//
CNN Model
____
maXbox Starter 89 - Build a CNN (Convolutional Neural Network) - Max Kleiner

"space your code"
    — mX4
 https://my6.code.blog/2021/09/08/improver-4/

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.
We just discuss to build the model, not the training or classification init.

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.

Neuralvolume, neuralnetwork, neuralab, neuraldatasets, 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 youll be almost ready to go! With maXbox this in not necessary cause the libs are precompiled out of the box.
The docu for a next tutorial can found at:

Docu of main Neural Unit for maXbox:
http://www.softwareschule.ch/examples/uPSI_NeuralNetworkCAI.txt
http://www.softwareschule.ch/examples/uPSI_neuralnetworkcai.txt
http://www.softwareschule.ch/examples/uPSI_neuralvolume.txt
http://www.softwareschule.ch/examples/uPSI_neuraldatasets.txt
http://www.softwareschule.ch/examples/uPSI_neuralfit.txt
Source of the Lib and Lab:
https://github.com/joaopauloschuler/neural-api

Further with Google Colab you can import an image dataset, train an image classifier on it, and evaluate the model, all in just a few lines of code. Colab notebooks execute code on Googles cloud servers, meaning you can leverage the power of Google hardware, including GPUs and TPUs, regardless of the power of your machine.

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.
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! So we start with the test-class from the CAI library and we create the neural net layers too, this is how a sequential CNN array of layers is added:

```
procedure CreatesimpleCNN;
 var NumClasses: byte;  NN: TNNet;
      InputVolume, PredictedVolume, vDesiredVolume: TNNetVolume;
  //Example - How to Create Your Neural Network
 begin
   NumClasses:= 10;
   NN:= TNNet.Create();
   InputVolume:= TNNetVolume.create;
   //inputvolume.resize
   NN.AddLayer( TNNetInput.Create4(32,32,3) );
   NN.AddLayer( TNNetConvolutionReLU.Create( 16,5,0,0,1) );
   NN.AddLayer( TNNetMaxPool.Create44(2,0,0) );
   NN.AddLayer( TNNetConvolutionReLU.Create(128,5,0,0,1) );
   NN.AddLayer( TNNetMaxPool.Create44(2,0,0) );
   NN.AddLayer( TNNetConvolutionReLU.Create(128,5,0,0,1) );
   NN.AddLayer( TNNetFullConnectReLU.Create30(64,0) );
   NN.AddLayer( TNNetFullConnect.Create27(NumClasses, 0) );
   NN.AddLayer( TNNetSoftMax.Create());
   NN.SetLearningRate(0.01,0.8);
   //Example - How to Train Your Network
  // InputVolume and vDesiredVolume are of the type TNNetVolume
   NN.DebugStructure();
   NN.SaveDataToString;
   NN.Compute65(InputVolume, 0);
   //NN.GetOutput(PredictedVolume);
   //vDesiredVolume.SetClassForReLU(DesiredClass);
   //NN.Backpropagate69(vDesiredVolume);
   InputVolume.Free;
   writeln('InputVolume.Freed;')
   writeln(flots(PI));
 end;
```

The first layer on line 42 sets the 32x32x3 Input Image to define. The second layer defines features and feature size with a first Convolution:

```
 NN.AddLayer(TNNetConvolutionReLU.Create({Features=}16,
             {FeatureSize=}5, {Padding=}0, {Stride=}1, {SuppressBias=}0));
```

Layer 1 has 16 Neurons: 16 Weights: 1200 TNNetConvolutionReLU(16,5,0,0,1) Output: 28,28,16 Learning Rate: 0.0100 Inertia: 0.80 Weight Sum: -1.5885
this you can see with the NN.DebugStructure();

```
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();
 end;
```

Convolutional neural networks are distinguished from other neural networks by their superior performance with image, speech, text or audio signal inputs. They have three main types of layers, which are:

- Convolutional layer
- Pooling layer
- Fully-connected (FC) layer

The convolutional layer is the first layer of a convolutional network. While convolutional layers can be followed by additional convolutional layers or pooling layers, the fully-connected layer is the final layer. With each layer, the CNN increases its complexity, identifying greater portions of the image. Earlier layers focus on simple features, such as colors and edges. As the image data progresses through the layers of the CNN, it starts to recognize larger elements as feature maps or shapes of the object until it finally identifies the intended object.

https://www.ibm.com/cloud/learn/convolutional-neural-networks

Our examples has also padding and stride as layer parameters: Stride is the distance, or number of pixels, that the kernel moves over the input matrix. While stride values of two or greater is rare, a larger stride yields a smaller output. Zero-padding or padding is usually used when the filters do not fit the input image. This sets all elements that fall outside of the input matrix to zero, producing a larger or equally sized output. There are three types of padding:

- Valid padding: This is also known as no padding. In this case, the last convolution is dropped if dimensions do not align.
- Same padding: This padding ensures that the output layer has the same size as the input layer
- Full padding: This type of padding increases the size of the output by adding zeros to the border of the input.

After each convolution operation, a CNN applies a Rectified Linear Unit (ReLU) transformation to the feature map, introducing nonlinearity to the model.

```
NN.AddLayer(TNNetConvolutionReLU.Create({Features=}32,
   {FeatureSize=}5, {Padding=}0, {Stride=}1, {SuppressBias=}0));
```

Convolutional Layer Tuning

Performance is a crucial thing with many aspects. Other Pascal or Python implementations (or older or still-under development versions of CPython or maXbox bytecode) may have slightly different performance characteristics. However, it is generally safe to assume that they are not slower by more than a factor of O(log n).

So this is how we can measure or predict performance, namely by the O-Notation: Generally, 'n' is the number of elements currently in a list or container. 'k' is either the value of a parameter or the number of elements in the parameter.

{$Conclusion}:

Convolutional neural networks power image recognition and computer vision tasks. Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs, and based on those inputs, it can take action.

With most algorithms that handle image processing, the filters are typically created by an engineer based on heuristics. CNNs can learn what characteristics in the filters are the most important. That saves a lot of time and trial and error work since we dont need as many parameters.

Script Ref: 1065__CAI_2_SiImageClassifier21_Tutor_89.txt

http://www.softwareschule.ch/examples/1065__CAI_2_SiImageClassifier21_Tutor_89.txt
https://entwickler-konferenz.de/blog/machine-learning-mit-cai/
https://www.freecodecamp.org/news/convolutional-neural-network-tutorial-for-beginners/