```
network = models.Sequential()
network.add(Conv2D(filters=64,
                   kernel_size=(5,5),
                   padding='same',
                   input_shape=(28,28,3),
                   activation='relu'))
network.add(MaxPooling2D(pool_size=(2, 2)))
network.add(layers.Flatten(input_shape=(28,28,3)))
network.add(layers.Dense(256, activation='relu'))
network.add(layers.Dense(10, activation='softmax'))

network.compile(optimizer='rmsprop',
                loss='categorical_crossentropy',
                metrics=['accuracy'])

# KC: start to train the model
history = network.fit(  train_set,
                        validation_data   = validation_set,
                        epochs            = 10 )
```

Add an additional convolutional layer.

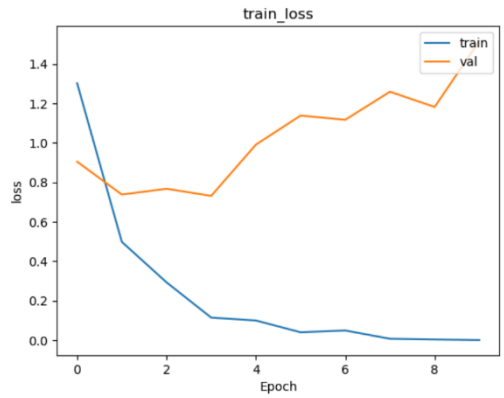
Add a pooling layer.

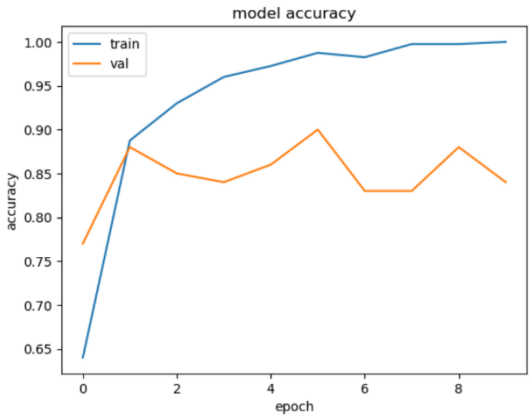
Adjust the number of neurons to 256 (originally 16).

Change the number of epochs to 10 (originally 5).

Accuracy: 81.0%

```
Accuracy for Testset: 0.8100000023841858
```





```
network = models.Sequential()
network.add(Conv2D(filters=64,
                   kernel_size=(5,5),
                   padding='same',
                   input_shape=(28,28,3),
                   activation='relu'))
#max pooling
network.add(MaxPooling2D(pool_size=(2, 2)))
network.add(Conv2D(filters=64,
                   kernel_size=(5,5),
                   padding='same',
                   input_shape=(28,28,3),
                   activation='relu'))
#max pooling
network.add(MaxPooling2D(pool_size=(2, 2)))
network.add(Conv2D(filters=64,
                   kernel_size=(5,5),
                   padding='same',
                   input_shape=(28,28,3),
                   activation='relu'))
#max pooling
network.add(MaxPooling2D(pool_size=(2, 2)))
network.add(layers.Flatten(input_shape=(28,28,3)))
network.add(layers.Dense(256, activation='relu'))
network.add(layers.Dense(256, activation='relu'))
network.add(layers.Dense(10, activation='softmax'))

network.compile(optimizer='rmsprop',
                loss='categorical_crossentropy',
                metrics=['accuracy'])

# KC: start to train the model
history = network.fit(  train_set,
                        validation_data  = validation_set,
                        epochs           = 10 )
```

Add an additional convolutional layer.

Add magnetization.

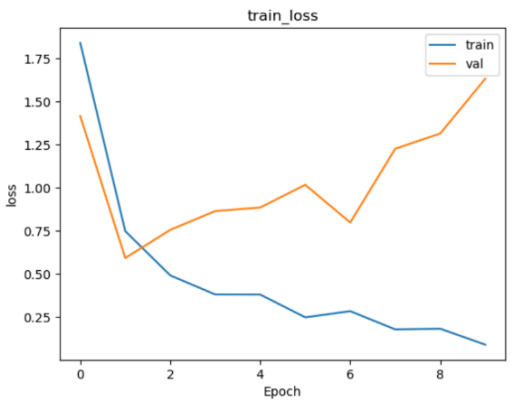
Add one more layer.

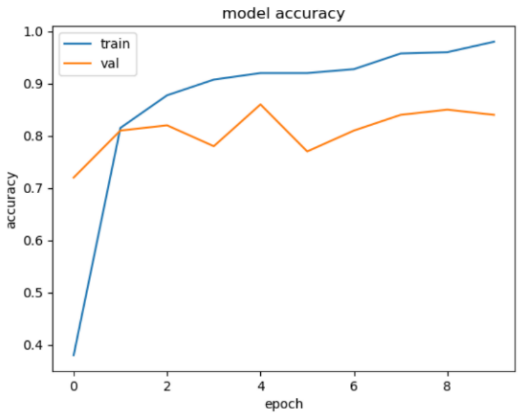
Adjust the number of neurons to 256 (originally 16).

Change the number of epochs to 10 (originally 5).

Accuracy: 81.0%

```
Accuracy for Testset: 0.8640000224113464
```



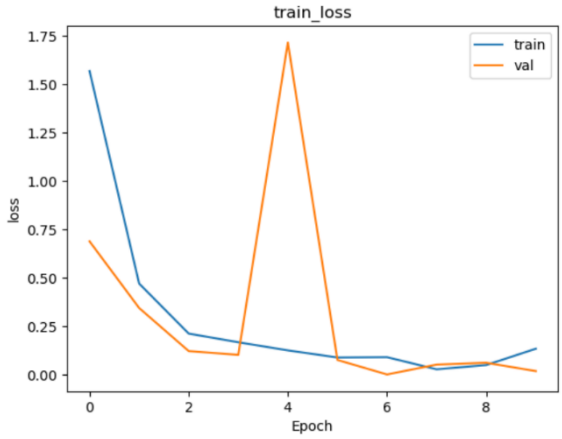


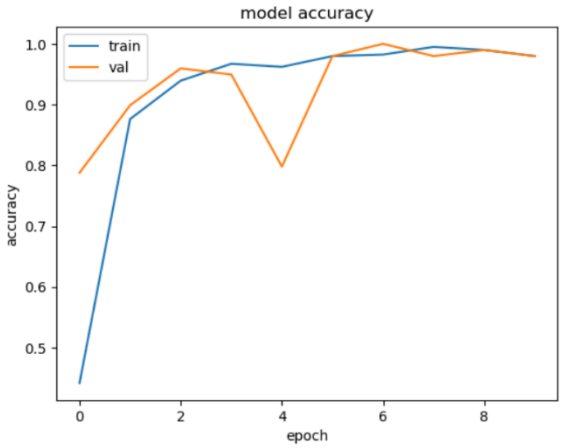
更改:

Perform data preprocessing (train data for image classification)"

Accuracy: 92.7%

```
Accuracy for Testset: 0.9279999732971191
```





```
image_datagen = ImageDataGenerator(rescale = 1./255, dtype='float32',
validation_split=0.2,rotation_range=10,fill_mode='nearest',brightness_range = [0.1, 0.2],channel_shift_range = 4)
```

```
#max pooling
network.add(MaxPooling2D(pool_size=(2, 2)))
network.add(layers.Flatten(input_shape=(28,28,3)))
network.add(layers.Dense(128, activation='relu'))
network.add(layers.Dense(256, activation='relu'))
network.add(layers.Dense(512, activation='relu'))
network.add(layers.Dense(512, activation='relu'))
network.add(layers.Dense(10, activation='softmax'))
```

更改:

Three convolutional layers.

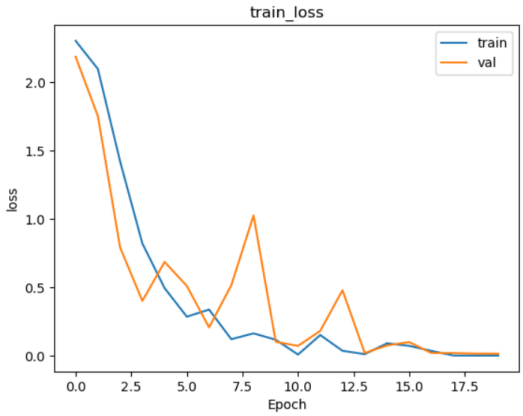
Add data augmentation.

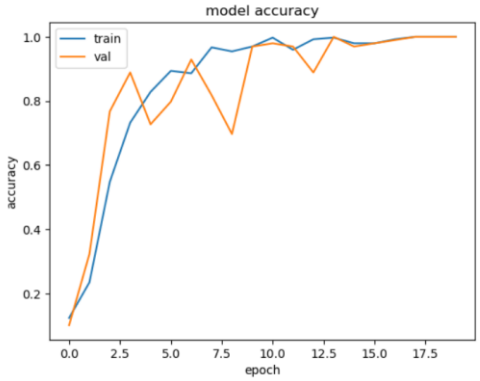
Increase to four layers.

Change the number of epochs to 20 (originally 15).

Accuracy: 92.9%

```
Accuracy for Testset: 0.9290000200271606
```

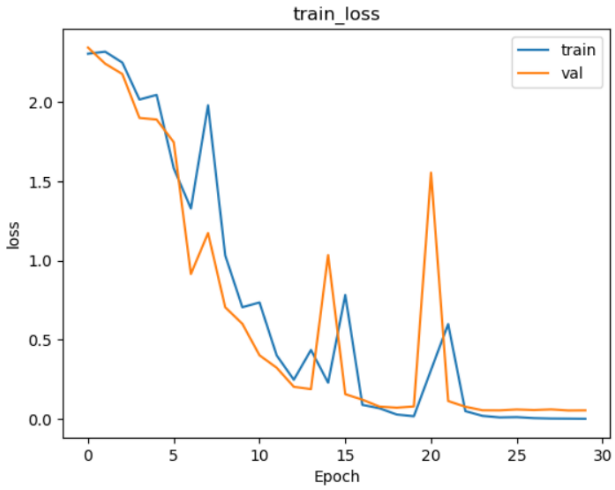




Add rotation_range=30

Accuracy: 90.89%

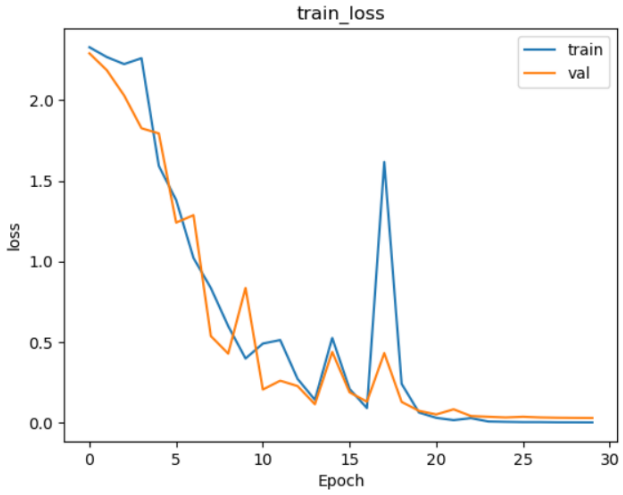
```
Accuracy for Testset: 0.9089999794960022
```



Add rotation_range=40

Accuracy: 91.5%

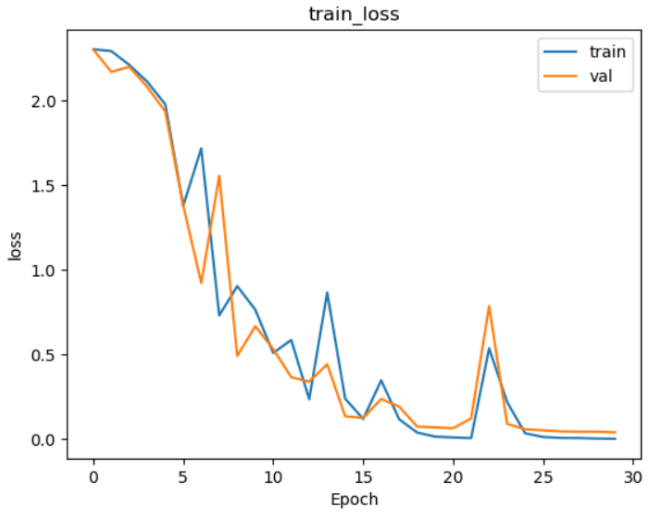
```
Accuracy for Testset: 0.9150000214576721
```



Add rotation_range=50

Accuracy: 91.5%

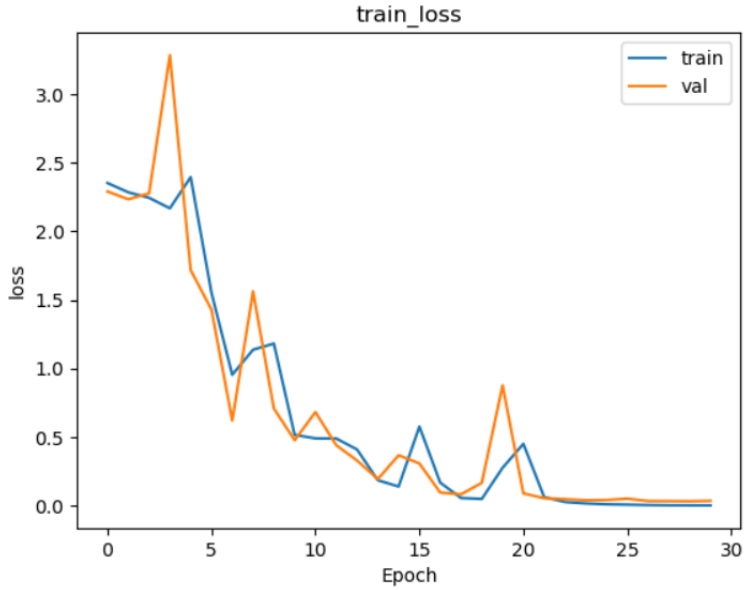
```
Accuracy for Testset: 0.9150000214576721
```



Add rotation_range=60

Accuracy: 91.6%

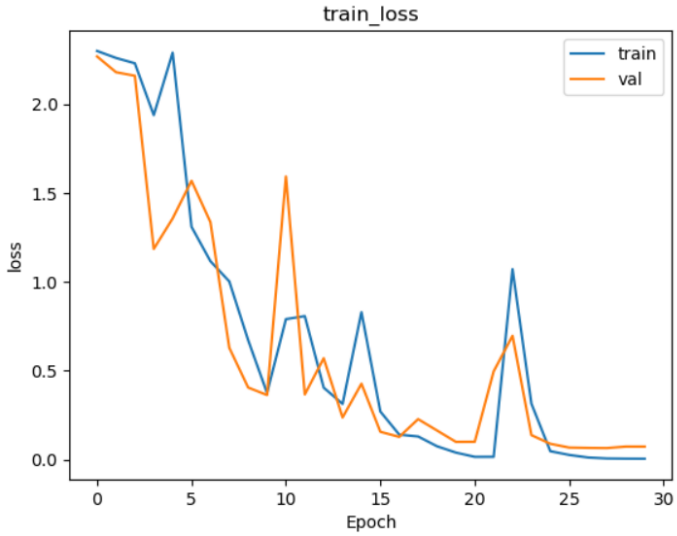
```
Accuracy for Testset: 0.9160000085830688
```



Add rotation_range=90

Accuracy: 91.9%

```
Accuracy for Testset: 0.9190000295639038
```



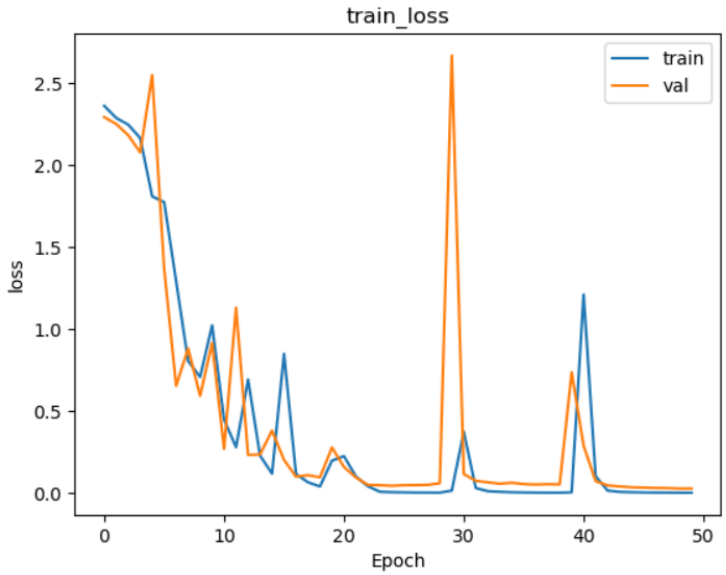
Add rotation_range=40

Maintain accuracy at 91%, and increase the number of epochs to 50.

Accuracy: 92.6%

```
# KC: start to train the model
history = network.fit(  train_set,
                        validation_data  = validation_set,
                        epochs           = 50 )
```

```
Accuracy for Testset: 0.9269999861717224
```



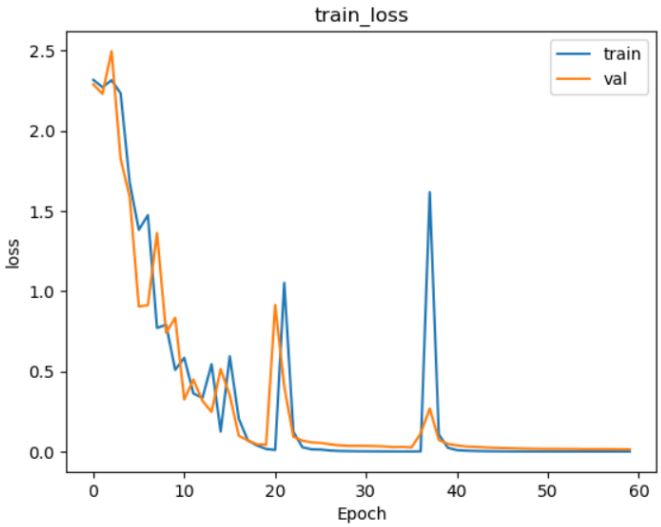
rotation_range=40

Maintain accuracy at 91%, increase the number of epochs to 60, and add brightness_range=[0.1,0.2]

Accuracy: 92.29%

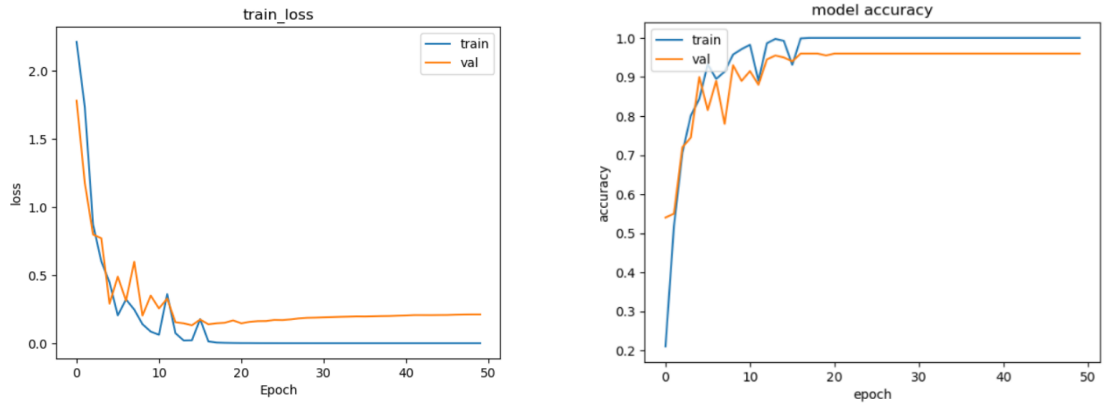
```
image_datagen = ImageDataGenerator(rescale = 1./255, dtype='float32',
validation_split=0.6 ,rotation_range=40,fill_mode='nerest',brightness_range=[0.1,0.2])
```

```
Accuracy for Testset: 0.9229999780654907
```



```
network = models.Sequential()
network.add(Conv2D(filters=64,
                   kernel_size=(5,5),
                   padding='same',
                   input_shape=(28,28,3),
                   activation='relu'))
#max pooling
network.add(MaxPooling2D(pool_size=(2, 2)))
network.add(Conv2D(filters=32,
                   kernel_size=(5,5),
                   padding='same',
                   input_shape=(28,28,3),
                   activation='relu'))

#network.add(AveragePooling2D(pool_size=(2, 2)))
# network.add(Conv2D(filters=32,
#                    kernel_size=(5,5),
#                    padding='same',
#                    input_shape=(28,28,3),
#                    activation='relu'))

#max pooling
network.add(MaxPooling2D(pool_size=(2, 2)))
network.add(layers.Flatten())
network.add(layers.Dense(128, activation='relu'))
network.add(layers.Dense(256, activation='relu'))
network.add(layers.Dense(256, activation='relu'))
network.add(layers.Dense(10, activation='softmax'))

network.compile(optimizer='rmsprop',
                loss='categorical_crossentropy',
                metrics=['accuracy'])


# KC: start to train the model
history = network.fit(  train_set,
                        validation_data  = validation_set,
                        epochs           = 50 )#60
```

```
Accuracy for Testset: 0.9515151381492615
```



Replace the training and testing datasets, and increase the number of testing samples.

Two convolutional layers.

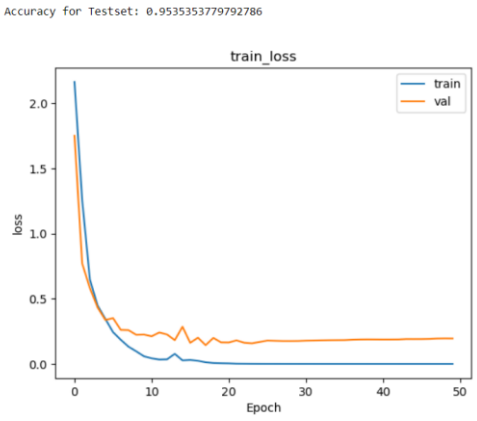
Filters set to 32 (originally set to 64).

Accuracy: 95.1%."

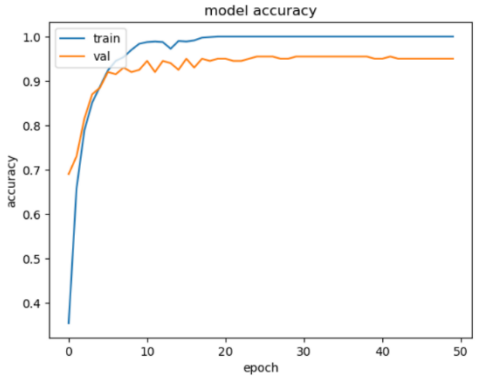
```
network = models.Sequential()
network.add(Conv2D(filters=64,
                   kernel_size=(5,5),
                   padding='same',
                   input_shape=(28,28,3),
                   activation='relu'))
#max pooling
network.add(MaxPooling2D(pool_size=(2, 2)))
network.add(Conv2D(filters=32,
                   kernel_size=(5,5),
                   padding='same',
                   input_shape=(28,28,3),
                   activation='relu'))

#network.add(AveragePooling2D(pool_size=(2, 2)))
# network.add(Conv2D(filters=32,
#                    kernel_size=(5,5),
#                    padding='same',
#                    input_shape=(28,28,3),
#                    activation='relu'))

#max pooling
network.add(MaxPooling2D(pool_size=(2, 2)))
network.add(layers.Flatten())
network.add(layers.Dense(128, activation='relu'))
network.add(layers.Dense(256, activation='relu'))
network.add(layers.Dense(256, activation='relu'))
network.add(layers.Dense(10, activation='softmax'))

network.compile(optimizer='adam',
                loss='categorical_crossentropy',
                metrics=['accuracy'])
```

Replace the training and testing datasets, and increase the number of testing samples.

Change the optimizer to Adam.



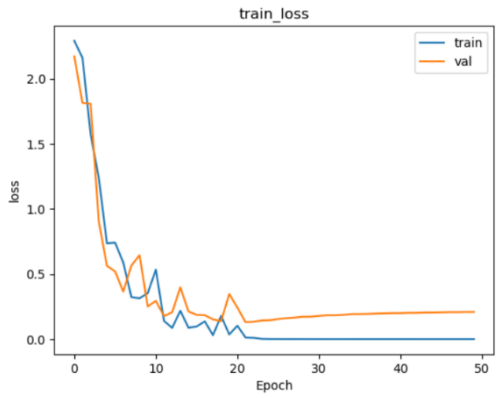
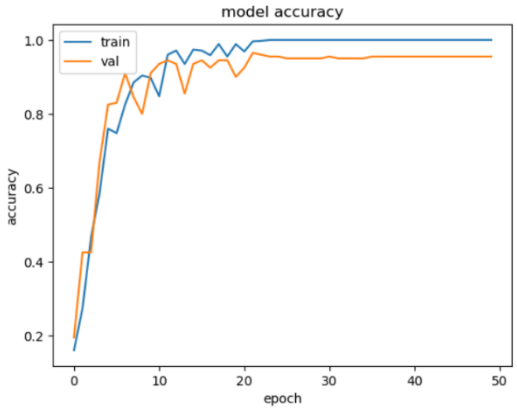
Accuracy: 95.3%."


model accuracy
train
val
accuracy
epoch
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
10
20
30
40
50

```
# KC: ImageDataGenerator can create an object used to generate batches of image data
image_datagen = ImageDataGenerator(rescale = 1./255, dtype='float32', validation_split=0.2)

# KC: The ImageDataGenerator class has the method flow_from_directory()
#       wihch can read the images from folders containing images.
train_set = image_datagen.flow_from_directory(  r'C:\Users\user\Desktop\final_project_dataset\final_project_dataset\Test',
                                                target_size = (28, 28),
                                                batch_size  = 128,
                                                class_mode  = 'categorical',
                                                subset='training'
                                                )

validation_set = image_datagen.flow_from_directory(  r'C:\Users\user\Desktop\final_project_dataset\final_project_dataset\Test',
                                                target_size = (28, 28),
                                                batch_size  = 128,
                                                class_mode  = 'categorical',
                                                subset='validation'
                                                )

test_set = image_datagen.flow_from_directory(r'C:\Users\user\Desktop\final_project_dataset\final_project_dataset\Train',
                                               target_size = (28, 28),
                                               batch_size  = 1,
                                               class_mode  = 'categorical',
                                              shuffle = True
                                               )
```

```
# ------------------------------------------------[ Densely con
network = models.Sequential()
network.add(Conv2D(filters=64,
                   kernel_size=(5,5),
                   padding='same',
                   input_shape=(28,28,3),
                   activation='relu'))
#max pooling
network.add(MaxPooling2D(pool_size=(2, 2)))
network.add(Conv2D(filters=64,
                   kernel_size=(5,5),
                   padding='same',
                   input_shape=(28,28,3),
                   activation='relu'))

network.add(AveragePooling2D(pool_size=(2, 2)))
network.add(Conv2D(filters=64,
                    kernel_size=(5,5),
                    padding='same',
                    input_shape=(28,28,3),
                    activation='relu'))

#max pooling
network.add(MaxPooling2D(pool_size=(2, 2)))
network.add(layers.Flatten())
network.add(layers.Dense(128, activation='relu'))
network.add(layers.Dense(256, activation='relu'))
network.add(layers.Dense(256, activation='relu'))
network.add(layers.Dense(10, activation='softmax'))

network.compile(optimizer='rmsprop',
                loss='categorical_crossentropy',
                metrics=['accuracy'])



# KC: start to train the model
history = network.fit(  train_set,
                        validation_data  = validation_set,
                        epochs           = 50 )#60
```

Replace the training and testing datasets, and increase the number of testing samples.

Accuracy: 95.7%."

```
Accuracy for Testset: 0.9575757384300232
```