Istanbul Technical University- Spring 2017
BLG527E Machine Learning
Homework 3

Omid Abdollahi Aghdam
student number: 504151520
abdollahi15@itu.edu.tr
abdollahi.omid@gmail.com

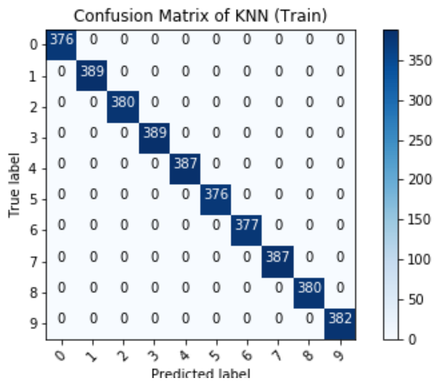
April 26, 2017

Note: Running hw3.py file will output the result on terminal (linux, MacOsx) or command prompt (Windows). Source codes lines are referenced in report. In addition to hw3.py, hw3.ipynb is also provided.

Q1) Spliting optdigits.tra: Training set (optdigits.tra) is randomly divided into 90% train and 10% validation sets (line 34).

knn hyper-parameters tuning: Best k is selected based on validation accuracy, and corresponding metric will be printed on terminal (lines 94-121).

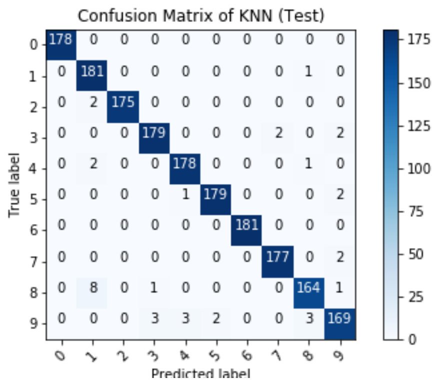
Training with best k on the whole training set: We train on whole data and report confusion matrix of train set (lines 125-134). It is shown in Figure 1. This part of source code will output train time and confusion matrix on terminal.



	0	1	2	3	4	5	6	7	8	9
0	376	0	0	0	0	0	0	0	0	0
1	0	389	0	0	0	0	0	0	0	0
2	0	0	380	0	0	0	0	0	0	0
3	0	0	0	389	0	0	0	0	0	0
4	0	0	0	0	387	0	0	0	0	0
5	0	0	0	0	0	376	0	0	0	0
6	0	0	0	0	0	0	377	0	0	0
7	0	0	0	0	0	0	0	387	0	0
8	0	0	0	0	0	0	0	0	380	0
9	0	0	0	0	0	0	0	0	0	382

Figure 1: Confusion Matrix of Train set (knn).

Evaluating knn model on test set: In lines 138-147, we predict test labels and report the metrics. Test set confusion matrix is shown in Figure 2.



	0	1	2	3	4	5	6	7	8	9
0	178	0	0	0	0	0	0	0	0	0
1	0	181	0	0	0	0	0	0	1	0
2	0	2	175	0	0	0	0	0	0	0
3	0	0	0	179	0	0	0	2	0	2
4	0	2	0	0	178	0	0	0	1	0
5	0	0	0	0	1	179	0	0	0	2
6	0	0	0	0	0	0	181	0	0	0
7	0	0	0	0	0	0	0	177	0	2
8	0	8	0	1	0	0	0	0	164	1
9	0	0	0	3	3	2	0	0	3	169

Figure 2: Confusion Matrix of Test set (knn).

Decision tree hyper-parameters tuning: We selected the maximum depth of decision tree based on validation accuracy, and corresponding metric will be printed on terminal (lines 159-187).

Training with best max_depth on the whole training set: We train on whole data and report confusion matrix of train set (lines 190-201). It is shown in Figure 3. This part of source code will output train time and confusion matrix on terminal.

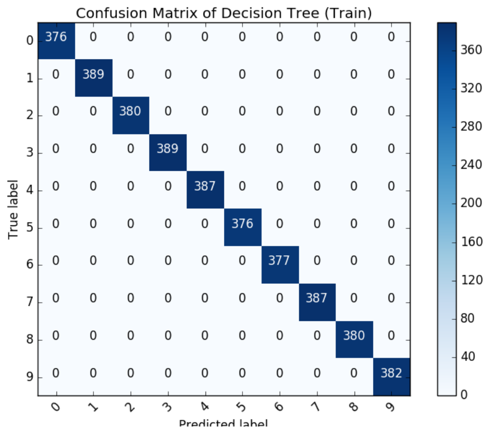


Figure 3: Confusion Matrix of Train set (Decision tree).

Evaluating decision tree model on test set: In lines 204-213, we predict test labels and report the metrics. Test set confusion matrix is shown in Figure 4.

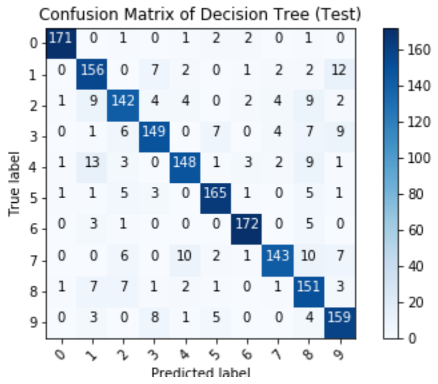
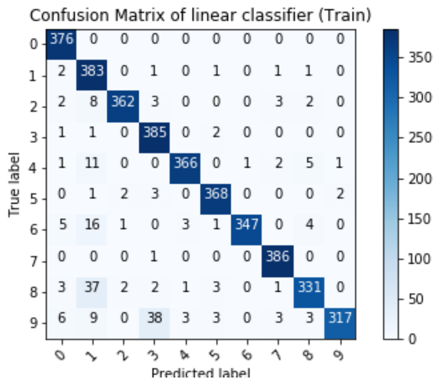


Figure 4: Confusion Matrix of Test set (Decision tree).

Linear discrimination hyper-parameters tuning: We selected the regularization strength of linear classifier based on validation accuracy, and corresponding metric will be printed on terminal (lines 224-252).

Training with best alpha on the whole training set: We train on whole data and report confusion matrix of train set (lines 255-265). It is shown in Figure 5. This part of source code will output train time and confusion matrix on terminal.

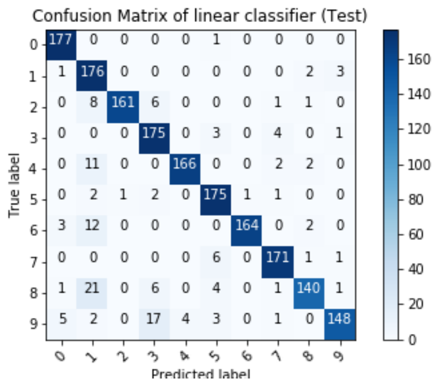
Evaluating linear discrimination model on test set: In lines 268-277, we predict test



	0	1	2	3	4	5	6	7	8	9
0	376	0	0	0	0	0	0	0	0	0
1	2	383	0	1	0	1	0	1	1	0
2	2	8	362	3	0	0	0	3	2	0
3	1	1	0	385	0	2	0	0	0	0
4	1	11	0	0	366	0	1	2	5	1
5	0	1	2	3	0	368	0	0	0	2
6	5	16	1	0	3	1	347	0	4	0
7	0	0	0	1	0	0	0	386	0	0
8	3	37	2	2	1	3	0	1	331	0
9	6	9	0	38	3	3	0	3	3	317

Figure 5: Confusion Matrix of Train set (linear classifier).

labels and report the metrics. Test set confusion matrix is shown in Figure 6.

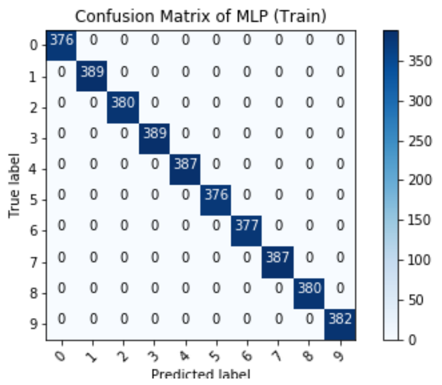


	0	1	2	3	4	5	6	7	8	9
0	177	0	0	0	0	1	0	0	0	0
1	1	176	0	0	0	0	0	0	2	3
2	0	8	161	6	0	0	0	1	1	0
3	0	0	0	175	0	3	0	4	0	1
4	0	11	0	0	166	0	0	2	2	0
5	0	2	1	2	0	175	1	1	0	0
6	3	12	0	0	0	0	164	0	2	0
7	0	0	0	0	0	6	0	171	1	1
8	1	21	0	6	0	4	0	1	140	1
9	5	2	0	17	4	3	0	1	0	148

Figure 6: Confusion Matrix of Test set (linear classifier).

MLP hyper-parameters tuning: We selected the best parameters of neural network classifier (number of hidden layers, layer sizes, and regularization strength) based on validation accuracy, and corresponding metrics will be printed on terminal (lines 290-320).

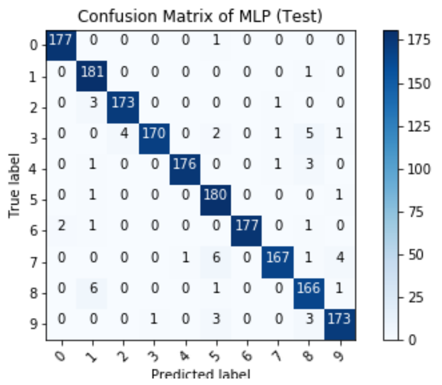
Training with best parameters on the whole training set: We train on whole data and report confusion matrix of train set (lines 323-335). It is shown in Figure 7. This part of source code will output train time and confusion matrix on terminal.
Evaluating neural network model on test set: In lines 338-349, we predict test labels



	0	1	2	3	4	5	6	7	8	9
0	376	0	0	0	0	0	0	0	0	0
1	0	389	0	0	0	0	0	0	0	0
2	0	0	380	0	0	0	0	0	0	0
3	0	0	0	389	0	0	0	0	0	0
4	0	0	0	0	387	0	0	0	0	0
5	0	0	0	0	0	376	0	0	0	0
6	0	0	0	0	0	0	377	0	0	0
7	0	0	0	0	0	0	0	387	0	0
8	0	0	0	0	0	0	0	0	380	0
9	0	0	0	0	0	0	0	0	0	382

Figure 7: Confusion Matrix of Train set (MLP).

and report the metrics. Test set confusion matrix is shown in Figure 8.



	0	1	2	3	4	5	6	7	8	9
0	177	0	0	0	0	1	0	0	0	0
1	0	181	0	0	0	0	0	0	1	0
2	0	3	173	0	0	0	0	1	0	0
3	0	0	4	170	0	2	0	1	5	1
4	0	1	0	0	176	0	0	1	3	0
5	0	1	0	0	0	180	0	0	0	1
6	2	1	0	0	0	0	177	0	1	0
7	0	0	0	0	1	6	0	167	1	4
8	0	6	0	0	0	1	0	0	166	1
9	0	0	0	1	0	3	0	0	3	173

Figure 8: Confusion Matrix of Test set (MLP).

Q2) In all models except linear discrimination, accuracy per class on train set are 100%. For knn the best k is equal 1, so its accuracy on train set must be 100%, regarding decision tree there might be over-fitting as the validation and train accuracy differ by 8%. Thus, as a result test accuracy is even worse, 85.5%. we could reduce the effect of over-fitting by cross validation. Furthermore, when we look at MLP accuracies, 100%, 99.4%, 96.8%, train, validation, and test respectively, we can conclude that there is not over-fitting. In addition, linear discrimination accuracies shows that it is not capable of achieving state-of-the-art results. Looking at all confusion matrix, we observed that class 8 is miss-classified as class 1 the most.
In terms of training time, the fastest classifier is knn, because it does not have any operation in training time and simply store the data, and the slowest model is neural network, because it has many matrix multiplication during forward pass and back-propagation. In test time, decision tree is the fastest model, and slowest model is knn because it compare each test set with all instances in training set.

Q3) Source code (382-409) identifies misclassified instances for each model and prints the results on Terminal. We are supposed to eliminate 10% of training instances, however the percentage of noisy data is less than 10% (we didn't shuffle the data), thus we remove all of those data from training set that are mislabeled (about 5%). Finally, we randomly split new reduced training data into 90% train, and 10% validation and repeat all the process which are asked in **Q1**. We reference the related lines in source code and provide confusion matrix figures in the rest of this report.

knn hyper-parameters tuning with train data after removal: Best k is selected based on validation accuracy, and corresponding metric will be printed on terminal (lines 424-452).

Training with best k on the whole training set after removal: We train on whole data and report confusion matrix of train set (lines 456-465). It is shown in Figure 9. This part of source code will output train time and confusion matrix on terminal.

Evaluating knn model on test set: In lines 469-478, we predict test labels and report the metrics. Test set confusion matrix is shown in Figure 10.

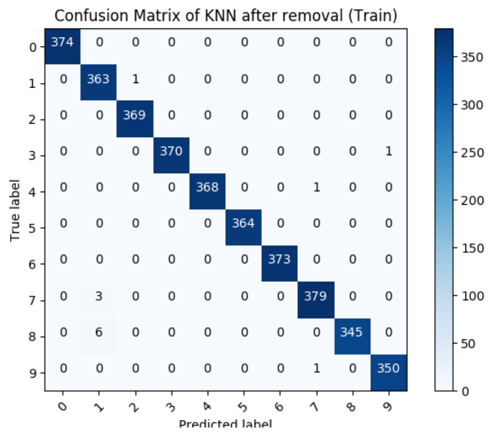


Figure 9: Confusion Matrix of Train set after removal (knn).

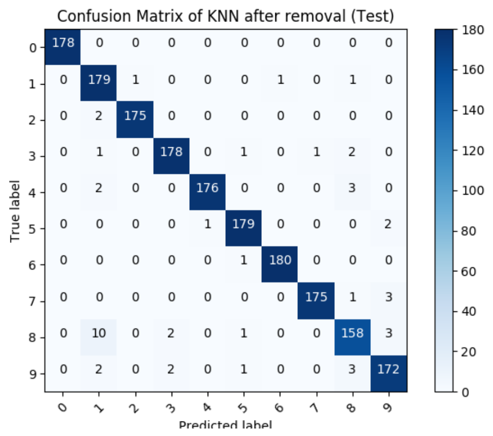


Figure 10: Confusion Matrix of Test set after removal(knn).

Decision tree hyper-parameters tuning after removal: We selected the maximum depth of decision tree based on validation accuracy, and corresponding metric will be printed on terminal (lines 491-520).

Training with best max_depth on the whole training set after removal: We train on whole data and report confusion matrix of train set (lines 523-534). It is shown in Figure 11. This part of source code will output train time and confusion matrix on terminal.

Evaluating decision tree model on test set after removal: In lines 537-546, we predict test labels and report the metrics. Test set confusion matrix is shown in Figure 12.

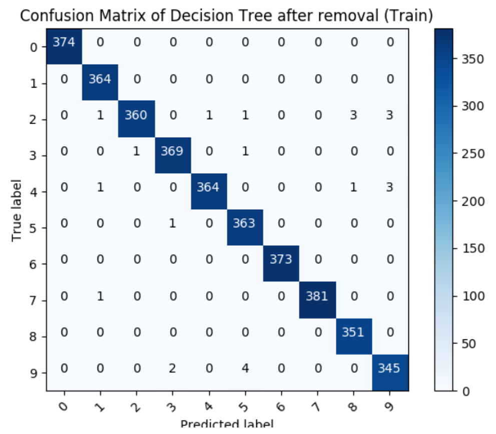


Figure 11: Confusion Matrix of Train set after removal (Decision tree).

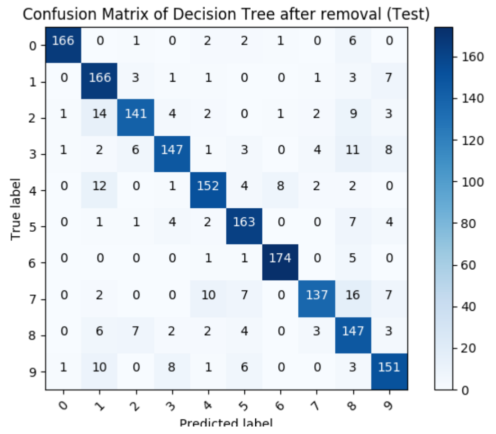


Figure 12: Confusion Matrix of Test set after removal (Decision tree).

Linear discrimination hyper-parameters tuning after removal: We selected the regularization strength of linear classifier based on validation accuracy, and corresponding metric will be printed on terminal (lines 558-586).

Training with best alpha on the whole training set after removal: We train on whole data and report confusion matrix of train set (lines 589-599). It is shown in Figure 13. This part of source code will output train time and confusion matrix on terminal.
Evaluating linear discrimination model on test set after removal: In lines 602-611, we predict test labels and report the metrics. Test set confusion matrix is shown in Figure 14.

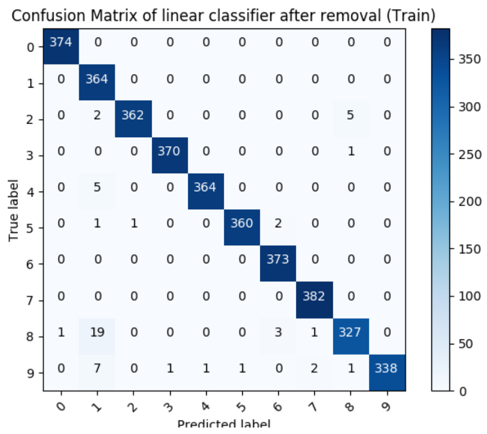


Figure 13: Confusion Matrix of Train set after removal (linear classifier).

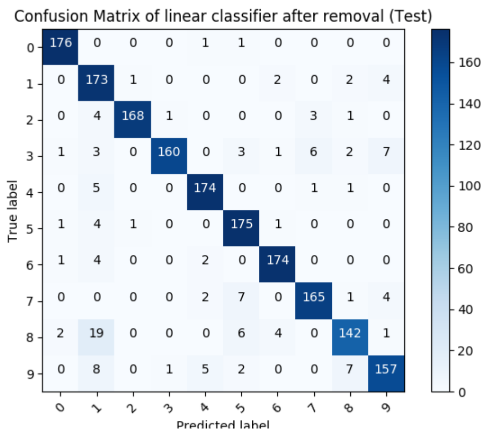


Figure 14: Confusion Matrix of Test set after removal (linear classifier).

MLP hyper-parameters tuning after removal: We selected the best parameters of neural network classifier (number of hidden layers, layer sizes, and regularization strength) based on validation accuracy, and corresponding metrics will be printed on terminal (lines 624-654).

Training with best parameters on the whole training set after removal: We train on whole data and report confusion matrix of train set (lines 657-69). It is shown in Figure 15. This part of source code will output train time and confusion matrix on terminal.
Evaluating neural network model on test set after removal: In lines 672-683, we predict test labels and report the metrics. Test set confusion matrix is shown in Figure 16.

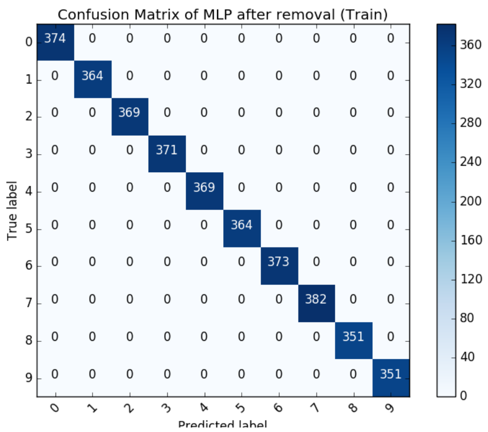


Figure 15: Confusion Matrix of Train set after removal (MLP).

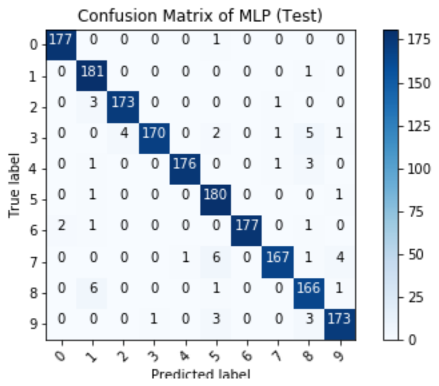
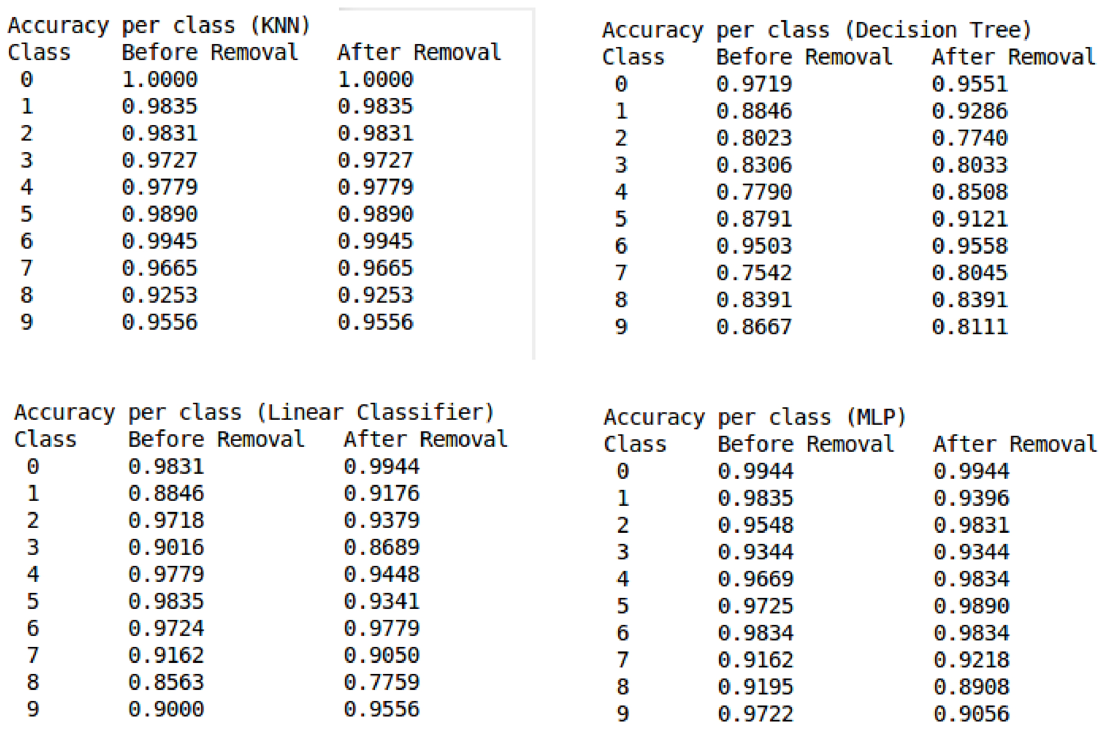


Figure 16: Confusion Matrix of Test set after removal (MLP).

Finally we compare the accuracy per classes of test data before and after removal for all models (lines 692-711). As it is shown in Figure 17, in our experiments, some classes accuracies get better and some get worse. Considering overall accuracy of models, decision tree model get better, and others decreased. However, performance of classifiers on test data set only changed less than 1%.



```
Accuracy per class (KNN)
Class     Before Removal    After Removal
 0        1.0000            1.0000
 1        0.9835            0.9835
 2        0.9831            0.9831
 3        0.9727            0.9727
 4        0.9779            0.9779
 5        0.9890            0.9890
 6        0.9945            0.9945
 7        0.9665            0.9665
 8        0.9253            0.9253
 9        0.9556            0.9556
```

```
Accuracy per class (Decision Tree)
Class     Before Removal    After Removal
 0        0.9719            0.9551
 1        0.8846            0.9286
 2        0.8023            0.7740
 3        0.8306            0.8033
 4        0.7790            0.8508
 5        0.8791            0.9121
 6        0.9503            0.9558
 7        0.7542            0.8045
 8        0.8391            0.8391
 9        0.8667            0.8111
```

```
Accuracy per class (Linear Classifier)
Class     Before Removal    After Removal
 0        0.9831            0.9944
 1        0.8846            0.9176
 2        0.9718            0.9379
 3        0.9016            0.8689
 4        0.9779            0.9448
 5        0.9835            0.9341
 6        0.9724            0.9779
 7        0.9162            0.9050
 8        0.8563            0.7759
 9        0.9000            0.9556
```

```
Accuracy per class (MLP)
Class     Before Removal    After Removal
 0        0.9944            0.9944
 1        0.9835            0.9396
 2        0.9548            0.9831
 3        0.9344            0.9344
 4        0.9669            0.9834
 5        0.9725            0.9890
 6        0.9834            0.9834
 7        0.9162            0.9218
 8        0.9195            0.8908
 9        0.9722            0.9056
```

Figure 17: Test accuracy of models per class, before and after removal.