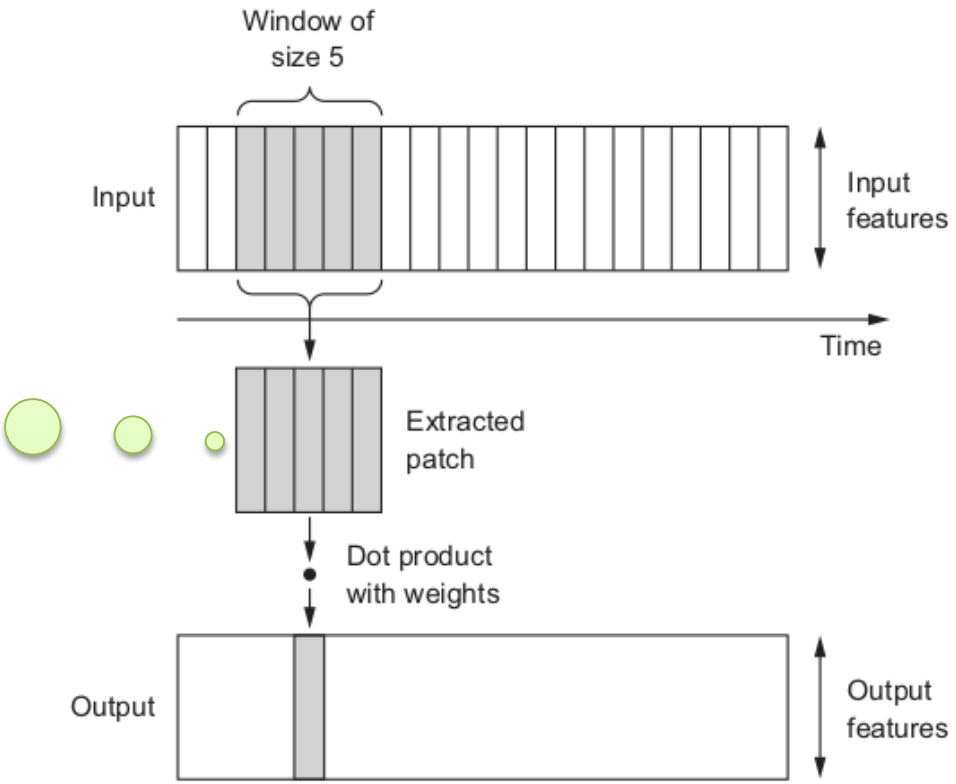
- Konvolüsyon ile 1 boyutlu dizi işleme
- 1D convnet
- Keras Functional API

1D convnet

- 1D convnet ile belirli pencere büyüklüğünde seçilen parçalar ile lokal özellikler çıkarılabilmektedir.
- Böylece bir parça üzerinden öğrenilen özellik dizinin başka bir yerinde ortaya çıktığı zaman da tanınabilir.
- 1D convnet'ler bazı dizi işleme problemlerinde RNN ile birlikte kullanıldıklarında güçlü modeller oluşturabilirler.

1D convnet

Dizi üzerinde hareket edecek pencere



Kaynak: Deep Learning with Python by Francois Chollet.

1D convnet

```
11 X1=np.array((range(10)),dtype=float)
12 model = Sequential()
13 model.add(layers.Conv1D(1,
14                          3,
15                          input_shape=(10,1),
16                          activation='relu'))
17 model.summary()
18
19 w=model.get_weights()
20 w[0][0]=1.0
21 w[0][1]=1.0
22 w[0][2]=1.0
23 model.set_weights(w)
24
25 print(X1)
26 y=model.predict(X1.reshape(1,10,1))
27 print(y.reshape(1,8))
```

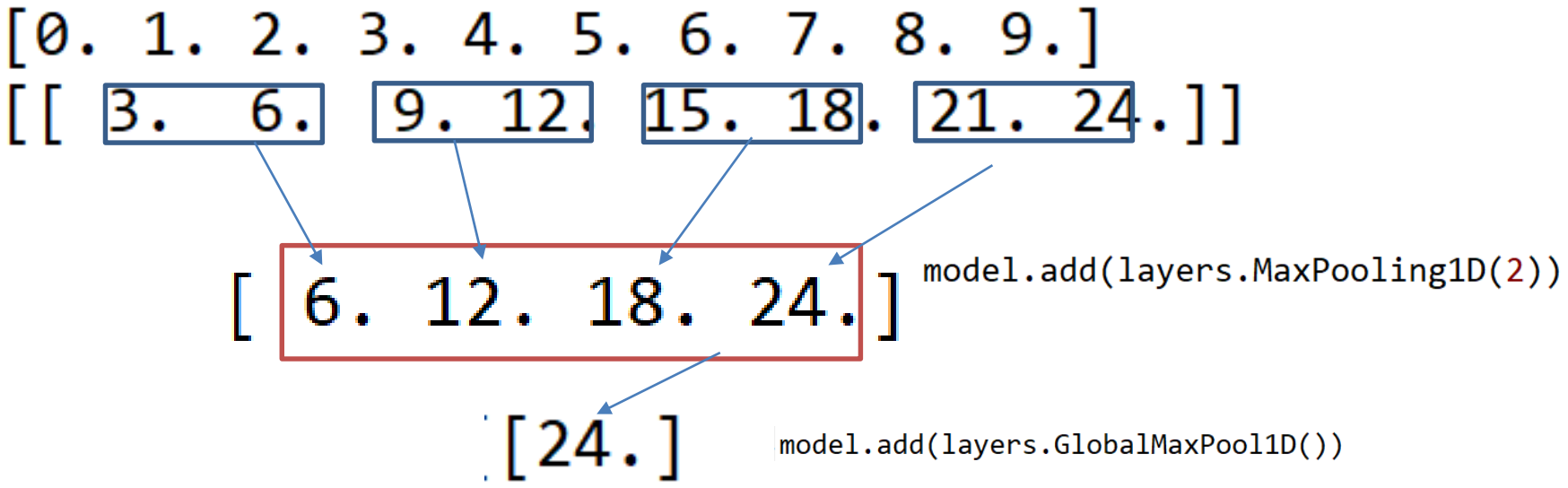
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_64 (Conv1D)           (None, 8, 1)              4
=================================================================
Total params: 4
Trainable params: 4
Non-trainable params: 0
_________________________________________________________________
```

Konvolüsyon katmanının çalşmasını test etmek için tek katmanlı bir ağ oluşturduk ve ağırlıklarını 1.0 yaptık. (bias=0,default)

```
[0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
[[ 3.  6.  9. 12. 15. 18. 21. 24.]]
```

1D pooling for sequence data

- Max pooling
- Global max pooling
- Average pooling
- Global average pooling



Örnek: imdb üzerinde convnet

```
from keras.datasets import imdb
from keras.preprocessing import sequence

max_features = 10000  # number of words to consider as features
max_len = 500  # cut texts after this number of words (among top max_features m

print('Loading data...')
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)
print(len(x_train), 'train sequences')
print(len(x_test), 'test sequences')

print('Pad sequences (samples x time)')
x_train = sequence.pad_sequences(x_train, maxlen=max_len)
x_test = sequence.pad_sequences(x_test, maxlen=max_len)
print('x_train shape:', x_train.shape)
print('x_test shape:', x_test.shape)
```

```
model = Sequential()
model.add(layers.Embedding(max_features,
                           128,
                           input_length=max_len))

model.add(layers.Conv1D(32, 7, activation='relu'))

model.add(layers.MaxPooling1D(5))

model.add(layers.Conv1D(32, 7, activation='relu'))

model.add(layers.GlobalMaxPooling1D())

model.add(layers.Dense(1))

model.summary()
```

Kaynak: https://github.com/fchollet/deep-learning-with-python-notebooks/blob/master/6.4-sequence-processing-with-convnets.ipynb

Örnek: Eğitilebilir (trainable) parametreler

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 500, 128)	1280000
conv1d_1 (Conv1D)	(None, 494, 32)	28704
max_pooling1d_1 (MaxPooling1	(None, 98, 32)	0
conv1d_2 (Conv1D)	(None, 92, 32)	7200
global_max_pooling1d_1 (Glob	(None, 32)	0
dense_1 (Dense)	(None, 1)	33

Total params: 1,315,937
Trainable params: 1,315,937
Non-trainable params: 0

- Kullanılan ağ yapısında giriş verisi 500 kelime ile sınırlanmış, her biri 128 elemanlı embedding vektör ile tanımlanmış yorumlardan oluşmaktadır.
- Toplam 10000 kelime kullanıldığı için giriş katmanında eğitilmesi gereken 128*10000 adet parametre bulunmaktadır.
- Ağın girişine 500 kelimelik bir yoruma ait indisler vektörü uygulandığında çıkışta her bir kelimeye karşılık gelen 128 elemanlı vektörler elde edilir. Dolayısıyla Embedding katmanın çıkışında her biri 500 elemanlı 128 adet veri dizisi elde edilir.
- Konvolüsyon katmanında 32 adet konvolüsyon uygulandığını düşünürsek toplam 32*7 eğitilebilir parametre olması gerekir. Ancak veri dizisi 500*128 boyutlu olduğu için biaslar ile birlikte toplam 32*7*128+32=28704 parametrenin eğitilmesi gerektiği ortaya çıkar.

Örnek: Eğitilebilir (trainable) parametreler

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 500, 128)	1280000
conv1d_1 (Conv1D)	(None, 494, 32)	28704
max_pooling1d_1 (MaxPooling1	(None, 98, 32)	0
conv1d_2 (Conv1D)	(None, 92, 32)	7200
global_max_pooling1d_1 (Glob	(None, 32)	0
dense_1 (Dense)	(None, 1)	33

Total params: 1,315,937
Trainable params: 1,315,937
Non-trainable params: 0

- Konvolüsyon katmanının çıkışında veri dizisinin ilk 3 ve son 3 elemanı için filtreleme gerçekleşmez (default olarak padding='valid') ve konvolüsyon işleminden elde edilen 32 çıkış bulunduğu için çıkış dizisinin boyutu 494x32 olarak elde edilir.
- İlk MaxPooling katmanında ise pooling boyutu 5 seçildiği için, veri dizisinden 5 eleman arayla maksimum değerler seçilerek 494/5=98 elemanlı yeni bir dizi oluşturulur.
- İkinci konvolüsyon katmanına 98x32 boyutuna düşerek gelen veri birinci katmandaki gibi 7 uzunluklu 32 adet konvolüsyon işlemine tabi tutulur ve çıkışında 92x32 boyutlu bir dizi elde edilir. Bu katmandaki eğitilebilir parametre sayısı=32*7*32+32=7200 olarak ortaya çıkar.
- Global Max Pooling ile son olarak elde edilen 92 elemanlı dizilerin maksimum değerleri seçilerek giriş dizisi 32 elemanlı bir diziye indirgenir.
- Ağın son katmanı ise 32 girişli tek çıkışlı bir Dense Layer olduğu için toplam 32+1=33 eğitilebilir parametre bulunmaktadır.

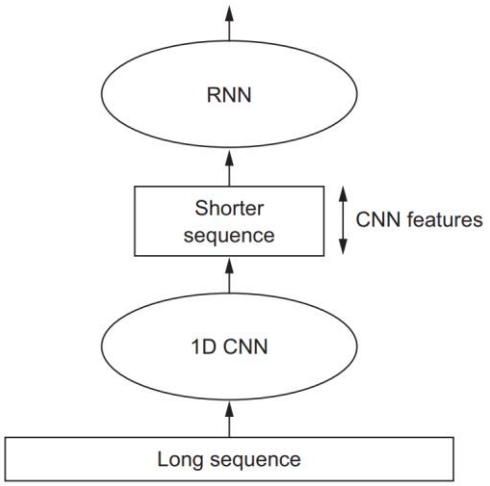
CNN ve RNN katmanlarının uzun dizilerin işlenmesi için birleştirilmesi

- RNN katmanı öncesinde, 1D convnet katmanları ile önişleme yapılarak özellikler çıkartılır.
- 2D convnet'de olduğu gibi katman sayısı arttıkça daha yüksek seviyeli özellikler çıkartılır ve dizi boyutu RNN ile işlenecek makul boyutta olacak şekilde küçültülür

```
model = Sequential()
model.add(layers.Conv1D(32, 5, activation='relu',
                        input_shape=(None, float_data.shape[-1])))
model.add(layers.MaxPooling1D(3))
model.add(layers.Conv1D(32, 5, activation='relu'))
model.add(layers.GRU(32, dropout=0.1, recurrent_dropout=0.5))
model.add(layers.Dense(1))

model.summary()

model.compile(optimizer=RMSprop(), loss='mae')
history = model.fit_generator(train_gen,
                              steps_per_epoch=500,
                              epochs=20,
                              validation_data=val_gen,
                              validation_steps=val_steps)
```



Keras Functional API

Sequential Model

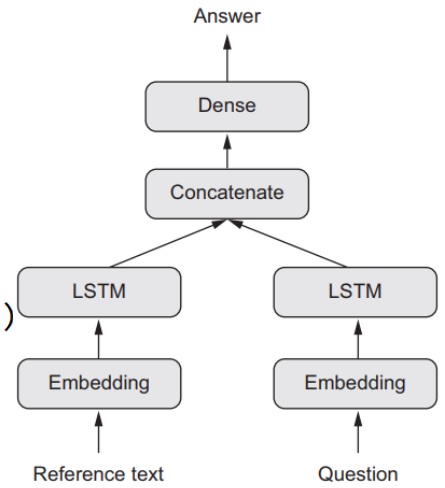
```
12 model = Sequential()
13 model.add(layers.Dense(32, activation='relu', input_shape=(64,)))
14 model.add(layers.Dense(32, activation='relu'))
15 model.add(layers.Dense(10, activation='softmax'))
```

Yukarıdaki programın Functional API eşdeğeri

```
17 input_tensor = Input(shape=(64,))
18 x = layers.Dense(32, activation='relu')(input_tensor)
19 x = layers.Dense(32, activation='relu')(x)
20 output_tensor = layers.Dense(10, activation='softmax')(x)
21 model = Model(input_tensor, output_tensor)
```

Çok girişli model

```
14 #Referans metin giriş katmanı
15 text_input = Input(shape=(None,), dtype='int32', name='text')
16
17 embedded_text = layers.Embedding( 64, text_vocabulary_size)(text_input)
18
19 encoded_text = layers.LSTM(32)(embedded_text)
20
21 #Soru girişi katmanı
22 question_input = Input(shape=(None,),dtype='int32',name='question')
23
24 embedded_question = layers.Embedding(32, question_vocabulary_size)(question_input)
25
26 encoded_question = layers.LSTM(16)(embedded_question)
27
28 #İki katmanı birleştir
29 concatenated = layers.concatenate([encoded_text, encoded_question],axis=-1)
30
31 answer = layers.Dense(answer_vocabulary_size,activation='softmax')(concatenated)
32
33 model = Model([text_input, question_input], answer)
```



Çok çıkışlı model

```
from keras import layers
from keras import Input
from keras.models import Model
vocabulary_size = 50000
num_income_groups = 10
#giriş
posts_input = Input(shape=(None,), dtype='int32', name='posts')
#Embedding layer: 50000 kelime, 256 elemanlı embedding vektör
embedded_posts = layers.Embedding(256, vocabulary_size)(posts_input)
#konvolüsyon katmanı
x = layers.Conv1D(128, 5, activation='relu')(embedded_posts)
x = layers.MaxPooling1D(5)(x)
x = layers.Conv1D(256, 5, activation='relu')(x)
x = layers.Conv1D(256, 5, activation='relu')(x)
x = layers.MaxPooling1D(5)(x)
x = layers.Conv1D(256, 5, activation='relu')(x)
x = layers.Conv1D(256, 5, activation='relu')(x)
x = layers.GlobalMaxPooling1D()(x)
x = layers.Dense(128, activation='relu')(x)
#yaş tahmini
age_prediction = layers.Dense(1, name='age')(x)
#gelir tahmini
income_prediction = layers.Dense(num_income_groups,
                                 activation='softmax',name='income')(x)
#cinsiyet tahmini
gender_prediction = layers.Dense(1,
                                 activation='sigmoid', name='gender')(x)
#modeli oluştur
model = Model(posts_input,[age_prediction, income_prediction, gender_prediction])
```