**T.C.**
**SAKARYA ÜNİVERSİTESİ**
**BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ**
**BİLİŞİM SİSTEMLERİ MÜHENDİSLİĞİ**

# 2.ÖDEV RAPORU

**B211200009**
**İrem KABAOĞLU**

**ISE 456 Bilgisayar Görmesine Giriş**
**Dr.Öğr.Üyesi Fatma AKALIN**

# Diabetic Retinopathy 2015 Data Colored Resized

2.Ödev kapsamında Diabetic Retinopathy 2015 Data Colored Resized isimli veri kümesinin içinden Severe, Proliferate_DR, No_DR, Moderate ve Mild kategorileri için rastgele 200'er görüntüyü Python kullanarak bir klasöre kopyalayıp kendi data setimi oluşturdum.

Bununla ilgili Python kodunu aşağıda görebilirisiniz.

In [2]:
```python
import os
import random
import shutil

# Kaynak ve hedef klasörler
source_dir = r'C:\Users\USER\Desktop\colored_images'  # Orijinal veri setinin olduğu klasör
output_dir = r'C:\Users\USER\Desktop\colored_dataset'  # Seçilen görüntülerin kaydedileceği klasör

# Kategoriler
categories =["Mild", "Moderate", "Severe", "Proliferate_DR", "No_DR"]
image_count = 200  # Her kategoriden seçilecek görüntü sayısı


os.makedirs(output_dir, exist_ok=True)
for category in categories:
    os.makedirs(os.path.join(output_dir, category), exist_ok=True)

# Görüntüleri rastgele seçip kopyalama işlemi
for category in categories:
    source_category_path = os.path.join(source_dir, category)
    output_category_path = os.path.join(output_dir, category)

    # Kategori klasöründeki tüm görüntüleri listele
    all_images = [img for img in os.listdir(source_category_path) if img.endswith((".jpg", ".png", ".jpeg"))]

    # Eğer klasörde 200'den az görüntü varsa hata vermemesi için kontrol
    selected_images = random.sample(all_images, min(image_count, len(all_images)))

    # Seçilen görüntüleri yeni klasöre kopyala
    for img in selected_images:
        shutil.copy(os.path.join(source_category_path, img), os.path.join(output_category_path, img))

    print(f"{category} kategorisinden {len(selected_images)} görüntü seçildi ve kopyalandı.")

print(" İşlem tamamlandı! Seçilen görüntüler 'colored_dataset' klasörüne kaydedildi.")
```

```
Mild kategorisinden 200 görüntü seçildi ve kopyalandı.
Moderate kategorisinden 200 görüntü seçildi ve kopyalandı.
Severe kategorisinden 200 görüntü seçildi ve kopyalandı.
Proliferate_DR kategorisinden 200 görüntü seçildi ve kopyalandı.
No_DR kategorisinden 200 görüntü seçildi ve kopyalandı.
 İşlem tamamlandı! Seçilen görüntüler 'colored_dataset' klasörüne kaydedildi.
```

[ ]:

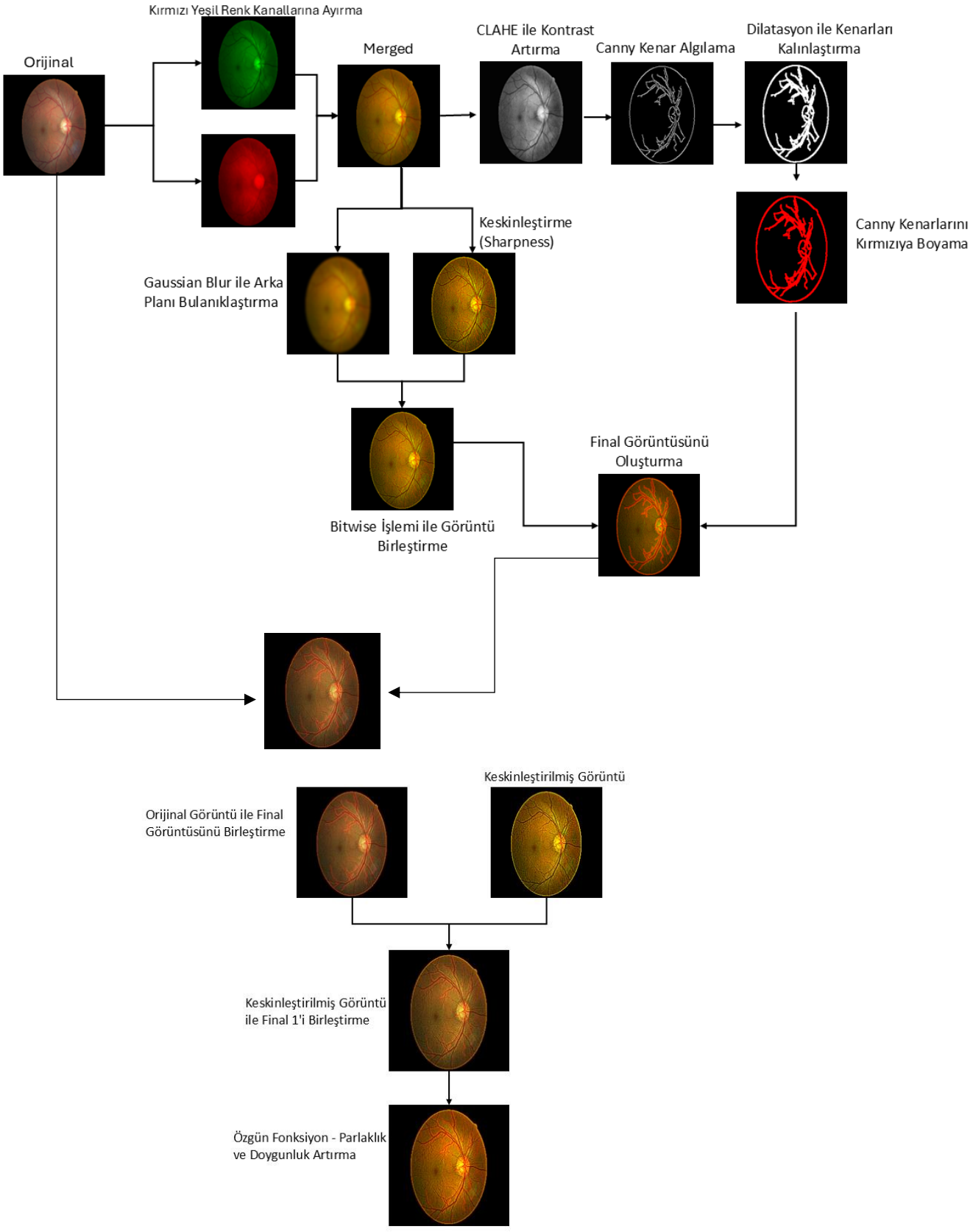
Bu bölümde görsel üzerinde yapılan işlemleri adım adım inceleyeceğiz:

1. **Orijinal Görsel:**
   Orijinal renkli görsel (img_rgb) doğrudan okunur. Görselde hiçbir işlem yapılmaz, sadece görüntü üzerinde işlem yapabilmek için okunur ve RGB formatına dönüştürülür.
2. **Kanalların Ayrılması:**
   extract_channels fonksiyonu kullanılarak, orijinal görselin kırmızı (R) ve yeşil (G) kanalları ayrılır ve yeni bir görüntü (merged_img) oluşturulur. Bu yeni görüntü, sadece kırmızı ve yeşil kanalları içerir, mavi kanal sıfırlanır.
3. **CLAHE ile Kontrast Artırma:**
   Yeni oluşturulan merged_img görüntüsü gri tonlama (cv2.cvtColor) işlemine tabi tutulur. Ardından CLAHE (Contrast Limited Adaptive Histogram Equalization) ile kontrast artırılır.

4. **Keskinleştirme (Sharpness):**
   Keskinleştirme (Sharpness) işlemi yapılır, burada özel bir keskinleştirme filtresi (sharpening kernel) kullanılır.
5. **Gaussian Blur ile Arka Planı Bulanıklaştırma:**
   Görüntüye Gaussian Blur uygulanarak arka plan bulanıklaştırılır.
6. **Kenar Tespiti (Canny Edge Detection):**
   Canny Edge Detection ile kenarlar tespit edilir ve görüntüdeki kenarları ortaya çıkaran bir mask oluşturulur.
7. **Dilatasyon ile Kenarları Kalınlaştırma:**
   Dilatasyon işlemi ile tespit edilen kenarların kalınlığı artırılır.
8. **Bitwise İşlemi (Ortalama Birleştirme):**
   Keskinleştirilmiş görüntü ve bulanıklaştırılmış arka plan bitwise işlemi ile birleştirilir.
9. **Canny Kenarlarını Kırmızı Renge Boyama:**
   Canny kenarları kırmızı renkte boyanır ve diğer kanallar (yeşil ve mavi) sıfırlanır.
10. **Final Görüntüsünü Oluşturma:**
    Canny kenarları ile birleştirilmiş final görüntüsü elde edilir.
11. **Orijinal Görüntü ile Final Görüntüsünü Birleştirme:**
    Orijinal görüntü ile final görüntüsü birleştirilir (orta seviyede karıştırılarak).
12. **Keskinleştirilmiş Görüntü ile Final 1'i Birleştirme:**
    Keskinleştirilmiş görüntü ile final1 görüntüsü birleştirilir.
13. **Özgün Fonksiyon - Parlaklık ve Doygunluk Artırma:**
    Parlaklık ve doygunluk artırma fonksiyonu, HSV renk uzayında parlaklık ve doygunluk oranlarını artırmak için uygulanır.Ödev kapsamında özgün bir fonksiyon geliştirmemiz istendi.Geliştirdiğim fonksiyon, bir resmi renk açısından daha canlı ve gözle görülür şekilde daha parlak hale getiriyor.

```
41]: def enhance_saturation_brightness(image, saturation_scale=1.3, brightness_scale=1.2):
         """
         HSV renk uzayında doygunluk ve parlaklık artırma.

         :param image: Girdi RGB görüntüsü
         :param saturation_scale: Doygunluk için çarpan
         :param brightness_scale: Parlaklık için çarpan
         :return: Parlaklık ve doygunluğu artırılmış RGB görüntüsü
         """
         hsv_image = cv2.cvtColor(image, cv2.COLOR_RGB2HSV)  # RGB'den HSV'ye dönüştür
         h, s, v = cv2.split(hsv_image)  # HSV kanallarını ayır

         # Doygunluğu ve parlaklığı artır
         s = cv2.multiply(s, saturation_scale)  # Doygunluk artırma
         v = cv2.multiply(v, brightness_scale)  # Parlaklık artırma

         # Kanalları birleştir
         hsv_image = cv2.merge([h, s, v])

         # RGB'ye geri dönüştür
         enhanced_image = cv2.cvtColor(hsv_image, cv2.COLOR_HSV2RGB)
         return enhanced_image

     # Parlaklık ve doygunluk artırma
     enhanced_img_with_saturation = enhance_saturation_brightness(final2_image)
```

Bu akışın görselini aşağıda bulabilirsiniz. Biraz uzun ve karışık bir akış olduğundan dolayı ikiye bölünerek gösterilmiştir.



# DenseNet121 Modeli ile Transfer Learning Uygulanması

Ödev kapsamında bir göz hastalığı sınıflandırma problemi için DenseNet121 modelini kullandım. Modeli sıfırdan eğitmek yerine, ImageNet veri kümesinde önceden eğitilmiş ağırlıkları kullanarak fine-tuning yöntemi uyguladım.Overfitting'i önlemek ve eğitim süresini azaltmak amacıyla Early Stopping kullandım.Eğitim sırasında validation loss 5 epoch boyunca iyileşmezse duracak. Orijinal veriler ile işlenmiş veriler üzerinde modeli uyguladıktan sonra model performans karşılaştırması kapsamında grafik oluşturdum.

Bu karşılaştırma, doğruluk (accuracy), duyarlılık (recall), özgüllük (specificity), kesinlik (precision) ve F1-skoru (F1 score) gibi ölçütler üzerinden yapılmış ve işlenmiş görüntüler ile orijinal görüntüler arasındaki performans farkları analiz edilmiştir. Analiz tablosunu ve histogram grafiğini aşağıda görebilirsiniz.

| Ölçüt | Orijinal Veri | İşlenmiş Veri |
|---|---|---|
| **Doğruluk (Accuracy)** | 38.0 | 34.0 |
| **Duyarlılık (Recall)** | 38.0 | 34.0 |
| **Özgüllük (Specificity)** | 86.6 | 83.6 |
| **Kesinlik (Precision)** | 37.0 | 35.0 |
| **F1-Score** | 37.0 | 34.0 |