CNN בסיסי   
**תהליך כללי של המודל מתחילתו ועד סופו**

הפרויקט נועד לנתח את הקשר בין רצפי DNA לבין עוצמת הקישור שלהם לחלבונים מסוימים, באמצעות מודל רשת נוירונים קונבולוציונית (CNN). המטרה היא לקחת נתונים מ-SELEX, סוג של ניסוי המיועד למדוד את האפיניות של רצפי DNA לחלבונים, ולנסות לחזות את עוצמת הקישור של רצפים דומים בנתונים מ-RNAcompete.

**שלב 1: הכנת נתוני ה-SELEX**

בשלב הראשון, אנחנו עובדים עם נתוני SELEX, אשר מחולקים למספר סבבים (Cycles). עבור כל רצף בנתונים, אנחנו מציינים את מספר ההופעות שלו בכל סבב. רצפים מהסבב האחרון נחשבים כרצפים בעלי האפיניות הגבוהה ביותר לחלבון, ולכן הם מתויגים בתווית חיובית (1). רצפים מהסבב הראשון, שלא מופיעים בסבבים אחרים, מתויגים בתווית שלילית (0). אנחנו בוחרים כ-50,000 רצפים חיוביים ושליליים לאימון המודל.

**שלב 2: הכנת נתוני ה-RNAcompete**

לאחר שהמודל אומן על נתוני ה-SELEX, אנו רוצים להשתמש בו כדי לחזות את עוצמת הקישור של רצפי DNA חדשים הנמצאים בנתוני RNAcompete. תחילה אנו ממירים את הרצפים ב-RNAcompete לייצוג מתאים באמצעות one-hot encoding, המאפשר למודל לעבד את המידע בצורה נוחה.

**שלב 3: בניית המודל (CNN)**

אנו בונים רשת נוירונים קונבולוציונית (CNN) כדי לעבד את הנתונים ולהפיק תחזיות. הרשת מורכבת ממספר שכבות קונבולוציה עם מסננים (kernels) בגודל 5, שמטרתן לזהות תבניות ברצפי ה-DNA באורך של חמישיות בסיסים (k-mers). לאחר מכן, שכבות Fully Connected מתווספות כדי לבצע סיווג סופי של הרצפים.

**שלב 4: אימון המודל**

אנו מאמנים את המודל על ה-50,000 רצפים שנבחרו מהסבבים של SELEX, תוך שימוש במטריקות של דיוק (Accuracy) ושטח מתחת לעקומה (AUC) כדי להעריך את הביצועים. תהליך האימון כולל גם עצירת מוקדמת כדי למנוע Overfitting.

**שלב 5: חיזוי עם RNAcompete**

לאחר שהמודל אומן, אנו משתמשים בו כדי לחזות את עוצמת הקישור של רצפי ה-RNAcompete. תחזיות אלו נשמרות בקובץ לשימושים עתידיים.

**שלב 6: ניתוח תוצאות**

בסופו של דבר, אנו מנתחים את התוצאות שהתקבלו על מנת להבין את רמת הדיוק של המודל, כולל חישוב המתאם בין התחזיות לבין הערכים בפועל. אנו בודקים האם המודל מצליח לזהות נכון את עוצמות הקישור ומשווים את ביצועי המודל עם תוצאות אחרות כדי להבין את האפקטיביות של הגישה.

**תוצאה סופית**

בסוף התהליך, יש לנו מודל שמסוגל לחזות את עוצמת הקישור של רצפי DNA לחלבונים על בסיס נתוני SELEX, ולאחר מכן להשתמש במודל זה כדי לנתח נתונים חדשים מ-RNAcompete

חלוקה לשלבים :

* בניית מודל CNN

```python

def build\_model(input\_shape):

model = models.Sequential()

# First convolutional layer with kernel size 5 to capture five-mer patterns

model.add(layers.Conv1D(filters=128, kernel\_size=5, activation='relu', input\_shape=input\_shape))

# Adding a second convolutional layer with kernel size 5

model.add(layers.Conv1D(filters=128, kernel\_size=5, activation='relu'))

# Global max pooling layer to reduce the dimensionality

model.add(layers.GlobalMaxPooling1D())

# Fully connected layers

model.add(layers.Dense(64, activation='relu'))

model.add(layers.Dense(32, activation='relu'))

# Output layer with a single neuron and sigmoid activation for binary classification

model.add(layers.Dense(1, activation='sigmoid'))

# Compile the model

model.compile(optimizer='adam',

loss='binary\_crossentropy',

metrics=['accuracy', tf.keras.metrics.AUC(name='auc')])

return model

```

**הסבר:**

פונקציה זו יוצרת מודל רשת עצבית קונבולוציונית (CNN) עם שכבת קלט בגודל `input\_shape`. המודל כולל:

1. שתי שכבות קונבולוציה עם גודל גרעין (kernel) של 5, כדי לתפוס תבניות באורך של חמישה בסיסים (five-mer).

2. שכבת Max Pooling גלובלית כדי להקטין את הממדיות של המידע.

3. שתי שכבות Fully Connected עם 64 ו-32 נוירונים בהתאמה.

4. שכבת פלט עם נוירון אחד ושימוש בפונקציית סיגמואיד לסיווג בינארי (חיובי או שלילי).

5. הקומפילציה של המודל מתבצעת עם `adam` כמייעל (optimizer), `binary\_crossentropy` כאובדן, ומדידת דיוק (accuracy) ו-AUC.

* פונקציה להמרת רצפי DNA ל-One Hot Encoding

```python

def one\_hot\_encode(seq, maxlen=41):

mapping = {'A': [1, 0, 0, 0], 'C': [0, 1, 0, 0], 'G': [0, 0, 1, 0], 'T': [0, 0, 0, 1], 'N': [0.25, 0.25, 0.25, 0.25]}

encoded = np.zeros((maxlen, len(CHARS)))

for i, char in enumerate(seq):

if i < maxlen:

encoded[i] = mapping.get(char, [0, 0, 0, 0])

# Padding אם הרצף קצר מהאורך המקסימלי

if len(seq) < maxlen:

padding\_length = maxlen - len(seq)

encoded[len(seq):] = [0.25, 0.25, 0.25, 0.25]

return encoded

```

**הסבר:**

פונקציה זו ממירה רצפי DNA לייצוג `One Hot Encoding`. כל בסיס (A, C, G, T) מומר למערך של ארבעה מספרים (1 או 0) בהתאם למיקומו. במקרה שיש תו 'N', הוא מומר ל-0.25 בכל אחד מהמיקומים. אם הרצף קצר מ-41 תווים, נוסף לו Padding עם ערכים של 0.25 בכל מקום.

* פונקציה לערבוב רצף

```python

def suffle(sequence):

sequence\_list = list(sequence)

random.shuffle(sequence\_list)

return ''.join(sequence\_list)

```

**הסבר:**

פונקציה זו מבצעת ערבוב רנדומלי של תווי הרצף ומחזירה את הרצף המעורבל.

* פונקציה לקביעת תוויות (Labels) על פי הסייקלים

```python

def assign\_labels\_based\_on\_cycle(cycle\_files):

sequence\_labels = {}

highest\_cycle = len(cycle\_files)

for cycle\_index, cycle\_file in enumerate(cycle\_files):

with open(cycle\_file, 'r') as file:

for line in file:

sequence, count = line.strip().split(',')

if cycle\_index + 1 == highest\_cycle:

sequence\_labels[sequence] = 1

elif cycle\_index == 0 and sequence not in sequence\_labels:

sequence\_labels[sequence] = 0

if len(cycle\_files) == 1:

for sequence in list(sequence\_labels.keys()):

if sequence\_labels[sequence] == 0:

shuffled\_sequence = suffle(sequence)

if shuffled\_sequence not in sequence\_labels:

sequence\_labels[shuffled\_sequence] = 0

return sequence\_labels

```

**הסבר:**

פונקציה זו קובעת תוויות (labels) לרצפים על פי הסייקל שלהם:

- רצפים מהסייקל הגבוה ביותר מקבלים תווית 1.

- רצפים מהסייקל הנמוך ביותר (שלא נמצאו בסייקל גבוה יותר) מקבלים תווית 0.

- אם יש רק סייקל אחד, מתווספים גם רצפים מעורבלים עם תווית 0.

* הכנת נתונים לאימון המודל

```python

def prepare\_data\_for\_training(rbp\_name, cycle\_files):

positive\_cycle = cycle\_files[-1]

negative\_cycle = cycle\_files[0]

positive\_sequences = []

negative\_sequences = []

with open(positive\_cycle, 'r') as pos\_file:

for line in pos\_file:

sequence = line.strip().split(',')[0]

positive\_sequences.append(sequence)

if len(positive\_sequences) == 50000:

break

with open(negative\_cycle, 'r') as neg\_file:

for line in neg\_file:

sequence = line.strip().split(',')[0]

if sequence not in positive\_sequences:

negative\_sequences.append(sequence)

if len(negative\_sequences) == 50000:

break

if len(cycle\_files) == 1:

negative\_sequences = [suffle(seq) for seq in positive\_sequences]

positive\_labels = [1] \* len(positive\_sequences)

negative\_labels = [0] \* len(negative\_sequences)

X\_train = positive\_sequences + negative\_sequences

y\_train = positive\_labels + negative\_labels

X\_train = np.array([one\_hot\_encode(seq) for seq in X\_train])

return X\_train, y\_train

```

**הסבר:**

פונקציה זו מכינה את הנתונים לאימון המודל:

- היא לוקחת את 50,000 הרצפים הראשונים מהסייקל הגבוה ביותר כתווית 1 (חיוביים).

- היא לוקחת את 50,000 הרצפים הראשונים מהסייקל הנמוך ביותר כתווית 0 (שליליים), אם הם לא נמצאים ברצפים החיוביים.

- אם יש רק סייקל אחד, הרצפים השליליים הם הרצפים החיוביים המעורבלים.

- הרצפים ממירים ל-`One Hot Encoding`.

* אימון המודל וחיזוי על נתוני RNAcompete

```python

def train\_and\_predict(rbp\_name, cycle\_files, rncmpt\_path):

X\_train, y\_train = prepare\_data\_for\_training(rbp\_name, cycle\_files)

shuffled\_indices = np.arange(len(X\_train))

np.random.shuffle(shuffled\_indices)

X\_train = X\_train[shuffled\_indices]

y\_train = np.array(y\_train)[shuffled\_indices]

model = build\_model(input\_shape=(41, 4))

model.fit(X\_train, y\_train, epochs=30, batch\_size=64, validation\_split=0.3, callbacks=[

tf.keras.callbacks.EarlyStopping(monitor='val\_loss', patience=3, restore\_best\_weights=True)

])

# Now predict on the RNAcompete data

with open(rncmpt\_path, 'r') as file:

rncmpt\_sequences = [line.strip() for line in file]

X\_test = np.array([one\_hot\_encode(seq) for seq in rncmpt\_sequences])

predictions = model.predict(X\_test)

np.savetxt(f"{rbp\_name}\_predictions.txt", predictions)

return predictions

```

**הסבר:**

פונקציה זו מבצעת את האימון של המודל על נתוני ה-SELEX ומבצעת חיזוי על נתוני ה-RNAcompete:

- הנתונים להכשרה מעורבבים.

- המודל מאומן על הנתונים עם עצירה מוקדמת במקרה שבו האובדן על קבוצת האימות לא משתפר.

- לאחר האימון, המודל מבצע חיזוי על נתוני ה-RNAcompete.

- התוצאות נשמרות לקובץ טקסט.

* קריאת מידע על קבצי RBP

```python

def load\_rbp\_cycle\_info(file\_path):

rbp\_files = {}

with open(file\_path, 'r') as file:

lines = file.readlines()

current\_rbp = None

for line in lines:

line = line.strip()

if line.startswith("RBP"):

current\_rbp = line.split(':')[0]

rbp\_files[current\_rbp] = []

elif line.startswith("htr-selex"):

rbp\_files[current\_rbp].append(line)

return rbp\_files

```

**הסבר:**

פונקציה זו טוענת את המידע על קבצי RBP מתוך קובץ טקסט ומכניסה אותו למילון. כל מפתח במילון הוא שם של RBP, והערך הוא רשימה של קבצי SELEX המתאימים לכל RBP.

CNN עם סקלאר

### הסבר כללי על השלבים:

1. \*\*שלב קידוד הרצפים (One Hot Encoding):\*\*

הקוד מתחיל בפונקציה `one\_hot\_encode`, שממירה רצפי DNA לייצוג מתאים עבור רשת נוירונים. כל אות ברצף ממופה לווקטור באורך 4, כאשר לכל בסיס (A, C, G, T) יש ייצוג בינארי ייחודי. תו 'N' שמייצג חוסר ודאות מקודד עם ערכים של 0.25 בכל מקום.

2. \*\*שלב ערבוב רצפים (Shuffle):\*\*

פונקציית `suffle` מערבבת את תווי הרצף באקראי. זה יכול להיות שימושי ליצירת נתונים שליליים או נתונים נוספים עבור אימון המודל.

3. \*\*חישוב ציונים לכל רצף (Scoring):\*\*

ישנה פונקציה בשם `get\_seqs\_with\_scores` שמחשבת ציונים לכל רצף DNA על בסיס נתוני SELEX והמשקלים שניתנים לכל מחזור (cycle). הפונקציה עושה שימוש בקובצי SELEX, מחשבת את כמות הרצפים בתוך כל קובץ לפי משקלות, ומחזירה מילון עם ציונים משוקללים לכל רצף.

4. \*\*חישוב סקלארים מראש (Precomputing Scalars):\*\*

בשלב זה, באמצעות הפונקציה `precompute\_scalars`, מחושבים מראש הסקלארים (מספר חמישיות תואמות לכל רצף) עבור כל הרצפים האפשריים באורך 5. זה מאפשר לייעל את תהליך האימון בכך שאין צורך לחשב את הסקלארים בכל פעם מחדש.

5. \*\*הכנת נתוני אימון עם סקלארים מחושבים מראש:\*\*

הפונקציה `prepare\_data\_for\_training\_with\_precomputed` מכינה את נתוני האימון עבור המודל תוך שימוש בסקלארים המחושבים מראש. היא יוצרת את מערכי הקלט והפלט עבור הרצפים החיוביים והשליליים, וכן מחשבת את הערכים הסקלאריים בהתאמה לכל רצף.

6. \*\*בניית מודל ה-CNN:\*\*

הפונקציה `build\_model` בונה את מודל רשת הנוירונים הקונבולוציונית (CNN). המודל מכיל שכבות קונבולוציה שמטרתן לזהות תבניות של חמישיות בתוך רצפי ה-DNA, ושכבות fully connected שמבצעות את הסיווג. בנוסף, המודל מקבל גם את הערכים הסקלאריים ומאחד אותם עם הפלט של השכבות הקונבולוציוניות.

7. \*\*אימון המודל וחיזוי:\*\*

הפונקציה `train\_and\_predict\_with\_precomputed` מאמנת את המודל על נתוני SELEX ומשתמשת בערכים הסקלאריים המחושבים מראש כדי לבצע חיזוי על נתוני RNAcompete.

### הסבר על הפונקציות השונות מה-CNN הנאיבי:

1. \*\*פונקציית `get\_seqs\_with\_scores`:\*\*

הפונקציה הזו מחושבת ציונים לכל רצף DNA על סמך קובצי SELEX. הפונקציה הזו מייחסת משקלים לכל מחזור (cycle) של SELEX, ומשקללת את כמות הרצפים בכל מחזור כדי לקבל ציון משוקלל לכל רצף.

2. \*\*פונקציית `precompute\_scalars`:\*\*

פונקציה זו מחשבת מראש את הסקלארים עבור כל הרצפים האפשריים באורך 5 ומאחסנת אותם במילון. זה מאפשר שימוש חוזר בסקלארים בשלב האימון והחיזוי ללא צורך לחשב אותם שוב.

3. \*\*פונקציית `prepare\_data\_for\_training\_with\_precomputed`:\*\*

פונקציה זו מכינה את נתוני האימון תוך שימוש בסקלארים המחושבים מראש. היא יוצרת את מערכי הקלט (רצפי DNA מקודדים) ואת הפלט (תוויות) עבור הרצפים החיוביים והשליליים, וכן מחשבת את הסקלארים בהתאמה לכל רצף.

4. \*\*פונקציית `build\_model`:\*\*

פונקציה זו בונה את המודל הכולל הן את קלטי הרצפים המקודדים והן את הערכים הסקלאריים. המודל מאחד את הפלט של השכבות הקונבולוציוניות עם הסקלארים ומבצע את הסיווג הסופי.

5. \*\*פונקציית `train\_and\_predict\_with\_precomputed`:\*\*

פונקציה זו מאמנת את המודל על נתוני SELEX, תוך שימוש בסקלארים המחושבים מראש. לאחר האימון, הפונקציה מבצעת חיזוי על נתוני RNAcompete ומשווה את התוצאות.