Intro to TensorFlow

הטכניון
היחידה ללימודי המשך
ולימודי חוץ ע"ש עזריאלי

TECHNION
Azrieli Continuing Education and
External Studies Division

- **The difference compared to PyTorch**
- **Steps in model building**

TensorFlow היא ספריית **קוד פתוח** ללמידת מכונה **ולמידה עמוקה**

פותחה על ידי **Google**.

מאפשרת בנייה, אימון ופריסה של רשתות נוירונים בצורה גמישה ומותאמת ליישומים

תעשייתיים.

אחת מנקודות החוזק של TensorFlow היא **שימוש בגרף חישובי** (**Computational**

Graph) המאפשר ביצועים אופטימליים בפריסות Tensor Processing Units

בהשוואה ל- PyTorch, שהוא דינמי ומבוסס על גישה אימפרטיבית- TensorFlow, היה בעבר סטטי יותר, אך עם **TensorFlow 2.0** נוספה ברירת המחדל של **Eager Execution**, מה שהופך את העבודה בו ליותר אינטראקטיבית וידידותית. עדיין, TensorFlow נחשב מתאים יותר לסביבות **תעשייתיות ופרודקשן**, בזכות כלים כמו **TensorFlow Serving**ו- **TensorFlow Lite** לפריסה במכשירים ניידים. מנגד, **PyTorch נחשב אינטואיטיבי יותר למחקר ולפיתוח מודלים חדשים**, והוא הבחירה העיקרית באקדמיה.

בסופו של דבר, **TensorFlow מתאים לפרויקטים הדורשים ביצועים גבוהים ופריסה מסודרת**,

בעוד ש- PyTorchמאפשר גמישות ופשטות בשלבי המחקר והניסוי.

איך זה היה ב PyTorch	איך זה ב TensorFlow	מה זה?
tensor.view(shape)	tf.reshape(tensor, shape)	משנה את צורת הטנסור מבלי לשנות את הנתונים
tensor.reshape(shape)	tf.reshape(tensor, shape)	מקביל ל- view ב-PyTorch
tensor.permute(dims)	tf.transpose(tensor)	משנה סדר של צירים (axis)
tensor.T	tf.transpose(tensor)	טרנספוזה (עבור טנסור דו-ממדי)
tensor.squeeze()	tf.squeeze(tensor)	מסיר ממדים בגודל 1
tensor.unsqueeze(dim)	tf.expand_dims(tensor, axis=dim)	מוסיף ממד חדש

```
tensor
```

```
<tf.Tensor: shape=(2, 3, 4), dtype=int32, numpy=
array([[[  1,   2,   3,   4],
        [  5,   6,   7,   8],
        [  9,  10,  11,  12]],

       [[ 10,  20,  30,  40],
        [ 50,  60,  70,  80],
        [ 90, 100, 110, 120]]])>
```

```
tf.reduce_max(tensor)
```

```
<tf.Tensor: shape=(), dtype=int32, numpy=120>
```

```
tf.reduce_max(tensor, axis =0)
```

```
<tf.Tensor: shape=(3, 4), dtype=int32, numpy=
array([[ 10,  20,  30,  40],
       [ 50,  60,  70,  80],
       [ 90, 100, 110, 120]])>
```

פונקציה	מה היא עושה?
tf.reduce_sum(tensor)	מחזירה את סכום כל הערכים
tf.reduce_mean(tensor)	מחזירה את הממוצע
tf.reduce_min(tensor)	מחזירה את הערך המינימלי
tf.reduce_prod(tensor)	מחזירה את מכפלת כל הערכים
tf.reduce_std(tensor)	מחשבת סטיית תקן (בגרסאות חדשות)
tf.reduce_variance(tensor)	מחשבת שונות

- stack –מוסיף ממד חדש.
- Concat – מחבר לאורך ציר קיים.
- vstack – במו concat(axis=0)

```
tf.stack([tensor1, tensor2], axis=0)
```

```
tf.concat([tensor1, tensor2], axis=1)
```

```
tf.concat([tensor1, tensor2], axis=0)
```

```python
# מקומי seed בלי shuffles נריץ כמה
print("--- מקומי seed ללא ---")
shuffle1 = tf.random.shuffle(data)
shuffle2 = tf.random.shuffle(data)
print("Shuffle1:", shuffle1.numpy())
print("Shuffle2:", shuffle2.numpy())

# גלובלי seed נאפס ונריץ שוב עם אותו
tf.random.set_seed(42)
print("\n--- מקומי seed אחרי איפוס, עדיין בלי ---")
shuffle3 = tf.random.shuffle(data)
shuffle4 = tf.random.shuffle(data)
print("Shuffle3:", shuffle3.numpy())
print("Shuffle4:", shuffle4.numpy())
```

```
--- מקומי seed ללא ---
Shuffle1: [7 6 3 0 8 9 5 4 1 2]
Shuffle2: [8 5 1 4 0 3 6 9 2 7]

--- מקומי seed אחרי איפוס, עדיין בלי ---
Shuffle3: [7 6 3 0 8 9 5 4 1 2]
Shuffle4: [8 5 1 4 0 3 6 9 2 7]
```

התנהגות הרנדומלית ב tensor flow מוגדרת עי זרע גלובלי שמאתחל זרם רנדומלי.

ניתן להציב גם זרע מקומי לפעולות ספציפיות.

מרגע שנקבע ואותחל זרע רנדומלי התוצאה הראשונה בהרצה תהיה זהה להרצה שניה בכל הרצה,

והתוצאה השניה תהיה זהה לתוצאה השניה בכל הרצה וכן הלאה, אך התוצאה הראשונה לא תהיה זהה לתוצאה השניה.

כדי לשחזר את התוצאה הראשונה יש לשחזר את הזרע הגלובלי

הזרע המקומי נועד לסייע לשלוט בהתנהגות זו, ולשמור נקודה

רנדומלית בצורה יותר מסויימת

```
tf.random.set_seed(42)
print("\n--- עם seeds שונים מקומיים ---")
shuffle5 = tf.random.shuffle(data, seed=123)
shuffle6 = tf.random.shuffle(data, seed=456)
print("Shuffle5 (seed=123):", shuffle5.numpy())
print("Shuffle6 (seed=456):", shuffle6.numpy())
```

```
--- עם seeds שונים מקומיים ---
Shuffle5 (seed=123): [1 3 5 2 9 4 0 8 7 6]
Shuffle6 (seed=456): [8 6 2 1 9 7 3 5 4 0]
```

```

tf.random.set_seed(42)
print("\n--- עם seeds שונים מקומיים ---")
shuffle5 = tf.random.shuffle(data, seed=456)
shuffle6 = tf.random.shuffle(data, seed=123)
shuffle7 = tf.random.shuffle(data, seed=123)
print("Shuffle5 (seed=456):", shuffle5.numpy())
print("Shuffle6 (seed=123):", shuffle6.numpy())

```

```
--- עם seeds שונים מקומיים ---
Shuffle5 (seed=456): [8 6 2 1 9 7 3 5 4 0]
Shuffle6 (seed=123): [1 3 5 2 9 4 0 8 7 6]
```

1. הגדרת המודל:

בשלב זה יוגדרו השכבות השונות, גודלן ופונקציות אקטיבציה

```
# Build Model 🚀
model = tf.keras.Sequential ([
    tf.keras.Input(shape=(3,)),
    tf.keras.layers.Dense (100,activation="relu"),
    tf.keras.layers.Dense(1,activation=None)
])
```

2. קומפילציית המודל:

בשלב זה יוגדרו פונקציית העלות, אופן המדידה והאופטימייזר

```
# Compile Model
model.compile(
    loss=tf.keras.losses.mae,
    optimizer= tf.keras.optimizers.Adam(learning_rate=0.1),
    metrics = ["mae"]
)
```

3. אימון המודל:

בשלב זה ינתנו למודל נתוני האימון וישמרו מדדיו.

כמו כן בשלב זה יגדרו מספר הצעדים (איפוקים))

```
# Train Model 🎯
history = model.fit(X_train,y_train, epochs=5)
```

בשלב זה אפשר גם לקבוע את גודל המנה (batch) אם יש,
ואת רמת הפלט

verbose: `"auto"`, 0, 1, or 2. Verbosity mode.
0 = silent, 1 = progress bar, 2 = one line per epoch.

```
# Train Model 🎯
history = model.fit(X_train,y_train, epochs=5)
```

*** מה יש בתוך history:**

history הוא אובייקט שמחזירה הפונקציה model.fit()

history.history הוא dictionary שמכיל איברים עבור כל מדד שעקבנו אחריו באימון.

מאפשר לנתח ולצייר גרפים של ביצועי המודל לאורך הזמן.