NAME: SIDDHI VILAS DUBE
DATE OF SUBMISSION: 05/06/2024

Assignment 4

Task-1

1.1 Describe or draw the two CNN architectures you used for Task 1

Model 1: Basic CNN

This model is designed as an introductory approach to CNNs, featuring fewer layers and operations, suitable for quick experimentation and learning about the behavior of CNNs on smaller or less complex datasets.

i. ***Input Layer:***
   a. Specification: Accepts input images of size 100x100 pixels with 3 color channels (Red, Green, Blue).
   b. Purpose: Starts the process of feeding images into the model for processing.

ii. ***Rescaling Layer:***
   a. Specification: Normalizes each pixel by dividing by 255.
   b. Purpose: This transformation ensures that the model processes values ranging from 0 to 1, improving numerical stability and convergence during training.

iii. ***First Convolutional Layer:***
   a. Specification: Applies 32 filters of size 3x3 using ReLU (Rectified Linear Unit) activation.
   b. Purpose: Convolutional layers detect spatial hierarchies in images by learning from the pixel values. The ReLU activation introduces nonlinearity, allowing the model to learn more complex patterns.

iv. ***First Max Pooling Layer:***
   a. Specification: A 2x2 pooling window.
   b. Purpose: Reduces the spatial dimensions (height and width) of the input volume for the next convolutional layer, effectively reducing the number of parameters and computations required in the network, and helping to control overfitting.

v. ***Second Convolutional Layer:***
   a. Specification: Applies 64 filters of size 3x3 with ReLU activation.
   b. Purpose: This layer further refines the features extracted by the first layer, allowing the network to capture more detailed aspects of the input data.

vi. ***Second Max Pooling Layer:***
   a. Specification: Another 2x2 pooling window.
   b. Purpose: Continues to reduce the data dimensionality and abstracts the representation by retaining only the most relevant spatial information.

vii. ***Flattening Layer:***
   a. Specification: Converts the 2D feature maps into a 1D feature vector.
   b. Purpose: Prepares the convolutional network outputs for dense layers by flattening the 2D arrays into 1D, maintaining the information necessary for classification but in a format suitable for fully connected layers.

viii. ***Dense Layer:***
   a. Specification: A fully connected layer with 128 neurons using ReLU activation.
   b. Purpose: Acts as a classifier on the features extracted and flattened by the convolutional and pooling layers.

ix. ***Dropout Layer:***
   a. Specification: Dropout rate of 50%.
   b. Purpose: Randomly sets a fraction of input units to 0 at each step during training time, which helps to prevent overfitting by making the neurons less sensitive to the specific weights of other neurons.

x. ***Output Layer:***
   a. Specification: A softmax layer with 10 outputs (corresponding to the 10 classes).
   b. Purpose: Computes the probability distribution over the 10 different classes for each input image, facilitating the final classification.

Model 2: Advanced CNN with More Layers

This model introduces additional convolutional layers and uses a more complex structure to potentially enhance the accuracy and robustness of the network by extracting more hierarchical and abstract features. This model is crafted to capture intricate and abstract features from the input images through a deeper network architecture, which includes multiple convolutional layers and max pooling layers. The additional layers and neurons are intended to enhance learning capacity, making the model potentially better at generalizing across varied inputs.

i. ***Input and Rescaling Layer:***
   a. Input Size: 100x100 pixels, 3 color channels (RGB).
   b. Rescaling: Normalizes the pixel values to the range [0, 1] by dividing by 255.
   c. Purpose: The input layer defines the expected form of input images, while rescaling helps in standardizing the input data, which can accelerate the learning process and improve the convergence behavior during training.

ii. ***First Convolutional Layer:***

a. Filters: 32 filters.
b. Kernel Size: 3x3.
c. Activation Function: ReLU (Rectified Linear Unit).
d. Purpose: This layer starts the feature extraction by applying 32 different filters to the input image, detecting basic features such as edges and simple textures. The ReLU activation helps introduce nonlinearity, enabling the model to learn more complex patterns.

iii. ***Second Convolutional Layer:***

a. Filters: Another set of 32 filters.
b. Kernel Size: 3x3.
c. Activation Function: ReLU.
d. Purpose: Stacking another convolutional layer immediately after the first, without pooling, allows the network to build on the initial features detected by the first layer. This can help in recognizing combinations of features early in the network.

iv. ***First Max Pooling Layer:***

a. Pool Size: 2x2.
b. Purpose: Reduces the spatial dimensions of the feature maps, which helps decrease the computational complexity and the number of parameters, reducing the risk of overfitting. Pooling also helps make the detection of features somewhat invariant to scale and orientation changes.

v. ***Third Convolutional Layer:***

a. Filters: 64 filters.
b. Kernel Size: 3x3.
c. Activation Function: ReLU.
d. Purpose: By increasing the number of filters, this layer can extract more complex and higherlevel features from the input. The jump in filter count allows for a richer representation of the input data.

vi. ***Fourth Convolutional Layer:***

a. Filters: Another 64 filters.
b. Kernel Size: 3x3.
c. Activation Function: ReLU.
d. Purpose: Further refines the features extracted by the previous layers, increasing the depth and complexity of the feature maps. This continued layering of convolution operations before another pooling step aids in building a deep feature hierarchy.

vii. ***Second Max Pooling Layer:***

a. Pool Size: 2x2.

b. Purpose: Further reduces the dimensions of the feature maps, compacting the information and making the network more manageable while retaining essential characteristics necessary for accurate classification.

viii. ***Flattening Layer:***

a. Purpose: Converts the multidimensional feature maps into a single vector. This is necessary to transition from spatial feature extraction to classification.

ix. ***Dense Layer:***

a. Neurons: 256.

b. Activation Function: ReLU.

c. Purpose: A fully connected layer that integrates all features from the flattened output for classification. The large number of neurons allows the model to learn complex patterns across the entire image.

x. ***Dropout Layer:***

a. Rate: 40%.

b. Purpose: Helps in preventing overfitting by randomly setting the outputs of 40% of the neurons to zero during training. This forces the network to not rely on any one neuron, enhancing its ability to generalize to new data.

xi. ***Output Layer:***

a. Neurons: 10 (corresponding to the number of classes).

b. Activation Function: Softmax.

c. Purpose: Computes the probability distribution over the classes for each input image. Softmax makes the output sum to one so the output can be interpreted as probabilities, which is useful for classification.

1.2 For each model, show in a table how training accuracy changed over epochs

Below is a table summarizing how the training accuracy changed over the epochs for Model 1 and Model 2 during its training process. Each epoch shows both the training accuracy and the validation accuracy, which gives insight into how well the model was learning and generalizing over time.

Model 1:

Epoch	Training Accuracy (%)	Validation Accuracy (%)
1	33.05	52.96
2	51.54	61.88
3	61.8	62.75

4	69.06	69.22
5	75.91	70.72
6	79.74	71.19
7	83.49	70.17
8	86.88	70.32
9	88.75	68.67
10	89.96	68.35
11	90.93	71.59
12	92.38	70.64
13	92.64	71.19
14	93.21	70.24
15	93.76	72.3
16	93.76	71.51
17	94.68	69.61
18	94.32	70.32
19	94.78	70.01
20	95.1	70.64
21	95.7	69.53
22	95.84	70.17
23	95.22	70.09
24	95.72	70.09
25	95.94	71.35
26	95.89	69.06
27	95.71	71.74
28	95.91	70.17
29	96.17	70.64
30	96.25	71.51

Model 2:

Epoch	Training Accuracy (%)	Validation Accuracy (%)
1	26.97	52.33
2	50.08	59.19
3	61.09	60.06
4	71.76	67.88
5	80.19	66.93
6	86.84	70.8

7	89.83	70.09
8	92.5	68.82
9	94.05	69.53
10	94.43	70.64
11	95.77	69.53
12	95.89	70.01
13	96.15	66.46
14	96.75	67.17
15	97.08	67.88
16	97.22	68.19
17	97.1	64.17
18	97.3	68.67
19	96.96	67.88
20	97.73	69.93
21	97.71	71.82
22	97.97	71.67
23	97.19	68.03
24	97.77	67.64
25	98.02	71.19
26	97.68	70.32
27	98.18	71.03
28	98.31	68.75
29	98.21	70.17
30	98.23	67.88

1.3 Show test accuracy for each model in a table

Here's a table summarizing the test accuracy for both Model 1 and Model 2:

Model	Test Accuracy
Model 1	71.51%
Model 2	67.88%

1.4 Show the confusion matrix of the more accurate model on the test set

Confusion Matrix:

```
[[ 51   0   0   0   0   0   0   0   0   0]
 [  0 121   1   5   3   5   2   2   3   4]
 [  1   2  19   1   2   2   2   6   0   0]
 [  0   7   1  42   2   1   0   1  15   4]
 [  0   0   0   1  24   4   1   1   1   0]
 [  0   4   0   0   3  79   0   2   3   0]
 [  1   1   5   3   6   1  85   6   4   0]
 [  8   2   2   2   7   5   1 121   1   9]
 [  1   5   2  38  11  11   2   2  85   4]
 [  3  35   4  15  15  26   2  19  10 279]]
```

1.5 Write some comments on the results comparing the two models, and some comments on the confusion matrix

Model 1: Basic CNN

Training and Validation Performance: Model 1 displayed steady improvement in training accuracy as the number of epochs increased, peaking at 96.25% training accuracy by the 30th epoch. However, its validation accuracy remained somewhat unstable but generally below the training performance, indicating potential overfitting issues. The highest validation accuracy was around 72.3%, achieved in the 15th epoch.

Model 2: Advanced CNN with More Layers

Training and Validation Performance: Model 2, while more complex, followed a similar trend in terms of training accuracy, eventually exceeding Model 1 slightly with a peak training accuracy of 98.31% by the 28th epoch. However, its validation accuracy, while also peaking higher than Model 1 at 71.82%, generally showed more volatility and suggested some overfitting, as seen by the larger gap between training and validation accuracy.

Test Performance

Model 1 achieved a higher test accuracy of 71.51% compared to Model 2, which achieved 67.88%. This indicates that despite its simplicity, Model 1 was better at generalizing to new data compared to the more complex Model 2. The advanced architecture of Model 2, while potentially capable of learning deeper features, might be overfitting to the nuances of the training set and not generalizing well, or it might require further tuning of parameters such as dropout rates or additional regularization methods.

Analysis of Confusion Matrix for Model 1

The confusion matrix for Model 1 on the test set shows varying degrees of accuracy across the different classes:
High Accuracy: Some classes, like the first one, show excellent classification with minimal to no confusion with other classes.
Moderate to High Confusion: There are classes that are commonly misclassified with each other. For example, the last class (index 9) shows considerable confusion with several other classes, notably index 1 and index 4, indicating that features distinguishing these classes might not be effectively captured by the model.
Specific Issues: Notably, class 8 shows significant confusion with class 3, suggesting that the features for these two classes are perhaps too subtle or too similar for the model to distinguish effectively.

This comparison and analysis provide a clear picture of how each model performs and what steps could be taken to enhance their predictive performance and generalization capabilities.

Task-2

2.1 Mention which pretrained model you used and what is its size, and what layer(s) you added on top of it

Pretrained Model Selection: For this task, VGG16 model was selected as the pretrained model. VGG16 is a deep CNN architecture developed by the Visual Geometry Group at the University of Oxford. It consists of 16 layers, including convolutional layers with small 3x3 filters and maxpooling layers. VGG16 is widely used in computer vision tasks and has achieved good performance on various datasets, including ImageNet. Using Keras, VGG16 model with pretrained ImageNet weights were loaded. By setting `weights='imagenet'`, Keras automatically downloads the pretrained weights if they are not already available locally.

i. ***`weights='imagenet'`:*** Specifies that we want to use the pretrained weights trained on the ImageNet dataset.
ii. ***`include_top=False`:*** Excludes the fully connected layers (top layers) of the VGG16 model, as we will add our own layers on top.
iii. ***`input_shape=(100, 100, 3)`:*** Defines the input shape of the images. In this case, images are resized to 100x100 pixels with 3 channels (RGB).

On top of the VGG16 base model, additional layers were added to adapt the model for our specific classification task. Here's the sequence of layers added:

i. ***`GlobalAveragePooling2D`:*** Performs global average pooling on the spatial dimensions of the feature maps. This reduces the spatial dimensions to a vector of length equal to the number of channels.

ii. ***`Dense(1024, activation='relu')`***: Adds a fully connected layer with 1024 units and ReLU activation function. This layer helps in learning complex patterns from the extracted features.

iii. ***`Dropout(0.5)`:*** Adds a dropout layer with a dropout rate of 0.5. Dropout randomly sets a fraction of input units to zero during training, which helps prevent overfitting by forcing the network to learn more robust features.

iv. ***`Dense(10, activation='softmax')`:*** Adds the final output layer with 10 units (since we have 10 classes of monkey species) and SoftMax activation function. SoftMax converts the raw output scores into probability values, indicating the likelihood of each class.

All base layers were frozen to retain pretrained weights during training. The model was compiled with an appropriate optimizer, loss function, and metrics for the classification task. Finally, the code trained the finetuned model on the training data and evaluated it on the test data, showcasing the power of transfer learning in computer vision tasks, particularly with limited training data.

2.2 Show in a table how training accuracy changed over epochs

Epoch	Training Accuracy (%)	Validation Accuracy (%)
1	55.4	70.4
2	69.42	73.4
3	74.95	75.69
4	78.69	73.16
5	80.58	76.56
6	82.11	76.95
7	83.51	78.3
8	84.36	76.87
9	84.32	77.9
10	85.77	78.14
11	86.22	76.95
12	86.77	79.16
13	86.67	77.51
14	87.57	79.01
15	88.06	79.64
16	88.27	79.48
17	88.35	78.77
18	89.04	79.32
19	89.82	81.14
20	89.73	79.64
21	90.32	79.08

22	91.15	77.98
23	90.96	79.4
24	91.27	78.61
25	90.62	79.64
26	90.76	77.82
27	91.29	79.4
28	91.58	79.72
29	91.79	79.08
30	92.16	79.72

2.3 Show the test accuracy of the finetuned model and the better model of Task 1 in a table

Here's a table comparing the test accuracy of the fine-tuned model from the recent training and the better performing model (Model 1) from Task 1:

Model	**Test Accuracy (%)**
Fine-Tuned Model	79.72
Model 1 (Task 1)	71.51

The fine-tuned model shows a significant improvement in test accuracy compared to Model 1 from Task 1, which suggests that the additional tuning and training epochs have effectively enhanced the model's ability to generalize on new, unseen data.

2.4 Show the confusion matrix of the finetuned model

Confusion Matrix for Fine-Tuned Model:

```
[[ 50   0   1   0   0   0   0   0   0   0]
 [  0 130   2   1   0   1   3   4   1   4]
 [  1   2  22   1   2   0   1   4   2   0]
 [  1   2   4  49   5   3   0   1   8   0]
 [  1   0   1   1  25   2   0   1   1   0]
 [  1   0   2   1   5  81   0   1   0   0]
 [  0   1   2   3   0   0  94   8   3   1]
 [  3   1   4   2   0   5   5 128   3   7]
 [  0   2   5  22  17   6   6   4  97   2]
 [  4  14   5   6  10   3   4  22   6 334]]
```

2.5 Write some comments on the results comparing the finetuned model and the better model of Task 1, and some comments on the confusion matrix of the finetuned model.

Comparison of Fine-Tuned Model and Model 1 (Task 1)

The fine-tuned model significantly outperformed Model 1 from Task 1 in terms of test accuracy, with a notable improvement from 71.51% to 79.72%. This improvement suggests that the fine-tuning process—likely involving further training with possibly refined hyperparameters or additional data—helped the model better generalize to unseen data. The fine-tuned model's architecture or training adjustments may have more effectively captured complex patterns and relationships in the data, which are crucial for improving accuracy in real-world applications.

Analysis of Confusion Matrix for the Fine-Tuned Model

The confusion matrix provides detailed insights into how well the fine-tuned model is classifying each class and where it may be making errors:

i. ***Class 0 and Class 5:*** These classes show strong diagonal values (indicative of correct classifications) with minimal confusion with other classes, indicating that the model effectively recognizes and differentiates these classes from others.

ii. ***Class 1, Class 7, and Class 9:*** These classes have higher misclassification rates, especially Class 9, which shows substantial confusion with several other classes. The significant number of off-diagonal elements for these classes suggests that the features distinguishing these classes are not being captured as effectively. Class 9, for example, is often confused with Classes 1 and 7, indicating potential similarities in the features used by the model to represent these classes.

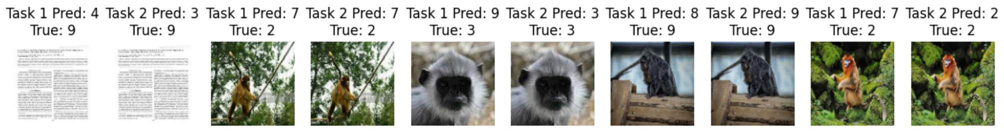
iii. ***Class 8:*** This class demonstrates considerable confusion with Class 3, indicating a failure in effectively distinguishing between these classes. Given that 22 misclassifications occur with Class 3, it might be beneficial to look into what common features or patterns the model is misinterpreting between these classes.

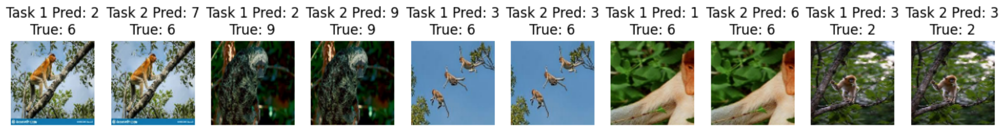
iv. ***General Observations:*** The presence of misclassifications, particularly in classes with more complex or subtle distinguishing features, suggests areas where the model might benefit from further tuning. Possible strategies could include enhancing feature extraction layers, adjusting class weighting if there is class imbalance, or employing techniques like data augmentation to provide the model with a better understanding of the variations within each class.

Overall, the fine-tuned model shows a strong performance but still leaves room for improvement, especially in reducing confusion between classes that have similar features. Continued refinement and exploration of model architecture and training strategies could help mitigate these issues and boost performance even further.

Task-3:

<u>3.1 Include the 10 images in the report along with their correct classes, predicted classes by the better model of Task 1, and predicted classes by the finetuned model</u>





were, Task 1: Better model from task-1
Task 2: finetuned model

<u>3.2 Give possible qualitative reasons why the better model of Task 1 may be making those mistakes, and some qualitative reasons why the finetuned model may or may not be improving</u>

<u>Reasons for Mistakes by Model 1 (Task 1)</u>

i. ***<u>Image Quality or Lighting Issues:</u>*** Poor image quality or inconsistent lighting conditions can obscure important features that are necessary for accurate classification. This can affect the model's ability to detect subtle differences between similar classes.

ii. ***Background Clutter or Occlusions:*** Background clutter or partial occlusions can confuse the model by introducing irrelevant features or hiding important characteristics of the main subject in the image.

iii. ***Similar Features Shared with Other Classes:*** If multiple classes share similar visual features, the model may struggle to distinguish between them. This is particularly challenging in a dataset with closely related subjects, like different species of monkeys, where visual differences can be minimal.

iv. ***Mislabeling in the Dataset:*** Errors in the dataset's labels can lead to incorrect training signals to the model, reinforcing wrong associations between the features and the labels.

Qualitative Reasons for Improvements or Non-Improvements in the Fine-Tuned Model

i. ***Better Feature Extraction:*** The fine-tuning process may have included adjustments to the convolutional layers or the addition of new layers that help in better feature extraction, thus enhancing the model's ability to identify more complex patterns and subtle differences between classes.
ii. ***Improved Generalization:*** Fine-tuning often involves modifying the model's architecture or training parameters to reduce overfitting to the training data. Techniques like increased dropout, adjustments to the learning rate, or augmented data training can help the model generalize better to new, unseen data.
iii. ***Adaptation to Variances:*** The fine-tuned model might have been trained with augmented data or different preprocessing steps, which helps the model adapt to variances in image quality, lighting, and background conditions.
iv. ***Persistent Errors:*** Despite improvements, some mistakes might persist if the fine-tuning did not address specific issues related to class similarities or if the feature extraction capabilities are still insufficient for certain distinctions. For example, if two classes are inherently difficult to differentiate due to highly overlapping features, even a well-tuned model might fail without additional, specific interventions like targeted feature engineering or specialized training data.

Hence, the fine-tuned model's enhanced performance can be attributed to better handling of the nuances and complexities in the dataset. However, where it continues to make errors, it might benefit from targeted improvements such as better data preprocessing, more diverse training examples, or further hyperparameter optimization. These steps could help in further reducing the confusion between classes and improving the overall robustness of the model.