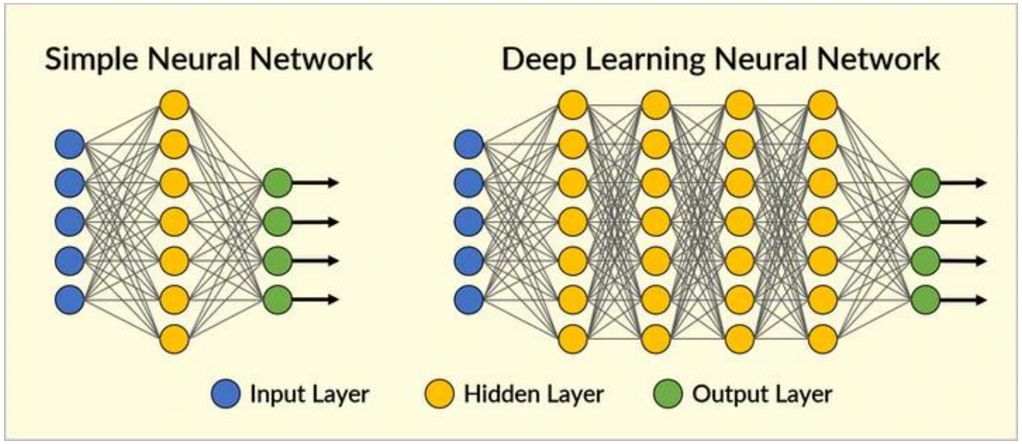
Introduction

Layers

개요

상상은 현실이 된다.

- Model을 구성하는 층
  - 하나 이상의 tensor을 입력 받아 하나 이상의 tensor를 출력 한다.



출처: https://gwoolab.tistory.com/46

Graph에 비해 단순화된 모형

- Layer
  - y = f(x) : 함수
    - 데이터 변환
      - keras.layers.BatchNormalization : input data를 정규화. 학습 속도가 빨라짐
      - keras.layers.Conv2D : Convolution
      - keras.layers.MaxPooling2D : Max Pooling
    - 데이터 가공
      - keras.layers.Dropout : 샘플링 (일부 데이터 삭제)
      - keras.layers.Flatten : 다 차원 데이터를 1차원으로 가공

Layers

작성중

Layer 종류 : Dense, Conv, Pooling, RNN, LSTM

- Layer
  - Dense : 밀집망

  - BatchNormalization : Input data를 정규화. 학습 속도가 빨라짐
  - Normalization
  - LayerNormalization
  - UnitNormalization

  - Dropout : 샘플링 (일부 데이터 삭제)
  - AlphaDropout
  - SpatialDropout1D
  - SpatialDropout2D
  - SpatialDropout3D
  - GaussianDropout

  - UpSampling1D
  - UpSampling2D
  - UpSampling3D

  - Flatten : 평탄화. 다 차원 데이터를 1차원으로 가공

- Dense : 1차원 벡터 처리

Layers

작성중

Layer 종류 : Dense, Conv, Pooling, RNN, LSTM

- Layer
  - Conv1D : Convolution 1D
  - Conv2D : Convolution 2D
  - Conv3D : Convolution 3D
  - Conv1DTranspose
  - Conv2DTranspose
  - Conv3DTranspose

  - DepthwiseConv1D
  - DepthwiseConv2D

  - SeparableConv1D
  - SeparableConv2D

  - ZeroPadding1D
  - ZeroPadding2D
  - ZeroPadding3D

- Conv : 공간 정보 추가

Layers

작성중

Layer 종류 : Dense, Conv, Pooling, RNN, LSTM

- Layer
  - AveragePooling1D : 평균 Pooling 1D
  - AveragePooling2D : 평균 Pooling 2D
  - AveragePooling3D : 평균 Pooling 3D
  - MaxPooling1D : 최대 Pooling 1D
  - MaxPooling2D : 최대 Pooling 2D
  - MaxPooling3D : 최대 Pooling 3D

  - GlobalAveragePooling1D
  - GlobalAveragePooling2D
  - GlobalAveragePooling3D
  - GlobalMaxPooling1D
  - GlobalMaxPooling2D
  - GlobalMaxPooling3D

- Pooling : 축소

Layers

작성중

Layer 종류 : Dense, Conv, Pooling, RNN, LSTM

- Layer
  - RNN
  - AbstractRNNCell
  - SimpleRNN
  - SimpleRNNCell
  - StackedRNNCells

  - LSTM
  - LSTMCell
  - ConvLSTM1D
  - ConvLSTM2D
  - ConvLSTM3D

- RNN : 과거 정보 반영
  - 시계열

- LSTM : RNN의 개선된 종류

Layers

작성중

Layer 종류 : Dense, Conv, Pooling, RNN, LSTM

- Layer
  - RandomBrightness
  - RandomContrast
  - RandomCrop
  - RandomFlip
  - RandomHeight
  - RandomRotation
  - RandomTranslation
  - RandomWidth
  - RandomZoom

- Random : 생성기

Layers

작성중

Layer 종류 : Dense, Conv, Pooling, RNN, LSTM

- Layer
  - Add
  - add
  - Average
  - average
  - Concatenate
  - concatenate
  - Dot
  - dot
  - Maximum
  - maximum
  - Minimum
  - minimum
  - Multiply
  - multiply
  - Subtract
  - subtract

Layers

작성중

Layer 종류 : Dense, Conv, Pooling, RNN, LSTM

- Layer
  - LeakyReLU
  - Reshape
  - Activation
  - EinsumDense
  - Embedding
  - Lambda
  - Masking
  - LocallyConnected1D
  - LocallyConnected2D

  - CategoryEncoding
  - Discretization
  - Hashing
  - CenterCrop

Layers

작성중

Layer 종류 : Dense, Conv, Pooling, RNN, LSTM

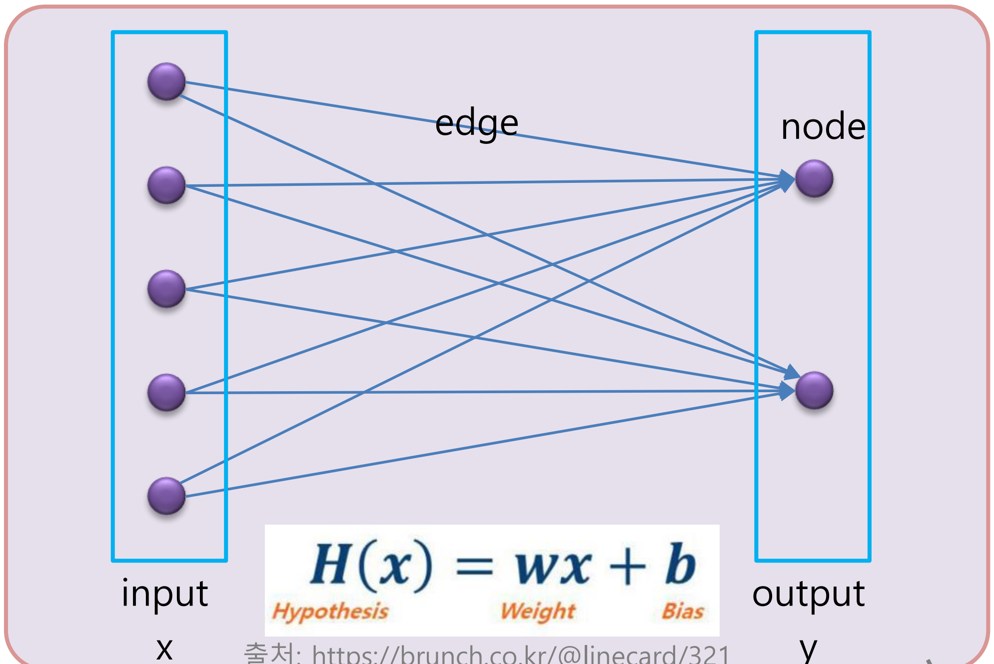
- Layer
  - Rescaling
  - Resizing
  - IntegerLookup
  - StringLookup
  - TextVectorization
  - ActivityRegularization
  - GaussianNoise
  - Cropping1D
  - Cropping2D
  - Cropping3D
  - Permute
  - RepeatVector
  - Reshape
  - Wrapper
  - Bidirectional
  - GRU
  - GRUCell
  - TimeDistributed

Layers

Layer 종류 : keras.layers.Dense (밀집망)

- NN (Neural Network)

Activation Function : f()

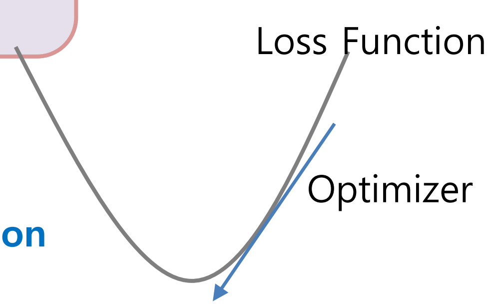


node, edge
Back Propagation

Optimizer

Loss Function
훈련 평가

Metrics
모델 평가



- Parameter 갯수
  - edge, node
  - x * y + y
- 작업량
  - Optimizer
  - Loss Function
  - Activation Function
  - Predict
- Model 비교 기준
  - 속도
  - 가중치 크기
- x = f′(y) : 전치 함수
  - 역행렬
- 장점
  - Loss Function을 사용하여 학습이 가능함
- 단점
  - 이미지 분류시 위상 정보를 활용하지 못함

Layers

Layer 종류 : keras.layers.Dense (밀집망)

- Activation Functions (활성화 함수)
  - https://reniew.github.io/12/

  - sigmoid : Sigmoid, [0, 1] : 이진 분류

  - tanh : TanH, [-1, 1] ← Sigmoid 개선 : 이진 분류
  - softmax : Softmax, Output의 총합은 1 (확률) : 다중 분류

  - relu : ReLU (Rectified Linear Unit), [0, x] : 연산이 빠름
  - LeakyReLU : [0.01x, x] ← ReLU 개선
  - PReLU : [ax, x] ← Leakly ReLU 개선
  - elu : ELU (Exponential Linear Unit), [a(e^x -1), x] ← TanH + ReLU
  - linear : Linear, 입력값을 그대로 출력

Layers

작성중

Layer 종류 : keras.layers.Dense (밀집망)

- Activation Functions (활성화 함수)
  - exponential
  - gelu
  - hard_sigmoid
  - selu
  - softplus
  - softsign
  - swish
  - ThresholderdReLU

Layers

작성중

Layer 종류 : keras.layers.Dense (밀집망)

- Loss Functions (손실 함수)
  - binary_crossentropy
  - binary_focal_crossentropy
  - categorical_crossentropy : 분류시 사용
  - sparse_categorical_crossentropy

  - mean_squared_error : MSE : 예측시 사용
  - mean_squared_logarithmic_error : MSLE
  - mean_absolute_error : MAE
  - mean_absolute_percentage_error : MAPE

  - categorical_hinge
  - cosine_similarity
  - hinge
  - huber
  - kl_divergence
  - log_cosh
  - log_cosh as logcosh
  - poisson
  - squared_hinge

Layers

작성중

Layer 종류 : keras.layers.Dense (밀집망)

- Optimizer (최적화)
  - experimental
  - legacy
  - schedules

  - adadelta : Adadelta
  - adagrad : Adagrad : 학습률을 조정하여 학습
  - adam : Adam : 예측시 사용
  - adamax : Adamax
  - ftrl : Ftrl
  - gradient_descent : SGD (Stochastic Gradient Descent, 확률적 경사 하강법)
  - nadam : Nadam
  - rmsprop : RMSprop

Layers

Layer 종류 : keras.layers.Dense (밀집망)

- Matric (척도)
  - https://ek-koh.github.io/data analysis/evaluation/

  - Accuracy : 정확도 (예측과 실제값이 같은 비율)
  - BinaryAccuracy
  - CategoricalAccuracy
  - binary_accuracy
  - categorical_accuracy
  - sparse_categorical_accuracy
  - sparse_top_k_categorical_accuracy
  - top_k_categorical_accuracy
  - SparseCategoricalAccuracy
  - SparseTopKCategoricalAccuracy
  - TopKCategoricalAccuracy

  - Precision : 정밀도 (예측이 a인 경우, 예측과 실제값이 a이 비율)
  - PrecisionAtRecall

  - Recall : 재현율 (실제값이 a인 경우, 예측과 실제값이 a이 비율)
  - RecallAtPrecision

Layers

작성중

Layer 종류 : keras.layers.Dense (밀집망)

- Matric (척도)
  - Mean
  - MeanMetricWrapper
  - MeanTensor
  - Metric
  - Sum

  - AUC
  - BinaryCrossentropy
  - BinaryIoU
  - CategoricalCrossentropy
  - CategoricalHinge
  - CosineSimilarity
  - FalseNegatives
  - FalsePositives
  - Hinge
  - IoU
  - KLDivergence

작성중

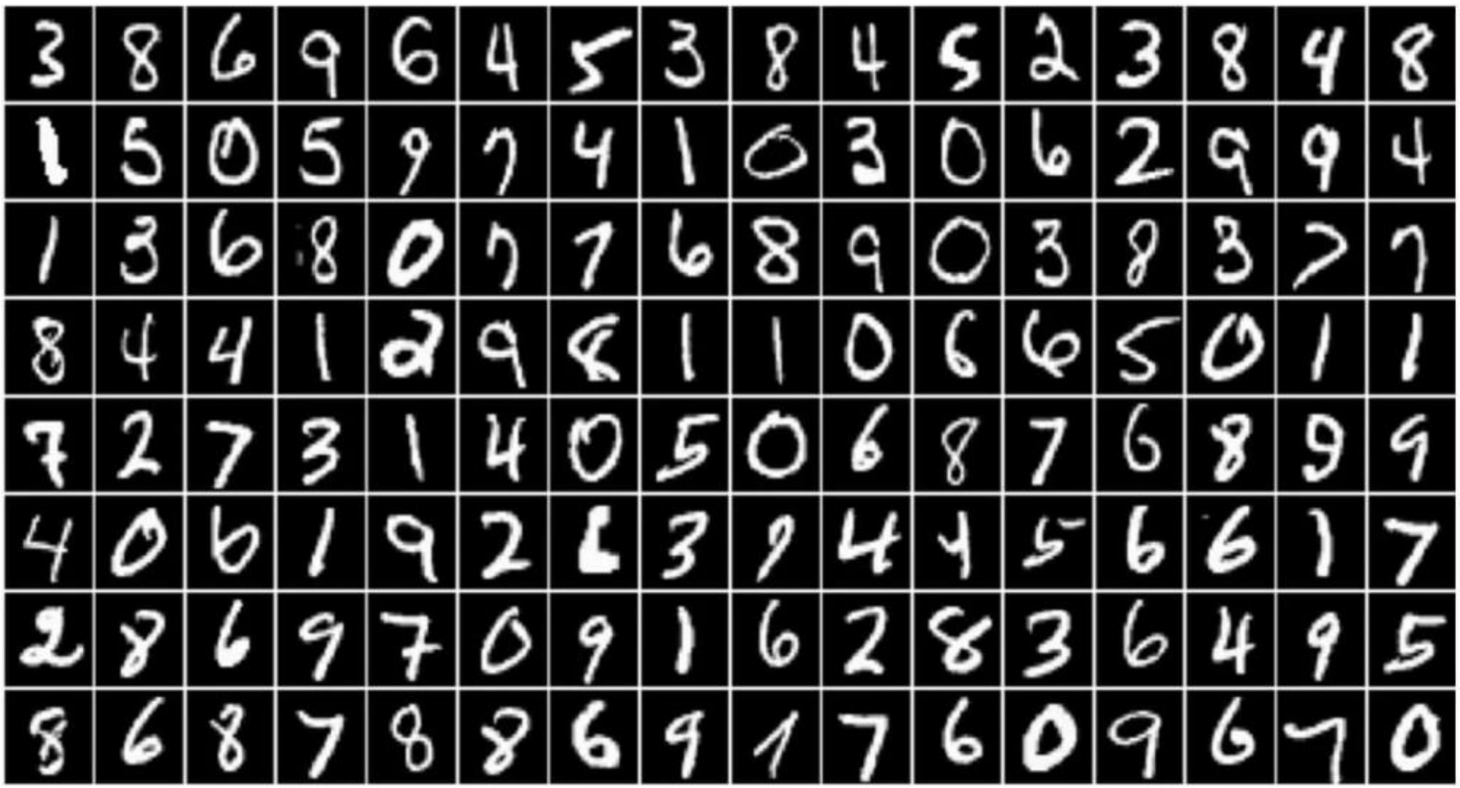
Layer 종류 : keras.layers.Dense (밀집망)

- Matric (척도)
  - LogCoshError
  - MeanAbsoluteError
  - MeanAbsolutePercentageError
  - MeanIoU
  - MeanRelativeError
  - MeanSquaredError
  - MeanSquaredLogarithmicError
  - OneHotIoU
  - OneHotMeanIoU
  - Poisson
  - RootMeanSquaredError
  - SensitivityAtSpecificity
  - SparseCategoricalCrossentropy
  - SpecificityAtSensitivity
  - SquaredHinge
  - TrueNegatives
  - TruePositives

Layers

Layer 종류 : keras.layers.Dense (밀집망)

- Input
  - 숫자 손글씨
  - 28 * 28



출처: https://learnopencv.com/implementing-mlp-tensorflow-keras/

Layers

Layer 종류 : keras.layers.Dense (밀집망)

- 데이터 가공
  - Nomalization
  - Layer에서 데이터 가공과 다른 점은 "**한번만 실행된다**"는 것 이다.

- 훈련 데이터 : train
  - 훈련용 : train : 80%
  - 검증용 : validation : 20%

- 평가 데이터 : test

- ephochs
  - 훈련 집합 횟수

- batch_size
  - 훈련 집합당 훈련 횟수

Layers

Layer 종류 : keras.layers.Dense (밀집망)

- Model

```
model = keras.models.Sequential()
#--- nb_classes = 28 * 28
model.add(keras.layers.Dense(self.nb_classes, input_shape=(reshaped,), activation='softmax'))

model.compile(
    optimizer=self.optimizer,                #--- Adam
    loss=self.loss_function,                 #--- categorical_crossentropy
    metrics=[ self.metrics ]                 #--- accuracy
)
```

Layers

Layer 종류 : keras.layers.Dense (밀집망)

- Output

Model: "sequential"

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 10)                7850           #--- (28 * 28) * 10 + 10

=================================================================
Total params: 7,850
Trainable params: 7,850
Non-trainable params: 0
```

Layers

Layer 종류 : keras.layers.Dense (밀집망)

- Output

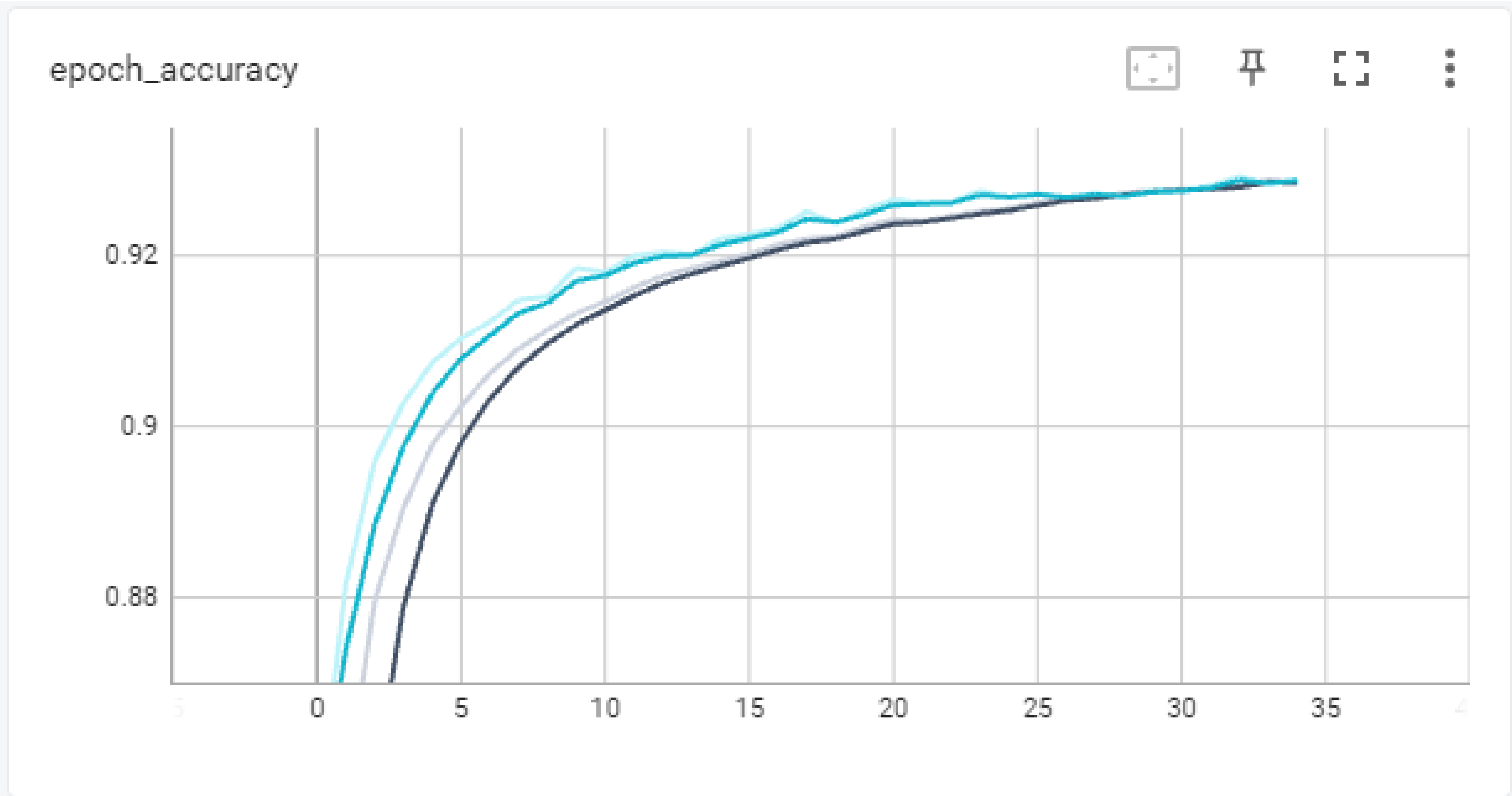
  Epoch 1/35
  94/94 [==============================] - 1s 4ms/step - loss: 1.2428 - accuracy: 0.6838 - val_loss: 0.7018 - val_accuracy: 0.8438
  Epoch 2/35
  94/94 [==============================] - 0s 3ms/step - loss: 0.6172 - accuracy: 0.8523 - val_loss: 0.4953 - val_accuracy: 0.8803
  ... 생략 ...
  Epoch 34/35
  94/94 [==============================] - 0s 3ms/step - loss: 0.2570 - accuracy: 0.9283 - val_loss: 0.2617 - val_accuracy: 0.9294
  Epoch 35/35
  94/94 [==============================] - 0s 3ms/step - loss: 0.2555 - accuracy: 0.9287 - val_loss: 0.2627 - val_accuracy: 0.9287

  313/313 [==============================] - 0s 1ms/step - loss: 0.2669 - accuracy: 0.9269
  Test accuracy: 0.9269000291824341

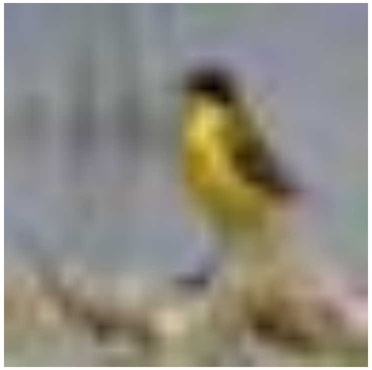
Layers

Layer 종류 : keras.layers.Dense (밀집망)

- Graph



Deep Learning 응용



60,000 이미지
- 100개 분류
- 분류당 600개 이미지

32 * 32 picels

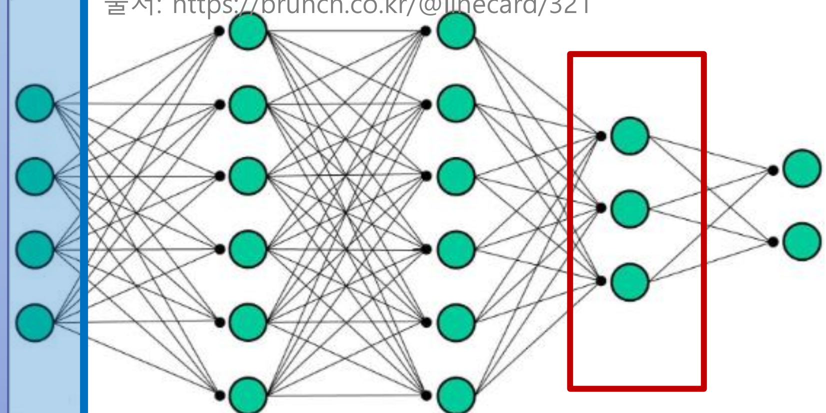
출처: https://www.cs.toronto.edu/~kriz/cifar.html

분류기

$$H(x) = wx + b$$

Hypothesis Weight Bias

출처: https://brunch.co.kr/@linecard/321



출처: https://www.freshworks.com/ko/freshdesk/kblogs/deep-learning/

1

생성기

이미지 생성

y = f(x)
  = softmax(H(x))
  = softmax(wx + b)

H(x) = wx + b

wx = H(x) - b
x = (H(x) - b) / w

x = H(x) / w - b / w
(y = wx + b 형태)

난수를 발생시켜 H(x)로 사용하면
이미지(x)를 생성할 수 있음

학습 기능 없음

생성기 → 분류기

난수
random

학습

이미지 생성

이미지

분류기

판별기로 사용

학습 X
사용

Generative AI (생성형 AI)

Deep Learning 응용

출처: https://www.cs.toronto.edu/~kriz/cifar.html

60,000 이미지
- 100개 분류
- 분류당 600개 이미지

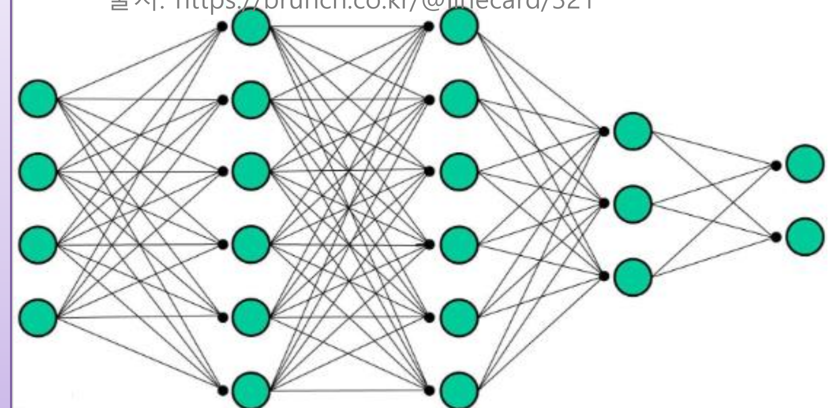
32 * 32 picels

분류기

$$H(x) = wx + b$$

Hypothesis Weight Bias

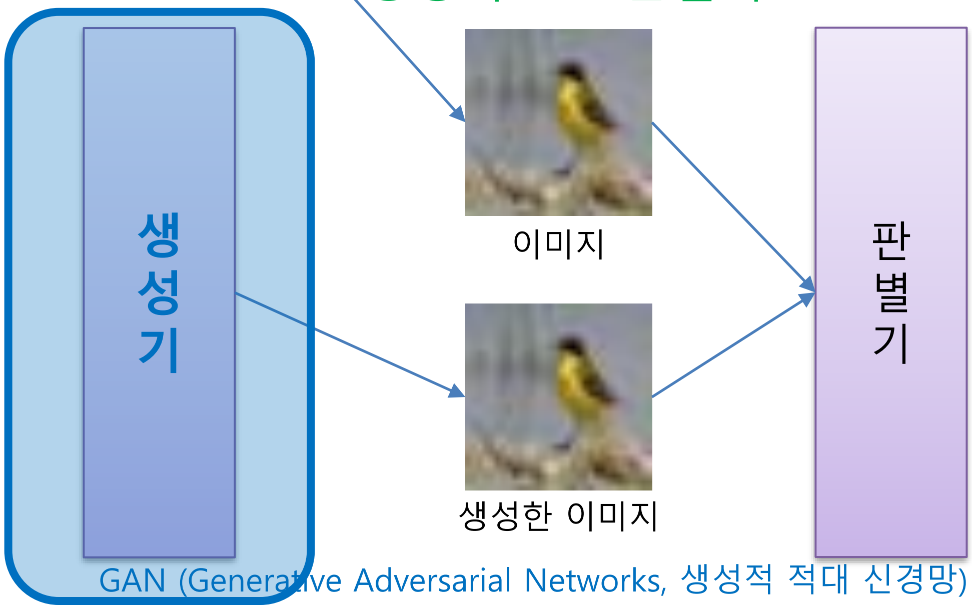
출처: https://brunch.co.kr/@linecard/321



출처: https://www.freshworks.com/ko/freshdesk/kblogs/deep-learning/

2

생성기 → 판별기

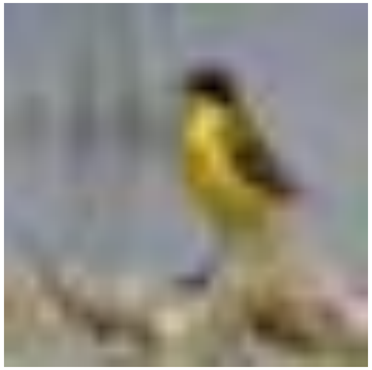


이미지를 생성하는 모델과
이미지를 판별하는 모델을
모두 학습 시킨다

x : 이미지 또는 생성한 이미지
y : 사진, 기계가 생성한 이미지

GAN (Generative Adversarial Networks, 생성적 적대 신경망)

Deep Learning 응용



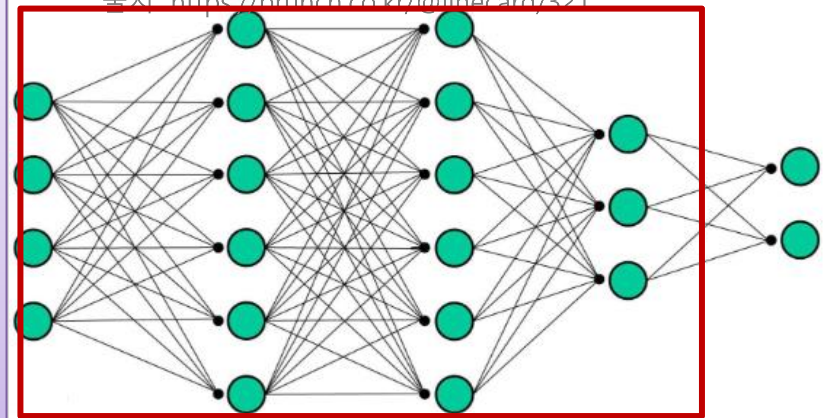
출처: https://www.cs.toronto.edu/~kriz/cifar.html

60,000 이미지
100개 분류
분류당 600개 이미지

32 * 32 picels

분류기

출처: https://brunch.co.kr/@linecard/321



출처: https://www.freshworks.com/ko/freshdesk/kblogs/deep-learning/

3

생성기

이미지 생성

y = softmax(H(x))
= softmax(wx + b)

H(x) = wx + b

wx = H(x) - b
x = (H(x) - b) / w

난수를 발생시켜 H(x)로 사용하면
이미지(x)를 생성할 수 있음

학습 기능 없음

Encoder → Decoder

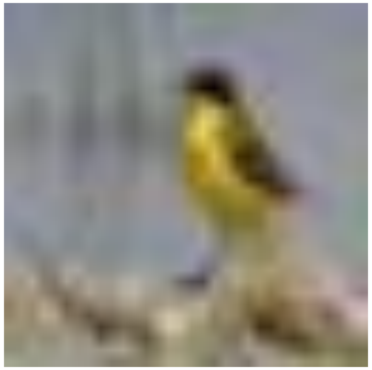
분류기 → 이미지 생성

원본 이미지와 생성된 이미지의 유사성으로 학습 가능

단, 학습의 의미가 없음

학습 기능 없음

AutoEncoder



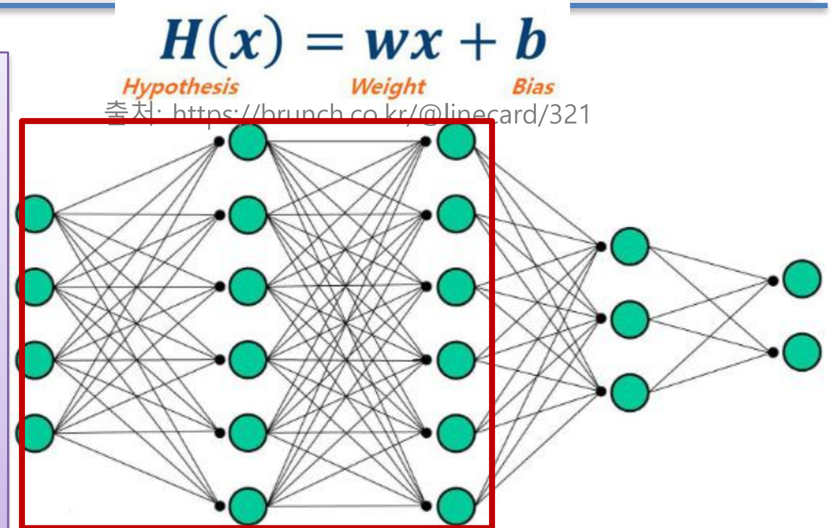
출처: https://www.cs.toronto.edu/~kriz/cifar.html

60,000 이미지
- 100개 분류
- 분류당 600개 이미지

32 * 32 picels

회귀

x, y, width, height



출처: https://www.freshworks.com/ko/freshdesk/kblogs/deep-learning/

분류와 지역화

분류

회귀

x, y, width, height

- 객체 탐지 (분류 + 회귀)
  - 동물, 동물의 위치

RNN

회귀

~~상태~~

다음 입력으로 사용

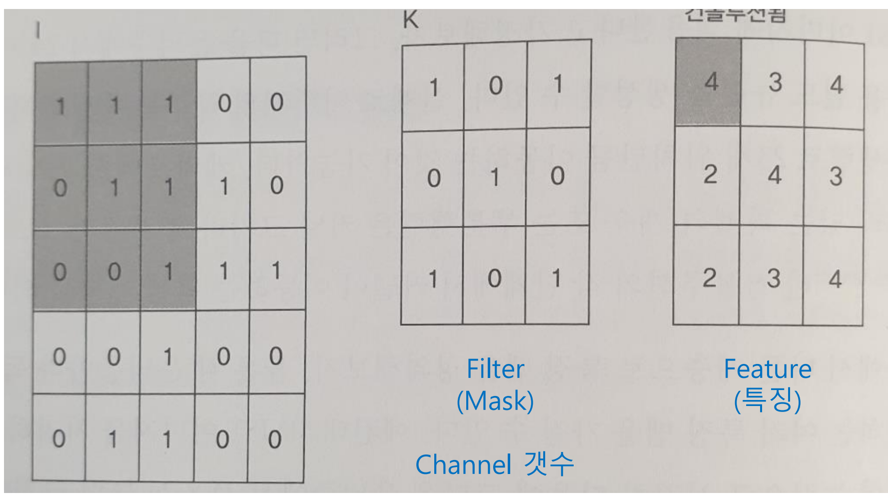
학습 기능 없음

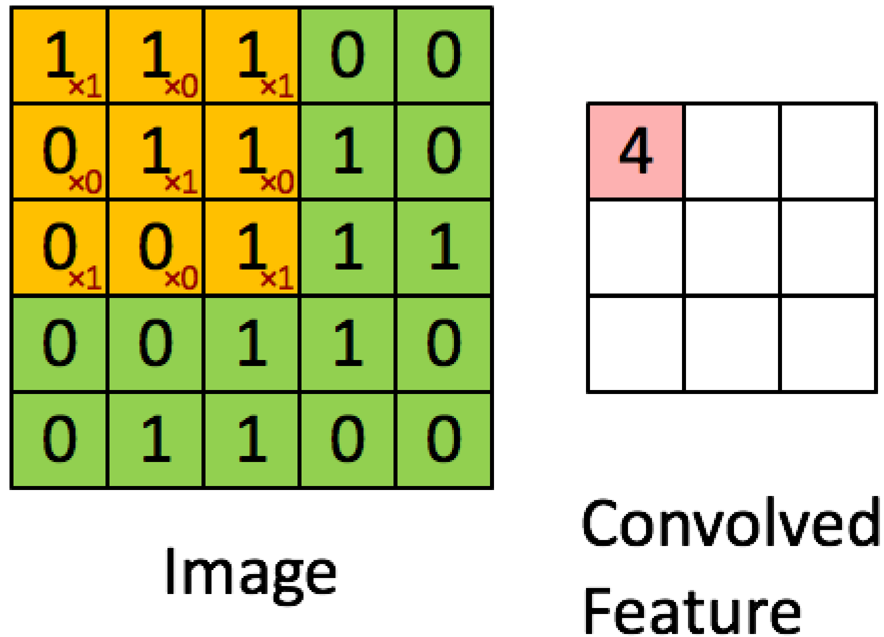
RNN (Recurrent Neural Network, 순환 신경망)

Layer 종류 : keras.layers.Conv2D

- Convolution : 관계 정보를 활용하는 방법
  - Dense : 선형 (1차원 배열)
  - Convolution : n차원 부분 행열
    - 행열에 Convolution을 적용하여 특징맵을 생성
    - 2차원 : 이미지, 오디오, 텍스트
    - 3차원 : 비디오

- Transpose Convolution
  - 전치 컨볼루션
  - 입력과 출력을 반전



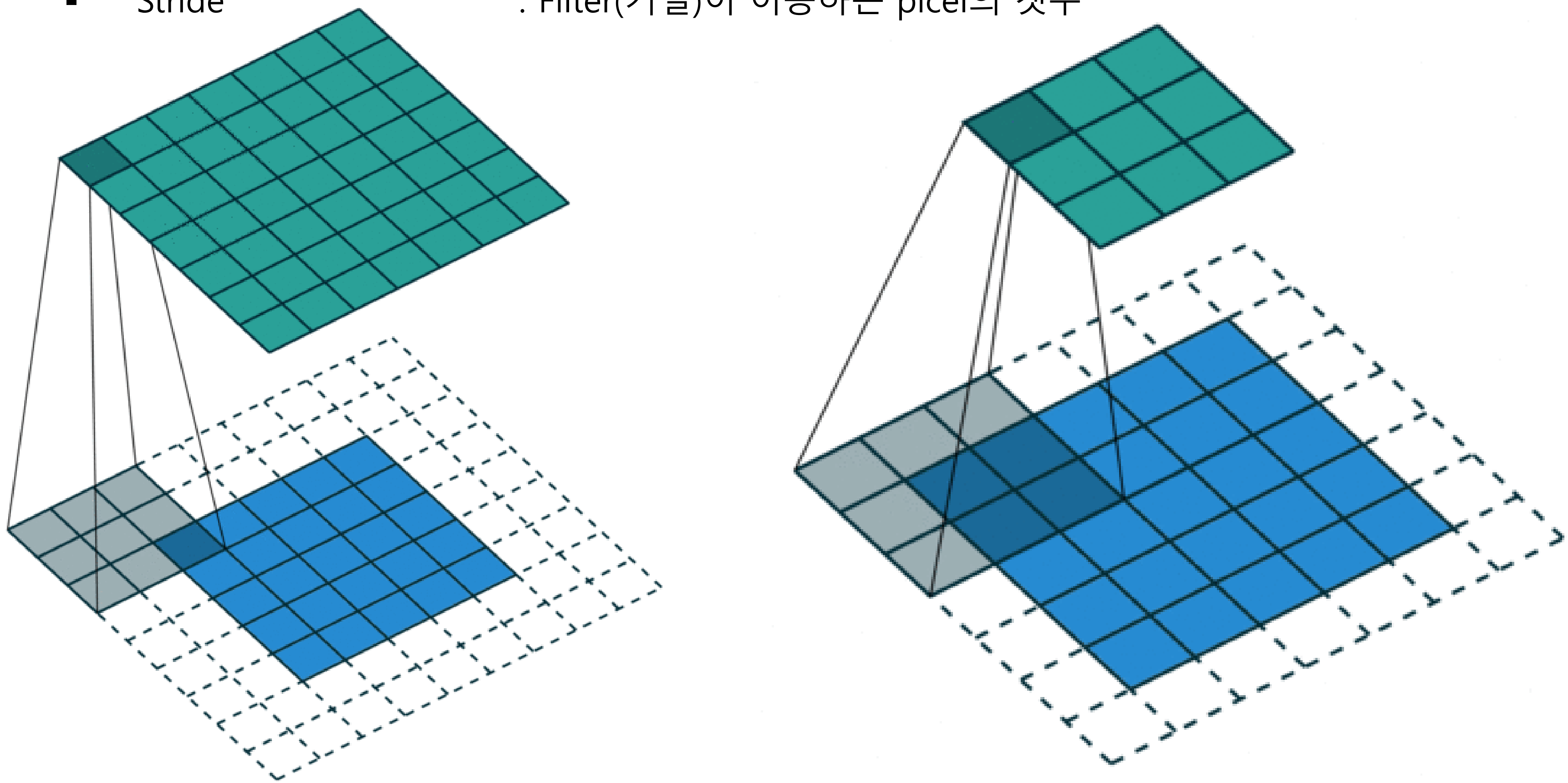


출처: https://pub.towardsai.net/convolutional-neural-networks-cnns-tutorial-with-python-417c29f0403f

Layers

Layer 종류 : keras.layers.Conv2D

- Convolution
  - Padding : 주로 입력과 출력의 크기를 동일하도록 조정
  - Stride : Filter(커널)이 이동하는 picel의 갯수



출처: https://pub.towardsai.net/convolutional-neural-networks-cnns-tutorial-with-python-417c29f0403f

Layers

Layer 종류 : keras.layers.Conv2D

- Convolution
  - Filter

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Input Image

Convoluted Image

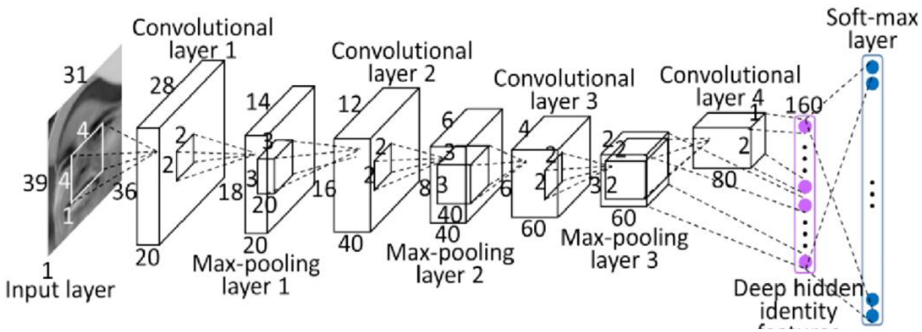
- Filter
  - 특정 효과를 가지는 filter에 대한 학습이 가능 하다.
  - 특정 이미지만을 위한 filter가 생성 된다.

출처: https://pub.towardsai.net/convolutional-neural-networks-cnns-tutorial-with-python-417c29f0403f

Layers

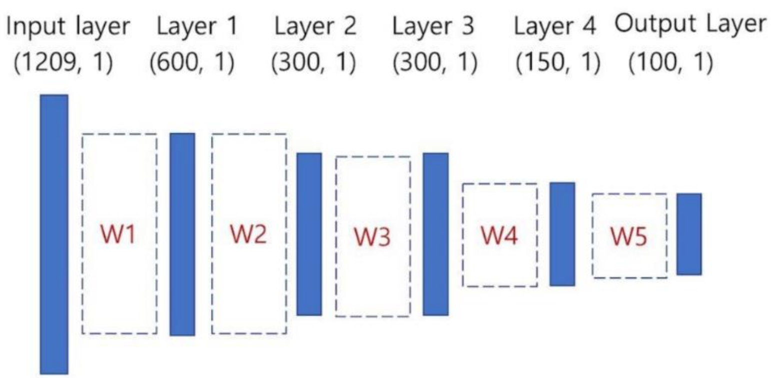
Layer 종류 : keras.layers.Conv2D

- 4개의 컨볼루션 레이어, 39x31x1 크기의 입력 데이터, 100개의 클래스로 분류
  - 20만개 parameter → Dense에 비해 학습이 쉽고 처리 속도가 빠르다.



출처: https://github.com/soofware/Speech-Recognition-Tutorial/blob/master/seminar/CNN.pdf

- 4개의 dmsslrcmd, 1209x1(39x31x1) 크기의 입력 데이터, 100개의 클래스로 분류
  - 100만개 parameter. 은닉층이 깊어질 수록 급격히 늘어남

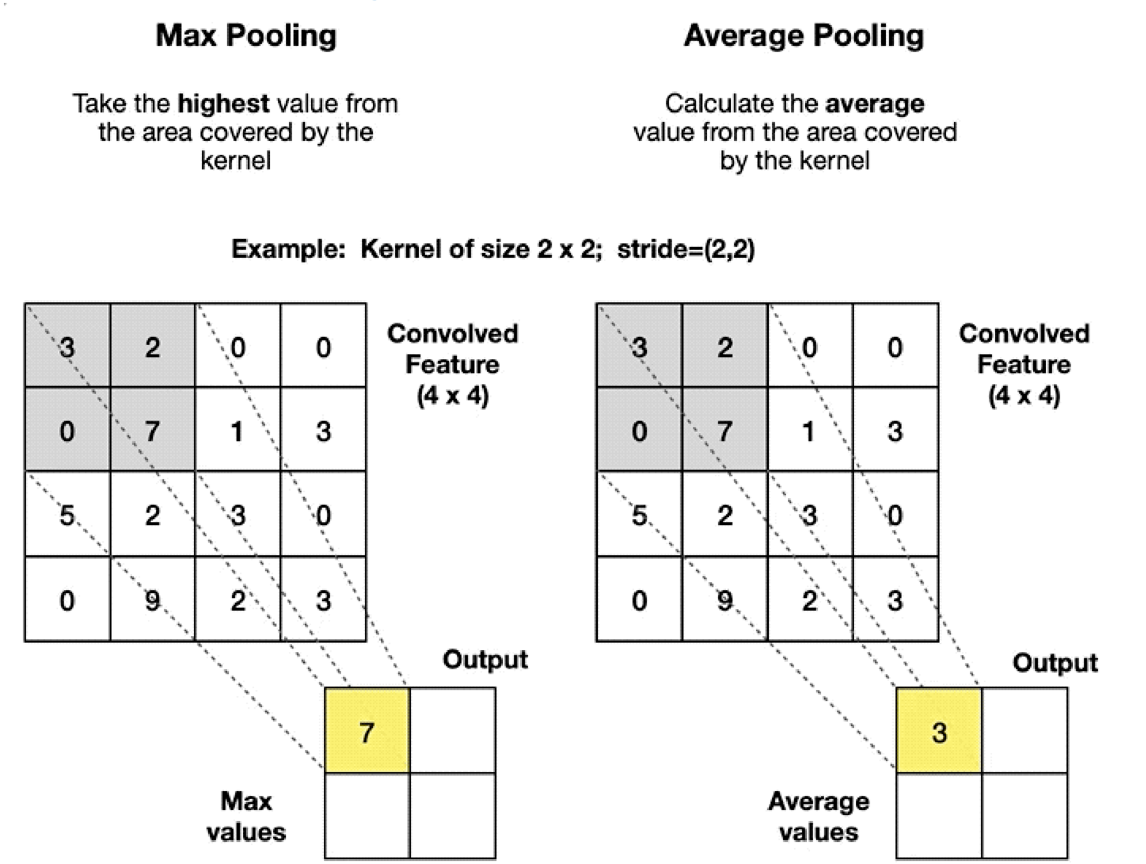


출처: https://github.com/soofware/Speech-Recognition-Tutorial/blob/master/seminar/CNN.pdf

Layers

Layer 종류 : keras.layers.MaxPooling2D

- Pooling
  - n차원 부분 행열을 사용하여 특징맵을 요약
  - 주로 Max Pooling을 사용 한다.



출처: https://pub.towardsai.net/convolutional-neural-networks-cnns-tutorial-with-python-417c29f0403f

Layers

Layer 종류 : keras.layers.MaxPooling2D

- Pooling

- 효과
  - 오류 보완
  - 확대

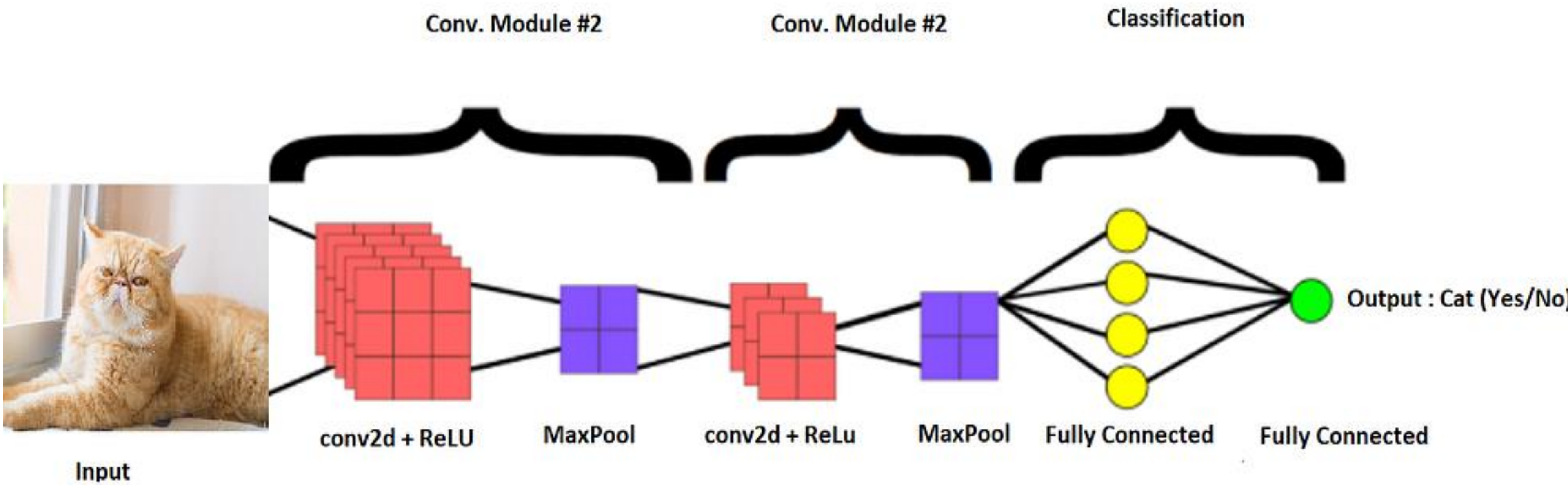
출처: https://pub.towardsai.net/convolutional-neural-networks-cnns-tutorial-with-python-417c29f0403f

Layers

Layer 종류 : keras.layers.Conv2D, keras.layers.MaxPooling2D

- CNN (Convolutional Neural Networks, 합성곱 신경망)
  - DCNN (Deep CNN, 심층 합성곱 신경망)
  - Convolution layer
  - ReLU (Rectified Linear Unit)
  - Pooling layer
  - FC (Fully connected) layer

- 데이터 변환 계층
- Receptive field (로컬 수용 필드)
- 가중치 공유

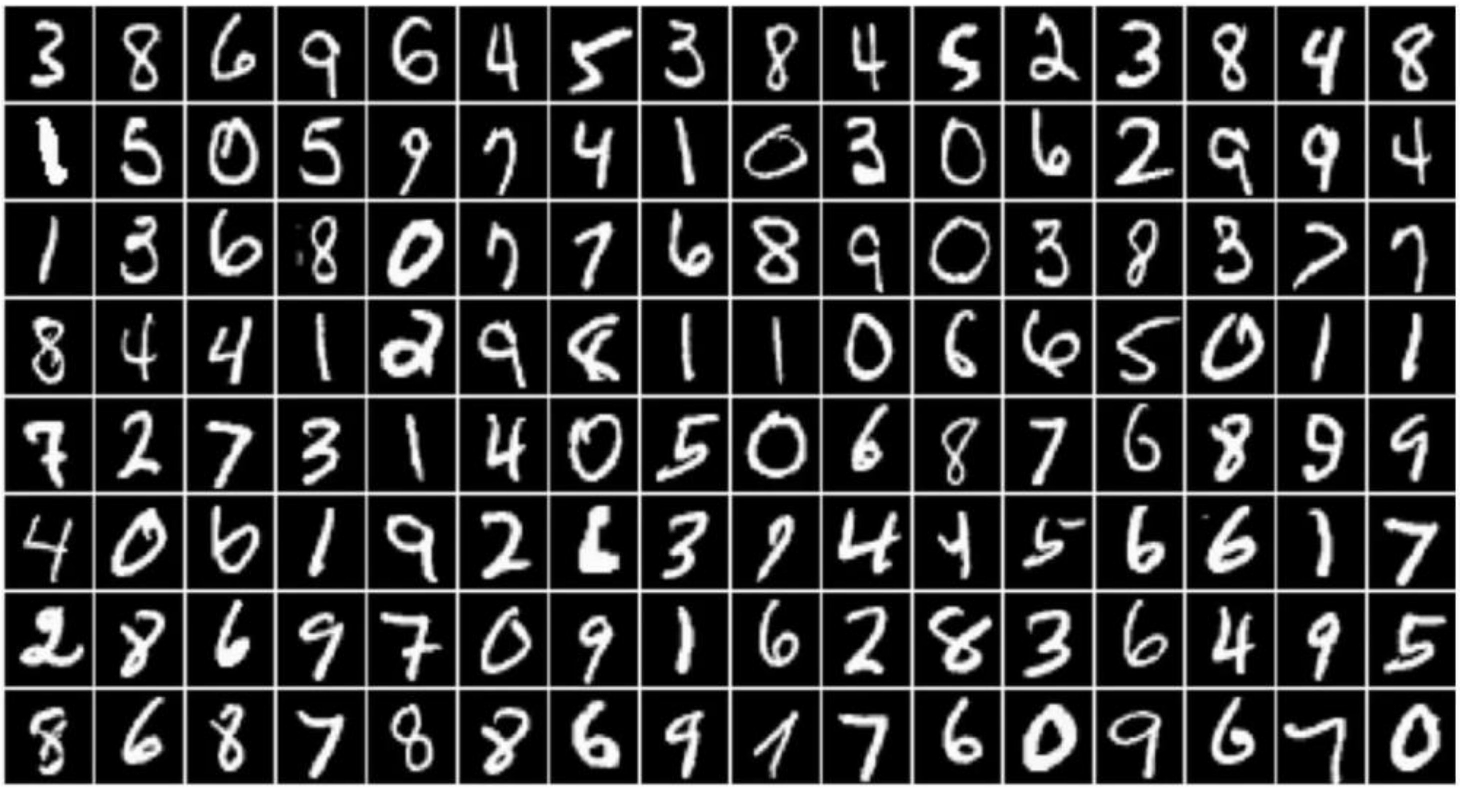


출처: https://pub.towardsai.net/convolutional-neural-networks-cnns-tutorial-with-python-417c29f0403f

Layers

CNN (Convolutional Neural Networks, 합성곱 신경망)

- Input
  - 숫자 손글씨
  - 28 * 28



출처: https://learnopencv.com/implementing-mlp-tensorflow-keras/

Layers

CNN (Convolutional Neural Networks, 합성곱 신경망)

- 데이터 가공
  - Nomalization
  - Layer에서 데이터 가공과 다른 점은 "**한번만 실행된다**"는 것 이다.

- 훈련 데이터 : train
  - 훈련용 : train : 80%
  - 검증용 : validation : 20%

- 평가 데이터 : test

- ephochs
  - 훈련 집합 횟수

- batch_size
  - 훈련 집합당 훈련 횟수

Layers

CNN (Convolutional Neural Networks, 합성곱 신경망)

- Model

```
IMG_ROWS, IMG_COLS = 28, 28
input_shape = (IMG_ROWS, IMG_COLS, 1)

model = keras.models.Sequential()
model.add(keras.layers.Conv2D(20, (5, 5), activation='relu', input_shape=input_shape))
model.add(keras.layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

model.add(keras.layers.Conv2D(50, (5, 5), activation='relu'))
model.add(keras.layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(500, activation="relu"))
model.add(keras.layers.Dense(self.nb_classes, activation="softmax"))

model.compile(
    optimizer=keras.optimizers.Adam(),
    loss='categorical_crossentropy',
    metrics=['accuracy' ]
)
```

Layers

CNN (Convolutional Neural Networks, 합성곱 신경망)

- Output

  Model: "sequential"

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 24, 24, 20)        520
 max_pooling2d (MaxPooling2D)  (None, 12, 12, 20)      0
 conv2d_1 (Conv2D)           (None, 8, 8, 50)          25050
 max_pooling2d_1 (MaxPooling2D)  (None, 4, 4, 50)      0
 flatten (Flatten)           (None, 800)               0
 dense (Dense)               (None, 500)               400500
 dense_1 (Dense)             (None, 10)                5010
=================================================================
Total params: 431,080
Trainable params: 431,080
Non-trainable params: 0
```

Layers

CNN (Convolutional Neural Networks, 합성곱 신경망)

- Output

Epoch 1/5

2023-08-03 14:06:54.469518: I tensorflow/stream_executor/cuda/cuda_dnn.cc:384] Loaded cuDNN version 8600

24/24 [==============================] - 5s 52ms/step - loss: 1.1939 - accuracy: 0.7053 - val_loss: 0.4708 - val_accuracy: 0.8558

Epoch 2/5

24/24 [==============================] - 1s 39ms/step - loss: 0.3677 - accuracy: 0.8913 - val_loss: 0.3250 - val_accuracy: 0.9014

Epoch 3/5

24/24 [==============================] - 1s 38ms/step - loss: 0.2287 - accuracy: 0.9403 - val_loss: 0.2305 - val_accuracy: 0.9316

Epoch 4/5

24/24 [==============================] - 1s 38ms/step - loss: 0.1650 - accuracy: 0.9547 - val_loss: 0.1785 - val_accuracy: 0.9457

Epoch 5/5

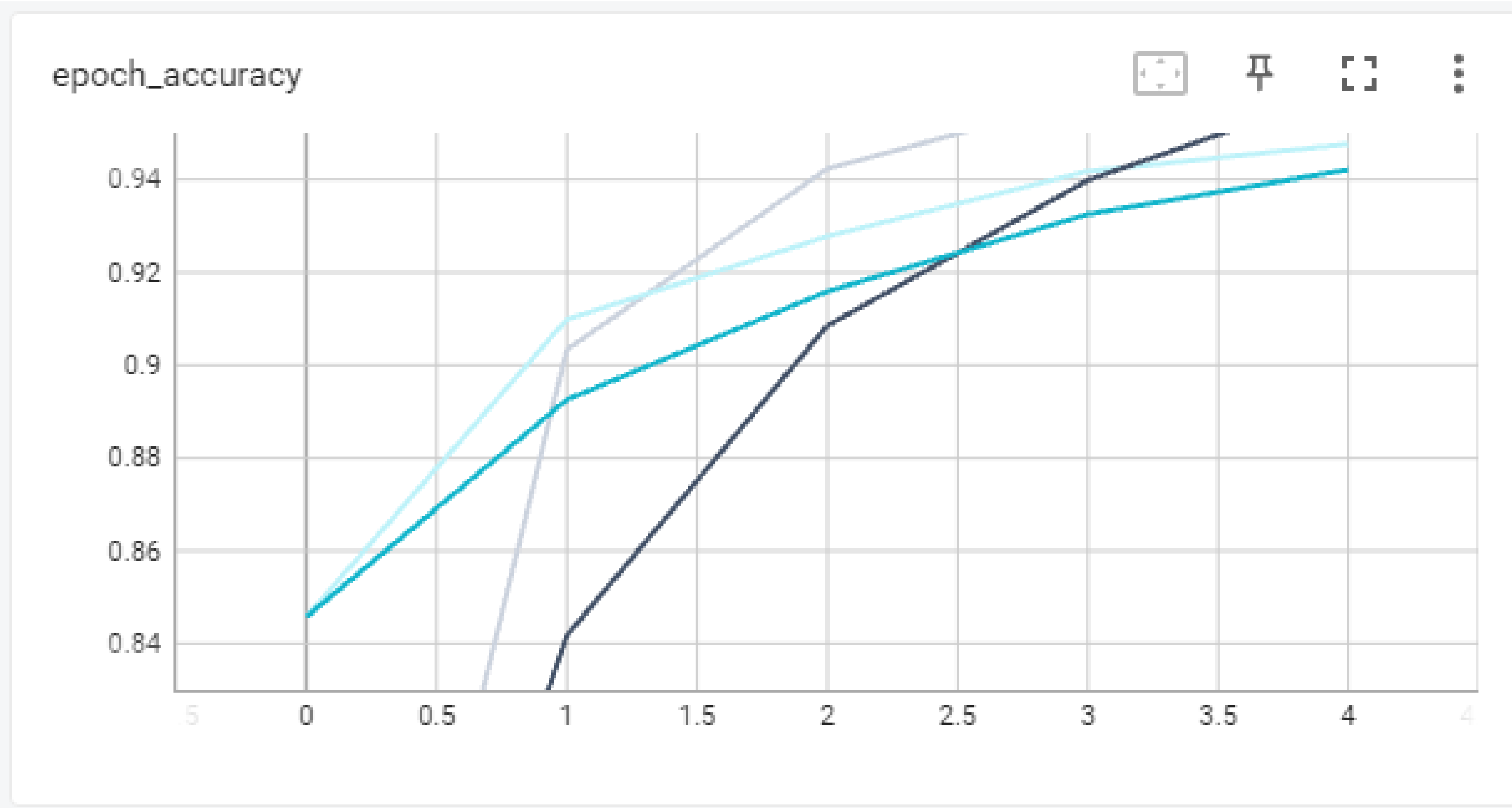
24/24 [==============================] - 1s 38ms/step - loss: 0.1240 - accuracy: 0.9697 - val_loss: 0.1629 - val_accuracy: 0.9504

313/313 [==============================] - 1s 2ms/step - loss: 0.1474 - accuracy: 0.9541

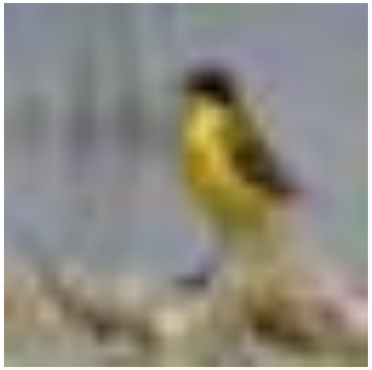
Layers

CNN (Convolutional Neural Networks, 합성곱 신경망)

- Graph



Deep Learning 응용

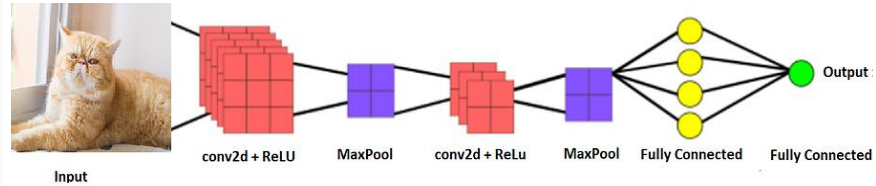


60,000 이미지
100개 분류
분류당 600개 이미지

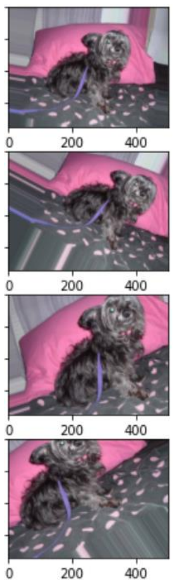
32 * 32 picels

출처: https://www.cs.toronto.edu/~kriz/cifar.html

분류



출처: https://pub.towardsai.net/convolutional-neural-networks-cnns-tutorial-with-python-417c29f0403f



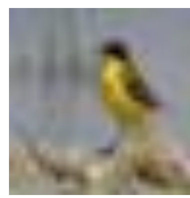
출처: https://ysyblog.tistory.com/152

*** 활용 ***
Data Augmentation
(데이터 증식)

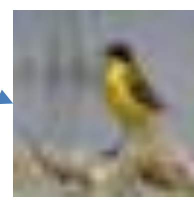
1

생성기 → 판별기

생성기



이미지



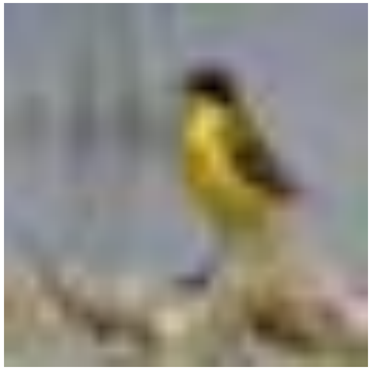
생성한 이미지

판별기

이미지를 생성하는 모델과
이미지를 판별하는 모델을
모두 학습 시킨다

GAN (Generative Adversarial Networks, 생성적 적대 신경망)

Deep Learning 응용

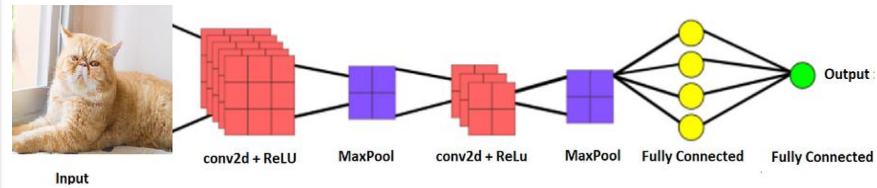


60,000 이미지
- 100개 분류
- 분류당 600개 이미지

32 * 32 picels

출처: https://www.cs.toronto.edu/~kriz/cifar.html

분류

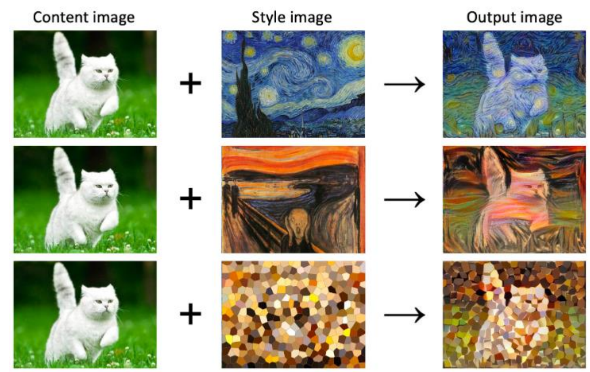


출처: https://pub.towardsai.net/convolutional-neural-networks-cnns-tutorial-with-python-417c29f0403f

2 Encoder → Decoder

분류기 → 변환기

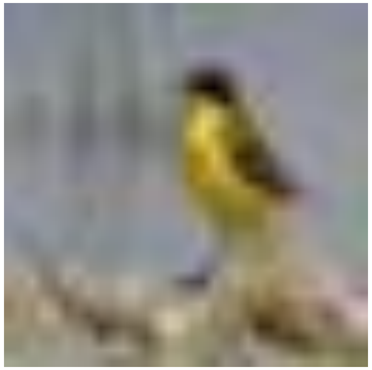
- Convolution 효과
  - Style transfer : 학습한 스타일(Filter)로 스타일 변환
  - 습득한 스타일(Filter)로 효과 추가
  - 이미지 해상도 확대
- Pooling 효과
  - 손상된 부분 복원
    - 이미지에서 사람 지우기



https://gm-note.tistory.com/entry/머신러닝-Style-transfer스타일-변환

원본 이미지와 비교하여 학습

AutoEncoder

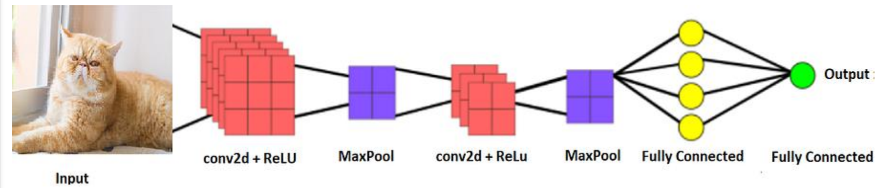


60,000 이미지
- 100개 분류
- 분류당 600개 이미지

32 * 32 picels

출처: https://www.cs.toronto.edu/~kriz/cifar.html

회귀
x, y, width, height



출처: https://pub.towardsai.net/convolutional-neural-networks-cnns-tutorial-with-python-417c29f0403f

분류와 지역화

분류

회귀

- 객체 탐지 (분류 + 회귀)

RNN

회귀

~~상태~~ 다음 입력으로 사용

학습 기능 없음

RNN (Recurrent Neural Network, 순환 신경망)

- 1D

- 2D
  - 이미지 (x, y)

- 3D
  - 동영상 (x, y, time)
  - 컬러 이미지 (x, y, rgb)

- 4D
  - 컬러 동영상 (x, y, rgb, time)

- Word Embedding (단어 임베딩)
  - To 1D
  - To 2D : NLP (Natural Language Process)

- Music

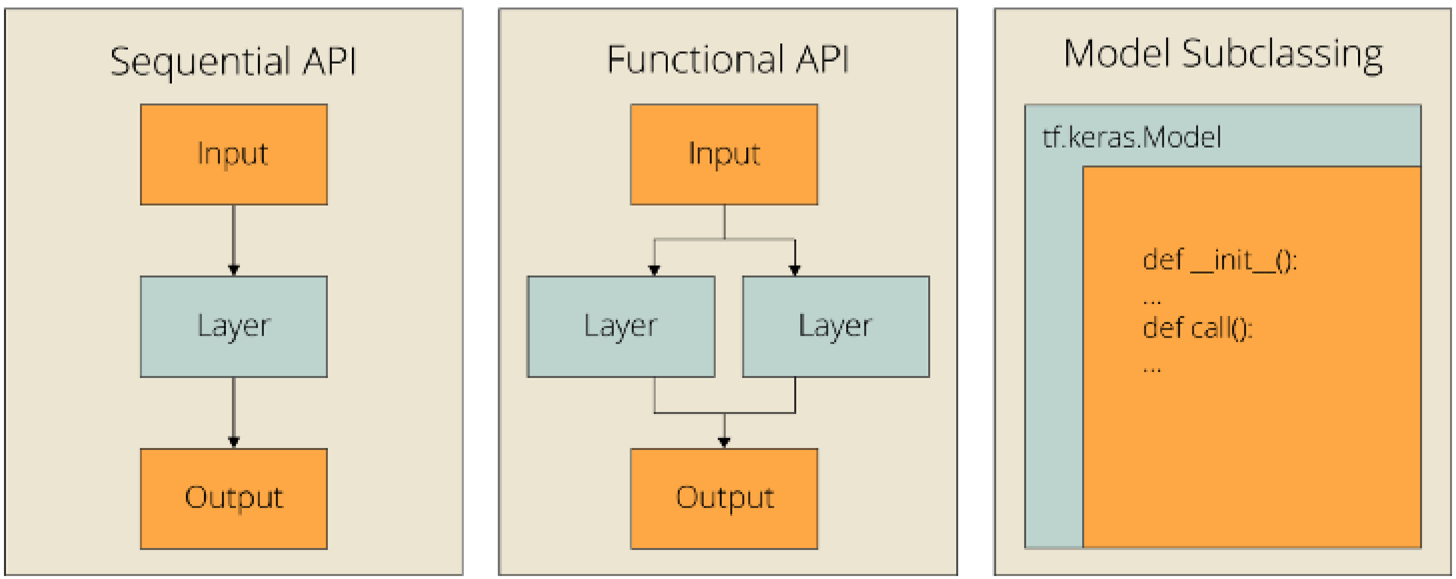
분류

입력 데이터의 종류에 따른 모델의 적용 방안을 검토할 것

Model 구현 방식

Model 구현 방식

- Sequential API : 간단한 모델 구현
- Functional API : 복잡한 모델 구현
- Model Subclassing
  - Function API로 구현하기 힘든 모델 구현
  - 자유도가 제일 높은 모델 구축 방법



출처: https://wikidocs.net/106897

Model 구현 방식

Sequential API

- CNN

```
model = keras.models.Sequential()
model.add(keras.layers.Conv2D(20, (5, 5), activation='relu', input_shape=input_shape))
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(keras.layers.Dropout(self.dropout))

model.add(keras.layers.Conv2D(50, (5, 5), activation='relu'))
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(keras.layers.Dropout(self.dropout))

model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(self.nb_classes, activation="softmax"))

model.compile(
    optimizer=self.optimizer,
    loss=self.loss_function,
    metrics=[ self.metrics ]
)
```

Model 구현 방식

작성중

Functional API

- ppp

Model 구현 방식

Model Subclassing

- AutoGraph
  - Graph : tensorflow.Operation 객체의 집합을 포함하고 있는 데이터 구조
  - 연산의 단위와 텐서 객체, 연산간에 흐르는 데이터의 단위

```
@tf.function
def dense_layer(x, w, b):
    return tf.matmul(x, w) + b
```

```
dense_layer.pyton_function(x, w, b)
```
: 원본 Python 함수 호출

Model 구현 방식

Model Subclassing

- Custom (사용자 정의)

```
class CustomModel(keras.Model):

class CustomLayer(keras.layers.Layer):

@tf.function
def CustomActivationFunction(x, axis=-1):

class CustomLossFunction(keras.losses.Loss):

class CustomOptimizer(keras.optimizers.Optimizer):

class CustomMetric(keras.metrics.Metric):

class CustomCallback(keras.callbacks.Callback):
```

Model 구현 방식

Model Subclassing

- Custom (사용자 정의)

```
model = tf.keras.Sequential([])

model.save('my_model')
keras.models.load_model("my_model", custom_objects={ "CustomModel": CustomModel })



with open('my_model.json', 'w') as json_file:
    json_file.write(model.to_json())

my_model = keras.models.model_from_json(
                                        open('my_model.json').read(),
                                        custom_objects={ "CustomModel": CustomModel }
)
```

Application

Application

- Application
  - Model + Weight
  - 최적화
  - AutoML
  - TFLite for TensorFlow Lite
  - tensorflow.js

```
json = model.to_json()
model keras.models.model_from_json(json)

model.save_weights('~.h5)
model.load_weights(filename)
```

Application

- json model

```
{
    class_name: 'Sequential',
    config: {
        name: 'sequential',
        layers: [
            { class_name: 'InputLayer', config: { 생략 } },
            { class_name: 'Conv2D', config: { 생략 } },
            { class_name: 'MaxPooling2D', config: { 생략 } },
            ......
            { class_name: 'Dropout', config: { 생략 } },
            { class_name: 'Dense', config: { 생략 } }
        ]
    },
    keras_version: '2.10.0',
    backend: 'tensorflow'
}
```

Application

Application

- weight

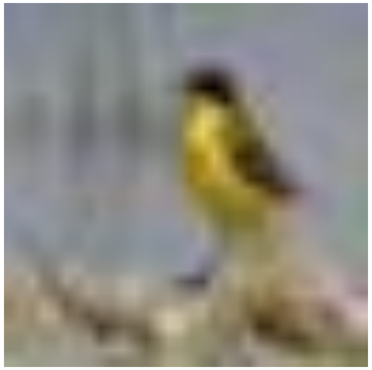
  dense/kernel:0 : Array<float32>
  dense/bias:0 : Array<float32>
  dense_1/kernel:0 : Array<float32>
  dense_1/bias:0 : Array<float32>
  dense_2/kernel:0 : Array<float32>
  dense_2/bias:0 : Array<float32>

$$\underbrace{H(x)}_{\text{Hypothesis}} = \underbrace{w}_{\text{Weight}}x + \underbrace{b}_{\text{Bias}}$$

출처: https://brunch.co.kr/@linecard/321

Deep Learning 응용

작성중



60,000 이미지
  100개 분류
  분류당 600개 이미지

32 * 32 picels

출처: https://www.cs.toronto.edu/~kriz/cifar.html

ppp

분류

기존 application 재활용
분류 종류 확대

y = softmax(H(x))
  = softmax(wx + b)

H(x) = wx + b

wx = H(x) - b
x = (H(x) - b) / w

난수를 발생시켜 H(x)로 사용하면
이미지(x)를 생성할 수 있음

Application

작성중

Application

- Application
  - resnet
  - resnet50
  - resnet_rs
  - resnet_v2
  - ResNet101
  - ResNet152
  - ResNet50
  - ResNetRS101
  - ResNetRS152
  - ResNetRS200
  - ResNetRS270
  - ResNetRS350
  - ResNetRS420
  - ResNetRS50
  - ResNet101V2
  - ResNet152V2
  - ResNet50V2

Application

작성중

Application

- Application
  - RegNetX002
  - RegNetX004
  - RegNetX006
  - RegNetX008
  - RegNetX016
  - RegNetX032
  - RegNetX040
  - RegNetX064
  - RegNetX080
  - RegNetX120
  - RegNetX160
  - RegNetX320

작성중

Application

- Application
  - RegNetY002
  - RegNetY004
  - RegNetY006
  - RegNetY008
  - RegNetY016
  - RegNetY032
  - RegNetY040
  - RegNetY064
  - RegNetY080
  - RegNetY120
  - RegNetY160
  - RegNetY320

Application

작성중

Application

- Application
  - EfficientNetB0
  - EfficientNetB1
  - EfficientNetB2
  - EfficientNetB3
  - EfficientNetB4
  - EfficientNetB5
  - EfficientNetB6
  - EfficientNetB7

  - EfficientNetV2B0
  - EfficientNetV2B1
  - EfficientNetV2B2
  - EfficientNetV2B3
  - EfficientNetV2L
  - EfficientNetV2M
  - EfficientNetV2S

Application

작성중

Application

- Application
  - MobileNet
  - MobileNetV2
  - MobileNetV3Large
  - MobileNetV3Small
  - mobilenet
  - mobilenet_v2
  - mobilenet_v3

  - vgg16
  - vgg19
  - VGG16
  - VGG19

작성중

Application

- Application
  - xception
  - Xception

  - inception_resnet_v2
  - inception_v3
  - InceptionResNetV2
  - InceptionV3

  - convnext
  - densenet
  - efficientnet
  - efficientnet_v2
  - imagenet_utils
  - nasnet
  - regnet

Application

작성중

Application

- Application
  - ConvNeXtBase
  - ConvNeXtLarge
  - ConvNeXtSmall
  - ConvNeXtTiny
  - ConvNeXtXLarge

  - DenseNet121
  - DenseNet169
  - DenseNet201

  - NASNetLarge
  - NASNetMobile

작성중

Model과 Application

- Application 전체를 재사용
  - Capsule : Application을 하나의 layout로 사용

Application

작성중

Model과 Application

- Application 일부를 재활용
  - 유사한 문제를 해결하는데 기존에 존재하는 application을 재활용
    - model의 일부를 삭제 후 추가
    - weigh의 일부를 삭제 후 추가

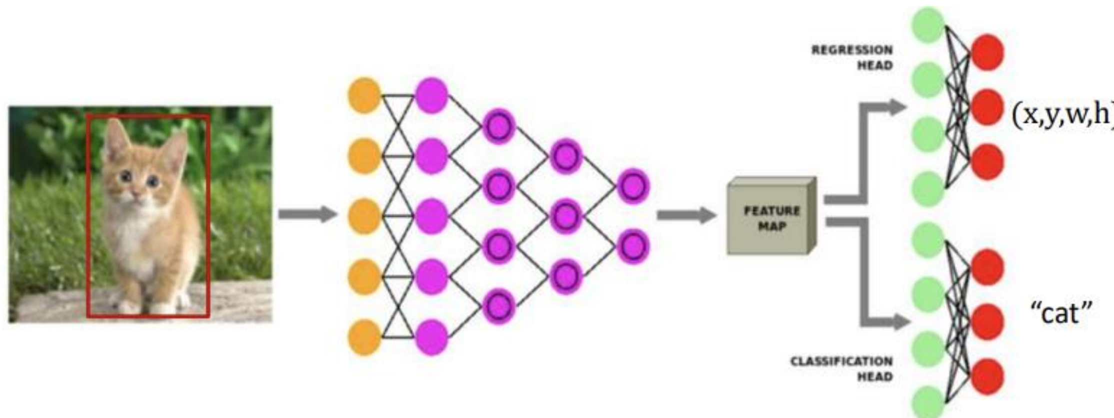
  - 학습에 필요한 충분한 데이터가 없는 경우 사용 가능
  - 이미지를 100가지로 분류하는 application이 있는데 이를 재활용하여 200가지로 분류하는 application을 만들려고 하는 경우

Application

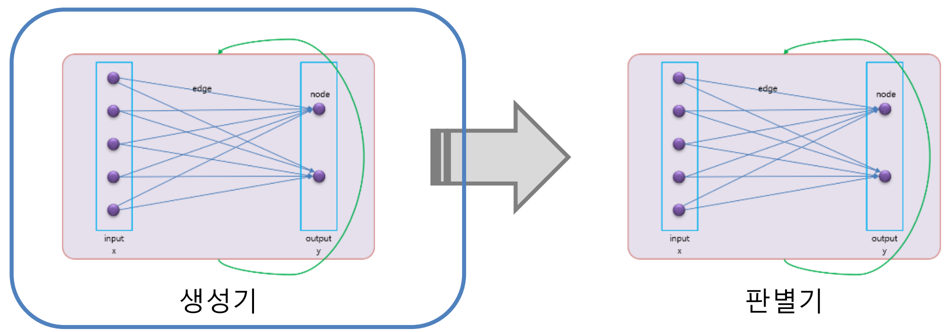
작성중

Model과 Application

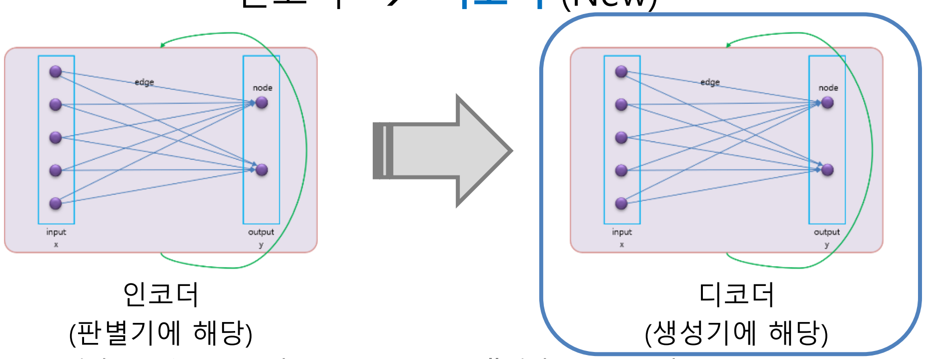
- Model과 Application
  - Model을 여러 개 결합



  - x를 생성
    - GAN (Generative Adversarial Networks, 생성적 적대 신경망)
      - **생성기** (New) → 판별기



  - y로 x를 생성
    - AutoEncoder : 인코더-디코더
    - x = f′(y) : 전치 함수
    - 인코더 → **디코더** (New)



Neural Networks

작성중

최적화

- Optimizer : 방식, 매개변수 변동폭
- epochs * batch_size
- hidden 개수
- model

- AutoML
  - HyperParameter
    - 좌측 항목 등
  - Loss Function

- 모델 선택 기준
  - Parameter 개수 최소화

Logistic Regression

Logistic Regression

- Activation Functions
  - Y = A * W + b : 수식. X = A
    - W (Weight, 가중치)
    - b (Bias, 오차)
- Loss Functions
  - Least squares : 오차 제곱합
- Optimizers
  - 미분을 사용하여 W와 b를 추정
- Metrics
- Back Propagation (오차 역전파)

Algorithms

Algorithms

작성중

layers 종류

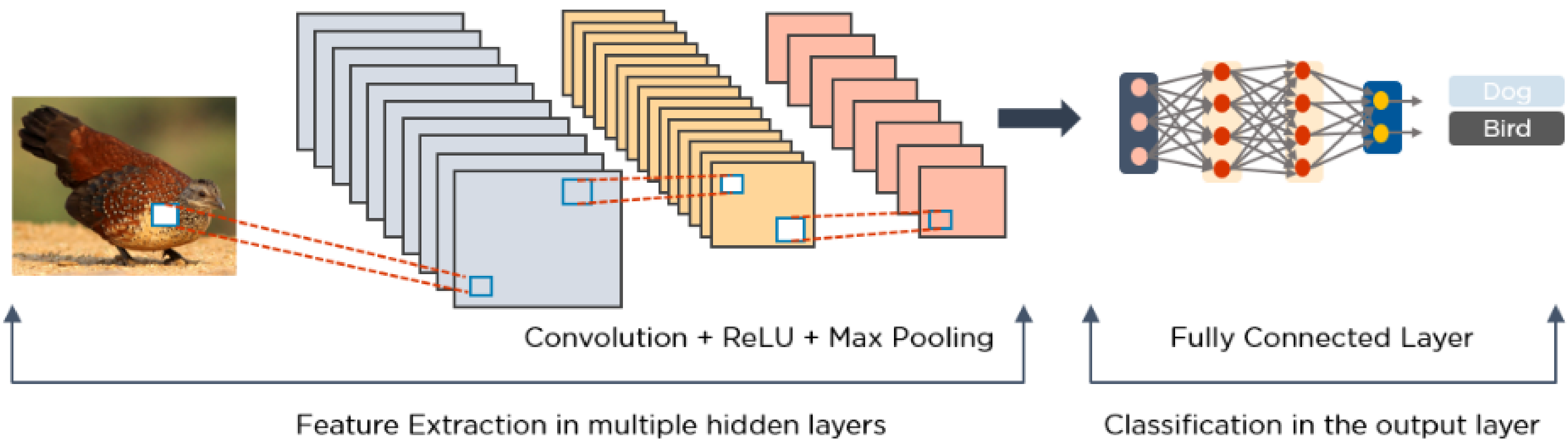
- CNN (Convolutional Neural Network)
- LSTM (Long Short Term Memory Network)
- RNN (Recurrent Neural Network)
- GAN (Generative Adversarial Network)
- RBFN (Radial Basis Function Network)
- MLP (Multilayer Perceptron)
- SOM (Self Organizing Map)
- DBN (Deep Belief Network)
- RBM (Restricted Boltzmann Machine)
- AutoEncoder

- https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithm
- https://www.projectpro.io/article/deep-learning-algorithms/443

작성중

CNN (Convolutional Neural Network)

- 이미지 처리
- 객체 탐지

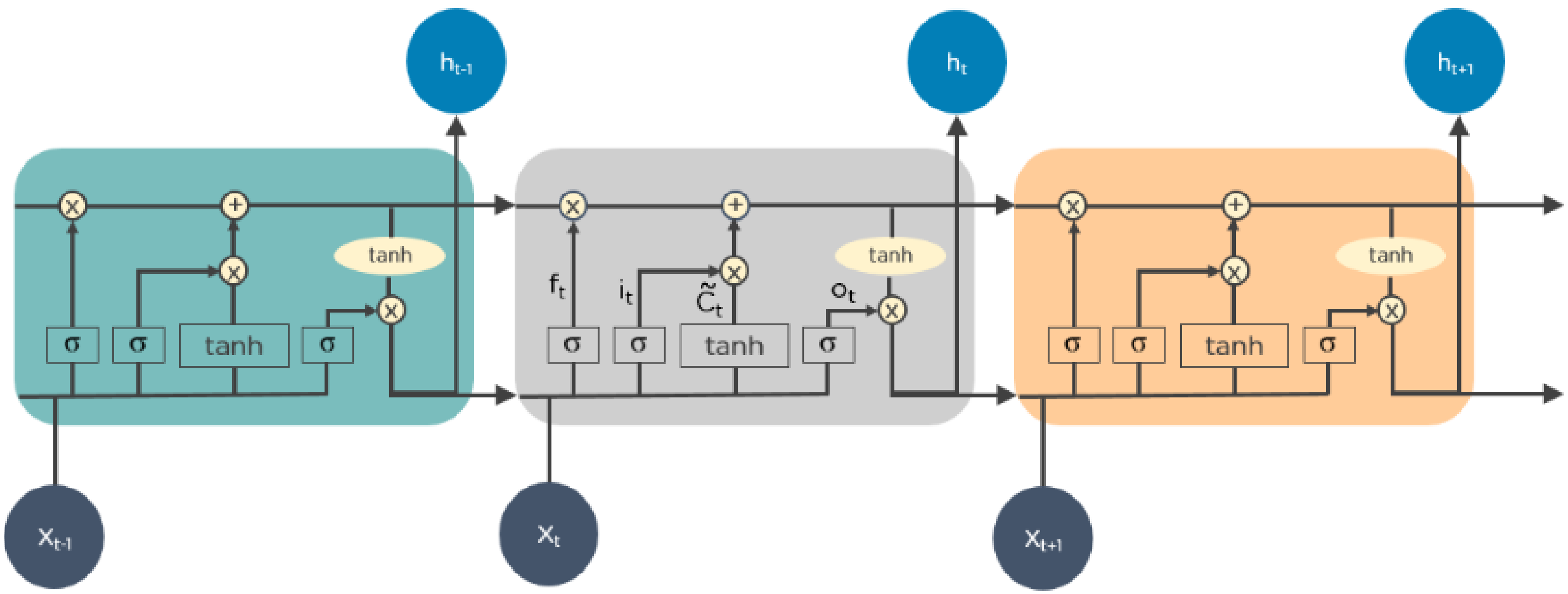


출처: https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithm

작성중

LSTM (Long Short Term Memory Network)

- 시계열 예측
- 음악 작곡. 음성 인식

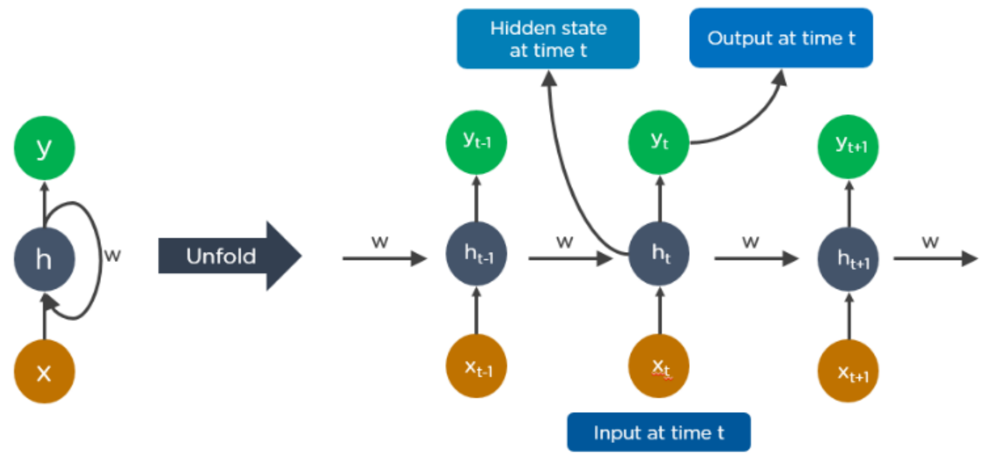


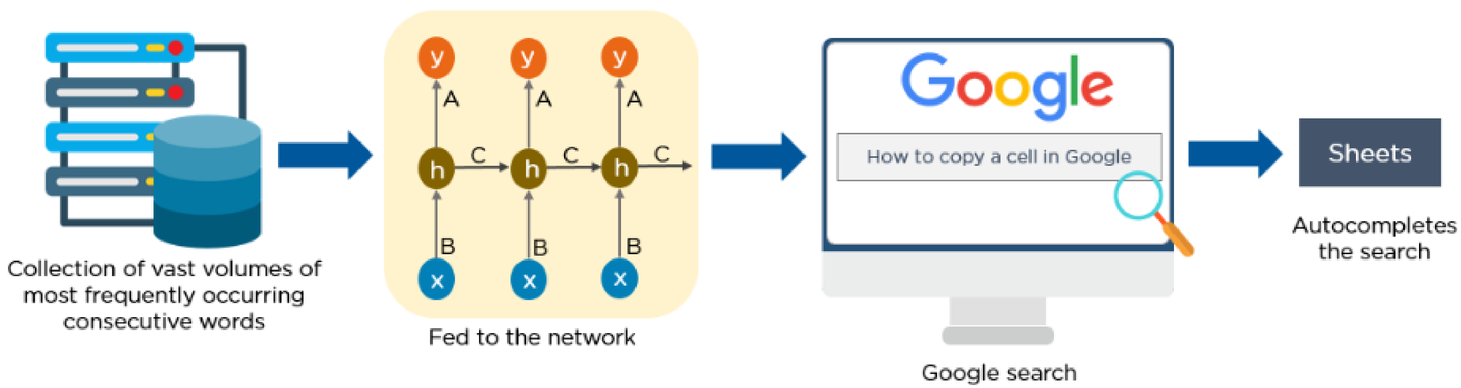
출처: https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithm

작성중

RNN (Recurrent Neural Network)

- 시계열 분석
- 자연어 처리
- 필기 인식
- 기계 번역
- 이미지 캡션



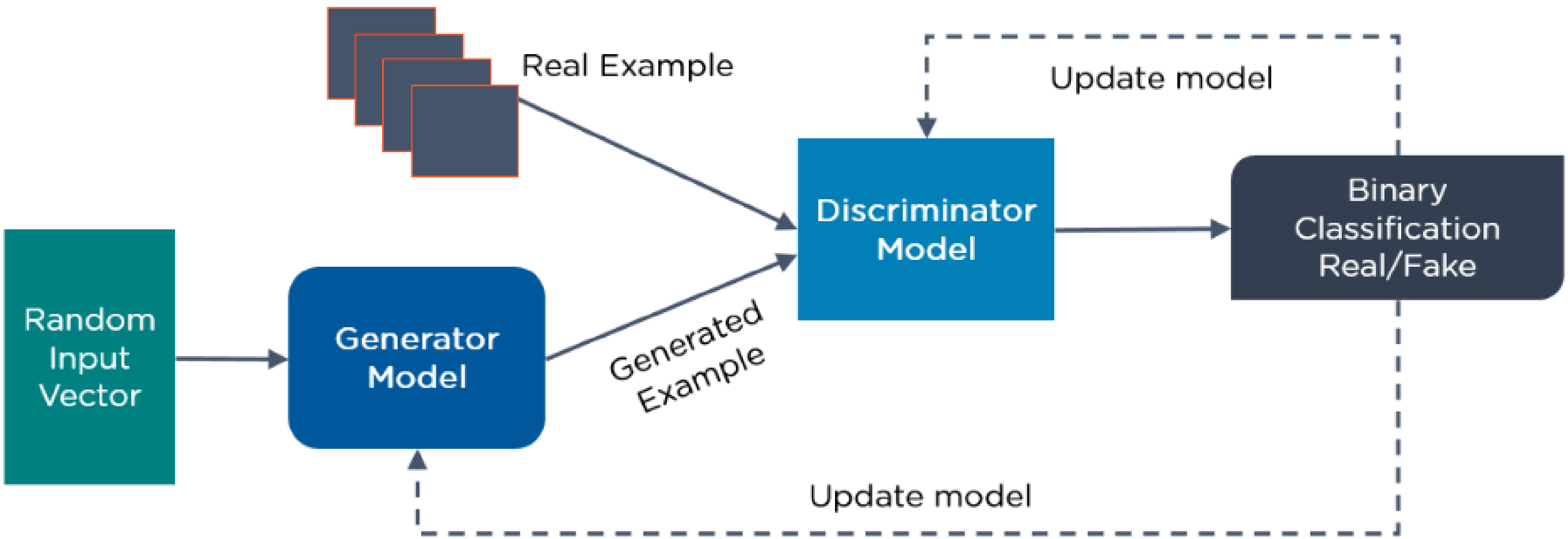


출처: https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithm

작성중

GAN (Generative Adversarial Network)

- 이미지 생성



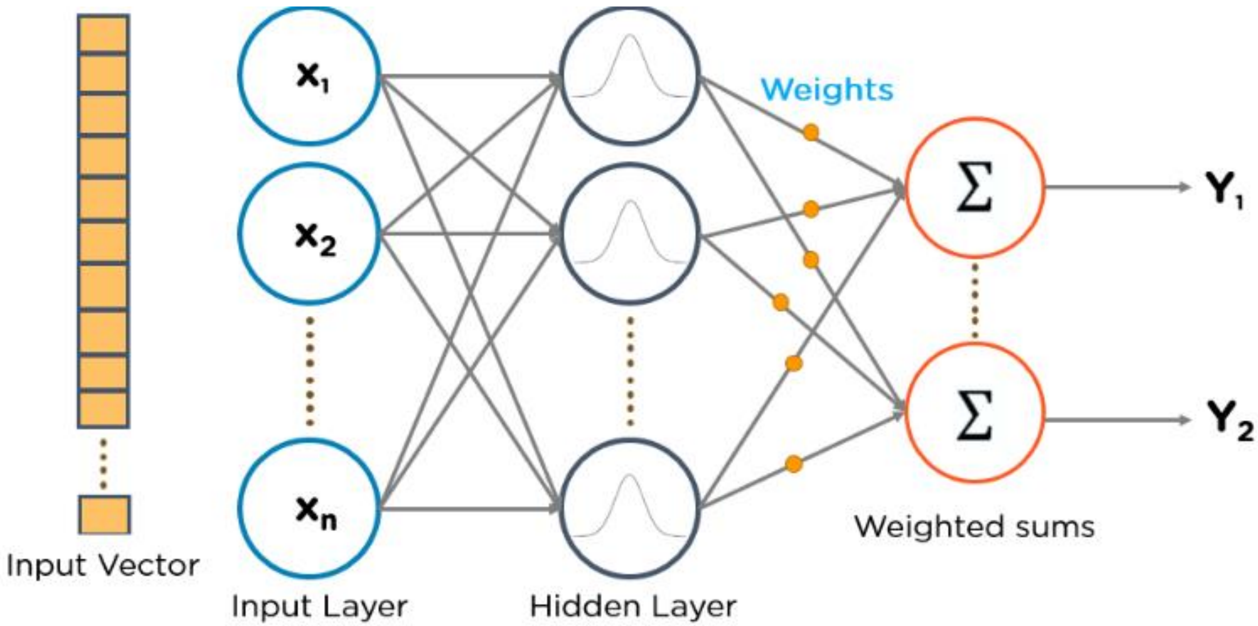
출처: https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithm

Algorithms

작성중

RBFN (Radial Basis Function Network)

- 분류, 회귀, 시계열 예측



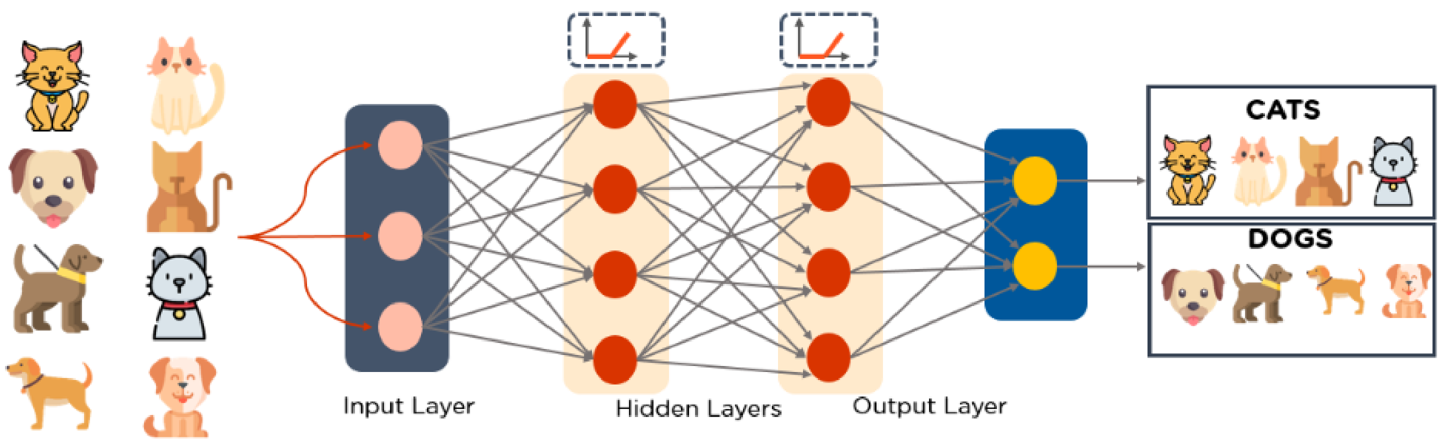
출처: https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithm

Algorithms

작성중

MLP (Multilayer Perceptron)

- 음성 인식
- 이미지 인식
- 기계 번역
- 소프트웨어 구축



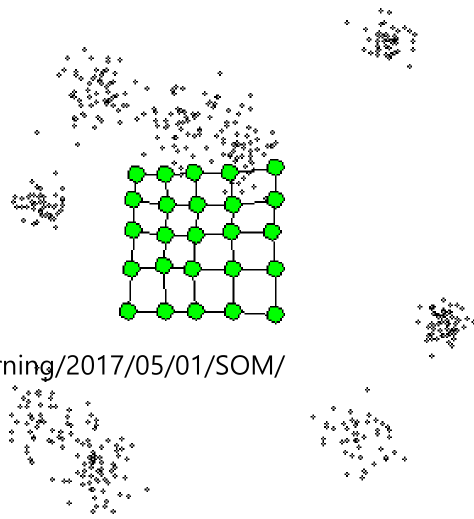
출처: https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithm

Algorithms

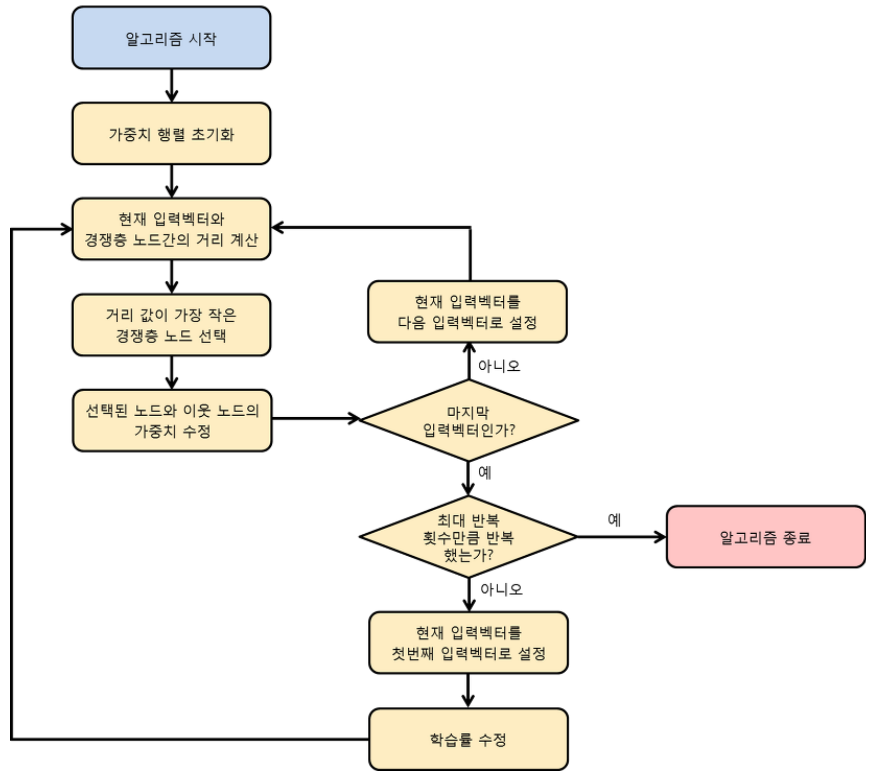
작성중

SOM (Self Organizing Map)

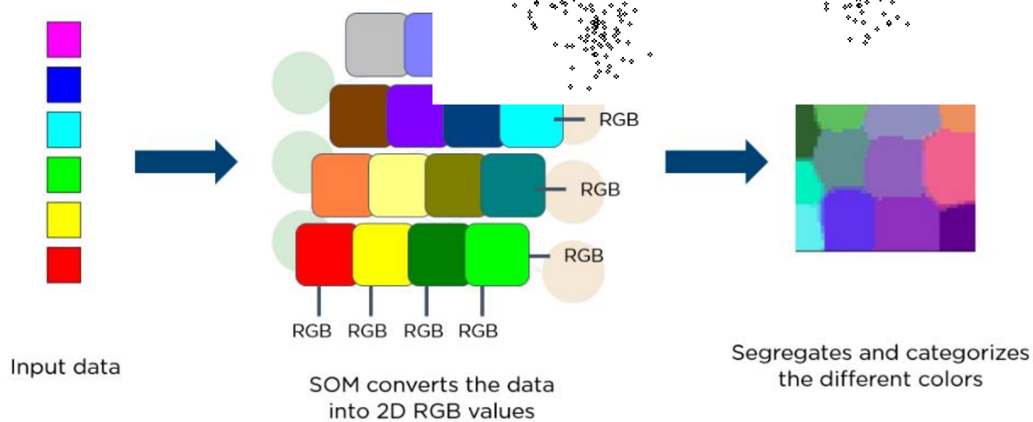
- 고차원의 데이터를 저차원의 격자 형태로 변환
  - 입력 데이터의 분포를 보존하면서 비슷한 패턴이나 특징을 가진 데이터를 인접한 노드에 매핑
- 구조 탐색 (분류)
- 차원 축소
- 시각화



출처: https://ratsgo.github.io/machine%20learning/2017/05/01/SOM/



출처: https://untitledtblog.tistory.com/5

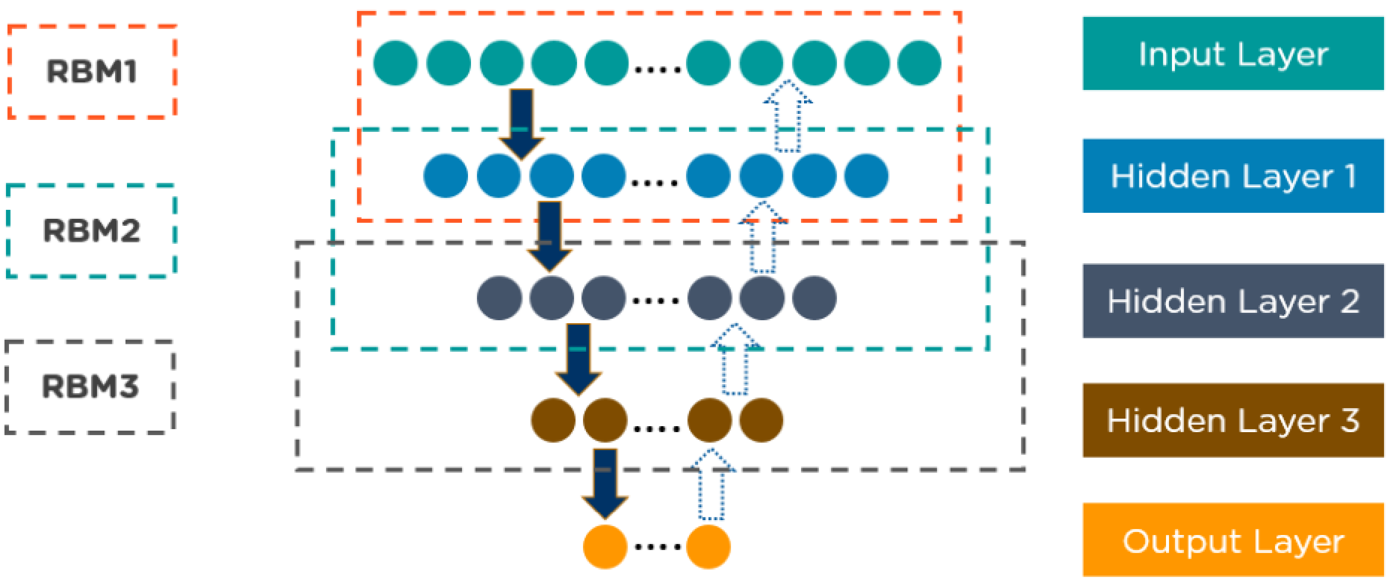


출처: https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithm

작성중

DBN (Deep Belief Network) ← RBM

- 이미지 인식
- 비디오 인식
- 모션 캡쳐

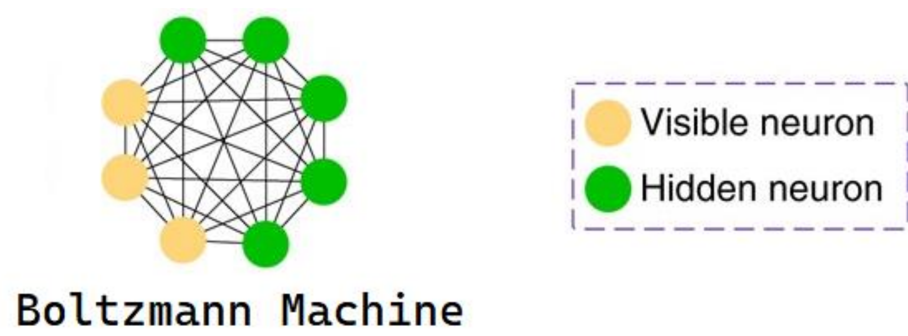


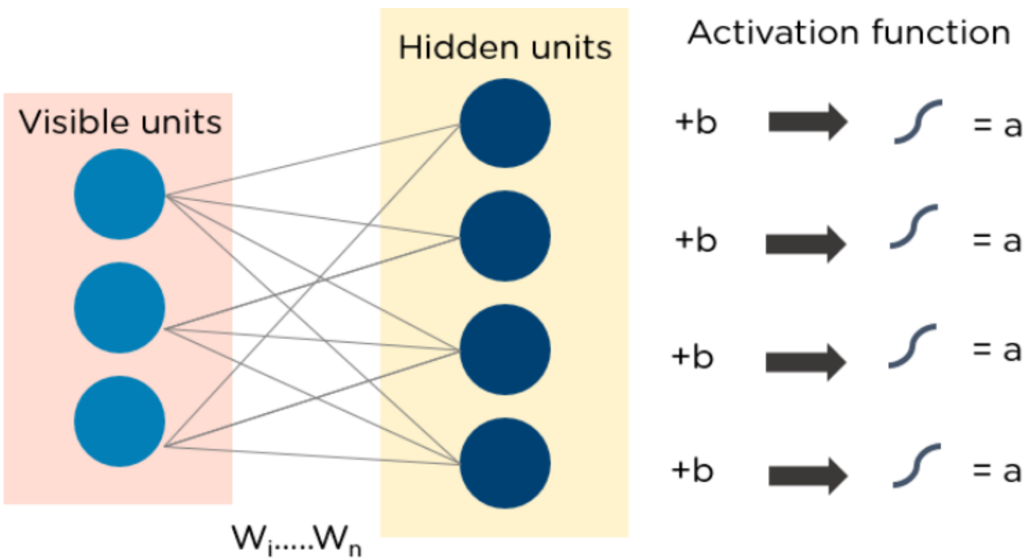
출처: https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithm

Algorithms

RBM (Restricted Boltzmann Machine) : 생성 AI

- Boltzmann Distribution를 기반으로한 확률 모형
  - Visible units : 특징 데이터
  - Hidden units : 확률 분포
- 학습 : 샘플 데이터와 유사한 데이터 생성
  - 분류, 협업, 특징값 학습
  - 선형 회귀 분석, 협업 필터링, 주제 모델링
  - 차원 감소



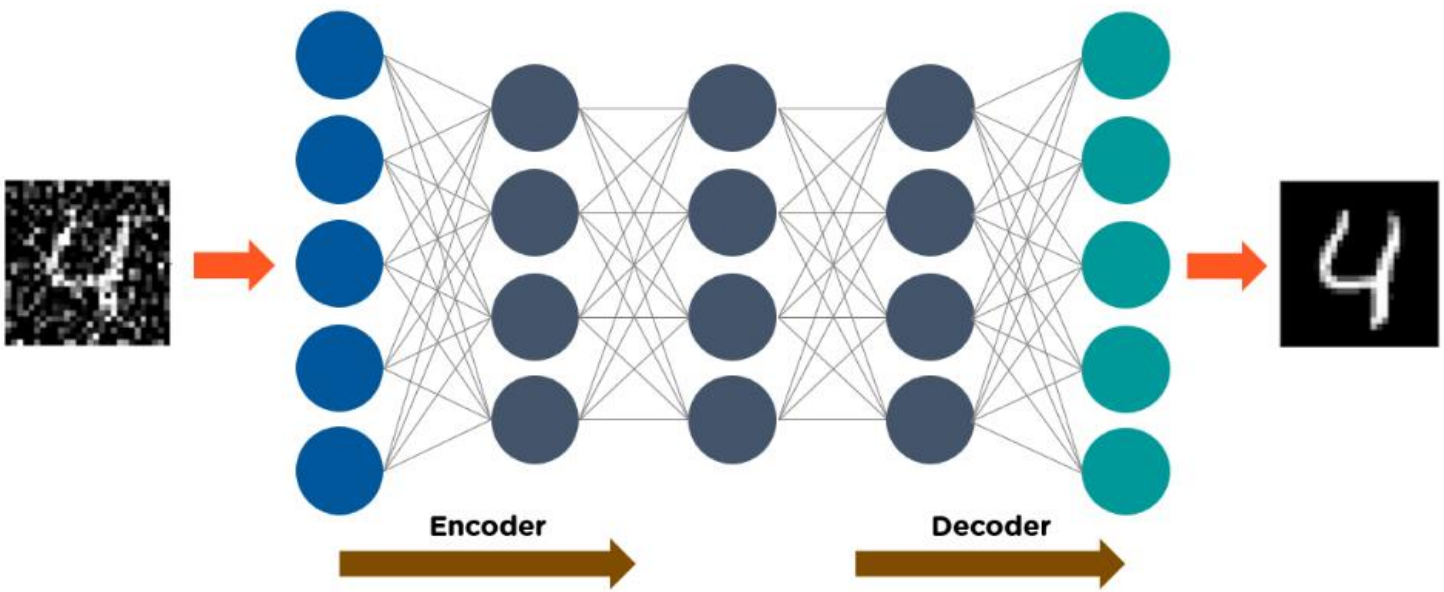


출처: https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithm

작성중

AutoEncoder : 생성 AI

- 입력 데이터의 분포를 학습하고, 분포에서 랜덤하게 샘플링하여 새로운 콘텐츠 생성
  - 이미지 복원과 생성. 이미지의 스타일 변경
  - 음악 생성. 텍스트 생성
  - 추천 시스템에서 상품을 추천



출처: https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithm