stanowski_homework

June 1, 2025

[ ]:
```python
from medmnist import VesselMNIST3D
from torch.utils.data import random_split
from torch.utils.data import DataLoader
from torch.utils.data import Subset
from torchvision.transforms import v2
import torch
from torch import nn
import numpy as np
import gdown
import os
import random
import matplotlib.pyplot as plt
import sys
repo_root = os.path.abspath(os.path.join(os.getcwd(), ".."))
sys.path.append(repo_root)

def take_middle_slice(inpt: np.ndarray):
    """
    NoduleMNIST 3D contains whole nodule volumes, however for this tutorial
    we will utilize only central slice of each example.
    We repeat this slice 3 times, as model expects input to have 3 channels.
    """
    inpt = inpt.squeeze()
    X, Y, Z = inpt.shape
    slice_ = inpt[:, :, Z//2]
    slice_ = torch.Tensor(slice_).unsqueeze(dim=0).repeat(3,1,1)
    return slice_

TRANSFORMS = v2.Compose([v2.Lambda(take_middle_slice),
                         v2.Resize(size=(224,224))
                         ])

data_dir = "./example_data"
os.makedirs(data_dir, exist_ok=True)


def download_weights(url, output_dir, filename):
    """
    Downloads weights from the given URL if they are not already downloaded.
    """
    os.makedirs(output_dir, exist_ok=True)
    output_path = os.path.join(output_dir, filename)

    if not os.path.exists(output_path):
        print(f"Downloading weights to {output_path}...")
        gdown.download(url, output_path)
    else:
        print(f"Weights already exist at {output_path}. Skipping download.")

url = "https://drive.google.com/uc?id=1xUevCbvII5yXDxVxb7bR65CPmgz2sGQA"
output_dir = "tuned_models"
filename = "lidc_dino_s8.pth"
download_weights(url, output_dir, filename)

CLASS_NAMES = ["benign", "malignant"]

LOGIT2NAME = {
    0: "benign",
    1: "malignant",
}

train_data = VesselMNIST3D(root=data_dir, split="train", size=64,␣
 ↪transform=TRANSFORMS, download=True)

zero_indices = [i for i, (_, label) in enumerate(train_data) if label == 0]
one_indices = [i for i, (_, label) in enumerate(train_data) if label == 1]

random.seed(42)
zero_indices_downsampled = random.sample(zero_indices, len(one_indices))

balanced_indices = zero_indices_downsampled + one_indices
random.shuffle(balanced_indices)

balanced_train_data = Subset(train_data, balanced_indices)
train_loader = DataLoader(balanced_train_data, batch_size=16, shuffle=True)

validation_data = VesselMNIST3D(root=data_dir, split="val", size=64,␣
 ↪transform=TRANSFORMS, download=True)
print("length of Train Datasets: ", len(train_data))
print("length of Validation Datasets: ", len(validation_data))


val_loader = DataLoader(validation_data, batch_size=16, shuffle = False)
```

```
test_dataset = VesselMNIST3D(root=data_dir, split="test", transform=TRANSFORMS)
test_loader = DataLoader(test_dataset, batch_size=16, shuffle=False)

CLASS_NAMES = ["benign", "malignant"]

LOGIT2NAME = {
    0: "benign",
    1: "malignant",
}
```

```
c:\Users\Michał\Documents\Studia\StudiaMagisterskie\Bioinformatyka\RokISemII\CBS
\medical_images
Weights already exist at tuned_models\lidc_dino_s8.pth. Skipping download.
length of Train Datasets:  1335
length of Validation Datasets:  191
```

[2]:
```
from transformers import ViTConfig, ViTModel

class DINO(nn.Module):
    """
    DINO Transformer model based on Huggingface implementation.
    """
    def __init__(self):
        super().__init__()
        # Backbone
        config = ViTConfig.from_pretrained('facebook/dino-vits8',␣
↪attn_implementation="eager") # We propose eager implementation to return att␣
↪scores gracefully.
        self.backbone = ViTModel(config)
        # Classfication head
        self.head = torch.nn.Linear(384, 1) # takes vector of length 384 and␣
↪outputs 1 number

    def forward(self, x: torch.Tensor, output_attentions:bool=False):
        out = self.backbone(x, output_attentions=output_attentions)
        x = out["pooler_output"]
        x = self.head(x)
        if output_attentions:
            att = out["attentions"]
            return x, att
        else:
            return x


DEVICE = "cuda" if torch.cuda.is_available() else "cpu"
WEIGHTS_PATH = "./tuned_models/lidc_dino_s8.pth"
```

```
MODEL = DINO()
MODEL.to(DEVICE)
```

```
[2]: DINO(
       (backbone): ViTModel(
         (embeddings): ViTEmbeddings(
           (patch_embeddings): ViTPatchEmbeddings(
             (projection): Conv2d(3, 384, kernel_size=(8, 8), stride=(8, 8))
           )
           (dropout): Dropout(p=0.0, inplace=False)
         )
         (encoder): ViTEncoder(
           (layer): ModuleList(
             (0-11): 12 x ViTLayer(
               (attention): ViTAttention(
                 (attention): ViTSelfAttention(
                   (query): Linear(in_features=384, out_features=384, bias=True)
                   (key): Linear(in_features=384, out_features=384, bias=True)
                   (value): Linear(in_features=384, out_features=384, bias=True)
                 )
                 (output): ViTSelfOutput(
                   (dense): Linear(in_features=384, out_features=384, bias=True)
                   (dropout): Dropout(p=0.0, inplace=False)
                 )
               )
               (intermediate): ViTIntermediate(
                 (dense): Linear(in_features=384, out_features=1536, bias=True)
                 (intermediate_act_fn): GELUActivation()
               )
               (output): ViTOutput(
                 (dense): Linear(in_features=1536, out_features=384, bias=True)
                 (dropout): Dropout(p=0.0, inplace=False)
               )
               (layernorm_before): LayerNorm((384,), eps=1e-12,
     elementwise_affine=True)
               (layernorm_after): LayerNorm((384,), eps=1e-12,
     elementwise_affine=True)
             )
           )
         )
         (layernorm): LayerNorm((384,), eps=1e-12, elementwise_affine=True)
         (pooler): ViTPooler(
           (dense): Linear(in_features=384, out_features=384, bias=True)
           (activation): Tanh()
         )
       )
       (head): Linear(in_features=384, out_features=1, bias=True)
)
```

[ ]:
```
import torch.nn.functional as F

class FocalLoss(nn.Module):
    def __init__(self, alpha=0.25, gamma=2.0, pos_weight=None):
        super().__init__()
        self.alpha = alpha
        self.gamma = gamma
        self.pos_weight = pos_weight

    def forward(self, logits, targets):
        BCE_loss = F.binary_cross_entropy_with_logits(logits, targets,␣
 ↪pos_weight=self.pos_weight, reduction='none')
        pt = torch.exp(-BCE_loss)  # pt = prob dla prawidłowej klasy
        focal_loss = self.alpha * (1 - pt) ** self.gamma * BCE_loss
        return focal_loss.mean()
```

[9]:
```
def set_seed(seed=42):
    random.seed(seed)
    np.random.seed(seed)
    torch.manual_seed(seed)
    torch.cuda.manual_seed_all(seed)
    torch.backends.cudnn.deterministic = True
    torch.backends.cudnn.benchmark = False

set_seed(17)

optimizer = torch.optim.Adam(MODEL.parameters(), lr=1e-4)
num_epochs = 10

all_labels = []
for _, labels in train_loader:
    all_labels.append(labels)

all_labels = torch.cat(all_labels).float().to(DEVICE)


num_pos = (all_labels == 1).sum().item()
num_neg = (all_labels == 0).sum().item()

pos_weight = torch.tensor(num_neg / num_pos).to(DEVICE)
criterion = FocalLoss(alpha=0.25, gamma=2.0, pos_weight=pos_weight)  # lub inne␣
 ↪wartości
print(f"num_pos: {num_pos}, num_neg: {num_neg}, pos_weight: {pos_weight.item():.
 ↪4f}")
val_accuracies = []
```

```
for epoch in range(num_epochs):
    MODEL.train()
    total_loss = 0

    for batch_idx, (imgs, labels) in enumerate(train_loader):
        imgs = imgs.to(DEVICE)
        labels = labels.to(DEVICE).float().squeeze(1)
        optimizer.zero_grad()
        logits = MODEL(imgs).squeeze(1)
        loss = criterion(logits, labels)
        loss.backward()
        optimizer.step()

        total_loss += loss.item()

        print("loss:", loss.item())

    avg_loss = total_loss / (batch_idx + 1)
    print(f"Epoch {epoch + 1}, Loss: {avg_loss:.4f}")

    MODEL.eval()
    correct = 0
    total = 0
    with torch.no_grad():
        for imgs, labels in val_loader:
            imgs = imgs.to(DEVICE)
            labels = labels.to(DEVICE).float()

            logits = MODEL(imgs)
            probs = torch.sigmoid(logits)
            preds = probs.round().squeeze().cpu().numpy()
            labels = labels.squeeze().cpu().numpy()
            correct += (preds == labels).sum()
            total += labels.shape[0]
            print("preds (val):", preds)
            print("labels (val):", labels)

    val_acc = 100 * correct / total
    val_accuracies.append(val_acc)
    print(f"Validation accuracy: {val_acc:.2f}%")

plt.figure(figsize=(8,5))
plt.plot(range(1, num_epochs+1), val_accuracies, marker='o')
plt.title("Validation accuracy per E]epoch")
plt.xlabel("Epoch")
plt.ylabel("Validation accuracy (%)")
```

```python
plt.grid(True)
plt.show()

MODEL.eval()
correct_test = 0
total_test = 0
with torch.no_grad():
    for imgs, labels in test_loader:
        imgs = imgs.to(DEVICE)
        labels = labels.to(DEVICE).float()

        logits = MODEL(imgs)
        probs = torch.sigmoid(logits)
        preds = probs.round().squeeze().cpu().numpy()
        labels = labels.squeeze().cpu().numpy()
        correct_test += (preds == labels).sum()
        total_test += labels.shape[0]

test_acc = 100 * correct_test / total_test
print(f"Test accuracy: {test_acc:.2f}%")
```

```
num_pos: 150, num_neg: 150, pos_weight: 1.0000
loss: 0.04300834238529205
loss: 0.04325180500745773
loss: 0.043579354882240295
loss: 0.04682915657758713
loss: 0.04417857900261879
loss: 0.04481332749128342
loss: 0.043162751942873
loss: 0.04279378801584244
loss: 0.04281925782561302
loss: 0.047194704413414
loss: 0.041718997061252594
loss: 0.042780302464962006
loss: 0.046555232256650925
loss: 0.043366312980651855
loss: 0.043800704181194305
loss: 0.04212180897593498
loss: 0.044897936284542084
loss: 0.04256697744131088
loss: 0.04594413563609123
Epoch 1, Loss: 0.0440
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [1. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

```
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Validation accuracy: 11.52%
loss: 0.042663101106882095
loss: 0.04313429445028305
loss: 0.04364151135087013
loss: 0.04230774939060211
loss: 0.04234901815652847
loss: 0.04329695552587509
loss: 0.0465267188847065
loss: 0.04273360222578049
loss: 0.043264277279376984
loss: 0.04409707337617874
loss: 0.04413458704948425
loss: 0.032267533242702484
loss: 0.040179282426834106
loss: 0.0447886660695076
loss: 0.051717180758714676
loss: 0.057219475507736206
loss: 0.04236173629760742
loss: 0.056350573897361755
loss: 0.041551925241947174
Epoch 2, Loss: 0.0445
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [1. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

```
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Validation accuracy: 11.52%
loss: 0.04443400725722313
loss: 0.04578990489244461
loss: 0.0428425632417202
loss: 0.04781711846590042
loss: 0.044237345457077026
loss: 0.044832367449998856
loss: 0.04306745529174805
loss: 0.04313982278108597
loss: 0.04310635104775429
loss: 0.04173729196190834
loss: 0.04819495975971222
loss: 0.048133525997400284
loss: 0.04408537223935127
loss: 0.04300447553396225
loss: 0.043381720781326294
loss: 0.04456016421318054
loss: 0.043213460594415665
loss: 0.04254582151770592
loss: 0.045935750007629395
Epoch 3, Loss: 0.0444
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [1. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

```
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Validation accuracy: 11.52%
loss: 0.04211811721324921
loss: 0.043328188359737396
loss: 0.04356037452816963
loss: 0.04347986727952957
loss: 0.042762838304042816
loss: 0.04204624146223068
loss: 0.0446857288479805
loss: 0.04428337514400482
loss: 0.042882274836301804
loss: 0.04117551073431969
loss: 0.04209674149751663
loss: 0.04849919676780701
loss: 0.04038695991039276
loss: 0.04613905027508736
loss: 0.04377277195453644
loss: 0.04281231760978699
loss: 0.04310618340969086
loss: 0.04201662540435791
loss: 0.04362444579601288
Epoch 4, Loss: 0.0433
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [1. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

```
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Validation accuracy: 11.52%
loss: 0.044930361211299896
loss: 0.04293310269713402
loss: 0.046754397451877594
loss: 0.041083864867687225
loss: 0.04126129299402237
loss: 0.04366737976670265
loss: 0.04388493299484253
loss: 0.04169062152504921
loss: 0.03885439783334732
loss: 0.03584957495331764
loss: 0.04315894842147827
loss: 0.04333731532096863
loss: 0.044141024351119995
loss: 0.04622920602560043
loss: 0.03870704397559166
loss: 0.04374990612268448
loss: 0.04215342551469803
loss: 0.042265020310878754
loss: 0.04607529938220978
Epoch 5, Loss: 0.0427
preds (val): [1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 0. 0. 0. 1. 1.]
labels (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
preds (val): [1. 1. 1. 0. 1. 1. 0. 1. 0. 0. 0. 0. 0. 1. 1. 1.]
labels (val): [1. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 1. 1. 1. 0. 0. 1. 0. 0. 0. 1. 0. 1. 0. 1. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 1. 1. 0. 1. 0. 1. 1. 1. 0. 0. 0. 0. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 0. 1. 0. 0. 0. 0. 1. 0. 1. 1. 0. 1. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 0. 0. 1. 0. 0. 0. 1. 1. 1. 0. 0. 1. 0. 1. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 1. 1. 0. 0. 0. 1. 1. 0. 0. 1. 0. 1. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 1. 0. 1. 1. 0. 0. 0. 1. 0. 0. 0. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 0. 1. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

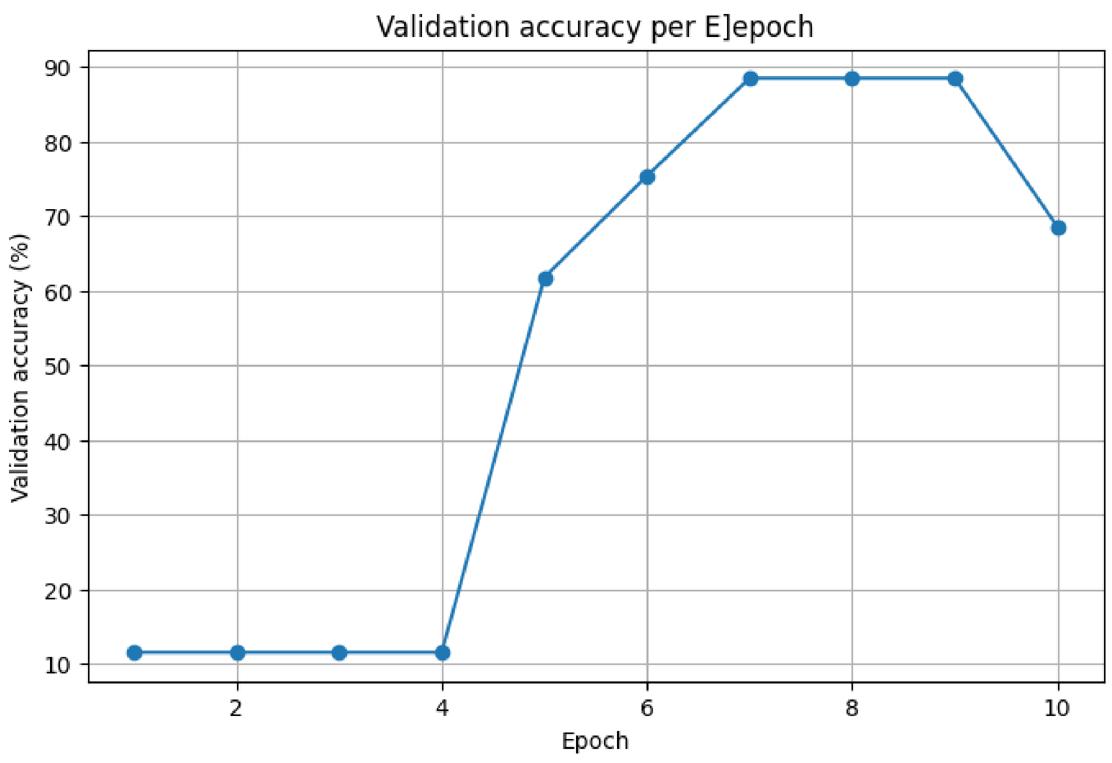
```
preds (val): [0. 1. 0. 1. 0. 0. 0. 1. 0. 0. 0. 1. 1. 1. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 1. 0. 1. 0. 1. 0. 1. 0. 0. 0. 0. 0. 1. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Validation accuracy: 61.78%
loss: 0.04168332740664482
loss: 0.040684863924980164
loss: 0.041919466108083725
loss: 0.041031043976545334
loss: 0.04248547926545143
loss: 0.046005524694919586
loss: 0.04096801578998566
loss: 0.040868014097213745
loss: 0.042971618473529816
loss: 0.04100576788187027
loss: 0.040030524134635925
loss: 0.04248647391796112
loss: 0.03814980387687683
loss: 0.0384003147482872
loss: 0.04930650070309639
loss: 0.05034059286117554
loss: 0.051574379205703735
loss: 0.043141745030879974
loss: 0.048935666680336
Epoch 6, Loss: 0.0433
preds (val): [0. 1. 1. 0. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 1. 1.]
labels (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
preds (val): [1. 1. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 0. 1. 1. 0.]
labels (val): [1. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 1. 1. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 1. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 1. 1. 0. 1. 0. 0. 1. 1. 0. 0. 0. 0. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 1. 0. 1. 0. 0. 0. 0. 1. 0. 1. 0. 0. 1. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 1. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
```

```
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 1. 0. 0. 0. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Validation accuracy: 75.39%
loss: 0.04697189852595329
loss: 0.04338392615318298
loss: 0.04256824776530266
loss: 0.04676556959748268
loss: 0.04251391440629959
loss: 0.04845169186592102
loss: 0.04611023887991905
loss: 0.044374383985996246
loss: 0.04333212226629257
loss: 0.041364140808582306
loss: 0.045427754521369934
loss: 0.041548047214746475
loss: 0.04274538904428482
loss: 0.048650242388248444
loss: 0.04727083072066307
loss: 0.038787540048360825
loss: 0.04398748651146889
loss: 0.04218759387731552
loss: 0.04253920540213585
Epoch 7, Loss: 0.0442
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [1. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

```
Validation accuracy: 88.48%
loss: 0.04272708296775818
loss: 0.04005105420947075
loss: 0.04390508681535721
loss: 0.04495500028133392
loss: 0.04138987511396408
loss: 0.04506964981555939
loss: 0.04330778494477272
loss: 0.04095372185111046
loss: 0.04532180353999138
loss: 0.04115685448050499
loss: 0.04739806428551674
loss: 0.04558210447430611
loss: 0.04269131273031235
loss: 0.04406913369894028
loss: 0.042142633348703384
loss: 0.04321584850549698
loss: 0.04350939020514488
loss: 0.0442655012011528
loss: 0.04028628021478653
Epoch 8, Loss: 0.0433
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [1. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Validation accuracy: 88.48%
loss: 0.045077163726091385
loss: 0.04143216088414192
```

```
loss: 0.04476833716034889
loss: 0.04255266487598419
loss: 0.044371314346790314
loss: 0.04248636215925217
loss: 0.04122951999306679
loss: 0.038899436593055725
loss: 0.04573369026184082
loss: 0.03941306471824646
loss: 0.045729510486125946
loss: 0.04701581969857216
loss: 0.04112890735268593
loss: 0.0434756763279438
loss: 0.04327689856290817
loss: 0.04029560834169388
loss: 0.04180140793323517
loss: 0.040570203214883804
loss: 0.04411071538925171
Epoch 9, Loss: 0.0428
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [1. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Validation accuracy: 88.48%
loss: 0.042507901787757874
loss: 0.0436214841902256
loss: 0.039300329983234406
loss: 0.04385992884635925
loss: 0.04245500639081001
```

```
loss: 0.047173723578453064
loss: 0.04441433027386665
loss: 0.040076982229948044
loss: 0.04662550240755081
loss: 0.04082127660512924
loss: 0.041431963443756104
loss: 0.0417076051235199
loss: 0.04288667440414429
loss: 0.04068044200539589
loss: 0.04218637943267822
loss: 0.04318384453654289
loss: 0.04171774908900261
loss: 0.0417044535279274
loss: 0.0393633209168911
Epoch 10, Loss: 0.0424
preds (val): [1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 0. 0. 0. 1. 1.]
labels (val): [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
preds (val): [1. 1. 0. 0. 1. 1. 0. 1. 0. 0. 0. 0. 0. 1. 1. 1.]
labels (val): [1. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 1. 1. 1. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 1. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 1. 1. 0. 1. 0. 1. 1. 1. 0. 0. 0. 0. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 1. 0. 1. 0. 0. 0. 0. 1. 0. 1. 0. 0. 1. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 1. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 0. 0. 0. 0. 0. 1. 1. 0. 0. 1. 0. 0. 1. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 1. 0. 1. 0. 0. 0. 0. 1. 0. 0. 0. 1. 1.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [1. 0. 1. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 1. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
preds (val): [0. 1. 0. 1. 0. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
labels (val): [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Validation accuracy: 68.59%
```

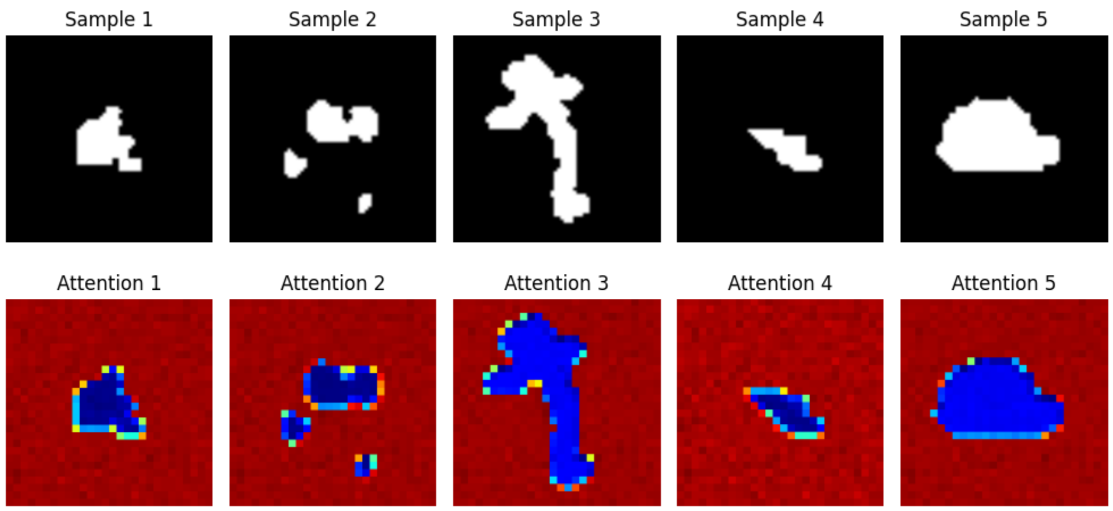


```
Test accuracy: 65.71%
```

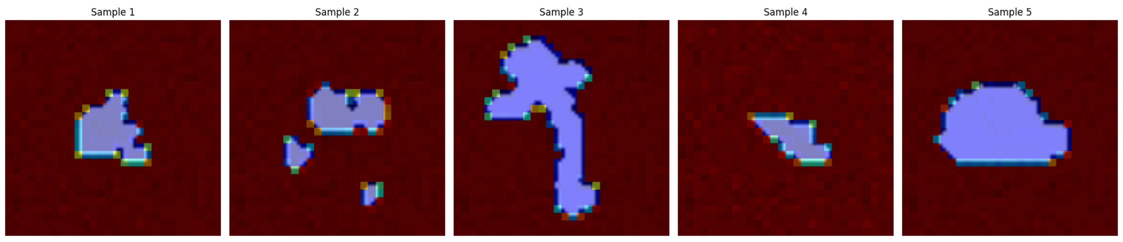
[11]:
```
#### When loading Obz, you may need to do:
## from obz.data_inspector.extractor import FirstOrderExtractor

# Setup OutlierDetector
from obzai.data_inspector.extractor import FirstOrderExtractor
from obzai.data_inspector.detector import GMMDetector
# Choose desired feature extractor. Chosen extractor will be used for␣
↪monitoring.
first_order_extrc = FirstOrderExtractor()

# Pass choosen extractor(s) to chosen OutlierDetector. Below we utilize outlier␣
↪detector based on Gaussian Mixture Models.
gmm_detector = GMMDetector(extractors=[first_order_extrc], n_components=3,␣
↪outlier_quantile=0.01)
# Call .fit() method with passed reference dataloader.
# Method will extract desired image features and fit outlier detection model␣
↪(in that case GMM).
gmm_detector.fit(val_loader)
```

[13]:
```
# Setup XAI Tools
## from obz.xai_tool import ...
from obzai.xai.xai_tool import CDAM, AttentionMap

# Choose desired XAI Tools
cdam_tool = CDAM(model=MODEL,
                 mode='vanilla',                      # CDAM mode
                 gradient_type="from_logits",  # Whether backpropagate␣
 ↪gradients from logits or probabilities.
                 gradient_reduction="average",        # Gradient reduction␣
 ↪method.
                 activation_type="sigmoid")           # Activation function␣
 ↪applied on logits. (Needed when gradients are backpropagated from␣
 ↪probabilities.)
# In CDAM you need to specify on which layer you want to create hooks.
cdam_tool.create_hooks(layer_name="backbone.encoder.layer.11.layernorm_before")

attention_tool = AttentionMap(model=MODEL,
                              attention_layer_id = -1,# ID of an attention␣
 ↪layer from which to extract attention weights
                              head = None             # ID of attention head to␣
 ↪choose. If None, attention scores are averaged.
                              )
```

[14]:
```
samples, labels = next(iter(val_loader))
attention_maps = attention_tool.explain(samples)

# Visualize samples and attention maps
fig, axes = plt.subplots(2, 5, figsize=(10, 5))

# First row: Original samples
for i in range(5):
    original_image = samples[i].permute(1, 2, 0).cpu().numpy()  # Convert␣
 ↪tensor to numpy array and rearrange dimensions
    axes[0, i].imshow(original_image, cmap='gray')
    axes[0, i].set_title(f"Sample {i + 1}")
    axes[0, i].axis('off')

# Second row: Attention maps
for i in range(5):
    attention_map = attention_maps[i].cpu().numpy()
    axes[1, i].imshow(attention_map, cmap='jet')
    axes[1, i].set_title(f"Attention {i + 1}")
    axes[1, i].axis('off')

plt.tight_layout()
plt.show()
```

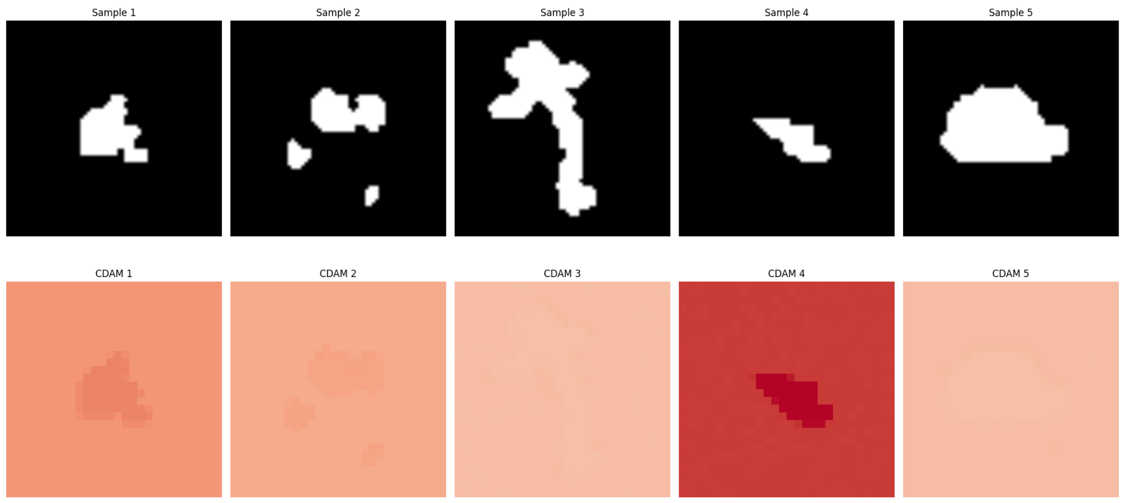


[ ]:
```
fig, axes = plt.subplots(1, 5, figsize=(20, 5))

for i in range(5):
    original_image = samples[i].permute(1, 2, 0).cpu().numpy()  # Convert␣
↪tensor to numpy array and rearrange dimensions
    attention_map = attention_maps[i].cpu().numpy()  # Convert attention map to␣
↪numpy array

    axes[i].imshow(original_image, cmap='gray')
    axes[i].imshow(attention_map, cmap='jet', alpha=0.5)  # Use alpha for␣
↪transparency
    axes[i].set_title(f"Sample {i + 1}")
    axes[i].axis('off')

plt.tight_layout()
plt.show()
```



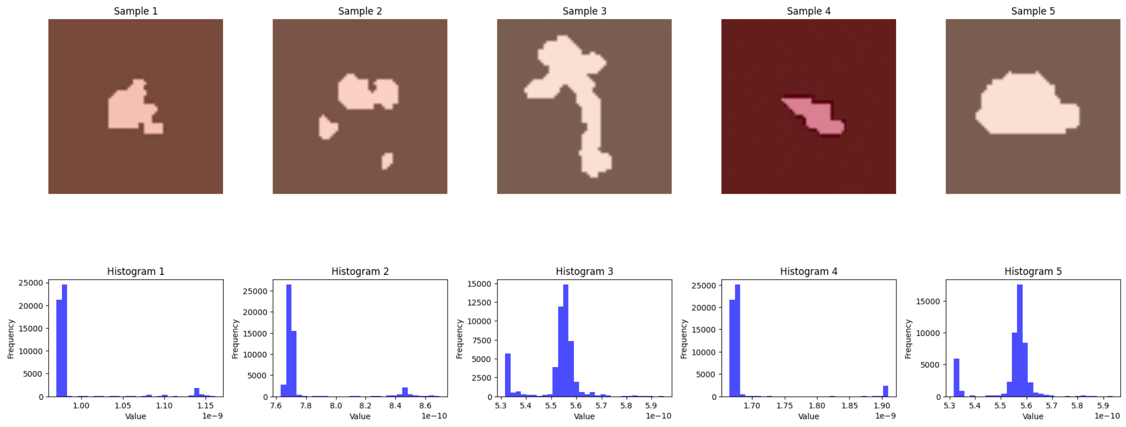
[18]:
```
cdam_maps = cdam_tool.explain(samples, target_idx=[0]*len(samples))
```

```
[ ]: fig, axes = plt.subplots(2, 5, figsize=(20, 10))

     for i in range(5):
         original_image = samples[i].permute(1, 2, 0).cpu().numpy()  # Convert␣
      ↪tensor to numpy array and rearrange dimensions
         axes[0, i].imshow(original_image, cmap='gray')
         axes[0, i].set_title(f"Sample {i + 1}")
         axes[0, i].axis('off')

     for i in range(5):
         cdam_map = cdam_maps[i].squeeze().cpu().numpy()  # Convert CDAM map to␣
      ↪numpy array
         axes[1, i].imshow(cdam_map, cmap='coolwarm', vmin=-cdam_maps.abs().max(),␣
      ↪vmax=cdam_maps.abs().max())  # Diverging colormap
         axes[1, i].set_title(f"CDAM {i + 1}")
         axes[1, i].axis('off')

     plt.tight_layout()
     plt.show()
```



```
[ ]: fig, axes = plt.subplots(2, 5, figsize=(20, 10), gridspec_kw={'height_ratios':␣
      ↪[4, 1]})

     for i in range(5):
         original_image = samples[i].permute(1, 2, 0).cpu().numpy()
         cdam_map = cdam_maps[i].squeeze().cpu().numpy()

         im = axes[0, i].imshow(original_image, cmap='gray')
```

```
    im = axes[0, i].imshow(cdam_map, cmap='coolwarm', alpha=0.5,␣
↪vmin=-cdam_maps.abs().max(), vmax=cdam_maps.abs().max())  # Use alpha for␣
↪transparency
    axes[0, i].set_title(f"Sample {i + 1}")
    axes[0, i].axis('off')

for i in range(5):
    cdam_map = cdam_maps[i].squeeze().cpu().numpy()
    axes[1, i].hist(cdam_map.ravel(), bins=30, color='blue', alpha=0.7)
    axes[1, i].set_title(f"Histogram {i + 1}")
    axes[1, i].set_xlabel('Value')
    axes[1, i].set_ylabel('Frequency')

plt.tight_layout()
plt.show()
```


Sample 1
Sample 2
Sample 3
Sample 4
Sample 5
Histogram 1
Histogram 2
Histogram 3
Histogram 4
Histogram 5
Frequency
Value
1e−9
1e−10
1e−10
1e−9
1e−10