KAHRAMANMARAŞ SÜTÇÜ İMAM ÜNİVERSİTESİ
MÜHENDİSLİK VE MİMARLIK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
MAKİNE ÖĞRENMESİ DERSİ PROJESİ

MAKİNE ÖĞRENMESİ YÖNTEMLERİ İLE İŞARET DİLİ TANIMA

AHMET ÖZBERK, 18110131310
ABDURRAHMAN KARAOĞLU, 18110131307
Dr.Öğr.Üyesi YAVUZ CANBAY

OCAK 2022

MAKİNE ÖĞRENMESİ YÖNTEMLERİ İLE İŞARET DİLİ TANIMA
(Makine Öğrenmesi Projesi Raporu)

Ahmet ÖZBERK

Abdurrahman KARAOĞLU

KAHRAMANMARAŞ SÜTÇÜ İMAM ÜNİVERSİTESİ
MÜHENDİSLİK VE MİMARLIK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
Ocak 2022

ÖZET

Çalışma kapsamında gerçekleştirilen projemizde, makine öğrenmesi yöntemleri kullanılarak, kullanıcının hareketlerinde görüntü işleme ve tahmin etmeyi kullanarak işaretleri tanımayı amaçlamaktadır. Makine öğrenmesi, yapay zeka ve veri analizi gibi bir çok alanda, gerek kütüphaneleri gerekse de kendi sistemi ile öncü olan Python programlama dili kullanılmıştır. Proje için kendimizin ürettiği veri seti kullanılarak modelimiz eğitildi. Daha sonra test aşaması gerçekleştirildi. Kamera karşısında yapılan 3 kelimenin işaret dili hareketleri ile eğitilen CNN + LSTM modellerinde tahminlemesinde deneyler ile görülmüştür ki %80 oranında

Anahtar Kelimeler : İşaret Dili, CNN, LSTM.
Sayfa Adedi : 12
Danışman : Dr.Öğr.Üyesi YAVUZ CANBAY

İÇİNDEKİLER

Sayfa

1. GİRİŞ

Problem tanımı / Konunun tanımı

Proje kapsamında işaret dilini tanıyan bir makine öğrenmesi uygulaması geliştirilmesi amaçlanmaktadır. Geliştirilen projede, popüler Python kütüphaneleri ve CNN algoritması kullanılmıştır. Geliştirilen uygulamada video kamera desteği canlı olarak yapılan işaretlerin tahmin edilmesi ve yazıya çevrilmesi istenmektedir.

Projenin amacı

Projenin amacı, kamera desteği ile belirlenen işaret dillerindeki kelimeleri belirlenen sıralamaya göre makineye eğitim verisi olarak vererek, makinenin öğrenmesini sağlamaktır. Ardından eğitilen modele test verilerini yine kamera desteği ile vererek, test aşamasını gerçekleştirip, başarı oranını değerlendirmek olarak özetlenebilir. Proje kapsamı için gerçekleştirilen araştırma detayında, daha önce yapılan benzer projelerde kullanılan makine öğrenmesi algoritmalarından çıkan başarı sonuçları karşılaştırılmış ve bu oranların olumlu sonuçlarına göre projede kullanılan makine öğrenmesi algoritması seçilmiştir. Yapılan araştırma sonuçları neticesinde CNN algoritması ile gerçekleştirilecek olan projede, gerekli yardımcı Python dili popüler kütüphaneleri kullanılmıştır.

Projenin önemi

Projenin önemi konusunda, günümüz dünyasının evrildiği, ona yön veren popüler ve güçlü konuların günlük hayatta bizlere vereceği desteği, bizleri ne kadar anlayabildiklerini görebiliriz. Geliştirilen kütüphaneler, oluşturulan makine öğrenmesi algoritmaları, ve tüm bunların oluşturduğu güç ile bizlerin günlük hayat problemlerini çözmede, bundan ziyade, artık makinelerin bizleri anlayabilme yetilerinin gelişmesine tanık olmak ve kullanarak deneyimleme gözlemlerini edinebiliriz.

Projenin Kapsamı

Proje makine öğrenmesi dersi kapsamında, makine öğrenmesi konusu üzerinde yapılan çalışmaların gerçekleştirilmesini ele almıştır.

Projenin Özgün Değeri

Proje kapsamında makine öğrenmesi alanları, kullanılan algoritmalar ile ilgili bilgiler edinip nasıl kullanıldığı incelenmiştir. Kazanılan değerli bilgiler içerisinde şüphesiz bu algoritmaların çalışma mantıklarını bir nebze de olsa anlayabilip, onlar üzerinde çalışarak yapılan bu araştırmaların gerçek hayatta karşılıklarını görmek olarak nitelendirilebilirler.

1.2. Literatür Taraması

İşaret dilinden cümlelerin metne çevirme sistemlerinin geliştirilme aşamasında makine öğrenmesi ve derin öğrenme teknikleri ile birçok çalışma gerçekleştirilmiştir. Starner vd. (1998), Saklı Markov Modeli (HMM) makine öğrenme tekniğini kullanarak, Amerikan İşaret Dili cümlelerini metne çevirme çalışmalarında bulunmuştur [1]. Globel ve Assan (1997) ise %94 lük başarı oranı ile Hollanda İşaret Dili için bir sistem geliştirmiştir. Chai vd. (2013.) ise çalışmasında Çin İşaret Dilinden Çince'ye çevirme üzerinde çalışmalar yapmıştır [2].

Tkashashi ve Kishino (1992), Wang vd. (2006), Shanableh ve Assaleh (2011) çalışmalarında donanımsal cihazlar kullanarak işaret dilinden metne çevirme üzerine çalışmalarda bulunmuşlardır. Donanımsal cihazların başında Microsoft firmasının üretmiş olduğu yaygın kullanılan Microsoft Kinect cihazı gelmiştir. Son yıllardaki çalışmalarda ise Kinect cihazı özelliklerini taşıyan Intel firmasının geliştirmiş olduğu Intel RealSense ve el hareketlerini izleyen daha küçük Leap Motion cihazları da kullanılmıştır.

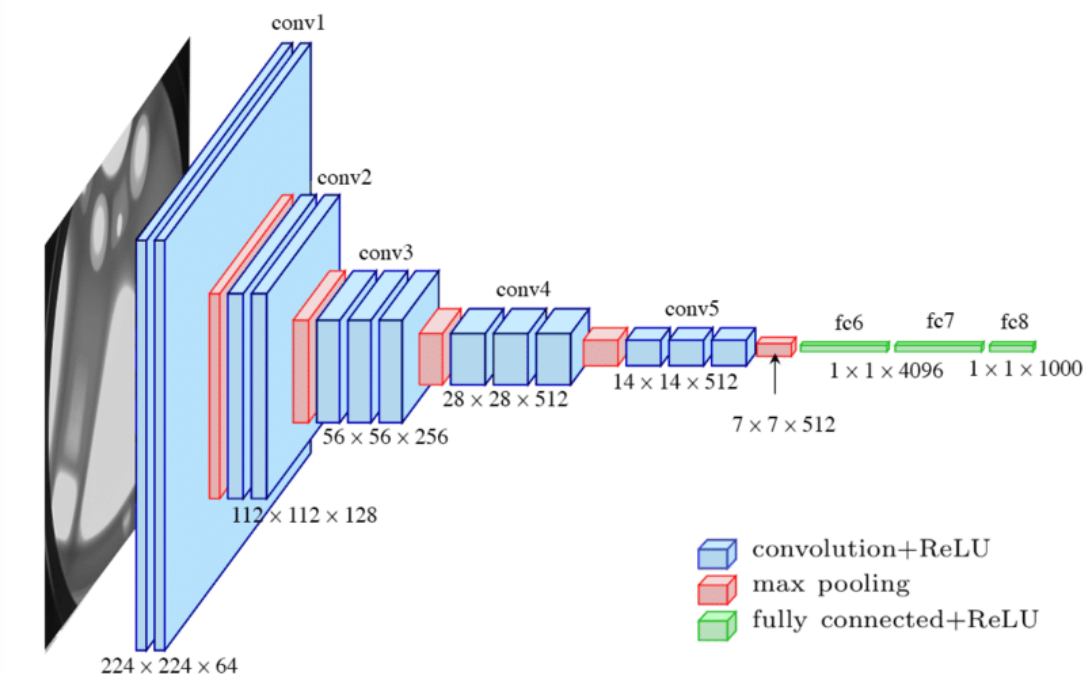
Haberdar ve Albayrak (2005), Işıkdoğan ve Albayrak (2011), Ketenci vd.(2015) ise Türk işaret dili görüntülerinden Türkçe'ye çevirme sistemleri üzerinden çalışmalarda bulunmuşlardır. Türk İşaret Dili tanıma çalışmalarında makine öğrenme methodlarından Saklı Markov Model (HMM), K-En Yakın Komşu (KNN), Destek Vektör Makineleri (Support Vector Machine- SVM) ve Temel Bileşen Analizi (PCA) ağırlıklı olarak kullanılmıştır. Son yıllardaki çalışmalarda ise derin öğrenme tekniklerin Convolusion Neural Network (CNN) modellerin kullanıldığı gözlemlenmiştir [3-5].

Bu çalışmalardan farklı olarak web kamerasından alınan görüntülerin önce CNN ile tahminlendirilip, daha sonra ise LSTM ile yeni bir model oluşturulmuştur. Böylelikle hareketli olan işaretlerinde doğru tahminlenmesi sağlanmıştır.

2. Makine Öğrenmesi Yöntemi

Konvolüsyonel Yapay Ağlar (CNN) ileri beslemeli hayvanların görme merkezinden esinlenerek ortaya çıkan çok katmanlı yapay sinir ağıdır. CNN, görüntüleri girdi olarak içeren problemlerle çalışmak için özel olarak tasarlanmıştır. Facebook ve Google gibi büyük teknoloji şirketleri, yüz tanıma ve görsel arama gibi çeşitli amaçlar için çok sayıda konvolüsyonel katmanı olan derin konvolüsyonel sinir ağları kullanırlar. CNN algoritmaları başta görüntü işleme olmak üzere ses ve doğal dil işleme gibi bir çok alanda etkin bir biçimde kullanılır. CNN'e verilen girişler, her değeri 0 ile 255 arasında değişen bir piksel değerleri dizisidir. Örneğin giriş 35x35 boyutlarında bir görüntü ise dizi, 35x35x3 biçiminde 3 boyutlu bir matris oluşturacaktır. Bir görüntüdeki her piksel 3 değerle temsil edilir. Bu 3 değer Red, Green ve Blue (RGB), yani kırmızı, yeşil ve mavi yoğunluklarını temsil eder. Görüntü sınıflandırma durumunda CNN algoritmasının işi, bu görüntüyü, yani bir piksel değerleri dizisini, bir girdi olarak almak ve belirli bir sınıfa ait olma ihtimallerini çıkarmaktadır. Tahmin problemlerinde ise, model girdi olarak piksel değerlerini alır ve karşılık gelen çıktı değerlerini tahmin eder.

Konvolüsyon katmanı CNN'in ilk katmanıdır. Bu katman verilen bir girdideki düşük seviyenin yanı sıra yüksek seviye karmaşık özelliklerin tanımlanmasından sorumludur. Konvonüsyonel aşaması ağın daha önce öğrendiklerine bakarak giriş sinyallerini etkilemeye çalıştığı bölümdür. Aktivasyon katmanı CNN mimarisinin sonuna veya arasına koyulan bir katmandır. Aktivasyon katmanı sinyalin bir katmandan diğerine nasıl aktarıldığını kontrol eder. ReLU fonksiyonu günümüzde sinir ağlarında en yaygın kullanılan aktifleştirme fonksiyonudur. ReLU'nun diğer fonksiyonlara göre avantajı, tüm nöronları aynı anda aktifleştirmemesidir.



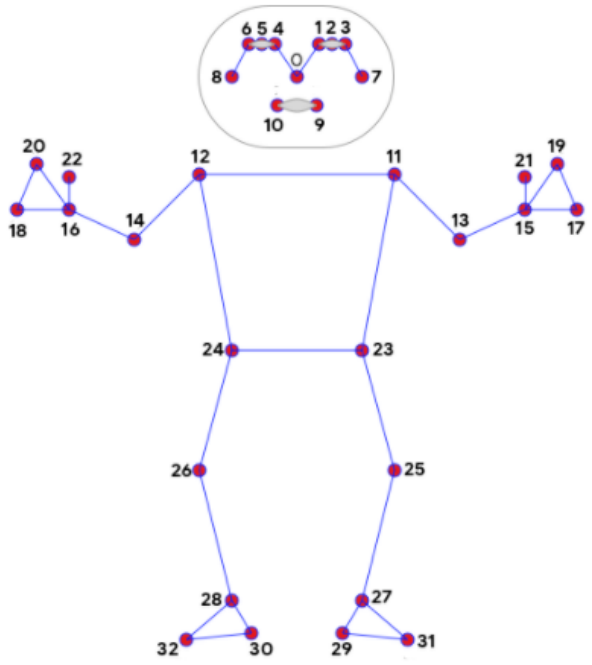
Örnek Cnn Katmanları Mimarisi

2.2. MediaPipe Nedir

MediaPipe, Google tarafından oluşturulan makine öğrenmesi çözümleri oluşturmak için kullandığımız açık kaynak kodlu bir frameworktür. MediaPipe modüler yapısı sayesinde bize kullanımı kolay ve hızlı uygulanabilir bir yapı sunuyor. Bir çok platformda kullanıyor olması da projemiz için avantaj sağlıyor. MediaPipe kütüphanesi ile vücuttaki bölgeleri işaretleyebilir, onlardan modeller oluşturabilir. Proje içerisinde MediaPipe ile yüz, kollar ve eklem yerlerini belirtmek ve modellemek için kullanılmıştır. El, kol, bilek ve kafa hareketlerini izleyerek tahmin etme konusunda bize en büyük desteği bu framework sağlayacak.
MediaPipe'ın Bize Sunduğu Olanaklar

- İnsan poz algılama ve takibi konusunda, RGB video karelerinden minimum 25 adet 2D üst vücut yer işareti çıkarımı yapan, yüksek kaliteli insan vücudu poz takibi
- Face Mesh 468 ile çoklu yüz desteği ile 3D yüz modelleme
- El izleme, yüksek performanslı avuç içi algılama ve el işareti modeline dayalı, çoklu el desteğiyle 3D olarak 21 nokta üzerinden takip
- El başına 21, yüz için 468 nokta ile eş zamanlı ve performanslı izleme
- Videodaki nesneleri tek bir ardışık düzende algılama ve izleme
- İris göz bebeği ve göz çervresini izle

gibi bir bir çok özelliği kullanılabilir durumdadır.

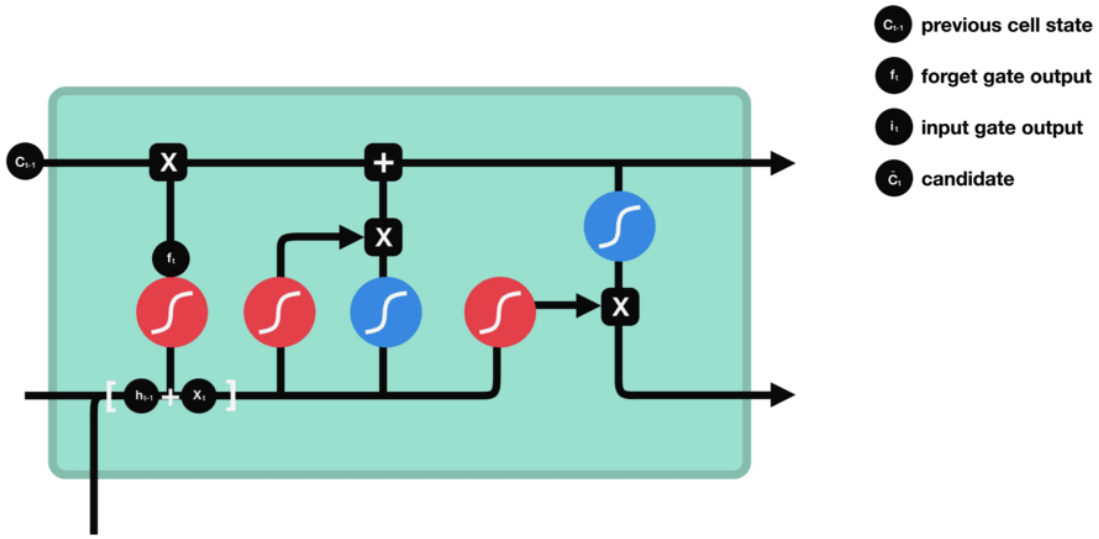


0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

MediaPipe Model Katmanları

2.3.Uzun Kısa Süreli Bellek Ünitesi(LTSM)

Alman araştırmacılar tarafından kaybolan gradyan sorununa bir çözüm olarak önerilmiştir. LSTM birimleri, tekrarlayan bir sinir ağının (RNN) katmanları için bir yapı birimidir. Birimlerden oluşan RNN'ye LTSM ağı denir. Ortak bir birimi bir hücreden, bir giriş geçidinden, bir çıkış geçidinden ve bir unutma geçidinden oluşur. Hücre, rastgele zaman aralıklarındaki değerleri hatırlamaktan sorumludur. Üç geçitten her biri, çok katmanlı bir sinir ağında olduğu gibi geleneksel bir yapay nöron olarak düşünülebilir.



Basit bir LTSM yapısı

3. Kullanılan Veri Seti

Projede halihazırda olan, video hali dışında, resimlerle oluşturulmuş, sadece harflere yönelik veri setleri Kaggle gibi popüler veri setleri bulunan internet sitelerinde bulunabiliyor. Bu proje kapsamında, kelimeye yönelik harfler baz alınmadan kullanılacak video formatlı verilere ihtiyaç duyulduğu için, canlı kamera desteği ile veri setini oluşturma işlemini kendimiz gerçekleştirdik. Veri seti oluşturulurken belirlenen kelimeler çerçevesinde (selam, teşekkürler, seni seviyorum) her kelime katarından 30'ar adet, belirli aralıklarla oluşturulmuştur.

4. Veri Setinde Ön İşleme Adımları ve Analizi

Veri setinin oluşturulması aşamasında boş değer eklenmediği, verileri belirli index değerlerine göre belirli sıralama adımlarında üretilmesi sağlanmıştır. Veri setinin oluşturulması adımında, mediapipe kullanıldığı için, yüz, sağ-sol el ve eklem noktaları baz alınarak OpenCv ile canlı kamera üzerinden, belirlenen indexteki, Örneğin ilk girişi sağlanacak veri, "selam" kelimesi katarıdır. Bu kelimeyi oluşturma aşamasında baş ve kolların hareketlerinden, ilgili işaret diline ait veri oluşturulur.

5. Eğitim Modelinin Oluşturulması

Modeli eğitme aşamasında, LTSM sinir ağı oluşturulur. Oluşturulan sinir ağı eğitilir.

6. Tahminlerin Oluşturulması ve Kayıt Edilmesi

Oluşturulan model üzerinden tahminler yapılır ve ".h5" uzantılı dosyaya kayıt edilir.

7. Modelin Test Edilmesi

Son aşamada oluşturulan modelimiz test edilmeye başlanır. Test edilme aşamasında, kamera yardımıyla kullanıcıdan alınan görüntü, uygun model değişkenine eşleştirilir ve tahmin sonucu işaret hangi kelime katarına aitse ekranda o kelime katarı yazdırılır ve boyanır.

8. Uygulamanın Gerçekleştirilmesi

8.2. Gerekli Yüklemelerin Yapılması ve Veri Setinin Oluşturulması

4. Koleksiyon için Kurulum Klasörleri

```
DATA_PATH = os.path.join('MP_Data')

# Tespit etmeye çalıştığımız eylemler
actions = np.array(['hello', 'thanks', 'iloveyou'])

# Otuz video değerinde veri
no_sequences = 30

# Videolar 30 kare uzunluğunda olacak
sequence_length = 30
```

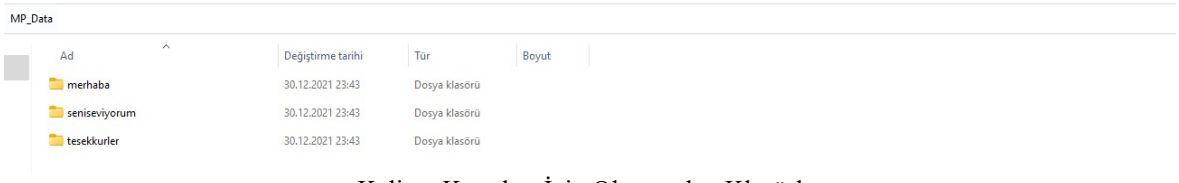
```
for action in actions:
    for sequence in range(no_sequences):
        try:
            os.makedirs(os.path.join(DATA_PATH, action, str(sequence)))
        except:
            pass
```

8.3. Video Kameradan Veri Modelinin Oluşturulması

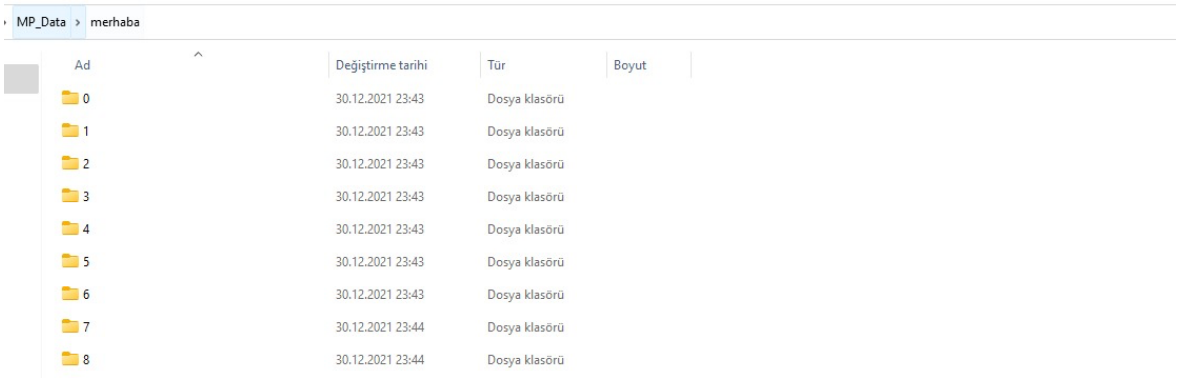
5. Eğitim ve Test için Anahtar Nokta Değerlerini Toplayın

```
cap = cv2.VideoCapture(0)

# min_detection_confidence: Kişi tespit modelinden tespitin başarılı olarak
#                           kabul edilmesi gereken minimum güven değerini
#                           belirtmek için kullanılır. [0.0 - 1.0] içinde bir değer belirtebilir.
#                           Varsayılan değer 0.5'tir.

with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:

    # Eylemler arasında geçiş yapın
    for action in actions:
        # Videolar olarak adlandırılan diziler arasında döngü yapın
        for sequence in range(no_sequences):
           # Video uzunluğu, yani dizi uzunluğu boyunca döngü
            for frame_num in range(sequence_length):

                # Kamerayı oku
                ret, frame = cap.read()

                # Tespitler yap
                image, results = mediapipe_detection(frame, holistic)

                # Yer işaretleri çizin
                draw_styled_landmarks(image, results)

                if frame_num == 0:
                    cv2.putText(image, 'KOLEKSİYONA BASLIYOR', (120,200),
                                cv2.FONT_HERSHEY_SIMPLEX, 1, (0,255, 0), 4, cv2.LINE_AA)

                    cv2.putText(image, '{} kelimesinin {} ornegi isleniyor.'.format(action, sequence), (15,12),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE_AA)
                   # Ekrana göster
                    cv2.imshow('İsaret Dili Tanima', image)
                    cv2.waitKey(500)
                else:
                    cv2.putText(image, '{} kelimesinin {} ornegi isleniyor.'.format(action, sequence), (15,12),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE_AA)
                    # Ekrana göster
                    cv2.imshow('İsaret Dili Tanima', image)

              # YENİ Önemli noktaları dışa aktar
                keypoints = extract_keypoints(results)
                npy_path = os.path.join(DATA_PATH, action, str(sequence), str(frame_num))
                np.save(npy_path, keypoints)


                if cv2.waitKey(10) & 0xFF == ord('q'):
                    break

    cap.release()
    cv2.destroyAllWindows()
```

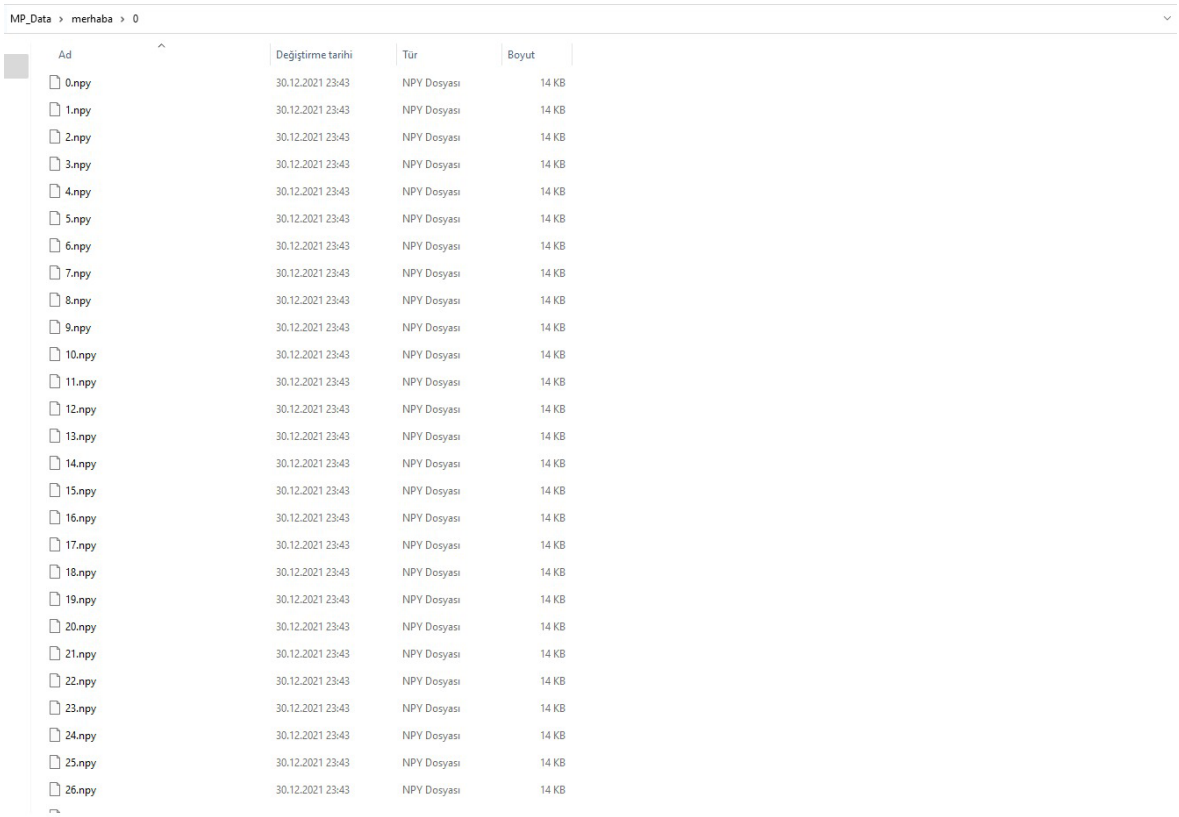
Kameradan oluşturulan verilerin dosyaları oluşturma işlemi gerçekleştiriliyor. Her kelime katarından 30 adet dosya ve veri oluşturuluyor.



Kelime Katarları İçin Oluşturulan Klasörler



Her Katar İçinde Oluşturulan Klasör Yapısı



Oluşturulan Video Dosyalarına Ait Veriler

8.4. Veriyi Ön İşleme Alma ve Özelliklerin Oluşturulması

6. Verileri Ön İşleme Alın ve Etiketler ve Özellikler Oluşturun

```
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
```

```
label_map = {label:num for num, label in enumerate(actions)}
```

```
label_map
```

```
{'hello': 0, 'thanks': 1, 'iloveyou': 2}
```

```
sequences, labels = [], []
for action in actions:
    for sequence in np.array(os.listdir(os.path.join(DATA_PATH, action))).astype(int):
        window = []
        for frame_num in range(sequence_length):
            res = np.load(os.path.join(DATA_PATH, action, str(sequence), "{}.npy".format(frame_num)))
            window.append(res)
        sequences.append(window)
        labels.append(label_map[action])
```

Oluşturulan veri setinin test ve eğitim olarak ayrılması işlemi gerçekleştirilir.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.05)
```

8.5. LTSM Sinir Ağının Oluşturulması ve Modelin Eğitilmesi

7. LSTM Sinir Ağı Oluşturun ve Eğitin

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.callbacks import TensorBoard
```

```
log_dir = os.path.join('Logs')
tb_callback = TensorBoard(log_dir=log_dir)
```

```
model = Sequential()
model.add(LSTM(64, return_sequences=True, activation='relu', input_shape=(30,1662)))
model.add(LSTM(128, return_sequences=True, activation='relu'))
model.add(LSTM(64, return_sequences=False, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(actions.shape[0], activation='softmax'))
```

```
model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['categorical_accuracy'])
```

```
model.fit(X_train, y_train, epochs=2000, callbacks=[tb_callback])
```

```
Epoch 174/2000
3/3 [==============================] - 0s 129ms/step - loss: 0.5102 - categorical_accuracy: 0.7182
Epoch 175/2000
3/3 [==============================] - 0s 133ms/step - loss: 0.4936 - categorical_accuracy: 0.8552
Epoch 176/2000
3/3 [==============================] - 0s 139ms/step - loss: 0.4222 - categorical_accuracy: 0.9882
Epoch 177/2000
3/3 [==============================] - 0s 139ms/step - loss: 0.3878 - categorical_accuracy: 0.8864
Epoch 178/2000
3/3 [==============================] - 0s 117ms/step - loss: 0.3395 - categorical_accuracy: 0.9217
Epoch 179/2000
3/3 [==============================] - 0s 134ms/step - loss: 0.4783 - categorical_accuracy: 0.9354
Epoch 180/2000
3/3 [==============================] - 0s 133ms/step - loss: 0.2664 - categorical_accuracy: 0.9374
Epoch 181/2000
3/3 [==============================] - 0s 125ms/step - loss: 0.2294 - categorical_accuracy: 0.9589
Epoch 182/2000
3/3 [==============================] - 0s 136ms/step - loss: 0.1975 - categorical_accuracy: 0.9784
Epoch 183/2000
3/3 [==============================] - 0s 129ms/step - loss: 0.2506 - categorical_accuracy: 0.8885
```

Daha sonra eğitilen verinin özetini summary parametresi ile gösteriyoruz.

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 30, 64)            442112
_________________________________________________________________
lstm_1 (LSTM)                (None, 30, 128)           98816
_________________________________________________________________
lstm_2 (LSTM)                (None, 64)                49408
_________________________________________________________________
dense (Dense)                (None, 64)                4160
_________________________________________________________________
dense_1 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 99
=================================================================
Total params: 596,675
Trainable params: 596,675
Non-trainable params: 0
_________________________________________________________________
```

İşlem bittikten sonra tahmin yapılır ve model dosyası ".h5" uzantılı şekilde kaydedilir.

8.6. Karışıklık Matrisi ve Doğruluğu Kullanarak Projenin Değerlendirilmesi

10. Karışıklık Matrisi ve Doğruluğu Kullanarak Değerlendirme

```
from sklearn.metrics import multilabel_confusion_matrix, accuracy_score
```

```
yhat = model.predict(X_test)
```

```
ytrue = np.argmax(y_test, axis=1).tolist()
yhat = np.argmax(yhat, axis=1).tolist()
```

```
multilabel_confusion_matrix(ytrue, yhat)
```

```
array([[[4, 0],
        [1, 0]],

       [[0, 1],
        [0, 4]]], dtype=int64)
```

```
accuracy_score(ytrue, yhat)
```

```
0.8
```

Sonuç aşamasında doğruluk oranımız %80 gibi bir oran çıkıyor.

8.7. Canlı Test Aşaması

11. Gerçek Zamanlı Test

```
from scipy import stats
```

```
colors = [(245,117,16), (117,245,16), (16,117,245)]
def prob_viz(res, actions, input_frame, colors):
    output_frame = input_frame.copy()
    for num, prob in enumerate(res):
        cv2.rectangle(output_frame, (0,60+num*40), (int(prob*100), 90+num*40), colors[num], -1)
        cv2.putText(output_frame, actions[num], (0, 85+num*40), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255), 2, cv2.LINE_AA)

    return output_frame
```

```
plt.figure(figsize=(18,18))
plt.imshow(prob_viz(res, actions, image, colors))
```

```
<matplotlib.image.AxesImage at 0x1a448257970>
```

KAYNAKLAR

1. Starner T., Weaver J., Pentland A., "Real-time American Sign Language Recognition using Desk and Wearable Computer based Video", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, pp. 1371–1375, 1998.
2. Chai X., Li G., Chen X., Zhou M., Wu G., Li H., "VisualComm: A Tool to Support Communication Between Deaf and Hearing Persons with the Kinect", 15th International ACM SIGACCESS Conference on Computers and Accessibility, p. 76, 2013.
3. Haberdar H., 2005, Saklı Markov Modelleri Kullanılarak Görüntüden Gerçek Zamanlı Türk İşaret Dili Tanıma Sistemi, Bilgisayar Mühendisliği, Yüksek Lisans Tezi, Yıldız Teknik Üniversitesi.N. El-Makky et al., Sentiment analysis of colloquial Arabic tweets, 2015.
4. Işikdoğan F., Albayrak S., 2011, June, Automatic recognition of Turkish fingerspelling, In Innovations in Intelligent Systems and Applications (INISTA), 2011 International Symposium on (pp. 264-267), IEEE.
5. Ketenci S., Kayikçioğlu T., Gangal A., 2015, May, Recognition of sign language numbers via electromyography signals, In Signal Processing and Communications Applications Conference (SIU), 2015 23th (pp. 2593-2596), IEEE.


T.C.
KAHRAMANMARAŞ
SÜTÇÜ İMAM ÜNİVERSİTESİ
1992