GREEK LETTER SIGN LANGUAGE RECOGNITION

 Github Project

Marogianni Sofia, 2022202204017
Papadopoulos Nikolaos, 2022202204025
03/07/2023

Contents

1 Introduction

1.1 Greek Sign Language

Greek Sign Language is a sign language used by the Greek deaf community. Greek Sign has been legally recognized as the official language area of the Deaf community for educational purposes in Greece since 2000. The Greek Sign Language is estimated to be used by some 40,600 people.

The Greek Sign Language is not international. Each country develops its own sign language with fundamentally different meanings and a different alphabet. There are both common features and differences at a morphological level. Nevertheless, the Deaf of different states can communicate comfortably through the International Sign, which is essentially a code that serves the simple daily communication needs. As is the case with the spoken languages, dialects exist in each country.

The educational material that exists in the Greek Sign Language is limited. Since 2010 there has been a steady increase in the production of accessible educational material in electronic and printed form for deaf students.

Each letter of the Greek alphabet is represented by one of 24 possible signs in the Greek Sign Alphabet (GSL). Each letter's symbol is displayed in the image below. Nearly half of them, such as the letters A and B, are shared with American Sign Language (ASL), but the remaining are unique.

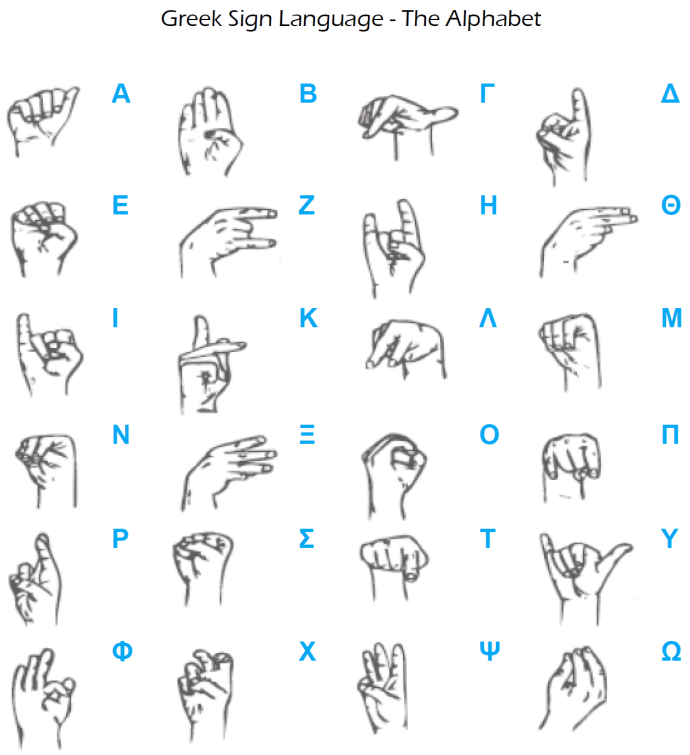


Figure 1: 'The Greek Sign Language alphabet.'

1.2 Purpose of the Project

This project's goal is to recognize **Greek Sign Letters** as they are made by a user in front of a camera. So, after running our model and accessing the laptop's camera, we sign Greek letters, and the output is displayed in the terminal.

We choose 6 letters ($\mathbf{B}, \mathbf{\Gamma}, \mathbf{\Theta}, \mathbf{H}, \mathbf{Z}, \mathbf{\Phi}$) to train and test our model.

We came to the conclusion that CNN and VGG are 2 architects we should use after thoroughly analyzing the issue. As we will explain later, the CNN model was created from scratch, whereas the VGG was transferred with its weights and only the last layer was modified.

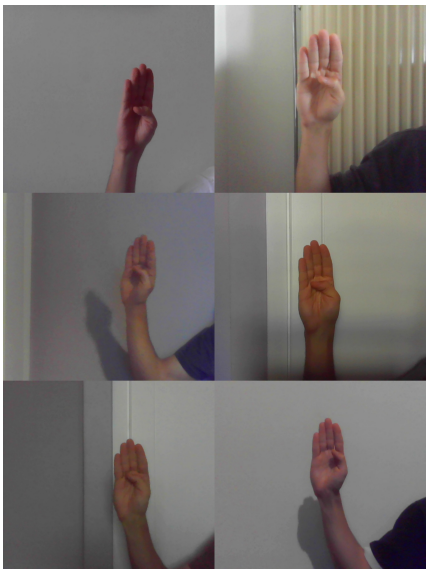
2 Dataset creation

Initially, we were looking for a GSL dataset in kaggle and other data sources on the internet, however we couldn't find any source available. That's why we made the decision to start from scratch by creating a new dataset for the six letters we indicated previously. The plan for the generated dataset was to take frames 3 times per day (morning / evening / night) for 2 days, having different backgrounds and different lighting. Regarding the lighting, we tried to capture images with both technical light (coming from a different side each time, occasionally shadowing our hands) and natural light (both inside and outside the house). The final dataset consists of 3 types of output, rgb frames, gray frames and csv files with 784 (28x28) pixels for each shot.

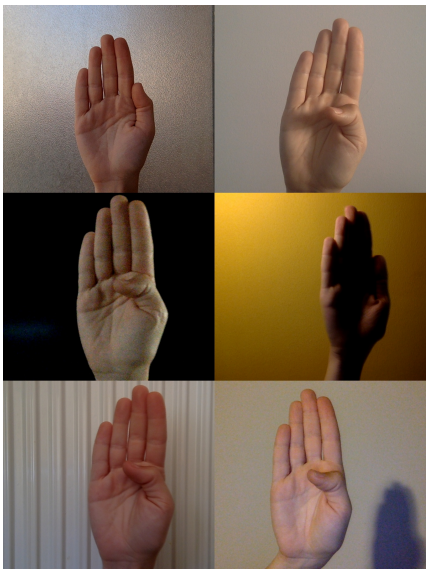
Dataset created by 2 users, the first user took 100 shots per letter for each session and the second user took 50 shots per letter. The final dataset consists of 900 rgb frames, 900 gray frames (28x28) and 900 rows with pixels (784 pixels per row) retrieved from gray frames.

The variety of the frames for each letter, regarding the zoom in/out, the location of the hand in front of the camera, as well as the personal way of each signer, is really important for the training of our model and it helps us to avoid overfitting. Because of this, we made an effort to alter the location of our hand in each photo and produce a collection of frames with a wide range of features. The approach of data augmentation, which enhances the training set by making updated copies of a dataset using existing data, was another way to accomplish so. However, since the frame capturing was a very rapid operation, we chose to construct the entire dataset on our own.

Below you can find a sample of letter B, for the 12 different sessions (6 by Signer 1 and 6 by Signer 2):



(a) Sample of letter B for the 6 sessions - Signer 1



(b) Sample of letter B for the 6 sessions - Signer 2

Figure 2: Sample of the letter B for the 6 sessions of the two users.

Signer 1 took 50 shots for each letter/session, so we have 300 rgb pictures in total, 300 gray pictures (28x28) and 36 csv files with the pixels corresponding to each letter/gray frame.

Signer 2 took 100 shots for each letter/session, so we have 600 rgb pictures in total, 600 gray pictures (28x28) and 36 csv files with the pixels corresponding to each letter/gray frame.

For the purpose of training our model, the dataset has been divided into three sections: train, test, and validation. Photos from six sessions—three taken by Signer 1 and three by Signer 2-are included in the train set, four sessions—two taken by Signer 1 and two by Signer 2—are included in the validation set, and two sessions—one taken by Signer 1 and one by Signer 2-are included in the test set. We divided the dataset in this way to prevent having frames from the same session with the same background and features in more than one part.

3 CNN model

In order to achieve the best **CNN model** we start of with a small network and continue with deepening the network to find the best architecture. The dataset has been preprocessed by converting the photos to size 28x28 and grayscale. For each session, each letter has a corresponding 'Letter-Session.csv' with $\mathbf{28 * 28 = 784}$ **columns** corresponding to each pixel, and $\boldsymbol{n}$ **rows**, each one representing one photo.

The baseline **CNN** architecture is depicted below:

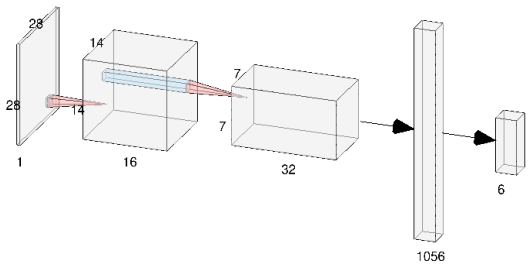


Figure 3: 'Baseline architecture of the CNN model.'

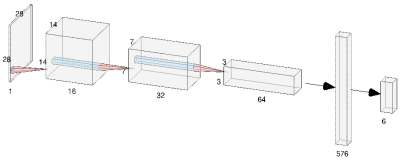
Each convolutional layer of the network includes:

- Conv2d(d_in, d_out, kernel=3, stride=1, padding=1)
- BatchNorm2d(d_out)
- ReLU(inplace=True)
- Dropout(d)
- MaxPool2d(kernel=2, stride=2)

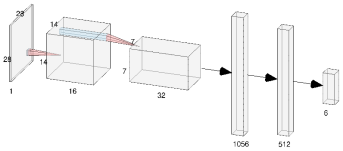
The optimization strategy we followed for the development of the CNN model was:

- Small depth → Large depth
- Different batch sizes (32 → 2700)
- Tune learning rate (0.01-0.0001)
- Regularization if needed (Dropout-L2)

The larger depth architectures that were proposed were:



(a) Added depth in the convolutional layers of the model.

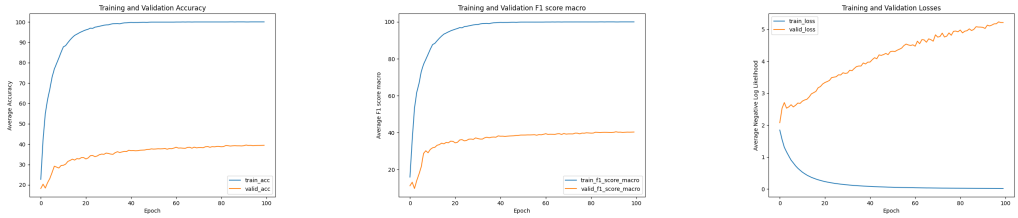


(b) Added depth in the classification layers of the model.

Figure 4: The two alternative architecture proposals.

After several different runs, we concluded that the best CNN model has the following parameters:

- Baseline architecture
- Lr = 0.001
- batch_size = 1024
- n_epochs = 100
- Patience = 15
- Dropout = 0.
- No L2-Regularization



(a) Accuracy learning curve. (b) F1 macro learning curve. (c) Loss learning curve.

Figure 5: The learning curve figures of the final CNN model

The above figures show a large difference between the train and the validation sets. This overfitting of the training data and lack of generalisation is of course due to the small train dataset. Reguralisation techniques to imporove generalisation such as L2-Reguralisation and Dropout did not improve results. Furthermore, a deeper model architecture did not manage to improve results.

The best CNN model results are as follows:

- **"valid_f1_score_macro_best"**: 0.4036
- **"valid_best_acc"**: 0.3944

Letter	**f1_score**
B	0.44
Γ	0.31
Z	0.36
H	0.52
Θ	0.35
Φ	0.42

4 VGG-Transfer model

VGG stands for Visual Geometry Group; it is a standard deep Convolutional Neural Network (CNN) architecture with multiple layers. The "deep" refers to the number of layers with VGG-16 or VGG-19 consisting of 16 and 19 convolutional layers. The VGG architecture is the basis of ground-breaking object recognition models. Developed as a deep neural network, the VGGNet also surpasses baselines on many tasks and datasets beyond ImageNet. Moreover, it is now still one of the most popular image recognition architectures.

Since CNN model didn't work as we expected, we looked for other architectures to overcome our problems. One solution was transfer learning from a VGG16 model, as this would utilize the already trained network which has proved to be really good to image recognition. For this purpose we cut the last classification layer (fc8) and replaced it with a costum layer corresponding to 6 class output.

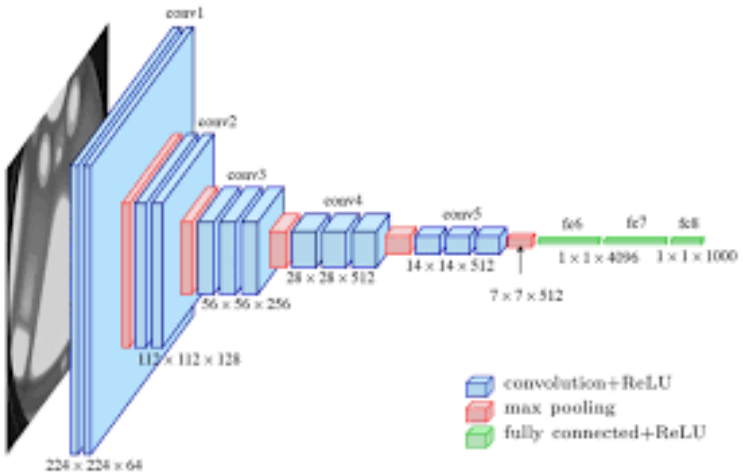


Figure 6: 'VGG Model architecture.'

To get the optimal model we used the following plan:

- 1 Unfrozen Classification Layer → 3 Unfrozen Classification Layers
- Different batch sizes (32-256)
- Tune learning rate (0.01-0.0001)
- Dropout in the last custom layer (0.2-0.4)

Following the procedure explained above we trained the model for different hyperparameters and trainable classification layers. The two best models are as follows:

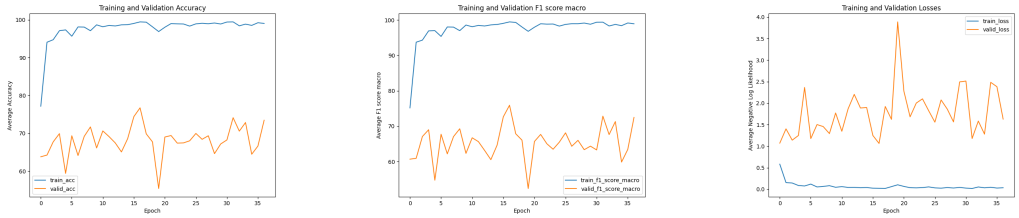
1. **VGG-Transfer_full_unfrozen**

   - 3 of the 3 classification layers trainable
   - Lr = 0.001
   - batch_size = 64
   - n_epochs = 100
   - Patience = 20
   - Dropout = 0.2

   The results of the first model are:

   - **"valid_f1_score_macro_best"**: 0.76
   - **"valid_best_acc"**: 0.77

Letter	**f1_score**
B	0.69
Γ	0.85
Z	0.65
H	0.92
Θ	0.69
Φ	0.76



(a) Accuracy learning curve. (b) F1 macro learning curve. (c) Loss learning curve.

Figure 7: The learning curve figures of the first candidate VGG-transfer model

The above figures show a smaller difference between validation and training set, in contrast with the CNN model results. On the other hands the learning curves seem to be turbulent, something that could not be prevented with a lower learning rate or a lower dropout.

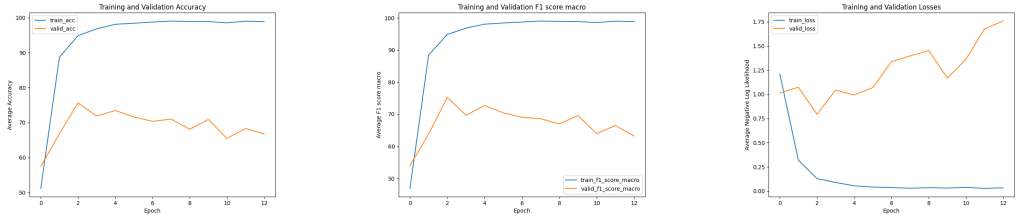
2. VGG-transfer_2_unfrozen

- 2 of the 3 classification layers trainable
- Lr = 0.001
- batch_size = 248
- n_epochs = 100
- Patience = 10
- Dropout = 0.4

The results of the second model are:

- **"valid_f1_score_macro_best"**: 0.75
- **"valid_best_acc"**: 0.76

Letter	**f1_score**
B	0.6
Γ	0.89
Z	0.72
H	0.94
Θ	0.68
Φ	0.67



(a) Accuracy learning curve. (b) F1 macro learning curve. (c) Loss learning curve.

Figure 8: The learning curve figures of the second candidate VGG-transfer model

We can see smoother learning curves in contrast with the first candidate model, although again the validation curve does not seem to converge with the training curve.

5 Final model and results

Finally we can test the 3 models in the independent test set. First, we test the CNN model which had the lowest performance out of the three:

1. CNN model

- **"valid_f1_score_macro_best"**: 0.56
- **"valid_best_acc"**: 0.55

Letter	f1_score
B	0.54
Γ	0.3
Z	0.52
H	0.66
Θ	0.77
Φ	0.55

2. VGG-Transfer_full_unfrozen

- **"valid_f1_score_macro_best"**: 0.85
- **"valid_best_acc"**: 0.86

Letter	f1_score
B	0.84
Γ	0.97
Z	0.84
H	0.93
Θ	0.74
Φ	0.76

3. VGG-transfer_2_unfrozen

- **"valid_f1_score_macro_best"**: 0.74
- **"valid_best_acc"**: 0.77

Letter	f1_score
B	0.82
Γ	0.97
Z	0.72
H	0.89
Θ	0.41
Φ	0.6

Out of the three models the best one overall is '2.VGG-Transfer_full_unfrozen'. With a better performance in the key metrics off accuracy and f1_macro in conjunction with a more balanced performance amongst all the letters. The minimum f1 is that of Θ:0.74 . Furthermore it is important to mention the very good performance of letter Γ. This letter had dostinctly different variation of signing between the two signers and we can see that the model produced outstanding results in understanding both variations.

6 Conclusion

In conclusion we created our own dataset, and tried 2 different model approaches, one simple CNN model and a VGG-Transfer learning model. After the hyperparameter tuning on the validation set we picked the best models -1 CNN model and 2 VGG-Transfer models- and compared them in the test set. The test set made it clear that one model was better than the other two. More precisely "VGG-Transfer_full_unfrozen" as mentioned above, produced the best results reaching [**"valid_f1_score_macro_best"**: 0.85, **"valid_best_acc"**: 0.86] and even performing great on letter Γ where there were two distinct signing variations from the two users.

6.1 Future work

Finally we mention some thoughts on methods we did not manage to use in this project but seem promising for the future to make the model more general, precise and not dependent on the background and user.

First of all, the size of the dataset was a big hurdle we had to overcome. This made the CNN model have a really low ceiling while making it hard to divide the different datasets into train, validation and test sets in order to gain the balance between learning different backgrounds and lighting and at the same time validating on some challenging and some simpler sessions. Furthermore we should mention that a cross validation technique amongst the different datasets, training and validating in different combinations in order to achieve the best model.

The use of hand-tracking algorithms in order to make the classification centered around only the hand of the user could also prove to be really useful. Finally a voting technique used for a small amount of frames could make the results even more robust and less dependent on subtle changes.