Multimodal machine learning techniques for sign language recognition

Papadopoulos Georgios, Zidianakis Matthaios
Department of Digital Systems, University of Piraeus
Institute of Informatics and Telecommunication, NCSR “Demokritos”
MSc Artificial Intelligence

Abstract

Sign language recognition problem requires a plethora of computational and equipment resources for a required feature representation to express every hand gesture. Various studies have proposed different deep learning and machine learning approaches using depth images to achieve extremely high accuracy. In this work, we propose a solution based on the *Mediapipe* framework for extracting the body points from video frames, and on Neural Networks and the SVM classifier that accept them as input. The outcomes proved to be satisfactory given the chosen preprocessing methods that yielded 94-95% F1 macro score on the test set for 10 glosses as classes.

Ⅰ. Introduction

Automatic recognition of sign language is a challenging field which depends on each language specification and attributes, as well as on hand gesture detection complexity, capture equipment capabilities and efficiency of algorithms applied. Several studies have been made on this topic using different techniques in various settings and exploiting datasets with gestures which apply to sign language. Mainly, images with hand gestures are used to classify different kinds of distinct interpretations or specifically sign language words using machine learning or even deep learning methods.

A great issue of this kind of implementation is that it needs to be applied in real-time, achieving high performance in terms of accuracy, and identically to be able to run in low cost devices, thus it can be used widely. Another significant matter is that, mostly, the whole images are feeded to the corresponding algorithm, even though just some points of hands, face and maybe of shoulders are necessary. This happens because such implementations are engaged both with body parts detection and gesture recognition in the same technique, constructing techniques that cannot be used separately, like only detecting the body parts or just recognizing the different kinds of gestures given some body parts information.

In this work, we investigate such an approximation of the problem using the *MediaPipe* framework for extracting body parts information, and deep learning as well as traditional machine learning methods to classify several different sign language words. We ended up using the *GSL* dataset [1], in order to train and evaluate our methods.

Ⅱ. Related work

As is mentioned above, a lot of work has been made in the field of hand-gesture recognition, as also, particularly in automation of sign language detection. In [2], SVM classifier was applied using SIFT features, which were extracted from images, side by side with bag-of-features approach in order to solve the dimensionality curse. The accuracy achieved is in the range of 94.85% - 98.04%, but the target gestures are time-independent, which means that the classification task is less complex than the one which separates non-static sign language gestures.

In the latter cases, due to the fact that the recognition is based on videos, CNN models followed by LSTM or RNN layers are utilized to detect hands and extract spatial and temporal features, achieving high scores in terms of accuracy. In [3], a similar approach is used, but rather than just RGB images, Depth images are used as well. The authors mention that they applied different CNN layers for each image type, merging and leading them to some Fully Connected layers, and finally to RNN with LSTM cells for temporal feature extraction, resulting to 93% of validation accuracy when both RGB and Depth images were used and 82% with just RGB images. These Depth images increase the model performance, but the required equipment (Kinect camera) remains highly expensive in production level.
Despite this limitation, various works have made use of Depth images with state-of-the-art models like *ResNet-50* in [4], achieving 91.28% accuracy, being their best result for the *NVIDIA* dataset. In addition, using just Depth images, the authors of [5] performed an offline classification with 94,04% accuracy on the *EgoGesture* dataset and 83.82% on the *NVIDIA* one employing *ResNeXt-101* model.

Furthermore, as in [6], the most recent surveys are focused on the use of RGB videos with a moderate analysis as is in our daily life, to reduce the cost of equipment needed, as stated in the above paragraph. Specifically, the authors of the aforementioned paper worked with plain RGB video streams and a *3D-CNN* model, which aims on extracting both temporal and spatial features, reporting 90.1% test accuracy on *CSSL500* dataset. Here, a special video preprocessing technique is used, called *Optical Flow Calculation*, which gathers the most important aspects of the video frames. Therefore, instead of feeding the model with the whole frames of video, they feed it with just the most significant movement details extracted by this technique.

Moving one step forward, the release of *OpenPose* led to various applications using body keypoints, one of these being gesture recognition, as referred to [7]. In this work, the authors do not utilize deep neural networks methods to classify gesture type, but *DTW* along with *One-Nearest-Neighbor* (1NN). Another important feature of this work is that the authors performed *isolated* gesture recognition, which means that they did not use *continuous* mixed gestures in a single video. The resulting evaluation yielded an accuracy score of 77,4% for a selected subset of six different gestures from the *UTD-MHAD* dataset. In a similar way, the authors of [8] exploited the capabilities of *OpenPose* to extract anatomical key points from RGB images which then, in contrast with [7], were feeded into a *RNN* (Recurrent Neural Network) with *LSTM* cells in order to distinguish different types of human activities. In this work, they calculated the *magnitude* and the *angles* of the body keypoints between two consecutive frames, instead of using the mere keypoints of each frame. This approach led to an average accuracy of 92,4%, which is the highest between all the methods that were investigated (*SVM*, *Decision Trees*, *Random Forest*), but it should be noticed that 4 cameras were used in order to make the system view-independent. A major discrepancy of *HAR* (Human Activity Recognition) and *SLR* (Sign Language Recognition) is that in the latter, the observer of the task should be at the front side of the person who executes the gestures, thus *SLR* is a view-dependent task by nature.

Finally, one more survey should be referred to demonstrate an issue with keypoints representation methods. Using either *OpenPose* or *MediaPipe*, there are times that some keypoints are not available due to several factors like the point of view and the speed of movements. The authors of [9], where *OpenPose* was used, tried to solve this problem by estimating the trajectory of lost keypoints refining the equation of the trajectory to *Bézier* curves. The method of inferring the type of gesture was by just comparing the trajectory differences between some standard *Tai Chi* gestures and the real-time gesture. Even though the method of inference is very simple, it was able to distinguish a specific gesture group with a 92% accuracy score, but it was totally unable to do the same for a more complex gesture.

Ⅲ. Methodology

i. Dataset

For this work we used the *GSL* dataset (which can be found here: https://vcl.iti.gr/dataset/gsl/) that consists of a corpus with 310 different unique glosses in greek sign language and is free for use. Even though it contains both RGB and Depth images, we used just RGB for the cost specific reasons that we mentioned above. We took only the *isolated* part of the dataset which merely contains single greek sign words and we ignored the continuous sentences which are also provided. The samples were captured at 30 FPS and the size of each frame is 848 x 480. We should mention, a division to train, validation and test set was already performed from the creators of the dataset. It is important to note that these sets share different signers which means that the probability of the results being biased to specific signers is nullified.

We modified the whole dataset in 3 different ways resulting to 3 different datasets:

1. Keeping the first 10 unique glosses. This dataset is unbalanced and its distribution plot is shown below at *Figure 1*.
2. Keeping the first 12 unique glosses which have more than 500 samples. In particular, we got these glosses and we kept only 500 samples for each gloss in order to create a balanced dataset. Its distribution plot is presented at *FIgure 2*.
3. Keeping the first 10 unique glosses containing more than 500 samples for each gloss. The distribution is almost the same as *Figure 2*, excluding the last two classes.
4. Keeping all the glosses (310). This version is also unbalanced, but its distribution plot is not provided because of its size.

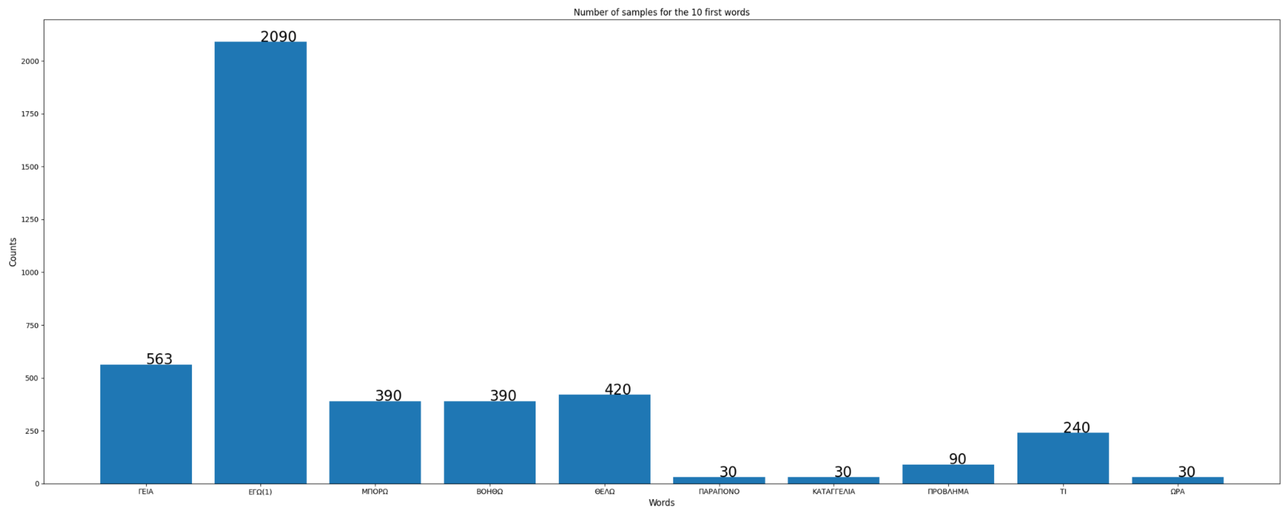


Figure 1: Word distribution plot for dataset 1 of train set.

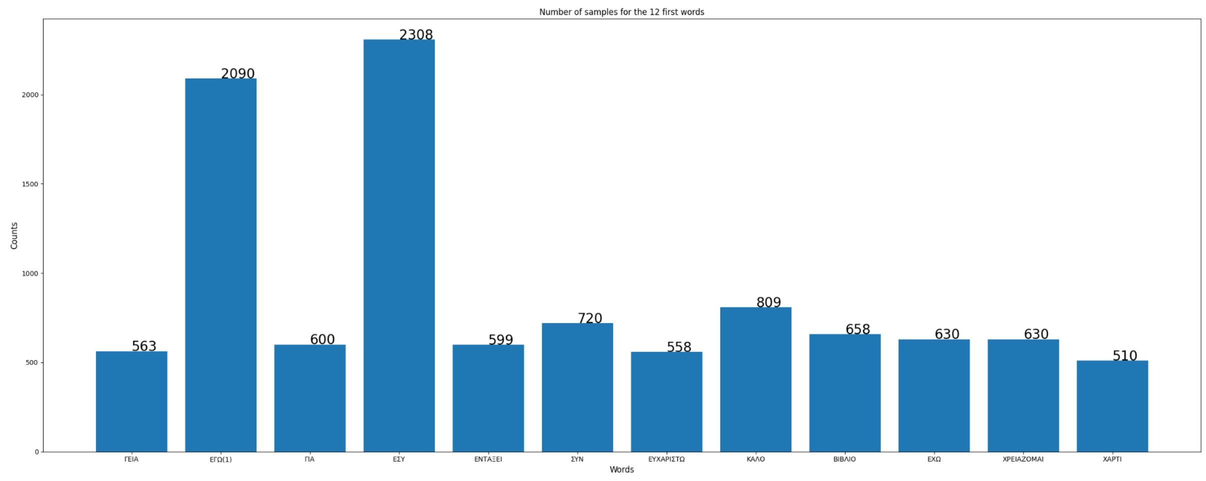


Figure 2: Word distribution plot for dataset 2 of train set before undersampling.

ii. Methods

a. Preprocessing

At first, considering the preprocessing stage we chose the *MediaPipe* framework to extract the normalized body points relative to the image frame. Those points represent hand and face features, because sign language generally is depicted by them. Our experiments involved a sampling of face points with a rate of five points per sample. Specifically, we tried 125 and 468 total face points and the sampling reduced them to 25 and 94 respectively. We also experimented with basic face points and upper body pose points provided by the *pose* function of *MediaPipe*.

Figure 3 and Figure 4 show the outcome of the corresponding process:

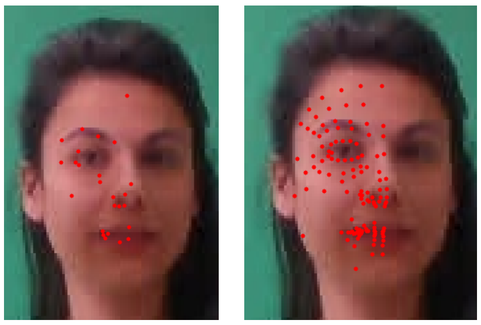


Figure 3: At left face points after sampling. At right original face points found.

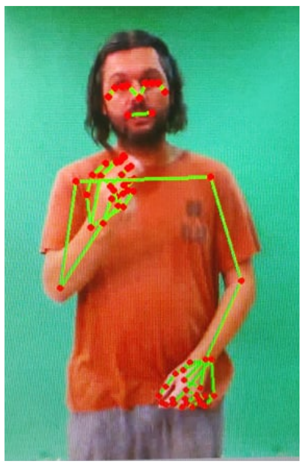


Figure 4: Face and body points of pose function.

Another crucial issue, as was stated in the previous section, is that sometimes there are lost points because of the limited efficiency of the algorithms which are used by *Mediapipe.* The inaccurate or deficient recognition of body points seems to occur when fast movements appear. To eliminate this problem, we applied *blur* filtering provided by *opencv* library after resizing the image by 400% and applying *interpolation*. This technique was tested on a few images and there were some examples that yielded correct results while in the initial corresponding images without this technique *MediaPipe* failed to find any points, like *Figure 5*.

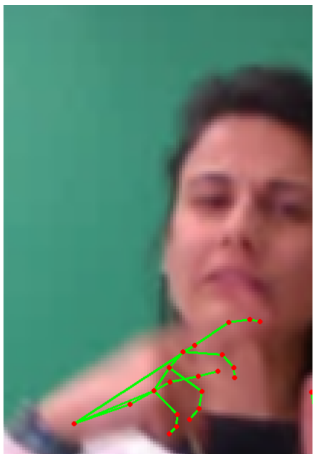


Figure 5: Outcome of the applied filter

Contrary to the previous approach, we now placed the desired keypoints and their connections on blank images in order to exclude noisy pixel information. We also resized the images to 128 x 128, as in *Figure 6*.

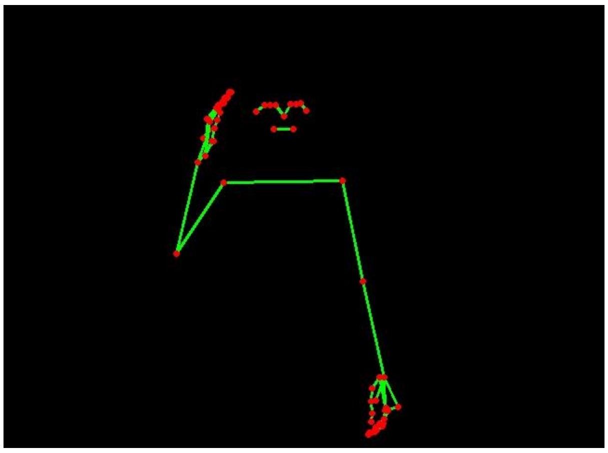


Figure 6: Pose skeleton in blank image

Concluding with preprocessing, based on paper [8], we calculated the magnitudes M_t^N and angles A_t^N between two consecutive frames, where *N* is a keypoint and *t* is a time-point indicating the corresponding frame, using the following formulas:

$$M_t^N = \sqrt{\left(D_{N_{x(t+1)}} - D_{N_{x(t)}}\right)^2 + \left(D_{N_{y(t+1)}} - D_{N_{y(t)}}\right)^2}$$

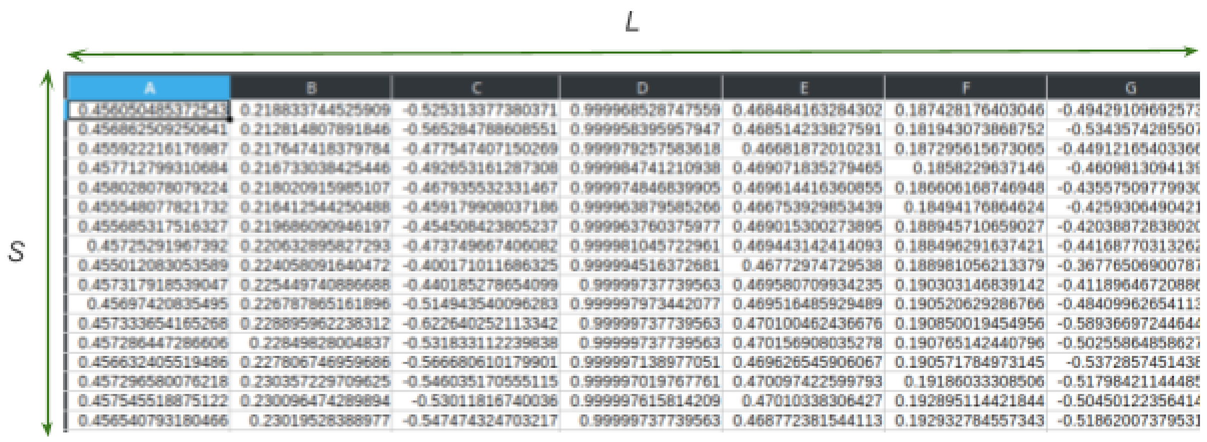
$$A_t^N = tanh^{-1}\left(\frac{D_{N_{y(t+1)}} - D_{N_{y(t)}}}{D_{N_{x(t+1)}} - D_{N_{x(t)}}}\right)$$

where D_{N_x} and D_{N_y} denote the distances of the *N* keypoint coordinates between two successive frames in *x* and *y* axis, respectively. These features were used instead of the pure keypoints coordinates to feed the models which are described in the next section.

b. Main methods

Considering the methods applied to classify the different sign language gestures, we examined several neural network architectures and we are going to present the ones that achieved the best results in terms of both *macro F1-score* and loss performance. Specifically, we refer to a CNN based architecture consisting of a combination of convolution layers, a single RNN layer with GRU cells followed by three fully connected layers, including the output layer, as well as an architecture with just a RNN-LSTM layer followed by the same sequential fully connected layers. Furthermore, we also tried an SVM classifier as a lot of glosses do not contain enough samples for a deep learning method.

Regarding the first type of architecture, we should define *Architecture 1* and *Architecture 2* which are differentiated at the type of CNN. At *FIgure 7*, *Architecture 1* is illustrated which begins with one 1D Convolution layer accepting the numeric values of the normalized keypoints as input with size $S \times L$. *L* denotes the number of the elements of the vector containing the flattened key points which is described as $\vec{L} = (p_x^1, p_y^1, p_x^2, p_y^2, ..., p_x^k, p_y^k)$ where p_x^i, p_y^i are the normalized coordinates of *i-th* point. *S* is the number of sequence elements in the form of frame number which are used for each gesture video sample. In this point, we should mention that the default number of frames was set to 10, because the model has to be fed with a static sequence length.



A	B	C	D	E	F	G
0.456050485372543	0.218833744525909	-0.525313377380371	0.999968528747559	0.468484163284302	0.187428176403046	-0.49429109692573
0.456862509250641	0.212814807891846	-0.565284788608551	0.999958395957947	0.468514233827591	0.181943073868752	-0.5343574285507
0.455922216176987	0.217647418379784	-0.477547407150269	0.999979257583618	0.46681872010231	0.187295615673065	-0.4491216540336
0.457712799310684	0.216733038425446	-0.492653161287308	0.999984741210938	0.469071835279465	0.1858229637146	-0.4609813094139
0.458028078079224	0.218020915985107	-0.467935532331467	0.999974846839905	0.469614416360855	0.186606168746948	-0.4355750977993
0.455548077821732	0.216412544250488	-0.459179908037186	0.999963879585266	0.466753929853439	0.18494176864624	-0.425930649042
0.455685317516327	0.219686090946197	-0.454508423805237	0.999963760375977	0.469015300273895	0.188945710659027	-0.4203887283802
0.45725291967392	0.220632895827293	-0.473749667406082	0.999981045722961	0.469443142414093	0.188496291637421	-0.4416877031326
0.455012083053589	0.224058091640472	-0.400171011686325	0.999994516372681	0.46772974729538	0.188981056213379	-0.3677650690078
0.457317918539047	0.225449740886688	-0.440185278654099	0.99999737739563	0.469580709934235	0.190303146839142	-0.4118964672088
0.45697420835495	0.226787865161896	-0.514943540096283	0.999997973442077	0.469516485929489	0.190520629286766	-0.4840996265411
0.457333654165268	0.228895962238312	-0.622640252113342	0.99999737739563	0.470100462436676	0.190850019454956	-0.5893669724464
0.457286447286606	0.22849828004837	-0.531833112239838	0.99999737739563	0.470156908035278	0.190765142440796	-0.5025586485862
0.456632405519486	0.227806746959686	-0.566680610179901	0.999997138977051	0.469626545906067	0.190571784973145	-0.537285745143
0.457296580076218	0.230357229709625	-0.546035170555115	0.999997019767761	0.470097422599793	0.19186033308506	-0.5179842114448
0.457545518875122	0.230096474289894	-0.53011816740036	0.999997615814209	0.47010338306427	0.192895114421844	-0.5045012235641
0.456540793180466	0.23019528388977	-0.547474324703217	0.99999737739563	0.468772381544113	0.192932784557343	-0.5186200737953

Figure 7: Illustration of $S \times L$ input with $\vec{L} = (p_x^1, p_y^1, p_x^2, p_y^2, ..., p_x^k, p_y^k)$

As some video samples contain more than 10 frames, we removed the surplus, and for those with less than 10 frames we performed zero-padding at the *S* dimension, meaning that we added zeros at the coordinates of all keypoints for the missing frames. After the 1D Convolution layer, a 1D Max pooling layer is placed reducing the size of *L*. In this way, the CNN layer is responsible for a high-level spatial feature extraction leading to an alternative form of the input representation without changing the temporal information. Afterwards, the output of the CNN layer is introduced in the GRU layer which specializes in the temporal aspect of the data. The output of the last hidden cell of GRU, which is equal to *m* as the number of hidden GRU cells, is considered to be flattened and fed as input of the First Fully Connected layer followed by the last Output layer. In addition, we placed a Dropout layer between CNN and GRU layer as well as among Fully Connected layers to apply a regularization factor in order to avoid the overfitting which we observed beforehand. The logits output of the network are used to calculate the loss based on the *cross entropy* method. Furthemore, the optimizer applied for network weights update of the backpropagation process was set to be *Adam*.

As far as *Architecture 2* is concerned, we chose a similar outline to the previous architecture with some alterations in CNN dimensions and input data type, introducing 2D Convolution and Max pooling layers and feeding the model with pose skeleton images like in *Figure 6.* Here the input of CNN is also of shape $S \times L$, but $\vec{L}$ has a shape of $C \times H \times W$, where *C* denotes the image channels and *H, W* denote the height and width of the image respectively. Finally, the version of zero padding in this architecture refers to adding blank images until 10 frames are filled.

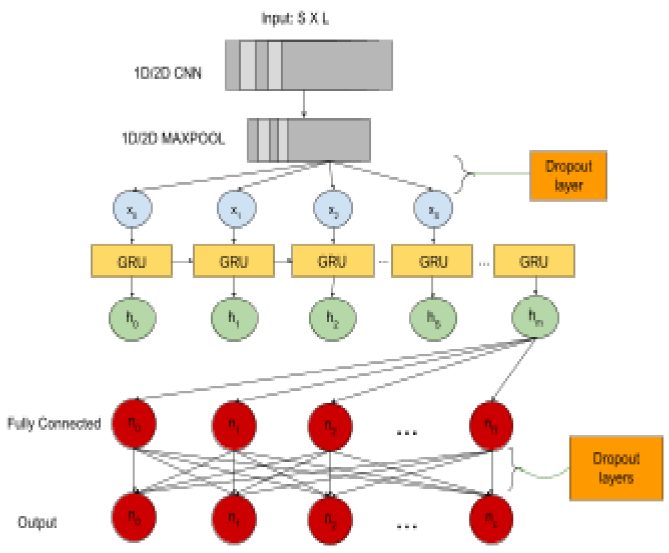


Figure 8: Model of Architectures 1 and 2 (1D for Architecture 1 and 2D for Architecture 2)

In the last experiment (*Architecture 3),* we selected only one layer of LSTM followed by a single fully connected layer. A Dropout layer was also selected for regularization to reduce overfitting. The input is the same as in *Architecture 1*, but no convolution layers were used.

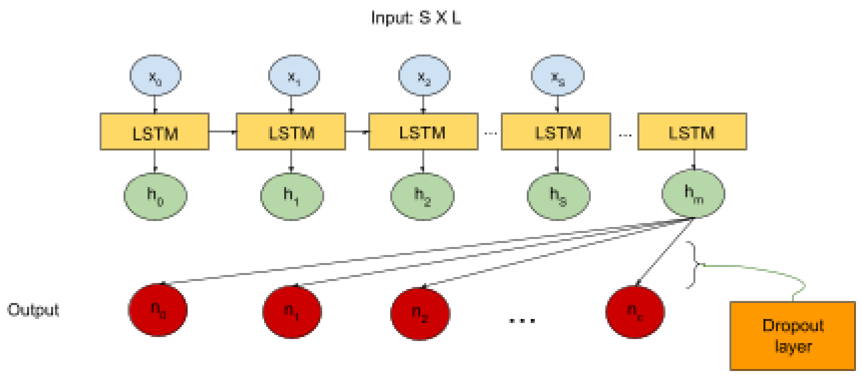


Figure 9: Model of Architecture 3

Apart from the Deep Learning methods, we also experimented with a classic Machine Learning algorithm, specifically Support Vector Machines, because ML methods tend to perform better than Neural Networks for fewer samples. In this case, sample shortage is considered to be when a specific class has fewer samples than 30, which takes place most frequently in *dataset 3* (310 unique classes).

Ⅳ. Results

In this section, the results of each learning method are presented taking into account the specific configuration of the dataset and preprocessing technique. In this point, we should mention that all the models were implemented exploiting *Pytorch* framework capabilities, except from SVM which was implemented using *sklearn* python library. At the experiments that no learning rate and optimizer are reported, it should be considered that these are 0.001 and *Adam*, respectively. Beginning with *Architecture 1*, the outcomes of *dataset 1* are displayed below:

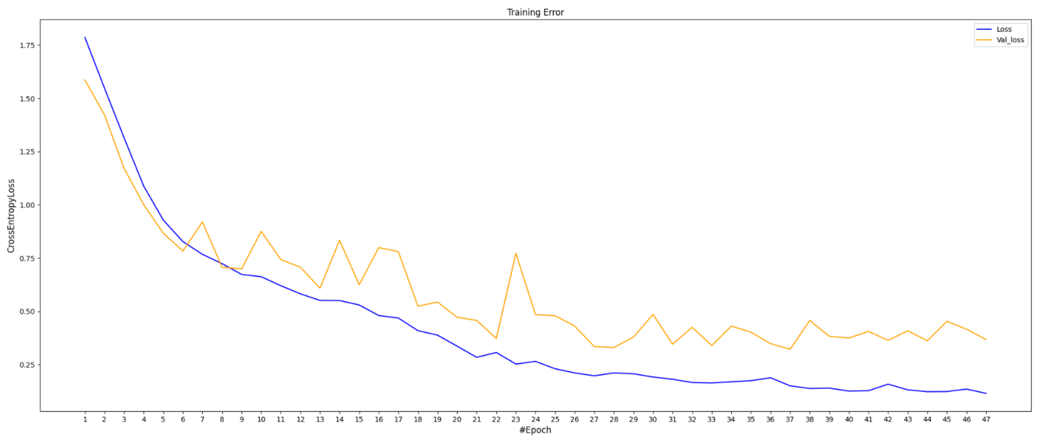


Figure 10: Validation and training losses for Architecture 1 and dataset 1.

As we can observe both validation and training losses are decreasing simultaneously until epoch 37, where overfitting seems to appear. Concerning confusion matrix (*Figure 11*), there are a very few misclassification examples for each class except for two of them ('ΠΑΡΑΠΟΝΟ', 'ΚΑΤΑΓΓΕΛΙΑ') that completely misclassified them. Examining the reason for the incident, we figured out that this happens due to the lack of training samples, 30 for each of them, which is apparent from *Figure 1*. In order to avoid pure class separation, we introduced class weighting in the *cross entropy loss* function, encouraging it to emphasize on classes with the least examples. The weight of each class was calculated according to the following equations:

$$weight_i = 1/\left(\left|samples_i\right| / \sum_{i=1}^{n} \left|samples_i\right| \right)$$

$$normalized\ weight_i = weight_i / \sum_{i=1}^{n} weight_i$$

As a result of this method was the model to obtain the ability to separate the classes with the least samples as it could not separate entirely the ones with the highest samples number (e.g 'ΕΓΩ(1)' class).

From *Table 1* results, we can conclude that the *weighted cross entropy* loss reduces the overall F1-macro score and also that both with and without weights the test set achieves higher score than validation, which is an indication of adequate model generalization. Nevertheless, this model setting (*Architecture 1*) along with the specific dataset (*dataset 1*) performs purely. We should state that the parameters used are: 128 batch size, 64 filters for CNN layers, 5 kernel size, 1 stride, 0 padding and 100 GRU cells, 128 neurons of Fully connected layer and 0.2 percentage for both dropout layers.

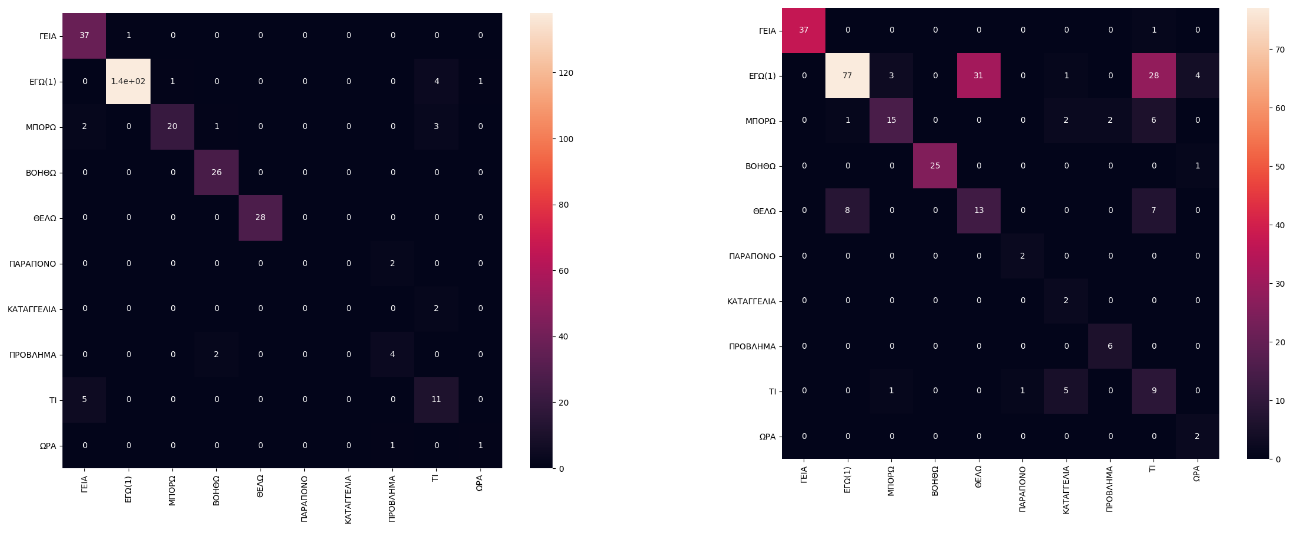


Figure 11: At right confusion matrix without weights. At left confusion matrix with weights in cross entropy loss function. Both for validation set, Architecture 1 and dataset 1.

Architecture 1 - Dataset 1		
	Validation set	Test set
With class weights in loss function	53%	59%
Without class weights in loss function	61%	67%

Table1: F1-macro score results for Architecture 1 - Dataset 1

We performed further experimentation exploiting *Architecture 3* which led to results with lower variance between validation and test set, 64% and 65% respectively in terms of F1-macro score (without class weights in loss function). Based on this last fact in conjunction with the Figure *12* losses plot, where losses appear to be closer than in *Figure 10*, we could infer that *Architecture 3* has the ability to generalize further probably due to the lack of samples which CNN layer needs to be trained. The architecture parameters used for these reported results are 128 batch size, 100 LSTM and 0.2 dropout percentage.

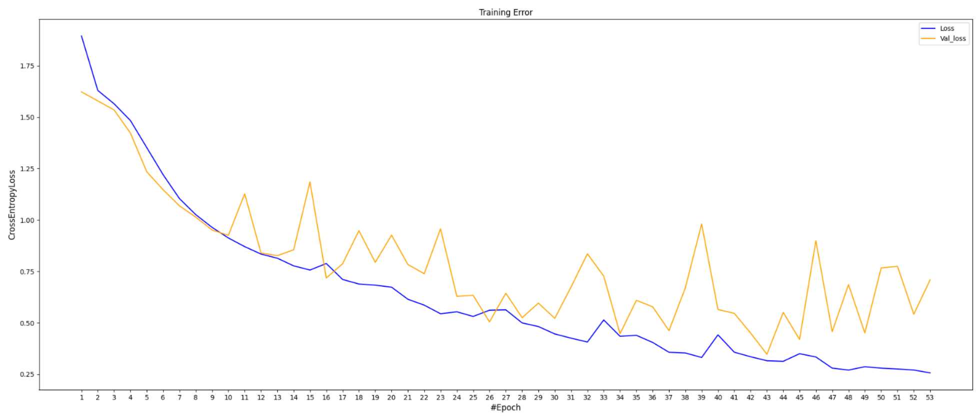


Figure 12: Validation and training losses for Architecture 3 and dataset 1.

Regarding *dataset 2*, which is balanced (in training set) with 500 samples in each class and sampling every 5 face points as is shown in *Figure 3*, *Architecture 1* succeeds to distinguish the different gestures better than the previous dataset, and specifically with an almost 94% F1-macro score (in test set, see *Table 2*), which happens to be the largest achieved score among all the experiments performed. We ended up with this kind of face points sampling as it led to a higher score than the corresponding experiments with all 124 face points. This observation led us to assume that too many face points inject unnecessary noise to the dataset which makes the model efficiency unstable. Finally, it should be noticed that the parameters used in this case are identical with those mentioned below *Figure 10.*

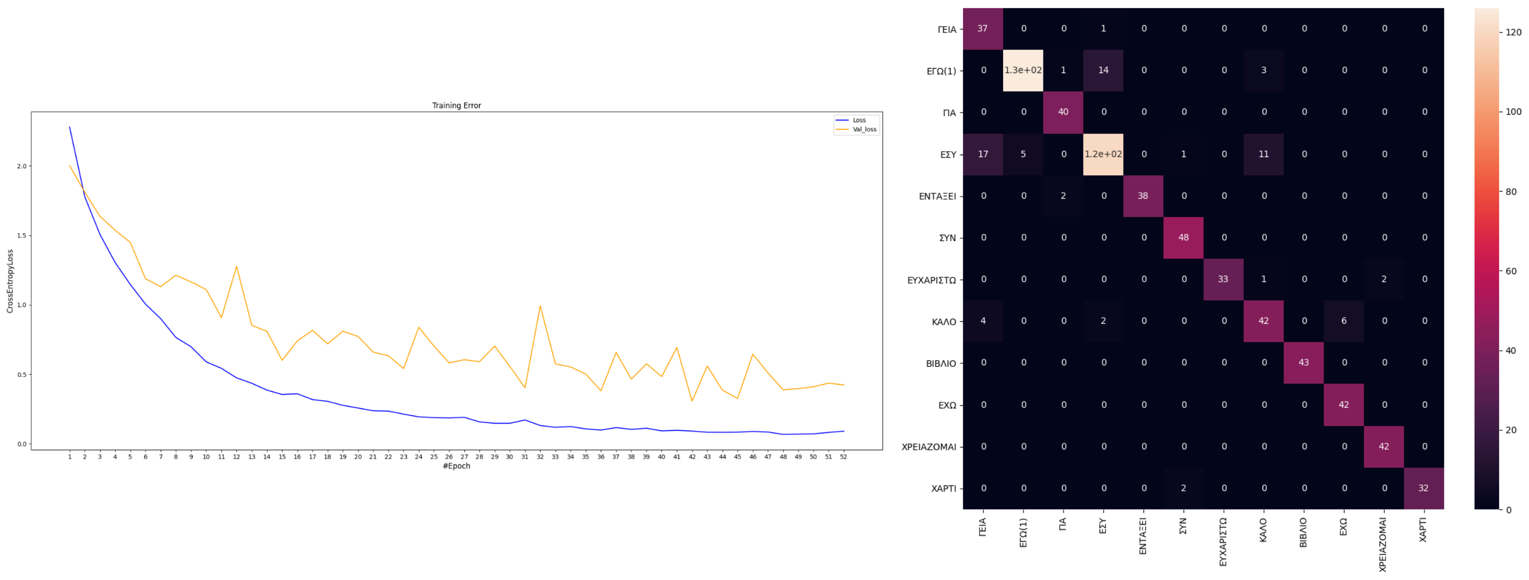


Figure 13: At left validation and training losses. At right confusion matrix. Both for Architecture 1 and dataset 2.

Architecture 1 - dataset 2	
Validation set	Test set
91.77%	93.89%

Table 2: F1-macro score results for Architecture 1 - dataset 2 with sampling every 5 face points.

Additionally, it is worth mentioning that we performed experiments with *dataset 2* using *Architecture 3* resulting in lower scores (91.1% and 89.9% F1-macro for validation and test set, respectively). This fact could mean that the CNN layer before RNN provides a positive impact when using at least 500 samples in each class comparing the corresponding results of *dataset 1* which are mentioned above (see *Table 1* and the below paragraph). The architecture parameters utilized for these outcomes are 128 batch size, 100 LSTM and 0.2 dropout percentage.

A special attribute of *dataset 3* is that two specific words of classes 2 and 6 respectively (*Figure 14 and 15 below)* are very similar in terms of the shape and the sequence of the gesture. As a result, when undersampled *dataset 3* was tested with *Architecture 3* yielded a validation F1 macro score of 88% but as indicated in *Figure 14* the model confuses these two classes. Specifically, the recall of class 2 has the value of 41% and the precision of class 6 has the value of 57%. A possible explanation for this phenomenon would be the lack of samples due to undersampling, because when we performed experiments with the full dataset we did not notice this extreme misclassification of the aforementioned classes. For these experiments we set batch size equal to 64, LSTM cells number equal to 128 and a percentage of dropout equal to 0.2.

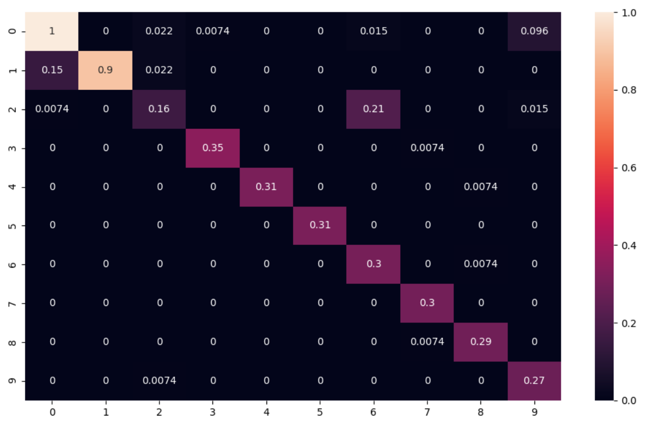


Figure 14: Confusion matrix for Architecture 3, dataset 3 (balanced version by undersampling to 500 samples per class), with sampling every 5 face points.

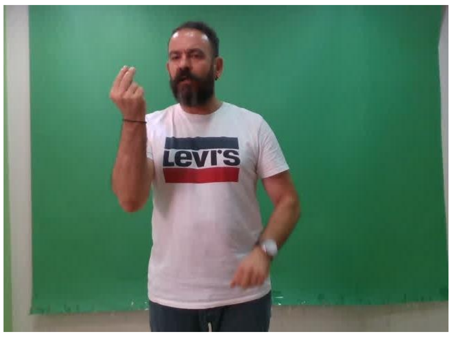
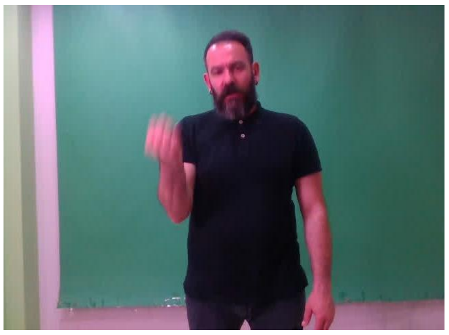
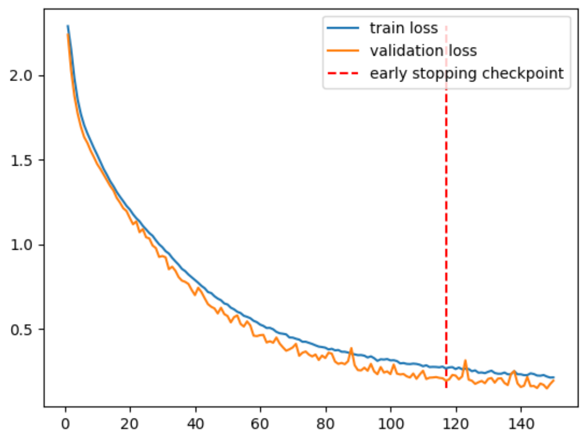



Figure 15: At left a representative frame of gloss 'ΚΑΛΟ' (class 2 of Figure 14). At right a representative frame of gloss 'ΕΧΩ' (class 6 of Figure 14).

Lastly, we combined *Architecture 3* with the unbalanced *dataset 3* keeping only the upper body keypoints of the pose (face, shoulders, elbows and wrist) and the hand keypoints (see *Figure 4*). Class weights were also involved in training and testing. This configuration yielded a non-overfitting model achieving an F1 macro score of 94% in validation mode and 95% in test mode. The training session of the model included 150 epochs that early-stopped in the epoch 117 and 32 hidden LSTM cells. We made a final tuning with learning rate and weight decay equal to 0.0001 and with adding a 20% dropout layer to the output of the LSTM layer.

The total performance of the model is shown below:



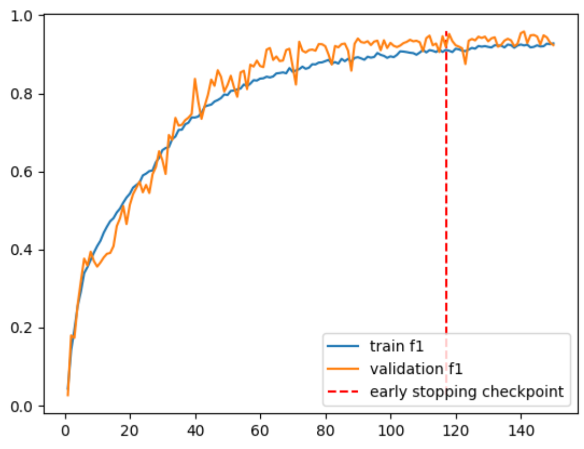


Figure 16: Validation loss (left) and F1 macro score (right) plots

Architecture 3 - Dataset 3	
Validation	Test
94%	95%

Table 3: F1 macro score for dataset 3 with Architecture 3

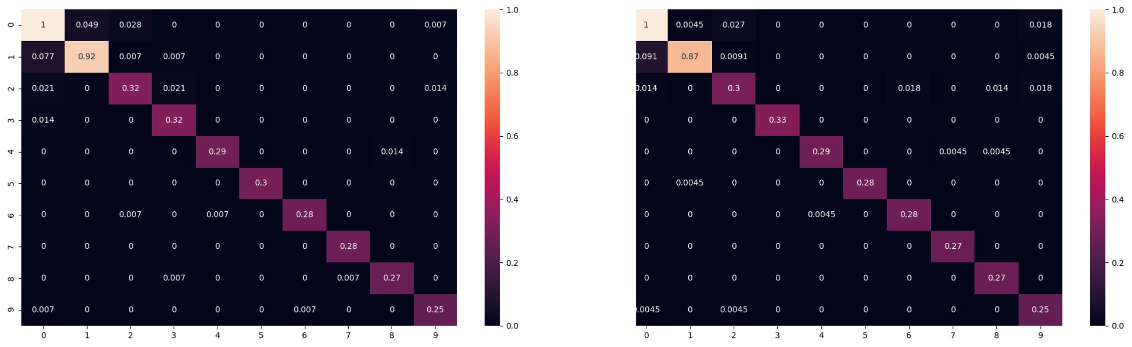


Figure 17: Left, confusion matrix of the validation set. Right, confusion matrix of the test set

Moving forward to *dataset 4*, where all glosses (310) are included, the performance of *Architecture 1* was reduced compared with the aforementioned results, employing the same parameters. This behaviour is reasonable as the number of individual glosses is almost 30 times higher than before leading to 56.81% and 60.5% F1-macro scores of validation and test sets, respectively. To improve this performance by manipulating overfitting (see *Figure 16*) we applied a weight *decay* regularization method with a factor equal to 0.0001 and also a lower learning rate (from 0.001 to 0.0001). Nonetheless, the outcomes did not improve leading us to conclude that presumably the samples of each class are not enough compared to the particular number of glosses, as the most of them had less than 50 samples.

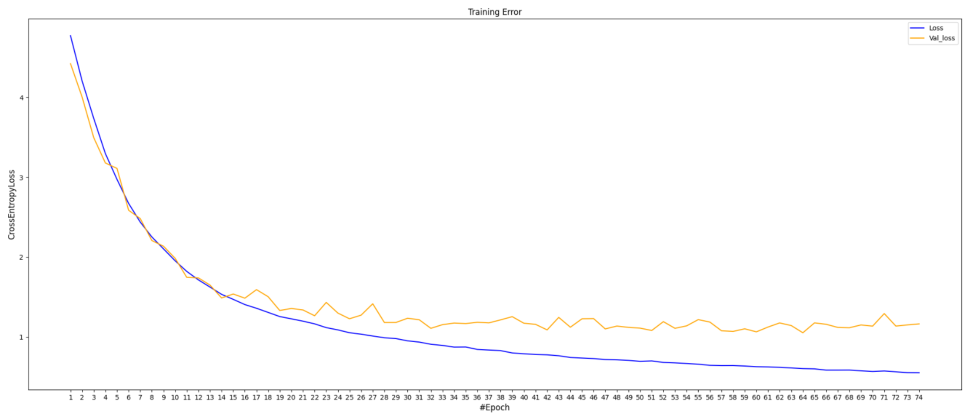


Figure 18: Validation and training losses for Architecture 1 and dataset 4.

As it was mentioned in section (Ⅲ. ii. b.), we tried out the efficiency of the SVM classifier in this kind of problem. By observing *Table 4*, it is obvious that SVM cannot outweigh Deep Learning methods except in the case of *dataset 1*, but with great variance between validation and test sets scores. These high outcomes (for *dataset 1*) might occur because there are not only a few samples but also few classes. The efficiency of SVM in this case falls in the range of capabilities of such a classical Machine learning approach. The best parameters were found after applying grid searches for each of the dataset presented at *Table 4*, and these are ("C": 0.1, "kernel": "linear"), ("C": 10, "gamma": 0.001, "kernel": "rbf") and ("C": 1000, "kernel": "linear") for *dataset 1, 2 and 4* respectively. Moreover, we should note that we used class weights multiplied with parameter *C* in order to achieve a higher score without receiving better results.

SVM scores		
	Validation set	Test set
Dataset 1	78%	88%
Dataset 2	81%	82%
Dataset 4	39%	42%

Table 4: SVM F1-macro scores for datasets 1, 2, 4.

In this point, we should provide the results which arise by utilizing *Architecture 2* along with skeleton images (see *Figure 6*) of *dataset 2*. Unfortunately, these outcomes were quite pure, 42.4% and 43.1% F1-macro scores in validation and test set respectively, indicating that the use of whole black images introduce an additional noise to the model which does not appear when the raw information of normalized points is employed. It is significant to be noticed that in this case, the number of CNN filters were just 8 and the GRU cells only 20, in contrast with all the previous experiments. This is due to the high complexity and parameters number resulting from the use of a 2D convolution layer followed by a GRU layer and the lack of computational resources. The aforementioned parameters applied should be taken into consideration as an extra possible reason for the model's low performance. Here again, we tried unsuccessfully to improve the performance by reducing the learning rate and adding *weight decay* regularization.

Thereafter, the outcomes of using the magnitudes and the angles between two consecutive frames, as described in section (Ⅲ. ii. a.), should be reported even if they are not better than the rest which have been displayed till now. Specifically, regarding the experiments where *Architecture 1* was applied in *dataset 2*, F1-macro scores are 70.29% and 65.82% for validation and test set, which are far enough than the corresponding case of using the points themselves, shown in *Table 2.* The parameters used are the same as those described for the experiments which led to the results of *Table2.* In addition, the SVM classifier was tested for this setting (magnitudes and angles of *dataset 2* body points) resulting in 45% and 49% for validation and test set accordingly harnessing as parameters: ("C": 10, "gamma": 0.001, "kernel": "rbf"). These outcomes are also much lower than those of *Table 4* (second row of table) which is a fact that confirms the negative impact of using the magnitudes and angles of body points. Nevertheless, the original concept has to be further examined, which was that when these features are considered, the model trained to recognize the different gestures could learn some relative consecutive movements of body points instead of the sequence of their absolute position on each image. In this way, a more general knowledge could be obtained from the model which could be applied on videos captured with different angles between the camera and the person who performs the sign language gestures.

At the end, regarding the impact of applying the filters (*blur*, rescale and interpolation) to smooth quick body movements, they did not have the expected boosting effect. Specifically, after performing the corresponding experiments (with the same parameters as at the experiments of the reported results of *Table 2* and also by sampling every 5 face points) the achieved F1 macro score for validation set of *dataset 2* is 88.25% and 91.86% for the test set accordingly. By comparing them with those of *Table 2*, we can infer that these filters reduce the model efficiency to some extent. This is in contrast with the observed positive affect to *Mediapipe* body points recognition illustrated in *Figure 5* (chapter Ⅲ. ii. a.). After examining more video frames processed by this method, we concluded that there are many cases in which the identified body points (especially hand points) were totally wrong and inaccurate like the example exhibited in *Figure 19* below. Therefore, further examination is necessary about the appropriate image preprocessing in order to remove the existing noise in the frames caused by the high speed of hands motion.

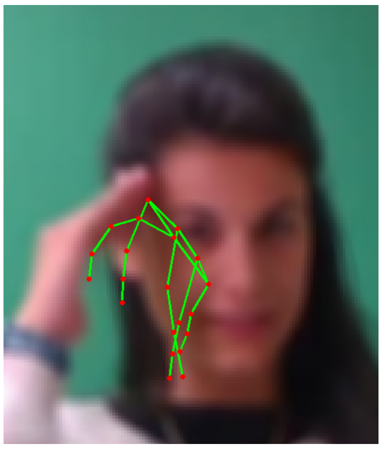


Figure 19: Inaccurate recognition of right hand points after applied image preprocessing with blur filter.

Ⅴ. Conclusions

In this work the efficiency of deep and classical machine learning methods was examined to classify different sign gestures of greek language. Using several preprocessing techniques applied on the video frames we achieved to build models that have the ability to distinguish among this kind of gestures with the highest score of 94-95% in the F1 macro scale. Specifically, exploiting *Architecture 3* and *dataset 3* as well as with *Architecture 1* and *dataset 2*. Even though the outcomes are great, this is true only with a low number of individual glosses because increasing this number to 310 the results were much more reduced. In this case, after sufficient experimentation we ended up finding that the ratio between the number classes and the number of dataset examples is a significant factor for the successive solution of the problem with respect to its complexity. On the other hand SVM classifier performs almost as high as deep learning models for a low number of classes, but it totally fails when this number as the number of samples are increased which is a fact validated in theory.

We figured out that the number of face points used has an important part as after a specific threshold of this number the model strains to classify the examples. We managed to locate the issue by sampling down the total face points and this aims to improve the performance of the model.

As a future work, further preprocessing could be done regarding the normalization of the body points in order to become irrelevant from their absolute location in the image. This proved to be necessary after investigating the model efficiency in real world application by altering the angle of captured video frames in relation with the corresponding angle of the initial *GSL* dataset. In this case, the model cannot learn the point positions relative to each other but instead it learns only the normalized position in the image. A possible solution could be a division of all used points with the normalized magnitude of the shoulder points which would make the values of body points coordinates to be expressed by the appropriate relativity. Another suggestion could be the implementation of data augmentation techniques such as image tilting and image angle altering, zooming in and out as well as introducing uniform image noise. The tests performed using the body points magnitude and angles of two consecutive frames falls in the category of position relativity dependency, but the outcomes with the general model setup were not encouraging and limited attention was given to improve the performance.

Concluding, the use of image blurring did not have a positive effect on the outcome. Therefore, we could examine the utilization of *GANs* architecture in order to eliminate the motion blur caused by the rapid movement of body parts, as it is suggested for this specific phenomenon.

References

[1] N. Adaloglou, T. Chatzis, I. Papastratis, A. Stergioulas, G. T. Papadopoulos, V. Zacharopoulou, G. J. Xydopoulos, K. Atzakas, D. Papazachariou and P. Daras, “A Comprehensive Study on Sign Language Recognition Methods,” *IEEE Transactions on Multimedia*, Apr. 2021, doi: 10.1109/TMM.2021.3070438.

[2] N. H. Dardas and N. D. Georganas, “Real-Time Hand Gesture Detection and Recognition Using Bag-of-Features and Support Vector Machine Techniques,” *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 1, pp. 3592 - 3607, Nov. 2011, doi: 10.1109/TIM.2011.2161140.

[3] F. Obaid, A. Babadi and Ah. Yoosofan, “Hand Gesture Recognition in Video Sequences Using Deep Convolutional and Recurrent Neural Networks,” *Applied Computer Systems*, vol. 25, no.1, pp. 57-61, May 2020, doi: 10.2478/acss-2020-0007

[4] P. Narayana, R. J. Beveridge and B. A. Draper, “Gesture Recognition: Focus on the Hands,” *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5235-5244, doi: 10.1109/CVPR.2018.00549.

[5] O. Köpüklü, A. Gunduz, N. Kose and G. Rigoll, “Real-time Hand Gesture Detection and Classification Using Convolutional Neural Networks,” *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019),* 2019, pp. 1-8, doi: 10.1109/FG.2019.8756576.

[6] K. Zhao, K. Zhang, Y. Zhai, D. Wang, J. Su, “Real-Time Sign Language Recognition Based on Video Stream,” *International Journal of Systems, Control and Communications*, vol. 12, no. 2, pp. 91-174, doi: 10.1504/IJSCC.2021.114616.

[7] P. Schneider, R. Memmesheimer, I. Kramer and D. Paulus, “Gesture Recognition in RGB Videos Using Human Body Keypoints and Dynamic Time Warping,” in *RoboCup 2019: Robot World Cup XXIII, 2-8 July, 2019, Sydney, Australia,* S. Chalup, T. Niemueller, J. Suthakorn and M. A. Williams, Ed. Springer International Publishing, pp. 281-293, doi: 10.1007/978-3-030-35699-6_22.

[8] F. M. Noori, B. Wallace, M. Z. Uddin and J. Tørresen, “A Robust Human Activity Recognition Approach Using OpenPose, Motion Features, and Deep Recurrent Neural Network,“ in *21st Scandinavian Conference on Image Analysis (SCIA 2019), Norrköping, Sweden, June 11–13, 2019*, M. Felsberg, P. E. Forssén, I. M. Sintorn and J. Unger, Ed. Springer International Publishing, pp 299-310, doi: 10.1007/978-3-030-20205-7_25.