MACHINE LEARNING LAB 1

SATHISH KUMAR SUBRAMANI
ROBOTICS ENGINEERING
UNIGE
Email: sathishmerwe@gmail.com

***Abstract*—In this assignment the Naive Bayes Classifier is applied to the data sets. And the data in the same data set is altered to check the smoothing effect. And the accuracy of the model is tested.**

I. INTRODUCTION

Naive Bayes classifiers is a simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features. The all input variables are assumed independence in Naive Bayes classifier.

The naive classifier equation can be derived as,

$$P(C_k|X) = \frac{P(C_k) \quad p(X|C_k)}{p(X)} \tag{1}$$

where X=(x$_1, ..., x_n$)

$$P(C_k|X) \quad \alpha \quad P(C_k) \quad p(X|C_k) \tag{2}$$

The function is created for this classifier and tested with the given weather data set. And their accuracy is calculated. Then the data set is altered to check the accuracy. All the functions are created in MATLAB environment.

II. TASK 1 - DATA PROCESSING

In this task the data set is prepared to run in the MATLAB. The data set has 14 instances with 4 features and 2 output classes. The data set is converted into the numerical format from text format in the text editor. Then they are used in the MATLAB.

III. TASK 2 - BUILD A NAIVE BAYES CLASSIFIER

In the initial stage of the MATLAB coding the given data set is processed. First all the values in the data set are not less than one were checked. Then the number of instances and number of features are stored in the table row wise and column wise respectively. Then the levels of each features are calculated and stored. Then the number of classes are calculated and stored. Then the data is randomly shuffled with the randperm function in the MATLAB.

After this the number of patterns for training is set. Then the training set and test set are defined. Then the model of our training is computed with the test set. Then the model is tested with the test set. Then the accuracy function is called to compute the accuracy and the result are published.

Here are the functions used:

A. Naive Model Function

This is the training model function. This function computes the number of class instances N_c and the number of instances of each feature of the class N_f_c. And then the N_c and is divided by the number of instances and N_f_c id divided by the number of instances of the classes. Then the result are saved in the matrix format.

$$prob\_class = \frac{N\_c}{observation} \tag{3}$$

$$prob\_attributes\_class = \frac{N\_f\_c}{N\_c} \tag{4}$$

- Inputs of the function are data-set(training data-set), result(training data-set result), classes and levels of attributes.
- Outputs of the functions are P_class and p_attributes_class.

B. Classifier Function

This is the testing function. It computes the P(C—X) for every test set.

$$P(C|X) \quad \alpha \quad \prod_{i=1}^{n} p(x_i|C) \quad P(C) \tag{5}$$

Since P(X) is same for all the classes it is been omitted in the equation as derived from the introduction.

- Inputs of the function are test set, P_class , p_attributes_class and classes.
- Output is prob_classes.

C. Accuracy Function

This function computes the accuracy of the prediction. It gives the percentage of result by comparing the correct guesses and the actual number of tests.

- Inputs of the function are prob_classes and test_result.
- Output is accuracy in prediction.

IV. TASK 3 - IMPROVE THE CLASSIFIER WITH LAPLACE (ADDITIVE) SMOOTHING

In this task the smoothing effect is applied to the model in order solve the inaccuracy in the model. The data in the data set is altered. The data in the first feature is replaced. The and the accuracy of the smoothing function is also tested. Due to the small values of data the result and their accuracy are not enough to study the model and their accuracy. In order to counter this problem the code is looped for 1000 times for both the normal data and alter data, their result are studied.

Below is the Naive smooth function

A. *Naive model smooth function*

This solves the inaccuracy of the naive model. The naive model equation is altered by adding “a” in the numerator and “a*v” in the denominator. Where a=1 and “v” is the level of the feature.

$$prob\_attributes\_class\_smooth = \frac{N\_f\_c + a}{N\_c + a * v} \tag{6}$$

- Inputs of the function are data-set(training data-set, result(training data-set result), classes and levels of attributes.
- Outputs are prob_class_smooth and prob_attr_class_smooth.

V. Conclusion

First the accuracy obtained with out altering the data for the normal and smooth function are 50 and 50 respectively. And it varied rapidly. And the result of the accuracy are not reliable because it only depends on the randomly chosen data lines.
In order to test the smoothing effect the data is altered and the result percentage are still the same and the same problem arises like the results are not reliable.
In order to overcome this issue, the code is looped for 1000 times and the results obtained are 66.1750 for normal function and 58.0250 for smooth function. This is the mean value of 1000 times. Then the result 62.9750 for normal function and 55.3250 for smooth function by altering the data. This helps in studying the smooth function for both the original data and their altered data.
The one more thing from optimizing the model and testing the model, the log function can be introduced in the classifier function to improve the accuracy.

References

[1] https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/
[2] MATLAB Documentation Page
[3] https://en.wikipedia.org/wiki/Naive_Bayes_classifier

MACHINE LEARNING LAB 2

Abstract—**In this assignment the the regression model is implemented in two data sets. Then the mean square error are computed. Then only the portion of the data is tested.**

I. INTRODUCTION

Linear Regression is a supervised machine learning algorithm which model a relationship between a dependent variable and a target variable. It finds the relationship between a target variable t with the dependent variable x. There are many number of dependent variable.

$$t \approx y \tag{1}$$

$$y = wx \tag{2}$$

Then in a vector form,

$$\begin{bmatrix} t1 \\ t2 \\ t3 \\ \vdots \\ tn \end{bmatrix} \simeq \begin{bmatrix} y1 \\ y2 \\ y3 \\ \vdots \\ yn \end{bmatrix} = \begin{bmatrix} x1\mathbf{1} & \dots & x1d \\ x2\mathbf{1} & \dots & x2d \\ x3\mathbf{1} & \dots & x3d \\ \vdots & \ddots & \vdots \\ xn\mathbf{1} & \dots & xNd \end{bmatrix} \begin{bmatrix} w1 \\ w2 \\ w3 \\ \vdots \\ wn \end{bmatrix} \tag{3}$$

The square loss is,

$$\lambda_E(t, y) = (t - y)^2 \tag{4}$$

Then the mean square loss for the whole data is,

$$J = \frac{1}{N}\sum_{i=1}^{n}(t_i - y_i)^2 \tag{5}$$

II. TASK 1 GET DATA

The two data sets are used in this assignments. One is MT cars data and other is Turkish stock exchange data. This both csv files are loaded using (importdata) and (readtable) MATLAB functions.

III. TASK 2 FIT A LINEAR REGRESSION MODEL

There are three linear regression functions are created in this task, they are one dimensional linear regression without intercept, one dimensional linear regression with intercept and multi dimensional linear regression function. First the one dimensional linear regression function is called to compute the linear model on the Turkish stock exchange data. And their results are published. Then the 10 percent of the different subsets are used in the same function. Then their results are compared graphically. In the second step the one dimensional linear regression with intercept is called, it computes on the MT cars data by using columns of mpg and weight. Then the results are published. In the last step of the task, the multi dimensional linear regression function is called and it computes on the whole Mt cars data. And their results are published.

Below are the functions used in task 2,

A. *One Dimensional Linear Regression Without Intercept*

This is the simple regression function follows the simple the equation number 2. It computes the weight for the each input.

$$w = \sum_{i=1}^{n} \frac{(x_i t_i)^2}{(x_i)^2} \tag{6}$$

- inputs for the function are x(input or data) and t(target data)
- Output of the function is w

B. *One Dimensional Linear Regression With Intercept*

This function computes with the intercept, It follows the equation y=w$_1x + w_0$.

$$w_1 = \sum_{i=1}^{n} \frac{(x_i - x)(t_i - t)}{(x_i - x)^2} \tag{7}$$

$$w_0 = \sum_{i=1}^{n} (t_i - w_1 x) \tag{8}$$

- inputs for the function are x(input or data) and t(target data).
- Outputs of the function are slopes and intercept.

C. *Multi Dimensional Linear Regression Function*

This function weight vector rather from a one dimensional. In this function Moore-Penrose pseudoinverse is used in the MATLAB.

$$w = (X^T X)^{-1} X^T t \tag{9}$$

- inputs of the function are all x(car data column except mpg)and t(mpg)
- output of the function is w

IV. TASK 3 TEST REGRESSION MODEL

In this task first the data is reduced into 5 percentage and use this a training set. This training set is used to compute the one dimensional intercept, one dimensional without intercept and multi dimensional model again. Then computes the mean square error on the training data. Then the remaining 95 percentage of the data is taken as the test set and test the models. Then the mean square error is computed. All the codes been looped for 10 times to reduce the error that comes in small data. And they are randomly generated in each loop. And the overall average for the 5 percent test set and 95 percent test set mean square error is computed. All the results are published graphically. Below are the functions used in this task,

A. *Mean Square Error for One Dimensional*

For the one dimensional problem mean square error is computed using the formula number 5.

- inputs are x(data), t(target), w1(slope), w0(intercept)
- output is Mean square error

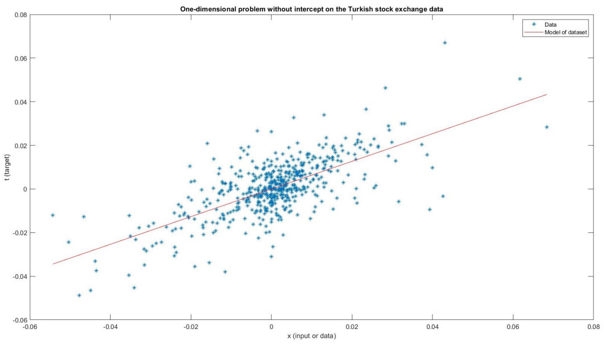
B. *Mean Square Error for Multi Dimensional*

For the multi dimensional problem mean square error is computed using the "immse" MATLAB mean square error function.

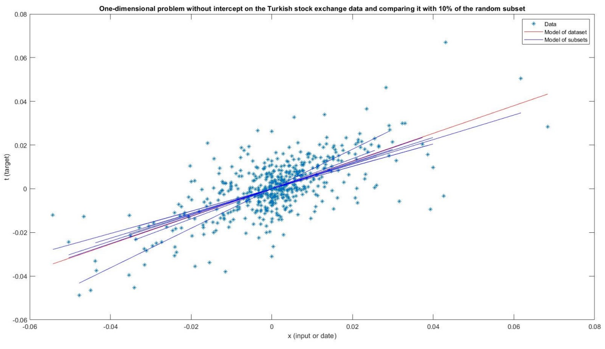
- Inputs are x(data), t(target), w1(slope)
- Output is Mean square error.

V. Conclusion

- The linear regression performed on the Turkish data set without intercept is shown below,

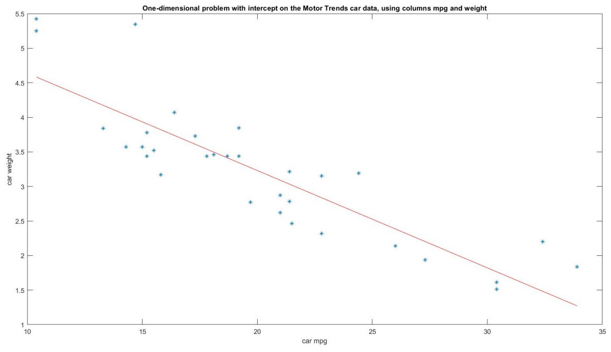


- Below is the result when the 10 percent of the Turkish data has been randomly generated and their result are compared with the full data set.

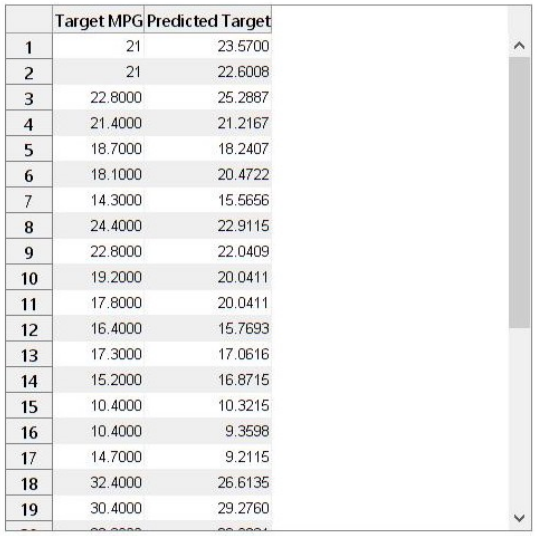


What we can conclude from the graph is due to the small data set the result are not accurate in comparison with the full data set.

- Below is the result obtained when the one dimensional model with intercept computed on the 2 features of the MT car data set.

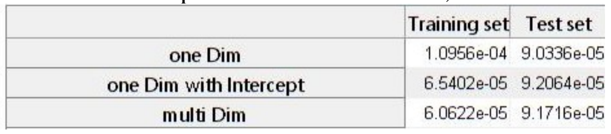


- The result obtained when multi dimensional model with intercept computed on the whole MT car data set.



	Target MPG	Predicted Target
1	21	23.5700
2	21	22.6008
3	22.8000	25.2887
4	21.4000	21.2167
5	18.7000	18.2407
6	18.1000	20.4722
7	14.3000	15.5656
8	24.4000	22.9115
9	22.8000	22.0409
10	19.2000	20.0411
11	17.8000	20.0411
12	16.4000	15.7693
13	17.3000	17.0616
14	15.2000	16.8715
15	10.4000	10.3215
16	10.4000	9.3598
17	14.7000	9.2115
18	32.4000	26.6135
19	30.4000	29.2760

- The comparison result of mean square error for all the model been compared. Here is the result,



	Training set	Test set
one Dim	1.0956e-04	9.0336e-05
one Dim with Intercept	6.5402e-05	9.2064e-05
multi Dim	6.0622e-05	9.1716e-05

References

[1] https://ml-cheatsheet.readthedocs.io/en/latest/linear_regression.html
[2] MATLAB documentation

MACHINE LEARNING LAB 3

***Abstract*—In this assignment the (K nearest neighbour) classifier is implemented.**

I. INTRODUCTION

KNN or K nearest neighbour classifier works on the principal that the similar class have similarities. So it is sensitive to the local structure of the data. It is used for classification and regression.

The algorithm for KNN classifier is,

$$X = \{x_1, ..., x_l, ..., x_n\} \tag{1}$$

$$\{n_1, .., n_k\} = top - k \parallel x_l - \overline{x} \parallel \tag{2}$$

$$y = node\{t_{n_1}, ..., t_{n_k}\} \tag{3}$$

KNN is a type of classification where the function is only approximated locally and all computation is deferred until function evaluation. Since this algorithm relies on distance for classification, if the features represent different physical units or come in vastly different scales then normalizing the training data can improve its accuracy dramatically.

II. TASK 1 OBTAIN A DATA SET

The mnist data set is used to in this assignment. The data represent handwritten digits in 28x28 greyscale images. It contains the hand written images 0 to 9. The data is added to the MATLAB using the loadMNIST function.

III. TASK 2 BUILD A KNN CLASSIFIER

In this task the data is prepared to be inserted into the classifier. First the data is divided into the training set and into test set. While computing the programs takes longer time to run in the MATLAB, hence the data has been reduced by the factor 1/50. Then they are randomised with the randperm function. Them the KNN classifier function is called and the result are plotted. The functions used in this task,

A. KNN function

This function first test three conditions before computing the KNN classifier. First it check for the number of inputs are sufficient, then it checks the test and training set columns coincides i.e the testing column exactly lesser one column than training set. Them it checks for the k¿0 and k¡= cardinality and threw error if finds any condition fails.

Then it computes the KNN with the help of the MATLAB function ”pdist2”. Then it index the training labels for KNN. Then it classifies the problem and return the mode of K labels. Then the error is computed.

- Input of the function are training set, testing set, k and test label
- Outputs of the function are classification and error in the classifier

B. Plotting function

This is the plotting function to plot the results.

- This function take this as inputs Test_set, classification, k, d and plot.

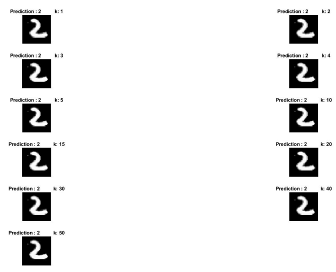
IV. TASK 3 EST THE KNN CLASSIFIER

This is testing task. In this task the KNN classifier is tested and the results are published.

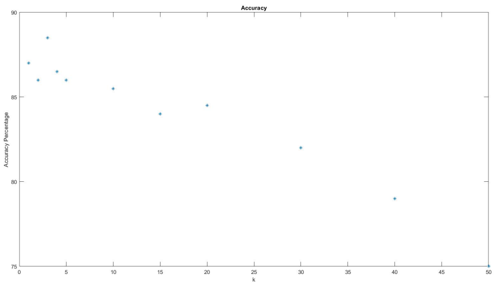
V. CONCLUSION

The result of the classifier build and tested on mnist data is shown below. The results are plotted. Then the individual digits are plotted for accuracy.

Prediction from the data

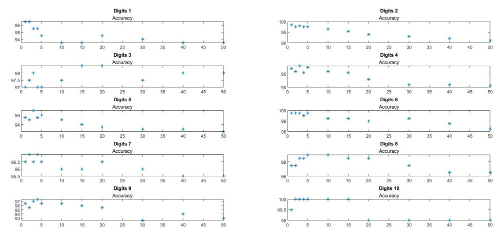


Accuracy plotting



Based on the plot the accuracy are low at the lower value of k. If the k value increased accuracy decreases. This explains that in the higher values of K, it searches for too many neighbours.

Accuracy plotting for individual digits



For individual digits as well the accuracy statement holds. If the k value increased accuracy decreases.

References

[1] https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
[2] https://towardsdatascience.com/machine-learning-basics
[3] MATLAB documentation page

MACHINE LEARNING LAB 4

***Abstract*—The assignment been divided into two task.**

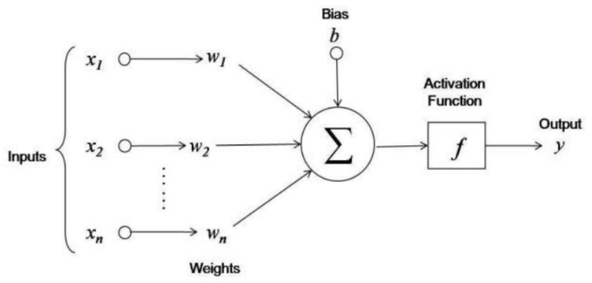
In the first task single unit layer neural network is implemented. They are adaline and perceptron algorithm on three different data sets.

In the second test is to get acquainted with the MATLAB neural environment and their user interface. Get acquainted with the MATLAB toolbox and with the multi layer MATLAB neural network.

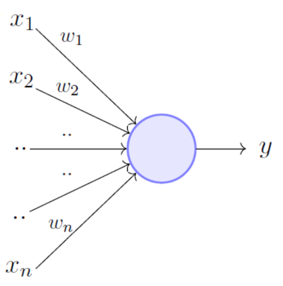
I. INTRODUCTION

The single unit neural network is implemented on the three different data sets. The algorithms are Adaline and Perceptron algorithms are the two single unit neural network. The basic principle is that the all the inputs are weighed and added together to proceed to the activation function. Then the result are shown in confusion matrix. Confusion matrix shows the result of the performance of both the algorithm. The diagonal of the matrix shows the right guesses.

A. Basic Single Unit Neural Network Algorithm



B. Perceptron Algorithm



$$y = 1 \quad if \sum_{i=1}^{n} w_i * x_i \geq \theta$$

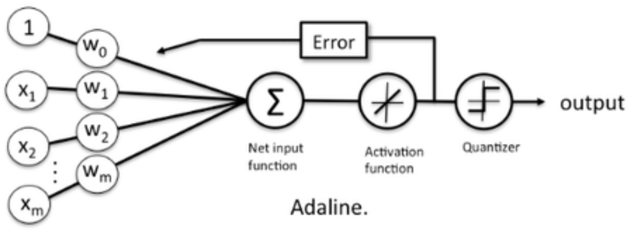
$$= 0 \quad if \sum_{i=1}^{n} w_i * x_i < \theta$$

Rewriting the above,

$$y = 1 \quad if \sum_{i=1}^{n} w_i * x_i - \theta \geq 0$$

$$= 0 \quad if \sum_{i=1}^{n} w_i * x_i - \theta < 0$$

C. Adaline Algorithm



II. TASK A

A. Task 1 and task 2

In this assignment three data types been used to implement the two algorithms. First the data set been loaded into the MATLAB. Then the data splitting with the split function, then the algorithm functions been called and processed. And with the help of image functions result are printed. The data set and functions used,

1) Data Sets:

- The Iris reduced data set with two classes been loaded into the program.
- MNIST data set been reduced by the factor of 1/10th in order to reduce the run time of the program.
- The last is the XOR table data set with two classes.

2) Split Function: This function splits the data set into training and test set. It splits the data with the four conditions. The number of sets depends on the value of k. If the value of k is 2, then it splits the data into two equal parts with one data set. If the value of k is equal to rows, then it splits into number of sets into rows. If the value is 2¡k¡rows, then the k is number of sets. If the value is K¡2 and k¿rows it shows error.

- Inputs are the data sets
- Output is number of sets.

3) Perceptron Function: The function implements the perceptron algorithm and computes the weight and confusion matrix. The training set and test set enters the loop, the random weight is generated. Then the threshold is set for comparison. For each loop, the (r,a and δ) are calculated and weights are updated. After the loop set of weights been generated. Then the labels are predicted. Then the mean confusion matrix is build.

- Inputs are Set, number of sets and eta.
- Output are confusion matrix and weights.

4) Adaline Function: The function implements the Adaline algorithm and computes the weight and confusion matrix, the only change if delta is lower than the threshold at the loop, it exits. In all the case the basic algorithm are same for both the Perceptron and Adaline.

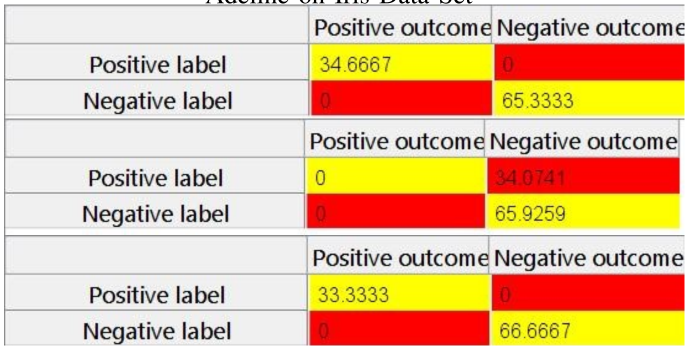
- Inputs are Set, number of sets and eta.
- Output are confusion matrix and weights.

5) Image Function: This function is to save the results of the confusion matrix for all the data sets in a image format.

B. Results

The result for the Adaline and Perceptron algorithm on the Iris data set for the k values (2,10,150). The three results for each algorithm is shown below,

Adeline on Iris Data Set

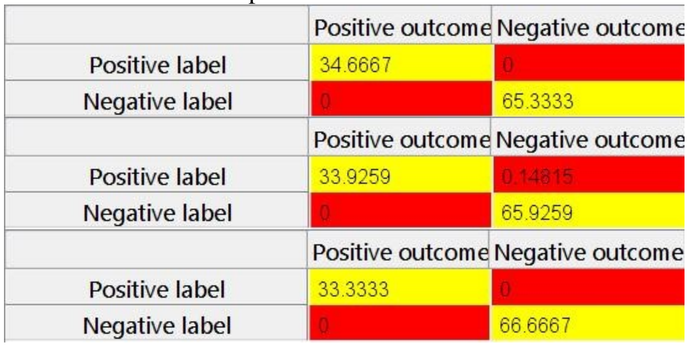


	Positive outcome	Negative outcome
Positive label	34.6667	0
Negative label	0	65.3333

	Positive outcome	Negative outcome
Positive label	0	34.0741
Negative label	0	65.9259

	Positive outcome	Negative outcome
Positive label	33.3333	0
Negative label	0	66.6667

Perceptron on Iris Data Set



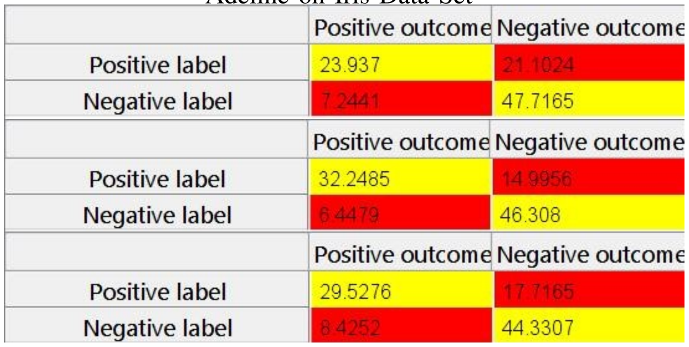
	Positive outcome	Negative outcome
Positive label	34.6667	0
Negative label	0	65.3333

	Positive outcome	Negative outcome
Positive label	33.9259	0.14815
Negative label	0	65.9259

	Positive outcome	Negative outcome
Positive label	33.3333	0
Negative label	0	66.6667

The result for the Adaline and Perceptron algorithm on the MNIST data set for the k values (2,10,150). The three results for each algorithm is shown below,

Adeline on Iris Data Set

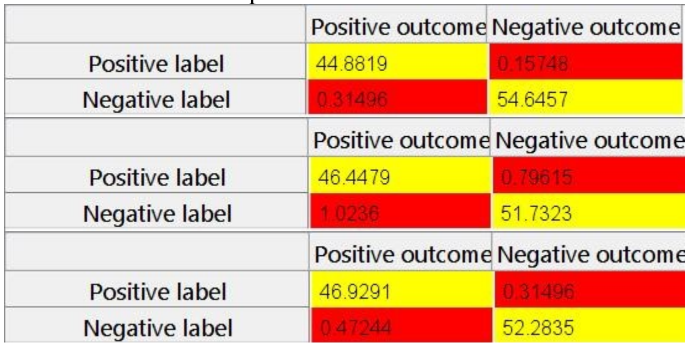


	Positive outcome	Negative outcome
Positive label	23.937	21.1024
Negative label	7.2441	47.7165

	Positive outcome	Negative outcome
Positive label	32.2485	14.9956
Negative label	6.4479	46.308

	Positive outcome	Negative outcome
Positive label	29.5276	17.7165
Negative label	8.4252	44.3307

Perceptron on Iris Data Set



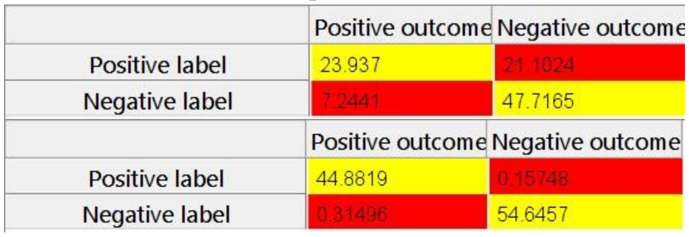
	Positive outcome	Negative outcome
Positive label	44.8819	0.15748
Negative label	0.31496	54.6457

	Positive outcome	Negative outcome
Positive label	46.4479	0.79615
Negative label	1.0236	51.7323

	Positive outcome	Negative outcome
Positive label	46.9291	0.31496
Negative label	0.47244	52.2835

The result for the Adaline and Perceptron algorithm on the XOR data set. The results for each algorithm is shown below,

Adeline and Perceptron on XOR Data Set



	Positive outcome	Negative outcome
Positive label	23.937	21.1024
Negative label	7.2441	47.7165

	Positive outcome	Negative outcome
Positive label	44.8819	0.15748
Negative label	0.31496	54.6457

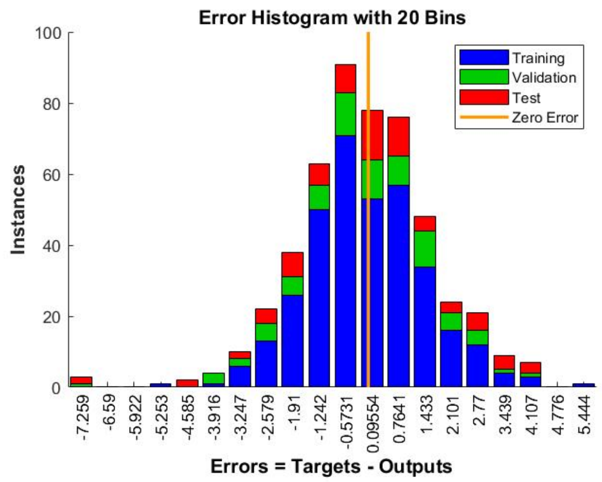
III. Task B

In this task the tutorials are practised in MATLAB.

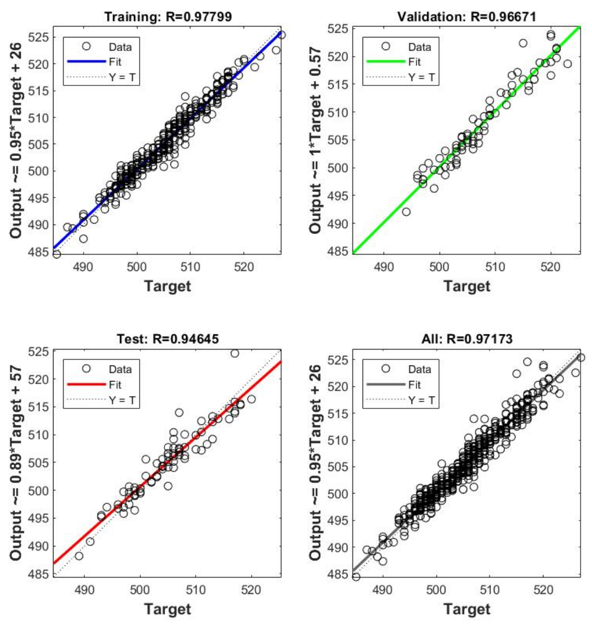
A. Task 0 and Task1

1) Fit Data with a Neural Network: This part is about get oriented with the MATLAB neural network toolbox, the chemical data set is used in the UI. Then the 10 hidden layer is selected. Then the error histogram is plotted. The regression is also plotted and the result are published below,

Error Histogram

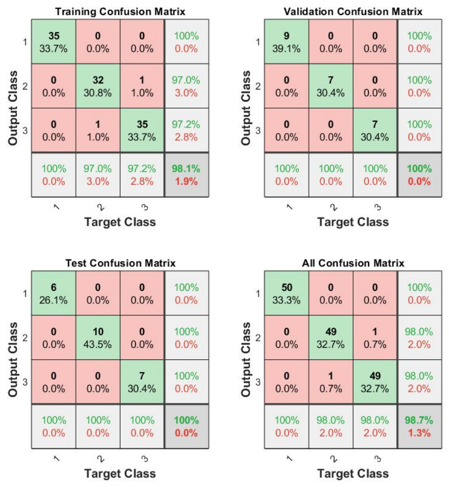


Regression



2) Classify Patterns with a Neural Network: In this section MATLAB classifier neural network UI is used with the data set Iris and wine data set. With different hidden layer on the two data set, the confusion matrix is computed with the correct diagonal classfication. Below are the result,

Confusion matrix for Iris data set



Training Confusion Matrix

	1	2	3	
1	35 33.7%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	32 30.8%	1 1.0%	97.0% 3.0%
3	0 0.0%	1 1.0%	35 33.7%	97.2% 2.8%
	100% 0.0%	97.0% 3.0%	97.2% 2.8%	98.1% 1.9%

Validation Confusion Matrix

	1	2	3	
1	9 39.1%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	7 30.4%	0 0.0%	100% 0.0%
3	0 0.0%	0 0.0%	7 30.4%	100% 0.0%
	100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%

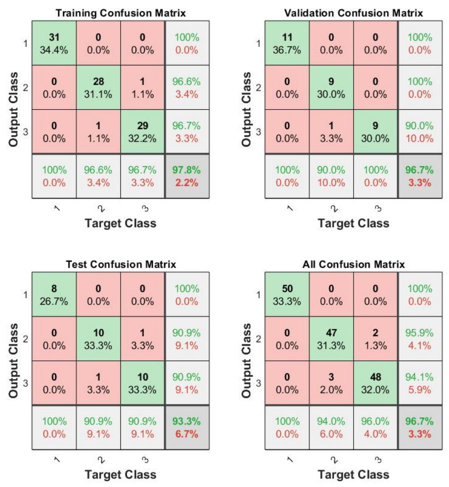
Test Confusion Matrix

	1	2	3	
1	6 26.1%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	10 43.5%	0 0.0%	100% 0.0%
3	0 0.0%	0 0.0%	7 30.4%	100% 0.0%
	100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%

All Confusion Matrix

	1	2	3	
1	50 33.3%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	49 32.7%	1 0.7%	98.0% 2.0%
3	0 0.0%	1 0.7%	49 32.7%	98.0% 2.0%
	100% 0.0%	98.0% 2.0%	98.0% 2.0%	98.7% 1.3%

Output Class
Target Class



Training Confusion Matrix

	1	2	3	
1	31 34.4%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	28 31.1%	1 1.1%	96.6% 3.4%
3	0 0.0%	1 1.1%	29 32.2%	96.7% 3.3%
	100% 0.0%	96.6% 3.4%	96.7% 3.3%	97.8% 2.2%

Validation Confusion Matrix

	1	2	3	
1	11 36.7%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	9 30.0%	0 0.0%	100% 0.0%
3	0 0.0%	1 3.3%	9 30.0%	90.0% 10.0%
	100% 0.0%	90.0% 10.0%	100% 0.0%	96.7% 3.3%

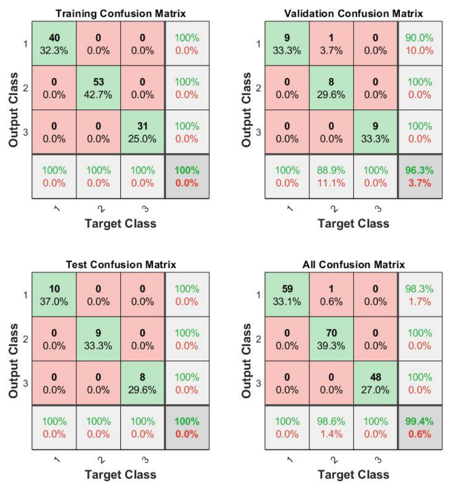
Test Confusion Matrix

	1	2	3	
1	8 26.7%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	10 33.3%	1 3.3%	90.9% 9.1%
3	0 0.0%	1 3.3%	10 33.3%	90.9% 9.1%
	100% 0.0%	90.9% 9.1%	90.9% 9.1%	93.3% 6.7%

All Confusion Matrix

	1	2	3	
1	50 33.3%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	47 31.3%	2 1.3%	95.9% 4.1%
3	0 0.0%	3 2.0%	48 32.0%	94.1% 5.9%
	100% 0.0%	94.0% 6.0%	96.0% 4.0%	96.7% 3.3%

Confusion matrix for wine data set



Training Confusion Matrix

	1	2	3	
1	40 32.3%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	53 42.7%	0 0.0%	100% 0.0%
3	0 0.0%	0 0.0%	31 25.0%	100% 0.0%
	100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%

Validation Confusion Matrix

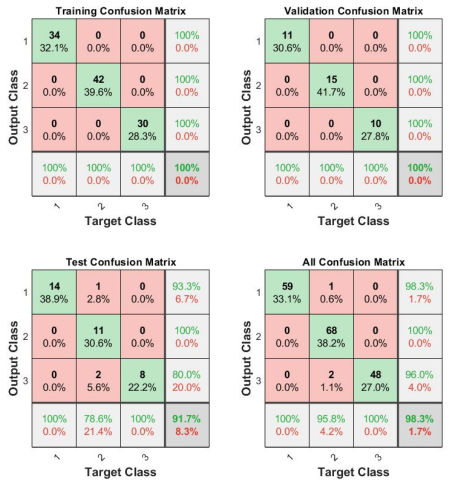
	1	2	3	
1	9 33.3%	1 3.7%	0 0.0%	90.0% 10.0%
2	0 0.0%	8 29.6%	0 0.0%	100% 0.0%
3	0 0.0%	0 0.0%	9 33.3%	100% 0.0%
	100% 0.0%	88.9% 11.1%	100% 0.0%	96.3% 3.7%

Test Confusion Matrix

	1	2	3	
1	10 37.0%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	9 33.3%	0 0.0%	100% 0.0%
3	0 0.0%	0 0.0%	8 29.6%	100% 0.0%
	100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%

All Confusion Matrix

	1	2	3	
1	59 33.1%	1 0.6%	0 0.0%	98.3% 1.7%
2	0 0.0%	70 39.3%	0 0.0%	100% 0.0%
3	0 0.0%	0 0.0%	48 27.0%	100% 0.0%
	100% 0.0%	98.6% 1.4%	100% 0.0%	99.4% 0.6%



Training Confusion Matrix

	1	2	3	
1	34 32.1%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	42 39.6%	0 0.0%	100% 0.0%
3	0 0.0%	0 0.0%	30 28.3%	100% 0.0%
	100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%

Validation Confusion Matrix

	1	2	3	
1	11 30.6%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	15 41.7%	0 0.0%	100% 0.0%
3	0 0.0%	0 0.0%	10 27.8%	100% 0.0%
	100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%

Test Confusion Matrix

	1	2	3	
1	14 38.9%	1 2.8%	0 0.0%	93.3% 6.7%
2	0 0.0%	11 30.6%	0 0.0%	100% 0.0%
3	0 0.0%	2 5.6%	8 22.2%	80.0% 20.0%
	100% 0.0%	78.6% 21.4%	100% 0.0%	91.7% 8.3%

All Confusion Matrix

	1	2	3	
1	59 33.1%	1 0.6%	0 0.0%	98.3% 1.7%
2	0 0.0%	68 38.2%	0 0.0%	100% 0.0%
3	0 0.0%	2 1.1%	48 27.0%	96.0% 4.0%
	100% 0.0%	95.8% 4.2%	100% 0.0%	98.3% 1.7%

REFERENCES

[1] https://towardsdatascience.com/first-neural-network-for-beginners-explained-with-code-4cfd37e06eaf
[2] https://en.wikipedia.org/wiki/ADALINE

[3] https://en.wikipedia.org/wiki/Perceptron#:~:text=In%20machine%20learning%2C%20the%20perceptron,supervised%20learning%20of%20binary%20classifiers.&text=It%20is%20a%20type%20of,weights%20with%20the%20feature%20vector.
[4] https://it.mathworks.com/help/deeplearning/gs/shallow-networks-for-pattern-recognition-clustering-and-time-series.html;jsessionid=50ac9740cb8c23a4959c825ba987
[5] https://it.mathworks.com/help/deeplearning/gs/fit-data-with-a-neural-network.html
[6] https://it.mathworks.com/help/deeplearning/gs/classify-patterns-with-a-neural-network.html
[7] MATLAB documentation

MACHINE LEARNING LAB 5

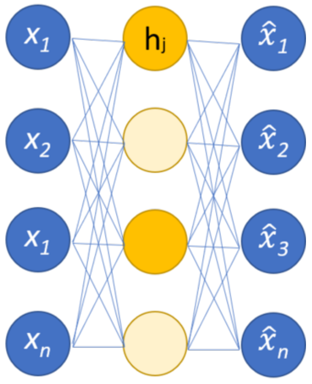
***Abstract*—The assignment been divided into two task. In the first task Autoencoder is implemented. In the second test is to get acquainted with the MATLAB deep learning tool box and their user interface.**

I. TASK A

A. *Introduction*

An Autoencoder is a type of artificial neural network used to learn efficient data codings in an unsupervised manner. The aim of an autoencoder is to learn a representation (encoding) for a set of data, typically for dimensionality reduction, by training the network to ignore signal "noise".

Basic Autoencoder



It has one input and output layer with the hidden layer inside. Simple schema of a single-layer sparse autoencoder. The hidden nodes in bright yellow are activated, while the light yellow ones are inactive. The activation depends on the input.

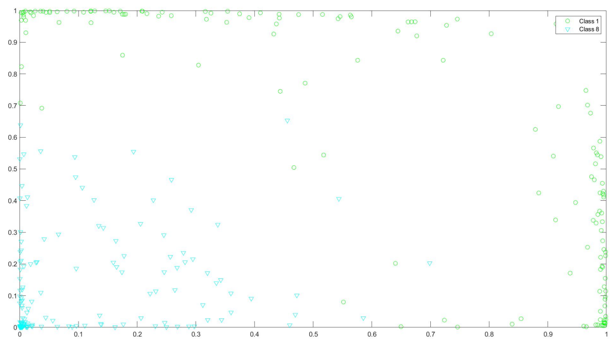
B. *Task 1 Autoencoder*

In this task the autoencoder function is run on MNIST data set. First the MNIST data is loaded into the MATLAB. Then the data been randomly splited into the subset. Then the two digits are selected from the data set. The three sets been selected in order to test the autoencoder. And by using the encode() and trainAutoencoder() function to train and obtain in compressed format on the data set. The it plots the result.

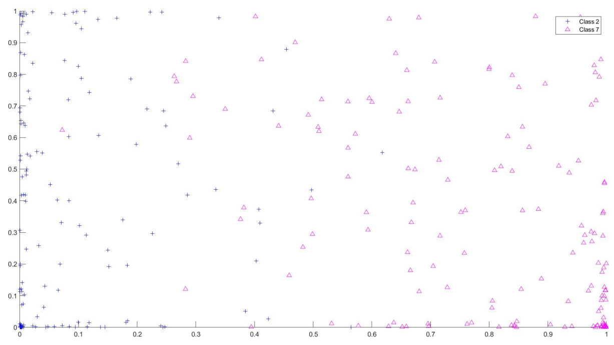
C. *Result*

There are three sets of two digits selected, (1,8) (2,7) (5,10) these are the data selected. The results are plotted and shown below.

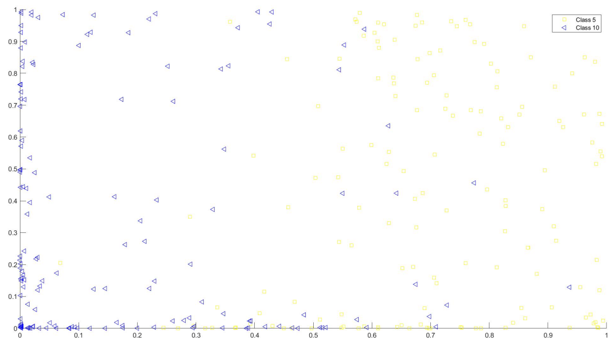
Graphical Result of 1 and 8



Graphical Result of 2 and 7



Graphical Result of 5 and 10



II. TASK B

A. *Introduction*

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural network, most commonly applied to analyze visual imagery. It takes image as an input classify them by taking weights on the input and classify them.

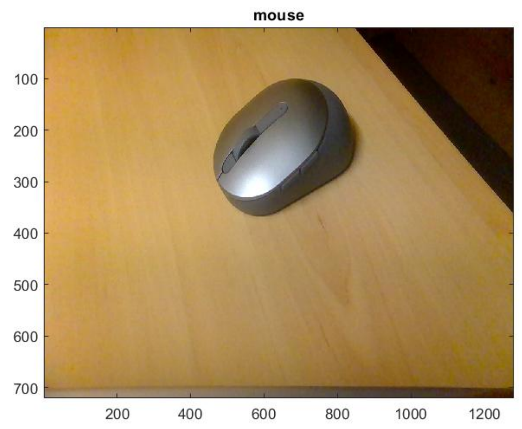
CNNs are regularized versions of multilayer perceptrons. These are the layers,

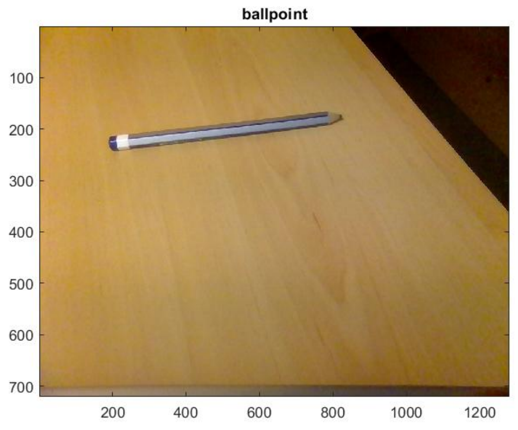
- Convolutional Layers In a CNN, the input is a tensor with a shape: (number of inputs) x (input height) x (input width) x (input channels). In convolutional layer first extract the low levels on the image and builds from it.
- Pooling Layers Pooling layers reduce the dimensions of data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. There are two common types of pooling in popular use: max and average pooling +.
- Fully Connected Layers Fully connected layers connect every neuron in one layer to every neuron in another layer. It is the same as a traditional multi-layer perceptron neural network. The flattened matrix goes through a fully connected layer to classify the images.
- Receptive Field In a convolutional layer, each neuron receives input from only a restricted area of the previous layer called the neuron's receptive field. This is due to applying the convolution over and over, which takes into account the value of a pixel, as well as its surrounding pixels.
- Weights Learning consists of iteratively adjusting these biases and weights. The vector of weights and the bias are called filters and represent particular features of the input.

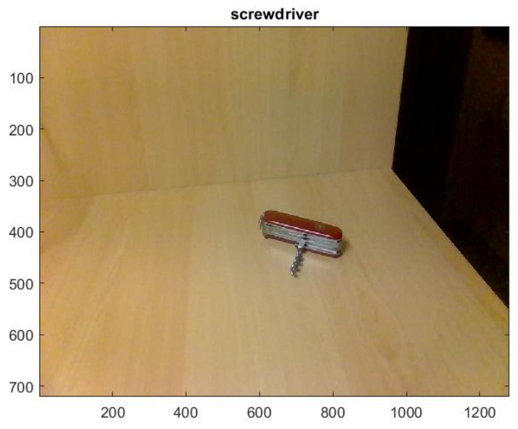
B. Task 0 Deep learning MATLAB demo

In this task using deep learning the objects detected in the webcam are defined with the help of 10 lines of code in the MATLAB. This uses AlexNet pretrained network to implement the image classification. Here are some of the results obtained,

Image Classification Examples



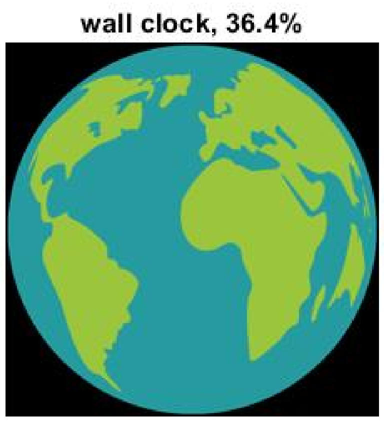




C. Task 1 Use Pretrained Convolutional Neural Networks

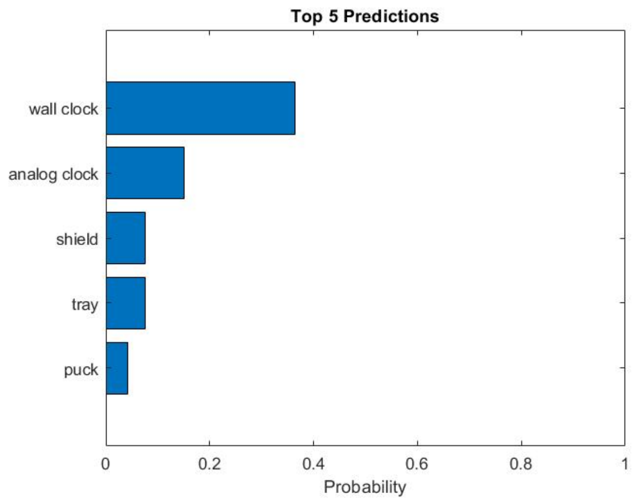
Int this task using pretrained convolutional neural network the image classification is studied. I have used to two random image as a input, and it predict the cat image correctly and predicted the earth as wall clock as wrongly. This result shows that the network been confused by the round shape. There are some disadvantages in it. Here are the results,

Image Classification Examples



With the top prediction algorithm we can obtain the top 5 predictions and their percenatage. Here is the result of the earth image,

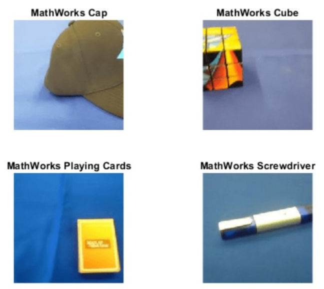
Image Classification Examples



D. *Task 2 Transfer Learning Via Feature Extraction*

In this task the learned image features from a pretrained convolutional neural network and use those features to train an image classifier is studied in the MATLAB. The MATLAB tutorial is executed and the result are studied.

Image Classification Examples

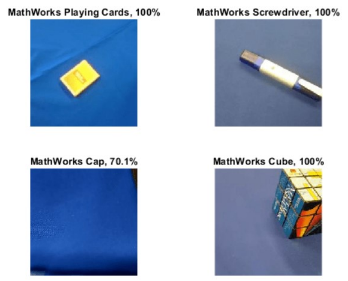


Using shallow network the accuracy is computed, and it is high always. The MerchData is the data set used in this tutorial, it is predefined in the MATLAB.

E. *Task 3 Transfer Learning Via Retraining*

In this task retrain a convolutional neural network to classify a new set of images. The key things need to be noted in this tutorial is, it used google net as nuerla network. Then Replace Final Layers in the pretrained network, it is followed by Freeze Initial Layers. And it trains the network and classify the images. In the freeze we can ”freeze” the weights of earlier layers in the network by setting the learning rates in those layers to zero. Then in the train stage use augmented image and define the epochs. This train the CNN to our application. Below is the image classification with the probability labelled.

Image Classification Examples with Probability



References

[1] https://en.wikipedia.org/wiki/Autoencoder#:~:text=An%20autoencoder%20is%20a%20type,to%20ignore%20signal%20%E2%80%9Cnoise%E2%80%9D.
[2] https://en.wikipedia.org/wiki/Convolutional_neural_network
[3] https://it.mathworks.com/help/deeplearning/gs/try-deep-learning-in-10-lines-of-matlab-code.html
[4] https://www.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html
[5] https://it.mathworks.com/help/deeplearning/ug/extract-image-features-using-pretrained-network.html
[6] https://www.mathworks.com/help/deeplearning/ug/train-deep-learning-network-to-classify-new-images.html
[7] MATLAB documentation page