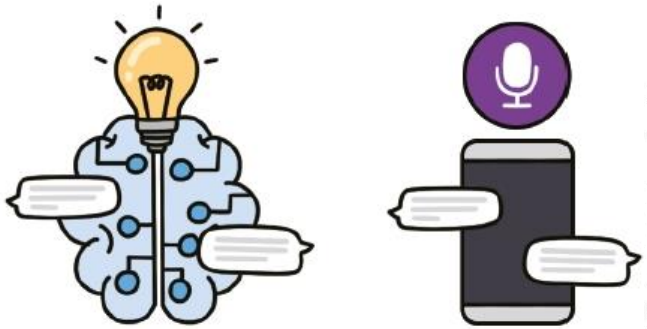
What is Machine Learning
(Basic Machine Learning)

Introduction

❖ Concepts of AI, ML, and DL

- Artificial intelligence
  - A technology that embodies the intellectual abilities of humans through computers
- Classification of artificial intelligence
  - Strong AI: AI with performance beyond human capabilities
  - Weak AI: AI designed for use as a tool in certain areas



Introduction

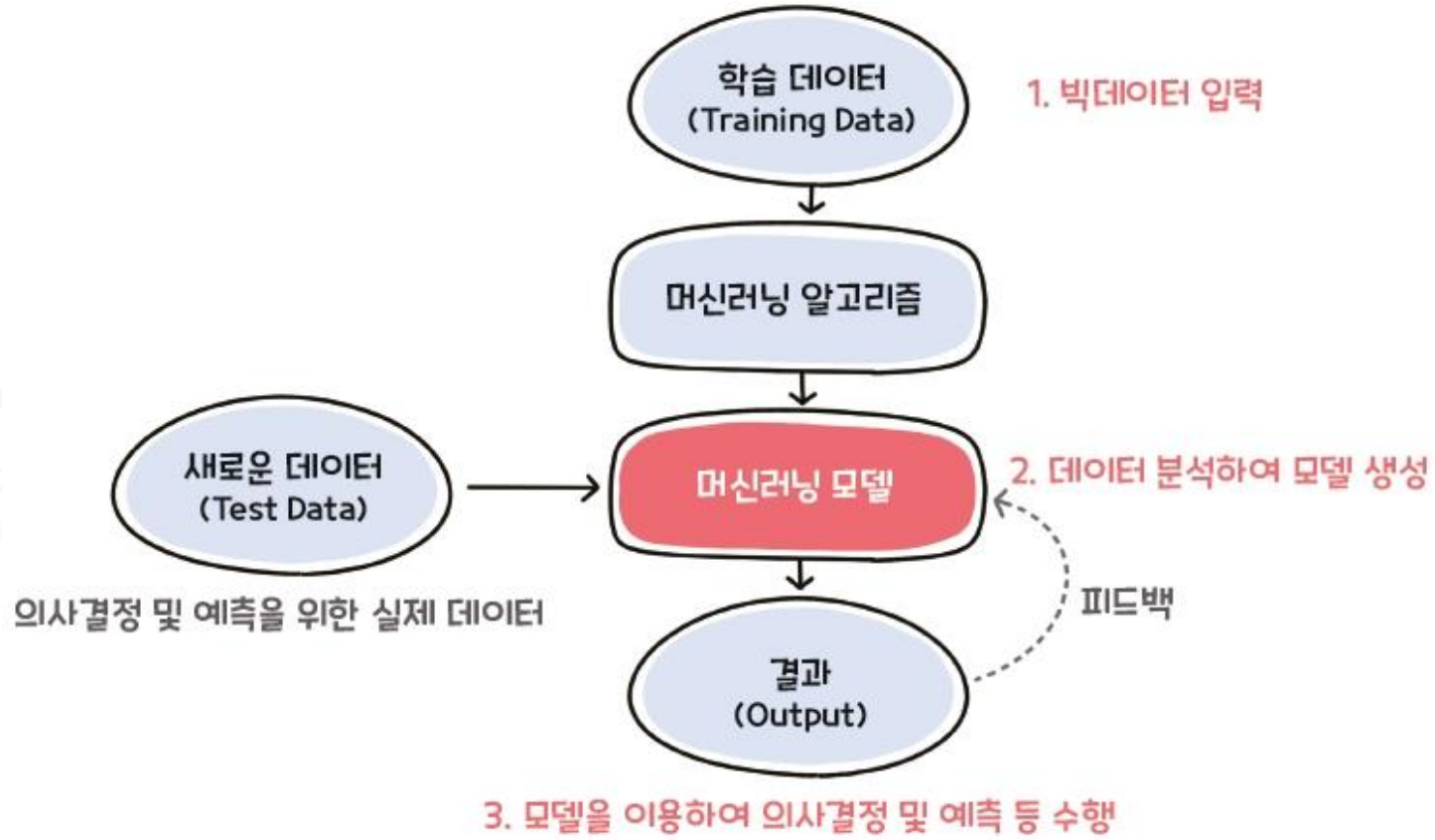
❖ Concepts of AI, ML, and DL

- Machine Learning
  - A technology that allows computers to learn like humans so that computers themselves can discover new rules without human help
  - Machine learning basically analyzes data using algorithms, learns through analysis, and makes judgments or predictions based on what is learned
  - Machine learning is the process of self-learning and processing data
    - Insert Big Data
    - Analyze data to create a model
    - Use models to make decisions, predictions, etc

Introduction

❖ Concepts of AI, ML, and DL

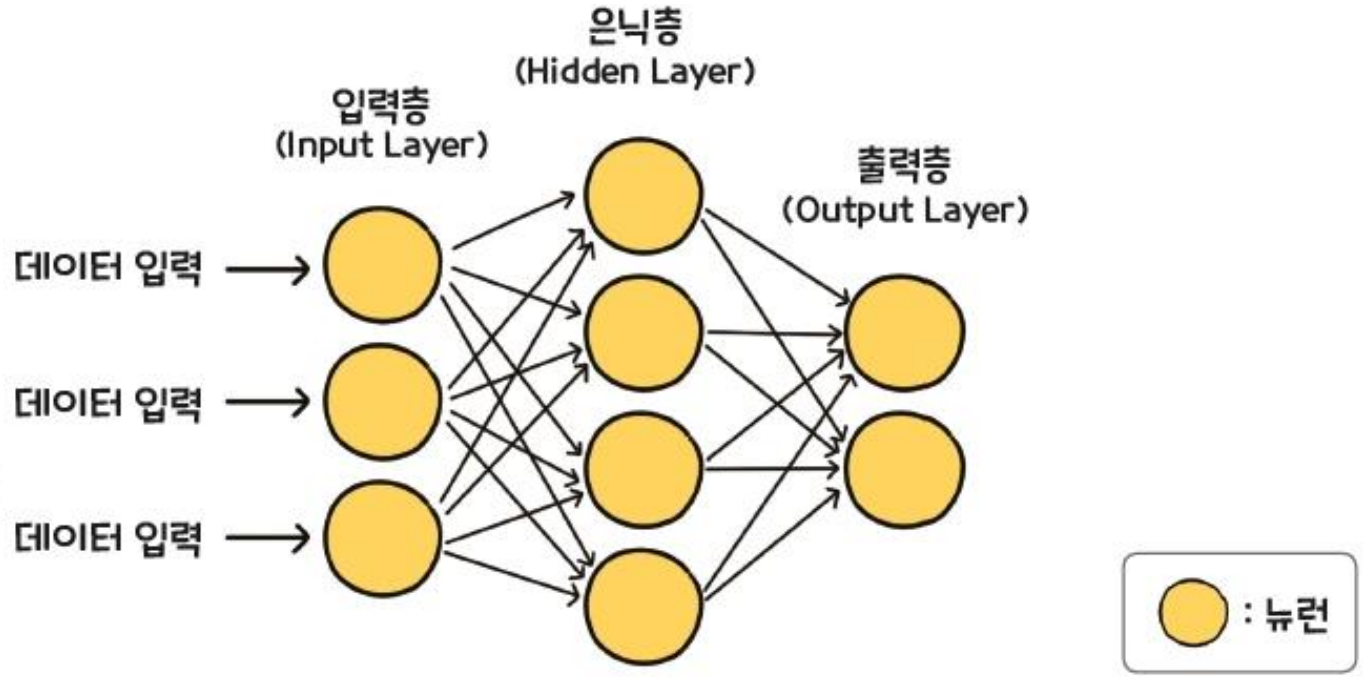
- Machine Learning



Introduction

❖ Concepts of AI, ML, and DL

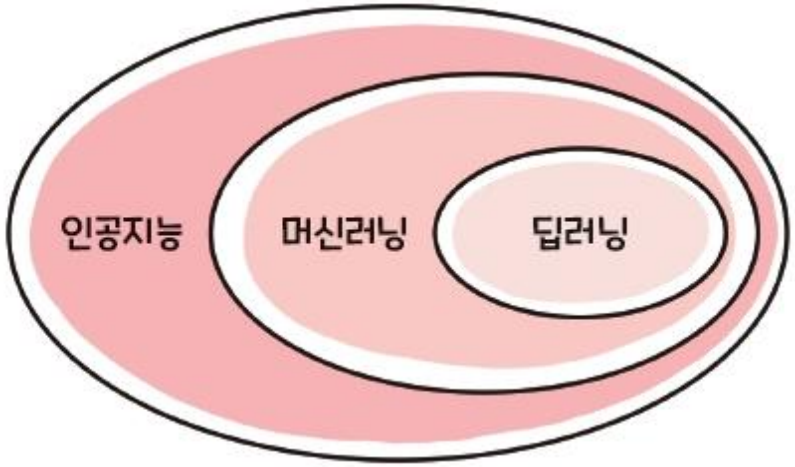
- Deep Learning
  - ANN, Artificial Neural Network
    - A network of interconnected neurons



Introduction

❖ Concepts of AI, ML, and DL

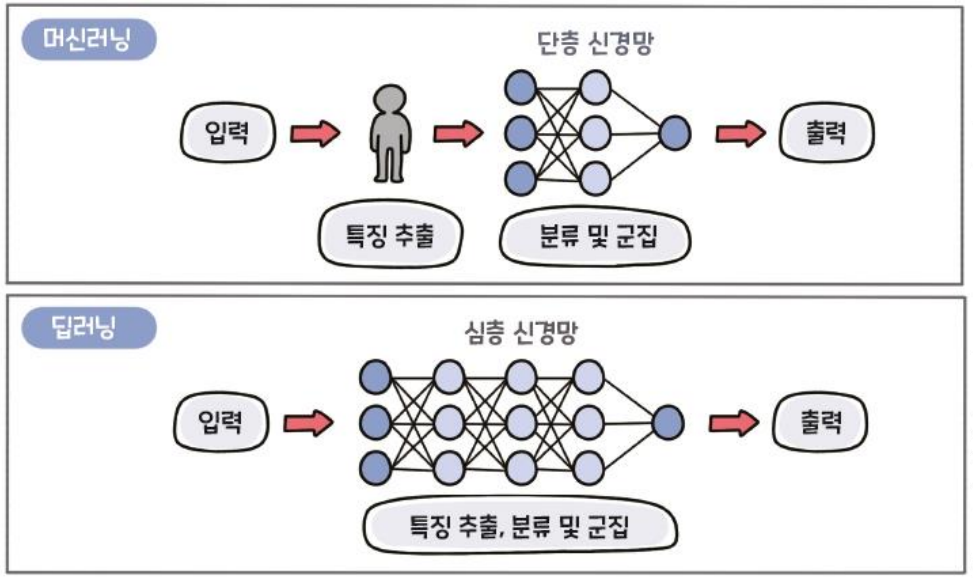
- Deep Learning
  - Technology for performing machine learning using artificial neural networks with multiple hidden layers
  - “Deep” in deep learning means deep layers of continuous neural networks
  - Performance increases as this neural network deepens



Introduction

❖ Difference between ML and DL

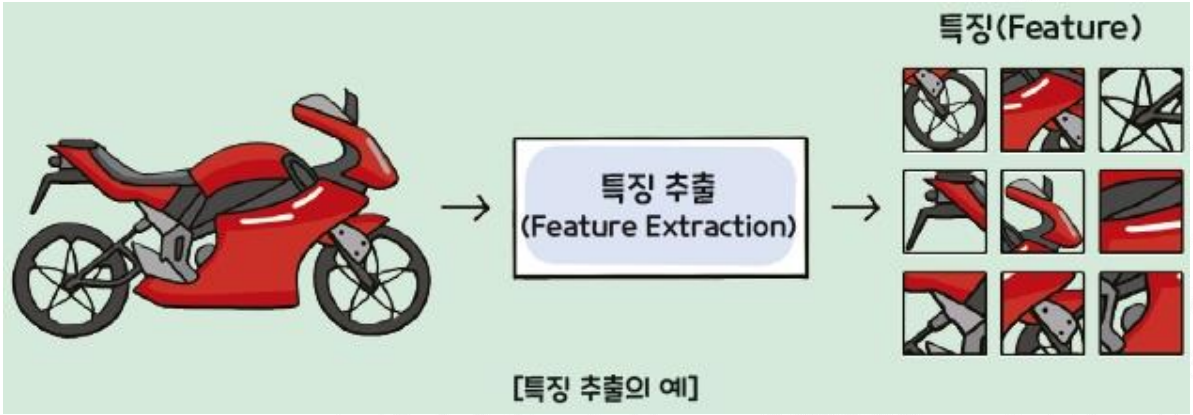
- Human in the loop
  - Machine learning involves some degree of intervention, such as human informing the learning data of labels (corrects) or extracting the characteristics of the data
  - Deep learning learns on its own without human intervention



Introduction

❖ Difference between ML and DL

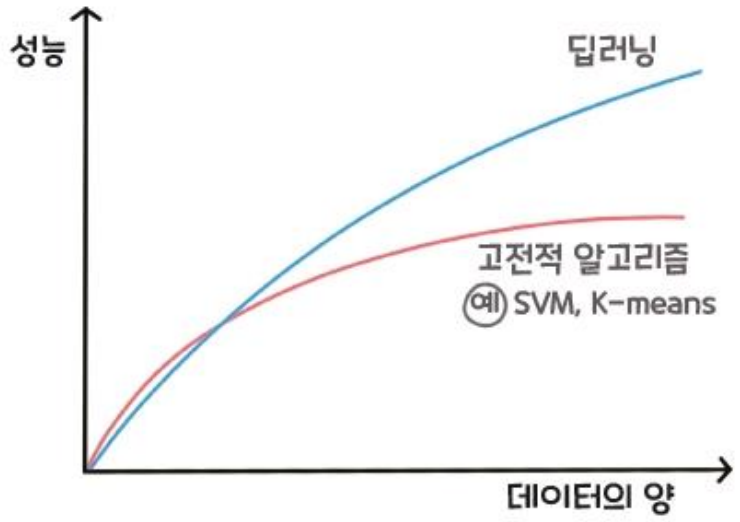
- Feature extraction
  - In machine learning, in order for a computer to learn on its own, it has to convert human-recognized data into computer-aware data
  - For this task, it finds out what characteristics each data has and converts the data into vector



Introduction

❖ Difference between ML and DL

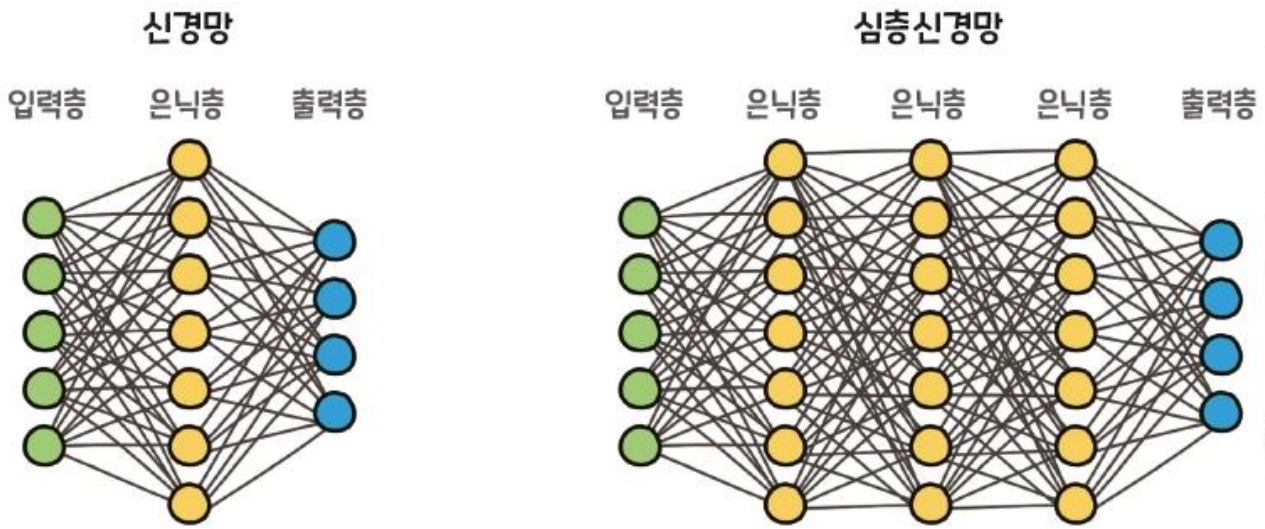
- Data dependencies
  - Deep learning directly extracts important features to solve a given problem
  - If you don't have enough data, you can't extract the exact features
  - On the other hand, if sufficient data is given, it performs well enough to identify important features that humans do not recognize



Introduction

❖ Difference between ML and DL

- Using neural network
  - Deep learning uses a deep neural network to extract features from input data and derive results (prediction or classification) on its own
  - The use of deep neural networks is a distinct characteristic of deep learning



Introduction

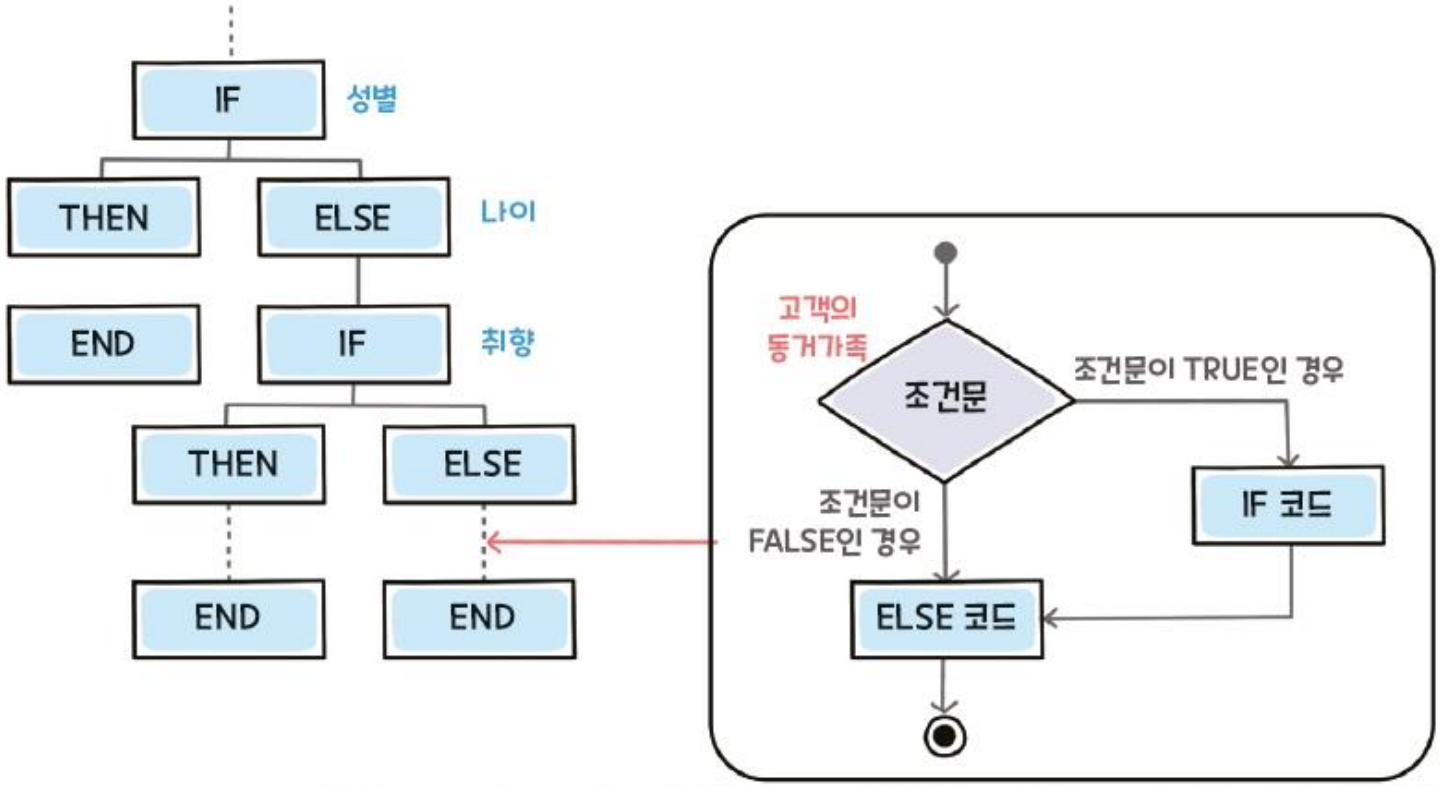
❖ Difference between ML and DL

- Using neural network

구분	머신러닝	딥러닝
필요한 데이터의 양	적은 양의 데이터도 가능	빅데이터
정확도	낮음	높음
훈련 시간	짧은 시간 안에 가능	오래 걸림
하드웨어	CPU만으로도 가능	GPU
하이퍼파라미터 튜닝	제한적	다양한 방법으로 튜닝 가능

Why Machine Learning?

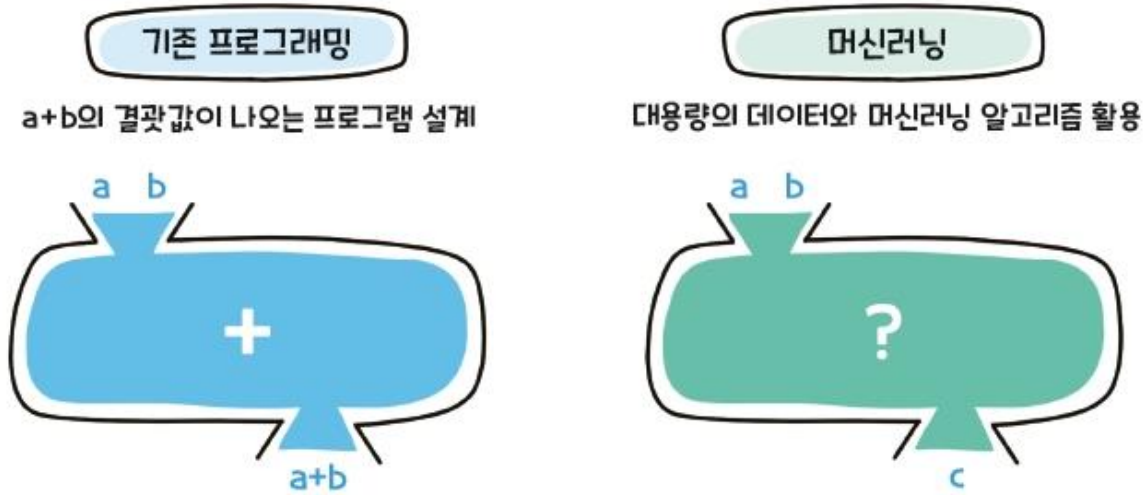
❖ Limitation of basic programming



Why Machine Learning?

❖ Usability of machine learning

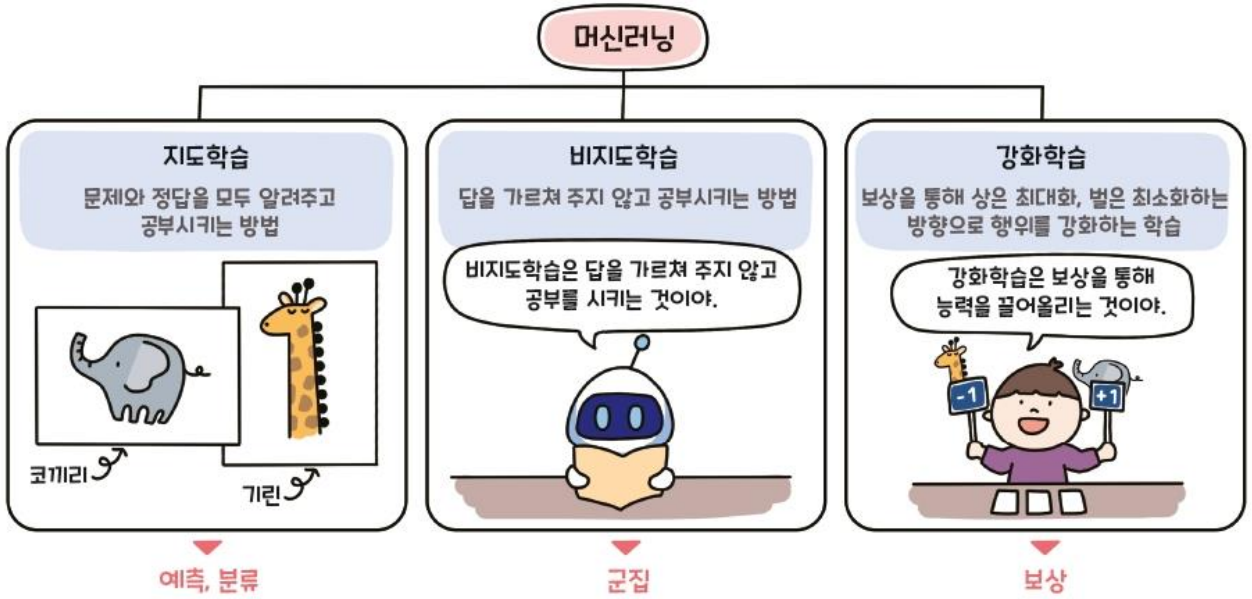
- But it's not the right time to make a quick decision
  - Using machine learning to solve this problem
- Machine learning is a very useful solution when large amounts of data and many variables are involved, and programs with conventional rules cannot solve complex tasks or problems



Kinds of Machin Learning

❖ Categorization for training

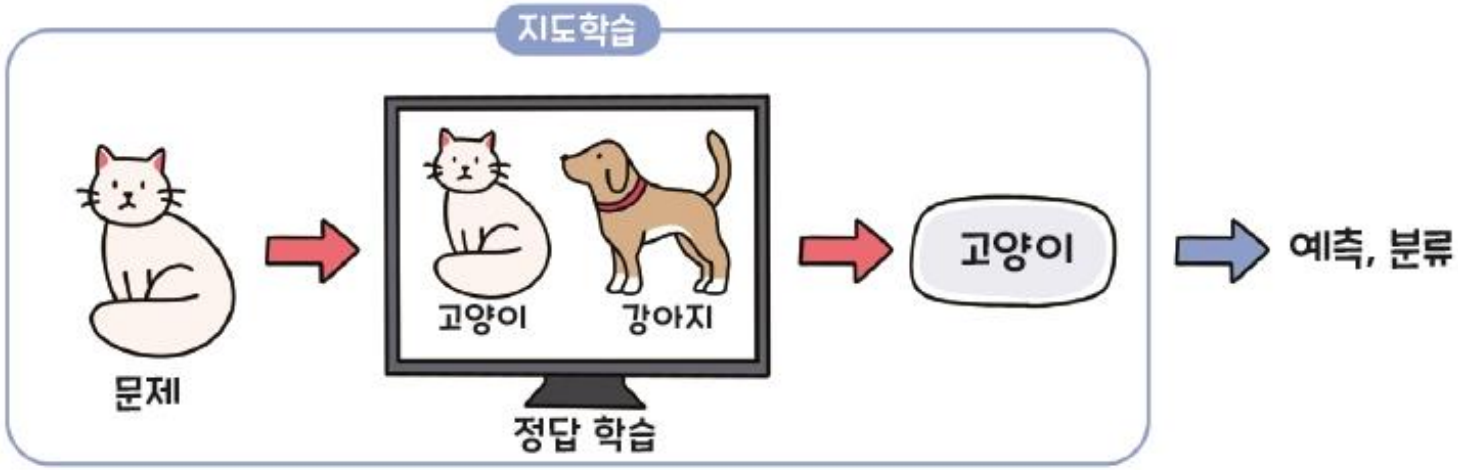
- Supervised learning : classification and regression
- Unsupervised learning : clustering
- Reinforcement learning : use rewards for actions taken in the environment to conduct learning



Kinds of Machine Learning

❖ Supervised learning

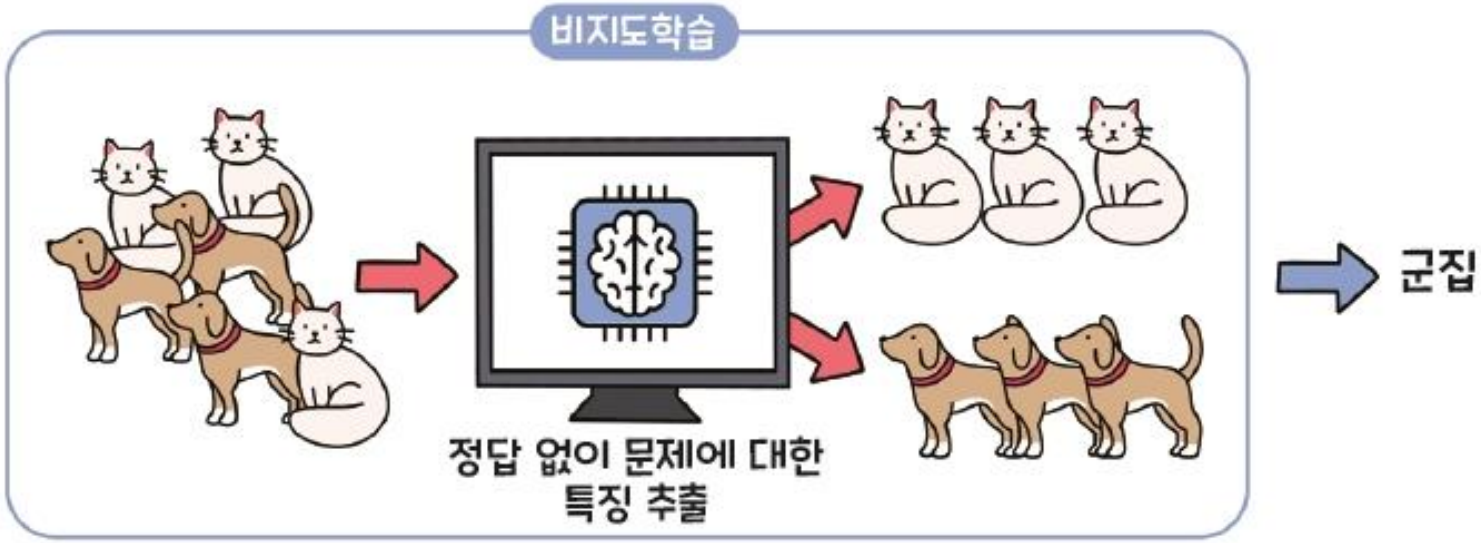
- Learning to predict the right answer to an unknown problem by learning questions and answers together
- The models used in supervised learning include prediction and classification



Kinds of Machine Learning

❖ Unsupervised learning

- A form of computer learning without the help
- Computer uses training data to find regularity between data



Kinds of Machine Learning

❖ Unsupervised learning

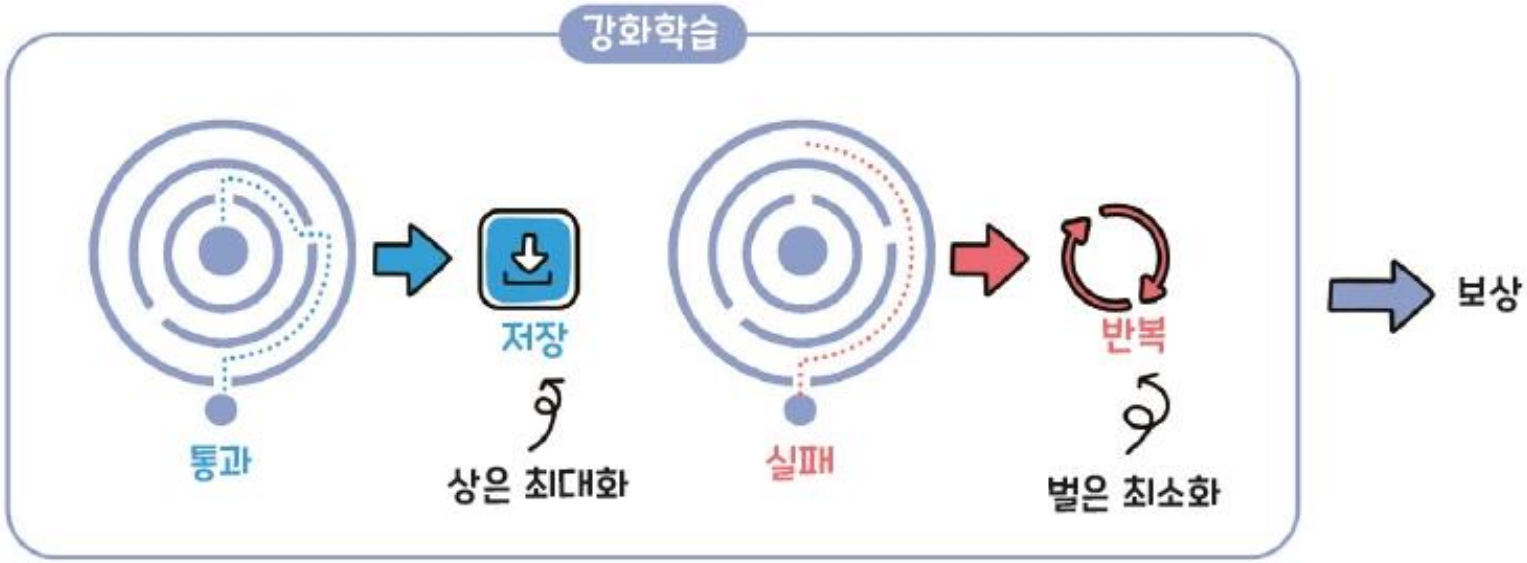
- Unlike supervised learning, which identified the relationship between x (input data) and y (labels in supervised learning),
- Unsupervised learning identifies the relationship between x by itself
- In other words, the difference between y (label)
  - Clustering is a model used in unsupervised learning

구분	지도학습	비지도학습
필요한 데이터 종류	x(학습 데이터), y(레이블)	x(학습 데이터)

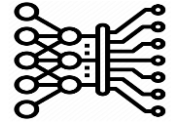
Kinds of Machine Learning

❖ Reinforcement learning

- Learning to be rewarded for what you've done
- How computers learn to choose the best behavior for a given state



Kinds of Machine Learning

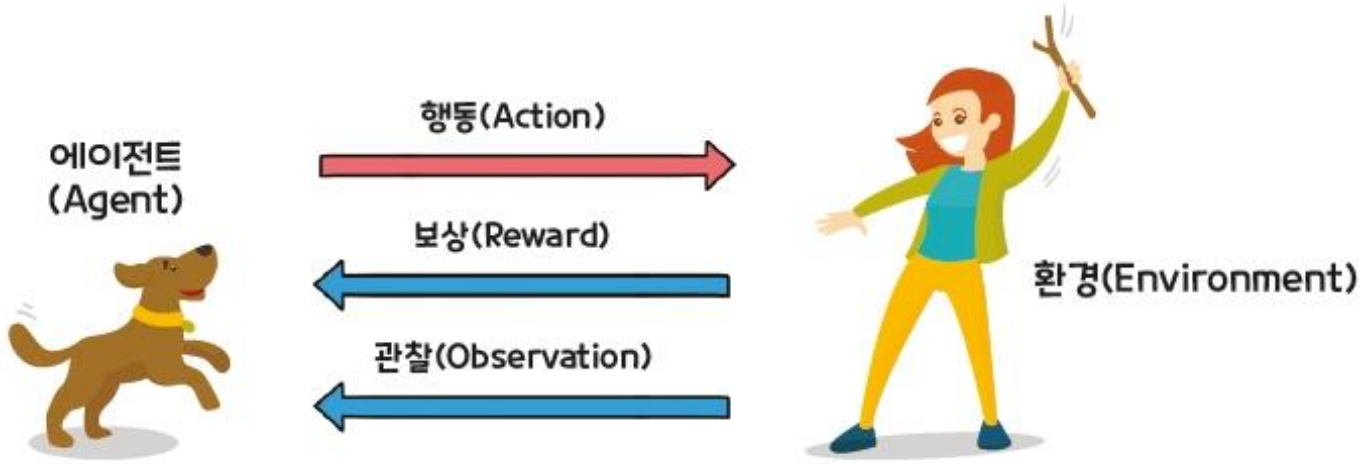


- Reinforcement learning
  - Agent: Subject to act in a given problem situation
  - State : Current situation
  - Action: Options that the player can take
  - Rewards: Benefits that follow when a player does something
  - Environment: means the problem itself
  - Observation : Information about the collected by the agent

Kinds of Machine Learning

❖ Reinforcement learning

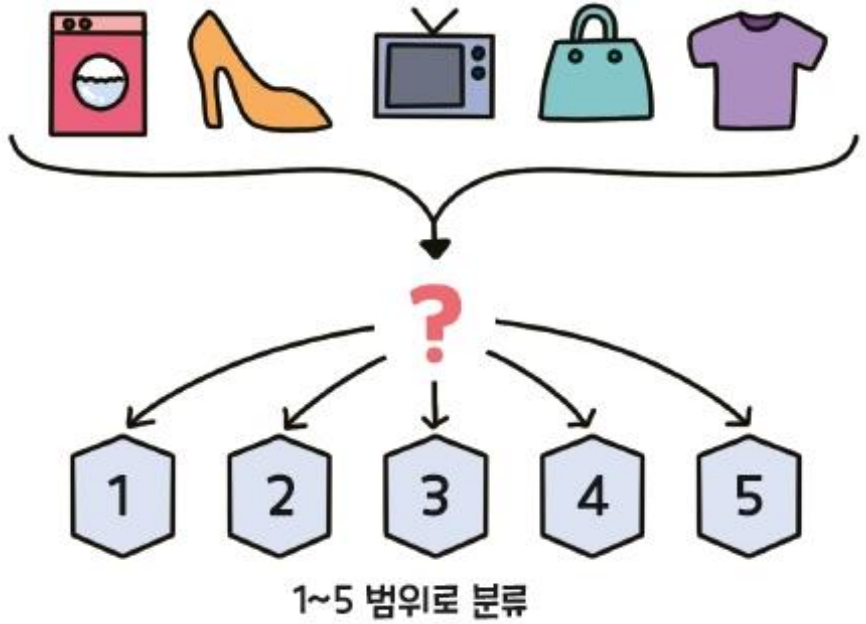
- Depending on the behavior chosen by the agent in a given environment, you are rewarded if the behavior is the right choice, and punished if the behavior is the wrong choice
- Reinforcement learning allows the agent to keep an eye on the status and learn (behavior) toward higher rewards



Types of Machine Learning Algorithm

❖ Classification

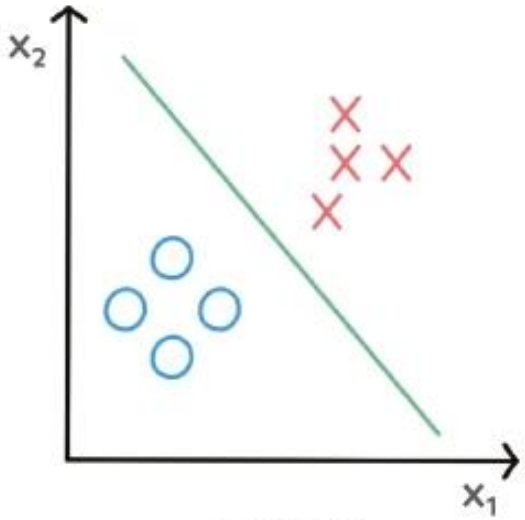
- A technique for learning labeled data, classifying data with similar properties, and finding out which group the newly entered data



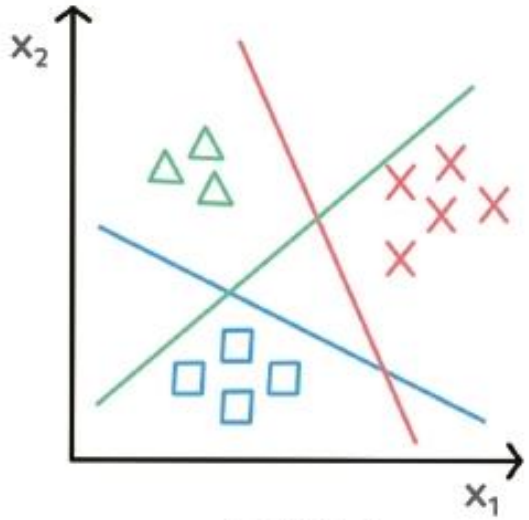
Types of Machine Learning Algorithm

❖ Classification

- Binary classification : categorize data into 2 groups
- Multiclass classification : categorize data into 3 or more groups



이진 분류



다중 분류

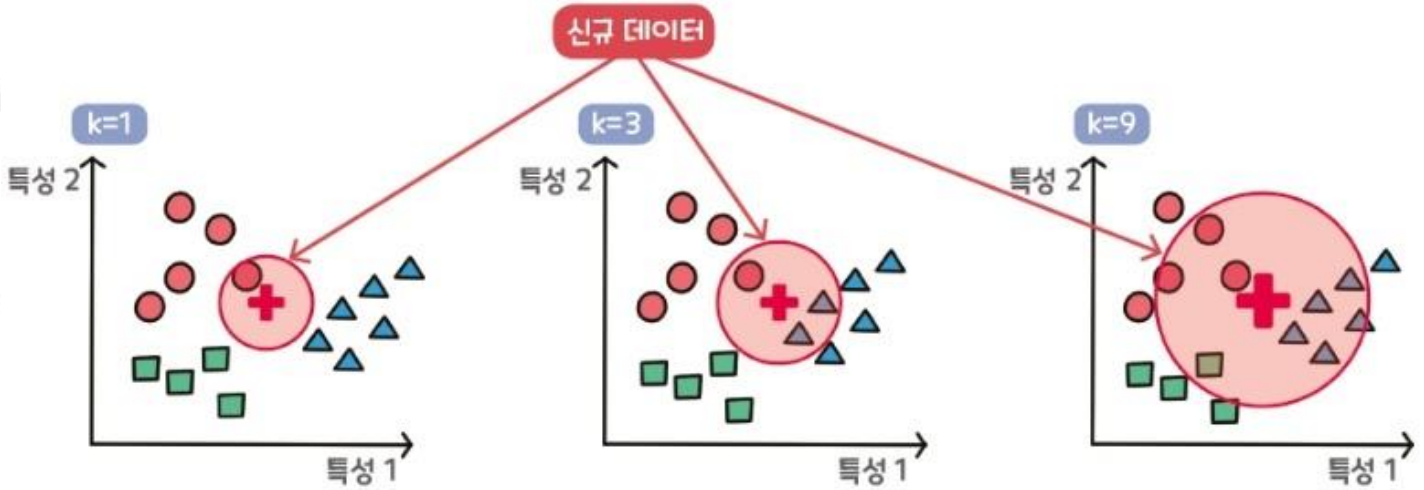
Types of Machine Learning Algorithm

- Classification (Algorithm)
  - K-neighbor nearest
  - Support vector machine
  - Decision tree

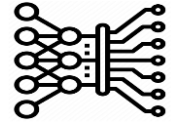
Types of Machine Learning Algorithm

❖ Classification (Algorithm)

- K-neighbor nearest
  - Algorithms to classify which of the existing groups of data (K groups) belongs to when new data comes in
  - (Example) When new data is entered when K=1, new data is classified as a red circle, when K=3, and when K=9, it is classified as a blue triangle



Types of Machine Learning Algorithm



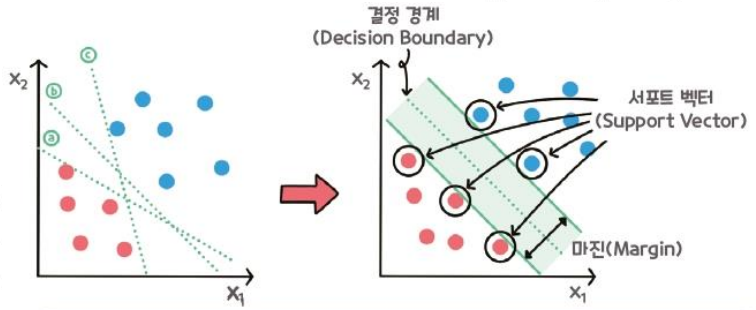
- Classification (Algorithm)
  - K-neighbor nearest
    - KNNs are not significantly affected by the noise present in the learning data and are quite effective when the number of learning data is large
    - However, it is unclear which hyperparameters are suitable for analysis, so there is a disadvantage that researchers should randomly select according to each characteristic of the data

Types of Machine Learning Algorithm

❖ Classification (Algorithm)

▪ Support vector machine

- Categorize data in the direction of maximizing margin, which means margin between two categories
- SVMs find and classify lines that maximize margins, so larger margins are more likely to be classified even if new data comes in
- SVM is easy to use and highly predictive
  - However, it takes time to build a model and the results are less descriptive

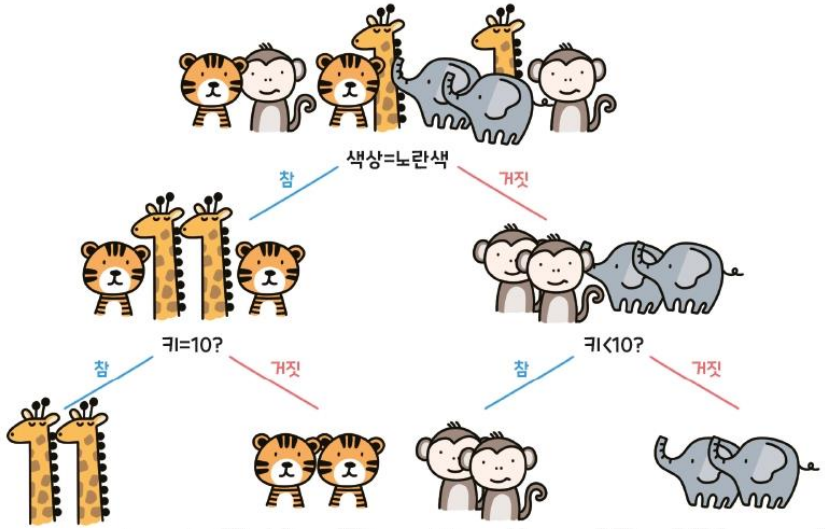


Types of Machine Learning Algorithm

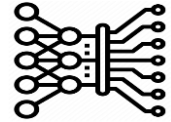
❖ Classification (Algorithm)

▪ Decision tree

- An analysis method for classifying decision-making rules into tree forms
- It is called 'decision tree' because the method of starting from the upper node and expanding to the lower node according to the classification criteria resembles 'tree'



Types of Machine Learning Algorithm

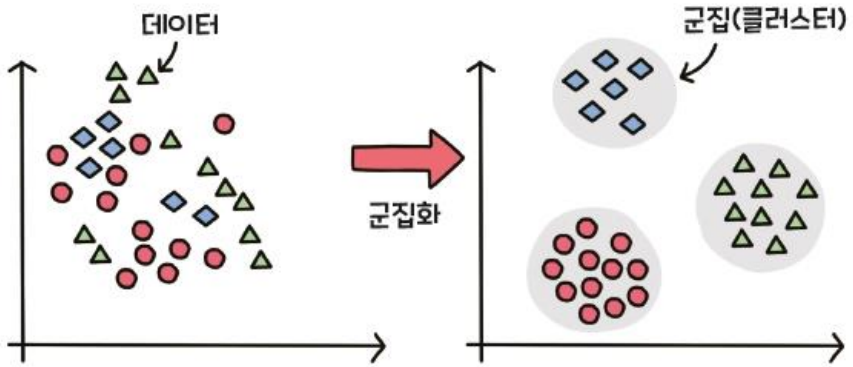


- Classification (Algorithm)
  - Decision tree
    - Decision Tree is intuitive and easy to understand the analysis process
    - In the case of artificial neural networks, it is a black box model that is difficult to explain the analysis results, while decision trees can observe the analysis process with their eyes
    - Need for a clear explanation of the results

Types of Machine Learning Algorithm

❖ Unsupervised learning

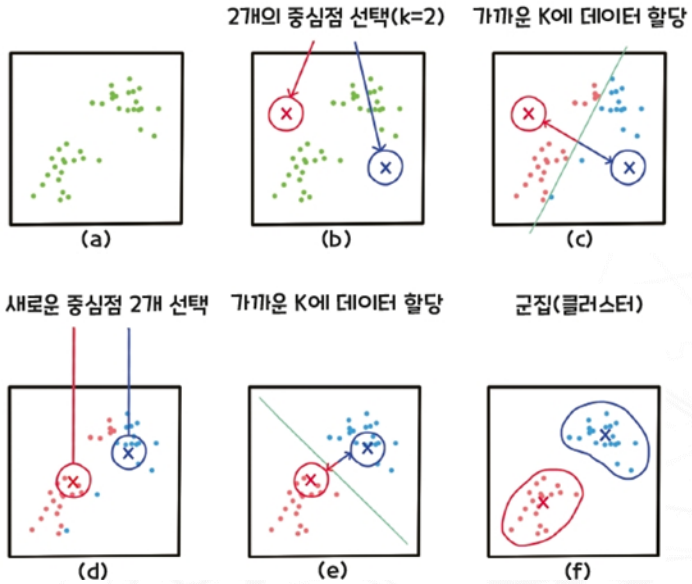
- Cluster
  - A group of data with similar characteristics
- Clustering
  - Classifying the data into clusters according to a similar degree when given the data
  - Various data are mixed together, but the clustering process groups similar data as shown in the graph on the right



Types of Machine Learning Algorithm

❖ Unsupervised learning

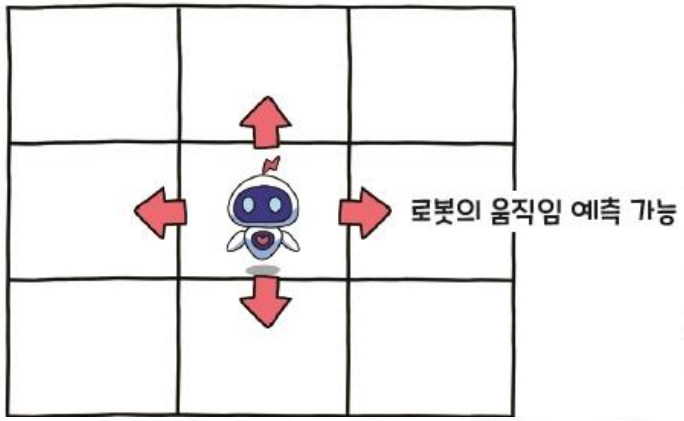
- K-means clustering
  - 'K' is the number of groups to be grouped from the given data
  - ‘Means’ means the average distance between the center of each cluster and the data
  - The center of the cluster is called centroids



Types of Machine Learning Algorithm

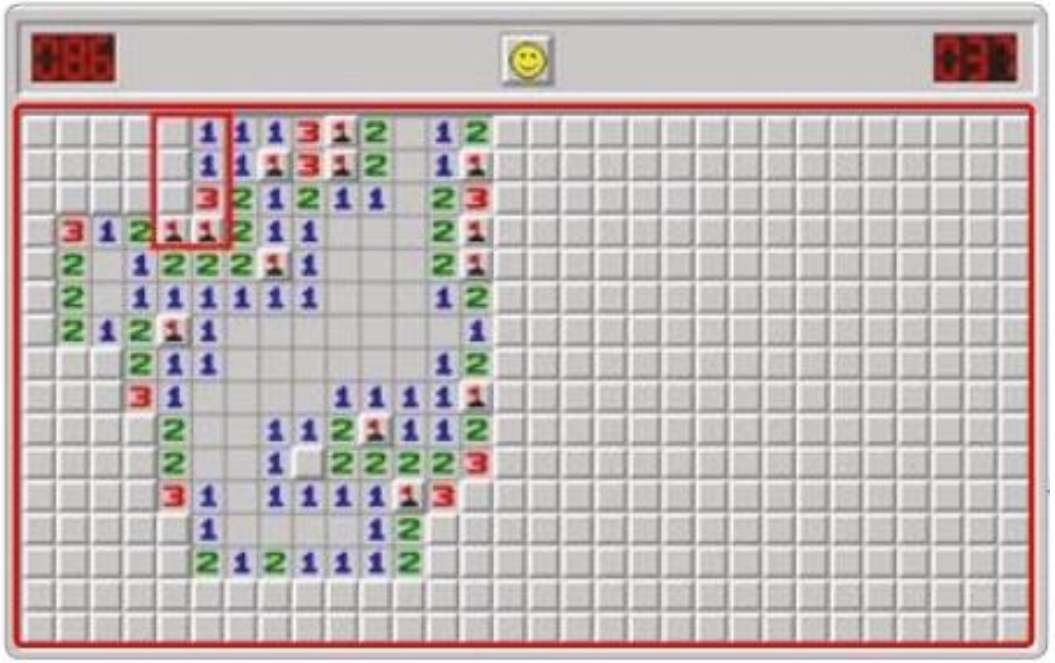
❖ Reinforcement learning

- Algorithm
  - Model-based algorithms refer to the probability that an action in the current state will result in the next state
  - Intuitive visibility of the robot's next state as it moves up, down, left, and right in a grid space
    - Model-based algorithms can predict changes in state according to behavior, resulting in optimal solutions

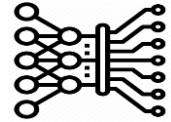


Types of Machine Learning Algorithm

- Reinforcement learning
  - Algorithm
    - Finding a policy that maximizes the rewards an agent receives through action



Examples

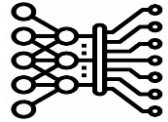


❖ Install the scikit-learn library

- pip install scikit-learn

TIP https://scikit-learn.org/dev/_downloads/scikit-learn-docs.pdf에 접속하면 가장 최신 버전의 사이킷 런 사용 설명서를 무료로 다운로드할 수 있다. 무려 2,500여 쪽에 달하는 방대한 문서다. 그렇다고 겁먹을 필요는 없다. 필요한 부분을 선택적으로 참조하면 된다.

Load ‘iris’ Dataset



❖ Load the dataset

프로그램 3-1(a) iris 데이터셋 읽기

```
01  from sklearn import datasets
02
03  d=datasets.load_iris()     # iris 데이터셋을 읽고
04  print(d.DESCR)             # 내용을 출력
```

- 01행: sklearn 모듈의 datasets 클래스를 불러옴
- 03행: load_iris 함수를 호출해 iris 데이터셋을 읽어 객체 d에 저장
- 04행: 객체 d의 DESCR 변수를 출력

❖ Terminology

- Dataset
- Feature vector
- Class

TIP 기계 학습이 사용하는 데이터는 여러 개의 샘플을 담고 있어서 데이터셋(data set)이라 부르기도 한다. 이 책에서는 데이터와 데이터셋을 엄밀히 구분하지 않고 함께 사용하는데, 데이터셋은 iris처럼 특정한 데이터를 가리킬 때 주로 사용한다.

Load ‘iris’ Dataset

```
Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

:Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988
...
```

150개의 샘플

네 개의 특징(feature)

세 개의 부류

Load ‘iris’ Dataset

❖ ‘iris’ dataset

프로그램 3-1(b) iris의 내용 살펴보기

```
05  for i in range(0,len(d.data)):        # 샘플을 순서대로 출력
06     print(i+1,d.data[i],d.target[i])
```

```
1 [5.1 3.5 1.4 0.2] 0
2 [4.9 3.  1.4 0.2] 0
3 [4.7 3.2 1.3 0.2] 0
4 [4.6 3.1 1.5 0.2] 0
...

51 [7.  3.2 4.7 1.4] 1
52 [6.4 3.2 4.5 1.5] 1
53 [6.9 3.1 4.9 1.5] 1
54 [5.5 2.3 4.  1.3] 1
...
101 [6.3 3.3 6.  2.5] 2
102 [5.8 2.7 5.1 1.9] 2
103 [7.1 3.  5.9 2.1] 2
104 [6.3 2.9 5.6 1.8] 2
...
```

d.target(레이블)

d.data(특징 벡터)

Representation of dataset

❖ Representing samples as feature vectors and labels

- Feature vectors are denoted by x

  특징 벡터: $\mathbf{x}=(x_1,x_2,\cdots,x_d)$

  - d is the number of features called the dimension of the feature vector
- Labels are 0,1,2,...A value of ,c-1 or 1,2,...A value of ,c-1,c or one hot code
  - One hot code is a binary sequence with only one element
  - Ex) Setosa: (1,0,0), Versicolor: (0,1,0), Virginica: (0,0,1)

	특징 벡터 $\mathbf{x}=(x_1, x_2,\cdots, x_d)$	레이블(참값) y
샘플 1:	(5.1, 3.5, 1.4, 0.2)	0
샘플 2:	(4.9, 3.0, 1.4, 0.2)	0
…	…	…
샘플 51:	(7.0, 3.2, 4.7, 1.4)	1
샘플 52:	(6.4, 3.2, 4.5, 1.5)	1
…	…	…
샘플 101:	(6.3, 3.3, 6.0, 2.5)	2
샘플 102:	(5.8, 2.7, 5.1, 1.9)	2
…	…	…
샘플 n:	(5.9, 3.0, 5.1, 1.8)	2

iris 데이터셋 (n=150, d=4)

Data Distribution of Feature Space

❖ iris dataset

- Distribution of data in a three-dimensional space, excluding one data dimension

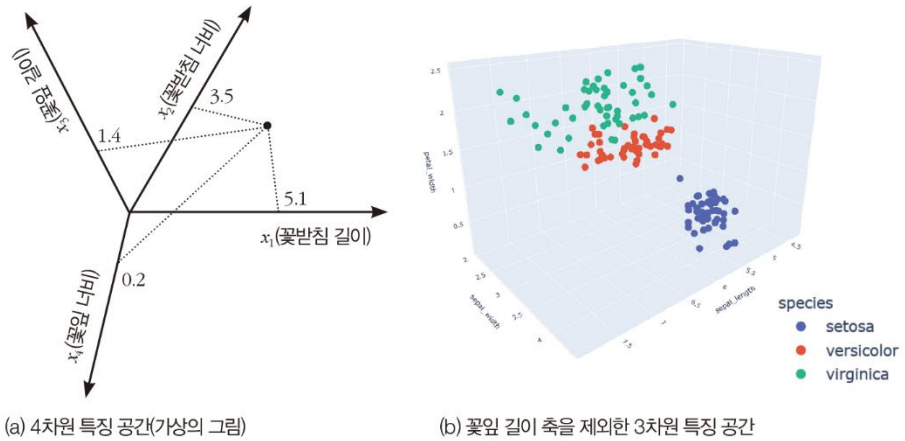
프로그램 3-2 iris 데이터의 분포를 특징 공간에 그리기

```
01  import plotly.express as px
02
03  df = px.data.iris()
04  fig = px.scatter_3d(df, x='sepal_length', y='sepal_width', z='petal_width',
    color='species')        # petal_length를 제외하여 3차원 공간 구성
05  fig.show(renderer="browser")
```

Data Distribution of Feature Space

❖ Observe the distribution of data in the feature space

- Setosa is distributed downward and Virginica is distributed upward for the vertical width
  - Petal width is excellent in discernment
- The segmental width axis overlaps a lot in three categories, so it is less sensible
- As a whole, the three classes occupy different areas of the three-dimensional space, with several samples overlapping



(a) 4차원 특징 공간(가상의 그림)

(b) 꽃잎 길이 축을 제외한 3차원 특징 공간

Data Distribution of Feature Space

NOTE 다차원 특징 공간

종이에 그릴 수 있는 공간은 3차원으로 제한되지만, 수학은 아주 높은 차원까지 다룰 수 있다. 예를 들어 2차원 상의 두 점 $\mathbf{x}=(x_1,x_2)$와 $\mathbf{y}=(y_1,y_2)$의 거리를 $d(\mathbf{x},\mathbf{y})=\sqrt{(x_1-y_1)^2+(x_2-y_2)^2}$으로 계산할 수 있는데, 4차원 상의 두 점 $\mathbf{x}=(x_1,x_2,x_3,x_4)$와 $\mathbf{y}=(y_1,y_2,y_3,y_4)$의 거리는 $d(\mathbf{x},\mathbf{y})=\sqrt{(x_1-y_1)^2+(x_2-y_2)^2+(x_3-y_3)^2+(x_4-y_4)^2}$로 계산할 수 있다.

일반적으로 d차원 상의 두 점의 거리는 $d(\mathbf{x},\mathbf{y})=\sqrt{\sum_{i=1}^{d}(x_i-y_i)^2}$로 계산한다. 기계 학습에서는 d=수백~수만에 달하는 매우 고차원 특징 공간의 데이터를 주로 다룬다.

Modeling and Prediction

❖ Using the support vector machine model

프로그램 3-1(c) iris에 기계 학습 적용: 모델링과 예측

```
07  from sklearn import svm
08
09  s=svm.SVC(gamma=0.1,C=10)                    # svm 분류 모델 SVC 객체 생성하고
10  s.fit(d.data,d.target)                       # iris 데이터로 학습
11
12  new_d=[[6.4,3.2,6.0,2.5],[7.1,3.1,4.7,1.35]]  # 101번째와 51번째 샘플을 변형하여
                                                   새로운 데이터 생성
13  res=s.predict(new_d)
14  print("새로운 2개 샘플의 부류는", res)
```

Hyperparameter

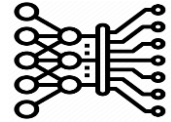
Training set

Test set

```
새로운 2개 샘플의 부류는 [2 1]
```

- 09행: SVM의 분류기 모델 SVC 클래스의 객체를 생성하여 s에 저장
- 10행: 객체 s의 fit 함수는 훈련 집합을 가지고 학습을 수행
  (매개변수로 특징 벡터 iris.data와 레이블 iris,target을 설정)
- 13행: 객체 s의 predict 함수는 테스트 집합을 가지고 예측 수행

Performance Measurement



- The importance of objective performance measurements
  - Important when choosing a model
  - Important when deciding whether to install on-site
- Generalization capabilities
  - Performance on new data not used for learning
  - The most obvious way is to install it on-site and measure performance
  - Cost makes it difficult to apply it in real life
  - Requires wisdom to segment and use given data

Performance Measurement

❖ Confusion matrix

- Matrix recording the number of correct and incorrect classifications by class
  - n_{ij}는 모델이 i라고 예측했는데 실제 부류는 j인 샘플의 개수

		참값(그라운드 트루스)					
		부류 1	부류 2	…	부류 j	…	부류 c
예측한 부류	부류 1	n_{11}	n_{12}		n_{1j}		n_{1c}
	부류 2	n_{21}	n_{22}		n_{2j}		n_{2c}
	…						
	부류 i	n_{i1}	n_{i2}		n_{ij}		n_{ic}
	…						
	부류 c	n_{c1}	n_{c2}		n_{cj}		n_{cc}

(a) 부류가 c개인 경우

		그라운드 트루스	
		긍정	부정
예측값	긍정	TP	FP
	부정	FN	TN

(b) 부류가 2개인 경우

- Positive and negative negative in binary classification
- True positive, false negative, false positive, true negative

Performance Measurement

❖ Performance metric

- Accuracy

$$\text{정확률} = \frac{\text{맞힌 샘플 수}}{\text{전체 샘플 수}} = \frac{\text{대각선 샘플 수}}{\text{전체 샘플 수}}$$

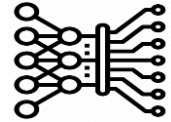
- Specificity and sensitivity

$$\text{특이도} = \frac{TN}{TN+FP}, \quad \text{민감도} = \frac{TP}{TP+FN}$$

- Precision and recall

$$\text{정밀도} = \frac{TP}{TP+FP}, \quad \text{재현율} = \frac{TP}{TP+FN}$$

Divide into Training/Validation/Test



❖ Training/Validation/Test

- Training set
  - Data used to learn machine learning models that provide both feature vector and label information
- Test Set
  - Data used to measure the performance of a learned model, which provides only feature vector information when predicting, and uses label information when measuring accuracy with prediction results

NOTE 하이퍼 매개변수 설정

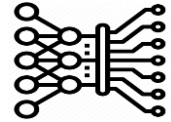
하이퍼 매개변수hyper parameter란 모델의 동작을 제어하는 데 쓰는 변수이다. 모델의 학습을 시작하기 전에 설정해야 하는데, 적절한 값으로 설정해야 좋은 성능을 얻을 수 있다. 최적의 하이퍼 매개변수 값을 자동으로 설정하는 일을 하이퍼 매개변수 최적화(hyper parameter optimization)라 하는데, 이것은 기계 학습의 중요한 주제 중 하나다. 하이퍼 매개변수 최적화는 4.10절에서 다룬다.

Divide into Training/Validation/Test

❖ Divide the given data into training, validation, and test sets at an appropriate rate

- Model selection included: divided into training/validation/test sets
- Exclude model selection: split into training/test sets

Divide into Training/Validation/Test



❖ Exclude the model selection

- 08행: train_test_split 함수로 훈련 60%, 테스트 40%로 랜덤 분할
- 12행: 훈련 집합 x_train, y_train을 fit 함수에 주어 학습 수행
- 14행: 테스트 집합의 특징 벡터 x_test를 predict 함수에 주어 예측 수행
- 17~20행: 테스트 집합의 레이블 y_test를 가지고 혼동 행렬 계산

프로그램 3-5 필기 숫자 인식 – 훈련 집합으로 학습하고 테스트 집합으로 성능 측정

```
01  from sklearn import datasets
02  from sklearn import svm
03  from sklearn.model_selection import train_test_split
04  import numpy as np
05
06  # 데이터셋을 읽고 훈련 집합과 테스트 집합으로 분할
07  digit=datasets.load_digits()
08  x_train,x_test,y_train,y_test=train_test_split(digit.data,digit.target,train_size=0.6)
09
```

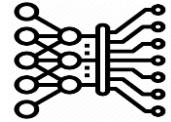
Divide into Training/Validation/Test

```
10  # svm의 분류 모델 SVC를 학습
11  s=svm.SVC(gamma=0.001)
12  s.fit(x_train,y_train)
13
14  res=s.predict(x_test)
15
16  # 혼동 행렬 구함
17  conf=np.zeros((10,10))
18  for i in range(len(res)):
19      conf[res[i]][y_test[i]]+=1
20  print(conf)
21
22  # 정확률 측정하고 출력
23  no_correct=0
24  for i in range(10):
25      no_correct+=conf[i][i]
26  accuracy=no_correct/len(res)
27  print("테스트 집합에 대한 정확률은", accuracy*100, "%입니다.")
```

예) 부류 3에 속하는 75개 샘플 중 73개를 3, 1개를 2, 1개를 7로 인식

```
[[76.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0. 78.  0.  0.  0.  0.  0.  0.  3.  0.]
 [ 0.  0. 66.  1.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0. 73.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0. 63.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0. 70.  0.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.  0. 77.  0.  0.  0.]
 [ 0.  0.  0.  1.  0.  0.  0. 77.  0.  1.]
 [ 0.  0.  0.  0.  0.  0.  0.  0. 74.  0.]
 [ 0.  0.  0.  0.  0.  1.  0.  0.  0. 56.]]

테스트 집합에 대한 정확률은 98.74826147426981%입니다.
```

Cross-Validation



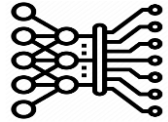
- Limitations of training/test set division
  - Likelihood of accidental high or accidental low accuracy
- k-fold cross validation
  - Use the training set divided into k subsets
  - Measure performance by learning with k-1 leaving one and then leaving it
  - Increase reliability by averaging k performance

Cross-Validation

(a) 모델 선택 포함

(b) 모델 선택 제외

Cross-Validation



프로그램 3-6 필기 숫자 인식 – 교차 검증으로 성능 측정

```
01  from sklearn import datasets
02  from sklearn import svm
03  from sklearn.model_selection import cross_val_score
04  import numpy as np
05
06  digit=datasets.load_digits()
07  s=svm.SVC(gamma=0.001)
08  accuracies=cross_val_score(s,digit.data,digit.target,cv=5) # 5-겹 교차 검증
09
10  print(accuracies)
11  print("정확률(평균)=%0.3f, 표준편차=%0.3f"%(accuracies.mean()*100,accuracies.std()))
```

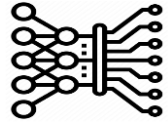
```
[0.97527473 0.95027624 0.98328691 0.99159664 0.95774648]
정확률(평균)=97.164, 표준편차=0.015
```

Challenging Issues for ML

❖ Need for a variety of data

- Computers can only recognize numbers 0 and 1 unlike humans
- In other words, the object in the image should be expressed as a number, and a separate processing (weight of machine learning) should be performed to recognize only the car among the objects
  - Through this complex process, a lot of data is needed to accurately recognize various types of cars

Challenging Issues for ML

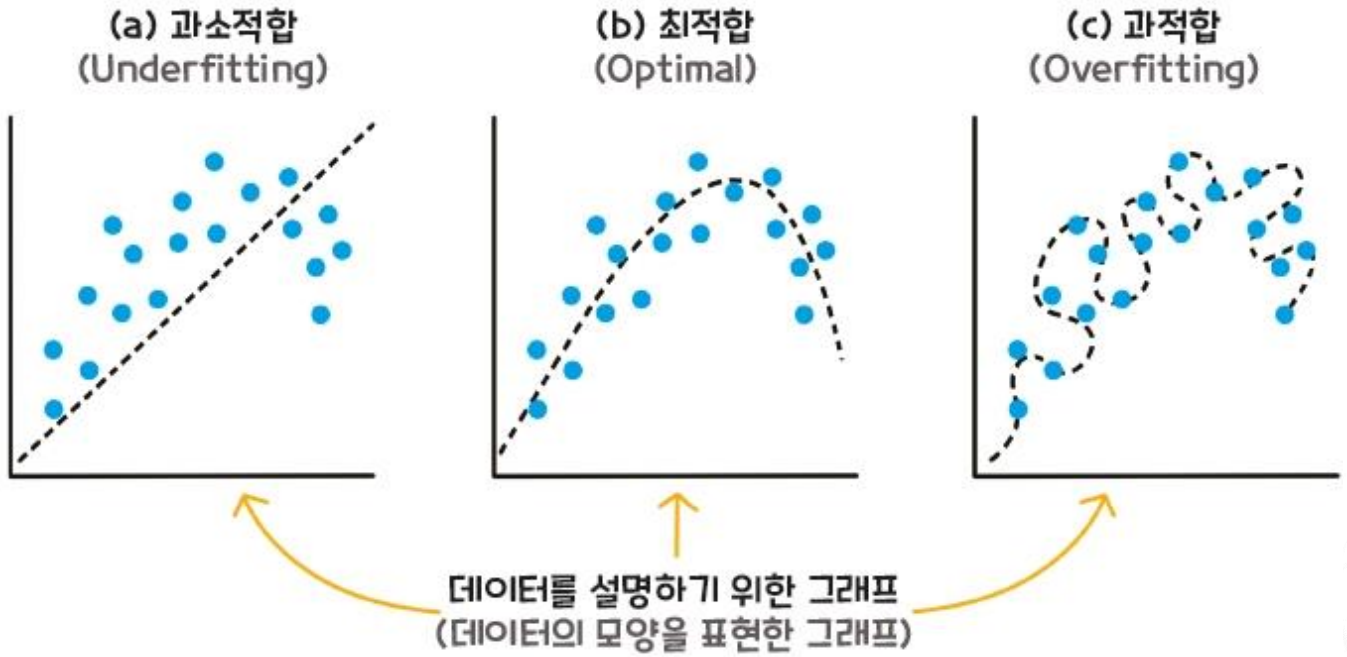


❖ Overfitting

- Overlearning training data means poor performance when analyzing real data
- This occurs when data is significantly lacking compared to the complexity of the problem
  - i.e., Learning data does not cover the entire space where the problem is defined and focuses only in some cases

Challenging Issues for ML

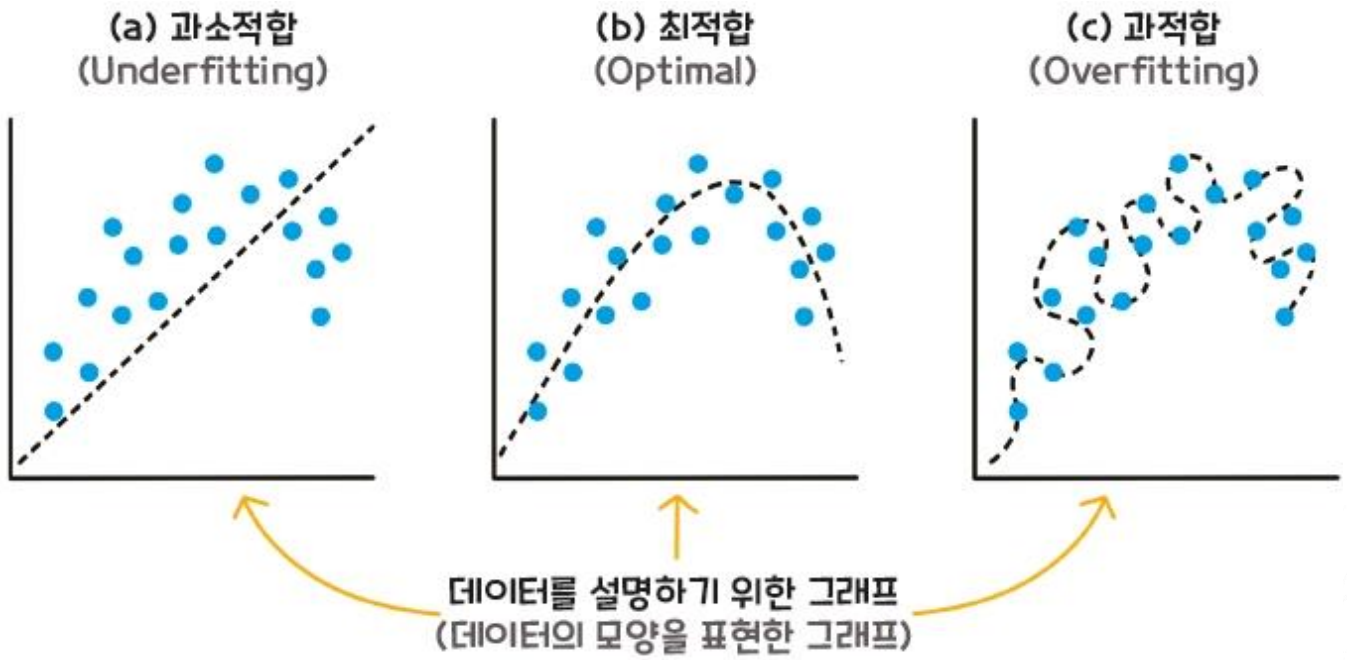
❖ Overfitting



- (a) 그래프 : 변수가 0인 쪽의 데이터 몇 개는 비교적 잘 근사하지만, 일정 시점 이후 데이터는 우하향 하고 있음
- (b) 그래프 : 데이터와 비슷하게 우상향하고 있어 제대로 반영하고 있다고 볼 수 있음

Challenging Issues for ML

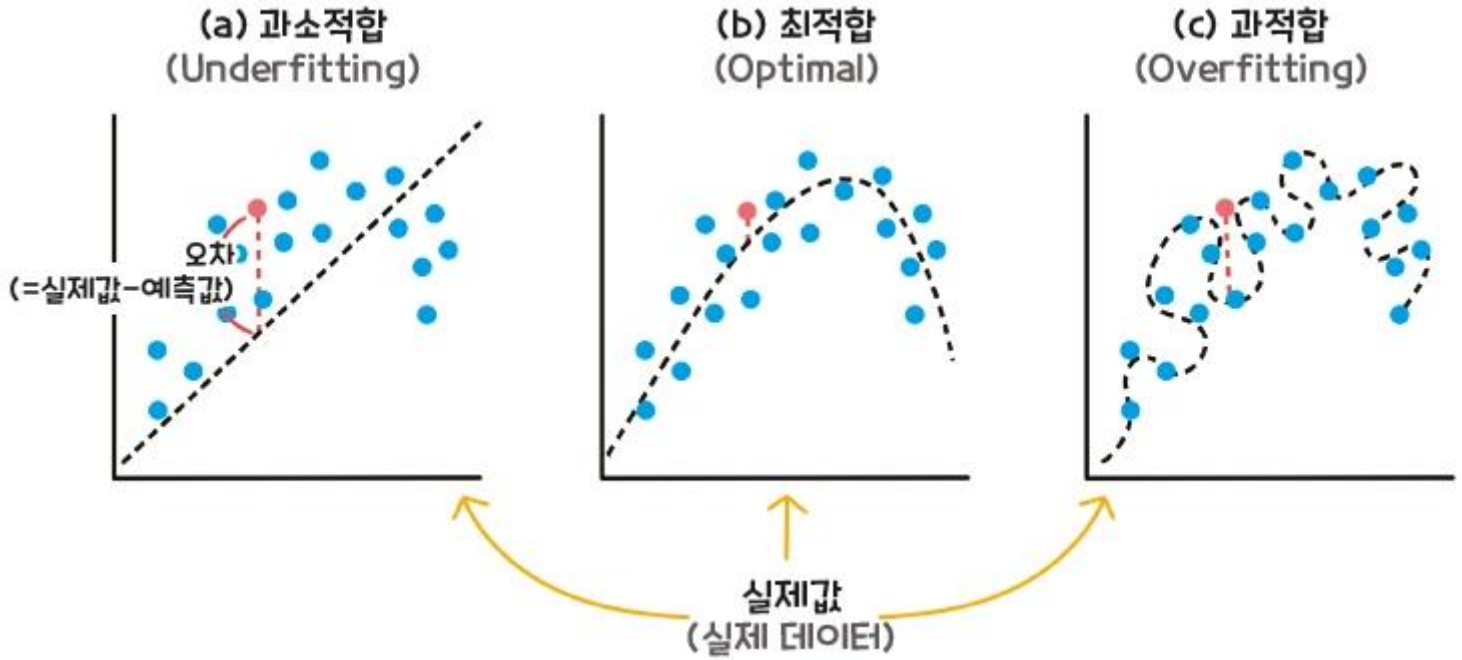
❖ Overfitting



- (c) 그래프 : 학습 데이터와 생성된 모델의 오차를 구해보면 0에 가까울 것임. 즉, 그래프가 모든 점을 지나고 있음
- 그렇다면 (b) 그래프보다 (c) 그래프가 더 좋은 그래프일까? 어떤 그래프가 좋은 그래프인지 알아보기 위해서는 실제 데이터값을 불러오면 됨

Challenging Issues for ML

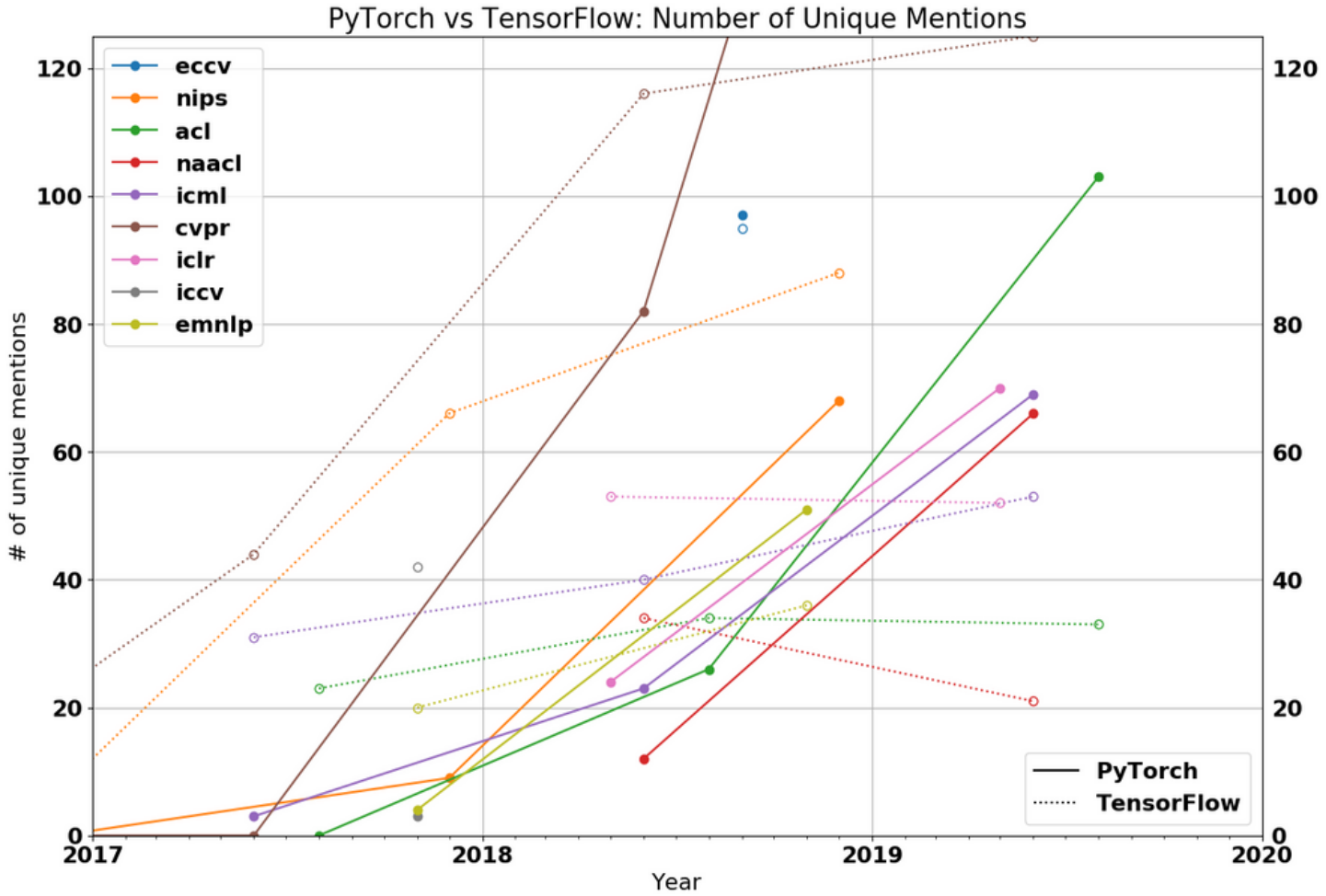
❖ Overfitting



- 하나의 실제값을 불러왔을 때, 실제값과 모델이 내놓은 예측값의 차이인 오차가 가장 작은 그래프가 좋은 그래프라고 할 수 있음
- 확인해 보면 (b) 그래프의 오차가 가장 작으므로 최적합(Optimal), 즉 가장 좋은 그래프라고 할 수 있음

Comparison of Deep Learning Libraries

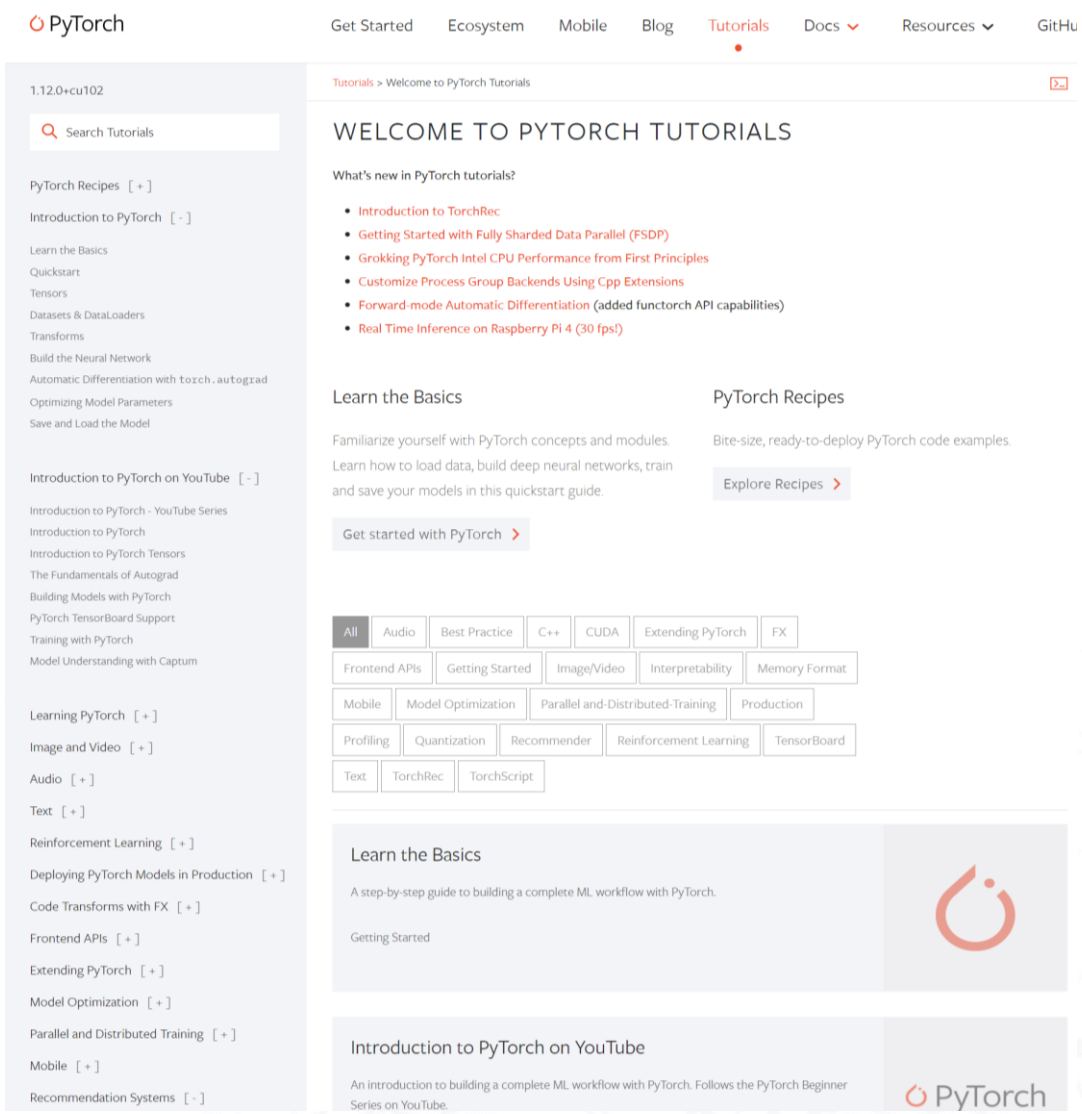
PyTorch vs. Tensorflow



PyTorch is a Python based scientific computing package

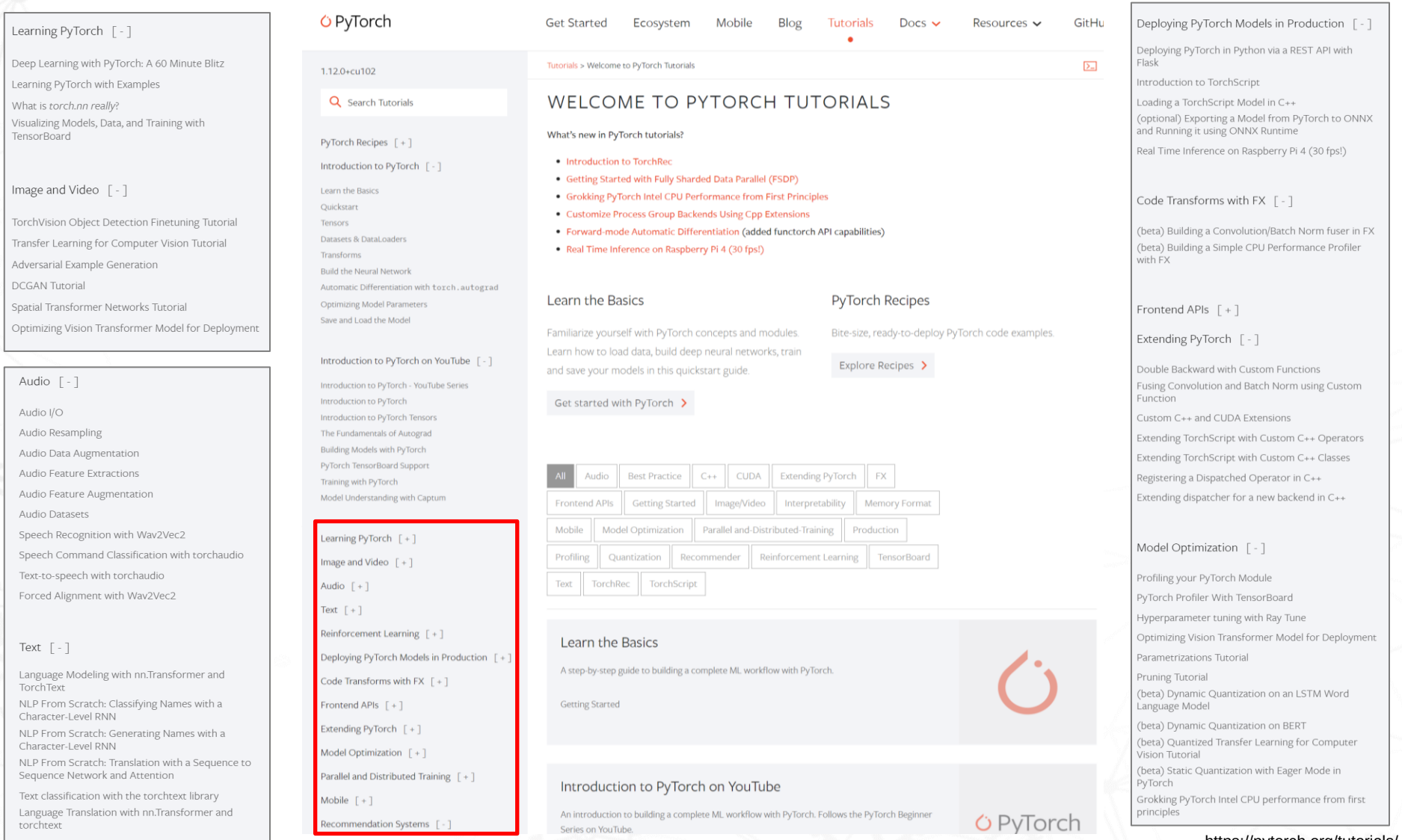
- ✓ A cousin of LUA-based Torch framework
- ✓ Not a simple set of wrappers to support popular language

PyTorch Tutorials (1/3)



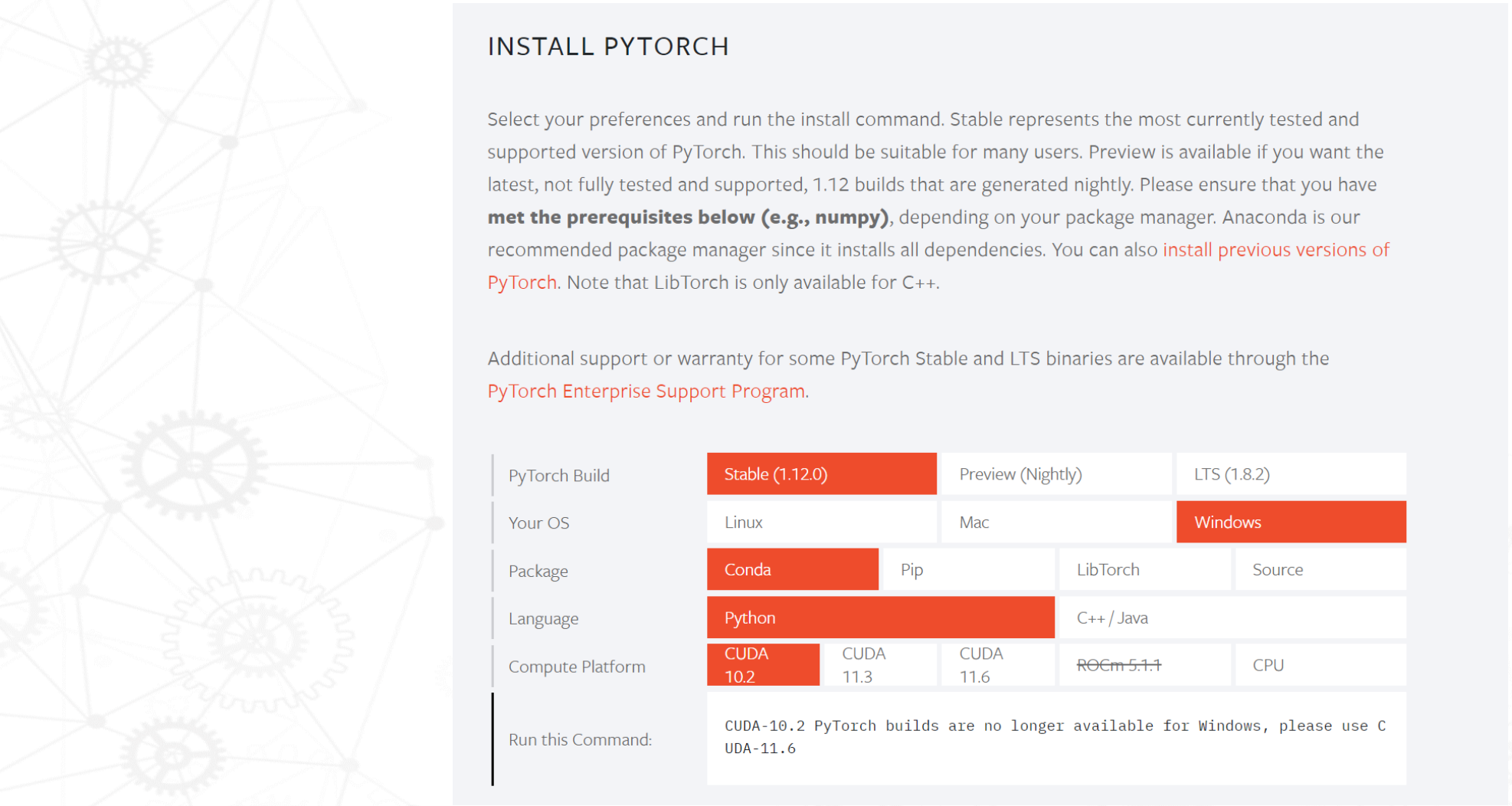
https://pytorch.org/tutorials/

PyTorch Tutorials (2/3)



https://pytorch.org/tutorials/

PyTorch Tutorials (3/3)



https://pytorch.org

PyTorch Examples (1/3)

PyTorch

Various PyTorch examples can use to learn and experiment

PyTorch Examples

Run Examples passing

https://pytorch.org/examples/

`pytorch/examples` is a repository showcasing examples of using PyTorch. The goal is to have curated, short, few/no dependencies *high quality* examples that are substantially different from each other that can be emulated in your existing work.

- For tutorials: https://github.com/pytorch/tutorials
- For changes to pytorch.org: https://github.com/pytorch/pytorch.github.io
- For a general model hub: https://pytorch.org/hub/ or https://huggingface.co/models
- For recipes on how to run PyTorch in production: https://github.com/facebookresearch/recipes
- For general Q&A and support: https://discuss.pytorch.org/

Available models

- Image classification (MNIST) using Convnets
- Word-level Language Modeling using RNN and Transformer
- Training Imagenet Classifiers with Popular Networks
- Generative Adversarial Networks (DCGAN)
- Variational Auto-Encoders
- Superresolution using an efficient sub-pixel convolutional neural network
- Hogwild training of shared ConvNets across multiple processes on MNIST
- Training a CartPole to balance in OpenAI Gym with actor-critic
- Natural Language Inference (SNLI) with GloVe vectors, LSTMs, and torchtext
- Time sequence prediction - use an LSTM to learn Sine waves
- Implement the Neural Style Transfer algorithm on images
- Reinforcement Learning with Actor Critic and REINFORCE algorithms on OpenAI gym
- PyTorch Module Transformations using fx
- Distributed PyTorch examples with Distributed Data Parallel and RPC
- Several examples illustrating the C++ Frontend

Additionally, a list of good examples hosted in their own repositories:

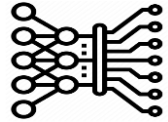
- Neural Machine Translation using sequence-to-sequence RNN with attention (OpenNMT)

main · 32 branches · 0 tags · Go to fi

fduwjj [PT-D] Add TP and DDP into the example run script (#1014) 78acb79 5 days ago

.github	Update .github/workflows/doc-build.yml
cpp	Merge pull request #809 from soumyadipghosh/dist-cpp
dcgan	updated dcgan docs
distributed	[PT-D] Add TP and DDP into the example run script (#1014)
docs	included siamese network example to docs
fast_neural_style	fast neural style: run onnxmodel using onnxruntime and correcting war...
fx	deleted vmap and nnc compile from fx
imagenet	If the dataset is not exactly divisible by world_size, the validation...
legacy/snli	added fx tests
mnist	Use regular dropout rather than dropout2d
mnist_hogwild	tweak running examples without cuda (#794)
regression	fix the issue PR #799 left (#781)
reinforcement_learning	updated CartPole-v0 to CartPole-v1
siamese_network	better description
super_resolution	Remove deprecated & unused Variable() (#496)
time_sequence_prediction	Replace chunk with split
vae	Update main.py
word_language_model	Merge pull request #909 from kyox24/master
.gitignore	add siamese network example
CODEOWNERS	added fx tests
CODE_OF_CONDUCT.md	OSS Automated Fix: Addition of Code of Conduct
CONTRIBUTING.md	Update a few punctuations in CONTRIBUTING.md (#1016)
LICENSE	Create LICENSE
README.md	Update README.md
run_python_examples.sh	[PT-D] Add TP and DDP into the example run script (#1014)
runtime.txt	Adding runtime.txt for Netlify

PyTorch Examples (2/3)



Various PyTorch examples can use to learn and experiment

Image Classification Using ConvNets

This example demonstrates how to run image classification with Convolutional Neural Networks ConvNets on the MNIST database.

GO TO EXAMPLE

Measuring Similarity using Siamese Network

This example demonstrates how to measure similarity between two images using Siamese network on the MNIST database.

GO TO EXAMPLE

Super-resolution Using an Efficient Sub-Pixel CNN

This example demonstrates how to use the sub-pixel convolution layer described in Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network paper. This example trains a super-resolution network on the BSD300 dataset.

GO TO EXAMPLE

HOGWILD! Training of Shared ConvNets

HOGWILD! is a scheme that allows Stochastic Gradient Descent (SGD) parallelization without memory locking. This example demonstrates how to perform HOGWILD! training of shared ConvNets on MNIST.

GO TO EXAMPLE

Word-level Language Modeling using RNN and Transformer

This example demonstrates how to train a multi-layer recurrent neural network (RNN), such as Elman, GRU, or LSTM, or Transformer on a language modeling task by using the Wikitext-2 dataset.

GO TO EXAMPLE

Training ImageNet Classifiers

This example demonstrates how you can train some of the most popular model architectures, including ResNet, AlexNet, and VGG on the ImageNet dataset.

GO TO EXAMPLE

Training a CartPole to balance in OpenAI Gym with actor-critic

This reinforcement learning tutorial demonstrates how to train a CartPole to balance in the OpenAI Gym toolkit by using the Actor-Critic method.

GO TO EXAMPLE

Time Sequence Prediction

This beginner example demonstrates how to use LSTMCell to learn sine wave signals to predict the signal values in the future.

GO TO EXAMPLE

Generative Adversarial Networks (DCGAN)

This example implements the Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks paper.

GO TO EXAMPLE

Variational Auto-Encoders

This example implements the Auto-Encoding Variational Bayes paper with ReLUs and the Adam optimizer.

GO TO EXAMPLE

Implement the Neural Style Transfer algorithm on images

This tutorial demonstrates how you can use PyTorch's implementation of the Neural Style Transfer (NST) algorithm on images.

GO TO EXAMPLE

PyTorch Module Transformations using fx

This set of examples demonstrates the torch.fx toolkit. For more information about *torch.fx*, see torch.fx Overview.

GO TO EXAMPLE

PyTorch Examples (3/3)

PyTorch

Various PyTorch examples can use to learn and experiment

Image Classification Using ConvNets

This example demonstrates how to run image classification with Convolutional Neural Networks ConvNets on the MNIST database.

GO TO EXAMPLE

Measuring Similarity using Siamese Network

This example demonstrates how to measure similarity between two images using Siamese network on the MNIST database.

GO TO EXAMPLE

Super-resolution Using an Efficient Sub-Pixel CNN

This example demonstrates how to use the sub-pixel convolution layer described in Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network paper. This example trains a super-resolution network on the BSD300 dataset.

GO TO EXAMPLE

HOGWILD! Training of Shared ConvNets

HOGWILD! is a scheme that allows Stochastic Gradient Descent (SGD) parallelization without memory locking. This example demonstrates how to perform HOGWILD! training of shared ConvNets on MNIST.

GO TO EXAMPLE

Word-level Language Modeling using RNN and Transformer

This example demonstrates how to train a multi-layer recurrent neural network (RNN), such as Elman, GRU, or LSTM, or Transformer on a language modeling task by using the Wikitext-2 dataset.

GO TO EXAMPLE

Training ImageNet Classifiers

This example demonstrates how you can train some of the most popular model architectures, including ResNet, AlexNet, and VGG on the ImageNet dataset.

GO TO EXAMPLE

Training a CartPole to balance in OpenAI Gym with actor-critic

This reinforcement learning tutorial demonstrates how to train a CartPole to balance in the OpenAI Gym toolkit by using the Actor-Critic method.

GO TO EXAMPLE

Time Sequence Prediction

This beginner example demonstrates how to use LSTMCell to learn sine wave signals to predict the signal values in the future.

GO TO EXAMPLE

Generative Adversarial Networks (DCGAN)

This example implements the Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks paper.

GO TO EXAMPLE

Variational Auto-Encoders

This example implements the Auto-Encoding Variational Bayes paper with ReLUs and the Adam optimizer.

GO TO EXAMPLE

Implement the Neural Style Transfer algorithm on images

This tutorial demonstrates how you can use PyTorch's implementation of the Neural Style Transfer (NST) algorithm on images.

GO TO EXAMPLE

PyTorch Module Transformations using fx

This set of examples demonstrates the torch.fx toolkit. For more information about *torch.fx*, see torch.fx Overview.

GO TO EXAMPLE

Image Classification on the MNIST (1/8)

PyTorch

Implement of CNN architecture for visual recognition

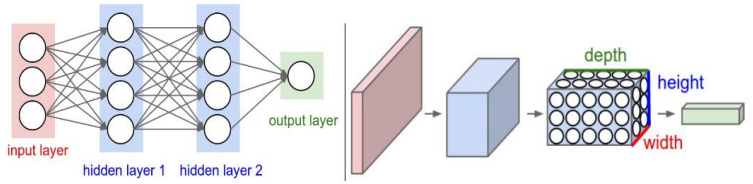
CS231n Convolutional Neural Networks for Visual Recognition

Table of Contents:

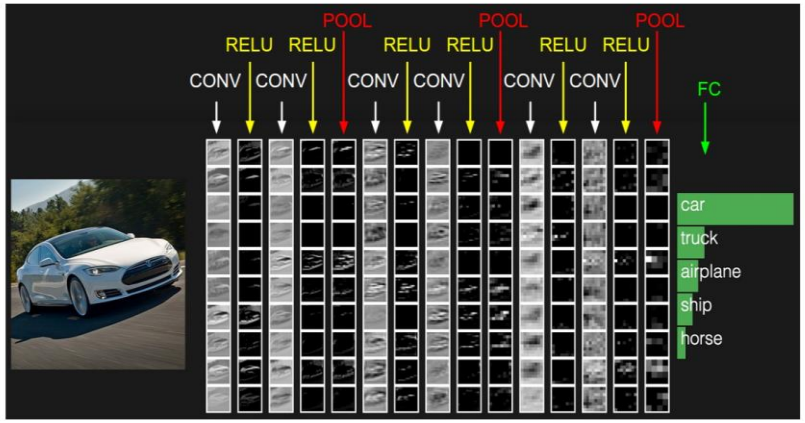
Convolutional Neural Networks (CNNs / ConvNets)

Convolutional Neural Networks are very similar to ordinary Neural Networks from the previous chapter: they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer and all the tips/tricks we developed for learning regular Neural Networks still apply.

So what changes? ConvNet architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network.



Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

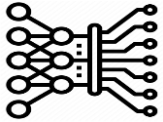


The activations of an example ConvNet architecture. The initial volume stores the raw image pixels (left) and the last volume stores the class scores (right). Each volume of activations along the processing path is shown as a column. Since it's difficult to visualize 3D volumes, we lay out each volume's slices in rows. The last layer volume holds the scores for each class, but here we only visualize the sorted top 5 scores, and print the labels of each one. The full web-based demo is shown in the header of our website. The architecture shown here is a tiny VGG Net, which we will discuss later.

https://cs231n.github.io/convolutional-networks/

Image Classification on the MNIST (2/8)

PyTorch



Developing flow diagram

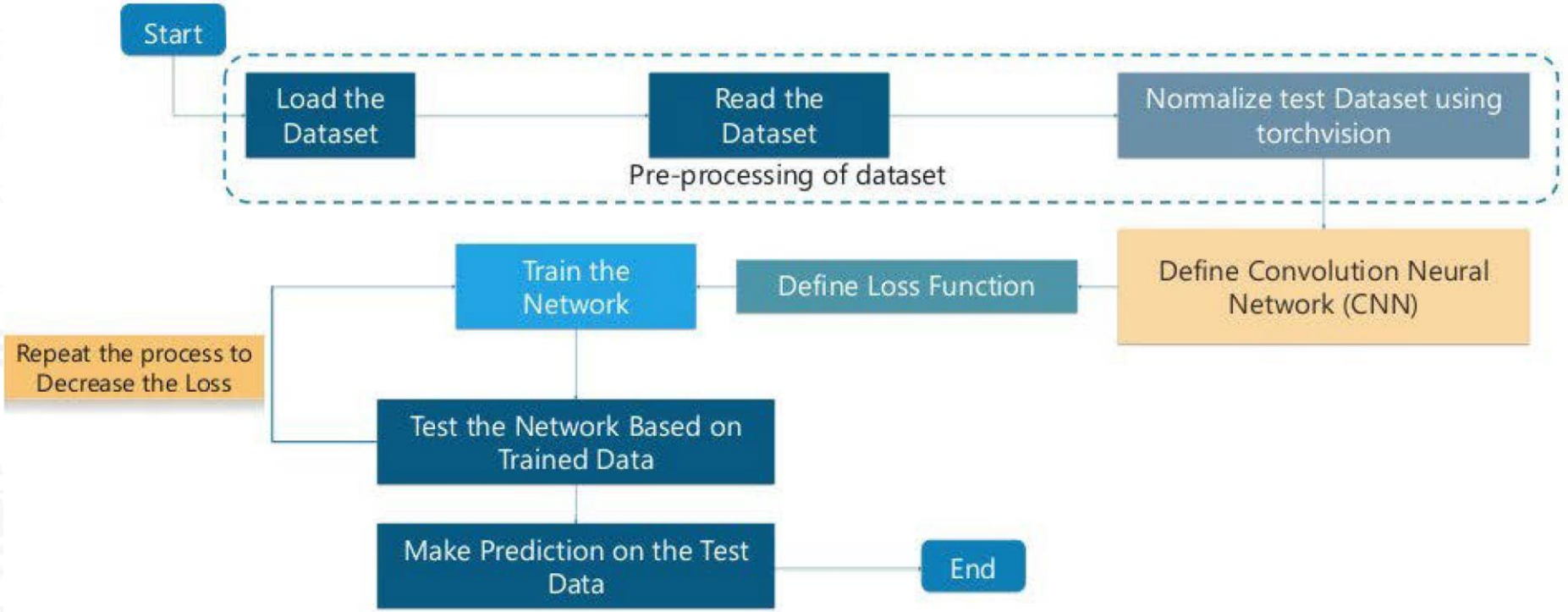
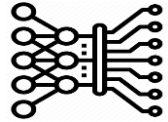


Image Classification on the MNIST (3/8)

PyTorch



Step 1: Loading and normalizing MNIST image dataset

- **Load and normalize dataset**
- Define CNN
- Define loss function
- Train the network
- Validate the network
- Update the weights

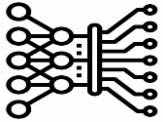
```python
from __future__ import print_function
import argparse
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torchvision import datasets, transforms
from torch.optim.lr_scheduler import StepLR

```

```python
    dataset1 = datasets.MNIST('../data', train=True, download=True,
                              transform=transform)
    dataset2 = datasets.MNIST('../data', train=False,
                              transform=transform)
```

https://github.com/pytorch/examples/blob/main/mnist/main.py

Image Classification on the MNIST (4/8)

PyTorch



Step 2: Define a convolution neural network

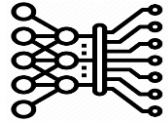
- Load and normalize dataset
- **Define CNN**
- Define loss function
- Train the network
- Validate the network
- Update the weights

```python
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, 3, 1)
        self.conv2 = nn.Conv2d(32, 64, 3, 1)
        self.dropout1 = nn.Dropout(0.25)
        self.dropout2 = nn.Dropout(0.5)
        self.fc1 = nn.Linear(9216, 128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        x = self.conv1(x)
        x = F.relu(x)
        x = self.conv2(x)
        x = F.relu(x)
        x = F.max_pool2d(x, 2)
        x = self.dropout1(x)
        x = torch.flatten(x, 1)
        x = self.fc1(x)
        x = F.relu(x)
        x = self.dropout2(x)
        x = self.fc2(x)
        output = F.log_softmax(x, dim=1)
        return output

```

https://github.com/pytorch/examples/blob/main/mnist/main.py

Image Classification on the MNIST (5/8)

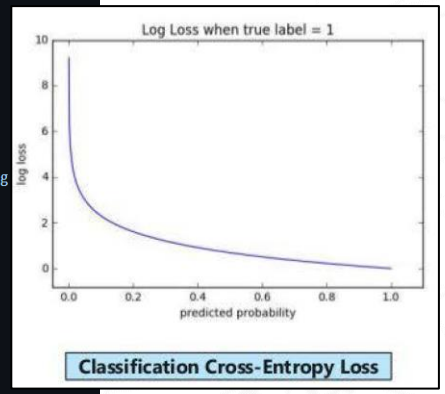
PyTorch



Step 3: Define a loss function and optimizer

- Load and normalize dataset
- Define CNN
- **Define loss function**
- Train the network
- Validate the network
- Update the weights

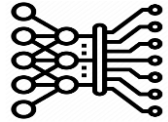
```python
def main():
    # Training settings
    parser = argparse.ArgumentParser(description='PyTorch MNIST Example')
    parser.add_argument('--batch-size', type=int, default=64, metavar='N',
                        help='input batch size for training (default: 64)')
    parser.add_argument('--test-batch-size', type=int, default=1000, metavar='N',
                        help='input batch size for testing (default: 1000)')
    parser.add_argument('--epochs', type=int, default=14, metavar='N',
                        help='number of epochs to train (default: 14)')
    parser.add_argument('--lr', type=float, default=1.0, metavar='LR',
                        help='learning rate (default: 1.0)')
    parser.add_argument('--gamma', type=float, default=0.7, metavar='M',
                        help='Learning rate step gamma (default: 0.7)')
    parser.add_argument('--no-cuda', action='store_true', default=False,
                        help='disables CUDA training')
    parser.add_argument('--dry-run', action='store_true', default=False,
                        help='quickly check a single pass')
    parser.add_argument('--seed', type=int, default=1, metavar='S',
                        help='random seed (default: 1)')
    parser.add_argument('--log-interval', type=int, default=10, metavar='N',
                        help='how many batches to wait before logging training
    parser.add_argument('--save-model', action='store_true', default=False,
                        help='For Saving the current Model')
    args = parser.parse_args()
    use_cuda = not args.no_cuda and torch.cuda.is_available()

    torch.manual_seed(args.seed)

    device = torch.device("cuda" if use_cuda else "cpu")

```



https://github.com/pytorch/examples/blob/main/mnist/main.py

Image Classification on the MNIST (6/8)

PyTorch



Step 4: Train the network

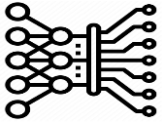
- Load and normalize dataset
- Define CNN
- Define loss function
- **Train the network**
- Validate the network
- Update the weights

```python
def train(args, model, device, train_loader, optimizer, epoch):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        output = model(data)
        loss = F.nll_loss(output, target)
        loss.backward()
        optimizer.step()
        if batch_idx % args.log_interval == 0:
            print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
                epoch, batch_idx * len(data), len(train_loader.dataset),
                100. * batch_idx / len(train_loader), loss.item()))
            if args.dry_run:
                break

```

https://github.com/pytorch/examples/blob/main/mnist/main.py

Image Classification on the MNIST (7/8)

PyTorch



Step 5: Validate the network

- Load and normalize dataset
- Define CNN
- Define loss function
- Train the network
- Validate the network
- Update the weights

```
53
54  def test(model, device, test_loader):
55      model.eval()
56      test_loss = 0
57      correct = 0
58      with torch.no_grad():
59          for data, target in test_loader:
60              data, target = data.to(device), target.to(device)
61              output = model(data)
62              test_loss += F.nll_loss(output, target, reduction='sum').item()  # sum up batch loss
63              pred = output.argmax(dim=1, keepdim=True)  # get the index of the max log-probability
64              correct += pred.eq(target.view_as(pred)).sum().item()
65
66      test_loss /= len(test_loader.dataset)
67
68      print('\nTest set: Average loss: {:.4f}, Accuracy: {}/{} ({:.0f}%)\n'.format(
69          test_loss, correct, len(test_loader.dataset),
70          100. * correct / len(test_loader.dataset)))
```

https://github.com/pytorch/examples/blob/main/mnist/main.py

Image Classification on the MNIST (8/8)

Step 5: Update the wights and parameter tuning and save the model

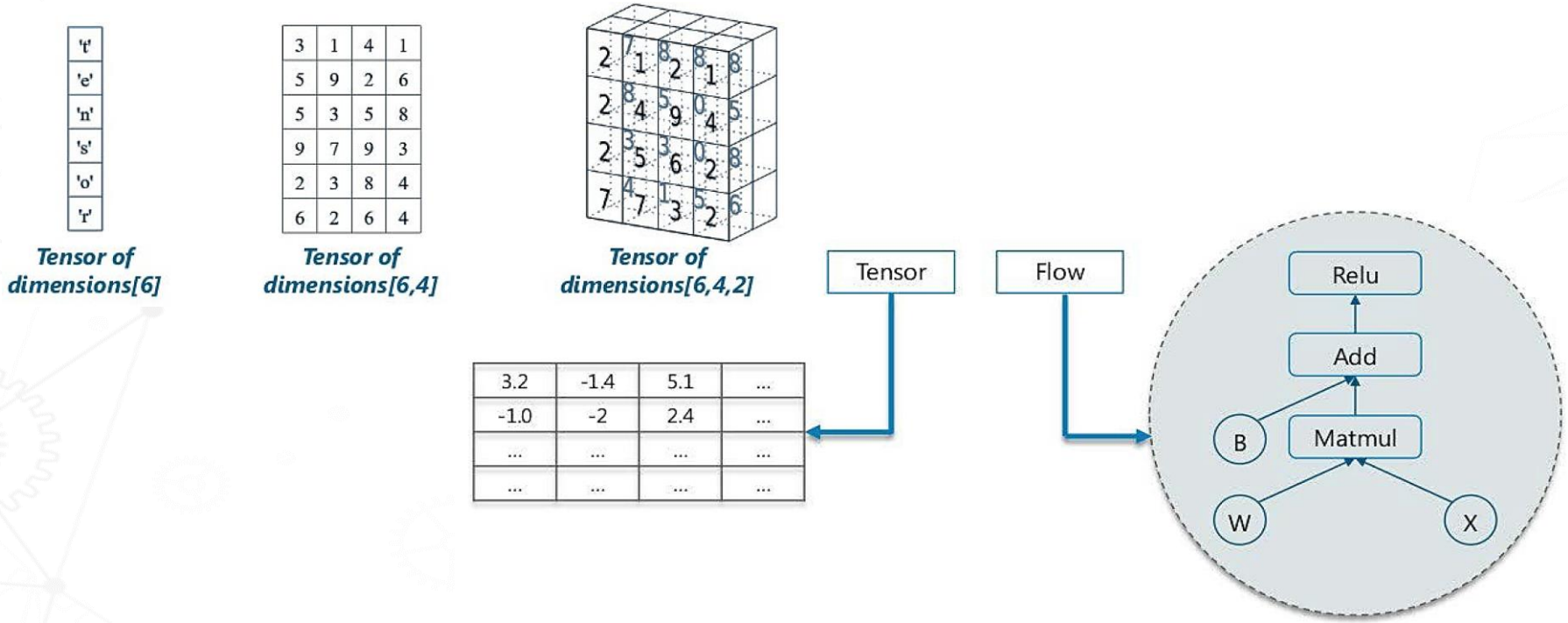
- Load and normalize dataset
- Define CNN
- Define loss function
- Train the network
- Validate the network
- Update the weights

```
120     train_loader = torch.utils.data.DataLoader(dataset1,**train_kwargs)
121     test_loader = torch.utils.data.DataLoader(dataset2, **test_kwargs)
122
123     model = Net().to(device)
124     optimizer = optim.Adadelta(model.parameters(), lr=args.lr)
125
126     scheduler = StepLR(optimizer, step_size=1, gamma=args.gamma)
127     for epoch in range(1, args.epochs + 1):
128         train(args, model, device, train_loader, optimizer, epoch)
129         test(model, device, test_loader)
130         scheduler.step()
131
132     if args.save_model:
133         torch.save(model.state_dict(), "mnist_cnn.pt")
134
135
136 if __name__ == '__main__':
137     main()
```

https://github.com/pytorch/examples/blob/main/mnist/main.py

The standard way of representing data in deep learning

- ✓ Just multidimensional arrays, an extension of two-dimensional matrices to data with higher dimension
- ✓ In Tensorflow, computation is approached as a dataflow graph



TensorFlow (Cont.)

Why TensorFlow

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

About →

Easy model building

Build and train ML models easily using intuitive high-level APIs like Keras with eager execution, which makes for immediate model iteration and easy debugging.

Robust ML production anywhere

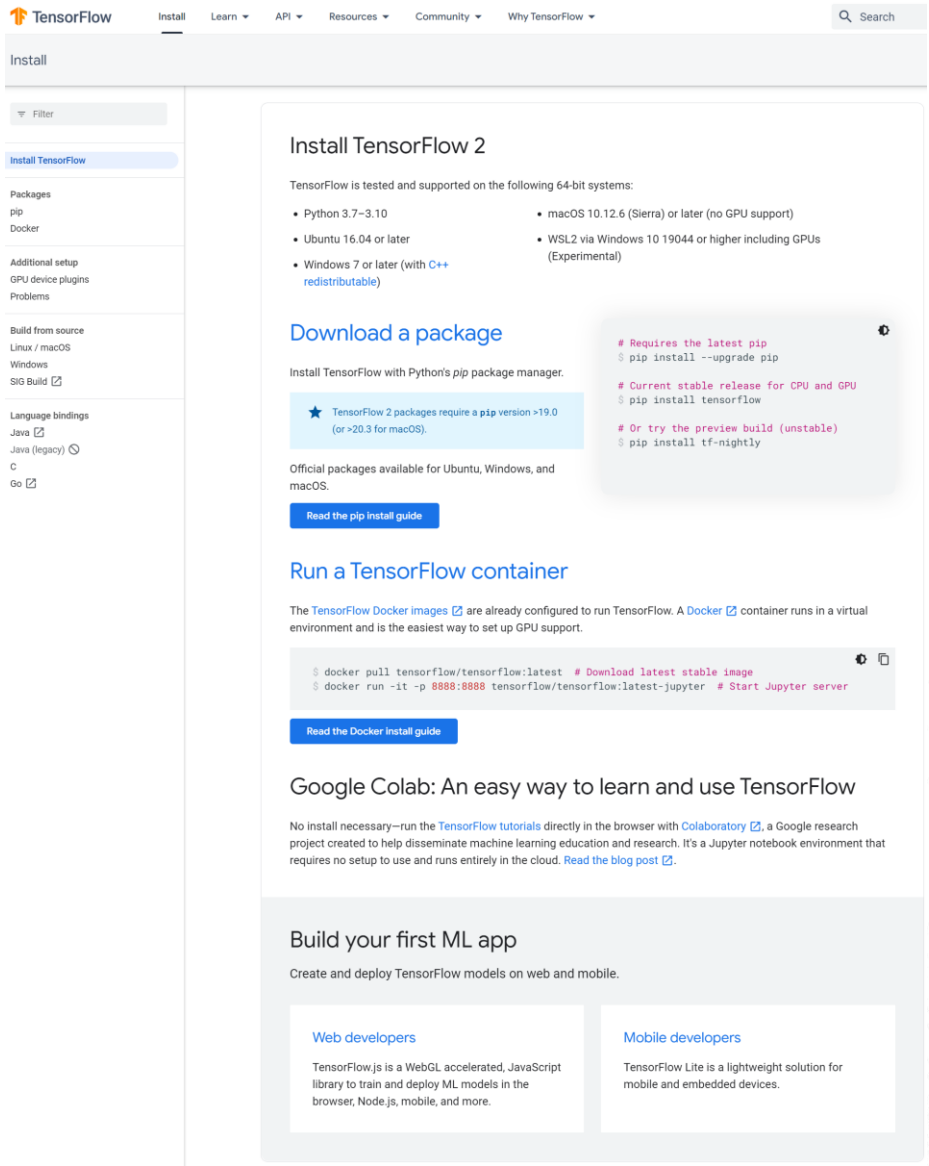
Easily train and deploy models in the cloud, on-prem, in the browser, or on-device no matter what language you use.

Powerful experimentation for research

A simple and flexible architecture to take new ideas from concept to code, to state-of-the-art models, and to publication faster.

https://www.tensorflow.org/

TensorFlow Tutorials (1/9)



https://www.tensorflow.org/

TensorFlow Tutorials (2/9)

GPU device plugins

Note: This page is for non-NVIDIA® GPU devices. For NVIDIA® GPU support, go to the Install TensorFlow with pip guide.

TensorFlow's pluggable device{.external} architecture adds new device support as separate plug-in packages that are installed alongside the official TensorFlow package.

The mechanism requires no device-specific changes in the TensorFlow code. It relies on C APIs to communicate with the TensorFlow binary in a stable manner. Plug-in developers maintain separate code repositories and distribution packages for their plugins and are responsible for testing their devices.

Use device plugins

To use a particular device, like one would a native device in TensorFlow, users only have to install the device plug-in package for that device. The following code snippet shows how the plugin for a new demonstration device, *Awesome Processing Unit (APU)*, is installed and used. For simplicity, this sample APU plug-in only has one custom kernel for ReLU:

```
# Install the APU example plug-in package
$ pip install tensorflow-apu-0.0.1-cp36-cp36m-linux_x86_64.whl
...
Successfully installed tensorflow-apu-0.0.1
```

With the plug-in installed, test that the device is visible and run an operation on the new APU device:

```
import tensorflow as tf   # TensorFlow registers PluggableDevices here.
tf.config.list_physical_devices()  # APU device is visible to TensorFlow.
[PhysicalDevice(name='/physical_device:CPU:0', device_type='CPU'), PhysicalDevice(name='/physical_device:

a = tf.random.normal(shape=[5], dtype=tf.float32)  # Runs on CPU.
b =  tf.nn.relu(a)         # Runs on APU.

with tf.device("/APU:0"):  # Users can also use 'with tf.device' syntax.
  c = tf.nn.relu(a)        # Runs on APU.

with tf.device("/CPU:0"):
  c = tf.nn.relu(a)        # Runs on CPU.

@tf.function  # Defining a tf.function
def run():
  d = tf.random.uniform(shape=[100], dtype=tf.float32)  # Runs on CPU.
  e = tf.nn.relu(d)        # Runs on APU.

run()  # PluggableDevices also work with tf.function and graph mode.
```

https://www.tensorflow.org/

TensorFlow Tutorials (3/9)

TensorFlow is an end-to-end open source platform for machine learning

TensorFlow makes it easy for beginners and experts to create machine learning models. See the sections below to get started.

See tutorials — Tutorials show you how to use TensorFlow with complete, end-to-end examples.

See the guide — Guides explain the concepts and components of TensorFlow.

For beginners

The best place to start is with the user-friendly Sequential API. You can create models by plugging together building blocks. Run the "Hello World" example below, then visit the tutorials to learn more.

To learn ML, check out our education page . Begin with curated curriculums to improve your skills in foundational ML areas.

```python
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Run code now — Try in Google's interactive notebook

For experts

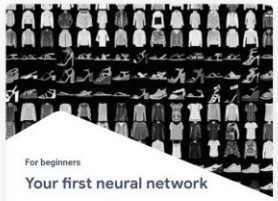
The Subclassing API provides a define-by-run interface for advanced research. Create a class for your model, then write the forward pass imperatively. Easily author custom layers, activations, and training loops. Run the "Hello World" example below, then visit the tutorials to learn more.

```python
class MyModel(tf.keras.Model):
  def __init__(self):
    super(MyModel, self).__init__()
    self.conv1 = Conv2D(32, 3, activation='relu')
    self.flatten = Flatten()
    self.d1 = Dense(128, activation='relu')
    self.d2 = Dense(10, activation='softmax')

  def call(self, x):
    x = self.conv1(x)
    x = self.flatten(x)
    x = self.d1(x)
    return self.d2(x)
model = MyModel()

with tf.GradientTape() as tape:
  logits = model(images)
  loss_value = loss(logits, labels)
grads = tape.gradient(loss_value, model.trainable_variable
optimizer.apply_gradients(zip(grads, model.trainable_varia
```

Run code now — Try in Google's interactive notebook

Solutions to common problems

Explore step-by-step tutorials to help you with your projects.



For beginners

Your first neural network

Train a neural network to classify images of clothing, like sneakers and shirts, in this fast-paced overview of a complete TensorFlow program.

For experts

Generative adversarial networks

Train a generative adversarial network to generate images of handwritten digits, using the Keras Subclassing API.

For experts

Neural machine translation with attention

Train a sequence-to-sequence model for Spanish to English translation using the Keras Subclassing API.

News & announcements

Check out our blog for additional updates, and subscribe to our TensorFlow newsletter to get the latest announcements sent directly to your inbox.

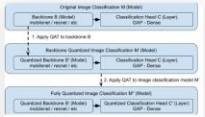
Sign up ↗

June 28, 2022

Bringing Machine Learning to every developer's toolbox

With the release of the recent Stack Overflow Developer Survey, we're delighted to see the growth of TensorFlow as the most-used...

Read the blog →

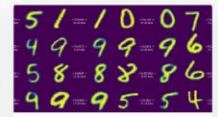


June 10, 2022

Adding Quantization-aware Training and Pruning to the TensorFlow Model Garden

The TensorFlow model optimization toolkit (TFMOT) provides modern optimization techniques such as quantization...

Read the blog →



June 2, 2022

New documentation on tensorflow.org

As Google I/O took place, we published a lot of exciting new docs on tensorflow.org, including updates to model parallelism and...

Read the blog →

May 19, 2022

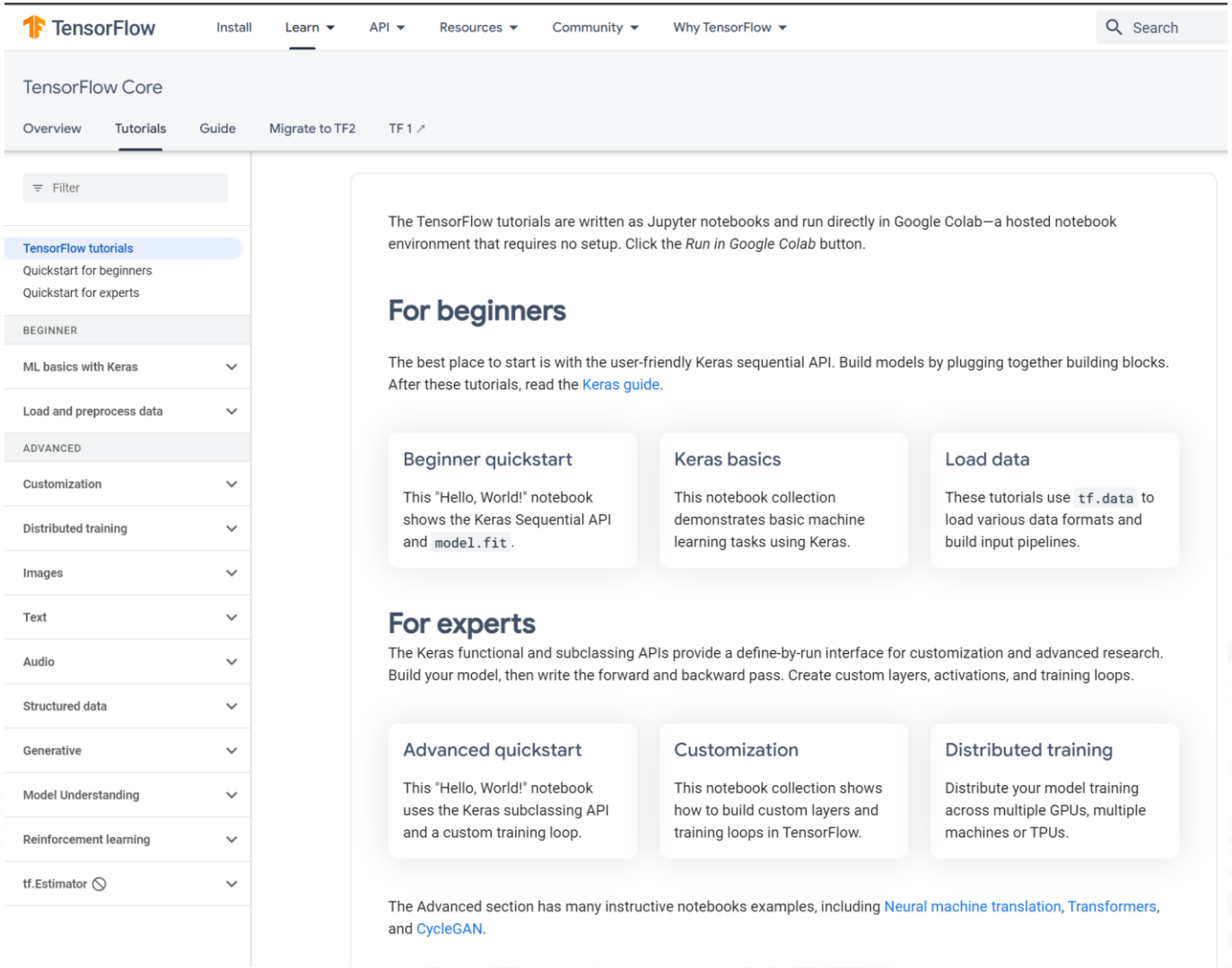
What's new in TensorFlow 2.9?

TensorFlow 2.9 has been released! Highlights include performance improvements with oneDNN, and the release of DTensor, a new API...

Read the blog →

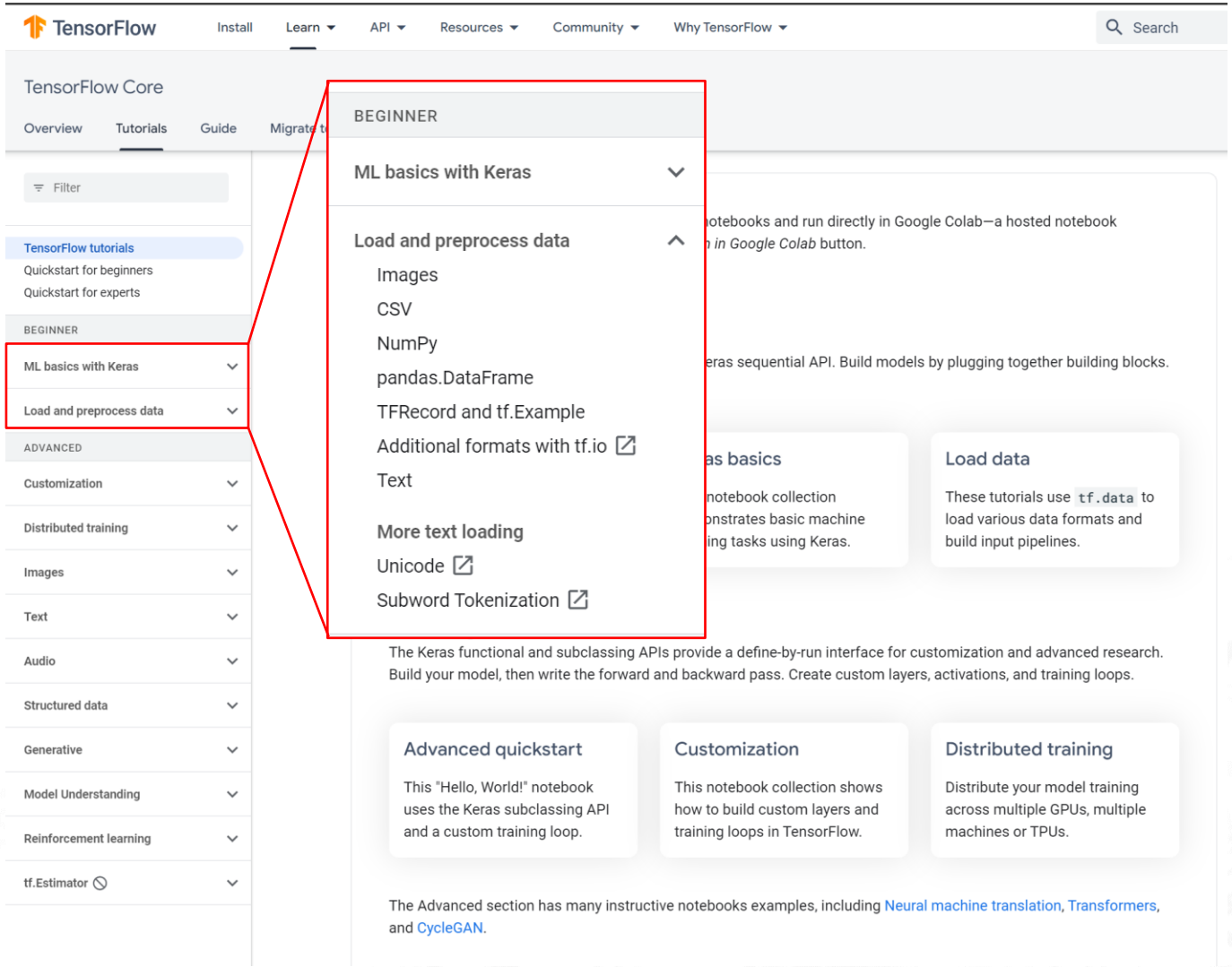
https://www.tensorflow.org/

TensorFlow Tutorials (4/9)



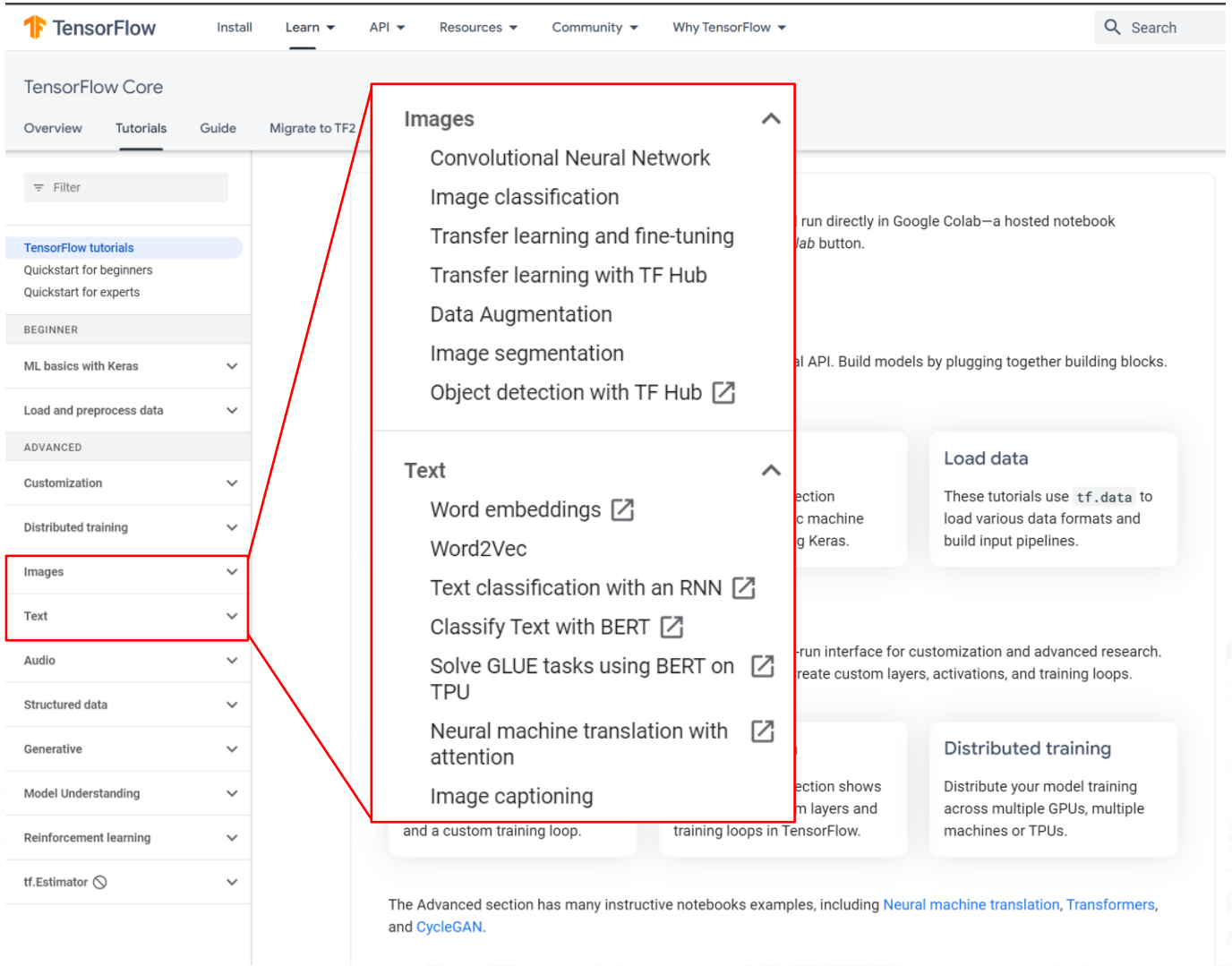
https://www.tensorflow.org/

TensorFlow Tutorials (5/9)



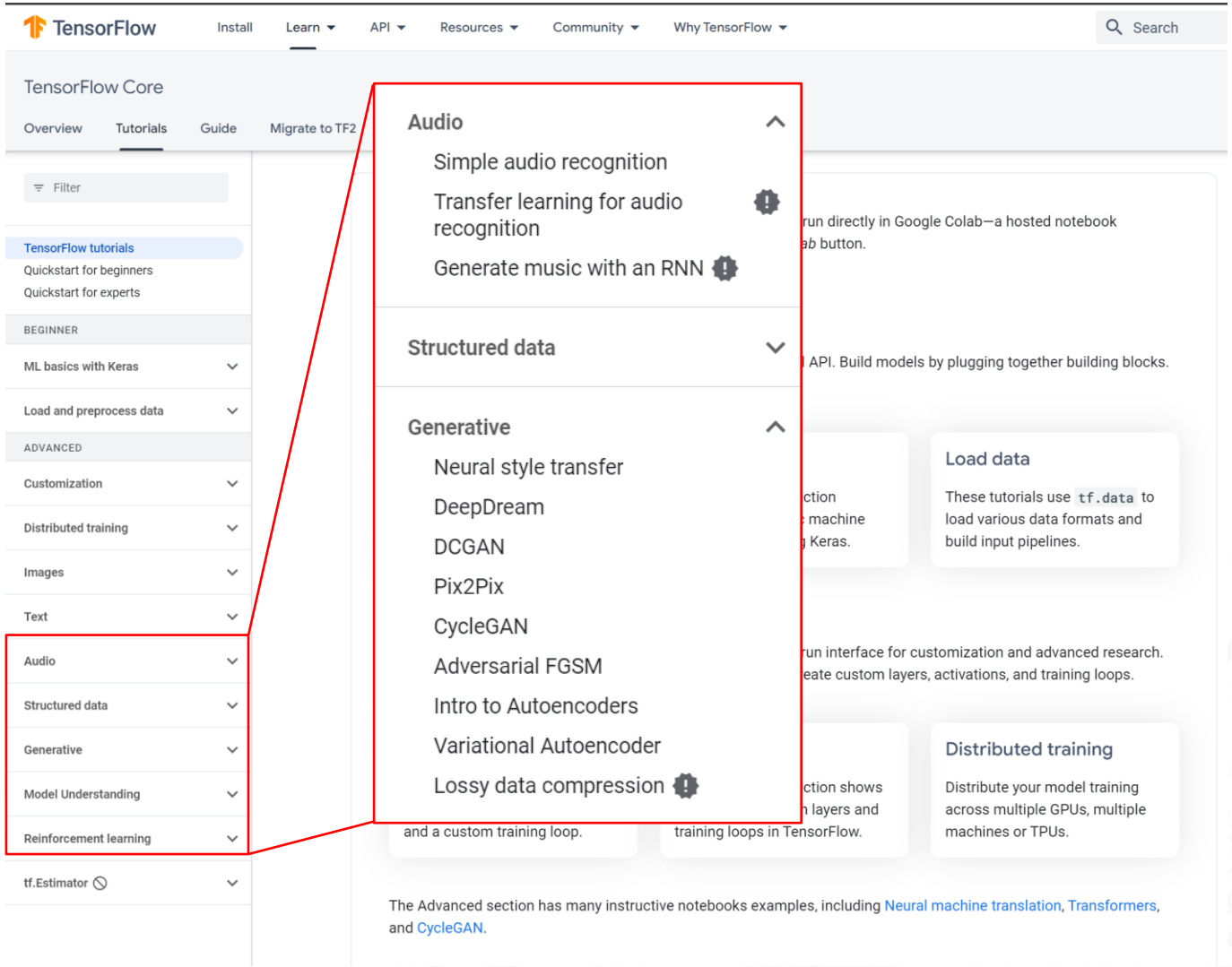
https://www.tensorflow.org/

TensorFlow Tutorials (6/9)



https://www.tensorflow.org/

TensorFlow Tutorials (7/9)



https://www.tensorflow.org/

TensorFlow Tutorials (8/9)

TensorFlow Install Learn API Resources Community Why TensorFlow Search

TensorFlow Core

Overview Tutorials Guide Migrate to TF2 TF 1

Filter

TensorFlow tutorials
Quickstart for beginners
Quickstart for experts

BEGINNER

ML basics with Keras
Load and preprocess data

ADVANCED

Customization
Distributed training
Images
Convolutional Neural Network
Image classification
Transfer learning and fine-tuning
Transfer learning with TF Hub
Data Augmentation
Image segmentation
Object detection with TF Hub
Text
Audio
Structured data
Generative
Model Understanding
Reinforcement learning
tf.Estimator

TensorFlow > Learn > TensorFlow Core > Tutorials

Was this helpful?

Convolutional Neural Network (CNN)

Run in Google Colab | View source on GitHub | Download notebook

This tutorial demonstrates training a simple Convolutional Neural Network (CNN) to classify CIFAR images. Because this tutorial uses the Keras Sequential API, creating and training your model will take just a few lines of code.

Import TensorFlow

```
import tensorflow as tf

from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
```

Download and prepare the CIFAR10 dataset

The CIFAR10 dataset contains 60,000 color images in 10 classes, with 6,000 images in each class. The dataset is divided into 50,000 training images and 10,000 testing images. The classes are mutually exclusive and there is no overlap between them.

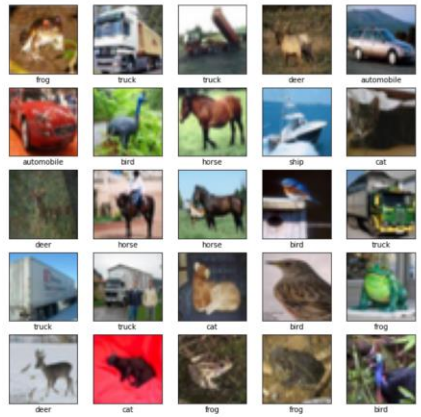
```
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()

# Normalize pixel values to be between 0 and 1
train_images, test_images = train_images / 255.0, test_images / 255.0
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 11s 0us/step
170508288/170498071 [==============================] - 11s 0us/step
```

Verify the data

To verify that the dataset looks correct, let's plot the first 25 images from the training set and display the class name below each image:

```
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']

plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i])
    # The CIFAR labels happen to be arrays,
    # which is why you need the extra index
    plt.xlabel(class_names[train_labels[i][0]])
plt.show()
```



Create the convolutional base

The 6 lines of code below define the convolutional base using a common pattern: a stack of Conv2D and MaxPooling2D layers.

As input, a CNN takes tensors of shape (image_height, image_width, color_channels), ignoring the batch size. If you are new to these dimensions, color_channels refers to (R,G,B). In this example, you will configure your CNN to process inputs of shape (32, 32, 3), which is the format of CIFAR images. You can do this by passing the argument `input_shape` to your first layer.

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

Let's display the architecture of your model so far:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 32)        896

max_pooling2d (MaxPooling2D  (None, 15, 15, 32)        0
)

conv2d_1 (Conv2D)            (None, 13, 13, 64)        18496

max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)          0
2D)
```

https://www.tensorflow.org/

TensorFlow Tutorials (9/9)

TensorFlow > Learn > TensorFlow Core > Tutorials

Was this helpful?

Simple audio recognition: Recognizing keywords

Run in Google Colab | View source on GitHub | Download notebook

This tutorial demonstrates how to preprocess audio files in the WAV format and build and train a basic automatic speech recognition (ASR) model for recognizing ten different words. You will use a portion of the Speech Commands dataset (Warden, 2018), which contains short (one-second or less) audio clips of commands, such as "down", "go", "left", "no", "right", "stop", "up" and "yes".

Real-world speech and audio recognition systems are complex. But, like image classification with the MNIST dataset, this tutorial should give you a basic understanding of the techniques involved.

Setup

Import necessary modules and dependencies. Note that you'll be using seaborn for visualization in this tutorial.

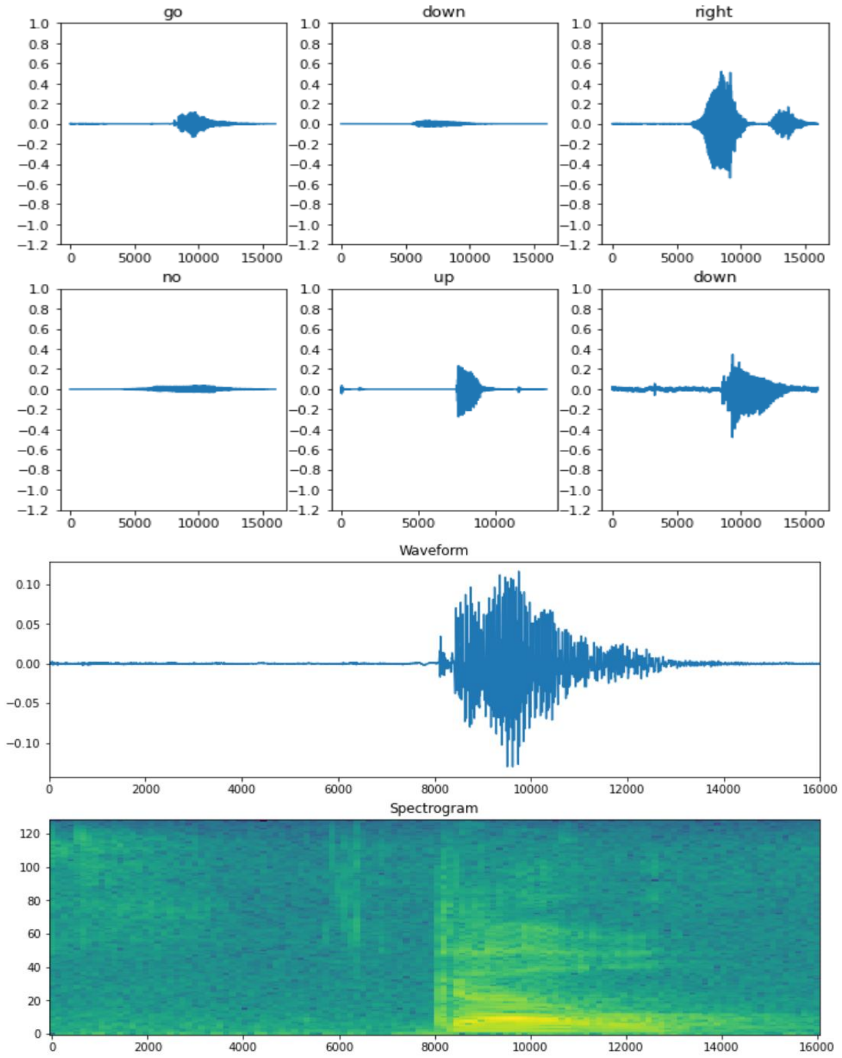
```
import os
import pathlib

import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import tensorflow as tf

from tensorflow.keras import layers
from tensorflow.keras import models
from IPython import display

# Set the seed value for experiment reproducibility.
seed = 42
tf.random.set_seed(seed)
np.random.seed(seed)
```

Import the mini Speech Commands dataset

To save time with data loading, you will be working with a smaller version of the Speech Commands dataset. The original dataset consists of over 105,000 audio files in the WAV (Waveform) audio file format of people saying 35 different words. This data was collected by Google and released under a CC BY license.

Download and extract the `mini_speech_commands.zip` file containing the smaller Speech Commands datasets with `tf.keras.utils.get_file`:

```
DATASET_PATH = 'data/mini_speech_commands'

data_dir = pathlib.Path(DATASET_PATH)
if not data_dir.exists():
  tf.keras.utils.get_file(
      'mini_speech_commands.zip',
      origin="http://storage.googleapis.com/download.tensorflow.org/data/mini_speech_commands.zip",
      extract=True,
      cache_dir='.', cache_subdir='data')
```



https://www.tensorflow.org/

Comparison of libraries

	Keras	TensorFlow	PyTorch
Level of API	high-level API[1]	Both high & low level APIs	Lower-level API[2]
Speed	Slow	High	High
Architecture	Simple, more readable and concise	Not very easy to use	Complex[3]
Debugging	No need to debug	Difficult to debugging	Good debugging capabilities
Dataset Compatibility	Slow & Small	Fast speed & large	Fast speed & large datasets
Popularity Rank	1	2	3
Uniqueness	Multiple back-end support	Object Detection Functionality	Flexibility & Short Training Duration
Created By	Not a library on its own	Created by Google	Created by Facebook[4]
Ease of use	User-friendly	Incomprehensive API	Integrated with Python language
Computational graphs used	Static graphs	Static graphs	Dynamic computation graphs[5]