**Sign language – picture to text**

Ivona Mircheska, Tiana Paunoska

Faculty of Electrical Engineering and Information Technologies,

University of Ss. Cyril and Methodius Skopje, Macedonia

***Аbstract –* Sign language is a visual language that uses hand gestures, facial expressions, and body movementsto provide a way of communicating. The importance of the sign language is especially perceived through its utilization as a primary way of accomplishing communication for people with fully or partially impaired hearing. Depending on the country or region, sign languages can vary in great measure. The Macedonian Sign Language is widely used within the Association of deaf and hard of hearing people in Macedonia. The language has its own vocabulary, syntax and grammar which are used to convey meaning in a way that is specific to the language. Our research involves creating and implementing Convolutional Neural Network (CNN) model for detecting features from a large set of images that will lead to recognizing particular, preselected signs.**

***Keywords – sign language, convolutional neural network, real time recognition, ResNet***

1. INTRODUCTION

Sign languages are used as a primary means of communication by deaf and hard of hearing people worldwide. It is the most potent and effective way to bridge the communication gap and social interaction between them and the able people. Sign language interpreters help solve the communication gap with the hearing impaired by translating sign language into spoken words and vice versa. However, the challenges of employing interpreters are the flexible structure of sign languages combined with insufficient numbers of expert sign language interpreters across the globe.Therefore, there is a need for a technology-based system that can complement conventional sign language interpreters. One of the main challenges is the variety of sign language that occur. Each sign language has its own grammar and syntax. Sometimes one gesture represents a whole word while other times it can represent only a single alphabet or number. There are mainly three variants of sign language:

1) Non-manual features: Tongue, facial expression, body poses, and hand gestures - all of them are used to communicate.

2) Word level sign spelling: Each gesture represents a whole word.

3) Finger vocabulary: One gesture represents one alphabet/numbers

In order to recognize a certain sign from an image, we used 2 methods. The first method was to predict the image with a model using already existing images, with the second method, a sign language prediction was made using a camera in real time.

1. DATASET

The dataset contains 12 180 images with size 224 x 224, organized in 58 separate folders which correspond to the number of signs used in our project. Each folder has 210 image samples, all made under different circumstances, such as different angle and lightning, position of hands and surrounding all in order to obtain data diversity. The 58 different classes contain 28 letters plus several randomly chosen Macedonian words.

1. *Preparing the data*

A csv file was created which contains the file paths and target values for each image. In order to label each sign, the LabelBinarizer from the Scikit-Learn library was used. The LabelBinarizer binarizes the labels in a one-vs-all fashion and returns the one-hot encoded vectors. For instance, if we have label ‘A’ which is first in our list of labels, the LabelBinarizer will create a vector with length same as the number of classes, in this case 58.

The first value in the vector will be ‘1’ and the rest will be ‘0’s since the one-hot encoding only places ‘1’ in the position corresponding to the index of the class, while the rest of the values are set to zero. The index of the class is afterwards written as a target value for the label in question in the ‘data.csv’.



Figure 1. Images from the dataset

1. *Data Augmentation*

To make the model more generalized and avoid over-ﬁtting, data augmentation is a well-proven and common technique. We apply data augmentation in the images that are used for training. The following table shows the augmentation technique and values up to which the data is augmented. Each image has only one augmentation version, which means that the number of images used during training stays the same, only the images are being modified.

|  |  |
| --- | --- |
| Technique | Value |
| Rotation range | 10 |
| Width shift range | 0.03 |
| Height shift range | 0.03 |
| Shear range | 0.03 |
| Zoom range | 0.03 |
| Horizontal flip | True |

Table 1. Augmentation techniques and values

The dataset before training is split into two parts: training set consisting of 9 135 images which represent 75% of the dataset and validation set consisting of 3 045 images which represent 25% of the dataset. In the test set initially only 58 images are used for testing. The distribution of the classes in the test set is equal, due the fact that each image represents an instance from one of the classes.

1. MODEL

For our specific task, we built several models, but only one model showed very high validation and test accuracy, and was very good at generalizing new unseen data. The model was created using ResNet50 architecture as a base model without using the top classifier. Instead, we used our own built classifier. ResNet50 is a deep convolutional neural network (CNN) architecture for image classification tasks. The architecture of ResNet50 consists of several blocks of convolutional layers with the batch normalization layers and activation functions, followed by an average pooling layer and a final fully connected layer. Each block consists of multiple residual units, which are the building blocks of the ResNet architecture. Each residual unit consists of three convolutional layers together with the batch normalization layers and activation functions. The key idea behind residual units is that with them we can make use of residual connections between each residual unit, but also we can use the so called skip connections, where the input of the residual unit is directly transferred to the output and it is summed with the initial output of the residual unit. This can manage to avoid the problems with vanishing gradients because the identity of the original input is preserved and passed directly on the residual connection leading to the next residual unit. In summary, residual units are an important component of the ResNet architecture, and they play a key role in allowing the ResNet50 model to learn deep representations for image classification tasks effectively.

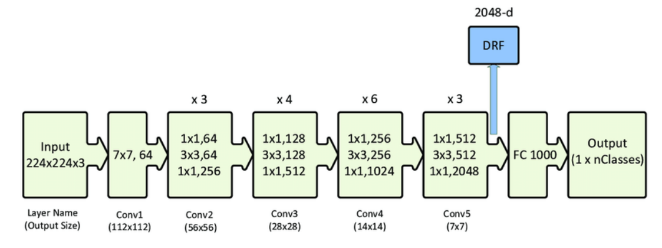


Figure 2. Detailed overview of the ResNet50 architecture

Our fully connected classifier has a flattening layer, which is used to flatten the output from the base model into a one-dimensional array, which is then inputted to the next layer. After the flattening layer there are four dense layers. The first dense layer has 256 neurons and uses the ReLU activation function, the second has 128 neurons and uses the ReLU activation function, the third has 64 neurons and uses the ReLU activation function, and the fourth has 32 neurons and uses the ReLU activation function. And the final layer is a dense layer with num\_classes neurons and a softmax activation function, which outputs the predicted class probabilities. Initially, our classifier together with the base from ResNet50 is trained in which case all of the layers from ResNet50 are set to untrainable. However, pre-trained models like ResNet50 can be useful for sign language gesture recognition tasks by fine-tuning the model on a smaller sign language gesture recognition dataset. Because of that, after the initial training, we set some of the last layers to trainable and then use them with our classifier for the new training. The last layers are effective of learning complex shapes and textures of signs and recognizing them in new images. With unfreezing some of the last layers we utilize batch normalization from the convolution block which can be particularly useful, since our task involves image data that can have large variations in brightness, contrast, and colour.

1. EXPERIMENTAL RESULTS

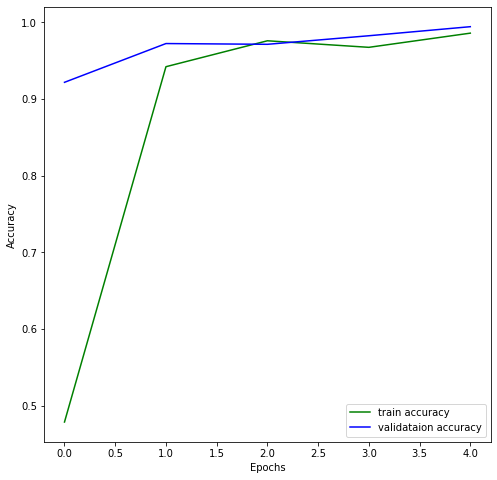
Using fine-tuning gave better results regarding the test accuracy compared to the initial way of training. Before unfreezing some of the layers the test accuracy was 96.55 %. After unfreezing, the test accuracy was 100 %. For these results, the test dataset only contained 58 images, each image corresponding to each class. The first model was trained with 5 epochs and batch size of 64. In figures 3 and 4 we analyse the accuracies and loss values for the train and validation sets.

Figure 3. Train and validation accuracy for the first model

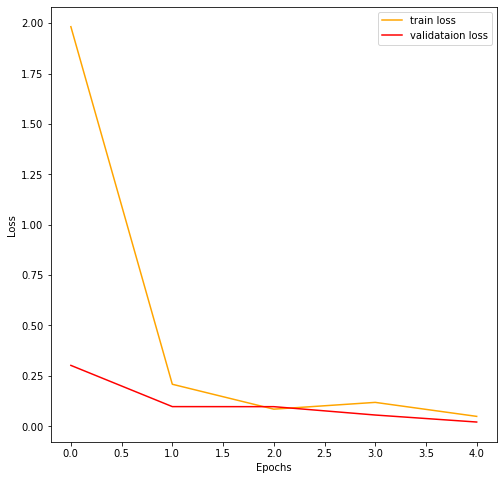
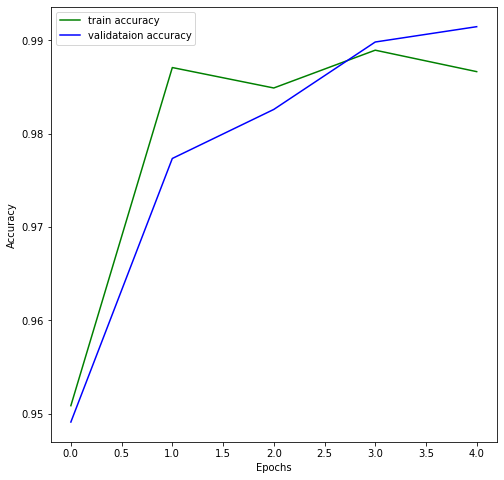


Figure 4. Train and validation loss for the first model

We can see from the Figure 3 that starting from the second epoch onward the train and validation accuracies have close values. There is a slight intersect in the third epoch, but after the third epoch the train and validation accuracies have very close values with the validation accuracy being higher. Similar conclusion can be drawn regarding the loss values. The second model with the fine-tuning, which is the best model for our task, was trained with 5 epochs and batch size of 64. In figures 5 and 6 we analyse the accuracies and loss values for the train and validation sets.



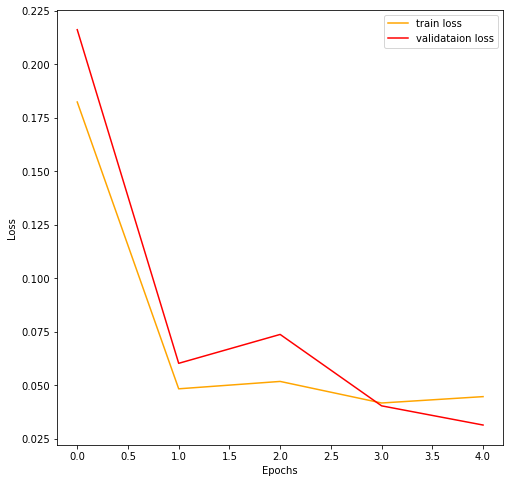
Figure 5. Train and validation accuracy for the second (best) model

Figure 6. Train and validation loss for the second (best) model

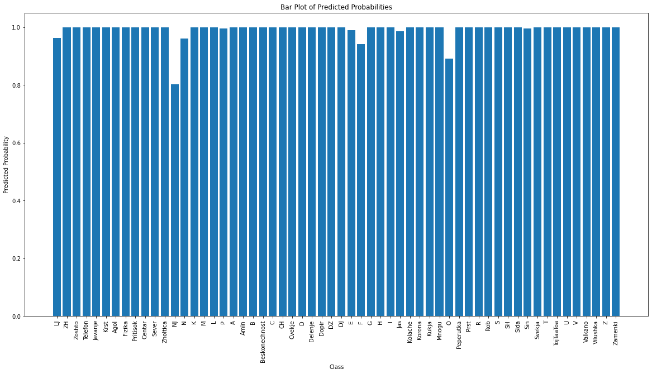
The values for both the accuracies and loss are increasingly close. After the third epoch there is a stabilization and the validation values are higher and lower for the accuracy and loss, respectively. We updated the test dataset with new images, made under different circumstances. It is evident that the model generalizes well with some chances for error.

Figure 7. Probability for each class

The Figure 7 displays the probabilities how likely the model will correctly predict each class.

1. TESTING THE DATA

The ‘test.csv’ file contained the file paths and the target value for each test image. For the model testing besides the 58 images that were provided to us, we added additional 11 images. 9 images out of the 11 were correctly classified, while the other 2 images were not. This is due to the fact that the predicted class and the actual class were roughly similar.



Figure 8. Actual class ‘Fizika’ predicted correctly by the model



Figure 9. Actual class ‘Beskonechnost’ mistaken by the model as class ‘Fizika’

1. SIGN LANGUAGE RECOGNITION IN REAL TIME

For recognising signs in real time using web camera the OpenCV() library is used. The OpenCV() library provides a simple way of connecting the code with the default camera. With VideoCapture() a connection with the default camera is established. After that, the window for capturing the frames is opened. The rectangle area that is displayed is set to capture the hand gesture and has a size of 224x224 matching the size of all the images used in the task. Each frame that is captured is saved as a ‘frame.jpg’ file in the test directory. The file ‘frame.jpg’ is being rewritten each time a new frame is captured. The ‘frame.jpg’ is then used as an input for the model. The position of the hands needs to be carefully placed in the rectangle area so that ‘frame.jpg’ can have a proper look and the model can make a satisfactory prediction. Additionally, the rectangle area manages to avoid the capture of certain unwanted background elements in the frame and with that the chances of correct prediction are increased. Our model showed accurate predictions in most cases with some errors that occurred due to wrong position of the hand, incorrectly made gesture or bad lightning.

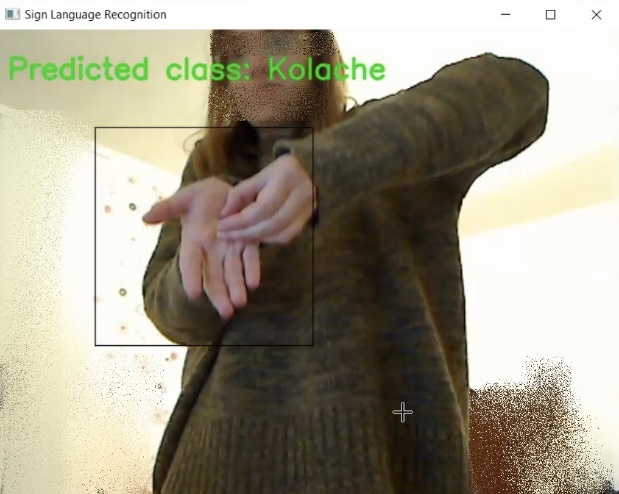


Figure 10. The camera recognises the class “Kolache”

1. CONCLUSION

In this task we proposed a ResNet50 based model for sign language recognition. Although for most part of the cases the model predicted the actual class, there were still some evident mistakes. In order to overcome such problems, the model should be trained on more diverse data and the images should be made with additional types of lightning, background and position of the hands. With this we can create a more efficient model for sign language recognition.

REFERENCES

[1] <https://towardsdatascience.com/sign-language-to-text-using-deep-learning-7f9c8018c593>

[2] <https://towardsdatascience.com/batch-normalisation-in-deep-neural-network-ce65dd9e8dbf#:~:text=Batch%20normalization%20is%20a%20technique,to%20train%20deep%20neural%20networks>

[3] <https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758>

[4] <https://kharshit.github.io/blog/2018/09/07/skip-connections-and-residual-blocks>

[5] <https://datagen.tech/guides/computer-vision/resnet-50/>