KIRD
국가과학기술인력개발원

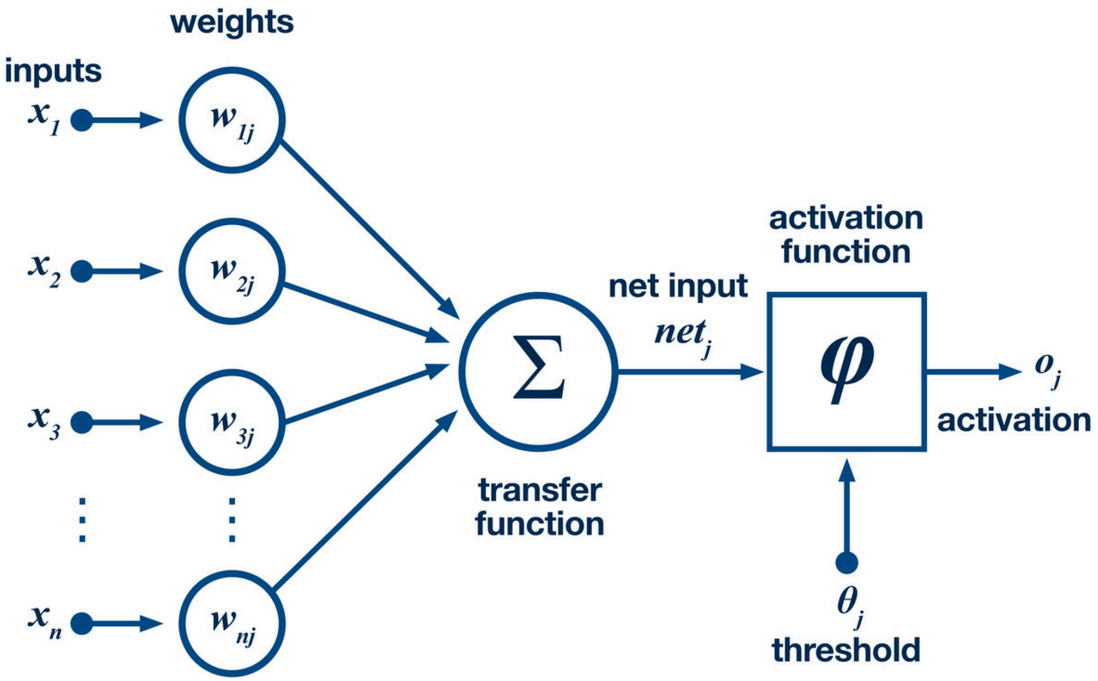
Artificial Neural Networks

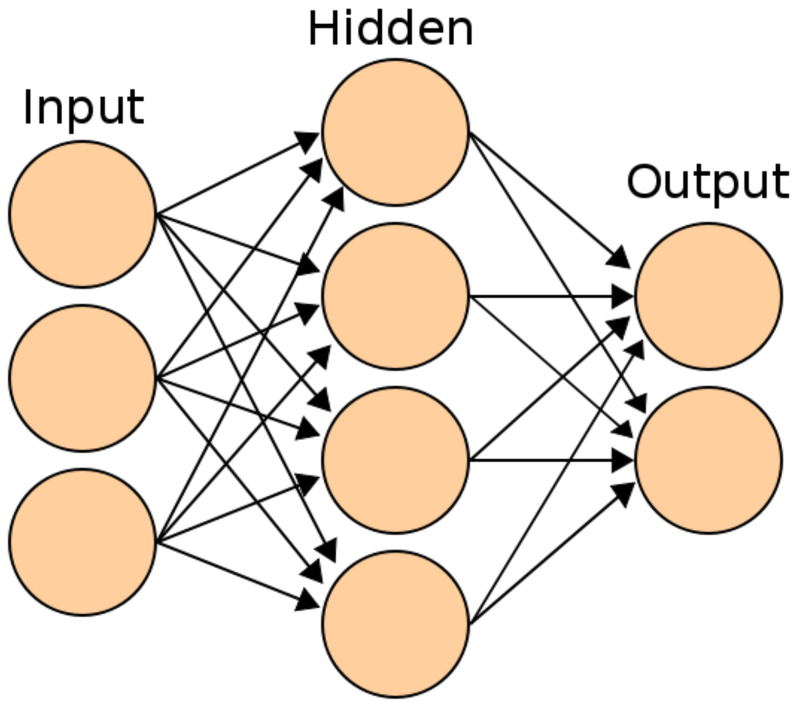
세종대학교 AI로봇학과 류승형

목차

Chapter 1.
Introduction to Neural Network

인공신경망


weights
inputs
x_1
x_2
x_3
x_n
w_{1j}
w_{2j}
w_{3j}
w_{nj}
Σ
transfer function
net input
net_j
activation function
φ
o_j
activation
θ_j
threshold

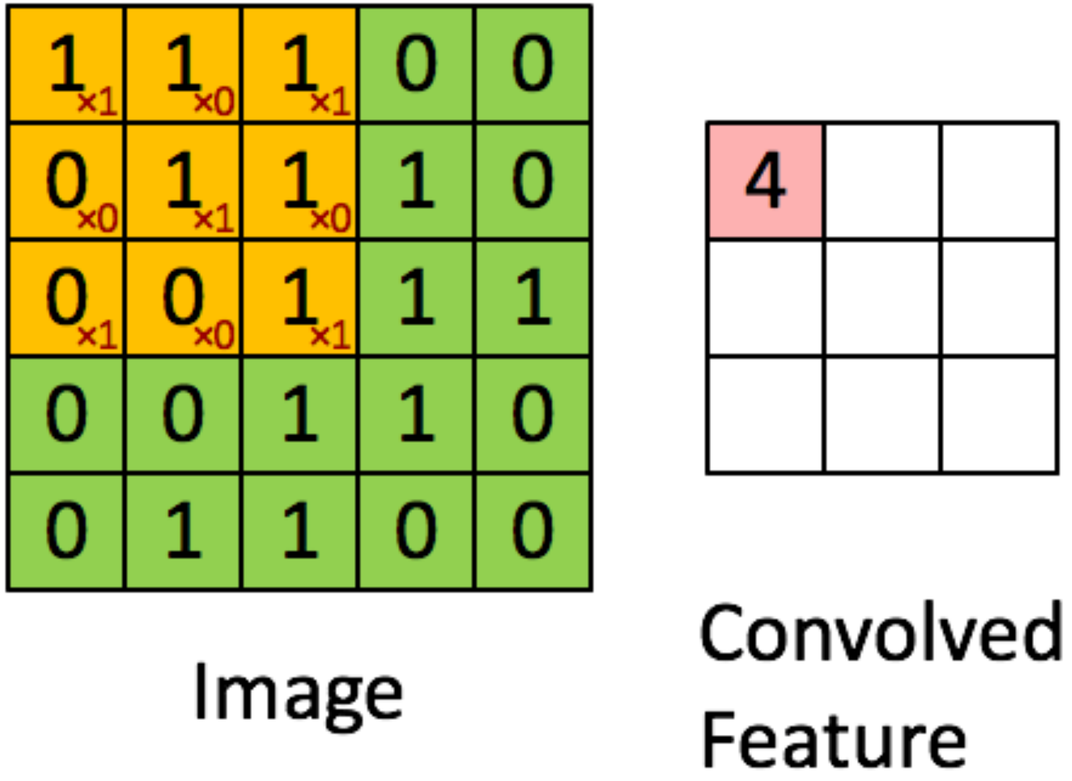

Input
Hidden
Output

Dense (Linear) layer

- 가장 기본적인 신경망 레이어 : Dense (keras) 또는 Linear (torch)
- Matrix multiplication을 통해서 d_{in} 벡터를 → d_{out} 벡터로…
  - $+\alpha$ (activation function, dropout)
- [N x d] → Dense (d, d') → [N x d’]
- [N x L x d] → Dense (d, d') → [N x L x d’]

Convolutional layer

- Kernel 또는 Filter를 움직이면서
- 국소 지역에 대한 weighted summation

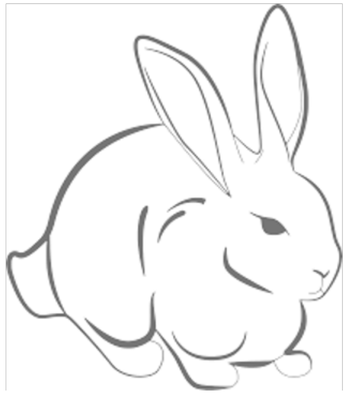


http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

Convolution?

Input data

Kernel



고양이

or

토끼

Multiple kernels

Input data

고양이 1

토끼 1

고양이 2

Kernel

Kernel

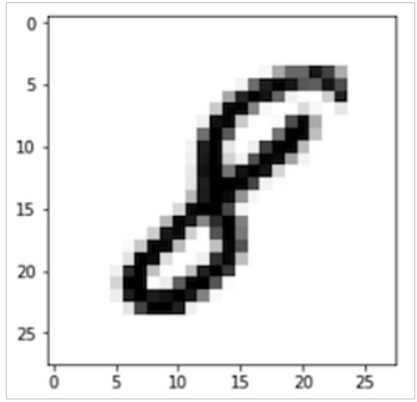
귀

수염

발

Applying multiple kernels

원본 이미지



2D tensor

Kernel 3

Kernel 2

w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}
w_{31}	w_{32}	w_{33}

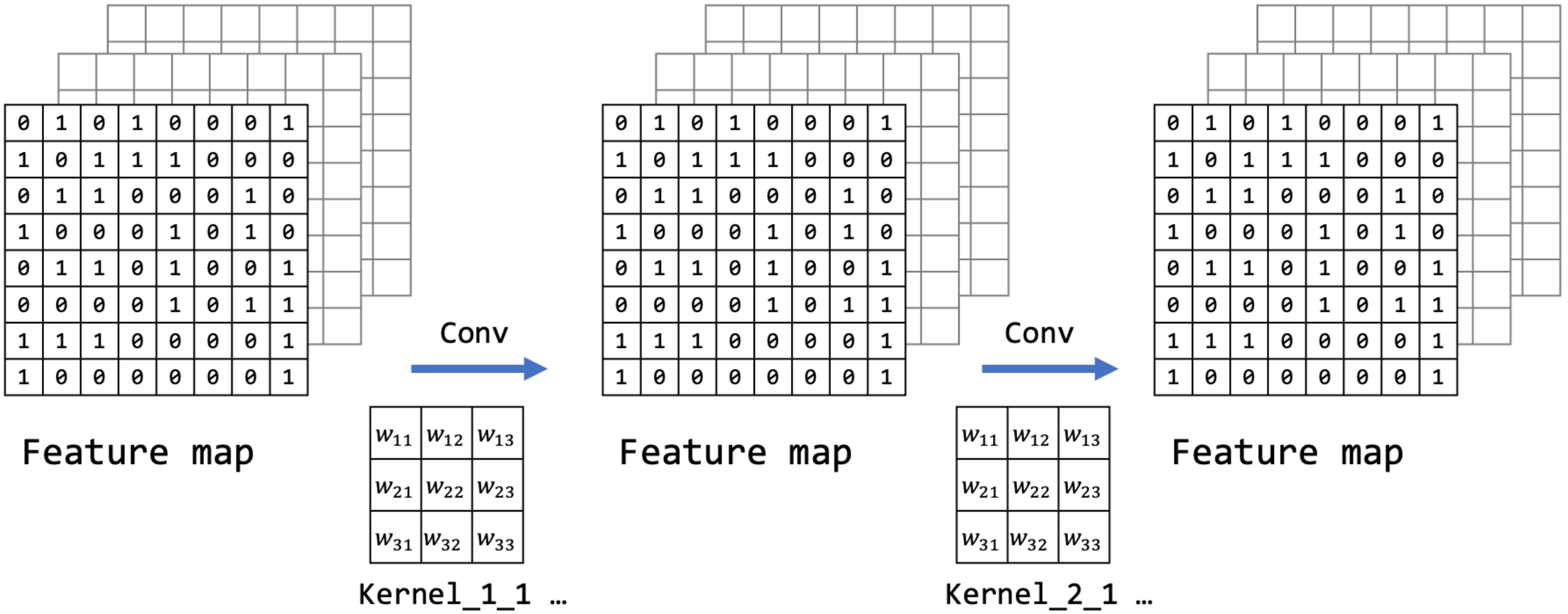
Kernel 1

Convolution

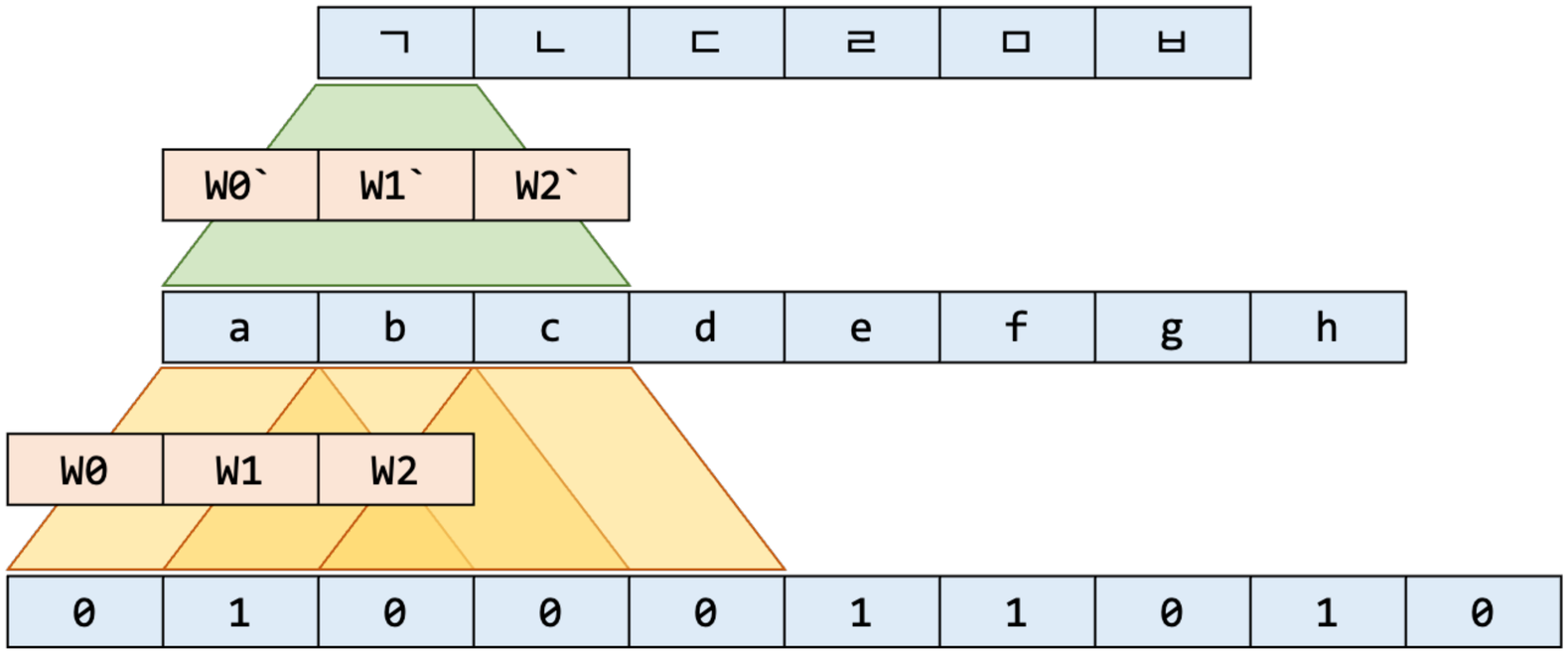
0	1	0	1	0	0	0	1
1	0	1	1	1	0	0	0
0	1	1	0	0	0	1	0
1	0	0	0	1	0	1	0
0	1	1	0	1	0	0	1
0	0	0	0	1	0	1	1
1	1	1	0	0	0	0	1
1	0	0	0	0	0	0	1

Feature map

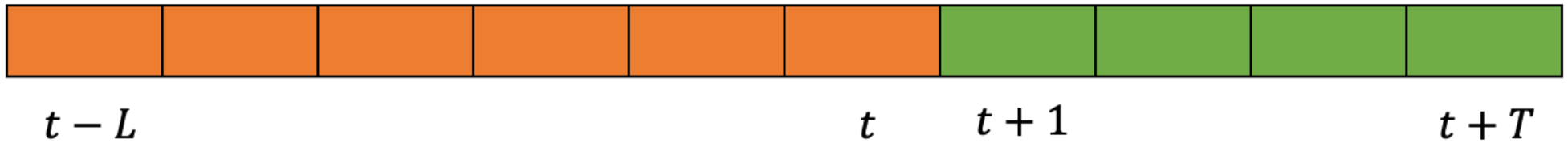
Stack of Convolution Layers



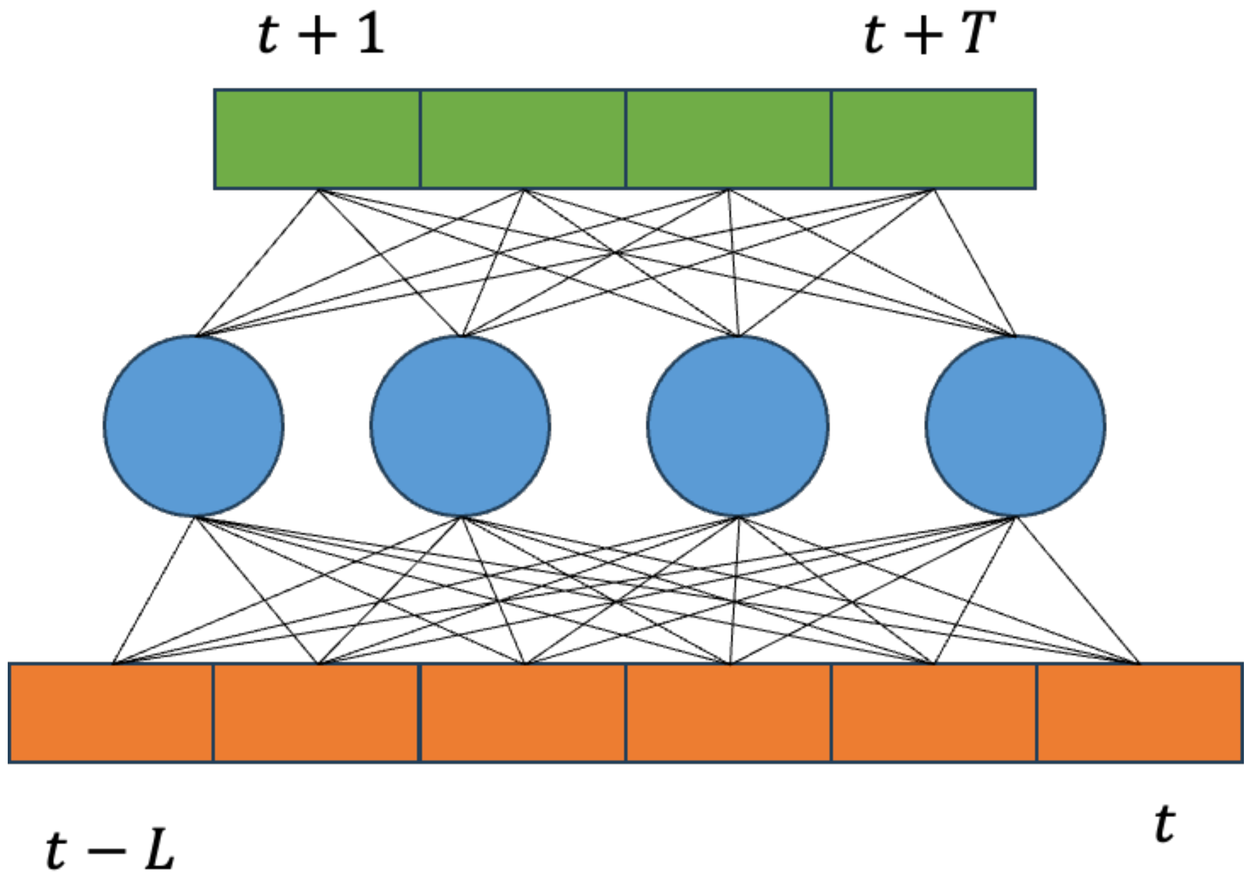
Increasing Region of Interest



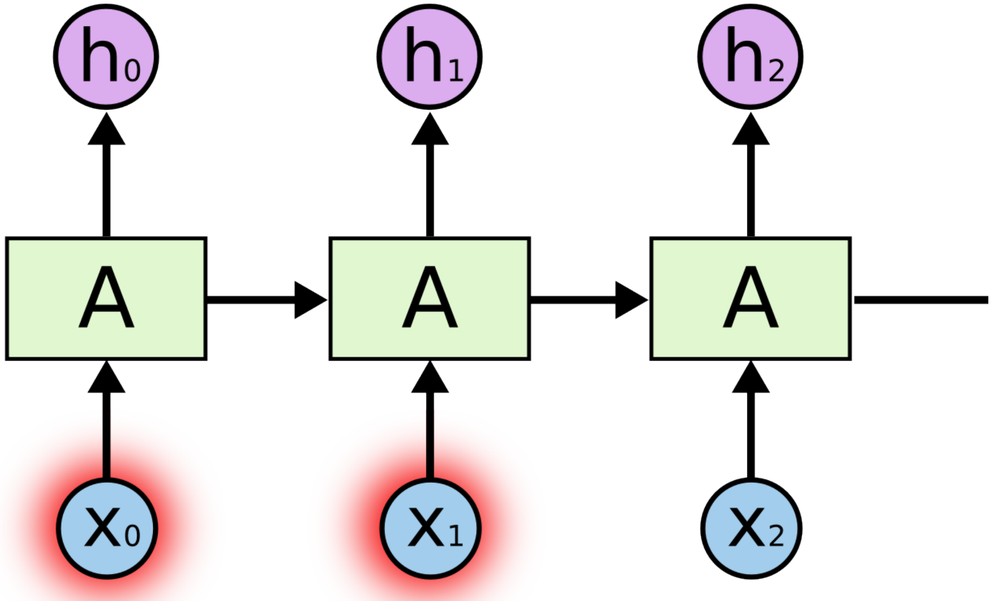
Time Series Data



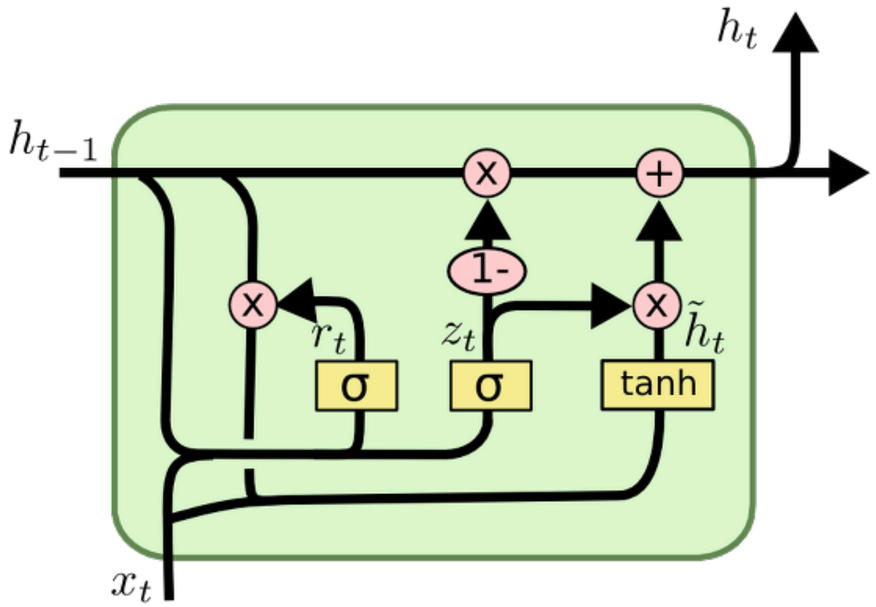
MLP for Time series?



RNN layer

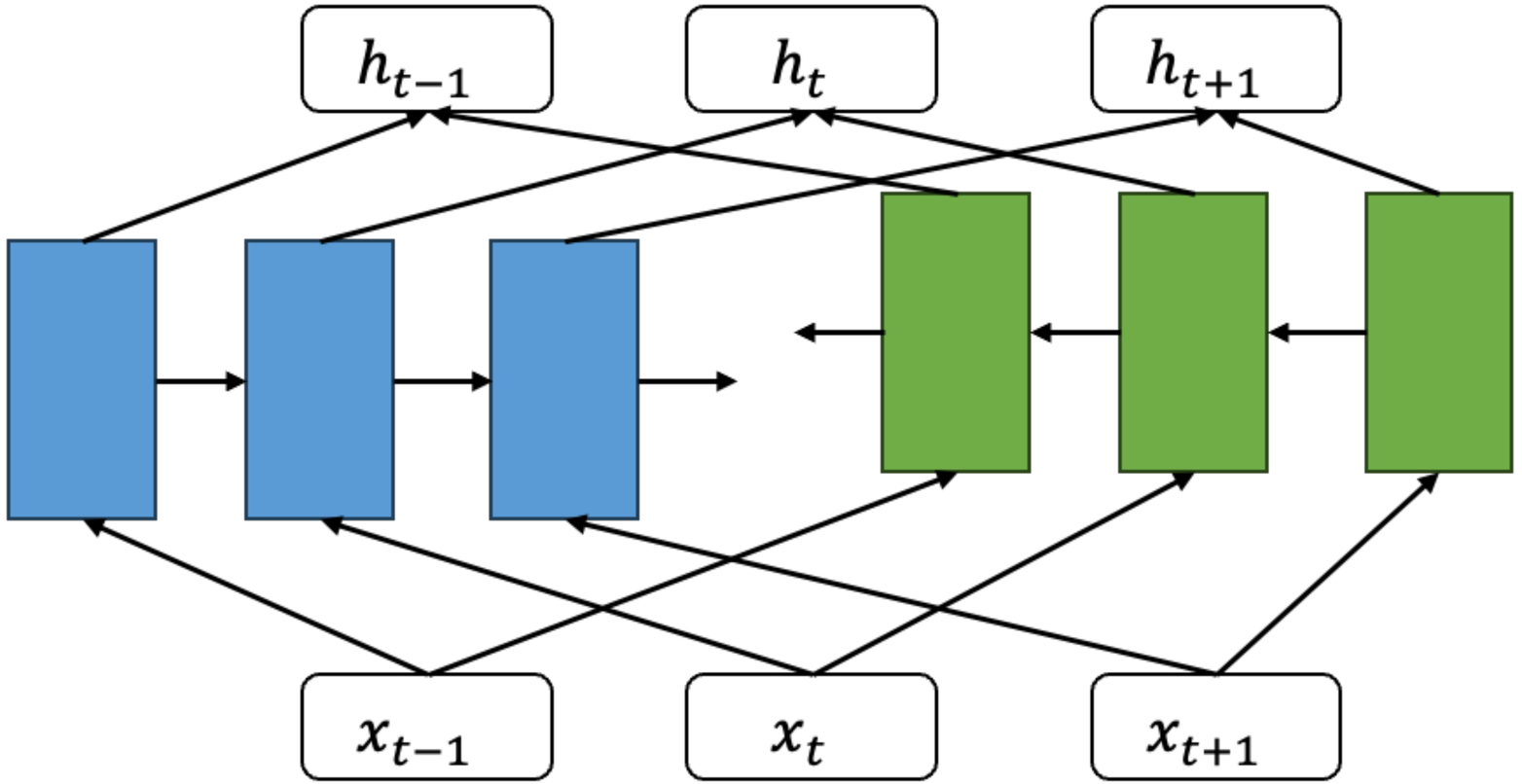


RNN

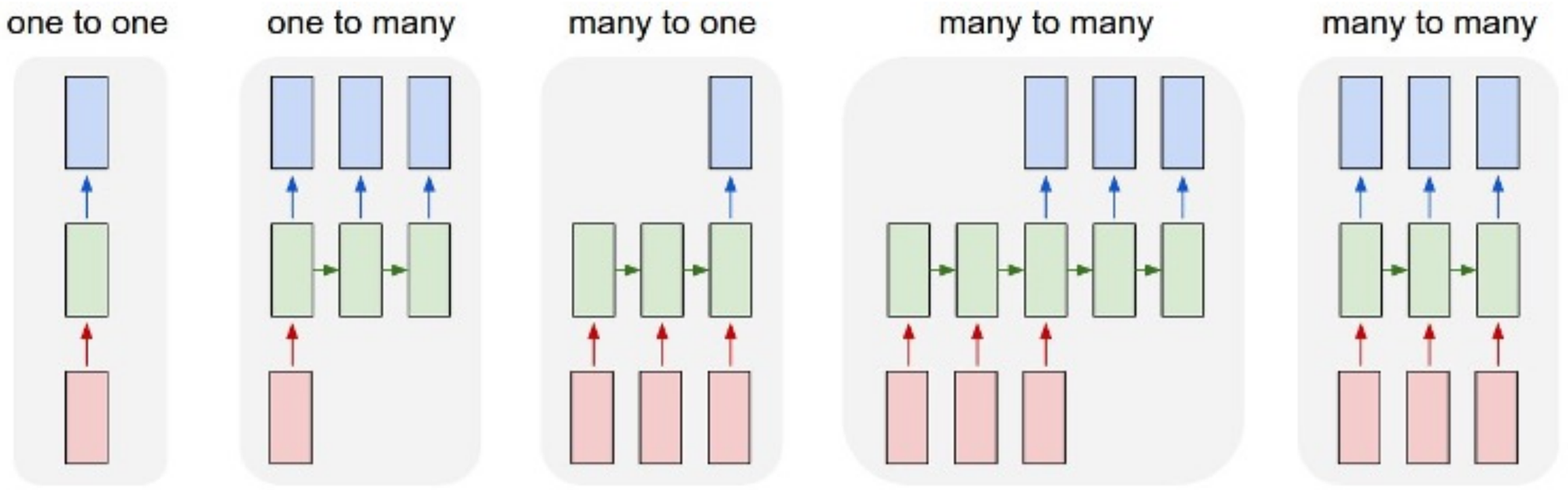


LSTM

RNN option



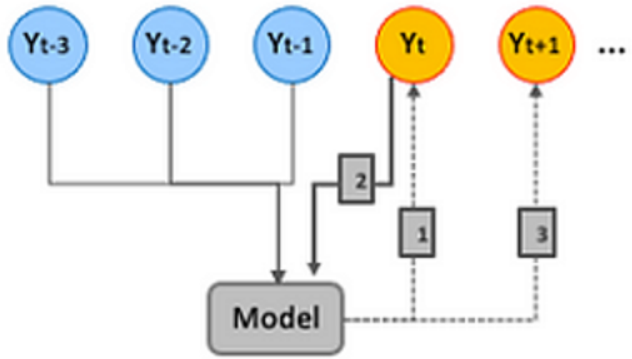
Dealing with Time-Series Data



Forecasting Approach

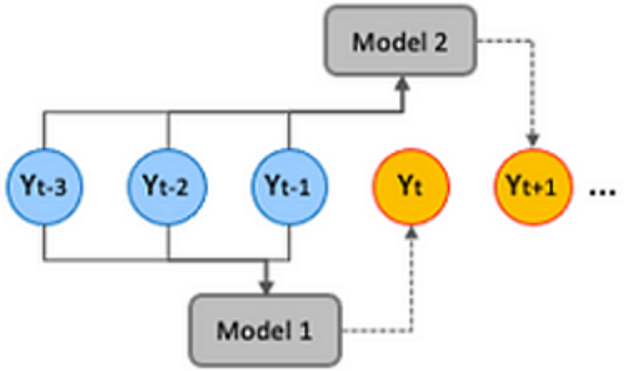
Recursive Approach

The one-step predicted value is used for the feature in the next step. This continues recursively until the model predict the final period.



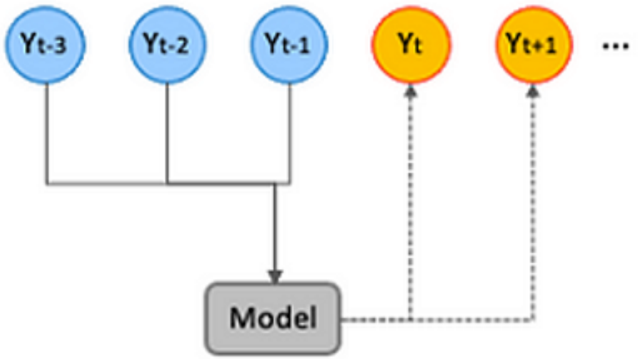
Direct Approach

The multiple different models are developed to predict multiple forecasting periods. Each single model only takes care of a single prediction period.

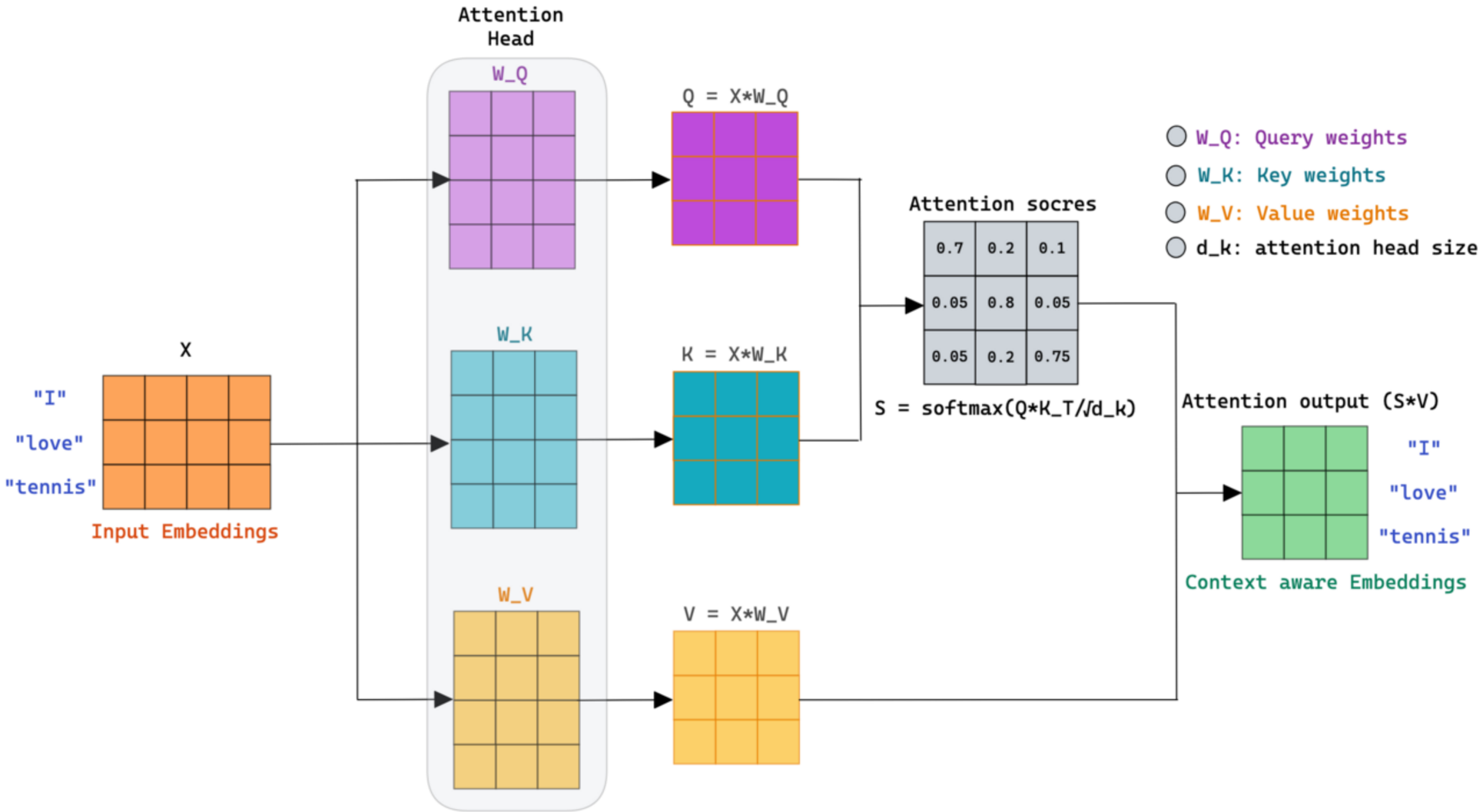


Multiple Output Approach

A single model predicts all the forecasting periods.

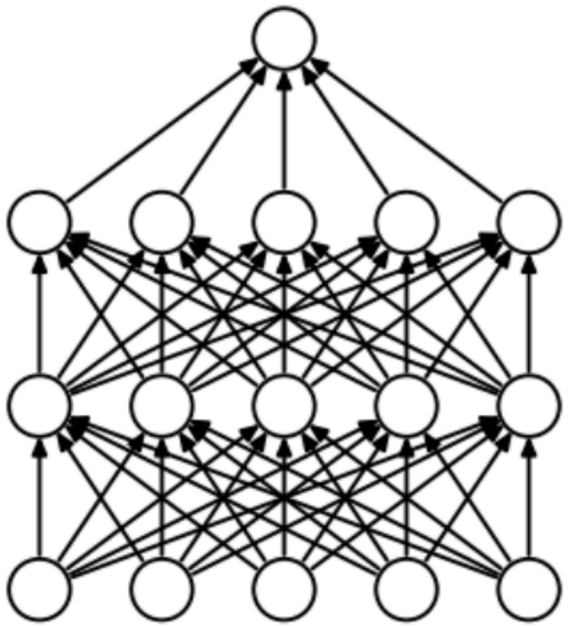


Attention/Transformer Layer

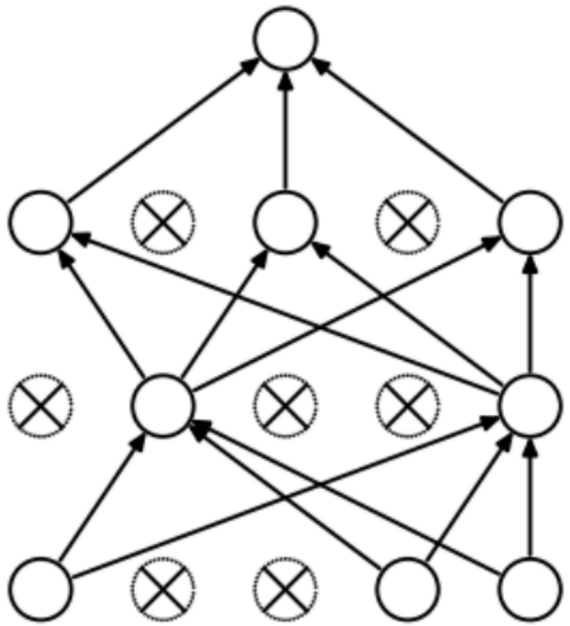


Dropout Layer

- Dropout rate를 바탕으로 랜덤하게 뉴런을 비활성화



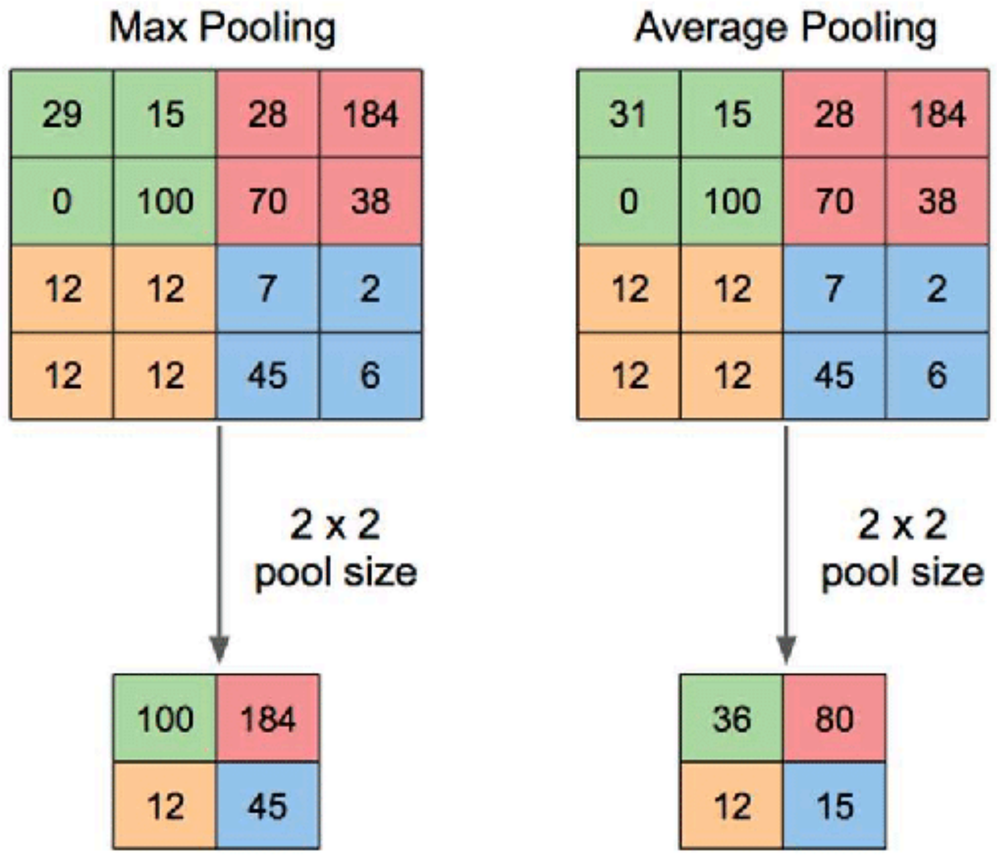
(a) Standard Neural Net



(b) After applying dropout.

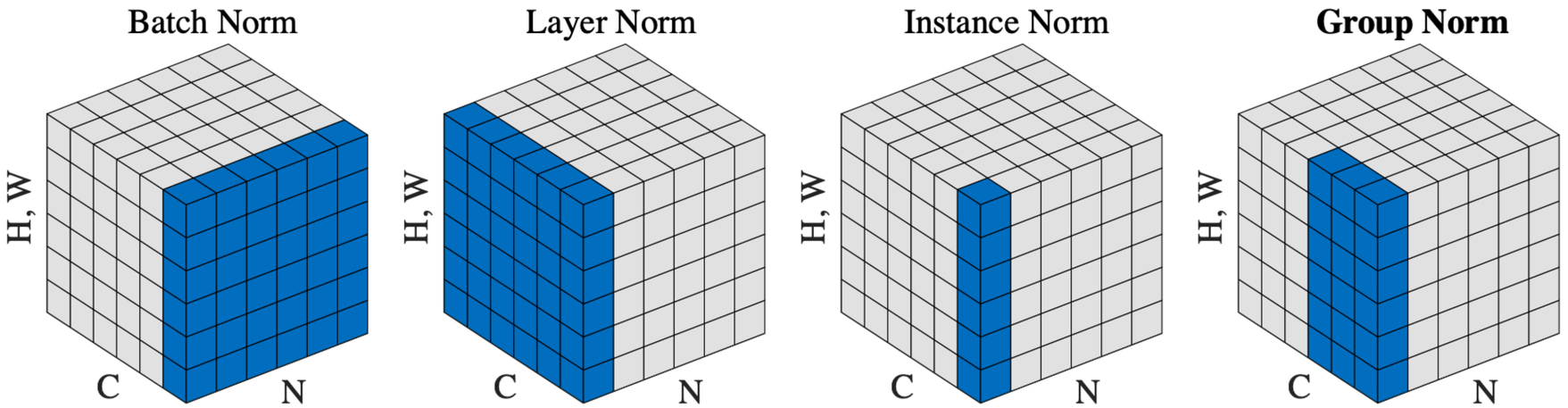
Pooling

- MaxPooling : 각 채널의 구간별로 최대값을 샘플링
- AvgPooling : 각 채널의 구간별로 평균값을 샘플링
- GlobalPooling : 채널별로 평균 또는 최대값을 샘플링



Normalization

- Batch / Layer / Instance / Group normalization



Padding

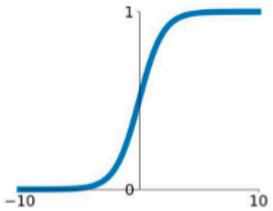
- 데이터의 양 끝단에 특정 값들을 채워주는 기능
- Convolution layer의 출력값 형태를 맞춰줄 수 있음.
  - Zero padding : 0 0 [1, 2, 3, 4] 0 0
  - Reflection padding : 3 2 [1, 2, 3, 4] 3 2
  - Replication padding : 1 1 [1, 2, 3, 4] 4 4
  - Constant padding : a a [1, 2, 3, 4] b b

Activation functions

- Sigmoid,
- Tanh
- ReLU
- Leaky ReLU
- ELU
- GELU

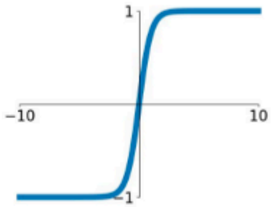
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



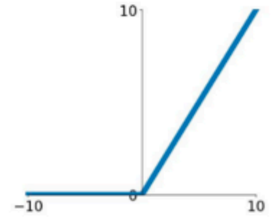
tanh

$$\tanh(x)$$



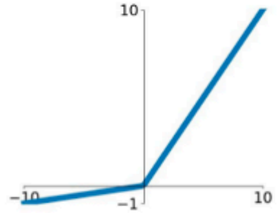
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

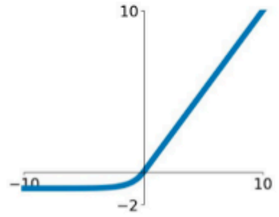


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

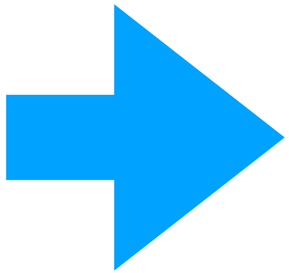
ELU

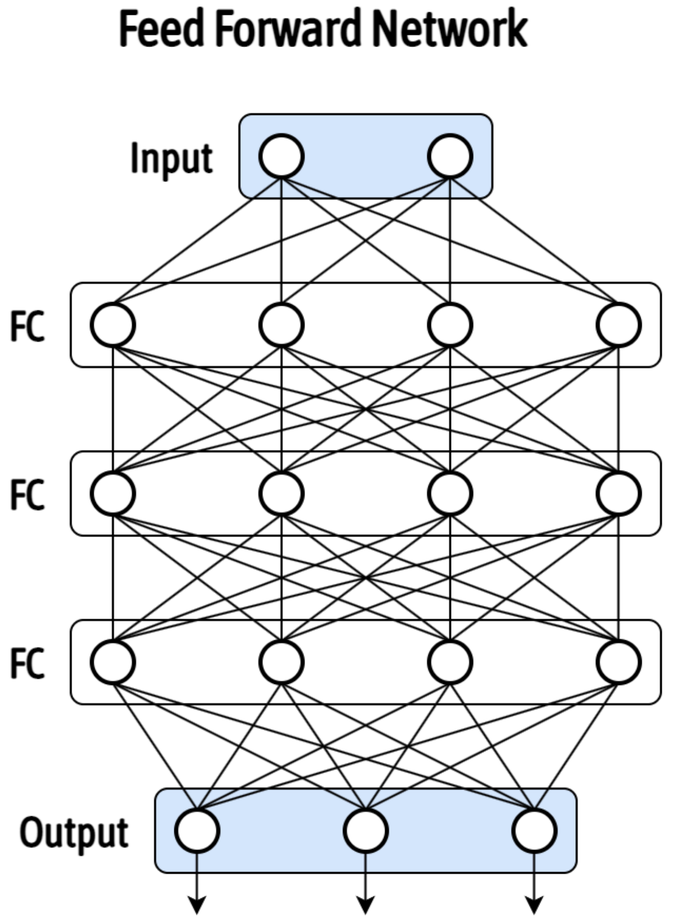
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



다양한 레이어의 조합

- Dense
- Recurrent
- Convolution
- Activation function
- Pooling
- Dropout
- Normalization
- Embedding





https://wikidocs.net/165167

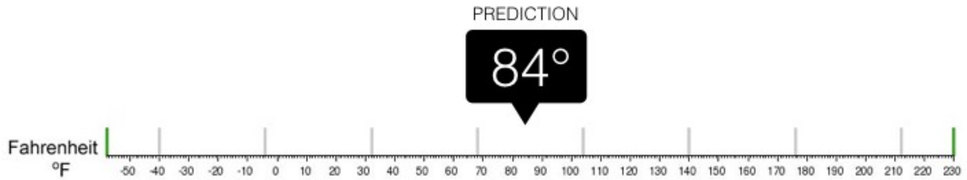
인공신경망 = 비선형 함수

- 어떠한 복잡한 모델을 구현하더라도 입력과 출력 관점에서 신경망은 비선형 함수임.
- 그리고 함수의 형태가 신경망의 학습가능한 웨이트에 의해 결정됨.
- 학습은 주어진 태스크를 더 잘하기 위한 방향으로 이루어짐.

기본적인 두가지 태스크

Regression

What is the temperature going to be tomorrow?

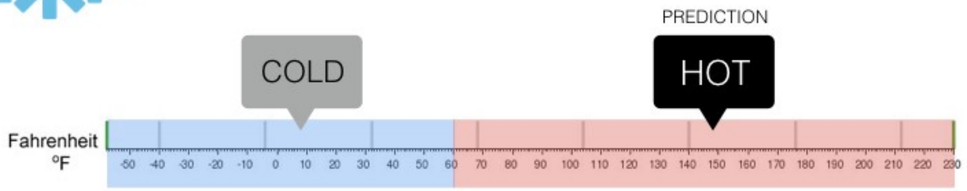


Minimize Squared errors

$$\frac{1}{N}\sum_{i}^{N}(y_i - \hat{y}_i)^2$$

Classification

Will it be Cold or Hot tomorrow?



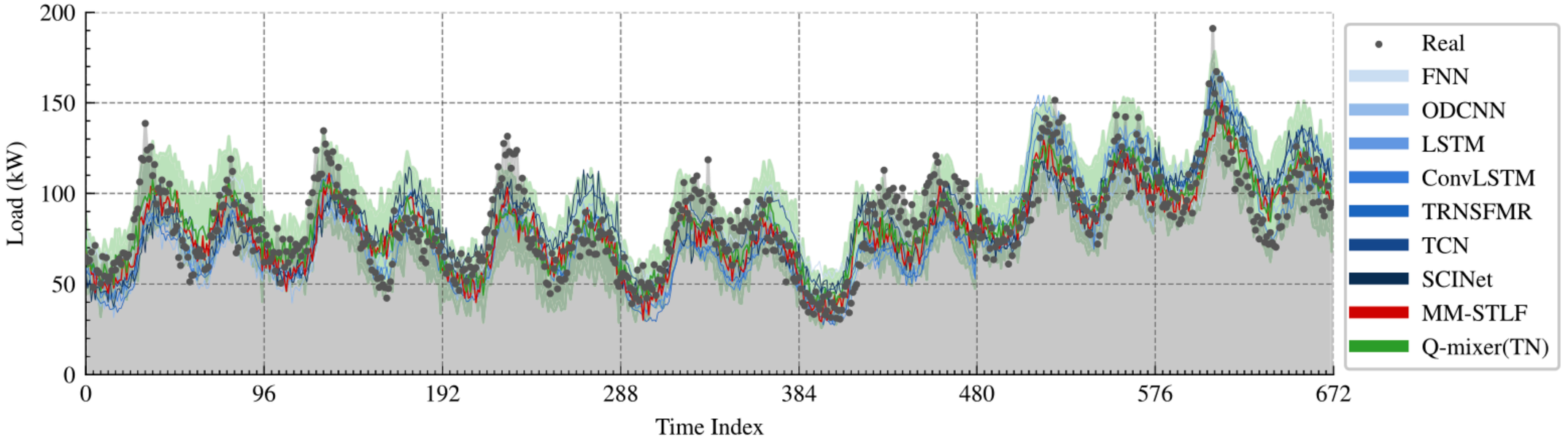
Minimize cross entropy

$$-\sum_{c\in C} y_c \log(\hat{y}_c)$$

불확실성 정량화

- Gaussian Negative Log Likelihood loss
- Pinball loss

$$L_\tau\left(x, x^\tau\right) = \begin{cases} \tau(x - x^\tau) & \text{if } x \geq x^\tau \\ (1-\tau)(x^\tau - x) & \text{if } x < x^\tau \end{cases}$$

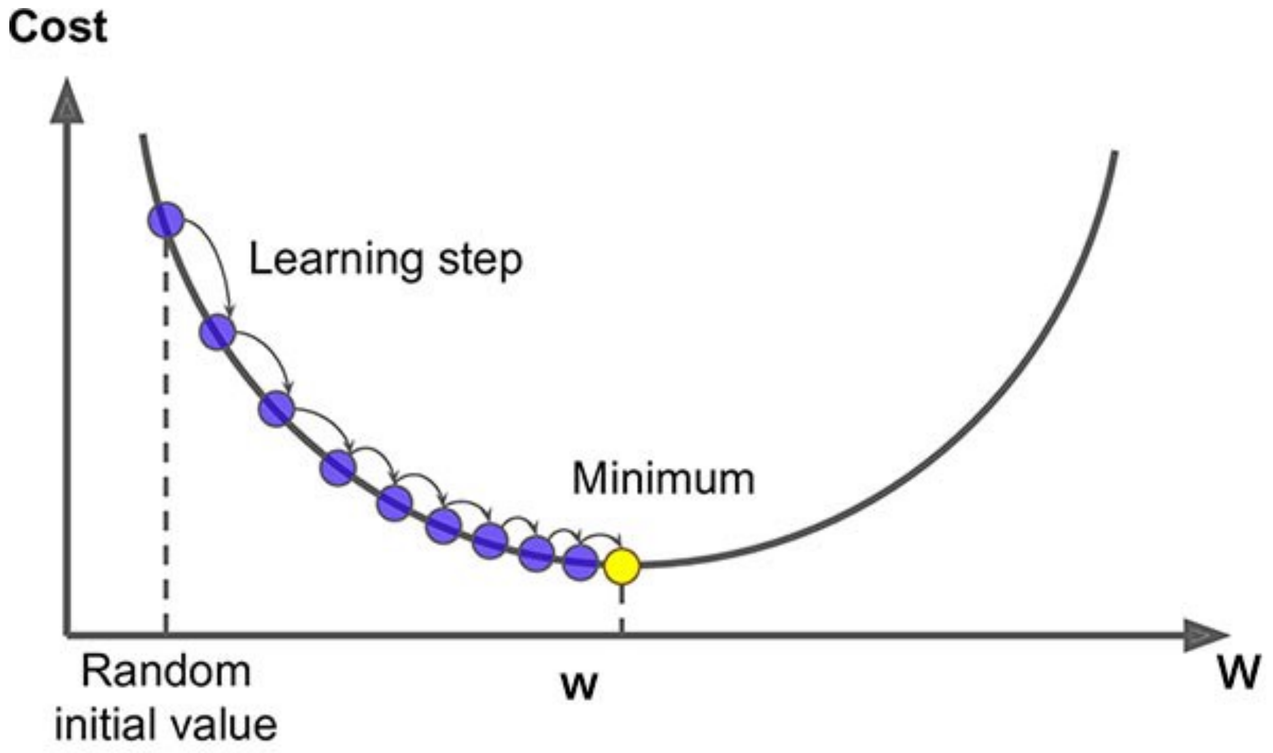


모델과 목표가 정해지면.

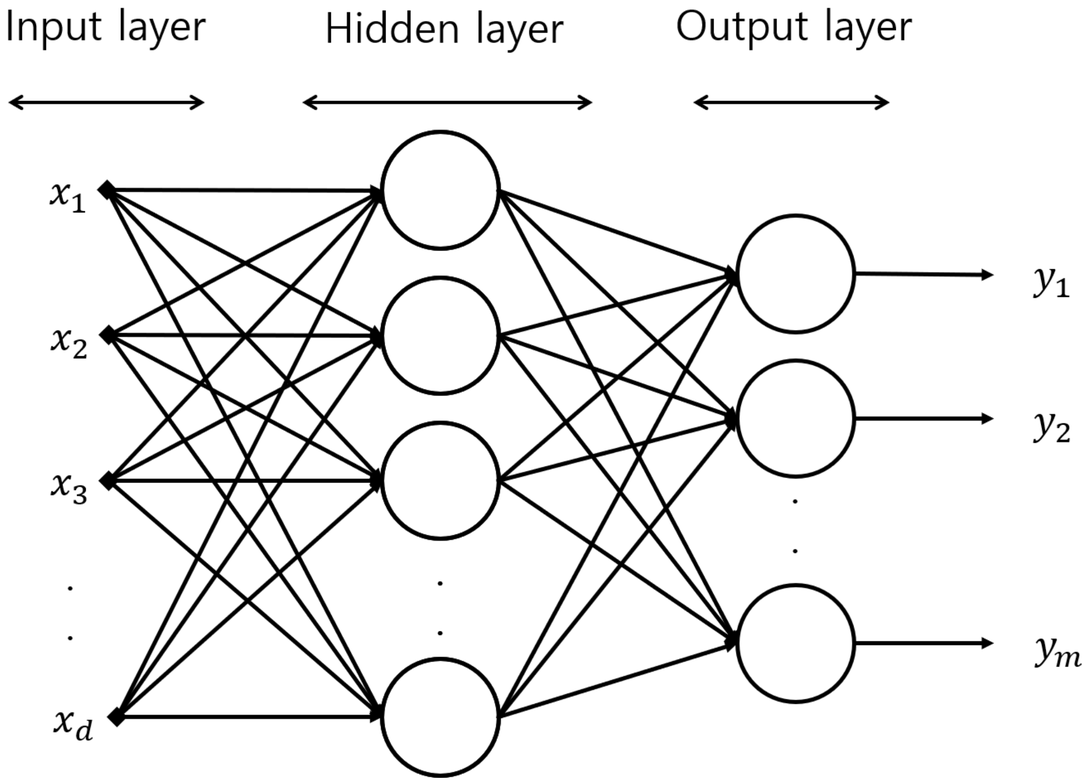
- 모델 : 인공신경망 모델
- 목표 : Loss, Cost function
- 학습을 진행해야 함. 즉 인공신경망의 weight parameter 학습.
  - Gradient descent method : general approach for iterative optimization
  - Error backpropagation : way to update parameters of inner layers

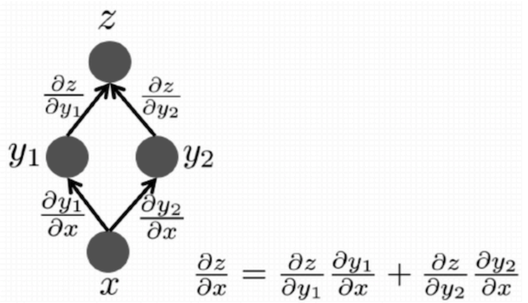
경사하강법

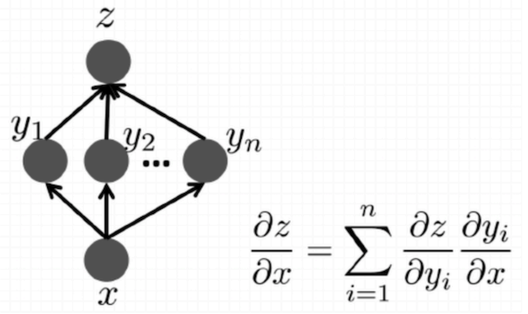
1. $w_0 \rightarrow w_1 \rightarrow w_2 \rightarrow \cdots \rightarrow w_t$
2. $w_{t+1} = w_t - \gamma \frac{\partial L}{\partial w}$



오류 역전파

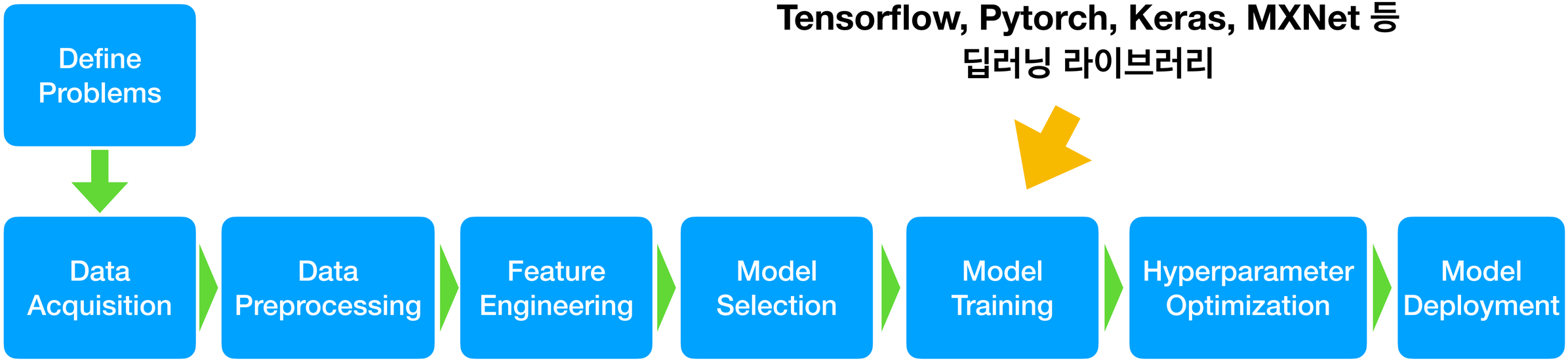






Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors." *nature* 323.6088 (1986): 533-536.

머신 러닝 모델 개발 프로세스



Keras를 통한 신경망 구성

```
# 모델 구성
model = Sequential()
model.add(Dense(512, activation = 'relu', input_shape = (28*28,)))
model.add(Dense(10, activation = 'softmax'))

# 모델 컴파일
model.compile(optimizer = 'rmsprop',
              loss = 'categorical_crossentropy',
              metrics = ['accuracy'])

# 모델 훈련
model.fit(train_images, train_labels, epoch = 5, batch_size = 128)

# 모델 평가
test_loss, test_acc = model.evaluate(test_images, test_labels)
```

Torch를 사용한 신경망 구성

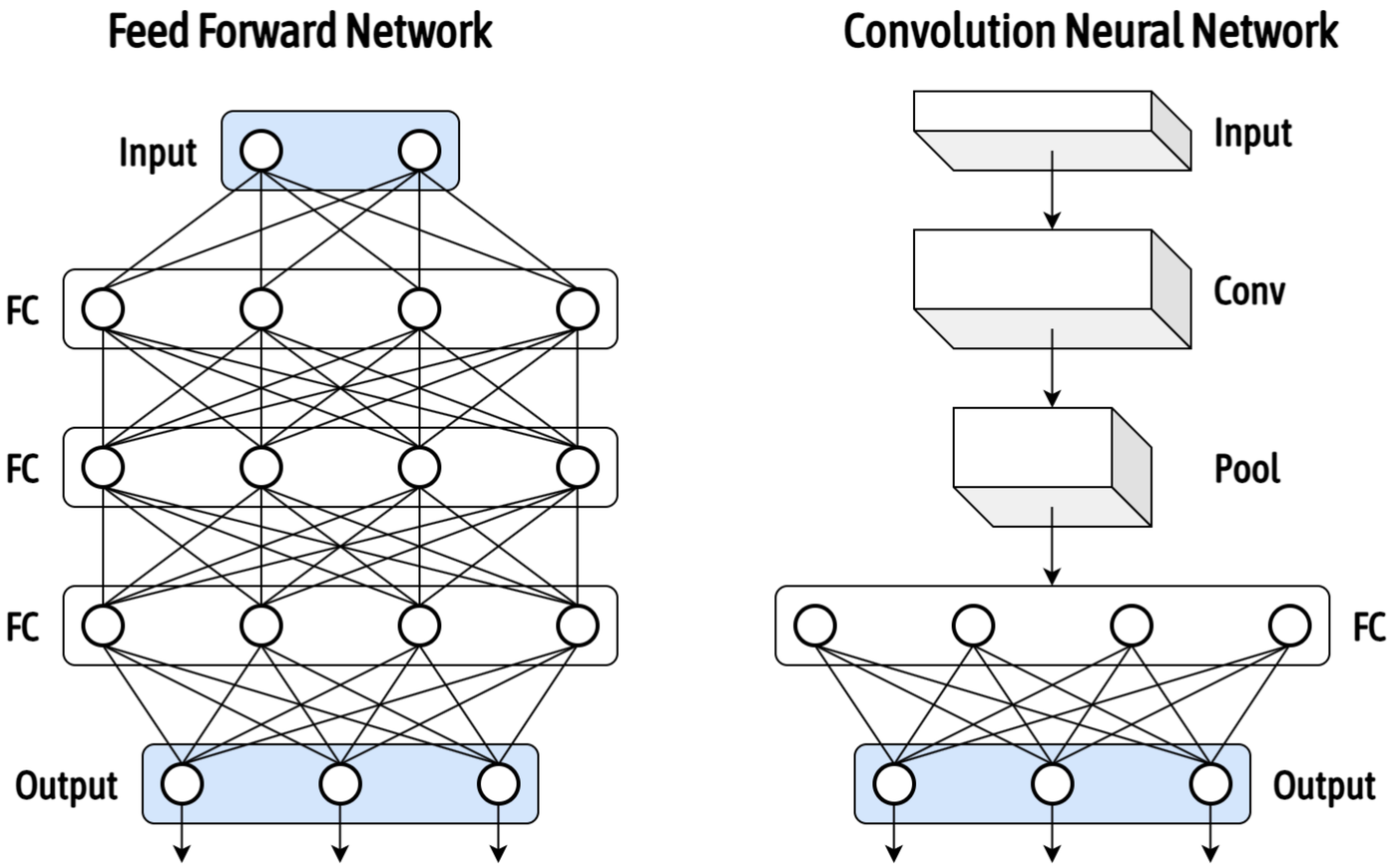
```
# 신경망 모델 구조 정의
model = nn.Sequential(
    nn.Linear(2, 3),  # 입력층 -> 은닉층 (입력 2차원, 은닉층 노드 3개)
    nn.ReLU(),        # 활성화 함수로 ReLU 사용
    nn.Linear(3, 1),  # 은닉층 -> 출력층 (은닉층 노드 3개, 출력 1차원)
    nn.Sigmoid()      # 출력층에서 Sigmoid 활성화 함수 사용
)

# 손실 함수와 옵티마이저 설정
criterion = nn.BCELoss()  # Binary Cross Entropy Loss
optimizer = optim.SGD(model.parameters(), lr=0.01)  # Stochastic Gradient Descent 옵티마이저

# 훈련 루프
for epoch in range(1000):  # 1000번의 에폭 동안 훈련
    # 순전파 수행
    outputs = model(X)
    # 손실 계산
    loss = criterion(outputs, y)
    # 기울기 초기화
    optimizer.zero_grad()
    # 역전파 수행
    loss.backward()
    # 옵티마이저를 통해 가중치 갱신
    optimizer.step()
```

Dense layer를 사용했을때

```
tf.keras.layers.Dense(
    units,
    activation=None,
    use_bias=True,
    kernel_initializer="glorot_uniform",
    bias_initializer="zeros",
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    **kwargs
)
```

- Unit : neuron의 수 = 출력 차원
- Activation : 활성함수.
  e.g., ReLU, Sigmoid, SELU, Tanh, etc
- Initializer : weight 초기화 관련
- 레이어 수 등

간단한 형태의 신경망



https://wikidocs.net/165167

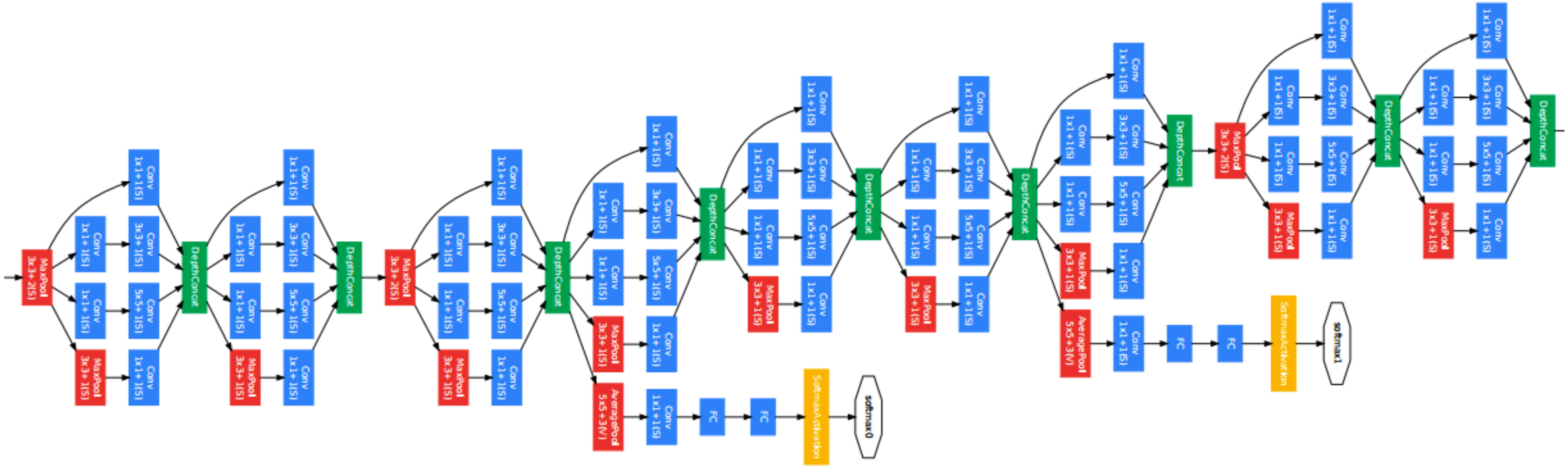
Keras를 통한 신경망 구성

```
# 모델 구성
model = Sequential()
model.add(Dense(512, activation = 'relu', input_shape = (28*28,)))
model.add(Dense(10, activation = 'softmax'))

# 모델 컴파일
model.compile(optimizer = 'rmsprop',
              loss = 'categorical_crossentropy',
              metrics = ['accuracy'])

# 모델 훈련
model.fit(train_images, train_labels, epoch = 5, batch_size = 128)

# 모델 평가
test_loss, test_acc = model.evaluate(test_images, test_labels)
```

CNN 예시

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 3)))
# 32개의 3x3 필터를 가진 컨볼루션 레이어
model.add(MaxPooling2D((2, 2)))
# 2x2 크기의 맥스 풀링 레이어
model.add(Conv2D(64, (3, 3), activation='relu'))
# 64개의 3x3 필터를 가진 컨볼루션 레이어
model.add(MaxPooling2D((2, 2)))
# 2x2 크기의 맥스 풀링 레이어
model.add(Conv2D(128, (3, 3), activation='relu'))
# 128개의 3x3 필터를 가진 컨볼루션 레이어
model.add(MaxPooling2D((2, 2)))
# 2x2 크기의 맥스 풀링 레이어
model.add(Flatten())
# 1차원으로 평탄화
model.add(Dense(128, activation='relu'))
# 128개의 뉴런을 가진 fully connected layer
model.add(Dense(10, activation='softmax'))
# 10개의 클래스에 대한 softmax 출력
```

보다 복잡한 신경망



Google Inception architecture

Neural Architecture Search

- 인공신경망의 구조 설계 과정을 자동화하여 최적의 구조를 찾는 방법.

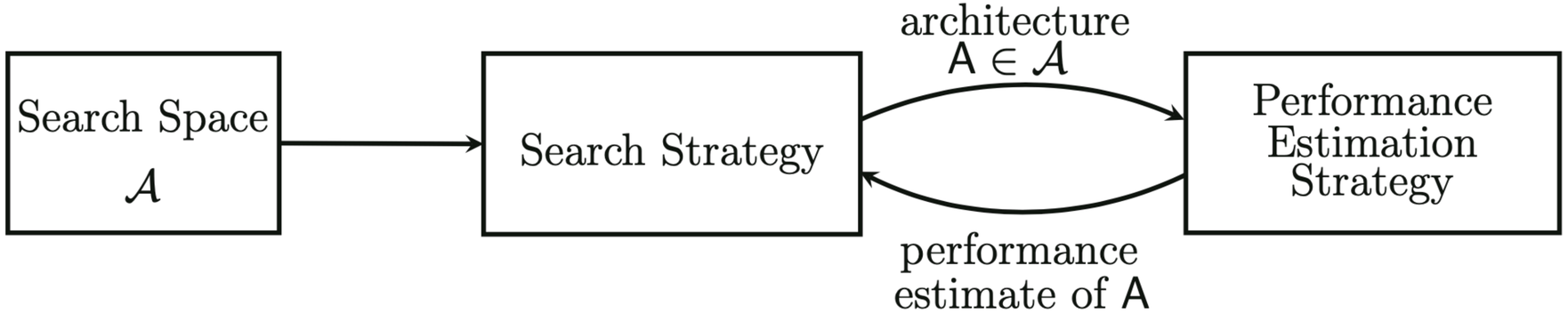
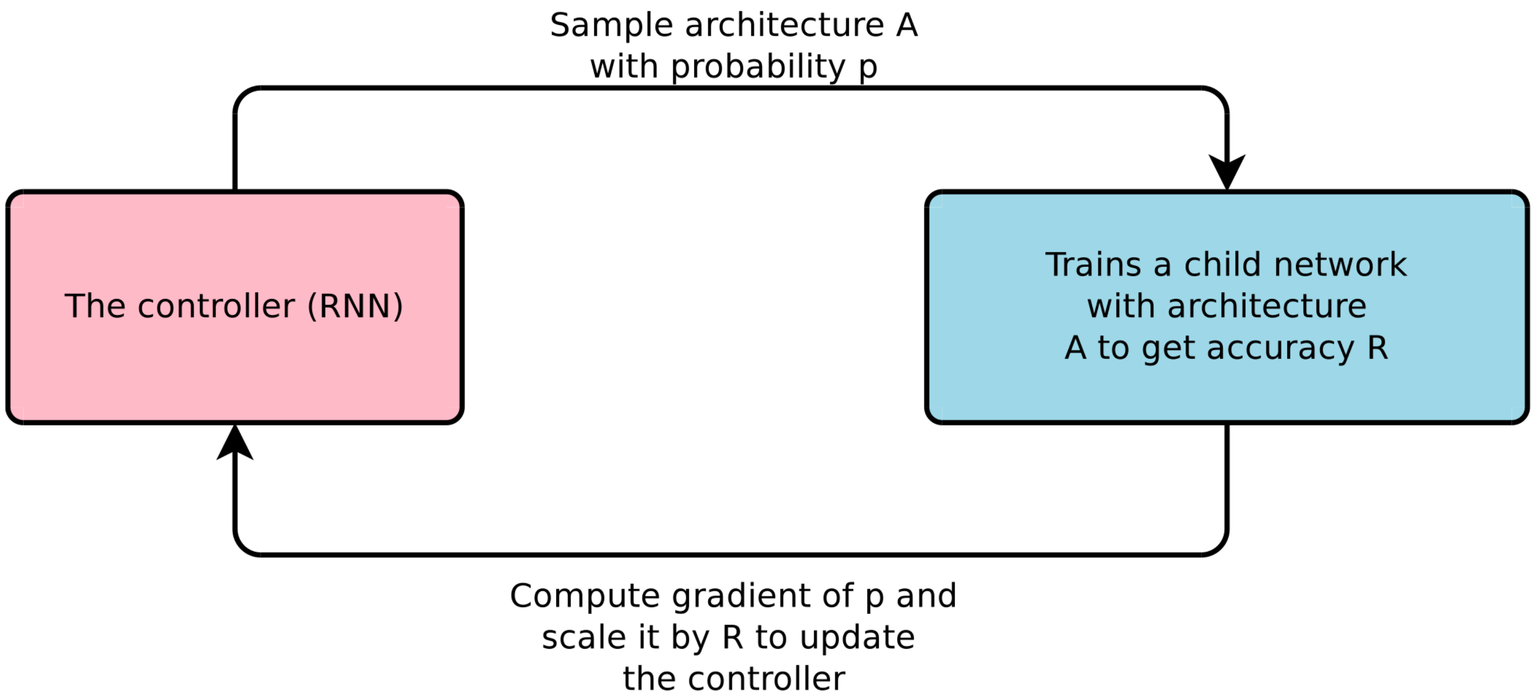


Fig. 3.1 Abstract illustration of Neural Architecture Search methods. A search strategy selects an architecture A from a predefined search space $\mathcal{A}$. The architecture is passed to a performance estimation strategy, which returns the estimated performance of A to the search strategy

Search Strategy

- Random search
- Bayesian optimization: 베이지안 최적화를 활용하여 네트워크 구조 탐색
- Reinforcement learning: 에이전트의 액션 = 네트워크 구조 선택, 보상 = 모델 성능
- Evolutionary methods: 유전알고리즘을 바탕으로 모델 풀에서 교차/변이 수행
- Gradient-based methods: Discrete search space에 대한 relaxation을 통해 미분 가능한 구조로 만들어 Gradient descent를 통해 최적 구조 탐색

Neural Architecture Search with Reinforcement Learning



Neural Architecture Search with Reinforcement Learning

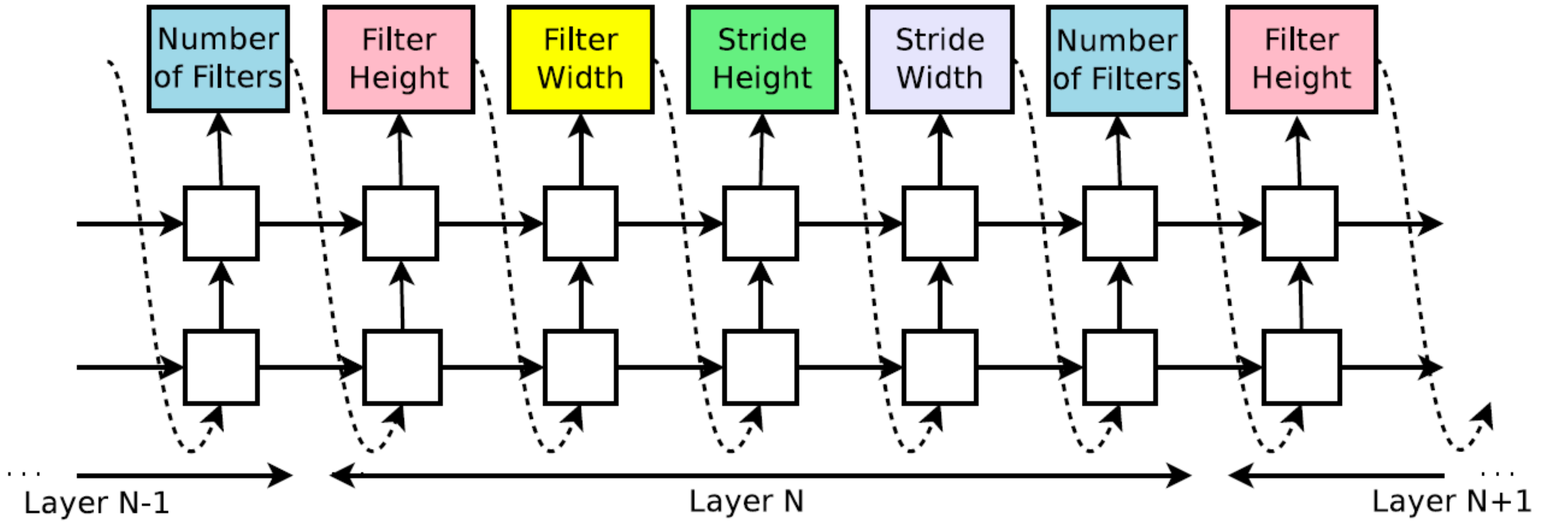


Figure 2: How our controller recurrent neural network samples a simple convolutional network. It predicts filter height, filter width, stride height, stride width, and number of filters for one layer and repeats. Every prediction is carried out by a softmax classifier and then fed into the next time step as input.

NAS with Evolutionary Computation

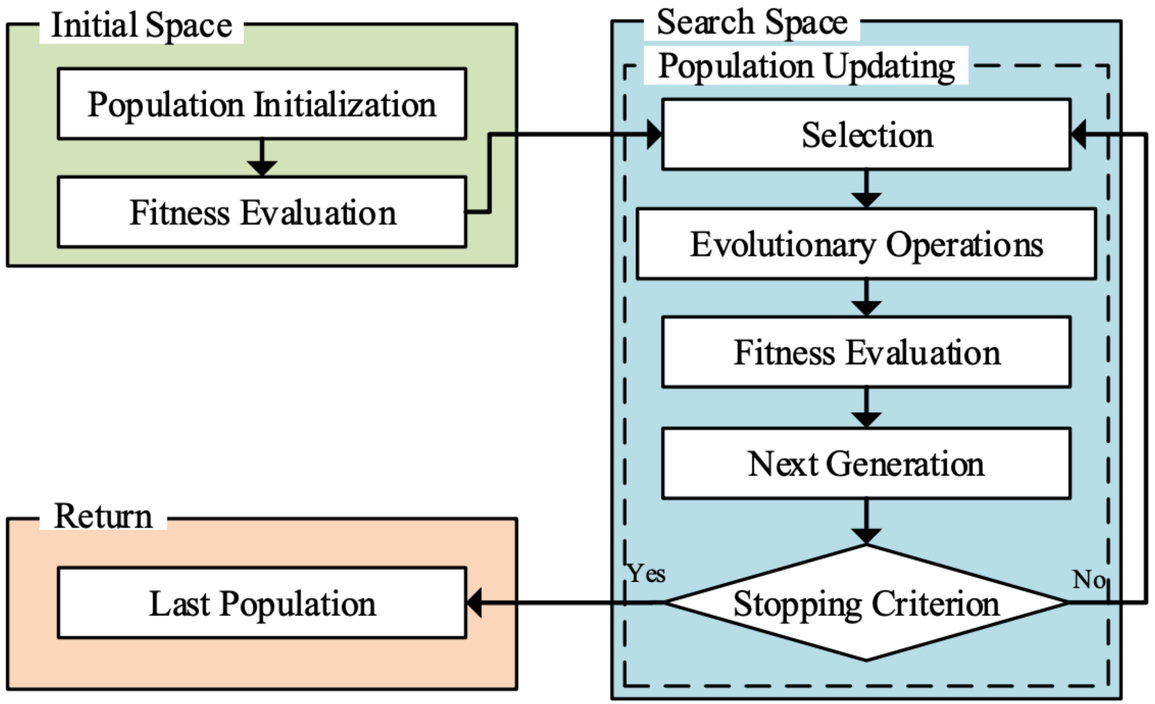


Fig. 2. The flowchart of a common ENAS algorithm.

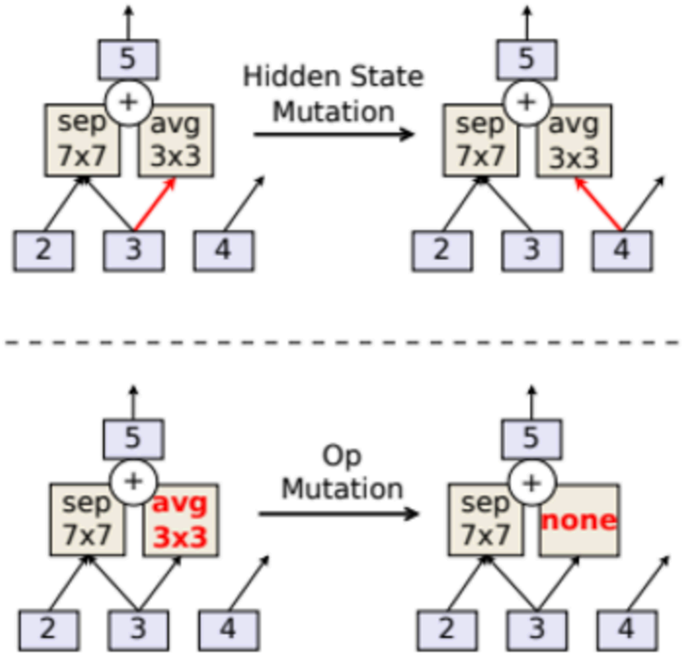


Figure 2: Illustration of the two mutation types.

AutoKeras

- Keras 기반 AutoML 시스템
- Bayesian optimization 기반 NAS 수행
- 다양한 태스크에 대한 간단한 인터페이스 제공
  - Image Classification/Regression
  - Text Classification/Regression
  - Structured Data Classification/Regression

In AutoKeras

```
# Initialize the structured data classifier.
clf = ak.StructuredDataClassifier(
    overwrite=True, max_trials=3
)  # It tries 3 different models.
# Feed the structured data classifier with training data.
clf.fit(
    # The path to the train.csv file.
    train_file_path,
    # The name of the label column.
    "survived",
    epochs=10,
)
# Predict with the best model.
predicted_y = clf.predict(test_file_path)
# Evaluate the best model with testing data.
print(clf.evaluate(test_file_path, "survived"))
```

감사합니다.