Deep Learning for Chest X-Ray Classification

A Technical Implementation with PyTorch Lightning

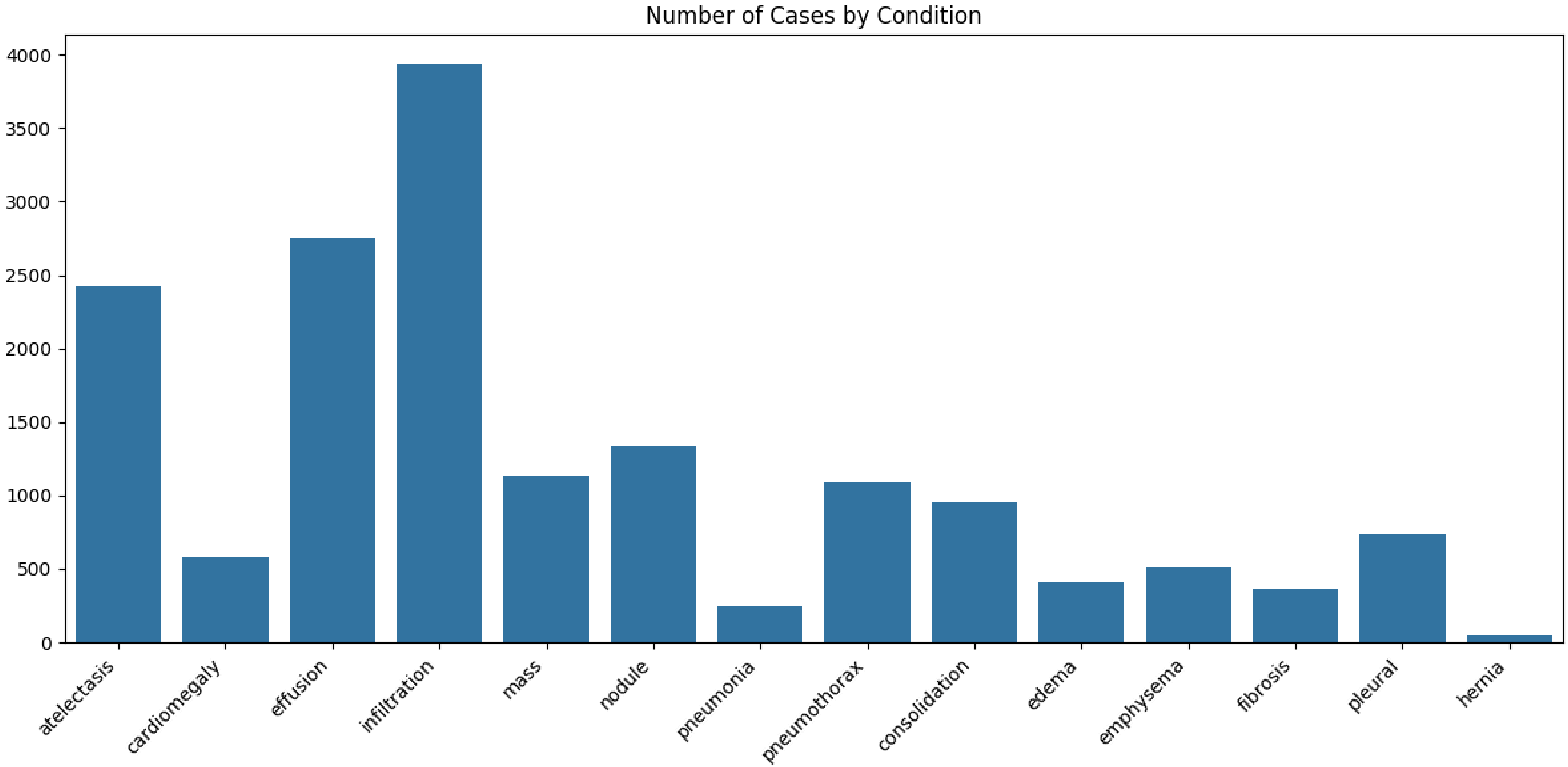
Project Overview

- Multi-label classification of chest X-rays
- 14 different pathological conditions
- Custom CNN and transfer learning approaches
- MLFlow for full experiment tracking and reproducibility
- Explainable AI techniques (Grad-CAM, Integrated Gradients)
- ❌ Didn't have time to implement hyperparameter optimization

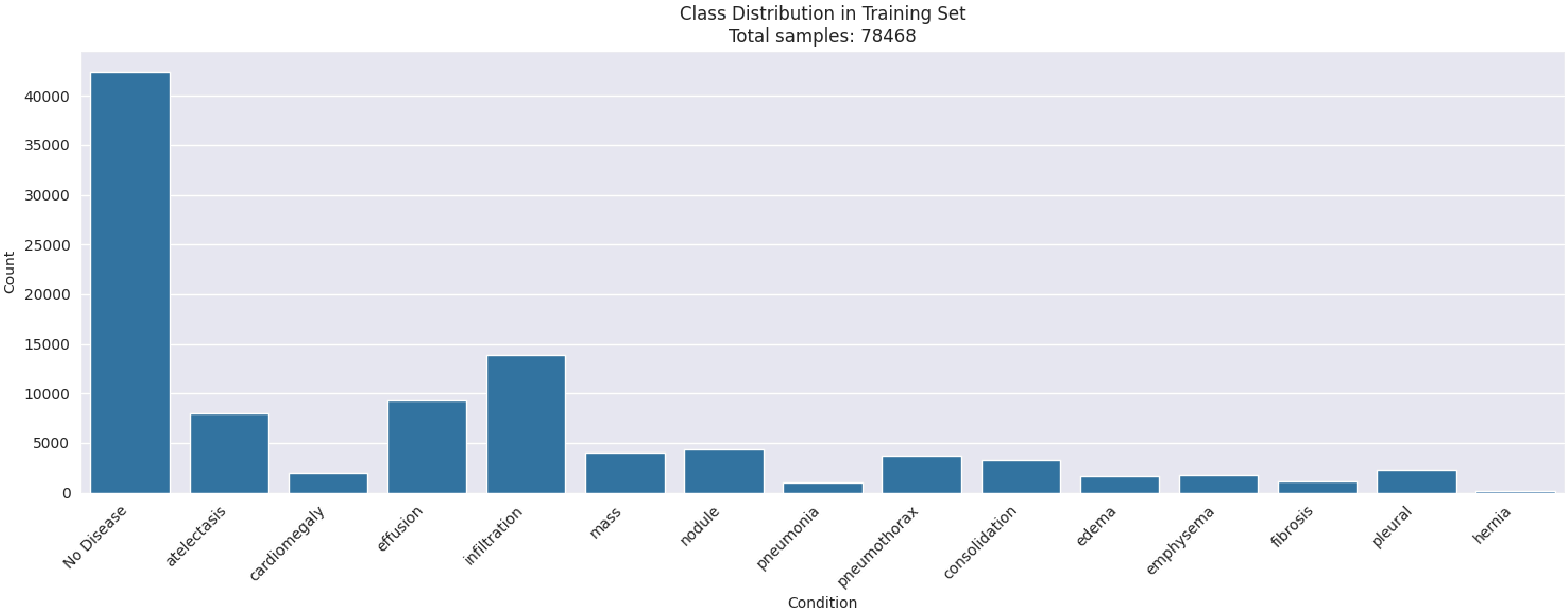
Medical Context

- Chest X-rays: Primary diagnostic tool
- Common conditions detected:
  - Pneumonia
  - Cardiomegaly
  - Edema
  - Pneumothorax
- Critical for rapid diagnosis and triage

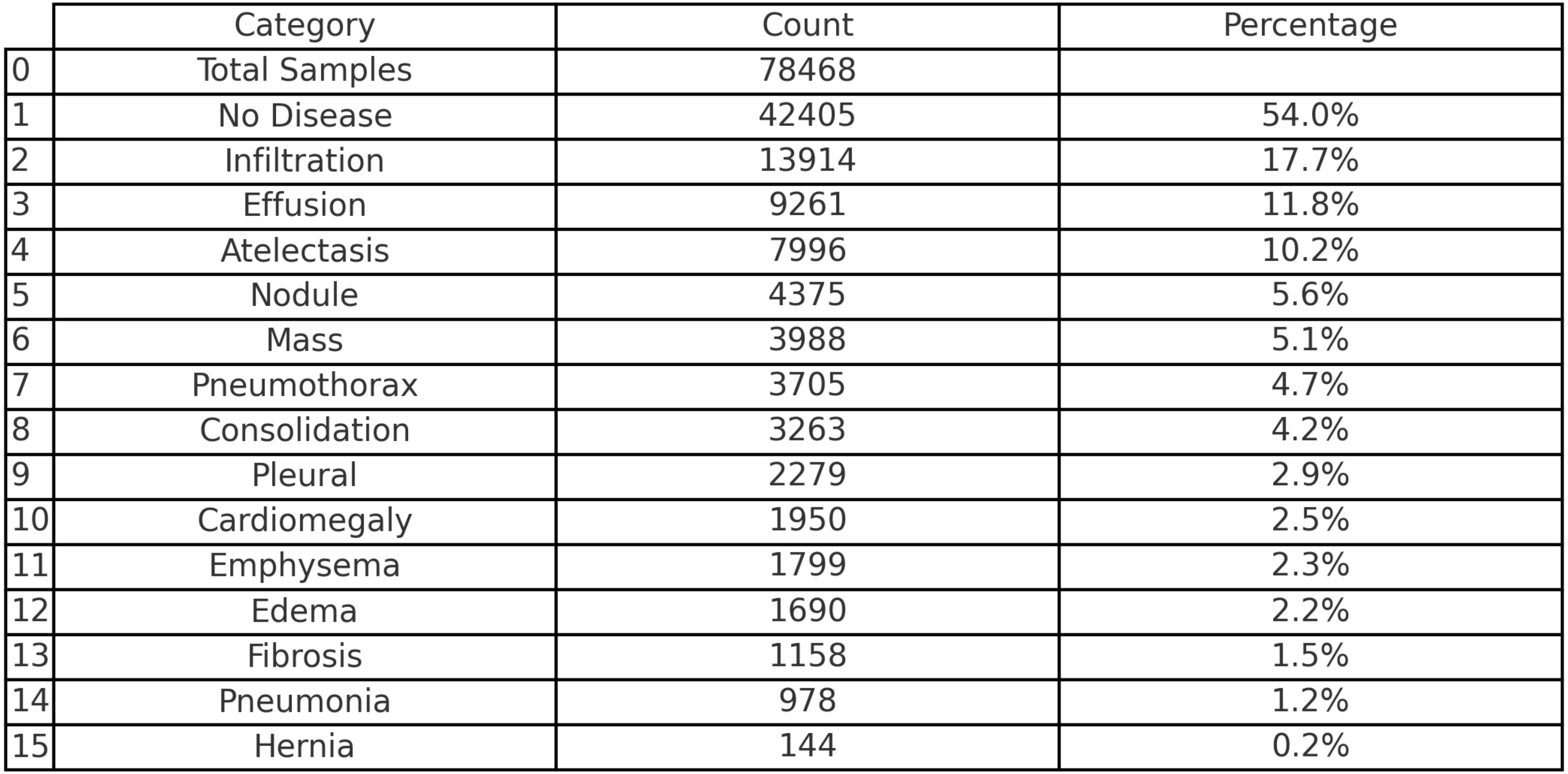
Class distribution (unbalanced)



Class Distribution with Normal



Dataset Statistics



	Category	Count	Percentage
0	Total Samples	78468	
1	No Disease	42405	54.0%
2	Infiltration	13914	17.7%
3	Effusion	9261	11.8%
4	Atelectasis	7996	10.2%
5	Nodule	4375	5.6%
6	Mass	3988	5.1%
7	Pneumothorax	3705	4.7%
8	Consolidation	3263	4.2%
9	Pleural	2279	2.9%
10	Cardiomegaly	1950	2.5%
11	Emphysema	1799	2.3%
12	Edema	1690	2.2%
13	Fibrosis	1158	1.5%
14	Pneumonia	978	1.2%
15	Hernia	144	0.2%

Technical Architecture

Infrastructure

- PyTorch Lightning for training
- MLflow for experiment tracking
- Mixed precision training (FP16)
- GPU acceleration

Model Design

- Custom CNN architecture
- ResNet transfer learning option
- Early stopping and learning rate scheduling

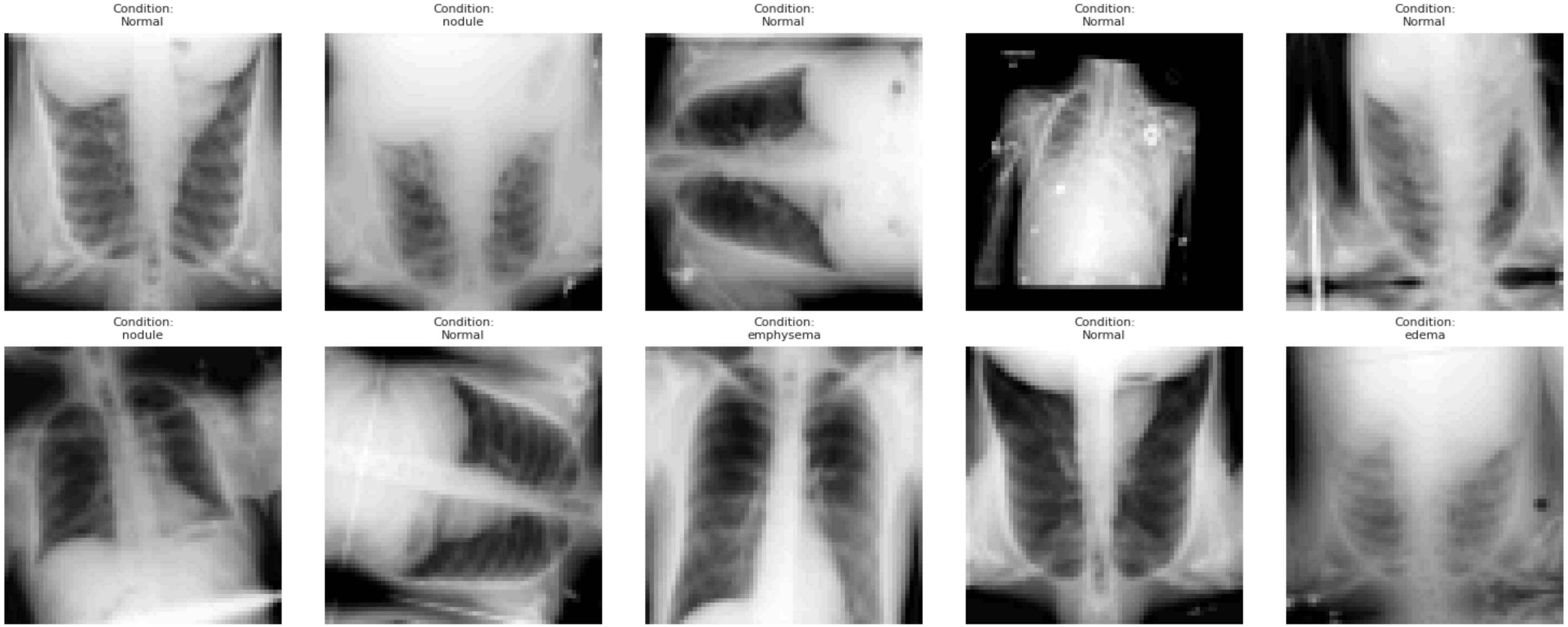
Training Process

1. Data Pipeline
    - Custom DataModule
    - Augmentation strategies
    - ❌ Class weight balancing (wasn't sure about the best approach)
2. Training Loop
    - Early stopping
    - Learning rate scheduling
    - Metrics monitoring
    - Experiment tracking

Augnementation Strategies

- Preprocessing

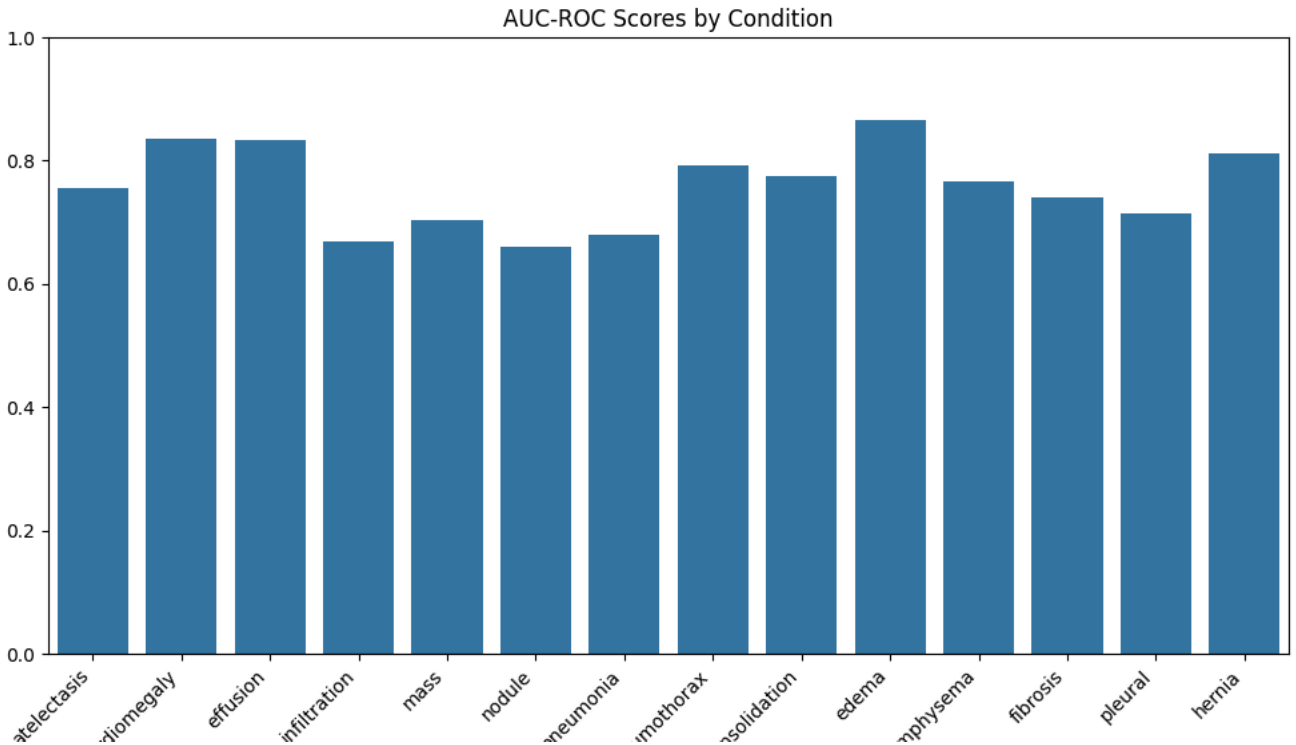
```
transforms = A.Compose([
                  A.RandomRotate90(p=0.5),
                  A.VerticalFlip(p=0.5),
                  A.ShiftScaleRotate(
                      shift_limit=0.1,
                      scale_limit=0.1,
                      rotate_limit=rotate_limit,
                      p=0.5,
                  ),
                  A.RandomBrightnessContrast(
                      brightness_limit=brightness, contrast_limit=contrast, p=0.2
                  ),
                  A.Normalize(mean=[0.5], std=[0.5]),
                  ToTensorV2(),
])
```

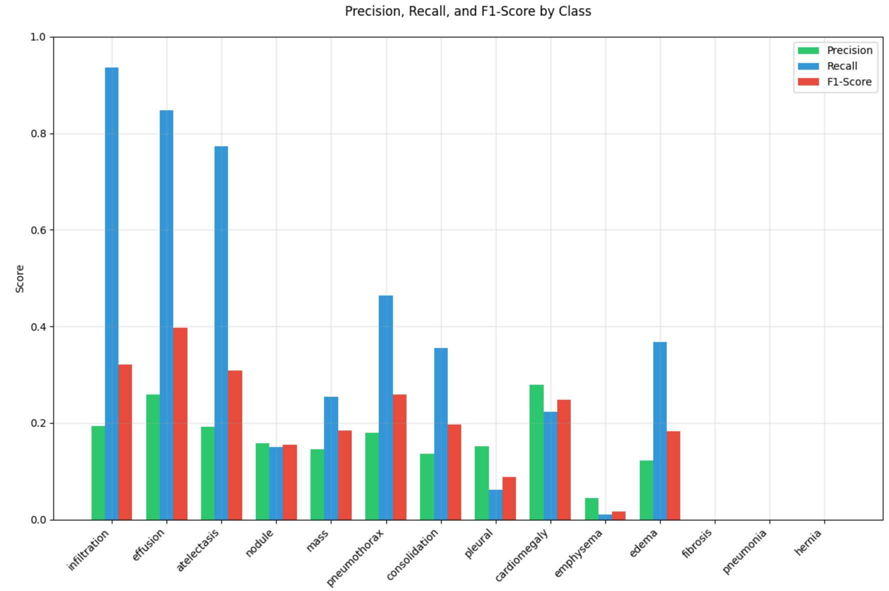

Condition: Normal
Condition: nodule
Condition: Normal
Condition: Normal
Condition: Normal
Condition: nodule
Condition: Normal
Condition: emphysema
Condition: Normal
Condition: edema

Custom CNN Architecture

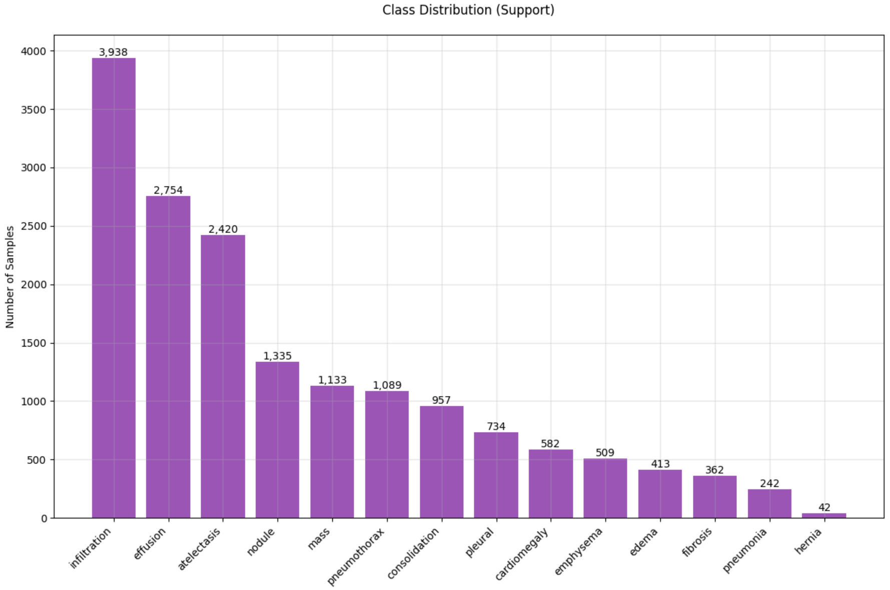
```python
class ChestNetS(ChestNetBase):
    """
    Simple CNN with three conv blocks and a classifier head. Each convolutional block includes convolution, batch normalization, ReLU activation,
    and max pooling operations. The classifier uses dropout for regularization and fully connected layers for final classification.
    Takes 64x64 grayscale images, outputs 14 binary classifications.
    """
        # Feature extraction backbone
        self.features = nn.Sequential(
            # Block 1: Input (1, 64, 64) -> Output (32, 32, 32)
            nn.Conv2d(1, 32, kernel_size=3, padding=1),
            nn.BatchNorm2d(32),  # Normalize activations for stable training
            nn.ReLU(inplace=True),
            nn.MaxPool2d(2, 2),  # Reduce spatial dimensions by 2x

            # Block 2: Input (32, 32, 32) -> Output (64, 16, 16)
            nn.Conv2d(32, 64, kernel_size=3, padding=1),
            nn.BatchNorm2d(64),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(2, 2),

            # Block 3: Input (64, 16, 16) -> Output (128, 8, 8)
            nn.Conv2d(64, 128, kernel_size=3, padding=1),
            nn.BatchNorm2d(128),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(2, 2),
        )

        # Classification head
        self.classifier = nn.Sequential(
            nn.Dropout(0.5),  # Prevent overfitting
            nn.Linear(128 * 8 * 8, 512),  # Flatten and project to 512 dimensions
            nn.ReLU(inplace=True),
            nn.Dropout(0.5),  # Additional dropout layer
            nn.Linear(512, 14),  # Final projection to output classes
        )

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        x = self.features(x)  # Extract visual features
        x = torch.flatten(x, 1)  # Flatten spatial dimensions
        x = self.classifier(x)  # Generate classification logits
        return x
```

CNN Metrics

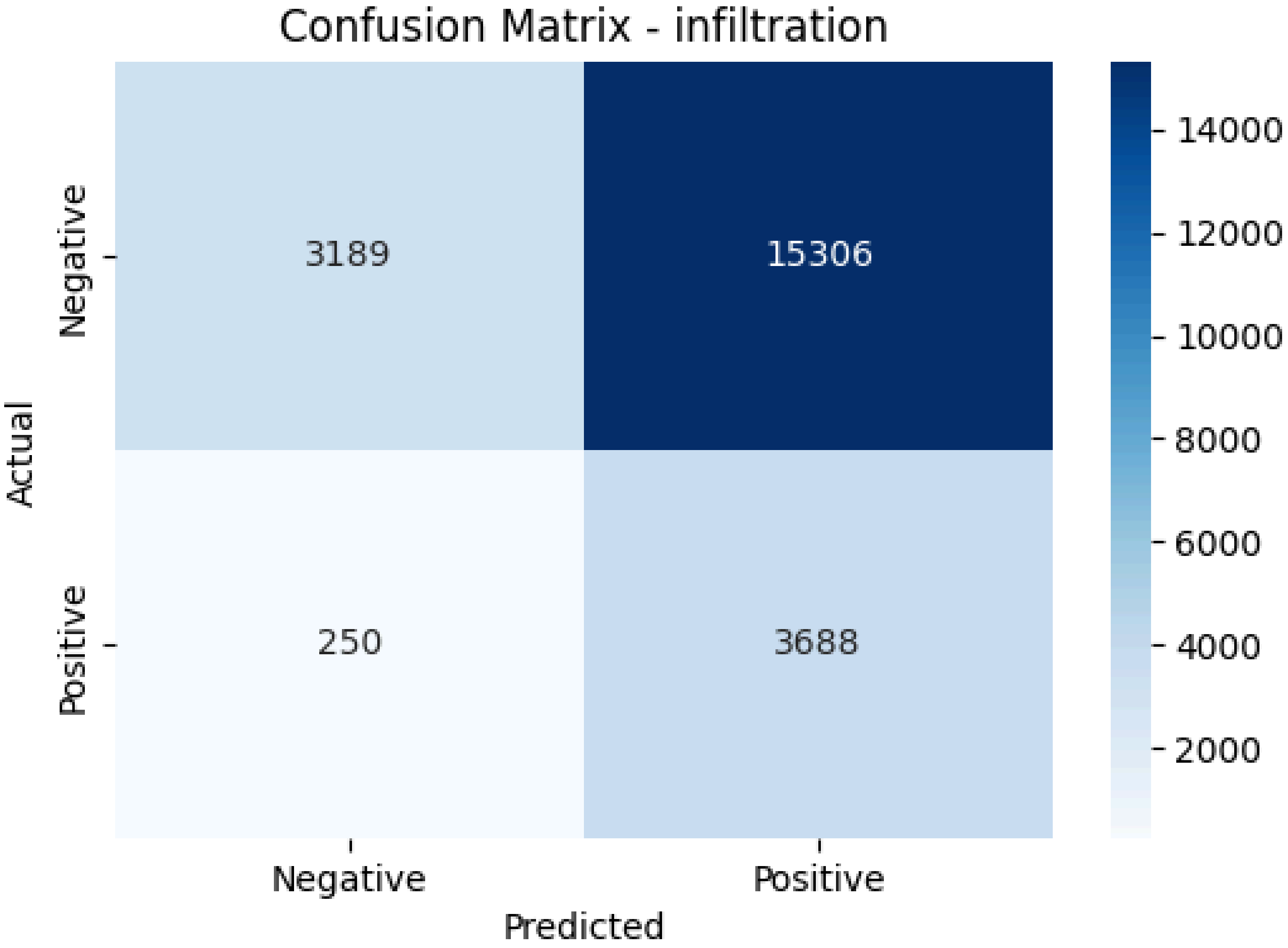

AUC-ROC Scores by Condition
1.0
0.8
0.6
0.4
0.2
0.0
atelectasis
rdiomegaly
effusion
infiltration
mass
nodule
neumonia
umothorax
nsolidation
edema
mphysema
fibrosis
pleural
hernia

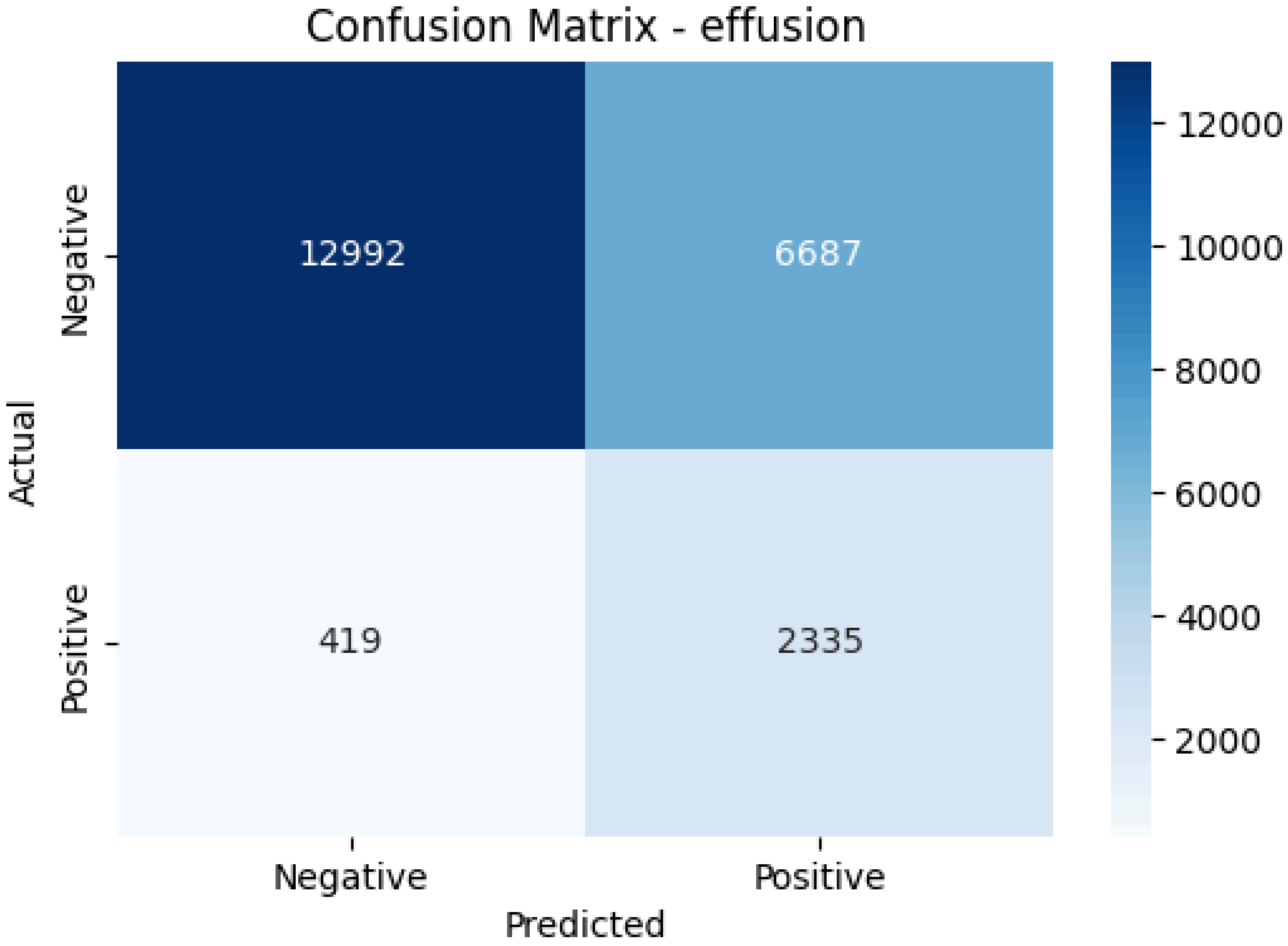


Precision, Recall, and F1-Score by Class

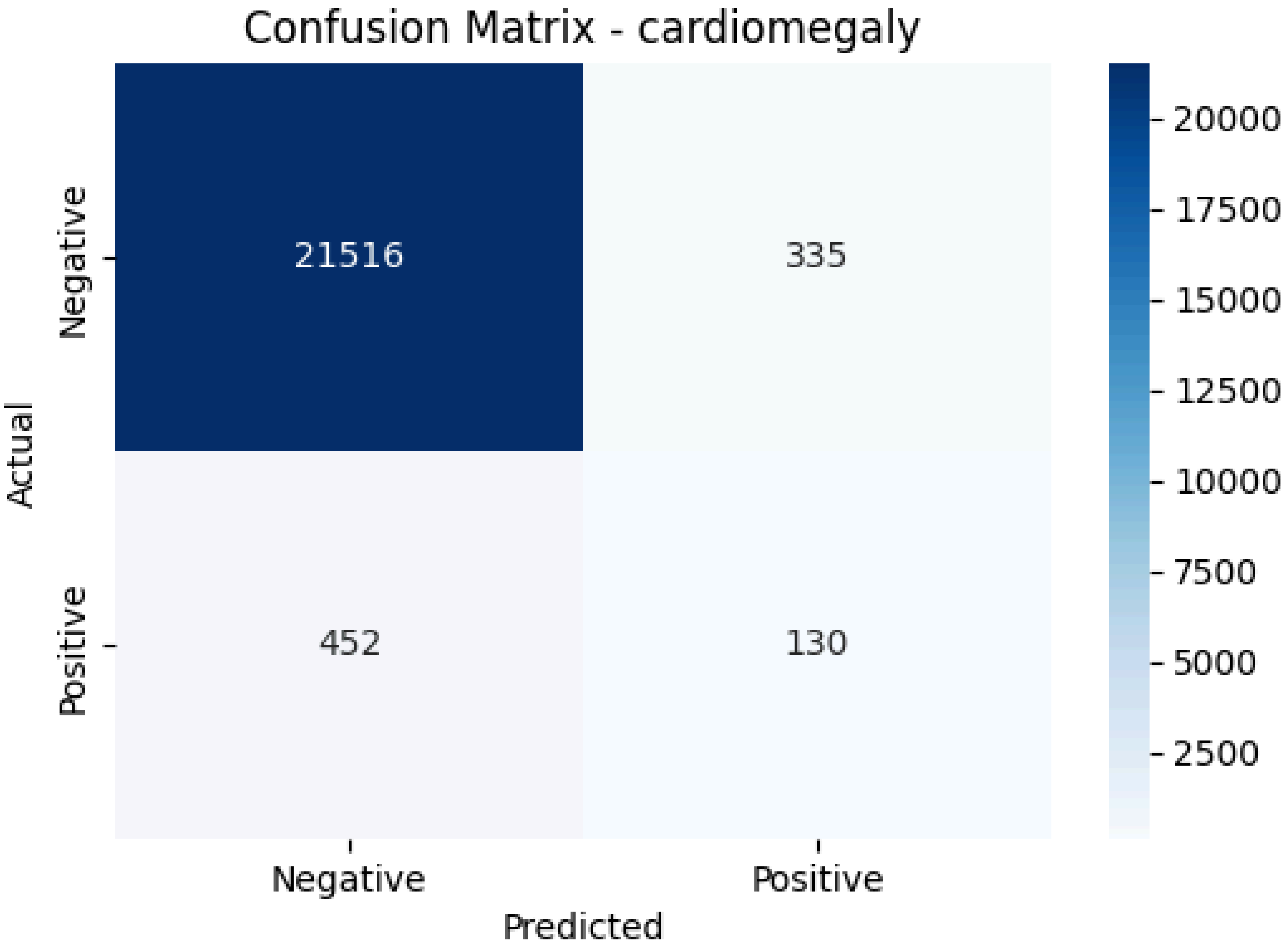


Class Distribution (Support)


Confusion Matrix - infiltration
Actual
Negative
Positive
3189
15306
250
3688
Negative
Positive
Predicted
14000
12000
10000
8000
6000
4000
2000

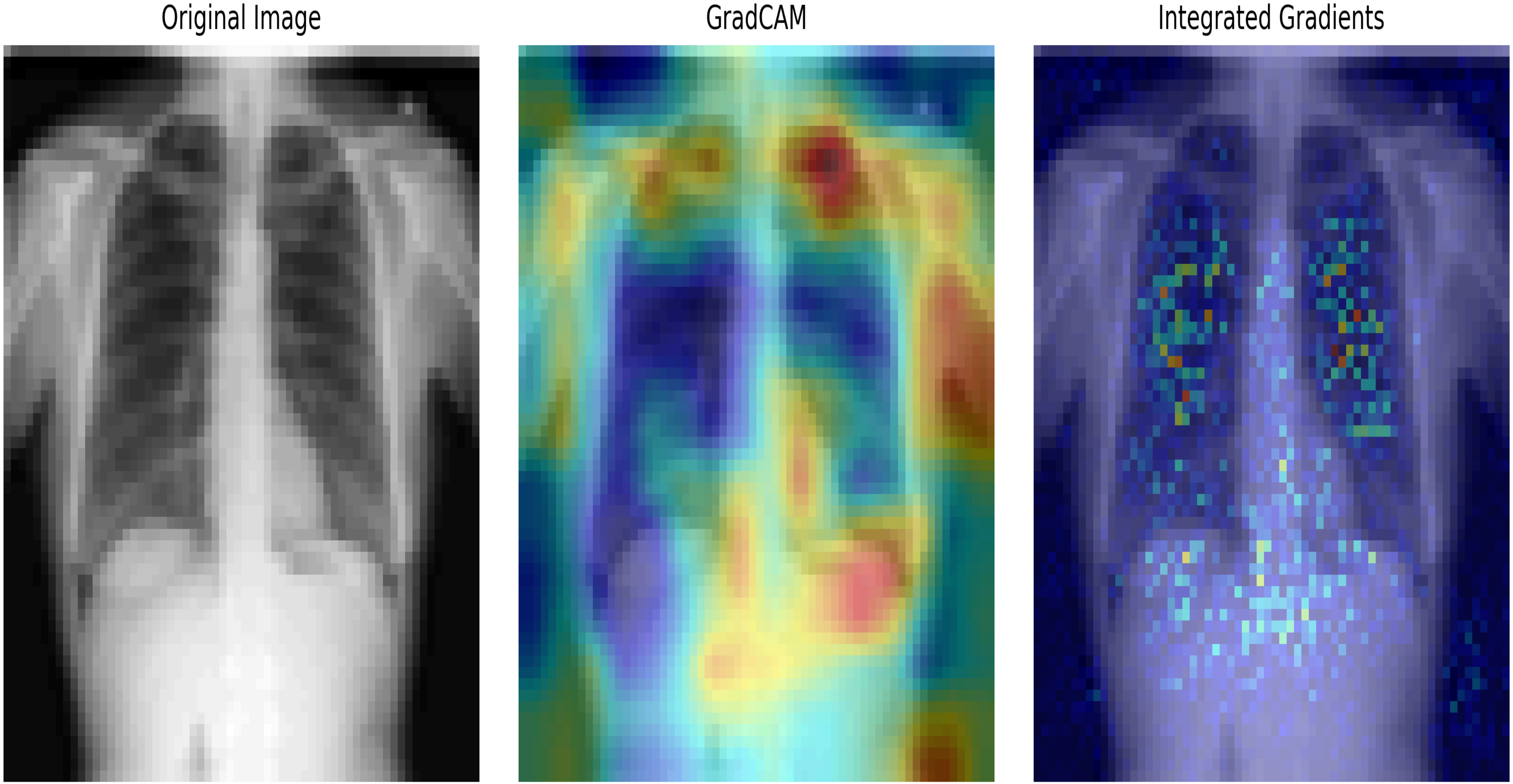


	Negative	Positive
Negative	12992	6687
Positive	419	2335



Confusion Matrix - cardiomegaly

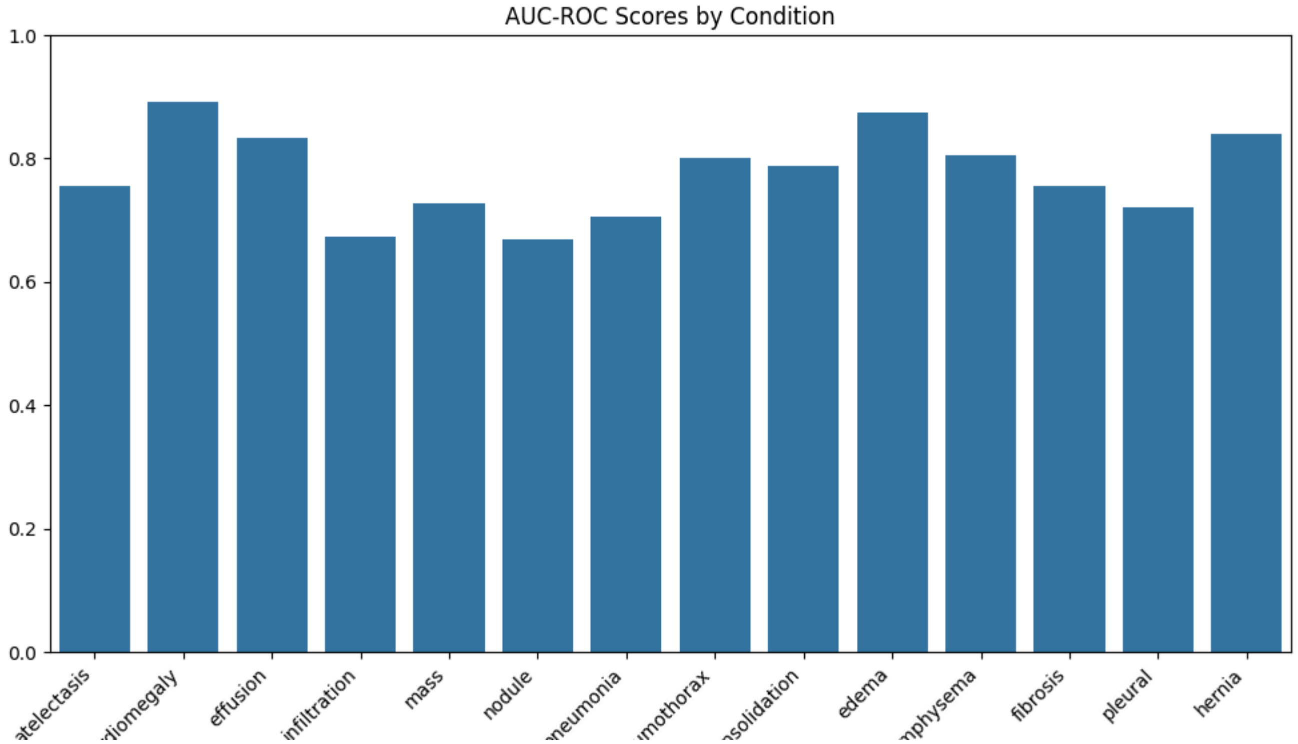
Actual \ Predicted	Negative	Positive
Negative	21516	335
Positive	452	130

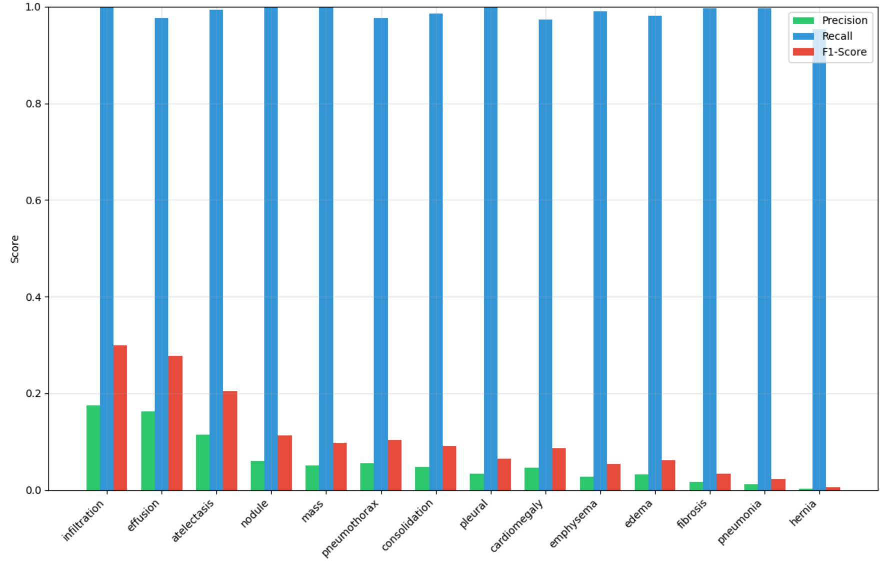

Original Image
GradCAM
Integrated Gradients

Resnet Architecture

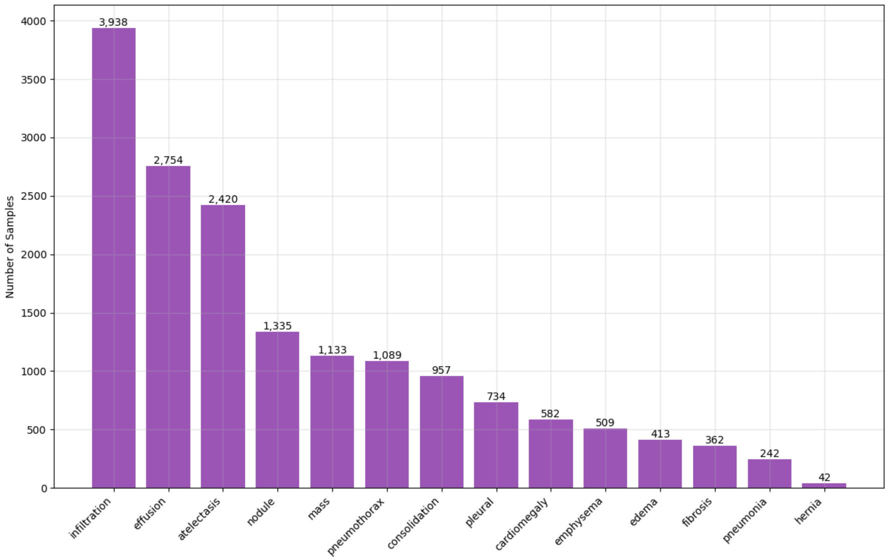
```python
class ChestNetResnet(ChestNetBase):
    # Input Size Compatibility: The ResNet-18 model is compatible with 64x64 inputs due to the
    # adaptive average pooling layer, which adjusts to varying spatial dimensions.
    # Pretrained Weights Handling: The first convolutional layer's weights are initialized by
    # averaging the pretrained RGB weights, preserving some pretrained features even with grayscale input.
    def __init__(
    ):
        # Load pretrained ResNet-18
        backbone = models.resnet18(pretrained=pretrained)
        # Modify first convolutional layer for grayscale input,
        original_conv1 = backbone.conv1
        self.backbone = backbone
        self.backbone.conv1 = nn.Conv2d(
            1,  # Input channels changed to 1
            original_conv1.out_channels,
            kernel_size=original_conv1.kernel_size,
            stride=original_conv1.stride,
            padding=original_conv1.padding,
            bias=False,
        )
        # Initialize weights from pretrained model
        if pretrained:
            with torch.no_grad():
                self.backbone.conv1.weight.copy_(
                    original_conv1.weight.mean(dim=1, keepdim=True)
                )
        # Replace final fully connected layer
        in_features = self.backbone.fc.in_features
        self.backbone.fc = nn.Linear(in_features, num_classes)
        # Define feature extractor
        self.features = nn.Sequential(
            self.backbone.conv1,
            self.backbone.bn1,
            self.backbone.relu,
            self.backbone.maxpool,
            self.backbone.layer1,
            self.backbone.layer2,
            self.backbone.layer3,
            self.backbone.layer4,
            self.backbone.avgpool,
        )

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        x = self.features(x)
        x = torch.flatten(x, 1)
        x = self.backbone.fc(x)
        return x
```

Resnet Metrics

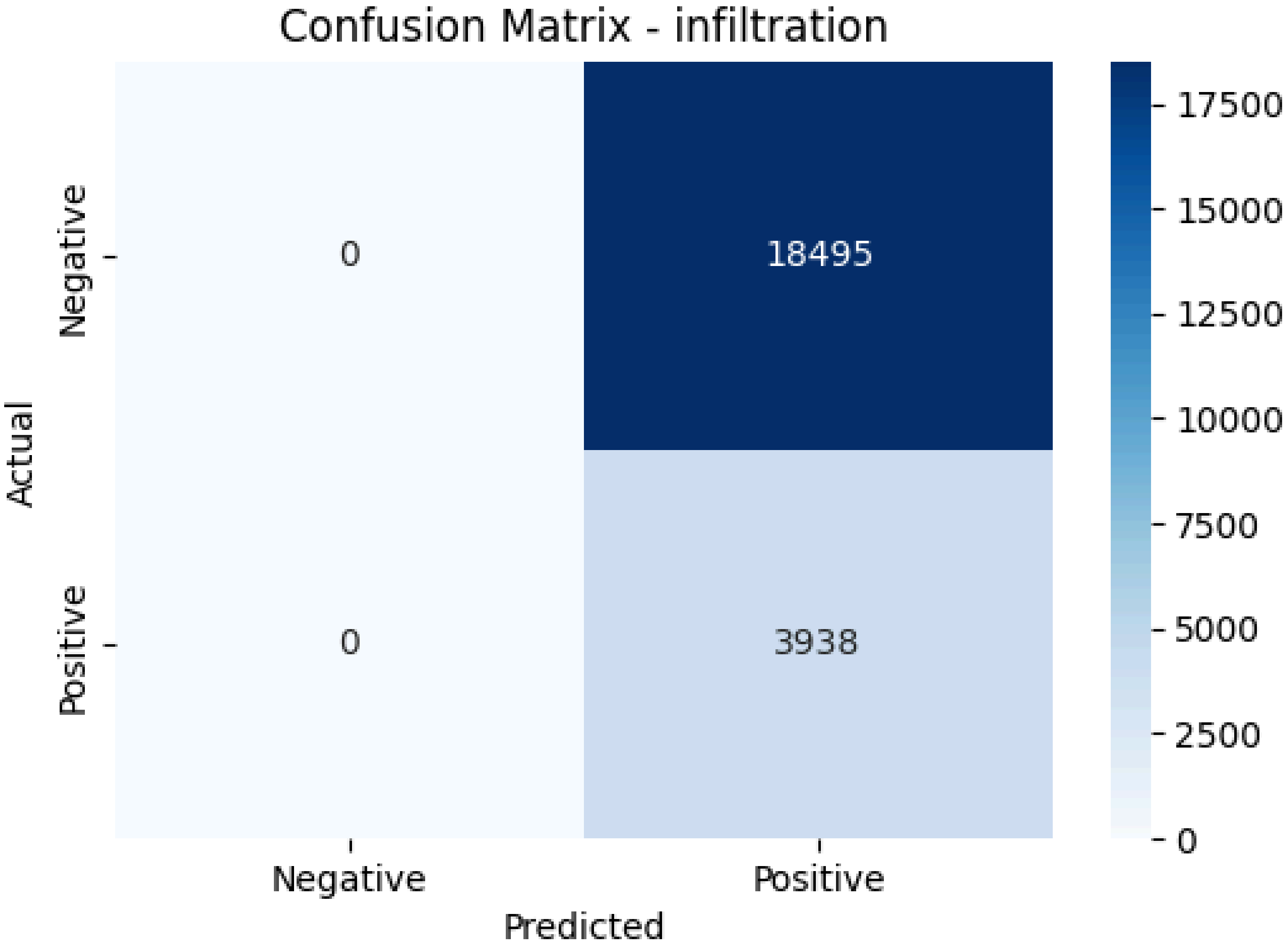


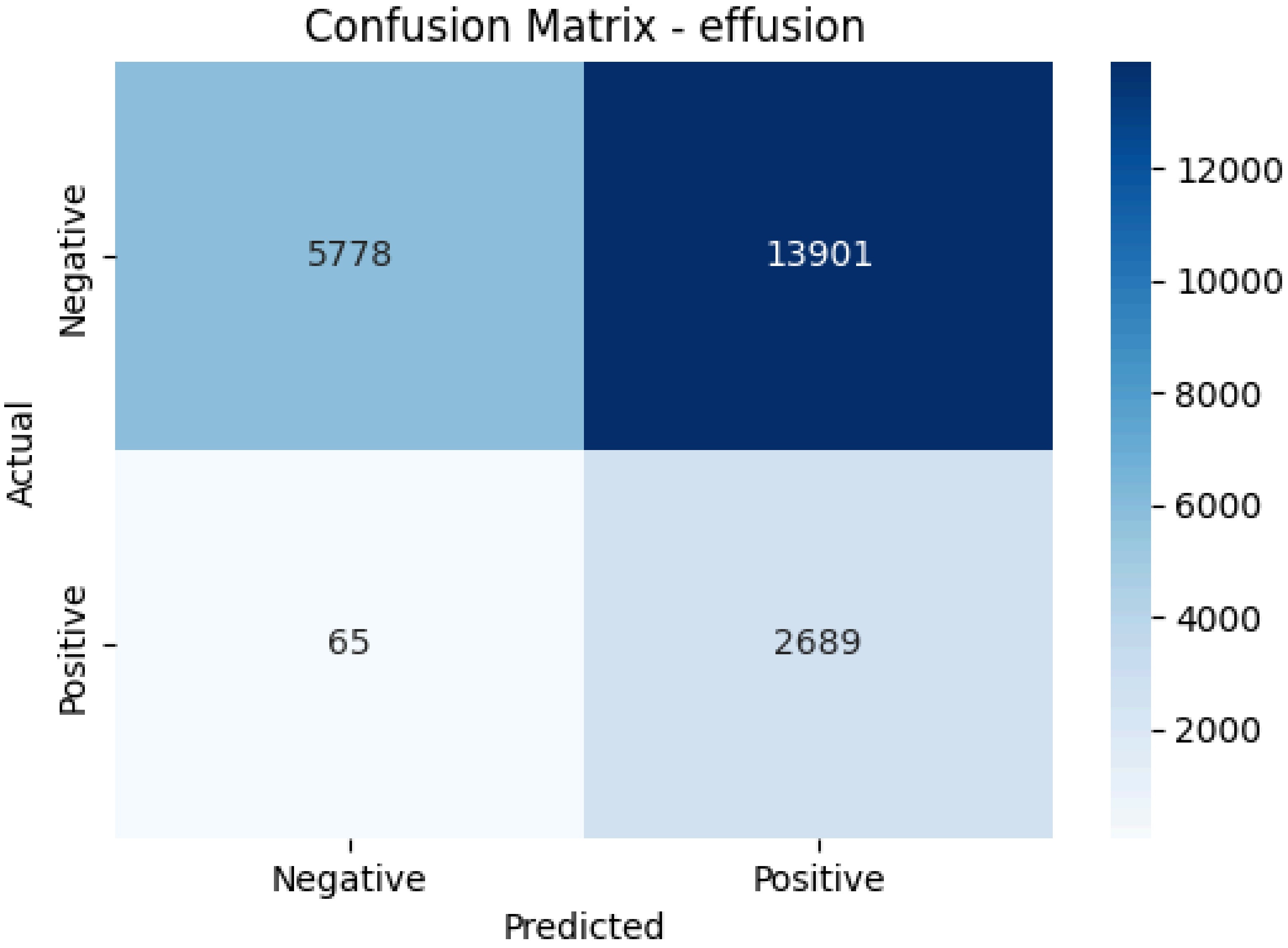


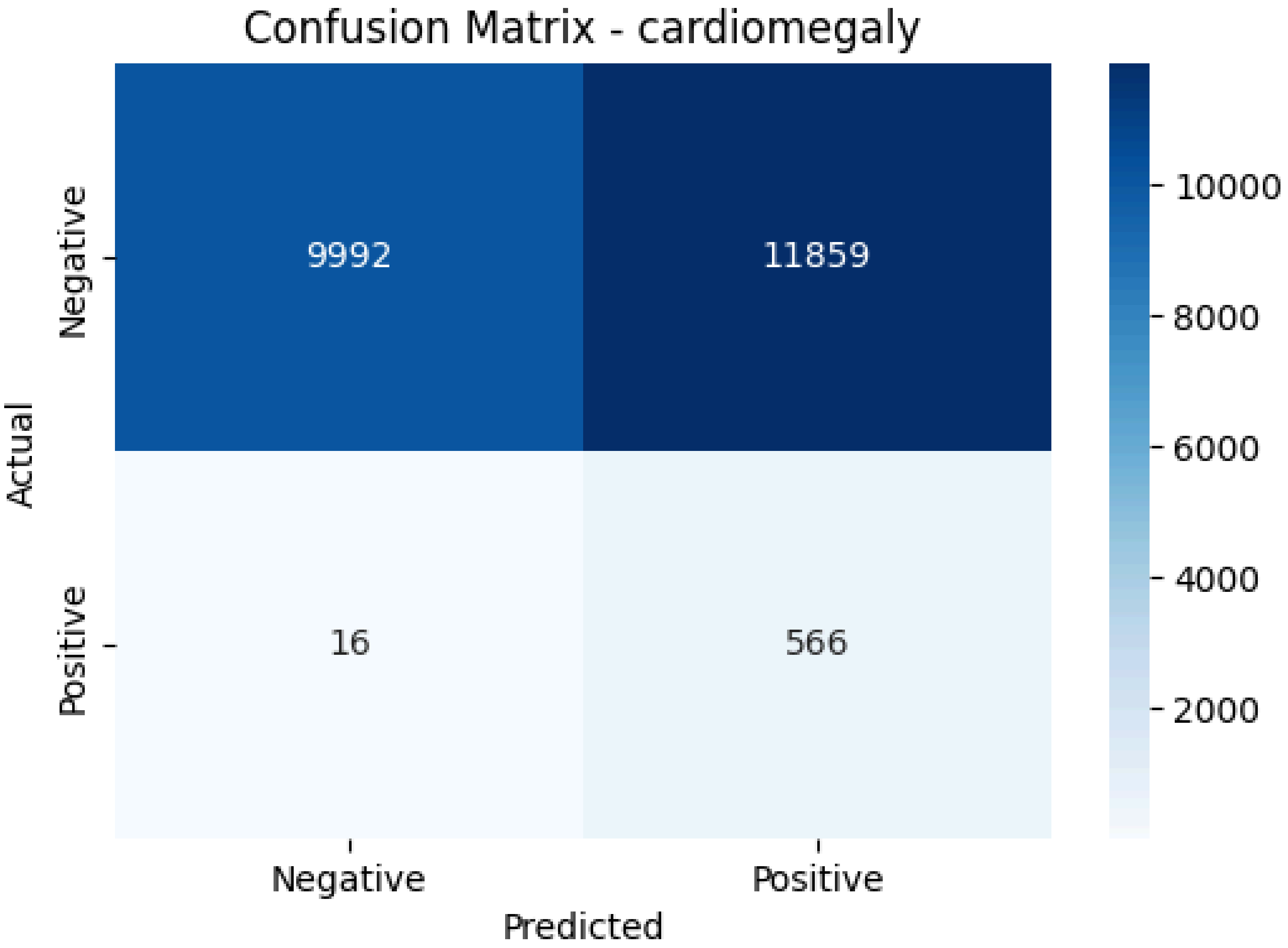
Precision, Recall, and F1-Score by Class



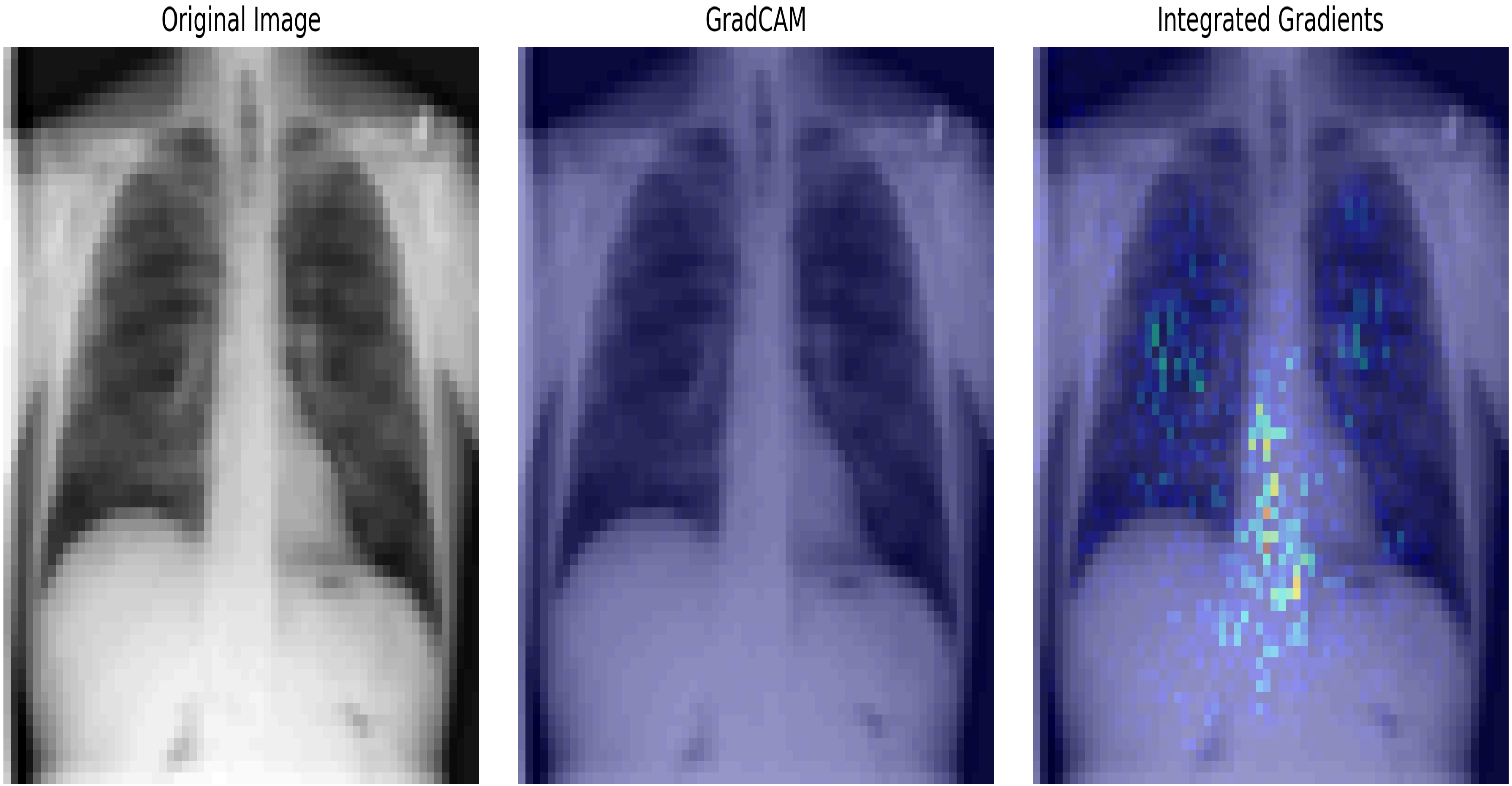
Class Distribution (Support)


Confusion Matrix - infiltration
Actual
Negative
Positive
Predicted
Negative
Positive
0
18495
0
3938
17500
15000
12500
10000
7500
5000
2500
0


Confusion Matrix - effusion
Actual
Negative
Positive
5778
13901
65
2689
Negative
Positive
Predicted
12000
10000
8000
6000
4000
2000



Confusion Matrix - cardiomegaly



Original Image

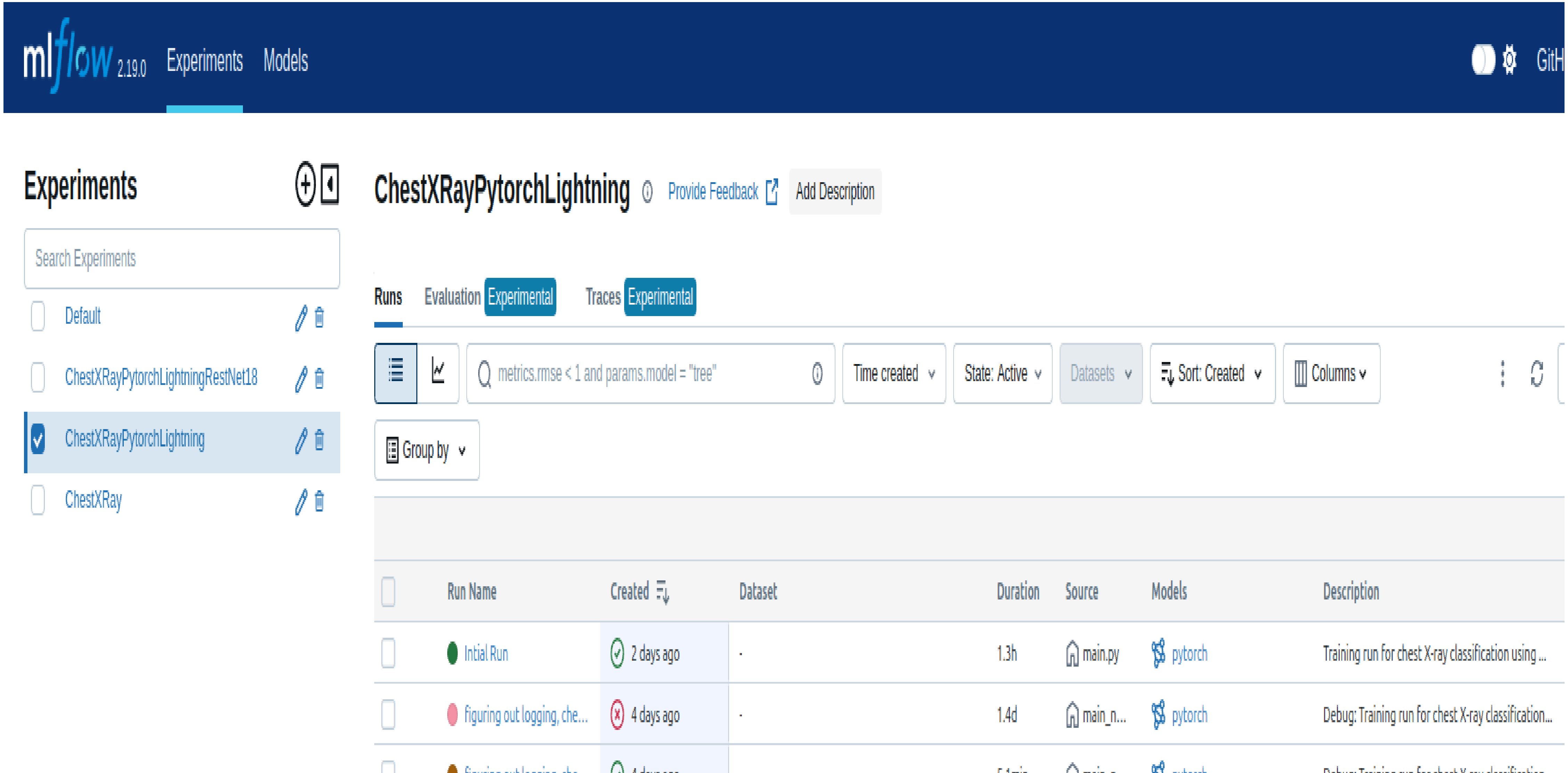
GradCAM

Integrated Gradients

Hypothesis

- I think my early stopping might be flawed, Resnet early stopped at 40 epochs, while CNN at 125 epochs
- Class imbalance affecting the model performance
- I had to write statistics by hand, so I might have made a mistake
- 64x64 resolution might be too low for the model to learn the features (can a human learn from 64x64 images?)
- I didn't have time to implement hyperparameter optimization

Experiment Tracking



Experiments

Search Experiments

- ☐ Default
- ☐ ChestXRayPytorchLightningRestNet18
- ☑ ChestXRayPytorchLightning
- ☐ ChestXRay

ChestXRayPytorchLightning

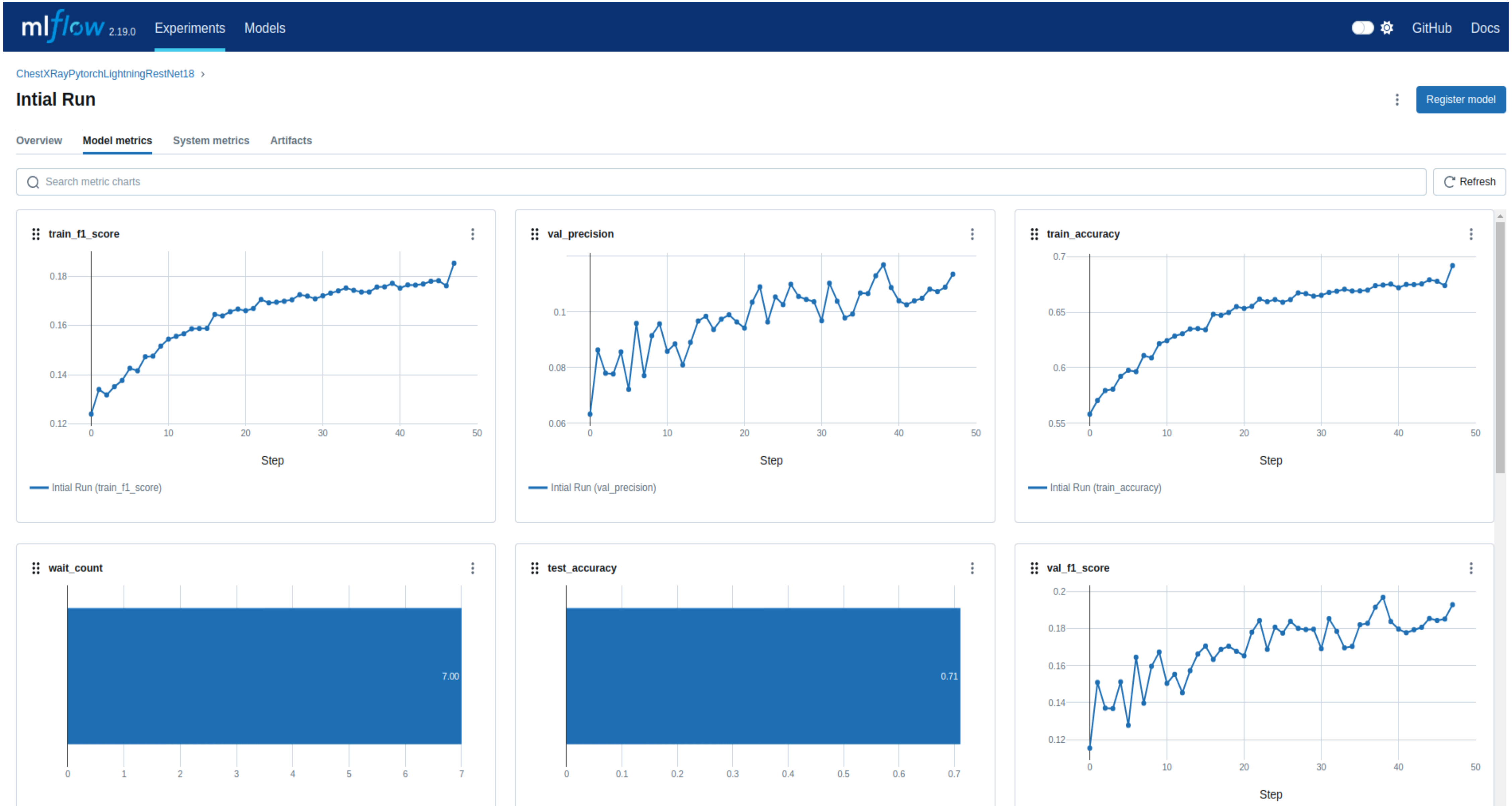
metrics.rmse < 1 and params.model = "tree"

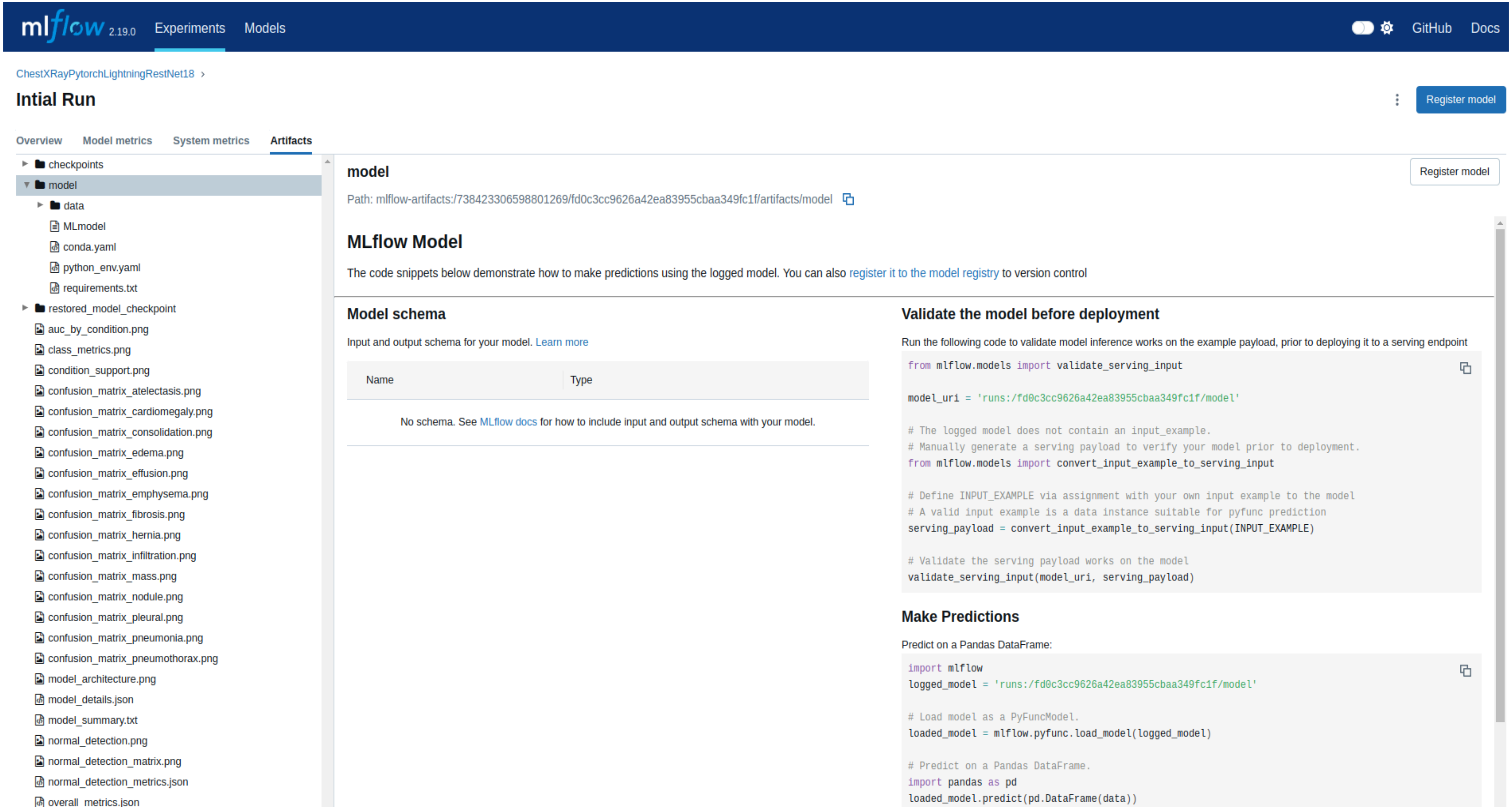
Time created

State: Active

Sort: Created

☐	Run Name	Created	Dataset	Duration	Source	Models	Description
☐	● Intial Run	✓ 2 days ago	-	1.3h	main.py	pytorch	Training run for chest X-ray classification using ...
☐	● figuring out logging, che...	☒ 4 days ago	-	1.4d	main_n...	pytorch	Debug: Training run for chest X-ray classification...


mlflow 2.19.0
Experiments
Models
GitHub
Docs
ChestXRayPytorchLightningRestNet18 ›
Intial Run
Register model
Overview
Model metrics
System metrics
Artifacts
Search metric charts
Refresh
train_f1_score
0.18
0.16
0.14
0.12
0
10
20
30
40
50
Step
Intial Run (train_f1_score)
val_precision
0.1
0.08
0.06
0
10
20
30
40
50
Step
Intial Run (val_precision)
train_accuracy
0.7
0.65
0.6
0.55
0
10
20
30
40
50
Step
Intial Run (train_accuracy)
wait_count
7.00
0
1
2
3
4
5
6
7
test_accuracy
0.71
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
val_f1_score
0.2
0.18
0.16
0.14
0.12
0
10
20
30
40
50
Step



Intial Run

- checkpoints
- model
  - data
  - MLmodel
  - conda.yaml
  - python_env.yaml
  - requirements.txt
- restored_model_checkpoint
- auc_by_condition.png
- class_metrics.png
- condition_support.png
- confusion_matrix_atelectasis.png
- confusion_matrix_cardiomegaly.png
- confusion_matrix_consolidation.png
- confusion_matrix_edema.png
- confusion_matrix_effusion.png
- confusion_matrix_emphysema.png
- confusion_matrix_fibrosis.png
- confusion_matrix_hernia.png
- confusion_matrix_infiltration.png
- confusion_matrix_mass.png
- confusion_matrix_nodule.png
- confusion_matrix_pleural.png
- confusion_matrix_pneumonia.png
- confusion_matrix_pneumothorax.png
- model_architecture.png
- model_details.json
- model_summary.txt
- normal_detection.png
- normal_detection_matrix.png
- normal_detection_metrics.json
- overall_metrics.json

model

Path: mlflow-artifacts:/738423306598801269/fd0c3cc9626a42ea83955cbaa349fc1f/artifacts/model

MLflow Model

The code snippets below demonstrate how to make predictions using the logged model. You can also register it to the model registry to version control

Model schema

Input and output schema for your model. Learn more

Name	Type
No schema. See MLflow docs for how to include input and output schema with your model.	

Validate the model before deployment

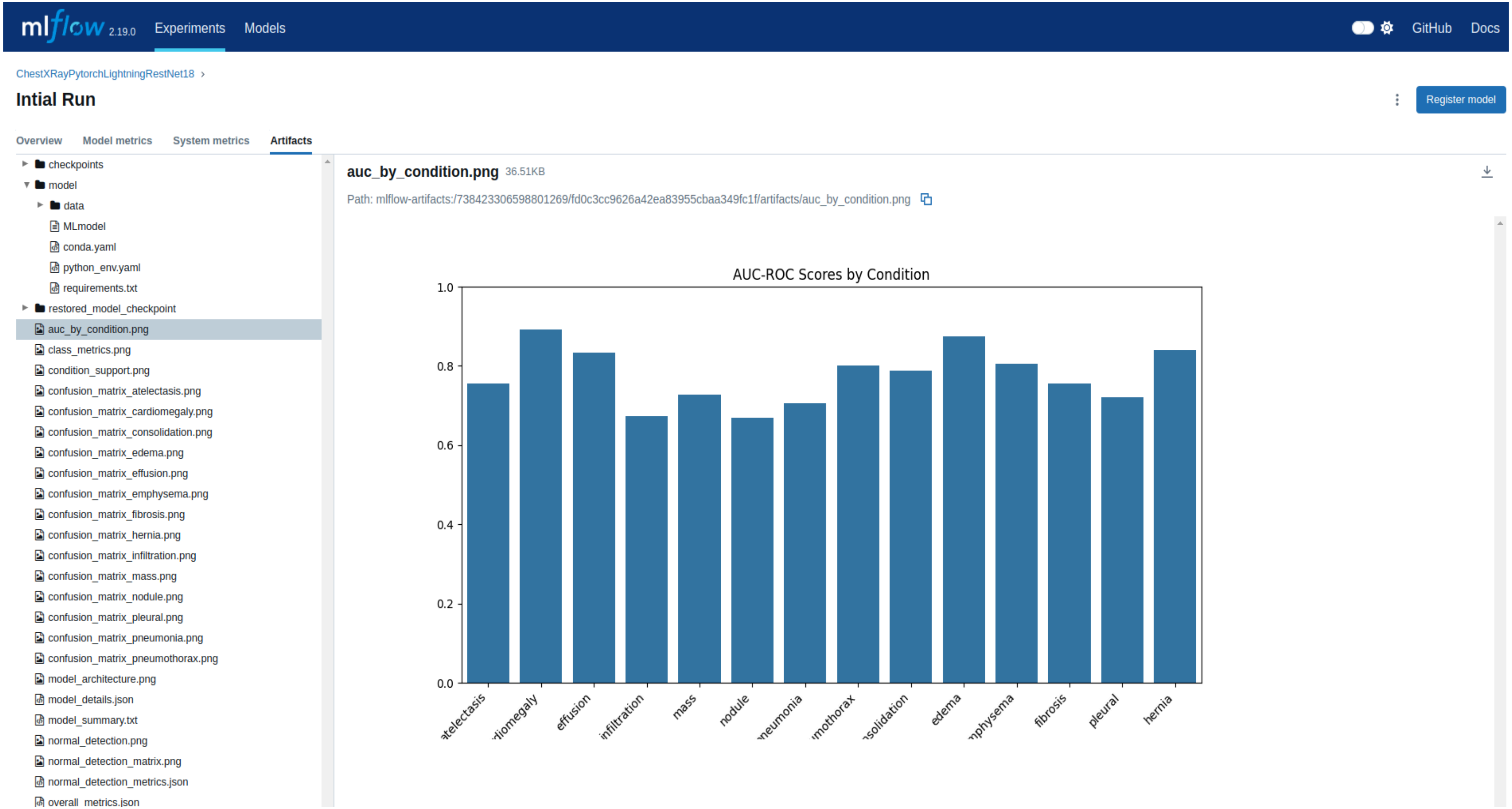
Run the following code to validate model inference works on the example payload, prior to deploying it to a serving endpoint

```
from mlflow.models import validate_serving_input

model_uri = 'runs:/fd0c3cc9626a42ea83955cbaa349fc1f/model'

# The logged model does not contain an input_example.
# Manually generate a serving payload to verify your model prior to deployment.
from mlflow.models import convert_input_example_to_serving_input

# Define INPUT_EXAMPLE via assignment with your own input example to the model
# A valid input example is a data instance suitable for pyfunc prediction
serving_payload = convert_input_example_to_serving_input(INPUT_EXAMPLE)

# Validate the serving payload works on the model
validate_serving_input(model_uri, serving_payload)
```

Make Predictions

Predict on a Pandas DataFrame:

```
import mlflow
logged_model = 'runs:/fd0c3cc9626a42ea83955cbaa349fc1f/model'

# Load model as a PyFuncModel.
loaded_model = mlflow.pyfunc.load_model(logged_model)

# Predict on a Pandas DataFrame.
import pandas as pd
loaded_model.predict(pd.DataFrame(data))
```



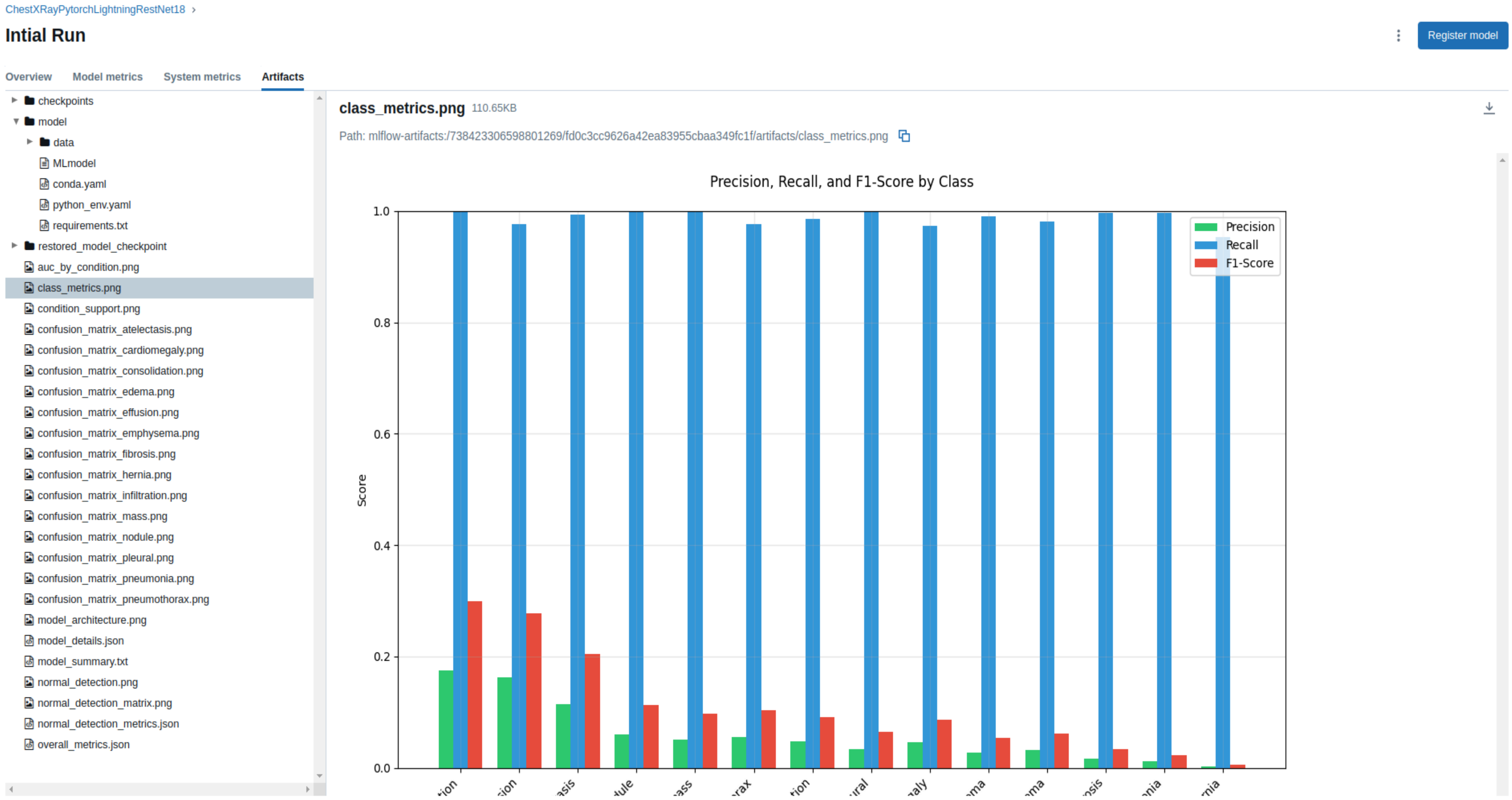
ChestXRayPytorchLightningRestNet18 ›

Intial Run

- checkpoints
- model
  - data
  - MLmodel
  - conda.yaml
  - python_env.yaml
  - requirements.txt
- restored_model_checkpoint
- auc_by_condition.png
- class_metrics.png
- condition_support.png
- confusion_matrix_atelectasis.png
- confusion_matrix_cardiomegaly.png
- confusion_matrix_consolidation.png
- confusion_matrix_edema.png
- confusion_matrix_effusion.png
- confusion_matrix_emphysema.png
- confusion_matrix_fibrosis.png
- confusion_matrix_hernia.png
- confusion_matrix_infiltration.png
- confusion_matrix_mass.png
- confusion_matrix_nodule.png
- confusion_matrix_pleural.png
- confusion_matrix_pneumonia.png
- confusion_matrix_pneumothorax.png
- model_architecture.png
- model_details.json
- model_summary.txt
- normal_detection.png
- normal_detection_matrix.png
- normal_detection_metrics.json
- overall_metrics.json

auc_by_condition.png 36.51KB

Path: mlflow-artifacts:/738423306598801269/fd0c3cc9626a42ea83955cbaa349fc1f/artifacts/auc_by_condition.png



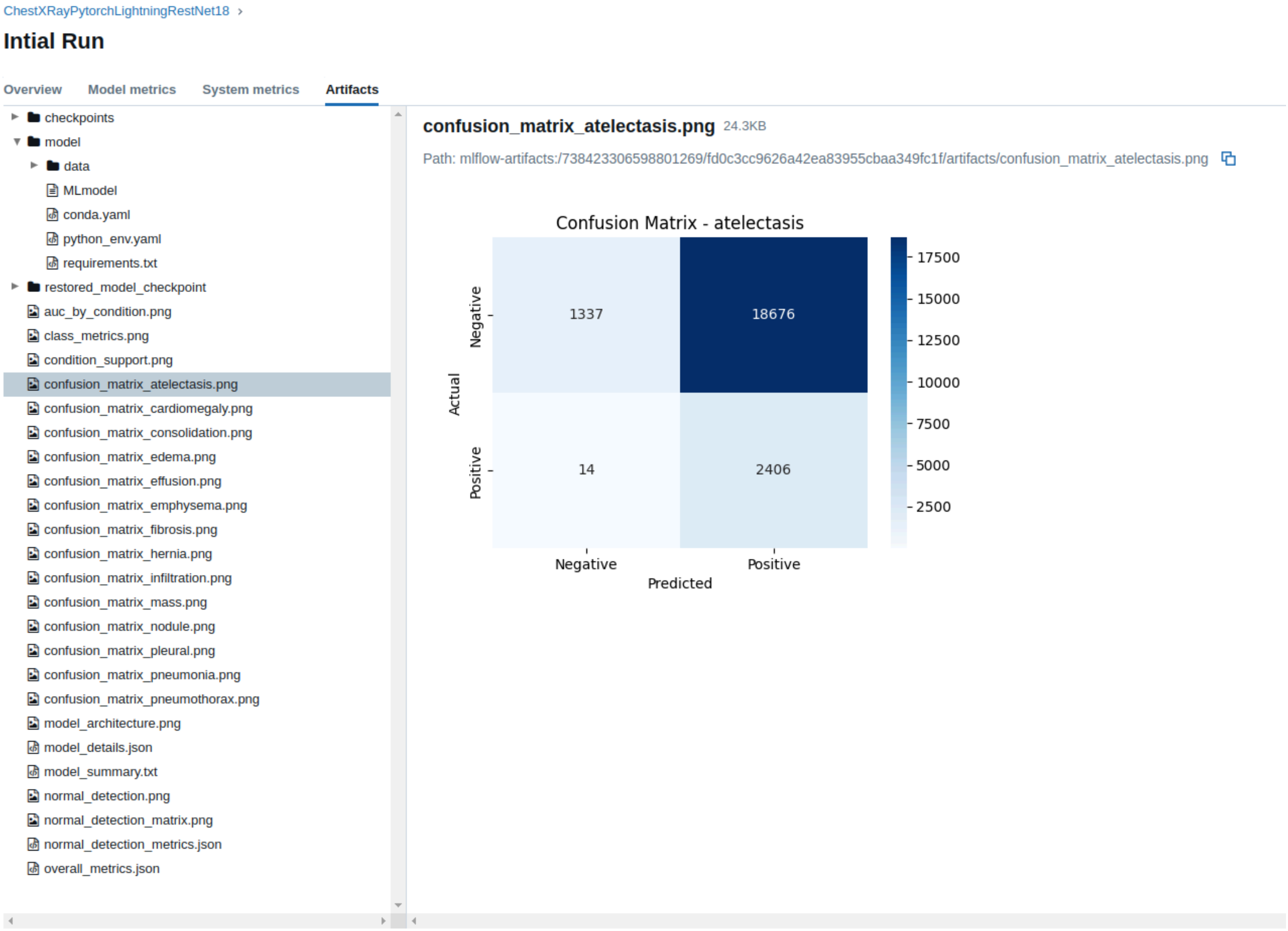
ChestXRayPytorchLightningRestNet18 ›

Intial Run

- ▸ checkpoints
- ▾ model
  - ▸ data
  - MLmodel
  - conda.yaml
  - python_env.yaml
  - requirements.txt
- ▸ restored_model_checkpoint
- auc_by_condition.png
- class_metrics.png
- condition_support.png
- confusion_matrix_atelectasis.png
- confusion_matrix_cardiomegaly.png
- confusion_matrix_consolidation.png
- confusion_matrix_edema.png
- confusion_matrix_effusion.png
- confusion_matrix_emphysema.png
- confusion_matrix_fibrosis.png
- confusion_matrix_hernia.png
- confusion_matrix_infiltration.png
- confusion_matrix_mass.png
- confusion_matrix_nodule.png
- confusion_matrix_pleural.png
- confusion_matrix_pneumonia.png
- confusion_matrix_pneumothorax.png
- model_architecture.png
- model_details.json
- model_summary.txt
- normal_detection.png
- normal_detection_matrix.png
- normal_detection_metrics.json
- overall_metrics.json

class_metrics.png 110.65KB

Path: mlflow-artifacts:/738423306598801269/fd0c3cc9626a42ea83955cbaa349fc1f/artifacts/class_metrics.png



ChestXRayPytorchLightningRestNet18 ›

Intial Run

Overview | Model metrics | System metrics | **Artifacts**

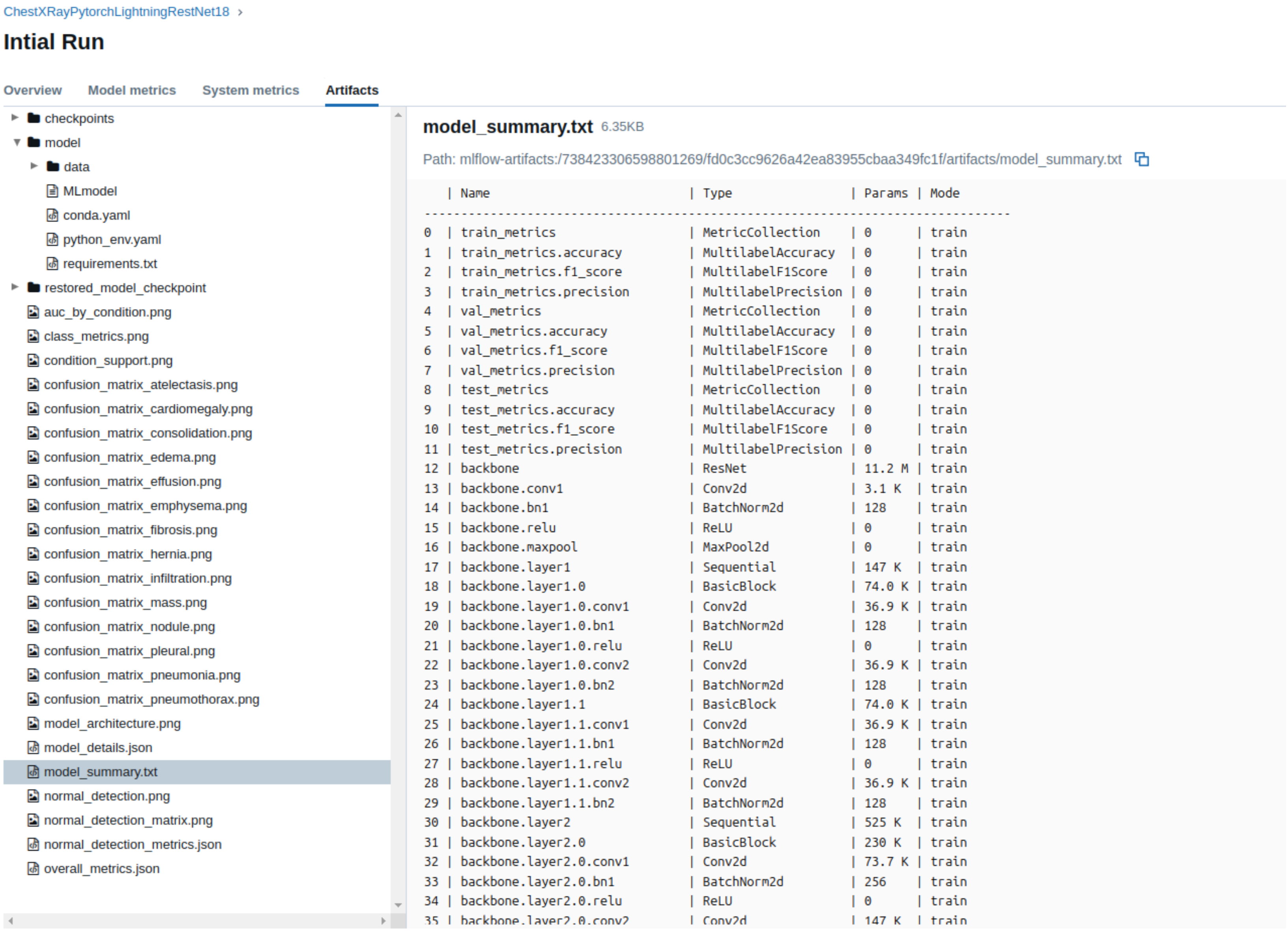
- checkpoints
- model
  - data
  - MLmodel
  - conda.yaml
  - python_env.yaml
  - requirements.txt
- restored_model_checkpoint
- auc_by_condition.png
- class_metrics.png
- condition_support.png
- confusion_matrix_atelectasis.png
- confusion_matrix_cardiomegaly.png
- confusion_matrix_consolidation.png
- confusion_matrix_edema.png
- confusion_matrix_effusion.png
- confusion_matrix_emphysema.png
- confusion_matrix_fibrosis.png
- confusion_matrix_hernia.png
- confusion_matrix_infiltration.png
- confusion_matrix_mass.png
- confusion_matrix_nodule.png
- confusion_matrix_pleural.png
- confusion_matrix_pneumonia.png
- confusion_matrix_pneumothorax.png
- model_architecture.png
- model_details.json
- model_summary.txt
- normal_detection.png
- normal_detection_matrix.png
- normal_detection_metrics.json
- overall_metrics.json

confusion_matrix_atelectasis.png 24.3KB

Path: mlflow-artifacts:/738423306598801269/fd0c3cc9626a42ea83955cbaa349fc1f/artifacts/confusion_matrix_atelectasis.png

Confusion Matrix - atelectasis

Actual \ Predicted	Negative	Positive
Negative	1337	18676
Positive	14	2406



ChestXRayPytorchLightningRestNet18 ›

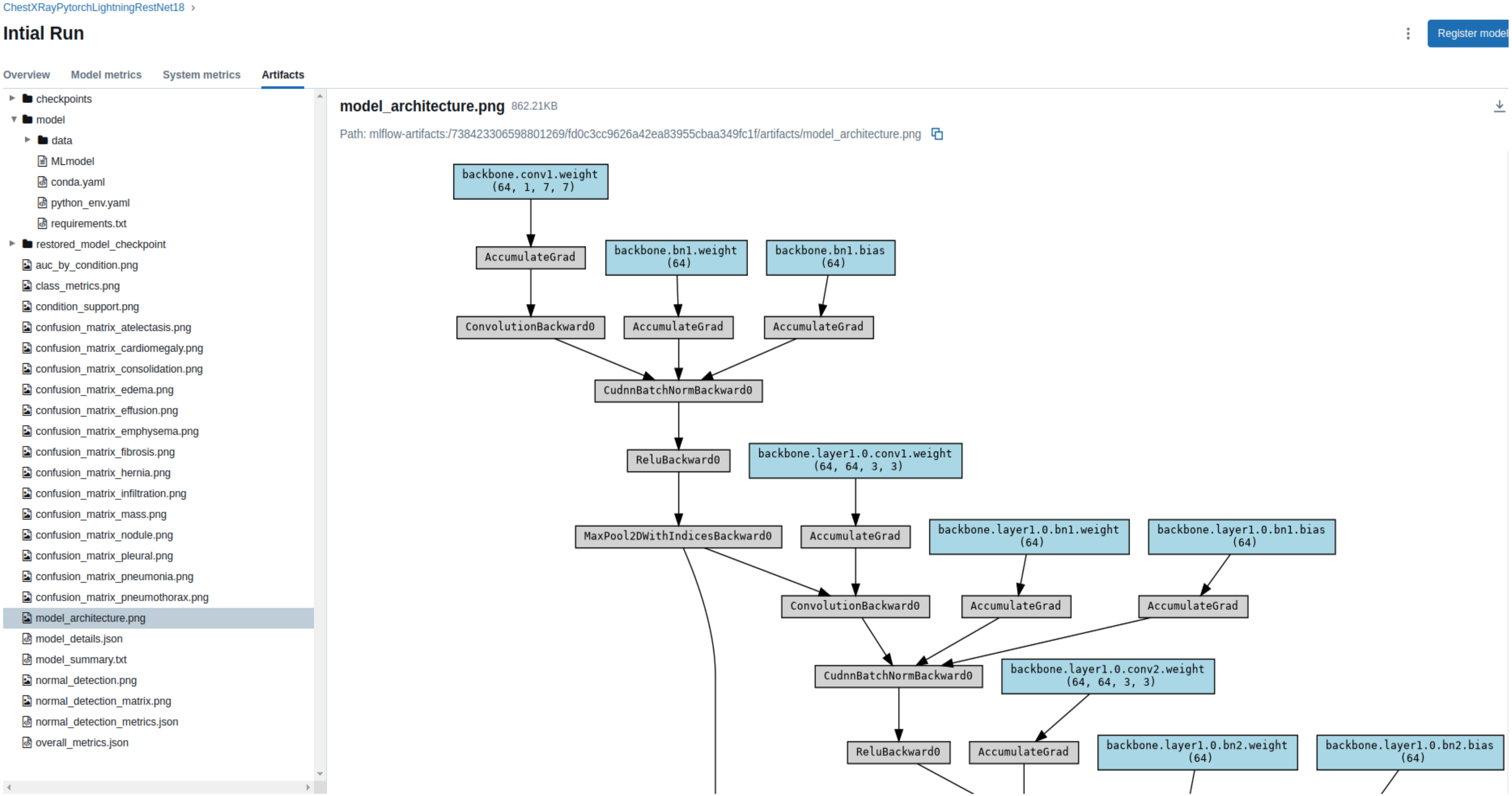
Intial Run

- ▸ checkpoints
- ▾ model
  - ▸ data
  - MLmodel
  - conda.yaml
  - python_env.yaml
  - requirements.txt
- ▸ restored_model_checkpoint
- auc_by_condition.png
- class_metrics.png
- condition_support.png
- confusion_matrix_atelectasis.png
- confusion_matrix_cardiomegaly.png
- confusion_matrix_consolidation.png
- confusion_matrix_edema.png
- confusion_matrix_effusion.png
- confusion_matrix_emphysema.png
- confusion_matrix_fibrosis.png
- confusion_matrix_hernia.png
- confusion_matrix_infiltration.png
- confusion_matrix_mass.png
- confusion_matrix_nodule.png
- confusion_matrix_pleural.png
- confusion_matrix_pneumonia.png
- confusion_matrix_pneumothorax.png
- model_architecture.png
- model_details.json
- model_summary.txt
- normal_detection.png
- normal_detection_matrix.png
- normal_detection_metrics.json
- overall_metrics.json

model_summary.txt 6.35KB

Path: mlflow-artifacts:/738423306598801269/fd0c3cc9626a42ea83955cbaa349fc1f/artifacts/model_summary.txt

```
   | Name                           | Type                | Params | Mode
--------------------------------------------------------------------------------
0  | train_metrics                  | MetricCollection    | 0      | train
1  | train_metrics.accuracy         | MultilabelAccuracy  | 0      | train
2  | train_metrics.f1_score         | MultilabelF1Score   | 0      | train
3  | train_metrics.precision        | MultilabelPrecision | 0      | train
4  | val_metrics                    | MetricCollection    | 0      | train
5  | val_metrics.accuracy           | MultilabelAccuracy  | 0      | train
6  | val_metrics.f1_score           | MultilabelF1Score   | 0      | train
7  | val_metrics.precision          | MultilabelPrecision | 0      | train
8  | test_metrics                   | MetricCollection    | 0      | train
9  | test_metrics.accuracy          | MultilabelAccuracy  | 0      | train
10 | test_metrics.f1_score          | MultilabelF1Score   | 0      | train
11 | test_metrics.precision         | MultilabelPrecision | 0      | train
12 | backbone                       | ResNet              | 11.2 M | train
13 | backbone.conv1                 | Conv2d              | 3.1 K  | train
14 | backbone.bn1                   | BatchNorm2d         | 128    | train
15 | backbone.relu                  | ReLU                | 0      | train
16 | backbone.maxpool               | MaxPool2d           | 0      | train
17 | backbone.layer1                | Sequential          | 147 K  | train
18 | backbone.layer1.0              | BasicBlock          | 74.0 K | train
19 | backbone.layer1.0.conv1        | Conv2d              | 36.9 K | train
20 | backbone.layer1.0.bn1          | BatchNorm2d         | 128    | train
21 | backbone.layer1.0.relu         | ReLU                | 0      | train
22 | backbone.layer1.0.conv2        | Conv2d              | 36.9 K | train
23 | backbone.layer1.0.bn2          | BatchNorm2d         | 128    | train
24 | backbone.layer1.1              | BasicBlock          | 74.0 K | train
25 | backbone.layer1.1.conv1        | Conv2d              | 36.9 K | train
26 | backbone.layer1.1.bn1          | BatchNorm2d         | 128    | train
27 | backbone.layer1.1.relu         | ReLU                | 0      | train
28 | backbone.layer1.1.conv2        | Conv2d              | 36.9 K | train
29 | backbone.layer1.1.bn2          | BatchNorm2d         | 128    | train
30 | backbone.layer2                | Sequential          | 525 K  | train
31 | backbone.layer2.0              | BasicBlock          | 230 K  | train
32 | backbone.layer2.0.conv1        | Conv2d              | 73.7 K | train
33 | backbone.layer2.0.bn1          | BatchNorm2d         | 256    | train
34 | backbone.layer2.0.relu         | ReLU                | 0      | train
35 | backbone.layer2.0.conv2        | Conv2d              | 147 K  | train
```



Intial Run

- checkpoints
- model
  - data
  - MLmodel
  - conda.yaml
  - python_env.yaml
  - requirements.txt
- restored_model_checkpoint
- auc_by_condition.png
- class_metrics.png
- condition_support.png
- confusion_matrix_atelectasis.png
- confusion_matrix_cardiomegaly.png
- confusion_matrix_consolidation.png
- confusion_matrix_edema.png
- confusion_matrix_effusion.png
- confusion_matrix_emphysema.png
- confusion_matrix_fibrosis.png
- confusion_matrix_hernia.png
- confusion_matrix_infiltration.png
- confusion_matrix_mass.png
- confusion_matrix_nodule.png
- confusion_matrix_pleural.png
- confusion_matrix_pneumonia.png
- confusion_matrix_pneumothorax.png
- model_architecture.png
- model_details.json
- model_summary.txt
- normal_detection.png
- normal_detection_matrix.png
- normal_detection_metrics.json
- overall_metrics.json

model_architecture.png 862.21KB

Path: mlflow-artifacts:/738423306598801269/fd0c3cc9626a42ea83955cbaa349fc1f/artifacts/model_architecture.png

backbone.conv1.weight (64, 1, 7, 7)
AccumulateGrad
backbone.bn1.weight (64)
backbone.bn1.bias (64)
ConvolutionBackward0
AccumulateGrad
AccumulateGrad
CudnnBatchNormBackward0
ReluBackward0
backbone.layer1.0.conv1.weight (64, 64, 3, 3)
MaxPool2DWithIndicesBackward0
AccumulateGrad
backbone.layer1.0.bn1.weight (64)
backbone.layer1.0.bn1.bias (64)
ConvolutionBackward0
AccumulateGrad
AccumulateGrad
CudnnBatchNormBackward0
backbone.layer1.0.conv2.weight (64, 64, 3, 3)
ReluBackward0
AccumulateGrad
backbone.layer1.0.bn2.weight (64)
backbone.layer1.0.bn2.bias (64)

Thank You

Questions?

Contact: piotr.gryko@gmail.com