**TOBB ETÜ Bilgisayar Mühendisliği Bölümü**
**BİL 468 Dönem Projesi**
**2023-24 Bahar Dönemi**

**Proje Grup No:** 7

## 1. Problem Tanımı (girdi, çıktı, sınıflar)

Bu proje, bilgisayar görü teknolojilerini kullanarak yüz ifadelerini tanıyarak duygusal durumu analiz etmeyi amaçlar. Temel amaç, girdi olarak alınan görsellerdeki insanların yüz ifadelerinden duygusal durumlarını anlamak ve buna uygun çıktı verebilmektir. Proje kapsamında geliştirilecek model, görüntüdeki insanın yüz ifadesinden mutlu, üzgün, sinirli, şaşkın, normal, iğrenmiş, korkmuş olmak üzere 7 farklı duygudan hangisini ifade ettiğini saptayacaktır.

Geliştirilecek modelin girdileri yakın çekimden yüzün tamamını kaplayan görüntüler olacaktır. Sınıflarımız yukarıda bahsettiğimiz 7 farklı duygu durumu olacaktır. Çıktılar ise görüntülerdeki yüzün bu 7 sınıftan hangisine sahip olduğunun bir değeridir.

## 2. Veri Setleri (veri setlerinin kısa açıklaması ve linki, sınıflara ait veri sayıları ve örneği)

1. https://www.kaggle.com/datasets/msambare/fer2013

   FER-2013 veri seti, yüzleri duygusal ifadelerine dayalı olarak yedi kategoriye (0=Kızgın, 1=İğrenmiş, 2=Korku, 3=Mutlu, 4=Üzgün, 5=Şaşkın, 6=Normal) sınıflandırmak amacıyla kullanılan gri tonlamalı görüntülerden oluşmaktadır.

   Yüzler otomatik olarak kaydedilmiştir, böylece yüz neredeyse merkezlenmiş ve her görüntüde yaklaşık olarak aynı miktarda alanı kaplamaktadır.

   Eğitim seti, 28,709 örneği içermekte olup, halka açık test seti ise 3,589 örneği içermektedir

   Aşağıda veri setinden örnekler görebilirsiniz.



   Korku Üzgün Şaşkın

2. https://www.kaggle.com/datasets/jonathanoheix/face-expression-recognition-dataset/data

   Aynı şekilde bu veri seti de yedi katogoriye (0=Kızgın, 1=İğrenmiş, 2=Korku, 3=Mutlu, 4=Üzgün, 5=Şaşkın, 6=Normal) sınıflandırmak amacıyla kullanılan gri tonlamalı görüntülerden oluşmaktadır.

   Veri seti, 28,821 eğitim ve 7,066 test örneğinden oluşmaktadır.

   Bu iki veri setinin aynı duygusal sınıflandırmalara sahip olması, ayrıcaı bir avantajdır. Bu, modelin genel duygusal ifadeleri daha iyi anlamasına ve doğru sınıflandırma yapmasına yardımcı olabilir.

   Aşağıda veri setinden örnekler birkaç örnek bulabilirsiniz.



   İğrenmiş Kızgın Doğal

**3. Yöntemler**

## Yöntem 1 – Görüntü İşleme ve Özellik Eşleştirme (SIFT)

SIFT (Scale-Invariant Feature Transform) tabanlı bir yüz ifadesi sınıflandırma sistemi için, görüntüden özellikler (features) çıkardık ve bu özellikleri eşleştirmek için kullandık. Bu yaklaşım, genellikle referans görüntülerle eşleştirme yaparak çalışır ve bu yüzden tipik olarak bir makine öğrenmesi modelinin eğitimini gerektirmez. Bunun yerine, her yüz ifadesi sınıfı için referans görüntüler kullanılır ve test görüntüsündeki özellikler bu referanslarla karşılaştırılarak en iyi eşleşme bulunur ve bu yöntemle yüz ifadesi sınıflandırılmış olur.

**Adımlar:**

1- Referans görüntüler seti oluşturduk. Her bir yüz ifadesi sınıfı için bir dizi referans görüntüsünü veri setlerimizdeki train verilerinden seçtik.
2- SIFT ile her referans görüntüsü için özellikler (keypoints ve descriptors) çıkardık ve kaydettik. Bu aşamada keypoint ve descriptor'ları OpenCV kütüphanesi kullanarak hesapladık.
3- Adım 2'deki işlemi veri setlerimizdeki test verileri için uyguladık. Bunun sonucunda referans özellik seti çıkardık.
4- Çıkarılan test görüntüsü özelliklerini referans görüntülerin özellikleriyle eşleştirdik. Bu aşamayı OpenCV'nin knnMatch fonksiyonunu kullanarak yaptık. Bu fonksiyon, iki descriptor seti arasında k-en yakın komşular eşleşmesi yapar.
5- Her sınıf için eşleşme sayısını hesapladık ve en yüksek eşleşmeye sahip sınıfı test görüntüsünün sınıfı olarak belirledik.

Bu aşamadaki işlemleri yapacak pyhton class'ını utils/sift.py dosyasındaki SIFT sınıfı içerisinde uyguladık. Notebook içerisinde bu class import edilerek yazdığımız methodlar yardımıyla modelin testlerini gerçekleştirdik.

## Yöntem 2 – Görüntü İşleme ve Makine Öğrenmesi

Temel olarak, Histogram of Oriented Gradients (HOG) tekniği kullanılarak özellik çıkarımı yaptık ve böylece yüz ifadelerini temsil eden özellikler vektörlerini elde ettik. Daha sonra, bu HOG özellikleri ile ayrı ayrı SVM, KNN, RF ve MLP modellerini 2 farklı veri seti için eğittik ve test ettik

**Genel (SVM, KNN, RF, MLP) Eğitim ve Test Aşamalarının Adımları:**

**Özellik Çıkarımı:**

Özellik çıkarımı olarak HoG tekniğini kullanarak özellik çıkarımını yaptık. HoG görüntülerdeki önemli bilgileri yakalayarak modelin yüz ifadelerini daha iyi anlamasını sağlar. HoG descriptor'larını OpenCV kütüphanesinden faydalanarak hesapladık. Proje içerisinde utils/hog.py içerisinde tanımlanmış HOGDescriptor sınıfını yazdık. Bu sınıfı eğitim ve test süreçleri için geliştirdiğimiz notebooklarda import ederek kullandık. HoG için kullandığımız parametre değerlerini aşağıdaki gibi kullandık.

```python
class HOGDescriptor:
    def __init__(self, win_size=(48, 48), block_size=(16, 16), block_stride=(8, 8), cell_size=(8, 8), nbins=9):
        # Initialize HOG descriptor parameters
        self.win_size = win_size  # Size of the image window
        self.block_size = block_size  # Size of blocks
        self.block_stride = block_stride  # Block stride
        self.cell_size = cell_size  # Size of cells
        self.nbins = nbins  # Number of bins for histograms

        # Create the HOG descriptor and detector
        self.hog = cv2.HOGDescriptor(self.win_size, self.block_size, self.block_stride, self.cell_size, self.nbins)
```

**win_size: Pencere boyutu (48, 48)** - Bu, özelliklerin çıkarılacağı ana görüntü bölgesinin boyutudur. Genellikle, tanınacak nesnenin tümünü kaplayacak şekilde seçilir. Burada (48, 48) demek, her bir özellik çıkarım penceresinin 48x48 piksel boyutunda olduğunu gösterir.
**block_size: Blok boyutu (16, 16**) - HOG, görüntü üzerinde yerel olarak hesaplanır ve bu hesaplama bloklar halinde yapılır. Bir blok, birden fazla hücreyi içerebilir. Burada (16, 16) demek, her bir bloğun 16x16 piksel boyutunda olduğunu ifade eder.
**block_stride: Blok adımı (8, 8) -** Blok adımı, bir sonraki bloğun ne kadar kaydırılacağını belirler. Bu değer, blokların birbirleri üzerinde ne kadar örtüşeceğini kontrol eder. (8, 8) burada, her yönde 8 piksel adım ile blokların kaydırılacağını gösterir.
**cell_size: Hücre boyutu (8, 8) -** Hücre boyutu, her bir hücre içindeki piksel sayısını belirler. HOG özelliklerinin hesaplanması sırasında, gradyan yönleri ve büyüklükleri bu hücreler içinde toplanır. Buradaki (8, 8) değeri, her bir hücrenin 8x8 piksel boyutunda olduğunu gösterir. Hücreler, özellik çıkarımının temel birimidir
**nbins: Yönlendirme kutularının sayısı (9) -** nbins, her bir hücre içindeki gradyan yönlendirme aralığının kaç farklı kutuya bölüneceğini belirtir. Bu parametre, gradyan yönlendirme histogramının çözünürlüğünü tanımlar. Buradaki değer olan 9, gradyan yönlendirmelerinin 0'dan 180 dereceye kadar 9 farklı aralığa bölündüğü anlamına gelir. Bu, her bir hücre için gradyan yönlendirme bilgisinin ne kadar detaylı tutulacağını belirler

**Eğitim**:

Eğitim aşamasında, çıkarılan HOG özellikleri ve ilgili etiketler (ifadeler) eğitim verileri ile tek tek hesaplanmıştır ve daha sonrasında yukarıda bahsettiğimiz 4 farklı Makine Öğrenmesi modellerini besleyerek eğittik. Modelleri 2 farklı veriseti için ayrı ayrı kaydettik. Eğitim aşamasında Pyton Scikit-Learn kütüphanesini kullandık.

**Test**: Son adım olarak, makine öğrenmesi modellerini daha önce görmediği veriler üzerinde test ettik. Bu, modelin gerçek dünya verileri üzerindeki performansını değerlendirmek için kullanılır.
Test aşamasının sonuçlarını farklı metrikler üzerinden hesapladık ve kaydettik. Test aşamasında sonuçların hesaplanması, metriklerin hesaplanması gibi işlemlerin hepsini Scikit-Learn kütüphanesini kullanarak yaptık. Ayrıca Confusion Matrix grafiğini yazdırmak için matplotlib ve seaborn kütüphanelerini kullandık.

## Makine Öğrenmesi Modelleri

1. **SVM**

```
# Train SVM model with the features and labels
clf = svm.SVC(kernel='linear', decision_function_shape='ovo')  # One-vs-one SVM
clf.fit(features, labels)
```

**Model Türü: Destek Vektör Makinesi (SVM) -** Yüksek ve düşük boyutlu veri setleri için etkili bir sınıflandırma yöntemi.

**kernel: 'linear' -** Karar sınırının lineer (doğrusal) olduğunu belirtir. Bu, öznitelik uzayında doğrusal olarak ayrılabilir veri setleri için uygun bir seçenektir.

**decision_function_shape: 'ovo' (One-vs-One) –** Normalde SVM 2 farklı sınıf için bir ayrıştırıcı modelidir. Bizim problemimizde birden fazla sınıf olduğu için her bir sınıf çifti için ayrı bir sınıflandırıcı oluşturulur. Dolayısıyla toplam n adet sınıf için, toplamda $n.(n-1)/2$ adet sınıflandırıcı oluşturulmuş oldu. Her bir sınıflandırıcı, bir sınıf çiftini diğerlerinden bağımsız olarak ayırmaya çalışır. Bir test örneği için, tüm sınıflandırıcıların kararları birleştirilerek final sınıfı belirlenir.

2. **KNN**

```
# Train KNN model with the features and labels
clf = KNeighborsClassifier(n_neighbors=3)  # Adjust the number of neighbors as needed
clf.fit(features, labels)
```

**Model Türü: K-En Yakın Komşu (KNN)** - Basit bir sınıflandırma ve regresyon yöntemi. Yeni bir örnek geldiğinde, bu örneğe en yakın olan k adet eğitim örneğine bakılır ve bu örneklerin çoğunluk sınıfı ya da ortalama değeri, yeni örneğin sınıfı veya değeri olarak atanır.

**n_neighbors: 3** - Modelin bakacağı en yakın komşu sayısını belirtir. Bu parametre, modelin karmaşıklığını ve genelleştirme kabiliyetini doğrudan etkiler. Daha küçük bir k değeri, modelin eğitim verisine daha yakın olmasını sağlayarak daha fazla varyans yaratırken, daha büyük bir k değeri modelin daha genelleştirilmiş ve daha az duyarlı olmasına neden olur.

3. **RF**

```
# Train Random Forest model with the features and labels
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(features, labels)
```

**Model Türü: Rastgele Orman (Random Forest)** - Karar ağaçlarını kullanarak çalışan bir öğrenme yöntemi.

**n_estimators: 100** - Oluşturulacak karar ağacı sayısı. Daha fazla ağaç, genellikle modelin doğruluğunu artırır, ancak hesaplama maliyetini ve zamanını da artırabilir. Bu parametre, modelin hem genelleme kabiliyetini hem de eğitim süresini etkiler.

**random_state: 42** - Modelin eğitimi sırasında kullanılan rastgelelik tohum değeri. Bu değer sayesinde, modelin eğitimi tekrarlandığında aynı sonuçların elde edilmesi sağlanır, böylece sonuçlar tekrarlanabilir olur.

4. **MLP**

```
# Train MLP model with the features and labels
clf = MLPClassifier(hidden_layer_sizes=(100,), activation='relu', solver='adam', max_iter=200, random_state=42)
clf.fit(features, labels)
```

**Model Türü: Çok Katmanlı Algılayıcı (MLP) -** Yapay Sinir Ağlarına dayalı bir sınıflandırma algoritması.
**hidden_layer_sizes: (100,) -** Modelin gizli katmanlarının boyutunu belirtir. Bizim projemizde, model tek bir gizli katmana sahip ve bu katman 100 nöron içeriyor. Gizli katman sayısı ve her birindeki nöron sayısı modelin karmaşıklığını ve öğrenebilme kapasitesini doğrudan etkiler.
**activation: 'relu' -** Gizli katmanlardaki nöronların aktivasyon fonksiyonu olarak ReLU (Düzeltilmiş Lineer Ünite) kullanılır. ReLU, genellikle sinir ağlarında gizli katmanlar için tercih edilen aktivasyon fonksiyonudur çünkü gradient kaybını azaltır ve daha hızlı eğitim sağlar.
**solver: 'adam' -** Optimizasyon algoritması olarak Adam kullanılır. Adam, adaptif öğrenme hızları sağlar ve geniş bir yelpazede problemlerde iyi performans gösterir. Özellikle büyük veri setleri ve parametreler üzerinde etkilidir.
**max_iter: 200 -** Maksimum iterasyon (epoch) sayısı. Modelin eğitimi, belirtilen iterasyon sayısına ulaştığında sonlanır. Bu, eğitim sürecinin ne kadar sürdüğünü kontrol etmeye yardımcı olur.
**random_state: 42 -** Modelin eğitimi sırasında kullanılan rastgelelik tohum değeri. Bu sayede, modelin eğitimi tekrarlandığında aynı sonuçların elde edilmesi sağlanır, yani sonuçlar tekrarlanabilir.

## 4. Test Sonuçları ve Tartışma

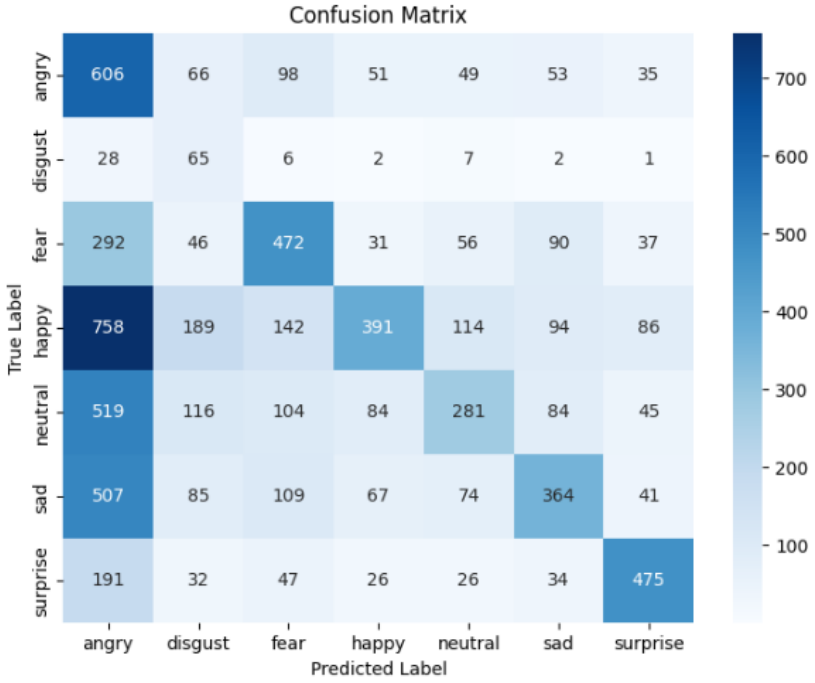
## Yöntem 1 – Görüntü İşleme ve Özellik Eşleştirme (SIFT)

SIFT ile yüz ifadesi sınıflandırmasını 7 farklı sınıf üzerinden yaklaşık %37 doğruluk payı ile yapabilecek bir model geliştirmiştir.

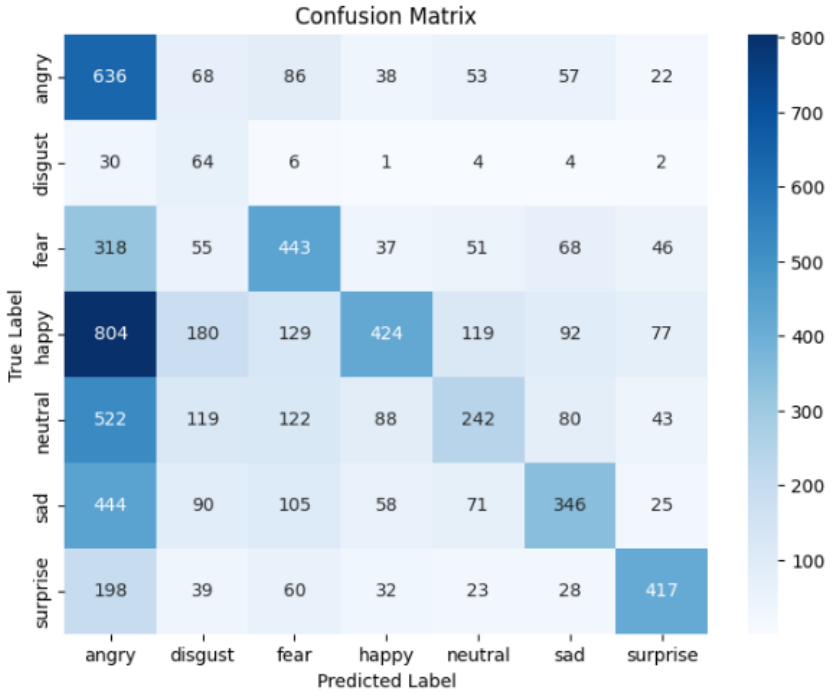
Bu modelin performansı yöntem 2'deki modellerin performansına nazaran daha düşük çıkmıştır. Ayrıca modelin tüm veri seti üzerinden tahmin süresi yine yöntem 2'deki modellere nazaran daha uzun olmuştur. Başarım ve zaman ölçütleri göz önünde alındığında, problemimiz için yöntem 2'nin daha uygun olduğunu gözlemledik.

**SIFT**
**Veriseti 1**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| angry | 0.21 | 0.63 | 0.31 | 958 |
| disgust | 0.11 | 0.59 | 0.18 | 111 |
| fear | 0.48 | 0.46 | 0.47 | 1024 |
| happy | 0.60 | 0.22 | 0.32 | 1774 |
| neutral | 0.46 | 0.23 | 0.31 | 1233 |
| sad | 0.50 | 0.29 | 0.37 | 1247 |
| surprise | 0.66 | 0.57 | 0.61 | 831 |
| accuracy | | | 0.37 | 7178 |
| macro avg | 0.43 | 0.43 | 0.37 | 7178 |
| weighted avg | 0.49 | 0.37 | 0.38 | 7178 |



**Veriseti 2**




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| angry | 0.22 | 0.66 | 0.33 | 960 |
| disgust | 0.10 | 0.58 | 0.18 | 111 |
| fear | 0.47 | 0.44 | 0.45 | 1018 |
| happy | 0.63 | 0.23 | 0.34 | 1825 |
| neutral | 0.43 | 0.20 | 0.27 | 1216 |
| sad | 0.51 | 0.30 | 0.38 | 1139 |
| surprise | 0.66 | 0.52 | 0.58 | 797 |
| accuracy | | | 0.36 | 7066 |
| macro avg | 0.43 | 0.42 | 0.36 | 7066 |
| weighted avg | 0.49 | 0.36 | 0.37 | 7066 |

## Yöntem 2 – Görüntü İşleme ve Makine Öğrenmesi

Makine ögrenmesi modelleri arasında benzer sonuçlar gözlemlenmiş olup aralarında diğerlerine nazaran dikkat çekici performans gösteren bir modele rastlanmamıştır.
Ayrıca, 2 veri seti üzerinde yapılan test sonuçlarını kıyasladığımızda benzer olduğunu gözlemledik.
Bu çalışma sonucunda, Histogram of Gradients yöntemi kullanarak görsellerdeki detaylar yakalanmış ve yüz ifadesi sınıflandırması yapabilecek modeller 7 sınıf üzerinden yaklaşık %50 doğruluk payı ile geliştirmiştir.

Aşağıya bırakılan görsellerden, her bir modelin 2 veri seti üzerindeki performanslarını precision, recall, f1 score, accuracy sonuçlarını görebilir ve confusion matrixlerini inceleyebilirsiniz.
Yukarıda da bahsettiğimiz üzere, yöntem 2'nin performansı yöntem 1'den daha iyi olsa da beklediğimiz performansı elde edemedik. Bir sonraki sprintte, hiper parametre optimizasyonları ile daha yüksek performanslı modeller eğitmeyi hedeflemekteyiz.

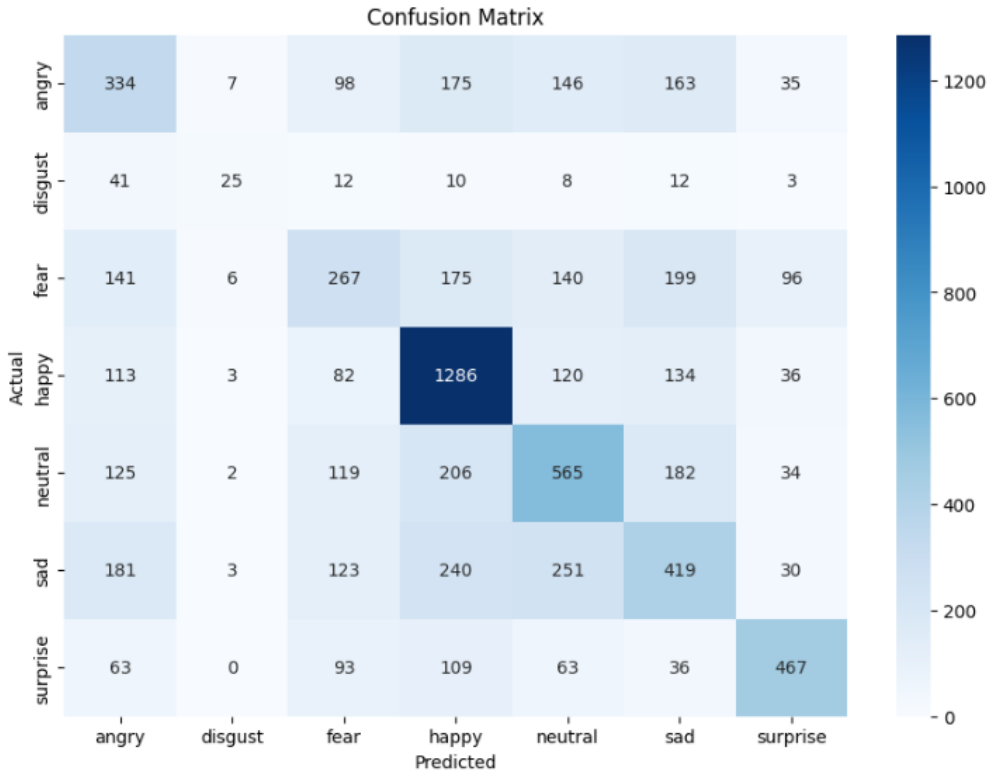
**SVM**
**Veriseti 1**

```
Accuracy: 0.46851490665923656
Precision: 0.46120185204207803
Recall: 0.46851490665923656
F1 Score: 0.4606059623542692

Classification Report:
              precision    recall  f1-score   support

           0       0.33      0.35      0.34       958
           1       0.54      0.23      0.32       111
           2       0.34      0.26      0.29      1024
           3       0.58      0.72      0.65      1774
           4       0.44      0.46      0.45      1233
           5       0.37      0.34      0.35      1247
           6       0.67      0.56      0.61       831

    accuracy                           0.47      7178
   macro avg       0.47      0.42      0.43      7178
weighted avg       0.46      0.47      0.46      7178
```



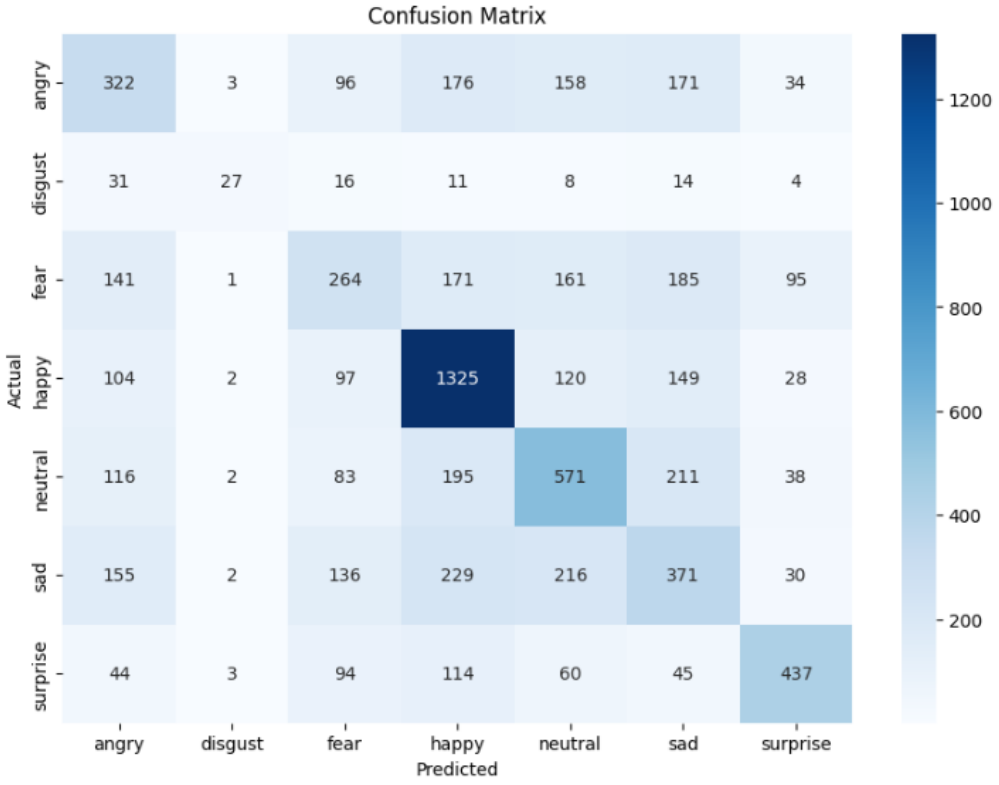
**Veriseti 2**

```
Accuracy: 0.46943107840362297
Precision: 0.46312642360188117
Recall: 0.46943107840362297
F1 Score: 0.4616886690300917

Classification Report:
              precision    recall  f1-score   support

           0       0.35      0.34      0.34       960
           1       0.68      0.24      0.36       111
           2       0.34      0.26      0.29      1018
           3       0.60      0.73      0.65      1825
           4       0.44      0.47      0.45      1216
           5       0.32      0.33      0.32      1139
           6       0.66      0.55      0.60       797

    accuracy                           0.47      7066
   macro avg       0.48      0.42      0.43      7066
weighted avg       0.46      0.47      0.46      7066
```



**KNN**

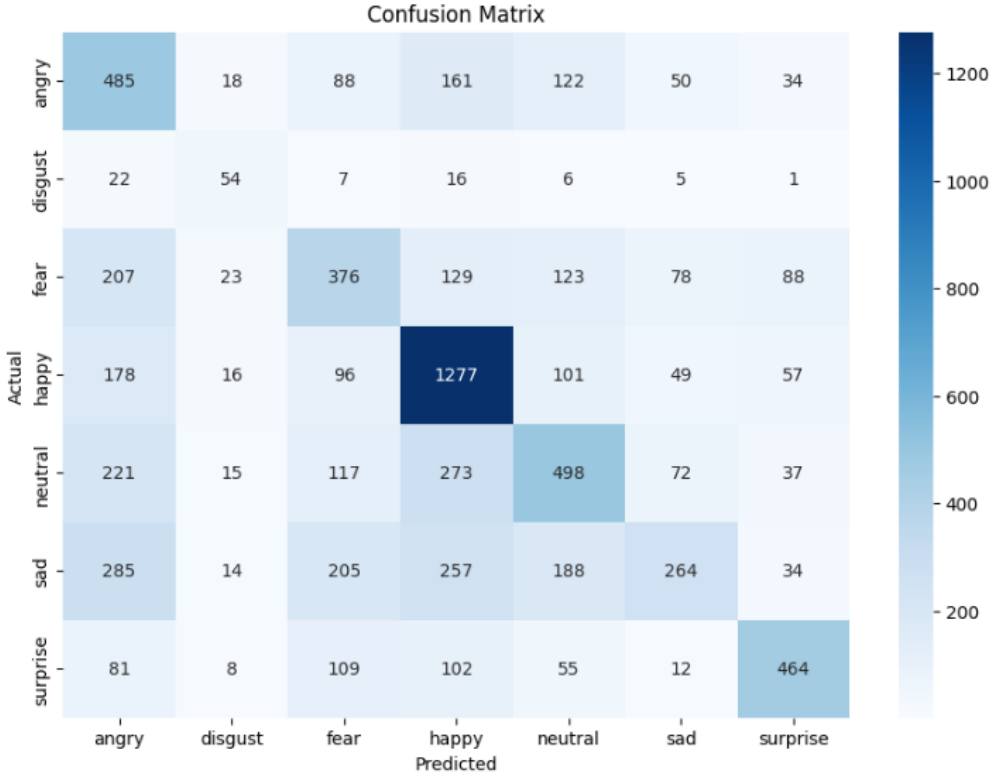
**Veriseti 1**

```
Accuracy: 0.47617720813597103
Precision: 0.48556886652628545
Recall: 0.47617720813597103
F1 Score: 0.46552890041305817

Classification Report:
              precision    recall  f1-score   support

           0       0.33      0.51      0.40       958
           1       0.36      0.49      0.42       111
           2       0.38      0.37      0.37      1024
           3       0.58      0.72      0.64      1774
           4       0.46      0.40      0.43      1233
           5       0.50      0.21      0.30      1247
           6       0.65      0.56      0.60       831

    accuracy                           0.48      7178
   macro avg       0.46      0.46      0.45      7178
weighted avg       0.49      0.48      0.47      7178
```



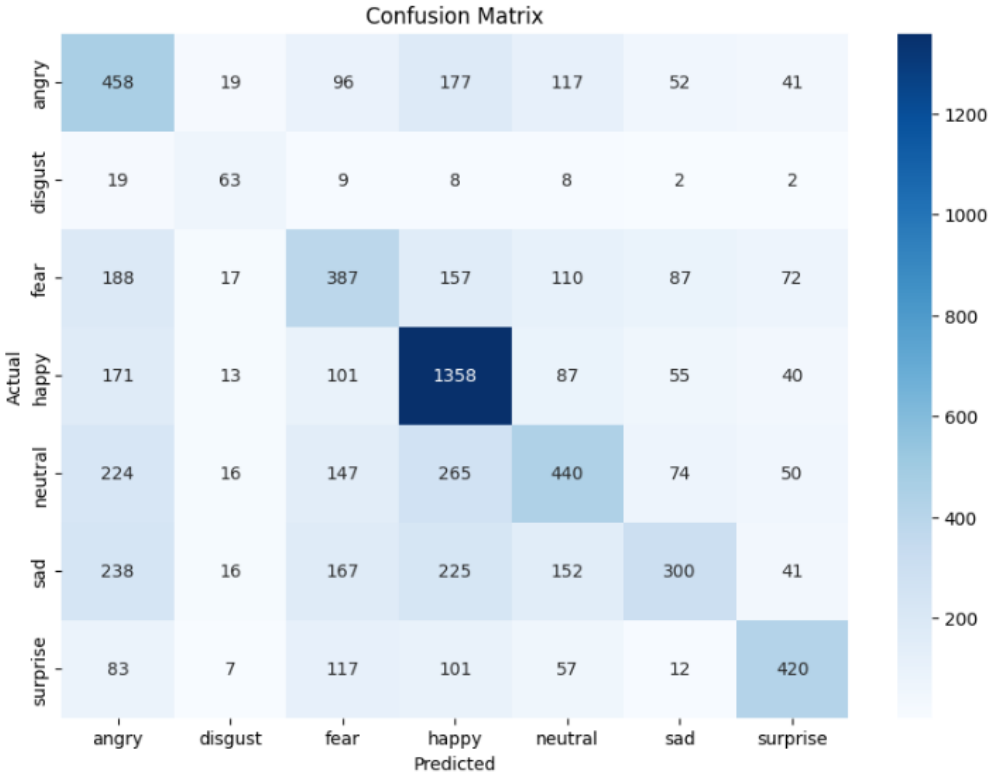
## Veriseti 2

```
Accuracy: 0.4848570619869799
Precision: 0.49135915172016703
Recall: 0.4848570619869799
F1 Score: 0.47595870907881427

Classification Report:
              precision    recall  f1-score   support

           0       0.33      0.48      0.39       960
           1       0.42      0.57      0.48       111
           2       0.38      0.38      0.38      1018
           3       0.59      0.74      0.66      1825
           4       0.45      0.36      0.40      1216
           5       0.52      0.26      0.35      1139
           6       0.63      0.53      0.57       797

    accuracy                           0.48      7066
   macro avg       0.47      0.47      0.46      7066
weighted avg       0.49      0.48      0.48      7066
```




## RF

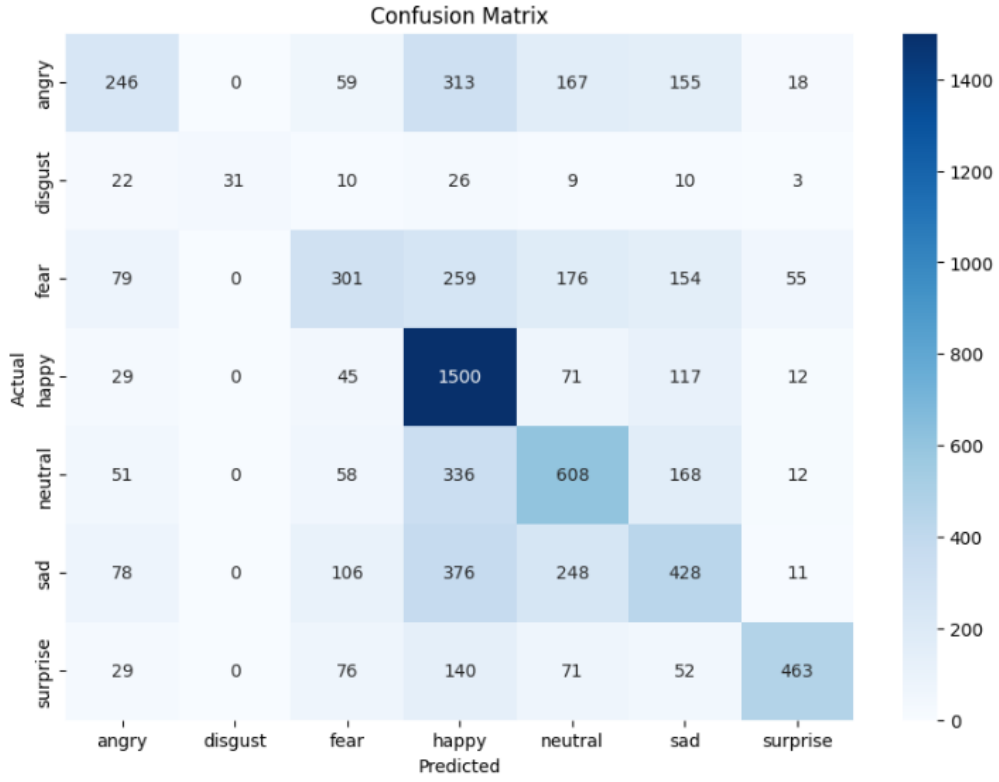
**Veriseti 1**

```
Accuracy: 0.49832822513234887
Precision: 0.5075085995182992
Recall: 0.49832822513234887
F1 Score: 0.47982627041997733

Classification Report:
              precision    recall  f1-score   support

           0       0.46      0.26      0.33       958
           1       1.00      0.28      0.44       111
           2       0.46      0.29      0.36      1024
           3       0.51      0.85      0.64      1774
           4       0.45      0.49      0.47      1233
           5       0.39      0.34      0.37      1247
           6       0.81      0.56      0.66       831

    accuracy                           0.50      7178
   macro avg       0.58      0.44      0.47      7178
weighted avg       0.51      0.50      0.48      7178
```



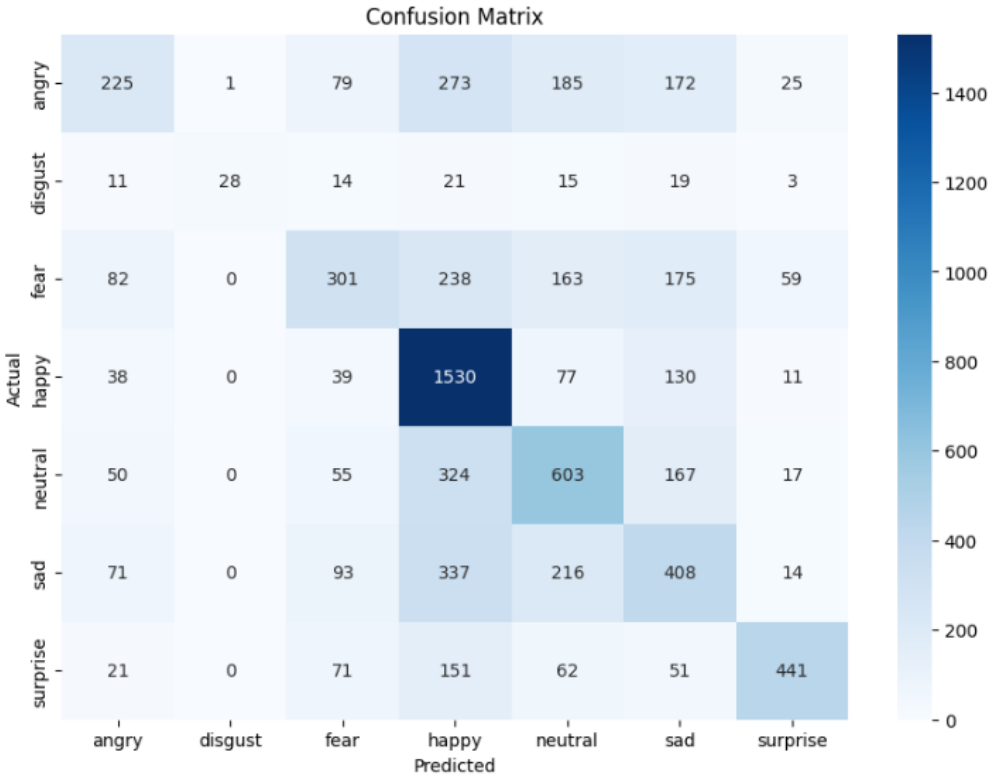
## Veriseti 2

```
Accuracy: 0.5004245683555052
Precision: 0.5049968114908238
Recall: 0.5004245683555052
F1 Score: 0.48109970300448696

Classification Report:
              precision    recall  f1-score   support

           0       0.45      0.23      0.31       960
           1       0.97      0.25      0.40       111
           2       0.46      0.30      0.36      1018
           3       0.53      0.84      0.65      1825
           4       0.46      0.50      0.48      1216
           5       0.36      0.36      0.36      1139
           6       0.77      0.55      0.65       797

    accuracy                           0.50      7066
   macro avg       0.57      0.43      0.46      7066
weighted avg       0.50      0.50      0.48      7066
```



**MLP**

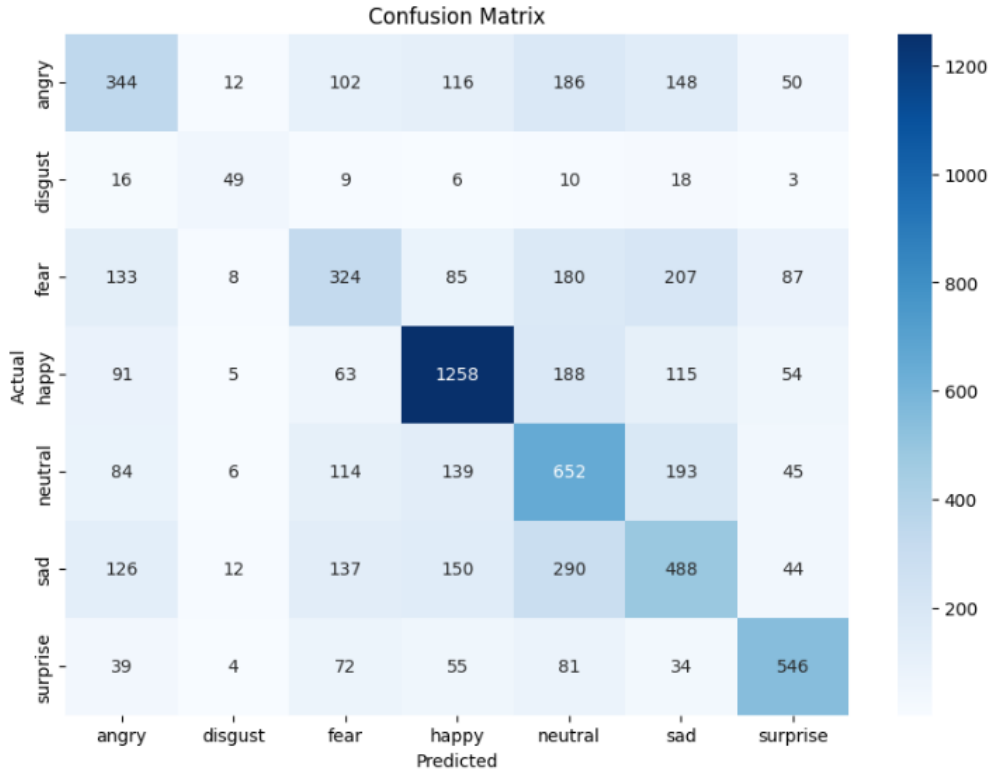
**Veriseti 1**

```
Accuracy: 0.5100306492059069
Precision: 0.5084673768415573
Recall: 0.5100306492059069
F1 Score: 0.5070346834974679

Classification Report:
              precision    recall  f1-score   support

           0       0.41      0.36      0.38       958
           1       0.51      0.44      0.47       111
           2       0.39      0.32      0.35      1024
           3       0.70      0.71      0.70      1774
           4       0.41      0.53      0.46      1233
           5       0.41      0.39      0.40      1247
           6       0.66      0.66      0.66       831

    accuracy                           0.51      7178
   macro avg       0.50      0.49      0.49      7178
weighted avg       0.51      0.51      0.51      7178
```



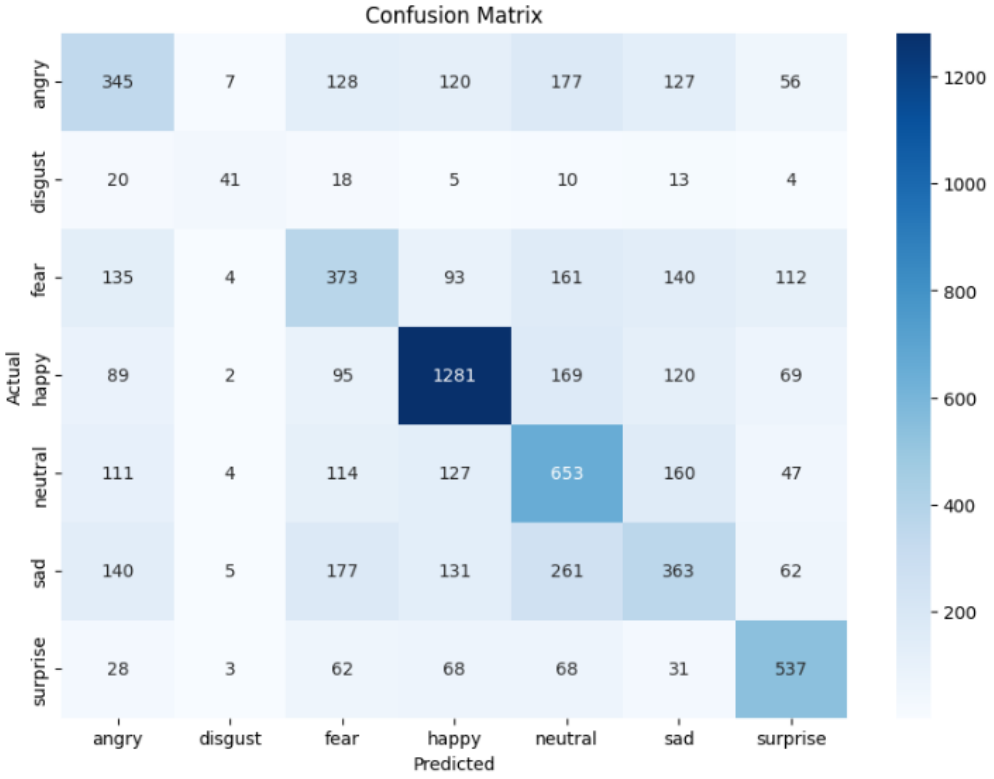
**Veriseti 2**

```
Accuracy: 0.5084913671101047
Precision: 0.5052105963112055
Recall: 0.5084913671101047
F1 Score: 0.5046264379923439

Classification Report:
              precision    recall  f1-score   support

           0       0.40      0.36      0.38       960
           1       0.62      0.37      0.46       111
           2       0.39      0.37      0.38      1018
           3       0.70      0.70      0.70      1825
           4       0.44      0.54      0.48      1216
           5       0.38      0.32      0.35      1139
           6       0.61      0.67      0.64       797

    accuracy                           0.51      7066
   macro avg       0.50      0.48      0.48      7066
weighted avg       0.51      0.51      0.50      7066
```




**5. Sonuçlar**

**Çalışma Sonuçları:**

Çalışmalarımız sonucunda 2 farklı yöntemle toplamda 5 farklı model elde etmiş olduk. Yaptığımız test sonuçlarına göre Yöntem 2'nin toplamda iki yönden daha avantajlı olduğunu gözlemledik. Başarım performansı olarak Yöntem 2'deki modeller benzer sonuçlar yakalamıştır ve Yöntem 1'deki sonuca nazaran çok daha yüksektir. Ayrıca Yöntem 1'deki tahmin süresi Yöntem 2'deki modellere nazaran çok daha fazla vakit almaktadır. Test verisinin özellikler için referans resimlerdeki özellikler ile eşleştirmek epey maliyetli bir işlem olduğu gözlemlenmiştir. Başarım ve Zaman ölçütleri göz önünde bulundurularak Yöntem 2'deki modellerin problemimiz için daha uygun olduğu gözlemlenmiştir.

**Yöntemlerin eksik veya zayıf yönleri:**

**Yöntem 1 – Görüntü İşleme ve Özellik Eşleştirme (SIFT)**

SIFT, dönüşüm ve ölçek değişikliklerine dayanıklı sabit noktaları tespit etmeye odaklanır, fakat yüz ifadeleri gibi değişken ve ince detayları barındıran verilerde bu yaklaşım sınırlı kalmıştır. Farklı cinsiyetlerden, yaşlardan çeşitli senaryoları göz önüne aldığımızda SIFT yetersiz kalmaktadır.

**Yöntem 2 – Görüntü işleme ve Makine Öğrenmesi (SVM, KNN, RF, MLP)**

HoG (Histogram of Oriented Gradients) ve Makine Öğrenmesi tabanlı yöntemler, yüz ifadesi sınıflandırmada SIFT'ten daha iyi performans göstermesine rağmen beklenildiği kadar yüksek performans gösterememektedir. Bunlar temel olarak aşağıdaki eksikliklerden kaynaklanmış olabilir: Yetersiz Detay Algılama: HoG, yerel görüntü gradyanlarının yönelimlerinin histogramını kullanır, bu da ince yüz detaylarını ve mikro ifadeleri yakalama konusunda sınırlı olabilir. Özellikle hafif yüz ifadeleri veya ince mimik değişiklikleri HoG tarafından algılanamayabilir. HoG hiper parametreleri optimize edilerek daha yüksek performansta çalışması sağlanabilir. Öğrenme Modelinin Karmaşıklığı: Kullanılan makine öğrenmesi modelinin karmaşıklığı, özellikle büyük veri setleri ve karmaşık sınıflandırma görevleri için bir sorun oluşturabilir. Modelin eğitimi zaman alabilir ve fazla parametre ayarı gerektirebilir. Makine öğrenmesi modellerinin hiper parametreleri optimize edilerek daha yüksek performansta çalışması sağlanabilir.

**Süreç:**

Devamındaki süreçte Yöntem 2'nin eksikleri kısmında bahsettiğimiz iyileştirmeleri yapılması hedeflenmektedir. HoG ve ML modelleri için farklı hiper parametreler denenerek sonuçların kıyaslaması yapılacaktır. HoG özelinde ayrıntıları daha iyi yakalayabilecek win_size, block_size, block_stride, cell_size ve n_bins hiper parametrelerini optimize edilecektir. Burada denemeyi amaçladığımız temel değişiklik block ve cell size'ları düşürerek detayları daha fazla yakalayacak bir descriptor elde etmek. Ayrıca geliştirilen ML modelleri MLP başta olmak üzere farklı hiper parametrelerde eğitilerek optimize edilecektir