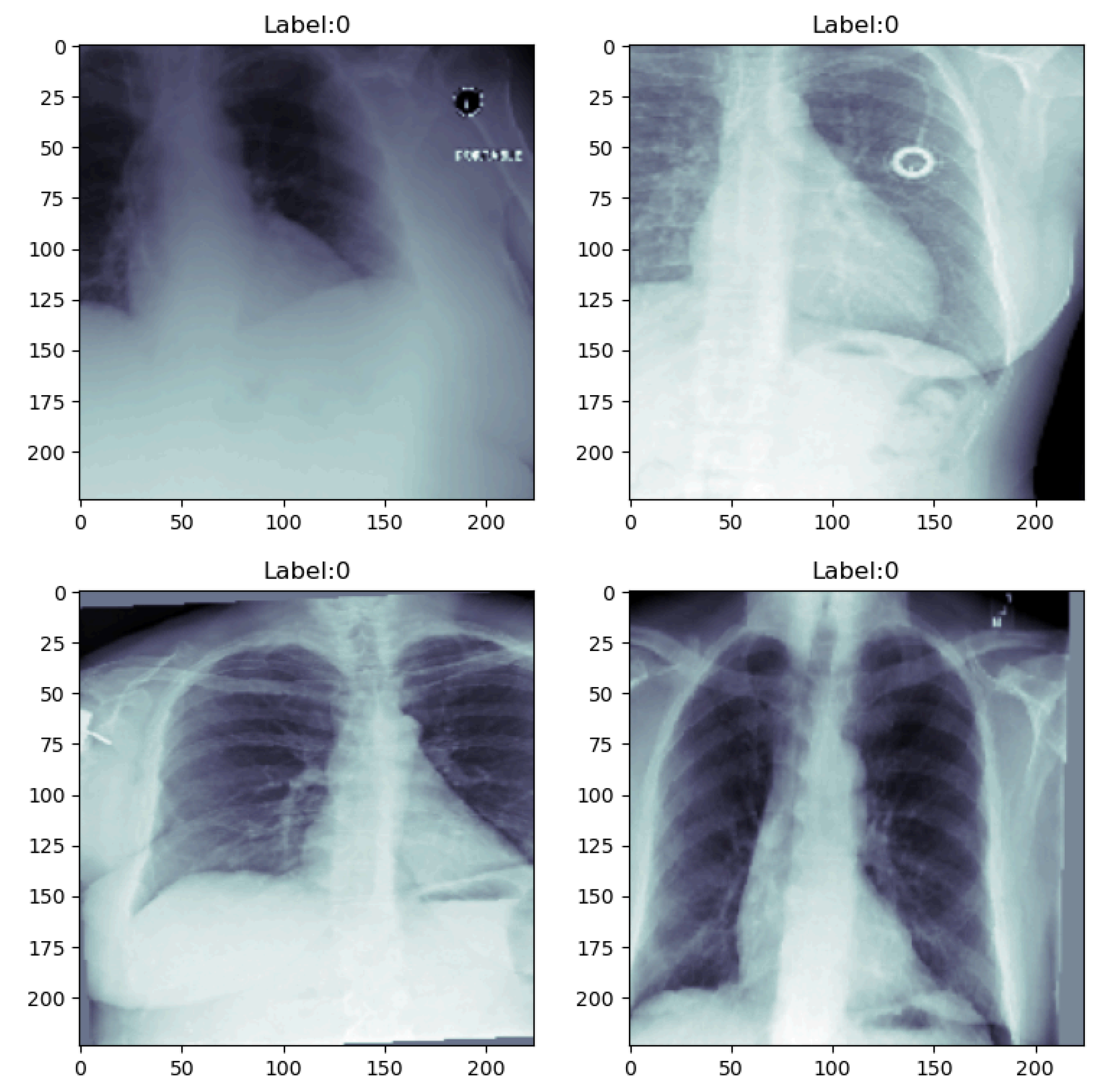
In [1]:
```python
import torch
import torchvision
from torchvision import transforms
import torchmetrics
import pytorch_lightning as pl
from pytorch_lightning.callbacks import ModelCheckpoint
from pytorch_lightning.loggers import TensorBoardLogger
from tqdm.notebook import tqdm
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```python
def load_file(path):
    return np.load(path).astype(np.float32)
```

In [3]:
```python
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]),
])
```

In [13]:
```python
train_dataset = torchvision.datasets.DatasetFolder(
    r"C:\Users\kisha\OneDrive\Desktop\Projects\MedAI\AI-IN-MEDICAL-MATERIALS\04-Pneumonia-ClassificationProcessed/train/",
    loader=load_file, extensions="npy", transform=train_transforms)

val_dataset = torchvision.datasets.DatasetFolder(
    r"C:\Users\kisha\OneDrive\Desktop\Projects\MedAI\AI-IN-MEDICAL-MATERIALS\04-Pneumonia-ClassificationProcessed/val/",
    loader=load_file, extensions="npy", transform=val_transforms)
```

In [30]:
```python
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, 20000)
        x_ray, label = train_dataset[random_index]
        axis[i][j].imshow(x_ray[0], cmap="bone")
        axis[i][j].set_title(f"Label:{label}")
```



In [6]:
```python
batch_size = 64
num_workers = 0

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)

print(f"There are {len(train_dataset)} train images and {len(val_dataset)} val images")
```

```
There are 24000 train images and 2684 val images
```

In [7]:
```python
np.unique(train_dataset.targets, return_counts=True), np.unique(val_dataset.targets, return_counts=True)
```

Out[7]:
```
((array([0, 1]), array([18593,  5407], dtype=int64)),
 (array([0, 1]), array([2079,  605], dtype=int64)))
```

In [8]:
```python
class PneumoniaModel(pl.LightningModule):
    def __init__(self, weight=1):
        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)

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

        # simple accuracy computation
        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()  # Convert label to float (just needed for loss computation)
        pred = self(x_ray)[:,0]  # Prediction: Make sure prediction and label have same shape
        #while using pytorch lightening we don't need to write self.forward
        #our prediction are of shape (batchsize,1) but our lebels are only of shape batchsize
        #hence remove second dimension from prediction by slicing
        loss = self.loss_fn(pred, label)  # Compute the loss

        # Log loss and batch accuracy
        self.log("Train Loss", loss)
        self.log("Step Train Acc", self.train_acc(torch.sigmoid(pred), label.int()))
        return loss


    def on_train_epoch_end(self):
        # After one epoch compute the whole train_data accuracy
        self.log("Train Acc", self.train_acc.compute())


    def validation_step(self, batch, batch_idx):
        # Same steps as in the training_step
        x_ray, label = batch
        label = label.float()
        pred = self(x_ray)[:,0]  # to make sure prediction and label have same shape

        loss = self.loss_fn(pred, label)

        # Log validation metrics
        self.log("Val Loss", loss)
        self.log("Step Val Acc", self.val_acc(torch.sigmoid(pred), label.int()))
        return loss

    def on_validation_epoch_end(self):
        self.log("Val Acc", self.val_acc.compute())

    def configure_optimizers(self):
        #Caution! You always need to return a list here (just pack your optimizer into one :))
        return [self.optimizer]
```

In [9]:
```python
model = PneumoniaModel()
```

In [10]:
```python
# Create the checkpoint callback
checkpoint_callback = ModelCheckpoint(
    monitor='Val Acc',
    save_top_k=10,
    mode='max')
```

In [11]:
```python
# Create the trainer
# Change the gpus parameter to the number of available gpus on your system. Use 0 for CPU training

#here provide whole path to your logs directory
trainer = pl.Trainer(devices=[0], accelerator="gpu",logger=TensorBoardLogger(save_dir=r"C:\Users\kisha\OneDrive\Desktop\Projects\MedAI\AI-IN-MEDICAL-MATERIALS\04-Pneumonia-Classification/logs"), log_every_n_steps=1,
                     callbacks=checkpoint_callback,
                     max_epochs=15)
```

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

In [12]:
```python
trainer.fit(model, train_loader, val_loader)
```

```
You are using a CUDA device ('NVIDIA GeForce RTX 3050 Laptop GPU') that has Tensor Cores. To properly utilize them, you should set `torch.set_float32_matmul_precision('medium' | 'high')` which will trade-off precision for performance. For more details, read https://pytorch.org/docs/stable/generated/torch.set_float32_matmul_precision.html#torch.set_float32_matmul_precision
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name      | Type              | Params
------------------------------------------------
0 | model     | ResNet            | 11.2 M
1 | loss_fn   | BCEWithLogitsLoss | 0
2 | train_acc | BinaryAccuracy    | 0
3 | val_acc   | BinaryAccuracy    | 0
------------------------------------------------
11.2 M    Trainable params
0         Non-trainable params
11.2 M    Total params
44.683    Total estimated model params size (MB)
Sanity Checking: |                                                                        | 0/? [00:00<?…
C:\Users\kisha\anaconda3\envs\kishan\lib\site-packages\pytorch_lightning\trainer\connectors\data_connector.py:441: The 'val_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of the `num_workers` argument` to `num_workers=15` in the `DataLoader` to improve performance.
C:\Users\kisha\anaconda3\envs\kishan\lib\site-packages\pytorch_lightning\trainer\connectors\data_connector.py:441: The 'train_dataloader' does not have many workers which may be a bottleneck. Consider increasing the value of the `num_workers` argument` to `num_workers=15` in the `DataLoader` to improve performance.
Training: |                                                                               | 0/? [00:00<?…
Validation: |                                                                             | 0/? [00:00<?…
Validation: |                                                                             | 0/? [00:00<?…
Validation: |                                                                             | 0/? [00:00<?…
Validation: |                                                                             | 0/? [00:00<?…
Validation: |                                                                             | 0/? [00:00<?…
Validation: |                                                                             | 0/? [00:00<?…
Validation: |                                                                             | 0/? [00:00<?…
Validation: |                                                                             | 0/? [00:00<?…
Validation: |                                                                             | 0/? [00:00<?…
Validation: |                                                                             | 0/? [00:00<?…
Validation: |                                                                             | 0/? [00:00<?…
Validation: |                                                                             | 0/? [00:00<?…
Validation: |                                                                             | 0/? [00:00<?…
Validation: |                                                                             | 0/? [00:00<?…
Validation: |                                                                             | 0/? [00:00<?…
`Trainer.fit` stopped: `max_epochs=15` reached.
```

In [14]:
```python
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
#We evaluate to calculate accuracy,precision,recall and confusion matrix
# Use strict=False, otherwise we would want to match the pos_weight which is not necessary
#write path correctly
model = PneumoniaModel.load_from_checkpoint(r"C:\Users\kisha\OneDrive\Desktop\Projects\MedAI\AI-IN-MEDICAL-MATERIALS\04-Pneumonia-Classification\weights\weights_3.ckpt")
model.eval()
model.to(device);
```

```
C:\Users\kisha\anaconda3\envs\kishan\lib\site-packages\pytorch_lightning\utilities\migration\migration.py:208: You have multiple `ModelCheckpoint` callback states in this checkpoint, but we found state keys that would end up colliding with each other after an upgrade, which means we can't differentiate which of your checkpoint callbacks needs which states. At least one of your `ModelCheckpoint` callbacks will not be able to reload the state.
Lightning automatically upgraded your loaded checkpoint from v1.3.4 to v2.2.1. To apply the upgrade to your files permanently, run `python -m pytorch_lightning.utilities.upgrade_checkpoint C:\Users\kisha\OneDrive\Desktop\Projects\MedAI\AI-IN-MEDICAL-MATERIALS\04-Pneumonia-Classification\weights\weights_3.ckpt`
```

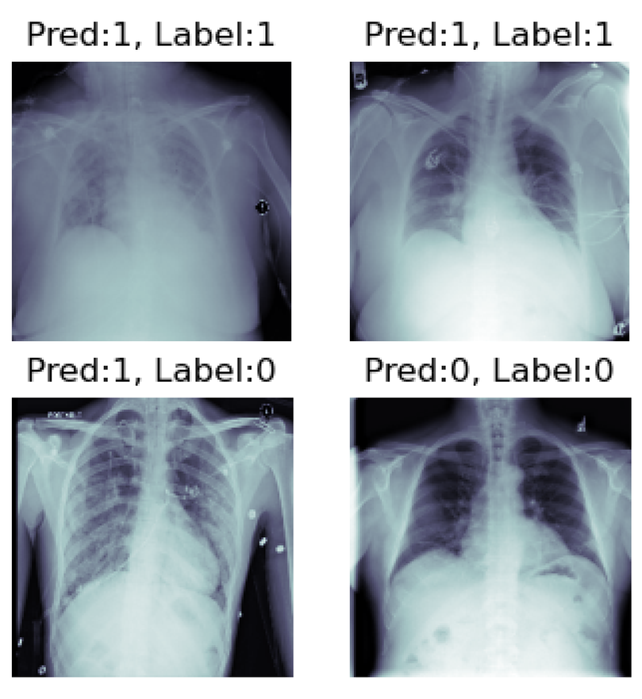
In [15]:
```python
preds = []
labels = []

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()
```

```
 0%|          | 0/2684 [00:00<?, ?it/s]
```

In [20]:
```python
acc = torchmetrics.Accuracy(task="binary")(preds, labels)
precision = torchmetrics.Precision(task="binary")(preds, labels)
recall = torchmetrics.Recall(task="binary")(preds, labels)
cm = torchmetrics.ConfusionMatrix(num_classes=2,task="binary")(preds, labels)
cm_threshed = torchmetrics.ConfusionMatrix(num_classes=2, threshold=0.25,task="binary")(preds, labels)

print(f"Val Accuracy: {acc}")
print(f"Val Precision: {precision}")
print(f"Val Recall: {recall}")
print(f"Confusion Matrix:\n {cm}")
print(f"Confusion Matrix 2:\n {cm_threshed}")
#recall is much larger,this means our model rarely misses a case of pneumonia
#however low precision means lots of False positive

#we can rerun the complete evaluation with the model which was not trained with weighted loss
#this time it will fails to detect many amount of pneumonias
```

```
Val Accuracy: 0.7652757167816162
Val Precision: 0.48819640278816223
Val Recall: 0.8545454740524292
Confusion Matrix:
 tensor([[1537,  542],
        [  88,  517]])
Confusion Matrix 2:
 tensor([[1135,  944],
        [  23,  582]])
```

In [32]:
```python
fig, axis = plt.subplots(2, 2, figsize=(4, 4))

for i in range(2):
    for j in range(2):
        rnd_idx = np.random.randint(0, len(preds))
        axis[i][j].imshow(val_dataset[rnd_idx][0][0], cmap="bone")
        axis[i][j].set_title(f"Pred:{int(preds[rnd_idx] > 0.5)}, Label:{labels[rnd_idx]}")
        axis[i][j].axis("off")
```



In [ ]:

In [ ]: