ASSIGNMENT-2
CONVOLUTION NETWORKS REPORT

RUPESH SURAGANI

KEERTHI TIYYAGURA

Introduction:

The purpose assignment is typically to apply convolution networks (Convnets) to image data to classify images of cats and dogs correctly. This is to demonstrate how CNNs can learn to differentiate between different categories of images and to provide a practical introduction of training and evaluating deep learning models for image classification tasks.

Goal of the project:

To classify what the nature of the image, among all the images of cats and dogs.

Methodology:

There are two broad approaches to classifying Cats and Dogs using convnets:

- Training a neural network from the scratch.
- Using a pretrained convnet.

We have used google colab notebook for our assignment. For downloading the cats-vs-dogs dataset, we have chosen Kaggle platform as a data source and uploaded Kaggle Json file which enabled us to create a directory and directly download the required dataset into the notebook environment in a compressed version. The cats-vs-dogs dataset consists of nearly 25000 images of both cats and dogs.

Question 1:

Consider the Cats & Dogs example. Start initially with a training sample of 1000, a validation sample of 500, and a test sample of 500 (like in the text). Use any technique to reduce overfitting and improve performance in developing a network that you train from scratch. What performance did you achieve?

For this we have created a new directory called “Cats_vs_dogs_1” and created a subset of larger dataset containing the images of cats and dogs by taking selected range of images to the new directory. We have made three subsets for training, validation, and testing.

- Took a range of 1000 sample images to training subset.
- Range of 500 sample images to validation subset.
- Range of 500 sample images to testing subset.

For providing a convenient and efficient way to represent and manipulate data for training, validation, and testing machine learning models we have preprocessed the images as TensorFlow datasets resizing the images to 180 x 180 pixels with the batch size of 32 with the help of NumPy array.

To accommodate larger images and address a more intricate problem, we scaled up our model accordingly. This entails incorporating two additional sets of Conv2D and MaxPooling2D layers. This augmentation serves a dual purpose: enhancing the model's capacity and further downsizing the feature maps to prevent them from becoming excessively large by the time they reach the Flatten layer.

Beginning with input images sized at 180 pixels × 180 pixels (a somewhat arbitrary decision), this adjustment results in feature maps sized at 7 × 7 just prior to the Flatten layer.

To address a binary classification task, we ended the model with a single unit, a Dense layer of size 1, accompanied by a sigmoid activation function. This unit will represent the probability that the model assigns to one of the two classes. Additionally, we introduced a Rescaling layer at the beginning of the model. This

layer will normalize the input images, originally ranging from 0 to 255, to fit within the [0, 1] range.

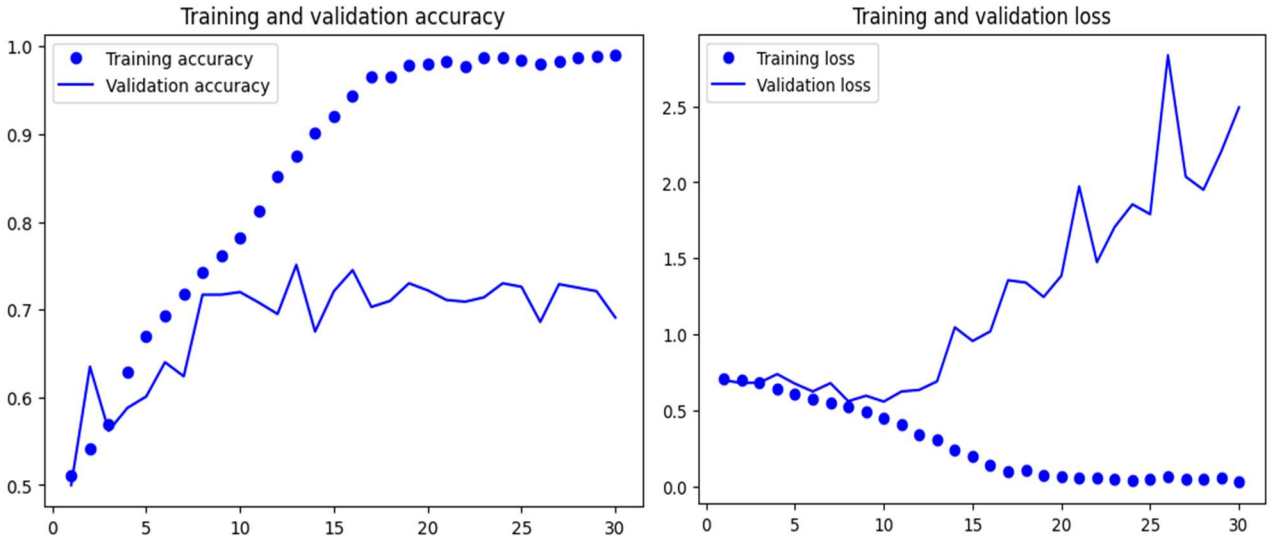
Summary of Model_1:

The model consists of several layers including convolutional layers (Conv2D), max pooling layers (MaxPooling2D), a flattening layer (Flatten), and a dense layer (Dense). The input layer (input_1) has a shape of (180, 180, 3), indicating images with a height and width of 180 pixels and 3 color channels (RGB). Each layer is described with its type, output shape, and number of parameters. The output shape for each layer is represented as **None, height, width, channels.**

Trainable and non-trainable parameters:

The number of parameters for each layer indicates the total number of trainable and non-trainable parameters in the layer. These parameters include weights and biases that the model learns during training to make predictions. The total number of parameters for the model is 991,041, indicating that all are trainable and can learn from data during training.

With the batch size taken as 225, we have applied the data flattening technique to make data transformation. By applying 30 epochs, we gained a validation accuracy of 69.1% and test accuracy of 68.6%. Below is its graphical representation.



From the above plots, we came to know that when the training accuracy is 98.95% with no data augmentation, the test accuracy is 69.1%.

Question 2:

Increase your training sample size. You may pick any amount. Keep the validation and test samples the same as above. Optimize your network (again training from scratch). What performance did you achieve?

For raising the accuracy of the model, we have chosen Data augmentation approach. Overfitting generally occurs when the available training data is insufficient, preventing the model from generalizing well to unseen data. Data augmentation involves creating additional training data by applying random transformations to existing samples. The aim is to ensure that during training, the model encounters diverse variations of the same image, enhancing its ability to generalize effectively. In our scenario, the Data augmentation approach helps in applying random changes to the training data to produce new data.

By training our new model with data augmentation configuration, the model will not see the same image for the second time. However, the inputs it observes remain strongly correlated because they originate from a limited set of original images.

Here, we have increased our training samples from 1000 to 1500 keeping the validation and test samples same at 500 each.

Images showing the trained augmented pictures:

We have applied 30 epochs and have gained a validation accuracy of 76.7% and test accuracy of 76% which is better than our previous result in the 1st Task (Validation accuracy – 69.1%, test accuracy – 68.6%)

Question 3:

Now change your training sample so that you achieve better performance than those from Steps1 and 2. This sample size may be larger, or smaller than those in the previous steps. The objective is to find the ideal training sample size to get the best prediction results.?

- We know that usage of more and more samples will help in enhancing the model's performance.

- So, we chose to increase our training samples from 1500 to 2000.

- Now, our test sets include 2000 samples for training, 500 for validation and 500 for testing.

- On increasing the training samples to 2000 we observed that the validation accuracy and test accuracy are much higher than the previous task.

The results for different training sample ranges from task2 and task3 with data augmentation are as follows:

Tasks with data augmentation	Training Sample Range	Training Accuracy	Validation Accuracy	Test Accuracy
Question 2	1500	79.95%	76.7%	76%
Question 3	2000	81.5%	81.2%	79.2%

Question 4:

Repeat Steps 1-3, but now using a pretrained network. The sample sizes you use in Steps 2 and 3 for the pretrained network may be the same or different from those using the network where you trained from scratch. Again, use all optimization techniques to get the best performance.

Using Pretrained network:

Pretrained Models are the common and highly effective approach for deep learning on image datasets. Here, the model is previously trained on a larger dataset typically for a large-scale image-classification task. When the original dataset is sufficiently large and diverse, the spatial hierarchy of features learned by a pretrained model can serve as a generic representation of the visual world. Consequently, these features can be valuable for various computer vision tasks, even if those tasks involve entirely different classes than the original training task.

Pretrained networks are mostly used for feature extraction and fine-tuning. An instance of analyzing a large, pretrained convolutional neural network involves utilizing the ImageNet dataset. This dataset comprises 1.4 million annotated images across 1,000 distinct classes. Among these classes are various animal categories, including multiple breeds of dogs and cats. The specific architecture employed for this network is VGG16, a popular and straightforward convolutional neural network design tailored for Image dataset.

Data augmentation involves creating new data samples by applying random modifications and without data augmentation, the model learns from the original data without any modifications. The prediction results for the pretrained model network with feature extraction and fine tuning are as follows:

Pretrained Network	**Training Accuracy**	**Validation Accuracy**	**Test Accuracy**
Feature Extraction	99.9%	97.8%	97%
Fine Tuning	91.95%	97.8%	96.6%

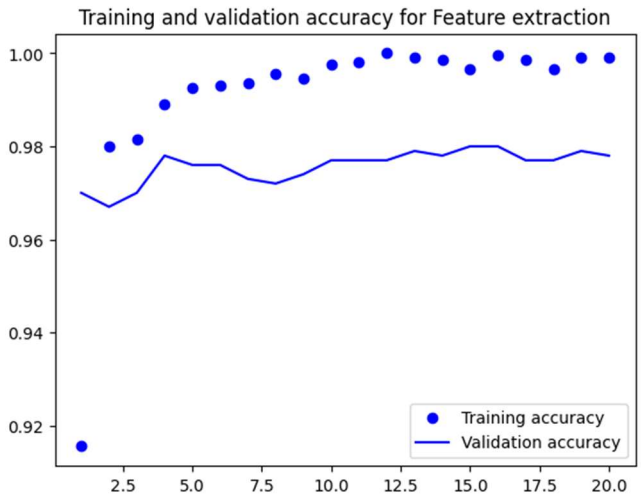


Fig: ***Feature Extraction***

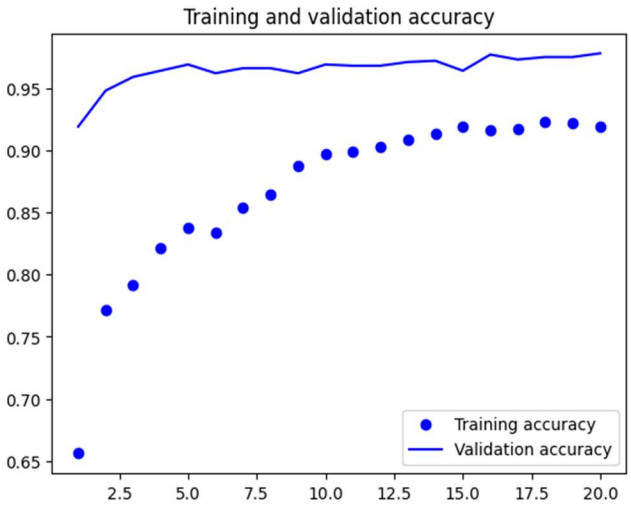


Fig: ***Fine Tuning***

Similarly, we have run the pretrained models repeating 1-3 steps i.e increasing the training samples for three times and found better predictions which are as follows:

Pretrained model samples	Validation Accuracy	Test Accuracy
1000	97.8%	96.6%
8000	97.7%	97.6%
10000	97.5%	97.7%

From the above results we understood that the overall performance metrices went better on increasing the training sample sizes with fine tuning approach.

Conclusion:

The conclusion drawn from the experimentation on building a convolutional neural network (CNN) for classifying cat and dog images is rich in insights into the various aspects of model development and optimization. Let's break down each key point:

Increasing Training Sample Size:

As the size of the training dataset increased, the model's performance improved. This observation aligns with the intuition that more diverse and abundant data provide the model with better learning opportunities and a richer understanding of the task at hand.

Data Augmentation:

Data augmentation emerged as a crucial technique for enhancing model generalization. By artificially expanding the training dataset through random transformations applied to existing images, the model was exposed to a wider variety of data, thereby improving its ability to generalize to unseen examples.

Pre-Trained Models:

Leveraging pre-trained models, particularly those trained on large and diverse datasets like ImageNet, proved to be highly beneficial. These models come with learned features that capture a broad range of visual patterns and concepts, enabling them to perform well on related tasks with minimal fine-tuning.

Overfitting Mitigation:

Overfitting, a common issue in machine learning, was effectively addressed through techniques such as dropout regularization and limiting the complexity of the network. Dropout regularization involves randomly dropping out a proportion of neurons during training, preventing the model from relying too heavily on specific features or patterns present only in the training data.

Overall Importance:

The conclusion underscores the critical importance of thoughtful model architecture selection, hyperparameter tuning, and the application of advanced techniques like data augmentation and transfer learning. These components play pivotal roles in achieving accurate and robust image classification models, particularly in scenarios where training data is limited or when tackling complex tasks.

Advantages of Pre-Trained Models:

Pre-trained models offer significant advantages, including faster development time, improved performance, and better generalization. By leveraging the knowledge distilled from vast datasets, these models provide a head start in training and often yield superior results compared to models built from scratch, especially when fine-tuned to specific tasks.