Assignment 2: Convolution

Submitted by: **Nikitha Begari & Kavyasree Bommakanti**

Group Number: **4**

Overall Summary:

- Our project's goal is to create a new convolutional neural network from starting level that is intended only for computer vision applications. The "Dog-vs-Cats" dataset on Kaggle is the source of the dataset we are using. The task of developing an effective model is difficult because we have insufficient data accessible.
- Popular deep learning models known as convolutional neural networks, or convnets, have shown to be quite successful in computer vision applications. Convnets have a number of benefits, one of which is their capacity to recognize and identify spatial patterns in images. They are therefore excellent at tasks like segmentation, object detection, and image recognition.
- Even with the restricted amount of data available to us, we think our convnet model can still yield acceptable outcomes. This is because convolutional neural networks (convnets) can extract and identify key features from images, allowing them to learn and generalize from limited samples. Our model will be trained on the small dataset, improved through transfer learning methods, and its performance will be assessed with suitable assessment measures. Our ultimate objective is to create a convolutional neural network that, given a small amount of data, can accurately and efficiently categorize photos from the "Dog-vs-Cats" dataset.

Steps:

1.Creation of subset:

- A dataset comprising images of dogs and cats can be divided into subgroups using the make_subset function.
- Three subset creation sets are made, each with a unique start and end index:
- There are 1000 images in the first batch for training, 500 for validation, and 500 for testing.
- There are 1500 images in the second batch for training, 500 for validation, and 500 for testing.
- There are 1500 images in the third set for training, 1000 for validation, and 500 for testing.

2. Convolutional Neural Network (CNN) with Data Augmentation:

- With TensorFlow, a CNN model is defined via the Keras API.

- The data augmentation model is used to apply data augmentation techniques to the input images, such as rotation, zoom, and random horizontal flipping.
- The convolutional layers of the model are followed by layers that perform max pooling, a layer that flattens, dropout regularization, and a dense output layer that is activated by sigmoid.

Accuracy is the metric, RMSprop is the optimizer, and binary cross-entropy serves as the loss function while compiling the model.

3.Training Model and Evaluation:

- The model is trained using the training dataset with data augmentation and validated using the validation dataset.
- During training, the best model is saved based on validation loss using the ModelCheckpoint callback.
- The stored model is loaded after training, and its results are assessed using the test dataset.
- To evaluate the model's performance on unknown data, its test accuracy is printed.

Given Problem Aim and objective:

The objective of the Cats-vs-Dogs dataset binary classification task is to identify if an image is part of the dog or cat class.

Methods

Dataset:

The Cats-vs-Dogs dataset is 543 MB in size (compressed) and includes 25,000 images of dogs and cats (12,500 from each class). We will generate a new dataset after downloading and unzipping it, which will have three subsets: a training set with 1000 samples of every class, a validation set with 500 samples of every class, and a test set with 500 samples of every class. We must enlarge our neural network in order to handle the larger image size and more complicated task we are working on. We will build an additional stage into our current Conv2D + MaxPooling2D architecture in order to achieve this.

In addition to increasing network capacity, this will also shrink the feature maps, ensuring that they are not too big when we get to the Flatten layer. The feature maps in our input photos start out as 150x150 pixels, and as we move through the network's layers, they progressively get smaller until they are 7x7 just before the Flatten layer. Although rather arbitrary, this input size selection is suitable for the task at hand.

Preprocessing:

Check the image files:

- Go through the image file directory in a loop.
- Verify that the files are JPEG images by looking at their extensions.
- Visualize or examine a few sample images if you'd like to better comprehend the dataset.

Transform the JPEG data into grids of RGB pixels:

- Utilizing a suitable library such as PIL or OpenCV, read each JPEG image.
- Create RGB pixel grids from the image data; these are effectively three-dimensional arrays with dimensions (height, width, and channels).

Make tensors with floating points:

- Transform the RGB pixel grids into tensors with floating points.
- Tensors are multi-dimensional arrays that, in this instance, represent images in a way that neural networks can understand.

Change the pixel values' scale:

- To rescale the pixel values into the interval [0, 1], divide them by 255.
- Because small ranges of input values are often better for neural networks, rescaling aids in normalizing and enhances convergence during training.

Data Augmentation:

A vital technique in machine learning for artificially increasing a dataset's diversity and size is data augmentation. Data augmentation creates additional instances of data by randomly transforming the preexisting training samples. This improves the dataset and improves the performance of the model.

Pre-trained Model Utilization:

Pretrained networks are beneficial when the original dataset is extensive and varied, as they offer learned features that can be applied to various computer vision tasks.

The capability to transfer learned features across different tasks is a key strength of deep learning compared to other machine learning techniques.

Example: ImageNet and VGG16:

ImageNet is a large dataset containing 1.4 million labeled images across 1,000 different classes, including various animal classes such as cats and dogs.

VGG16 is a popular convolutional neural network architecture trained on the ImageNet dataset. It is simple yet effective for many computer vision tasks.

Utilization Methods:

There are two primary methods for utilizing a pretrained network: feature extraction and fine-tuning.

In feature extraction, we leverage the learned features of the pretrained network and add new layers on top for the specific task at hand.

Fine-tuning involves adjusting the pretrained model's weights during training to better suit the target task.

Feature Extraction:

In this approach, we use the pretrained network's convolutional base as a feature extractor, freezing its weights to prevent them from being updated during training.

We then add new layers on top of the pretrained model to adapt it to the target task, such as classification of cats and dogs.

Data Augmentation with Feature Extraction:

- By combining data augmentation with feature extraction, we aim to further enhance the model's performance.
- Data augmentation introduces variations to the training data, increasing its diversity and improving the model's ability to generalize to unseen images.
- This combined approach leverages the power of pretrained features while benefiting from the increased robustness provided by data augmentation.

In this instance, we want to use data augmentation approaches to increase the model's accuracy. In particular, we intend to randomly apply several modifications to the training data images, including flipping, rotating, and zooming. By adding variants to the images, these changes guarantee that during training, the model never sees the same image twice. As a result, the model gains robustness and improves its ability to generalize to new data.

In this way, we may successfully improve the diversity of the training dataset, which will help the model learn more resilient and invariant characteristics. Even when trained on a small dataset, this method reduces overfitting and improves the model's ability to categorize images reliably. In the end, data augmentation helps improve the model's overall effectiveness and consistency in everyday applications.

Results:

- After incorporating data augmentation techniques, the model's performance has improved significantly across all datasets. The gap between the training, validation, and test accuracies has also reduced, indicating better generalization ability.
- Increase your training sample size by Keeping the validation and test samples.
- Increasing the training sample size and applying data augmentation have significantly improved the model's performance, reducing the gap between the training, validation, and test accuracies.

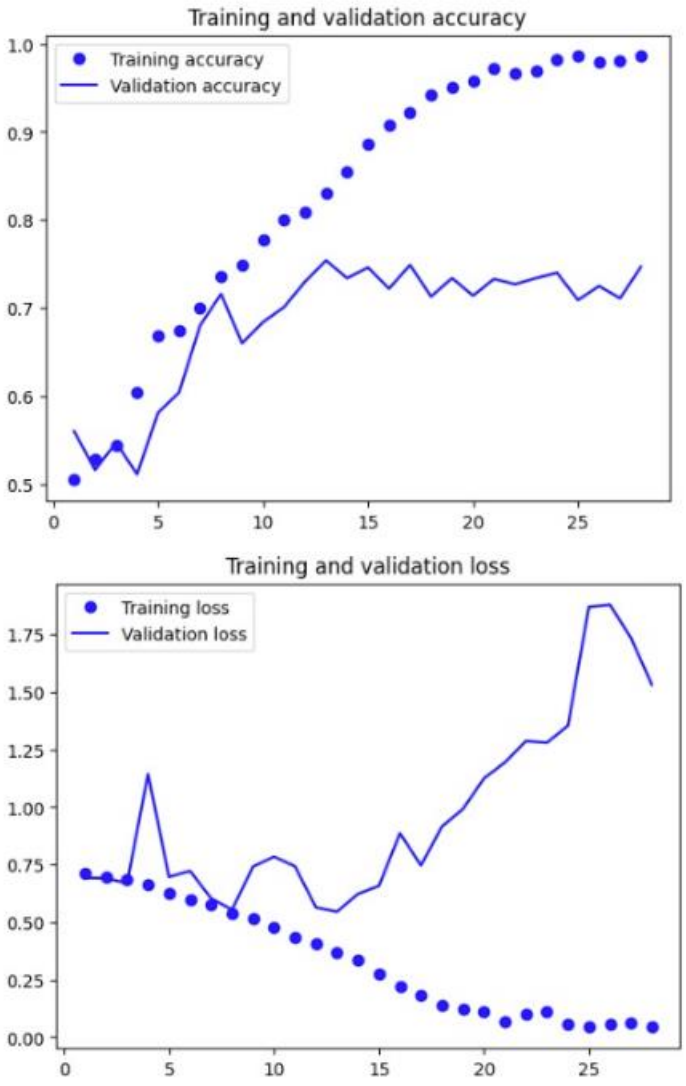
For each approach, the accuracy and validation accuracy are displayed in the table below.

Model Table Results:

Training Size	Test Size	Validation Size	Data Augmentation	Accuracy (Test) %	Accuracy (Validation) %	Accuracy %
1000	500	500	No	72.6	74.7	98.6
1000	500	500	Yes	77.1	80.3	82.9
1500	500	500	No	74.3	75.3	98.4
1500	500	500	Yes	84.7	83.4	86.3
1500	500	1000	No	73.6	75.7	98.8
1500	500	1000	Yes	75.0	77.9	79.0

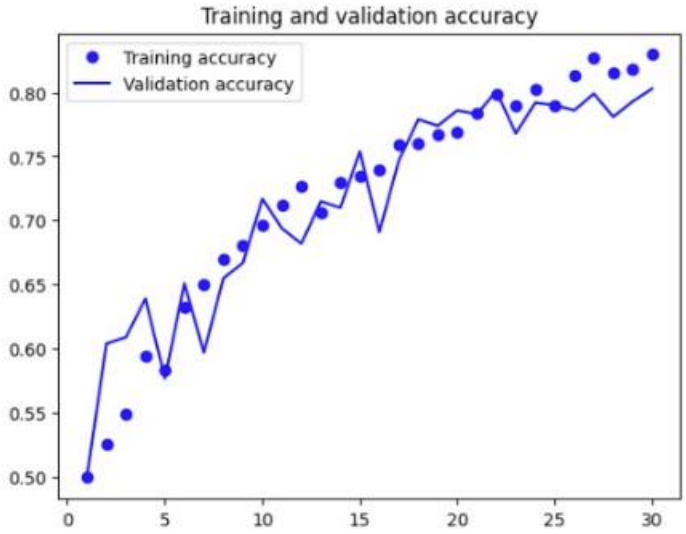
Training size:1000, Test Size:500, Validation Size:500-without Augmentation

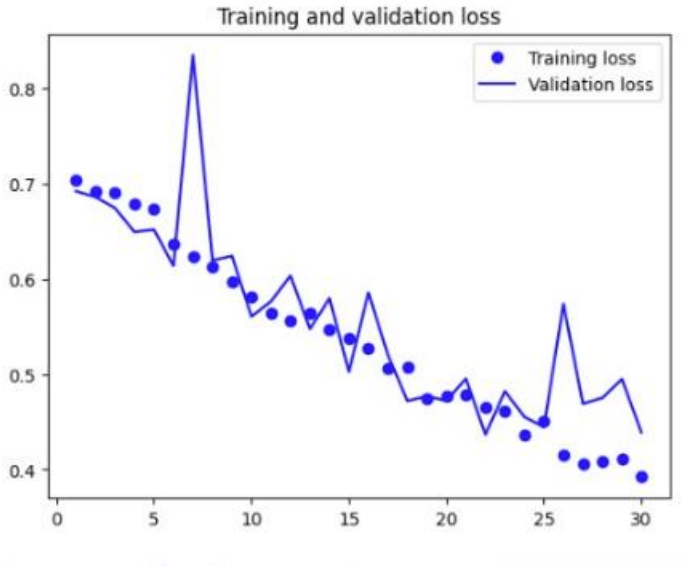
[25]


Training and validation accuracy
Training accuracy
Validation accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0
5
10
15
20
25
Training and validation loss
Training loss
Validation loss
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
5
10
15
20
25

Training size:1000, Test Size:500, Validation Size:500-with Augmentation

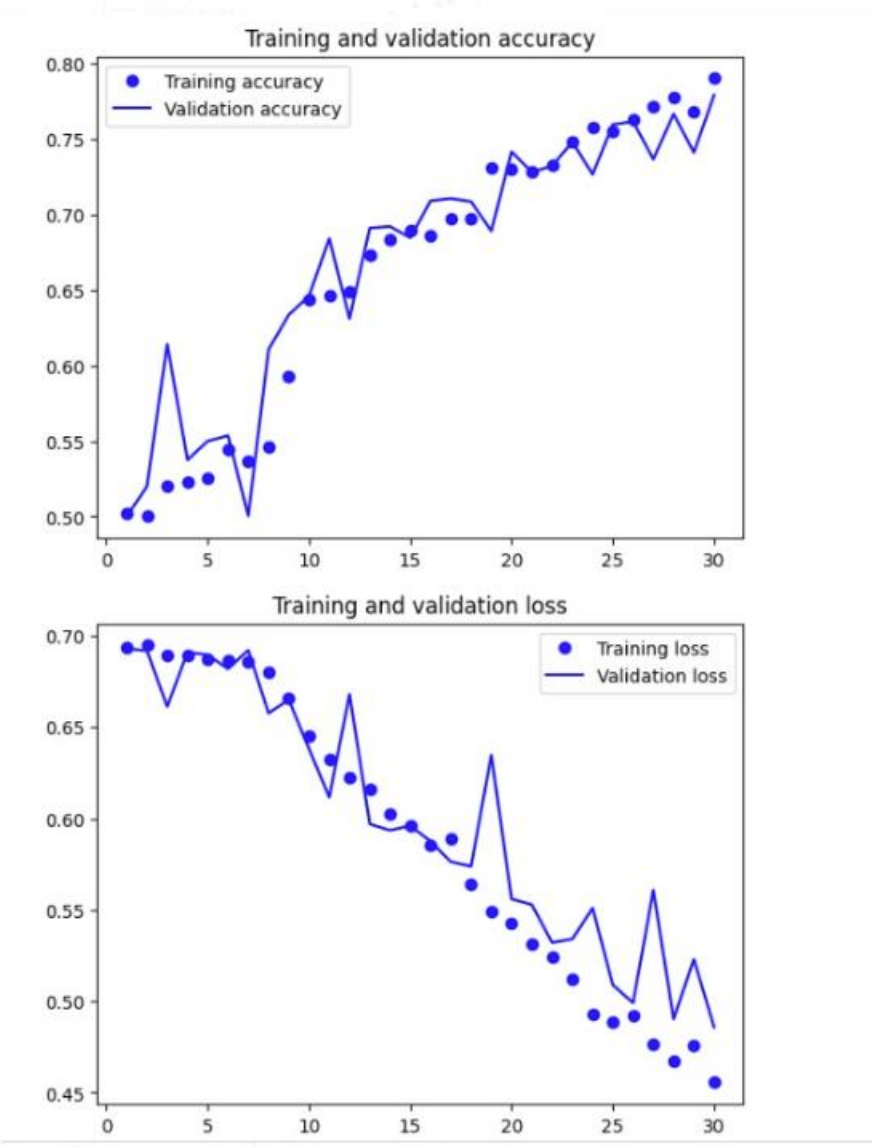
[32]



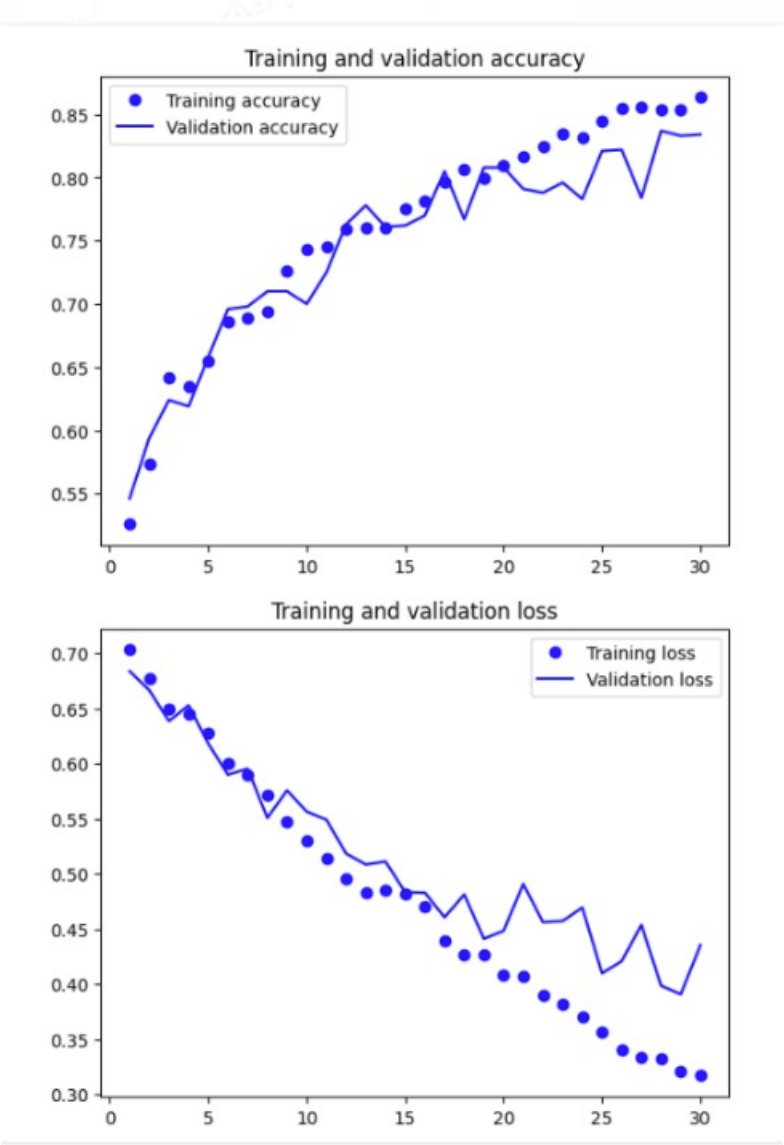


Increase your training sample size

Training size:1500, Test Size:500, Validation Size:500-without Augmentation

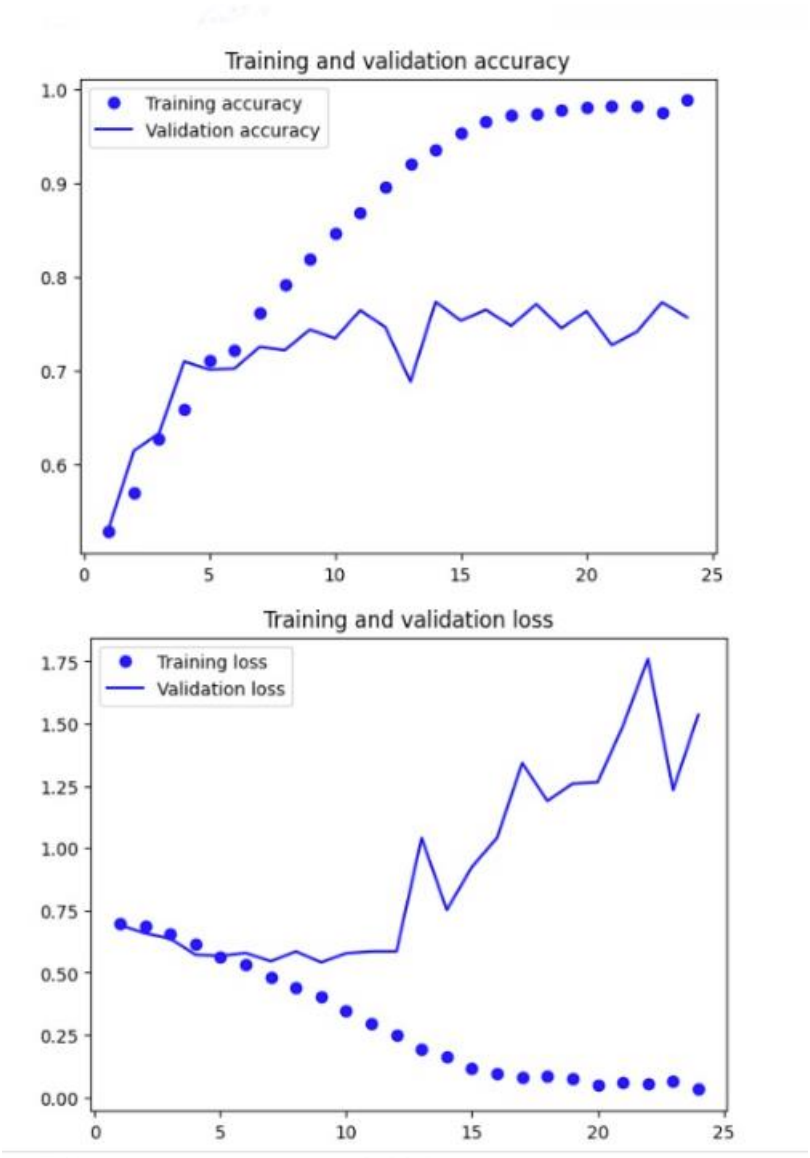


Training size:1500, Test Size:500, Validation Size:500-with Augmentation

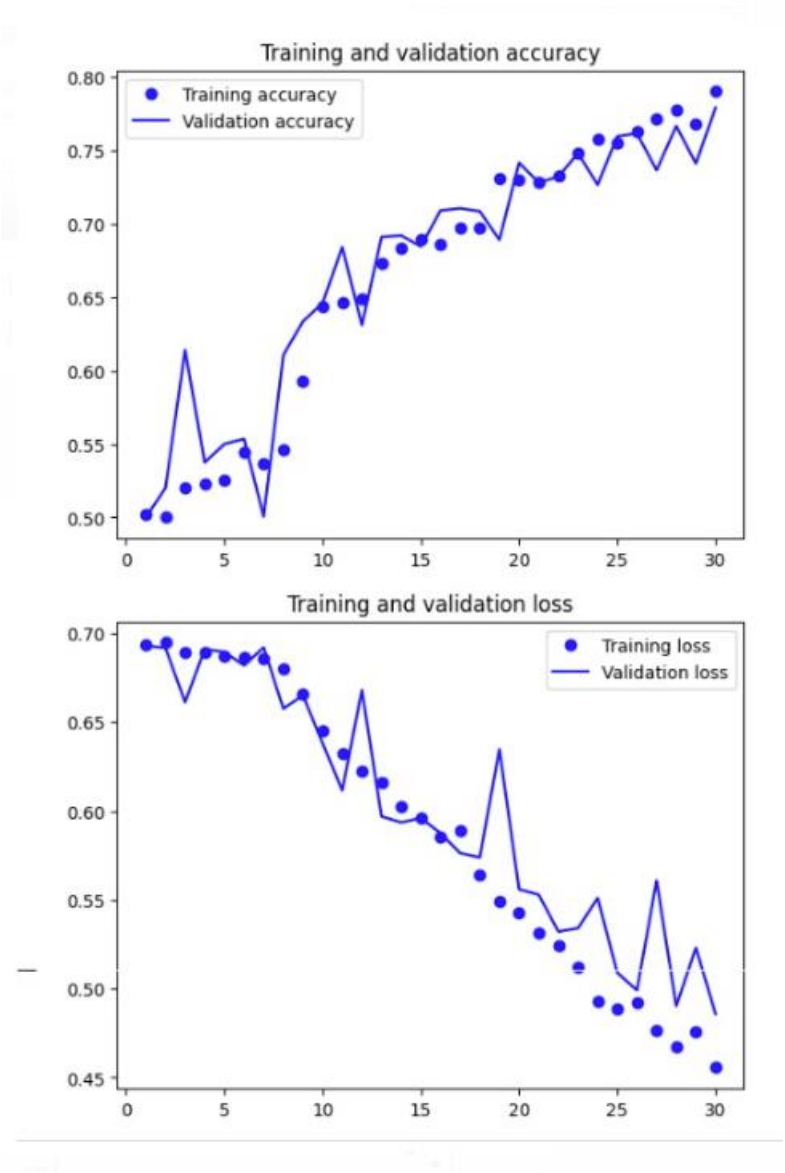


Finding the ideal training sample size -We set the training, validation, and test set sizes, respectively, to 1500, 1000, and 500.

Without Augmentation



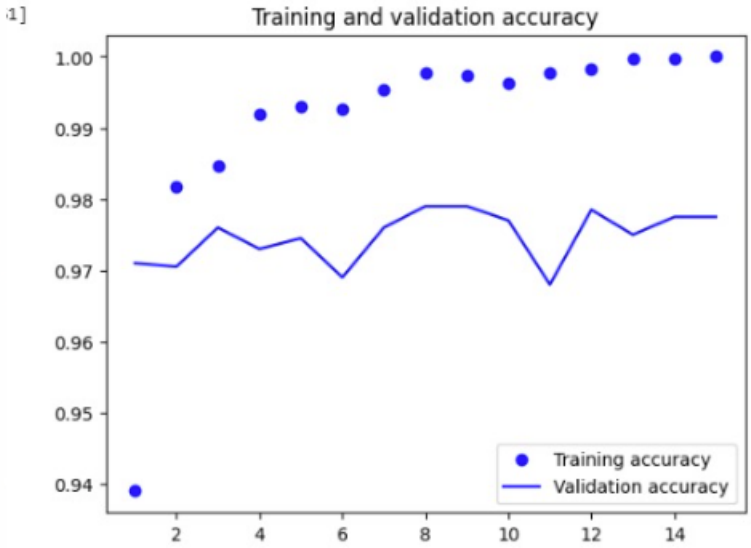
With Augmentation



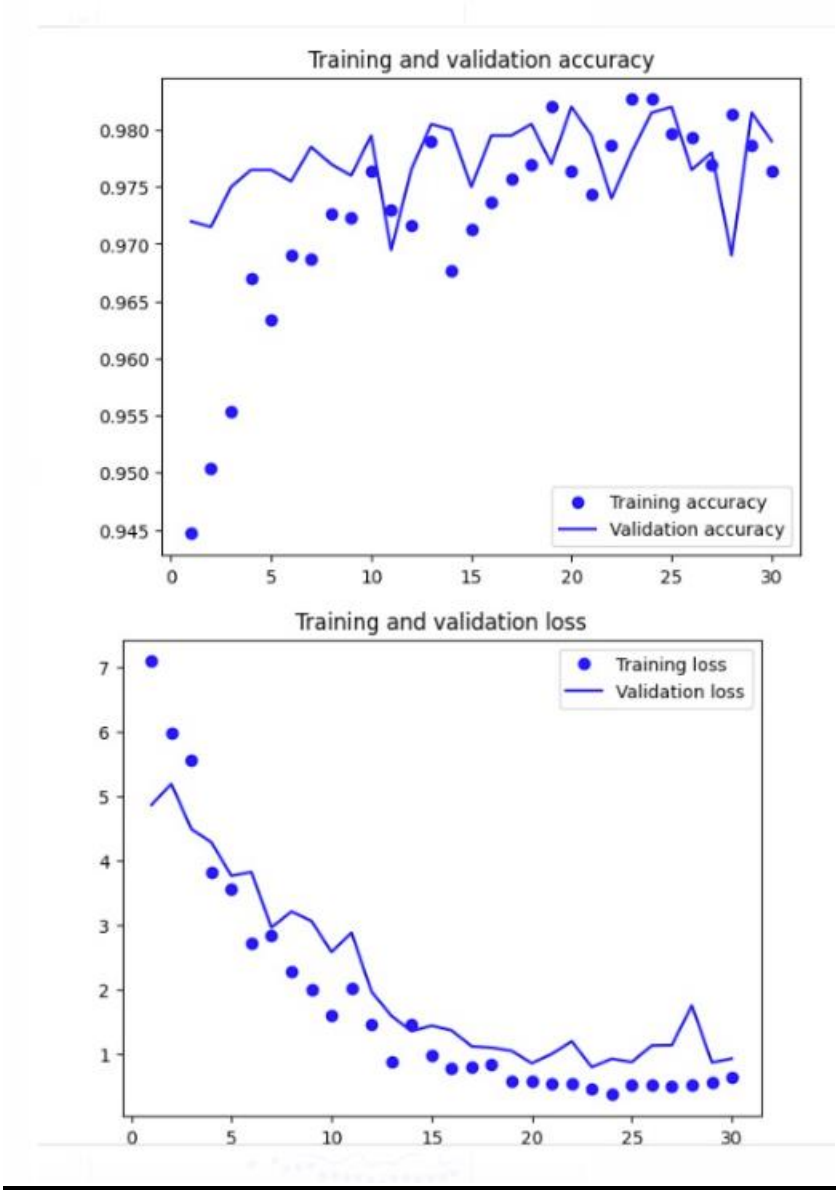
Pre-Trained Model Results:

Data Augmentation	Accuracy %	Accuracy (Validation) %
Yes	97.6	97.9
No	100	97.7

Pretrained Model-without Augmentation:



Pretrained Model-with Augmentation:



From above Table:

List of Model Configurations and Sample Sizes:

- Model configurations are provided for both models trained from scratch and pretrained models.
- Sample sizes for the train, test, and validation sets are listed in the tables for each model configuration.

Model Training Results:

- For models trained from scratch, results are presented with and without data augmentation, as well as with variations in the size of the training and validation sets.
- For pretrained models, accuracy and validation accuracy are compared with and without data augmentation.

Observations:

- Models trained with data augmentation did not consistently outperform those trained without it.
- Increasing the size of the training set or adjusting the size of the validation set generally improved the model's accuracy.
- Pretrained models tended to achieve higher accuracy compared to models trained from scratch, especially when dealing with limited training data.

Impact of Data Augmentation:

- Data augmentation did not lead to a significant improvement in the accuracy or validation accuracy of the pretrained models.
- This suggests that the pretrained models already capture a wide range of features and may not benefit as much from additional data augmentation.

Overall Comparison:

- Pretrained models demonstrate superior performance, particularly when trained on limited datasets.
- While data augmentation can be beneficial in some cases, its effectiveness may vary depending on the dataset and model architecture.
- Experimentation with different model configurations and training techniques is essential to identify the most effective approach for a given task.