In [1]:
```python
## Preparation

# Import the necessary libraries
import torch
import torch.nn as nn
from sklearn.model_selection import train_test_split
from torchvision import models, datasets, transforms
from torch.utils.data import DataLoader, Subset
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classificatio
import matplotlib.pyplot as plt
import numpy as np
import os
import optuna
import time

from activation import Swish, ResidualBlock

## Global Var

dataset_path = os.getcwd() + '/dataset/classified'

class_labels = ["Normal", "Cataract"]
```

In [2]:
```python
print(f"Is using CUDA? {torch.cuda.is_available()}")  # Should return True if CUDA
print(torch.version.cuda)  # Check the CUDA version PyTorch is using
print(torch.cuda.current_device()) # Check CUDA device used
```

```
Is using CUDA? True
12.6
0
```

In [3]:
```python
# Augmentation
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.RandomRotation(15),
    transforms.RandomHorizontalFlip(p=0.5),
    transforms.RandomCrop(224, padding=4),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])  # Normalize
])

# Load Dataset
ds = datasets.ImageFolder(root=dataset_path, transform=transform)

indices = list(range(len(ds)))
# labels = [ds.targets[i] for i in indices]

# Split into train and test dataset
train_indices, test_indices = train_test_split(indices, test_size=0.2, random_state

train_ds = Subset(ds, train_indices)
test_ds = Subset(ds, test_indices)

train_loader = DataLoader(train_ds, batch_size=128, shuffle=True, pin_memory=True,
```

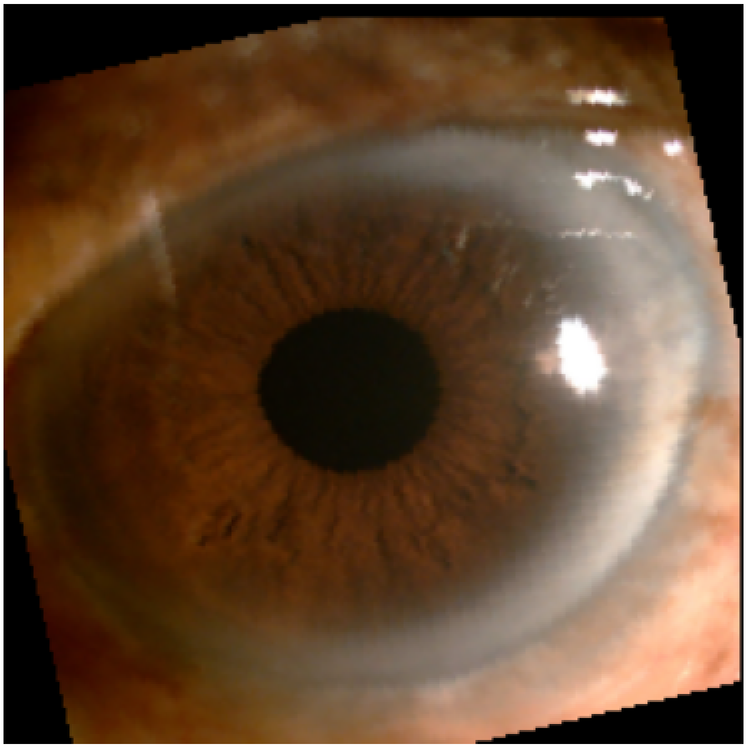
```
test_loader = DataLoader(test_ds, batch_size=128, shuffle=False, pin_memory=True, n

total_samples  = len(train_ds) + len(test_ds)

print(f"Train size: {(len(train_ds) / total_samples) * 100:.2f}%, Test size: {(len(
print(f"Total samples: {total_samples}, Train size: {len(train_ds)}, Test size: {le
```

```
Train size: 79.98%, Test size: 20.02%
Total samples: 1159, Train size: 927, Test size: 232
```

In [4]:

```
def denormalize(tensor, mean=None, std=None):
    if std is None:
        std = [0.5, 0.5, 0.5]
    if mean is None:
        mean = [0.5, 0.5, 0.5]
    mean = torch.tensor(mean).view(3, 1, 1)
    std = torch.tensor(std).view(3, 1, 1)
    return tensor * std + mean  # Reverse normalization

# Get a batch of images
dataiter = iter(train_loader)
images, labels = next(dataiter)

# Select one image
img = images[0]
label = labels[0].item()

# Denormalize image
img = denormalize(img)

# Convert from Tensor (C, H, W) to NumPy (H, W, C)
img = np.transpose(img.numpy(), (1, 2, 0))

# Plot the image
plt.imshow(img)
plt.title(f"Label: {class_labels[label]}")  # Display label
plt.axis("off")
plt.show()
```

Label: Normal



In [5]:

```python
# Optuna Hyperparameter

best_trial = None
best_model = None

torch.backends.cudnn.benchmark = True

device_name = "cuda" if torch.cuda.is_available() else "cpu"
use_amp = device_name == "cuda"


def objective(trial: optuna.Trial) -> float:
    device = torch.device(device_name)

    # Define hyperparameters
    lr = trial.suggest_float("lr", 1e-3, 1e-1, log=True)
    dropout_rate = trial.suggest_float("dropout", 0.3, 0.5)
    optimizer_name = trial.suggest_categorical("optimizer", ["AdamW", "SGD"])
    num_epochs = trial.suggest_int("num_epochs", 5, 15)

    model = models.efficientnet_b0(progress=True, weights=models.EfficientNet_B0_We

    for param in model.features[:-2].parameters():
        param.requires_grad = False

    number_of_features = model.classifier[1].in_features
    model.classifier = nn.Sequential(
        nn.Linear(number_of_features, 256),
        nn.BatchNorm1d(256),
        Swish(),
```

```
        ResidualBlock(256, 256),
        nn.Dropout(dropout_rate),

        ResidualBlock(256, 128),
        nn.Dropout(dropout_rate),

        ResidualBlock(128, 128),
        nn.Dropout(dropout_rate),

        ResidualBlock(128, 64),
        nn.Dropout(dropout_rate),

        nn.Linear(64, 1)
    ).to(device)

    model.to(device)

    # Define loss and optimizer
    criterion = nn.BCEWithLogitsLoss()

    optimizer = torch.optim.AdamW(model.parameters(), lr=lr, weight_decay=1e-4)
    if optimizer_name == "SGD":
        optimizer = torch.optim.SGD(model.parameters(), lr=lr, momentum=0.9)

    scheduler = torch.optim.lr_scheduler.CosineAnnealingLR(optimizer, T_max=10)

    train_losses = []
    train_accuracies = []

    scaler = torch.amp.GradScaler(device=device_name)

    # Training Loop
    for epoch in range(num_epochs):

        model.train()
        total_loss, correct, total = 0, 0, 0

        for images, labels in train_loader:
            images, labels = images.to(device), labels.float().to(device).unsqueeze

            optimizer.zero_grad()

            with torch.amp.autocast("cuda", enabled=use_amp):
                outputs = model(images)
                loss = criterion(outputs, labels)

            scaler.scale(loss).backward()
            scaler.step(optimizer)
            scaler.update()

            total_loss += loss.item()
            preds = (torch.sigmoid(outputs) > 0.5).float()
            correct += (preds == labels).sum().item()
            total += labels.size(0)

        scheduler.step()
```

```
        epoch_loss = total_loss / len(train_loader)
        epoch_acc = correct / total * 100

        train_losses.append(epoch_loss)
        train_accuracies.append(epoch_acc)

        print(f"Epoch [{epoch+1}/{num_epochs}], Loss: {epoch_loss:.4f}, Accuracy: {

        trial.set_user_attr("train_losses", train_losses)
        trial.set_user_attr("train_accuracies", train_accuracies)

        # Pruning: Stop bad trials early
        trial.report(epoch_acc, epoch)
        if trial.should_prune():
            raise optuna.exceptions.TrialPruned()

    # Evaluate Model
    model.eval()
    correct, total = 0, 0
    trial_preds = []
    trial_labels = []

    with torch.no_grad():
        for images, labels in test_loader:
            images, labels = images.to(device), labels.float().to(device).unsqueeze
            outputs = model(images)
            preds = (torch.sigmoid(outputs) > 0.5).float()

            correct += (preds == labels).sum().item()
            total += labels.size(0)

            trial_preds.extend(preds.cpu().numpy())
            trial_labels.extend(labels.cpu().numpy())

    test_acc = correct / total * 100
    print(f"🎯 Test Accuracy: {test_acc:.2f}%")

    # **Save best model globally**
    global best_model, best_trial

    try:
        best_trial = trial.study.best_trial
        best_value = trial.study.best_value
    except ValueError:
        best_value = float('-inf')

    if best_model is None or (best_trial is not None and test_acc > best_value):
        best_model = model.state_dict()

    best_acc = trial.study.user_attrs.get("best_accuracy", 0)

    if test_acc > best_acc:
        study.set_user_attr("best_accuracy", test_acc)

        trial_preds = np.array(trial_preds).flatten().tolist()
```

```python
        trial_labels = np.array(trial_labels).flatten().tolist()

        study.set_user_attr("best_preds", trial_preds)
        study.set_user_attr("best_labels", trial_labels)

    return test_acc
```

In [6]:

```python
study = optuna.create_study(
    direction="maximize",
    study_name=f"hyperparam cataract classifier_{int(time.time())}",
    pruner=optuna.pruners.MedianPruner(),
    storage="sqlite:///optuna.db",
    load_if_exists=True
)

study.optimize(
    objective,
    n_trials=50,
    n_jobs=4,
    show_progress_bar=True
)

acc = study.best_value

print("\nBest Hyperparameters:", study.best_params)
print("\nAccuracy:", acc)
```

```
[I 2025-03-13 11:32:40,450] A new study created in RDB with name: hyperparam catarac
t classifier_1741840359
  0%|          | 0/50 [00:00<?, ?it/s]
```

```
Epoch [1/10], Loss: nan, Accuracy: 49.19%
Epoch [1/13], Loss: 0.6537, Accuracy: 63.43%
Epoch [1/11], Loss: 0.4760, Accuracy: 76.38%
Epoch [1/10], Loss: 0.5714, Accuracy: 68.28%
Epoch [2/10], Loss: nan, Accuracy: 69.04%
Epoch [2/11], Loss: 0.3068, Accuracy: 89.21%
Epoch [2/13], Loss: 0.4138, Accuracy: 88.24%
Epoch [2/10], Loss: 0.3232, Accuracy: 89.64%
Epoch [3/10], Loss: nan, Accuracy: 73.03%
Epoch [3/11], Loss: 0.2068, Accuracy: 92.77%
Epoch [3/13], Loss: 0.3018, Accuracy: 92.23%
Epoch [3/10], Loss: 0.2286, Accuracy: 93.64%
Epoch [4/10], Loss: nan, Accuracy: 73.25%
Epoch [4/11], Loss: 0.1959, Accuracy: 92.45%
Epoch [4/13], Loss: 0.2552, Accuracy: 93.85%
Epoch [4/10], Loss: 0.1636, Accuracy: 94.93%
Epoch [5/10], Loss: nan, Accuracy: 77.02%
Epoch [5/11], Loss: 0.1643, Accuracy: 93.74%
Epoch [5/13], Loss: 0.2091, Accuracy: 94.61%
Epoch [5/10], Loss: 0.1517, Accuracy: 94.93%
Epoch [6/10], Loss: nan, Accuracy: 80.26%
Epoch [6/11], Loss: 0.1440, Accuracy: 95.25%
Epoch [6/13], Loss: 0.1714, Accuracy: 96.01%
Epoch [6/10], Loss: 0.1639, Accuracy: 94.50%
Epoch [7/10], Loss: nan, Accuracy: 83.50%
Epoch [7/13], Loss: 0.1753, Accuracy: 96.01%
Epoch [7/11], Loss: 0.1149, Accuracy: 95.58%
Epoch [7/10], Loss: 0.1401, Accuracy: 95.15%
Epoch [8/10], Loss: nan, Accuracy: 82.42%
Epoch [8/11], Loss: 0.1054, Accuracy: 95.36%Epoch [8/13], Loss: 0.1547, Accuracy: 9
6.55%

Epoch [8/10], Loss: 0.1281, Accuracy: 95.79%
Epoch [9/10], Loss: nan, Accuracy: 83.60%
Epoch [9/13], Loss: 0.1405, Accuracy: 96.76%
Epoch [9/11], Loss: 0.1149, Accuracy: 96.01%
Epoch [9/10], Loss: 0.1036, Accuracy: 96.98%
Epoch [10/10], Loss: nan, Accuracy: 84.90%
Epoch [10/13], Loss: 0.1364, Accuracy: 96.87%
Epoch [10/11], Loss: 0.0949, Accuracy: 96.44%
Epoch [10/10], Loss: 0.1071, Accuracy: 96.44%
🎯 Test Accuracy: 75.00%
[I 2025-03-13 11:39:22,012] Trial 3 finished with value: 75.0 and parameters: {'lr':
0.015891770629740936, 'dropout': 0.4409585661434964, 'optimizer': 'SGD', 'num_epoch
s': 10}. Best is trial 3 with value: 75.0.
Epoch [11/13], Loss: 0.1628, Accuracy: 96.76%
Epoch [11/11], Loss: 0.1299, Accuracy: 95.25%
🎯 Test Accuracy: 93.97%
[I 2025-03-13 11:39:32,445] Trial 1 finished with value: 93.96551724137932 and param
eters: {'lr': 0.003945523595412479, 'dropout': 0.41096327896191026, 'optimizer': 'Ad
amW', 'num_epochs': 10}. Best is trial 1 with value: 93.96551724137932.
Epoch [1/15], Loss: nan, Accuracy: 63.65%
Epoch [12/13], Loss: 0.1192, Accuracy: 98.17%
🎯 Test Accuracy: 93.97%
[I 2025-03-13 11:39:54,827] Trial 0 finished with value: 93.96551724137932 and param
eters: {'lr': 0.09355775354425618, 'dropout': 0.3640124818079397, 'optimizer': 'Adam
```

```
W', 'num_epochs': 11}. Best is trial 0 with value: 93.96551724137932.
Epoch [1/9], Loss: 0.6146, Accuracy: 66.56%
Epoch [2/15], Loss: nan, Accuracy: 79.29%
Epoch [13/13], Loss: 0.1284, Accuracy: 97.09%
Epoch [1/8], Loss: 0.7618, Accuracy: 41.32%
Epoch [2/9], Loss: 0.4594, Accuracy: 76.81%
Epoch [3/15], Loss: nan, Accuracy: 90.18%
🎯 Test Accuracy: 92.67%
[I 2025-03-13 11:40:58,093] Trial 2 finished with value: 92.67241379310344 and param
eters: {'lr': 0.002036571246773132, 'dropout': 0.42707677332951344, 'optimizer': 'Ad
amW', 'num_epochs': 13}. Best is trial 0 with value: 93.96551724137932.
Epoch [2/8], Loss: 0.7129, Accuracy: 49.62%
Epoch [3/9], Loss: 0.3071, Accuracy: 86.84%
Epoch [4/15], Loss: nan, Accuracy: 92.34%
Epoch [1/14], Loss: 0.4938, Accuracy: 74.22%
Epoch [3/8], Loss: 0.6811, Accuracy: 54.05%
Epoch [4/9], Loss: 0.2187, Accuracy: 91.05%
Epoch [5/15], Loss: nan, Accuracy: 93.74%
Epoch [2/14], Loss: 0.2491, Accuracy: 88.13%
Epoch [4/8], Loss: 0.6723, Accuracy: 59.01%
Epoch [5/9], Loss: 0.1801, Accuracy: 92.02%
Epoch [6/15], Loss: nan, Accuracy: 94.39%
Epoch [3/14], Loss: 0.2526, Accuracy: 90.72%
Epoch [5/8], Loss: 0.6570, Accuracy: 60.73%
Epoch [6/9], Loss: 0.1401, Accuracy: 94.17%
Epoch [7/15], Loss: nan, Accuracy: 95.04%
Epoch [4/14], Loss: 0.1659, Accuracy: 95.15%
Epoch [6/8], Loss: 0.6476, Accuracy: 64.83%
Epoch [7/9], Loss: 0.1744, Accuracy: 94.82%
Epoch [8/15], Loss: nan, Accuracy: 95.36%
Epoch [5/14], Loss: 0.1603, Accuracy: 93.96%
Epoch [7/8], Loss: 0.6336, Accuracy: 65.26%
Epoch [8/9], Loss: 0.1242, Accuracy: 94.82%
Epoch [9/15], Loss: nan, Accuracy: 95.90%
Epoch [6/14], Loss: 0.1308, Accuracy: 95.58%
Epoch [8/8], Loss: 0.6240, Accuracy: 66.77%
Epoch [9/9], Loss: 0.1137, Accuracy: 95.90%
Epoch [10/15], Loss: nan, Accuracy: 96.44%
🎯 Test Accuracy: 78.02%
[I 2025-03-13 11:44:48,020] Trial 6 finished with value: 78.01724137931035 and param
eters: {'lr': 0.0016267162256154537, 'dropout': 0.3449338994878817, 'optimizer': 'SG
D', 'num_epochs': 8}. Best is trial 0 with value: 93.96551724137932.
Epoch [7/14], Loss: 0.1308, Accuracy: 94.50%
🎯 Test Accuracy: 90.95%
[I 2025-03-13 11:45:05,822] Trial 5 finished with value: 90.94827586206897 and param
eters: {'lr': 0.07652929109466927, 'dropout': 0.4075213005173787, 'optimizer': 'SG
D', 'num_epochs': 9}. Best is trial 0 with value: 93.96551724137932.
Epoch [11/15], Loss: nan, Accuracy: 97.20%
Epoch [1/6], Loss: 0.6422, Accuracy: 60.73%
[I 2025-03-13 11:45:19,555] Trial 8 pruned.
Epoch [8/14], Loss: 0.1129, Accuracy: 95.15%
Epoch [1/7], Loss: 0.6679, Accuracy: 60.84%
[I 2025-03-13 11:45:39,696] Trial 9 pruned.
Epoch [12/15], Loss: nan, Accuracy: 96.55%
[I 2025-03-13 11:45:50,418] Trial 4 pruned.
Epoch [1/8], Loss: 0.5992, Accuracy: 68.28%
```

```
Epoch [9/14], Loss: 0.1079, Accuracy: 96.12%
Epoch [1/5], Loss: 0.6772, Accuracy: 57.82%
[I 2025-03-13 11:46:13,157] Trial 11 pruned.
Epoch [1/11], Loss: nan, Accuracy: 61.92%
[I 2025-03-13 11:46:23,353] Trial 12 pruned.
Epoch [2/8], Loss: 0.2983, Accuracy: 87.70%
Epoch [10/14], Loss: 0.0987, Accuracy: 97.41%
Epoch [1/12], Loss: 0.6372, Accuracy: 64.51%
[I 2025-03-13 11:46:47,016] Trial 13 pruned.
Epoch [1/12], Loss: nan, Accuracy: 68.18%
Epoch [3/8], Loss: 0.1958, Accuracy: 92.23%
Epoch [11/14], Loss: 0.0946, Accuracy: 96.01%
Epoch [1/13], Loss: 0.5324, Accuracy: 73.57%
Epoch [2/12], Loss: nan, Accuracy: 86.08%
Epoch [4/8], Loss: 0.2040, Accuracy: 92.45%
Epoch [12/14], Loss: 0.0951, Accuracy: 97.09%
[I 2025-03-13 11:47:36,761] Trial 7 pruned.
Epoch [2/13], Loss: 0.2812, Accuracy: 92.13%
Epoch [3/12], Loss: nan, Accuracy: 90.51%
Epoch [5/8], Loss: 0.1616, Accuracy: 94.07%
Epoch [1/10], Loss: 0.5089, Accuracy: 73.03%
Epoch [3/13], Loss: 0.2139, Accuracy: 92.56%
Epoch [4/12], Loss: nan, Accuracy: 92.02%
Epoch [6/8], Loss: 0.1521, Accuracy: 94.61%
Epoch [2/10], Loss: 0.2592, Accuracy: 90.18%
Epoch [4/13], Loss: 0.1725, Accuracy: 93.74%
Epoch [7/8], Loss: 0.1415, Accuracy: 94.93%
Epoch [5/12], Loss: nan, Accuracy: 92.34%
[I 2025-03-13 11:49:10,009] Trial 10 pruned.
[I 2025-03-13 11:49:10,019] Trial 14 pruned.
Epoch [3/10], Loss: 0.2158, Accuracy: 93.85%
Epoch [5/13], Loss: 0.1258, Accuracy: 95.90%
Epoch [1/10], Loss: nan, Accuracy: 78.86%
Epoch [1/10], Loss: 0.5798, Accuracy: 68.39%
Epoch [4/10], Loss: 0.1850, Accuracy: 93.31%
Epoch [6/13], Loss: 0.1234, Accuracy: 96.12%
Epoch [2/10], Loss: nan, Accuracy: 90.40%
Epoch [2/10], Loss: 0.2737, Accuracy: 91.59%
Epoch [5/10], Loss: 0.1406, Accuracy: 95.47%
Epoch [7/13], Loss: 0.0910, Accuracy: 97.20%
Epoch [3/10], Loss: 0.2090, Accuracy: 93.53%
Epoch [3/10], Loss: nan, Accuracy: 93.74%
Epoch [6/10], Loss: 0.1084, Accuracy: 96.12%
Epoch [8/13], Loss: 0.0829, Accuracy: 96.98%
Epoch [4/10], Loss: 0.2046, Accuracy: 91.80%
Epoch [4/10], Loss: nan, Accuracy: 94.07%
Epoch [7/10], Loss: 0.0896, Accuracy: 96.55%
Epoch [9/13], Loss: 0.0734, Accuracy: 97.41%
Epoch [5/10], Loss: nan, Accuracy: 94.61%
Epoch [5/10], Loss: 0.1740, Accuracy: 92.77%
Epoch [8/10], Loss: 0.0986, Accuracy: 97.20%
Epoch [10/13], Loss: 0.0651, Accuracy: 97.84%
Epoch [6/10], Loss: nan, Accuracy: 95.90%
Epoch [6/10], Loss: 0.1267, Accuracy: 95.04%
Epoch [9/10], Loss: 0.0754, Accuracy: 97.63%
Epoch [11/13], Loss: 0.0826, Accuracy: 96.66%
```

```
Epoch [7/10], Loss: 0.0998, Accuracy: 96.76%
Epoch [7/10], Loss: nan, Accuracy: 96.66%
Epoch [10/10], Loss: 0.0615, Accuracy: 98.27%
Epoch [12/13], Loss: 0.0831, Accuracy: 97.95%
[I 2025-03-13 11:53:29,102] Trial 15 pruned.
Epoch [8/10], Loss: 0.0877, Accuracy: 96.98%
Epoch [8/10], Loss: nan, Accuracy: 96.01%
🎯 Test Accuracy: 92.67%
[I 2025-03-13 11:53:41,511] Trial 16 finished with value: 92.67241379310344 and para
meters: {'lr': 0.006405507221462147, 'dropout': 0.30234245442985785, 'optimizer': 'A
damW', 'num_epochs': 10}. Best is trial 0 with value: 93.96551724137932.
Epoch [1/10], Loss: 0.6969, Accuracy: 53.07%
[I 2025-03-13 11:54:01,080] Trial 19 pruned.
Epoch [9/10], Loss: 0.0653, Accuracy: 98.17%
Epoch [9/10], Loss: nan, Accuracy: 96.98%
Epoch [1/11], Loss: 0.4647, Accuracy: 76.27%
Epoch [1/11], Loss: 0.5039, Accuracy: 77.35%
Epoch [10/10], Loss: nan, Accuracy: 97.41%
Epoch [10/10], Loss: 0.0871, Accuracy: 97.09%
Epoch [2/11], Loss: 0.2463, Accuracy: 90.51%
Epoch [2/11], Loss: 0.2277, Accuracy: 92.02%
🎯 Test Accuracy: 75.00%
[I 2025-03-13 11:55:16,177] Trial 17 finished with value: 75.0 and parameters: {'l
r': 0.006237881579923652, 'dropout': 0.30326686081704674, 'optimizer': 'AdamW', 'num
_epochs': 10}. Best is trial 0 with value: 93.96551724137932.
🎯 Test Accuracy: 95.26%
[I 2025-03-13 11:55:16,854] Trial 18 finished with value: 95.25862068965517 and para
meters: {'lr': 0.00803525510239778, 'dropout': 0.3233933125426155, 'optimizer': 'Ada
mW', 'num_epochs': 10}. Best is trial 18 with value: 95.25862068965517.
Epoch [3/11], Loss: 0.1839, Accuracy: 93.10%
Epoch [3/11], Loss: 0.1824, Accuracy: 92.23%
[I 2025-03-13 11:55:42,946] Trial 21 pruned.
Epoch [1/12], Loss: nan, Accuracy: 74.76%
Epoch [1/12], Loss: 0.4912, Accuracy: 75.84%
Epoch [4/11], Loss: 0.1626, Accuracy: 93.64%
Epoch [1/9], Loss: 0.5651, Accuracy: 68.72%
Epoch [2/12], Loss: nan, Accuracy: 89.00%
[I 2025-03-13 11:56:22,128] Trial 22 pruned.
Epoch [2/12], Loss: 0.2403, Accuracy: 90.83%
Epoch [5/11], Loss: 0.1668, Accuracy: 92.45%
[I 2025-03-13 11:56:28,418] Trial 20 pruned.
Epoch [2/9], Loss: 0.3489, Accuracy: 90.72%
Epoch [1/9], Loss: nan, Accuracy: 61.38%
[I 2025-03-13 11:56:55,767] Trial 25 pruned.
Epoch [3/12], Loss: 0.2125, Accuracy: 92.99%
Epoch [1/9], Loss: 0.5506, Accuracy: 75.19%
Epoch [3/9], Loss: 0.2488, Accuracy: 92.88%
Epoch [4/12], Loss: 0.2245, Accuracy: 92.77%
Epoch [1/9], Loss: nan, Accuracy: 68.93%
Epoch [2/9], Loss: 0.3062, Accuracy: 90.94%
Epoch [4/9], Loss: 0.2028, Accuracy: 94.07%
Epoch [5/12], Loss: 0.1505, Accuracy: 95.04%
Epoch [2/9], Loss: nan, Accuracy: 86.95%
[I 2025-03-13 11:58:05,196] Trial 27 pruned.
Epoch [3/9], Loss: 0.2008, Accuracy: 93.64%
Epoch [5/9], Loss: 0.1657, Accuracy: 95.47%
```

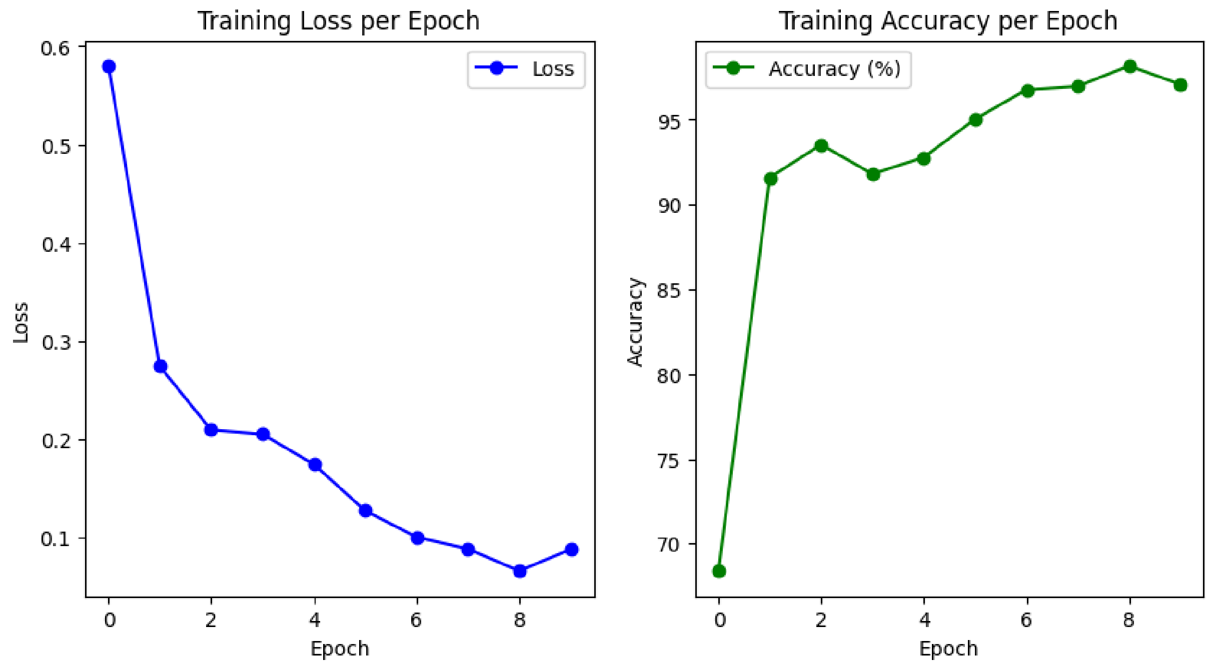
```
Epoch [6/12], Loss: 0.1036, Accuracy: 96.33%
Epoch [1/8], Loss: nan, Accuracy: 70.44%
Epoch [4/9], Loss: 0.1956, Accuracy: 93.31%
Epoch [6/9], Loss: 0.1206, Accuracy: 96.76%
Epoch [7/12], Loss: 0.0849, Accuracy: 97.09%
Epoch [2/8], Loss: nan, Accuracy: 89.64%
Epoch [5/9], Loss: 0.1571, Accuracy: 95.36%
Epoch [7/9], Loss: 0.1170, Accuracy: 96.98%
Epoch [8/12], Loss: 0.0933, Accuracy: 96.76%
Epoch [3/8], Loss: nan, Accuracy: 93.20%
Epoch [6/9], Loss: 0.1181, Accuracy: 96.12%
Epoch [8/9], Loss: 0.1038, Accuracy: 96.76%
Epoch [9/12], Loss: 0.0839, Accuracy: 96.76%
Epoch [4/8], Loss: nan, Accuracy: 93.64%
Epoch [7/9], Loss: 0.0967, Accuracy: 97.30%
Epoch [9/9], Loss: 0.0786, Accuracy: 98.27%
Epoch [10/12], Loss: 0.0859, Accuracy: 96.76%
Epoch [5/8], Loss: nan, Accuracy: 95.58%
Epoch [8/9], Loss: 0.0833, Accuracy: 97.73%
🎯 Test Accuracy: 93.53%
[I 2025-03-13 12:01:16,037] Trial 24 finished with value: 93.53448275862068 and para
meters: {'lr': 0.002934013686591769, 'dropout': 0.3258742763233171, 'optimizer': 'Ad
amW', 'num_epochs': 9}. Best is trial 18 with value: 95.25862068965517.
Epoch [6/8], Loss: nan, Accuracy: 96.01%
Epoch [11/12], Loss: 0.0906, Accuracy: 96.55%
Epoch [9/9], Loss: 0.0870, Accuracy: 97.73%
Epoch [1/8], Loss: 0.5246, Accuracy: 70.66%
Epoch [7/8], Loss: nan, Accuracy: 95.90%
Epoch [12/12], Loss: 0.0808, Accuracy: 97.63%
[I 2025-03-13 12:02:01,217] Trial 23 pruned.
🎯 Test Accuracy: 92.24%
[I 2025-03-13 12:02:05,127] Trial 26 finished with value: 92.24137931034483 and para
meters: {'lr': 0.003402045495039703, 'dropout': 0.36676754770394543, 'optimizer': 'A
damW', 'num_epochs': 9}. Best is trial 18 with value: 95.25862068965517.
Epoch [2/8], Loss: 0.2659, Accuracy: 89.97%
Epoch [8/8], Loss: nan, Accuracy: 96.76%
Epoch [1/8], Loss: 0.5145, Accuracy: 73.14%
Epoch [1/8], Loss: 0.4818, Accuracy: 76.05%
Epoch [3/8], Loss: 0.2147, Accuracy: 91.80%
[I 2025-03-13 12:02:55,411] Trial 29 pruned.
🎯 Test Accuracy: 75.00%
[I 2025-03-13 12:03:01,793] Trial 28 finished with value: 75.0 and parameters: {'l
r': 0.008527221773849195, 'dropout': 0.3709684598726777, 'optimizer': 'AdamW', 'num_
epochs': 8}. Best is trial 18 with value: 95.25862068965517.
Epoch [2/8], Loss: 0.2521, Accuracy: 91.37%
Epoch [2/8], Loss: 0.2668, Accuracy: 91.26%
Epoch [1/13], Loss: 0.6281, Accuracy: 62.78%
[I 2025-03-13 12:03:27,643] Trial 32 pruned.
Epoch [1/13], Loss: nan, Accuracy: 69.04%
Epoch [3/8], Loss: 0.1840, Accuracy: 93.10%
Epoch [3/8], Loss: 0.1710, Accuracy: 94.28%
Epoch [1/11], Loss: 0.5123, Accuracy: 75.19%
Epoch [2/13], Loss: nan, Accuracy: 83.17%
[I 2025-03-13 12:04:07,620] Trial 33 pruned.
Epoch [4/8], Loss: 0.1775, Accuracy: 93.20%
[I 2025-03-13 12:04:10,958] Trial 30 pruned.
```

```
Epoch [4/8], Loss: 0.1527, Accuracy: 93.10%
Epoch [2/11], Loss: 0.3044, Accuracy: 91.05%
Epoch [1/11], Loss: nan, Accuracy: 73.89%
Epoch [1/10], Loss: 0.5329, Accuracy: 76.59%
Epoch [5/8], Loss: 0.1193, Accuracy: 96.33%
Epoch [3/11], Loss: 0.2032, Accuracy: 92.66%
[I 2025-03-13 12:05:07,147] Trial 34 pruned.
Epoch [2/11], Loss: nan, Accuracy: 89.97%
Epoch [2/10], Loss: 0.3122, Accuracy: 90.61%
Epoch [6/8], Loss: 0.1115, Accuracy: 95.79%
Epoch [1/10], Loss: 0.5989, Accuracy: 68.61%
Epoch [3/11], Loss: nan, Accuracy: 91.80%
[I 2025-03-13 12:05:48,357] Trial 35 pruned.
Epoch [3/10], Loss: 0.2082, Accuracy: 94.39%
Epoch [7/8], Loss: 0.0888, Accuracy: 96.55%
Epoch [2/10], Loss: 0.3870, Accuracy: 88.78%
[I 2025-03-13 12:06:13,552] Trial 37 pruned.
Epoch [1/10], Loss: nan, Accuracy: 72.28%
Epoch [4/10], Loss: 0.1904, Accuracy: 95.04%
Epoch [8/8], Loss: 0.0999, Accuracy: 97.52%
Epoch [1/9], Loss: 0.5907, Accuracy: 65.16%
[I 2025-03-13 12:06:47,680] Trial 39 pruned.
Epoch [2/10], Loss: nan, Accuracy: 89.43%
[I 2025-03-13 12:06:55,071] Trial 38 pruned.
Epoch [5/10], Loss: 0.1376, Accuracy: 95.47%
🎯 Test Accuracy: 92.24%
[I 2025-03-13 12:07:00,300] Trial 31 finished with value: 92.24137931034483 and para
meters: {'lr': 0.007681237679943294, 'dropout': 0.3493632933167954, 'optimizer': 'Ad
amW', 'num_epochs': 8}. Best is trial 18 with value: 95.25862068965517.
Epoch [1/7], Loss: 0.7357, Accuracy: 45.31%
[I 2025-03-13 12:07:19,945] Trial 40 pruned.
Epoch [1/15], Loss: nan, Accuracy: 57.39%
[I 2025-03-13 12:07:27,472] Trial 41 pruned.
Epoch [6/10], Loss: 0.1440, Accuracy: 95.47%
[I 2025-03-13 12:07:28,476] Trial 36 pruned.
Epoch [1/9], Loss: 0.6817, Accuracy: 56.42%
[I 2025-03-13 12:07:34,101] Trial 42 pruned.
Epoch [1/7], Loss: 0.6385, Accuracy: 63.11%
[I 2025-03-13 12:07:55,242] Trial 43 pruned.
Epoch [1/14], Loss: nan, Accuracy: 59.22%
[I 2025-03-13 12:08:03,132] Trial 44 pruned.
Epoch [1/14], Loss: 0.4375, Accuracy: 77.02%
Epoch [1/14], Loss: 0.4722, Accuracy: 74.97%
Epoch [1/14], Loss: 0.6210, Accuracy: 59.87%
[I 2025-03-13 12:08:29,580] Trial 47 pruned.
Epoch [2/14], Loss: 0.2525, Accuracy: 90.51%
Epoch [1/14], Loss: nan, Accuracy: 71.74%
Epoch [2/14], Loss: 0.2426, Accuracy: 91.15%
Epoch [1/12], Loss: 0.5561, Accuracy: 70.23%
Epoch [2/14], Loss: nan, Accuracy: 89.97%
Epoch [3/14], Loss: 0.1799, Accuracy: 92.45%
[I 2025-03-13 12:09:11,894] Trial 45 pruned.
Epoch [3/14], Loss: 0.2332, Accuracy: 90.08%
[I 2025-03-13 12:09:17,307] Trial 46 pruned.
Epoch [2/12], Loss: 0.3028, Accuracy: 89.64%
Epoch [3/14], Loss: nan, Accuracy: 91.91%
```

```
[I 2025-03-13 12:09:39,176] Trial 48 pruned.
Epoch [3/12], Loss: 0.1951, Accuracy: 93.64%
Epoch [4/12], Loss: 0.1645, Accuracy: 95.90%
Epoch [5/12], Loss: 0.1695, Accuracy: 94.93%
Epoch [6/12], Loss: 0.1808, Accuracy: 93.74%
Epoch [7/12], Loss: 0.0990, Accuracy: 97.09%
Epoch [8/12], Loss: 0.1114, Accuracy: 96.98%
Epoch [9/12], Loss: 0.0896, Accuracy: 97.20%
Epoch [10/12], Loss: 0.0816, Accuracy: 97.52%
Epoch [11/12], Loss: 0.1025, Accuracy: 97.09%
Epoch [12/12], Loss: 0.0975, Accuracy: 97.41%
[I 2025-03-13 12:13:24,824] Trial 49 pruned.

Best Hyperparameters: {'lr': 0.00803525510239778, 'dropout': 0.3233933125426155, 'op
timizer': 'AdamW', 'num_epochs': 10}

Accuracy: 95.25862068965517
```

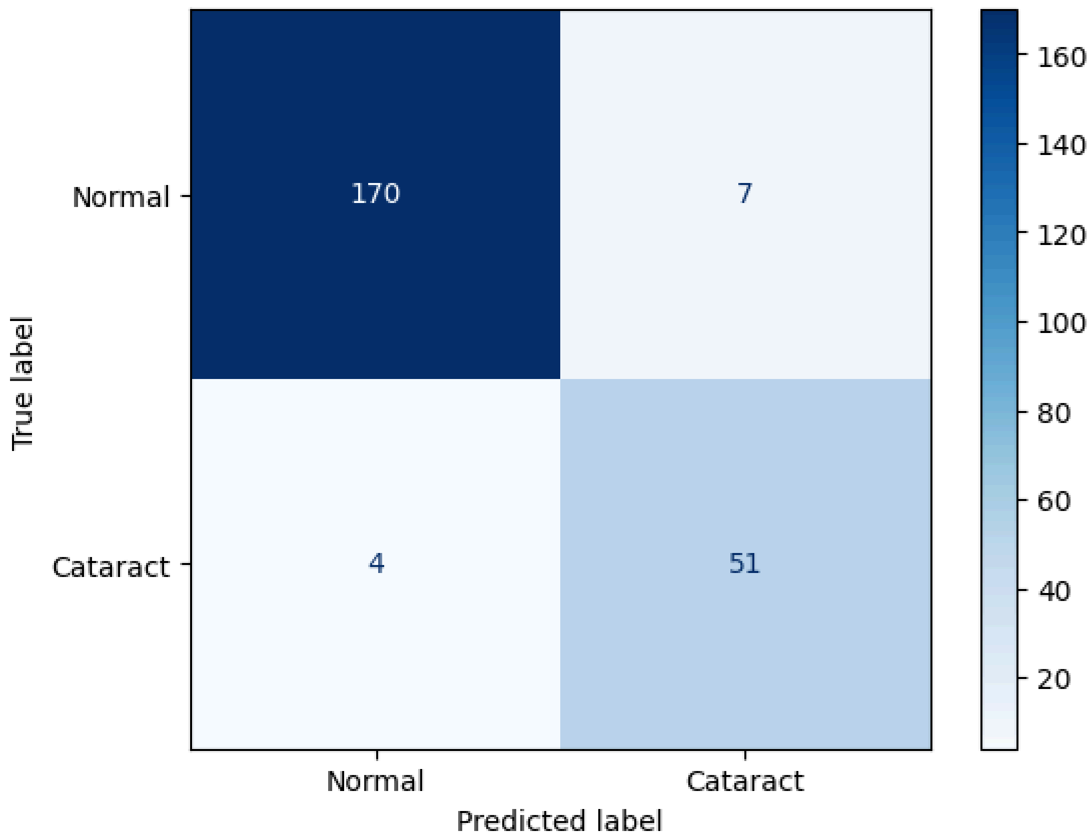
In [7]:

```python
best_trial = study.best_trial
best_train_losses = best_trial.user_attrs.get("train_losses", [])
best_train_accuracies = best_trial.user_attrs.get("train_accuracies", [])

plt.figure(figsize=(10, 5))

# Loss Graph
plt.subplot(1, 2, 1)
plt.plot(best_train_losses, label="Loss", marker="o", linestyle="-", color="b")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.title("Training Loss per Epoch")
plt.legend()

# Accuracy Graph
plt.subplot(1, 2, 2)
plt.plot(best_train_accuracies, label="Accuracy (%)", marker="o", linestyle="-", co
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.title("Training Accuracy per Epoch")
plt.legend()

plt.show()
```



In [8]:
```
best_preds = np.array(study.user_attrs.get("best_preds", []))
best_labels = np.array(study.user_attrs.get("best_labels", []))

conf_matrix = confusion_matrix(best_preds, best_labels)
ConfusionMatrixDisplay(conf_matrix, display_labels=class_labels).plot(cmap=plt.cm.B
cr = classification_report(best_preds, best_labels)

print(f"Accuracy: {acc:.2f}%")
print(cr)
```

```
Accuracy: 95.26%
              precision    recall  f1-score   support

         0.0       0.98      0.96      0.97       177
         1.0       0.88      0.93      0.90        55

    accuracy                           0.95       232
   macro avg       0.93      0.94      0.94       232
weighted avg       0.95      0.95      0.95       232
```



In [9]:
```
# # save
output_model_path = f"output/checkpoint-{acc}-hyperparam.pth"

torch.save({
    "model_state_dict": study.user_attrs.get("best_model_state"),  # Best model wei
    "optimizer_state_dict": study.user_attrs.get("best_optimizer_state"),  # Best o
    "best_hyperparameters": study.best_params,
    "best_accuracy": study.best_value,
    "best_train_losses": study.user_attrs.get("best_train_losses", []),  # Best tra
    "best_train_accuracies": study.user_attrs.get("best_train_accuracies", []),  #
}, output_model_path)
```

In [10]:
```
# checkpoint = torch.load("checkpoint.pth")
# model.load_state_dict(checkpoint['model_state_dict'])
# optimizer.load_state_dict(checkpoint['optimizer_state_dict'])
# model.eval()
```