University of Windsor

Exploring the Effectiveness of Deep Learning Architectures in Multiclass Classification Task Using the CIFAR-100 Dataset

Akshat Sharma *sharma8d@uwindsor.ca*
Justin Neal *nealj@uwindsor.ca*
George Kaceli *kaceli@uwindsor.ca*

December 14th, 2023

Abstract

Deep learning architectures have revolutionized image classification tasks by exhibiting remarkable performance across various datasets. This study investigates the diversity of different neural network models for multiclass classification using the CIFAR-100 dataset. The CIFAR-100 dataset comprises 100 object classes which present a challenging scenario for image recognition systems. In this research, an analysis and comparative evaluation of different deep learning models is explored. The focus is on exploring their performance, generalization capabilities, and robustness across the diverse range of CIFAR-100 images. The research involves the examination of a Dense Neural Network (DenseNN), a Convolutional Neural Network (CNN) and a Residual Network (ResNet). These architectures are implemented and fine-tuned for the CIFAR-100 dataset. Through rigorous experimentation and with the evaluation metrics the aim to discern the strengths and weaknesses of these models and to distinguish the most effective architectures for tackling this multiclass classification problem. Furthermore, this study delves into transfer learning strategies and data augmentation techniques to enhance model performance and generalize well across the dataset's varied classes. The findings from this research endeavor seek to provide insights into the comparative performance of deep learning architectures on the CIFAR-100 dataset but also contribute valuable knowledge towards designing robust image classification systems for diverse real-world applications.

Section I: Research Objective

Introduction

The field of computer vision has witnessed significant advancements in image classification tasks, primarily attributed to the advent of deep learning techniques. The CIFAR-100 Image Classification Project introduces the goal of building and training different types of Neural Networks for classifying images within the dataset. The CIFAR-100 dataset challenges image classification algorithms with its 100 varied classes, small 32x32 pixel images, and a diverse range of visual characteristics, making accurate object recognition and classification particularly difficult due to its varied object classes, diverse visual characteristics, and relatively low-resolution images. Each image is assigned a label. These labels are integer values representing one of the 100 classes. This research aims to determine whether conducting an extensive examination of the strengths and weaknesses of these distinct models via hyperparameter tuning will achieve the objective of improving the accuracy of image classification within the CIFAR-100 dataset.

Dataset

This dataset is comprised of 60,000 32x32 color images distributed across 100 diverse classes as seen in Figure 1.[1] It contains many images across 100 non-overlapping classes. Since the dataset contains 60,000 samples in total, this results in each class only having 600 samples. The classes can be seen in Table 1.

Table 1 The classes of the dataset

Superclass	Classes
Aquatic mammals	Beaver, dolphin, otter, seal, whale
Fish	Aquarium fish, flatfish, ray, shark, trout
Flowers	Orchids, poppies, roses, sunflowers, tulips
Food containers	Bottles, bowls, cans, cups, plates
Fruit and vegetables	Apples, mushrooms, oranges, pears, sweet peppers
Household electrical devices	Clock, computer keyboard, lamp, telephone, television
Household furniture	Bed, chair, couch, table, wardrobe
Insects	Bee, beetle, butterfly, caterpillar, cockroach
Large carnivores	Bear, leopard, lion, tiger, wolf
Large man-made outdoor things	Bridge, castle, house, road, skyscraper
Large natural outdoor scenes	Cloud, forest, mountain, plain, sea
Large omnivores and herbivores	Camel, cattle, chimpanzee, elephant, kangaroo
Medium-sized mammals	Fox, porcupine, possum, raccoon, skunk
Non-insect invertebrates	Crab, lobster, snail, spider, worm
People	Baby, boy, girl, man, woman
Reptiles	Crocodile, dinosaur, lizard, snake, turtle
Small mammals	Hamster, mouse, rabbit, shrew, squirrel
Trees	Maple, oak, palm, pine, willow
Vehicles 1	Bicycle, bus, motorcycle, pickup truck, train
Vehicles 2	Lawnmower, rocket, streetcar, tank, tractor

In the context of the CIFAR-100 dataset, each image is represented as a three-dimensional array with dimensions 32 x 32 x 3. There is no concept of columns in the traditional tabular sense. Instead in the context of image data, the number of features corresponds to the dimensionality of the input space. This means that the dimension is determined by the size and colour channels of the images. Where 32 x 32 is the spatial resolution which is composed of the height and width. While the 3 corresponds to the three-colour channels representing red, green, and blue. [2]

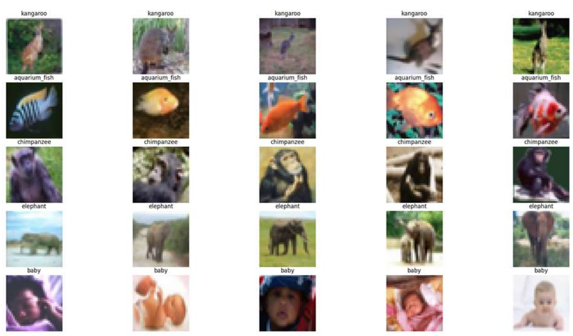

Fig.1: Sample Images and their Labels

Network

The neural network architecture is encapsulated within the Network class. This is achieved through a dynamic import mechanism, where the script imports the desired model architecture from a module by the model's variable name. The flexibility of dynamically importing the network class allows for a modular and configurable approach to building different neural network architectures based on the specified model names. The model variable holds the constructed neural network model, and subsequent operations, such as displaying the architecture summary, compiling the model, and training are performed using this instantiated model. [3]

Section II: Methodology

Data Preprocessing

Upon loading the data, the initial action is to preprocess it. This is handled by a preprocessing function that scales the pixel values to a range between 0 and 1 by dividing each value by 255 and returning the results. Once the model is successfully imported, an instance of the network is created, and the model is built. Afte the training is complete the important metrics such as training, testing loss, and accuracy values are used from the training history for further analysis. [4]

Dense Neural Network

The first fundamental architecture that will be explored is the dense neural network, also known as a fully connected neural network. In a dense network, every neuron in each layer is connected to every neuron in the subsequent layer. This creates a dense matrix of connections. This architecture allows the model to capture intricate relationships and patterns within the input data, making it particularly effective for tasks that involve complex feature recognition. The hierarchical structure of dense networks enables them to learn representations of increasing abstraction as information flows through the layers. Typically, a dense network begins with an input layer, followed by multiple hidden layers where the majority of the network's parameters reside, and concludes with an output layer that produces the final predictions. [5]

The dense model's input layer accepts 32x32 pixel color images, the subsequent integration of a Flatten layer prepares the data for dense connectivity. Three dense layers follow, progressively reducing the number of neurons (512, 256, 128) with each layer. Each layer is accompanied by a rectified linear unit (ReLU) activation, dropout regularization, and batch normalization. [6] These parameters collectively supply the network with the capacity to discover intricate patterns while reducing overfitting. An additional flatten layer precedes the final classification layer with 100 neurons. This layer employs the SoftMax activation function for the multiclass classification.

Table 2 Dense Neural Network Training Data

Dropout	Learning Rate	Epoch Stopped	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
None	0.1	11	3.7686	13.57%	479.8282	11.35%
None	0.01	24	2.3279	38.87%	3.4299	23.35%
None	0.001	23	2.0690	44.75%	3.4917	23.59%
None	0.0001	25	2.1768	45.18%	3.6905	19.50%
0.25	0.1	11	4.2775	5.78%	961.7994	2.82%
0.25	0.01	48	3.2679	20.70%	3.2072	23.13%
0.25	0.001	65	3.2135	21.40%	3.1580	28.30%
0.25	0.0001	99	3.0186	25.09%	3.1264	25.40%
0.50	0.1	11	4.6109	2.11%	418.1822	2.08%
0.50	0.01	99	3.7515	11.96%	3.5437	16.12%
0.50	0.001	45	3.7669	11.68%	3.6014	16.14%
0.50	0.0001	126	3.5786	14.99%	3.4240	18.73%

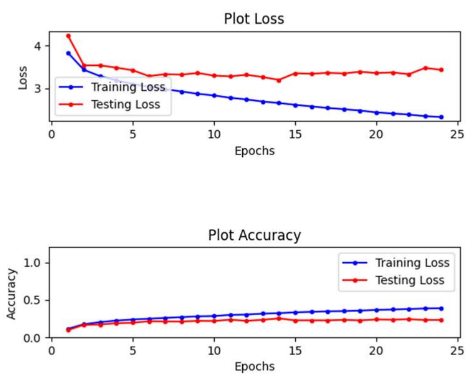

Fig.2: Dropout rate equal to none

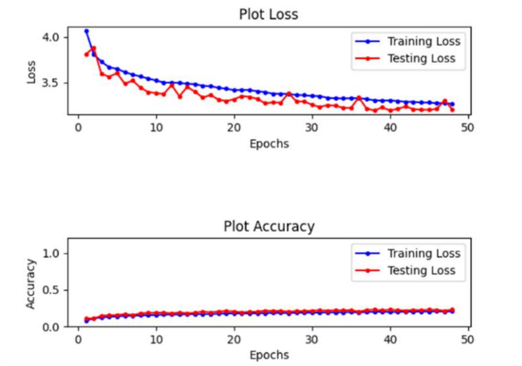

Fig. 3: Dropout at 0.25

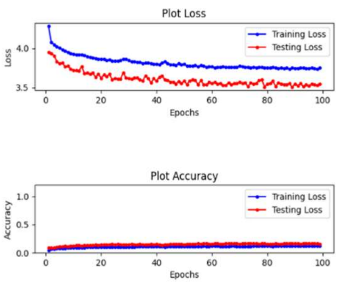

Fig.4: Dropout is at 0.50

Table 2 provides a comprehensive overview of the performance metrics for the DNN under various configurations, emphasizing the effects of dropout regularization and learning rate adjustments on how the model learns and generalizes. Analyzing the DNN configurations with learning rates fixed at 0.001 and varying dropout rates of none, 0.25, and 0.50, the results indicate that the introduction of dropout improves the model's generalization on validation data. Without dropout, the model achieves a higher training accuracy of 44.75% but a lower validation accuracy of 23.59% in figure 2. Introducing a 0.25 dropout rate the training accuracy drops to 21.40%, yet the validation accuracy increases to 28.30% marking it as the most effective at generalizing as seen in figure 3. With a 0.50 dropout rate, there's a further reduction in training accuracy to 11.96% and a validation accuracy of 16.12% from figure 4. This suggests that while dropout helps with generalization too much can hinder the network's ability to learn from the training data.

The influence of the learning rate on the DNN's reveals critical insights into its role in model training and generalization. A learning rate of 0.001, applied across different dropout settings (none, 0.25, 0.50), consistently results in a notable balance between training accuracy and validation accuracy. Specifically, in the absence of dropout, this learning rate leads to a robust training accuracy of 44.75% and a reasonable validation accuracy of 23.59%. This indicates a decent ability to generalize without overfitting. As dropout is introduced, a 0.25 dropout setting achieves the highest validation accuracy of 28.30%, despite a lower training accuracy of 21.40%. This suggests that the learning rate of 0.001, in conjunction with moderate dropout, optimally counters overfitting while maintaining learning efficiency. Conversely, with a higher dropout rate of 0.50, the same learning rate results in a lower training accuracy of 11.96% and a validation accuracy of 16.12%, indicating that while the learning rate is effective in managing the model's learning pace, excessive dropout can impede the model's ability to learn effectively. Therefore, the learning rate of 0.001 emerges as a key facilitator in balancing the trade-off between learning depth and generalization.

CNN

The CNN architecture has an increased significance when applied to datasets like CIFAR-100, where visual information requires a specialized approach. CNNs excel at capturing intricate spatial features through their convolutional layers. The convolutional operations strategically detect complex patterns within the dataset by finding subtle or fine distinctions, details, or variations within the images. Max pooling operations further obtain essential information, reducing spatial dimensions and enhancing computational efficiency. The incorporation of activation functions, such as ReLU, introduces non-linearity to the neural network. This non-linearity is crucial for the network to learn and capture intricate mappings within the data. Activation functions enable the model to introduce complexity and non-linear relationships, allowing it to represent and understand more sophisticated patterns and features in the input data. Without non-linear activation functions, the network would essentially be a linear system, limiting its capacity to learn complex representations. [6]

The CNN takes shape through a composition of convolutional blocks, each using a 2D convolutional layer, spatial distillation through max pooling, stability via batch normalization, and regularization facilitated by dropout. The convolutional layers have varying filter sizes ranging from 64, 128, 256, 512 to capture hierarchical features at different scales. The flattened output from these convolutional ensembles converges into fully connected layers, following with a dense layer of 512 neurons using an ReLU activation function, complemented by batch normalization and dropout. The final layer contains 100 neurons and a SoftMax activation function that will be used for the multiclass classification. [7]

The base CNN model was trained for 150 epochs with batch size of 128 and no augmentation to the data. The Adaptive Moment Estimation (Adam) and Stochastic Gradient

Descent (SGD) optimizer were tested with a learning rate of 0.0001. Early stopping was implemented if the model did not improve.

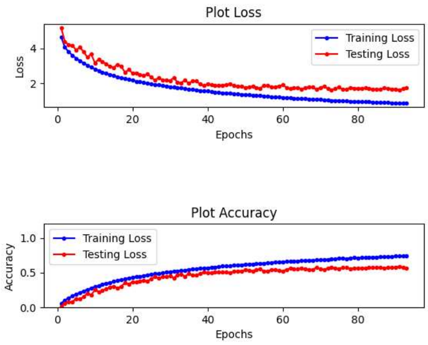

Fig.5: CNN with Adam

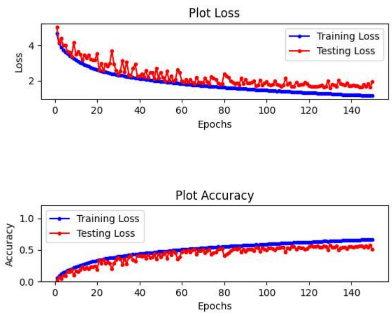

Fig.6: CNN with SGD

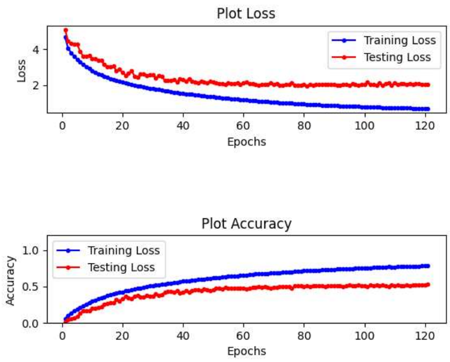

Fig.7: CNN with augmentation

The CNN with Adam stopped around 93/150 epochs as seen in figure 5. As the loss started stagnating and there was no significant improvement over the last few epochs. Instead of wasting computational resources, and not overfit it further, early stopping was utilized as a valuable regularization tool. In figure 6 the model's accuracy on the training data was around 70% and the accuracy on the testing data was 56%. From this it was concluded that the model was overfitting on training data. [7]

The CNN with SGD was worse and trained for over 150 epochs and it is observable that the loss was very unstable compared to Adam. It achieved an accuracy of 66% on training data and 51% on testing data. This shows that SGD did slightly worse than Adam and this was overfitting as well. [8]

From figure 6, it is inferred that the model trained with the Adam optimizer converges faster than the one with SGD, which is consistent with the expected behavior of these optimizers. Adam combines the benefits of two other extensions of stochastic gradient descent: AdaGrad and RMSProp. Adam computes adaptive learning rates for each parameter. In contrast, SGD maintains a constant learning rate throughout the training process or requires the learning rate to be manually changed.

Adam has adaptive learning rates that can lead to faster convergence and alleviate some of the tuning required for the learning rate schedule. This explains why the model with Adam optimizer shows a steeper decrease in loss and a quicker rise in accuracy. However, it is also observed that models trained with Adam sometimes plateau earlier, which could be a result of its rapid convergence. The models trained with SGD show more stable but slower progress this results in better generalization after sufficient epochs. To combat overfitting the CNN was ran with augmentation using Tensorflow's inbuilt ImageDataGenerator, with Horizontal_flip, and zoom_range, and shear_range. This training stopped at 121 epochs with a training accuracy of 78% and testing accuracy of 52% that can be seen in figure 7. The full results of the CNN training can be seen in Table 3.

Table 3 Convolutional Neural Network Training Data

Models	Epoch Stopped	Training Accuracy	Training loss	Testing Accuracy	Testing loss
Base CNN with Adam	93	70%	0.841	56%	1.843
Base CNN with SGD	150	66%	0.7513	51%	2.9763
Base CNN with augmentation	121	78%	0.7865	52%	2.0306

ResNet

The CIFAR-100 dataset, being very complex as discussed earlier causes a lot of challenges for basic models. Along with the small image size this causes significant problems for basic models like a standard CNN to extract meaningful data for classification. To combat the issue, a pre-trained models via Transfer Learning is used. This is the process of utilizing a model developed for a one task to be reused as a starting point for a model on a second task. It is useful in scenarios where there is a limited amount of training data or other limitations such as hardware. The popular residual network architecture (ResNet50) is used because of the "Residual blocks". These blocks help in training very deep networks by addressing the vanishing gradient problem. This is one of the key challenges in training very deep neural networks. The reason for this is that the gradients become too small to make any significant change in the weights during backpropagation this results in hindering the learning process. [9]

By introducing skip connection that are also known as shortcut connections. One or more layers within this network are bypassed which allows the training of much deeper networks. [10] These connections enable learning an identity function which represent and reproduce the input data without significant alteration. The skip connections ensure that higher layers can perform at least as well as lower layers. The depth of ResNet50 allows it to learn a rich hierarchy of features. In image classification tasks, lower layers often learn basic features like edges and textures, while deeper layers learn more complex features specific to the images in the dataset. This hierarchical learning is particularly important for CIFAR-100, where the ability to discern fine-grained differences between similar categories can significantly impact classification accuracy. Despite its depth, ResNet50 is relatively efficient to train. The residual connections help in propagating gradients throughout the network assisting in faster and more effective training. This efficiency is crucial when working with extensive datasets, where training time and computational resources can have specific constraints.

The ResNet50 architecture uses base layers that were frozen and initialized using ImageNet weights. To train on the ResNet50 model the images are typically 224x224 pixels. This presented a challenge with the CIFAR-100 dataset since is consists of 32x32 pixel images. Directly inputting these smaller images into ResNet50 would be inappropriate due to the size mismatch. To address these issues, an UpSampling2D layer, was added using a size of 7x7 which results in each local neighborhood in the input being expanded to a 7x7 region in the output. Bilinear interpolation is used to compute the values of the pixels in the output feature map based on the values of surrounding pixels in the input feature map. This resized the smaller images to an input shape of (224,224,3) making it a compatible size for ResNet50 to be trained on. [11]

Following this, a GlobalAveragePooling2D layer is applied to reduce the spatial dimensions of the feature maps to a single value per feature map. A dropout layer with a tunable rate is then incorporated to reduce overfitting. The data is further processed with a dense layer of 256 units and an ReLU activation function. This is followed by a batch normalization layer. Batch normalization layers were incorporated to enhance the network's performance and stability as it searches deeper by helping it converge. The final touch involves passing the output through a dense layer consisting of 100 units, employing the Softmax activation function. [12]

Table 4 ResNet Training Data

Dropout	Learning Rate	Epoch Stopped	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
None	0.1	24	1.2681	74.46%	3.4204	58.39%
None	0.01	13	0.2011	93.18%	1.5910	69.75%
None	0.001	14	0.0637	98.17%	1.4429	72.65%
None	0.0001	24	0.1195	98.56%	1.0609	72.13%
0.25	0.1	28	1.3124	67.56%	1.8961	61.06%
0.25	0.01	16	0.3102	89.46%	1.0400	74.77%
0.25	0.001	17	0.2036	93.02%	0.9481	76.90%
0.25	0.0001	41	0.3082	90.35%	0.8042	76.91%
0.50	0.1	31	1.7009	55.43%	2.0214	51.45%
0.50	0.01	19	0.4600	84.69%	0.9668	74.61%
0.50	0.001	21	0.3680	87.38%	0.7613	78.17%
0.50	0.0001	50+	0.4959	84.26%	0.7196	78.25%

Table 4 contains the results from various configurations of the ResNet model training. It appears that lower learning rates with a reasonable dropout of 0.25 resulted in the best validation accuracy. This shows the model is effective at regularization without compromising the ability to learn from the training data. In particular, the combination of a 0.001 learning rate and 0.25 dropout rate achieves a balance between overfitting control and learning efficiency. This is indicated by the highest validation accuracy of 76.91% and accuracy 90.35% in figure 8.

Despite high training accuracy the diminishing returns at a very low learning rate of 0.0001 suggest that the model is overfitting to the training data. This is further reinforced by the lower validation accuracy. Conversely, higher dropout rates seem to lead to underfitting, where the model is unable to learn sufficiently from the training data. This can be observed by the lower training and validation accuracies. These findings underscore the importance of tuning hyperparameters to match the complexity of the dataset and the capacity of the model. [10]

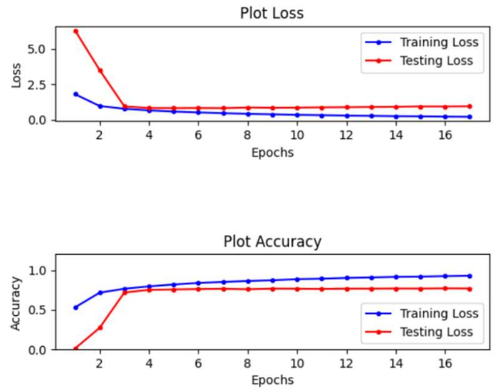

Fig.8:ResNet model training

Section III: Conclusion & Future Works

Conclusion

This research explored the efficacy of various deep learning architectures for multiclass image classification using the CIFAR-100 dataset, focusing on Dense Neural Networks, Convolutional Neural Networks, and Residual Networks. The study revealed that while Dense Neural Networks struggle with image classification due to limited feature interpretation capabilities, CNNs excel in capturing spatial features. However, CNNs faced overfitting challenges, requiring regularization techniques. ResNets, with their residual blocks, proved the most effective in handling the dataset's complexity by demonstrating superior performance through optimized hyperparameter tuning. The comparative analysis underscores the strengths and limitations of each architecture, with ResNets emerging as the most robust model is shown in Table 5. The research highlights the critical role of hyperparameter tuning in enhancing model performance and sets the stage for future integration of dimensionality reduction techniques like PCA to further improve deep learning models.

Table 5 Conclusion Data

Model Type	Best Dropout Rate	Optimal Learning Rate	Training Accuracy	Testing Accuracy
Dense Neural Network	None	0.001	44.75%	23.59%
Convolutional Neural Network	0.4	0.0001	70%	56%
Residual Network (ResNet)	0.25	0.001	93.02%	76.91%

Future Work

PCA

This exploration lays the groundwork for future research endeavors at the intersection of dimensionality reduction and neural network modeling. Principal Component Analysis (PCA) serves as a pivotal tool in the realm of dimensionality reduction, particularly when applied to intricate datasets such as CIFAR-100. The preprocessing steps include normalization and flattening of the images. PCA is then applied to extract the data into its principal components and converting the data into its principal components. This allows for enabling more efficient training and interpretation of neural networks. This approach aims to balance the trade-off between preserving meaningful data features and reducing computational complexity, thereby enhancing the effectiveness and efficiency of machine learning models in handling high-dimensional data.

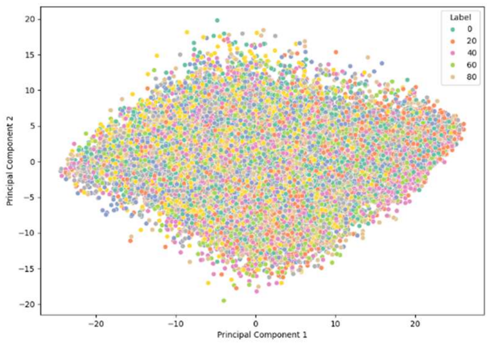

Fig.8: 2D Scatter Plot of CIFAR-100 with PCA

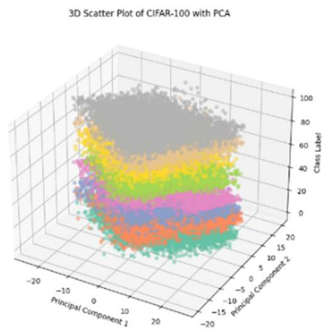

Fig.9: 3D Scatter Plot of CIFAR-100 with PCA

The incorporation of PCA-transformed data into neural network architectures not only prompts future inquiries into the impact of dimensionality reduction on training dynamics and model generalization but also establishes a foundation for comprehending the intricate relationship between these techniques. This study sets the stage for subsequent investigations aiming to optimize these approaches for enhanced image classification, acknowledging the current resource demand as a potential limitation.

Description of Student Participation

The successful completion of the project was a true collaboration of teamwork. Every team member actively participated and contributed their knowledge and skills to achieve a common goal.

References

[1] Alex Krizhevsky. “Learning Multiple Layers of Features from Tiny Images”. In: University of Toronto (May 2012). (Dataset CIFAR-100)

[2] CIFAR-10 and CIFAR-100 datasets, https://www.cs.toronto.edu/~kriz/cifar.html (accessed Dec. 11, 2023).

[3] Manarabdelaty, “Manarabdelaty/cifar100-nn-classifier,” GitHub, https://github.com/Manarabdelaty/Cifar100-NN-Classifier (accessed Dec. 11, 2023).

[4] C. Khanna, “CIFAR-100: Pre-processing for image recognition task,” Medium, https://towardsdatascience.com/cifar-100-pre-processing-for-image-recognition-task-68015b43d658 (accessed Dec. 11, 2023).

[5] Y. Verma, “A complete understanding of dense layers in neural networks,” Analytics India Magazine, https://analyticsindiamag.com/a-complete-understanding-of-dense-layers-in-neural-networks/ (accessed Dec. 11, 2023).

[6] GitHub, https://github.com/christianversloot/machine-learning-articles/blob/main/how-to-build-a-convnet-for-cifar-10-and-cifar-100-classification-with-keras.md (accessed Dec. 11, 2023).

[7] Nagamsurya, “CIFAR-10 image classification using pytorch.,” Medium, https://medium.com/@nagam808surya/cifar-10-image-classification-using-pytorch-90332041ba1f (accessed Dec. 11, 2023).

[8] Second-order step-size tuning of SGD for non-convex optimization, https://www.researchgate.net/publication/358130155_Second-Order_Step-Size_Tuning_of_SGD_for_Non-Convex_Optimization (accessed Dec. 11, 2023).

[9] D. Orellana, “Training model to classify CIFAR100 with ResNet,” Medium, https://medium.com/@damian.c036/training-model-to-classify-cifar100-with-resnet-4512d7a596a1 (accessed Dec. 11, 2023).

[10] S. T, “What are skip connections in deep learning?,” Analytics Vidhya, https://www.analyticsvidhya.com/blog/2021/08/all-you-need-to-know-about-skip-connections/ (accessed Dec. 11, 2023).

[11] CIFAR-10 and CIFAR-100 datasets, https://www.cs.toronto.edu/~kriz/cifar.html (accessed Dec. 11, 2023).

[12] GitHub, https://github.com/christianversloot/machine-learning-articles/blob/main/how-to-build-a-convnet-for-cifar-10-and-cifar-100-classification-with-keras.md (accessed Dec. 11, 2023).