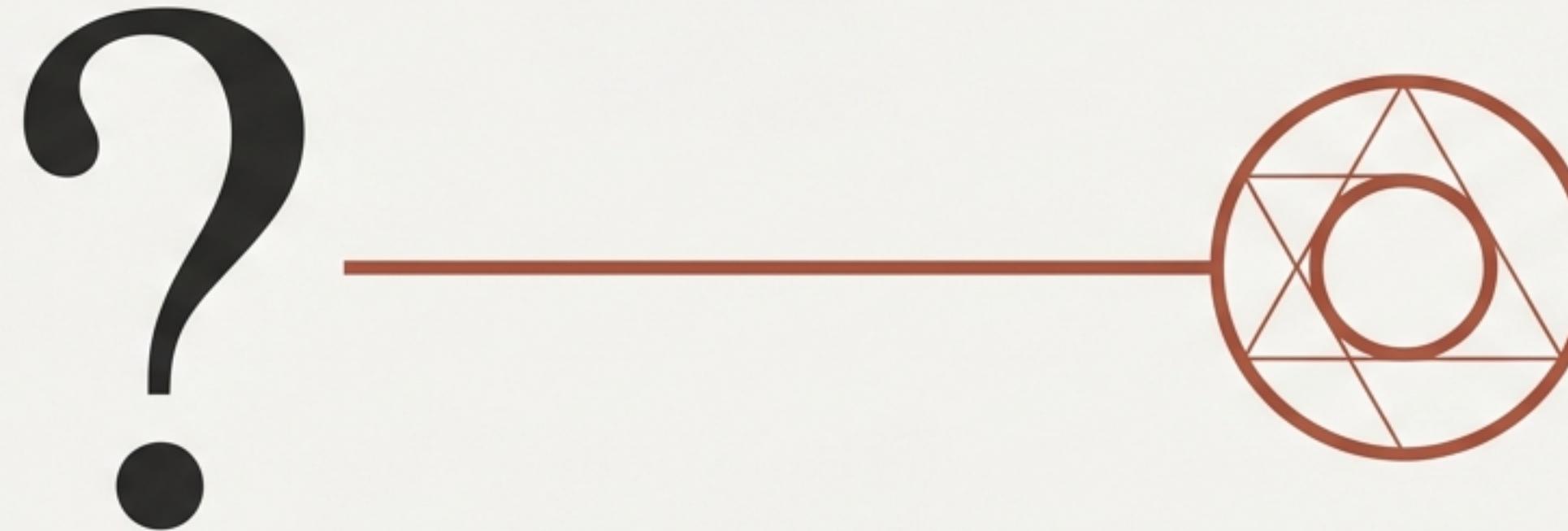


האם מכונה יכולהגלות חוק טבע?

חקירת תהליכי הלמידה של רשת נירוניים, צעד אחר צעד.



במקרה לתוכנת נוסחה באופן מפורש, אנו ניתן למודל מספר דוגמאות בלבד. האם הוא יכול להסיק את הקשר הבסיסי ביניהן?

תיק החקירה: איסוף הראיות

פרנהייט (פלט רצוי)	צלזיוں (קלט)
-40°F	-40°C
-10°C	14°F
0°C	32°F
8°C	46.4°F
15°C	59°F
22°C	71.6°F
38°C	100°F

המשךה שלנו היא למד מודל להמיר
טמפרטורה מצלזיוں לפרנהייט.
הנוסחה המוכרת היא: $f = c * 1.8 + 32 + c = f$.
אר המודל שלנו אינו מכיר אותה. הוא
יקבל רק את 'הראיות' הבאות: 7
זוגות של ערכים ידועים.

```
celsius_q      = np.array([-40, -10,  0,   8, 15, 22, 38], dtype=float)
fahrenheit_a = np.array([-40,  14, 32, 46.4, 59, 71.6, 100], dtype=float)
```

'חוקר' אחד בלבד (נוירון אחד) שמנסה לפתור את הבעיה. זהו גם גודל הפלט של המודל – ערך פרנהייט בודד.

בנין הבלש: מודל עם נוירון בודד

```
# We define a single layer named '研究员'  
layer = tf.keras.layers.Dense(units=1, input_shape=[1])  
  
# We assemble it into a sequential model  
model = tf.keras.Sequential([layer])
```

הקלט ל'חוקר' שלנו הוא תמיד ערך בודד – טמפרטורה בצלזים.

ניצור את המודל הפשטוט ביותר האפשרי: רשת עם שכבה אחת (Dense Layer) ונוירון אחד בלבד. הנוירון זהה הוא ה'חוקר' שלנו, ש�피קידו למצוא את הקשר בין הקלט לפלט.

קביעת אסטרטגיית החקירה

לפני תחילת האימון, علينا להגדיר למודל כיצד למדוד. אנו עושים זאת באמצעות שני כלים מרכזיים:

אופטימיזר (Optimizer)

המנגן שמתוקן את המשתנים הפנימיים של המודל כדי **להקטין את הפסד** בכל צעד.

במילים אחרות: "איך נטעה פחות בפעם הבאה?".

פונקציית הפסד (Loss Function)

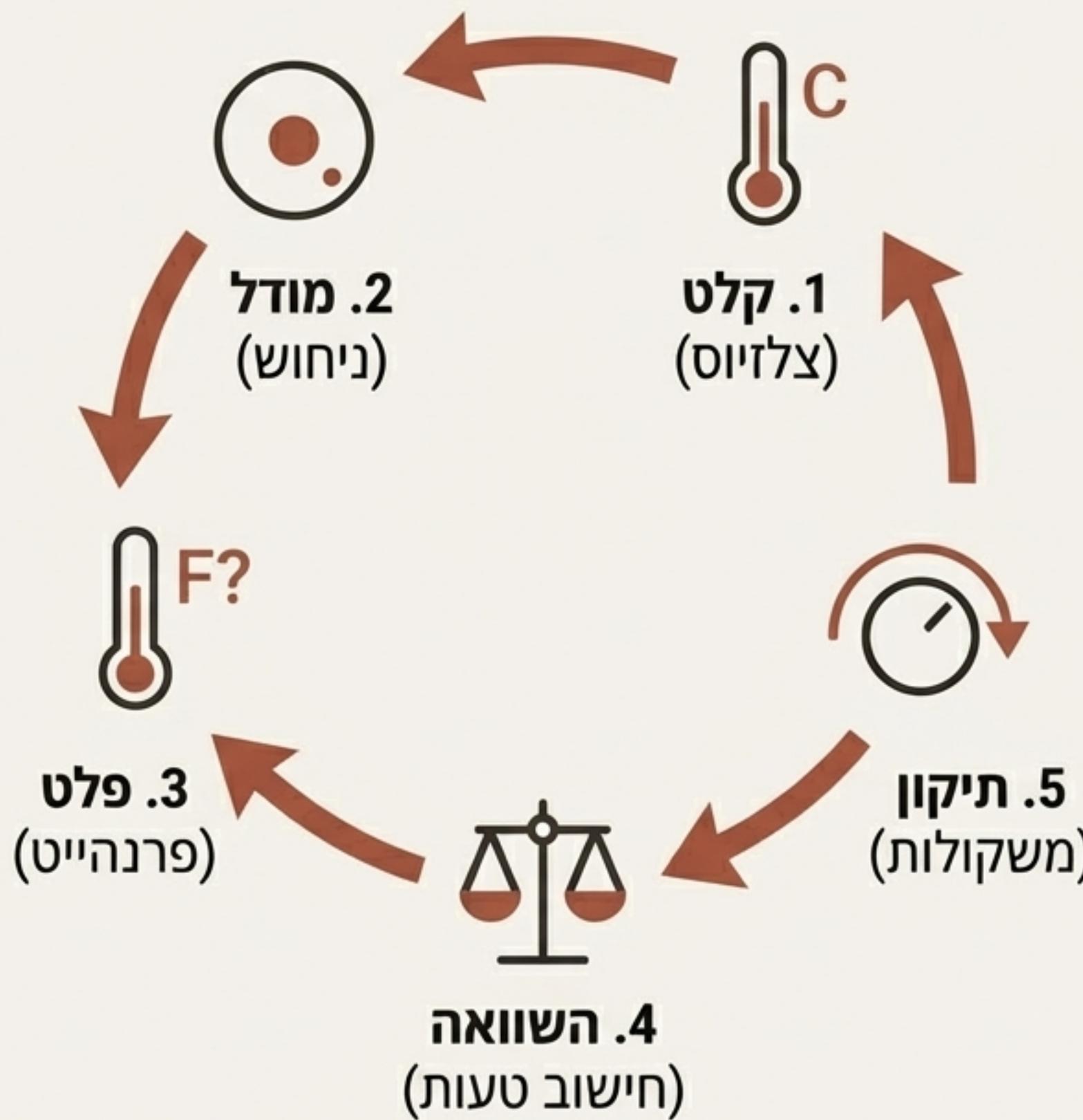
המדד שקבע **כמה רחוק** החיזוי של המודל מהתשובה הנכונה.

במילים אחרות: "עד כמה אנחנו טועים?".

```
model.compile(loss='mean_squared_error',  
               optimizer=tf.keras.optimizers.Adam(0.1))
```

האופטימיזר Adam הוא בחירה סטנדרטית ויעילה. קצב הלמידה הלמידה (0.1) קובע את גודל הצעד בכל תיקון שהמודל מבצע.

החקירה: 500 סבבי אימון ארציזן על הראיות

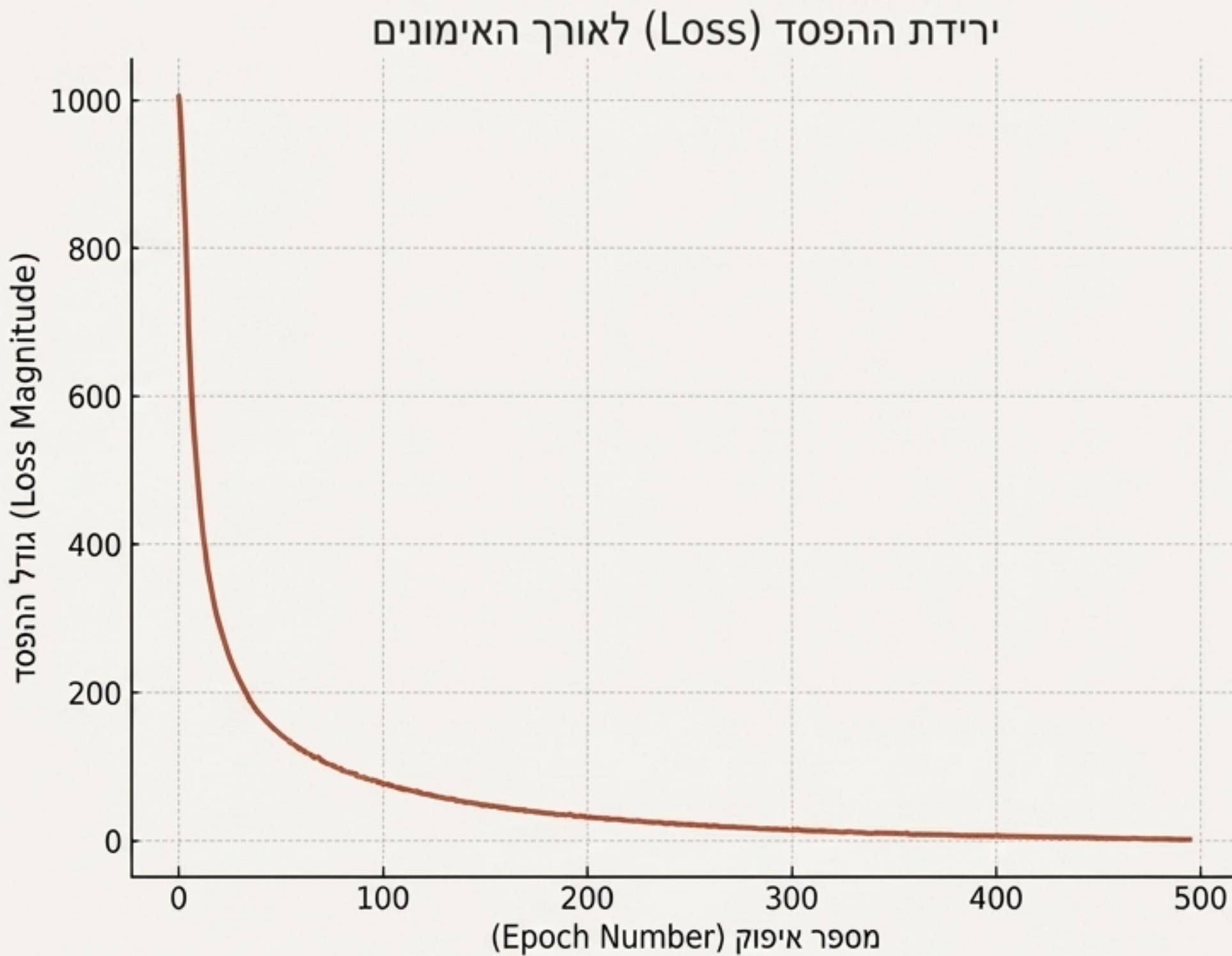


תהליך האימון הוא למעשה פעולה לולאה:
המודל מוחש, בודק את טעותו (loss), ומתקן
ומתקן את עצמו (optimizer).

אנו מורים לו לחזור על התהליך זהה 500 פעמים
על כל סט הראיות שהציגנו.

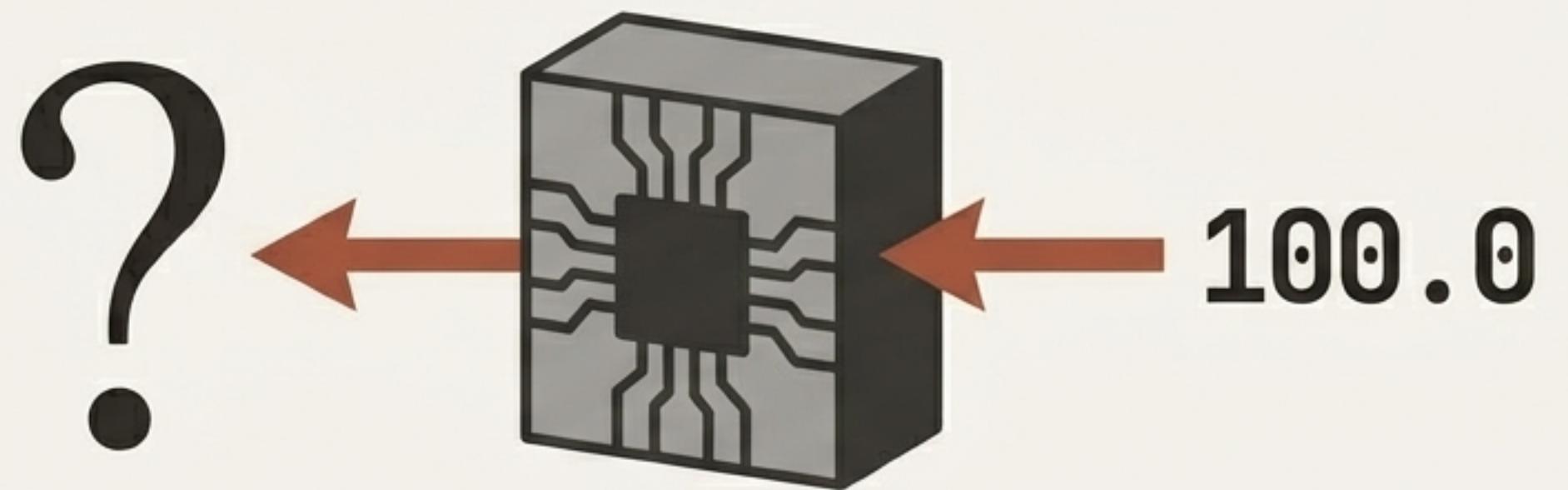
```
history = model.fit(celsius_q,  
                     fahrenheit_a, epochs=500,  
                     verbose=False)  
print("Finished training the model")
```

עקוב אחר התקדמות: עקומת הלמידה



אובייקט ה-'history' שקיבלנו ממتدמת ה-'fit' מאפשר לנו לראות כיצד 'הפסד' (גודל הטעות) של המודל קטן רעmodoל קטן ככל שהאימון מתתקדם. בתחילת השיפור דרמטי, ולאחר מכן הוא מתמתן כשהמודל מבצע כוונוניים עדינים.

מבחן האמת: חיזוי על נתון לא מוכר



המודל אומן על 7 דוגמאות בלבד. בעת,
נבחן אותו על ערך חדש לחלוטין שלא
שלא היה חלק מתוכני האימון: 100
מעלות צליזום.
התשובה הנכונה, לפי הנוסחה, היא 212.
מה לדעתכם המודל יחזיר?

```
print(model.predict([100.0]))
```

פסק הדין: המודל מגיע לתשובה הנכונה

המודל, שלמד אך ורק מהדוגמאות, הצליח לחשב את התוצאה הנכונה עברו 100° בדיק מרשים.

[211.9...]

בדיקה כמעט מושלמת. המודל למד את חוק ההמרה, ולא רק שינן את הדוגמאות.

הגילוי: הוצאה אל תור 'מוחו' של המודל

```
[array([[1.8...], dtype=float32),  
 array([31.9...], dtype=float32)]
```

1.8... **31.9...**

m **b**

ההצלחה בחיזוי מרשימה, אך
הגילוי האמתי מגיע כשבוחנים את
המשתנים הפנימיים (ה'משקלות')
שהמודל למד.

אלו הם הערכים שהנוiron הבודד
משתמש בהם כדי לבצע את החישוב.

```
print(θ.get_weights())
```

הפענוח: המכונה גילתה את הנוסחה

המודל הגיע באופן עצמאי לשני הרכיבים המרכיבים את נוסחת ההמרה. המשוואה של ניירון בודד היא $b + ax = y$. בתהליכי האימון, המודל התאים את a ו- b כך שיתקרבו בצורה אופטימלית לערכיהם בנוסחה האמיתית.

מה שהמודל למד

$$f = 1.8... * c + 31.9...$$

הנוסחה האמיתית

$$f = 1.8 * c + 32$$

למידת מכונה, במקרה זה, היא תהליכי אופטימיזציה נומרית למציאת הפרמטרים של פונקציה מתמטית.

ומה אם החקירה היתה מסובכת יותר?

```
l0 = tf.keras.layers.Dense(units=4, input_shape=[1])
l1 = tf.keras.layers.Dense(units=4)
l2 = tf.keras.layers.Dense(units=1)

model = tf.keras.Sequential([l0, l1, l2])
# ... compile and fit ...

print(model.predict([100.0]))
```

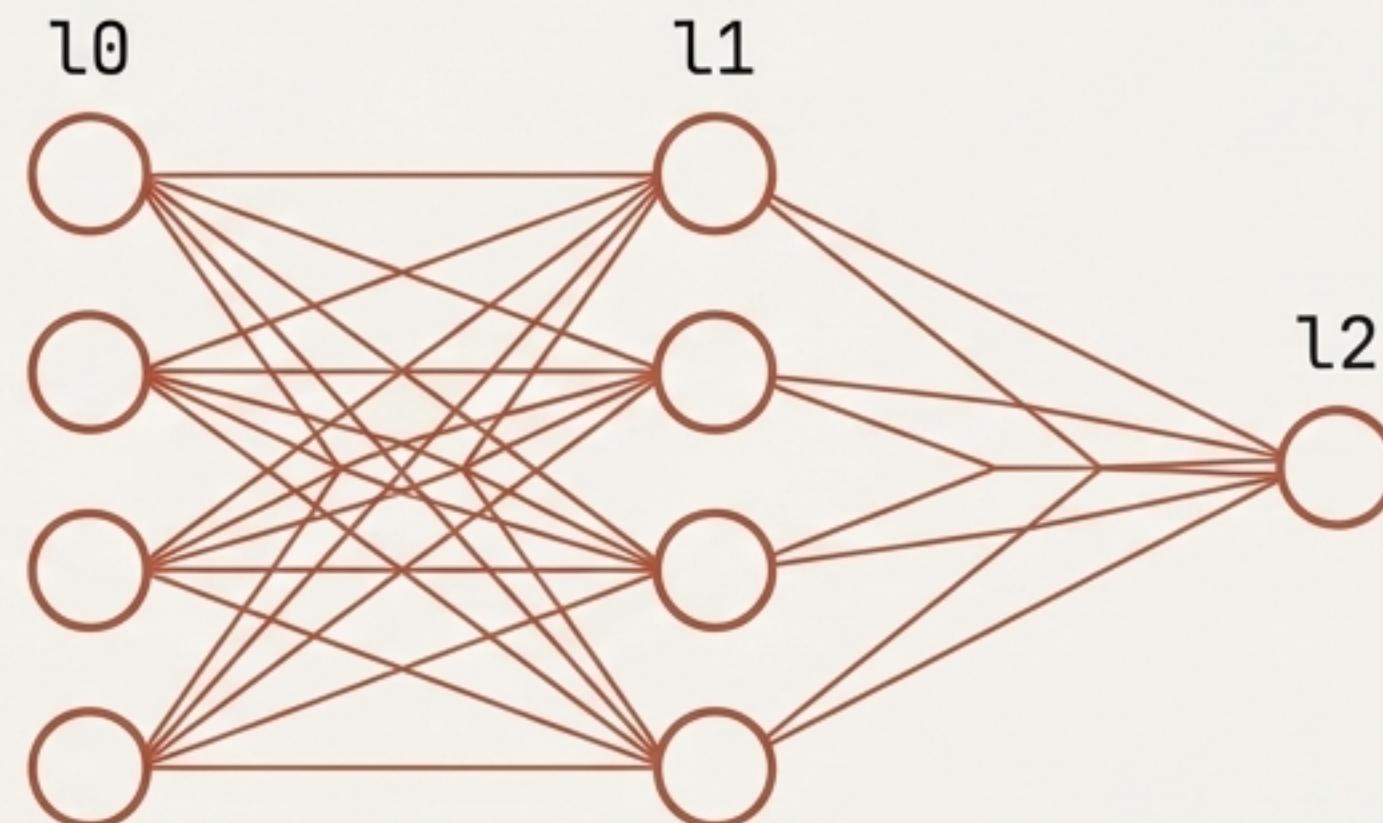
['[[211.9...]]']

המודל עדין מגיע לתשובה המדויקת.

העלומה והפתרון החבוי

כשאנו בוחנים את המשקלות של המודל המורכב, הערכים נראהים אקרים לחלוטין. הם אינם דומים יותר 1.8-32. המודל עדיין פתר את הבעיה, אך הפתרון כתוב 'מפוזר' על פני רשת מורכבת של משתנים. זהו לב ההבדל בין מודל פשוט וניתן לפירוש, לבין הכוח (ומורכבות) של רשתות עמוקות.

```
10 variables:  
[array([[ 0.4985..., -0.7856...],  
       [ 1.2094...,  0.3421...],  
       [-0.8763...,  0.6502...],  
       [ 0.1234..., -0.5678...]]...]
```



המודל הפשוט אפשר לנו לראות את ה'בלש' מגלה את הנוסחה. מודלים מורכבים פותרים בעיות מסובכות יותר, אך לעיתים קרובות, דרך הפעולה שלהם נשארת עלומה.