In [7]:
```
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 [8]:
```
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 [9]:

```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 [04:04<00:00,  2.15it/s]
Processing Normal: 100%|██████████| 649/649 [05:01<00:00,  2.16it/s]
Processing Cataract: 100%|██████████| 544/544 [04:14<00:00,  2.13it/s]
Processing Uveitis: 100%|██████████| 223/223 [01:44<00:00,  2.13it/s]
Processing Conjunctivitis: 100%|██████████| 357/357 [02:48<00:00,  2.12it/s]
All images preprocessed and saved successfully!
```

In [ ]:

In [10]:
```
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 [11]:

```
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]:

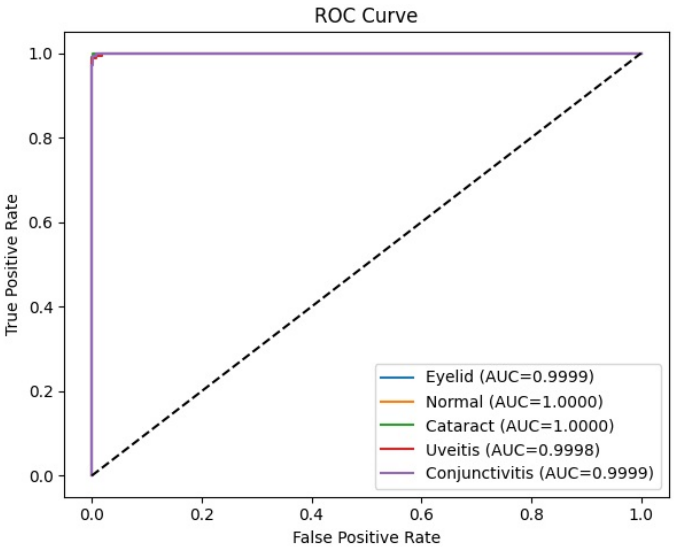
```
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import transforms
from torchvision.models import vgg19, VGG19_Weights
from torch.utils.data import Dataset, DataLoader
from PIL import Image
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score, precision_recall_curve, auc,
import time
import numpy as np
import matplotlib.pyplot as plt
import os

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)}

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

def create_dataset_loader(split_name, split_data):
    image_paths, labels = [], []
    for cls in classes:
        paths = split_data[split_name][cls]
```

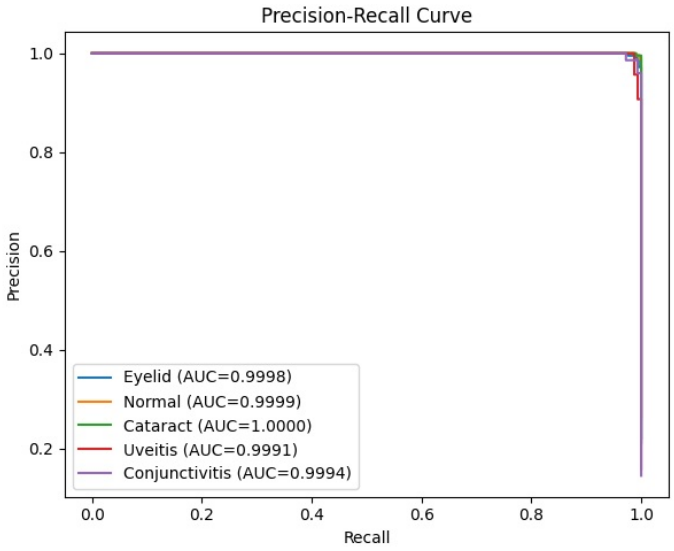
```
        image_paths.extend(paths)
        labels.extend([class_to_idx[cls]]*len(paths))
    dataset = EyeDataset(image_paths, labels, transform=data_transforms[split_name])
    loader = DataLoader(dataset, batch_size=32, shuffle=(split_name=='train'), num_workers=2)
    return dataset, loader

# Assume split_data dictionary is already defined, structured as:
# split_data = {'train': {...}, 'val': {...}, 'test': {...}}

train_dataset, train_loader = create_dataset_loader('train', split_data)
val_dataset, val_loader = create_dataset_loader('val', split_data)
test_dataset, test_loader = create_dataset_loader('test', split_data)

weights = VGG19_Weights.IMAGENET1K_V1
vgg19_model = vgg19(weights=weights)

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)

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

patience = 5
best_val_loss = float('inf')
epochs_no_improve = 0
num_epochs = 30

def train_epoch(model, dataloader):
    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)

train_start_time = time.time()
for epoch in range(num_epochs):
```

```
    train_loss, train_acc = train_epoch(vgg19_model, train_loader)
    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: {val_lo
    if val_loss < best_val_loss:
        best_val_loss = val_loss
        epochs_no_improve = 0
        torch.save(vgg19_model.state_dict(), "best_vgg19.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("best_vgg19.pth"))

def print_classification_reports(model, loader, split_name):
    _, _, labels, preds, probs = eval_model(model, loader)
    print(f"\nClassification report for {split_name}:")
    print(classification_report(labels, preds, target_names=classes, digits=4))

print_classification_reports(vgg19_model, train_loader, 'Train')
print_classification_reports(vgg19_model, val_loader, 'Validation')

test_loss, test_acc, test_labels, test_preds, test_probs = eval_model(vgg19_model, test_loader)
print(f"\nTest Accuracy: {test_acc:.4f}")
print("\nConfusion Matrix:")
print(confusion_matrix(test_labels, test_preds))
print("\nClassification Report (Test):")
print(classification_report(test_labels, test_preds, target_names=classes, digits=4))

plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
for i in range(num_classes):
    fpr, tpr, _ = roc_curve(test_labels == i, test_probs[:, i])
    plt.plot(fpr, tpr, label=f"{classes[i]} (AUC={roc_auc_score(test_labels==i, test_probs[:,i]):.4f})")
plt.plot([0,1], [0,1], 'k--')
plt.title("ROC Curve")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()

plt.subplot(1,2,2)
for i in range(num_classes):
    precision, recall, _ = precision_recall_curve(test_labels == i, test_probs[:, i])
    pr_auc = auc(recall, precision)
    plt.plot(recall, precision, label=f"{classes[i]} (AUC={pr_auc:.4f})")
plt.title("Precision-Recall Curve")
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.legend()
plt.tight_layout()
plt.show()

start_infer = time.time()
_ = eval_model(vgg19_model, test_loader)
end_infer = time.time()
total_infer_time = end_infer - start_infer
inference_per_sample = total_infer_time / len(test_dataset)
print(f"Total inference time on test set: {total_infer_time:.4f} seconds")
print(f"Inference time per sample: {inference_per_sample:.6f} seconds")
```

```
Downloading: "https://download.pytorch.org/models/vgg19-dcbb9e9d.pth" to /root/.cache/torch/hub/checkpoints/vgg1
9-dcbb9e9d.pth
100%|██████████| 548M/548M [00:03<00:00, 190MB/s]
```

```
Epoch 1/30 | Train Loss: 0.6092 Acc: 0.7704 | Val Loss: 0.3281 Acc: 0.8811
Epoch 2/30 | Train Loss: 0.2933 Acc: 0.8993 | Val Loss: 0.2101 Acc: 0.9228
Epoch 3/30 | Train Loss: 0.1479 Acc: 0.9466 | Val Loss: 0.2576 Acc: 0.9187
Epoch 4/30 | Train Loss: 0.1047 Acc: 0.9668 | Val Loss: 0.2062 Acc: 0.9329
Epoch 5/30 | Train Loss: 0.0683 Acc: 0.9779 | Val Loss: 0.1803 Acc: 0.9390
Epoch 6/30 | Train Loss: 0.0881 Acc: 0.9734 | Val Loss: 0.1024 Acc: 0.9654
Epoch 7/30 | Train Loss: 0.0748 Acc: 0.9760 | Val Loss: 0.1896 Acc: 0.9339
Epoch 8/30 | Train Loss: 0.0611 Acc: 0.9811 | Val Loss: 0.1718 Acc: 0.9472
Epoch 9/30 | Train Loss: 0.0413 Acc: 0.9854 | Val Loss: 0.1299 Acc: 0.9624
Epoch 10/30 | Train Loss: 0.0358 Acc: 0.9895 | Val Loss: 0.1234 Acc: 0.9726
Epoch 11/30 | Train Loss: 0.0101 Acc: 0.9972 | Val Loss: 0.0763 Acc: 0.9817
Epoch 12/30 | Train Loss: 0.0011 Acc: 1.0000 | Val Loss: 0.0777 Acc: 0.9848
Epoch 13/30 | Train Loss: 0.0006 Acc: 1.0000 | Val Loss: 0.0795 Acc: 0.9848
Epoch 14/30 | Train Loss: 0.0004 Acc: 1.0000 | Val Loss: 0.0805 Acc: 0.9837
Epoch 15/30 | Train Loss: 0.0003 Acc: 1.0000 | Val Loss: 0.0828 Acc: 0.9837

Epoch 16/30 | Train Loss: 0.0002 Acc: 1.0000 | Val Loss: 0.0829 Acc: 0.9837
Early stopping triggered
Training time: 1265.85 seconds

Classification report for Train:
                precision    recall  f1-score   support

        Eyelid     1.0000    1.0000    1.0000      1680
        Normal     1.0000    1.0000    1.0000      2076
      Cataract     1.0000    1.0000    1.0000      1740
       Uveitis     1.0000    1.0000    1.0000      1248
Conjunctivitis     1.0000    1.0000    1.0000      1142

      accuracy                         1.0000      7886
     macro avg     1.0000    1.0000    1.0000      7886
  weighted avg     1.0000    1.0000    1.0000      7886


Classification report for Validation:
                precision    recall  f1-score   support

        Eyelid     0.9717    0.9810    0.9763       210
        Normal     0.9961    0.9961    0.9961       259
      Cataract     0.9861    0.9816    0.9838       217
       Uveitis     0.9625    0.9872    0.9747       156
Conjunctivitis     0.9854    0.9507    0.9677       142

      accuracy                         0.9817       984
     macro avg     0.9804    0.9793    0.9797       984
  weighted avg     0.9818    0.9817    0.9817       984


Test Accuracy: 0.9889

Confusion Matrix:
[[209   0   0   1   0]
 [  2 258   0   1   0]
 [  1   0 217   1   0]
 [  0   0   0 155   2]
 [  2   0   1   0 141]]

Classification Report (Test):
                precision    recall  f1-score   support

        Eyelid     0.9766    0.9952    0.9858       210
        Normal     1.0000    0.9885    0.9942       261
      Cataract     0.9954    0.9909    0.9931       219
       Uveitis     0.9810    0.9873    0.9841       157
Conjunctivitis     0.9860    0.9792    0.9826       144

      accuracy                         0.9889       991
     macro avg     0.9878    0.9882    0.9880       991
  weighted avg     0.9890    0.9889    0.9889       991
```






```
Total inference time on test set: 3.4276 seconds
Inference time per sample: 0.003459 seconds
```

In [11]:

```python
from torchinfo import summary
from torchvision.models import vgg19, VGG19_Weights
import torch

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

weights = VGG19_Weights.IMAGENET1K_V1
model = vgg19(weights=weights)
model.classifier = torch.nn.Sequential(
    torch.nn.Linear(25088, 4096),
    torch.nn.ReLU(inplace=True),
    torch.nn.Dropout(0.5),
    torch.nn.Linear(4096, 4096),
    torch.nn.ReLU(inplace=True),
    torch.nn.Dropout(0.5),
    torch.nn.Linear(4096, 5)  # number of classes
)
model = model.to(device)

summary(model, input_size=(1, 3, 224, 224))
```

Out[11]:
```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
VGG                                      [1, 5]                    --
├─Sequential: 1-1                        [1, 512, 7, 7]            --
│    └─Conv2d: 2-1                       [1, 64, 224, 224]         1,792
│    └─ReLU: 2-2                         [1, 64, 224, 224]         --
│    └─Conv2d: 2-3                       [1, 64, 224, 224]         36,928
│    └─ReLU: 2-4                         [1, 64, 224, 224]         --
│    └─MaxPool2d: 2-5                    [1, 64, 112, 112]         --
│    └─Conv2d: 2-6                       [1, 128, 112, 112]        73,856
│    └─ReLU: 2-7                         [1, 128, 112, 112]        --
│    └─Conv2d: 2-8                       [1, 128, 112, 112]        147,584
│    └─ReLU: 2-9                         [1, 128, 112, 112]        --
│    └─MaxPool2d: 2-10                   [1, 128, 56, 56]          --
│    └─Conv2d: 2-11                      [1, 256, 56, 56]          295,168
│    └─ReLU: 2-12                        [1, 256, 56, 56]          --
│    └─Conv2d: 2-13                      [1, 256, 56, 56]          590,080
│    └─ReLU: 2-14                        [1, 256, 56, 56]          --
│    └─Conv2d: 2-15                      [1, 256, 56, 56]          590,080
│    └─ReLU: 2-16                        [1, 256, 56, 56]          --
│    └─Conv2d: 2-17                      [1, 256, 56, 56]          590,080
│    └─ReLU: 2-18                        [1, 256, 56, 56]          --
│    └─MaxPool2d: 2-19                   [1, 256, 28, 28]          --
│    └─Conv2d: 2-20                      [1, 512, 28, 28]          1,180,160
│    └─ReLU: 2-21                        [1, 512, 28, 28]          --
│    └─Conv2d: 2-22                      [1, 512, 28, 28]          2,359,808
│    └─ReLU: 2-23                        [1, 512, 28, 28]          --
│    └─Conv2d: 2-24                      [1, 512, 28, 28]          2,359,808
│    └─ReLU: 2-25                        [1, 512, 28, 28]          --
│    └─Conv2d: 2-26                      [1, 512, 28, 28]          2,359,808
│    └─ReLU: 2-27                        [1, 512, 28, 28]          --
│    └─MaxPool2d: 2-28                   [1, 512, 14, 14]          --
│    └─Conv2d: 2-29                      [1, 512, 14, 14]          2,359,808
│    └─ReLU: 2-30                        [1, 512, 14, 14]          --
│    └─Conv2d: 2-31                      [1, 512, 14, 14]          2,359,808
│    └─ReLU: 2-32                        [1, 512, 14, 14]          --
│    └─Conv2d: 2-33                      [1, 512, 14, 14]          2,359,808
│    └─ReLU: 2-34                        [1, 512, 14, 14]          --
│    └─Conv2d: 2-35                      [1, 512, 14, 14]          2,359,808
│    └─ReLU: 2-36                        [1, 512, 14, 14]          --
│    └─MaxPool2d: 2-37                   [1, 512, 7, 7]            --
├─AdaptiveAvgPool2d: 1-2                 [1, 512, 7, 7]            --
├─Sequential: 1-3                        [1, 5]                    --
│    └─Linear: 2-38                      [1, 4096]                 102,764,544
│    └─ReLU: 2-39                        [1, 4096]                 --
│    └─Dropout: 2-40                     [1, 4096]                 --
│    └─Linear: 2-41                      [1, 4096]                 16,781,312
│    └─ReLU: 2-42                        [1, 4096]                 --
│    └─Dropout: 2-43                     [1, 4096]                 --
│    └─Linear: 2-44                      [1, 5]                    20,485
==========================================================================================
Total params: 139,590,725
Trainable params: 139,590,725
Non-trainable params: 0
Total mult-adds (Units.GIGABYTES): 19.64
==========================================================================================
Input size (MB): 0.60
Forward/backward pass size (MB): 118.88
Params size (MB): 558.36
Estimated Total Size (MB): 677.85
==========================================================================================
```

In [12]:
```python
from sklearn.metrics import cohen_kappa_score
from statsmodels.stats.proportion import proportion_confint

# Compute Cohen's Kappa
kappa = cohen_kappa_score(test_labels, test_preds)
print(f"Cohen's Kappa: {kappa:.4f}")

# Compute 95% Confidence Interval for Accuracy
correct_preds = np.sum(test_labels == test_preds)
n = len(test_labels)
acc_lower, acc_upper = proportion_confint(count=correct_preds, nobs=n, alpha=0.05, method='wilson')
print(f"Test Accuracy: {test_acc:.4f}")
print(f"95% Confidence Interval for Accuracy: [{acc_lower:.4f}, {acc_upper:.4f}]")
```

```
Cohen's Kappa: 0.9860
Test Accuracy: 0.9889
95% Confidence Interval for Accuracy: [0.9802, 0.9938]
```

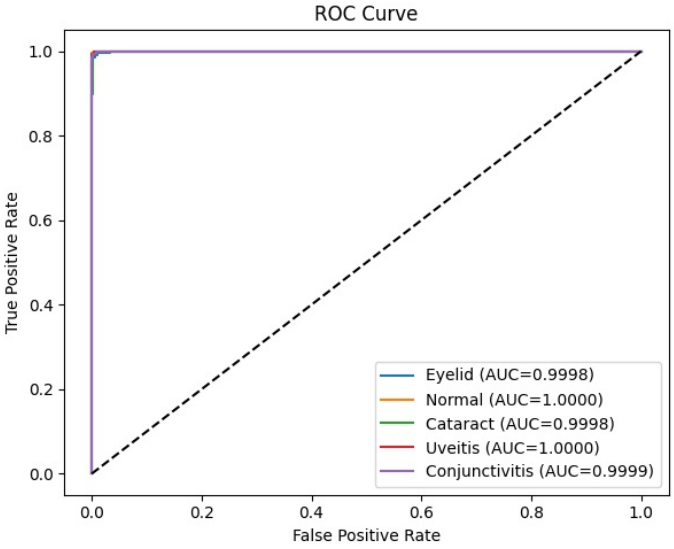
In [12]:
```python
import torch
import torch.nn as nn
import torch.optim as optim
```

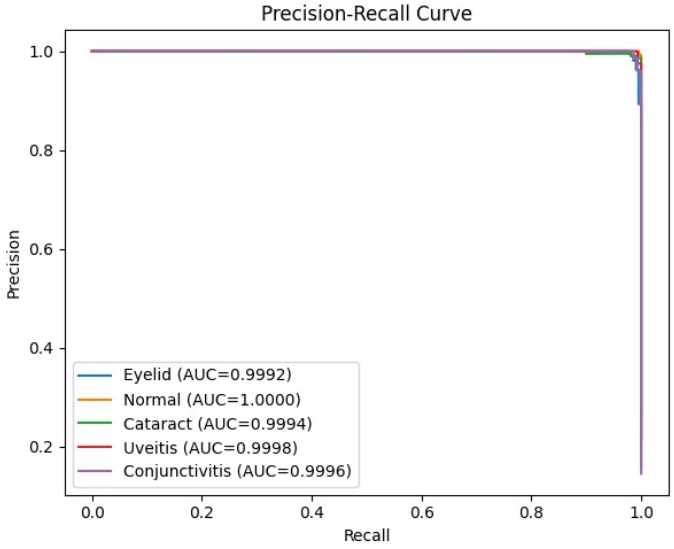
```
from torchvision import transforms
from torchvision.models import mobilenet_v2, MobileNet_V2_Weights
from torch.utils.data import Dataset, DataLoader
from PIL import Image
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score, precision_recall_curve, auc,
import time
import numpy as np
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)}

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

def create_dataset_loader(split_name, split_data):
    image_paths, labels = [], []
    for cls in classes:
        paths = split_data[split_name][cls]
        image_paths.extend(paths)
        labels.extend([class_to_idx[cls]]*len(paths))
    dataset = EyeDataset(image_paths, labels, transform=data_transforms[split_name])
    loader = DataLoader(dataset, batch_size=32, shuffle=(split_name=='train'), num_workers=2)
    return dataset, loader

train_dataset, train_loader = create_dataset_loader('train', split_data)
val_dataset, val_loader = create_dataset_loader('val', split_data)
test_dataset, test_loader = create_dataset_loader('test', split_data)

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

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

patience = 5
best_val_loss = float('inf')
epochs_no_improve = 0
num_epochs = 30

def train_epoch(model, dataloader):
    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)

train_start_time = time.time()
for epoch in range(num_epochs):
    train_loss, train_acc = train_epoch(mobilenet_model, train_loader)
    val_loss, val_acc, _, _, _ = eval_model(mobilenet_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: {val_lo
    if val_loss < best_val_loss:
        best_val_loss = val_loss
        epochs_no_improve = 0
        torch.save(mobilenet_model.state_dict(), "best_mobilenetv2.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")

mobilenet_model.load_state_dict(torch.load("best_mobilenetv2.pth"))

def print_classification_reports(model, loader, split_name):
    _, _, labels, preds, probs = eval_model(model, loader)
    print(f"\nClassification report for {split_name}:")
    print(classification_report(labels, preds, target_names=classes, digits=4))

print_classification_reports(mobilenet_model, train_loader, 'Train')
print_classification_reports(mobilenet_model, val_loader, 'Validation')

test_loss, test_acc, test_labels, test_preds, test_probs = eval_model(mobilenet_model, test_loader)
print(f"\nTest Accuracy: {test_acc:.4f}")
print("\nConfusion Matrix:")
print(confusion_matrix(test_labels, test_preds))
print("\nClassification Report (Test):")
print(classification_report(test_labels, test_preds, target_names=classes, digits=4))

plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
for i in range(num_classes):
    fpr, tpr, _ = roc_curve(test_labels == i, test_probs[:, i])
    plt.plot(fpr, tpr, label=f"{classes[i]} (AUC={roc_auc_score(test_labels==i, test_probs[:,i]):.4f})")
plt.plot([0,1], [0,1], 'k--')
plt.title("ROC Curve")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()

plt.subplot(1,2,2)
```

```python
for i in range(num_classes):
    precision, recall, _ = precision_recall_curve(test_labels == i, test_probs[:, i])
    pr_auc = auc(recall, precision)
    plt.plot(recall, precision, label=f"{classes[i]} (AUC={pr_auc:.4f})")
plt.title("Precision-Recall Curve")
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.legend()
plt.tight_layout()
plt.show()

start_infer = time.time()
_ = eval_model(mobilenet_model, test_loader)
end_infer = time.time()
total_infer_time = end_infer - start_infer
print(f"Total inference time on test set: {total_infer_time:.4f} seconds")
print(f"Inference time per sample: {total_infer_time/len(test_dataset):.6f} seconds")
```

```
Epoch 1/30 | Train Loss: 0.4644 Acc: 0.8312 | Val Loss: 0.1512 Acc: 0.9482
Epoch 2/30 | Train Loss: 0.1159 Acc: 0.9612 | Val Loss: 0.1073 Acc: 0.9654
Epoch 3/30 | Train Loss: 0.0392 Acc: 0.9893 | Val Loss: 0.0702 Acc: 0.9807
Epoch 4/30 | Train Loss: 0.0284 Acc: 0.9923 | Val Loss: 0.0714 Acc: 0.9776
Epoch 5/30 | Train Loss: 0.0309 Acc: 0.9916 | Val Loss: 0.0698 Acc: 0.9776
Epoch 6/30 | Train Loss: 0.0230 Acc: 0.9938 | Val Loss: 0.0606 Acc: 0.9766
Epoch 7/30 | Train Loss: 0.0262 Acc: 0.9923 | Val Loss: 0.0498 Acc: 0.9837
Epoch 8/30 | Train Loss: 0.0136 Acc: 0.9966 | Val Loss: 0.0648 Acc: 0.9837
Epoch 9/30 | Train Loss: 0.0094 Acc: 0.9977 | Val Loss: 0.0380 Acc: 0.9888
Epoch 10/30 | Train Loss: 0.0097 Acc: 0.9967 | Val Loss: 0.0518 Acc: 0.9837
Epoch 11/30 | Train Loss: 0.0281 Acc: 0.9915 | Val Loss: 0.0506 Acc: 0.9848
Epoch 12/30 | Train Loss: 0.0138 Acc: 0.9959 | Val Loss: 0.0292 Acc: 0.9919
Epoch 13/30 | Train Loss: 0.0055 Acc: 0.9984 | Val Loss: 0.0172 Acc: 0.9949
Epoch 14/30 | Train Loss: 0.0034 Acc: 0.9995 | Val Loss: 0.0230 Acc: 0.9939
Epoch 15/30 | Train Loss: 0.0160 Acc: 0.9951 | Val Loss: 0.1171 Acc: 0.9715
Epoch 16/30 | Train Loss: 0.0325 Acc: 0.9896 | Val Loss: 0.0565 Acc: 0.9807
Epoch 17/30 | Train Loss: 0.0156 Acc: 0.9947 | Val Loss: 0.0333 Acc: 0.9878
Epoch 18/30 | Train Loss: 0.0051 Acc: 0.9990 | Val Loss: 0.0214 Acc: 0.9929
Early stopping triggered
Training time: 441.63 seconds

Classification report for Train:
                precision    recall  f1-score   support

        Eyelid     1.0000    1.0000    1.0000      1680
        Normal     1.0000    1.0000    1.0000      2076
      Cataract     1.0000    1.0000    1.0000      1740
       Uveitis     1.0000    1.0000    1.0000      1248
Conjunctivitis     1.0000    1.0000    1.0000      1142

      accuracy                         1.0000      7886
     macro avg     1.0000    1.0000    1.0000      7886
  weighted avg     1.0000    1.0000    1.0000      7886


Classification report for Validation:
                precision    recall  f1-score   support

        Eyelid     0.9952    0.9905    0.9928       210
        Normal     0.9885    1.0000    0.9942       259
      Cataract     1.0000    0.9954    0.9977       217
       Uveitis     1.0000    0.9872    0.9935       156
Conjunctivitis     0.9930    1.0000    0.9965       142

      accuracy                         0.9949       984
     macro avg     0.9954    0.9946    0.9950       984
  weighted avg     0.9950    0.9949    0.9949       984


Test Accuracy: 0.9919

Confusion Matrix:
[[206   1   2   0   1]
 [  0 261   0   0   0]
 [  1   1 217   0   0]
 [  0   0   0 156   1]
 [  0   1   0   0 143]]

Classification Report (Test):
                precision    recall  f1-score   support

        Eyelid     0.9952    0.9810    0.9880       210
        Normal     0.9886    1.0000    0.9943       261
      Cataract     0.9909    0.9909    0.9909       219
       Uveitis     1.0000    0.9936    0.9968       157
Conjunctivitis     0.9862    0.9931    0.9896       144

      accuracy                         0.9919       991
     macro avg     0.9922    0.9917    0.9919       991
  weighted avg     0.9920    0.9919    0.9919       991
```






```
Total inference time on test set: 1.3847 seconds
Inference time per sample: 0.001397 seconds
```

In [14]:
```python
from sklearn.metrics import cohen_kappa_score
from statsmodels.stats.proportion import proportion_confint

# Compute Cohen's Kappa
kappa = cohen_kappa_score(test_labels, test_preds)
print(f"Cohen's Kappa: {kappa:.4f}")

# Compute 95% Confidence Interval for Accuracy
correct_preds = np.sum(test_labels == test_preds)
n = len(test_labels)
acc_lower, acc_upper = proportion_confint(count=correct_preds, nobs=n, alpha=0.05, method='wilson')
print(f"Test Accuracy: {test_acc:.4f}")
print(f"95% Confidence Interval for Accuracy: [{acc_lower:.4f}, {acc_upper:.4f}]")
```

```
Cohen's Kappa: 0.9898
Test Accuracy: 0.9919
95% Confidence Interval for Accuracy: [0.9842, 0.9959]
```

In [15]:
```python
from torchinfo import summary

summary(mobilenet_model, input_size=(32, 3, 224, 224), device=device.type)
```

Out[15]:

```
==========================================================================================
Layer (type:depth-idx)                                  Output Shape              Param #
==========================================================================================
MobileNetV2                                             [32, 5]                   --
├─Sequential: 1-1                                       [32, 1280, 7, 7]          --
│    └─Conv2dNormActivation: 2-1                        [32, 32, 112, 112]        --
│    │    └─Conv2d: 3-1                                 [32, 32, 112, 112]        864
│    │    └─BatchNorm2d: 3-2                            [32, 32, 112, 112]        64
│    │    └─ReLU6: 3-3                                  [32, 32, 112, 112]        --
│    └─InvertedResidual: 2-2                            [32, 16, 112, 112]        --
│    │    └─Sequential: 3-4                             [32, 16, 112, 112]        896
│    └─InvertedResidual: 2-3                            [32, 24, 56, 56]          --
│    │    └─Sequential: 3-5                             [32, 24, 56, 56]          5,136
│    └─InvertedResidual: 2-4                            [32, 24, 56, 56]          --
│    │    └─Sequential: 3-6                             [32, 24, 56, 56]          8,832
│    └─InvertedResidual: 2-5                            [32, 32, 28, 28]          --
│    │    └─Sequential: 3-7                             [32, 32, 28, 28]          10,000
│    └─InvertedResidual: 2-6                            [32, 32, 28, 28]          --
│    │    └─Sequential: 3-8                             [32, 32, 28, 28]          14,848
│    └─InvertedResidual: 2-7                            [32, 32, 28, 28]          --
│    │    └─Sequential: 3-9                             [32, 32, 28, 28]          14,848
│    └─InvertedResidual: 2-8                            [32, 64, 14, 14]          --
│    │    └─Sequential: 3-10                            [32, 64, 14, 14]          21,056
│    └─InvertedResidual: 2-9                            [32, 64, 14, 14]          --
│    │    └─Sequential: 3-11                            [32, 64, 14, 14]          54,272
│    └─InvertedResidual: 2-10                           [32, 64, 14, 14]          --
│    │    └─Sequential: 3-12                            [32, 64, 14, 14]          54,272
│    └─InvertedResidual: 2-11                           [32, 64, 14, 14]          --
│    │    └─Sequential: 3-13                            [32, 64, 14, 14]          54,272
│    └─InvertedResidual: 2-12                           [32, 96, 14, 14]          --
│    │    └─Sequential: 3-14                            [32, 96, 14, 14]          66,624
│    └─InvertedResidual: 2-13                           [32, 96, 14, 14]          --
│    │    └─Sequential: 3-15                            [32, 96, 14, 14]          118,272
│    └─InvertedResidual: 2-14                           [32, 96, 14, 14]          --
│    │    └─Sequential: 3-16                            [32, 96, 14, 14]          118,272
│    └─InvertedResidual: 2-15                           [32, 160, 7, 7]           --
│    │    └─Sequential: 3-17                            [32, 160, 7, 7]           155,264
│    └─InvertedResidual: 2-16                           [32, 160, 7, 7]           --
│    │    └─Sequential: 3-18                            [32, 160, 7, 7]           320,000
│    └─InvertedResidual: 2-17                           [32, 160, 7, 7]           --
│    │    └─Sequential: 3-19                            [32, 160, 7, 7]           320,000
│    └─InvertedResidual: 2-18                           [32, 320, 7, 7]           --
│    │    └─Sequential: 3-20                            [32, 320, 7, 7]           473,920
│    └─Conv2dNormActivation: 2-19                       [32, 1280, 7, 7]          --
│    │    └─Conv2d: 3-21                                [32, 1280, 7, 7]          409,600
│    │    └─BatchNorm2d: 3-22                           [32, 1280, 7, 7]          2,560
│    │    └─ReLU6: 3-23                                 [32, 1280, 7, 7]          --
├─Sequential: 1-2                                       [32, 5]                   --
│    └─Dropout: 2-20                                    [32, 1280]                --
│    └─Linear: 2-21                                     [32, 5]                   6,405
==========================================================================================
Total params: 2,230,277
Trainable params: 2,230,277
Non-trainable params: 0
Total mult-adds (Units.GIGABYTES): 9.59
==========================================================================================
Input size (MB): 19.27
Forward/backward pass size (MB): 3419.19
Params size (MB): 8.92
Estimated Total Size (MB): 3447.38
==========================================================================================
```

In [16]:

```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 classification_report, confusion_matrix, roc_auc_score, precision_recall_curve, auc,
import time
import numpy as np
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)}

data_transforms = {
    'train': transforms.Compose([
        transforms.Resize((224, 224)),  # ShuffleNetV2 default input size 224x224
```

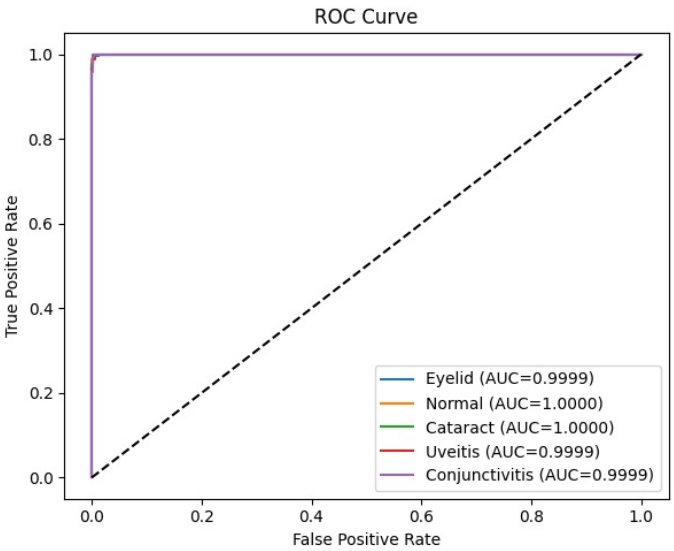
```
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],
                             [0.229, 0.224, 0.225])
    ]),
    'val': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],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

def create_dataset_loader(split_name, split_data):
    image_paths, labels = [], []
    for cls in classes:
        paths = split_data[split_name][cls]
        image_paths.extend(paths)
        labels.extend([class_to_idx[cls]]*len(paths))
    dataset = EyeDataset(image_paths, labels, transform=data_transforms[split_name])
    loader = DataLoader(dataset, batch_size=32, shuffle=(split_name=='train'), num_workers=2)
    return dataset, loader

train_dataset, train_loader = create_dataset_loader('train', split_data)
val_dataset, val_loader = create_dataset_loader('val', split_data)
test_dataset, test_loader = create_dataset_loader('test', split_data)

weights = ShuffleNet_V2_X1_0_Weights.IMAGENET1K_V1
shufflenet_model = shufflenet_v2_x1_0(weights=weights)

# Replace final classifier to output num_classes
shufflenet_model.fc = nn.Linear(shufflenet_model.fc.in_features, num_classes)
shufflenet_model = shufflenet_model.to(device)

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

patience = 5
best_val_loss = float('inf')
epochs_no_improve = 0
num_epochs = 30

def train_epoch(model, dataloader):
    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
```

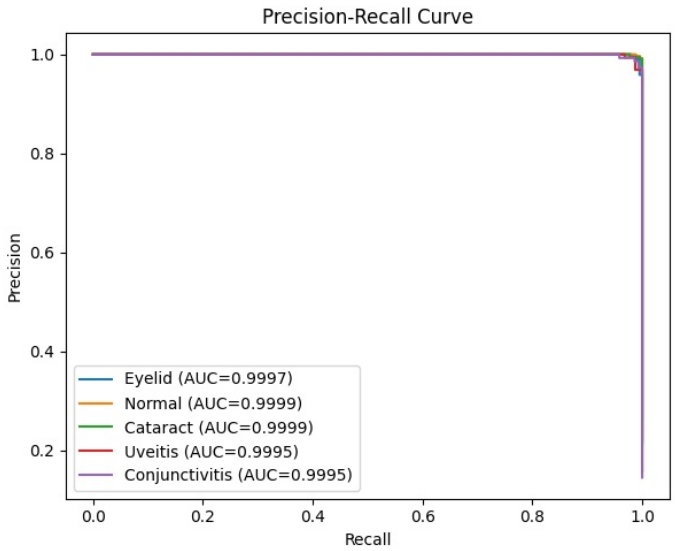
```
    correct = 0
    total = 0
    all_labels = []
    all_preds = []
    all_probs = []
    with torch.no_grad():
        for inputs, labels in dataloader:
            inputs, labels = inputs.to(device), labels.to(device)
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            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)

train_start_time = time.time()
for epoch in range(num_epochs):
    train_loss, train_acc = train_epoch(shufflenet_model, train_loader)
    val_loss, val_acc, _, _, _ = eval_model(shufflenet_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: {val_lo
    if val_loss < best_val_loss:
        best_val_loss = val_loss
        epochs_no_improve = 0
        torch.save(shufflenet_model.state_dict(), "best_shufflenetv2.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")

shufflenet_model.load_state_dict(torch.load("best_shufflenetv2.pth"))

def print_classification_reports(model, loader, split_name):
    _, _, labels, preds, probs = eval_model(model, loader)
    print(f"\nClassification report for {split_name}:")
    print(classification_report(labels, preds, target_names=classes, digits=4))

print_classification_reports(shufflenet_model, train_loader, 'Train')
print_classification_reports(shufflenet_model, val_loader, 'Validation')

test_loss, test_acc, test_labels, test_preds, test_probs = eval_model(shufflenet_model, test_loader)
print(f"\nTest Accuracy: {test_acc:.4f}")
print("\nConfusion Matrix:")
print(confusion_matrix(test_labels, test_preds))
print("\nClassification Report (Test):")
print(classification_report(test_labels, test_preds, target_names=classes, digits=4))

plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
for i in range(num_classes):
    fpr, tpr, _ = roc_curve(test_labels == i, test_probs[:, i])
    plt.plot(fpr, tpr, label=f"{classes[i]} (AUC={roc_auc_score(test_labels==i, test_probs[:,i]):.4f})")
plt.plot([0,1], [0,1], 'k--')
plt.title("ROC Curve")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()

plt.subplot(1,2,2)
for i in range(num_classes):
    precision, recall, _ = precision_recall_curve(test_labels == i, test_probs[:, i])
    pr_auc = auc(recall, precision)
    plt.plot(recall, precision, label=f"{classes[i]} (AUC={pr_auc:.4f})")
plt.title("Precision-Recall Curve")
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.legend()
plt.tight_layout()
plt.show()

start_infer = time.time()
_ = eval_model(shufflenet_model, test_loader)
end_infer = time.time()
total_infer_time = end_infer - start_infer
print(f"Total inference time on test set: {total_infer_time:.4f} seconds")
```

```
print(f"Inference time per sample: {total_infer_time/len(test_dataset):.6f} seconds")
```

```
Downloading: "https://download.pytorch.org/models/shufflenetv2_x1-5666bf0f80.pth" to /root/.cache/torch/hub/checkpoints/shufflenetv2_x1-5666bf0f80.pth
100%|██████████| 8.79M/8.79M [00:00<00:00, 93.6MB/s]
```

```
Epoch 1/30 | Train Loss: 1.2422 Acc: 0.6046 | Val Loss: 0.7726 Acc: 0.7724
Epoch 2/30 | Train Loss: 0.5853 Acc: 0.8289 | Val Loss: 0.3661 Acc: 0.8933
Epoch 3/30 | Train Loss: 0.2955 Acc: 0.9214 | Val Loss: 0.2093 Acc: 0.9380
Epoch 4/30 | Train Loss: 0.1628 Acc: 0.9597 | Val Loss: 0.1468 Acc: 0.9563
Epoch 5/30 | Train Loss: 0.0958 Acc: 0.9774 | Val Loss: 0.1034 Acc: 0.9705
Epoch 6/30 | Train Loss: 0.0701 Acc: 0.9838 | Val Loss: 0.0876 Acc: 0.9665
Epoch 7/30 | Train Loss: 0.0477 Acc: 0.9890 | Val Loss: 0.0990 Acc: 0.9715
Epoch 8/30 | Train Loss: 0.0441 Acc: 0.9886 | Val Loss: 0.0478 Acc: 0.9827
Epoch 9/30 | Train Loss: 0.0277 Acc: 0.9938 | Val Loss: 0.0554 Acc: 0.9868
Epoch 10/30 | Train Loss: 0.0287 Acc: 0.9926 | Val Loss: 0.0588 Acc: 0.9817
Epoch 11/30 | Train Loss: 0.0196 Acc: 0.9954 | Val Loss: 0.0625 Acc: 0.9827
Epoch 12/30 | Train Loss: 0.0213 Acc: 0.9951 | Val Loss: 0.0778 Acc: 0.9746
Epoch 13/30 | Train Loss: 0.0171 Acc: 0.9966 | Val Loss: 0.0459 Acc: 0.9848
Epoch 14/30 | Train Loss: 0.0115 Acc: 0.9980 | Val Loss: 0.0453 Acc: 0.9858
Epoch 15/30 | Train Loss: 0.0097 Acc: 0.9986 | Val Loss: 0.0461 Acc: 0.9858
Epoch 16/30 | Train Loss: 0.0086 Acc: 0.9990 | Val Loss: 0.0463 Acc: 0.9848
Epoch 17/30 | Train Loss: 0.0083 Acc: 0.9990 | Val Loss: 0.0427 Acc: 0.9848
Epoch 18/30 | Train Loss: 0.0093 Acc: 0.9982 | Val Loss: 0.0413 Acc: 0.9858
Epoch 19/30 | Train Loss: 0.0094 Acc: 0.9986 | Val Loss: 0.0398 Acc: 0.9868
Epoch 20/30 | Train Loss: 0.0070 Acc: 0.9994 | Val Loss: 0.0385 Acc: 0.9858
Epoch 21/30 | Train Loss: 0.0098 Acc: 0.9984 | Val Loss: 0.0455 Acc: 0.9858
Epoch 22/30 | Train Loss: 0.0091 Acc: 0.9984 | Val Loss: 0.0404 Acc: 0.9868
Epoch 23/30 | Train Loss: 0.0065 Acc: 0.9992 | Val Loss: 0.0385 Acc: 0.9878
Epoch 24/30 | Train Loss: 0.0064 Acc: 0.9994 | Val Loss: 0.0415 Acc: 0.9868
Epoch 25/30 | Train Loss: 0.0056 Acc: 0.9996 | Val Loss: 0.0520 Acc: 0.9858
Epoch 26/30 | Train Loss: 0.0075 Acc: 0.9991 | Val Loss: 0.0511 Acc: 0.9858
Epoch 27/30 | Train Loss: 0.0062 Acc: 0.9992 | Val Loss: 0.0478 Acc: 0.9858
Epoch 28/30 | Train Loss: 0.0085 Acc: 0.9991 | Val Loss: 0.0460 Acc: 0.9848
Early stopping triggered
Training time: 430.90 seconds

Classification report for Train:
                precision    recall  f1-score   support

        Eyelid     1.0000    1.0000    1.0000      1680
        Normal     1.0000    1.0000    1.0000      2076
      Cataract     1.0000    1.0000    1.0000      1740
       Uveitis     1.0000    1.0000    1.0000      1248
Conjunctivitis     1.0000    1.0000    1.0000      1142

      accuracy                         1.0000      7886
     macro avg     1.0000    1.0000    1.0000      7886
  weighted avg     1.0000    1.0000    1.0000      7886


Classification report for Validation:
                precision    recall  f1-score   support

        Eyelid     1.0000    0.9857    0.9928       210
        Normal     1.0000    1.0000    1.0000       259
      Cataract     0.9954    0.9908    0.9931       217
       Uveitis     0.9448    0.9872    0.9655       156
Conjunctivitis     0.9856    0.9648    0.9751       142

      accuracy                         0.9878       984
     macro avg     0.9852    0.9857    0.9853       984
  weighted avg     0.9881    0.9878    0.9879       984


Test Accuracy: 0.9919

Confusion Matrix:
[[207   2   0   0   1]
 [  1 260   0   0   0]
 [  0   0 218   1   0]
 [  0   0   1 155   1]
 [  0   0   0   1 143]]

Classification Report (Test):
                precision    recall  f1-score   support

        Eyelid     0.9952    0.9857    0.9904       210
        Normal     0.9924    0.9962    0.9943       261
      Cataract     0.9954    0.9954    0.9954       219
       Uveitis     0.9873    0.9873    0.9873       157
Conjunctivitis     0.9862    0.9931    0.9896       144

      accuracy                         0.9919       991

     macro avg     0.9913    0.9915    0.9914       991
  weighted avg     0.9919    0.9919    0.9919       991
```






```
Total inference time on test set: 1.4236 seconds
Inference time per sample: 0.001437 seconds
```

In [17]:
```python
from sklearn.metrics import cohen_kappa_score
from statsmodels.stats.proportion import proportion_confint

# Compute Cohen's Kappa
kappa = cohen_kappa_score(test_labels, test_preds)
print(f"Cohen's Kappa: {kappa:.4f}")

# Compute 95% Confidence Interval for Accuracy
correct_preds = np.sum(test_labels == test_preds)
n = len(test_labels)
acc_lower, acc_upper = proportion_confint(count=correct_preds, nobs=n, alpha=0.05, method='wilson')
print(f"Test Accuracy: {test_acc:.4f}")
print(f"95% Confidence Interval for Accuracy: [{acc_lower:.4f}, {acc_upper:.4f}]")
```

```
Cohen's Kappa: 0.9898
Test Accuracy: 0.9919
95% Confidence Interval for Accuracy: [0.9842, 0.9959]
```

In [18]:
```python
from torchinfo import summary

# Assuming shufflenet_model is already defined and on device (cpu or cuda)
summary(shufflenet_model, input_size=(1, 3, 224, 224), device=str(device))
```

Out[18]:

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
ShuffleNetV2                             [1, 5]                    --
├─Sequential: 1-1                        [1, 24, 112, 112]         --
│    └─Conv2d: 2-1                       [1, 24, 112, 112]         648
│    └─BatchNorm2d: 2-2                  [1, 24, 112, 112]         48
│    └─ReLU: 2-3                         [1, 24, 112, 112]         --
├─MaxPool2d: 1-2                         [1, 24, 56, 56]           --
├─Sequential: 1-3                        [1, 116, 28, 28]          --
│    └─InvertedResidual: 2-4             [1, 116, 28, 28]          --
│    │    └─Sequential: 3-1              [1, 58, 28, 28]           1,772
│    │    └─Sequential: 3-2              [1, 58, 28, 28]           5,626
│    └─InvertedResidual: 2-5             [1, 116, 28, 28]          --
│    │    └─Sequential: 3-3              [1, 58, 28, 28]           7,598
│    └─InvertedResidual: 2-6             [1, 116, 28, 28]          --
│    │    └─Sequential: 3-4              [1, 58, 28, 28]           7,598
│    └─InvertedResidual: 2-7             [1, 116, 28, 28]          --
│    │    └─Sequential: 3-5              [1, 58, 28, 28]           7,598
├─Sequential: 1-4                        [1, 232, 14, 14]          --
│    └─InvertedResidual: 2-8             [1, 232, 14, 14]          --
│    │    └─Sequential: 3-6              [1, 116, 14, 14]          14,964
│    │    └─Sequential: 3-7              [1, 116, 14, 14]          28,652
│    └─InvertedResidual: 2-9             [1, 232, 14, 14]          --
│    │    └─Sequential: 3-8              [1, 116, 14, 14]          28,652
│    └─InvertedResidual: 2-10            [1, 232, 14, 14]          --
│    │    └─Sequential: 3-9              [1, 116, 14, 14]          28,652
│    └─InvertedResidual: 2-11            [1, 232, 14, 14]          --
│    │    └─Sequential: 3-10             [1, 116, 14, 14]          28,652
│    └─InvertedResidual: 2-12            [1, 232, 14, 14]          --
│    │    └─Sequential: 3-11             [1, 116, 14, 14]          28,652
│    └─InvertedResidual: 2-13            [1, 232, 14, 14]          --
│    │    └─Sequential: 3-12             [1, 116, 14, 14]          28,652
│    └─InvertedResidual: 2-14            [1, 232, 14, 14]          --
│    │    └─Sequential: 3-13             [1, 116, 14, 14]          28,652
│    └─InvertedResidual: 2-15            [1, 232, 14, 14]          --
│    │    └─Sequential: 3-14             [1, 116, 14, 14]          28,652
├─Sequential: 1-5                        [1, 464, 7, 7]            --
│    └─InvertedResidual: 2-16            [1, 464, 7, 7]            --
│    │    └─Sequential: 3-15             [1, 232, 7, 7]            56,840
│    │    └─Sequential: 3-16             [1, 232, 7, 7]            111,128
│    └─InvertedResidual: 2-17            [1, 464, 7, 7]            --
│    │    └─Sequential: 3-17             [1, 232, 7, 7]            111,128
│    └─InvertedResidual: 2-18            [1, 464, 7, 7]            --
│    │    └─Sequential: 3-18             [1, 232, 7, 7]            111,128
│    └─InvertedResidual: 2-19            [1, 464, 7, 7]            --
│    │    └─Sequential: 3-19             [1, 232, 7, 7]            111,128
├─Sequential: 1-6                        [1, 1024, 7, 7]           --
│    └─Conv2d: 2-20                      [1, 1024, 7, 7]           475,136
│    └─BatchNorm2d: 2-21                 [1, 1024, 7, 7]           2,048
│    └─ReLU: 2-22                        [1, 1024, 7, 7]           --
├─Linear: 1-7                            [1, 5]                    5,125
==========================================================================================
Total params: 1,258,729
Trainable params: 1,258,729
Non-trainable params: 0
Total mult-adds (Units.MEGABYTES): 143.91
==========================================================================================
Input size (MB): 0.60
Forward/backward pass size (MB): 31.20
Params size (MB): 5.03
Estimated Total Size (MB): 36.84
==========================================================================================
```

In [19]:

```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 classification_report, confusion_matrix, roc_auc_score, precision_recall_curve, auc,
import time
import numpy as np
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)}

data_transforms = {
```

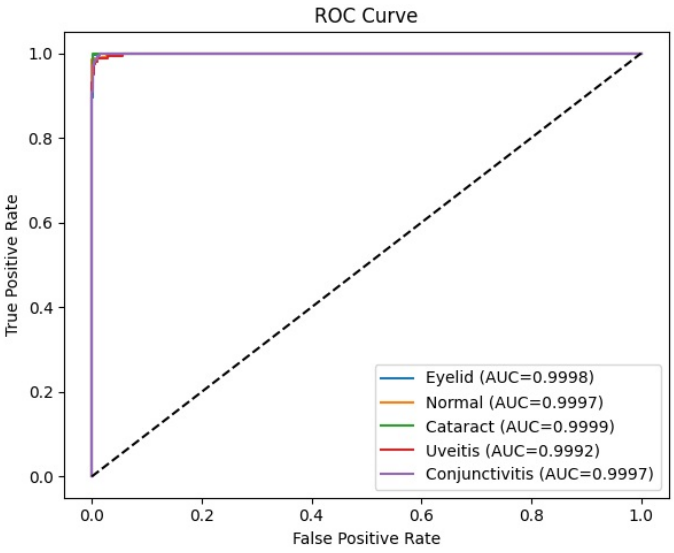
```
    'train': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],
                             [0.229, 0.224, 0.225])
    ]),
    'val': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),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

def create_dataset_loader(split_name, split_data):
    image_paths, labels = [], []
    for cls in classes:
        paths = split_data[split_name][cls]
        image_paths.extend(paths)
        labels.extend([class_to_idx[cls]]*len(paths))
    dataset = EyeDataset(image_paths, labels, transform=data_transforms[split_name])
    loader = DataLoader(dataset, batch_size=32, shuffle=(split_name=='train'), num_workers=2)
    return dataset, loader

# Create data loaders
train_dataset, train_loader = create_dataset_loader('train', split_data)
val_dataset, val_loader = create_dataset_loader('val', split_data)
test_dataset, test_loader = create_dataset_loader('test', split_data)

# Load pre-trained SqueezeNet
weights = SqueezeNet1_1_Weights.IMAGENET1K_V1
model = squeezenet1_1(weights=weights)

# Modify classifier for your num_classes
model.classifier[1] = nn.Conv2d(512, num_classes, kernel_size=(1,1))
model.num_classes = num_classes
model = model.to(device)

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

patience = 5
best_val_loss = float('inf')
epochs_no_improve = 0
num_epochs = 30

def train_epoch(model, dataloader):
    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)
```

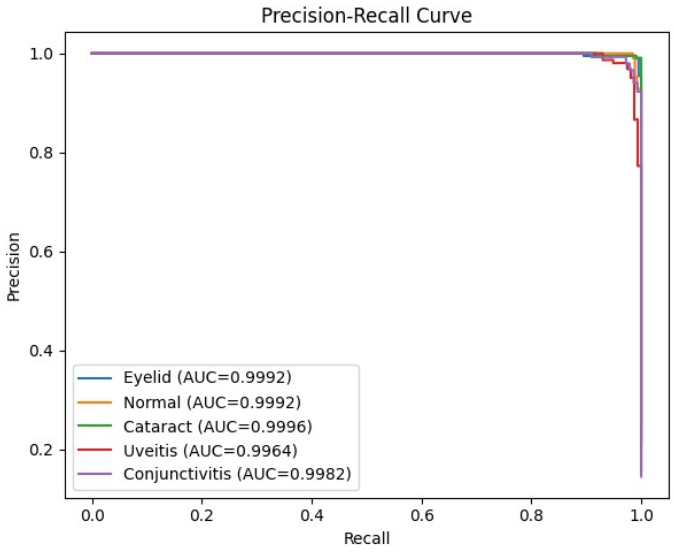
```
    return running_loss / total, correct / total

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)

train_start_time = time.time()
for epoch in range(num_epochs):
    train_loss, train_acc = train_epoch(model, train_loader)
    val_loss, val_acc, _, _, _ = eval_model(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: {val_lo
    if val_loss < best_val_loss:
        best_val_loss = val_loss
        epochs_no_improve = 0
        torch.save(model.state_dict(), "best_squeezenet.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")

model.load_state_dict(torch.load("best_squeezenet.pth"))

def print_classification_reports(model, loader, split_name):
    _, _, labels, preds, probs = eval_model(model, loader)
    print(f"\nClassification report for {split_name}:")
    print(classification_report(labels, preds, target_names=classes, digits=4))

print_classification_reports(model, train_loader, 'Train')
print_classification_reports(model, val_loader, 'Validation')

test_loss, test_acc, test_labels, test_preds, test_probs = eval_model(model, test_loader)
print(f"\nTest Accuracy: {test_acc:.4f}")
print("\nConfusion Matrix:")
print(confusion_matrix(test_labels, test_preds))
print("\nClassification Report (Test):")
print(classification_report(test_labels, test_preds, target_names=classes, digits=4))

plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
for i in range(num_classes):
    fpr, tpr, _ = roc_curve(test_labels == i, test_probs[:, i])
    plt.plot(fpr, tpr, label=f"{classes[i]} (AUC={roc_auc_score(test_labels==i, test_probs[:,i]):.4f})")
plt.plot([0,1], [0,1], 'k--')
plt.title("ROC Curve")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()

plt.subplot(1,2,2)
for i in range(num_classes):
    precision, recall, _ = precision_recall_curve(test_labels == i, test_probs[:, i])
    pr_auc = auc(recall, precision)
    plt.plot(recall, precision, label=f"{classes[i]} (AUC={pr_auc:.4f})")
plt.title("Precision-Recall Curve")
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.legend()
plt.tight_layout()
plt.show()
```

```
start_infer = time.time()
_ = eval_model(model, test_loader)
end_infer = time.time()
total_infer_time = end_infer - start_infer
print(f"Total inference time on test set: {total_infer_time:.4f} seconds")
print(f"Inference time per sample: {total_infer_time/len(test_dataset):.6f} seconds")
```

```
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, 62.5MB/s]
```

```
Epoch 1/30 | Train Loss: 0.8400 Acc: 0.6647 | Val Loss: 0.4727 Acc: 0.8374
Epoch 2/30 | Train Loss: 0.4414 Acc: 0.8359 | Val Loss: 0.4813 Acc: 0.8211
Epoch 3/30 | Train Loss: 0.3230 Acc: 0.8842 | Val Loss: 0.2909 Acc: 0.8974
Epoch 4/30 | Train Loss: 0.2313 Acc: 0.9182 | Val Loss: 0.2243 Acc: 0.9085
Epoch 5/30 | Train Loss: 0.1824 Acc: 0.9363 | Val Loss: 0.2781 Acc: 0.9045
Epoch 6/30 | Train Loss: 0.1511 Acc: 0.9472 | Val Loss: 0.1903 Acc: 0.9319
Epoch 7/30 | Train Loss: 0.1174 Acc: 0.9592 | Val Loss: 0.1856 Acc: 0.9329
Epoch 8/30 | Train Loss: 0.0929 Acc: 0.9699 | Val Loss: 0.1738 Acc: 0.9380
Epoch 9/30 | Train Loss: 0.0812 Acc: 0.9715 | Val Loss: 0.1328 Acc: 0.9482
Epoch 10/30 | Train Loss: 0.0676 Acc: 0.9776 | Val Loss: 0.0869 Acc: 0.9665
Epoch 11/30 | Train Loss: 0.0542 Acc: 0.9817 | Val Loss: 0.1108 Acc: 0.9573
Epoch 12/30 | Train Loss: 0.0539 Acc: 0.9822 | Val Loss: 0.0962 Acc: 0.9654
Epoch 13/30 | Train Loss: 0.0545 Acc: 0.9810 | Val Loss: 0.1063 Acc: 0.9654
Epoch 14/30 | Train Loss: 0.0360 Acc: 0.9878 | Val Loss: 0.1350 Acc: 0.9522
Epoch 15/30 | Train Loss: 0.0125 Acc: 0.9976 | Val Loss: 0.0505 Acc: 0.9858
Epoch 16/30 | Train Loss: 0.0070 Acc: 0.9989 | Val Loss: 0.0499 Acc: 0.9858
Epoch 17/30 | Train Loss: 0.0051 Acc: 0.9997 | Val Loss: 0.0448 Acc: 0.9827
Epoch 18/30 | Train Loss: 0.0043 Acc: 0.9996 | Val Loss: 0.0489 Acc: 0.9868
Epoch 19/30 | Train Loss: 0.0036 Acc: 0.9997 | Val Loss: 0.0513 Acc: 0.9827
Epoch 20/30 | Train Loss: 0.0034 Acc: 0.9999 | Val Loss: 0.0459 Acc: 0.9848
Epoch 21/30 | Train Loss: 0.0032 Acc: 0.9997 | Val Loss: 0.0450 Acc: 0.9848
Epoch 22/30 | Train Loss: 0.0027 Acc: 0.9999 | Val Loss: 0.0471 Acc: 0.9848
Early stopping triggered
Training time: 286.83 seconds

Classification report for Train:
                precision    recall  f1-score   support

        Eyelid     1.0000    1.0000    1.0000      1680
        Normal     1.0000    0.9995    0.9998      2076
      Cataract     0.9994    1.0000    0.9997      1740
       Uveitis     1.0000    0.9992    0.9996      1248
Conjunctivitis     0.9991    1.0000    0.9996      1142

      accuracy                         0.9997      7886
     macro avg     0.9997    0.9997    0.9997      7886
  weighted avg     0.9997    0.9997    0.9997      7886


Classification report for Validation:
                precision    recall  f1-score   support

        Eyelid     0.9856    0.9810    0.9833       210
        Normal     0.9961    0.9961    0.9961       259
      Cataract     0.9908    0.9954    0.9931       217
       Uveitis     0.9557    0.9679    0.9618       156
Conjunctivitis     0.9714    0.9577    0.9645       142

      accuracy                         0.9827       984
     macro avg     0.9799    0.9796    0.9798       984
  weighted avg     0.9828    0.9827    0.9827       984


Test Accuracy: 0.9859

Confusion Matrix:
[[208   0   0   2   0]
 [  0 257   2   1   1]
 [  0   0 219   0   0]
 [  2   0   0 153   2]
 [  1   0   0   3 140]]

Classification Report (Test):
                precision    recall  f1-score   support

        Eyelid     0.9858    0.9905    0.9881       210
        Normal     1.0000    0.9847    0.9923       261
      Cataract     0.9910    1.0000    0.9955       219
       Uveitis     0.9623    0.9745    0.9684       157
Conjunctivitis     0.9790    0.9722    0.9756       144

      accuracy                         0.9859       991
     macro avg     0.9836    0.9844    0.9840       991

  weighted avg     0.9860    0.9859    0.9859       991
```






```
Total inference time on test set: 1.3838 seconds
Inference time per sample: 0.001396 seconds
```

In [20]:
```python
from sklearn.metrics import cohen_kappa_score
from statsmodels.stats.proportion import proportion_confint

# Compute Cohen's Kappa
kappa = cohen_kappa_score(test_labels, test_preds)
print(f"Cohen's Kappa: {kappa:.4f}")

# Compute 95% Confidence Interval for Accuracy
correct_preds = np.sum(test_labels == test_preds)
n = len(test_labels)
acc_lower, acc_upper = proportion_confint(count=correct_preds, nobs=n, alpha=0.05, method='wilson')
print(f"Test Accuracy: {test_acc:.4f}")
print(f"95% Confidence Interval for Accuracy: [{acc_lower:.4f}, {acc_upper:.4f}]")
```

```
Cohen's Kappa: 0.9821
Test Accuracy: 0.9859
95% Confidence Interval for Accuracy: [0.9764, 0.9916]
```

In [21]:
```python
from torchinfo import summary
import torch
from torchvision.models import squeezenet1_1, SqueezeNet1_1_Weights
import torch.nn as nn

num_classes = 5

# Load pre-trained SqueezeNet
weights = SqueezeNet1_1_Weights.IMAGENET1K_V1
model = squeezenet1_1(weights=weights)

# Modify classifier to match number of classes
model.classifier[1] = nn.Conv2d(512, num_classes, kernel_size=(1,1))
model.num_classes = num_classes

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

# Print summary
summary(model, input_size=(1, 3, 224, 224))
```

Out[21]:

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
SqueezeNet                               [1, 5]                    --
├─Sequential: 1-1                        [1, 512, 13, 13]          --
│    └─Conv2d: 2-1                       [1, 64, 111, 111]         1,792
│    └─ReLU: 2-2                         [1, 64, 111, 111]         --
│    └─MaxPool2d: 2-3                    [1, 64, 55, 55]           --
│    └─Fire: 2-4                         [1, 128, 55, 55]          --
│    │    └─Conv2d: 3-1                  [1, 16, 55, 55]           1,040
│    │    └─ReLU: 3-2                    [1, 16, 55, 55]           --
│    │    └─Conv2d: 3-3                  [1, 64, 55, 55]           1,088
│    │    └─ReLU: 3-4                    [1, 64, 55, 55]           --
│    │    └─Conv2d: 3-5                  [1, 64, 55, 55]           9,280
│    │    └─ReLU: 3-6                    [1, 64, 55, 55]           --
│    └─Fire: 2-5                         [1, 128, 55, 55]          --
│    │    └─Conv2d: 3-7                  [1, 16, 55, 55]           2,064
│    │    └─ReLU: 3-8                    [1, 16, 55, 55]           --
│    │    └─Conv2d: 3-9                  [1, 64, 55, 55]           1,088
│    │    └─ReLU: 3-10                   [1, 64, 55, 55]           --
│    │    └─Conv2d: 3-11                 [1, 64, 55, 55]           9,280
│    │    └─ReLU: 3-12                   [1, 64, 55, 55]           --
│    └─MaxPool2d: 2-6                    [1, 128, 27, 27]          --
│    └─Fire: 2-7                         [1, 256, 27, 27]          --
│    │    └─Conv2d: 3-13                 [1, 32, 27, 27]           4,128
│    │    └─ReLU: 3-14                   [1, 32, 27, 27]           --
│    │    └─Conv2d: 3-15                 [1, 128, 27, 27]          4,224
│    │    └─ReLU: 3-16                   [1, 128, 27, 27]          --
│    │    └─Conv2d: 3-17                 [1, 128, 27, 27]          36,992
│    │    └─ReLU: 3-18                   [1, 128, 27, 27]          --
│    └─Fire: 2-8                         [1, 256, 27, 27]          --
│    │    └─Conv2d: 3-19                 [1, 32, 27, 27]           8,224
│    │    └─ReLU: 3-20                   [1, 32, 27, 27]           --
│    │    └─Conv2d: 3-21                 [1, 128, 27, 27]          4,224
│    │    └─ReLU: 3-22                   [1, 128, 27, 27]          --
│    │    └─Conv2d: 3-23                 [1, 128, 27, 27]          36,992
│    │    └─ReLU: 3-24                   [1, 128, 27, 27]          --
│    └─MaxPool2d: 2-9                    [1, 256, 13, 13]          --
│    └─Fire: 2-10                        [1, 384, 13, 13]          --
│    │    └─Conv2d: 3-25                 [1, 48, 13, 13]           12,336
│    │    └─ReLU: 3-26                   [1, 48, 13, 13]           --
│    │    └─Conv2d: 3-27                 [1, 192, 13, 13]          9,408
│    │    └─ReLU: 3-28                   [1, 192, 13, 13]          --
│    │    └─Conv2d: 3-29                 [1, 192, 13, 13]          83,136
│    │    └─ReLU: 3-30                   [1, 192, 13, 13]          --
│    └─Fire: 2-11                        [1, 384, 13, 13]          --
│    │    └─Conv2d: 3-31                 [1, 48, 13, 13]           18,480
│    │    └─ReLU: 3-32                   [1, 48, 13, 13]           --
│    │    └─Conv2d: 3-33                 [1, 192, 13, 13]          9,408
│    │    └─ReLU: 3-34                   [1, 192, 13, 13]          --
│    │    └─Conv2d: 3-35                 [1, 192, 13, 13]          83,136
│    │    └─ReLU: 3-36                   [1, 192, 13, 13]          --
│    └─Fire: 2-12                        [1, 512, 13, 13]          --
│    │    └─Conv2d: 3-37                 [1, 64, 13, 13]           24,640
│    │    └─ReLU: 3-38                   [1, 64, 13, 13]           --
│    │    └─Conv2d: 3-39                 [1, 256, 13, 13]          16,640
│    │    └─ReLU: 3-40                   [1, 256, 13, 13]          --
│    │    └─Conv2d: 3-41                 [1, 256, 13, 13]          147,712
│    │    └─ReLU: 3-42                   [1, 256, 13, 13]          --
│    └─Fire: 2-13                        [1, 512, 13, 13]          --
│    │    └─Conv2d: 3-43                 [1, 64, 13, 13]           32,832
│    │    └─ReLU: 3-44                   [1, 64, 13, 13]           --
│    │    └─Conv2d: 3-45                 [1, 256, 13, 13]          16,640
│    │    └─ReLU: 3-46                   [1, 256, 13, 13]          --
│    │    └─Conv2d: 3-47                 [1, 256, 13, 13]          147,712
│    │    └─ReLU: 3-48                   [1, 256, 13, 13]          --
├─Sequential: 1-2                        [1, 5, 1, 1]              --
│    └─Dropout: 2-14                     [1, 512, 13, 13]          --
│    └─Conv2d: 2-15                      [1, 5, 13, 13]            2,565
│    └─ReLU: 2-16                        [1, 5, 13, 13]            --
│    └─AdaptiveAvgPool2d: 2-17           [1, 5, 1, 1]              --
==========================================================================================
Total params: 725,061
Trainable params: 725,061
Non-trainable params: 0
Total mult-adds (Units.MEGABYTES): 265.48
==========================================================================================
Input size (MB): 0.60
Forward/backward pass size (MB): 19.37
Params size (MB): 2.90
Estimated Total Size (MB): 22.87
==========================================================================================
```

PROPOSE

In [23]:

```
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 classification_report, confusion_matrix, roc_auc_score, precision_recall_curve, auc,
import time
import numpy as np
import matplotlib.pyplot as plt

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

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

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

def create_dataset_loader(split_name, split_data):
    image_paths, labels = [], []
    for cls in classes:
        paths = split_data[split_name][cls]
        image_paths.extend(paths)
        labels.extend([class_to_idx[cls]]*len(paths))
    dataset = EyeDataset(image_paths, labels, transform=data_transforms[split_name])
    loader = DataLoader(dataset, batch_size=32, shuffle=(split_name=='train'), num_workers=2)
    return dataset, loader

# Define your split_data dictionary before this point:
# Example:
# split_data = {
#   'train': {'Eyelid': [...], 'Normal': [...], ...},
#   'val': {...},
#   'test': {...}
# }

train_dataset, train_loader = create_dataset_loader('train', split_data)
val_dataset, val_loader = create_dataset_loader('val', split_data)
test_dataset, test_loader = create_dataset_loader('test', split_data)

# Load pretrained models
shufflenet_weights = ShuffleNet_V2_X1_0_Weights.IMAGENET1K_V1
mobilenet_weights = MobileNet_V2_Weights.IMAGENET1K_V1
```

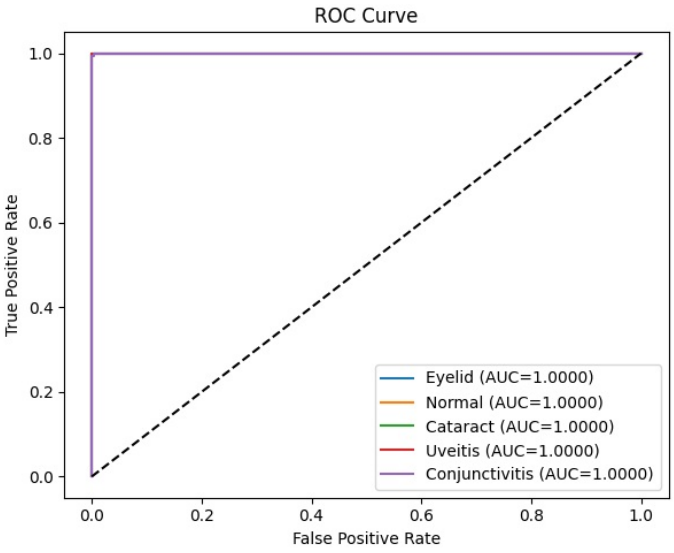
```
shufflenet_base = shufflenet_v2_x1_0(weights=shufflenet_weights)
mobilenet_base = mobilenet_v2(weights=mobilenet_weights)

# Feature extractor modules that output feature vectors
class ShuffleNetFeatureExtractor(nn.Module):
    def __init__(self, model):
        super().__init__()
        # All layers except final fc
        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

shufflenet_feat = ShuffleNetFeatureExtractor(shufflenet_base).to(device)
mobilenet_feat = MobileNetFeatureExtractor(mobilenet_base).to(device)

# Check feature dims
with torch.no_grad():
    dummy = torch.randn(1,3,224,224).to(device)
    sn_feat_dim = shufflenet_feat(dummy).shape[1]
    mn_feat_dim = mobilenet_feat(dummy).shape[1]

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

# Fusion classifier with batch norm on each feature before concatenation
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

fusion_model = FusionModel(sn_feat_dim, mn_feat_dim, num_classes).to(device)

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

patience = 5
best_val_loss = float('inf')
epochs_no_improve = 0
num_epochs = 30

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

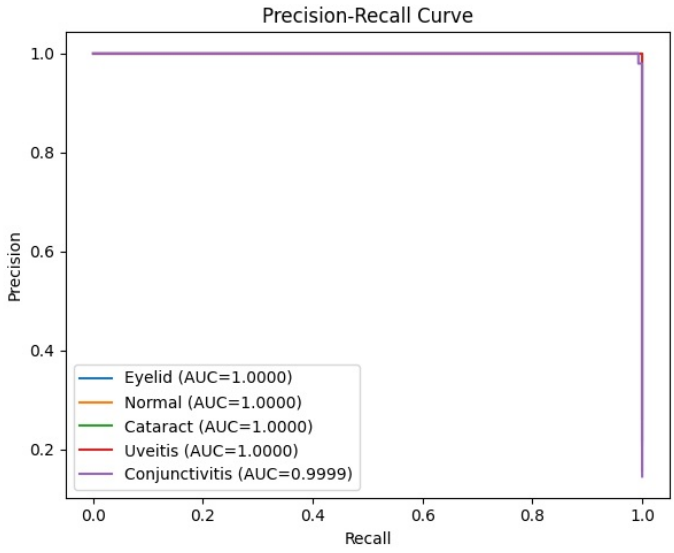
```
        feat_sn = sn_feat_extractor(inputs)
        feat_mn = mn_feat_extractor(inputs)
        outputs = model((feat_sn, feat_mn))
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item() * inputs.size(0)
        _, preds = torch.max(outputs,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

def eval_epoch(model, sn_feat_extractor, mn_feat_extractor, loader):
    model.eval()
    sn_feat_extractor.eval()
    mn_feat_extractor.eval()
    running_loss = 0.0
    correct = 0
    total = 0
    all_labels = []
    all_preds = []
    all_probs = []
    with torch.no_grad():
        for inputs, labels in loader:
            inputs, labels = inputs.to(device), labels.to(device)
            feat_sn = sn_feat_extractor(inputs)
            feat_mn = mn_feat_extractor(inputs)
            outputs = model((feat_sn, feat_mn))
            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)

# Training loop
train_start_time = time.time()
for epoch in range(num_epochs):
    train_loss, train_acc = train_epoch(fusion_model, shufflenet_feat, mobilenet_feat, train_loader)
    val_loss, val_acc, _, _, _ = eval_epoch(fusion_model, shufflenet_feat, mobilenet_feat, val_loader)
    scheduler.step(val_loss)
    print(f"Epoch {epoch+1}/{num_epochs} | Train Loss: {train_loss:.4f} Acc: {train_acc:.4f} | Val Loss: {val_lo
    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()
        }, "best_fusion.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("best_fusion.pth")
fusion_model.load_state_dict(checkpoint['fusion'])
shufflenet_feat.load_state_dict(checkpoint['sn_feat'])
mobilenet_feat.load_state_dict(checkpoint['mn_feat'])

# Print classification reports
def print_classification_reports(model, sn_feat_extractor, mn_feat_extractor, loader, split_name):
    _, _, labels, preds, probs = eval_epoch(model, sn_feat_extractor, mn_feat_extractor, loader)
    print(f"\nClassification report for {split_name}:")
    print(classification_report(labels, preds, target_names=classes, digits=4))

print_classification_reports(fusion_model, shufflenet_feat, mobilenet_feat, train_loader, 'Train')
print_classification_reports(fusion_model, shufflenet_feat, mobilenet_feat, val_loader, 'Validation')

# Test evaluation
test_loss, test_acc, test_labels, test_preds, test_probs = eval_epoch(fusion_model, shufflenet_feat, mobilenet_f
print(f"\nTest Accuracy: {test_acc:.4f}")
print("\nConfusion Matrix:")
print(confusion_matrix(test_labels, test_preds))
print("\nClassification Report (Test):")
```

```python
print(classification_report(test_labels, test_preds, target_names=classes, digits=4))

# Plot ROC and PR curves
plt.figure(figsize=(12,5))

plt.subplot(1,2,1)
for i in range(num_classes):
    fpr, tpr, _ = roc_curve(test_labels == i, test_probs[:, i])
    plt.plot(fpr, tpr, label=f"{classes[i]} (AUC={roc_auc_score(test_labels==i, test_probs[:,i]):.4f})")
plt.plot([0,1], [0,1], 'k--')
plt.title("ROC Curve")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()

plt.subplot(1,2,2)
for i in range(num_classes):
    precision, recall, _ = precision_recall_curve(test_labels == i, test_probs[:, i])
    pr_auc = auc(recall, precision)
    plt.plot(recall, precision, label=f"{classes[i]} (AUC={pr_auc:.4f})")
plt.title("Precision-Recall Curve")
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.legend()

plt.tight_layout()
plt.show()

# Inference timing
start_infer = time.time()
_ = eval_epoch(fusion_model, shufflenet_feat, mobilenet_feat, test_loader)
end_infer = time.time()
total_infer_time = end_infer - start_infer
inference_per_sample = total_infer_time / len(test_dataset)
print(f"Total inference time on test set: {total_infer_time:.4f} seconds")
print(f"Inference time per sample: {inference_per_sample:.6f} seconds")
```

```
ShuffleNet feature dim: 1024, MobileNetV2 feature dim: 1280
Epoch 1/30 | Train Loss: 0.4389 Acc: 0.8438 | Val Loss: 0.1488 Acc: 0.9492
Epoch 2/30 | Train Loss: 0.1093 Acc: 0.9655 | Val Loss: 0.0732 Acc: 0.9756
Epoch 3/30 | Train Loss: 0.0503 Acc: 0.9834 | Val Loss: 0.0621 Acc: 0.9837
Epoch 4/30 | Train Loss: 0.0353 Acc: 0.9891 | Val Loss: 0.0587 Acc: 0.9817
Epoch 5/30 | Train Loss: 0.0312 Acc: 0.9881 | Val Loss: 0.0654 Acc: 0.9827
Epoch 6/30 | Train Loss: 0.0237 Acc: 0.9934 | Val Loss: 0.0506 Acc: 0.9848
Epoch 7/30 | Train Loss: 0.0180 Acc: 0.9942 | Val Loss: 0.0867 Acc: 0.9807
Epoch 8/30 | Train Loss: 0.0222 Acc: 0.9924 | Val Loss: 0.0717 Acc: 0.9776
Epoch 9/30 | Train Loss: 0.0214 Acc: 0.9934 | Val Loss: 0.0646 Acc: 0.9837
Epoch 10/30 | Train Loss: 0.0271 Acc: 0.9907 | Val Loss: 0.0503 Acc: 0.9888
Epoch 11/30 | Train Loss: 0.0087 Acc: 0.9981 | Val Loss: 0.0372 Acc: 0.9878
Epoch 12/30 | Train Loss: 0.0171 Acc: 0.9942 | Val Loss: 0.0583 Acc: 0.9797
Epoch 13/30 | Train Loss: 0.0172 Acc: 0.9939 | Val Loss: 0.0648 Acc: 0.9807
Epoch 14/30 | Train Loss: 0.0110 Acc: 0.9962 | Val Loss: 0.0359 Acc: 0.9888
Epoch 15/30 | Train Loss: 0.0273 Acc: 0.9909 | Val Loss: 0.0560 Acc: 0.9858
Epoch 16/30 | Train Loss: 0.0143 Acc: 0.9953 | Val Loss: 0.0529 Acc: 0.9878
Epoch 17/30 | Train Loss: 0.0202 Acc: 0.9940 | Val Loss: 0.0567 Acc: 0.9848
Epoch 18/30 | Train Loss: 0.0102 Acc: 0.9963 | Val Loss: 0.0456 Acc: 0.9868
Epoch 19/30 | Train Loss: 0.0066 Acc: 0.9976 | Val Loss: 0.0256 Acc: 0.9909
Epoch 20/30 | Train Loss: 0.0031 Acc: 0.9992 | Val Loss: 0.0251 Acc: 0.9919
Epoch 21/30 | Train Loss: 0.0018 Acc: 0.9997 | Val Loss: 0.0210 Acc: 0.9919
Epoch 22/30 | Train Loss: 0.0015 Acc: 0.9996 | Val Loss: 0.0236 Acc: 0.9919
Epoch 23/30 | Train Loss: 0.0020 Acc: 0.9995 | Val Loss: 0.0244 Acc: 0.9929
Epoch 24/30 | Train Loss: 0.0013 Acc: 0.9997 | Val Loss: 0.0214 Acc: 0.9929
Epoch 25/30 | Train Loss: 0.0008 Acc: 0.9999 | Val Loss: 0.0172 Acc: 0.9939
Epoch 26/30 | Train Loss: 0.0010 Acc: 0.9997 | Val Loss: 0.0175 Acc: 0.9939
Epoch 27/30 | Train Loss: 0.0013 Acc: 0.9996 | Val Loss: 0.0250 Acc: 0.9929
Epoch 28/30 | Train Loss: 0.0011 Acc: 0.9996 | Val Loss: 0.0217 Acc: 0.9939
Epoch 29/30 | Train Loss: 0.0021 Acc: 0.9996 | Val Loss: 0.0213 Acc: 0.9939
Epoch 30/30 | Train Loss: 0.0008 Acc: 0.9996 | Val Loss: 0.0202 Acc: 0.9949
Early stopping triggered
Training time: 1018.31 seconds

Classification report for Train:
                precision    recall  f1-score   support

       Eyelid     1.0000    1.0000    1.0000      1680
       Normal     1.0000    1.0000    1.0000      2076
     Cataract     1.0000    1.0000    1.0000      1740
      Uveitis     1.0000    1.0000    1.0000      1248
Conjunctivitis     1.0000    1.0000    1.0000      1142

     accuracy                         1.0000      7886
    macro avg     1.0000    1.0000    1.0000      7886
 weighted avg     1.0000    1.0000    1.0000      7886
```

```
Classification report for Validation:
               precision    recall  f1-score   support

       Eyelid     1.0000    0.9952    0.9976       210
       Normal     1.0000    1.0000    1.0000       259
     Cataract     0.9954    0.9954    0.9954       217
      Uveitis     0.9689    1.0000    0.9842       156
Conjunctivitis     1.0000    0.9718    0.9857       142

     accuracy                         0.9939       984
    macro avg     0.9929    0.9925    0.9926       984
 weighted avg     0.9941    0.9939    0.9939       984


Test Accuracy: 0.9970

Confusion Matrix:
[[210   0   0   0   0]
 [  0 261   0   0   0]
 [  0   0 219   0   0]
 [  0   0   0 157   0]
 [  1   0   2   0 141]]

Classification Report (Test):
               precision    recall  f1-score   support

       Eyelid     0.9953    1.0000    0.9976       210
       Normal     1.0000    1.0000    1.0000       261
     Cataract     0.9910    1.0000    0.9955       219
      Uveitis     1.0000    1.0000    1.0000       157
Conjunctivitis     1.0000    0.9792    0.9895       144

     accuracy                         0.9970       991
    macro avg     0.9972    0.9958    0.9965       991
 weighted avg     0.9970    0.9970    0.9970       991
```






```
Total inference time on test set: 1.6198 seconds
Inference time per sample: 0.001634 seconds
```

In [25]:
```python
from sklearn.metrics import cohen_kappa_score
from statsmodels.stats.proportion import proportion_confint

# Compute Cohen's Kappa
kappa = cohen_kappa_score(test_labels, test_preds)
print(f"Cohen's Kappa: {kappa:.4f}")

# Compute 95% Confidence Interval for Accuracy
correct_preds = np.sum(test_labels == test_preds)
n = len(test_labels)
acc_lower, acc_upper = proportion_confint(count=correct_preds, nobs=n, alpha=0.05, method='wilson')
print(f"Test Accuracy: {test_acc:.4f}")
print(f"95% Confidence Interval for Accuracy: [{acc_lower:.4f}, {acc_upper:.4f}]")
```

```
Cohen's Kappa: 0.9962
Test Accuracy: 0.9970
95% Confidence Interval for Accuracy: [0.9911, 0.9990]
```

In [28]:
```python
def print_model_summary():
    print("=== ShuffleNetV2 Feature Extractor ===")
    print(shufflenet_feat)
    sn_params = sum(p.numel() for p in shufflenet_feat.parameters())
```

```
    print(f"Total parameters in ShuffleNetV2 feature extractor: {sn_params:,}\n")

    print("=== MobileNetV2 Feature Extractor ===")
    print(mobilenet_feat)
    mn_params = sum(p.numel() for p in mobilenet_feat.parameters())
    print(f"Total parameters in MobileNetV2 feature extractor: {mn_params:,}\n")

    print("=== Fusion Classifier ===")
    print(fusion_model)
    fusion_params = sum(p.numel() for p in fusion_model.parameters())
    print(f"Total parameters in Fusion classifier: {fusion_params:,}\n")

    total_params = sn_params + mn_params + fusion_params
    print(f"=== Total parameters in the whole fusion model: {total_params:,} ===")

print_model_summary()
```

```
=== ShuffleNetV2 Feature Extractor ===
ShuffleNetFeatureExtractor(
  (features): Sequential(
    (0): Sequential(
      (0): Conv2d(3, 24, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
      (1): BatchNorm2d(24, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU(inplace=True)
    )
    (1): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
    (2): Sequential(
      (0): InvertedResidual(
        (branch1): Sequential(
          (0): Conv2d(24, 24, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=24, bias=False)
          (1): BatchNorm2d(24, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): Conv2d(24, 58, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (3): BatchNorm2d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (4): ReLU(inplace=True)
        )
        (branch2): Sequential(
          (0): Conv2d(24, 58, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU(inplace=True)
          (3): Conv2d(58, 58, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=58, bias=False)
          (4): BatchNorm2d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (5): Conv2d(58, 58, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (6): BatchNorm2d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (7): ReLU(inplace=True)
        )
      )
      (1): InvertedResidual(
        (branch1): Sequential()
        (branch2): Sequential(
          (0): Conv2d(58, 58, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU(inplace=True)
          (3): Conv2d(58, 58, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=58, bias=False)
          (4): BatchNorm2d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (5): Conv2d(58, 58, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (6): BatchNorm2d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (7): ReLU(inplace=True)
        )
      )
      (2): InvertedResidual(
        (branch1): Sequential()
        (branch2): Sequential(
          (0): Conv2d(58, 58, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU(inplace=True)
          (3): Conv2d(58, 58, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=58, bias=False)
          (4): BatchNorm2d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (5): Conv2d(58, 58, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (6): BatchNorm2d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (7): ReLU(inplace=True)
        )
      )
      (3): InvertedResidual(
        (branch1): Sequential()
        (branch2): Sequential(
          (0): Conv2d(58, 58, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU(inplace=True)
          (3): Conv2d(58, 58, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=58, bias=False)
          (4): BatchNorm2d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (5): Conv2d(58, 58, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (6): BatchNorm2d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (7): ReLU(inplace=True)
        )
```

```
    )
  )
  (3): Sequential(
    (0): InvertedResidual(
      (branch1): Sequential(
        (0): Conv2d(116, 116, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=116, bias=False)
        (1): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (3): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (4): ReLU(inplace=True)
      )
      (branch2): Sequential(
        (0): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU(inplace=True)
        (3): Conv2d(116, 116, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=116, bias=False)
        (4): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (5): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (6): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (7): ReLU(inplace=True)
      )
    )
    (1): InvertedResidual(
      (branch1): Sequential()
      (branch2): Sequential(
        (0): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU(inplace=True)
        (3): Conv2d(116, 116, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=116, bias=False)
        (4): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (5): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (6): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (7): ReLU(inplace=True)
      )
    )
    (2): InvertedResidual(
      (branch1): Sequential()
      (branch2): Sequential(
        (0): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU(inplace=True)
        (3): Conv2d(116, 116, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=116, bias=False)
        (4): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (5): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (6): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (7): ReLU(inplace=True)
      )
    )
    (3): InvertedResidual(
      (branch1): Sequential()
      (branch2): Sequential(
        (0): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU(inplace=True)
        (3): Conv2d(116, 116, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=116, bias=False)
        (4): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (5): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (6): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (7): ReLU(inplace=True)
      )
    )
    (4): InvertedResidual(
      (branch1): Sequential()
      (branch2): Sequential(
        (0): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU(inplace=True)
        (3): Conv2d(116, 116, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=116, bias=False)
        (4): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (5): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (6): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (7): ReLU(inplace=True)
      )
    )
    (5): InvertedResidual(
      (branch1): Sequential()
      (branch2): Sequential(
        (0): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU(inplace=True)
        (3): Conv2d(116, 116, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=116, bias=False)
        (4): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (5): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
```

```
      (6): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (7): ReLU(inplace=True)
    )
  )
  (6): InvertedResidual(
    (branch1): Sequential()
    (branch2): Sequential(
      (0): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU(inplace=True)
      (3): Conv2d(116, 116, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=116, bias=False)
      (4): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (5): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (6): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (7): ReLU(inplace=True)
    )
  )
  (7): InvertedResidual(
    (branch1): Sequential()
    (branch2): Sequential(
      (0): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU(inplace=True)
      (3): Conv2d(116, 116, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=116, bias=False)
      (4): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (5): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (6): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (7): ReLU(inplace=True)
    )
  )
)
(4): Sequential(
  (0): InvertedResidual(
    (branch1): Sequential(
      (0): Conv2d(232, 232, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=232, bias=False)
      (1): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): Conv2d(232, 232, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (3): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (4): ReLU(inplace=True)
    )
    (branch2): Sequential(
      (0): Conv2d(232, 232, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU(inplace=True)
      (3): Conv2d(232, 232, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=232, bias=False)
      (4): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (5): Conv2d(232, 232, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (6): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (7): ReLU(inplace=True)
    )
  )
  (1): InvertedResidual(
    (branch1): Sequential()
    (branch2): Sequential(
      (0): Conv2d(232, 232, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU(inplace=True)
      (3): Conv2d(232, 232, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=232, bias=False)
      (4): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (5): Conv2d(232, 232, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (6): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (7): ReLU(inplace=True)
    )
  )
  (2): InvertedResidual(
    (branch1): Sequential()
    (branch2): Sequential(
      (0): Conv2d(232, 232, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU(inplace=True)
      (3): Conv2d(232, 232, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=232, bias=False)
      (4): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (5): Conv2d(232, 232, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (6): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (7): ReLU(inplace=True)
    )
  )
  (3): InvertedResidual(
    (branch1): Sequential()
    (branch2): Sequential(
      (0): Conv2d(232, 232, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU(inplace=True)
```

```
          (3): Conv2d(232, 232, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=232, bias=False)
          (4): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (5): Conv2d(232, 232, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (6): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (7): ReLU(inplace=True)
        )
      )
    )
    (5): Sequential(
      (0): Conv2d(464, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU(inplace=True)
    )
  )
  (pool): AdaptiveAvgPool2d(output_size=1)
  (flatten): Flatten(start_dim=1, end_dim=-1)
)
Total parameters in ShuffleNetV2 feature extractor: 1,253,604

=== MobileNetV2 Feature Extractor ===
MobileNetFeatureExtractor(
  (features): Sequential(
    (0): Conv2dNormActivation(
      (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
      (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6(inplace=True)
    )
    (1): InvertedResidual(
      (conv): Sequential(
        (0): Conv2dNormActivation(
          (0): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=32, bias=False)
          (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU6(inplace=True)
        )
        (1): Conv2d(32, 16, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (2): InvertedResidual(
      (conv): Sequential(
        (0): Conv2dNormActivation(
          (0): Conv2d(16, 96, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU6(inplace=True)
        )
        (1): Conv2dNormActivation(
          (0): Conv2d(96, 96, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=96, bias=False)
          (1): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU6(inplace=True)
        )
        (2): Conv2d(96, 24, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (3): BatchNorm2d(24, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (3): InvertedResidual(
      (conv): Sequential(
        (0): Conv2dNormActivation(
          (0): Conv2d(24, 144, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(144, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU6(inplace=True)
        )
        (1): Conv2dNormActivation(
          (0): Conv2d(144, 144, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=144, bias=False)
          (1): BatchNorm2d(144, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU6(inplace=True)
        )
        (2): Conv2d(144, 24, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (3): BatchNorm2d(24, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (4): InvertedResidual(
      (conv): Sequential(
        (0): Conv2dNormActivation(
          (0): Conv2d(24, 144, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(144, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU6(inplace=True)
        )
        (1): Conv2dNormActivation(
          (0): Conv2d(144, 144, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=144, bias=False)
          (1): BatchNorm2d(144, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU6(inplace=True)
        )
        (2): Conv2d(144, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
```

```
    (3): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(5): InvertedResidual(
  (conv): Sequential(
    (0): Conv2dNormActivation(
      (0): Conv2d(32, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6(inplace=True)
    )
    (1): Conv2dNormActivation(
      (0): Conv2d(192, 192, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=192, bias=False)
      (1): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6(inplace=True)
    )
    (2): Conv2d(192, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (3): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(6): InvertedResidual(
  (conv): Sequential(
    (0): Conv2dNormActivation(
      (0): Conv2d(32, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6(inplace=True)
    )
    (1): Conv2dNormActivation(
      (0): Conv2d(192, 192, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=192, bias=False)
      (1): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6(inplace=True)
    )
    (2): Conv2d(192, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (3): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(7): InvertedResidual(
  (conv): Sequential(
    (0): Conv2dNormActivation(
      (0): Conv2d(32, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6(inplace=True)
    )
    (1): Conv2dNormActivation(
      (0): Conv2d(192, 192, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=192, bias=False)
      (1): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6(inplace=True)
    )
    (2): Conv2d(192, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (3): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(8): InvertedResidual(
  (conv): Sequential(
    (0): Conv2dNormActivation(
      (0): Conv2d(64, 384, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(384, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6(inplace=True)
    )
    (1): Conv2dNormActivation(
      (0): Conv2d(384, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=384, bias=False)
      (1): BatchNorm2d(384, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6(inplace=True)
    )
    (2): Conv2d(384, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (3): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(9): InvertedResidual(
  (conv): Sequential(
    (0): Conv2dNormActivation(
      (0): Conv2d(64, 384, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(384, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6(inplace=True)
    )
    (1): Conv2dNormActivation(
      (0): Conv2d(384, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=384, bias=False)
      (1): BatchNorm2d(384, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6(inplace=True)
    )
    (2): Conv2d(384, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (3): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
)
```

```
(10): InvertedResidual(
  (conv): Sequential(
    (0): Conv2dNormActivation(
      (0): Conv2d(64, 384, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(384, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6(inplace=True)
    )
    (1): Conv2dNormActivation(
      (0): Conv2d(384, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=384, bias=False)
      (1): BatchNorm2d(384, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6(inplace=True)
    )
    (2): Conv2d(384, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (3): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(11): InvertedResidual(
  (conv): Sequential(
    (0): Conv2dNormActivation(
      (0): Conv2d(64, 384, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(384, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6(inplace=True)
    )
    (1): Conv2dNormActivation(
      (0): Conv2d(384, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=384, bias=False)
      (1): BatchNorm2d(384, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6(inplace=True)
    )
    (2): Conv2d(384, 96, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (3): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(12): InvertedResidual(
  (conv): Sequential(
    (0): Conv2dNormActivation(
      (0): Conv2d(96, 576, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(576, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6(inplace=True)
    )
    (1): Conv2dNormActivation(
      (0): Conv2d(576, 576, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=576, bias=False)
      (1): BatchNorm2d(576, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6(inplace=True)
    )
    (2): Conv2d(576, 96, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (3): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(13): InvertedResidual(
  (conv): Sequential(
    (0): Conv2dNormActivation(
      (0): Conv2d(96, 576, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(576, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6(inplace=True)
    )
    (1): Conv2dNormActivation(
      (0): Conv2d(576, 576, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=576, bias=False)
      (1): BatchNorm2d(576, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6(inplace=True)
    )
    (2): Conv2d(576, 96, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (3): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(14): InvertedResidual(
  (conv): Sequential(
    (0): Conv2dNormActivation(
      (0): Conv2d(96, 576, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(576, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6(inplace=True)
    )
    (1): Conv2dNormActivation(
      (0): Conv2d(576, 576, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=576, bias=False)
      (1): BatchNorm2d(576, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6(inplace=True)
    )
    (2): Conv2d(576, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (3): BatchNorm2d(160, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(15): InvertedResidual(
  (conv): Sequential(
    (0): Conv2dNormActivation(
```

```
          (0): Conv2d(160, 960, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(960, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU6(inplace=True)
        )
        (1): Conv2dNormActivation(
          (0): Conv2d(960, 960, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=960, bias=False)
          (1): BatchNorm2d(960, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU6(inplace=True)
        )
        (2): Conv2d(960, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (3): BatchNorm2d(160, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (16): InvertedResidual(
      (conv): Sequential(
        (0): Conv2dNormActivation(
          (0): Conv2d(160, 960, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(960, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU6(inplace=True)
        )
        (1): Conv2dNormActivation(
          (0): Conv2d(960, 960, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=960, bias=False)
          (1): BatchNorm2d(960, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU6(inplace=True)
        )
        (2): Conv2d(960, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (3): BatchNorm2d(160, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (17): InvertedResidual(
      (conv): Sequential(
        (0): Conv2dNormActivation(
          (0): Conv2d(160, 960, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(960, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU6(inplace=True)
        )
        (1): Conv2dNormActivation(
          (0): Conv2d(960, 960, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=960, bias=False)
          (1): BatchNorm2d(960, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU6(inplace=True)
        )
        (2): Conv2d(960, 320, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (3): BatchNorm2d(320, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (18): Conv2dNormActivation(
      (0): Conv2d(320, 1280, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(1280, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6(inplace=True)
    )
  )
  (pool): AdaptiveAvgPool2d(output_size=1)
  (flatten): Flatten(start_dim=1, end_dim=-1)
)
Total parameters in MobileNetV2 feature extractor: 2,223,872

=== Fusion Classifier ===
FusionModel(
  (bn1): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (bn2): BatchNorm1d(1280, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (classifier): Sequential(
    (0): Linear(in_features=2304, out_features=512, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.5, inplace=False)
    (3): Linear(in_features=512, out_features=5, bias=True)
  )
)
Total parameters in Fusion classifier: 1,187,333

=== Total parameters in the whole fusion model: 4,664,809 ===
```

In [27]:
```
torch.save(fusion_model.state_dict(), "fusion_model.pth")
print("Fusion model saved to fusion_model.pth")
```

```
Fusion model saved to fusion_model.pth
```

In [29]:
```
import torch.nn as nn

def get_last_conv_layer(model):
    last_conv = None
    for name, module in model.named_modules():
        if isinstance(module, nn.Conv2d):
            last_conv = (name, module)
    return last_conv
```

```
# For ShuffleNetV2
last_conv_sn = get_last_conv_layer(shufflenet_base)
print("Last conv layer in ShuffleNetV2:")
print(last_conv_sn[0])  # layer name
print(last_conv_sn[1])  # layer details

# For MobileNetV2
last_conv_mn = get_last_conv_layer(mobilenet_base)
print("\nLast conv layer in MobileNetV2:")
print(last_conv_mn[0])
print(last_conv_mn[1])
```

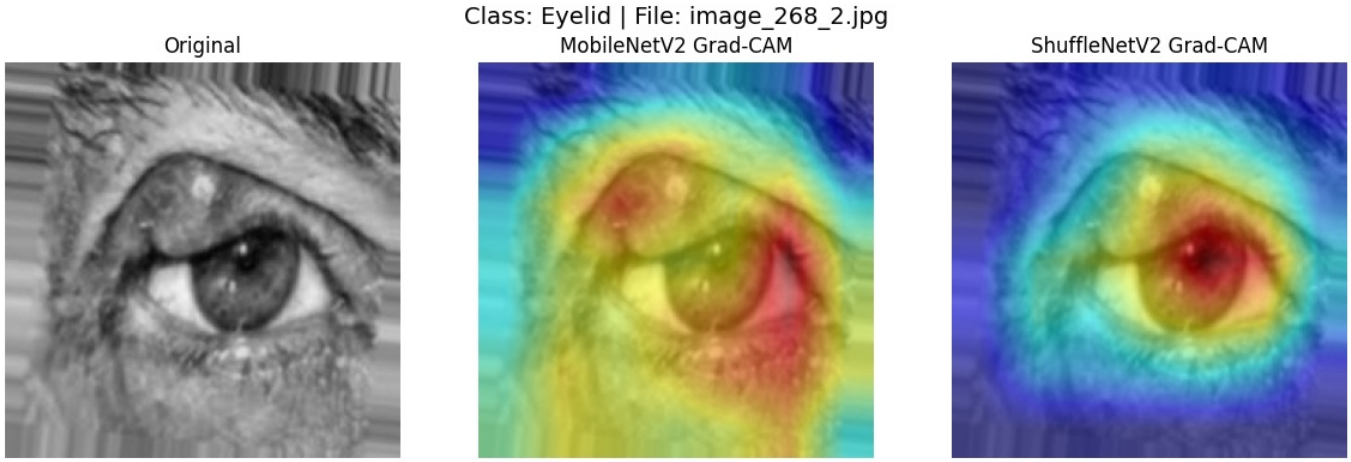
```
Last conv layer in ShuffleNetV2:
conv5.0
Conv2d(464, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)

Last conv layer in MobileNetV2:
features.18.0
Conv2d(320, 1280, kernel_size=(1, 1), stride=(1, 1), bias=False)
```

In [31]:

```
import torch
import torch.nn.functional as F
import torchvision.transforms as transforms
from torchvision.models import mobilenet_v2, shufflenet_v2_x1_0, MobileNet_V2_Weights, ShuffleNet_V2_X1_0_Weight
from PIL import Image
import matplotlib.pyplot as plt
import numpy as np
import cv2
import random
import os

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

# Load pretrained models
mobilenet = mobilenet_v2(weights=MobileNet_V2_Weights.DEFAULT).eval().to(device)
shufflenet = shufflenet_v2_x1_0(weights=ShuffleNet_V2_X1_0_Weights.DEFAULT).eval().to(device)

# Target layers for Grad-CAM
# Last conv layer of MobileNetV2 is features.18.0 (Conv2d)
mobilenet_target_layer = mobilenet.features[18][0]
shufflenet_target_layer = shufflenet.conv5[0]  # Final block of ShuffleNetV2

# Grad-CAM class
class GradCAM:
    def __init__(self, model, target_layer):
        self.model = model
        self.target_layer = target_layer
        self.gradients = None
        self.activations = None
        self._register_hooks()

    def _register_hooks(self):
        def forward_hook(module, input, output):
            self.activations = output.detach()
        def backward_hook(module, grad_input, grad_output):
            self.gradients = grad_output[0].detach()
        self.target_layer.register_forward_hook(forward_hook)
        self.target_layer.register_backward_hook(backward_hook)

    def generate(self, input_tensor, class_idx=None):
        output = self.model(input_tensor)
        if class_idx is None:
            class_idx = output.argmax(dim=1).item()
        self.model.zero_grad()
        output[0, class_idx].backward()

        pooled_grad = torch.mean(self.gradients, dim=(2, 3), keepdim=True)
        heatmap = torch.sum(self.activations * pooled_grad, dim=1).squeeze()
        heatmap = F.relu(heatmap)
        heatmap -= heatmap.min()
        heatmap /= (heatmap.max() + 1e-6)
        return heatmap.cpu().numpy()

# Overlay heatmap on original image
def overlay_heatmap(heatmap, img_pil, alpha=0.5):
    heatmap = cv2.resize(heatmap, img_pil.size)
    heatmap_color = cv2.applyColorMap(np.uint8(255 * heatmap), cv2.COLORMAP_JET)
    img_np = np.array(img_pil)[:, :, ::-1]
    overlay = cv2.addWeighted(img_np, 1 - alpha, heatmap_color, alpha, 0)
    return overlay[:, :, ::-1]  # Convert BGR to RGB

# Image preprocessing
transform = transforms.Compose([
```

```
    transforms.Resize((224, 224)),
    transforms.ToTensor()
])

# Make sure you have split_data defined like:
# split_data = {'test': {'Eyelid': [...], 'Normal': [...], ...}}

classes = list(split_data['test'].keys())
num_samples = 5  # Number of images per class to show

for cls in classes:
    files = split_data['test'][cls]
    selected = random.sample(files, min(len(files), num_samples))

    for img_path in selected:
        img_pil = Image.open(img_path).convert("RGB")
        input_tensor = transform(img_pil).unsqueeze(0).to(device)

        # Grad-CAM for both models
        heatmap_mn = GradCAM(mobilenet, mobilenet_target_layer).generate(input_tensor)
        heatmap_sn = GradCAM(shufflenet, shufflenet_target_layer).generate(input_tensor)

        overlay_mn = overlay_heatmap(heatmap_mn, img_pil)
        overlay_sn = overlay_heatmap(heatmap_sn, img_pil)

        # Plot results
        plt.figure(figsize=(12, 4))
        plt.suptitle(f"Class: {cls} | File: {os.path.basename(img_path)}", fontsize=14)

        plt.subplot(1, 3, 1)
        plt.imshow(img_pil)
        plt.title("Original")
        plt.axis("off")

        plt.subplot(1, 3, 2)
        plt.imshow(overlay_mn)
        plt.title("MobileNetV2 Grad-CAM")
        plt.axis("off")

        plt.subplot(1, 3, 3)
        plt.imshow(overlay_sn)
        plt.title("ShuffleNetV2 Grad-CAM")
        plt.axis("off")

        plt.tight_layout()
        plt.show()
```

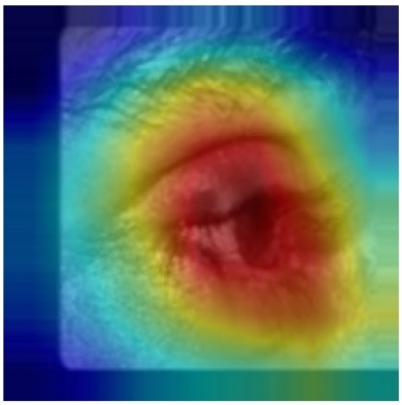
```
Downloading: "https://download.pytorch.org/models/mobilenet_v2-7ebf99e0.pth" to /root/.cache/torch/hub/checkpoin
ts/mobilenet_v2-7ebf99e0.pth
100%|██████████| 13.6M/13.6M [00:00<00:00, 112MB/s]
/usr/local/lib/python3.11/dist-packages/torch/nn/modules/module.py:1830: FutureWarning: Using a non-full backwar
d hook when the forward contains multiple autograd Nodes is deprecated and will be removed in future versions. T
his hook will be missing some grad_input. Please use register_full_backward_hook to get the documented behavior.
  self._maybe_warn_non_full_backward_hook(args, result, grad_fn)
```



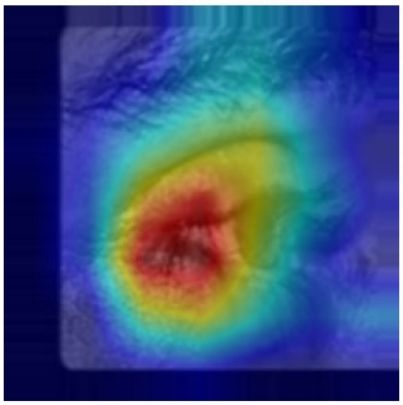
Class: Eyelid | File: image_116_2.jpg

Original

MobileNetV2 Grad-CAM



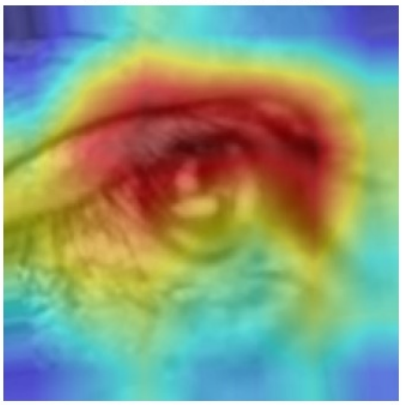
ShuffleNetV2 Grad-CAM



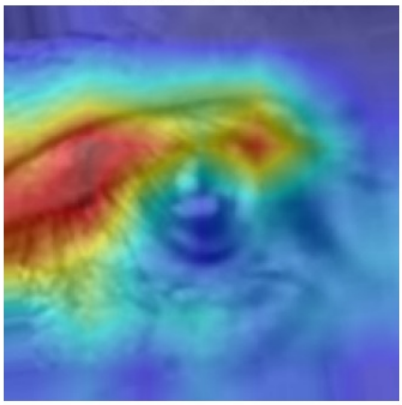
Class: Eyelid | File: image_167_1.jpg

Original

MobileNetV2 Grad-CAM



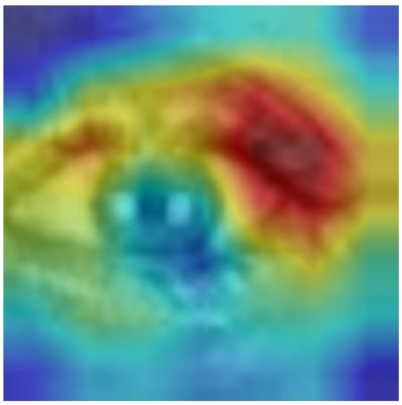
ShuffleNetV2 Grad-CAM



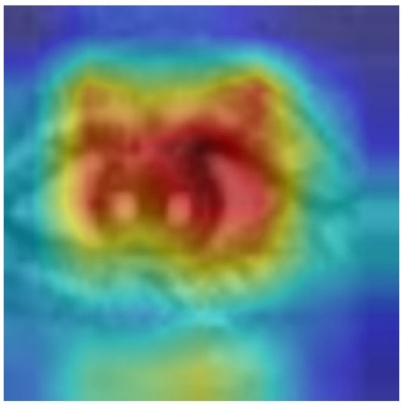
Class: Eyelid | File: image_41_2.jpg

Original

MobileNetV2 Grad-CAM



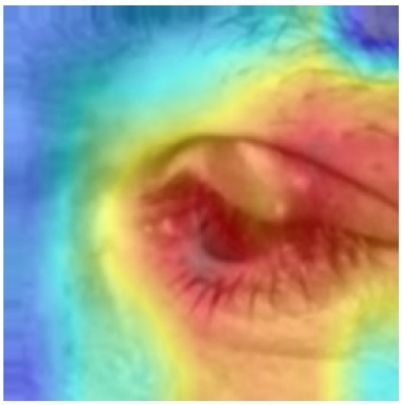
ShuffleNetV2 Grad-CAM



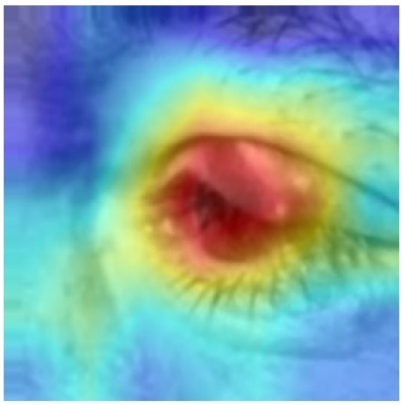
Class: Eyelid | File: image_439_2.jpg

Original

MobileNetV2 Grad-CAM



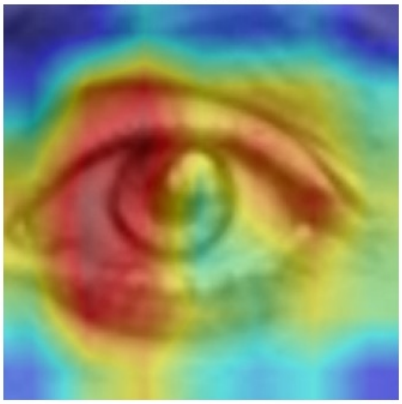
ShuffleNetV2 Grad-CAM



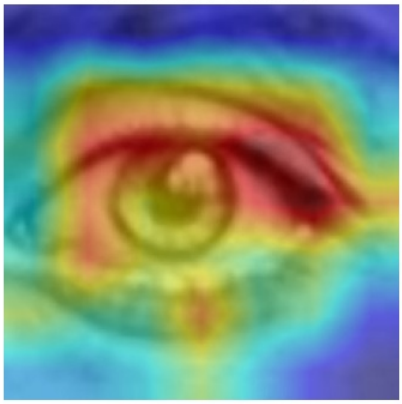
Class: Normal | File: image_111_2.jpg

Original

MobileNetV2 Grad-CAM



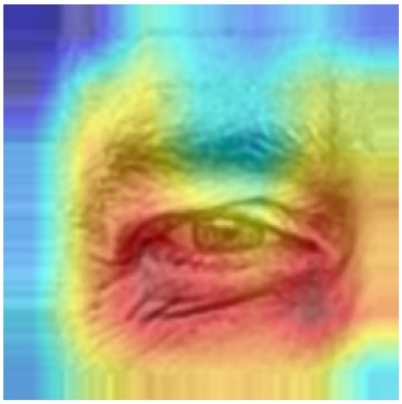
ShuffleNetV2 Grad-CAM



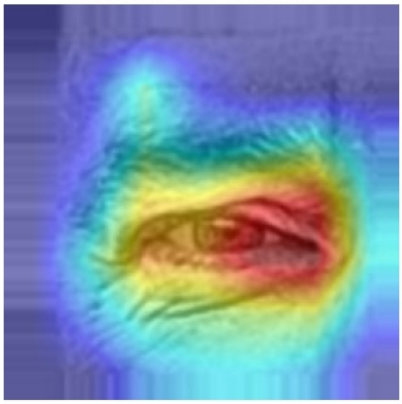
Class: Normal | File: image_559_2.jpg

Original

MobileNetV2 Grad-CAM



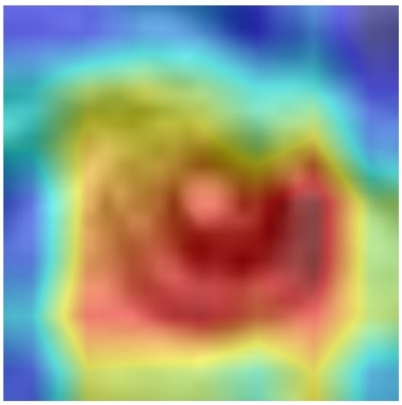
ShuffleNetV2 Grad-CAM



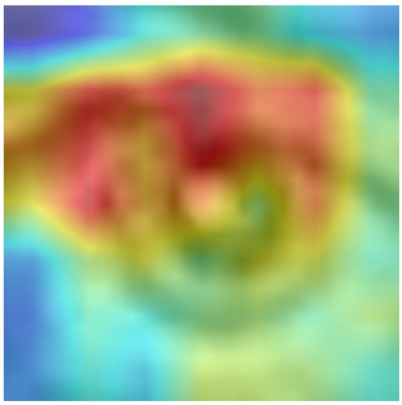
Class: Normal | File: image_186_0.jpg

Original

MobileNetV2 Grad-CAM



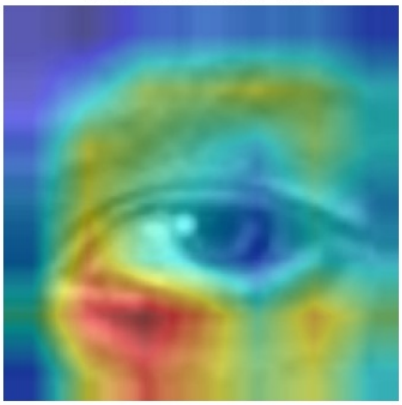
ShuffleNetV2 Grad-CAM



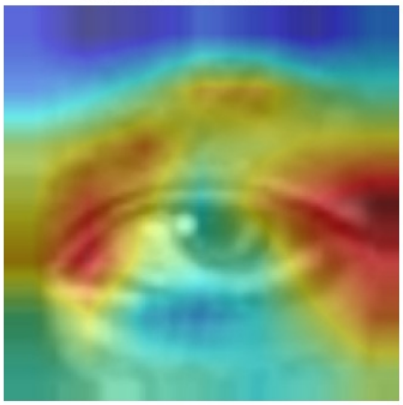
Class: Normal | File: image_248_3.jpg

Original

MobileNetV2 Grad-CAM



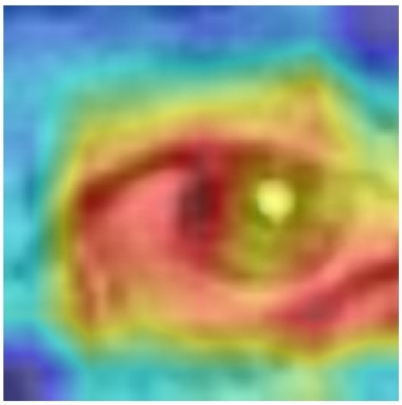
ShuffleNetV2 Grad-CAM



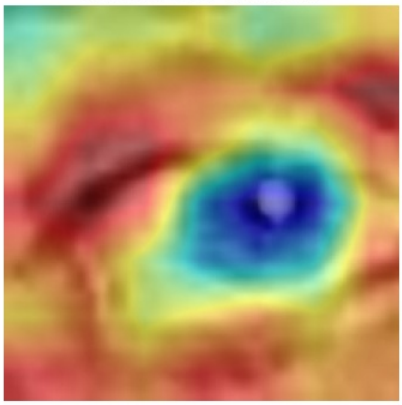
Class: Normal | File: image_550_0.jpg

Original

MobileNetV2 Grad-CAM

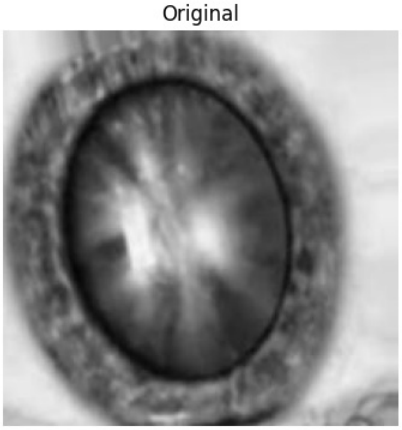


ShuffleNetV2 Grad-CAM

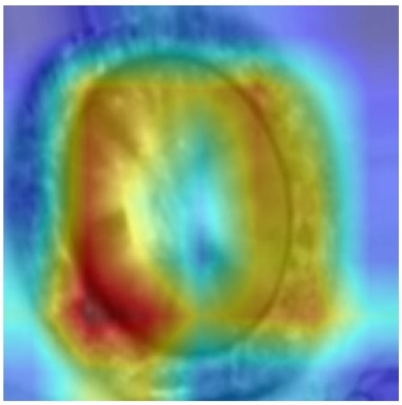


Class: Cataract | File: image_514_3.jpg

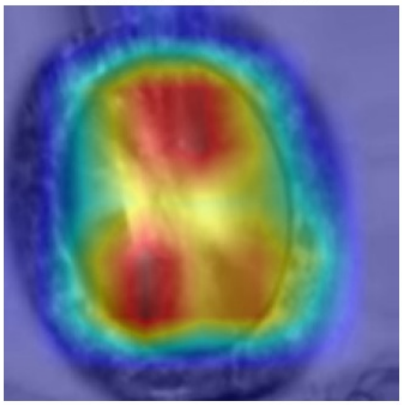
Original



MobileNetV2 Grad-CAM

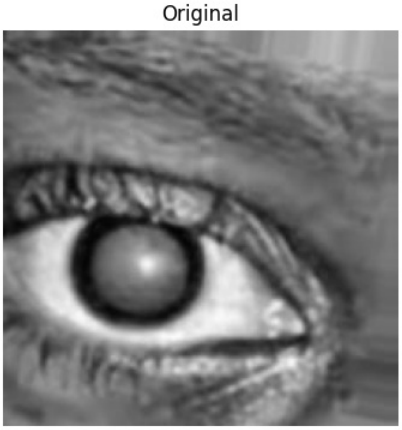


ShuffleNetV2 Grad-CAM

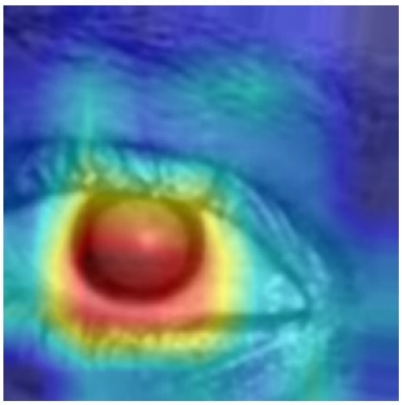


Class: Cataract | File: image_478_2.jpg

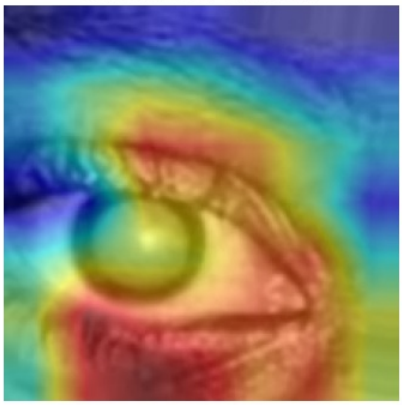
Original



MobileNetV2 Grad-CAM

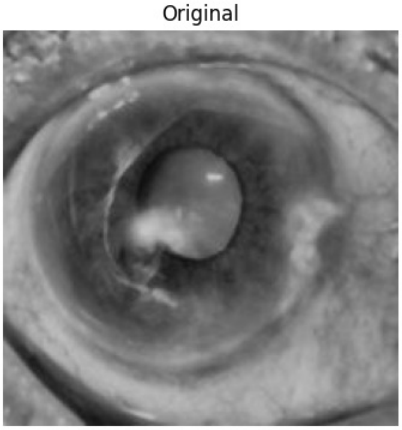


ShuffleNetV2 Grad-CAM

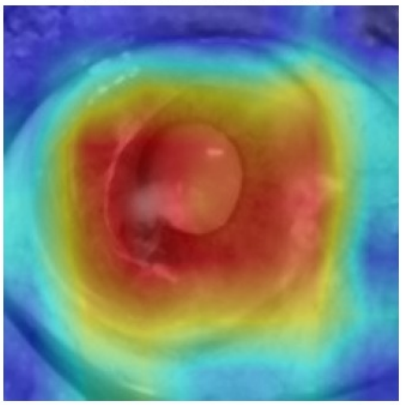


Class: Cataract | File: image_334_3.jpg

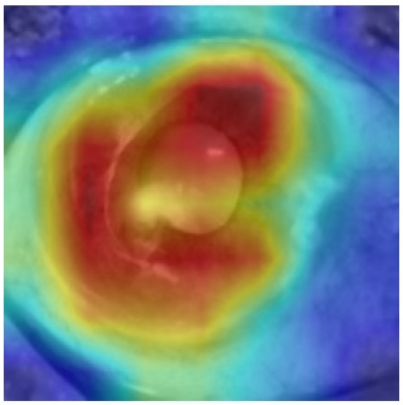
Original



MobileNetV2 Grad-CAM



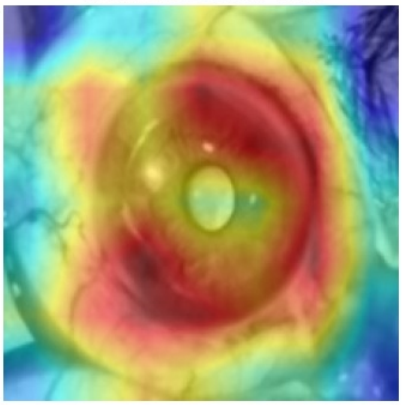
ShuffleNetV2 Grad-CAM



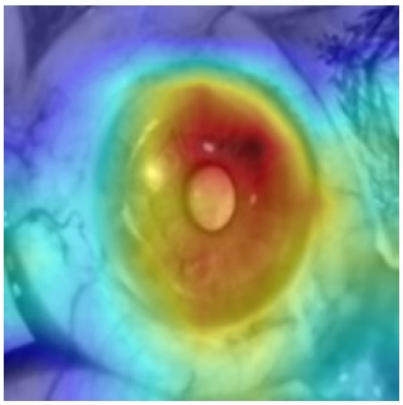
Class: Cataract | File: image_76_0.jpg

Original

MobileNetV2 Grad-CAM

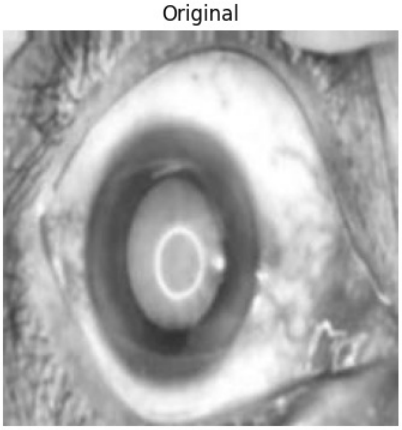


ShuffleNetV2 Grad-CAM

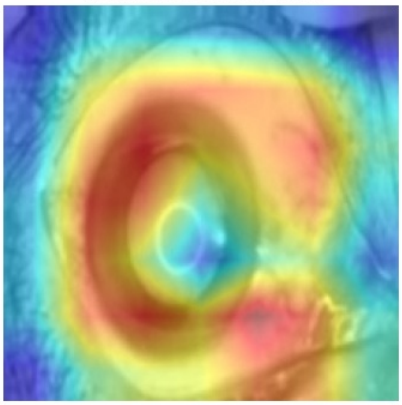


Class: Cataract | File: image_331_1.jpg

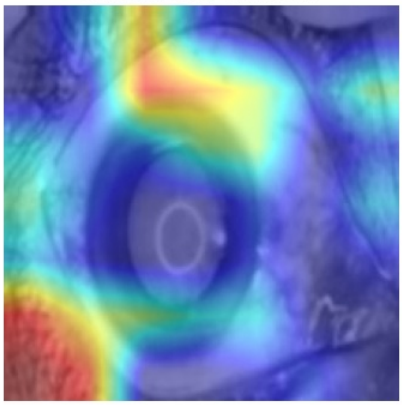
Original



MobileNetV2 Grad-CAM



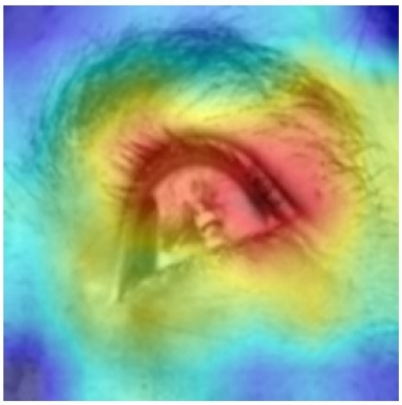
ShuffleNetV2 Grad-CAM



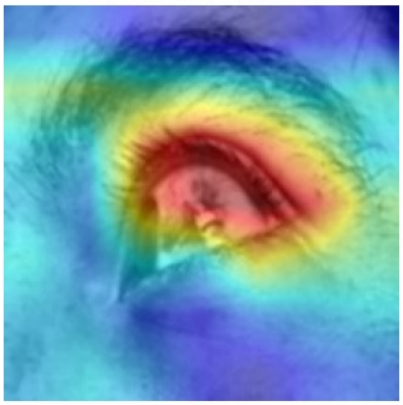
Class: Uveitis | File: image_208_6.jpg

Original

MobileNetV2 Grad-CAM



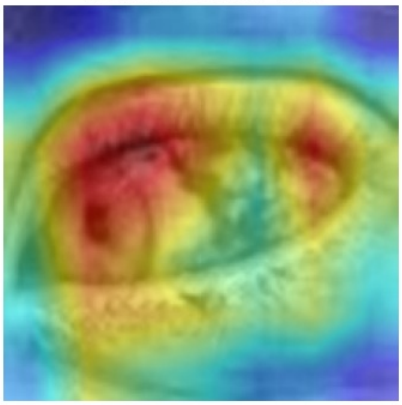
ShuffleNetV2 Grad-CAM



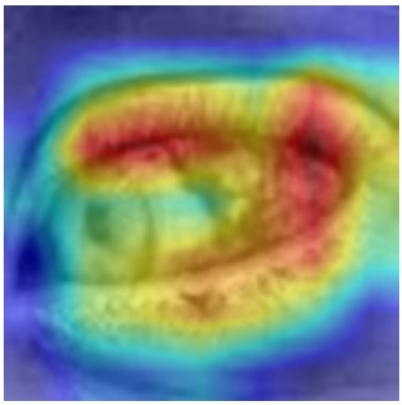
Class: Uveitis | File: image_106_0.jpg

Original

MobileNetV2 Grad-CAM

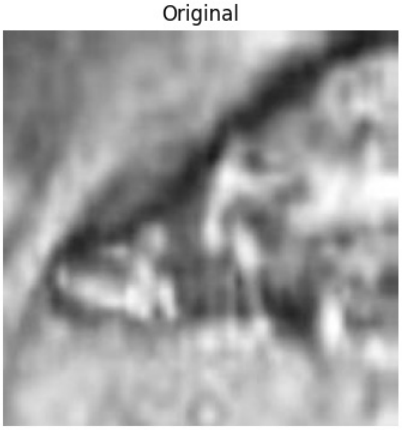


ShuffleNetV2 Grad-CAM

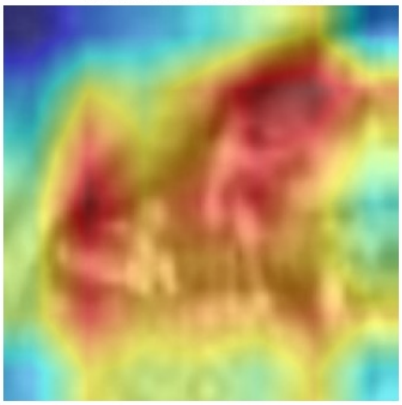


Class: Uveitis | File: image_13_5.jpg

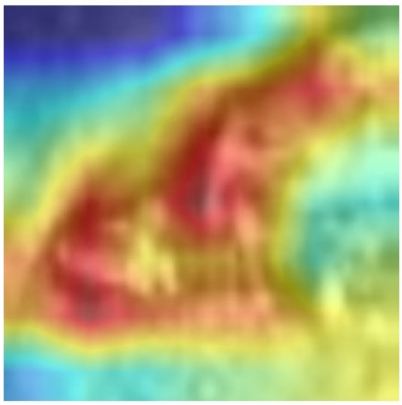
Original



MobileNetV2 Grad-CAM



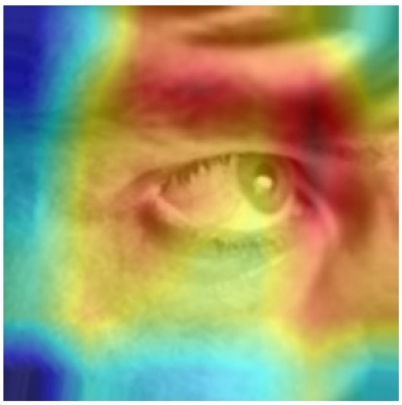
ShuffleNetV2 Grad-CAM



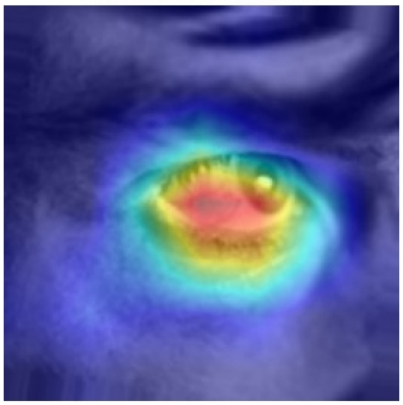
Class: Uveitis | File: image_66_6.jpg

Original

MobileNetV2 Grad-CAM



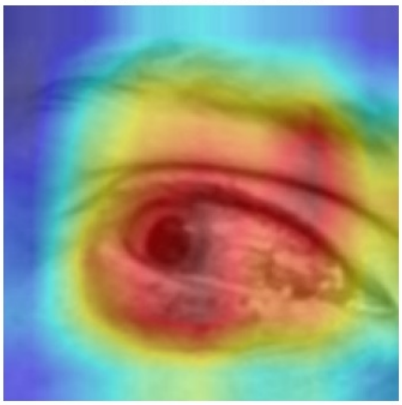
ShuffleNetV2 Grad-CAM



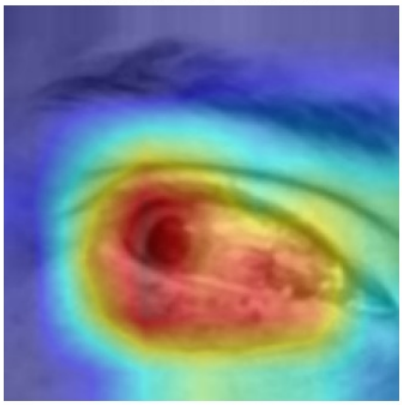
Class: Uveitis | File: image_101_4.jpg

Original

MobileNetV2 Grad-CAM

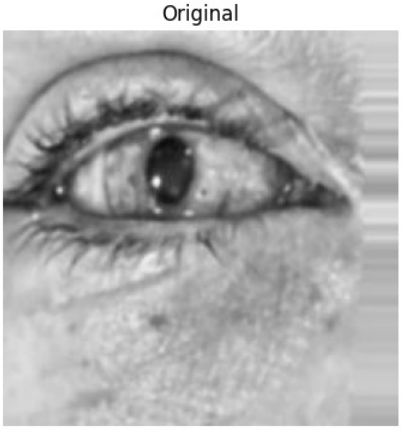


ShuffleNetV2 Grad-CAM

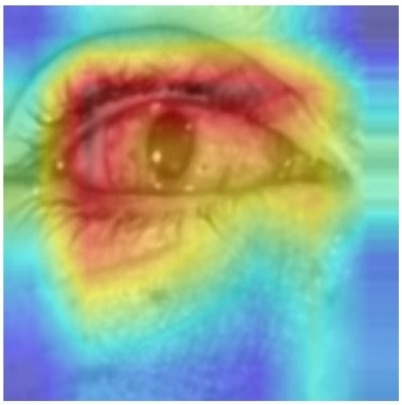


Class: Conjunctivitis | File: image_172_1.jpg

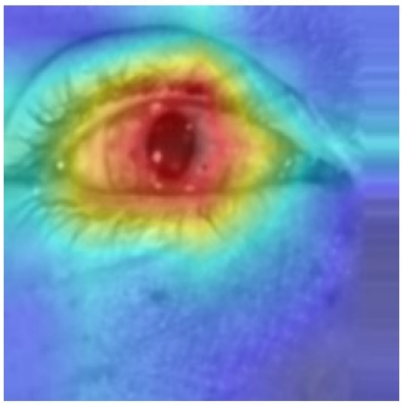
Original



MobileNetV2 Grad-CAM

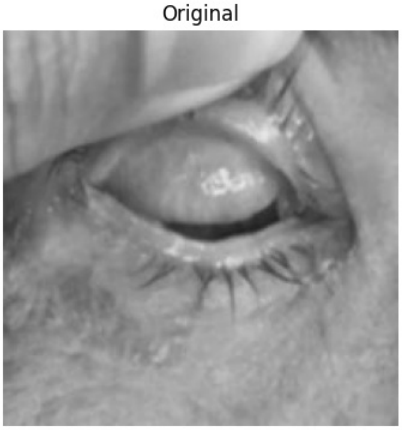


ShuffleNetV2 Grad-CAM

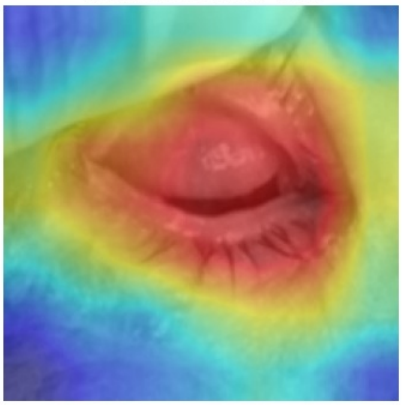


Class: Conjunctivitis | File: image_320_0.jpg

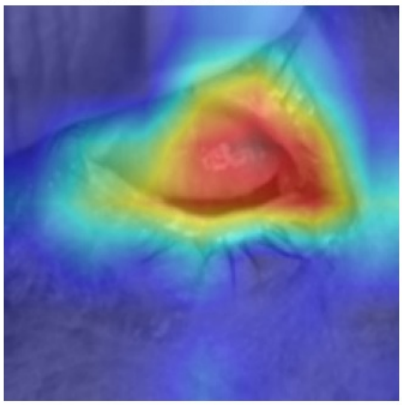
Original



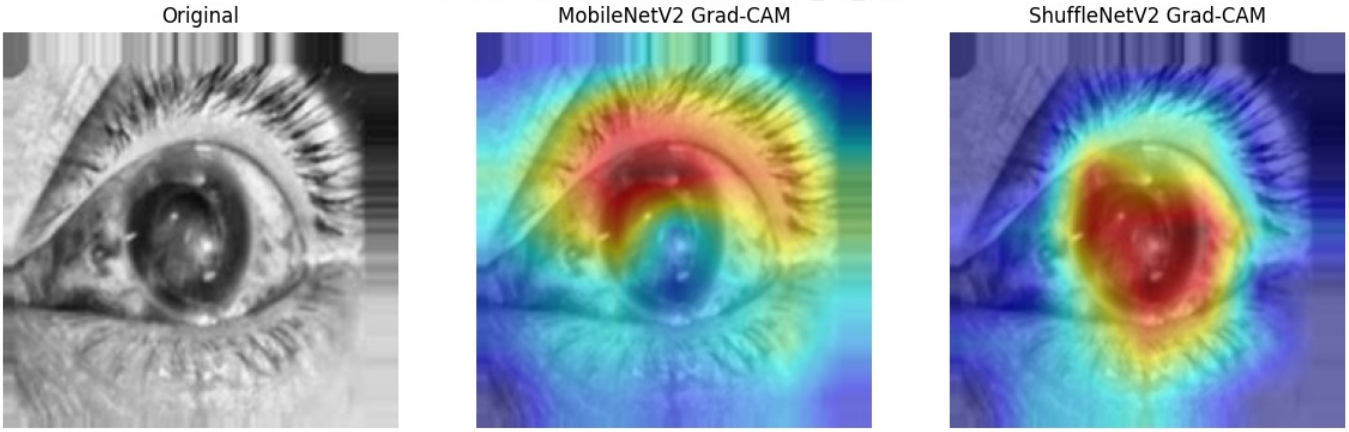
MobileNetV2 Grad-CAM



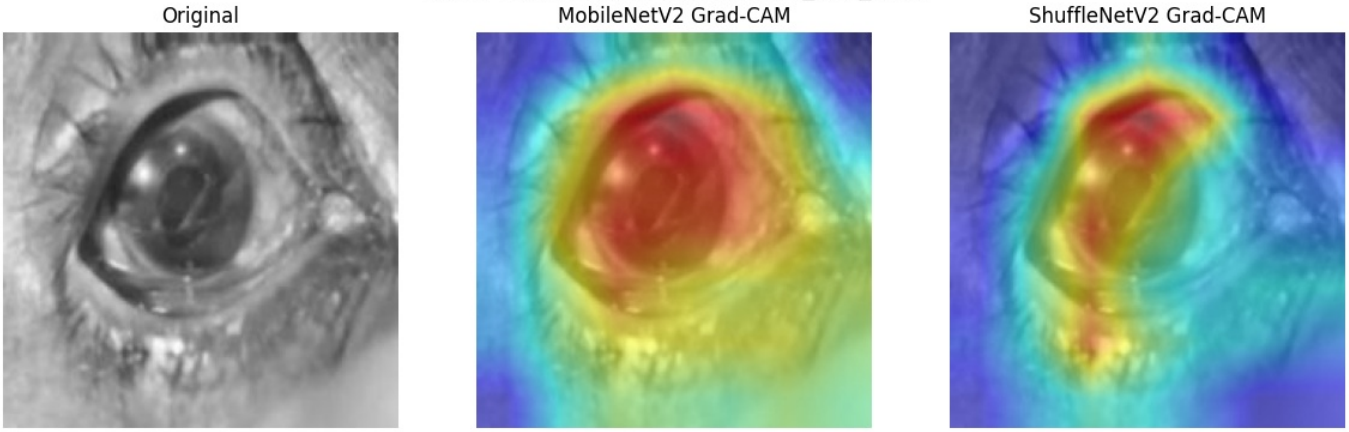
ShuffleNetV2 Grad-CAM



Class: Conjunctivitis | File: image_41_3.jpg



Class: Conjunctivitis | File: image_257_1.jpg



Class: Conjunctivitis | File: image_165_3.jpg