In [1]:
```python
import kagglehub

# Download latest version
path = kagglehub.dataset_download("sohansakib75/yuyvvty")

print("Path to dataset files:", path)
```

```
Path to dataset files: /kaggle/input/yuyvvty
```

In [2]:
```python
import kagglehub
import os

# Download latest version
path = kagglehub.dataset_download("sohansakib75/yuyvvty")

print("Path to dataset files:", path)

# List subfolders and files inside the downloaded path
for root, dirs, files in os.walk(path):
    print(f"\nIn directory: {root}")
    for d in dirs:
        print("Subfolder:", d)
    for f in files:
        print("File:", f)
```

```
Path to dataset files: /kaggle/input/yuyvvty

In directory: /kaggle/input/yuyvvty
Subfolder: Image Dataset on Eye Diseases Classification (Uveitis, Conjunctivitis, Cataract, Eyelid) with Symptom
s and SMOTE Validation

In directory: /kaggle/input/yuyvvty/Image Dataset on Eye Diseases Classification (Uveitis, Conjunctivitis, Catar
act, Eyelid) with Symptoms and SMOTE Validation
Subfolder: Eyelid
Subfolder: Normal
Subfolder: Cataract
Subfolder: Uveitis
Subfolder: Conjunctivitis

In directory: /kaggle/input/yuyvvty/Image Dataset on Eye Diseases Classification (Uveitis, Conjunctivitis, Catar
act, Eyelid) with Symptoms and SMOTE Validation/Eyelid
File: 266.jpeg
File: 228.jpeg
File: 761.jpeg
File: 833.jpeg
File: 130.jpeg
File: 158.jpeg
File: 977.jpeg
File: aug_990.jpg
File: 381.jpeg
File: 335.jpeg
File: aug_986.jpg
File: 265.jpeg
File: 843.jpeg
File: aug_962.jpg
File: aug_894.jpg
File: 251.jpeg
File: aug_896.jpg
File: 399.jpeg
File: 992.jpeg
File: 252.jpeg
File: 155.jpeg
File: 626.jpeg
File: 152.jpeg
File: 389.jpeg
File: 462.jpeg
File: 676.jpeg
File: 469.jpeg
File: 385.jpeg
File: 311.jpeg
File: 124.jpeg
File: 283.jpeg
File: 139.jpeg
File: 365.jpeg
File: 367.jpeg
File: 179.jpeg
File: 108.jpeg
File: 835.jpeg
File: 111.jpeg
File: 776.jpeg
File: 43.jpeg
File: 197.jpeg
```

File: aug_995.jpg
File: 339.jpeg
File: 277.jpeg
File: aug_979.jpg
File: aug_987.jpg
File: 115.jpeg
File: 249.jpeg
File: aug_949.jpg
File: 334.jpeg
File: 170.jpeg
File: 411.jpeg
File: 409.jpeg
File: 104.jpeg
File: 131.jpeg
File: 559.jpeg
File: 194.jpeg
File: 349.jpeg
File: 27.jpeg
File: 603.jpeg
File: 809.jpeg
File: 24.jpeg
File: aug_915.jpg
File: aug_953.jpg
File: 896.jpeg
File: 436.jpeg
File: 442.jpeg
File: 912.jpeg
File: 943.jpeg
File: 897.jpeg
File: 600.jpeg
File: 854.jpeg
File: 196.jpeg
File: 848.jpeg
File: 414.jpeg
File: 1000.jpeg
File: 269.jpeg
File: 924.jpeg
File: 203.jpeg
File: 754.jpeg
File: 102.jpeg
File: 824.jpeg
File: 354.jpeg
File: 45.jpeg
File: 913.jpeg
File: 314.jpeg
File: 254.jpeg
File: 480.jpeg
File: aug_910.jpg
File: 445.jpeg
File: 512.jpeg
File: 612.jpeg
File: aug_887.jpg
File: 189.jpeg
File: aug_946.jpg
File: 522.jpeg
File: 28.jpeg
File: 955.jpeg
File: aug_916.jpg
File: aug_950.jpg
File: 652.jpeg
File: 857.jpeg
File: 494.jpeg
File: 101.jpeg
File: 239.jpeg
File: 70.jpeg
File: 605.jpeg
File: 97.jpeg
File: 453.jpeg
File: 499.jpeg
File: 920.jpeg
File: 204.jpeg
File: 225.jpeg
File: 868.jpeg
File: 39.jpeg
File: 832.jpeg
File: 851.jpeg
File: aug_908.jpg
File: 207.jpeg
File: aug_919.jpg
File: 885.jpeg
File: 869.jpeg
File: 160.jpeg
File: 465.jpeg

File: 2.jpeg
File: 542.jpeg
File: 351.jpeg
File: 515.jpeg
File: 884.jpeg
File: 306.jpeg
File: 873.jpeg
File: 138.jpeg
File: aug_929.jpg
File: 860.jpeg
File: 648.jpeg
File: 153.jpeg
File: aug_888.jpg
File: aug_992.jpg
File: 520.jpeg
File: 288.jpeg
File: 870.jpeg
File: 988.jpeg
File: 477.jpeg
File: 844.jpeg
File: 526.jpeg
File: 435.jpeg
File: 688.jpeg
File: 172.jpeg
File: 350.jpeg
File: 227.jpeg
File: aug_966.jpg
File: 396.jpeg
File: aug_884.jpg
File: 33.jpeg
File: 304.jpeg
File: 374.jpeg
File: 821.jpeg
File: 222.jpeg
File: 318.jpeg
File: aug_945.jpg
File: 541.jpeg
File: 953.jpeg
File: aug_957.jpg
File: 446.jpeg
File: 65.jpeg
File: 178.jpeg
File: 246.jpeg
File: 482.jpeg
File: 284.jpeg
File: 184.jpeg
File: aug_893.jpg
File: 770.jpeg
File: 307.jpeg
File: aug_997.jpg
File: 363.jpeg
File: 831.jpeg
File: 859.jpeg
File: 390.jpeg
File: 528.jpeg
File: 200.jpeg
File: 319.jpeg
File: aug_948.jpg
File: 279.jpeg
File: aug_930.jpg
File: 379.jpeg
File: 799.jpeg
File: 900.jpeg
File: 473.jpeg
File: 415.jpeg
File: 690.jpeg
File: 569.jpeg
File: 407.jpeg
File: 606.jpeg
File: 161.jpeg
File: 918.jpeg
File: 960.jpeg
File: 547.jpeg
File: 741.jpeg
File: aug_922.jpg
File: 470.jpeg
File: aug_989.jpg
File: 295.jpeg
File: 852.jpeg
File: 527.jpeg
File: 403.jpeg
File: 433.jpeg
File: aug_978.jpg

File: 466.jpeg
File: 966.jpeg
File: 241.jpeg
File: 697.jpeg
File: 369.jpeg
File: 906.jpeg
File: 531.jpeg
File: 154.jpeg
File: 801.jpeg
File: 149.jpeg
File: 613.jpeg
File: 190.jpeg
File: 127.jpeg
File: 635.jpeg
File: 183.jpeg
File: 417.jpeg
File: 296.jpeg
File: 355.jpeg
File: 450.jpeg
File: 37.jpeg
File: aug_975.jpg
File: 837.jpeg
File: 836.jpeg
File: 740.jpeg
File: 145.jpeg
File: 21.jpeg
File: 753.jpeg
File: 922.jpeg
File: aug_895.jpg
File: 186.jpeg
File: 979.jpeg
File: 732.jpeg
File: aug_937.jpg
File: 402.jpeg
File: 180.jpeg
File: aug_904.jpg
File: 815.jpeg
File: 475.jpeg
File: 540.jpeg
File: 370.jpeg
File: 840.jpeg
File: aug_959.jpg
File: aug_963.jpg
File: 588.jpeg
File: 680.jpeg
File: 965.jpeg
File: 610.jpeg
File: 188.jpeg
File: 408.jpeg
File: 110.jpeg
File: 129.jpeg
File: 401.jpeg
File: aug_931.jpg
File: 878.jpeg
File: 297.jpeg
File: 599.jpeg
File: 137.jpeg
File: 321.jpeg
File: 221.jpeg
File: 994.jpeg
File: 257.jpeg
File: 51.jpeg
File: 208.jpeg
File: 657.jpeg
File: 841.jpeg
File: 774.jpeg
File: 372.jpeg
File: aug_947.jpg
File: 135.jpeg
File: 1.jpeg
File: aug_905.jpg
File: 434.jpeg
File: 592.jpeg
File: 273.jpeg
File: 373.jpeg
File: 850.jpeg
File: 174.jpeg
File: 392.jpeg
File: 579.jpeg
File: 202.jpeg
File: 607.jpeg
File: 176.jpeg
File: aug_958.jpg

File: 898.jpeg
File: 226.jpeg
File: 769.jpeg
File: 195.jpeg
File: 3.jpeg
File: aug_967.jpg
File: 193.jpeg
File: 413.jpeg
File: 595.jpeg
File: 406.jpeg
File: 751.jpeg
File: 683.jpeg
File: aug_988.jpg
File: 114.jpeg
File: 451.jpeg
File: 320.jpeg
File: aug_982.jpg
File: 255.jpeg
File: 326.jpeg
File: 123.jpeg
File: aug_936.jpg
File: 963.jpeg
File: 866.jpeg
File: 946.jpeg
File: aug_902.jpg
File: aug_961.jpg
File: 100.jpeg
File: aug_980.jpg
File: 684.jpeg
File: 972.jpeg
File: 441.jpeg
File: 13.jpeg
File: 328.jpeg
File: aug_913.jpg
File: 951.jpeg
File: 103.jpeg
File: aug_996.jpg
File: 206.jpeg
File: 236.jpeg
File: 54.jpeg
File: 343.jpeg
File: 596.jpeg
File: 375.jpeg
File: 793.jpeg
File: aug_885.jpg
File: aug_938.jpg
File: 778.jpeg
File: 604.jpeg
File: 767.jpeg
File: 496.jpeg
File: 366.jpeg
File: 570.jpeg
File: 36.jpeg
File: 598.jpeg
File: 937.jpeg
File: 316.jpeg
File: 169.jpeg
File: aug_968.jpg
File: 90.jpeg
File: 539.jpeg
File: 315.jpeg
File: aug_926.jpg
File: 443.jpeg
File: 807.jpeg
File: 209.jpeg
File: 887.jpeg
File: aug_977.jpg
File: aug_971.jpg
File: 358.jpeg
File: 95.jpeg
File: 625.jpeg
File: 264.jpeg
File: 301.jpeg
File: 66.jpeg
File: 400.jpeg
File: aug_932.jpg
File: 644.jpeg
File: 471.jpeg
File: 80.jpeg
File: aug_917.jpg
File: 575.jpeg
File: 497.jpeg
File: 886.jpeg

File: 383.jpeg
File: 312.jpeg
File: 871.jpeg
File: 219.jpeg
File: 768.jpeg
File: 611.jpeg
File: 847.jpeg
File: 10.jpeg
File: 230.jpeg
File: aug_918.jpg
File: 468.jpeg
File: 615.jpeg
File: 910.jpeg
File: 593.jpeg
File: 630.jpeg
File: 903.jpeg
File: aug_956.jpg
File: 368.jpeg
File: 973.jpeg
File: 240.jpeg
File: 143.jpeg
File: 968.jpeg
File: 919.jpeg
File: 185.jpeg
File: 498.jpeg
File: 771.jpeg
File: 699.jpeg
File: aug_976.jpg
File: 529.jpeg
File: 536.jpeg
File: aug_898.jpg
File: 987.jpeg
File: aug_942.jpg
File: aug_925.jpg
File: aug_911.jpg
File: 412.jpeg
File: aug_969.jpg
File: 291.jpeg
File: 842.jpeg
File: 398.jpeg
File: 140.jpeg
File: 210.jpeg
File: 637.jpeg
File: 302.jpeg
File: aug_924.jpg
File: 907.jpeg
File: 85.jpeg
File: 675.jpeg
File: 474.jpeg
File: 94.jpeg
File: 839.jpeg
File: 198.jpeg
File: 410.jpeg
File: 959.jpeg
File: 397.jpeg
File: 136.jpeg
File: 581.jpeg
File: 818.jpeg
File: 890.jpeg
File: 537.jpeg
File: 742.jpeg
File: 717.jpeg
File: 300.jpeg
File: 69.jpeg
File: 856.jpeg
File: 679.jpeg
File: 608.jpeg
File: 893.jpeg
File: 858.jpeg
File: 191.jpeg
File: 105.jpeg
File: 949.jpeg
File: 971.jpeg
File: 775.jpeg
File: 654.jpeg
File: 181.jpeg
File: 863.jpeg
File: 993.jpeg
File: aug_907.jpg
File: 147.jpeg
File: aug_886.jpg
File: 560.jpeg
File: 915.jpeg

```
File: aug_901.jpg
File: 223.jpeg
File: 262.jpeg
File: 668.jpeg
File: 109.jpeg
File: 242.jpeg
File: 285.jpeg
File: aug_954.jpg
File: 530.jpeg
File: 748.jpeg
File: 371.jpeg
File: 275.jpeg
File: 532.jpeg
File: 669.jpeg
File: 26.jpeg
File: 212.jpeg
File: 199.jpeg
File: aug_983.jpg
File: 280.jpeg
File: 667.jpeg
File: 614.jpeg
File: 258.jpeg
File: 141.jpeg
File: 305.jpeg
File: 317.jpeg
File: aug_964.jpg
File: aug_955.jpg
File: 53.jpeg
File: aug_912.jpg
File: 387.jpeg
File: 286.jpeg
File: 133.jpeg
File: 661.jpeg
File: aug_960.jpg
File: 458.jpeg
File: aug_970.jpg
File: 982.jpeg
File: 220.jpeg
File: 347.jpeg
File: 192.jpeg
File: aug_998.jpg
File: 235.jpeg
File: 958.jpeg
File: 211.jpeg
File: 773.jpeg
File: 112.jpeg
File: 746.jpeg
File: 764.jpeg
File: 394.jpeg
File: 864.jpeg
File: 187.jpeg
File: 855.jpeg
File: aug_934.jpg
File: aug_889.jpg
File: 364.jpeg
File: 678.jpeg
File: 247.jpeg
File: 260.jpeg
File: aug_903.jpg
File: 452.jpeg
File: 237.jpeg
File: 755.jpeg
File: 313.jpeg
File: aug_972.jpg
File: 201.jpeg
File: aug_906.jpg
File: 674.jpeg
File: 619.jpeg
File: 538.jpeg

In directory: /kaggle/input/yuyvvty/Image Dataset on Eye Diseases Classification (Uveitis, Conjunctivitis, Catar
act, Eyelid) with Symptoms and SMOTE Validation/Normal
File: 623.jpg
File: 208.jpg
File: 333.jpg
File: 537.jpg
File: 45.jpg
File: 56.jpg
File: 654.jpg
File: 89.jpg
File: 20.jpg
File: 275.jpg
File: 212.jpg
```

File: 239.jpg
File: 58.jpg
File: 150.jpg
File: 109.jpg
File: 149.jpg
File: 187.jpg
File: 521.jpg
File: 436.jpg
File: 76.jpg
File: 539.jpg
File: 355.jpg
File: 474.jpg
File: 501.jpg
File: 342.jpg
File: 682.jpg
File: 544.jpg
File: 377.jpg
File: 272.jpg
File: 270.jpg
File: 182.jpg
File: 215.jpg
File: 489.jpg
File: 576.jpg
File: 185.jpg
File: 613.jpg
File: 243.jpg
File: 153.jpg
File: 189.jpg
File: 143.jpg
File: 476.jpg
File: 327.jpg
File: 253.jpg
File: 343.jpg
File: 131.jpg
File: 446.jpg
File: 626.jpg
File: 425.jpg
File: 366.jpg
File: 151.jpg
File: 426.jpg
File: 503.jpg
File: 622.jpg
File: 440.jpg
File: 260.jpg
File: 534.jpg
File: 202.jpg
File: 84.jpg
File: 577.jpg
File: 237.jpg
File: 273.jpg
File: 286.jpg
File: 283.jpg
File: 486.jpg
File: 513.jpg
File: 85.jpg
File: 564.jpg
File: 359.jpg
File: 67.jpg
File: 265.jpg
File: 289.jpg
File: 591.jpg
File: 82.jpg
File: 443.jpg
File: 295.jpg
File: 176.jpg
File: 279.jpg
File: 668.jpg
File: 30.jpg
File: 97.jpg
File: 334.jpg
File: 106.jpg
File: 163.jpg
File: 113.jpg
File: 160.jpg
File: 518.jpg
File: 38.jpg
File: 463.jpg
File: 349.jpg
File: 490.jpg
File: 328.jpg
File: 211.jpg
File: 42.jpg
File: 291.jpg

File: 197.jpg
File: 630.jpg
File: 33.jpg
File: 234.jpg
File: 178.jpg
File: 54.jpg
File: 672.jpg
File: 271.jpg
File: 683.jpg
File: 251.jpg
File: 665.jpg
File: 62.jpg
File: 589.jpg
File: 278.jpg
File: 500.jpg
File: 156.jpg
File: 380.jpg
File: 558.jpg
File: 120.jpg
File: 456.jpg
File: 35.jpg
File: 629.jpg
File: 290.jpg
File: 61.jpg
File: 190.jpg
File: 124.jpg
File: 441.jpg
File: 353.jpg
File: 191.jpg
File: 512.jpg
File: 427.jpg
File: 59.jpg
File: 604.jpg
File: 680.jpg
File: 595.jpg
File: 379.jpg
File: 313.jpg
File: 601.jpg
File: 188.jpg
File: 274.jpg
File: 98.jpg
File: 396.jpg
File: 41.jpg
File: 480.jpg
File: 527.jpg
File: 620.jpg
File: 248.jpg
File: 230.jpg
File: 580.jpg
File: 685.jpg
File: 408.jpg
File: 509.jpg
File: 256.jpg
File: 559.jpg
File: 247.jpg
File: 477.jpg
File: 94.jpg
File: 60.jpg
File: 627.jpg
File: 538.jpg
File: 221.jpg
File: 167.jpg
File: 57.jpg
File: 227.jpg
File: 690.jpg
File: 112.jpg
File: 664.jpg
File: 372.jpg
File: 388.jpg
File: 478.jpg
File: 193.jpg
File: 639.jpg
File: 452.jpg
File: 152.jpg
File: 368.jpg
File: 506.jpg
File: 192.jpg
File: 412.jpg
File: 91.jpg
File: 653.jpg
File: 101.jpg
File: 651.jpg
File: 99.jpg

File: 311.jpg
File: 241.jpg
File: 397.jpg
File: 37.jpg
File: 340.jpg
File: 435.jpg
File: 524.jpg
File: 177.jpg
File: 186.jpg
File: 642.jpg
File: 358.jpg
File: 69.jpg
File: 468.jpg
File: 75.jpg
File: 403.jpg
File: 660.jpg
File: 81.jpg
File: 684.jpg
File: 390.jpg
File: 299.jpg
File: 254.jpg
File: 410.jpg
File: 381.jpg
File: 535.jpg
File: 393.jpg
File: 424.jpg
File: 494.jpg
File: 481.jpg
File: 401.jpg
File: 444.jpg
File: 276.jpg
File: 46.jpg
File: 560.jpg
File: 319.jpg
File: 137.jpg
File: 525.jpg
File: 455.jpg
File: 574.jpg
File: 267.jpg
File: 44.jpg
File: 561.jpg
File: 563.jpg
File: 65.jpg
File: 50.jpg
File: 314.jpg
File: 467.jpg
File: 530.jpg
File: 127.jpg
File: 196.jpg
File: 569.jpg
File: 29.jpg
File: 140.jpg
File: 531.jpg
File: 235.jpg
File: 482.jpg
File: 322.jpg
File: 79.jpg
File: 302.jpg
File: 179.jpg
File: 284.jpg
File: 419.jpg
File: 285.jpg
File: 387.jpg
File: 590.jpg
File: 105.jpg
File: 695.jpg
File: 649.jpg
File: 371.jpg
File: 16.jpg
File: 111.jpg
File: 634.jpg
File: 55.jpg
File: 317.jpg
File: 619.jpg
File: 356.jpg
File: 145.jpg
File: 611.jpg
File: 658.jpg
File: 135.jpg
File: 246.jpg
File: 541.jpg
File: 659.jpg
File: 548.jpg

File: 434.jpg
File: 214.jpg
File: 699.jpg
File: 225.jpg
File: 252.jpg
File: 77.jpg
File: 166.jpg
File: 292.jpg
File: 80.jpg
File: 466.jpg
File: 159.jpg
File: 464.jpg
File: 458.jpg
File: 121.jpg
File: 28.jpg
File: 173.jpg
File: 378.jpg
File: 645.jpg
File: 573.jpg
File: 691.jpg
File: 671.jpg
File: 400.jpg
File: 678.jpg
File: 171.jpg
File: 329.jpg
File: 258.jpg
File: 498.jpg
File: 603.jpg
File: 565.jpg
File: 392.jpg
File: 103.jpg
File: 174.jpg
File: 40.jpg
File: 261.jpg
File: 391.jpg
File: 451.jpg
File: 357.jpg
File: 543.jpg
File: 332.jpg
File: 296.jpg
File: 700.jpg
File: 346.jpg
File: 554.jpg
File: 308.jpg
File: 264.jpg
File: 633.jpg
File: 199.jpg
File: 310.jpg
File: 687.jpg
File: 126.jpg
File: 142.jpg
File: 48.jpg
File: 339.jpg
File: 631.jpg
File: 532.jpg
File: 280.jpg
File: 572.jpg
File: 555.jpg
File: 169.jpg
File: 656.jpg
File: 421.jpg
File: 545.jpg
File: 449.jpg
File: 194.jpg
File: 663.jpg
File: 180.jpg
File: 104.jpg
File: 24.jpg
File: 567.jpg
File: 155.jpg
File: 88.jpg
File: 465.jpg
File: 344.jpg
File: 64.jpg
File: 352.jpg
File: 287.jpg
File: 269.jpg
File: 326.jpg
File: 485.jpg
File: 216.jpg
File: 450.jpg
File: 636.jpg
File: 158.jpg

File: 587.jpg
File: 575.jpg
File: 148.jpg
File: 168.jpg
File: 31.jpg
File: 586.jpg
File: 263.jpg
File: 195.jpg
File: 502.jpg
File: 255.jpg
File: 552.jpg
File: 693.jpg
File: 406.jpg
File: 281.jpg
File: 43.jpg
File: 138.jpg
File: 650.jpg
File: 496.jpg
File: 582.jpg
File: 207.jpg
File: 209.jpg
File: 598.jpg
File: 100.jpg
File: 376.jpg
File: 13.jpg
File: 228.jpg
File: 74.jpg
File: 617.jpg
File: 669.jpg
File: 323.jpg
File: 667.jpg
File: 337.jpg
File: 644.jpg
File: 507.jpg
File: 223.jpg
File: 350.jpg
File: 417.jpg
File: 447.jpg
File: 236.jpg
File: 68.jpg
File: 581.jpg
File: 53.jpg
File: 83.jpg
File: 588.jpg
File: 336.jpg
File: 107.jpg
File: 609.jpg
File: 347.jpg
File: 354.jpg
File: 154.jpg
File: 206.jpg
File: 164.jpg
File: 338.jpg
File: 301.jpg
File: 146.jpg
File: 571.jpg
File: 161.jpg
File: 698.jpg
File: 681.jpg
File: 198.jpg
File: 528.jpg
File: 277.jpg
File: 618.jpg
File: 331.jpg
File: 594.jpg
File: 624.jpg
File: 72.jpg
File: 257.jpg
File: 445.jpg
File: 578.jpg
File: 233.jpg
File: 585.jpg
File: 139.jpg
File: 318.jpg
File: 416.jpg
File: 200.jpg
File: 321.jpg
File: 32.jpg
File: 676.jpg
File: 488.jpg
File: 469.jpg
File: 102.jpg
File: 244.jpg

File: 675.jpg
File: 220.jpg
File: 17.jpg
File: 394.jpg
File: 492.jpg
File: 157.jpg
File: 438.jpg
File: 696.jpg
File: 26.jpg
File: 351.jpg
File: 638.jpg
File: 183.jpg
File: 39.jpg
File: 219.jpg
File: 373.jpg
File: 240.jpg
File: 602.jpg
File: 459.jpg
File: 288.jpg
File: 242.jpg
File: 86.jpg
File: 229.jpg
File: 523.jpg
File: 222.jpg
File: 330.jpg
File: 15.jpg
File: 628.jpg
File: 692.jpg
File: 383.jpg
File: 402.jpg
File: 557.jpg
File: 487.jpg
File: 238.jpg
File: 165.jpg
File: 204.jpg
File: 570.jpg
File: 606.jpg
File: 12.jpg
File: 566.jpg
File: 224.jpg
File: 141.jpg
File: 616.jpg
File: 92.jpg
File: 495.jpg
File: 293.jpg
File: 497.jpg
File: 686.jpg
File: 305.jpg
File: 11.jpg
File: 491.jpg
File: 70.jpg
File: 592.jpg
File: 652.jpg
File: 439.jpg
File: 210.jpg
File: 181.jpg
File: 484.jpg
File: 596.jpg
File: 405.jpg
File: 533.jpg
File: 579.jpg
File: 568.jpg
File: 550.jpg
File: 259.jpg
File: 389.jpg
File: 34.jpg
File: 661.jpg
File: 411.jpg
File: 520.jpg
File: 27.jpg
File: 432.jpg
File: 399.jpg
File: 294.jpg
File: 479.jpg
File: 409.jpg
File: 51.jpg
File: 262.jpg
File: 673.jpg
File: 448.jpg
File: 132.jpg
File: 546.jpg
File: 52.jpg
File: 341.jpg

File: 21.jpg
File: 418.jpg
File: 309.jpg
File: 365.jpg
File: 312.jpg
File: 184.jpg
File: 300.jpg
File: 250.jpg
File: 583.jpg
File: 688.jpg
File: 297.jpg
File: 556.jpg
File: 540.jpg
File: 493.jpg
File: 125.jpg
File: 666.jpg
File: 128.jpg
File: 172.jpg
File: 95.jpg
File: 553.jpg
File: 442.jpg
File: 36.jpg
File: 384.jpg
File: 610.jpg
File: 217.jpg
File: 615.jpg
File: 694.jpg
File: 536.jpg
File: 657.jpg
File: 608.jpg
File: 386.jpg
File: 96.jpg
File: 600.jpg
File: 348.jpg
File: 584.jpg
File: 404.jpg
File: 144.jpg
File: 662.jpg
File: 110.jpg
File: 655.jpg
File: 413.jpg
File: 407.jpg
File: 522.jpg
File: 325.jpg
File: 510.jpg
File: 437.jpg
File: 385.jpg
File: 625.jpg
File: 475.jpg
File: 471.jpg
File: 63.jpg
File: 621.jpg
File: 398.jpg
File: 162.jpg
File: 231.jpg
File: 643.jpg
File: 499.jpg
File: 508.jpg
File: 201.jpg
File: 303.jpg
File: 514.jpg
File: 87.jpg
File: 670.jpg
File: 47.jpg
File: 612.jpg
File: 562.jpg
File: 282.jpg
File: 547.jpg
File: 93.jpg
File: 470.jpg
File: 170.jpg
File: 428.jpg
File: 324.jpg
File: 457.jpg
File: 689.jpg
File: 14.jpg
File: 549.jpg
File: 519.jpg
File: 18.jpg
File: 697.jpg
File: 175.jpg
File: 268.jpg
File: 517.jpg

File: 526.jpg
File: 607.jpg
File: 345.jpg
File: 316.jpg
File: 226.jpg
File: 637.jpg
File: 462.jpg
File: 614.jpg
File: 78.jpg
File: 320.jpg
File: 367.jpg
File: 433.jpg
File: 266.jpg
File: 245.jpg
File: 335.jpg
File: 460.jpg
File: 648.jpg
File: 232.jpg
File: 108.jpg
File: 49.jpg
File: 203.jpg
File: 640.jpg
File: 679.jpg
File: 542.jpg
File: 505.jpg
File: 66.jpg
File: 511.jpg
File: 632.jpg
File: 298.jpg
File: 483.jpg
File: 306.jpg
File: 647.jpg
File: 395.jpg
File: 423.jpg
File: 307.jpg
File: 593.jpg
File: 504.jpg
File: 205.jpg
File: 422.jpg
File: 133.jpg
File: 304.jpg
File: 551.jpg
File: 315.jpg
File: 134.jpg
File: 218.jpg
File: 635.jpg
File: 249.jpg
File: 213.jpg
File: 431.jpg
File: 597.jpg
File: 461.jpg
File: 136.jpg
File: 605.jpg
File: 90.jpg
File: 599.jpg
File: 25.jpg
File: 147.jpg

In directory: /kaggle/input/yuyvvty/Image Dataset on Eye Diseases Classification (Uveitis, Conjunctivitis, Cataract, Eyelid) with Symptoms and SMOTE Validation/Cataract
File: aug_878.jpg
File: aug_928.jpg
File: 208.jpg
File: 333.jpg
File: aug_838.jpg
File: 20.jpg
File: 275.jpg
File: 212.jpg
File: aug_820.jpg
File: aug_944.jpg
File: 150.jpg
File: 109.jpg
File: 149.jpg
File: 187.jpg
File: 436.jpg
File: aug_876.jpg
File: 708.jpg
File: aug_896.jpg
File: aug_829.jpg
File: 342.jpg
File: aug_909.jpg
File: aug_927.jpg
File: 429.jpg

File: 272.jpg
File: 270.jpg
File: 182.jpg
File: 215.jpg
File: 185.jpg
File: 153.jpg
File: 703.jpg
File: 189.jpg
File: 143.jpg
File: 717.jpg
File: 327.jpg
File: 253.jpg
File: 343.jpg
File: aug_941.jpg
File: 115.jpg
File: 131.jpg
File: 446.jpg
File: aug_802.jpg
File: 425.jpg
File: 426.jpg
File: 732.jpg
File: aug_760.jpg
File: 440.jpg
File: 260.jpg
File: 534.JPG
File: 202.jpg
File: aug_979.jpg
File: aug_987.jpg
File: 273.jpg
File: 286.jpg
File: 283.jpg
File: aug_949.jpg
File: 265.jpg
File: 289.jpg
File: 118.jpg
File: aug_792.jpg
File: aug_915.jpg
File: aug_834.jpg
File: 723.jpg
File: aug_778.jpg
File: 443.jpg
File: 295.jpg
File: 176.jpg
File: aug_953.jpg
File: 279.jpg
File: 30.jpg
File: 334.jpg
File: aug_790.jpg
File: 163.jpg
File: 113.jpg
File: 160.jpg
File: 38.jpg
File: 328.jpg
File: aug_805.jpg
File: 211.jpg
File: 291.jpg
File: aug_847.jpg
File: 197.jpg
File: 704.jpg
File: 33.jpg
File: 10.jpg
File: 178.jpg
File: 271.jpg
File: 251.jpg
File: 278.jpg
File: 743.jpg
File: 156.jpg
File: 120.jpg
File: aug_994.jpg
File: aug_887.jpg
File: aug_940.jpg
File: 35.jpg
File: 290.jpg
File: aug_786.jpg
File: aug_800.jpg
File: 190.jpg
File: aug_855.jpg
File: 441.jpg
File: aug_946.jpg
File: 126.jpeg
File: 191.jpg
File: 427.jpg
File: aug_921.jpg

File: aug_974.jpg
File: aug_916.jpg
File: 313.jpg
File: 188.jpg
File: aug_950.jpg
File: 274.jpg
File: 396.jpg
File: 248.jpg
File: 123.jpg
File: aug_881.jpg
File: 710.jpg
File: aug_984.jpg
File: 230.jpg
File: 408.jpg
File: 256.jpg
File: 247.jpg
File: 221.jpg
File: 167.jpg
File: 227.jpg
File: 112.jpg
File: aug_827.jpg
File: 193.jpg
File: aug_856.jpg
File: 745.jpg
File: 738.jpg
File: aug_908.jpg
File: 192.jpg
File: 412.jpg
File: 701.jpg
File: aug_919.jpg
File: 705.jpg
File: 101.jpg
File: aug_868.jpg
File: 311.jpg
File: aug_768.jpg
File: 37.jpg
File: 340.jpg
File: 435.jpg
File: 1.jpg
File: 524.jpg
File: aug_850.jpg
File: 177.jpg
File: 130.png
File: 186.jpg
File: 720.jpg
File: 414.jpg
File: 403.jpg
File: 117.jpg
File: aug_929.jpg
File: 721.jpg
File: 390.jpg
File: 299.jpg
File: aug_888.jpg
File: 254.jpg
File: 410.jpg
File: 393.jpg
File: 424.jpg
File: aug_773.jpg
File: 401.jpg
File: 444.jpg
File: 276.jpg
File: aug_891.jpg
File: 319.jpg
File: 137.jpg
File: 716.jpg
File: aug_870.jpg
File: aug_966.jpg
File: aug_811.jpg
File: aug_769.jpg
File: 737.jpg
File: 267.jpg
File: aug_759.jpg
File: aug_774.jpg
File: 719.jpg
File: 314.jpg
File: aug_945.jpg
File: 127.jpg
File: 196.jpg
File: aug_957.jpg
File: 29.jpg
File: 140.jpg
File: aug_771.jpg
File: 713.jpg

File: 415.jpg
File: 322.jpg
File: aug_973.jpg
File: 302.jpg
File: aug_997.jpg
File: 179.jpg
File: 284.jpg
File: 419.jpg
File: 285.jpg
File: aug_930.jpg
File: 132.png
File: aug_793.jpg
File: 727.jpg
File: 16.jpg
File: 317.jpg
File: aug_861.jpg
File: aug_853.jpg
File: 145.jpg
File: aug_803.jpg
File: 135.jpg
File: aug_922.jpg
File: 23.jpg
File: 246.jpg
File: aug_852.jpg
File: aug_989.jpg
File: 434.jpg
File: 214.jpg
File: 699.jpg
File: 225.jpg
File: aug_770.jpg
File: 252.jpg
File: 166.jpg
File: aug_899.jpg
File: 159.jpg
File: aug_978.jpg
File: 430.jpg
File: aug_755.jpg
File: 121.jpg
File: 28.jpg
File: 22.jpg
File: 173.jpg
File: 400.jpg
File: 171.jpg
File: aug_883.jpg
File: 329.jpg
File: 258.jpg
File: 392.jpg
File: 103.jpg
File: aug_842.jpg
File: 174.jpg
File: aug_814.jpg
File: 261.jpg
File: 391.jpg
File: 747.jpg
File: aug_897.jpg
File: 332.jpg
File: 296.jpg
File: aug_846.jpg
File: 700.jpg
File: 750.jpg
File: 346.jpg
File: 308.jpg
File: 264.jpg
File: aug_895.jpg
File: aug_830.jpg
File: 199.jpg
File: 310.jpg
File: aug_858.jpg
File: 142.jpg
File: 339.jpg
File: aug_937.jpg
File: aug_796.jpg
File: 730.jpg
File: aug_826.jpg
File: 748.jpg
File: 280.jpg
File: aug_904.jpg
File: aug_816.jpg
File: 169.jpg
File: 421.jpg
File: 449.jpg
File: 194.jpg
File: aug_865.jpg

File: aug_963.jpg
File: 180.jpg
File: 104.jpg
File: 24.jpg
File: 155.jpg
File: 344.jpg
File: 734.jpg
File: aug_751.jpg
File: 287.jpg
File: 269.jpg
File: 326.jpg
File: 420.jpg
File: 216.jpg
File: 702.jpg
File: 158.jpg
File: 148.jpg
File: 168.jpg
File: 31.jpg
File: 263.jpg
File: 744.jpg
File: 195.jpg
File: aug_947.jpg
File: 255.jpg
File: 114.jpg
File: aug_762.jpg
File: 406.jpg
File: 281.jpg
File: 709.jpg
File: aug_920.jpg
File: 207.jpg
File: 209.jpg
File: aug_837.jpg
File: aug_943.jpg
File: 100.jpg
File: 13.jpg
File: 323.jpg
File: 337.jpg
File: 223.jpg
File: 417.jpg
File: aug_809.jpg
File: 447.jpg
File: aug_806.jpg
File: 336.jpg
File: 154.jpg
File: 206.jpg
File: aug_866.jpg
File: 164.jpg
File: aug_967.jpg
File: 338.jpg
File: 301.jpg
File: aug_823.jpg
File: aug_988.jpg
File: 146.jpg
File: 161.jpg
File: aug_812.jpg
File: aug_982.jpg
File: 198.jpg
File: 277.jpg
File: aug_936.jpg
File: 331.jpg
File: aug_822.jpg
File: 257.jpg
File: aug_902.jpg
File: aug_961.jpg
File: 139.jpg
File: 318.jpg
File: aug_980.jpg
File: 416.jpg
File: 200.jpg
File: 321.jpg
File: 32.jpg
File: aug_788.jpg
File: 741.jpg
File: aug_913.jpg
File: aug_841.jpg
File: aug_819.jpg
File: 102.jpg
File: aug_756.jpg
File: 220.jpg
File: 17.jpg
File: 394.jpg
File: aug_862.jpg
File: 157.jpg

File: 438.jpg
File: aug_882.jpg
File: 726.jpg
File: 183.jpg
File: 742.jpg
File: 219.jpg
File: aug_789.jpg
File: 288.jpg
File: 229.jpg
File: aug_933.jpg
File: aug_833.jpg
File: aug_938.jpg
File: 222.jpg
File: 330.jpg
File: 15.jpg
File: aug_753.jpg
File: 402.jpg
File: aug_825.jpg
File: 119.jpg
File: aug_845.jpg
File: 165.jpg
File: 204.jpg
File: aug_758.jpg
File: 12.jpg
File: 722.jpg
File: 224.jpg
File: aug_926.jpg
File: 706.jpg
File: 141.jpg
File: 746.jpg
File: 749.jpg
File: aug_971.jpg
File: aug_752.jpg
File: 122.jpg
File: 293.jpg
File: aug_965.jpg
File: aug_772.jpg
File: 305.jpg
File: 11.jpg
File: aug_917.jpg
File: 439.jpg
File: 210.jpg
File: aug_874.jpg
File: 181.jpg
File: aug_804.jpg
File: 405.jpg
File: aug_832.jpg
File: aug_791.jpg
File: 715.jpg
File: aug_787.jpg
File: 259.jpg
File: 34.jpg
File: aug_918.jpg
File: 411.jpg
File: 718.jpg
File: 432.jpg
File: 399.jpg
File: 294.jpg
File: aug_782.jpg
File: 409.jpg
File: aug_956.jpg
File: 107(1).jpg
File: aug_779.jpg
File: 262.jpg
File: 729.jpg
File: aug_761.jpg
File: 341.jpg
File: 21.jpg
File: 418.jpg
File: aug_848.jpg
File: aug_807.jpg
File: 309.jpg
File: aug_892.jpg
File: 312.jpg
File: 184.JPG
File: 300.jpg
File: aug_794.jpg
File: 297.jpg
File: aug_898.jpg
File: 128.png
File: 125.jpg
File: aug_942.jpg
File: aug_828.jpg

File: 172.jpg
File: aug_836.jpg
File: aug_831.jpg
File: aug_764.jpg
File: 442.jpg
File: aug_911.jpg
File: 36.jpg
File: aug_969.jpg
File: 217.jpg
File: aug_924.jpg
File: aug_871.jpg
File: aug_860.jpg
File: aug_766.jpg
File: 736.jpg
File: aug_808.jpg
File: 144.jpg
File: 110.jpg
File: 413.jpg
File: 407.jpg
File: aug_810.jpg
File: aug_785.jpg
File: 325.jpg
File: 739.jpg
File: 437.jpg
File: 724.jpg
File: aug_777.jpg
File: 712.jpg
File: aug_993.jpg
File: 731.jpg
File: 733.jpg
File: 398.jpg
File: aug_859.jpg
File: 162.jpg
File: 19.jpg
File: 201.jpg
File: 303.jpg
File: 725.jpg
File: aug_799.jpg
File: 282.jpg
File: aug_901.jpg
File: 170.jpg
File: aug_780.jpg
File: 707.jpg
File: 428.jpg
File: 324.jpg
File: aug_815.jpg
File: aug_939.jpg
File: aug_954.jpg
File: 14.jpg
File: aug_843.jpg
File: 18.jpg
File: 175.jpg
File: 268.jpg
File: aug_818.jpg
File: 316.jpg
File: 226.jpg
File: 740.jpg
File: 116.jpg
File: aug_983.jpg
File: aug_763.jpg
File: aug_757.jpg
File: aug_776.jpg
File: aug_867.jpg
File: aug_754.jpg
File: aug_964.jpg
File: aug_955.jpg
File: 320.jpg
File: 433.jpg
File: aug_912.jpg
File: 266.jpg
File: 335.jpg
File: aug_960.jpg
File: aug_879.jpg
File: 108.jpg
File: aug_985.jpg
File: 203.jpg
File: 134.png
File: 714.jpg
File: aug_900.jpg
File: 298.jpg
File: 306.jpg
File: aug_934.jpg
File: aug_914.jpg

File: aug_889.jpg
File: 395.jpg
File: 423.jpg
File: 307.jpg
File: 129.jpg
File: 735.jpg
File: 2.jpg
File: 205.jpg
File: 422.jpg
File: aug_877.jpg
File: 133.jpg
File: 106(1).jpg
File: 304.jpg
File: aug_903.jpg
File: 315.jpg
File: 218.jpg
File: 213.jpg
File: aug_801.jpg
File: 136.jpg
File: aug_784.jpg
File: aug_906.jpg
File: 147.jpg
File: 728.jpg

In directory: /kaggle/input/yuyvvty/Image Dataset on Eye Diseases Classification (Uveitis, Conjunctivitis, Cataract, Eyelid) with Symptoms and SMOTE Validation/Uveitis
File: aug_878.jpg
File: 14.jpeg
File: 771.jpg
File: 150.jpg
File: 109.jpg
File: 149.jpg
File: aug_986.jpg
File: aug_962.jpg
File: aug_894.jpg
File: aug_876.jpg
File: aug_896.jpg
File: 915.jpg
File: 817.jpg
File: 182.jpg
File: 185.jpg
File: 16.jpeg
File: 153.jpg
File: 469.jpeg
File: 131.jpg
File: 25.jpeg
File: 824.jpg
File: 151.jpg
File: aug_915.jpg
File: 106.jpg
File: 163.jpg
File: 493.jpeg
File: 160.jpg
File: 788.jpg
File: 203.jpeg
File: 178.jpg
File: 759.jpg
File: 130.jpg
File: 916.jpg
File: 951.jpg
File: 743.jpg
File: 156.jpg
File: 189.jpeg
File: 190.jpg
File: 20.jpeg
File: 191.jpg
File: 680.jpg
File: aug_974.jpg
File: aug_916.jpg
File: 313.jpg
File: 341.jpeg
File: 123.jpg
File: aug_984.jpg
File: 509.jpg
File: 904.jpg
File: 932.jpg
File: 330.jpeg
File: 167.jpg
File: 690.jpg
File: 822.jpg
File: 193.jpg
File: 639.jpg
File: 152.jpg

File: 192.jpg
File: 653.jpg
File: aug_868.jpg
File: 340.jpg
File: 177.jpg
File: 186.jpg
File: 684.jpg
File: 535.jpg
File: 874.jpg
File: 774.jpg
File: 137.jpg
File: 350.jpeg
File: aug_884.jpg
File: 737.jpg
File: 314.jpg
File: 127.jpg
File: 196.jpg
File: 140.jpg
File: 19.jpeg
File: 246.jpeg
File: 826.jpg
File: 184.jpeg
File: 322.jpg
File: 878.jpg
File: 7.jpeg
File: 179.jpg
File: aug_930.jpg
File: 105.jpg
File: 59.jpeg
File: aug_873.jpg
File: 145.jpg
File: 46.jpeg
File: 135.jpg
File: 64.jpeg
File: 765.jpg
File: 292.jpg
File: 331.jpeg
File: 159.jpg
File: 241.jpeg
File: 173.jpg
File: 52.jpeg
File: 77.jpeg
File: aug_883.jpg
File: 103.jpg
File: 942.jpg
File: 174.jpg
File: aug_897.jpg
File: 21.jpeg
File: 308.jpg
File: 687.jpg
File: 142.jpg
File: 730.jpg
File: 631.jpg
File: 748.jpg
File: 931.jpg
File: 169.jpg
File: aug_959.jpg
File: 332.jpeg
File: 155.jpg
File: 465.jpg
File: 734.jpg
File: aug_931.jpg
File: aug_981.jpg
File: 775.jpg
File: 158.jpg
File: 148.jpg
File: 168.jpg
File: 61.jpeg
File: 744.jpg
File: 1.jpeg
File: 761.jpg
File: 138.jpg
File: 380.jpeg
File: 100.jpg
File: 323.jpg
File: 667.jpg
File: 789.jpg
File: 154.jpg
File: 164.jpg
File: 161.jpg
File: aug_982.jpg
File: 326.jpeg
File: 277.jpg

File: 299.jpeg
File: 911.jpg
File: 139.jpg
File: 318.jpg
File: aug_980.jpg
File: 13.jpeg
File: 676.jpg
File: 102.jpg
File: aug_996.jpg
File: 157.jpg
File: 726.jpg
File: 343.jpeg
File: aug_926.jpg
File: 141.jpg
File: 616.jpg
File: 122.jpg
File: aug_917.jpg
File: 181.jpg
File: 63.jpeg
File: 779.jpg
File: 10.jpeg
File: 661.jpg
File: 132.jpg
File: 143.jpeg
File: 325.jpeg
File: 309.jpg
File: aug_892.jpg
File: 862.jpg
File: 919.jpg
File: 29.jpeg
File: 128.jpg
File: 172.jpg
File: 3.jpg
File: 41.jpeg
File: 610.jpg
File: 816.jpg
File: 615.jpg
File: 536.jpg
File: 144.jpg
File: 110.jpg
File: 964.jpg
File: 300.jpeg
File: 69.jpeg
File: 790.jpg
File: 510.jpg
File: 62.jpeg
File: 927.jpg
File: 733.jpg
File: 162.jpg
File: aug_901.jpg
File: 333.jpeg
File: 707.jpg
File: 428.jpg
File: 71.jpeg
File: 336.jpeg
File: 697.jpg
File: 175.jpg
File: 26.jpeg
File: 116.jpg
File: 898.jpg
File: 962.jpg
File: 40.jpeg
File: 305.jpeg
File: 53.jpeg
File: aug_912.jpg
File: 286.jpeg
File: 329.jpeg
File: 648.jpg
File: 108.jpg
File: aug_998.jpg
File: 679.jpg
File: aug_900.jpg
File: 129.jpg
File: 735.jpg
File: 133.jpg
File: 134.jpg
File: 830.jpg
File: 136.jpg
File: aug_972.jpg
File: 75.jpeg
File: 754.jpg
File: 753.jpg
File: 599.jpg

```
In directory: /kaggle/input/yuyvvty/Image Dataset on Eye Diseases Classification (Uveitis, Conjunctivitis, Cataract, Eyelid) with Symptoms and SMOTE Validation/Conjunctivitis
File: 208.jpg
File: 333.jpg
File: 45.jpg
File: 369.jpg
File: 89.jpg
File: 275.jpg
File: 239.jpg
File: aug_820.jpg
File: 675.png
File: 150.jpg
File: 677.png
File: 149.jpg
File: 187.jpg
File: 377.jpg
File: 272.jpg
File: 182.jpg
File: 215.jpg
File: 153.jpg
File: 189.jpg
File: 143.jpg
File: 327.jpg
File: 131.jpg
File: aug_802.jpg
File: 425.jpg
File: aug_849.jpg
File: 366.jpg
File: 151.jpg
File: 440.jpg
File: 260.jpg
File: 534.jpg
File: 202.jpg
File: 681.png
File: 237.jpg
File: 273.jpg
File: 283.jpg
File: 359.jpg
File: 265.jpg
File: 118.jpg
File: 361.jpg
File: aug_792.jpg
File: aug_834.jpg
File: 605.png
File: aug_778.jpg
File: 443.jpg
File: 295.jpg
File: aug_775.jpg
File: 279.jpg
File: 30.jpg
File: 97.jpg
File: 334.jpg
File: 106.jpg
File: aug_790.jpg
File: 113.jpg
File: 38.jpg
File: 660.png
File: 328.jpg
File: aug_805.jpg
File: 42.jpg
File: 291.jpg
File: 608.png
File: 234.jpg
File: 178.jpg
File: 271.jpg
File: 251.jpg
File: 130.jpg
File: 62.jpg
File: 500.jpg
File: 156.jpg
File: 380.jpg
File: 120.jpg
File: 669.png
File: aug_786.jpg
File: aug_800.jpg
File: 190.jpg
File: 441.jpg
File: 191.jpg
File: 427.jpg
File: 672.png
File: 682.png
File: 379.jpg
```

File: 313.jpg
File: 188.jpg
File: 274.jpg
File: 98.jpg
File: 375.jpg
File: 248.jpg
File: 230.jpg
File: 408.jpg
File: 256.jpg
File: 604.png
File: 247.jpg
File: 94.jpg
File: 362.jpg
File: 673.png
File: 658.png
File: 112.jpg
File: 372.jpg
File: 388.jpg
File: 193.jpg
File: 374.jpg
File: 152.jpg
File: 368.jpg
File: 412.jpg
File: 91.jpg
File: 101.jpg
File: aug_781.jpg
File: 99.jpg
File: 382.jpg
File: 311.jpg
File: 241.jpg
File: 397.jpg
File: 37.jpg
File: 1.jpg
File: 186.jpg
File: 358.jpg
File: 414.jpg
File: 607.png
File: 403.jpg
File: aug_844.jpg
File: 117.jpg
File: 390.jpg
File: 299.jpg
File: 381.jpg
File: 535.jpg
File: aug_795.jpg
File: 393.jpg
File: 424.jpg
File: 494.jpg
File: 401.jpg
File: 276.jpg
File: 319.jpg
File: 137.jpg
File: 674.png
File: 267.jpg
File: aug_774.jpg
File: 65.jpg
File: 314.jpg
File: 683.png
File: 127.jpg
File: 29.jpg
File: 140.jpg
File: 531.jpg
File: 235.jpg
File: 415.jpg
File: 322.jpg
File: 79.jpg
File: 302.jpg
File: 419.jpg
File: 387.jpg
File: aug_793.jpg
File: 105.jpg
File: 111.jpg
File: 55.jpg
File: 317.jpg
File: 356.jpg
File: 145.jpg
File: aug_803.jpg
File: 135.jpg
File: 23.jpg
File: 246.jpg
File: 548.jpg
File: 434.jpg
File: 292.jpg

File: 430.jpg
File: 28.jpg
File: 22.jpg
File: 378.jpg
File: 400.jpg
File: 329.jpg
File: 258.jpg
File: 498.jpg
File: 392.jpg
File: 103.jpg
File: aug_814.jpg
File: 261.jpg
File: 391.jpg
File: 357.jpg
File: 332.jpg
File: 296.jpg
File: aug_846.jpg
File: aug_830.jpg
File: 199.jpg
File: 310.jpg
File: aug_821.jpg
File: 142.jpg
File: aug_826.jpg
File: 532.jpg
File: 280.jpg
File: aug_816.jpg
File: 169.jpg
File: 421.jpg
File: 545.jpg
File: 684.png
File: 104.jpg
File: 24.jpg
File: 155.jpg
File: 64.jpg
File: 287.jpg
File: 326.jpg
File: 216.jpg
File: 617.png
File: 667.png
File: 148.jpg
File: 168.jpg
File: 263.jpg
File: 195.jpg
File: 650.png
File: 680.png
File: aug_824.jpg
File: 114.jpg
File: 406.jpg
File: 281.jpg
File: 138.jpg
File: 496.jpg
File: 207.jpg
File: 100.jpg
File: 376.jpg
File: 228.jpg
File: 503(1).jpg
File: 323.jpg
File: 223.jpg
File: aug_809.jpg
File: 236.jpg
File: 363.jpg
File: aug_806.jpg
File: 107.jpg
File: 666.png
File: 154.jpg
File: 206.jpg
File: 301.jpg
File: 146.jpg
File: 198.jpg
File: 277.jpg
File: 331.jpg
File: aug_822.jpg
File: 257.jpg
File: 233.jpg
File: 139.jpg
File: 318.jpg
File: 416.jpg
File: 200.jpg
File: aug_841.jpg
File: aug_819.jpg
File: 102.jpg
File: 244.jpg
File: 157.jpg

File: 685.png
File: 373.jpg
File: 240.jpg
File: 288.jpg
File: 360.jpg
File: 242.jpg
File: 229.jpg
File: 330.jpg
File: 383.jpg
File: 402.jpg
File: 238.jpg
File: 119.jpg
File: 141.jpg
File: 92.jpg
File: 495.jpg
File: 293.jpg
File: 497.jpg
File: 439.jpg
File: 181.jpg
File: 670.png
File: aug_804.jpg
File: 405.jpg
File: 533.jpg
File: aug_832.jpg
File: aug_791.jpg
File: 671.png
File: 259.jpg
File: 389.jpg
File: 34.jpg
File: 411.jpg
File: 370.jpg
File: 399.jpg
File: 294.jpg
File: aug_782.jpg
File: 409.jpg
File: 665.png
File: 262.jpg
File: 132.jpg
File: 546.jpg
File: 309.jpg
File: 365.jpg
File: 312.jpg
File: 250.jpg
File: 297.jpg
File: 668.png
File: 128.jpg
File: 95.jpg
File: 442.jpg
File: 686.png
File: 384.jpg
File: 606.png
File: 386.jpg
File: 96.jpg
File: 404.jpg
File: aug_808.jpg
File: 144.jpg
File: 110.jpg
File: 413.jpg
File: 407.jpg
File: aug_785.jpg
File: 325.jpg
File: 437.jpg
File: 385.jpg
File: 63.jpg
File: 398.jpg
File: 231.jpg
File: 664.png
File: 499.jpg
File: 201.jpg
File: 303.jpg
File: aug_799.jpg
File: 687.png
File: 282.jpg
File: 93.jpg
File: 170.jpg
File: 428.jpg
File: 679.png
File: 324.jpg
File: 678.png
File: 268.jpg
File: aug_818.jpg
File: 676.png
File: 316.jpg

```
File: 116.jpg
File: aug_776.jpg
File: 320.jpg
File: 367.jpg
File: 433.jpg
File: 266.jpg
File: 245.jpg
File: 335.jpg
File: 502(1).jpg
File: 613.png
File: 108.jpg
File: 203.jpg
File: 66.jpg
File: 298.jpg
File: 395.jpg
File: 2.jpg
File: 133.jpg
File: 304.jpg
File: 315.jpg
File: 134.jpg
File: 249.jpg
File: aug_801.jpg
File: 136.jpg
File: 364.jpg
File: 90.jpg
File: 25.jpg
File: 659.png
File: 147.jpg
```

In [3]:

```python
import cv2
import numpy as np
import os
from PIL import Image
from tqdm import tqdm

base_dir = "/kaggle/input/yuyvvty/Image Dataset on Eye Diseases Classification (Uveitis, Conjunctivitis, Cataract
classes = ["Eyelid", "Normal", "Cataract", "Uveitis", "Conjunctivitis"]
TARGET_SIZE = (224, 224)

output_base = "/kaggle/working/preprocessed_all"

eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

def alpha_trimmed_mean_filter(img, kernel_size=3, alpha=2):
    pad = kernel_size // 2
    padded = cv2.copyMakeBorder(img, pad, pad, pad, pad, cv2.BORDER_REFLECT)
    filtered = np.zeros_like(img)
    for i in range(pad, padded.shape[0] - pad):
        for j in range(pad, padded.shape[1] - pad):
            kernel = padded[i - pad:i + pad + 1, j - pad:j + pad + 1].flatten()
            kernel.sort()
            trimmed = kernel[alpha:-alpha]
            filtered[i - pad, j - pad] = np.mean(trimmed)
    return filtered

def zoom_roi_with_padding(img, bbox, zoom_factor=1.3):
    h, w = img.shape[:2]
    x, y, bw, bh = bbox

    # Calculate ROI center
    cx = x + bw // 2
    cy = y + bh // 2

    # Calculate new ROI size
    new_w = int(bw * zoom_factor)
    new_h = int(bh * zoom_factor)

    # Calculate new bounding box with padding around ROI center
    x1 = max(cx - new_w // 2, 0)
    y1 = max(cy - new_h // 2, 0)
    x2 = min(cx + new_w // 2, w)
    y2 = min(cy + new_h // 2, h)

    # Crop and zoom ROI area
    roi = img[y1:y2, x1:x2]
    roi_resized = cv2.resize(roi, (w, h), interpolation=cv2.INTER_LINEAR)

    return roi_resized

def preprocess_image(img_path):
    img = cv2.imread(img_path)
    if img is None:
```

```
        return None
    original_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    # Resize full image first
    resized = cv2.resize(original_rgb, TARGET_SIZE)

    # Convert to grayscale for detection
    gray = cv2.cvtColor(resized, cv2.COLOR_RGB2GRAY)

    # Detect eyes
    eyes = eye_cascade.detectMultiScale(gray, 1.3, 5)
    if len(eyes) > 0:
        bbox = eyes[0]
    else:
        # Detect face
        faces = face_cascade.detectMultiScale(gray, 1.3, 5)
        if len(faces) > 0:
            bbox = faces[0]
        else:
            # If no detection, use whole image
            bbox = (0, 0, resized.shape[1], resized.shape[0])

    # Zoom ROI with padding, keep image size
    zoomed_img = zoom_roi_with_padding(resized, bbox, zoom_factor=1.3)

    # Apply CLAHE on grayscale
    gray_zoomed = cv2.cvtColor(zoomed_img, cv2.COLOR_RGB2GRAY)
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
    clahe_img = clahe.apply(gray_zoomed)

    # Apply alpha trimmed mean filter
    filtered_img = alpha_trimmed_mean_filter(clahe_img)

    # Convert back to RGB
    processed_img = cv2.merge([filtered_img]*3)

    return processed_img

# Process all images, save results
for cls in classes:
    class_input_dir = os.path.join(base_dir, cls)
    class_output_dir = os.path.join(output_base, cls)
    os.makedirs(class_output_dir, exist_ok=True)

    image_files = [f for f in os.listdir(class_input_dir) if f.lower().endswith(('.jpg', '.jpeg', '.png'))]

    for i, img_file in enumerate(tqdm(image_files, desc=f"Processing {cls}")):
        img_path = os.path.join(class_input_dir, img_file)
        preprocessed_img = preprocess_image(img_path)
        if preprocessed_img is not None:
            save_path = os.path.join(class_output_dir, f"image_{i+1}.jpg")
            Image.fromarray(preprocessed_img).save(save_path)

print("All images preprocessed and saved successfully!")
```

```
Processing Eyelid: 100%|██████████| 525/525 [03:17<00:00,  2.66it/s]
Processing Normal: 100%|██████████| 649/649 [04:02<00:00,  2.68it/s]
Processing Cataract: 100%|██████████| 544/544 [03:25<00:00,  2.65it/s]
Processing Uveitis: 100%|██████████| 223/223 [01:24<00:00,  2.65it/s]
Processing Conjunctivitis: 100%|██████████| 357/357 [02:15<00:00,  2.64it/s]
All images preprocessed and saved successfully!
```

In [ ]:

In [4]:
```
import cv2
import numpy as np
import os
from PIL import Image
import random

preprocessed_dir = "/kaggle/working/preprocessed_all"  # Folder where all preprocessed images saved
augmented_dir = "/kaggle/working/augmented_all"

TARGET_SIZE = (224, 224)
classes_3aug = ["Cataract", "Conjunctivitis", "Eyelid", "Normal"]
classes_4aug = ["Uveitis"]

def random_brightness(img):
    factor = 0.7 + 0.6 * random.random()  # brightness factor between 0.7-1.3
    img = cv2.convertScaleAbs(img, alpha=factor, beta=0)
    return img

def random_rotation(img, angle_range=15):
```

```
    angle = random.uniform(-angle_range, angle_range)
    h, w = img.shape[:2]
    M = cv2.getRotationMatrix2D((w//2, h//2), angle, 1)
    rotated = cv2.warpAffine(img, M, (w, h), borderMode=cv2.BORDER_REPLICATE)
    return rotated

def horizontal_flip(img):
    return cv2.flip(img, 1)

def random_shift_zoom(img, shift_ratio=0.1, zoom_range=0.1):
    h, w = img.shape[:2]
    max_dx = int(w * shift_ratio)
    max_dy = int(h * shift_ratio)
    dx = random.randint(-max_dx, max_dx)
    dy = random.randint(-max_dy, max_dy)
    M_shift = np.float32([[1, 0, dx], [0, 1, dy]])
    shifted = cv2.warpAffine(img, M_shift, (w, h), borderMode=cv2.BORDER_REPLICATE)
    zx = 1 + random.uniform(-zoom_range, zoom_range)
    M_zoom = cv2.getRotationMatrix2D((w/2, h/2), 0, zx)
    zoomed = cv2.warpAffine(shifted, M_zoom, (w, h), borderMode=cv2.BORDER_REPLICATE)
    return zoomed

def augment_image(img):
    aug_funcs = [random_brightness, random_rotation, horizontal_flip, random_shift_zoom]
    np.random.shuffle(aug_funcs)
    img_aug = img.copy()
    for func in aug_funcs[:2]:
        img_aug = func(img_aug)
    return img_aug

def save_augmented_images(class_name, image_paths, n_augments):
    save_dir = os.path.join(augmented_dir, class_name)
    os.makedirs(save_dir, exist_ok=True)

    for idx, orig_img_path in enumerate(image_paths):
        img = cv2.imread(orig_img_path)
        if img is None:
            continue
        img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

        # Save original image
        orig_save_path = os.path.join(save_dir, f"image_{idx}_0.jpg")
        Image.fromarray(img_rgb).save(orig_save_path)

        # Save augmented images
        for i in range(1, n_augments + 1):
            aug_img = augment_image(img_rgb)
            aug_save_path = os.path.join(save_dir, f"image_{idx}_{i}.jpg")
            Image.fromarray(aug_img).save(aug_save_path)

# Run for all classes
for cls in classes_3aug:
    class_dir = os.path.join(preprocessed_dir, cls)
    image_files = [os.path.join(class_dir, f) for f in os.listdir(class_dir) if f.lower().endswith(('.jpg', '.jp
    save_augmented_images(cls, image_files, 3)

for cls in classes_4aug:
    class_dir = os.path.join(preprocessed_dir, cls)
    image_files = [os.path.join(class_dir, f) for f in os.listdir(class_dir) if f.lower().endswith(('.jpg', '.jp
    save_augmented_images(cls, image_files, 6)  # Augment 6 times for Uveitis

print(f"All images augmented and saved to {augmented_dir}/")
```

```
All images augmented and saved to /kaggle/working/augmented_all/
```

In [9]:

```
import os
import random
from collections import defaultdict

augmented_dir = "/kaggle/working/augmented_all"
classes = ["Eyelid", "Normal", "Cataract", "Uveitis", "Conjunctivitis"]

split_ratios = {'train': 0.8, 'val': 0.1, 'test': 0.1}

# Store split paths here
split_data = {'train': defaultdict(list), 'val': defaultdict(list), 'test': defaultdict(list)}

for cls in classes:
    class_dir = os.path.join(augmented_dir, cls)
    image_files = [os.path.join(class_dir, f) for f in os.listdir(class_dir)
                   if f.lower().endswith(('.jpg', '.jpeg', '.png'))]
    random.shuffle(image_files)

    n = len(image_files)
```

```
    n_train = int(split_ratios['train'] * n)
    n_val = int(split_ratios['val'] * n)
    n_test = n - n_train - n_val

    split_data['train'][cls] = image_files[:n_train]
    split_data['val'][cls] = image_files[n_train:n_train + n_val]
    split_data['test'][cls] = image_files[n_train + n_val:]

# Print summary table
print(f"{'Class':<15} | {'Train':<6} | {'Val':<6} | {'Test':<6} | {'Total':<6}")
print("-" * 50)
for cls in classes:
    n_train = len(split_data['train'][cls])
    n_val = len(split_data['val'][cls])
    n_test = len(split_data['test'][cls])
    total = n_train + n_val + n_test
    print(f"{cls:<15} | {n_train:<6} | {n_val:<6} | {n_test:<6} | {total:<6}")
```

```
Class           | Train  | Val    | Test   | Total
--------------------------------------------------
Eyelid          | 1680   | 210    | 210    | 2100
Normal          | 2076   | 259    | 261    | 2596
Cataract        | 1740   | 217    | 219    | 2176
Uveitis         | 1248   | 156    | 157    | 1561
Conjunctivitis  | 1142   | 142    | 144    | 1428
```

In [10]:

```
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import transforms
from torchvision.models import vgg19, VGG19_Weights
from torch.utils.data import Dataset, DataLoader
from PIL import Image
from sklearn.metrics import accuracy_score, f1_score, cohen_kappa_score, roc_auc_score, precision_recall_curve,
import time
import numpy as np
import os
import random

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

classes = ["Eyelid", "Normal", "Cataract", "Uveitis", "Conjunctivitis"]
num_classes = len(classes)
class_to_idx = {cls: i for i, cls in enumerate(classes)}

augmented_dir = "/kaggle/working/augmented_all"

data_transforms = {
    'train': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],
                             [0.229, 0.224, 0.225])
    ]),
    'val': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],
                             [0.229, 0.224, 0.225])
    ]),
    'test': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],
                             [0.229, 0.224, 0.225])
    ]),
}

class EyeDataset(Dataset):
    def __init__(self, image_paths, labels, transform=None):
        self.image_paths = image_paths
        self.labels = labels
        self.transform = transform
    def __len__(self):
        return len(self.image_paths)
    def __getitem__(self, idx):
        img = Image.open(self.image_paths[idx]).convert('RGB')
        label = self.labels[idx]
        if self.transform:
            img = self.transform(img)
        return img, label

all_data = []
for cls in classes:
```

```
    class_dir = os.path.join(augmented_dir, cls)
    image_files = [os.path.join(class_dir, f) for f in os.listdir(class_dir)
                   if f.lower().endswith(('.jpg', '.jpeg', '.png'))]
    image_files = sorted(image_files)
    all_data.extend([(img_path, class_to_idx[cls]) for img_path in image_files])

random.shuffle(all_data)

k = 5
fold_size = len(all_data) // k
folds = []
for i in range(k):
    start_idx = i * fold_size
    if i == k - 1:
        fold = all_data[start_idx:]
    else:
        fold = all_data[start_idx:start_idx + fold_size]
    folds.append(fold)

def get_train_val_test(folds, test_fold_index, val_ratio=0.1):
    test_data = folds[test_fold_index]
    train_val_data = [item for i, f in enumerate(folds) if i != test_fold_index for item in f]
    n_val = int(len(train_val_data) * val_ratio)
    val_data = train_val_data[:n_val]
    train_data = train_val_data[n_val:]
    return train_data, val_data, test_data

def create_dataset_loader_from_list(data_list, transform, shuffle=False):
    image_paths = [item[0] for item in data_list]
    labels = [item[1] for item in data_list]
    dataset = EyeDataset(image_paths, labels, transform=transform)
    loader = DataLoader(dataset, batch_size=32, shuffle=shuffle, num_workers=2)
    return dataset, loader

criterion = nn.CrossEntropyLoss()

def train_epoch(model, dataloader, optimizer):
    model.train()
    running_loss = 0.0
    correct = 0
    total = 0
    for inputs, labels in dataloader:
        inputs, labels = inputs.to(device), labels.to(device)
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item() * inputs.size(0)
        _, preds = torch.max(outputs, 1)
        correct += (preds == labels).sum().item()
        total += labels.size(0)
    epoch_loss = running_loss / total
    epoch_acc = correct / total
    return epoch_loss, epoch_acc

def eval_model(model, dataloader):
    model.eval()
    running_loss = 0.0
    correct = 0
    total = 0
    all_labels = []
    all_preds = []
    all_probs = []
    with torch.no_grad():
        for inputs, labels in dataloader:
            inputs, labels = inputs.to(device), labels.to(device)
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            probs = torch.softmax(outputs, dim=1)
            _, preds = torch.max(outputs, 1)
            running_loss += loss.item() * inputs.size(0)
            correct += (preds == labels).sum().item()
            total += labels.size(0)
            all_labels.extend(labels.cpu().numpy())
            all_preds.extend(preds.cpu().numpy())
            all_probs.extend(probs.cpu().numpy())
    epoch_loss = running_loss / total
    epoch_acc = correct / total
    return epoch_loss, epoch_acc, np.array(all_labels), np.array(all_preds), np.array(all_probs)

num_epochs = 30
patience = 5
```

```python
fold_test_accuracies = []

for fold_idx in range(k):
    print(f"\n===== Fold {fold_idx + 1} / {k} =====")
    train_data, val_data, test_data = get_train_val_test(folds, fold_idx)
    train_dataset, train_loader = create_dataset_loader_from_list(train_data, data_transforms['train'], shuffle=
    val_dataset, val_loader = create_dataset_loader_from_list(val_data, data_transforms['val'], shuffle=False)
    test_dataset, test_loader = create_dataset_loader_from_list(test_data, data_transforms['test'], shuffle=Fals

    vgg19_model = vgg19(weights=VGG19_Weights.IMAGENET1K_V1)
    vgg19_model.classifier = nn.Sequential(
        nn.Linear(25088, 4096),
        nn.ReLU(inplace=True),
        nn.Dropout(0.5),
        nn.Linear(4096, 4096),
        nn.ReLU(inplace=True),
        nn.Dropout(0.5),
        nn.Linear(4096, num_classes)
    )
    vgg19_model = vgg19_model.to(device)

    optimizer = optim.Adam(vgg19_model.parameters(), lr=1e-4, weight_decay=1e-4)
    scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, mode='min', patience=3)

    best_val_loss = float('inf')
    epochs_no_improve = 0
    train_start_time = time.time()
    for epoch in range(num_epochs):
        train_loss, train_acc = train_epoch(vgg19_model, train_loader, optimizer)
        val_loss, val_acc, _, _, _ = eval_model(vgg19_model, val_loader)
        scheduler.step(val_loss)
        print(f"Epoch {epoch + 1}/{num_epochs} | Train Loss: {train_loss:.4f} Acc: {train_acc:.4f} | Val Loss: {
        if val_loss < best_val_loss:
            best_val_loss = val_loss
            epochs_no_improve = 0
            torch.save(vgg19_model.state_dict(), f"best_vgg19_fold{fold_idx}.pth")
        else:
            epochs_no_improve += 1
            if epochs_no_improve >= patience:
                print("Early stopping triggered")
                break
    train_end_time = time.time()
    train_time = train_end_time - train_start_time
    print(f"Training time: {train_time:.2f} seconds")

    vgg19_model.load_state_dict(torch.load(f"best_vgg19_fold{fold_idx}.pth"))

    test_loss, test_acc, test_labels, test_preds, test_probs = eval_model(vgg19_model, test_loader)

    acc = accuracy_score(test_labels, test_preds)
    f1 = f1_score(test_labels, test_preds, average='macro')
    kappa = cohen_kappa_score(test_labels, test_preds)

    try:
        auc_macro = roc_auc_score(test_labels, test_probs, multi_class='ovo', average='macro')
    except Exception:
        auc_macro = float('nan')

    pr_auc_per_class = []
    for i in range(num_classes):
        precision, recall, _ = precision_recall_curve(test_labels == i, test_probs[:, i])
        pr_auc_per_class.append(auc(recall, precision))
    pr_auc_macro = np.mean(pr_auc_per_class)

    print(f"Fold {fold_idx + 1} Test Metrics:")
    print(f"Accuracy: {acc:.4f}")
    print(f"F1-score (macro): {f1:.4f}")
    print(f"AUC (macro): {auc_macro:.4f}")
    print(f"PR AUC (macro): {pr_auc_macro:.4f}")
    print(f"Cohen's Kappa: {kappa:.4f}")

    fold_test_accuracies.append(acc)

print(f"\nAverage Test Accuracy over {k} folds: {np.mean(fold_test_accuracies):.4f} ± {np.std(fold_test_accuraci
```

```
===== Fold 1 / 5 =====
Epoch 1/30 | Train Loss: 0.6019 Acc: 0.7761 | Val Loss: 0.3399 Acc: 0.8617
Epoch 2/30 | Train Loss: 0.2481 Acc: 0.9141 | Val Loss: 0.2438 Acc: 0.9213
Epoch 3/30 | Train Loss: 0.1362 Acc: 0.9586 | Val Loss: 0.1251 Acc: 0.9581
Epoch 4/30 | Train Loss: 0.1267 Acc: 0.9594 | Val Loss: 0.2049 Acc: 0.9327
Epoch 5/30 | Train Loss: 0.1028 Acc: 0.9706 | Val Loss: 0.1156 Acc: 0.9581
Epoch 6/30 | Train Loss: 0.0433 Acc: 0.9863 | Val Loss: 0.1245 Acc: 0.9543
Epoch 7/30 | Train Loss: 0.0872 Acc: 0.9716 | Val Loss: 0.1117 Acc: 0.9695
Epoch 8/30 | Train Loss: 0.0519 Acc: 0.9835 | Val Loss: 0.1477 Acc: 0.9569
```

```
Epoch 9/30 | Train Loss: 0.0622 Acc: 0.9799 | Val Loss: 0.1867 Acc: 0.9492
Epoch 10/30 | Train Loss: 0.0468 Acc: 0.9842 | Val Loss: 0.1152 Acc: 0.9657
Epoch 11/30 | Train Loss: 0.0318 Acc: 0.9897 | Val Loss: 0.0373 Acc: 0.9886
Epoch 12/30 | Train Loss: 0.0005 Acc: 1.0000 | Val Loss: 0.0265 Acc: 0.9911
Epoch 13/30 | Train Loss: 0.0001 Acc: 1.0000 | Val Loss: 0.0250 Acc: 0.9937
Epoch 14/30 | Train Loss: 0.0001 Acc: 1.0000 | Val Loss: 0.0247 Acc: 0.9937
Epoch 15/30 | Train Loss: 0.0000 Acc: 1.0000 | Val Loss: 0.0242 Acc: 0.9937
Epoch 16/30 | Train Loss: 0.0000 Acc: 1.0000 | Val Loss: 0.0236 Acc: 0.9937
Epoch 17/30 | Train Loss: 0.0000 Acc: 1.0000 | Val Loss: 0.0238 Acc: 0.9937
Epoch 18/30 | Train Loss: 0.0000 Acc: 1.0000 | Val Loss: 0.0240 Acc: 0.9937
Epoch 19/30 | Train Loss: 0.0001 Acc: 1.0000 | Val Loss: 0.0250 Acc: 0.9937
Epoch 20/30 | Train Loss: 0.0000 Acc: 1.0000 | Val Loss: 0.0227 Acc: 0.9949
Epoch 21/30 | Train Loss: 0.0000 Acc: 1.0000 | Val Loss: 0.0228 Acc: 0.9949
Epoch 22/30 | Train Loss: 0.0000 Acc: 1.0000 | Val Loss: 0.0227 Acc: 0.9949
Epoch 23/30 | Train Loss: 0.0000 Acc: 1.0000 | Val Loss: 0.0221 Acc: 0.9949
Epoch 24/30 | Train Loss: 0.0000 Acc: 1.0000 | Val Loss: 0.0231 Acc: 0.9949
Epoch 25/30 | Train Loss: 0.0000 Acc: 1.0000 | Val Loss: 0.0225 Acc: 0.9949
Epoch 26/30 | Train Loss: 0.0000 Acc: 1.0000 | Val Loss: 0.0245 Acc: 0.9949
Epoch 27/30 | Train Loss: 0.0000 Acc: 1.0000 | Val Loss: 0.0262 Acc: 0.9937
Epoch 28/30 | Train Loss: 0.0000 Acc: 1.0000 | Val Loss: 0.0245 Acc: 0.9949
Early stopping triggered
Training time: 1978.56 seconds
Fold 1 Test Metrics:
Accuracy: 0.9914
F1-score (macro): 0.9905
AUC (macro): 0.9998
PR AUC (macro): 0.9992
Cohen's Kappa: 0.9891

===== Fold 2 / 5 =====
Epoch 1/30 | Train Loss: 0.6351 Acc: 0.7612 | Val Loss: 0.3608 Acc: 0.8896
Epoch 2/30 | Train Loss: 0.2627 Acc: 0.9107 | Val Loss: 0.2956 Acc: 0.8972
Epoch 3/30 | Train Loss: 0.1787 Acc: 0.9403 | Val Loss: 0.3472 Acc: 0.8959
Epoch 4/30 | Train Loss: 0.1049 Acc: 0.9665 | Val Loss: 0.1184 Acc: 0.9569
Epoch 5/30 | Train Loss: 0.0706 Acc: 0.9789 | Val Loss: 0.1955 Acc: 0.9442
Epoch 6/30 | Train Loss: 0.0644 Acc: 0.9806 | Val Loss: 0.1535 Acc: 0.9569
Epoch 7/30 | Train Loss: 0.0614 Acc: 0.9811 | Val Loss: 0.1925 Acc: 0.9365
Epoch 8/30 | Train Loss: 0.0602 Acc: 0.9827 | Val Loss: 0.1263 Acc: 0.9594
Epoch 9/30 | Train Loss: 0.0080 Acc: 0.9975 | Val Loss: 0.0630 Acc: 0.9810
Epoch 10/30 | Train Loss: 0.0011 Acc: 1.0000 | Val Loss: 0.0639 Acc: 0.9822
Epoch 11/30 | Train Loss: 0.0008 Acc: 1.0000 | Val Loss: 0.0641 Acc: 0.9822
Epoch 12/30 | Train Loss: 0.0004 Acc: 1.0000 | Val Loss: 0.0666 Acc: 0.9835
Epoch 13/30 | Train Loss: 0.0003 Acc: 1.0000 | Val Loss: 0.0684 Acc: 0.9822
Epoch 14/30 | Train Loss: 0.0003 Acc: 1.0000 | Val Loss: 0.0686 Acc: 0.9822
Early stopping triggered
Training time: 984.08 seconds
Fold 2 Test Metrics:
Accuracy: 0.9863
F1-score (macro): 0.9841
AUC (macro): 0.9994
PR AUC (macro): 0.9977
Cohen's Kappa: 0.9827

===== Fold 3 / 5 =====
Epoch 1/30 | Train Loss: 0.5900 Acc: 0.7810 | Val Loss: 0.3299 Acc: 0.8896
Epoch 2/30 | Train Loss: 0.2210 Acc: 0.9251 | Val Loss: 0.1967 Acc: 0.9340
Epoch 3/30 | Train Loss: 0.1228 Acc: 0.9596 | Val Loss: 0.2268 Acc: 0.9365
Epoch 4/30 | Train Loss: 0.1222 Acc: 0.9596 | Val Loss: 0.1380 Acc: 0.9569
Epoch 5/30 | Train Loss: 0.0541 Acc: 0.9838 | Val Loss: 0.1348 Acc: 0.9683
Epoch 6/30 | Train Loss: 0.0879 Acc: 0.9739 | Val Loss: 0.0791 Acc: 0.9708
Epoch 7/30 | Train Loss: 0.0622 Acc: 0.9797 | Val Loss: 0.1031 Acc: 0.9645
Epoch 8/30 | Train Loss: 0.0620 Acc: 0.9811 | Val Loss: 0.2235 Acc: 0.9480
Epoch 9/30 | Train Loss: 0.0566 Acc: 0.9818 | Val Loss: 0.1566 Acc: 0.9543
Epoch 10/30 | Train Loss: 0.0358 Acc: 0.9906 | Val Loss: 0.0981 Acc: 0.9784
Epoch 11/30 | Train Loss: 0.0079 Acc: 0.9980 | Val Loss: 0.0659 Acc: 0.9810
Epoch 12/30 | Train Loss: 0.0017 Acc: 0.9997 | Val Loss: 0.0630 Acc: 0.9848
Epoch 13/30 | Train Loss: 0.0006 Acc: 1.0000 | Val Loss: 0.0637 Acc: 0.9835
Epoch 14/30 | Train Loss: 0.0004 Acc: 1.0000 | Val Loss: 0.0635 Acc: 0.9835
Epoch 15/30 | Train Loss: 0.0003 Acc: 1.0000 | Val Loss: 0.0640 Acc: 0.9835
Epoch 16/30 | Train Loss: 0.0002 Acc: 1.0000 | Val Loss: 0.0657 Acc: 0.9835
Epoch 17/30 | Train Loss: 0.0002 Acc: 1.0000 | Val Loss: 0.0657 Acc: 0.9835
Early stopping triggered
Training time: 1198.12 seconds
Fold 3 Test Metrics:
Accuracy: 0.9797
F1-score (macro): 0.9771
AUC (macro): 0.9989
PR AUC (macro): 0.9961
Cohen's Kappa: 0.9743

===== Fold 4 / 5 =====
Epoch 1/30 | Train Loss: 0.6154 Acc: 0.7665 | Val Loss: 0.3211 Acc: 0.8909
Epoch 2/30 | Train Loss: 0.2747 Acc: 0.9058 | Val Loss: 0.2652 Acc: 0.9023
```

```
Epoch 3/30 | Train Loss: 0.1387 Acc: 0.9548 | Val Loss: 0.1505 Acc: 0.9480
Epoch 4/30 | Train Loss: 0.1054 Acc: 0.9683 | Val Loss: 0.1949 Acc: 0.9391
Epoch 5/30 | Train Loss: 0.0714 Acc: 0.9777 | Val Loss: 0.2004 Acc: 0.9518
Epoch 6/30 | Train Loss: 0.0800 Acc: 0.9748 | Val Loss: 0.1526 Acc: 0.9619
Epoch 7/30 | Train Loss: 0.0316 Acc: 0.9901 | Val Loss: 0.0974 Acc: 0.9772
Epoch 8/30 | Train Loss: 0.0498 Acc: 0.9849 | Val Loss: 0.1130 Acc: 0.9746
Epoch 9/30 | Train Loss: 0.0542 Acc: 0.9848 | Val Loss: 0.1457 Acc: 0.9581
Epoch 10/30 | Train Loss: 0.0558 Acc: 0.9820 | Val Loss: 0.0960 Acc: 0.9708
Epoch 11/30 | Train Loss: 0.0503 Acc: 0.9854 | Val Loss: 0.1114 Acc: 0.9695
Epoch 12/30 | Train Loss: 0.0423 Acc: 0.9872 | Val Loss: 0.1346 Acc: 0.9619
Epoch 13/30 | Train Loss: 0.0233 Acc: 0.9934 | Val Loss: 0.0817 Acc: 0.9835
Epoch 14/30 | Train Loss: 0.0051 Acc: 0.9986 | Val Loss: 0.1229 Acc: 0.9619
Epoch 15/30 | Train Loss: 0.0629 Acc: 0.9814 | Val Loss: 0.1018 Acc: 0.9721
Epoch 16/30 | Train Loss: 0.1156 Acc: 0.9668 | Val Loss: 0.1172 Acc: 0.9632
Epoch 17/30 | Train Loss: 0.0281 Acc: 0.9937 | Val Loss: 0.1350 Acc: 0.9695
Epoch 18/30 | Train Loss: 0.0048 Acc: 0.9985 | Val Loss: 0.0785 Acc: 0.9746
Epoch 19/30 | Train Loss: 0.0007 Acc: 1.0000 | Val Loss: 0.0783 Acc: 0.9746
Epoch 20/30 | Train Loss: 0.0005 Acc: 1.0000 | Val Loss: 0.0797 Acc: 0.9746
Epoch 21/30 | Train Loss: 0.0003 Acc: 1.0000 | Val Loss: 0.0802 Acc: 0.9746
Epoch 22/30 | Train Loss: 0.0002 Acc: 1.0000 | Val Loss: 0.0805 Acc: 0.9746
Epoch 23/30 | Train Loss: 0.0002 Acc: 1.0000 | Val Loss: 0.0811 Acc: 0.9772
Epoch 24/30 | Train Loss: 0.0002 Acc: 1.0000 | Val Loss: 0.0811 Acc: 0.9772
Early stopping triggered
Training time: 1690.13 seconds
Fold 4 Test Metrics:
Accuracy: 0.9838
F1-score (macro): 0.9818
AUC (macro): 0.9994
PR AUC (macro): 0.9979
Cohen's Kappa: 0.9794

===== Fold 5 / 5 =====
Epoch 1/30 | Train Loss: 0.5681 Acc: 0.7954 | Val Loss: 0.2426 Acc: 0.9162
Epoch 2/30 | Train Loss: 0.2115 Acc: 0.9268 | Val Loss: 0.2136 Acc: 0.9213
Epoch 3/30 | Train Loss: 0.1254 Acc: 0.9589 | Val Loss: 0.2090 Acc: 0.9353
Epoch 4/30 | Train Loss: 0.1140 Acc: 0.9608 | Val Loss: 0.3562 Acc: 0.8896
Epoch 5/30 | Train Loss: 0.0636 Acc: 0.9790 | Val Loss: 0.1325 Acc: 0.9632
Epoch 6/30 | Train Loss: 0.0894 Acc: 0.9756 | Val Loss: 0.1480 Acc: 0.9670
Epoch 7/30 | Train Loss: 0.0457 Acc: 0.9859 | Val Loss: 0.1842 Acc: 0.9594
Epoch 8/30 | Train Loss: 0.0487 Acc: 0.9854 | Val Loss: 0.1553 Acc: 0.9480
Epoch 9/30 | Train Loss: 0.0568 Acc: 0.9838 | Val Loss: 0.1644 Acc: 0.9480
Epoch 10/30 | Train Loss: 0.0092 Acc: 0.9982 | Val Loss: 0.1010 Acc: 0.9721
Epoch 11/30 | Train Loss: 0.0012 Acc: 0.9999 | Val Loss: 0.1047 Acc: 0.9734
Epoch 12/30 | Train Loss: 0.0007 Acc: 1.0000 | Val Loss: 0.1085 Acc: 0.9734
Epoch 13/30 | Train Loss: 0.0004 Acc: 1.0000 | Val Loss: 0.1106 Acc: 0.9734
Epoch 14/30 | Train Loss: 0.0004 Acc: 1.0000 | Val Loss: 0.1137 Acc: 0.9734
Epoch 15/30 | Train Loss: 0.0003 Acc: 1.0000 | Val Loss: 0.1139 Acc: 0.9734
Early stopping triggered
Training time: 1056.36 seconds
Fold 5 Test Metrics:
Accuracy: 0.9838
F1-score (macro): 0.9824
AUC (macro): 0.9988
PR AUC (macro): 0.9960
Cohen's Kappa: 0.9795

Average Test Accuracy over 5 folds: 0.9850 ± 0.0038
```

In [11]:

```python
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import transforms
from torchvision.models import mobilenet_v2, MobileNet_V2_Weights
from torch.utils.data import Dataset, DataLoader
from PIL import Image
from sklearn.metrics import (
    accuracy_score, f1_score, cohen_kappa_score, roc_auc_score,
    precision_recall_curve, auc, classification_report, confusion_matrix, roc_curve
)
import time
import numpy as np
import os
import random
import matplotlib.pyplot as plt

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

classes = ["Eyelid", "Normal", "Cataract", "Uveitis", "Conjunctivitis"]
num_classes = len(classes)
class_to_idx = {cls: i for i, cls in enumerate(classes)}

augmented_dir = "/kaggle/working/augmented_all"
```

```
data_transforms = {
    'train': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],
                             [0.229, 0.224, 0.225])
    ]),
    'val': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],
                             [0.229, 0.224, 0.225])
    ]),
    'test': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],
                             [0.229, 0.224, 0.225])
    ]),
}

class EyeDataset(Dataset):
    def __init__(self, image_paths, labels, transform=None):
        self.image_paths = image_paths
        self.labels = labels
        self.transform = transform
    def __len__(self):
        return len(self.image_paths)
    def __getitem__(self, idx):
        img = Image.open(self.image_paths[idx]).convert('RGB')
        label = self.labels[idx]
        if self.transform:
            img = self.transform(img)
        return img, label

# Prepare all data (image paths and labels)
all_data = []
for cls in classes:
    class_dir = os.path.join(augmented_dir, cls)
    image_files = [os.path.join(class_dir, f) for f in os.listdir(class_dir)
                   if f.lower().endswith(('.jpg', '.jpeg', '.png'))]
    image_files = sorted(image_files)
    all_data.extend([(img_path, class_to_idx[cls]) for img_path in image_files])

random.shuffle(all_data)

k = 5
fold_size = len(all_data) // k
folds = []
for i in range(k):
    start_idx = i * fold_size
    if i == k - 1:
        fold = all_data[start_idx:]
    else:
        fold = all_data[start_idx:start_idx + fold_size]
    folds.append(fold)

def get_train_val_test(folds, test_fold_index, val_ratio=0.1):
    test_data = folds[test_fold_index]
    train_val_data = [item for i, f in enumerate(folds) if i != test_fold_index for item in f]
    n_val = int(len(train_val_data) * val_ratio)
    val_data = train_val_data[:n_val]
    train_data = train_val_data[n_val:]
    return train_data, val_data, test_data

def create_dataset_loader_from_list(data_list, transform, shuffle=False):
    image_paths = [item[0] for item in data_list]
    labels = [item[1] for item in data_list]
    dataset = EyeDataset(image_paths, labels, transform=transform)
    loader = DataLoader(dataset, batch_size=32, shuffle=shuffle, num_workers=2)
    return dataset, loader

criterion = nn.CrossEntropyLoss()
num_epochs = 30
patience = 5

def train_epoch(model, dataloader, optimizer):
    model.train()
    running_loss = 0.0
    correct = 0
    total = 0
    for inputs, labels in dataloader:
        inputs, labels = inputs.to(device), labels.to(device)
```

```
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item() * inputs.size(0)
        _, preds = torch.max(outputs, 1)
        correct += (preds == labels).sum().item()
        total += labels.size(0)
    return running_loss / total, correct / total

def eval_model(model, dataloader):
    model.eval()
    running_loss = 0.0
    correct = 0
    total = 0
    all_labels = []
    all_preds = []
    all_probs = []
    with torch.no_grad():
        for inputs, labels in dataloader:
            inputs, labels = inputs.to(device), labels.to(device)
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            probs = torch.softmax(outputs, dim=1)
            _, preds = torch.max(outputs, 1)
            running_loss += loss.item() * inputs.size(0)
            correct += (preds == labels).sum().item()
            total += labels.size(0)
            all_labels.extend(labels.cpu().numpy())
            all_preds.extend(preds.cpu().numpy())
            all_probs.extend(probs.cpu().numpy())
    return running_loss / total, correct / total, np.array(all_labels), np.array(all_preds), np.array(all_probs)

fold_test_accuracies = []

for fold_idx in range(k):
    print(f"\n===== Fold {fold_idx + 1} / {k} =====")
    train_data, val_data, test_data = get_train_val_test(folds, fold_idx)
    train_dataset, train_loader = create_dataset_loader_from_list(train_data, data_transforms['train'], shuffle=
    val_dataset, val_loader = create_dataset_loader_from_list(val_data, data_transforms['val'], shuffle=False)
    test_dataset, test_loader = create_dataset_loader_from_list(test_data, data_transforms['test'], shuffle=Fals

    mobilenet_model = mobilenet_v2(weights=MobileNet_V2_Weights.IMAGENET1K_V1)
    mobilenet_model.classifier[1] = nn.Linear(mobilenet_model.classifier[1].in_features, num_classes)
    mobilenet_model = mobilenet_model.to(device)

    optimizer = optim.Adam(mobilenet_model.parameters(), lr=1e-4, weight_decay=1e-4)
    scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, mode='min', patience=3)

    best_val_loss = float('inf')
    epochs_no_improve = 0
    train_start_time = time.time()
    for epoch in range(num_epochs):
        start_epoch_time = time.time()
        train_loss, train_acc = train_epoch(mobilenet_model, train_loader, optimizer)
        val_loss, val_acc, _, _, _ = eval_model(mobilenet_model, val_loader)
        scheduler.step(val_loss)
        epoch_time = time.time() - start_epoch_time
        print(f"Epoch {epoch+1}/{num_epochs} | Train Loss: {train_loss:.4f} Acc: {train_acc:.4f} | "
              f"Val Loss: {val_loss:.4f} Acc: {val_acc:.4f} | Time: {epoch_time:.2f}s")
        if val_loss < best_val_loss:
            best_val_loss = val_loss
            epochs_no_improve = 0
            torch.save(mobilenet_model.state_dict(), f"best_mobilenetv2_fold{fold_idx}.pth")
        else:
            epochs_no_improve += 1
            if epochs_no_improve >= patience:
                print("Early stopping triggered")
                break
    train_end_time = time.time()
    print(f"Training time fold {fold_idx+1}: {train_end_time - train_start_time:.2f} seconds")

    mobilenet_model.load_state_dict(torch.load(f"best_mobilenetv2_fold{fold_idx}.pth"))

    test_loss, test_acc, test_labels, test_preds, test_probs = eval_model(mobilenet_model, test_loader)

    acc = accuracy_score(test_labels, test_preds)
    f1 = f1_score(test_labels, test_preds, average='macro')
    kappa = cohen_kappa_score(test_labels, test_preds)

    try:
        auc_macro = roc_auc_score(test_labels, test_probs, multi_class='ovo', average='macro')
```

```
    except Exception:
        auc_macro = float('nan')

    pr_auc_per_class = []
    for i in range(num_classes):
        precision, recall, _ = precision_recall_curve(test_labels == i, test_probs[:, i])
        pr_auc_per_class.append(auc(recall, precision))
    pr_auc_macro = np.mean(pr_auc_per_class)

    print(f"Fold {fold_idx + 1} Test Metrics:")
    print(f"Accuracy: {acc:.4f}")
    print(f"F1-score (macro): {f1:.4f}")
    print(f"AUC (macro): {auc_macro:.4f}")
    print(f"PR AUC (macro): {pr_auc_macro:.4f}")
    print(f"Cohen's Kappa: {kappa:.4f}")

    fold_test_accuracies.append(acc)

print(f"\nAverage Test Accuracy over {k} folds: {np.mean(fold_test_accuracies):.4f} ± {np.std(fold_test_accuraci
```

```
===== Fold 1 / 5 =====
Downloading: "https://download.pytorch.org/models/mobilenet_v2-b0353104.pth" to /root/.cache/torch/hub/checkpoin
ts/mobilenet_v2-b0353104.pth
100%|██████████| 13.6M/13.6M [00:00<00:00, 80.6MB/s]
Epoch 1/30 | Train Loss: 0.4801 Acc: 0.8316 | Val Loss: 0.2282 Acc: 0.9150 | Time: 21.62s
Epoch 2/30 | Train Loss: 0.1204 Acc: 0.9610 | Val Loss: 0.1243 Acc: 0.9556 | Time: 21.65s
Epoch 3/30 | Train Loss: 0.0567 Acc: 0.9832 | Val Loss: 0.0717 Acc: 0.9721 | Time: 21.40s
Epoch 4/30 | Train Loss: 0.0326 Acc: 0.9906 | Val Loss: 0.1831 Acc: 0.9442 | Time: 21.46s
Epoch 5/30 | Train Loss: 0.0297 Acc: 0.9920 | Val Loss: 0.0426 Acc: 0.9860 | Time: 21.56s
Epoch 6/30 | Train Loss: 0.0143 Acc: 0.9969 | Val Loss: 0.0385 Acc: 0.9848 | Time: 21.48s
Epoch 7/30 | Train Loss: 0.0141 Acc: 0.9961 | Val Loss: 0.1006 Acc: 0.9721 | Time: 21.50s
Epoch 8/30 | Train Loss: 0.0270 Acc: 0.9904 | Val Loss: 0.0553 Acc: 0.9759 | Time: 21.55s
Epoch 9/30 | Train Loss: 0.0254 Acc: 0.9925 | Val Loss: 0.0493 Acc: 0.9835 | Time: 21.42s
Epoch 10/30 | Train Loss: 0.0105 Acc: 0.9970 | Val Loss: 0.0475 Acc: 0.9848 | Time: 21.53s
Epoch 11/30 | Train Loss: 0.0049 Acc: 0.9993 | Val Loss: 0.0319 Acc: 0.9873 | Time: 21.60s
Epoch 12/30 | Train Loss: 0.0037 Acc: 0.9994 | Val Loss: 0.0285 Acc: 0.9873 | Time: 21.50s
Epoch 13/30 | Train Loss: 0.0022 Acc: 0.9999 | Val Loss: 0.0290 Acc: 0.9886 | Time: 21.50s
Epoch 14/30 | Train Loss: 0.0023 Acc: 0.9997 | Val Loss: 0.0279 Acc: 0.9898 | Time: 21.39s
Epoch 15/30 | Train Loss: 0.0019 Acc: 0.9997 | Val Loss: 0.0243 Acc: 0.9924 | Time: 21.46s
Epoch 16/30 | Train Loss: 0.0018 Acc: 0.9997 | Val Loss: 0.0217 Acc: 0.9937 | Time: 21.62s
Epoch 17/30 | Train Loss: 0.0017 Acc: 0.9997 | Val Loss: 0.0196 Acc: 0.9949 | Time: 21.50s
Epoch 18/30 | Train Loss: 0.0014 Acc: 0.9999 | Val Loss: 0.0216 Acc: 0.9924 | Time: 21.73s
Epoch 19/30 | Train Loss: 0.0017 Acc: 0.9994 | Val Loss: 0.0203 Acc: 0.9911 | Time: 21.62s
Epoch 20/30 | Train Loss: 0.0012 Acc: 0.9999 | Val Loss: 0.0184 Acc: 0.9937 | Time: 21.62s
Epoch 21/30 | Train Loss: 0.0016 Acc: 0.9997 | Val Loss: 0.0165 Acc: 0.9924 | Time: 21.53s
Epoch 22/30 | Train Loss: 0.0008 Acc: 1.0000 | Val Loss: 0.0191 Acc: 0.9962 | Time: 21.43s
Epoch 23/30 | Train Loss: 0.0009 Acc: 0.9999 | Val Loss: 0.0188 Acc: 0.9937 | Time: 21.58s
Epoch 24/30 | Train Loss: 0.0011 Acc: 0.9999 | Val Loss: 0.0196 Acc: 0.9924 | Time: 21.49s
Epoch 25/30 | Train Loss: 0.0010 Acc: 0.9999 | Val Loss: 0.0189 Acc: 0.9911 | Time: 21.44s
Epoch 26/30 | Train Loss: 0.0011 Acc: 0.9999 | Val Loss: 0.0186 Acc: 0.9949 | Time: 21.40s
Early stopping triggered
Training time fold 1: 560.16 seconds
Fold 1 Test Metrics:
Accuracy: 0.9954
F1-score (macro): 0.9948
AUC (macro): 0.9999
PR AUC (macro): 0.9996
Cohen's Kappa: 0.9942

===== Fold 2 / 5 =====
Epoch 1/30 | Train Loss: 0.4807 Acc: 0.8285 | Val Loss: 0.2267 Acc: 0.9239 | Time: 21.39s
Epoch 2/30 | Train Loss: 0.1284 Acc: 0.9592 | Val Loss: 0.1742 Acc: 0.9340 | Time: 21.47s
Epoch 3/30 | Train Loss: 0.0568 Acc: 0.9847 | Val Loss: 0.1089 Acc: 0.9607 | Time: 21.65s
Epoch 4/30 | Train Loss: 0.0370 Acc: 0.9897 | Val Loss: 0.1074 Acc: 0.9594 | Time: 21.42s
Epoch 5/30 | Train Loss: 0.0175 Acc: 0.9961 | Val Loss: 0.0807 Acc: 0.9746 | Time: 21.79s
Epoch 6/30 | Train Loss: 0.0124 Acc: 0.9970 | Val Loss: 0.0719 Acc: 0.9784 | Time: 21.67s
Epoch 7/30 | Train Loss: 0.0316 Acc: 0.9896 | Val Loss: 0.0998 Acc: 0.9734 | Time: 21.38s
Epoch 8/30 | Train Loss: 0.0343 Acc: 0.9880 | Val Loss: 0.0802 Acc: 0.9657 | Time: 21.67s
Epoch 9/30 | Train Loss: 0.0151 Acc: 0.9958 | Val Loss: 0.0679 Acc: 0.9746 | Time: 21.42s
Epoch 10/30 | Train Loss: 0.0126 Acc: 0.9969 | Val Loss: 0.0570 Acc: 0.9848 | Time: 21.44s
Epoch 11/30 | Train Loss: 0.0052 Acc: 0.9990 | Val Loss: 0.0324 Acc: 0.9886 | Time: 21.50s
Epoch 12/30 | Train Loss: 0.0082 Acc: 0.9976 | Val Loss: 0.0853 Acc: 0.9657 | Time: 21.66s
Epoch 13/30 | Train Loss: 0.0100 Acc: 0.9966 | Val Loss: 0.0851 Acc: 0.9759 | Time: 21.55s
Epoch 14/30 | Train Loss: 0.0102 Acc: 0.9972 | Val Loss: 0.0820 Acc: 0.9657 | Time: 21.72s
Epoch 15/30 | Train Loss: 0.0292 Acc: 0.9911 | Val Loss: 0.0843 Acc: 0.9670 | Time: 21.52s
Epoch 16/30 | Train Loss: 0.0099 Acc: 0.9979 | Val Loss: 0.0542 Acc: 0.9784 | Time: 21.50s
Early stopping triggered
Training time fold 2: 345.17 seconds
Fold 2 Test Metrics:
Accuracy: 0.9899
F1-score (macro): 0.9886
AUC (macro): 0.9996
PR AUC (macro): 0.9985
```

```
Cohen's Kappa: 0.9872

===== Fold 3 / 5 =====
Epoch 1/30 | Train Loss: 0.4835 Acc: 0.8276 | Val Loss: 0.2281 Acc: 0.9188 | Time: 21.47s
Epoch 2/30 | Train Loss: 0.1215 Acc: 0.9628 | Val Loss: 0.1084 Acc: 0.9683 | Time: 21.50s
Epoch 3/30 | Train Loss: 0.0581 Acc: 0.9823 | Val Loss: 0.0665 Acc: 0.9797 | Time: 21.67s
Epoch 4/30 | Train Loss: 0.0306 Acc: 0.9907 | Val Loss: 0.0652 Acc: 0.9708 | Time: 21.39s
Epoch 5/30 | Train Loss: 0.0233 Acc: 0.9934 | Val Loss: 0.0407 Acc: 0.9873 | Time: 21.41s
Epoch 6/30 | Train Loss: 0.0199 Acc: 0.9941 | Val Loss: 0.0566 Acc: 0.9784 | Time: 21.48s
Epoch 7/30 | Train Loss: 0.0176 Acc: 0.9942 | Val Loss: 0.0666 Acc: 0.9784 | Time: 21.41s
Epoch 8/30 | Train Loss: 0.0192 Acc: 0.9951 | Val Loss: 0.0465 Acc: 0.9848 | Time: 21.67s
Epoch 9/30 | Train Loss: 0.0077 Acc: 0.9985 | Val Loss: 0.0335 Acc: 0.9898 | Time: 21.60s
Epoch 10/30 | Train Loss: 0.0074 Acc: 0.9985 | Val Loss: 0.0542 Acc: 0.9822 | Time: 21.52s
Epoch 11/30 | Train Loss: 0.0205 Acc: 0.9934 | Val Loss: 0.1099 Acc: 0.9619 | Time: 21.52s
Epoch 12/30 | Train Loss: 0.0221 Acc: 0.9930 | Val Loss: 0.0514 Acc: 0.9784 | Time: 21.54s
Epoch 13/30 | Train Loss: 0.0253 Acc: 0.9924 | Val Loss: 0.0482 Acc: 0.9873 | Time: 21.49s
Epoch 14/30 | Train Loss: 0.0131 Acc: 0.9965 | Val Loss: 0.0256 Acc: 0.9937 | Time: 21.44s
Epoch 15/30 | Train Loss: 0.0037 Acc: 0.9996 | Val Loss: 0.0266 Acc: 0.9898 | Time: 21.48s
Epoch 16/30 | Train Loss: 0.0028 Acc: 0.9997 | Val Loss: 0.0244 Acc: 0.9924 | Time: 21.46s
Epoch 17/30 | Train Loss: 0.0034 Acc: 0.9996 | Val Loss: 0.0203 Acc: 0.9949 | Time: 21.30s
Epoch 18/30 | Train Loss: 0.0019 Acc: 0.9999 | Val Loss: 0.0202 Acc: 0.9949 | Time: 21.54s
Epoch 19/30 | Train Loss: 0.0016 Acc: 0.9999 | Val Loss: 0.0226 Acc: 0.9937 | Time: 21.50s
Epoch 20/30 | Train Loss: 0.0021 Acc: 0.9999 | Val Loss: 0.0259 Acc: 0.9911 | Time: 21.62s
Epoch 21/30 | Train Loss: 0.0015 Acc: 1.0000 | Val Loss: 0.0192 Acc: 0.9949 | Time: 21.52s
Epoch 22/30 | Train Loss: 0.0014 Acc: 0.9999 | Val Loss: 0.0169 Acc: 0.9975 | Time: 21.45s
Epoch 23/30 | Train Loss: 0.0009 Acc: 1.0000 | Val Loss: 0.0186 Acc: 0.9949 | Time: 21.55s
Epoch 24/30 | Train Loss: 0.0011 Acc: 1.0000 | Val Loss: 0.0160 Acc: 0.9962 | Time: 21.44s
Epoch 25/30 | Train Loss: 0.0012 Acc: 0.9999 | Val Loss: 0.0175 Acc: 0.9949 | Time: 21.46s
Epoch 26/30 | Train Loss: 0.0016 Acc: 0.9997 | Val Loss: 0.0161 Acc: 0.9962 | Time: 21.61s
Epoch 27/30 | Train Loss: 0.0007 Acc: 1.0000 | Val Loss: 0.0158 Acc: 0.9962 | Time: 21.42s
Epoch 28/30 | Train Loss: 0.0007 Acc: 1.0000 | Val Loss: 0.0196 Acc: 0.9949 | Time: 21.38s
Epoch 29/30 | Train Loss: 0.0006 Acc: 1.0000 | Val Loss: 0.0178 Acc: 0.9962 | Time: 21.41s
Epoch 30/30 | Train Loss: 0.0006 Acc: 1.0000 | Val Loss: 0.0174 Acc: 0.9937 | Time: 21.52s
Training time fold 3: 645.42 seconds
Fold 3 Test Metrics:
Accuracy: 0.9934
F1-score (macro): 0.9932
AUC (macro): 0.9996
PR AUC (macro): 0.9988
Cohen's Kappa: 0.9917

===== Fold 4 / 5 =====
Epoch 1/30 | Train Loss: 0.4709 Acc: 0.8316 | Val Loss: 0.2159 Acc: 0.9264 | Time: 21.54s
Epoch 2/30 | Train Loss: 0.1213 Acc: 0.9621 | Val Loss: 0.1326 Acc: 0.9543 | Time: 21.54s
Epoch 3/30 | Train Loss: 0.0538 Acc: 0.9848 | Val Loss: 0.0931 Acc: 0.9632 | Time: 21.61s
Epoch 4/30 | Train Loss: 0.0332 Acc: 0.9907 | Val Loss: 0.1202 Acc: 0.9543 | Time: 21.43s
Epoch 5/30 | Train Loss: 0.0364 Acc: 0.9894 | Val Loss: 0.0707 Acc: 0.9759 | Time: 21.53s
Epoch 6/30 | Train Loss: 0.0160 Acc: 0.9955 | Val Loss: 0.0989 Acc: 0.9734 | Time: 21.68s
Epoch 7/30 | Train Loss: 0.0156 Acc: 0.9951 | Val Loss: 0.0733 Acc: 0.9784 | Time: 21.43s
Epoch 8/30 | Train Loss: 0.0193 Acc: 0.9944 | Val Loss: 0.0857 Acc: 0.9645 | Time: 21.66s
Epoch 9/30 | Train Loss: 0.0137 Acc: 0.9970 | Val Loss: 0.0254 Acc: 0.9886 | Time: 21.64s
Epoch 10/30 | Train Loss: 0.0055 Acc: 0.9989 | Val Loss: 0.0377 Acc: 0.9810 | Time: 21.49s
Epoch 11/30 | Train Loss: 0.0129 Acc: 0.9952 | Val Loss: 0.0551 Acc: 0.9784 | Time: 21.52s
Epoch 12/30 | Train Loss: 0.0275 Acc: 0.9910 | Val Loss: 0.0517 Acc: 0.9810 | Time: 21.62s
Epoch 13/30 | Train Loss: 0.0284 Acc: 0.9901 | Val Loss: 0.1206 Acc: 0.9645 | Time: 21.30s
Epoch 14/30 | Train Loss: 0.0102 Acc: 0.9973 | Val Loss: 0.0412 Acc: 0.9835 | Time: 21.53s
Early stopping triggered
Training time fold 4: 301.97 seconds
Fold 4 Test Metrics:
Accuracy: 0.9873
F1-score (macro): 0.9855
AUC (macro): 0.9996
PR AUC (macro): 0.9984
Cohen's Kappa: 0.9840

===== Fold 5 / 5 =====
Epoch 1/30 | Train Loss: 0.4741 Acc: 0.8325 | Val Loss: 0.2207 Acc: 0.9213 | Time: 21.64s
Epoch 2/30 | Train Loss: 0.1205 Acc: 0.9620 | Val Loss: 0.1246 Acc: 0.9505 | Time: 21.53s
Epoch 3/30 | Train Loss: 0.0573 Acc: 0.9820 | Val Loss: 0.0980 Acc: 0.9683 | Time: 21.65s
Epoch 4/30 | Train Loss: 0.0280 Acc: 0.9935 | Val Loss: 0.0746 Acc: 0.9721 | Time: 21.59s
Epoch 5/30 | Train Loss: 0.0194 Acc: 0.9956 | Val Loss: 0.0601 Acc: 0.9784 | Time: 21.68s
Epoch 6/30 | Train Loss: 0.0114 Acc: 0.9966 | Val Loss: 0.0656 Acc: 0.9797 | Time: 21.49s
Epoch 7/30 | Train Loss: 0.0235 Acc: 0.9935 | Val Loss: 0.0694 Acc: 0.9772 | Time: 21.75s
Epoch 8/30 | Train Loss: 0.0354 Acc: 0.9892 | Val Loss: 0.1001 Acc: 0.9683 | Time: 21.62s
Epoch 9/30 | Train Loss: 0.0229 Acc: 0.9928 | Val Loss: 0.0579 Acc: 0.9848 | Time: 21.66s
Epoch 10/30 | Train Loss: 0.0187 Acc: 0.9935 | Val Loss: 0.0523 Acc: 0.9848 | Time: 21.51s
Epoch 11/30 | Train Loss: 0.0116 Acc: 0.9959 | Val Loss: 0.0302 Acc: 0.9886 | Time: 21.72s
Epoch 12/30 | Train Loss: 0.0047 Acc: 0.9990 | Val Loss: 0.0339 Acc: 0.9898 | Time: 21.57s
Epoch 13/30 | Train Loss: 0.0204 Acc: 0.9935 | Val Loss: 0.0506 Acc: 0.9797 | Time: 21.54s
Epoch 14/30 | Train Loss: 0.0136 Acc: 0.9952 | Val Loss: 0.0279 Acc: 0.9911 | Time: 21.54s
Epoch 15/30 | Train Loss: 0.0128 Acc: 0.9963 | Val Loss: 0.0343 Acc: 0.9886 | Time: 21.71s
Epoch 16/30 | Train Loss: 0.0049 Acc: 0.9986 | Val Loss: 0.0395 Acc: 0.9848 | Time: 21.60s
Epoch 17/30 | Train Loss: 0.0083 Acc: 0.9979 | Val Loss: 0.0307 Acc: 0.9924 | Time: 21.54s
```

```
Epoch 18/30 | Train Loss: 0.0054 Acc: 0.9990 | Val Loss: 0.0378 Acc: 0.9797 | Time: 21.53s
Epoch 19/30 | Train Loss: 0.0026 Acc: 0.9993 | Val Loss: 0.0317 Acc: 0.9848 | Time: 21.52s
Early stopping triggered
Training time fold 5: 410.81 seconds
Fold 5 Test Metrics:
Accuracy: 0.9863
F1-score (macro): 0.9850
AUC (macro): 0.9998
PR AUC (macro): 0.9992
Cohen's Kappa: 0.9826

Average Test Accuracy over 5 folds: 0.9905 ± 0.0035
```

In [12]:

```python
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import transforms
from torchvision.models import shufflenet_v2_x1_0, ShuffleNet_V2_X1_0_Weights
from torch.utils.data import Dataset, DataLoader
from PIL import Image
from sklearn.metrics import (
    accuracy_score, f1_score, cohen_kappa_score, roc_auc_score,
    precision_recall_curve, auc, classification_report, confusion_matrix, roc_curve
)
import time
import numpy as np
import os
import random
import matplotlib.pyplot as plt

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

classes = ["Eyelid", "Normal", "Cataract", "Uveitis", "Conjunctivitis"]
num_classes = len(classes)
class_to_idx = {cls: i for i, cls in enumerate(classes)}

augmented_dir = "/kaggle/working/augmented_all"

data_transforms = {
    'train': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],
                             [0.229, 0.224, 0.225])
    ]),
    'val': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],
                             [0.229, 0.224, 0.225])
    ]),
    'test': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],
                             [0.229, 0.224, 0.225])
    ]),
}

class EyeDataset(Dataset):
    def __init__(self, image_paths, labels, transform=None):
        self.image_paths = image_paths
        self.labels = labels
        self.transform = transform
    def __len__(self):
        return len(self.image_paths)
    def __getitem__(self, idx):
        img = Image.open(self.image_paths[idx]).convert('RGB')
        label = self.labels[idx]
        if self.transform:
            img = self.transform(img)
        return img, label

all_data = []
for cls in classes:
    class_dir = os.path.join(augmented_dir, cls)
    image_files = [os.path.join(class_dir, f) for f in os.listdir(class_dir)
                   if f.lower().endswith(('.jpg', '.jpeg', '.png'))]
    image_files = sorted(image_files)
    all_data.extend([(img_path, class_to_idx[cls]) for img_path in image_files])

random.shuffle(all_data)
```

```
k = 5
fold_size = len(all_data) // k
folds = []
for i in range(k):
    start_idx = i * fold_size
    if i == k - 1:
        fold = all_data[start_idx:]
    else:
        fold = all_data[start_idx:start_idx + fold_size]
    folds.append(fold)

def get_train_val_test(folds, test_fold_index, val_ratio=0.1):
    test_data = folds[test_fold_index]
    train_val_data = [item for i, f in enumerate(folds) if i != test_fold_index for item in f]
    n_val = int(len(train_val_data) * val_ratio)
    val_data = train_val_data[:n_val]
    train_data = train_val_data[n_val:]
    return train_data, val_data, test_data

def create_dataset_loader_from_list(data_list, transform, shuffle=False):
    image_paths = [item[0] for item in data_list]
    labels = [item[1] for item in data_list]
    dataset = EyeDataset(image_paths, labels, transform=transform)
    loader = DataLoader(dataset, batch_size=32, shuffle=shuffle, num_workers=2)
    return dataset, loader

criterion = nn.CrossEntropyLoss()
num_epochs = 30
patience = 5

def train_epoch(model, dataloader, optimizer):
    model.train()
    running_loss = 0.0
    correct = 0
    total = 0
    for inputs, labels in dataloader:
        inputs, labels = inputs.to(device), labels.to(device)
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item() * inputs.size(0)
        _, preds = torch.max(outputs, 1)
        correct += (preds == labels).sum().item()
        total += labels.size(0)
    return running_loss / total, correct / total

def eval_model(model, dataloader):
    model.eval()
    running_loss = 0.0
    correct = 0
    total = 0
    all_labels = []
    all_preds = []
    all_probs = []
    with torch.no_grad():
        for inputs, labels in dataloader:
            inputs, labels = inputs.to(device), labels.to(device)
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            probs = torch.softmax(outputs, dim=1)
            _, preds = torch.max(outputs, 1)
            running_loss += loss.item() * inputs.size(0)
            correct += (preds == labels).sum().item()
            total += labels.size(0)
            all_labels.extend(labels.cpu().numpy())
            all_preds.extend(preds.cpu().numpy())
            all_probs.extend(probs.cpu().numpy())
    return running_loss / total, correct / total, np.array(all_labels), np.array(all_preds), np.array(all_probs)

fold_test_accuracies = []

for fold_idx in range(k):
    print(f"\n===== Fold {fold_idx + 1} / {k} =====")
    train_data, val_data, test_data = get_train_val_test(folds, fold_idx)
    train_dataset, train_loader = create_dataset_loader_from_list(train_data, data_transforms['train'], shuffle=
    val_dataset, val_loader = create_dataset_loader_from_list(val_data, data_transforms['val'], shuffle=False)
    test_dataset, test_loader = create_dataset_loader_from_list(test_data, data_transforms['test'], shuffle=Fals

    shufflenet_model = shufflenet_v2_x1_0(weights=ShuffleNet_V2_X1_0_Weights.IMAGENET1K_V1)
    shufflenet_model.fc = nn.Linear(shufflenet_model.fc.in_features, num_classes)
    shufflenet_model = shufflenet_model.to(device)
```

```
    optimizer = optim.Adam(shufflenet_model.parameters(), lr=1e-4, weight_decay=1e-4)
    scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, mode='min', patience=3)

    best_val_loss = float('inf')
    epochs_no_improve = 0
    train_start_time = time.time()
    for epoch in range(num_epochs):
        start_epoch_time = time.time()
        train_loss, train_acc = train_epoch(shufflenet_model, train_loader, optimizer)
        val_loss, val_acc, _, _, _ = eval_model(shufflenet_model, val_loader)
        scheduler.step(val_loss)
        epoch_time = time.time() - start_epoch_time
        print(f"Epoch {epoch+1}/{num_epochs} | Train Loss: {train_loss:.4f} Acc: {train_acc:.4f} | "
              f"Val Loss: {val_loss:.4f} Acc: {val_acc:.4f} | Time: {epoch_time:.2f}s")
        if val_loss < best_val_loss:
            best_val_loss = val_loss
            epochs_no_improve = 0
            torch.save(shufflenet_model.state_dict(), f"best_shufflenetv2_fold{fold_idx}.pth")
        else:
            epochs_no_improve += 1
            if epochs_no_improve >= patience:
                print("Early stopping triggered")
                break
    train_end_time = time.time()
    print(f"Training time fold {fold_idx+1}: {train_end_time - train_start_time:.2f} seconds")

    shufflenet_model.load_state_dict(torch.load(f"best_shufflenetv2_fold{fold_idx}.pth"))

    test_loss, test_acc, test_labels, test_preds, test_probs = eval_model(shufflenet_model, test_loader)

    acc = accuracy_score(test_labels, test_preds)
    f1 = f1_score(test_labels, test_preds, average='macro')
    kappa = cohen_kappa_score(test_labels, test_preds)

    try:
        auc_macro = roc_auc_score(test_labels, test_probs, multi_class='ovo', average='macro')
    except Exception:
        auc_macro = float('nan')

    pr_auc_per_class = []
    for i in range(num_classes):
        precision, recall, _ = precision_recall_curve(test_labels == i, test_probs[:, i])
        pr_auc_per_class.append(auc(recall, precision))
    pr_auc_macro = np.mean(pr_auc_per_class)

    print(f"Fold {fold_idx + 1} Test Metrics:")
    print(f"Accuracy: {acc:.4f}")
    print(f"F1-score (macro): {f1:.4f}")
    print(f"AUC (macro): {auc_macro:.4f}")
    print(f"PR AUC (macro): {pr_auc_macro:.4f}")
    print(f"Cohen's Kappa: {kappa:.4f}")

    fold_test_accuracies.append(acc)

print(f"\nAverage Test Accuracy over {k} folds: {np.mean(fold_test_accuracies):.4f} ± {np.std(fold_test_accuraci
```

```
===== Fold 1 / 5 =====
Downloading: "https://download.pytorch.org/models/shufflenetv2_x1-5666bf0f80.pth" to /root/.cache/torch/hub/chec
kpoints/shufflenetv2_x1-5666bf0f80.pth
100%|██████████| 8.79M/8.79M [00:00<00:00, 88.3MB/s]
Epoch 1/30 | Train Loss: 1.2586 Acc: 0.5758 | Val Loss: 0.8865 Acc: 0.7525 | Time: 12.87s
Epoch 2/30 | Train Loss: 0.6617 Acc: 0.8047 | Val Loss: 0.4351 Acc: 0.8921 | Time: 13.01s
Epoch 3/30 | Train Loss: 0.3540 Acc: 0.9062 | Val Loss: 0.2434 Acc: 0.9404 | Time: 13.15s
Epoch 4/30 | Train Loss: 0.2054 Acc: 0.9454 | Val Loss: 0.1518 Acc: 0.9581 | Time: 13.23s
Epoch 5/30 | Train Loss: 0.1196 Acc: 0.9717 | Val Loss: 0.0933 Acc: 0.9734 | Time: 13.25s
Epoch 6/30 | Train Loss: 0.0848 Acc: 0.9810 | Val Loss: 0.0917 Acc: 0.9708 | Time: 12.89s
Epoch 7/30 | Train Loss: 0.0503 Acc: 0.9897 | Val Loss: 0.0659 Acc: 0.9797 | Time: 13.16s
Epoch 8/30 | Train Loss: 0.0405 Acc: 0.9923 | Val Loss: 0.0638 Acc: 0.9797 | Time: 12.93s
Epoch 9/30 | Train Loss: 0.0346 Acc: 0.9930 | Val Loss: 0.0756 Acc: 0.9759 | Time: 12.71s
Epoch 10/30 | Train Loss: 0.0374 Acc: 0.9907 | Val Loss: 0.0635 Acc: 0.9848 | Time: 12.60s
Epoch 11/30 | Train Loss: 0.0234 Acc: 0.9955 | Val Loss: 0.0580 Acc: 0.9797 | Time: 13.41s
Epoch 12/30 | Train Loss: 0.0186 Acc: 0.9962 | Val Loss: 0.0382 Acc: 0.9911 | Time: 12.84s
Epoch 13/30 | Train Loss: 0.0157 Acc: 0.9970 | Val Loss: 0.0562 Acc: 0.9848 | Time: 12.77s
Epoch 14/30 | Train Loss: 0.0116 Acc: 0.9976 | Val Loss: 0.0370 Acc: 0.9873 | Time: 13.12s
Epoch 15/30 | Train Loss: 0.0187 Acc: 0.9954 | Val Loss: 0.0583 Acc: 0.9797 | Time: 13.15s
Epoch 16/30 | Train Loss: 0.0171 Acc: 0.9951 | Val Loss: 0.0485 Acc: 0.9822 | Time: 13.31s
Epoch 17/30 | Train Loss: 0.0123 Acc: 0.9977 | Val Loss: 0.0533 Acc: 0.9873 | Time: 12.68s
Epoch 18/30 | Train Loss: 0.0100 Acc: 0.9980 | Val Loss: 0.0434 Acc: 0.9873 | Time: 13.13s
Epoch 19/30 | Train Loss: 0.0069 Acc: 0.9986 | Val Loss: 0.0409 Acc: 0.9886 | Time: 13.03s
Early stopping triggered
Training time fold 1: 247.70 seconds
Fold 1 Test Metrics:
```

```
Accuracy: 0.9817
F1-score (macro): 0.9790
AUC (macro): 0.9986
PR AUC (macro): 0.9942
Cohen's Kappa: 0.9769

===== Fold 2 / 5 =====
Epoch 1/30 | Train Loss: 1.2889 Acc: 0.5606 | Val Loss: 0.8677 Acc: 0.7348 | Time: 12.76s
Epoch 2/30 | Train Loss: 0.6610 Acc: 0.8189 | Val Loss: 0.4089 Acc: 0.8947 | Time: 12.98s
Epoch 3/30 | Train Loss: 0.3282 Acc: 0.9193 | Val Loss: 0.2365 Acc: 0.9315 | Time: 13.06s
Epoch 4/30 | Train Loss: 0.1908 Acc: 0.9518 | Val Loss: 0.1950 Acc: 0.9454 | Time: 12.58s
Epoch 5/30 | Train Loss: 0.1173 Acc: 0.9716 | Val Loss: 0.1299 Acc: 0.9683 | Time: 13.18s
Epoch 6/30 | Train Loss: 0.0789 Acc: 0.9813 | Val Loss: 0.1144 Acc: 0.9746 | Time: 13.10s
Epoch 7/30 | Train Loss: 0.0543 Acc: 0.9861 | Val Loss: 0.1086 Acc: 0.9708 | Time: 13.17s
Epoch 8/30 | Train Loss: 0.0404 Acc: 0.9914 | Val Loss: 0.1206 Acc: 0.9670 | Time: 13.01s
Epoch 9/30 | Train Loss: 0.0347 Acc: 0.9932 | Val Loss: 0.1427 Acc: 0.9556 | Time: 13.21s
Epoch 10/30 | Train Loss: 0.0328 Acc: 0.9918 | Val Loss: 0.0862 Acc: 0.9772 | Time: 12.90s
Epoch 11/30 | Train Loss: 0.0318 Acc: 0.9930 | Val Loss: 0.0785 Acc: 0.9708 | Time: 12.75s
Epoch 12/30 | Train Loss: 0.0228 Acc: 0.9949 | Val Loss: 0.0751 Acc: 0.9810 | Time: 13.17s
Epoch 13/30 | Train Loss: 0.0185 Acc: 0.9966 | Val Loss: 0.0661 Acc: 0.9822 | Time: 13.12s
Epoch 14/30 | Train Loss: 0.0134 Acc: 0.9970 | Val Loss: 0.1083 Acc: 0.9683 | Time: 13.06s
Epoch 15/30 | Train Loss: 0.0178 Acc: 0.9962 | Val Loss: 0.1288 Acc: 0.9670 | Time: 13.14s
Epoch 16/30 | Train Loss: 0.0207 Acc: 0.9951 | Val Loss: 0.0906 Acc: 0.9746 | Time: 12.99s
Epoch 17/30 | Train Loss: 0.0142 Acc: 0.9969 | Val Loss: 0.0944 Acc: 0.9721 | Time: 13.35s
Epoch 18/30 | Train Loss: 0.0101 Acc: 0.9986 | Val Loss: 0.0731 Acc: 0.9772 | Time: 13.16s
Early stopping triggered
Training time fold 2: 235.09 seconds
Fold 2 Test Metrics:
Accuracy: 0.9853
F1-score (macro): 0.9833
AUC (macro): 0.9996
PR AUC (macro): 0.9982
Cohen's Kappa: 0.9814

===== Fold 3 / 5 =====
Epoch 1/30 | Train Loss: 1.2528 Acc: 0.6584 | Val Loss: 0.7679 Acc: 0.7970 | Time: 13.13s
Epoch 2/30 | Train Loss: 0.5895 Acc: 0.8392 | Val Loss: 0.3577 Acc: 0.9048 | Time: 13.25s
Epoch 3/30 | Train Loss: 0.3132 Acc: 0.9168 | Val Loss: 0.2278 Acc: 0.9277 | Time: 13.24s
Epoch 4/30 | Train Loss: 0.1777 Acc: 0.9562 | Val Loss: 0.1622 Acc: 0.9492 | Time: 13.27s
Epoch 5/30 | Train Loss: 0.0993 Acc: 0.9776 | Val Loss: 0.1237 Acc: 0.9645 | Time: 13.07s
Epoch 6/30 | Train Loss: 0.0595 Acc: 0.9870 | Val Loss: 0.0964 Acc: 0.9759 | Time: 13.24s
Epoch 7/30 | Train Loss: 0.0473 Acc: 0.9904 | Val Loss: 0.0766 Acc: 0.9734 | Time: 13.19s
Epoch 8/30 | Train Loss: 0.0358 Acc: 0.9920 | Val Loss: 0.0862 Acc: 0.9695 | Time: 13.14s
Epoch 9/30 | Train Loss: 0.0460 Acc: 0.9893 | Val Loss: 0.0744 Acc: 0.9759 | Time: 12.83s
Epoch 10/30 | Train Loss: 0.0200 Acc: 0.9970 | Val Loss: 0.0627 Acc: 0.9772 | Time: 13.15s
Epoch 11/30 | Train Loss: 0.0237 Acc: 0.9939 | Val Loss: 0.0591 Acc: 0.9784 | Time: 12.93s
Epoch 12/30 | Train Loss: 0.0196 Acc: 0.9956 | Val Loss: 0.0518 Acc: 0.9848 | Time: 13.03s
Epoch 13/30 | Train Loss: 0.0199 Acc: 0.9956 | Val Loss: 0.0508 Acc: 0.9810 | Time: 13.01s
Epoch 14/30 | Train Loss: 0.0150 Acc: 0.9969 | Val Loss: 0.0411 Acc: 0.9860 | Time: 12.66s
Epoch 15/30 | Train Loss: 0.0086 Acc: 0.9986 | Val Loss: 0.0458 Acc: 0.9822 | Time: 12.97s
Epoch 16/30 | Train Loss: 0.0167 Acc: 0.9962 | Val Loss: 0.0615 Acc: 0.9810 | Time: 13.07s
Epoch 17/30 | Train Loss: 0.0073 Acc: 0.9993 | Val Loss: 0.0454 Acc: 0.9835 | Time: 13.40s
Epoch 18/30 | Train Loss: 0.0112 Acc: 0.9972 | Val Loss: 0.0590 Acc: 0.9784 | Time: 13.26s
Epoch 19/30 | Train Loss: 0.0068 Acc: 0.9985 | Val Loss: 0.0499 Acc: 0.9822 | Time: 13.17s
Early stopping triggered
Training time fold 3: 249.49 seconds
Fold 3 Test Metrics:
Accuracy: 0.9782
F1-score (macro): 0.9758
AUC (macro): 0.9990
PR AUC (macro): 0.9963
Cohen's Kappa: 0.9723

===== Fold 4 / 5 =====
Epoch 1/30 | Train Loss: 1.2789 Acc: 0.5553 | Val Loss: 0.8836 Acc: 0.7335 | Time: 13.23s
Epoch 2/30 | Train Loss: 0.6747 Acc: 0.7978 | Val Loss: 0.4355 Acc: 0.8896 | Time: 13.25s
Epoch 3/30 | Train Loss: 0.3473 Acc: 0.9094 | Val Loss: 0.2545 Acc: 0.9302 | Time: 13.06s
Epoch 4/30 | Train Loss: 0.1914 Acc: 0.9514 | Val Loss: 0.1747 Acc: 0.9505 | Time: 13.05s
Epoch 5/30 | Train Loss: 0.1198 Acc: 0.9723 | Val Loss: 0.1218 Acc: 0.9657 | Time: 12.80s
Epoch 6/30 | Train Loss: 0.0748 Acc: 0.9827 | Val Loss: 0.0928 Acc: 0.9670 | Time: 13.25s
Epoch 7/30 | Train Loss: 0.0561 Acc: 0.9877 | Val Loss: 0.1085 Acc: 0.9657 | Time: 12.97s
Epoch 8/30 | Train Loss: 0.0377 Acc: 0.9920 | Val Loss: 0.0702 Acc: 0.9784 | Time: 12.73s
Epoch 9/30 | Train Loss: 0.0355 Acc: 0.9921 | Val Loss: 0.0686 Acc: 0.9810 | Time: 13.63s
Epoch 10/30 | Train Loss: 0.0206 Acc: 0.9961 | Val Loss: 0.0552 Acc: 0.9848 | Time: 13.16s
Epoch 11/30 | Train Loss: 0.0228 Acc: 0.9948 | Val Loss: 0.0618 Acc: 0.9860 | Time: 13.24s
Epoch 12/30 | Train Loss: 0.0212 Acc: 0.9948 | Val Loss: 0.0556 Acc: 0.9759 | Time: 13.13s
Epoch 13/30 | Train Loss: 0.0135 Acc: 0.9975 | Val Loss: 0.0665 Acc: 0.9822 | Time: 13.25s
Epoch 14/30 | Train Loss: 0.0182 Acc: 0.9952 | Val Loss: 0.0449 Acc: 0.9898 | Time: 12.77s
Epoch 15/30 | Train Loss: 0.0171 Acc: 0.9955 | Val Loss: 0.0518 Acc: 0.9797 | Time: 12.91s
Epoch 16/30 | Train Loss: 0.0147 Acc: 0.9966 | Val Loss: 0.0547 Acc: 0.9784 | Time: 12.86s
Epoch 17/30 | Train Loss: 0.0104 Acc: 0.9983 | Val Loss: 0.0334 Acc: 0.9886 | Time: 13.26s
Epoch 18/30 | Train Loss: 0.0142 Acc: 0.9966 | Val Loss: 0.0379 Acc: 0.9886 | Time: 13.03s
Epoch 19/30 | Train Loss: 0.0102 Acc: 0.9985 | Val Loss: 0.0294 Acc: 0.9924 | Time: 13.32s
```

```
Epoch 20/30 | Train Loss: 0.0082 Acc: 0.9983 | Val Loss: 0.0524 Acc: 0.9822 | Time: 13.14s
Epoch 21/30 | Train Loss: 0.0071 Acc: 0.9987 | Val Loss: 0.0487 Acc: 0.9810 | Time: 12.83s
Epoch 22/30 | Train Loss: 0.0122 Acc: 0.9963 | Val Loss: 0.0426 Acc: 0.9886 | Time: 13.29s
Epoch 23/30 | Train Loss: 0.0054 Acc: 0.9992 | Val Loss: 0.0297 Acc: 0.9898 | Time: 13.02s
Epoch 24/30 | Train Loss: 0.0052 Acc: 0.9990 | Val Loss: 0.0353 Acc: 0.9886 | Time: 13.19s
Early stopping triggered
Training time fold 4: 314.84 seconds
Fold 4 Test Metrics:
Accuracy: 0.9883
F1-score (macro): 0.9868
AUC (macro): 0.9997
PR AUC (macro): 0.9989
Cohen's Kappa: 0.9853

===== Fold 5 / 5 =====
Epoch 1/30 | Train Loss: 1.2423 Acc: 0.5944 | Val Loss: 0.7754 Acc: 0.7843 | Time: 13.15s
Epoch 2/30 | Train Loss: 0.5892 Acc: 0.8448 | Val Loss: 0.3679 Acc: 0.9112 | Time: 12.69s
Epoch 3/30 | Train Loss: 0.3043 Acc: 0.9255 | Val Loss: 0.2303 Acc: 0.9378 | Time: 13.03s
Epoch 4/30 | Train Loss: 0.1720 Acc: 0.9583 | Val Loss: 0.1591 Acc: 0.9556 | Time: 13.29s
Epoch 5/30 | Train Loss: 0.1013 Acc: 0.9775 | Val Loss: 0.1229 Acc: 0.9657 | Time: 13.05s
Epoch 6/30 | Train Loss: 0.0817 Acc: 0.9823 | Val Loss: 0.1290 Acc: 0.9683 | Time: 13.02s
Epoch 7/30 | Train Loss: 0.0592 Acc: 0.9869 | Val Loss: 0.1011 Acc: 0.9670 | Time: 13.04s
Epoch 8/30 | Train Loss: 0.0399 Acc: 0.9932 | Val Loss: 0.1154 Acc: 0.9670 | Time: 13.06s
Epoch 9/30 | Train Loss: 0.0292 Acc: 0.9938 | Val Loss: 0.0950 Acc: 0.9759 | Time: 12.95s
Epoch 10/30 | Train Loss: 0.0260 Acc: 0.9948 | Val Loss: 0.0774 Acc: 0.9784 | Time: 13.23s
Epoch 11/30 | Train Loss: 0.0204 Acc: 0.9963 | Val Loss: 0.0763 Acc: 0.9772 | Time: 13.17s
Epoch 12/30 | Train Loss: 0.0150 Acc: 0.9976 | Val Loss: 0.0937 Acc: 0.9721 | Time: 13.22s
Epoch 13/30 | Train Loss: 0.0151 Acc: 0.9968 | Val Loss: 0.0987 Acc: 0.9670 | Time: 12.90s
Epoch 14/30 | Train Loss: 0.0170 Acc: 0.9968 | Val Loss: 0.0880 Acc: 0.9708 | Time: 12.88s
Epoch 15/30 | Train Loss: 0.0150 Acc: 0.9962 | Val Loss: 0.0976 Acc: 0.9721 | Time: 13.14s
Epoch 16/30 | Train Loss: 0.0106 Acc: 0.9975 | Val Loss: 0.0847 Acc: 0.9759 | Time: 12.96s
Early stopping triggered
Training time fold 5: 209.09 seconds
Fold 5 Test Metrics:
Accuracy: 0.9726
F1-score (macro): 0.9704
AUC (macro): 0.9991
PR AUC (macro): 0.9966
Cohen's Kappa: 0.9654

Average Test Accuracy over 5 folds: 0.9812 ± 0.0055
```

In [13]:

```python
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import transforms
from torchvision.models import squeezenet1_1, SqueezeNet1_1_Weights
from torch.utils.data import Dataset, DataLoader
from PIL import Image
from sklearn.metrics import (
    accuracy_score, f1_score, cohen_kappa_score, roc_auc_score,
    precision_recall_curve, auc, classification_report, confusion_matrix, roc_curve
)
import time
import numpy as np
import os
import random
import matplotlib.pyplot as plt

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

classes = ["Eyelid", "Normal", "Cataract", "Uveitis", "Conjunctivitis"]
num_classes = len(classes)
class_to_idx = {cls: i for i, cls in enumerate(classes)}

augmented_dir = "/kaggle/working/augmented_all"  # Adjust path to your dataset root

data_transforms = {
    'train': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],
                             [0.229, 0.224, 0.225])
    ]),
    'val': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],
                             [0.229, 0.224, 0.225])
    ]),
    'test': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
```

```
        transforms.Normalize([0.485, 0.456, 0.406],
                             [0.229, 0.224, 0.225])
    ]),
}

class EyeDataset(Dataset):
    def __init__(self, image_paths, labels, transform=None):
        self.image_paths = image_paths
        self.labels = labels
        self.transform = transform
    def __len__(self):
        return len(self.image_paths)
    def __getitem__(self, idx):
        img = Image.open(self.image_paths[idx]).convert('RGB')
        label = self.labels[idx]
        if self.transform:
            img = self.transform(img)
        return img, label

all_data = []
for cls in classes:
    class_dir = os.path.join(augmented_dir, cls)
    image_files = [os.path.join(class_dir, f) for f in os.listdir(class_dir)
                   if f.lower().endswith(('.jpg', '.jpeg', '.png'))]
    image_files = sorted(image_files)
    all_data.extend([(img_path, class_to_idx[cls]) for img_path in image_files])

random.shuffle(all_data)

k = 5
fold_size = len(all_data) // k
folds = []
for i in range(k):
    start_idx = i * fold_size
    if i == k - 1:
        fold = all_data[start_idx:]
    else:
        fold = all_data[start_idx:start_idx + fold_size]
    folds.append(fold)

def get_train_val_test(folds, test_fold_index, val_ratio=0.1):
    test_data = folds[test_fold_index]
    train_val_data = [item for i, f in enumerate(folds) if i != test_fold_index for item in f]
    n_val = int(len(train_val_data) * val_ratio)
    val_data = train_val_data[:n_val]
    train_data = train_val_data[n_val:]
    return train_data, val_data, test_data

def create_dataset_loader_from_list(data_list, transform, shuffle=False):
    image_paths = [item[0] for item in data_list]
    labels = [item[1] for item in data_list]
    dataset = EyeDataset(image_paths, labels, transform=transform)
    loader = DataLoader(dataset, batch_size=32, shuffle=shuffle, num_workers=2)
    return dataset, loader

criterion = nn.CrossEntropyLoss()
num_epochs = 30
patience = 5

def train_epoch(model, dataloader, optimizer):
    model.train()
    running_loss = 0.0
    correct = 0
    total = 0
    for inputs, labels in dataloader:
        inputs, labels = inputs.to(device), labels.to(device)
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item() * inputs.size(0)
        _, preds = torch.max(outputs, 1)
        correct += (preds == labels).sum().item()
        total += labels.size(0)
    return running_loss / total, correct / total

def eval_model(model, dataloader):
    model.eval()
    running_loss = 0.0
    correct = 0
    total = 0
    all_labels = []
```

```
    all_preds = []
    all_probs = []
    with torch.no_grad():
        for inputs, labels in dataloader:
            inputs, labels = inputs.to(device), labels.to(device)
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            probs = torch.softmax(outputs, dim=1)
            _, preds = torch.max(outputs, 1)
            running_loss += loss.item() * inputs.size(0)
            correct += (preds == labels).sum().item()
            total += labels.size(0)
            all_labels.extend(labels.cpu().numpy())
            all_preds.extend(preds.cpu().numpy())
            all_probs.extend(probs.cpu().numpy())
    return running_loss / total, correct / total, np.array(all_labels), np.array(all_preds), np.array(all_probs)

fold_test_accuracies = []

for fold_idx in range(k):
    print(f"\n===== Fold {fold_idx + 1} / {k} =====")
    train_data, val_data, test_data = get_train_val_test(folds, fold_idx)
    train_dataset, train_loader = create_dataset_loader_from_list(train_data, data_transforms['train'], shuffle=
    val_dataset, val_loader = create_dataset_loader_from_list(val_data, data_transforms['val'], shuffle=False)
    test_dataset, test_loader = create_dataset_loader_from_list(test_data, data_transforms['test'], shuffle=Fals

    model = squeezenet1_1(weights=SqueezeNet1_1_Weights.IMAGENET1K_V1)
    model.classifier[1] = nn.Conv2d(512, num_classes, kernel_size=(1,1))
    model.num_classes = num_classes
    model = model.to(device)

    optimizer = optim.Adam(model.parameters(), lr=1e-4, weight_decay=1e-4)
    scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, mode='min', patience=3)

    best_val_loss = float('inf')
    epochs_no_improve = 0
    train_start_time = time.time()
    for epoch in range(num_epochs):
        start_epoch_time = time.time()
        train_loss, train_acc = train_epoch(model, train_loader, optimizer)
        val_loss, val_acc, _, _, _ = eval_model(model, val_loader)
        scheduler.step(val_loss)
        epoch_time = time.time() - start_epoch_time
        print(f"Epoch {epoch+1}/{num_epochs} | Train Loss: {train_loss:.4f} Acc: {train_acc:.4f} | "
              f"Val Loss: {val_loss:.4f} Acc: {val_acc:.4f} | Time: {epoch_time:.2f}s")
        if val_loss < best_val_loss:
            best_val_loss = val_loss
            epochs_no_improve = 0
            torch.save(model.state_dict(), f"best_squeezenet_fold{fold_idx}.pth")
        else:
            epochs_no_improve += 1
            if epochs_no_improve >= patience:
                print("Early stopping triggered")
                break
    train_end_time = time.time()
    print(f"Training time fold {fold_idx+1}: {train_end_time - train_start_time:.2f} seconds")

    model.load_state_dict(torch.load(f"best_squeezenet_fold{fold_idx}.pth"))

    test_loss, test_acc, test_labels, test_preds, test_probs = eval_model(model, test_loader)

    acc = accuracy_score(test_labels, test_preds)
    f1 = f1_score(test_labels, test_preds, average='macro')
    kappa = cohen_kappa_score(test_labels, test_preds)

    try:
        auc_macro = roc_auc_score(test_labels, test_probs, multi_class='ovo', average='macro')
    except Exception:
        auc_macro = float('nan')

    pr_auc_per_class = []
    for i in range(num_classes):
        precision, recall, _ = precision_recall_curve(test_labels == i, test_probs[:, i])
        pr_auc_per_class.append(auc(recall, precision))
    pr_auc_macro = np.mean(pr_auc_per_class)

    print(f"Fold {fold_idx + 1} Test Metrics:")
    print(f"Accuracy: {acc:.4f}")
    print(f"F1-score (macro): {f1:.4f}")
    print(f"AUC (macro): {auc_macro:.4f}")
    print(f"PR AUC (macro): {pr_auc_macro:.4f}")
    print(f"Cohen's Kappa: {kappa:.4f}")
```

```
    fold_test_accuracies.append(acc)

print(f"\nAverage Test Accuracy over {k} folds: {np.mean(fold_test_accuracies):.4f} ± {np.std(fold_test_accuraci
```

```
===== Fold 1 / 5 =====
Downloading: "https://download.pytorch.org/models/squeezenet1_1-b8a52dc0.pth" to /root/.cache/torch/hub/checkpoi
nts/squeezenet1_1-b8a52dc0.pth
100%|██████████| 4.73M/4.73M [00:00<00:00, 65.9MB/s]
Epoch 1/30 | Train Loss: 0.7720 Acc: 0.7068 | Val Loss: 0.4896 Acc: 0.8464 | Time: 11.70s
Epoch 2/30 | Train Loss: 0.4430 Acc: 0.8437 | Val Loss: 0.3305 Acc: 0.8731 | Time: 11.42s
Epoch 3/30 | Train Loss: 0.3321 Acc: 0.8809 | Val Loss: 0.3147 Acc: 0.8845 | Time: 11.36s
Epoch 4/30 | Train Loss: 0.2385 Acc: 0.9121 | Val Loss: 0.2677 Acc: 0.9061 | Time: 11.35s
Epoch 5/30 | Train Loss: 0.2073 Acc: 0.9259 | Val Loss: 0.2166 Acc: 0.9315 | Time: 11.15s
Epoch 6/30 | Train Loss: 0.1576 Acc: 0.9441 | Val Loss: 0.2440 Acc: 0.9074 | Time: 11.12s
Epoch 7/30 | Train Loss: 0.1209 Acc: 0.9590 | Val Loss: 0.1540 Acc: 0.9467 | Time: 11.20s
Epoch 8/30 | Train Loss: 0.0952 Acc: 0.9675 | Val Loss: 0.1109 Acc: 0.9581 | Time: 11.25s
Epoch 9/30 | Train Loss: 0.0921 Acc: 0.9689 | Val Loss: 0.1156 Acc: 0.9543 | Time: 11.08s
Epoch 10/30 | Train Loss: 0.0700 Acc: 0.9759 | Val Loss: 0.2349 Acc: 0.9213 | Time: 11.24s
Epoch 11/30 | Train Loss: 0.0539 Acc: 0.9823 | Val Loss: 0.0977 Acc: 0.9607 | Time: 11.22s
Epoch 12/30 | Train Loss: 0.0738 Acc: 0.9724 | Val Loss: 0.0951 Acc: 0.9607 | Time: 11.20s
Epoch 13/30 | Train Loss: 0.0367 Acc: 0.9880 | Val Loss: 0.1964 Acc: 0.9378 | Time: 11.24s
Epoch 14/30 | Train Loss: 0.0438 Acc: 0.9844 | Val Loss: 0.1628 Acc: 0.9391 | Time: 11.44s
Epoch 15/30 | Train Loss: 0.0405 Acc: 0.9872 | Val Loss: 0.1181 Acc: 0.9569 | Time: 11.42s
Epoch 16/30 | Train Loss: 0.0470 Acc: 0.9851 | Val Loss: 0.0693 Acc: 0.9746 | Time: 11.32s
Epoch 17/30 | Train Loss: 0.0440 Acc: 0.9861 | Val Loss: 0.0474 Acc: 0.9822 | Time: 11.19s
Epoch 18/30 | Train Loss: 0.0195 Acc: 0.9934 | Val Loss: 0.0645 Acc: 0.9822 | Time: 11.46s
Epoch 19/30 | Train Loss: 0.0337 Acc: 0.9887 | Val Loss: 0.0720 Acc: 0.9759 | Time: 11.26s
Epoch 20/30 | Train Loss: 0.0296 Acc: 0.9911 | Val Loss: 0.2471 Acc: 0.9302 | Time: 11.37s
Epoch 21/30 | Train Loss: 0.0450 Acc: 0.9854 | Val Loss: 0.1017 Acc: 0.9695 | Time: 11.41s
Epoch 22/30 | Train Loss: 0.0068 Acc: 0.9983 | Val Loss: 0.0471 Acc: 0.9873 | Time: 11.27s
Epoch 23/30 | Train Loss: 0.0031 Acc: 0.9999 | Val Loss: 0.0418 Acc: 0.9898 | Time: 11.21s
Epoch 24/30 | Train Loss: 0.0028 Acc: 0.9996 | Val Loss: 0.0380 Acc: 0.9860 | Time: 11.27s
Epoch 25/30 | Train Loss: 0.0022 Acc: 0.9996 | Val Loss: 0.0396 Acc: 0.9898 | Time: 11.38s
Epoch 26/30 | Train Loss: 0.0024 Acc: 0.9996 | Val Loss: 0.0340 Acc: 0.9898 | Time: 11.38s
Epoch 27/30 | Train Loss: 0.0015 Acc: 0.9999 | Val Loss: 0.0349 Acc: 0.9886 | Time: 11.34s
Epoch 28/30 | Train Loss: 0.0014 Acc: 0.9999 | Val Loss: 0.0403 Acc: 0.9898 | Time: 11.17s
Epoch 29/30 | Train Loss: 0.0014 Acc: 1.0000 | Val Loss: 0.0354 Acc: 0.9911 | Time: 11.33s
Epoch 30/30 | Train Loss: 0.0011 Acc: 1.0000 | Val Loss: 0.0369 Acc: 0.9898 | Time: 11.25s
Training time fold 1: 339.18 seconds
Fold 1 Test Metrics:
Accuracy: 0.9878
F1-score (macro): 0.9871
AUC (macro): 0.9997
PR AUC (macro): 0.9988
Cohen's Kappa: 0.9846

===== Fold 2 / 5 =====
Epoch 1/30 | Train Loss: 0.7813 Acc: 0.7023 | Val Loss: 0.5754 Acc: 0.7893 | Time: 11.39s
Epoch 2/30 | Train Loss: 0.4611 Acc: 0.8366 | Val Loss: 0.4048 Acc: 0.8426 | Time: 11.34s
Epoch 3/30 | Train Loss: 0.3318 Acc: 0.8785 | Val Loss: 0.3062 Acc: 0.8985 | Time: 11.42s
Epoch 4/30 | Train Loss: 0.2512 Acc: 0.9104 | Val Loss: 0.2548 Acc: 0.9137 | Time: 11.25s
Epoch 5/30 | Train Loss: 0.2128 Acc: 0.9225 | Val Loss: 0.2144 Acc: 0.9213 | Time: 11.24s
Epoch 6/30 | Train Loss: 0.1621 Acc: 0.9424 | Val Loss: 0.1898 Acc: 0.9277 | Time: 11.34s
Epoch 7/30 | Train Loss: 0.1197 Acc: 0.9596 | Val Loss: 0.2120 Acc: 0.9226 | Time: 11.36s
Epoch 8/30 | Train Loss: 0.1090 Acc: 0.9631 | Val Loss: 0.1187 Acc: 0.9505 | Time: 11.53s
Epoch 9/30 | Train Loss: 0.0713 Acc: 0.9768 | Val Loss: 0.1401 Acc: 0.9416 | Time: 11.21s
Epoch 10/30 | Train Loss: 0.1288 Acc: 0.9547 | Val Loss: 0.1724 Acc: 0.9327 | Time: 11.27s
Epoch 11/30 | Train Loss: 0.0580 Acc: 0.9807 | Val Loss: 0.1121 Acc: 0.9645 | Time: 11.52s
Epoch 12/30 | Train Loss: 0.0417 Acc: 0.9866 | Val Loss: 0.1882 Acc: 0.9340 | Time: 11.33s
Epoch 13/30 | Train Loss: 0.0393 Acc: 0.9877 | Val Loss: 0.1556 Acc: 0.9480 | Time: 11.28s
Epoch 14/30 | Train Loss: 0.0643 Acc: 0.9770 | Val Loss: 0.1082 Acc: 0.9645 | Time: 11.31s
Epoch 15/30 | Train Loss: 0.0317 Acc: 0.9901 | Val Loss: 0.1044 Acc: 0.9569 | Time: 11.41s
Epoch 16/30 | Train Loss: 0.0403 Acc: 0.9880 | Val Loss: 0.2922 Acc: 0.9099 | Time: 11.29s
Epoch 17/30 | Train Loss: 0.0271 Acc: 0.9908 | Val Loss: 0.0828 Acc: 0.9721 | Time: 11.29s
Epoch 18/30 | Train Loss: 0.0318 Acc: 0.9893 | Val Loss: 0.0934 Acc: 0.9683 | Time: 11.33s
Epoch 19/30 | Train Loss: 0.0361 Acc: 0.9870 | Val Loss: 0.0896 Acc: 0.9695 | Time: 11.10s
Epoch 20/30 | Train Loss: 0.0172 Acc: 0.9952 | Val Loss: 0.0652 Acc: 0.9772 | Time: 11.25s
Epoch 21/30 | Train Loss: 0.0471 Acc: 0.9842 | Val Loss: 0.1003 Acc: 0.9607 | Time: 11.27s
Epoch 22/30 | Train Loss: 0.0272 Acc: 0.9920 | Val Loss: 0.2681 Acc: 0.9175 | Time: 11.45s
Epoch 23/30 | Train Loss: 0.0271 Acc: 0.9914 | Val Loss: 0.1135 Acc: 0.9632 | Time: 11.42s
Epoch 24/30 | Train Loss: 0.0235 Acc: 0.9924 | Val Loss: 0.0843 Acc: 0.9708 | Time: 11.35s
Epoch 25/30 | Train Loss: 0.0035 Acc: 0.9993 | Val Loss: 0.0528 Acc: 0.9848 | Time: 11.29s
Epoch 26/30 | Train Loss: 0.0018 Acc: 1.0000 | Val Loss: 0.0507 Acc: 0.9848 | Time: 11.60s
Epoch 27/30 | Train Loss: 0.0015 Acc: 0.9997 | Val Loss: 0.0486 Acc: 0.9860 | Time: 11.18s
Epoch 28/30 | Train Loss: 0.0013 Acc: 1.0000 | Val Loss: 0.0483 Acc: 0.9822 | Time: 11.34s
Epoch 29/30 | Train Loss: 0.0012 Acc: 0.9999 | Val Loss: 0.0458 Acc: 0.9911 | Time: 11.27s
Epoch 30/30 | Train Loss: 0.0009 Acc: 1.0000 | Val Loss: 0.0440 Acc: 0.9911 | Time: 11.49s
Training time fold 2: 340.32 seconds
Fold 2 Test Metrics:
Accuracy: 0.9817
F1-score (macro): 0.9810
AUC (macro): 0.9995
PR AUC (macro): 0.9980
```

```
Cohen's Kappa: 0.9770

===== Fold 3 / 5 =====
Epoch 1/30 | Train Loss: 0.7949 Acc: 0.6929 | Val Loss: 0.5654 Acc: 0.7855 | Time: 11.25s
Epoch 2/30 | Train Loss: 0.4814 Acc: 0.8234 | Val Loss: 0.4020 Acc: 0.8426 | Time: 11.21s
Epoch 3/30 | Train Loss: 0.3590 Acc: 0.8695 | Val Loss: 0.3356 Acc: 0.8782 | Time: 11.57s
Epoch 4/30 | Train Loss: 0.2724 Acc: 0.9027 | Val Loss: 0.3149 Acc: 0.8921 | Time: 11.48s
Epoch 5/30 | Train Loss: 0.2167 Acc: 0.9238 | Val Loss: 0.2555 Acc: 0.9010 | Time: 11.15s
Epoch 6/30 | Train Loss: 0.1765 Acc: 0.9362 | Val Loss: 0.1637 Acc: 0.9442 | Time: 11.35s
Epoch 7/30 | Train Loss: 0.1319 Acc: 0.9561 | Val Loss: 0.1360 Acc: 0.9518 | Time: 11.54s
Epoch 8/30 | Train Loss: 0.1082 Acc: 0.9620 | Val Loss: 0.2583 Acc: 0.9023 | Time: 11.49s
Epoch 9/30 | Train Loss: 0.0901 Acc: 0.9701 | Val Loss: 0.1851 Acc: 0.9289 | Time: 11.50s
Epoch 10/30 | Train Loss: 0.0741 Acc: 0.9759 | Val Loss: 0.2683 Acc: 0.9074 | Time: 11.19s
Epoch 11/30 | Train Loss: 0.0760 Acc: 0.9759 | Val Loss: 0.1768 Acc: 0.9404 | Time: 11.37s
Epoch 12/30 | Train Loss: 0.0257 Acc: 0.9938 | Val Loss: 0.0710 Acc: 0.9797 | Time: 11.19s
Epoch 13/30 | Train Loss: 0.0163 Acc: 0.9968 | Val Loss: 0.0667 Acc: 0.9822 | Time: 11.29s
Epoch 14/30 | Train Loss: 0.0127 Acc: 0.9969 | Val Loss: 0.0637 Acc: 0.9810 | Time: 11.52s
Epoch 15/30 | Train Loss: 0.0116 Acc: 0.9977 | Val Loss: 0.0623 Acc: 0.9810 | Time: 11.13s
Epoch 16/30 | Train Loss: 0.0107 Acc: 0.9977 | Val Loss: 0.0641 Acc: 0.9797 | Time: 11.25s
Epoch 17/30 | Train Loss: 0.0095 Acc: 0.9979 | Val Loss: 0.0572 Acc: 0.9848 | Time: 11.41s
Epoch 18/30 | Train Loss: 0.0077 Acc: 0.9989 | Val Loss: 0.0518 Acc: 0.9873 | Time: 11.43s
Epoch 19/30 | Train Loss: 0.0071 Acc: 0.9993 | Val Loss: 0.0563 Acc: 0.9848 | Time: 11.17s
Epoch 20/30 | Train Loss: 0.0065 Acc: 0.9989 | Val Loss: 0.0507 Acc: 0.9886 | Time: 11.12s
Epoch 21/30 | Train Loss: 0.0069 Acc: 0.9985 | Val Loss: 0.0506 Acc: 0.9835 | Time: 11.33s
Epoch 22/30 | Train Loss: 0.0058 Acc: 0.9990 | Val Loss: 0.0478 Acc: 0.9848 | Time: 11.38s
Epoch 23/30 | Train Loss: 0.0040 Acc: 0.9997 | Val Loss: 0.0516 Acc: 0.9835 | Time: 11.32s
Epoch 24/30 | Train Loss: 0.0047 Acc: 0.9993 | Val Loss: 0.0510 Acc: 0.9797 | Time: 11.42s
Epoch 25/30 | Train Loss: 0.0047 Acc: 0.9993 | Val Loss: 0.0611 Acc: 0.9797 | Time: 11.31s
Epoch 26/30 | Train Loss: 0.0053 Acc: 0.9990 | Val Loss: 0.0569 Acc: 0.9797 | Time: 11.26s
Epoch 27/30 | Train Loss: 0.0038 Acc: 0.9992 | Val Loss: 0.0426 Acc: 0.9898 | Time: 11.25s
Epoch 28/30 | Train Loss: 0.0027 Acc: 1.0000 | Val Loss: 0.0444 Acc: 0.9886 | Time: 11.35s
Epoch 29/30 | Train Loss: 0.0024 Acc: 1.0000 | Val Loss: 0.0444 Acc: 0.9886 | Time: 11.24s
Epoch 30/30 | Train Loss: 0.0027 Acc: 0.9999 | Val Loss: 0.0455 Acc: 0.9886 | Time: 11.31s
Training time fold 3: 340.00 seconds
Fold 3 Test Metrics:
Accuracy: 0.9878
F1-score (macro): 0.9864
AUC (macro): 0.9996
PR AUC (macro): 0.9983
Cohen's Kappa: 0.9846

===== Fold 4 / 5 =====
Epoch 1/30 | Train Loss: 0.7283 Acc: 0.7223 | Val Loss: 0.5902 Acc: 0.7766 | Time: 11.30s
Epoch 2/30 | Train Loss: 0.4292 Acc: 0.8466 | Val Loss: 0.3871 Acc: 0.8363 | Time: 11.18s
Epoch 3/30 | Train Loss: 0.2940 Acc: 0.8911 | Val Loss: 0.2973 Acc: 0.8820 | Time: 11.41s
Epoch 4/30 | Train Loss: 0.2357 Acc: 0.9127 | Val Loss: 0.2718 Acc: 0.9048 | Time: 11.26s
Epoch 5/30 | Train Loss: 0.1701 Acc: 0.9417 | Val Loss: 0.2142 Acc: 0.9162 | Time: 11.28s
Epoch 6/30 | Train Loss: 0.1375 Acc: 0.9507 | Val Loss: 0.2134 Acc: 0.9213 | Time: 11.51s
Epoch 7/30 | Train Loss: 0.1133 Acc: 0.9596 | Val Loss: 0.1565 Acc: 0.9429 | Time: 11.34s
Epoch 8/30 | Train Loss: 0.0841 Acc: 0.9706 | Val Loss: 0.1712 Acc: 0.9429 | Time: 11.48s
Epoch 9/30 | Train Loss: 0.0978 Acc: 0.9679 | Val Loss: 0.1009 Acc: 0.9670 | Time: 11.42s
Epoch 10/30 | Train Loss: 0.0647 Acc: 0.9772 | Val Loss: 0.1028 Acc: 0.9607 | Time: 11.38s
Epoch 11/30 | Train Loss: 0.0551 Acc: 0.9830 | Val Loss: 0.1025 Acc: 0.9645 | Time: 11.32s
Epoch 12/30 | Train Loss: 0.0619 Acc: 0.9793 | Val Loss: 0.0992 Acc: 0.9645 | Time: 11.35s
Epoch 13/30 | Train Loss: 0.0543 Acc: 0.9811 | Val Loss: 0.0953 Acc: 0.9708 | Time: 11.21s
Epoch 14/30 | Train Loss: 0.0749 Acc: 0.9799 | Val Loss: 0.0769 Acc: 0.9708 | Time: 11.23s
Epoch 15/30 | Train Loss: 0.0479 Acc: 0.9841 | Val Loss: 0.1570 Acc: 0.9467 | Time: 11.30s
Epoch 16/30 | Train Loss: 0.0458 Acc: 0.9845 | Val Loss: 0.0558 Acc: 0.9734 | Time: 11.19s
Epoch 17/30 | Train Loss: 0.0173 Acc: 0.9951 | Val Loss: 0.0501 Acc: 0.9835 | Time: 11.21s
Epoch 18/30 | Train Loss: 0.0419 Acc: 0.9862 | Val Loss: 0.0880 Acc: 0.9721 | Time: 11.28s
Epoch 19/30 | Train Loss: 0.0399 Acc: 0.9865 | Val Loss: 0.0613 Acc: 0.9797 | Time: 11.27s
Epoch 20/30 | Train Loss: 0.0308 Acc: 0.9907 | Val Loss: 0.0728 Acc: 0.9784 | Time: 11.55s
Epoch 21/30 | Train Loss: 0.0109 Acc: 0.9969 | Val Loss: 0.0367 Acc: 0.9873 | Time: 11.26s
Epoch 22/30 | Train Loss: 0.0185 Acc: 0.9951 | Val Loss: 0.0463 Acc: 0.9860 | Time: 11.27s
Epoch 23/30 | Train Loss: 0.0138 Acc: 0.9962 | Val Loss: 0.1071 Acc: 0.9683 | Time: 11.45s
Epoch 24/30 | Train Loss: 0.0540 Acc: 0.9825 | Val Loss: 0.1164 Acc: 0.9683 | Time: 11.29s
Epoch 25/30 | Train Loss: 0.0372 Acc: 0.9883 | Val Loss: 0.0382 Acc: 0.9848 | Time: 11.44s
Epoch 26/30 | Train Loss: 0.0032 Acc: 0.9999 | Val Loss: 0.0251 Acc: 0.9898 | Time: 11.41s
Epoch 27/30 | Train Loss: 0.0022 Acc: 0.9999 | Val Loss: 0.0254 Acc: 0.9911 | Time: 11.23s
Epoch 28/30 | Train Loss: 0.0017 Acc: 1.0000 | Val Loss: 0.0238 Acc: 0.9898 | Time: 11.47s
Epoch 29/30 | Train Loss: 0.0014 Acc: 1.0000 | Val Loss: 0.0244 Acc: 0.9898 | Time: 11.42s
Epoch 30/30 | Train Loss: 0.0012 Acc: 0.9999 | Val Loss: 0.0255 Acc: 0.9898 | Time: 11.41s
Training time fold 4: 340.33 seconds
Fold 4 Test Metrics:
Accuracy: 0.9848
F1-score (macro): 0.9830
AUC (macro): 0.9993
PR AUC (macro): 0.9975
Cohen's Kappa: 0.9808

===== Fold 5 / 5 =====
Epoch 1/30 | Train Loss: 0.8341 Acc: 0.6801 | Val Loss: 0.5081 Acc: 0.8058 | Time: 11.17s
Epoch 2/30 | Train Loss: 0.4774 Acc: 0.8220 | Val Loss: 0.4003 Acc: 0.8388 | Time: 11.40s
```

```
Epoch 3/30 | Train Loss: 0.3346 Acc: 0.8790 | Val Loss: 0.3045 Acc: 0.8744 | Time: 11.48s
Epoch 4/30 | Train Loss: 0.2576 Acc: 0.9049 | Val Loss: 0.2569 Acc: 0.9112 | Time: 11.39s
Epoch 5/30 | Train Loss: 0.2055 Acc: 0.9256 | Val Loss: 0.2341 Acc: 0.9112 | Time: 11.34s
Epoch 6/30 | Train Loss: 0.1633 Acc: 0.9435 | Val Loss: 0.1923 Acc: 0.9302 | Time: 11.46s
Epoch 7/30 | Train Loss: 0.1279 Acc: 0.9541 | Val Loss: 0.1565 Acc: 0.9404 | Time: 11.25s
Epoch 8/30 | Train Loss: 0.1045 Acc: 0.9648 | Val Loss: 0.1411 Acc: 0.9454 | Time: 11.01s
Epoch 9/30 | Train Loss: 0.0830 Acc: 0.9718 | Val Loss: 0.2132 Acc: 0.9264 | Time: 11.19s
Epoch 10/30 | Train Loss: 0.0723 Acc: 0.9770 | Val Loss: 0.0910 Acc: 0.9721 | Time: 11.20s
Epoch 11/30 | Train Loss: 0.0616 Acc: 0.9793 | Val Loss: 0.2803 Acc: 0.9099 | Time: 11.44s
Epoch 12/30 | Train Loss: 0.0790 Acc: 0.9724 | Val Loss: 0.0885 Acc: 0.9657 | Time: 11.36s
Epoch 13/30 | Train Loss: 0.0392 Acc: 0.9880 | Val Loss: 0.0843 Acc: 0.9695 | Time: 11.08s
Epoch 14/30 | Train Loss: 0.0415 Acc: 0.9866 | Val Loss: 0.1518 Acc: 0.9429 | Time: 11.62s
Epoch 15/30 | Train Loss: 0.0505 Acc: 0.9835 | Val Loss: 0.0841 Acc: 0.9695 | Time: 11.57s
Epoch 16/30 | Train Loss: 0.0426 Acc: 0.9846 | Val Loss: 0.1058 Acc: 0.9543 | Time: 11.19s
Epoch 17/30 | Train Loss: 0.0385 Acc: 0.9882 | Val Loss: 0.1007 Acc: 0.9632 | Time: 11.26s
Epoch 18/30 | Train Loss: 0.0450 Acc: 0.9854 | Val Loss: 0.0908 Acc: 0.9746 | Time: 11.37s
Epoch 19/30 | Train Loss: 0.0267 Acc: 0.9918 | Val Loss: 0.1099 Acc: 0.9683 | Time: 11.38s
Epoch 20/30 | Train Loss: 0.0068 Acc: 0.9992 | Val Loss: 0.0485 Acc: 0.9848 | Time: 11.28s
Epoch 21/30 | Train Loss: 0.0035 Acc: 0.9996 | Val Loss: 0.0471 Acc: 0.9835 | Time: 11.49s
Epoch 22/30 | Train Loss: 0.0027 Acc: 0.9999 | Val Loss: 0.0458 Acc: 0.9835 | Time: 11.33s
Epoch 23/30 | Train Loss: 0.0031 Acc: 0.9996 | Val Loss: 0.0483 Acc: 0.9848 | Time: 11.46s
Epoch 24/30 | Train Loss: 0.0022 Acc: 0.9999 | Val Loss: 0.0481 Acc: 0.9860 | Time: 11.27s
Epoch 25/30 | Train Loss: 0.0022 Acc: 0.9996 | Val Loss: 0.0391 Acc: 0.9873 | Time: 11.65s
Epoch 26/30 | Train Loss: 0.0016 Acc: 0.9999 | Val Loss: 0.0385 Acc: 0.9848 | Time: 11.47s
Epoch 27/30 | Train Loss: 0.0017 Acc: 0.9997 | Val Loss: 0.0417 Acc: 0.9873 | Time: 11.26s
Epoch 28/30 | Train Loss: 0.0013 Acc: 1.0000 | Val Loss: 0.0416 Acc: 0.9848 | Time: 11.42s
Epoch 29/30 | Train Loss: 0.0017 Acc: 0.9997 | Val Loss: 0.0484 Acc: 0.9860 | Time: 11.34s
Epoch 30/30 | Train Loss: 0.0012 Acc: 1.0000 | Val Loss: 0.0429 Acc: 0.9898 | Time: 11.50s
Training time fold 5: 340.85 seconds
Fold 5 Test Metrics:
Accuracy: 0.9894
F1-score (macro): 0.9883
AUC (macro): 0.9998
PR AUC (macro): 0.9993
Cohen's Kappa: 0.9865

Average Test Accuracy over 5 folds: 0.9863 ± 0.0027
```

In [14]:

```python
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import transforms
from torchvision.models import shufflenet_v2_x1_0, ShuffleNet_V2_X1_0_Weights
from torchvision.models import mobilenet_v2, MobileNet_V2_Weights
from torch.utils.data import Dataset, DataLoader
from PIL import Image
from sklearn.metrics import accuracy_score, f1_score, cohen_kappa_score, roc_auc_score, precision_recall_curve,
import time
import numpy as np
import os
import random

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

classes = ["Eyelid", "Normal", "Cataract", "Uveitis", "Conjunctivitis"]
num_classes = len(classes)
class_to_idx = {cls: i for i, cls in enumerate(classes)}

augmented_dir = "/kaggle/working/augmented_all"

data_transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406],
                         [0.229, 0.224, 0.225])
])

class EyeDataset(Dataset):
    def __init__(self, image_paths, labels, transform=None):
        self.image_paths = image_paths
        self.labels = labels
        self.transform = transform
    def __len__(self):
        return len(self.image_paths)
    def __getitem__(self, idx):
        img = Image.open(self.image_paths[idx]).convert('RGB')
        label = self.labels[idx]
        if self.transform:
            img = self.transform(img)
        return img, label

all_data = []
for cls in classes:
```

```
    class_dir = os.path.join(augmented_dir, cls)
    image_files = [os.path.join(class_dir, f) for f in os.listdir(class_dir)
                   if f.lower().endswith(('.jpg', '.jpeg', '.png'))]
    image_files = sorted(image_files)
    all_data.extend([(img_path, class_to_idx[cls]) for img_path in image_files])

random.shuffle(all_data)

k = 5
fold_size = len(all_data) // k
folds = []
for i in range(k):
    start_idx = i * fold_size
    if i == k - 1:
        fold = all_data[start_idx:]
    else:
        fold = all_data[start_idx:start_idx + fold_size]
    folds.append(fold)

def get_train_val_test(folds, test_fold_index, val_ratio=0.1):
    test_data = folds[test_fold_index]
    train_val_data = [item for i, f in enumerate(folds) if i != test_fold_index for item in f]
    n_val = int(len(train_val_data) * val_ratio)
    val_data = train_val_data[:n_val]
    train_data = train_val_data[n_val:]
    return train_data, val_data, test_data

def create_dataset_loader_from_list(data_list, transform, shuffle=False):
    image_paths = [item[0] for item in data_list]
    labels = [item[1] for item in data_list]
    dataset = EyeDataset(image_paths, labels, transform=transform)
    loader = DataLoader(dataset, batch_size=32, shuffle=shuffle, num_workers=2)
    return dataset, loader

# Feature extractor models

class ShuffleNetFeatureExtractor(nn.Module):
    def __init__(self, model):
        super().__init__()
        self.features = nn.Sequential(*list(model.children())[:-1])
        self.pool = nn.AdaptiveAvgPool2d(1)
        self.flatten = nn.Flatten()
    def forward(self, x):
        x = self.features(x)
        x = self.pool(x)
        x = self.flatten(x)
        return x

class MobileNetFeatureExtractor(nn.Module):
    def __init__(self, model):
        super().__init__()
        self.features = model.features
        self.pool = nn.AdaptiveAvgPool2d(1)
        self.flatten = nn.Flatten()
    def forward(self, x):
        x = self.features(x)
        x = self.pool(x)
        x = self.flatten(x)
        return x

# Fusion classifier

class FusionModel(nn.Module):
    def __init__(self, sn_dim, mn_dim, num_classes):
        super().__init__()
        self.bn1 = nn.BatchNorm1d(sn_dim)
        self.bn2 = nn.BatchNorm1d(mn_dim)
        self.classifier = nn.Sequential(
            nn.Linear(sn_dim + mn_dim, 512),
            nn.ReLU(),
            nn.Dropout(0.5),
            nn.Linear(512, num_classes)
        )
    def forward(self, x):
        feat_sn, feat_mn = x
        feat_sn = self.bn1(feat_sn)
        feat_mn = self.bn2(feat_mn)
        fused = torch.cat((feat_sn, feat_mn), dim=1)
        out = self.classifier(fused)
        return out

criterion = nn.CrossEntropyLoss()
```

```
num_epochs = 30
patience = 5

fold_metrics = []

# Initialize base pretrained models once to avoid repeated download
shufflenet_base = shufflenet_v2_x1_0(weights=ShuffleNet_V2_X1_0_Weights.IMAGENET1K_V1).to(device)
mobilenet_base = mobilenet_v2(weights=MobileNet_V2_Weights.IMAGENET1K_V1).to(device)

# Get feature dims from dummy input
with torch.no_grad():
    dummy = torch.randn(1,3,224,224).to(device)
    sn_feat_dim = ShuffleNetFeatureExtractor(shufflenet_base)(dummy).shape[1]
    mn_feat_dim = MobileNetFeatureExtractor(mobilenet_base)(dummy).shape[1]

print(f"ShuffleNet feature dim: {sn_feat_dim}, MobileNetV2 feature dim: {mn_feat_dim}")

for fold_idx in range(k):
    print(f"\n===== Fold {fold_idx + 1} / {k} =====")
    train_data, val_data, test_data = get_train_val_test(folds, fold_idx)
    train_dataset, train_loader = create_dataset_loader_from_list(train_data, data_transform, shuffle=True)
    val_dataset, val_loader = create_dataset_loader_from_list(val_data, data_transform, shuffle=False)
    test_dataset, test_loader = create_dataset_loader_from_list(test_data, data_transform, shuffle=False)

    # Create fresh models for each fold
    shufflenet_feat = ShuffleNetFeatureExtractor(shufflenet_base)
    mobilenet_feat = MobileNetFeatureExtractor(mobilenet_base)
    fusion_model = FusionModel(sn_feat_dim, mn_feat_dim, num_classes)

    shufflenet_feat = shufflenet_feat.to(device)
    mobilenet_feat = mobilenet_feat.to(device)
    fusion_model = fusion_model.to(device)

    params = list(fusion_model.parameters()) + list(shufflenet_feat.parameters()) + list(mobilenet_feat.paramete
    optimizer = optim.Adam(params, lr=1e-4, weight_decay=1e-4)
    scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, mode='min', patience=3)

    best_val_loss = float('inf')
    epochs_no_improve = 0

    # Training loop
    train_start_time = time.time()
    for epoch in range(num_epochs):
        shufflenet_feat.train()
        mobilenet_feat.train()
        fusion_model.train()

        running_loss = 0
        correct = 0
        total = 0
        for inputs, labels in train_loader:
            inputs, labels = inputs.to(device), labels.to(device)
            optimizer.zero_grad()
            feat_sn = shufflenet_feat(inputs)
            feat_mn = mobilenet_feat(inputs)
            outputs = fusion_model((feat_sn, feat_mn))
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()
            running_loss += loss.item() * inputs.size(0)
            preds = outputs.argmax(dim=1)
            correct += (preds == labels).sum().item()
            total += labels.size(0)
        train_loss = running_loss / total
        train_acc = correct / total

        # Validation
        shufflenet_feat.eval()
        mobilenet_feat.eval()
        fusion_model.eval()

        val_running_loss = 0
        val_correct = 0
        val_total = 0
        val_labels = []
        val_preds = []
        with torch.no_grad():
            for inputs, labels in val_loader:
                inputs, labels = inputs.to(device), labels.to(device)
                feat_sn = shufflenet_feat(inputs)
                feat_mn = mobilenet_feat(inputs)
                outputs = fusion_model((feat_sn, feat_mn))
                loss = criterion(outputs, labels)
```

```
            val_running_loss += loss.item() * inputs.size(0)
            preds = outputs.argmax(dim=1)
            val_correct += (preds == labels).sum().item()
            val_total += labels.size(0)
            val_labels.extend(labels.cpu().numpy())
            val_preds.extend(preds.cpu().numpy())
    val_loss = val_running_loss / val_total
    val_acc = val_correct / val_total

    scheduler.step(val_loss)

    print(f"Epoch {epoch+1}/{num_epochs} | Train Loss: {train_loss:.4f} Acc: {train_acc:.4f} | Val Loss: {va

    if val_loss < best_val_loss:
        best_val_loss = val_loss
        epochs_no_improve = 0
        torch.save({
            'fusion': fusion_model.state_dict(),
            'sn_feat': shufflenet_feat.state_dict(),
            'mn_feat': mobilenet_feat.state_dict()
        }, f"best_fusion_fold{fold_idx}.pth")
    else:
        epochs_no_improve += 1
        if epochs_no_improve >= patience:
            print("Early stopping triggered")
            break
train_end_time = time.time()
print(f"Training time: {train_end_time - train_start_time:.2f} seconds")

# Load best weights
checkpoint = torch.load(f"best_fusion_fold{fold_idx}.pth")
fusion_model.load_state_dict(checkpoint['fusion'])
shufflenet_feat.load_state_dict(checkpoint['sn_feat'])
mobilenet_feat.load_state_dict(checkpoint['mn_feat'])

# Test evaluation
fusion_model.eval()
shufflenet_feat.eval()
mobilenet_feat.eval()

test_running_loss = 0
test_correct = 0
test_total = 0
test_labels = []
test_preds = []
test_probs = []

with torch.no_grad():
    for inputs, labels in test_loader:
        inputs, labels = inputs.to(device), labels.to(device)
        feat_sn = shufflenet_feat(inputs)
        feat_mn = mobilenet_feat(inputs)
        outputs = fusion_model((feat_sn, feat_mn))
        loss = criterion(outputs, labels)
        probs = torch.softmax(outputs, dim=1)
        test_running_loss += loss.item() * inputs.size(0)
        preds = outputs.argmax(dim=1)
        test_correct += (preds == labels).sum().item()
        test_total += labels.size(0)
        test_labels.extend(labels.cpu().numpy())
        test_preds.extend(preds.cpu().numpy())
        test_probs.extend(probs.cpu().numpy())

test_loss = test_running_loss / test_total
test_acc = test_correct / test_total
test_labels = np.array(test_labels)
test_preds = np.array(test_preds)
test_probs = np.array(test_probs)

f1 = f1_score(test_labels, test_preds, average='macro')
kappa = cohen_kappa_score(test_labels, test_preds)
try:
    auc_macro = roc_auc_score(test_labels, test_probs, multi_class='ovo', average='macro')
except Exception:
    auc_macro = float('nan')

pr_auc_per_class = []
for i in range(num_classes):
    precision, recall, _ = precision_recall_curve(test_labels == i, test_probs[:, i])
    pr_auc_per_class.append(auc(recall, precision))
pr_auc_macro = np.mean(pr_auc_per_class)

print(f"Fold {fold_idx + 1} Test Metrics:")
```

```
    print(f"Accuracy: {test_acc:.4f}")
    print(f"F1-score (macro): {f1:.4f}")
    print(f"AUC (macro): {auc_macro:.4f}")
    print(f"PR AUC (macro): {pr_auc_macro:.4f}")
    print(f"Cohen's Kappa: {kappa:.4f}")

    fold_metrics.append({
        'accuracy': test_acc,
        'f1': f1,
        'auc': auc_macro,
        'pr_auc': pr_auc_macro,
        'kappa': kappa
    })

# Average CV results
accs = [m['accuracy'] for m in fold_metrics]
f1s = [m['f1'] for m in fold_metrics]
aucs = [m['auc'] for m in fold_metrics if not np.isnan(m['auc'])]
pr_aucs = [m['pr_auc'] for m in fold_metrics]
kappas = [m['kappa'] for m in fold_metrics]

print(f"\n===== Cross-Validation Results =====")
print(f"Average Accuracy: {np.mean(accs):.4f} ± {np.std(accs):.4f}")
print(f"Average F1-score (macro): {np.mean(f1s):.4f} ± {np.std(f1s):.4f}")
if aucs:
    print(f"Average AUC (macro): {np.mean(aucs):.4f} ± {np.std(aucs):.4f}")
else:
    print("Average AUC (macro): NaN (skipped)")
print(f"Average PR AUC (macro): {np.mean(pr_aucs):.4f} ± {np.std(pr_aucs):.4f}")
print(f"Average Cohen's Kappa: {np.mean(kappas):.4f} ± {np.std(kappas):.4f}")
```

```
ShuffleNet feature dim: 1024, MobileNetV2 feature dim: 1280

===== Fold 1 / 5 =====
Epoch 1/30 | Train Loss: 0.4725 Acc: 0.8288 | Val Loss: 0.1440 Acc: 0.9607
Epoch 2/30 | Train Loss: 0.1187 Acc: 0.9616 | Val Loss: 0.0853 Acc: 0.9683
Epoch 3/30 | Train Loss: 0.0529 Acc: 0.9823 | Val Loss: 0.0719 Acc: 0.9784
Epoch 4/30 | Train Loss: 0.0331 Acc: 0.9906 | Val Loss: 0.0375 Acc: 0.9873
Epoch 5/30 | Train Loss: 0.0299 Acc: 0.9904 | Val Loss: 0.0502 Acc: 0.9822
Epoch 6/30 | Train Loss: 0.0211 Acc: 0.9931 | Val Loss: 0.0232 Acc: 0.9924
Epoch 7/30 | Train Loss: 0.0168 Acc: 0.9948 | Val Loss: 0.0634 Acc: 0.9784
Epoch 8/30 | Train Loss: 0.0246 Acc: 0.9923 | Val Loss: 0.0375 Acc: 0.9911
Epoch 9/30 | Train Loss: 0.0213 Acc: 0.9921 | Val Loss: 0.0397 Acc: 0.9835
Epoch 10/30 | Train Loss: 0.0201 Acc: 0.9946 | Val Loss: 0.0517 Acc: 0.9848
Epoch 11/30 | Train Loss: 0.0079 Acc: 0.9975 | Val Loss: 0.0251 Acc: 0.9898
Early stopping triggered
Training time: 326.37 seconds
Fold 1 Test Metrics:
Accuracy: 0.9787
F1-score (macro): 0.9764
AUC (macro): 0.9996
PR AUC (macro): 0.9983
Cohen's Kappa: 0.9730

===== Fold 2 / 5 =====
Epoch 1/30 | Train Loss: 0.0992 Acc: 0.9748 | Val Loss: 0.0686 Acc: 0.9784
Epoch 2/30 | Train Loss: 0.0270 Acc: 0.9927 | Val Loss: 0.0620 Acc: 0.9784
Epoch 3/30 | Train Loss: 0.0279 Acc: 0.9916 | Val Loss: 0.0572 Acc: 0.9797
Epoch 4/30 | Train Loss: 0.0230 Acc: 0.9917 | Val Loss: 0.0562 Acc: 0.9797
Epoch 5/30 | Train Loss: 0.0218 Acc: 0.9925 | Val Loss: 0.0770 Acc: 0.9784
Epoch 6/30 | Train Loss: 0.0221 Acc: 0.9937 | Val Loss: 0.0412 Acc: 0.9886
Epoch 7/30 | Train Loss: 0.0231 Acc: 0.9920 | Val Loss: 0.0452 Acc: 0.9886
Epoch 8/30 | Train Loss: 0.0181 Acc: 0.9956 | Val Loss: 0.0395 Acc: 0.9848
Epoch 9/30 | Train Loss: 0.0133 Acc: 0.9952 | Val Loss: 0.0591 Acc: 0.9822
Epoch 10/30 | Train Loss: 0.0192 Acc: 0.9935 | Val Loss: 0.0547 Acc: 0.9797
Epoch 11/30 | Train Loss: 0.0060 Acc: 0.9982 | Val Loss: 0.0301 Acc: 0.9886
Epoch 12/30 | Train Loss: 0.0125 Acc: 0.9958 | Val Loss: 0.0531 Acc: 0.9886
Epoch 13/30 | Train Loss: 0.0175 Acc: 0.9941 | Val Loss: 0.0453 Acc: 0.9886
Epoch 14/30 | Train Loss: 0.0074 Acc: 0.9975 | Val Loss: 0.0839 Acc: 0.9797
Epoch 15/30 | Train Loss: 0.0134 Acc: 0.9961 | Val Loss: 0.0479 Acc: 0.9822
Epoch 16/30 | Train Loss: 0.0043 Acc: 0.9987 | Val Loss: 0.0260 Acc: 0.9886
Epoch 17/30 | Train Loss: 0.0037 Acc: 0.9989 | Val Loss: 0.0200 Acc: 0.9924
Epoch 18/30 | Train Loss: 0.0021 Acc: 0.9993 | Val Loss: 0.0193 Acc: 0.9937
Epoch 19/30 | Train Loss: 0.0024 Acc: 0.9996 | Val Loss: 0.0229 Acc: 0.9911
Epoch 20/30 | Train Loss: 0.0012 Acc: 0.9997 | Val Loss: 0.0195 Acc: 0.9924
Epoch 21/30 | Train Loss: 0.0007 Acc: 1.0000 | Val Loss: 0.0211 Acc: 0.9924
Epoch 22/30 | Train Loss: 0.0007 Acc: 0.9999 | Val Loss: 0.0174 Acc: 0.9937
Epoch 23/30 | Train Loss: 0.0006 Acc: 1.0000 | Val Loss: 0.0164 Acc: 0.9949
Epoch 24/30 | Train Loss: 0.0014 Acc: 0.9996 | Val Loss: 0.0178 Acc: 0.9949
Epoch 25/30 | Train Loss: 0.0011 Acc: 0.9996 | Val Loss: 0.0165 Acc: 0.9949
Epoch 26/30 | Train Loss: 0.0014 Acc: 0.9993 | Val Loss: 0.0166 Acc: 0.9949
Epoch 27/30 | Train Loss: 0.0013 Acc: 0.9997 | Val Loss: 0.0163 Acc: 0.9949
Epoch 28/30 | Train Loss: 0.0006 Acc: 0.9999 | Val Loss: 0.0202 Acc: 0.9949
Epoch 29/30 | Train Loss: 0.0008 Acc: 0.9996 | Val Loss: 0.0146 Acc: 0.9949
```

```
Epoch 30/30 | Train Loss: 0.0006 Acc: 1.0000 | Val Loss: 0.0149 Acc: 0.9962
Training time: 890.76 seconds
Fold 2 Test Metrics:
Accuracy: 0.9990
F1-score (macro): 0.9989
AUC (macro): 1.0000
PR AUC (macro): 1.0000
Cohen's Kappa: 0.9987

===== Fold 3 / 5 =====
Epoch 1/30 | Train Loss: 0.0553 Acc: 0.9894 | Val Loss: 0.0339 Acc: 0.9886
Epoch 2/30 | Train Loss: 0.0138 Acc: 0.9966 | Val Loss: 0.0250 Acc: 0.9937
Epoch 3/30 | Train Loss: 0.0127 Acc: 0.9963 | Val Loss: 0.0215 Acc: 0.9962
Epoch 4/30 | Train Loss: 0.0132 Acc: 0.9958 | Val Loss: 0.0228 Acc: 0.9924
Epoch 5/30 | Train Loss: 0.0143 Acc: 0.9954 | Val Loss: 0.0557 Acc: 0.9822
Epoch 6/30 | Train Loss: 0.0287 Acc: 0.9916 | Val Loss: 0.0277 Acc: 0.9949
Epoch 7/30 | Train Loss: 0.0125 Acc: 0.9963 | Val Loss: 0.0397 Acc: 0.9873
Epoch 8/30 | Train Loss: 0.0107 Acc: 0.9962 | Val Loss: 0.0114 Acc: 0.9962
Epoch 9/30 | Train Loss: 0.0041 Acc: 0.9987 | Val Loss: 0.0108 Acc: 0.9975
Epoch 10/30 | Train Loss: 0.0022 Acc: 0.9997 | Val Loss: 0.0091 Acc: 0.9962
Epoch 11/30 | Train Loss: 0.0013 Acc: 0.9999 | Val Loss: 0.0082 Acc: 0.9975
Epoch 12/30 | Train Loss: 0.0020 Acc: 0.9994 | Val Loss: 0.0124 Acc: 0.9962
Epoch 13/30 | Train Loss: 0.0015 Acc: 0.9997 | Val Loss: 0.0096 Acc: 0.9975
Epoch 14/30 | Train Loss: 0.0011 Acc: 0.9999 | Val Loss: 0.0091 Acc: 0.9975
Epoch 15/30 | Train Loss: 0.0013 Acc: 0.9996 | Val Loss: 0.0090 Acc: 0.9962
Epoch 16/30 | Train Loss: 0.0008 Acc: 1.0000 | Val Loss: 0.0096 Acc: 0.9962
Early stopping triggered
Training time: 476.07 seconds
Fold 3 Test Metrics:
Accuracy: 0.9995
F1-score (macro): 0.9996
AUC (macro): 1.0000
PR AUC (macro): 1.0000
Cohen's Kappa: 0.9994

===== Fold 4 / 5 =====
Epoch 1/30 | Train Loss: 0.0417 Acc: 0.9920 | Val Loss: 0.0271 Acc: 0.9873
Epoch 2/30 | Train Loss: 0.0066 Acc: 0.9979 | Val Loss: 0.0279 Acc: 0.9898
Epoch 3/30 | Train Loss: 0.0197 Acc: 0.9932 | Val Loss: 0.0584 Acc: 0.9835
Epoch 4/30 | Train Loss: 0.0164 Acc: 0.9945 | Val Loss: 0.0190 Acc: 0.9911
Epoch 5/30 | Train Loss: 0.0137 Acc: 0.9951 | Val Loss: 0.0406 Acc: 0.9873
Epoch 6/30 | Train Loss: 0.0147 Acc: 0.9949 | Val Loss: 0.0353 Acc: 0.9911
Epoch 7/30 | Train Loss: 0.0083 Acc: 0.9975 | Val Loss: 0.0207 Acc: 0.9924
Epoch 8/30 | Train Loss: 0.0022 Acc: 0.9992 | Val Loss: 0.0240 Acc: 0.9937
Epoch 9/30 | Train Loss: 0.0026 Acc: 0.9990 | Val Loss: 0.0234 Acc: 0.9911
Early stopping triggered
Training time: 268.44 seconds
Fold 4 Test Metrics:
Accuracy: 1.0000
F1-score (macro): 1.0000
AUC (macro): 1.0000
PR AUC (macro): 1.0000
Cohen's Kappa: 1.0000

===== Fold 5 / 5 =====
Epoch 1/30 | Train Loss: 0.0397 Acc: 0.9924 | Val Loss: 0.0170 Acc: 0.9949
Epoch 2/30 | Train Loss: 0.0072 Acc: 0.9983 | Val Loss: 0.0298 Acc: 0.9873
Epoch 3/30 | Train Loss: 0.0190 Acc: 0.9938 | Val Loss: 0.0320 Acc: 0.9924
Epoch 4/30 | Train Loss: 0.0128 Acc: 0.9963 | Val Loss: 0.0135 Acc: 0.9962
Epoch 5/30 | Train Loss: 0.0022 Acc: 0.9997 | Val Loss: 0.0105 Acc: 0.9962
Epoch 6/30 | Train Loss: 0.0013 Acc: 0.9999 | Val Loss: 0.0122 Acc: 0.9962
Epoch 7/30 | Train Loss: 0.0095 Acc: 0.9965 | Val Loss: 0.0919 Acc: 0.9772
Epoch 8/30 | Train Loss: 0.0280 Acc: 0.9917 | Val Loss: 0.0206 Acc: 0.9949
Epoch 9/30 | Train Loss: 0.0197 Acc: 0.9951 | Val Loss: 0.0213 Acc: 0.9937
Epoch 10/30 | Train Loss: 0.0051 Acc: 0.9982 | Val Loss: 0.0162 Acc: 0.9949
Early stopping triggered
Training time: 297.28 seconds
Fold 5 Test Metrics:
Accuracy: 1.0000
F1-score (macro): 1.0000
AUC (macro): 1.0000
PR AUC (macro): 1.0000
Cohen's Kappa: 1.0000

===== Cross-Validation Results =====
Average Accuracy: 0.9954 ± 0.0084
Average F1-score (macro): 0.9950 ± 0.0093
Average AUC (macro): 0.9999 ± 0.0002
Average PR AUC (macro): 0.9997 ± 0.0007
Average Cohen's Kappa: 0.9942 ± 0.0106
```