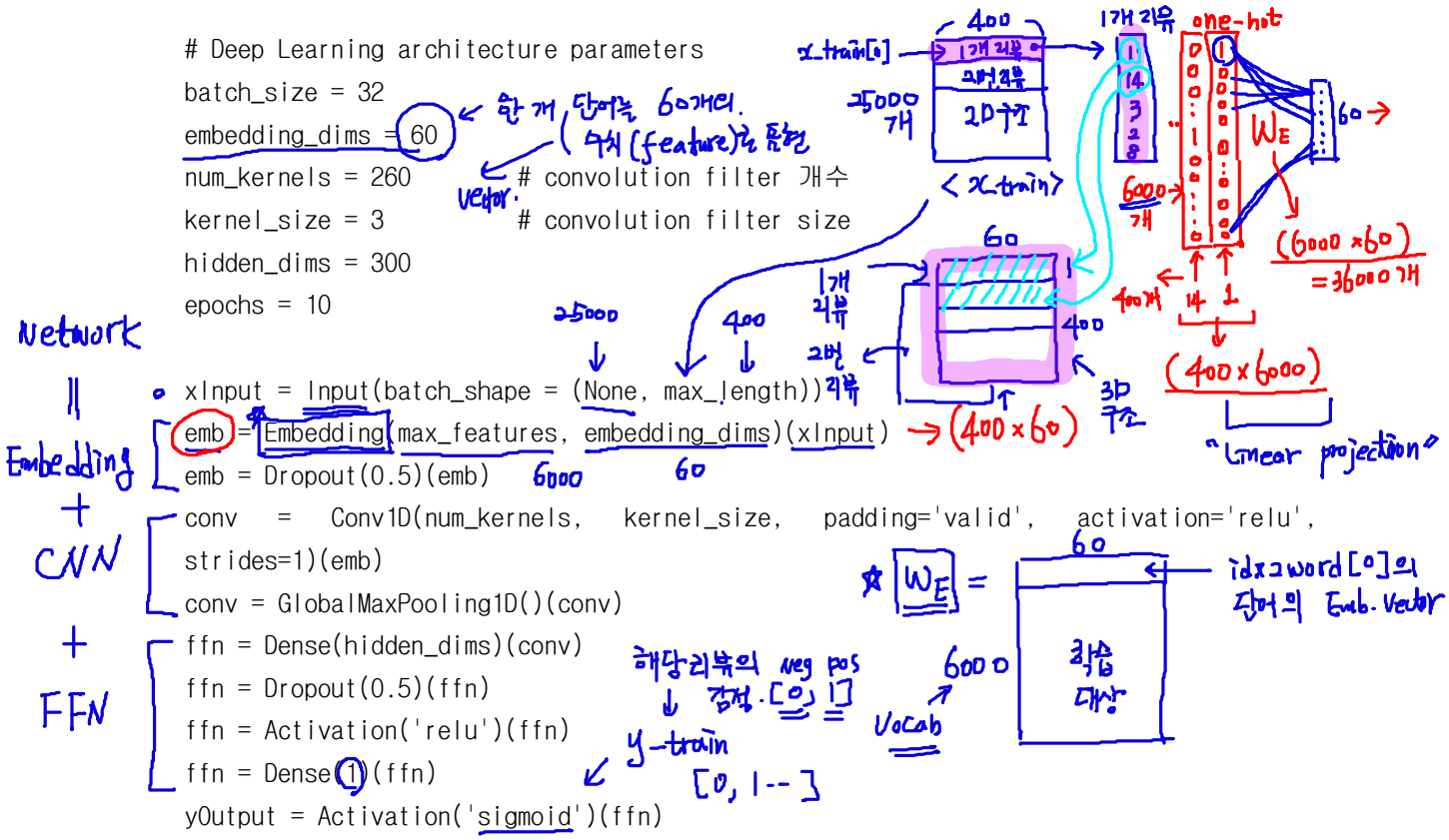
```
# IMDB Classification using Word Embedding and Conv1D
# ----------------------------------------------------
from tensorflow.keras.preprocessing import sequence
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense, Dropout, Activation
from tensorflow.keras.layers import Embedding
from tensorflow.keras.layers import Conv1D, GlobalMaxPooling1D
from tensorflow.keras.datasets import imdb
from sklearn.metrics import accuracy_score
from sklearn.metrics.pairwise import euclidean_distances
import matplotlib.pyplot as plt
import numpy as np

max_features = 6000    # max_features : 최대 단어수
max_length = 400       # 한 개 리뷰 문서의 최대 단어 길이

# IMDB 데이터를 읽어온다.
# IMDB 데이터에 사용된 총 단어의 종류는 88,584개 (vocabulary 크기)이다.
# IMDB 학습데이터와 시험데이터에는 빈도가 높은 단어 6,000개의 index가 표시돼 있다.
# vocabulary의 6,000번째 이후 데이터는 out-of-vocabulary 표시인 '2'가 표시돼 있다.
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)

# 첫 번째 리뷰 문서 x_train[0]의 내용을 확인한다.
# 0 : padding, 1 : start, 2 : OOV, 3 : Invalid를 의미한다.
print(x_train[0])  # 6,000 이하의 word index로 구성돼 있다.

# vocabulary를 생성한다.
# word2idx : {'단어' : idx} 구조
# idx2word : {idx : '단어'} 구조
word2idx = imdb.get_word_index()
idx2word = dict((v,k) for k,v in word2idx.items())

# volcaburary idx는 1부터 시작한다. idx2word[1] = 'the'
# x_train에는 단어들이 vocabulary의 index로 표시돼 있다.
# 그러나 idx2word에는 padding=0, start=1, OOV=2, Invalid=3은 포함돼 있지 않다.
# idx2word의 idx를 3증가 시키고, 아래와 같이 0, 1, 2, 3을 추가한다.
idx2word = dict((v+3, k) for k, v in word2idx.items())
idx2word[0] = '<PAD>'  # padding 문자 표시
idx2word[1] = '<START>'  # start 문자 표시
idx2word[2] = '<OOV>'  # OOV 문자 표시
idx2word[3] = '<INV>'  # Invalid 문자 표시
```

```
word2idx = dict((k, v) for v, k in idx2word.items())

# 숫자로 표시된 x_train을 실제 단어로 변환해서 육안으로 확인해 본다.
# 학습과는 무관하다.
def decode(review):
    x = [idx2word[s] for s in review]
    return ' '.join(x)
decode(x_train[0])
####### 여기까지가 주어진 데이터에 관한 부분이다.

# 1개 리뷰 문서의 단어 개수를 max_length = 400으로 맞춘다.
# 400개 보다 작으면 padding = 0을 추가하고, 400개 보다 크면 뒷 부분을 자른다.
x_train = sequence.pad_sequences(x_train, maxlen=max_length)
x_test = sequence.pad_sequences(x_test, maxlen=max_length)

# Deep Learning architecture parameters
batch_size = 32
embedding_dims = 60
num_kernels = 260
kernel_size = 3
hidden_dims = 300
epochs = 10

xInput = Input(batch_shape = (None, max_length))
emb = Embedding(max_features, embedding_dims)(xInput)
emb = Dropout(0.5)(emb)
conv = Conv1D(num_kernels, kernel_size, padding='valid', activation='relu',
strides=1)(emb)
conv = GlobalMaxPooling1D()(conv)
ffn = Dense(hidden_dims)(conv)
ffn = Dropout(0.5)(ffn)
ffn = Activation('relu')(ffn)
ffn = Dense(1)(ffn)
yOutput = Activation('sigmoid')(ffn)

model = Model(xInput, yOutput)
model.compile(loss='binary_crossentropy', optimizer='adam')
print(model.summary())

# 학습
hist = model.fit(x_train, y_train,
```



```
                    batch_size=batch_size,
                    epochs=epochs,
                    validation_data = (x_test, y_test))

# Loss history를 그린다
plt.plot(hist.history['loss'], label='Train loss')
plt.plot(hist.history['val_loss'], label = 'Test loss')
plt.legend()
plt.title("Loss history")
plt.xlabel("epoch")
plt.ylabel("loss")
plt.show()

# 성능 확인
y_pred = model.predict(x_test)
y_pred = np.where(y_pred > 0.5, 1, 0)
print ("Test accuracy:", accuracy_score(y_test, y_pred))

# 1. 특정 단어의 Embedding vector를 확인한다.
# ----------------------------------------
# Embedding layer의 W를 읽어온다.
# 이것이 6,000개 단어에 대한 word embedding vector가 된다.
w_emb = np.array(model.layers[1].get_weights())       # shape = (1, 6000, 60)
w_emb = w_emb.reshape(max_features, embedding_dims)   # shape = (6000, 60)

# father - mother - daughter - son 간의 거리를 측정한다.
father = w_emb[word2idx['father']]
mother = w_emb[word2idx['mother']]
daughter = w_emb[word2idx['daughter']]
son = w_emb[word2idx['son']]
euclidean_distances([father, mother, daughter, son])

# 2. 특정 문장의 Embedding vector를 확인한다.
# ----------------------------------------
embModel = Model(xInput, emb)
m = embModel.predict(x_train[0].reshape(1, max_length))
m.shape
```

80~90 %

336

60

6000

WE

father

수치 vector

거리, 유사도 계산.

Embedding의 출력

첫번째 리뷰.

king - man =

king

man

women

queen

60차원.

Vector 연산 가능

60

400

첫 번째 리뷰의 첫 번째 단어의 vector.

단점 1. OOV 문제

2. bank : 은행, bank~ 등 > 모두 동일한 vector로 표현함.