In [1]:
```
import os
import cv2
import numpy as np
from skimage.feature import hog
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import OneHotEncoder
```

In [2]:
```
import tensorflow as tf
from tensorflow import keras
```

In [3]:
```
# Import required modules
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, MaxPooling2D, Flatten
```

In [4]:
```
from tensorflow.keras import layers
```

In [5]:
```
# Define the directories for the images
defective_dir = r"C:\Users\arivu\OneDrive\dataset_2\new_defect"
non_defective_dir = r"C:\Users\arivu\OneDrive\dataset_2\new_norfabric"
```

In [6]:
```
# Create empty lists for the images and labels
images = []
labels = []

#defective - 1
#non-defective - 0
```

In [7]:
```
# Load the defective images
for filename in os.listdir(defective_dir):
    img = cv2.imread(os.path.join(defective_dir, filename))
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    images.append(img)
    labels.append(1)
```

In [8]:
```
# Load the non-defective images
for filename in os.listdir(non_defective_dir):
    img = cv2.imread(os.path.join(non_defective_dir, filename))
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    images.append(img)
    labels.append(0)
```

In [9]:
```
# Convert the lists to numpy arrays
X = np.array(images)
y = np.array(labels)
```

In [10]:
```
# One-hot encode the labels
encoder = OneHotEncoder(sparse=False)
y = encoder.fit_transform(y.reshape(-1, 1))
```

In [11]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_st
```

In [12]:
```
resized_img = cv2.resize(X_train, (224, 224))
```

In [ ]:
```
```

In [35]:

```
# Define model
def define_model(input_shape, num_classes):
    model = Sequential([
        layers.Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Dropout(0.25),
        layers.Conv2D(64, kernel_size=(3, 3), activation='relu'),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Dropout(0.25),
        layers.Flatten(),
        layers.Dense(128, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation='softmax')
    ])
    return model
```

In [36]:

```
# Train model
def train_model(model, x_train, y_train, x_test, y_test, batch_size, epochs):
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['acc
    model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1, va
```

In [37]:

```
# Set the desired output shape
output_shape = (28, 28)

# Initialize an empty array for the resized images
resized_images = np.zeros((X_train.shape[0], *output_shape))

# Resize each image and store it in the resized_images array
for i, image in enumerate(X_train):
    resized_images[i] = cv2.resize(image, output_shape)

# Reshape the array to add the channel dimension
X_train_resized = resized_images.reshape((resized_images.shape[0], *output_shape,
```

In [38]:

```
X_train_resized.shape
```

Out[38]:

```
(886, 28, 28, 1)
```

In [39]:

```
# Set the desired output shape
output_shape = (28, 28)

# Initialize an empty array for the resized images
resized_images = np.zeros((X_test.shape[0], *output_shape))

# Resize each image and store it in the resized_images array
for i, image in enumerate(X_test):
    resized_images[i] = cv2.resize(image, output_shape)

# Reshape the array to add the channel dimension
X_test_resized = resized_images.reshape((resized_images.shape[0], *output_shape, 1
```

In [40]:

```
X_test_resized.shape
```

Out[40]:

```
(222, 28, 28, 1)
```

In [41]:

```
# Define input shape and number of classes
input_shape = (28, 28, 1)
num_classes = 10
```

In [42]:
```python
# Define model
model = define_model(input_shape, num_classes)
```

In [43]:
```python
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_2 (Conv2D)           (None, 26, 26, 32)        320       
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 13, 13, 32)       0         
 2D)                                                             
                                                                 
 dropout_3 (Dropout)         (None, 13, 13, 32)        0         
                                                                 
 conv2d_3 (Conv2D)           (None, 11, 11, 64)        18496     
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 5, 5, 64)         0         
 2D)                                                             
                                                                 
 dropout_4 (Dropout)         (None, 5, 5, 64)          0         
                                                                 
 flatten_1 (Flatten)         (None, 1600)              0         
                                                                 
 dense_2 (Dense)             (None, 128)               204928    
                                                                 
 dropout_5 (Dropout)         (None, 128)               0         
                                                                 
 dense_3 (Dense)             (None, 10)                1290      
                                                                 
=================================================================
Total params: 225,034
Trainable params: 225,034
Non-trainable params: 0
_________________________________________________________________
```

In [44]:
```python
import cv2
import numpy as np

# Resize images to (28, 28) shape
X_train_resized = np.zeros((X_train.shape[0], 28, 28))
for i in range(X_train.shape[0]):
    img = X_train[i]
    img_resized = cv2.resize(img, (28, 28), interpolation=cv2.INTER_AREA)
    X_train_resized[i] = img_resized.reshape((28, 28))
```

In [45]:
```python
from keras.utils import to_categorical

# Convert labels to one-hot encoded format
y_train_encoded = to_categorical(y_train[:, 0], num_classes=10)
y_test_encoded = to_categorical(y_test[:, 0], num_classes=10)
```

In [46]:
```python
# Train model
batch_size = 128
epochs = 70
train_model(model, X_train_resized, y_train_encoded, X_test_resized, y_test_encode
```

```
Epoch 1/70
7/7 [==============================] - 1s 114ms/step - loss: 26.7647 - accuracy:
0.4165 - val_loss: 6.7503 - val_accuracy: 0.4775
Epoch 2/70
7/7 [==============================] - 1s 74ms/step - loss: 7.0390 - accuracy: 0.5
372 - val_loss: 1.2366 - val_accuracy: 0.5991
Epoch 3/70
7/7 [==============================] - 1s 82ms/step - loss: 3.2137 - accuracy: 0.5
485 - val_loss: 1.1866 - val_accuracy: 0.6036
Epoch 4/70
7/7 [==============================] - 1s 73ms/step - loss: 1.4795 - accuracy: 0.5
801 - val_loss: 0.7132 - val_accuracy: 0.6081
Epoch 5/70
7/7 [==============================] - 0s 70ms/step - loss: 1.0086 - accuracy: 0.5
847 - val_loss: 0.6178 - val_accuracy: 0.6577
Epoch 6/70
7/7 [==============================] - 0s 73ms/step - loss: 0.8412 - accuracy: 0.5
993 - val_loss: 0.6835 - val_accuracy: 0.7387
Epoch 7/70
7/7 [==============================] - 0s 67ms/step - loss: 0.7275 - accuracy: 0.6
038 - val_loss: 0.8186 - val_accuracy: 0.8153
Epoch 8/70
7/7 [==============================] - 0s 70ms/step - loss: 0.7512 - accuracy: 0.5
914 - val_loss: 0.8470 - val_accuracy: 0.8468
Epoch 9/70
7/7 [==============================] - 1s 76ms/step - loss: 0.6764 - accuracy: 0.6
366 - val_loss: 0.8353 - val_accuracy: 0.8243
Epoch 10/70
7/7 [==============================] - 1s 81ms/step - loss: 0.6355 - accuracy: 0.6
332 - val_loss: 0.7650 - val_accuracy: 0.8333
Epoch 11/70
7/7 [==============================] - 1s 81ms/step - loss: 0.6055 - accuracy: 0.6
591 - val_loss: 0.6821 - val_accuracy: 0.8333
Epoch 12/70
7/7 [==============================] - 1s 84ms/step - loss: 0.5965 - accuracy: 0.6
456 - val_loss: 0.6558 - val_accuracy: 0.8153
Epoch 13/70
7/7 [==============================] - 1s 82ms/step - loss: 0.5820 - accuracy: 0.6
569 - val_loss: 0.6501 - val_accuracy: 0.8198
Epoch 14/70
7/7 [==============================] - 1s 80ms/step - loss: 0.5761 - accuracy: 0.6
682 - val_loss: 0.6404 - val_accuracy: 0.8243
Epoch 15/70
7/7 [==============================] - 1s 83ms/step - loss: 0.5887 - accuracy: 0.6
772 - val_loss: 0.5549 - val_accuracy: 0.8108
Epoch 16/70
7/7 [==============================] - 1s 97ms/step - loss: 0.5660 - accuracy: 0.6
964 - val_loss: 0.5815 - val_accuracy: 0.8153
Epoch 17/70
7/7 [==============================] - 1s 86ms/step - loss: 0.5480 - accuracy: 0.7
043 - val_loss: 0.4933 - val_accuracy: 0.8468
Epoch 18/70
7/7 [==============================] - 1s 78ms/step - loss: 0.5403 - accuracy: 0.7
099 - val_loss: 0.4431 - val_accuracy: 0.8829
Epoch 19/70
7/7 [==============================] - 1s 76ms/step - loss: 0.5414 - accuracy: 0.6
953 - val_loss: 0.4195 - val_accuracy: 0.8739
Epoch 20/70
7/7 [==============================] - 1s 88ms/step - loss: 0.5401 - accuracy: 0.7
099 - val_loss: 0.5083 - val_accuracy: 0.8649
Epoch 21/70
7/7 [==============================] - 1s 79ms/step - loss: 0.5052 - accuracy: 0.7
167 - val_loss: 0.4215 - val_accuracy: 0.8514
Epoch 22/70
```

```
7/7 [==============================] - 1s 82ms/step - loss: 0.5240 - accuracy: 0.7
122 - val_loss: 0.4650 - val_accuracy: 0.8333
Epoch 23/70
7/7 [==============================] - 1s 72ms/step - loss: 0.5155 - accuracy: 0.7
099 - val_loss: 0.3816 - val_accuracy: 0.8874
Epoch 24/70
7/7 [==============================] - 0s 71ms/step - loss: 0.4669 - accuracy: 0.7
540 - val_loss: 0.4341 - val_accuracy: 0.8288
Epoch 25/70
7/7 [==============================] - 1s 79ms/step - loss: 0.4667 - accuracy: 0.7
438 - val_loss: 0.3476 - val_accuracy: 0.8739
Epoch 26/70
7/7 [==============================] - 1s 84ms/step - loss: 0.4442 - accuracy: 0.7
585 - val_loss: 0.3834 - val_accuracy: 0.8468
Epoch 27/70
7/7 [==============================] - 1s 96ms/step - loss: 0.4444 - accuracy: 0.7
754 - val_loss: 0.3709 - val_accuracy: 0.8559
Epoch 28/70
7/7 [==============================] - 1s 106ms/step - loss: 0.4526 - accuracy: 0.
7980 - val_loss: 0.3608 - val_accuracy: 0.8784
Epoch 29/70
7/7 [==============================] - 1s 83ms/step - loss: 0.4134 - accuracy: 0.8
160 - val_loss: 0.3284 - val_accuracy: 0.8919
Epoch 30/70
7/7 [==============================] - 0s 70ms/step - loss: 0.3985 - accuracy: 0.8
059 - val_loss: 0.3569 - val_accuracy: 0.9009
Epoch 31/70
7/7 [==============================] - 0s 72ms/step - loss: 0.3580 - accuracy: 0.8
217 - val_loss: 0.3411 - val_accuracy: 0.8694
Epoch 32/70
7/7 [==============================] - 0s 71ms/step - loss: 0.3376 - accuracy: 0.8
476 - val_loss: 0.3265 - val_accuracy: 0.8694
Epoch 33/70
7/7 [==============================] - 1s 87ms/step - loss: 0.3250 - accuracy: 0.8
544 - val_loss: 0.3789 - val_accuracy: 0.8559
Epoch 34/70
7/7 [==============================] - 1s 121ms/step - loss: 0.3226 - accuracy: 0.
8409 - val_loss: 0.3705 - val_accuracy: 0.8829
Epoch 35/70
7/7 [==============================] - 1s 76ms/step - loss: 0.3764 - accuracy: 0.8
205 - val_loss: 0.4454 - val_accuracy: 0.8649
Epoch 36/70
7/7 [==============================] - 1s 76ms/step - loss: 0.3455 - accuracy: 0.8
375 - val_loss: 0.3358 - val_accuracy: 0.8874
Epoch 37/70
7/7 [==============================] - 1s 78ms/step - loss: 0.3084 - accuracy: 0.8
612 - val_loss: 0.3887 - val_accuracy: 0.9009
Epoch 38/70
7/7 [==============================] - 1s 88ms/step - loss: 0.2886 - accuracy: 0.8
849 - val_loss: 0.3937 - val_accuracy: 0.9009
Epoch 39/70
7/7 [==============================] - 1s 90ms/step - loss: 0.3192 - accuracy: 0.8
623 - val_loss: 0.3129 - val_accuracy: 0.9054
Epoch 40/70
7/7 [==============================] - 1s 83ms/step - loss: 0.3025 - accuracy: 0.8
747 - val_loss: 0.3395 - val_accuracy: 0.9099
Epoch 41/70
7/7 [==============================] - 1s 88ms/step - loss: 0.3831 - accuracy: 0.8
115 - val_loss: 0.3627 - val_accuracy: 0.8874
Epoch 42/70
7/7 [==============================] - 1s 80ms/step - loss: 0.3111 - accuracy: 0.8
521 - val_loss: 0.3479 - val_accuracy: 0.8964
Epoch 43/70
7/7 [==============================] - 1s 85ms/step - loss: 0.2815 - accuracy: 0.8
```

```
725 - val_loss: 0.4035 - val_accuracy: 0.9144
Epoch 44/70
7/7 [==============================] - 1s 91ms/step - loss: 0.2693 - accuracy: 0.8
679 - val_loss: 0.3749 - val_accuracy: 0.9009
Epoch 45/70
7/7 [==============================] - 1s 75ms/step - loss: 0.2513 - accuracy: 0.8
883 - val_loss: 0.3503 - val_accuracy: 0.9009
Epoch 46/70
7/7 [==============================] - 1s 74ms/step - loss: 0.2647 - accuracy: 0.8
939 - val_loss: 0.3446 - val_accuracy: 0.9189
Epoch 47/70
7/7 [==============================] - 0s 69ms/step - loss: 0.2580 - accuracy: 0.9
041 - val_loss: 0.3992 - val_accuracy: 0.9099
Epoch 48/70
7/7 [==============================] - 1s 73ms/step - loss: 0.2300 - accuracy: 0.9
074 - val_loss: 0.4540 - val_accuracy: 0.9054
Epoch 49/70
7/7 [==============================] - 0s 73ms/step - loss: 0.2144 - accuracy: 0.8
950 - val_loss: 0.3533 - val_accuracy: 0.8964
Epoch 50/70
7/7 [==============================] - 0s 72ms/step - loss: 0.2281 - accuracy: 0.9
041 - val_loss: 0.4217 - val_accuracy: 0.8739
Epoch 51/70
7/7 [==============================] - 1s 97ms/step - loss: 0.2128 - accuracy: 0.9
187 - val_loss: 0.4319 - val_accuracy: 0.8829
Epoch 52/70
7/7 [==============================] - 1s 86ms/step - loss: 0.1960 - accuracy: 0.9
278 - val_loss: 0.4239 - val_accuracy: 0.8874
Epoch 53/70
7/7 [==============================] - 1s 92ms/step - loss: 0.1917 - accuracy: 0.9
142 - val_loss: 0.4366 - val_accuracy: 0.8919
Epoch 54/70
7/7 [==============================] - 0s 71ms/step - loss: 0.1762 - accuracy: 0.9
312 - val_loss: 0.4903 - val_accuracy: 0.8739
Epoch 55/70
7/7 [==============================] - 0s 67ms/step - loss: 0.1837 - accuracy: 0.9
255 - val_loss: 0.3521 - val_accuracy: 0.9234
Epoch 56/70
7/7 [==============================] - 0s 64ms/step - loss: 0.1602 - accuracy: 0.9
323 - val_loss: 0.4244 - val_accuracy: 0.9144
Epoch 57/70
7/7 [==============================] - 1s 82ms/step - loss: 0.2016 - accuracy: 0.9
210 - val_loss: 0.3736 - val_accuracy: 0.9279
Epoch 58/70
7/7 [==============================] - 0s 68ms/step - loss: 0.1876 - accuracy: 0.9
233 - val_loss: 0.4530 - val_accuracy: 0.9234
Epoch 59/70
7/7 [==============================] - 0s 66ms/step - loss: 0.1929 - accuracy: 0.9
153 - val_loss: 0.3582 - val_accuracy: 0.9054
Epoch 60/70
7/7 [==============================] - 1s 76ms/step - loss: 0.2217 - accuracy: 0.8
939 - val_loss: 0.2777 - val_accuracy: 0.9414
Epoch 61/70
7/7 [==============================] - 1s 79ms/step - loss: 0.2166 - accuracy: 0.9
187 - val_loss: 0.4561 - val_accuracy: 0.9144
Epoch 62/70
7/7 [==============================] - 1s 91ms/step - loss: 0.1853 - accuracy: 0.9
300 - val_loss: 0.5694 - val_accuracy: 0.9054
Epoch 63/70
7/7 [==============================] - 1s 111ms/step - loss: 0.1717 - accuracy: 0.
9323 - val_loss: 0.3562 - val_accuracy: 0.9189
Epoch 64/70
7/7 [==============================] - 1s 88ms/step - loss: 0.2020 - accuracy: 0.9
142 - val_loss: 0.3221 - val_accuracy: 0.9369
```

```
Epoch 65/70
7/7 [==============================] - 1s 84ms/step - loss: 0.1900 - accuracy: 0.9
334 - val_loss: 0.3117 - val_accuracy: 0.9189
Epoch 66/70
7/7 [==============================] - 1s 85ms/step - loss: 0.1581 - accuracy: 0.9
357 - val_loss: 0.4545 - val_accuracy: 0.9279
Epoch 67/70
7/7 [==============================] - 1s 83ms/step - loss: 0.1747 - accuracy: 0.9
334 - val_loss: 0.4520 - val_accuracy: 0.9189
Epoch 68/70
7/7 [==============================] - 1s 74ms/step - loss: 0.1565 - accuracy: 0.9
436 - val_loss: 0.3085 - val_accuracy: 0.9279
Epoch 69/70
7/7 [==============================] - 1s 81ms/step - loss: 0.1461 - accuracy: 0.9
424 - val_loss: 0.3310 - val_accuracy: 0.9324
Epoch 70/70
7/7 [==============================] - 1s 77ms/step - loss: 0.1408 - accuracy: 0.9
458 - val_loss: 0.3564 - val_accuracy: 0.9324
```

In [47]:

```
history = model.fit(X_train_resized, y_train_encoded, validation_data=(X_test_resi
```

```
Epoch 1/70
28/28 [==============================] - 1s 26ms/step - loss: 0.1510 - accuracy:
0.9391 - val_loss: 0.3800 - val_accuracy: 0.9279
Epoch 2/70
28/28 [==============================] - 1s 23ms/step - loss: 0.1935 - accuracy:
0.9312 - val_loss: 0.3097 - val_accuracy: 0.9189
Epoch 3/70
28/28 [==============================] - 1s 24ms/step - loss: 0.2204 - accuracy:
0.9097 - val_loss: 0.3749 - val_accuracy: 0.8649
Epoch 4/70
28/28 [==============================] - 1s 32ms/step - loss: 0.2875 - accuracy:
0.8849 - val_loss: 0.3764 - val_accuracy: 0.9099
Epoch 5/70
28/28 [==============================] - 1s 28ms/step - loss: 0.1745 - accuracy:
0.9379 - val_loss: 0.2295 - val_accuracy: 0.9144
Epoch 6/70
28/28 [==============================] - 1s 23ms/step - loss: 0.1652 - accuracy:
0.9391 - val_loss: 0.2753 - val_accuracy: 0.9189
Epoch 7/70
28/28 [==============================] - 1s 31ms/step - loss: 0.1592 - accuracy:
0.9368 - val_loss: 0.3331 - val_accuracy: 0.9054
Epoch 8/70
28/28 [==============================] - 1s 36ms/step - loss: 0.1921 - accuracy:
0.9334 - val_loss: 0.2965 - val_accuracy: 0.9234
Epoch 9/70
28/28 [==============================] - 1s 30ms/step - loss: 0.1522 - accuracy:
0.9368 - val_loss: 0.3576 - val_accuracy: 0.9234
Epoch 10/70
28/28 [==============================] - 1s 33ms/step - loss: 0.1299 - accuracy:
0.9503 - val_loss: 0.2385 - val_accuracy: 0.9234
Epoch 11/70
28/28 [==============================] - 1s 37ms/step - loss: 0.1395 - accuracy:
0.9357 - val_loss: 0.2031 - val_accuracy: 0.9414
Epoch 12/70
28/28 [==============================] - 1s 36ms/step - loss: 0.1559 - accuracy:
0.9447 - val_loss: 0.2947 - val_accuracy: 0.9324
Epoch 13/70
28/28 [==============================] - 1s 32ms/step - loss: 0.1315 - accuracy:
0.9503 - val_loss: 0.1918 - val_accuracy: 0.9234
Epoch 14/70
28/28 [==============================] - 1s 32ms/step - loss: 0.0894 - accuracy:
0.9661 - val_loss: 0.2842 - val_accuracy: 0.9459
Epoch 15/70
28/28 [==============================] - 1s 34ms/step - loss: 0.1053 - accuracy:
0.9594 - val_loss: 0.1521 - val_accuracy: 0.9234
Epoch 16/70
28/28 [==============================] - 1s 27ms/step - loss: 0.0932 - accuracy:
0.9605 - val_loss: 0.5032 - val_accuracy: 0.8784
Epoch 17/70
28/28 [==============================] - 1s 24ms/step - loss: 0.1556 - accuracy:
0.9458 - val_loss: 0.1729 - val_accuracy: 0.9279
Epoch 18/70
28/28 [==============================] - 1s 28ms/step - loss: 0.1358 - accuracy:
0.9549 - val_loss: 0.2958 - val_accuracy: 0.9369
Epoch 19/70
28/28 [==============================] - 1s 40ms/step - loss: 0.0998 - accuracy:
0.9594 - val_loss: 0.2557 - val_accuracy: 0.9459
Epoch 20/70
28/28 [==============================] - 1s 32ms/step - loss: 0.1082 - accuracy:
0.9661 - val_loss: 0.4216 - val_accuracy: 0.9414
Epoch 21/70
28/28 [==============================] - 1s 33ms/step - loss: 0.1024 - accuracy:
0.9515 - val_loss: 0.3635 - val_accuracy: 0.9459
Epoch 22/70
```

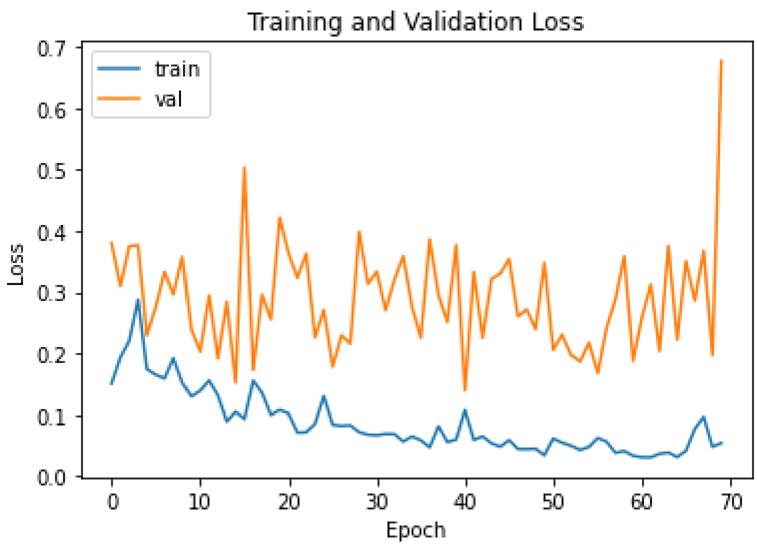
```
28/28 [==============================] - 1s 32ms/step - loss: 0.0714 - accuracy:
0.9752 - val_loss: 0.3233 - val_accuracy: 0.9459
Epoch 23/70
28/28 [==============================] - 1s 29ms/step - loss: 0.0719 - accuracy:
0.9740 - val_loss: 0.3630 - val_accuracy: 0.9414
Epoch 24/70
28/28 [==============================] - 1s 27ms/step - loss: 0.0854 - accuracy:
0.9729 - val_loss: 0.2262 - val_accuracy: 0.9144
Epoch 25/70
28/28 [==============================] - 1s 29ms/step - loss: 0.1306 - accuracy:
0.9515 - val_loss: 0.2710 - val_accuracy: 0.9369
Epoch 26/70
28/28 [==============================] - 1s 36ms/step - loss: 0.0841 - accuracy:
0.9707 - val_loss: 0.1785 - val_accuracy: 0.9414
Epoch 27/70
28/28 [==============================] - 1s 36ms/step - loss: 0.0823 - accuracy:
0.9661 - val_loss: 0.2294 - val_accuracy: 0.9414
Epoch 28/70
28/28 [==============================] - 1s 32ms/step - loss: 0.0832 - accuracy:
0.9707 - val_loss: 0.2160 - val_accuracy: 0.9459
Epoch 29/70
28/28 [==============================] - 1s 23ms/step - loss: 0.0722 - accuracy:
0.9752 - val_loss: 0.3984 - val_accuracy: 0.9459
Epoch 30/70
28/28 [==============================] - 1s 24ms/step - loss: 0.0680 - accuracy:
0.9797 - val_loss: 0.3134 - val_accuracy: 0.9459
Epoch 31/70
28/28 [==============================] - 1s 24ms/step - loss: 0.0671 - accuracy:
0.9752 - val_loss: 0.3335 - val_accuracy: 0.9414
Epoch 32/70
28/28 [==============================] - 1s 26ms/step - loss: 0.0692 - accuracy:
0.9707 - val_loss: 0.2704 - val_accuracy: 0.9279
Epoch 33/70
28/28 [==============================] - 1s 29ms/step - loss: 0.0690 - accuracy:
0.9729 - val_loss: 0.3201 - val_accuracy: 0.9324
Epoch 34/70
28/28 [==============================] - 1s 28ms/step - loss: 0.0569 - accuracy:
0.9763 - val_loss: 0.3591 - val_accuracy: 0.9459
Epoch 35/70
28/28 [==============================] - 1s 25ms/step - loss: 0.0651 - accuracy:
0.9763 - val_loss: 0.2757 - val_accuracy: 0.9414
Epoch 36/70
28/28 [==============================] - 1s 24ms/step - loss: 0.0588 - accuracy:
0.9797 - val_loss: 0.2257 - val_accuracy: 0.9414
Epoch 37/70
28/28 [==============================] - 1s 28ms/step - loss: 0.0474 - accuracy:
0.9763 - val_loss: 0.3863 - val_accuracy: 0.9459
Epoch 38/70
28/28 [==============================] - 1s 25ms/step - loss: 0.0817 - accuracy:
0.9695 - val_loss: 0.2948 - val_accuracy: 0.9369
Epoch 39/70
28/28 [==============================] - 1s 27ms/step - loss: 0.0562 - accuracy:
0.9740 - val_loss: 0.2513 - val_accuracy: 0.9459
Epoch 40/70
28/28 [==============================] - 1s 30ms/step - loss: 0.0599 - accuracy:
0.9729 - val_loss: 0.3769 - val_accuracy: 0.9234
Epoch 41/70
28/28 [==============================] - 1s 28ms/step - loss: 0.1080 - accuracy:
0.9661 - val_loss: 0.1397 - val_accuracy: 0.9414
Epoch 42/70
28/28 [==============================] - 1s 24ms/step - loss: 0.0594 - accuracy:
0.9774 - val_loss: 0.3331 - val_accuracy: 0.9369
Epoch 43/70
28/28 [==============================] - 1s 21ms/step - loss: 0.0652 - accuracy:
```

```
0.9819 - val_loss: 0.2255 - val_accuracy: 0.9324
Epoch 44/70
28/28 [==============================] - 1s 21ms/step - loss: 0.0540 - accuracy:
0.9808 - val_loss: 0.3215 - val_accuracy: 0.9414
Epoch 45/70
28/28 [==============================] - 1s 27ms/step - loss: 0.0484 - accuracy:
0.9786 - val_loss: 0.3302 - val_accuracy: 0.9414
Epoch 46/70
28/28 [==============================] - 1s 26ms/step - loss: 0.0592 - accuracy:
0.9729 - val_loss: 0.3540 - val_accuracy: 0.9414
Epoch 47/70
28/28 [==============================] - 1s 27ms/step - loss: 0.0447 - accuracy:
0.9842 - val_loss: 0.2603 - val_accuracy: 0.9414
Epoch 48/70
28/28 [==============================] - 1s 37ms/step - loss: 0.0444 - accuracy:
0.9808 - val_loss: 0.2716 - val_accuracy: 0.9414
Epoch 49/70
28/28 [==============================] - 1s 38ms/step - loss: 0.0452 - accuracy:
0.9786 - val_loss: 0.2395 - val_accuracy: 0.9414
Epoch 50/70
28/28 [==============================] - 1s 27ms/step - loss: 0.0344 - accuracy:
0.9831 - val_loss: 0.3480 - val_accuracy: 0.9369
Epoch 51/70
28/28 [==============================] - 1s 23ms/step - loss: 0.0618 - accuracy:
0.9786 - val_loss: 0.2056 - val_accuracy: 0.9505
Epoch 52/70
28/28 [==============================] - 1s 24ms/step - loss: 0.0550 - accuracy:
0.9797 - val_loss: 0.2306 - val_accuracy: 0.9459
Epoch 53/70
28/28 [==============================] - 1s 30ms/step - loss: 0.0501 - accuracy:
0.9819 - val_loss: 0.1976 - val_accuracy: 0.9414
Epoch 54/70
28/28 [==============================] - 1s 27ms/step - loss: 0.0430 - accuracy:
0.9865 - val_loss: 0.1867 - val_accuracy: 0.9414
Epoch 55/70
28/28 [==============================] - 1s 30ms/step - loss: 0.0483 - accuracy:
0.9786 - val_loss: 0.2175 - val_accuracy: 0.9414
Epoch 56/70
28/28 [==============================] - 1s 29ms/step - loss: 0.0626 - accuracy:
0.9763 - val_loss: 0.1679 - val_accuracy: 0.9369
Epoch 57/70
28/28 [==============================] - 1s 27ms/step - loss: 0.0565 - accuracy:
0.9797 - val_loss: 0.2420 - val_accuracy: 0.9459
Epoch 58/70
28/28 [==============================] - 1s 30ms/step - loss: 0.0385 - accuracy:
0.9831 - val_loss: 0.2882 - val_accuracy: 0.9459
Epoch 59/70
28/28 [==============================] - 1s 26ms/step - loss: 0.0413 - accuracy:
0.9842 - val_loss: 0.3585 - val_accuracy: 0.9459
Epoch 60/70
28/28 [==============================] - 1s 24ms/step - loss: 0.0337 - accuracy:
0.9842 - val_loss: 0.1879 - val_accuracy: 0.9414
Epoch 61/70
28/28 [==============================] - 1s 21ms/step - loss: 0.0312 - accuracy:
0.9876 - val_loss: 0.2584 - val_accuracy: 0.9414
Epoch 62/70
28/28 [==============================] - 1s 21ms/step - loss: 0.0311 - accuracy:
0.9876 - val_loss: 0.3126 - val_accuracy: 0.9324
Epoch 63/70
28/28 [==============================] - 1s 24ms/step - loss: 0.0370 - accuracy:
0.9853 - val_loss: 0.2037 - val_accuracy: 0.9505
Epoch 64/70
28/28 [==============================] - 1s 30ms/step - loss: 0.0389 - accuracy:
0.9808 - val_loss: 0.3752 - val_accuracy: 0.9459
```

```
Epoch 65/70
28/28 [==============================] - 1s 32ms/step - loss: 0.0316 - accuracy:
0.9865 - val_loss: 0.2226 - val_accuracy: 0.9459
Epoch 66/70
28/28 [==============================] - 1s 36ms/step - loss: 0.0418 - accuracy:
0.9865 - val_loss: 0.3508 - val_accuracy: 0.9369
Epoch 67/70
28/28 [==============================] - 1s 24ms/step - loss: 0.0775 - accuracy:
0.9786 - val_loss: 0.2861 - val_accuracy: 0.9459
Epoch 68/70
28/28 [==============================] - 1s 21ms/step - loss: 0.0970 - accuracy:
0.9673 - val_loss: 0.3675 - val_accuracy: 0.9324
Epoch 69/70
28/28 [==============================] - 1s 21ms/step - loss: 0.0483 - accuracy:
0.9808 - val_loss: 0.1974 - val_accuracy: 0.9414
Epoch 70/70
28/28 [==============================] - 1s 23ms/step - loss: 0.0544 - accuracy:
0.9865 - val_loss: 0.6766 - val_accuracy: 0.9369
```

In [54]:
```python
import matplotlib.pyplot as plt

# Plot the training and validation loss
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='val')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



In [55]:
```python
# Plot the training and validation accuracy
plt.plot(history.history['accuracy'], label='train')
plt.plot(history.history['val_accuracy'], label='val')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

In [56]:
```
# Predict classes for test set
y_pred = model.predict(X_test_resized)
```

```
7/7 [==============================] - 0s 7ms/step
```

In [57]:
```
# Get predicted class for each sample
y_pred_class = np.argmax(y_pred, axis=1)
```

In [58]:
```
# Evaluate model performance
score = model.evaluate(X_test_resized, y_test_encoded, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.6766079068183899
Test accuracy: 0.9369369149208069
```

Save model

In [59]:
```
from keras.models import load_model

model.save('my_model.h5')  # creates a HDF5 file 'my_model.h5'
```

In [60]:
```
import numpy as np
from keras.preprocessing import image
from keras.applications.vgg16 import VGG16, preprocess_input
```

In [71]:
```
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

model = Sequential([
    Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(224, 224, 3)),
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(10, activation='softmax')
])
```

In [77]:
```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import numpy as np

# Load the trained model
model = tf.keras.models.load_model(r'C:\Users\arivu\my_model.h5')
```

```python
# Load and preprocess the input image
img_path = r"C:\Users\arivu\OneDrive\dataset_2\new_defect\20180531_135032(1).jpg"

img = cv2.imread(img_path)

# Resize image to (28, 28)
img = cv2.resize(img, (28, 28))

# Convert to grayscale
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Normalize pixel values
img = img / 255.0

# Reshape to add the channel dimension
img = img.reshape((1, 28, 28, 1))


# Make prediction
prediction = model.predict(img)
if np.any(prediction[0]) < 0.5:
    print("No defect detected")
else:
    print("Defect detected")
```

```
WARNING:tensorflow:6 out of the last 19 calls to <function Model.make_predict_func
tion.<locals>.predict_function at 0x000001E7C9B0C0D0> triggered tf.function retrac
ing. Tracing is expensive and the excessive number of tracings could be due to (1)
creating @tf.function repeatedly in a loop, (2) passing tensors with different sha
pes, (3) passing Python objects instead of tensors. For (1), please define your @t
f.function outside of the loop. For (2), @tf.function has reduce_retracing=True op
tion that can avoid unnecessary retracing. For (3), please refer to https://www.te
nsorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/a
pi_docs/python/tf/function for  more details.
1/1 [==============================] - 0s 47ms/step
Defect detected
```

In [39]:

```python
from PIL import Image
import os

# Set the desired size
size = (224, 224)

# Set the path to the folder containing the images
path = r"C:\Users\arivu\OneDrive\dataset_2\defected"

# Set the output path for the resized images
output_path = r"C:\Users\arivu\OneDrive\dataset_2\new_defect"

# Loop through each image in the folder and resize it
for filename in os.listdir(path):
    img_path = os.path.join(path, filename)
    output_img_path = os.path.join(output_path, filename)
    with Image.open(img_path) as img:
        img = img.resize(size)
        img.save(output_img_path)
```

In [ ]:

```python

```

In [40]:

```python
size = (224,224)

# Set the path to the folder containing the images
path = r"C:\Users\arivu\OneDrive\dataset_2\normalfabric"
```

```
# Set the output path for the resized images
output_path = r"C:\Users\arivu\OneDrive\dataset_2\new_norfabric"

# Loop through each image in the folder and resize it
for filename in os.listdir(path):
    img_path = os.path.join(path, filename)
    output_img_path = os.path.join(output_path, filename)
    with Image.open(img_path) as img:
        img = img.resize(size)
        img.save(output_img_path)
```

In [ ]:

In [ ]:

In [ ]: