Article

Spelling Correction Real-Time American Sign Language Alphabet Translation System Based on YOLO Network and LSTM

Miguel Rivera-Acosta ¹ , Juan Manuel Ruiz-Varela ¹ , Susana Ortega-Cisneros ^{1,*} , Jorge Rivera ² ,
Ramón Parra-Michel ¹ and Pedro Mejia-Alvarez ¹

¹ Advanced Studies and Research Center (CINVESTAV), National Polytechnic Institute (IPN), Zapopan 45019, Mexico; marivera@gdl.cinvestav.mx (M.R.-A.); jmruiz@gdl.cinvestav.mx (J.M.R.-V.); ramon.parra@cinvestav.mx (R.P.-M.); pmejia@gdl.cinvestav.mx (P.M.-A.)

² CONACYT—Advanced Studies and Research Center (CINVESTAV), National Polytechnic Institute (IPN), Guadalajara Campus, Zapopan 45019, Mexico; riveraj@gdl.cinvestav.mx

* Correspondence: sortega@gdl.cinvestav.mx; Tel.: +52-33-3777-3600

Abstract: In this paper, we present a novel approach that aims to solve one of the main challenges in hand gesture recognition tasks in static images, to compensate for the accuracy lost when trained models are used to interpret completely unseen data. The model presented here consists of two main data-processing stages. A deep neural network (DNN) for performing handshape segmentation and classification is used in which multiple architectures and input image sizes were tested and compared to derive the best model in terms of accuracy and processing time. For the experiments presented in this work, the DNN models were trained with 24,000 images of 24 signs from the American Sign Language alphabet and fine-tuned with 5200 images of 26 generated signs. The system was real-time tested with a community of 10 persons, yielding a mean average precision and processing rate of 81.74% and 61.35 frames-per-second, respectively. As a second data-processing stage, a bidirectional long short-term memory neural network was implemented and analyzed for adding spelling correction capability to our system, which scored a training accuracy of 98.07% with a dictionary of 370 words, thus, increasing the robustness in completely unseen data, as shown in our experiments.

Keywords: american sign language; artificial intelligence; deep learning; deep neural networks; hand posture recognition

Citation: Rivera-Acosta, M.; Ruiz-Varela, J.M.; Ortega-Cisneros, S.; Rivera, J.; Parra-Michel, R.; Mejia-Alvarez, P. Spelling Correction Real-Time American Sign Language Alphabet Translation System Based on YOLO Network and LSTM. *Electronics* **2021**, *10*, 1035. <https://doi.org/10.3390/electronics10091035>

Academic Editor:
Athanasios Voulodimos

Received: 16 March 2021
Accepted: 13 April 2021
Published: 27 April 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Deafness and hearing loss are severe problems that have been harassing society for many years. Unfortunately, according to data from World Health Organization (WHO) [1], from year to year, the number of people with disabling hearing loss (DHL) will be increasing worldwide. The WHO announces that for 2050 the number of people with DHL will rise from 466 million (2019) to 933 million [2].

These projections highlight the importance of developing a real-time system to facilitate the communication of the people with DHL to those with no knowledge of sign languages. Such relevance can be proven by the extensive research aiming to develop real-time sign language translators with different approaches found in the state-of-the-art.

Sign language is a language that uses hand and facial expressions for communication. It is mainly used by people with DHL for communication with others; nevertheless, hand kinematics are also used for other purposes, such as human-machine interfacing and other human activity recognition (HAR) tasks.

Considering this, computer-based sign language recognition (SLR) is an important topic that could provide many benefits in a wide variety of activities if it is efficiently performed. One of the most common approaches to capture information for handshape

classification is through RGB images. Nevertheless, solving SLR in images in an efficient manner is a challenging task due to many factors. First, images represent a vast amount of information, where most of the scene's area is irrelevant for the interpretation of the gesture. In addition, due to the nature of non-spoken languages, gestures are widely variable in many aspects, handshapes, scene lighting, image noise, sensor specifications, among others. Thus, the recompilation and labeling of images for covering all of the cases represent a hard workload, making it difficult to develop a system that provides good results in unseen data.

Sign language recognition mainly consists of three tasks: hand segmentation within the scene, handshape classification, and hand shape translation. Previous works can be classified by the sensor used for hand segmentation and hand gesture extraction. Microsoft Kinect is a device that provides RGB and depth images. Therefore, hand segmentation is commonly performed using depth information because it is usually closer to sensors than the rest of the user body and background [3,4]. In [3], a colored glove is used for hand segmentation with the Kinect device. These authors proposed a per-pixel approach classifier, where features of a set of points in the hand area are extracted to classify 24 static American Sign Language (ASL) alphabet signs. Nevertheless, due to the size and nature of the Kinect device, portability is a serious drawback for this approach.

Another method for hand feature extraction is to use sensory gloves. This approach avoids the hand segmentation problem. Therefore, many glove-based systems are found in the state-of-the-art [5,6].

This approach yields good results with low computation requirements since no image processing is required for hand features extraction; however, the necessity of a glove is a huge drawback for this methodology, resulting in high implementation costs.

An event-based neuromorphic sensor is used in [7] for the translation of 24 static ASL signs. Their proposed method is based on the generation of a characteristic array that contains information about the contour of the handshape, then a feedforward artificial neural network implemented in a field-programmable gate array (FPGA) is used for the classification of the characteristic array.

As modern algorithms for image processing are emerging, RGB and RGB-D cameras are gaining popularity. Specifically, deep learning and machine-learning algorithms reach high scores in image classification tasks, i.e., deep neural network (DNN) and support vector machine (SVM) have already been applied in SLR [8,9].

In recent years, the convolutional neural network (CNN) algorithm is gaining relevance in image classification due to the promising results obtained in real-world applications. Therefore, many CNN-based sign language translators are found in the state-of-the-art [10–12]. Although CNNs are excellent tools for classifying handshapes in SLR, they lack hand segmentation capability, requiring specialized algorithms for this task. Thus, DNNs for object detection in images are being proposed and implemented to perform SLR, such as faster R-CNN [13,14], where the real-time translation is being reported.

Razieh Rastgoo et al. [15] used a multimodal approach for handshape recognition using static visual data. Their proposal is completely based on deep learning; a fine-tuned CNN is used to detect hands in the input images, and seven restricted Boltzmann machines are used to recognize the final hand label. The authors use two modalities of input, RGB images and depth information. The system was trained using four open-sourced datasets, one RGB image dataset, and three RGB-depth images datasets. Nevertheless, experimental results presented by authors are based on noisy images from datasets (noise was introduced artificially).

Modern devices are emerging, yielding the possibility of designing highly efficient HAR systems based on trainable models. A leap motion controller (LMC) is being used for this purpose. The LMC is an optical sensor, which captures the movement of user hands allowing researchers to perform handshape recognition without requiring special algorithms for hand segmentation.

Since LMC automatically detects and tracks user's hands, it is used to classify dynamic gestures, as in [16], where authors proposed a dynamic gesture recognition system. Their

approach consists of the following stages: a series of frames are captured from the LMC. These frames contain a set of parameters, such as the acceleration, speed, and position of the hand, palm, and fingers information. Using this information, they compute a set of features, such as Euclidian distance between a fingertip and the palm, angles, among others; these array features are concatenated to create a full dynamic gesture vector. Lastly, a bidirectional long short-term memory is used to classify the time-based gesture vector. In addition to their proposal scored an accuracy of 95.23%, the training dataset contains only 312 samples, which is significantly smaller than our image datasets.

Jordan J. Bird et al. [17] proposal is based on a fused modality of LMC and visual data, according to results presented by authors (in unseen data); this multimodal approach scored 76.5% accuracy.

A mixed CNN-LSTM model called DeepConvLSTM for the classification of 60 ASL signs with a leap motion sensor is presented in [18]. The authors used data augmentation to reduce overfitting for maximum testing accuracy of 91.1% using the DeepConvLSTM model, which outperformed recurrent neural networks for SLR. Thus, according to the authors, this demonstrates the importance of using convolutional layers for HAR. Nevertheless, as shown by [17], besides portability loss, RGB sensors score higher accuracy than LMC in unseen data.

A dynamic hand gestures (air-written digits) feedforward network-based recognition system is proposed in [19]. The authors use an inertial measurement unit (IMU); because their model is implemented on an FPGA device and taking advantage of FPGA parallelism, their system can learn in real time from time-dependent IMU data.

As proved by the state-of-the-art proposals in sign language recognition, for many years, researchers have been focusing on increasing sign recognition accuracy by using more complex algorithms or modern and complex sensors, even multiple sensors. Besides that, this approaches considerably reduces portability it increases overall system complexity and costs.

However, focusing on increasing recognition accuracy, partially solves the problem for two reasons; at first, in most literature, reported accuracy is calculated using data from the same dataset or captured under similar conditions. Therefore, this accuracy is expected to be drastically lower under real-world conditions. The second reason is that previous proposals have a major drawback that must be highlighted; the predictions of the recognition systems based on a single sensor often depend only on the result of a single classifier that only considers information of present time. Thus, if their classifier makes a wrong prediction, the SLR system's output will be wrong.

On the other hand, for solving the above-mentioned issues, we are proposing a handshape classifier whose output does not depend on a single prediction, but information from past, present and future is used for computing the system output, increasing the robustness of our system in real-world conditions, as shown in our experiments.

This problem yields the main motivation of this work, to propose a methodology to compensate the accuracy loss of SLR in RGB images when unseen data are presented to a trained model; this, with a single sensor under real-time conditions. Initially, we analyze two models of DNNs for performing the classification of 24 classes of ASL alphabet signs plus two additional signs, i.e., 26 static signs in total.

These experiments were carried out with completely unseen data (images were captured in different background scenes and with a different camera sensor) to support our hypothesis.

With these experiments' findings, we select the most suitable DNN model; selection metrics are the model with the highest accuracy while accomplishing real-time restrictions.

Following, we present an approach for dismissing DNN misclassifications with a bidirectional long short-term memory (LSTM), i.e., the LSTM is used for performing spelling corrections in the predictions of the DNN. Next, the results of a series of experiments of the here presented multimodality model with a community of 10 users are presented. Results

have shown that our proposal can achieve higher accuracy than those in the state-of-the-art in the field of sign language recognition in RGB images.

Figure 1 shows a block diagram of the process performed by the here presented American Sign Language alphabet translator (ASLAT). It mainly consists of three processing stages. The first stage is an image to letter conversion, which performs the sign language recognition task, this stage contains an image resizer and an object detection DNN, whose input is the image streaming from the camera, and the output consists of a time-series of letters that are the predictions of the DNN. The second stage is a letter sequence to word conversion that removes all equal consecutive predictions (e.g., RREEEDD is converted into RED). The third data processing stage is a word-spelling correction module, consisting of a bidirectional LSTM neural network, whose input is the reduced letter sequence, and the output is the predicted translated word of its dictionary. In our work, this correction feature is designed to compensate for accuracy lost in the ASL alphabet translator. Nevertheless, it can be used to enhance the results in other DNN applications by considering past, present and future predictions.

Figure 1. Processing stages of proposed American Sign Language alphabet translator.

In summary, the main contributions of this work are listed below:

- A novel approach for solving SLR is proposed, where output does not depend on a single algorithm (classifier), but a second stage is added for considering past, present and future input information, thus, increasing the robustness of the system in unseen data;
- Proof that with our SLR multimodality model, interpretation accuracy of complex words can be improved by adding a bidirectional LSTM for spelling correction concerning state-of-the-art approaches;
- A methodology to generate SLR systems using regression DNNs yielding to high accuracy, at real-time speeds, which was demonstrated in our experiments with different users;
- An open-source ASL dataset that can be used for training ASL translation systems and natural language processing algorithms.

The rest of this paper is organized as follows: Section 2 presents the methodology and resources used during developing this work. Section 3 contains the results of our proposal. Section 4 presents the discussions.

2. Materials and Methods

Our proposal is based on an open-access top object detection network, trained with an open-access ASL alphabet dataset [20], and fine-tuned with an own created image dataset called ASLYset.

In addition, a semi-character level word spelling correction (WSC) was designed. The purpose of the WSC is to add the SLR output sequence to word conversion and spelling correction features. Currently, its dictionary contains 370 words (words WSC

is trained to correct, including pronouns, colors, numbers, animals, names, verbs, and objects). However, since artificial intelligence is used, it can correct more words by properly generating the training dataset.

The here presented ASLAT is based on a graphical user interface, which contains a frame showing real-time images captured by the camera, and three lines of text, containing, SLR response of current image, sentence obtained from sign sequence to word converter and the corrected sentence resulted from WSC, respectively, in descending order, as shown in Figure 2. However, these lines were added for testing purposes since a text-to-speech stage will be added in future releases.

Figure 2. Proposed American Sign Language alphabet translator graphical user interface.

The experiments for testing the here presented ASLAT for the processing of live camera streaming (Figure 3) were carried out with two hardware configurations using Nvidia GPUs (1×4 GB GTX-1050 and 1×8 GB GTX-1080, Santa Clara, CA, USA) [21].

Figure 3. American Sign Language alphabet translator experiments environment.

Three groups of user volunteers participated in the generation of the datasets and testing experiments of this ASLAT. The first group consisted of 4 persons (male) aged between 23 and 33 years. They generated the ASLYset used for fine-tuning the DNN networks. The second consisted of 5 persons (3 persons of the first group plus two volunteers), which generated the testing dataset (not considered for training nor fine-tuning) used for calculating the results presented in Section 3. The third group, formed by 10 persons (8 males and 2 females) aged between 23 and 35 years, tested the whole ASLAT in real-time conditions with the word spelling correction feature included.

The open-source ASL alphabet dataset and the ASLYset were used to train two different DNN architectures with three different input image sizes, which were analyzed to

select the architectures with higher mean average precision (mAP) [22] while maintaining real-time speed to be included in the ASLAT.

2.1. YOLO Networks

You only look once (YOLO) is an open-source object detection model developed by Joseph Redmond et al. [23–25]. YOLO networks use a single CNN to predict bounding boxes and class probabilities for those boxes in a single network evaluation. According to the authors, this behavior allows the YOLO algorithm to detect objects in images at a faster frame rate with higher accuracy than other object detection algorithms, such as deformable parts models [26], R-CNN [27], Fast R-CNN [28], and faster R-CNN [29].

In the state-of-the-art, YOLO networks are commonly used for many applications in the field of object detection in images, mainly due to their capacity to perform real-time video processing with high accuracy. YOLO networks were successfully applied in pedestrian detection [30], vehicle license plate location [31], medical applications as in [32], and sign language recognition [33], where 99.91 (mAP@50) is reported.

2.2. Images Dataset

To design a functional ASLAT, a proper image dataset must be generated, and every image must be labeled to train the YOLO networks. These images must accomplish some requirements to increase the robustness of the ASLAT, such as variation in orientation, distance (camera to hand), hand shape, light conditions, among others. For the training of DNNs, an open-source ASL alphabet dataset was used [20], which consists of 87,000 RGB images of 29 classes, of which 26 are letters A–Z and 3 classes for SPACE, DELETE, and NOTHING. The images have a size of 200×200 pixels and were generated with variations in terms of persons, background and lighting conditions. However, for the purpose of this work, we used 24,000 images of this dataset, consisting of 1000 images for every A–Y letter. Since the here presented ASLAT is based on static image processing, J and Z signs were excluded from the experiments due to their movement-based nature. The 24,000 images of the ASL alphabet dataset were labeled with YOLO format using LabelingMaster [34] software, the bounding box was added in the hand shape only, so the most relevant features of every ASL sign are prioritized, as can be seen in sign “A” (marked with a blue box) of Figure 4.

Figure 4. Samples of the American Sign Language fingerspelling dataset.

We generated an ASL alphabet fingerspelling dataset (called ASLYset) for fine-tuning the DNNs. All the images in the ASLYset were captured using an RGB camera HP Wide Vision HD camera. This dataset was generated with four users (non-speakers of ASL) who were asked to perform the signs shown in Figure 5 [7].

It consisted of 5200 images of size 416×416 , volunteers hand-spelled 24 ASL alphabet signs (whole alphabet excluding “J” and “Z”) and two additional signs called “SP” and “FN”, each volunteer generated 50 images for each of the 26 signs. All of the images were labeled for object detection using YOLO format; however, this dataset could be useful for training or testing other computer vision algorithms (e.g., faster R-CNN) by properly labeling images. The image labeling was performed using the methodology mentioned above where only the hand shape was marked (arm position and orientation were not considered), so the most relevant features of each sign were prioritized for learning. The ASLYset was divided into two categories, training (ASLYtrain) and testing (ASLYtest) datasets of 3900 and 1300 images, respectively.

This dataset can be downloaded in <https://data.mendeley.com/datasets/xs6mvhx6rh/1> (accessed on 14 April 2021).

Figure 5. American Sign Language alphabet used in ASLYset.

A sample of the generated ASLYset resized to 352×352 can be seen in Figure 6.

Figure 6. Samples of the signs H, R, K, F, A, and C of the ASLYset.

Presented experimental results were obtained using a dataset generated for this specific purpose (not considered for training) (this testing set will be called testset in the rest of this paper). The testset contained 1300 images from five volunteers. The testing dataset consisted of 50 images per sign. All the images were captured with an e-CAM132_TX2 RGB camera (different sensors than training and fine-tuning datasets) and in two different background scenes. A sample of the testset resized to 352×352 can be seen in Figure 7.

Figure 7. Samples of the signs B, E, I, S, G, and D of the testing dataset.

The ASLYset dataset contains the following contributions:

- All of the images contained in this dataset were captured in a complex background scene, with different persons and handshapes; this allows training DNNs with more effectiveness for real-world conditions;
- Researchers in the field of computer vision for natural language processing, sign language recognition, and human–computer interfacing using handshape commands can benefit from this dataset;
- These data can be used to train DNNs for object detection in images to perform American Sign Language alphabet translation;
- The dataset does not contain only images for ASL alphabet translation, but every image is labeled with YOLO format for a faster and easier usage of this data;
- The good practical results obtained by using DNNs greatly depend on the quality and amount of data used for its training; this dataset can be used to train a YOLO network or to enrich another dataset for sign language recognition.

2.3. YOLO Network Models

To obtain a real-time ASLAT with high mAP, some experiments were performed to select the proper YOLO network architecture for this specific application. For this purpose, two architectures (called A1 and A2 in the rest of the paper) (Figure 8) were tested and analyzed. The architectures are based on the network YOLOv3-tiny and YOLOv3, respectively [35]. Each architecture was implemented with three different input image sizes to obtain a wider variety of options for selecting the most suitable architecture. The input image sizes are 288×288 , 352×352 , and 416×416 , based on sizes used by authors in [24].

Figure 8. A1 and A2 You only look once (YOLO) architectures used for the experiments.

Table 1 shows the layer parameters of YOLO models A1 and A2.

Table 1. Architectures A1 and A2 layers parameters description.

A1 (416×416) ¹	A2 (352×352) ¹	
C1— $416 \times 416, 32 \times 3 \times 3, 1$	C1— $352 \times 352, 32 \times 3 \times 3, 1$	C22— $11 \times 11, 256 \times 1 \times 1, 1$
P2— $416 \times 416, 2 \times 2, 2$	C2— $352 \times 352, 64 \times 3 \times 3, 2$	U23
C3— $208 \times 208, 64 \times 3 \times 3, 1$	C3— $176 \times 176, 32 \times 1 \times 1, 1$	C24— $22 \times 22, 256 \times 1 \times 1, 1$
P4— $208 \times 208, 2 \times 2, 2$	C4— $176 \times 176, 64 \times 3 \times 3, 1$	C25— $22 \times 22, 512 \times 3 \times 3, 1$
C5— $104 \times 104, 128 \times 3 \times 3, 1$	C5— $176 \times 176, 128 \times 3 \times 3, 2$	C26— $22 \times 22, 256 \times 1 \times 1, 1$
P6— $104 \times 104, 2 \times 2, 2$	C6— $88 \times 88, 64 \times 1 \times 1, 1$	C27— $22 \times 22, 512 \times 3 \times 3, 1$
C7— $52 \times 52, 256 \times 3 \times 3, 1$	C7— $88 \times 88, 128 \times 3 \times 3, 1$	C28— $22 \times 22, 256 \times 1 \times 1, 1$
P8— $52 \times 52, 2 \times 2, 2$	C8— $88 \times 88, 256 \times 3 \times 3, 2$	C29— $22 \times 22, 512 \times 3 \times 3, 1$
C9— $26 \times 26, 512 \times 3 \times 3, 1$	C9— $44 \times 44, 128 \times 1 \times 1, 1$	C30— $22 \times 22, 93 \times 1 \times 1, 1$
P10— $26 \times 26, 2 \times 2, 1$	C10— $44 \times 44, 512 \times 3 \times 3, 1$	D31
C11— $26 \times 26, 1024 \times 3 \times 3, 1$	C11— $44 \times 44, 1024 \times 3 \times 3, 2$	C32— $22 \times 22, 128 \times 1 \times 1, 1$
C12— $26 \times 26, 256 \times 1 \times 1, 1$	C12— $22 \times 22, 512 \times 1 \times 1, 1$	U33
C13— $26 \times 26, 512 \times 3 \times 3, 1$	C13— $22 \times 22, 1024 \times 3 \times 3, 1$	C34— $44 \times 44, 128 \times 1 \times 1, 1$
C14— $26 \times 26, 93 \times 1 \times 1, 1$	C14— $22 \times 22, 512 \times 1 \times 1, 1$	C35— $44 \times 44, 256 \times 3 \times 3, 1$
D15	C15— $22 \times 22, 1024 \times 3 \times 3, 2$	C36— $44 \times 44, 128 \times 1 \times 1, 1$
C16— $26 \times 26, 128 \times 1 \times 1, 1$	C16— $11 \times 11, 512 \times 1 \times 1, 1$	C37— $44 \times 44, 256 \times 3 \times 3, 1$
U17	C17— $11 \times 11, 1024 \times 3 \times 3, 1$	C38— $44 \times 44, 128 \times 1 \times 1, 1$
C18— $52 \times 52, 256 \times 1 \times 1, 1$	C18— $11 \times 11, 512 \times 1 \times 1, 1$	C39— $44 \times 44, 256 \times 3 \times 3, 1$
C19— $52 \times 52, 93 \times 1 \times 1, 1$	C19— $11 \times 11, 1024 \times 3 \times 3, 1$	C40— $44 \times 44, 93 \times 1 \times 1, 1$
D20	C20— $11 \times 11, 93 \times 1 \times 1, 1$	D41
	D21	

¹ Cxx is a convolutional layer with parameters $IW \times IH, NF \times FW \times FH, ST$, Pxx is pooling layer with parameters $IW \times IH, FW \times FH, ST$, where IW, IH, NF, FW, FH , and ST are input feature map width and height, and number, filters width, and height and stride, respectively.

2.4. Word Spelling Correction

As previously mentioned, the main processing module of the WSC is a bidirectional LSTM-based Recurrent Neural Network (RNN) [36]. Its input consists of the sequence of letters obtained from the SLR subsystem. Nevertheless, as seen in Figure 1, to decrease the number of letters that form the output phrase, the ASLAT contains a signs sequence to word converter, which omits equal subsequent letters, this yields to the misspelling of words that contains this property (e.g., HELLO, DIFFERENCE); however, this can be dismissed by the WSC. Since the RNN is trained to receive a single word at a time, the WSC is required to detect when all the letters to represent the desired word were issued to

the SLR subsystem; this is achieved by issuing the “SP” sign (Figure 5). Sign “FN” is used to clear the output phrase. Nevertheless, this can be used to include the text-to-speech capability in future releases.

RNN Architecture

The used semi-character level RNN is based on the work presented in [37], where input word codification and methodology to generate training words were used to design our WSC. As above-mentioned, each time step (t) is represented by a word issued to the RNN. Every word is encoded in a vector of size $3N$, where N is the number of possible letters forming the words (24 ASL alphabet signs). Thus, the size of the input vector (x) of the RNN is 72. Vector (x) is formed by the concatenation of three sub-vectors (b , i , e) corresponding to the position of the letter in the word as seen in the following equation:

$$x(t) = (b(t), i(t), e(t)) \quad (1)$$

Vectors b and e are one-hot representations of the beginning and ending letters of the word, respectively, and vector i represents internal letters. For example, the word “ENVIRONMENT” is encoded as $b = \{E = 1\}$, $i = \{N = 3, V = 1, I = 1, R = 1, O = 1, M = 1, E = 1\}$, $e = \{T = 1\}$, with remaining elements equal to zero. The output of the RNN is a classification dense layer using softmax activation function (calculated with the next equation), x^n is the output n of LSTM dense layer, and with W number of outputs, where W is the number of words in the dictionary (370 for the experiments), as shown in Figure 9:

$$y(n) = e^{x^n} / (\sum_{n=1..w} e^{x^1} + e^{x^2} + e^{x^3} + \dots + e^{x^W}), \quad (2)$$

Figure 9. Recurrent neural network (RNN) proposed model.

2.5. Spelling Correction Dataset

Since most of the open-sourced spelling correction datasets are extracted from digital documents, their misspelled words represent mostly keyboard typing errors. On the other hand, the WSC is intended to correct SLR misclassifications, which greatly depends on ASL sign similarities (such as letters “T” and “N” see Figure 5). The RNN training dataset is based on 235 English sentences used on a daily basis. Using these sentences, 11,750 training sentences (formed by incorrect words) were generated by adding errors by applying the following methodology. All words containing three or more letters are randomly modified

by replacing a single letter of vector i with one of its most similar ASL signs. For example, the word “GRATEFUL” was modified, resulting in incorrect words, such as “GRAMEFUL” and “GRATSFUL”. In addition, as previously mentioned, one of the equal subsequent letters is removed, i.e., the word “APPOINTMENT”, was modified forming the words “APOINTSENT” and “APOINTMENT”.

3. Results

3.1. YOLO Models Results

All the metrics used for selecting the proper YOLO network architecture were calculated using the testset. On the other hand, ASLYtest was used for performing a proper comparison of our method with state-of-the-art proposals. The criterion to select the YOLO network architectures was the architecture with higher mAP that accomplished real-time frames per second (fps) rate; since the here presented translator was intended to process live camera streaming, where 15 and 30 fps were common framerates, 20 fps was considered as real-time in our experiments since it provided smooth video streaming. Table 2 shows the results obtained from the YOLO network architectures presented in Figure 8 by using the testset (see Figure 7), besides that A2 (416×416) with 81.76 percentage resulted in the higher mAP, its 16.65 frame processing rate shown not being sufficient to translate the American Sign Language alphabet properly at real-time speed in the experiments.

On the other hand, A2 (352×352) presented a higher mAP than all the other models while offering a real-time response time. Therefore, this YOLO network model was the most suitable for the ASLAT based on previously mentioned evaluation metrics. Thus, architecture A2 (352×352) was used to carry out the experiments performed by processing live-camera streaming and testing with the WSC system.

Table 2. YOLO Network models performance.

Image Size	Frame-Rate (fps)			mAP @ 50 (%) ASLYtest/testset		
	288×288	352×352	416×416	288×288	352×352	416×416
A1 GTX-1050	47.15	36.00	29.54	97.07/72.66	98.86/74.26	99.46/75.63
A1 GTX-1080	122.24	95.32	72.90			
A2 GTX-1050	29.81	22.85	16.65	98.06/69.87	99.81/81.74	99.85/81.76
A2 GTX-1080	81.79	61.35	46.03			

The per-sign average precision (AP) for every architecture is shown in Figure 10, where signs “M” and “N” are the letters with less AP for most of the models due to the similarities with other signs. Other signs, such as “F” and “X”, showed a large discrepancy in some models. This was the consequence of three main factors. Overfitting for some models and signs—under some lighting and hand rotation conditions, these signs could be misinterpreted as other signs, such as “R” and “W” in the case of sign “F”. In addition, since a different camera sensor was used for calculating the AP and raw camera resolutions and aspect ratios were considerably different, during the image rescaling process, signs could be significantly deformed. However, this could be solved by enhancing the training dataset by including images with varying aspect ratios of different camera sensors or by applying data augmentation as in [18].

Figure 10. YOLO networks average precision per sign using testset.

Since the here presented ASLAT is designed to be used by a single person at a time (in real-time conditions), the SLR subsystem is configured to present the prediction with higher probability. Besides this, the SLR is designed in such a way that a response is issued only if eight (this number is configurable) consecutive images are classified as the same sign.

Figure 11 presents the recall of model A2 (352×352) for every sign using the 1300 images for testing (testset). This represents how successful the model is to detect presented signs since it depends on the true positives and false negatives. This metric is important to our system since, as previously mentioned, eight consecutive signs must be detected to be issued to the letter sequence to word converter (See Figure 1).

Figure 11. Sign language recognition (SLR) system signs detection recall using testset.

Recall was calculated using:

$$\text{Recall}(n) = \text{TP}(n) / (\text{TP}(n) + \text{FN}(n)), 26 \geq n \geq 1, \quad (3)$$

where TP is true positive, and FN is false-negative. The mean recall of YOLO model A2 (352×352) is 71.23%, which was calculated using:

$$\text{Mean recall} = \sum_{n=1..26} \text{Recall}(n)/26, \quad (4)$$

To better understand the performance of A2 (352×352), Figure 12 shows a confusion matrix of the selected YOLO model. The confusion matrix was generated with 50 images per sign (testing set). In Figure 12, rows and columns represent target and predictions, respectively; N/D is not detected. As can be seen in the confusion matrix, where false positives (FP) and false negatives (FN) are shown; signs, such as “X”, “M”, and “F” presents the highest FN, as above-mentioned, overfitting, and sign deformation during image rescaling can explain these misclassifications, and signs with the most similarities, such as, “N”, and “T” presents the higher FP.

A	24	0	0	0	0	0	0	0	0	2	0	0	2	0	0	0	0	0	12	0	0	0	0	0	0	0	10
B	0	48	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	
C	0	0	36	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	13	
D	0	0	0	35	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	13	
E	0	0	0	0	44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	
F	0	0	0	0	0	24	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	24	
G	0	0	1	0	0	0	34	3	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	11	
H	0	0	0	0	0	0	2	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	
I	0	0	0	4	0	0	0	0	0	36	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	9	
K	0	0	0	0	0	0	0	0	0	41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	
L	0	0	0	0	0	0	0	0	0	1	38	0	0	0	0	0	0	0	0	0	0	0	1	0	0	10	
M	0	0	0	0	0	0	0	0	0	0	0	12	10	0	0	0	0	0	1	0	0	0	0	0	0	27	
N	1	0	0	0	0	0	0	0	0	0	0	2	32	0	1	0	0	0	1	0	0	1	0	0	0	12	
O	0	0	1	0	0	0	0	0	0	0	0	0	0	38	2	0	0	0	0	0	0	0	0	0	0	9	
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0	0	0	0	0	0	0	0	0	0	0	
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	48	0	0	0	0	0	0	0	0	0	1	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	0	0	0	0	7	0	0	0	21	
S	0	0	0	0	0	1	0	0	0	0	0	1	4	0	0	0	0	0	34	2	0	0	0	0	0	8	
T	4	1	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	21	0	0	0	0	0	0	12	
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	44	0	0	0	0	0	0	5	
V	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	41	1	0	0	0	0	0	6	
W	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	38	0	0	0	0	9	
X	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	8	0	0	3	0	0	10	0	0	0	27	
Y	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	39	0	0	0	8	
SP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	47	0	0	3	
FN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0	0	
	A	B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	SP	FN	N/D

Figure 12. YOLO model A2 (352×352) confusion matrix.

3.2. Bidirectional LSTM Results

The RNN was trained to correct words within an own generated dictionary. The dictionary consists of 235 English sentences, which are of common use on a daily basis. A sample is presented in Table 3. Resulting in a training accuracy of 98.07%.

Table 3. Recurrent Neural Network (RNN) training dataset sample.

Correct Sentence	Wrong Sentence
I would like to make an appointment	i would liue to mave an apointsent
is there a restaurant close to us	is theue a restaubant clote to us
oscar is helping me with my homework	oscar is helpsg me winh my honework
hello I am cesar	hslo i am cesyr
I think it tastes good	i think it tases god

3.3. Real-Time Experiment Results

The experiments consisted of processing live camera streaming with previously mentioned hardware, using the model A2 (352×352), an RGB camera HP wide vision HD camera was used for capturing images. The process to test the ASLAT consisted of 10 non-speakers of ASL persons (including the dataset voluntaries) who were asked to spell a sentence found in the dictionary, obtaining results shown in Table 4. Similarly to the process of generating a dataset, users were asked to perform signs with variations in hand position

and allowing natural variations, such as rotation and occlusions, since these problems are often presented in non-controlled environments.

As seen in Table 4, the WSC presents some severe wrong predictions, such as words “licenxe” and “tabout” to “outside” and “vacant”, respectively, besides their Levenshtein distance is 1 concerning ground truth words. For the case of “licenxe”, it generated noisy training words such as the following: lscense, liense, liscnse, licemse, licese, licetse, and licente; “licenxe” was not found in the training dataset (not even letter “x”), which suggest that the LSTM model was overfitting the training noisy words. Therefore, we upgraded the LSTM training dataset by improving our noisy word generation methodology and by optimizing the RNN model.

Table 4. American Sign Language alphabet translator (ASLAT) real-time results.

User	Sentence	A2 (352 × 352) Prediction	WSC
1	I like pancakes	i like patcakes	i like pancakes
2	Is there a restaurant close to us	is there a reistaurat close to us	is there a restaurant close to us
3	Please wait for me	pletse wait for me	please wait for me
4	Andrea plays the piano	atdrea plays the siatosts	andrea plays the piano
5	The boss is busy	the bcs is brusy	the boss is busy
6	We need to rent a room	we ned to rent a rom	we need to rent a room
7	I am getting my license	si am getting my licenxe	of am getting my outside
8	Could be better	could be beaer	could be better
9	I should leave	i shculd letve	i should leave
10	I am not sure about that	xi am not sure tabout thta	of am not sure vacant wanna

Table 5 shows the Levenshtein distance between correct sentence and SLR output and WSC output. Since WSC output represents the whole system response, its misclassifications have huge repercussions in the output sentence, i.e., the sentences of users 7 and 10, where the WSC misclassifications significantly increased Levenshtein distance between ground truth and predictions. This is a major drawback of our system. Therefore, the WSC must be improved to reduce these errors as possible.

Table 5. Levenshtein distance between sentence and A2 (352 × 352) prediction and sentence and word spelling correction (WSC) output.

User	Levenshtein Distance Sentence/A2 (352 × 352) Prediction	Levenshtein Distance Sentence/WSC
1	1	0
2	2	0
3	1	0
4	6	0
5	3	0
6	2	0
7	2	8
8	2	0
9	2	0
10	4	11

A video of the here presented ASLAT working in real time can be seen in the Supplementary Materials.

Table 6 presents a comparison of the herein presented translation system compared to other proposals found in the state-of-the-art. The presented mAP (for our work) corresponded to an accuracy of A2 (352 × 352) network/and accuracy using testset since not all works provide results with completely unseen data.

Table 6. Comparison of the proposed translator with state-of-the-art gesture recognition approaches. Note that not all works are based on visual data.

Proposal	Sensor	Classification Level	Dataset Samples	Accuracy (%)
Ours	RGB	Letters (single sign)	29,200	99.81/81.74 (mAP@50)
[9]	RGB	Letters (single sign)	2400	99.5
[10]	RGB–depth	Letters (single sign)	48,000	99.9
[33]	RGB	Letters (single sign)	4547	99.91 (mAP@50)
[15]	RGB–depth	Letters (single sign)	2524–131,000	99.31–98.13
[19]	IMU	Numbers (dynamic gesture)	1000	98.60
[16]	LMC	Words (dynamic gesture)	300–480	96.67–95.23
[11]	RGB	Letters (single sign)	14,781	95.54
[17]	RGB–LMC	Gestures (dynamic–BSL)	16,498	94.44
[13]	RGB	Letters (single sign)	130,000	93.33
[18]	LMC	Words and numbers	16,890	91.1

4. Discussion

Approximately 6.1% of the world’s population is affected by hearing loss; according to WHO, this number will increase in the future. Therefore, there are many related works aiming to solve hand gesture recognition [7–19]. Modern algorithms and resources are available, which make it viable to develop an easy-to-use ASLAT for common users with spell correction features as herein presented.

Nevertheless, efficiently performing sign language recognition in images is a challenging task due to many factors. First, images represent a vast amount of information, where most of the scene’s area is irrelevant for the interpretation of the gesture. Therefore, the segmentation and classification of the relevant/irrelevant information in every image consume most of the processing time. Thus, a specialized algorithm for performing these tasks is required, increasing the system complexity and making it difficult to fulfill real-time timing requirements. In addition, due to the nature of non-spoken languages, gestures are widely variable in many aspects, including handshape, scene lighting, image noise, sensor specifications, among others. Thus, the recompilation and labeling of images for covering all the cases represent a hard workload, making it difficult to develop a system that provides good results in unseen data.

Experiments have shown that the system presented in this paper can real-time translate 24 ASL alphabet signs, along with two additional signs in an efficient manner, with a mAP@50 of 99.81% at a frame processing rate of 61.35. At real-time conditions, letters with the most similarities, such as “N”, “M”, and “T” can be optimized to improve the precision. However, by adding word spelling correction capacity, these misclassifications are almost fully dismissed in controlled environments, as shown in experiments. Since the text is naturally sequential, RNNs are often used for text classification [38] and word spelling correction [37].

This work presents a proposal for the challenging task of sign language recognition by providing a method for the segmentation, classification and translation of the American Sign Language alphabet in static RGB images. Multiple experiments were carried out with unseen data and images captured with a camera sensor with different specifications than those used for generating the training dataset. Besides, precision with unseen data is low compared to training accuracy, presented results with our novel proposal of adding a second stage of prediction correction shown that the here presented ASL translator is feasible for real-time processing under real-world conditions. However, our proposal has limitations that must be resolved for this.

In the experiments, a set of 26 static gestures were considered for the interpretation of a non-spoken language. Nevertheless, in practice, complex dynamic gestures (where not only the hand is involved) are used for everyday interaction. Thus, additional classes are required to allow the interpretation of gesture spoken languages practically.

In addition, as shown in the results, some signs present drastically lower accuracy in unseen data, one of the causes is the model overfitting, to solve this, the richest dataset must be used for training the models; not only by adding more images under same conditions or using or data augmentation, but more sensors, background scenes, and handshapes must be included in the experiments. In addition, a model pretrained with COCO dataset [39] could be used rather than training models from scratch. This could lead to a better performance in the SLR.

Since DNN is used for the segmentation and classification of handshape, required computation resources are massive to fulfill the timings restrictions to be considered real-time processing. This is particularly challenging for devices with hardware with low levels of parallelism, such as microprocessors. Thus, quantized models must be included in the experiments for analyzing how feasible is the implementation of this ASL translator in mobile devices such as smartphones.

In addition, since RNNs are trained to recognize time-based patterns, they can be used to classify and correct the sequence of words obtained with SLR and WSC subsystems along with movement-based ASL alphabet signs such as letters “J” and “Z”; even complex non-alphabet movement-based ASL signs such as “Hello”. This represents several advantages since currently, all the words that form a phrase must be issued one-per-one, making it difficult to use this ASLAT for long periods. By adding this feature, its output phrase can be displayed as sound for enabling efficient real-time communication between persons affected with hearing loss or speech difficulties with those with no knowledge in sign languages. Currently, the SLR subsystem is configured to present a single sign as classification response (i.e., with higher probability), but, due to the methodology used to design this ASLAT (YOLO–LSTM synergy), a more accurate translation could be achieved by training the WSC to contain as input (for example), the two signs with higher accuracy. This could be useful for some cases. For example (hypothetical case), if the word “leave” is being spelled, when issuing letter “a”, SLR classification results could be, t = 85%, a = 84%; using the current methodology, only the information of letter “t” is considered in the WSC (i.e., “a” information is deprecated besides it is the correct letter). Thus, by considering more than one SLR classification response, a higher accuracy could be achieved. However, this experimentation will be performed in future releases.

Supplementary Materials: The following are available online at <https://www.mdpi.com/article/10.3390/electronics10091035/s1>.

Author Contributions: Conceptualization, M.R.-A., J.M.R.-V., S.O.-C., J.R. and P.M.-A.; methodology, M.R.-A., J.M.R.-V. and S.O.-C.; software, M.R.-A., J.M.R.-V. and J.R.; validation, M.R.-A., J.M.R.-V. and P.M.-A.; formal analysis, M.R.-A., J.M.R.-V. and S.O.-C.; investigation, M.R.-A. and J.M.R.-V.; resources, S.O.-C. and P.M.-A.; writing the paper, M.R.-A., J.M.R.-V., S.O.-C., J.R. and P.M.-A.; Reviewing the paper, M.R.-A., J.R. and R.P.-M.; funding acquisition, R.P.-M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by CONACYT, grant number 336595.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. World Health Organization. Deafness and Hearing Loss. Available online: <https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss> (accessed on 9 September 2020).
2. World Health Organization. *WHO Global Estimates on Prevalence of Hearing Loss, Prevention of Deafness WHO*. 2018. Available online: <https://www.who.int/deafness/Global-estimates-on-prevalence-of-hearing-loss-for-website.pptx?ua=1> (accessed on 9 September 2020).
3. Dong, C.; Leu, M.C.; Yin, Z. Sign Language Alphabet Recognition Using Microsoft Kinect. In Proceedings of the 2015 IEEE Conference on CVPRW, Boston, MA, USA, 7–12 June 2015.
4. Hee-Deok, Y. Sign Language Recognition with the Kinect Sensor Based on Conditional Random Fields. *Sensors* **2015**, *15*, 135–147.

5. Cemil, O.; Ming, C.L. American Sign Language word recognition with a sensory glove using artificial neural networks. *Eng. Appl. Artif. Intell.* **2011**, *4*, 1204–1213.
6. Ognjan, L.; Miroslav, P. Hand gesture recognition using low-budget data glove and cluster-trained probabilistic neural network. *Assem. Autom.* **2014**, *34*, 94–105.
7. Rivera-Acosta, M.; Ortega-Cisneros, S.; Rivera, J.; Sandoval-Ibarra, F. American Sign Language Alphabet Recognition Using a Neuromorphic Sensor and an Artificial Neural Network. *Sensors* **2017**, *17*, 2176. [CrossRef]
8. Jie, G.; Wengang, Z.; Houqiang, L.; Weiping, L. Sing Language Recognition Using Real-Sense. In Proceedings of the 2015 IEEE China Summit and International Conference on Signal and Information Processing (ChinaSIP), Chengdu, China, 12–15 July 2015.
9. Md Azher, U.; Shayhan, A.C. Hand Sign Language Recognition for Bangla Alphabet using Support Vector Machine. In Proceedings of the International Conference on Innovations in Science, Engineering and Technology (ICISSET), Dhaka, Bangladesh, 28–29 October 2016.
10. Wenjin, T.; Ming, C.L.; Zhaozheng, Y. American Sign Language alphabet recognition using Convolutional Neural Networks with multiview augmentation and inference fusion. *Eng. Appl. Artif. Intell.* **2018**, *76*, 202–213.
11. Sarfaraz, M.; Harish, C.T.; Adhyan, S. American Sign Language Character Recognition Using Convolution Neural Network. *Smart Computing and Informatics. Smart Innov. Syst. Technol.* **2018**, *78*, 403–412.
12. Yuanheng, Y.; Yingli, T.; Matt, H.; Yingya, L. Recognizing American Sign Language Gestures from within Continuous Videos. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Salt Lake City, UT, USA, 18–22 June 2018; pp. 2064–2073.
13. Dinesh, S.; Sivaprakash, S.; Keshav, M.; Ramya, K. Real-Time American Sign Language Recognition with Faster Regional Convolutional Neural networks. *Int. J. Innov. Res. Sci. Eng. Technol.* **2018**, *7*, 297–305.
14. Oishee, B.H.; Mohammad, I.J.; Md, S.I.; Al-Farabi, A.; Alving, S.P. Real Time Bangladeshi Sign Language Detection using Faster R-CNN. In Proceedings of the International Conference on Innovation in Engineering and Technology (ICIET), Dhaka, Bangladesh, 27–28 December 2018.
15. Rastgoo, R.; Kiani, K.; Escalera, S. Multi-Modal Deep Hand Sign Language Recognition in Still Images Using Restricted Boltzmann Machine. *Entropy* **2018**, *20*, 809. [CrossRef] [PubMed]
16. Yang, L.; Chen, J.; Zhu, W. Dynamic Hand Gesture Recognition Based on a Leap Motion Controller and Two-Layer Bidirectional Recurrent Neural Network. *Sensors* **2020**, *20*, 2106. [CrossRef] [PubMed]
17. Jordan, J.B.; Anikó, E.; Diego, R.F. British Sign Language Recognition via Late Fusion of Computer Vision and Leap Motion with Transfer Learning to American Sign Language. *Sensors* **2020**, *20*, 5151.
18. Vincent, H.; Tomoya, S.; Gentiane, V. Convolutional and Recurrent Neural Network for Human Activity Recognition: Application on American Sign Language. *PLoS ONE* **2020**, *15*, 1–12.
19. Kim, M.; Cho, J.; Lee, S.; Jung, Y. IMU Sensor-Based Hand Gesture Recognition for Human-Machine Interfaces. *Sensors* **2019**, *19*, 3827. [CrossRef] [PubMed]
20. Akash. ASL Alphabet Image Data Set for Alphabets in the American Sign Language. 2018. Available online: <https://www.kaggle.com/grassknoted/asl-alphabet> (accessed on 9 September 2020).
21. Nvidia, CUDA GPUs. Available online: <https://developer.nvidia.com/cuda-gpus> (accessed on 9 September 2020).
22. Padilla, R.; Passos, W.L.; Dias, T.L.B.; Netto, S.L.; da Silva, E.A.B. A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit. *Electronics* **2021**, *10*, 279. [CrossRef]
23. Joseph, R.; Santosh, D.; Ross, G.; Ali, F. You Only Look Once: Unified, Real-Time Object Detection. *arXiv* **2016**, arXiv:1506.02640.
24. Joseph, R.; Ali, F. YOLO9000: Better, Faster, Stronger. *arXiv* **2016**, arXiv:1612.08242.
25. Joseph, R.; Ali, F. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767.
26. Pedro, F.; Ross, B.; David, M.; Deva, R. Object Detection with Discriminatively Trained Part Based Models. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 1627–1645.
27. Ross, G.; Jeff, D.; Trevor, D.; Jitendra, M. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, 23–28 June 2014.
28. Ross, G. Fast R-CNN. In Proceedings of the International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015.
29. Shaoqing, R.; Kaiming, H.; Ross, G.; Jian, S. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv* **2015**, arXiv:1506.01497.
30. Wenbo, L.; Jianwu, D.; Yangping, W.; Song, W. Pedestrian Detection Based on YOLO Network Model. In Proceedings of the 2018 IEEE International Conference on Mechatronics and Automation, Changchun, China, 5–8 August 2018.
31. Weidong, M.; Xiangpeng, L.; Qi, W.; Qingpeng, Z.; Yanqiu, L. New approach to vehicle license plate location based on new model YOLO-L and plate pre-identification. *IET Image Proc.* **2019**, *13*, 1041–1049.
32. Zuzanna, K.; Jacek, S. Bones detection in the pelvic area on the basis of YOLO neural network. In Proceedings of the 19th International Conference Computational Problems of Electrical Engineering, Banska Stiavnica, Slovakia, 9–12 September 2018.
33. Steve, D.; Nanik, S.; Chastine, F. Indonesian Sign Language Recognition using YOLO Method. *IOP Conf. Ser. Mater. Sci. Eng.* **2021**, *1077*, 012029. [CrossRef]
34. Tzutalin, LabelImg. Git Code. 2015. Available online: <https://github.com/tzutalin/labelImg/> (accessed on 9 September 2020).

-
35. YOLO: Real-Time Object Detection. Available online: <https://pjreddie.com/darknet/yolo/> (accessed on 5 July 2019).
 36. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Computation* **1997**, *9*, 1735–1780. [[CrossRef](#)] [[PubMed](#)]
 37. Keisuke, S.; Kevin, D.; Matt, P.; Benjamin, V. Robust Word Recognition via Semi-Character Recurrent Neural Network. *arXiv* **2017**, arXiv:1608.02214.
 38. Pengfei, L.; Xipeng, Q.; Xuanjing, H. Recurrent Neural Network for Text Classification with Multi-Task Learning. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16), New York City, NY, USA, 9–15 July 2016.
 39. Lin, T.-Y.; Maire, M.; Belongie, S.; Bourdev, L.; Girshick, R.; Hays, J.; Perona, P.; Ramanan, D.; Zitnick, L.; Dollar, P. Microsoft COCO: Common Objects in Context. *arXiv* **2015**, arXiv:1405.0312v3, 2015.