In [ ]:

```
##Lab14
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.preprocessing.image import ImageDataGenerator

base_model = VGG16(weights='imagenet', include_top=False, input_shape=(150,150
base_model.trainable = False

model = Sequential([
    base_model,
    Flatten(),
    Dense(128, activation='relu'),
    Dense(2, activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['acc

train_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(
    'train_data/', target_size=(150,150), batch_size=32, class_mode='categoric

validation_generator = train_datagen.flow_from_directory(
    'val_data/', target_size=(150,150), batch_size=32, class_mode='categorical

history = model.fit(train_generator, epochs=5, validation_data=validation_gene

# Plot training & validation accuracy
import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy vs Epochs')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

Exp-14: Implement a Pre-Trained CNN Model as a Feature Extractor Using Transfer Learning

Aim:
To use a pre-trained CNN (like VGG16 or ResNet50) as a feature extractor for a new image classification task.

Description:
Transfer learning leverages features learned from large datasets and applies them to smaller, related tasks.
The convolutional base of the model is frozen, and new dense layers are added for specific classification outputs.

Procedure:
1.) Load a pre-trained without its top classifier.
2.) Freeze convolutional layers.
3.) Add a dense layers for the target dataset
4.) Compile and train the new model.
5.) Evaluate accuracy and visualize predictions.

Pseudocode:

base = VGG16(include_top = false, weights = 'imagenet')
freeze base layers.
Add Flatten → Dense → Output layers
Compile and train on new dataset
Evaluate and predict

Observation:
Model acheives high accuracy with fewer epochs due to resused feature maps.
feature extraction reduces training time and data requirements.

Result:
The transfer-learning model performed efficient classification using pre-trained CNN features.

Output:

-- start training (Feature Extraction) --

Epoch [1/3], Loss: 2.6552

Epoch [2/3], Loss: 2.3339

Epoch [3/3], Loss: 2.3197

Training of the new head complete.

Avg loss vs Epoch