

Midterm 2

Assignment 3

Intelligent Systems for Pattern
Recognition

Master Degree in Computer
Science, AI Curriculum

A.Y. 2020/2021



Alessandro Ristori

RBM Code

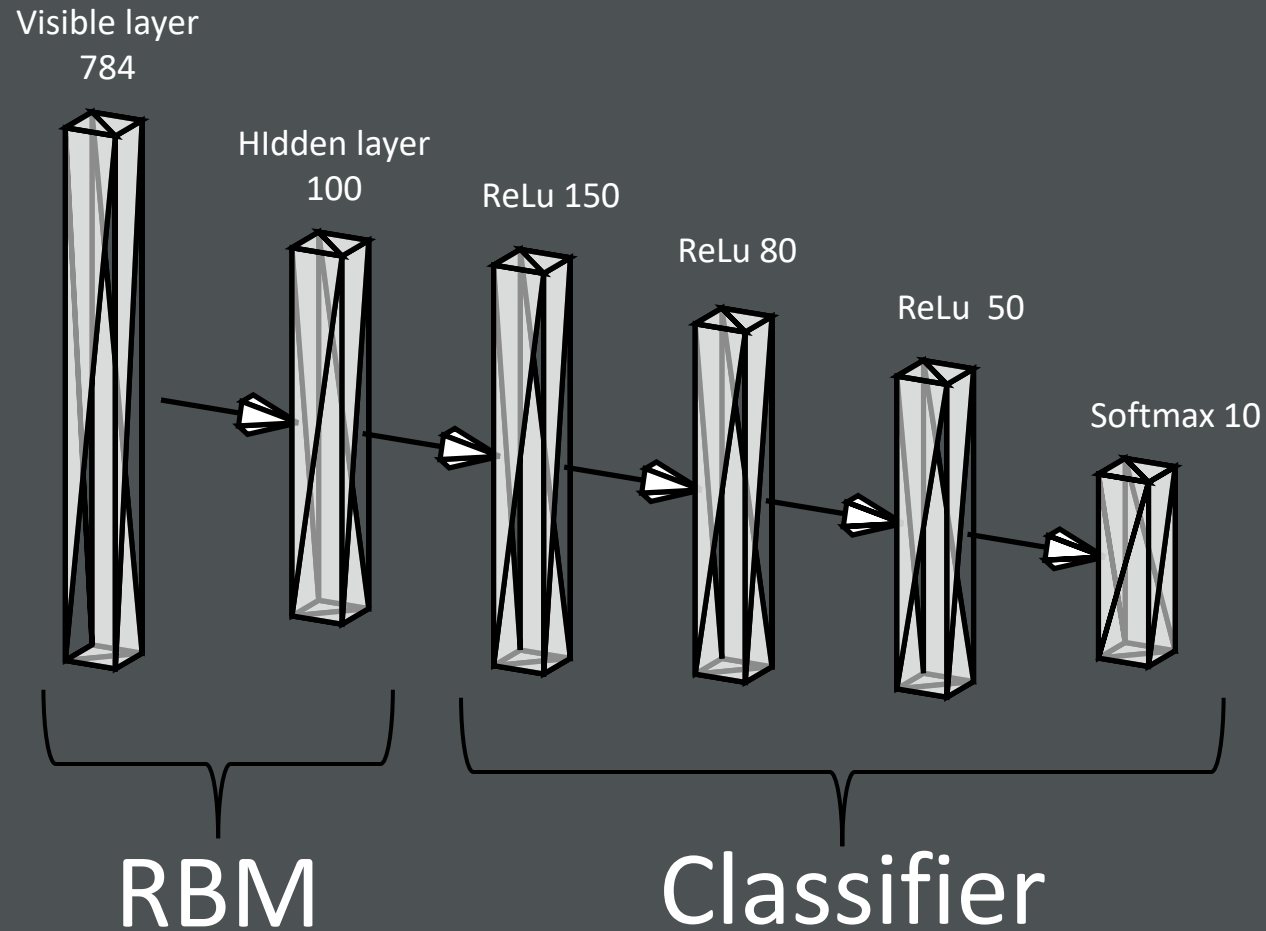
```
def __contrastive_divergence_step(self, values, step: str):
    bin_values = self.__stochastic_data(values)
    if step == "pos_data" or step == "neg_data":
        nets = np.dot(self.weights, bin_values) + self.hidden_bias
        nodes_p = self.__sigmoid(nets)
        if step == "pos_data":
            data = np.outer(nodes_p, values)
            return nodes_p, data
        else:
            data = np.outer(nodes_p, bin_values)
            return bin_values, nodes_p, data
    else:
        nets = np.dot(self.weights.T, bin_values) + self.visible_bias
        nodes_p = self.__sigmoid(nets)
        return nodes_p

def train(self, tr_set, eta, alpha, epochs: int, batch_size: int, save_weights: bool = False):
    tr_set_copy = np.copy(tr_set)
    d_w = np.zeros_like(self.weights)
    d_visible = np.zeros_like(self.visible_bias)
    d_hidden = np.zeros_like(self.hidden_bias)
    for i in range(epochs):
        np.random.shuffle(tr_set_copy)
        for pattern in tqdm.tqdm(tr_set, desc="Training {0}, Epoch {1}".format(self.name, i)):
            h_p, wake = self.__contrastive_divergence_step(pattern, "pos_data")
            v_p = self.__contrastive_divergence_step(h_p, "reconstruction")
            bin_reconstruction, neg_h_p, dream = self.__contrastive_divergence_step(v_p, "neg_data")

            d_w = alpha * d_w + (eta / batch_size) * (wake - dream)
            d_visible = alpha * d_visible + (eta / batch_size) * (np.sum(pattern) - np.sum(bin_reconstruction))
            d_hidden = alpha * d_hidden + (eta / batch_size) * (np.sum(h_p) - np.sum(neg_h_p))
            self.weights += d_w
            self.visible_bias += d_visible
            self.hidden_bias += d_hidden

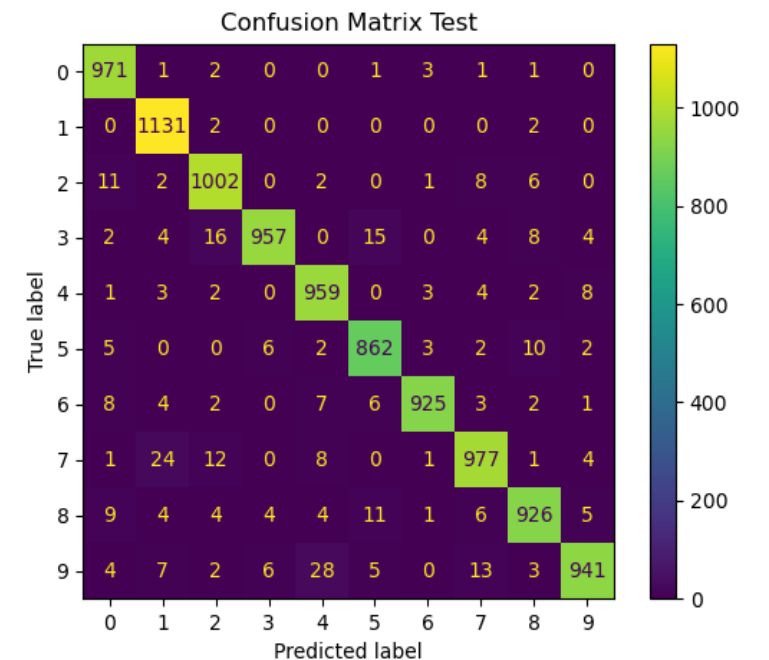
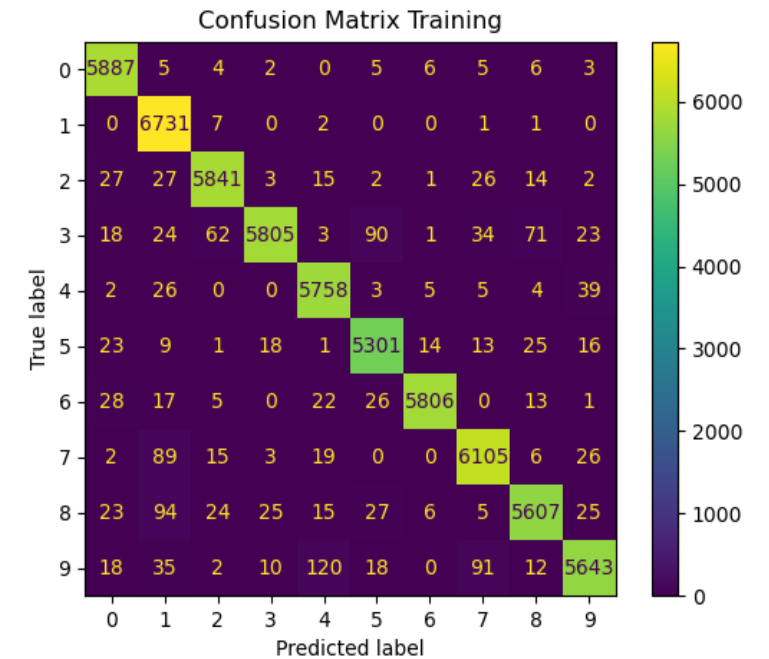
    if save_weights:
        self.__save_weights("weights_{0}.csv".format(self.name))
```

Model and confusion matrices

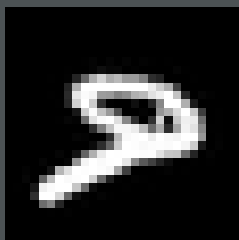


Training Error: 0.08392723649740219, Training accuracy: 0.9741166830062866

Test Error: 0.12995806336402893, Test accuracy: 0.9652000069618225



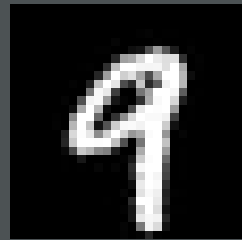
What if we rotate the digits by a few radians?



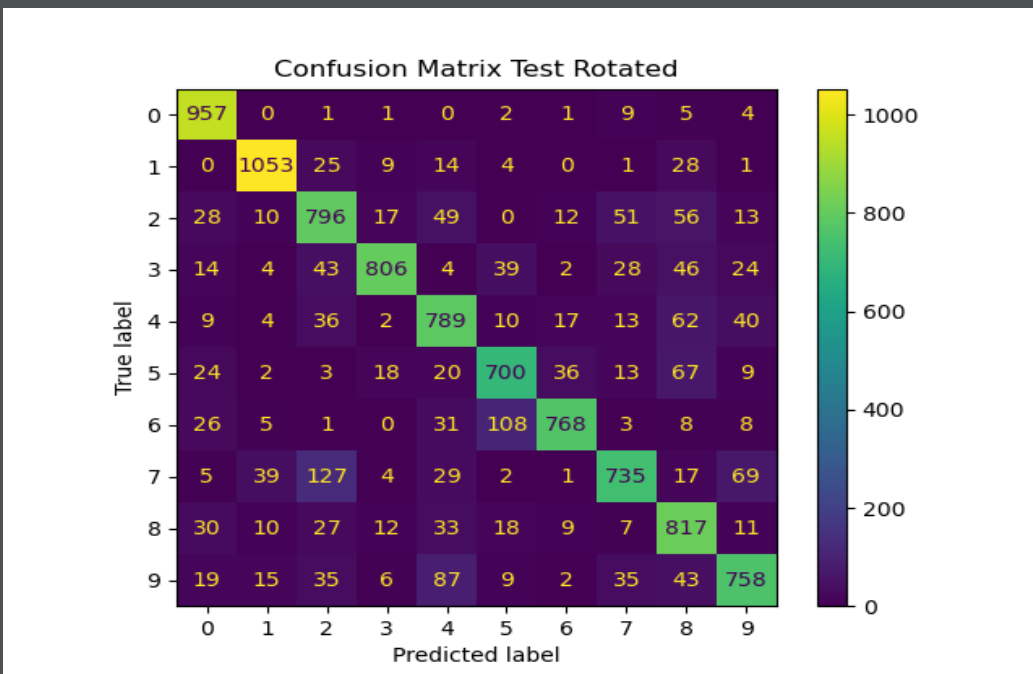
$-\pi/6$ rotation



Original digit

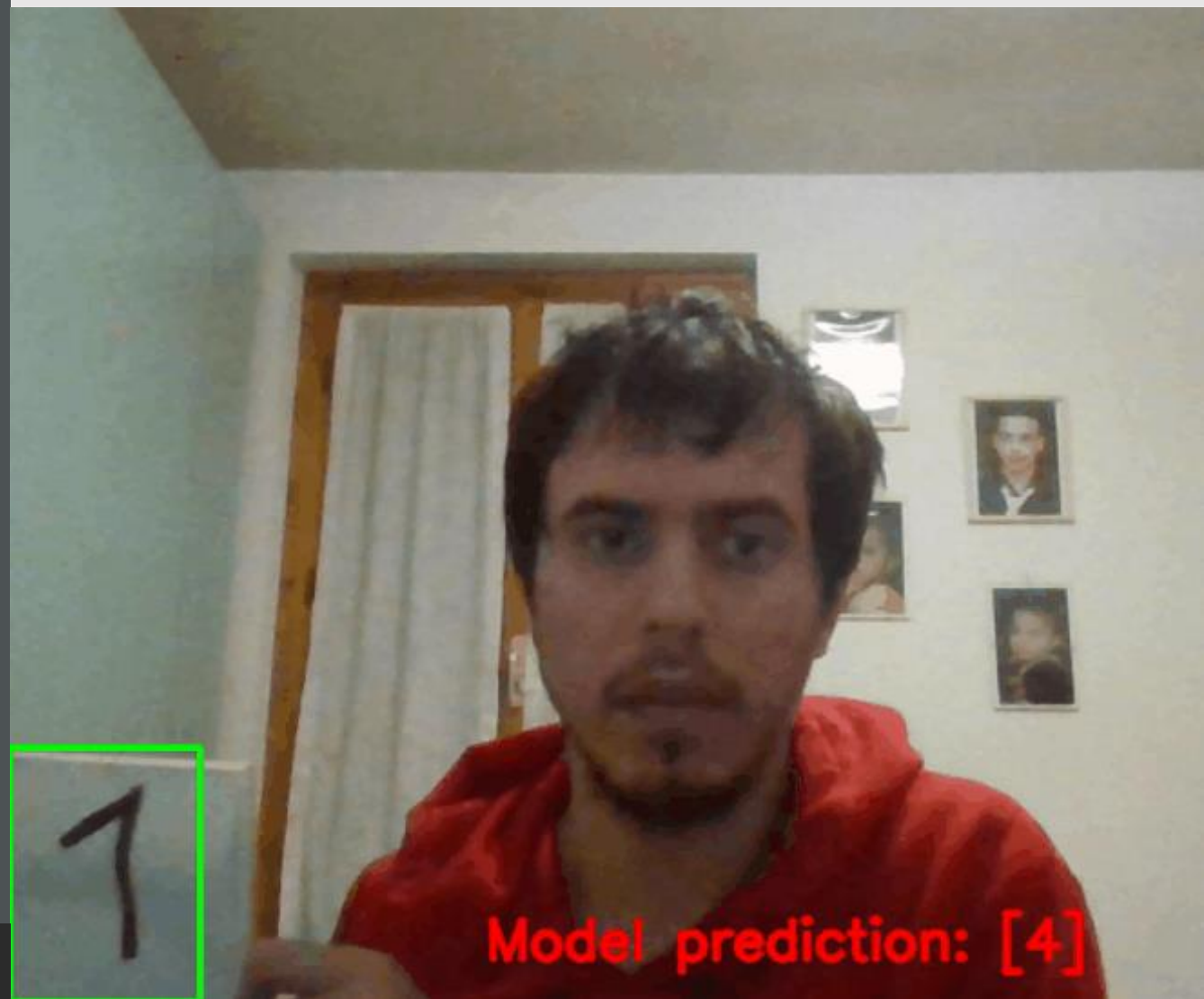


$\pi/6$ rotation



Test Rotated Error: 0.8769211173057556, Test Rotated accuracy: 0.8154000043869019

It seemed that the model was somewhat sensitive to the rotations (as expected), even the most insignificant ones; so i tried testing it in real time using opencv.



Final considerations



Stacking RBMs, before passing the encodings to the classifier, didn't have the desired effect, in some cases it made the accuracy worse;



CD-k was abandoned since it was too time expensive and it didn't produce any visible improvement on the final accuracy;



The random rotation were limited in the range $[-\pi/4, \pi/4]$, any more rotations would have made the test useless;



The real time recognition works only if the digits are written with a decent-size font, otherwise the network would receive a very noised pattern.



Even though i obtained nice results with a simple network, the same results can be achieved with fewer and smaller layers in the classifier.

Thank you for your attention