מפרט תכן תוכנה

HOCR

2011

Maxim Drabkin

Niv Maman

*אוניברסיטת חיפה*

*החוג למדעי המחשב*

# תוכן העניינים

[Class & Dependency Diagram 3](#_Toc298607808)

[מחלקת MainForm 4](#_Toc298607809)

[תיאור המחלקה 4](#_Toc298607810)

[משתנים גלובליים 4](#_Toc298607811)

[מתודות 5](#_Toc298607812)

[מחלקת FileActions 10](#_Toc298607813)

[תיאור המחלקה 10](#_Toc298607814)

[משתנים גלובליים 10](#_Toc298607815)

[מתודות 10](#_Toc298607816)

[מחלקת NeuralNetwork 12](#_Toc298607817)

[תיאור המחלקה 12](#_Toc298607818)

[משתנים גלובליים 12](#_Toc298607819)

[מתודות 12](#_Toc298607820)

[מחלקת TrainArgs 13](#_Toc298607821)

[תיאור המחלקה 13](#_Toc298607822)

[משתנים גלובליים ומתודות (get & set): 13](#_Toc298607823)

[מחלקת TrainingUC 14](#_Toc298607824)

[תיאור המחלקה 14](#_Toc298607825)

[משתנים גלובליים 14](#_Toc298607826)

[מתודות 14](#_Toc298607827)

[מחלקת ImageFilter 16](#_Toc298607828)

[תיאור המחלקה 16](#_Toc298607829)

[משתנים גלובליים 16](#_Toc298607830)

[מתודות 17](#_Toc298607831)

[מחלקת FontsActions 19](#_Toc298607832)

[תיאור המחלקה 19](#_Toc298607833)

[משתנים גלובליים 19](#_Toc298607834)

[מתודות 19](#_Toc298607835)

[מחלקת TextImageActions 22](#_Toc298607836)

[תיאור המחלקה 22](#_Toc298607837)

[משתנים גלובליים 22](#_Toc298607838)

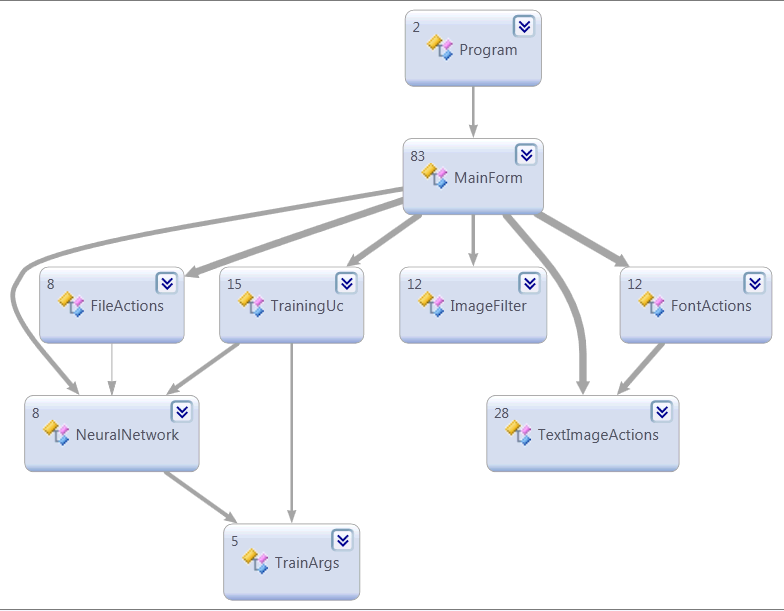
[מתודות 22](#_Toc298607839)

[Sequence Diagrams 28](#_Toc298607840)

[בחירת תמונה (תמונה או קובץ PDF) 28](#_Toc298607841)

[אימון רשת ניורונים 30](#_Toc298607842)

# Class & Dependency Diagram



# מחלקת MainForm

### תיאור המחלקה

מחלקה זו היא המחלקה העיקרית בתוכנה, אשר אחראית על ניהול ה- User Interface מול המשתמש.

מחלקה זו אחראית על קבלת פקודות מהמשתמש, קריאה למחלקות השונות על מנת לבצע את הפעולות, והצגת התוצאות עבור המשתמש.

### משתנים גלובליים

private const int MaxWidth = 500; // Max width size for the image. larger images will be

// resized to fit

private const int MaxHeight = 800;// Max height size for the image. larger images will be

// resized to fit

//GUI

private Form \_trainForm; //training dialog

private int \_currentPage; //saves the current page of image

private int? \_markedLine; //number of marked line

private int? \_markedLetter; //number of marked letter

//Image

private Bitmap \_bitmap; //bitmap of current image

private Bitmap \_picture; //bitmap of current shown image

private Bitmap[][] \_letters; //letters of current image

private int[][][] \_letterBounds; //letters bounds of current image

private string \_currentImagePath; //path of current image

//Neural Network

private NeuralNetwork \_fontNetwork; //pixels neural network of current font

private string \_currentFontPath; //path of current font pixels

private int \_inputLayer; //size of input layer pixels

private readonly int[] \_middleLayers = new[] { 50, 50 }; //size of middle layer

private readonly int \_outputLayer = FontActions.LettersInNetwork.Length; //size of output

// layer

//Script

private bool \_scriptMode; //indicates if we use script mode

### מתודות

private **void** **AdjustWindow**()

**Summary:**

Adjust window to size of content

private **void** **AutoDetectFontTask**()

**Summary:**

read image, recognize which font using in it and load that font. (it reads only the first line to determine)

private **int** **CalculateLetter**(**System.Drawing.Bitmap** *letter*)

**Summary:**

Get letter bitmap, calculate which letter is it, and return it's index.

**Parameters:**

*letter*: letter bitmap

**Returns:**

index of letter

private **void** **CenterizeClick**(**object** *sender***, System.EventArgs** *e*)

**Summary:**

Event of clicking on Centerize button that call for centeize task and center the image

private **void** **CenterizeTask**()

**Summary:**

Center the image and save it into current bitmap. It also show the result on screen

private **void** **CleanClick**(**object** *sender***, System.EventArgs** *e*)

**Summary:**

Event of clicking on Clean button, that call for Clean task

private **void** **CleanTask**()

**Summary:**

Task that take the current image bitmap and clean it using the median filter

private **void** **CreateAllFontsScriptClick**(**object** *sender***, System.EventArgs** *e*)

**Summary:**

Event of clicking on Create All Fonts Script that call for the appropriate task that create the font images and then create the networks from it

private **void** **CreateAllFontsScriptTask**()

**Summary:**

Task that create the font images and then create the networks from it

private **void** **CreateFontClick**(**object** *sender***, System.EventArgs** *e*)

**Summary:**

Event of clicking Create Font button, that loads font image from using dialog and create font network from it.

private **void** **CreateFontImages**()

**Summary:**

Create font images of all known fonts of windows

private **string** **CreateFontTask**([**string** *name* = null])

**Summary:**

Create font action that take the current image and make font network from it. return message in case of error, otherwise returns null.

**Returns:**

message in case of error

private **void** **CreateNetworksFromImages**()

**Summary:**

Create all networks from images of fonts that are now known to the application

private **void** **EmptyFontClick**(**object** *sender***, System.EventArgs** *e*)

**Summary:**

Event of clicking on Empty Font button, that create an empty font network

private **void** **FontsComboBoxSelectedIndexChanged**(**object** *sender***, System.EventArgs** *e*)

**Summary:**

Event of changing Font ComboBox selection, that loads the selected network.

private **void** **GetMultiPageTextTask**()

**Summary:**

Task of recognizing text of a multi pages image or pdf

private **bool** **GetPageTextTask**()

**Summary:**

Task of recognizing text of one page out of multi pages image. It also returns parameter that indicates whether operation succeed

**Returns:**

is succeed

private **void** **GetTextCallBack**(**System.IAsyncResult** *ar*)

**Summary:**

Called back after auto detect font, and call for get text task

**Parameters:**

*ar*: async result

private **void** **GetTextClick**(**object** *sender***, System.EventArgs** *e*)

**Summary:**

Event of clicking Get Text button, that use the network to retrieve the text.

private **void** **GetTextTask**()

**Summary:**

Task that been called when click on Get Text button

private **void** **InAction**(**bool** *inAction*)

**Summary:**

Get boolean that indicates if application in action, and enabled or disable the buttons depending on state.

**Parameters:**

*inAction*: application action state

private **void** **InitializeComponent**()

**Summary:**

Required method for Designer support - do not modify the contents of this method with the code editor.

private static **void** **InitializeFolders**()

**Summary:**

Check existence of fonts and results folders and create them if needed

private **System.Drawing.Bitmap** **LoadImage**(**string** *path*)

**Summary:**

Load the image to picturebox and return bitmap. returns null in case of failure loading image.

**Parameters:**

*path*: path of image

**Returns:**

image bitmap

private **void** **MarkCurrentLetter**(**int** *line***, int** *letter***, System.Drawing.Color** *color*)

**Summary:**

Mark the current letter on source picturebox, using the letters bounds

private **void** **PageNextClick**(**object** *sender***, System.EventArgs** *e*)

**Summary:**

Event of clicking on Next Page button that showing the next image

private **void** **PagePrevClick**(**object** *sender***, System.EventArgs** *e*)

**Summary:**

Event of clicking on Prev Page button that showing the prev image

private **void** **RepairNetworkClick**(**object** *sender***, System.EventArgs** *e*)

**Summary:**

Event of clicking Repair Network, that get letter bitmap and correct letter in text and repair the network for current letter.

private **void** **RestoreWindowState**()

**Summary:**

Restoring window to be enabled again after finishing loading text result

private **void** **RotateClick**(**object** *sender***, System.EventArgs** *e*)

**Summary:**

Event of clicking on Rotate button, that call for Rotate task

private **void** **RotateTask**()

**Summary:**

Task that take the current image bitmap and rotate it till straight if needed

private **void** **SelectImageClick**(**object** *sender***, System.EventArgs** *e*)

**Summary:**

Event of Clicking Select Image Button, that open file dialog and asks you for image to use.

private **bool** **SelectImageDialog**()

**Summary:**

Open selection image dialog, set image and return boolean value indicates if it succeed

**Returns:**

boolean value indicates if it succeed

private **void** **ShowLetterResult**(**System.Drawing.Image** *bitmap*)

**Summary:**

Get image bitmap and show it at the result picture box

**Parameters:**

*bitmap*: path of image

private **void** **ShowPagesButtons**(**bool** *visibility***, int** *numberOfPages*)

**Summary:**

Get boolean indicates whether to show next\prev buttons. if true, set their location.

**Parameters:**

*visibility*: visibility value

*numberOfPages*: number of pages in current image

private **void** **ShowResultLetterText**(**int** *lineNumber***, int** *letterNumber*)

**Summary:**

Get line and letter numbers and show that letter on screen at the repair section

private **void** **ShowSource**(**System.Drawing.Bitmap** *bitmap*)

**Summary:**

Get image bitmap and show it at the source picture box

**Parameters:**

*bitmap*: path of image

private **void** **ShowText**(**int** *numberOfLines***, int** *maxLine*)

**Summary:**

Show Text Result

private **void** **StopProcessClick**(**object** *sender***, System.EventArgs** *e*)

**Summary:**

Event of clicking on Stop Process button that stop any process by setting his own button visibility to false. It is only posible if the process checkes the visiblity of the button every cycle.

private **void** **TextResultSelectionChanged**(**object** *sender***, System.EventArgs** *e*)

**Summary:**

Event of changing selection text in Text Result, that calc selected letter and show it on repair section

private **void** **ThresholdClick**(**object** *sender***, System.EventArgs** *e*)

**Summary:**

Event of clicking on Threshold button, that call for Threshold task

private **void** **ThresholdTask**()

**Summary:**

Task that take the current image bitmap and fix it using the threshold filter

private **void** **TrainAllScriptClick**(**object** *sender***, System.EventArgs** *e*)

**Summary:**

Event of clicking on Train All Script button that opens the training user control and start training all fonts.

private **void** **TrainNetworkClick**(**object** *sender***, System.EventArgs** *e*)

**Summary:**

Event of clicking Train Network button, that train the network and show test results.

private **void** **UpdateFontsComboBox**()

**Summary:**

Enter fonts items into fonts combobox

# מחלקת FileActions

### תיאור המחלקה

מחלקה זו אחראית על ניהול הממשק מול מערכת הקבצים (גם Windows וגם Linux).

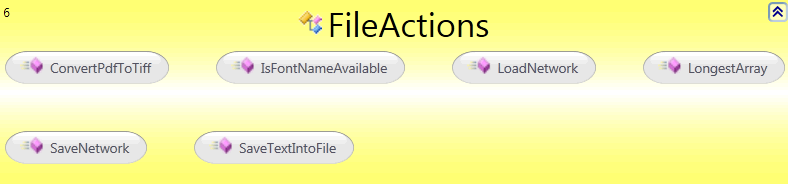
מחלקה זו מכילה מתודות סטטיות אשר מספקות utilities על מנת לקרוא, לערוך ולשמור קבצים במכונה שעליה התוכנה עובדת.

### משתנים גלובליים

public const string ResultsFolderPath = @"Results\"; // Path to where all results will be

//saved

### מתודות



public static **string** **ConvertPdfToTiff**(**string** *path*)

**Summary:**

Get path of pdf file, create tif image from it, and return the path of the new tif file. Assumption: GhostScript\gswin32c.exe is exists

**Parameters:**

*path*: path of pdf file

**Returns:**

path of new tif file

public static **bool** **IsFontNameAvailable**(**string** *fontName*)

**Summary:**

Get font name and return value indicates if the name is available (true) or already in use (false).

**Parameters:**

*fontName*: font name

**Returns:**

boolean indicates if it available

public static **HOCR.NeuralNetwork** **LoadNetwork**(**string** *path*)

**Summary:**

Get path of network file and return the network

**Parameters:**

*path*: path of network file

**Returns:**

Network

public static **int** **LongestArray**(**System.Collections.Generic.IEnumerable<object[]>** *items*)

**Summary:**

Get array of objects array and return the size of the longest array in it.

**Parameters:**

*items*: array of arrays

**Returns:**

size of longest array

public static **string** **SaveNetwork**(**string** *path***, HOCR.NeuralNetwork** *network*)

**Summary:**

Get path of file to store network and network. stores the network in that file.

**Parameters:**

*path*: path of file

*network*: network

public static **string** **SaveTextIntoFile**(**string** *text***, string** *path*)

**Summary:**

Get text and path and write the text into that path

**Parameters:**

*text*: text to write

*path*: path to write to

**Returns:**

exception message or null if ok

# מחלקת NeuralNetwork

### תיאור המחלקה

מחלקה זו מייצגת את הרשת הניורונית אשר מכילה את המידע הנלמד עד כה, ואפשרויות לאימון נוסף בהנתן קלט נוסף (או תיקון קלט קיים)

### משתנים גלובליים

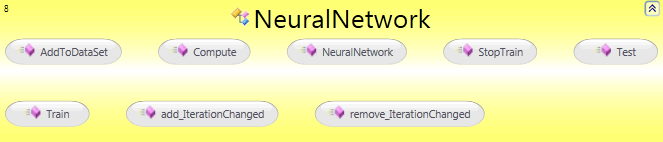
private readonly BasicNetwork \_network; //neural network

private readonly INeuralDataSet \_dataSet; //network data

private bool \_isActive; //is network active

public event EventHandler<TrainArgs> IterationChanged;//event that raised every iteration

### מתודות



public **void** **AddToDataSet**(**double[][]** *data*)

**Summary:**

Get new letter data and add it to data set

**Parameters:**

*data*: new letter data

public **double[]** **Compute**(**double[]** *letter*)

**Summary:**

Get input, compute output and return it.

**Parameters:**

*letter*: input sample

**Returns:**

output result

public **NeuralNetwork**(**int** *inputLayer***, System.Collections.Generic.IEnumerable<int>** *middleLayers***, int** *outputLayer***, double[][]** *inputData***, double[][]** *outputData*)

**Summary:**

C'tor. create the network and load dataset

**Parameters:**

*inputLayer*: number of neuron in input layer

*middleLayers*: array of number of neuron in middle layer

*outputLayer*: number of neuron in output layer

*inputData*: array of vector represents input data

*outputData*: array of vector represents output data

public **void** **StopTrain**()

**Summary:**

Stop training

public **string** **Test**()

**Summary:**

Test the network and return a string of all the samples with the network result

**Returns:**

string of result of all samples

public **void** **Train**([**int** *repeats* = 10000]**,** [**double** *error* = 0.001]**,** [**double** *learnRate* = 0.1]**,** [**double** *momentum* = 0.1])

**Summary:**

Train the neural network on the dataset

**Parameters:**

*repeats*: number of repeats

*error*: error to stop

*learnRate*: learn rate

*momentum*: momentum

# מחלקת TrainArgs

### תיאור המחלקה

מחלקה זו יורשת מ- EventArgs ונשלחת כפרמטר בכל פעם באימון הרשת משתנה.

### משתנים גלובליים ומתודות (get & set):

public int Iterations { get; set; } // number of iterations in training

public double Error { get; set; } // percent of errors in training

# מחלקת TrainingUC

### תיאור המחלקה

מחלקה זו אחראית על אימון רשת ניורונים חדשה. היא מספקת למשתמש User Interface שבו המשתמש יכול להגדיר את הפרמטרים לאימון הרשת.

### משתנים גלובליים

private int \_repeats; //number of repeats

private double \_error, \_learnRate, \_momentum;// training parameters

private NeuralNetwork \_fontNetwork; //pixels neural network of current font

private readonly Action<int, double, double, double> \_trainActionP; //action used for train task P

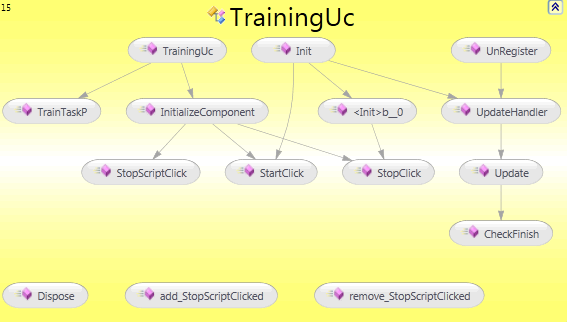
private Form \_parentForm; //parent form, used for closing

public event Action StopScriptClicked; //event of stoping script

private bool \_activeTrain; //indicates if train is active

private bool \_scriptMode; //indicates if we use script

### מתודות



private **void** **CheckFinish**()

**Summary:**

Check if both trainig finished and update user interface if needed.

protected override **void** **Dispose**(**bool** *disposing*)

**Summary:**

Clean up any resources being used.

**Parameters:**

*disposing*: true if managed resources should be disposed; otherwise, false.

public **void** **Init**(**HOCR.NeuralNetwork** *fontNetwork***, bool** *script***, System.Windows.Forms.Form** *parantForm***, string** *fontName*)

**Summary:**

Init font network of training dialog

**Parameters:**

*fontNetwork*: neural network font

*script*: indicates if we use script

*parantForm*: parent form, used for closing

*fontName*: name of font we train

private **void** **InitializeComponent**()

**Summary:**

Required method for Designer support - do not modify the contents of this method with the code editor.

private **void** **StartClick**(**object** *sender***, System.EventArgs** *e*)

**Summary:**

Event of clicking on Start button that read and init variables of training and invoke it.

private **void** **StopClick**(**object** *sender***, System.EventArgs** *e*)

**Summary:**

Event of clicking on Stop button that stop the training process

private **void** **StopScriptClick**(**object** *sender***, System.EventArgs** *e*)

**Summary:**

Event of clicking on Stop Script button that stops script by invoking event and close window

public **TrainingUc**()

**Summary:**

Training user control dialog C'tor

private **void** **TrainTaskP**(**int** *repeats***, double** *error***, double** *learnRate***, double** *momentum*)

**Summary:**

Get training parameters and start pixels training

public **void** **UnRegister**()

**Summary:**

Unregister all events. you should call this method from outside class after closing dialog

private **void** **Update**(**int** *iterations***, double** *error*)

**Summary:**

Update pixels training user interface

**Parameters:**

*iterations*: number of iterations

*error*: current error

private **void** **UpdateHandler**(**object** *sender***, HOCR.TrainArgs** *e*)

**Summary:**

Event handler of updating pixels training state

# מחלקת ImageFilter

### תיאור המחלקה

מחלקה זו אחראית על כל פעולות עיבוד התמונה. בהנתן תמונה עם רעש או בזווית סריקה לא ישרה, או תמונת צבע, המשתמש יכול לבחור לבצע פילטרים על התמונה לפני הקריאה לזיהוי הטקסט שבה.

מחלקה זו אחראית על כל הפילטרים הללו ובלחיצת כפתור תבצע את כל הפילטרים.

### משתנים גלובליים

//colors

static readonly PixelData Black = new PixelData { B = 0, R = 0, G = 0 }; // represents the black pixel

static readonly PixelData White = new PixelData { B = 255, R = 255, G = 255 }; // represents the wgite pixel

//threshold

public const double ThresholdLoose = 0.3; //threshold that more close to black (loose)

public const double ThresholdTight = 0.8; //threshold that more close to white (tight)

// the range of angles to search for lines

private const double MinimalRotationAngle = 0.0001;

private const double CAlphaStart = -20;

private const double CAlphaStep = 0.001;

private const int CSteps = 40 \* 1000;

// pre-calculation of sin and cos

private static double[] \_cSinA;

private static double[] \_cCosA;

// range of d

private static double \_cDMin;

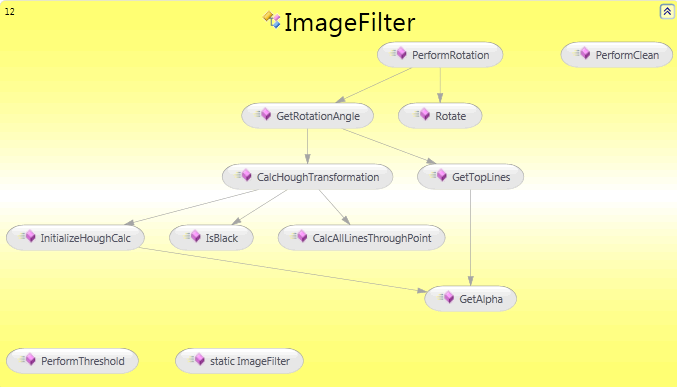
private const double CdStep = 1.0;

private static int \_cDCount;

// count of points that fit in a line

private static int[] \_cHMatrix;

### מתודות



private static **void** **CalcAllLinesThroughPoint**(**int** *x***, int** *y*)

**Summary:**

calculate all lines through the point (x,y)

private static **void** **CalcHoughTransformation**(**HOCR.UnsafeBitmap** *bitmap*)

**Summary:**

perform the Hough Trnasform algorithm in order to find all lines in the picture

private static **double** **GetAlpha**(**int** *i*)

**Summary:**

Get the rotation angle alpha

private static **double** **GetRotationAngle**(**HOCR.UnsafeBitmap** *bitmap*)

**Summary:**

Find the rotation angle for a given bitmap image

private static **HOCR.HoughLine[]** **GetTopLines**(**int** *linesNum*)

**Summary:**

Get the most standout lines from the picture. call this method after the Hough Transorm is done

**Parameters:**

*linesNum*: Number of lines to return

private static **void** **InitializeHoughCalc**(**System.Drawing.Bitmap** *bitmap*)

**Summary:**

Initialize all parameters before starting Hough Transform calculations

private static **bool** **IsBlack**(**HOCR.UnsafeBitmap** *bitmap***, int** *x***, int** *y*)

**Summary:**

check if pixel at point (x,y) is black

public static **System.Drawing.Bitmap** **PerformClean**(**System.Drawing.Bitmap** *bitmap*)

**Summary:**

Get image bitmap and return cleaned image with median filter

**Parameters:**

*bitmap*: image bitmap

**Returns:**

clean image

public static **System.Drawing.Bitmap** **PerformRotation**(**System.Drawing.Bitmap** *bitmap*)

**Summary:**

Get image bitmap and rotate it to the right angle that it's straight

**Parameters:**

*bitmap*: image bitmap

**Returns:**

straight image

public static **System.Drawing.Bitmap** **PerformThreshold**(**System.Drawing.Bitmap** *bitmap*)

**Summary:**

Get image bitmap and return new image after thresholding.

**Parameters:**

*bitmap*: image bitmap

**Returns:**

image after threshold

private static **System.Drawing.Bitmap** **Rotate**(**System.Drawing.Bitmap** *bitmap***, double** *rotationAngle***, int** *midWidth***, int** *midHeight*)

**Summary:**

Perform the rotation of an image, given the rotation angle, and center coordinates

# מחלקת FontsActions

### תיאור המחלקה

מחלקה זו אחראית על ניהול כל הפעולות שניתן לבצע על פונטים, בין היתר יצירת פונט חדש, טעינת פונט קיים וכדומה.

### משתנים גלובליים

public const int LetterSize = 20; //size of letter after resizing

public const string LettersInNetwork = "אבגדהוזחטיכלמנסעפצקרשתךםןףץ0123456789.,;:?!-'\""; //letters

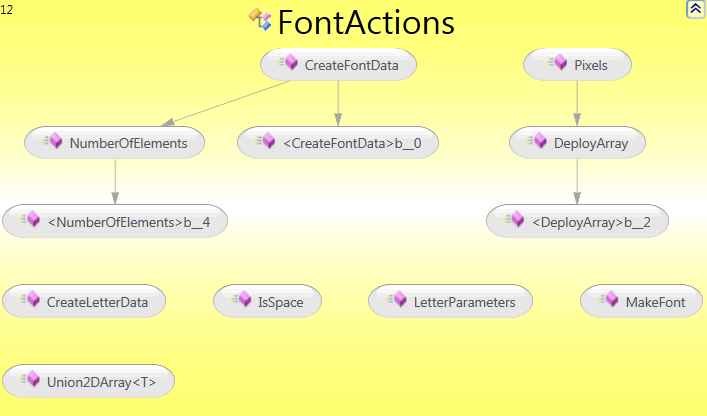
public const string LettersOfInitial = "אבגדהוזחטיכלמנסעפצקרשתךםןףץ0123456789.,;:?!-'"; //letters

private const string LettersPrint1 = "א ב ג ד ה ו ז ח ט י כ ל מ נ ס ע פ צ ק ר ש ת ך ם ן ף ץ";

private const string LettersPrint2 = "' - ! ? : ; , . 9 8 7 6 5 4 3 2 1 0";

public const string FontsFolderPath = @"Fonts\"; //path of font networks

### מתודות



public static **double[][][]** **CreateFontData**(**System.Drawing.Bitmap[][]** *letters***, System.Func<IEnumerable<Bitmap[]>,int,double[][]>** *func*)

**Summary:**

Get array of letters and return array of font data. [input\output][sample number][bitmap vector].

**Parameters:**

*letters*: array of letters

*func*: function

**Returns:**

array of font data

public static **double[][]** **CreateLetterData**(**System.Drawing.Bitmap** *letter***, int** *letterNumber***, int** *numberOfLetters***, System.Func<IEnumerable<Bitmap[]>,int,double[][]>** *func*)

**Summary:**

Get letter bitmap, letter number, and total number of letters in font and return letter data includes input and output of this letter

**Parameters:**

*letter*: letter bitmap

*letterNumber*: letter number

*numberOfLetters*: total number of letters

*func*: function

**Returns:**

letter data

public static **System.Collections.Generic.IEnumerable<Bitmap>** **DeployArray**(**System.Collections.Generic.IEnumerable<Bitmap[]>** *lines***, int** *size*)

**Summary:**

Get array of lines and number of total letters, and return a deployed array of it.

**Parameters:**

*lines*: array of lines

*size*: number of letters

public static **bool** **IsSpace**(**System.Drawing.Bitmap** *letter*)

**Summary:**

Get letter bitmap and return bool value indicates if the letter is space or not.

**Parameters:**

*letter*: letter bitmap

**Returns:**

bool value indicates if the letter is space or not

public static **double[][]** **LetterParameters**(**System.Drawing.Bitmap** *letter***, System.Func<IEnumerable<Bitmap[]>,int,double[][]>** *func*)

**Summary:**

Get letter and function and return letter parameters using that function.

**Parameters:**

*letter*: letter bitmap

*func*: function that get bitmaps and returns parameters

**Returns:**

letter parameters

public static **void** **MakeFont**(**System.Drawing.FontFamily** *font*)

**Summary:**

Get font name and style, draw font image from it and save it on C:\Temp\Fonts\

**Parameters:**

*font*: font

public static **int** **NumberOfElements**(**System.Collections.Generic.IEnumerable<Bitmap[]>** *letters*)

**Summary:**

Get array of letters and return number of elements in it

**Parameters:**

*letters*: array of letters

**Returns:**

number of elements

public static **double[][]** **Pixels**(**System.Collections.Generic.IEnumerable<Bitmap[]>** *letters***, int** *size*)

**Summary:**

Get bitmap array and size, and return vector array

**Parameters:**

*letters*: bitmap array

*size*: size of bitmap array

**Returns:**

vector array

private static **T[][]** **Union2DArray<T>**(**T[][]** *array1***, T[][]** *array2*)

**Summary:**

Get two 2D arrays and unify their inner level array. the array must be of the same size in the first level array, otherwise return null.

**Type Parameters:**

*T*: parameter type of arrays

**Parameters:**

*array1*: first array

*array2*: second array

**Returns:**

unify of 2 arrays

# מחלקת TextImageActions

### תיאור המחלקה

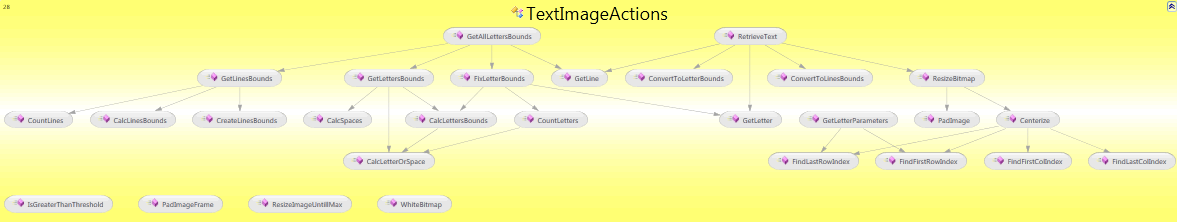
מחלקה סטטית אשר מכילה מתודות לעבודה עם Bitmap. המתודות הן מתודות Utilities אשר מאפשרות להעתיק תמונה, לקבל שורה ספציפית או אות מתוך תמונה ועוד.

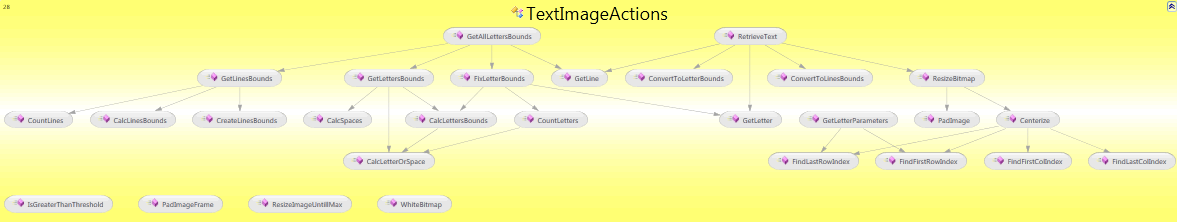
### משתנים גלובליים

private const double MekademFixCoef = 0.6;

private const double MekademLetterSpace = 1.5;

### מתודות





private static **int[]** **CalcLetterOrSpace**(**System.Drawing.Bitmap** *line***, double** *threshold*)

**Summary:**

Get line Bitmap and return array of int that indicates if the Bitmap row is part of a letter (1) or space (0)

**Parameters:**

*line*: line Bitmap

*threshold*: threshold value

**Returns:**

line with letters marked in black and space in white

private static **int[][]** **CalcLettersBounds**(**System.Drawing.Bitmap** *line***, double** *threshold*)

**Summary:**

Get line and number of letters and return array of letters bounds pairs

**Parameters:**

*line*: Bitmap of line

*threshold*: threshold value

**Returns:**

array of letters bounds pairs

private static **int[][]** **CalcLinesBounds**(**System.Collections.Generic.IList<int>** *lineOrSpace***, int** *numberOfLines*)

**Summary:**

Get array of int and the number of lines in Bitmap, and return array of bound pairs (begin and end)

**Parameters:**

*lineOrSpace*: array of int

*numberOfLines*: number of lines

**Returns:**

pairs of bounds

private static **int[][]** **CalcSpaces**(**int[][]** *bounds***, System.Collections.Generic.IList<int>** *letterOrSpace*)

**Summary:**

Get line bounds and letter or space of a line, and return new bounds of line letters with spaces.

**Parameters:**

*bounds*: bounds of line

*letterOrSpace*: letter or space of a line

**Returns:**

new bounds of line letters with spaces

public static **System.Drawing.Bitmap** **Centerize**(**System.Drawing.Bitmap** *bitmap*)

**Summary:**

Get lettet Bitmap and return it's Bitmap centerized

**Parameters:**

*bitmap*: letter Bitmap

**Returns:**

centerized Bitmap

private static **int[][]** **ConvertToLetterBounds**(**int[][][]** *bounds***, int** *lineNumber*)

**Summary:**

Get bounds of bitmap in [line][letter][down,up,left,right] format and line number. and return letter bounds of the given line.

**Parameters:**

*bounds*: bounds of bitmap

*lineNumber*: number of line

**Returns:**

letters bounds

private static **int[][]** **ConvertToLinesBounds**(**int[][][]** *bounds*)

**Summary:**

Get bounds of bitmap in [line][letter][down,up,left,right] format and return lines bounds of it.

**Parameters:**

*bounds*: bounds of bitmap

**Returns:**

lines bounds

private static **int** **CountLetters**(**System.Drawing.Bitmap** *line***, double** *threshold*)

**Summary:**

Get line Bitmap and return number of letters in it

**Parameters:**

*line*: line Bitmap

*threshold*: threshold value

**Returns:**

number of letters in line

private static **int** **CountLines**(**System.Collections.Generic.IList<int>** *lineOrSpace*)

**Summary:**

Get array of int and return the number of lines

**Parameters:**

*lineOrSpace*: array of int

**Returns:**

number of lines

private static **int[]** **CreateLinesBounds**(**System.Drawing.Bitmap** *bitmap*)

**Summary:**

Get Bitmap of full text and return array of int that indicates if the Bitmap line is part of a line (1) or space (0)

**Parameters:**

*bitmap*: full text Bitmap

**Returns:**

array of int

private static **int[][]** **FixLetterBounds**(**System.Drawing.Bitmap** *line***, int[][]** *letterBounds*)

**Summary:**

Get line and letter bounds and return fixed letter bounds. sometimes letters combined and this method separate it.

**Parameters:**

*line*: Bitmap line

*letterBounds*: letter bounds

**Returns:**

fixed letter bounds

public static **int[][][]** **GetAllLettersBounds**(**System.Drawing.Bitmap** *bitmap*)

**Summary:**

Get bitmap and return 3D int array of bounds. [line][letter][top,bottom,right,left]

**Parameters:**

*bitmap*:

private static **System.Drawing.Bitmap** **GetLetter**(**System.Drawing.Bitmap** *line***, System.Collections.Generic.IList<int[]>** *bounds***, int** *number*)

**Summary:**

Get line Bitmap, array of bounds pairs, number of line, and return the letter we asked in the line

**Parameters:**

*line*: Bitmap of line

*bounds*: array of bounds pairs

*number*: number of letter we want

**Returns:**

Bitmap of letter

public static **double[]** **GetLetterParameters**(**System.Drawing.Bitmap** *bitmap*)

**Summary:**

Get Bitmap of letter and return array of letter parameters: relative location of upper and lower pixel and length of letter.

**Parameters:**

*bitmap*: letter Bitmap

**Returns:**

letter parameters

private static **int[][]** **GetLettersBounds**(**System.Drawing.Bitmap** *line***, double** *threshold*)

**Summary:**

Get line and return array of letters bounds pairs

**Parameters:**

*line*: line Bitmap

*threshold*: threshold value

**Returns:**

array of letters bounds pairs

private static **System.Drawing.Bitmap** **GetLine**(**System.Drawing.Bitmap** *bitmap***, System.Collections.Generic.IList<int[]>** *bounds***, int** *number*)

**Summary:**

Get Bitmap, bounds of lines and number of line we want, and return a Bitmap of the line

**Parameters:**

*bitmap*: Bitmap of full text

*bounds*: bounds of lines

*number*: number of line we want

**Returns:**

Bitmap of the line

private static **int[][]** **GetLinesBounds**(**System.Drawing.Bitmap** *bitmap*)

**Summary:**

Get Bitmap, calc the bounds and return it

**Parameters:**

*bitmap*: Bitmap

**Returns:**

pairs of bounds

public static **double** **IsGreaterThanThreshold**(**System.Drawing.Color** *color*)

**Summary:**

Get pixel color and return number that indicates if it greater than threshold (0) or not (1)

**Parameters:**

*color*: pixel color

**Returns:**

indicates if it greater than threshold

private static **System.Drawing.Bitmap** **PadImage**(**System.Drawing.Bitmap** *bitmap*)

**Summary:**

Get Bitmap and return new Bitmap, padded with zeros

**Parameters:**

*bitmap*: image

**Returns:**

padded image

public static **System.Drawing.Bitmap** **PadImageFrame**(**System.Drawing.Bitmap** *bitmap*)

**Summary:**

Get Bitmap and return it with white frame. It used to avoid problems in recognition where a letter extend to the edge of frame.

**Parameters:**

*bitmap*: Bitmap

**Returns:**

fixed Bitmap

public static **System.Drawing.Bitmap** **ResizeBitmap**(**System.Drawing.Bitmap** *bitmap***, int** *width***, int** *height*)

**Summary:**

Get Bitmap and new Bitmap size, and return new Bitmap in that size.

**Parameters:**

*bitmap*: Bitmap

*width*: new width

*height*: new height

public static **System.Drawing.Bitmap** **ResizeImageUntillMax**(**System.Drawing.Bitmap** *bitmap***, int** *newWidth***, int** *newHeight*)

**Summary:**

Get Bitmap and return resized Bitmap with maximum size as parameter. the Bitmap stay in the current width:height ratio

**Parameters:**

*bitmap*: Bitmap to resize

*newWidth*: maximum new width

*newHeight*: maximum new height

**Returns:** new Bitmap

public static **System.Drawing.Bitmap[][]** **RetrieveText**(**System.Drawing.Bitmap** *bitmap***, int[][][]** *bounds***,** [**int** *letterSize* = 0])

**Summary:**

Get Bitmap of text and his bounds and return the Bitmap separated into lines and letters. if failed, return null.

**Parameters:**

*bitmap*: Bitmap of text

*bounds*: bounds of lines and letters

*letterSize*: size of letter

**Returns:**

array of lines and letters

public static **System.Drawing.Bitmap** **WhiteBitmap**(**int** *width***, int** *height*)

**Summary:**

Get width and height and return white Bitmap in that size

**Parameters:**

*width*: width of Bitmap

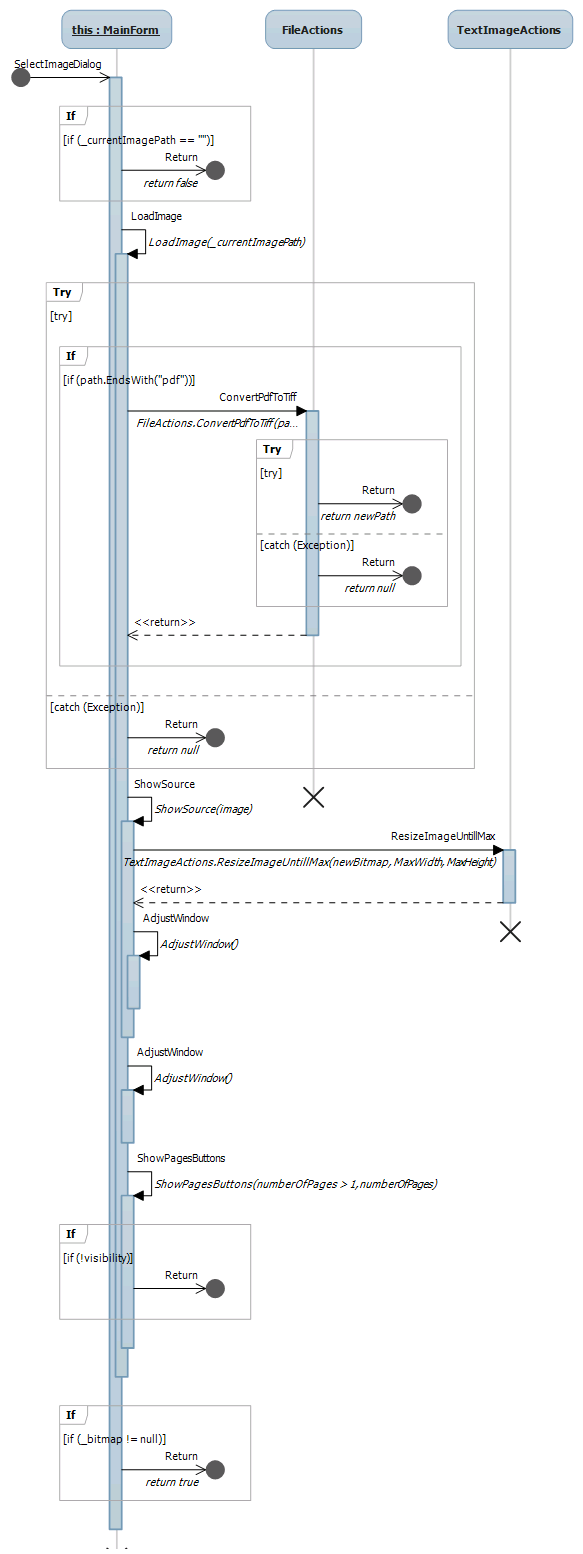
*height*: height of Bitmap

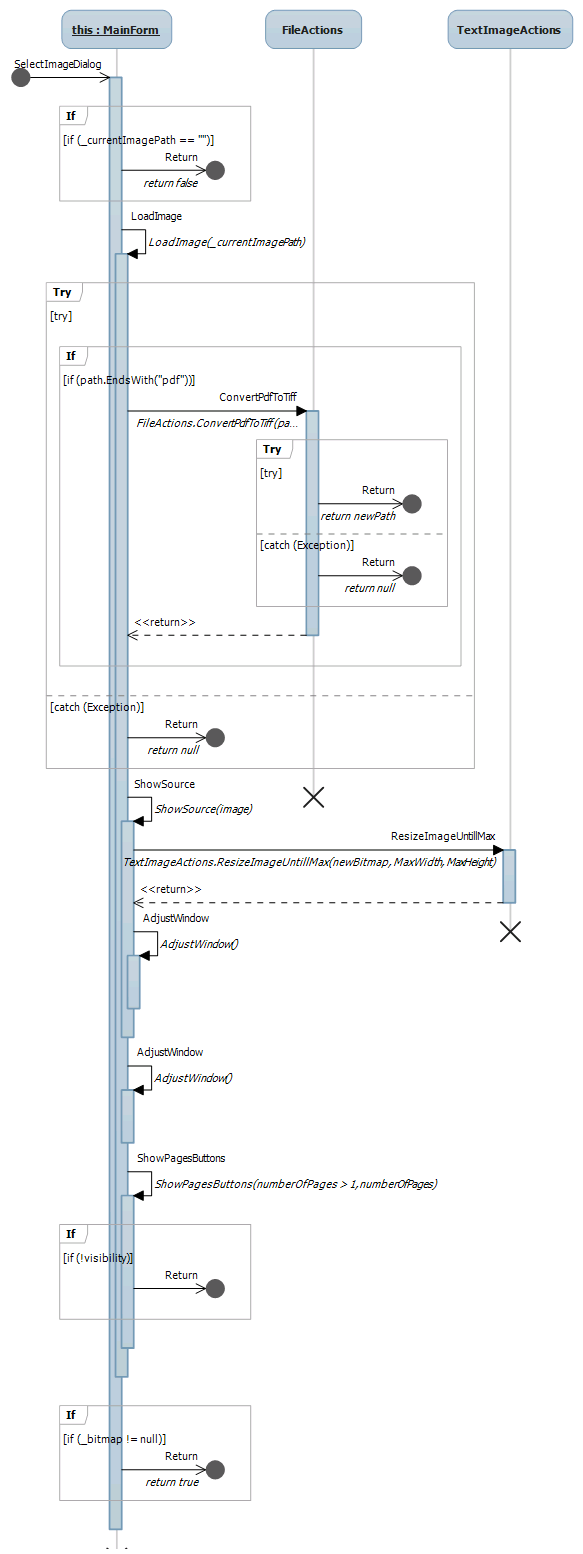
**Returns:**

new white Bitmap

# Sequence Diagrams

## בחירת תמונה (תמונה או קובץ PDF)





## אימון רשת ניורונים

