ROB311 - TP4
SVM DIGIT RECOGNITION

Group 2 : Iad Abdul Raouf and Madeleine Becker

Monday, Mai 2020 12th

In order to use SVM to classify the digits, we first uploaded the data into a numpy array, to be able to work with it. Then we used the library `sklearn.decomposition` to do a PCA, Principal Components Analysis, to reduce the dimensions of the dataset.

Then we used the `sklearn.svm` library to create the SVM classifier, fit it to the traning data, and predict the classes of the testing data.

You can find our code here : https://github.com/iad-ABDUL-RAOUF/TP4-ROB311.

We have different results according to how many components are left after the PCA.

Here are the overall detection accuracies :

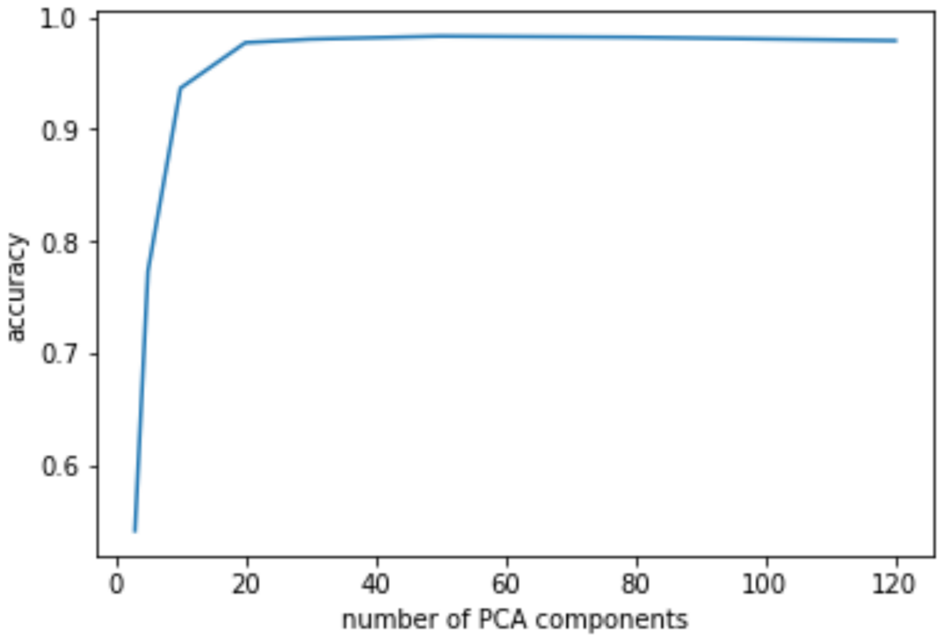


Figure 1: Overall accuracies by the number of PCA components

We can notice that the best overall detection accuracy is for 50 components (98,33%). Below that, there isn't enough information to properly fit the classifier. With 80 and 120, the accuracy isn't as good (98,23% and 97,23%), because the classifier is probably overfitted.

We also measured the time needed to process the PCA on the data, fit the classifier and use it to predict the classes. Surprisingly, for very low number of components, there is more time needed. It may be because the algorithm tries to have better results.

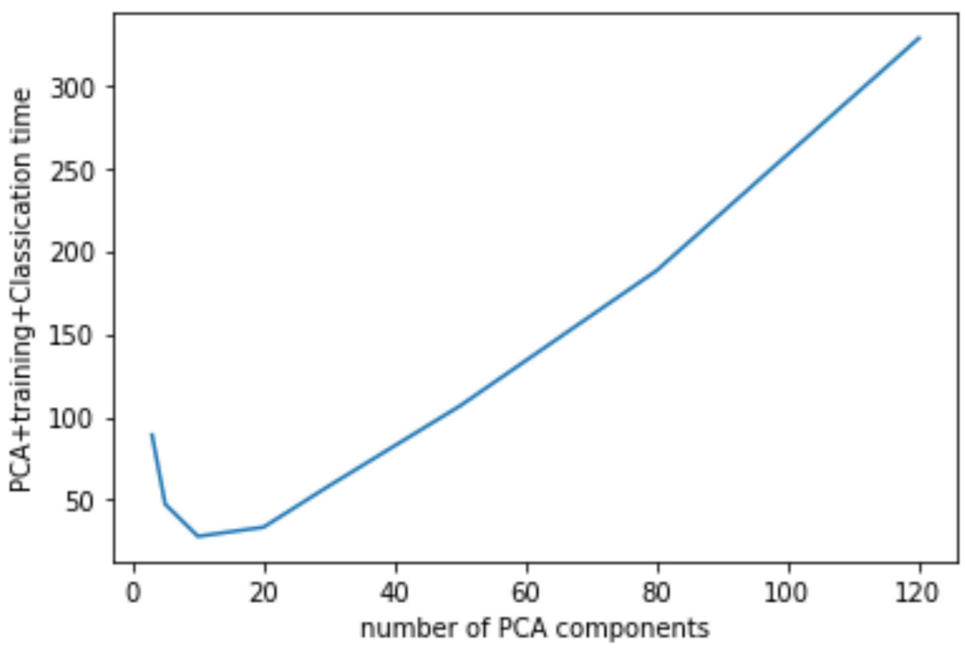
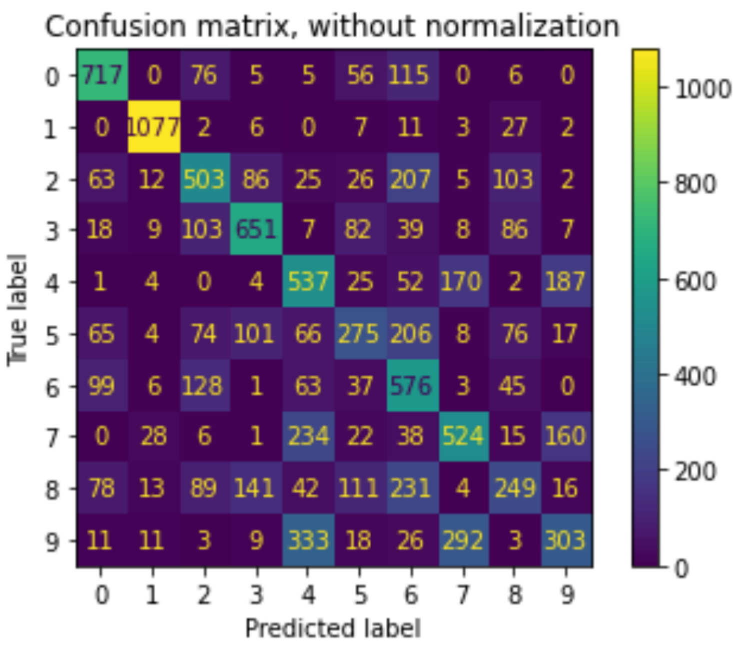


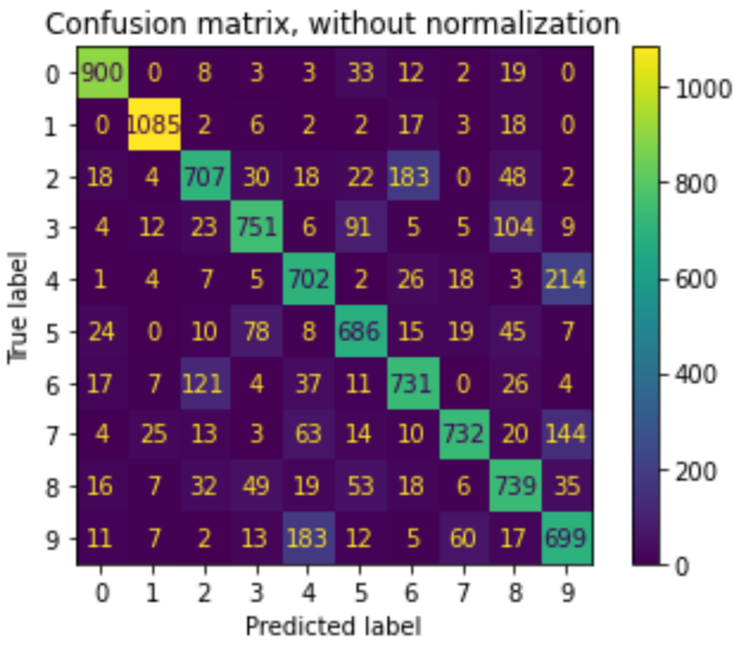
Figure 2: Time needed

Below are all the confusion matrix for each number of components we tried.



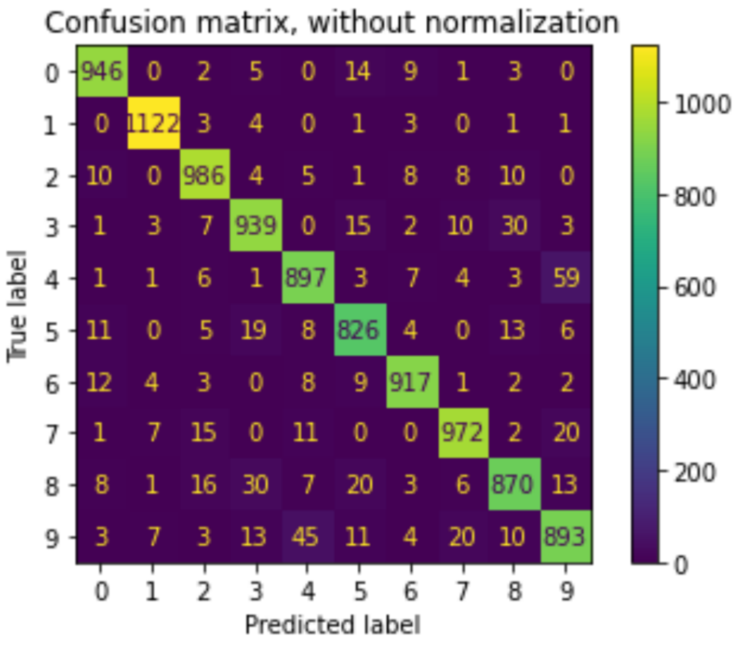
	0	1	2	3	4	5	6	7	8	9
0	717	0	76	5	5	56	115	0	6	0
1	0	1077	2	6	0	7	11	3	27	2
2	63	12	503	86	25	26	207	5	103	2
3	18	9	103	651	7	82	39	8	86	7
4	1	4	0	4	537	25	52	170	2	187
5	65	4	74	101	66	275	206	8	76	17
6	99	6	128	1	63	37	576	3	45	0
7	0	28	6	1	234	22	38	524	15	160
8	78	13	89	141	42	111	231	4	249	16
9	11	11	3	9	333	18	26	292	3	303

Figure 3: Confusion matrix for 3 components



True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	900	0	8	3	3	33	12	2	19	0
1	0	1085	2	6	2	2	17	3	18	0
2	18	4	707	30	18	22	183	0	48	2
3	4	12	23	751	6	91	5	5	104	9
4	1	4	7	5	702	2	26	18	3	214
5	24	0	10	78	8	686	15	19	45	7
6	17	7	121	4	37	11	731	0	26	4
7	4	25	13	3	63	14	10	732	20	144
8	16	7	32	49	19	53	18	6	739	35
9	11	7	2	13	183	12	5	60	17	699

Figure 4: Confusion matrix for 5 components



True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	946	0	2	5	0	14	9	1	3	0
1	0	1122	3	4	0	1	3	0	1	1
2	10	0	986	4	5	1	8	8	10	0
3	1	3	7	939	0	15	2	10	30	3
4	1	1	6	1	897	3	7	4	3	59
5	11	0	5	19	8	826	4	0	13	6
6	12	4	3	0	8	9	917	1	2	2
7	1	7	15	0	11	0	0	972	2	20
8	8	1	16	30	7	20	3	6	870	13
9	3	7	3	13	45	11	4	20	10	893

Figure 5: Confusion matrix for 10 components

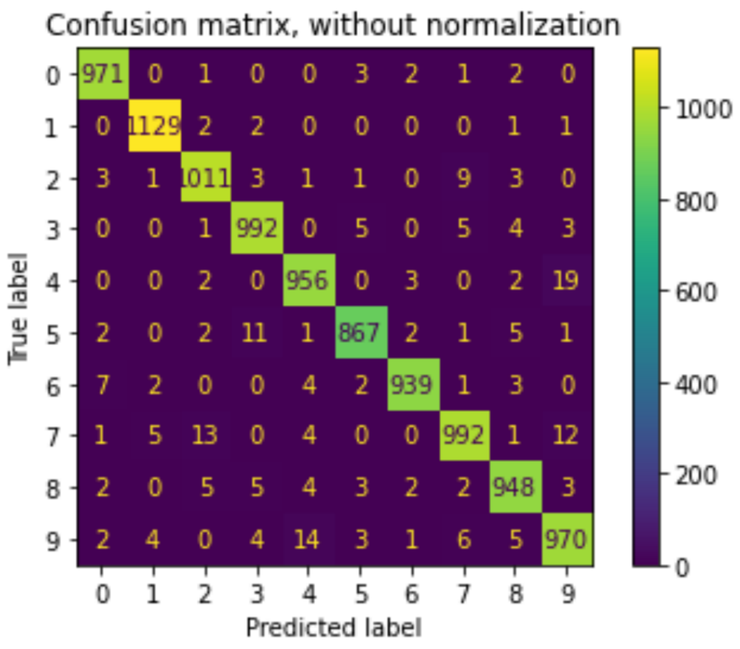


Figure 6: Confusion matrix for 20 components

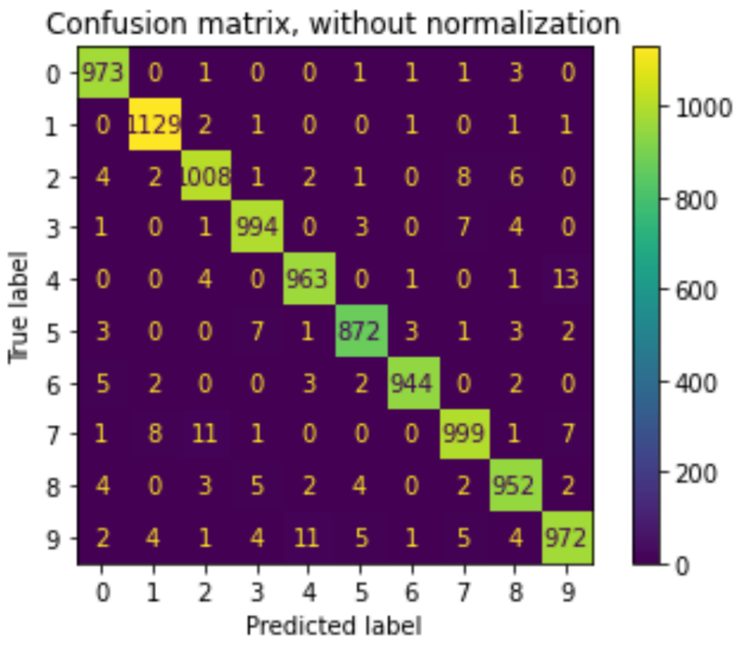
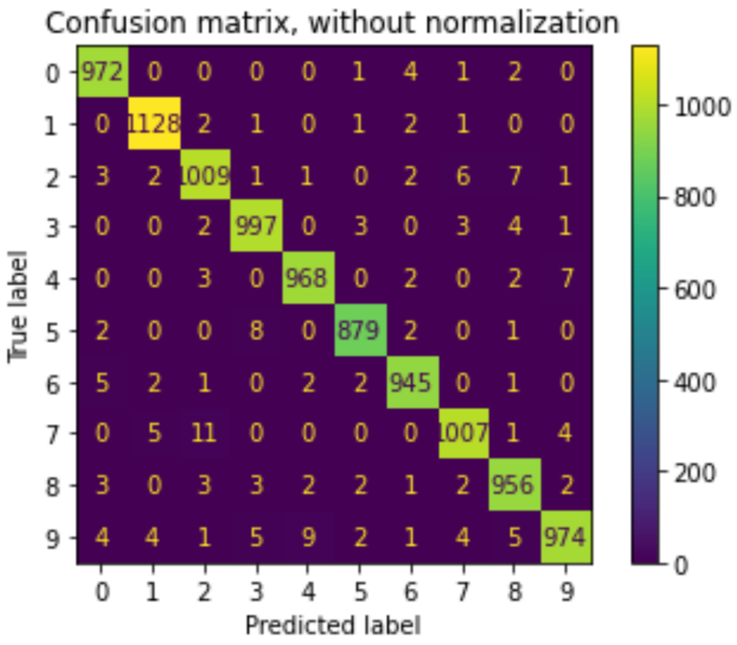
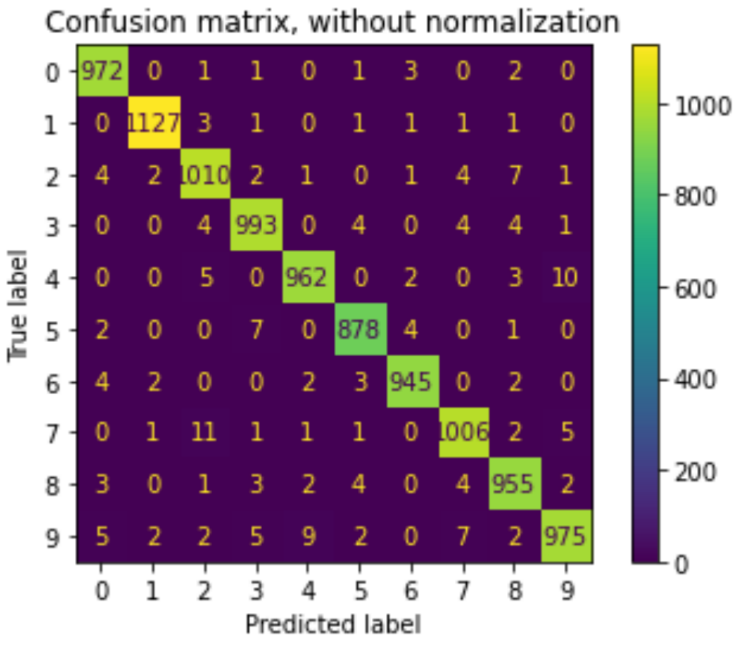


Figure 7: Confusion matrix for 30 components



True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	972	0	0	0	0	1	4	1	2	0
1	0	1128	2	1	0	1	2	1	0	0
2	3	2	1009	1	1	0	2	6	7	1
3	0	0	2	997	0	3	0	3	4	1
4	0	0	3	0	968	0	2	0	2	7
5	2	0	0	8	0	879	2	0	1	0
6	5	2	1	0	2	2	945	0	1	0
7	0	5	11	0	0	0	0	1007	1	4
8	3	0	3	3	2	2	1	2	956	2
9	4	4	1	5	9	2	1	4	5	974

Figure 8: Confusion matrix for 50 components



True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	972	0	1	1	0	1	3	0	2	0
1	0	1127	3	1	0	1	1	1	1	0
2	4	2	1010	2	1	0	1	4	7	1
3	0	0	4	993	0	4	0	4	4	1
4	0	0	5	0	962	0	2	0	3	10
5	2	0	0	7	0	878	4	0	1	0
6	4	2	0	0	2	3	945	0	2	0
7	0	1	11	1	1	1	0	1006	2	5
8	3	0	1	3	2	4	0	4	955	2
9	5	2	2	5	9	2	0	7	2	975

Figure 9: Confusion matrix for 80 components

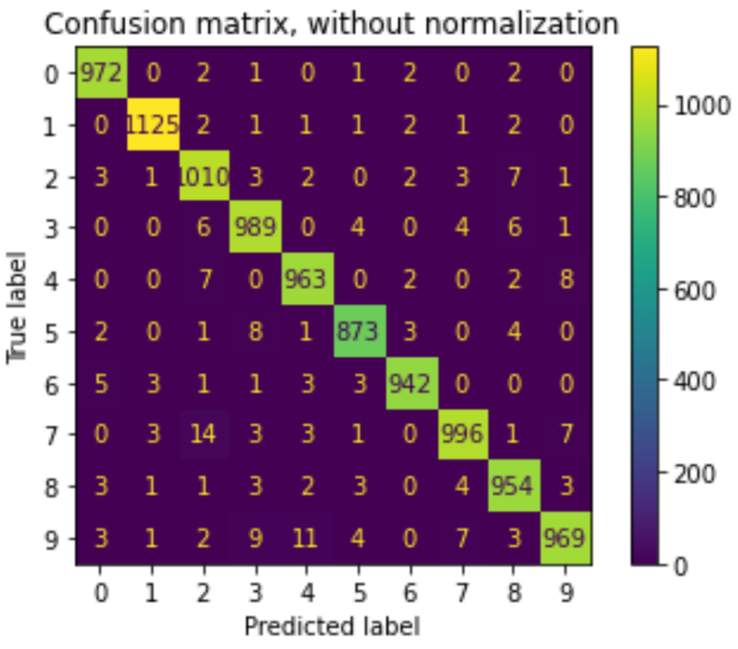


Figure 10: Confusion matrix for 120 components