S. B. JAIN INSTITUTE OF TECHNOLOGY, MANAGEMENT & RESEARCH, NAGPUR.

Practical No. 12 (Post Lab)

Aim: To build a deep learning model using transfer learning (ResNet-18) for classifying chest X-ray images into **COVID-19 infected** and **normal** categories.

Name of Student: shrutika Pradeep Bagdi

Roll No.: CS22130

Semester/Year: VIIth/IVth

Academic Session: 2024-25

Date of Performance:

Date of Submission:

AIM: To build a deep learning model using transfer learning (ResNet-18) for classifying chest X-ray images into **COVID-19 infected** and **normal** categories.

OBJECTIVE/EXPECTED LEARNING OUTCOME:

- To understand image classification using deep learning.
- To apply transfer learning with ResNet-18.
- To train and test a model for COVID-19 detection.

HARDWARE AND SOFTWARE REQUIRMENTS:

Hardware Requirement:

- Processor: Dual Core
- RAM: 1GB
- HARD DISK: > 80 GB

Software Requirement:

- OS: Windows
- PyCharm IDE

THEORY:

Image classification is a key application of deep learning in healthcare that helps in diagnosing diseases from medical images. Convolutional Neural Networks (CNNs) are widely used for this task as they can automatically extract important visual features.

In this experiment, transfer learning is applied using a pretrained **ResNet-18** model, which has already learned rich image features from a large dataset (ImageNet). By fine-tuning the final layers, the model is adapted to classify **chest X-ray images** into two categories — **COVID-19 infected** and **Normal**.

This approach reduces training time, improves accuracy, and helps in automating the early detection of COVID-19 from X-ray scans, supporting faster and more reliable medical diagnosis.

Applications:

- Early detection of COVID-19 from chest X-ray images.
- Assisting radiologists in diagnosing lung infections.
- Automating medical image analysis in hospitals and clinics.
- Supporting telemedicine and remote healthcare services.

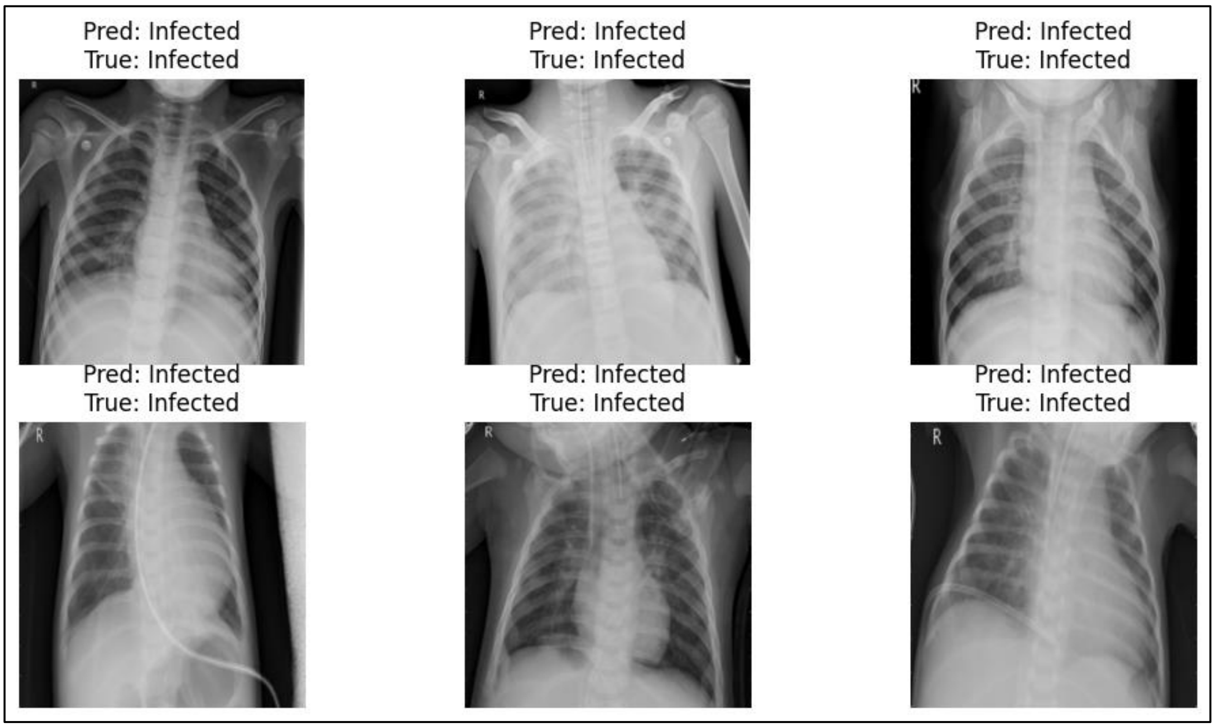
CODE:

```
# 1. Install and Import Dependencies
!pip install kaggle tqdm torch torchvision matplotlib --quiet
import os
import shutil
import zipfile
import torch
from torch import nn, optim
from torchvision import datasets, transforms, models
from torch.utils.data import DataLoader
import matplotlib.pyplot as plt
from tqdm import tqdm
import pandas as pd
from sklearn.model_selection import train_test_split
import numpy as np
# 2. Download Dataset from Kaggle
from google.colab import files
files.upload()   # upload kaggle.json
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
!kaggle datasets download -d praveengovi/coronahack-chest-xraydataset
os.makedirs("/content/dataset", exist_ok=True)
with zipfile.ZipFile("coronahack-chest-xraydataset.zip", "r") as zip_ref:
    zip_ref.extractall("/content/dataset")
# 3. Organize the Dataset
base_train   =   "/content/dataset/Coronahack-Chest-XRay-Dataset/Coronahack-Chest-XRay-
Dataset/train"
base_test   =   "/content/dataset/Coronahack-Chest-XRay-Dataset/Coronahack-Chest-XRay-
Dataset/test"
metadata_path = "/content/dataset/Chest_xray_Corona_Metadata.csv"
metadata = pd.read_csv(metadata_path)
metadata = metadata.dropna(subset=["Label"])
```

```
metadata["Label"] = metadata["Label"].apply(lambda x: "Infected" if x != "Normal" else
"Normal")
train_df, temp_df = train_test_split(metadata, test_size=0.3, stratify=metadata["Label"],
random_state=42)
val_df, test_df = train_test_split(temp_df, test_size=0.5, stratify=temp_df["Label"],
random_state=42)
organized_dir = "/content/dataset/organized_data"
for subset in ["train", "val", "test"]:
    for label in ["Normal", "Infected"]:
        os.makedirs(os.path.join(organized_dir, subset, label), exist_ok=True)
def find_image_path(filename):
    if os.path.exists(os.path.join(base_train, filename)):
        return os.path.join(base_train, filename)
    elif os.path.exists(os.path.join(base_test, filename)):
        return os.path.join(base_test, filename)
    else:
        return None
def copy_images(file_df, subset):
    for _, row in tqdm(file_df.iterrows(), total=len(file_df), desc=f"Organizing {subset} data"):
        src = find_image_path(row["X_ray_image_name"])
        if src:
            dst = os.path.join(organized_dir, subset, row["Label"], row["X_ray_image_name"])
            shutil.copy(src, dst)
copy_images(train_df, "train")
copy_images(val_df, "val")
copy_images(test_df, "test")
for folder in ["train", "val", "test"]:
    print(f"{folder}: {sum([len(files) for r, d, files in os.walk(os.path.join(organized_dir,
folder))])} images")
# 4. Data Preprocessing
data_transforms = {
    'train': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],
                             [0.229, 0.224, 0.225])
    ]),
    'val': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],
                             [0.229, 0.224, 0.225])
    ]),
    'test': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],
                             [0.229, 0.224, 0.225])
    ]),
}
# 5. Load Data
data_dir = organized_dir
image_datasets = {x: datasets.ImageFolder(os.path.join(data_dir, x), data_transforms[x])
                  for x in ['train', 'val', 'test']}
dataloaders = {x: DataLoader(image_datasets[x], batch_size=16, shuffle=True,
num_workers=2)
               for x in ['train', 'val', 'test']}
class_names = image_datasets['train'].classes
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("Classes:", class_names)
# 6. Define the Model (Transfer Learning)
model = models.resnet18(pretrained=True)
for param in model.parameters():
    param.requires_grad = False
num_ftrs = model.fc.in_features
model.fc = nn.Linear(num_ftrs, len(class_names))
model = model.to(device)
```

```
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.fc.parameters(), lr=0.001)
# 7. Train the Model
epochs = 5
for epoch in range(epochs):
    model.train()
    running_loss = 0.0
    for inputs, labels in tqdm(dataloaders['train'], desc=f"Epoch {epoch+1}/{epochs}"):
        inputs, labels = inputs.to(device), labels.to(device)
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item() * inputs.size(0)
    epoch_loss = running_loss / len(image_datasets['train'])
    print(f"Epoch {epoch+1} - Training Loss: {epoch_loss:.4f}")
print("Training completed!")
# Evaluate the Model
model.eval()
correct, total = 0, 0
with torch.no_grad():
    for inputs, labels in dataloaders['test']:
        inputs, labels = inputs.to(device), labels.to(device)
        outputs = model(inputs)
        _, preds = torch.max(outputs, 1)
        total += labels.size(0)
        correct += (preds == labels).sum().item()
accuracy = 100 * correct / total
print(f"Test Accuracy: {accuracy:.2f}%")
# Visualize Some Predictions
def imshow(inp, title=None):
    inp = inp.numpy().transpose((1, 2, 0))
    mean = np.array([0.485, 0.456, 0.406])
    std = np.array([0.229, 0.224, 0.225])
    inp = std * inp + mean
    inp = np.clip(inp, 0, 1)
    plt.imshow(inp)
    if title:
        plt.title(title)
    plt.axis('off')
inputs, classes_idx = next(iter(dataloaders['test']))
outputs = model(inputs.to(device))
_, preds = torch.max(outputs, 1)
plt.figure(figsize=(12, 6))
for i in range(6):
    ax = plt.subplot(2, 3, i+1)
    imshow(inputs[i])
    ax.set_title(f"Pred: {class_names[preds[i]]}\nTrue: {class_names[classes_idx[i]]}")
plt.show()
```

OUTPUT :



CONCLUSION:

The deep learning model successfully classified chest X-ray images into **COVID-19 infected** and **Normal** categories, demonstrating the effectiveness of **transfer learning** in medical image diagnosis.

DISCUSSION AND VIVA VOCE:

1) What is the purpose of using ResNet-18 in this experiment?
2) Why is data preprocessing important for image classification?
3) How is the performance of the model evaluated?

REFERENCE:

- *www.w3schools.com*
- *www.tutorialsmade.com*
- *www.towardsdatascience.com*