מטלה 2

חלק ב:-

Experiment number 1:

```
layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),
  tf.keras.layers.Dense(95, kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),

  tf.keras.layers.Dense(28, kernel_regularizer=tf.keras.regularizers.l2(0.001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),

  tf.keras.layers.Dense(23, kernel_regularizer=tf.keras.regularizers.l2(0.01)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),

   tf.keras.layers.Dense(13, kernel_regularizer=tf.keras.regularizers.l2(0.1)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),

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

Results:

loss: 0.1293 - sparse_categorical_accuracy: 0.9764 - val_loss: 0.1464 - val_sparse_categorical_accuracy: 0.9740

Best result: round-19\50

loss: 0.1571 - sparse_categorical_accuracy: 0.9708 - val_loss: 0.1424 - val_sparse_categorical_accuracy: 0.9774

Experiment number 2:

```
layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),
  tf.keras.layers.Dense(95, kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),
  tf.keras.layers.Dropout(0.1),

  tf.keras.layers.Dense(28, kernel_regularizer=tf.keras.regularizers.l2(0.001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),
  tf.keras.layers.Dropout(0.2),

  tf.keras.layers.Dense(23, kernel_regularizer=tf.keras.regularizers.l2(0.01)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),
  tf.keras.layers.Dropout(0.3),

   tf.keras.layers.Dense(13, kernel_regularizer=tf.keras.regularizers.l2(0.1)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),
  tf.keras.layers.Dropout(0.4),

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

Results:

loss: 0.7382 - sparse_categorical_accuracy: 0.8020 - val_loss: 0.2305 - val_sparse_categorical_accuracy: 0.9712

Best result: round-43\50

loss: 0.7261 - sparse_categorical_accuracy: 0.8064 - val_loss: 0.2258 - val_sparse_categorical_accuracy: 0.9734

Experiment number 3:

```
layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),
  tf.keras.layers.Dense(95, kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('elu'),
  tf.keras.layers.Dropout(0.1),

  tf.keras.layers.Dense(28, kernel_regularizer=tf.keras.regularizers.l2(0.001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('elu'),
  tf.keras.layers.Dropout(0.2),

  tf.keras.layers.Dense(23, kernel_regularizer=tf.keras.regularizers.l2(0.01)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('elu'),
  tf.keras.layers.Dropout(0.3),

   tf.keras.layers.Dense(13, kernel_regularizer=tf.keras.regularizers.l2(0.1)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('elu'),
  tf.keras.layers.Dropout(0.4),

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

Results:

loss: 0.4007 - sparse_categorical_accuracy: 0.9122 - val_loss: 0.1953 - val_sparse_categorical_accuracy: 0.9726

Best result: round-47\50

loss: 0.3968 - sparse_categorical_accuracy: 0.9149 - val_loss: 0.1756 - val_sparse_categorical_accuracy: 0.9762

Experiment number 4:

```
layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),
  tf.keras.layers.Dense(95, kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),
  tf.keras.layers.Dropout(0.1),

  tf.keras.layers.Dense(28, 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(23, kernel_regularizer=tf.keras.regularizers.l2(0.01)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),
  tf.keras.layers.Dropout(0.1),

   tf.keras.layers.Dense(13, kernel_regularizer=tf.keras.regularizers.l2(0.1)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),
  tf.keras.layers.Dropout(0.1),

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

Results:

loss: 0.2823 - sparse_categorical_accuracy: 0.9450 - val_loss: 0.1623 - val_sparse_categorical_accuracy: 0.9773

Best result: round-24\50

loss: 0.2712 - sparse_categorical_accuracy: 0.9481 - val_loss: 0.1630 - val_sparse_categorical_accuracy: 0.9788

Experiment number 5:

```
layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),
  tf.keras.layers.Dense(95, kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('elu'),
  tf.keras.layers.Dropout(0.1),

  tf.keras.layers.Dense(28, kernel_regularizer=tf.keras.regularizers.l2(0.001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('elu'),
  tf.keras.layers.Dropout(0.1),

  tf.keras.layers.Dense(23, kernel_regularizer=tf.keras.regularizers.l2(0.01)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('elu'),
  tf.keras.layers.Dropout(0.1),

   tf.keras.layers.Dense(13, kernel_regularizer=tf.keras.regularizers.l2(0.1)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('elu'),
  tf.keras.layers.Dropout(0.1),

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

Results:

loss: 0.2217 - sparse_categorical_accuracy: 0.9570 - val_loss: 0.1504 - val_sparse_categorical_accuracy: 0.9758

Best result: round-33\50

loss: 0.2214 - sparse_categorical_accuracy: 0.9559 - val_loss: 0.1473 - val_sparse_categorical_accuracy: 0.9792

Experiment number 6:

```
layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),
  tf.keras.layers.Dense(95, kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('elu'),

  tf.keras.layers.Dense(28, kernel_regularizer=tf.keras.regularizers.l2(0.001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('elu'),

  tf.keras.layers.Dense(23, kernel_regularizer=tf.keras.regularizers.l2(0.01)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('elu'),

   tf.keras.layers.Dense(13, kernel_regularizer=tf.keras.regularizers.l2(0.1)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('elu'),

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

Results:

loss: 0.1358 - sparse_categorical_accuracy: 0.9755 - val_loss: 0.1434 - val_sparse_categorical_accuracy: 0.9749

Best result: round-27\50

loss: 0.1468 - sparse_categorical_accuracy: 0.9730 - val_loss: 0.1313 - val_sparse_categorical_accuracy: 0.9799

Important note: Based on the provided training log, there are some signs that might indicate potential overfitting, but it's not conclusive.

Experiment number 7:

```
layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),
  tf.keras.layers.Dense(95, kernel_regularizer=tf.keras.regularizers.l1(0.0001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('elu'),

  tf.keras.layers.Dense(28, kernel_regularizer=tf.keras.regularizers.l1(0.001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('elu'),

  tf.keras.layers.Dense(23, kernel_regularizer=tf.keras.regularizers.l1(0.01)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('elu'),

   tf.keras.layers.Dense(13, kernel_regularizer=tf.keras.regularizers.l1(0.1)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('elu'),

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

Results:

loss: 0.2903 - sparse_categorical_accuracy: 0.9624 - val_loss: 0.3258 - val_sparse_categorical_accuracy: 0.9497

Best result: round-20\50

loss: 0.3340 - sparse_categorical_accuracy: 0.9556 - val_loss: 0.2767 - val_sparse_categorical_accuracy: 0.9720

Important note: Based on the provided training log, there are some signs that might indicate potential overfitting, but it's not conclusive.

Experiment number 8:

```
layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),
  tf.keras.layers.Dense(95, kernel_regularizer=tf.keras.regularizers.l1(0.0001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),

  tf.keras.layers.Dense(28, kernel_regularizer=tf.keras.regularizers.l1(0.001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),

  tf.keras.layers.Dense(23, kernel_regularizer=tf.keras.regularizers.l1(0.01)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),

   tf.keras.layers.Dense(13, kernel_regularizer=tf.keras.regularizers.l1(0.1)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),

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

Results:

loss: 0.3009 - sparse_categorical_accuracy: 0.9598 - val_loss: 0.2859 - val_sparse_categorical_accuracy: 0.9655

Best result: round-17\50

loss: 0.3494 - sparse_categorical_accuracy: 0.9549 - val_loss: 0.2841 - val_sparse_categorical_accuracy: 0.9733

Important note: there are clear signs of overfitting in this training log

Experiment number 9:

```
layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),
  tf.keras.layers.Dense(95, kernel_regularizer=tf.keras.regularizers.l1(0.0001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),
  tf.keras.layers.Dropout(0.1),

  tf.keras.layers.Dense(28, kernel_regularizer=tf.keras.regularizers.l1(0.001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),
  tf.keras.layers.Dropout(0.2),

  tf.keras.layers.Dense(23, kernel_regularizer=tf.keras.regularizers.l1(0.01)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),
   tf.keras.layers.Dropout(0.3),

   tf.keras.layers.Dense(13, kernel_regularizer=tf.keras.regularizers.l1(0.1)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),
  tf.keras.layers.Dropout(0.4),

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

Results:

loss: 1.1378 - sparse_categorical_accuracy: 0.7684 - val_loss: 0.5632 - val_sparse_categorical_accuracy: 0.9631

Best result: round-21\50

loss: 1.1807 - sparse_categorical_accuracy: 0.7428 - val_loss: 0.5443 - val_sparse_categorical_accuracy: 0.9667

Experiment number 10:

```
layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),
  tf.keras.layers.Dense(95, kernel_regularizer=tf.keras.regularizers.l1(0.0001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),
  tf.keras.layers.Dropout(0.1),

  tf.keras.layers.Dense(28, kernel_regularizer=tf.keras.regularizers.l1(0.001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),
  tf.keras.layers.Dropout(0.2),

  tf.keras.layers.Dense(23, kernel_regularizer=tf.keras.regularizers.l1(0.01)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),
   tf.keras.layers.Dropout(0.3),

   tf.keras.layers.Dense(13, kernel_regularizer=tf.keras.regularizers.l1(0.1)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),
  tf.keras.layers.Dropout(0.4),

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

Results:

loss: 0.4316 - sparse_categorical_accuracy: 0.9041 - val_loss: 0.1985 - val_sparse_categorical_accuracy: 0.9696

Best result: round-34\50

loss: 0.4354 - sparse_categorical_accuracy: 0.9050 - val_loss: 0.1941 - val_sparse_categorical_accuracy: 0.9715

Experiment number 10:

```
layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),
  tf.keras.layers.Dense(95, kernel_regularizer=tf.keras.regularizers.l1(0.0001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),
  tf.keras.layers.Dropout(0.1),

  tf.keras.layers.Dense(28, kernel_regularizer=tf.keras.regularizers.l1(0.001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),
  tf.keras.layers.Dropout(0.1),

  tf.keras.layers.Dense(23, kernel_regularizer=tf.keras.regularizers.l1(0.01)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),
  tf.keras.layers.Dropout(0.1),

   tf.keras.layers.Dense(13, kernel_regularizer=tf.keras.regularizers.l1(0.1)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),
  tf.keras.layers.Dropout(0.1),

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

Results:

loss: 0.5172 - sparse_categorical_accuracy: 0.9320 - val_loss: 0.3872 - val_sparse_categorical_accuracy: 0.9703

Best result: round-34\50

loss: 0.5214 - sparse_categorical_accuracy: 0.9307 - val_loss: 0.3823 - val_sparse_categorical_accuracy: 0.9714

Experiment number 12:

```
  layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),
  tf.keras.layers.Dense(95,
kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.0001,l2=0.0001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),
  tf.keras.layers.Dropout(0.1),

  tf.keras.layers.Dense(28, kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.001,l2=0.001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),
  tf.keras.layers.Dropout(0.1),

  tf.keras.layers.Dense(23, kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.01,l2=0.01)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),
   tf.keras.layers.Dropout(0.1),

   tf.keras.layers.Dense(13, kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.1,l2=0.1)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),
  tf.keras.layers.Dropout(0.1),

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

Results:

loss: 0.5428 - sparse_categorical_accuracy: 0.9247 - val_loss: 0.3958 - val_sparse_categorical_accuracy: 0.9664

Best result: round-41\50

loss: 0.5473 - sparse_categorical_accuracy: 0.9231 - val_loss: 0.3959 - val_sparse_categorical_accuracy: 0.9686

Experiment number 13:

```
  layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),
  tf.keras.layers.Dense(95,
kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.0001,l2=0.0001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),
  tf.keras.layers.Dropout(0.1),

  tf.keras.layers.Dense(28, kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.001,l2=0.001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),
  tf.keras.layers.Dropout(0.1),

  tf.keras.layers.Dense(23, kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.01,l2=0.01)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),
   tf.keras.layers.Dropout(0.1),

   tf.keras.layers.Dense(13, kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.1,l2=0.1)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),
  tf.keras.layers.Dropout(0.1),

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

Results:

loss: 0.6388 - sparse_categorical_accuracy: 0.9086 - val_loss: 0.4129 - val_sparse_categorical_accuracy: 0.9715

Best result: round-45\50

loss: 0.6496 - sparse_categorical_accuracy: 0.9068 - val_loss: 0.4111 - val_sparse_categorical_accuracy: 0.9716

Experiment number 14:

```
layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),
  tf.keras.layers.Dense(95,
kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.0001,l2=0.0001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),
  tf.keras.layers.Dropout(0.1),

  tf.keras.layers.Dense(28, kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.001,l2=0.001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),
  tf.keras.layers.Dropout(0.2),

  tf.keras.layers.Dense(23, kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.01,l2=0.01)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),
   tf.keras.layers.Dropout(0.3),

   tf.keras.layers.Dense(13, kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.1,l2=0.1)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('relu'),
  tf.keras.layers.Dropout(0.4),

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

Results:

loss: 1.1331 - sparse_categorical_accuracy: 0.7764 - val_loss: 0.5858 - val_sparse_categorical_accuracy: 0.9568

Best result: round-41\50

loss: 0.5473 - sparse_categorical_accuracy: 0.9231 - val_loss: 0.3959 - val_sparse_categorical_accuracy: 0.9686

Experiment number 15:

```
layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),
  tf.keras.layers.Dense(95,
kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.0001,l2=0.0001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('elu'),
  tf.keras.layers.Dropout(0.1),

  tf.keras.layers.Dense(28, kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.001,l2=0.001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('elu'),
  tf.keras.layers.Dropout(0.1),

  tf.keras.layers.Dense(23, kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.01,l2=0.01)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('elu'),
   tf.keras.layers.Dropout(0.1),

   tf.keras.layers.Dense(13, kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.1,l2=0.1)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('elu'),
  tf.keras.layers.Dropout(0.1),

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

Results:

loss: 0.5037 - sparse_categorical_accuracy: 0.9330 - val_loss: 0.3854 - val_sparse_categorical_accuracy: 0.9661

Best result: round-21\50

loss: 0.5332 - sparse_categorical_accuracy: 0.9258 - val_loss: 0.3691 - val_sparse_categorical_accuracy: 0.9696

Experiment number 16:

```
layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),
  tf.keras.layers.Dense(95,
kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.0001,l2=0.0001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('sigmoid'),
  tf.keras.layers.Dropout(0.1),

  tf.keras.layers.Dense(28, kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.001,l2=0.001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('sigmoid'),
  tf.keras.layers.Dropout(0.1),

  tf.keras.layers.Dense(23, kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.01,l2=0.01)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('sigmoid'),
   tf.keras.layers.Dropout(0.1),

   tf.keras.layers.Dense(13, kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.1,l2=0.1)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('sigmoid'),
  tf.keras.layers.Dropout(0.1),

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

Results:

loss: 0.7263 - sparse_categorical_accuracy: 0.8911 - val_loss: 0.4754 - val_sparse_categorical_accuracy: 0.9634

Best result: round-41\50

loss: 0.7249 - sparse_categorical_accuracy: 0.8947 - val_loss: 0.4709 - val_sparse_categorical_accuracy: 0.9655

Experiment number 17:

```
layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),
  tf.keras.layers.Dense(95,
kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.0001,l2=0.0001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('sigmoid'),
  tf.keras.layers.Dropout(0.1),

  tf.keras.layers.Dense(28, kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.001,l2=0.001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('sigmoid'),
  tf.keras.layers.Dropout(0.2),

  tf.keras.layers.Dense(23, kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.01,l2=0.01)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('sigmoid'),
   tf.keras.layers.Dropout(0.3),

   tf.keras.layers.Dense(13, kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.1,l2=0.1)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('sigmoid'),
  tf.keras.layers.Dropout(0.4),

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

Results:

loss: 1.3551 - sparse_categorical_accuracy: 0.7300 - val_loss: 0.7309 - val_sparse_categorical_accuracy: 0.9346

Best result: round-45\50

loss: 1.3584 - sparse_categorical_accuracy: 0.7284 - val_loss: 0.7107 - val_sparse_categorical_accuracy: 0.9371

Experiment number 18:

```
layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),
  tf.keras.layers.Dense(95,
kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.0001,l2=0.0001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),


  tf.keras.layers.Dense(28, kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.001,l2=0.001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),


  tf.keras.layers.Dense(23, kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.01,l2=0.01)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),


   tf.keras.layers.Dense(13, kernel_regularizer=tf.keras.regularizers.L1L2(l1=0.1,l2=0.1)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),


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

Results:

loss: 0.3183 - sparse_categorical_accuracy: 0.9579 - val_loss: 0.3142 - val_sparse_categorical_accuracy: 0.9589

Best result: round-40\50

loss: 0.3228 - sparse_categorical_accuracy: 0.9601 - val_loss: 0.2918 - val_sparse_categorical_accuracy: 0.9706

important note: towards the end suggests that the model might be starting to overfit the training data.

Experiment number 19:

```
layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),
  tf.keras.layers.Dense(95, kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),
  tf.keras.layers.Dropout(0.2),

  tf.keras.layers.Dense(28, kernel_regularizer=tf.keras.regularizers.l2(0.001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),
  tf.keras.layers.Dropout(0.2),

  tf.keras.layers.Dense(23, kernel_regularizer=tf.keras.regularizers.l2(0.01)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),
   tf.keras.layers.Dropout(0.2),

   tf.keras.layers.Dense(13, kernel_regularizer=tf.keras.regularizers.l2(0.1)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),
  tf.keras.layers.Dropout(0.2),

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

Results:

loss: 0.3315 - sparse_categorical_accuracy: 0.9293 - val_loss: 0.1783 - val_sparse_categorical_accuracy: 0.9714

Best result: round-37\50

loss: 0.3456 - sparse_categorical_accuracy: 0.9259 - val_loss: 0.1753 - val_sparse_categorical_accuracy: 0.9720

Experiment number 20:

```
layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),
  tf.keras.layers.Dense(95, kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),
  tf.keras.layers.Dropout(0.1),

  tf.keras.layers.Dense(28, kernel_regularizer=tf.keras.regularizers.l2(0.001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),
  tf.keras.layers.Dropout(0.1),

  tf.keras.layers.Dense(23, kernel_regularizer=tf.keras.regularizers.l2(0.01)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),
   tf.keras.layers.Dropout(0.1),

   tf.keras.layers.Dense(13, kernel_regularizer=tf.keras.regularizers.l2(0.1)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),
  tf.keras.layers.Dropout(0.1),

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

Results:

loss: 0.2229 - sparse_categorical_accuracy: 0.9554 - val_loss: 0.1600 - val_sparse_categorical_accuracy: 0.9732

Best result: round-27\50

loss: 0.2356 - sparse_categorical_accuracy: 0.9506 - val_loss: 0.1517 - val_sparse_categorical_accuracy: 0.9774

Experiment number 21:

```
layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),
  tf.keras.layers.Dense(95, kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),
  tf.keras.layers.Dropout(0.1),

  tf.keras.layers.Dense(28, kernel_regularizer=tf.keras.regularizers.l2(0.001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),
  tf.keras.layers.Dropout(0.2),

  tf.keras.layers.Dense(23, kernel_regularizer=tf.keras.regularizers.l2(0.01)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),
   tf.keras.layers.Dropout(0.3),

   tf.keras.layers.Dense(13, kernel_regularizer=tf.keras.regularizers.l2(0.1)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),
  tf.keras.layers.Dropout(0.4),

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

Results:

loss: 0.4278 - sparse_categorical_accuracy: 0.9081 - val_loss: 0.2000 - val_sparse_categorical_accuracy: 0.9715

Best result: round-49\50

loss: 0.4379 - sparse_categorical_accuracy: 0.9067 - val_loss: 0.1954 - val_sparse_categorical_accuracy: 0.9739

Experiment number 22:

```
layers = [
  tf.keras.layers.Flatten(input_shape=image_shape),
  tf.keras.layers.Dense(95, kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),

  tf.keras.layers.Dense(28, kernel_regularizer=tf.keras.regularizers.l2(0.001)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),

  tf.keras.layers.Dense(23, kernel_regularizer=tf.keras.regularizers.l2(0.01)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),

   tf.keras.layers.Dense(13, kernel_regularizer=tf.keras.regularizers.l2(0.1)),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Activation('tanh'),

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

Results:

loss: 0.1413 - sparse_categorical_accuracy: 0.9744 - val_loss: 0.1560 - val_sparse_categorical_accuracy: 0.9723

Best result: round-49\50

loss: 0.1489 - sparse_categorical_accuracy: 0.9721 - val_loss: 0.1401 - val_sparse_categorical_accuracy: 0.9775

Important note: there are clear signs of overfitting in this training log

Summary: There was an experiment in which the accuracy of the test was better than the best experiment I chose, but it had overfitting, so that's what made it a not-so-good experiment.