ANNdotNET- for developers

deep learning open source project

Contents

[Introduction 3](#_Toc524979423)

[Structure of the project 5](#_Toc524979424)

[System requirements 6](#_Toc524979425)

[How to start with ANNdotNET CMD Tool 6](#_Toc524979426)

[How to start with ANNdotNET GUI Tool 8](#_Toc524979427)

[How to run ANNdotNET GUI Tool from release section 8](#_Toc524979428)

[How to run ANNdotNET GUI Tool from source code 9](#_Toc524979429)

[Components of ANNdotNET solution 11](#_Toc524979430)

[nnetwork.core project 12](#_Toc524979431)

[ProgressData 12](#_Toc524979432)

[mlfactorylib.network.extension 13](#_Toc524979433)

[anndotnet.core.net 13](#_Toc524979434)

[MinibatchSourceEx 13](#_Toc524979435)

[MLTrainer 14](#_Toc524979436)

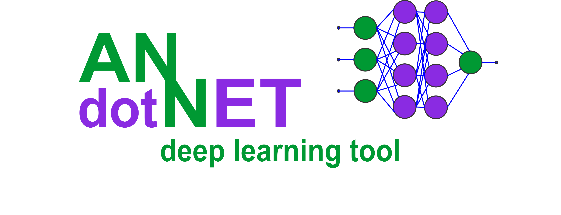
[MLFactory 14](#_Toc524979437)

[Machine Learning Configuration File (mlconfig) 16](#_Toc524979438)

[ANNdotNET Tool 16](#_Toc524979439)

[MachineLearning class 16](#_Toc524979440)

# Introduction



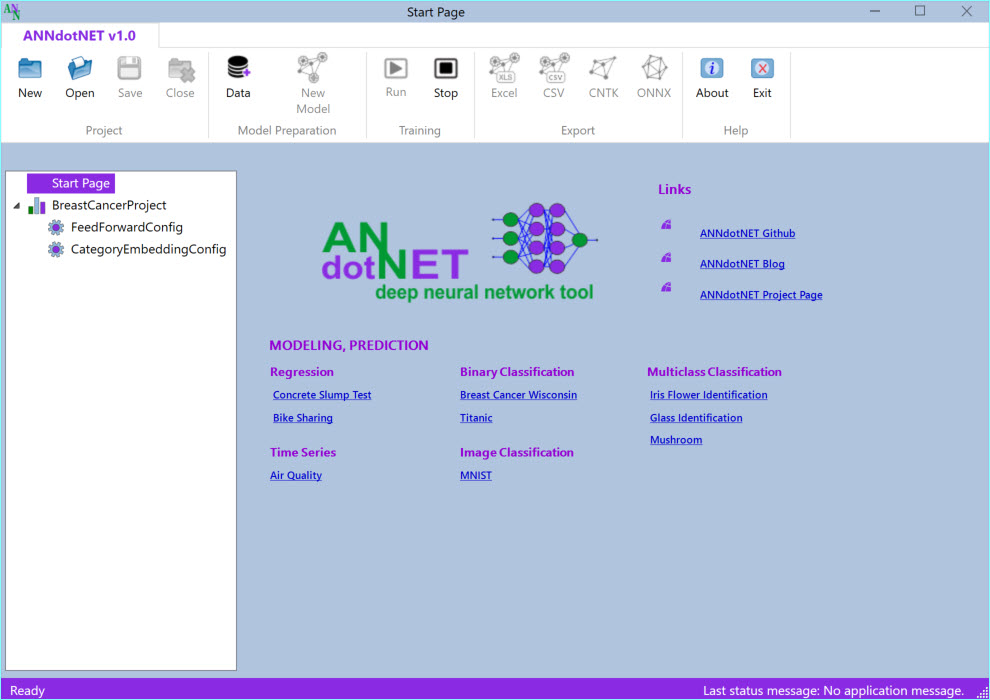
ANNdotNET – is an open source project for deep learning. The project is written in C# on .NET Core platform, except GUI Tool which is written in .NET Framework. The project can be run as standard Windows Desktop application, but also provides API and infrastructure to run as a Console Application in .NET Core.

The project is based on Microsoft CNTK (CogNitive ToolKit) Microsoft open source library for deep learning. It is supposed to be higher API for deep learning in .NET, but also provides, data preparation and transformation from rawDataSet in to mlreadyDataset, monitoring the training process with additional evaluation functions, capability of early stopping during training, model evaluation and validation, exporting and deployment options.

The process of creating, training, evaluating and exporting models is provided from the GUI Application and does not require knowledge for supported programming languages. The ANNdotNET is ideal for engineers which are not familiar with programming languages. There are dozens of pre-calculated projects included in the installer which can be opened from the Start page as well as from Console tool. The projects are based on famous datasets from several categories: regression, binary and multiclass classification problems, image classifications, times series, ….

In pre-calculated projects the user can find how to use various neural network configurations e.g. feed forward, deep neural network, LSTM recurrent nets, embedding and drop out layers. Also, each project can be modified in terms of change its network configuration, learning and training parameters, as well as create new ml configuration.

In order to change ml configuration, ANNdotNET implement***s network designer*** capable of creating neural network of any configuration, and any combination of layers. The network designer is based on layer concept which the user can add, delete or modify simply as list view items.



Note: The application automatically detects GPU capability on your machine and use it in training and evaluation, otherwise it will use CPU.

## Structure of the project

The ANNdotNET open source project is Visual Studio based solution, contains several projects grouped into logical solution folders. In order to build the solution, the user need at least Visual Studio 2017 Community version which can be freely downloaded.

ANNdotNET solution can be grouped on several components:

* The library
* CMD Tool
* GUI Tool
* Excel AddIn
* Unit Tests and Test applications

**The library** consists of several dlls which logically separate the implementation. It provides foundation of data processing and preparation, neural network configuration and implementation of neural network layers, training and handling with minibatches. The library also provides API for the model evaluation, testing, export and deployment.

**CMD Tool** is console-based tool, which can be run from Visual Studio, and can perform mlconfig handling, training and evaluation of models.

**GUI Tool** is Windows Desktop application which provides rich set of options and visualizations during machine learning steps: project and mlconfig files creation, data preparation, model training, evaluation and validation, export options and model deployment. Currently only WPF based GUI Tool is implemented. Easily it can be implemented ASP.NET or another web based or desktop-based GUI.

**Excel AddIn** is implementation of Microsoft Office AddIn for model deployment in to Excel. Using ANNdotNET Excel AddIn, trained model can be used in Excel like ordinary excel formula. This is very handy for model deployment into production when only Excel is need in order to used the model.

**Unit Tests** – set of unit test and console projects which test implementation of the library.

## System requirements

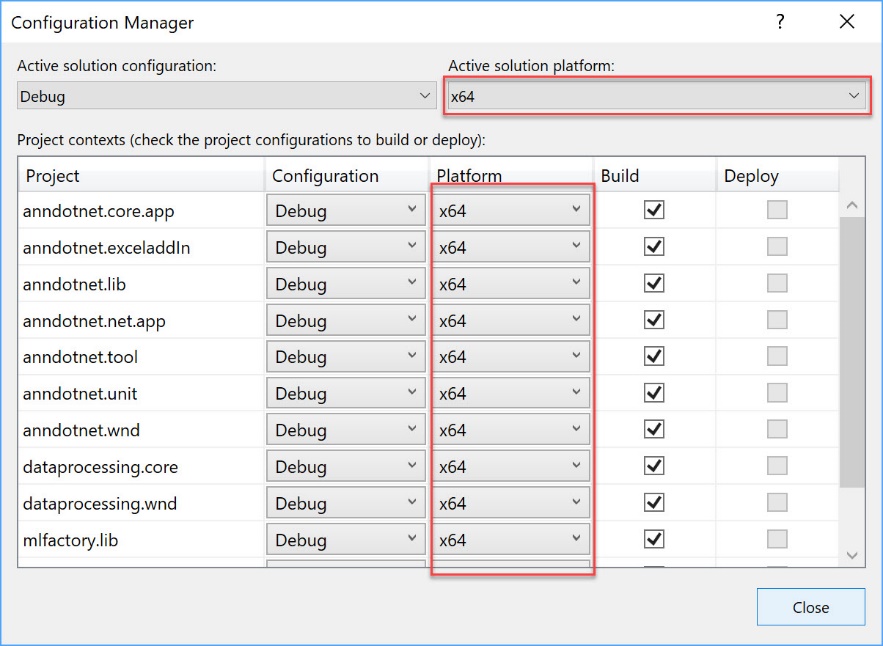
In order to run and develop ANNdotNET based solution the following system requirements must be met:

* Windows 8 x64 or higher,
* Visual Studio 2017 (Community, Professional or Enterprise),
* .NET Framework 4.7.2 and newer,
* .NET Core 2.0 and newer
* Git source control tool.

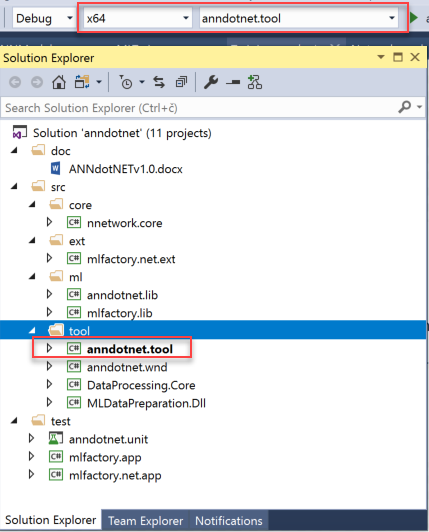
## How to start with ANNdotNET CMD Tool

In order to compile and build ANNdotNET solution, the following actions must be performed:

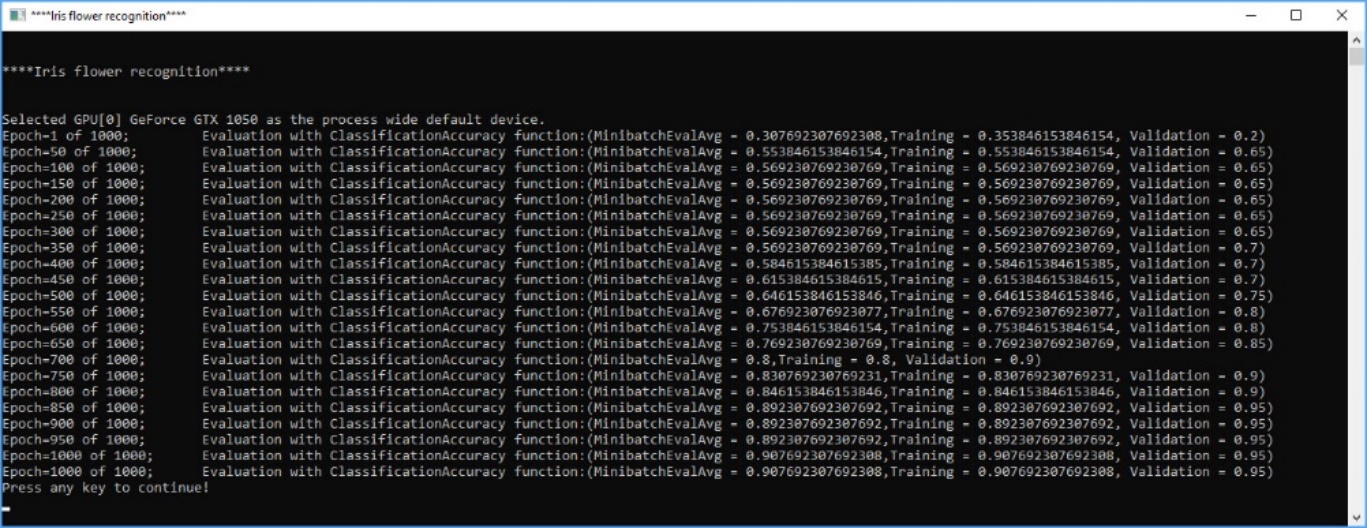
1. Clone solution by typing: git clone https://github.com/bhrnjica/anndotnet
2. Run Visual Studio 2017 and open anndotnet.sln solution file.
3. Set anndotnet.tool – as startup project
4. Change system architecture of the solution from Any CPU to x64



1. Once the previous actions are performed the Solution window looks like:



1. From the Program.cs file select one of several implemented solutions uncomment the code and run the solution.
2. Once the project is executed the following output is present:



The output Window start by problem title ( in this case “Iris Flower Identification”), then depending on training parameters the training progress is written in the console window. Once the training process is finished, the best model is selected and evaluated against test dataset. The result of evaluation is stored in csv file format which can be easily opened for further analysis.

## How to start with ANNdotNET GUI Tool

In order to start ANNdotNET GUI Tool, the user has two choices:

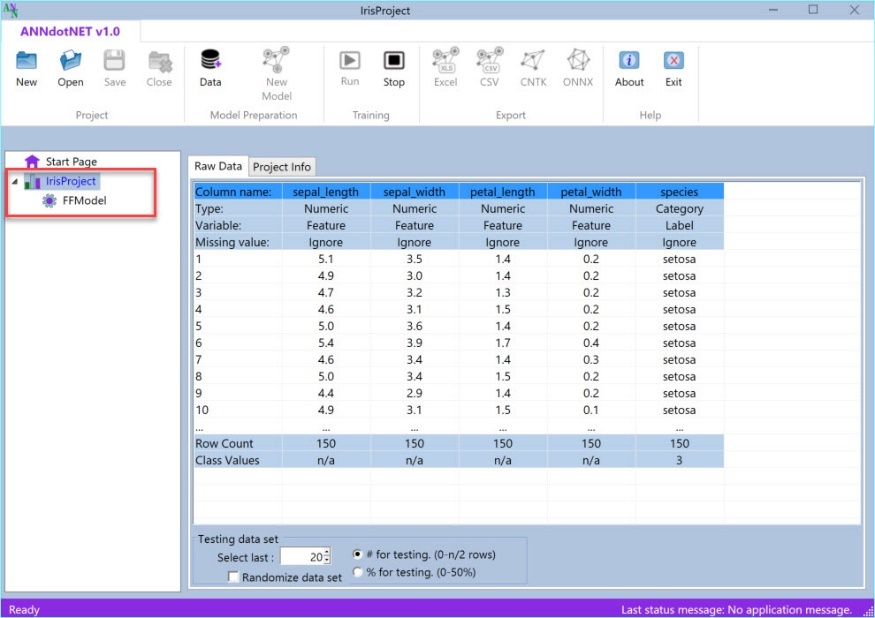
1. Download stable version of ANNdotNET GUI Tool from Release section, or
2. Compile and run the project from Visual Studio

### How to run ANNdotNET GUI Tool from release section

This option is handy in case you don't have installed Visual Studio or you want to use the application without source code. The following actions should be performed:

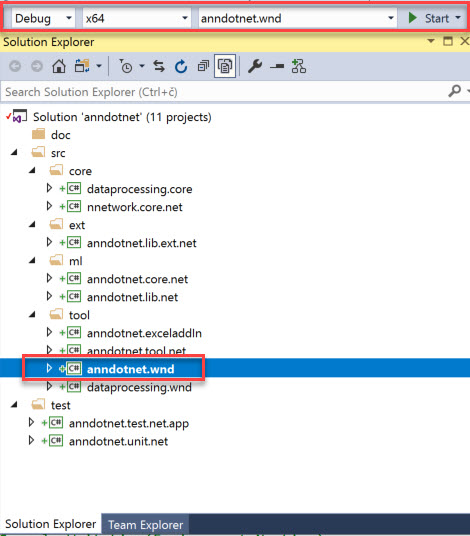
* Download binaries from the release section at: https://github.com/bhrnjica/anndotnet/releases,
* Unzip the binaries on your machine and run `anndotnet.wnd.exe` exe file.
* Once the application is run, select one of many pre-calculated projects placed on Start Page.

The following image shows Iris Flower Identification project opened in ANNdotNET GUI Tool



### How to run ANNdotNET GUI Tool from source code

The second option is similar as in case of ANNdotNET CMD Tool. Only difference is the startup project which in this case should be `anndotnet.wnd`, and it is shown on the image below.



To recap the section, once the user download the source code and open it by using Visual Studio, the **anndotnet.wnd** should be selected as startup project. Beside startup project, user also should change the system architecture to x64, and press F5. Once the Visual Studio compiles and builds the solution, the ANNdotNET windows desktop application is shown on the user’s machine.

# Components of ANNdotNET solution

ANNdotNET solution consists of several projects and components. All projects in the solution ends with three suffixes:

* .core or .lib - those project is configured for .NET Core Framework
* .net - project are configured for .NET Framework, and it is fully compatible for .NET Core.
* .wnd – those project are configured for .NET Framework, and contains WinForms or WPF components.

The reason of having two project version for the same code is, since in time this manual is written, the latest release of CNTK supports .NET Framework only. Once the CNTK would supports the .NET Core, ANNdotNET will support it as well. doc – folder contains documentation for the solution.

1. nnnetwork.core.net(nnnetwork.core) – implement core concept of the CNTK neural network by providing higher API, to create various networks configuration, learning and training parameters.
2. dataprocessing.core and dataprocessing.wnd projects are related for data preparation. They transform the raw dataset into ml-ready dataset.
3. anndotnet.lib.ext.net(anndotnet.lib.ext)-extension component which can be used for custom network implementation, specific to the problem. Sometimes network designer of ANNdotNET cannot design the network, so in that situation the user can implement manualy using ml-configuration file only. Usually the extension is used when custom neural network model is implemented.
4. anndotnet.core.net (anndotnet.core) – component consist of specific implementation, where the main concept relies on machine learning configuration file (mlconfig file).
5. anndotnet.lib.net (anndotnet.lib) – component consist of specific implementation related for handling project and multiple mlfonfiguration file within the same project. The component is mostly related for Windows Desktop application.
6. anndotnet.app – client app which provides implementation how to use anndotnet using console app, and command line. The ANNdotNET Tool is supposed to be cross OS tool which can be run on any OS .NET Core supports.
7. anndotnet.wnd windows desktop application which provides rich set of option to training, validate and evaluate mlconfigurations.
8. anndotnet.unitest and related test application - vs projects consist of unit test methods and testing code of the solution.

# nnetwork.core project

nnetwork.core project contains CNTK based API for creating neural network. Furthermore, the project consists of:

* Feedforward network implementation
* LSTM Recurrent network implementation
* NNParameters class which implements NN parameters needed for handling network model
* StatMetrics – set of custom defined CNTK function for evaluate and train nn models.
* ProgressData class implementation

## ProgressData

The ProgressData contains properties which is passed by progress writer method back to the client, in order to show the current state of the trainer. The ProgressData class contains the following information:

* EpochTotal – total number of epoch, defined in the ML configuration file.
* EpochCurrent – current epoch, which trainer has been processed,
* EvaluationFunName- the name of the Evaluation function, defined by the learning parameters, in the ml config file.
* MinibatchAveragegeEval and MinibatchAverageLoss – evaluation and loss values of the current processed minibatch.
* TrainEval– value returned the evaluation function against whole training dataset during model testing,
* ValidationEval – value returned the evaluation function against whole training dataset during model testing,

# mlfactorylib.network.extension

This project extends the implementation of anndotnet.cor.net project in case the current implementation, and ML configuration file cannot implemented desired neural network model. Typical example of such solution are “Predict Future Sales” which implements custom NN model. The example of the custom model can be found in the project. The extension project uses the network.core.net project, but it is not dependent of the anndotnet.core.net. With such dependency we can implement custom neural network model, referenced it with the client app (ASP.NET or other Application) and use it as ordinary ML solution by providing the ml configuration file and custom neural network implementation.

# anndotnet.core.net

anndotnet.core.net is central project of the ANNdotNET solution, since it implements all machine learning steps. The project contains the following class implementations:

* MachineLearning class implementation,
* MLFactory class implementation,
* MLTrainer class implementation,
* MinibatchSourceEx class implementation.

## MinibatchSourceEx

The class implements the unique way of feeding the trainer. It extends the current TextFormatMinibatchSource CNTK implementation of txt file-based reader for feeding the trainer. Currently there are two types of minibatch:

* default - which is based on default CNTK TextFileMinibachSource,
* custom – which is based on custom implemented MinibachSource for sequence of variables length which is not supported by TextFileMinibachSource CNTK minibatch.

The MinibatchSourceEx contains methods:

* Constructor – for creation the object,
* GetNextMinibatch – common method for the trainer. It returns chunk of data specified by the MinibatchSize values.
* GetFullBatch – returns whole dataset for validation and testing,
* NormalizeInput – methods for input normalization and creation Normalization layer.

## MLTrainer

Thet MLTrainer class implements only two public methods:

* CreateTrainer – the method searches previous checkpoint state of the trainer in case the user want to continue training from the previous try. Otherwise, create brand new trainer by using current training and learning parameters.
* Train – method starts the training process. During training, trainer sends the training progress as the ProgressData object so the client knows what is going on with training. Once the training process is completed, it returns the TrainResult object which contains the information about:
* ProcessState - how the training process ended (completed, stopped, crashed)
* Iteration - the last iteration before trainer ends,
* BestModel - file path of the best-found model.

## MLFactory

The class implements basic actions for creating all machine learning components:

* Loading ml configuration files into memory,
* Streams needed for defining input and output variables,
* Network models,
* Learning parameters,
* Training parameters
* Other ML configuration files options.

MLFactory class defines set of static methods which are independent form each other, since it performs unique task. At the beginning MLFactory class object must be created:

//create factory object

MLFactory f = new MLFactory();

Once the object is created, the method CrateIOVariables can be called in order to create input and output variables which are needed for further ML tasks.

//extract features and label strings and create CNTK input variables

var strFeatures = dicMParameters["features"];

var strLabels = dicMParameters["labels"];

//create config streams

f.CreateIOVariables(strFeatures, strLabels, DataType.Float);

The method requires two string which define input and output variables which are parsed and created within the method.

Once the Input and Output variables are created the rest of the ML steps are straight forward.

# Machine Learning Configuration File (mlconfig)

# ANNdotNET Tool

## MachineLearning class

The MachineLearning class is statically defined class with two methods:

* Run – for run training process, which is previously described.
* EvaluateModel for model evaluation and exporting the results.

EvaluateModel is called by providing the arguments about input and output variables, MinibatchSource, path of the best model, path of the dataset, and path where the result will be stored.