X_train = Matrix (batch_size, 28*28)
y_train = Matrix (1 x batch_size

h hidden_Layer_size (100)

W_1 Matrix (28*28, h)
b_1 Matrix (1 , h)

W_2 Matrix (h , k) } k = 10 [Anzahl Klassen]
b_2 Matrix (1 , k)

step_size float
reg float

Learning

for 1 to epochs (100):

forward pass:

- d = mul(X_train, W_1) $d_{ik} = \sum_{j=1}^{m} x\_train_{ij} \cdot W_{1jk}$ Matrix multiplikation
- $hl = d + b_1$ $hl_{ik} = d_{ik} + b_{1_{ik}}$ Add Matrix & Vector
- $hl_{relu} = max(0, hl)$ element wise max(0, x)
- $d_2 = mul(hl_{relu}, W_2)$
- score = d_2 + b_2

gradient berechnen:

- score => alle Werte die <=0 auf 0 setzen
- scores => den Wert der richtigen Klasse um 1 verringern
- scores => Jeden Wert durch batch_size dividieren

- dW_2 = mul(hl_{relu}^T, scores)
- db_2 = summe scores axis=0, wenn batch_size=1 db_2 = scores
- dhidden = mul(scores, W_2^T)
  dhidden => alle Werte <=0 auf 0 setzen
- dW_1 = mul(X_train^T, dhidden)
- db_1 = summe dhidden axis=0, wenn batch_size=1 db_1 = dhidden

dW_2 += reg * W_2
dW_1 += reg * W_1

W_1 += -step_size * dW_1
b_1 += -step_size * db_1
W_2 += -step_size * dW_2
b_2 += -step_size * db_2