Introduction to neural networks and convolutional deep learning

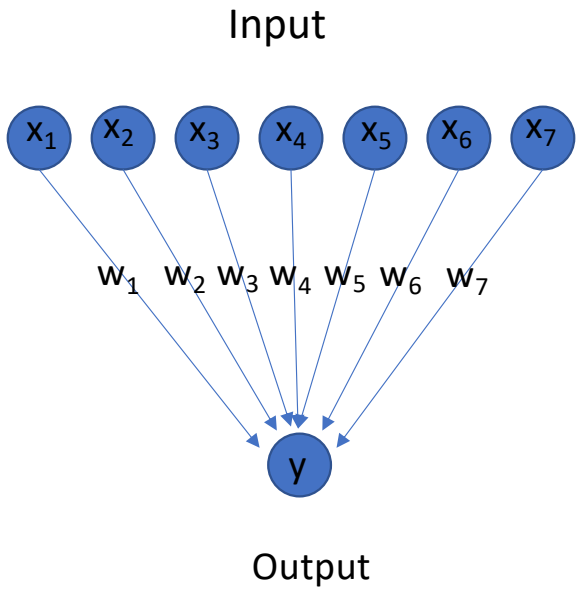
Jon Rowe

Artificial neurons

- Neural networks are made up of artificial neurons.
- An artificial neuron has several numerical inputs $x_1, \ldots, x_n$, and one numerical output, y.
- Each input has an associated ***weight*** $w_1, \ldots, w_n$
- There is also an ***activation function*** *f.*
- The output of the neuron is given by

$$y = f(\sum w_i x_i)$$

- We can combine neurons together to form more complex neural networks.

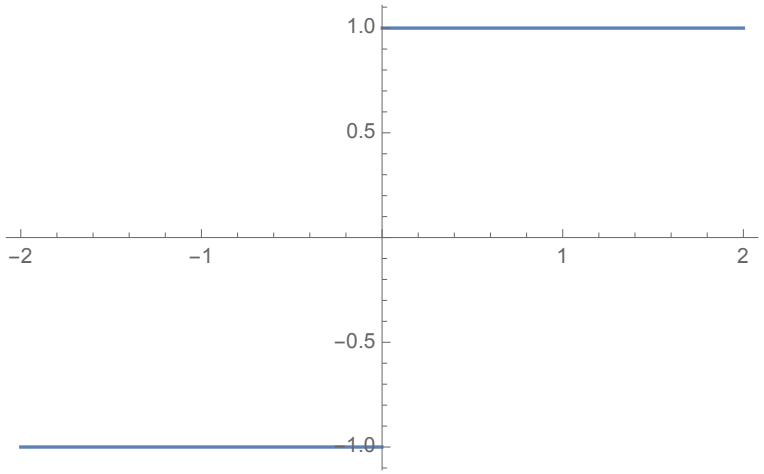


The learning and generalization problems

- Given a set of example inputs, and the corresponding output, we would like to find values for the weights so that the neural network gives the correct output for each input.
- Once we have found weights which work well on the training set, we would like the neural network to continue to work on other inputs which it has not been trained on. This is called ***generalization***.
- Sometimes we end up with a network that learns the training set perfectly, but doesn't generalize well – this is ***overfitting***.

Perceptrons

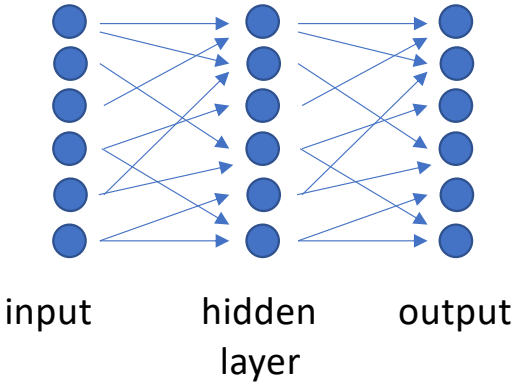
- A single artificial neuron, which uses a ***threshold activation function*** is called a ***perceptron***.

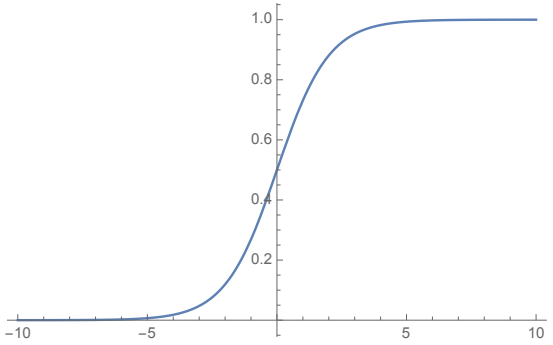


- Perceptrons can only learn certain functions (linearly separable).
- There is an algorithm that can find the right weights.

Multi-layer perceptrons

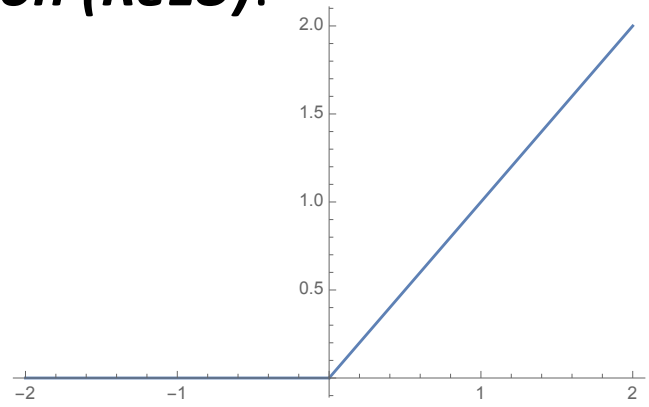
- We can connect artificial neurons together in layers to form ***multi-layer perceptrons (MLPs)***.
- These can represent many more functions than a single perceptron.
- The learning algorithm is called ***backpropagation***. It starts with random weights and adjusts them in small steps to reduce the error.
- MLPs typically use a ***sigmoid activation function***.





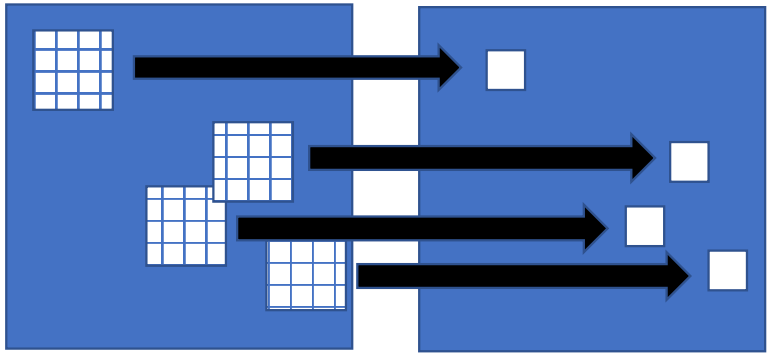
Deep neural nets

- Deep neural nets use many hidden layers.
- This allows them to represent much more complex functions.
- They take longer to train, and can suffer from ***overfitting***.
- They use backpropagation to learn, but use a ***rectified linear activation function (ReLU)***.



Working with images – convolution

- When working with image data sets, the important features are not just pixel values, but local patterns of neighbouring pixels.
- To capture these, we use ***convolution*** layers.
- These cover the image with small **overlapping** patches. There is one artificial neuron for each patch.
- The outputs retain the 2D structure of the image.

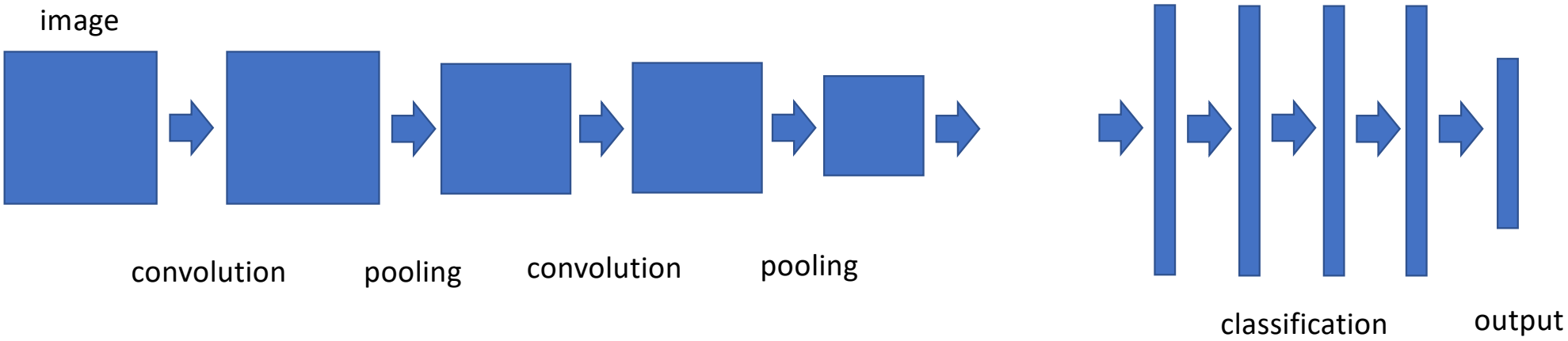


Working with images - pooling

- Visual features can occur at different scales in a picture.
- This might be because they are large features built up out of smaller ones. Or it might be that an object appears smaller or larger in different images.
- To handle this we use a ***pooling*** layer. These cover the image with small ***disjoint*** patches, with one neuron for each patch.
- The output is the ***average*** or ***maximum*** of the values in the patch.
- The outputs retain the 2D structure of the image, but this is now smaller than the original.

Convolutional neural nets

- A convolutional neural net has a number of convolution and pooling layers, followed by several layers which are complete connected.
- The convolution and pooling layers learn what the relevant visual features are.
- The final layers learn the appropriate classification.



Convolutional neural networks solved a long-standing problem in AI – how to recognize objects in images under many different conditions.

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.