THE UNIVERSITY OF MEMPHIS

COMP8740-Neural Networks

Department of Computer Science

Lokesh Das, Navid Imran, Sahil Nowkhal

U00740183, U00761100, U00741712

Impact of Custom Activation Functions on Image Classification Tasks using Deep CNN

December 5, 2021

Introduction

Image classification is one of the core problems in computer vision. In it's simplest form, it refers to the task of assigning a label to an input image from a preordained list of categories. Despite the apparent simplicity, image classification has lots of practical applications. For example, it can be used by social networks for recognition tasks, businesses for product placement and searching, medical professionals for disease diagnosis, traffic monitoring systems for controlling traffic etc.

However, the success of an image classification task depends significantly on the dataset being used and the model architecture. Traditionally these type of tasks were handled by support vector machines, logistic regression etc. but with the advent of machine learning (ML) techniques such as convolutional neural networks (CNN), image classification is mostly handled by ML methods now-a-days. Since 2012, when AlexNet[5] was first introduced, deep neural networks have become the predominant approach for image classification. Since then, various architectures based on CNN has been proposed such as VGG[7], Inception v1[8], v2 and v3 [9], and ResNet[4] etc. which improved accuracy on classification tasks substantially.

Still, the progress did not only come from upgraded architectures. Changes in activation functions and loss functions, newer data preprocessing methods and optimization procedures also performed a vital role. In this project, we will explore the impact of using different custom activation functions on the ResNet50 architecture and then the obtained results will be compared with a model trained using transfer learning. For the transfer learning, the base model will be based on ResNet50 using pretrained weights from Imagenet and then the model will be fine-tuned for better accuracy. Although the original ResNet50 was trained using the Imagenet dataset, that is very computationally intensive, therefore, our models will be trained with the fruits 360 dataset, which is publicly available in Kaggle.

Methodology

- Import necessary packages and libraries
- Mount Google Colab
- Import the dataset
- Define the training, validation and testing data
- Load pretrained model weights for transfer learning

- Alternatively build model layer by layer for regular ResNet50
- Freeze some layers of the pretrained model and compile the model
- Perform data augmentation if needed
- Train the model with specified number of epochs and other parameters
- Save the model for future predictions
- Evaluate the model for loss and accuracy
- Make predictions for the test data and compare with original

Deep Learning Architecture

Recently, neural network shows significant improvement in very complex tasks, i.e., visual recognition, speech modeling, sequence predictions, etc. The very first neural network is perceptron which has only a single unit. The neural network is a special perceptron called multi-layer perceptron with an input layer, one or more hidden layer(s), and an output layer. When the number of hidden layers is more than one, it is called a deep neural network. Convolutional Neural Networks (CNNs), the deep learning algorithms, are very powerful tools in computer vision, classification/-analysis tasks. CNN takes an image as input, assignment weights on various important features and classifies it.

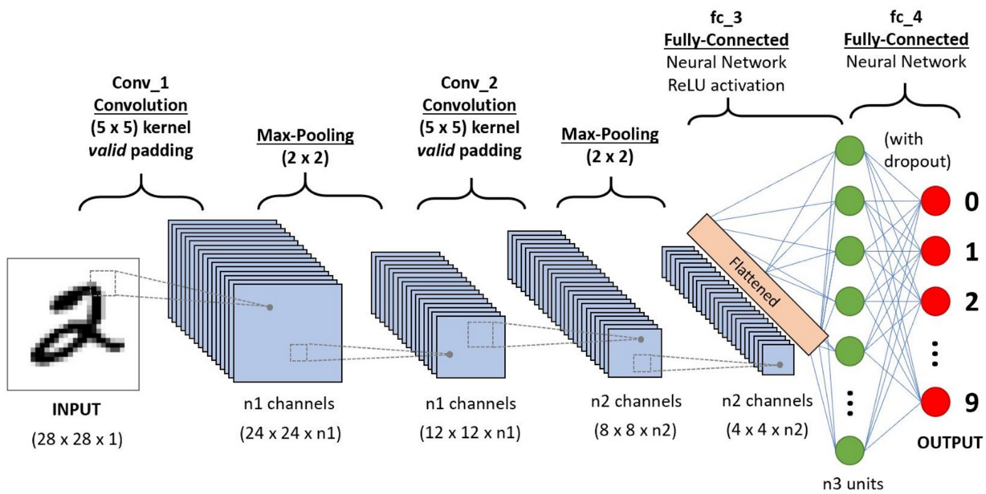

Figure 1: Deep learning architecture in image classification task[1]

CNN performs series of operations such as convolution operation with kernel weights, subsampling to reduce the number of parameters and some other regularizations, i.e., Batch Normalization, l1/l2 regularization or dropout. Figure 1 shows the deep learning architecture in image recognition task.In this project, we have used ResNet50 as our base structure that has very good accuracy in image classification tasks.

Residual Networks (ResNets) are CNNs that can handle an exceptionally large number of layers and yet avoid the vanishing gradient and model performance degradation problem introducing shortcut-connection.Figure 2 depicts the deep architecture of ResNet50. ResNets perform identity shortcuts if the input and output dimensions are the same. When the dimensions increase because of Convolutional-Batch Normalization operations, the skip-connection performs either identity mapping with extra zero entities padded to increase the dimension introducing no extra parameters or shortcut projections done by 1x1 convolution. The makers of the first ResNets, in [3], conjectured rightly that these identity connections are closer to zero and easier to learn than the original functions.

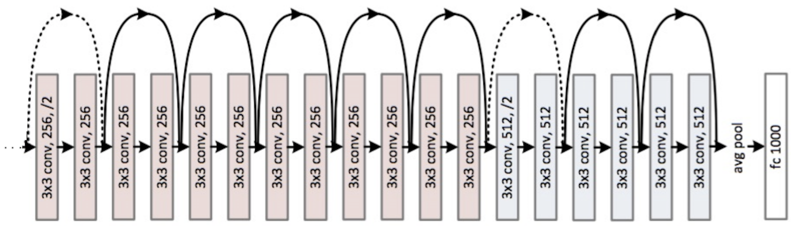

Figure 2: ResNet50 Architecture

Skip-connections(or shortcut-connections) are network connections that skip the one or more layers in a ResNet 'building block' as shown in Fig 3.A ResNet is comprised of several such blocks. The

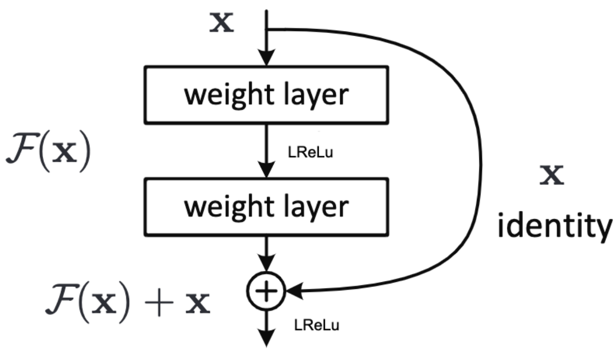

Figure 3: Residual Learning: a building block[3]

skip-connections directly map the input to the output, which is called identity mapping, and the outputs of the skip-connection are added to the output of the stacked layers in the building block as shown in figure 3. The skip-connection can allow deeper layers to act like shallower layers, and the network can pass all of its learned features to deeper layers. These connections add neither extra parameters nor computational complexity and cost us nothing. The only potential restriction they cause to the network is to have the same dimensions as the convolutional layers they skip.

Experimental Setup

The experiment was carried out on Jupyter Notebook hosted on Google colab, in an environment of Keras 2.7.0 with Tensorflow 2.7.0 backend. The RAM and GPU was allocated by Google colab on the fly.

Dataset:

The dataset being used is the fruit 360 dataset imported from Kaggle. This dataset contains 90483 images of fruits or vegetables in a 100×100 pixel size that belongs to 131 classes. Among them $67,692$ are training images and $22,688$ are test images. For our purposes, the images were resized into 32×32 pixels.

ResNet50 with transfer learning:

In transfer learning initially a base network is trained on a certain dataset and the features learned from this task are re-purposed to a second network to train on a second dataset. This method only works if the features have some similarity between both the base and target tasks. As we know, deploying pre-trained models on similar data usually generate good results in image classification related tasks [6]. In this experiment, we used the pre-trained weights for Imagenet dataset in the ResNet50 model. Then we removed the top layer of ResNet50 and added our own layers with global average pooling and dense layers with the target number of classes. This modified model is trained with the fruit dataset after freezing all the layers except the last three of the base model. The parameters used for training are the following:

Parameters	**Values**
Batch Size	256/512
Optimizer	Adam
Learning Rate	0.001
Loss	Categorical Crossentropy
Number of Epoch	100

Table 1: Hyperparameters for ResNet50 with transfer learning

ResNet50 with custom activation function:

We have developed and trained ResNet50 from scratch with default skip-connection. Each block of ResNet50 has Conv2D-BatchNormalization-Relu form, but in our implementation, we define a custom activation function "***LReLu***" by modifying the existing ReLu activation function. The equation 1 defines how LReLu is formed. Rectified Linear Unit (ReLU) suffers from the vanishing gradient problem and might also have the exploding gradient problem. The explosion is caused by continually multiplying gradients through network layers with values greater than 1.0, resulting in exponential growth[2]. However, the **'LReLU'** will prevent the network from suffering gradient exploding problems as every time we are multiplying the positive values of weight with β whose value should be less than 1.0. Initially, we assumed the β value between 0.85 to 1.0, but it can be any positive value less than 1.0. We also consider the gradient vanishing problem as defined in 'LeakyReLu' and we define α as a parameter to tackle this problem. The α value is assumed to be close to 0.0 but greater than 0.0. Figure 4 shows how **'LReLu'** is scaled up for different values of α and β.

$$R(X) = \begin{cases} \alpha\text{X} & \text{if X}\leq 0 \\ \beta\text{X} & \text{otherwise} \end{cases} \tag{1}$$

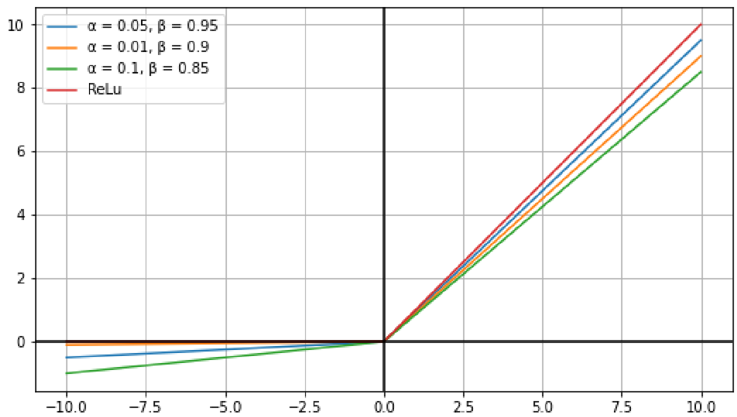

Figure 4: Custom Activation function: LReLu

Results and Discussion

To observe how the use of a custom activation function influences the performance of ResNet50, we first implemented a custom model by transferring the training knowledge from Imagenet and fine-tuned it. As can be seen from Figure 5 and 6, we obtained a competitive training accuracy of 95%. The test accuracy was 94.27%. The time taken to run each epoch was around 75 seconds in

Google colab.

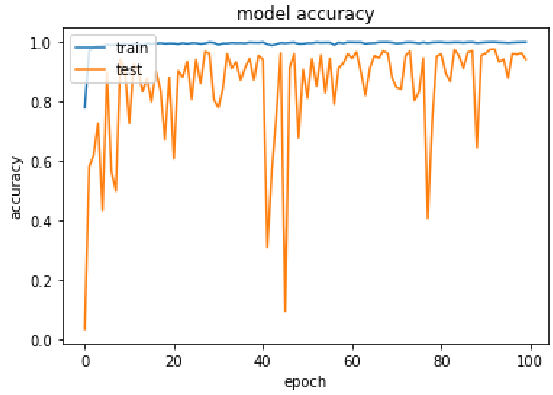

Figure 5: Accuracy (Transfer Learning)

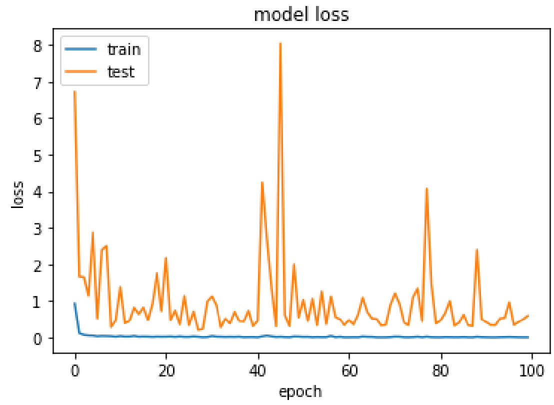

Figure 6: Loss(Transfer Learning)

Then we developed the ResNet50 model from scratch using the **LReLU** custom activation function. We tested with three different versions of **LReLU**. The best training and validation accuracy for this model was extremely good at 99.98% and 94.19% (see Figure 7, 8). In this case, the time to run each epoch varied a lot since the models were run on different machines and in different time which meant they had access to different amount of resources at Google colab. However, in general they took between 75-105 seconds. Finally, the highest testing accuracy was 94.83% which was achieved with $\alpha = 0.001, \beta = 0.95$.

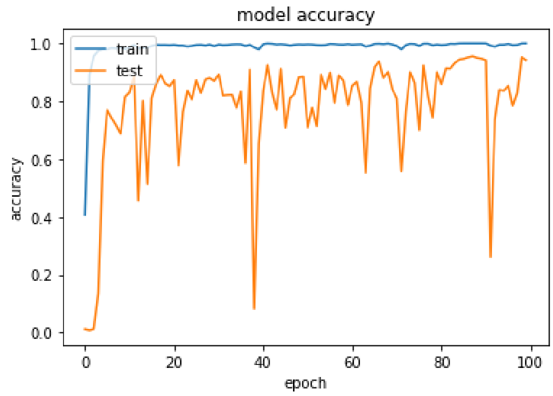

Figure 7: Accuracy (Custom Activation)

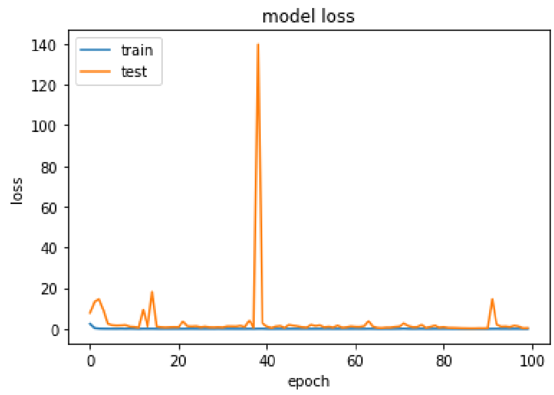

Figure 8: Loss(Custom Activation)

The different training and testing accuracy and losses are summarized in table 2:

Methods	**Training Accuracy**	**Testing Accuracy**	**Training Loss**	**Testing Loss**
Transfer Learning (Fine Tuning)	99.98%	94.27%	0.0012	0.54
ResNet50 Custom Activation ($\alpha = 0.00001$, $\beta = 0.99995$)	99.41%	86.98%	0.0210	0.7557
ResNet50 Custom Activation ($\alpha = 0.001$, $\beta = 0.95$)	99.98%	94.83%	0.00078	0.3281
ResNet50 Custom Activation ($\alpha = 0.02$, $\beta = 0.90$)	99.57%	93.53%	0.0151	0.3436

Table 2: Performance comparison of Transfer Learning and Custom Activation models

As we can see from our training and validation accuracy plots, there are lots of spikes in the output. It is possible that there exists some outliers in the mini-batches during gradient descent. Another possible reason is that the model needed to be trained for more epochs to converge smoothly. Unfortunately, in our experiment we were able to only run 100 epochs for each model because of resource and time constraint.

Conclusion

Image classification is one of the most important applications of CNN architecture. In this project, we developed a custom activation function **LReLu** which is based on the widely utilized ReLu function, then we trained the popular ResNet50 model from scratch using this. Our proposed model with LReLu activation function achieved excellent accuracy on the test dataset of fruit-360. To evaluate the performance of our model, we also trained another ResNet50 model using the pretrained weights from Imagenet and we were able to show that our model with custom activation performed slightly better. It was observed that using different values for α and β had a significant impact on the final testing accuracy. It is possible that using an ensemble of models can achieve even higher accuracy on this dataset. We also believe that given enough time, it may become possible to find an optimal value for α and β that produces the state-of-the-art results for this dataset in this architecture.

References

[1] *deep CNN architecture*. `https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53`. Accessed: 2021-12-05.

[2] *Gradient Exploding*. `https://analyticsindiamag.com/can-relu-cause-exploding-gradients-if-applied-to-solve-vanishing-gradients/`. Accessed: 2021-12-05.

[3] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: `1512.03385`. URL: `http://arxiv.org/abs/1512.03385`.

[4] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[6] A Sai Bharadwaj Reddy and D Sujitha Juliet. "Transfer learning with ResNet-50 for malaria cell-image classification". In: *2019 International Conference on Communication and Signal Processing (ICCSP)*. IEEE. 2019, pp. 0945–0949.

[7] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[8] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[9] Christian Szegedy et al. "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

A Appendix (Transfer Learning)

Epoch 1/100

212/212 - 88s 367ms/step - loss: 0.9275 - accuracy: 0.7798 - val_loss: 6.7068 - val_accuracy: 0.0354

Epoch 2/100

212/212 - 77s 361ms/step - loss: 0.1161 - accuracy: 0.9696 - val_loss: 1.6623 - val_accuracy: 0.5820

Epoch 3/100

212/212 - 75s 354ms/step - loss: 0.0698 - accuracy: 0.9827 - val_loss: 1.6466 - val_accuracy: 0.6188

Epoch 4/100

212/212 - 77s 361ms/step - loss: 0.0545 - accuracy: 0.9859 - val_loss: 1.1396 - val_accuracy: 0.7263

Epoch 5/100

212/212 - 77s 361ms/step - loss: 0.0501 - accuracy: 0.9875 - val_loss: 2.8755 - val_accuracy: 0.4348

Epoch 6/100

212/212 - 77s 361ms/step - loss: 0.0318 - accuracy: 0.9915 - val_loss: 0.5203 - val_accuracy: 0.8985

Epoch 7/100

212/212 - 75s 353ms/step - loss: 0.0405 - accuracy: 0.9898 - val_loss: 2.3993 - val_accuracy: 0.5652

Epoch 8/100

212/212 - 75s 355ms/step - loss: 0.0369 - accuracy: 0.9905 - val_loss: 2.5059 - val_accuracy: 0.4994

Epoch 9/100

212/212 - 75s 354ms/step - loss: 0.0324 - accuracy: 0.9920 - val_loss: 0.2833 - val_accuracy: 0.9422

Epoch 10/100

212/212 - 75s 355ms/step - loss: 0.0182 - accuracy: 0.9954 - val_loss: 0.4710 - val_accuracy: 0.9091

Epoch 11/100

212/212 - 75s 354ms/step - loss: 0.0367 - accuracy: 0.9908 - val_loss: 1.3787 - val_accuracy: 0.7260

Epoch 12/100

212/212 - 75s 353ms/step - loss: 0.0214 - accuracy: 0.9946 - val_loss: 0.3905 - val_accuracy: 0.9254

Epoch 13/100

212/212 - 75s 353ms/step - loss: 0.0231 - accuracy: 0.9943 - val_loss: 0.4580 - val_accuracy: 0.9067

Epoch 14/100

212/212 - 75s 355ms/step - loss: 0.0406 - accuracy: 0.9893 - val_loss: 0.8177 - val_accuracy: 0.8333

Epoch 15/100

212/212 - 74s 351ms/step - loss: 0.0180 - accuracy: 0.9960 - val_loss: 0.6423 - val_accuracy: 0.8791

Epoch 16/100

212/212 - 75s 352ms/step - loss: 0.0226 - accuracy: 0.9944 - val_loss: 0.8215 - val_accuracy: 0.8002

Epoch 17/100

212/212 - 74s 351ms/step - loss: 0.0203 - accuracy: 0.9953 - val_loss: 0.4749 - val_accuracy: 0.9016

Epoch 18/100

212/212 - 76s 359ms/step - loss: 0.0130 - accuracy: 0.9969 - val_loss: 0.9217 - val_accuracy: 0.8384

Epoch 19/100

212/212 - 75s 353ms/step - loss: 0.0204 - accuracy: 0.9946 - val_loss: 1.7568 - val_accuracy: 0.6713

Epoch 20/100

212/212 - 76s 358ms/step - loss: 0.0175 - accuracy: 0.9956 - val_loss: 0.7252 - val_accuracy: 0.8810

Epoch 21/100

212/212 - 77s 362ms/step - loss: 0.0186 - accuracy: 0.9952 - val_loss: 2.1814 - val_accuracy: 0.6081

Epoch 22/100

212/212 - 75s 355ms/step - loss: 0.0238 - accuracy: 0.9932 - val_loss: 0.4754 - val_accuracy: 0.9033

Epoch 23/100

212/212 - 75s 354ms/step - loss: 0.0129 - accuracy: 0.9967 - val_loss: 0.7394 - val_accuracy: 0.8824

Epoch 24/100

212/212 - 74s 350ms/step - loss: 0.0262 - accuracy: 0.9942 - val_loss: 0.3581 - val_accuracy: 0.9351

Epoch 25/100

212/212 - 75s 352ms/step - loss: 0.0132 - accuracy: 0.9967 - val_loss: 1.1334 - val_accuracy: 0.8075

Epoch 26/100

212/212 - 75s 356ms/step - loss: 0.0133 - accuracy: 0.9966 - val_loss: 0.3364 - val_accuracy: 0.9408

Epoch 27/100

212/212 - 74s 350ms/step - loss: 0.0268 - accuracy: 0.9937 - val_loss: 0.7119 - val_accuracy: 0.8617

Epoch 28/100

212/212 - 75s 354ms/step - loss: 0.0168 - accuracy: 0.9958 - val_loss: 0.2051 - val_accuracy: 0.9676

Epoch 29/100

212/212 - 74s 351ms/step - loss: 0.0026 - accuracy: 0.9996 - val_loss: 0.2316 - val_accuracy: 0.9615

Epoch 30/100

212/212 - 80s 376ms/step - loss: 0.0086 - accuracy: 0.9978 - val_loss: 0.9875 - val_accuracy: 0.8074

Epoch 31/100

212/212 - 75s 354ms/step - loss: 0.0382 - accuracy: 0.9909 - val_loss: 1.1249 - val_accuracy: 0.7793

Epoch 32/100

212/212 - 75s 354ms/step - loss: 0.0189 - accuracy: 0.9956 - val_loss: 0.8742 - val_accuracy: 0.8422

Epoch 33/100

212/212 - 75s 354ms/step - loss: 0.0171 - accuracy: 0.9955 - val_loss: 0.2778 - val_accuracy: 0.9599

Epoch 34/100

212/212 - 76s 358ms/step - loss: 0.0115 - accuracy: 0.9977 - val_loss: 0.5168 - val_accuracy: 0.9115

Epoch 35/100

212/212 - 75s 352ms/step - loss: 0.0156 - accuracy: 0.9967 - val_loss: 0.3807 - val_accuracy: 0.9336

Epoch 36/100

212/212 - 75s 353ms/step - loss: 0.0118 - accuracy: 0.9970 - val_loss: 0.7011 - val_accuracy: 0.8720

Epoch 37/100

212/212 - 75s 353ms/step - loss: 0.0185 - accuracy: 0.9962 - val_loss: 0.4509 - val_accuracy: 0.9133

Epoch 38/100

212/212 - 76s 358ms/step - loss: 0.0041 - accuracy: 0.9990 - val_loss: 0.4405 - val_accuracy: 0.9449

Epoch 39/100

212/212 - 75s 355ms/step - loss: 0.0078 - accuracy: 0.9983 - val_loss: 0.7345 - val_accuracy: 0.8721

Epoch 40/100

212/212 - 75s 354ms/step - loss: 0.0082 - accuracy: 0.9980 - val_loss: 0.3105 - val_accuracy: 0.9547

Epoch 41/100

212/212 - 75s 354ms/step - loss: 0.0017 - accuracy: 0.9996 - val_loss: 0.4543 - val_accuracy: 0.9413

Epoch 42/100

212/212 - 75s 354ms/step - loss: 0.0281 - accuracy: 0.9929 - val_loss: 4.2406 - val_accuracy: 0.3098

Epoch 43/100

212/212 - 77s 361ms/step - loss: 0.0487 - accuracy: 0.9890 - val_loss: 2.7077 - val_accuracy: 0.5687

Epoch 44/100

212/212 - 76s 356ms/step - loss: 0.0288 - accuracy: 0.9928 - val_loss: 1.3759 - val_accuracy: 0.7498

Epoch 45/100

212/212 - 75s 354ms/step - loss: 0.0103 - accuracy: 0.9976 - val_loss: 0.3012 - val_accuracy: 0.9627

Epoch 46/100

212/212 - 76s 356ms/step - loss: 0.0169 - accuracy: 0.9964 - val_loss: 8.0376 - val_accuracy: 0.0952

Epoch 47/100

212/212 - 76s 359ms/step - loss: 0.0079 - accuracy: 0.9980 - val_loss: 0.6161 - val_accuracy: 0.9136

Epoch 48/100

212/212 - 75s 354ms/step - loss: 0.0035 - accuracy: 0.9991 - val_loss: 0.3118 - val_accuracy: 0.9601

Epoch 49/100

212/212 - 75s 352ms/step - loss: 0.0244 - accuracy: 0.9943 - val_loss: 1.9916 - val_accuracy: 0.6777

Epoch 50/100

212/212 - 74s 350ms/step - loss: 0.0213 - accuracy: 0.9943 - val_loss: 0.5517 - val_accuracy: 0.9077

Epoch 51/100

212/212 - 76s 357ms/step - loss: 0.0130 - accuracy: 0.9968 - val_loss: 1.0278 - val_accuracy: 0.8114

Epoch 52/100

212/212 - 74s 350ms/step - loss: 0.0144 - accuracy: 0.9968 - val_loss: 0.4520 - val_accuracy: 0.9436

Epoch 53/100

212/212 - 75s 355ms/step - loss: 0.0036 - accuracy: 0.9993 - val_loss: 1.0600 - val_accuracy: 0.8521

Epoch 54/100

212/212 - 75s 353ms/step - loss: 0.0098 - accuracy: 0.9981 - val_loss: 0.3365 - val_accuracy: 0.9567

Epoch 55/100

212/212 - 76s 360ms/step - loss: 0.0058 - accuracy: 0.9985 - val_loss: 1.2593 - val_accuracy: 0.8292

Epoch 56/100

212/212 - 75s 352ms/step - loss: 0.0070 - accuracy: 0.9985 - val_loss: 0.3661 - val_accuracy: 0.9447

Epoch 57/100

212/212 - 75s 353ms/step - loss: 0.0416 - accuracy: 0.9902 - val_loss: 1.1219 - val_accuracy: 0.7915

Epoch 58/100

212/212 - 75s 356ms/step - loss: 0.0049 - accuracy: 0.9986 - val_loss: 0.5594 - val_accuracy: 0.9130

Epoch 59/100

212/212 - 76s 359ms/step - loss: 0.0173 - accuracy: 0.9963 - val_loss: 0.4957 - val_accuracy: 0.9273

Epoch 60/100

212/212 - 76s 357ms/step - loss: 0.0015 - accuracy: 0.9996 - val_loss: 0.3408 - val_accuracy: 0.9599

Epoch 61/100

212/212 - 75s 356ms/step - loss: 0.0021 - accuracy: 0.9995 - val_loss: 0.4777 - val_accuracy: 0.9453

Epoch 62/100

212/212 - 75s 353ms/step - loss: 0.0042 - accuracy: 0.9991 - val_loss: 0.3568 - val_accuracy: 0.9662

Epoch 63/100

212/212 - 76s 360ms/step - loss: 0.0030 - accuracy: 0.9992 - val_loss: 0.6303 - val_accuracy: 0.8974

Epoch 64/100

212/212 - 76s 358ms/step - loss: 0.0236 - accuracy: 0.9946 - val_loss: 1.0890 - val_accuracy: 0.8204

Epoch 65/100

212/212 - 76s 358ms/step - loss: 0.0154 - accuracy: 0.9963 - val_loss: 0.6987 - val_accuracy: 0.9115

Epoch 66/100

212/212 - 76s 358ms/step - loss: 0.0134 - accuracy: 0.9969 - val_loss: 0.5201 - val_accuracy: 0.9536

Epoch 67/100

212/212 - 77s 361ms/step - loss: 0.0016 - accuracy: 0.9996 - val_loss: 0.5077 - val_accuracy: 0.9466

Epoch 68/100

212/212 - 76s 356ms/step - loss: 6.4529e-04 - accuracy: 0.9999 - val_loss: 0.3298 - val_accuracy: 0.9695 Epoch 69/100

212/212 - 75s 354ms/step - loss: 0.0016 - accuracy: 0.9998 - val_loss: 0.3470 - val_accuracy: 0.9617

Epoch 70/100

212/212 - 76s 357ms/step - loss: 0.0075 - accuracy: 0.9985 - val_loss: 0.8877 - val_accuracy: 0.8788

Epoch 71/100

212/212 - 77s 363ms/step - loss: 0.0208 - accuracy: 0.9958 - val_loss: 1.2057 - val_accuracy: 0.8476

Epoch 72/100

212/212 - 76s 358ms/step - loss: 0.0186 - accuracy: 0.9966 - val_loss: 0.9149 - val_accuracy: 0.8424

Epoch 73/100

212/212 - 75s 356ms/step - loss: 0.0037 - accuracy: 0.9991 - val_loss: 0.4087 - val_accuracy: 0.9550

Epoch 74/100

212/212 - 76s 359ms/step - loss: 0.0026 - accuracy: 0.9995 - val_loss: 0.3375 - val_accuracy: 0.9688

Epoch 75/100

212/212 - 76s 359ms/step - loss: 0.0067 - accuracy: 0.9985 - val_loss: 1.0968 - val_accuracy: 0.8036

Epoch 76/100

212/212 - 75s 355ms/step - loss: 0.0186 - accuracy: 0.9961 - val_loss: 1.3376 - val_accuracy: 0.8312

Epoch 77/100

212/212 - 76s 358ms/step - loss: 0.0031 - accuracy: 0.9993 - val_loss: 0.4463 - val_accuracy: 0.9462

Epoch 78/100

212/212 - 75s 356ms/step - loss: 0.0195 - accuracy: 0.9967 - val_loss: 4.0699 - val_accuracy: 0.4068

Epoch 79/100

212/212 - 75s 354ms/step - loss: 0.0046 - accuracy: 0.9990 - val_loss: 1.4633 - val_accuracy: 0.7396

Epoch 80/100

212/212 - 76s 360ms/step - loss: 0.0021 - accuracy: 0.9995 - val_loss: 0.3810 - val_accuracy: 0.9521

Epoch 81/100

212/212 - 76s 359ms/step - loss: 0.0014 - accuracy: 0.9998 - val_loss: 0.4720 - val_accuracy: 0.9596

Epoch 82/100

212/212 - 76s 356ms/step - loss: 0.0080 - accuracy: 0.9984 - val_loss: 0.6706 - val_accuracy: 0.8965

Epoch 83/100

212/212 - 75s 353ms/step - loss: 0.0061 - accuracy: 0.9986 - val_loss: 0.9987 - val_accuracy: 0.8672

Epoch 84/100

212/212 - 76s 357ms/step - loss: 0.0032 - accuracy: 0.9993 - val_loss: 0.3236 - val_accuracy: 0.9744

Epoch 85/100

212/212 - 75s 356ms/step - loss: 0.0042 - accuracy: 0.9992 - val_loss: 0.4056 - val_accuracy: 0.9536

Epoch 86/100

212/212 - 75s 355ms/step - loss: 0.0087 - accuracy: 0.9982 - val_loss: 0.6220 - val_accuracy: 0.9115

Epoch 87/100

212/212 - 75s 356ms/step - loss: 0.0025 - accuracy: 0.9995 - val_loss: 0.3391 - val_accuracy: 0.9653

Epoch 88/100

212/212 - 77s 363ms/step - loss: 6.9986e-04 - accuracy: 0.9998 - val_loss: 0.3102 - val_accuracy: 0.9705 Epoch 89/100

212/212 - 76s 360ms/step - loss: 0.0164 - accuracy: 0.9972 - val_loss: 2.3999 - val_accuracy: 0.6452

Epoch 90/100

212/212 - 77s 362ms/step - loss: 0.0068 - accuracy: 0.9983 - val_loss: 0.4985 - val_accuracy: 0.9550

Epoch 91/100

212/212 - 76s 358ms/step - loss: 0.0031 - accuracy: 0.9994 - val_loss: 0.4201 - val_accuracy: 0.9623

Epoch 92/100

212/212 - 76s 358ms/step - loss: 5.5491e-05 - accuracy: 1.0000 - val_loss: 0.3408 - val_accuracy: 0.9753 Epoch 93/100

212/212 - 76s 357ms/step - loss: 1.9292e-05 - accuracy: 1.0000 - val_loss: 0.3391 - val_accuracy: 0.9758 Epoch 94/100

212/212 - 76s 356ms/step - loss: 0.0048 - accuracy: 0.9990 - val_loss: 0.5196 - val_accuracy: 0.9330

Epoch 95/100

212/212 - 76s 356ms/step - loss: 0.0076 - accuracy: 0.9984 - val_loss: 0.5356 - val_accuracy: 0.9422

Epoch 96/100

212/212 - 76s 360ms/step - loss: 0.0113 - accuracy: 0.9975 - val_loss: 0.9627 - val_accuracy: 0.8779

Epoch 97/100

212/212 - 76s 357ms/step - loss: 0.0078 - accuracy: 0.9984 - val_loss: 0.3409 - val_accuracy: 0.9606

Epoch 98/100

212/212 - 76s 360ms/step - loss: 0.0029 - accuracy: 0.9993 - val_loss: 0.4246 - val_accuracy: 0.9588

Epoch 99/100

212/212 - 76s 359ms/step - loss: 0.0013 - accuracy: 0.9996 - val_loss: 0.4960 - val_accuracy: 0.9636

Epoch 100/100

212/212 - 76s 361ms/step - loss: 0.0012 - accuracy: 0.9998 - val_loss: 0.5943 - val_accuracy: 0.9425

B Appendix (ResNet50 with Custom Activation)

Epoch 1/100

106/106 - 95s 737ms/step - loss: 2.3594 - accuracy: 0.4069 - val_loss: 7.8124 - val_accuracy: 0.0109

Epoch 2/100

106/106 - 75s 711ms/step - loss: 0.3541 - accuracy: 0.8918 - val_loss: 13.2238 - val_accuracy: 0.0071

Epoch 3/100

106/106 - 75s 704ms/step - loss: 0.1413 - accuracy: 0.9553 - val_loss: 14.5815 - val_accuracy: 0.0117

Epoch 4/100

106/106 - 74s 700ms/step - loss: 0.0851 - accuracy: 0.9743 - val_loss: 9.3073 - val_accuracy: 0.1341

Epoch 5/100

106/106 - 74s 702ms/step - loss: 0.0691 - accuracy: 0.9796 - val_loss: 2.3396 - val_accuracy: 0.5951

Epoch 6/100

106/106 - 75s 707ms/step - loss: 0.0690 - accuracy: 0.9792 - val_loss: 1.7637 - val_accuracy: 0.7691

Epoch 7/100

106/106 - 76s 713ms/step - loss: 0.0579 - accuracy: 0.9831 - val_loss: 1.5913 - val_accuracy: 0.7411

Epoch 8/100

106/106 - 74s 700ms/step - loss: 0.0940 - accuracy: 0.9818 - val_loss: 1.6690 - val_accuracy: 0.7154

Epoch 9/100

106/106 - 75s 704ms/step - loss: 0.0545 - accuracy: 0.9857 - val_loss: 1.7985 - val_accuracy: 0.6867

Epoch 10/100

106/106 - 75s 705ms/step - loss: 0.0899 - accuracy: 0.9794 - val_loss: 1.0495 - val_accuracy: 0.8145

Epoch 11/100

106/106 - 77s 724ms/step - loss: 0.0611 - accuracy: 0.9856 - val_loss: 0.9097 - val_accuracy: 0.8299

Epoch 12/100

106/106 - 75s 706ms/step - loss: 0.0210 - accuracy: 0.9946 - val_loss: 0.6217 - val_accuracy: 0.8853

Epoch 13/100

106/106 - 75s 704ms/step - loss: 0.0763 - accuracy: 0.9819 - val_loss: 9.4272 - val_accuracy: 0.4558

Epoch 14/100

106/106 - 75s 703ms/step - loss: 0.0607 - accuracy: 0.9845 - val_loss: 1.1839 - val_accuracy: 0.8014

Epoch 15/100

106/106 - 76s 714ms/step - loss: 0.0670 - accuracy: 0.9844 - val_loss: 18.1424 - val_accuracy: 0.5133

Epoch 16/100

106/106 - 74s 702ms/step - loss: 0.0332 - accuracy: 0.9907 - val_loss: 1.1784 - val_accuracy: 0.8101

Epoch 17/100

106/106 - 74s 703ms/step - loss: 0.0198 - accuracy: 0.9948 - val_loss: 0.8110 - val_accuracy: 0.8597

Epoch 18/100

106/106 - 74s 701ms/step - loss: 0.0228 - accuracy: 0.9942 - val_loss: 0.6919 - val_accuracy: 0.8908

Epoch 19/100

106/106 - 75s 706ms/step - loss: 0.0221 - accuracy: 0.9943 - val_loss: 0.7879 - val_accuracy: 0.8618

Epoch 20/100

106/106 - 74s 699ms/step - loss: 0.0256 - accuracy: 0.9936 - val_loss: 0.8520 - val_accuracy: 0.8519

Epoch 21/100

106/106 - 74s 700ms/step - loss: 0.0215 - accuracy: 0.9945 - val_loss: 0.8244 - val_accuracy: 0.8725

Epoch 22/100

106/106 - 74s 703ms/step - loss: 0.0362 - accuracy: 0.9923 - val_loss: 3.5651 - val_accuracy: 0.5764

Epoch 23/100

106/106 - 76s 714ms/step - loss: 0.0401 - accuracy: 0.9921 - val_loss: 1.2538 - val_accuracy: 0.7621

Epoch 24/100

106/106 - 74s 700ms/step - loss: 0.0489 - accuracy: 0.9887 - val_loss: 1.1353 - val_accuracy: 0.8361

Epoch 25/100

106/106 - 74s 703ms/step - loss: 0.0372 - accuracy: 0.9911 - val_loss: 1.2606 - val_accuracy: 0.8058

Epoch 26/100

106/106 - 74s 700ms/step - loss: 0.0240 - accuracy: 0.9942 - val_loss: 0.7951 - val_accuracy: 0.8722

Epoch 27/100

106/106 - 75s 709ms/step - loss: 0.0219 - accuracy: 0.9948 - val_loss: 1.0803 - val_accuracy: 0.8279

Epoch 28/100

106/106 - 74s 700ms/step - loss: 0.0310 - accuracy: 0.9925 - val_loss: 0.7647 - val_accuracy: 0.8748

Epoch 29/100

106/106 - 78s 733ms/step - loss: 0.0169 - accuracy: 0.9956 - val_loss: 0.7189 - val_accuracy: 0.8797

Epoch 30/100

106/106 - 74s 702ms/step - loss: 0.0493 - accuracy: 0.9902 - val_loss: 0.8090 - val_accuracy: 0.8697

Epoch 31/100

106/106 - 75s 707ms/step - loss: 0.0176 - accuracy: 0.9956 - val_loss: 0.7151 - val_accuracy: 0.8922

Epoch 32/100

106/106 - 74s 702ms/step - loss: 0.0266 - accuracy: 0.9936 - val_loss: 1.2180 - val_accuracy: 0.8192

Epoch 33/100

106/106 - 74s 701ms/step - loss: 0.0221 - accuracy: 0.9942 - val_loss: 1.2005 - val_accuracy: 0.8210

Epoch 34/100

106/106 - 74s 699ms/step - loss: 0.0164 - accuracy: 0.9959 - val_loss: 1.1509 - val_accuracy: 0.8217

Epoch 35/100

106/106 - 75s 703ms/step - loss: 0.0137 - accuracy: 0.9967 - val_loss: 1.4874 - val_accuracy: 0.7761

Epoch 36/100

106/106 - 75s 711ms/step - loss: 0.0148 - accuracy: 0.9963 - val_loss: 0.9142 - val_accuracy: 0.8328

Epoch 37/100

106/106 - 75s 708ms/step - loss: 0.0409 - accuracy: 0.9907 - val_loss: 4.0993 - val_accuracy: 0.5848

Epoch 38/100

106/106 - 74s 696ms/step - loss: 0.0224 - accuracy: 0.9949 - val_loss: 0.5937 - val_accuracy: 0.9093

Epoch 39/100

106/106 - 74s 701ms/step - loss: 0.0667 - accuracy: 0.9858 - val_loss: 139.6458 - val_accuracy: 0.0810 Epoch 40/100

106/106 - 75s 708ms/step - loss: 0.0805 - accuracy: 0.9784 - val_loss: 2.6266 - val_accuracy: 0.6523

Epoch 41/100

106/106 - 74s 699ms/step - loss: 0.0104 - accuracy: 0.9972 - val_loss: 1.1015 - val_accuracy: 0.8365

Epoch 42/100

106/106 - 75s 706ms/step - loss: 0.0028 - accuracy: 0.9994 - val_loss: 0.4799 - val_accuracy: 0.9244

Epoch 43/100

106/106 - 75s 707ms/step - loss: 0.0072 - accuracy: 0.9984 - val_loss: 1.1245 - val_accuracy: 0.8300

Epoch 44/100

106/106 - 75s 709ms/step - loss: 0.0182 - accuracy: 0.9959 - val_loss: 1.3893 - val_accuracy: 0.7707

Epoch 45/100

106/106 - 74s 701ms/step - loss: 0.0135 - accuracy: 0.9968 - val_loss: 0.5083 - val_accuracy: 0.9114

Epoch 46/100

106/106 - 74s 703ms/step - loss: 0.0202 - accuracy: 0.9953 - val_loss: 1.9042 - val_accuracy: 0.7079

Epoch 47/100

106/106 - 75s 709ms/step - loss: 0.0296 - accuracy: 0.9925 - val_loss: 1.5363 - val_accuracy: 0.8106

Epoch 48/100

106/106 - 76s 715ms/step - loss: 0.0248 - accuracy: 0.9946 - val_loss: 1.2481 - val_accuracy: 0.8234

Epoch 49/100

106/106 - 75s 709ms/step - loss: 0.0171 - accuracy: 0.9959 - val_loss: 0.8182 - val_accuracy: 0.8819

Epoch 50/100

106/106 - 75s 708ms/step - loss: 0.0201 - accuracy: 0.9953 - val_loss: 0.6718 - val_accuracy: 0.8839

Epoch 51/100

106/106 - 74s 702ms/step - loss: 0.0167 - accuracy: 0.9961 - val_loss: 1.9194 - val_accuracy: 0.7089

Epoch 52/100

106/106 - 75s 706ms/step - loss: 0.0199 - accuracy: 0.9956 - val_loss: 1.4124 - val_accuracy: 0.7768

Epoch 53/100

106/106 - 74s 703ms/step - loss: 0.0254 - accuracy: 0.9938 - val_loss: 1.7395 - val_accuracy: 0.7134

Epoch 54/100

106/106 - 74s 703ms/step - loss: 0.0272 - accuracy: 0.9932 - val_loss: 0.7626 - val_accuracy: 0.8919

Epoch 55/100

106/106 - 75s 703ms/step - loss: 0.0241 - accuracy: 0.9938 - val_loss: 1.0423 - val_accuracy: 0.8419

Epoch 56/100

106/106 - 76s 712ms/step - loss: 0.0101 - accuracy: 0.9979 - val_loss: 0.6594 - val_accuracy: 0.8988

Epoch 57/100

106/106 - 75s 706ms/step - loss: 0.0129 - accuracy: 0.9971 - val_loss: 1.5730 - val_accuracy: 0.7950

Epoch 58/100

106/106 - 74s 698ms/step - loss: 0.0200 - accuracy: 0.9955 - val_loss: 0.6150 - val_accuracy: 0.8891

Epoch 59/100

106/106 - 74s 699ms/step - loss: 0.0203 - accuracy: 0.9951 - val_loss: 0.7540 - val_accuracy: 0.8705

Epoch 60/100

106/106 - 74s 702ms/step - loss: 0.0150 - accuracy: 0.9970 - val_loss: 1.2150 - val_accuracy: 0.7859

Epoch 61/100

106/106 - 75s 709ms/step - loss: 0.0201 - accuracy: 0.9950 - val_loss: 0.9816 - val_accuracy: 0.8531

Epoch 62/100

106/106 - 75s 703ms/step - loss: 0.0174 - accuracy: 0.9959 - val_loss: 0.9483 - val_accuracy: 0.8668

Epoch 63/100

106/106 - 75s 703ms/step - loss: 0.0144 - accuracy: 0.9967 - val_loss: 1.3157 - val_accuracy: 0.7949

Epoch 64/100

106/106 - 75s 703ms/step - loss: 0.0492 - accuracy: 0.9884 - val_loss: 3.8500 - val_accuracy: 0.5521

Epoch 65/100

106/106 - 76s 711ms/step - loss: 0.0341 - accuracy: 0.9918 - val_loss: 1.0977 - val_accuracy: 0.8444

Epoch 66/100

106/106 - 76s 714ms/step - loss: 0.0084 - accuracy: 0.9980 - val_loss: 0.5139 - val_accuracy: 0.9176

Epoch 67/100

106/106 - 75s 707ms/step - loss: 0.0053 - accuracy: 0.9987 - val_loss: 0.4272 - val_accuracy: 0.9379

Epoch 68/100

106/106 - 75s 710ms/step - loss: 0.0119 - accuracy: 0.9971 - val_loss: 0.6435 - val_accuracy: 0.8804

Epoch 69/100

106/106 - 76s 715ms/step - loss: 0.0027 - accuracy: 0.9993 - val_loss: 0.6527 - val_accuracy: 0.9008

Epoch 70/100

106/106 - 75s 709ms/step - loss: 0.0184 - accuracy: 0.9956 - val_loss: 0.9461 - val_accuracy: 0.8396

Epoch 71/100

106/106 - 75s 709ms/step - loss: 0.0386 - accuracy: 0.9908 - val_loss: 1.1615 - val_accuracy: 0.8091

Epoch 72/100

106/106 - 75s 712ms/step - loss: 0.1039 - accuracy: 0.9787 - val_loss: 2.5956 - val_accuracy: 0.5578

Epoch 73/100

106/106 - 76s 715ms/step - loss: 0.0232 - accuracy: 0.9940 - val_loss: 1.4034 - val_accuracy: 0.7653

Epoch 74/100

106/106 - 76s 715ms/step - loss: 0.0073 - accuracy: 0.9982 - val_loss: 0.8141 - val_accuracy: 0.9002

Epoch 75/100

106/106 - 77s 731ms/step - loss: 0.0189 - accuracy: 0.9974 - val_loss: 0.8636 - val_accuracy: 0.8622

Epoch 76/100

106/106 - 75s 712ms/step - loss: 0.0427 - accuracy: 0.9910 - val_loss: 1.9252 - val_accuracy: 0.7000

Epoch 77/100

106/106 - 76s 714ms/step - loss: 0.0056 - accuracy: 0.9987 - val_loss: 0.5037 - val_accuracy: 0.9236

Epoch 78/100

106/106 - 75s 710ms/step - loss: 0.0059 - accuracy: 0.9989 - val_loss: 0.9513 - val_accuracy: 0.8178

Epoch 79/100

106/106 - 75s 704ms/step - loss: 0.0271 - accuracy: 0.9931 - val_loss: 1.5915 - val_accuracy: 0.7437

Epoch 80/100

106/106 - 75s 707ms/step - loss: 0.0192 - accuracy: 0.9955 - val_loss: 0.6666 - val_accuracy: 0.9002

Epoch 81/100

106/106 - 76s 715ms/step - loss: 0.0271 - accuracy: 0.9938 - val_loss: 0.8552 - val_accuracy: 0.8588

Epoch 82/100

106/106 - 75s 704ms/step - loss: 0.0277 - accuracy: 0.9946 - val_loss: 0.5131 - val_accuracy: 0.9133

Epoch 83/100

106/106 - 75s 703ms/step - loss: 0.0063 - accuracy: 0.9984 - val_loss: 0.4914 - val_accuracy: 0.9125

Epoch 84/100

106/106 - 74s 702ms/step - loss: 0.0095 - accuracy: 0.9979 - val_loss: 0.4204 - val_accuracy: 0.9309

Epoch 85/100

106/106 - 75s 711ms/step - loss: 0.0012 - accuracy: 0.9997 - val_loss: 0.3345 - val_accuracy: 0.9439

Epoch 86/100

106/106 - 75s 704ms/step - loss: 2.8855e-04 - accuracy: 1.0000 - val_loss: 0.2989 - val_accuracy: 0.9466 Epoch 87/100

106/106 - 74s 702ms/step - loss: 2.7905e-04 - accuracy: 0.9999 - val_loss: 0.2517 - val_accuracy: 0.9515 Epoch 88/100

106/106 - 76s 714ms/step - loss: 4.5668e-05 - accuracy: 1.0000 - val_loss: 0.2619 - val_accuracy: 0.9553 Epoch 89/100

106/106 - 76s 719ms/step - loss: 3.7078e-04 - accuracy: 0.9999 - val_loss: 0.3215 - val_accuracy: 0.9495 Epoch 90/100

106/106 - 75s 710ms/step - loss: 8.5931e-05 - accuracy: 1.0000 - val_loss: 0.2975 - val_accuracy: 0.9472 Epoch 91/100

106/106 - 75s 710ms/step - loss: 7.7349e-04 - accuracy: 0.9997 - val_loss: 0.3556 - val_accuracy: 0.9410 Epoch 92/100

106/106 - 76s 719ms/step - loss: 0.0344 - accuracy: 0.9918 - val_loss: 14.5459 - val_accuracy: 0.2606 Epoch 93/100

106/106 - 76s 715ms/step - loss: 0.0487 - accuracy: 0.9880 - val_loss: 2.1375 - val_accuracy: 0.7364 Epoch 94/100

106/106 - 76s 713ms/step - loss: 0.0194 - accuracy: 0.9949 - val_loss: 1.0635 - val_accuracy: 0.8391 Epoch 95/100

106/106 - 75s 705ms/step - loss: 0.0228 - accuracy: 0.9950 - val_loss: 1.1466 - val_accuracy: 0.8360 Epoch 96/100

106/106 - 75s 710ms/step - loss: 0.0087 - accuracy: 0.9980 - val_loss: 0.8861 - val_accuracy: 0.8537 Epoch 97/100

106/106 - 75s 704ms/step - loss: 0.0313 - accuracy: 0.9935 - val_loss: 1.5336 - val_accuracy: 0.7840 Epoch 98/100

106/106 - 75s 708ms/step - loss: 0.0247 - accuracy: 0.9943 - val_loss: 1.1566 - val_accuracy: 0.8277 Epoch 99/100

106/106 - 74s 702ms/step - loss: 0.0012 - accuracy: 0.9997 - val_loss: 0.3541 - val_accuracy: 0.9522 Epoch 100/100

106/106 - 74s 702ms/step - loss: 7.8204e-04 - accuracy: 0.9998 - val_loss: 0.4096 - val_accuracy: 0.9419