UG-PROJECT

LUNG CANCER DETECTION USING ZERO AND FEW-SHOT TECHNIQUES

SUPERVISED BY-
DR.SUNIL KUMAR

ANSHIKA YADAV
(22124010)

INTRODUCTION

About Lung Cancer:

Cancer is a disease in which cells in the body grow out of control.
When cancer starts in the lungs, it is called lung cancer.
Lung cancer begins in the lungs and may spread to lymph nodes or other organs in the body, such as the brain.
Cancer from other organs also may spread to the lungs.
One of the main cause of lung cancer is smoking.Fever, cough, and difficulty breathing are among the symptoms of respiratory issues.

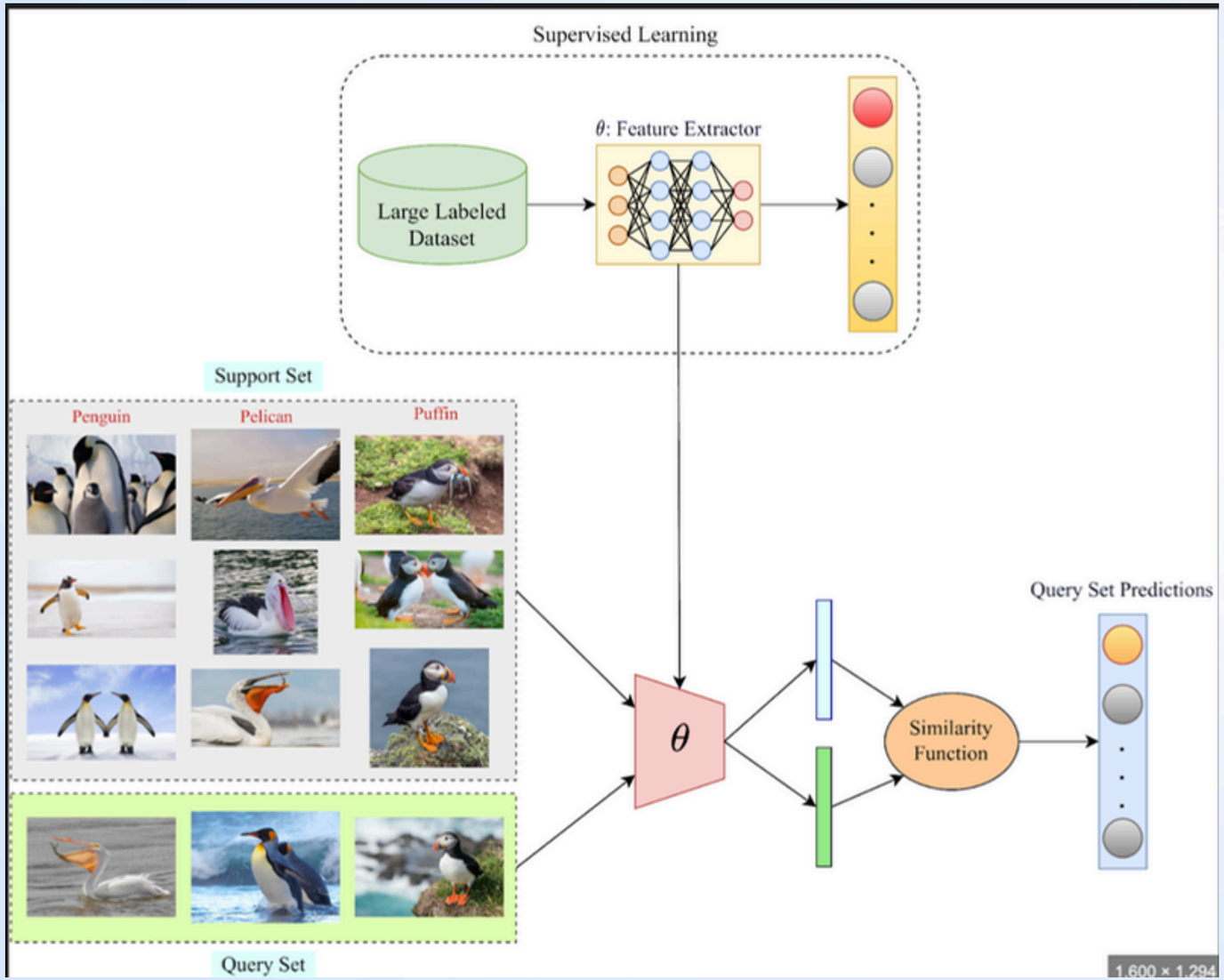
SHOT TECHNIQUES

Shot techniques refer to how many examples (shots) per class are used to train a model in low-data settings. It mimics human intelligence by generalizing from a small number of examples.

Types of Shot Learning:

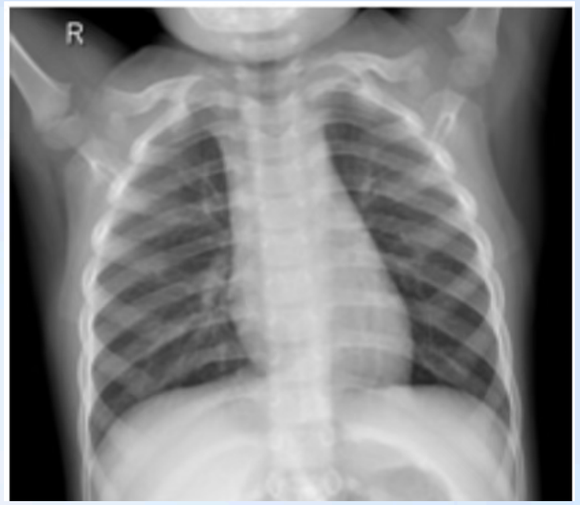
- **Zero-Shot Learning:** The model generalizes to new classes without any labeled examples uses semantic knowledge to classify unseen categories.
- **One-Shot Learning:** The model learns from a single example per class.
- **Few-Shot Learning:**The model learns from a few example (usually 5-10) per class.

Few-Shot Learning

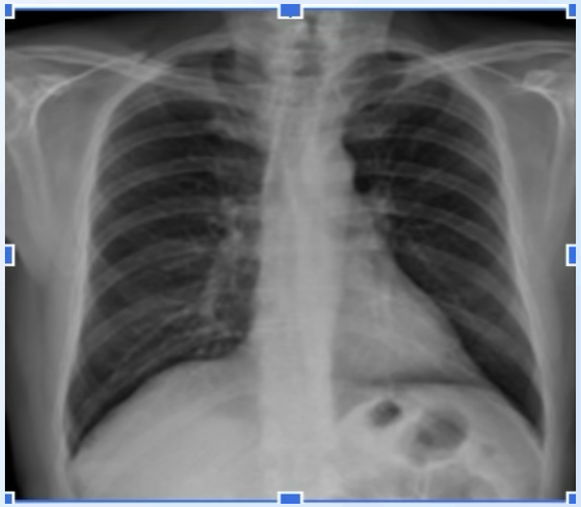


This image illustrates the concept of Few-Shot Learning (FSL) using a support set, query set(consists of unlabeled images that the model needs to classify based on a small support set),
and a feature extractor(model or a part of a model that processes raw data and converts it into a meaningful feature representation.These extracted features are then used for classification, clustering, or other machine learning tasks)
trained via supervised learning.

DATASET INFORMATION



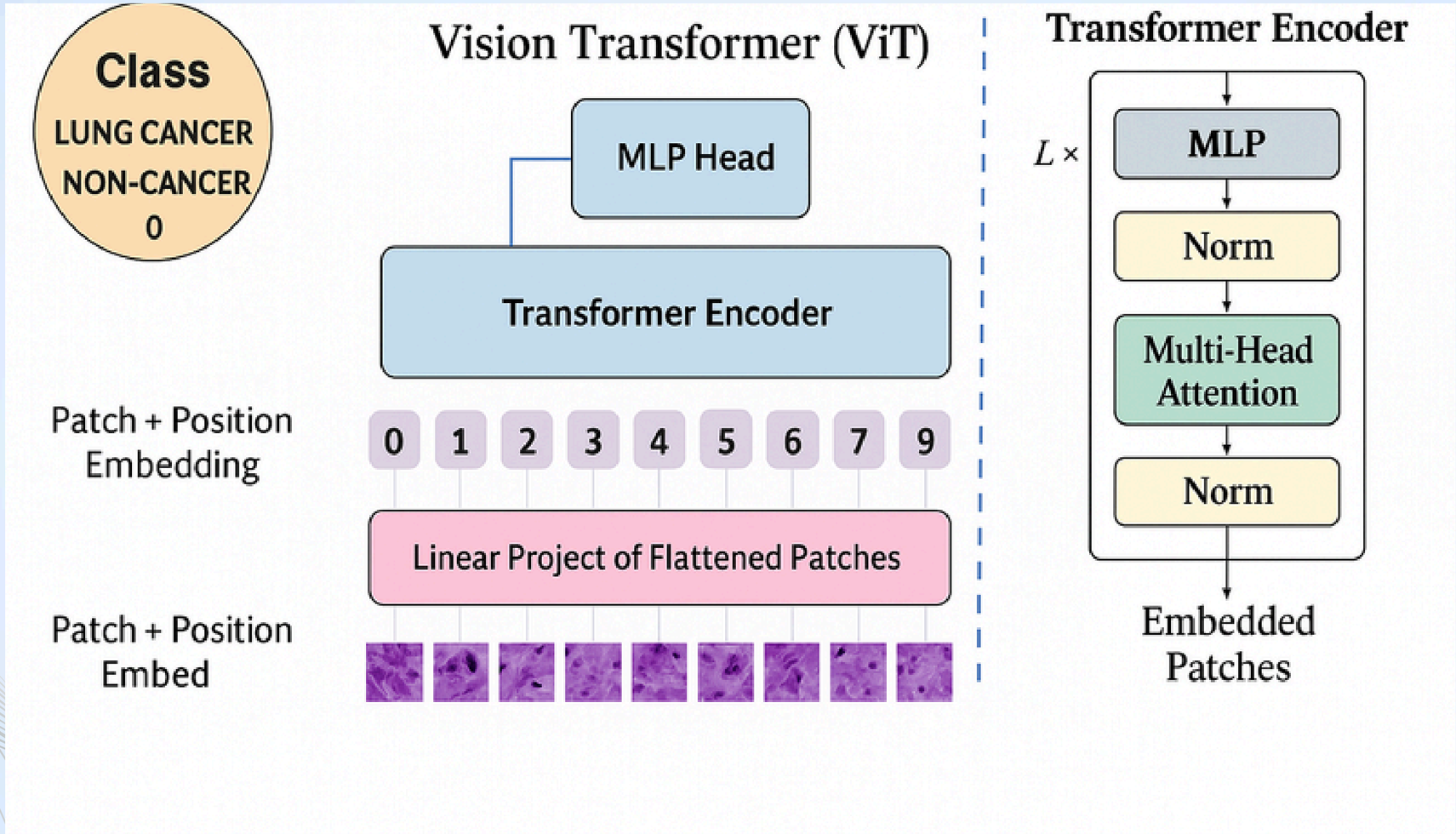
- Total Number of Images: 284
- Training Dataset: 224 Images
  - Class 1 – Lung Cancer
  - Class 0 – Non-Cancer
- Testing Dataset: 60 Images
  - Used to evaluate model performance on unseen data
- Classification Goal: Detect and classify medical images as:
  - 1 → Lung Cancer
  - 0 → Non-Cancer
- Data Format: Chest X-ray / CT scan images
- Task Type: Binary Classification



MODEL

The model was first pretrained on a large-scale image dataset (such as **ImageNet**) using Vision Transformer, which enabled it to learn general visual features like edges, textures, and patterns. This pretraining step gave the model a strong foundational understanding of images, allowing it to be adapted to new tasks more effectively, especially when only limited data is available.After pretraining, the model was fine-tuned using few-shot learning techniques on a small, domain-specific dataset containing lung cancer and non-cancer images.Few-shot learning allowed the model to learn effectively from a small number of labeled examples, which is crucial in medical domains where data is often scarce.By leveraging the pretrained knowledge from the **Vision Transformer** and fine-tuning it on a limited dataset, the model was able to achieve strong performance even with minimal training data.

Model-Overview : Vision Transformer for lung Cancer Classification



The image above illustrates the workflow of the Vision Transformer (ViT)
used in lung cancer classification task

Workflow of Vision Transformer

- **Input Image Splitting:**
  The input image (e.g., lung tissue scan) is divided into small fixed-size image patches.
- **Patch Embedding:**
  Each patch is flattened and passed through a linear layer to create a vector.
  Positional embeddings are added to retain spatial information (since transformers lack inherent spatial awareness).
- **Transformer Encoder:**
  Embedded patches are fed into the Transformer Encoder.
- **Each encoder block contains:**
  Multi-Head Attention: Captures relationships between patches.
  Layer Normalization (Norm): Stabilizes learning.
  MLP (Multi-layer Perceptron): Adds non-linearity and depth.
  These layers are repeated L times.
- **Classification Head (MLP Head):**
  The final encoded representation is passed to an MLP head for classification.
  Example: Predicting 0 for “Non-Cancer” and 1 for “Lung Cancer”.

WORKING OF THE CODE

```
import os
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, Subset
from torchvision import datasets, transforms
import timm
import random
```

- Importing Python Libraries.
- timm (PyTorch Image Models) is a popular library for accessing pretrained models here this library helps to load a pretrained Vision Transformer and fine-tune it on the dataset.

```
train_transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.RandomAffine(degrees=0, shear=10),
    transforms.RandomResizedCrop(224, scale=(0.8, 1.2)),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])
])

test_transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])
])
```

- Data augmentation was applied to the training images to improve generalization and prevent overfitting.
- The test images were preprocessed consistently to ensure reliable evaluation.
- These transformations ensured that the input to the Vision Transformer was standardized and informative.

```python
train_dir = r'/content/drive/MyDrive/Data/Train'
test_dir = r'/content/drive/MyDrive/Data/Val'

if not os.path.exists(train_dir) or not os.path.exists(test_dir):
    raise FileNotFoundError("Dataset directories not found. Check the paths!")

train_dataset = datasets.ImageFolder(root=train_dir, transform=train_transform)
test_dataset = datasets.ImageFolder(root=test_dir, transform=test_transform)
```

This code loads training and testing images from Google Drive using PyTorch's ImageFolder. It checks if the paths are correct, assigns labels based on folder names, and applies the appropriate transformations (like resizing, normalization, and augmentation) to prepare the images for model training and evaluation.

```python
num_samples = int(0.05 * len(train_dataset))
indices = random.sample(range(len(train_dataset)), num_samples)
train_subset = Subset(train_dataset, indices)
print(num_samples)
```

This code creates a small subset (0.5%) of the original training dataset using random sampling. This is done to simulate a few-shot learning scenario, where the model is trained with limited labeled data.

```python
class ViTCancerClassifier(nn.Module):
    def __init__(self):
        super(ViTCancerClassifier, self).__init__()
        self.vit = timm.create_model('vit_base_patch16_224', pretrained=True)
        in_features = self.vit.head.in_features
        self.vit.head = nn.Sequential(
            nn.Linear(in_features, 256),
            nn.ReLU(),
            nn.Dropout(0.5),
            nn.Linear(256, 1)
        )

    def forward(self, x):
        return self.vit(x)
```

Implemented a Vision Transformer (ViT)-based deep learning model to classify lung CT scan images into cancerous (1) and non-cancerous (0) categories.

The ViT model was pretrained on ImageNet, enabling it to extract high-level visual features efficiently, even with limited medical data.

To adapt the pretrained model for our binary classification task, we replaced the original classification head with a custom neural network that includes:

- A fully connected (Linear) layer
- ReLU activation for non-linearity
- Dropout (0.5) to prevent overfitting
- A final output layer for binary classification

```python
criterion = nn.BCEWithLogitsLoss()
optimizer = optim.Adam(model.parameters(), lr=1e-4)
```

The model uses BCEWithLogitsLoss for binary classification and the Adam optimizer for efficient weight updates. A small learning rate of 0.0001 ensures stable training on a limited dataset.

```python
def train(model, train_loader, criterion, optimizer, device, epochs=5):
    model.train()
    epoch_accuracies = []

    for epoch in range(epochs):
        total_loss = 0.0
        correct = 0
        total = 0

        for images, labels in train_loader:
            images, labels = images.to(device), labels.to(device).float().unsqueeze(1)
            optimizer.zero_grad()
            outputs = model(images)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

            preds = torch.sigmoid(outputs) > 0.5
            correct += (preds == labels).sum().item()
            total += labels.size(0)
            total_loss += loss.item()

        accuracy = correct / total
        epoch_accuracies.append(accuracy)
        print(f"Epoch [{epoch+1}/{epochs}], Loss: {total_loss/len(train_loader):.4f}, Accuracy: {accuracy:.4f}")
```

This function is responsible for training the Vision Transformer (ViT)model on the lung cancer dataset. It performs the following key tasks:

- Initializes training mode so the model can update its parameters.
- Loads training images in batches, moves them to GPU/CPU.
- Performs forward pass to generate predictions for each batch.
- Calculates Binary Cross-Entropy loss between predicted and actual labels.
- Backpropagates the loss to compute gradients.
- Uses the Adam optimizer to update model weights and improve performance.
- Tracks accuracy and loss after each epoch to monitor learning progress.

This function ensures that the model progressively learns to distinguish between cancerous (label = 1) and non-cancerous (label = 0) images by minimizing prediction errors and refining its internal representations.

```python
def evaluate(model, test_loader, criterion, device):
    model.eval()
    correct = 0
    total = 0
    total_loss = 0.0

    with torch.no_grad():
        for images, labels in test_loader:
            images, labels = images.to(device), labels.to(device).float().unsqueeze(1)
            outputs = model(images)
            loss = criterion(outputs, labels)

            preds = torch.sigmoid(outputs) > 0.5
            correct += (preds == labels).sum().item()
            total += labels.size(0)
            total_loss += loss.item()

    print(f"Test Loss: {total_loss/len(test_loader):.4f}, Test Accuracy: {correct/total:.4f}")
```

The evaluation function checks how well the model performs on test images by calculating the average loss and accuracy without updating the model weights. It uses sigmoid activation for binary classification and reports the final performance metrics.

```python
if __name__ == '__main__':
    train(model, train_loader, criterion, optimizer, device, epochs=10)
    evaluate(model, test_loader, criterion, device)
```

This block initiates the entire model pipeline—training on the training set and testing on the evaluation set. It ensures the script executes the main learning workflow when run.

Output

```
Using device: cuda
11
Class Mapping: {'cancer': 0, 'normal lung tissue': 1}
Epoch [1/10], Loss: 0.7906, Accuracy: 0.4545
Epoch [2/10], Loss: 0.6261, Accuracy: 0.6364
Epoch [3/10], Loss: 0.4557, Accuracy: 0.8182
Epoch [4/10], Loss: 0.4897, Accuracy: 0.8182
Epoch [5/10], Loss: 0.3140, Accuracy: 0.9091
Epoch [6/10], Loss: 0.8181, Accuracy: 0.5455
Epoch [7/10], Loss: 0.1895, Accuracy: 1.0000
Epoch [8/10], Loss: 0.2130, Accuracy: 0.9091
Epoch [9/10], Loss: 0.0996, Accuracy: 1.0000
Epoch [10/10], Loss: 0.0208, Accuracy: 1.0000
Test Loss: 0.0870, Test Accuracy: 0.9667
```

MODEL PERFORMANCE OVERVIEW

EPOCHS	TECHNIQUE	ACCURACY
	Zero-shot	45
5	One-shot	50
10	One-shot	50
5	Few-shot(5)	90
10	Few-shot(5)	92
10	Few-shot(11)	96

RESULTS

- The model achieved only 45% accuracy in the Zero-shot setting, indicating poor performance without prior examples.
- In the One-shot setup, accuracy improved slightly to 50%, showing limited learning from a single example.
- Few-shot learning significantly enhanced model performance:
  With 5 examples over 5 epochs, accuracy jumped to 90%.
  Extending to 10 epochs raised accuracy to 92%.
  Using 11 examples and 10 epochs, the model reached a peak accuracy of 96%.

Model performance improves substantially with more examples and training epochs — Few-shot learning outperforms both Zero-shot and One-shot approaches.

Conclusion

This study demonstrates that Vision Transformers (ViT), pretrained on ImageNet and fine-tuned with few-shot learning, can effectively classify lung cancer from chest X-rays with minimal data. Unlike traditional CNNs, ViT effectively captures long-range dependencies in images through self-attention mechanisms, leading to strong classification performance. Despite training on only a small fraction of the dataset, the model achieves high accuracy, highlighting the power of transfer learning.

THANK YOU