Advanced Machine Learning

Assignment-2

Convolution Report

Objective:

The primary objective is to train and construct a convolutional neural network system that, even with a little dataset, can accurately classify "Dog-vs-Cats" images.

This project's goal is to design, develop, and apply a particular type of convolutional neural network that will be used in computer vision applications. The dataset that I have utilized is essentially a portion of "The Dog-vs-Cats" dataset from Kaggle. A primary obstacle that may hinder a predictive model's ability to be used as intended is insufficient data.

Convolutional Neural Network:

Currently, one of the most popular deep-learning models that has shown especially high efficiency in machine perception problems is convolutional neural networks, or CNNs for short. Their greatest advantage is their ability to recognize and memorize spatial patterns in the visuals. For tasks like object detection, segmentation, and picture recognition, this is the reason they are frequently selected. Despite the persistent lack of data, the authors are confident that the model will yield reliable and good results. First, the basis for this particularity is that the learning model may learn from the smaller datasets by recognizing and understanding pertinent image aspects. Training of the model using small datasets and transfer learning using the built-in training function.

Problem to be Executed:

Predicting if a picture belongs in the dog or cat class is the aim of the binary classification task for the Cats-vs-Dogs dataset.

Dataset:

The Kaggle Cats vs Dogs dataset is a popular dataset used for image classification tasks. It contains images of cats and dogs, which are used to train and evaluate machine learning

models, particularly convolutional neural networks (CNNs). Here is some detailed information about the dataset:

Dataset Overview:

Name: Dogs vs. Cats

Source: Kaggle

Content: The dataset contains images of cats and dogs in separate categories.

Number of Images: Approximately 25,000 labeled images.

Classes: 2 (cats and dogs)

Format: JPEG images

• A training group of one thousand images of each breed (cats and dogs).

• A group including 500 images of each category, used to assess the model's performance.

• A group containing 500 images of each kind for the final testing.

I chose to enlarge the neural network because the pictures are large, and the challenge is complex. I expanded the idea by adding a layer that combines MaxPooling2D and Conv2D. This reduces the size of the image fragments as they move through the layers and boosts the network's power. They are precisely the appropriate size, measuring around 7 by 7, when they get to the Flatten layer. The initial images are 150 × 150 pixels in size, which may appear arbitrary.

Preprocessing:

Image File Processing:

- Look at the picture files.
- Turn the JPEG images into grids of colored dots.
- Change these grids into numbers with decimal points.
- Ensure that the numbers fall within a range that computers can comprehend, such as from dark to extremely light. To achieve this, we ensure that the numbers move from 0 to 1 rather than 0 to 255. This improves how well the computer processes the images.

Data Augmentation:

Data augmentation is a technique used to artificially expand the size of a training dataset by creating modified versions of the original data. In the context of convolutional neural networks (CNNs), which are widely used for image recognition and classification tasks, data augmentation helps improve the performance and robustness of the model by preventing overfitting and making the model generalize better to new, unseen data.

By applying these transformations, the model is exposed to a wider variety of images during training, which helps it learn more robust features and improves its ability to generalize to new data. This is particularly useful when the available training dataset is small or lacks diversity.

To accomplish this, we apply various edits to the existing images, such as cropping or flipping them. In this approach, during the learning process, the model is exposed to several variations of the same images.

I am going to change the images in our training set at random to add some variation to this assignment. I will flip some of these, rotate others, and enlarge them in a few. This will improve the model's ability to identify objects in pictures by providing a wider range of images for it to learn from.

Pre-trained Network:

The code demonstrates the process of using a pre-trained convolutional neural network (VGG16) for feature extraction from images. This involves loading a pre-trained model, preprocessing the images, extracting features using the convolutional base of VGG16, and preparing these features for training a custom classifier.

- The VGG16 model is loaded with pre-trained weights from the ImageNet dataset.
- Images are preprocessed using VGG16's preprocess_input function.
- The preprocessed images are passed through the VGG16 model to extract features.

The features and labels are collected and concatenated into numpy arrays.

The VGG16 model summary shows the architecture of the convolutional base, including the input layer and various convolutional and pooling layers.

The feature extraction process is executed, and the feature shapes are confirmed, showing that the model successfully processed the dataset and extracted features.

The extracted features from the training dataset have a shape of (3000, 5, 5, 512), indicating that the convolutional layers of VGG16 reduced the spatial dimensions of the images to 5x5 with 512 feature maps per sample.

Key Points:

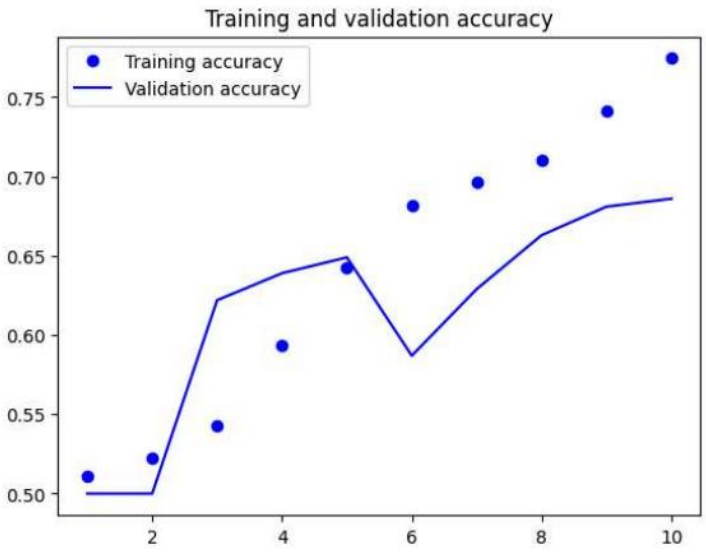
Feature Extraction: Using the convolutional base of a pre-trained VGG16 model to extract high-level features from images.

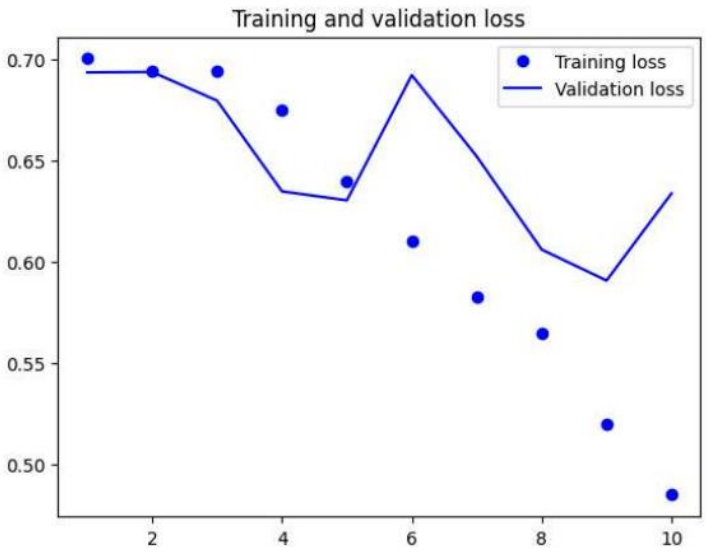
Preprocessing: Images are preprocessed to match the input requirements of the VGG16 model.

Datasets: Features and labels are extracted from training, validation, and test datasets.

Shape Confirmation: The shape of the extracted features confirms the successful execution of the feature extraction process.

```
import matplotlib.pyplot as plt
accuracy = history.history["accuracy"]
val_accuracy = history.history["val_accuracy"]
loss = history.history["loss"]
val_loss = history.history["val_loss"]
epochs = range(1, len(accuracy) + 1)
plt.plot(epochs, accuracy, "bo", label="Training accuracy")
plt.plot(epochs, val_accuracy, "b", label="Validation accuracy")
plt.title("Training and validation accuracy")
plt.legend()
plt.figure()
plt.plot(epochs, loss, "bo", label="Training loss")
plt.plot(epochs, val_loss, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.legend()
plt.show()
```



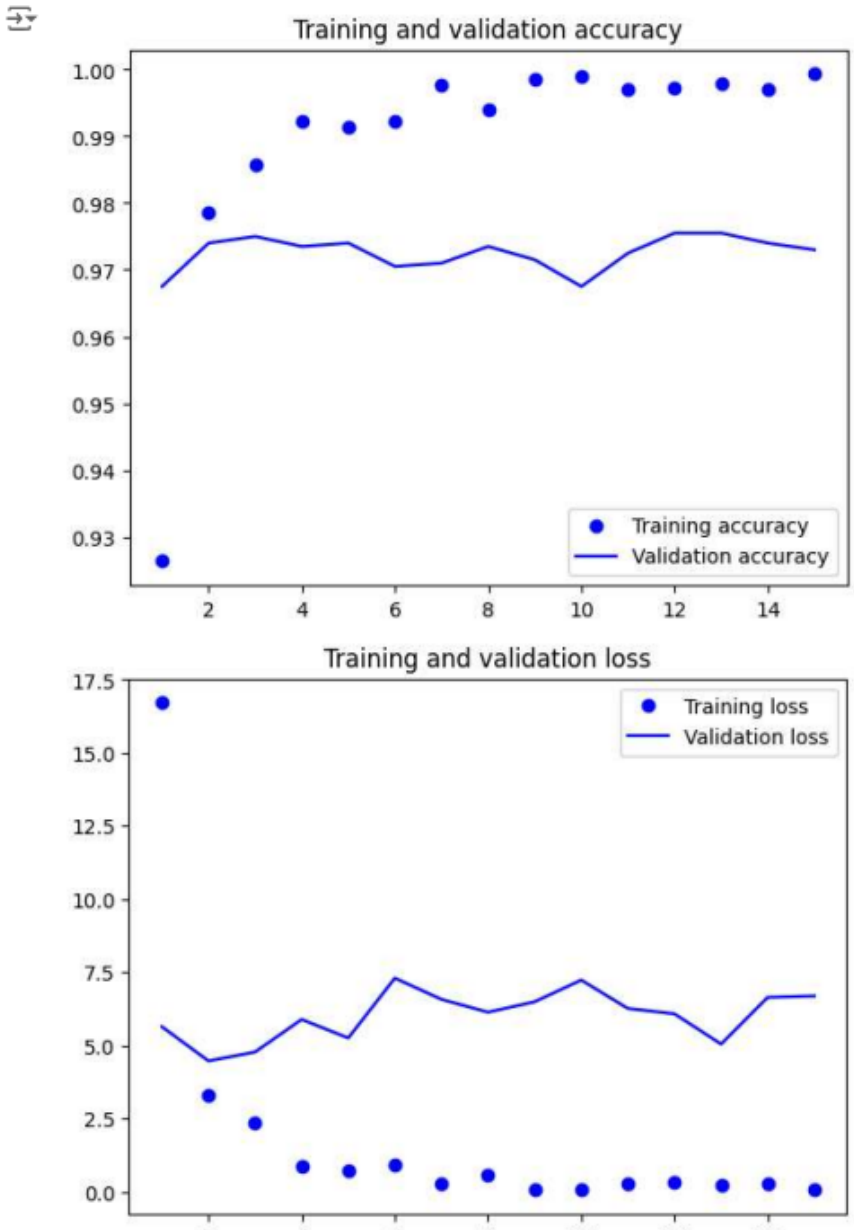


The plot indicates the training and validation accuracy and loss over the epochs. It shows that the model is learning well, as the training accuracy increases, and the training loss decreases over time. The validation accuracy and loss also show similar trends, indicating that the model is generalizing well to unseen data.

Training and validation accuracy: The training accuracy is higher than the validation accuracy, which is expected as the model is trained on the training data. However, the validation accuracy is also increasing, indicating that the model is not overfitting.

Training and validation loss: The training loss is decreasing over time, indicating that the model is learning. The validation loss also shows a similar trend but is slightly higher than the training loss. This is expected, as the model is trained on the training data and the validation data is unseen by the model during training.

```
import matplotlib.pyplot as plt
acc = history.history["accuracy"]
val_acc = history.history["val_accuracy"]
loss = history.history["loss"]
val_loss = history.history["val_loss"]
epochs = range(1, len(acc) + 1)
plt.plot(epochs, acc, "bo", label="Training accuracy")
plt.plot(epochs, val_acc, "b", label="Validation accuracy")
plt.title("Training and validation accuracy")
plt.legend()
plt.figure()
plt.plot(epochs, loss, "bo", label="Training loss")
plt.plot(epochs, val_loss, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.legend()
plt.show()
```



Analysis of the Graph:

The above graph represents:

Accuracy: The training and validation accuracy are similar and increase over time, reaching a peak around epoch 10. This suggests that the model is learning well and generalizing to unseen data.

Loss: The training and validation loss are both decreasing over time, indicating that the model is making fewer mistakes and improving its performance. The validation loss is slightly higher than the training loss, but it is not significantly different.

Results:

The below table represents the results of accuracy and validation loss for each approach.

Train Size	Test Size	Validation Size	Data Augmentation	Train Accuracy (%)	Validation Accuracy (%)
1000	500	500	NO	77.45	68.6
1000	500	500	YES	69.75	70.2
1500	500	500	NO	85.27	75.10
1500	500	500	YES	64.30	61.2
1500	1000	500	YES	85.13	74.35
1500	1000	500	NO	59.07	53.95

The below table shows results of pre-trained model

Data Augmentation	Train Accuracy (%)	Validation Accuracy (%)
NO	99.93	97.30
YES	96.6	96.85

The tables compare the performance of models trained with different approaches on training, validation, and test datasets, including the effects of data augmentation and the use of a pre-trained model (such as the VGG16)

Pre-Trained Model Performance

1. **Impact of Data Augmentation**:
    - Without data augmentation, the pre-trained model achieves extremely high train (99.93%) and validation (97.30%) accuracies, indicating it fits the training data very well and generalizes excellently.
    - With data augmentation, the pre-trained model's train accuracy is slightly lower (96.6%), but validation accuracy is the same (96.85%), suggesting that while the model's fit on training data is slightly less perfect, its generalization capability remains good.

    These tables illustrate the importance of data augmentation in enhancing generalization, the superior performance of pre-trained models, and the positive impact of increased training data size on model performance.

Conclusion:

The Kaggle Cats vs Dogs project aimed to classify images of cats and dogs using Convolutional Neural Networks (CNNs). The project explored various approaches, including custom models trained from scratch and models leveraging pre-trained architectures like VGG16.

Key findings and insights from the project are summarized below:

- With and Without Data Augmentation
- Impact of Training Size
- Pre-trained VGG16 Model

The project demonstrates the effectiveness of using pre-trained models for image classification tasks, achieving high accuracies with minimal training effort. Data augmentation proves to be a crucial technique for improving the generalization of custom CNN models. The findings emphasize the advantages of leveraging pre-trained architectures and the importance of dataset size and augmentation in developing robust image classifiers.