Turkish Journal of Science & Technology
17(2), 251-260, 2022

Research Paper
https://doi.org/10.55525/tjst.1073116

Deep Learning Based Recognition of Turkish Sign Language Letters with Unique Data Set

Mustafa KAYA[1*], Fatih BANKUR[2]
[1.2] Department of Digital Forensics Engineering, Fırat University, Elazığ, Turkey
[1*] mkaya@firat.edu.tr, [2] fth.bnkr23@gmail.com

(Geliş/Received: 14/02/2022; Kabul/Accepted: 01/05/2022)

Deep Learning Based Recognition of Turkish Sign Language Letters with Unique Data Set

Abstract: With its development, artificial intelligence has formed the basis for many studies aimed at facilitating people's lives. More successful results have been tried to be obtained with the increasing data and developing equipment in these studies. It is seen that these developments in artificial intelligence are reflected in the studies related to sign language conversion.
In this study, a data set belonging to the letters in the Turkish Sign Language Alphabet was created, and the classification process was carried out with both the deep learning model we created and VGG16, Inceptionv3, Resnet, and Mobilnet models, which are frequently used in image classification. In addition, an open-source data set containing the letters in the American Sign Language Alphabet was organized similar to the data set containing the letters in the Turkish Sign Language Alphabet we created, and Deep Learning models were used to classify the letters in the American Sign Language Alphabet by using this data set. Performance evaluations of the classifications made by Deep Learning Models using both data sets were made. With this study, the results obtained from training Deep Learning methods with different data sets were compared. In addition, it is thought that the study will be useful in determining both the data set and the deep learning method to be used for the studies on the recognition of Sign Language Letters.
Key words: Turkish Sign Language Alphabet, Deep Learning, SSD Mobilenet, EfficientNet, YOLOv5.

Özgün Veri Seti ile Derin Öğrenme Temelli Türk İşaret Dili Harflerinin Tanınması

Öz: Derin Öğrenme kavramı, artan veri miktarı ve geliştirilen kapsamlı donanımlarla birlikte önem kazanmaya başlamıştır. Derin Öğrenme yöntemleri birçok farklı alanda kullanılmış ve başarılı sonuçlar alınmıştır. Bu alanlar doğal dil işleme, görüntü işleme, sanal veri üretme, otonom araçlar, ses tanıma ve sağlık, sanayi ve savunma sanayi gibi birçok alanı kapsamaktadır. İnsan hayatını kolaylaştırma adına birçok alanda kullanılmakta olan derin öğrenme yöntemleri yapılan bu çalışma ile Türk İşaret Dili (TİD) Harflerinin tanınması amacı ile kullanılmıştır.

Bu çalışmada TİD harflerine ait özgün bir veri seti oluşturulmuş, oluşturulan bu veri seti içerisindeki verileri çeşitlendirmek ve arttırmak için faklı veri çoğaltma yöntemleri de kullanılmıştır. Sonuç olarak toplamda 10.000 adet resimden oluşan özgün bir veri seti elde edilmiştir. Hazırlanan özgün veri seti, TİD harflerinin hızlı bir şekilde tanınması amacıyla tek aşamalı nesne tespiti modelleri olan SSD Mobilenet, EfficientNet ve YOLO modelleri ile eğitime tabi tutulmuştur. Yakın eğitim süreleri sonucunda YOLOv5 modelinin diğer nesne tespiti modellerine oranla daha hızlı ve daha doğru sonuçlar verdiği gözlemlenmiştir. Bu sebeple hazırlanan özgün veri seti ile YOLOv5 modeli daha uzun süreli olarak eğitilmiş ve sonuç olarak YOLOv5 modeli ile TİD harflerinin tanınması %92,3 oranında başarı ile gerçekleştirilmiştir. TİD harflerinin hızlı bir şekilde tanınmasını sağlayan bu çalışma daha sonra yapılacak olan gerçek zamanlı TİD çevirmen uygulamalarına hem veri seti, hem de kullanılacak derin öğrenme yönteminin belirlenmesi açısından fayda sağlayacaktır.
Anahtar kelimeler: Türk İşaret Dili Alfabesi, Derin Öğrenme, SSD Mobilenet, EfficientNet, YOLOv5.

1. Introduction

Speech is the most important way of human communication. It is seen that people who are deaf, mute or have difficulty in speaking mostly communicate with TSL. With the recent developments in the field of object recognition, TSL signs can also be recognized as objects, which constitutes the source of motivation for this study.

* Corresponding author: mkaya@firat.edu.tr. ORCID Number of authors: [1] 0000-0002-0160-4469, [2] 0000-0002-2455-1195

Since 1990, signs and data in sign language have been started to be processed. In one of the first studies conducted for this purpose, Charahpayan and Marble developed a computer vision-based system based on the use of the speed of the hand for the creation of ASL (American Sign Language) [1]. Takahashi and Kishimo tried to classify and recognize the Japanese Sign Language alphabet over 46 samples taken through the data glove [2]. In 1995, Waldron and Kim performed the recognition of 14 ASL words using the artificial neural network method, using hand shape, hand position, and movement [3]. Allen et al. performed a spelling system for ASL. This system could recognize 24 of 26 hand shapes [4]. In 2004, Wang, ÖZ et al. implemented an ASL hand shape recognition system. Data gloves and motion tracking are used in the system, and artificial neural networks and HMM (Hidden Markov Model) are used in classification [5]. In the study conducted in 2018, after the hands in the images were detected using skin color discrimination, it was observed that real-time sign recognition was performed using deep learning methods with an accuracy rate of 98.05% [6]. It has been observed that there are studies on many foreign sign languages.

When we overview the studies conducted by Haberdar and Albayrak in 2005 on the TSL, a classification was conducted with the HMM and a success rate of 95.7% was achieved [7]. In the study conducted by Işıkdoğan and Albayrak in 2011, achieved a 99.39% classification success by using feature extraction with Histograms of Oriented Gradients (HOG), and then reducing dimensions Principal Components Analysis (PCA) has been applied to the extracted features [8]. In 2013, the recognition of motion pictures was carried out by Albayrak and Memiş using discrete cosine transformation of RGB video data and data received with Kinect device, and 90% success was achieved [9]. In the study conducted by Demircioğlu et al. in 2016, basic hand movements selected from Turkish sign language were recognized with a 99.03% success rate using the Leap Motion device[10]. In 2017, the classification process of the data obtained using the data glove on TSL was carried out[11]. In the study carried out by Fırat and Uğurlu in 2018, it is seen that some of the words in the Turkish sign language are recognized from the images obtained by using the Kinect device [12]. In the study presented by Çelik and Odabaş in August 2020, it was observed that a 97% success rate was achieved in estimating 10 numbers and 29 letters by using CNN + LSTM models trained with obtained hand images in front of the camera [13].

As a result of the literature review, it is seen that systems with extra equipment such as data glove and Kinect sensor used in the studies have high-performance rates, but it is evaluated that the systems to be created will not be sufficient in terms of their contribution to daily life due to the cost. Although it is seen that the 97% success rate of the work done against the normal camera will make more contribution to daily life, there are more usable and easier systems to perform. It can be observed that determining the skin color and hands in the study will not achieve the desired success as a result of the change in people's clothes.

In this study, the process of recognizing the letters in the TSL alphabet in front of a normal camera was carried out using deep learning models. Unlike other studies, the letters of the TSL alphabet created in front of a normal camera are recognized without any preprocessing. In addition, SSD Mobilenet, EfficientNet, and YOLO models, which are one-stage object detection models used in object recognition, were trained with the original data set. It is observed that the performance values obtained from the object detection models as a result of the training will be beneficial in the selection of the model to be used in further studies on object detection.

This study consists of 6 parts. In the introduction, general information about the study and studies on the recognition of TSL were presented. In Chapter 2, general information about deep learning as well as content about single and two-stage models used in object recognition were included. In Chapter 3, information about the data set created by us for the letters TSL was presented. In Chapter 4, the original data set containing the letters TSL was trained with SSD Mobilenet, EfficientNet and YOLO models, one of the one-stage models used in object detection, and the performance values obtained as a result of the training were compared. With the YOLOv5 model, which has better performance values than the other models trained in Chapter 5, the training period was increased, and the training results were indicated. Chapter 6 includes conclusions and evaluations.

2. Deep Learning and Single Stage Object Detection Models

2.1. Deep learning

Artificial intelligence systems are systems that observe the existing situation and process these observations in line with the predetermined parameters and react to the desired situations to solve the problem. The concept of machine learning has emerged as artificial intelligence allows it to learn from the data in a certain system. Machine learning algorithms determine collecting data from many data sources, transforming these data sets, and processing

according to the results. With the development of these algorithms, artificial neural network models have emerged with logical software that imitates the working mechanism of the human brain and can derive new information by learning, remembering, and generalizing.

With the increasing importance of large amounts of data, deep neural network models have been developed to facilitate feature extraction. Deep learning is a type of artificial neural network consisting of many specialized hidden layers and processing elements, eliminating the cost of feature extraction in classification problems by using large amounts of unsupervised data.

Deep learning methods have gained more importance with the development of graphics cards. Deep learning methods are used in many areas such as natural language processing, image processing, virtual data generation, autonomous vehicles, voice recognition, health, industry, and defense industry. In this study, object recognition, which is one of the fields of image processing, has been performed. Two different methods were used while performing the object recognition process. It has been evaluated that it would be appropriate to use one-stage object detection models, where more efficient results in terms of speed were obtained from these methods, which were distinguished as two-stage and one-stage object detection models.

2.2. One-stage object detection models

The importance of faster detection of objects in object detection applications is increasing day by day for realtime implementation. For this reason, one-stage object detection models, which detect objects much faster than two-stage object detection models, have been developed. In one-stage object recognition models, unlike two-stage object detection models, the whole picture is handled as a whole and object detection is made in this way. The general structure of one-stage object detection applications consists of the backbone, which is called the spine, the neck, which is called the neck between the spine and the head, and the head, which is defined as the head. The process, which starts with the given input image, reaches its final goal with the determination of the class of the object and the coordinates in the head part. One-stage object detection models include models such as SSD, EfficientNet, and YOLO. The general structure of one-stage object detection models is shown in Figure1. [14].

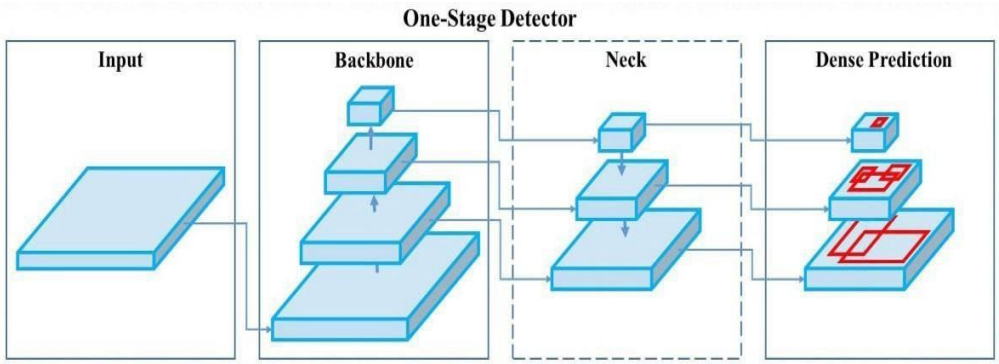

Figure 1. The General structure of one-stage object detection [14]

2.2.1. Single Shot Multibox Detector (SSD)

The SSD model has emphasized that a regression problem to be solved to perform the object detection process would be useful. It has been argued that it would be beneficial not to make proposals for the region containing the object, but instead to consider the picture as a whole. In order to train the SSD model, besides the input images, the bounding box information indicating the locations of the objects is needed. The image given as input is divided into matrix cells by using matrices arranged in different sizes from each other. The maps of the features contain these matrices as a part of the content. Object detection is performed using bounding boxes with different properties from these feature maps.

2.2.2. EfficientNet

The EfficientNet model has two different features compared to other models. The first is the use of a twoway feature pyramid network that offers learnable weights to learn how different input images can make a difference in the model. Secondly, the developed model uses a combined scaling method in the form of resolution, depth, and width as a class prediction network and as a backbone network.

2.2.3. Yolo Models

The biggest advantage of YOLO models over other models is that they can detect objects faster. In the developed YOLO models, the class of the object and its location information are obtained by performing only one operation on the input picture, without any suggestion of a place where the object to be detected can be found. While YOLO extracts this information, it tackles a regression problem and tries to solve it. This regression process combines the coordinates of a bounding box containing the object from the pixels in the picture and the probabilistic classification of which class the object will be included in. The convolutional neural network created with the YOLO model allows us to find the class probabilities of many different objects and the bounding boxes of these objects at the same time in the picture content given as inputs. In this way, the YOLO model, which performs both operations together, stands out a little more than other models with its speed as well as fewer errors in background extraction and more generalizable inferences.

The latest developed version of the YOLO model is the YOLOv5 model. It includes different models such as Yolov5s, Yolov5m, Yolov5l, and Yolov5x. The YOLOv5 model is a model created by implementing the YOLOv4 model developed in the Darknet library on the PyTorch Library. The biggest difference of the Yolov5 model is that it uses the PyTorch structure, unlike the Darknet structure used in previous models. The PyTorch ecosystem is easier to deploy and simpler to support. One of its newest features is that it is simpler to deploy, especially to mobile devices. The weight files created for the YOLOv5 model are quite small compared to other YOLO models. The file includes weights for the YOLOv5 is 27 megabytes, while the weight file for the YOLOv4 model is 244 megabytes. The YOLOv5 model has shown that it can be applied to embedded devices more easily with this feature.

3. Material and Method

3.1. The development process of the proposed model

The model used in the application to be developed for the recognition of the letters in the TSL Alphabet must be satisfactory in terms of both fast and accurate results. The block diagram regarding the development process and performance evaluations of the models is given in Figure 2 below.

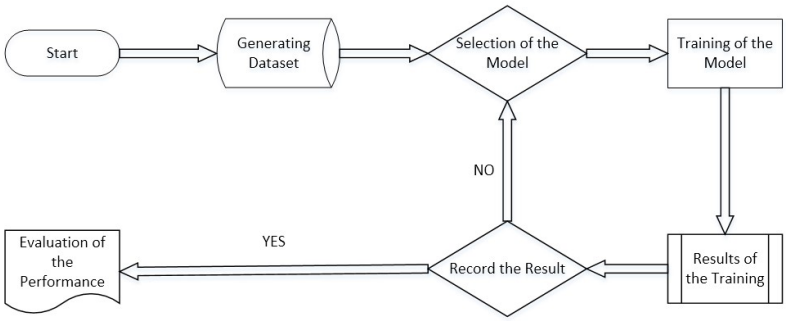

Figure 2. The development process of the models

After the dataset is created and the model is selected, the data is randomly divided into 3 separate groups for training, validation, and test data. Then, the training process is started for the selected model to learn. As a result of the training, model and parameter selection and improvements continued until the system learned enough and the results were satisfactory. The performance evaluation of the system obtained the best performance values is presented in the results section.

3.2. Dataset

TSL is a new and untouched field in terms of applying computer science and artificial intelligence models. With the development of deep learning methods, datasets have started to be created for the recognition of TSL. However, it is seen that the datasets prepared in the studies examined are mostly not open to the public, it has been seen that the datasets are specific to the study. For this reason, it was decided to create a new dataset. Therefore, it was decided to generate a new dataset. While generating the new dataset, it was aimed to prepare a dataset consisting of 29 classes, corresponding to 29 letters in our alphabet. The original dataset creation process consists of four steps. In the first step, videos that can be viewed publicly and that contain the TSL alphabet were detected and downloaded. In the second step, a total of 1000 416x416 pictures were obtained by adding the videos we took to the letters where the downloaded videos were insufficient. The images obtained in the third step are tagged with www.roboflow.com.

Since there is no Turkish character support, the letter 'Ç' is labeled as 'CCC', 'I' as 'III', 'Ğ' as 'GGG', 'Ö' as 'OOO', 'Ş' as 'SSS' and ' The letter 'Ü' is labeled as 'UUU'. In the last step, 1000 tagged images were subjected to data duplication by keeping the same proportions, and a total of 10000 tagged original datasets were obtained with the brightness, contrast, color, saturation, noise, and geometric changes. Some images of the original dataset prepared are shown in Figure 3 and Figure 4 below.

Figure 3. Some TSL image examples before augmentation

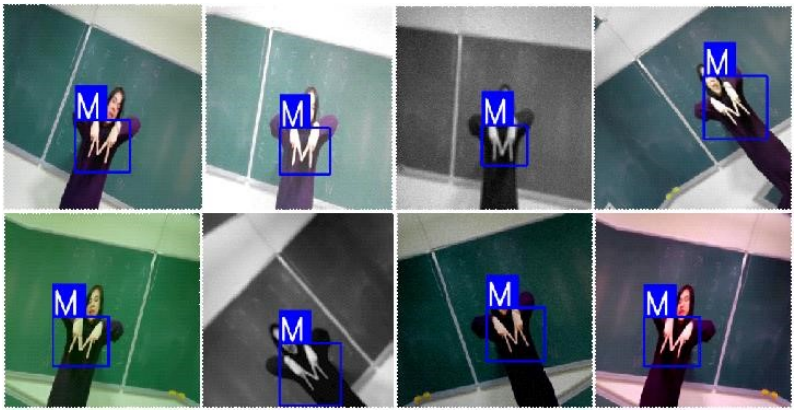

Figure 4. A sample augmentation of "M" letters

The data set, which was preprocessed, labeled, and reproduced, was randomly grouped as 70% training, 20% validation, 10% testing, and the training process was started.

At this stage of the study, the original data set containing the letters TSL was trained with SSD Mobilenet, EfficientNet, and YOLOv5s models, which are one-stage object detection models. The training process of all models was carried out in Google Colab environment using Tesla T4 GPU. Loss values were calculated for the models. Classification/cls_loss loss value was calculated to determine whether the detected sign is in the correct class. In order to determine whether the position and size of the frame (BBox) are correct, the Loss value of Localization (box_loss), the loss value of Confidence (Regularization/obj_loss) related to the confidence value of the box performing the detection operation, and finally the Total (Total) loss values were calculated. Graphs were drawn regarding the variation of these calculated values. The parameters were updated for each model, taking into account the parameter values with the highest performance.

3.3. SSD Mobilenet

The SSD Mobilenet model was designed with the Tensorflow API, which includes many trained network structures in the field of object recognition and classification. With the arrangement made, files in the form of record and .pbtxt were created. 7000 trained and 2000 valid images are set as input data to the designed model. In the model, the input images were resized, the size of the input data was resized to 640x640, the Batch Size was determined as 14 and the training process was provided in 5000 steps. Total training time is 2 hours and 5 minutes. The total loss value calculated as a result of the training was calculated as 0.84 and is shown in Figure 5 below along with the other loss values.

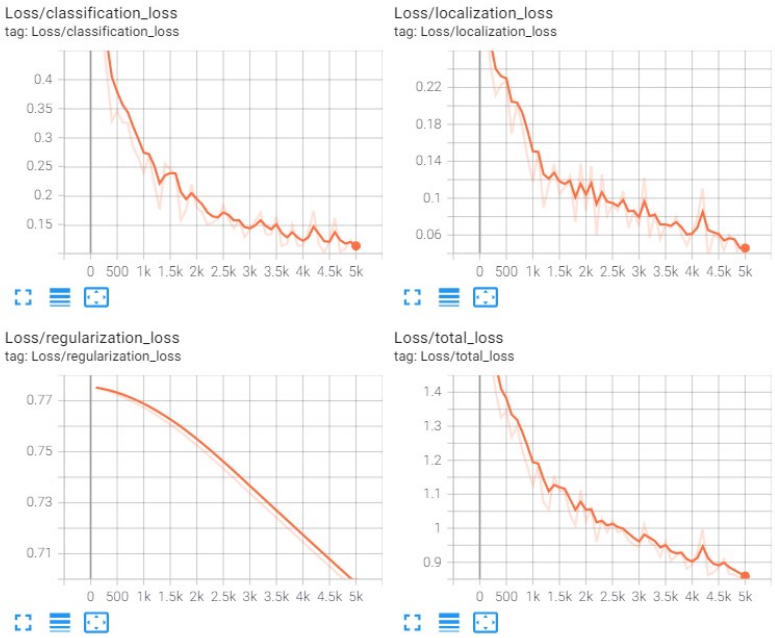

Figure 5. SSD Mobilenet loss values

3.4. EfficientNet

In this model, which was created using Tensorflow API, the input images were resized, the size of the input data was reduced to 512x512 by the model, the Batch Size was determined as 16 and the training process was provided in 5000 steps. The training process was completed in 2 hours. The total loss value calculated as a result of the training was calculated as 0.78 and is shown separately in Figure 6 below along with the other loss values.

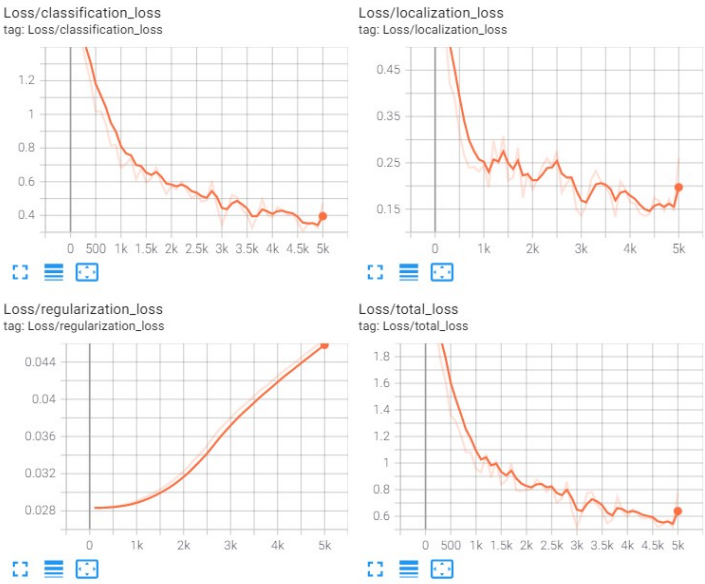

Figure 6. EfficientNet loss values

3.5. YOLOv5s

The YOLOv5s model is a one-stage object detection model using the Pytorch library. In this model, input images are resized to 416x416, Batch Size is set to 16 and, the training process is defined as 100 epochs. The total training was completed in 1 hour 59 minutes and 47 seconds. The total loss value was calculated as 0.04 at the end of the training and results are shown separately in Figure 7 below along with the other loss values.

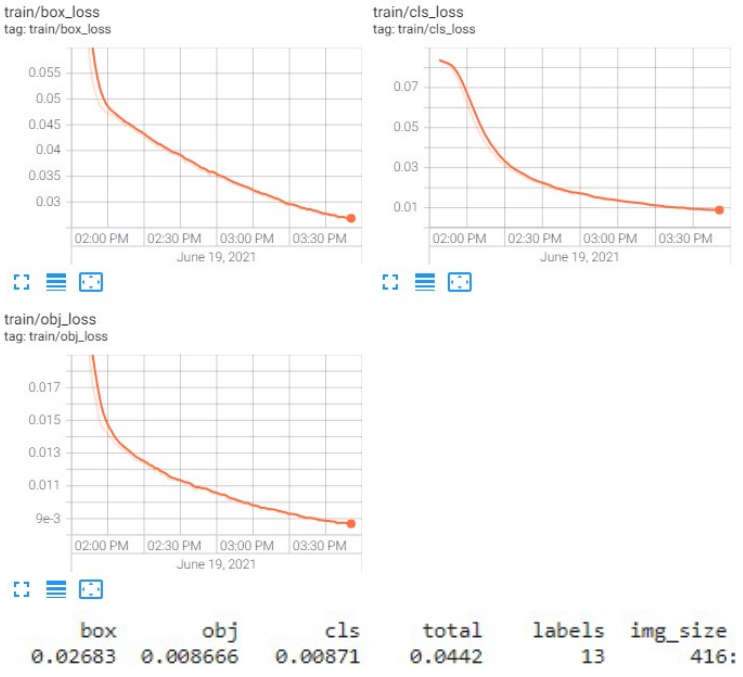

Figure 7. YOLOv5s loss values

Considering that all the developed models take approximately 2 hours and the input dimensions are close to each other, it is seen that the YOLOv5s model with a total loss value of 0.04 comes to the fore. Then, in the trials

with the Yolov5s model, a Colab file running Tesla K4 extracted an image in approximately 0.009 seconds. In other words, approximately 110 FPS operations are performed per second. When compared with other models, it was found appropriate to use the YOLOv5s model in the application to be prepared for recognizing the letters in the Turkish Sign Language Alphabet, since it has the lowest loss value and operates faster than other models. Information about all the models prepared is shown in Table 1.

Table 1. Performance values of implemented models

Model	Size of inputs	Step / Epoch	Batch Size	Cls Loss	Lclzsn /Bbox Loss	Total Loss	Training time
SSD Mobilenet	640x640	5000	14	0,10	0,04	0,84	2 h 5 min.
EfficientdetD0	512x512	5000	16	0,47	0,25	0,78	2 h
YOLOv5s	416x416	100	16	0,008	0,02	0,04	1 h 59 min. 47 sec.

3.6. Classification of TSL letters based on YOLOv5s Model

As a result of the training process of the original dataset of TSL letters, it was understood that the model with the highest success rate was the YOLOv5s model. It recognized the TSL letters of the YOLOv5 model at 110 FPS per second, with a success rate of 92.3.

To recognize TSL letters, the training process was carried out by changing only the number of epochs as 200 from the above-mentioned parameters in the YOLOv5s model. Each epoch performed, the training process averaged 1 minute for 438 groups and 11 seconds for evaluation on 63 groups. The total duration of the training was calculated as 4 hours and 43 seconds. As a result of the training, Precision (Precision), Recall (Sensitivity) and Mean Average Precision (Mean Average Precision) performance values are calculated both in total and separately for each class. The formulas used to calculate the performance values are given in formulas 1,2 and 3, respectively. The expression presented with Formula 1 was used to calculate the Accuracy value.

$$Acc = (TP+TN)/(TP+TN+FN+FP) \quad (1)$$

Precision shown by Formula 2 is expressed as the ratio of correctly classified data to all classified data.

$$P = TP/(TP+FP) \quad (2)$$

Recall, also known as sensitivity value, this value presented by Formula 3. expresses the ratio of correctly classified data to all correctly classified and misclassified data.

$$SN = TP/(TP+FN) \quad (3)$$

Precision Value 0.921, Recall Value 0.89, and mAP@.5 Value 0.923 after training performed at 200 epochs. Some numerical information about these calculated values is presented in Figure 8, and the complexity matrix is shown in Figure 9.

```
      Epoch    gpu_mem       box       obj       cls     total    labels  img_size
    199/199      1.73G   0.02332  0.007587  0.005633   0.03654        13       416: 100% 438/438 [01:00<00:00,  7.23it/s]
                 Class     Images     Labels          P          R     mAP@.5  mAP@.5:.95: 100% 63/63 [00:11<00:00,  5.27it/s]
                   all       2000       2000      0.921      0.893      0.923      0.505
                     A       2000         70      0.977      0.843      0.937      0.454
                     B       2000         70      0.868          1      0.968      0.527
                     C       2000         70      0.882      0.957      0.987      0.649
                   CCC       2000         70      0.976      0.857      0.932      0.493
                     D       2000         70      0.836      0.726      0.794      0.518
                     E       2000         70      0.966        0.9       0.93      0.248
                     F       2000         70      0.946      0.843      0.914        0.6
                     G       2000         70      0.983      0.971      0.994      0.635
                   GGG       2000         70      0.976          1      0.995       0.55
                     H       2000         70      0.867      0.935       0.88      0.538
                     I       2000         60      0.925      0.983      0.969       0.48
                   III       2000         60      0.846      0.983      0.931      0.505
                     J       2000         60      0.731      0.867      0.703      0.351
                     K       2000         80      0.962      0.487      0.845       0.36
                     L       2000         60      0.999       0.95      0.993      0.574
                     M       2000         70      0.843      0.857       0.76      0.432
                     N       2000         70       0.99      0.857      0.935      0.625
                     O       2000         70      0.984      0.882      0.991      0.569
                   OOO       2000         70      0.994          1      0.995      0.536
                     P       2000         70       0.92          1      0.988      0.501
                     R       2000         70          1      0.948      0.985      0.329
                     S       2000         70      0.985      0.943      0.994      0.513
                   SSS       2000         70      0.877          1      0.994      0.571
                     T       2000         70          1      0.995      0.995      0.454
                     U       2000         70      0.752      0.986      0.909      0.578
                   UUU       2000         70          1      0.547      0.765      0.417
                     V       2000         70      0.813      0.957       0.97      0.638
                     Y       2000         70      0.955      0.914      0.912      0.527
                     Z       2000         70      0.861      0.714      0.814      0.461
200 epochs completed in 3.833 hours.

Optimizer stripped from runs/train/yolov5s_results15/weights/last.pt, 14.9MB
Optimizer stripped from runs/train/yolov5s_results15/weights/best.pt, 14.9MB
CPU times: user 2min 57s, sys: 22.1 s, total: 3min 19s
Wall time: 4h 43s
```

Figure 8. YOLOv5s loss values

In the confusion matrix shown in Figure 9. below, the TP (True Positive) class was predicted correctly, TN (True Negative) was predicted correctly if it should not be in the relevant class, FP (False Positive) was incorrectly predicted in a class even though it should not be, and FN (False Negative) indicates that it was incorrectly guessed when it should not be in the relevant class.

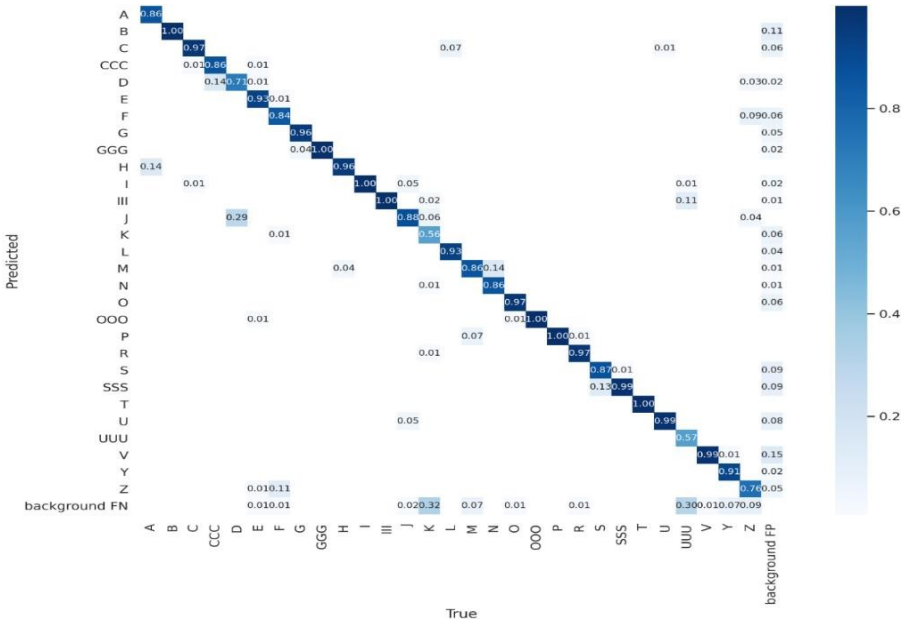

Figure 9. YOLOv5s Confusion Matrix

When the values obtained for each letter were checked after the training, it was determined that there were letters with mAP@.5 value below 0.9. It is seen that these detected letters are D, H, J, K, M, Ü, and Z. It is evaluated

that the most important factors in the confusion of these letters in the TSL by the model are the movements that make up these letters using both hands and the positions of the fingers.

4. Conclusion and Evaluation

It is aimed to recognize the letters in the TSL alphabet quickly and accurately. It has been understood that there is no publicly shared data set suitable for the application to be carried out in line with this goal. Thereupon, videos shared publicly by 15 different people to learn the letters in TSL were used to generate a data set. In cases where the images obtained from the videos were not sufficient, new videos were prepared. Using all the videos recorded, a new and unique data set was generated with a total of 10000 images containing 29 letters in our alphabet. As a result of the research, it was found appropriate to use the YOLO model, which can reach 140 FPS in the fields of object recognition and object tracking. The generated data set was trained in the Google-Colab environment with the latest version of the YOLO model, YOLOv5, and a success rate of 92.3% was achieved.

Although 92.3% success was achieved with the implemented application, one of the biggest factors affecting this success rate is the data set. The dataset used in the application was generated by people of different genders in many different age groups. It is thought that the improvements to be made in the data set can increase the performance of the model since most of the movements that make up the TSL alphabet are made using both hands and that the movements that make up some letters are not fixed.

References

[1] Charayaphan, C., & Marble, A. E. (1992). Image processing system for interpreting motion in American Sign Language, Journal of Biomedical Engineering, 14(5), 419-425.

[2] Takahashi, T., & Kishino, F. (1991). Hand gesture coding based on experiments using a hand gesture interface device. Acm Sigchi Bulletin, 23(2), 67-74.

[3] Waldron, M. B., & Kim, S. (1995). Isolated ASL sign recognition system for deaf persons. IEEE Transactions on rehabilitation engineering, 3(3), 261-271.

[4] Allen, J. M., Asselin, P. K., & Foulds, R. (2003, March). American Sign Language finger spelling recognition system. In 2003 IEEE 29th Annual Proceedings of Bioengineering Conference (pp. 285- 286). IEEE.

[5] Wang, H. G., Sarawate, N. N., & Leu, M. C. (2004, July). Recognition of American sign language gestures with a sensory glove. In Japan USA Symposium on Flexible Automation, Denver, CO (pp. 102-109).

[6] M. Taskiran, M. Killioglu and N. Kahraman, (2018) "A Real-Time System for Recognition of American Sign Language by using Deep Learning," 2018 41st International Conference on Telecommunications and Signal Processing (pp. 1-5)

[7] Haberdar, H., & Albayrak, S. (2005, October). Real time isolated turkish sign language recognition from video using hidden markov models with global features. In International Symposium on Computer and Information Sciences (pp. 677687). Springer, Berlin, Heidelberg.

[8] 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.

[9] Memiş, A., & Albayrak, S. (2013, April). Turkish Sign Language recognition using spatio-temporal features on Kinect RGB video sequences and depth maps. In 2013 21st Signal Processing and Communications Applications Conference (SIU) (pp. 1-4). IEEE.

[10] Demircioglu, Burcak & Bülbül, Güllü & Kose, Hatice. (2016). Leap Motion ile Türk İşaret Dili Tanıma / Turkish Sign Language Recognition with Leap Motion. 10.13140/RG.2.1.4923.3529.

[11] Ceber, Y. E., Karacaoğlan, E., Uysaf, F., & Tokmakçi, M. (2017, October). The design of glove that can translate sign language to Turkish language. In 2017 Medical Technologies National Congress (TIPTEKNO) (pp. 1-4). IEEE.

[12] Firat, Yelda & Uğurlu, Taşkın. (2018). LATİS TABANLI ANLAM ÇÖZÜMLENMESİ İLE TÜRKÇE İŞARET DİLİ TERCÜME SİSTEMİ. Ömer Halisdemir Üniversitesi Mühendislik Bilimleri Dergisi. 10.28948/ngumuh.443157. [13] Çelik, Ö , Odabas, A . (2020). Sign2Text: Konvolüsyonel Sinir Ağları Kullanarak Türk İşaret Dili Tanıma . Avrupa Bilim ve Teknoloji Dergisi , (19) , 923-934 .

[14] https://towardsdatascience.com/yolo-v4-optimal-speed-accuracy-for-object-detection-79896ed47b50 Erişim Tarihi 06/06/2021.