In [1]:
```
import os
import pandas as pd
import numpy as np
import tensorflow as tf
import warnings
from sklearn.model_selection import train_test_split
import importlib
import warnings
import urllib3
warnings.filterwarnings('ignore')
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'  # Suppress TensorFlow logging
```

1. Create Directories and Load Data

In [2]:
```
import config
importlib.reload(config)

from config import DatasetConfig

# Create necessary directories
os.makedirs(DatasetConfig.PATHS['results'], exist_ok=True)
os.makedirs('model', exist_ok=True)

# Load the annotations
print("Loading annotations...")
annotations = pd.read_csv(DatasetConfig.PATHS['csv'])
print("Annotations shape:", annotations.shape)
```

```
Loading annotations...
Annotations shape: (5758, 2)
```

2. Analyze Dataset

In [3]:
```
import data_analyzer
importlib.reload(data_analyzer)

from data_analyzer import DataAnalyzer

analyzer = DataAnalyzer(annotations, DatasetConfig.PATHS['results'])
analyzer.analyze_data()
```

Dataset Overview

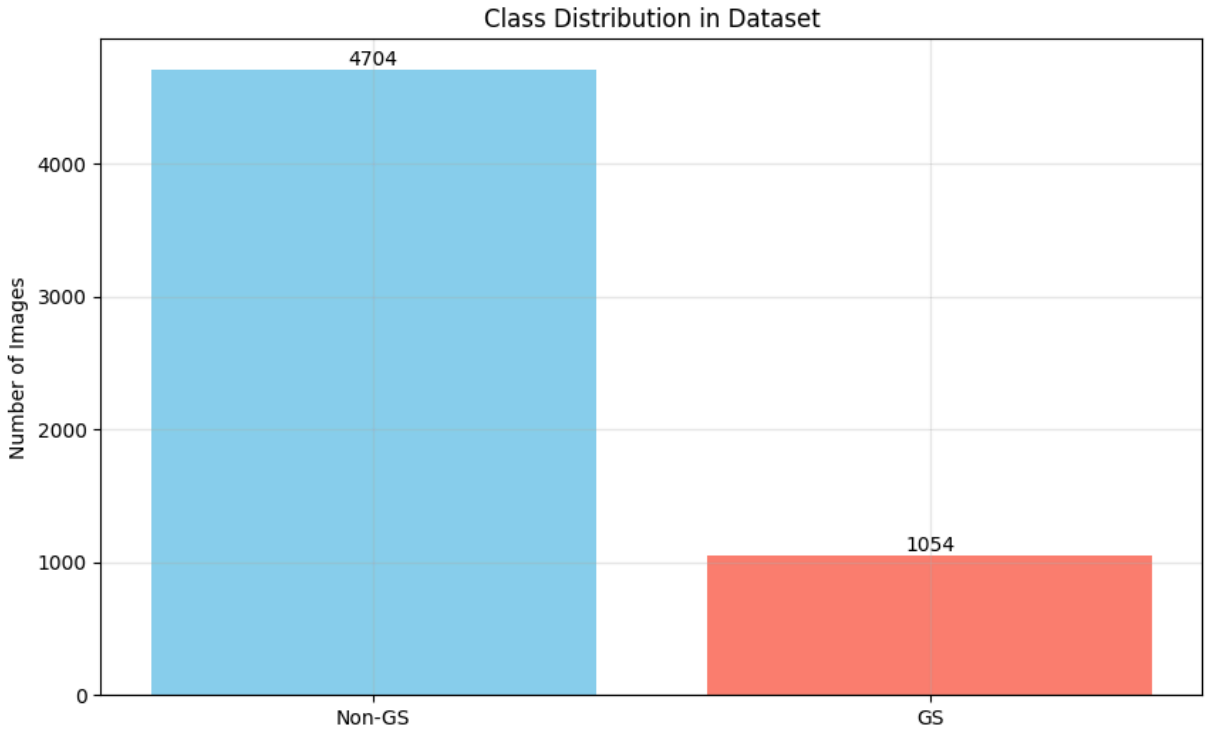
Dataset Statistics

	ground truth
count	5758.000000
mean	0.183050
std	0.386741
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

Category Distribution

```
ground truth
0    4704
1    1054
Name: count, dtype: int64
```

Class Distribution Analysis



Class Distribution Statistics:

- Globally Sclerotic: 1054 images
- Non-Globally Sclerotic: 4704 images
- Ratio (GS:Non-GS): 1:4.46

Image Properties Analysis

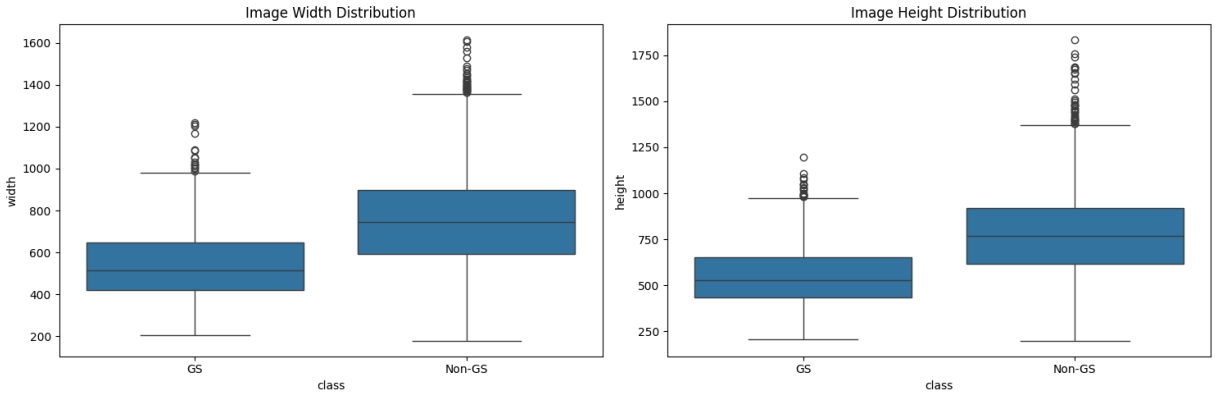
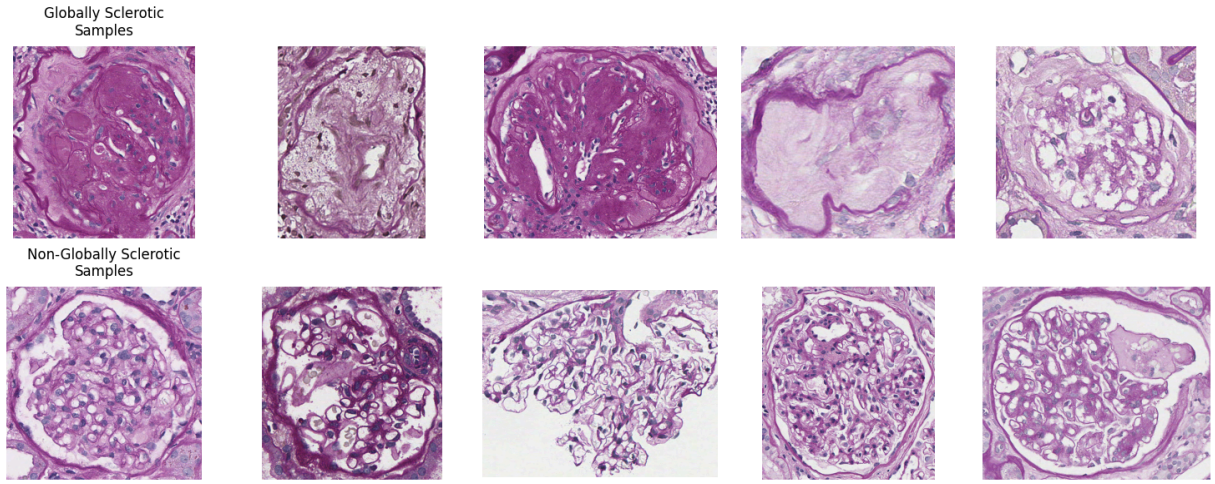


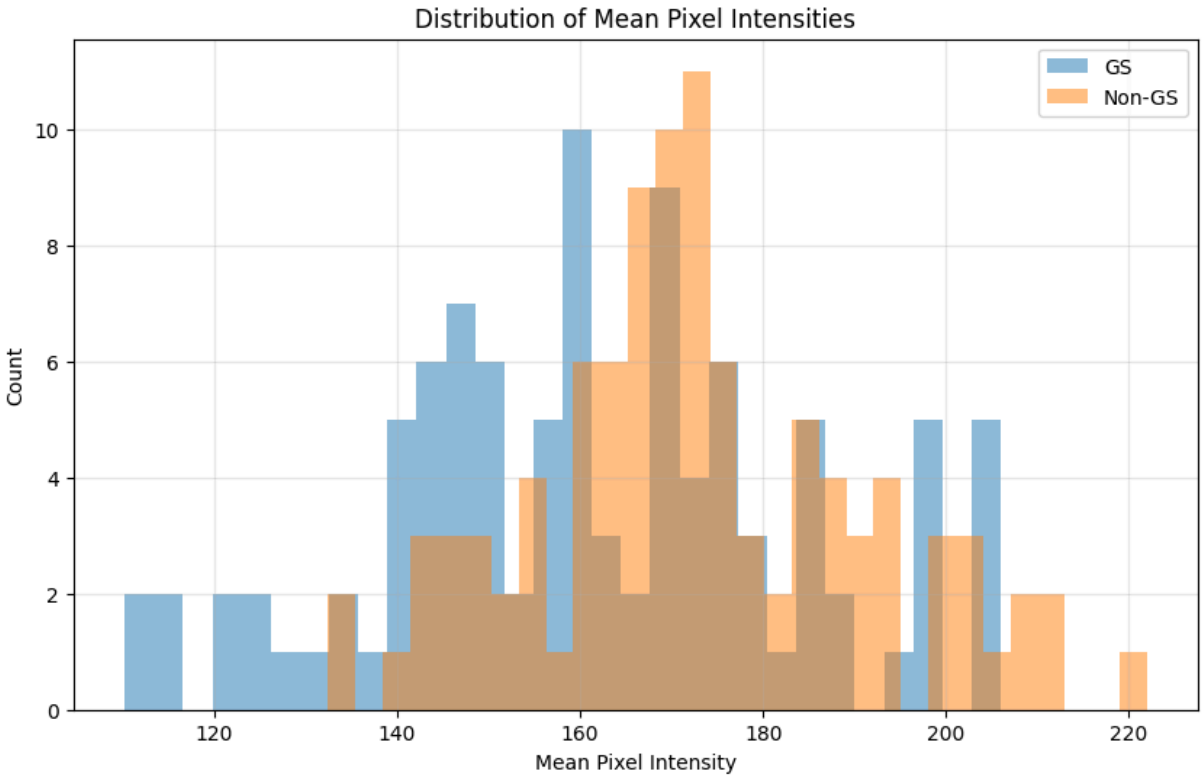
Image Properties Summary:

	width			height			size_kb			chan
	mean	min	max	mean	min	max	mean	min	max	
class										
GS	541.36	205.00	1220.00	555.25	209.00	1196.00	683.92	118.04	2616.29	
Non-GS	748.86	176.00	1616.00	772.68	197.00	1834.00	1286.38	105.64	5091.83	

Sample Images



Pixel Intensity Analysis



Pixel Intensity Statistics:

- GS Mean: 161.11 ± 22.66
- Non-GS Mean: 172.93 ± 18.18

3. Split Dataset and Create Data Generators

In [4]:
```
import data_generator
importlib.reload(data_generator)

from data_generator import GlomeruliDataGenerator

train_val_df, test_df = train_test_split(annotations, test_size=0.2, random_
train_df, val_df = train_test_split(train_val_df, test_size=0.25, random_sta

print("Dataset splits:")
print(f"Training samples: {len(train_df)}")
print(f"Validation samples: {len(val_df)}")
print(f"Test samples: {len(test_df)}")
```

```
Dataset splits:
Training samples: 3454
Validation samples: 1152
Test samples: 1152
```

In [5]:
```
# Create data generators for each set
train_generator = GlomeruliDataGenerator(
    train_df,
    DatasetConfig.PATHS['base'],
```

```
    batch_size=DatasetConfig.TRAINING['batch_size'],
    image_size=DatasetConfig.TRAINING['image_size']
)

val_generator = GlomeruliDataGenerator(
    val_df,
    DatasetConfig.PATHS['base'],
    batch_size=DatasetConfig.TRAINING['batch_size'],
    image_size=DatasetConfig.TRAINING['image_size'],
    shuffle=False
)

test_generator = GlomeruliDataGenerator(
    test_df,
    DatasetConfig.PATHS['base'],
    batch_size=DatasetConfig.TRAINING['batch_size'],
    image_size=DatasetConfig.TRAINING['image_size'],
    shuffle=False
)

print("Data generators created successfully")
```

```
Data generators created successfully
```

4. Create and Compile Model

In [6]:

```
import resnet_model_builder
importlib.reload(resnet_model_builder)

from resnet_model_builder import ModelBuilder

# Create model with your input shape
model = ModelBuilder.create_model(input_shape=(224, 224, 3))

# Print model summary to verify architecture
model.summary()
```

```
WARNING:absl:At this time, the v2.11+ optimizer `tf.keras.optimizers.Adam` r
uns slowly on M1/M2 Macs, please use the legacy Keras optimizer instead, loc
ated at `tf.keras.optimizers.legacy.Adam`.
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 7, 7, 2048)        23587712

 global_average_pooling2d (  (None, 2048)              0
 GlobalAveragePooling2D)

 dense (Dense)               (None, 512)               1049088

 batch_normalization (Batch  (None, 512)               2048
 Normalization)

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 256)               131328

 batch_normalization_1 (Bat  (None, 256)               1024
 chNormalization)

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 2)                 514

=================================================================
Total params: 24771714 (94.50 MB)
Trainable params: 15632642 (59.63 MB)
Non-trainable params: 9139072 (34.86 MB)
_________________________________________________________________
```

5. Train Model

In [7]:

```
import warnings
warnings.filterwarnings('ignore')
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

import resnet_model_trainer
importlib.reload(resnet_model_trainer)

from resnet_model_trainer import ModelTrainer

# Your training code
model = ModelBuilder.create_model(input_shape=(224, 224, 3))
trainer = ModelTrainer(model, train_generator, val_generator, results_dir=Da
history = trainer.train()
```

```
WARNING:absl:At this time, the v2.11+ optimizer `tf.keras.optimizers.Adam` r
uns slowly on M1/M2 Macs, please use the legacy Keras optimizer instead, loc
ated at `tf.keras.optimizers.legacy.Adam`.
```

```
Epoch 1/25
108/108 [==============================] - 181s 2s/step - loss: 0.6643 - acc
uracy: 0.7415 - auc: 0.7970 - val_loss: 0.4269 - val_accuracy: 0.8585 - val_
auc: 0.9525 - lr: 1.0000e-04
Epoch 2/25
108/108 [==============================] - 188s 2s/step - loss: 0.4756 - acc
uracy: 0.8332 - auc: 0.8881 - val_loss: 0.2773 - val_accuracy: 0.9036 - val_
auc: 0.9578 - lr: 1.0000e-04
Epoch 3/25
108/108 [==============================] - 188s 2s/step - loss: 0.3387 - acc
uracy: 0.9001 - auc: 0.9399 - val_loss: 1.1733 - val_accuracy: 0.8429 - val_
auc: 0.8346 - lr: 1.0000e-04
Epoch 4/25
108/108 [==============================] - 197s 2s/step - loss: 0.3401 - acc
uracy: 0.9042 - auc: 0.9392 - val_loss: 0.3414 - val_accuracy: 0.8628 - val_
auc: 0.9333 - lr: 1.0000e-04
Epoch 5/25
108/108 [==============================] - 191s 2s/step - loss: 0.2584 - acc
uracy: 0.9239 - auc: 0.9604 - val_loss: 0.2307 - val_accuracy: 0.9210 - val_
auc: 0.9704 - lr: 1.0000e-04
Epoch 6/25
108/108 [==============================] - 182s 2s/step - loss: 0.2626 - acc
uracy: 0.9186 - auc: 0.9588 - val_loss: 0.2312 - val_accuracy: 0.9158 - val_
auc: 0.9671 - lr: 1.0000e-04
Epoch 7/25
108/108 [==============================] - 185s 2s/step - loss: 0.2653 - acc
uracy: 0.9224 - auc: 0.9567 - val_loss: 0.2912 - val_accuracy: 0.9288 - val_
auc: 0.9613 - lr: 1.0000e-04
Epoch 8/25
108/108 [==============================] - 189s 2s/step - loss: 0.2069 - acc
uracy: 0.9346 - auc: 0.9731 - val_loss: 0.3472 - val_accuracy: 0.9227 - val_
auc: 0.9550 - lr: 1.0000e-04
Epoch 9/25
108/108 [==============================] - 183s 2s/step - loss: 0.1542 - acc
uracy: 0.9537 - auc: 0.9840 - val_loss: 0.1995 - val_accuracy: 0.9132 - val_
auc: 0.9798 - lr: 1.0000e-05
Epoch 10/25
108/108 [==============================] - 187s 2s/step - loss: 0.1523 - acc
uracy: 0.9522 - auc: 0.9842 - val_loss: 0.1696 - val_accuracy: 0.9366 - val_
auc: 0.9815 - lr: 1.0000e-05
Epoch 11/25
108/108 [==============================] - 191s 2s/step - loss: 0.1520 - acc
uracy: 0.9543 - auc: 0.9840 - val_loss: 0.1734 - val_accuracy: 0.9349 - val_
auc: 0.9823 - lr: 1.0000e-05
Epoch 12/25
108/108 [==============================] - 188s 2s/step - loss: 0.1203 - acc
uracy: 0.9635 - auc: 0.9895 - val_loss: 0.1690 - val_accuracy: 0.9453 - val_
auc: 0.9835 - lr: 1.0000e-05
Epoch 13/25
108/108 [==============================] - 183s 2s/step - loss: 0.1332 - acc
uracy: 0.9615 - auc: 0.9883 - val_loss: 0.2564 - val_accuracy: 0.9462 - val_
auc: 0.9712 - lr: 1.0000e-05
Epoch 14/25
108/108 [==============================] - 183s 2s/step - loss: 0.1123 - acc
uracy: 0.9647 - auc: 0.9907 - val_loss: 0.2220 - val_accuracy: 0.9410 - val_
auc: 0.9714 - lr: 1.0000e-05
```

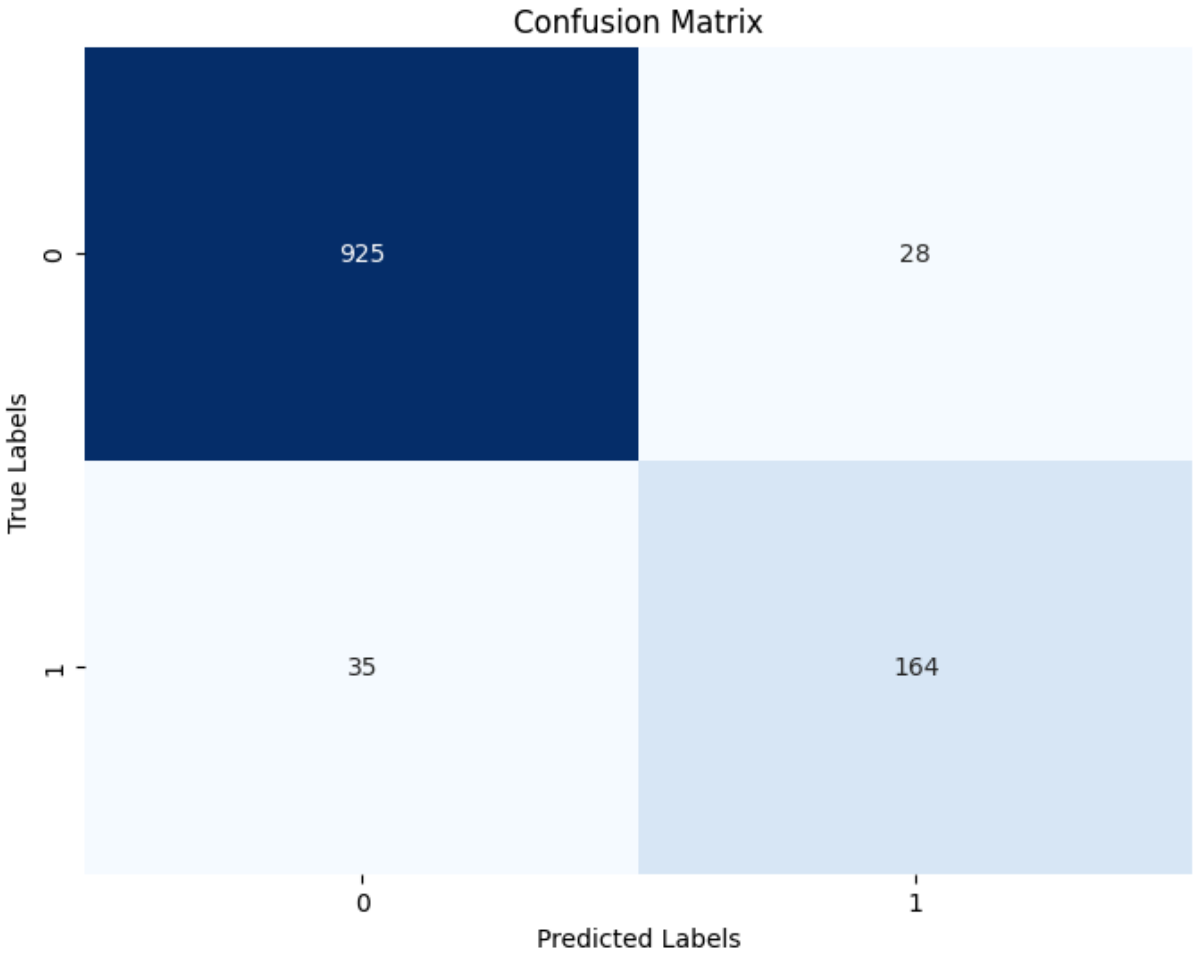
```
Epoch 15/25
108/108 [==============================] - 190s 2s/step - loss: 0.1205 - acc
uracy: 0.9641 - auc: 0.9900 - val_loss: 0.1742 - val_accuracy: 0.9392 - val_
auc: 0.9782 - lr: 1.0000e-05
Epoch 16/25
108/108 [==============================] - 193s 2s/step - loss: 0.1158 - acc
uracy: 0.9638 - auc: 0.9909 - val_loss: 0.1667 - val_accuracy: 0.9453 - val_
auc: 0.9817 - lr: 1.0000e-06
Epoch 17/25
108/108 [==============================] - 193s 2s/step - loss: 0.0922 - acc
uracy: 0.9722 - auc: 0.9944 - val_loss: 0.1681 - val_accuracy: 0.9444 - val_
auc: 0.9817 - lr: 1.0000e-06
Epoch 18/25
108/108 [==============================] - 188s 2s/step - loss: 0.1181 - acc
uracy: 0.9690 - auc: 0.9898 - val_loss: 0.1617 - val_accuracy: 0.9453 - val_
auc: 0.9825 - lr: 1.0000e-06
Epoch 19/25
108/108 [==============================] - 187s 2s/step - loss: 0.0987 - acc
uracy: 0.9722 - auc: 0.9933 - val_loss: 0.1642 - val_accuracy: 0.9453 - val_
auc: 0.9824 - lr: 1.0000e-06
Epoch 20/25
108/108 [==============================] - 187s 2s/step - loss: 0.1010 - acc
uracy: 0.9664 - auc: 0.9934 - val_loss: 0.1592 - val_accuracy: 0.9418 - val_
auc: 0.9831 - lr: 1.0000e-06
Epoch 21/25
108/108 [==============================] - 185s 2s/step - loss: 0.1045 - acc
uracy: 0.9699 - auc: 0.9926 - val_loss: 0.1597 - val_accuracy: 0.9427 - val_
auc: 0.9832 - lr: 1.0000e-06
Epoch 22/25
108/108 [==============================] - 196s 2s/step - loss: 0.0855 - acc
uracy: 0.9760 - auc: 0.9953 - val_loss: 0.1638 - val_accuracy: 0.9444 - val_
auc: 0.9824 - lr: 1.0000e-06
Epoch 23/25
108/108 [==============================] - 194s 2s/step - loss: 0.1110 - acc
uracy: 0.9705 - auc: 0.9912 - val_loss: 0.1675 - val_accuracy: 0.9462 - val_
auc: 0.9819 - lr: 1.0000e-06
Epoch 24/25
108/108 [==============================] - 196s 2s/step - loss: 0.0880 - acc
uracy: 0.9719 - auc: 0.9951 - val_loss: 0.1734 - val_accuracy: 0.9470 - val_
auc: 0.9811 - lr: 1.0000e-06
Epoch 25/25
108/108 [==============================] - 191s 2s/step - loss: 0.0909 - acc
uracy: 0.9742 - auc: 0.9948 - val_loss: 0.1628 - val_accuracy: 0.9418 - val_
auc: 0.9818 - lr: 1.0000e-06
```

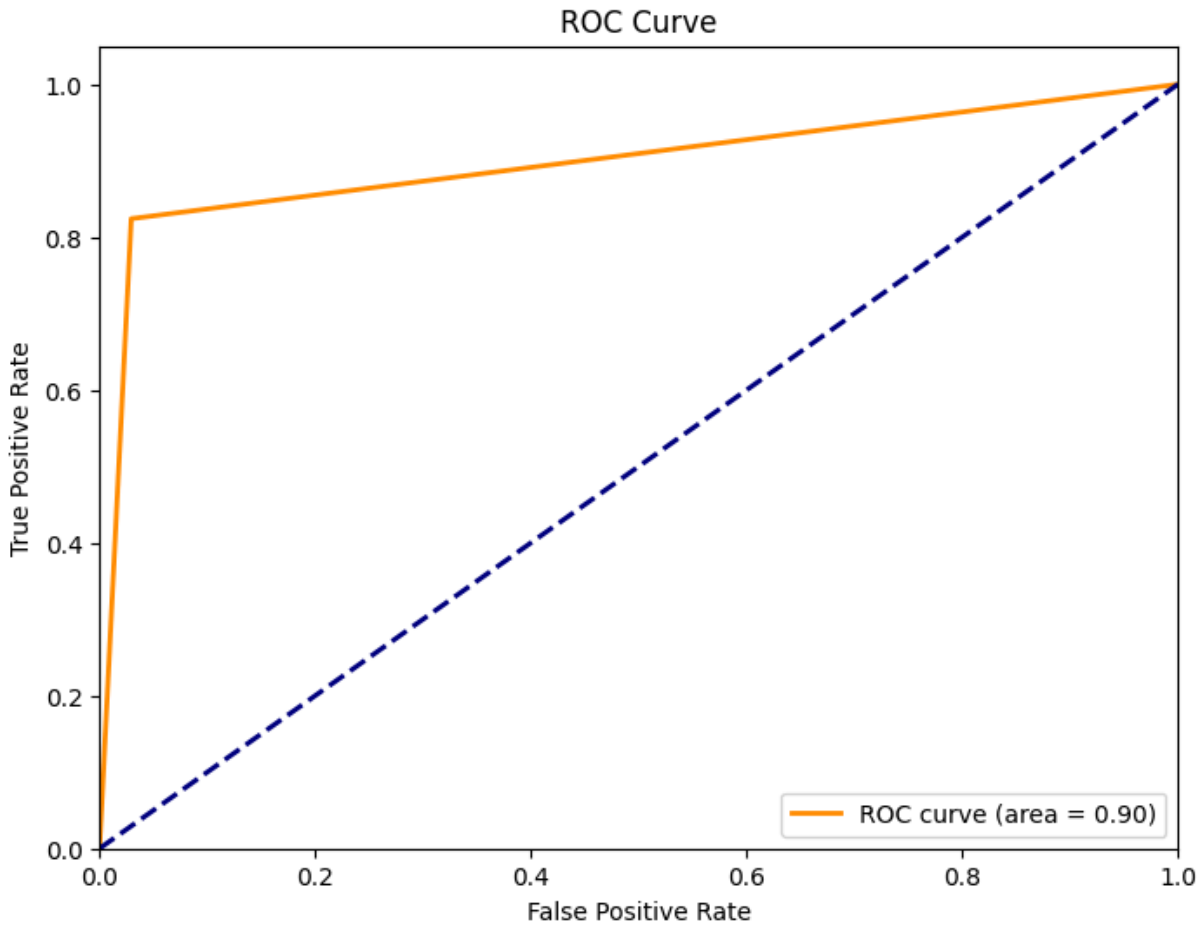
6. Evaluate Model Performance

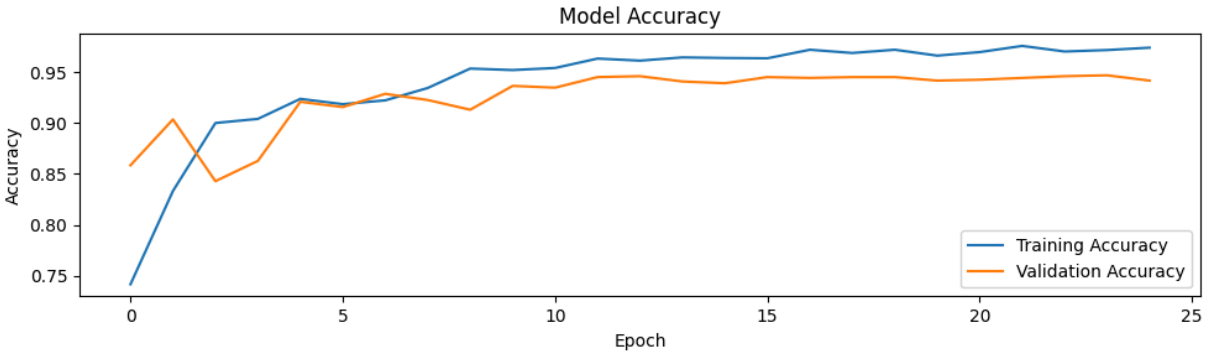
In [14]:
```python
import model_evaluator
importlib.reload(model_evaluator)

from model_evaluator import ModelEvaluator

# Initialize evaluator and perform evaluation
print("Evaluating model performance...")
```

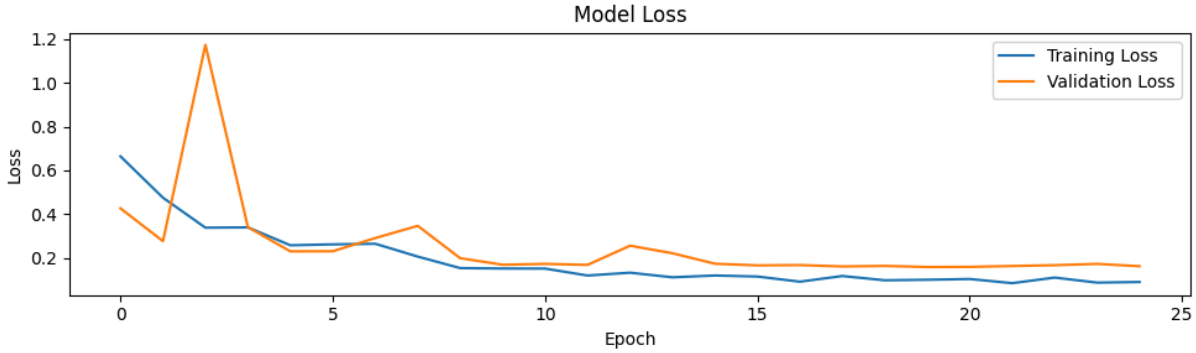
```
evaluator = ModelEvaluator(model, test_generator, DatasetConfig.PATHS['resul
evaluator.evaluate(history)
```

```
Evaluating model performance...
1/1 [==============================] - 1s 788ms/step
1/1 [==============================] - 1s 713ms/step
1/1 [==============================] - 1s 707ms/step
1/1 [==============================] - 1s 720ms/step
1/1 [==============================] - 1s 711ms/step
1/1 [==============================] - 1s 720ms/step
1/1 [==============================] - 1s 712ms/step
1/1 [==============================] - 1s 711ms/step
1/1 [==============================] - 1s 697ms/step
1/1 [==============================] - 1s 707ms/step
1/1 [==============================] - 1s 724ms/step
1/1 [==============================] - 1s 723ms/step
1/1 [==============================] - 1s 718ms/step
1/1 [==============================] - 1s 711ms/step
1/1 [==============================] - 1s 705ms/step
1/1 [==============================] - 1s 709ms/step
1/1 [==============================] - 1s 717ms/step
1/1 [==============================] - 1s 706ms/step
1/1 [==============================] - 1s 708ms/step
1/1 [==============================] - 1s 728ms/step
1/1 [==============================] - 1s 720ms/step
1/1 [==============================] - 1s 712ms/step
1/1 [==============================] - 1s 705ms/step
1/1 [==============================] - 1s 732ms/step
1/1 [==============================] - 1s 706ms/step
1/1 [==============================] - 1s 707ms/step
1/1 [==============================] - 1s 690ms/step
1/1 [==============================] - 1s 730ms/step
1/1 [==============================] - 1s 729ms/step
1/1 [==============================] - 1s 728ms/step
1/1 [==============================] - 1s 740ms/step
1/1 [==============================] - 1s 728ms/step
1/1 [==============================] - 1s 731ms/step
1/1 [==============================] - 1s 713ms/step
1/1 [==============================] - 1s 722ms/step
1/1 [==============================] - 1s 719ms/step
Accuracy: 0.9453
Precision: 0.8542
Recall: 0.8241
F1 Score: 0.8389
```


Confusion Matrix
925
28
35
164
0
1
True Labels
0
1
Predicted Labels







7. Evaluation on New Dataset

In [16]:
```
import evaluation
importlib.reload(evaluation)
```

```
from evaluation import ModelPredictor

# Define paths
model_path = os.path.join('model', 'best_model.keras')
evaluation_dir = '/Users/durgasrithadongla/Desktop/final_glomeruli/evaluatic

# Create predictor and run evaluation
predictor = ModelPredictor(
    model_path=model_path,
    evaluation_dir=evaluation_dir
)
predictor.predict_images()
```

```
WARNING:absl:At this time, the v2.11+ optimizer `tf.keras.optimizers.Adam` r
uns slowly on M1/M2 Macs, please use the legacy Keras optimizer instead, loc
ated at `tf.keras.optimizers.legacy.Adam`.
Found 4 images to evaluate
Results saved to evaluation.csv
```