딥러닝 이야기

Lec 00 - Machine/Deep learning 수업의 개요와 일정

조회수 191,707회

Sung Kim
게시일: 2016. 3. 21.

$$Y = w \times X + b$$

“이정도의 수학을 이해할 수 있으면 충분히 따라올 수가 있고 머신러닝을 이해할 수가 있습니다” 김성훈 교수님

실측값 $$Y = w \times X + b$$

예측값 $$\hat{Y} = w \times \hat{X} + b$$

실측값

$$Y = w \times X + b$$

예측값

$$\hat{Y} = w \times \hat{X} + b$$

w와 b를 구해봅시다.

실측값

$$3 = w \times 1 + b$$

$$5 = w \times 3 + b$$

예측값

$$\hat{Y} = w \times 2 + b$$

w와 b를 구해봅시다.

$$3 = w \times 1 + b$$

실측값

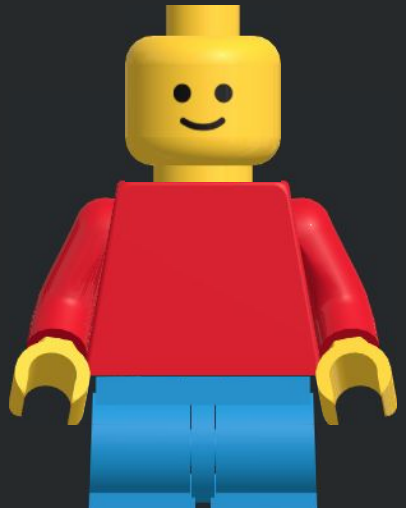
$$5 = w \times 3 + b$$

$$5 = w \times 4 + b$$

예측값

$$\hat{Y} = w \times 2 + b$$

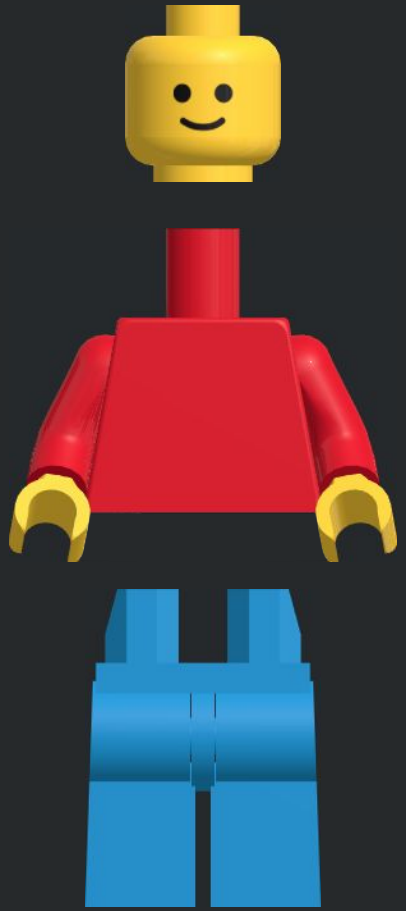
Model
모델

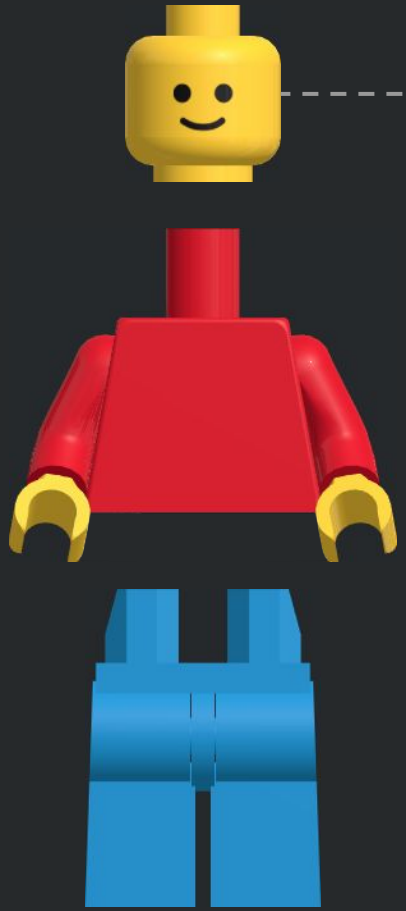


Model
모델

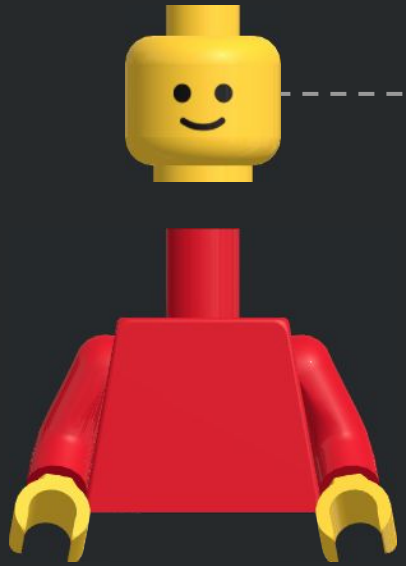
```
from keras.models import Sequential

model = Sequential()
```





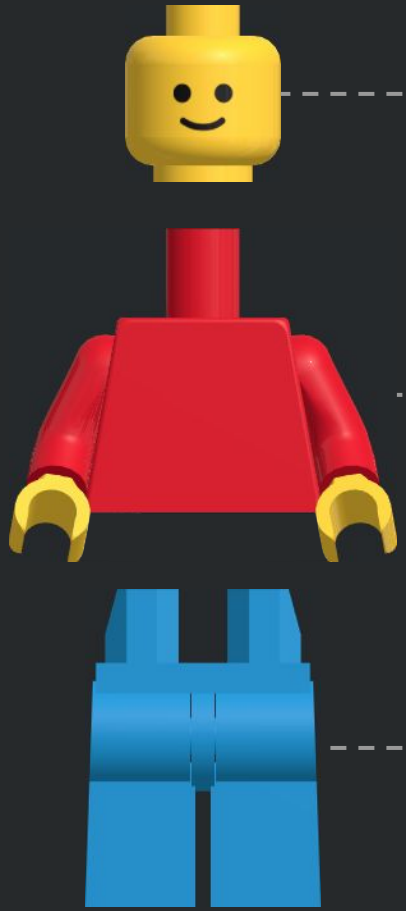
Network
네트워크



Network
네트워크

```
from keras.layers import Dense

model.add(Dense(units=64, activation='relu', input_dim=100))
model.add(Dense(units=10, activation='softmax'))
```

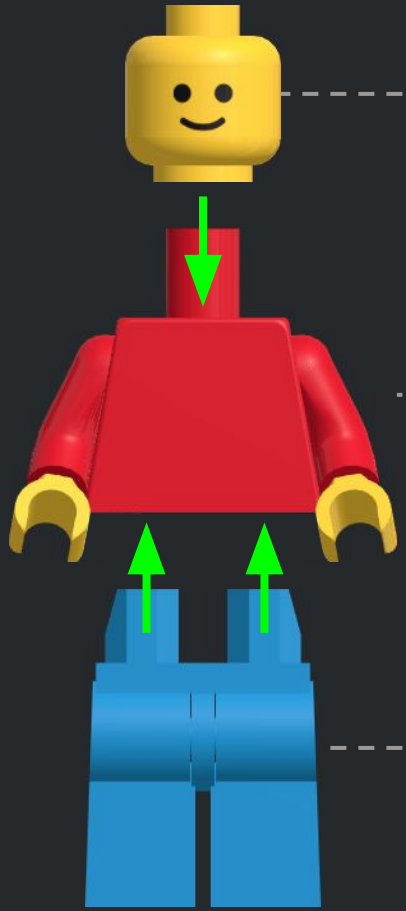


Network
네트워크

Objective Function
목표함수

Optimizer
최적화기

Compile
컴파일



Network
네트워크

Objective Function
목표함수

Optimizer
최적화기

Compile
컴파일

Network
네트워크

Objective Function
목표함수

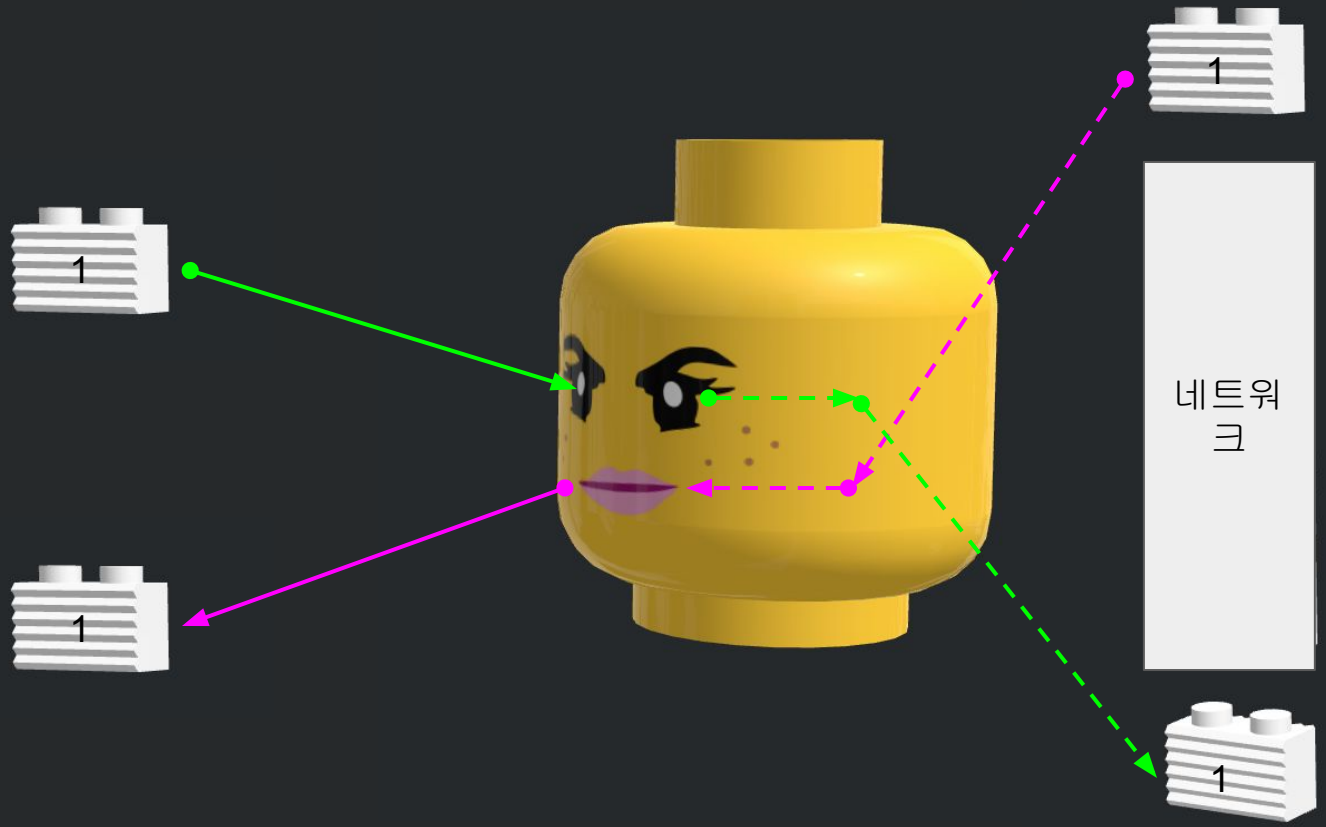
```
model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])
```

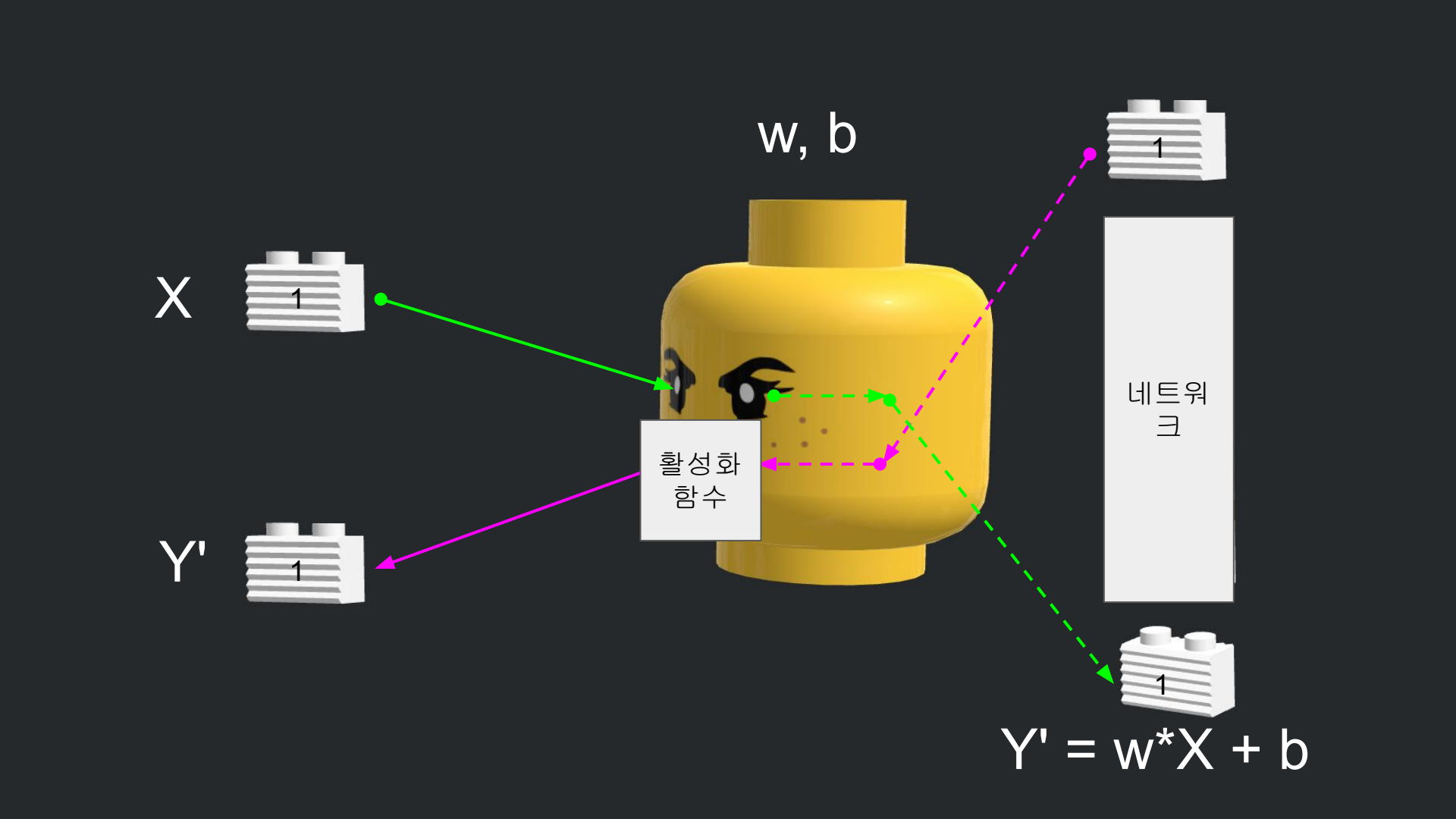
Model
모델

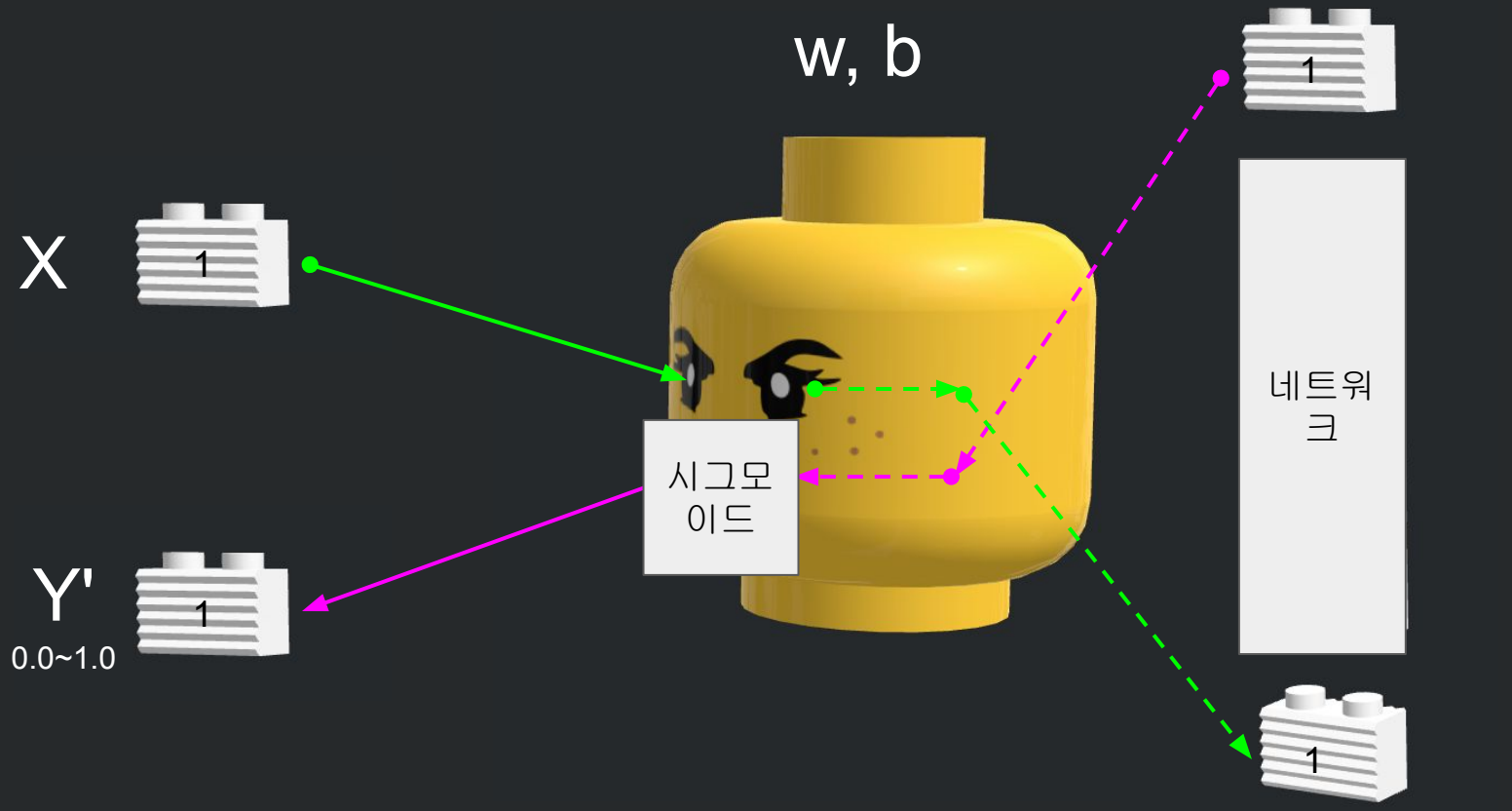
네트워크
network

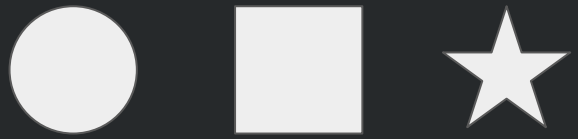
목표함수
objective function

최적화기
optimizer


1
1
1
1
네트워크
1


w, b
X
1
Y'
1
활성화
함수
1
네트워
크
1
Y' = w*X + b


w, b
X
1
Y'
0.0~1.0
1
시그모
이드
1
네트워
크
1
Y' = w*X + b



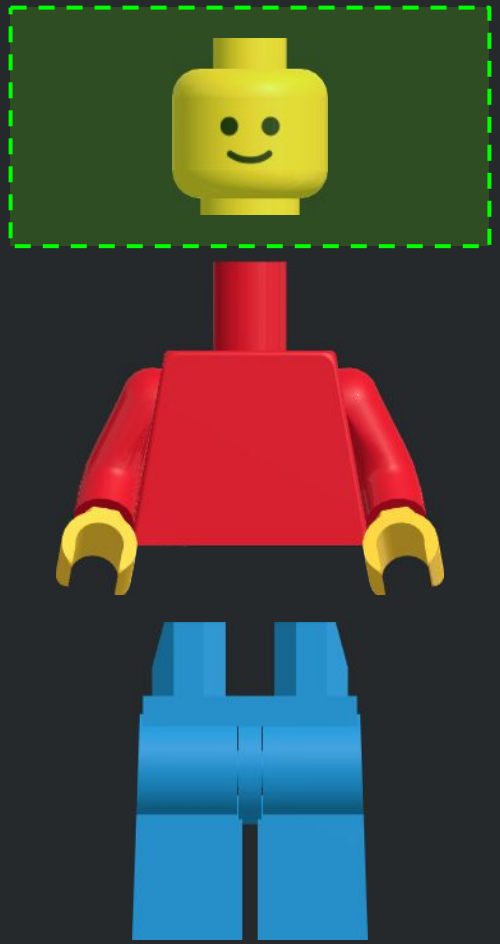
Activation Function
활성화 함수

출력형태를 정한다.

시그모이드(sigmoid) : 출력을 0.0과 1.0사이의 실수

리니어(linear) : 출력이 -무한대 ~ +무한대

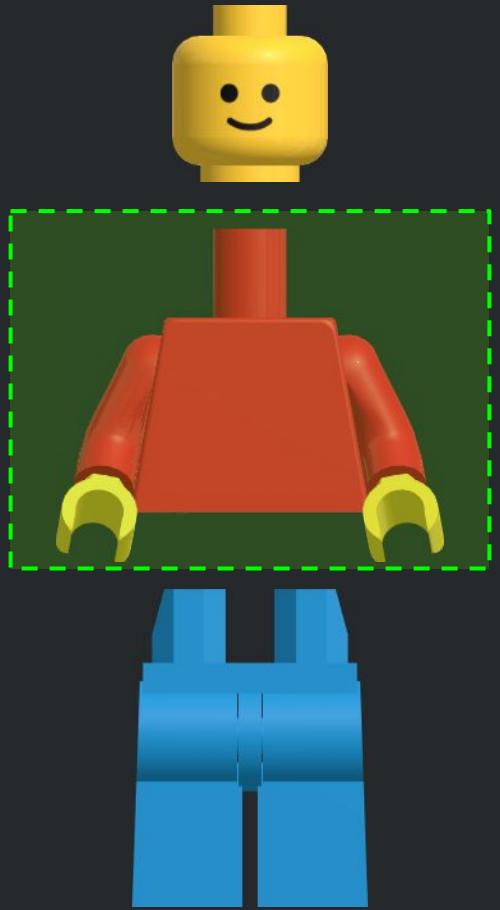
소프트맥스(softmax) :
- 클래스 수 만큼 벡터 수를 결정
- 각 벡터의 값은 0.0과 1.0 사이의 실수
- 벡터들의 그 합은 1.0이다.



https://keras.io/applications/

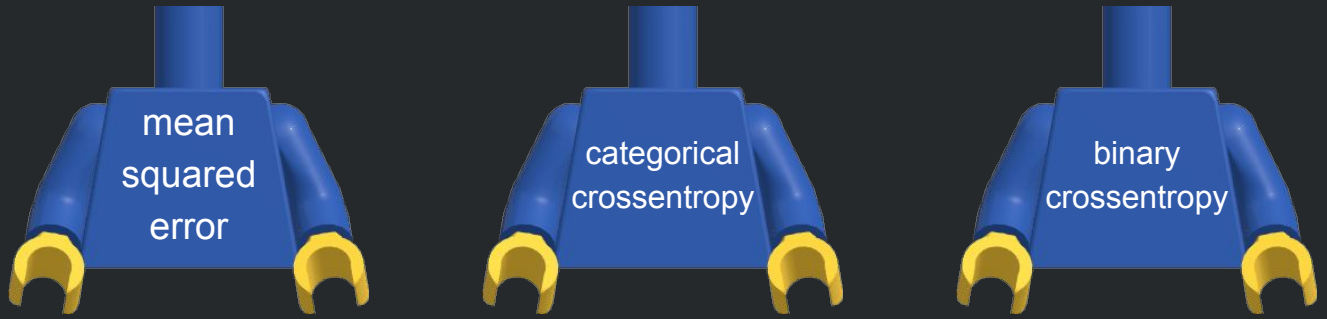
- Xception (88 MB, 126)
- VGG16 (528 MB, 23)
- VGG19 (549 MB, 26)
- ResNet50 (99 MB, 168)
- InceptionV3 (92 MB, 159)
- InceptionResNetV2 (215 MB, 572)
- MobileNet (17 MB, 88)
- DenseNet121 (33 MB, 121)
- DenseNet169 (57 MB, 169)
- DenseNet201 (80 MB, 201)
- NASNet

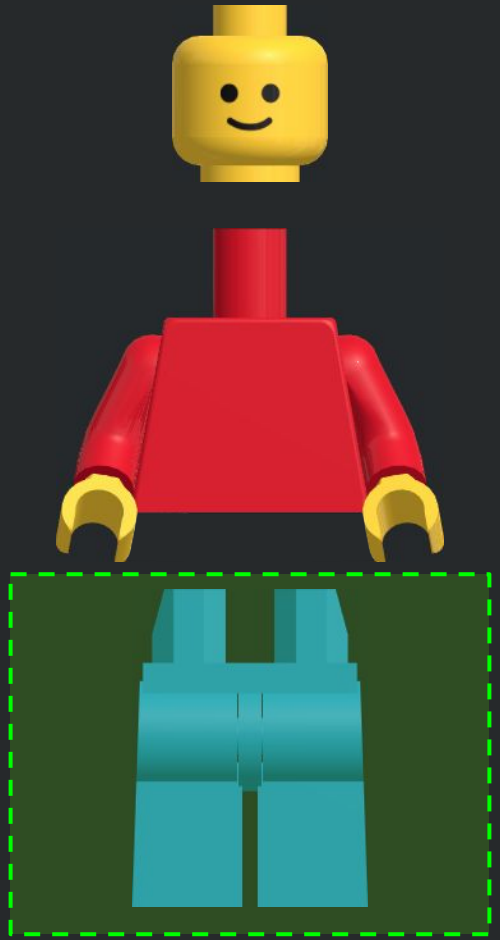

Xception
VGG16
VGG19
ResNet 50
Inception V3
Inception ResNet V2
Mobile Net
Dense Net
NAS Net



https://keras.io/losses/

- mean_squared_error
- mean_absolute_error
- mean_absolute_percentage_error
- mean_squared_logarithmic_error
- squared_hinge
- hinge
- categorical_hinge
- logcosh
- categorical_crossentropy
- sparse_categorical_crossentropy
- binary_crossentropy
- kullback_leibler_divergence
- poisson
- cosine_proximity

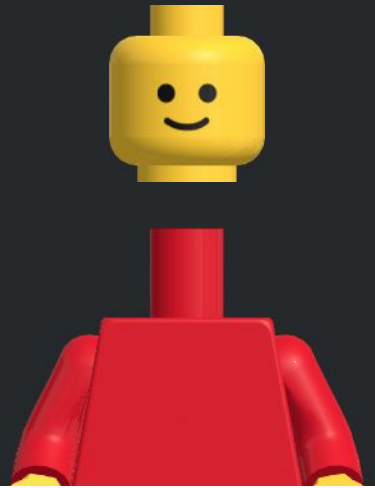

mean
squared
error
categorical
crossentropy
binary
crossentropy



https://keras.io/optimizers/

- SGD
- RMSprop
- Adagrad
- Adadelta
- Adam
- Adamax
- Nadam
- TFOptimizer


SGD
RMSprop
Adagrad
Adam



https://keras.io/optimizers/

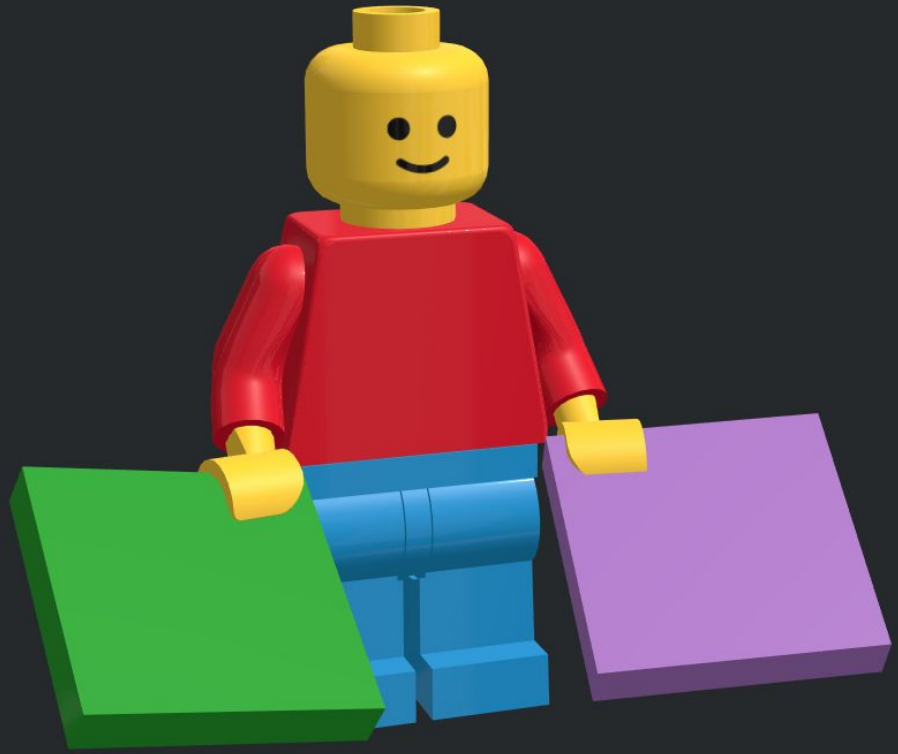
- SGD
- RMSprop
- Adagrad
- Adadelta
- Adam
- Adamax

```
sgd = optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='mean_squared_error', optimizer=sgd)
```

```
# pass optimizer by name: default parameters will be used
model.compile(loss='mean_squared_error', optimizer='sgd')
```

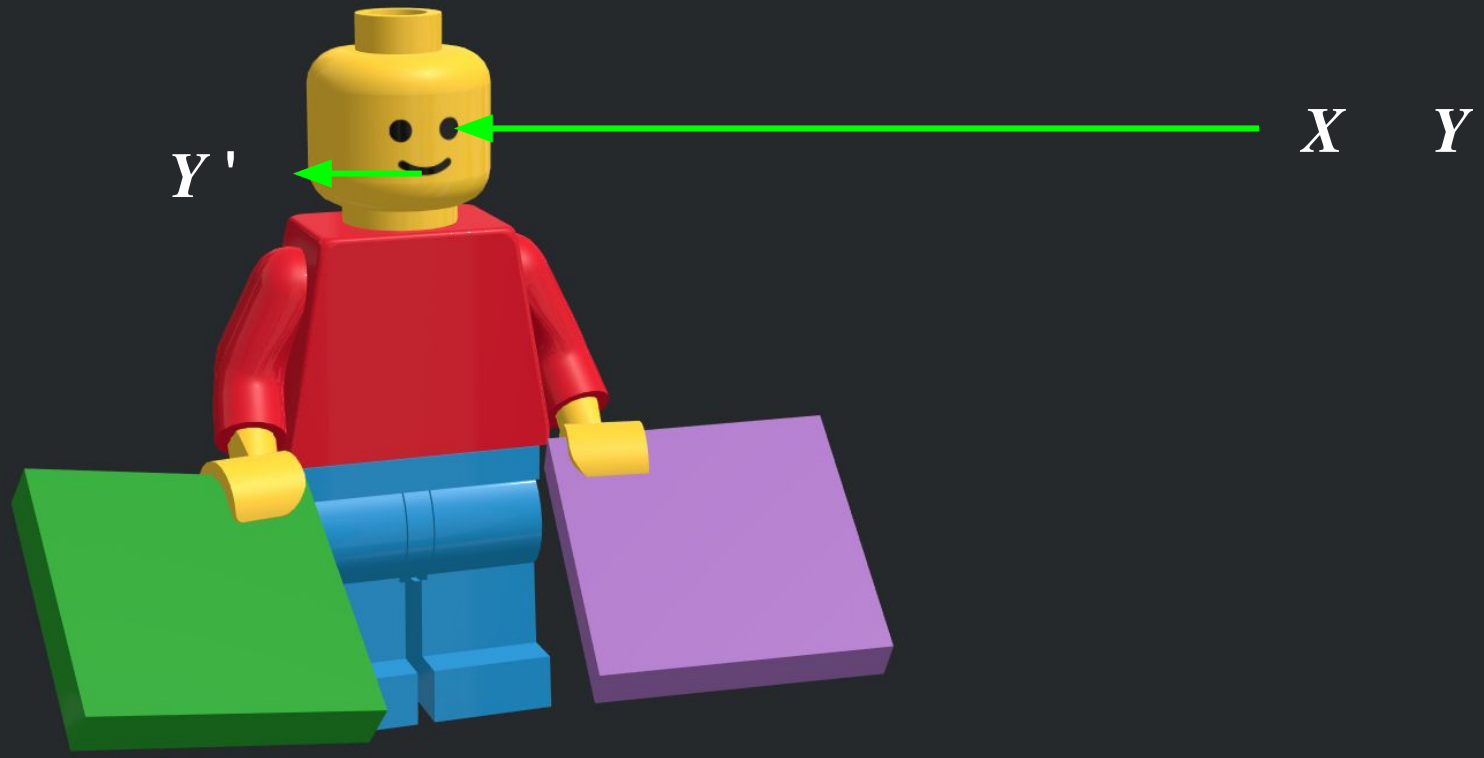

binary
crossentropy
adam


categorical
crossentropy
SGD

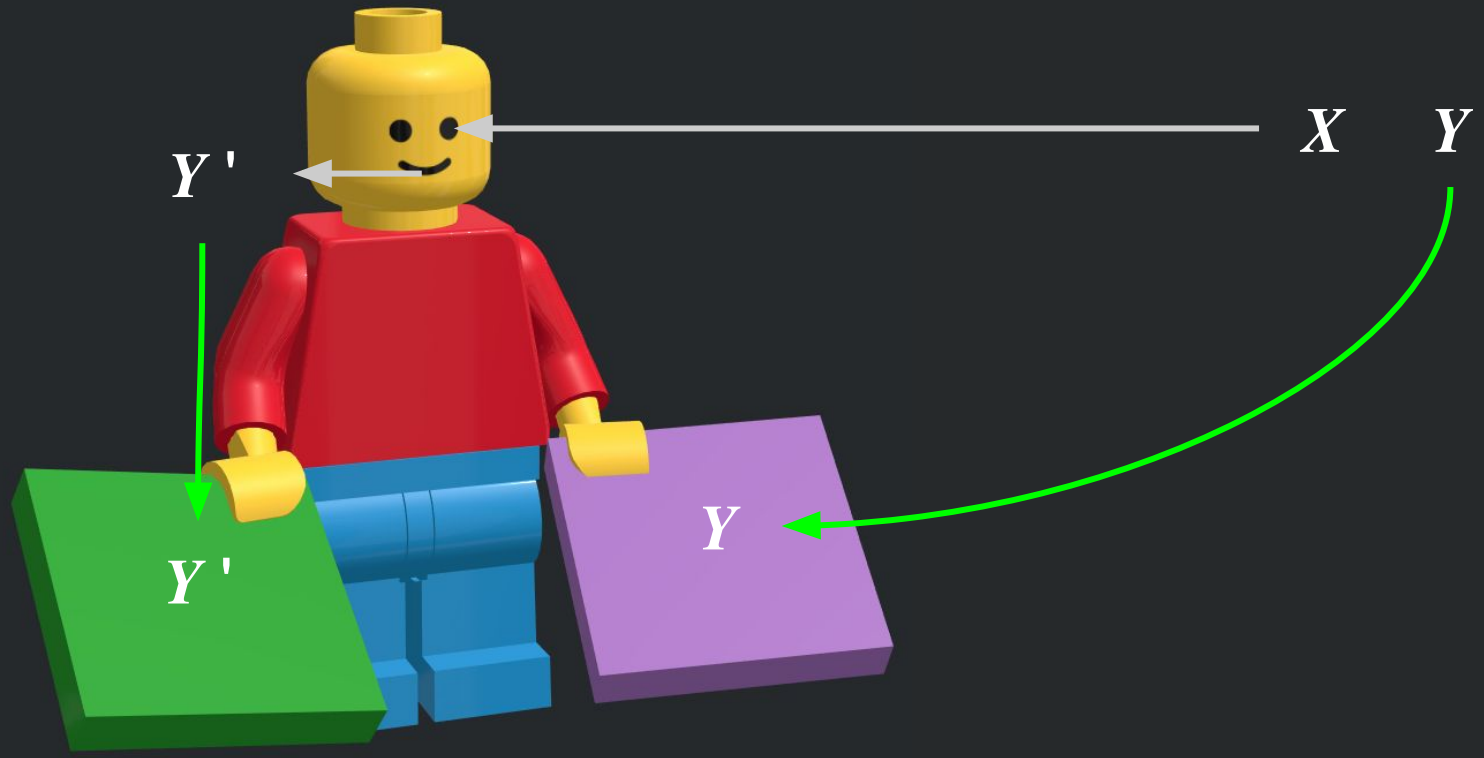


$X \quad Y$

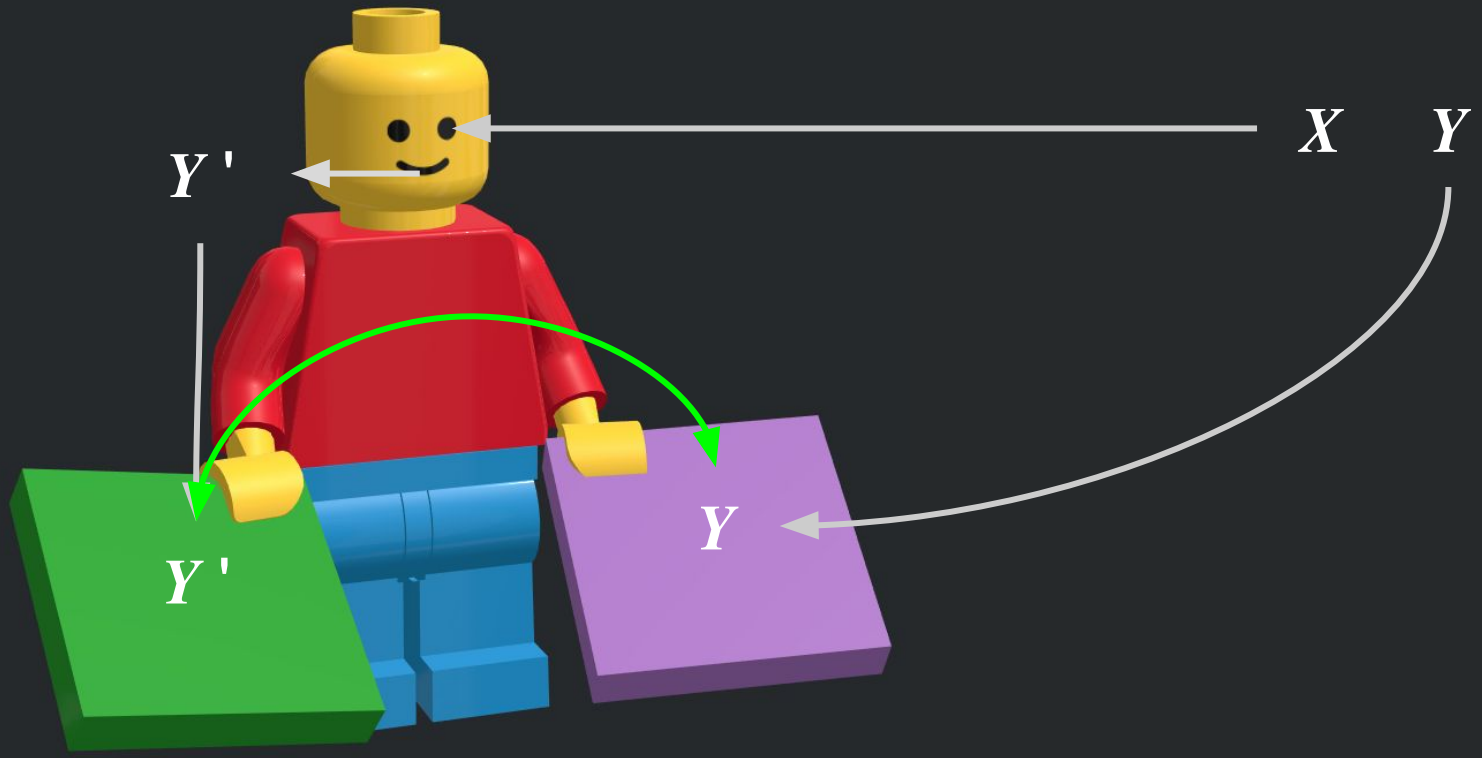
$$Y' = w \times X + b$$



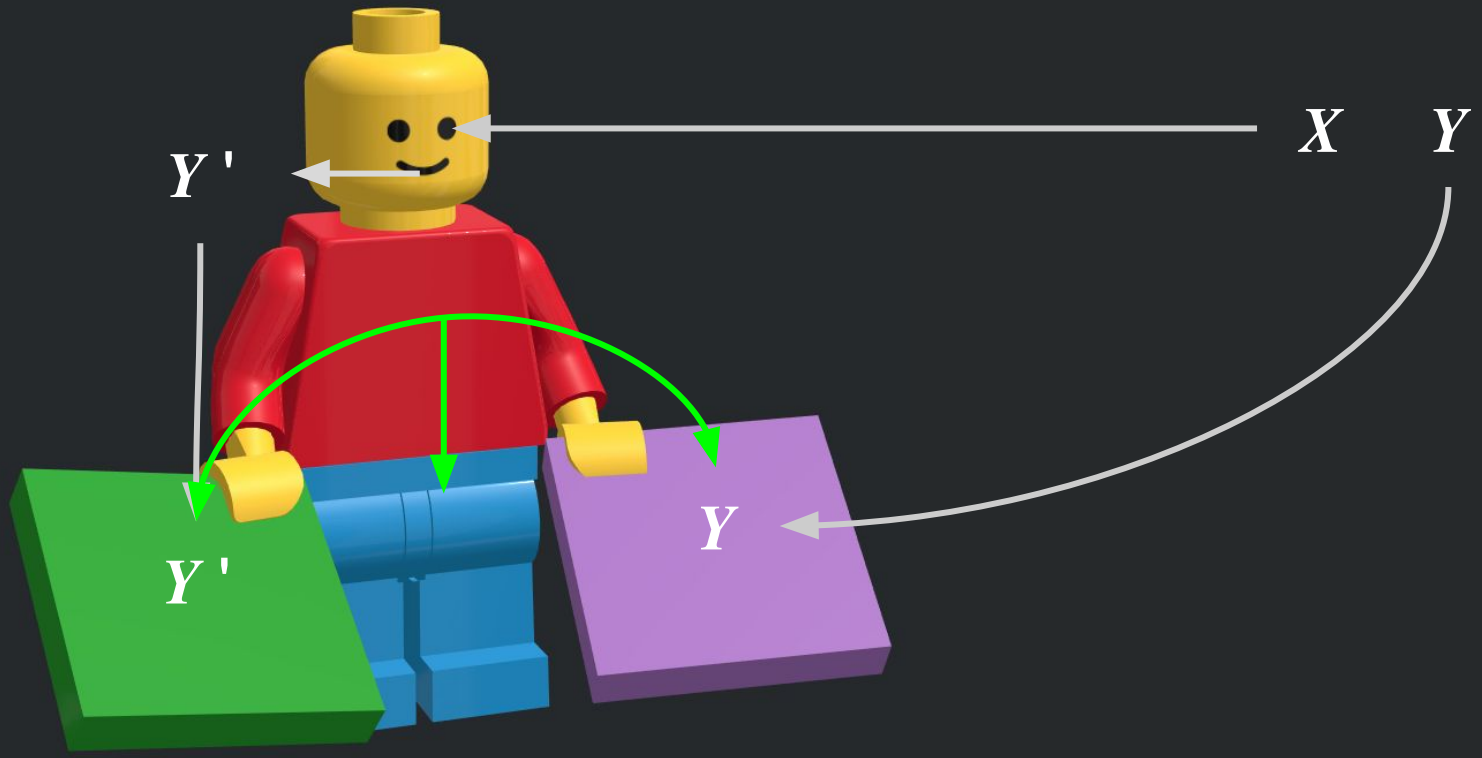
$$Y' = w \times X + b$$



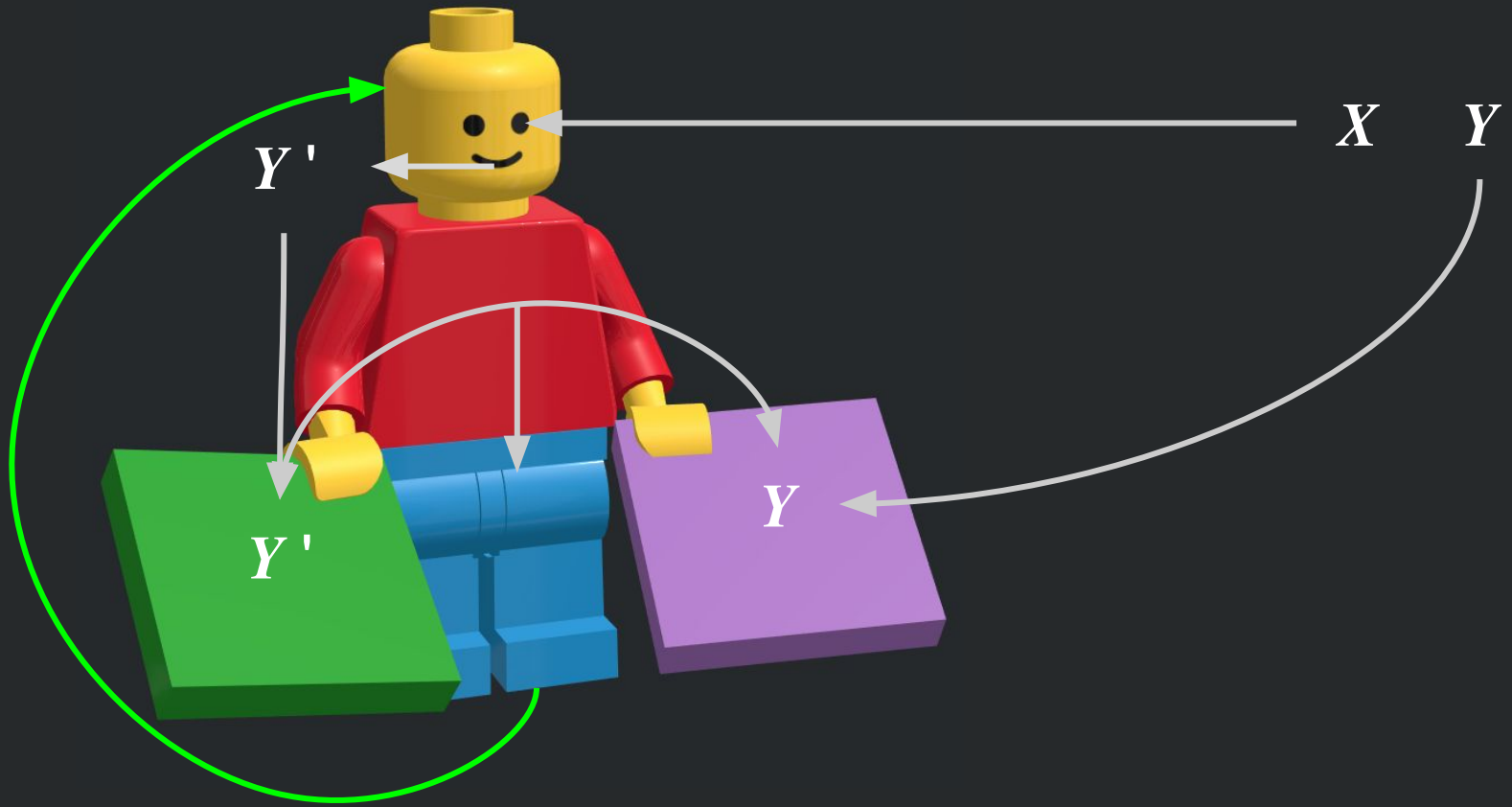
$$Y' = w \times X + b$$



$$Y' = w \times X + b$$



$$Y' = w \times X + b$$



- 데이터셋 준비하기
- 모델 구성하기
- 모델 엮기
- 모델 학습시키기
- 모델 사용하기

```
# 2. 모델 구성하기
model = Sequential()
model.add(Dense(output_dim=64, input_dim=28*28,
activation='relu'))
model.add(Dense(output_dim=10, activation='softmax'))
```

- 시퀀스 모델을 생성한 뒤 필요한 레이어를 추가하여 구성합니다.
- 좀 더 복잡한 모델이 필요할 때는 케라스 함수 API를 사용합니다.

- 데이터셋 준비하기
- 모델 구성하기
- 모델 엮기
- 모델 학습시키기
- 모델 사용하기

```
# 3. 모델 엮기
model.compile(loss='categorical_crossentropy', optimizer='sgd',
metrics=['accuracy'])
```

- 학습하기 전에 학습에 대한 설정을 수행합니다.
- 손실 함수 및 최적화 방법을 정의합니다.
- 케라스에서는 compile() 함수를 사용합니다.

- 데이터셋 준비하기
- 모델 구성하기
- 모델 엮기
- 모델 학습시키기
- 모델 사용하기

```
# 4. 모델 학습시키기
model.fit(X_train, Y_train, nb_epoch=5, batch_size=32)
```

```
Epoch 1/5
60000/60000 [==============================] - 1s - loss: 0.6704 - acc: 0.8259
Epoch 2/5
60000/60000 [==============================] - 1s - loss: 0.3438 - acc: 0.9043
Epoch 3/5
60000/60000 [==============================] - 1s - loss: 0.2959 - acc: 0.9162
Epoch 4/5
60000/60000 [==============================] - 1s - loss: 0.2678 - acc: 0.9239
Epoch 5/5
60000/60000 [==============================] - 1s - loss: 0.2474 - acc: 0.9305
```

- 훈련셋을 이용하여 구성한 모델로 학습시킵니다.
- 케라스에서는 **fit()** 함수를 사용합니다.

- 데이터셋 준비하기
- 모델 구성하기
- 모델 엮기
- 모델 학습시키기
- 모델 사용하기

```
# 5. 모델 사용하기
loss_and_metrics = model.evaluate(X_test, Y_test, batch_size=32)

print('loss_and_metrics : ' + str(loss_and_metrics))
```

```
 8704/10000 [=========================>....] - ETA: 0sloss_and_metrics :
[0.2339999158129096, 0.93320000000000003]
```

- 학습한 모델을 사용합니다.
- 평가를 한다면, 준비된 테스트셋으로 evaluate() 함수를 사용하여 평가합니다.
- 예측을 하고자 한다면 predict() 함수를 사용합니다.

```
from keras.utils import np_utils
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Activation

# 1. 데이터셋 준비하기
(X_train, Y_train), (X_test, Y_test) = mnist.load_data()
X_train = X_train.reshape(60000, 784).astype('float32') / 255.0
X_test = X_test.reshape(10000, 784).astype('float32') / 255.0
Y_train = np_utils.to_categorical(Y_train)
Y_test = np_utils.to_categorical(Y_test)

# 2. 모델 구성하기
model = Sequential()
model.add(Dense(output_dim=64, input_dim=28*28, activation='relu'))
model.add(Dense(output_dim=10, activation='softmax'))

# 3. 모델 엮기
model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])

# 4. 모델 학습시키기
model.fit(X_train, Y_train, nb_epoch=5, batch_size=32)

# 5. 모델 사용하기
loss_and_metrics = model.evaluate(X_test, Y_test, batch_size=32)

print('loss_and_metrics : ' + str(loss_and_metrics))
```

```
import autokeras as ak

clf = ak.ImageClassifier()
clf.fit(x_train, y_train)
results = clf.predict(x_test)
```