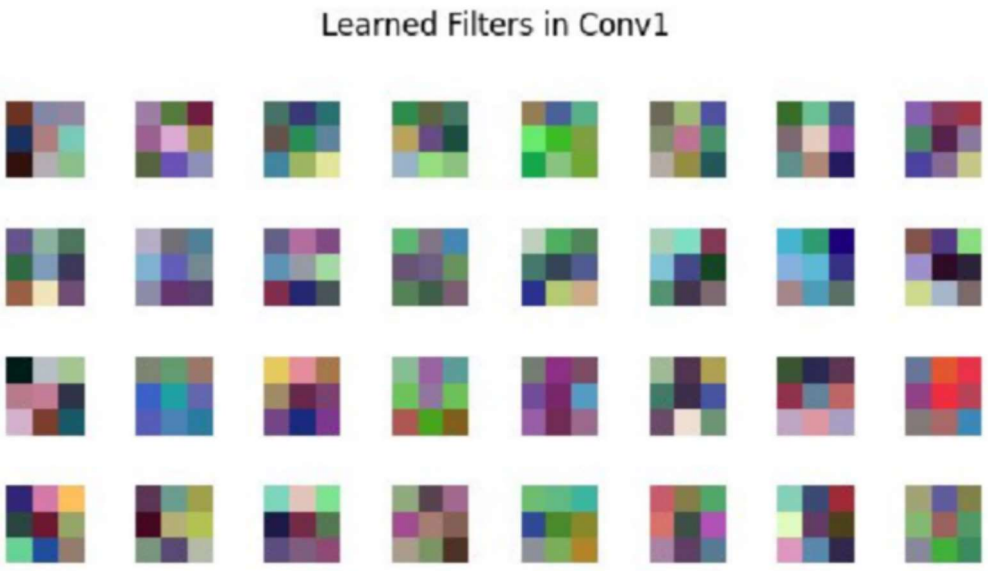
```
# Filter Visualization
weights = model.conv1.weight.data.cpu()
grid = torchvision.utils.make_grid(weights, nrow=8, normalize=True, pad_value=1)
plt.figure(figsize=(8,8))
plt.imshow(np.transpose(grid.numpy(), (1,2,0)))
plt.title("Learned Filters in Conv1")
plt.axis("off")
plt.show()
```

OUTPUT:



RESULT: The CNN model achieved a test accuracy of 72% on the CIFAR-10 dataset, and the visualization clearly shows the learned edge- and texture-detecting filters from the first convolution layer.

EXP NO: 05 DATE: 06/09/2025	**IMAGE CLASSIFICATION USING VGGNet, ResNet and GoogLeNet**

AIM: To implement and compare the performance of three well-known Convolutional Neural Network (CNN) architectures: VGG16, ResNet50, and InceptionV3 for image classification using Dogs vs Cats dataset. The comparison is based on validation accuracy and validation loss.

ALGORITHM:

- Load the Cats vs Dogs dataset from TensorFlow Datasets.
- Preprocess images: resize to 150×150 and normalize pixel values.
- Create separate models using pretrained architectures (VGG16, ResNet50, InceptionV3).
- Freeze base layers to perform transfer learning.
- Add classifier layers: Dense(128, relu) + Dense(1, sigmoid).
- Train each model for equal epochs and store history.
- Plot and compare validation accuracy and validation loss.
- Interpret which model performs best.

CODE:

```
import tensorflow as tf
import tensorflow_datasets as tfds
import matplotlib.pyplot as plt

# Load dataset (8% train, 2% validation)
(train_data, val_data), info = tfds.load(
    'cats_vs_dogs',
    split=['train[:8%]', 'train[8%:10%]'],
```

231501058

```
    with_info=True,
    as_supervised=True
)

# Preprocessing function
def preprocess(image, label):
    image = tf.image.resize(image, (150, 150))
    image = image / 255.0
    return image, label

batch_size = 32
train_ds = train_data.map(preprocess).shuffle(500).batch(batch_size)
val_ds = val_data.map(preprocess).batch(batch_size)

# CNN architectures
models = {
    "VGG16": tf.keras.applications.VGG16,
    "ResNet50": tf.keras.applications.ResNet50,
    "InceptionV3": tf.keras.applications.InceptionV3
}

history_dict = {}

# Train each model
for name, architecture in models.items():
    print(f"\n===== Training {name} =====")

    base_model = architecture(
```

```
    weights='imagenet',
    include_top=False,
    input_shape=(150,150,3),
    pooling='avg'
  )
  base_model.trainable = False

  model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
  ])

  model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy']
  )

  history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=5,
    verbose=1
  )

  history_dict[name] = history
```