ערכי דיפולט:

```
layers = [
    tf.keras.layers.Flatten(input_shape=image_shape),
    tf.keras.layers.Dense(10),
    tf.keras.layers.Dense(92),
    tf.keras.layers.Dense(87),
    tf.keras.layers.Dense(18),
    tf.keras.layers.Dense(num_of_classes),
    tf.keras.layers.Softmax()
```

```
Best performance at epoch 49 with validation sparse categorical accuracy
of 0.9262
Epoch 49/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - loss: 0.2551 -
sparse_categorical_accuracy: 0.9312 - val_loss: 0.2770 -
val_sparse_categorical_accuracy: 0.9262
```

ניסוי מספר 1 : הוספת אקטיבציה , התאמת יתר

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

    tf.keras.layers.Dense(10),
    tf.keras.layers.Activation('sigmoid'),

    tf.keras.layers.Dense(92),
    tf.keras.layers.Activation('sigmoid'),

    tf.keras.layers.Dense(87),
    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()
]
```

```
Best performance at epoch 47 with validation sparse categorical accuracy
of 0.9358
Epoch 47/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.1939 -
sparse_categorical_accuracy: 0.9433 - val_loss: 0.2289 -
val_sparse_categorical_accuracy: 0.9358
```

ניסוי מספר 2:שינוי ל RELU, התאמת יתר

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

    tf.keras.layers.Dense(10),
    tf.keras.layers.Activation('relu'),

    tf.keras.layers.Dense(92),
    tf.keras.layers.Activation('relu'),

    tf.keras.layers.Dense(87),
    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()
]
```

```
Best performance at epoch 12 with validation sparse categorical accuracy
of 0.9550
Epoch 12/50
1875/1875 ──────────────────────────── 4s 2ms/step - loss: 0.1349 -
sparse_categorical_accuracy: 0.9607 - val_loss: 0.1780 -
val_sparse_categorical_accuracy: 0.9550
```

ניסוי מספר 3: הוספת BATCH NORMALIZITION עדיין מעט התאמת יתר

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

    tf.keras.layers.Dense(10),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('relu'),

    tf.keras.layers.Dense(92),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('relu'),

    tf.keras.layers.Dense(87),
    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()
]
```

Best performance at epoch 26 with validation sparse categorical accuracy of 0.9666
Epoch 26/50
1875/1875 ——————————————————— **10s** 2ms/step - loss: 0.0947 - sparse_categorical_accuracy: 0.9687 - val_loss: 0.1342 - val_sparse_categorical_accuracy: 0.9666

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

    tf.keras.layers.Dense(10),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('sigmoid'),

    tf.keras.layers.Dense(92),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('sigmoid'),

    tf.keras.layers.Dense(87),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('sigmoid'),

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

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

```
Best performance at epoch 36 with validation sparse categorical accuracy
of 0.9644
Epoch 36/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - loss: 0.1219 -
sparse_categorical_accuracy: 0.9609 - val_loss: 0.1166 -
val_sparse_categorical_accuracy: 0.9644
```

ניסוי מספר 5: ניסיון להוספת DROPOUT והורדת ה חNormalization שהוריד את האחוזים

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

    tf.keras.layers.Dense(10),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Activation('sigmoid'),

    tf.keras.layers.Dense(92),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Activation('sigmoid'),

    tf.keras.layers.Dense(87),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Activation('sigmoid'),

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

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

Best performance at epoch 48 with validation sparse categorical accuracy
of 0.9268
Epoch 48/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **6s** 3ms/step - loss: 0.2934 -
sparse_categorical_accuracy: 0.9149 - val_loss: 0.2687 -
val_sparse_categorical_accuracy: 0.9268

ניסוי מספר 6: שינוי האקטיבציה עדיין עם DROPOUT

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

    tf.keras.layers.Dense(10),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Activation('relu'),

    tf.keras.layers.Dense(92),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Activation('relu'),

    tf.keras.layers.Dense(87),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Activation('relu'),

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

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

```
Best performance at epoch 18 with validation sparse categorical accuracy
of 0.9311
Epoch 18/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - loss: 0.4182 -
sparse_categorical_accuracy: 0.8907 - val_loss: 0.2605 -
val_sparse_categorical_accuracy: 0.9311
```

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

    tf.keras.layers.Dense(10),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Activation('relu'),

    tf.keras.layers.Dense(92),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Activation('relu'),

    tf.keras.layers.Dense(87),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Activation('relu'),

    tf.keras.layers.Dense(18),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Activation('relu'),

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

```
Best performance at epoch 41 with validation sparse categorical accuracy
of 0.9418
Epoch 41/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - loss: 0.3852 -
sparse_categorical_accuracy: 0.8850 - val_loss: 0.2057 -
val_sparse_categorical_accuracy: 0.9418
```

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

    tf.keras.layers.Dense(10),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Activation('sigmoid'),

    tf.keras.layers.Dense(92),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Activation('sigmoid'),

    tf.keras.layers.Dense(87),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Activation('sigmoid'),

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

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

```
Best performance at epoch 39 with validation sparse categorical accuracy
of 0.9530
Epoch 39/50
1875/1875 ——————————————————————— 9s 3ms/step - loss: 0.2852 -
sparse_categorical_accuracy: 0.9141 - val_loss: 0.1616 -
val_sparse_categorical_accuracy: 0.9530
```

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


    tf.keras.layers.Dense(10),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Activation('sigmoid'),


    tf.keras.layers.Dense(92),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Activation('sigmoid'),


    tf.keras.layers.Dense(87),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Activation('sigmoid'),


    tf.keras.layers.Dense(18),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Activation('sigmoid'),


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

Best performance at epoch 45 with validation sparse categorical accuracy of 0.9117
Epoch 45/50
1875/1875 ──────────────────────────────── **6s** 3ms/step - loss: 0.6148 - sparse_categorical_accuracy: 0.8147 - val_loss: 0.3378 - val_sparse_categorical_accuracy: 0.9117

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

tf.keras.layers.Dense(10,kernel_regularizer=tf.keras.regularizers.l2(0.000
5)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Activation('sigmoid'),

tf.keras.layers.Dense(92,kernel_regularizer=tf.keras.regularizers.l2(0.000
5)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Activation('sigmoid'),

tf.keras.layers.Dense(87,kernel_regularizer=tf.keras.regularizers.l2(0.000
5)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Activation('sigmoid'),

tf.keras.layers.Dense(18,kernel_regularizer=tf.keras.regularizers.l2(0.000
5)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Activation('sigmoid'),

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

Best performance at epoch 43 with validation sparse categorical accuracy
of 0.9061
Epoch 43/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━ **8s** 4ms/step - loss: 0.6898 -
sparse_categorical_accuracy: 0.8046 - val_loss: 0.3807 -
val_sparse_categorical_accuracy: 0.9061

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


tf.keras.layers.Dense(10,kernel_regularizer=tf.keras.regularizers.l2(0.000
5)),
    tf.keras.layers.BatchNormalization(),
    #tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Activation('sigmoid'),

tf.keras.layers.Dense(92,kernel_regularizer=tf.keras.regularizers.l2(0.000
5)),
    tf.keras.layers.BatchNormalization(),
    #tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Activation('sigmoid'),

tf.keras.layers.Dense(87,kernel_regularizer=tf.keras.regularizers.l2(0.000
5)),
    tf.keras.layers.BatchNormalization(),
    #tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Activation('sigmoid'),

tf.keras.layers.Dense(18,kernel_regularizer=tf.keras.regularizers.l2(0.000
5)),
    tf.keras.layers.BatchNormalization(),
    #tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Activation('sigmoid'),

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

```
Best performance at epoch 29 with validation sparse categorical accuracy
of 0.9608
Epoch 29/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - loss: 0.1960 -
sparse_categorical_accuracy: 0.9499 - val_loss: 0.1718 -
val_sparse_categorical_accuracy: 0.9608
```

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


tf.keras.layers.Dense(10,kernel_regularizer=tf.keras.regularizers.l2(0.002
5)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('sigmoid'),


tf.keras.layers.Dense(92,kernel_regularizer=tf.keras.regularizers.l2(0.002
5)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('sigmoid'),


tf.keras.layers.Dense(87,kernel_regularizer=tf.keras.regularizers.l2(0.002
5)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('sigmoid'),


tf.keras.layers.Dense(18,kernel_regularizer=tf.keras.regularizers.l2(0.002
5)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('sigmoid'),


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

```
Best performance at epoch 49 with validation sparse categorical accuracy
of 0.9537
Epoch 49/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 10s 4ms/step - loss: 0.2355 -
sparse_categorical_accuracy: 0.9425 - val_loss: 0.2136 -
val_sparse_categorical_accuracy: 0.9537
```

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

tf.keras.layers.Dense(10,kernel_regularizer=tf.keras.regularizers.l2(0.000
5)),
    tf.keras.layers.Activation('relu'),


tf.keras.layers.Dense(92,kernel_regularizer=tf.keras.regularizers.l2(0.000
5)),
    tf.keras.layers.Activation('relu'),


tf.keras.layers.Dense(87,kernel_regularizer=tf.keras.regularizers.l2(0.000
5)),
    tf.keras.layers.Activation('relu'),


tf.keras.layers.Dense(18,kernel_regularizer=tf.keras.regularizers.l2(0.000
5)),
    tf.keras.layers.Activation('relu'),


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


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

```
Best performance at epoch 47 with validation sparse categorical accuracy
of 0.9566
Epoch 47/50
1875/1875 ——————————————————————————— 4s 2ms/step - loss: 0.1750 -
sparse_categorical_accuracy: 0.9612 - val_loss: 0.1948 -
val_sparse_categorical_accuracy: 0.9566
```

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


tf.keras.layers.Dense(10,kernel_regularizer=tf.keras.regularizers.l2(0.000
5)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Activation('relu'),



tf.keras.layers.Dense(92,kernel_regularizer=tf.keras.regularizers.l2(0.000
5)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Activation('relu'),

tf.keras.layers.Dense(87,kernel_regularizer=tf.keras.regularizers.l2(0.000
5)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Activation('relu'),

tf.keras.layers.Dense(18,kernel_regularizer=tf.keras.regularizers.l2(0.000
5)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Activation('relu'),


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

```
Best performance at epoch 40 with validation sparse categorical accuracy
of 0.9340
Epoch 40/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11s 5ms/step - loss: 0.4814 -
sparse_categorical_accuracy: 0.8744 - val_loss: 0.2848 -
val_sparse_categorical_accuracy: 0.9340
```

ניסוי מספר 15:הורדת הDROPOUT והוספת נורמליזציה ורגולריזציה

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


tf.keras.layers.Dense(10,kernel_regularizer=tf.keras.regularizers.l2(0.000
5)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('relu'),


tf.keras.layers.Dense(92,kernel_regularizer=tf.keras.regularizers.l2(0.000
5)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('relu'),


tf.keras.layers.Dense(87,kernel_regularizer=tf.keras.regularizers.l2(0.000
5)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('relu'),


tf.keras.layers.Dense(18,kernel_regularizer=tf.keras.regularizers.l2(0.000
5)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('relu'),


    tf.keras.layers.Dense(num_of_classes),
    tf.keras.layers.Softmax()
]
```
Best performance at epoch 41 with validation sparse categorical accuracy
of 0.9614
Epoch 41/50
1875/1875 ———————————————————————— **5s** 3ms/step - loss: 0.1997 -
sparse_categorical_accuracy: 0.9516 - val_loss: 0.1716 -
val_sparse_categorical_accuracy: 0.9614

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


tf.keras.layers.Dense(10,kernel_regularizer=tf.keras.regularizers.l2(0.000
1)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('relu'),


tf.keras.layers.Dense(92,kernel_regularizer=tf.keras.regularizers.l2(0.000
1)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('relu'),


tf.keras.layers.Dense(87,kernel_regularizer=tf.keras.regularizers.l2(0.000
1)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('relu'),


tf.keras.layers.Dense(18,kernel_regularizer=tf.keras.regularizers.l2(0.000
1)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('relu'),


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

```
Best performance at epoch 44 with validation sparse categorical accuracy
of 0.9629
Epoch 44/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - loss: 0.1563 -
sparse_categorical_accuracy: 0.9600 - val_loss: 0.1556 -
val_sparse_categorical_accuracy: 0.9629
```