//

Machine Learning with CAI

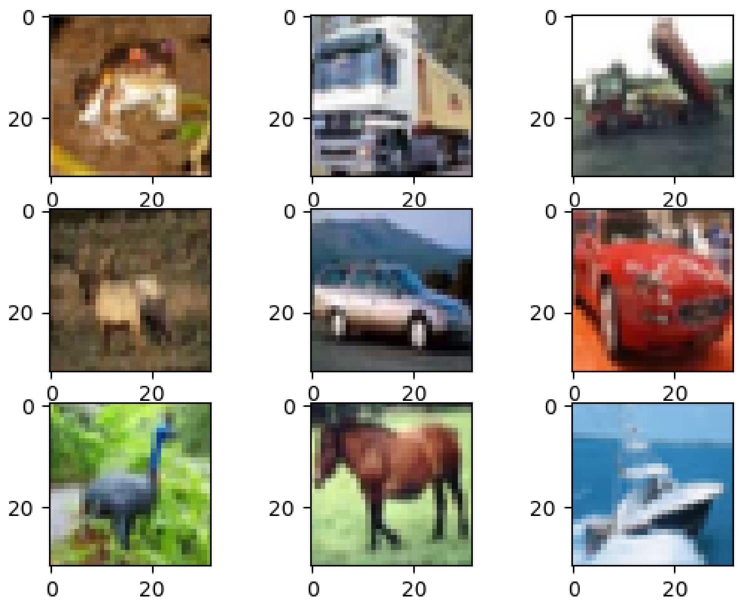
maXbox Starter 76 – Image Classifier

From Document to Real World Recognition

This tutor explains a trip to the kingdom of object recognition with computer vision knowledge and an image classifier from the CAI framework in Lazarus and Delphi, the so called CIFAR-10 Image Classifier.

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 for GPU capabilities. This API has been tested under Windows and Linux.
On January this year we got in Delphi support for OpenCL and Threads.
This project and API is a sub-project from a bigger and older project called CAI and its sister to Keras/TensorFlow based K-CAI NEURAL API.

Image detection has been witnessing a rapid revolutionary change in some fields of computer vision. Its involvement in the combination of object classification as well as object recognition makes it one of the most challenging topics in the domain of machine learning & vision.



First we need the library with modules. **Neural-API** is a Pascal library built to empower developers to build and run applications and systems with self-contained deep learning and Computer Vision capabilities using a few lines of straight forward code. You'll need Lazarus or Delphi development environment. If you have an OpenCL capable device, you'll need its OpenCL GPU drivers. But to use CAI first you need to install a few dependencies as units namely:

- dglOpenGL
- OpenCV as CL or OpenCL
- CL_Platform

- Neuralvolume, neuralnetwork, neuralab, etc., itself to install with Lazarus and with the help of Git so clone this project, add the neural folder to your Lazarus unit search path and you'll be almost ready to go!

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

After installing CAI, you can find documentation in a readme.
Concerning Delphi a number of units do compile with Delphi and you can create and run neural networks with Delphi or in my case with the community edition 10.3 see picture below. You'll be able to compile these units with Delphi: neuralvolume, neuralnetwork, neuralab, neuralabfun, neuralbit, neuralbyteprediction, neuralcache, neuraldatasets, neuralgeneric, neuralplanbuilder and neuralfit.
For Keras and tensorflow you get it also with git:

git clone https://github.com/joaopauloschuler/k-neural-api.git

Another fascinating way is to run the whole system on google colab or colab.research container with the help of a Jupyter notebook running on Ubuntu in the cloud including the build of Free Pascal with Lazarus as I did and tested too:

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

```
!apt-get install fpc fpc-source lazarus git subversion
```

Jupyter is a spin-off project from IPython, aiming to standardize interactive computing in any programming languages. The kernel provides interactive environment that executes user code as a server, connected with a frontend through socket. Those who needs rapid prototyping or quick coding might be the best target users. I don’t see any reason why Pascal or Free Pascal should implement one, but it’s up to those who want to implement, just like in other languages.

Now download the CIFAR model file (162 MB) that contains 5 volumes for the classification model that will be used for object training and detection:

https://www.cs.toronto.edu/~kriz/cifar-10-binary.tar.gz

```
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'
```

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

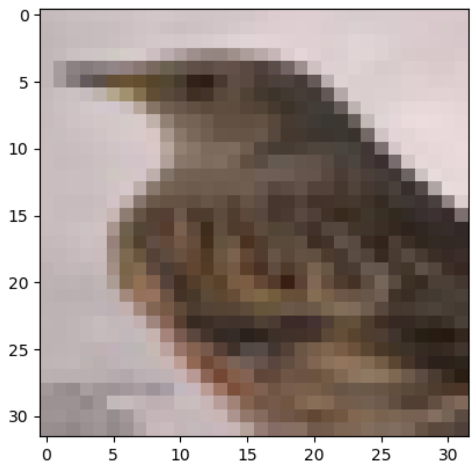
The dataset is divided into five training batches (data_batch_1.bin -5) and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

Then we need 3 necessary folders or files:
- neural\
- neural\data_batch_1.bin - data_batch_5.bin
- neural\test_batch.bin

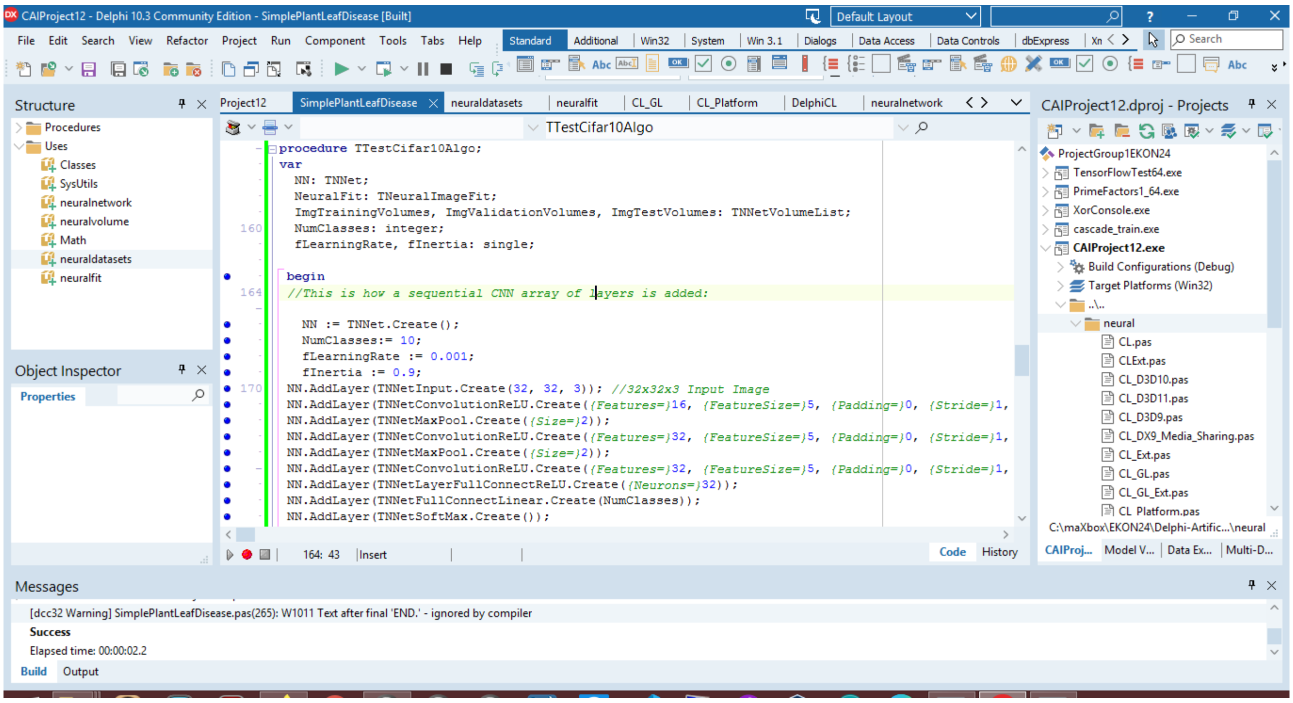
The classes are completely mutually exclusive. There is no overlap between automobiles and trucks or cat and dog. The classes are:

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

These results were obtained with a convolutional neural network. Briefly, they are 18% test error without data augmentation and 11% with. A single image has a low resolution for training the data features:



Open now your preferred code editor for writing Pascal code (in my case maXbox and Delphi Community Edition) and load the file *examples/SimpleImageClassifier/SimpleImageClassifier.lpi* or some valid file name like in my case *SimpleImageClassifier_CPU_Cifar.pas*.



```
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());
```

```
[dcc32 Warning] SimplePlantLeafDisease.pas(265): W1011 Text after final 'END.' - ignored by compiler
Success
Elapsed time: 00:00:02.2
```

This example has interesting aspects to look at: Its source code is very small and Layers are added sequentially. Then the Training hyper-parameters are defined before calling the fit method. You need fit() to train the model! In line 155 we start with the test-class from the **CAI** library and we create the neural net layers too:

```
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, {SuppressBias=}0));
 NN.AddLayer(TNNetMaxPool.Create({Size=}2));
 NN.AddLayer(TNNetConvolutionReLU.Create({Features=}32, {FeatureSize=}5,
{Padding=}0, {Stride=}1, {SuppressBias=}0));
 NN.AddLayer(TNNetMaxPool.Create({Size=}2));
 NN.AddLayer(TNNetConvolutionReLU.Create({Features=}32, {FeatureSize=}5,
{Padding=}0, {Stride=}1, {SuppressBias=}0));
 NN.AddLayer(TNNetLayerFullConnectReLU.Create({Neurons=}32));
 NN.AddLayer(TNNetFullConnectLinear.Create(NumClasses));
 NN.AddLayer(TNNetSoftMax.Create());
 writeln(NN.SaveDataToString);
 //readln;
```

Then we load the data volumes for training. There is a trick that you can do with this API or any other API when working with image classification: you can increase the input image size. As per the following example (train, test and validate), by increasing CIFAR-10 input image sizes from 32x32 to 48x48, you can gain up to 2% in classification accuracy.

```
CreateCifar10Volumes(ImgTrainingVolumes, ImgValidationVolumes, ImgTestVolumes);

 WriteLn
    (
      'Training Images:', ImgTrainingVolumes.Count,
      ' Validation Images:', ImgValidationVolumes.Count,
      ' Test Images:', ImgTestVolumes.Count
    ); //*)

WriteLn('Neural Network will minimize error with:');
WriteLn(' Layers: ', NN.CountLayers());
WriteLn(' Neurons: ', NN.CountNeurons());
WriteLn(' Weights: ', NN.CountWeights());
writeln('Start Convolution Net...');
readln;
```

As an output on the shell we see the Neural Network will minimize error with:

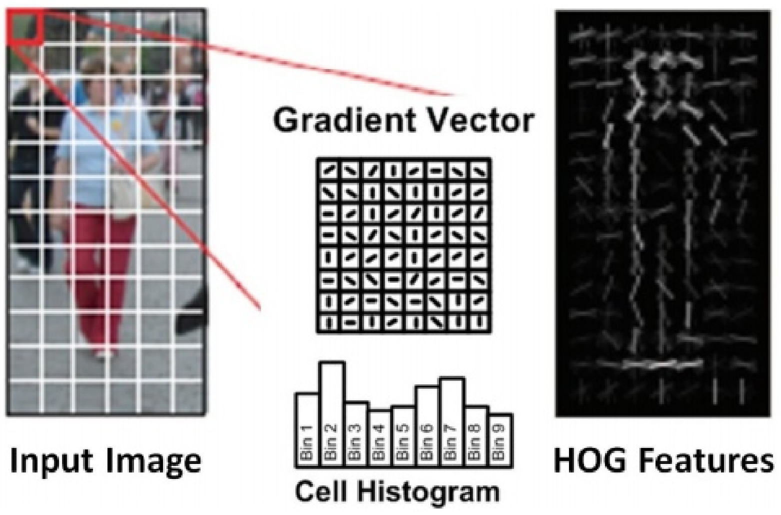
```
 Layers: 9
 Neurons: 122
 Weights: 40944
Start Convolution Net...
```

Later on, this is how the training/fitting method is called:

```
NeuralFit := TNeuralImageFit.Create;
//readln;
NeuralFit.FileNameBase := 'EKONSimpleImageClassifier2';
NeuralFit.InitialLearningRate := fLearningRate;
NeuralFit.Inertia := fInertia;
NeuralFit.LearningRateDecay := 0.005;
    NeuralFit.StaircaseEpochs := 17;
  //  NeuralFit.Inertia := 0.9;
    NeuralFit.L2Decay := 0.00001;

//readln;  best fit: batch 128 epochs 100
// just for test and evaluate the process - epochs = 1, otherwise 10 or 100!

NeuralFit.Fit(NN, ImgTrainingVolumes, ImgValidationVolumes, ImgTestVolumes,
NumClasses,
                                    {batchsize}128, {epochs}1);
    writeln('End Convolution Net...');
    readln;
    NeuralFit.Free;

    NN.Free;
    ImgTestVolumes.Free;
    ImgValidationVolumes.Free;
    ImgTrainingVolumes.Free;
end;
```

The Learning rate in `NeuralFit.InitialLearningRate := fLearningRate;` is an important hyperparameter that controls how much we adjust the weights in the network according to the gradient.

Like in a histogram of oriented gradients (HOG) is basically a feature descriptor that is used to detect objects in image processing and other computer vision techniques. The Histogram of oriented gradients descriptor technique includes occurrences of gradient orientation in localised portions of an image, such as detection window, region of interest (ROI), among others. Advantage of HOG-like features is their simplicity, and it is easier to understand information they carry.

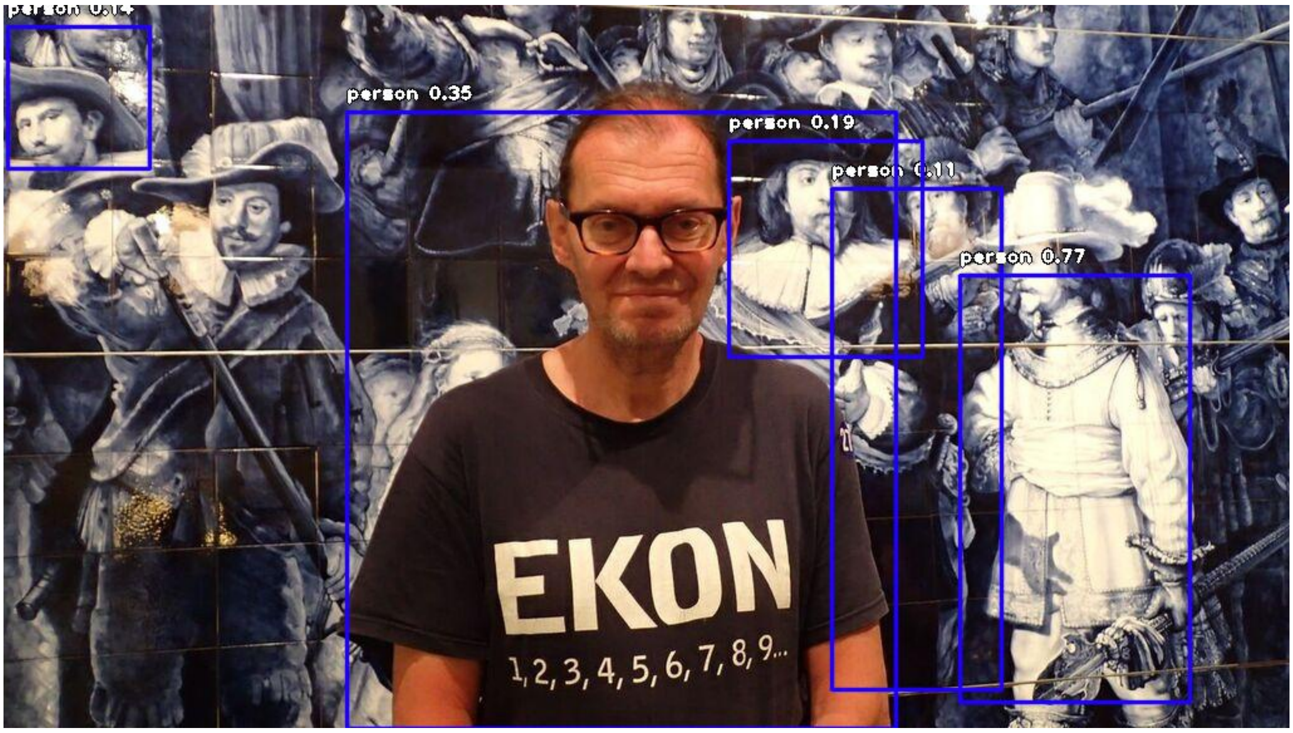


https://maxbox4.files.wordpress.com/2020/07/gradienthistogramscreenshot-501.png

In the last step we see an image recognition that specializes in people or humans. Surprisingly, from a well known picture in Delft (The Night Watch), the people in the background are more likely to be recognized than the real person. That has to do with the image section in the sense of focus of the rectangle.

```
person : 11.248066276311874
person : 14.372693002223969
person : 19.247493147850037
person : 34.878602623939514
person : 77.2484838962555
image detector compute ends…
```

```
#loads model from path specified above using the setModelPath() class method.
detector.loadModel()
```



https://maxbox4.files.wordpress.com/2020/07/max_netherland_out2.jpg?w=1000&h=

To detect only some of the objects above, I will need to call a *CustomObjects* method and set the name of the object(s) we want to detect to through. The rest are False by default. In our example, we detect customized only person, laptop, cat and dogs. I will explain that in one of the next magazine.

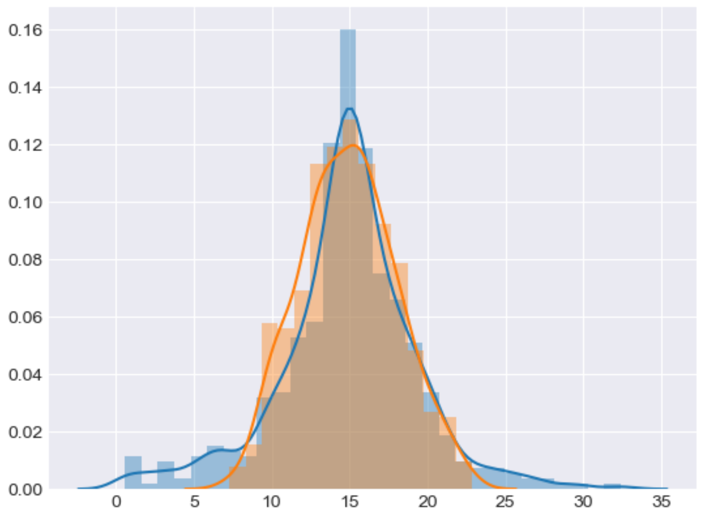
So we get the whole validation for demo purpose in a total time about only 12 minutes, but with a weak accuracy. But the real validation has a longer duration or must have to proof the concept (*best fit: batch 128 - epochs 100*).

```
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.
```

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

```
Starting Validation Demo just to make it shorter.
VALIDATION RECORD! Saving NN at EKONSimpleImageClassifier.nn
Epochs: 1 Examples seen:40000 Validation Accuracy: 0.4608 Validation Error: 1.3855 Validation Loss: 1.4801 Total time:  11.39min
Epoch time: 8.6 minutes. 100 epochs: 14 hours.
Epochs: 1. Working time: 0.19 hours.
Finished.
```

Conclusion:
So how can we shorten the validation time? With a pretrained model! Pretrained models are a wonderful source of help for people looking to learn an algorithm or try out an existing framework online or offline. But the predictions made using pretrained models would not be so effective as you train the model with your data, that's the trade-off. Due to time restrictions or computational restraints, it's not always possible to build a model from scratch (like we did) which is why pretrained models exist like the pretrained-yolov3.h5!



The script and data can be found:

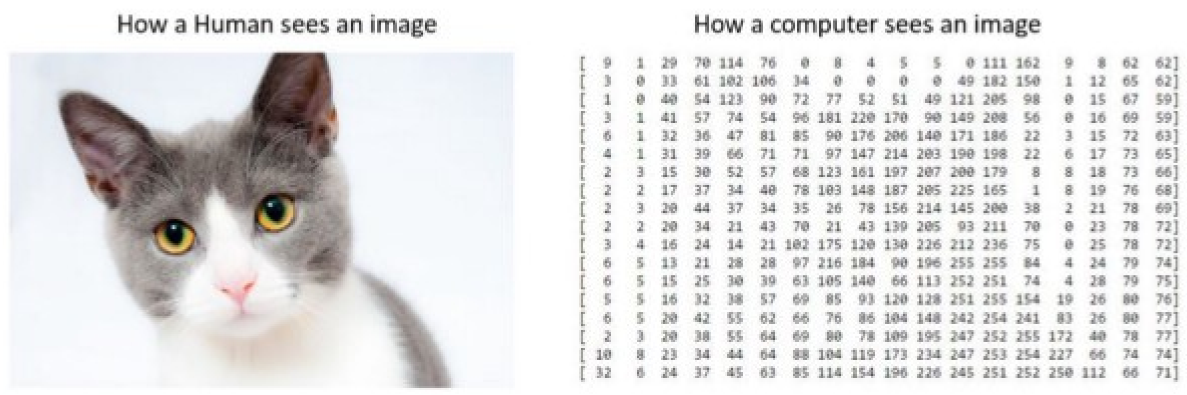
http://www.softwareschule.ch/examples/detector2.htm
https://sourceforge.net/projects/maxbox/files/Examples/EKON/EKON24/ImageDetector/

https://sourceforge.net/projects/maxbox/files/Examples/EKON/EKON24/SimpleImageClassifier_CPU_Cifar2.pas/download

Author: Max Kleiner

Ref:
    http://www.softwareschule.ch/box.htm

Doc:
    https://maxbox4.wordpress.com



```
[   9   1  29  70 114  76   0   8   4   5   5   0 111 162   9   8  62  62]
[   3   0  33  61 102 106  34   0   0   0   0  49 182 150   1  12  65  62]
[   1   0  40  54 123  90  72  77  52  51  49 121 205  98   0  15  67  59]
[   3   1  41  57  74  54  96 181 220 170  90 149 208  56   0  16  69  59]
[   6   1  32  36  47  81  85  90 176 206 140 171 186  22   3  15  72  63]
[   4   1  31  39  66  71  71  97 147 214 203 190 198  22   6  17  73  65]
[   2   3  15  30  52  57  68 123 161 197 207 200 179   8   8  18  73  66]
[   2   2  17  37  34  40  78 103 148 187 205 225 165   1   8  19  76  68]
[   2   3  20  44  37  34  35  26  78 156 214 145 200  38   2  21  78  69]
[   2   2  20  34  21  43  70  21  43 139 205  93 211  70   0  23  78  72]
[   3   4  16  24  14  21 102 175 120 130 226 212 236  75   0  25  78  72]
[   6   5  13  21  28  28  97 216 184  90 196 255 255  84   4  24  79  74]
[   6   5  15  25  30  39  63 105 140  66 113 252 251  74   4  28  79  75]
[   5   5  16  32  38  57  69  85  93 120 128 251 255 154  19  26  80  76]
[   6   5  20  42  55  62  66  76  86 104 148 242 254 241  83  26  80  77]
[   2   3  20  38  55  64  69  80  78 109 195 247 252 255 172  40  78  77]
[  10   8  23  34  44  64  88 104 119 173 234 247 253 254 227  66  74  74]
[  32   6  24  37  45  63  85 114 154 196 226 245 251 252 250 112  66  71]
```

```
For Internal Use
-------------------------------------------------------------

OpenGL 4.5 - Headertranslation (Personal Fork)
       Version 4.5a (Personal Fork)

       Supported environments and targets :
        - (Win32) Delphi 7 and up
        - (Win32, Win64) Delphi XE2
        - (Win32, Win64, Linux, MacOSX) FreePascal (1.9.3 and up)

               OpenCL1.2 and Delphi and Windows                     *)
(*                                                                      *)
(*      headers versions: 0.07                                          *)
(*      file name       : CL.pas                                        *)
(*      last modify     : 10.12.11                                      *)
(*      license         : BSD


program CAIProject12_64bit;

{$APPTYPE CONSOLE}

{$R *.res}

uses
  System.SysUtils,
  SimpleImageClassifier_CPU_Cifar in 'SimpleImageClassifier_CPU_Cifar.pas',
  neuralab in '..\..\neural\neuralab.pas',
  neuralabfun in '..\..\neural\neuralabfun.pas',
  neuralbit in '..\..\neural\neuralbit.pas',
  neuralbyteprediction in '..\..\neural\neuralbyteprediction.pas',
  neuralcache in '..\..\neural\neuralcache.pas',
  neuraldatasets in '..\..\neural\neuraldatasets.pas',
  neuraldatasetsv in '..\..\neural\neuraldatasetsv.pas',
  neuralevolutionary in '..\..\neural\neuralevolutionary.pas',
  neuralfit in '..\..\neural\neuralfit.pas',
  neuralgeneric in '..\..\neural\neuralgeneric.pas',
  neuralnetwork in '..\..\neural\neuralnetwork.pas',
  neuralopencl in '..\..\neural\neuralopencl.pas',
  neuralopenclv in '..\..\neural\neuralopenclv.pas',
  neuralplanbuilder in '..\..\neural\neuralplanbuilder.pas',
  neuralthread in '..\..\neural\neuralthread.pas',
  neuralvolume in '..\..\neural\neuralvolume.pas',
  neuralvolumev in '..\..\neural\neuralvolumev.pas',
  CL in '..\..\neural\CL.pas',
  CL_D3D9 in '..\..\neural\CL_D3D9.pas',
  CL_D3D10 in '..\..\neural\CL_D3D10.pas',
  CL_DX9_Media_Sharing in '..\..\neural\CL_DX9_Media_Sharing.pas',
  CL_Ext in '..\..\neural\CL_Ext.pas',
  CL_GL in '..\..\neural\CL_GL.pas',
  CL_GL_Ext in '..\..\neural\CL_GL_Ext.pas',
  CL_Platform in '..\..\neural\CL_Platform.pas',
  CLExt in '..\..\neural\CLExt.pas',
  dglOpenGL in '..\..\neural\dglOpenGL.pas';

begin
  try
    { TODO -oUser -cConsole Main : Insert code here }
  except
    on E: Exception do
      Writeln(E.ClassName, ': ', E.Message);
  end;
end.
```

```
Loading 10K images from file "C:\maXbox\EKON24\Delphi-Artificial-NN-Library-mast
er\neural-api-master\neural-api-master\neural\cifar-10-batches-bin\data_batch_1.
bin" ... GLOBAL MIN MAX  -2.0000  1.9844 -2.0000  1.9844 -2.0000  1.9844 Done.
Loading 10K images from file "C:\maXbox\EKON24\Delphi-Artificial-NN-Library-mast
er\neural-api-master\neural-api-master\neural\cifar-10-batches-bin\data_batch_2.
bin" ... GLOBAL MIN MAX  -2.0000  1.9844 -2.0000  1.9844 -2.0000  1.9844 Done.
Loading 10K images from file "C:\maXbox\EKON24\Delphi-Artificial-NN-Library-mast
er\neural-api-master\neural-api-master\neural\cifar-10-batches-bin\data_batch_3.
bin" ... GLOBAL MIN MAX  -2.0000  1.9844 -2.0000  1.9844 -2.0000  1.9844 Done.
Loading 10K images from file "C:\maXbox\EKON24\Delphi-Artificial-NN-Library-mast
er\neural-api-master\neural-api-master\neural\cifar-10-batches-bin\data_batch_4.
bin" ... GLOBAL MIN MAX  -2.0000  1.9844 -2.0000  1.9844 -2.0000  1.9844 Done.
Loading 10K images from file "C:\maXbox\EKON24\Delphi-Artificial-NN-Library-mast
er\neural-api-master\neural-api-master\neural\cifar-10-batches-bin\data_batch_5.
bin" ... GLOBAL MIN MAX  -2.0000  1.9844 -2.0000  1.9844 -2.0000  1.9844 Done.
Loading 10K images from file "C:\maXbox\EKON24\Delphi-Artificial-NN-Library-mast
er\neural-api-master\neural-api-master\neural\cifar-10-batches-bin\test_batch.bi
n" ... GLOBAL MIN MAX  -2.0000  1.9844 -2.0000  1.9844 -2.0000  1.9844 Done.
Training Images:40000 Validation Images:10000 Test Images:10000
Neural Network will minimize error with:
 Layers: 9
 Neurons: 122
 Weights: 40944
Start Convolution Net...


File name is: EKONSimpleImageClassifier
Learning rate:0.001000 L2 decay:0.000010 Inertia:0.900000 Batch size:128 Step si
ze:128 Staircase ephocs:17
Training images:40000
Validation images:10000
Test images:10000
Computing...

Starting Validation.
VALIDATION RECORD! Saving NN at EKONSimpleImageClassifier.nn
Epochs: 1 Examples seen:40000 Validation Accuracy: 0.4715 Validation Error: 1.37
04 Validation Loss: 1.4699 Total time:  11.36min
Epoch time: 8.6 minutes. 100 epochs: 14 hours.
```

```
Epochs: 1. Working time: 0.19 hours.
Finished.
End Convolution Net...


///This file has an implementation to classify
//plant leaf diseases and Image Classifier. You can get the dataset at
//https://data.mendeley.com/datasets/tywbtsjrjv/1/files/d5652a28-c1d8-4b76-97f3-
72fb80f94efc/Plant_leaf_diseases_dataset_without_augmentation.zip?dl=1 .
//Folders with plant diseases will need to be stored inside of a folder named
"plant".
{The example above shows how to load the dataset used with the paper
Identification of Plant Leaf Diseases Using a 9-layer Deep Convolutional Neural
Network. 90% is loaded as training data while 5% is loaded for each validation
and testing. Images are being resized to 128x128.

As you can see on these raw results, the test classification
(or the plant disease diagnostic) accuracy is 98.95%.      }

//https://github.com/joaopauloschuler/neural-
api/blob/master/examples/SimplePlantLeafDisease/README.md

unit SimpleImageClassifier_CPU_Cifar;
(*
 Coded by Joao Paulo Schwarz Schuler. add by Max for EKON24 in Community Edition
 https://github.com/joaopauloschuler/neural-api

https://github.com/maxkleiner/maXbox/blob/master/EKON24_SimpleImageClassificatio
nCPU.ipynb

https://colab.research.google.com/github/maxkleiner/maXbox/blob/master/EKON24_Si
mpleImageClassificationCPU.ipynb
*)
//{$mode objfpc}{$H+}

//{$DEFINE FPC}
interface

uses {$IFDEF UNIX} {$IFDEF UseCThreads}
  cthreads, {$ENDIF} {$ENDIF}
  Classes, SysUtils, {CustApp,} neuralnetwork, neuralvolume, Math,
neuraldatasets,
  neuralfit;

//type
 { TTestCNNAlgo = class(TForm)
  protected
    procedure DoRun; override;
  end; }

 implementation

  (*
 procedure CreateVolumesFromImagesFromFolder(out ImgTrainingVolumes,
ImgValidationVolumes,
  ImgTestVolumes: TNNetVolumeList;
  FolderName, pImageSubFolder: string;
  color_encoding: integer;
  TrainingProp, ValidationProp, TestProp: single;
  NewSizeX: integer = 0; NewSizeY: integer = 0);
var
```

```
  ClassesAndElements: TClassesAndElements;
begin
  ImgTrainingVolumes := TNNetVolumeList.Create();
  ImgValidationVolumes := TNNetVolumeList.Create();
  ImgTestVolumes := TNNetVolumeList.Create();
  ClassesAndElements := TClassesAndElements.Create();

  if ValidationProp > 0 then
  begin
    ClassesAndElements.LoadFoldersAsClassesProportional(FolderName,
pImageSubFolder, TrainingProp, ValidationProp);
    ClassesAndElements.LoadImages(color_encoding, NewSizeX, NewSizeY);
    ClassesAndElements.AddVolumesTo(ImgValidationVolumes, {EmptySource=}true);
    ClassesAndElements.Clear;
  end;

  if TestProp > 0 then
  begin
    ClassesAndElements.LoadFoldersAsClassesProportional(FolderName,
pImageSubFolder, TrainingProp + ValidationProp, TestProp);
    ClassesAndElements.LoadImages(color_encoding, NewSizeX, NewSizeY);
    ClassesAndElements.AddVolumesTo(ImgTestVolumes, {EmptySource=}true);
    ClassesAndElements.Clear;
  end;

  if TrainingProp > 0 then
  begin
    ClassesAndElements.LoadFoldersAsClassesProportional(FolderName,
pImageSubFolder, 0, TrainingProp);
    ClassesAndElements.LoadImages(color_encoding, NewSizeX, NewSizeY);
    ClassesAndElements.AddVolumesTo(ImgTrainingVolumes, {EmptySource=}true);
    ClassesAndElements.Clear;
  end;
  ClassesAndElements.Free;
end;
*)

  procedure TTestCNNAlgo;
  var
    NN: TNNet;
    NeuralFit: TNeuralImageFit;
    ImgTrainingVolumes, ImgValidationVolumes, ImgTestVolumes: TNNetVolumeList;
    ProportionToLoad: Single;
  begin
    WriteLn('Creating Neural Network...');
    NN := TNNet.Create();
    NN.AddLayer([
      TNNetInput.Create(128, 128, 3),
      TNNetConvolutionLinear.Create({Features=}64, {FeatureSize=}5, {Padding=}4,
{Stride=}2),
      TNNetMaxPool.Create(2),
      TNNetMovingStdNormalization.Create(),
      TNNetConvolutionReLU.Create({Features=}64, {FeatureSize=}3, {Padding=}1,
{Stride=}1),
      TNNetConvolutionReLU.Create({Features=}64, {FeatureSize=}3, {Padding=}1,
{Stride=}1),
      TNNetMaxPool.Create(2),
      TNNetConvolutionReLU.Create({Features=}64, {FeatureSize=}3, {Padding=}1,
{Stride=}1),
      TNNetConvolutionReLU.Create({Features=}64, {FeatureSize=}3, {Padding=}1,
{Stride=}1),
      TNNetConvolutionReLU.Create({Features=}64, {FeatureSize=}3, {Padding=}1,
{Stride=}2),
      TNNetDropout.Create(0.5),
      TNNetMaxPool.Create(2),
      TNNetFullConnectLinear.Create(39),
      TNNetSoftMax.Create()
    ]);
    NN.DebugStructure();
    // change ProportionToLoad to a smaller number if you don't have available
32GB of RAM.
    ProportionToLoad := 1;
    WriteLn('Loading ', Round(ProportionToLoad*100), '% of the Plant leave
disease dataset into memory.');
    {$IFDEF FPC}
    CreateVolumesFromImagesFromFolder
    (
      ImgTrainingVolumes, ImgValidationVolumes, ImgTestVolumes,
      {FolderName=}'plant', {pImageSubFolder=}'',
      {color_encoding=}0{RGB},
      {TrainingProp=}0.9*ProportionToLoad,
      {ValidationProp=}0.05*ProportionToLoad,
      {TestProp=}0.05*ProportionToLoad,
      {NewSizeX=}128, {NewSizeY=}128
    );     //*)
    {$ENDIF}

    ImgTrainingVolumes := TNNetVolumeList.Create();
    ImgValidationVolumes := TNNetVolumeList.Create();
    ImgTestVolumes := TNNetVolumeList.Create();

    CreateCifar10Volumes(ImgTrainingVolumes, ImgValidationVolumes,
                                          ImgTestVolumes,csEncodeRGB);

    WriteLn
    (
      'Training Images:', ImgTrainingVolumes.Count,
      ' Validation Images:', ImgValidationVolumes.Count,
      ' Test Images:', ImgTestVolumes.Count
    );

    NeuralFit := TNeuralImageFit.Create;
    NeuralFit.FileNameBase := 'SimplePlantLeafDisease';
    NeuralFit.InitialLearningRate := 0.001;
    NeuralFit.LearningRateDecay := 0.01;
    NeuralFit.StaircaseEpochs := 10;
    NeuralFit.Inertia := 0.9;
    NeuralFit.L2Decay := 0.00001;
    //NeuralFit.MaxThreadNum := 8;
    NeuralFit.Fit(NN, ImgTrainingVolumes, ImgValidationVolumes, ImgTestVolumes,
{NumClasses=}39, {batchsize=}64, {epochs=}50);
    NeuralFit.Free;
    NN.Free;
    ImgTestVolumes.Free;
    ImgValidationVolumes.Free;
    ImgTrainingVolumes.Free;
    //Terminate;
  end;

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, {SuppressBias=}0));
  NN.AddLayer(TNNetMaxPool.Create({Size=}2));
  NN.AddLayer(TNNetConvolutionReLU.Create({Features=}32, {FeatureSize=}5,
 {Padding=}0, {Stride=}1, {SuppressBias=}0));
  NN.AddLayer(TNNetMaxPool.Create({Size=}2));
  NN.AddLayer(TNNetConvolutionReLU.Create({Features=}32, {FeatureSize=}5,
 {Padding=}0, {Stride=}1, {SuppressBias=}0));
  NN.AddLayer(TNNetLayerFullConnectReLU.Create({Neurons=}32));
  NN.AddLayer(TNNetFullConnectLinear.Create(NumClasses));
  NN.AddLayer(TNNetSoftMax.Create());
  writeln(NN.SaveDataToString);
  //readln;

     {ImgTrainingVolumes := TNNetVolumeList.Create();
     ImgValidationVolumes := TNNetVolumeList.Create();
     ImgTestVolumes := TNNetVolumeList.Create();}

  CreateCifar10Volumes(ImgTrainingVolumes, ImgValidationVolumes, ImgTestVolumes);

  WriteLn
     (
       'Training Images:', ImgTrainingVolumes.Count,
       ' Validation Images:', ImgValidationVolumes.Count,
       ' Test Images:', ImgTestVolumes.Count
     ); //*)

 WriteLn('Neural Network will minimize error with:');
 WriteLn(' Layers: ', NN.CountLayers());
 WriteLn(' Neurons: ', NN.CountNeurons());
 WriteLn(' Weights: ', NN.CountWeights());
 writeln('Start Convolution Net...');
 readln;

 //Later on, this is how the training/fitting is called:

 NeuralFit := TNeuralImageFit.Create;
 //readln;
 NeuralFit.FileNameBase := 'EKONSimpleImageClassifier';
 NeuralFit.InitialLearningRate := fLearningRate;
 NeuralFit.Inertia := fInertia;
 NeuralFit.LearningRateDecay := 0.005;
     NeuralFit.StaircaseEpochs := 17;
   //  NeuralFit.Inertia := 0.9;
     NeuralFit.L2Decay := 0.00001;

 //r//eadln;  best fit: batch 128 epochs 100
 // just for test and evaluate the process - epochs = 1, otherwise 10 or 100!
```

```
NeuralFit.Fit(NN, ImgTrainingVolumes, ImgValidationVolumes, ImgTestVolumes,
NumClasses,
                                   {batchsize}128, {epochs}1);
    writeln('End Convolution Net...');
    readln;
    NeuralFit.Free;

    NN.Free;
    ImgTestVolumes.Free;
    ImgValidationVolumes.Free;
    ImgTrainingVolumes.Free;
end;

//var
  //Application: TTestCNNAlgo;
begin
  {Application := TTestCNNAlgo.Create(nil);
  Application.Title:='Plant Leaf Disease Classification';
  Application.Run;
  Application.Free;    }
  //TTestCNNAlgo;
  TTestCifar10Algo;
end.


(*import os
import urllib.request

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')

  import tarfile
tar = tarfile.open("file.tar")
tar.extractall()
tar.close()

  >>> urllib.request.urlretrieve(url, './file.tar')
('./file.tar', <http.client.HTTPMessage object at 0x0000009F53FE8CF8>)

  There is a trick that you can do with this API or any other API when working
with image classification: you can increase the input image size.

As per the following example, by increasing CIFAR-10 input image sizes from
32x32 to 48x48, you can gain up to 2% in classification accuracy.

You can change image sizes with:

ImgTrainingVolumes.ResizeImage(48, 48);
ImgValidationVolumes.ResizeImage(48, 48);
ImgTestVolumes.ResizeImage(48, 48);

  *)


Build started 03/09/2020 21:20:40.
__________________________________________________
Project "C:\maXbox\EKON24\Delphi-Artificial-NN-Library-master\neural-api-
master\neural-api-
```

```
master\examples\SimplePlantLeafDisease\CAIProject12_64bit.dproj" (Make
target(s)):
Target _PasCoreCompile:
    c:\program files (x86)\embarcadero\studio\20.0\bin\dcc64.exe -$O- -$W+ --no-
config -M -Q -TX.exe
-AGenerics.Collections=System.Generics.Collections;Generics.Defaults=System.Gene
rics.Defaults;WinTypes=Winapi.Windows;WinProcs=Winapi.Windows;DbiTypes=BDE;DbiPr
ocs=BDE;DbiErrs=BDE -DDEBUG -E.\Win64\Debug -I"c:\program files
(x86)\embarcadero\studio\20.0\lib\Win64\debug";"c:\program files
(x86)\embarcadero\studio\20.0\lib\Win64\release";C:\Users\max\Documents\Embarcad
ero\Studio\20.0\Imports;"c:\program files
(x86)\embarcadero\studio\20.0\Imports";C:\Users\Public\Documents\Embarcadero\Stu
dio\20.0\Dcp\Win64;"c:\program files
(x86)\embarcadero\studio\20.0\include";C:\Users\max\Documents\Embarcadero\Studio
\20.0\CatalogRepository\dcliot-
260\Src\Delphi\Win64\Debug;C:\Users\max\Documents\Embarcadero\Studio\20.0\Catalo
gRepository\VeraSchlageLock-
260\Src;C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\SynEdit
-
1.5\Src\source;C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\
SysTools-
4.3\source;C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\Vera
EverspringTempHumiditySensor-260\Src
-LEC:\Users\Public\Documents\Embarcadero\Studio\20.0\Bpl\Win64
-LNC:\Users\Public\Documents\Embarcadero\Studio\20.0\Dcp\Win64 -NU.\Win64\Debug
-NSWinapi;System.Win;Data.Win;Datasnap.Win;Web.Win;Soap.Win;Xml.Win;System;Xml;D
ata;Datasnap;Web;Soap;Vcl;Vcl.Imaging;Vcl.Touch;Vcl.Samples;Vcl.Shell;
-O"c:\program files
(x86)\embarcadero\studio\20.0\lib\Win64\release";C:\Users\max\Documents\Embarcad
ero\Studio\20.0\Imports;"c:\program files
(x86)\embarcadero\studio\20.0\Imports";C:\Users\Public\Documents\Embarcadero\Stu
dio\20.0\Dcp\Win64;"c:\program files
(x86)\embarcadero\studio\20.0\include";C:\Users\max\Documents\Embarcadero\Studio
\20.0\CatalogRepository\dcliot-
260\Src\Delphi\Win64\Debug;C:\Users\max\Documents\Embarcadero\Studio\20.0\Catalo
gRepository\VeraSchlageLock-
260\Src;C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\SynEdit
-
1.5\Src\source;C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\
SysTools-
4.3\source;C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\Vera
EverspringTempHumiditySensor-260\Src -R"c:\program files
(x86)\embarcadero\studio\20.0\lib\Win64\release";C:\Users\max\Documents\Embarcad
ero\Studio\20.0\Imports;"c:\program files
(x86)\embarcadero\studio\20.0\Imports";C:\Users\Public\Documents\Embarcadero\Stu
dio\20.0\Dcp\Win64;"c:\program files
(x86)\embarcadero\studio\20.0\include";C:\Users\max\Documents\Embarcadero\Studio
\20.0\CatalogRepository\dcliot-
260\Src\Delphi\Win64\Debug;C:\Users\max\Documents\Embarcadero\Studio\20.0\Catalo
gRepository\VeraSchlageLock-
260\Src;C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\SynEdit
-
1.5\Src\source;C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\
SysTools-
4.3\source;C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\Vera
EverspringTempHumiditySensor-260\Src -U"c:\program files
(x86)\embarcadero\studio\20.0\lib\Win64\debug";"c:\program files
(x86)\embarcadero\studio\20.0\lib\Win64\release";C:\Users\max\Documents\Embarcad
ero\Studio\20.0\Imports;"c:\program files
(x86)\embarcadero\studio\20.0\Imports";C:\Users\Public\Documents\Embarcadero\Stu
dio\20.0\Dcp\Win64;"c:\program files
```

```
(x86)\embarcadero\studio\20.0\include";C:\Users\max\Documents\Embarcadero\Studio
\20.0\CatalogRepository\dcliot-
260\Src\Delphi\Win64\Debug;C:\Users\max\Documents\Embarcadero\Studio\20.0\Catalo
gRepository\VeraSchlageLock-
260\Src;C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\SynEdit
-
1.5\Src\source;C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\
SysTools-
4.3\source;C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\Vera
EverspringTempHumiditySensor-260\Src -CC -V -VN -VR
-NBC:\Users\Public\Documents\Embarcadero\Studio\20.0\Dcp\Win64
-NHC:\Users\Public\Documents\Embarcadero\Studio\20.0\hpp\Win64 -NO.\Win64\Debug
CAIProject12_64bit.dpr
    C:\maXbox\EKON24\Delphi-Artificial-NN-Library-master\neural-api-
master\neural-api-
master\examples\SimplePlantLeafDisease\SimpleImageClassifier_CPU_Cifar.pas(270,1
): warning W1011: W1011 Text after final 'END.' - ignored by compiler
Done building target "_PasCoreCompile" in project "CAIProject12_64bit.dproj".
Done building project "CAIProject12_64bit.dproj".
Build succeeded.
C:\maXbox\EKON24\Delphi-Artificial-NN-Library-master\neural-api-master\neural-
api-
master\examples\SimplePlantLeafDisease\SimpleImageClassifier_CPU_Cifar.pas(270,1
): warning W1011: W1011 Text after final 'END.' - ignored by compiler
    1 Warning(s)
    0 Error(s)
Time Elapsed 00:00:12.75

Build - Output

Checking project dependencies...
Compiling CAIProject12_64bit.dproj (Debug, Win64)
dcc64 command line for "CAIProject12_64bit.dpr"
  c:\program files (x86)\embarcadero\studio\20.0\bin\dcc64.exe -$O- -$W+ --no-
config -M -Q -TX.exe -AGenerics.Collections=System.Generics.Collections;

Generics.Defaults=System.Generics.Defaults;WinTypes=Winapi.Windows;WinProcs=Wina
pi.Windows;DbiTypes=BDE;DbiProcs=BDE;DbiErrs=BDE -DDEBUG
  -E.\Win64\Debug -I"c:\program files
(x86)\embarcadero\studio\20.0\lib\Win64\debug";"c:\program files
(x86)\embarcadero\studio\20.0\lib\Win64\release";
  C:\Users\max\Documents\Embarcadero\Studio\20.0\Imports;"c:\program files
(x86)\embarcadero\studio\20.0\Imports";
  C:\Users\Public\Documents\Embarcadero\Studio\20.0\Dcp\Win64;"c:\program files
(x86)\embarcadero\studio\20.0\include";
  C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\dcliot-
260\Src\Delphi\Win64\Debug;

C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\VeraSchlageLock
-260\Src;
  C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\SynEdit-
1.5\Src\source;
  C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\SysTools-
4.3\source;

C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\VeraEverspringT
empHumiditySensor-260\Src
  -LEC:\Users\Public\Documents\Embarcadero\Studio\20.0\Bpl\Win64
-LNC:\Users\Public\Documents\Embarcadero\Studio\20.0\Dcp\Win64 -NU.\Win64\Debug

-NSWinapi;System.Win;Data.Win;Datasnap.Win;Web.Win;Soap.Win;Xml.Win;System;Xml;D
```

```
ata;Datasnap;Web;Soap;Vcl;Vcl.Imaging;Vcl.Touch;Vcl.Samples;Vcl.Shell;
   -O"c:\program files
(x86)\embarcadero\studio\20.0\lib\Win64\release";C:\Users\max\Documents\Embarcad
ero\Studio\20.0\Imports;"c:\program files

(x86)\embarcadero\studio\20.0\Imports";C:\Users\Public\Documents\Embarcadero\Stu
dio\20.0\Dcp\Win64;"c:\program files

(x86)\embarcadero\studio\20.0\include";C:\Users\max\Documents\Embarcadero\Studio
\20.0\CatalogRepository\dcliot-260\Src\Delphi\Win64\Debug;

C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\VeraSchlageLock
-260\Src;
  C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\SynEdit-
1.5\Src\source;
  C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\SysTools-
4.3\source;
C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\VeraEverspringT
empHumiditySensor-260\Src -R"c:\program files
(x86)\embarcadero\studio\20.0\lib\Win64\release";C:\Users\max\Documents\Embarcad
ero\Studio\20.0\Imports;"c:\program files
(x86)\embarcadero\studio\20.0\Imports";C:\Users\Public\Documents\Embarcadero\Stu
dio\20.0\Dcp\Win64;"c:\program files

(x86)\embarcadero\studio\20.0\include";C:\Users\max\Documents\Embarcadero\Studio
\20.0\CatalogRepository\dcliot-260\Src\Delphi\Win64\Debug;
C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\VeraSchlageLock
-260\Src;
  C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\SynEdit-
1.5\Src\source;
  C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\SysTools-
4.3\source;

C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\VeraEverspringT
empHumiditySensor-260\Src -U"c:\program files
  (x86)\embarcadero\studio\20.0\lib\Win64\debug";"c:\program files
(x86)\embarcadero\studio\20.0\lib\Win64\release";
  C:\Users\max\Documents\Embarcadero\Studio\20.0\Imports;"c:\program files
(x86)\embarcadero\studio\20.0\Imports";
  C:\Users\Public\Documents\Embarcadero\Studio\20.0\Dcp\Win64;"c:\program files
(x86)\embarcadero\studio\20.0\include";
  C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\dcliot-
260\Src\Delphi\Win64\Debug;
C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\VeraSchlageLock
-260\Src;
  C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\SynEdit-
1.5\Src\source;
  C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\SysTools-
4.3\source;

C:\Users\max\Documents\Embarcadero\Studio\20.0\CatalogRepository\VeraEverspringT
empHumiditySensor-260\Src -CC -V -VN -VR
  -NBC:\Users\Public\Documents\Embarcadero\Studio\20.0\Dcp\Win64
-NHC:\Users\Public\Documents\Embarcadero\Studio\20.0\hpp\Win64 -NO.\Win64\Debug
  CAIProject12_64bit.dpr
[dcc64 Warning] SimpleImageClassifier_CPU_Cifar.pas(270): W1011 Text after final
'END.' - ignored by compiler
Success
Elapsed time: 00:00:13.0
```