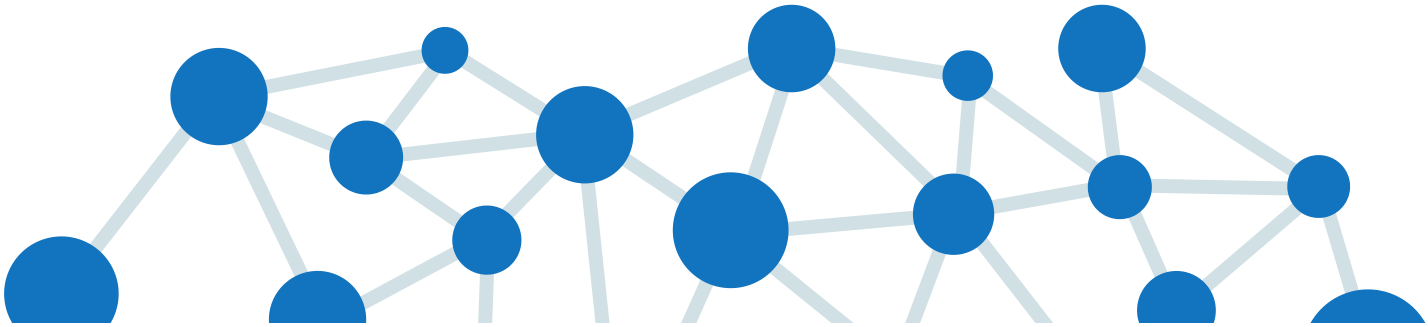
Elements of AI



III. Advanced neural network techniques

In the previous section, we have discussed the basic ideas behind most neural network methods: multilayer networks, non-linear activation functions, and learning rules such as the backpropagation algorithm.

They power almost all modern neural network applications. However, there are some interesting and powerful variations of the theme that have lead to great advances in deep learning in many areas.

Convolutional neural networks (CNNs)

One area where deep learning has achieved spectacular success is image processing. The simple classifier that we studied in detail in the previous section is severely limited – as you noticed it wasn't even possible to classify all the smiley faces correctly. By adding more layers in the network and using backpropagation to learn the weights does in principle solve the problem, but another one emerges: the number of weights becomes

extremely large and consequently, the amount of training data required to achieve satisfactory accuracy can become too large to be realistic.

Fortunately, a very elegant solution to the problem of too many weights exists: a special kind of neural networks, or rather, a special kind of layer that can be included in a deep neural network. This special kind of layer is a so called **convolutional layer**. Networks including convolutional layers are called **convolutional neural networks** (CNNs). Their key property is that they can detect image features such as bright or dark (or specific color) spots, edges in various orientations, patterns, and so on. These form the basis for detecting more abstract features such as a cat's ears, a dog's snout, a person's eye, or the octagonal shape of a stop sign. It would normally be hard to train a neural network to detect such features based on the pixels of the input image, because the features can appear in different positions, different orientations, and in different sizes in the image: moving the object or the camera angle will change the pixel values dramatically even if the object itself looks just the same to us. In order to learn to detect a stop sign in all these different conditions would require vast of amounts of training data because the network would only detect the sign in conditions where it has appeared in the training data. So, for example, a stop sign in the top right corner of the image would be detected only if the training data included an image with the stop sign in the top right corner. CNNs can recognize the object anywhere in the image no matter where it has been observed in the training images.

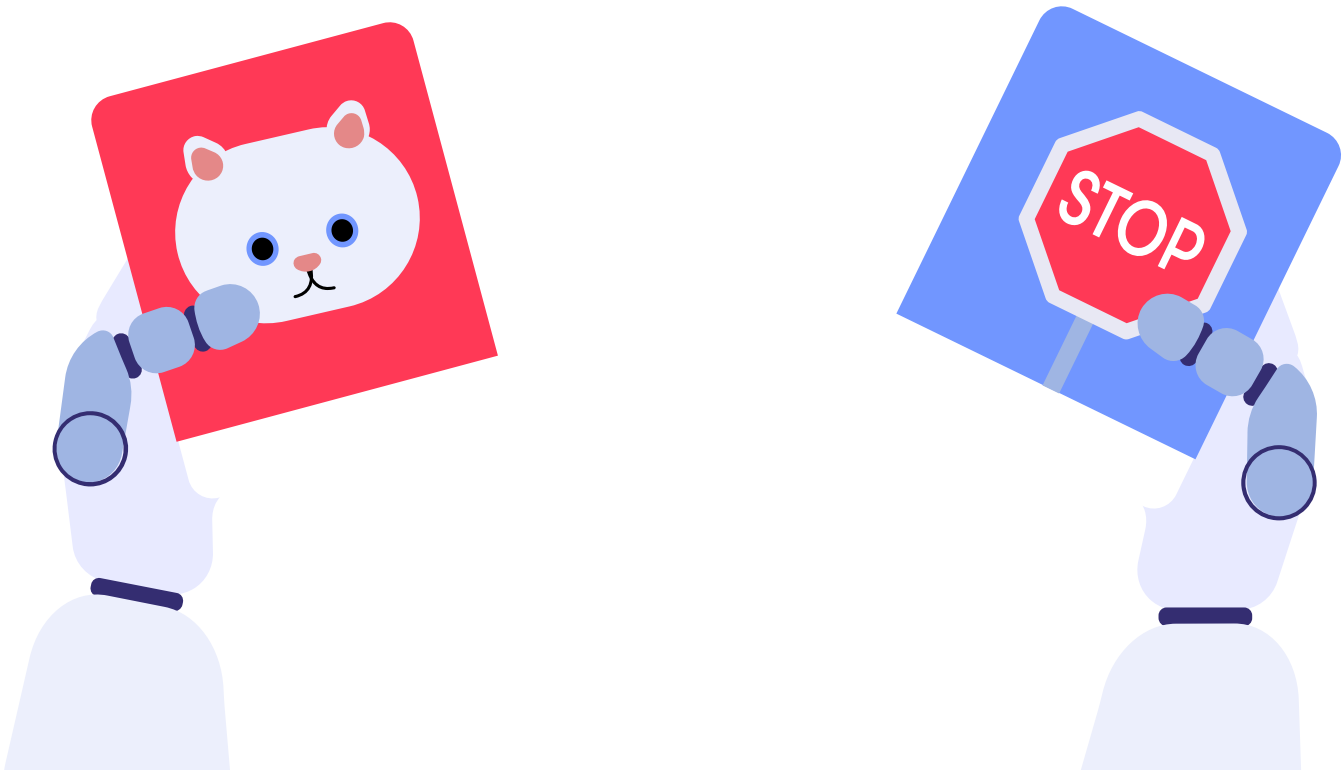
Note

Why we need CNNs

CNNs use a clever trick to reduce the amount of training data required to detect objects in different conditions. The trick basically amounts to using the same input weights for many neurons — so that all of these neurons are activated by the same pattern — but with different input pixels. We can for example have a set of neurons that are activated by a cat's pointy ear. When the input is a photo of a cat, two neurons are activated, one for the left ear and another for the right. We can also let the neuron's input pixels be taken from a smaller or a larger area, so that different neurons are activated by the ear appearing in different scales

(sizes), so that we can detect a small cat's ears even if the training data only included images of big cats.

The convolutional neurons are typically placed in the bottom layers of the network, which processes the raw input pixels. Basic neurons (like the perceptron neuron discussed above) are placed in the higher layers, which process the output of the bottom layers. The bottom layers can usually be trained using unsupervised learning, without a particular prediction task in mind. Their weights will be tuned to detect features that appear frequently in the input data. Thus, with photos of animals, typical features will be ears and snouts, whereas in images of buildings, the features are architectural components such as walls, roofs, windows, and so on. If a mix of various objects and scenes is used as the input data, then the features learned by the bottom layers will be more or less generic. This means that pre-trained convolutional layers can be reused in many different image processing tasks. This is extremely important since it is easy to get virtually unlimited amounts of unlabeled training data - images without labels - which can be used to train the bottom layers. The top layers are always trained by supervised machine learning techniques such as backpropagation.



Do neural networks dream of electric sheep? Generative adversarial networks (GANs)

Having learned a neural network from data, it can be used for prediction. Since the top layers of the network have been trained in a supervised manner to perform a particular classification or prediction task, the top layers are really useful only for that task. A network trained to detect stop signs is useless for detecting handwritten digits or cats.

A fascinating result is obtained by taking the pre-trained bottom layers and studying what the features they have learned look like. This can be achieved by generating images that activate a certain set of neurons in the bottom layers. Looking at the generated images, we can see what the neural network “thinks” a particular feature looks like, or what an image with a select set of features in it would look like. Some even like to talk about the networks “dreaming” or “hallucinating” images (see Google’s DeepDream system).

Note

Be careful with metaphors

However, we'd like to once again emphasize the problem with metaphors such as dreaming when simple optimization of the input image is meant — remember the suitcase words discussed in Chapter 1. The neural network doesn't really dream, and it doesn't have a concept of a cat that it would understand in a similar sense as a human understands. It is simply trained to recognize objects and it can generate images that are similar to the input data that it is trained on.

To actually generate real looking cats, human faces, or other objects (you'll get whatever you used as the training data), Ian Goodfellow who currently works at Google Brain, proposed a clever combination of two neural networks. The idea is to let the two networks compete against each other. One of the networks is trained to generate images like the ones in the training data. The other network's task is to separate images generated by the first network from real images from the training data — it is called the adversarial network, and the whole system is called generative adversarial network or a GAN.

The system trains the two models side by side. In the beginning of the training, the adversarial model has an easy task to tell apart the real images from the training data and the clumsy attempts by the generative model. However, as the generative network slowly gets better and better, the adversarial model has to improve as well, and the cycle continues until eventually the generated images are almost indistinguishable from real ones. The GAN tries to not only reproduce the images in the training data: that would be a way too simple strategy to beat the adversarial network. Rather, the system is trained so that it has to be able to generate new, real-looking images too.


The above images were generated by a GAN developed by NVIDIA in a project led by Prof Jaakko Lehtinen (see this article for more details).

Could you have recognized them as fakes?

After completing Chapter 5 you should be able to:

- Explain what a neural network is and where they are being successfully used
- Understand the technical methods that underpin neural networks

Please join the Elements of AI community at Spectrum to discuss and ask questions about this chapter.

Give us your feedback on this part

Write your feedback here..

Submit

You reached the end of Chapter 5!

Correct answers

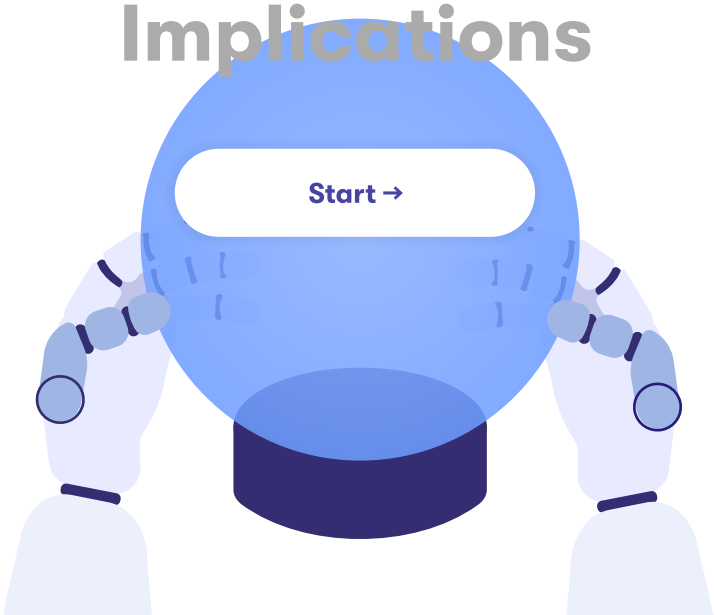
0 %

Exercises completed

0 /25

Next Chapter

Implications



Course overview

About

FAQ

Privacy Policy

My profile Sign out