]*Sagarmatha Engineering College, Sanepa, Lalitpur Nepal, gaurav.shah@sagarnatha.edu.np*
[3]*Sagarmatha Engineering College, Sanepa, Lalitpur, Nepal, 076bct036.roshan@sagarmatha.edu.np*
[4]*Sagarmatha Engineering College, Sanepa, Lalitpur, Nepal, 076bct037.roshan@sagarmatha.edu.np*
[5]*Sagarmatha Engineering College, Sanepa, Lalitpur, Nepal, bharat.bhatta@sagarmatha.edu.np*

Abstract

This paper demonstrates image classification using deep learning. Deep learning has the inherent ability to automatically discover and extract meaningful features for specific applications. Among the popular techniques in deep learning, the convolutional neural network (CNN) stands out. CNN consists of an input layer, hidden layers, and an output layer, where meaningful features are automatically extracted from input images.

This paper presents the performance and identifies the most effective CNN architectures for accurately classifying images in the CIFAR-10 dataset. Five CNN architectures were implemented, namely [Architecture 1], [Architecture 2], [Architecture 3], [Architecture 4], and [Architecture 5] using the CIFAR-10 dataset. The architectures were selected based on the need to explore variations in convolutional filter sizes, dense layers, and batch normalization to assess their impact on CIFAR-10 image classification performance. Each architecture was trained on a standard training set and evaluated on a validation set. We used specific details on data preprocessing and training settings for a consistent and fair comparison. After training and evaluation, we have obtained the following results for each architecture.

Architecture 1 has a training accuracy of 74.7% and validation accuracy of 76.6%, Architecture 2 has 96.09% and 86.08%, Architecture 3 has 77% and 78.1%, Architecture 4 has 67.91% and 69.54%, and Architecture 5 has 94.64% and 87.34% of training accuracy and validation accuracy respectively. After conducting a comparative analysis, we found that Architecture 5 has achieved the highest validation accuracy in classifying images in the CIFAR-10 dataset. These findings suggest that Architecture 5 is a promising choice for image classification tasks involving the CIFAR-10 dataset.

Keywords: Convolutional Neural Network (CNN), CIFAR-10 dataset, image classification, architecture comparison, performance evaluation.

1. Introduction

Image classification is a fundamental task in computer vision that involves the application of computer algorithms to process digital images. This process typically comprises several essential steps, including importing the image through an acquisition tool, analyzing the acquired image, and generating results based on the analysis. Various operations can be performed on the input image throughout these steps to enhance its quality or extract valuable information (MOHAMED NOUR, 2022).

**Corresponding Author*

Image classification involves categorizing and labeling groups of pixels or vectors in an image based on predefined rules. In supervised image classification, training data is selected from the image assigned to predefined categories or classes. Deep learning, a type of machine learning, utilizes layered algorithms structured as artificial neural networks (ANNs), and ANNs are inspired by the biological networks of the human brain and can be simulated to perform complex tasks such as image classification (MOHAMED NOUR, 2022).

In recent years, Convolutional Neural Networks (CNNs) have become widely acclaimed for their effectiveness in image classification tasks. The initial work by Yann LeCun et al. (LeCun, 1998) known as LeNet, introduced CNNs for the classification of handwritten digits and Alex Krizhevsky's AlexNet (Krizhevsky, 2012) achieved remarkable performance on the CIFAR-10 dataset (Ajala, 2021) and set a benchmark in the ImageNet classification competition in 2012.

This groundbreaking contribution has contributed to the surge in popularity and utilization of CNNs in image classification tasks. CNNs can automatically extract higher-level representations from visual data, eliminating the need for manual feature extraction. This eliminates the time-consuming and costly process of constructing features based on domain expertise for machine learning algorithms (Maminiaina Alphonse Rafidison, 2023).

The study aims to explore and understand various CNN architectures used for image classification. Our primary focus is on the CIFAR-10 dataset, which contains 60,000 color images, each sized 32 x 32 pixels, grouped into ten distinct categories. CIFAR-10 is widely recognized as a benchmark for rigorously evaluating classification models. To delve into the intricacies of CNN performance, we meticulously design and train these architectures by progressively adding layers. Importantly, we augment the third, fourth, and fifth layers with dropout layers, strategically included to mitigate overfitting during training. The culmination of this comprehensive exploration is our quest to identify the most optimal architecture, driven by the pursuit of accuracy optimization.

In the following sections, we'll explain the methods we used, the experiments we conducted, and the results we got. We aim to help others understand which CNN architecture works best for classifying images with the CIFAR-10 dataset.

2. Literature Review

Vision is the act or state of seeing. We humans rely on our senses of sight more than a lot of other animals (Ryan, 2011). It is easy for us to recognize people whom we just met but on the other hand, it is very difficult for state-of-the-art classifiers (Kris, 2020).

Machine learning is an application of artificial intelligence (AI) that provides systems with the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves (Megantara, 2021)

Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos, and other visual inputs and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe, and understand. Computer vision works much the same as human vision, except humans have a head start. Human sight has the advantage of lifetimes of context to train how to tell objects apart, how far away they are, whether they are moving, and whether there is something wrong with an image (Megantara, 2022).

Deep Learning is a sub-field within Machine Learning, and it is concerned with the utilization of Artificial Neural Networks (ANN) for solving natural language and computer vision tasks such as object detection, object recognition, face detection, pose estimation, semantic segmentation, and more (Jiahuan Zhang, 2022).

There are varieties of configurations of ANN that are present within the deep learning field, and notable configurations are convolutional neural networks (CNN), recurrent neural networks (RNN), and deep neural networks (DNN) (Jiahuan Zhang, 2022).

Convolutional neural networks (CNN) are one of the Deep Learning models that are often used to classify an image in .jpeg form [8]. Convolutional neural networks are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs. Convolutional neural networks power image recognition and computer vision tasks. A CNN helps a machine learning or deep learning model "look" by breaking images down into pixels that are given tags or labels. It uses the labels to perform convolutions (a mathematical operation on two functions to produce a third function) and makes predictions about what it is "seeing." The neural network runs convolutions and checks the accuracy of its predictions in a series of iterations until the predictions start to come true. It is then recognizing or seeing images in a way like humans(Megantara, 2022).

3. Methodology

The following steps were performed while building architectures using CNN:

3.1. Dataset

We used the CIFAR-10 tiny images dataset from image classification. It provided 10 classes of data with 5000 training and 1000 testing images per class. The classes obtained from the CIFAR-10 dataset are: 'airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', and 'truck' (Doon, 2018).

3.2. Preprocessing

The collected dataset underwent pre-processing. Various augmentation techniques were applied to the data, such as re-scaling, shear range, zoom range, horizontal flip, and normalization, to ensure consistency in image size and quality. We have used the Shear Range of 0.2 to control deformations to images, aiding the model's ability to generalize across variations in object orientations, a Zoom Range of 0.2 to handle different object sizes, and exposing the model to diverse scales of objects and horizontal flip which helps CNN recognize objects regardless of whether they are oriented to the left or right.

These augmentation techniques collectively contribute to the dataset's robustness and help the model generalize better across a wide range of scenarios.

The resulting images were of 32x32 size with RGB color channels. The dataset was then split into a training set of 50,000 images which were further divided into batches of 32, and a validation set with a split ratio of 0.1.

To facilitate these preprocessing steps, we leveraged the Image Data Generator utility function provided by Keras. This powerful tool not only automates the application of augmentation techniques but also ensures data consistency and scalability throughout the training process. By automating these tasks, we maintain data quality and reduce the risk of overfitting, ultimately contributing to the robustness of the generalization capability of the trained model (Boualam, 2021). For normalization the following equation is used (Chansung, 2018):

$$x_{normalized} = (x - m)/(x_{max} - x_{min}) \quad \text{(Equation 1)}$$

Where,

x = data point to normalize,
m = mean of the dataset,
x_{min} = minimum value,
x_{max} = maximum value

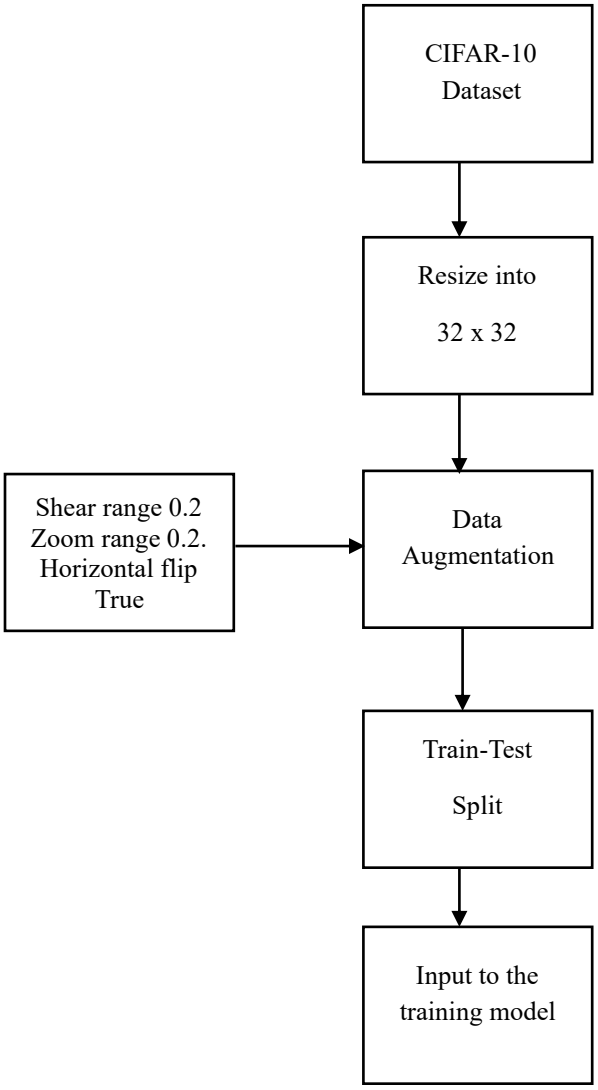

Fig 1: Data Flowchart

3.3. Train-validate-test Split

The CIFAR-10 dataset contains 5000 images per class for training which was split into 4500 images for testing and 500 images for validation. Further, there are 1000 images per class for testing purposes.

The split categories

- Training data: 4500 images per class
- Validation data: 500 images per class
- Testing data: 1000 images per class

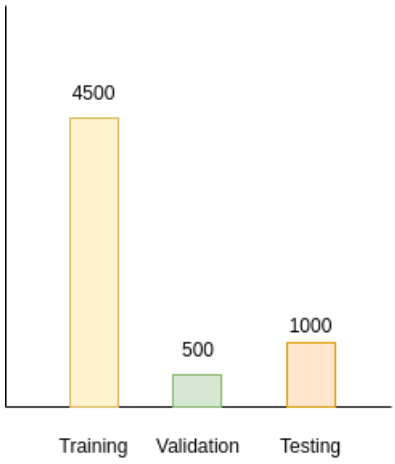

Fig 2: Data split chart

3.4. CNN model architecture

A local machine was used to build and train the image classifier model. The different layers used in each architecture are described below in each section.

3.4.1. Input Layer

The input layer takes 32x32-sized images as input. The images used for training are of 3 color channels (i.e., RGB) to increase the feature count of the image. The input convolutional layers consist of 32 filters of size 3x3.

3.4.2. Convolutional Layer

In total, 6 convolutional layers were used in this architecture. In each convolutional layer, the number of filters doubles that of the previous one to increase the feature extraction in the model. However, the filter size was kept constant at 3x3.

3.4.3. Padding

For padding, we have utilized the padding = "same" feature of Kera's library. This feature adds a single layer of '0' values at the edge of the images so that the edge pixels are not lost while features are being extracted.

3.4.4. Batch Normalization

Batch normalization is a technique for training very deep neural networks that normalize the contributions to a layer for every mini batch. This has the impact of settling the learning process and drastically decreasing the number of training epochs required to train deep neural networks. After each Convolutional layer, batch normalization is applied to normalize the parameters to normalize the results.

3.4.5. Max Pooling

Max Pooling layers were applied after a pair of convolutional layers. By doing this, we have been able to reduce the information the image contains (by keeping only half of the pixels) but keep and intensify the useful features the filters show when convolving the image.

3.4.6. Dropout

Our model has over 10 million parameters, out of which some of them might not be necessary. Therefore, to deactivate the unnecessary neurons in the architecture, we have applied two Dropout layers of value 0.5 each in the Fully connected layers which causes 50% of the neurons to disable randomly. Moreover, by adding dropout layers we have reduced the chances of our model over-fitting.

3.4.7. Callbacks

During the training process, we might need to check the model at various stages for performance and stop the training process if the model is not improving. Also, we need to record the logs of training to visualize the parameter changes for improving the model architecture in the next iteration. Therefore, we have used three callback functions for the following purposes in our model:

1. Checkpoint saving: To prevent the unexpected halt of model training and continue from the same point where we stopped training, we have used Model Checkpoint callback which periodically stores the temporary values of all the parameters as checkpoints.

2. Early item stopping: If the model does not improve significantly over the training period, we can stop the training process automatically by using the early stopping function.

3.TensorBoard: We have used the Tensor Board callback function to record logs during the training process so that it can be used for visualizing the model.

3.4.9. Adam Optimizer

The optimization function can drastically reduce the training time for a model. So, optimization is heavily used in neural network implementation. For our model, we have used ADAM optimizer. It is an adaptive learning rate optimization algorithm that is known for its fast convergence and good performance on a wide range of deep learning tasks.

ADAM optimizer combines the advantages of two other optimization algorithms, namely, gradient descent and RMSprop. It uses a first-order momentum term that calculates the exponential moving average of the gradient and a second-order momentum term that calculates the exponential moving average of the squared gradient.

The learning rate for ADAM was set at 0.001 For the Adam optimizer, the following equation is used for the calculation (Mohammed, 2020):

$$m_t = \beta m_{t-1} + (1 - \beta) \left[\frac{\delta L}{\delta w_t} \right] \quad \text{(Equation 2)}$$

3.4.10. Loss Function

In machine learning, a loss function is a mathematical function that calculates the difference between the predicted output and the actual output for a given input. The loss function is used to determine the quality of the model's predictions and is used as a measure of how well the model is performing during training.

We have implemented Categorical cross-entropy as our loss function. Categorical cross-entropy is a loss function used in machine learning for multi-class classification problems. It is commonly used when the predicted output is a probability distribution over multiple class.

The categorical cross-entropy loss function measures the dissimilarity between the predicted probability distribution and the actual probability distribution of the classes. It calculates the sum of the negative logarithm of the predicted probability of the correct class, weighted by the true probability of that class. For categorical cross entropy(Chansung, 2018),

$$L = -1/N * (\mathrm{X})^{N}_{i} (\mathrm{X})^{M}_{j} t_{i,j} * log(y_{i,j}) \quad \text{(Equation 3)}$$

3.4.11. Activation function

In machine learning, an activation function is a mathematical function that is applied to the output of a neural network layer. The activation function helps to introduce non-linearity into the model, which is important for the model's ability to learn complex patterns in the data.

Our model has 'ReLU' as an activation function in the hidden layers and 'softmax' in the output layer.

The ReLU (Rectified Linear Unit) activation function is commonly used in the hidden layers of a neural network because it is computationally efficient and addresses the vanishing gradient problem. ReLU is a simple function that sets the output to zero if the input is negative and returns the input if it is positive. This makes the ReLU function very fast to compute and allows for faster training of deep neural networks.

In addition, ReLU is known to address the vanishing gradient problem, which occurs when the gradients become too small to effectively update the weights during backpropagation. The ReLU function avoids this problem by returning non-zero gradients for positive input values, which ensures that the gradients do not become too small during backpropagation.

On the other hand, the softmax function is commonly used in the output layer of a neural network for multi-class classification problems. The softmax function maps the output of the last layer to a probability distribution over the different classes. It is a generalization of the logistic function used in binary classification problems.

The softmax function is used in the output layer because it can handle multiple classes and returns the probability of each class given the input data. This makes it useful for classification problems where the output is a probability distribution over multiple class. For ReLU activation (Chansung, 2018),

$$f(x) = max(0,x) \quad \text{(Equation 4)}$$

For softmax activation,

$$softmax(z_i) = exp(z_i)/f(\mathrm{X})_j(exp_j)) \quad \text{(Equation 5)}$$

4. Implementation

We employed various software tools and libraries to investigate the performance of different Convolutional Neural Network (CNN) architectures for image classification using the CIFAR-10 dataset. Python served as the programming language, and we harnessed the power of TensorFlow and Keras to construct and train our neural networks.

Additionally, we had access to a local machine equipped with a 4GB NVIDIA 1650Ti GPU, which accelerated our training processes.

To enhance the quality and consistency of our dataset, we applied essential data preprocessing steps. This involved employing data augmentation techniques such as re-scaling, shear range, zoom range, horizontal flip, and normalization. To streamline this process and ensure data consistency, we made use of Kera's Image Data Generator utility function.

Our model architectures were carefully designed to explore variations in complexity, ranging from the basic Architecture 1 with convolutional and fully connected layers to the more complex Architecture 5, which integrated batch normalization and dropout layers. Training employed the ADAM optimizer with a learning rate of 0.001, utilizing Categorical Cross Entropy as the loss function. To monitor and potentially halt training, when necessary, Early Stopping callbacks were implemented, Finally, we used Tensor Board to record training logs for visualization and analysis.

Throughout the implementation, we faced challenges related to overfitting, prompting the inclusion of dropout layers to address this issue. Tuning hyperparameters like the learning rate and the number of training epochs played a vital role in optimizing our models.

5. Results and Analysis

5.1. Experimental setups

We have trained five CNN architectures for image classification in our paper. The model was trained on a local machine where it was provided with 16.00GB of RAM and a 4GB NVIDIA 1650Ti GPU. We ran experiments over 5 types of models and the best was chosen. Our experimental setups of five architectures are tabularized below:

Architecture 1: With Convolutional layers and fully connected layers

Table 1: Architecture1 configuration table

Particulars	Specifications
Total Convolutional Layers	6
Kernel Size	3x3
Total Maxpooling2D layers	3
Optimizer	ADAM (Learning rate = 0.001)
Loss Function	Categorical Cross Entropy
Epoch	38

Architecture 2: With Convolutional layers and Batch Normalization layers

Table 2: Architecture2 configuration table

Particulars	Specifications
Total Convolutional Layers	6
Kernel Size	3x3
Batch Normalization Layers	3
Total Maxpooling2D layers	3
Optimizer	ADAM (Learning rate = 0.001)
Loss Function	Categorical Cross Entropy
Epoch	30

Architecture 3: With Convolutional layers and a single dropout layer

Table 3: Architecture3 configuration table

Particulars	System Specification
Total Convolutional Layers	6
Kernel Size	3x3
Dropout Layer	0.5
Total Maxpooling2D layers	3
Optimizer	ADAM (Learning rate = 0.001)
Loss Function	Categorical Cross Entropy
Epoch	45

Architecture 4: With Convolutional layers and two dropout layers

Table 4: Architecture4 configuration table

Particulars	System Specification
Total Convolutional Layers	6
Kernel Size	3x3
Dropout Layers	0.5+0.5
Total Maxpooling2D layers	3
Optimizer	ADAM (Learning rate = 0.001)
Loss Function	Categorical Cross Entropy
Epoch	33

Architecture 5: With Convolutional layers, a Batch Normalization layer, and two dropout layers

Table 5: Architecture5 configuration table

Particulars	System Specification
Total Convolutional Layers	6
Kernel Size	3x3
Dropout Layers	0.5+0.5
Total Maxpooling2D layers	3
Total Maxpooling2D layers	3
Optimizer	ADAM (Learning rate = 0.001)
Loss Function	Categorical Cross Entropy
Epoch	45

The evaluation metrics that we observed based on the evaluation of our results are:

- Training loss
- Training accuracy
- Validation loss
- Validation accuracy

5.2. Experimental results

5.2.1. Architecture 1

For our first setup, we have only used Convolutional layers and Fully Connected Layers. We can see the training accuracy graph increases smoothly until the 30th epoch, after which it begins to show downward spikes recovering immediately. The validation curve follows the training curve but the high fluctuations.

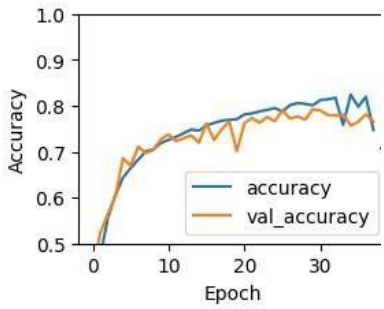

Fig 3: Training accuracy vs. Validation accuracy vs. Epoch

As per the loss function, we can see the minimum validation loss at 0.834 at the 37th epoch. The validation loss tends to decrease with high fluctuations until the 15 epochs after which it tends to increase in a parabolic arc manner. The training loss gradually decreases until the 35 epochs after which it begins to show high spikes of loss eventually settling with the validation loss at the very end of the training.

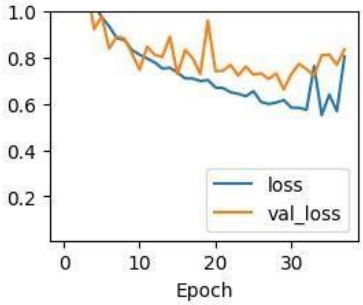

Fig 4: Training loss vs. Validation loss vs. epoch

5.2.2. Architecture 2

For architecture2, we have introduced batch normalization layers. The training accuracy curve increases smoothly until the end of training. However, the validation accuracy of the model increases sharply until the 7th epoch after which it begins to settle down around a baseline while showing downward spikes randomly.

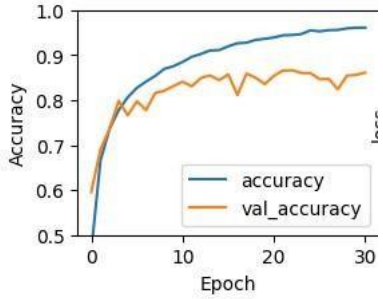

Fig 5: Training accuracy vs. Validation accuracy vs. Epoch

As per the loss function, the training loss decreases smoothly until the end of the training, but the validation curve shows random large spikes. This setup executes until the 30th epoch after which it is stopped by the callback function.

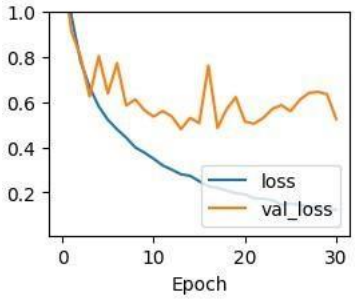

Fig 6: Training loss vs. Validation loss vs. Epoch

5.2.3. Architecture 3

In our architecture3, we have included a single dropout layer of value 0.5 in the fully connected layer before the output layer. We can see a smooth training accuracy curve throughout the training process. Regarding the validation accuracy curve, we can see the curve following the training accuracy curve closely, but it shows fluctuations.

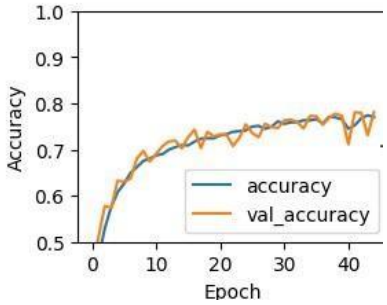

Fig 7: Training accuracy vs. Validation accuracy vs. Epoch

For the loss graph, we can see an insignificant drop in the loss of both training and validation. The training loss decreases gradually but the validation loss curves show high peaks after 20 epochs.

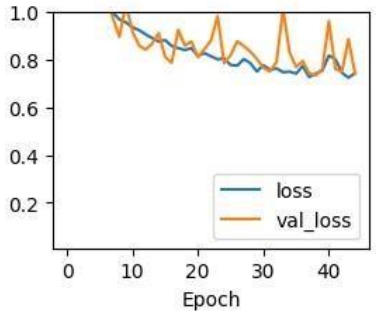

Fig 8: Training loss vs. Validation loss vs. Epoch

5.2.4. Architecture 4

In our architecture4, we have included double dropout layers of value 0.5 each in the fully connected layer. We can see a gradual increase in the training and validation accuracy of the model. Both curves maintain similar values throughout the training session. However, we can see the validation accuracy curve showing some fluctuations in the results.

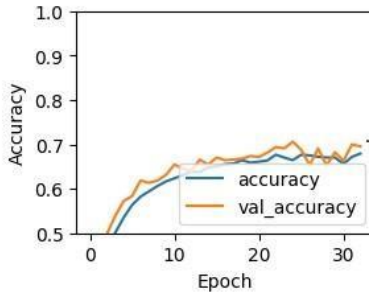

Fig 9: Training accuracy vs. Validation accuracy vs. Epoch

For the loss graph. We can see no drop in the loss of both training and validation. The loss for both curves is way beyond the limitations of the graph. We can state that this is due to adding double dropout layers as the dropout layers cause an increase in the loss of the model. A similar curve was seen in the single dropout layer setup but with a lower training and validation loss. But the training stops after 35 epochs only due to the triggering of an early Stopping callback.

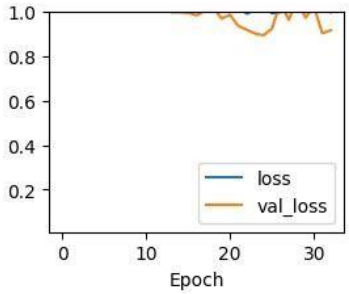

Fig 10: Training loss vs. Validation loss vs. Epoch

5.2.5. Architecture 5

In our architecture5, we have included batch normalization with double dropout layers. We can see an exponential increase in both the training and validation accuracy of the model concerning the number of epochs. However, after 10 epochs of training the training accuracy increases smoothly but fluctuations are seen on the validation accuracy curve. Also, the validation accuracy graph is shifted down from that of the training accuracy graph.

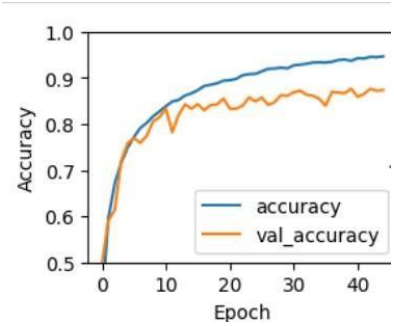

Fig 11: Training accuracy vs. Validation accuracy vs. Epoch

For the loss graph. We can see an exponential drop in training and validation loss concerning the number of epochs. However, after 10 epochs only the training loss continues to decrease but the validation loss maintains a baseline but shows high fluctuations around that baseline.

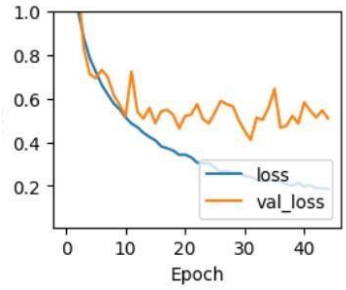

Fig 12: Training loss vs. Validation loss vs. Epoch

This model continues to train to 45 epochs generating a validation accuracy of 87.34% and a validation loss of 0.5089.

5.3. Comparative Analysis of all Architectures

Comparing all setups' training and validation graphs side-by-side we can see that architecture5 has the largest validation accuracy. Comparing all setups' training and validation graph side-by-side we can see that architecture2 and 5 have almost similar validation losses.

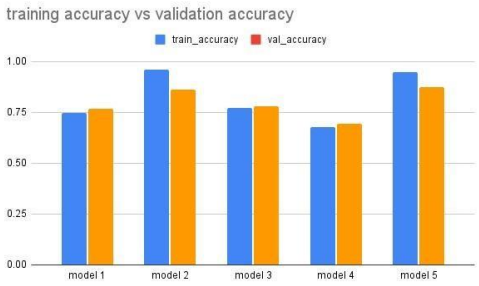

Fig 13: Accuracy graph comparative analysis of all architectures

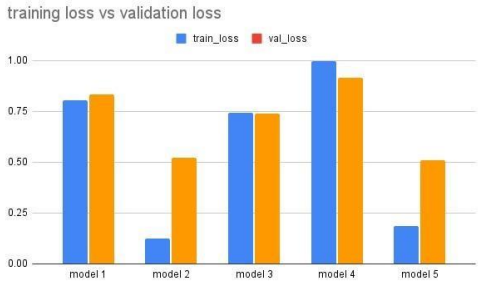

Fig 14: Loss graph comparative analysis of all architectures

5.4. Summary of our architectures

Table 6: Summary of all architectures

	setup 1	setup 2	setup 3	setup 4	setup 5
Train accuracy	0.747	0.9609	0.77	0.6791	0.9464
Val accuracy	0.766	0.8608	0.781	0.6954	0.8734
Train loss	0.803	0.1243	0.7424	0.9965	0.1869
Val loss	0.8342	0.524	0.7391	0.9149	0.5089
Epochs	38	30	45	33	45

6. Discussion

The discussion section provides a platform for a deeper understanding of the significance and implications of our findings. We observed compelling trends in the training and validation metrics across our various model architectures. Architecture 5, which incorporated both batch normalization and dropout layers, emerged as the standout performer, achieving the highest validation accuracy. This highlights the practical significance of well-designed model configurations.

Furthermore, fluctuations in the validation metrics of some architectures underscore the critical role of regularization techniques, such as dropout layers, in mitigating overfitting. These findings have practical implications for image classification tasks, particularly on the CIFAR-10 dataset. The choice of architecture can significantly impact both accuracy and training time, necessitating a thoughtful selection based on specific task requirements. Our study aligns with existing literature on techniques like batch normalization and dropout layers in deep learning models. It adds the importance of careful model design and regularization techniques in achieving optimal results.

Finally, our findings in this paper include batch normalization layers contributing to smoother training by normalizing layer outputs, reducing internal covariate shifts, and often leading to smoother training curves. Dropout layers can be used for regularization and model size reduction but require careful tuning to avoid over-regularization. Their effects can sometimes lead to lower accuracy and higher loss due to the reduction in trainable parameters. Combining both techniques (batch normalization and dropout) can sometimes yield models that strike a balance between stability and regularization, as seen in Architecture 5. To enhance the analysis, it is essential to consider dataset-specific factors and conduct rigorous evaluations on a separate test dataset to ensure the generalizability of the observed trends to unseen data.

7. Conclusion

Analyzing all the architectures that we performed in our paper. The validation accuracy and validation loss of architecture2 and architecture5 are very near to each other and that of architecture 1, architecture 3, and architecture 4 have much less accuracy. Despite the validation accuracy, the validation loss of Architecture 5 is less in comparison to the validation loss of Architecture 2. So, our choice of Architecture 5 as the preferred model is underpinned by its superior performance and efficiency, considering both accuracy and loss metrics. Our findings underscore the importance of thoughtful model design and regularization techniques in achieving optimal results for image classification tasks.

Acknowledgments

We would like to extend our sincere gratitude to Er. Bharat Bhatta, the Head of Electronics and Computer Engineering, at Sagarmatha Engineering College, Tribhuvan University, served as our research supervisor. His unwavering support, patience, motivation, and enthusiasm were instrumental in helping us develop our research and prepare its report, as well as obtaining the necessary resources for our research. Additionally, we would like to acknowledge the valuable assistance of our project Coordinator, Er. Bipin Thapa Magar provided us with prompt and useful troubleshooting assistance during research. Furthermore, we express our appreciation to Er. Baikuntha Acharya, the Deputy Head of Electronics and Computer Engineering, for his support and insightful guidance. We would like to extend our heartfelt appreciation to the KEC Journal (International Journal on Engineering Technology -InJET) for their invaluable recognition of our paper and its subsequent publication. We are truly grateful for the opportunity to share our research findings with a wide audience through such a reputable and esteemed platform. We also wish to thank all faculty members of the Department of Electronics and Computer Engineering for sharing their knowledge with us, which enabled us to efficiently complete our research. Finally, we would like to express our gratitude to everyone directly and indirectly involved in our project.

References

Ajala, S., 2021. Convolutional Neural Network Implementation for Image Classification using CIFAR-10 Dataset. *Norfolk State University, 700 Park Avenue, Norfolk, USA, 23504.*

Boualam, M. E. Y. K. G. &. M. M., 2021. Arabic Handwriting Word Recognition Based on Convolutional Recurrent Neural Network.. *Proceedings of the 6th International Conference on Wireless Technologies, Embedded, and Intelligent Systems.*

Chansung, P., 2018. *CIFAR-10 Image Classification in TensorFlow.* [Online]
Available at: https://towardsdatascience.com/cifar-10-image-classification-in-tensorflow-5b501f7d
[Accessed 02 11 2023].

Doon, R. R. T. K. &. G. S., 2018. Cifar-10 Classification using Deep Convolutional Neural Network. *In 2018 IEEE Punecon (pp. 1-5).*

Jiahuan Zhang, K. M. T. O. M. H., 2022. Regularization Meets Enhanced Multi-Stage Fusion Features:. *Graduate School of Information Science and Technology, Hokkaido University,* 22(14)(5431).

Kris, B., 2020. *Bridging the gap between human and machine vision.* [Online]
Available at: https://news.mit.edu/2020/bridging-gap-between-human-and-machine-vision-0211
[Accessed 02 11 2023].

Krizhevsky, A. S. I. &. H. G. E., 2012. ImageNet Classification with Deep Convolutional Neural Networks.. *In Advances in Neural Information Processing Systems (pp. 10971105)* .

LeCun, Y. B. L. B. Y. &. H. P., 1998. Gradient-based learning applied to document recognition.. *Proceedings of the IEEE,* Issue 2278-2323, p. 86(11).

Maminiaina Alphonse Rafidison, H. M. R. A. R. S. H. J. R. F. M. R. T. P. R. H. R., 2023. Image Classification Based on Light Convolutional Neural Network Using Pulse Couple Neural Network. *Telecommunication Automatic Signal Image Research, Laboratory/Doctoral School in Science and Technology of Engineering and Innovation/University of Antananarivo,* Volume 2023.

Megantara, E., 2021. *Simple Image Detection and Classification using CNN Algorithm.* [Online]
Available at: https://enricotara100.medium.com/simple-image-detection-and-classification-using-cnn-algorithm-ecf69e9eaab3
[Accessed 02 11 2023].

Megantara, E., 2022. *What is computer vision?.* [Online]
[Accessed 02 11 2023].

MOHAMED NOUR, R. M. A.-M. M. K., 2022. Performance Analysis and Evaluation of Image Classification Models Using Machine Learning.. *Journal of Theoretical and Applied Information Technology,* Volume 100(23).

Mohammed, A. I. &. T. A. A. K., 2020. A New Optimizer for Image Classification using Wide ResNet (WRN). *Academic Journal of Nawroz University,* 9(4)(1–13).

Ryan, E., 2011. *How does vision work?.* [Online]
[Accessed 02 11 2023].