ASL sign classification through artificial intelligence

Roberto Calabrese
Department of Computer Science
University of Salerno (Italy)
r.calabrese21@studenti.unisa.it

ABSTRACT

American Sign Language (ASL) is a complete, natural language that has the same linguistic properties as spoken languages, with grammar that differs from English. ASL is expressed by movements of the hands and face. It is the primary language of many North Americans who are deaf and hard of hearing and is used by some hearing people as well.

Memorizing the American Sign Language alphabet is the first step when learning American Sign Language and most new sign language students rely on fingerspelling from the ASL alphabet when they don't know the sign for something.

The aim of this project is to model a network that can classify a ASL sign by an image using artificial intelligence.

RELATED WORKS

The importance of ASL sign classification through artificial intelligence has been examined in many past works. The following studies provide a general idea regarding the topic.

The study conducted by A. Kasapbasi et al. [1] aimed to develop an ASL recognition dataset and use it in the deep learning model which depends on neural networks to interpret gestures of sign language and hand poses to natural language. The proposed system may be considered a promising solution in medical applications that use deep learning with superior accuracy.

Another study conducted by Mohamed Z. Amrani et al. [2] presented and assessed a multi-sensory hand pattern recognizer. They experimentally evaluated multiple features from the selected sensors, detected the most significant ones, and set the best combination possible to reach optimal hand shape detection. They evaluated the model on the classification of the American Sign Language (ASL) reaching an average classification of 98.31% with the optimal subset.

The work of Mehrez Boulares et al. [3] aimed at creating an automatic sign selection and validation solution based on unsupervised clustering of sign motion parameters related to the different sign replicas. They conducted an experimental study to validate 300 ASL signs based on four unsupervised clustering methods, namely, Kernel PCA Kmeans, GMM, Spectral clustering and kernel Kmeans. They concluded that their sign validation process using Spectral clustering method allowed them to select the right sign replicas to be used to generate the user sign signature.

Wenjin Tao et al. [4] used Convolutional Neural Networks with multiview augmentation and inference fusion to recognize ASL alphabet. Multiview augmentation is more effective than traditional augmentation methods. Multiview inference fusion further improves the inference performance. They concluded that their method significantly outperformed the state-of-the-arts on the public datasets.

The study of Qutaishat Munib et al. [5] aimed at developing a system for automatic translation of static gestures of alphabets and signs in American Sign Language. In doing so, they used Hough transform and neural networks trained to recognize signs. Their system does not rely on using any gloves or visual markings to achieve the recognition task. Instead, it deals with images of bare hands, which allows the user to interact with the system in a natural way. An image is processed and converted to a feature vector that will be compared with the feature vectors of a training set of signs. The extracted features are not affected by the rotation, scaling or translation of the gesture within the image, which makes the system more flexible. The system was implemented and tested using a data set of 300 samples of hand sign images, 15 images for each sign. Experiments revealed that their system was able to recognize selected ASL signs with an accuracy of 92.3%.

PROPOSED METHOD

To reach the desired goal, a convolutional neural network was developed, as they are optimal for image classification. A convolutional neural network (also called ConvNet or CNN) is a deep learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets can learn these filters.

The architecture of a ConvNet is analogous to that of the connectivity pattern of neurons in the human brain and was inspired by the organization of the visual cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. A collection of such fields overlaps to cover the entire visual area.

The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.

The convolutional neural network created for the project is a sequential classifier, that means that the information flows from the input to the output. Convolutional layers are the major building blocks used in convolutional neural networks. A convolution is the simple application of a filter to an input that results in an activation. Three convolutional layers were added, applying 64 filters for the first one, 128 filters for the second one and 256 filters for the third one. Each filter is squared with a shape of 3 rows and 3 columns. 200x200 pixel images compose the dataset but they were resized to 32x32 pixel images. All images are represented in the RGB domain, since each pixel is represented with 3 different values. The activation function used is ReLU.

Maximum pooling, or max pooling, is a pooling operation that calculates the maximum, or largest, value in each patch of each feature map. The results are down sampled or pooled feature maps that highlight the most present feature in the patch, not the average presence of the feature in the case of average pooling. Three maximum pooling layers were added with shape 2x2; this means that in all regions with shape 2x2 in the feature map the maximum value is saved.

Batch normalization was used as well. Batch normalization is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch. This has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks.

The flatten layer was added; it flattens the input and does not affect the batch size, it is only responsible for taking as input the pooled layers and turn them into a vector.

The Dropout layer was added; it randomly sets input units to 0 with a frequency of 0.5 at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by 1/(1 – 0.5) such that the sum over all inputs is unchanged.

In any neural network, a Dense layer is a layer that is deeply connected with its preceding layer which means the neurons of the layer are connected to every neuron of its preceding layer. This layer is the most used layer in artificial neural network networks. A dense network with 1024 units was added and the activation function used is sigmoid.

The dataset was divided into two sets: 90% of the dataset was used for the training set and the remaining 10% was used for the test set. After the images were resized, the training was started. After the training, the model was tested on other images that were not a part of the original dataset.

EXPERIMENTS

The dataset used is an image dataset for the American Sign Language alphabet [6]. The training dataset contains 87,000 images, which are 200x200 pixel images. There are 29 classes, of which 26 are for the letters of the English alphabet and the remaining 3 are for “delete”, “nothing” and “space”. The original test set contains only 28 images.

Each image depicts a hand gesture that corresponds to one of the 29 classes used (Figure 1). The background of the images is simple and is composed of few colored parts. The light of the images is variable.

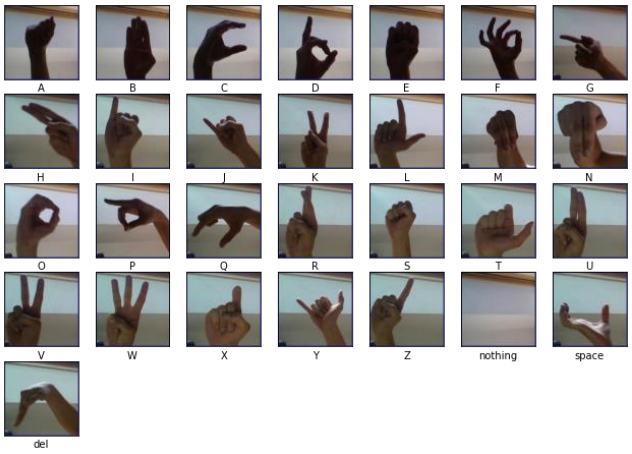

Figure 1: Dataset sample images for each of the 29 classes

A first configuration of the network was composed of only one convolutional layer formed of 32 filters, one maximum pooling layer and a flatten layer (Figure 2).

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 196, 196, 32)      2432
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 24, 24, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 18432)             0
=================================================================
Total params: 2,432
Trainable params: 2,432
Non-trainable params: 0
_________________________________________________________________
```

Figure 2: First model configuration

The training of this model was completed with 25 epochs, 100 steps per epoch and 29 validation steps. The training for this model took 58 minutes. The training and validation accuracies were respectively 88% and 44% (Figure 3).

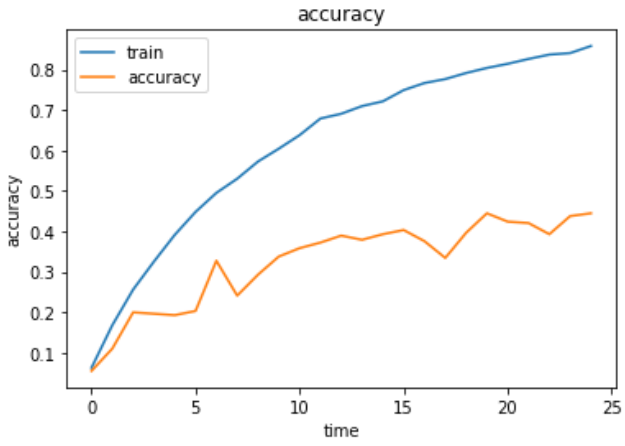

Figure 3: First model training and validation accuracy

The training and validation losses were respectively 3% and 10% (Figure 4).

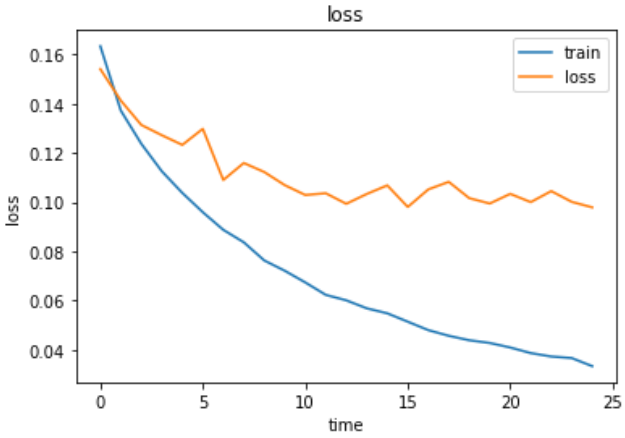

Figure 4: First model training and validation loss

To test the model, considering the presence of only 28 images in the original test set, other images were created to fill the test set. A folder for each class was created in the test set folder. Each folder contains the original image already present in the test set and other images captured manually. The images added represent hand gestures that correspond to the classes, they have a neutral background and different types of light were used (Figure 5).

Figure 5: One of the images manually added in the test set

The accuracy obtained on the test set under this first configuration of weights was 6%.

At this point, the structure of the network was modified. Three convolutional layers and three maximum pooling layers were added, as well as other additional layers (Figure 6).

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 200, 200, 64)      1792
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 100, 100, 64)      0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 100, 100, 128)     73856
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 50, 50, 128)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 50, 50, 256)       295168
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 25, 25, 256)       0
_________________________________________________________________
batch_normalization_1 (Batch (None, 25, 25, 256)       1024
_________________________________________________________________
flatten_1 (Flatten)          (None, 160000)            0
_________________________________________________________________
dropout_1 (Dropout)          (None, 160000)            0
=================================================================
Total params: 371,840
Trainable params: 371,328
Non-trainable params: 512
_________________________________________________________________
```

Figure 6: Second model configuration

The training of this model was completed with 5 epochs, 221 steps per epoch and 29 validation steps. The training for this model took 154 minutes. The training and validation accuracies were respectively 34% and 12% (Figure 7).

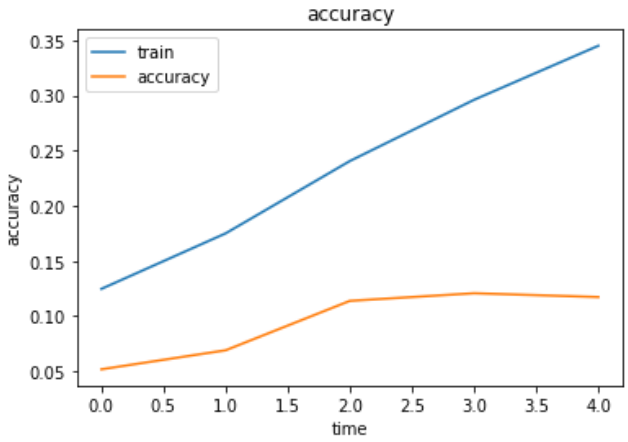

Figure 7: Second model training and validation accuracy

The training and validation losses were respectively 11% and 14% (Figure 8).

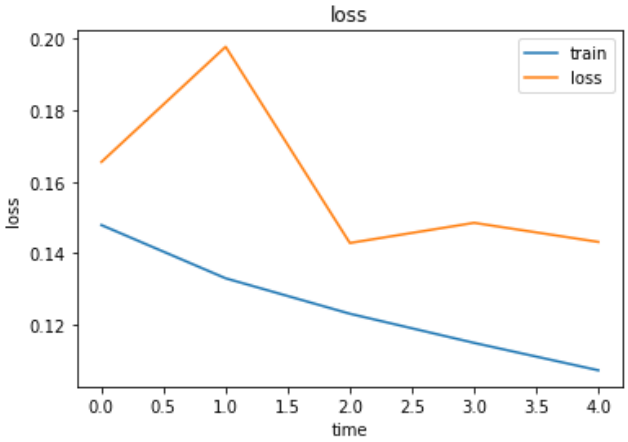

Figure 8: Second model training and validation loss

The accuracy obtained on the test set under this second configuration of weights was 3%.

Considering the low accuracies obtained, the model was further modified. The images were resized to 32x32 pixel images, the learning rate was set to 0.01 and other small parameters were changed (Figure 9).

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 64)        1792
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 16, 16, 128)       73856
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 128)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 8, 8, 256)         295168
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 256)         0
_________________________________________________________________
batch_normalization (BatchNo (None, 4, 4, 256)         1024
_________________________________________________________________
flatten (Flatten)            (None, 4096)              0
_________________________________________________________________
dropout (Dropout)            (None, 4096)              0
_________________________________________________________________
dense (Dense)                (None, 1024)              4195328
_________________________________________________________________
dense_1 (Dense)              (None, 29)                29725
=================================================================
Total params: 4,596,893
Trainable params: 4,596,381
Non-trainable params: 512
_________________________________________________________________
```

Figure 9: Final model configuration

The training of the final model was complete with 5 epochs and 1102 steps per epoch. The final training and validation took 26 minutes. The final training and validation accuracies were respectively 99% and 98% (Figure 10).

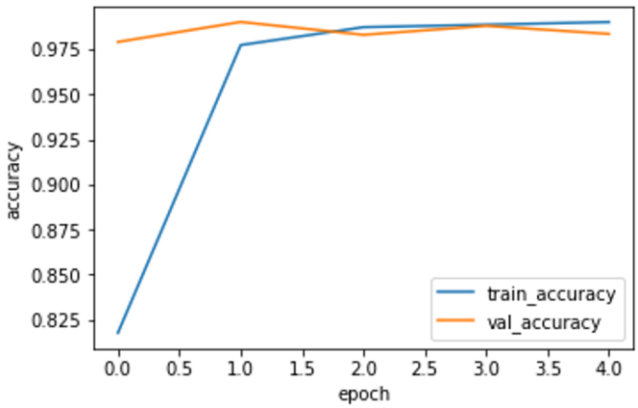

Figure 10: Final training and validation accuracy

The final training and validation losses were respectively 3% and 4% (Figure 11).

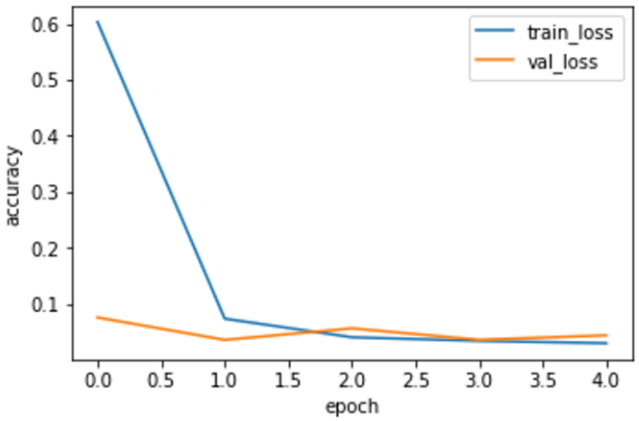

Figure 11: Final training and validation loss

For this final configuration, the testing was executed first on each of the images for each class and then on all the images of the test set. The accuracies obtained are shown in Table 1.

Class	*Accuracy percentage*
A	28%
B	43%
C	71%
D	28%
delete	43%
E	14%
F	28%
G	14%
H	28%
I	43%
J	86%
K	57%
L	100%
M	57%
N	43%
nothing	100%
O	14%
P	100%
Q	100%
R	28%
S	28%
space	43%
T	71%
U	57%
V	14%
W	57%
X	14%
Y	14%
Z	14%

Table 1: Accuracy for each class of the test set

The accuracy obtained on the total test set under this second configuration of weights was 46%.

CONCLUSION

The accuracies obtained for each class were not always optimal. However, the accuracy of 46% for the total test set is quite a good result, considering that the number of classes is 29. In the future, with a larger test set of images collected, the accuracy could improve. Additionally, a further modification of the network structure, perhaps adding more layers and modifying several parameters, could potentially improve the classification even more.

BIBLIOGRAPHY

[1] Ahmed Kasapbasi, Ahmed Eltayeb Ahmed Elbushra, Omar Al-Hardanee, Arif Yilmaz. "DeepASLR: A CNN based human computer interface for American Sign Language recognition for hearing-impaired individuals"

[2] Mohamed Z. Amrani, Christoph W.Borst, Nouara Achour. "Multi-sensory assessment for hand pattern recognition"

[3] Mehrez Boulares, Ahmed Barnawi. "Unsupervised sign language validation process based on hand-motion parameter clustering"

[4] Wenjin Tao, Ming C. Leu, Zhaozheng Yin. "American Sign Language alphabet recognition using Convolutional Neural Networks with multiview augmentation and inference fusion"

[5] Qutaishat Munib, Moussa Habeeb, Bayan Takruri, Hiba Abed Al-Malik. "American sign language (ASL) recognition based on Hough transform and neural networks"

[6] ASL Alphabet dataset: https://www.kaggle.com/grassknoted/asl-alphabet