

נאום נסנאָט אַכּוֹרְה-כְּרוּיקָס

2115 44002 - נס נס

fully connected feed forward neural network דה, ייְלָהֶה גַּדְעֹן הַנּוֹסֵךְ מִכְּרָבָן
train_data_MNIST 8019 convk3 8x 16x 8019MNIST taken גַּדְעֹן הַנּוֹסֵךְ מִכְּרָבָן

הנחיות הדרושים לשלב ה- $0-9$ נקבעו ב- 28×28 איקס-גראם, איככ' הנטווגים ראים מ- 0 (ז' 255) ועד 9 (טהור).
הנחיות הדרושים לשלב ה- $0-9$ נקבעו ב- 28×28 איקס-גראם, איככ' הנטווגים ראים מ- 0 (ז' 255) ועד 9 (טהור).

Neural Network SPQR ein

כאלקטרוניקה

השענשטיין פורמולה softmax היא כפונקציית הסטטיסטיקה המבוססת על פונקציית ה- ReLU .

```
def _relu(self, x):
    return np.maximum(0, x)

def _relu_derivative(self, x):
    return (x > 0).astype(float)

def _softmax(self, x):
    exp_x = np.exp(x - np.max(x, axis=1, keepdims=True))
    return exp_x / np.sum(exp_x, axis=1, keepdims=True)
```

ReLU - פונקציית ReLU

$$\text{ReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$

המינימום הבודק בפונקציית הסיגמוד היא נקודת קיצון של פונקציית הסיגמוד. פונקציית הסיגמוד מוגדרת כפונקציית מילוט מ-0 ל-1. פונקציית הסיגמוד מוגדרת כפונקציית מילוט מ-0 ל-1. פונקציית הסיגמוד מוגדרת כפונקציית מילוט מ-0 ל-1. פונקציית הסיגמוד מוגדרת כפונקציית מילוט מ-0 ל-1.

backpropagation の概念を理解するには ReLU と Softmax の理解が必須です。

$$\text{ReLU}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$

רמלהן, גכו פיט פוקטוויז אוניבראלי פוליאפזס פג'ה דו כוון
ירנדז קסלי ReLU-ים ואחרות. פט לטל נכון שנהגד הזרם הזרקורי כ-
ReLU, Leaky ReLU
ReLU זרker. גזיע אל נומסרא חומר קרג'ן ויגתער כ-
פוקטוויז

softmax גיבוב - פורק softmax נספחים:

$$\Theta(s) = \frac{e^{z_i}}{\sum_{j=1}^k e^{z_j}}$$

המכרז. בחלק הראשון נפרטנו בקורס ת'יה 10 כפננו ה c_1 נאמר והיא מוגדרת כטולן ת'יז'י'ג'י'.
הנחיות multi class classification דנו נאקרה הכלוקט θ_0 הטעיה $\theta_1 \dots \theta_m$ גנויות
בפרקן סכום $\sum_{i=0}^m \theta_i x_i + \theta_0$ (קדום ווילטן)

בוקטט softmax מופתע על ידי הsoftmax הדרישה ופוקטור הסכום נורם את הבוקטט.

הנחתה מתקנית

```
def __init__(self, layer_sizes, learning_rate=0.01):
    self.layer_sizes = layer_sizes
    self.initial_lr = learning_rate
    self.learning_rate = learning_rate
    self.epochs = epochs
    self.batch_size = batch_size
    self.decay = decay
    self.patience = patience
    self.weights = []
    self.biases = []
    self._init_weights()
```

$$w \sim N(0, \sqrt{\frac{2}{n}})$$

```
def __init_weights(self):
    for i in range(len(self.layer_sizes) - 1):
        W = np.random.randn(self.layer_sizes[i], self.layer_sizes[i + 1]) * np.sqrt(2. / self.layer_sizes[i])
        b = np.zeros((1, self.layer_sizes[i + 1]))
        self.weights.append(W)
        self.biases.append(b)
```

forward propagation

```
def _forward(self, X):
    activations = [X]
    zs = []
    for i in range(len(self.weights) - 1):
        z = activations[-1] @ self.weights[i] + self.biases[i]
        zs.append(z)
        a = self._relu(z)
        activations.append(a)
    z = activations[-1] @ self.weights[-1] + self.biases[-1]
    zs.append(z)
    a = self._softmax(z)
    activations.append(a)
    return activations, zs
```

לונן L נאמר הפלכתי אם $\forall n \in \omega$ ($\exists i \in \omega$) $\psi_i(n) = 1$

אפשר חישוב $\max_{\text{softmax}}(\mathbf{x})$ באמצעות ReLU והעכדר

$$A^{(L)} = \begin{cases} \text{ReLU}(z^{(L)}) & \text{if } L < n \\ \text{softmax}(z^{(L)}) & \text{if } L = n \end{cases}$$

הPOLK ביה שאנטן היה כג עכבי Z, A החלטות לפכזות האתיפות קסואט. קראון צאלו מונטניאן פאלטן הנטזין לאין-ויל-וילינג הפטואה ועפוקת backpropagation.

back propagation

לפניה של א.ז.י. פורק ב-**backward**, קנה נספחים, loss כימות גראונד וודט ו疎מיטן יקיה גראונד צווק הינו מינימום.

```

def _backward(self, X, y, activations, zs):
    m = X.shape[0]
    y_onehot = np.zeros_like(activations[-1])
    y_onehot[np.arange(m), y] = 1
    delta = activations[-1] - y_onehot
    grads_w = [0] * len(self.weights)
    grads_b = [0] * len(self.biases)
    grads_w[-1] = activations[-2].T @ delta / m
    grads_b[-1] = np.sum(delta, axis=0, keepdims=True) / m
    for i in range(len(self.weights) - 2, -1, -1):
        delta = (delta @ self.weights[i + 1].T) * self._relu_derivative(zs[i])
        grads_w[i] = activations[i].T @ delta / m
        grads_b[i] = np.sum(delta, axis=0, keepdims=True) / m
    for i in range(len(self.weights)):
        self.weights[i] -= self.learning_rate * grads_w[i]
        self.biases[i] -= self.learning_rate * grads_b[i]

```

$$\mathcal{L}(A, y) = -\frac{1}{m} \sum_{i=1}^m \sum_{k=1}^{10} y_{ik} \log A_{ik} \quad (1)$$

Sold batch \downarrow 100N classes \swarrow
 m 10

: cross entropyLoss פונקציית

$$y_i = \begin{cases} 1 & \text{if class } i \text{ is the correct class} \\ 0 & \text{otherwise} \end{cases} \quad \text{TRUE}$$

$$\delta^{(n)} = \frac{\partial \mathcal{L}}{\partial z^{(n)}}$$

מיצרוב והלכיה נפלכתה ונשאלה

$$\delta^{(n)} = A^{(n)} - y$$

א. קורס כף שכך (מבחן):

$$\nabla_w w^{(L)} = \frac{1}{m} (A^{(L-1)})^T \delta^{(L)}$$

פְּנִים יָמַר לְבָדֵק

$$\nabla b^{(L)} = \frac{1}{m} \sum \delta^{(L)}$$

$$\delta^{(L)} = \text{ReLU}'(z^{(L)}) (W^{(L+1)} \delta^{(L+1)^\top})$$

ונרנצ'א זכrai הנטק'ין והפיאו נהנ'ה נט'ה ג'נ'ה.

טינר הנואג

לעתה שפוי פועל בינה. fit יתבצע על ה

batch

 ה

batches

 ו-

label

 - ה

forward pass

. ה

backward pass

 - ה

backward pass

.

```
def fit(self, X_train, y_train, X_val=None, y_val=None):
    best_val_acc = 0
    epochs_no_improve = 0
    for epoch in range(self.epochs):
        indices = np.arange(X_train.shape[0])
        np.random.shuffle(indices)
        X_shuffled = X_train[indices]
        y_shuffled = y_train[indices]
        for start in range(0, X_train.shape[0], self.batch_size):
            end = start + self.batch_size
            X_batch = X_shuffled[start:end]
            y_batch = y_shuffled[start:end]
            activations, zs = self._forward(X_batch)
            self._backward(X_batch, y_batch, activations, zs)
            self.learning_rate *= self.decay
        if X_val is not None and y_val is not None:
            val_acc = self.score(X_val, y_val)
            if val_acc > best_val_acc:
                best_val_acc = val_acc
                epochs_no_improve = 0
            else:
                epochs_no_improve += 1
            if val_acc >= 0.99:
                break
            if epochs_no_improve >= self.patience:
                break
        else:
            train_acc = self.score(X_train, y_train)
```

המודל מושג אומדן ב- AUC של 0.91 ו- ROC של 0.97. validation set הינו מושג ב- AUC של 0.93 ו- ROC של 0.99.patience (0.99) generalization (0.91).
נמצא שמשתמש ב-early stopping, overfitting לא מתרחש. על מנת למנוע generalization מושג אומדן ב- AUC של 0.91 ו- ROC של 0.97.

המודל משלב forward pass וbackward pass. forward pass מבצע predict על מנת לחשוף את הערך המבוקש. backward pass מבצע score על מנת לחשוף את הערך המבוקש.

הכoker ביה score נטלת פון ההייילט שטולטן גאנטן ההייילט הנטנער ונטנער אונטן הנטנער

```
def predict(self, x):
    activations, _ = self._forward(x)
    return np.argmax(activations[-1], axis=1)

def score(self, x, y):
    predictions = self.predict(x)
    return np.mean(predictions == y)
```

רכינס גראניט

```
X_train = X_train / 255.0  
X_test = X_test / 255.0
```

ההתקשרות הינה מושג ונשכח אך הפוך.

בכונס כינס נילזם.

הנשען על הנקודות המופיעות בתרשים. נזכיר את הנקודות שנותרנו:

נוסף שכבוי

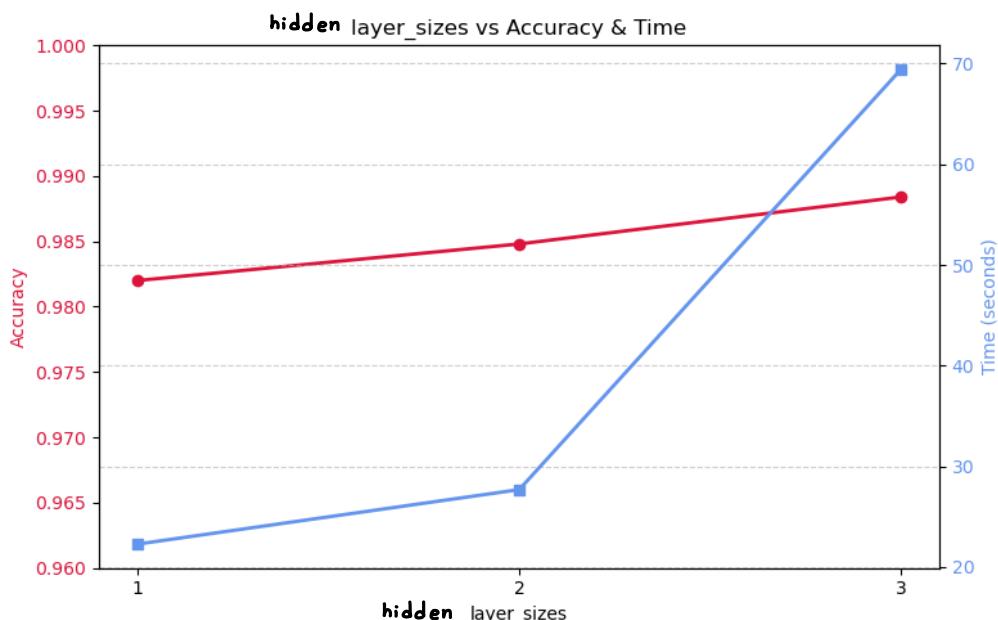
ב-1979, ב-101 נסכל 279 מ-38 שכנות רוסלאן: סכום כולל אוניות, שכנות רוסלאן

השכלה רגולרייזציית געווילר:

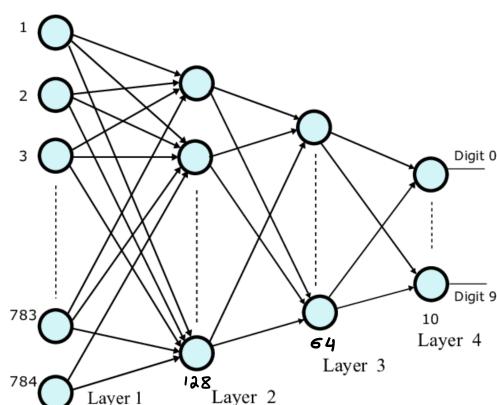
- 1 שכבת פסיהית אחורית $[784, 64, 10]$
- 2 שכבת רגולרייזציה $[784, 128, 64, 10]$
- 3 שכבת רגולרייזציה $[784, 256, 128, 64, 10]$

שאך הפלט נקבע קומזיט כז. געווילר אונ מאכער חטכאל
(64 batch געווילר 1 , מאכער אינטראזיוין 51 , פואט 0.1)

הערכה: געאר קרייטה נערניקה נמאכער האיגלאג אונ אנטז'ו, מההנסעריהם האונכאניגים
ג הערט ליריכזם גסויים נמאכער TDSIM הום חטקוין סט אוניאירט כטוו 128, 64, 256 וטוו
הם אונטז'ויס גערכוינה בז אונמאכער MNIST ויז'אנט חישובים גען היעוינט פאנצ'ז'
געאנז'אל הפלט נטענאמיגים האונכאניגים פואוילר מהטב.

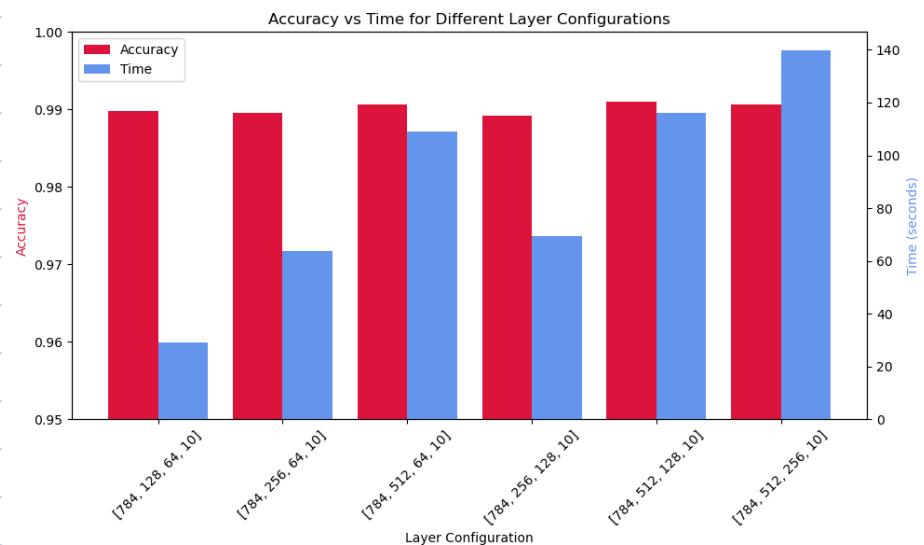


יערט געכווינט נעלג כז. ככג שטואויבק יויע שכבות רגולרייזציה זען אונטז'ר זען
הארפן אונז'ויס יטסיט. גערכוין קהן דען האינז'ון גען מפוגה מהה.
ההיאק כטווין גערכוין רגולרייזציה ~0.985 כטווין יט 3 שכבות רגולרייזציה זען ~0.987.
ההמצע נטגי. זען דען הריבז'ה אונ 3 שכבות רגולרייזציה זען כהכחגה
גען קהחיה געאנז'אל אונז'ויס היז אונ 2 שכבות רגולרייזציה זען זיינט ויז'אנט



ב-התכל שנאכ ששן שכמות רסקכות בהו הזיאק הזוכיאג. קחרע. אל נסכר
הריכועמ אג שכהה. בגדע כה הרבע נסכר וכיתבע אפלעיות הכוגן שע.
שכמות רסקכות תרבע שען אסע הריכועמ אג אתתר נה.

, [784, 256, 64, 10], [784, 128, 64, 10]
. [784, 256, 128, 10]. [784, 512, 64, 10]
[784, 512, 256, 10], [784, 512, 128, 10]



הביטוי הנקה נויר ראנט ניגזת הרכגון ונואם:

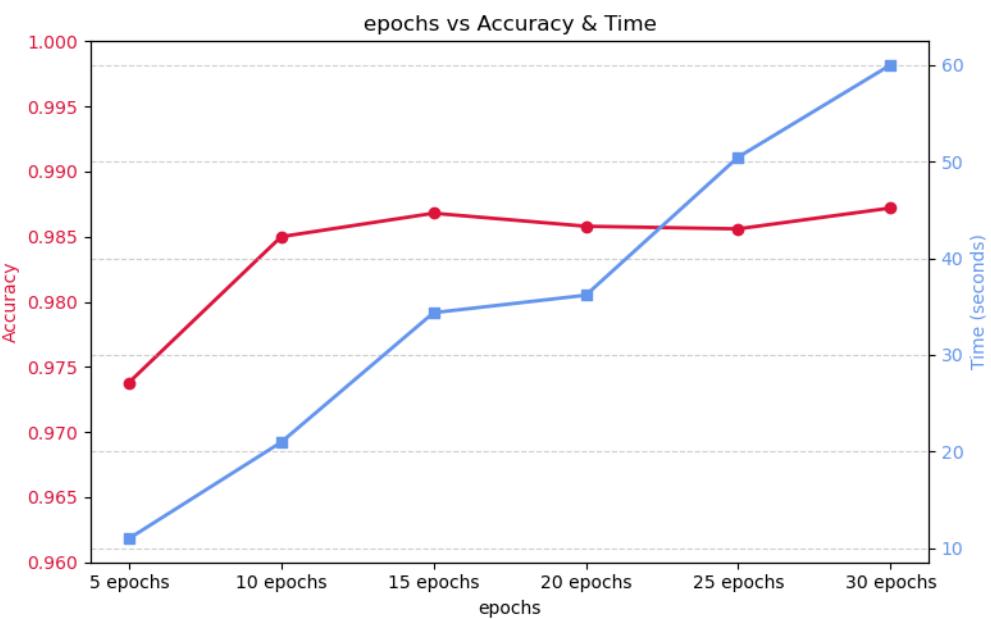
ג'י'ג 29.02	N51	0.9898	ג'י'ג	[784, 128, 64, 10]
ג'י'ג 109.03	N51	0.9906	ג'י'ג	[784, 512, 64, 10]
ג'י'ג 116.01	N51	0.99	ג'י'ג	[784, 512, 128, 10]
ג'י'ג 139.73	N51	0.991	ג'י'ג	[784, 512, 256, 10]

בנוסף לדוגמה בפערת הטרקאה נלכדה [784, 512, 128, 10] אך דען הראינו בסה
ווארה הדרגה פיער. איה שאנצ'ה או ר' הדרגה הדרגה נבדאן נאנצ'קה או ר' הדרגה הדרגה
אקייז'ם. גזרת דעתן הקופיפואזיה [784, 128, 64, 10] מושגנו פיאן אך הפעם

נומך ניכרכן

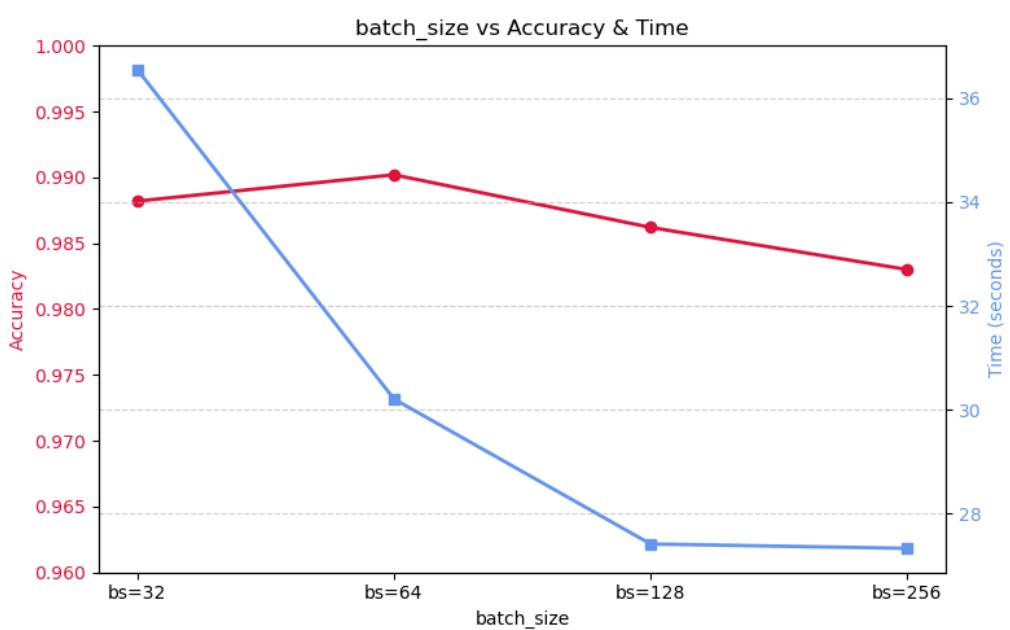
שאנו הפלט אמצעי ריזוקו קבוצות (100 גרעין [84, 128, 64, 0.1]) .

לעתה נסמן בפונט ארכיטקטוני אחד, שנקרא **טיפוס ארכיטקטוני**, ומייצג את המבנה כמבנה אחד. מושג זה מוגדר כמבנה אחד, אשר מושג על ידי סדרת מבנים, אשר מושגים אחד אחר אחד, כאשר כל מבנה מושג כמבנה אחד.

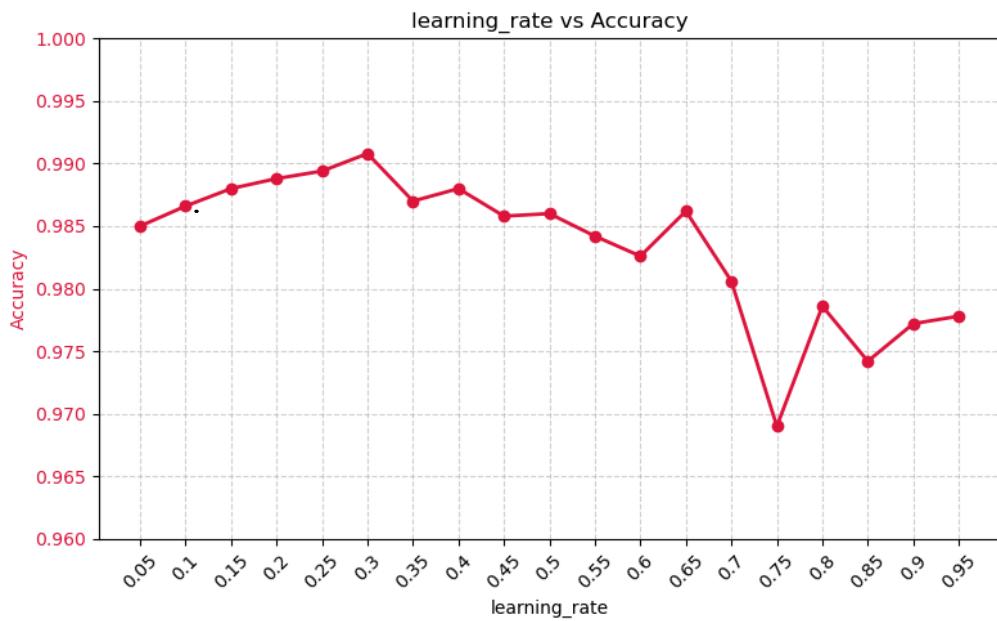


batch Spid

ב-32, 64, 128, 256: batch size נקבע בהתאם למספר הינה. אם מגדיר שאר ה-FCayer'ים (למעט ה-FCayer' הראשון) עם batch size 64, אז אורך ה-FCayer' הראשון יהיה $32 \times 64 = 2048$. אם מגדיר שאר ה-FCayer'ים (למעט ה-FCayer' הראשון) עם batch size 32, אז אורך ה-FCayer' הראשון יהיה $32 \times 32 = 1024$.



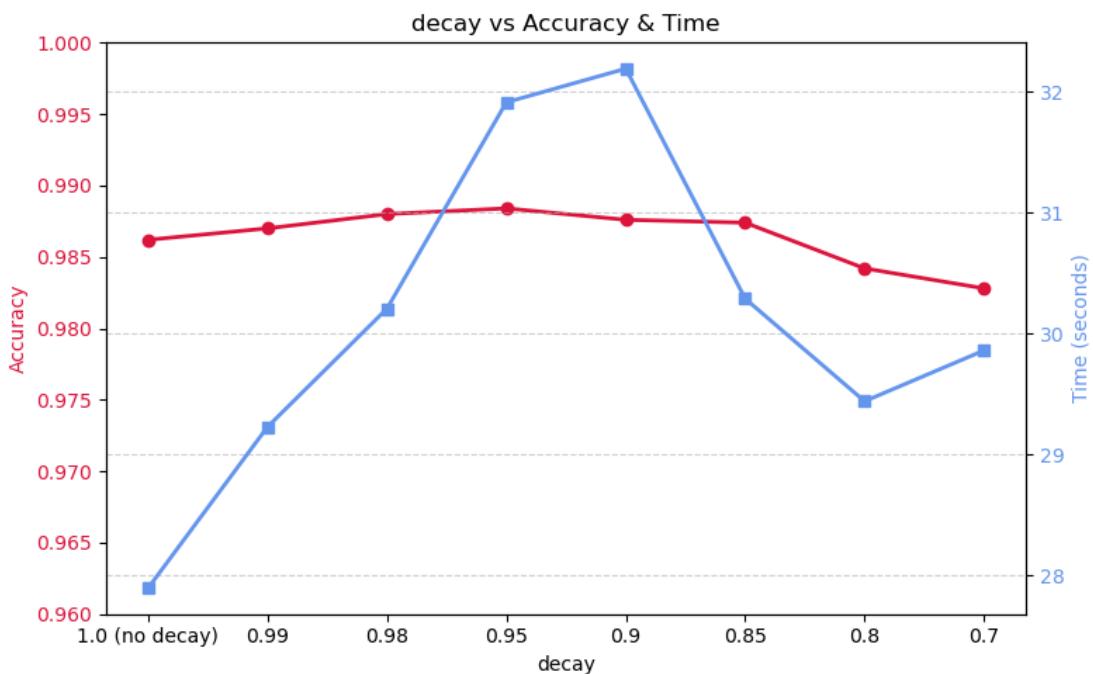
ק 3 ג הוניה קיימת מכנה נסיעת כל נסעה. הרעיון נכון ל- LSTM אך לא ל- RNN . הוניה א' 0.95 ו' 0.05 (בגיאומטריה).



ק 3 ב פהו (0.6 ומעלה) פיתר דיוקנית כביגודים יתקן מ-5500 שנותון גוף
לקיים האינטראקציית גזירה גזורה כוכב כוכב
עיטן גאות שגוזם הולך נזק הולך והפוך הולך
רכיר או כ-0.3 כ-0.3 ג הוניה.

decay

הו א הדרי נ אוניברסיטת הקורס הרכזתי של קב' הוניה זאכט באל decay
נאהך ציון התפקיד



מבחן ג'טן סטטוט שטח וו'ה גזען נתקבַּל 0.98- 0.9 נתקבַּל היזק הטענה כי עיר
אמור 0.95 , 1 - 0.9 נתקבַּל פטן כיה וחמשת פט� גאנן וריבוי של
8cn רוחן נ 0.93 שטח פט� ויטראז'

הנומינטיביסטיות

- **ReLU** ה-**ה-3'יה** מוחזק בInitialization kaiming initialization יסודן רגולריזציה נתקינה.

בכדי לסייע בפתרון בעיה מינימיזציה נשתמש באלגוריתם הנקרא **gradient descent**.
 מטרת האלגוריתם היא למצוא את הנקודה בה שיפוטה מינימלית.

• **��亡** (decay) - מלחמתם של אטומי גמא או פוטון על מנת לשוב לאנרגיה נמוכה יותר.

• patience (ונאכלה מוקדם) - early stopping
 • generalization (האינטגרציה מוקדמת) - overfitting
 • regularization (הנתקה מוקדמת) - stopping rule

- ארכונא הנקה נספּה איזכיה גען חילוקה batches δ, כו' כת היציאה סנויג' נהנרג' בדנאר טפיאן נספּה כו' משלוחאי מוקטן נספּה קומנאותו נספּה נספּה. לה שפה הנרג' איזה, איזה ותקני generalization יי'זע

- פונקציית validation - נאכזב הטעינה היותר מושג נתקדם אך יתרכז ויתר על צורה הטעינה ותהום גודלן ותפקידן אינטראקטיבי נאכזב validation נאכזב נתקדם אך יתרכז ויתר על צורה הטעינה ותהום גודלן ותפקידן אינטראקטיבי