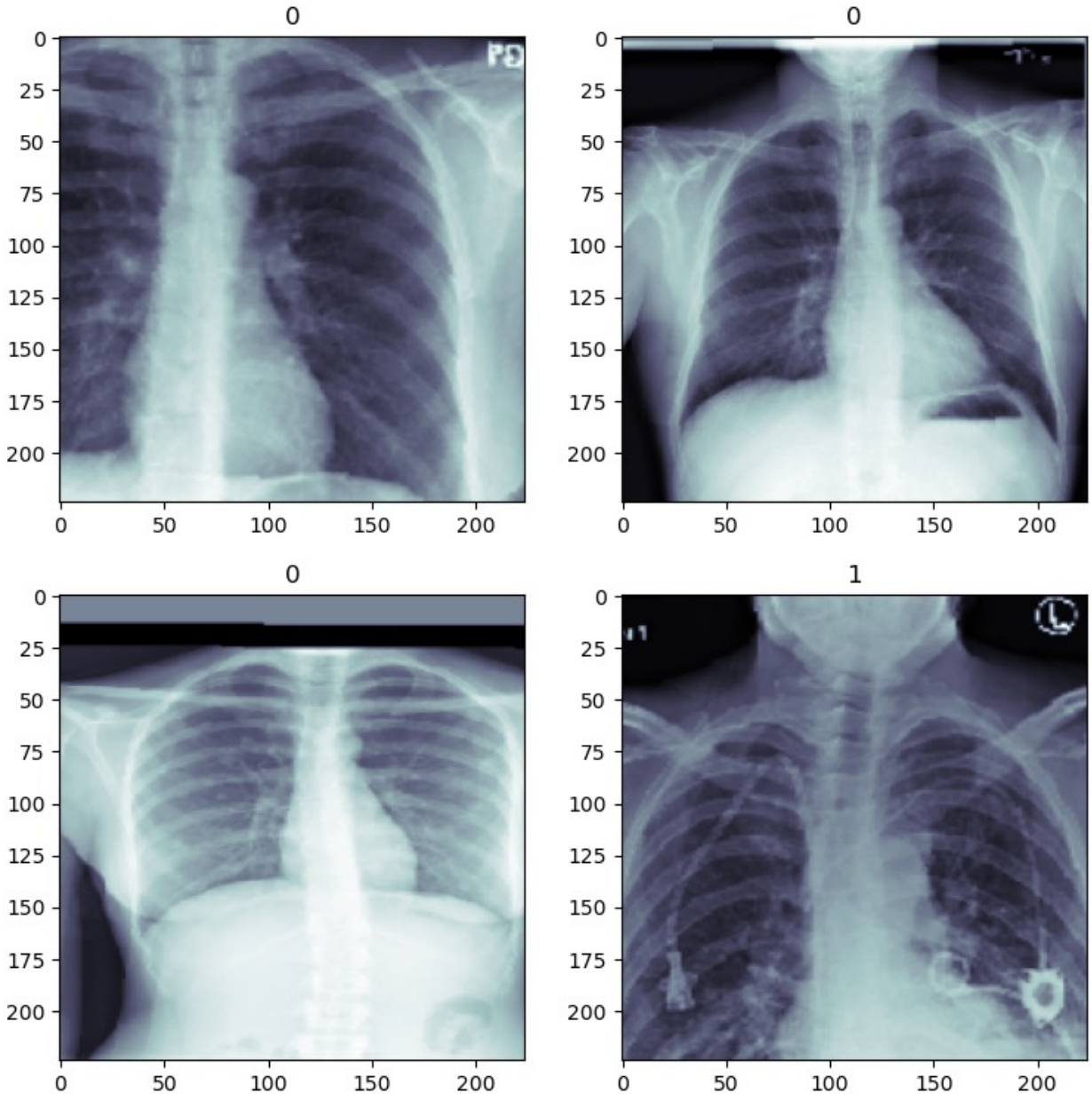
```
# pip install torch torchvision torchmetrics pytorch-lightning tqdm
numpy matplotlib tensorboard tensorboardX

import torch
import torchvision
from torchvision import transforms
import torchmetrics
import pytorch_lightning as pl
from pytorch_lightning.callbacks import ModelCheckpoint
from pytorch_lightning.loggers import TensorBoardLogger
from tqdm.notebook import tqdm
import numpy as np
import matplotlib.pyplot as plt

# Load dataset

def load_file(path):
    return np.load(path).astype(np.float32)

train_transforms = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize(0.49, 0.248),
    transforms.RandomAffine(degrees=(-5, 5), translate=(0, 0.05),
scale=(0.9, 1.1)),
    transforms.RandomResizedCrop((224,224), scale=(0.35,1))
])

val_transforms = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize(0.49, 0.248),
])

train_dataset = torchvision.datasets.DatasetFolder("processed/train/",
loader=load_file, extensions="npy", transform=train_transforms)
val_dataset = torchvision.datasets.DatasetFolder("processed/val/",
loader=load_file, extensions="npy", transform=val_transforms)

fig, axis = plt.subplots(2, 2, figsize=(9,9))
for i in range(2):
    for j in range(2):
        random_index = np.random.randint(0,24000)
        x_ray, label = train_dataset[random_index]
        axis[i][j].imshow(x_ray[0], cmap="bone")
        axis[i][j].set_title(label)
```




```
batch_size = 64
num_workers = 4

train_loader = torch.utils.data.DataLoader(train_dataset,
batch_size=batch_size, num_workers=num_workers, shuffle=True)
val_loader = torch.utils.data.DataLoader(val_dataset,
batch_size=batch_size, num_workers=num_workers, shuffle=False)

np.unique(train_dataset.targets, return_counts=True) #there are aprox
three times more cases without Phneumonia than cases with Phneumonia,
this is an unbalaced dataset
```

```
(array([0, 1]), array([18593,  5407]))
```

```
torchvision.models.resnet18() #see the structure of the previously
trained model i will adapt
```

```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2),
padding=(3, 3), bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1,
ceil_mode=False)
  (layer1): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
    (1): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer2): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2),
```

```
bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer3): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2),
bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer4): Sequential(
    (0): BasicBlock(
```

```
      (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2),
bias=False)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Linear(in_features=512, out_features=1000, bias=True)
)

# Train model

class PneumoniaModel(pl.LightningModule):
    def __init__(self):
        super().__init__()
        self.model = torchvision.models.resnet18()
        self.model.conv1 = torch.nn.Conv2d(1, 64, kernel_size=(7, 7),
stride=(2, 2), padding=(3, 3), bias=False)
        self.model.fc = torch.nn.Linear(in_features=512,
out_features=1, bias=True)

        self.optimizer = torch.optim.Adam(self.model.parameters(),
lr=1e-4)
        self.loss_fn =
torch.nn.BCEWithLogitsLoss(pos_weight=torch.tensor([3]))

        self.train_acc = torchmetrics.Accuracy(task="binary")
        self.val_acc = torchmetrics.Accuracy(task="binary")
```

```
    def forward(self, data):
        pred = self.model(data)
        return pred

    def training_step(self, batch, batch_idx):
        x_ray, label = batch
        label = label.float()
        pred = self(x_ray)[:,0] # forward
        loss = self.loss_fn(pred, label)

        self.log("Train Logs", loss)
        self.log("Step Train ACC", self.train_acc(torch.sigmoid(pred),
label.int()))

        return loss

    def on_train_epoch_end(self):
        self.log("Train ACC", self.train_acc.compute())

    def validation_step(self, batch, batch_idx):
        x_ray, label = batch
        label = label.float()
        pred = self(x_ray)[:,0] # forward
        loss = self.loss_fn(pred, label)

        self.log("Val Logs", loss)
        val_acc = self.val_acc(torch.sigmoid(pred), label.int())
        self.log("Val ACC", val_acc, prog_bar=True, on_epoch=True)
        return loss

    def on_val_epoch_end(self):
        self.log("Val ACC", self.val_acc.compute())

    def configure_optimizers(self):
        return [self.optimizer]

model = PneumoniaModel()

checkpoint_callback = ModelCheckpoint(
    monitor= "Val ACC",
    save_top_k = 10,
    mode = "max",)

trainer = pl.Trainer(
    accelerator="gpu",
    devices=1,
    logger=TensorBoardLogger(save_dir="./logs"),
    log_every_n_steps=1,
    callbacks=checkpoint_callback,
```

```
    max_epochs=35
)

GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
HPU available: False, using: 0 HPUs

trainer.fit(model, train_loader, val_loader)

LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name      | Type               | Params | Mode
-------------------------------------------------------
0 | model     | ResNet             | 11.2 M | train
1 | loss_fn   | BCEWithLogitsLoss | 0      | train
2 | train_acc | BinaryAccuracy     | 0      | train
3 | val_acc   | BinaryAccuracy     | 0      | train
-------------------------------------------------------
11.2 M    Trainable params
0         Non-trainable params
11.2 M    Total params
44.683    Total estimated model params size (MB)
71        Modules in train mode
0         Modules in eval mode

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"bc05d77fbf95461ba30d16ef9dfe7c79","version_major":2,"vers
ion_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}
```

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

{"model_id":"","version_major":2,"version_minor":0}

`Trainer.fit` stopped: `max_epochs=35` reached.

```
device = torch.device("cuda:0" if torch.cuda.is_available() else
"cpu")

ckpt_path = "./logs/lightning_logs/version_6/checkpoints/epoch=28-
step=10875.ckpt"
model = PneumoniaModel.load_from_checkpoint(ckpt_path)
model.eval().to(device)
```

```
PneumoniaModel(
  (model): ResNet(
    (conv1): Conv2d(1, 64, kernel_size=(7, 7), stride=(2, 2),
padding=(3, 3), bias=False)
    (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (relu): ReLU(inplace=True)
    (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1,
dilation=1, ceil_mode=False)
    (layer1): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
      (1): BasicBlock(
        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (layer2): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (downsample): Sequential(
          (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2),
bias=False)
          (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        )
```

```
      )
      (1): BasicBlock(
        (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (layer3): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (downsample): Sequential(
          (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2),
bias=False)
          (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        )
      )
      (1): BasicBlock(
        (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (layer4): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
```

```
        (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (downsample): Sequential(
          (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2),
bias=False)
          (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        )
      )
      (1): BasicBlock(
        (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
    (fc): Linear(in_features=512, out_features=1, bias=True)
  )
  (loss_fn): BCEWithLogitsLoss()
  (train_acc): BinaryAccuracy()
  (val_acc): BinaryAccuracy()
)

preds = []
labels = []

with torch.no_grad():
    for data, label in tqdm(val_dataset):
        data = data.to(device).float().unsqueeze(0)
        pred = torch.sigmoid(model(data)[0].cpu())
        preds.append(pred)
        labels.append(label)
preds = torch.tensor(preds)
labels = torch.tensor(labels).int()

{"model_id":"de76db55ae974a7a9dfd99b246e198db","version_major":2,"vers
ion_minor":0}

import pandas as pd
from torchmetrics import Accuracy, Precision, Recall, ConfusionMatrix

acc_metric       = Accuracy(task="binary")
```

```
precision_metric = Precision(task="binary")
recall_metric    = Recall(task="binary")
cm_metric        = ConfusionMatrix(task="binary", num_classes=2)

acc       = acc_metric(preds, labels).item()
precision = precision_metric(preds, labels).item()
recall    = recall_metric(preds, labels).item()
cm        = cm_metric(preds, labels).numpy()

df = pd.DataFrame({
    "Accuracy":       [acc],
    "Precision":      [precision],
    "Recall":         [recall],
    "Confusion Mat.": [cm.tolist()]
})

df
```

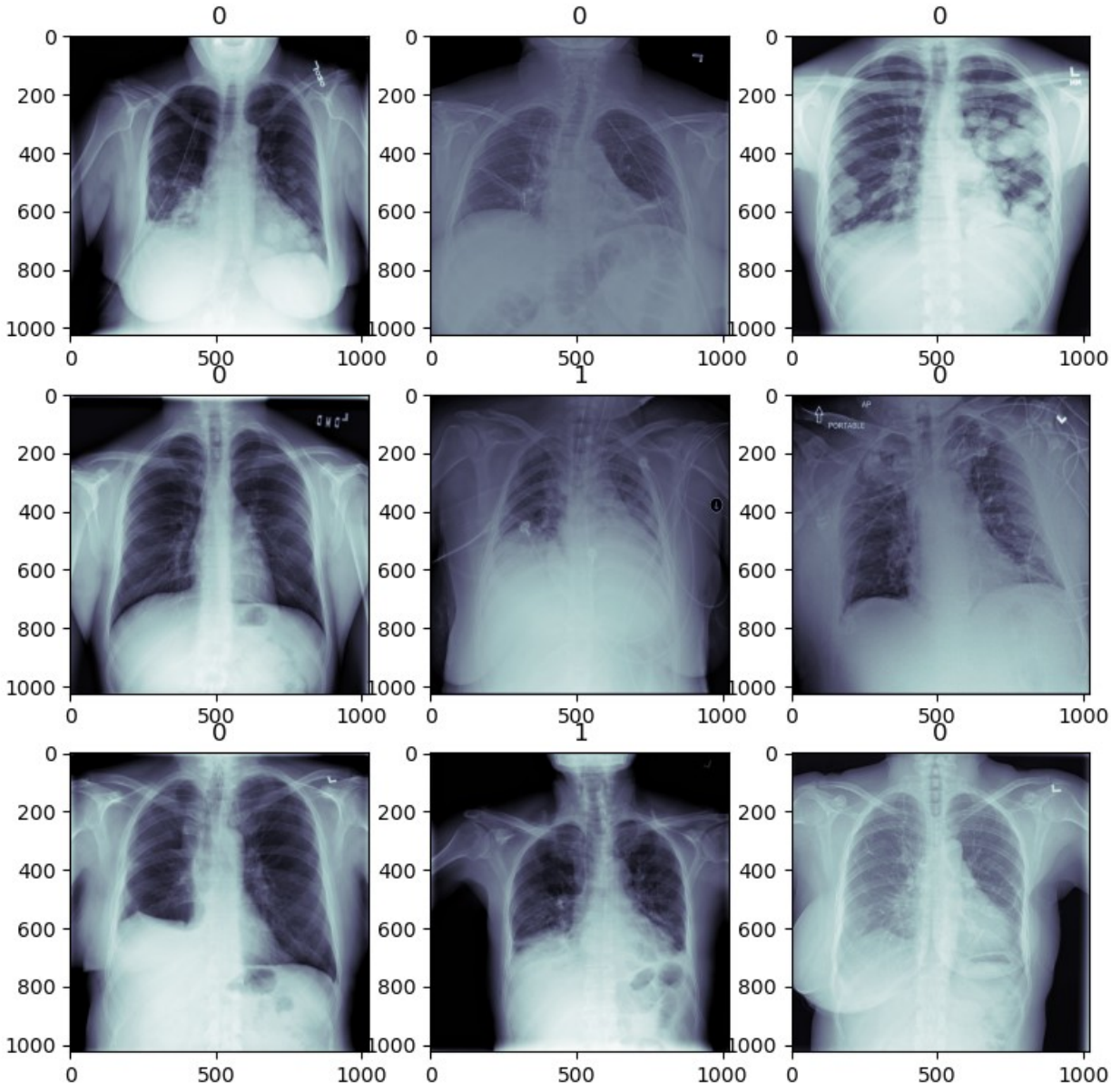
	Accuracy	Precision	Recall	Confusion Mat.
0	0.805514	0.544768	0.834711	[[1657, 422], [100, 505]]

```
# https://www.kaggle.com/competitions/rsna-pneumonia-detection-challenge

import pathlib
import pydicom
import numpy as np
import cv2
import pandas as pd
import matplotlib.pyplot as plt
from tqdm.notebook import tqdm
import os

labels =
pd.read_csv("./rsna-pneumonia-detection-challenge/stage_2_train_labels.csv")
labels.head(6)
```

	patientId	x	y	width	height	Target
0	0004cfab-14fd-4e49-80ba-63a80b6bddd6	NaN	NaN	NaN	NaN	0
1	00313ee0-9eaa-42f4-b0ab-c148ed3241cd	NaN	NaN	NaN	NaN	0
2	00322d4d-1c29-4943-afc9-b6754be640eb	NaN	NaN	NaN	NaN	0
3	003d8fa0-6bf1-40ed-b54c-ac657f8495c5	NaN	NaN	NaN	NaN	0
4	00436515-870c-4b36-a041-de91049b9ab4	264.0	152.0	213.0	379.0	1
5	00436515-870c-4b36-a041-de91049b9ab4	562.0	152.0	256.0	453.0	1

```
labels = labels.drop_duplicates("patientId")

ROOT_PATH =
pathlib.Path("./rsna-pneumonia-detection-challenge/stage_2_train_images/")
SAVE_PATH = pathlib.Path("./processed")

fig, axis = plt.subplots(3, 3, figsize=(9, 9))
c = 0

for i in range(3):
    for j in range(3):
        patient_id = labels.patientId.iloc[c]
        dcm_path = ROOT_PATH/patient_id
        dcm_path = dcm_path.with_suffix(".dcm")
        dcm = pydicom.dcmread(dcm_path).pixel_array
        label = labels["Target"].iloc[c]
        axis[i][j].imshow(dcm, cmap="bone")
        axis[i][j].set_title(label)
```

```
        c+=1
```




```
sums, sums_squared = 0, 0

for c, patient_id in enumerate(tqdm(labels.patientId)):
    patient_id = labels.patientId.iloc[c]
    dcm_path = ROOT_PATH/patient_id
    dcm_path = dcm_path.with_suffix(".dcm")
    dcm = pydicom.dcmread(dcm_path).pixel_array / 255
    dcm_array = cv2.resize(dcm, (224,224)).astype(np.float16)
```

```
    label = labels.Target.iloc[c]
    train_or_val = "train" if c < 24000 else "val"
    current_save_path = SAVE_PATH/train_or_val/str(label)
    current_save_path.mkdir(parents=True, exist_ok=True)
    np.save(current_save_path/patient_id, dcm_array)
    normalizer = 224*224
    if train_or_val == "train":
        sums += np.sum(dcm_array) / normalizer
        sums_squared += (dcm_array ** 2).sum() / normalizer
```

{"model_id":"4db04e85af3646d1ad7af2640238b35e","version_major":2,"version_minor":0}

```
mean = sums / 24000
std = np.sqrt((sums_squared / 24000) - mean**2)
print(mean, std)
```

```
0.4903962485384803 0.24795070634161256
```