

Assignment 1

areej haddaj alqarni

252001729

1.Setup & Libraries

```
import os
import numpy as np
import torch
from torchvision import datasets, transforms as T, models
from torch import nn, optim
from torch.utils.data import DataLoader
import warnings
warnings.filterwarnings("ignore")
from tqdm import tqdm
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, classification_report
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.metrics import roc_curve
import matplotlib.pyplot as plt
from sklearn.metrics import auc
from sklearn.model_selection import train_test_split
from torch.utils.data import Subset
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

2.dataset Description

download the dataset from kaggle "<https://www.kaggle.com/datasets/samuelcortinhas/cats-and-dogs-image-classification?select=test>"

The dataset contains over 1,000 images collected from Google Images. It includes images of cats and dogs in equal proportions, classified for binary classification.

3. Image augmentation and preprocessing

```
train_dir = '/kaggle/input/datasets/areejhaddaj/drag-and-drop/train'
test_dir = '/kaggle/input/datasets/areejhaddaj/drag-and-drop/test'

train_transforms = T.Compose([
    T.Resize((224, 224)),
    T.RandomHorizontalFlip(),
    T.ToTensor(),
    T.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])

val_test_transforms = T.Compose([
    T.Resize((224, 224)),
    T.ToTensor(),
    T.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])

full_train_data = datasets.ImageFolder(train_dir, transform=train_transforms)
test_data = datasets.ImageFolder(test_dir, transform=val_test_transforms)

train_idx, val_idx = train_test_split(
    list(range(len(full_train_data))),
    test_size=0.2,
    random_state=42,
    stratify=full_train_data.targets
)

train_data = Subset(full_train_data, train_idx)
val_data = Subset(datasets.ImageFolder(train_dir, transform=val_test_transforms), val_idx)
```

```
train_loader = DataLoader(train_data, batch_size=32, shuffle=True)
val_loader = DataLoader(val_data, batch_size=32, shuffle=False)
test_loader = DataLoader(test_data, batch_size=32, shuffle=False)
```

4. Model Architecture and Configuration

```
model = models.resnet18(pretrained=True)
model.fc = nn.Linear(model.fc.in_features, 2)
model = model.to(device)

criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

Downloading: "<https://download.pytorch.org/models/resnet18-f37072fd.pth>" to /root/.cache/torch/hub/checkpoints/resnet18-f37072fd.pth [100%|██████████| 44.7M/44.7M [00:00<00:00, 162MB/s]]

5. Model Training and Validation

```
def train_and_evaluate(model, train_loader, val_loader, epochs=5): # 5 دورات
    for epoch in range(epochs):
        model.train()
        train_loss = 0.0
        train_correct = 0

        for images, labels in tqdm(train_loader, desc=f"Epoch {epoch+1}/{epochs} [Train]"):
            images, labels = images.to(device), labels.to(device)

            optimizer.zero_grad()
            outputs = model(images)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

            train_loss += loss.item()
            _, preds = torch.max(outputs, 1)
            train_correct += (preds == labels).sum().item()

        model.eval()
        val_loss = 0.0
        val_correct = 0
        with torch.no_grad():
            for images, labels in val_loader:
                images, labels = images.to(device), labels.to(device)
                outputs = model(images)
                loss = criterion(outputs, labels)
                val_loss += loss.item()
                _, preds = torch.max(outputs, 1)
                val_correct += (preds == labels).sum().item()

        avg_train_loss = train_loss / len(train_loader)
        avg_val_loss = val_loss / len(val_loader)
        train_acc = 100 * train_correct / len(train_data)
        val_acc = 100 * val_correct / len(val_data)

        print(f"Epoch {epoch+1}: Train Acc: {train_acc:.2f}% | Val Acc: {val_acc:.2f}%")
        print(f"Train Loss: {avg_train_loss:.4f} | Val Loss: {avg_val_loss:.4f}")
        print("-" * 30)
```

```
train_and_evaluate(model, train_loader, val_loader, epochs=5)
```

```
Epoch 1/5 [Train]: 100%|██████████| 14/14 [00:10<00:00, 1.31it/s]
Epoch 1: Train Acc: 81.12% | Val Acc: 61.61%
Train Loss: 0.4776 | Val Loss: 10.5494
-----
Epoch 2/5 [Train]: 100%|██████████| 14/14 [00:07<00:00, 2.00it/s]
Epoch 2: Train Acc: 88.54% | Val Acc: 58.04%
Train Loss: 0.3044 | Val Loss: 3.1784
-----
Epoch 3/5 [Train]: 100%|██████████| 14/14 [00:06<00:00, 2.03it/s]
Epoch 3: Train Acc: 89.21% | Val Acc: 78.57%
Train Loss: 0.2246 | Val Loss: 0.8596
-----
Epoch 4/5 [Train]: 100%|██████████| 14/14 [00:07<00:00, 1.99it/s]
Epoch 4: Train Acc: 97.08% | Val Acc: 77.68%
Train Loss: 0.0928 | Val Loss: 0.5327
-----
Epoch 5/5 [Train]: 100%|██████████| 14/14 [00:07<00:00, 1.98it/s]
Epoch 5: Train Acc: 96.40% | Val Acc: 82.14%
Train Loss: 0.0894 | Val Loss: 0.4450
```

6.Evaluation and Confusion Matrix

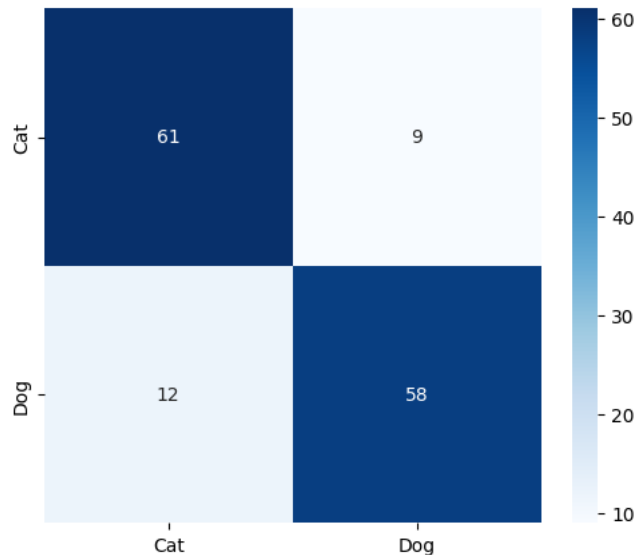
```
def final_test(model, loader):
    model.eval()
    y_true, y_pred = [], []
    with torch.no_grad():
        for images, labels in tqdm(loader, desc="Final Test"):
            images = images.to(device)
            outputs = model(images)
            _, preds = torch.max(outputs, 1)
            y_true.extend(labels.numpy())
            y_pred.extend(preds.cpu().numpy())

    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(6, 5))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Cat', 'Dog'], yticklabels=['Cat', 'Dog'])
    plt.show()

    print(classification_report(y_true, y_pred, target_names=['Cat', 'Dog']))

final_test(model, test_loader)
```

Final Test: 100% | 5/5 [00:02<00:00, 1.96it/s]



	precision	recall	f1-score	support
Cat	0.84	0.87	0.85	70
Dog	0.87	0.83	0.85	70
accuracy			0.85	140
macro avg	0.85	0.85	0.85	140
weighted avg	0.85	0.85	0.85	140