




# T-SignSys: An Efficient CNN-Based Turkish Sign Language Recognition System

Sevval Colak<sup>1</sup>, Arezoo Sadeghzadeh<sup>1</sup>(✉) , and Md Baharul Islam<sup>1,2</sup> 

<sup>1</sup> Bahcesehir University, 34349 Besiktas, Istanbul, Turkey  
arezoo.sadeghzadeh@bahcesehir.edu.tr

<sup>2</sup> Florida Gulf Coast University, Fort Myers, FL 33965, USA

**Abstract.** Sign language (SL) is a communication tool playing a crucial role in facilitating the daily life of deaf or hearing-impaired people. Large varieties in the existing SLs and lack of interpretation knowledge in the general public lead to a communication barrier between the deaf and hearing communities. This issue has been addressed by automated sign language recognition (SLR) systems, mostly proposed for American Sign Language (ASL) with limited number of research studies on the other SLs. Consequently, this paper focuses on static Turkish Sign Language (TSL) recognition for its alphabets and digits by proposing an efficient novel Convolutional Neural Network (CNN) model. Our proposed CNN model comprises 9 layers, of which 6 layers are employed for feature extraction, and the remaining 3 layers are adopted for classification. The model is prevented from overfitting while dealing with small-scale datasets by benefiting from two regularization techniques: 1) ignoring a specified portion of neurons during training by applying a dropout layer, and 2) applying penalties during loss function optimization by employing  $L_2$  kernel regularizer in the convolution layers. The arrangement of the layers, learning rate, optimization technique, model hyper-parameters, and dropout layers are carefully adjusted so that the proposed CNN model can recognize both TSL alphabets and digits fast and accurately. The feasibility of our proposed *T-SignSys* is investigated through a comprehensive ablation study. Our model is evaluated on two datasets of TSL alphabets and digits with an accuracy of 97.85% and 99.52%, respectively, demonstrating its competitive performance despite straightforward implementation.

**Keywords:** CNN · Digits and Alphabets · Static Sign language recognition · Turkish Sign Language

---

This work is supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under the 2232 Outstanding Researchers program, Project No. 118C301 and the 2247-C Trainee Researcher Scholarship Program.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024  
A. Ortis et al. (Eds.): ICAETA 2023, CCIS 1983, pp. 226–241, 2024.  
[https://doi.org/10.1007/978-3-031-50920-9\\_18](https://doi.org/10.1007/978-3-031-50920-9_18)

# 1 Introduction

Sign language (SL) is a form of non-verbal communication means used to convey information through hand gestures and facial expressions. It is utilized by deaf and hard-of-hearing people to interact with others and access services. There are more than 300 SLs worldwide that significantly differ from each other in various terms such as vocabulary and grammatical structures [17]. Due to these wide varieties, learning and interpreting different SLs is time-consuming and infeasible for hearing and hearing-impaired people. On the other hand, accessing human interpreters is costly and not practical in daily life. These challenges form a communication barrier in deaf-to-deaf and deaf-to-hearing people interactions, which not only limits the deaf people's social and professional life (e.g., fewer employment chances, social withdrawal, low academic performance) but also has substantial adverse mental impacts on them, leading to depression, loneliness, and anger. To tackle these issues, researchers have been captivated to develop automated sign language recognition (SLR) models that can accurately identify the signs performed by the signer and assist the public people in interpreting them effortlessly.

Automated SLR systems can significantly improve the life quality of the deaf community by smoothing over their communication and facilitating social service usage. Additionally, they can be used in the form of gesture recognition in many other human-computer interaction applications ranging from virtual/augmented reality (VR/AR) and video games [26] to medical purposes [11]. In these applications, a gesture recognition system tracks the user's hand movements and converts them into actions within a program. The first successful attempts regarding automated SLR have been made using direct measurements through sensor-based devices. These systems are generally more accurate due to their ability to detect the exact position, speed, and other characteristics of the user's hands. However, they require specialized devices [21], which are costly and inconvenient for performing complex signs. These limitations restrict their applicability in real-life scenarios, inspiring researchers to convert their attention to vision-based systems as a viable alternative.

Using images and videos acquired by cameras as input data in the vision-based system makes them appropriate for performing complex signs in different environments. Although the affordability, portability, and high flexibility of the vision-based systems make them superior to the sensor-based systems, they still have their own challenges caused by large varieties that exist either in signers (e.g., hand shapes, skin colors, and way of performing a sign), environment conditions (e.g., complex background, and lighting changes), or both. To address these issues, numerous works have been proposed in the last decade, which are categorized into two main groups considering the input data modality: 1) Static SLR (SSLR), and 2) Dynamic SLR (DSLRL).

SSLR is defined as recognizing the digits and alphabets of a sign language from the images [24]. Static sign language (also referred to as fingerspelling) is utilized to perform ages, dates, proper nouns, and technical words with no specific sign, constituting a considerable portion of each SL [29]. This prominent role

is important to SSLR attracting researchers' attention to develop highly accurate systems over the years. SSLR approaches can be further divided into two main categories: conventional machine learning (ML)-based and deep learning (DL)-based methods. In ML-based approaches, hand-crafted features are extracted from the input images, then utilized in the ML-based classifiers for final sign recognition. Despite the favorable performance achieved by these methods, their accuracy is highly dependent on the extracted features limiting their applicability to a specified dataset with poor generalization ability [15].

With the advent of DL-based models, highly powerful GPUs, and sign language datasets with large quantities, Convolutional Neural Networks (CNNs) have been widely employed in this domain to improve the performance and the generalization ability [25]. The capability of these models in extracting highly representative features from the complex structures through the backpropagation technique enables them to accurately differentiate the images from one another and achieve high performance. However, most of the approaches in the literature have focused on American Sign Language (ASL) recognition [7]. Although ASL is a globally well-known SL, many other SLs (e.g., Bangla, Arabic, Persian, and Turkish) are also significant for their own communities. However, there are limited studies on these SLs compared to those of ASL. Additionally, even for ASL recognition, alphabet recognition got more attention than digit recognition, while both are significant components of SSLR.

Turkish Sign Language (TSL) is one of the critical SLs used by around 3.5 million deaf people in Turkey. At the same time, its fingerspelling has been studied in only a limited number of research works in the literature. This paper aims to investigate static TSL recognition to ease the communication between deaf people and the general public unfamiliar with it in Turkey. Our proposed *T-SignSys* is based on a novel CNN model for fast and accurate TSL recognition, considering both alphabets and digits. Overall, the main contributions of this paper are listed as follows:

- A novel end-to-end CNN-based model, namely *T-SignSys*, is proposed to efficiently recognize both the alphabets and digits of TSL with a single optimized architecture following a straightforward implementation avoiding the challenging task of hand segmentation for complex backgrounds.
- Carefully tuning the number of layers, activation function, kernel, and filter sizes, and optimizer technique along with taking advantage of dropout layer and  $L_2$  kernel regularizer as regularization techniques, our model obtains high accuracy without overfitting issues even for small-scale datasets.
- A comprehensive ablation study is conducted for the arrangements of the layers, the value of the dropout layer, the learning rate, and the optimization technique to investigate the effectiveness and feasibility of the proposed architecture. As an important factor in achieving high performance and fast convergence during training, Adam and Adamax are selected as optimizer techniques for digits and alphabets, respectively, through our ablation study.

- The performance of the proposed model is evaluated on two benchmark datasets of TSL alphabets and digits and compared with those of existing approaches proving its superiority and capabilities.

## 2 Related Works

During the last two decades, a significant number of studies have been conducted to advance automated SLR systems, which are categorized into two main groups: conventional machine learning (ML)-based and deep learning (DL)-based approaches. The automated systems in both groups follow three steps to accomplish sign language recognition: pre-processing, feature extraction, and classification. In conventional methods, feature extraction has been performed by extracting hand-crafted features classified using ML-based classifiers. In contrast, feature extraction and classification are carried out in DL-based systems through a single deep network. Some of the recent sign language recognition (SLR) approaches from both categories are briefly discussed in this section.

### 2.1 Machine Learning-Based Approaches

Machine learning is a subset of artificial intelligence that involves training algorithms to learn patterns and make predictions or decisions based on data. ML techniques have been used for both sensor-based and vision-based SLR. In the sensor-based approaches, the data acquired directly from the sensor-based devices are used in ML-based classifiers (e.g., Support Vector Machine (SVM), K-Nearest Neighbor (KNN)) for sign recognition, while in the vision-based systems, the inputs of the classifiers are the hand-crafted features (e.g., Histogram of Oriented Gradients (HOG), Local Binary Pattern (LBP)) extracted from sign images. Guardino et al. [9] used a Leap Motion sensor-based device and two classifiers of KNN and SVM for recognizing the ASL alphabets. Their acquired data included features from the fingers and palm, such as position, direction, and velocity, which were transmitted to the computer via a USB connection. Instead of employing the raw data, they computed average distance, average spread, and average tri-spread, which were fed into the classifiers for final recognition. In their system, KNN and SVM classifiers obtained an accuracy of 72.78% and 79.83%, respectively, proving the superiority of SVM over KNN. Another sensor-based approach was proposed by Yalçın et al. [30] using a glove equipped with elastic detectors and a gyroscope to detect sign language and translate it into text. Despite its successful performance for SL translation, it was inconvenient and costly for real-life scenarios due to using gloves equipped with detectors and wires connected to an Arduino.

As an affordable and more convenient system, a vision-based approach was proposed in [14] by Kumar et al. using both images and videos as input data. In the pre-processing step, they applied face detection and removal using the Viola and Jones algorithm to prevent the head skin color from interfering with the hand detection. Then, the hand was segmented using HSV thresholding. To improve

accuracy, instead of using a global threshold value in the segmentation process, they sampled the skin color of the signer before sign recognition. For static sign language recognition, they utilized the feature vectors extracted through Zernike moments in an SVM classifier achieving an accuracy of 93%. A multi-kernel SVM was trained in [8] using a proper fusion of three different hand-crafted features. The same classifier was also employed in [18], but this time with LBP features.

To tackle the curse of dimensionality as the main limitation of the ML-based approaches, Principal Component Analysis (PCA) was used in [23] to reduce the dimensionality of the feature vector acquired by a combination of different image descriptors. They evaluated their proposed system for user-dependent and -independent scenarios by applying three different classifiers of KNN, Multi-Layer Perceptron (MLP), and Probabilistic Neural Network (PNN). Amrutha and Prabu [4] designed a model capable of recognizing single-handed signs using convex hull features and a KNN classifier. Their training dataset was collected in a controlled environment with a stable and plain background, and image backgrounds were removed in a pre-processing step using the threshold method. Contour-based segmentation was then applied to obtain the contours of the fingers. Despite achieving an accuracy of 65%, the performance of their system was highly dependent on the distance between the camera and the signer. In a recent work by Bansal et al. [5], a system was proposed based on HOG features and SVM classifier whose performance was evaluated on seven different datasets. They utilized Minimum Redundancy and Maximum Relevance (mRMR) and Particle Swarm Optimization (PSO) techniques to remove the feature redundancy while maintaining accuracy. Generally, in ML-based approaches, satisfactory performance is obtained if the extracted features are representative and strong enough. However, they still suffer from poor generalization ability and time-consuming feature extraction steps with high dimensionality.

## 2.2 Deep Learning-Based Approaches

Deep learning (DL) is another subset of artificial intelligence that involves training artificial neural networks to learn patterns and make predictions or decisions based on data. CNN is a powerful DL model that automatically extracts deeper and more effective low- and high-level features from the input images alleviating the need for manual feature engineering and hand-crafted feature extraction. The superior performance of the CNNs over the conventional ML-based methods and the emergence of the advanced GPUs and large datasets inspired the researchers to employ them in various computer vision domains such as action recognition, object detection, classification, etc. They also have been extensively used in SLR to enhance the systems' accuracy and provide real-time performance.

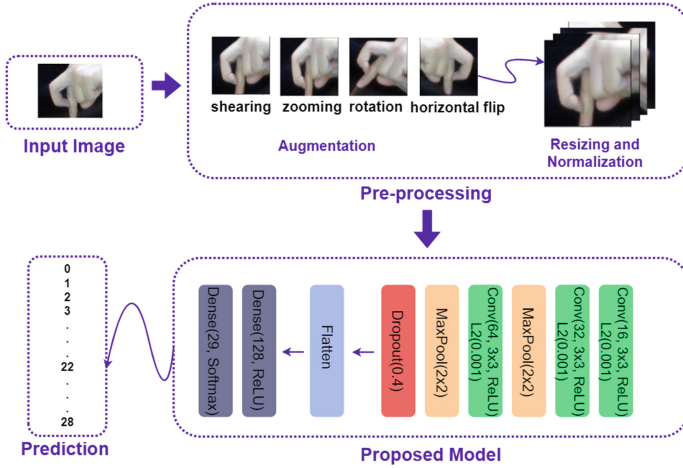
In DL-based approaches, CNN models can be used in two schemes: 1) only extracting deep features, which are then classified by ML-based classifiers, and 2) conducting both deep feature extraction and classification through a single architecture. Sanchez-Riera [27] conducted a comparative study to analyze and compare the performance of CNN models with the other classifiers using two input modalities of RGB and depth. A CNN-based ASL recognition model was

also proposed in [22] to investigate the superiority of the DL models over the ML classifiers of SVM, KNN, and RF. In [20], another novel CNN model was proposed for ASL recognition whose accuracy was enhanced through applying a pre-processing step and hand segmentation. They also developed a new ASL dataset. Das et al. [10] utilized a CNN model to recognize the ASL alphabets, achieving an accuracy of 94.34% for images with plain black backgrounds. Sevli and Kemalöglu [28] developed a CNN model for recognizing TSL digits, achieving an accuracy of 98.55%. They conducted extensive experiments on four optimizers to investigate their impacts on the model's performance. Another research work regarding TSL recognition was carried out by Öztürk et al. [19] using Faster-RCNN. They trained their model using the TSL alphabet dataset and tested it using by real-time data, achieving an accuracy of 88%. A new TSL dataset was developed by Aksoy et al. [2], including 10223 images for 29 letters captured at different distances from the camera on plain white background. They performed TSL recognition using different pre-trained models and their proposed model as TSLNet. Among these models, CapsNet and TSLNet models outperformed the others with an accuracy of 99.7% and 99.6%, respectively.

One of the techniques recently attracted the researchers' attention for enhancing the performance of SLR systems is using a combination of several models. Kodandaram et al. [13] used an ensemble of three models (i.e., LeNet-5, MobineNetV2, and a custom CNN) for ASL recognition obtaining an accuracy of 99.89%. Bhaumik et al. [6] created a portable end-to-end CNN model named as ExtriDeNet using two main modules: the intensive feature fusion block (IFFB) and the intensive feature assimilation block (IFAB). The capability of the ExtriDeNet in dealing with challenging environmental conditions (i.e., illumination variations and complex backgrounds) was demonstrated through their experiments. Bousbai et al. [7] enhanced the recognition performance for four ASL datasets by ensembling the features extracted by a custom CNN and a CapsNet. Once PCA reduced the dimensionality of the combined feature vector, it was used in an SVM classifier for final recognition. Alnuaim et al. [3] combined ResNet50 and MobileNetV2 architectures for Arabic SLR achieving an accuracy of about 97% after applying various data augmentation techniques. Zakariah et al. [31] used various pre-trained models for Arabic SLR based on transfer learning. Applying different pre-processing and augmentation techniques, the EfficientNetB4 model outperformed the others with an accuracy of 95%. However, it was computationally inefficient due to being a heavy-weight model.

### 3 Methodology

The overall flowchart of the proposed TSL recognition system is illustrated in Fig. 1. It comprises three main sections: pre-processing (including augmentation and image resizing), feature extraction, and classification. The last two steps are implemented through a single novel CNN model whose layer arrangement and hyper-parameters are optimized to achieve high accuracy for both TSL digit and alphabet recognition. Details of each step are discussed in the following sections.



**Fig. 1.** The main flowchart of the proposed *T-SignSys*.

### 3.1 Data Preprocessing

Pre-processing is a crucial step in DL-based models to enhance the input data quality and quantity to boost the proposed model’s performance. Hence, we first apply data augmentation in our pre-processing step. Data augmentation refers to the techniques employed for modifying and generating new data. It plays a significant role as a regularizer to prevent the model from overfitting and make it robust for various input patterns by increasing the diversity and quantity of the data in the training set (i.e., oversampling). Hence, proper selection of the augmentation techniques is significant to achieve optimized performance. In our model, the quantity of the training samples (RGB images) is increased through four augmentation techniques of shearing (ranges between 0–0.2), zooming (ranges between 0–0.2), rotation (ranges between 0–45), and horizontal flip. All RGB images, including the augmented ones, are resized into the same size ( $100 \times 100$  for digits and  $224 \times 224$  for alphabets). Then, pixel intensities are divided by 255 for normalization.

### 3.2 Model Architecture

CNNs are multi-layer architectures trained to extract features from the input images in the form of feature maps that include the corresponding images’ main characteristics. A novel efficient CNN architecture, namely *T-SignSys*, is proposed in our paper for accurate and fast TSL digit and alphabet recognition, whose details are presented in Table 1. It is formed by a total of 9 layers. The first 6 layers (i.e., three convolutional layers, two max-pooling layers, and a dropout layer) are selected for feature extraction, and the remaining 3 layers (flatten and fully-connected/dense) are used for classification. Having a hierarchical struc-

**Table 1.** Summary of the proposed CNN architecture presenting its hyper-parameters.

Layer	Filters	Kernel	Pool	Strides	Activation	Regularizer	Output Shape	Param #
Conv2d	16	$3 \times 3$	–	–	ReLU	L2	(222, 222, 16)	448
Conv2d	32	$3 \times 3$	–	–	ReLU	L2	(220, 220, 32)	4640
Max Pool	–	–	$2 \times 2$	2	–	–	(110, 110, 32)	0
Conv2d	64	$3 \times 3$	–	–	ReLU	L2	(108, 108, 64)	18496
Max Pool	–	–	$2 \times 2$	2	–	–	(54, 54, 64)	0
Dropout(40%)	–	–	–	–	–	–	(54, 54, 64)	0
Flatten	–	–	–	–	–	–	(186624)	0
Dense	–	–	–	–	ReLU	–	(128)	23888000
Dense	–	–	–	–	Softmax	–	(29)	3741

ture, each layer is in charge of non-linearly transforming the input images, and the results are fed into the next layer for another non-linear transformation.

**Feature Extraction Module:** The inputs of our feature extraction module are RGB images which are resized in the pre-processing step. They are fed into the first convolutional block which is composed of two convolutional layers and a maxpooling layer. These convolutional layers are responsible for learning and extracting the low-level features from the input images, such as edges, curves, and textures. Filters and kernels in these layers are the hyper-parameters responsible for extracting the representative features. Each filter is a small matrix of weights that is used to perform the convolution operation on the input data. The weights in the filter are adjusted during training so that the filter becomes increasingly sensitive to the specific pattern or feature it is trying to detect. The number of filters in a convolutional layer can significantly affect the overall performance of the model. More filters aid in extracting more specific patterns and features by the convolutional layer as the model gets more powerful. However, this filter size increment leads to more parameters and so the risk of overfitting. Considering a good balance between these two issues, we selected the number of filters as 16 and 32 in the first and second layers, respectively. The third convolutional layer which is placed in the second convolutional block is considered the deeper layer for our model. Consequently, we selected a larger filter number in comparison to the first two convolutional layers so that more complex abstract patterns are extracted from the images. Kernel size is another crucial parameter in the convolutional layer which determines the level of abstraction and the number of parameters in the model. A convolutional layer with a larger kernel size can capture more contextual information from the input image and potentially extract more meaningful features. However, similar to the filter size, increasing the kernel size increases the number of parameters leading to overfitting if the model is not properly regularized. On the other hand, a smaller kernel size can capture more fine-grained details from the input image with less number of parameters, but it may not be able to capture the global context as effectively. Making a good trade-off, we select kernel size as  $3 \times 3$  in all convolutional layers achieving the optimal performance with our input images.



In our feature extraction module, two regularization techniques are applied to further improve the performance of the system by increasing the accuracy, improving the convergence, speeding up the training, enhancing the generalization ability, and alleviating the overfitting risk. The first technique is applying  $L_2$  kernel regularizer in all convolutional layers. It is a form of regularization that adds a penalty term to the objective function of the model, which penalizes large weights and encourages the model to learn a simpler, more generalized solution for new unseen data. A dropout layer is used at the end of our feature extraction module as the other regularization technique. Its value is set to 0.4, which means that 40% of the neurons are ignored for weight updates during the training, which prevents the neurons from co-adapting and forces the model to rely on the remaining neurons to make predictions. This dropout layer significantly improves the accuracy of the model and minimizes the gap between the training and the test accuracy (detailed results are discussed in the ablation study section). Two maxpooling layers adopted in our model have identical hyperparameters, i.e.,  $2 \times 2$  kernel size. They are used in both convolutional blocks after convolutional layers to downsample the feature maps by reducing their dimensionality, while still retaining the most important information. It is worth mentioning that the stride value (i.e., the number of pixels that are skipped by the kernel filter in each movement) is selected as one and two in the convolutional and maxpooling layers, respectively, with the “valid” padding scheme. The outputs of the convolutional layers are fed into a non-linear activation function known as Rectified Linear Unit (ReLU). Using this activation function, only the neurons with positive values are activated ( $f(x) = \max(0, x)$ ), which leads to fast training and minimizes the possibility of gradient vanishing.

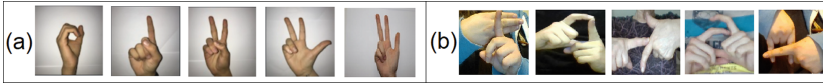
**Classification Module:** The generated 2D feature maps from the feature extraction module are reshaped into a 1D feature vector using a flattening layer before the fully connected (FC)/dense layers. This feature vector is fed into a FC layer with 128 neurons and a ReLU activation function. To set the number of neurons to 128 in the FC layer as its crucial hyperparameter, we start with a relatively small number of neurons and gradually increase it until reaching the convergence and plateau performance. This layer is followed by the second FC layer, whose number of neurons equals the number of class labels in the corresponding dataset. The last FC layer, known as the classification layer, uses *Softmax* as its activation function. It is a generalization of the logistic function used to convert arbitrary values into a probability distribution, where the sum of the probabilities is 1. It is computed as  $Softmax(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$ , where  $x_i$  is a vector of arbitrary values used as its input. The calculated probabilities by this function determine the membership of the input to each class. The highest value probability determines the corresponding input’s predicted class. Our model is trained based on the *categorical cross-entropy* loss function as it is dealing with a multi-class classification task. It is defined as  $Loss = - \sum_{i=1}^{output\ size} y_i \cdot \log \hat{y}_i$ , where  $\hat{y}_i$  and  $y_i$  represent the probabilities of the model prediction and the corresponding true target, respectively, and it is demanded to reduce the difference between them during the training process.

## 4 Experimental Results

### 4.1 Datasets

Two TSL datasets are selected for our experiments:

- 1) TSL Digits [1, 16]: This dataset includes a total of 2062 RGB images with a fixed size of  $100 \times 100$  in 10 classes for the digits from 0 to 9. The signs of this dataset were performed by 218 different right-handed signers. The images were all captured on a plain white background. Some samples of this dataset are illustrated in Fig. 2(a).
- 2) TSL Alphabets [12]: As the first alphabet dataset for static Turkish Sign Language, it is composed of 2974 RGB images categorized into 29 classes out of which 23 categories are for Turkish letters (excluding “Ç, Ğ, İ, Ö, Ş, and Ü”) and the remaining 6 classes are allocated for punctuation marks. The images in this dataset have different sizes which were captured on cluttered backgrounds. It contains both single- and double-handed signs performed by both left- and right-handed signers. Some samples of this dataset are illustrated in Fig. 2(b).



**Fig. 2.** Illustration of some sample images from (a) TSL Digits, and (b) TSL Alphabets.

### 4.2 Experimental Setup

All experiments are conducted with Python using Keras and Tensorflow on a PC with 32 GB RAM, Intel Core i7-9700 CPU, and NVIDIA GeForce RTX 2070 GPU with 8 GB video memory. We use Adam and Adamax optimizers for digit and alphabet datasets, respectively, with a learning rate of 0.001. Epoch and batch sizes are selected as 300 and 32, respectively, with data split of 80%–20%.

### 4.3 Evaluation Metrics

The performance of our proposed model is evaluated in terms of four evaluation metrics common for classification tasks as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall/Sensitivity = \frac{TP}{FN + TP} \quad (3)$$

$$F1 - score = \frac{2 \times (Precision \times Recall)}{Precision + Recall} \quad (4)$$

where  $TP$ ,  $TN$ ,  $FP$ , and  $FN$  stand for True Positive, True Negative, False Positive, and False Negative, respectively, obtained from the confusion matrix.

#### 4.4 Performance Assessment

The performance and efficiency of the proposed CNN model are evaluated on two benchmark TSL datasets for digits and alphabets. The experimental results are presented in Table 2 in terms of four evaluation metrics for both training and test sets. Achieving a test accuracy of 99.52% and 97.85% for digit and alphabet datasets, respectively, demonstrates the high capabilities of our proposed model despite its straightforward architecture. As *T-SignSys* is assessed on two diverse datasets with both plain and cluttered backgrounds, single- and double-handed signs performed by left- and right-handed signers, one can draw the inference that it is efficient and robust against hand appearances and environmental conditions. Additionally, our model achieves a great convergence without experiencing any underfitting or overfitting as the difference between the training and test accuracy is minimized in both datasets. Owing to the less number of parameters and benefiting from the regularization techniques in our model, it is fast both in training and test processes making it feasible and practical for real-time application. Training of the digit and alphabet datasets takes 17 and 105 min for 300 epochs, respectively. 418 digit images and 604 alphabet images are recognized in the test phase during 0.47 and 5 s, respectively, proving the real-time performance of our proposed method.

**Table 2.** Performance evaluation in terms of four evaluation metrics on two datasets.

	Digits Dataset		Alphabet Dataset	
	Training	Test	Training	Test
Accuracy	98.11%	99.52%	99.92%	97.85%
Precision	99.02%	98.56%	99.24%	97.33%
Recall	98.78%	98.33%	98.90%	93.54%
F1 Score	98.50%	98.32%	99.08%	94.96%

**Table 3.** Performance comparison with state-of-the-art TSL recognition approaches on two TSL datasets.

Dataset	Approaches	Training(%) / Testing(%) Split	Test Accuracy
TSL Digits	Sevli and Kemaloglu [28]	80/20	98.55%
	Bansal et al. [5]	90/10	90.90%
	<b>Proposed Model</b>	<b>80/20</b>	<b>99.52%</b>
TSL Alphabets	Ozturk et al. [19]	40/-	88.00%
	<b>Proposed Model</b>	<b>80/20</b>	<b>97.85%</b>

#### 4.5 Comparison with State-of-the-Art

The performance of the proposed model is compared with the available state-of-the-art approaches for TSL recognition on two datasets in Table 3 presenting their test accuracy along with the training and test data split scheme. In the digit dataset, our model outperforms two other approaches with 0.97% and 8.62% accuracy enhancement. For TSL alphabets, it should be mentioned that Ozturk et al. [19] employed the TSL alphabet dataset only for training their model and they tested it by real-time images while we used the TSL alphabet dataset in both training and test phases. Comparing the test accuracy, our model surpasses their approach by achieving 9.85% more accuracy.

#### 4.6 Ablation Study

To deeply investigate the impact of different parameters on the overall performance of our proposed *T-SignSys*, a comprehensive ablation study is carried out over the arrangement of the layers (9 different combinations), dropout layer value (0.25 and 0.40), optimizer (RMSProp, SGD, Adamax, and Adam), and learning rate (for 0.001, and 0.0001). The results for different layer arrangements of our CNN model are presented in Table 4 for two datasets in terms

**Table 4.** The results of ablation study over 9 different arrangements for CNN layers.

Layer Arrangement	Digits Dataset		Alphabet Dataset	
	Train	Test	Train	Test
1 Conv + 1 FC	93.07%	88.28%	87.55%	80.96%
2 Conv + 1 FC	98.11%	99.52%	97.30%	90.56%
3 Conv + 1 FC	97.81%	97.00%	94.39%	88.08%
3 Conv + 1 MP + 1 FC	96.96%	97.00%	98.82%	92.00%
3 Conv + 2 MP + 1 FC	96.96%	94.00%	98.90%	93.38%
4 Conv + 2 MP + 1 FC	96.53%	96.17%	98.02%	92.38%
5 Conv + 2 MP + 1 FC	94.95%	94.98%	95.02%	91.39%
3 Conv + 2 MP + 2 FC	<b>99.20%</b>	97.37%	98.27%	94.04%
3 Conv + 2 MP + 1 DP + 2 FC (ours)	98.11%	<b>99.52%</b>	<b>99.92%</b>	<b>97.85%</b>

**Table 5.** The results of ablation study over different values for dropout layer.

Dropout Values	Digits Dataset		Alphabet Dataset	
	Training	Test	Training	Test
<b>0.25</b>	<b>99.09%</b>	98.33%	99.79%	96.52%
<b>0.40 (ours)</b>	98.11%	<b>99.52%</b>	<b>99.92%</b>	<b>97.85%</b>

**Table 6.** The results of ablation study over four different optimization techniques.

Optimizers	Digits Dataset		Alphabet Dataset	
	Training	Test	Training	Test
SGD	78.47%	88.04%	83.76%	85.43%
RMSprop	<b>99.09%</b>	97.85%	98.23%	95.70%
Adamax	96.00%	94.00%	<b>99.92%</b>	<b>97.85%</b>
Adam	98.11%	<b>99.52%</b>	99.66%	95.53%

**Table 7.** The results of ablation study over two different learning rates on two datasets.

Learning Rate	Digits Dataset		Alphabet Dataset	
	Training	Test	Training	Test
<b>0.001 (ours)</b>	<b>98.11%</b>	<b>99.52%</b>	<b>99.92%</b>	<b>97.85%</b>
<b>0.0001</b>	96.05%	95.21%	98.06%	94.37%

of test accuracy. Employing two convolutional (Conv) layers with the last fully connected (FC) layer leads to high performance for digit dataset while its accuracy is very low for alphabet dataset. To achieve high performance for both digit and alphabet datasets, we gradually increase the number of convolutional layers and add maxpooling (MP) and FC layers to the architecture. Implementing different combinations, the architecture with three Conv layers, two MP layers, and two FC layers obtains a high performance for both datasets. However, there is still overfitting especially for the TSL alphabet. To overcome overfitting, a dropout (DP) layer is added before FC layers, which significantly enhances the test accuracy and minimized the accuracy difference between the training set (98.11% and 99.92% for digits and alphabets) and the new unseen data (99.52% and 97.85% for digits and alphabets). The performance of the dropout layer is investigated for two values of 0.25 and 0.40 in Table 5. The dropout layer with a value of 0.4 outperforms the other with 1.19% and 1.33% higher test accuracy for TSL digit and alphabet datasets, respectively.

Performance of *T-SignSys* for four different optimizers is investigated on two datasets in terms of accuracy for the learning rate of 0.001 in Table 6. The lowest performance is achieved by the SGD optimizer on both datasets as it is highly likely to be stuck in the local minimum and has slow convergence in comparison to the other optimizers for the same number of epochs. The other

three optimizers of RMSProp, Adamax, and Adam have significantly higher performance. Achieving the highest test accuracy of 99.52% by Adam for digits and 97.85% by Adamax for alphabet, they are selected as the optimal optimizers for our model. Learning rate is also studied for two values of 0.001 and 0.0001 on both TSL digit and alphabet datasets in Table 7. Adopting a higher value for the learning rate results in faster convergence rather than using a lower value for an equal number of epochs. Comparing the results of two learning rates, the test accuracy of our model is improved by 4.31% and 3.48% for digit and alphabet datasets, respectively, once we set the learning rate as 0.001 without experiencing overfitting.



**Fig. 3.** Visualization of a sample failure case for our model where the letter “S” is misclassified as “F” when they are performed by two left- and right-handed signers.

#### 4.7 Failure Cases

Although our proposed method can efficiently recognize both TSL digits and alphabets with a straightforward implementation scheme and fast recognition rate, it leads to misclassification when dealing with some specific alphabet classes. One of these classes with a high number of misclassification samples is “S”. In most cases, the letter “S” is misclassified as “F”. This misclassification occurred mainly due to using both right- and left-handed signers in the dataset. To delve deeper into this issue, one example is illustrated in Fig. 3. Performing letters of “S” and “F” by the same person shows the high inter-class variations between these two classes, which helps the model to distinguish them accurately. On the contrary, when one of them is performed by a left-handed signer and the other is performed by a right-handed signer, the captured images have slight inter-class variations which degrade the performance of our model and lead to misclassification.

## 5 Conclusion

In this paper, a novel efficient CNN model was proposed for fast and accurate classification of Turkish Sign Language (TSL) digits and alphabets. Despite the great importance of TSL for a large number of deaf people in Turkey, there are only a handful number of studies in the literature conducted on TSL static fingerspelling. To this end, a novel CNN architecture was designed with a total of 9 layers whose number of layers, hyper-parameters, optimizer, and learning

rate were carefully adjusted through extensive experiments so that high performance was obtained. To further enhance the performance and prevent the system from overfitting, we used kernel regularizer and dropout layer as two regularization techniques. Conducting a comprehensive ablation study, we investigated the effectiveness of four different optimizer techniques, two learning rates, two dropout values, and 9 different arrangements for layers of the proposed CNN. Achieving an accuracy of 99.52% and 97.85% for TSL digits and alphabets, respectively, with high recognition speed, demonstrated the high capabilities of our model and its feasibility for real-time applications considering cluttered backgrounds. As our future work direction, we plan to enhance the model performance for the signs with slight inter- and high intra-class variations as well as make it robust to environmental variations.

## References

1. ASL digits (2017). <https://www.kaggle.com/datasets/ardamavi/sign-language-digits-dataset>. Accessed 29 Dec 2022
2. Aksoy, B., Salman, O.K.M., Ekrem, Ö.: Detection of Turkish sign language using deep learning and image processing methods. *Appl. Artif. Intell.* **35**(12), 952–981 (2021)
3. Alnuaim, A., Zakariah, M., Hatamleh, W.A., Tarazi, H., Tripathi, V., Amoatey, E.T.: Human-computer interaction with hand gesture recognition using resnet and mobilenet. *Comput. Intell. Neurosci.* **2022** (2022)
4. Amrutha, K., Prabu, P.: ML based sign language recognition system. In: 2021 International Conference on Innovative Trends in Information Technology (ICITIIT), pp. 1–6. IEEE (2021)
5. Bansal, S.R., Wadhawan, S., Goel, R.: MRMR-PSO: a hybrid feature selection technique with a multiobjective approach for sign language recognition. *Arabian J. Sci. Eng.* 1–16 (2022)
6. Bhaumik, G., Verma, M., Govil, M.C., Vipparthi, S.K.: ExtriDenNt: an intensive feature extrication deep network for hand gesture recognition. *Vis. Comput.* **38**(11), 3853–3866 (2022)
7. Bousbai, K., Morales-Sánchez, J., Merah, M., Sancho-Gómez, J.L.: Improving hand gestures recognition capabilities by ensembling convolutional networks. *Expert Syst.* e12937 (2022)
8. Cao, J., Yu, S., Liu, H., Li, P.: Hand posture recognition based on heterogeneous features fusion of multiple kernels learning. *Multimed. Tools Appl.* **75**(19), 11909–11928 (2016)
9. Chuan, C.H., Regina, E., Guardino, C.: American sign language recognition using leap motion sensor. In: 2014 13th International Conference on Machine Learning and Applications, pp. 541–544. IEEE (2014)
10. Das, P., Ahmed, T., Ali, M.F.: Static hand gesture recognition for American sign language using deep convolutional neural network. In: 2020 IEEE Region 10 Symposium (TENSYP), pp. 1762–1765. IEEE (2020)
11. Ichimura, K., Magatani, K.: Development of the bedridden person support system using hand gesture. In: 2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 4550–4553. IEEE (2015)
12. Kalkan, S.C.: Turkish sign language (fingerspelling) (2018). <https://www.kaggle.com/datasets/feronial/turkish-sign-languagefinger-spelling>. Accessed 29 Dec 2022

13. Kodandaram, S.R., Kumar, N.P., et al.: Sign language recognition. *Turkish J. Comput. Math. Educ. (TURCOMAT)* **12**(14), 994–1009 (2021)
14. Kumar, A., Thankachan, K., Dominic, M.: Sign language recognition, pp. 422–428 (2016). <https://doi.org/10.1109/RAIT.2016.7507939>
15. Latif, G., Mohammad, N., Alghazo, J., AlKhalaf, R., AlKhalaf, R.: ArASL: Arabic alphabets sign language dataset. *Data Brief* **23**, 103777 (2019)
16. Mavi, A.: A new dataset and proposed convolutional neural network architecture for classification of American sign language digits. arXiv preprint [arXiv:2011.08927](https://arxiv.org/abs/2011.08927) (2020)
17. Murray, J.: World federation of the deaf (2018). <https://wfdeaf.org/our-work/>. Accessed 29 Dec 2022
18. Muthukumar, K., Poorani, S., Gobhinath, S.: Vision based hand gesture recognition for Indian sign languages using local binary patterns with support vector machine classifier. *Adv. Nat. Appl. Sci.* **11**(6), 314–322 (2017)
19. Öztürk, A., Karatekin, M., Saylar, İ.A., Bardakci, N.B.: Recognition of sign language letters using image processing and deep learning methods. *J. Intell. Syst. Theory Appl.* **4**(1), 17–23 (2021)
20. Pinto, R.F., Borges, C.D., Almeida, A., Paula, I.C.: Static hand gesture recognition based on convolutional neural networks. *J. Electr. Comput. Eng.* **2019** (2019)
21. Qi, J., Jiang, G., Li, G., Sun, Y., Tao, B.: Surface EMG hand gesture recognition system based on PCA and GRNN. *Neural Comput. Appl.* **32**(10), 6343–6351 (2020)
22. Ranga, V., Yadav, N., Garg, P.: American sign language fingerspelling using hybrid discrete wavelet transform-gabor filter and convolutional neural network. *J. Eng. Sci. Technol.* **13**(9), 2655–2669 (2018)
23. Sadeddine, K., Chelali, F.Z., Djeradi, R., Djeradi, A., Benabderrahmane, S.: Recognition of user-dependent and independent static hand gestures: application to sign language. *J. Vis. Commun. Image Represent.* **79**, 103193 (2021)
24. Sadeghzadeh, A., Islam, M.B.: BiSign-Net: fine-grained static sign language recognition based on bilinear CNN. In: 2022 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), pp. 1–4. IEEE (2022)
25. Sadeghzadeh, A., Islam, M.B.: Triplet loss-based convolutional neural network for static sign language recognition. In: 2022 Innovations in Intelligent Systems and Applications Conference (ASYU), pp. 1–6. IEEE (2022)
26. Sagayam, K.M., Hemanth, D.J.: Hand posture and gesture recognition techniques for virtual reality applications: a survey. *Virtual Reality* **21**(2), 91–107 (2017)
27. Sanchez-Riera, J., Hua, K.L., Hsiao, Y.S., Lim, T., Hidayati, S.C., Cheng, W.H.: A comparative study of data fusion for RGB-D based visual recognition. *Pattern Recogn. Lett.* **73**, 1–6 (2016)
28. Sevli, O., Kemaloglu, N.: Turkish sign language digits classification with CNN using different optimizers. *Int. Adv. Res. Eng. J.* **4**(3), 200–207 (2020)
29. Shi, B., Brentari, D., Shakhnarovich, G., Livescu, K.: Fingerspelling detection in American sign language. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4166–4175 (2021)
30. Yalçın, M., Ilgaz, S., Özkul, G., Yildiz, Ş.K.: Turkish sign language alphabet translator. In: 2018 26th Signal Processing and Communications Applications Conference (SIU), pp. 1–4. IEEE (2018)
31. Zakariah, M., Alotaibi, Y.A., Koundal, D., Guo, Y., Mamun Elahi, M.: Sign language recognition for Arabic alphabets using transfer learning technique. *Comput. Intell. Neurosci.* **2022** (2022)