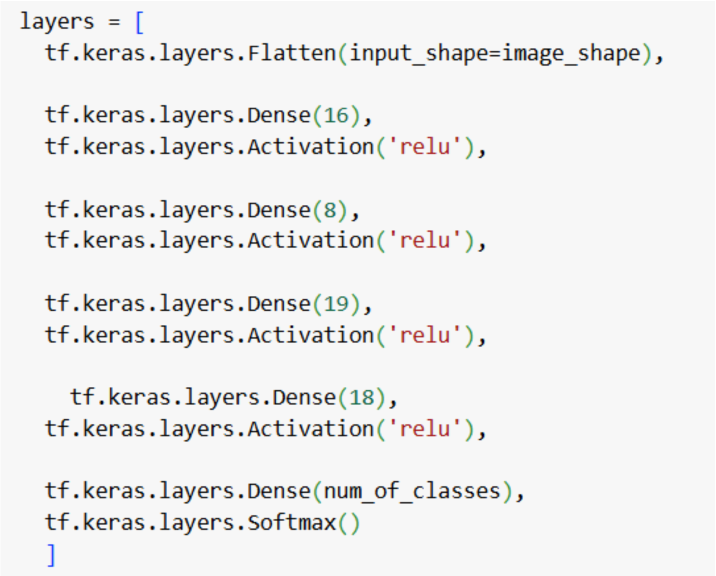
קובץ ניסויים – מטלה 2

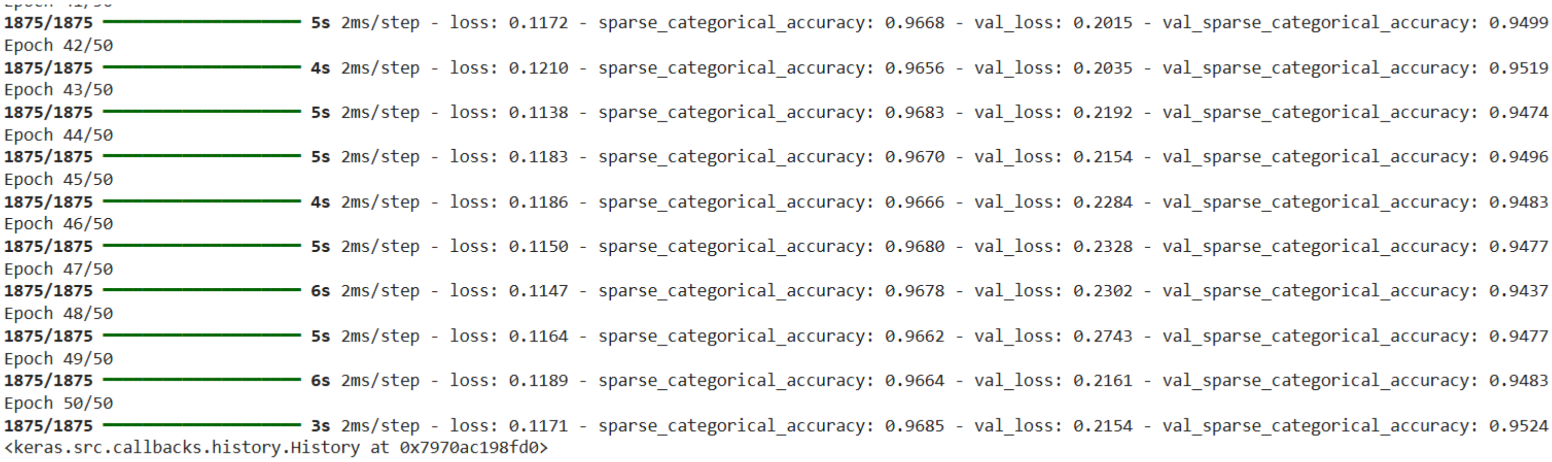
אציג את הניסוי ואז את התוצאה (אציג את האפוקים האחרונים
כמובן) אדגיש שאנחנו מחפשים בעצם את התוצאות הכי טובות אבל
עם זאת ביחד נחפש גם פרמטרים שלא יוצרים over fitting וזה
אומר שאנחנו צריכים לקבל תוצאות בtest קצת יותר טובות מאשר
באימון.
הת.ז שלי היא 318190816 ולכן המספרים יהיו : 16,08,19,18.
08 אכתוב כמובן כ – 8 :

ניסוי 1 :
הפרמטרים:



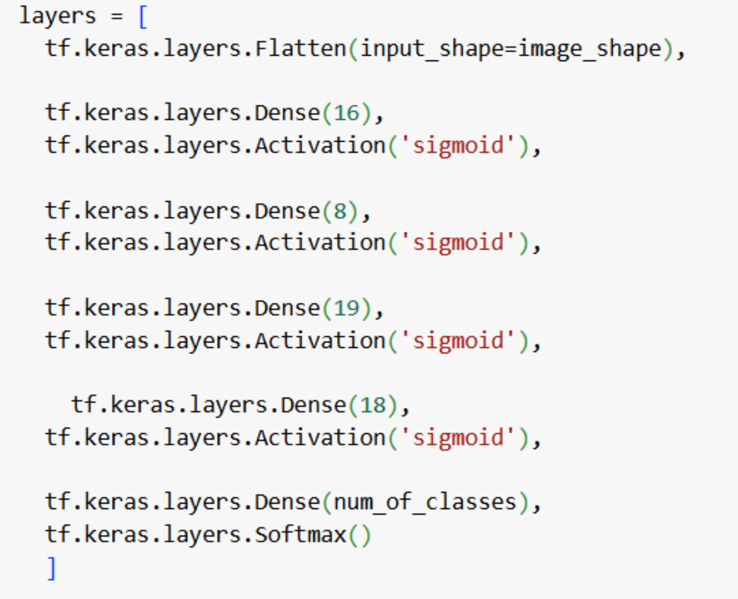
```
layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),

  tf.keras.layers.Dense(16),
  tf.keras.layers.Activation('relu'),

  tf.keras.layers.Dense(8),
  tf.keras.layers.Activation('relu'),

  tf.keras.layers.Dense(19),
  tf.keras.layers.Activation('relu'),

    tf.keras.layers.Dense(18),
  tf.keras.layers.Activation('relu'),

  tf.keras.layers.Dense(num_of_classes),
  tf.keras.layers.Softmax()
  ]
```

התוצאות :



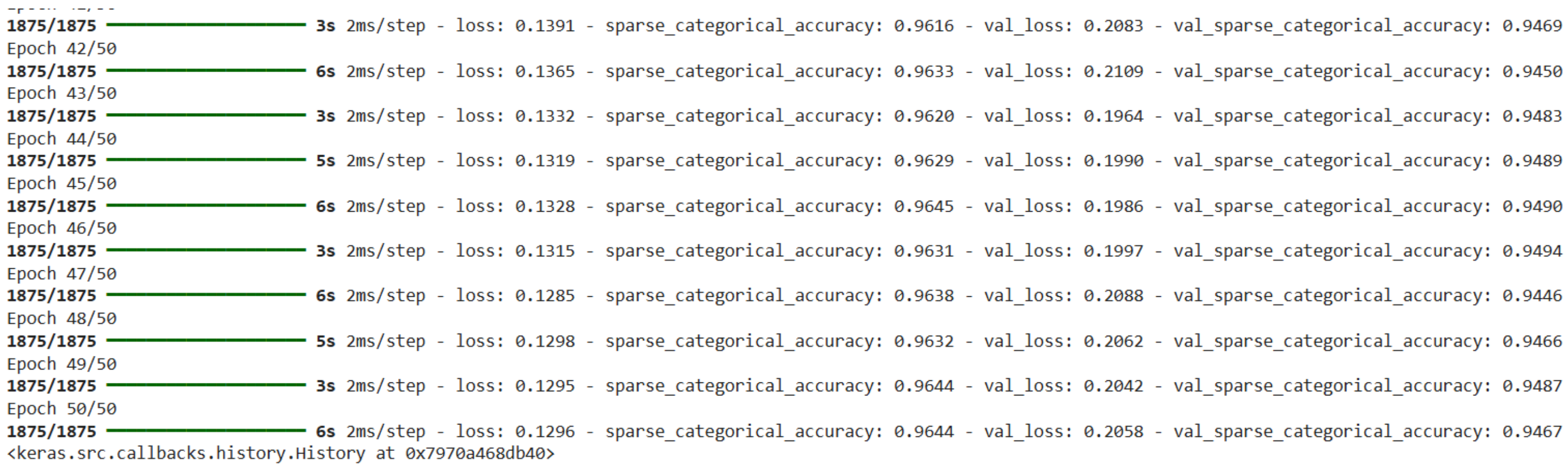
```
Epoch 41/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.1172 - sparse_categorical_accuracy: 0.9668 - val_loss: 0.2015 - val_sparse_categorical_accuracy: 0.9499
Epoch 42/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - loss: 0.1210 - sparse_categorical_accuracy: 0.9656 - val_loss: 0.2035 - val_sparse_categorical_accuracy: 0.9519
Epoch 43/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.1138 - sparse_categorical_accuracy: 0.9683 - val_loss: 0.2192 - val_sparse_categorical_accuracy: 0.9474
Epoch 44/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.1183 - sparse_categorical_accuracy: 0.9670 - val_loss: 0.2154 - val_sparse_categorical_accuracy: 0.9496
Epoch 45/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - loss: 0.1186 - sparse_categorical_accuracy: 0.9666 - val_loss: 0.2284 - val_sparse_categorical_accuracy: 0.9483
Epoch 46/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.1150 - sparse_categorical_accuracy: 0.9680 - val_loss: 0.2328 - val_sparse_categorical_accuracy: 0.9477
Epoch 47/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - loss: 0.1147 - sparse_categorical_accuracy: 0.9678 - val_loss: 0.2302 - val_sparse_categorical_accuracy: 0.9437
Epoch 48/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.1164 - sparse_categorical_accuracy: 0.9662 - val_loss: 0.2743 - val_sparse_categorical_accuracy: 0.9477
Epoch 49/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - loss: 0.1189 - sparse_categorical_accuracy: 0.9664 - val_loss: 0.2161 - val_sparse_categorical_accuracy: 0.9483
Epoch 50/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 0.1171 - sparse_categorical_accuracy: 0.9685 - val_loss: 0.2154 - val_sparse_categorical_accuracy: 0.9524
<keras.src.callbacks.history.History at 0x7970ac198fd0>
```

ניסוי 2 : החלפתי את האקטיבציה לסיגמויד במקום ראלו שהיה ליפני
והתוצאות ממש קצת התקלקלו:
הפרמטרים:



```
layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),

  tf.keras.layers.Dense(16),
  tf.keras.layers.Activation('sigmoid'),

  tf.keras.layers.Dense(8),
  tf.keras.layers.Activation('sigmoid'),

  tf.keras.layers.Dense(19),
  tf.keras.layers.Activation('sigmoid'),

    tf.keras.layers.Dense(18),
  tf.keras.layers.Activation('sigmoid'),

  tf.keras.layers.Dense(num_of_classes),
  tf.keras.layers.Softmax()
  ]
```

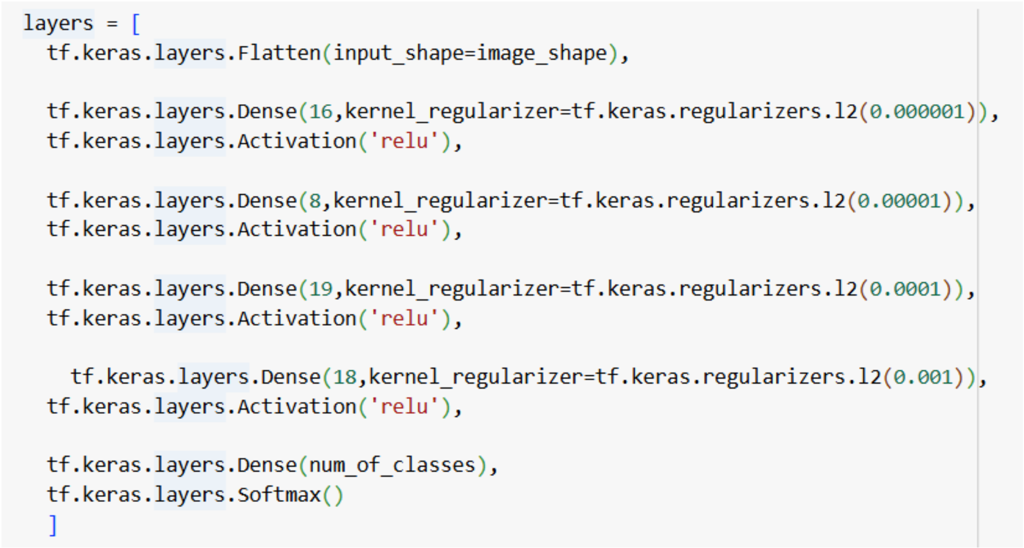
התוצאות :



```
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 0.1391 - sparse_categorical_accuracy: 0.9616 - val_loss: 0.2083 - val_sparse_categorical_accuracy: 0.9469
Epoch 42/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - loss: 0.1365 - sparse_categorical_accuracy: 0.9633 - val_loss: 0.2109 - val_sparse_categorical_accuracy: 0.9450
Epoch 43/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 0.1332 - sparse_categorical_accuracy: 0.9620 - val_loss: 0.1964 - val_sparse_categorical_accuracy: 0.9483
Epoch 44/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.1319 - sparse_categorical_accuracy: 0.9629 - val_loss: 0.1990 - val_sparse_categorical_accuracy: 0.9489
Epoch 45/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - loss: 0.1328 - sparse_categorical_accuracy: 0.9645 - val_loss: 0.1986 - val_sparse_categorical_accuracy: 0.9490
Epoch 46/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 0.1315 - sparse_categorical_accuracy: 0.9631 - val_loss: 0.1997 - val_sparse_categorical_accuracy: 0.9494
Epoch 47/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - loss: 0.1285 - sparse_categorical_accuracy: 0.9638 - val_loss: 0.2088 - val_sparse_categorical_accuracy: 0.9446
Epoch 48/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.1298 - sparse_categorical_accuracy: 0.9632 - val_loss: 0.2062 - val_sparse_categorical_accuracy: 0.9466
Epoch 49/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 0.1295 - sparse_categorical_accuracy: 0.9644 - val_loss: 0.2042 - val_sparse_categorical_accuracy: 0.9487
Epoch 50/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - loss: 0.1296 - sparse_categorical_accuracy: 0.9644 - val_loss: 0.2058 - val_sparse_categorical_accuracy: 0.9467
<keras.src.callbacks.history.History at 0x7970a468db40>
```

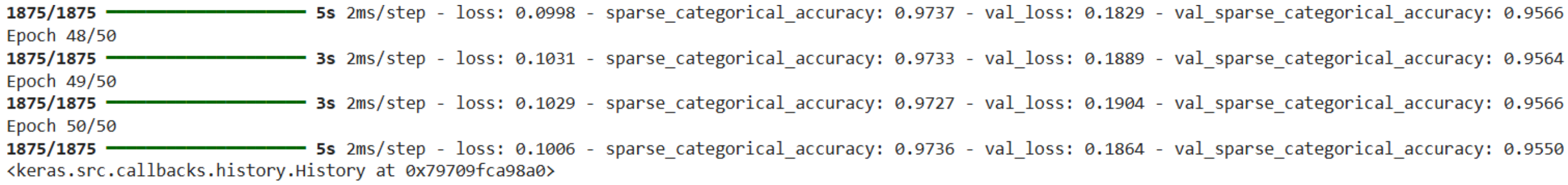
ניסוי 3 :

הפרמטרים:



```
layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),

  tf.keras.layers.Dense(16,kernel_regularizer=tf.keras.regularizers.l2(0.000001)),
  tf.keras.layers.Activation('relu'),

  tf.keras.layers.Dense(8,kernel_regularizer=tf.keras.regularizers.l2(0.00001)),
  tf.keras.layers.Activation('relu'),

  tf.keras.layers.Dense(19,kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
  tf.keras.layers.Activation('relu'),

    tf.keras.layers.Dense(18,kernel_regularizer=tf.keras.regularizers.l2(0.001)),
  tf.keras.layers.Activation('relu'),

  tf.keras.layers.Dense(num_of_classes),
  tf.keras.layers.Softmax()
  ]
```

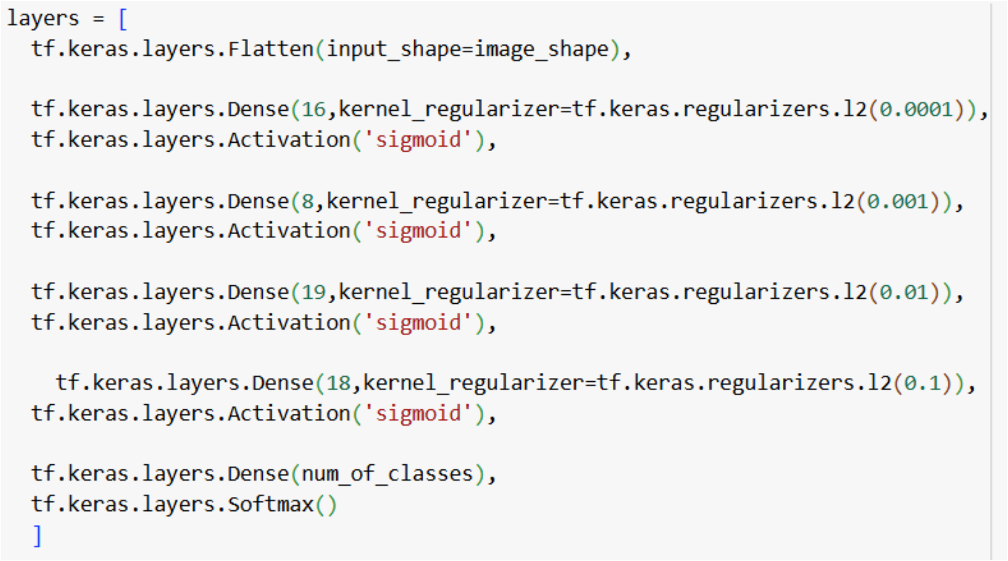
התוצאות :



```
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.0998 - sparse_categorical_accuracy: 0.9737 - val_loss: 0.1829 - val_sparse_categorical_accuracy: 0.9566
Epoch 48/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 0.1031 - sparse_categorical_accuracy: 0.9733 - val_loss: 0.1889 - val_sparse_categorical_accuracy: 0.9564
Epoch 49/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 0.1029 - sparse_categorical_accuracy: 0.9727 - val_loss: 0.1904 - val_sparse_categorical_accuracy: 0.9566
Epoch 50/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.1006 - sparse_categorical_accuracy: 0.9736 - val_loss: 0.1864 - val_sparse_categorical_accuracy: 0.9550
<keras.src.callbacks.history.History at 0x79709fca98a0>
```

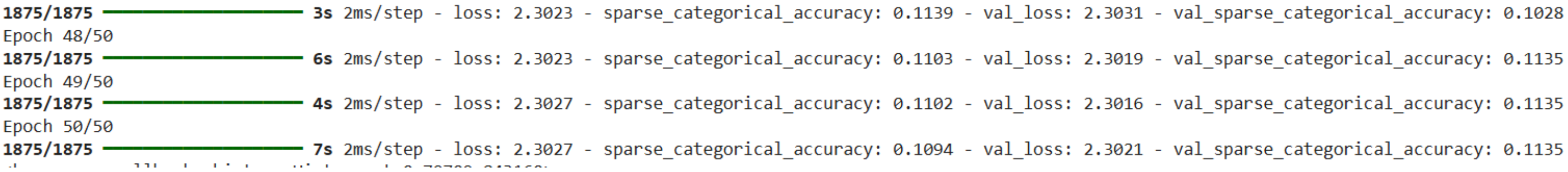
ניסוי 4 : דוגמא לתוצאות ממש גרועות :

הפרמטרים:



```
layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),

  tf.keras.layers.Dense(16,kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
  tf.keras.layers.Activation('sigmoid'),

  tf.keras.layers.Dense(8,kernel_regularizer=tf.keras.regularizers.l2(0.001)),
  tf.keras.layers.Activation('sigmoid'),

  tf.keras.layers.Dense(19,kernel_regularizer=tf.keras.regularizers.l2(0.01)),
  tf.keras.layers.Activation('sigmoid'),

    tf.keras.layers.Dense(18,kernel_regularizer=tf.keras.regularizers.l2(0.1)),
  tf.keras.layers.Activation('sigmoid'),

  tf.keras.layers.Dense(num_of_classes),
  tf.keras.layers.Softmax()
  ]
```

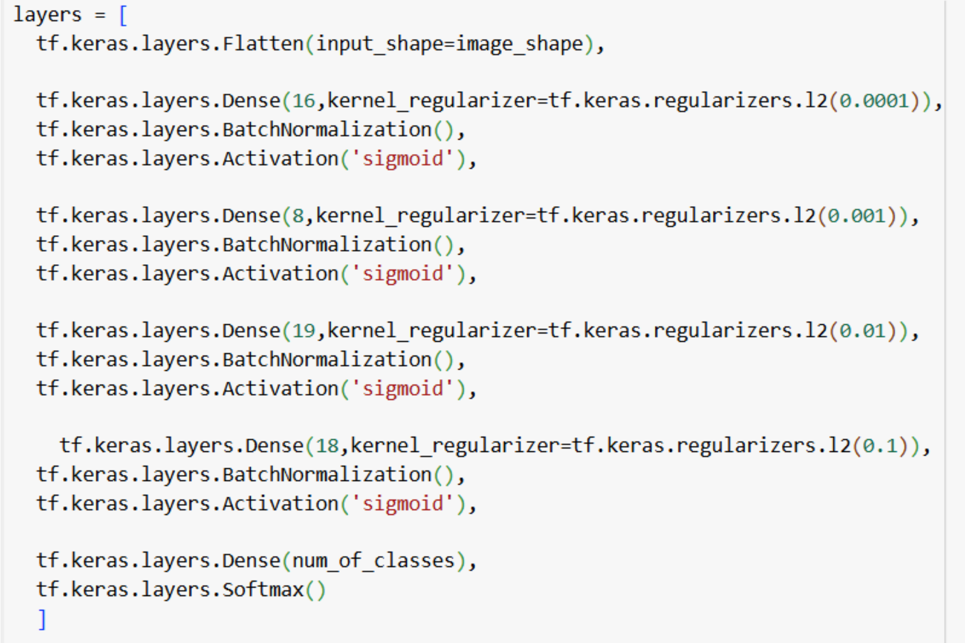
התוצאות :



```
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 2.3023 - sparse_categorical_accuracy: 0.1139 - val_loss: 2.3031 - val_sparse_categorical_accuracy: 0.1028
Epoch 48/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - loss: 2.3023 - sparse_categorical_accuracy: 0.1103 - val_loss: 2.3019 - val_sparse_categorical_accuracy: 0.1135
Epoch 49/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - loss: 2.3027 - sparse_categorical_accuracy: 0.1102 - val_loss: 2.3016 - val_sparse_categorical_accuracy: 0.1135
Epoch 50/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 2ms/step - loss: 2.3027 - sparse_categorical_accuracy: 0.1094 - val_loss: 2.3021 - val_sparse_categorical_accuracy: 0.1135
```

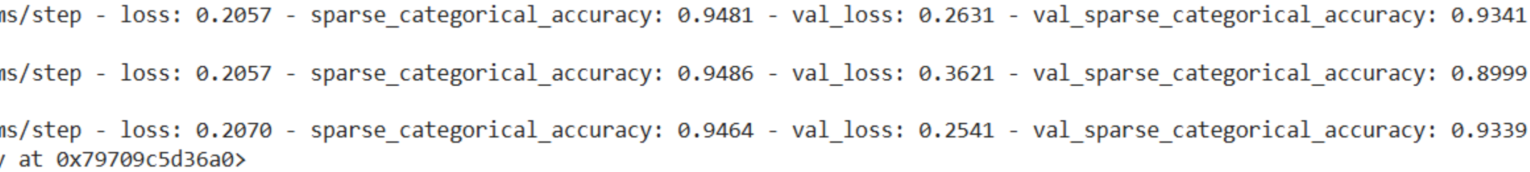
ניסוי 5 : מעניין לראות שנקח את הפרמטרים האלה שנתנו לנו
תוצאות ממש גרועות ונוסיף BATCHNORMALIZATION ונראה
שהתוצאות ישתפרו בפער מטורף :

הפרמטרים:



```
layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),

  tf.keras.layers.Dense(16,kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('sigmoid'),

  tf.keras.layers.Dense(8,kernel_regularizer=tf.keras.regularizers.l2(0.001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('sigmoid'),

  tf.keras.layers.Dense(19,kernel_regularizer=tf.keras.regularizers.l2(0.01)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('sigmoid'),

    tf.keras.layers.Dense(18,kernel_regularizer=tf.keras.regularizers.l2(0.1)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('sigmoid'),

  tf.keras.layers.Dense(num_of_classes),
  tf.keras.layers.Softmax()
  ]
```

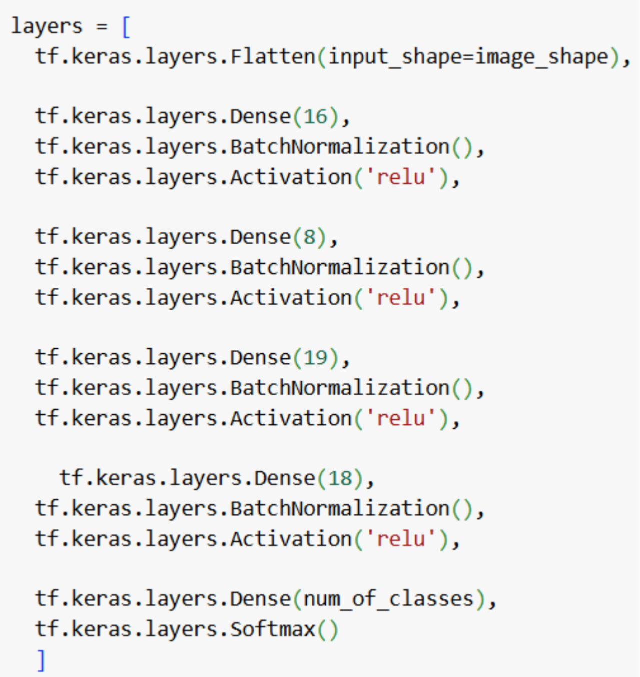
התוצאות :



```
ms/step - loss: 0.2057 - sparse_categorical_accuracy: 0.9481 - val_loss: 0.2631 - val_sparse_categorical_accuracy: 0.9341

ms/step - loss: 0.2057 - sparse_categorical_accuracy: 0.9486 - val_loss: 0.3621 - val_sparse_categorical_accuracy: 0.8999

ms/step - loss: 0.2070 - sparse_categorical_accuracy: 0.9464 - val_loss: 0.2541 - val_sparse_categorical_accuracy: 0.9339
/ at 0x79709c5d36a0>
```

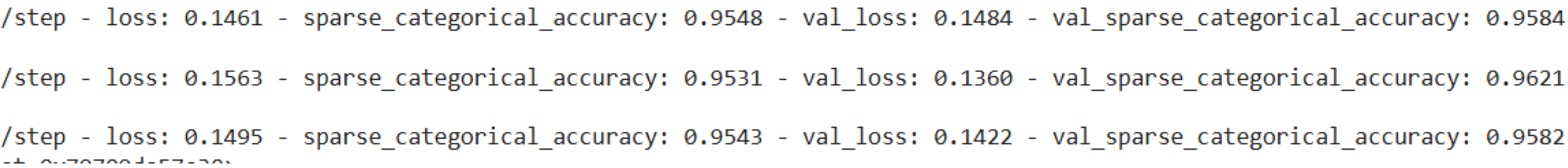
ניסוי 6 : :

הפרמטרים:



```
layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),

  tf.keras.layers.Dense(16),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),

  tf.keras.layers.Dense(8),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),

  tf.keras.layers.Dense(19),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),

    tf.keras.layers.Dense(18),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),

  tf.keras.layers.Dense(num_of_classes),
  tf.keras.layers.Softmax()
  ]
```

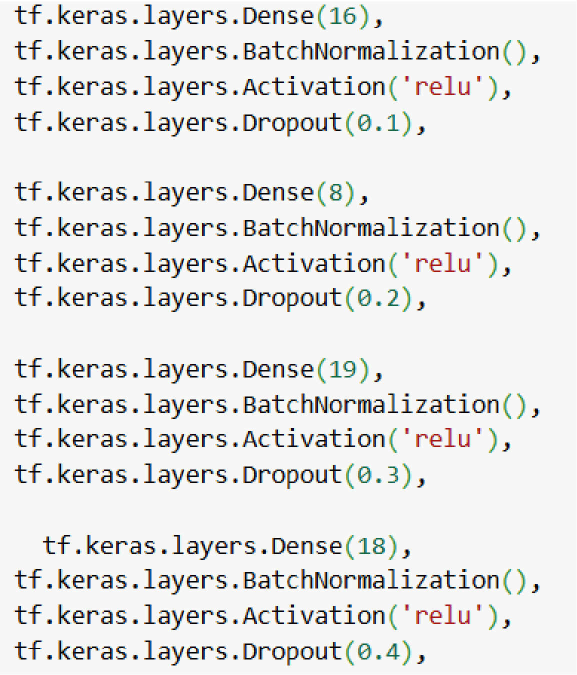
התוצאות :



```
/step - loss: 0.1461 - sparse_categorical_accuracy: 0.9548 - val_loss: 0.1484 - val_sparse_categorical_accuracy: 0.9584

/step - loss: 0.1563 - sparse_categorical_accuracy: 0.9531 - val_loss: 0.1360 - val_sparse_categorical_accuracy: 0.9621

/step - loss: 0.1495 - sparse_categorical_accuracy: 0.9543 - val_loss: 0.1422 - val_sparse_categorical_accuracy: 0.9582
```

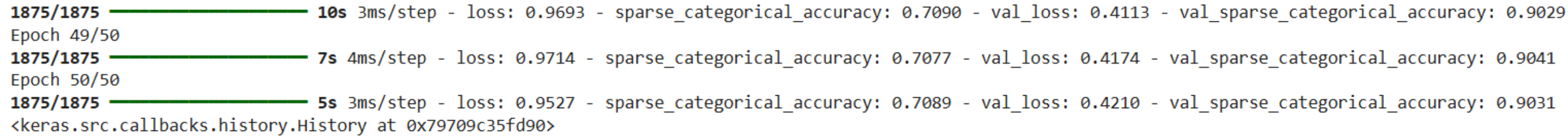
ניסוי 7 :

הפרמטרים:



```
tf.keras.layers.Dense(16),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Activation('relu'),
tf.keras.layers.Dropout(0.1),

tf.keras.layers.Dense(8),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Activation('relu'),
tf.keras.layers.Dropout(0.2),

tf.keras.layers.Dense(19),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Activation('relu'),
tf.keras.layers.Dropout(0.3),

  tf.keras.layers.Dense(18),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Activation('relu'),
tf.keras.layers.Dropout(0.4),
```

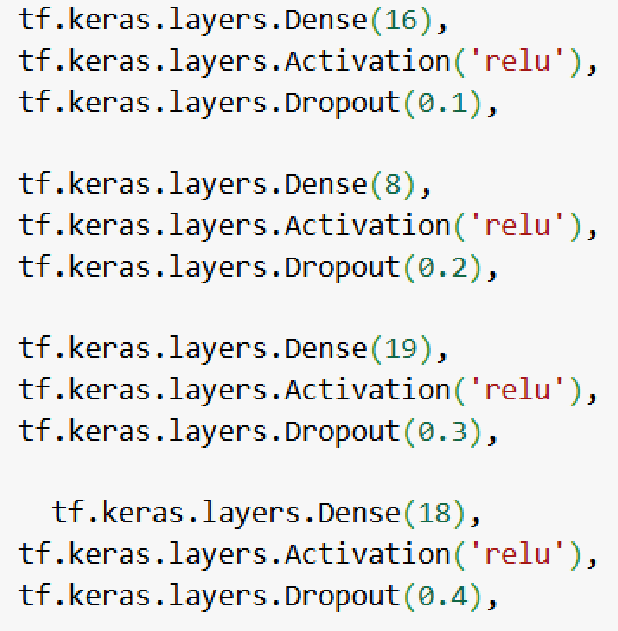
התוצאות :



```
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - loss: 0.9693 - sparse_categorical_accuracy: 0.7090 - val_loss: 0.4113 - val_sparse_categorical_accuracy: 0.9029
Epoch 49/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - loss: 0.9714 - sparse_categorical_accuracy: 0.7077 - val_loss: 0.4174 - val_sparse_categorical_accuracy: 0.9041
Epoch 50/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - loss: 0.9527 - sparse_categorical_accuracy: 0.7089 - val_loss: 0.4210 - val_sparse_categorical_accuracy: 0.9031
<keras.src.callbacks.history.History at 0x79709c35fd90>
```

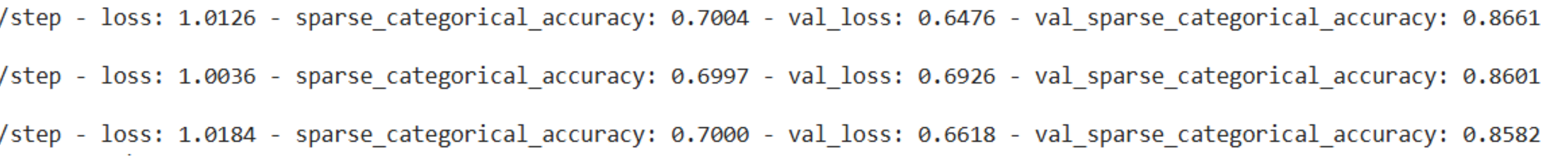
ניסוי 8: ופה נראה שאין אובר פיטינג אבל התוצאות לא הכי מוצלחות:

הפרמטרים:



```
tf.keras.layers.Dense(16),
tf.keras.layers.Activation('relu'),
tf.keras.layers.Dropout(0.1),

tf.keras.layers.Dense(8),
tf.keras.layers.Activation('relu'),
tf.keras.layers.Dropout(0.2),

tf.keras.layers.Dense(19),
tf.keras.layers.Activation('relu'),
tf.keras.layers.Dropout(0.3),

  tf.keras.layers.Dense(18),
tf.keras.layers.Activation('relu'),
tf.keras.layers.Dropout(0.4),
```

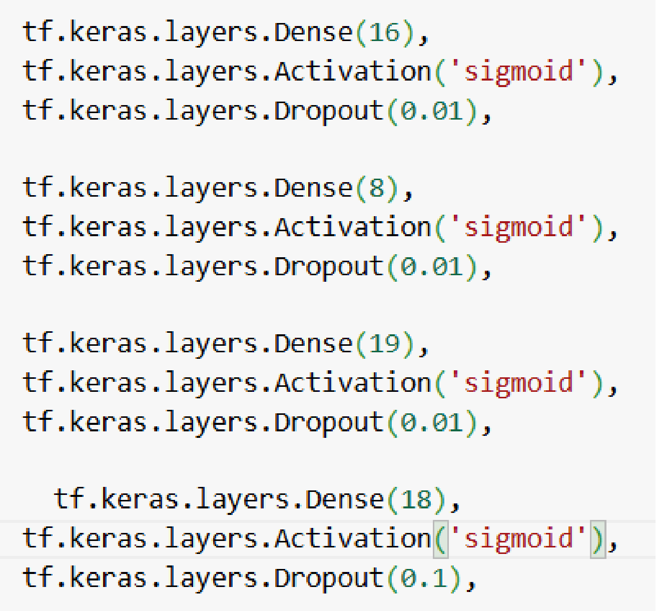
התוצאות :



```
/step - loss: 1.0126 - sparse_categorical_accuracy: 0.7004 - val_loss: 0.6476 - val_sparse_categorical_accuracy: 0.8661

/step - loss: 1.0036 - sparse_categorical_accuracy: 0.6997 - val_loss: 0.6926 - val_sparse_categorical_accuracy: 0.8601

/step - loss: 1.0184 - sparse_categorical_accuracy: 0.7000 - val_loss: 0.6618 - val_sparse_categorical_accuracy: 0.8582
```

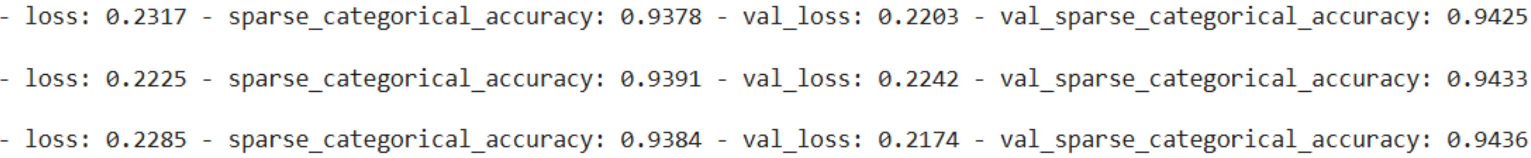
ניסוי 9 : :

הפרמטרים:



```
tf.keras.layers.Dense(16),
tf.keras.layers.Activation('sigmoid'),
tf.keras.layers.Dropout(0.01),

tf.keras.layers.Dense(8),
tf.keras.layers.Activation('sigmoid'),
tf.keras.layers.Dropout(0.01),

tf.keras.layers.Dense(19),
tf.keras.layers.Activation('sigmoid'),
tf.keras.layers.Dropout(0.01),

  tf.keras.layers.Dense(18),
tf.keras.layers.Activation('sigmoid'),
tf.keras.layers.Dropout(0.1),
```

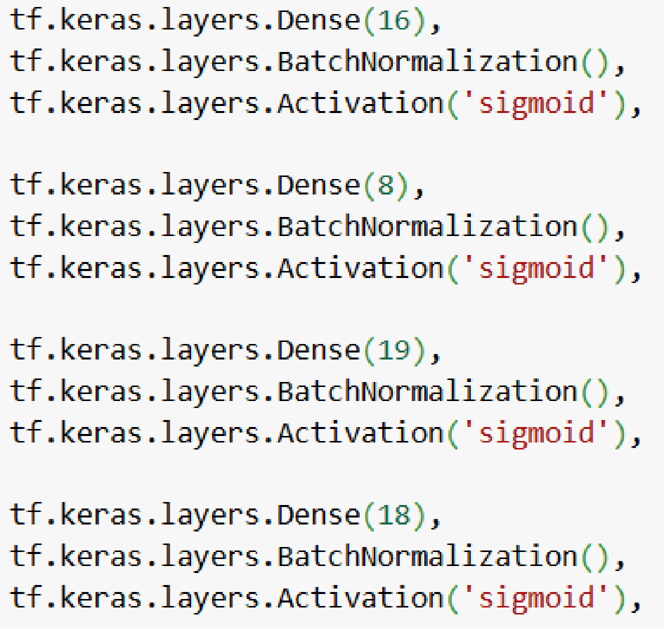
התוצאות :



```
- loss: 0.2317 - sparse_categorical_accuracy: 0.9378 - val_loss: 0.2203 - val_sparse_categorical_accuracy: 0.9425

- loss: 0.2225 - sparse_categorical_accuracy: 0.9391 - val_loss: 0.2242 - val_sparse_categorical_accuracy: 0.9433

- loss: 0.2285 - sparse_categorical_accuracy: 0.9384 - val_loss: 0.2174 - val_sparse_categorical_accuracy: 0.9436
```

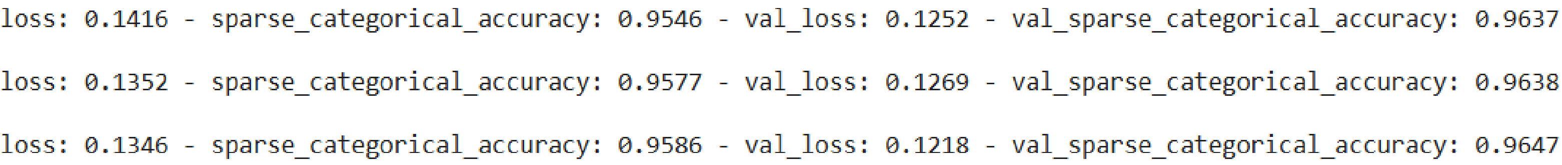
ניסוי 10 : :

הפרמטרים:



```
tf.keras.layers.Dense(16),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Activation('sigmoid'),

tf.keras.layers.Dense(8),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Activation('sigmoid'),

tf.keras.layers.Dense(19),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Activation('sigmoid'),

tf.keras.layers.Dense(18),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Activation('sigmoid'),
```

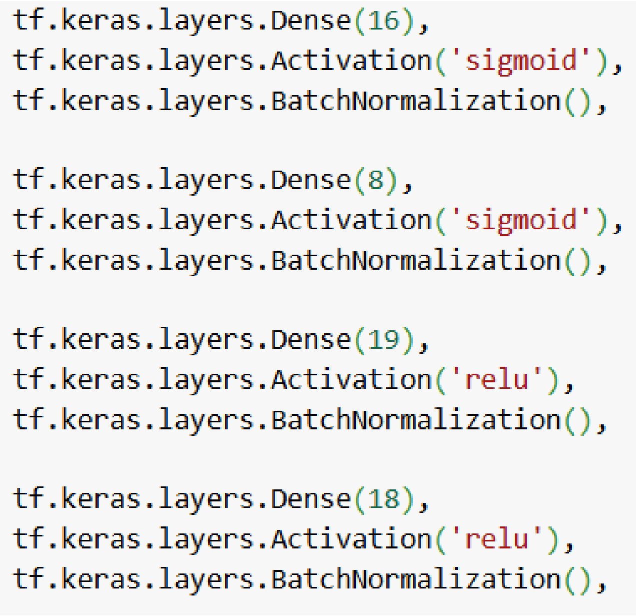
התוצאות :



```
loss: 0.1416 - sparse_categorical_accuracy: 0.9546 - val_loss: 0.1252 - val_sparse_categorical_accuracy: 0.9637

loss: 0.1352 - sparse_categorical_accuracy: 0.9577 - val_loss: 0.1269 - val_sparse_categorical_accuracy: 0.9638

loss: 0.1346 - sparse_categorical_accuracy: 0.9586 - val_loss: 0.1218 - val_sparse_categorical_accuracy: 0.9647
```

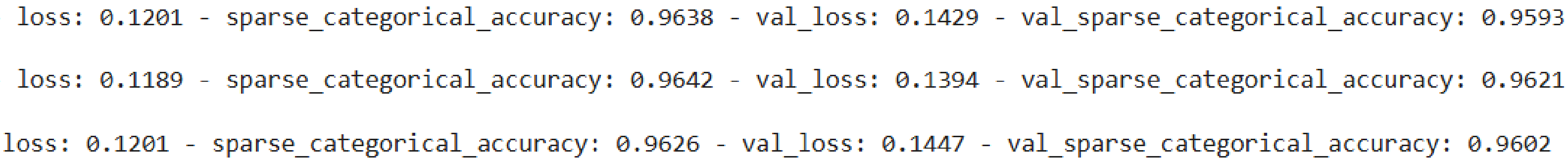
ניסוי 11 : :

הפרמטרים:



```
tf.keras.layers.Dense(16),
tf.keras.layers.Activation('sigmoid'),
tf.keras.layers.BatchNormalization(),

tf.keras.layers.Dense(8),
tf.keras.layers.Activation('sigmoid'),
tf.keras.layers.BatchNormalization(),

tf.keras.layers.Dense(19),
tf.keras.layers.Activation('relu'),
tf.keras.layers.BatchNormalization(),

tf.keras.layers.Dense(18),
tf.keras.layers.Activation('relu'),
tf.keras.layers.BatchNormalization(),
```

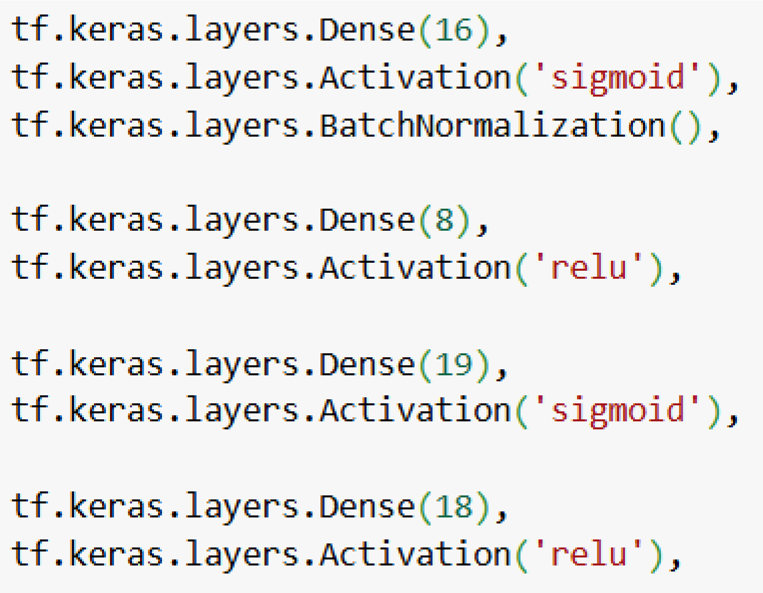
התוצאות :



```
loss: 0.1201 - sparse_categorical_accuracy: 0.9638 - val_loss: 0.1429 - val_sparse_categorical_accuracy: 0.9593

loss: 0.1189 - sparse_categorical_accuracy: 0.9642 - val_loss: 0.1394 - val_sparse_categorical_accuracy: 0.9621

loss: 0.1201 - sparse_categorical_accuracy: 0.9626 - val_loss: 0.1447 - val_sparse_categorical_accuracy: 0.9602
```

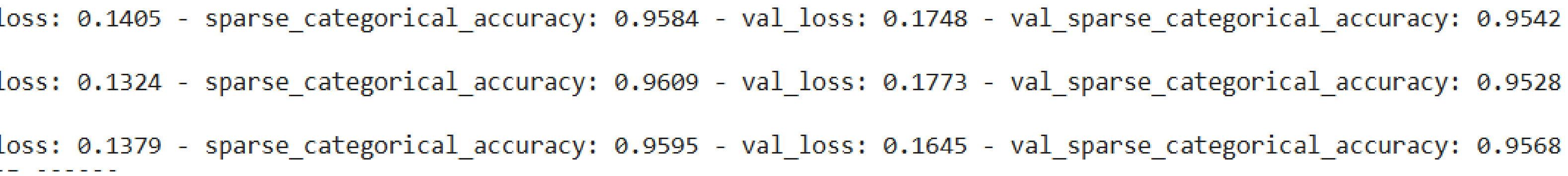
ניסוי 12 : :

הפרמטרים:



```
tf.keras.layers.Dense(16),
tf.keras.layers.Activation('sigmoid'),
tf.keras.layers.BatchNormalization(),

tf.keras.layers.Dense(8),
tf.keras.layers.Activation('relu'),

tf.keras.layers.Dense(19),
tf.keras.layers.Activation('sigmoid'),

tf.keras.layers.Dense(18),
tf.keras.layers.Activation('relu'),
```

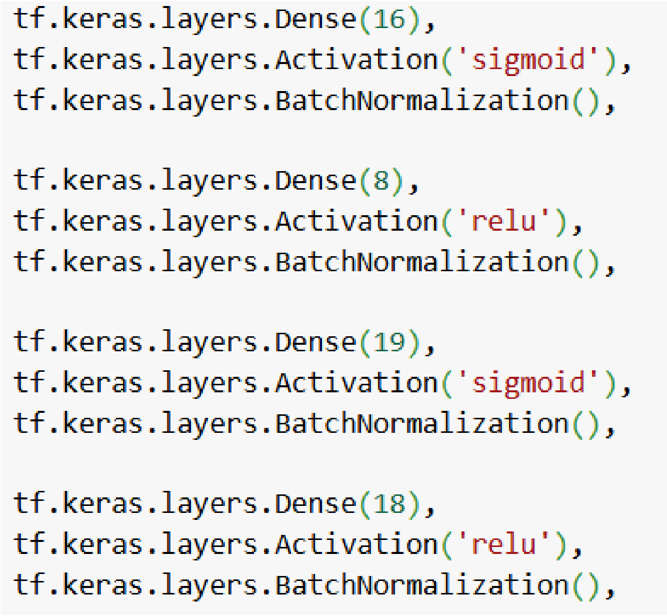
התוצאות :



```
loss: 0.1405 - sparse_categorical_accuracy: 0.9584 - val_loss: 0.1748 - val_sparse_categorical_accuracy: 0.9542

loss: 0.1324 - sparse_categorical_accuracy: 0.9609 - val_loss: 0.1773 - val_sparse_categorical_accuracy: 0.9528

loss: 0.1379 - sparse_categorical_accuracy: 0.9595 - val_loss: 0.1645 - val_sparse_categorical_accuracy: 0.9568
-- ------
```

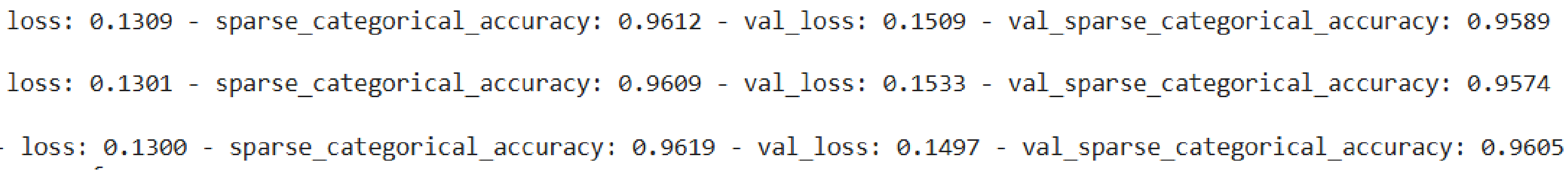
ניסוי 13 : :

הפרמטרים:



```
tf.keras.layers.Dense(16),
tf.keras.layers.Activation('sigmoid'),
tf.keras.layers.BatchNormalization(),

tf.keras.layers.Dense(8),
tf.keras.layers.Activation('relu'),
tf.keras.layers.BatchNormalization(),

tf.keras.layers.Dense(19),
tf.keras.layers.Activation('sigmoid'),
tf.keras.layers.BatchNormalization(),

tf.keras.layers.Dense(18),
tf.keras.layers.Activation('relu'),
tf.keras.layers.BatchNormalization(),
```

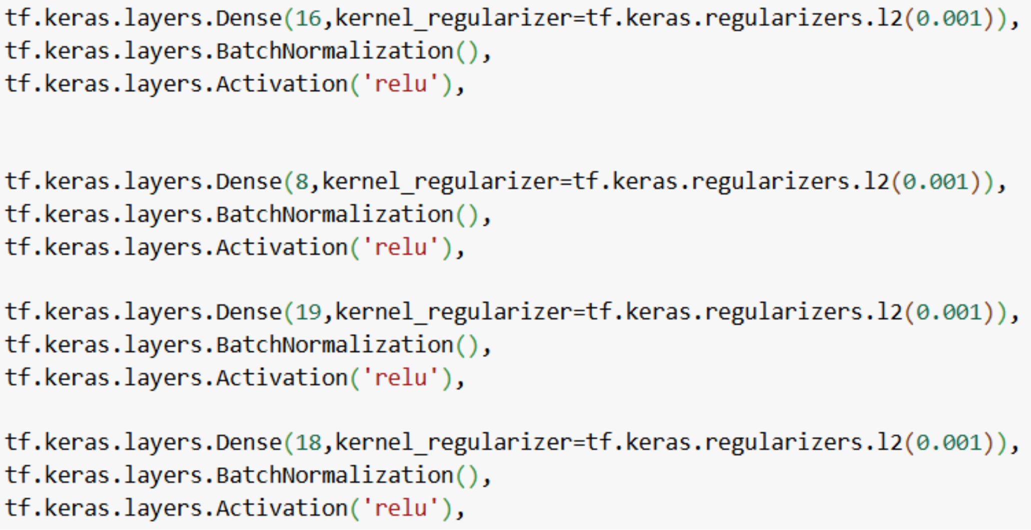
התוצאות :



```
loss: 0.1309 - sparse_categorical_accuracy: 0.9612 - val_loss: 0.1509 - val_sparse_categorical_accuracy: 0.9589

loss: 0.1301 - sparse_categorical_accuracy: 0.9609 - val_loss: 0.1533 - val_sparse_categorical_accuracy: 0.9574

- loss: 0.1300 - sparse_categorical_accuracy: 0.9619 - val_loss: 0.1497 - val_sparse_categorical_accuracy: 0.9605
```

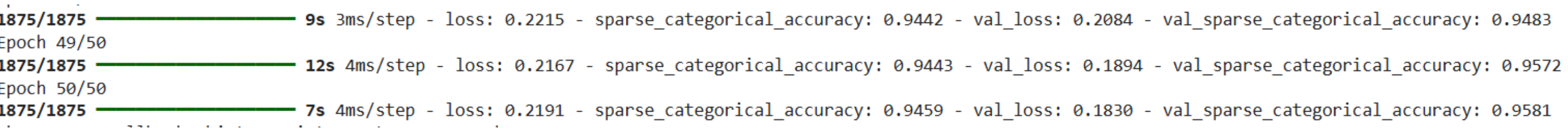
ניסוי 14 : :

הפרמטרים:



```
tf.keras.layers.Dense(16,kernel_regularizer=tf.keras.regularizers.l2(0.001)),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Activation('relu'),


tf.keras.layers.Dense(8,kernel_regularizer=tf.keras.regularizers.l2(0.001)),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Activation('relu'),

tf.keras.layers.Dense(19,kernel_regularizer=tf.keras.regularizers.l2(0.001)),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Activation('relu'),

tf.keras.layers.Dense(18,kernel_regularizer=tf.keras.regularizers.l2(0.001)),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Activation('relu'),
```

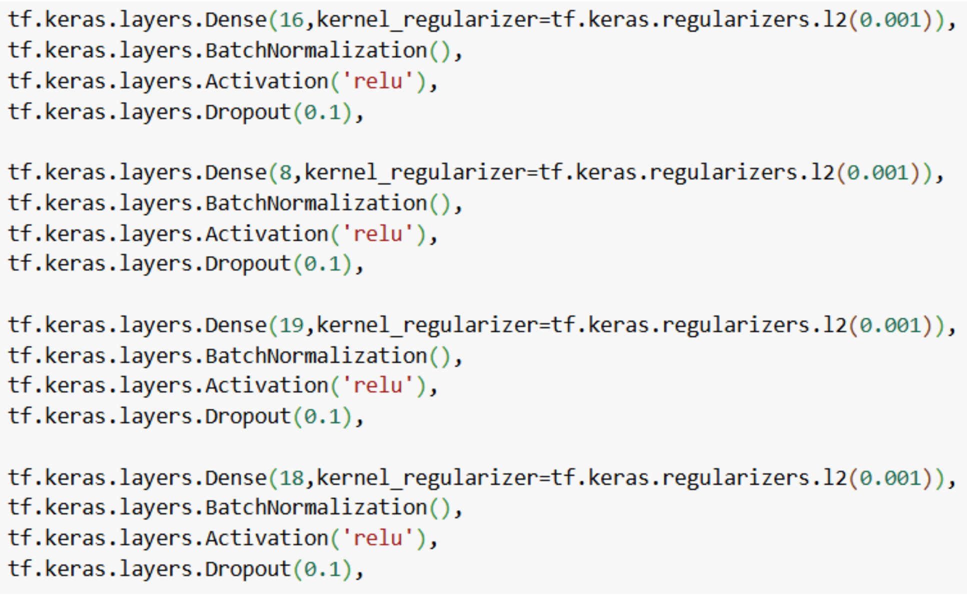
התוצאות :



```
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - loss: 0.2215 - sparse_categorical_accuracy: 0.9442 - val_loss: 0.2084 - val_sparse_categorical_accuracy: 0.9483
Epoch 49/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 12s 4ms/step - loss: 0.2167 - sparse_categorical_accuracy: 0.9443 - val_loss: 0.1894 - val_sparse_categorical_accuracy: 0.9572
Epoch 50/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - loss: 0.2191 - sparse_categorical_accuracy: 0.9459 - val_loss: 0.1830 - val_sparse_categorical_accuracy: 0.9581
```

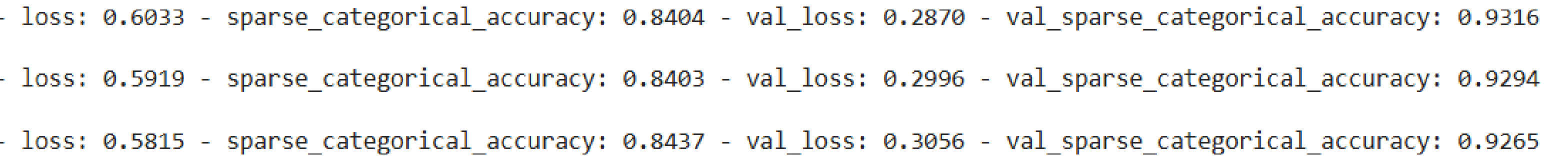
ניסוי 15 : :

הפרמטרים:



```
tf.keras.layers.Dense(16,kernel_regularizer=tf.keras.regularizers.l2(0.001)),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Activation('relu'),
tf.keras.layers.Dropout(0.1),

tf.keras.layers.Dense(8,kernel_regularizer=tf.keras.regularizers.l2(0.001)),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Activation('relu'),
tf.keras.layers.Dropout(0.1),

tf.keras.layers.Dense(19,kernel_regularizer=tf.keras.regularizers.l2(0.001)),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Activation('relu'),
tf.keras.layers.Dropout(0.1),

tf.keras.layers.Dense(18,kernel_regularizer=tf.keras.regularizers.l2(0.001)),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Activation('relu'),
tf.keras.layers.Dropout(0.1),
```

התוצאות :



```
- loss: 0.6033 - sparse_categorical_accuracy: 0.8404 - val_loss: 0.2870 - val_sparse_categorical_accuracy: 0.9316

- loss: 0.5919 - sparse_categorical_accuracy: 0.8403 - val_loss: 0.2996 - val_sparse_categorical_accuracy: 0.9294

- loss: 0.5815 - sparse_categorical_accuracy: 0.8437 - val_loss: 0.3056 - val_sparse_categorical_accuracy: 0.9265
```

התוצאות הכי טובות הם מניסוי 10 והן יצאו גם ללא התאמת יתר אז לכן נשאיר בקוד את הפרמטרים של ניסוי 10.