American Sign Language Classification

Allegra Martina MARTINA.ALLEGRA14@GMAIL.COM
Saitta Susanna SUSANNASAITTA8@GMAIL.COM

1. Dataset

The original dataset is a collection of 200x200 RGB images of alphabets from the American Sign Language (see Fig. 1); it was taken from: https://www.kaggle.com/grassknoted/asl-alphabet. Specifically, 87,000 images are provided in 29 separate folders representing classes: 26 classes for the letters A-Z and 3 for Space, Delete and Nothing.

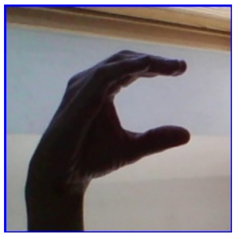

Figure 1: ASL - **C** letter.

Due to computational issues, the size of the dataset was reduced; by applying the *splitfolders* function, images were split as follows:

- train set with 17,168 samples, that is 592 per class;
- validation set with 2,929 samples, that is 101 per class;
- test set with 2,523 samples, that is 87 per class.

Furthermore, we normalized and resized data to be sure that all images were the same height and width, i.e. 200x200.

2. Trained CNNs

With the aim of performing multi-class classification, 8 Convolutional Neural Networks were trained, validated and tested for 10 epochs. For all models, batch size was set to 64, Cross-Entropy[1] was used as loss function and Stochastic Gradient Descent was chosen as optimizer algorithm, using a learning rate of 0.05. CNNs were defined as follows:
- 3 convolutional layers with respectively 8, 16 and 32, 3x3 kernels applied with a stride of

1. *CrossEntropyLoss* function in Pytorch combines the *LogSoftmax* function and the *negative log-likelihood* loss function. See Pytorch documentation on https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html.

1 and padding of 1 in each dimension and followed by a ReLu activation function and by Max polling with filter of 2x2 and stride of 2 + 1 fully connected layer with 4096 neurons followed by a ReLu activation function + classifier.
- 3 convolutional layers with respectively 8, 16 and 32, 3x3 kernels applied with a stride of 1 and padding of 1 in each dimension and followed by a ReLu activation function and by Max polling with filter of 2x2 and stride of 2 + 2 fully connected layers, with respectively 4096 and 1024 neurons, followed by a Relu activation function + classifier.
- 4 convolutional layers with respectively 8, 16 and 32, 64 3x3 kernels applied with a stride of 1 and padding of 1 in each dimension and followed by a ReLu activation function and by Max polling with filter of 2x2 and stride of 2 + 1 fully connected layer with 4096 neurons followed by a Relu activation function + classifier.
- 4 convolutional layers with respectively 8, 16, 32, 64, 3x3 kernels applied with a stride of 1 and padding of 1 in each dimension and followed by a ReLu activation function and by Max polling with filter of 2x2 and stride of 2 + 2 fully connected layers, with respectively 4096 and 1024 neurons, followed by a Relu activation function + classifier.
- 5 convolutional layers with respectively 8, 16, 32, 64, 128 3x3 kernels applied with a stride of 1 and padding of 1 in each dimension and followed by a ReLu activation function and Max polling with filter of 2x2 and stride of 2 + 1 fully connected layer with 4096 neurons followed by a Relu activation function + classifier.
- 5 convolutional layers with respectively 8, 16, 32, 64, 128, 3x3 kernels applied with a stride of 1 and padding of 1 in each dimension and followed by a ReLu activation function and Max polling with filter of 2x2 and stride of 2+ 2 fully connected layers, with respectively 4096 and 1024 neurons, followed by a Relu activation function + classifier.
- 6 convolutional layers with respectively 8, 16, 32, 64, 128, 256, 3x3 kernels applied with a stride of 1 and padding of 1 in each dimension and followed by a ReLu activation function and Max polling with filter of 2x2 and stride of 2 + 1 fully connected layer with 4096 neurons followed by a Relu activation function + classifier.
- 6 convolutional layers with respectively 8, 16, 32, 64, 128, 256, 3x3 kernels applied with a stride of 1 and padding of 1 in each dimension and followed by a ReLu activation function and by Max polling with filter of 2x2 and stride of 2 + 2 fully connected layers, with respectively 4096 and 1024 neurons, followed by a Relu activation function + classifier.

3. Experimental Results

Models were trained, validated and tested as described in the previous section.
Table 1 shows **accuracy** and **loss** results for the proposed architectures. The model with 5 Convolutional Layers and 1 Fully Connected Layer obtained the highest validation accuracy, 96.74%.

Model	Train Acc.	Train Loss	**Val. Acc.**	**Val. Loss**	Test Acc.	Test Loss
3 Conv Layers + **1** FC Layer	96.83%	0.1049	94.10%	0.2091	93.27%	0.2074
3 Conv Layers + **2** FC Layers	92.05%	0.2437	93.19%	0.2110	92.73%	0.2229
4 Conv Layers + **1** FC Layer	97.87%	0.0724	93.30%	0.2323	93.45%	0.2217
4 Conv Layers + **2** FC Layers	90.87%	0.2791	90.12%	0.2840	89.28%	0.2831
5 Conv Layers + **1** FC Layer	99.63%	0.0165	96.74%	0.1314	96.88%	0.1181
5 Conv Layers + **2** FC Layers	97.60%	0.0708	95.60%	0.1523	96.13%	0.1460
6 Conv Layers + **1** FC Layer	93.03%	0.2271	93.13%	0.2161	92.37%	0.2401
6 Conv Layers + **2** FC Layers	16.72%	2.8841	26.51%	2.4128	27.75%	2.3980

Table 1: Training, Validation and Test performance of the models.

4. Best CNN architecture

The model consists of two parts, structured as shown in Fig. 2: the **feature extraction** part is composed by five convolutional layers, followed by ReLu activation function and almost always by a max-pooling layer; the **classification** part is composed by one fully connected layer, followed by ReLu activation function and the final classification layer.
Convolutional layers apply 3x3 filters; the stride and the padding (zero padding) fixed to 1 allow us to preserve the spatial resolution after convolution. ReLu activation function adds non-linearity keeping the input size unchanged. Max-pooling layers with filters of 2x2 applied with a stride of 2, downsample every depth slice in the input by 2, along both width and height. Each of pooling layers reduces the dimensions of each feature map independently, so as to reduce the amount of parameters keeping the depth intact. The 3D output of the last max-pooling layer is flattened to a 1D vector, in order to be used as input from the fully connected layer.

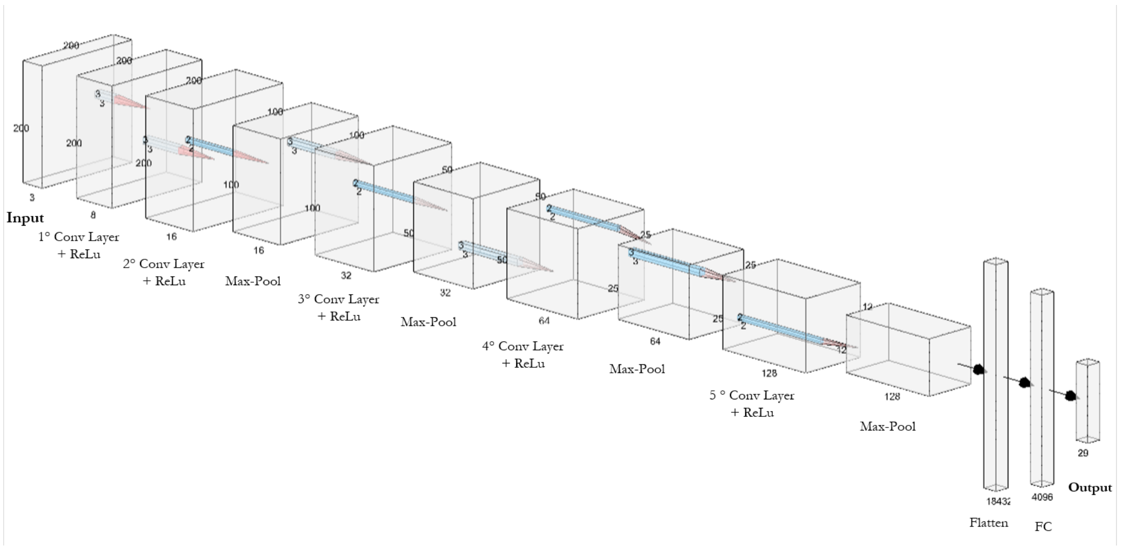

Figure 2: Best CNN architecture.

The first convolutional layer filters the 3x200x200 input image with 8 kernels of size 3x3; It provides an output volume of size 8x200x200 – i.e. 8 feature maps of size 200x200. After ReLu activation function, the second convolutional layer applies 16 filters to produce an output of size 16x200x200. It is followed by ReLu activation function and max-pooling layer which returns an output of 16x100x100. The third convolutional layer applies 32 kernels and provides an output of size 32x100x100. This is also followed by ReLu activation function and max-pooling layer which produces a 32x50x50 output. It is taken as input from the fourth convolutional layer, which applies 64 kernels, producing an output of 64x50x50. Follows ReLu activation function and the max-pooling layer, which returns a 64x25x25 output. Applying 128 kernels, the fifth convolutional layer produces an output of size 128x25x25; after ReLu activation function, the max-pooling layer returns an output volume of 124x12x12. The output is flattened in an vector of size 18432 to be provided as input of a fully connected layer; it has 4096 neurons and it is followed by ReLu activation function. Eventually, the classifier layer contains 10 neurons as the numbers of classes.