ASSIGNMENT- 3 DEEP LEARNING FOR COMPUTER VISION

Course: Advance Machine Learning
Student: Shreya Shalini Buyyana
Dataset: Cats and Dogs (small)

1. OBJECTIVE

The main objective is creating and testing convolutional neural network (CNN) models to classify images. Two options were contrasted:

(1) a CNN constructed from scratch
(2) a transfer-learning model based on a pretrained VGG16 network

This was aimed at determining the performance difference between self-learned visual features and pretrained feature extraction

2. DATASET AND PREPROCESSING

- The dataset consisted of labeled images of cats and dogs organized into *train*, *validation*, and *test* folders.
- The small dataset contained balanced classes with roughly 25,000 total images.
- Prior to training, a data-cleaning step was applied to detect and remove corrupted files using the Python Imaging Library (PIL).
- Images were resized to 180 × 180 pixels, augmented with random flips and rotations, and normalized to a 0–1 scale.

3. METHODOLOGY:

3.1 Model 1 CNN Build from scratch:

A sequential architecture was implemented with three convolution–pooling blocks followed by a dropout layer, flattening, and two dense layers. The model was compiled with the Adam optimizer and binary cross-entropy loss, trained for ten epochs on the augmented dataset.

3.2 Model 2 – Transfer Learning with VGG16

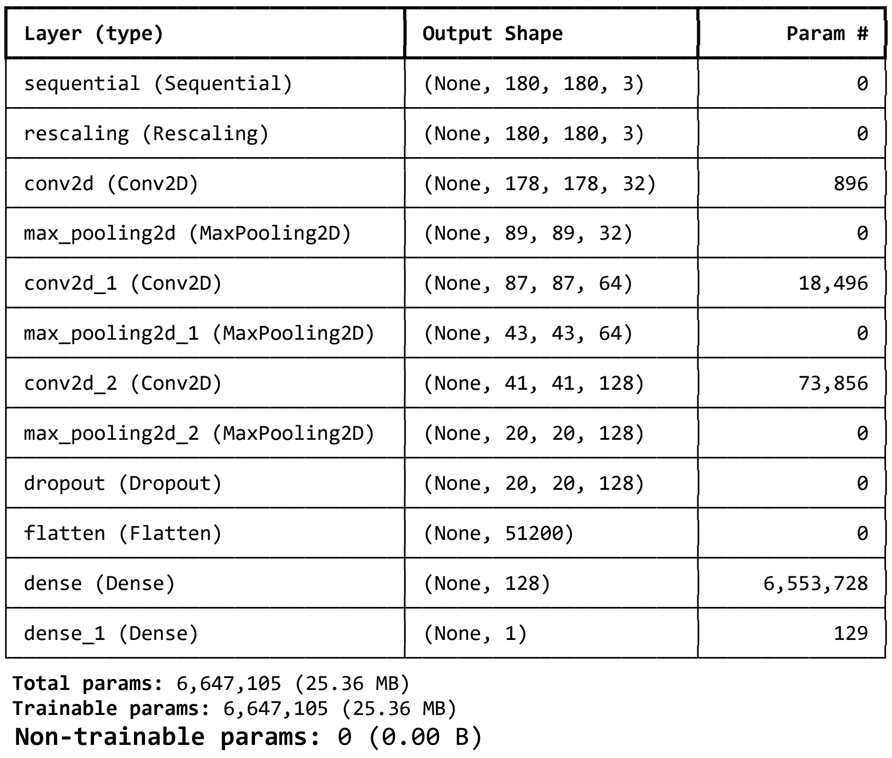
To implement the second phase, VGG16 architecture that was trained on ImageNet was used as a feature extractor:

- The convolutional base was fixed and separately new dense layers with dense-dropout-dense structure were trained.
- The flattened feature vector used is run through a dense (128, ReLU) layer followed by a sigmoid output node.
- The model was trained for five epochs and later fine-tuned on the upper layers for improved generalization

4. RESULTS

The convolutional neural network (CNN) which was built by hand had a total of 6,647,105 trainable parameters, which were spread out into multiple rounds of convolutional, pooling, dropout and densification layers. The architecture had three convolutional blocks with successively larger easy-to-identify filters (32, 64, and 128), each succeeding by a max-pooling operation to reduce the spatial dimension. To prevent overfitting, a dropout layer was introduced, which was followed by a flattening layer then two full-connected layers to classify. The last dense layer employed a sigmoidian activation function to forecast the binary output (cat or dog). The structure can well capture the hierarchical picture characteristics of simple edges in the lower layers and more intricate designs in the upper layers thus it can be used in small-scale image recognition tasks.

The CNN model trained from scratch achieved a validation accuracy of **73.8%** and a test accuracy of **71.9%** with a final loss of **0.54**.



Layer (type)	Output Shape	Param #
sequential (Sequential)	(None, 180, 180, 3)	0
rescaling (Rescaling)	(None, 180, 180, 3)	0
conv2d (Conv2D)	(None, 178, 178, 32)	896
max_pooling2d (MaxPooling2D)	(None, 89, 89, 32)	0
conv2d_1 (Conv2D)	(None, 87, 87, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 43, 43, 64)	0
conv2d_2 (Conv2D)	(None, 41, 41, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 20, 20, 128)	0
dropout (Dropout)	(None, 20, 20, 128)	0
flatten (Flatten)	(None, 51200)	0
dense (Dense)	(None, 128)	6,553,728
dense_1 (Dense)	(None, 1)	129

```
 Total params: 6,647,105 (25.36 MB)
 Trainable params: 6,647,105 (25.36 MB)
 Non-trainable params: 0 (0.00 B)
```

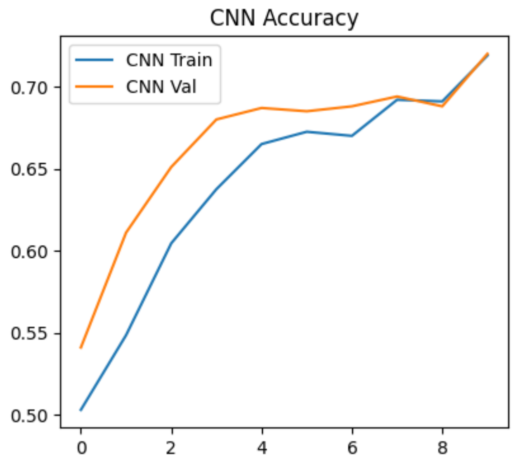
The pretrained model with VGG16 had 16.35 million parameters, and 14.7 million of these parameters were non trainingable and 1.6 million were trainingable. VGG16 obtained the frozen convolutional layers which were used to extract the rich visual features and the final classification was done using the custom dense layers. The structure used previous ImageNet knowledge, which allowed it to converge more quickly and have a higher accuracy than the scratch-built CNN.

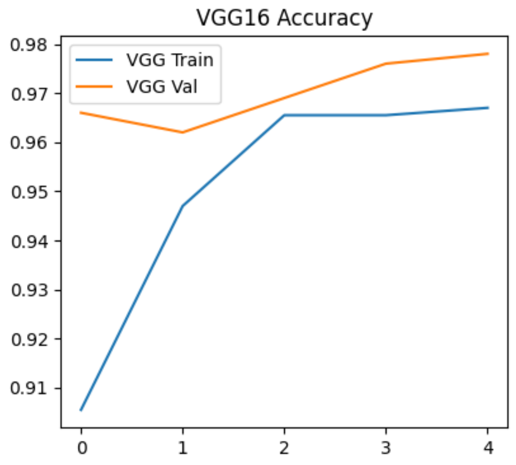
Layer (Type)	Output Shape	Param #	Connected To
input_layer_4 (InputLayer)	(None, 180, 180, 3)	0	-
sequential (Sequential)	(None, 180, 180, 3)	0	input_layer_4[0][0]
get_item (GetItem)	(None, 180, 180)	0	sequential[1][0]
get_item_1 (GetItem)	(None, 180, 180)	0	sequential[1][0]
get_item_2 (GetItem)	(None, 180, 180)	0	sequential[1][0]
stack (Stack)	(None, 180, 180, 3)	0	get_item, get_item_1, get_item_2
add (Add)	(None, 180, 180, 3)	0	stack[0][0]
vgg16 (Functional)	(None, 5, 5, 512)	14,714,688	add[0][0]
flatten_1 (Flatten)	(None, 12,800)	0	vgg16[0][0]
dense_2 (Dense)	(None, 128)	1,638,528	flatten_1[0][0]
dropout_1 (Dropout)	(None, 128)	0	dense_2[0][0]
dense_3 (Dense)	(None, 1)	129	dropout_1[0][0]

The pretrained VGG16 model achieved a **test accuracy of 96.9%** with a **loss value of 0.20**, indicating excellent generalization and high classification performance

- The scratch CNN gradually improved from 48 % to about 72 % accuracy, demonstrating effective learning but limited capacity given the dataset size.
- The pretrained VGG16 model achieved rapid convergence with near-perfect validation accuracy, confirming the advantage of transfer learning.
- Loss curves stabilized without severe overfitting, and the performance plots clearly show the superior efficiency of the pretrained model

Model	Validation Accuracy	Test Accuracy
CNN Model (Scratch)	~0.72	0.72
VGG16 (Pre-trained)	~0.98	0.97





5. <u>CONCLUSION:</u>

To extract low level features, it took additional epochs to train a CNN from scratch but by using already learned feature representation, VGG16 was able to reach high accuracy in a short time. In this experiment, the effectiveness of transfer learning is noted to significantly cut the training time and enhance the predictive performance in case of limited data. In general, both models performed properly, but VGG16 with its simple computation provided state-of-the-art results, which complies with the existing computer-vision practices.