

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

- 1 Evrişimli Sinir Ağları
 - Giriş
 - Convolution
 - Non-linearity
 - Stride ve Padding
 - Pooling
 - Hyperparameters
 - Fully Connected

- 2 Word Embeddings
 - Eğitim
 - Giriş
 - Metin Veri Kümesi
 - Word Embeddings
- 3 Recurrent Neural Networks
 - Giriş
- 4 LSTM
 - Long Short-Term Memory

İçindekiler

- 1 Evrişimli Sinir Ağları
 - Giriş
 - Convolution
 - Non-linearity
 - Stride ve Padding
 - Pooling
 - Hyperparameters
 - Fully Connected

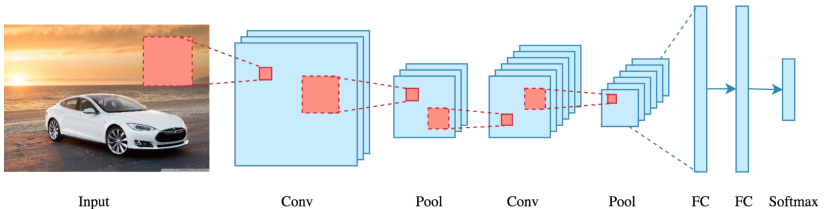
- Eğitim
- 2 Word Embeddings
 - Giriş
 - Metin Veri Kümesi
 - Word Embeddings
- 3 Recurrent Neural Networks
 - Giriş
- 4 LSTM
 - Long Short-Term Memory

Evrışimli 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

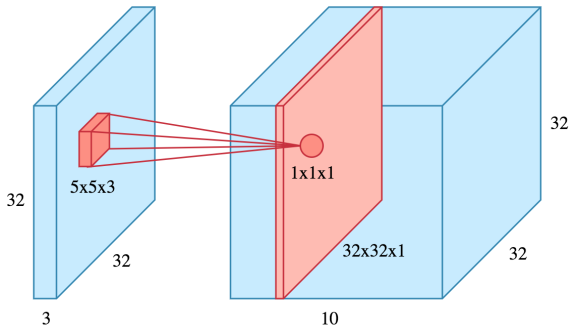
Filter / Kernel

Convolution

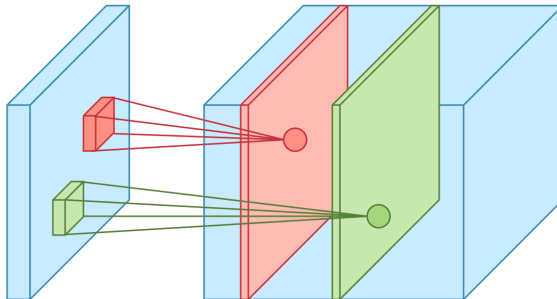
Convolution I

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

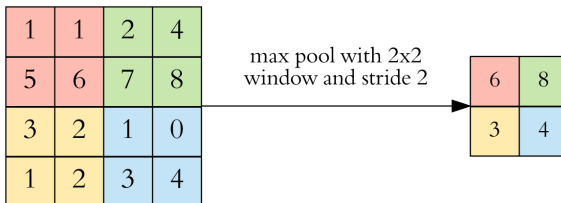
- ▶ Input map ve feature map aynı olması isteniyorsa, 0'lardan oluşan değerler eklenir.

Pooling I

Downsampling

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*

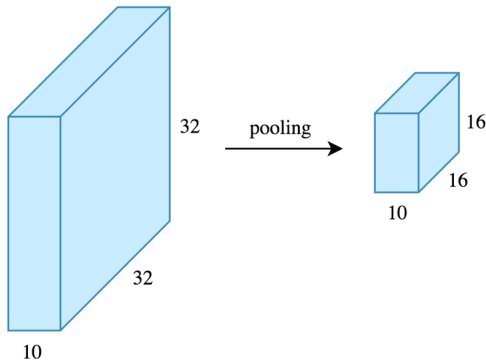


Pooling II

Downsampling

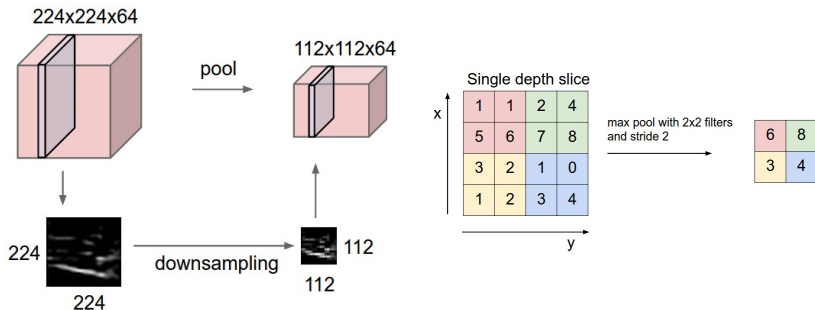
Örnek

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



Pooling III

Downsampling



Hyperparameters

Hyperparameters

- ▶ **Depth:** the number of filters. A set of neurons that are all looking at the same region of the input as a depth column. Genellikle 3×3 , 5×5 , 7×7 .
- ▶ *Filter count:* 2'nin katları şeklindedir. [32, 1024]. Filtre sayısı artıkcça daha güçlü ama aşırı öğrenmeye daha yatkın model oluşur.
- ▶ *Stride*
- ▶ *Padding*

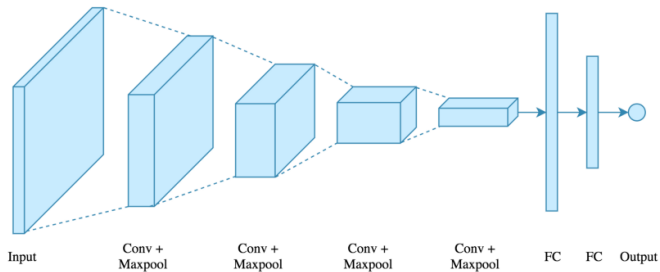
Fully Connected

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.

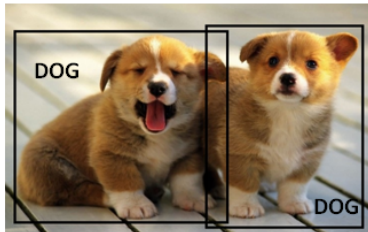
Eğitim

- ▶ *Gradient Descent*
- ▶ Backpropagation In Convolutional Neural Networks ¹



Eğitim II

Training



Object Detection involves localization of multiple objects (doesn't have to be the same class).



Object Segmentation involves the class label as well as an outline of the object in interest.

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

Lab-1

İçindekiler

- 1 Evrişimli Sinir Ağları
 - Giriş
 - Convolution
 - Non-linearity
 - Stride ve Padding
 - Pooling
 - Hyperparameters
 - Fully Connected

- 2 Word Embeddings
 - Eğitim
 - Giriş
 - Metin Veri Kümesi
 - Word Embeddings
- 3 Recurrent Neural Networks
 - Giriş
- 4 LSTM
 - Long Short-Term Memory

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 algoritmalarında 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.
 - ▶ Metnin farklı birimlere ayrıştırılması (kelimeler, karakterler, n-grams)

Word Embeddings I

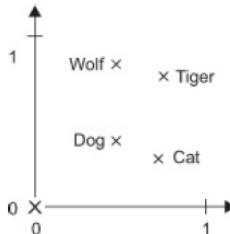
Word Embeddings

- ▶ 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 çevrilmiştir.
- ▶ 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-2

İçindekiler

- 1 Evrişimli Sinir Ağları
 - Giriş
 - Convolution
 - Non-linearity
 - Stride ve Padding
 - Pooling
 - Hyperparameters
 - Fully Connected

- 2 Word Embeddings
 - Eğitim
 - Giriş
 - Metin Veri Kümesi
 - Word Embeddings
- 3 Recurrent Neural Networks
 - Giriş
- 4 LSTM
 - Long Short-Term Memory

Recurrent Neural Networks I

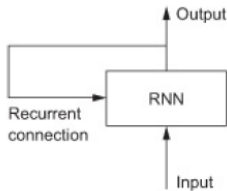
Mevcut Durum

- ▶ 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)



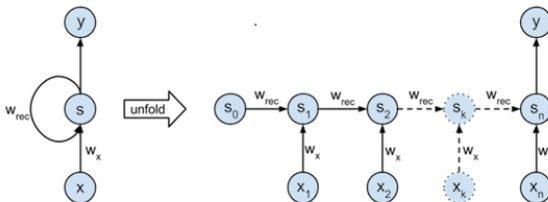
Recurrent Neural Networks III

RNN

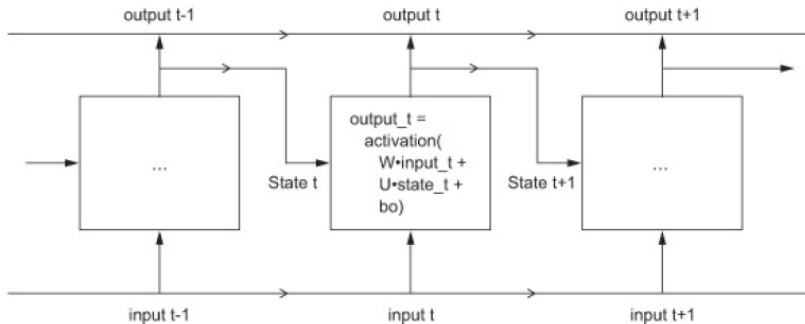
- RNN bir **for loop** gibi düşünülebilir.

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

$$\mathbf{s}_t = f(\mathbf{s}_{t-1} \cdot \mathbf{w}_{rec} + \mathbf{x}_k \cdot \mathbf{w}_x) \quad (1)$$



Recurrent Neural Networks IV



Recurrent Neural Networks V

Keras

```
from keras.layers import SimpleRNN
SimpleRNN(units, activation='tanh')
```

Recurrent Neural Networks VI

```
>>> 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 VII

```
>>> 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 3

İçindekiler

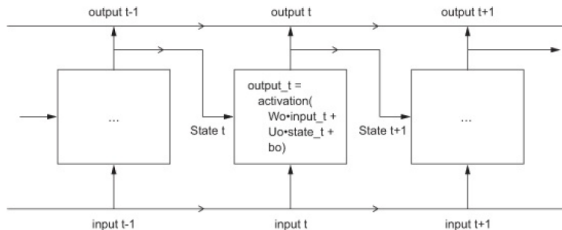
- 1 Evrişimli Sinir Ağları
 - Giriş
 - Convolution
 - Non-linearity
 - Stride ve Padding
 - Pooling
 - Hyperparameters
 - Fully Connected

- 2 Word Embeddings
 - Eğitim
 - Giriş
 - Metin Veri Kümesi
 - Word Embeddings
- 3 Recurrent Neural Networks
 - Giriş
- 4 LSTM
 - Long Short-Term Memory

Long Short-Term Memory (LSTM) I

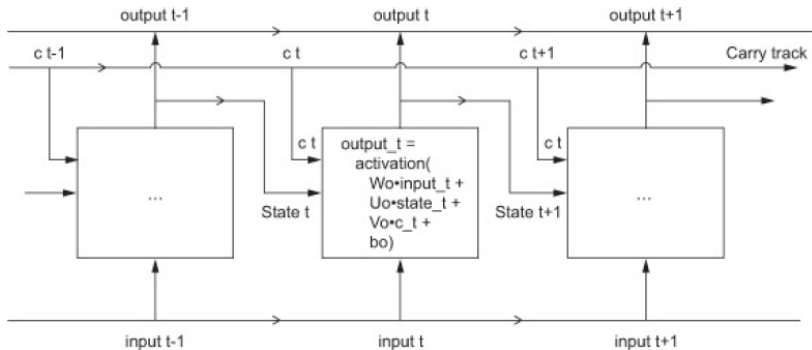
LSTM

- ▶ 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.



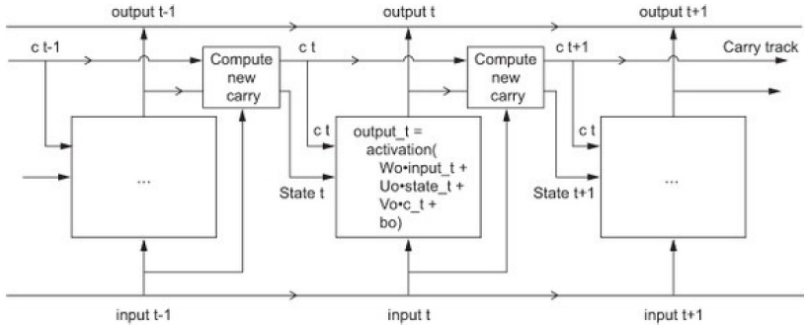
Şekil: SimpleRNN

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)
```