10 Writeln 'Hi Al';
20 Goto 10;

ML Community Edition

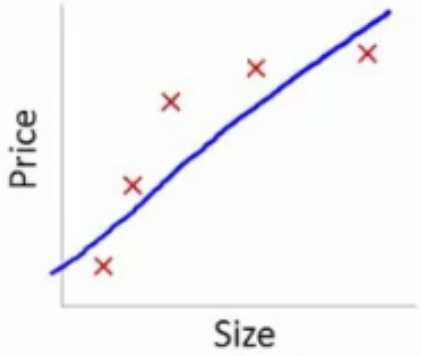
EKON 24

maXbox

Agenda EKON 24

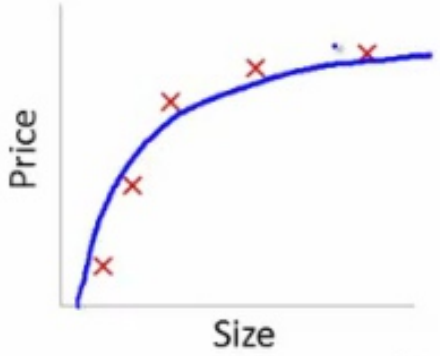
- Tensorflow64 Library
- FANN (Fast Artificial Neural Network)
- CAI NEURAL API
- K-CAI NEURAL API (Jupyter Notebook)
- Installation, Optimisation & Sources

Machine Learning



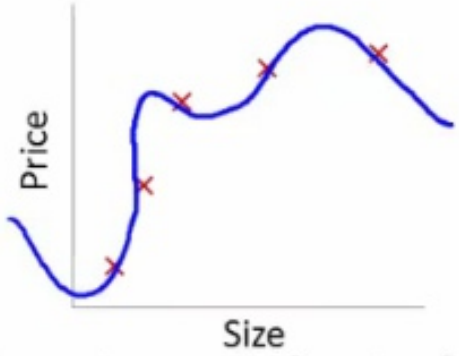
$\theta_0 + \theta_1 x$

High bias
(underfit)



$\theta_0 + \theta_1 x + \theta_2 x^2$

"Just right"

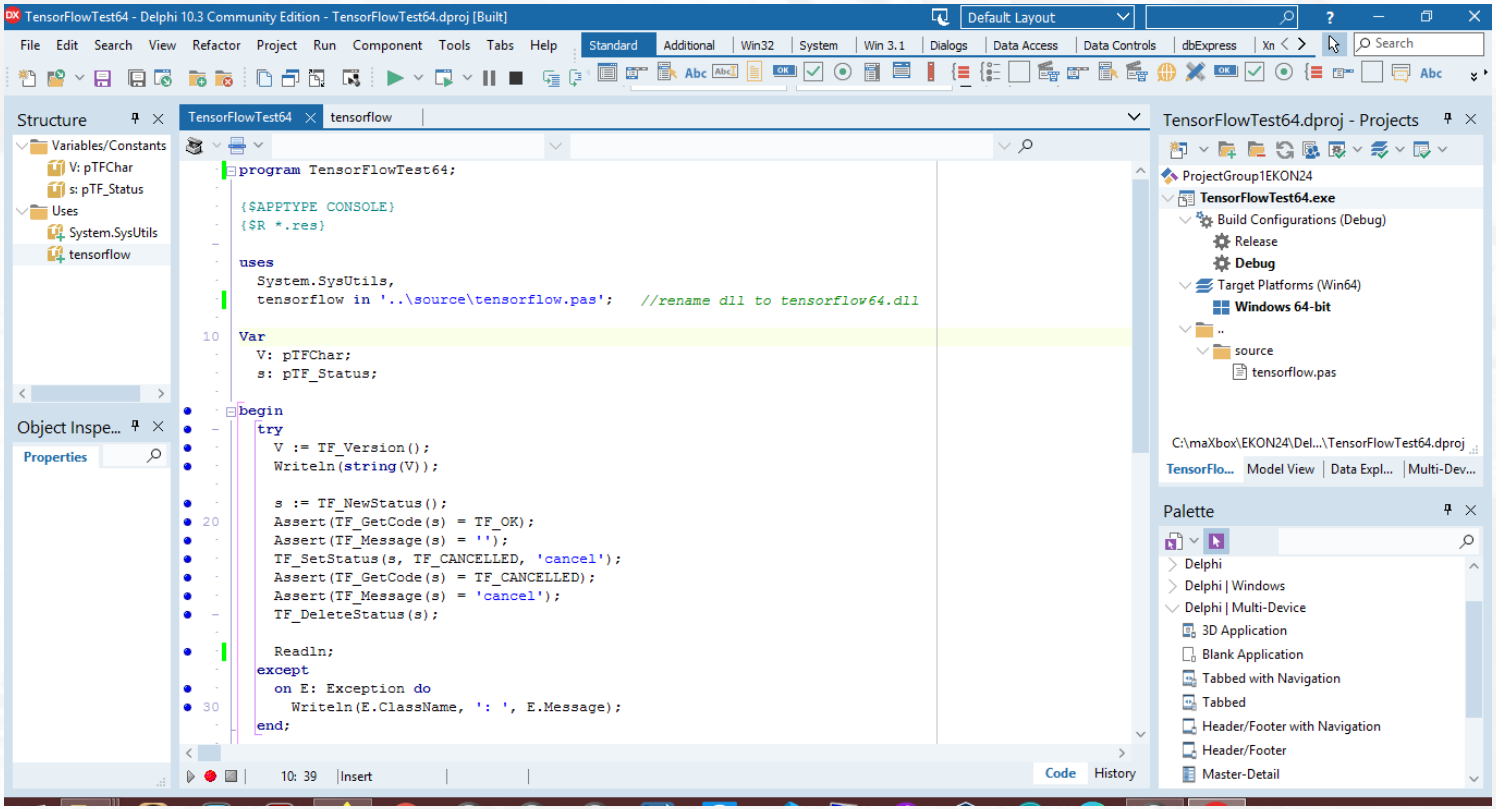


$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

High variance
(overfit)

http://docs.codehaus.org/display/SONAR/Plugin+Library

Develop for Multiple Devices



https://www.tensorflow.org/

EKON 24

maXbox

```
unit tensorflow;

interface

Const
  tensorflow_dll = 'tensorflow64.dll';

Type
  pTFChar = PAnsiChar;
  ppTFChar = ^pTFChar;
  size_t = NativeUInt;
  int64_t = Int64;
  pint64_t = ^int64_t;
  ppint64_t = ^pint64_t;
  psize_t = ^size_t;
  TEnumType = Cardinal;
  Float = Single;
  pFloat = ^Float;
```

Now Demo: TensorFlowTest64.dproj

FANN

- Fast Artificial Neural Network (FANN) Library is a free open source neural network library, which implements multilayer artificial neural networks in C with support for both fully connected and sparsely connected networks.

- Cross-platform execution in both fixed and floating point are supported. It includes a framework for easy handling of training data sets. It is easy to use, versatile, well documented, and fast.
- Bindings to more than 15 programming languages are available.
- https://github.com/libfann/fann

Delphi Wrapper

- Fast Artificial Neural Network Library (FANN) v2.2.0 (Original: https://github.com/libfann/fann)
- Fast Artificial Neural Network Library (FANN) v2.2.0 (https://github.com/hatsunearu/fann with FANN_RELU and FANN_LEAKY_RELU)
- TensorFlow 1.3.0 → Demo
-

https://github.com/Laex/Delphi-Artificial-Neural-Network-Library

-

FANN Scripting

```
NN:= TFannNetwork.create(self)
  with NN do begin
    Layers.add('2')
    Layers.add('3')
    Layers.add('1')
    LearningRate:= 0.69999998807907100
    ConnectionRate:= 1.000
    TrainingAlgorithm:= taFANN_TRAIN_RPROP
    ActivationFunctionHidden:= afFANN_SIGMOID
    ActivationFunctionOutput:= afFANN_SIGMOID
  end;
```

C:\maXbox\EKON24\examples\814_FANN_XorSample2.pas

CAI NEURAL API

- K-CAI NEURAL API is a Keras based neural network API for machine learning that will allow you to prototype with a lots of possibilities of Tensorflow! Python, Free Pascal and Delphi together in Google Colab, Git or the Community Edition.

- https://github.com/joaopauloschuler/neural-api

CAI NEURAL API II

- CAI NEURAL API is a pascal based neural network API optimized for AVX, AVX2 and AVX512 instruction sets plus OpenCL capable devices including AMD, Intel and NVIDIA. This API has been tested under Windows and Linux.
- This project is a subproject from a bigger and older project called CAI and is sister to Keras based K-CAI NEURAL API.

https://github.com/joaopauloschuler/neural-api

Colab as Universal Platform

Simple Image Classification with any Dataset:

this example shows how to create a model and train it with a dataset (samples and features) passed as parameter. Open In Colab

https://colab.research.google.com/github/maxkleiner/maXbox/blob/master/Copy_of_simple_image_classification_with_any_dataset.ipynb

https://colab.research.google.com/github/maxkleiner/maXbox/blob/master/Copy_of_simple_image_classification_with_any_dataset.ipynb

CIFAR-10 Image Classifier

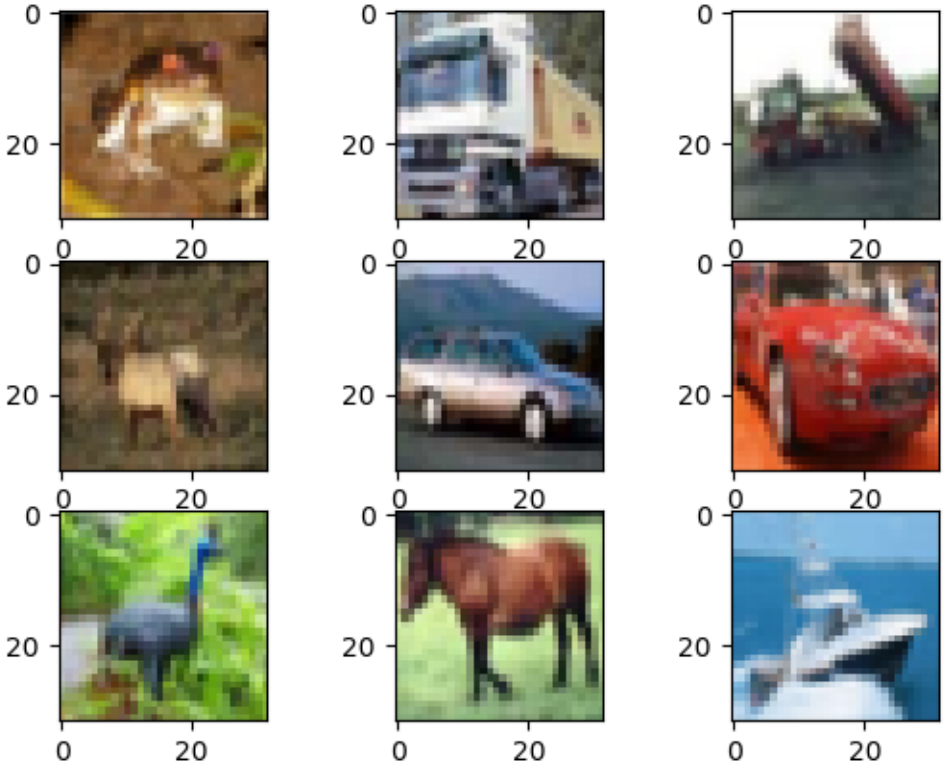
This example has interesting aspects to look at:

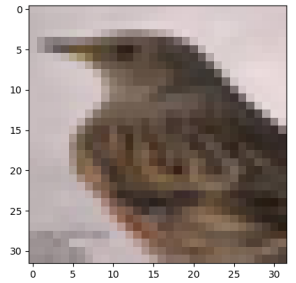
- Its source code is very small.
- Layers are added sequentially.
- Training hyper-parameters are defined before calling fit method.
- Model parameters are saved as hdf5 EKONSimpleImageClassifier.nn

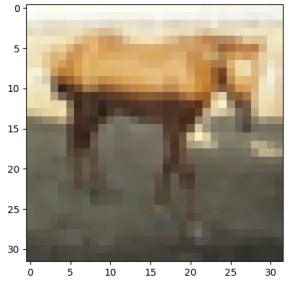
https://github.com/maxkleiner/maXbox/blob/master/EKON24_SimpleImageClassificationCPU.ipynb

and the same in colab.research:

https://colab.research.google.com/github/maxkleiner/maXbox/blob/master/EKON24_SimpleImageClassificationCPU.ipynb



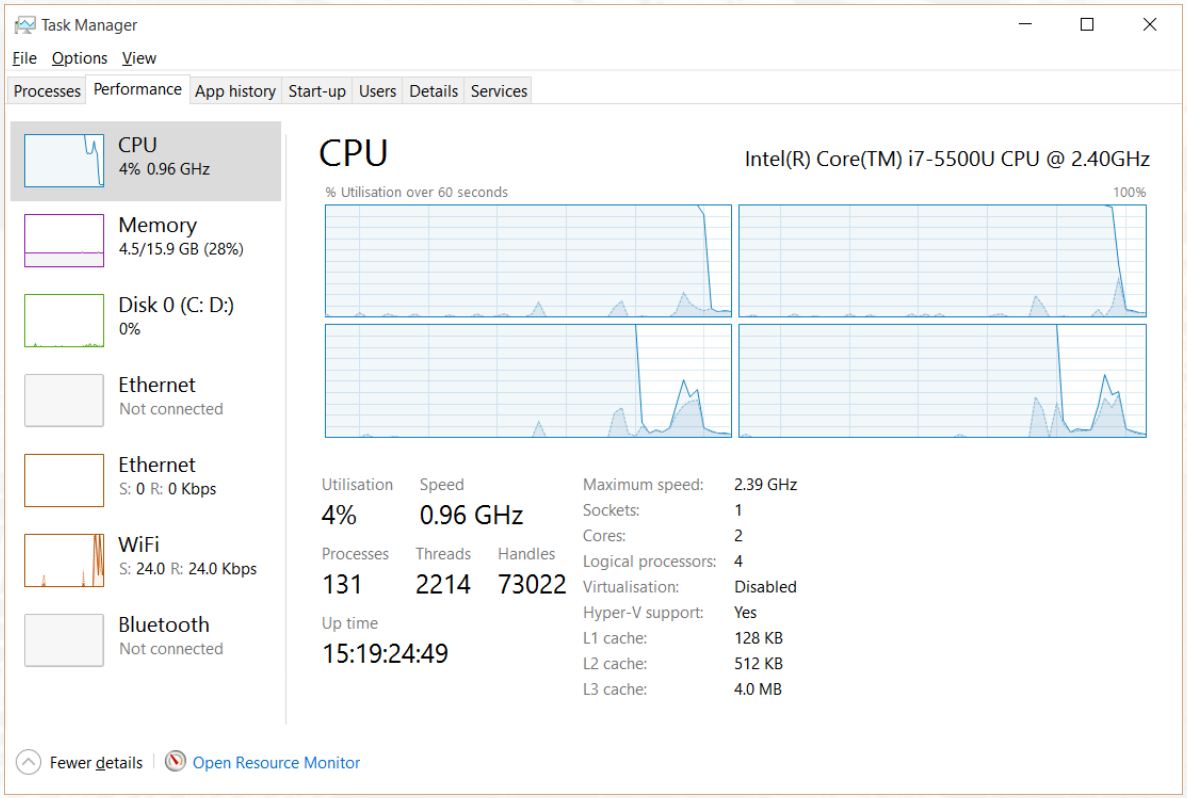




```
classes=('plane','car','bird','cat',
         'deer','dog','frog','horse','ship','truck')
```

https://colab.research.google.com/github/maxkleiner/maXbox/blob/master/EKON24_SimpleImageClassificationCPU.ipynb

Win 10 Core Optimisation



CPU or GPU or TPU?

Win 32 or 64 API ?

C:\maXbox\EKON24\Delphi-Artificial-NN-Library-master\neural-api-master\ne...

```
rward time:  4.66s Backward time:  1.00s Step time: 14.09s
Starting Validation.
VALIDATION RECORD! Saving NN at EKONSimpleImageClassifier.nn
Epochs: 1 Examples seen:40000 Validation Accuracy: 0.4036 Validation Error: 1.46
46 Validation Loss: 1.6349 Total time:  12.54min
Epoch time: 9.2 minutes. 100 epochs: 15 hours.
Epochs: 1. Working time: 0.21 hours.
Finished.
```

C:\maXbox\EKON24\Delphi-Artificial-NN-Library-master\neural-api-master\ne...

```
18560 Examples seen. Accuracy:0.2835 Error:   1.59489 Loss:1.85568 Threads: 2 Fo
rward time:  1.83s Backward time:  0.24s Step time: 16.11s
Starting Validation.
VALIDATION RECORD! Saving NN at EKONSimpleImageClassifier.nn
Epochs: 1 Examples seen:40000 Validation Accuracy: 0.2500 Validation Error: 1.65
98 Validation Loss: 1.9632 Total time:   8.75min
Epoch time: 10 minutes. 100 epochs: 17 hours.
Epochs: 1. Working time: 0.15 hours.
Finished.
```

EKON 24

maXbox

Save the model

- Keras separates the concerns of saving your model architecture and saving your model weights.
- Model weights are saved to HDF5 format. This is a grid format that is ideal for storing multi-dimensional arrays of numbers.
-

•Layer 11 Max Output: 0.812 Min Output: 0.000 TNNetSoftMax 10,1,1 Times: 0.00s 0.00s Parent:10
•Starting Testing.
•Epochs: 50 Examples seen:2000000 Test Accuracy: 0.8383 Test Error: 0.4463 Test Loss: 0.4969 Total time: 162.32min
•Epoch time: 2.7 minutes. 100 epochs: 4.5 hours.
•Epochs: 50. **Working time: 2.71 hours.**

•Finished.

Save files local

- import os
- import urllib.request
- import tarfile
-
- if not os.path.isfile('cifar-10-batches-bin/data_batch_1.bin'):
- print("Downloading CIFAR-10 Files")
- url = 'https://www.cs.toronto.edu/~kriz/cifar-10-binary.tar.gz'
- urllib.request.urlretrieve(url, './file.tar')
-
- tar = tarfile.open("file.tar")
- tar.extractall()
- tar.close()

Finally you can get with git

- C:\maXbox\mX399100\maxbox4\maxbox47\maxbox4>git clone https://github.com/joaopauloschuler/k-neural-api.git k
-
- Cloning into 'k'...
- remote: Enumerating objects: 65, done.
- remote: Counting objects: 100% (65/65), done.
- remote: Compressing objects: 100% (43/43), done.
- remote: Total 356 (delta 38), reused 38 (delta 18), pack-reused 291
- Receiving objects: 100% (356/356), 224.47 KiB | 1.57 MiB/s, done.
- Resolving deltas: 100% (225/225), done.

CAIProject12 - Delphi 10.3 Community Edition - SimplePlantLeafDisease [Built]

Default Layout

File Edit Search View Refactor Project Run Component Tools Tabs Help | Standard Additional Win32 System Win 3.1 Dialogs Data Access Data Controls dbExpress Xn | Search

Structure

- Procedures
- Uses
  - Classes
  - SysUtils
  - neuralnetwork
  - neuralvolume
  - Math
  - neuraldatasets
  - neuralfit

Object Inspector

Properties

Project12 | SimplePlantLeafDisease | neuraldatasets | neuralfit | CL_GL | CL_Platform | DelphiCL | neuralnetwork

TTestCifar10Algo

```
procedure TTestCifar10Algo;
var
  NN: TNNet;
  NeuralFit: TNeuralImageFit;
  ImgTrainingVolumes, ImgValidationVolumes, ImgTestVolumes: TNNetVolumeList;
  NumClasses: integer;
  fLearningRate, fInertia: single;

begin
//This is how a sequential CNN array of layers is added:

  NN := TNNet.Create();
  NumClasses:= 10;
  fLearningRate := 0.001;
  fInertia := 0.9;
NN.AddLayer(TNNetInput.Create(32, 32, 3)); //32x32x3 Input Image
NN.AddLayer(TNNetConvolutionReLU.Create({Features=}16, {FeatureSize=}5, {Padding=}0, {Stride=}1,
NN.AddLayer(TNNetMaxPool.Create({Size=}2));
NN.AddLayer(TNNetConvolutionReLU.Create({Features=}32, {FeatureSize=}5, {Padding=}0, {Stride=}1,
NN.AddLayer(TNNetMaxPool.Create({Size=}2));
NN.AddLayer(TNNetConvolutionReLU.Create({Features=}32, {FeatureSize=}5, {Padding=}0, {Stride=}1,
NN.AddLayer(TNNetLayerFullConnectReLU.Create({Neurons=}32));
NN.AddLayer(TNNetFullConnectLinear.Create(NumClasses));
NN.AddLayer(TNNetSoftMax.Create());
```

164: 43 | Insert | Code | History

CAIProject12.dproj - Projects

- ProjectGroup1EKON24
- TensorFlowTest64.exe
- PrimeFactors1_64.exe
- XorConsole.exe
- cascade_train.exe
- **CAIProject12.exe**
  - Build Configurations (Debug)
  - Target Platforms (Win32)
  - ..\..
    - neural
      - CL.pas
      - CLExt.pas
      - CL_D3D10.pas
      - CL_D3D11.pas
      - CL_D3D9.pas
      - CL_DX9_Media_Sharing.pas
      - CL_Ext.pas
      - CL_GL.pas
      - CL_GL_Ext.pas
      - CL_Platform.pas

C:\maXbox\EKON24\Delphi-Artific...\neural

CAIProj... | Model V... | Data Ex... | Multi-D...

Messages

[dcc32 Warning] SimplePlantLeafDisease.pas(265): W1011 Text after final 'END.' - ignored by compiler

Success

Elapsed time: 00:00:02.2

Build | Output

EKON 24

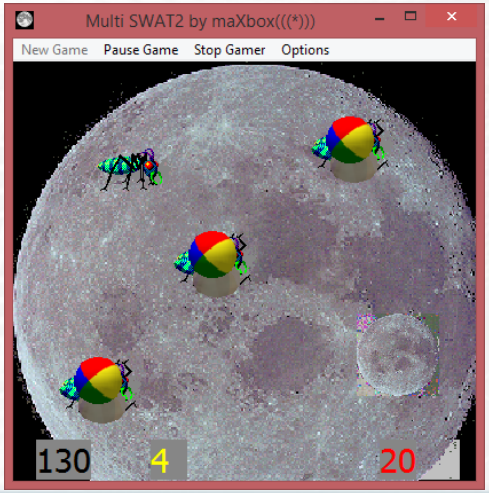
maXbox

Links & Sources

- Almost all files:
-
- https://sourceforge.net/projects/maxbox/files/Examples/EKON/EKON24/
-
- https://maxbox4.wordpress.com/blog/
-
- https://github.com/maxkleiner/maXbox4/releases
-

May the source be with you!

max@kleiner.com
www.softwareschule.ch



EKON 24

maXbox