אורט בראודה
ORT Braude College

Software Engineering Department

Israeli Sign Language Letters Recognition

Software Engineering Final Project

Karmiel – July 2019

Authors:
Netanel Azoulay
Roman Koifman

Supervisor:
Prof. Valery Kirzner

Advisor:
Prof. Zeev Volkovich

Contents

1. INTRODUCTION

The Sign Language is not universal, each deaf community has a unique Sign Language that was created and developed naturally. Israeli deaf community uses the Israeli Sign Language, which its spelling based on the Hebrew Alphabet [1]. The aim of this project is to convert Israeli Sign Language letters, given by hand sign, to Hebrew text letters and to enable the user to build words and sentences by using unique signs such as space and delete sign. The conversion is handled by Convolutional Neural Network (CNN) that was presented in Stanford's tutorials. In this project, a specific CNN model is used - CNN 12. After trying several different models, CNN12 achieved the best results in term of accuracy. After training the model on the dataset, it is ready to receive Israeli Sign Language letters, classify them and convert them to text. The input is a frame that is taken by the user camera. A pre-processing is being applied on the extracted frame and afterwards the image is passed through the CNN 12 model. The model uses Convolutional Layers with relatively small filters but proved as working with high accuracy. In the last layers of the model, which are Fully Connected layers, classification is being done and the output is a probability vector of the letters. The highest probability value in the vector, if passed a specific threshold, is converted to a textual letter that is displayed to the user.

2. THEORY

2.1. Israeli Sign Language

The Sign language is a visual language expressed through the hands, face and body. It was not invented artificially, but was created naturally, same as spoken language. The Sign Language is not a universal language; each deaf community develops a unique sign language that they use. Similar to spoken languages, the sign languages differ by their vocabulary and grammar.

The Israeli sign language was created with the growth of the Israeli deaf community and was affected by the immigration waves over the years. Although it is the main sign language in Israel, there are still multiple sign languages in Israel that are being used. The Israeli sign language is not called "The Hebrew sign language" because it was meant to emphasize that the language is standalone and is not related to Hebrew. Yet, the Hebrew Alphabet can be represented by Israeli sign language as described in [Figure1]Sign languages spellings are based upon the local languages spelling and hence the Hebrew Alphabet is being used by the Israeli sign language.

The sign language words are glyphs and gestures. This project focus on letters recognition rather than words recognition, that means processing single image frames.

Figure1 - Israeli sign language alphabet. The Institute for the Advancement of Deaf Persons in Israel [2].

2.2. Deep Learning & Artificial Neural Networks

Deep learning is a branch of machine learning [*Figure 2*], that imitates the work of the human brain in processing data and creating patterns for use in decision making [3] . Deep learning architectures relay on the use of Artificial Neural Networks, which are capable of learning data that is instructed or unlabeled. Deep Learning networks are fed with large data sets of diverse examples, from which the model learns to look for features in order to produce a probability vector that answers a classification problem. The class with the highest probability is commonly chosen as the answer.

The Artificial Neural Networks (or simply Neural Networks) are frameworks for different machine learning algorithms to work together and process data inputs. Neural Networks are built upon layers of Artificial Neurons which contain an Activation function. Activation functions define the output of a Neuron and maps the resulting value to a desired range. Commonly, the Activation function's name is being used as the Artificial Neuron's name. The first layer of a Neural Network is the input layer, which contain input Neurons. The last layer is the output layer, which contains output Neurons. The middle layers are called hidden layers since they are not representing input nor output. Neural networks with multiple hidden layers are named Deep Neural Networks. The process of delivering data from the input layer, through the hidden layers, to the output layer is called feed-forward [3].

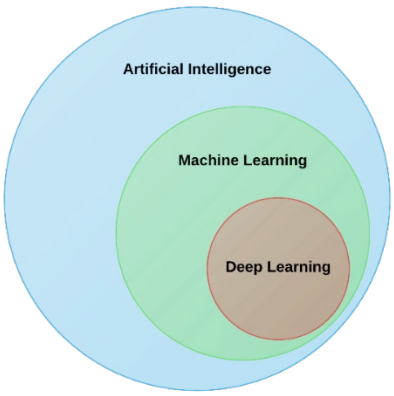


Figure 2– Deep Learning class

2.3. Perceptron's

Perceptron's are the most basic type of Artificial Neurons [5]. The Perceptron takes a set of binary input accompanied by set of real numbers called weights. The weights represent the importance of the inputs. The Perceptron's output is binary and determined by some threshold value compared to the sum of the inputs multiplied by the weights. The threshold is reflected via the bias, that is $bias = -threshold$. Therfore the output could be written as:

$$output = \begin{cases} 0, & if \sum_i w_i x_i + bias \leq 0 \\ 1, & if \sum_i w_i x_i + bias > 0 \end{cases}$$

$Equation\ (1). x_i - represents\ nueron's\ input. w_i - represent\ neuron\ input's\ weight.$

The problem with Perceptrons is that a small change in the weights or bias might cause wrong output to flip, hence, limiting the network from learning by amending weights.

2.4. Sigmoid Function

The Sigmoid Neurons are very similar to Perceptrons, but unlike Perceptrons, a small change in the weights or bias will cause only a minor change in the output [5]. This property allowing the Neural Network to learn. The Sigmoid Neuron has a set of numeric input between 0 and 1, numeric weights and bias. The output is defined as $\sigma(w \cdot x + b)$.

$\sigma\ is\ the\ sigmoid\ activation\ function\ (sometime\ called\ the\ logistic\ function).\ \sigma(z) = \frac{1}{1+e^{-z}}$

The Sigmoid function has similar properties to the Perceptron because $\sigma(z \to \infty) = 1$ and $\sigma(z \to -\infty) = 0$.

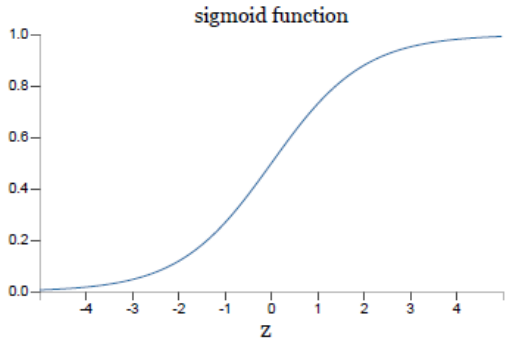


Figure 3 – Sigmoid Function

2.5. ReLU

Rectified Linear Unit (ReLU) is an Activation function which improves the training of Deep Neural Networks [6]. ReLU defined as:

$$f(x) = \begin{cases} x, & if\ x > 0 \\ 0, & if\ x \leq 0 \end{cases}$$

$$Equation\ (2).$$

The ReLU function has better response than the Sigmoid function on a problem called Vanishing Gradient. The problem occurs when an Activation function squashes the input to a very small output range in a non-linear way. Large regions of inputs are being mapped to very small regions, thus, making large changes in the input reflect only small changes in the output (Small Gradient). On the other hand, a main problem with the ReLU function is that all the negative values become zero immediately, hence, decreasing the ability of the model to fit or train from the data properly. This affects the resulting graph by not mapping the negative values appropriately. Leaky ReLU was born to overcome this problem. Instead of applying 0 to all negative input values, the Leak helps to increase the range of the ReLU function by multiply a small number.

The Leaky ReLU [7] function is defined as: $f(x) = \begin{cases} x, & if\ x > 0 \\ \alpha x, & if\ x \leq 0 \end{cases}$ $Equation\ (3).$

Parametric ReLU (PReLU) is a type of leaky ReLU, but instead of having a predetermined slope like 0.01, the PReLu function makes it a parameter for the neural network to determine.

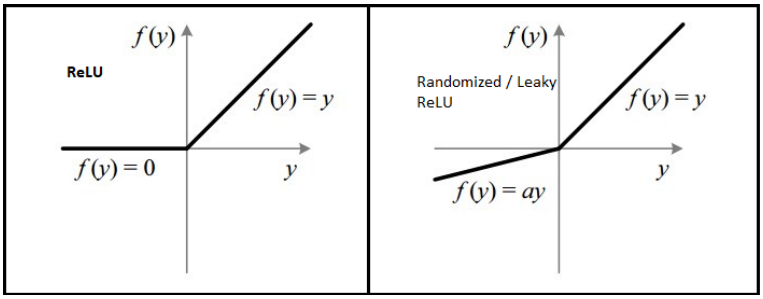


Figure 4 – ReLU vs Leaky / Parametric ReLU

2.6. Pooling Function

A Pooling Layer is used to reduce the spatial dimensions on a Convolution Neural Network [5]. The advantages of reducing spatial dimensions are gaining computation performance [8] and less chance of over-fitting by having less parameters

2.6.1 Max Pooling

Max Pooling is a pooling layer between the Convolutional Layers that uses 2x2 matrices in order to reduce the spatial dimension. Each block of 2x2 in the input, is reduced to 1x1 and the value is the maximum value of the original block. The pooling is performed with a stride of 2 in the CNN 12 architecture, that means that the change in x and y axis will be done by steps of 2. The idea behind Max Pooling is to truncate the parameters that have less influence on the Neural Network, so it can still recognize the image correctly.

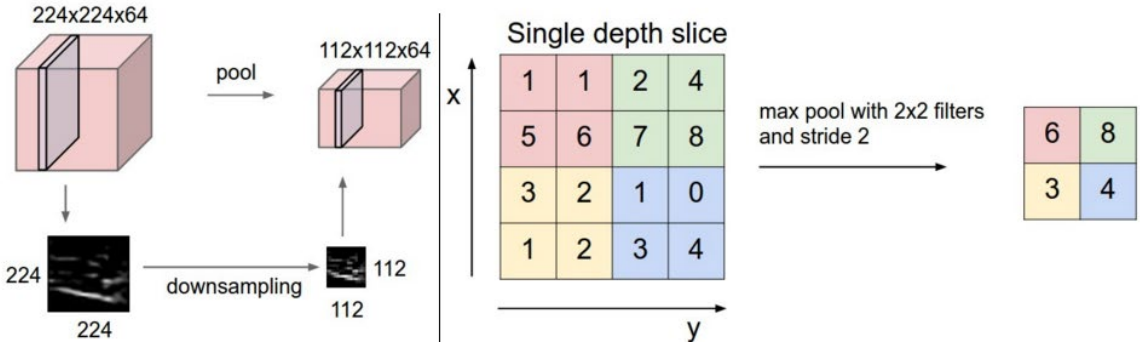


Figure 5 - Applying 2x2 Max Pooling with a stride of 2.

2.6.2 Average Pooling

Average pooling Similar to max pooling layers, are used to reduce the spatial dimensions of a three-dimensional tensor. Average pooling involves calculating the average value for each patch of the feature map. This means that each 2×2 square of the feature map is down sampled to average value in the square. However, these layers perform a more extreme type of dimensionality reduction, where a tensor with dimensions $HxWxD$ is reduced to $1x1xD$. The average pooling reduces each HxW feature map to a single number by simply taking the average of all HxW values.

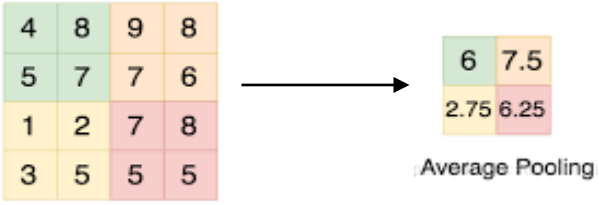


Figure 6 - Applying 2x2 Average Pooling with a stride of 2.

2.7. SoftMax Function

The SoftMax function is used by the last layer of the CNN 12 model in order to generate a probability distribution vector [9]. Each value in the vector corresponds to one of the possible image categories. The function normalizes its input vector $\boldsymbol{x}$ to produce output vector $\boldsymbol{y}$, such, that the output sum is *1*. $\boldsymbol{c}$ is the Temperature coefficient. Upon increasing *c* value, the high distribution's values become even larger and the small values decreases. On the other hand, decreasing *c* value flatten the distribution. That means large values are decreasing and small values are increasing [5]. The SoftMax function is:

$$y_i = \frac{e^{cx_i}}{\sum_{k=1}^{m} e^{cx_k}} \qquad Equation\ (4).$$

2.8. Convolutional Neural Networks

A common class of neural networks is Convolutional Neural Networks (CNN) which are being used in various domains such as speech, audio and image recognition. A major purpose of the CNN is to solve classification problems. The CNN is fed with large data sets of diverse examples, from which the model learns to look for features in order to produce a probability vector that answers a classification problem. The class with the highest probability is commonly chosen as the answer. CNN, similar to the regular Neural Network, contains an input layer, multiple hidden layers and an output layer. The difference is that CNN, apart from having the regular hidden layers, has also hidden layers called Convolutional Layers [9]. The Convolutional layers can detect patterns using filters (which are also the weights). Convolutional layer, like Neuron layer, receives an input and send an output to the next layer. The input is transformed by Convolution operation using a set of filters. When processing an image, The CNN might optionally contain image processing layer which is a layer that contains predefined filters that are kept fixed during training. For sign language gestures recognition, given as images, the CNN is used. The last layers of the CNN are called Fully Connected Layers.

In a Fully Connected (FC) Layer, each output Neuron is connected to each input Neuron with a specific Weight [9]. The collection of the Weights is the parameters set of the FC layer. The connection can be described with a mathematical equation:

$$y_j = \sum_{i=1}^{m} w_{ji} * x_i \qquad Equation\ (5).$$

The equation describes a dot product between the weights *w* and the input Neuron *x*. The output Neuron is *y*.

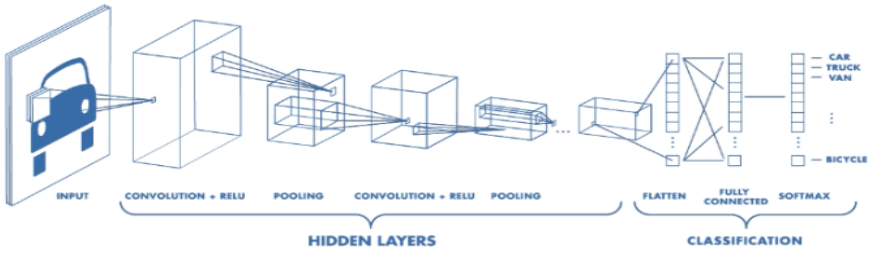


Figure7 - Architecture of a CNN. — Source: https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html

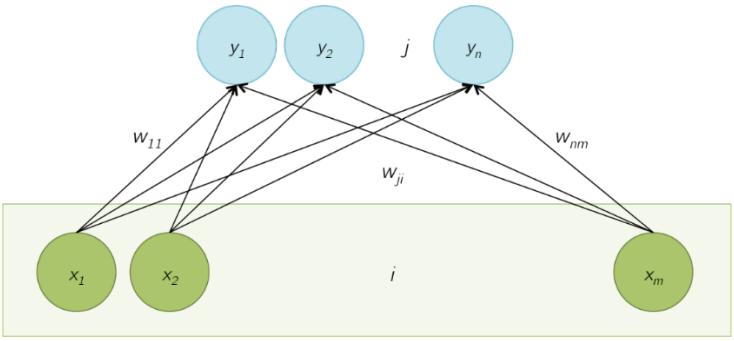


Figure8 - A Fully Connected Neuron Layer

2.9. Back Propagation

Back Propagation is an efficient algorithm intended to calculate the derivative of the cost function with respect to each parameter of the neural network [5]. The cost function (or loss function) estimates how a model is performing respectively to the right output. Loss function range lies between 0 (no errors) and infinity (very bad model performance). The main principal behind back propagation is *error = right answer- actual answer.* Error interval is being calculated after any input that processed through the Neural Network in order to improve the Neural Network reliability. Generally, a greater error that has been discovered will decrease the weights. This is done by updating weights and biases such that will minimise the cost function value. For each feature pixel the function is adjusting up or down. The amount of adjustment is determined by the error value. For example, no errors will cause no change at all. The process of adjustment and minimising the cost function is called gradient descent.

The Backpropagation Learning Rule is given by:

$$\Delta w_{ji} = \eta \delta_j o_i.$$

$$\delta_j = o_j(1 - o_j)(t_j - o_j) \qquad \text{if } j \text{ is an output unit.}$$
$$\delta_j = o_j(1 - o_j) \sum_k \delta_k w_{kj} \qquad \text{if } j \text{ is a hidden unit.}$$

$\eta -$ *Learning Rate constant.*
t_j – *Correct value for unit j.*
o_i – *Output for unit j.*
$\delta_j -$ *Error measure for unit j.*
$Equation$ (6).

2.10. CNN 12

CNN12 architecture is a Convolutional Neural Network with 12 Convolution Layers. The input is 128x128 Grayscale images. Each Convolutional Layer is followed by zero padding, batch normalization and Parametric Rectified Linear Unit (PReLU) activation function. After each two Convolutional Layers a Max Pooling layer is applied. The last pooling layer is Average Pooling instead of Max Pooling. The last two layers are fully connected with applied dropout rate of 30% between them. The last activation function is SoftMax which outputs a probability vector representing 25 classes (22 Letters, space, Delete and Empty class).

The CNN12 was based upon Stanford's tutorials [3] and was used in Hemalatha Vakade's work [12] for classifying camera-trap images of wild animals. The difference between Vakade's architecture between the CNN12 presented here is in the fully connected layers.

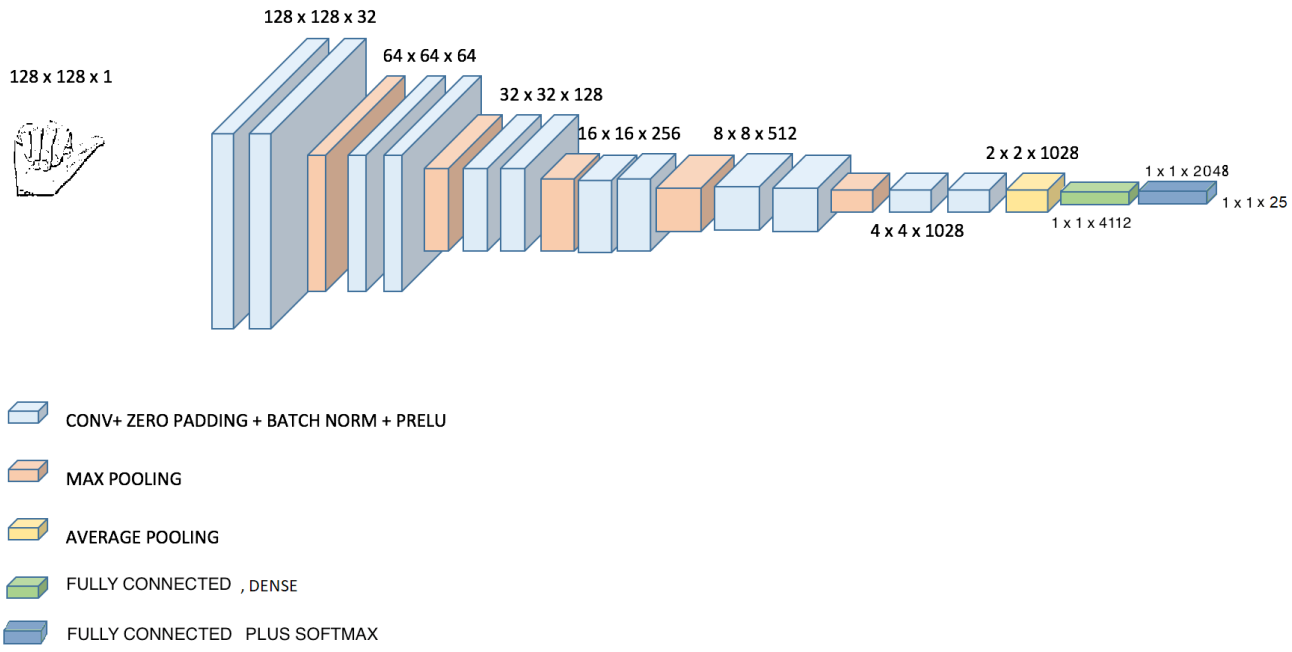


Figure9 – CNN12 Architecture

3. SYSTEM FLOW

The project is based on CNN (CNN 12 model) in order to detect and classify letters in Israeli sign language.

A general diagram illustrating the system flow:

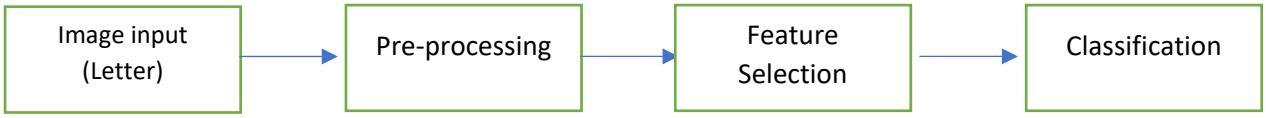


Figure10 – Preprocessing pipe.

3.1 Pre-processing

Images are captured within a Region of Interest (ROI) sized **128x128** from the webcam using OpenCV. To simplify the input images that are analyzed by the CNN12, a binary mask is applied, and the hand's edges are highlighted. The binary mask consists of gray-scaling, blurring, and applying thresholding. The following code snippet is the binary mask application:

```
[1] img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
[2] img = cv2.GaussianBlur(img, (7,7), 3)
[3] img = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
      cv2.THRESH_BINARY_INV, 11, 2)
[4] ret, img = cv2.threshold(img, 25, 255, cv2.THRESH_BINARY_INV +
cv2.THRESH_OTSU)
```

Figure11 – Preprocessing pipe.

[1] Image converted to grayscale.

[2] **Gaussian Blur** is applied in order to remove gaussian noise from the image. The Kernel size is 7x7 and the standard deviation in X and Y direction is (3,3).

[3] **Adaptive Threshold** algorithm calculate the threshold for small regions of the image. The outcome is different thresholds for different regions of the same image which is better for images with varying illumination.

[4] **Threshold** is the simple case of [3]. The algorithm assigns a specific value if pixel value is greater than a threshold value and assigned another value otherwise. The outcome in this case is color inversion.

3.2 Feature Selection

This section consists of repetitive structures, which are mostly the same. The difference is the amount of the Convolutional Filters that are being used (32 to 1028), after every second a Max pooling function is applied, and the size of the analysed matrix decreases. Deeper Layers in the network would be able to detect more abstract features.

Each block [Figure12] contains the following steps:

- Input – the initial input is 128x128 grayscale frame. Afterwards, the output of the convolution block is the input of the next block (in a form of a matrix which its size decreases as the process advances).
- Applying Convolutional Function with 3x3 filters (stride 1) and adding the bias respectively in each calculation. The result is convolved matrix that will be used as an input for the next layer.
- Applying Zero Padding, which pads the matrix with zeros around the border in order to match dimensions of the next convolution block.
- Apply batch normalization. It normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation in order to increase the stability of a network.
- Execute Activation Function Parametric Rectified Linear Unit (PReLU).
- Every second convolutional block Max pooling is applied on the processed matrix (stride 2), resulting a smaller matrix size.

The process is being repeated a total of 12 times, until the last Convolutional block is reached. The different in the last Convolutional block is that it executes an Average pooling function instead of Max pooling function. The result is a small dimensional vector which can be applied to a fully connected layer.

3.3 Classification

After the Convolutional Layers, the processed image attains the three Fully Connected Layers. These Layers along with the implementation of SoftMax Function, convert the data into Probability Vector.

The system output vector dimension is 25x1, each cell in the vector represents a probability of an Israeli Sign Language letter including space and delete signs. The system result is a letter with the highest probability value. If the Probability Vector does not contain a value that passes a pre-defined threshold (80%), the sign is ignored and not displayed.

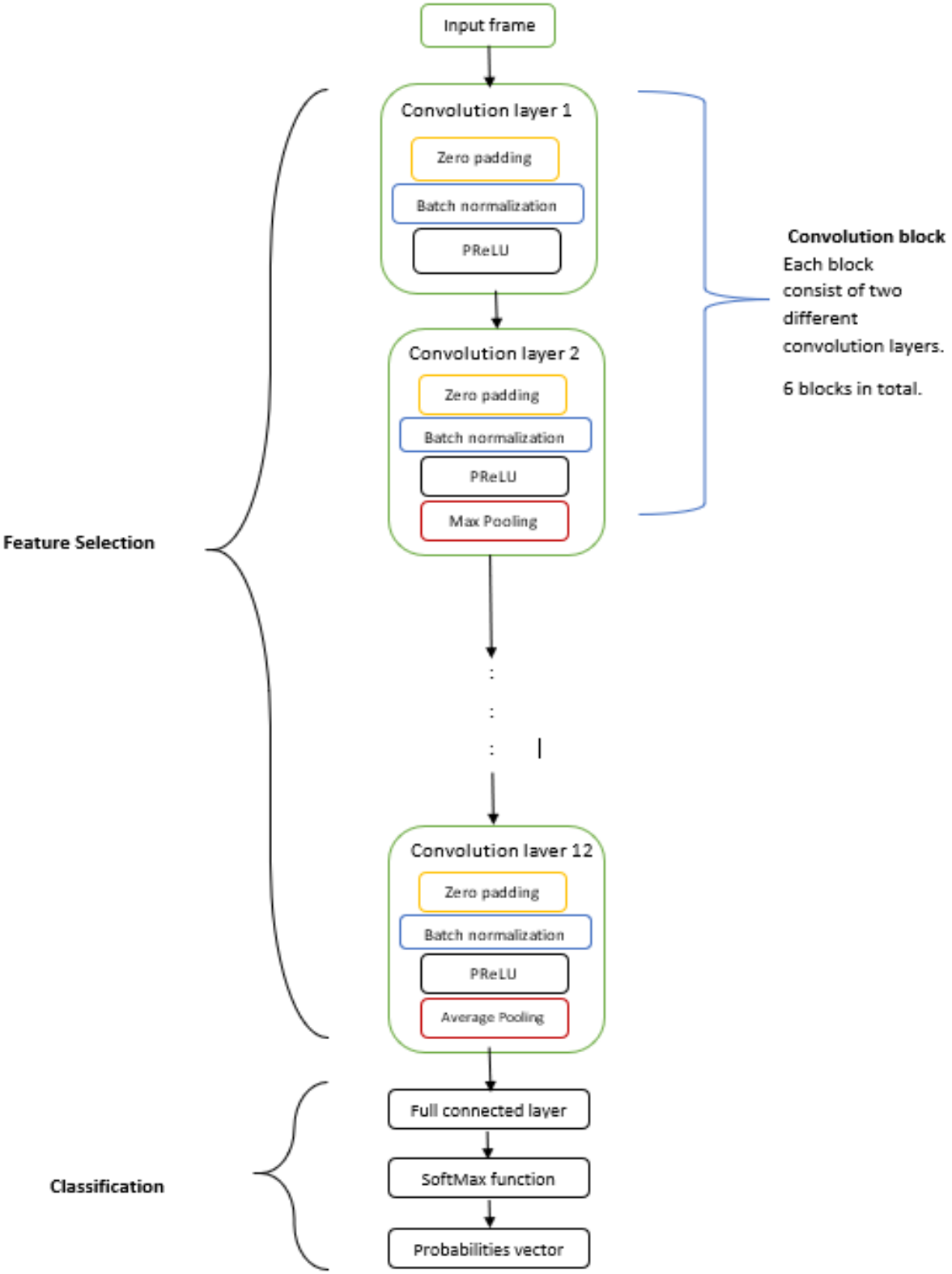


Figure12 – System Architecture

4. TRAINING

a Deep CNN basically consists of discovering the right values on each of the filters.
During the training process, the images are passed through the network by applying several filters on the images at each layer. The values of the filter matrices are multiplied with the activations of the image at each layer. The activations coming out of the final layer are used to find out which class the image belongs to. When a deep network is being trained, the main goal is to find the optimum values on each of these filter matrices so that when an image is propagated through the network, the output activations can be used to accurately find the class to which the image belongs. Later, the error value is being calculated in order to make the network reliable by adjusting the weights in each layer until minimum error is achieved.

4.1. Dataset

The Data Set contains ~150,000 images of 25 classes: 22 Israeli Sign Language letter, space sign, delete sign and empty class containing only white background images. 3,000 images from three different persons (1,000 each) were captured and preprocessed for each class. To enlarge the set, the images were also horizontally flipped resulting 150,000 images total. The images were preprocessed before the training to decrease training time. The Data Set was divided into Train, Test and Validation data.

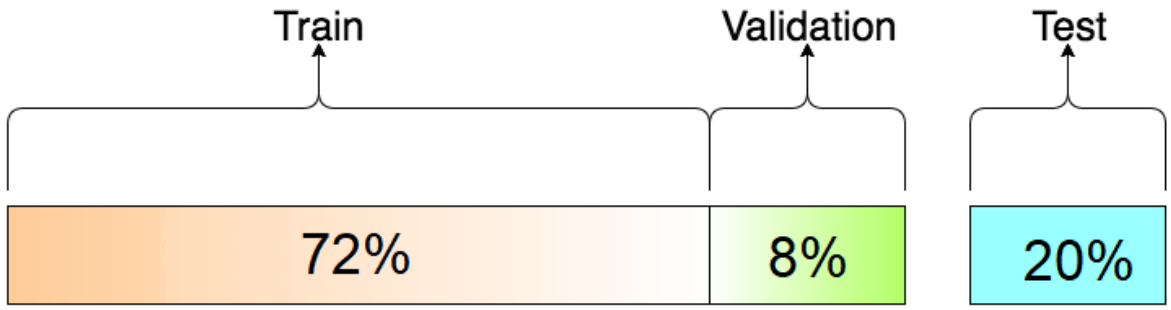


Figure 13 - Dataset division.

The initial split was 80% to the Train data and 20% to the Test data. 10% of the Train data was transferred for the Validation data. The Train data was used to train the model and amend the weights (filters) while the Validation data did not affect the weights but used as indication for model overfitting. The Test data was used in model evaluation test and not in the training process.

4.2. The Training process

The training process took place on CUDA enabled virtual machine via Azure cloud services with Tesla K80 GPU. The model's training configurations that were used are: Standard Learning Rate of 0.001, Batch size of 128 and 50 Epochs. The loss function that was used is Categorical Cross entropy which is a popular function used in image classification models. The model was compiled with Adam optimizer which is adaptive learning rate optimization algorithm that was designed specifically for training Deep Neural Networks.

Four models have been trained:

1. CNN12 without empty class (24 classes) and without dropout regularization.

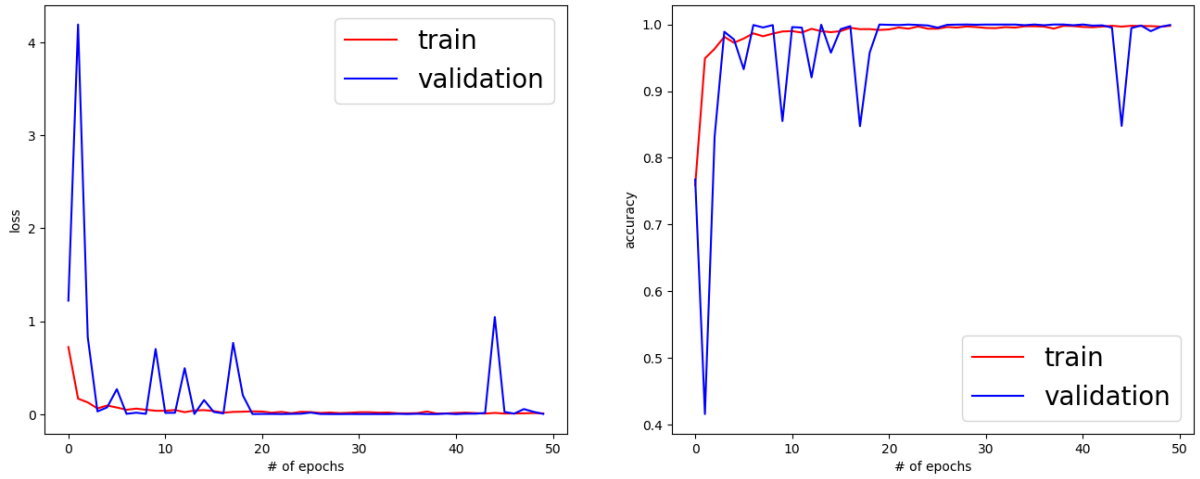


Figure14 - CNN12 Model without empty class and without dropout regularization.

The best accuracy and loss value were between Epoch20 to Epoch41. Using any of these weights yielded a bad performance of the model because overfitting.

2. CNN12 without empty class (24 classes) and with dropout rate of 30%.

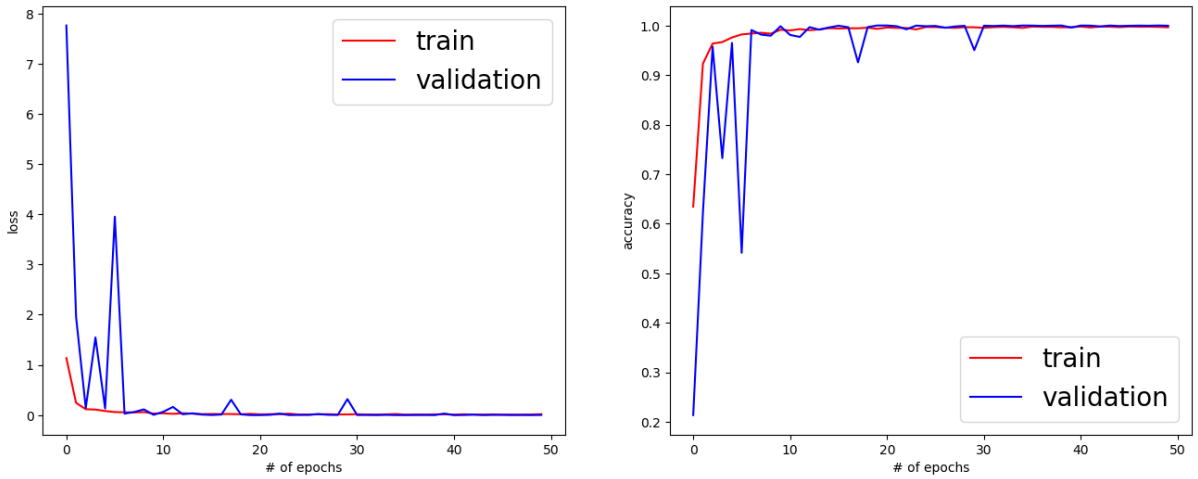


Figure15 - CNN12 Model without empty class and with dropout rate of 30%.

The best accuracy and loss value were from Epoch18. But using any if these weights yielded a bad performance of the model. Blank Input image was recognized as deletion sign with 99% accuracy. Hence, Empty class was added.

3. CNN12 with empty class (25 classes) and without dropout regularization.

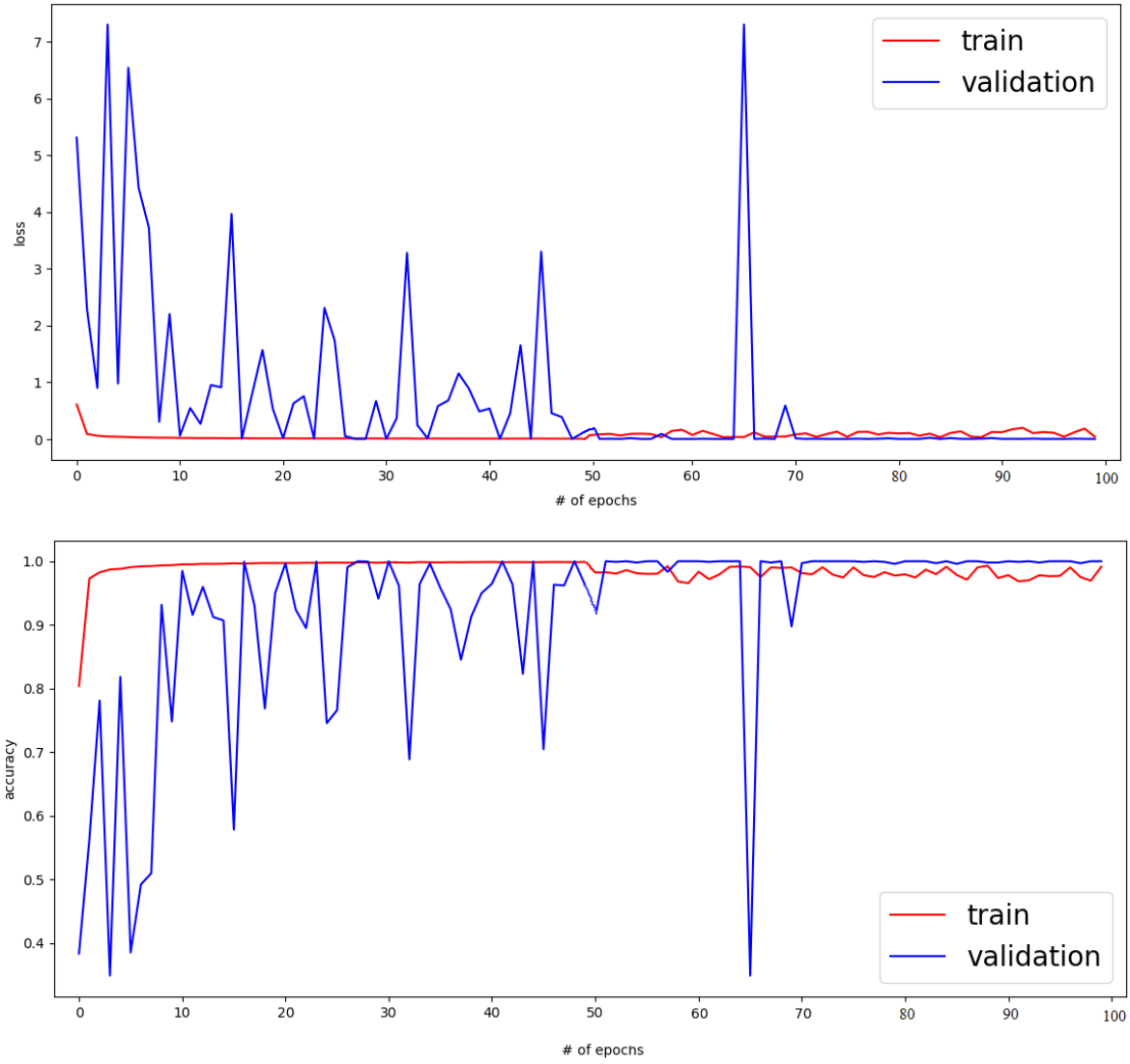


Figure16 - CNN12 Model with empty class and without dropout regularization.

This model had to be trained more than 50 epochs. And yet had bad performance.

4. CNN12 with empty class (25 classes) and with dropout rate of 30%.

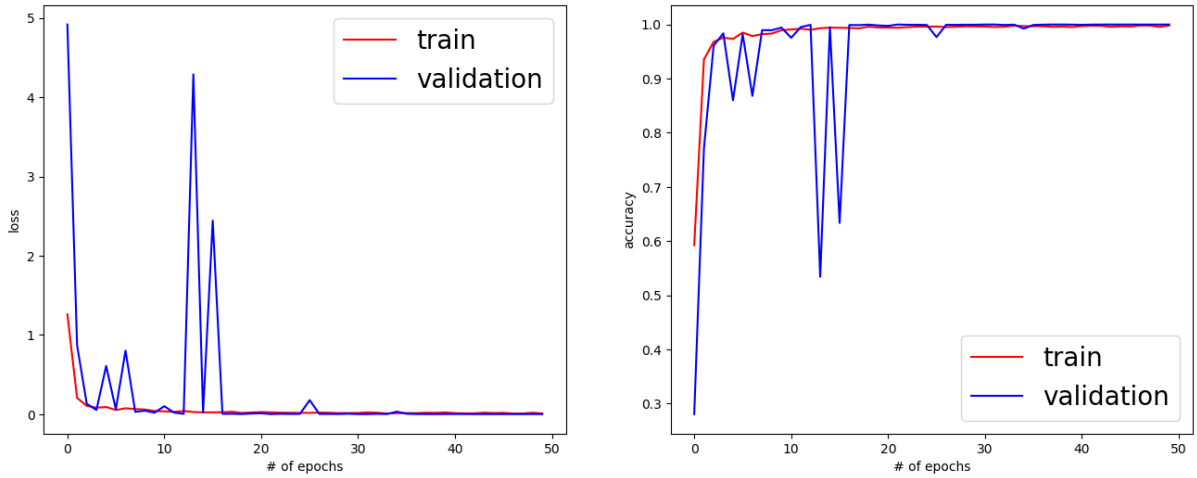


Figure17 - CNN12 Model with empty class and with dropout rate of 30%.

This plot represents the model performance, the best performance that was reached in the research. The weights that has been used are from epoch19 to epoch25.

5. SOFTWARE ENGRINEERING DOCUMENTS

5.1 Requirements and Use-Cases

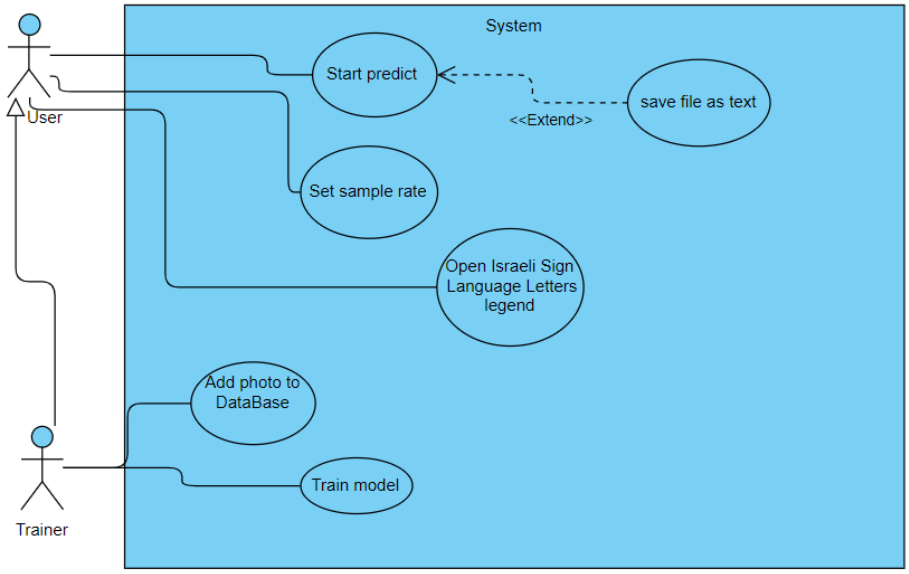
ID	Requirement
1	The System will recognize Israeli Sign Language Letters.
2	The Recognized Letters will be converted into text.
3	The system will use the PC camera in order to analyze in real time.
4	The recognition will be done with CNN 12 model.
5	The Trainer can re-train the model.
6	The user can save the translated text as text file
7	The use can see the Hebrew Sign Language Letters in new window
8	The user can set the sample rate.

Use Case	Sign Language Video Anlyzation
Actors and Goals	User: Receive text output.
Pre-conditions	The user camera is working
Post-conditions	The signs were correctly translated
Trigger	The user clicks on the Predict button.(P)
Main Success Scenario	The user click on the Predict button.(P). The system analyzes the video. The system outputs the recognized text.
Branching	-
Further Requirements	-

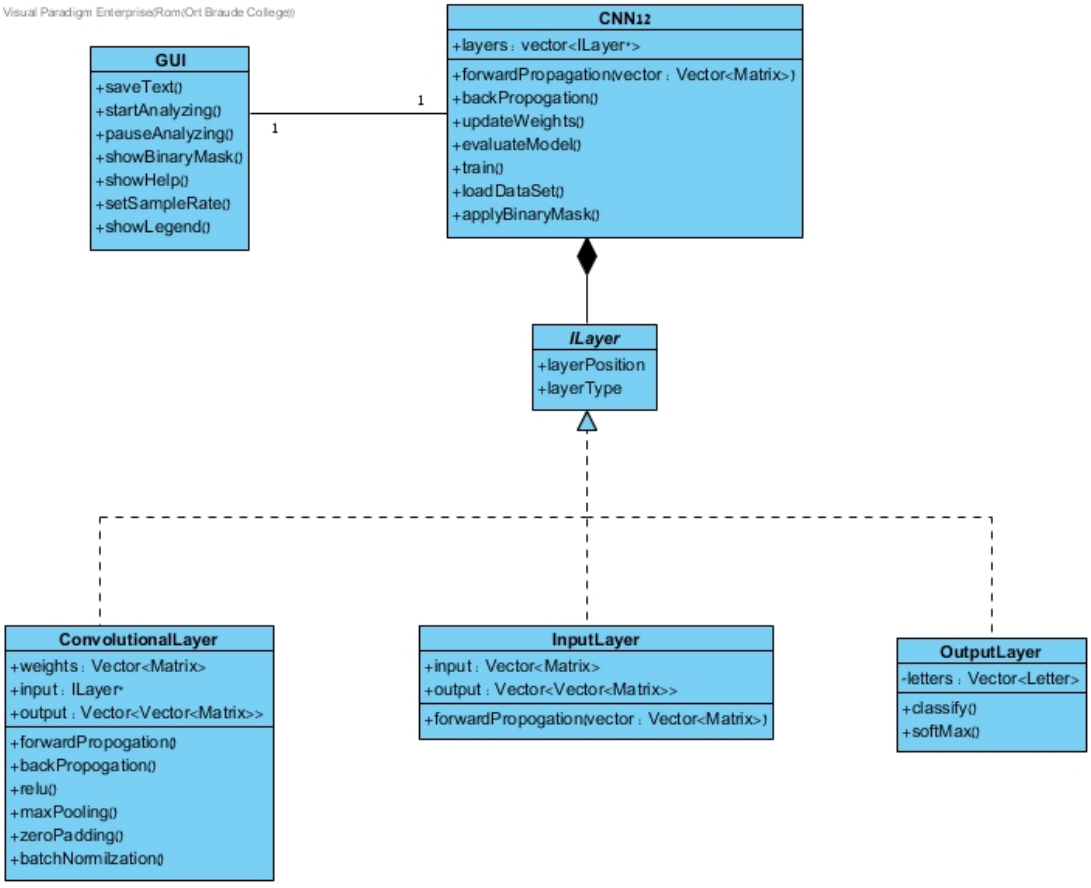
Use Case	Model Training
Actors and Goals	Trainer: Train the Model.
Pre-conditions	Dataset is ready.
Post-conditions	Model is trained.
Trigger	The Trainer runs the training command.
Main success scenario	The Training runs the training command. The system is training the Model. The Model is trained.
Branching	-
Further Requirements	Training data must be provided.

5.2. UML Diagrams

5.2.1. Use Case Diagram



5.2.2. Class Diagram



5.3. Testing Plan

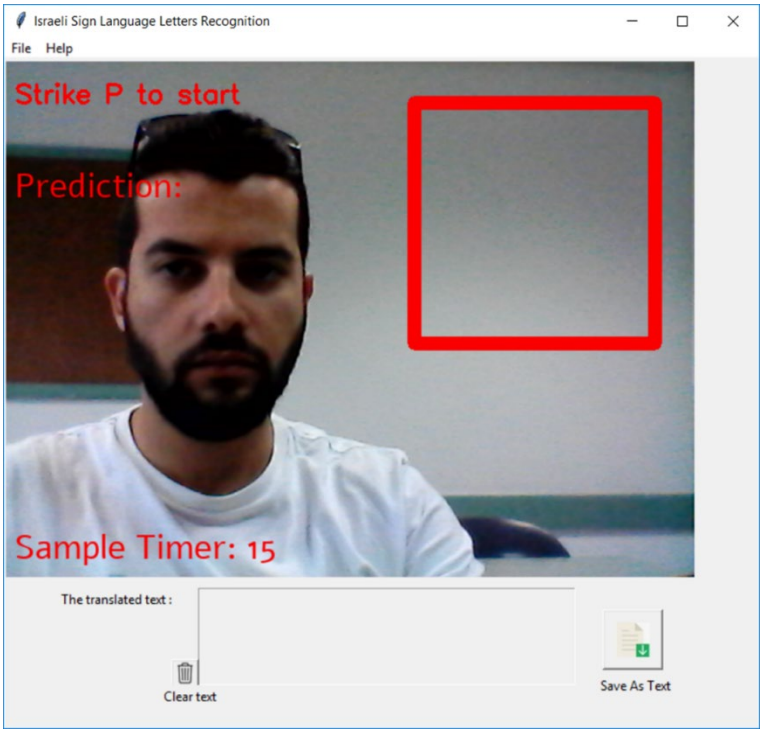
In order to check out the system performance and accuracy we ran the program on some significant inputs and tests.

Test No.	Test subject	Expected result	Pass/Fail
1	Save the processed output as text file.	Text file will be created with the recognized text.	Pass
2	Analyze hands signs of different people.	The system will display the right result, the correct translated letter.	Pass Accuracy of 89.66 % was reached while testing the system on a new Dataset.
3	Set correct capture rate (a value between 5-40 second)	The sample rate should be updated successfully, and the letters needs to display in a faster/lower rate according to the new value.	Pass
4	Set **wrong** capture rate	The System will display error message: “The rate is invalid”.	Pass

5.4 Graphic User Interface

When opening the application, the main window will appear.

Main screen:

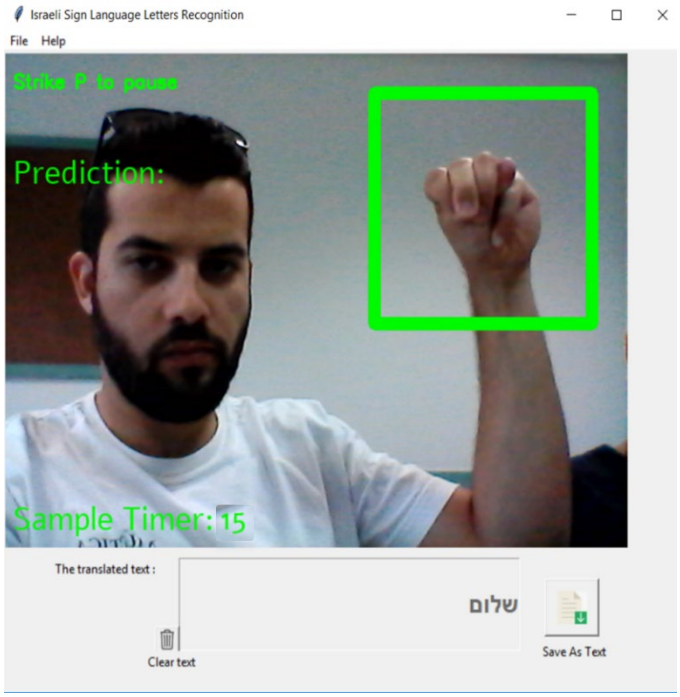


In the main window, all the options that the system offers are visible.

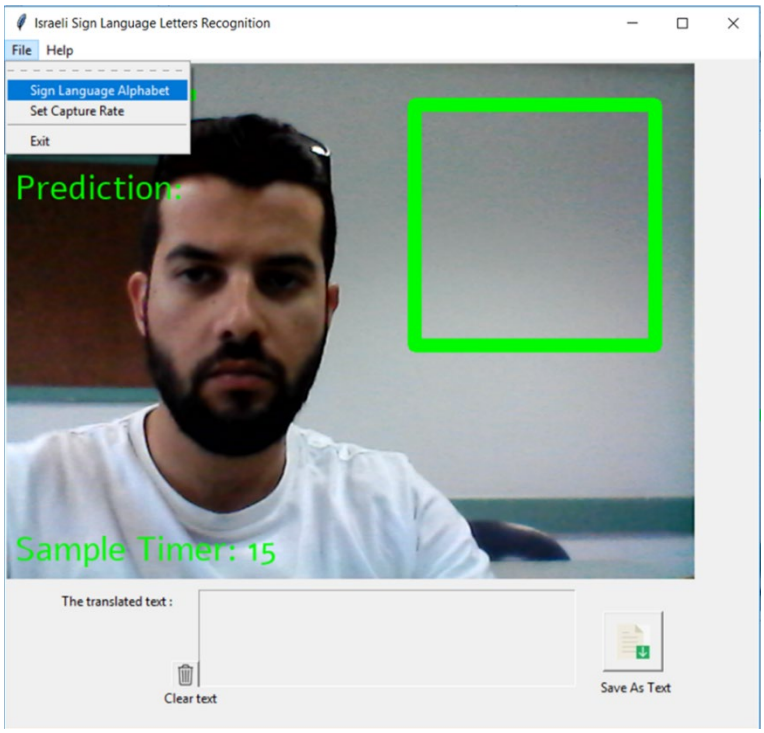
Components description:

- A red square (ROI - region of interest) represents the area where the frame is taken, due to this fact, the user needs to put his hand in the middle of the square while marking the wanted letter.
- The menu bar including the tabs File and Help.
- The “Sample Timer” label indicates the rate time(seconds) that each letter is written to the text box, in this short time the user could adjust his hand to mark the wanted sign.
- In the lower part of the window the translated text will appear, with the options of saving the translated text as a text file and clearing the text box by clicking the garbage icon.

In order to start the prediction procedure, the user must press the “P” button. When pressed, the ROI will turn into green and the user could start the recognition process.

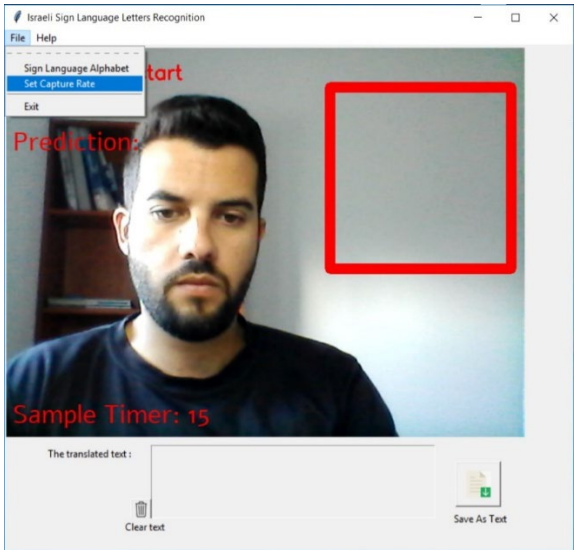


When “Save As Text” button is pressed, a text file is created with the translated text.
in case, the user pressed “Clear text” button, the text in the text box will be deleted.
If “P” is pressed again the predict process will pause.

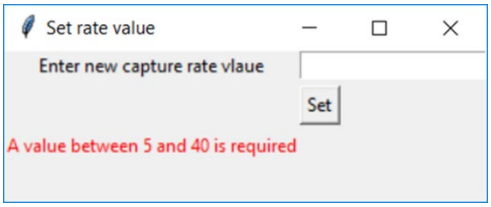


During the recognition process the user can open a legend of the Israeli sign language letters, by click File ->Sign Language Alphabet and the next window will appear. This Legend can assist the user usage in the application.

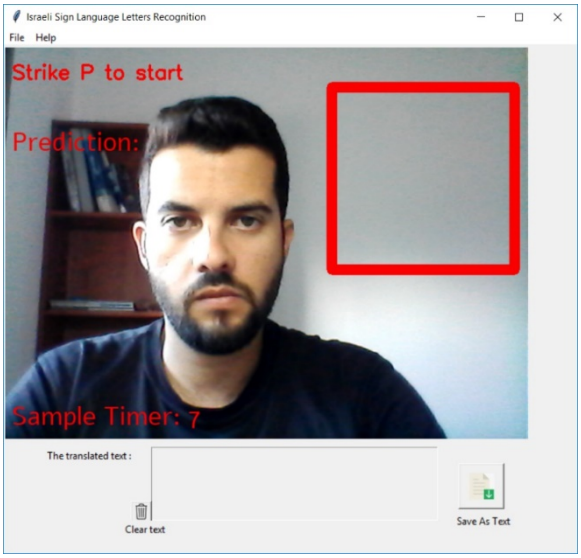
Capture rate can be determined according to user convenience.



After clicking on the Set Capture Rate in the File menu the next window will open.

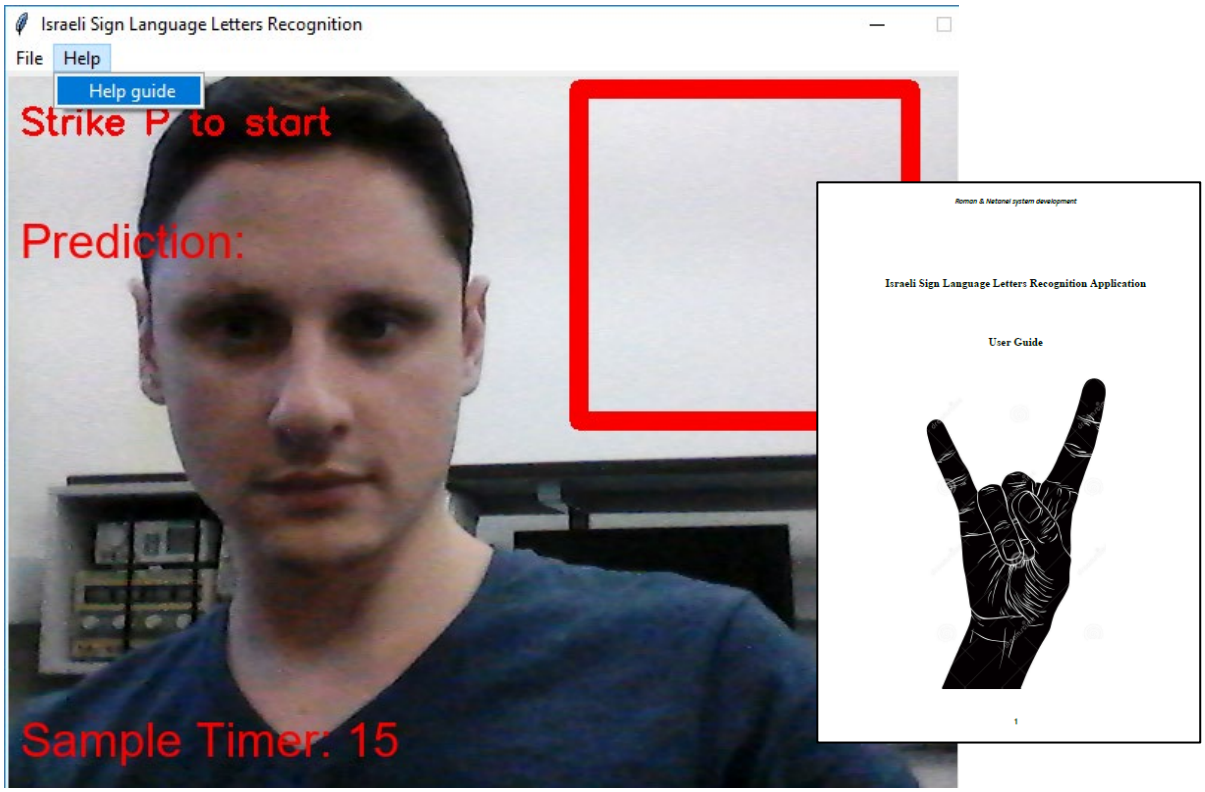


The user can set a value between 5-40 seconds. After choosing the wanted value it will be automatic update on the main screen as shown.



The change in the sample value will determine the speed that the letters will written in the text box.

For a new user in the application there is a “user guide” file that elaborates and demonstrates the system functionality. The file can be opened by pressing the Help tab -> help guide.
The guide is PDF format.



6. RESULTS AND CONCLUSION

6.1 Model Evaluation

The initial model evaluation was performed with the 20% Test Data that was explained under Training section. The accuracy was extremely high (99%). The conclusion was that the test data is very similar to the train data and another evaluation was needed.
In order to test the system accuracy in the most significant way, a relatively small Dataset was built which contains 140 image per letter (a total of 3500 images). The evaluation script that assess the accuracy of the recognition process was ran on the new dataset.
The accuracy rate, after evaluating the application, is **89.66%** *[Figure19]*. A confusion matrix [*figure18]* was produced, which shows the confusion rate for each class (Hand Sign).

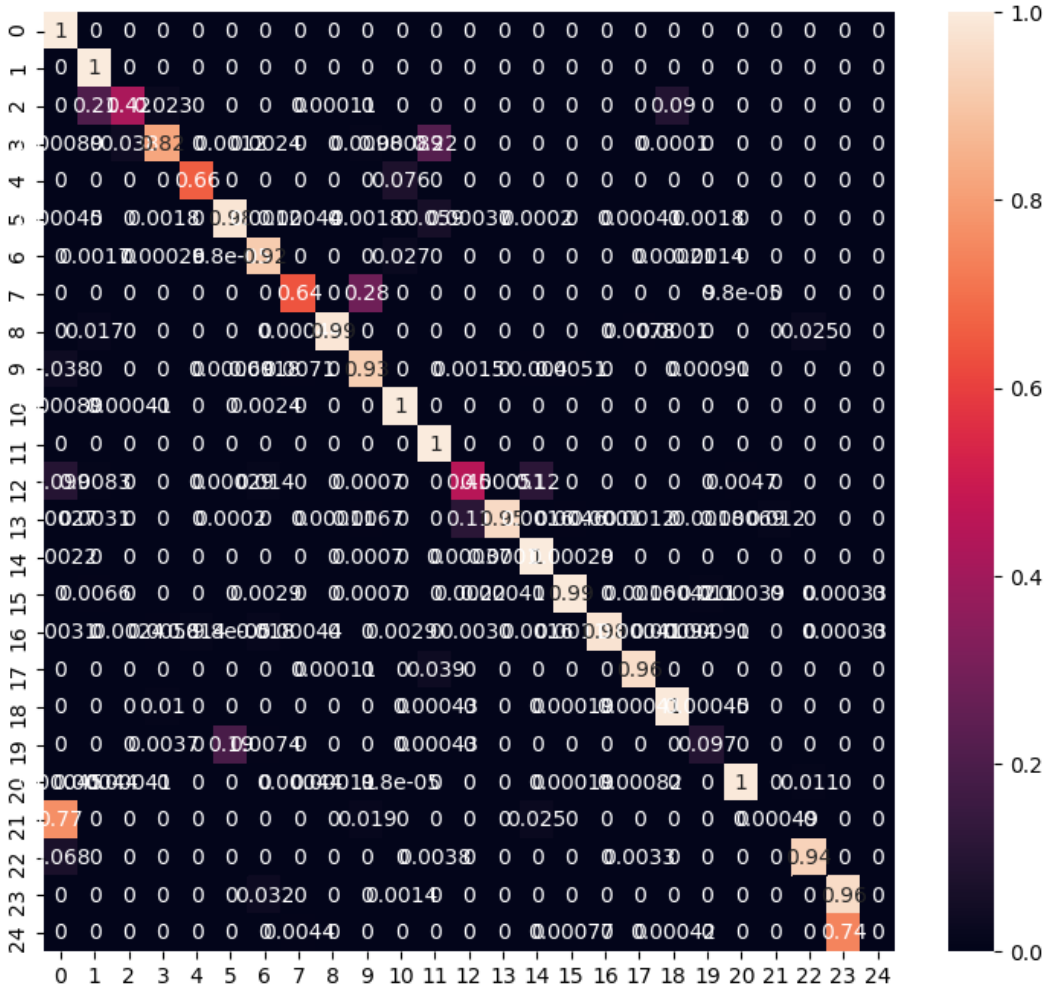


Figure 18 – Confusion matrix with 25 classes.

```
Loading test data..
Found 7248 images belonging to 25 classes.
Batch Progress: 100%
Predicting..
130256/130256 [==============================] - 1908s 15ms/step
Testing Accuracy = 89.66%
```

Figure 19 – Accuracy evaluation.

6.2. Real-Time performance and Conclusion

The application prediction threshold was determined to be above 80% meaning that if the model recognized a letter with accuracy above 0.8, it will be printed tot the text box. Otherwise the letter will be shown only in the main window but will not be printed.
Experience with the application shows that the recognition process has difficulties to identify similar signs (letters) such as: "ת“," נ" ,"מ" or "י" ,"ו". In the testing and the evaluation section, these similar signs decrease the accuracy of our model and the final recognition result of these letters was mostly wrong.
In order to increase the accuracy in these specific signs, a larger dataset of various people is needed along with a new model retraining and evaluation.
Furthermore another weakness of the recognition process in the need of clean white background in the ROI. This limits the environment that the software could operate.

Nonetheless, the software performs very well with high accuracy. The architecture that has been chosen is relatively light compared to other examined architectures allowing the average user to use the program smoothly on his personal computer. The software was used by external users, and they have found it user-friendly and useful.

The software can be used to teach children the Israeli sign language letters due to the easy interface. It can also be integrated in communication softwares such as Skype to involve the deaf community.

7. REFERENCES

[1] Meir, I., & Sandler, W. (2004). Language in space: A window on Israeli Sign Language.

[2] The Institute for the Advancement of Deaf Persons in Israel. http://www.dpii.org.

[3] CS231n: Convolutional Neural Networks for Visual Recognition. http://cs231n.stanford.edu/

[4] Goodfellow, Yoshua Bengio & Aaron Courville (2016). *Deep Learning.* MIT Press. http://www.deeplearningbook.org/

[5] Michael A. Nielsen. (2015), "Neural Networks and Deep Learning", Determination Press. http://neuralnetworksanddeeplearning.com/

[6] Agarap, A. F. (2018). Deep Learning using Rectified Linear Units (ReLU). *arXiv preprint arXiv:1803.08375*.

[7] Xu, B., Wang, N., Chen, T., & Li, M. (2015). Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*.

[8] Nagi, J., Ducatelle, F., Di Caro, G. A., Cireşan, D., Meier, U., Giusti, A., ... & Gambardella, L. M. (2011, November). Max-pooling convolutional neural networks for vision-based hand gesture recognition. In *Signal and Image Processing Applications (ICSIPA), 2011 IEEE International Conference on*(pp. 342-347). IEEE.

[9] Cireşan, D. C., Meier, U., Masci, J., Gambardella, L. M., & Schmidhuber, J. (2011). High-performance neural networks for visual object classification. *arXiv preprint arXiv:1102.0183*.

[10] Torrey, L., & Shavlik, J. (2010). Transfer learning. In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques* (pp. 242-264). IGI Global.

[11] Gershenson, C. (2003). Artificial neural networks for beginners. arXiv preprint cs/0308031.

[12] Image classification of Camera Trap Images using Convolutional Neural Network [https://github.com/hemavakade/CNN-for-Image-Classification]