In the beginning data should be preprocessed, neural network accepts only real valued inputs and there are total 52 cards, so I transformed the provided categorical data into the 52-dimensional one-hot vector. First problem was to find appropriate hyperparameters such as network architecture, initial learning rate and regularization coefficient. For this task we exploited the GridSearch library from the scikit learn. I changed the depth of the network from 2 to 10 and the number of hidden units in each layer was varied from 50 to 200. The resulting optimal architecture I got based on our search was to use 5 layer network with 100 nodes in each layer. Neural network package of the scikit learn allows to set the learning rate as *adaptive* which means the learning rate keeps constant to the initial learning rate as long as training loss keeps decreasing. Each time two consecutive epochs fail to decrease training loss by at least some certain amount (set to 10-4 in our case), the current learning rate is divided by 5. The grid search values for initial learning rate was randomly chosen in range between 10-1 to 10-6, the optimal value was approximately 0.001, in the similar way we looked for the value of the *alpha* in L2 regularizer and found that the optimal value is 0.0001. The test accuracy was ~96%.

The data is highly imbalanced, the number of samples becomes extremely small as the hand value increases. For example, there are only 16 samples for the hand with value 9 and 32613 samples for the hand with value 0. To resolve this problem I performed the random oversampling over the training set, this gave us almost perfectly balanced data but with high number of repetitions for smaller classes. This step proved to be beneficial since initial imbalanced data gave only ~96% accuracy whereas with properly preprocessed training data we could achieve ~98% test accuracy.

To maximize the value of hand we changed every card with other cards excluding ones in our ‘hand’. So, we had 47\*5=235 combinations. Then I chose the one with highest hand value using np.argmax().