Hafta 08 - Sinir Ağları - 2

BGM 565 - Siber Güvenlik için Makine Öğrenme Yöntemleri

Bilgi Güvenliği Mühendisliği
Yüksek Lisans Programı

Dr. Ferhat Özgür Çatak
ozgur.catak@tubitak.gov.tr

İstanbul Şehir Üniversitesi
2018 - Bahar

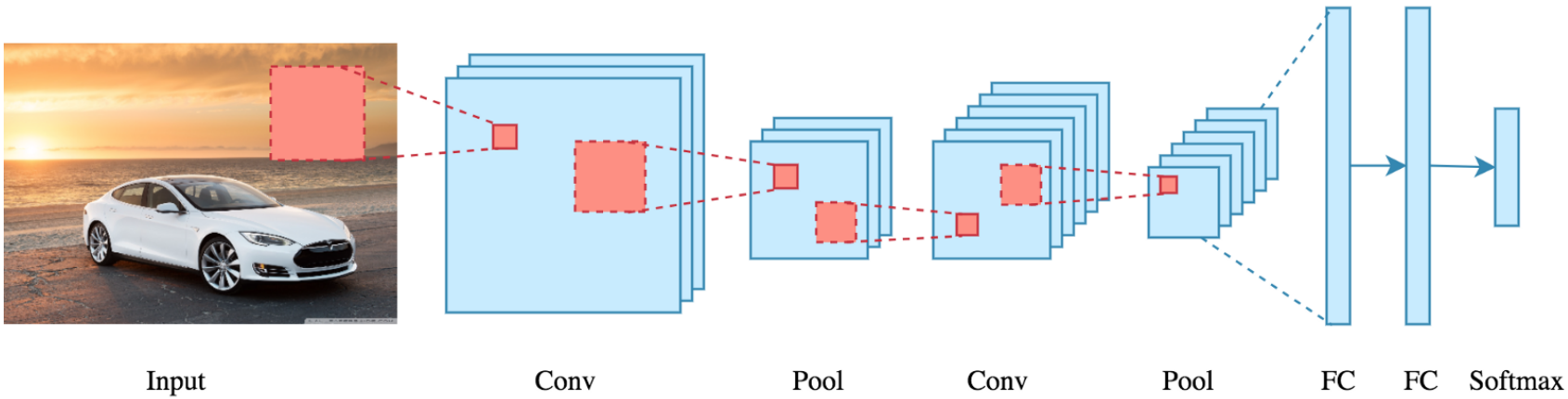
İçindekiler

İçindekiler

Evrişimli Sinir Ağları I
Convolutional Neural Networks

Genel Mimari

- Sırayla *convolution* ve *pooling* işlemleri gerçekleştirilir.
- *Fully connected layers*



Convolution

Convolution

- Main building block of CNN: convolutional layer
- **Convolution**: mathematical operation to merge two sets of information
- Convolution is applied on the input data using a *convolution filter* to produce a *feature map*.

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

Input

1	0	1
0	1	0
1	0	1

Filter / Kernel

Convolution

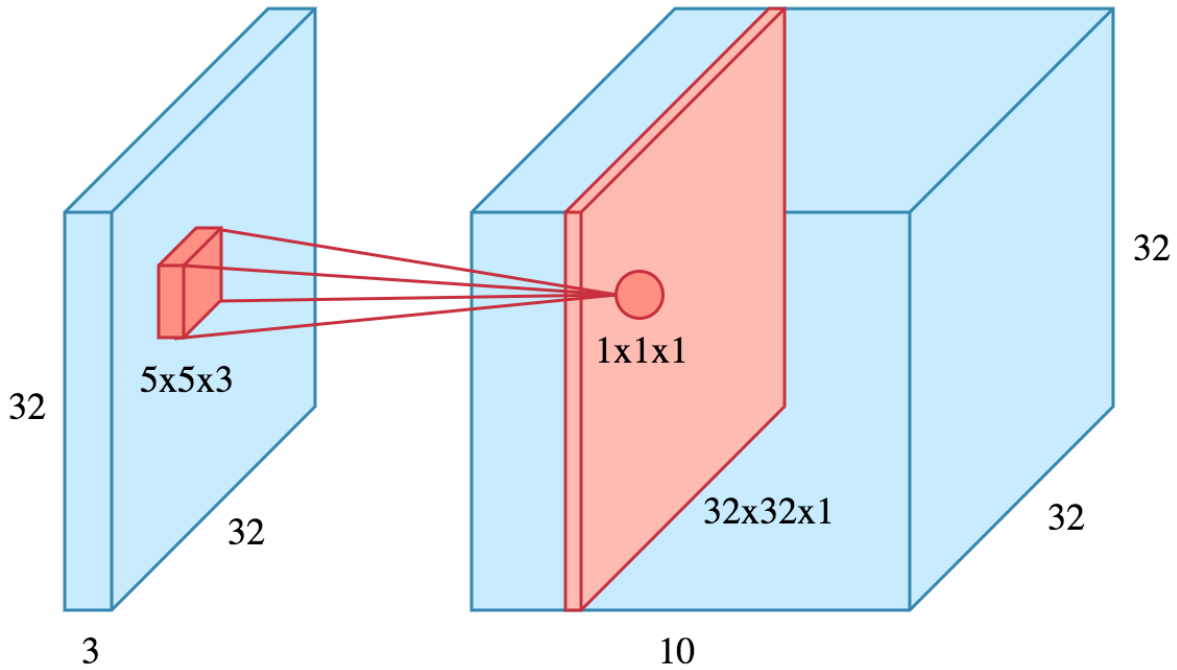
1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

4		

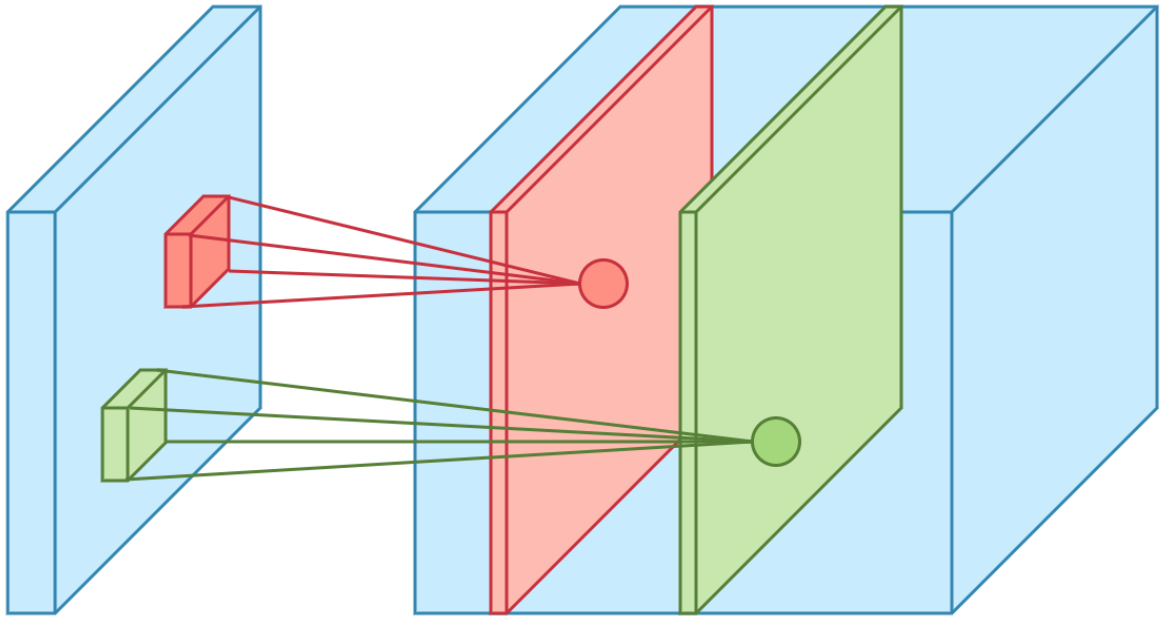
Convolution I

Convolution

- Convolution 3-boyutlu olarak hesaplanır.
- Resimler 3-boyutlu olarak gösterilmektedir. (height, width, depth)
  - depth: color channels (**RGB**)
- **Örnek**: $32 \times 32 \times 3$ resim, $5 \times 5 \times 3$ filtre boyutu



Convolution II



Non-linearity

Non-linearity

- ANN ve AutoEncoders ağırlıklı toplam ifadesi aktivasyon fonksiyonları kullanır.
- CNN'de aynı şekilde **aktivasyon fonksiyonları** kullanacaktır.

Stride ve Padding

Stride

- Stride specifies how much we move the convolution filter at each step.
- By default the value is 1

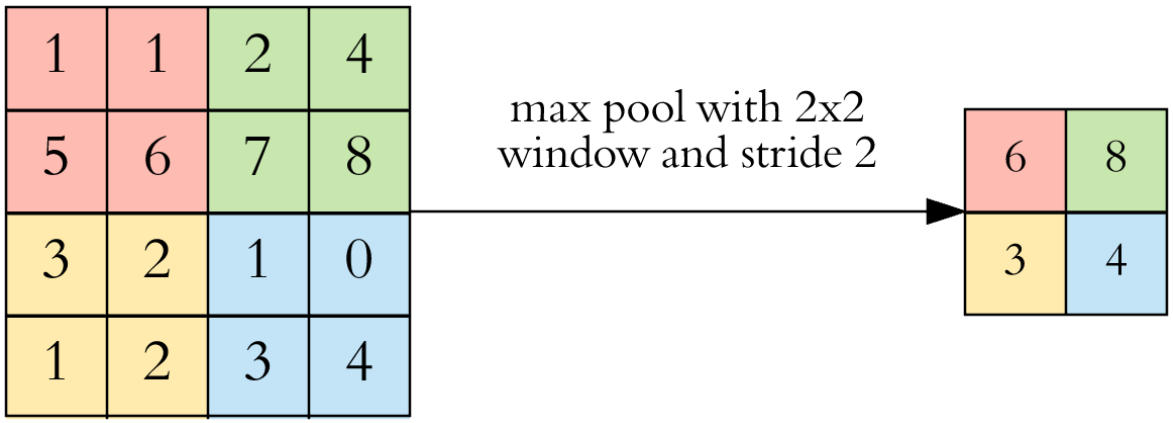
Padding

- Input map ve feature map aynı olması isteniyorsa, 0'lardan oluşan değerler eklenir.

Pooling I

Pooling

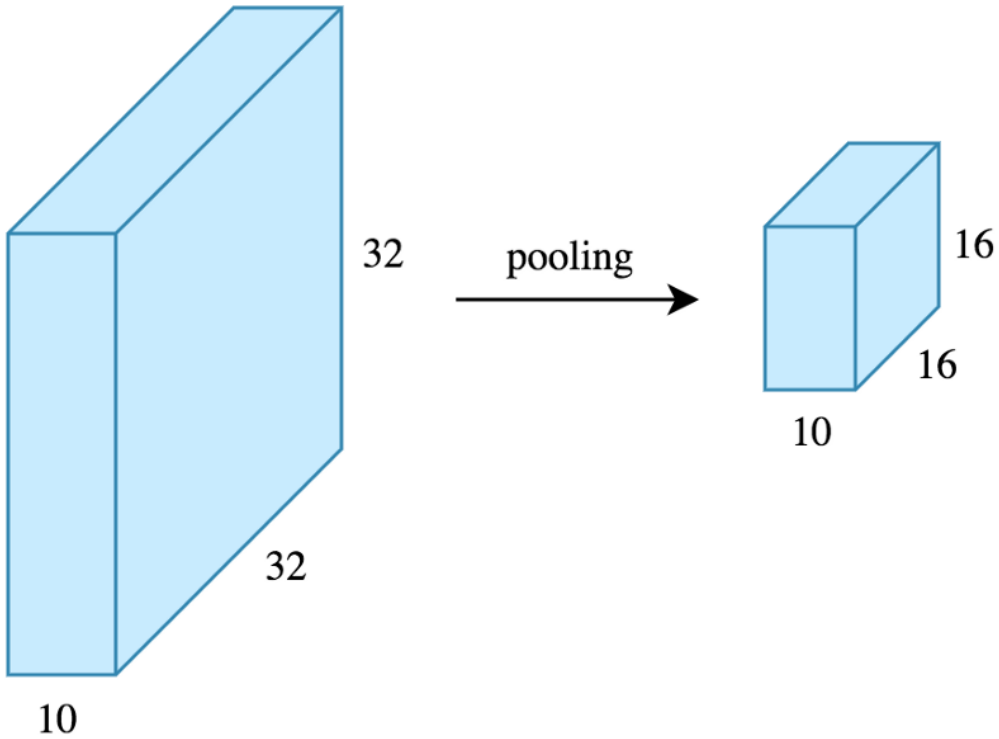
- **Convolution** işleminden sonra boyut azaltmak amacıyla **pooling** işlemi yapılır.
- **Amaç**: Eğitim zamanını azaltmak ve aşırı öğrenmeyi (overfitting) engellemek.
- Pooling katmanları her bir feature map üzerinden örnekler almaktadır.
- En çok kullanılan pooling: *max pooling*



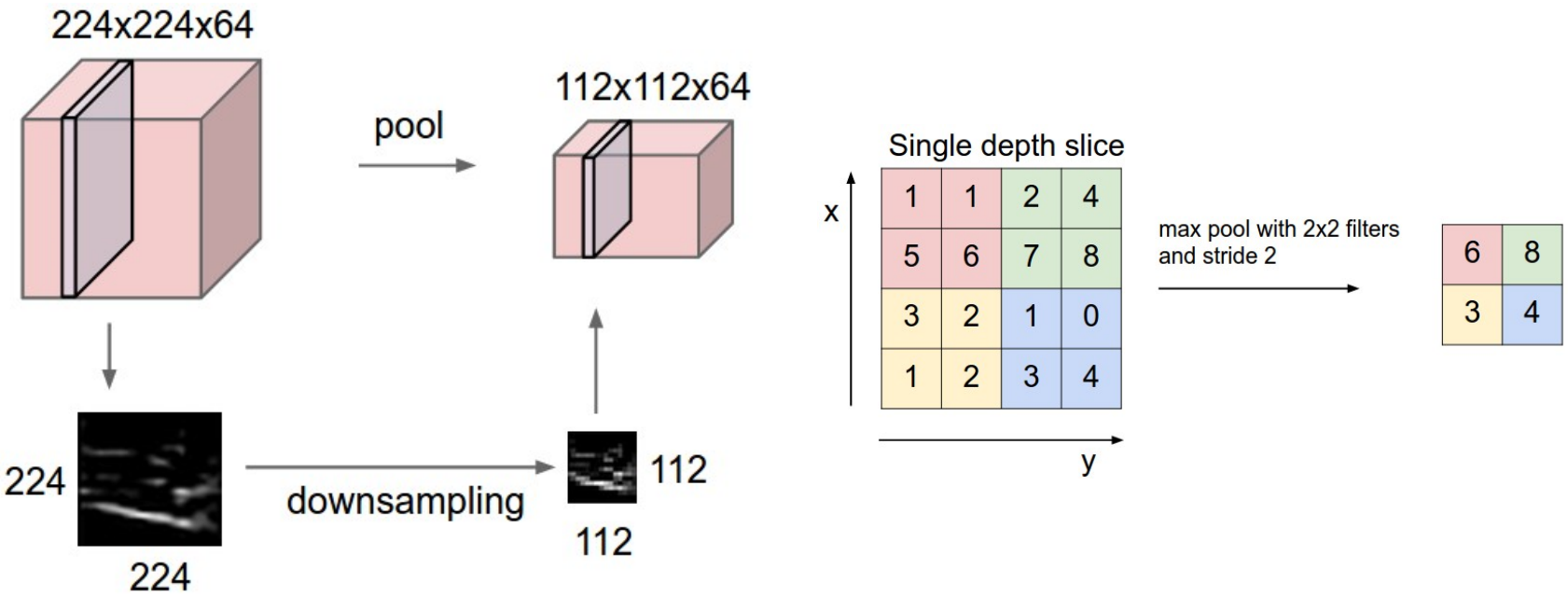
Pooling II

Örnek

- Pooling layer input: $32 \times 32 \times 10$
- max pooling: 2×2 window



Pooling III


224x224x64
pool
112x112x64
224
224
downsampling
112
112
Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

Hyperparameters

Hyperparameters

- *Filter size*: genellikle $3 \times 3, 5 \times 5, 7 \times 7$. Bu filtreler 3-boyutludur. Bütün katmanlarda aynı olması sebebiyle genellikle gösterilmez.
- *Filter count*: 2'nin katları şeklindedir. [32, 1024]. Filtre sayısı artıkça daha güçlü ama aşırı öğrenmeye daha yatkın model oluşur.
- *Stride*
- *Padding*

Fully Connected

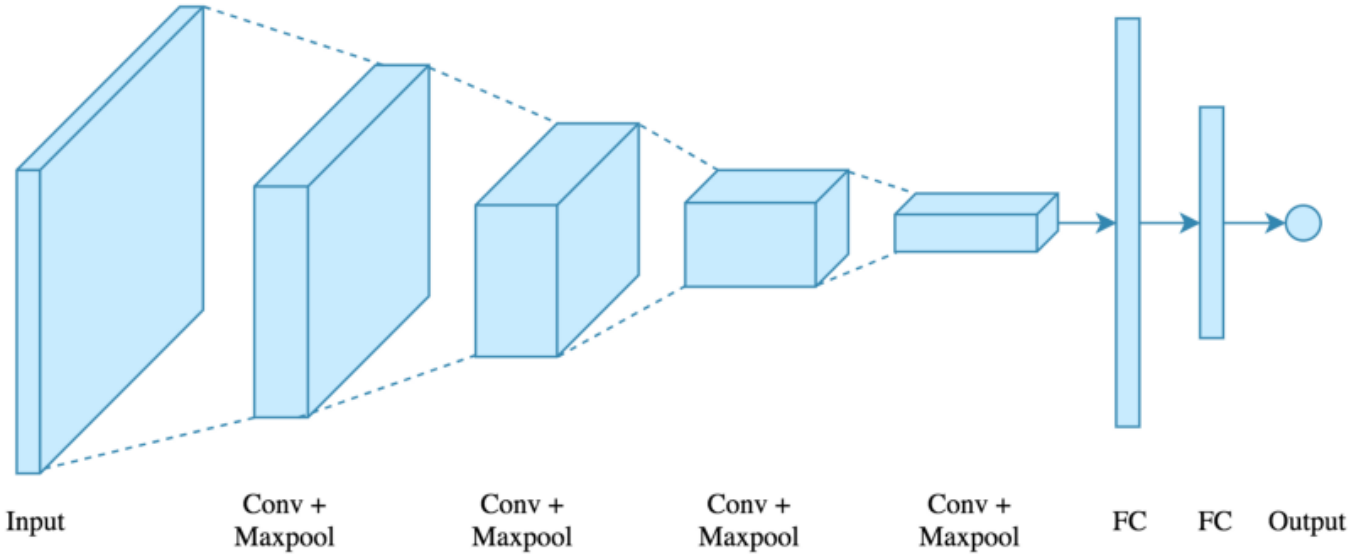
Fully Connected

- *Convolution* + *Pooling* katmanlarından sonra CNN mimarisi için *fully-connected* katmanlar oluşturulur.
- *Convolution* ve *pooling* katmanları 3-boyutludur. Fakat *fully-connected* katmanı tek boyutludur. Bu sebeple flatten ile 1D vektöre dönüşüm yapılır.

Eğitim
Training

Eğitim

- *Gradient Descent*
- Backpropagation In Convolutional Neural Networks [1]



[1] http://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/

Lab

Lab-1

İçindekiler

Word Embeddings

Uygulama Alanları

- Doküman sınıflandırma: Bir kitap veya makalede yer alan topic identification
- Sekans-Sekans öğrenme (Sequence-to-sequence learning): İngilizce'den - Türkçe'ye çeviri.
- Duygu analizi (Sentiment Analysis): Film yorumlarının pozitif veya negatif olduğunun algılanması
- Zaman serisi analizi: Bir bölgede yapılan hava tahmini
- Zararlı yazılım analizi: Sandbox API çağırımları

Metin Veri Kümesi
Working with text data

Veri Dönüşümü

- Bütün makine öğrenmesi algoritmalarında olduğu gibi derin öğrenme algoritmalarıda sayısal değerlerle çalışmaktadır.
- **Vectorizing text**: Metinlerin nümerik tensorlere dönüştürülmesi
  - Segment text into words, and transform each word into a vector.
  - Segment text into characters, and transform each character into a vector.
  - Extract n-grams of words or characters, and transform each n-gram into a vector.
  - Metinin farklı birimlere ayrıştırılması (kelimeler, karakterler, n-grams)

Word Embeddings I

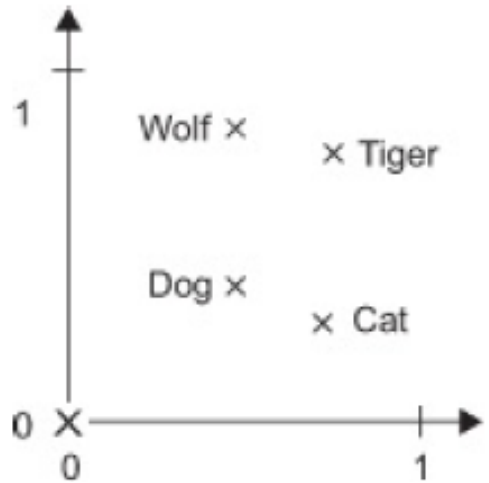
Word Embeddings

- Kelimelerin yüksek boyutlu bir uzayda gerçek değerler olarak kodlanması
- Kelimeler arasında bulunan anlam benzerlikleri kullanılarak vektörlerin birbirlerine olan yakınlıkları kullanılır.
- **Keras** üzerinde bulunan *Embeddings* katmanı kullanılarak kelimelerin tam sayı gösterimleri *word embedding'*e çevrilmektedir.
- Word embedding eğitimi: ana görevle beraber (döküman sınıflandırma, duygu analizi) eğitilmektedir.
- Farklı bir görev için eğitilmiş olan word embedding kullanılabilir. *pretrained word embeddings*

Word Embeddings II

Geometrik İlişki

- **Word embeddings**: map human language into a geometric space
- **Örnek**: Eş anlamlı kelimeler birbirlerine yakın olması beklenir.
  - *cat* → *tiger* ve *dog* → *wolf*
  - *pet* → *wild animal*
- **Örnek**: Cinsiyet ve çoğul vektörleri
  - *King* → *kings queen* → *queens*



Lab

Lab-2

İçindekiler

Recurrent Neural Networks I

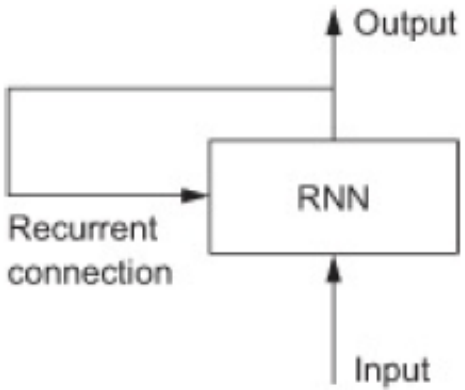
Mevcut Durum

- Diğer sinir ağlarında herhangi bir hafıza (memory) yoktur.
- no state kept in between inputs.
- Bir cümle okunurken kelime-kelime ilerlenir. Anlam oluşturulurken sıralama önemlidir.
- Biyolojik zeka işleyişi, bir modeli korunurken bilgiyi aşamalı olarak işler.

Recurrent Neural Networks II

Recurrent Neural Networks

- RNN son derece basitleştirilmiş bir versiyonda da olsa, aynı prensibi benimsemektedir.
- Dizi elemanları arasında yineleme yaparak dizileri işler.
- O ana kadar gördüklerine göre bilgi içeren bir durumun sürdürülür.
- internal loop
- RNN'nin durumu, iki farklı bağımsız sekansın işlenmesi arasında sıfırlanır. (İki farklı IMDB yorumu gibi)

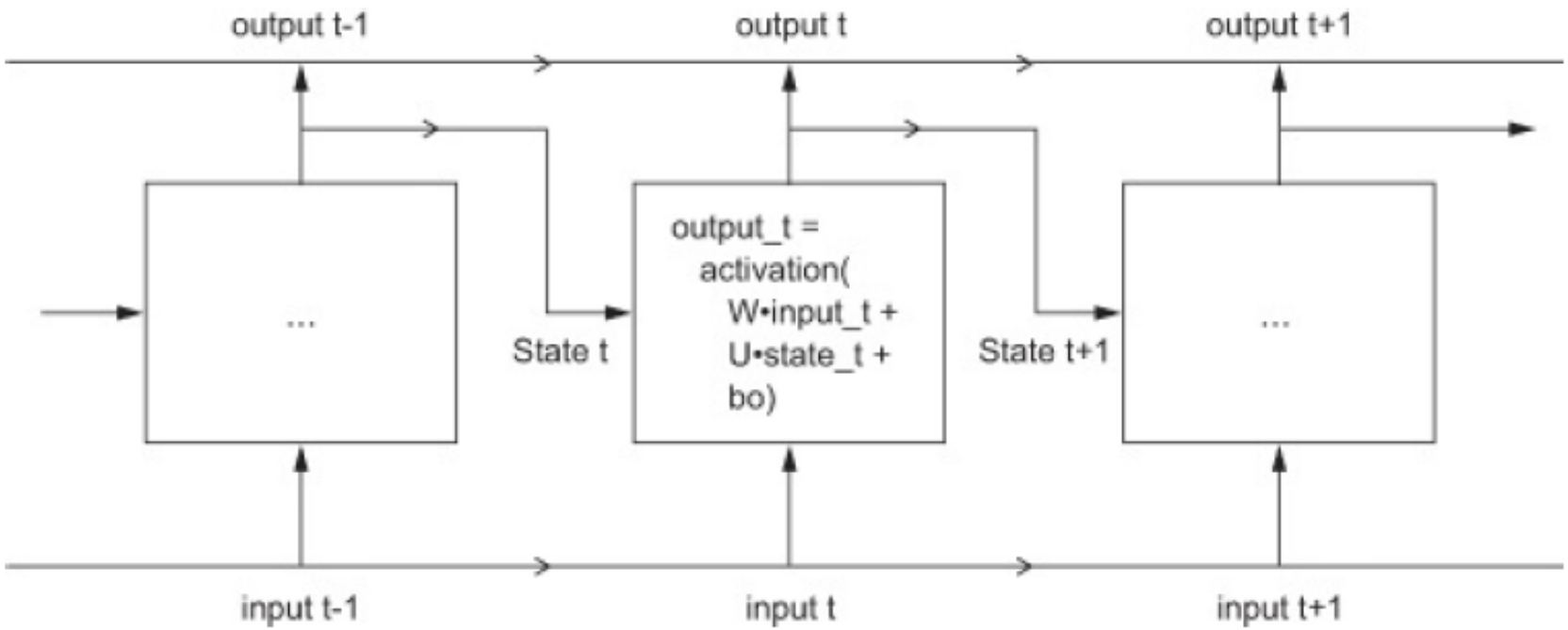


Recurrent Neural Networks III

RNN

- RNN bir **for loop** gibi düşünülebilir.

```
output_t = np.tanh(np.dot(W, input_t) + np.dot(U, state_t) + b)
```



Recurrent Neural Networks IV

Keras

```
from keras.layers import SimpleRNN
SimpleRNN(batch_size, timesteps, input_features)
```

Recurrent Neural Networks V

```
>>> from keras.models import Sequential
>>> from keras.layers import Embedding, SimpleRNN
>>> model = Sequential()
>>> model.add(Embedding(10000, 32))
>>> model.add(SimpleRNN(32))
>>> model.summary()
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_22 (Embedding)     (None, None, 32)          320000
_________________________________________________________________
simplernn_10 (SimpleRNN)     (None, 32)                2080
=================================================================
Total params: 322,080
Trainable params: 322,080
Non-trainable params: 0
```

Recurrent Neural Networks VI

```
>>> model = Sequential()
>>> model.add(Embedding(10000, 32))
>>> model.add(SimpleRNN(32, return_sequences=True))
>>> model.add(SimpleRNN(32, return_sequences=True))
>>> model.add(SimpleRNN(32, return_sequences=True))
>>> model.add(SimpleRNN(32)) 1
>>> model.summary()
_________________________________________________________________
Layer (type) Output Shape Param #
=================================================================
embedding_24 (Embedding) (None, None, 32) 320000
_________________________________________________________________
simplernn_12 (SimpleRNN) (None, None, 32) 2080
_________________________________________________________________
simplernn_13 (SimpleRNN) (None, None, 32) 2080
_________________________________________________________________
simplernn_14 (SimpleRNN) (None, None, 32) 2080
_________________________________________________________________
simplernn_15 (SimpleRNN) (None, 32) 2080
=================================================================
Total params: 328,320
Trainable params: 328,320
Non-trainable params: 0
```

Lab

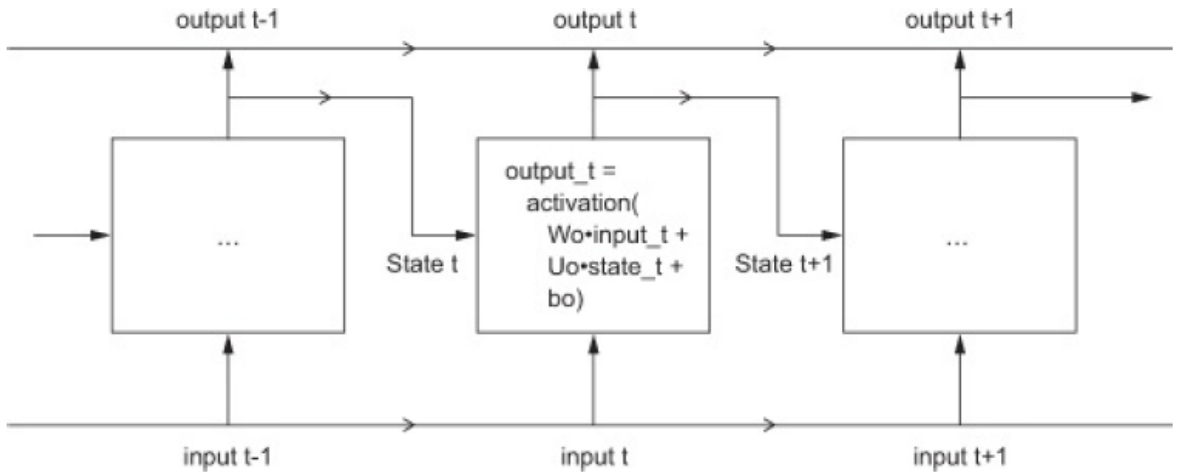
Lab 3

İçindekiler

Long Short-Term Memory (LSTM) I

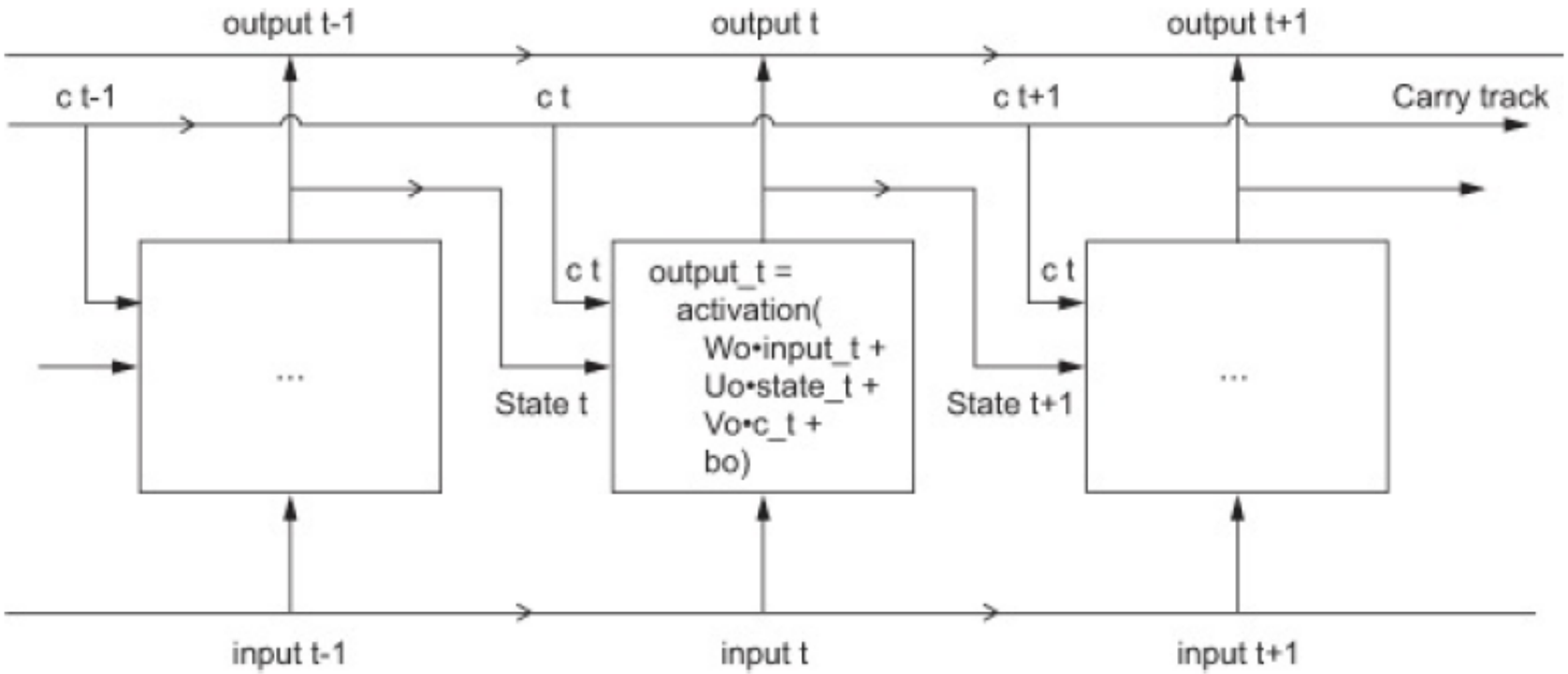
LSTM

- Yoshua Bengio, Patrice Simard, and Paolo Frasconi, "Learning Long-Term Dependencies with Gradient Descent Is Difficult," IEEE Transactions on Neural Networks 5, no. 2 (1994).
- Bilgi bir çok defa taşınmaktadır.



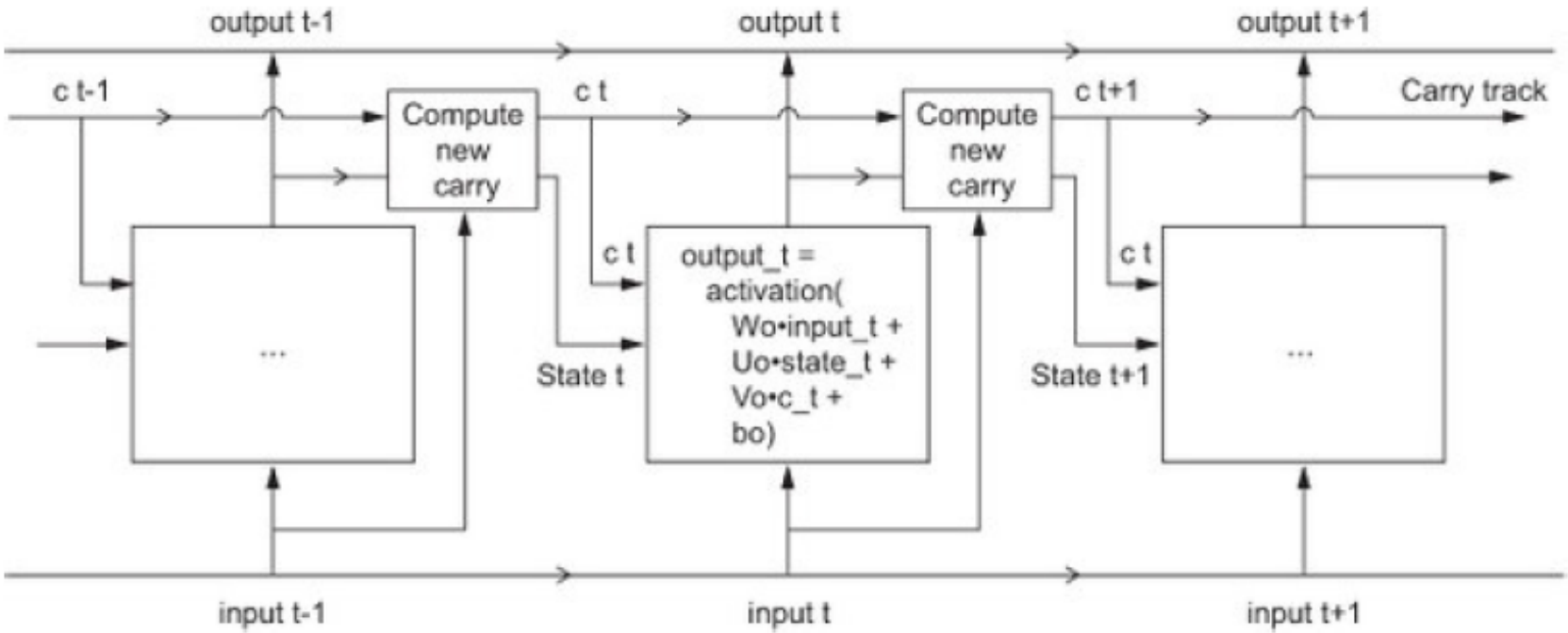
Şekil: SimpleRNN

Long Short-Term Memory (LSTM) II



Şekil: *simplernn* to an *lstm*: adding a **carry track**

Long Short-Term Memory (LSTM) III



Şekil: Anatomy of an lstm

Long Short-Term Memory (LSTM) IV

```
from keras.layers import LSTM
model = Sequential()
model.add(Embedding(max_features, 32))
model.add(LSTM(32))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='rmsprop',
loss='binary_crossentropy',
metrics=['acc'])
history = model.fit(input_train, y_train,
epochs=10,
batch_size=128,
validation_split=0.2)
```