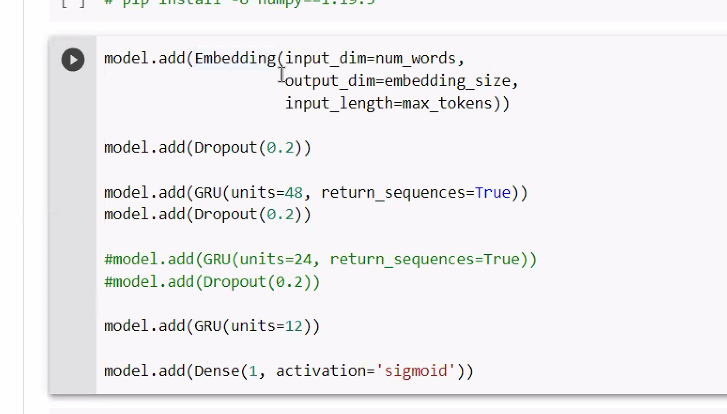
Embedding layer NLP’ye özgü bir şeydir. Numeric yapmak için kullanırız.

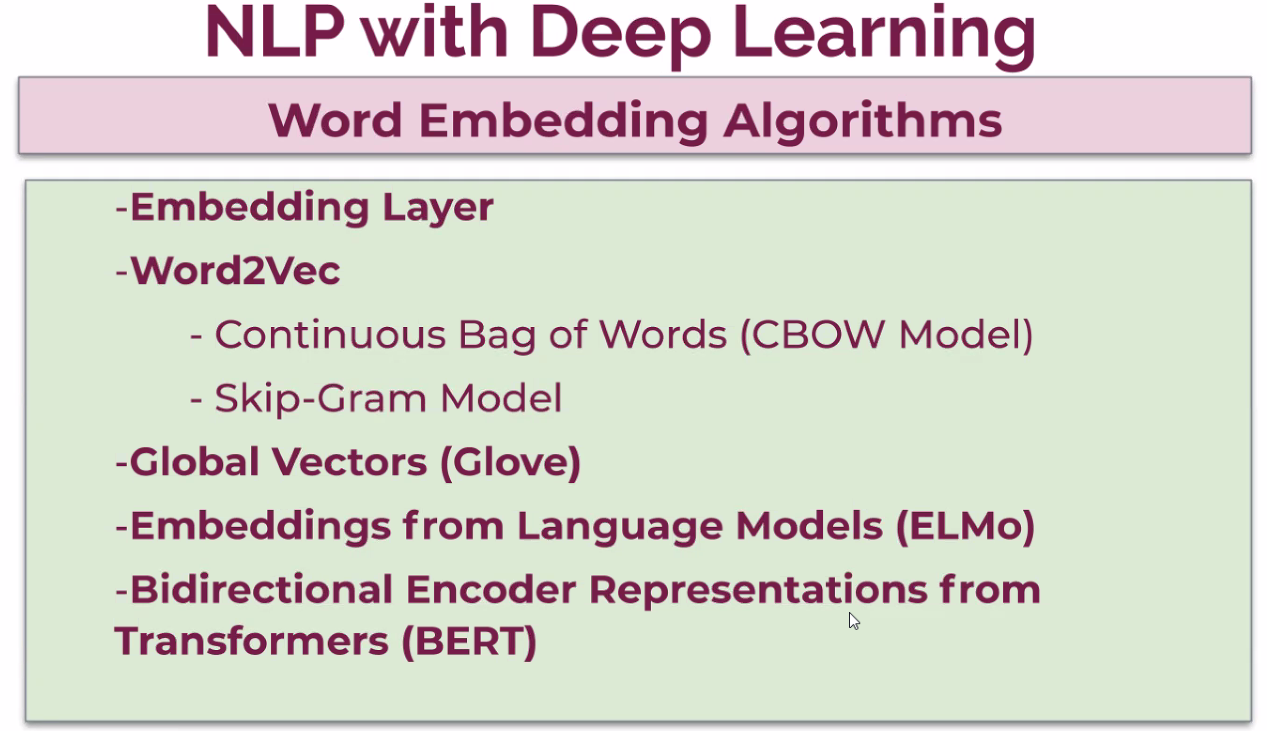


Word2Vec hangi token hangi tokenle kullanılıyor bunu tespit eder, her defasında katsayıları günceller.

Mesela “deep” ve “learning” in birbiriyle kullanım sıklığını buluyor. Eğer bunu 2. kez görürse tekrar, 4 de 5 de … Corpus içerisinde her denk gelişlerinde ağırlıklarını güncelliyor ( bunu da backpropagation sırasında yapıyor, zaten bu model bir RNN modeli).

Global Vectors ise direkt Deep ve learning kaç kere kullanılmış ilk başta onu hesaplıyor, her defasında güncelleme yapmak yerine.

ELMo ve BERT çift yönlü çalıştıkları için anlamsal ilişkileri daha iyi yakalayabiliyor.



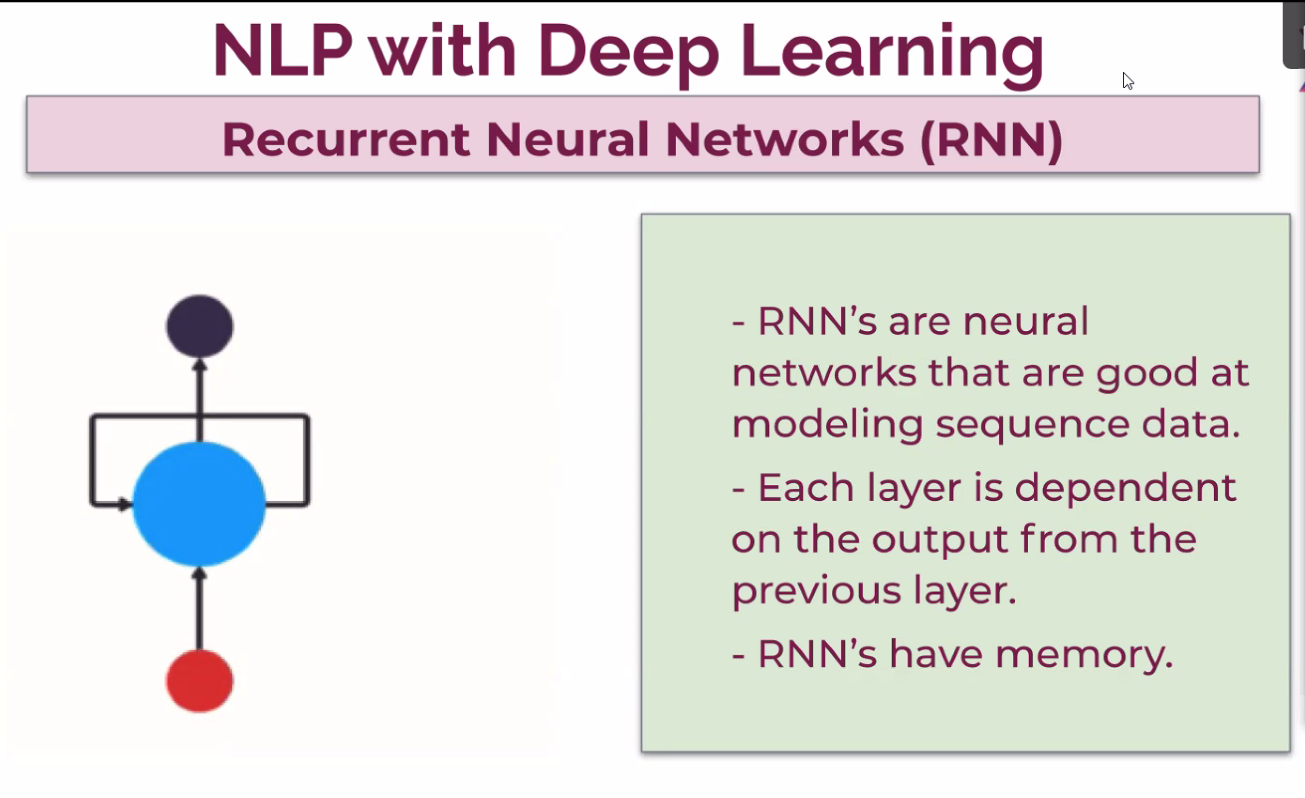
Windows\_size = orta kelimemden kaç adım ilerleyeyim?

Windows size ne kadar büyük o kadar iyi öğrenir. Best practice 5 ve 10 veriliyor.

Kırmızı araba trafik kazasına karıştı.

Burda kırmızı için orta kelime diyoruz. Burda model tamlamaları öğreniyor. Anlamsal ilişkileri çözüyor. Mesela burada renkler ve arabaların bir ilişkisi olduğunu öğreniyor.

Corpusumuzun çok büyük olması lazım tabii, milyon milyar…

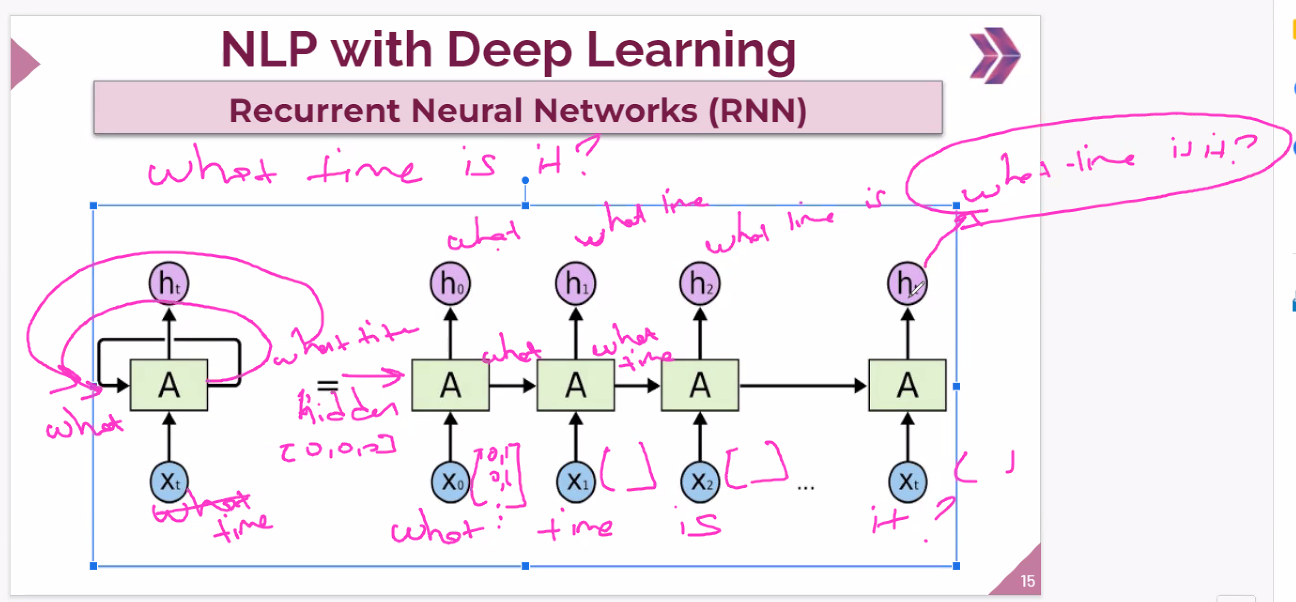


Dizi modellemede RNN iyi.

CNN ve ANN’den en büyük farkları RNN’in 🡪

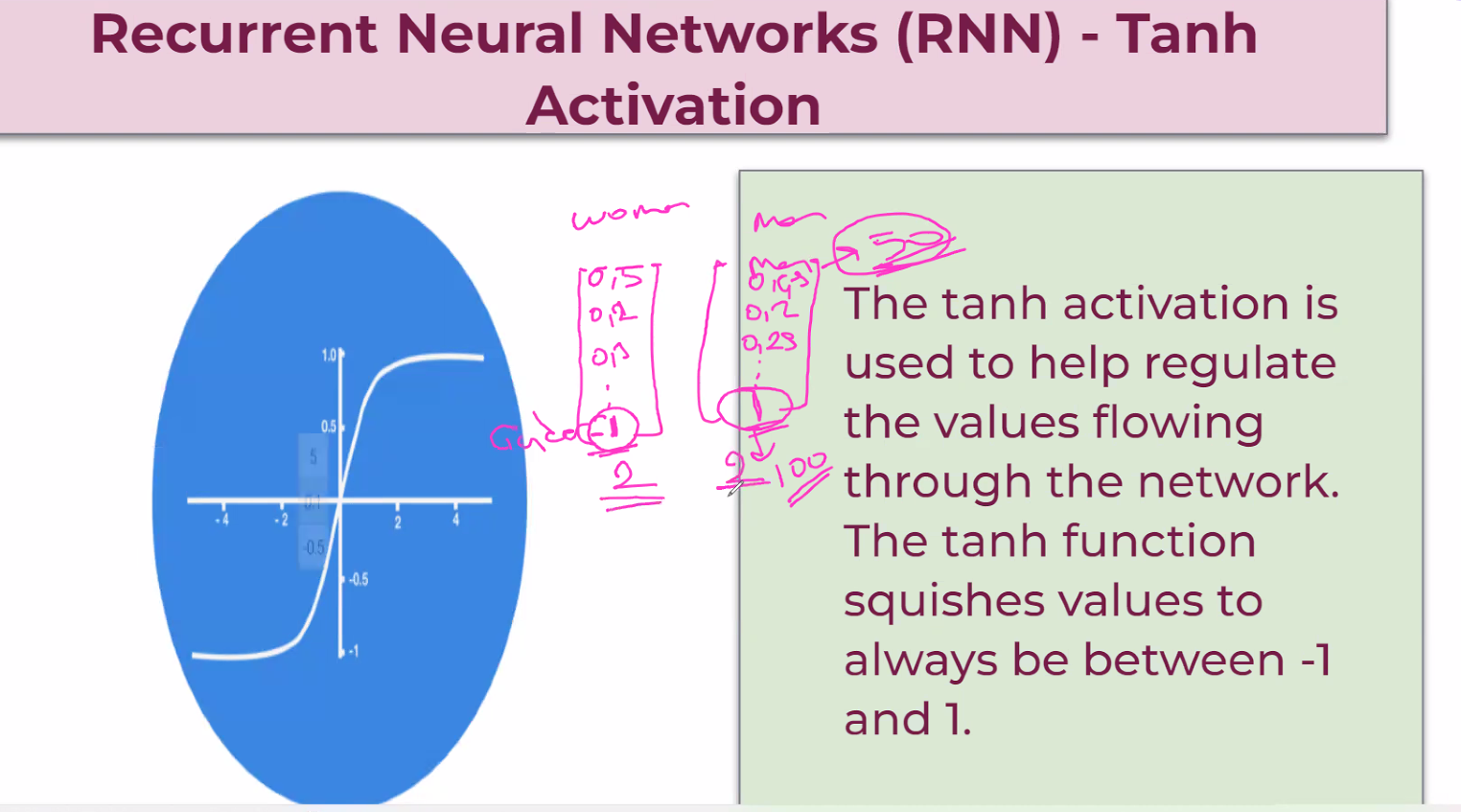
-Her katman bir önceki katmandan bilgiye mahkum.

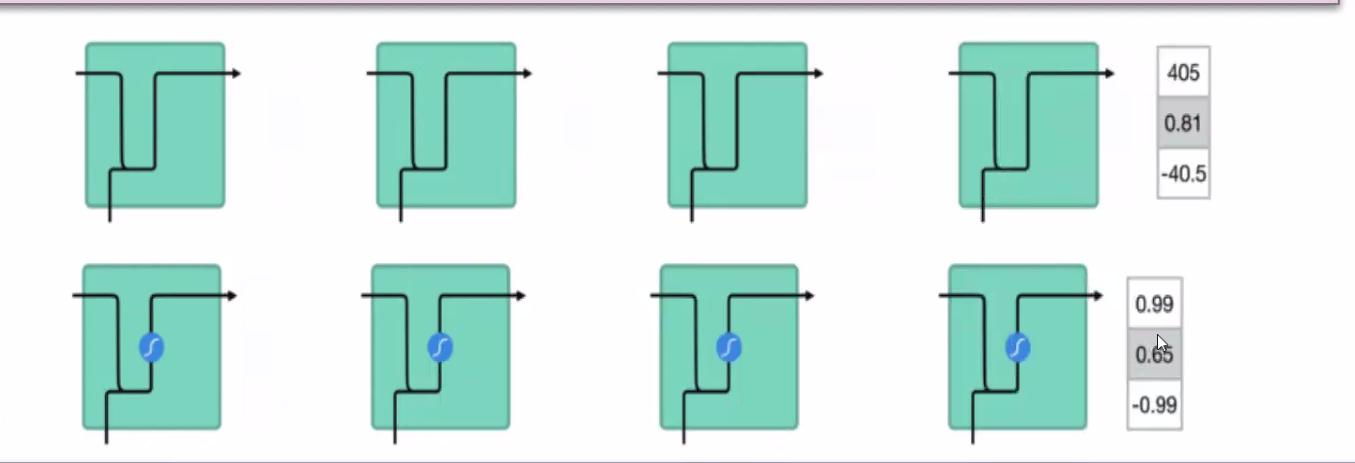
-Ve hafızası var RNN’in

Zaten o bilgileri almazsa yine kötü sonuç verir.

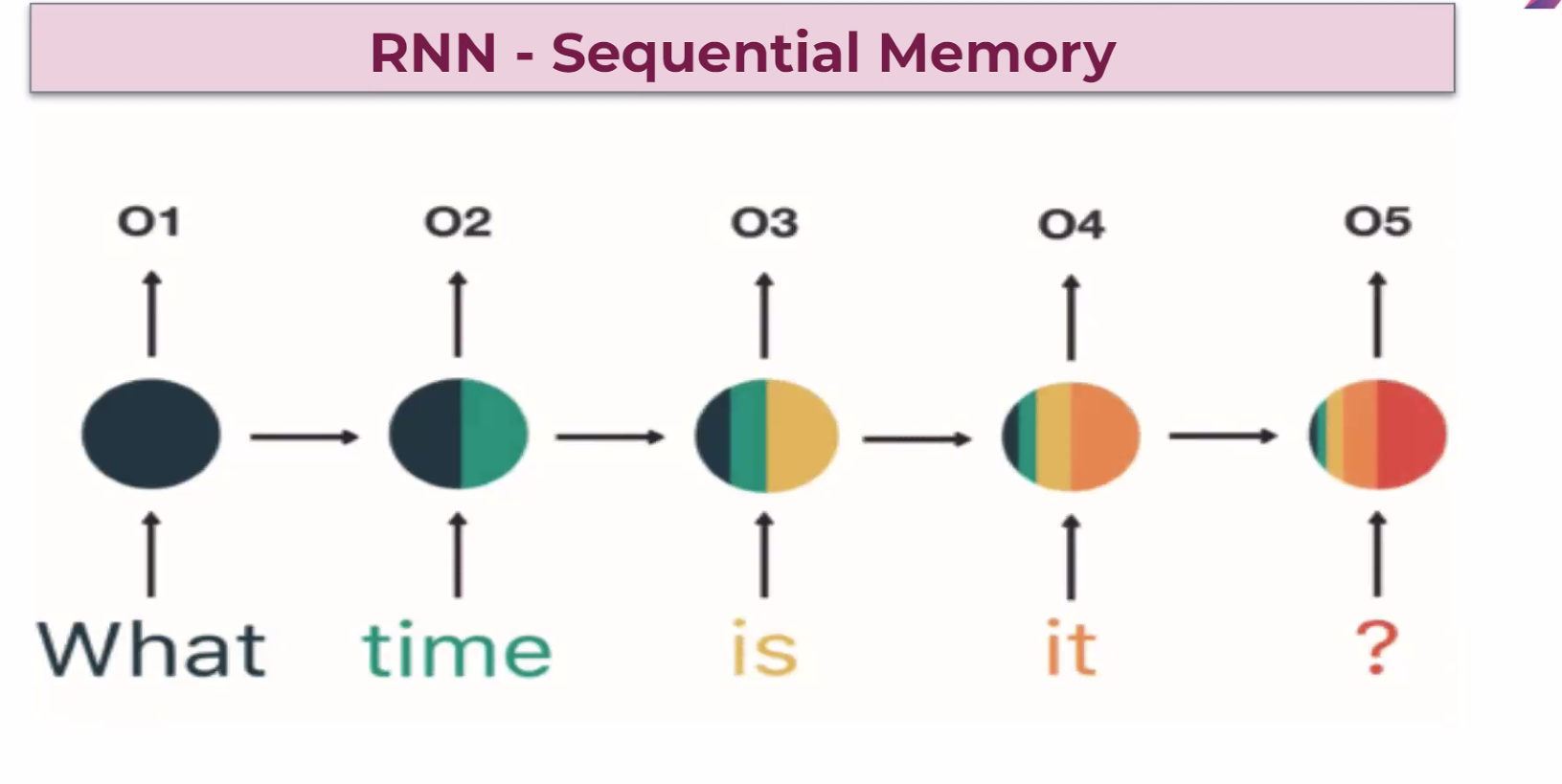
RNN’de ağırlıklar ve biaslar aynı.

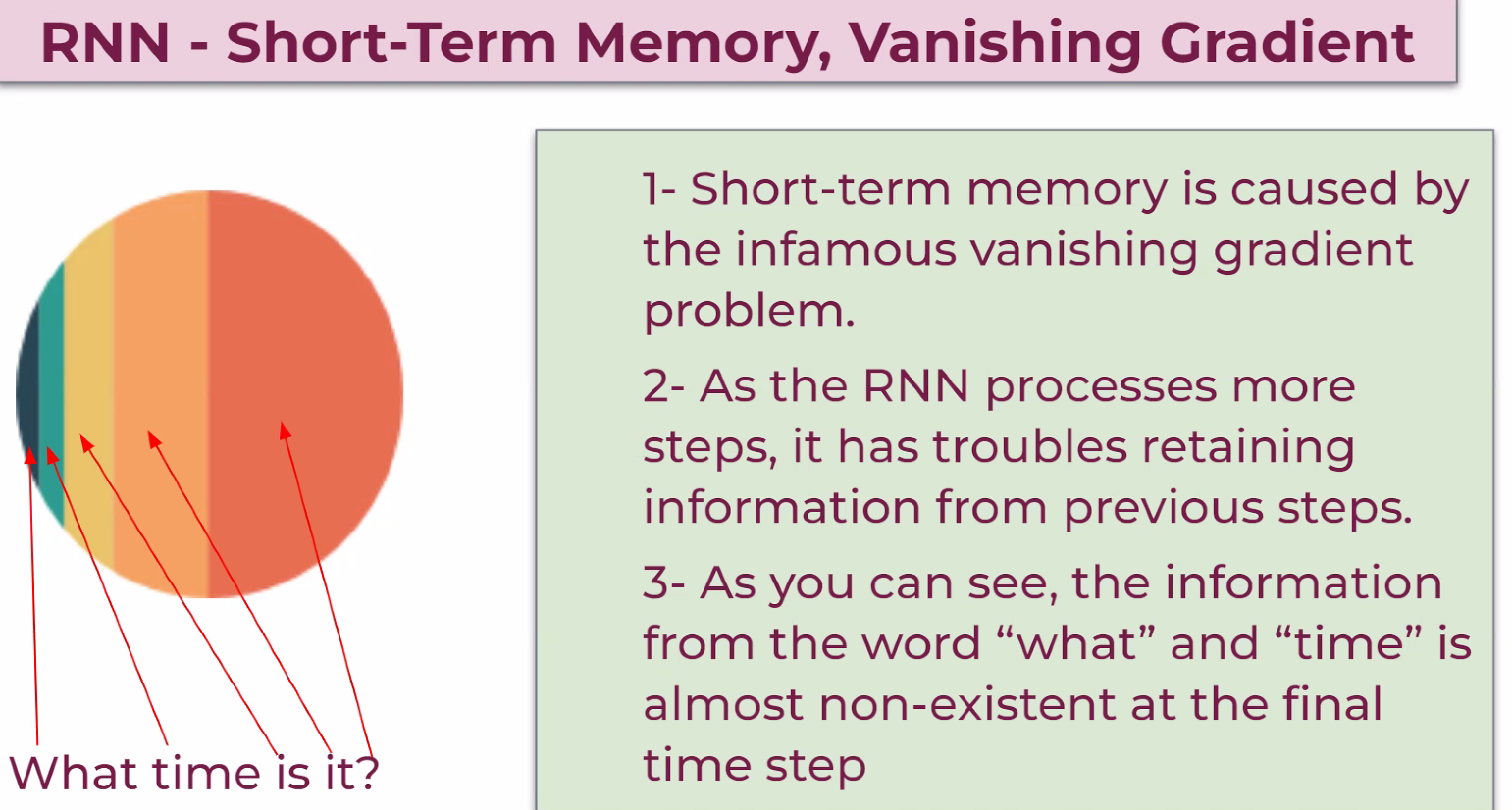
TANH 🡪 Cok buyuk ve cok kucuk sayilari dengelemek icin 🡪 -1 and 1



Tanh bizim için doğal scale işlemini yapıyor aslında.

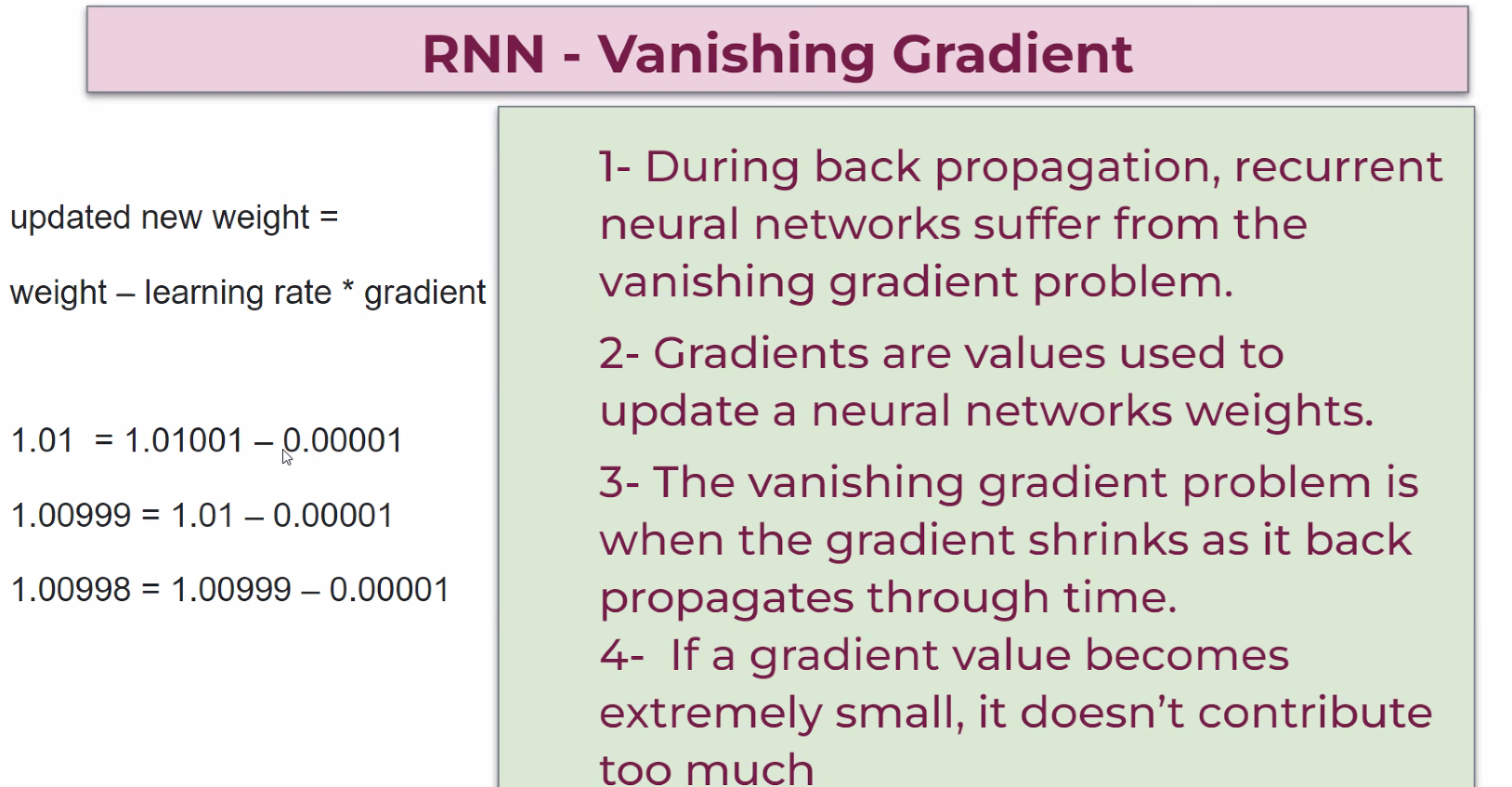
Sigmoid 🡪 Datayı güncellerken veya unuturken 🡪 1 and 0

Gördüğümüz gibi “What time”I neredeyse unutmuş. İşte bu unutma problem oluyor.

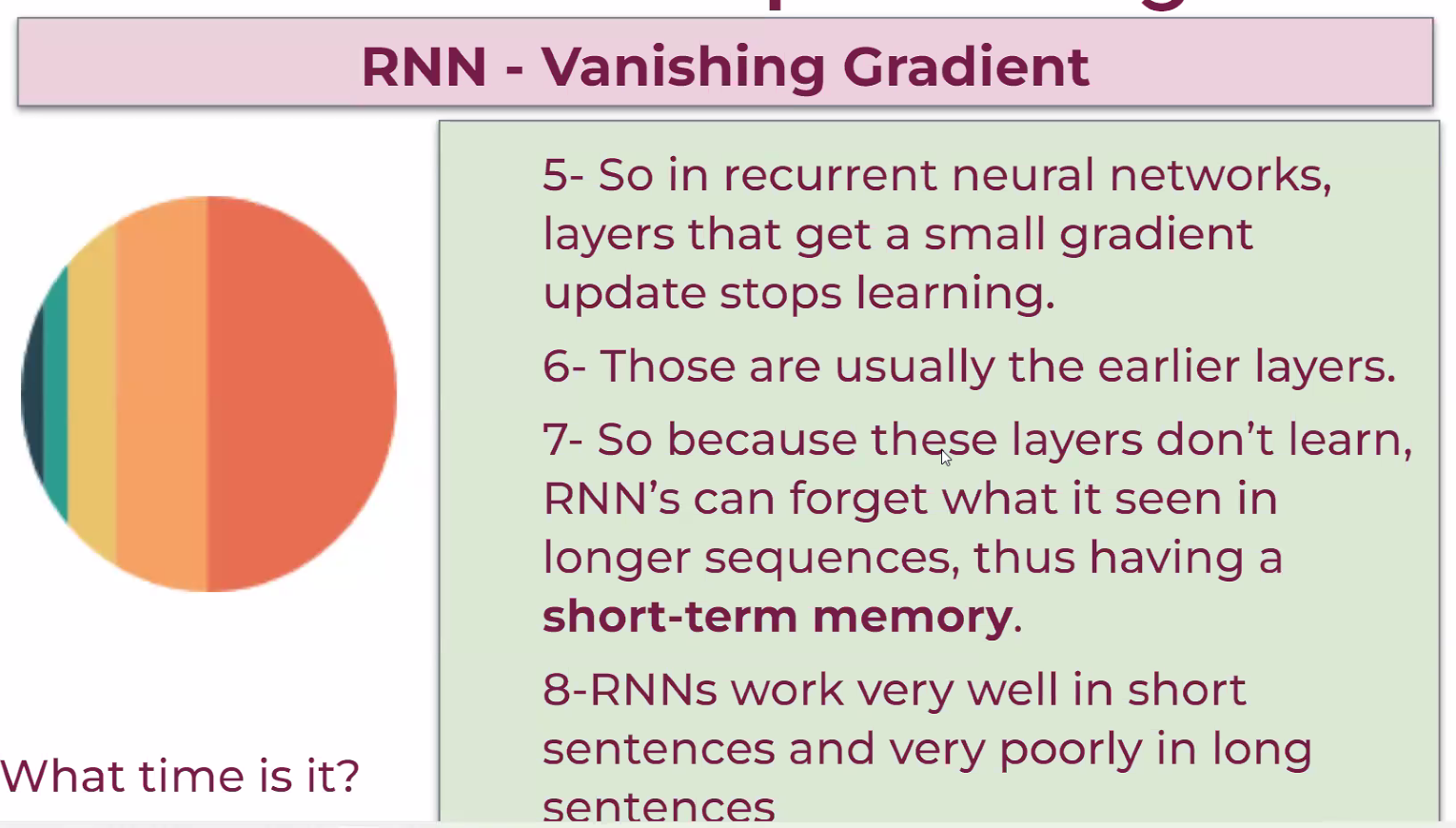


to solve  short term memory problem use LSTM

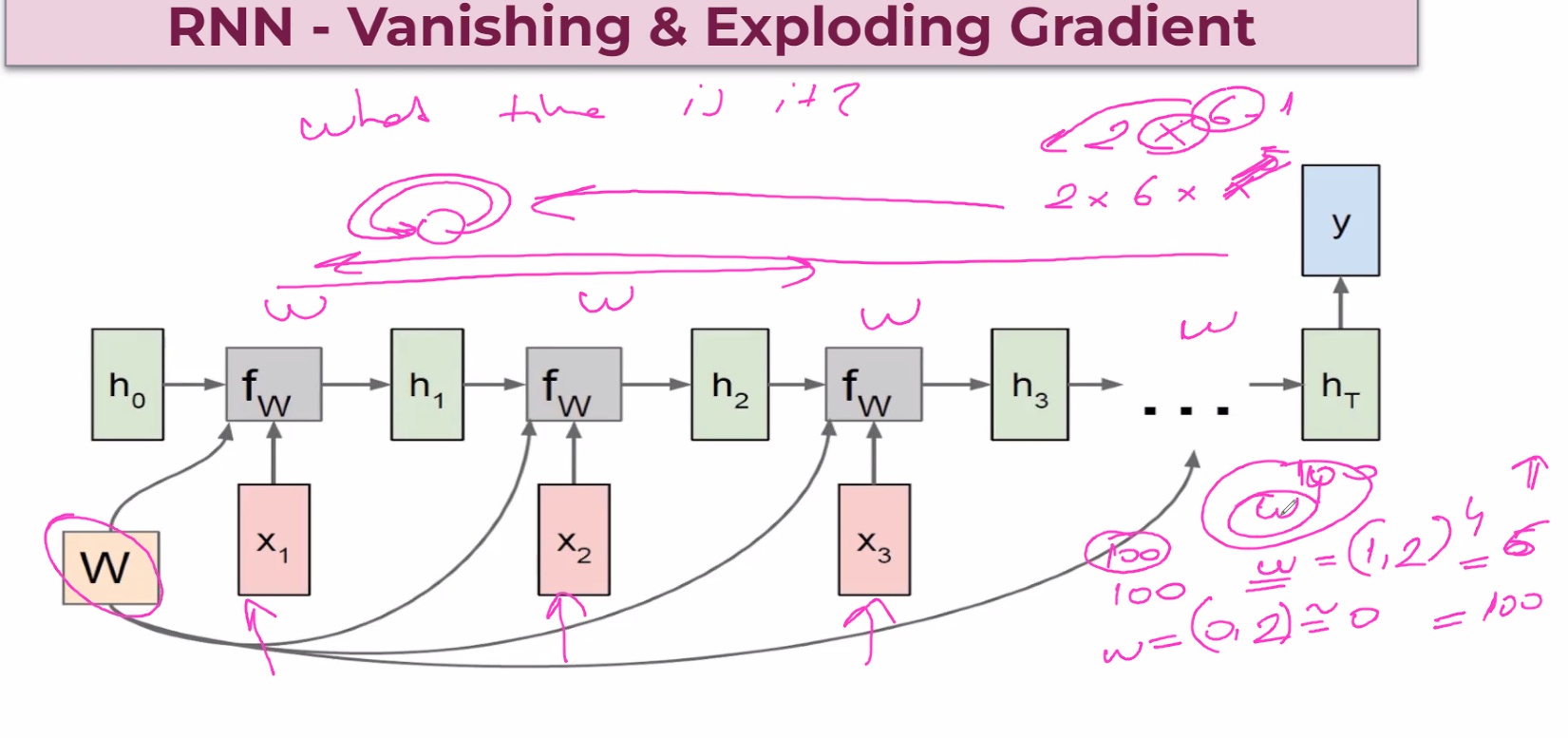
Loss değerlerimiz NaN içeriyorsa kat sayılarımız çok büyüdü ve eğitim durdu demek yanii 🡪 Exploding Gradient



Eğitim var ama artık hiçbir şey değişmiyor 🡪 Vanishing Gradient



Patlama olayı neyden kaynaklanıyor? Şimdi ona gelelim 



Explodingde değerlerimiz çok büyüyor vanishingde de çok küçüyor.

Exploding’de katsayılar o kadar büyür ki tanımlayamaz ve NaN döndürür.

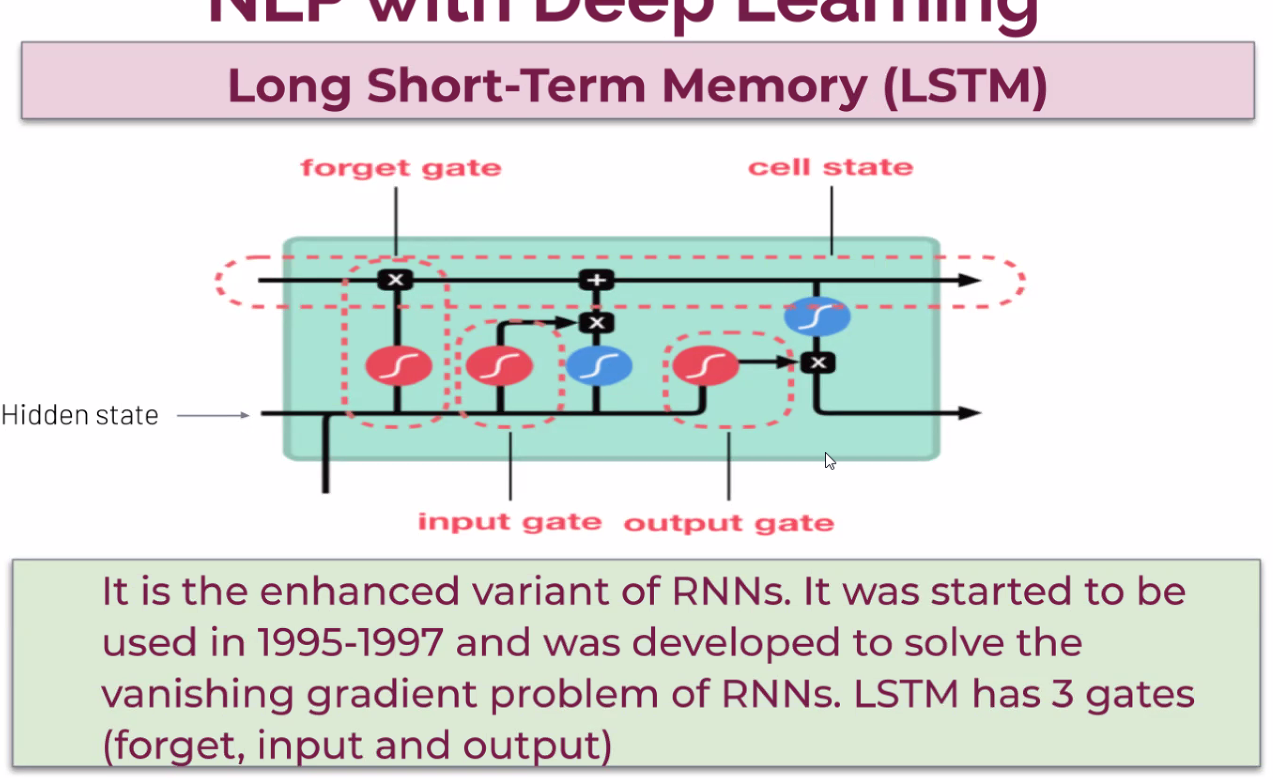
Vanishing’de eğitim devam eder ama katsayılar o kadar az değişir ki bir değişiklik olmaz.

Vanishing de exploding de ikisi de UNUTMA problemidir!!!!

Biz genelde Vanishing Gradient ile karşılaşacağız.



Bu unutma problemlerini de LSTM ve GRU ile çözeceğizz



Long term memory forget rate ve cell state arasında tutuluyor.

Forget rate --> Bize gerekli olanları bırakacak kalanları unutturacak.

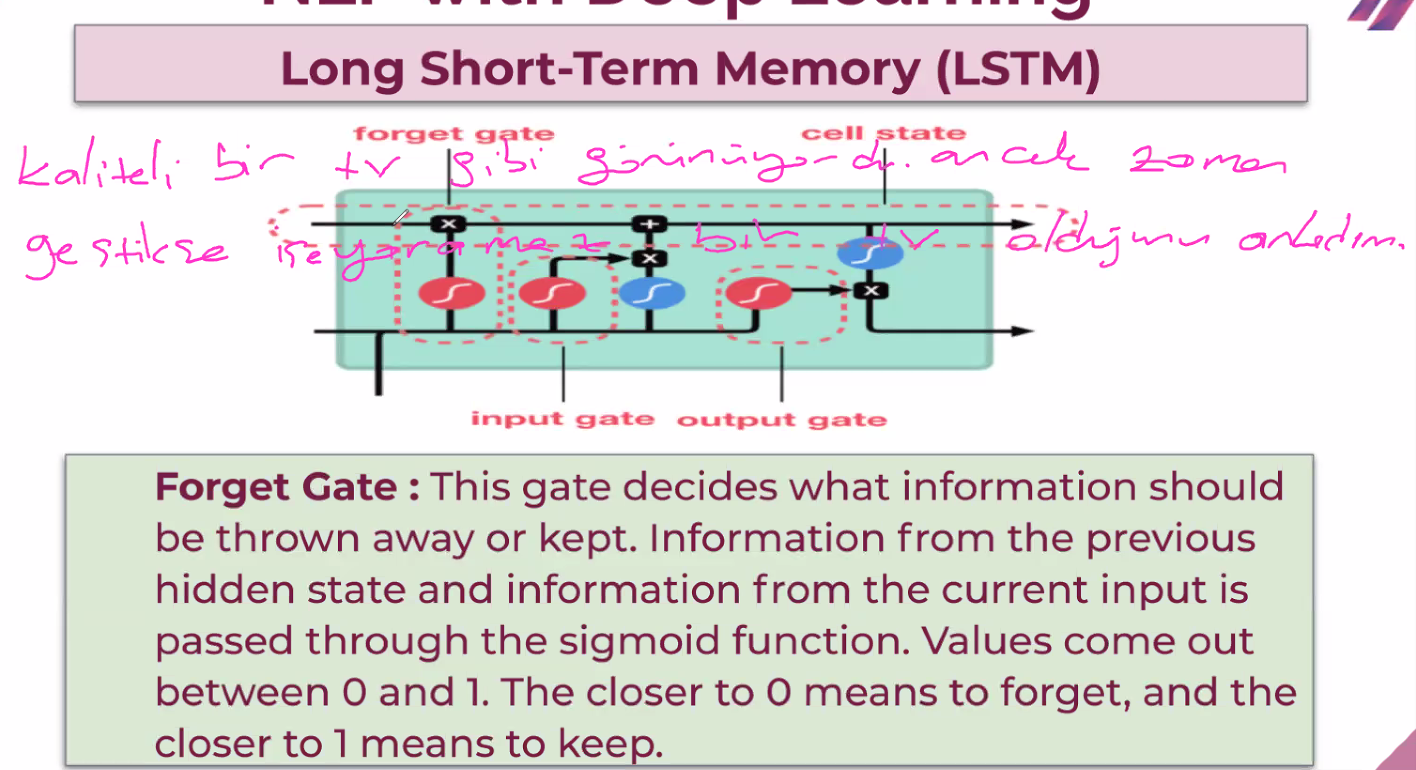
Dün mağazadan aldığım televizyonun görüntü kalitesi mükemmel.

(Burda mesela sadece televizyon, görüntü, kalite ve mükemmel kelimelerini tutacak diğerlerini unutacak)

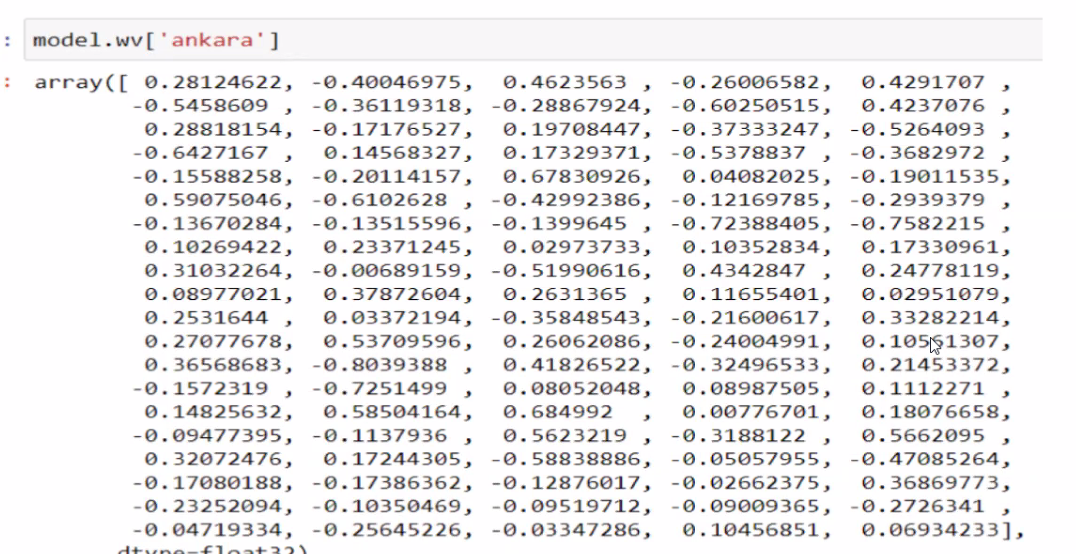
Forget gate cell state’de bulunan (uzun süreli hafızada bulunan) bilgiyi unutturur, onu ordan çıkartır.

Input gate cell state’de bulunmayan bilgiyi oraya ilave eder.

Output ise hidden ve cell state’in buluştuğu yer. Hidden’da hem yararlı hem yararsız bilgiler var. Cell state’de gerekli olanları alarak onları combine ediyor. 🡪 Hidden state’I cell state ile güncellemek yani aslında amaç.

Biz şuan eğitimi tamamlanmış olan modele prediction yaptırıyoruz. Yani burası eğitim kısmı değil, prediction kısmı

[**C9229-Safiyye**](https://app.slack.com/team/U02E2HG7A05)  : “Hocam 50 boyutlu vektör derken corpus içinde o kelime 50 kere geçiyorsa bunların toplam sayısı demek mi?”

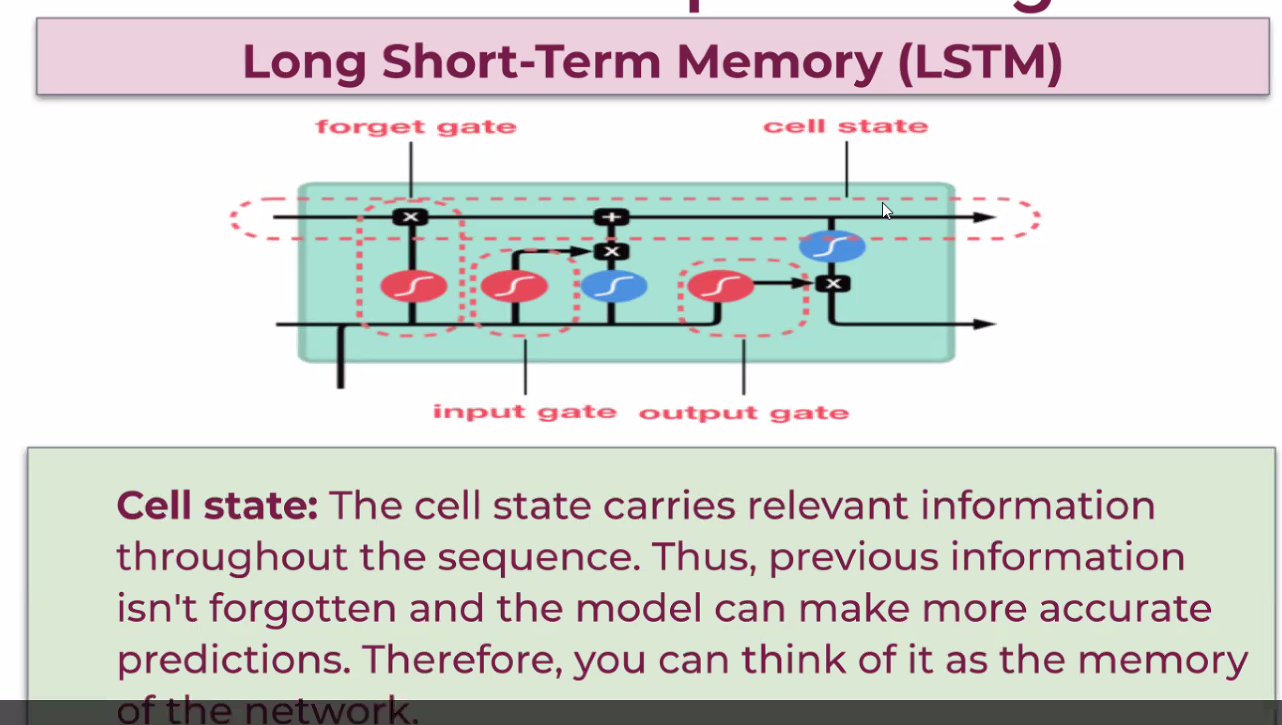
Cevap: No. 50 boyut şu oluyor 🡪

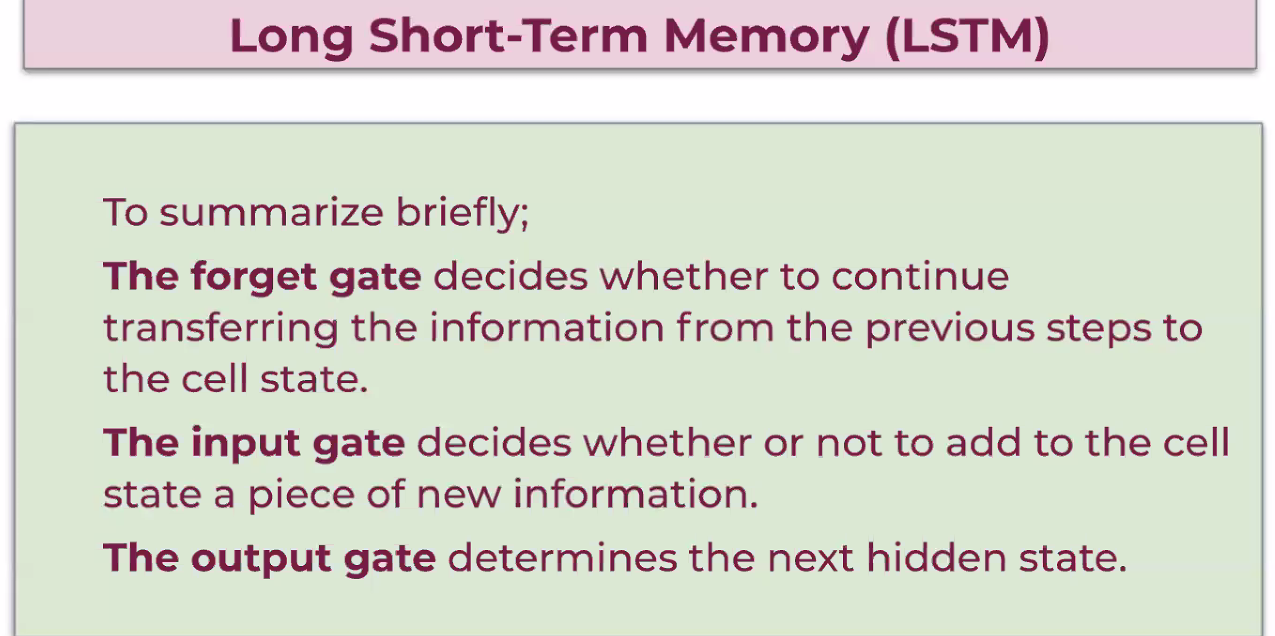
Input gate yeni bilgiyi

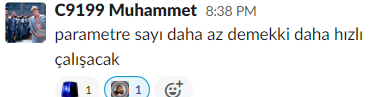
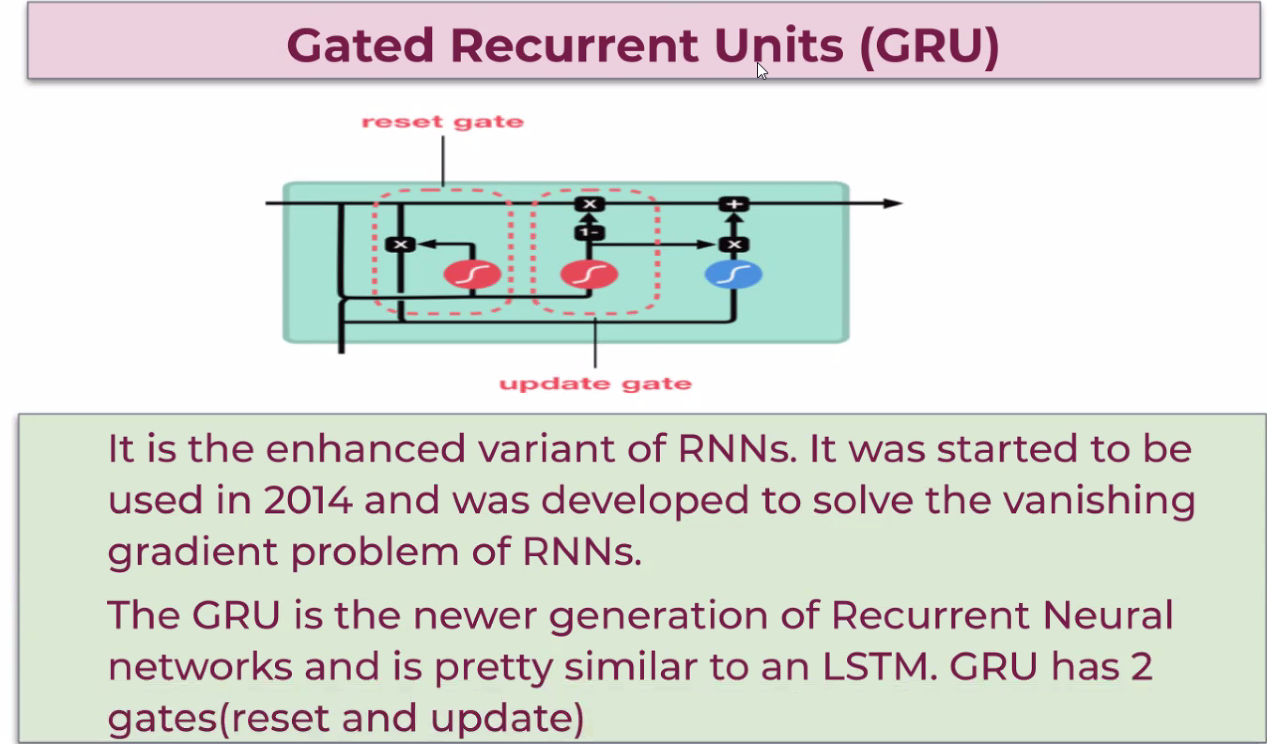
Hem cell state hem hidden state en baştaki girişte birbirlerine eşit oluyorlar.

Cel state = Long term memory

Hidden state = Short term memory



LSTM’in de belli bir sınırı var aktarılacak bilgi konusunda ama transformers alt yapısını kullanan BERT bu konuda kral.

LSTM de GRU da iyidir ama maliyet hesabı olarak GRU daha iyidir. GRU’da LSTM gibi 4 kapımız olmadığı için işlemler daha kısa sürüyor.

