

# Deep Learning (Fall 2023)

**Ikbeom Jang**

**[ijang@hufs.ac.kr](mailto:ijang@hufs.ac.kr)**

**CES HUFS**

# Contents

- **Background**
- **Installation**

# Background

---

- Imagine just writing a program to respond to a wake word such as “Alexa”, “OK Google”, and “Hey Siri”.
- Try coding it up in a room by yourself with nothing but a computer and a code editor, as illustrated in Fig. 1.1.1. How would you write such a program from first principles?
- Think about it... the problem is hard. Every second, the microphone will collect roughly 44,000 samples. Each sample is a measurement of the amplitude of the sound wave.
- What rule could map reliably from a snippet of raw audio to confident predictions {yes, no} about whether the snippet contains the wake word?
- If you are stuck, do not worry. We do not know how to write such a program from scratch either. That is why we use machine learning (ML).

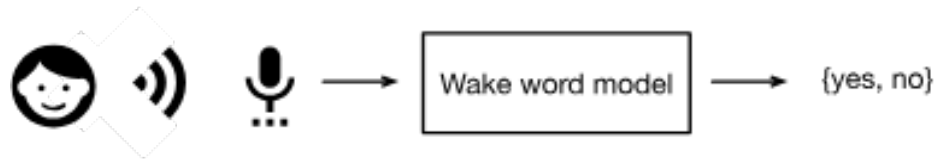


Fig. 1.1.1 Identify a wake word

# Background

- In the currently dominant approach to ML, we do not attempt to design a system explicitly to recognize wake words.
- Instead, we define a flexible program whose behavior is determined by a number of parameters.
- Then we use the dataset to determine the best possible parameter values, i.e., those that improve the performance of our program with respect to a chosen performance measure.
- You can think of the parameters as knobs that we can turn, manipulating the behavior of the program.
- Once the parameters are fixed, we call the program a **model**.
- The set of all distinct programs (input–output mappings) that we can produce just by manipulating the parameters is called a **family of models**.
- And the “meta-program” that uses our dataset to choose the parameters is called a **learning algorithm**.

인공 지능  
시스템  
학습  
과정

파라미터 조정  
과정

# Background

- We train our model with data.
- The training process usually looks like the following:
  1. Start off with a randomly initialized model that cannot do anything useful. *model (생성)*
  2. Grab some of your data (e.g., audio snippets and corresponding {yes,no} labels). *데이터를 랜덤하게*
  3. Tweak the knobs to make the model perform better as assessed on those examples. *조정*
  4. Repeat Steps 2 and 3 until the model is awesome.

*결과를 시각화함  
결과보고 파라미터 수정*

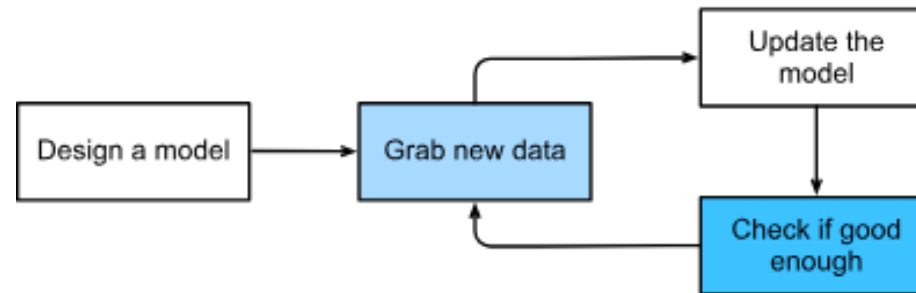


Fig. 1.1.2 A typical training process

# Background

---

- In our wake word example, we described a dataset consisting of audio snippets and binary labels, and we gave a hand-wavy sense of how we might train a model to approximate a mapping from snippets to classifications.
- This sort of problem, where we try to predict a designated unknown label based on known inputs given a dataset consisting of examples for which the labels are known, is called **supervised learning**. This is one among many kinds of machine learning problems.

- **Key Components:** *머신러닝 학습용 중요한 conference*
  - The **data** that we can learn from. *유익한 중요한 가치가 있는 데이터 필요. 쓰러지거나 깨지는 데이터 필요.*
  - A **model** of how to transform the data. *변형시켜주는 model 선정*
  - An **objective function** that quantifies how well (or badly) the model is doing. *model이 얼마나 잘하는지*
  - An **algorithm** to adjust the model's parameters to optimize the objective function. *함께 잘되도록 파악*

# Background

- Kinds of Machine Learning Problems

- **Supervised Learning**

- Regression
- Classification
- Tagging — 이미지와 있으면 원리 파악해줄것 세팅이 안된 원리
- Search 사용자에게 이벤트를 추천해줄것임
- Recommender Systems
- Sequence Learning 언어 시계열 등

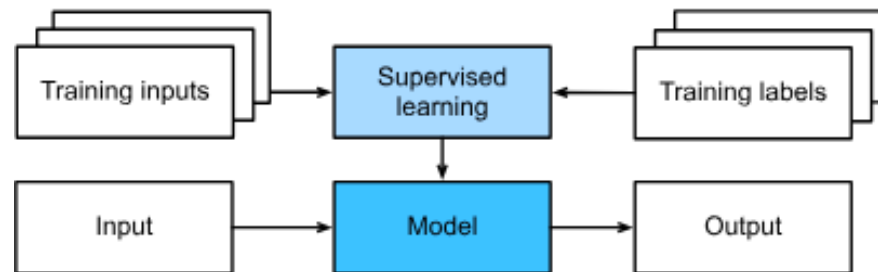


Fig. 1.3.1 Supervised learning.

# Background

- Kinds of Machine Learning Problems

- **Unsupervised Learning**

- Your boss just hand you a giant dump of data and tell you to do some data science with it

- **Self-Supervised Learning**

- Techniques that leverage some aspect of the unlabeled data to provide supervision.
- For text, we can train models to “fill in the blanks” by predicting randomly masked words using their surrounding words (contexts) in big corpora without any labeling effort.
- For images, we may train models to tell the relative position between two cropped regions of the same image, to predict an occluded part of an image based on the remaining portions of the image, or to predict whether two examples are perturbed versions of the same underlying image.
- Self-supervised models often learn representations that are subsequently leveraged by fine-tuning the resulting models on some downstream task of interest.

레이블이 레이블링이 있지 않음 레이블을 만들지 않음 ex) I play a Golf on Wednesday swimming

미리조각이 잘 안되는 경우 fine tuning을 해줘야 잘되는 경우가 많음.



# Background

---

- Kinds of Machine Learning Problems
  - **Interacting with an Environment**

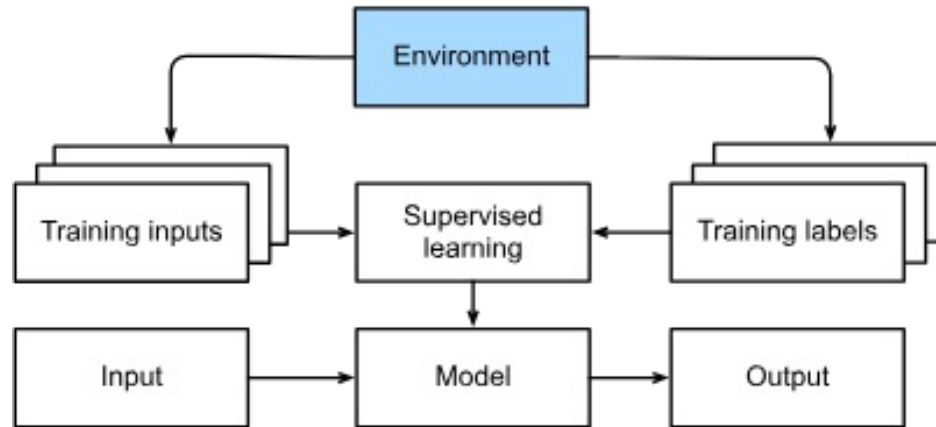


Fig. 1.3.6 Collecting data for supervised learning from an environment.

# Background

---

- Kinds of Machine Learning Problems

- **Reinforcement Learning**

*reward 0/1*

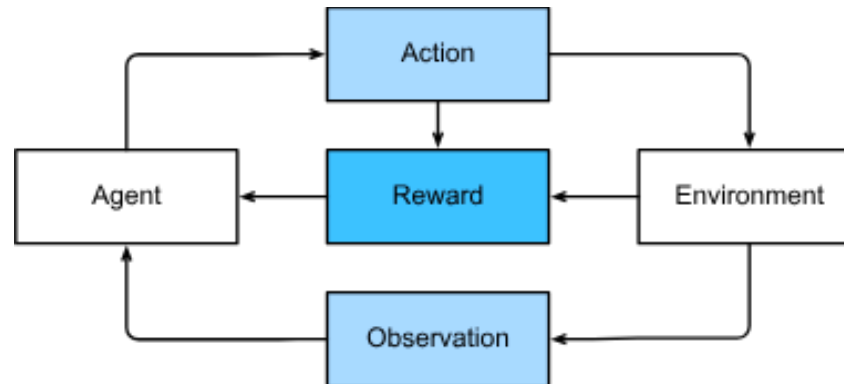


Fig. 1.3.7 The interaction between reinforcement learning and an environment.

# Background

---

- Deep learning (DL) became successful with
  - the availability of massive amounts of data, thanks to the World Wide Web
  - the advent of companies serving hundreds of millions of users online
  - a dissemination of low-cost, high-quality sensors, inexpensive data storage (Kryder's law), and cheap computation (Moore's law). *→ 더 빠른 하드웨어*
- In particular, the landscape of computation in DL was revolutionized by advances in GPUs that were originally engineered for computer gaming. *CPU vs GPU  
정확도 측면에서는 크게 차이 나지 않음*
- Suddenly algorithms and models that seemed computationally infeasible were within reach. *안타깝게도*

# Background

- Dataset vs. computer memory & computational power

Table 1.5.1 label:tab\_intro\_decade

Decade	Dataset	Memory	Floating point calculations per second
1970	100 (Iris)	1 KB	100 KF (Intel 8080)
1980	1 K (house prices in Boston)	100 KB	1 MF (Intel 80186)
1990	10 K (optical character recognition)	10 MB	10 MF (Intel 80486)
2000	10 M (web pages)	100 MB	1 GF (Intel Core)
2010	10 G (advertising)	1 GB	1 TF (NVIDIA C2050)
2020	1 T (social network)	100 GB	1 PF (NVIDIA DGX-2)

# Installation

---

## Installation

- In order to get up and running, we will need an environment for running Python, the Jupyter Notebook, the relevant libraries, and the code needed to run the book itself.


## Installing Miniconda

- Your simplest option is to install Miniconda. Note that the Python 3.x version is required. You can skip the following steps if your machine already has conda installed.
- Visit the Miniconda website and determine the appropriate version for your system based on your Python 3.x version and machine architecture. Suppose that your Python version is 3.9 (our tested version). If you are using macOS, you would download the bash script whose name contains the strings “MacOSX”, navigate to the download location, and execute the installation as follows (taking Intel Macs as an example):

# Installation


---

```
# The file name is subject to changes  
sh Miniconda3-py39_4.12.0-MacOSX-x86_64.sh -b
```



A Linux user would download the file whose name contains the strings “Linux” and execute the following at the download location:

```
# The file name is subject to changes  
sh Miniconda3-py39_4.12.0-Linux-x86_64.sh -b
```



A Windows user would download and install Miniconda by following its [online instructions](https://docs.conda.io/projects/miniconda/en/latest/). On Windows, you may search for cmd to open the Command Prompt (command-line interpreter) for running commands.

<https://docs.conda.io/projects/miniconda/en/latest/>

# Installation

---

Next, initialize the shell so we can run conda directly.

```
~/miniconda3/bin/conda init
```



Then close and reopen your current shell. You should be able to create a new environment as follows:

```
conda create --name d2l python=3.9 -y
```



Now we can activate the d2l environment:

```
conda activate d2l
```



# Installation

## Installing the Deep Learning Framework and the d2l Package

Before installing any deep learning framework, please first check whether or not you have proper GPUs on your machine (the GPUs that power the display on a standard laptop are not relevant for our purposes). For example, if your computer has NVIDIA GPUs and has installed [CUDA](#), then you are all set. If your machine does not house any GPU, there is no need to worry just yet. Your CPU provides more than enough horsepower to get you through the first few chapters. Just remember that you will want to access GPUs before running larger models.

PYTORCH

MXNET

JAX

TENSORFLOW

You can install PyTorch (the specified versions are tested at the time of writing) with either CPU or GPU support as follows:

```
pip install torch==2.0.0 torchvision==0.15.1
```



Our next step is to install the d2l package that we developed in order to encapsulate frequently used functions and classes found throughout this book:

```
pip install d2l==1.0.3
```





# Installation

---

Download Notebooks from eclass, and then

start the Jupyter Notebook server by running:

```
jupyter notebook
```



At this point, you can open <http://localhost:8888> (it may have already opened automatically) in your

# Preliminaries

---

- Data Manipulation
- Data Preprocessing
- Linear Algebra
- Calculus
- Automatic Differentiation
- Probability and Statistics