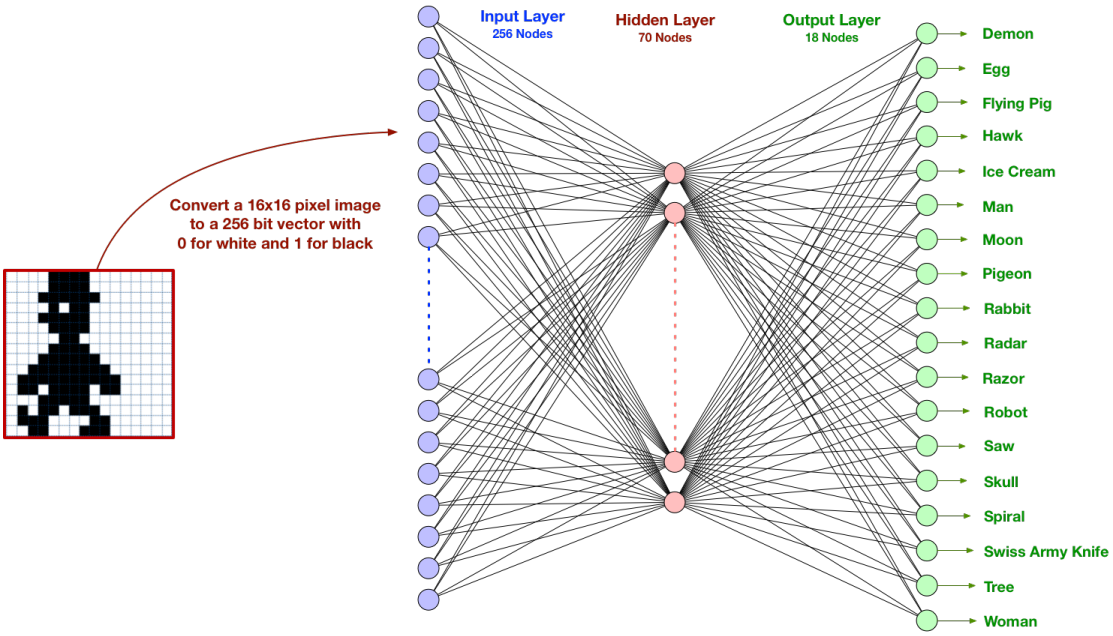
GALWAY-MAYO INSTITUTE OF TECHNOLOGY

Department of Computer Science & Applied Physics

Using a Neural Network to Identify an Image

Overview

Image analysis is one of the most important applications of neural networks. In this practical we will create a neural network capable of classifying an image from a range of 18 different possibilities. The input data consists of a set of 72 black and white images with dimensions of 16x16 pixels.



A neural network can classify an image by re-encoding the image as a one-dimensional bit vector. For the 16x16 images used in this practical, there are a total of 16x16=256 pixels to encode. A separate node in the input layer is required for each pixel. Because the image is in black and white, a black pixel can be represented with a 1 and white area with 0. Note that for a coloured image, the total number of nodes will be the width x height x 4 (RGB + alpha channel). Thus even a relatively modest size of image will require millions of input nodes. Note also that deep neural networks are required for more sophisticated image analysis. These consist of multiple hidden layers, where each hidden layer performs some processing / computation on the input bit vector.

Exercises

- ***Create a new Eclipse project called ImageAnalysis*** with a Java package **ie.gmit.sw.ai**. Download the file **aiImageAnalysisData.zip** and unpack the resources correctly into the Eclipse project. The directory ***sprites*** consist of the 16x16 pixel images named as *image-number*.png. There are four different versions of each image and 18 different characters, giving 72 images in total.

- ***Edit the constructor*** of the class *ImageClassifier* and add the following neural network definition under the comment "Configure Neural Network Here". We will use an instance of the class ***NeuralNetwork*** with 256 input nodes, 70 hidden nodes, 18 output nodes and a sigmoidal activation function to classify the inputs.

```
NeuralNetwork nn = new NeuralNetwork(Activator.ActivationFunction.Sigmoid, 256, 70,
NUM_CATEGORIES);
```

  Note that the constant *NUM_CATEGORIES* is defined in the class *ImageClassifier*. The number of nodes in the hidden layer is computed from the Geometric Pyramid Rule.

- ***Instantiate the back-propagation training algorithm*** and ask it to train the network with normalized training data, a learning rate of 0.01 and a maximum of 10000 epochs.

```
BackpropagationTrainer trainer = new BackpropagationTrainer(nn);
trainer.train(Utils.normalize(data, 0, 1), Utils.normalize(expected, 0, 1), 0.01, 10000);
```

- Add the following to the constructor of the class *ImageClassifier* and to test if the network is trained:

```
int testIndex = 30;
double[] result = nn.process(Utils.normalize(data[testIndex], 0, 1));
System.out.println("It's " + names[Utils.getMaxIndex(result)]);

for (int i = 0; i < result.length; i++){
  System.out.println(result[i]);
}

System.out.println(Utils.getMaxIndex(result) + 1);
```

- ***Execute the programme.*** The neural network should identify the image as a **pigeon**. Run the programme over and over and examine the result. ***Is the network stable?***

- ***Check the result for all four different images of the pigeon***. ***Can the neural network correctly classify the image?*** Change the image to the "moon" sprite and execute the programme again for all four type of this image. ***Explain the result.***

- ***Write a method that adds a small amount of noise*** to the input and examine the impact that this has on the classification. ***How robust is the network and what changes can be made to improve it?***