**به نام خدا**

**GRU (Gated Recurrent Unit)** یک نوع ساده‌شده و کارآمد از شبکه‌های عصبی بازگشتی (RNN) است که برای پردازش داده‌های متوالی (مانند متن، سری‌های زمانی، گفتار) طراحی شده. در مقایسه با LSTM، ساختار ساده‌تری دارد اما عملکرد مشابهی ارائه می‌دهد. در اینجا به زبان ساده توضیح می‌دهم:

**مقایسه GRU با LSTM :**

| **ویژگی** | **GRU** | **LSTM** |
| --- | --- | --- |
| **تعداد دریچه‌ها** | ۲ دریچه (Update و Reset) | ۳ دریچه (Input, Forget, Output) |
| **حافظه** | یک حالت (hidden state) | دو حالت (hidden state + cell state) |
| **سرعت آموزش** | سریع‌تر | کندتر |
| **دقت** | معمولاً مشابه LSTM | در برخی موارد بهتر |

### ****ساختار GRU در یک نگاه****:

هر سلول GRU دو دریچه دارد:

1. **دریچه به‌روزرسانی (Update Gate)**:
   * تصمیم می‌گیرد چه اطلاعاتی را حفظ یا دور بریزد.
2. **دریچه بازنشانی (Reset Gate)**:
   * تصمیم می‌گیرد چه اطلاعاتی از گذشته نادیده گرفته شود.

### 💡 ****مثال کاربردی****:

**پیش‌بینی کلمه بعدی در جمله**:

* جمله: "امروز هوا بارانی است، پس من \_\_\_\_ می‌پوشم"
* GRU با توجه به کل جمله پیش‌بینی می‌کند: "بارانی"

### ****پیاده‌سازی ساده در پایتون****:

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import GRU, Dense

model = Sequential([

GRU(64, input\_shape=(10, 1)), *# 64 نورون، 10 گام زمانی، 1 ویژگی*

Dense(1) *# خروجی*

])

### ✅ ****مزایای کلیدی****:

1. **ساده‌تر از LSTM** با حذف cell state.
2. **سرعت آموزش بالاتر** به دلیل پارامترهای کمتر.
3. **حفظ اطلاعات بلندمدت** مانند LSTM.

### ****کاربردهای رایج****:

* ترجمه ماشینی
* تولید متن (چت‌بات‌ها)
* پیش‌بینی قیمت سهام
* تشخیص گفتار

#### **کاربردهای عملی**

* پردازش گفتار در دستیارهای صوتی
* مدل‌سازی زبان برای پیش‌بینی کلمه بعدی

### ****چه زمانی از GRU استفاده کنیم؟****

* وقتی **داده‌های متوالی** دارید.
* وقتی به **سرعت آموزش** بیشتر نیاز دارید.
* وقتی **منابع محاسباتی محدود** دارید

### ****مثال عملی با ارزیابی**** پیش‌بینی سری زمانی:

*# ساخت مدل*

model = Sequential([

GRU(50, activation='tanh', input\_shape=(n\_steps, 1)),

Dense(1)

])

model.compile(optimizer='adam', loss='mse')

*# آموزش*

history = model.fit(X\_train, y\_train, epochs=50, validation\_data=(X\_test, y\_test))

*# ارزیابی*

y\_pred = model.predict(X\_test)

print(f'MSE: {mean\_squared\_error(y\_test, y\_pred):.4f}')

### **جمع‌بندی**: GRU مثل یک **LSTM فشرده** است که با حذف بخش‌های غیرضروری، سرعت را افزایش داده اما همچنان قدرت یادگیری وابستگی‌های بلندمدت را حفظ می‌کند. برای شروع کار با داده‌های متوالی، GRU انتخاب عالی‌ای است! 😊

#### **کد کامل برای تحلیل احساسات**

from tensorflow.keras.layers import GRU, Embedding

model = Sequential([

Embedding(input\_dim=10000, output\_dim=128, input\_length=100),

GRU(128, return\_sequences=True),

GRU(64),

Dense(1, activation='sigmoid')

])

model.compile(optimizer='adam',

loss='binary\_crossentropy',

metrics=['accuracy'])

#### آموزش روی دیتاست **IMDB**

from tensorflow.keras.datasets import imdb

(X\_train, y\_train), (X\_test, y\_test) = imdb.load\_data(num\_words=10000)

model.fit(X\_train, y\_train, epochs=5, batch\_size=128)

**Dropout:**

GRU(64, dropout=0.2, recurrent\_dropout=0.2)

**Batch Normalization:**

from tensorflow.keras.layers import BatchNormalization

model.add(BatchNormalization())

**راه‌اندازی Early Stopping:**

from tensorflow.keras.callbacks import EarlyStopping

callbacks = [EarlyStopping(patience=3)]

#### **پروژه ۱: سیستم تشخیص اسم موجودیت‌ها (NER)**

from tensorflow.keras.layers import TimeDistributed

model = Sequential([

Embedding(5000, 128),

GRU(256, return\_sequences=True),

TimeDistributed(Dense(5, activation='softmax')) *# 5 کلاس برای موجودیت‌ها*

])

#### **پروژه ۲: چت‌بات ساده**

*# تبدیل متن به دنباله‌های عددی*

tokenizer = Tokenizer(char\_level=True)

tokenizer.fit\_on\_texts(texts)

sequences = tokenizer.texts\_to\_sequences(texts)

*# معماری مدل*

model.add(GRU(256, input\_shape=(None, len(chars))))

model.add(Dense(len(chars), activation='softmax'))