# שכבות

אבן הבניין הבסיסית של רשת נוירונים היא השכבה. רוב הלמידה העמוקה מורכבת משרשור שכבות. בראש התוכנית יש לייבא את השכבות באמצעות הפקודה:

from tensorflow.keras import layers

### layers.Flatten

הופך את הקלט ממערך רב ממדי למערך חד ממדי. לשכבה זו אין פרמטרים ללמוד, אלא רק מעצבת מחדש את הנתונים.

### layers.Dense

מייצגת שכבת fully connected. מקבלת את מספר הנוירונים בפלט של שכבה זו, פונקציית אקטיבציה ופרמטרים נוספים.

### layers.TextVectorization

מודל

כדי ליצור מודל מאומן ב-Keras שניתן להריץ עליו אובייקטים ולקבל חיזוי, יש לדעת לבצע חמש פעולות:

1. יצירה – הגדרת רצף השכבות במודל.
2. קמפול – הגדרת פונקציית loss, אופטימייזר ומה רוצים למקסם.
3. אימון – נאמן המודל באמצעות הדאטה.
4. הערכה – באמצעות השוואה לקבוצת ה-test.
5. חיזוי – נתינת אובייקט חדש וקבלת חיזוי.

## יצירת מודל

כדי ליצור מודל שהוא שרשור של שכבות נשתמש בפונקציה tf.keras.Sequential המקבלת מערך של שכבות ומחזירה מודל.

model = tf.keras.Sequential([

tf.keras.layers.Flatten(input\_shape=(28, 28)),

tf.keras.layers.Dense(128, activation='relu'),

tf.keras.layers.Dense(10)

])

## קמפול

באמצעות הפעלת הפונקציה compile על המודל ניתן להגדיר למודל את הרכיבים הבאים:

* פונקציית loss - מודד עד כמה המודל מדויק במהלך האימון. אנו רוצים מינימום של הפונקציה הזו כדי לכוון את המודל בכיוון הנכון.
* אופטימייזר - באיזה שיטה לעדכן את הפרמטרים של המודל על סמך הנתונים שהוא רואה פונקציית ה-loss.
* מדדים - רשימת מדדים להערכת המודל במהלך אימון ובדיקות. בדרך כלל accuracy.

model.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True),

metrics=['accuracy'])

## אימון

באמצעות הפעלת הפונקציה fit נאמן את המודל. פונקציה זו צריכה לקבל את:

* x – קבוצת המידע עליו המודל מתאמן.
* y – התיוג של קבוצת האימון.
* batch\_size - מספר שלם או None. מייצג את מספר הדגימות בכל עדכון של הגרדיאנט. אם לא צוין, ברירת המחדל 32.
* epochs – מספר האיטרציות שעוברים על כל הדאטה כדי לאמן את במודל.

model.fit(train\_images, train\_labels, epochs=10)

## הערכה

באמצעות הפונקציה evaluate על המודל נוכל להעריך אותו. הפונקציה מקבלת את קבוצת ה-test ואת התיוג שלה, ומחזירה את הערך הסופי של הפונקציית loss ואת המדדים שהכנסנו למודל במהלך הקמפול.

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels)

## חיזוי

נפעיל את הפונקציה predict על המודל כדי לקבל חיזוי עבור קבוצת אובייקטים. הפונקציה מקבלת מערך של אובייקטים x ומחזירה מערך numpy של החיזויים לכל הקלטים.

במקרה שהמודל צריך לסווג אובייקט ליותר משני מחלקות הוא מחזיר logits לכל מחלקה, ולכן יש להפעיל שכבת softmax כדי להמיר את ה-logits להסתברויות.

probability\_model = tf.keras.Sequential([model,

tf.keras.layers.Softmax()])

predictions = probability\_model.predict(test\_images)

np.argmax(predictions[0])

# עיבוד DATA

## העלאת קובץ

### באמצעות כתובת UML

להוריד קובץ באמצעות כתובת URL, אם אינו נמצא כבר ב-cache נשתמש בפונקציה tf.keras.utils.get\_file. הפונקציה מחזירה Path לקובץ שהורדנו. הארגומנטים של פונקציה זו הם:

* Fname - שם הקובץ. אם צוין כל ה-path בערך זה, לדוגמה /path/to/file.txt, הקובץ יישמר במיקום זה. אם שדה זה ריק, השם של הקובץ יהיה לפי הערך ב-origin.
* origin – כתובת url של הקובץ.
* extract – ערך בוליאני. אם True, מנסה לחלץ את הקובץ כארכיון, כמו tar או zip.
* cache\_dir - מיקום לאחסון הקבצים. ברירת המחדל היא ~/.keras/.

## יצירת Dataset מקבצי טקסט

באמצעות הפונקציה [text\_dataset\_from\_directory](https://www.tensorflow.org/api_docs/python/tf/keras/utils/text_dataset_from_directory) ניתן לטעון נתונים מהדיסק ולהכין אותם לפורמט המתאים לאימון. הפונקציה שימושית כאשר הדאטה מסודר כך:

main\_directory/

class\_a/

a\_text\_1.txt

a\_text\_2.txt

class\_b/

b\_text\_1.txt

b\_text\_2.txt

קריאה לפונקציה text\_dataset\_from\_directory(main\_directory, …) מחזירה אובייקט tf.data.Dataset המכיל קבוצות של טקסטים מהתיקיות class\_a ו-class\_b, יחד עם התוויות 0 ו-1 (0 ל-class\_a ו-1 ל-class\_b). בנוסף מודיעה כמה אובייקטים ומחלקות נמצאו. בשלב זה רק קבצי txt נתמכים.

הארגומנטים של פונקציה זו הם:

* + - directory – path למיקום של ה-data.
    - batch\_size - מספר שלם או None. מייצג מספר הדגימות בכל עדכון של הגרדיאנט. אם לא צוין ברירת המחדל 32.
      * validation\_split – כמה אחוז מה-train ישמש לקבוצת ה-validation.
    - subset – משמש רק אם validation\_split מוגדר. מגדיר האם האובייקט המוחזר הוא “training” או “validation”.
    - shuffle - האם לערבב את הנתונים. ברירת מחדל היא true. אם מוגדר כ-false, ממיין את הנתונים בסדר אלפאנומרי.
    - seed – ערך אקראי עבור ערבוב הדאטה.

בעת שימוש בארגומנטים validation\_split ו-subset, צריך לציין סיד אקראי, או לציין shuffle=False, כך שלפיצול לא יהיה חפיפה.

raw\_train\_ds = tf.keras.preprocessing.text\_dataset\_from\_directory(

'data/train',

validation\_split=0.2,

subset='training',

seed=42)

raw\_val\_ds = tf.keras.preprocessing.text\_dataset\_from\_directory(

'data/train',

validation\_split=0.2,

subset='validation',

seed=42)