## **Recurrent Neural Network (RNN)**

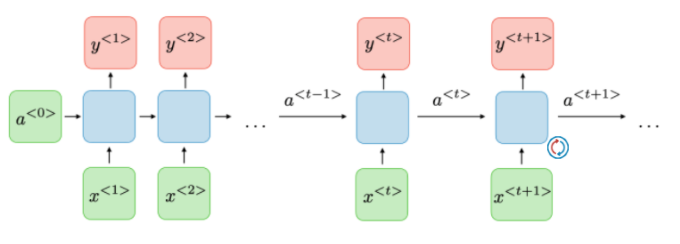
## **RNN Theory**

### **What is RNN? [RNN nedir?]**

Recurrent Neural Networks (RNNs) are a well-known technique in sequence modeling. [Tekrarlayan Sinir Ağları (RNN'ler), dizi modellemede iyi bilinen bir tekniktir.]  Machine learning models that input or output data sequences are known as sequence models. [Veri dizilerini giren veya çıkaran makine öğrenimi modelleri, dizi modelleri olarak bilinir.] Text streams, audio clips, video clips, time-series data, and other types of sequential data are examples of sequential data. [Metin akışları, ses klipleri, video klipler, zaman serisi verileri ve diğer sıralı veri türleri sıralı veri örnekleridir.]

A recurrent neural network (RNN) is a type of artificial neural network in which node connections form a directed graph along a time sequence. [Tekrarlayan bir sinir ağı (RNN), düğüm bağlantılarının bir zaman dizisi boyunca yönlendirilmiş bir grafik oluşturduğu bir tür yapay sinir ağıdır.] As a result, it can display temporal dynamic behavior. [Sonuç olarak, zamansal dinamik davranış gösterebilir.] RNNs, which are derived from feedforward neural networks, can handle variable length sequences of inputs using their internal state (memory). [İleri beslemeli sinir ağlarından türetilen RNN'ler, dahili durumlarını (belleklerini) kullanarak değişken uzunluktaki girdi dizilerini işleyebilir.]

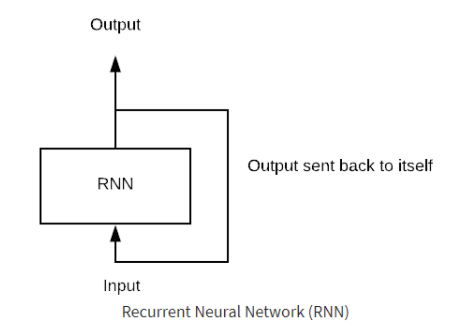
RNNs are a type of neural network that has hidden states which allows past outputs to be utilized as inputs. [RNN'ler, geçmiş çıktıların girdi olarak kullanılmasına izin veren gizli durumlara sahip bir tür sinir ağıdır.]  Recurrent implies that the current time step's output becomes the input for the next time step. [Tekrarlayan, mevcut zaman adımının çıktısının bir sonraki zaman adımının girdisi olduğu anlamına gelir.] The model analyzes not just the current input, but also what it knows about the preceding components, at each element of the sequence. [Model, yalnızca mevcut girdiyi değil, aynı zamanda dizinin her bir öğesinde önceki bileşenler hakkında ne bildiğini de analiz eder.]



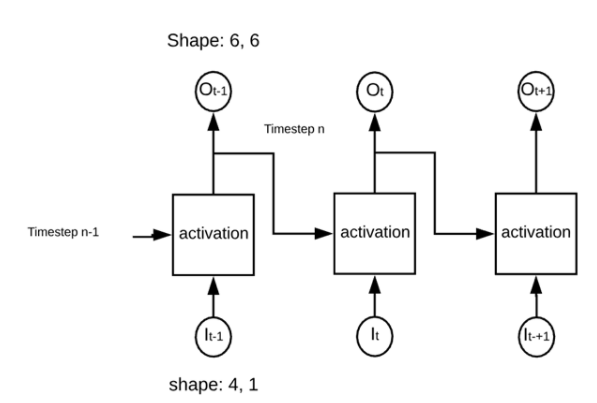
### **How does RNN work? [RNN nasıl çalışır?]**

A recurrent neural network resembles a standard neural network except that the neurons have a memory-state. [Tekrarlayan bir sinir ağı, nöronların bir hafıza durumuna sahip olması dışında standart bir sinir ağına benzer.] The computation to incorporate a memory is straightforward. [Bir hafızayı dahil etmek için hesaplama basittir.]

Consider a basic model with only one neuron that is fed by a batch of data. [Bir yığın veri tarafından beslenen tek bir nörona sahip temel bir model düşünün.] The output of a conventional neural net is produced by multiplying the input by the weight and the activation function. [Geleneksel bir sinir ağının çıktısı, girdinin ağırlık ve aktivasyon fonksiyonu ile çarpılmasıyla üretilir.] This output is transmitted back to the RNN a number of times. [Bu çıktı birkaç kez RNN'ye geri iletilir.] The length of time it takes for the result to become the input of the next matrice multiplication is referred to as a timestep. [Sonucun bir sonraki matris çarpımının girdisi olması için geçen süreye zaman adımı denir.]



In the image above, one neuron network multiplies the input and weight matrices and adds non-linearity with the activation function. [Yukarıdaki görüntüde, bir nöron ağı, girdi ve ağırlık matrislerini çarpar ve aktivasyon fonksiyonu ile doğrusal olmayanlık ekler.] At t-1, it becomes the output. [t-1'de çıktı olur.] This output is the second matrix multiplication's input. [Bu çıktı, ikinci matris çarpımının girdisidir.]



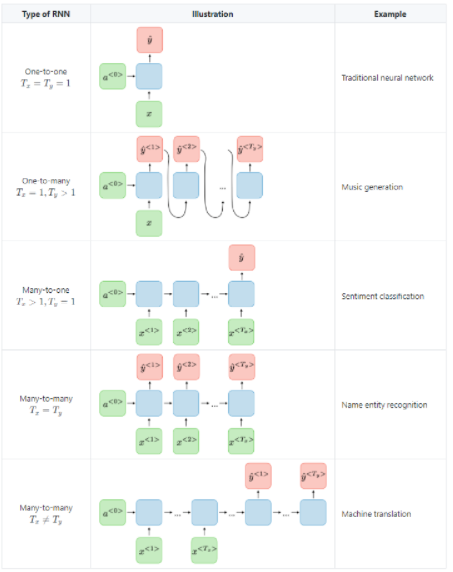
Here is another network is referred to as recurrent because it executes the same function in each activated square. [Burada başka bir ağ, aktif olan her karede aynı işlevi yerine getirdiği için tekrarlayan olarak adlandırılır.] Before using an activation function, the network calculated the weights of the inputs and the previous output. [Aktivasyon fonksiyonunu kullanmadan önce ağ, girdilerin ve önceki çıktıların ağırlıklarını hesapladı.]

# Illustrated Guide to Recurrent Neural Networks: Understanding the Intuition

<https://www.youtube.com/watch?v=LHXXI4-IEns>

### **Types of RNN? [RNN türleri?]**

RNN models are mostly utilized in natural language processing and speech recognition. [RNN modelleri çoğunlukla doğal dil işleme ve konuşma tanımada kullanılmaktadır.] The following table summarizes the many applications. [Aşağıdaki tablo birçok uygulamayı özetlemektedir.]



## **Vanishing Gradients**

### **Vanishing Gradients [Kaybolan Gradyanlar]**

RNNs are plagued by the problem of vanishing gradients, which makes learning lengthy data sequences difficult. [RNN'ler, uzun veri dizilerini öğrenmeyi zorlaştıran, kaybolan gradyanlar sorunuyla karşı karşıyadır.] The gradients convey information utilized in the RNN parameter update, and as the gradient shrinks, the parameter updates become negligible, implying that no actual learning occurs. [Gradyanlar, RNN parametre güncellemesinde kullanılan bilgileri iletir ve gradyan küçüldükçe parametre güncellemeleri ihmal edilebilir hale gelir, bu da gerçek bir öğrenmenin gerçekleşmediğini gösterir.]

When we backpropagate over time in our RNN model, the gradient can both fall and rise exponentially (as we observed above). [RNN modelimizde zaman içinde geri yayıldığımızda, gradyan üstel olarak hem düşebilir hem de yükselebilir (yukarıda gözlemlediğimiz gibi).] These expanding gradients can be terrible for our networks since they can lead our parameters to grow so huge that our network just goes crazy! [Bu genişleyen gradyanlar ağlarımız için korkunç olabilir, çünkü parametrelerimizi o kadar büyük hale getirebilirler ki ağımız çılgına döner!]

Exploding gradients are more visible than vanishing gradients. [Patlayan degradeler, kaybolan degradelerden daha görünür.] The network may show NaN (Not a Number), indicating that our neural network calculations have reached a numerical overrun. [Ağ, sinir ağı hesaplamalarımızın sayısal bir aşmaya ulaştığını gösteren NaN (Sayı Değil) gösterebilir.]

Gradient clipping can be used to alleviate the problem of exploding gradients. [Gradyan kırpma, gradyanların patlaması sorununu hafifletmek için kullanılabilir.] This is just scaling or rescaling our gradient vectors when they hit a threshold or maximum value. [Bu, yalnızca bir eşiğe veya maksimum değere ulaştıklarında gradyan vektörlerimizi ölçeklendirmek veya yeniden ölçeklendirmektir.] On the other hand vanishing gradients are quite tricky to solve. [Öte yandan, kaybolan gradyanları çözmek oldukça zordur.]

Vanishing gradients are common when the Sigmoid or Tanh activation functions are used in the hidden layer units. [Gizli katman birimlerinde Sigmoid veya Tanh etkinleştirme işlevleri kullanıldığında kaybolan gradyanlar yaygındır.] When the inputs grow extremely tiny or very big, the sigmoid function saturates at 0 and 1 while the tanh function saturates at -1 and 1. [Girişler aşırı derecede küçük veya çok büyük büyüdüğünde, sigmoid işlevi 0 ve 1'de doygun hale gelirken tanh işlevi -1 ve 1'de doyurulur.] In each of these situations, the derivatives are very near to zero. [Bu durumların her birinde türevler sıfıra çok yakındır.]

### **Solutions of Vanishing Gradients Problem [Kaybolan Gradyan Probleminin Çözümleri]**

Vanishing gradients are common when the Sigmoid or Tanh activation functions are used in the hidden layer units. [Gizli katman birimlerinde Sigmoid veya Tanh etkinleştirme işlevleri kullanıldığında kaybolan gradyanlar yaygındır.] When the inputs grow extremely tiny or very big, the sigmoid function saturates at 0 and 1 while the tanh function saturates at -1 and 1. [Girişler aşırı derecede küçük veya çok büyük büyüdüğünde, sigmoid işlevi 0 ve 1'de doygun hale gelirken tanh işlevi -1 ve 1'de doyurulur.] In each of these situations, the derivatives are very near to zero. [Bu durumların her birinde türevler sıfıra çok yakındır.] To get rid of vanishing gradients problem following techniques can be used. [Kaybolan gradyan probleminden kurtulmak için aşağıdaki teknikler kullanılabilir.]

1. Weight initialization is one method for resolving the vanishing gradient problem. [Ağırlık başlatma, kaybolan gradyan problemini çözmek için bir yöntemdir.] It entails generating a fake starting value for weights in a neural network to avoid the backpropagation algorithm from assigning excessively tiny weights. [Geri yayılım algoritmasının aşırı küçük ağırlıklar atamasını önlemek için bir sinir ağındaki ağırlıklar için sahte bir başlangıç ​​değeri oluşturmayı gerektirir.]
2. Another obvious option would be to utilize activation functions that are not constrained by 1. (a popular choice being the ReLU function). [Bir diğer bariz seçenek, 1 ile sınırlandırılmayan etkinleştirme işlevlerini kullanmaktır (popüler bir seçim ReLU işlevidir).]
3. Batch normalization may be viewed as an extra layer in the network that normalizes your data (using a mean and standard deviation) before putting it into the hidden unit activation function, omitting the rigorous mathematical intricacies. [Toplu normalleştirme, ağda verilerinizi (ortalama ve standart sapma kullanarak) gizli birim etkinleştirme işlevine koymadan önce katı matematiksel karmaşıklıkları atlayarak normalleştiren ekstra bir katman olarak görülebilir.]
4. A kind of neural network known as Long Short-Term Memory Networks is the most effective solution to the vanishing gradient problem (LSTMs). [Uzun Kısa Süreli Bellek Ağları olarak bilinen bir tür sinir ağı, kaybolan gradyan problemine (LSTM'ler) en etkili çözümdür.]

# Vanishing and exploding gradients | Deep Learning Tutorial 35 (Tensorflow, Keras & Python)

<https://www.youtube.com/watch?v=qowp6SQ9_Oo>

## **LSTM and GRU**

### 

### **Long Short Term Memory [Uzun Kısa Süreli Bellek]**

Long Short Term Memory networks, abbreviated as "LSTMs," are a kind of RNN capable of learning long-term dependencies. ['LSTM'ler' olarak kısaltılan Uzun Kısa Süreli Bellek ağları, uzun vadeli bağımlılıkları öğrenebilen bir tür RNN'dir.] Hochreiter and Schmidhuber (1997) introduced them, and numerous others developed and popularized them in subsequent work. [Hochreiter ve Schmidhuber (1997) bunları tanıtmış ve daha sonraki çalışmalarda pek çok başka kişi geliştirmiş ve popülerleştirmiştir.]   They function fantastically well on a wide range of problems and are now extensively utilized. [Çok çeşitli problemlerde fevkalade iyi çalışırlar ve şimdi yaygın olarak kullanılmaktadırlar.]

LSTMs are deliberately designed to avoid the problem of long-term dependency. [LSTM'ler, uzun vadeli bağımlılık probleminden kaçınmak için kasıtlı olarak tasarlanmıştır.] Remembering knowledge over extended periods of time is essentially their default habit. [Bilgiyi uzun süreler boyunca hatırlamak, esasen onların varsayılan alışkanlığıdır.]

All recurrent neural networks take the form of a chain of repeating neural network modules. [Tüm tekrarlayan sinir ağları, tekrarlayan sinir ağı modülleri zinciri şeklini alır.] This repeating module in conventional RNNs will have a relatively basic structure, such as a single tanh layer. [Geleneksel RNN'lerdeki bu yinelenen modül, tek bir tanh katmanı gibi nispeten temel bir yapıya sahip olacaktır.]

### **How LSTM Work [LSTM Nasıl Çalışır?]**

The cell state and its numerous gates are at the heart of LSTMs. [Hücre durumu ve onun sayısız kapısı, LSTM'lerin kalbinde yer alır.] The cell state serves as a highway for relative information to be sent all the way down the sequence chain. [Hücre durumu, dizi zinciri boyunca tüm yol boyunca gönderilecek ilgili bilgiler için bir otoyol görevi görür.] You may think of it as the network's "memory." [Bunu ağın 'hafızası' olarak düşünebilirsiniz.] In principle, the cell state can carry meaningful information throughout the sequence's processing. [Prensipte hücre durumu, dizinin işlenmesi boyunca anlamlı bilgiler taşıyabilir.] As a result, knowledge from earlier time steps can travel to later time steps, decreasing the impact of short-term memory. [Sonuç olarak, daha önceki zaman adımlarından elde edilen bilgi, kısa süreli belleğin etkisini azaltarak sonraki zaman adımlarına gidebilir.] As the cell state travels, information is added or withdrawn from the cell state via gates. [Hücre durumu ilerledikçe, kapılar aracılığıyla hücre durumuna bilgi eklenir veya geri çekilir.] The gates are several neural networks that determine whether information about the cell state is permitted.During training, the gates might learn which information is important to remember and which to discard. [Kapılar, hücre durumuyla ilgili bilgilere izin verilip verilmediğini belirleyen birkaç sinir ağıdır. Eğitim sırasında kapılar, hangi bilgilerin hatırlanması ve hangilerinin atılması gerektiğini öğrenebilir.]

There are sigmoid activations in Gates. [Gates'te sigmoid aktivasyonlar var.] Instead of squishing numbers ranging from -1 to 1 like tanh activation, it squishes values ranging from 0 to 1. [Tanh aktivasyonu gibi -1'den 1'e kadar olan sayıları ezmek yerine, 0'dan 1'e kadar olan değerleri eziyor.] This is useful for updating or forgetting data since any integer multiplied by 0 equals 0, leading values to vanish or be "forgotten." [Bu, verileri güncellemek veya unutmak için kullanışlıdır, çünkü 0 ile çarpılan herhangi bir tamsayı 0'a eşittir, bu da değerlerin kaybolmasına veya 'unutulmasına' neden olur.]  Because every integer multiplied by one has the same value, that value remains constant or is "maintained." [Bir ile çarpılan her tamsayı aynı değere sahip olduğundan, bu değer sabit kalır veya 'korunur'.] The network can learn which data is unimportant and should be deleted, and which data should be kept. [Ağ, hangi verilerin önemsiz olduğunu ve silinmesi gerektiğini ve hangi verilerin saklanması gerektiğini öğrenebilir.]

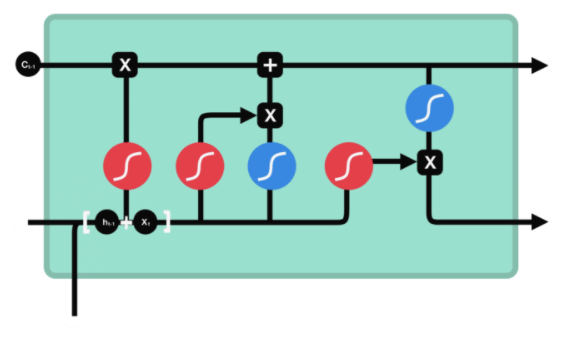
# Deep Learning: Long Short-Term Memory Networks (LSTMs)

<https://www.youtube.com/watch?v=5dMXyiWddYs>

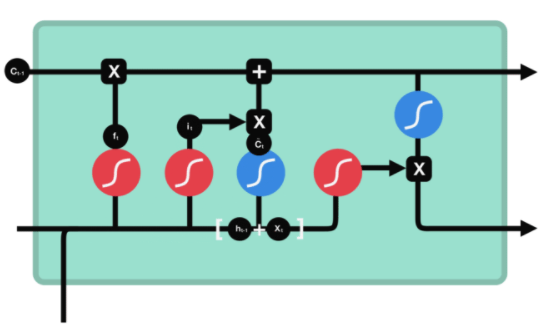
### **Gates of LSTM [LSTM Kapıları]**

In an LSTM cell, we have three distinct gates that govern information flow. [Bir LSTM hücresinde, bilgi akışını yöneten üç farklı kapımız vardır.] A forget gate, an input gate, and an output gate are all included. [Bir unutma kapısı, bir giriş kapısı ve bir çıkış kapısı dahildir.]

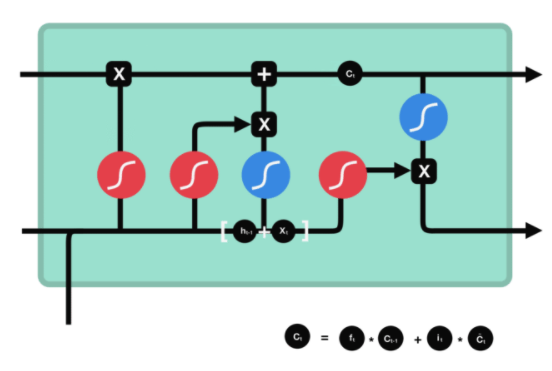
**Forget gate** determines whether information should be discarded or saved. [Unut kapısı, bilgilerin atılması veya kaydedilmesi gerektiğini belirler.] The sigmoid function is used to process information from the prior hidden state as well as information from the current input. [Sigmoid işlevi, mevcut girdiden gelen bilgilerin yanı sıra önceki gizli durumdan gelen bilgileri işlemek için kullanılır.] The values range from 0 to 1. [Değerler 0 ile 1 arasındadır.] The closer to 0 you are, the more likely you are to forget, and the closer to 1 you are, the more likely you are to keep. [0'a ne kadar yakınsanız, unutma olasılığınız o kadar yüksek ve 1'e ne kadar yakınsanız, tutma olasılığınız o kadar yüksek olur.]



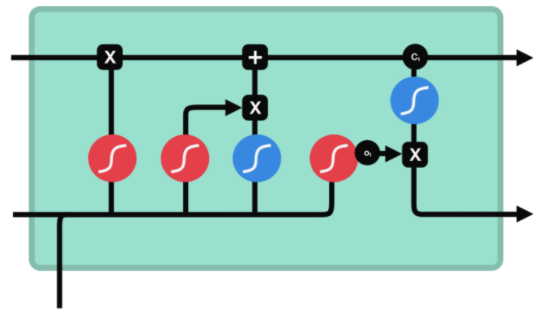
**Input gate** is used to update the cell state. [Giriş kapısı, hücre durumunu güncellemek için kullanılır.] First, we use a sigmoid function to combine the prior hidden state and the current input. [İlk olarak, önceki gizli durumu ve mevcut girişi birleştirmek için bir sigmoid işlevi kullanırız.] This determines which values are changed by converting them to be between 0 and 1. [Bu, 0 ile 1 arasında olacak şekilde dönüştürülerek hangi değerlerin değiştirileceğini belirler.] A value of 0 indicates that it is unimportant, whereas a value of 1 indicates that it is very significant. [0 değeri önemsiz olduğunu, 1 değeri ise çok önemli olduğunu gösterir.] To assist control the network, you also feed the hidden state and current input into the tanh function, which squishes values between -1 and 1. [Ağı kontrol etmeye yardımcı olmak için, gizli durumu ve mevcut girişi -1 ile 1 arasındaki değerleri ezen tanh işlevine de beslersiniz.] The sigmoid result is then multiplied by the tanh output. [Sigmoid sonucu daha sonra tanh çıktısı ile çarpılır.] The sigmoid output will determine which information from the tanh output should be retained. [Sigmoid çıktısı, tanh çıktısından hangi bilgilerin saklanması gerektiğini belirleyecektir.]



Having enough data to compute the cell status now **cell state** can be pointwise multiplied by the forget vector. [Hücre durumunu hesaplamak için yeterli veriye sahip olmak artık hücre durumu unutma vektörü ile noktasal olarak çarpılabilir.] If multiplied by values close to zero, this has the potential to drop values in the cell state. [Sıfıra yakın değerlerle çarpılırsa, bunun hücre durumundaki değerleri düşürme potansiyeli vardır.] The output of the input gate is then used to perform a pointwise addition, which changes the cell state to new values that the neural network considers meaningful. [Giriş kapısının çıkışı daha sonra hücre durumunu sinir ağının anlamlı bulduğu yeni değerlere değiştiren noktasal bir ekleme yapmak için kullanılır.] This results in our new cell state. [Bu, yeni hücre durumumuzla sonuçlanır.]



Finally, we have the **output gate**. [Son olarak, çıkış kapısına sahibiz.] The output gate determines the next hidden state. [Çıkış kapısı, bir sonraki gizli durumu belirler.] Keep in mind that the hidden state comprises data from earlier inputs. [Gizli durumun daha önceki girdilerden gelen verileri içerdiğini unutmayın.] Predictions are also made using the hidden state. [Gizli durum kullanılarak da tahminler yapılır.] First, we use a sigmoid function to combine the prior hidden state and the current input. [İlk olarak, önceki gizli durumu ve mevcut girişi birleştirmek için bir sigmoid işlevi kullanırız.] The newly changed cell state is then sent to the tanh function. [Yeni değiştirilen hücre durumu daha sonra tanh işlevine gönderilir.] To determine what information the hidden state should contain, we multiply the tanh output by the sigmoid output. [Gizli durumun hangi bilgileri içermesi gerektiğini belirlemek için tanh çıktısını sigmoid çıktısıyla çarpıyoruz.] The hidden state is the result. [Gizli durum sonuçtur.] The modified cell state and hidden value are then passed forward to the next time step. [Değiştirilen hücre durumu ve gizli değer daha sonra bir sonraki zaman adımına iletilir.]

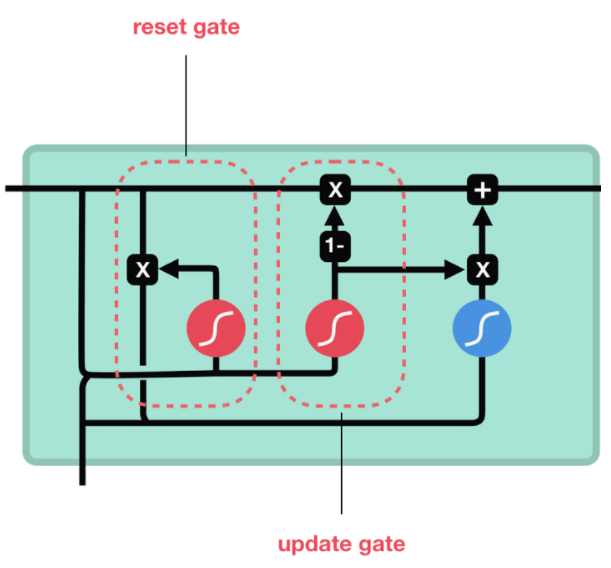


# Illustrated Guide to LSTM's and GRU's: A step by step explanation

<https://www.youtube.com/watch?v=8HyCNIVRbSU>

### **Gated Recurrent Unit**

The GRU is a newer version of Recurrent Neural Network that is similar to an LSTM. GRUs got rid of the cell state and instead used the hidden state to transmit data. It has two gates, one for reset and one for update.



**Update gate** determines what information to discard and what fresh information to include. [Güncelleme kapısı, hangi bilgilerin atılacağını ve hangi yeni bilgilerin ekleneceğini belirler.]  It is analogous to the combination of the Input Gate and the Forget Gate in an LSTM. [Bir LSTM'deki Giriş Kapısı ve Unut Kapısının kombinasyonuna benzer.]

**Reset gate** determines how much of the past knowledge to forget. [Reset gate, geçmiş bilginin ne kadarının unutulacağını belirler.]

# Simple Explanation of GRU (Gated Recurrent Units) | Deep Learning Tutorial 37 (Tensorflow & Python)

<https://www.youtube.com/watch?v=tOuXgORsXJ4>