**ÜSKÜDAR ÜNİVERSİTESİ**

**FEN BİLİMLERİ ENSTİTÜSÜ**

**YAPAY SİNİR AĞI-FİNAL Ödevi**

1. **Gerekli Kütüphanelerin Yüklenmesi**

* pandas, numpy (veri işleme)
* sklearn, TensorFlow/Keras (model oluşturma, veri ayırma, performans metrikleri)
* matplotlib ve seaborn (görselleştirme)

1. **Veri Seti Yükleme**

Face Mask Detection Dataset’i yükleyiniz.

<https://www.kaggle.com/datasets/omkargurav/face-mask-dataset>

1. **Önceden Eğitilmiş (Pre-trained) Modellerin Hazırlanması**

Sınıflandırıcı olarak DenseNet121 ve MobileNetV1 (pre-trained) modellerinden en az biri kullanılacaktır. Base modeller indirildikten sonra 2 dense katmanı ve bir dropout katmanı ekleyiniz.

Örn: Kütüphane yükleme

from tensorflow.keras.applications import DenseNet121

from tensorflow.keras.applications import MobileNet

Örn: Model oluşturma

        base\_model = DenseNet121(weights='imagenet', include\_top=False, input\_shape=(image\_size[0], image\_size[1], 3))

        x = base\_model.output

        x = GlobalAveragePooling2D()(x)

        x = Dropout(0.5)(x)

        # x = Dense(128, activation='relu')(x)

        x = Dense(256, activation='relu')(x)

        x = Dense(256, activation='relu')(x)

        x = Dense(128, activation='relu')(x)

        x = Dense(64, activation='relu')(x)

        predictions = Dense(2, activation='softmax')(x)

        model = Model(inputs=base\_model.input, outputs=predictions)

1. **Veri Setinin Eğitim/Validasyon ve Test Olarak Ayrılması:**

Toplam verinin **%20’si** **test set** olarak ayrılmalı, **%80’i** eğitim/validasyon için bırakılmalıdır.

1. **Eğitim/Validasyon Setinin Bölünmesi ve Modelin Eğitilmesi**

Daha önce ayırdığınız **eğitim+validasyon** verisini **%80’i** eğitim **%20’si** validasyon olarak rastgele bölün. Eğitim verisine veri arttırımı uygulayın

1. datagen = ImageDataGenerator(
2. rescale=1.0/255,
3. rotation\_range=10,
4. width\_shift\_range=0.1,
5. height\_shift\_range=0.1,
6. shear\_range=0.1,
7. zoom\_range=0.1,
8. horizontal\_flip=True,
9. fill\_mode='nearest'
10. )

* Modellerinizi **100 epok** boyunca eğitin.
* **Early Stopping** kullanın: patience=25 olarak ayarlayın. 25 epok boyunca validasyon kaybında (validation loss) iyileşme olmazsa eğitim duracaktır.
* **Optimizer** olarak **ADAM** (Adaptive Moment Estimation) veya **SGD** (Stochastic Gradient Descent) kullanın.
* Öğrenme hızı lr başlangıç değerini 0.001 olarak başlatın.
* Momentum değeri isteğe bağlı ayarlanabilir (örn. 0.9).
* **ReduceLROnPlateau** mekanizması ile validasyon kaybı iyileşme göstermediğinde öğrenme hızını (lr) otomatik olarak azaltın. Örneğin: factor=0.1, patience=5 vs.
* **Eğitim/Validasyon Accuracy ve Loss Grafiklerini** rapora ekleyin.

Her epok sonunda eğitim kaybı (loss) ve doğruluğu (accuracy) ile validasyon kaybı ve doğruluğunu kaydedin ve eğriyi çizdirin.

**4. Test Verisi İle Tahmin (Prediction) ve Sonuç Analizi**

* Eğitilen model(ler)in en iyi kayıtlı ağırlıklarını (weights) yükleyip test set üzerinde tahmin (inference) yapın.

Çıktı olarak aşağıdaki metrikleri rapora ekleyin:

* ROC Eğrisi (Receiver Operating Characteristic Curve) **ve AUC (Area Under Curve)**
* Confusion Matrix **(Karışıklık Matrisi)**
* Accuracy **(Doğruluk)**
* Precision **(Duyarlılık)**
* Recall **(Hassasiyet / Duyarlılık)**
* Specificity **(Özgüllük)**
* F1-score

**Teslim Edilecekler**

1. **Kod**: Tüm adımları içeren ve çalıştırılabilir bir Jupyter Notebook dosyası.
2. **Sonuçlar ve Yorumlar**: Confusion matrix, performans metrikleri ve hiperparametre denemelerine göre elde edilen sonuçlarla birlikte MLP ve RBF modellerinin karşılaştırılması.
3. **Yorumlar**: Model performansını etkileyen hiperparametreler, model seçimleri ve sonuçlarla ilgili kısa açıklamalar.
4. Github linki

<https://github.com/zeki-onur-yuksek/y.zeka.prenispleri_final>

1. Word dosyası

Model, toplam 100 epoch’a kadar eğitilecek şekilde ayarlanmış, ancak 38. epoch’ta erken durdurulmuş (early stopping).

En iyi epoch 13. epoch olarak belirlenmiş ve model bu noktadaki ağırlıklarla geri yüklenmiş.

Eğitim doğruluğu (accuracy): %98.51 - %99.22 arasında.

Doğrulama doğruluğu (val\_accuracy): %99.92'ye kadar çıkmış, yani model doğrulama verisinde oldukça yüksek performans gösteriyor.

Eğitim kaybı (loss): 0.0257’ye kadar düşmüş, bu iyi bir işaret.

Doğrulama kaybı (val\_loss): 0.0074 - 0.0092 civarında, düşük bir kayıp değeri.

Öğrenme oranı (learning\_rate): 1e-8’e kadar düşmüş, bu da modelin daha fazla öğrenemediğini gösteriyor.

Eğitim doğruluğu doğrulama doğruluğuna çok yakın olduğu için overfitting riski düşük görünüyor. Ancak 13. epoch'tan sonra iyileşme olmaması, çok uzun eğitim yapılırsa overfitting riskinin artabileceğini gösteriyor.

Accuracy (Doğruluk): 0.9940 (yaklaşık %99.40)

Tüm örneklerin %99.40’ının doğru sınıflandırıldığını gösterir. Yani, modelin genel olarak hata yapma oranı çok düşüktür.

Precision (Kesinlik): 0.9933 (yaklaşık %99.33)

Modelin pozitif olarak tahmin ettiği örneklerin %99.33’ünün gerçekten pozitif olduğunu belirtir. Bu, modelin pozitif sınıfı ne kadar "güvenle" tahmin ettiğini ifade eder.

Recall (Duyarlılık / Hassasiyet): 0.9946 (yaklaşık %99.46)

Gerçek pozitif örneklerin %99.46’sının doğru şekilde tespit edildiğini gösterir. Yani, model neredeyse tüm pozitif örnekleri kaçırmamış.

F1-score: 0.9940 (yaklaşık %99.40)

Precision ve recall değerlerinin harmonik ortalamasıdır. Hem yanlış pozitif hem de yanlış negatif oranlarının düşük olması, bu değerin yüksek çıkmasına neden olmuştur.

Specificity (Özgüllük): 0.9946 (yaklaşık %99.46)

Gerçek negatif örneklerin %99.46’sının doğru şekilde negatif olarak sınıflandırıldığını belirtir. Bu, modelin negatif sınıfa ilişkin de çok başarılı olduğunu gösterir.