עבודה עם
TensorFlow

הטכניון
היחידה ללימודי המשך
ולימודי חוץ ע"ש עזריאלי

TECHNION
Azrieli Continuing Education and
External Studies Division

- סיכום שלבי בניית מודל
- סוגי מודלים והיפר פרמטרים המתאימים להם
- callbacks

```python
# 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)
])
```

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

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

1. **הגדרת המודל:**

בשלב זה יוגדרו השכבות השונות, גודלן ופונקציות אקטיבציה

2. **קומפילציית המודל:**

בשלב זה יוגדרו פונקציית העלות, אופן המדידה והאופטימייזר

3. **אימון המודל:**

בשלב זה ינתנו למודל נתוני האימון וישמרו מדדיו.

כמו כן בשלב זה יגדרו מספר הצעדים (איפוקים))

4. הערכת המודל:

המודל ירוץ epoch אחד לפנים ללא חזרה לאחור

וביצוע תיקונים (בלומר ללא למידה) ויחזיר עלות

ואמצעי מדידה (metric)

```
#Model Evaluation ✔

loss, metric = model.evaluate(X_test,y_test)
```

5. יצירת תחזית:

משתמשים במודל הסופי כדי לחזות תוצאות, גם בשלב

זה המודל לא לומד.

```
#5 Predicted Price 🔮
predicted_price = model.predict(new_X_shaped_date)
```

מידע על המודל (השכבות, הפרמטרים וכו'

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 4)	12
dense_1 (Dense)	(None, 4)	20
dense_2 (Dense)	(None, 1)	5

Total params: 113 (456.00 B)

Trainable params: 37 (148.00 B)

Non-trainable params: 0 (0.00 B)

Optimizer params: 76 (308.00 B)

פרמטר	ערך אופייני
שכבת קלט	מס' נויירונים שווה למספר הפיצ'רים (מספר העמודות של X)
שכבות נסתרות	ML: 1-3 DL : כמה שצריך... (בד"כ 1-3 לבעיות פשוטות ; 5-6 לבעיות מורכבות)
פונקציות אקטיבציה בשכבות נסתרות	הכי נפוץ: ReLU אם יוצר בעיה אפשר להחליף בשכבות דומות כמו למשל LeakyReLU TanH\Sigmoid- במצבי מחזוריות למשל RNN
מס' נוירונים בשכבה	תלוי במספר הפיצ'רים תלוי במורכבות המידע להתחיל במספר גדול יותר בשכבות הראשונות, ואז להקטין בהדרגה.
optimizer	Adam/SGD (ניתן להשתמש גם ב- RMSprop, Adagrad וכו' בהתאם למקרה).

multi label	multi class	binary	רגרסיה	פרמטר
כמספר ה labels (כל label הוא סיווג בינארי עצמאי)	כמספר ה classes	1	1	**מספר נוירונים בשכבת הפלט**
סיגמואיד (לעיתים קיים באופן לא מפורש בפונק' ההפסד)	softmax	סיגמואיד (לעיתים קיים באופן לא מפורש בפונק' ההפסד)	ללא	**פונקציית אקטיבציה של שכבת הפלט**
cross entropy	Cross Entropy	Binary Cross Entropy אם היה סיגמואיד באקטיבציה של שכבת הפלט או Binary Cross Entropy with Logistic- במקום סגמואיד בשכבת הפלט	MSE- הכי נפוץ MAE - מתאים לערכים עם רעש גבוה Huber - מאזן את שני הנ"ל	**loss function**

Callbacks

פונקציות מובנות של TensorFlow העוזרות לשלוט בתהליך האימון. לדוגמא:

- **EarlyStopping** – עוצר אימון אם אין שיפור.
- **ReduceLROnPlateau** – מקטין learning rate אם אין שיפור.
- **ModelCheckpoint** – שומר את המודל הכי טוב.
- **LearningRateScheduler** – משנה learning rate בכל אפוק.
- **CSVLogger** – שומר תוצאות אימון בקובץ CSV.
- **TerminateOnNaN** – עוצר אימון אם יש NaN בתוצאות.
- **BackupAndRestore** – שומר התקדמות ומאפשר שחזור אם האימון נקטע.
- **History** – שומר את ההיסטוריה של האימון (נמצא כברירת מחדל).

Callbacks

איך משתמשים:

יוצרים משתנה ומאתחילים בו את הפרמטרים הדרושים לפי סוג ה callback הנבחר למשל:

```
callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience = 5)
```

או

```
callback = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', patience = 3,factor =0.5)
```

מציבים את המשתנה בתוך model.fit בערך של הפרמטר callbacks

```
# train the model
history = model.fit(X_train, y_train, epochs=100,
                    batch_size=32,
                    validation_data=(X_test, y_test), verbose=0,
                   callbacks = [callback])
```

אם רוצים אפשר ליצור משתנה שהוא רשימה של כמה callbacks

```
callback = [tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience = 5),
            tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', patience = 3,factor =0.5)]
```