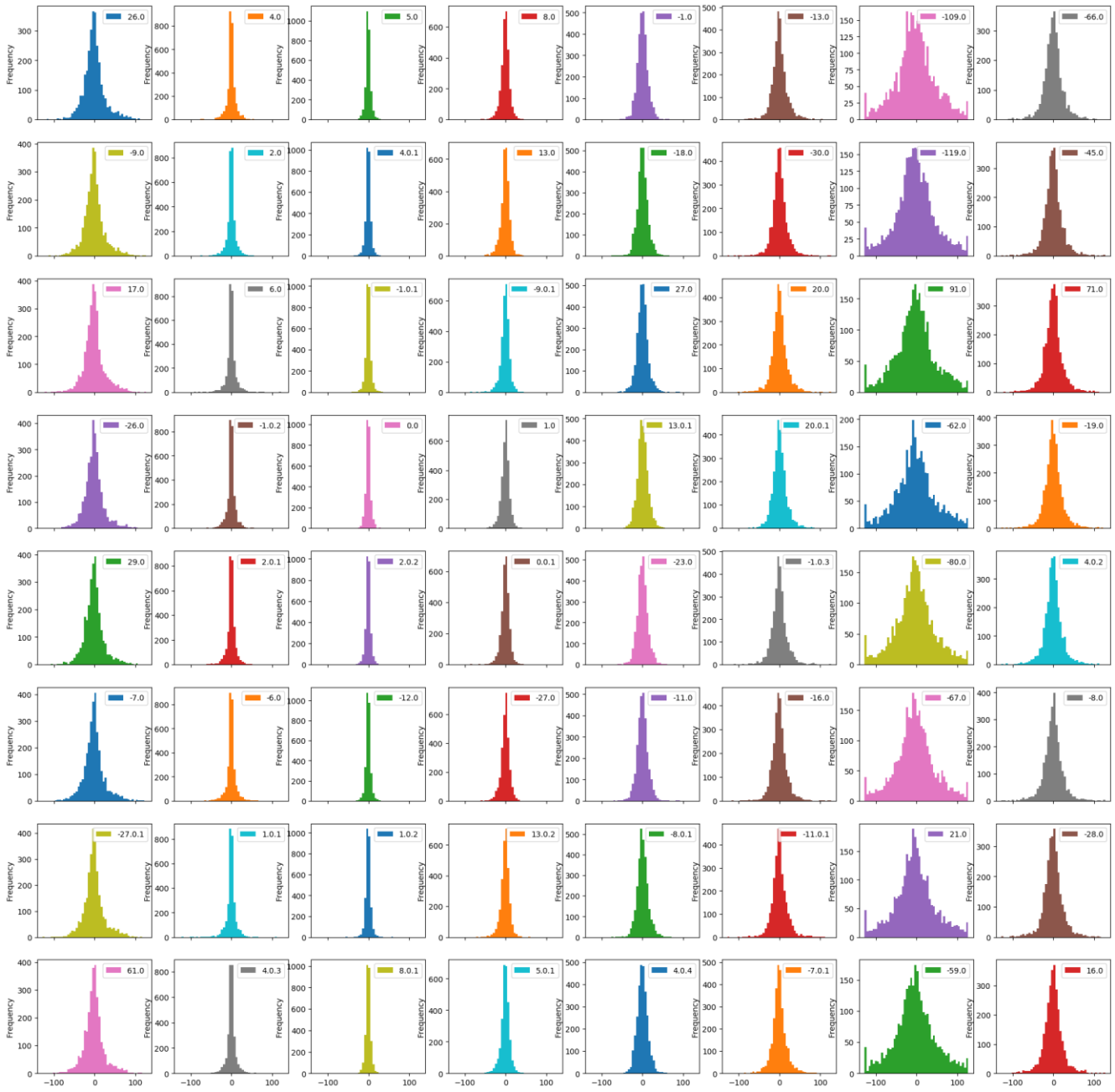
Classify gestures by reading muscle activity

A recording of human hand muscle activity producing four different hand gestures

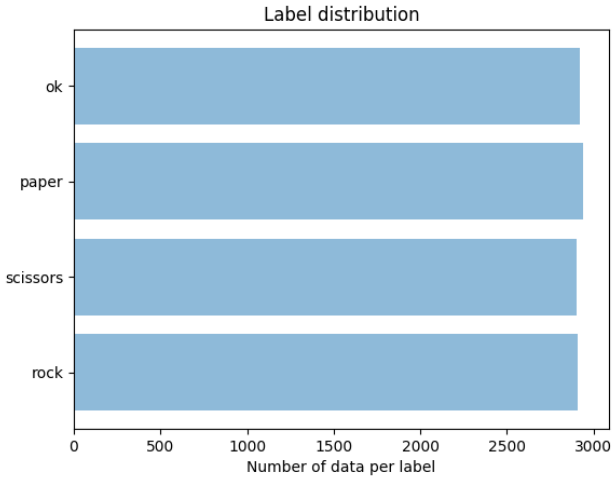
DESCRIPTION OF THE DATASET

The dataset consists of 11,674 examples, each with 64 features, distributed in 4 classes (rock - 0, scissors - 1, paper - 2, ok - 3). One line in the dataset represents the repetitive readings of 8 muscles of the human hand (reading1 muscles 1, reading1 muscles 2, ..., reading1 muscles 8, reading2 muscles 2, ... reading 8 muscles 8) The dataset was divided into 3 parts in train (80%), validation (10%) and test (10%). In Figure 1 you can see the distribution of data only from reading the muscles for gesture 0 (stone). More precisely on the horizontal direction we have the readings for the 8 distinct muscles, and on the horizontal direction we have the repetitive readings for the same muscle.



Figură 1

Furthermore the distribution of data by class is as follows: 2909 (24.91%) data for class 0 (rock), 2902 (24.85%) data for class 1 (scissors), 2942 (25.2%) for class 2 (paper), 2921 (25.02%) for class 3 (ok). This can also be seen in Figure 2, and in addition we can say that the data is balanced.



Figură 2

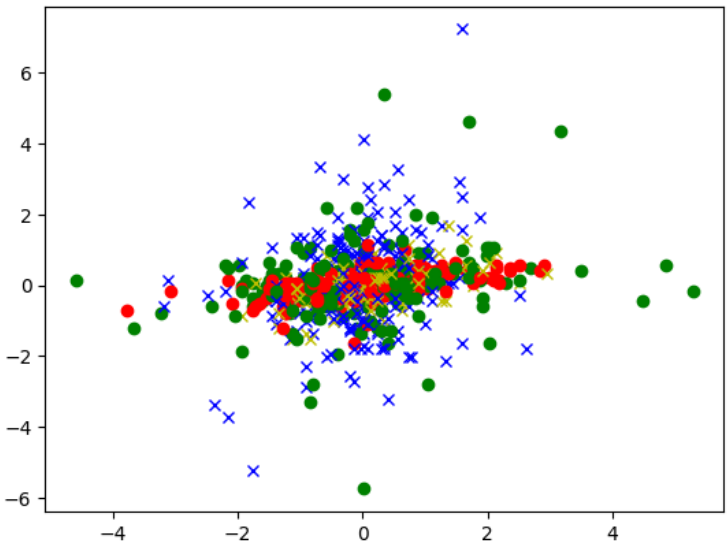
DATA NORMALIZATION

For data are applied a standardization that subtract the average and divides by the standard deviation. This is done independently for each feature, but also independent of the label, by a priori assuimng the fact that the data comes from the same class distribution (which does not contradict intuition because the data is balanced).

DESCRIPTION OF THE METHODS USED

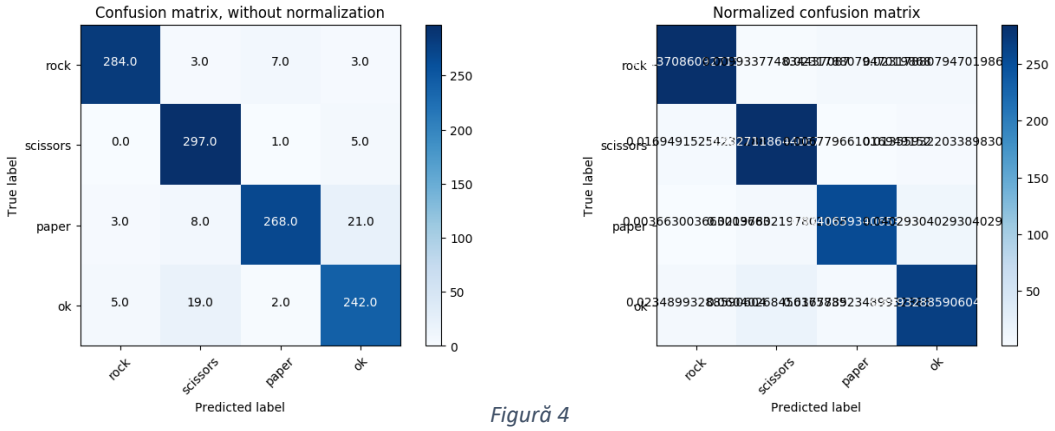
1. SVM

In Figure 3, 1000 random data are plotted in 2-D ie, only the first 2 features (muscle 1 and muscle 2 at first reading) were considered. It is easy to see that the data are not separable at all in 2-D. One solution would be to map the data in a larger dimensional space with the hope of a separation in that space. Well the SVM (Support Vector Machines) method does this, it tries to map the data in a space of a higher dimensionality and then try to separate the data in the new space. In addition, the SVM method is a binary classification, as we have 4 classes we need a trick to use SVM. This trick is to train 6 different classifiers (combinations of 2 taken as many classes, we have 4, so 6 classifiers). It is already implemented and is called "one-against-one".



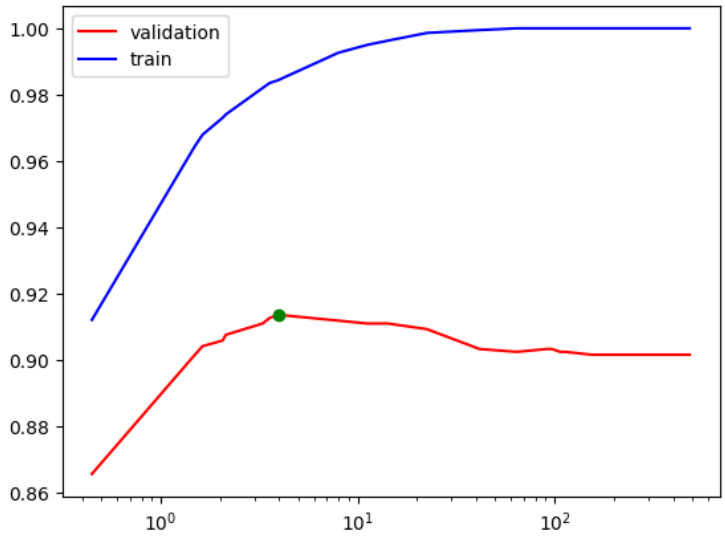
Figură 3

Then after a grid search by parameter C on the logarithmic space [0.99, 999] from which 25 equidistant points are chosen, a maximum accuracy of 93.15% is obtained for C = 4.633413251903491.



Figură 4

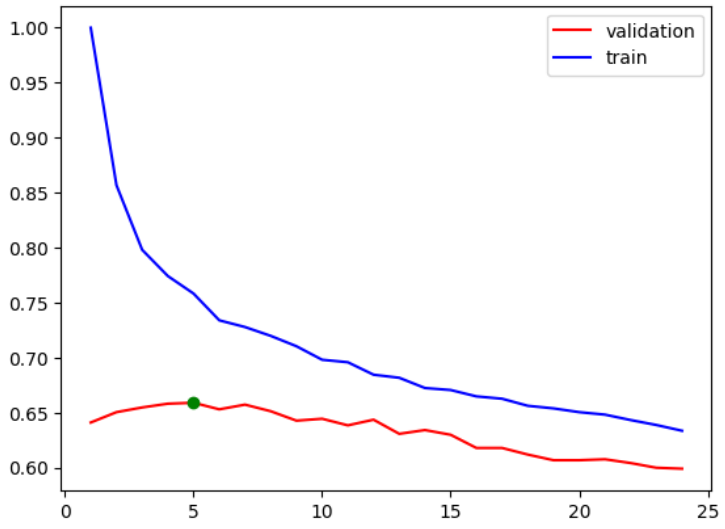
Then for a random search on the same logarithmic space [0.99, 999], from which 25 points are chosen randomly, a maximum accuracy is obtained on the set of 93.40% for C = 3.9732891501042005. This can also be seen in Figure 5. In Figure 4 we have the normal and normalized confusion matrices for the optimal SVM obtained after random search.



Figură 5

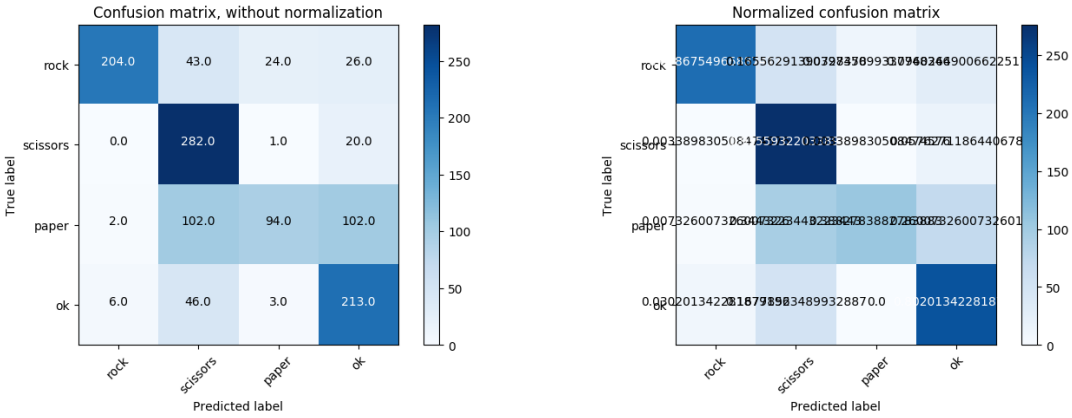
2. K-NN

The k-NN method consists of identifying the nearest k neighbors in the train set for an example in the test. We calculate these k neighbors in a 64-dimensional space more precisely R ^ 64. The metric used is Minkowski. Next we need to find optimal k for which the accuracy on the validation set is maximum.



Figură 6

After a grid search (random search does not make sense because we want natural numbers between 1 and a predefined maximum, we have taken 25) it is identified optimally from all possible up to a rank (at us 25). Identify k optimally as 5, with an accuracy on the test set 67.83%.



Figură 7

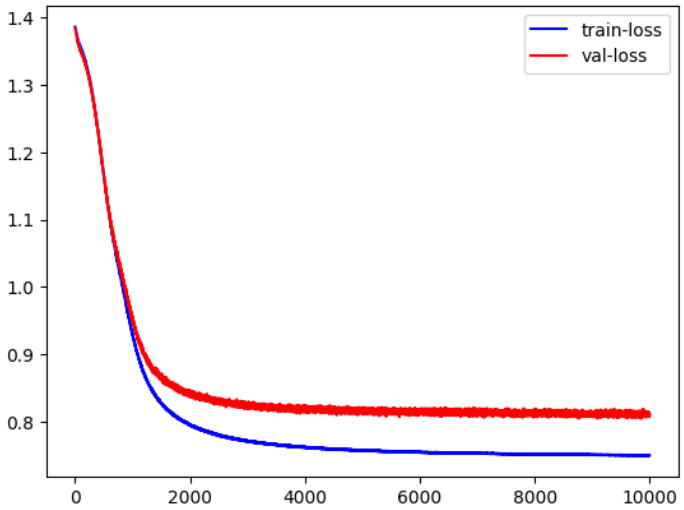
It can be seen in Figure 6 that the more we increase the number of neighbors, the less the accuracy decreases on validation but also on the train from 5 in there. In Figure 7 we have the confusion matrices for the best k-NN model, ie 5-NN.

3. NN

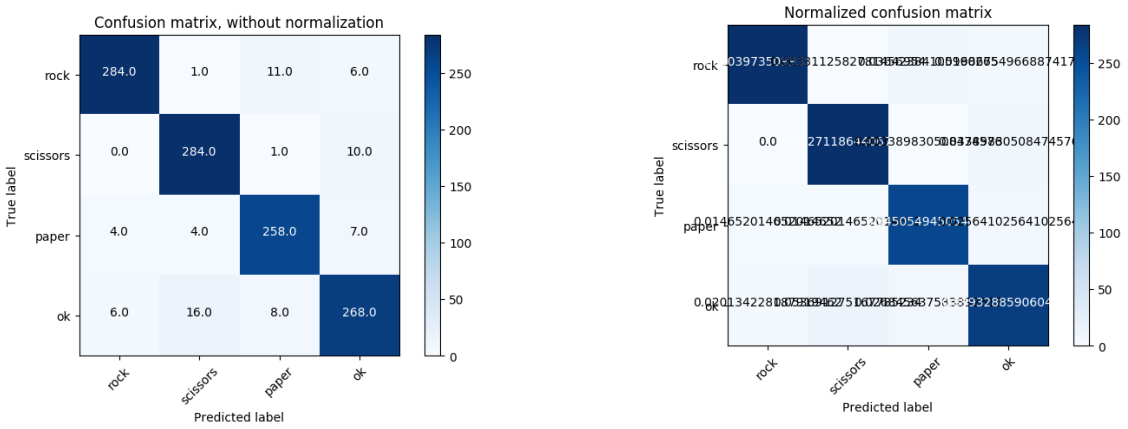
The architecture of the first network consists of several linear layers: the input layer of size 64 (the number of features), two hidden layers the first of size 512, the second layer of size 128 and the output layer of 4 perceivers. Each hidden layer has the ReLU activation function and the output layer has softmax activation (since we want a clear delimitation of the classes at the exit of the classifier). In addition to avoid overfitting after each hidden layer there is also the Dropout procedure (with custom probability p = 0.05) that ignores the data with the set probability. A more detailed description is shown below:

```
Net(
  (_layer1): Linear(in_features=64, out_features=512, bias=True)
  (_layer2): Linear(in_features=512, out_features=128, bias=True)
  (_layer3): Linear(in_features=128, out_features=4, bias=True)
  (_dropout1): Dropout(p=0.05, inplace=False)
  (_dropout2): Dropout(p=0.05, inplace=False)
)
```

For example, for a series of hyperparameters (learning rate = 0.01, and dropout probabilities = 0.05) and SGD optimizer (gradient descent), a performance of 94.09% is chosen after 10,000 times. In Figure 8 we have the evolution function of loss with red on validation and blue on the train set during the 10,000 times. I should mention that as a function of loss, crossentropy was used.



Figură 8



Figură 9

In addition, after a random search by the learning rate parameter for the SGD optimizer, we obtain that the best learning rate is 4.434704499077834e-05 and an accuracy of 95.29%. I mention that this search is done on a discrete logarithmic space from 〖10〗 ^ (- 5) to 10 out of which 25 potential learning rates are chosen randomly.

Further in Figure 10 we have the accuracy progress on the validation set according to the learning rate (the green dot indicates the optimal learning rate chosen) for which the accuracy on the validation set is maximum. In Figure 9 we have the normal and normalized confusion matrices of the optimal model (the one with the chosen learning rate).

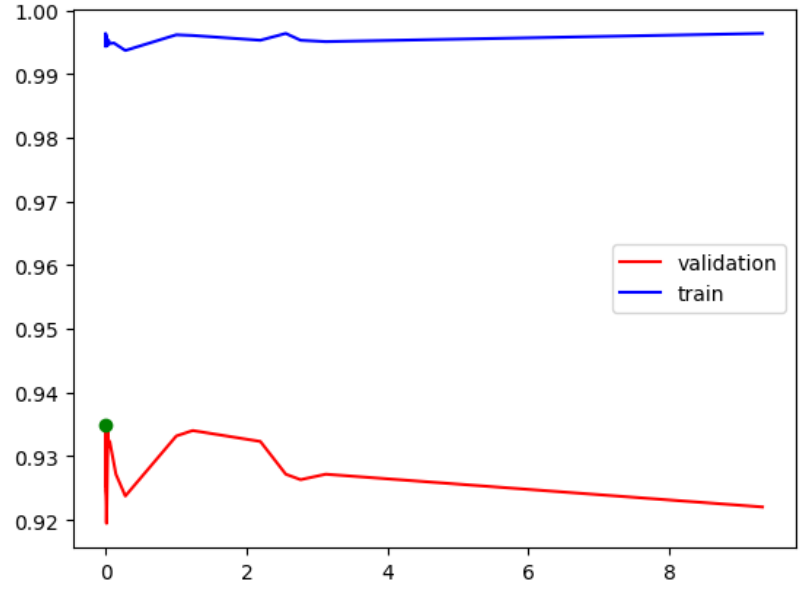


Figure 10

COMMENT OF RESULTS

As can be seen above, the best performance is achieved by the neural network with the architecture 64-512-128-4 and with the gradient descent algorithm learning_rate = 4.434704499077834e-05. I should mention that the current network was also trained with the optimization of Adagrad, RMSprop and Adam but the performances are lower than SGD. In addition it is clear that the network has generalization power proven by performance. Moreover, she is able to learn fact proven by almost 100% accuracy on the train, so she has learning capacity.

In addition, this performance is comparable to that achieved by SVM. I mention that the SVM works well in this case because it is known that this method is one that does not overfitting. Then to us the number of features is 64 much smaller than the number of examples which is

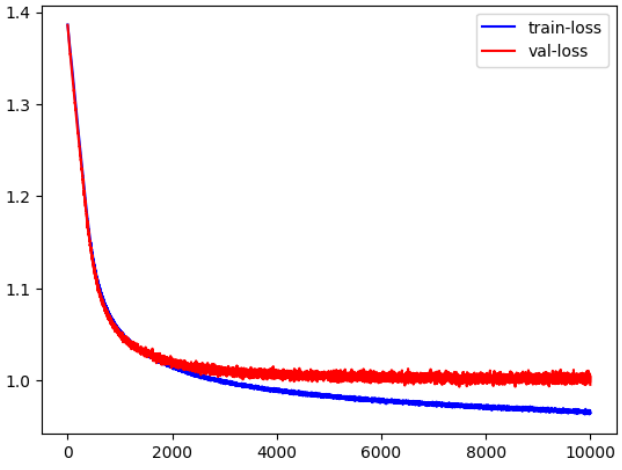
11674 and 64 is a great dimensionality, in which the SVM method is a good one. Another reason why the SVM works well for us is because we scaled the data (mean 0 and deviation 1). Moreover, as for us an optimal parameter for C is 3.97> 1, it turns out, among other things, that our data are not noisy (sensors that read muscles are accurate).

Compared to the k-NN method, the previous methods are clearly superior on this data-set because the data are not separable in the 64-dimensional space (we saw this in the SVM section that they were not separable in 2-D on the first two features). while SVM and NN work in dimensionality greater than the number of features. In addition it turns out that a neighborhood of 5 neighbors is the most significant, but it is not relevant to our problem.

SVM	k-NN	NN with SGD optimizer and architecture (64,512,128.4)
C = 3.9732	K = 5	Lr = 4.434704499077834e-05
93.40%	67.83%	**95.29%**

Table 1

Moreover, if we consider only the first reading of the 8 muscles, and all 8 obtain an accuracy with the same architecture of the neural network (8-512-128-4) of 73.11%. (Figure 11). For an SVM (C = 1) the accuracy of 67.12 is obtained%.



Figură 10

CONCLUSIONS

The present project managed to predict with 95.29% accuracy the signs of human hands (only 4, stone, scissors, paper and ok) giving 8 readings of 8 muscles of the hand. Several ML techniques are used, such as SVM with an accuracy of 2% lower than the best of the paper, k-NN with an accuracy of 67% and the neural networks which give the highest of the ones listed, namely 95.29 %. In addition, it shows that if we reduce the number of features by 87.5% (i.e. we divide the number of features to 8) we get a performance of at least 73.11%.