UNIT-IV

ACCELERATING DEEP LEARNING MODELS

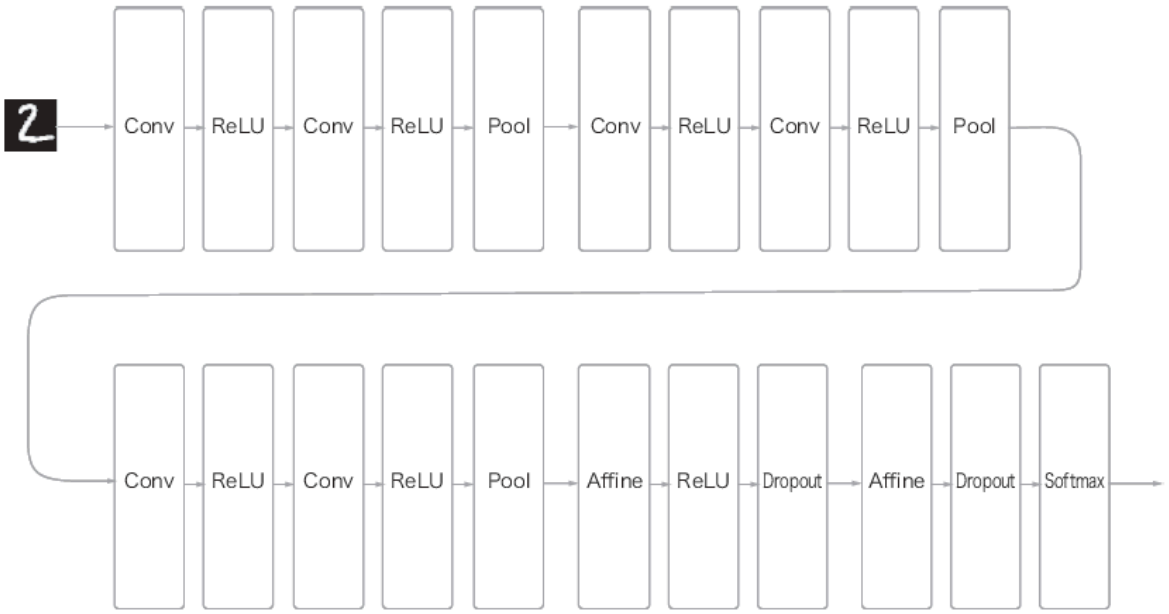
Making a Network Deeper – ImageNet – VGG – GoogLeNet – ResNet – Accelerating Deep Learning – Practical Uses of Deep Learning – The Future of Deep Learning. CHAPTER – 8 (T1)

Deep learning is a machine learning method based on deep neural networks.

Making a Network Deeper

Deeper Networks

First, we will create a CNN that has the network architecture



Deep CNN for handwritten digit recognition

All the convolution layers used here are small 3x3 filters. Here, the number of channels becomes larger as the network deepens (as the number of channels in a convolution layer increases from 16 in the first layer to 16, 32, 32, 64, and 64). As you can see, pooling layers are inserted to reduce the spatial size of intermediate data gradually, while dropout layers are used for the latter fully connected layers

This network uses the "He initializer" to initialize the weights, and Adam to update the weight parameters, resulting in the following characteristics:

- Convolution layers which use small 3×3 filters
- ReLU as the activation function
- A dropout layer used after a fully connected layer

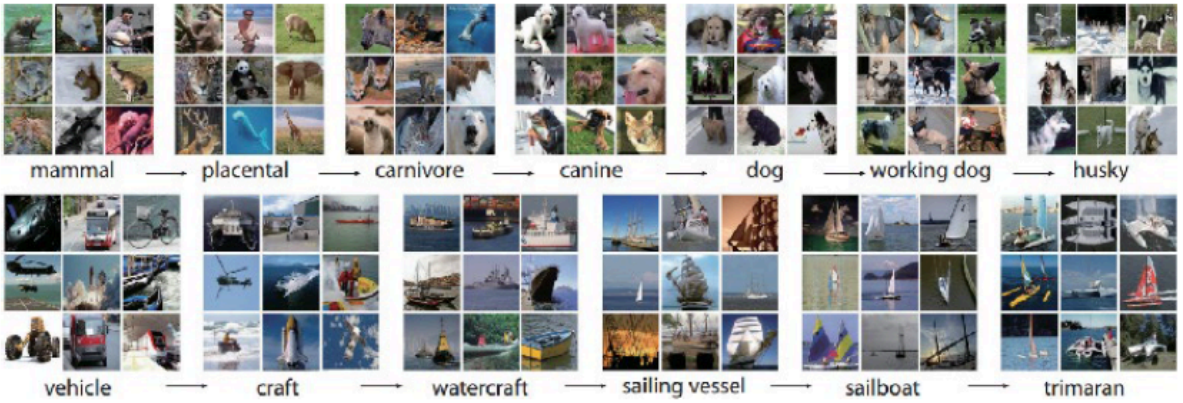
• Optimization is done by Adam
• "He initializer" for initial weight values

Note

The source code that implemented the network shown in *Figure 8.1* is located at **ch08/deep_convnet.py**. The code for training is provided at **ch08/ train_deepnet.py**. You can use this code to reproduce the training that will be conducted here. Training in a deep network takes a lot of time (probably more than half a day). This book provides trained weight parameters in **ch08/deep_ conv_net_params.pkl**. The **deep_convnet.py** code file provides a feature for loading trained parameters. You can use it as required.

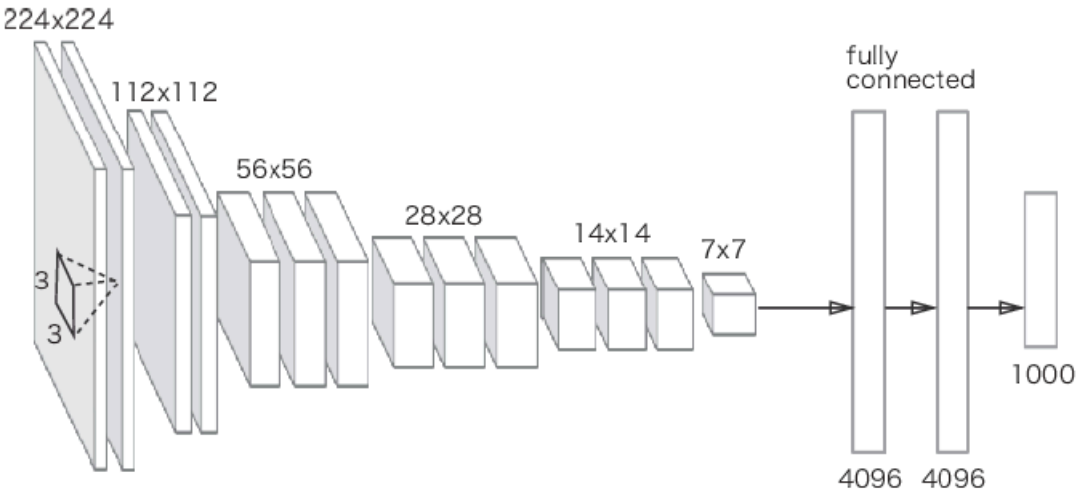
ImageNet

ImageNet (*J. Deng, W. Dong, R. Socher, L.J. Li, Kai Li, and Li Fei-Fei (2009): ImageNet: A large-scale hierarchical image database*. In IEEE Conference on *Computer Vision and Pattern Recognition, 2009. CVPR 2009. 248 – 255. DOI*: (http://dx.doi.org/10.1109/CVPR.2009.5206848)) is a dataset that contains more than 1 million images. As shown in *Figure 8.7*, it contains various types of images, and each image is associated with a label (class name). An image recognition competition called ILSVRC is held every year using this huge dataset:



VGG

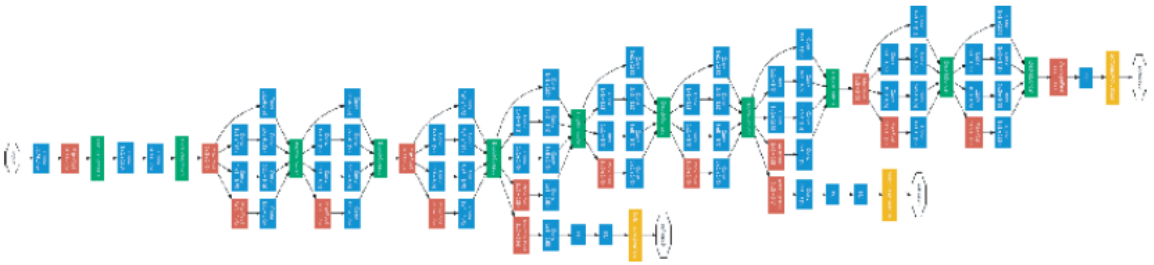
VGG is a "basic" CNN that consists of convolution layers and pooling layers. As shown in *Figure* , it can have as many as 16 (or 19) layers with weights (convolution layers and fully connected layers) to make itself deep and is sometimes called "VGG16" or "VGG19" based on the number of layers:



VGG contains consecutive convolution layers with a small 3x3 filter. As shown in the preceding image, two or four consecutive convolution layers and a pooling layer halve the size, and this process is repeated. Finally, the result is provided via fully connected layers.

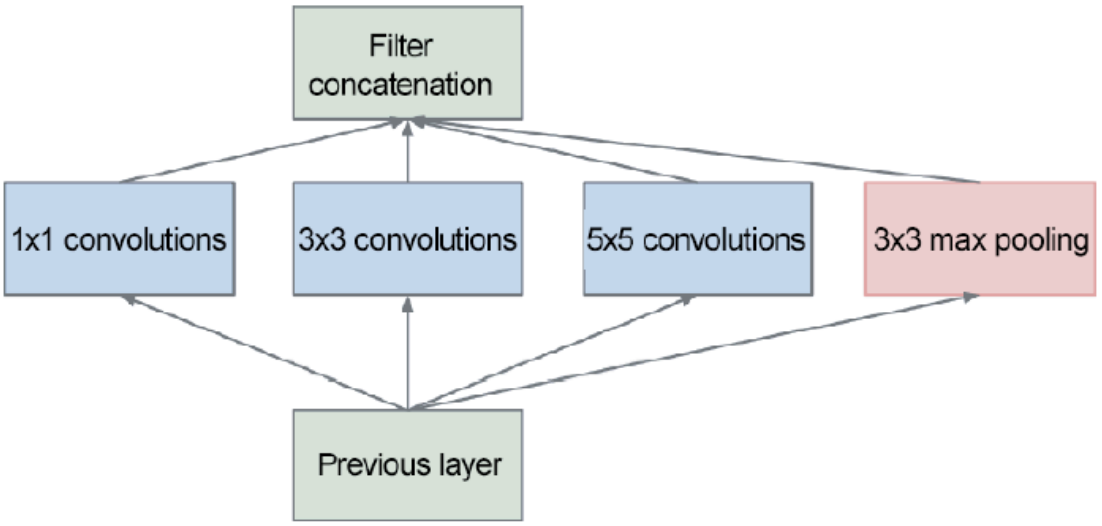
GoogLeNet

Figure shows the network architecture for GoogLeNet. The rectangles represent the various layers, such as convolution and pooling layers:



Its network architecture seems very complicated when you look at it, but it is basically the same as that of a CNN. What is distinctive about GoogLeNet is that the network not only has depth in the vertical direction but also in the horizontal direction (spread).

GoogLeNet has "width" in the horizontal direction. It is called an "inception architecture" and is based on the structure shown in *Figure*



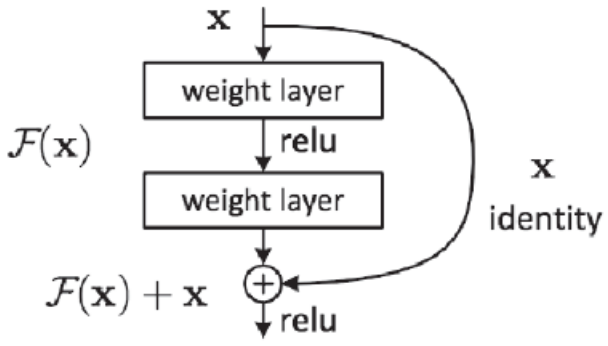
Inception architecture of GoogLeNet

the inception architecture applies multiple filters of different sizes (and pooling) and combines the results. Using this inception architecture as one building block (component) is the main characteristic of GoogLeNet.
GoogLeNet uses convolution layers with a 1x1 filter in many places. This 1x1 convolution operation reduces the size in the channel direction to reduce the number of parameters and accelerate processing.

ResNet

ResNet (*Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2015): Deep Residual Learning for Image Recognition. arXiv:1512.03385[cs] (December 2015)*) is a network developed by a team at Microsoft. It is characterized by a "mechanism" that can make the network deeper than ever.

Making a network deeper is important to improve its performance. However, when a network becomes too deep, deep learning fails and the final performance is often poor. To solve this problem, ResNet introduced a "skip architecture" (also called "shortcut" or "bypass"). By introducing this skip architecture, performance can be improved as the network becomes deeper (though there is a limit to permissible depth).
The skip architecture skips convolution layers in the input data to add the input data to the output, as shown in *Figure*

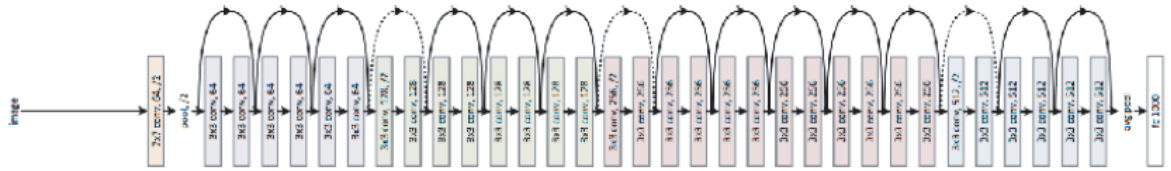


Components of ResNet – the "weight layer" here indicates a convolution layer

the input, *x*, is connected to the output by skipping two consecutive convolution layers. The output of two convolution layers is originally *F(x)*, while the skip architecture changes it to *F(x)* + *x*.

Adopting this skip architecture enables efficient learning, even when the network is deep. This is because the skip architecture transmits signals without decay during backward propagation.

ResNet is based on the VGG network we described earlier and adopts the skip architecture to make the network deeper



ResNet – blocks support 3x3 convolution layers. Its characteristic is the skip architecture, which skips layers.

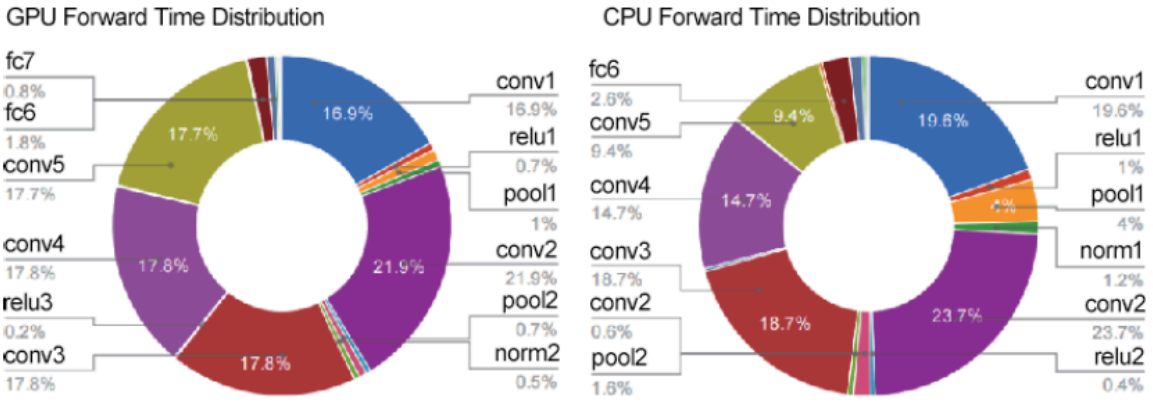
ResNet skips two convolution layers to make the network deeper. Experiments have shown that recognition accuracy continues to improve, even when the network contains 150 or more layers. In the ILSVRC competition, it achieved an amazing result of 3.5% in terms of error rate (the percentage of correct classes that were not included in the top 5 predictions).

Note
Weight data that's trained by using the huge ImageNet dataset is often used effectively. This is called **transfer learning**

Accelerating Deep Learning

Big data and large-scale networks require massive operations in deep learning. We have used CPUs for calculations so far, but CPUs alone are not sufficient to tackle deep learning. In fact, many deep learning frameworks support **Graphics Processing Units** (**GPUs**) to process a large number of operations quickly. Recent frameworks are starting to support distributed learning by using multiple GPUs or machines. This section describes accelerating calculations in deep learning

Challenges to Overcome

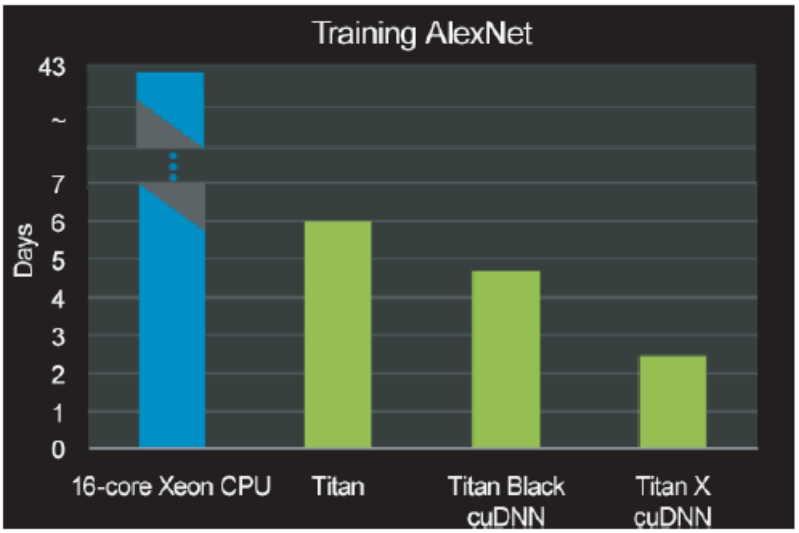


Percentage of time that each layer spends in the forward processing of AlexNet – the left-hand chart shows GPU time, while the right-hand one shows CPU time

Using GPUs for Acceleration

Originally, GPUs were exclusively used for graphics. Recently, they have been used for general numerical calculations, as well as graphics processing. Because GPUs can conduct parallel arithmetic operations quickly, GPU computing uses its overwhelming power for various purposes.

Deep learning requires massive multiply-accumulate operations (or products of large matrices). GPUs are good at such massive parallel operations, while CPUs are good at continuous and complicated calculations. You can use a GPU to accelerate deep learning operations surprisingly compared to using only a CPU. *Figure 8.15* compares the time that AlexNet took for learning between a CPU and a GPU



Comparing the time that AlexNet took for learning between a "16-core Xeon CPU" and a "Titan series" GPU

the CPU took more than 40 days, while the GPU took only 6 days. We can also see that using the cuDNN library, which is optimized for deep learning, accelerates the training further

GPUs are mainly provided by two companies, NVIDIA and AMD. Although you can use both of their GPUs for general arithmetic operations, NVIDIA's GPUs are more "familiar" with deep learning. Actually, many deep learning frameworks can benefit only from NVIDIA's GPUs. This is because CUDA, which is an integrated development environment for GPU computing provided by NVIDIA, is used in deep learning frameworks. cuDNN, which can be seen in *Figure* , is a library that runs on CUDA in which the functions optimized for deep learning are implemented.

Distributed Training

to reduce the time required for training as much as possible. Then, scaling deep learning out or "distributed training" becomes important.

To further accelerate the calculations required for deep learning, you may want to distribute them among multiple GPUs or machines. Now, some deep learning frameworks support distributed training by multiple GPUs or machines. Among them, Google's TensorFlow and Microsoft's **Computational Network Toolkit** (**CNTK**) have been developed to focus on distributed training. Based on low-delay and high-throughput networks in huge data centers, distributed training by these frameworks achieves surprising results.

How much can distributed training accelerate deep learning? The answer is that the larger the number of GPUs, the faster the training speed. In fact, 100 GPUs (a total of 100 GPUs installed on multiple machines) achieves a 56-fold speedup compared with one GPU. This means that training that usually takes 7 days is completed in only 3 hours, for example, and indicates the surprising effect of distributed training.

"How to distribute calculations" in distributed training is a very difficult problem. It contains many problems that are not easy to solve, such as communication and data synchronization between machines. You can leave such difficult problems to excellent frameworks such as TensorFlow. Here, we will not discuss the details of distributed training. For the technical details of distributed training, please see the technical paper (white paper) about TensorFlow (*Mart í n Abadi et al. (2016): TensorFlow: Large- Scale Machine Learning on Heterogeneous Distributed Systems. arXiv:1603.04467[cs] (March 2016)*).

Reducing the Bit Number for Arithmetic Precision

A computer mainly uses 64- or 32-bit floating-point numbers to represent real numbers. Using many bits to represent a number reduces the influence of the error at numerical calculation but increases the processing cost and memory usage, placing a load on the bus bandwidth.

From what we know about deep learning regarding numerical precision (how many bits are used to represent a numeric value), it does not need very high precision. This is one of the

most important characteristics of a neural network due to its robustness. The robustness here means that, for example, the output result will not change in a neural network, even if the input images contain a small amount of noise. Think of it as a small influence on the output result because of the robustness, even if the data flowing in a network is "deteriorated."
A computer usually uses 32-bit single-precision floating-point representations or 64-bit double-precision floating-point representations to represent a decimal. Experiments have shown that 16-bit **half-precision floating-point representations** (half **float**) are sufficient in deep learning

Practical Uses of Deep Learning

Object Detection

Object detection identifies the positions of objects in images and classifies them. Object detection is more difficult than object recognition. While object recognition targets the entire image, object detection must identify the positions of classes in an image, and multiple objects may exist.

Among CNN-based object detection techniques, a technique called R-CNN is famous

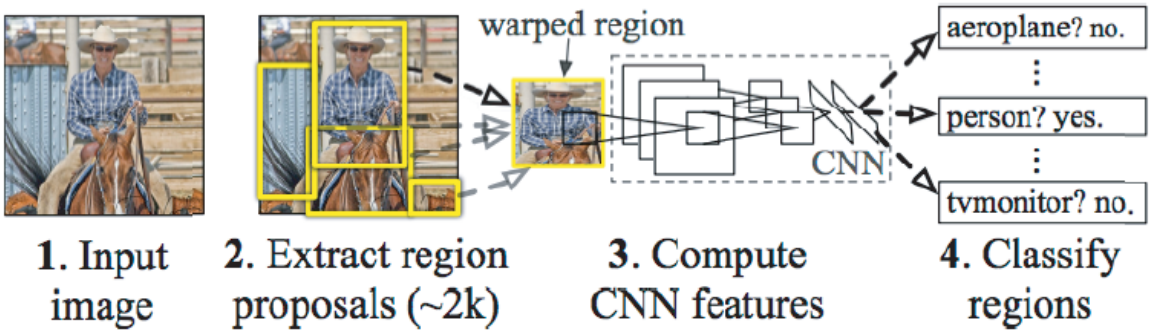


Figure 8.16: Process flow of R-CNN

Process flow of R-CNN

2. Extract region proposals and *3. Compute CNN features* sections. The first technique detects the areas that seem to be objects (in some way) and then applies a CNN to the extracted areas to classify them. R-CNN converts an image into squares and uses **support vector machines** (**SVMs**) for classification. Its actual process flow is slightly complicated but mainly consists of the aforementioned processes: the extraction of candidate regions and to compute CNN features.
In the "Extract region proposals" process of R-CNN, candidates for objects are detected, and this is where various techniques that have been developed in computer vision can be used.

Segmentation

Segmentation classifies an image on a pixel basis. It learns by using training data where objects are colored on a pixel basis and classifies all the pixels of an input image during

inference. The neural networks we've implemented so far classify the entire image. So, how can we classify it on a pixel basis?

The simplest method of performing segmentation with a neural network is to make a prediction for each pixel.

Example of segmenting an image by using deep learning – the road, cars, buildings, and sidewalks are recognized accurately

Generating Image Captions

There is some interesting research being conducted that combines natural language and computer vision. When an image is provided, the text explaining the image (the image caption) is automatically generated.

For example, an image of a motorcycle from a dirt bike competition could include the caption: "A person riding a motorcycle on a dirt road" (this text is automatically generated from the image). It is surprising that the system even "understands" that it is on a dirt road and that a person is riding a motorcycle.

A model called **Neural Image Caption** (**NIC**) is typically used to generate image captions for deep learning. NIC consists of a deep CNN and a **Recurrent Neural Network** (**RNN**) for handling natural language. An RNN has recursive connections and is often used for sequential data such as natural language and time-series data.

The Future of Deep Learning

Converting Image Styles

There is research being conducted that uses deep learning to "draw" a picture as an artist would. One popular use case of neural networks is to create a new image based on two provided images. One of them is called a "content image," while the other is called a "style image." A new image is created based on these two images.

Generating Images

The preceding example of image style transfer required two images to generate a new image. On the other hand, some research has tried to generate new images without requiring any images (the technique trains by using many images beforehand but needs no images to "draw" a new image.) For example, you can use deep learning to generate the image of a "bedroom" from scratch

They may seem to be real photographs, but they were newly generated by a DCGAN. The images that were generated by the DCGAN are images that nobody has ever seen (those that do not exist in the training data) and were newly created from scratch.

When a DCGAN generates images that look like real ones, it creates a model of the process where the images were generated. The model learns by using many images (such as those of bedrooms). After training finishes, you can use the model to generate new images.
DCGANs use deep learning. The main point of the DCGAN technique is that it uses two neural networks: a generator and a discriminator. The generator generates an image that seems real, while the discriminator determines whether it is real, that is, whether it was generated by the generator or whether it was really photographed. In this way, two networks are trained by making them compete against each other.

The generator learns a more elaborate technique of creating fake images, while the discriminator grows like an appraiser who can detect fakes with higher precision. What is interesting is that in a technology called a **Generative Adversarial Network** (**GAN**), both of them grow through competition. Finally, the generator that has grown through competition can draw images that look real (or may grow even more).

Automated Driving

"Automated driving" technology, in which a computer drives a car instead of a human, is likely to be realized soon. IT companies, universities, and research institutions, as well as car manufacturers, are competing to realize automated driving. This can only happen when various technologies such as path plan technology, which determines a traffic route, and sensing technology, including cameras and lasers, are combined. It is said that the technology used to recognize the surrounding environment properly is the most important. It is very difficult to recognize an environment that changes every moment of every day, as well as the cars and people that move around freely.
If the system can properly recognize the travel area robustly and reliably, even in various environments, automated driving may be realized in the near future—a task for which deep learning should prove invaluable.

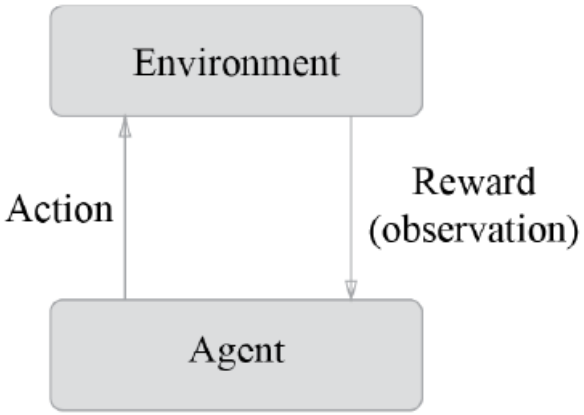
Example of segmenting an image by using deep learning – the road, cars, buildings, and sidewalks are recognized accurately

The result indicates that the road, buildings, sidewalks, trees, cars, and motorcycles are distinguished somewhat accurately. If deep learning improves the accuracy and speed of these recognition technologies from now on, automated driving may be put into practical use in the not too distant future.

Deep Q-Networks (Reinforcement Learning)

There is a research field called **reinforcement learning** in which computers learn independently through trial and error, just as humans learn how to ride a bicycle, for example. This is different from "supervised learning," where a "supervisor" teaches face to face.

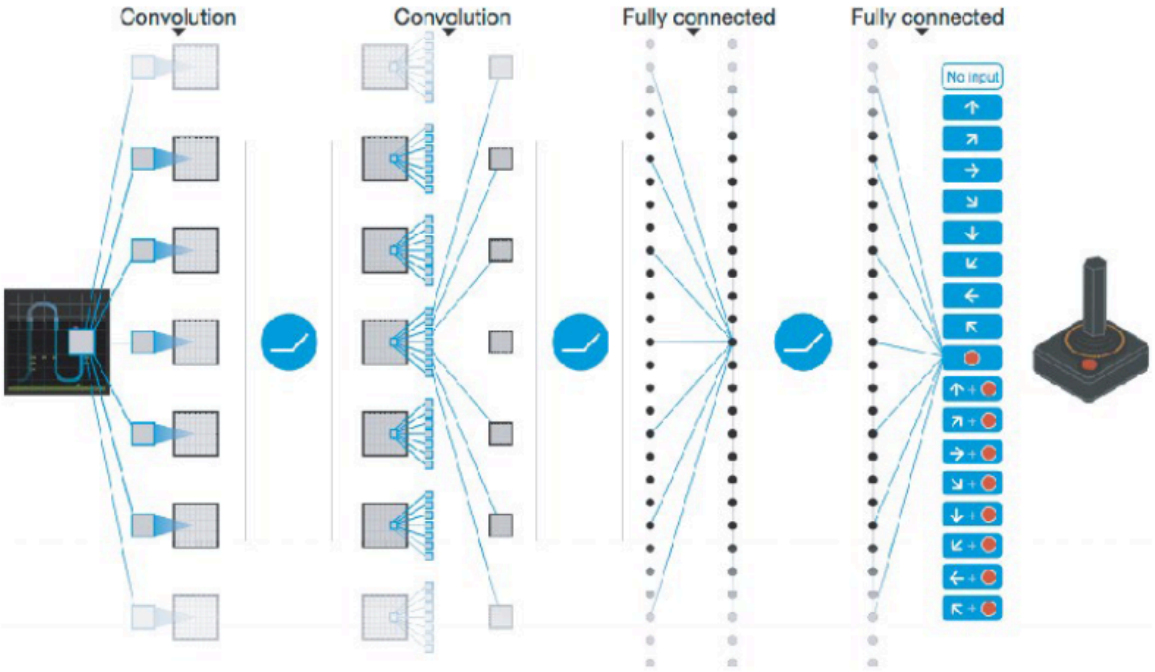
The basic framework of reinforcement learning is that an agent selects actions, depending on the situation of the environment, and its actions change the environment. After taking an action, the environment offers the agent some reward. The purpose of reinforcement learning is to determine the action policy of the agent so that it can obtain a better reward, as shown here



Basic framework of reinforcement learning – the agent learns independently to obtain a better reward

the reward is not labeled data, as it is in supervised learning. For example, in the video game "Super Mario Brothers," the exact quantity of rewards you gain by moving Mario to the right is not necessarily clear. In that case, the "prospective" reward must be determined by clear indicators such as the game scores (obtaining coins, defeating enemies, and so on) and game-over logic. In supervised learning, each action can be evaluated correctly by the "supervisor."

A **Deep Q-Network** (**DQN**) is a reinforcement learning technique (*Volodymyr Mnih et al (2015): Human-level control through deep reinforcement learning. Nature 518, 7540 (2015), 529 – 533*) that uses deep learning. It is based on the algorithm of reinforcement learning called Q-learning. Q-learning determines a function called the optimal action-value function to determine the optimal action. A DQN uses deep learning (CNNs) to approximate the function.



Using a Deep Q-Network to learn the operations of a video game. Here, the network receives the images of a video game as an input and learns the operation of the game controller (joystick) through trial and error